AI

MDPI

Article

A New Method to Compare the Interpretability of Rule-Based Algorithms

Vincent Margot [1,*] and George Luta [2]

[1] Advestis, 69 Boulevard Haussmann, F-75008 Paris, France
[2] Department of Biostatistics, Bioinformatics and Biomathematics, Georgetown University, Washington, DC 20057, USA; george.luta@georgetown.edu
* Correspondence: vmargot@advestis.com

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 paper is to propose the definition of a score that allows for quickly comparing interpretable 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 rule sets generated by an algorithm using two independent samples. The simplicity is based on the sum of the lengths of the rules derived from the predictive model. The proposed score is a weighted sum of the three terms mentioned above. We use this score to compare the interpretability of a set of rule-based algorithms and tree-based algorithms for the regression case and for the classification case.

Keywords: interpretability; transparency; explainability; predictivity; stability; simplicity

Citation: Margot, V.; Luta, G. A New Method to Compare the Interpretability of Rule-Based Algorithms. *AI* **2021**, *2*, 621–635. https://doi.org/10.3390/ai2040037

Academic Editor: Gianni D'Angelo

Received: 27 September 2021
Accepted: 22 November 2021
Published: 25 November 2021

1. Introduction

The widespread use of machine learning (ML) methods in many important areas such as health care, justice, defense or asset management has underscored the importance of interpretability for the decision-making process. In recent years, the number of publications on interpretability has increased exponentially. For a complete overview of interpretability in ML, the reader may see the book [1] and the article [2]. 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. One common solution is to use graphic tools, such as the Partial Dependence Plot (PDP) [3] or the Individual Conditional Expectation (ICE) [4]. A drawback of these methods is that they are limited by the human perception. Indeed, a plot with more than three dimensions cannot be interpreted by humans and so it is not useful for data sets with many features. An alternative way consists in using a surrogate model to explain the model generated by a black-box. We refer to the algorithms Local Interpretable Model-agnostic Explanations (LIME) [5], DeepLIFT [6] and SHapley Additive exPlanations (SHAP) [7] that attempt to measure the importance of a feature in the prediction process (we refer to [8] for an overview of the available methods). However, as outlined in [9], the explanations generated by these algorithms may not be sufficient to allow a reasonable decision process.

The second approach 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 Classification And Regression Trees (CART) [10], Iterative Dichotomiser 3 (ID3) [11], C4.5 [12], M5P [13], Logistic Model Trees (LMT) [14] and rule-based algorithms that are generating rule sets such as Repeated Incremental Pruning to Produce Error Reduction (RIPPER) [15], First Order Regression

(FORS) [16], M5 Rules [17], RuleFit [18], Ensemble of Decision Rules (Ender) [19], Node Harvest [20] or more recently Stable and Interpretable RUle Set (SIRUS) [21,22] and the Coverage Algorithm [23]. 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 an *If-Then* statement of the form:

$$\begin{aligned} \text{IF} \quad & c_1 \text{ In addition, } c_2 \text{ In addition, } \ldots \text{ In addition, } c_k \\ \text{THEN} \quad & \text{Prediction} = p, \end{aligned}$$

The condition part *If* is a logical conjunction, where the 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 exact mathematical definition for the concept of interpretability. This is due to the fact that interpretability involves multiple concepts as explained in [24–27]. The goal of this paper is to propose a definition that combines these concepts in order to generate an interpretability score. It is important to note that related concepts such as *justice*, *ethics*, and *morality*, which are associated with specific applications to health care, justice, defense or asset management, cannot be measured quantitatively.

As proposed in [21,26], we describe an interpretability score 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 for evaluating the *robustness* of the algorithm. The simplicity score could be conceptualized as the ability to easily verify the prediction. A simple model makes it easy to evaluate some qualitative criteria such as *justice*, *ethics* and *morality*. By measuring these three concepts, we are therefore able to evaluate the interpretability of several algorithms for a given problem.

A similar idea has been proposed in [28] in the area of the Logical Analysis of Data (LAD), by introducing the concept of Pareto-optimal patterns or strong patterns. The main part of the LAD is to select the best patterns from the dataset based on a triptych that includes simplicity, selectivity and evidence. The authors identify two extreme cases of patterns: strong prime patterns and strong spanned patterns. The first are the most specific strong patterns while the last are the simplest strong patterns. In [29], the authors have studied the effects of pattern filtering on classification accuracy. They show that the prime patterns do provide somewhat higher classification accuracy, although the loss of accuracy by using strong spanned patterns is relatively small. For an overview of the LAD, we refer the readers to [30].

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

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

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 [31], the Equation (1) includes most cases of the classical statistical models.

Given a sample $D_n = ((X_1, Y_1), \ldots, (X_n, Y_n))$, 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 Empirical Risk Minimization principle (ERM) [32], meaning that

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

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 extensively 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 $\gamma(g; (X, Y)) = (g(X) - Y)^2$. In this case, the minimizer of the risk (1) is the regression function defined by

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

If $Y \in \{0, 1\}$, we use the 0–1 contrast function $\gamma(g; (X, Y)) := \mathbf{1}_{g(X) \neq Y}$, 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. The risk (1) is a positive function, so $\mathcal{P}_n(g_n, h_n) < 1$. Moreover, if $\mathcal{P}_n(g_n, h_n) < 0$, it means that the predictor g_n is less accurate than the chosen baseline predictor h_n. Thus, in this case, it is better to use the predictor h_n instead of g_n. Hence, we can assume that the predictivity score is a positive number between 0 and 1.

3. *q*-Stability Score

Usually, stability refers to the stability of the prediction [32]. Indeed, it has been shown that stability and predictive accuracy are closely connected (see, for example, [33,34]). In this paper, we are more interested in the stability of the generated model. The importance of the stability for interpretability has been presented in [35]. Nevertheless, generating a stable set of rules is challenging as explained in [36]. In [21,22], the authors have proposed a measure of the stability for rule-based algorithms based on the following idea:

> *"A rule learning algorithm is stable if two independent estimations based on two independent samples, drawn from the same distribution $\mathbb{Q}$, result in two similar lists of rules."*

The q-stability score is based on the same definition. This concept is problematic 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 for two independent samples is zero. For this reason, this

definition of stability is too stringent in this case. One way to avoid this problem is to discretize all continuous features. The discretization of features is a common solution to control the complexity of a rule generator. In [37], for example, the authors use entropy minimization heuristics to discretize features and, for the algorithms Bayesian rule lists (BRL) [36], SIRUS [21,22] and Rule Induction Partioning Estimator (RIPE) [38], the authors have discretized the features by using their empirical quantiles. We refer to [39] for an overview of the common discretization methods.

In this paper, to generate the q-stability score, we consider a discretization process on the conditions of the selected rules based on the empirical q-quantile of the implied continuous features. Because this process is only used for the calculation of the q-stability, it does not affect the accuracy of the generated model.

First, we discretize the continuous features that are involved in the selected rules. Let $q \in \mathbb{N}$ be the number of quantiles considered for the q-stability score and let X be a continuous feature. An integer $p \in \{1, \ldots, q\}$, 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 $a \in [x_{(p_a-1)/q}, x_{p_a/q}]$.

Then, we extend this discretization to selected rules 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 $b \in [x_{(p_b-1)/q}, x_{p_b/q}]$.

Finally, 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 rule sets generated by an algorithm $\mathcal{A}$ based on D_n and D'_n, respectively. Then, the q-stability score is calculated as

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

where $Q_q(R)$ is the discretized version of the rule set 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

Simplicity as a component of interpretability has been studied in [40] for the classification trees and in [41] for the rule-based models. In [23], 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 rule set R_n is defined by*

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

Even if (4) seems naive, we consider it to be a reasonable 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 lengths, the easier their understanding should be.

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 $\mathcal{A}_1^m = \{\mathcal{A}_1, \ldots, \mathcal{A}_m\}$. 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 previously defined 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 $\mathcal{A}_i$.

We note that it would be useful to be able to calculate the simplicity score of only one algorithm. To do this, we would need to have a threshold value for the simplicity score. In practice, this information could be obtained by using a survey on the maximum size of a rule set that people are willing to accept to use.

5. Interpretability Score

In [25], 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 [25]. Indeed, a high predictivity score ensures confidence in and truthfulness of the generated model, a high stability score ensures robustness and a good noise insensibility, and a high simplicity score ensures that the generated model is easy to understand for humans and can also be easily audited.

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 interpretability score only makes sense within this set of algorithms and for the given setting.

6. Application

The goal of this application is to compare several algorithms which are considered interpretable: Regression Tree [10], RuleFit (RF) [18], NodeHarvest (NH) [20], Covering Algorithm (CA) [23] and SIRUS [22] for regression settings, and RIPPER [15], PART [42] and Classification Tree [10] for classification settings (we have excluded algorithms developed only for binary classification, such as M5Rules [43], NodeHarvest [20], and SIRUS [21]).

6.1. Brief Overview of the Selected Algorithms

RIPPER is a sequential coverage algorithm. It is based on the "divide-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, with the classes being examined in the order of increasing size. PART is also a "divide-and-conquer" rule learner. The main difference is that, in order 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 [44]. Then, the rules

are used as additional binary features in a sparse linear regression model that is using the Lasso [45]. A feature generated by a rule is equal to 1 if the rule is activated, and it is 0 otherwise.

NodeHarvest 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.

The Covering Algorithm has been designed to generate a very simple model. The algorithm extracts a sparse rule set considering all nodes and leaves of a tree ensembles (using the Random Forest algorithm, Gradient Boosting algorithm [3] or Stochastic Gradient Boosting algorithm [46]). 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* [38] to form a consistent estimator of the regression function.

SIRUS 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 [47,48], while, for a comprehensive review of interpretable machine learning, we refer to [1].

6.2. Datasets

We have used publicly available databases from the UCI Machine Learning Repository [49] and from [50]. We have selected six datasets for regression which are summarized in Table 1, and three datasets for classification which are summarized in Table 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 [51], the authors specify that it is more difficult to predict $G3$ without $G2$ and $G1$, although such prediction is much more useful.

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

Name	$(n \times d)$	Description
Ozone	330×9	Prediction of atmospheric ozone concentration from daily meteorological measurements [50].
Machine	209×8	Prediction of published relative performance [49].
MPG	398×8	Prediction of city-cycle fuel consumption in miles per gallon [49].
Boston	506×13	Prediction of the median price of neighborhoods, [52].
Student	649×32	Prediction of the final grade of the student based on attributes collected by reports and questionnaires [51].
Abalone	4177×7	Prediction of the age of abalone from physical measurements [49].

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

Name	$(n \times d)$	Description
Wine	4898×11	Classification of white wine quality from 0 to 10 [49].
Covertype	$581{,}012 \times 54$	Classification of forest cover type $[1,7]$ based on cartographic variables [49].
Speaker	329×12	Classification of accent, six possibilities, based on features extracted from the first reading of a word [53].

6.3. Execution

For each dataset, we perform 10-fold cross-validation. 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. It means that all rules generated by algorithms have a bounded length. The parameters of the algorithms are not tuned because it is not the purpose of the paper to rank the algorithms. The aim of this section is to illustrate how this score is computed.

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

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

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

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, accessed on 25 May 2021.

Table 3. Algorithms' parameter settings.

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

The choice of α's in (6) is an important step in the process of comparing interpretability. For these applications, we use an equally weighted average. It means that

$$\alpha_1 = 1/3, \qquad \alpha_2 = 1/3, \qquad \alpha_3 = 1/3.$$

Another possibility is to set each α to be inversely proportional to the variance of the associated score for each data set. In our application, the results were very similar to the equally weighted case (data not shown).

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 interesting algorithm among this selection of algorithms and by our score (6). Figures 1–3 are the box-plots of the predictivity scores, q-stability scores and simplicity scores, respectively, of each algorithms on the dataset Ozone.

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 public regression datasets. Best values are in bold, as well as values within 10% of the maximum value for each dataset.

Dataset	$\mathcal{P}_n$				
	RT	RuleFit	NodeHarvest	CA	SIRUS
Ozone	0.55	**0.74**	0.66	0.56	0.6
Machine	0.79	**0.95**	0.73	0.59	0.46
MPG	0.75	**0.85**	**0.78**	0.59	0.74
Boston	0.61	**0.74**	**0.67**	0.26	0.57
Student	0.08	0.16	**0.22**	0.13	**0.24**
Abalone	0.4	**0.55**	0.37	0.39	0.3
Dataset	$\mathcal{S}_n^q$				
	RT	RuleFit	NodeHarvest	CA	SIRUS
Ozone	**1.0**	0.11	0.92	0.24	**0.99**
Machine	0.63	0.27	**0.91**	0.17	**1.0**
MPG	**1.0**	0.14	0.87	0.25	**1.0**
Boston	0.85	0.15	0.81	0.26	**0.97**
Student	**0.98**	0.14	**1.0**	0.26	**1.0**
Abalone	**1.0**	0.21	0.86	0.25	**0.99**
Dataset	$\mathbb{S}_n$				
	RT	RuleFit	NodeHarvest	CA	SIRUS
Ozone	0.12	0.01	0.04	**0.96**	0.29
Machine	0.14	0.02	0.04	**0.9**	0.25
MPG	0.15	0.01	0.05	**0.98**	0.34
Boston	0.26	0.01	0.07	**1.0**	0.52
Student	0.37	0.05	0.25	0.91	**0.97**
Abalone	0.58	0.02	0.13	0.66	**1.0**
Dataset	$\mathcal{I}$				
	RT	RuleFit	NodeHarvest	CA	SIRUS
Ozone	0.56	0.29	0.54	**0.59**	**0.63**
Machine	**0.52**	0.41	**0.56**	**0.55**	**0.57**
MPG	**0.63**	0.33	0.57	0.61	**0.69**
Boston	0.57	0.3	0.52	0.5	**0.69**
Student	0.47	0.12	0.49	0.43	**0.74**
Abalone	0.66	0.26	0.45	0.43	**0.76**

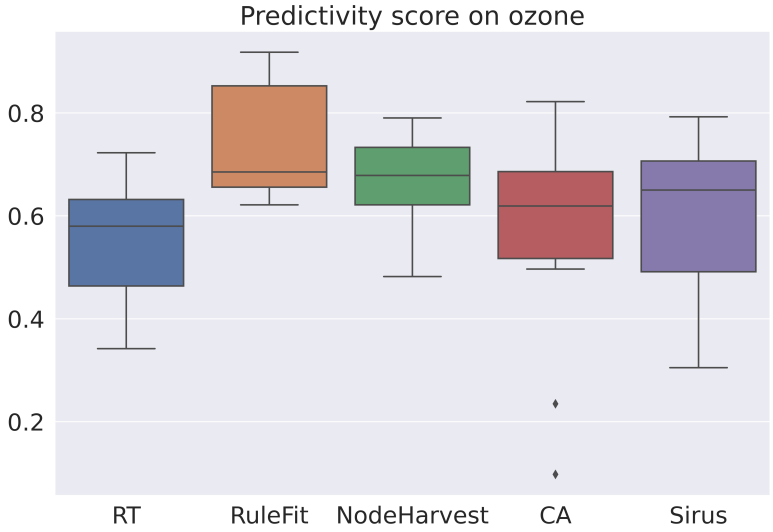

Figure 1. Box-plot of the prediction scores for each algorithm for the dataset Ozone.

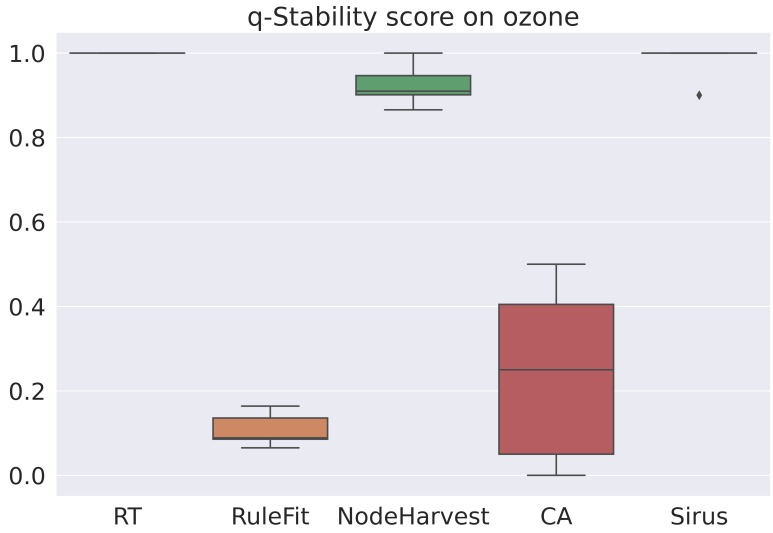

Figure 2. Box-plot of the q-stability scores for each algorithm for the dataset Ozone.

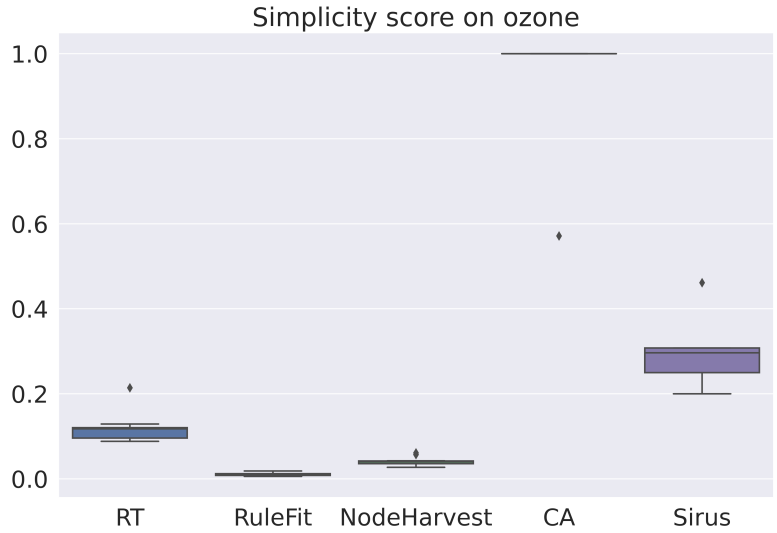

Figure 3. Box-plot of the simplicity scores for each algorithms for the data set Ozone.

Another interesting result is obtained from the correlation matrix Table 5, which was calculated considering all results generated by the 10-fold cross-validation 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 score, but negatively correlated with the simplicity score, a result which is less expected.

Table 5. Correlation between the scores for the regressions' experiments.

	$\mathcal{P}_n$	$\mathcal{S}_n^q$	$\mathbb{S}_n$
$\mathcal{P}_n$	1	−0.1	−0.27
$\mathcal{S}_n^q$	–	1	−0.10
$\mathbb{S}_n$	–	–	1

One may note that the distributions of the scores are very different. Indeed, the ranges for q-stability and simplicity are small relative to the predictivity scores. This may be explained by the fact that all algorithms are designed to be accurate, but not necessarily stable or simple. For example, SIRUS was thought to be stable, and, according to the q-stability score, it is with a score of about 1. On the other hand, stability was not considered for RuleFit, and its q-stability score is always low. We can apply the same reasoning for the simplicity score.

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). Figures 4–6 are the box-plots of the predictivity scores, q-stability scores and simplicity scores, respectively, of each algorithms for the dataset Speaker.

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 public classification datasets. Best values are in bold, as well as values within 10% of the maximum value for each dataset.

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

Table 6. *Cont.*

Dataset	$\mathcal{I}$		
	CART	RIPPER	PART
Wine	**0.71**	0.59	0.34
Covertype	**0.79**	0.53	0.50
Speaker	0.63	**0.77**	0.6

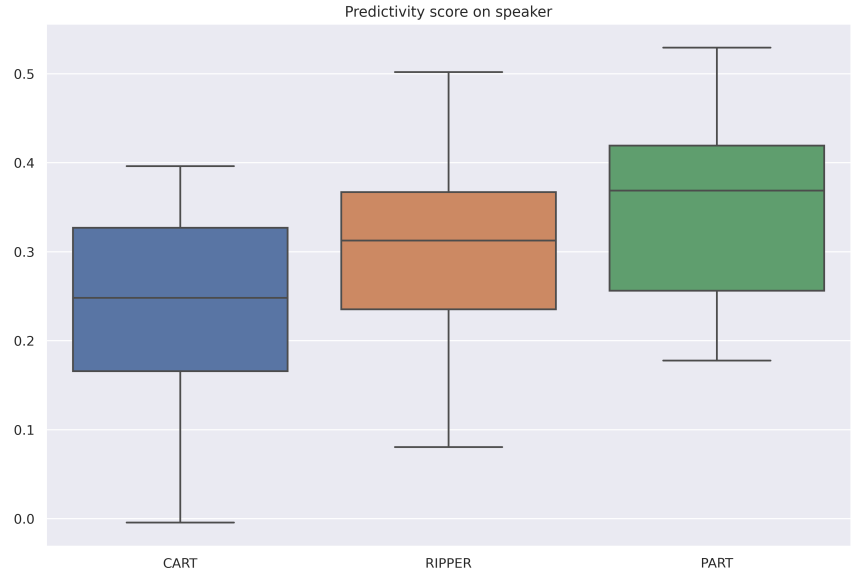

Figure 4. Box-plot of the prediction scores for each algorithm for the dataset Speaker.

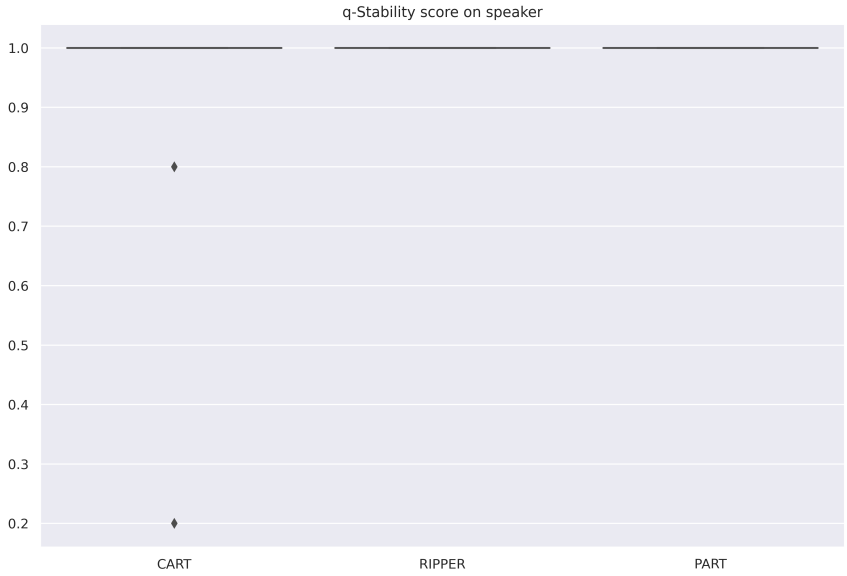

Figure 5. Box-plot of the q-stability scores for each algorithm for the dataset Speaker.

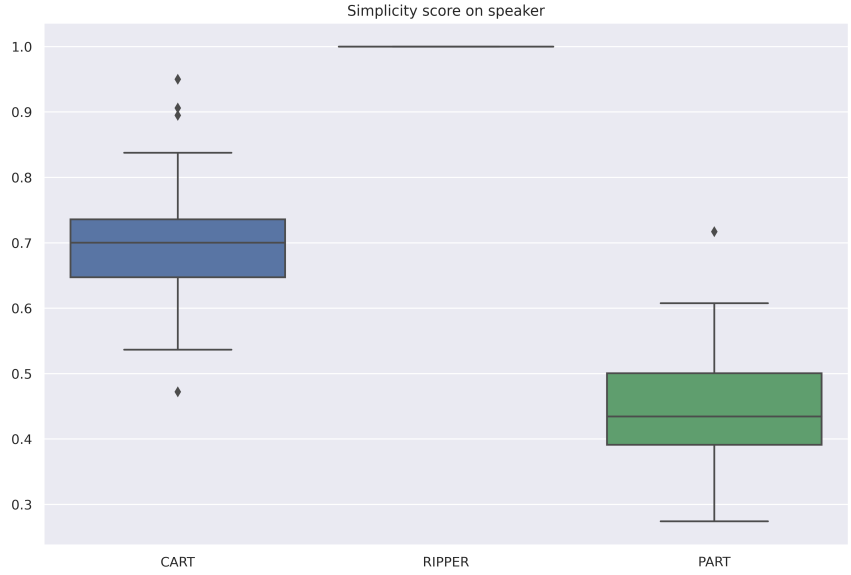

Figure 6. Box-plot of the simplicity scores for each algorithm for the dataset Speaker.

In contrast to the regression case, the correlation matrix Table 7, which was calculated considering all scores generated by the 10-fold cross-validation for all datasets, shows that the scores do not seem to be correlated.

Table 7. Correlation between scores for the classification experiments.

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

These results should take into account that, for the classification part, we have tested fewer algorithms on fewer data sets than for the regression part. The accuracy of the models for these datasets was small. If the algorithms are not accurate enough, it may not be useful to look at the other scores. The algorithms appear to be very stable, which may be explained by the fact that they are not complex. Since it is not enough to have a good predictivity score, these algorithms must be tuned to be more complex.

7. Conclusions and Perspectives

In this paper, we propose a score that may be used to compare the interpretability of tree-based algorithms and rule-based algorithms. This score is based on the triptych: predictivity (2), stability (3) and simplicity (5), as proposed in [21,26]. The proposed methodology seems to provide an easy way to rank the interpretability of a set of algorithms by being composed of three different scores that allow for integrating 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 properties of the algorithms; indeed, an algorithm designed for accuracy, stability or simplicity should maintain this property independent 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 may be 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). Another issue is the selection of the

weights in the interpretability formula (6). For simplicity, we have used equal weights in this paper, but future work is needed on the optimal choice of these weights to match the specific goals of the analyst.

As seen from the paper, the proposed interpretability score is meaningless unless it is used for the comparison of two or more algorithms. In future work, we intend to develop an interpretability score that can be computed for an algorithm regardless if other algorithms are considered or not. We also plan to adapt the measure of interpretability to other well-known ML algorithms and ML problems such as clustering or dimension reduction methods. To achieve this goal, we will need to modify the definitions of the q-stability score and the simplicity score. Indeed, these two scores can be currently computed only for rule-based algorithms or tree-based algorithms (after a transformation of the generated tree into a set of rules).

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.

Author Contributions: Conceptualization, V.M.; methodology, V.M.; software, V.M.; validation, V.M. and G.L.; formal analysis, V.M.; investigation, V.M. and G.L.; writing—original draft preparation, V.M.; writing—review and editing, V.M. and G.L.; visualization, V.M. All authors have read and agreed to the published version of the manuscript.

Funding: No funding has been provided for this research project.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: We have used publicly available databases from the UCI Machine Learning Repository https://archive.ics.uci.edu/ml/index.php (accessed on 25 May 2021) and from [50].

Acknowledgments: We thank the anonymous reviewers for their useful questions and valuable comments.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Molnar, C. Interpretable Machine Learning. 2020. Available online: https://www.lulu.com (accessed on 25 May 2021).
2. Molnar, C.; Casalicchio, G.; Bischl, B. Interpretable machine learning—A brief history, state-of-the-art and challenges. In Proceedings of the Joint European Conference on Machine Learning and Knowledge Discovery in Databases, Ghent, Belgium, 14–18 September 2020; Springer: Berlin/Heidelberg, Germany, 2020; pp. 417–431.
3. Friedman, J.H. Greedy function approximation: A gradient boosting machine. *Ann. Stat.* **2001**, *29*, 1189–1232. [CrossRef]
4. Goldstein, A.; Kapelner, A.; Bleich, J.; Pitkin, E. Peeking inside the black box: Visualizing statistical learning with plots of individual conditional expectation. *J. Comput. Graph. Stat.* **2015**, *24*, 44–65. [CrossRef]
5. 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, San Francisco, CA, USA, 13–17 August 2016; pp. 1135–1144.
6. Shrikumar, A.; Greenside, P.; Kundaje, A. Learning important features through propagating activation differences. In Proceedings of the International Conference on Machine Learning, Sydney, Australia, 6–11 August 2017; pp. 3145–3153.
7. Lundberg, S.M.; Lee, S.I. A unified approach to interpreting model predictions. In Proceedings of the Advances in Neural Information Processing Systems, Long Beach, CA, USA, 4–9 December 2017; pp. 4765–4774.
8. Guidotti, R.; Monreale, A.; Ruggieri, S.; Turini, F.; Giannotti, F.; Pedreschi, D. A survey of methods for explaining black box models. *ACM Comput. Surv.* **2018**, *51*, 1–42. [CrossRef]
9. Rudin, C. Stop explaining black box machine learning models for high stakes decisions and use interpretable models instead. *Nat. Mach. Intell.* **2019**, *1*, 206–215. [CrossRef]
10. Breiman, L.; Friedman, J.; Olshen, R.; Stone, C. *Classification and Regression Trees*; CRC Press: Boca Raton, FL, USA, 1984.
11. Quinlan, J.R. Induction of decision trees. *Mach. Learn.* **1986**, *1*, 81–106. [CrossRef]
12. Quinlan, J.R. $C_{4.5}$*: Programs for Machine Learning*; Elsevier: Amsterdam, The Netherlands, 1993.

13. Wang, Y.; Witten, I.H. Inducing model trees for continuous classes. In Proceedings of the European Conference on Machine Learning, Prague, Czech Republic, 23–25 April 1997.
14. Landwehr, N.; Hall, M.; Frank, E. Logistic model trees. *Mach. Learn.* **2005**, *59*, 161–205. [CrossRef]
15. Cohen, W. Fast effective rule induction. In *Machine Learning Proceedings*; Elsevier: Amsterdam, The Netherlands, 1995; pp. 115–123.
16. Karalič, A.; Bratko, I. First order regression. *Mach. Learn.* **1997**, *26*, 147–176. [CrossRef]
17. Holmes, G.; Hall, M.; Prank, E. Generating rule sets from model trees. In Proceedings of the Australasian Joint Conference on Artificial Intelligence, Sydney, Australia, 6–10 December 1999; Springer: Berlin/Heidelberg, Germany, 1999; pp. 1–12.
18. Friedman, J.; Popescu, B. Predective learning via rule ensembles. *Ann. Appl. Stat.* **2008**, *2*, 916–954. [CrossRef]
19. Dembczyński, K.; Kotłowski, W.; Słowiński, R. Solving regression by learning an ensemble of decision rules. In Proceedings of the International Conference on Artificial Intelligence and Soft Computing, Zakopane, Poland, 22–26 June 2008; pp. 533–544.
20. Meinshausen, N. Node harvest. *Ann. Appl. Stat.* **2010**, *4*, 2049–2072. [CrossRef]
21. Bénard, C.; Biau, G.; Da Veiga, S.; Scornet, E. Sirus: Stable and interpretable rule set for classification. *Electron. J. Stat.* **2021**, *15*, 427–505. [CrossRef]
22. Bénard, C.; Biau, G.; Veiga, S.; Scornet, E. Interpretable random forests via rule extraction. In Proceedings of the International Conference on Artificial Intelligence and Statistics, San Diego, CA, USA, 13–15 April 2021; pp. 937–945.
23. Margot, V.; Baudry, J.P.; Guilloux, F.; Wintenberger, O. Consistent regression using data-dependent coverings. *Electron. J. Stat.* **2021**, *15*, 1743–1782. [CrossRef]
24. Lipton, Z.C. The mythos of model interpretability. *Queue* **2018**, *16*, 31–57. [CrossRef]
25. Doshi-Velez, F.; Kim, B. Towards a rigorous science of interpretable machine learning. *arXiv* **2017**, arXiv:1702.08608.
26. Yu, B.; Kumbier, K. Veridical data science. *Proc. Natl. Acad. Sci. USA* **2020**, *117*, 3920–3929. [CrossRef] [PubMed]
27. Murdoch, W.J.; Singh, C.; Kumbier, K.; Abbasi-Asl, R.; Yu, B. Interpretable machine learning: Definitions, methods, and applications. *arXiv* **2019**, arXiv:1901.04592.
28. Hammer, P.L.; Kogan, A.; Simeone, B.; Szedmák, S. Pareto-optimal patterns in logical analysis of data. *Discret. Appl. Math.* **2004**, *144*, 79–102. [CrossRef]
29. Alexe, G.; Alexe, S.; Hammer, P.L.; Kogan, A. Comprehensive vs. comprehensible classifiers in logical analysis of data. *Discret. Appl. Math.* **2008**, *156*, 870–882. [CrossRef]
30. Alexe, G.; Alexe, S.; Bonates, T.O.; Kogan, A. Logical analysis of data—The vision of Peter L. Hammer. *Ann. Math. Artif. Intell.* **2007**, *49*, 265–312. [CrossRef]
31. Arlot, S.; Celisse, A. A survey of cross-validation procedures for model selection. *Stat. Surv.* **2010**, *4*, 40–79. [CrossRef]
32. Vapnik, V. *The Nature of Statistical Learning Theory*; Springer Science & Business Media: Berlin/Heidelberg, Germany, 2013.
33. Bousquet, O., Elisseeff, A.: Stability and generalization. *J. Mach. Learn. Res.* **2002**, *2*, 499–526.
34. Poggio, T.; Rifkin, R.; Mukherjee, S.; Niyogi, P. General conditions for predictivity in learning theory. *Nature* **2004**, *428*, 419–422. [CrossRef] [PubMed]
35. Yu, B. Stability. *Bernoulli* **2013**, *19*, 1484–1500. [CrossRef]
36. Letham, B.; Rudin, C.; McCormick, T.; Madigan, D. Interpretable classifiers using rules and bayesian analysis: Building a better stroke prediction model. *Ann. Appl. Stat.* **2015**, *9*, 1350–1371. [CrossRef]
37. Fayyad, U.M.; Irani, K.B. Multi-interval discretization of continuous-valued attributes for classification learning. In Proceedings of the 13th International Joint Conference on Artificial Intelligence, 28 August–3 September 1993; pp. 1022–1027.
38. Margot, V.; Baudry, J.P.; Guilloux, F.; Wintenberger, O. Rule induction partitioning estimator. In Proceedings of the International Conference on Machine Learning and Data Mining in Pattern Recognition, New York, NY, USA, 15–19 July 2018; pp. 288–301; Springer: Berlin/Heidelberg, Germany, 2018.
39. Dougherty, J.; Kohavi, R.; Sahami, M. Supervised and unsupervised discretization of continuous features. In *Machine Learning Proceedings*; Elsevier: Amsterdam, The Netherlands, 1995; pp. 194–202.
40. Luštrek, M.; Gams, M.; Martinčić-Ipšić, S. What makes classification trees comprehensible? Expert Syst. Appl. **2016**, *6*, 333–346.
41. Fürnkranz, J.; Kliegr, T.; Paulheim, H. On cognitive preferences and the plausibility of rule-based models. *Mach. Learn.* **2020**, *109*, 853–898. [CrossRef]
42. Frank, E.; Witten, I.H. Generating accurate rule sets without global optimization. In Proceedings of the Fifteenth International Conference on Machine Learning, Madison, WI, USA, 24–27 July 1998; Morgan Kaufmann Publishers Inc.: San Francisco, CA, USA, 1998; pp. 144–151.
43. Hornik, K.; Buchta, C.; Zeileis, A. Open-source machine learning: R meets Weka. *Comput. Stat.* **2009**, *24*, 225–232. [CrossRef]
44. Friedman, J.; Popescu, B. Importance sampled learning ensembles. *J. Mach. Learn. Res.* **2003**, *94305*, 1–32.
45. Tibshirani, R. Regression shrinkage and selection via the lasso. *J. R. Stat. Soc. Ser. B* **1996**, *58*, 267–288. [CrossRef]
46. Friedman, J.H. Stochastic gradient boosting. *Comput. Stat. Data Anal.* **2002**, *38*, 367–378. [CrossRef]
47. Fürnkranz, J.; Gamberger, D.; Lavrač, N. *Foundations of Rule Learning*; Springer Science & Business Media: Berlin/Heidelberg, Germany, 2012.
48. Fürnkranz, J.; Kliegr, T. A brief overview of rule learning. In Proceedings of the International Symposium on Rules and Rule Markup Languages for the Semantic Web, Berlin, Germany, 3–5 August 2015; pp. 54–69; Springer: Berlin/Heidelberg, Germany.

49. Dua, D.; Graff, C. Uci Machine Learning Repository. 2017. Available online: http://archive.ics.uci.edu/ml (accessed on 25 May 2021).
50. Hastie, T.; Friedman, J.; Tibshirani, R. *The Elements of Statistical Learning*; Springer Series in Statistics; Springer: Berlin, Germeny, 2001; Volume 1.
51. Cortez, P.; Silva, A.M.G. Using data mining to predict secondary school student performance. In Proceedings of the 5th FUture BUsiness TEChnology Conference, Porto, Portugal, 9–11 April 2008.
52. Harrison, D., Jr.; Rubinfeld, D.L. Hedonic housing prices and the demand for clean air. *J. Environ. Econ. Manag.* **1978**, *5*, 81–102. [CrossRef]
53. Fokoue, E. UCI Machine Learning Repository. 2020. Available online: https://archive.ics.uci.edu/ml/index.php (accessed on 25 May 2021).