Machine Learning manuscript No.
(will be inserted by the editor)

A new method to compare the interpretability of rule-based algorithms.

Vincent Margot · George Luta

Received: date / Accepted: date

Abstract Interpretability is becoming increasingly important for predictive model analysis. Unfortunately, as remarked by many authors, there is still no consensus regarding this notion. The goal of this article is to propose a definition of the notion of interpretability that allows comparisons of rule-based algorithms. This definition consists of three terms, each one being quantitatively measured with a simple formula: predictivity, stability and simplicity. While predictivity has been extensively studied to measure the accuracy of predictive algorithms, stability is based on the Dice-Sorensen index for comparing two sets of rules generated by an algorithm using two independent samples. The simplicity is based on the sum of the length of the rules derived from the predictive model. The new measure for the interpretability of a rule-based algorithm is a weighted sum of the three terms mentioned above. We use the new measure to compare the interpretability of several rule-based algorithms, specifically CART, RuleFit, Node Harvest, Covering algorithm and SIRUS for the regression case, and CART, PART and RIPPER for the classification case.

Keywords Interpretability · Transparency · Explainability · Predictivity · Stability · Simplicity · Rule-based algorithms · Machine Learning.

1 Introduction

The widespread use of machine learning (ML) methods in many sensitive areas such as health care, justice, defense or asset management has underscored the importance of interpretability in the decision-making process. In recent years, the

Vincent Margot
Chief Algorithm Officer
Advestis, 69 boulevard Haussmann, F-75008 Paris
E-mail: vmargot@advestis.com

George Luta
Georgetown University, Washington, DC 20057-1484, USA
E-mail: george.luta@georgetown.edu

number of publications on interpretability has increased exponentially. For a complete overview of interpretability in ML see the book [33]. We distinguish two main approaches to generate interpretable prediction models. The first approach is to use a non-interpretable ML algorithm to generate the predictive model, and then create a so-called post-hoc interpretable model. We may refer to the algorithms LIME [37], DeepLIFT [39] and SHAP [29] that attempt to measure the importance of a feature in the prediction process (we refer to [20] for an overview of the available methods). However, as outlined in [38], the explanations may not be sufficient to allow a reasonable decision process.

The other possibility is to use intrinsically interpretable algorithms to directly generate interpretable models. There are two main families of intrinsically interpretable algorithms: tree-based algorithms that are based on decision trees such as CART [4], ID3 [35], C4.5 [36], M5P [42], LMT [26] and rule-based algorithms that are generating rule sets such as RIPPER [5], FORS [25], M5 Rules [23], RuleFit [14], Ender [7], Node Harvest [32] or more recently SIRUS [2,3], RIPE [30] and Coverage Algorithm [31]. It is important to note that any tree can be converted into a set of rules, while the opposite is not true.

These algorithms generate predictive models based on the notion of a rule. A rule is a If-Then statement of the form

$$\begin{aligned} &\text{IF} \quad c_1 \text{ And } c_2 \text{ And } \dots \text{ And } c_k \\ &\text{THEN} \quad \text{Prediction} = p, \end{aligned}$$

The condition part If is a logical conjunction, where c_i's are tests that check whether the observation has the specified properties or not. The number k is called the length of the rule. If all c_i's are fulfilled the rule is said to be activated. The conclusion part Then is the prediction of the rule if it is activated.

Even though rule-based algorithms and tree-based algorithms seem to be easy to understand, there is no strict mathematical definition for the concept of interpretability. This is due to the fact that interpretability involves many concepts as explained in [28], [8], [43] and [34]. The goal of this paper is to propose a definition that combines these concepts in order to generate a quantitative measure of interpretability. It is important to note that related concepts such as justice, ethics, and morality, which are associated with specific applications to justice, mortgage, or health care, cannot be measured quantitatively.

As proposed in [43] and [2], we describe an interpretability measure for any model formed by rules based on the triptych predictivity, stability, and simplicity: The predictivity score measures the accuracy of the generated prediction model. The accuracy ensures a high degree of confidence in the generated model. The stability score quantifies the sensitivity of an algorithm to noise, and it allows to evaluate the robustness of the algorithm. The simplicity score could be interpreted as the ability to easily verify the prediction. A simple model makes it easy to evaluate some qualitative criteria such as ethics and morality. By measuring these three quantities we are therefore able to evaluate the interpretability of several algorithms for a given problem.

2 Predictivity score

The aim of a predictive model is to predict the value of a random variable of interest $Y \in \mathcal{Y}$, given features $X \in \mathcal{X}$ where $\mathcal{X}$ is a d-dimensional space. Formally, we consider the standard setting as follows: Let (X, Y) be a random vector in $\mathcal{X} \times \mathcal{Y}$ of unknown distribution $\mathbb{Q}$ such that

$$Y = g^*(X) + Z,$$

where $\mathbb{E}[Z] = 0$ and $\mathbb{V}(Z) = \sigma^2$ and g^* is a measurable function from $\mathcal{X}$ to $\mathcal{Y}$.

We denote by $\mathcal{G}$ the set of all measurable functions from $\mathcal{X}$ to $\mathbb{R}$. The accuracy of a predictor $g \in \mathcal{G}$ is measured by its risk, defined as

$$\mathcal{L}(g) := \mathbb{E}_{\mathbb{Q}}\left[\gamma\left(g; (X, Y)\right)\right], \tag{1}$$

where $\gamma : \mathcal{G} \times (\mathcal{X} \times \mathcal{Y}) \to [0, \infty[$ is called a contrast function and its choice depends on the nature of Y. The risk measures the average discrepancy between $g(X)$ and Y, given a new observation (X, Y) from the distribution $\mathbb{Q}$. As mentioned in [1], the definition (1) includes most of the classical statistical cases.

Given a sample $D_n = ((X_1, Y_1), \dots, (X_n, Y_n))$, our aim is to predict Y given X. The observations (X_i, Y_i) are assumed to be independent and identically distributed (i.i.d) from the distribution $\mathbb{Q}$.

We consider a statistical algorithm which is a measurable mapping from $(\mathcal{X} \times \mathcal{Y})^n$ to a class of measurable functions $\mathcal{G}_n \subseteq \mathcal{G}$. This algorithm generates a predictor g_n by using the the Empirical Risk Minimization principle (ERM) [41], meaning that

$$g_n = \underset{g \in \mathcal{G}_n}{\arg\min}\, \mathcal{L}_n(g),$$

where $\mathcal{L}_n(g) = \frac{1}{n}\sum_{i=1}^{n} \gamma(g, (X_i, Y_i))$ is the empirical risk.

The notion of predictivity is based on the ability of an algorithm to provide an accurate predictor. This notion has been well studied before. In this paper we define the predictivity score as:

$$\mathcal{P}_n(g_n, h_n) := 1 - \frac{\mathcal{L}_n(g_n)}{\mathcal{L}_n(h_n)}, \tag{2}$$

where h_n is a baseline predictor chosen by the analyst. The idea is to consider a naïve and easily built predictor chosen according to the contrast function.

For instance, if $Y \in \mathbb{R}$, we generally use the quadratic contrast with $\gamma\left(g; (X, Y)\right) = (g(X) - Y)^2$. In this case, the minimizer of the risk (1) is the regression function defined by

$$g^*(X) = \mathbb{E}_{\mathbb{Q}}\left[Y \mid X\right], \text{ hence we set } h_n = \frac{1}{n}\sum_{i=1}^{n} Y_i.$$

If $Y \in \{0, 1\}$, we use the $0-1$ contrast function $\gamma\left(g; (X, Y)\right) := \mathbf{1}_{g(X) \neq Y}$, and the minimizer of the risk is the Bayes classifier defined by

$$g^*(X) = \mathbf{1}_{\mathbb{Q}(Y=1|X) \geq 1/2}, \text{ hence we set } h_n = \mathbf{1}_{\sum_{i=1}^{n} Y_i \geq n/2}.$$

The predictivity score (2) is a measure of accuracy which is independent of the range of Y. We can assume that it is a positive number between 0 and 1. In fact, the risk (1) is a positive function, and if $\mathcal{P}_n(g_n, \gamma) > 1$, it means that the predictor g_n is worse than the chosen baseline predictor h_n.

3 q-Stability score

In [2,3], authors have proposed a measure of the stability for rule-based algorithms based on the following definition:

"A rule learning algorithm is stable if two independent estimations based on two independent samples result in two similar lists of rules."

The q-stability score is based on the same definition. This term seems to be biased for algorithms that do not use feature discretization and work with real values. Indeed, if the feature is continuous, the probability that a decision tree algorithm will cut on the same exact value for the same rule on two independent samples is zero. For this reason, this definition of the stability is too stringent in this case.

To avoid this problem, we discretize all continuous features. The discretization of features is a common solution to control the complexity of a rule generator. In [11], for example, the authors use entropy minimization heuristics to discretize features and for the algorithms BRL (Bayesian rule lists) [27], SIRUS [2,3] and RIPE [30] the authors have discretized the features by using their empirical quantiles. We refer to [9] for an overview of the common discretization methods.

In this paper we consider the empirical quantile discretization to generate the q-stability score. Let $q \in \mathbb{N}$ be the number of quantiles considered for the discretization, and let X be a continuous feature. An integer $p \in \{1, \dots, q\}$, called bin, is assigned to each interval $[x_{(p-1)/q}, x_{p/q}]$, where $x_{p/q}$ is the p-th q-quantile of X. A discrete version of the X feature, designated by $Q_q(X)$, is constructed by replacing each value with its corresponding bin. In other words, a value p_a is assigned to all $a \in X$ such that $a \in [x_{(p_a-1)/q}, x_{p_a/q}]$.

This discretization process can be extended to a rule set by replacing the interval boundaries of the individual tests c_i with the corresponding bins. For example, the test $X \in [a, b]$ becomes $Q_q(X) \in [p_a, p_b]$, where p_a and p_b are such that $a \in [x_{(p_a-1)/q}, x_{p_a/q}]$ and $b \in [x_{(p_b-1)/q}, x_{p_b/q}]$.

The formula for the q-stability score is based on the so-called Dice-Sorensen index. Let $\mathcal{A}$ be an algorithm and let D_n and D'_n be two independent samples of n i.i.d. observations drawn from the same distribution $\mathbb{Q}$. We denote by R_n and R'_n the sets of rules generated by an algorithm $\mathcal{A}$ based on D_n and D'_n, respectively. Then, the q-stability score is calculated as

$$\mathcal{S}_n^q(\mathcal{A}) := \frac{2\,|Q_q(R_n) \cap Q_q(R'_n)|}{|Q_q(R_n)| + |Q_q(R'_n)|}, \tag{3}$$

where $Q_q(R)$ is the discretized version of the set of rules R, with the convention that $0/0 = 0$, and the discretization process is performed by using D_n and D'_n, respectively.

The q-stability score (3) is the ratio of the common rules between $Q_q(R_n)$ and $Q_q(R'_n)$. It is a positive number between 0 and 1: If $Q_q(R_n)$ and $Q_q(R'_n)$ have no common rules, then $\mathcal{S}_n^q(\mathcal{A}) = 0$, while if $Q_q(R_n)$ and $Q_q(R'_n)$ have the same rules, then $\mathcal{S}_n^q(\mathcal{A}) = 1$.

4 Simplicity score

In [31] the authors have introduced the concept of an interpretability index, which is based on the sum of the length of all the rules of the prediction model. Such an interpretability index should not be confused with the broader concept of interpretability that is developed in this paper. As discussed in section 5, the former will be interpreted as one of the components of the latter.

Definition 1 The interpretability index of an estimator g_n generated by a set of rules R_n is defined by

$$Int(g_n) := \sum_{r \in R_n} \text{length}(r). \tag{4}$$

Even if (4) seems naive, we consider it a good measure for the simplicity of a tree-based algorithm or a rule-based algorithm. Indeed, as the number of rules or the length of the rules increases, $Int(g_n)$ also increases. The fewer the number of rules and their length, the easier their understanding is.

It is important to note that the value (4), which is a positive number, cannot be directly compared to the scores from (2) and (3), which are between 0 and 1.

The simplicity score is based on the definition 1. The idea is to compare (4) relatively to a set of algorithms $\mathcal{A}_1^m = \{\mathcal{A}_1, \dots, \mathcal{A}_m\}$. Hence the simplicity of an algorithm $\mathcal{A}_i \in \mathcal{A}_1^m$ is defined in relative terms as follows:

$$\mathbb{S}_n(\mathcal{A}_i, \mathcal{A}_1^m) = \frac{min\{Int(g_n^A : A \in \mathcal{A}_1^m)\}}{Int(g_n^{\mathcal{A}_i})}. \tag{5}$$

Similar to the previous scores, this quantity is also a positive number between 0 and 1: If $\mathcal{A}_i$ generates the simplest predictor among the set of algorithms $\mathcal{A}_1^m$ then $\mathbb{S}_n(\mathcal{A}_i, \mathcal{A}_1^m) = 1$, and the simplicity of other algorithms in $\mathcal{A}_1^m$ are evaluated relatively to $\mathcal{A}_i$.

5 Interpretability score

In [8] the authors define interpretability as "the ability to explain or present to a person in an understandable form". We claim that an algorithm with a high predictivity score (2), stability score (3) and simplicity score (5) is interpretable in the sense of [8]. Indeed, a high predictivity score ensures confidence, a high stability score ensures robustness, and a high simplicity score ensures that the generated model is easy to understand for humans, since there are only a few rules with small lengths.

The main idea behind the proposed definition of interpretability is to use a weighted sum of these three scores. Let $\mathcal{A}_1^m$ be a set of algorithms. Then, the interpretability of any algorithm $\mathcal{A}_i \in \mathcal{A}_1^m$ is defined as:

$$\mathcal{I}(\mathcal{A}_i, D_n, D_n', \gamma, q) = \alpha_1 \mathcal{P}(g_n^{\mathcal{A}_i}, \gamma) + \alpha_2 \mathcal{S}_n^q(\mathcal{A}_i) + \alpha_3 \mathbb{S}_n(\mathcal{A}_i, \mathcal{A}_1^m), \tag{6}$$

where the coefficients α_1, α_2 and α_3 have been chosen according to the analyst's objective, such that $\alpha_1 + \alpha_2 + \alpha_3 = 1$.

It is important to note that the definition of interpretability (6) depends on the set of algorithms under consideration and the specific setting. Therefore, the interpretable score only makes sense within this set of algorithms and for a given setting.

6 Application

The goal of this application is to compare several algorithms which are considered interpretable: Regression Tree [4], RuleFit (RF) [14], NodeHarvest (NH) [32], Covering Algorithm (CA) [31] and SIRUS [3] for regression settings, and RIPPER [5], PART [13] and Classification Tree [4] for classification settings[1].

6.1 Brief overview of the selected algorithms

RIPPER (Repeated Incremental Pruning to Produce Error Reduction) is a sequential coverage algorithm. It is based on the "separate-and-conquer" approach. This means that for a selected class it searches for the best rule according to a criterion and removes the points covered by that rule. Then it searches for the best rule for the remaining points and so on until all points of this class are covered. Then it moves on to the next class. The classes are examined by increasing size.

PART is also a "separate-and-conquer" rule learner. The main difference is that to create the "best rule", the algorithm uses a pruned decision tree and keeps the leaf with the largest coverage.

RuleFit is a very accurate rule-based algorithm. First it generates a list of rules by considering all nodes and leaves of a boosted tree ensemble ISLE [15]. Then the rules are used as additional binary features in a sparse linear regression model that is using the Lasso [40]. A feature generated by a rule is equal to 1 if the rule is activated, and it is 0 otherwise.

Node harvest also uses a tree ensemble as a rule generator. The algorithm considers all nodes and leaves of a Random Forest as rules and solves a linear quadratic problem to fit a weight for each node. Hence, the estimator is a convex combination of the nodes.

Covering Algorithm has been designed to generate a very simple model. The algorithm extracts a sparse set of rules considering all nodes and leaves of a tree ensembles (using the Random Forest algorithm, Gradient Boosting algorithm [16] or Stochastic Gradient Boosting algorithm [17]). Rules are selected according to their statistical properties to form a "quasi-covering". The covering is then turned into a partition using the so-called partitioning trick [30] to form a consistent estimator of the regression function.

SIRUS (Stable and Interpretable RUle Set) has been designed to be a stable predictive algorithm. SIRUS uses a modified Random Forest to generate a large number of rules, and selects rules with a redundancy greater than the tuning parameter p_0. To be sure that redundancy is achieved, the features are discretized.

For a comprehensive review of rule-based algorithms we refer to [18, 19], while for a comprehensive review of interpretable machine learning we refer to [33].

[1] We have excluded algorithms developed only for binary classification, such as M5Rules [24], NodeHarvest [32], and SIRUS [2]

6.2 Datasets

We have used publicly available databases from the UCI Machine Learning Repository [10] and from [22]. We have selected six datasets for regression[2] which are summarized in Table 1, and three datasets for classification which are summarized in Table 2.

Name	$(n \times d)$	Description
Ozone	330×9	Prediction of atmospheric ozone concentration from daily meteorological measurements [22].
Machine	209×8	Prediction of published relative performance [10].
MPG	398×8	Prediction of city-cycle fuel consumption in miles per gallon [10].
Boston	506×13	Prediction of the median price of neighborhoods, [21].
Student	649×32	Prediction of the final grade of the student based on attributes collected by reports and questionnaires [6].
Abalone	4177×7	Prediction of the age of abalone from physical measurements [10].

Table 1: Presentations of the publicly available regression datasets used in this paper.

Name	$(n \times d)$	Description
Wine	4898×11	Classification of withe wine quality from 0 to 10 [10].
Covertype	581012×54	Classification of forest cover type $[1, 7]$ based on cartographic variables [10].
Speaker	329×12	Classification of accent, six possibilities, based on features extracted from the first reading of a word [12].

Table 2: Presentations of the publicly available classification datasets used in this paper.

6.3 Execution

For each dataset we perform 10 experiments. For each experiment, the data is randomly divided into a training set and a test set, with a ratio of 80%/20% each. The parameter settings for the algorithm are summarized in Table 3. These parameters were selected according to the author's recommendations to generate models based on rules with equivalent lengths.

[2] For the dataset Student we have removed variables $G1$ and $G2$ which are the first and the second grade, respectively, because the target attribute $G3$ has a strong correlation with the attributes $G2$ and $G1$. In [6] authors specify that it is more difficult to predict $G3$ without $G2$ and $G1$, although such prediction is much more useful.

For each algorithm, a model is fitted on the training set to obtain the simplicity score (4), while we measure the predictivity score (2) on the test set. To obtain the predictivity score we set

$$\gamma\left(g;(X,Y)\right) = \left(g(X) - Y\right)^2 \quad \text{and} \quad h_n = \frac{1}{n}\sum_{i=1}^{n} y_i \qquad \text{for regression,}$$

$$\gamma\left(g;(X,Y)\right) = \mathbf{1}_{g(X)\neq Y} \quad \text{and} \quad h_n = mode\left(\{y_1,\dots,y_n\}\right) \quad \text{for classification.}$$

Then, to obtain the stability score, the training set is randomly divided into two sets of equal length and two models are constructed. The code is a combination of Python and R and it is available on GitHub https://github.com/Advestis/Interpretability.

Algorithm	Parameters
CART	$max\_leaf\_nodes = 20.$
RuleFit	$tree\_size = 4,$ $max\_rules = 2000.$
Node harvest	$max.inter = 3.$
CA	$generator\_func = RandomForestRegressor,$ $n\_estimators = 500,$ $max\_leaf\_nodes = 4,$ $alpha = 1/2 - 1/100,$ $gamma = 0.95,$ $k\_max = 3$
SIRUS	$max.depth = 3,$ $num.rule = 10.$

Table 3: Algorithms parameter settings.

The choice of α's in (6) is an important step in the process of comparing interpretability. In these applications we use a PCA considering all scores generated by the 10 experiments for all data sets for regression and classification separately. We obtain the following coefficients :

$$\alpha_1 = 0.49, \quad \alpha_2 = 0.35, \quad \alpha_3 = 0.16 \quad \text{for regression.}$$
$$\alpha_1 = 0.78, \quad \alpha_2 = 0.20, \quad \alpha_3 = 0.02 \quad \text{for classification.}$$

6.4 Results for regression

The averaged scores are summarized in Table 4. As expected RuleFit is the most accurate algorithm. However, RuleFit is neither stable nor simple. SIRUS is the most stable algorithm and the Covering algorithm is one of the simplest. For all datasets, SIRUS seems to be the most interpretable algorithm among this selection of algorithms and by our measure (6).

Another interesting result is obtained from the correlation matrix table 5, which was calculated considering all results generated by the 10 experiments for all datasets. It shows that the simplicity score is negatively correlated with the predictivity score, which illustrates the well-known predictivity / simplicity trade-off. Furthermore, the stability score seems to be uncorrelated with the predictivity

Dataset	$\mathcal{P}_n$				
	RT	RuleFit	Node harvest	CA	SIRUS
Ozone	0.63	**0.72**	**0.69**	0.59	**0.69**
Machine	0.78	**0.87**	0.77	0.36	0.76
MPG	**0.79**	**0.86**	**0.79**	0.72	**0.79**
Boston	0.76	**0.88**	0.76	0.64	0.73
Student	0.13	**0.24**	**0.25**	0.17	**0.24**
Abalone	0.44	**0.56**	0.4	0.41	0.34

Dataset	$\mathcal{S}_n^q$				
	RT	RuleFit	Node harvest	CA	SIRUS
Ozone	**1.0**	0.12	**0.9**	0.31	**0.96**
Machine	0.77	0.15	**0.88**	0.06	**0.95**
MPG	**1.0**	0.19	0.87	0.35	**1.0**
Boston	0.87	0.16	0.82	0.1	**0.98**
Student	**0.94**	0.13	**1.0**	0.15	**1.0**
Abalone	**1.0**	0.19	0.84	0.23	**1.0**

Dataset	$\mathbb{S}_n$				
	RT	RuleFit	Node harvest	CA	SIRUS
Ozone	0.14	0.01	0.05	**1.0**	0.11
Machine	0.27	0.03	0.08	**0.85**	0.15
MPG	0.14	0.01	0.05	**0.97**	0.1
Boston	0.3	0.01	0.09	**1.0**	0.2
Student	0.31	0.03	0.21	0.72	**1.0**
Abalone	0.9	0.04	0.2	**0.94**	0.47

Dataset	$\mathcal{I}$				
	RT	RuleFit	Node harvest	CA	SIRUS
Ozone	**0.68**	0.39	**0.66**	0.56	**0.69**
Machine	**0.69**	0.48	**0.7**	0.33	**0.73**
MPG	**0.76**	0.49	**0.7**	0.63	**0.75**
Boston	**0.72**	0.49	**0.67**	0.51	**0.73**
Student	0.44	0.17	0.51	0.25	**0.63**
Abalone	**0.71**	0.35	0.52	0.43	0.59

Table 4: Average of predictivity score ($\mathcal{P}_n$), stability score ($\mathcal{S}_n^q$), simplicity score ($\mathcal{S}_n$) and interpretability score ($\mathcal{I}$) over a 10-fold cross-validation of commonly used interpretable algorithms for various publicly available regression datasets. Best values are in bold, as well as values within 10% of the maximum for each dataset.

score, but negatively correlated with the simplicity score, a result which is less expected.

	$\mathcal{P}_n$	$\mathcal{S}_n^q$	$\mathbb{S}_n$
$\mathcal{P}_n$	1	0.01	-0.36
$\mathcal{S}_n^q$	0.01	1	-019
$\mathbb{S}_n$	-0.36	-0.19	1

Table 5: Correlation between scores for the regressions experiments.

6.5 Results for classification

The averaged scores are summarized in Table 6. All selected algorithms have the same accuracy for all datasets. However, RIPPER and PART are both very stable algorithms, and RIPPER is the simplest of the three algorithms. Therefore, for these datasets and among these three algorithms, RIPPER is the algorithm that is most interpretable according to our measure (6).

Dataset		$\mathcal{P}_n$	
	CART	RIPPER	PART
Wine	0.13	0.12	0.01
Covertype	0.37	0.46	0.5
Speaker	0.24	0.31	0.35

Dataset		$\mathcal{S}_n^q$	
	CART	RIPPER	PART
Wine	1.0	1.0	1.0
Covertype	1.0	1.0	1.0
Speaker	0.95	1.0	1.0

Dataset		$\mathbb{S}_n$	
	CART	RIPPER	PART
Wine	0.99	0.64	0.01
Covertype	1.0	0.12	0.01
Speaker	0.71	1.0	0.45

Dataset		$\mathcal{I}$	
	CART	RIPPER	PART
Wine	0.33	0.31	0.21
Covertype	0.51	0.56	0.59
Speaker	0.39	0.46	0.48

Table 6: Average of predictivity score ($\mathcal{P}_n$), stability score ($\mathcal{S}_n^q$), simplicity score ($\mathbb{S}_n$) and interpretability score ($\mathcal{I}$) over a 10-fold cross-validation of commonly used interpretable algorithms for various publicly available classification datasets.

In contrast to the regression case, the correlation matrix table 7, which was calculated considering all scores generated by the 10 experiments for all datasets, shows that all scores are not correlated.

	$\mathcal{P}_n$	$\mathcal{S}_n^q$	$\mathbb{S}_n$
$\mathcal{P}_n$	1	0.09	-0.04
$\mathcal{S}_n^q$	0.09	1	0.06
$\mathbb{S}_n$	-0.04	0.06	1

Table 7: Correlation between scores for the classification experiments.

7 Conclusion and perspectives

In this paper we present a quantitative measure for the interpretability of tree-based algorithms and rule-based algorithms. This measure is based on the triptych: predictivity (2), stability (3), and simplicity (5), as proposed in [43,2]. The proposed methodology seems to provide an unbiased measure for ranking the interpretability of a set of algorithms by being composed of three different evaluations that allow to integrate the main components of interpretability. It may be seen from our applications that the q-stability score and the simplicity score are quite stable regardless of the datasets. This observation is related to the functioning of the algorithms; indeed, an algorithm designed for accuracy, stability or simplicity should maintain this property independently of the datasets.

It is important to note that, according to the definition 1, 100 rules of length 1 have the same interpretability index (5) as a single rule of length 100, which is debatable. Furthermore, the stability score is purely syntactical and quite restrictive. If some features are duplicated, two rules can have two different syntactical conditions, but they are otherwise identical due to their activations. One possibility to relax the stability score could be to compare the rules on the basis of their activation sets (i.e. by searching for observations where the conditions are fulfilled simultaneously).

In future work we plan to adapt the measure of interpretability to other well-known ML algorithms like linear regression or other ML problems like clustering or dimension reduction using non-negative matrix/tensor factorization (NMF, NTF). For this we will have to modify the definition for the q-stability score and the simplicity score. Another interesting extension would be the addition of a semantic analysis of the variables involved in the rules. In fact, NLP methods could be used to measure the distance between the target and these variables in a text corpus. This distance could be interpreted as the relevance of using such variables to describe the target.

References

1. Arlot, S., Celisse, A.: A survey of cross-validation procedures for model selection. Statistics Surveys **4**, 40–79 (2010). Published in Statistics Surveys
2. Bénard, C., Biau, G., Da Veiga, S., Scornet, E.: Sirus: making random forests interpretable. arXiv preprint arXiv:1908.06852 (2019)
3. Bénard, C., Biau, G., Da Veiga, S., Scornet, E.: Interpretable random forests via rule extraction. arXiv preprint arXiv:2004.14841 (2020)
4. Breiman, L., Friedman, J., Olshen, R., Stone, C.: Classification and Regression Trees. CRC press (1984)
5. Cohen, W.: Fast effective rule induction. In: Machine Learning Proceedings 1995, pp. 115–123. Elsevier (1995)
6. Cortez, P., Silva, A.M.G.: Using data mining to predict secondary school student performance. In: Proceedings of 5th FUture BUsiness TEChnology Conference (FUBUTEC 2008) (2008)
7. Dembczyński, K., Kotłowski, W., Słowiński, R.: Solving regression by learning an ensemble of decision rules. In: International Conference on Artificial Intelligence and Soft Computing, pp. 533–544. Springer (2008)
8. Doshi-Velez, F., Kim, B.: Towards a rigorous science of interpretable machine learning. arXiv preprint arXiv:1702.08608 (2017)
9. Dougherty, J., Kohavi, R., Sahami, M.: Supervised and unsupervised discretization of continuous features. In: Machine Learning Proceedings 1995, pp. 194–202. Elsevier (1995)

10. Dua, D., Graff, C.: Uci machine learning repository (2017). URL http://archive.ics.uci.edu/ml
11. Fayyad, U.M., Irani, K.B.: Multi-interval discretization of continuous-valued attributes for classification learning. In: Proc. 13th Int. Joint Conf. on Artificial Intelligence, pp. 1022–1027 (1993)
12. Fokoue, E.: UCI machine learning repository (2020). URL [WebLink]
13. Frank, E., Witten, I.H.: Generating accurate rule sets without global optimization. In: Proceedings of the Fifteenth International Conference on Machine Learning, ICML '98, p. 144–151. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA (1998)
14. Friedman, J., Popescu, B.: Predective learning via rule ensembles. The Annals of Applied Statistics pp. 916–954 (2008)
15. Friedman, J., Popescu, B., et al.: Importance sampled learning ensembles. Journal of Machine Learning Research **94305** (2003)
16. Friedman, J.H.: Greedy function approximation: a gradient boosting machine. Annals of statistics pp. 1189–1232 (2001)
17. Friedman, J.H.: Stochastic gradient boosting. Computational statistics & data analysis **38**(4), 367–378 (2002)
18. Fürnkranz, J., Gamberger, D., Lavrač, N.: Foundations of rule learning. Springer Science & Business Media (2012)
19. Fürnkranz, J., Kliegr, T.: A brief overview of rule learning. In: International Symposium on Rules and Rule Markup Languages for the Semantic Web, pp. 54–69. Springer (2015)
20. Guidotti, R., Monreale, A., Ruggieri, S., Turini, F., Giannotti, F., Pedreschi, D.: A survey of methods for explaining black box models. ACM computing surveys (CSUR) **51**(5), 1–42 (2018)
21. Harrison Jr, D., Rubinfeld, D.L.: Hedonic housing prices and the demand for clean air. Journal of environmental economics and management **5**(1), 81–102 (1978)
22. Hastie, T., Friedman, J., Tibshirani, R.: The Elements of Statistical Learning, vol. 1. Springer series in statistics Springer, Berlin (2001)
23. Holmes, G., Hall, M., Prank, E.: Generating rule sets from model trees. In: Australasian Joint Conference on Artificial Intelligence, pp. 1–12. Springer (1999)
24. Hornik, K., Buchta, C., Zeileis, A.: Open-source machine learning: R meets Weka. Computational Statistics **24**(2), 225–232 (2009). DOI 10.1007/s00180-008-0119-7
25. Karalič, A., Bratko, I.: First order regression. Machine Learning **26**(2-3), 147–176 (1997)
26. Landwehr, N., Hall, M., Frank, E.: Logistic model trees. Machine learning **59**(1-2), 161–205 (2005)
27. Letham, B., Rudin, C., McCormick, T., Madigan, D.: Interpretable classifiers using rules and bayesian analysis: Building a better stroke prediction model. The Annals of Applied Statistics **9**(3), 1350–1371 (2015)
28. Lipton, Z.C.: The mythos of model interpretability. Queue **16**(3), 31–57 (2018)
29. Lundberg, S.M., Lee, S.I.: A unified approach to interpreting model predictions. In: Advances in Neural Information Processing Systems, pp. 4765–4774 (2017)
30. Margot, V., Baudry, J.P., Guilloux, F., Wintenberger, O.: Rule induction partitioning estimator. In: International Conference on Machine Learning and Data Mining in Pattern Recognition, pp. 288–301. Springer (2018)
31. Margot, V., Baudry, J.P., Guilloux, F., Wintenberger, O.: Consistent regression using data-dependent coverings. arXiv preprint arXiv:1907.02306 (2019)
32. Meinshausen, N., et al.: Node harvest. The Annals of Applied Statistics **4**(4), 2049–2072 (2010)
33. Molnar, C.: Interpretable Machine Learning. Lulu.com (2020)
34. Murdoch, W.J., Singh, C., Kumbier, K., Abbasi-Asl, R., Yu, B.: Interpretable machine learning: definitions, methods, and applications. arXiv preprint arXiv:1901.04592 (2019)
35. Quinlan, J.R.: Induction of decision trees. Machine learning **1**(1), 81–106 (1986)
36. Quinlan, J.R.: C4. 5: programs for machine learning. Elsevier (1993)
37. Ribeiro, M.T., Singh, S., Guestrin, C.: Why should i trust you?: Explaining the predictions of any classifier. In: Proceedings of the 22nd ACM SIGKDD international conference on knowledge discovery and data mining, pp. 1135–1144. ACM (2016)
38. Rudin, C.: Please stop explaining black box models for high stakes decisions. arXiv preprint arXiv:1811.10154 (2018)
39. Shrikumar, A., Greenside, P., Kundaje, A.: Learning important features through propagating activation differences. arXiv preprint arXiv:1704.02685 (2017)

40. Tibshirani, R.: Regression shrinkage and selection via the lasso. Journal of the Royal Statistical Society. Series B (Methodological) pp. 267–288 (1996)
41. Vapnik, V.: The nature of statistical learning theory. Springer science & business media (2013)
42. Wang, Y., Witten, I.H.: Inducing model trees for continuous classes. In: Proceedings of the European Conference on Machine Learning (1997)
43. Yu, B., Kumbier, K.: Veridical data science. Proceedings of the National Academy of Sciences **117**(8), 3920–3929 (2020)