ECAI 2023
K. Gal et al. (Eds.)

doi:10.3233/FAIA230350

Evolutionary Explainable Rule Extraction from (Modal) Random Forests

Michele Ghiotti[a], Federico Manzella[a], Giovanni Pagliarini[a,b], Guido Sciavicco[a] and Ionel Eduard Stan [a,c;*]

[a]ACLAI Laboratory, Department of Mathematics and Computer Science, University of Ferrara
[b]Department of Mathematical, Physical, and Computer Sciences, University of Parma
[c]DBS Group, Faculty of Engineering, Free University of Bozen-Bolzano
ORCiD ID: Michele Ghiotti https://orcid.org/0009-0008-0456-4213,
Federico Manzella https://orcid.org/0000-0002-4944-2163,
Giovanni Pagliarini https://orcid.org/0000-0002-8403-3250,
Guido Sciavicco https://orcid.org/0000-0002-9221-879X, Ionel Eduard Stan
https://orcid.org/0000-0001-9260-102X

Abstract. Symbolic learning is the subfield of machine learning concerned with learning predictive models with knowledge represented in logical form, such as decision tree and decision list models. Ensemble learning methods, such as random forests, are usually deployed to improve the performance of decision trees; unfortunately, interpreting tree ensembles is challenging. In order to deal with unstructured (e.g., temporal or spatial) data, moreover, decision trees and random forests have been recently generalized to the use of modal logics, which are harder to interpret than their propositional counterpart. Recently, a methodology for extracting simple rules from propositional random forests, based on a sequence of optimization steps, was proposed. In this work, we generalize this approach along two directions: from propositional to modal logic and from a sequence of optimization steps to a single multi-objective optimization problem. Even if confined to the temporal domain, our experimental results, based on open-source implementations and public data, show that our method is robust and able to extract small, accurate, and informative decision lists even for complex classification problems.

1 Introduction

Machine learning (ML) focuses on designing algorithms that build predictive models from large amounts of data, and its applicability is widespread in the era of big data. Symbolic learning, a subfield of ML, is concerned with developing symbolic models that represent propositional logic theories from structured data. The investigation of symbolic learning dates back to the early days of artificial intelligence (AI). One of the key advantages of symbolic models is their ability to support reasoning, a cornerstone of traditional AI. This allows humans to interpret and explain their decision-making process, enabling them to address ethical concerns, such as gender and racial biases.

In recent years, however, ML has evolved towards sophisticated black-box models, such as deep neural networks, that significantly challenge human understanding. In fields such as defense, law, and medicine, where the decisions made by AI models have a direct impact on human lives, interpretability is as crucial as accuracy [26]. As a result, eXplainable AI (XAI) has emerged as a field of study focused on developing data-driven models that can reveal and clarify their logic to end-users, including domain experts, developers, managers, and regulatory entities and agencies.

Symbolic learning natively addresses several XAI-related issues, making it a compelling area of research. Numerous literature surveys on XAI and Interpretable ML (IML) have been conducted, offering researchers valuable taxonomies to guide their work (e.g., see [9, 21, 27]). However, despite these efforts, a clear distinction between interpretability and explainability has yet to be established, prompting us to use both terms interchangeably.

At the scope level, models can either be globally interpretable to end-users who can fully comprehend their logic or they can provide local explanations for individual decisions. At the stage level, ante-hoc approaches consider interpretability from the model's inception, while post-hoc methods mimic and explain black-box models; the latter can be either model-specific or model-agnostic, depending on whether they explain specific black-box models or a range of models. At the input level, a crucial factor in developing XAI techniques is understanding the input data type of models (e.g., tabular, time series, and images). Finally, at the output level, various formats, such as logical rules, textual summaries, and graphical visualizations, are essential to cater to the needs, in terms of model explainability, of different audiences.

Symbolic learning represents data and relationships using symbolic structures. Decision trees are a classic example of symbolic learning, where the tree branches typically represent formulas of propositional logic. Despite the effectiveness of decision trees, they suffer from two significant issues. First, their ability to generalize to new data is limited, as they tend to overfit to the training data. Second, they cannot natively learn from unstructured data, such as time series, images, and graphs, due to the limited expressive power of propositional logic. To address the first issue, ensembles of independent decision trees, such as random forests [5], are commonly used to improve the generalization ability of single trees; unfortunately,

* Corresponding Author. Email: ioneleduard.stan@unibz.it.

such ensembles become equivalent to black-box models, as different trees may label an instance in different ways, sometimes making their interpretation very difficult. To overcome the limited expressive power of standard decision trees, it has been recently proposed to replace propositional logic with more expressive, powerful, and task-specific modal logic (e.g., temporal) in decision trees [18].

Modal logic, which extends propositional logic, can be adopted by decision trees as the underlying logical language, in order to capture complex relationships between objects and their properties, ultimately enabling models to learn from unstructured data with no feature-representation embedding steps (i.e., representing unstructured data as structured). Modal decision trees have been successfully used in complex tasks such as time series knowledge extraction, image understanding, and knowledge graph reasoning [13, 32, 36]. In the same spirit of canonical ones, branches of modal decision trees represent modal logic formulas. When modal random forests are used to enhance the performances of single trees, they result in black-box models whose interpretation is even more challenging than in the case of their propositional counterparts because of the added complexity of modal formulas over propositional ones.

The development of XAI methods for interpreting and explaining random forests is paramount, and several solutions have been proposed in the past. Intrinsic interpretable methods include, among others, impurity-based feature importance [5] and tree-based feature importance [44] algorithms, as well as interaction strength and higher-order feature interaction techniques [24]. Post-hoc model-agnostic explainable techniques for local explanations include *Local Interpretable Model-agnostic Explanations* (*LIME*) [40], *Individual Conditional Expectation* (*ICE*) plots [25], and methods for counterfactual explanation [45]. As for those for global explanations, it is worth mentioning *Partial Dependence Plots* (*PDP*) [23], *Accumulated Local Effects* (*ALE*) plots [2], and global surrogate models [15]; *SHapley Additive exPlanations* (*SHAP*) [31], on the other hand, provides unified local and global explanations. Post-hoc model-specific explainable approaches include rule extraction methods from single trees [38], *RuleFit*, frequent pattern mining, bayesian rule extraction, and *Simplified Tree Ensemble Learner (STEL)* [19]. Finally, the numerous contributions to visualization techniques for explaining random forests include decision surfaces and decision boundaries, $t-$*distributed Stochastic Neighbor Embedding (t-SNE)*, and *Uniform Manifold Approximation and Projection (UMAP)*. Among this plethora of techniques and proposals, STEL [19] seems particularly interesting for us, being a recent, open-source, clear approach for extracting rules from random forests that has been successfully applied numerous times in the past few years (e.g., see, among others, [4, 37, 42]).

In this paper, we address the problem of extracting (modal) decision lists from (modal) random forests. To this end, we propose to generalize STEL [19] along two directions: from propositional to modal logic and from a cascade of multiple optimization problems to a single multi-objective optimization one. The resulting algorithm, *ModalSTEL*, fully generalizes the original one, and produces simple (modal) decision lists from (modal) random forests optimizing performance and interpretability. To solve the resulting optimization problem, we propose to use evolutionary computation; in our implementation, we choose *NSGA-II* [17], a well-known multi-objective evolutionary algorithm that has been successfully applied in a variety of situations. Our open-source implementation is written in the Julia programming language as part of a long-term, comprehensive project for symbolic learning for unstructured data.[1]

Our implementation is tested using a group of public datasets in the temporal domain. In particular, we consider the problem of diagnosing COVID-19 from labeled cough examples; the temporal component of such data emerges by interpreting cough sounds as time series, and a symbolic solution to this problem is particularly important due to the medical nature of the problem (e.g, see [32]). Previous symbolic approaches to this problem had already shown the notable performances of modal decision trees and forests; such performance, however, came at the cost of complex models. The aim of our experiments is to show that such models can be simplified with minimal loss in performances to facilitate inspection, ultimately fostering the communication between humans and machines.

2 Preliminaries

Modal logic. Let $\mathcal{P}$ be a (possibly infinite, but countable) set of *proposition letters* (or, simply, *propositions*). The well-formed formulas of *(propositional) modal logic* ($\mathcal{ML}$) are generated by the following grammar:

$$\varphi ::= p \mid \neg\varphi \mid \varphi \wedge \varphi \mid \Diamond\varphi,$$

where $p \in \mathcal{P}$. The remaining propositional abbreviations are derived as usual.

The semantics of $\mathcal{ML}$ is given in terms of Kripke models. A *Kripke model* $\mathfrak{K} = (\mathcal{W}, \mathcal{R}, v)$ consists of a non-empty (possible infinite, but countable) set of *(possible) worlds* $\mathcal{W}$, an *accessibility relation over worlds* $\mathcal{R} \subseteq \mathcal{W} \times \mathcal{W}$, and a *valuation function* $v : \mathcal{W} \to 2^{\mathcal{P}}$, which associates each world w with the set of propositions $v(w) \subseteq \mathcal{P}$ that are true on it. The truth relation $\mathfrak{K}, w \Vdash \varphi$, for a (Kripke) model $\mathfrak{K}$, a world w (in that model) and a formula φ of $\mathcal{ML}$ (to be interpreted on that model), is defined by induction on the complexity of the formula:

$\mathfrak{K}, w \Vdash p$	iff	$p \in v(w)$, for all $p \in \mathcal{P}$;
$\mathfrak{K}, w \Vdash \neg\psi$	iff	$\mathfrak{K}, w \nVdash \psi$;
$\mathfrak{K}, w \Vdash \psi_1 \wedge \psi_2$	iff	$\mathfrak{K}, w \Vdash \psi_1$ and $\mathfrak{K}, w \Vdash \psi_2$;
$\mathfrak{K}, w \Vdash \Diamond\psi$	iff	there exists w' s.t. $w\mathcal{R}w'$ and $\mathfrak{K}, w' \Vdash \psi$.

Blending $\mathcal{ML}$ and ML offers a unique opportunity to advance AI by drawing on the strengths of both fields. By integrating $\mathcal{ML}$'s robust formalism for representing knowledge, beliefs, and uncertainty, ML models can gain a more profound understanding of data. By integrating $\mathcal{ML}$ in both decision trees and random forests, one obtains modal decision trees (known as *ModalCART*) and modal random forests (*ModalRF*) able to learn more expressive theories from unstructured data. The theoretical properties of ModalCART have been studied in [18]. Such an integration is particularly useful when $\mathcal{ML}$ is replaced by tailored temporal/spatial, but still modal, logic, as in [13, 32, 36], for specific learning tasks. The theoretical aspects of such integration, however, are independent from the particular modal logic used for learning; as a matter of fact, modal decision trees can be designed in the same way for any unary modal logic.

In many ML tasks, the modal versions of decision trees and forests show promising results, such as breath and cough recording COVID-19 diagnosis [32] and multivariate time series gas turbine predictive maintenance [3]; specifically, such applications exploit the integration of *Halpern and Shoham's Modal Logic of Allen's Relations ($\mathcal{HS}$)* [28] to learn interval-based patterns from temporal data.

[1] https://github.com/aclai-lab/Sole.jl

(Modal) Decision trees, random forests, and decision lists. Decision trees and forests are popular classification models, associated with their learning algorithms [5, 6, 39]. Decision trees, in particular, are standalone models with a tree-like structure, and make decisions based on specific features. On the other hand, random forests average the results of multiple trees, each constructed from randomly selected subsets of input data and features; this approach reduces the variance, making them more robust and less prone to overfitting than single trees. Decision trees are easy to interpret and visualize, while random forests are more complex, and can handle high-dimensional datasets with many features.

A unifying way to view both standard and modal decision trees is by through the lens of pure decision trees. For our purpose, let $\mathcal{L}$ be a logic, $\Phi(\mathcal{L})$ the set of formulas of logic $\mathcal{L}$, and $\mathbb{Y}$ the label space for a given classification task. Let $T = (\mathsf{V}, \mathsf{E})$ be a full directed binary tree with nodes in V, denoted by $\nu, \nu', \ldots, \nu_1, \nu_2, \ldots$, and edges in $\mathsf{E} \subseteq \mathsf{V} \times \mathsf{V}$. We denote by $root(T)$ the root of T and by $\mathsf{V}^\ell \subseteq \mathsf{V}$ the set of its leaf nodes (or, simply, leaves), denoted by $\ell, \ell', \ldots, \ell_1, \ell_2, \ldots$. Each non-leaf node ν of T has a left child $\swarrow(\nu)$ and a right child $\searrow(\nu)$, and each non-root node ν has a parent $\nwarrow(\nu)$. A path $\pi^T = \nu_1 \rightsquigarrow \nu_h$ in T, with $h \geq 1$, is a finite sequence of h nodes such that $\nu_i = \nwarrow(\nu_{i+1})$, for each $i = 1, \ldots, h-1$. Finally, we denote by π^T_ℓ the unique path $root(T) \rightsquigarrow \ell$, for some leaf $\ell \in \mathsf{V}^\ell$, which is also called branch. We omit the superscript notation, $\cdot^T$, when T is clear from the context.

A *pure decision tree* is a structure defined as:

$$T = (\mathsf{V}, \mathsf{E}, l, e),$$

where (V, E) is a full directed binary tree, $l : \mathsf{V}^\ell \to \mathbb{Y}$ is a leaf-labeling function that assigns to each leaf node in V^ℓ a label from the label space $\mathbb{Y}$, $e : \mathsf{E} \to \Phi(\mathcal{L})$ is an edge-labeling function that assigns to each edge in E a formula from the formulas $\Phi(\mathcal{L})$ such that $e(\nu, \nu') \equiv \neg e(\nu, \nu'')$, for all $(\nu, \nu'), (\nu, \nu'') \in \mathsf{E}$. Moreover, for a path $\pi^T = \nu_1 \rightsquigarrow \nu_h$, the *path-formula* φ^T_π is defined inductively as:

$$\varphi^T_\pi = \begin{cases} \top & \text{if } h = 1; \\ e(\nu_1, \nu_2) \wedge \varphi^T_{\nu_2 \rightsquigarrow \nu_h} & \text{if } h > 1. \end{cases}$$

Pure decision trees are a convenient representation for propositional [6, 39], oblique [33], and modal [18] decision trees. Different types of trees are linked to different techniques and learning algorithms. For example, in the propositional, single-attribute case, structured (numerical) data are interpreted via propositions of the type:

$$\mathcal{P} = \{A \bowtie a \mid A \in \mathcal{A} \text{ and } a \in \mathbb{R}\},$$

where $\mathcal{A} = \{A_1, \ldots, A_n\}$ is a set of numerical attributes and $\bowtie \in \{<, \leq, \geq, >\}$. For an attribute A, we denote $dom(A) \subset \mathbb{R}$ the *domain* of A (i.e., the set of all values that A exhibits within the input dataset). Oblique trees, instead, can capture functional patterns between several attributes with a single proposition; for example, linear patterns can be captured by propositions of the type:

$$\mathcal{P} = \{[A_1, \ldots, A_n] \cdot \mathbf{w} \bowtie a \mid a \in \mathbb{R}\},$$

where $\mathbf{w}$ is a (column) vector of n real-valued attribute weights. Typically, in the propositional case, the formulas $\Phi(\mathcal{L})$ assigned to the edges of the tree simply consist of propositions. In the case of modal decision trees, where (numerical) unstructured data can be seen in terms of Kripke models over a set of worlds, attributes can be the result of scalar feature extractions performed on a specific world. For instance, for a multivatiate time series dataset with variables

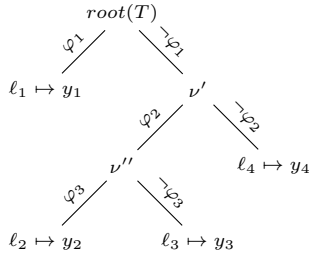


Figure 1: Example of a pure decision tree.

$\mathcal{V} = \{V_1, \ldots, V_m\}$, worlds are time intervals, and an attribute A can evaluate, for example, the average or maximum value of a variable within the interval; then, propositions of the same type as in the previous case can be used within temporal formulas that describe complex temporal patterns, built over the set of attributes:

$$\mathcal{A} = \{f(V) \mid V \in \mathcal{V}, f \in \mathcal{F}\},$$

in turn, based on a set $\mathcal{F}$ of *feature extraction functions*, which can be as simple as the average or the maximum, or as complex as (scalar) feature extraction via neural networks. Moreover, in modal decision trees each single condition φ is itself a complex formula, and, in the most general case, different conditions at different edges on the same path may be independent from each other.

In a pure decision tree, a path-formula is simply the conjunction of the formulas along the edges of the path. So, for example, consider the tree in Figure 1, and the leaf ℓ_2 on it. Each edge is labeled with a formula of a logic $\mathcal{L}$; hence, $\neg\varphi_1$ is the condition that instances have to meet to reach ν', $\neg\varphi_1 \wedge \varphi_2$ is the condition for reaching ν'', and $\neg\varphi_1 \wedge \varphi_2 \wedge \varphi_3$ is the condition for reaching ℓ_2. Thus, in the case of propositional decision trees, path-formulas are conjunctive normal (CNF) form propositional formulas, and in the case of modal decision trees they are conjunctions of modal formulas, which have a non-trivial structure designed to guarantee that modal decision trees are correct and complete [18].

Now, decision forests can be formalized given the foundations of decision trees. A *pure decision forest* is a structure defined as:

$$F = (T_1, T_2, \ldots, T_N, aggr),$$

where each T_i is a pure decision tree, for all $1 \leq i \leq N$, and $aggr : \mathbb{Y}^N \to \mathbb{Y}$ is an aggregation function of the set of individual tree predictions. In the following, we use the terms decision trees and decision forests (or, more simply, forests) to indicate their pure versions. Random forests [5] are the typical implementation of decision forests. To make a prediction using random forests, the input passes through all the N decision trees, and the aggregation function is applied to the resulting set of predictions; for example, in regression tasks, the final prediction of the forest can be the mean (or the median) value of the N predictions. At the propositional level, random forests have been used in a wide variety of situations; at the modal one, they have been used, for example, in [32].

Another noteworthy symbolc ML model for supervised tasks is the decision list [10, 41]. While decision trees have a hierarchical, branching structure, decision lists are linear, ordered sequences of if-then rules. A *decision list* is an ordered list of $r + 1$ rules

$R_1, R_2, \ldots, R_r, R_{def}$ defined as:

$$\Gamma = \begin{pmatrix} \psi_1 & \Rightarrow & y_1 \\ \psi_2 & \Rightarrow & y_2 \\ & \vdots & \\ \psi_r & \Rightarrow & y_r \\ \top & \Rightarrow & y_* \end{pmatrix},$$

where ψ_i is the *antecedent* (or condition) and $y_i \in \mathbb{Y}$ is the *consequent* (or outcome) of a rule $\psi_i \Rightarrow y_i$, for all $1 \leq i \leq r$, while $\top \Rightarrow y_*$ is the *default rule*. Decision lists so defined are already in their pure version, and there is no structural difference between the propositional and modal case.

Despite their structural differences, deriving decision lists from decision trees can be accomplished efficiently: each path from the root to a leaf node in a decision tree corresponds to a rule antecedent in a decision list, with the associated output label as the consequent. Owing to the recursive semantics of decision trees, the resultant lists exhibit non-overlapping rules with antecedents that are conjunctions of formulas, but no real need ordering and a default rule. More in general, however, rules in decision lists may be overlapping and may not cover the whole instance space, leading to the need of a default rule and of a linear order on the rules. Then, the decision list is interpreted as a sequence of if-then blocks: the rules are tested in order on a given instance and, in case no other rule is applicable, the default rule is invoked and the default consequent y_* is returned. As it is already clear, a rule antecedent can be, in general, a conjunction of complex formulas, and the algorithm that extracts a decision list from a decision tree in the propositional case readily applies to pure decision trees, and therefore to modal ones as well.

Multi-objective optimization problems and evolutionary algorithms. Let $\vec{x} \in \mathbb{R}^n$ be a vector of decision variables, and $o, u, s \in \mathbb{N}$. As in [14, 16], a *multi-objective optimization problem (MOOP)* is a problem defined as follows:

$$\begin{array}{rll} \min \ / \ \max & \vec{f}(\vec{x}) & (o \text{ objective functions}) \\ \text{s.t.} & \vec{g}(\vec{x}) \leq 0 & (u \text{ inequality constraints}) \\ \text{and} & \vec{h}(\vec{x}) = 0 & (s \text{ equality constraints}), \end{array}$$

where $\vec{f}(\vec{x}) \in \mathbb{R}^o, \vec{g}(\vec{x}) \in \mathbb{R}^u$, and $\vec{h}(\vec{x}) \in \mathbb{R}^s$. A MOOP is *constrained* if $u > 0$ or $s > 0$; otherwise, it is *unconstrained*. A MOOP can be *continuous*, in which we look for real values, or *combinatorial*, in which we look for objects from a countably (in)finite set, typically integers, permutations, or graphs. For a solution to be interesting, there must be a *dominance* relation between solutions. A point $\vec{x}$ *dominates* $\vec{x}'$ if and only if $f(\vec{x}) \leq f(\vec{x}')$ for all $f \in \vec{f}$ and $f(\vec{x}) < f(\vec{x}')$ for some $f \in \vec{f}$. A solution is *non dominated* (or *Pareto optimal*) if and only if there is no other solution that dominates it, and the set of non dominated solutions is called *Pareto front*.

Solving MOOPs can be approached in several ways, such as scalar, interactive, fuzzy, and metaheuristic-based methods [14], which include evolutionary strategies. In evolutionary strategies for MOOPs, the solving procedure begins by generating an initial population of candidate solutions, either through random generation or guided by background knowledge. Each solution is then evaluated by calculating its objective functions and constraints. Next, the population is modified iteratively using genetic operators: selection/reproduction, crossover, and mutation. The fittest solutions are chosen for the next generation, while the unfit ones are discarded. The remaining solutions undergo a crossover operator, producing novel solutions that are further mutated to maintain population diversity. Finally, the new population is evaluated, and the process is repeated until the termination criterion is satisfied. Evolutionary algorithms have been applied to solve MOOPs in several cases (e.g., see [16]); in our case, we leverage on their integration with genetic programming [30] due to the nature of the problem.

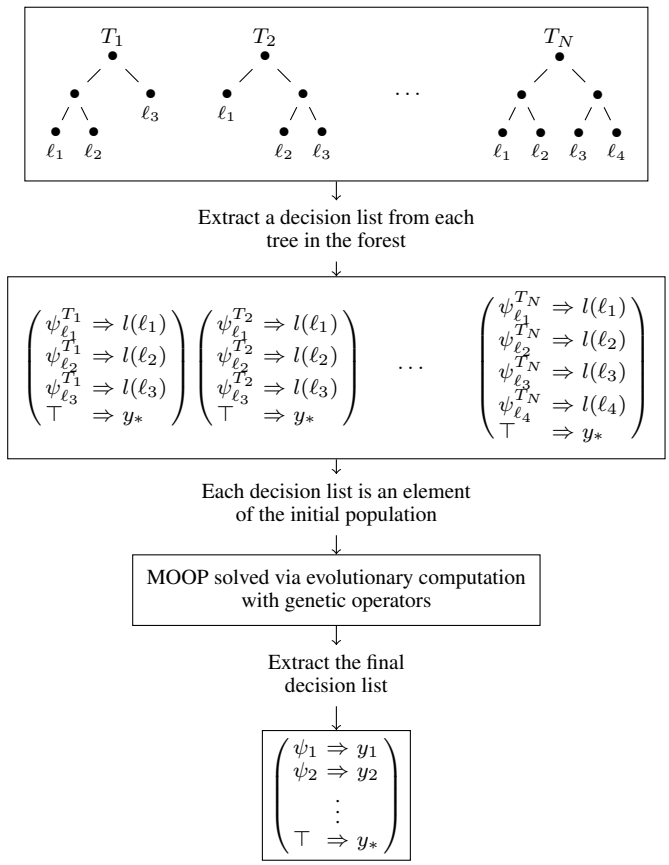


Figure 2: The proposed framework.

Among the many evolutionary algorithms for solving MOOPs, NSGA-II [17] uses a fast non-dominated sorting algorithm, sharing, elitism, and crowded comparison. Elitism implies that the best solutions of the previous iteration are kept unchanged in the current one, which significantly increases the convergence speed of the algorithm. Additionally, its use of a fast non-dominated sorting algorithm contributes to a significant reduction of its computational complexity.

3 Random Forests to Decision Lists as a MOOP

Problem definition. Extracting a decision list from a trained random forest model can be formalized by solving the following MOOP:

$$\begin{array}{rl} \min & Error(\Gamma) \\ \min & Complexity(\Gamma) \\ \text{s.t.} & |\Gamma| \geq |\mathbb{Y}| \\ & |\Gamma| \leq K, \end{array}$$

where $Error$ and $Complexity$ are measures of the statistical error and the complexity of a decision list Γ, and $K \geq |\mathbb{Y}|$ is the maximum number of rules in Γ (including the default rule). Observe that a model must have at least a rule for each label; solutions to this MOOP are decision list objects with high performance and low complexity.

Approach. Our approach, based on a multi-objective evolutionary algorithm, is graphically represented in Figure 2. In particular, given a forest F of N trees, the initial population of candidate solutions is computed as the set of N decision lists derived from the trees in the forest. Then, a multi-objective evolutionary algorithm is applied on the initial population, having chosen a specific measure of error and a specific measure of complexity as objective functions. Finally, from the last evolved population by the algorithm, single decision lists are extracted and analyzed.

In a classification scenario, the statistical performance of a decision list can be measured via global metrics such as the *overall accuracy*, *average accuracy*, *average precision*, *F1-score*, or the *error* (ε). As for complexity measures for decision lists, we identify structural ones such as the *number of rules* (ρ) and the *total length of the antecedents* (σ), defined as the total number of syntax tokens within the formulas; specifically, they only depend on the structure of the decision list. Furthermore, inspired by [34], we also consider the *average delay* (δ), computed as the average index of the fired rule within the decision list, or, in a similar way, the number of rules that are tested before the input is classified given a set of inputs to be classified by the decision list. While performance and complexity will be objects of optimization, the delay will only be used to evaluate the results.

Genetic operators. Similarly to [19], the underlying idea to extract rules, and therefore a decision list, from a random forest is to explore the search space in the vicinity of the rules extracted from the forest directly. This translates into genetic crossover/mutation operators designed to perform minimal modifications to the individuals (i.e., decision lists) in the current population. Considering that one of the objectives is to lower the complexity (in terms of interpretability) of the solutions, such operators propose decision lists with the same or fewer rules, having the rules' antecedents with the same or fewer conjuncts, which, in turn, are equally long or smaller.

The proposed genetic operators can be elegantly ordered by their level of abstraction (see Figure 3). From the most abstract one, *crossover*, which affects pairs of decision lists in the population, to mutation at the *decision list level*, which affects a single decision list, at the *rule level*, which affects a single rule of a single decision list, down to the *conjunct level*, which affects a single conjunct, and to the *leaf level*, which affects a single propositional letter of a single conjunct.

Our *crossover* considers two individuals Γ_1, Γ_2 such that at least one of them has at least one non-default rule (if it exists) and two random indexes $1 \leq i \leq r_1$ and $1 \leq j \leq r_2$ (r_1, r_2 being, respectively, Γ_1's and Γ_2's number of non-default rules). Then, it returns two new individuals Γ_1', Γ_2', as follows:

$$\Gamma_1 = \begin{pmatrix} & \dots & \\ \psi_i^1 & \Rightarrow & y_i^1 \\ & \dots & \end{pmatrix} \quad \Gamma_2 = \begin{pmatrix} & \dots & \\ \psi_j^2 & \Rightarrow & y_j^2 \\ & \dots & \end{pmatrix}$$
$$\Downarrow$$
$$\Gamma_1' = \begin{pmatrix} & \dots & \\ \psi_j^2 & \Rightarrow & y_j^2 \\ & \dots & \end{pmatrix} \quad \Gamma_2' = \begin{pmatrix} & \dots & \\ \psi_i^1 & \Rightarrow & y_i^1 \\ & \dots & \end{pmatrix}.$$

At the decision list level there are three mutations. The first one, *rule block position mutation*, considers a decision list Γ with at least two rules (if it exists) and chooses two indexes $1 \leq i \leq j \leq r$. Then, it returns Γ' obtained by moving all rules from the i-th to the j-th to a different position, as in the following example:

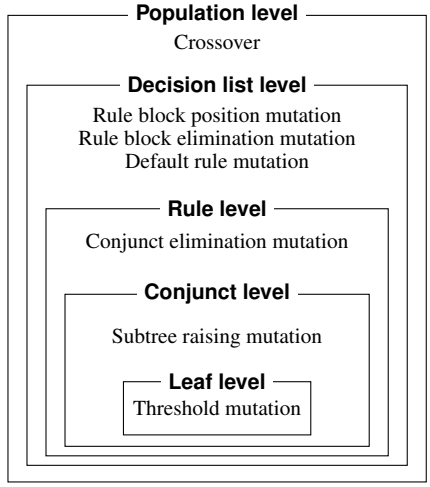


Figure 3: Hierarchical representation of the proposed genetic operators.

$$\Gamma = \begin{pmatrix} \psi_1 & \Rightarrow & y_1 \\ & \dots & \\ \psi_{i-1} & \Rightarrow & y_{i-1} \\ \psi_i & \Rightarrow & y_i \\ & \dots & \\ \psi_j & \Rightarrow & y_j \\ \psi_{j+1} & \Rightarrow & y_{j+1} \\ & \dots & \\ \psi_r & \Rightarrow & y_r \\ \top & \Rightarrow & y_* \end{pmatrix} \Rightarrow \Gamma' = \begin{pmatrix} \psi_i & \Rightarrow & y_i \\ & \dots & \\ \psi_j & \Rightarrow & y_j \\ \psi_1 & \Rightarrow & y_1 \\ & \dots & \\ \psi_{i-1} & \Rightarrow & y_{i-1} \\ \psi_{j+1} & \Rightarrow & y_{j+1} \\ & \dots & \\ \psi_r & \Rightarrow & y_r \\ \top & \Rightarrow & y_* \end{pmatrix}.$$

The second operator, *rule block elimination mutation*, considers a decision list Γ with at least one non-default rule (if it exists) and chooses two indexes $1 \leq i \leq j \leq r$. Then, it returns Γ' obtained by removing all rules from the i-th to the j-th, as follows:

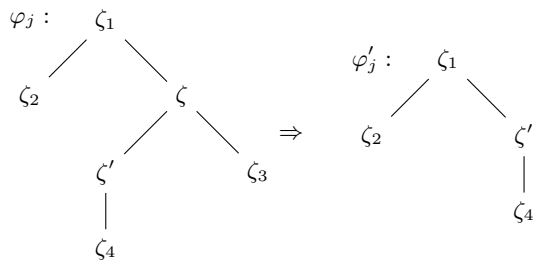
$$\Gamma = \begin{pmatrix} \psi_1 & \Rightarrow & y_1 \\ & \dots & \\ \psi_{i-1} & \Rightarrow & y_{i-1} \\ \psi_i & \Rightarrow & y_i \\ & \dots & \\ \psi_j & \Rightarrow & y_j \\ \psi_{j+1} & \Rightarrow & y_{j+1} \\ & \dots & \\ \psi_r & \Rightarrow & y_r \\ \top & \Rightarrow & y_* \end{pmatrix} \Rightarrow \Gamma' = \begin{pmatrix} \psi_1 & \Rightarrow & y_1 \\ & \dots & \\ \psi_{i-1} & \Rightarrow & y_{i-1} \\ \psi_{j+1} & \Rightarrow & y_{j+1} \\ & \dots & \\ \psi_r & \Rightarrow & y_r \\ \top & \Rightarrow & y_* \end{pmatrix}.$$

The third operator, *default rule mutation*, considers a decision list Γ and returns Γ' obtained by substituting the default rule's consequent with a different, randomly chosen one.

At the rule level, we propose the *conjunct elimination mutation* which, given a a decision list Γ, it considers one of the non-default rules with an antecedent consisting of more than one conjunct, and returns a new individual Γ', removing a randomly selected conjunct of the chosen (non-default) rule.

As for mutation at the conjunct level, recall that every single conjunct in a rule's antecedent may be, in general, a complex formula. Our mutation operator at this level, *subtree raising mutation*, considers a decision list Γ with at least one non-default rule (if it exists) and an index $1 \leq i \leq r$ of a non-default rule, chooses an index $1 \leq j \leq k$ (k being the number of conjuncts in the antecedent in the

chosen rule) of a rule R_j with a non-propositional antecedent (if it exists), and two nodes ζ, ζ' in the syntax tree that represents R_i such that ζ' is a descendant of ζ. Then, it returns the decision list Γ' obtained by substituting R_i with R'_i, whose j-th conjunct φ_j, in turn, is obtained by transplanting the subtree rooted in ζ' over the one rooted in ζ, as follows:



$$\Gamma = \left(\begin{array}{lllll} & & & \dots & \\ R_i: & \varphi_1 \wedge \ldots \wedge \varphi_j \wedge \ldots \wedge \varphi_k & \Rightarrow & & y_i \\ & & & \dots & \end{array} \right)$$

$$\Downarrow$$

$$\Gamma' = \left(\begin{array}{lllll} & & & \dots & \\ R'_i: & \varphi_1 \wedge \ldots \wedge \varphi'_j \wedge \ldots \wedge \varphi_k & \Rightarrow & & y_i \\ & & & \dots & \end{array} \right).$$

In the above example, $\zeta, \zeta', \ldots$ are syntax tokens of logic formulas; in the modal case, for instance, ζ could be a conjunction and ζ' could be a modal operator (e.g., $\Diamond$). A formula taken from an antecedent of a rule of an individual that belongs to the initial population is not arbitrary; its shape depends on how the modal decision tree learning algorithm effectively learns modal formulas. Since the mutation operator modifies its syntax tree, it must work for an arbitrary formula. Observe that, as expected, subtree rising too changes a rule, and therefore an individual, in the general direction of searching for simpler ones; shorter antecedents are, in fact, more general ones (being less restrictive).

Finally, we designed a single leaf level operator, which operates on the threshold occurring in propositional letters. Recall that, in the most general case, a propositional letter of the logical language (either propositional or modal) in which rules are written are of the type $A \bowtie a$, with $\bowtie \in \{<, \leq, \geq, >\}$ and $a \in dom(A)$. Let $std(dom(A))$ be the standard deviation of $dom(A)$. Our *threshold mutation* takes as input an individual Γ with at least one non-default rule (if it exists) and an index $1 \leq i \leq r$ of a non-default rule R_i, chooses an index $1 \leq j \leq k$ (k being the number of conjuncts in the antecedent in the chosen rule) of R_i, and a propositional letter p of the type $A \bowtie a$ in the conjunct φ_j of R_i. Then, it returns the individual Γ' obtained by substituting R_i with R'_i, whose j-th conjunct φ_j is substituted by φ'_j, in turn, obtained by replacing p with p' of the type $A \bowtie a'$, where $a' = \pm 0.1 \cdot std(dom(A)) + a$, as follows:

$$p: A \bowtie a \quad \Rightarrow \quad p': A \bowtie a'$$

$$\Gamma = \left(\begin{array}{lllll} & & & \dots & \\ R_i: & \varphi_1 \wedge \ldots \wedge \varphi_j \wedge \ldots \wedge \varphi_k & \Rightarrow & & y_i \\ & & & \dots & \end{array} \right)$$

$$\Downarrow$$

$$\Gamma' = \left(\begin{array}{lllll} & & & \dots & \\ R'_i: & \varphi_1 \wedge \ldots \wedge \varphi'_j \wedge \ldots \wedge \varphi_k & \Rightarrow & & y_i \\ & & & \dots & \end{array} \right).$$

4 Experiments and Results

To assess the performances of our approach in an experimental setting, we consider two tasks in the temporal domain. In particular, we consider two versions of the problem of diagnosing COVID-19 from audio samples of coughs, which can be posed as a binary, multivariate time series classification problem. *COVID-19* is a respiratory disease caused by the *SARS-CoV2* virus. The disease was classified in 2019, and caused a pandemic that lasted between 2020 and 2023.

The current scientific literature on COVID-19 is immense, and it ranges everywhere from medicine to economy, sociology, psychology, among many other fields; AI is no exception. Perhaps one of the most appealing lines of research, in this regard, deals with the possibility of deriving computational models for the diagnosis of COVID-19 from respiratory sounds of human subjects, an idea largely explored for diagnosing other respiratory diseases, such as bronchitis or pneumonia. Diagnosis is usually enabled via coughs, breaths and oral speech audio samples, which can be easily recorded with smartphone hardware.

Data and models. There are several public datasets that can be used to train a model to classify a cough sample as positive or negative to COVID-19. Among them, we consider the one originally crowdsourced by researchers at the University of Cambridge [7], which includes 9986 labeled audio samples recorded by 6613 volunteers. While most of the cough recordings contained several coughs, a semi-automated segmentation of the audio samples of coughs was performed, deriving a pool of single-cough audio samples. From this pool, three binary classification tasks were designed, namely, to distinguish between: *(T1)* declared positive subjects, and negative ones that have a clean medical history, have never smoked, and have no symptoms; *(T2)* declared positive subjects with cough as a symptom, and negative ones that have a clean medical history, have never smoked, and have cough as a symptom; *(T3)* declared positive subjects with cough as symptom, and negative ones that have asthma, have never smoked, and have cough as a symptom. Moreover, in the original dataset, $T2$ and $T3$ are coupled with their augmented version $T2+$ and $T3+$, obtained with standard data augmentation techniques.

The existing work concerning predictive models trained on this particular dataset includes several strategies. In particular, the temporal versions of modal decision trees and random forests have been proposed as a suitable method in [32]. The temporal component of this data emerges by considering sound as a collection of temporal variables, each representing the power at a specific frequency; then, each variable V_i refers to the i-th of a set of frequencies. As in the original paper, we considered 30 frequencies $V_1, \ldots, V_{30}$ distributed following the Mel scale between 0 and 8000Hz.

Experimental setting. We tested the ability of our methodology of extracting temporal decision lists from temporal random forest, that is, we used the modal version of random forests instantiated on $\mathcal{HS}$, following [32]. In this setting, the formula on each edge of a pure decision tree is an interval temporal formula, with unary operators that allow one to capture an interval in any of the possible Allen's relations [1], plus a *global* operator in its universal ($[G]$) and *existential* ($\langle G \rangle$) version. Briefly, $[G]\varphi$ predicates φ on each interval (of the Kripke model), while $\langle G \rangle \varphi$ predicates φ on some interval.

For each task, the experiments were conducted in two steps. First, we trained 10 random forests using ModalRF, implemented in `ModalDecisionTrees.jl` [35], each with 100 trees and associated with a specific pair training/test; as feature extraction func-

Table 1: Cross-validation results on the two tasks.

Dataset		RF		Avg DT			DL^{ε}				DL^{σ}				DL^{δ}			
		ε	ρ	ε	ρ	σ	ε	ρ	σ	δ	ε	ρ	σ	δ	ε	ρ	σ	δ
T2+	avg	5.0	392.9	15.1	3.9	45.8	7.7	2.1	10.4	0.6	8.6	2.0	8.3	0.6	9.1	2.3	10.8	0.6
	std	5.0	22.9	3.0	0.2	5.7	4.8	0.3	6.8	0.1	5.8	0.0	2.1	0.1	4.8	0.5	4.8	0.1
T3+	avg	8.8	366.6	19.7	3.7	36.9	10.0	2.1	14.0	0.5	15.6	1.7	8.3	0.4	18.1	1.8	10.4	0.3
	std	6.0	13.7	3.6	0.1	3.2	6.7	0.3	7.9	0.2	13.6	0.7	3.3	0.3	18.3	0.6	5.2	0.2

tions, we used *minimum* and *maximum* in an interval. The training/test splits are performed so that the two sets are strongly disjointed; this protocol prevents data leakage, guaranteeing a more accurate performance estimation. Second, we run ModalSTEL on each resulting forest. In particular, ModalSTEL operates a hyperparameter search, thanks to `Hyperopt.jl` [8], on the first forest, with an internal two-split cross-validation on the training set only, and then used the obtained parametrization on all 10 forests. The evolutionary part was realized by instantiating NSGA-II to this task, implemented in `Evolutionary.jl` [46].

In regard to parameters and objective functions, we proceeded as follows. All tasks are binary, that is, $|\mathbb{Y}| = 2$. As a measure of statistical error for an individual, we used the training classification error (computed as 1 minus the classification accuracy), while as a measure of its complexity, we used the absolute value of the difference between the number of rules and the number of classes; this steers the learning procedure to produce decision lists with exactly one rule per class. Summarizing, we instantiated our MOOP as follows:

$$\begin{array}{rl} \min & 1 - accuracy(\Gamma) \\ \min & |nrules(\Gamma) - 2| \\ \text{s.t.} & nrules(\Gamma) \geq 1. \end{array}$$

Finally, the hyperparameter search was single-objective, and based on the error metric only. During the search, we explored crossover rates ranging in $\{0.75, 0.8, 0.85, 0.9, 0.95\}$, as well as mutation rates ranging in $\{0.05, 0.1, 0.15, 0.2\}$. In particular, a single mutation operator was designed, which randomly applies one of the seven mutations explained in the previous section.

Results and discussion. Table 1 gives an overview of the results for each task; we limited the study to T2+ and T3+ only. For each task, we displayed, first, the average across the 10 forests (RF) of the performance and the structural characteristics of the initial forests, in terms of error (ε) and number of rules (ρ), that is, leaves in the entire forest. The second group of columns (Avg DT) gives us the average performance and structural characteristics of all initial decision trees across the 10 forests (in this case, for each forest we computed the average across the 100 initial decision trees), and we show, respectively, error, number of rules excluding the default rule, and number of syntax tokens (σ). The remaining grouped columns (DL^{ε}, DL^{σ}, and DL^{δ}) show the same results, plus the average delay (δ), for three decision lists taken from the last population, respectively, the one with the least error, with the least number of tokens, and with the least average delay.

There are several considerations that can be made upon an initial observation of Table 1. Let us compare, first, the (average behavior of the) original random forests (RF) against the (average behavior of the) decision lists with minimum error (DL^{ε}). In both cases the average error that is obtained by stepping from random forests to decision lists (in the case of decision lists with minimum error) increases by less than 2.7 percentage points. Such a loss, however, corresponds to a simplification in terms of number of rules of between 150 and 200 times. Moreover, comparing the average initial decision trees (Avg DT) with, again, the decision lists with minimum error in the final population, the error increased by a factor of $\sim$2 in correspondence with a simplification in terms of number of rules by about the same factor. Furthermore, we observe that the final decision lists with the least number of symbols (DL^{σ}) present a degradation in terms of error (1 point in the first case, 5 points in the second one) with an improvement in terms of number of rules of between 0.4 and 0.1, without considering the default rule. Finally, the final decision lists with the least delay (DL^{σ}) does not present a sensible improvement in terms of number of rules, and yet the error degrades.

Finally, consider the task $T3+$. On average, the original random forests for this task required combining 366.6 rules to attain a test error of 8.8%. As shown in [32], task $T3+$ on the same dataset was considered in at least other 6 occasions in the recent literature, and approached in several different ways (in most cases, with non-symbolic techniques); the errors of all such approaches ranged between 7% and 30%. This implies that temporal random forests are among the best approaches for this particular task, and, possibly, the absolute best one in the symbolic context. However, ModalSTEL was able to reduce the number of rules needed to one. As a matter of fact, the following decision list, extracted from the last population for this task, has a 100% accuracy on its corresponding test set:

$$\left(\begin{array}{lcl} \big([G](\min(V_6) < 6.16) \wedge & & \\ [G](\min(V_{13}) < 2396.00) \wedge & & \\ \langle G\rangle(\min(V_8) \geq 0.99)\big) & \Rightarrow & No \\ \top & \Rightarrow & Yes \end{array}\right).$$

5 Conclusion

Random forests provide superior predictive models compared to individual decision trees, albeit with limited interpretability. Building upon recent advancements, we have developed ModalSTEL, a methodology that can explain trained (modal) random forests through (modal) decision lists. Our contribution is two-fold. First, we lifted the existing STEL approach from traditional decision trees to modal ones. Modal decision trees harness the higher expressivity power of tailored modal logics (e.g., temporal and spatial) to learn from unstructured data (e.g., time series and images), which is not readily possible with conventional trees. Second, we formulated ModalSTEL for multi-objective optimization rather than an ad-hoc series of single-objective optimization steps. Based on open-source code and public time series data, our experiments showed auspicious results.

In modal logic (and alike), model checking is a notable problem that has been studied for decades in the case of deductive reasoning [11]. Learning, seen as an inductive process, is no exception to model checking (e.g., verifying the truth of formulas to split instances and classifying inputs with modal decision trees are applications of model checking). Thus, efficient model checking is directly proportional to scaling modal symbolic learning algorithms and models. As such, our future purposes are to exploit significant results on symmetry-based model checking algorithms [12, 22, 29, 43] and bisimulation procedures [20], effectively addressing the scalability issue. Finally, in terms of MOOPs, we plan to identify an adequate realization for the objective functions, while conducting ablation studies on specific genetic operators, investigating their influence on the overall convergence.

Acknowledgements

We thank the FIRD2022 project *Modal Geometric Symbolic Learning*, founded by the University of Ferrara, for partial financial support.

References

[1] J. F. Allen, 'Maintaining Knowledge about Temporal Intervals', *Communication of the ACM*, **26**(11), 832–843, (1983).
[2] D. W. Apley and J. Zhu, 'Visualizing the effects of predictor variables in black box supervised learning models', *Journal of the Royal Statistical Society Series B*, **82**(4), 1059–1086, (2020).
[3] G. Bechini, E. Losi, L. Manservigi, G. Pagliarini, G. Sciavicco, I. E. Stan, and M. Venturini, 'Statistical Rule Extraction for Gas Turbine Trip Prediction', *Journal of Engineering for Gas Turbines and Power*, **145**(5), (2023).
[4] J. Bertomeu, E. Cheynel, and E. Floyd, 'Using machine learning to detect misstatements', *Review of Accountinf Studies*, (26), 468—-519, (2021).
[5] L. Breiman, 'Random forests', *Machine Learning*, **45**(1), 5–32, (2001).
[6] L. Breiman, J. H. Friedman, R. A. Olshen, and C. J. Stone, *Classification and regression trees*, Wadsworth Publishing Company, 1984.
[7] C. Brown, J. Chauhan, A. Grammenos, J. Han, A. Hasthanasombat, D. Spathis, T. Xia, P. Cicuta, and C. Mascolo, 'Exploring Automatic Diagnosis of COVID-19 from Crowdsourced Respiratory Sound Data', in *Proc. of the 26th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, pp. 3474–3484, (2020).
[8] F. Bagge Carlson. Hyperopt. jl: Hyperparameter optimization in julia. https://github.com/bensadeghi/DecisionTree.jl, 2018.
[9] D. V. Carvalho, E. M. Pereira, and J. S. Cardoso, 'Machine Learning Interpretability: A Survey on Methods and Metrics', *Electronics*, **8**(8), 1–34, (2019).
[10] P. Clark and T. Niblett, 'The CN2 Induction Algorithm', *Machine Learning*, **3**, 261–283, (1989).
[11] E. M. Clarke, O. Grumberg, D. Kroening, D. A. Peled, and H. Veith, *Model Checking*, MIT Press, 2nd edn., 2018.
[12] E. M. Clarke, S. Jha, R. Enders, and T. Filkorn, 'Exploiting symmetry in temporal logic model checking', *Formal Methods in Systems Design*, **9**(1/2), 77–104, (1996).
[13] M. Coccagna, F. Manzella, S. Mazzacane, G. Pagliarini, and G. Sciavicco, 'Statistical and symbolic neuroaesthetics rules extraction from EEG signals', in *Proc. of the 9th International Work-Conference on the Interplay Between Natural and Artificial Computation (IWINAC)*, volume 13258 of *LNCS*, pp. 536–546. Springer, (2022).
[14] Y. Collette and P. Siarry, *Multiobjective Optimization: Principles and Case Studies*, Springer, 2004.
[15] M. W. Craven and J. W. Shavlik, 'Extracting Tree-Structured Representations of Trained Networks', in *Proceedings of the 8th Advances in Neural Information Processing Systems (NIPS)*, pp. 24–30, (1995).
[16] K. Deb, *Multi-objective optimization using evolutionary algorithms*, Wiley, 2001.
[17] K. Deb, S. Agrawal, A. Pratap, and T. Meyarivan, 'A fast and elitist multiobjective genetic algorithm: NSGA-II', *IEEE Trans. Evol. Comput.*, **6**(2), 182–197, (2002).
[18] D. Della Monica, G. Pagliarini, G. Sciavicco, and I.E. Stan, 'Decision trees with a modal flavor', in *Proc. of the 21st International Conference of the Italian Association for Artificial Intelligence (AIxIA)*, volume 13796 of *LNCS*, pp. 47–59. Springer, (2022).
[19] H. Deng, 'Interpreting tree ensembles with inTrees', *International Journal of Data Science and Analytics*, **7**(4), 277–287, (2019).
[20] A. Dovier, C. Piazza, and A. Policriti, 'An efficient algorithm for computing bisimulation equivalence', *Theoretical Computer Science*, **311**(1-3), 221–256, (2004).
[21] M. Du, N. Liu, and X. Hu, 'Techniques for interpretable machine learning', *Communications of the ACM*, **63**(1), 68–77, (2020).
[22] E. A. Emerson and A. P. Sistla, 'Symmetry and model checking', *Formal Methods in System Design*, **9**(1/2), 105–131, (1996).
[23] J. H. Friedman, 'Greedy function approximation: A gradient boosting machine.', *The Annals of Statistics*, **29**(5), 1189–1232, (2001).
[24] J. H. Friedman and B. E. Popescu, 'Predictive learning via rule ensembles', *The Annals of Applied Statistics*, **2**(3), 916–954, (2008).
[25] A. Goldstein, A. Kapelner, J. Bleich, and E. Pitkin, 'Peeking Inside the Black Box: Visualizing Statistical Learning With Plots of Individual Conditional Expectation', *Journal of Computational and Graphical Statistics*, **24**(1), 44–65, (2015).
[26] B. Goodman and S. R. Flaxman, 'European Union Regulations on Algorithmic Decision-Making and a "Right to Explanation"', *AI Magazine*, **38**(3), 50–57, (2017).
[27] R. Guidotti, A. Monreale, S. Ruggieri, F. Turini, F. Giannotti, and D. Pedreschi, 'A Survey of Methods for Explaining Black Box Models', *ACM Computing Surveys*, **51**(5), 93:1–93:42, (2019).
[28] J. Y. Halpern and Y. Shoham, 'A Propositional Modal Logic of Time Intervals', *Journal of the ACM*, **38**(4), 935–962, (1991).
[29] C. N. Ip and D. L. Dill, 'Better verification through symmetry', *Formal Methods in System Design*, **9**(1/2), 41–75, (1996).
[30] W. B. Langdon and R. Poli, *Foundations of genetic programming*, Springer, 2002.
[31] S. M. Lundberg and S.-I. Lee, 'A unified approach to interpreting model predictions', in *Proceedings of the 31st International Conference on Neural Information Processing Systems (NIPS)*, pp. 4768–4777, (2017).
[32] F. Manzella, G. Pagliarini, G. Sciavicco, and I.E. Stan, 'The voice of COVID-19: Breath and cough recording classification with temporal decision trees and random forests', *Artificial Intelligence in Medicine*, **137**, 1–14, (2023).
[33] S. K. Murthy, S. Kasif, and S. Salzberg, 'A System for Induction of Oblique Decision Trees', *Journal of Artificial Intelligence Research*, **2**, 1–32, (1994).
[34] F.E.B. Otero and A.A. Freitas, 'Improving the interpretability of classification rules discovered by an ant colony algorithm: Extended results', *Evolutionary Computation*, **24**(3), 385–409, (2016).
[35] G. Pagliarini, F. Manzella, G. Sciavicco, and I. E. Stan. ModalDecisionTrees.jl: Interpretable models for native time-series & image classification (v0.80), 2022.
[36] G. Pagliarini and G. Sciavicco, 'Decision tree learning with spatial modal logics', in *Proc. of the 12th International Symposium on Games, Automata, Logics, and Formal Verification (GandALF)*, volume 346 of *EPTCS*, pp. 273–290, (2021).
[37] N. Prentzas, M. Pitsiali, E. Kyriacou, A. Nicolaides, A. Kakas, and C.S. Pattichis, 'Model agnostic explainability techniques in ultrasound image analysis', in *Proc. of the 21st International Conference on Bioinformatics and Bioengineering (BIBE)*, pp. 1–6. IEEE, (2021).
[38] J. R. Quinlan, 'Induction of Decision Trees', *Machine Learning*, **1**, 81–106, (1986).
[39] J. R. Quinlan, *C4.5: Programs for Machine Learning*, Morgan Kaufmann, 1993.
[40] M. T. Ribeiro, S. Singh, and C. Guestrin, '"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 (KDD)*, pp. 1135–1144, (2016).
[41] R. L. Rivest, 'Learning Decision Lists', *Machine Learning*, **2**(3), 229–246, (1987).
[42] S. Sarkar, R. Raj, S. Vinay, J. Maiti, and D.K. Pratihar, 'An optimization-based decision tree approach for predicting slip-trip-fall accidents at work', *Safety Science*, **118**, 57–69, (2019).
[43] A. P. Sistla, V. Gyuris, and E. A. Emerson, 'SMC: a symmetry-based model checker for verification of safety and liveness properties', *ACM Transactions on Software Engineering and Methodology*, **9**(2), 133–166, (2000).
[44] C. Strobl, A.-L. Boulesteix, T. Kneib, T. Augustin, and A. Zeileis, 'Conditional variable importance for random forests', *BMC Bioinformatics*, **9**, (2008).
[45] S. Wachter, B. Mittelstadt, and C. Russell, 'Counterfactual Explanations Without Opening the Black Box: Automated Decisions and the GDPR', *Harvard Journal of Law & Technology*, **34**(2), 814–887, (2018).
[46] Stewart Weiss. Evolutionary.jl. https://github.com/wildart/Evolutionary.jl, 2023.