SIRUS: Making Random Forests Interpretable

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

SIRUS: Making Random Forests Interpretable

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, random forests, rules, stability.

1 Introduction

Industrial context 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 line, and then to increase the process efficiency. State-of-the-art supervised learning algorithms can successfully catch patterns of such complex physical phenomena, characterized by nonlinear effects and low-order interactions between parameters. However, any decision impacting the production process has long-term and heavy consequences, 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

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

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.

Interpretability 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 to 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. 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. 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 some 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 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 (Breiman et al., 1984). Trees can model highly nonlinear patterns while having a simple structure, and are therefore good candidates when interpretability is required. However, as explained in Breiman (2001b), trees are unstable to small data perturbations, which is a strong limitation to their practical use. In an operational context, as a new batch of data is collected from a stationary production process, the conclusions can drastically change, and such unstable models provide us with a partial and arbitrary analysis of the underlying phenomena.

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 models 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. Rule learning originates from the influential AQ system of Michalski (1969). Many algorithms were subsequently developped in the 1980's and 1990's, including Decision List (Rivest, 1987), CN2 (Clark and Niblett, 1989), C4.5 (Quinlan, 1992), 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). The last decade has seen a resurgence of rule models, especially with RuleFit (Friedman et al., 2008), Node harvest (Meinshausen, 2010), ENDER (Ensemble of Decision Rules, Dembczyński et al., 2010), and BRL (Bayesian Rule Lists, Letham et al., 2015). Despite their simplicity and excellent predictive skills, these approaches are unstable and, from this point of view, share the same limitation as decision trees (Letham et al., 2015). 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. Besides, s-IRF is designed to model biological systems, characterized by high-order interactions, whereas we are more concerned with low-order interactions involved in the analysis of industrial processes—typically main effects and second-order interactions. In this industrial setting, the extraction of signed interactions via 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. Therefore an explicit rule model is required to identify input values of interest.

SIRUS In line with the above, we design in the present paper a new supervised classification algorithm that we call **SIRUS** (**S**table and **I**nterpretable **RU**le **S**et). SIRUS inherits the accuracy of random forests and the simplicity of decision trees, while having a stable structure for problems with low-order interaction effects. 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 4 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 $(X^{(1)}, X^{(2)}, \ldots, X^{(590)})$ to monitor the production. At the end of the line, each of the 1567 produced entities is associated to a pass/fail label, 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**	$p_f = 16.4\%$

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. In Section 3, we establish the consistency and the stability of the rule selection procedure. These results allow us to derive empirical guidelines for parameter tuning, gathered in Section 4, which is critical for good practical performance. One of the main contributions of this work is the development of a software implementation of SIRUS, via the `R` package `sirus` available from `CRAN`, based on `ranger`, a high-performance random forest implementation in `R` and `C++` (Wright and Ziegler, 2017). We illustrate, in Section 4, the efficiency of our procedure `sirus` through numerical experiments on real datasets.

2 SIRUS description

Within the general framework of supervised (binary) classification, we assume to be given an i.i.d. sample $\mathscr{D}_n = \{(\mathbf{X}_i, Y_i), i = 1, \dots, n\}$. Each $(\mathbf{X}_i, Y_i)$ is distributed as the generic pair $(\mathbf{X}, Y)$ independent of $\mathscr{D}_n$, where $\mathbf{X} = (X^{(1)}, \dots, X^{(p)})$ 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 with trees of depth 2 (i.e., interactions of order 2). 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. To present SIRUS, we first describe how individual rules are created in Subsection 2.1, and then show how to select and aggregate the individual rules to obtain a more robust classifier in Subsection 2.2.

2.1 Basic elements

Random forests 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 helps to stabilize the forest structure while keeping almost intact the predictive accuracy, provided q is not too small (typically of the order of 10—see the experimental Subsection 4.2). Also, because the targeted applications involve low-order interactions, the depth of the individual trees is limited to $d = 2$ (so, each tree has at most four terminal leaves). This produces shallow and simple trees, unlike traditional forests which use trees of maximal depth. Apart from these differences, 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. More precisely, we let $\Theta^{(S)} \subset \{1, \ldots, n\}^{a_n}$ be the indexes of the observations in $\mathscr{D}_n$ sampled with replacement to build the tree, where $a_n \in \{1, \ldots, n\}$ is the number of sampled observations (it is a parameter of SIRUS). As for $\Theta^{(V)}$, since the tree depth is limited to 2, it takes the form

$$\Theta^{(V)} = \big(\Theta_0^{(V)}, \Theta_L^{(V)}, \Theta_R^{(V)}\big),$$

where $\Theta_0^{(V)}$ (resp., $\Theta_L^{(V)}$ and $\Theta_R^{(V)}$) is the set of coordinates selected to split the root node (resp., its left and right children). As in the original forests, $\Theta_0^{(V)}$, $\Theta_L^{(V)}$, and $\Theta_R^{(V)}$ are of cardinality $\texttt{mtry} \in \{1, \ldots, p\}$, an additional parameter of SIRUS.

Throughout the manuscript, for a given integer $q \geq 2$ and $r \in \{1, \ldots, q-1\}$, we let $\hat{q}_{n,r}^{(j)}$ be the empirical r-th q-quantile of $\{X_1^{(j)}, \ldots, X_n^{(j)}\}$, i.e.,

$$\hat{q}_{n,r}^{(j)} = \inf\big\{x \in \mathbb{R} : \frac{1}{n}\sum_{i=1}^{n} \mathbb{1}_{X_i^{(j)} \leq x} \geq \frac{r}{q}\big\}. \tag{2.1}$$

The construction of the individual trees is summarized in the table below and is illustrated in Figure 1.

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 $s = 0$ (the root of the tree).
5: Draw uniformly at random a subset $\Theta_s^{(V)} \subset \{1, \ldots, p\}$ of cardinality `mtry`.
6: For all $j \in \Theta_s^{(V)}$, compute the CART-splitting criterion at all empirical q-quantiles of $X^{(j)}$ that split the cell s into two non-empty cells.
7: Choose the split that maximizes the CART-splitting criterion.
8: Repeat **lines** $5-7$ for the two resulting cells (i.e., $s = L$ and $s = R$).

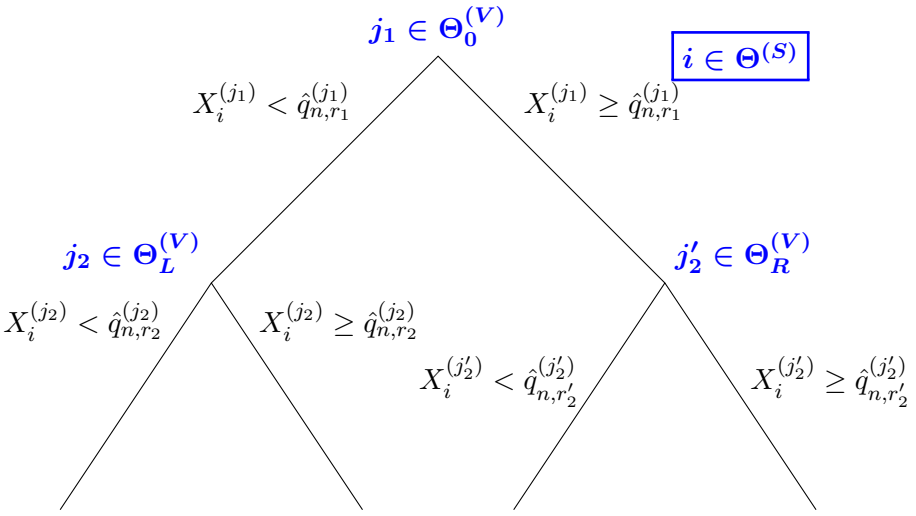

Figure 1: Schematic view of a randomized tree of depth 2. $\Theta_0^{(V)}$ (resp., $\Theta_L^{(V)}$ and $\Theta_R^{(V)}$) is the set of coordinates selected to split the root node (resp., its left and right children).

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:

$$\begin{aligned} L_n(H, \hat{q}_{n,r}^{(j)}) \stackrel{\text{def}}{=} & \frac{1}{N_n(H)} \sum_{i=1}^{n} (Y_i - \overline{Y}_H)^2 \mathbb{1}_{\mathbf{X}_i \in H} \\ & - \frac{1}{N_n(H)} \sum_{i=1}^{n} \left(Y_i - \overline{Y}_{H_L} \mathbb{1}_{X_i^{(j)} < \hat{q}_{n,r}^{(j)}} - \overline{Y}_{H_R} \mathbb{1}_{X_i^{(j)} \geq \hat{q}_{n,r}^{(j)}} \right)^2 \mathbb{1}_{\mathbf{X}_i \in H}, \end{aligned} \tag{2.2}$$

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, and

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

Note that, for the ease of reading, (2.2) is defined for a tree built with the entire dataset $\mathscr{D}_n$ without resampling.

Following the construction of Algorithm 1, SIRUS grows M randomized trees, where the extra randomness used to build the ℓ-th tree is denoted by Θ_ℓ. The random variables $\Theta_1, \ldots, \Theta_M$ are generated as i.i.d. copies of the generic variable $\Theta = (\Theta^{(S)}, \Theta_0^{(V)}, \Theta_L^{(V)}, \Theta_R^{(V)})$, so that tree structures are independent conditional on the dataset $\mathscr{D}_n$.

Path representation In order to go further in the presentation of SIRUS, we still need to introduce a useful notation, which describes the paths that go from the root of the tree to a given node. To this aim, we follow the example shown in Figure 2 with a tree of depth 2 partitioning the input space $\mathbb{R}^2$, as we will only consider trees of depth 2 throughout the

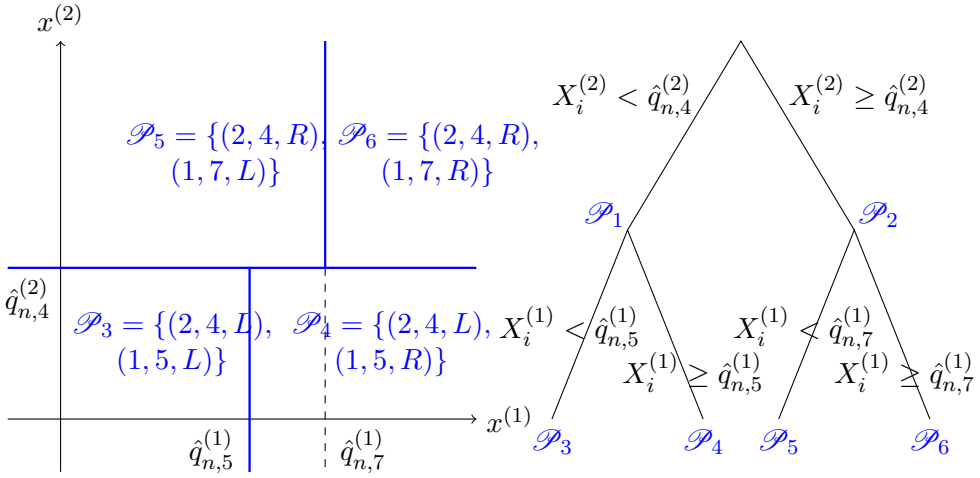

Figure 2: Example of a root node $\mathbb{R}^2$ partitionned by a randomized tree of depth 2: the tree on the right side, the associated paths and hyperrectangles of length $d = 2$ on the left side.

document. For instance, let us consider the node $\mathscr{P}_6$ defined by the sequence of two splits $X_i^{(2)} \geq \hat{q}_{n,4}^{(2)}$ and $X_i^{(1)} \geq \hat{q}_{n,7}^{(1)}$. 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 $\mathscr{P}_6 = \{(2, 4, R), (1, 7, R)\}$. Of course, this generalizes to each path $\mathscr{P}$ of length $d = 1$ or $d = 2$ under the symbolic compact form

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

where, for $k \in \{1, \ldots, d\}$ ($d \in \{1, 2\}$), 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—an easy calculation shows that Π is a finite set of cardinality $2p(q-1) + p(4p-1)(q-1)^2$. On the other hand, a Θ-random tree of depth 2 generates (at most) 6 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, which is therefore a random subset of Π. Note that, in very specific cases, we can have less than 6 paths in $T(\Theta, \mathscr{D}_n)$, typically if one of the two child nodes does not have any possible splits in the selected directions.

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

$$\hat{H}_n(\mathscr{P}) = \left\{ \mathbf{x} \in \mathbb{R}^p : \begin{cases} \mathbf{x}^{(j_k)} < \hat{q}_{n,r_k}^{(j_k)} & \text{if } s_k = L \\ \mathbf{x}^{(j_k)} \geq \hat{q}_{n,r_k}^{(j_k)} & \text{if } s_k = R \end{cases}, k = 1, \ldots, d \right\}. \tag{2.3}$$

Thus, for each path $\mathscr{P} \in \Pi$, we logically define the companion elementary rule $\hat{g}_{n,\mathscr{P}}$ by

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

with the convention $0/0 = 0$. For $\mathbf{x} \in \mathbb{R}^p$, the elementary rule $\hat{g}_{n,\mathscr{P}}(\mathbf{x})$ is an estimate of the probability that $\mathbf{x}$ is of class 1, depending whether $\mathbf{x}$ falls in $\hat{H}_n(\mathscr{P})$ or not. We note that such a rule depends on the dataset $\mathscr{D}_n$ and the particular path $\mathscr{P}$. One small word of caution: 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 can take two different values and simply reads "if *conditions on* $\boldsymbol{x}$, then *response*, else *default response*".

The elementary rules $\hat{g}_{n,\mathscr{P}}$ will serve as building blocks for SIRUS, which will learn from a collection of such rules. Since each Θ-random tree generates (at most) 6 rules through the path extraction, we can then generate a wide collection of rules using our modified random forest. The next subsection describes how we select and aggregate the most important rules of the forest to form a compact, stable, and predictive rule ensemble model.

2.2 SIRUS

Rule selection Using our modified random forest algorithm, we are able to generate a large number M of trees (typically $M = 10\,000$), randomized by $\Theta_1, \ldots, \Theta_M$. 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. As described above, for each Θ_ℓ-random tree, we extract 6 rules through the associated paths. To make this selection procedure explicit, we let $p_n(\mathscr{P})$ be the probability that a Θ-random tree of the forest contains a particular path $\mathscr{P} \in \Pi$, that is,

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

The Monte-Carlo estimate $\hat{p}_{M,n}(\mathscr{P})$ of $p_n(\mathscr{P})$, which can be directly computed using the random forest, 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 $\mathbb{E}[\hat{p}_{M,n}(\mathscr{P}) | \mathscr{D}_n] = p_n(\mathscr{P})$.

Now, let $p_0 \in (0, 1)$ be a fixed parameter to be selected later on. 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 p_0. We are thus interested in the set

$$\hat{\mathscr{P}}_{M,n,p_0} = \{\mathscr{P} \in \Pi : \hat{p}_{M,n}(\mathscr{P}) > p_0\}. \tag{2.4}$$

We see that if M is large enough, then $\hat{\mathscr{P}}_{M,n,p_0}$ is a good estimate of

$$\mathscr{P}_{n,p_0} = \{\mathscr{P} \in \Pi : p_n(\mathscr{P}) > p_0\}.$$

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 6 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 make the method operational. 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 $\hat{\mathscr{P}}_{M,n,p_0}$. The post-treatment mechanism is fully described and illustrated in Appendix A. Note that the theoretical properties of SIRUS will only be stated for $\hat{\mathscr{P}}_{M,n,p_0}$ without post-treatment. However, since the post-treatment is deterministic, all subsequent results still hold when $\hat{\mathscr{P}}_{M,n,p_0}$ is post-treated (except the second part of Theorem 2—see Remark 1).

Rule aggregation Recall that our objective is to estimate the conditional probability $\eta(\mathbf{x}) = \mathbb{P}(Y = 1|\mathbf{X} = \mathbf{x})$ with a few simple and stable rules. To reach this goal, we propose to 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 step of SIRUS. The aggregated estimate $\hat{\eta}_{M,n,p_0}(\mathbf{x})$ of $\eta(\mathbf{x})$ is thus defined by

$$\hat{\eta}_{M,n,p_0}(\mathbf{x}) = \frac{1}{|\hat{\mathscr{P}}_{M,n,p_0}|} \sum_{\mathscr{P} \in \hat{\mathscr{P}}_{M,n,p_0}} \hat{g}_{n,\mathscr{P}}(\mathbf{x}). \tag{2.5}$$

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 et al., 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 our 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 A for a comparison between $\hat{\eta}_{M,n,p_0}$ defined as simple average (2.5) and defined with a logistic regression.

3 Theoretical properties

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. 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 rule ensemble is challenging. In the second part of the section, we provide a definition of stability in the context of rule models, introduce relevant metrics, and prove the asymptotic stability of SIRUS.

Let us start by defining 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 $r \in \{1, \dots, q-1\}$, the theoretical q-quantiles are defined by

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

i.e., the population version of $\hat{q}_{n,r}^{(j)}$ defined in (2.1). 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 (2.2). 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 $\mathscr{P}_{p_0}^\star$ 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.

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. Note however that Theorem 2 is valid for subsampling with or without replacement.

3.1 Consistency of the path selection

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 $j \in \{1, \dots, p\}$, the marginal density $f^{(j)}$ of $X^{(j)}$ is continuous, bounded, and strictly positive.

We are now in a position to state the main result of this section.

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

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

The proof of Theorem 1 is to be found in the Supplementary Material A. It is however interesting to give a sketch of the proof here. 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).

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, as formalized in Corollary 1, 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.

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

$$\lim_{n\to\infty} \mathbb{P}(\hat{\mathscr{P}}_{M_n,n,p_0} \neq \mathscr{P}^\star_{p_0}) = 0.$$

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

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

□

Corollary 1 is a stability result, and thus a first step towards our objective of designing a stable rule ensemble algorithm. However, such an asymptotic result does not guarantee stability for finite samples. Metrics to quantify stability in that case are introduced in the next subsection.

3.2 Stability

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), and Bousquet and Elisseeff (2002) 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. 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

$$\hat{S}_{M,n,p_0} = \frac{2\left|\hat{\mathscr{P}}_{M,n,p_0} \cap \hat{\mathscr{P}}'_{M,n,p_0}\right|}{\left|\hat{\mathscr{P}}_{M,n,p_0}\right| + \left|\hat{\mathscr{P}}'_{M,n,p_0}\right|} \tag{3.1}$$

with the convention $\frac{0}{0} = 1$. 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 $\hat{S}_{M,n,p_0} = 1$. We also define S_{n,p_0}, the population counterpart of $\hat{S}_{M,n,p_0}$ based on $\mathscr{P}_{n,p_0}$ and $\mathscr{P}'_{n,p_0}$, as

$$S_{n,p_0} = \frac{2\left|\mathscr{P}_{n,p_0} \cap \mathscr{P}'_{n,p_0}\right|}{\left|\mathscr{P}_{n,p_0}\right| + \left|\mathscr{P}'_{n,p_0}\right|}. \tag{3.2}$$

SIRUS is stable regarding the metrics (3.1) and (3.2), as stated in the following corollary.

Corollary 2. *Assume that Assumptions (A1)-(A3) are satisfied. Then, provided* $p_0 \in [0,1] \setminus \mathcal{U}^\star$, *we have*

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

The same limiting result holds for S_{n,p_0}.

Proof of Corollary 2. 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. The case S_{n,p_0} is similar. □

Corollary 2 shows that the rule ensemble is asymptotically stable for both the infinite and finite forests, respectively corresponding to S_{n,p_0} and $\hat{S}_{M_n,n,p_0}$. Of course, the latter case is of greater interest since only a finite forest is grown in practice. Nevertheless, an important stability requirement for SIRUS is to output the same set of rules when fitted multiple times on the same dataset $\mathscr{D}_n$, for a fixed sample size n and a given p_0. This means that, conditionally on $\mathscr{D}_n$ and with $\mathscr{D}'_n = \mathscr{D}_n$, $\hat{S}_{M,n,p_0}$ should be close to 1. The first statement of Theorem 2 below shows that this is indeed the case. Theorem 2 also provides an asymptotic approximation of $\mathbb{E}[\hat{S}_{M,n,p_0}|\mathscr{D}_n]$ for large values of the number of trees M, which quantifies the influence of M on the mean stability, conditional on $\mathscr{D}_n$. We let $\mathcal{U}_n \overset{\text{def}}{=} \{p_n(\mathscr{P}) : \mathscr{P} \in \Pi\}$ be the empirical counterpart of $\mathcal{U}^\star$.

Theorem 2. *If $p_0 \in [0,1] \setminus \mathcal{U}_n$ and $\mathscr{D}'_n = \mathscr{D}_n$, then, conditional on $\mathscr{D}_n$, we have*

$$\lim_{M\to\infty} \hat{S}_{M,n,p_0} = 1 \quad \textit{in probability.} \tag{3.3}$$

In addition, for all $p_0 < \max \mathcal{U}_n$,

$$1-\mathbb{E}[\hat{S}_{M,n,p_0}|\mathscr{D}_n] \underset{M\to\infty}{\sim} \sum_{\mathscr{P}\in\Pi} \frac{\Phi(Mp_0, M, p_n(\mathscr{P}))(1-\Phi\big(Mp_0, M, p_n(\mathscr{P})\big))}{\frac{1}{2}\sum_{\mathscr{P}'\in\Pi} \mathbb{1}_{p_n(\mathscr{P}')>p_0} + \mathbb{1}_{p_n(\mathscr{P}')>p_0-\rho_n(\mathscr{P},\mathscr{P}')\frac{\sigma_n(\mathscr{P}')}{\sigma_n(\mathscr{P})}(p_0-p_n(\mathscr{P}))}},$$

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, and, for all $\mathscr{P}, \mathscr{P}' \in \Pi$,

$$\sigma_n(\mathscr{P}) = \sqrt{p_n(\mathscr{P})(1-p_n(\mathscr{P}))},$$

and

$$\rho_n(\mathscr{P}, \mathscr{P}') = \frac{\mathrm{Cov}(\mathbb{1}_{\mathscr{P}\in T(\Theta,\mathscr{D}_n)}, \mathbb{1}_{\mathscr{P}'\in T(\Theta,\mathscr{D}_n)}|\mathscr{D}_n)}{\sigma_n(\mathscr{P})\sigma_n(\mathscr{P}')}.$$

The proof of Theorem 2 is to be found in the Supplementary Material A. Despite its apparent complexity, the asymptotic approximation of $1-\mathbb{E}[\hat{S}_{M,n,p_0}|\mathscr{D}_n]$ can be easily estimated, and plays an essential role to stop the growing of the forest at an optimal number of trees M, as illustrated in the next section.

Remark 1. *As mentioned in Section 2, the equivalent provided in Theorem 2 is defined when the sets of rules $\hat{\mathscr{P}}_{M,n,p_0}$ and $\hat{\mathscr{P}}'_{M,n,p_0}$ are not post-treated. It considerably simplifies the analysis of the asymptotic behavior of $\mathbb{E}[\hat{S}_{M,n,p_0}|\mathscr{D}_n]$. Since the post-treatment is deterministic, this operation is not an additional source of instability. Then, if the estimation of the rule set without post-treatment is stable, it is also the case when the post-treatment is added. Therefore an efficient stopping criterion for the number of trees can be derived from Theorem 2.*

4 Tuning and experiments

We recall that our objective is to design simple, stable, and predictive rule models, with an acceptable computational cost. In practice, for a finite sample $\mathscr{D}_n$, SIRUS relies on two hyperparameters: the number of trees M and the selection threshold p_0. This section provides a procedure to set optimal values for M and p_0, and illustrates the good performance of SIRUS on real datasets.

4.1 Tuning of SIRUS

Throughout this section, we should keep in mind that in SIRUS, the random forest is only involved in the selection of the paths. Conditionally on $\mathscr{D}_n$, the set of selected paths $\hat{\mathscr{P}}_{M,n,p_0} = \{\mathscr{P} \in \Pi : \hat{p}_{M,n}(\mathscr{P}) > p_0\}$ is a good estimate of its population counterpart $\mathscr{P}_{n,p_0}$ when M is large.

Tuning of M to maximize stability As explained in Section 3, an important stability requirement is that SIRUS outputs the same set of rules when fitted multiple times on a given dataset $\mathscr{D}_n$. This is quantified by the mean stability $\mathbb{E}[\hat{S}_{M,n,p_0}|\mathscr{D}_n]$, which 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). Since the computational cost increases linearly with M, we propose to stop the growing of the forest when the mean stability is close enough to 1, with typically a gap smaller than $\alpha = 0.05$. Thus, the stopping criterion takes the form $1 - \mathbb{E}[\hat{S}_{M,n,p_0}|\mathscr{D}_n] < \alpha$.

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 evidence of their good practical behavior. First, Theorem 2 provides an asymptotic equivalent of $1 - \mathbb{E}[\hat{S}_{M,n,p_0}|\mathscr{D}_n]$, that we simply estimate by

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

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 of Figure 7 in Appendix A. 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 (see Appendix A for a precise definition) and use the resulting average as a gauge. Thus, in the experiments, we utilize the following criterion to stop the growing of the forest, with typically $\alpha = 0.05$:

$$\operatorname*{argmin}_M \Big\{ \frac{1}{|\hat{V}_{M,n}|} \sum_{p_0 \in \hat{V}_{M,n}} \varepsilon_{M,n,p_0} < \alpha \Big\}. \tag{4.1}$$

Experiments showing the good empirical performance of this criterion are presented in Appendix A.

Remark 2. *We emphasize that growing more trees does not improve predictive accuracy or stability with respect to data perturbation for a fixed sample size n. Indeed, the instability of the rule selection is generated by the variance of the estimates $\hat{p}_{M,n}(\mathscr{P}), \mathscr{P} \in \Pi$. Upon noting that we have two sources of randomness—Θ and $\mathscr{D}_n$—, the law of total variance shows that $\mathbb{V}[\hat{p}_{M,n}(\mathscr{P})]$ can be broken down into two terms: the variance generated by the Monte Carlo randomness Θ on the one hand, and the sampling variance on the other hand. In fact, equation (1.3) in the proof of Theorem 1 (Supplementary Material A) reveals that*

$$\mathbb{V}[\hat{p}_{M,n}(\mathscr{P})] = \frac{1}{M}\mathbb{E}[p_n(\mathscr{P})](1 - \mathbb{E}[p_n(\mathscr{P})]) + (1 - \frac{1}{M})\mathbb{V}[p_n(\mathscr{P})].$$

The stopping criterion (4.1) ensures that the first term becomes negligible as $M \to \infty$, so that $\mathbb{V}[\hat{p}_{M,n}(\mathscr{P})]$ reduces to the sampling variance $\mathbb{V}[p_n(\mathscr{P})]$, which is independent of M. Therefore, stability with respect to data perturbation cannot be further improved by increasing the number of trees. Additionally, the trees are only involved in the selection of the paths. For a given set of paths $\hat{\mathscr{P}}_{M,n,p_0}$, the construction of the final aggregated estimate $\hat{\eta}_{M,n,p_0}$ (see (2.5)) is independent of the forest. Thus, if further increasing the number of trees does not impact the path selection, neither it improves the predictive accuracy.

Tuning of p_0 to maximize accuracy The parameter p_0 is a threshold involved in the definition of $\hat{\mathscr{P}}_{M,n,p_0}$ to filter the most important rules, and therefore determines the complexity of the model. The 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 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 3 and 4 for examples.

4.2 Experiments

We have conducted experiments on 9 diverse public datasets from the UCI repository (Dua and Graff, 2017; data is described in Table 1), as well as on the SECOM data, collected from a semi-conductor manufacturing process. The first batch of experiments aims at illustrating the good behavior of SIRUS in various settings. Especially, we observe that the restrictions in the forest growing (cut values on quantiles and a tree depth of two) are not strong limitations, and that SIRUS provides a substantial improvement of stability compared to state-of-the-art rule algorithms. On the other hand, the SECOM dataset is an example of a manufacturing process

problem. Typically, data is unbalanced (since most of the production is valid), hundreds of parameters are collected along the production line, with many noisy ones, and the order of interaction between these parameters is low.

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

We use the `R/C++` software implementation `sirus` (available from `CRAN`), adapted from `ranger`, a fast random forest implementation (Wright and Ziegler, 2017). The hyperparameters M and p_0 are tuned as explained earlier, we set $mtry = \lfloor \frac{p}{3} \rfloor$ and $q = 10$ quantiles. Bootstrap is used for the resampling mechanism, i.e., resampling is done with replacement and $a_n = n$. Finally, categorical variables are transformed in multiple binary variables.

Performance metrics As we have seen several times, an interpretable classifier is based on three essential features: simplicity, stability, and predictive accuracy. We introduce relevant metrics to assess those 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 predictive accuracy, 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 Corollary 2 in Subsection 3.2. 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.

UCI datasets Now, SIRUS is run on the 9 selected UCI datasets. Figure 3 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 0.31 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 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 4 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.

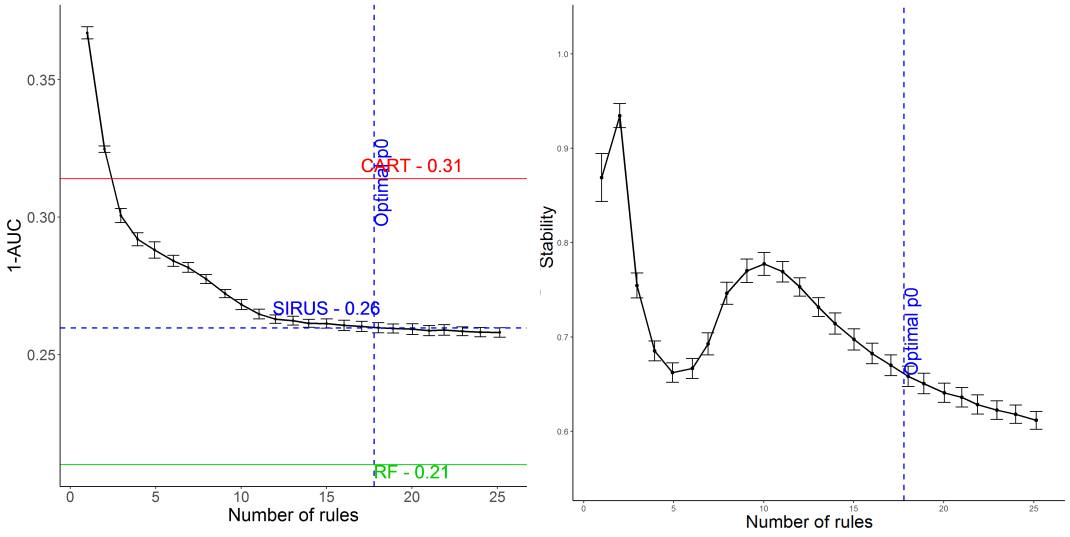

Figure 3: For the UCI dataset "Credit German", 1-AUC (on the left) and stability (on the right) versus the number of rules when p_0 varies, estimated via 10-fold cross-validation (results are averaged over 30 repetitions).

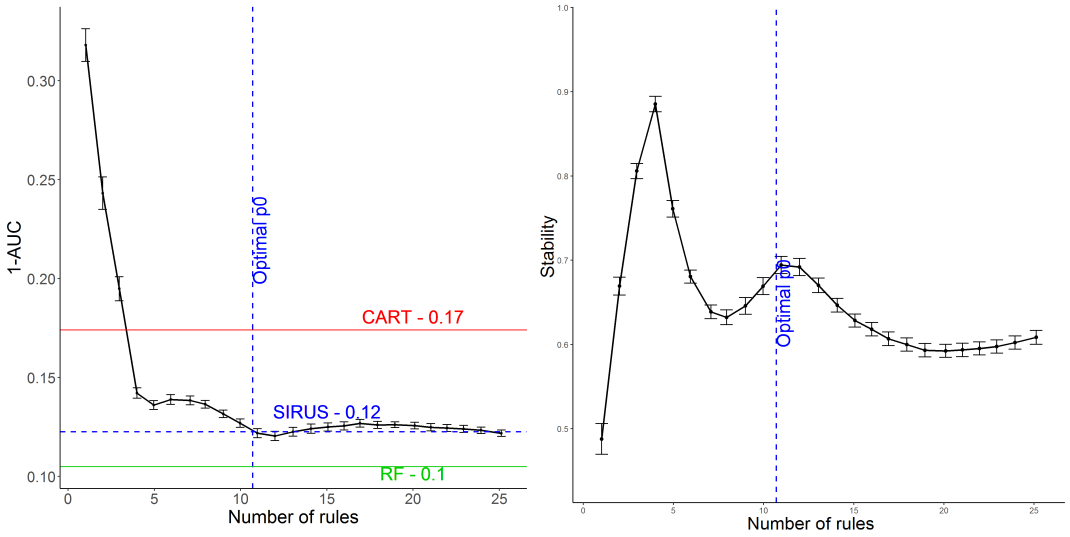

Figure 4: For the UCI dataset "Heart Statlog", 1-AUC (on the left) and stability (on the right) versus the number of rules when p_0 varies, estimated via 10-fold cross-validation (results are averaged over 30 repetitions).

Dataset	RuleFit	Node Harvest	BRL	SIRUS
Haberman	1.6	25.5	2.7	3.3
Diabetes	26.1	37.0	5.9	7.9
Heart Statlog	18.2	36.8	3.3	10.7
Liver Disorders	17.3	29.5	2.8	18.5
Heart C2	22.6	32.9	3.1	22.5
Heart H2	11.31	28.4	2.8	12.6
Credit German	30.4	32.1	3.2	17.8
Credit Approval	15.4	24.8	3.0	19.7
Ionosphere	18.3	38.0	4.3	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.)

Dataset	RuleFit	Node Harvest	BRL	SIRUS
Haberman	0.57	0.35	**0.71**	**0.65**
Diabetes	0.21	0.38	**0.80**	**0.79**
Heart Statlog	0.18	0.31	0.34	**0.69**
Liver Disorders	0.19	0.31	0.48	**0.57**
Heart C2	0.28	0.53	**0.66**	**0.66**
Heart H2	0.23	0.37	**0.61**	**0.65**
Credit German	0.12	0.46	0.33	**0.66**
Credit Approval	0.17	0.26	0.32	**0.66**
Ionosphere	0.06	0.25	**0.78**	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.)

We also evaluated main competitors: CART, RuleFit, Node harvest, and BRL, using available R implementations, respectively `rpart` (Therneau et al., 2018), `pre` (Fokkema, 2017), `nodeharvest` (Meinshausen, 2015), and `sbrl` (Yang et al., 2017). All algorithms were run with their default settings (CART trees are pruned, RuleFit is limited to rule predictors). To compare stability of the different methods, data is binned with 10-quantiles, so that the possible rules are the same for all algorithms, and the same stability metric is used. Experimental results are gathered in Table 2 for model size, Table 3 for stability, and Table 4 for predictive accuracy.

Clearly, SIRUS is more stable than its competitors. We see that BRL exhibits a comparable stability for a few datasets and generates shorter set of rules, but at the price of a weaker predictive accuracy. RuleFit and Node harvest have a slightly better predictive accuracy than SIRUS, but they are unstable and generate longer sets of rules. Overall, the general conclusion of this first batch of experiments is that SIRUS improves stability with a predictive accuracy comparable to state-of-the-art methods.

Dataset	Random Forest	CART	RuleFit	Node Harvest	BRL	SIRUS
Haberman	0.32	0.42	**0.35**	**0.35**	**0.36**	**0.36**
Diabetes	0.17	0.21	**0.19**	**0.20**	0.25	**0.20**
Heart Statlog	0.10	0.17	**0.13**	0.15	0.23	**0.12**
Liver Disorders	0.23	0.40	**0.27**	0.31	0.44	0.36
Heart C2	0.10	0.19	**0.11**	**0.11**	0.24	**0.12**
Heart H2	0.12	0.17	**0.11**	**0.11**	0.17	**0.12**
Credit German	0.21	0.31	**0.23**	**0.25**	0.34	**0.26**
Credit Approval	0.07	0.10	**0.07**	**0.07**	0.11	0.10
Ionosphere	0.03	0.10	**0.04**	0.07	0.11	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.)

Manufacturing process data In this second batch of experiments, 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 label pass/fail (0/1) for in-house line testing. As it is always 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. The threshold p_0 and the number of trees are tuned as previously explained.

Figure 5 displays predictivity versus the number of rules when p_0 varies. 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 34 rules with a value of 0.31 for 1-AUC.

Finally, the output of SIRUS may be displayed in the simple and interpretable form of Figure 6. 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 quite 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 parameter setting.

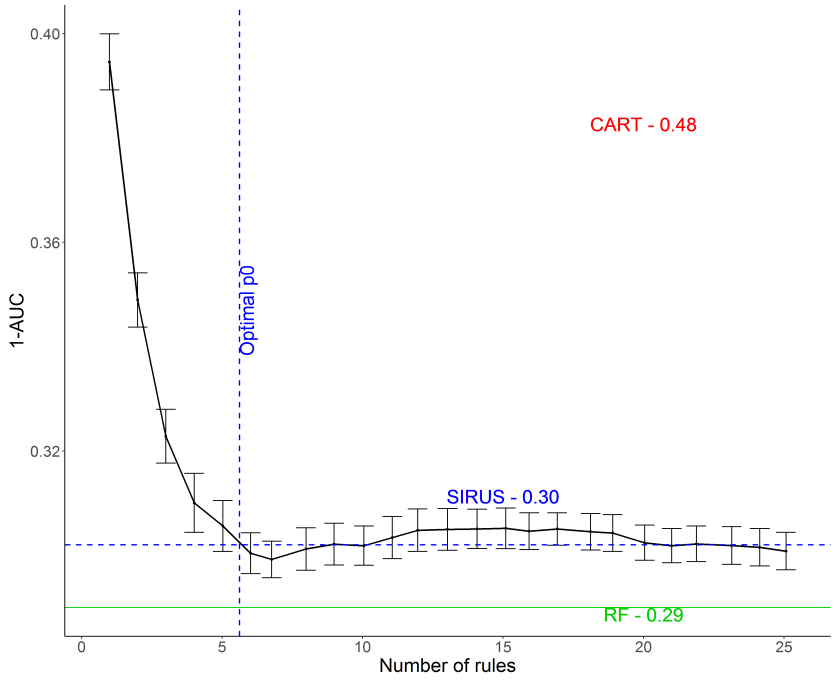

Figure 5: For the SECOM dataset, accuracy versus the number of rules when p_0 varies, estimated via 10-fold cross-validation (averaged over 30 repetitions of the cross-validation).

```
"Proportion of class 1: 0.0664"
"if X60 < 5.51 then 0.0415 else 0.166"
"if X104 < -0.00868 then 0.0392 else 0.13"
"if X349 < 0.0356 then 0.0539 else 0.178"
"if X206 < 12.7 then 0.0539 else 0.178"
"if X65 < 26.1 then 0.0546 else 0.172"
"if X60 < 5.51 & X349 < 0.0356 then 0.0346 else 0.164"
```

Figure 6: List of rules output by our software `sirus` in the R console for the SECOM dataset.

A Additional experiments and settings

This appendix specifies computational settings and provides additional experiments on the nine UCI datasets used in Section 4—see Table 1.

Rule set post-treatment As explained in Section 2, there is some redundancy in the list of rules generated by the set of distinct paths $\hat{\mathscr{P}}_{M,n,p_0}$, and a post-treatment to filter $\hat{\mathscr{P}}_{M,n,p_0}$ is needed to make the method operational. The general principle 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 to 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 2 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 $\mathscr{P}_2$ 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 $\hat{\mathscr{P}}_{M,n,p_0}$.

Random forest accuracy As described in Section 2, in the forest construction of SIRUS, the splits at each node of each tree are limited to the empirical q-quantiles of each component of $\mathbf{X}$. We then first 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—the standard deviation is displayed in parentheses in Table 5.

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.

Dataset	Mean stability
Haberman	0.950 (0.01)
Diabetes	0.950 (0.007)
Heart Statlog	0.954 (0.007)
Liver Disorders	0.951 (0.006)
Heart C2	0.955 (0.009)
Heart H2	0.952 (0.009)
Credit German	0.950 (0.008)
Credit Approval	0.941 (0.02)
Ionosphere	0.950 (0.009)

Table 6: Values of $\hat{S}_{M,n,p_0}$ averaged over $p_0 \in \hat{V}_{M,n}$ when the stopping criterion (4.1) is used to set M, for UCI datasets. Results are averaged over 10 repetitions and standard deviations are displayed in parentheses.

Definition of $\hat{V}_{M,n}$ To design the stopping criterion (4.1) of the number of trees, ε_{M,n,p_0} is averaged across a set $\hat{V}_{M,n}$ of diverse p_0 values. 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. Thus, as explained in Section 4, 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.

Number of trees We run experiments on the UCI datasets to assess the quality of the stopping criterion (4.1). Recall that the goal of this criterion is to determine the minimum number of trees M ensuring that two independent fits of SIRUS on the same dataset result on two lists of rules with an overlap of 95% in average. This is checked with a first batch of experiments—see next paragraph. Secondly, the stopping criterion (4.1) does not consider the optimal p_0, unknown when trees are grown in the first step of SIRUS. Then, another batch of experiments is run to show that the stability approximation $1 - \varepsilon_{M,n,p_0}$ is quite insensitive to p_0. Finally, a last batch of experiments provides examples of the number of trees grown when SIRUS is fit.

Experiments 1 For each dataset, the following procedure is applied. SIRUS is run a first time using criterion (4.1) to stop the number of trees. This initial run provides the optimal number of trees M as well as the set $\hat{V}_{M,n}$ of possible p_0. Then, SIRUS is fit twice independently using the precomputed number of trees M. For each $p_0 \in \hat{V}_{M,n}$, the stability metric $\hat{S}_{M,n,p_0}$ (with $\mathscr{D}'_n = \mathscr{D}_n$) is computed over the two resulting lists of rules. Finally $\hat{S}_{M,n,p_0}$ is averaged across all p_0 values in $\hat{V}_{M,n}$. This procedure is repeated 10 times: results are averaged and presented in Table 6, with standard deviations in parentheses. Across the considered datasets, resulting values range from 0.941 to 0.955, and are thus close to 0.95 as expected by construction of criterion (4.1).

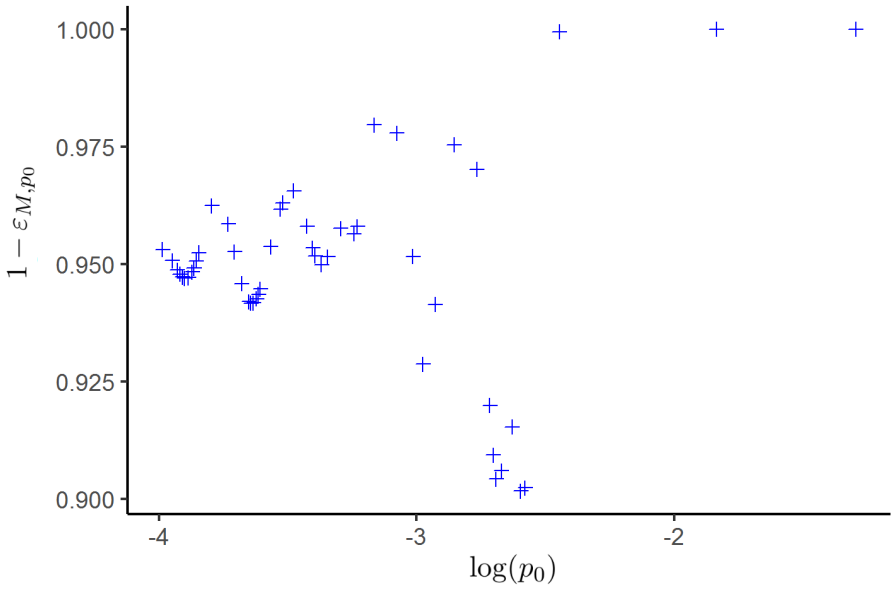

Figure 7: For the UCI dataset "Credit German", $1-\varepsilon_{M,n,p_0}$ for a sequence of $p_0 \in \hat{V}_{M,p_0}$ corresponding to final models ranging from 1 to about 25 rules.

Experiments 2 The second type of experiments illustrates that ε_{M,n,p_0} is quite insensitive to p_0 when M is set with criterion (4.1). For the "Credit German" dataset, we fit SIRUS and then compute $1-\varepsilon_{M,n,p_0}$ for each $p_0 \in \hat{V}_{M,n}$. Results are displayed in Figure 7. $1-\varepsilon_{M,n,p_0}$ ranges from 0.90 to 1, where the extreme values are reached for p_0 corresponding to very small number of rules, which are not of interest when p_0 is selected to maximize predictive accuracy. Thus, $1-\varepsilon_{M,n,p_0}$ is quite concentrated around 0.95 when p_0 varies.

Experiments 3 Finally, we display in Table 7 the optimal number of trees when the growing of SIRUS is stopped using criterion (4.1). It ranges from 4220 to 20 650 trees. In Breiman's forests, the number of trees above which the accuracy cannot be significantly improved is typically 10 times lower. However SIRUS grows shallow trees, and is thus not computationally more demanding than random forests overall.

Logistic regression In Section 2, $\hat{\eta}_{M,n,p_0}(\mathbf{x})$ (2.5) is a simple average of the set of rules, defined as

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

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

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

Dataset	Nb of trees (sd)
Haberman	10 920 (877)
Diabetes	18 830 (1538)
Heart Statlog	7840 (994)
Liver Disorders	14 650 (1242)
Heart C2	6840 (1270)
Heart H2	4220 (529)
Credit German	7940 (672)
Credit Approval	20 650 (8460)
Ionosphere	7320 (487)

Table 7: Number of trees M determined by the stopping criterion (4.1) for UCI datasets. Results are averaged over 10 repetitions and standard deviations are displayed in parentheses.

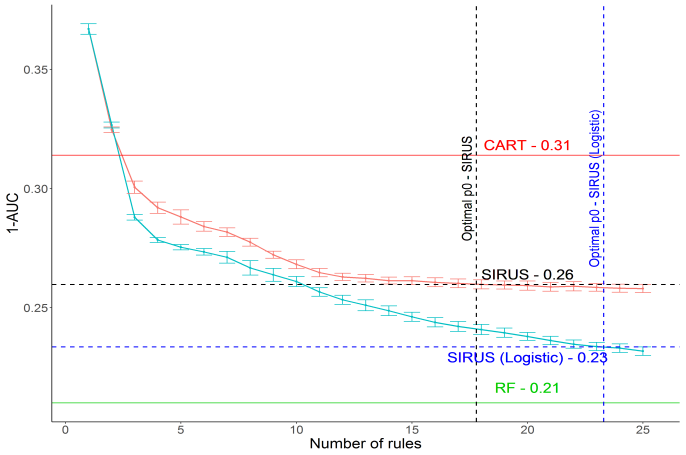

Figure 8: For the UCI dataset "Credit German", 1-AUC versus the number of rules when p_0 varies, estimated via 10-fold cross-validation (repeated 30 times) for two different methods of rule aggregation: the rule average (A.1) in red and a logistic regression (A.2) in blue.

where the coefficients $\beta_{\mathscr{P}}$ have to be estimated. To illustrate the performance of the logistic regression (A.2), we consider again the UCI dataset, "Credit German". We augment the previous results from Figure 3 (in Section 4) with the logistic regression error in Figure 8. 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

Proofs of Theorems 1 and 2 are available in **Supplementary Material for: SIRUS: Making random forests interpretable**.

References

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.

G. Biau and E. Scornet. A random forest guided tour. *Test*, 25:197–227, 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. 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. Morgan Kaufmann Publishers Inc., San Francisco, 1995.

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.

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

M. Fokkema. Pre: An r package for fitting prediction rule ensembles. *arXiv:1707.07149*, 2017.

Eibe Frank and Ian 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, B.E. Popescu, et al. 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.

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

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

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.

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

N. Meinshausen. Package 'nodeharvest', 2015.

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.

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

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, B. Atkinson, B. Ripley, and Maintainer B. Ripley. Package 'rpart', 2018.

V. Vapnik. *Statistical Learning Theory. 1998*, volume 3. 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.

Marvin N Wright and Andreas 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*, volume 70, pages 3921–3930. Proceedings of Machine Learning Research, 2017.

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. 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.