Interpretable Random Forests via Rule Extraction

Clément Bénard[1,2] **Gérard Biau**[2] **Sébastien da Veiga**[1] **Erwan Scornet**[3]

[1] Safran Tech
[2] Sorbonne Université
[3] Ecole Polytechnique
clement.benard@safrangroup.com

Abstract

We introduce SIRUS (**S**table and **I**nterpretable **RU**le **S**et) for regression, a stable rule learning algorithm, which takes the form of a short and simple list of rules. State-of-the-art learning algorithms are often referred to as "black boxes" because of the high number of operations involved in their prediction process. Despite their powerful predictivity, this lack of interpretability may be highly restrictive for applications with critical decisions at stake. On the other hand, algorithms with a simple structure—typically decision trees, rule algorithms, or sparse linear models—are well known for their instability. This undesirable feature makes the conclusions of the data analysis unreliable and turns out to be a strong operational limitation. This motivates the design of SIRUS, based on random forests, which combines a simple structure, a remarkable stable behavior when data is perturbed, and an accuracy comparable to its competitors. We demonstrate the efficiency of the method both empirically (through experiments) and theoretically (with the proof of its asymptotic stability). Our `R/C++` software implementation `sirus` is available from `CRAN`.

1 Introduction

State-of-the-art learning algorithms, such as random forests or neural networks, are often criticized for their "black-box" nature. This criticism essentially results from the high number of operations involved in their prediction mechanism, as it prevents to grasp how inputs are combined to generate predictions. Interpretability of machine learning algorithms is receiving an increasing amount of attention since the lack of transparency is a strong limitation for many applications, in particular those involving critical decisions. The analysis of production processes in the manufacturing industry typically falls into this category. Indeed, such processes involve complex physical and chemical phenomena that can often be successfully modeled by black-box learning algorithms. However, any modification of a production process has deep and long-term consequences, and therefore cannot simply result from a blind stochastic modelling. In this domain, algorithms have to be interpretable, i.e., provide a sound understanding of the relation between inputs and outputs, in order to leverage insights to guide physical analysis and improve efficiency of the production.

Although there is no agreement in the machine learning litterature about a precise definition of interpretability [23, 30], it is yet possible to define simplicity, stability, and predictivity as minimum requirements for interpretable models [2, 40]. Simplicity of the model structure can be assessed by the number of operations performed in the prediction mechanism. In particular, Murdoch et al. [30] introduce the notion of *simulatable models* when a human is able to reproduce the prediction process by hand. Secondly, Yu [39] argues that "interpretability needs stability", as the conclusions of a statistical analysis have to be robust to small data perturbations to be meaningful. Instability is the symptom of a partial and arbitrary modelling of the data, also known as the *Rashomon effect* [5]. Finally, as also explained in Breiman [5], if the decrease of predictive accuracy is significant

Preprint. Under review.

compared to a state-of-the-art black-box algorithm, the interpretable model misses some patterns in the data and is therefore misleading.

Decision trees [6] can model nonlinear patterns while having a simple structure. They are therefore often presented as interpretable. However, the structure of trees is highly sensitive to small data perturbation [5], which violates the stability principle and is thus a strong limitation to their practical use. Rule algorithms are another type of nonlinear methods with a simple structure, defined as a collection of elementary rules. An elementary rule is a set of constraints on input variables, which forms a hyperrectangle in the input space and on which the associated prediction is constant. As an example, such a rule typically takes the following simple form:

$$\textbf{If } \begin{cases} X^{(1)} < 1.12 \\ \textbf{\& } X^{(3)} \geq 0.74 \end{cases} \textbf{ then } \hat{Y} = 0.18 \textbf{ else } \hat{Y} = 4.1 \textbf{ .}$$

A large number of rule algorithms have been developed, among which the most influential Decision List [32], CN2 [8], C4.5 [31], IREP [Incremental Reduced Error Pruning, 20], RIPPER [Repeated Incremental Pruning to Produce Error Reduction, 9], PART [Partial Decision Trees, 15], SLIPPER [Simple Learner with Iterative Pruning to Produce Error Reduction, 10], LRI [Leightweight Rule Induction, 37], RuleFit [19], Node harvest [27], ENDER [Ensemble of Decision Rules, 11], BRL [Bayesian Rule Lists, 22], RIPE [Rule Induction Partitioning Estimator, 25, 26], and Wei et al. [36, Generalized Linear Rule Models]. It turns out, however, that despite their simplicity and high predictivity (close to the accuracy of tree ensembles), rule learning algorithms share the same limitation as decision trees: instability. Furthermore, among the hundreds of existing rule algorithms, most of them are designed for supervised classification and few have the ability to handle regression problems.

The purpose of this article is to propose a new stable rule algorithm for regression, SIRUS (**S**table and **I**nterpretable **RU**le **S**et), and therefore demonstrate that rule methods can address regression problems efficiently while producing compact and stable list of rules. To this aim, we build on Bénard et al. [2], who have introduced SIRUS for classification problems. Our algorithm is based on random forests [4], and its general principle is as follows: since each node of each tree of a random forest can be turned into an elementary rule, the core idea is to extract rules from a tree ensemble based on their frequency of appearance. The most frequent rules, which represent robust and strong patterns in the data, are ultimately linearly combined to form predictions. The main competitors of SIRUS are RuleFit [19] and Node harvest [27]. Both methods also extract large collection of rules from tree ensembles: RuleFit uses a boosted tree ensemble [ISLE, 18] whereas Node harvest is based on random forests. The rule selection is performed by a sparse linear aggregation, respectively the Lasso [34] for RuleFit and a constrained quadratic program for Node harvest. Yet, despite their powerful predictive skills, these two methods tend to produce long, complex, and unstable lists of rules (typically of the order of $30 - 50$), which makes their interpretability questionable. Because of the randomness in the tree ensemble, running these algorithms multiple times on the same dataset outputs different rule lists. As we will see, SIRUS considerably improves stability and simplicity over its competitors, while preserving a comparable predictive accuracy and computational complexity—see Section 2 of the Supplementary Material for the complexity analysis.

We present SIRUS algorithm in Section 2. In Section 3, experiments illustrate the good performance of our algorithm in various settings. Section 4 is devoted to studying the theoretical properties of the method, with, in particular, a proof of its asymptotic stability. Finally, Section 5 summarizes the main results and discusses research directions for future work. Additional details are gathered in the Supplementary Material.

2 SIRUS Algorithm

We consider a standard regression setting where we observe an i.i.d. sample $\mathscr{D}_n = \{(\mathbf{X}_i, Y_i), i = 1, \dots, n\}$, with each $(\mathbf{X}_i, Y_i)$ distributed as a generic pair $(\mathbf{X}, Y)$ independent of $\mathscr{D}_n$. The p-tuple $\mathbf{X} = (X^{(1)}, \dots, X^{(p)})$ is a random vector taking values in $\mathbb{R}^p$, and $Y \in \mathbb{R}$ is the response. Our objective is to estimate the regression function $m(\mathbf{x}) = \mathbb{E}(Y|\mathbf{X} = \mathbf{x})$ with a small and stable set of rules.

Rule generation The **first step** of SIRUS is to grow a random forest with a large number M of trees based on the available sample $\mathscr{D}_n$. The critical feature of our approach to stabilize the forest structure is to restrict node splits to the q-empirical quantiles of the marginals $X^{(1)}, \ldots, X^{(p)}$, with typically $q = 10$. This modification to Breiman's original algorithm has a small impact on predictive accuracy, but is essential for stability, as it is extensively discussed in Section 3 of the Supplementary Material. Next, the obtained forest is broken down in a large collection of rules in the following process. First, observe that each node of each tree of the resulting ensemble defines a hyperrectangle in the input space $\mathbb{R}^p$. Such a node can therefore be turned into an elementary regression rule, by defining a piecewise constant estimate whose value only depends on whether the query point falls in the hyperrectangle or not. Formally, a (inner or terminal) node of the tree is represented by a path, say $\mathscr{P}$, which describes the sequence of splits to reach the node from the root of the tree. In the sequel, we denote by Π the finite list of all possible paths, and insist that each path $\mathscr{P} \in \Pi$ defines a regression rule. Based on this principle, in the first step of the algorithm, both internal and external nodes are extracted from the trees of the random forest to generate a large collection of rules, typically 10^4.

Rule selection The **second step** of SIRUS is to select the relevant rules from this large collection. Despite the tree randomization in the forest construction, there are some redundancy in the extracted rules. Indeed those with a high frequency of appearance represent strong and robust patterns in the data, and are therefore good candidates to be included in a compact, stable, and predictive rule ensemble. This occurrence frequency is denoted by $\hat{p}_{M,n}(\mathscr{P})$ for each possible path $\mathscr{P} \in \Pi$. Then a threshold $p_0 \in (0, 1)$ is simply used to select the relevant rules, that is

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

The threshold p_0 is a tuning parameter, whose influence and optimal setting are discussed and illustrated later in the experiments (Figures 1 and 2). Optimal p_0 values essentially select rules made of one or two splits. Indeed, rules with a higher number of splits are more sensitive to data perturbation, and thus associated to smaller values of $\hat{p}_{M,n}(\mathscr{P})$. Therefore, SIRUS grows shallow trees to reduce the computational cost while leaving the rule selection untouched—see Section 3 of the Supplementary Material. In a word, SIRUS uses the principle of randomized bagging, but aggregates the forest structure itself instead of predictions in order to stabilize the rule selection.

Rule set post-treatment The rules associated with the set of distinct paths $\hat{\mathscr{P}}_{M,n,p_0}$ are dependent by definition of the path extraction mechanism. As an example, let us consider the 6 rules extracted from a random tree of depth 2. Since the tree structure is recursive, 2 rules are made of one split and 4 rules of two splits. Those 6 rules are linearly dependent because their associated hyperrectangles overlap. Consequently, to properly settle a linear aggregation of the rules, the **third step** of SIRUS filters $\hat{\mathscr{P}}_{M,n,p_0}$ with the following post-treatment procedure: if the rule induced by 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 of appearance, then $\mathscr{P}$ is simply removed from $\hat{\mathscr{P}}_{M,n,p_0}$.

Rule aggregation By following the previous steps, we finally obtain a small set of regression rules. As such, a rule $\hat{g}_{n,\mathscr{P}}$ associated with a path $\mathscr{P}$ is a piecewise constant estimate: if a query point $\mathbf{x}$ falls into the corresponding hyperrectangle $H_{\mathscr{P}} \subset \mathbb{R}^p$, the rule returns the average of the Y_i's for the training points $\mathbf{X}_i$'s that belong to $H_{\mathscr{P}}$; symmetrically, if $\mathbf{x}$ falls outside of $H_{\mathscr{P}}$, the average of the Y_i's for training points outside of $H_{\mathscr{P}}$ is returned. Next, a non-negative weight is assigned to each of the selected rule, in order to combine them into a single estimate of $m(\mathbf{x})$. These weights are defined as the ridge regression solution, where each predictor is a rule $\hat{g}_{n,\mathscr{P}}$ for $\mathscr{P} \in \hat{\mathscr{P}}_{M,n,p_0}$ and weights are constrained to be non-negative. Thus, the aggregated estimate $\hat{m}_{M,n,p_0}(\mathbf{x})$ of $m(\mathbf{x})$ computed in the **fourth step** of SIRUS has the form

$$\hat{m}_{M,n,p_0}(\mathbf{x}) = \hat{\beta}_0 + \sum_{\mathscr{P} \in \hat{\mathscr{P}}_{M,n,p_0}} \hat{\beta}_{n,\mathscr{P}} \hat{g}_{n,\mathscr{P}}(\mathbf{x}), \tag{2.1}$$

where $\hat{\beta}_0$ and $\hat{\beta}_{n,\mathscr{P}}$ are the solutions of the ridge regression problem. More precisely, denoting by $\hat{\boldsymbol{\beta}}_{n,p_0}$ the column vector whose components are the coefficients $\hat{\beta}_{n,\mathscr{P}}$ for $\mathscr{P} \in \hat{\mathscr{P}}_{M,n,p_0}$, and letting $\mathbf{Y} = (Y_1, \ldots, Y_n)^T$ and $\boldsymbol{\Gamma}_{n,p_0}$ the matrix whose rows are the rule values $\hat{g}_{n,\mathscr{P}}(\mathbf{X}_i)$ for

$i \in \{1, \dots, n\}$, we have

$$(\hat{\boldsymbol{\beta}}_{n,p_0}, \hat{\beta}_0) = \underset{\boldsymbol{\beta} \geq 0, \beta_0}{\operatorname{argmin}} \frac{1}{n} ||\mathbf{Y} - \beta_0 \mathbf{1_n} - \boldsymbol{\Gamma}_{n,p_0} \boldsymbol{\beta}||_2^2 + \lambda ||\boldsymbol{\beta}||_2^2,$$

where $\mathbf{1_n} = (1, \dots, 1)^T$ is the n-vector with all components equal to 1, and λ is a positive parameter tuned by cross-validation that controls the penalization severity. The mininum is taken over $\beta_0 \in \mathbb{R}$ and all the vectors $\boldsymbol{\beta} = \{\beta_1, \dots, \beta_{c_n}\} \in \mathbb{R}_+^{c_n}$ where $c_n = |\hat{\mathscr{P}}_{M,n,p_0}|$ is the number of selected rules. Besides, notice that the rule format with an else clause differs from the standard format in the rule learning literature. This modification provides good properties of stability and modularity (investigation of the rules one by one [30]) to SIRUS—see Section 4 of the Supplementary Material.

This linear rule aggregation is a critical step and deserves additional comments. Indeed, in RuleFit, the rules are also extracted from a tree ensemble, but aggregated using the Lasso. However, the extracted rules are strongly correlated by construction, and the Lasso selection is known to be highly unstable in such correlated setting. This is the main reason of the instability of RuleFit, as the experiments will show. On the other hand, the sparsity of SIRUS is controlled by the parameter p_0, and the ridge regression enables a stable aggregation of the rules. Furthermore, the constraint $\boldsymbol{\beta} \geq 0$ is added to ensure that all coefficients are non-negative, as in Node harvest [27]. Also because of the rule correlation, an unconstrained regression would lead to negative values for some of the coefficients $\hat{\beta}_{n,\mathscr{P}}$, and such behavior drastically undermines the interpretability of the algorithm.

Interpretability As stated in the introduction, despite the lack of a precise definition of interpretable models, there are three minimum requirements to be taken into account: simplicity, stability, and predictivity. These notions need to be formally defined and quantified to enable comparison between algorithms. **Simplicity** refers to the model complexity, in particular the number of operations involved in the prediction mechanism. In the case of rule algorithms, a measure of simplicity is naturally given by the number of rules. Intuitively, a rule algorithm is **stable** when two independent estimations based on two independent samples return similar lists of rules. Formally, let $\hat{\mathscr{P}}'_{M,n,p_0}$ be the list of rules output by SIRUS fit on an independent sample $\mathscr{D}'_n$. Then the proportion of rules shared by $\hat{\mathscr{P}}_{M,n,p_0}$ and $\hat{\mathscr{P}}'_{M,n,p_0}$ gives a stability measure. Such a metric is known as the Dice-Sorensen index, and is often used to assess variable selection procedures [7, 41, 3, 21, 1]. In our case, the Dice-Sorensen index is then defined as

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

However, in practice one rarely has access to an additional sample $\mathscr{D}'_n$. Therefore, to circumvent this problem, we use a 10-fold cross-validation to simulate data perturbation. The stability metric is thus empirically defined as the average proportion of rules shared by two models of two distinct folds of the cross-validation. A stability of 1 means that the exact same list of rules is selected over the 10 folds, whereas a stability of 0 means that all rules are distinct between any 2 folds. For **predictivity** in regression problems, the proportion of unexplained variance is a natural measure of the prediction error. The estimation is performed by 10-fold cross-validation.

3 Experiments

Experiments are run over 8 diverse public datasets to demonstrate the improvement of SIRUS over state-of-the-art methods. Table 1 in Section 5 of the Supplementary Material provides dataset details.

SIRUS rule set Our algorithm is illustrated on the "LA Ozone" dataset from Friedman et al. [16], which records the level of atmospheric ozone concentration from eight daily meteorological measurements made in Los Angeles in 1976: wind speed ("wind"), humidity ("humidity"), temperature ("temp"), inversion base height ("ibh"), daggot pressure gradient ("dpg"), inversion base temperature ("ibt"), visibility ("vis"), and day of the year ("doy"). The response "Ozone" is the log of the daily maximum of ozone concentration. The list of rules output for this dataset is presented in Table 1. The column "Frequency" refers to $\hat{p}_{M,n}(\mathscr{P})$, the occurrence frequency of each rule in the forest, used for rule selection. It enables to grasp how weather conditions impact the ozone concentration. In particular, a temperature larger than 65°F or a high inversion base temperature result in high ozone concentrations. The third rule tells us that the interaction of a high temperature with a visibility

Average Ozone $= 12$			**Intercept** $= -7.8$				
Frequency		**Rule**					**Weight**
0.29	**if**	temp < 65	**then**	Ozone $= 7$	**else**	Ozone $= 19$	0.12
0.17	**if**	ibt < 189	**then**	Ozone $= 7$	**else**	Ozone $= 18$	0.07
0.063	**if**	temp ≥ 65 **&** vis < 150	**then**	Ozone $= 20$	**else**	Ozone $= 7$	0.31
0.061	**if**	vh < 5840	**then**	Ozone $= 10$	**else**	Ozone $= 20$	0.072
0.060	**if**	ibh < 2110	**then**	Ozone $= 16$	**else**	Ozone $= 7$	0.14
0.058	**if**	ibh < 2960	**then**	Ozone $= 15$	**else**	Ozone $= 6$	0.10
0.051	**if**	temp ≥ 65 **&** ibh < 2110	**then**	Ozone $= 21$	**else**	Ozone $= 8$	0.16
0.048	**if**	vis < 150	**then**	Ozone $= 14$	**else**	Ozone $= 7$	0.18
0.043	**if**	temp < 65 **&** ibt < 120	**then**	Ozone $= 5$	**else**	Ozone $= 15$	0.15
0.040	**if**	temp < 70	**then**	Ozone $= 8$	**else**	Ozone $= 20$	0.14
0.039	**if**	ibt < 227	**then**	Ozone $= 9$	**else**	Ozone $= 22$	0.21

Table 1: SIRUS rule list for the "LA Ozone" dataset.

lower than 150 miles generates even higher levels of ozone concentration. Interestingly, according to the ninth rule, especially low ozone concentrations are reached when a low temperature and and a low inversion base temperature are combined. Recall that to generate a prediction for a given query point $\mathbf{x}$, for each rule the corresponding ozone concentration is retrieved depending on whether $\mathbf{x}$ satisfies the rule conditions. Then all rule outputs for $\mathbf{x}$ are multiplied by their associated weight and added together. One can observe that rule importances and weights are not related. For example, the third rule has a higher weight than the most two important ones. It is clear that rule 3 has multiple constraints and is therefore more sensitive to data perturbation—hence a smaller frequency of appearance in the forest. On the other hand, its associated variance decrease in CART is more important than for the first two rules, leading to a higher weight in the linear combination. Since rules 5 and 6 are strongly correlated, their weights are diluted.

Tuning SIRUS has only one hyperparameter which requires fine tuning: the threshold p_0 to control the model size by selecting the most frequent rules in the forest. First, the range of possible values of p_0 is set so that the model size varies between 1 and 25 rules. This arbitrary upper bound is a safeguard to avoid long and complex list of rules that are difficult to interpret. In practice, this limit of 25 rules is rarely hit, since the following tuning of p_0 naturally leads to compact rule lists. Thus, p_0 is tuned within that range by cross-validation to maximize both stability and predictivity. To find a tradeoff between these two properties, we follow a standard bi-objective optimization procedure described and illustrated in Section 2 of the Supplementary Material: p_0 is chosen to be as close as possible to the ideal case of 0 unexplained variance and 90% stability. This tuning procedure is computationally fast: the cost of about 10 fits of SIRUS. Besides, the optimal number of trees M is set automatically by SIRUS: as stability, predictivity, and computation time increase with the number of trees, no fine tuning is required for M. Thus, a stopping criterion is designed to grow the minimum number of trees which enforces that stability and predictivity are greater than 95% of their maximum values (reached when $M \to \infty$)—see Section 6 of the Supplementary Material for a detailed definition of this criterion. Finally, we use the other standard settings of random forests (well-known for their excellent performance), set $q = 10$ quantiles, and transform categorical variables into multiple binary variables.

Performance We compare SIRUS with its two main competitors RuleFit (with rule predictors only) and Node harvest. For predictive accuracy, we ran random forests and (pruned) CART to provide the baseline. Only to compute stability metrics, data is binned using 10 quantiles to fit Rulefit and Node harvest. Our `R`/`C++` package `sirus` (available from `CRAN`) is adapted from `ranger`, a fast random forests implementation [38]. We also use available `R` implementations `pre` [14, RuleFit] and `nodeharvest` [28]. While the predictive accuracy of SIRUS is comparable to Node harvest and slightly below RuleFit, the stability is considerably improved with much smaller rule lists. Experimental results are gathered in Table 2a for model sizes, Table 2b for stability, and Table 3 for

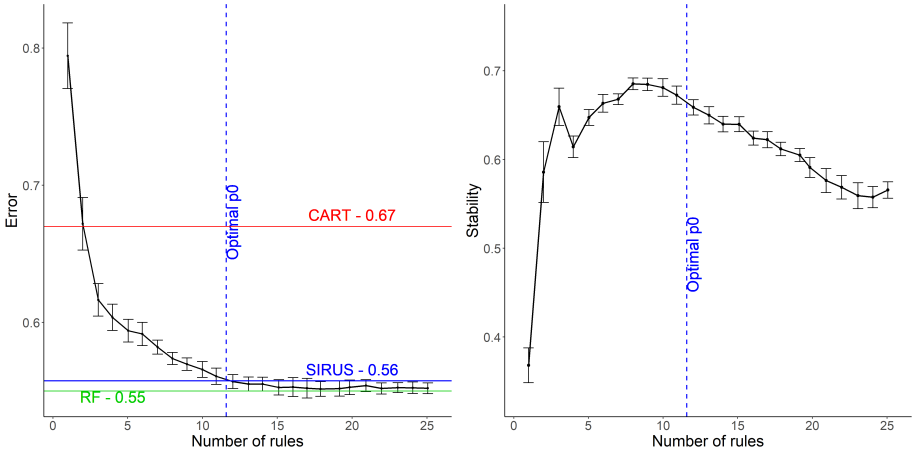

Figure 1: For the dataset "Diabetes", unexplained variance (left panel) and stability (right panel) versus the number of rules when p_0 varies, estimated via 10-fold cross-validation (results are averaged over 10 repetitions).

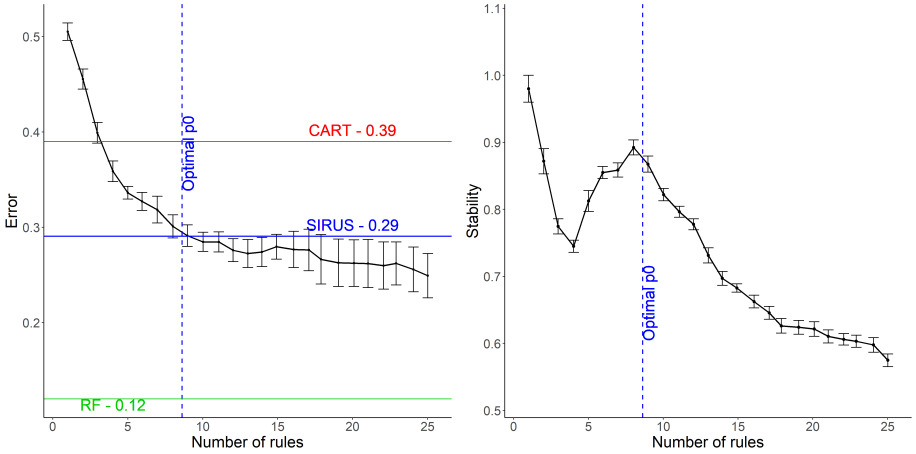

Figure 2: For the dataset "Machine", unexplained variance (left panel) and stability (right panel) versus the number of rules when p_0 varies, estimated via 10-fold cross-validation (results are averaged over 10 repetitions).

predictive accuracy. All results are averaged over 10 repetitions of the cross-validation procedure. Since standard deviations are negligible, they are not displayed to increase readability. Besides, in the last column of Table 3, p_0 is set to increase the number of rules in SIRUS to reach RuleFit and Node harvest model size (about 50 rules): predictivity is then as good as RuleFit.

To illustrate the typical behavior of our method, we comment the results for two specific datasets: "Diabetes" [13] and "Machine" [12]. The "Diabetes" data contains $n = 442$ diabetic patients and the response of interest Y is a measure of disease progression over one year. A total of 10 variables are collected for each patient: age, sex, body mass index, average blood pressure, and six blood serum measurements $s1, s2, \ldots, s6$. For this dataset, SIRUS is as predictive as a random forest, with only 12 rules when the forest performs about 10^4 operations: the unexplained variance is 0.56 for SIRUS and 0.55 for random forest. Notice that CART performs considerably worse with 0.67 unexplained variance. For the second dataset, "Machine", the output Y of interest is the CPU performance of computer hardware. For $n = 209$ machines, 7 variables are collected about the machine characteristics. For this dataset, SIRUS, RuleFit, and Node harvest have a similar predictivity, in-between CART and random forests. Our algorithm achieves such performance with a readable list of only 9 rules stable at 88%, while RuleFit and Node harvest incorporate respectively 44 and 42 rules with stability levels of 23% and 29%. Stability and predictivity are represented as p_0 varies for "Diabetes" and "Machine" datasets in Figures 1 and 2, respectively.

(a) Model Size

Dataset	CART	RuleFit	Node Harvest	SIRUS
Ozone	15	21	46	**11**
Mpg	15	40	43	**9**
Prostate	**11**	14	41	23
Housing	15	54	40	**6**
Diabetes	**12**	25	42	**12**
Machine	**8**	44	42	9
Abalone	20	58	35	**6**
Bones	17	5	13	**1**

(b) Stability

Dataset	RuleFit	Node Harvest	SIRUS
Ozone	0.22	0.30	**0.62**
Mpg	0.25	0.43	**0.83**
Prostate	0.32	0.23	**0.48**
Housing	0.19	0.40	**0.80**
Diabetes	0.18	0.39	**0.66**
Machine	0.23	0.29	**0.88**
Abalone	0.31	0.38	**0.82**
Bones	0.59	0.52	**0.89**

Table 2: Mean model size and stability over a 10-fold cross-validation for various public datasets.

Dataset	Random Forest	CART	RuleFit	Node Harvest	SIRUS	SIRUS 50 Rules
Ozone	0.25	0.36	**0.27**	0.31	0.32	**0.26**
Mpg	0.13	0.20	**0.15**	0.20	0.21	**0.15**
Prostate	0.48	0.60	**0.53**	**0.52**	**0.48**	0.55
Housing	0.13	0.28	**0.16**	0.24	0.31	0.21
Diabetes	0.55	0.67	**0.55**	**0.58**	**0.56**	**0.54**
Machine	0.13	0.39	**0.26**	**0.29**	**0.29**	**0.27**
Abalone	0.44	0.56	**0.46**	0.61	0.66	0.64
Bones	0.67	0.67	**0.70**	**0.70**	**0.74**	**0.72**

Table 3: Proportion of unexplained variance estimated over a 10-fold cross-validation for various public datasets. For rule algorithms only, i.e., RuleFit, Node harvest, and SIRUS, maximum values are displayed in bold, as well as values within 10% of the maximum for each dataset.

4 Theoretical Analysis

Among the three minimum requirements for interpretable models, stability is the critical one. In SIRUS, simplicity is explicitly controlled by the hyperparameter p_0. The wide literature on rule learning provides many experiments to show that rule algorithms have an accuracy comparable to tree ensembles. On the other hand, designing a stable rule procedure is more challenging [22, 30]. For this reason, we therefore focus our theoretical analysis on the asymptotic stability of SIRUS.

To get started, we need a rigorous definition of the rule extraction procedure. To this aim, we introduce a symbolic representation of a path in a tree, which describes the sequence of splits to reach a given (inner or terminal) node from the root. We insist that such path encoding can be used in both the empirical and theoretical algorithms to define rules. A path $\mathscr{P}$ is defined as

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

where, for $k \in \{1, \dots, d\}$ ($d \in \{1, 2\}$), 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 $j_k \in \{1, \dots, p\}$, the index $r_k \in \{1, \dots, q-1\}$ of the corresponding quantile, and a side $s_k = L$ if we go to the left and $s_k = R$ if we go to the right—see Figure 3. The set of all possible such paths is denoted by Π. Each tree of the forest is randomized in two ways: (i) the sample $\mathscr{D}_n$ is bootstrapped prior to the construction of the tree, and (ii) a subset of coordinates is randomly selected to find the best split at each node. This randomization mechanism is governed by a random variable that we call Θ. We define $T(\Theta, \mathscr{D}_n)$, a random subset of Π, as the collection of the extracted paths from the random tree built with Θ and $\mathscr{D}_n$. Now, let $\Theta_1, \dots, \Theta_\ell, \dots, \Theta_M$ be the independent randomizations of the M trees of the forest. With this notation, the empirical frequency of occurrence of a path $\mathscr{P} \in \Pi$ in the forest takes the form

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

which is simply the proportion of trees that contain $\mathscr{P}$. By definition, $\hat{p}_{M,n}(\mathscr{P})$ is the Monte Carlo estimate of the probability $p_n(\mathscr{P})$ that a Θ-random tree contains a particular path $\mathscr{P} \in \Pi$, that is,

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

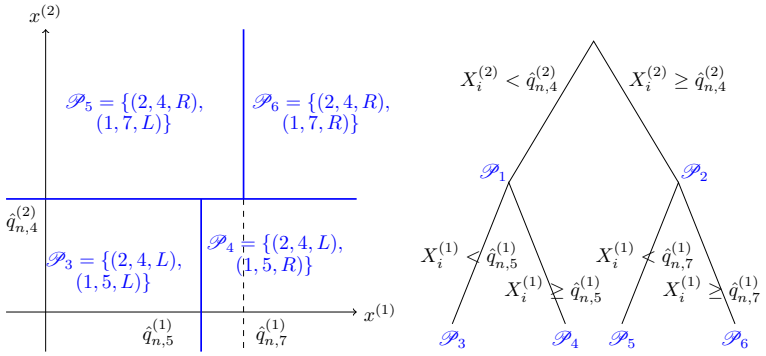

Figure 3: Example of a root node $\mathbb{R}^2$ partitioned by a randomized tree of depth 2: the tree on the right, the associated paths and hyperrectangles of length $d = 2$ on the left.

Next, we introduce all theoretical counterparts of the empirical quantities involved in SIRUS, which do not depend on the sample $\mathscr{D}_n$ but only on the unknown distribution of $(\mathbf{X}, Y)$. We let $T^\star(\Theta)$ be the list of all paths contained in the theoretical tree built with randomness Θ, in which splits are chosen to maximize the theoretical CART-splitting criterion instead of the empirical one. The probability $p^\star(\mathscr{P})$ that a given path $\mathscr{P}$ belongs to a theoretical randomized tree (the theoretical counterpart of $p_n(\mathscr{P})$) is

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

We finally define the theoretical set of selected paths $\mathscr{P}^\star_{p_0} = \{\mathscr{P} \in \Pi : p^\star(\mathscr{P}) > p_0\}$ (with the same post-treatment as for the data-based procedure—see Section 2—to remove linear dependence between rules, and discarding paths with a null coefficient in the rule aggregation). As it is often the case in the theoretical analysis of random forests, [33, 29], we assume throughout this section that the subsampling of a_n observations prior to each tree construction is done without replacement to alleviate the mathematical analysis. Our stability result holds under the following mild assumptions:

(A1) The subsampling rate a_n satisfies $\lim_{n\to\infty} a_n = \infty$ and $\lim_{n\to\infty} \frac{a_n}{n} = 0$, and the number of trees M_n satisfies $\lim_{n\to\infty} M_n = \infty$.

(A2) The random variable $\mathbf{X}$ has a strictly positive density f with respect to the Lebesgue measure on $\mathbb{R}^p$. Furthermore, for all $j \in \{1, \ldots, p\}$, the marginal density $f^{(j)}$ of $X^{(j)}$ is continuous, bounded, and strictly positive. Finally, the random variable Y is bounded.

Theorem 1. *Assume that Assumptions (A1) and (A2) are satisfied, and let* $\mathcal{U}^\star = \{p^\star(\mathscr{P}) : \mathscr{P} \in \Pi\}$ *be the set of all theoretical probabilities of appearance for each path* $\mathscr{P}$*. Then, provided* $p_0 \in [0,1] \setminus \mathcal{U}^\star$ *and* $\lambda > 0$*, we have*

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

Theorem 1 states that SIRUS is stable: provided that the sample size is large enough, the same list of rules is systematically output across several fit on independent samples. The analysis conducted in the proof—Section 1 of the Supplementary Material—highlights that the cut discretization (performed at quantile values only), as well as considering random forests (instead of boosted tree ensembles as in RuleFit) are the cornerstones to stabilize rule models extracted from tree ensembles. Furthermore, the experiments in Section 3 show the high empirical stability of SIRUS in finite-sample regimes.

5 Conclusion

Interpretability of machine learning algorithms is required whenever the targeted applications involve critical decisions. Although interpretability does not have a precise definition, we argued that simplicity, stability, and predictivity are minimum requirements for interpretable models. In this context, rule algorithms are well known for their good predictivity and simple structures, but also to be often highly unstable. Therefore, we proposed a new regression rule algorithm called SIRUS, whose general principle is to extract rules from random forests. Our algorithm exhibits an accuracy comparable to state-of-the-art rule algorithms, while producing much more stable and shorter list of rules. This remarkably stable behavior is theoretically understood since the rule selection is consistent. Our `R/C++` software implementation `sirus` is available from `CRAN`.

Broader Impact

This contribution falls in the field of interpretable machine learning. One of the main objective of this research area is to improve the understanding of machine learning algorithms, and potential positive impacts are numerous. As an example, this can help to detect unexpected behaviors of a machine learning system. More precisely, we hope that this work will contribute to advocate stability as a critical property for interpretable models. Secondly, since our proposed method is generic, it can be applied to any kind of data. Thus, the true impact of our work relies on the specific application it is used for. It is likely to be positive: for example, rule algorithms have many applications in healthcare. However, we acknowledge that the impact might also be negative if used with unethical motivations.

References

[1] 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.

[2] C. Bénard, G. Biau, S. Da Veiga, and E. Scornet. SIRUS: Making random forests interpretable. *arXiv:1908.06852*, 2019.

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

[4] L. Breiman. Random forests. *Machine Learning*, 45:5–32, 2001.

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

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

[7] 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.

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

[9] W.W. Cohen. Fast effective rule induction. In *Proceedings of the 12th International Conference on Machine Learning*, pages 115–123. Morgan Kaufmann Publishers Inc., San Francisco, 1995.

[10] W.W. Cohen and Y. Singer. A simple, fast, and effective rule learner. In *Proceedings of the 16th National Conference on Artificial Intelligence and 11th Conference on Innovative Applications of Artificial Intelligence*, pages 335–342, Palo Alto, 1999. AAAI Press.

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

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

[13] B. Efron, T. Hastie, I. Johnstone, and R. Tibshirani. Least angle regression. *The Annals of statistics*, 32:407–499, 2004.

[14] M. Fokkema. PRE: An R package for fitting prediction rule ensembles. *arXiv:1707.07149*, 2017.

[15] E. Frank and I. H. Witten. Generating accurate rule sets without global optimization. In *Proceedings of the 15th International Conference on Machine Learning*, pages 144–151, San Francisco, 1998. Morgan Kaufmann Publishers Inc.

[16] J. Friedman, T. Hastie, and R. Tibshirani. *The Elements of Statistical Learning*, volume 1. Springer, New York, 2001.

[17] J. Friedman, T. Hastie, and R. Tibshirani. Regularization paths for generalized linear models via coordinate descent. *Journal of statistical software*, 33(1):1, 2010.

[18] J.H. Friedman and B.E. Popescu. Importance sampled learning ensembles. *Journal of Machine Learning Research*, 94305:1–32, 2003.

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

[20] 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.

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

[22] 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.

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

[24] G. Louppe. Understanding random forests: From theory to practice. *arXiv preprint arXiv:1407.7502*, 2014.

[25] V. Margot, J.-P. Baudry, F. Guilloux, and O. Wintenberger. Rule induction partitioning estimator, 2018.

[26] V. Margot, J.-P. Baudry, F. Guilloux, and O. Wintenberger. Consistent regression using data-dependent coverings. *arXiv:1907.02306*, 2019.

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

[28] N. Meinshausen. Package 'nodeharvest', 2015. URL `https://cran.r-project.org/web/packages/nodeHarvest/`.

[29] 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.

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

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

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

[33] E. Scornet, G. Biau, and J.-P. Vert. Consistency of random forests. *The Annals of Statistics*, 43 (4):1716–1741, 2015.

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

[35] A.W. Van der Vaart. *Asymptotic statistics*, volume 3. Cambridge university press, 2000.

[36] D. Wei, S. Dash, T. Gao, and O. Günlük. Generalized linear rule models. *arXiv preprint arXiv:1906.01761*, 2019.

[37] S.M. Weiss and N. Indurkhya. Lightweight rule induction. In *Proceedings of the 17th International Conference on Machine Learning*, pages 1135–1142, San Francisco, 2000. Morgan Kaufmann Publishers Inc.

[38] 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.

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

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

[41] 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.

Supplementary Material For: Interpretable Random Forests via Rule Extraction

1 Proof of Theorem 1: Asymptotic Stability

Proof of Theorem 1. We recall that stability is assessed by the Dice-Sorensen index as

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

where $\hat{\mathscr{P}}'_{M,n,p_0}$ stands for the list of rules output by SIRUS fit with an independent sample $\mathscr{D}'_n$ and where the random forest is parameterized by independent copies $\Theta'_1, \ldots, \Theta'_M$.

We consider $p_0 \in [0,1] \setminus \mathcal{U}^\star$ and $\lambda > 0$. There are two sources of randomness in the estimation of the final set of selected paths: (i) the path extraction from the random forest based on $\hat{p}_{M,n}(\mathscr{P})$ for $\mathscr{P} \in \Pi$, and (ii) the final sparse linear aggregation of the rules through the estimate $\hat{\boldsymbol{\beta}}_{n,p_0}$. To show that the stability converges to 1, these estimates have to converge towards theoretical quantities that are independent of $\mathscr{D}_n$. Note that, throughout the paper, the final set of selected paths is denoted $\hat{\mathscr{P}}_{M_n,n,p_0}$. Here, for the sake of clarity, $\hat{\mathscr{P}}_{M_n,n,p_0}$ is now the post-treated set of paths extracted from the random forest, and $\hat{\mathscr{P}}_{M_n,n,p_0,\lambda}$ the final set of selected paths in the ridge regression.

(i) **Path extraction** The first step of the proof is to show that the post-treated path extraction from the forest is consistent, i.e., in probability

$$\lim_{n\to\infty} \mathbb{P}(\hat{\mathscr{P}}_{M_n,n,p_0} = \mathscr{P}^\star_{p_0}) = 1. \tag{1.1}$$

Using the continuous mapping theorem, it is easy to see that this result is a consequence of the consistency of $\hat{p}_{M,n}(\mathscr{P})$, i.e.,

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

Since the output Y is bounded (by Assumption (A2)), the consistency of $\hat{p}_{M,n}(\mathscr{P})$ can be easily adapted from Theorem 1 of Bénard et al. [2] using Assumptions (A1) and (A2). Finally, the result still holds for the post-treated rule set because the post-treatment is a deterministic procedure.

(ii) **Sparse linear aggregation** Recall that the estimate $(\hat{\boldsymbol{\beta}}_{n,p_0}, \hat{\beta}_0)$ is defined as

$$(\hat{\boldsymbol{\beta}}_{n,p_0}, \hat{\beta}_0) = \underset{\boldsymbol{\beta}\geq 0, \beta_0}{\operatorname{argmin}}\, \ell_n(\boldsymbol{\beta}, \beta_0), \tag{1.2}$$

where $\ell_n(\boldsymbol{\beta}, \beta_0) = \frac{1}{n}||\mathbf{Y} - \beta_0 \mathbf{1}_{\boldsymbol{n}} - \boldsymbol{\Gamma}_{n,p_0}\boldsymbol{\beta}||_2^2 + \lambda||\boldsymbol{\beta}||_2^2$. The dimension of β is stochastic since it is equal to the number of extracted rules. To get rid of this technical issue in the following of the proof, we rewrite $\ell_n(\boldsymbol{\beta}, \beta_0)$ to have $\boldsymbol{\beta}$ a parameter of fixed dimension $|\Pi|$, the total number of possible rules:

$$\ell_n(\boldsymbol{\beta}, \beta_0) = \frac{1}{n}\sum_{i=1}^n \big(Y_i - \beta_0 - \sum_{\mathscr{P}\in\Pi} \beta_{\mathscr{P}} g_{n,\mathscr{P}}(\mathbf{X}_i)\mathbb{1}_{\mathscr{P}\in\hat{\mathscr{P}}_{M_n,n,p_0}}\big)^2 + \lambda||\boldsymbol{\beta}||_2^2.$$

By the law of large numbers and the previous result (1.1), we have in probability

$$\lim_{n\to\infty} \ell_n(\boldsymbol{\beta}, \beta_0) = \mathbb{E}\big[\big(Y - \beta_0 - \sum_{\mathscr{P}\in\mathscr{P}^\star_{p_0}} \beta_{\mathscr{P}} g^\star_{\mathscr{P}}(\mathbf{X})\big)^2\big] + \lambda||\boldsymbol{\beta}||_2^2 \overset{\text{def}}{=} \ell^\star(\boldsymbol{\beta}, \beta_0),$$

Preprint. Under review.

where $g^\star_{\mathscr{P}}$ is the theoretical rule based on the path $\mathscr{P}$ and the theoretical quantiles. Since Y is bounded, it is easy to see that each component of $\hat{\boldsymbol{\beta}}_{n,p_0}$ is bounded from the following inequalities:

$$\lambda||\hat{\boldsymbol{\beta}}_{n,p_0}||_2^2 \leq \ell_n(\hat{\boldsymbol{\beta}}_{n,p_0}, \hat{\beta}_0) \leq \ell_n(\mathbf{0}, 0) \leq \frac{||Y||_2^2}{n} \leq \max_i Y_i^2.$$

Consequently, the optimization problem (1.2) can be equivalently written with $(\boldsymbol{\beta}, \beta_0)$ constrained to belong to a compact and convex set K. Since ℓ_n is convex and converges pointwise to $\ell^\star$ according to (1.3), the uniform convergence over the compact set K also holds, i.e., in probability

$$\lim_{n\to\infty} \sup_{(\boldsymbol{\beta},\beta_0)\in K} |\ell_n(\boldsymbol{\beta}, \beta_0) - \ell^\star(\boldsymbol{\beta}, \beta_0)| = 0. \tag{1.3}$$

Additionnally, since $\ell^\star$ is a quadratic convex function and the constraint domain K is convex, $\ell^\star$ has a unique minimum that we denote $\boldsymbol{\beta}^\star_{p_0,\lambda}$. Finally, since the maximum of $\ell^\star$ is unique and ℓ_n uniformly converges to $\ell^\star$, we can apply theorem 5.7 from Van der Vaart [35, page 45] to deduce that $(\hat{\boldsymbol{\beta}}_{n,p_0}, \hat{\beta}_0)$ is a consistent estimate of $\boldsymbol{\beta}^\star_{p_0,\lambda}$. We can conclude that, in probability,

$$\lim_{n\to\infty} \mathbb{P}\big(\hat{\mathscr{P}}_{M_n,n,p_0,\lambda} = \{\mathscr{P} \in \mathscr{P}^\star_{p_0} : \beta^\star_{\mathscr{P},p_0,\lambda} > 0\}\big) = 1,$$

and the final stability result follows from the continuous mapping theorem. □

2 Computational Complexity

The computational cost to fit SIRUS is similar to standard random forests, and its competitors: RuleFit, and Node harvest. The full tuning procedure costs about 10 SIRUS fits.

SIRUS SIRUS algorithm has several steps in its construction phase. We derive the computational complexity of each of them. Recall that M is the number of trees, p the number of input variables, and n the sample size.

1. **Forest growing:** $O(Mpnlog(n))$

   The forest growing is the most expensive step of SIRUS. The average computational complexity of a standard forest fit is $O(Mpnlog(n)^2)$ [24]. Since the depth of trees is fixed in SIRUS—see Section 3, it reduces to $O(Mpnlog(n))$ (notice that sorting the data prior to the forest fit may enable to reduce the complexity to $O(Mpn)$).

   A standard forest is grown so that its accuracy cannot be significantly improved with additional trees, which typically results in about 500 trees. In SIRUS, the stopping criterion of the number of trees enforces that 95% of the rules are identical over multiple runs with the same dataset (see Section 6). This is critical to have the forest structure converged and stabilize the final rule list. This leads to forests with a large number of trees, typically 10 times the number for standard forests. On the other hand, shallow trees are grown and the computational complexity is proportional to the tree depth, which is about $log(n)$ for fully grown forests.

   Overall, the modified forest used in SIRUS is about the same computational cost as a standard forest, and has a slightly better computational complexity thanks to the fixed tree depth.

2. **Rule extraction:** $O(M)$

   Extracting the rules in a tree requires a number of operations proportional to the number of nodes, i.e. $O(1)$ since tree depth is fixed. With the appropriate data structure (a map), updating the forest count of the number of occurrences of the rules of a tree is also $O(1)$. Overall, the rule extraction is proportional to the number of trees in the forest, i.e., $O(M)$.

3. **Rule post-treatment:** $O(1)$

   The post-treatment algorithm is only based on the rules and not on the sample. Since the number of extracted rules is bounded by a fixed limit of 25, this step has a computational complexity of $O(1)$.

4. **Rule aggregation:** $O(n)$

Efficient algorithms [17] enable to fit a ridge regression and find the optimal penalization λ with a linear complexity in the sample size n. In SIRUS, the predictors are the rules, whose number is upper bounded by 25, and then the complexity of the rule aggregation is independent of p. Therefore the computational complexity of this step is $O(n)$.

Overall, the computational complexity of SIRUS is $O(Mpnlog(n))$, which is slightly better than standard random forests thanks to the use of shallow trees. Because of the large number of trees and the final ridge regression, the computational cost of SIRUS is comparable to standard forests in practice.

RuleFit/Node harvest Comparison In both RuleFit and Node harvest, the first two steps of the procedure are also to grow a tree ensemble with limited tree depth and extract all possible rules. The complexity of this first phase is then similar to SIRUS: $O(Mpnlog(n))$. However, in the last step of the linear rule aggregation, all rules are combined in a sparse linear model, which is of linear complexity with n, but grows at faster rate than linear with the number of rules, i.e., the number of trees M [17].

As the tree ensemble growing is the computational costly step, SIRUS, RuleFit and Node harvest have a very comparable complexity. On one hand, SIRUS requires to grow more trees than its competitors. On the other hand, the final linear rule aggregation is done with few predictors in SIRUS, while it includes thousands of rules in RuleFit and Node harvest, which has a complexity faster than linear with M.

Tuning Procedure The only parameter of SIRUS which requires fine tuning is p_0, which controls model sparsity. The optimal value is estimated by 10-fold cross validation using a standard bi-objective optimization procedure to maximize both stability and predictivity. For a fine grid of p_0 values, the unexplained variance and stability metric are computed for the associated SIRUS model through a cross-validation. Recall that the bounds of the p_0 grid are set to get the model size between 1 and 25 rules. Next, we obtain a Pareto front, as illustrated in Figure 4, where each point corresponds to a p_0 value of the tuning grid. To find the optimal p_0, we compute the euclidean distance between each point and the ideal point of 0 unexplained variance and 90% stability. Notice that this ideal point is chosen for its empirical efficiency: the unexplained variance can be arbitrary close to 0 depending on the data, whereas we never observed a stability (with respect to data perturbation) higher than 90% accross many datasets. Finally, the optimal p_0 is the one minimizing the euclidean distance distance to the ideal point. Thus, the two objectives, stability and predictivity, are equally weighted.

Tuning Complexity The optimal p_0 value is estimated by a 10-fold cross validation. The costly computational step of SIRUS is the forest growing. However, this step has to be done only once per fold. Then, p_0 can vary along a fine grid to extract more or less rules from each forest, and thus, get the accuracy associated to each p_0 at a total cost of about 10 SIRUS fits.

3 Random Forest Modifications

As explained in Section 1 of the article, SIRUS uses random forests at its core. In order to stabilize the forest structure, we slightly modify the original algorithm from Breiman [4]: cut values at each tree node are limited to the 10-empirical quantiles. In the first paragraph, we show how this restriction have a small impact on predictive accuracy, but is critical to stabilize the rule extraction. On the other hand, the rule selection mechanism naturally only keeps rules with one or two splits. Therefore, tree depth is fixed to 2 to optimize the computational efficiency. In the second paragraph, this phenomenon is thoroughly explained.

Quantile discretization In a typical setting where the number of predictors is $p = 100$, limiting cut values to the 10-quantiles splits the input space in a fine grid of 10^{100} hyperrectangles. Therefore, restricting cuts to quantiles still leaves a high flexibility to the forest and enables to identify local patterns (it is still true in small dimension). To illustrate this, we run the following experiment: for each of the 8 datasets, we compute the unexplained variance of respectively the standard forest and the forest where cuts are limited to the 10-quantiles. Results are presented in Table 4, and we see that there is almost no decrease of accuracy except for one dataset. Besides, notice that setting $q = n$ is equivalent as using original forests.

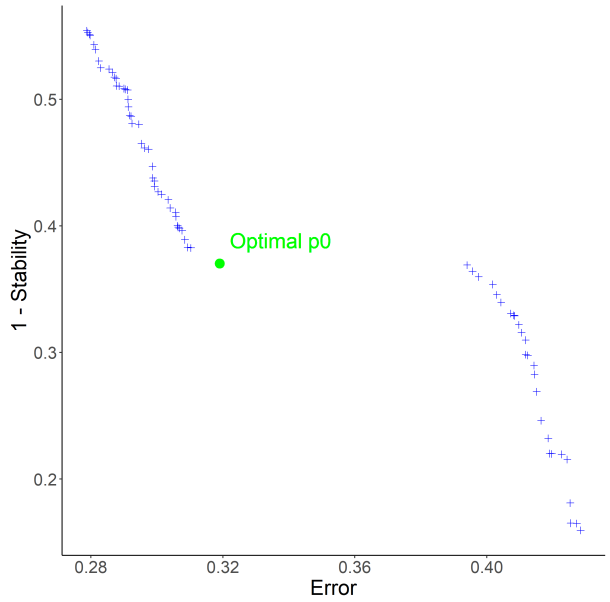

Figure 4: Pareto front of stability versus error (unexplained variance) when p_0 varies, with the optimal value in green for the "Ozone" dataset. The optimal point is the closest one to the ideal point $(0, 0.1)$ of 0 unexplained variance and 90% stability.

Dataset	**Breiman Random Forest**	**Random Forest 10-Quantile Cuts**
Ozone	0.25 (0.007)	0.25 (0.006)
Mpg	0.13 (0.003)	0.13 (0.003)
Prostate	0.46 (0.01)	0.47 (0.02)
Housing	0.13 (0.006)	0.16 (0.004)
Diabetes	0.55 (0.006)	0.55 (0.007)
Machine	0.13 (0.03)	0.24 (0.02)
Abalone	0.44 (0.002)	0.49 (0.003)
Bones	0.67 (0.01)	0.68 (0.01)

Table 4: Proportion of unexplained variance (estimated over a 10-fold cross-validation) for various public datasets to compare two algorithms: Breiman's random forest and the forest where split values are limited to the 10-empirical quantiles. Standard deviations are computed over multiple repetitions of the cross-validation and displayed in brackets.

On the other hand, such discretization is critical for the stability of the rule selection. Recall that the importance of each rule $\hat{p}_{M,n}(\mathscr{P})$ is defined as the proportion of trees which contain its associated path $\mathscr{P}$, and that the rule selection is based on $\hat{p}_{M,n}(\mathscr{P}) > p_0$. In the forest growing, data is bootstrapped prior to the construction of each tree. Without the quantile discretization, this data perturbation results in small variation between the cut values across different nodes, and then the dilution of $\hat{p}_{M,n}(\mathscr{P})$ between highly similar rules. Thus, the rule selection procedure becomes inefficient. More formally, $\hat{p}_{M,n}(\mathscr{P})$ is defined by

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

where $T(\Theta_\ell, \mathscr{D}_n)$ is the list of paths extracted from the ℓ-th tree of the forest. The expected value of the importance of a given rule is

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

Dataset	Random Forest	CART	RuleFit	Node Harvest	SIRUS	SIRUS 50 Rules	SIRUS 50 Rules & d=3
Ozone	0.25	0.36	**0.27**	0.31	0.32	**0.26**	**0.28**
Mpg	0.13	0.20	**0.15**	0.20	0.21	**0.15**	**0.14**
Prostate	0.46	0.60	**0.53**	**0.52**	**0.48**	0.55	0.59
Housing	0.13	0.28	**0.16**	0.24	0.31	0.21	0.20
Diabetes	0.55	0.67	**0.55**	**0.58**	**0.56**	**0.54**	**0.55**
Machine	0.13	0.39	**0.26**	**0.29**	**0.29**	**0.27**	**0.26**
Abalone	0.44	0.56	**0.46**	0.61	0.66	0.64	0.63
Bones	0.67	0.67	**0.70**	**0.70**	**0.74**	**0.72**	**0.71**

Table 3: Proportion of unexplained variance estimated over a 10-fold cross-validation for various public datasets. For rule algorithms only, i.e., RuleFit, Node harvest, and SIRUS, maximum values are displayed in bold, as well as values within 10% of the maximum for each dataset.

Without the discretization, $T(\Theta, \mathscr{D}_n)$ is a random set that takes value in an uncountable space, and consequently

$$\mathbb{E}[\hat{p}_{M,n}(\mathscr{P})] = \mathbb{P}(\mathscr{P} \in T(\Theta_\ell, \mathscr{D}_n)) = 0,$$

and all rules are equally not important in average. In practice, since $\mathscr{D}_n$ is of finite size and the random forest cuts at mid distance between two points, it is still possible to compute $\hat{p}_{M,n}(\mathscr{P})$ and select rules for a given dataset. However, such procedure is highly unstable with respect to data perturbation since we have $\mathbb{E}[\hat{p}_{M,n}(\mathscr{P})] = 0$ for all possible paths.

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 very 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 $2qp = 2 \times 100 \times 10 = 2 \times 10^3$ 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 (a 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 untouched as previously explained. We augment the experiments of Section 3 of the article with an additional column in Table 3: "**SIRUS** 50 **Rules & d**= 3". Recall that, in the column "**SIRUS** 50 **Rules**", p_0 is set manually to extract 100 rules from the forest leading to final lists of about 50 rules (similar size as RuleFit and Node harvest models), an improved accuracy (reaching RuleFit performance), while stability drops to around 50% ($70 - 80\%$ when p_0 is tuned). In the last column, tree depth is set to 3 with the same augmented model size. We observe no accuracy improvement over a tree depth of 2.

This analysis of tree depth is not new. Indeed, both RuleFit [19] and Node harvest [27] 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.

4 Rule Format

The format of the rules with an else clause for the uncovered data points differs from the standard format in the rule learning literature. Indeed, in classical algorithms, a prediction is generated for a

given query point by aggregating the outputs of the rules satisfied by the point. A default rule usually provides predictions for all query points which satisfy no rule. First, observe that the intercept in the final linear aggregation of rules in SIRUS can play the role of a default rule. Secondly, removing the else clause of the rules selected by SIRUS results in an equivalent formulation of the linear regression problem up to the intercept. More importantly, the format with an else clause is required for the stability and modularity [30] properties of SIRUS.

Equivalent Formulation Rules are originally defined in SIRUS as

$$\hat{g}_{n,\mathscr{P}}(\mathbf{x}) = \begin{cases} \bar{Y}_{\mathscr{P}}^{(1)} & \text{if } \mathbf{x} \in \mathscr{P} \\ \bar{Y}_{\mathscr{P}}^{(0)} & \text{otherwise,} \end{cases}$$

where if $\mathbf{x} \in \mathscr{P}$ indicates whether the query point $\mathbf{x}$ satisfies the rule associated with path $\mathscr{P}$ or not, $\bar{Y}_{\mathscr{P}}^{(1)}$ is the output average of the training points which satisfy the rule, and symmetrically $\bar{Y}_{\mathscr{P}}^{(0)}$ is the output average of the training point not covered by the rule. The original linear aggregation of the rules is

$$\hat{m}_{M,n,p_0}(\mathbf{x}) = \hat{\beta}_0 + \sum_{\mathscr{P} \in \hat{\mathscr{P}}_{M,n,p_0}} \hat{\beta}_{n,\mathscr{P}} \hat{g}_{n,\mathscr{P}}(\mathbf{x}).$$

Now we define the rules without the else clause by $\hat{h}_{n,\mathscr{P}}(\mathbf{x}) = (\bar{Y}_{\mathscr{P}}^{(1)} - \bar{Y}_{\mathscr{P}}^{(0)}) \mathbb{1}_{\mathbf{x} \in \mathscr{P}}$, and we can rewrite SIRUS estimate as

$$\begin{aligned} \hat{m}_{M,n,p_0}(\mathbf{x}) =& \big(\hat{\beta}_0 + \sum_{\mathscr{P} \in \hat{\mathscr{P}}_{M,n,p_0}} \hat{\beta}_{n,\mathscr{P}} \bar{Y}_{\mathscr{P}}^{(0)}\big) + \sum_{\mathscr{P} \in \hat{\mathscr{P}}_{M,n,p_0}} \hat{\beta}_{n,\mathscr{P}} \hat{h}_{n,\mathscr{P}}(\mathbf{x}) \\ =& \tilde{\beta}_0 + \sum_{\mathscr{P} \in \hat{\mathscr{P}}_{M,n,p_0}} \hat{\beta}_{n,\mathscr{P}} \hat{h}_{n,\mathscr{P}}(\mathbf{x}). \end{aligned}$$

Therefore the two models with or without the else clause are equivalent up to the intercept.

Stability The problem of defining rules without the else clause lies in the rule selection. Indeed, rules associated with left (L) and right (R) nodes at the first level of a tree are identical:

$$\hat{g}_{n,L}(\mathbf{x}) = \hat{g}_{n,R}(\mathbf{x}) = \bar{Y}_L \mathbb{1}_{\mathbf{x} \in L} + \bar{Y}_R \mathbb{1}_{\mathbf{x} \in R}.$$

Without the else clause, these two rules become different estimates:

$$\begin{aligned} \hat{h}_{n,L}(\mathbf{x}) &= (\bar{Y}_L - \bar{Y}_R) \mathbb{1}_{\mathbf{x} \in L}, \\ \hat{h}_{n,R}(\mathbf{x}) &= (\bar{Y}_R - \bar{Y}_L) \mathbb{1}_{\mathbf{x} \in R}. \end{aligned}$$

However, $\hat{h}_{n,L}$ and $\hat{h}_{n,R}$ are linearly dependent, since $\hat{h}_{n,L}(\mathbf{x}) - \hat{h}_{n,R}(\mathbf{x}) = \bar{Y}_L - \bar{Y}_R$, which does not depend on the query point $\mathbf{x}$. This linear dependence between predictors makes the linear aggregation of the rules ill-defined. One of two rule could be removed randomly, but this would strongly hurt stability.

Modularity Murdoch et al. [30] specify different properties to assess model simplicity: sparsity, simulatability, and modularity. A model is sparse when it uses only a small fraction of the input variables, e.g. the lasso. A model is simulatable if it is possible for humans to perform predictions by hands, e.g. shallow decision trees. A model is modular when it is possible to analyze a meaningful portion of it alone. Typically, rule models are modular since one can analyze the rules one by one. In that case, the average of the output values for instances not covered by the rule is an interesting insight.

5 Dataset Descriptions

Dataset	Sample Size	Total Number of Variables	Number of Categorical Variables
Ozone	203	12	0
Mpg	392	7	0
Prostate	97	8	0
Housing	506	13	0
Diabetes	442	10	0
Machine	209	7	1
Abalone	4177	8	1
Bones	485	3	2

Table 6: Description of datasets

6 Number of Trees

The stability, predictivity, and computation time of SIRUS increase with the number of trees. Thus a stopping criterion is designed to grow the minimum number of trees that ensures stability and predictivity to be close to their maximum. It happens in practice that stabilizing the rule list is computationally more demanding in the number of trees than reaching a high predictivity. Therefore the stopping criterion is only based on stability, and defined as the minimum number of trees such that when SIRUS is fit twice on the same given dataset, 95% of the rules are shared by the two models in average.

To this aim, we introduce $1-\varepsilon_{M,n,p_0}$, an estimate of the mean stability $\mathbb{E}[\hat{S}_{M_n,n,p_0}|\mathscr{D}_n]$ when SIRUS is fit twice on the same dataset $\mathscr{D}_n$. ε_{M,n,p_0} is defined by

$$\varepsilon_{M,n,p_0} = \frac{\sum_{\mathscr{P}\in\Pi} z_{M,n,p_0}(\mathscr{P})(1 - z_{M,n,p_0}(\mathscr{P}))}{\sum_{\mathscr{P}\in\Pi}(1 - z_{M,n,p_0}(\mathscr{P}))},$$

where $z_{M,n,p_0}(\mathscr{P}) = \Phi(Mp_0, M, p_n(\mathscr{P}))$, the cdf of a binomial distribution with parameter $p_n(\mathscr{P}) = \mathbb{E}[\hat{p}_{M_n,n}(\mathscr{P})|\mathscr{D}_n]$, M trials, evaluated at Mp_0. It happens that ε_{M,n,p_0} is quite insensitive to p_0. Consequently it is simply averaged over a grid $\hat{V}_{M,n}$ of many possible values of p_0. Therefore, the number of trees is set, for $\alpha = 0.05$, by

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

to ensure that 95% of the rules are shared by the two models in average. See Section 4 from Bénard et al. [2] for a thorough explanation of this stopping criterion.