Supplementary Material For: Interpretable Random Forests via Rule Extraction

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

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 $\Theta'_1, \dots, \Theta'_M$.

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. (2021) 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:

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

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

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

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

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:

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

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 (2000, 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 $O(Mpnlog(n)^2)$ (Louppe, 2014). Since the depth of trees is fixed in SIRUS—see Section 3, it reduces to $O(Mpnlog(n))$.

   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 7). 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 (Friedman et al., 2010) 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 (Friedman et al., 2010).

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 1, 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 do not observe 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. For a robust estimation of p_0, the cross-validation is repeated 10 times and the median p_0 value is selected.

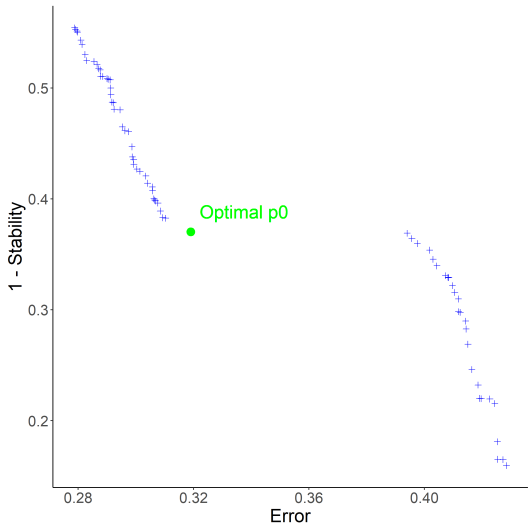

Figure 1: Pareto front of stability versus error (unexplained variance) when p_0 varies, with the optimal value in green for the "Ozone" dataset. The optimal point is the closest one to the ideal point $(0, 0.1)$ of 0 unexplained variance and 90% stability.

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 1: 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.

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 (Breiman, 2001): 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 1, 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.

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 sparse	SIRUS 50 rules	SIRUS 50 rules & d=3
Ozone	0.25	0.36	**0.27**	0.31	0.32	0.32	**0.26**	**0.27**
Mpg	0.13	0.20	**0.15**	0.20	0.20	0.20	**0.15**	**0.15**
Prostate	0.48	0.60	**0.53**	**0.52**	**0.55**	0.51	**0.54**	**0.55**
Housing	0.13	0.28	**0.16**	0.24	0.30	0.31	0.20	0.21
Diabetes	0.55	0.67	**0.55**	**0.58**	**0.56**	0.56	**0.55**	**0.55**
Machine	0.13	0.39	**0.26**	**0.29**	**0.29**	0.32	**0.27**	**0.26**
Abalone	0.44	0.56	**0.46**	0.61	0.66	0.64	0.64	0.65
Bones	0.67	0.67	**0.70**	**0.70**	**0.73**	0.77	**0.73**	**0.75**

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).

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 (Friedman and Popescu, 2008) and Node harvest (Meinshausen, 2010) articles discuss the optimal tree depth for the rule extraction from a tree ensemble in their experiments. They both conclude that the optimal depth is 2. Hence, the same hard limit of 2 is used in Node harvest. RuleFit is slightly less restrictive: for each tree, its depth is randomly sampled with an exponential distribution concentrated on 2, but allowing few trees of depth 1, 3 and 4. We insist that they both reach such conclusion without considering stability issues, but only focusing on accuracy.

4 Post-treatment Illustration

We illustrate the post-treatment procedure with the "Machine" dataset. Table 3 provides the initial raw list of 17 rules on the left, and the final post-treated 9-rule list on the right, using $p_0 = 0.072$. The rules removed from the

1	**if**	MMAX < 32000	**then**	$\hat{Y} = 61$	**else**	$\hat{Y} = 408$
2	**if**	MMAX ≥ 32000	**then**	$\hat{Y} = 408$	**else**	$\hat{Y} = 61$
3	**if**	MMIN < 8000	**then**	$\hat{Y} = 62$	**else**	$\hat{Y} = 386$
4	**if**	MMIN ≥ 8000	**then**	$\hat{Y} = 386$	**else**	$\hat{Y} = 62$
5	**if**	CACH < 64	**then**	$\hat{Y} = 56$	**else**	$\hat{Y} = 334$
6	**if**	CACH ≥ 64	**then**	$\hat{Y} = 334$	**else**	$\hat{Y} = 56$
7	**if**	MMAX ≥ 32000 & CACH ≥ 64	**then**	$\hat{Y} = 517$	**else**	$\hat{Y} = 67$
8	**if**	CHMIN < 8	**then**	$\hat{Y} = 50$	**else**	$\hat{Y} = 312$
9	**if**	CHMIN ≥ 8	**then**	$\hat{Y} = 312$	**else**	$\hat{Y} = 50$
10	**if**	MYCT < 50	**then**	$\hat{Y} = 335$	**else**	$\hat{Y} = 58$
11	**if**	MYCT ≥ 50	**then**	$\hat{Y} = 58$	**else**	$\hat{Y} = 335$
12	**if**	MMAX ≥ 32000 & CACH < 64	**then**	$\hat{Y} = 192$	**else**	$\hat{Y} = 102$
13	**if**	MMAX < 32000 & CHMIN ≥ 8	**then**	$\hat{Y} = 157$	**else**	$\hat{Y} = 100$
14	**if**	MMAX < 32000 & CHMIN ≥ 12	**then**	$\hat{Y} = 554$	**else**	$\hat{Y} = 73$
15	**if**	MMAX ≥ 32000 & CHMIN < 12	**then**	$\hat{Y} = 252$	**else**	$\hat{Y} = 96$
16	**if**	MMIN ≥ 8000 & CHMIN ≥ 12	**then**	$\hat{Y} = 586$	**else**	$\hat{Y} = 76$
17	**if**	MMIN ≥ 8000 & CHMIN < 12	**then**	$\hat{Y} = 236$	**else**	$\hat{Y} = 94$

1	**if**	MMAX < 32000	**then**	$\hat{Y} = 61$	**else**	$\hat{Y} = 408$
3	**if**	MMIN < 8000	**then**	$\hat{Y} = 62$	**else**	$\hat{Y} = 386$
5	**if**	CACH < 64	**then**	$\hat{Y} = 56$	**else**	$\hat{Y} = 334$
7	**if**	MMAX ≥ 32000 & CACH ≥ 64	**then**	$\hat{Y} = 517$	**else**	$\hat{Y} = 67$
8	**if**	CHMIN < 8	**then**	$\hat{Y} = 50$	**else**	$\hat{Y} = 312$
10	**if**	MYCT < 50	**then**	$\hat{Y} = 335$	**else**	$\hat{Y} = 58$
13	**if**	MMAX < 32000 & CHMIN ≥ 8	**then**	$\hat{Y} = 157$	**else**	$\hat{Y} = 100$
14	**if**	MMAX < 32000 & CHMIN ≥ 12	**then**	$\hat{Y} = 554$	**else**	$\hat{Y} = 73$
16	**if**	MMIN ≥ 8000 & CHMIN ≥ 12	**then**	$\hat{Y} = 586$	**else**	$\hat{Y} = 76$

Table 3: SIRUS post-treatment of the extracted raw list of rules for the "Machine" dataset: the raw list of rules on the left, and the final post-treated rule list on the right (removed rules are highlighted in red for one constraint rules and in orange for two constraint rules).

raw list are highlighted in red and orange. Red rules have one constraint and are identical to a previous rule with the constraint sign reversed. Notice that two such rules (e.g. rules 1 and 2) correspond to the left and right child nodes at the first level of a tree. Thus, they belong to the same trees of the forest and their associated occurrence frequencies $\hat{p}(\mathscr{P})$ are equal. We always keep the rule with the sign "$<$": this choice is somewhat arbitrary and of minor importance since the two rules are identical. Orange rules have two constraints and are linearly dependent on other previous rules. For example for rule 12, there exist 3 real numbers α_1, α_5, and α_7 such that, for all $\mathbf{x} \in \mathbb{R}^p$

$$g_{\mathscr{P}_{12}}(\mathbf{x}) = \alpha_1 g_{\mathscr{P}_1}(\mathbf{x}) + \alpha_5 g_{\mathscr{P}_5}(\mathbf{x}) + \alpha_7 g_{\mathscr{P}_7}(\mathbf{x}).$$

Observe that rules 12 and 7 involve the same variables and thresholds, but one of the sign constraints is reversed. The estimated rule outputs $\hat{Y}$ are of course different between rules 12 and 7 because they identify two different quarters of the input space. The outputs of rule 7 have a wider gap than the ones of rule 12, and consequently the CART-splitting criterion of rule 12 is smaller, which also implies a smaller occurrence frequency, i.e., $\hat{p}(\mathscr{P}_{12}) < \hat{p}(\mathscr{P}_7)$. Therefore rule 12 is removed rather than rule 7. The same reasoning applies to rules 15 and 17.

5 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 (Murdoch et al., 2019) 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

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

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

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

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. (2019) 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.

6 Dataset Descriptions

Dataset	Sample Size	Total Number of Variables	Number of Categorical Variables
Ozone	330	9	0
Mpg	398	7	0
Prostate	97	8	0
Housing	506	13	0
Diabetes	442	10	0
Machine	209	6	0
Abalone	4177	8	1
Bones	485	3	2

Table 4: Description of datasets

7 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

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

to ensure that 95% of the rules are shared by the two models in average. See Section 4 from Bénard et al. (2021) for a thorough explanation of this stopping criterion.

References

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.

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

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

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

Louppe, G. (2014). Understanding random forests: From theory to practice. *arXiv preprint arXiv:1407.7502*.

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

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

Van der Vaart, A. (2000). *Asymptotic Statistics*, volume 3. Cambridge University Press, Cambridge.