Interpretable Random Forests via Rule Extraction

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

[1]Safran Tech, Modeling & Simulation, 78114 Magny-Les-Hameaux, France
[2]Sorbonne Université, CNRS, LPSM, 75005 Paris, France
[3]Ecole Polytechnique, IP Paris, CMAP, 91128 Palaiseau, France

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). A `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 (Lipton, 2016; Murdoch et al., 2019), it is yet possible to define simplicity, stability, and predictivity as minimum requirements for interpretable models (Bénard et al., 2021; Yu and Kumbier, 2019). Simplicity of the model structure can be assessed by the number of operations performed in the prediction mechanism. In particular, Murdoch et al. (2019) introduce the notion of *simulatable models* when a human is able to reproduce the prediction process by hand. Secondly, Yu (2013) 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* (Breiman, 2001b). Finally, as also explained in Breiman (2001b), if the decrease of predictive accuracy is significant 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 (Breiman et al., 1984) can model nonlinear patterns while having a simple structure. They

Proceedings of the 24[th] International Conference on Artificial Intelligence and Statistics (AISTATS) 2021, San Diego, California, USA. PMLR: Volume 130.

are therefore often presented as interpretable. However, the structure of trees is highly sensitive to small data perturbation (Breiman, 2001b), 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:

$$\textbf{If} \quad \begin{cases} X^{(1)} < 1.12 \\ \textbf{\&}\ X^{(3)} \geq 0.7 \end{cases} \quad \textbf{then } \hat{Y} = 0.18 \textbf{ else } \hat{Y} = 4.1 \ .$$

A large number of rule algorithms have been developed, among which the most influential are 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), LRI (Leightweight Rule Induction, Weiss and Indurkhya, 2000), RuleFit (Friedman and Popescu, 2008), Node harvest (Meinshausen, 2010), ENDER (Ensemble of Decision Rules, Dembczyński et al., 2010), BRL (Bayesian Rule Lists, Letham et al., 2015), RIPE (Rule Induction Partitioning Estimator, Margot et al., 2018, 2019), and Wei et al. (2019, 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. (2021), who have introduced SIRUS for classification problems. Our algorithm is based on random forests (Breiman, 2001a), 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 (Friedman and Popescu, 2008) and Node harvest (Meinshausen, 2010). Both methods also extract large collection of rules from tree ensembles: RuleFit uses a boosted tree ensemble (ISLE, Friedman and Popescu, 2003) whereas Node harvest is based on random forests. The rule selection is performed by a sparse linear aggregation, respectively the Lasso (Tibshirani, 1996) 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 $X^{(1)}, \dots, X^{(p)}$, 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 2 and 3). 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 $\hat{\mathscr{P}}_{M,n,p_0}$. We refer to Section 4 of the Supplementary Material for a detailed illustration of the post-treatment procedure on real data.

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

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

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

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

where $\mathbf{1}_{\boldsymbol{n}} = (1, \ldots, 1)^T$ 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 $\boldsymbol{\beta} = \{\beta_1, \ldots, \beta_{c_n}\} \in \mathbb{R}_+^{c_n}$ 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 (Murdoch et al., 2019)) to SIRUS—see Section 5 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 (Meinshausen, 2010). 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 (Chao et al., 2006; Zucknick et al., 2008; Boulesteix and Slawski, 2009; He and Yu, 2010; Alelyani et al., 2011). 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 6 of the Supplementary Material provides dataset details.

SIRUS rule set. Our algorithm is illustrated on the "LA Ozone" dataset from Friedman et al. (2001), 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 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 as illustrated in Figure 1, and described 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. For a

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 (about 9000 trees are grown to reach convergence).

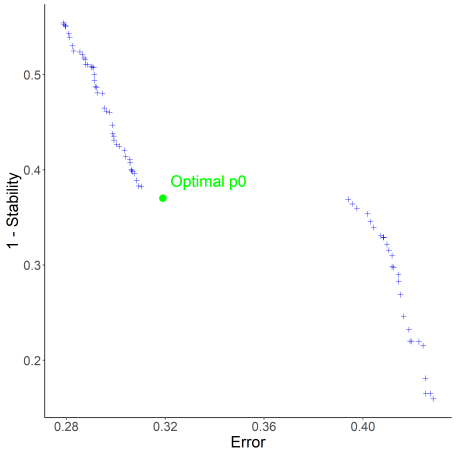

Figure 1: Pareto front of stability versus error when p_0 varies for the "Ozone" dataset (optimal value in green).

robust estimation of p_0, the cross-validation is repeated 10 times and the median p_0 value is selected. 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 7 of the Supplementary Material for a detailed definition of this criterion. Finally, we use the standard settings of random forests (well-known for their excellent performance, in particular *mtry* is $\lfloor p/3 \rfloor$ and at least 2), and set $q = 10$ quantiles, while categorical variables are handled as natively defined in trees.

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 (Wright and Ziegler, 2017). We also use available `R` implementations `pre` (Fokkema, 2017, RuleFit) and `nodeharvest` (Meinshausen, 2015). 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 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. Finally, the column "SIRUS sparse" of Tables 2 and 3 shows the excellent behavior of SIRUS in a sparse setting: for each dataset, 3 randomly permuted copies of each variable are added to the data, leaving SIRUS performance almost untouched.

To illustrate the typical behavior of our method, we comment the results for two specific datasets: "Diabetes" (Efron et al., 2004) and "Machine" (Dua and Graff, 2017). 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

(a) Model Size

Dataset	CART	RuleFit	Node harvest	SIRUS	SIRUS sparse
Ozone	15	21	46	**11**	10
Mpg	15	40	43	**10**	10
Prostate	11	14	41	**9**	12
Housing	15	54	40	**6**	6
Diabetes	**12**	25	42	**12**	15
Machine	**8**	44	42	9	7
Abalone	20	58	35	**8**	13
Bones	17	5	13	**1**	1

(b) Stability

Dataset	RuleFit	Node harvest	SIRUS	SIRUS sparse
Ozone	0.22	0.30	**0.62**	0.63
Mpg	0.25	0.43	**0.77**	0.76
Prostate	0.32	0.23	**0.58**	0.59
Housing	0.19	0.40	**0.82**	0.82
Diabetes	0.18	0.39	**0.69**	0.65
Machine	0.23	0.29	**0.86**	0.84
Abalone	0.31	0.38	**0.75**	0.74
Bones	0.59	0.52	**0.96**	0.78

Table 2: Mean model size and stability over a 10-fold cross-validation for various public datasets. Minimum size and maximum stability are in bold ("SIRUS sparse" put aside).

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, 6 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 86%, 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 2 and 3, respectively.

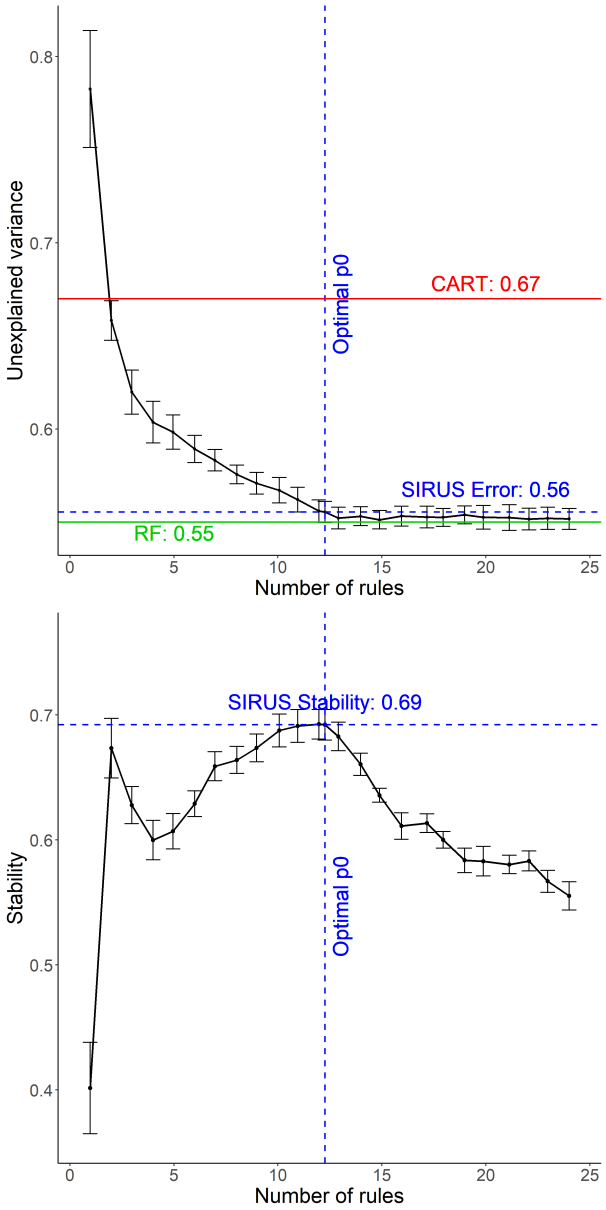

Figure 2: For the dataset "Diabetes", unexplained variance (top panel) and stability (bottom panel) versus the number of rules when p_0 varies, estimated via 10-fold cross-validation (results are averaged over 10 repetitions).

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 (Letham et al., 2015; Murdoch et al., 2019). For

Dataset	Random Forest	CART	RuleFit	Node harvest	SIRUS	SIRUS sparse	SIRUS 50 rules
Ozone	0.25	0.36	**0.27**	0.31	0.32	0.32	**0.26**
Mpg	0.13	0.20	**0.15**	0.20	0.20	0.20	**0.15**
Prostate	0.48	0.60	**0.53**	**0.52**	**0.55**	0.51	**0.54**
Housing	0.13	0.28	**0.16**	0.24	0.30	0.31	0.20
Diabetes	0.55	0.67	**0.55**	**0.58**	**0.56**	0.56	**0.55**
Machine	0.13	0.39	**0.26**	**0.29**	**0.29**	0.32	**0.27**
Abalone	0.44	0.56	**0.46**	0.61	0.66	0.64	0.64
Bones	0.67	0.67	**0.70**	**0.70**	**0.73**	0.77	**0.73**

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, minimum values are displayed in bold, as well as values within 10% of the minimum for each dataset ("SIRUS sparse" put aside).

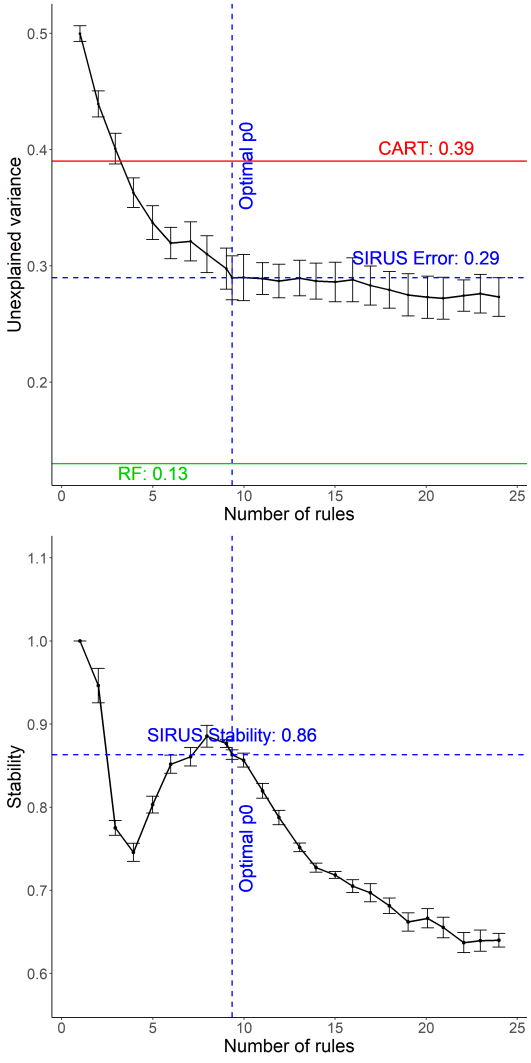

Figure 3: For the dataset "Machine", unexplained variance (top panel) and stability (bottom panel) versus the number of rules when p_0 varies, estimated via 10-fold cross-validation (results are averaged over 10 repetitions).

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

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

where d is the tree depth, and for $k \in \{1, \ldots, d\}$, the triplet (j_k, r_k, s_k) describes how to move from level $(k-1)$ to level k, with a split using the coordinate $j_k \in \{1, \ldots, p\}$, the index $r_k \in \{1, \ldots, q-1\}$ of the corresponding quantile, and a side $s_k = L$ if we go to the left and $s_k = R$ if we go to the right—see Figure 4. 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 $\Theta_1, \ldots, \Theta_\ell, \ldots, \Theta_M$ 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,

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

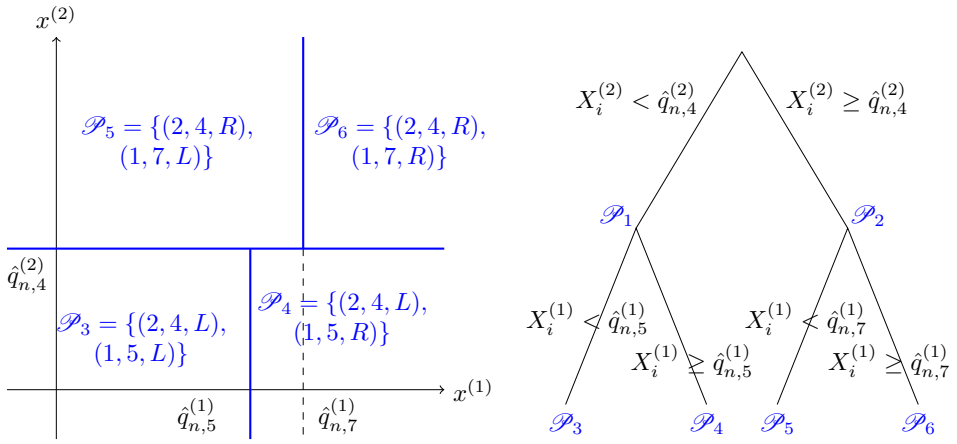

Figure 4: Example of a root node $\mathbb{R}^2$ partitioned by a randomized tree of depth 2: the tree on the right, the associated paths and hyperrectangles of length $d = 2$ on the left.

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, (Scornet et al., 2015; Mentch and Hooker, 2016), 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 fits 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 lists of rules. This remarkably stable behavior is theoretically understood since the rule selection is consistent. A `R/C++` software `sirus` is available from `CRAN`.

Acknowledgements

We thank the reviewers for their insightful comments and suggestions.

References

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

Bénard, C., Biau, G., Da Veiga, S., and Scornet, E. (2021). Sirus: Stable and interpretable rule set for classification. *Electronic Journal of Statistics*, 15:427–505.

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

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

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

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

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

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

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

Cohen, W. and Singer, Y. (1999). 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. AAAI Press.

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

Dua, D. and Graff, C. (2017). UCI machine learning repository.

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

Fokkema, M. (2017). PRE: An R package for fitting prediction rule ensembles. *arXiv:1707.07149*.

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

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

Friedman, J. and Popescu, B. (2003). Importance sampled learning ensembles. *Journal of Machine Learning Research*, 94305:1–32.

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

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

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

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

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

Margot, V., Baudry, J.-P., Guilloux, F., and Wintenberger, O. (2018). Rule induction partitioning estimator. In *Proceedings of the 14th International Conference on Machine Learning and Data Mining in Pattern Recognition*, pages 288–301, New York. Springer.

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

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

Meinshausen, N. (2015). Package 'nodeharvest'.

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

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

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

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

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

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

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

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

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

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

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

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