SIRUS: Stable and Interpretable RUle Set

Clément Bénard, Gérard Biau, Sébastien da Veiga, Erwan Scornet

SIRUS: Stable and Interpretable RUle Set for Classification

Clément Bénard* Gérard Biau † Sébastien Da Veiga ‡ Erwan Scornet§

Abstract

State-of-the-art learning algorithms, such as random forests or neural networks, are often qualified as "black-boxes" because of the high number and complexity of operations involved in their prediction mechanism. This lack of interpretability is a strong limitation for applications involving critical decisions, typically the analysis of production processes in the manufacturing industry. In such critical contexts, models have to be interpretable, i.e., simple, stable, and predictive. To address this issue, we design SIRUS (Stable and Interpretable RUle Set), a new classification algorithm based on random forests, which takes the form of a short list of rules. While simple models are usually unstable with respect to data perturbation, SIRUS achieves a remarkable stability improvement over cutting-edge methods. Furthermore, SIRUS inherits a predictive accuracy close to random forests, combined with the simplicity of decision trees. These properties are assessed both from a theoretical and empirical point of view, through extensive numerical experiments based on our `R/C++` software implementation `sirus` available from `CRAN`.

Keywords: classification, interpretability, rules, stability, random forests.

1 Introduction

1.1 Motivations

In the manufacturing industry, production processes involve complex physical and chemical phenomena, whose control and efficiency are of critical importance. In practice, data is collected along the manufacturing line, describing both the production environment and its conformity. The retrieved information enables to infer a link between the manufacturing conditions and the resulting quality at the end of the line, and then to increase the process efficiency. Since the quality of the produced entities is often characterized by a pass or fail output, the problem is in fact a classification task, and state-of-the-art learning algorithms can successfully catch patterns of these complex and nonlinear physical phenomena. However, any decision impacting the production process has long-term and heavy consequences,

*Safran Tech, Sorbonne Université
†Sorbonne Université
‡Safran Tech
§Ecole Polytechnique

and therefore cannot simply rely on a blind stochastic modelling. As a matter of fact, a deep physical understanding of the forces in action is required, and this makes black-box algorithms unappropriate. In a word, models have to be interpretable, i.e., provide an understanding of the internal mechanisms that build a relation between inputs and ouputs, to provide insights to guide the physical analysis. This is for example typically the case in the aeronautics industry, where the manufacturing of engine parts involves sensitive casting and forging processes. Interpretable models allow us to gain knowledge on the behavior of such production processes, which can lead, for instance, to identify or fine-tune critical parameters, improve measurement and control, optimize maintenance, or deepen understanding of physical phenomena. In the following paragraphs, we deepen the discussion about the definition of interpretability to highlight the limitations of the most popular interpretable nonlinear models: decision trees and rule algorithms (Guidotti et al., 2018). Despite their high predictivity and simple structure, these methods are unstable, which is a strong operational limitation. The goal of this article is to introduce **SIRUS** (**S**table and **I**nterpretable **RU**le **S**et), an interpretable rule classification algorithm which considerably improves stability over state-of-the-art methods, while preserving their simple structure, accuracy, and computational complexity.

As stated in Rüping (2006), Lipton (2016), Doshi-Velez and Kim (2017), or Murdoch et al. (2019), to date, there is no agreement in statistics and machine learning communities about a rigorous definition of interpretability. There are multiple concepts behind it, many different types of methods, and a strong dependence on the area of application and the audience. Here, we focus on models intrinsically interpretable, which directly provide insights on how inputs and outputs are related, as opposed to the post-processing of black-box models. In that case, we argue that it is possible to define minimum requirements for interpretability through the triptych "simplicity, stability, and predictivity", in line with the framework recently proposed by Yu and Kumbier (2019). Indeed, in order to grasp how inputs and outputs are related, the structure of the model has to be simple. The notion of simplicity is implied whenever interpretability is invoked (e.g., Rüping, 2006; Freitas, 2014; Letham, 2015; Letham et al., 2015; Lipton, 2016; Ribeiro et al., 2016; Murdoch et al., 2019) and essentially refers to the model size, complexity, or the number of operations performed in the prediction mechanism. Yu (2013) defines stability as another fundamental requirement for interpretability: conclusions of a statistical analysis have to be robust to small data perturbations to be meaningful. Indeed, a specific analysis is likely to be run multiple times, eventually adding a small new batch of data, and an interpretable algorithm should be insensitive to such modifications. Otherwise, unstable models provide us with a partial and arbitrary analysis of the underlying phenomena, and arouses distrust of the domain experts. Finally, if the predictive accuracy of an interpretable model is significantly lower than the one of a state-of-the-art black-box algorithm, it clearly misses strong patterns in the data and will therefore be useless, as explained in Breiman (2001b). For example, the trivial model that outputs the empirical mean of the observations for any input is simple, stable, but brings in most cases no useful information. Thus, we add a good predictivity as an essential requirement for interpretability.

Decision trees are a class of supervised learning algorithms that recursively partition the input space and make local decisions in the cells of the resulting partition. Trees can model highly nonlinear patterns while having a simple structure, and are therefore good candidates when interpretability is required. However, trees are unstable to small data perturbations (Oates and Jensen, 1997; Guidotti and Ruggieri, 2019). More precisely, as explained in Breiman

(2001b): by randomly removing only $2-3\%$ of the training data, the tree structure can be quite different, which is a strong limitation to their practical use. Another class of supervised learning methods that can model nonlinear patterns while retaining a simple structure are the so-called rule models. As such, a rule is defined as a conjunction of constraints on input variables, which form a hyperrectangle in the input space where the estimated output is constant. A collection of rules is combined to form a model. Here, the term "rule" does not stand for "classification rule" but, as is traditional in the rule learning literature, to a piecewise constant estimate that simply reads "if *conditions on* $\boldsymbol{x}$, then *response*, else *default response*". Despite their simplicity and excellent predictive skills, rule algorithms are unstable and, from this point of view, share the same limitation as decision trees (Letham et al., 2015; Murdoch et al., 2019).

1.2 Literature Review

Main decision tree algorithms are CART (Breiman et al., 1984) and C4.5 (Quinlan, 1992). A widespread method to stabilize decision trees is bagging (Breiman, 1996), in which multiple trees are grown on perturbed data and aggregated together. Random forests is an algorithm developped by Breiman (2001a) that improves over bagging by randomizing the tree construction. Predictions are stable, accuracy is increased, but the final model is unfortunately a black box. Thus, simplicity of trees is lost, and some post-treatment mechanisms are needed to understand how random forests make their decisions. Nonetheless, even if they are useful, such treatments only provide partial information and can be difficult to operationalize for critical decisions (Rudin, 2018). For example, variable importance (Breiman, 2001a, 2003a) identifies variables that have a strong impact on the output, but not which inputs values are associated to output values of interest. Similarly, local approximation methods such as LIME (Ribeiro et al., 2016) do not provide insights on the global relation.

Rule learning originates from the influential AQ system of Michalski (1969). Many algorithms based on greedy heuristics were subsequently developped in the 1980's and 1990's, including Decision List (Rivest, 1987), CN2 (Clark and Niblett, 1989), FOIL (First-Order Inductive Learner, Quinlan, 1990; Quinlan and Cameron-Jones, 1995), IREP (Incremental Reduced Error Pruning, Fürnkranz and Widmer, 1994), RIPPER (Repeated Incremental Pruning to Produce Error Reduction, Cohen, 1995), PART (Partial Decision Trees, Frank and Witten, 1998), SLIPPER (Simple Learner with Iterative Pruning to Produce Error Reduction, Cohen and Singer, 1999), and LRI (Leightweight Rule Induction, Weiss and Indurkhya, 2000). At the end of the 1990's a new type of rule algorithms based on frequent pattern mining is introduced with CBA (Classification Based on Association Rules, Liu et al., 1998), then extended with CPAR (Classification based on Predictive Association Rules, Yin and Han, 2003). Frequent pattern mining is originally used to identify frequent occurrences in database mining, and is thus an efficient approach to identify good candidate rules in a classification setting. The last decade has seen a resurgence of rule models, especially powerful algorithms based on rule extraction from tree ensembles with RuleFit (Friedman and Popescu, 2008) and Node harvest (Meinshausen, 2010), but also ENDER (Ensemble of Decision Rules, Dembczyński et al., 2010), and new algorithms based on frequent pattern mining: BRL (Bayesian Rule Lists, Letham et al., 2015), and IDS (Lakkaraju et al., 2016, Interpretable Decision Sets). To the best of our knowledge, the signed iterative random forest method (s-iRF, Kumbier et al.,

2018) is the only procedure that tackles both rule learning and stability. Using random forests, s-IRF manages to extract stable signed interactions, i.e., feature interactions enriched with a thresholding behavior for each variable, lower or higher, but without specific thresholding values. Therefore, s-IRF can be difficult to operationalize since it does not provide any specific input thresholds, and thus no precise information about the influence of input variables. On the other hand, an explicit rule model identifies specific regions of interest in the input space.

1.3 SIRUS Overview

In line with the above, we design SIRUS in the present paper, a new rule classification algorithm which inherits an accuracy close to random forests and the simplicity of decision trees, while having a stable structure. The core aggregation principle of random forests is kept, but instead of aggregating predictions, SIRUS focuses on the probability that a given hyperrectangle (i.e., a node) is contained in a randomized tree. The nodes with the highest probability are robust to data perturbation and represent strong patterns. They are therefore selected to form a stable rule ensemble model. In Section 3 we illustrate SIRUS on a real and open dataset, SECOM (Dua and Graff, 2017), from a semi-conductor manufacturing process. Data is collected from 590 sensors and process measurement points $(X^{(1)}, X^{(2)}, \dots, X^{(590)})$ to monitor the production. At the end of the line, each of the 1567 produced entities is associated to a pass or fail output, with an average failure rate of $p_f = 6.6\%$. SIRUS outputs the following simple set of 6 rules:

Average failure rate $p_f = 6.6\%$					
if	$X^{(60)} < 5.51$	**then**	$p_f = 4.2\%$	**else**	$p_f = 16.6\%$
if	$X^{(104)} < -0.01$	**then**	$p_f = 3.9\%$	**else**	$p_f = 13.0\%$
if	$X^{(349)} < 0.04$	**then**	$p_f = 5.4\%$	**else**	$p_f = 17.8\%$
if	$X^{(206)} < 12.7$	**then**	$p_f = 5.4\%$	**else**	$p_f = 17.8\%$
if	$X^{(65)} < 26.1$	**then**	$p_f = 5.5\%$	**else**	$p_f = 17.2\%$
if	$X^{(60)} < 5.51$ & $X^{(349)} < 0.04$	**then**	$p_f = 3.6\%$	**else**	$p_f = 16.4\%$

To generate the prediction for a new query point $\mathbf{x}$, SIRUS checks for each rule whether the conditions are satisfied to assign one of the two possible p_f output values. Let us say for example that $x^{(60)} = 5.0$, then $\mathbf{x}$ satisfies the condition of the first rule, which returns $p_f = 4.2\%$. Next, the 6 rule outputs are averaged to provide the predicted probability of failure for $\mathbf{x}$. The model is stable: when a 10-fold cross-validation is run to simulate data perturbation, 4 to 5 rules are consistent across two folds in average. The predictive accuracy of SIRUS is similar to random forests whereas CART tree performs no better than the random classifier, as we will see for this dataset.

Section 2 is devoted to the detailed description of SIRUS. One of the main contributions of this work is the development of a software implementation, via the `R` package `sirus` (Benard and Wright, 2020) available from `CRAN`, based on `ranger`, a high-performance random forest implementation in `R` and `C++` (Wright and Ziegler, 2017). In Section 3, we illustrate the

efficiency of our procedure `sirus` through numerical experiments on real datasets. Then, in Section 4, we show that this good empirical behavior is theoretically understood by proving the asymptotic stability of SIRUS. Finally, Section 5 summarizes the main contributions of the article and provides directions for future research.

2 SIRUS Algorithm

Within the general framework of supervised (binary) classification, we assume to be given an i.i.d. sample $\mathscr{D}_n = \{(\mathbf{X}_i, Y_i), i = 1, \ldots, n\}$. Each $(\mathbf{X}_i, Y_i)$ is distributed as the generic pair $(\mathbf{X}, Y)$ independent of $\mathscr{D}_n$, where $\mathbf{X} = (X^{(1)}, \ldots, X^{(p)})$ is a random vector taking values in $\mathbb{R}^p$ and $Y \in \{0, 1\}$ is a binary response. Throughout the document, the distribution of $(\mathbf{X}, Y)$ is assumed to be unknown and is denoted by $\mathbb{P}_{\mathbf{X},Y}$. For $\mathbf{x} \in \mathbb{R}^p$, our goal is to accurately estimate the conditional probability $\eta(\mathbf{x}) = \mathbb{P}(Y = 1 | \mathbf{X} = \mathbf{x})$ with few simple and stable rules.

To tackle this problem, SIRUS first builds a (slightly modified) random forest. Next, each hyperrectangle of each tree of the forest is turned into a simple decision rule, and the collection of these elementary rules is ranked based on their frequency of appearance in the forest. Finally, the most significant rules are retained and are averaged together to form an ensemble model. We describe the four steps of SIRUS algorithm in the following paragraphs: the rule generation, rule selection, rule post-treatment, and the rule aggregation. This section ends with a discussion of SIRUS stability.

Rule generation. SIRUS uses at its core the random forest method (Breiman, 2001a), slightly modified for our purpose. As in the original procedure, each single tree in the forest is grown with a greedy heuristic that recursively partitions the input space using a random variable Θ. The essential difference between our approach and Breiman's one is that, prior to all tree constructions, the empirical q-quantiles of the marginal distributions over the whole dataset are computed: in each node of each tree, the best split can be selected among these empirical quantiles only. This constraint is critical to stabilize the forest structure and keeps almost intact the predictive accuracy, provided q is not too small (typically of the order of 10—see the experimental Subsection 3.2). Apart from this difference, the tree growing is similar to Breiman's original procedure. The tree randomization Θ is independent of the sample and has two independent components, denoted by $\Theta^{(S)}$ and $\Theta^{(V)}$, which are respectively used for the subsampling mechanism and randomization of the split direction. Throughout the manuscript, we let $\hat{q}_{n,r}^{(j)}$ be the empirical r-th q-quantile of $\{X_1^{(j)}, \ldots, X_n^{(j)}\}$, with typically $q = 10$. The construction of the individual trees is summarized in Algorithm 1 below.

The main step of SIRUS is to extract rules from the modified random forest. The cornerstone of this extraction mechanism is the notion of path in a decision tree. Indeed, a path describes the sequence of splits to go from the root of the tree to a specific (inner or terminal) node. Since a hyperrectangle is associated to each node, a rule can be defined as a piecewise constant estimate with this hyperrectangle as support. Therefore, to rigorously define the rule extraction, we introduce the symbolic representation of a path in a tree. We insist that such definition is valid for both terminal leaves and inner nodes, which are all used by SIRUS. To begin, we follow the example shown in Figure 1 with a tree of depth 2 partitioning the

Algorithm 1 Tree construction

1: **Parameters:** Number of quantiles q, number of subsampled observations a_n, number of eligible directions for splitting `mtry`.
2: Compute the empirical q-quantiles for each marginal distribution over the whole dataset.
3: Subsample with replacement a_n observations, indexed by $\Theta^{(S)}$. Only these observations are used to build the tree.
4: Initialize the cell H as the root of the tree.
5: Draw uniformly at random a subset $\Theta^{(V)} \subset \{1, \ldots, p\}$ of cardinality `mtry`.
6: For all $j \in \Theta^{(V)}$, compute the CART-splitting criterion at all empirical q-quantiles of $X^{(j)}$ that split the cell H into two non-empty cells.
7: Choose the split that maximizes the CART-splitting criterion.
8: Recursively repeat **lines** $5-7$ for the two resulting children cells H_L and H_R.

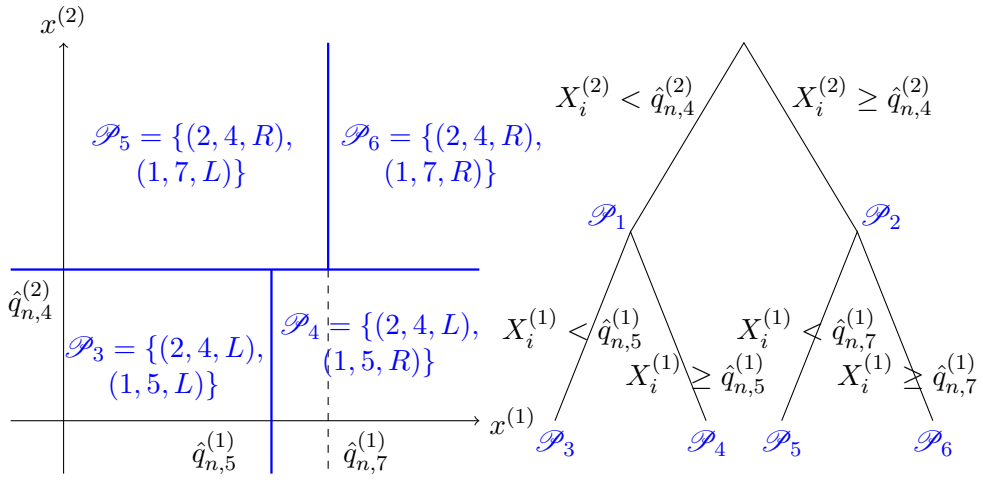

Figure 1: Example of a root node $\mathbb{R}^2$ partitionned by a randomized tree of depth 2: the tree on the right side, the associated paths and hyperrectangles of length $d = 2$ on the left side.

input space $\mathbb{R}^2$. For instance, let us consider the node $\mathscr{P}_6$ defined by the sequence of two splits $X_i^{(2)} \geq \hat{q}^{(2)}_{n,4}$ and $X_i^{(1)} \geq \hat{q}^{(1)}_{n,7}$. The first split is symbolized by the triplet $(2, 4, R)$, whose components respectively stand for the variable index 2, the quantile index 4, and the right side R of the split. Similarly, for the second split we cut coordinate 1 at quantile index 7, and pass to the right. Thus, the path to the considered node is defined by $\mathscr{P}_6 = \{(2, 4, R), (1, 7, R)\}$. Also notice that the first split already defines the path $\mathscr{P}_2 = \{(2, 4, R)\}$, associated to the right inner node at the first level of the tree. Of course, this generalizes to each path $\mathscr{P}$ of length d under the symbolic compact form

$$\mathscr{P} = \{(j_k, r_k, s_k),\ k = 1, \ldots, d\},$$

where, for $k \in \{1, \ldots, d\}$, the triplet (j_k, r_k, s_k) describes how to move from level $(k-1)$ to level k, with a split using the coordinate $j_k \in \{1, \ldots, p\}$, the index $r_k \in \{1, \ldots, q-1\}$ of the corresponding quantile, and a side $s_k = L$ if we go the the left and $s_k = R$ if we go to the right. The set of all possible such paths is denoted by Π. It is important to note that Π is in fact a deterministic (that is, non random) quantity, which only depends upon the

dimension p and the order q of the quantiles. Of course, given a path $\mathscr{P} \in \Pi$ one can recover the hyperrectangle (i.e., the tree node) $\hat{H}_n(\mathscr{P})$ associated with $\mathscr{P}$ and the entire dataset $\mathscr{D}_n$ via the correspondence

$$\hat{H}_n(\mathscr{P}) = \left\{ \mathbf{x} \in \mathbb{R}^p : \begin{cases} \mathbf{x}^{(j_k)} < \hat{q}_{n,r_k}^{(j_k)} & \text{if } s_k = L \\ \mathbf{x}^{(j_k)} \geq \hat{q}_{n,r_k}^{(j_k)} & \text{if } s_k = R \end{cases}, k = 1, \ldots, d \right\}. \tag{2.1}$$

Finally, an elementary rule $\hat{g}_{n,\mathscr{P}}$ can be defined from $\hat{H}_n(\mathscr{P})$ as a piecewise constant estimate: $\hat{g}_{n,\mathscr{P}}(\mathbf{x})$ returns the empirical probability that the output Y is of class 1 conditional on whether the query point $\mathbf{x}$ belongs to $\hat{H}_n(\mathscr{P})$ or not. Thus, the rule $\hat{g}_{n,\mathscr{P}}$ associated to the path $\mathscr{P} \in \Pi$ is formally defined by

$$\forall \mathbf{x} \in \mathbb{R}^p, \quad \hat{g}_{n,\mathscr{P}}(\mathbf{x}) = \begin{cases} \frac{1}{N_n(\hat{H}_n(\mathscr{P}))} \sum_{i=1}^n Y_i \mathbb{1}_{\mathbf{X}_i \in \hat{H}_n(\mathscr{P})} & \text{if } \mathbf{x} \in \hat{H}_n(\mathscr{P}) \\ \frac{1}{n - N_n(\hat{H}_n(\mathscr{P}))} \sum_{i=1}^n Y_i \mathbb{1}_{\mathbf{X}_i \notin \hat{H}_n(\mathscr{P})} & \text{otherwise} \end{cases},$$

using the convention $0/0 = 0$, and where $N_n(\hat{H}_n(\mathscr{P}))$ is the number of observations in the node associated with $\mathscr{P}$. This formal definition can be illustrated with the SECOM dataset presented in the introduction. For the first rule, since $\hat{q}_{n,8}^{(60)} = 5.51$, the corresponding path is $\mathscr{P} = \{(60, 8, L)\}$, and the associated rule is

$$\hat{g}_{n,\mathscr{P}}(\mathbf{x}) = \begin{cases} 0.042 & \text{if } x^{(60)} < 5.51 \\ 0.166 & \text{if } x^{(60)} \geq 5.51 \end{cases}.$$

Finally, a Θ-random tree generates a collection of paths in Π, one for each internal and terminal nodes. In the sequel, we let $T(\Theta, \mathscr{D}_n)$ be the list of such extracted paths, a random subset of Π.

Rule selection. Using our modified random forest algorithm, we are able to generate a large number M of trees, randomized by $\Theta_1, \ldots, \Theta_M$, i.i.d. copies of the generic variable Θ, and then to extract a large collection of rules. Since we are interested in selecting the most important rules, i.e., those which represent strong patterns between the inputs and the output, we select rules that are shared by a large portion of trees. Such occurrence frequency is formally defined by

$$\hat{p}_{M,n}(\mathscr{P}) = \frac{1}{M} \sum_{\ell=1}^{M} \mathbb{1}_{\mathscr{P} \in T(\Theta_\ell, \mathscr{D}_n)},$$

which is the Monte-Carlo estimate of the probability that a path $\mathscr{P}$ belongs to a Θ-random tree, that is

$$p_n(\mathscr{P}) = \mathbb{P}(\mathscr{P} \in T(\Theta, \mathscr{D}_n) | \mathscr{D}_n).$$

As a general strategy, once the modified random forest has been built, we draw the list of all paths that appear in the forest and only retain those that occur with a frequency larger than the threshold $p_0 \in (0, 1)$, the only influential parameter of SIRUS—see Subsection 3.3 for its tuning procedure. We are thus interested in the set of the extracted paths

$$\hat{\mathscr{P}}_{M,n,p_0} = \{\mathscr{P} \in \Pi : \hat{p}_{M,n}(\mathscr{P}) > p_0\}. \tag{2.2}$$

An important feature of SIRUS algorithm is to stop the growing of the forest with an appropriate number of trees M. Although the right order of magnitude for M is required, no fine tuning is necessary. Indeed, the uncertainty of the importance estimate $\hat{p}_{M,n}(\mathscr{P})$ of each rule decreases with M, whereas the computational cost linearly increases with M. Thus, to obtain a robust rule extraction, M needs to be high enough to make the uncertainty of $\hat{p}_{M,n}(\mathscr{P})$ negligible. More precisely, M is set to get the same list of selected rules $\hat{\mathscr{P}}_{M,n,p_0}$ when SIRUS is run multiple times on the same dataset $\mathscr{D}_n$. On the other hand, M should be small enough to avoid useless computations. Therefore, the growing of the forest is automatically stopped when 95% of the selected rules would be shared by a new run of SIRUS on $\mathscr{D}_n$ in average, as it is possible to derive a simple stopping criterion based on the properties of the estimates $\hat{p}_{M,n}(\mathscr{P})$—all the technical details are provided in Appendix B. A random forest is usually built with around 500 trees, as the predictive accuracy cannot be significantly increased by adding more trees. SIRUS typically grows 10 times more trees to obtain a robust rule extraction.

Besides, we insist that the quantile discretization is critical for the rule selection. The expected value of the rule importance is

$$\mathbb{E}[\hat{p}_{M,n}(\mathscr{P})] = \mathbb{P}(\mathscr{P} \in T(\Theta, \mathscr{D}_n)),$$

but without the discretization, the list of extracted paths from a random tree $T(\Theta, \mathscr{D}_n)$ takes values in an uncountable space when at least one component of $\mathbf{X}$ is a continuous random variable, and therefore the above quantity is null, making the path selection procedure unstable with respect to data perturbation.

Rule post-treatment. By construction, there is some redundancy in the list of rules generated by the set of distinct paths $\hat{\mathscr{P}}_{M,n,p_0}$. The hyperrectangles associated with the paths extracted from a Θ-random tree overlap, and so the corresponding rules are linearly dependent. Therefore a post-treatment to filter $\hat{\mathscr{P}}_{M,n,p_0}$ is needed to remove redundancy and obtain a compact rule model. The general idea is straightforward: if the rule associated with the path $\mathscr{P} \in \hat{\mathscr{P}}_{M,n,p_0}$ is a linear combination of rules associated with paths with a higher frequency in the forest, then $\mathscr{P}$ is removed from $\hat{\mathscr{P}}_{M,n,p_0}$.

To illustrate the post-treatment, let the tree of Figure 1 be the Θ_1-random tree grown in the forest. Since the paths of the first level of the tree, $\mathscr{P}_1$ and $\mathscr{P}_2$, always occur in the same trees, we have $\hat{p}_{M,n}(\mathscr{P}_1) = \hat{p}_{M,n}(\mathscr{P}_2)$. If we assume these quantities to be greater than p_0, then $\mathscr{P}_1$ and $\mathscr{P}_2$ both belong to $\hat{\mathscr{P}}_{M,n,p_0}$. However, by construction, $\mathscr{P}_1$ and $\mathscr{P}_2$ are associated with the same rule, and we therefore enforce SIRUS to keep only $\mathscr{P}_1$ in $\hat{\mathscr{P}}_{M,n,p_0}$. Each of the paths of the second level of the tree, $\mathscr{P}_3$, $\mathscr{P}_4$, $\mathscr{P}_5$, and $\mathscr{P}_6$, can occur in many different trees, and their associated $\hat{p}_{M,n}$ are distinct (except in very specific cases). Assume for example that $\hat{p}_{M,n}(\mathscr{P}_1) > \hat{p}_{M,n}(\mathscr{P}_4) > \hat{p}_{M,n}(\mathscr{P}_5) > \hat{p}_{M,n}(\mathscr{P}_3) > \hat{p}_{M,n}(\mathscr{P}_6) > p_0$. Since $\hat{g}_{n,\mathscr{P}_3}$ is a linear combination of $\hat{g}_{n,\mathscr{P}_4}$ and $\hat{g}_{n,\mathscr{P}_1}$, $\mathscr{P}_3$ is removed. Similarly $\mathscr{P}_6$ is redundant with $\mathscr{P}_1$ and $\mathscr{P}_5$, and it is therefore removed. Finally, among the six paths of the tree, only $\mathscr{P}_1$, $\mathscr{P}_4$, and $\mathscr{P}_5$ are kept in the list $\hat{\mathscr{P}}_{M,n,p_0}$.

Rule aggregation. Now, the resulting small set of rules $\hat{\mathscr{P}}_{M,n,p_0}$ is combined to form a simple, compact, and stable rule classification model. We simply average the set of elementary

rules $\{\hat{g}_{n,\mathscr{P}} : \mathscr{P} \in \hat{\mathscr{P}}_{M,n,p_0}\}$ that have been selected in the first steps of SIRUS. The aggregated estimate $\hat{\eta}_{M,n,p_0}(\mathbf{x})$ of $\eta(\mathbf{x})$ is thus defined by

$$\hat{\eta}_{M,n,p_0}(\mathbf{x}) = \frac{1}{|\hat{\mathscr{P}}_{M,n,p_0}|} \sum_{\mathscr{P} \in \hat{\mathscr{P}}_{M,n,p_0}} \hat{g}_{n,\mathscr{P}}(\mathbf{x}). \tag{2.3}$$

Finally, the classification procedure assigns class 1 to an input $\mathbf{x}$ if the aggregated estimate $\hat{\eta}_{M,n,p_0}(\mathbf{x})$ is above a given threshold, and class 0 otherwise. In the introduction, we presented an example of a list of 6 rules for the SECOM dataset. In this case, for a new input $\mathbf{x}$, $\hat{\eta}_{M,n,p_0}(\mathbf{x})$ is simply the average of the output p_f over the 6 selected rules.

In past works on rule ensemble models, such as RuleFit (Friedman and Popescu, 2008) and Node harvest (Meinshausen, 2010), rules are also extracted from a tree ensemble and then combined together through a regularized linear model. In our case, it happens that the parameter p_0 alone is enough to control sparsity. Indeed, in our experiments, we observe that adding such linear model in the aggregation method hardly increases the accuracy and hardly reduces the size of the final rule set, while it can significantly reduce stability, add a set of coefficients that makes the model less straightforward to interpret, and requires more intensive computations. We refer to the experiments in Appendix C for a comparison between $\hat{\eta}_{M,n,p_0}$ defined a as simple average (2.3) versus a definition with a logistic regression.

Stability. The three main properties to assess the interpretability of SIRUS are simplicity, stability, and predictivity, as already stated. On one hand, a measure of simplicity is naturally provided by the number of rules, and predictivity is given by the missclassification rate or the AUC. On the other hand, stability requires a more thorough discussion. In the statistical learning theory, stability refers to the stability of predictions (e.g., Vapnik, 1998). In particular, Rogers and Wagner (1978), Devroye and Wagner (1979), Bousquet and Elisseeff (2002), and Poggio et al. (2004) show that stability and predictive accuracy are closely connected. In our case, we are more concerned by the stability of the internal structure of the model, and, to our knowledge, no general definition exists. So, we state the following tentative definition: a rule learning algorithm is stable if two independent estimations based on two independent samples result in two similar lists of rules. Thus, given a new sample $\mathscr{D}'_n$ independent of $\mathscr{D}_n$, we define $\hat{p}'_{M,n}(\mathscr{P})$ and the corresponding set of paths $\hat{\mathscr{P}}'_{M,n,p_0}$ based on a modified random forest drawn with a parameter Θ' independent of Θ. Then, we measure the stability of SIRUS by the proportion of rules shared by the two sets $\hat{\mathscr{P}}_{M,n,p_0}$ and $\hat{\mathscr{P}}'_{M,n,p_0}$, selected over these two runs of SIRUS on independent samples. We take advantage of a dissimilarity measure between two sets, the so-called Dice-Sorensen index, often used to assess the stability of variable selection methods (Chao et al., 2006; Zucknick et al., 2008; Boulesteix and Slawski, 2009; He and Yu, 2010; Alelyani et al., 2011). This index is defined by

$$\hat{S}_{M,n,p_0} = \frac{2\big|\hat{\mathscr{P}}_{M,n,p_0} \cap \hat{\mathscr{P}}'_{M,n,p_0}\big|}{\big|\hat{\mathscr{P}}_{M,n,p_0}\big| + \big|\hat{\mathscr{P}}'_{M,n,p_0}\big|} \tag{2.4}$$

with the convention $0/0 = 1$. This is a measure of stability taking values between 0 and 1: if the intersection between $\hat{\mathscr{P}}_{M,n,p_0}$ and $\hat{\mathscr{P}}'_{M,n,p_0}$ is empty, then $\hat{S}_{M,n,p_0} = 0$, while if $\hat{\mathscr{P}}_{M,n,p_0} = \hat{\mathscr{P}}'_{M,n,p_0}$, then $\hat{S}_{M,n,p_0} = 1$. Notice that it is possible to use other metrics to assess

the distance between two finite sets (Zucknick et al., 2008): the Jaccard Index is another popular example. Although the stability values slightly vary with a different definition, both the empirical stability comparisons between algorithms—see Section 3—and the asymptotic stability of SIRUS—see Section 4—are insensitive to the stability metric choice.

3 Experiments

We begin this experimental section by providing additional details on the example given in the introduction with the semi-conductor manufacturing process data. This example shows the excellent performance of SIRUS on real data in a noisy and high-dimensional setting. Next, we use 9 UCI datasets (Dua and Graff, 2017) to perform extensive comparisons between SIRUS and its main competitors. We show that SIRUS produces much more stable rule lists, while preserving a predictive accuracy and computational complexity comparable to the top competitors. Such performance is reached with a tuning of the parameter p_0, and the details of this tuning procedure are provided in the third subsection. Finally, a thorough discussion on the design of SIRUS is conducted in the last subsection. In particular, the cut limitations to the quantiles and the number of constraints in the selected rules are analyzed.

We first introduce relevant metrics to assess the three interpretability properties in the experiments. By definition, the size (i.e., the simplicity) of the rule ensemble is the number of selected rules, i.e., $|\hat{\mathscr{P}}_{M,n,p_0}|$. To measure the error, 1-AUC is used and estimated by 10-fold cross-validation (repeated 30 times for robustness). With respect to stability, an independent dataset is not available for real data to compute $\hat{S}_{M,n,p_0}$ as defined in (2.4) in the previous section. Nonetheless, we can take advantage of the cross-validation process to compute a stability metric: the proportion of rules shared by two models built during the cross-validation, averaged over all possible pairs (Guidotti and Ruggieri, 2019).

3.1 Manufacturing Process Data

SIRUS is run on a real manufacturing process of semi-conductors, the SECOM dataset (Dua and Graff, 2017). Data is collected from sensors and process measurement points to monitor the production line, resulting in 590 numeric variables. Each of the 1567 data points represents a single production entity associated with a pass or fail output (0/1) for in-house line testing. As it is often the case for a production process, the dataset is unbalanced and contains 104 fails, i.e., a failure rate p_f of 6.6%. We proceed to a simple pre-processing of the data: missing values (about 5% of the total) are replaced by the median.

Figure 2 shows predictivity versus the number of rules when p_0 varies, with the optimal p_0 displayed. The 1-AUC value is 0.30 for SIRUS (for the optimal $p_0 = 0.04$), 0.29 for Breiman's random forests, and 0.48 for a pruned CART tree. Thus, in that case, CART tree predicts no better than the random classifier, whereas SIRUS has a similar accuracy to random forests. The final model has 6 rules and a stability of 0.74, i.e., in average 4 to 5 rules are shared by 2 models built in a 10-fold cross-validation process, simulating data perturbation. By comparison, Node harvest outputs 36 rules with a value of 0.32 for 1-AUC.

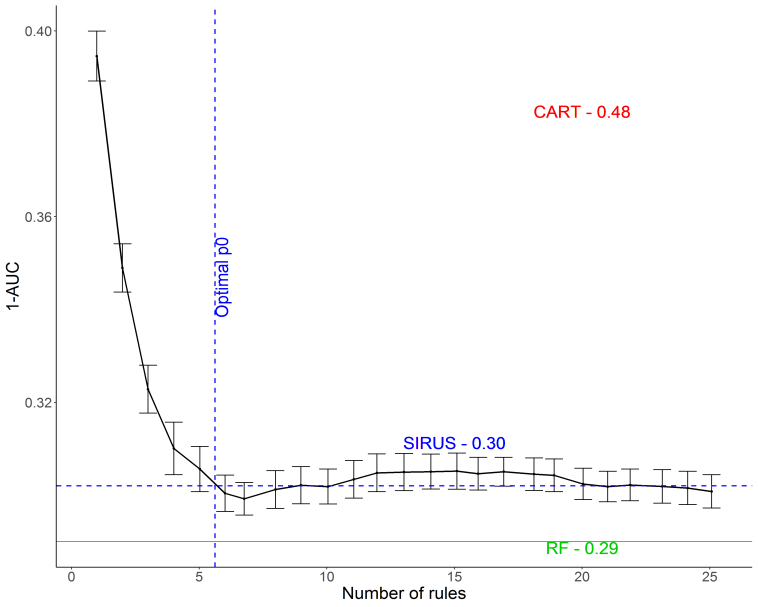

Figure 2: For the SECOM dataset, error (1-AUC) versus the number of rules when p_0 varies, estimated via 10-fold cross-validation (averaged over 30 repetitions of the cross-validation). Errors for CART and random forests are reported for comparisons.

```
"Proportion of class 1 = 0.0664 - Sample size n = 1567"
"if V60 < 5.51 then 0.0415 (n=1253) else 0.166 (n=314)"
"if V104 < -0.00868 then 0.0392 (n=1097) else 0.13 (n=470)"
"if V349 < 0.0356 then 0.0539 (n=1410) else 0.178 (n=157)"
"if V206 < 12.7 then 0.0539 (n=1410) else 0.178 (n=157)"
"if V65 < 26.1 then 0.0546 (n=1410) else 0.172 (n=157)"
"if V60 < 5.51 & V349 < 0.0356 then 0.0346 (n=1184) else 0.164 (n=383)"
```

Figure 3: List of rules output by our software `sirus` in the R console for the SECOM dataset.

Finally, the output of SIRUS may be displayed in the simple and interpretable form of Figure 3, the output in the `R` console of the package `sirus` for the SECOM data. Such a rule model enables to catch immediately how the most relevant variables impact failures. Among the 590 variables, 5 are enough to build a model as predictive as random forests, and such a selection is robust. Other rules alone may also be informative, but they do not add additional information to the model, since predictive accuracy is already minimal with the 6 selected rules. Then, production engineers should first focus on those 6 rules to investigate an improved setting of the production process. We insist that the stability of the output rule list is critical in practice. Indeed, the algorithm may be run multiple times during the analysis, eventually with an additional small new batch of data. The output rule list should be quite insensitive to such perturbation: domain experts are skeptical of unstable results, which are the symptoms of a partial and arbitrary modelling of the true phenomenon. SIRUS is stable, but it is not the case for decision trees or existing rule algorithms, as we show in the next

Dataset	Sample size	Total number of variables	Number of categorical variables
Haberman	306	3	0
Diabetes	768	8	0
Heart Statlog	270	13	6
Liver Disorders	345	6	0
Heart C2	303	13	8
Heart H2	294	13	7
Credit German	1000	20	13
Credit Approval	690	15	9
Ionosphere	351	33	0

Table 1: Description of UCI datasets

subsection and illustrate in the first section of the Supplementary Material.

3.2 Improvement over Competitors

We have conducted experiments on 9 diverse public datasets from the UCI repository (Dua and Graff, 2017; data is described in Table 1). This batch of experiments aims at illustrating the good behavior of SIRUS over its competitors in various settings. Overall, we observe that SIRUS provides a high improvement of stability compared to state-of-the-art rule algorithms. More precisely, SIRUS has two types of competitors: decision trees and rule algorithms. The latter can further be split into three different kinds: those based on tree ensembles, classical rule algorithms based on greedy heuristics, and those built on top of frequent pattern mining algorithms. To compare stability of the different methods, data is discretized using the 10-empirical quantiles for each variable and the same stability metric is used for all algorithm comparisons. For simplicity and predictivity metrics, we do not apply this pre-processing step of discretization, unless the algorithm only handles categorical data. For the top competitors, experimental results are gathered in Table 2 for model size, Table 3 for stability, and Table 4 for predictive accuracy. Experiments for additional competitors are provided in Appendix A in Tables 7, 8, and 9. In the following five paragraphs, we analyze the experiment results of SIRUS and its four groups of competitors: decision trees, tree ensemble rule algorithms, classical rule algorithms, and frequent pattern mining methods.

SIRUS. We use our `R`/`C++` software implementation `sirus` (Benard and Wright, 2020) (available from `CRAN`), adapted from `ranger`, a fast random forest implementation (Wright and Ziegler, 2017). Besides, notice that categorical variables are transformed in multiple binary variables, and that we use the default settings of random forests, well known for their excellent behavior, in particular $\texttt{mtry} = \lfloor \frac{p}{3} \rfloor$. We set $q = 10$ quantiles and tune p_0 as specified in Subsection 3.3. Figure 4 provides an example for the dataset "Credit German" of the dependence between predictivity and the number of rules when p_0 varies. In that case, the minimum of 1-AUC is about 0.26 for SIRUS, 0.21 for Breiman's forests, and 0.29 for CART tree. For the chosen p_0, SIRUS returns a compact set of 18 rules and its stability is 0.66, i.e., about 12

	Decision tree	Classical rule learning	Frequent pattern mining		Tree ensemble		
Dataset	**CART**	**RIPPER**	**CBA**	**BRL**	**RuleFit**	**Node harvest**	**SIRUS**
Haberman	2.1	1.0	2	2.2	1.6	25.2	3.3
Diabetes	13.6	2.6	22.6	5.6	26.1	37.8	7.9
Heart Statlog	9.6	2.9	27.1	3.5	18.2	21.6	10.7
Liver Disorders	14.4	3.4	2	2.9	17.3	33.7	18.5
Heart C2	9.8	3.6	44.9	3.8	22.6	33.8	22.5
Heart H2	4.3	2.1	25.8	3.1	11.31	23.3	12.6
Credit German	16.2	3.1	75.9	3.6	30.4	34.3	17.8
Credit Approval	4.3	3.1	55.7	4	15.4	26.8	19.7
Ionosphere	4.1	4.2	37.8	4.4	18.3	28.8	21.4

Table 2: Mean model size over a 10-fold cross-validation for UCI datasets. Results are averaged over 30 repetitions of the cross-validation. (Standard deviations are negligible, they are not displayed to increase readability.)

	Decision tree	Classical rule learning	Frequent pattern mining		Tree ensemble		
Dataset	**CART**	**RIPPER**	**CBA**	**BRL**	**RuleFit**	**Node harvest**	**SIRUS**
Haberman	0.01	0.10	**0.80**	0.49	0.57	0.34	0.65
Diabetes	0.25	0.18	0.46	**0.75**	0.21	0.43	**0.79**
Heart Statlog	0.23	0.30	0.36	**0.69**	0.18	0.56	**0.69**
Liver Disorders	0.18	0.08	0.51	0.50	0.19	0.28	**0.57**
Heart C2	0.20	0.21	0.37	0.57	0.28	0.51	**0.66**
Heart H2	0.34	0.29	**0.68**	**0.68**	0.23	0.42	**0.65**
Credit German	0.40	0.21	0.50	0.46	0.12	0.49	**0.66**
Credit Approval	0.49	0.31	0.43	0.52	0.17	0.23	**0.66**
Ionosphere	**0.93**	0.33	0.14	0.66	0.06	0.52	0.63

Table 3: Mean stability over a 10-fold cross-validation for UCI datasets. Results are averaged over 30 repetitions of the cross-validation. (Standard deviations are negligible, they are not displayed to increase readability. Values within 10% of the maximum are displayed in bold.)

	Black box	Decision tree	Classical rule learning	Frequent pattern mining		Tree ensemble		
Dataset	**Random Forest**	**CART**	**RIPPER**	**CBA**	**BRL**	**RuleFit**	**Node harvest**	**SIRUS**
Haberman	0.32	0.49	0.41	0.45	0.42	**0.35**	**0.34**	**0.36**
Diabetes	0.17	0.25	0.29	0.31	0.26	**0.19**	**0.19**	**0.20**
Heart Statlog	0.10	0.20	0.22	0.18	0.23	**0.13**	**0.13**	**0.12**
Liver Disorders	0.23	0.36	0.36	0.47	0.43	**0.27**	0.32	0.36
Heart C2	0.10	0.20	0.26	0.20	0.22	**0.11**	**0.12**	**0.12**
Heart H2	0.12	0.23	0.23	0.24	0.17	**0.11**	**0.11**	**0.12**
Credit German	0.21	0.29	0.38	0.39	0.33	**0.23**	**0.26**	**0.26**
Credit Approval	0.07	0.14	0.15	0.15	0.11	**0.07**	**0.07**	0.10
Ionosphere	0.03	0.11	0.12	0.13	0.10	**0.04**	0.07	0.06

Table 4: Model error (1-AUC) over a 10-fold cross-validation for UCI datasets. Results are averaged over 30 repetitions of the cross-validation. (Standard deviations are negligible, they are not displayed to increase readability. Values within 10% of the maximum are displayed in bold, random forest is put aside.)

rules are consistent between two different models built in a 10-fold cross-validation. Thus, the final model is simple (a set of only 18 rules), is quite robust to data perturbation, and has a predictive accuracy close to random forests. Figure 5 provides another example of the good practical performance of SIRUS with the "Heart Statlog" dataset. Here, the predictivity of random forests is reached with 11 rules, with a stability of 0.69.

Decision trees. Decision trees may be the most popular competitors of SIRUS because of their simple structure. Indeed, their simplicity—measured by the number of nodes—is comparable to SIRUS. The main algorithms are CART (Breiman et al., 1984) and C5.0 (Quinlan, 1992). However, as Breiman originally observes (Breiman, 2001b), by only randomly removing $2-3\%$ of the data, the tree structure can be quite different. Our experiments in Table 3 corroborate this unstable behavior. Additionally, decision trees are not as predictive as SIRUS. We use available R implementations, respectively `rpart` (Therneau and Atkinson, 2019) and `C50` (Kuhn and Quinlan, 2020), and trees are pruned. Notice that, to enable simplicity and stability comparisons for CART, a list of rules is extracted from its nodes, as it is originally possible for C5.0. Overall, SIRUS produces more stable and predictive rule lists than decision trees, for a comparable simplicity, but at the price of a higher computational complexity since many trees are grown.

Tree ensemble rule algorithms. RuleFit and Node harvest algorithms are close to SIRUS since they both extract rules from a tree ensemble. More specifically, RuleFit extracts all the rules of a boosted tree ensemble (Friedman and Popescu, 2003), while Node harvest is based on random forests. Then, the extracted rules are linearly combined in a sparse linear model, respectively a logistic regression with a Lasso penalty (Tibshirani, 1996) for RuleFit,

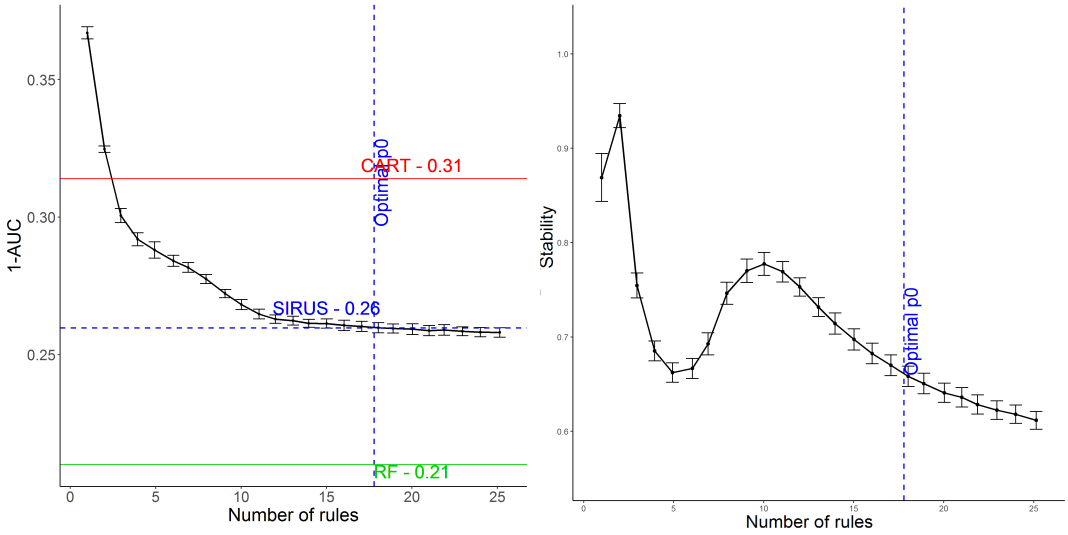

Figure 4: For the UCI dataset "Credit German", 1-AUC (on the left) and stability (on the right) versus the number of rules when p_0 varies, estimated via 10-fold cross-validation (results are averaged over 30 repetitions).

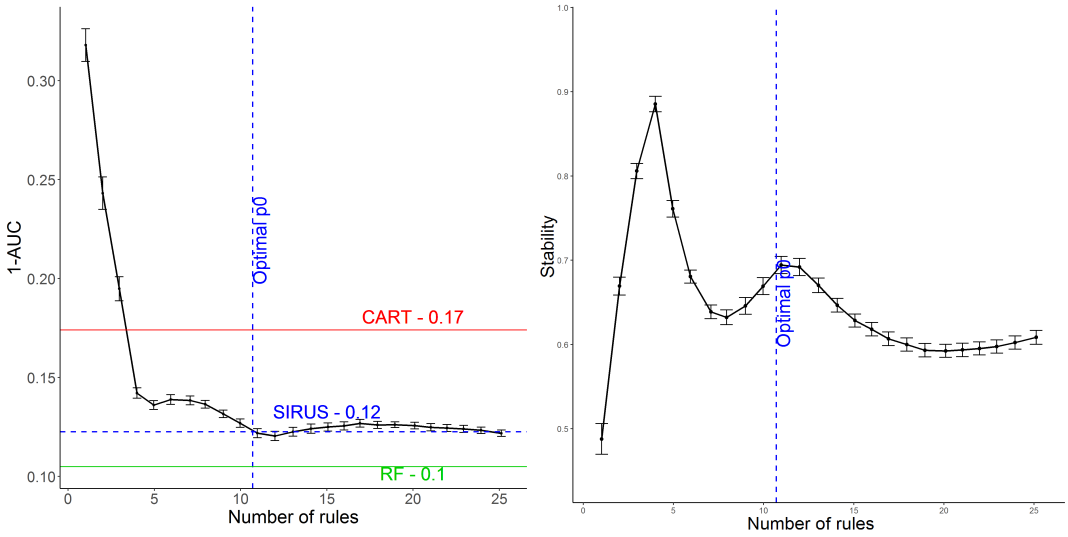

Figure 5: For the UCI dataset "Heart Statlog", 1-AUC (on the left) and stability (on the right) versus the number of rules when p_0 varies, estimated via 10-fold cross-validation (results are averaged over 30 repetitions).

and a constraint quadratic linear program for Node harvest. These two methods have an accuracy and computational complexity comparable to random forests and SIRUS, since the main step of all these algorithms is to grow a tree ensemble with a large number of trees. However, both algorithms are unstable, and both output quite complex and long lists of rules. Even running RuleFit or Node harvest multiple times on the same dataset produces quite different rule lists because of the randomness in the tree ensembles—see the first section of the Supplementary Material. On the other hand, SIRUS is built to have its structure converged for the given dataset, as explained in Section 2 and Appendix B. Consequently, SIRUS is more stable than RuleFit and Node harvest, and produces shorter rule lists. We use available R implementations, `pre` (Fokkema, 2020, RuleFit) and `nodeharvest` (Meinshausen, 2015). Note that categorical features are transformed in multiple binary variables as it is required by the software implementations, and RuleFit is limited to rule predictors. For RuleFit, the lasso penalty is tuned by cross-validation as defined in Friedman and Popescu (2008). As advertised in Meinshausen (2010), Node harvest is tuning parameter free by default, but it is also possible to add a regularization term to reduce the model size. We use the same tuning procedure as for SIRUS to maximize accuracy with the smallest possible model—see Subsection 3.3. Overall, SIRUS produces more stable and shorter rule lists than RuleFit and Node harvest, for a comparable accuracy and computational complexity.

Classical rule algorithms. Classical rule algorithms, typically RIPPER (Cohen, 1995), PART (Frank and Witten, 1998), and FOIL (Quinlan and Cameron-Jones, 1995) are based on greedy heuristics and exhibit similar properties as decision trees: a smaller computational complexity, but a high instability and a reduced predictivity. Simplicity varies across algorithms. We use available R implementations: `RWeka` (Hornik et al., 2009, RIPPER, PART) and `arulesCBA` (Johnson and Hahsler, 2020, FOIL).

Frequent pattern mining. A last type of rule algorithms is built on top of frequent pattern mining algorithms, originally used to identify frequent occurrences in database mining. Indeed, since the output $Y \in \{0, 1\}$ is discrete and the input data is discretized with quantiles, we can generate candidate rules for classification by identifying frequent patterns associated with each output label. This search for association rules is computationally costly (exponential with p and q), and efficient heuristics are used, essentially Apriori (Agrawal et al., 1993) and Eclat (Zaki et al., 1997). The rule aggregation mechanism is specific to each algorithm. Liu et al. (1998) first use this principle to introduce CBA (Classification Based on Association Rules). Later, BRL (Bayesian Rule List, Letham et al., 2015) uses a more sophisticated Bayesian framework for the rule aggregation. Interestingly, these methods exhibit quite good stability properties, higher than decision trees, classical rule algorithms, RuleFit, and Node harvest. On the other hand, their predictive accuracy is worse than decision trees. Experiments in Tables 2, 3, and 4 show that SIRUS exhibits a high stability and predictivity improvement over these methods. Besides, simplicity varies across algorithms: CBA produces much longer rule lists than SIRUS, whereas BRL generates shorter models. We use available implementations: `arulesCBA` (Johnson and Hahsler, 2020, CBA) and `sbrl` (Yang et al., 2017, BRL). Note that we use default settings for BRL, since modifying its parameters does not significantly improve accuracy and can hurt stability.

3.3 Tuning of SIRUS

SIRUS relies on a single hyperparameter: the selection threshold p_0 involved in the definition of $\hat{\mathscr{P}}_{M,n,p_0}$ to filter the most important rules, and which therefore determines the simplicity of the model. This parameter p_0 should be set to optimize a tradeoff between the number of rules, stability, and accuracy. In practice, it is difficult to settle such a criterion, and we choose to optimize p_0 to maximize the predictive accuracy with the smallest possible set of rules. To achieve this goal, we proceed as follows. The error 1-AUC is estimated by 10-fold cross-validation for a fine grid of p_0 values, defined such that $|\hat{\mathscr{P}}_{M,n,p_0}|$ varies from 1 to 25 rules. (We let 25 be an arbitrary upper bound on the maximum number of rules, considering that a bigger set is not readable anymore.) The randomization introduced by the partition of the dataset in the 10 folds of the cross-validation process has a significant impact on the variability of the size of the final model. Therefore, in order to get a robust estimation of p_0, the cross-validation is repeated multiple times (typically 30) and results are averaged. The standard deviation of the mean of 1-AUC is computed over these repetitions for each p_0 of the grid search. We consider that all models within 2 standard deviations of the minimum of 1-AUC are not significantly less predictive than the optimal one. Thus, among these models, the one with the smallest number of rules is selected, i.e., the optimal p_0 is shifted towards higher values to reduce the model size without decreasing predictivity—see Figures 4 and 5 for examples. This approach is very similar to the tuning procedure of the Lasso (Tibshirani, 1996).

3.4 Discussion on SIRUS Design

Quantile discretization. In the modified random forest grown in the first step of SIRUS, the split at each tree node is limited to the empirical q-quantiles of each component of $\mathbf{X}$, as described in Section 2. Thus, we check that this modification alone of the forest has little impact on its accuracy. Using the R package `ranger`, 1-AUC is estimated for each dataset with 10-fold cross-validation for $q = 10$. Results are averaged over 10 repetitions of the cross-validation and displayed in Table 5, with standard deviations in parentheses. Clearly, the decrease of accuracy generated by this discretization is negligible. In fact, this result is not surprising: with only $p = 10$ input variables, such quantile discretization splits the input space in a fine grid of 10^{10} hyperrectangles, providing a high flexibility to the modified random forest to identify local patterns.

Tree depth. When SIRUS is fit using fully grown trees, the final set of rules $\hat{\mathscr{P}}_{M,n,p_0}$ contains almost exclusively rules made of one or two splits, and rarely of three splits. Although this may appear surprising at first glance, this phenomenon is in fact expected. Indeed, rules made of multiple splits are extracted from deeper tree levels and are thus more sensitive to data perturbation by construction. This results in much smaller values of $\hat{p}_{M,n}(\mathscr{P})$ for rules with a high number of splits, and then deletion from the final set of path through the threshold p_0: $\hat{\mathscr{P}}_{M,n,p_0} = \{\mathscr{P} \in \Pi : \hat{p}_{M,n}(\mathscr{P}) > p_0\}$. To illustrate this, let us consider the following typical example with $p = 100$ input variables and $q = 10$ quantiles. There are $qp = 100 \times 10 = 10^3$ possible splits at the root node of a tree, and then $2pq = 2.10^3$ paths of one split. Since the left and right paths of one split at the root node are associated to the same rule, there are

Dataset	Breiman's RF	RF - limited splits ($q = 10$)
haberman	0.32 (0.006)	0.33 (0.01)
diabetes	0.17 (0.003)	0.18 (0.003)
heart statlog	0.10 (0.006)	0.10 (0.006)
liver disorders	0.22 (0.01)	0.25 (0.007)
heart C2	0.10 (0.003)	0.10 (0.004)
heart H2	0.12 (0.005)	0.12 (0.004)
credit german	0.21 (0.003)	0.21 (0.004)
credit approval	0.070 (0.002)	0.071 (0.002)
ionosphere	0.025 (0.002)	0.027 (0.002)

Table 5: Accuracy, measured by 1-AUC (standard deviation) on UCI datasets, for two algorithms: Breiman's random forests and random forests with splits limited to 10-quantiles.

$qp = 10^3$ distinct rules of one split, about $(2qp)^2 \approx 10^6$ distinct rules of two splits, and about $(2qp)^3 \approx 10^{10}$ distinct rules of three splits. Using only rules of one split is too restrictive since it generates a small model class (a thousand rules for 100 input variables) and does not handle variable interactions. On the other hand, rules of two splits are numerous (about one million) and thus provide a large flexibility to SIRUS. More importantly, since there are 10 billion rules of three splits, a stable selection of a few of them is clearly an impossible task, and such complex rules are naturally discarded by SIRUS.

In SIRUS, tree depth is set to 2 to reduce the computational cost while leaving the output list of rules almost untouched as explained above. We augment the experiments of the Subsection 3.2 with an additional column in Table 6: "**SIRUS (depth** $= 3$**)**" where SIRUS is run with a tree depth of 3. Over the nine UCI datasets, rules of three splits appear in SIRUS rule list only for "Heart C2" and "Ionosphere", but we observe no accuracy improvement over a tree depth of 2. In the software implementation `sirus`, the tree depth parameter `max.depth` is set to 2 by default but is left as a modifiable input.

This analysis of tree depth is not new. Indeed, both RuleFit (Friedman and Popescu, 2008) and Node harvest (Meinshausen, 2010) articles discuss the optimal tree depth for the rule extraction from a tree ensemble in their experiments. They both conclude that the optimal depth is 2. Hence, the same hard limit of 2 is used in Node harvest. RuleFit is slightly less restrictive: for each tree, its depth is randomly sampled with an exponential distribution concentrated on 2, but allowing few trees of depth 1, 3, and 4. We insist that they both reach such conclusion without considering stability issues, but only focusing on accuracy. Further considering stability properties consolidates that growing shallow trees is optimal for rule extraction from tree ensembles.

4 Theoretical Analysis of Stability

Among the three minimum requirements for interpretability defined in Section 1, simplicity and predictivity are quite easily met for rule models (Cohen and Singer, 1999; Meinshausen, 2010; Letham et al., 2015). On the other hand, as Letham et al. (2015) recall, building a stable

Dataset	SIRUS (depth = 2)	SIRUS (depth = 3)
Haberman	0.36	0.36
Diabetes	0.20	0.20
Heart Statlog	0.12	0.12
Liver Disorders	0.36	0.36
Heart C2	**0.12**	**0.12**
Heart H2	0.12	0.12
Credit German	0.26	0.26
Credit Approval	0.10	0.10
Ionosphere	**0.06**	**0.07**

Table 6: SIRUS error (1-AUC) over a 10-fold cross-validation when tree depth is limited to 2 or 3. The two datasets having rules of three splits output by SIRUS are in bold.

rule ensemble is challenging. Therefore the main goal of this section is to prove the asymptotic stability of SIRUS, i.e., provided that the sample size is large enough, SIRUS systematically outputs the same list of rules when run multiple times with independent samples. On the other hand, we also argue that existing tree-based rule algorithms are unstable by design.

In order to show the asymptotic stability of SIRUS, we first need to introduce formal definitions of the mathematical elements involved in the empirical algorithm. We additionally define the theoretical counterpart of SIRUS, an abstract procedure which is not based on the sample $\mathscr{D}_n$, but only on the unknown distribution $\mathbb{P}_{\mathbf{X},Y}$. Next, we will prove the stochastic convergence of SIRUS towards its theoretical counterpart. This means that the list of selected rules does not depend on the training data $\mathscr{D}_n$, but only on $\mathbb{P}_{\mathbf{X},Y}$, provided that the sample size is large enough. Therefore, the same list of rules is output when SIRUS is run multiple times on independent samples. This mathematical analysis highlights that the remarkable stable behavior of SIRUS in practice has theoretical groundings, and that the discretization of the cut values with the quantiles, as well as using random forests, are the cornerstones to stabilize rule models extracted from tree ensembles.

Empirical algorithm. First, we define the empirical CART-splitting criterion used to find the optimal split at each node of each tree of the forest. In our context of binary classification where the output $Y \in \{0,1\}$, maximizing the so-called empirical CART-splitting criterion is equivalent to maximizing the criterion based on Gini impurity (see, e.g., Biau and Scornet, 2016). More precisely, at node H and for a cut performed along the j-th coordinate at the empirical r-th q-quantile $\hat{q}_{n,r}^{(j)}$, this criterion reads

$$\begin{aligned} L_n(H, \hat{q}_{n,r}^{(j)}) \stackrel{\text{def}}{=} & \frac{1}{N_n(H)} \sum_{i=1}^{n} (Y_i - \overline{Y}_H)^2 \mathbb{1}_{\mathbf{X}_i \in H} \\ & - \frac{1}{N_n(H)} \sum_{i=1}^{n} \left(Y_i - \overline{Y}_{H_L} \mathbb{1}_{X_i^{(j)} < \hat{q}_{n,r}^{(j)}} - \overline{Y}_{H_R} \mathbb{1}_{X_i^{(j)} \geq \hat{q}_{n,r}^{(j)}} \right)^2 \mathbb{1}_{\mathbf{X}_i \in H}, \end{aligned} \tag{4.1}$$

where $\overline{Y}_H$ is the average of the Y_i's such that $\mathbf{X}_i \in H$, $N_n(H)$ is the number of data points $\mathbf{X}_i$ falling into H,

$$H_L \stackrel{\text{def}}{=} \{\mathbf{x} \in H : \mathbf{x}^{(j)} < \hat{q}_{n,r}^{(j)}\}, \quad H_R \stackrel{\text{def}}{=} \{\mathbf{x} \in H : \mathbf{x}^{(j)} \geq \hat{q}_{n,r}^{(j)}\},$$

and for $r \in \{1, \ldots, q-1\}$ the empirical r-th q-quantile of $\{X_1^{(j)}, \ldots, X_n^{(j)}\}$ is defined by

$$\hat{q}_{n,r}^{(j)} = \inf\big\{x \in \mathbb{R} : \frac{1}{n}\sum_{i=1}^{n} \mathbb{1}_{X_i^{(j)} \leq x} \geq \frac{r}{q}\big\}. \tag{4.2}$$

Note that, for the ease of reading, (4.1) is defined for a tree built with the entire dataset $\mathscr{D}_n$ without resampling. As it is often the case in the theoretical analysis of random forests, we assume throughout this section that the subsampling of a_n observations to build each tree is done without replacement to alleviate the mathematical analysis.

Recall that the rule selection is based on the probability $p_n(\mathscr{P})$ that a Θ-random tree of the forest contains a particular path $\mathscr{P} \in \Pi$, that is,

$$p_n(\mathscr{P}) = \mathbb{P}(\mathscr{P} \in T(\Theta, \mathscr{D}_n)|\mathscr{D}_n),$$

and that the Monte-Carlo estimate $\hat{p}_{M,n}(\mathscr{P})$ of $p_n(\mathscr{P})$ is directly computed using the random forest, and takes the form

$$\hat{p}_{M,n}(\mathscr{P}) = \frac{1}{M}\sum_{\ell=1}^{M} \mathbb{1}_{\mathscr{P} \in T(\Theta_\ell, \mathscr{D}_n)}.$$

Clearly, $\hat{p}_{M,n}(\mathscr{P})$ is a good estimate of $p_n(\mathscr{P})$ when M is large since, by the law of large numbers, conditional on $\mathscr{D}_n$,

$$\lim_{M\to\infty} \hat{p}_{M,n}(\mathscr{P}) = p_n(\mathscr{P}) \quad \text{a.s.}$$

We also see that $\hat{p}_{M,n}(\mathscr{P})$ is unbiased since $\mathbb{E}[\hat{p}_{M,n}(\mathscr{P})|\mathscr{D}_n] = p_n(\mathscr{P})$.

Theoretical algorithm. Next, we define all theoretical counterparts of the empirical quantities involved in SIRUS, which do not depend on $\mathscr{D}_n$ but only on the unknown distribution $\mathbb{P}_{\mathbf{X},Y}$ of $(\mathbf{X}, Y)$. For a given integer $q \geq 2$ and $r \in \{1, \ldots, q-1\}$, the theoretical q-quantiles are defined by

$$q_r^{\star(j)} = \inf\big\{x \in \mathbb{R} : \mathbb{P}(X^{(j)} \leq x) \geq \frac{r}{q}\big\},$$

i.e., the population version of $\hat{q}_{n,r}^{(j)}$ defined in (4.2). Similarly, for a given hyperrectangle $H \subseteq \mathbb{R}^p$, we let the theoretical CART-splitting criterion be

$$\begin{aligned} L^\star(H, q_r^{\star(j)}) = \; & \mathbb{V}[Y|\mathbf{X} \in H] \\ & - \mathbb{P}(X^{(j)} < q_r^{\star(j)}|\mathbf{X} \in H) \times \mathbb{V}[Y|X^{(j)} < q_r^{\star(j)}, \mathbf{X} \in H] \\ & - \mathbb{P}(X^{(j)} \geq q_r^{\star(j)}|\mathbf{X} \in H) \times \mathbb{V}[Y|X^{(j)} \geq q_r^{\star(j)}, \mathbf{X} \in H]. \end{aligned}$$

Based on this criterion, we denote by $T^\star(\Theta)$ the list of all paths contained in the theoretical tree built with randomness Θ, where splits are chosen to maximize the theoretical criterion

$L^\star$ instead of the empirical one L_n, defined in (4.1). We stress again that the list $T^\star(\Theta)$ does not depend upon $\mathscr{D}_n$ but only upon the unknown distribution of $(\mathbf{X}, Y)$. Next, we let $p^\star(\mathscr{P})$ be the theoretical counterpart of $p_n(\mathscr{P})$, that is

$$p^\star(\mathscr{P}) = \mathbb{P}(\mathscr{P} \in T^\star(\Theta)),$$

and finally define the theoretical set of selected paths $\mathscr{P}^\star_{p_0}$ by $\{\mathscr{P} \in \Pi : p^\star(\mathscr{P}) > p_0\}$ (with the same post-treatment as for the empirical procedure—see Section 2). Notice that, in the case where multiple splits have the same value of the theoretical CART-splitting criterion, one is randomly selected.

Consistency of the path selection. The construction of the rule ensemble model essentially relies on the path selection and on the estimates $\hat{p}_{M,n}(\mathscr{P})$, $\mathscr{P} \in \Pi$. Therefore, our theoretical analysis first focuses on the asymptotic properties of those estimates in Theorem 1. Our consistency results hold under conditions on the subsampling rate a_n and the number of trees M_n, together with some assumptions on the distribution of the random vector $\mathbf{X}$. They are given below.

(A1) The subsampling rate a_n satisfies $\lim_{n\to\infty} a_n = \infty$ and $\lim_{n\to\infty} \frac{a_n}{n} = 0$.

(A2) The number of trees M_n satisfies $\lim_{n\to\infty} M_n = \infty$.

(A3) $\mathbf{X}$ has a strictly positive density f with respect to the Lebesgue measure. Furthermore, for all $j \in \{1, \ldots, p\}$, the marginal density $f^{(j)}$ of $X^{(j)}$ is continuous, bounded, and strictly positive.

We can now state the consistency of the occurrence frequency of each possible path $\mathscr{P} \in \Pi$ in the modified random forest.

Theorem 1. *If Assumptions (A1)-(A3) are satisfied, then, for all $\mathscr{P} \in \Pi$, we have*

$$\lim_{n\to\infty} \hat{p}_{M_n,n}(\mathscr{P}) = p^\star(\mathscr{P}) \quad \textit{in probability.}$$

Stability. The only source of randomness in the selection of the rules lies in the estimates $\hat{p}_{M_n,n}(\mathscr{P})$. Since Theorem 1 states the consistency of such an estimation, the path selection consistency follows, for all threshold values p_0 that do not belong to the finite set $\mathcal{U}^\star = \{p^\star(\mathscr{P}) : \mathscr{P} \in \Pi\}$ of all theoretical probabilities of appearance for each path $\mathscr{P}$. Indeed, if $p_0 = p^\star(\mathscr{P})$ for some $\mathscr{P} \in \Pi$, then $\mathbb{P}(\hat{p}_{M_n,n}(\mathscr{P}) > p_0)$ does not necessarily converge to 0 and the path selection can be inconsistent. Then, we can deduce that SIRUS is asymptotically stable in the following Corollary 1.

Corollary 1. *Assume that Assumptions (A1)-(A3) are satisfied. Then, provided $p_0 \in [0,1] \setminus \mathcal{U}^\star$, we have*

$$\lim_{n\to\infty} \mathbb{P}(\hat{\mathscr{P}}_{M_n,n,p_0} = \mathscr{P}^\star_{p_0}) = 1,$$

and then

$$\lim_{n\to\infty} \hat{S}_{M_n,n,p_0} = 1 \quad \textit{in probability.}$$

Competitors. As discussed in the experimental Section 3, CART, C5.0, RuleFit, and Node harvest are top competitors of SIRUS, which are also based on rule extraction from trees. However, these algorithms do not include a pre-processing step of discretization, which makes them unstable by design. To see this, we first adapt the definition of an extracted path without discretization as $\mathscr{P} = \{(j_k, z_k, s_k),\ k = 1, \dots, d\}$, where $z_k \in \mathbb{R}$ is now the cutting value of the k-th split. For any rule algorithm, we also define $\hat{S}_{M,n}$ as the proportion of rules shared between the output rule lists over two runs with two independent samples. Note that $M = 1$ for CART and C5.0, and as already mentioned, it is possible to define a rule algorithm from CART, by extracting its nodes, as in C5.0. Thus, we obtain that for any tree-based rule algorithm, $\hat{S}_{M,n} = 0$ almost surely. Indeed, since the input $\mathbf{X}$ takes continuous values (Assumption (A3)) and decision trees can cut at the middle of two observations in all directions, the probability that a cutting value from the tree built with $\mathscr{D}_n$ and one from the tree built with $\mathscr{D}'_n$ are equal is null.

However, recall that in the experiments, we include a pre-processing discretization step to stabilize competitors and enable fair comparisons. With this modification, they reach a value of $\hat{S}_{M,n} > 0$, but still not in par with SIRUS. This shows that the high stability improvement of SIRUS does not only come from the discretization, but mainly from the rule selection procedure, based on the probability of the rule occurrence in a random tree.

Proofs. The proof of Theorem 1 is to be found in the Supplementary Material. It is however interesting to give a sketch of the proof here. Corollary 1 is a direct consequence of Theorem 1, the full proof follows.

Sketch of proof of Theorem 1. The consistency is obtained by showing that $\hat{p}_{M_n,n}(\mathscr{P})$ is asymptotically unbiased with a null variance. The result for the variance is quite straightforward since the variance of $\hat{p}_{M_n,n}(\mathscr{P})$ can be broken into two terms: the variance generated by the Monte-Carlo randomization, which goes to 0 as the number of trees increases (Assumption (A2)), and the variance of $p_n(\mathscr{P})$. Following Mentch and Hooker (2016), since $p_n(\mathscr{P})$ is a bagged estimate it can be seen as an infinite-order U-statistic, and a classic bound on the variance of U-statistics gives that $\mathbb{V}[p_n(\mathscr{P})]$ converges to 0 if $\lim_{n\to\infty} \frac{a_n}{n} = 0$, which is true by Assumption (A1). Next, proving that $\hat{p}_{M_n,n}(\mathscr{P})$ is asymptotically unbiased requires to dive into the internal mechanisms of the random forest algorithm. To do this, we have to show that the CART-splitting criterion is consistent (Lemma 3) and asymptotically normal (Lemma 4) when cuts are limited to empirical quantiles (estimated on the same dataset) and the number of trees grows with n. When cuts are performed on the theoretical quantiles, the law of large numbers and the central limit theorem can be directly applied, so that the proof of Lemmas 3 and 4 boils down to showing that the difference between the empirical CART-splitting criterion evaluated at empirical and theoretical quantiles converges to 0 in probability fast enough. This is done in Lemma 2 thanks to Assumption (A3). □

Proof of Corollary 1. The first result is a consequence of Theorem 1 since

$$\mathbb{P}\big(\hat{\mathscr{P}}_{M_n,n,p_0} \neq \mathscr{P}^\star_{p_0}\big) \leq \sum_{\mathscr{P}\in\Pi} \mathbb{P}(\hat{p}_{M_n,n}(\mathscr{P}) > p_0)\mathbb{1}_{p^\star(\mathscr{P})\leq p_0} + \mathbb{P}(\hat{p}_{M_n,n}(\mathscr{P}) \leq p_0)\mathbb{1}_{p^\star(\mathscr{P})>p_0}.$$

Next, we have

$$\hat{S}_{M_n,n,p_0} = \frac{2 \sum_{\mathscr{P} \in \Pi} \mathbb{1}_{\hat{p}_{M_n,n}(\mathscr{P}) > p_0 \cap \hat{p}'_{M_n,n}(\mathscr{P}) > p_0}}{\sum_{\mathscr{P} \in \Pi} \mathbb{1}_{\hat{p}_{M_n,n}(\mathscr{P}) > p_0} + \mathbb{1}_{\hat{p}'_{M_n,n}(\mathscr{P}) > p_0}}.$$

Since $p_0 \notin \mathcal{U}^\star$, we deduce from Theorem 1 and the continuous mapping theorem that, for all $\mathscr{P} \in \Pi$,

$$\lim_{n \to \infty} \mathbb{1}_{\hat{p}_{M_n,n}(\mathscr{P}) > p_0} = \mathbb{1}_{p^\star(\mathscr{P}) > p_0} \quad \text{in probability.}$$

Therefore, $\lim_{n \to \infty} \hat{S}_{M_n,n,p_0} = 1$ in probability. □

5 Conclusion

Interpretability of learning algorithms is required for applications involving critical decisions, for example the analysis of production processes in the manufacturing industry. Although interpretability does not have a precise definition, we argued that simplicity, stability, and predictivity are minimum requirements. In particular, decision trees and rule algorithms both combine a simple structure and a good accuracy for nonlinear data, and are thus considered as state-of-the-art interpretable algorithms. However, these methods are unstable with respect to data perturbation, which is a strong operational limitation. Therefore, we proposed a new rule algorithm for classification, SIRUS (Stable and Interpretable RUle Set), which takes the form of a short list of rules. We proved that SIRUS considerably improves stability over state-of-the-art algorithms, while preserving simplicity, accuracy, and computational complexity of top competitors. The principle of SIRUS is to extract rules from a random forest, based on their probability of occurrence in a random tree, and to stop the growing of the forest when the rule selection is converged. Thus, SIRUS inherits the computational complexity of random forests, and has only one tuning parameter. A software implementation, the `R/C++` package `sirus` (Benard and Wright, 2020), is available from `CRAN`. Besides, we believe that the extension of SIRUS to regression is a promising future research direction: the main challenge is the construction of an appropriate rule aggregation framework to accurately estimate continuous outputs without hurting stability. Furthermore, although SIRUS has the ability to handle high-dimensional data, as illustrated with the SECOM dataset (590 inputs), specific variable selection strategies could be used to reduce the number of possible rules and then improve SIRUS performance.

A Additional Experiments

Additional experiments are provided to compare SIRUS to other competitors: C5.0 (Quinlan, 1992) (decision tree), PART (Frank and Witten, 1998), and FOIL (Quinlan and Cameron-Jones, 1995) (classical rule learning algorithms). Model size results are provided in Table 7, stability in Table 8, and error in Table 9. The stability and accuracy improvement of SIRUS is clear.

Dataset	C5.0	PART	FOIL	SIRUS
Haberman	2.2	2.2	4.0	3.3
Diabetes	12.5	6.4	44.5	7.9
Heart Statlog	10.6	17.8	28.7	10.7
Liver Disorders	13.7	6.9	2.3	18.5
Heart C2	10.0	19.7	33.7	22.5
Heart H2	4.0	15.2	29.6	12.6
Credit German	22.2	68.7	95.8	17.8
Credit Approval	7.8	30.5	38.5	19.7
Ionosphere	9	6.7	16.8	21.4

Table 7: Mean model size over a 10-fold cross-validation for UCI datasets. Results are averaged over 30 repetitions of the cross-validation. (Standard deviations are negligible, they are not displayed to increase readability.)

Dataset	C5.0	PART	FOIL	SIRUS
Haberman	0.23	0.27	0.07	**0.65**
Diabetes	0.07	0.14	0.17	**0.79**
Heart Statlog	0.11	0.15	0.14	**0.69**
Liver Disorders	0.04	0.07	0.04	**0.57**
Heart C2	0.08	0.19	0.17	**0.66**
Heart H2	0.33	0.30	0.50	**0.65**
Credit German	0.04	0.16	0.11	**0.66**
Credit Approval	0.22	0.27	0.17	**0.66**
Ionosphere	0.22	0.14	0.07	**0.63**

Table 8: Mean stability over a 10-fold cross-validation for UCI datasets. Results are averaged over 30 repetitions of the cross-validation. (Standard deviations are negligible, they are not displayed to increase readability.)

Dataset	C5.0	PART	FOIL	SIRUS
Haberman	0.45	0.42	0.46	**0.36**
Diabetes	0.29	0.30	0.30	**0.20**
Heart Statlog	0.21	0.24	0.21	**0.12**
Liver Disorders	**0.35**	**0.38**	0.48	**0.36**
Heart C2	0.21	0.22	0.20	**0.12**
Heart H2	0.24	0.23	0.22	**0.12**
Credit German	0.37	0.36	0.41	**0.26**
Credit Approval	0.15	0.16	0.16	**0.10**
Ionosphere	0.11	0.11	0.15	**0.06**

Table 9: Model error (1-AUC) over a 10-fold cross-validation for UCI datasets. Results are averaged over 30 repetitions of the cross-validation. (Standard deviations are negligible, they are not displayed to increase readability.)

B Stopping Criterion for the Number of Trees M

The accuracy, stability, and computational cost of SIRUS increase with the number of trees M. Thus, we simply design a stopping criterion to grow the minimum number of trees which ensures that accuracy and stability are higher than 95% of their maximum asymptotic values with respect to M and conditionally on $\mathscr{D}_n$. We empirically observe that the stability requirement is met for a much higher number of trees than the accuracy requirement (about 10 times). Therefore, the stopping criterion is only based on stability. More precisely, we require that 95% of the rules are identical across two runs of SIRUS on a given dataset $\mathscr{D}_n$ in average. Formally, the mean stability $\mathbb{E}[\hat{S}_{M,n,p_0}|\mathscr{D}_n]$ measures the expected proportion of rules shared by two fits of SIRUS on $\mathscr{D}_n$, for fixed n (sample size), p_0 (threshold), and M (number of trees). Thus, the stopping criterion takes the form $1 - \mathbb{E}[\hat{S}_{M,n,p_0}|\mathscr{D}_n] < \alpha$, with typically $\alpha = 0.05$.

There are two obstacles to operationalize this stopping criterion: its estimation and its dependence to p_0. We make two approximations to overcome these limitations and give empirical and theoretical evidence of their good practical behavior in the second section of the Supplementary Material. First, Theorem 2 of the Supplementary Material provides an asymptotic equivalent with respect to M of $1 - \mathbb{E}[\hat{S}_{M,n,p_0}|\mathscr{D}_n]$, that we simply estimate by

$$\varepsilon_{M,n,p_0} = \frac{\sum_{\mathscr{P}\in\Pi} \Phi(Mp_0, M, \hat{p}_{M,n}(\mathscr{P}))(1 - \Phi(Mp_0, M, \hat{p}_{M,n}(\mathscr{P})))}{\sum_{\mathscr{P}\in\Pi}(1 - \Phi(Mp_0, M, \hat{p}_{M,n}(\mathscr{P})))},$$

where $\Phi(Mp_0, M, p_n(\mathscr{P}))$ is the cdf of a binomial distribution with parameter $p_n(\mathscr{P})$, M trials, evaluated at Mp_0. Secondly, ε_{M,n,p_0} depends on p_0, whose optimal value is unknown in the first step of SIRUS, when trees are grown. It turns out however that ε_{M,n,p_0} is not very sensitive to p_0, as shown by the experiments in the Supplementary Material. Consequently, our strategy is to simply average ε_{M,n,p_0} over a set $\hat{V}_{M,n}$ of many possible values of p_0 and use the resulting average as a gauge. These p_0 values are chosen to scan all possible path sets $\hat{\mathscr{P}}_{M,n,p_0}$, of size ranging from 1 to 50 paths. When a set of 50 paths is post-treated, its size reduces to around 25 paths (as explained in Section 3, 25 is an arbitrarily threshold on the maximum number of rules above which a rule model is not readable anymore). In order to generate path sets of such sizes, p_0 values are chosen halfway between two distinct consecutive $\hat{p}_{M,n}(\mathscr{P})$, $\mathscr{P} \in \Pi$, restricted to the highest 50 values. Thus, in the experiments, we utilize the following criterion to stop the growing of the forest, with typically $\alpha = 0.05$:

$$\underset{M}{\operatorname{argmin}}\Big\{\frac{1}{|\hat{V}_{M,n}|}\sum_{p_0\in\hat{V}_{M,n}} \varepsilon_{M,n,p_0} < \alpha\Big\}. \tag{B.1}$$

C Rule Aggregation

In Section 2, $\hat{\eta}_{M,n,p_0}(\mathbf{x})$ (2.3) is a simple average of the set of rules, defined as

$$\hat{\eta}_{M,n,p_0}(\mathbf{x}) = \frac{1}{|\hat{\mathscr{P}}_{M,n,p_0}|}\sum_{\mathscr{P}\in\hat{\mathscr{P}}_{M,n,p_0}} \hat{g}_{n,\mathscr{P}}(\mathbf{x}). \tag{C.1}$$

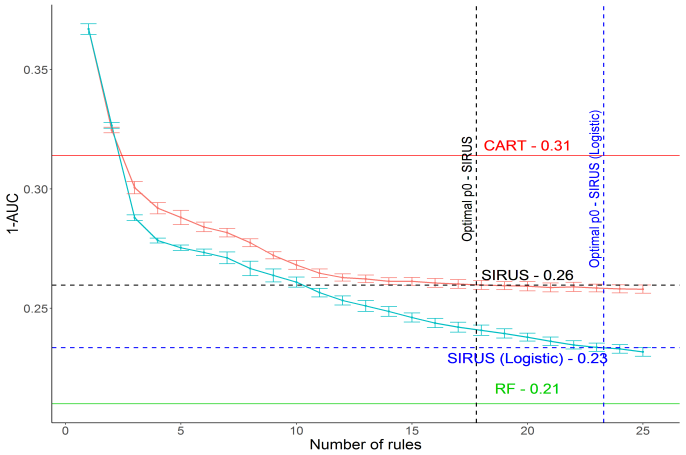

Figure 6: For the UCI dataset "Credit German", 1-AUC versus the number of rules when p_0 varies, estimated via 10-fold cross-validation (repeated 30 times) for two different methods of rule aggregation: the rule average (C.1) in red and a logistic regression (C.2) in blue.

To tackle our binary classification problem, a natural approach would be to use a logistic regression and define

$$\ln\left(\frac{\hat{\eta}_{M,n,p_0}(\mathbf{x})}{1-\hat{\eta}_{M,n,p_0}(\mathbf{x})}\right) = \sum_{\mathscr{P}\in\hat{\mathscr{P}}_{M,n,p_0}} \beta_{\mathscr{P}}\hat{g}_{n,\mathscr{P}}(\mathbf{x}), \tag{C.2}$$

where the coefficients $\beta_{\mathscr{P}}$ have to be estimated. To illustrate the performance of the logistic regression (C.2), we consider again the UCI dataset, "Credit German". We augment the previous results from Figure 4 (in Section 3) with the logistic regression error in Figure 6. One can observe that the predictive accuracy is slightly improved but it comes at the price of an additional set of coefficients that can be hard to interpret (some can be negative), and an increased computational cost.

Supplementary Material

Details about the stopping criterion for the number of trees (experiments and theoretical properties), and the proof of Theorem 1 are available in **Supplementary Material for: SIRUS: Stable and Interpretable RUle Set for Classification**.

References

R. Agrawal, T. Imieliński, and A. Swami. Mining association rules between sets of items in large databases. In *Proceedings of the 1993 ACM SIGMOD International Conference on Management of Data*, pages 207–216, New York, 1993. ACM.

S. Alelyani, Z. Zhao, and H. Liu. A dilemma in assessing stability of feature selection algorithms. In *13th IEEE International Conference on High Performance Computing & Communication*, pages 701–707, Piscataway, 2011. IEEE.

C. Benard and M.N. Wright. *sirus: Stable and Interpretable RUle Set*, 2020. URL https://CRAN.R-project.org/package=sirus. R package version 0.2.1.

G. Biau and E. Scornet. A random forest guided tour (with comments and a rejoinder by the author). *TEST*, 25:197–268, 2016.

A.-L. Boulesteix and M. Slawski. Stability and aggregation of ranked gene lists. *Briefings in Bioinformatics*, 10:556–568, 2009.

O. Bousquet and A. Elisseeff. Stability and generalization. *Journal of Machine Learning Research*, 2:499–526, 2002.

L. Breiman. Bagging predictors. *Machine Learning*, 24:123–140, 1996.

L. Breiman. Random forests. *Machine Learning*, 45:5–32, 2001a.

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

L. Breiman. Setting up, using, and understanding random forests v3.1. Technical report, UC Berkeley, 2003a. URL https://www.stat.berkeley.edu/~breiman/Using_random_forests_V3.1.pdf.

L. Breiman, J.H. Friedman, R.A. Olshen, and C.J. Stone. *Classification and Regression Trees*. Chapman & Hall/CRC, Boca Raton, 1984.

A. Chao, R.L. Chazdon, R.K. Colwell, and T.-J. Shen. Abundance-based similarity indices and their estimation when there are unseen species in samples. *Biometrics*, 62:361–371, 2006.

P. Clark and T. Niblett. The CN2 induction algorithm. *Machine Learning*, 3:261–283, 1989.

W.W. Cohen. Fast effective rule induction. In *Proceedings of the Twelfth International Conference on Machine Learning*, pages 115–123, San Francisco, 1995. Morgan Kaufmann Publishers Inc.

W.W. Cohen and Y. Singer. A simple, fast, and effective rule learner. In *Proceedings of the Sixteenth National Conference on Artificial Intelligence and Eleventh Conference on Innovative Applications of Artificial Intelligence*, pages 335–342, Palo Alto, 1999. AAAI Press.

K. Dembczyński, W. Kotłowski, and R. Słowiński. ENDER: A statistical framework for boosting decision rules. *Data Mining and Knowledge Discovery*, 21:52–90, 2010.

L. Devroye and T. Wagner. Distribution-free inequalities for the deleted and holdout error estimates. *IEEE Transactions on Information Theory*, 25:202–207, 1979.

F. Doshi-Velez and B. Kim. Towards a rigorous science of interpretable machine learning. *arXiv:1702.08608*, 2017.

D. Dua and C. Graff. UCI machine learning repository, 2017. URL http://archive.ics.uci.edu/ml.

M. Fokkema. Fitting prediction rule ensembles with R package pre. *Journal of Statistical Software*, 92:1–30, 2020.

E. Frank and I.H. Witten. Generating accurate rule sets without global optimization. In *Proceedings of the Fifteenth International Conference on Machine Learning*, pages 144–151, San Francisco, 1998. Morgan Kaufmann Publishers Inc.

A.A. Freitas. Comprehensible classification models: A position paper. *ACM SIGKDD Explorations Newsletter*, 15:1–10, 2014.

J.H. Friedman and B.E. Popescu. Importance sampled learning ensembles. Technical report, Stanford University, 2003.

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

J. Fürnkranz and G. Widmer. Incremental reduced error pruning. In *Proceedings of the 11th International Conference on Machine Learning*, pages 70–77, San Francisco, 1994. Morgan Kaufmann Publishers Inc.

R. Guidotti and S. Ruggieri. On the stability of interpretable models. In *International Joint Conference on Neural Networks*, pages 1–8, Piscataway, 2019. IEEE.

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

Z. He and W. Yu. Stable feature selection for biomarker discovery. *Computational Biology and Chemistry*, 34:215–225, 2010.

K. Hornik, C. Buchta, and A. Zeileis. Open-source machine learning: R meets Weka. *Computational Statistics*, 24:225–232, 2009.

I. Johnson and M. Hahsler. *arulesCBA: Classification Based on Association Rules*, 2020. URL https://CRAN.R-project.org/package=arulesCBA. R package version 1.1.6.

M. Kuhn and R. Quinlan. *C50: C5.0 Decision Trees and Rule-Based Models*, 2020. URL https://CRAN.R-project.org/package=C50. R package version 0.1.3.

K. Kumbier, S. Basu, J.B. Brown, S. Celniker, and B. Yu. Refining interaction search through signed iterative random forests. *arXiv:1810.07287*, 2018.

H. Lakkaraju, S.H. Bach, and J. Leskovec. 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*, pages 1675–1684, New York, 2016. ACM.

B. Letham. *Statistical learning for decision making: Interpretability, uncertainty, and inference*. PhD thesis, Massachusetts Institute of Technology, 2015.

B. Letham, C. Rudin, T.H. McCormick, and D. Madigan. Interpretable classifiers using rules and bayesian analysis: Building a better stroke prediction model. *The Annals of Applied Statistics*, 9:1350–1371, 2015.

Z.C. Lipton. The mythos of model interpretability. *arXiv:1606.03490*, 2016.

B. Liu, W. Hsu, and Y. Ma. Integrating classification and association rule mining. In *Proceedings of the 14th International Conference on Knowledge Discovery and Data Mining*, volume 98, pages 80–86, New York, 1998. ACM.

N. Meinshausen. Node harvest. *The Annals of Applied Statistics*, 4:2049–2072, 2010.

N. Meinshausen. *Node harvest*, 2015. URL https://CRAN.R-project.org/package=nodeHarvest. R package version 0.7-3.

L. Mentch and G. Hooker. Quantifying uncertainty in random forests via confidence intervals and hypothesis tests. *Journal of Machine Learning Research*, 17:841–881, 2016.

R.S. Michalski. On the quasi-minimal solution of the general covering problem. In *Proceedings of the Fifth International Symposium on Information Processing*, pages 125–128, New York, 1969. ACM.

W.J. Murdoch, C. Singh, K. Kumbier, R. Abbasi-Asl, and B. Yu. Interpretable machine learning: Definitions, methods, and applications. *arXiv:1901.04592*, 2019.

T. Oates and D. Jensen. The effects of training set size on decision tree complexity. In *Proceedings of the 14th International Conference on Machine Learning*, pages 254–262, San Francisco, 1997. Morgan Kaufmann Publishers Inc.

T. Poggio, R. Rifkin, S. Mukherjee, and P. Niyogi. General conditions for predictivity in learning theory. *Nature*, 428:419–422, 2004.

J.R. Quinlan. Learning logical definitions from relations. *Machine learning*, 5:239–266, 1990.

J.R. Quinlan. *C4.5: Programs for Machine Learning*. Morgan Kaufmann Publishers Inc., San Mateo, 1992.

J.R. Quinlan and R.M. Cameron-Jones. Induction of logic programs: Foil and related systems. *New Generation Computing*, 13:287–312, 1995.

M.T. Ribeiro, S. Singh, and C. Guestrin. Why should I trust you? Explaining the predictions of any classifier. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 1135–1144, New York, 2016. ACM.

R.L. Rivest. Learning decision lists. *Machine Learning*, 2:229–246, 1987.

W.H. Rogers and T.J. Wagner. A finite sample distribution-free performance bound for local discrimination rules. *The Annals of Statistics*, 6:506–514, 1978.

C. Rudin. Please stop explaining black box models for high stakes decisions. *arXiv:1811.10154*, 2018.

S. Rüping. *Learning interpretable models*. PhD thesis, Universität Dortmund, 2006.

T. Therneau and B. Atkinson. *rpart: Recursive Partitioning and Regression Trees*, 2019. URL https://CRAN.R-project.org/package=rpart. R package version 4.1-15.

R. Tibshirani. Regression shrinkage and selection via the lasso. *Journal of the Royal Statistical Society. Series B (Methodological)*, pages 267–288, 1996.

V. Vapnik. *Statistical Learning Theory*. Wiley, New York, 1998.

S.M. Weiss and N. Indurkhya. Lightweight rule induction. In *Proceedings of the Seventeenth International Conference on Machine Learning*, pages 1135–1142, San Francisco, 2000. Morgan Kaufmann Publishers Inc.

M.N. Wright and A. Ziegler. ranger: A fast implementation of random forests for high dimensional data in C++ and R. *Journal of Statistical Software*, 77:1–17, 2017.

H. Yang, C. Rudin, and M. Seltzer. Scalable Bayesian rule lists. In *Proceedings of the 34th International Conference on Machine Learning*, pages 3921–3930, Cambridge MA, 2017. JMLR.

X. Yin and J. Han. CPAR: Classification based on predictive association rules. In *Proceedings of the 2003 SIAM International Conference on Data Mining*, pages 331–335, Philadelphia, 2003. SIAM.

B. Yu. Stability. *Bernoulli*, 19:1484–1500, 2013.

B. Yu and K. Kumbier. Three principles of data science: Predictability, computability, and stability (PCS). *arXiv:1901.08152*, 2019.

M.J. Zaki, S. Parthasarathy, M. Ogihara, and W. Li. Parallel algorithms for discovery of association rules. *Data Mining and Knowledge Discovery*, 1:343–373, 1997.

M. Zucknick, S. Richardson, and E.A. Stronach. Comparing the characteristics of gene expression profiles derived by univariate and multivariate classification methods. *Statistical Applications in Genetics and Molecular Biology*, 7:1–34, 2008.