CONCLUSIVE LOCAL INTERPRETATION RULES FOR RANDOM FORESTS

A PREPRINT

Ioannis Mollas
Aristotle University of
Thessaloniki, 54636, Greece
iamollas@csd.auth.gr

Nick Bassiliades
Aristotle University of
Thessaloniki, 54636, Greece
nbassili@csd.auth.gr

Grigorios Tsoumakas
Aristotle University of
Thessaloniki, 54636, Greece
greg@csd.auth.gr

June 20, 2022

ABSTRACT

In critical situations involving discrimination, gender inequality, economic damage, and even the possibility of casualties, machine learning models must be able to provide clear interpretations of their decisions. Otherwise, their obscure decision-making processes can lead to socioethical issues as they interfere with people's lives. Random forest algorithms excel in the aforementioned sectors, where their ability to explain themselves is an obvious requirement. In this paper, we present LionForests, which relies on a preliminary work of ours. LionForests is a random forest-specific interpretation technique that provides rules as explanations. It applies to binary classification tasks up to multi-class classification and regression tasks, while a stable theoretical background supports it. A time and scalability analysis suggests that LionForests is much faster than our preliminary work and is also applicable to large datasets. Experimentation, including a comparison with state-of-the-art techniques, demonstrate the efficacy of our contribution. LionForests outperformed the other techniques in terms of *precision*, *variance*, and *response time*, but fell short in terms of *rule length* and *coverage*. Finally, we highlight conclusiveness, a unique property of LionForests that provides interpretation validity and distinguishes it from previous techniques.

***K*eywords** Explainable Artificial Intelligence · Interpretable Machine Learning · Local Interpretation · Model-Specific Interpretation · Random Forests

1 Introduction

It is apparent that machine learning (ML) models will be integrated into our society and daily life. However, in critical domains such as banking and healthcare, where models can contain errors or may suffer from biases, it is more than necessary to ensure their transparency. Gender inequality [18], inappropriate patient treatments [8], or tricked models [11] are only a few of the common problems when automated systems are used. Thereby, it is vital for secure, fair, and trustworthy intelligent systems to be able to explain how they work and why they predict a particular outcome. In addition, guaranteeing certain virtues in the behaviour of a model allows it to be compliant with legal frameworks such as the General Data Protection Regulation (GDPR) [45] of the EU and the Equal Credit Opportunity Act of the US [1]. These needs boosted the visibility of the explainable artificial intelligence (XAI) research area in the scientific community [15]. Interpretable machine learning (IML), a subfield of XAI, attempts to address these issues by proposing techniques to shed light on the inner workings of ML models [1, 50, 4].

Random forest (RF) is an accurate learning algorithm [17] that has been proven robust to overfitting [5], as well as to learning difficulties like class imbalance or noisy and anomalous data [55]. RF excels in a lot of sectors. From applications related to security, such as intrusion detection [46], to the financial sector dealing with tasks including credit card fraud detection [32] and loan approval [58]. The presence of RF algorithms is also apparent in healthcare applications, like the patient safety culture [52] and different stages of Parkinson's disease classification [49]. Such

[1] ECOA 15 U.S. Code §1691 et seq.

models are also used in the industry sector to power fault diagnosis tools in self-aligning conveyor idlers [44], among other tasks. Finally, RF algorithms are being employed in the law sector, with one well-known application being the prediction of crime hotspots [56].

The positive aspects of RFs, in conjunction with their black box nature, have drawn the attention of the IML research community, which introduced a variety of techniques to provide interpretation in the form of rules, trees, or feature importance. For RF models in particular, interpretation techniques follow two main directions. The first one is to create a surrogate model, intending to distil the knowledge of a complex RF model into a single tree [26, 13]. The other direction is to take full advantage of the inner structure of the RF and derive information from the individual trees that make it up [29, 40, 31]. However, these approaches have significant limitations. For example, most of them only apply to binary classification tasks [40, 31]. Another point to consider is that the interpretations produced are not always valid [29, 26]. This is evident in techniques that seek to approximate the actual interpretations of a complex model.

This work presents LionForests (LF), an approach for interpreting individual predictions of RF models. LF performs path and feature reduction towards smaller interpretation rules with wider ranges by using unsupervised techniques, such as association rules and k-medoids clustering, as well as a path-oriented dissimilarity metric. We extend our preliminary work [31] in multiple dimensions. First, we provide a stable theorem and a property called conclusiveness, to support the quality and soundness of the produced rules. Besides that, we broaden LF's capabilities by making it applicable to multi-class classification tasks, and we introduce a new concept, the *"allowed error"*, to render the technique applicable to regression tasks as well. In addition, we optimise LF's performance in terms of response time per produced interpretation. Further parameterisation options are also implemented in the technique concerning the association rules and clustering algorithms. An improved and clearer method for dealing with categorical features is also provided. Moreover, a simple visualisation functionality increases the expressiveness and clarity of the generated interpretation rules. Experiments including sensitivity analysis, analysis of response time, scalability analysis, and comparison with other state-of-the-art (SOTA) interpretation techniques, as well as qualitative examples of actual interpretations, are provided to support LF's efficiency. Finally, we are investigating whether SOTA techniques have the same conclusiveness property as LF.

The rest of this paper is structured as follows. Section 2 presents the related work, while Section 3 establishes the theoretical background. Section 4 describes the LF approach, and the proposed conclusiveness property. Exhaustive experimentation from multiple perspectives, such as sensitivity analysis, time analysis, comparison with other methods, and qualitative assessments, is provided in Section 5. Section 6 presents an analysis of the experimental results. Finally, in Section 7, we discuss conclusions and future directions.

2 Related work

Over the last few years, IML has advanced so much, making the range of solutions provided wider than ever. Of the several dimensions of interpretability, two are primarily used for categorisation and comparison of such approaches. The first one refers to the global-local aspect of the model, while the second refers to the agnostic-specific applicability of the technique to a model or architecture. Local-based techniques concern the interpretation of predictions for a single instance. Global-based techniques uncover the entire structure of a model. Both local and global-based approaches include techniques that are model-agnostic or model-specific. Model-agnostic approaches can interpret any machine learning model, while model-specific techniques interpret particular types of models, or even architectures.

Another interesting aspect of interpretation techniques is the form in which they deliver their results. It is possible for a technique to provide an explanation as a set of rules, a set of feature weights, images with skewed or highlighted sections, or prototype examples. In this section, we will present a few techniques related to these dimensions in the context of ensemble models, such as random forests. These techniques share the same rule-based interpretation form. This means that the output of these techniques is a set of rules, or single rules, explaining a model or a specific prediction. Such explanations are intuitive to end users since they have a schematic and logical format, which is why we focused on this type of explanation in this work.

Surrogate models [53], based on the principle of model compression [7], are a global-based, model-agnostic interpretation technique that attempts to imitate the behaviour of more complex models. A decision tree (DT), for example, is a surrogate model of an RF model, when trained on training data and labelled by the RF model. Metrics such as fidelity have emerged in order to evaluate the ability of these models to mimic the original models. High fidelity means the approximation is sufficient. However, it is still questionable how well the surrogate models should approximate black-box models in order to be trusted [50].

Two approaches which enabled the XAI and IML areas are LIME and SHAP. LIME is a cutting-edge method for explaining machine learning models. Depending on the type of data, LIME constructs a local neighbourhood of a given size using a different algorithm. LIME searches for linear correlations between features and ML model predictions in this local neighbourhood to provide local explanations in the form of feature importance [47]. Shapley's values are a game theory-inspired approach for determining how much each "player" contributed to a collaborative game's result, feature value, and decision-making process in our case. SHAP calculates the Shapley values for an instance using a technique similar to LIME's sub/local space concept [36]. SHAP can provide the interpretation by utilising a variety of visualisations, one of which is feature importance, either local or global. A tree-specific variant known as TreeExplainer that provides faster and more accurate interpretations of trees or tree ensembles is also a part of SHAP [35].

Anchors [48] and LORE [24], local-based model-agnostic techniques leveraging synthetically generated neighbourhoods, create local surrogate models to explain particular instance predictions. Specifically, Anchors provides a single rule for interpreting a single instance's prediction. This rule is the anchor that keeps the prediction the same. On the other hand, for each instance, LORE creates a local neighbourhood using genetic algorithms. Then, it constructs a surrogate decision tree from which extracts a single rule interpretation and counterfactual instances.

The RuleFit algorithm is an intriguing technique that is similar to the knowledge distillation — student-teacher training scheme [30]. Using decision trees or tree ensembles, a set of rules is extracted and utilised as input to a sparse linear model. Because the resulting linear model bases its decisions on the most important rules, this approach is highly interpretable [22]. NodeHarvest, a technique similar to RuleFit, combines the best of both worlds: trees and tree ensembles. Generating rules from a tree ensemble using its nodes and leaves, NodeHarvest attempts to weight each rule to solve a linear quadratic problem. This results in a transparent, high-performing model [39].

Another interesting technique is the inTrees [13]. InTrees applies a series of actions like extraction, evaluation using three metrics, pruning, and selection of rules from tree ensembles, and it also calculates frequent feature interactions. Moreover, inTrees can combine these rules into a simple rule-based learner.

Single-tree approximation approaches like defragTrees [26], among others [14, 60], are global model-specific techniques that interpret tree ensembles by approximating the output of the model they attempt to explain through a single tree. However, these procedures are highly problematic because it is not feasible to summarise a complex model like tree ensembles to a single simple tree or rule-based learner, as reported by other researchers [23].

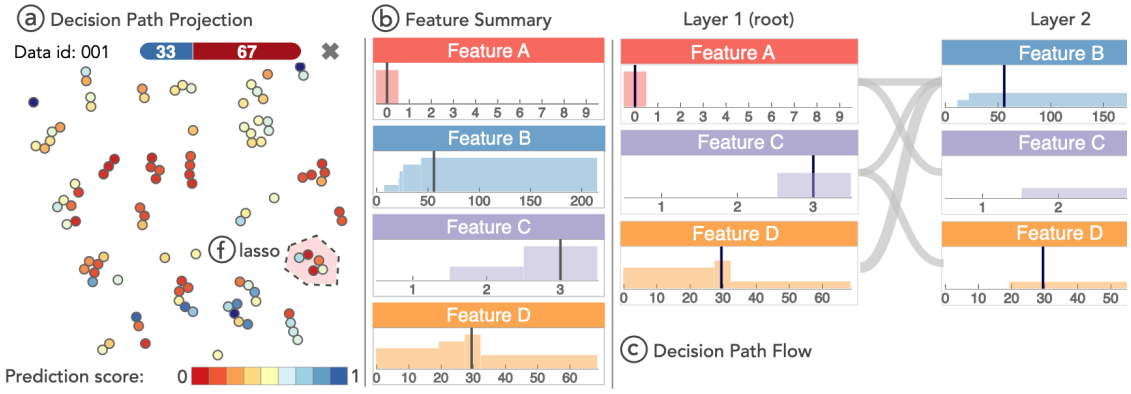

Figure 1: A sample example from the visualisation tool of iForest adapted from [59]

iForest [59] is a visualisation method that offers global and local interpretations of RF models. The most interesting aspect of this tool, though, concerns local explanations. Using a path distance metric, iForest projects the paths of an instance to a two-dimensional space (Figure 1a) using t-Distributed Stochastic Neighbour Embedding (t-SNE) [38]. Then, it provides a feature summary (Figure 1b) and a decision path flow (Figure 1c). To produce the path flow, iForest requires user input, in the form of drawing a lasso around a set of paths (Figure 1f). This is a disadvantage because the user may give incorrect input, leading to an incorrect feature summary and path flow, which will result in a flawed interpretation.

Another local model-specific interpretation strategy [40] interprets RF models by providing a collection of features with their ranges, rated based on their importance, as explanations (Figure 2). The process of interpretation comprises two steps. First, they measure the effect of a feature j on the prediction for a given instance x, and later this effect will be used for the ranking process. To accomplish this, a node per tree monitoring mechanism is used to find the

Class A (95.1%)

Rank	Feature	Influence	Min	Max
1	Feature D	+74.9	7073.5	7074
2	Feature A	+22.0	12.5	∞
3	Feature C	+19.3	0.5	1
1	Feature E	-12.27	$-\infty$	0.5
2	Feature B	-0.73	0.25	0.5
3	Feature I	-0.60	$-\infty$	174

Figure 2: Template of explanation of Moore et al. [40]

Decision	Explanation	Contrast (%)	Confidence
Class A	Feature D $\leq$ 7074 $\wedge$	−80.6	Covers 39.4% of historical
	Feature C $\neq$ 1 $\wedge$	−24.9	Matches 97.3 of covered
	Feature A $\leq$ 20	−16.9%	Vote margin 40.0%

Figure 3: Comparison of techniques in terms of the response time [29]

aggregated effect of all the features for a particular instance's prediction. The second step is to identify the narrowest range for each feature across all trees.

Finally, CHIRPS [29] proposes a technique for multi-class tasks, using frequent pattern (FP) mining on the paths of the majority class to identify the most influencing features. CHIRPS promises interpretations (Figure 3) that will be minimally complete, providing information about counterfactual cases and referring to real data, and not synthetic data. A drawback of this approach is that it lacks a strict restriction on the number of paths that will be covered by the generated rule, particularly in multi-class classification tasks. An extension of CHIRPS to gradient boosted tree ensembles is gbt-HIPS [28].

LF overcomes the issues of single-tree approximations, does not require user input like iForest, and comes with a low computational cost, in contrast to Anchors. Most importantly, LF provides rules that are always valid. Finally, it applies to a wider range of machine learning tasks compared to Anchors, LORE, and CHIRPS, among other competitors.

3 Main concepts and notation

We here define the main concepts and notation concerning decision trees and random forests, which are necessary for the presentation of LF in the next section.

3.1 Decision trees

Decision Trees (DT) [6] is a classic and well-known machine learning algorithm. DTs have played an important role in the evolution of Ensemble algorithms throughout history. A DT can be shown as an acyclic directed graph, as seen in Figure 4, containing a root node, decision nodes, and leaf nodes, which are the prediction nodes. Regarding the learning algorithm, each node concerns a particular feature f_i and a condition relation. In the case of input instances, the decision tree traces the path to a leaf node containing a prediction. The prediction can be a class, in the case of a binary or multi-class classification, or a real number estimate, in the scenario of a regression task.

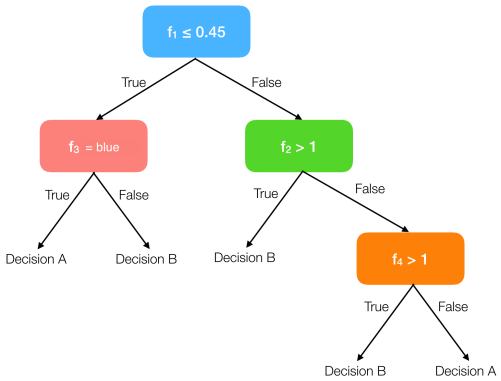

Figure 4: A simple decision tree classifier with 4 features

Each decision path p is a conjunction of conditions, and the conditions are features and values with relations $\leq$, $>$ and $=$. To provide an illustration, a path from the tree on Figure 4 would be: 'if $f_1 > 0.45$ and $f_2 \leq 1$ and $f_3 = blue$ then *Decision A*'. Thus, each path p is expressed as a set:

$$p = \{f_i \boxtimes v_j | f_i \in F, ((\boxtimes \in \{\leq, >\} \rightarrow v_j \in \mathbb{R}) \wedge (\boxtimes \in \{=\} \rightarrow v_j \in S_j))\} \quad (1)$$

where $F = f_i$ is the set of features used to train the RF model, v_j the instance's value for a feature f_i, and S_j is the set of the categorical values if f_i is a discrete feature.

However, LF's implementation relies on a library (scikit-learn [42])[2], which uses an optimised version of DTs, where the categorical features must be either encoded to numerical features, with encoding procedures like OneHot or Ordinal encoding. Specifically, when employing DTs, Onehot and Ordinal encoding are common processes. Hence, in the rest of the paper, each path p is expressed as a set:

$$p = \{f_i \boxtimes v_j | f_i \in F, v_j \in \mathbb{R}, \boxtimes \in \{\leq, >\}\}. \tag{2}$$

3.2 Random forests

One of the first ensemble algorithms using DTs as foundation is the Random Forests (RF) [5] algorithm. An RF is a collection of a specific number of trees, which are combined under an equal voting scheme. Abstractly, the inference process of an RF model can be seen as a voting procedure, where the outcome (prediction) corresponds to the majority of the votes cast.

These trees are trained under different data, bootstrap aggregation - bagging, and feature partitions, feature bagging, towards higher variance and lower bias, dealing with the overfitting problem. Then, for a specific prediction, the trees vote:

$$h(x_i) = \frac{1}{|T|} \sum_{t \in T} h_t(x_i) \tag{3}$$

where $h_t(x_i)$ is the vote cast from the tree $t \in T$ for the instance $x_i \in X$, representing the probability $P(C = c_j | X = x_i)$ of x_i to be assigned to class $c_j \in C$, and finally choosing the $\text{argmax}_{c \in \mathcal{C}} P(C = c | X = x_i)$ for classification problems, and $h_t(x_i) \in \mathbb{R}$ in regression tasks.

4 Our approach

LionForests (LF) is a local interpretability technique for RF models. LF provides a rule interpretation for an RF's decision regarding a single test instance without altering the RF model architecture and performance, as it just extracts the interpretation, exploiting the RF's knowledge. In this section, we present the core of LF, along with its extensions, which concern primarily the theoretical grounding, the main algorithm's optimisation, as well as the adaptation from binary classification to multi-class and regression tasks.

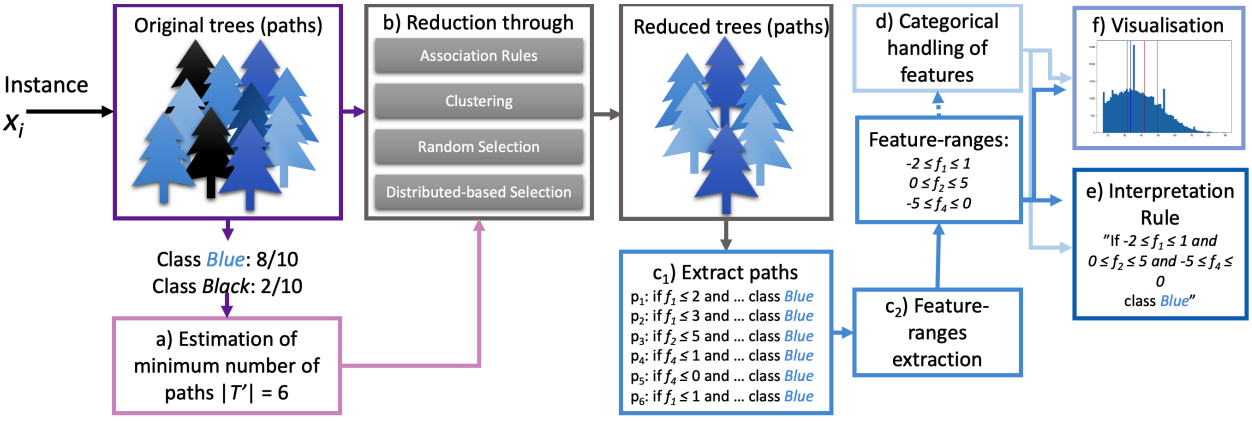

Figure 5: LionForests architecture

LF is a technique for local interpretation of binary, multi-class, and regression RF models. The ultimate purpose of LF is to give interpretations with properties related to the size of the rules, the ranges of the rules' conjunctions, and the rules' overall comprehensiveness. These are achieved via the following sequence of actions: a) estimation of the minimum number of paths, b) reduction through association rules, clustering, random selection, or distribution-based selection, c) extraction of feature-ranges, d) categorical handling of features, and, ultimately, e) composition of interpretation and f) visualisation, as depicted in Figure 5.

[2]We use scikit-learn as core library (`https://scikit-learn.org`)

Before presenting these actions, we introduce a new property called *conclusiveness* (in the following subsection), which is the last, but not least, property that we want LF to have (Section 4.1). In order to produce rules with fewer features and broader ranges, LF first estimates the minimum number of paths that have to be maintained to produce *conclusive* rules, as presented in Section 4.2. Then, it reduces the redundant paths with the different techniques introduced in Section 4.3 to achieve feature and path reduction, and extracts the feature-ranges from the rest with the procedure presented in Section 4.4. In case of categorical features, a related procedure takes place to present them more comprehensively (Section 4.5). Finally, a single rule in natural language is provided (Section 4.6), as well as a visualisation of the rule's details (Section 4.7).

4.1 Proposed conclusiveness property

conclusiveness - the quality of being final or definitely settled

We here introduce a property which concerns an aspect of a rule's quality, which we call *conclusiveness*. This property attempts to portray the quality and soundness of an explanation, specifically whether the explanation contains erroneous or misleading elements. In the name of shorter rules, such erroneous elements can include feature ranges that do not cover the correct areas or completely missing features from the interpretation rule.

"Why do we need such property in our explanations?" one might argue. So, using a simple example, let's explore why this property is essential. Assume a socially responsible customer visits a bank to enquire about loan eligibility. Based on the prediction of an RF model, a bank employee informs them that they are eligible to apply. Out of curiosity about any social biases, the customer seeks an explanation, and they obtain the following interpretation rule offered via a local post-hoc interpretability technique.

if $27 \leq age$ and $\$1000 \leq MonthlyIncome$ and $2 \leq YearsOnSameJob$ and ... then *Loan Approved*

Option 1. The customer returns home, and on the same day, they receive news from work that small temporary salary cuts will be applied for the two following months due. Their salary has been reduced from $2,200 to $1,700. Nevertheless, they applied for the loan a week later. However, when they return to the bank one week later, a bank representative informs them that the loan cannot be approved. The customer wants an explanation, claiming that the only difference in the application was the lower salary, which was eligible based on the initial explanation. As a result, the local post-hoc interpretation technique provides them with another explanation:

if $27 \leq age$ and $MonthlyIncome \leq \$2000$ and $2 \leq YearsOnSameJob$ and ... then *Loan Disapproved*

Option 2. He accepts to apply for the loan after being relieved that there is no gender discrimination in the bank's system, and he receives payment one week later. Meanwhile, in a talk with a female co-worker who also needs a loan, it appears like she will be eligible as well, based on the explanation he received last week on his loan application. When she visits the bank, they inform her that she is ineligible to apply for a loan. When she requests an explanation, she is given the following:

if $27 \leq age$ and $\$1000 \leq MonthlyIncome$ and $2 \leq YearsOnSameJob$ and $Gender = Female$ and ... then *Loan Disapproved*

In the first option, we can see that a modification within a defined range resulted in a different decision, whereas in the second option, we discovered that changing the value of a feature not included in the interpretation altered the prediction as well. In both circumstances, social and legal issues arise. For example, the algorithm discriminated against women. Furthermore, the bank must revise its algorithm and cope with fines imposed for noncompliance with legislation. We call such rules *inconclusive*. Below, we introduce the definition of a conclusive rule.

Given a dataset with a feature set $F = [f_1, f_2, \ldots, f_q]$, a number of classes $C = [c_k, c_l, \ldots, c_m]$, a predictive model h, an instance x with values for the corresponding features $[v_1, v_2, \ldots, v_q]$, and the prediction c_k, $h(x) = c_k$, we have rules like the following: "$r =$ if $2 \leq f_1 \leq 10$ and $0.5 \leq f_3 \leq 1.2$ then c_k".

$$r = \{p \rightarrow c_i | c_i \in C\}, \tag{4}$$

where p corresponds to a path as presented in Eq. 2. Then, a rule is conclusive when:

$$\text{conclusive}(r, c_k) = \begin{cases} False, & \exists v_l \in D(f_i), f_i = v_l \boxtimes v_j | f_i \boxtimes v_j \in r, \\ & h(x'_{f_i = v_l}) \neq c_k \\ False, & \exists v_l \in D(f_i), f_i = v_l | f_i \notin r, h(x'_{f_i = v_l}) \neq c_k \\ True, & \text{elsewhere,} \end{cases} \tag{5}$$

where $D(f_i)$ is the domain of feature f_i, and $x'_{f_i=v_l}$ is an alteration of instance x but with a different value (v_l) for the specific feature f_i. Therefore, a rule will be conclusive if and only if the following restrictions are met:

1. if the instance's values for the excluded features are modified to *any possible value*, then the prediction will not be influenced,
2. if the value of one of the included features is modified *within the specified range*, then the prediction will not be influenced.

A technique that always produces conclusive rules owns the property of conclusiveness. As presented in the experiments of Section 5, however, a conclusive rule contains more features than an inconclusive one, increasing the rule length and being specific to a particular instance, reducing the rule's coverage.

4.2 Estimation of minimum number of needed paths

LF attempts to identify and use only a subset of the paths that voted for the predicted class (C_M), in order to reduce the features appearing in a rule and to broaden the ranges of the features. However, to acquire the conclusiveness property, LF always has as many paths as needed to remain consistent with the original prediction. For each learning task (binary, multi-class, and regression), it is necessary to define the theoretical foundations on which the estimation of the minimum number of needed paths will rely on.

We should mention here that in LF's implementation, apart from the minimum number of paths, an additional restriction concerning the average probability of a class has been added. This happens because, unlike the original RF's majority voting schema (hard voting) [5], the implementation of RF we use [42], it combines the probabilistic predictions (soft voting) of the trees.

4.2.1 Minimum number of paths in binary tasks

First, we will go through the theoretical foundations for estimating the minimum required paths on binary tasks. Based on Proposition 1, which states that we need at least $\frac{|T|}{2}+1$ of the paths in order to maintain the same prediction C_M, LF will select at least a quorum. For example, if we had $|T|=100$ trees in an RF model and 89 of them voted for class M for an instance x_i, it has to select at least 51 of the 89 paths extracted from the trees in order to produce the interpretation rule.

Proposition 1 *An RF model, with T trees casting $|T|$ votes, predicts always class M (C_M) if and only if class M has at least a $quorum$ of votes or more, where $quorum = \frac{|T|}{2}+1$ out of $|T|$ votes.*

As a result, the optimisation problem (Eq. 6) is formulated to minimise the number of features ($|F'|$) that satisfy a reduced set of paths ($|T'|$). The constraint is to keep the same classification result as the original set of trees, making the number of the reduced paths equal to or greater than the quorum. The final interpretation rule will include any feature that appears in these paths, guaranteeing the property of conclusiveness.

$$
\begin{aligned}
\underset{F' \subseteq F}{\text{minimise}} \quad & |F'| \\
\text{subject to} \quad & p = \{f_i \boxtimes v_j | f_i \in F'\}, p \in P_t \; \forall \, t \in T', \\
& \lfloor \frac{1}{|T|} \sum_{t \in T'} h_t(x_i) + \frac{1}{2} \rfloor = \lfloor \frac{1}{|T|} \sum_{t \in T} h_t(x_i) + \frac{1}{2} \rfloor, \\
& |T'| \geq \frac{|T|}{2} + 1
\end{aligned}
\tag{6}
$$

An example for the equation $\lfloor \frac{1}{|T|} \sum_{t \in T} h_t(x_i) + \frac{1}{2} \rfloor$ follows. When 70 out of $|T|=100$ trees are voting for class 1, then we have $\lfloor \frac{1}{100} 70 + 0.5 \rfloor = \lfloor 1.2 \rfloor \rightarrow 1$. On the other hand, if 25 out of $|T|=100$ trees are voting class 1 (the minority), then we have $\lfloor \frac{1}{100} 25 + 0.5 \rfloor = \lfloor 0.75 \rfloor \rightarrow 0$. Therefore, we are aiming to find a subset $T' \subseteq T$, which will produce the same classification as the original T trees, with a smaller feature set $|F'|$.

4.2.2 Minimum number of paths in multi-class tasks

In a multi-class classification task, the voting system is more complex than in a binary task, and thus Proposition 1 cannot cover every case. The following theorems are presented in order to apply the framework of LF to multi-class tasks. A running example will be given, along with the propositions and theorems, for better comprehension.

An RF model always predicts class M (C_M) if and only if class M has a majority of votes, while any other class J (C_J) has fewer votes. Let, the available classes for the RF model with $|T| = 100$ trees be C_1 (*red*), C_2 (*blue*) and C_3 (*green*). For a specific case, each tree voted, and the result was $|C_1| = 45$, $|C_2| = 35$ and $|C_3| = 20$. Thus, the prediction was *red* with $|C_1| = |C_M| = 45$, and the second highest voting class was *blue* with $|C_2| = 35$. Proposition 1 cannot be used in this example because $45 = |C_1| = |C_M| < quorum = \frac{|T|}{2} + 1 = 51$.

Based on this, the result of the RF model is class M with the majority of votes. However, if we minimise the voting paths to the majority class votes, as we did in binary tasks (only if the number of votes is less than the quorum), and extract the interpretation rule, it is possible for another class to gain votes from other classes (see the following running example). This is happening because the paths voting for those classes will not be covered by the interpretation rule. Then, if a feature not appearing in the interpretation rule changes its value, this may lead to a change in the votes (predictions) of the decision trees voted for the other classes, if this feature was important to them. This may lead to a reallocation of votes between the other classes, and maybe a class will surpass the originally first class. As a consequence, the outcome of the prediction could change.

Lemma 1 *An RF model always predicts class M (C_M) if and only if class M has a majority of votes, while any other class J (C_J) can not exceed the votes of class M by obtaining at least $S = |C_M| - |C_J| + 1$ votes from the other classes.*

In our running example, if we apply Lemma 1, we can assume that RF will always predict the *red* class if the other classes do not exchange votes. For example, if $|C_1| - |C_2| + 1 = 45 - 35 + 1 = 11$ votes from the *green* class move to the *blue* class, the number of votes from the *blue* class would rise to $|C_2| = 35 + 11 = 46$, which is more than the *red* class. Or if $|C_1| - |C_3| + 1 = 45 - 20 + 1 = 26$ votes from the *blue* class move to the *green* class, the number of votes from the *green* class would rise to $|C_3| = 20 + 26 = 46$ more than the *red* class. As a result, if we had reduced the paths to the number of class *red* votes with LF, then the other votes could change and, therefore, the outcome of the prediction could also change, while Lemma 1 would be invalid.

Proposition 2 *An RF model always predicts class M (C_M) if and only if the K number of votes from any other class remains stable, where $K = |T| + |C_L| - |C_M| + 1$, including the C_M and C_L, votes from the majority class and the second most voted class, respectively, out of the $|T|$ votes.*

Proposition 2 argues that in situations where the majority of the class has less than a quorum of votes, the number of votes can be reduced to K without affecting the outcome of the prediction. Maintaining only the votes of the two key classes, majority class and second most voted class, is not enough and hence we need to retain $K_{other} = K - |C_M| - |C_L|$ from the other classes, also at random.

Thus, in our running example, the minimum number of paths we can reduce to is $K = 100 + 35 - 45 + 1 = 91$. From these 91 paths, we will retain the 45 of class 'red' (C_1) and the 35 of class 'blue' (C_2) paths. Then we would have to hold $K_{other} = 91 - 45 - 35 = 11$ paths from the remaining classes, in this example from the 'green' class (C_3) which holds 20 paths. We use the LF's techniques, which will be detailed later in this section, to select 11 out of 20 paths. Thus, holding 91 paths out of 100, the rest of the 9 paths of the class 'green', if they all switch from 'green' to 'blue', the class 'blue' will not exceed the votes of the class 'red', and the result will remain the same. This is presented in Theorem 1.

Theorem 1 *For $R = |T| - |C_M| - |C_L|$ as the remainder of the votes and $K_{other} = K - |C_M| - |C_L|$ as the remainder of the votes to be selected, S is always greater than $R - K_{other}$, which means that there is not a sufficient number of votes to be received by class L, or any other class, in order to surpass the votes of class M.*

In order to prove that the aforementioned Theorem holds, we will use contradiction to prove that $R - K_{other} < S$ is true.

Proof 1 *To prove Theorem 1 by contradiction we assume that the statement $R - K_{other} < S$ is false. We will prove that $R - K_{other} \geq S$ is true.*

$$
\begin{aligned}
R - K_{other} \geq S \iff \\
(|T| - |C_M| - |C_L|) - (K - |C_M| - |C_L|) \geq |C_M| - |C_L| + 1 \iff \\
|T| - |C_M| - |C_L| - K + |C_M| + |C_L| \geq |C_M| - |C_L| + 1 \iff \\
|T| - K \geq |C_M| - |C_L| + 1 \iff \\
|T| - (|T| + |C_L| - |C_M| + 1) \geq |C_M| - |C_L| + 1 \iff \\
|T| - |T| - |C_L| + |C_M| - 1 \geq |C_M| - |C_L| + 1 \iff \\
|C_M| - |C_L| - 1 \geq |C_M| - |C_L| + 1 \iff \\
-1 \geq 1
\end{aligned}
$$

Thus, we proved that the statement $R - K_{other} \geq S$ is not true. ■

Therefore, the number of paths we can keep in order to always maintain the same classification result, is provided by the following rules:

- If $|C_M| \geq quorum$ then apply LF reduction to features and paths as it happens to binary tasks, based on Proposition 1,
- If $|C_M| < quorum$ then identify the class with the second-highest number of votes, denoted as C_L. The number of paths we need to keep is equal to $quorum' = |C_L| - |C_M| + T + 1$. Then, we keep the paths from the C_M and C_L, and a random selection process is employed to collect $quorum'$ from the R remaining paths.

As we will see in Section 4.4, by aggregating these paths into a single rule, we explain the "majority class", even though we have paths voting for different classes. Therefore, the paths voting for other classes contribute to the formation of the aggregated explanation for why the "majority class" was the predicted class.

Nonetheless, the end user is given the option of specifying the desired number of paths that LF must reduce to, or the average probability, for both binary and multi-class classification. In a binary configuration classification task with 100 trees, for example, instead of 51, which would have been the $quorum$, a user would choose to adjust the minimum number of trees to 80, or the average probability to 80%. The LF will then attempt to minimise the paths to this number. This is extremely similar to the strategy used in the regression tasks, which is discussed in the following section. However, for the rest of this work, we will assume that we always aim to reduce to the bare minimum of trees as stated in this and the prior section.

4.2.3 Minimum number of paths in regression tasks

The estimation of the required number of paths, as defined in the binary and multi-class tasks, is not applicable for regression. Hence, we introduce an algorithm to adapt LF to regression models. In RF for regression, the prediction is determined by the average of the individual trees' predictions of the RF,

$$h(x_i) = \frac{1}{|T|} \sum_{t \in T} h_t(x_i) \tag{7}$$

where $h_t(x_i) \in \mathbb{R}$. On this basis, we cannot assume that by removing a few trees (for example, preserving a quorum) we can get the same outcome. To overcome this, we introduce the $allowed\_error$ definition.

The idea is to reduce the number of features and paths with regard to an $allowed\_\ error$. We measure the mean absolute error of the RF model with respect to the test set, and we set it as the default value for $allowed\_error$. We also let the user set a preferred $allowed\_error$. A sensitivity analysis between $allowed\_error$ and the feature and path reduction ratio is provided in Section 5.2 to guide users to choose the appropriate $allowed\_error$ for their application based on their needs. The value of $allowed\_error$ is strongly associated with the importance of each task. We have a regression model that predicts the indication of blood sugar. The importance of the error has a mandatory sense, since it involves a health issue. On the other hand, when predicting, e.g., the quality of wine, the error might be less sensitive.

In order to measure the error of an interpretation and to use the following reduction techniques, the min and max predictions per tree of RF must be collected. Thus, for each tree of an RF model, we traverse to its leaves to identify the min and max predictions that the tree can provide. This kind of information will be used to estimate the worst-case error that may be added to the prediction given an interpretation. Therefore, the final interpretation will have the following form:

if $0.47 \leq f_1 \leq 0.6$ and $22 \leq f_3 \leq 54$ then Prediction: $24.15 \pm local\_error$,
where $local\_error \in [0, allowed\_error]$

Algorithm 1: Calculation process of $local\_error$

Input: $trees, allowed\_error, tree\_stats$
Output: $local\_error$

$L, K \leftarrow reduction\_method(trees, allowed\_error)$
$original\_pred \leftarrow 0, prediction \leftarrow 0$
for $tree \in L$ **do**
  $original\_pred \leftarrow original\_pred + tree.predict(instance)$
  $prediction \leftarrow prediction + tree.predict(instance)$
end
for $tree \in K$ **do**
  $tree\_pred = tree.predict(instance)$
  $min\_pred = tree\_stats[tree].min$
  $max\_pred = tree\_stats[tree].max$
  $extreme\_pred \leftarrow max\_pred$
  if $|tree\_pred - min\_pred| > |tree\_pred - max\_pred|$ **then**
    $extreme\_pred \leftarrow min\_pred$
  end
  $original\_pred \leftarrow original\_pred + tree\_pred$
  $prediction \leftarrow prediction + extreme\_pred$
end
$prediction = \frac{prediction}{|L+K|}$
$original\_pred = \frac{original\_pred}{|L+K|}$
$local\_error \leftarrow |original\_pred - prediction|$
return $local\_error$

The $local\_error$ is calculated based on a set of paths $|L|$. Three algorithms are introduced and presented in the following paragraphs that will lead to a selection of $|L|$ trees from the initial $|T|$ trees. The $|K|$ remaining trees, which will be excluded from the final interpretation rule, can cause an error in the prediction. That is because the rule does not completely cover the decision of these trees, and thus, for a change in the instance to a feature that does not appear in the rule, those trees may produce different predictions, and the final prediction may change to:

$$h'(x) = \frac{1}{|T|}\left(\sum_{t \in T \setminus K} h_t(x_i) + \sum_{t \in K} e_t(x_i)\right) \tag{8}$$

$$e_t(x_i) = \begin{cases} min\_pred_t, & |h_t(x_i) - min\_pred_t| > |h_t(x_i) - max\_pred_t| \\ max\_pred_t, & elsewhere \end{cases}$$

Consequently, for those $|K|$ trees, we replace their predictions with the min or max prediction that each tree can provide, choosing between them on the basis of which, min or max, is the most distant to the original prediction of the tree. The aforementioned process is depicted in Algorithm 1, which is the basis for Algorithm 6 presented later in Section 4.3.4.

4.3 Reduction techniques

We now introduce the reduction techniques applied after identifying how many paths we can reduce. We define four reduction techniques that we use to build rules with fewer features and larger ranges. Table 1 depicts an overview of the applicability of each algorithm presented in the following sections to the corresponding tasks. All the techniques aim to reduce both the paths and the features, but with different degrees of effectiveness.

4.3.1 Reduction through association rules

The first reduction process, reduction through association rules (AR), is applicable to all learning tasks with small variations, and it starts with the implementation of the association rules [2]. In association rules, the attributes are

		Reduction through				
		AR	CR	RS	DS	
					Inner	Outer
	Binary	✓	✓	✓	-	-
	Multi-class	✓	✓	✓	-	-
	Regression	$\checkmark_{variant}$	-	✓	✓	✓
Designed Towards	Feature Reduction	✓	✓	-	-	-
	Path Redution	-	-	✓	✓	✓

Table 1: Overview of algorithms we use for feature and path reduction for each task

called items $I = \{i_1, i_2, \ldots, i_n\}$. Each dataset contains sets of items called itemsets $IS = \{is_1, is_2, \ldots, is_m\}$, where $is_i \subseteq I$. Using all the possible items of a dataset, we can find all the rules $X \Rightarrow Y$, where $X, Y \subseteq I$. X is called antecedent, while Y is called consequent. The purpose of the association rules is to determine the support and confidence of every rule in order to find useful relations. A straightforward observation is that X is independent of Y when the confidence level is particularly low. What is more, we can tell that X has high support, which implies it is probably very significant.

Algorithm 2 describes the reduction through association rules. The association rules will be implemented at the path level. The I items will contain the F features of our original dataset. The IS itemsets, which will be used to mine the association rules, will contain a collection of features that reflect each path $is_i = \{i_j | i_j = f_j, f_j \boxtimes v_k \in p_i, p_i \in P\}$. It is important to mention that we keep only the presence of a feature in the path, and we discard the value of v_j, in order to have itemsets only with features. It is then possible to apply association rules methods. To compute the association rules, we use Apriori [3] and FP-growth [25], influenced by comparative studies of association rules algorithms [41, 57], presenting these as options for the reduction through the association rules process.

The next step is to sort the association rules produced by the algorithm based on the ascending confidence score. For the rule $X \Rightarrow Y$ with the lowest confidence, we will take items of X and add them to an empty list of features F'. After that, we determine the number of paths containing conjunctions that are satisfied with the new feature set F'. We stop when we acquire a number of paths either equal to or more than the estimated required paths (as presented in Sections 4.2.1 and 4.2.2), or when we have a $local\_error$ from the paths that cannot be valid with the new feature set, smaller or equal to the $allowed\_error$ (Section 4.2.3).

Otherwise, we iterate and add more features taken from the next antecedent of the next in confidence score rule. By using this strategy, we keep the number of features low and smaller than the original $F' \subseteq F$. Reducing the features can also lead to a reduced collection of paths, as paths containing conjunctions with redundant features would no longer be valid. So, we have the following representation for every p path:

$$p = \{f_i \boxtimes v_j | f_i \in F', v_j \in \mathbb{R}, \boxtimes \in \{\leq, >\}\}. \tag{9}$$

It is evident that this reduction technique favours feature reduction, as its main criteria are based on the association rules discovered in the feature sets of the paths. We should also mention here that, in contrast to our preliminary work, we optimised the feature set extracted from the frequent patterns (Algorithm 2, L.11-18), and now the reduction through the association rules process is providing timely responses. This is also evident through our experiments in Section 5.3.

4.3.2 Reduction through clustering

On binary and multi-class tasks, a second reduction strategy based on clustering (CR) is used[3]. Aside from k-medoids [33], which was used in our preliminary work, OPTICS [51] and Spectral Clustering [37] (SC) were added as additional choices. These algorithms are well-known clustering algorithms which need a distance or dissimilarity metric to find optimum clusters. Thus, the clustering of paths will require a distance or dissimilarity metric between two paths. Therefore, firstly, a similarity metric has been constructed (Algorithm 3), in order to transform it to a dissimilarity metric in a way that favours the absence of a feature from both paths. Specifically, if a feature is missing from both paths, the similarity of these paths increases by 1. When a feature is present in both paths, the similarity increases by a value between 0 and 1, i.e., the intersection of the two ranges normalised by the union of the two ranges. The similarities have a range of $[0, 1]$, where 0 means the paths are not similar at all, and 1 means the paths are identical. To

[3]Reduction through clustering was not used in regression because reduction through association rules almost reaches the maximal local error allowed, and the overhead of clustering does not justify the effort

Algorithm 2: Reduction through association rules

Input: $paths, features, trees, minimum\_number\_paths/allowed\_error, task$
Output: $reduced\_paths, reduced\_feature\_set$

$itemsets \leftarrow []$; // Empty list of itemsets
for $path \in paths$ **do**
  $itemset \leftarrow []$
  for $f_i \in path$ **do**
    $itemset \leftarrow itemset + f_i$
  end
  $itemsets \leftarrow itemsets + itemset$
end
$frequent\_itemsets \leftarrow ar(itemsets)$; // ar can be Apriori, FP-growth
$antecedent\_features \leftarrow [], antecedents \leftarrow \emptyset$
for $antecedent \in frequent\_itemsets[antecedents]$ **do**
  if $antecedent \notin antecedents$ **then**
    $antecedents \leftarrow antecedents + antecedent$
    for $feature \in antecedent$ **do**
      if $feature \notin antecedent\_features$ **then**
        $antecedent\_features \leftarrow antecedent\_features + feature$;
    end
  end
end
$reduced\_paths \leftarrow [], reduced\_feature\_set \leftarrow []$
if $task \in \{regression\}$ **then** $local\_error \leftarrow 2 * allowed\_error$;
$k \leftarrow 1$
while $((task \in \{binary, multi-class\} \wedge |reduced\_paths| < minimum\_number\_paths) \vee (task \in \{regression\} \wedge local\_error > allowed\_error)) \wedge k < |antecedent\_features|$ **do**
  $reduced\_feature\_set \leftarrow reduced\_features + antecedent\_features[k-1]$
  $redundant\_features \leftarrow []$
  for $feature \in features$ **do**
    if $feature \notin reduced\_feature\_set$ **then** $redundant\_features \leftarrow redundant\_features + feature$;
  end
  $reduced\_paths \leftarrow []$
  for $pathinpaths$ **do**
    if $\forall feature \in path, feature \notin redundant\_features$ **then** $reduced\_paths \leftarrow reduced\_paths + path$;
  end
  if $task \in \{regression\}$ **then** $local\_error \leftarrow compute\_error(paths, reduced\_paths)$;
  $k \leftarrow k + 1$
end
return $reduced\_paths, reduced\_feature\_set$

convert the similarity metric to a dissimilarity metric, we subtract the measured similarities from 1 ($1 - similarity$), and we have a range of $[0, 1]$, where 1 means the paths are distant and 0 means the paths are identical.

Using one of the aforementioned clustering algorithms (by default we use k-medoids), the clusters are estimated using the dissimilarity metric discussed above. Subsequently, the ordering of the clusters is performed based on the number of paths they cover. Then, paths from larger clusters are accumulated into a list, until at least the required number of paths (as estimated in Section 4.2) has been obtained.

By collecting paths from larger clusters first, the probability of reducing the features is increasing. This is happening because the paths inside a cluster appear to be more similar among them, and therefore, similar paths will contain similar features. In addition, the biased dissimilarity metric will cluster paths with fewer insignificant features, leading to a subset of paths that are satisfied with a smaller set of features. As a result, we can assume that the reduction through clustering is also oriented toward feature reduction. The corresponding procedure is also represented in Algorithm 4.

Algorithm 3: Path similarity metric

input : $p_i, p_j, feature\_names, min\_max\_feature\_values$
return: $similarity_{ij}$

$s_{ij} \leftarrow 0$
for $f \in feature\_names$ **do**
  if $f \in p_i \wedge f \in p_j$ **then**
    find l_i, u_i, l_j, u_j lower and upper bounds
    $inter \leftarrow min(u_i, u_j) - max(l_i, l_j)$, $union \leftarrow max(u_i, u_j) - min(l_i, l_j)$
    if $inter > 0 \wedge union \neq 0$ **then** $s_{ij} \leftarrow s_{ij} + inter/union$;
  else if $f \notin p_i \wedge f \notin p_j$ **then**
    $s_{ij} \leftarrow s_{ij} + 1$
  end
end
return $s_{ij}/|feature\_names|$

Algorithm 4: Reduction through clustering

Input: $paths, minimum\_number\_paths, clusters$
Output: $reduced\_paths, reduced\_feature\_set$

$dissimilarity \leftarrow []$
for $i \in [0, |paths|]$ **do**
  $vector \leftarrow []$
  for $j \in [0, |paths|]$ **do**
    if $i = j$ **then**
      $vector \leftarrow vector + [0]$
    else
      $vector \leftarrow vector + [compute\_dissimilarity(path[i], path[j])]$
    end
  end
  $dissimilarity \leftarrow dissimilarity + [vector]$
end
$clusters = cl(dissimilarity, clusters)$; // cr can be k-medoids, OPTICS, SC
$sorted\_clusters = sort\_by\_size(clusters)$
$reduced\_paths \leftarrow []$, $k \leftarrow 0$
while $|reduced\_paths| < minimum\_number\_paths \wedge k < |antecedent\_features|$ **do**
  for $path \in sorted\_cluster[k]$ **do**
    $reduced\_paths \leftarrow reduced\_paths + path$
  end
  $k \leftarrow k + 1$
end
$reduced\_feature\_set \leftarrow \emptyset$; // Empty set of features
for $path \in reduced\_paths$ **do**
  for $f_i \in path$ **do**
    $reduced\_feature\_set \leftarrow reduced\_feature\_set + f_i$
  end
end
return $reduced\_paths, reduced\_feature\_set$

4.3.3 Reduction through random selection

In the case of reduction through clustering, random selection (Algorithm 5) (RS) of paths is used to achieve the minimum number of paths. For binary and multi-class tasks, RS randomly removes the unnecessary paths to only keep the minimum required number of paths. On the regression tasks, if the $local\_error$ after the AR is smaller than the $allowed\_error$, RS is applied in order to minimise the paths while maintaining the $local\_error$ less or equal to the $allowed\_error$.

Algorithm 5: Reduction through random selection

Input: $paths, minimum\_number\_paths/allowed\_error, task$
Output: $reduced\_paths, reduced\_feature\_set$

$reduced\_paths \leftarrow []$
if $task \in \{binary, multi-class\}$ **then**
  for $i \in [0, minimum\_number\_paths]$ **do**
    $j \leftarrow random(0, |paths|)$
    $reduced\_paths \leftarrow reduced\_paths + paths[j]$
    $paths \leftarrow paths - paths[j]$
  end
else
  $local\_error \leftarrow 2 * allowed\_error$
  $last\_path \leftarrow \emptyset$
  $temp\_paths \leftarrow paths$
  while $local\_error < allowed\_error \wedge |paths| > 1$ **do**
    $j \leftarrow random(0, |temp\_paths|)$
    $last\_path \leftarrow temp\_paths[j]$
    $temp\_paths \leftarrow temp\_paths - temp\_paths[j]$
    $local\_error \leftarrow compute\_error(paths, temp\_paths)$
  end
  $reduced\_paths \leftarrow temp\_paths + last\_path$
end
$reduced\_feature\_set \leftarrow \emptyset$; // Empty set of features
for $path \in reduced\_paths$ **do**
  for $f_i \in path$ **do**
    $reduced\_feature\_set \leftarrow reduced\_feature\_set + f_i$
  end
end
return $reduced\_paths, reduced\_feature\_set$

4.3.4 Reduction through distribution-based selection

The last technique, reduction through distribution-based selection (DS), is a reduction technique exclusively designed for regression tasks. This method utilises normal distributions, is not combined with any other method, and it attempts to reduce the features and the paths of a rule for a given instance. Each instance's prediction is the average of the individual predictions of the trees. These predictions have their own distribution.

In Figure 6, the actual distribution of the RF's prediction is shown in blue, while the green distribution is a normal distribution. The $generate\_distribution$ function creates this normal distribution by generating an array of a predetermined size and populating it with random values that belong to a Gaussian distribution with the mean and standard deviation (σ) values of the original distribution. We can change the way this artificial distribution is created by multiplying or dividing the σ value with a parameter s, generating wider or narrower distributions around the mean value. Therefore, for different values of s, we apply one of the following techniques, inner or outer selection, to choose the paths we will keep for the final rule. We select the distribution's σ value, which achieves the lowest number of paths with a $local\_error \in [0, allowed\_error]$ for an instance interpretation. This process is also presented in Algorithm 6, which is based on Algorithm 1.

Inner selection The distribution-based inner selection (DSi) technique, for all the different σ values, it selects the paths inside the range of the generated distribution to form the interpretation rule. For the paths outside the generated distribution, we compute the $local\_error$. This approach is going to reduce the number of paths, when the majority of trees are providing similar predictions close to the mean value. Then, by isolating and removing distant predictions, even if they shift, the mean will not be greatly affected, and thus the $local\_error$ will be relatively small.

Outer selection Similarly to inner selection, the distribution-based outer selection (DSo) technique, for all the different σ values, it selects the paths outside the range of the generated distribution to form the interpretation rule. Again, for the paths outside of the generated distribution, the $local\_error$ is computed. This approach is providing better results when the original distribution is either like an inverse normal distribution or uniform. In this case, the majority

Algorithm 6: Reduction through distribution-based selection

Input: $instance, trees, allowed\_error, variation$
Output: $reduced\_paths, reduced\_feature\_set, error$

$reduced\_paths \leftarrow [tree..decision\_path(instance), \forall tree \in trees]$
$error \leftarrow 0\ predictions \leftarrow [tree.predict(instance), \forall tree \in trees]$
$mean \leftarrow \frac{1}{|predictions|} \sum_{prediction \in predictions} prediction$
$\sigma \leftarrow \sqrt{\frac{1}{|predictions|} \sum_{prediction \in predictions} (prediction - mean)^2}$
for $s \in [.1, .2, .5, 1, 2, 4, 5, 6, 7, 8, 9, 10, 20, 50, 100]$ **do**
  $normal\_distribution \leftarrow generate\_distribution(mean, \frac{\sigma}{s})$
  $normal\_min = min(normal\_distribution)$
  $normal\_max = max(normal\_distribution)$
  $local\_reduced\_paths \leftarrow [], new\_prediction \leftarrow 0$
  for $tree \in trees$ **do**
    $path \leftarrow tree.decision\_path(instance), prediction \leftarrow tree.predict(instance)$
    if
     $(variation \in \{DSi\} \wedge \neg(prediction < normal\_min \vee prediction > normal\_max)) \vee (variation \in \{DSo\} \wedge (prediction < normal\_minorprediction > normal\_max))$ **then**
      $local\_reduced\_paths \leftarrow local\_reduces\_paths + path$
      $new\_prediction \leftarrow new\_prediction + prediction$
    else
      $distance\_with\_min \leftarrow |prediction - tree.min\_prediction|$
      $distance\_with\_max \leftarrow |prediction - tree.max\_prediction|$
      else if $distance\_with\_min > distance\_with\_max$ **then**
        $new\_prediction \leftarrow new\_prediction + tree.min\_prediction$
      end
      $new\_prediction \leftarrow new\_prediction + tree.max\_prediction$
    end
    $local\_error \leftarrow |mean - \frac{1}{|trees|} new\_prediction|$
    if $local\_error < allowed\_error \wedge |reduced\_paths| > |local\_reduces\_paths|$ **then**
    end
    $error \leftarrow local\_error, reduced\_paths \leftarrow local\_reduced\_paths$
  end
end
$reduced\_feature\_set \leftarrow \emptyset$; `// Empty set of features`
for $path \in reduced\_paths$ **do**
  for $f_i \in path$ **do**
    $reduced\_feature\_set \leftarrow reduced\_feature\_set + f_i$
  end
end
return $reduced\_paths, reduced\_feature\_set, error$

of the predictions will be distant to the mean value. Keeping those predictions, and leaving out from the interpretation rule the predictions closer to the mean, we will probably have a low $local\_error$, because the predictions closer to the mean will be fewer. Moreover, those predictions are less likely to deviate a lot when small changes to the input will occur, thus the error most of the time will be smaller than the estimated $local\_error$.

4.4 Feature-ranges formulation

In this section, we demonstrate how feature-ranges are formulated in a simple binary classification situation without taking into account the aforementioned reduction strategies. The formulation procedure is the same for all tasks, and it takes place after path reduction. The following illustrative example shows both a) why the reduction is required and b) how the aggregation of paths to a single rule occurs.

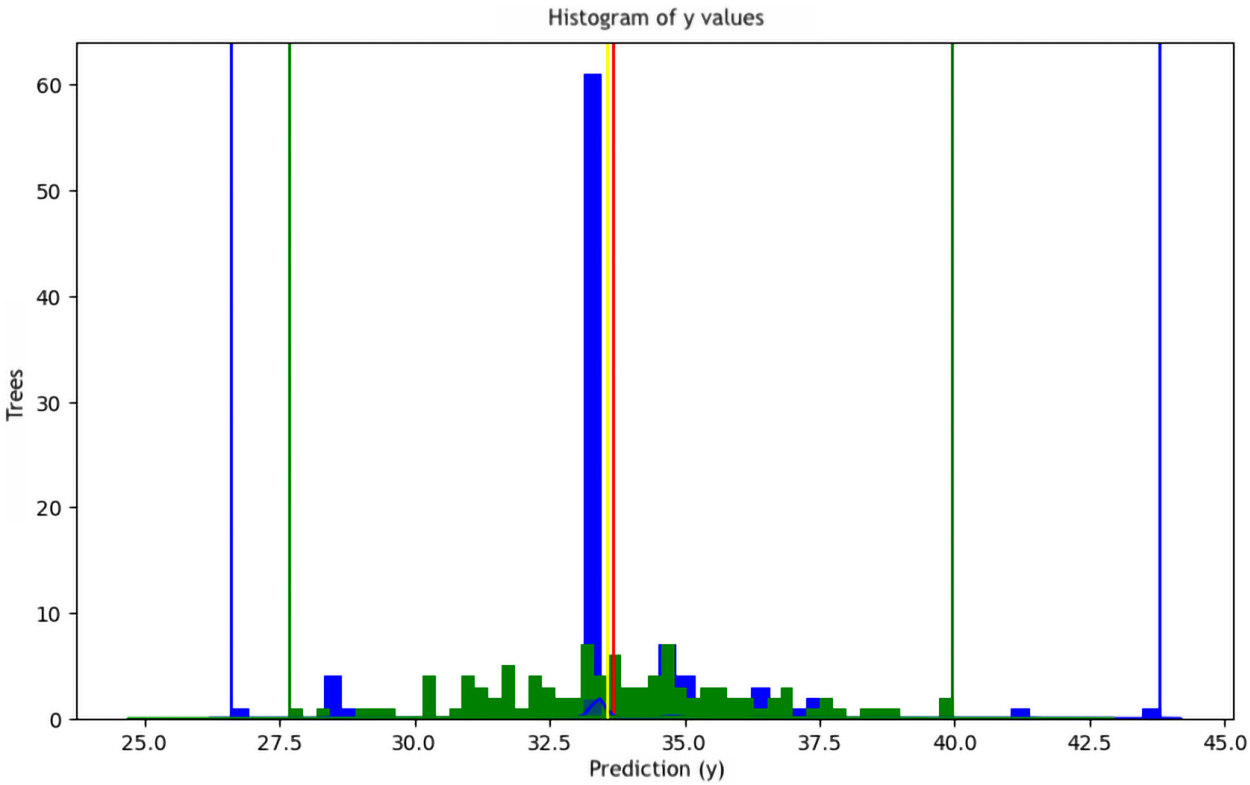

Figure 6: Distribution of predictions of a given instance

Consider an RF model of T trees that predicts instance x as $c_j \in C = \{0, 1\}$. We concentrate on the $K \geq \frac{|T|}{2} + 1$ trees that classify x as c_j. For each feature, we calculate the range of values imposed by the conditions in the root-to-leaf path of each of the K trees in which it appears.

For a concrete, real-world example, we train the RF model with $T = 50$ trees on the Banknote dataset [16]. We focus on the *skew* feature of Banknote and a test instance x, whose *skew* value is 0.25. The decision of the RF for this instance is supported by $K = 39$ of its trees. Figure 7 presents the value ranges for *skew* in each of the 38 trees, whose decision paths contain the *skew* feature. Figure 8 presents the corresponding stacked path ranges of *skew* feature, which shows the number of paths in the y axis whose value ranges include the corresponding value in the x axis.

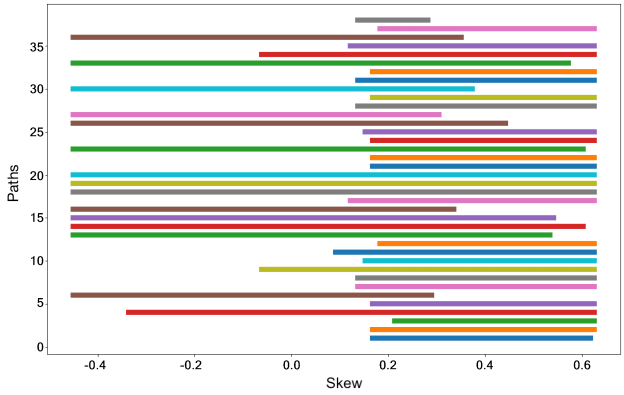

Figure 7: Path ranges of 'skew'

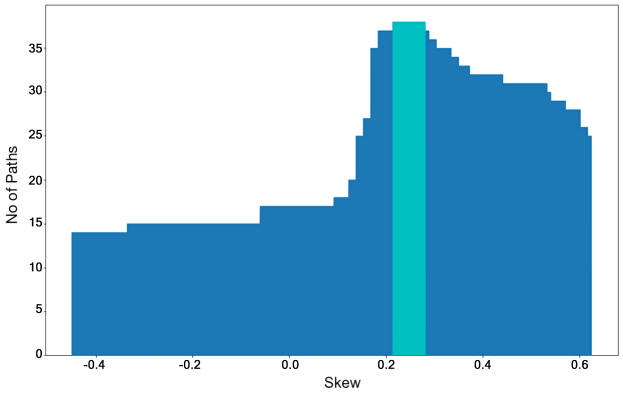

Figure 8: Stacked path ranges of 'skew'

The cyan/light grey region of Figure 8 is the intersection between these 38 decision paths for *skew*. Then, we could say that the range of this feature will be: $0.21 \leq skew \leq 0.29$, where $skew = 0.25$ for this instance. This intersection will always contain the value (0.25) for the particular feature of the examined instance. Furthermore, no matter how

much the value of the feature may shift, the decision paths will not change as long as it remains within this intersection. Thus, if the instance's value for *skew* changes from 0.25 to 0.28, each tree will take its decision via the same path. Summing up the above, the interpretation may have the following shape:

'if $0.21 \leq skew \leq 0.29$ and ... then *Fake Banknote*'

These paths will be used in order to form a final single rule. However, using all the available trees will produce interpretations with many features, making them less comprehensible to the end user. Moreover, a large number of paths can lead to small, strict, and very specific feature ranges. For example, the *skew* value of the instance x was 0.25 and the intersection range of all paths for this feature is $0.21 \leq skew \leq 0.29$, while the global range of *skew* is $[-1, 1]$. If the 10 first paths corresponding to the first 10 top rows (top to bottom) of the stacked area plot (Figure 8) could have been reduced, the range would get $0.15 \leq skew \leq 0.4$, thus a broader one.

A narrow range, like the one mentioned above, will result in an overly specific range that is neither useful nor informative. For example, $25 \leq age \leq 26$ is not preferred over $22 \leq age \leq 30$ since the latter covers a broader range and appears more trustworthy, stable, and reliable, indicating that minor changes in age will not affect the prediction. On the other hand, a wider range would be less rebuttable and preferable. Hence, this is the reason we compute the minimum number of paths required (Section 4.2), and then we apply reduction techniques (Section 4.3). Therefore, we can formulate aggregated rules that address both the large number of feature-ranges and the appearance of very narrow ranges if we minimise the number of pathways.

In this work, we also developed a much faster way of computing these intersections. This affected the responsiveness of any operation, including the extraction of an explanation without reduction. The following assumptions about the performance are confirmed by the time analysis studies in Section 5.3.

4.5 Categorical features

One more LF's adaptation concerns the handling of categorical features. Focusing only on OneHot encoded cases, we designed a handling procedure that is in accordance with this principle: if the OneHot encoded feature appears in the reduced paths, then the categorical feature where it originates is added to the rule. If this does not exist in the reduced paths, it means that the prediction has not been affected by this value of the categorical feature. However, we are looking for the other OneHot encoded features of the same categorical feature appearing in the reduced rules, but not in the instance (their values are equal to zero), and we present them to the user, along with the interpretation, as categorical values that may change the prediction if the category changes to one of them.

To give a brief example, let us have a feature $height$ with values $= [short, normal, tall]$. After OneHot encoding, there are three features: $height_{short}$, $height_{normal}$ and $height_{tall}$. An instance with $height = short$ will therefore have $height_{short} = 1$, $height_{normal} = 0$, and $height_{tall} = 0$ after encoding. If the feature $height_{short}$ appears in the extracted paths, it means that it influences the prediction and will appear in the final rule as '$height = short$'. Elsewhere, this indicates that this feature value does not affect the prediction. Then, we check whether the features $height_{normal}$ and $height_{tall}$ appear in the reduced paths. If any of them appear on the paths, it means that they influence the prediction if they change. Therefore, they are provided to the user as alternative values, or they can appear in the rule in the form '$height \neq [normal, tall]$', only if '$height = short$' has been reduced. In any case, we want to provide alternative values visually in order to have less cluttered rules. More information about visualising alternative values may be found in Section 4.7 and in the example presented in Section 5.5.

4.6 Interpretation composition

The last stage of LF combines the extracted ranges of features and the information derived from the categorical handling, if there was any categorical feature, in a single natural language rule. The order of appearance of the ranges of features in the rule is determined by a method of global interpretations, called permutation importance [5]. An example of an interpretation is as follows:

'if $0 \leq f_1 \leq 0.5$ and $-0.5 \leq f_2 \leq 0.15$ and $f_3 =$ *short* then *Prediction*'. (10)

This rule can be interpreted in this way: "As long as the value of the f_1 is between the ranges 0 and 0.5, and the value of f_2 is between the ranges -0.5 and 0.15, and f_3 equals *short*, the system will predict this instance as *Prediction*. If the value of f_1, f_2 or both, surpasses the limits of their ranges, or f_3 changes category, then the prediction may change. Note that the features are ranked through their influence". This type of interpretation is intuitive, human-readable, and more comprehensible than other interpretations, like feature importance. Furthermore, this rule is conclusive since

it will always include all the feature ranges appearing in the paths that produced the specific prediction, while these paths will be greater than or equal to the minimum number of paths required to maintain the prediction stable.

4.7 Visualisation

Apart from extending LF to multi-class and regression tasks, as well as optimising the core functionalities of LF, we are also presenting a proposed visualisation layout. Using this interface, the user will see the explanation rule for a prediction, and then have the ability to choose to inspect each of the features that appear in the rule. In the case of a feature with numerical values, the user will be supplied with a distribution plot with the values of the feature. Two vertical green lines are being used to represent the range that appears in the rule, while two blue vertical lines reflect the ranges of the original rule without applying any reduction technique.

On the other hand, if the selected feature is categorical, then if alternative values are available as described in Section 4.5, they will be visible to the user. When the original value of a categorical feature has been reduced by the rule, the alternative values become meaningful. This is because if this value is apparent to the rule, it implies that altering it to any other value may change the prediction. However, when the rule reduces the original value, the alternative values give helpful information to the end user as to which values may alter the prediction. An example is provided in Section 5.5 and Figures 9 and 10.

5 Evaluation

To support our proposed methodology, we conducted experiments on response time, a scalability analysis, and a sensitivity analysis to examine how RF parameters affect feature and path reduction. In addition, we provided a comparative analysis with similar well-known methodologies as well as a qualitative evaluation involving examples. LF's code and evaluation experiments are available at the GitHub repository "LionLearn"[4]. The code is written in Python.

5.1 Setup and datasets

For the series of experiments, we have used the Scikit-learn's [42] RandomForestClassifier and RandomForestRegressor. We used 8 different datasets, 3 for binary, 2 for multi-class, 2 for regression and 1 for multi-class and regression problems. The datasets for binary tasks are: Banknote [16], Heart (Statlog) [16] and Adult (Census) [34]. Glass [16] and Image Segmentation (Statlog) [16] for multi-class tasks. For regression, we used Wine Quality [12] and Boston Housing [27]. Additionally, Abalone [10] was used for both multi-class and regression. For the scalability analysis, we generated 16 datasets using the '*make_classification*' and '*make_regression*' functions of scikit-learn to generate datasets with 10, 50, 100 and 1000 features, and 1 (regression), 2 (binary), 10 and 100 (multi-class) targets.

We applied a 10-fold cross-validation grid search for each dataset using the following set of parameters: *depth* $\in$ {1, 5, 7, 10} (depth of each individual tree), *max features* $\in$ {'sqrt', 'log2', 75%, None = all features} (max number of features per individual tree), *min samples leaf* $\in$ {1, 2, 5, 10}, *bootstrap* $\in$ {True, False} (bootstrap samples or the whole dataset is used to train each tree), *estimators* $\in$ {10, 100, 500, 1000} (number of trees in the ensemble). The scoring metric of the grid search was the weighted F_1-score for the binary and multi-class datasets, and the mean absolute error (mae) for the regression datasets. Table 2 and 3 shows the number of instances and features of each dataset, the parameters' values that achieved the best F_1-score/mae for each dataset, along with the scores itself.

Dataset	Task	Classes	Instances	Features	Categorical
Banknote	Binary (B)	2	1372	4	-
Heart (Statlog)	Binary (B)	2	270	13	1
Adult (Census)	Binary (B)	2	45167	12 (85)	7 (80)
Glass	Multi-Class (MC)	6	214	9	-
Image Segmentation	Multi-Class (MC)	7	2310	19	-
Abalone	MC/R	4/1	4027	8	-
Boston	Regression (R)	1	506	13	-
Wine Quality	Regression (R)	1	4898	12	-

Table 2: Datasets' statistics

[4] https://git.io/JY0gF

Dataset	F_1/mae	# Estimators	Max Depth	Max Features	Min Samples Leaf	Bootstrap
Banknote	99.49%	500	10	0.75	1	True
Heart (Statlog)	84.60%	500	5	sqrt	5	False
Adult (Census)	84.84%	1000	10	0.75	2	True
Glass	71.95%	1000	10	sqrt	2	True
Image Segmentation	96.98%	500	10	0.75	1	False
Abalone	68.8%/1.4	10/100	10	0.75	10/5	True
Boston	2.95	1000	10	0.75	1	True
Wine Quality	0.55	1000	10	0.75	5	True

Table 3: Performance and best parameters for each dataset

5.2 Sensitivity analysis

We have conducted the following sensitivity analysis experiments in order to compare the capability of LF to reduce features (feature reduction ratio - FR%) and paths (path reduction ratio - PR%) while tuning both the parameters of LF (LF) and the configuration of the Random Forests (RF) model. This study also acts as an ablation study, since we evaluate LF utilising the various reduction strategies on their own and in combination. In order to simplify the information obtained from the analysis, we present each task (binary, multi-class, regression) separately. The parameters of the RF model under examination are the *depth* $\in$ {1, 5, 7, 10}, *max features* $\in$ {sqrt, log2, 75%, None} and *estimators* $\in$ {10, 100, 500, 1000}. LF parameters tested in this sensitivity analysis are the *AR algorithm* (AR) $\in$ {Apriori (AP), FP-growth (FP)}, *CL algorithm* (CR) $\in$ {k-medoids (k), OPTICS (OP), Spectral Clustering (SC)}, and *method* $\in$ {1, 2, 3, 12, 13, 23, 123} for the binary and multi-class classification, where 1 represents the use of AR, 2 the use of CL and 3 the use of RS. Therefore, 123 means the utilisation of all 3 methods, in this order. For regression, we used *method* $\in$ {AR+RS, DSi, DSo}, while an additional parameter of LF, $allowed\_error$ is also examined in the regression tasks. Therefore, the following analysis is based on these experiments. More information can be found in Appendix A.

Dataset	# Estimators	Max Depth	Max Features	AR	CL	RS	FR%	
Banknote	500	5	0.75	AP/FP	-	-	41%	
Heart (Statlog)	1000	5	0.75	AP/FP	k	✓	37%	
Adult (Census)	500	5	0.75	AP/FP	SC	-	42%	
Glass	100	5	None	AP	OP	✓	40%	
Segmentation	1000	7	0.75	FP	SC	✓	34%	
Abal. mc	100	1	0.75	-	SC	-	26%	
Abal. r	1000	5	None	AP/FP	-	-	37%	1.8 (1.6)
Boston	500	1	sqrt/log2	-	✓	-	41%	4.5 (2.3)
Wine	500	1	log2	-	✓	-	52%	0.6 (0.4)
					DSi	DSo		$allowed\ error$

Table 4: Parameters of RF and LF achieved the highest FR% based on the sensitivity analysis. Allowed Error (in parentheses is Local Error) is the parameter of LF for the regression tasks as described in Section 4.3.4.

The first sensitivity analysis we carried out applies to the Adult, Banknote and Heart (Statlog) datasets (binary classification). The parameters of LF that attained the highest FR and PR ratios are summarised in Tables 4 and 5. Table 4 shows that AR, using either the Apriori or FP-growth algorithm, is apparent in all three binary classification datasets, implying that it is required to achieve the maximum FR ratio. Consequently, we may say that CL is required in two of the three circumstances, while RS is required in one. On the other hand, we note that when the RF model includes 500 *estimators* or more, with a *depth* of 5 and 75 percent of the features per tree, we achieve a higher FR.

In terms of the PR ratio, Table 5 shows that RS was employed in all three cases, whereas AR and CL were used in only one. As a result, if we want to attain high FR and PR ratios, we recommend employing all three algorithms in binary classification datasets. When the RF models contain 1K estimators, a *depth* of equal to 10, and all the features accessible for each tree, LF appears to obtain a greater PR ratio.

Dataset	# Estimators	Max Depth	Max Features	AR	CL	RS	PR%	
Banknote	1000	10	None	-	-	✓	49%	
Heart (Statlog)	1000	7	None	AP/FP	k	✓	43%	
Adult (Census)	1000	10	None	-	-	✓	49%	
Glass	1000	10	None	AP/FP	SC	✓	48%	
Segmentation	1000	10	0.75	-	-	✓	49%	
Abal. mc	1000	10	None	AP/FP	SC	✓	40%	
Abal. r	500	1	0.75	-	✓	-	51%	1.8 (1.6)
Boston	500	1	sqrt/log2	-	✓	-	66%	4.5 (2.3)
Wine	1000	1	sqrt/log2	-	✓	-	85%	0.6 (0.4)
					DSi	DSo		$allowed\ error$

Table 5: Parameters of RF and LF achieved the highest PR% based on the sensitivity analysis. Allowed Error (in parentheses is Local Error) is the parameter of LF for the regression tasks as described in Section 4.3.4.

We continue with the multi-class datasets Glass, I. Segmentation and Abalone. Table 4 indicates that CL is present in all three multi-class classification datasets, using either the SC or OPTICS algorithm, showing that it is essential to attain the maximum FR ratio. We also see that AR is necessary in two of the three cases, whereas RS is required in the third. LF appears to perform best in terms of RF parameters when attempting to interpret models with 100 *estimators* or more, and with 75 percent or more of the features per tree. However, the impact of the *depth* parameter on the FR ratio cannot be determined from this table.

Table 5 indicates that RS was utilised in all three cases with the highest PR ratio, whereas AR and CL were used in two of the three. When the RF models have 1K *estimators*, a *depth* of 10, and all the features available for each tree, the LF models appear to have a higher PR ratio. Consequently, we advocate using all three techniques in multi-class classification datasets if we want to achieve high FR and PR ratios.

The last set of experiments is related to the analysis of the regression datasets, Abalone, Boston and Wine. Table 4 suggests that DSi is found in two of the three multi-class classification datasets, indicating that it is utilised to ensure the highest FR ratio. AR+RS is also employed in one of the three cases, but DSo is not used in any of them. In general, LF appears to perform best in terms of RF parameters when interpreting models with 500 or more *estimators* and a *depth* of 1 in two cases, and with 1K *estimators* and a *depth* of 5 in the third case. This table, however, cannot be used to determine the effect of the *max features* parameter on the FR ratio.

Table 5 shows that DSi was employed in all three cases with the highest PR ratio, whereas AR+RS and DSo were not. The LF models appear to have a higher PR ratio when the RF models include 500 or more *estimators*, a *maximum depth* of 1, and the root number of features available for each tree. As a result, we recommend utilising DSi in regression datasets in order to attain high FR and PR ratios.

Appendix A has a more in-depth sensitivity analysis with extensive statistics and explanations. Practitioners may use this extensive study to see how LF will perform in their existing models, or to decide whether they want to adjust their model (if the performance does not change significantly) to take advantage of higher LF performance.

5.3 Time and scalability analysis

Optimisation of LF implementation was an emerging issue to address, mainly due to the need for timely responses. The speed of providing explanations to a model's predictions in a variety of applications is indeed a very important factor, specifically, for on-line applications The second direction of our experiments targets the analysis of response time. We are comparing LF to its preliminary version in order to back up the assumption of Sections 4.3.1 and 4.4. For this analysis we use only the binary datasets of Banknote and Heart (statlog) because the preliminary version of LF is only applicable to binary tasks, and we do not use the Adult dataset, as the old version was unable to finish our exhaustive analysis within a relatively reasonable time.

The first aspect of the algorithm, we are going to inspect, concerns the identification of the ranges for each feature appearing in a rule without applying any reduction method. In Figure 25 and 26, we can see that LF is outperforming the preliminary version, with a huge decrease of 93% for Banknote and 95% for Heart datasets. Then, we compare the two versions on the generation of rules applying reduction. We remind here that in the reduction pipeline we optimised the reduction through association rules, thus this experiment will inspect the effect of this optimisation. In Figure 27

and 28, we can see that LF is once again outperforming the preliminary version, with a huge decrease of 94% for Banknote and 96% for Heart datasets.

We also carried out a scalability analysis. We investigated the runtime of LF under different RF setups by creating synthetic datasets with 10, 50, 100, and 1000 features for regression, binary (2 classes), and multi-class classification (10 and 100 classes). We particularly included RF models with varying number of *estimators*: 10, 100, 500, and 1000, as well as maximum *depth*: 1, 2, 5, and 10.

Starting with the binary setup, we found from our investigation that the *depth* parameter has the most impact on the runtime of LF. When the *depth* is small, such as 1 or 2, an explanation takes 0.3 to 0.4 seconds to generate. With a *depth* of 5, the explanation takes around 2.6 seconds, while with a *depth* of 10, it takes about 7 seconds. These are the average runtime results for various *depth* values for all combinations of the number of features in the dataset and the number of estimators.

In our experiments with multi-class classification tasks, we noticed that 10 classes and 100 classes perform almost identically in terms of runtime. This implies that the number of classes has no effect on LF's runtime performance. Similarly to binary classification, *depth* is the parameter which affects mostly the runtime of LF in these tasks as well. When *depth* equals 1 or 2 the runtime is between 0.55 to 0.71, while when equals 5, the runtime reaches almost 3 seconds. When *depth* is greater, for example 10, the runtime increases to 7.5 seconds.

In both cases, we can observe that the average runtime for datasets with 10 to 100 features is about one and a half seconds, including even 1000 *estimators* and a *depth* of 10. With a bigger dataset of 1000 features, the average time increases to 7 seconds.

Finally, neither the *depth* nor the number of *estimators* appeared to affect LF's performance in the regression task. Actually, the average runtime performance across all configurations is 0.4 seconds. Even with 1000 features, the average runtime is 0.48 seconds. The performance in the previously worst situations, with 1000 features, 1000 estimators, and a *depth* of 10, was 0.9.

More extensive analysis is available in the Appendix B, where the influence of each RF parameter is reflected in the LF runtime performance on each task.

5.4 Comparison study with other techniques

In our experiments, we include three techniques that produce direct rules as explanations from the literature, and they take into consideration the RF model to be explained. Those techniques are Anchors [48] (AN), CHIRPS [29] (CH) and DefragTrees [26] (DF), as well as a simple surrogate model based on a decision tree in a global (GS) and a local (LS) fashion, to use it as a baseline. AN is a local, model-agnostic interpretation technique for binary and multi-class classification. CH is a local, model-specific technique for interpreting random forest models for binary and multi-class classification tasks. DF is a global, model-specific technique that interprets tree ensembles on binary and multi-class classification and regression. The applicability of each of these techniques to each dataset is presented in Table 6.

	Baseline Surrogate	Defrag Trees	Anchors	CHIRPS
Banknote	✓	✓	✓	✓
Heart (Statlog)	✓	✓	✓	✓
Adult (Census)	✓	✓	✓	✓
Image Segmentation	✓	✓	✓	✓
Abalone (MC)	✓	✓	-	-
Abalone (R)	✓	✓	-	-
Wine Quality	✓	✓	-	-

Table 6: Applicability of interpretation techniques to each dataset

We use four different evaluation metrics and compute the runtime of each algorithm on the different datasets. The first metric computes the length of an explanation for an instance ($rule\_length$). In scenarios involving end-users, we want a lower $rule\_length$, while in high-risk applications we would like a higher $rule\_length$. The second measures the coverage of a rule, which describes the fraction of instances of a dataset that satisfy the antecedent of that rule ($coverage$). Higher coverage is better. The third is the precision of a rule, which measures the performance of that rule on the covered instances ($precision$), measuring the actual $precision$ in classification problems and the *mean absolute error* (mae) in regression problems. High $precision$ and low mae are better. We also measure a fourth metric, $variance$, which quantifies how different the explanations for an instance are on different runs of a technique. We measure $variance$ with the same data and setup as for the interpretation technique, but just modify the

random seed. Then, using the similarity distance presented in Section 4.3.2, we calculate the average variance between an instance's explanations over all examples. For each instance we produce three explanations with different random seeds, let them be p_1, p_2, p_3, and then we compute their average distance as $\frac{1}{3}\sum_{i,j\in\{1,2,3\},i\neq j}(1-similarity(p_i,p_j))$. As we use the similarity distance, lower values are better, with the best score being 0.

We use the 10-fold cross-validation method for the four smaller datasets (Glass, Heart (statlog), Boston and Banknote) and 80-20% train-test split for the larger datasets (I. Segmentation, Abalone, Wine Quality and Adult). In both cases, we use test sets to calculate these metrics. In larger datasets, 10-fold cross-validation is not always required, and due to the slow performance of some approaches, such as Anchors, we decided to use one holdout set to perform the evaluation rather than through 10-fold cross-validation. Furthermore, the experiment was unable to finish in the Adult and Wine dataset, where the 20% holdout set was around 9K and 1.3K instances, due to the slow performance of Anchors and DefragTrees, respectively. As a result, we executed the experiment on the first 500 instances. We repeat the same experiment 3 times, with the same random seeds for the whole process, except for the explanation process of each technique, where we change the random seed.

Finally, we propose a novel property called "conclusiveness". A rule-based explanation method is defined as "conclusive" when the antecedent of a rule - explanation - is a definite proof for the consequent. LF by design is "conclusive" because each rule contains every necessary feature to cover at least the minimum number of paths, which will always produce the same outcome. However, this automatically suggests that LF' explanations will be more specific, and they will have low coverage. In the following, we experimentally prove, with at least one example per method, that all the other methods are not "conclusive".

Dataset	LF	DF	GS	LS	CH	AN
Bankn.	$2.74_{\pm 0.0}$	$5.84_{\pm 0.5}$	$2.93_{\pm 0.1}$	**1.48**$_{\pm 0.0}$	$1.73_{\pm 0.0}$	$2.11_{\pm 0.0}$
Heart	$7.97_{\pm 0.0}$	**0.41**$_{\pm 0.1}$	$3.42_{\pm 0.1}$	$3.53_{\pm 0.1}$	$2.34_{\pm 0.0}$	$2.56_{\pm 0.0}$
Adult	$9.89_{\pm 0.1}$	$6.00_{\pm 2.3}$	$4.28_{\pm 0.0}$	**1.62**$_{\pm 0.0}$	$2.79_{\pm 0.0}$	$2.71_{\pm 0.2}$
Glass	$8.23_{\pm 0.0}$	**1.83**$_{\pm 0.2}$	$4.53_{\pm 0.3}$	$3.81_{\pm 0.1}$	$2.69_{\pm 0.0}$	$5.69_{\pm 0.1}$
Segment	$8.51_{\pm 0.0}$	$9.10_{\pm 0.3}$	$3.89_{\pm 0.1}$	**1.71**$_{\pm 0.1}$	$3.73_{\pm 0.0}$	$5.85_{\pm 0.1}$
Abal.mc	$6.80_{\pm 0.0}$	$12.19_{\pm 0.9}$	$3.60_{\pm 0.0}$	$2.33_{\pm 0.0}$	**2.11**$_{\pm 0.0}$	$3.50_{\pm 0.1}$
Abal.r	$4.93_{\pm 0.0}$	$11.20_{\pm 0.5}$	$4.77_{\pm 0.3}$	**3.88**$_{\pm 0.0}$	-	-
Boston	$10.55_{\pm 0.0}$	**0.56**$_{\pm 0.5}$	$4.32_{\pm 0.1}$	$4.09_{\pm 0.0}$	-	-
Wine	$8.43_{\pm 0.0}$	$9.96_{\pm 1.7}$	**2.89**$_{\pm 0.0}$	$3.16_{\pm 0.0}$	-	-
Avg.	$7.56_{\pm 2.3}$	$6.34_{\pm 4.3}$	$3.85_{\pm 0.6}$	$2.85_{\pm 1.0}$	**2.57**$_{\pm 0.6}$	$3.74_{\pm 1.5}$

Table 7: Comparison of techniques in terms of the $rule\_length$. The best performance is denoted in bold

5.4.1 Comparison with evaluation metrics

The first metric to be examined is the $rule\_length$. The results of the comparison are visible in Table 7. In these results, it is clear that LF and DF are in almost every case the algorithms providing rules with a lot of conditions. Specifically, we can see that LF and DF are producing rules of around $7.56^{\pm 2.29}$ and $6.34^{\pm 4.32}$ length by average. The other techniques, CH= $2.565^{\pm 0.63}$, AN= $3.74^{\pm 1.50}$, GS= $3.85^{\pm 0.64}$ and LS= $2.85^{\pm 1.00}$, seem to provide smaller rules. CH and AN are not present in the last three columns because these algorithms are not applicable in regression tasks.

Dataset	LF	DF	GS	LS	CH	AN
Bankn.	$6.5\%_{\pm 0.2}$	**98.6%**$_{\pm 0}$	$16.0\%_{\pm 1.7}$	$50.8\%_{\pm 1.2}$	$56.3\%_{\pm 0.4}$	$30.4\%_{\pm 0.3}$
Heart	$4.2\%_{\pm 0.0}$	**100%**$_{\pm 0}$	$21.0\%_{\pm 0.7}$	$20.4\%_{\pm 0.8}$	$33.9\%_{\pm 0.1}$	$21.1\%_{\pm 0.4}$
Adult	$2.1\%_{\pm 0.0}$	**99.9%**$_{\pm 0}$	$31.0\%_{\pm 0.0}$	$67.7\%_{\pm 0.6}$	$42.0\%_{\pm 0.1}$	$32.0\%_{\pm 0.9}$
Glass	$4.4\%_{\pm 0.0}$	**95.5%**$_{\pm 1}$	$14.3\%_{\pm 1.2}$	$14.9\%_{\pm 0.3}$	$27.8\%_{\pm 0.3}$	$8.4\%_{\pm 0.1}$
Segment	$7.8\%_{\pm 0.1}$	**99.3%**$_{\pm 0}$	$10.0\%_{\pm 0.5}$	$42.7\%_{\pm 0.1}$	$13.9\%_{\pm 0.1}$	$5.9\%_{\pm 0.2}$
Abal.mc	$0.2\%_{\pm 0.0}$	**99.2%**$_{\pm 0}$	$7.7\%_{\pm 0.0}$	$25.0\%_{\pm 0.3}$	$34.9\%_{\pm 0.4}$	$10.6\%_{\pm 0.3}$
Abal.r	$0.1\%_{\pm 0.0}$	**99.4%**$_{\pm 1}$	$0.2\%_{\pm 0.0}$	$0.4\%_{\pm 0.0}$	-	-
Boston	$1.8\%_{\pm 0.0}$	**99.1%**$_{\pm 0}$	$9.8\%_{\pm 0.3}$	$9.3\%_{\pm 0.2}$	-	-
Wine	$0.2\%_{\pm 0.0}$	**99.9%**$_{\pm 0}$	$6.4\%_{\pm 0.0}$	$7.8\%_{\pm 0.2}$	-	-
Avg.	$3.0\%_{\pm 2.7}$	**99.0%**$_{\pm 1}$	$12.9\%_{\pm 8.5}$	$26.6\%_{\pm 21.2}$	$34.8\%_{\pm 12.9}$	$18.1\%_{\pm 10.4}$

Table 8: Comparison of techniques in terms of the $coverage$. The best performance is denoted in bold

$Coverage$ is the second metric we investigate (Table 8). The most interesting findings in this experiment are the extremely high $coverage$ of DF ($99.0\%^{\pm 1.3}$), and the low coverage of LF ($3.00\%^{\pm 2.7}$). The performance of the remaining algorithms is much lower than the DF's performance, but higher than the LF's, CH= $34.8\%^{\pm 12.9}$, AN= $18.1\%^{\pm 10.4}$, GS= $12.9\%^{\pm 8.5}$ and LS= $26.6\%^{\pm 21.2}$.

Dataset	LF	DF	GS	LS	CH	AN
Bankn.	**100%**$_{\pm 0}$	90.8%$_{\pm 1.0}$	99.6%$_{\pm 0.3}$	96.4%$_{\pm 0.1}$	99.3%$_{\pm 0.0}$	55.4%$_{\pm 0.0}$
Heart	**100%**$_{\pm 0}$	88.4%$_{\pm 0.8}$	96.3%$_{\pm 0.3}$	93.7%$_{\pm 0.5}$	97.6%$_{\pm 0.2}$	60.9%$_{\pm 0.5}$
Adult	**100%**$_{\pm 0}$	85.1%$_{\pm 0.6}$	100%$_{\pm 0.0}$	80.4%$_{\pm 0.2}$	99.0%$_{\pm 0.0}$	79.7%$_{\pm 0.2}$
Glass	**100%**$_{\pm 0}$	46.0%$_{\pm 1.1}$	82.5%$_{\pm 2.8}$	80.3%$_{\pm 0.4}$	83.2%$_{\pm 0.2}$	37.0%$_{\pm 0.7}$
Segment	**100%**$_{\pm 0}$	64.1%$_{\pm 5.8}$	98.3%$_{\pm 0.4}$	44.2%$_{\pm 0.7}$	88.2%$_{\pm 0.3}$	14.5%$_{\pm 0.2}$
Abal.mc.	**100%**$_{\pm 0}$	75.7%$_{\pm 0.2}$	90.6%$_{\pm 0.3}$	81.0%$_{\pm 0.2}$	88.0%$_{\pm 0.2}$	03.0%$_{\pm 0.0}$
Abal.r	**0.78**$_{\pm 0}$	0.95$_{\pm 0.0}$	2.27$_{\pm 0.0}$	2.27$_{\pm 0.0}$	-	-
Boston	**5.64**$_{\pm 0}$	5.76$_{\pm 0.2}$	6.03$_{\pm 0.1}$	6.09$_{\pm 0.0}$	-	-
Wine	**0.25**$_{\pm 0}$	0.27$_{\pm 0.0}$	0.54$_{\pm 0.0}$	0.54$_{\pm 0.0}$	-	-
Avg. %	**100%**$_{\pm 0}$	75.0%$_{\pm 16}$	94.5%$_{\pm 6.2}$	79.3%$_{\pm 17}$	92.5%$_{\pm 6.3}$	41.7%$_{\pm 26.7}$
Avg. mae	**2.22**$_{\pm 2.4}$	2.33$_{\pm 2.4}$	2.95$_{\pm 2.3}$	2.97$_{\pm 2.3}$	-	-

Table 9: Comparison of techniques in terms of the $precision$ metric (first 6 rows) and mae (last 3 rows). The best performance is denoted in bold

We evaluated the techniques using the $precision$ metric. As already mentioned, $precision$ is measured differently across the classification and regression tasks. Therefore, in Table 9 the first 6 rows are measured using the actual $precision$ of classification tasks, while in regression we used mae. The very specific rules of LF achieved a high $100\%^{\pm 0.0}$ $precision$ and the lowest $2.224^{\pm 2.43}$ mae in all datasets. In the classification tasks, GS had a high $precision$ performance ($94.5\%^{\pm 6.2}$), as well as CH ($92.5\%^{\pm 6.3}$). LS ($79.3\%^{\pm 17.0}$) and DF ($75.0\%^{\pm 15.8}$) achieved similar to each other results. AN ($41.7\%^{\pm 26.7}$) had the worst performance. For the regression tasks, DF resulted in the second-lowest mae score ($2.233\%^{\pm 2.44}$), while the performance of GS ($2.946\%^{\pm 2.29}$) and LS ($2.968\%^{\pm 2.32}$) was the worst.

Dataset	LF	DF	GS	LS	CH	AN
Bankn.	0.17$_{\pm 0.2}$	0.29$_{\pm 0.2}$.	0.54$_{\pm 0.2}$	0.31$_{\pm 0.3}$	**0.02**$_{\pm 0.1}$	0.038$_{\pm 0.1}$
Heart	0.05$_{\pm 0.1}$	**0.04**$_{\pm 0.1}$	0.22$_{\pm 0.1}$	0.28$_{\pm 0.1}$	0.12$_{\pm 0.1}$	0.15$_{\pm 0.2}$
Adult	**0.01**$_{\pm 0.0}$	0.07$_{\pm 0.0}$	0.03$_{\pm 0.0}$	0.02$_{\pm 0.0}$	0.03$_{\pm 0.0}$	0.03$_{\pm 0.1}$
Glass	**0.07**$_{\pm 0.1}$	0.23$_{\pm 0.1}$	1.35$_{\pm 7.1}$	0.58$_{\pm 0.8}$	0.12$_{\pm 0.1}$	1.50$_{\pm 5.1}$
Segment	**0.02**$_{\pm 0.0}$	0.26$_{\pm 0.1}$	0.20$_{\pm 0.1}$	0.12$_{\pm 0.3}$	0.06$_{\pm 0.1}$	0.26$_{\pm 0.4}$
Abal.mc	**0.05**$_{\pm 0.1}$	0.30$_{\pm 0.1}$	0.08$_{\pm 0.1}$	0.28$_{\pm 0.2}$	0.08$_{\pm 0.1}$	0.14$_{\pm 0.2}$
Abal.r	**0.00**$_{\pm 0.0}$	0.42$_{\pm 0.1}$	0.28$_{\pm 0.1}$	0.28$_{\pm 0.2}$	-	-
Boston	**0.00**$_{\pm 0.0}$	0.09$_{\pm 0.1}$	0.50$_{\pm 0.9}$	0.68$_{\pm 2.7}$	-	-
Wine	**0.00**$_{\pm 0.0}$	0.50$_{\pm 0.1}$	0.06$_{\pm 0.1}$	0.72$_{\pm 2.9}$	-	-
Avg.	**0.04**$_{\pm 0.1}$	0.24$_{\pm 0.2}$	0.36$_{\pm 0.4}$	0.36$_{\pm 0.2}$	0.07$_{\pm 0.0}$	0.35$_{\pm 0.5}$

Table 10: Comparison of techniques in terms of the $variance$. The best performance is denoted in bold

We now compare the different techniques regarding their $variance$. Notice that we measure how distant the explanations provided for an instance are across multiple runs. In Table 10, we can see that LionForests achieved the best $variance$ score, with $0.041^{\pm 0.05}$ across the different runs. CHIRPS also achieved low variance, with an average of $0.070^{\pm 0.04}$, while the other techniques produced explanations with higher $variance$ across the different runs. Specifically, DF achieved $0.242^{\pm 0.15}$, AN $0.354^{\pm 0.52}$, GS $0.361^{\pm 0.39}$, and LS $0.363^{\pm 0.23}$.

Finally, we compare the different techniques regarding their $response\_time$. In Figure 11, we can see that, except from Anchors and CHIRPS, the other approaches can provide explanations in one or two seconds. Specifically, we can see that LF provides explanations in on average $1.45^{\pm 1.21}$ seconds. The other techniques are performing equally well. GS as a global interpretation approach is the fastest, providing immediate responses ($0.00^{\pm 0.00}$). LS achieves $2.63^{\pm 0.77}$ seconds per explanation, DefragTrees $0.55^{\pm 0.39}$, and CHIRPS $2.50^{\pm 2.69}$. However, Anchors is the most timely technique, providing explanations in approximately $48.03^{\pm 50.18}$.

Dataset	LF	DF	GS	LS	CH	AN
Bankn.	$1.04_{\pm 0.0}$	$0.28_{\pm 0.0}$	$\mathbf{0}_{\pm 0.0}$	$2.98_{\pm 0.0}$	$0.33_{\pm 0.0}$	$4.19_{\pm 0.2}$
Heart	$1.20_{\pm 0.0}$	$0.15_{\pm 0.0}$	$\mathbf{0}_{\pm 0.0}$	$2.97_{\pm 0.0}$	$0.40_{\pm 0.0}$	$13.20_{\pm 0.3}$
Adult	$1.48_{\pm 0.0}$	$0.49_{\pm 0.0}$	$\mathbf{0}_{\pm 0.0}$	$3.59_{\pm 0.0}$	$6.86_{\pm 0.1}$	$26.62_{\pm 1.1}$
Glass	$4.69_{\pm 0.0}$	$0.37_{\pm 0.0}$	$\mathbf{0}_{\pm 0.0}$	$2.17_{\pm 0.0}$	$0.56_{\pm 0.0}$	$90.31_{\pm 0.7}$
Segment	$1.33_{\pm 0.0}$	$0.65_{\pm 0.0}$	$\mathbf{0}_{\pm 0.0}$	$2.31_{\pm 0.0}$	$1.20_{\pm 0.0}$	$142.74_{\pm 7.9}$
Abal.mc	$0.34_{\pm 0.0}$	$0.14_{\pm 0.0}$	$\mathbf{0}_{\pm 0.0}$	$2.11_{\pm 0.0}$	$5.62_{\pm 0.1}$	$11.08_{\pm 0.6}$
Abal.r	$1.23_{\pm 0.0}$	$1.33_{\pm 0.1}$	$\mathbf{0}_{\pm 0.0}$	$3.33_{\pm 0.0}$	-	-
Boston	$1.31_{\pm 0.0}$	$1.09_{\pm 0.0}$	$\mathbf{0}_{\pm 0.0}$	$3.24_{\pm 0.0}$	-	-
Wine	$0.39_{\pm 0.0}$	$0.40_{\pm 0.0}$	$\mathbf{0}_{\pm 0.0}$	$0.99_{\pm 0.0}$	-	-
Avg.	$1.45_{\pm 1.2}$	$0.55_{\pm 0.4}$	$\mathbf{0}_{\pm 0.0}$	$2.63_{\pm 0.8}$	$2.50_{\pm 2.7}$	$48.02_{\pm 51.2}$

Table 11: Comparison of techniques in terms of the $response\_time$. The best performance is denoted in bold

5.4.2 Conclusiveness property investigation

Even if LF does not always have the best performance in terms of the aforementioned metrics, it can provide "conclusive" interpretation rules, making it trustworthy and competitive. This occurs because LF always retains at least the number of paths that will always produce the same result for a particular instance, and every feature (and its corresponding range) from those paths is included in the final rule (see Section 4.1 and 4.2). In fact, in this section, we are manually demonstrating that all the algorithms under consideration are not conclusive, by providing at least one example for each one of them.

We will use the Banknote dataset for this qualitative assessment. We retain 10 of the 1372 instances for manual inspection, and the rest are used to train the RF model. After normalisation, the distributions of Banknote's 4 features $F = [Variance, Skew, Curtosis, Entropy]$ are between $[-1, 1]$. Then, we choose a random instance among the 10, $x_1 = [0.53, -0.25, -0.24, 0.53]$, which was predicted to be a 'fake banknote'. The following rules were provided by the examined interpretation techniques:

LF: If $0.36 \leq Variance \leq 1$ and $-0.62 \leq Curtosis \leq 1$ then 'fake banknote'

CH: If $0.23 < Variance$ then 'fake banknote'

AN: If $0.42 < Variance$ then 'fake banknote'

LS: If $\{\}$ then 'fake banknote'

LF indicates that the value of the feature $Curtosis$ must be within the range $[-0.62, 1]$, while CH, AN, and LS provide no such requirement. When the instance's value for this feature is set to -1, the prediction is changed from "false banknote" to "real banknote". As a result, the three rules, except LF, were all not conclusive. We proceed looking at another instance $x_2 = [0.52, 0.79, -0.91, -0.35]$. We produced the interpretation rules again after RF predicted the instance as "false banknote".

LF: If $0.43 \leq Variance \leq 1$ and $-0.92 \leq Curtosis \leq -0.89$ then 'fake banknote'

DF: If $0.34 < Skew$ and $-0.97 < Curtosis$ then 'fake banknote'

However, we can see that DF lacks a condition for $Variance$, whereas LF lacks a condition for $Skew$. We changed both of them, and LF's decision not to include $Skew$ was correct because there was no change in the outcome when we altered $Skew$ from 0.79 to either 1 or -1. The prediction, on the other hand, changed when we changed $Variance$ from 0.52 to -1. As a result, DF is not conclusive. Finally, we examined another instance $x_3 = [-0.23, 0.24, -0.73, 0.2]$ to argue that GS is also not conclusive. The RF classified this instance as a "real banknote". We obtained the following two interpretation rules after requesting LF and GS to produce them:

LF: If $-0.23 \leq Variance \leq -0.12$ and $-1 \leq Skew \leq 0.32$ and $-1 \leq Curtosis \leq -0.57$ then 'real banknote'

GS: If $-0.26 < Variance \leq 0.06$ and $0.23 < Skew \leq 0.32$ then 'real banknote'

It is clear that the rules are not identical. GS omitted the $Curtosis$ feature, while LF provides a condition about this feature. We proceeded to modify this feature's values from -0.23 to 1 and the prediction changed to 'fake banknote'. Hence, we can say that GS is also not conclusive.

5.5 Qualitative evaluation

In this final section of the experiments, we will present an example using an instance from the Adult dataset. Apart from the prediction and the interpretation rule, we will show the interpretation visually, and then perform a few manual tests to guarantee that the interpretation rule is conclusive. Initially, we obtain the following prediction and interpretation rule, with and without LF's reduction, for a random instance of Adult:

Original: if *Marital Status* = *Married* and $5119.0 \leq$ *Capital Gain*≤ 5316.5 and $0 \leq$ *Capital Loss*≤ 1782.5 and $33.5 \leq$ *Age*≤ 49.5 and $38.5 \leq$ *Hours Per Week*≤ 40.5 and *Occupation* = *Exec Managerial* and $108326.999 \leq$ *fnlwgt*≤ 379670.501 and *Education* = *Bachelors* and *Sex* = *Male* and *Workclass* = *Private* and *Native Country* = *United Stated* then $Income > 50K$

LF: if *Marital Status* = *Married* and $5119.0 \leq$ *Capital Gain*≤ 5316.5 and $31.5 \leq$ *Age*≤ 49.5 and $108326.999 \leq$ *fnlwgt*≤ 379670.501 then $Income > 50K$

Based on the two explanations, we can draw the following conclusions. The capacity of LF to reduce the rules is obvious. The new interpretation is 63% shorter since it includes only four conditions rather than eleven. The ranges of the features *Age* and *fnlwgt* are also broader in the LF's rule, thus the rule is less sensitive to fluctuations. Finally, we can see that LF reduced nearly all of the categorical features, which is very positive. However, alternative categorical values are presented to the end user, either in the rule or through visualisation. To avoid having cluttered rules, we automatically hide the alternative values from the rule in this example.

We will analyse the features *Native Country* and *Age*, which are categorical and numerical, respectively, using the visualisation offered by LF. Someone may notice that the feature *Native Country* does not appear in this interpretation rule. This means that the value for *Native Country* in this case, *United States*, has no effect on the prediction and has been diminished in this instance's interpretation. Please notice that we either provide the alternative values as indicated in Section 4.5 or we use the LF visualisation tool. The second approach is being pursued in order to avoid having cluttered rules.

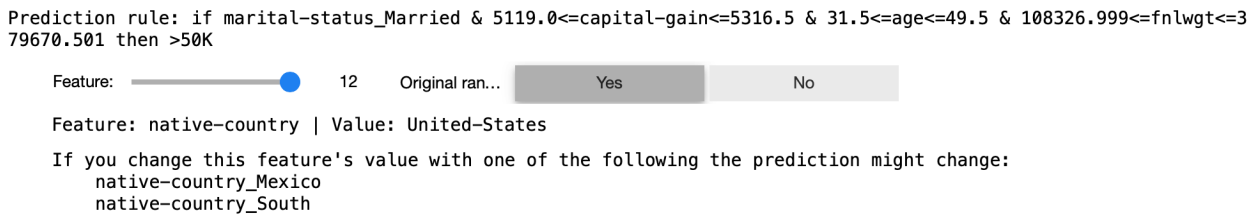

Figure 9: Visualisation of interpretation for feature *Native Country* (original value = *United States*)

As the visualisation tool in Figure 9 indicates, the only potential values that can influence the prediction are *Mexico* and *South*. This in the rule could appear as *Native Country* $\neq$ [*Mexico*, *South*], if the visualisation tool wasn't available. Then, if we change the value of *Native Country* to *Greece*, *United Kingdom*, *Japan* and another 36 countries not appearing in the list of values which may affect the prediction, the outcome indeed remains the same. Our next modification concerns the *Age* feature, as presented in Figure 10. Since *Age* is a numerical feature, the distribution of this feature among the training data, as well as the range of the allowed values of this feature for the examined instance, are presented in a plot.

This plot firstly informs us about how the ranges of the feature's value were extended. The red vertical line reflects the instance's value for this feature (42), while the blue and green lines reflect the initial rule's and reduced rule's ranges, respectively (blue and green right lines are overlapping). We then considered modifying the *Age* with two values within the range, 32 and 48, and indeed, the classification result remained the same as indicated by the extracted rule.

6 Discussion

We have used 8 different datasets from 3 different learning tasks in the above series of experiments to evaluate different aspects of LF and compare it to other techniques. For the scalability analysis, we also utilised 16 synthetically generated datasets.

Via a sensitivity study, we initially investigated how the LF and RF parameters affect the reduction of features and paths (Section 5.2). The parameters *'max features'*, *'depth'*, and *'estimators'* of the RF appear to be strongly associated with the reduction effect. Regarding the classification tasks, a larger number of *estimators* (500 or more) appears to

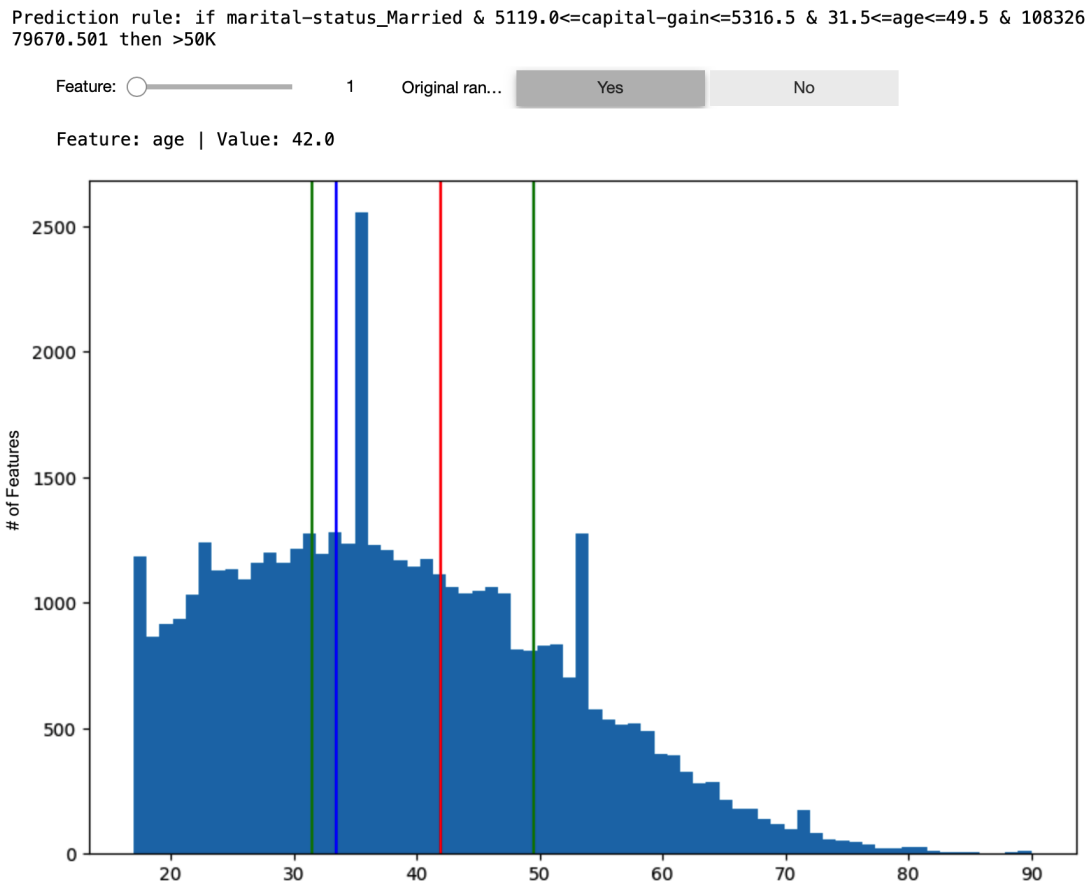

Figure 10: Visualisation of interpretation for feature *Age* (original value $= 42$)

favour feature and path reduction. This is most likely since there are more paths to choose from during the reduction phase. We also notice that LF increases feature reduction when the *depth* of the trees in RF is greater than 5. We believe this is because longer paths will have more in common features with each other, as opposed to very short paths, which may have fewer in common. We could have tested larger *depth* values, but while the computational cost would increase exponentially, we would notice the same trend, namely that higher depth leads to increased feature reduction. About the *'max features'*, it is noticed that using 75% of the features in RF's trees, helps LF to reduce better the features of the final rules. Again, this is happening because the reduction techniques will deal with paths with a lot of similarities in terms of features, as every tree will be allowed to use 75% of the features of the data. However, in regression tasks, we observe the same phenomenon about the *'estimators'*, but not the same for *'max features'* and *'depth'*, where values like *'sqrt'* and 1, respectively, are helping LF to perform better.

For the LF's parameters, we discovered that reduction through AR is required for high feature reduction in both binary and multi-class classification scenarios. More specifically, it is necessary to use reduction through AR to achieve high feature reduction. When reduction via CL and RS is paired with reduction via AR, the feature reduction can be slightly increased. Path reduction, on the other hand, always requires RS, whereas reduction via CL and AR does not boost the reduction. We should mention here that the CL reduction technique is the most time-consuming and increases the overall response time. As a result, for classification tasks, it is best to always utilise reduction via AR and RS to obtain high feature and path reduction. When there are no time constraints, we advise using all the techniques, AR, CL, and RS, to get the greatest possible feature reduction.

We can notice from the sensitivity analysis of the regression tasks that AR+RS reduction and DSi are the most promising reduction techniques. We also investigated the impact of the $local\_error$ parameter, which confirmed the hypothesis that higher $local\_error$ would lead to higher feature and path reduction.

After the sensitivity analysis, we proceeded to the time and scalability analysis in Section 5.3. Those experiments proved how much the LF response time was improved in comparison to the preliminary version. Two core procedures were optimised, and the result was a 93% to 95% faster run-time on the first procedure, and a 94% to 96% on the

second. This improvement renders the LF applicable even in online scenarios where immediate interpretations must be provided to users in real-time. Furthermore, the scalability analysis demonstrates that LF can be applied to large datasets and provides responsive explanations. In binary and multi-class experiments, the only visible limitation appears to be the slow performance of LF in RF models with *depths* of 10 or greater, especially when combined with a large number of *estimators*, 500 or more. Nonetheless, because the majority of available tabular datasets have fewer than 1K features, LF will be able to perform remarkably well in terms of time.

We then compared LF to SOTA algorithms using well-known metrics and a custom property. In terms of $rule\_length$ and $coverage$ the other techniques outperformed LF, but it did achieve perfect $precision$ in the classification experiments. The low performance of LF with respect to $coverage$ is due to the extremely specific rules offered by LF. At the same time, the rules' specificity renders them larger than the rules of the opposing algorithms. Except for the GS, LF outperforms all other techniques in terms of $response\_time$ performance. However, LF also has the conclusiveness property, which renders it reliable and trustworthy, and by examining the other algorithms, we proved that none of them has this property.

Regarding the $rule\_length$, even though LF provided larger rules than the other approaches, we can notice two points. First, it was still able to provide rules with fewer features than the maximum length (based on the size of the feature set) for each dataset. Second, the fact that LF provided larger rules than the other approaches is not necessarily a negative fact, especially when relevant researchers suggest that very small rules are not preferred over lengthier, more informative and expressive rules [19]. Nevertheless, despite the superior performance of the other algorithms in the metrics favoured in the literature ($coverage$ and $rule\_length$), they lack the "conclusiveness" property, which we believe is necessary for any interpretation algorithm that provides rules to the user towards more transparent, complete, and reliable explanations.

We finally presented a detailed use case from one dataset. We chose the Adult dataset, which facilitates both numerical and categorical features. By taking a random instance we generated both the prediction by the RF model, the interpretation rule from the LF and the visual interface as well. We investigated both the ranges of a numerical value and the alternate values of a categorical feature using the user interface. In comparison to other algorithms that only provide the feature and the category in the rule, this tool allows the user to see alternate categorical values.

7 Conclusion

Random forest is one of the top-performing machine learning algorithms in critical sectors such as health, industry, or retail. At the same time, its uninterpretable nature makes it an inappropriate solution, due to trustworthiness concerns, and even issues related to legal frameworks. Therefore, the need of injecting interpretability to such algorithms is evident. In a preliminary work, we introduced a random forest-specific local-based interpretation technique called LionForests. The interpretations are presented in the form of rules. Each rule is a conclusive set of conditions about the features that affected an instance's prediction. LionForests implements a series of feature and path reduction approaches in order to provide smaller rules containing conditions with broader ranges.

We are refining this methodology in a number of ways in this work. Providing a stable theoretic background, enhancing its core procedures towards timely responses, extending its applicability to a variety of learning tasks, LionForests technique undergoes a series of experiments. We investigated and evaluated how the parameters of a random forest model, as well as the parameters of LionForests, influence feature and path reduction in these experiments, which were focused on 8 separate datasets. A time analysis backs up our claim that LionForests responds in a timely manner, in contrast to its preliminary version. The scalability analysis with synthetically generated datasets indicates that LF can be used and can provide explanations for larger datasets as well. We compare LionForests using well-known metrics with cutting-edge techniques. However, while the performance of LionForests based on such metrics does not imply that our system is superior, its inherent property known as "conclusiveness" distinguishes it from the other techniques. We also demonstrate that all the other techniques are not conclusive. Eventually, we give a detailed example of how to best exploit the LionForests technique, in a qualitative manner.

Few of our next steps will be to evaluate LionForests in a human-oriented way, exploiting information from relevant works [54]. Another interesting study would be to investigate LF's adaptability to other types of data as well as different tree ensemble techniques such as GradBoost [20, 21], XGBoost [9], and CatBoost [43]. In addition, we can explore the transformation of the conclusiveness property to an evaluation metric. Finally, we can look into how additional information accompanying an explanation, such as precision and coverage statistics, might be useful to the end user.

Acknowledgements

This paper is supported by the European Union's Horizon 2020 research and innovation programme under grant agreement No 825619. AI4EU Project[5].

References

[1] Adadi, A., Berrada, M.: Peeking inside the black-box: A survey on explainable artificial intelligence (xai). IEEE Access **6**, 52138–52160 (2018)

[2] Agrawal, R., Imieliński, T., Swami, A.: Mining association rules between sets of items in large databases. In: Acm sigmod record, vol. 22, pp. 207–216. ACM (1993)

[3] Agrawal, R., Srikant, R., et al.: Fast algorithms for mining association rules. In: Proc. 20th int. conf. very large data bases, VLDB, vol. 1215, pp. 487–499 (1994)

[4] Bodria, F., Giannotti, F., Guidotti, R., Naretto, F., Pedreschi, D., Rinzivillo, S.: Benchmarking and survey of explanation methods for black box models. arXiv preprint arXiv:2102.13076 (2021)

[5] Breiman, L.: Random forests. Machine Learning **45**(1), 5–32 (2001). DOI 10.1023/A:1010933404324. URL https://doi.org/10.1023/A:1010933404324

[6] Breiman, L., Friedman, J.H., Olshen, R.A., Stone, C.J.: Classification and regression trees. belmont, ca: Wadsworth. International Group **432**, 151–166 (1984)

[7] Bucilua, C., Caruana, R., Niculescu-Mizil, A.: Model compression. In: Proceedings of the 12th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, KDD '06, p. 535–541. Association for Computing Machinery, New York, NY, USA (2006). DOI 10.1145/1150402.1150464. URL https://doi.org/10.1145/1150402.1150464

[8] Chen, A.: Ibm's watson gave unsafe recommendations for treating cancer. https://cutt.ly/keHQDma (2018). Accessed: 2019-11-18

[9] Chen, T., Guestrin, C.: Xgboost: A scalable tree boosting system. In: Proceedings of the 22nd acm sigkdd international conference on knowledge discovery and data mining, pp. 785–794. ACM (2016)

[10] Clark, D., Schreter, Z., Adams, A., Williamson, R.C., Burkitt, A., Bartlett, P.: A quantitative comparison of dystal and backpropagation, australian conference; 7th, neural networks. In: Neural networks, PROCEEDINGS OF THE SEVENTH AUSTRALIAN CONFERENCE ON NEURAL NETWORKS, Australian conference; 7th, Neural networks, pp. 132–137. Australian National University; (1996). URL https://www.tib.eu/de/suchen/id/BLCP%3ACN016972815

[11] Cole, S.: This trippy t-shirt makes you invisible to ai. https://cutt.ly/FeHQHAa (2019). Accessed: 2019-11-18

[12] Cortez, P., Cerdeira, A., Almeida, F., Matos, T., Reis, J.: Modeling wine preferences by data mining from physicochemical properties. Decision support systems **47**(4), 547–553 (2009)

[13] Deng, H.: Interpreting tree ensembles with intrees. International Journal of Data Science and Analytics **7**(4), 277–287 (2019)

[14] Domingos, P.: Knowledge discovery via multiple models. Intelligent Data Analysis **2**(1-4), 187–202 (1998)

[15] Došilović, F.K., Brčić, M., Hlupić, N.: Explainable artificial intelligence: A survey. In: 2018 41st International convention on information and communication technology, electronics and microelectronics (MIPRO), pp. 0210–0215. IEEE (2018)

[16] Dua, D., Graff, C.: UCI machine learning repository (2017). URL http://archive.ics.uci.edu/ml

[17] Fernández-Delgado, M., Cernadas, E., Barro, S., Amorim, D.: Do we need hundreds of classifiers to solve real world classification problems? Journal of Machine Learning Research **15**(90), 3133–3181 (2014). URL http://jmlr.org/papers/v15/delgado14a.html

[18] Firth, N.: Apple card is being investigated over claims it gives women lower credit limits. https://cutt.ly/oeGYCx5 (2019). Accessed: 2019-11-18

[19] Freitas, A.A.: Comprehensible classification models: A position paper. SIGKDD Explor. Newsl. **15**(1), 1–10 (2014). DOI 10.1145/2594473.2594475. URL https://doi.org/10.1145/2594473.2594475

[5] https://www.ai4eu.eu

[20] Friedman, J.H.: Greedy function approximation: a gradient boosting machine. Annals of statistics pp. 1189–1232 (2001)

[21] Friedman, J.H.: Stochastic gradient boosting. Computational statistics & data analysis **38**(4), 367–378 (2002)

[22] Friedman, J.H., Popescu, B.E.: Predictive learning via rule ensembles. The Annals of Applied Statistics **2**(3), 916–954 (2008)

[23] Gries, S.T.: On classification trees and random forests in corpus linguistics: Some words of caution and suggestions for improvement. Corpus Linguistics and Linguistic Theory (2019)

[24] Guidotti, R., Monreale, A., Ruggieri, S., Pedreschi, D., Turini, F., Giannotti, F.: Local rule-based explanations of black box decision systems. CoRR **abs/1805.10820** (2018). URL http://arxiv.org/abs/1805.10820

[25] Han, J., Pei, J., Yin, Y., Mao, R.: Mining frequent patterns without candidate generation: A frequent-pattern tree approach. Data mining and knowledge discovery **8**(1), 53–87 (2004)

[26] Hara, S., Hayashi, K.: Making tree ensembles interpretable: A bayesian model selection approach. In: A. Storkey, F. Perez-Cruz (eds.) Proceedings of the Twenty-First International Conference on Artificial Intelligence and Statistics, *Proceedings of Machine Learning Research*, vol. 84, pp. 77–85. PMLR, Playa Blanca, Lanzarote, Canary Islands (2018). URL http://proceedings.mlr.press/v84/hara18a.html

[27] 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)

[28] Hatwell, J., Gaber, M.M., Azad, R.: gbt-hips: Explaining the classifications of gradient boosted tree ensembles. Applied Sciences **11**(6), 2511 (2021)

[29] Hatwell, J., Gaber, M.M., Azad, R.M.A.: CHIRPS: explaining random forest classification. Artif. Intell. Rev. **53**(8), 5747–5788 (2020). DOI 10.1007/s10462-020-09833-6. URL https://doi.org/10.1007/s10462-020-09833-6

[30] Hinton, G., Vinyals, O., Dean, J.: Distilling the knowledge in a neural network. In: NIPS Deep Learning and Representation Learning Workshop (2015). URL http://arxiv.org/abs/1503.02531

[31] Ioannis, M., Nick, B., Ioannis, V., Grigorios, T.: Lionforests: local interpretation of random forests. In: S. Alessandro, S. Luciano, L. Paul (eds.) First International Workshop on New Foundations for Human-Centered AI (NeHuAI 2020), no. 2659 in CEUR Workshop Proceedings, p. 17–24. Aachen (2020). URL http://ceur-ws.org/Vol-2659/mollas.pdf

[32] Jemima Jebaseeli, T., Venkatesan, R., Ramalakshmi, K.: Fraud detection for credit card transactions using random forest algorithm. In: J.D. Peter, S.L. Fernandes, A.H. Alavi (eds.) Intelligence in Big Data Technologies—Beyond the Hype, pp. 189–197. Springer Singapore, Singapore (2021)

[33] Kaufman, L., Rousseeuw, P.J.: Clustering by means of medoids. statistical data analysis based on the l1 norm. Y. Dodge, Ed pp. 405–416 (1987)

[34] Kohavi, R.: Scaling up the accuracy of naive-bayes classifiers: a decision-tree hybrid. In: Proceedings of the Second International Conference on Knowledge Discovery and Data Mining, p. to appear (1996)

[35] Lundberg, S.M., Erion, G., Chen, H., DeGrave, A., Prutkin, J.M., Nair, B., Katz, R., Himmelfarb, J., Bansal, N., Lee, S.I.: From local explanations to global understanding with explainable AI for trees. Nature Machine Intelligence **2**(1), 56–67 (2020). DOI 10.1038/s42256-019-0138-9. URL https://doi.org/10.1038/s42256-019-0138-9

[36] Lundberg, S.M., Lee, S.I.: A unified approach to interpreting model predictions. In: I. Guyon, U.V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, R. Garnett (eds.) Advances in Neural Information Processing Systems 30, pp. 4765–4774. Curran Associates, Inc. (2017). URL http://papers.nips.cc/paper/7062-a-unified-approach-to-interpreting-model-predictions.pdf

[37] von Luxburg, U.: A tutorial on spectral clustering. Stat. Comput. **17**(4), 395–416 (2007). DOI 10.1007/s11222-007-9033-z. URL https://doi.org/10.1007/s11222-007-9033-z

[38] Maaten, L.v.d., Hinton, G.: Visualizing data using t-sne. Journal of machine learning research **9**(Nov), 2579–2605 (2008)

[39] Meinshausen, N.: Node harvest. The Annals of Applied Statistics pp. 2049–2072 (2010)

[40] Moore, A., Murdock, V., Cai, Y., Jones, K.: Transparent tree ensembles. In: The 41st International ACM SIGIR Conference on Research & Development in Information Retrieval, pp. 1241–1244. ACM (2018)

[41] Nigam, B., Nigam, A., Dalal, P.: Comparative study of top 10 algorithms for association rule mining. International Journal of Computer Sciences and Engineering **5**(8) (2017)

[42] Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., Vanderplas, J., Passos, A., Cournapeau, D., Brucher, M., Perrot, M., Duchesnay, E.: Scikit-learn: Machine learning in Python. Journal of Machine Learning Research **12**, 2825–2830 (2011)

[43] Prokhorenkova, L.O., Gusev, G., Vorobev, A., Dorogush, A.V., Gulin, A.: Catboost: unbiased boosting with categorical features. In: S. Bengio, H.M. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, R. Garnett (eds.) Advances in Neural Information Processing Systems 31: Annual Conference on Neural Information Processing Systems 2018, NeurIPS 2018, December 3-8, 2018, Montréal, Canada, pp. 6639–6649 (2018). URL https://proceedings.neurips.cc/paper/2018/hash/14491b756b3a51daac41c24863285549-Abstract.html

[44] Ravikumar, S., Muralidharan, V., Ramesh, P., Pandian, C.: Fault diagnosis of self-aligning conveyor idler in coal handling belt conveyor system by statistical features using random forest algorithm. In: N. Zhou, S. Hemamalini (eds.) Advances in Smart Grid Technology, pp. 207–219. Springer Singapore, Singapore (2021)

[45] Regulation, G.D.P.: Regulation (eu) 2016/679 of the european parliament and of the council of 27 april 2016 on the protection of natural persons with regard to the processing of personal data and on the free movement of such data, and repealing directive 95/46. Official Journal of the European Union (OJ) **59**(1-88), 294 (2016)

[46] Resende, P.A.A., Drummond, A.C.: A survey of random forest based methods for intrusion detection systems. ACM Comput. Surv. **51**(3) (2018). DOI 10.1145/3178582. URL https://doi.org/10.1145/3178582

[47] 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)

[48] Ribeiro, M.T., Singh, S., Guestrin, C.: Anchors: High-Precision Model-Agnostic Explanations. In: Thirty-Second AAAI Conference on Artificial Intelligence (2018). URL www.aaai.org

[49] Ricciardi, C., Amboni, M., De Santis, C., Ricciardelli, G., Improta, G., Iuppariello, L., D'Addio, G., Barone, P., Cesarelli, M.: Classifying different stages of parkinson's disease through random forests. In: J. Henriques, N. Neves, P. de Carvalho (eds.) XV Mediterranean Conference on Medical and Biological Engineering and Computing – MEDICON 2019, pp. 1155–1162. Springer International Publishing, Cham (2020)

[50] Rudin, C.: Stop explaining black box machine learning models for high stakes decisions and use interpretable models instead. Nature Machine Intelligence **1**(5), 206–215 (2019)

[51] Schubert, E., Gertz, M.: Improving the cluster structure extracted from OPTICS plots. In: R. Gemulla, S.P. Ponzetto, C. Bizer, M. Keuper, H. Stuckenschmidt (eds.) Proceedings of the Conference "Lernen, Wissen, Daten, Analysen", LWDA 2018, Mannheim, Germany, August 22-24, 2018, *CEUR Workshop Proceedings*, vol. 2191, pp. 318–329. CEUR-WS.org (2018). URL http://ceur-ws.org/Vol-2191/paper37.pdf

[52] Simsekler, M.C.E., Qazi, A., Alalami, M.A., Ellahham, S., Ozonoff, A.: Evaluation of patient safety culture using a random forest algorithm. Reliability Engineering & System Safety **204**, 107186 (2020). DOI https://doi.org/10.1016/j.ress.2020.107186. URL https://www.sciencedirect.com/science/article/pii/S0951832020306876

[53] Štrumbelj, E., Kononenko, I.: Explaining prediction models and individual predictions with feature contributions. Knowledge and information systems **41**(3), 647–665 (2014)

[54] van der Waa, J., Nieuwburg, E., Cremers, A., Neerincx, M.: Evaluating xai: A comparison of rule-based and example-based explanations. Artificial Intelligence **291**, 103404 (2021). DOI https://doi.org/10.1016/j.artint.2020.103404. URL https://www.sciencedirect.com/science/article/pii/S0004370220301533

[55] Vens, C., Costa, F.: Random forest based feature induction. In: D.J. Cook, J. Pei, W. Wang, O.R. Zaïane, X. Wu (eds.) 11th IEEE International Conference on Data Mining, ICDM 2011, Vancouver, BC, Canada, December 11-14, 2011, pp. 744–753. IEEE Computer Society (2011). DOI 10.1109/ICDM.2011.121. URL https://doi.org/10.1109/ICDM.2011.121

[56] Yao, S., Wei, M., Yan, L., Wang, C., Dong, X., Liu, F., Xiong, Y.: Prediction of crime hotspots based on spatial factors of random forest. In: 15th International Conference on Computer Science & Education, ICCSE 2020, Delft, The Netherlands, August 18-22, 2020, pp. 811–815. IEEE (2020). DOI 10.1109/ICCSE49874.2020.9201899. URL https://doi.org/10.1109/ICCSE49874.2020.9201899

[57] Yusuf, R., Lawal, Z.: Performance analysis of apriori and fp-growth algorithms (association rule mining). International Journal of Computer Applications in Technology **7**, 279–293 (2016)

[58] Zhang, H., Bi, Y., Jiang, W., Luo, C., Cao, S., Guo, P., Zhang, J.: Application of random forest classifier in loan default forecast. In: X. Sun, J. Wang, E. Bertino (eds.) Artificial Intelligence and Security, pp. 410–420. Springer Singapore, Singapore (2020)

[59] Zhao, X., Wu, Y., Lee, D.L., Cui, W.: iforest: Interpreting random forests via visual analytics. IEEE transactions on visualization and computer graphics **25**(1), 407–416 (2018)

[60] Zhou, Y., Hooker, G.: Interpreting models via single tree approximation. arXiv preprint arXiv:1610.09036 (2016)

Appendix A Deeper sensitivity analysis

In this appendix, we present a deeper sensitivity analysis, as originally presented in Section 5.2.

A.1 Binary Classification

Diving deeper to the sensitivity analysis, Figure 11 presents the FR% for the parameters of the RF while Figure 12 refers to the parameters of LF. The parameter analysis reveals that when the RF's *max features* parameter is set to 75%, the reduction in features in all datasets is higher. With regard to *depth* and *estimators*, LF achieves over 35% FR when the *depth* is greater than or equal to 5 and the *estimators* are 100 or more.

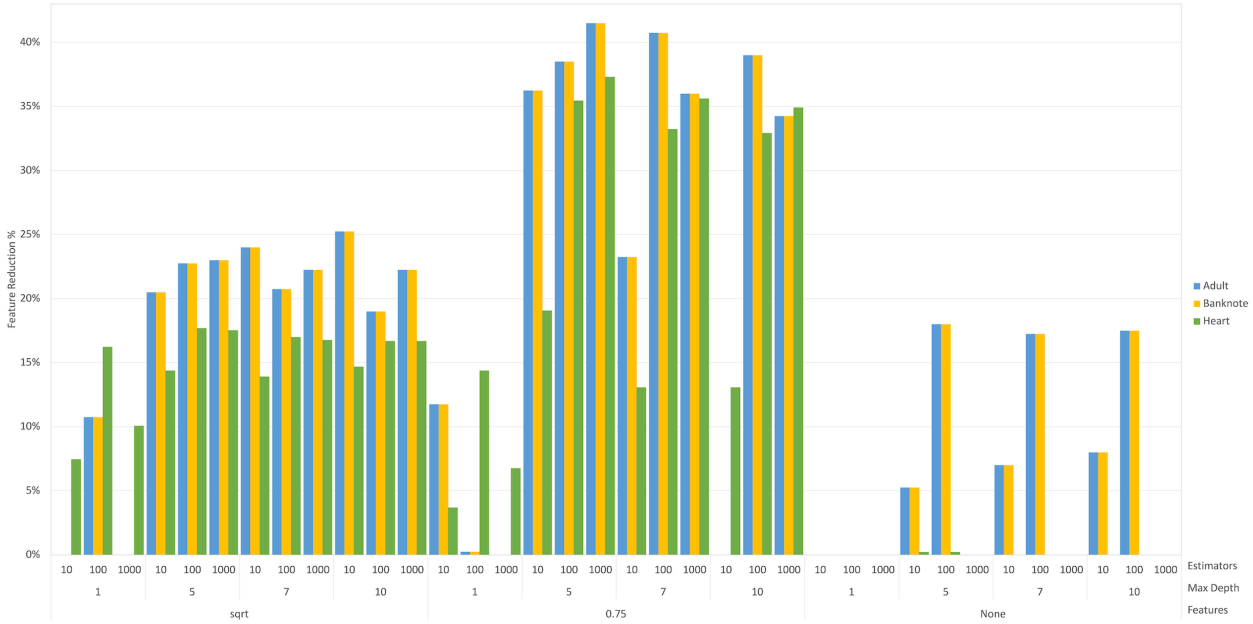

Figure 11: Binary classification: analysis of FR relation to RF's parameters. 'sqrt' and 'log2', as well as *estimators* 500 and 1000, have similar results, and they are grouped.

In Figure 12, we can see how the parameters of LF affect the FR% in these datasets. We observe that the two different AR (1) are performing identically in the FR%. In CR (2), the FR% seemed to diverge for the different algorithms. Specifically, k-medoids and SC manage to reduce the features by 20% or more in all datasets, while OPTICS could not manage to perform any reduction. RS (3) did not achieve any FR in Adult and Banknote, while it achieved low FR% in Heart (Statlog). Among the three approaches, AR seems necessary to achieve high FR%, while the combination of AR (1) with CL (2) seems to increase slightly in all three datasets the FR%.

About the PR analysis, Figure 13 reveals that when the RF's *max features* parameter is set to 'None', the reduction in paths in all datasets is higher. Regarding *depth* and *estimators*, LF achieves over 40% PR when the *depth* is greater or equal to 5 and the *estimators* are 100 or over.

In Figure 14, we can see how the parameters of LF affect the PR% in these datasets. In contrast to FR, for PR, CR (2) and RS (3) are both maximising the PR%. Recall that we cannot reduce more than a quorum in a binary setup, thus these techniques achieving 49% PR are performing optimally. AR (1), on the other hand, cannot seem to be able to optimally reduce paths. Finally, we observe that when combining all three techniques (123), for every parameter setting, the PR% is higher than 40%.

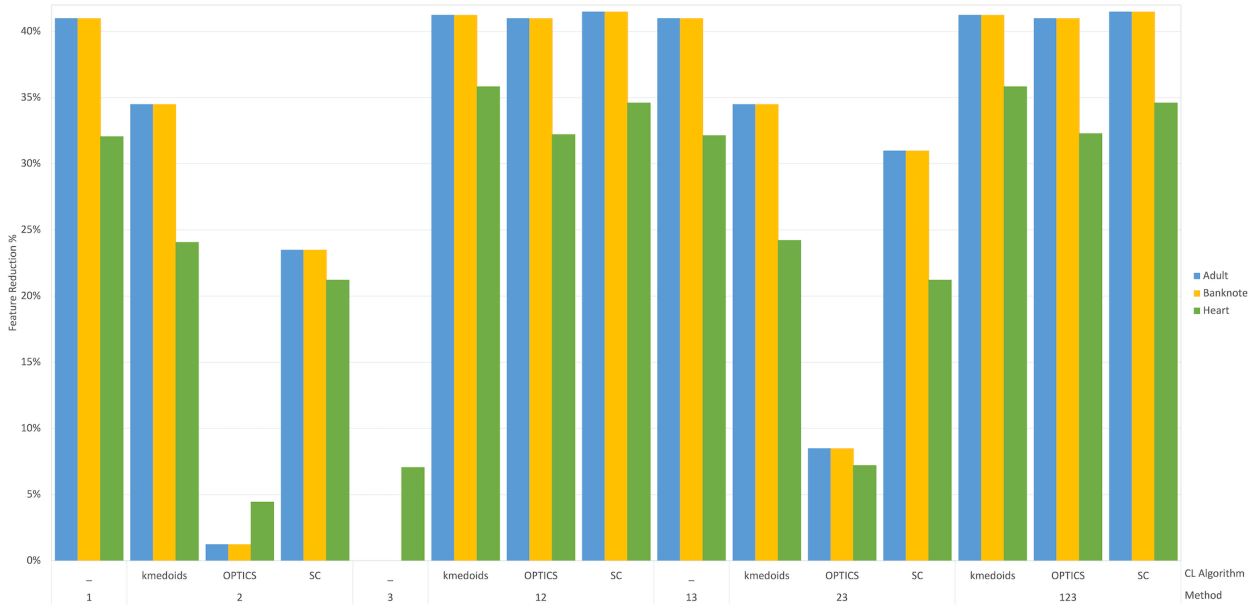

Figure 12: Binary classification: analysis of FR relation to LF's parameters

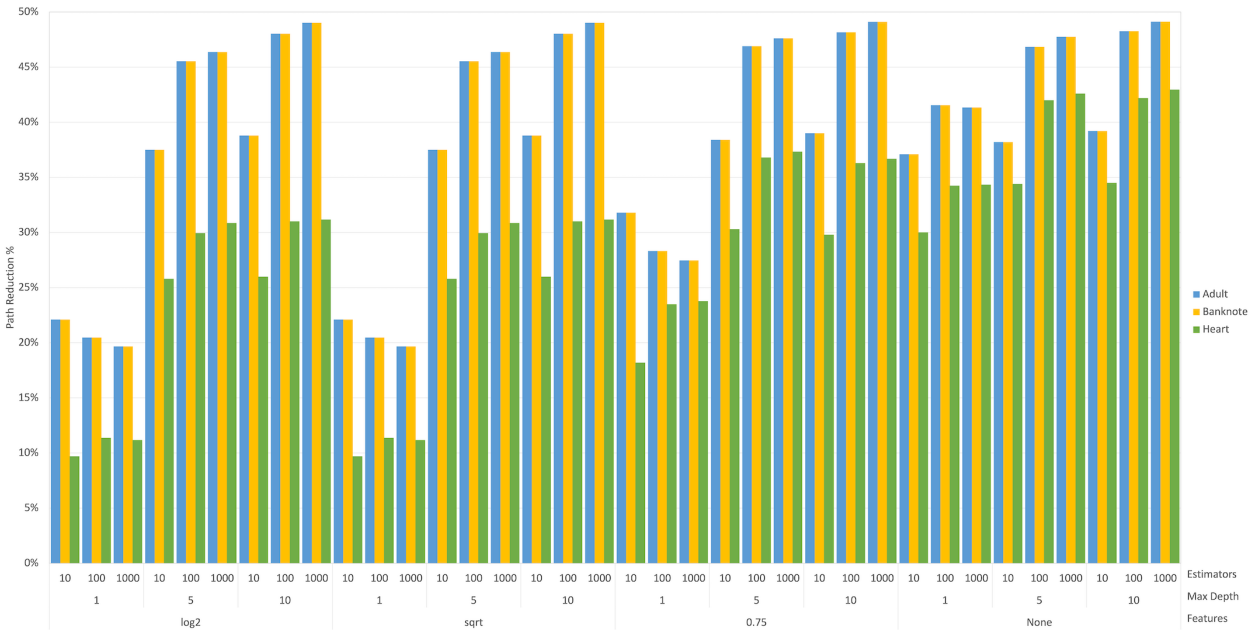

Figure 13: Binary classification: analysis of PR relation to RF's parameters. *Depth* 7 and 10, as well as *estimators* 500 and 1000 are grouped.

A.2 Multi-class classification

The tuning of RF's parameters and their impact to the FR% are visible in Figure 15. The analysis reveals that when the RF's *max features* parameter is set to 75%, the FR in all datasets is higher. LF achieves over 17% FR when the *depth* is greater than or equal to 5 and *estimators* are 100 or over, while it achieves more than 25% and 34% for the individual datasets, Abalone and I. Segmentation, respectively.

Figure 16 presents how FR% is affected based on the different parameters of LF. As observed in the binary classification sensitivity analysis, here as well it is eminent that AR (1) is performing equally in the FR%. CR (2) in the Abalone dataset achieved a higher FR than AR, and when combined (123) with the other techniques, AR and RS, the

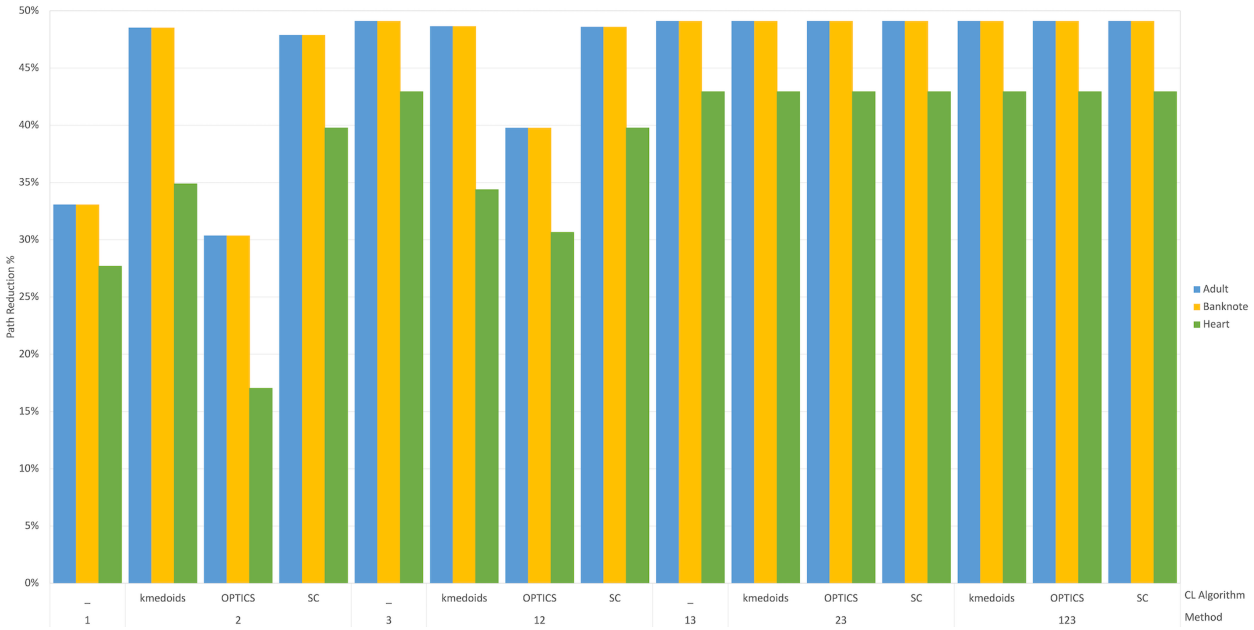

Figure 14: Binary classification: analysis of PR relation to LF's parameters

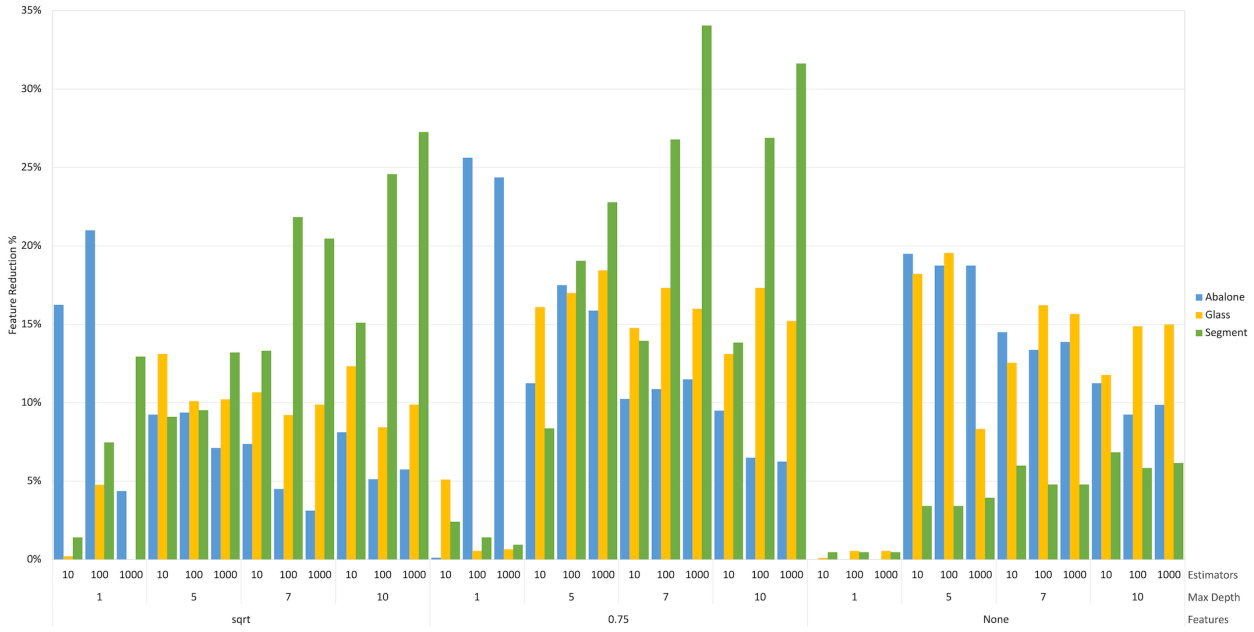

Figure 15: Multi-class classification: analysis FR relation to RF's parameters. 'sqrt' and 'log2', as well as *estimators* 500 and 1000, have similar results, and they are grouped.

FR% is not increasing. The analysis of Glass dataset revealed that rather than the FR achieved by the AR (1), no other method or combination managed to increase the FR%. Finally, on I. Segmentation seemed the combination of AR and CR (12), with specifically SC, to provide the highest FR%. Another interesting point is that the RS (3) managed to reduce the features of the rules in all three datasets, in contrast to the RS's performance on the binary's classification sensitivity analysis.

Through Figure 17, observing the PR while tuning the RF's parameters in these datasets, we can say that the *max features* parameters do not affect the PR%. We can not conclude the same for *depth* and *estimators*, where we need 5 or higher and 100 or more, respectively, to achieve higher PR%. The highest PR% it is achieved when *depth* equals 10 and *estimators* equals 1000.

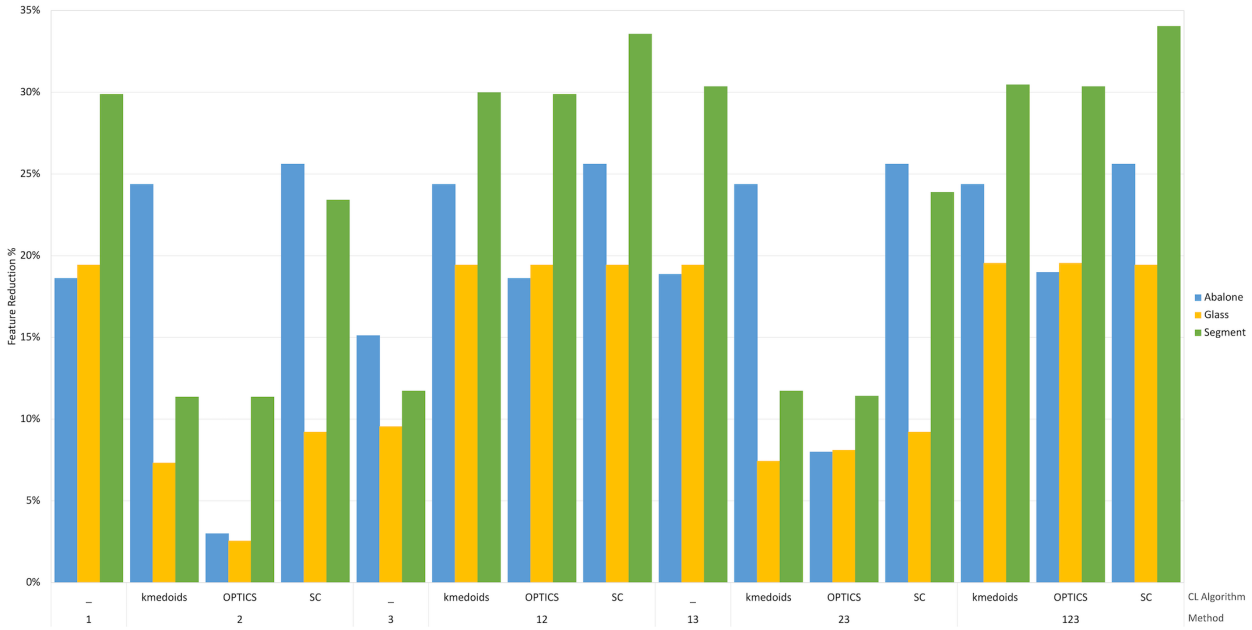

Figure 16: Multi-class classification: analysis of FR relation to LF's parameters

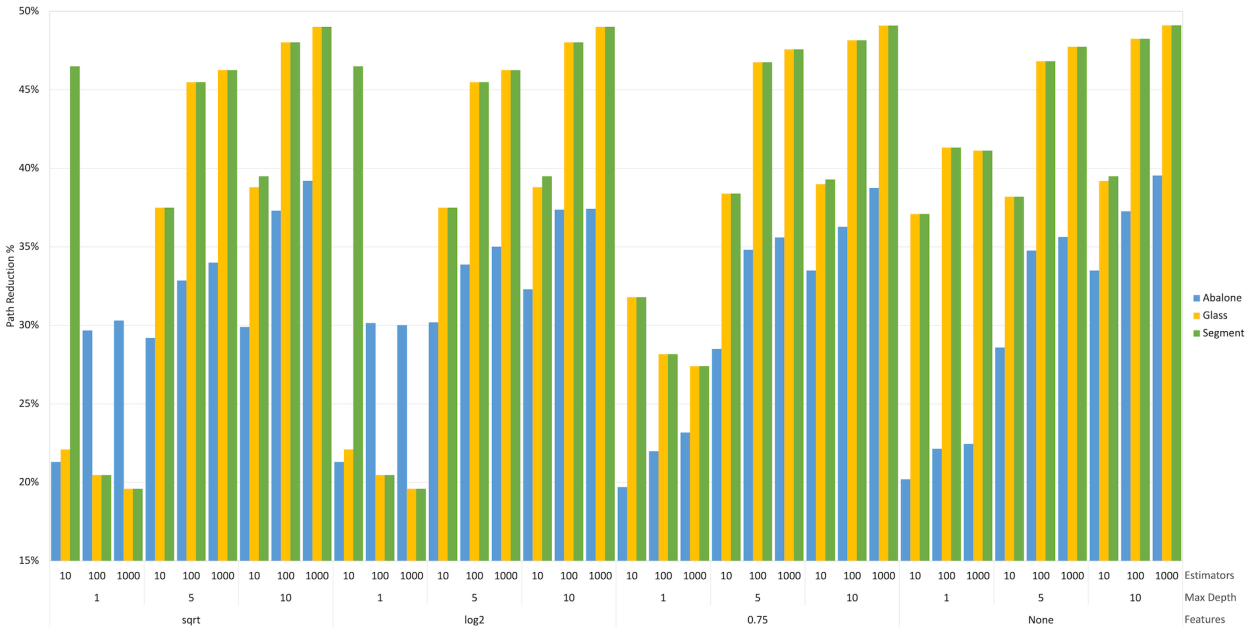

Figure 17: Multi-class classification: analysis of PR relation to RF's parameters. *Depth* 7 and 10, as well as *estimators* 500 and 1000 are grouped.

In Figure 18, we can see how the parameters of LF affect the PR% in these datasets. RS (3) is maxing out the PR%. AR (1) cannot seem to achieve the desirable PR results, while CL (2) is performing well, but not as good as RS (3). Thus, RS or any combination with RS leads to a PR% of 38% or more.

A.3 Regression

In Figure 19, the relation of RF's parameters to the FR% is visible. We can say that the most influencing parameter is *estimators*. When *estimators* are equal or more than 500 and *depth* is either 1 or 5, then the reduction is between 35%

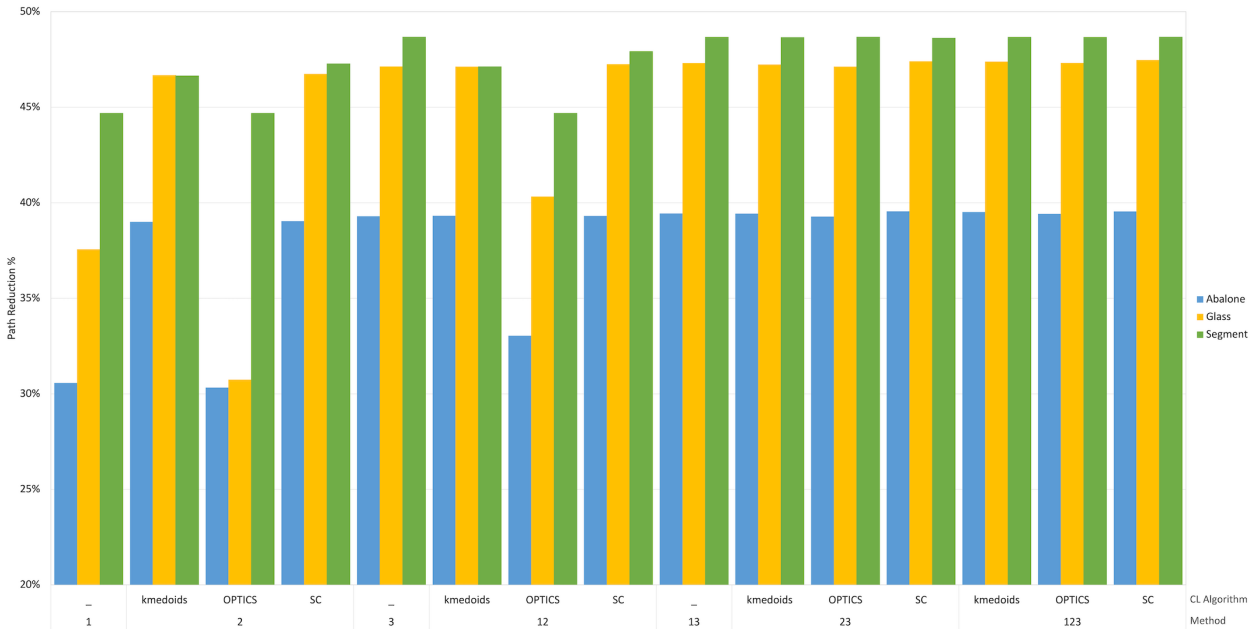

Figure 18: Multi-class classification: analysis of PR relation to LF's parameters

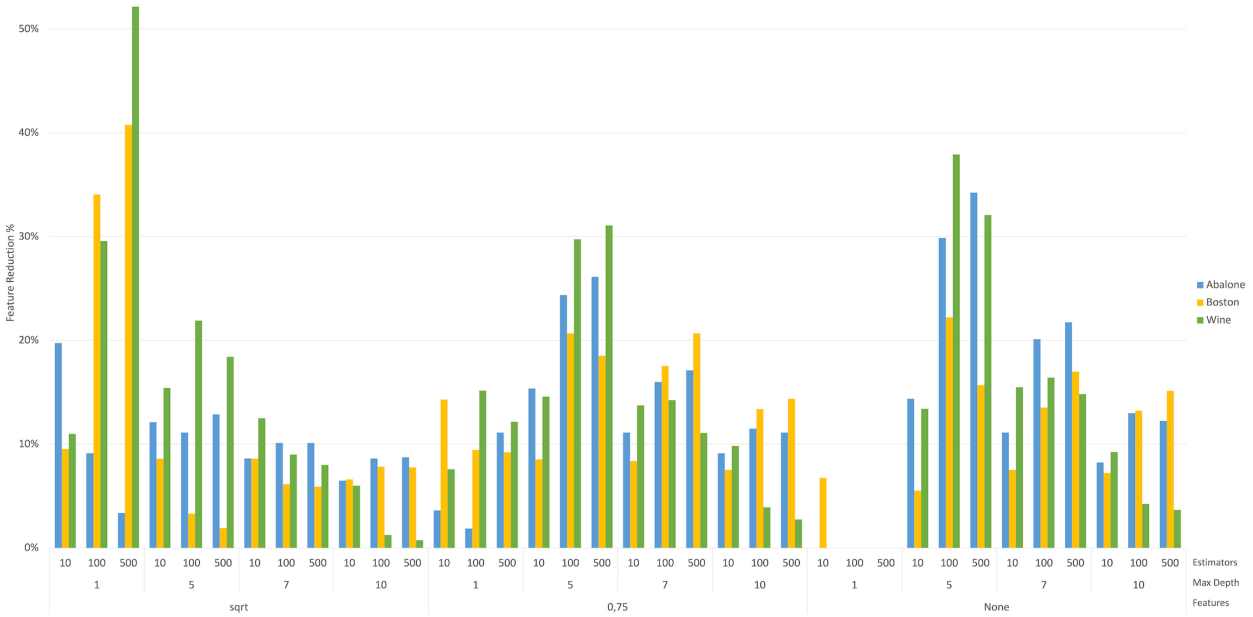

Figure 19: Regression: analysis FR relation to RF's parameters. 'sqrt' and 'log2', as well as *estimators* 500 and 1000, have similar results, and they are grouped.

to 51%. Moreover, for Boston and Wine we observe that when *max features* is set to either 'sqrt' or 'log2', the FR% is higher. On the other hand, higher *max features* values like '0.75' or 'None' seem to favour the FR% for Abalone.

Inspecting how the LF's parameters are affecting the FR%, in Figure 20, we can see that AR+RS method provides better results for Abalone, while DSi for Boston and Wine. However, DSo cannot reach desirable levels of FR% in any case.

The same pattern we identified for the FR% relation to RF's parameters, it is apparent for the relation of PR% with the RF's parameters as well (Figure 22). Setting *estimators* between 500 or 1000 and *depth* to either 1 or 5, the PR is between 50% to 85%. However, *max features* do not affect the PR%.

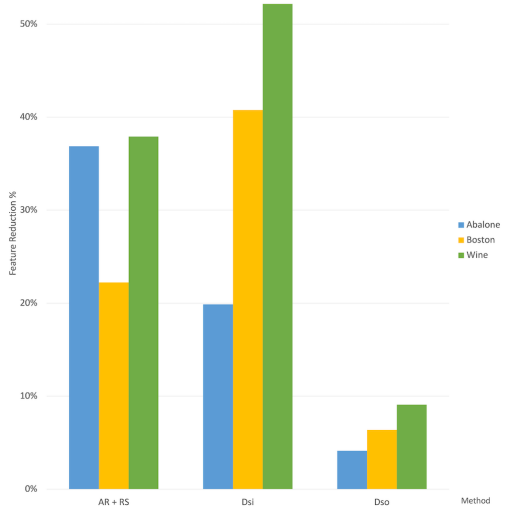

Figure 20: Regression: analysis of FR relation to LF's parameters

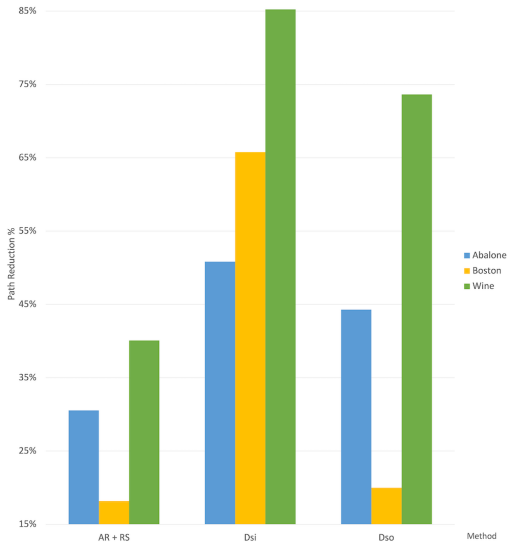

Figure 21: Regression: analysis of PR relation to LF's parameters

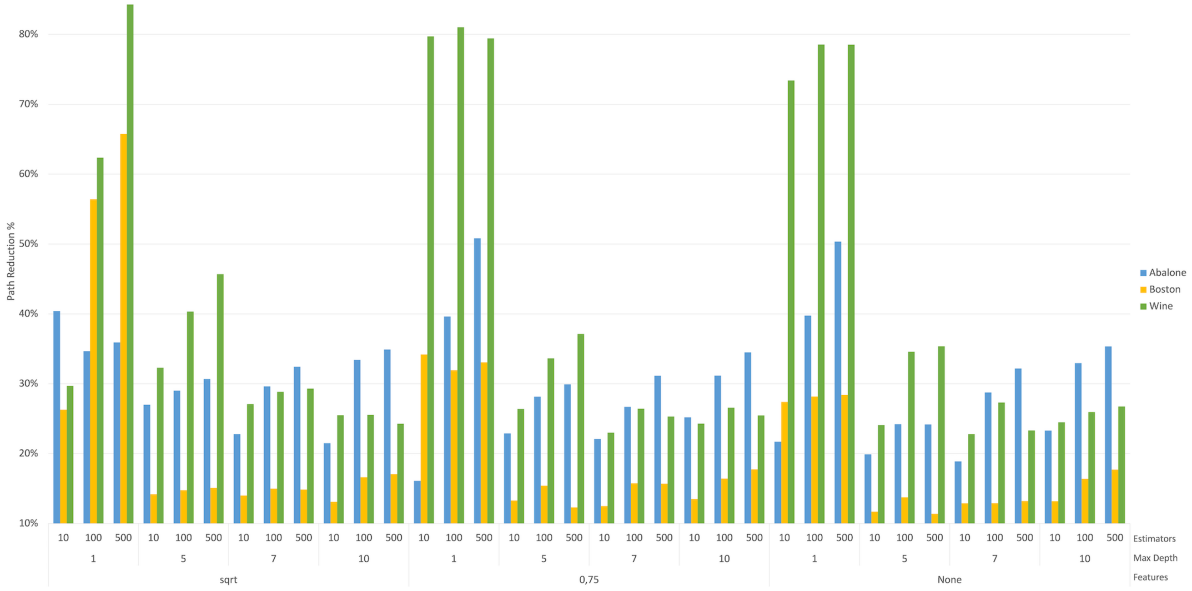

Figure 22: Regression: analysis of PR relation to RF's parameters. 'sqrt' and 'log2', as well as *estimators* 500 and 1000, have similar results, and they are grouped.

In Figure 21, we can see how the parameters of LF affect the PR% in these datasets. We observe the highest PR%, over 50%, with the DSi reduction method of LF. DSo is also better than AR+RS in terms of PR%.

Examining the relation of $local\_error$ with the FR% (Figure 23), we can say that for the Wine dataset we can achieve high FR%, over 50%, with a low $local\_error$ of around 0.36. For the Abalone dataset, we need a $local\_error$ with a value between $[1.1, 1.4]$ in order to achieve approximately 35% of FR. Finally, for Boston in order to achieve FR% higher than 40% we need a $local\_error$ around 2.2.

Finally, about the relation of PR% with the $local\_error$, we observe, in Figure 24, that we acquire higher PR% when we allow higher $local\_error$, in every dataset. In order to let the reader understand better the relation of both the FR% and PR% with the $local\_error$, we present the target variable statistics of each dataset in Table 12. This will help to associate the $local\_error$ with the actual values of the target variable of each dataset.

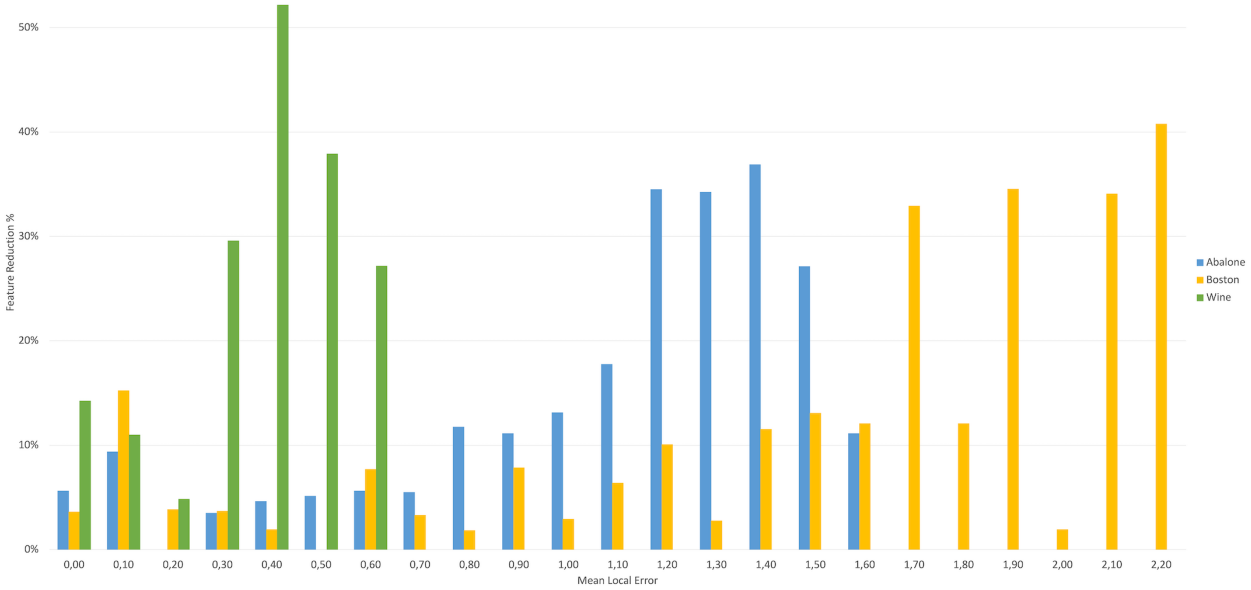

Figure 23: Regression: analysis of FR relation to $local\_error$

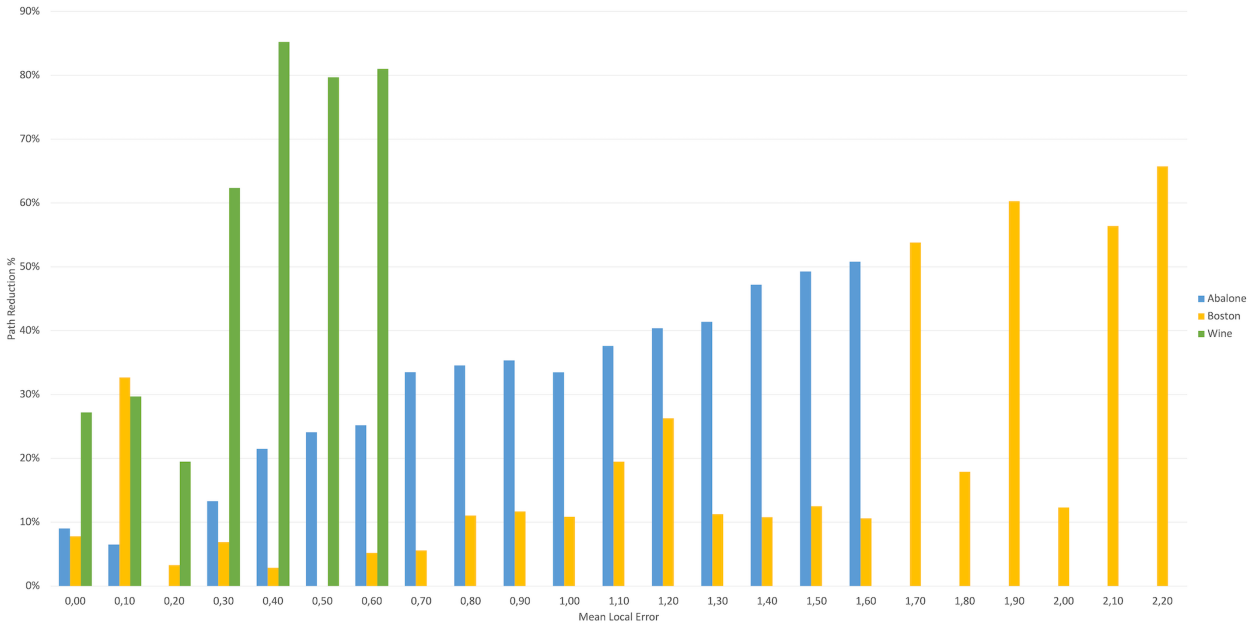

Figure 24: Regression: analysis of PR relation to $local\_error$

Appendix B Deeper analysis of time and scalability analysis

In this appendix, we present a deeper analysis regarding the runtime performance and scalability, as originally presented in Section 5.3.

In Figure 26 we are zooming the y-axis in order to make visible that LF runs approximately between 0.2 to 0.6 seconds per explanation, in contrast to the preliminary version which generates explanations from 0.2 to almost 80 seconds.

In Figure 28 we are zooming the y-axis in order to make visible that the version of LF runs approximately between 0.2 to 11 seconds per explanation, in contrast to the preliminary version which generates explanations from 2 to 128 seconds, and even over 280 in few extreme cases.

	Min	Max	Mean	Std
Abalone	3.0	19.0	9.74	2.86
Boston	5.0	50.0	22.53	9.19
Wine	3.0	9.0	5.82	0.87

Table 12: Statistics of target variable of regression task's datasets

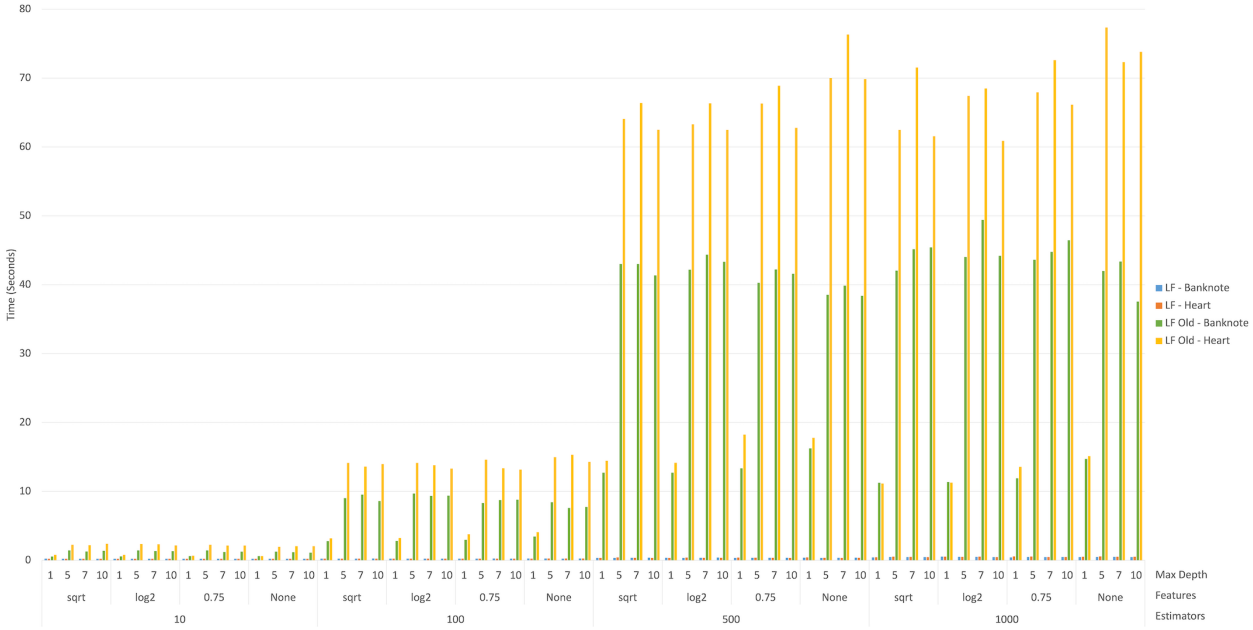

Figure 25: Comparison of preliminary LF and LF on features' ranges generation without reduction (y-axis in seconds)

As it is visible from Figure 29, the worst performance in thee binary setup occurred when we used a dataset with 1000 features, 1000 estimators, and a *depth* of 10, reaching over 1 minute per explanation. An explanation takes 4 seconds in a typical configuration with 1000 features, 500 estimators, and a *depth* of 5. While 100 features, 1000 estimators, and *depth* 2 produce an explanation in half a second.

In the multi-class experiments (Figure 30), the lowest performance was with 1000 features, 1000 estimators, and a *depth* of 10, with either 10 or 100 classes, reaching over 1 minute, actually 64 seconds. In a common configuration with 1000 features, 500 estimators, and a *depth* of 5, an explanation takes 4.5 seconds. An explanation takes 0.8 seconds to generate using 100 features, 1000 estimators, and *depth* 2.

In the regression experiments (Figure 31), the worst performance was with 10 features, 1000 estimators, and a *depth* of 10, reaching almost 1 second. In a common configuration with 1000 features, 500 estimators, and a *depth* of 5, an explanation takes 0.64 seconds. An explanation takes 0.48 seconds to generate using 100 features, 1000 estimators, and *depth* 2.

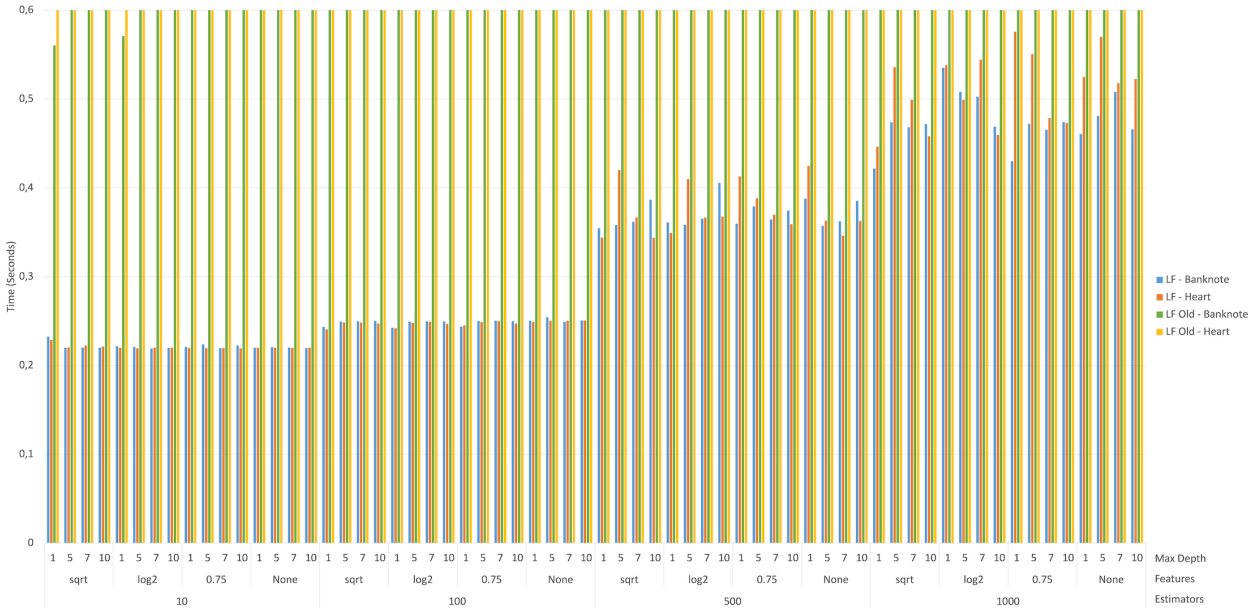

Figure 26: Comparison of preliminary LF and LF on features' ranges generation without reduction (y-axis in second - zoomed)

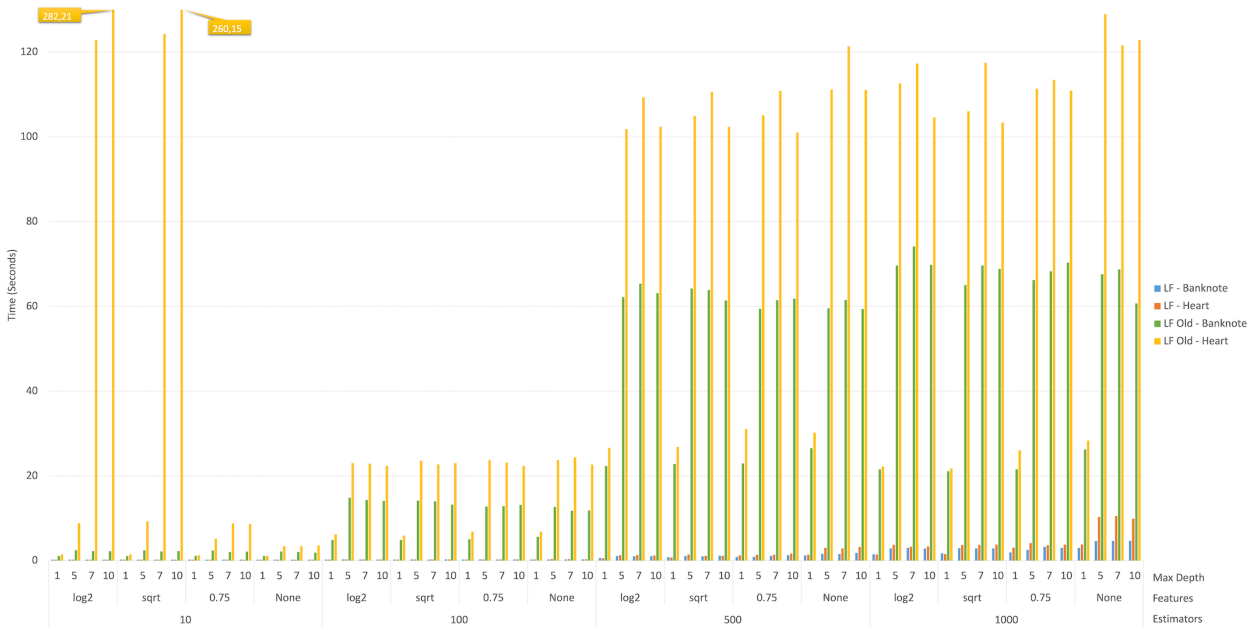

Figure 27: Comparison of preliminary LF and LF on features' ranges generation with reduction (y-axis in second)

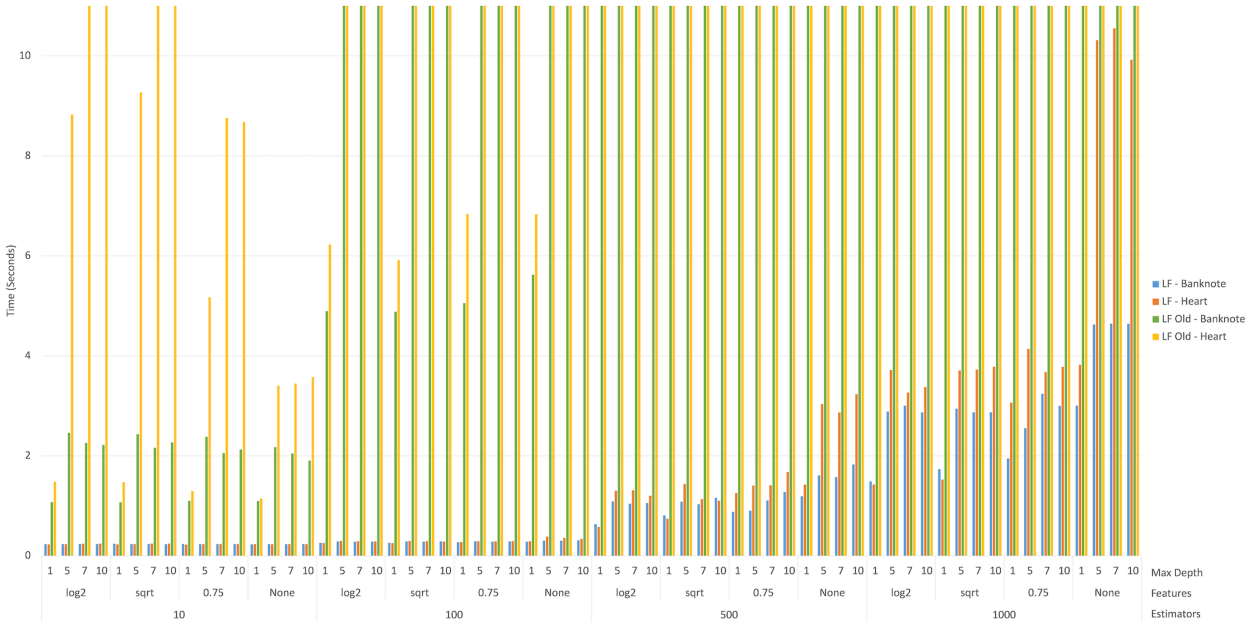

Figure 28: Comparison of preliminary LF and LF on features' ranges generation with reduction (y-axis in second - zoomed)

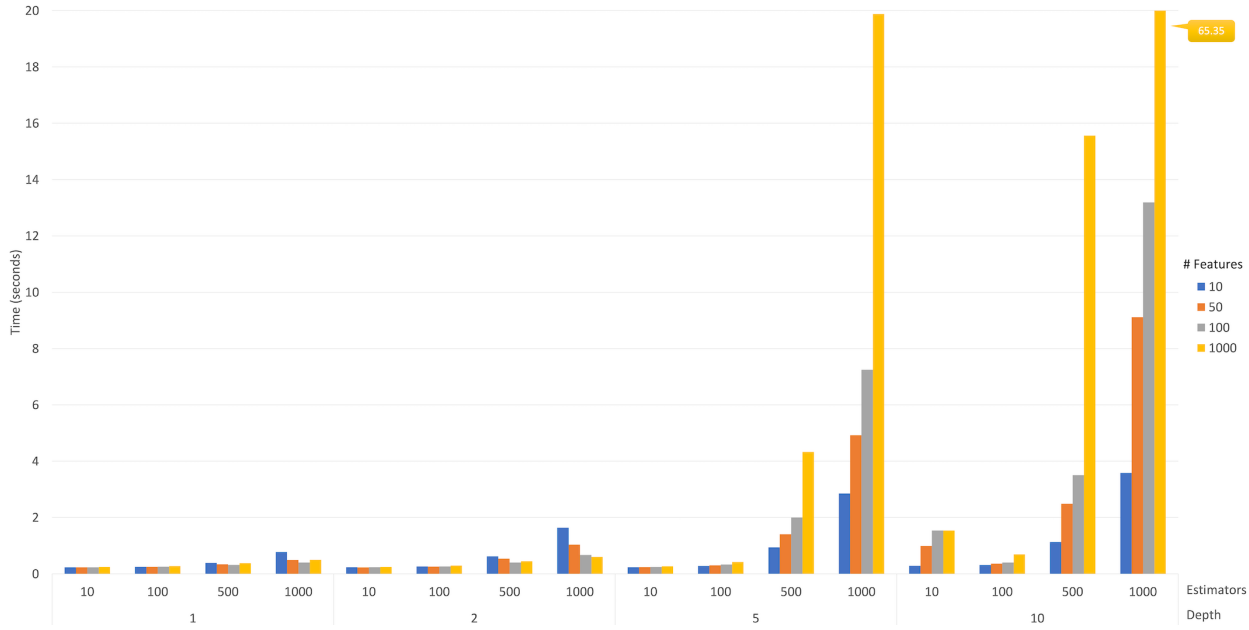

Figure 29: Binary classification: analysis of runtime performance of LF for different number of features and different parameters for RF

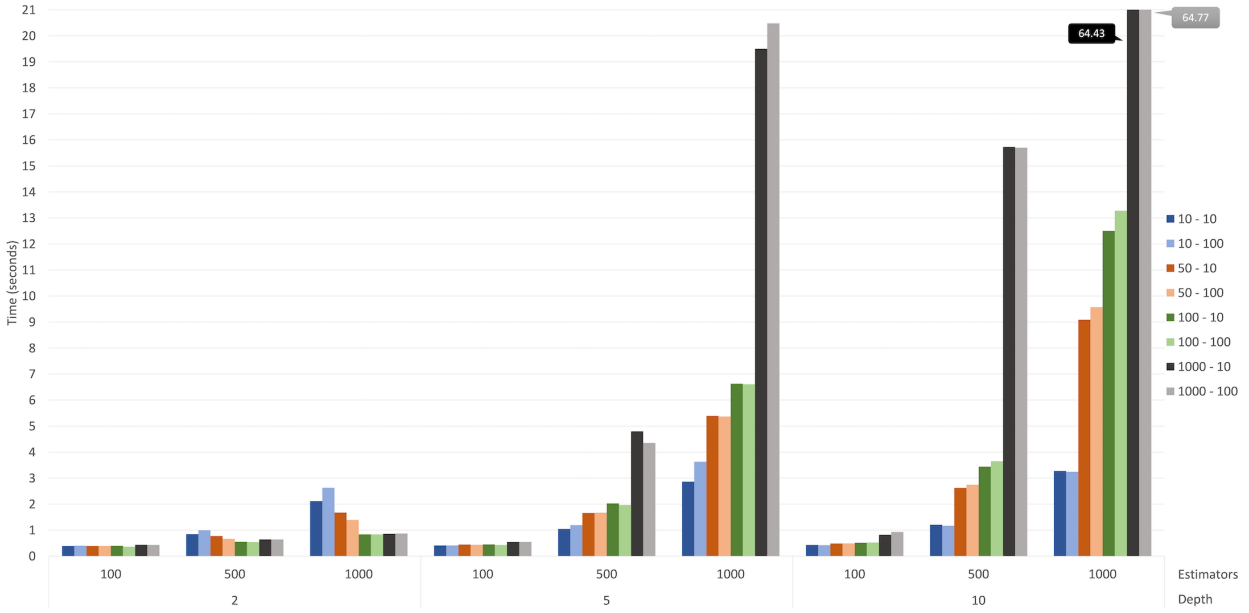

Figure 30: Multi-class classification: analysis of runtime performance of LF for different number of features and different parameters for RF

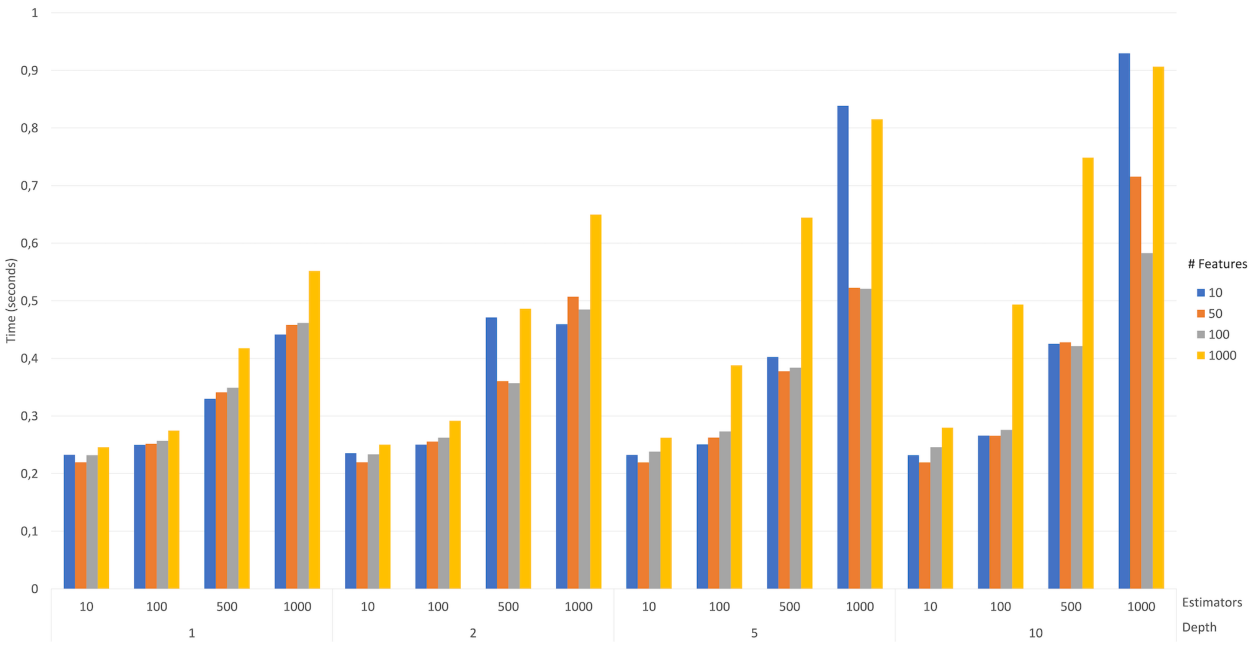

Figure 31: Regression: analysis of runtime performance of LF for different number of features and different parameters for RF