Efficient Exploration of the Rashomon Set of Rule-Set Models

Martino Ciaperoni[†]
martino.ciaperoni@aalto.fi
Aalto University
Espoo, Finland

Han Xiao[†]
xiaohan2012@gmail.com
The Upright Project
Helsinki, Finland

Aristides Gionis
argioni@kth.se
KTH Royal Institute of Technology
Stockholm, Sweden

ABSTRACT

Today, as increasingly complex predictive models are developed, simple rule sets remain a crucial tool to obtain interpretable predictions and drive high-stakes decision making. However, a single rule set provides a partial representation of a learning task. An emerging paradigm in interpretable machine learning aims at exploring the *Rashomon set* of all models exhibiting near-optimal performance. Existing work on Rashomon-set exploration focuses on exhaustive search of the Rashomon set for particular classes of models, which can be a computationally challenging task. On the other hand, exhaustive enumeration leads to redundancy that often is not necessary, and a representative sample or an estimate of the size of the Rashomon set is sufficient for many applications. In this work, we propose, for the first time, efficient methods to explore the Rashomon set of rule-set models with or without exhaustive search. Extensive experiments demonstrate the effectiveness of the proposed methods in a variety of scenarios.

CCS CONCEPTS

• **Computing methodologies** → **Supervised learning**; **Rule learning**.

KEYWORDS

Interpretable machine learning, Rashomon set, Rule-based classification, Scalable algorithms

ACM Reference Format:
Martino Ciaperoni[†], Han Xiao[†], and Aristides Gionis. 2024. Efficient Exploration of the Rashomon Set of Rule-Set Models. In *Proceedings of the 30th ACM SIGKDD Conference on Knowledge Discovery and Data Mining (KDD '24), August 25–29, 2024, Barcelona, Spain.* ACM, New York, NY, USA, 12 pages. https://doi.org/10.1145/3637528.3671818

1 INTRODUCTION

Following the impressive results achieved by modern machine-learning methods, automated decision making is used in consequential domains, such as health care, education, and criminal justice. However, many state-of-the-art models are opaque, and as such, they are difficult to interpret, understand, and trust. In other cases, they may hide harmful biases [34]. Thus, the research community has become increasingly aware of the importance of *inherently-interpretable machine-learning algorithms*, and there is a pressing need for models that can be understood and trusted by humans.

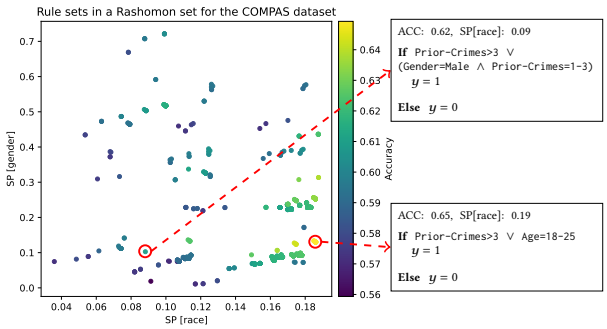

Figure 1: A Rashomon set of rule sets in the COMPAS dataset. For each rule set, we show accuracy (colour) and statistical parity (SP) [9] on race (x-axis) and gender (y-axis). Two example rule sets with similar accuracy, but highly different statistical parity on race, are additionally presented.

Logical models, based on "`if-then`" rules, are intrinsically interpretable models for predictive tasks. Among popular logical models, in this work we focus on *rule sets*, which are particularly easy to interpret [25]. Extension to more structured logical models, such as *rule lists* or *decision trees*, is left to future work.

Another significant aspect of interpretable machine learning is that, often, a single model does not offer an adequate representation of reality since there is a *large set of models with near-optimal predictive performance*. In the literature, such a set is referred to as *Rashomon set*. Rashomon sets have been shown to have applications in multiple domains, including credit-score estimation, natural-language processing, health-record analysis, recidivism prediction, and more [23, 34, 37, 43]. Considering the entire Rashomon set rather than a single model provides a wealth of actionable information. For instance, computing the proportion of models belonging to the Rashomon set allows us to characterize the complexity of a learning task [37]. Additionally, Rashomon sets allow us to study important properties of machine-learning models, such as fairness [15] and feature importance [43]. As a concrete example, Figure 1 depicts a Rashomon set of rule sets for the COMPAS dataset used for recidivism prediction. Although the rule sets in the Rashomon set have similar accuracy scores (ranging from 0.56 to 0.65), two important measures of fairness vary significantly.

Due to the combinatorial explosion of the search space, exhaustive enumeration or storage of the rule sets in the Rashomon set poses significant computational challenges, and may not always be feasible. In this paper, we propose, for the first time, methods to efficiently explore the Rashomon set with or without exhaustive

[†]Both authors contributed equally to this work.

KDD '24, August 25–29, 2024, Barcelona, Spain

ACM ISBN 979-8-4007-0490-1/24/08.
https://doi.org/10.1145/3637528.3671818

enumeration. As demonstrated in Section 7, the proposed methods accurately reveal the complexity inherent in tackling a learning task based on rule sets, as well as other key properties of rule sets including feature importance and fairness.

All the methods we propose rely on a highly-optimized branch-and-bound algorithm for exhaustive enumeration of the rule sets in the Rashomon set. To scale up, the branch-and-bound algorithm leverages (*i*) pruning bounds that effectively restrict the search space, and (*ii*) incremental computation to re-use previously computed results. Building on our branch-and-bound algorithm for exhaustive enumeration, we introduce two alternative approaches for non-exhaustive exploration of the Rashomon set by generating representative samples and estimating its size. The first approach partitions the solution space into random cells and enumerates the solutions in one randomly selected cell. The second approach instead simply visits subsets of the search space and constructs samples during the process. The samples generated by both approaches are supported by guarantees of near uniformity.

In summary, we make the following contributions.

- We formally describe exact and approximate variants of the problems of exhaustive and non-exhaustive enumeration of rule sets in the Rashomon set.
- We propose a branch-and-bound algorithm, named BBenum, for efficient exhaustive enumeration.
- As BBenum may incur high cost, we develop ApproxSample and ApproxCount, two highly-optimized algorithms with strong quality guarantees, which allow for non-exhaustive exploration of the Rashomon set by approximate uniform sampling and estimation of the size of the Rashomon set.
- We additionally devise BBsts, a faster, but generally less accurate alternative to ApproxSample and ApproxCount.
- We evaluate the proposed algorithms in a thorough experimental evaluation and through cases studies.

The rest of this paper is organized as follows. Section 2 discusses related work. Section 3 introduces our notation and problem formulations. Section 4 presents the proposed method for exhaustive enumeration of the Rashomon set, while Sections 5 and 6 describe the proposed methods for non-exhaustive exploration of the Rashomon set. Section 7 presents our experimental evaluation, and finally, Section 8 is a short conclusion.

2 RELATED WORK

Interpretable machine learning. The study of interpretable models to address machine-learning tasks is a fast-growing field. The topic is related to *explainable machine learning* [6], which aims at explaining the predictions of opaque models [8]. However, there is evidence that explaining opaque algorithms may provide misleading and even false characterizations [26, 33]. Therefore, there is a need for novel inherently interpretable models.

Optimal logical models. Logical models (including *rule sets*, *rule lists*, and *decision trees*) are prominent examples of interpretable models that have been successfully used in a variety of applications [34, 40, 45]. Over the years, due to the complexity inherent in the optimization, approximate algorithms and heuristic approaches have been employed to find logical models with good prediction performance. Recent advances in computing power and algorithmic techniques, however, motivate the search for *globally optimal* models for different classes of logical models. For finding optimal rule lists [5] and decision trees [22], ad hoc branch-and-bound algorithms have been proposed, while existing work on finding optimal rule sets relies on off-the-shelf SAT solvers [41] or integer programming solvers [28].

The Rashomon set. In recent years, research in interpretable machine learning has emphasized the importance of going beyond a single model. The *Rashomon effect* [7] expresses the idea that a real-world phenomenon can be explained equally well by multiple models. Such a set of models is referred to as the *Rashomon set* [34], and finds a number of interesting applications, such as measuring the complexity of a learning task [36], analyzing feature importance [18, 19] and investigating fairness in machine learning [29].

Recently, work has been carried out to develop techniques to exhaustively enumerate the Rashomon set for particular classes of models, including decision lists [30] and decision trees [43].

In a similar vein, Hara and Ishihata [21] consider approximate and exact enumeration of rule sets and lists sorted by an objective value, representing the quality of the model. Although this enumeration problem is similar to the problem studied in this work, there are crucial differences. Hara and Ishihata [21] consider a simplistic formulation of the rule-set learning problem, which neglects false positives. Further, the methods they propose hinge on particular assumptions and the output rule models are required to be sorted by their quality. In view of the mentioned differences, our proposed methods and those by Hara and Ishihata [21] are not directly comparable. Additionally, the methods of Hara and Ishihata [21] are not competitive with ours, in terms of runtime. For instance, for the Compas dataset, we observe that the time required by our methods to enumerate 50 rule sets is on par with the time required by the methods of Hara and Ishihata [21] to find a single rule set.

Rule sets can be regarded as less constrained extensions of decision lists and trees. The problem of enumerating the Rashomon set for rule sets is more challenging since the additional structure imposed by decision lists and trees allows for pruning additional portions of the search space. This computational challenge calls for novel ideas: we can effectively explore the Rashomon set for rule sets without exhaustive enumeration. Non-exhaustive enumeration, which is the main focus of this work, is a largely unexplored topic.

Constrained counting and sampling. *Constrained (or model) sampling and counting* is a fundamental problem in artificial intelligence involving sampling and counting the satisfying assignments of a propositional formula. The problem is known to be computationally hard [39]. Thus, approximate solutions have been investigated. Chakraborty et al. [11, 12] leverage hash functions to randomly partition the space of possible models into small cells, and satisfying assignments are sampled via calls to SAT solvers. In this work, we leverage this idea to design efficient sampling and counting algorithms that do not require exhaustive enumeration. Ermon et al. [17] propose an alternative approach for approximate model sampling. The algorithm leverages a SAT solver whilst enforcing a uniform exploration of the search space. We also build on this idea to design alternative efficient algorithms for sampling and counting without the need of exhaustive enumeration.

3 PROBLEM FORMULATION

We use boldface uppercase letters to denote matrices, e.g., $\boldsymbol{A}$, and boldface lowercase letters to denote vectors, e.g., $\boldsymbol{x}$ and $\boldsymbol{b}$. For a matrix $\boldsymbol{A}$, we use $\boldsymbol{A}_i$ to denote its i-th row, $\boldsymbol{A}_{:i}$ to denote its first i rows, and $\boldsymbol{A}_{i,j}$ to denote the j-th element of $\boldsymbol{A}_i$. Similarly, for a vector $\boldsymbol{b}$, we use $\boldsymbol{b}_i$ and $\boldsymbol{b}_{:i}$ to denote the i-th element and the first i elements of $\boldsymbol{b}$, respectively. Given a positive integer M and a sequence of positive integers S with values in the set $\{1, \ldots, M\}$, we use $\mathbf{1}_S \in \{0,1\}^M$ to denote the indicator vector of S, i.e., $\mathbf{1}_{S,i} = 1$ if $i \in S$ and $\mathbf{1}_{S,i} = 0$ otherwise.

3.1 Preliminaries

We restrict our setting to binary classification with binary-valued features. More general settings can be mapped to the setting we study via preprocessing, although the performance of the resulting methods will depend on the preprocessing methodology. Extending our methods to more general settings is left for future work.

We denote the training data as $\mathcal{D} = [(\boldsymbol{x}_n, y_n)]_{n=1}^N$, where $\boldsymbol{x}_n \in \{0,1\}^J$ are binary features and $y_n \in \{0,1\}$ is the label. Let $\boldsymbol{x}_{n,j}$ denote the value of the j-th feature of the observation vector $\boldsymbol{x}_n$.

A rule set $S = (r_1, \ldots, r_L)$ of size L consists of L distinct decision rules. A decision rule (or simply, rule) $r = p \rightarrow q$ is a logical implication "if p then q". An antecedent p is a clause consisting of a conjunction of features. For data point $\boldsymbol{x}_n$, antecedent p evaluates to *true* if all features of p have value 1, i.e., $\boldsymbol{x}_{n,j} = 1$ for all features j in p, and it evaluates to *false* otherwise. A consequent q is the predicted label. For instance, the rule $(\boldsymbol{x}_{n,2} = 1) \wedge (\boldsymbol{x}_{n,5} = 1) \rightarrow y_n = 1$ predicts $y_n = 1$ for any data point $\boldsymbol{x}_n$ with $\boldsymbol{x}_{n,2} = 1$ and $\boldsymbol{x}_{n,5} = 1$.

We say that a rule $r = p \rightarrow q$ *captures* a data point $\boldsymbol{x}_n$, written as $cap(\boldsymbol{x}_n, r) = 1$, if p evaluates $\boldsymbol{x}_n$ to true. We say that the rule set S *captures* $\boldsymbol{x}_n$, written as $cap(\boldsymbol{x}_n, S) = 1$, if at least one rule in S captures $\boldsymbol{x}_n$. If $\boldsymbol{x}_n$ is not captured by any rule in S, we write $cap(\boldsymbol{x}_n, S) = 0$. As it is common [13, 42], to prioritize interpretability, we consider rule sets consisting of positive rules only, i.e., $q = (y_n = 1)$.[1] In other words, if $cap(\boldsymbol{x}_n, S) = 1$, then the prediction is $y_n = 1$, while if $cap(\boldsymbol{x}_n, S) = 0$, the prediction is $y_n = 0$.

We assume that a set of candidate decision rules $\mathcal{U} = \{r_1, \ldots, r_M\}$ is provided.[2] We further assume that the rules in $\mathcal{U}$ are ordered, e.g., lexicographically, indicated by a subscript index. Hence, we say that rule r_k is *before* (or *after*) rule r_ℓ if $k < \ell$ (or $k > \ell$). We assume that the rules of a rule set S are sorted in ascending order. We say that a rule set S' *starts with* rule set S if $S \subseteq S'$ and all rules in $S' \setminus S$ are after the last rule in S. We denote by $S_{\max} = \arg\max_i \{r_i \in S\}$ the largest rule index in a given rule set $S \subseteq \mathcal{U}$.

For a rule $r_k \in S$, we define $cap(\boldsymbol{x}_n, r_k \mid S) = 1$ if $\boldsymbol{x}_n$ is captured by r_k, but not by rules in S that are before r_k, i.e.,

$$cap(\boldsymbol{x}_n, r_k \mid S) = cap(\boldsymbol{x}_n, r_k) \wedge \bigwedge_{r_\ell \in S \mid \ell < k} (\neg cap(\boldsymbol{x}_n, r_\ell)) .$$

When the context is clear, we use rules (e.g., r_i) and their indices (e.g., i) interchangeably. As a result, a rule set S can be represented as a sorted list of integers and $\mathbf{1}_S \in \{0,1\}^M$ represents the indicator vector of the rule indices in S.

[1] Negative rules, i.e. $q = (y_n = 0)$, when used together with positive rules, may hinder interpretability by simultaneously predicting labels as 0 and 1.

[2] For instance, the set of rules can be obtained via some association rule-mining algorithm [24], like the FP-growth algorithm [20].

3.2 Objective function

To assess a rule set S in terms of accuracy and interpretability, we consider the following objective function:

$$f(S; \lambda) = \ell(S) + \lambda |S|, \tag{1}$$

which consists of a misclassification error term $\ell(S)$ and a penalty term $|S|$ for model complexity. The intuition is that, for a given level of accuracy, shorter rule sets are preferred as they are easier to interpret and are less prone to overfitting. The regularization parameter $\lambda > 0$ controls the relative importance of the two terms.

The loss term ℓ can be decomposed into:

$$\ell(S) = \ell_p(S) + \ell_0(S) , \tag{2}$$

where

$$\ell_p(S) = \frac{1}{N} \sum_{n=1}^{N} \sum_{k=1}^{K} (cap(\boldsymbol{x}_n, r_k \mid S) \wedge \mathbb{1}[y_n \neq 1]) \text{ and} \tag{3}$$

$$\ell_0(S) = \frac{1}{N} \sum_{n=1}^{N} (\neg cap(\boldsymbol{x}_n, S) \wedge \mathbb{1}[y_n = 1]) . \tag{4}$$

The term ℓ_p is the proportion of false positives of the rule set S, while the term ℓ_0 is the proportion of false negatives.

3.3 The Rashomon set of decision sets

Given a set of candidate decision rules $\mathcal{U}$, an objective function $f(\cdot; \lambda)$ for evaluating rule sets, and a parameter $\theta \in \mathbb{R}^+$, we define the Rashomon set of rule sets for $\mathcal{U}$ with respect to λ and θ as:

$$\mathcal{R}(\mathcal{U}, \lambda, \theta) = \{S \subseteq \mathcal{U} \mid f(S; \lambda) \leq \theta\} . \tag{5}$$

When the context is clear, we use $\mathcal{R}(\mathcal{U})$ instead of $\mathcal{R}(\mathcal{U}, \lambda, \theta)$.

In the literature, the Rashomon set is sometimes alternatively defined with $\theta = f^* + \alpha$, where f^* is the optimal objective value.

3.4 Problem formulation

We consider a set of candidate rules $\mathcal{U} = \{r_1, \ldots, r_M\}$, each of which passes a given threshold on the number of captured training points. This definition of $\mathcal{U}$ is common in the literature [5, 24, 25]. To construct $\mathcal{U}$, we resort to the popular FP-growth algorithm [20].

We first consider the problem of exhaustively enumerating $\mathcal{R}(\mathcal{U})$.

Problem 1 (Enumeration). *Given a set of candidate rules $\mathcal{U}$, and parameters $\lambda > 0$ and $\theta > 0$, enumerate all rule sets in $\mathcal{R}(\mathcal{U}, \lambda, \theta)$.*

Solving this problem allows us to compute $|\mathcal{R}(\mathcal{U}, \lambda, \theta)|$ and draw uniform samples from $\mathcal{R}(\mathcal{U}, \lambda, \theta)$. The number $|\mathcal{R}(\mathcal{U}, \lambda, \theta)|$ can be further used to compute the *Rashomon ratio* [37], which is defined as the ratio between $|\mathcal{R}(\mathcal{U})|$ and the total number of models.[3] This ratio is a measure of complexity of a learning problem. The larger the ratio, the more likely that a simple-yet-accurate model exists.

Problem 1 is #**P**-hard and the problems of almost-uniform sampling and approximate counting, defined next, are also hard, as they can be shown to generalize similar problems whose complexity has been established in the literature [44].

We define as a sampling algorithm $\mathcal{S}$ (or sampler) any algorithm that, given as input the set of candidate rules $\mathcal{U}$, the objective function f and the value of the upper bound θ, returns a random

[3] In our case, the Rashomon ratio is computed as $|\mathcal{R}(\mathcal{U})|/(2^M - 1)$.

Algorithm 1 BBenum, a branch-and-bound algorithm to enumerate all rule sets in $\mathcal{R}(\mathcal{U}, \lambda, \theta)$.

```
1:  Q ← Queue([∅])
2:  while Q is not empty do
3:    S ← Q.pop()
4:    for i in {S_max + 1, ..., M} do
5:      S' ← S ∪ {i}
6:      if b(S') ≤ θ {Hierarchical lower bound} then
7:        if b(S') + λ ≤ θ {Look-ahead bound} then
8:          if |S'| ≤ ⌊(θ − b(S'))/λ⌋ {Size bound} then
9:            Q.push(S')
10:       if f(S') ≤ θ then
11:         yield S' {Yield a feasible solution}
```

element from $\mathcal{R}(\mathcal{U})$. Similarly, a counting algorithm C receives the same inputs and estimates $|\mathcal{R}(\mathcal{U})|$.

Problem 2 (Almost-uniform sampling). *Given objective function f, find a sampler $\mathcal{S}$, such that for a bound $\theta \in \mathbb{R}^+$, tolerance parameter $\epsilon \in \mathbb{R}^+$, and $S \in \mathcal{R}(\mathcal{U})$, it holds*

$$\frac{1}{(1+\epsilon)}\frac{1}{|\mathcal{R}(\mathcal{U})|} \leq Pr(\mathcal{S}(\mathcal{U}, f, \theta, \epsilon) = S) \leq (1+\epsilon)\frac{1}{|\mathcal{R}(\mathcal{U})|}. \quad (6)$$

We similarly define the approximate counting problem.

Problem 3 (Approximate counting). *Find a counting algorithm C, such that for a tolerance parameter $\epsilon \in \mathbb{R}^+$ and a confidence parameter $\delta \in [0, 1]$, it holds*

$$Pr\left(\frac{|\mathcal{R}(\mathcal{U})|}{1+\epsilon} \leq C(\mathcal{U}, f, \theta, \epsilon, \delta) \leq (1+\epsilon)\,|\mathcal{R}(\mathcal{U})|\right) \geq 1-\delta. \quad (7)$$

4 AN EXACT ALGORITHM VIA COMPLETE ENUMERATION

In this section, we describe our solution for Problem 1, a branch-and-bound algorithm equipped with effective pruning bounds and incremental computation techniques, which enumerates efficiently all rule sets in $\mathcal{R}(\mathcal{U}, \lambda, \theta)$. Similar enumeration problems have been studied for other types of logical models, such as decision lists [30] and decision trees [43], but new ideas are required for rule sets.

4.1 A branch-and-bound algorithm

In order to find the set of feasible solutions, the algorithm we propose, referred to as BBenum and presented in Algorithm 1, visits rule sets in a breadth-first fashion with the help of a queue and leverages a hierarchy among the rule sets to prune away the rule sets S' that start with a rule set S if certain criteria on S are met.

In particular, at each iteration, the rule set at the front of the queue is popped and extended with an additional rule, whose index is in the range $[S_{\max}+1, \ldots, M]$, to form S'. Next, we check using bounds (described shortly) whether rule set S' and any rule set starting with S' can be pruned. If S' is not pruned, we enqueue it. If in addition the objective value achieved by S' is below the upper bound θ, we add S' to the Rashomon set.

The proposed pruning bounds are based on two key observations: (i) rule sets form a hierarchy under prefix relations, i.e., $S' \subseteq \mathcal{U}$ is a descendant of $S \subseteq \mathcal{U}$ in the hierarchy if S' starts with S; (ii) certain characteristics of a given rule set can determine the feasibility of its descendants in the hierarchy. We next illustrate the details of the pruning bounds. Proofs of all non-trivial results presented in the following sections are provided in an extended version of this paper available online [14].

Hierarchical objective lower bound. For a rule set S, we define:

$$b(S) = \ell_p(S) + \lambda|S|. \quad (8)$$

Then, for any S' that starts with S, the quantity $b(S)$ serves as a lower bound for $f(S')$, as formalized next.

Theorem 1 (Hierarchical objective lower bound). *For any rule set $S \subseteq \mathcal{U}$ and any $S' \subseteq \mathcal{U}$ that starts with S, it is $f(S') \geq b(S)$.*

In other words, all rule sets S' starting with a rule set S for which $b(S) > \theta$ are infeasible.

Look-ahead lower bound. The next bound takes Theorem 1 one step further by observing that any superset of S must include at least an additional rule.

Theorem 2 (Look-ahead lower bound). *For a given rule set $S \subseteq \mathcal{U}$, if $b(S)+\lambda > \theta$, then for any rule set $S' \subseteq \mathcal{U}$ that starts with S and is a proper superset of S (i.e., $S' \neq S$), it holds that $f(S') > \theta$.*

Rule set size bound. Finally, we use the lower bound $b(S)$ to bound the size of any rule set that can be part of the Rashomon set.

Theorem 3 (Rule set size bound). *For a given rule set $S \subseteq \mathcal{U}$ and any rule set $S' \subseteq \mathcal{U}$ that starts with S, if $|S| > \lfloor(\theta - b(S))/\lambda\rfloor$, then $f(S') > \theta$.*

We empirically find that the look-ahead and the rule-set-size bounds are remarkably effective in pruning. Details are presented in the extended version of this paper [14].

4.2 Incremental computation

To further speed up BBenum, we update $b(\cdot)$ and $f(\cdot)$ incrementally. The update formulas are stated below.

Theorem 4 (Lower bound update). *For any rule set $S \subseteq \mathcal{U}$ and any $S' \subseteq \mathcal{U}$ that starts with S and has exactly one more rule r, i.e., $S' = S \cup \{r\}$, the following holds:*

$$b(S') = b(S) + \lambda + \frac{1}{N}\sum_{n=1}^{N}\left(cap(\boldsymbol{x}_n, r \mid S) \wedge \mathbb{1}\left[y_n \neq 1\right]\right).$$

Thus, provided that $b(S)$ is computed already, computing $b(S')$ requires evaluating only the last term in the above sum.

Theorem 5 (Objective update). *For any rule set $S \subseteq \mathcal{U}$ and any $S' \subseteq \mathcal{U}$ that starts with S and has exactly one more rule r, i.e., $S' = S \cup \{r\}$, the following holds:*

$$f(S') = b(S') + \frac{1}{N}\sum_{n=1}^{N}\left(\neg cap(\boldsymbol{x}_n, S) \wedge \neg cap(\boldsymbol{x}_n, r) \wedge \mathbb{1}\left[y_n = 1\right]\right).$$

The details of the branch-and-bound algorithm with incremental computation are provided in the extended version of this paper [14].

5 APPROXIMATION ALGORITHMS BASED ON RANDOM PARTITIONING

In this section, we address Problems 2 and 3. We develop efficient methods with theoretical quality guarantees. To achieve this objective, we leverage the SAT-based framework proposed by Meel [31].

However, since this framework scales poorly for our purposes, we propose novel methods to improve scalability.

5.1 An algorithmic framework based on random parity constraints

We illustrate the proposed framework by first discussing our algorithm for the counting problem, i.e., Problem 3.

Approximate counting. Algorithm ApproxCount, shown as Algorithm 2 in the extended version of this paper [14], generates random parity constraints to partition the solution space into "small cells." It then measures the size of a random cell (i.e., the number of solutions in the cell) and computes an estimate of $|\mathcal{R}(\mathcal{U})|$ by multiplying that cell size by the number of cells.[4] To achieve the desired confidence, the estimation is repeated on sufficiently many random cells, and the median is returned as the final estimate.

To achieve the desired estimation quality, the algorithm determines an upper bound B on cell sizes based on a tolerance parameter ϵ. Each evaluation, carried out by ApproxCountCore (Algorithm 3 in the extended version of this paper [14]) first generates $M-1$ random parity constraints. Then, it searches for the number of constraints that produce a cell of size closest to, but below, B. Finally, all the solutions in that cell are enumerated to obtain the cell size.

Solution space partitioning via parity constraints. A system of parity constraints is imposed on the original enumeration problem (Problem 1). The system consists of k linear equations in the finite field of 2 and can be written as $\boldsymbol{Ax} = \boldsymbol{b}$, where $\boldsymbol{A} \in \{0,1\}^{k\times M}$, $\boldsymbol{b} \in \{0,1\}^k$ and $\boldsymbol{x} \in \{0,1\}^M$(the solution variable). The system $\boldsymbol{Ax} = \boldsymbol{b}$ locates a specific cell among the 2^k counterparts (each corresponding to a different value in $\{0,1\}^k$). The set of feasible solutions in that cell is denoted by $\mathcal{R}(\mathcal{U}; \boldsymbol{A}, \boldsymbol{b}) = \{S \in \mathcal{R}(\mathcal{U}) \mid \boldsymbol{A}\mathbf{1}_S = \boldsymbol{b}\}$.

Searching for the desired k. Given constraints $\boldsymbol{Ax} = \boldsymbol{b}$, where $\boldsymbol{A} \in \{0,1\}^{(M-1)\times M}$ and $\boldsymbol{b} \in \{0,1\}^M$, procedure LogSearch (Algorithm 7 in the extended version of this paper [14]) finds the value of k such that the cell size $|\mathcal{R}(\mathcal{U} \mid \boldsymbol{A}_{:k}, \boldsymbol{b}_{:k})|$ is closest to, but below, B. For each attempted k, LogSearch invokes an oracle ParityConsEnum, which enumerates *at most* B solutions in $\mathcal{R}(\mathcal{U} \mid \boldsymbol{A}_{:k}, \boldsymbol{b}_{:k})$.

Near-uniform sampling. ApproxSample (Algorithm 8 in the extended version of this paper [14]) relies on a similar idea as ApproxCount; the solution space is partitioned into cells and samples are drawn from random cells. The algorithm accepts a tolerance parameter ϵ to determine a range of the desired cell sizes (to guarantee closeness to uniformity). To find the appropriate value of k, it first obtains an estimate $\hat{c}$ of $|\mathcal{R}(\mathcal{U})|$ using ApproxCount. Then, different values of k (determined by $\hat{c}$) are attempted until the resulting cell size falls within the desired range. Finally, a sample is drawn uniformly at random from that cell.

Statistical guarantee. Meel [31] proves that, provided that the oracle ParityConsEnum exists, the counting and sampling algorithms (Algorithm 2 and Algorithm 8 in the extended version of this paper [14]) indeed address the approximate sampling and counting problems (Problems 2 and 3), respectively.

[4]The size of a cell is the number of feasible solutions in it.

5.2 Parity constrained enumeration

The effectiveness of the above approach heavily depends on the implementation of the oracle ParityConsEnum. In the work of Meel [31], SAT-based solvers are used since the work deals with the general problem of constrained programming. In our setting, we rely on BBEnum and linear algebra to design a novel algorithm tailored for our problems for better scalability. Formally, the oracle ParityConsEnum addresses the following problem.

Problem 4 (Partial enumeration under parity constraints). *Given a set of candidate rules $\mathcal{U}$, an objective function f, an upper bound θ, a parity constraint system characterized by $\boldsymbol{Ax} = \boldsymbol{b}$, and an integer B, find a collection of rule sets $\mathcal{S}$ such that $|\mathcal{S}| \le B$, $f(S) \le \theta$, and $\boldsymbol{A}\mathbf{1}_S = \boldsymbol{b}$, for all $S \in \mathcal{S}$.*

Compared to Problem 1, the above problem asks to enumerate at most B solutions and further imposes parity constraints on the solution. Note that Problem 4 is at least as hard as Problem 1, since the latter is a special case.

Without loss of generality, we assume the matrix $\boldsymbol{A}$ is in its reduced row echelon form $\boldsymbol{A}^-$, resulting in the system $\boldsymbol{A}^-\boldsymbol{x} = \boldsymbol{b}^-$.[5] The reason is that for any S, it is $\boldsymbol{A}^-\boldsymbol{x} = \boldsymbol{b}^-$ if and only if $\boldsymbol{Ax} = \boldsymbol{b}$, so that replacing the constraint $\boldsymbol{Ax} = \boldsymbol{b}$ in Problem 4 with $\boldsymbol{A}^-\boldsymbol{x} = \boldsymbol{b}^-$ results in an equivalent problem. Further, important properties revealed by $\boldsymbol{A}^-$, such as the rank and pivot positions, turn out to be essential for the subsequent algorithmic developments. Finally, we assume there is at least one feasible solution to $\boldsymbol{Ax} = \boldsymbol{b}$.

Let ρ be the rank of $\boldsymbol{A}$ and let $pivot_{\boldsymbol{A}} : [\rho] \to [\rho]$ denote the *pivot table* of $\boldsymbol{A}$, where $pivot_{\boldsymbol{A}}[i]$ is the column index of the pivot position in the i-th row. We define $\mathcal{P}_{\boldsymbol{A}} = \{pivot_{\boldsymbol{A}}[i] \mid i \in [\rho]\}$, i.e., the indices of columns corresponding to pivot variables in $\boldsymbol{A}$. Similarly, we define $\mathcal{F}_{\boldsymbol{A}} = \{0, \ldots, M-1\} \setminus \mathcal{P}_{\boldsymbol{A}}$, i.e., the indices of columns corresponding to free columns. When context is clear, for brevity, we drop the subscript $\boldsymbol{A}$ and use $pivot[i]$, $\mathcal{P}$ and $\mathcal{F}$.

We relate the rules to the pivot positions. We call the j-th rule a *pivot rule* if the j-th column in $\boldsymbol{A}$ corresponds to some pivot position, i.e., exists $i \in [\rho_{\boldsymbol{A}}]$ such that $pivot[i] = j$. Otherwise, the rule is called a *free rule.* For rule set S, we denote $\mathcal{P}(S) = \mathcal{P} \cap S$ the set of pivot rules in S and $\mathcal{F}(S) = \mathcal{F} \cap S$ the set of free rules in S.

5.3 A branch-and-bound algorithm

The proposed algorithm builds upon a technique for enumerating solutions to a linear system $\boldsymbol{Ax} = \boldsymbol{b}$ in finite field of 2. During the enumeration process, solutions are pruned using the bounds (Section 4) to satisfy $f(S) \le \theta$.

Enumerating feasible solutions to $\boldsymbol{Ax} = \boldsymbol{b}$. We first consider the problem of enumerating all feasible solutions to $\boldsymbol{Ax} = \boldsymbol{b}$ alone. A straightforward way is by considering the reduced row echelon form of $\boldsymbol{A}$, identifying the pivot variables and free ones, and considering all possible assignments of the free variables .

We give a toy example of 3 constraints and 5 variables: the reduced row echelon form is shown on the left, while the formula for the feasible solutions on the right. The pivot columns (corresponding to x_1, x_2, and x_4) are highlighted in bold. The set of feasible solutions can be enumerated by substituting $[x_3, x_5] \in \{0,1\}^2$ in the equation below.

[5]$\boldsymbol{b}^-$ is obtained via the same operations done on $\boldsymbol{A}$.

$$\begin{bmatrix}\mathbf{1}&\mathbf{0}&0&\mathbf{0}&0\\ \mathbf{0}&\mathbf{1}&1&\mathbf{0}&0\\ \mathbf{0}&\mathbf{0}&0&\mathbf{1}&1\end{bmatrix}\begin{bmatrix}x_1\\x_2\\x_3\\x_4\\x_5\end{bmatrix}=\begin{bmatrix}1\\0\\1\end{bmatrix}\rightarrow x=\begin{bmatrix}1\\0\\0\\1\\0\end{bmatrix}+\begin{bmatrix}0\\1\\1\\0\\0\end{bmatrix}x_3+\begin{bmatrix}0\\0\\0\\1\\1\end{bmatrix}x_5$$

Main idea of the proposed algorithm. Algorithm 2 integrates the above ideas into the search process in Algorithm 1. The main changes are:

(1) In the for loop of Algorithm 1, we only check the free rules. In the example above, only x_3 and x_5 are checked.
(2) While adding a rule j to a given rule set, the procedure EnsMinNonViolation checks if the satisfiability of some parity constraints can already be determined. If this is the case, the associated pivot rules are added.
(3) When checking the look-ahead bound, the pivot rules added by EnsMinNonViolation are considered.
(4) Before yielding a solution, EnsSatisfaction adds relevant pivot rules to guarantee $\boldsymbol{Ax}=\boldsymbol{b}$ is satisfied.
(5) The algorithm terminates when either B solutions or all feasible solutions (at most B) are found.
(6) Finally, a priority queue is used to guide the search process, where the priority of a rule set S equals $-b(S)$.

Ensuring minimal non-violation. To describe the procedure EnsMinNonViolation, we need some additional definitions. Given a matrix $\boldsymbol{A}$, its *boundary table*, denoted by $B_{\boldsymbol{A}}:[\rho_{\boldsymbol{A}}]\rightarrow[M]$, maps a row index to the largest non-zero non-pivot column index of that row in $\boldsymbol{A}$. That is, $B_{\boldsymbol{A}}[i]=\max\left\{j \mid \boldsymbol{A}_{i,j}=1 \text{ and } j\neq pivot_{\boldsymbol{A}}[i]\right\}$ if $\sum_j \boldsymbol{A}_{i,j}>1$, otherwise $B_{\boldsymbol{A}}[i]=-1$, for every $i\in[\rho_{\boldsymbol{A}}]$. In our example, $B_{\boldsymbol{A}}=[-1,2,4]$.

We use the boundary table to check if the satisfiability of constraints in $\boldsymbol{Ax}=\boldsymbol{b}$ can be determined by a given S. Given a constraint $\boldsymbol{A}_{:i}\boldsymbol{x}=\boldsymbol{b}_{:i}$, we say its satisfiability is *determined* by S if $S_{\max}\geq B_{\boldsymbol{A}}[i]$. In other words, adding any rule after $S_{\max}$ does not affect its satisfiability. In our example, the satisfiability of $x_2+x_3=1$ is determined by $\{1,4\}$ and $\{4\}$ but not by $\{1\}$.

Given a rule set S, we say that S is *non-violating* if the constraints in $\boldsymbol{Ax}=\boldsymbol{b}$ that are determined by S are all satisfied. For instance, $\{1,4\}$ and $\{1\}$ are non-violating, while $\{4\}$ is not. Further, we say S is *minimally non-violating* if S is non-violating and removing any rule $\mathcal{P}(S)$ from S violates at least one constraint. For such S, we call each rule in $\mathcal{P}(S)$ a *necessary pivot* for $\mathcal{F}(S)$. In our example, $S=\{1,2,3\}$ is minimally non-violating.

We rely on minimal non-violation to determine the addition of a minimal set of pivot rules to ensure non-violation. Minimality ensures no redundant rules are added, thus the algorithm does not incorrectly prune feasible rule sets.

The procedure EnsMinNonViolation (Algorithm 4 in the extended version of this paper [14]) returns the set of pivot rules to ensure minimal non-violation of a given S. For each constraint, the process checks if it is determined and unsatisfied, and if yes, adds the associated pivot rule. Formally: let $\boldsymbol{r}=\boldsymbol{b}-\boldsymbol{A}\cdot\mathbf{1}_S$. For each $i\in[\rho(\boldsymbol{A})]$, if $\boldsymbol{r}_i=1$ and $S_{\max}\geq B_{\boldsymbol{A}}[i]$, then add the $pivot_{\boldsymbol{A}}[i]$-th pivot rule.

Theorem 6. *Given a parity constraint system $\boldsymbol{Ax}=\boldsymbol{b}$, for any rule set S with free rules only, it follows that EnsMinNonViolation$(S,\boldsymbol{A},\boldsymbol{b})$ returns a set of pivot rules E such that $S\cup E$ is minimally non-violating with respect to $\boldsymbol{Ax}=\boldsymbol{b}$.*

Algorithm 2 A branch-and-bound algorithm to solve Problem 4.

```
n ← 0
S_s ← EnsSatisfaction(∅, A, b)
if R(S_s) ≤ θ then
    Increment n and yield S_s
S_q ← EnsMinNonViolation(∅, A, b)
Q ← PriorityQueue((S_q, b(S_q)))
while Q is not empty and n < B do
    S ← Q.pop()
    for j = (F(S)_max + 1), ..., M and j is free do
        S' ← S ∪ {j}
        if b(S') ≤ θ then
            E_q ← EnsMinNonViolation(S', A, b)
            if b(S' ∪ E_q) + λ ≤ θ then
                Q.push(S', b(S' ∪ E_q))
            E_s ← EnsSatisfaction(S', A, b)
            S_s ← S' ∪ E_s
            if R(S_s) ≤ θ then
                Increment n and yield S_s
```

Ensuring satisfiability. Satisfiability to $\boldsymbol{Ax}=\boldsymbol{b}$ is guaranteed by EnsSatisfaction (Algorithm 5 in the extended version of this paper [14]), which works as follows: let $\boldsymbol{r}=\boldsymbol{b}-\boldsymbol{A}\cdot\mathbf{1}_S$, for each $i\in[\rho(\boldsymbol{A})]$, add the $pivot_{\boldsymbol{A}}[i]$-th pivot rule if $\boldsymbol{r}_i=1$.

Proposition 1. *Given a parity constraint system $\boldsymbol{Ax}=\boldsymbol{b}$, for any rule set S with free rules only, it follows that EnsSatisfaction$(S,\boldsymbol{A},\boldsymbol{b})$ returns a set of pivot rules E such that $\boldsymbol{A}\mathbf{1}_{S\cup E}=\boldsymbol{b}$.*

Extended look-ahead bound. Finally, we extend the look-ahead bound (Theorem 2) to account for the addition of necessary pivots.

Theorem 7 (Extended look-ahead bound). *Given a parity constraint system $\boldsymbol{Ax}=\boldsymbol{b}$, let S be a rule set with free rules only and let E be the set of necessary pivots associated with S with respect to $\boldsymbol{Ax}=\boldsymbol{b}$. If $b(S\cup E)+\lambda>\theta$, then for any S' that starts with S and $S'\neq S$, it follows that $f(S'\cup E')>\theta$, where E' is the set of necessary pivots for S' with respect to $\boldsymbol{Ax}=\boldsymbol{b}$.*

5.4 Incremental computation

We achieve further speed up by incrementally adding the pivots to ensure minimal non-violation and satisfaction. For instance, we address the following question: given a minimally non-violating rule set S, if rule j is added to S, which pivot rules should be added to maintain minimal non-violation of the new rule set?

Two arrays are used to represent the parity and satisfiability states of a rule set S. The *parity states array* $\boldsymbol{z}\in\{0,1\}^k$ stores the difference between $\boldsymbol{A}_i\mathbf{1}_S$ and $\boldsymbol{b}_i$, for each i. The *satisfiability array* $\boldsymbol{s}\in\{0,1\}^k$ stores whether the satisfiability of each constraint is *guaranteed* (meaning determined and satisfied) by S. Computations are saved by (i) skipping the check of constraints whose satisfiability is already guaranteed and (ii) determining the addition of pivots based only on the value of $\boldsymbol{z}$ and $\boldsymbol{b}$. Further, both $\boldsymbol{z}$ and $\boldsymbol{s}$ are updated incrementally. Details are provided in Appendixes C.5 and C.6 of the extended version of this paper [14].

5.5 Implementation details

We also propose a few implementation-level enhancements (details in Appendix C.8 of the extended version of this paper [14]) to speed up even more the above algorithms.

- The columns of $\boldsymbol{A}$ and the rules are permutated to increase the chances that IncEnsNoViolation returns a non-empty pivot sets, leading to more pruning of the search space.
- ApproxCountCore executions are parallelized in ApproxCount.
- We use a fast routine to compute the number of pivot rules required for satisfiability, before calling the more expensive IncEnsSatisfaction. This number is used to check the rule set size bound.

6 SEARCH-TREE-BASED APPROXIMATION ALGORITHMS

In this section we introduce BBsts, a fast alternative to ApproxCount, which draws approximately uniform samples and approximates the size of the Rashomon set. BBsts leverages ideas from the SearchTreeSampler method by Ermon et al. [17] for approximately uniform sampling of solutions (i.e., satisfying assignments) of a set of hard constraints in a combinatorial space.

BBsts assumes that rule sets are organized in a search tree. The root of the search tree is the empty rule set. All rule sets that are b-hops away from the tree root contain exactly b rules. BBsts explores the search tree in a breadth-first fashion. While exploring the tree, BBsts generates *partial rule sets*, which are progressively extended (by adding additional rules) to form the final solutions. Partial rule sets of *level* h are associated with the first h rules in $\mathcal{U}$.

BBsts does not traverse the search tree exhaustively. Given an input parameter ℓ, the search tree is partitioned into $L = \lceil \frac{M}{\ell} \rceil$ *depth levels*. The parameter ℓ controls the approximation level, the smaller ℓ, the larger runtime and expected solution quality. At depth i of the search tree, BBsts generates partial rule sets of level $i\,\ell$.

The steps of BBsts are summarized in Algorithm 3 and visually in Figure 2 of the extended version of this paper [14]. BBsts starts from the tree root which corresponds to the empty rule set being the partial solution P_0 at depth and level 0. Then, at the i-th iteration, partial rule sets P_{i-1} at depth $i-1$ (of level $(i-1)\,\ell$) are uniformly sub-sampled without replacement, and for each sampled partial solution S, BBsts finds all the partial rule sets S' at depth i (of level $i\,\ell$) that start with S. The set of all such partial rule sets at depth i that start with S is denoted by $\{S'\}_i^S$. To find all the partial solutions $\{S'\}_i^S$, BBsts starts from S and invokes a variant of BBenum (S, ℓ) which considers ℓ additional rules. BBenum (S, ℓ) is identical to BBenum, as described in Algorithm 1, except that it starts by enqueueing set S instead of the empty set $\emptyset$, and the main loop only iterates from $(i-1)\,\ell$ to $i\,\ell$, instead of from $S_{max}+1$ to M. The process of drawing a uniform sample S from P_{i-1} and finding the associated set $\{S'\}_i^S$ is repeated $\min(\kappa, |P_{i-1}|)$ times, for a user-specified parameter κ, which trades quality for efficiency. The larger κ is, the longer runtime but higher solution quality.

Eventually, BBsts yields approximate uniform samples from the Rashomon set by generating partial rule sets at depth L (of level M), and filtering out the rule sets that do not belong to the Rashomon set. In particular, Ermon et al. [17] show that, for any partial solutions

Algorithm 3 BBsts algorithm for Problem 2.

```
P_0 ← ∅.
L ← ⌈M/ℓ⌉.
for i in 1, ... L do
    P_i ← ∅
    for j in 1, ... min(κ, |P_{i-1}|) do
        draw S ~ U(P_{i-1}) without replacement
        {S'}_i^S ← BBenum(S, ℓ)
        P_i ← P_i ∪ {S'}_i^S
return {P_i} ∩ R(U)
```

S and S', it holds:

$$\frac{\kappa}{2^\ell + \kappa - 1} \leq \frac{\Pr(S)}{\Pr(S')} \leq \frac{2^\ell + \kappa - 1}{\kappa}, \tag{9}$$

where $\Pr(S)$ denotes the probability of sampling S. Equation (9) bounds the uniformity of the samples returned by BBsts, but it only holds for large κ. For values of κ used in practice, the uniformity guarantee in Equation (9) may not hold, and a rule set S may be arbitrarily more likely to be sampled than another rule set S'.

The use of a BBenum-like search is the main difference between BBsts and SearchTreeSampler [17], which, instead, uses expensive calls to SAT solvers. This difference leads to a drastic reduction in runtime, because, as shown in Section 7.2, BBenum outperforms a SAT-based solver in runtime by orders of magnitude.

Not only BBsts efficiently draws samples from the Rashomon set, but, as suggested by Ermon et al. [17], the partial rule sets constructed while executing BBsts pave the way for estimation of $|\mathcal{R}(\mathcal{U})|$ via the following formula:

$$|\mathcal{R}(\mathcal{U})| \approx |P_L| = \frac{|P_L|}{|P_{L-1}|} \frac{|P_{L-1}|}{|P_{L-2}|} \frac{|P_{L-2}|}{|P_{L-3}|} \cdots \frac{|P_1|}{1}, \tag{10}$$

where $\frac{|P_i|}{|P_{i-1}|} = \frac{1}{|P_{i-1}|} \sum_{S \in P_{i-1}} |\{S'\}_i^S|$. Note that Equation (10) does not provide any accuracy guarantee.

7 EXPERIMENTS

In this section, we present an empirical evaluation of our methods. The main goal of the evaluation is to demonstrate the effectiveness and scalability of the proposed methods for exploring the Rashoomon set of rule sets.

As our methods come with guarantees of near uniformity for sampling, we focus on demonstrating the accuracy of our methods in estimating the size of the Rashomon set (counting). Accurate counts obtained by ApproxCount and BBsts are also good indications of uniform output samples.

We describe the experimental setup in Section 7.1, present a performance comparison in Section 7.2, and describe two case studies in Section 7.3 and 7.4, respectively. We also provide more experiment results in the extended version of this paper [14].

7.1 Experimental setting

We describe our datasets, performance metrics, parameter configurations, and the choices of baselines.

Data. We consider four real-world datasets (whose summary statistics are presented in Table 1) from various domains where interpretability is of primary importance.

Table 1: Summary statistics for the datasets used in the experiments. We report the number of data records N, the number of attributes J, the density in the feature space and the imbalance ratio $(\sum_{n=1}^{N} \mathbb{1}[y_n = 0])/(\sum_{n=1}^{N} \mathbb{1}[y_n = 1])$.

Name	N	J	Feature density	Imbalance ratio
Compas	6 489	15	0.256	1.232
Mushrooms	8 124	117	0.188	1.074
Voting	435	48	0.333	1.589
Credit	690	566	0.019	1.248

- Compas dataset for two-year recidivism prediction [27].
- Mushrooms dataset for classification of mushrooms into the categories poisonous and edible [3].
- Credit dataset for credit scoring [1].
- Voting dataset for classification of american voters as republicans or democrats [2].

Baselines. We compare BBenum, ApproxCount and BBsts against three baselines, Naïve-BB-Enum, a naive variant of BBenum, which does not use pruning and thereby mirrors the theoretical worst-case behaviour of BBenum, CP-sat, a constraint programming solver, and IS, an importance sampler. Details of the baselines are given in Appendix A.

Metrics. Since the main goal in our experimental evaluation is to show that our methods efficiently and effectively explore the Rashomon set of near-optimal rule sets, we report runtime (in seconds) and the estimated Rashomon set size $|\mathcal{R}(\mathcal{U})|$.

Parameters. For fixed value of λ, the choice of upper bound θ affects the most the computational requirements of the proposed algorithms. Hence, we focus on showing runtime and accuracy of counts as a function of θ. Unless specified otherwise, we set $\lambda = 0.1$. This choice of λ shifts the Rashomon set towards concise rule sets prioritizing interpretability over performance. If instead performance is of primary importance, a smaller value of λ (e.g., $\lambda = 0.01$) is preferable. We vary the value of θ in arithmetic progression. For instance, $\theta \in [0.5, 0.7, 0.9, 1.1]$ in the Compas dataset. We construct the universe of rules $\mathcal{U}$ by considering the 50 rules capturing the most data records. In addition, in the extended version of this paper [14], we investigate the performance of BBenum, ApproxCount and BBsts as a function of $|\mathcal{U}|$ (see Figure 4).

When comparing with the baselines, Naïve-BB-Enum, CP-sat and IS, which do not scale well, we use only the 30 rules capturing the most data records and set $\theta = 0.3, 0.5, 0.8,$ and 0.8 for the Compas, Mushrooms, Voting and Credit datasets, respectively.

For ApproxCount, we fix $\epsilon = 0.2$ and $\delta = 0.9$ since varying ϵ and δ does not significantly affect the accuracy of ApproxCount. On the other hand, for BBsts, the parameters ℓ and κ affect accuracy greatly. We consider $\ell \in \{2, 4, 8\}$ and $\kappa \in \{50, 225, 506, 1138, 5760\}$ and we average results over different values of ℓ and κ.

Computing environment and source code. Experiments are executed on a machine with 2×10 core Xeon E5 processor and 256 GB memory. The source code is available at https://github.com/xiaohan2012/efficient-rashomon-rule-set.

7.2 Performance comparison

Comparison among the proposed algorithms. Figure 2 demonstrates how θ affects runtime (top row) and accuracy in estimating $|\mathcal{R}(\mathcal{U})|$ (bottom row), on all datasets. Note that BBenum always returns the correct value for $|\mathcal{R}(\mathcal{U})|$. BBsts is the fastest algorithm, although it can be rather inaccurate in estimating $|\mathcal{R}(\mathcal{U})|$. Instead, ApproxCount strikes the best balance between scalability and accuracy. For large values of θ both BBsts and ApproxCount are drastically more scalable than BBenum, while for small values of θ, BBenum is typically the preferred algorithm.

Comparison against the baselines. The runtime and estimated $|\mathcal{R}(\mathcal{U})|$ for the proposed algorithms and the baselines are provided in Table 2. CP-sat and Naïve-BB-Enum yield exact counts, but they take remarkably longer time than the proposed methods, meanwhile IS delivers estimates that are too inaccurate.

7.3 Case study: feature importance in Compas

We illustrate the application of the approximate sampling algorithm for the task of feature importance analysis. For a feature j in a rule set S, we use *model reliance* [18] to measure the importance of the feature for S. To show the variation of feature importance across the rule sets in the Rashomon set, we compute $\text{MCR}^-(j)$ and $\text{MCR}^+(j)$, the minimum and maximum model reliance values for each feature j. More details are provided in Appendix B.

We obtain the ground truth based on *all* models in $\mathcal{R}(\mathcal{U})$. We also estimate MCR^- and MCR^+ using samples of 400 rule sets extracted from $\mathcal{R}(\mathcal{U})$. In our experiment, we use the Compas dataset and consider a Rashomon set of 2 003 rule sets. The sampling process is repeated 48 times and the mean is reported. Sample estimates of MCR^- and MCR^+ as well as the corresponding ground-truth are shown in Figure 3. Sample estimates are close to the ground-truth, suggesting that exhaustive enumeration of the Rashomon set may not be needed to investigate feature importance. In Section 7.4 we reach similar conclusions regarding a use case on fairness.

7.4 Case study: fairness in Compas

Fairness has emerged as a central topic in machine learning since the influential work of [16]. Exploring the Rashomon set allows to address fairness considerations, which may arise in tackling classification tasks. The goal of this case study is two-fold. Focusing on the Compas dataset, we first show that samples drawn from the Rashomon set by the proposed algorithms yield reliable estimates of popular fairness metrics. Second, we show that the samples can be used to find a model satisfying specific fairness constraints.

Investigating fairness measures by sampling. Figure 4 shows that samples of increasing size provide an increasingly accurate representation of the distribution of fairness measures in the Rashomon set. While there can be some variability in the estimates of the minimum, the maximum, median, first and third quartiles are accurately estimated even in the smallest sample. All details on the fairness measures and experiment settings are given in Appendix C.

Finding accurate-yet-fair models by sampling. In Appendix C we also demonstrate that the algorithms we propose can be used to find an accurate model satisfying certain fairness constraints.

8 CONCLUSIONS

We study the problems of sampling from the Rashomon set of accurate rule set models and computing the size of the Rashomon

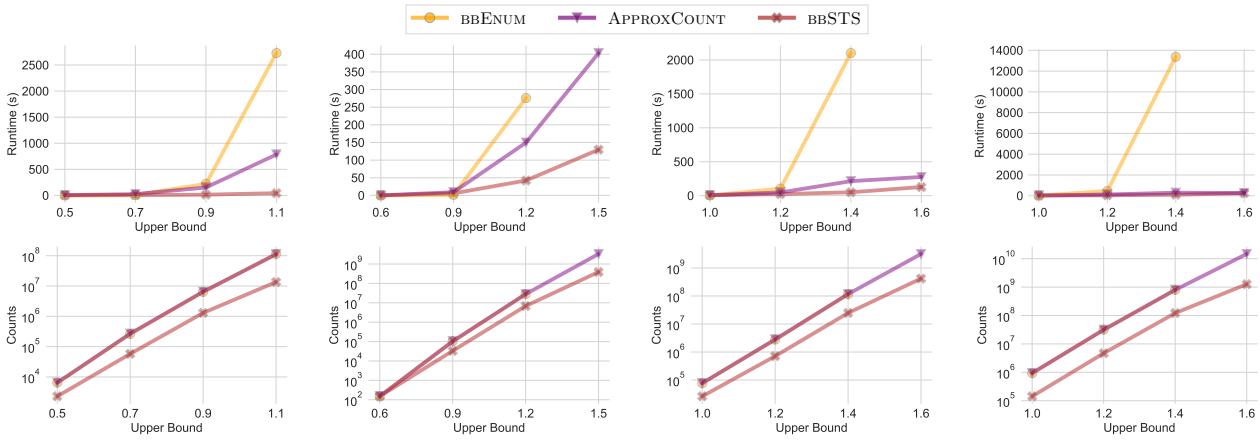

Figure 2: Runtime (in seconds, top row) and estimated $|\mathcal{R}(\mathcal{U})|$ (in log scale, bottom row) against objective upper bound θ.

Table 2: Performance on small problem instances. We report NA if runtime exceeds 12 hours. A * indicates exact counts.

	COMPAS		MUSHROOMS		VOTING		CREDIT	
	Runtime (s)	Count	Runtime (s)	Count	Runtime (s)	Count	Runtime (s)	Count
APPROXCOUNT	0.007	21	0.026	15	0.027	364	1.558	2 810
BBENUM	0.010	*21	0.006	*15	0.022	*364	0.068	*2 807
BBSTS	0.039	21	0.006	16	0.044	289	0.765	2 465
NAÏVE-BB-ENUM	NA	NA	30 381.5	*15	29 981.5	*364	31 161.3	*2 807
CP-SAT	10.277	*21	26.478	*15	2.072	*364	18.789	*2 807
IS	11.128	0	13.348	2	17.445	701	13.919	3 641

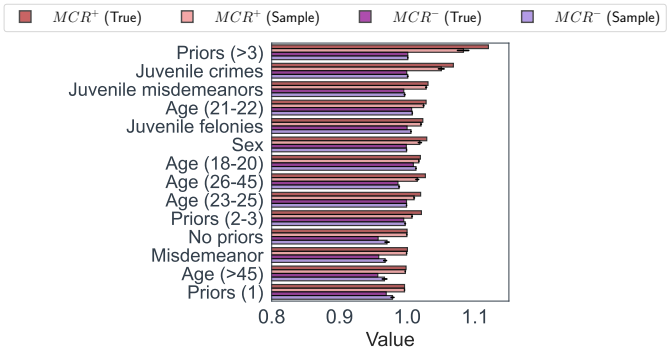

Figure 3: Estimated feature importance against the ground-truth in the COMPAS dataset. 95% confidence intervals are shown as black lines.

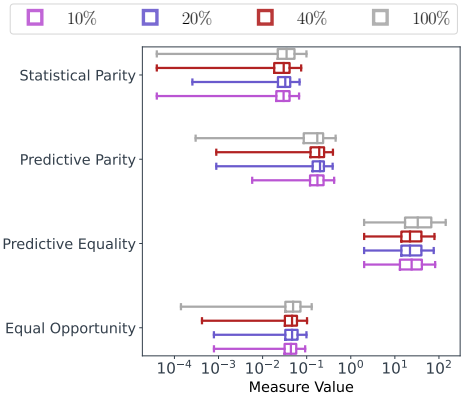

Figure 4: Distribution of four fairness measures in the entire Rashomon set (100%) as well as in samples of increasing size (10%, 20% and 40%). The x-axis is on log scale.

set. Unlike in related work, we consider both exhaustive and non-exhaustive enumeration. For the former, we propose an efficient branch-and-bound algorithm, optimized with pruning and incremental computation. For the latter, we devise two algorithms: one based on the random partitioning of the solution space and another based on subsampling partial solutions during the branch-and-bound exploration of the search tree of rule sets.

Our work opens interesting questions for future research. For example, (*i*) can we make APPROXCOUNT even faster by exploiting the parity constraint further? (*ii*) Can we improve the accuracy of BBSTS without sacrificing efficiency? (*iii*) Can we design algorithms for non-exhaustive exploration of the (possibly continuous) Rashomon set for other classes of interpretable models? And finally (*iv*) can we showcase algorithms for non-exhaustive exploration of the Rashomon set in unexplored application scenarios?

9 ACKNOWLEDGEMENTS

This research is supported by the ERC Advanced Grant REBOUND (834862), the EC H2020 RIA project SoBigData++ (871042), and the Wallenberg AI, Autonomous Systems and Software Program (WASP) funded by the Knut and Alice Wallenberg Foundation.

REFERENCES

[1] [n. d.]. Credit Score Classification task. http://kaggle.com/datasets/parisrohan/credit-score-classification. Accessed: 2023-10-01.

[2] 1987. Congressional Voting Records. UCI Machine Learning Repository. DOI: https://doi.org/10.24432/C5C01P.

[3] 1987. Mushroom. UCI Machine Learning Repository. DOI: https://doi.org/10.24432/C5959T.

[4] Ulrich Aïvodji, Julien Ferry, Sébastien Gambs, Marie-José Huguet, and Mohamed Siala. 2021. Faircorels, an open-source library for learning fair rule lists. In *Proceedings of the 30th ACM International Conference on Information & Knowledge Management*. 4665–4669.

[5] Elaine Angelino, Nicholas Larus-Stone, Daniel Alabi, Margo Seltzer, and Cynthia Rudin. 2017. Learning certifiably optimal rule lists. In *Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. 35–44.

[6] Vaishak Belle and Ioannis Papantonis. 2021. Principles and practice of explainable machine learning. *Frontiers in big Data* (2021), 39.

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

[8] Nadia Burkart and Marco F Huber. 2021. A survey on the explainability of supervised machine learning. *Journal of Artificial Intelligence Research* 70 (2021), 245–317.

[9] Toon Calders, Faisal Kamiran, and Mykola Pechenizkiy. 2009. Building classifiers with independency constraints. In *2009 IEEE international conference on data mining workshops*. IEEE, 13–18.

[10] Frédéric Cérou, Pierre Del Moral, Teddy Furon, and Arnaud Guyader. 2012. Sequential Monte Carlo for rare event estimation. *Statistics and computing* 22, 3 (2012), 795–808.

[11] Supratik Chakraborty, Kuldeep S Meel, and Moshe Y Vardi. 2013. A scalable and nearly uniform generator of SAT witnesses. In *Computer Aided Verification: 25th International Conference, CAV 2013, Saint Petersburg, Russia, July 13-19, 2013. Proceedings 25*. Springer, 608–623.

[12] Supratik Chakraborty, Kuldeep S Meel, and Moshe Y Vardi. 2014. Balancing scalability and uniformity in SAT witness generator. In *Proceedings of the 51st Annual Design Automation Conference*. 1–6.

[13] M. Ciaperoni, H. Xiao, and A. Gionis. 2022. Concise and interpretable multi-label rule sets. In *2022 IEEE International Conference on Data Mining (ICDM)*. IEEE Computer Society, Los Alamitos, CA, USA, 71–80. https://doi.ieeecomputersociety.org/10.1109/ICDM54844.2022.00017

[14] Martino Ciaperoni, Han Xiao, and Aristides Gionis. 2024. Efficient Exploration of the Rashomon Set of Rule Set Models. arXiv:2406.03059 [cs.LG]

[15] Amanda Coston, Ashesh Rambachan, and Alexandra Chouldechova. 2021. Characterizing fairness over the set of good models under selective labels. In *International Conference on Machine Learning*. PMLR, 2144–2155.

[16] Cynthia Dwork, Moritz Hardt, Toniann Pitassi, Omer Reingold, and Richard Zemel. 2012. Fairness through awareness. In *Proceedings of the 3rd innovations in theoretical computer science conference*. 214–226.

[17] Stefano Ermon, Carla Pedro Gomes, and Bart Selman. 2012. Uniform Solution Sampling Using a Constraint Solver As an Oracle. In *Conference on Uncertainty in Artificial Intelligence*. https://api.semanticscholar.org/CorpusID:16218653

[18] Aaron Fisher, Cynthia Rudin, and Francesca Dominici. 2019. All Models are Wrong, but Many are Useful: Learning a Variable's Importance by Studying an Entire Class of Prediction Models Simultaneously. *J. Mach. Learn. Res.* 20, 177 (2019), 1–81.

[19] Aristides Gionis, Theodoros Lappas, and Evimaria Terzi. 2012. Estimating entity importance via counting set covers. In *Proceedings of the 18th ACM SIGKDD international conference on Knowledge discovery and data mining*. 687–695.

[20] Jiawei Han, Jian Pei, and Yiwen Yin. 2000. Mining frequent patterns without candidate generation. *ACM sigmod record* 29, 2 (2000), 1–12.

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

[22] Xiyang Hu, Cynthia Rudin, and Margo Seltzer. 2019. Optimal sparse decision trees. *Advances in Neural Information Processing Systems* 32 (2019).

[23] Katarzyna Kobylińska, Mateusz Krzyziński, Rafał Machowicz, Mariusz Adamek, and Przemysław Biecek. 2023. Exploration of Rashomon Set Assists Explanations for Medical Data. *arXiv preprint arXiv:2308.11446* (2023).

[24] Trupti A Kumbhare and Santosh V Chobe. 2014. An overview of association rule mining algorithms. *International Journal of Computer Science and Information Technologies* 5, 1 (2014), 927–930.

[25] Himabindu Lakkaraju, Stephen H Bach, and Jure Leskovec. 2016. Interpretable decision sets: A joint framework for description and prediction. In *Proceedings of the 22nd ACM SIGKDD international conference on knowledge discovery and data mining*. 1675–1684.

[26] Himabindu Lakkaraju and Osbert Bastani. 2020. " How do I fool you?" Manipulating User Trust via Misleading Black Box Explanations. In *Proceedings of the AAAI/ACM Conference on AI, Ethics, and Society*. 79–85.

[27] Jeff Larson, Surya Mattu, Lauren Kirchner, and Julia Angwin. 2016. How we analyzed the COMPAS recidivism algorithm. *ProPublica (5 2016)* 9, 1 (2016), 3–3.

[28] Dmitry Malioutov and Kuldeep S Meel. 2018. MLIC: A MaxSAT-based framework for learning interpretable classification rules. In *International Conference on Principles and Practice of Constraint Programming*. Springer, 312–327.

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

[30] Kota Mata, Kentaro Kanamori, and Hiroki Arimura. 2022. Computing the Collection of Good Models for Rule Lists. *arXiv preprint arXiv:2204.11285* (2022).

[31] Kuldeep Singh Meel. 2017. *Constrained counting and sampling: bridging the gap between theory and practice*. Ph. D. Dissertation. Rice University.

[32] Laurent Perron and Frédéric Didier. [n. d.]. *CP-SAT*. Google. https://developers.google.com/optimization/cp/cp_solver/

[33] Cynthia Rudin. 2019. Stop explaining black box machine learning models for high stakes decisions and use interpretable models instead. *Nature machine intelligence* 1, 5 (2019), 206–215.

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

[35] Mirka Saarela and Susanne Jauhiainen. 2021. Comparison of feature importance measures as explanations for classification models. *SN Applied Sciences* 3 (2021), 1–12.

[36] Lesia Semenova, Cynthia Rudin, and Ronald Parr. 2019. A study in Rashomon curves and volumes: A new perspective on generalization and model simplicity in machine learning. *arXiv preprint arXiv:1908.01755* (2019).

[37] Lesia Semenova, Cynthia Rudin, and Ronald Parr. 2022. On the existence of simpler machine learning models. In *Proceedings of the 2022 ACM Conference on Fairness, Accountability, and Transparency*. 1827–1858.

[38] Surya T Tokdar and Robert E Kass. 2010. Importance sampling: a review. *Wiley Interdisciplinary Reviews: Computational Statistics* 2, 1 (2010), 54–60.

[39] Leslie G Valiant. 1979. The complexity of enumeration and reliability problems. *siam Journal on Computing* 8, 3 (1979), 410–421.

[40] Srishti Vashishtha and Seba Susan. 2019. Fuzzy rule based unsupervised sentiment analysis from social media posts. *Expert Systems with Applications* 138 (2019), 112834.

[41] Tong Wang and Cynthia Rudin. 2015. Learning optimized Or's of And's. *arXiv preprint arXiv:1511.02210* (2015).

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

[43] Rui Xin, Chudi Zhong, Zhi Chen, Takuya Takagi, Margo Seltzer, and Cynthia Rudin. 2022. Exploring the whole rashomon set of sparse decision trees. *Advances in Neural Information Processing Systems* 35 (2022), 14071–14084.

[44] Guangyi Zhang and Aristides Gionis. 2020. Diverse rule sets. In *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*. 1532–1541.

[45] Guangyi Zhang and Aristides Gionis. 2023. Regularized impurity reduction: accurate decision trees with complexity guarantees. *Data mining and knowledge discovery* 37, 1 (2023), 434–475.

Appendices

A ADDITIONAL BASELINES

In order to offer a more complete assessment of the performance of the methods we propose, we compare BBenum, BBsts and ApproxCount against three simpler alternative approaches. The details of such simple baselines are given next.

- Naïve-BB-Enum: a naïve search algorithm that does not enforce any pruning of the search space. The Naïve-BB-Enum algorithm is analogous to BBenum, but it exhaustively considers all rule sets and tests them for inclusion into the Rashomon set.
- CP-sat: a constraint programming solver that uses a highly optimized SAT (satisfiability) solver. In order to leverage the solver, we encode the problem as follows.
  A data record $(\boldsymbol{x}_n, y_n)$ is said to be positive if $y_n = 1$ and negative otherwise. The numbers of negative and positive data records are denoted by $|\mathcal{D}|^-$ and $|\mathcal{D}|^+$, respectively.

Given M input rules, let $\mathbf{N}_I$ be an indicator matrix such that $\mathbf{N}_{I_{i,j}} = 1$ if the i-th negative data record is covered by the j-th rule. Similarly, let $\mathbf{P}_I$ be an indicator matrix such that $\mathbf{P}_{I_{i,j}} = 1$ if the i-th positive data record is covered by the j-th rule. We encode the counting problem as the problem of finding all $\mathbf{x} \in \{0,1\}^M$, such that:

$$\frac{z_{FP}}{N} + \frac{z_{FN}}{N} + \lambda \sum_{j=1}^{M} \mathbf{x}[j] \leq \theta, \tag{11}$$

where $z_{FP} = \sum_{i\in 1}^{|\mathcal{D}|^-} \min(\mathbf{N}_{I_i}\mathbf{x}, 1)$ and $z_{FN} = \sum_{i\in 1}^{|\mathcal{D}|^+} \max(1 - \mathbf{P}_{I_i}\mathbf{x}, 0)$. Here, $\mathbf{x}[j]$ is the j-th entry of $\mathbf{x}$, $\mathbf{N}_{I_i}\mathbf{x}$ denotes the dot product between the i-th row of $\mathbf{N}_I$ and the vector $\mathbf{x}$. Similarly, $\mathbf{P}_{I_i}\mathbf{x}$ denotes the dot product between the i-th row of $\mathbf{P}_I$ and $\mathbf{x}$. As $\mathbf{N}_{I_i}$, $\mathbf{P}_{I_i}$ and $\mathbf{x}$ are all binary vectors, the dot product corresponds to a set intersection.

Given the set of constraints described in Equations 11, we find all rule sets by resorting to a state-of-the-art solver for constraint programming [32].

- IS: a method based on Monte Carlo simulation, where we simulate a large number of rule sets and evaluate the proportion of rule sets that belong to the Rashomon set. The proportion can then be mapped to the corresponding count by multiplying by the total number of rule-set models, that can be easily computed. Plain Monte Carlo sampling is extremely inefficient for very rare events [10]. As suggested by Semenova et al. [36], in order to estimate the size of the Rashomon set, it is preferable to use the Monte Carlo method known as importance sampling [38], where the training data are used to bias the sampling towards the Rashomon set. In particular, the importance sampler is designed as follows.
  - Given the set of pre-mined rules $\mathcal{U}$, compute the normalized reciprocal individual contribution of each rule r to the loss, namely $\Delta\ell(r) = \frac{1}{\ell_p(r)+\ell_0(r)} / \sum_{r'\in\mathcal{U}} \Delta\ell(r')$.
  - Sample N_{rep} (1,000,000 by default) integers t uniformly at random in the interval $[1, |\mathcal{U}|]$ and rule sets S of size t with probability $p(S) = \Delta\ell'(r_1)\Delta\ell'(r_2)\ldots\Delta\ell'(r_n)$.
  - Compute the importance sampling estimate $\frac{1}{N}\sum_{i=1}^{N} f_I(S)\frac{u(S)}{p(S)}$ where $f_I(S)$ is an indicator function for the event that S belongs to the Rashomon set, and $u(\cdot)$ is the uniform (target) distribution.

  In practice, to enhance the performance of IS, instead of sampling rule sets of length up to $|\mathcal{U}|$, we sample rule sets of length up to the upper bound obtained by setting $S = \emptyset$ in Theorem 3.

B CASE STUDY ON FEATURE IMPORTANCE

As a simple case study, we show how the proposed methods allow to efficiently estimate feature importance and, more specifically, we show that reliable estimates of feature importance can be derived from samples of rule sets in the Rashomon set. The main results are summarized in Section 7.3. In this section, we provide more details.

Different measures of feature importance have been proposed [35]. Recent work focuses on *model reliance* [18, 43]. Model reliance captures the extent to which a model relies on a given feature to achieve its predictive performance. For our purposes, given rule set S and feature v, we define model reliance as follows:

$$MR(S, v) = \frac{f(S; v', \lambda)}{f(S; v, \lambda)}, \tag{12}$$

where $f(S; v, \lambda)$ is the objective achieved by S in the original dataset, and $f(S; v', \lambda)$ is identical to $f(S; v, \lambda)$ except that v is replaced by its uninformative counterpart v'. Feature v' is obtained by swapping the first and second halves of the feature values of v, thereby retaining the marginal distribution of v, while destroying its predictive power. This measure is similar to the *model reliance* measure used by Xin et al. [43]. Model reliance evaluates how important a variable is for a given rule set. In particular, the higher model reliance, the more important feature v. If we have a single rule set S, we would simply estimate the importance of feature v by $MR(S, v)$. However, if we have access to the Rashomon set of all near-optimal rule sets, it is more informative to investigate the variation of $MR(S, v)$ across rule sets S in the Rashomon set. Hence, we compute $MCR^-(v)$ and $MCR^+(v)$, the minimum and maximum model reliance for feature v across rule sets in the Rashomon set.

In Figure 3 (Section 7.3), we compare $MCR^-(v)$ and $MCR^+(v)$ computed in the entire Rashomon set and in samples of rule sets drawn from the Rashomon set by ApproxSample, and we conclude that the sample estimates are consistently close to the measures computed in the entire Rashomon set, suggesting that exhaustive enumeration may be redundant when the goal is to investigate feature importance.

In addition, while $MCR^-(v)$ and $MCR^+(v)$ are adequate measures of the importance of features in rule sets, they fail to capture the idea that some features are more frequent than others in the Rashomon set. Inuitively, at parity model reliance, the more frequent a feature is in the Rashomon set, the more important. Hence, to provide a more complete assessment of feature importance, Figure 5 shows the proportion of rule sets including a given variable in the entire Rashomon set or in a sample of 400 rule sets obtained using ApproxSample. The reported sample estimates are obtained as averages over 10 repetitions of the sampling process. The relative frequency of the features estimated in the sample and in the entire Rashomon set are remarkably similar, corroborating the findings presented in Section 7.3 with respect to model reliance.

Finally, we mention that the results do not correspond exactly to the similar results presented by Xin et al. [43] because we consider a different class of models and a different loss. However, there are interesting commonalities. For instance, the variable $Prior > 3$ has the highest $MCR^+(v)$ in both studies.

C CASE STUDY ON FAIRNESS

The Rashomon set offers a novel perspective on fairness of machine learning models. Although all models in the Rashomon set achieve near-optimal predictive performance, they may exhibit different fairness characteristics. The Rashomon set allows to identify the range of predictive bias produced by the models and to search for models that are both accurate and fair.

We carry out a case study focusing on the Compas dataset, which has fueled intense debate and research in fair machine learning [4, 34], and fairness constraints are specified with respect to the *sex* attribute, which partitions the dataset into two groups, *males* (M) and *females* (F).

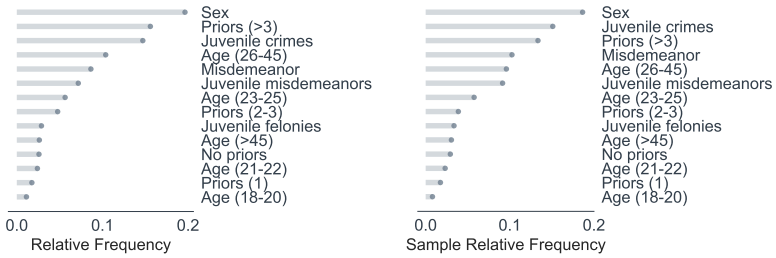

Figure 5: Relative frequencies of features in the Rashomon set (left) and associated sample estimates (right).

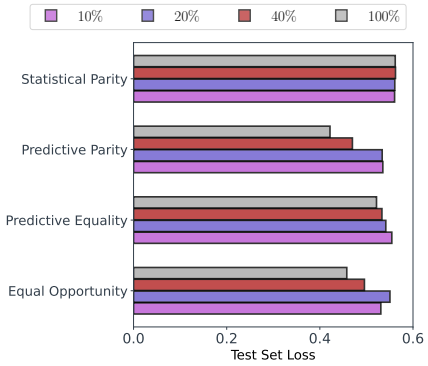

Figure 6: Objective f on the test set obtained by the optimal fair rule set found in the train set with respect to different fairness measures in the entire Rashomon set (100%) as well as in samples of increasing size (10%, 20% and 40%).

A concise summary of the case study is given in Section 7.4. Here, we discuss the details.

First, we introduce the considered measures of fairness.

Fairness measures. Let $\hat{y} = 1$ denote the event that the data record $\boldsymbol{x}$ is predicted as positive (i.e. $cap(\boldsymbol{x}, S)$). Moreover, let x_s denote the sex of data record $\boldsymbol{x}$. We consider the following fairness measures [4].

- *Statistical parity* measures the absolute difference of rate of positive predictions between the groups:
$$|\Pr(\hat{y} = 1|x_s = \text{F}) - \Pr(\hat{y} = 1|x_s = \text{M})|.$$
- *Predictive parity* measures the absolute difference of precision between the groups:
$$|\Pr(y = 1|\hat{y} = 1, x_s = \text{F}) - \Pr(y = 1|\hat{y} = 1, x_s = \text{M})|.$$
- *Predictive equality* measures the absolute difference of false positive rate between the groups:
$$|\Pr(\hat{y} = 1|y = 0, x_s = \text{F}) - \Pr(\hat{y} = 1|y = 0, x_s = \text{M})|.$$
- *Equal opportunity* measures the absolute difference of true positive rate between the groups:
$$|\Pr(\hat{y} = 1|y = 1, x_s = \text{F}) - \Pr(\hat{y} = 1|y = 1, x_s = \text{M})|.$$

For all four measures, the larger the value, the more unfair the model is.

Investigating fairness measures by sampling. Figure 4 in Section 7.4 shows the distribution of the above fairness measures in the entire Rashomon set and in samples of increasing size. For each measure, we show the range (minimum and maximum), the interquartile range (first and third quartiles) and the median. All such statistics describing the distributions of the fairness measures of interest in the samples of rule sets are obtained as average over 10 repetitions of the random sampling process.

The Rashomon set consists of $|\mathcal{R}(\mathcal{U})| = 1409$ rule sets and we use ApproxSample to draw samples of sizes 10%, 20% and 40% of $|\mathcal{R}(\mathcal{U})|$.

Finding accurate-yet-fair models by sampling. To demonstrate that the proposed sampling strategy can be used to find an accurate model while satisfying particular fairness constraints, we set up a simple two-step experiment.

First, given a sample of rule sets from the Rashomon set, we consider any of the four fairness measures described above, say M_{fair}, and we exclude all models with value of M_{fair} beyond the first quartile of the distribution of M_{fair} in the entire Rashomon set.

The remaining models are referred to as fair models (with respect to the chosen M_{fair}). Second, among the remaining (fair) models, we pick the model S^* which minimizes the objective f.

Figure 6 reports the value of f in the test set for the chosen rule set S^* in the entire Rashomon set and in samples of rule sets of increasing size. Again, the Rashomon set consists of $|\mathcal{R}(\mathcal{U})| = 1,409$ rule sets and we draw samples of sizes 10%, 20% and 40% of $|\mathcal{R}(\mathcal{U})|$ using ApproxSample. The reported losses are obtained as average over 10 repetitions of the sampling process. The performance of the optimal *fair* rule set chosen from the samples is not far from the performance of the optimal *fair* rule set chosen in the entire $\mathcal{R}(\mathcal{U})$, and the gap between the performance of the optimal *fair* rule set in the entire Rashomon set and in samples drawn from it quickly shrinks as the sample size increases. In the case of statistical parity, no significant difference is observed across different sample sizes, suggesting that even the smallest sample is enough to find a rule set which is fair with respect to statistical parity and exhibits predictive performance indistinguishable from the predictive performance of the fair rule set that would be chosen in the entire Rashomon set.

Thus, in view of the results reported in this section, we conclude that exhaustive enumeration of the Rashomon set may be redundant when the goal is to investigate fairness or find a model that is both accurate and fair. A representative sample may suffice.