EVOTER: Evolution of Transparent Explainable Rule sets

HORMOZ SHAHRZAD, University of Texas at Austin, USA and Cognizant AI Labs, USA
BABAK HODJAT, Cognizant AI Labs, USA
RISTO MIIKKULAINEN, University of Texas at Austin, USA and Cognizant AI Labs, USA

Most AI systems are black boxes generating reasonable outputs for given inputs. Some domains, however, have explainability and trustworthiness requirements that cannot be directly met by these approaches. Various methods have therefore been developed to interpret black-box models after training. This paper advocates an alternative approach where the models are transparent and explainable to begin with. This approach, EVOTER, evolves rule sets based on extended propositional logic expressions. The approach is evaluated in several prediction/classification and prescription/policy search domains with and without a surrogate. It is shown to discover meaningful rule sets that perform similarly to black-box models. The rules can provide insight into the domain and make hidden biases explicit. It may also be possible to edit the rules directly to remove biases and add constraints. EVOTER thus forms a promising foundation for building trustworthy AI systems for real-world applications in the future.

CCS Concepts: • Computing methodologies → Genetic algorithms; Genetic programming.

Additional Key Words and Phrases: Genetic Algorithms, rule set Evolution, Explainable AI, XAI

ACM Reference Format:
Hormoz Shahrzad, Babak Hodjat, and Risto Miikkulainen. 2018. EVOTER: Evolution of Transparent Explainable Rule sets. In Proceedings of Make sure to enter the correct conference title from your rights confirmation email (Conference acronym 'XX). ACM, New York, NY, USA, 30 pages. https://doi.org/XXXXXXX.XXXXXXX

1 Introduction

Most of today’s popular and powerful Artificial intelligence and machine learning approaches rely on inherently black-box methods like neural networks, deep learning, and random forests. Therefore, they are difficult to explain, and their functionality in critical/sensitive use cases cannot be thoroughly understood. It is also difficult to modify them to remove biases or add known constraints. In some domains, such lack of transparency can lead to costly failures (e.g., in self-driving cars), and in others, mandatory regulations (e.g. insurance) or ethical concerns (e.g. gender or racial biases) cannot be met [Tjoa and Guan 2021].

Methods have been developed to try to explain the behavior of black-box models (e.g., deep learning networks) by interrogating them after training. In this manner, it is possible to e.g. identify the areas of the input to which the model is paying attention when making a prediction. However, post-training interrogation-based explanations are not always accurate or complete, and they can be open to interpretation.

Authors’ Contact Information: Hormoz Shahrzad, hshahrzad@cs.utexas.edu, University of Texas at Austin, Austin, Texas, USA and Cognizant AI Labs, San Francisco, California, USA; Babak Hodjat, babak@cognizant.com, Cognizant AI Labs, San Francisco, California, USA; Risto Miikkulainen, risto@cs.utexas.edu, University of Texas at Austin, Austin, Texas, USA and Cognizant AI Labs, San Francisco, California, USA.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than the author(s) must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from permissions@acm.org.

© 2018 Copyright held by the owner/author(s). Publication rights licensed to ACM.

Manuscript submitted to ACM
Interpretability and explainability, therefore, are different concepts [Lage et al. 2019; Linardatos et al. 2020]. Interpretability can be defined as "the ability to explain or to present in understandable terms to a human" [Doshi-Velez and Kim 2017]. Another popular definition is "the degree to which a human can understand the cause of a decision" [Miller 2019]. These definitions point to a behavior-based interpretation of systems where the actual cause of the behavior is not transparent. In contrast, the internal processes of explainable models are transparent and can be understood in mechanical terms. Thus, explainability means interpretability with transparency. As a result, explainable systems can be audited for compliance, which is useful for various stakeholders, from data scientists and business owners to risk analysts and regulators [Adadi and Berrada 2018; Belle and Papantonis 2021].

This paper introduces such an approach, the EVOLUTION of Transparent Explainable Rule sets, or EVOTER. These rule sets are transparently explainable and lead to domain insights that would be difficult to achieve through other machine-learning methods. The paper begins with a review of earlier methods that paved the way for the EVOTER approach, emphasizing key distinctions between them. Next, the EVOTER rule representation approach, applicable to both prediction/classification and prescription/policy search tasks, is presented. Experiments with EVOTER in a range of real-world domains then demonstrate that the approach is general and effective:

1. EVOTER is as accurate and less complex, and thus more explainable, than other transparent machine learning methods such as naive Bayes, decision trees, and random forests (shown in UCI iris, breast cancer, and human activity recognition tasks).
2. Rule-set evolution results in effective classification (shown in the iris, breast cancer, and static human activity recognition tasks) and effective static prescription (shown in the heart-failure prevention and diabetes treatment tasks).
3. Rule-set evolution results in effective time-series prediction (shown in the dynamic human activity recognition and blood-pressure classification tasks) and effective policy search (shown in the flappy bird and cart-pole control task).
4. It results in more interpretable explanations than other transparent methods such as naive Bayes, decision trees, and random forests (in cart-pole control).
5. Similar results are obtained both through direct evolution in the task itself and through evolution with a surrogate model of the task (in cart-pole control).
6. EVOTER leads to useful insights, including possibly identifying biases in the results (in diabetes treatment).
7. Explainability does not incur a major cost in accuracy. The performance of evolved rule sets is similar to that of neural networks trained with the same data, both when the task has a single objective (in heart-failure prevention) and when it has two objectives (in diabetes treatment).
8. The evolved rule sets are compressible, and only a small number of rules apply to each case, making the approach focused and economical (in heart-failure prevention).

The paper thus demonstrates that EVOTER is an effective and versatile approach to forming transparent and explainable models in machine-learning tasks.

2 Background

Explainability refers to the ease with which a human can understand the mechanisms behind a model’s behavior [Schwalbe and Finzel 2021]. Explainability is difficult to capture quantitatively because it would require quantifying psychological processes, and there are outside factors that matter: Models are more interpretable if they align well.
EVOTER: Evolution of Transparent Explainable Rule sets

with domain knowledge and domain-specific expectations, and if their behavior can be visualized e.g. through plotting decision surfaces or clustering the types of behaviors. In terms of building explainable AI models, there are two main directions: (1) use a black-box modeling method and make it more transparent; and (2) use a transparent method and make it more interpretable. These approaches are reviewed in the subsections below.

2.1 Making Black-Box Methods More Transparent

Black-box learning methods such as deep learning have been shown powerful in many domains. However, they are inherently difficult to explain because behavior is defined by nonlinear interactions of a very large number of parameters (i.e. connection weights). Recently, techniques have been developed to make their behaviors transparent [Adadi and Berrada 2018; Gunning and Aha 2019; Lakkaraju et al. 2019; Lundberg and Lee 2017; Ribeiro et al. 2016; Townsend et al. 2019]. For instance, SHAP (SHapley Additive exPlanations; [Lundberg and Lee 2017]) explains the output of any model as a sum of the effects of its features, offering local interpretability based on Shapley values from cooperative game theory. It can allocate credit consistently and fairly among features and is model-agnostic. However, it is also computationally intensive and the information it provides is often overwhelming. Similarly, LIME (Local Interpretable Model-agnostic Explanations; [Ribeiro et al. 2016]) provides explanations for individual predictions by approximating the model locally with a transparent model. It can be applied to any model type and it is good at explaining specific predictions, but it may not explain global behavior well.

Such explainability techniques can be evaluated along several dimensions [Bodria et al. 2021; Nauta et al. 2022], i.e. whether the explanations they provide have high fidelity, which measures how accurately the explanations reflect the behavior of the original model; stability, evaluating the consistency of explanations under slight variations in input; comprehensiveness, ensuring that explanations capture all relevant aspects influencing the model’s decisions; sufficiency, assessing whether the provided explanations are adequate for understanding the model’s decisions; and transparency, quantifying how transparent the model is in terms of its decision-making process. These metrics facilitate a quantitative assessment of how interpretable and understandable a model is as a whole, enabling stakeholders to gauge the trustworthiness and reliability of the model’s decisions.

2.2 Making Transparent Models More Interpretable

In the second direction, the substrate of the model is replaced with an inherently explainable structure, such as a probability set, decision tree, random forest, or rule set. Instead of e.g. running gradient descent in a deep neural network, a transparent representation of knowledge is constructed. Thus, the resulting model consists of a human-readable set of rules or equations. By reviewing this set, it is possible to understand what input features are used and how they are brought together in rules to make a decision. It is thus possible to explain the behavior of the model comprehensively in an exact human-readable form.

Classic such approaches include naive Bayes for learning class probabilities, decision trees for learning a hierarchy of tests on selected features to arrive at a classification, and random forests as an ensemble of such trees. They each have their own learning algorithms, or they can be evolved, especially to optimize their structure and hyperparameters automatically as well [Miikkulainen 2023]. The most general such approach, however, is to learn sets of rules. Because rule sets are highly structured, evolution is well-suited for learning them. Many early such systems were based on a fixed- or variable-length-chromosome genetic algorithm (GA) or a tree- or list-based genetic programming (GP) [Jacques et al. 2015; Peñarroya 2004; Poghosyan et al. 2021; Srinivasan and Ramakrishnan 2011]. In these methods, the rules were propositional logic expressions where each term is built from a feature compared against a constant, for
example, IF (Quality = Medium OR High) AND (Advertisement = Yes OR Telemarketing = Yes) AND (Gifts = Yes) AND (Sales Profile = Good OR Medium) THEN (Profit = Good) ELSE (Profit = Low) [Srinivasan and Ramakrishnan 2011]. These techniques were applied to several data sets in the UCI machine learning repository [Dua and Graff 2017].

EVOTER, a type of Grammatical Evolution (GE), advances rule-evolution systems by using a list-based grammatical approach to evolve interpretable rule sets. However, unlike traditional GE methods, which often focus on evolving programs or symbolic expressions for tasks like symbolic regression [Fernandes et al. 2023; Fonseca and Poças 2023; O’Neill and Ryan 1998], EVOTER focuses on producing interpretable rule sets for classification and decision-making tasks. Traditional GE approaches often result in complex models, which can be challenging to interpret, especially in domains like healthcare and finance that require high transparency.

In contrast, EVOTER emphasizes simplicity by employing a list-based grammar to evolve rules that are easy to understand, while maintaining fair accuracy. This focus on interpretability ensures that EVOTER’s models can be easily understood, verified, and adjusted by domain experts, balancing accuracy and transparency in practical applications. In EVOTER, rules are extended to a more powerful syntax that brings about three key advantages:

1. Time lags can be evolved directly for features that form a time series;
2. Features can be compared with each other; and
3. Coefficients and exponents can be evolved to implement linear and nonlinear operations on features.

This approach makes more complex applications possible, as will be demonstrated systematically in this paper. However, they also make the rule-set solutions more interpretable. Each of these extensions allows for more compact coding of behavior. Therefore, the resulting rule sets are simpler, with fewer features, clauses, and rules, and therefore easier for humans to understand.

Whereas human interpretability in general is difficult to quantify, simplicity (or complexity) of rule sets and other transparent machine-learning approaches can be readily measured, simply by counting the free parameters in them, or the number of decision points in them. Moreover, metrics have been developed for model selection that combine simplicity with performance, thus identifying models that represent good tradeoffs between quality and explainability. For instance, AIC (Akaike Information Criterion; [Akaike 1974]) estimates the relative amount of information lost by a given model: the less information a model loses, the higher the quality of that model:

\[
AIC = 2k - 2\ln(L)
\]

where \(k\) is the number of parameters in the model and \(L\) is the maximum value of the likelihood function for the model. AIC balances the model accuracy and complexity, even when models are non-nested or have different numbers of parameters, and does not require prior knowledge of the true model. However, it may favor more complex models, potentially leading to overfitting, and does not incorporate sample size, which can be a drawback in large datasets. In contrast, BIC (Bayesian Information Criterion; [Schwarz 1978]) adjusts the AIC by incorporating a penalty for the number of parameters proportional to the log of the sample size:

\[
BIC = \ln(n)k - 2\ln(L)
\]

where \(n\) is the sample size. BIC results in more consistent model selection, favoring the true model as sample size increases. On the other hand, it can be overly punitive towards models with more parameters in small samples, and it may not perform well if the true model is not among the considered candidates. Therefore, this paper includes both the...
2.3 Deploying Explainable Models

How does one determine which explainability technique to apply to a given ML model? Post-facto explainability methods are useful for pre-existing deep learning-based models that are hard to replace, given the cost of training. In many cases, deep learning is the preferred model type, especially in very large data problems such as unstructured text, sound, images, and video. In contrast, in problems where feature engineering has produced a structured data set with familiar input and output features (e.g., tabular data), rule-set evolution can be competitive and, in some cases, even more accurate, making it a preferable choice.

However, rule-set evolution can also be applied to unstructured data sets with continuous input or output features, and deep learning methods can sometimes do well in structured and tabular data problems. So the second, equally important consideration is whether interpretability is sufficient, or whether transparency is required. If explainability is, for example, mandated by regulations (as it is e.g. in healthcare, insurance, and finance), then it is important to consider transparently explainable modeling techniques such as EVOTER. For that reason, many of the examples in this paper are drawn from the healthcare domain.

3 Method

In this section, the design of the rule-set representations is first reviewed, followed by the methods for evolving them effectively.

3.1 Representing Behavior as Rule Sets

Rule sets in EVOTER are based on propositional logic expressions. They are collections of statements of the form “IF antecedent A is met THEN consequent B occurs”; the antecedents are conjunctions of conditions, and logical OR applies between the rules (Figure 1). Conditions compare single features with constant values or compare features with each other; coefficients are used to scale the features linearly, power expressions to scale them nonlinearly, and time lags to access feature values at different points in time.

There are several reasons why such rules are a good representation. First, they closely resemble verbal reasoning we use in our daily lives. Thus, they are interpretable and transparent, which is vital in applications where predictions need to be auditable so that experts can understand how and why a forecast or a recommendation was made. Second, such rules have a linear list structure, which helps avoid the usual problems of tree evolution (e.g., bloat and complex mutation and crossover operators) while remaining logically complete. Third, they can uncover nonlinear relationships and interactions among the domain features, as well as their temporal dynamics. Fourth, the conditions can be easily augmented to represent probabilities, as well as a variety of further mathematical operations and functions [Hodjat et al. 2018].

When parsing an individual’s rule set, conditions are evaluated in order, and all actions for conditions that are met are returned. In some domains (e.g. those in Sections 5.1-5.3), only the first action is executed; in others (e.g. Sections 5.5–5.4), a hard-max filter is used to select one of the actions based on the action coefficients; in yet others, all actions may be executed in parallel or in sequence. There are two additional extensions at this level: (1) Each rule has a times_applied field, which is incremented each time its antecedent evaluates to true when applied to a data point. This bookkeeping helps simplify the rule set and to reduce bloat, as described below. (2) A default rule is added at the end of
(a) BNF representation of EVOTER rules

\[(0.21 \times \text{Feature}_1^2 < 0.76 \times \text{Feature}_2) \rightarrow \text{Action}_3\]

(b) An example EVOTER rule

Fig. 1. Rule-set Representation in EVOTER

(a) The BNF grammar for the rules, including time lags, feature comparisons, and feature exponentiation. (b) An example rule illustrating these concepts, with elements color-coded as in the grammar. Rules can have multiple conditions and a rule set consists of multiple such rules (not shown). In addition, EVOTER keeps track of how many times the antecedent is satisfied, and the rule set includes a default rule that is applied if none of the antecedents apply in the current situation.

Each rule set, ensuring that some action is always produced even if no specific rules are triggered. This mechanism also makes evolution more efficient: Often the most common action is discovered as the default, and other rules then fine-tuned to address exceptions.

### 3.2 Evolving Rule Sets

Rule sets are evolved through genetic algorithms, adapted to take advantage of their structure. Two parents are first chosen through tournament selection, and offspring are generated through various crossover and mutation methods. These methods are specified in pseudocode in Appendix A.

To implement crossover, a high-level rule-set crossover method (Algorithm 2) calls one of three different strategies. First, in single-point crossover (Algorithm 3; Figure 2a), a crossover point is first chosen separately in each parent’s rule.
Fig. 2. The three crossover strategies in EVOTER. (a) In single-point crossover, the offspring inherits rules from parent\textsubscript{1} up to its crossover point, followed by rules from parent\textsubscript{2} after its crossover point. (b) In uniform crossover, each rule is selected uniformly randomly from either parent at each point in the rule set; in addition, all remaining rules are included from the larger set. (c) In combine-rules crossover, the offspring is initially cloned from parent\textsubscript{2}; a rule with a matching action in parent\textsubscript{1} and parent\textsubscript{2} is then selected, and its conditions are augmented with conditions from parent\textsubscript{1}. Each of these crossover strategies serves a different purpose, and they are selected randomly during each crossover.
A sample rule to mutate: \( \text{Race} = \text{Asian} \) AND \( \text{Age} > 65 \) AND \( \text{Smoking} = \text{True} \) \( \rightarrow \) \( \text{Action}_1 \)

- \( \text{Race} = \text{Asian} \) AND \( \text{Age} > 65 \) AND \( \text{Smoking} = \text{True} \) \( \rightarrow \) \( \text{Action}_1 \) (allowed)
- \( \text{Race} = \text{Asian} \) AND \( \text{Smoking} = \text{False} \) AND \( \text{Smoking} = \text{True} \) \( \rightarrow \) \( \text{Action}_1 \) (allowed)
- \( \text{Race} = \text{Hispanic} \) AND \( \text{Age} > 65 \) AND \( \text{Smoking} = \text{True} \) \( \rightarrow \) \( \text{Action}_1 \) (allowed)
- \( \text{Race} = \text{Asian} \) AND \( \text{Age} > 65 \) AND \( \text{Age} < 50 \) AND \( \text{Smoking} = \text{True} \) \( \rightarrow \) \( \text{Action}_1 \) (allowed)
- \( \text{Race} = \text{Asian} \) AND \( \text{Smoking} = \text{True} \) \( \rightarrow \) \( \text{Action}_1 \) (allowed)
- \( \text{Race} = \text{Asian} \) AND \( \text{Race} = \text{Hispanic} \) AND \( \text{Smoking} = \text{True} \) \( \rightarrow \) \( \text{Action}_1 \) (not allowed)

Fig. 3. Examples of permitted and rejected rule mutations in EVOTER. Gray highlighting on the right specifies the proposed mutation, green highlighting on the left identifies the results of permitted mutations, and red highlighting identifies the rejected mutation. In this manner, mutations that lead to contradictions and redundancies are automatically ruled out, making evolution more efficient.

precise rule that replaces the old rule in one parent. The three strategies thus each serve a different purpose. They are selected randomly at each crossover.

Mutations are implemented as single random changes to elements in the rule set. \textit{MutateConditions} (Algorithm 6) operates at the condition level, altering elements like the condition’s value, operator, or time lags. \textit{MutateRule} (Algorithm 7) operates at the rule level: It can modify a rule’s action, replace, add, or remove conditions, or change the rule’s certainty. If any such mutation would result in a contradiction or redundancy among the conditions, it is immediately rejected, thus making evolution more efficient (Figure 3). \textit{MutateRuleSet} (Algorithm 8) applies at the rule-set level, adding a random rule to the individual, removing an entire rule from the individual, changing the default rule, or changing the rule order. Note that these mutations can make the offspring smaller or larger than the original individual.

The \textit{times\_applied} counter is used to filter out inactive individual rules from participating in offspring generation (Algorithm 2). To reduce bloat, all conditions recognized as tautologies or falsehoods are removed from offspring individuals (Algorithm 9). While this process works well, several extensions can be implemented to make it more efficient in large-scale experiments, as outlined in Section 6. The experiments that follow focus primarily on quantitative comparisons to other transparent machine learning methods, and on illustrating the capabilities of the approach in various challenging real-world domains.

4 Quantitative Comparison of Transparent Methods

The experiments in this section show quantitatively that EVOTER is equally accurate than other transparent machine learning methods, while at the same time significantly less complex and more explainable in terms of metrics such as...
AIC and BIC. Three standard methods were included: naive Bayes, decision trees, and random forests. The comparison was run on six UCI classification datasets, each with unique characteristics and challenges:

1. **Iris** tests the method’s ability to classify flower species based on morphological measurements. It comprises three classes (versicolor, virginica, setosa) across 150 data points with four features.

2. **Breast Cancer** challenges methods to distinguish between benign and malignant breast cancer tumors. It contains two classes (malignant and benign) with 30 features across 561 data points.

3. **Human Activity Recognition (Static Features)** tests the method’s capacity to recognize human activities from featurized sensor data. Six activity classes (WALKING, WALKING UPSTAIRS, WALKING DOWNSTAIRS, SITTING, STANDING, and LAYING) are described by 561 static features derived from nine different gyro signals sampled at 50 Hz from smartphones. The dataset has 10,299 data points.

4. **Human Activity Recognition (Time-series Features)** evaluates how well the methods handle time-series data. The same six activity classes are represented by 1152 features, i.e. 128 samples of the nine gyro measurements over time. This variant also contains 10,299 data points.

5. **Credit Approval** focuses on predicting whether a credit card application should be approved based on various personal and financial attributes. It has two classes (approved or denied) with 690 instances and 15 attributes.

6. **Contraceptive Method Choice** involves classifying the choice of contraceptive methods (no use, long-term, or short-term methods) based on demographic and socio-economic attributes. The dataset consists of 1,473 instances with 10 attributes.

As usual with UCI datasets, accuracy was measured through the proportion of correct classifications. For the Naive Bayes experiments, the `GaussianNB()` implementation from the scikit-learn library was used; it is specifically designed to handle continuous data by modeling each feature as a Gaussian (normal) distribution. To measure model complexity, the number of parameters was calculated. For Naive Bayes models, they include the mean and standard deviation for each feature within each class; thus, the total number of parameters equals two times the number of classes times the number of features. For decision trees, the parameters comprise the key decision-making points in the model’s structure, i.e. the number of decision nodes (excluding leaf nodes. For random forests, they consist of all such decision nodes across all trees. For EVOTER, analogously to decision nodes, the total number of conditions across all rules in the rule set was counted. As proxies for explainability, the AIC and BIC metrics were calculated for each model. The structure of the methods and the datasets made the AIC and BIC calculations efficient, as detailed in Appendix B.

All models were trained on a randomly chosen 70% partition of the data, with the remaining 30% reserved as an unseen test set to obtain the accuracy measurement. This procedure was repeated ten times for each experiment using bootstrapping, a statistical method where the training and test sets are re-sampled with replacement for each run. Bootstrapping captures the variability in the data and provides a more accurate assessment of the model’s generalization ability than e.g. k-fold cross-validation for small datasets and for datasets where maintaining the exact class distribution is critical [Efron 1992; Fushiki 2011; Hastie et al. 2009]. The average accuracy, complexity, AIC, and BIC were calculated across the ten runs and statistical significance of differences between models estimated.

The results are summarized in Tables 1–6. In terms of accuracy, EVOTER performs as well as the best model in each case (the 0.01 difference from RF is not statistically significant; p > 0.064 in a paired t-test). However, in terms of the number of parameters and AIC and BIC, EVOTER is in a category of its own, reducing these metrics by at least an order of magnitude, and sometimes up to seven orders of magnitude. Thus, EVOTER is significantly more explainable than the other methods. This result is further elucidated in the accuracy vs. complexity trade-offs shown in Appendix C.
The domain of human activity recognition using time-series features (as shown in Table 4) highlights the first major extension of EVOTER over other rule-set systems. Out of 1,154 total time-series features, EVOTER utilized an average of only 142. The set of features was remarkably consistent across the ten runs: On average, 88% of these features were the same, whereas picking 142 features randomly would result in only 6% overlap. Thus, EVOTER consistently identified the points in the time series where each feature was most useful, and included only those in the rule sets. This ability can be coded more compactly in EVOTER by allowing it to evolve a time lag as a parameter of the feature directly. This extension will be evaluated in the experiment in Section 5.1.

| Model        | Accuracy | Number of Parameters | AIC    | BIC   |
|--------------|----------|----------------------|--------|-------|
| Naive-Bayes  | 0.95     | 24                   | 532    | 604   |
| Decision Tree| 1.00     | 50                   | 100    | 252   |
| Random Forest| 1.00     | 525                  | 1,085  | 2,666 |
| EVOTER       | 0.99     | 5                    | 83     | 98    |

Table 1. Model Comparison for Iris.

| Model        | Accuracy | Number of Parameters | AIC    | BIC   |
|--------------|----------|----------------------|--------|-------|
| Naive-Bayes  | 0.94     | 120                  | 2,589  | 3,110 |
| Decision Tree| 0.98     | 127                  | 1,111  | 1,662 |
| Random Forest| 0.99     | 2,014                | 4,574  | 13,322|
| EVOTER       | 0.98     | 5                    | 795    | 816   |

Table 2. Model Comparison for Breast Cancer.

| Model        | Accuracy | Number of Parameters | AIC    | BIC   |
|--------------|----------|----------------------|--------|-------|
| Naive-Bayes  | 0.75     | 6,732                | 189,059| 237,797|
| Decision Tree| 0.96     | 262,143              | 552,635| 2,450,499|
| Random Forest| 0.97     | 41,184,451           | 82,388,229| 380,555,502|
| EVOTER       | 0.96     | 35                   | 28,580 | 29,025|

Table 3. Model Comparison for Human Activity Recognition (Static Features).

| Model        | Accuracy | Number of Parameters | AIC    | BIC   |
|--------------|----------|----------------------|--------|-------|
| Naive-Bayes  | 0.76     | 13,824               | 196,957| 297,040|
| Decision Tree| 0.92     | 2,097,151            | 4,250,289| 19,433,248|
| Random Forest| 0.94     | 587,779,267          | 1,175,600,063| 5,431,005,623|
| EVOTER       | 0.93     | 61                   | 49,871 | 50,126|

Table 4. Model Comparison for Human Activity Recognition (Time-series Features).

| Model        | Accuracy | Number of Parameters | AIC    | BIC   |
|--------------|----------|----------------------|--------|-------|
| Naive-Bayes  | 0.85     | 987                  | 2,457  | 2,589 |
| Decision Tree| 0.90     | 15,432               | 3,290  | 5,012 |
| Random Forest| 0.93     | 56,743               | 12,543 | 23,891|
| EVOTER       | 0.91     | 150                  | 1,125  | 1,342 |

Table 5. Model Comparison for Credit Approval.

| Model        | Accuracy | Number of Parameters | AIC    | BIC   |
|--------------|----------|----------------------|--------|-------|
| Naive-Bayes  | 0.73     | 232                  | 3,142  | 3,356 |
| Decision Tree| 0.78     | 8,932                | 4,512  | 8,567 |
| Random Forest| 0.80     | 45,341               | 9,231  | 18,123|
| EVOTER       | 0.79     | 97                   | 1,578  | 1,892 |

Table 6. Model Comparison for Contraceptive Method Choice.
This ability is then extended to prescriptions/policy search in Section 5.2, where the value of its two other extensions, the experiments in this section each illustrate a different aspect of EVOTER’s performance. In Section 5.1, EVOTER’s time lags for the features directly (i.e. lags $t \times t^2 \times \ldots \times t^3$). The experiment thus demonstrated that EVOTER’s time-lag ability to predict/classify time series is illustrated again, but now through its extension to evolving time-lags for features.

5 Illustrating EVOTER’s Performance

The experiments in this section each illustrate a different aspect of EVOTER’s performance. In Section 5.1, EVOTER’s ability to predict/classify time series is illustrated again, but now through its extension to evolving time-lags for features. This ability is then extended to prescriptions/policy search in Section 5.2, where the value of its two other extensions, i.e. comparing features with each other and evolving exponents are demonstrated. In Section 5.3, similar prescriptions are shown to emerge when evaluation is done directly and through a surrogate model. Section 5.4 demonstrates that EVOTER results in explainability that is understandable to humans: The evolved rules are meaningful to domain experts and, in some cases, provide useful insight to them. Finally, Section 5.5 shows that the prescription performance of evolved rule sets is similar to the performance of evolved neural networks, and that the sets can be compressed and executed efficiently.

5.1 Rule Sets for Prediction/Classification

In this experiment, EVOTER was again applied to a time series prediction/classification task: Based on a series of blood pressure measurements over time, predict whether a patient in the ICU will go into septic shock in the next 30 minutes. However, instead of choosing among fixed features at different time points (i.e. features $x_{t1}, x_{t2}, \ldots x_{tn}$), EVOTER evolved time lags for the features directly (i.e. lags $t$ of $x[t]$). The experiment thus demonstrated that EVOTER’s time-lag

```
1. (Mean[4] < 72.75mmHg) & (Kurtosis[3] < 4.09) → Low
2. (Skew[10] > 2.01) & (Mean[8] < 88.92mmHg) & (Skew[4] < 0.15) → Normal
3. (Mean[0] < 72.75mmHg) → Low
4. (Mean[10] < 73.10mmHg) → Normal
5. (Mean[1] < 121.96mmHg) & (Mean[4] < 88.92mmHg) & (Mean[1] > 73.10mmHg) → High
6. (Mean[0] < 97.53mmHg) → Normal
7. (Mean[0] < 97.53mmHg) & (Kurtosis[0] > 12.71) → Normal
8. (Mean[4] < 72.75mmHg) & (Kurtosis[7] > 4.03) → Low
9. (Mean[4] > 121.96mmHg) & (Kurtosis[5] > 12.71) & (Kurtosis[3] > 1.00) → Normal
10. (Std[0] < 10.76) → High
11. (Kurtosis[0] > 1.00) → High
12. (Mean[0] < 72.75mmHg) & (Std[4] > 0.01) → Low
13. (Kurtosis[0] < 4.09) & (Skew[5] > 2.01) → Normal
14. (Skew[9] > 0.06) → High
15. (Skew[0] < 1.95) → High
16. (Mean[0] < 72.75mmHg) & (Mean[5] < 52.12mmHg) → Low
17. Default → Normal
```

Fig. 4. A sample rule set evolved to predict blood pressure. All the features are extracted from aggregations of MAP [Hemberg et al. 2014]. EVOTER discovered sets of features at specific time points to provide a useful signal for prediction. For instance, Std[4] specifies the standard deviation of the aggregated mean arterial pressure (MAP) over four minutes earlier. The evolved rules predict sepsis accurately and are interpretable and meaningful to experts.
evolution extension is effective in handling temporal dependencies, and showed how its explainability can provide valuable insight for decision-making in critical care.

**Experiment.** Consider the mean arterial pressure (MAP) signal at time $t$ as $x(t) \in \mathbb{R}$. The goal is to predict if the value of the statistic $\bar{m} = \text{avg}(\{x(t + a), \ldots, x(t + a + \beta)\})$ falls into one of the three intervals:

- If $\bar{m} \leq 55$ mmHg $\rightarrow$ Low;
- If $55$ mmHg $< \bar{m} \leq 85$ mmHg $\rightarrow$ Normal;
- If $\bar{m} > 85$ mmHg $\rightarrow$ High.

The first of these intervals indicates acute hypotension indicative of septic shock. In this sense, the task is a prediction/classification task.

**Results.** EVOTER was trained with approximately 4000 patients’ arterial blood pressure waveforms from the MIMIC II v3 data set [Hemberg et al. 2014]. Rules were evolved to observe various features of these waveforms at various times and predict whether acute hypotension was likely to develop within the prediction window of 30 minutes (Figure 4). In a massively parallel implementation on 1000 clients running for several days, EVOTER achieved an accuracy of 0.895 risk-weighted error on the withheld set, with a true positive rate of 0.96 and a false positive rate of 0.394.

Note that the approach is general and does not need a transformation of the time-series data into large temporal feature sets. Even though many machine learning methods could be used for this prediction task, EVOTER provides a solution that is transparent and interpretable. Indeed, the rules were evaluated by emergency-room physicians who found them meaningful [Hemberg et al. 2014]. Without such an understanding and verification, it would be difficult to trust the system enough to deploy it. Thus, working in tandem with human experts, EVOTER can be useful in interpreting and understanding the progression of the patient’s health and sensitivity and serve as an early indicator of problems.

### 5.2 Rule Sets for Prescription/Policy Search

The second set of experiments focuses on problems where the goal is not to predict a set of known labels but instead to prescribe, i.e. generate actions for an agent in a reinforcement learning environment. Results from two standard such domains are included: flappy bird and cart-pole. In both domains, evaluation is done by observing the effects of the actions directly in a simulation of the domain.

**5.2.1 Flappy Bird.**

**Experiment.** Flappy Bird is a side-scroller game where the player attempts to fly an agent between columns of pipes without hitting them by performing flapping actions at carefully chosen times. The experiment was based on a PyGame [Tasfi 2016] implementation running at a speed of 30 frames per second. The goal of the game is to finish ten episodes of two minutes, or 3,600 frames each, through random courses of pipes. A reward is given for each frame where the bird does not collide with the boundaries or the pipes; otherwise, the episode ends. The score of each candidate is the average reward over the ten episodes [Francon et al. 2020].

**Results.** EVOTER finds perfect solutions to this problem typically within 400 generations. Figure 5 shows a sample solution rule set. As is typical in this domain, the rules identify cases where the agent should flap. The conditions appear complex, however, several of the clauses are redundant and can be removed to form a final solution (as will be demonstrated in Section 5.5.1). Such redundancy is common in evolved solutions, likely because it makes the solutions...
While this principle applies to many evolutionary settings (e.g., neuroevolution [Gomez and Miikkulainen 1997]), rule of evolutionary optimization [Schmidt and Lipson 2009].

In this manner, EVOTER can discover and take advantage of physical relationships, which is, in general, a powerful ability contrast to the usual rule-set solutions that identify different situations and develop rules for each of them, this set is indefinitely when started with situations in the standard validation set. Figure 6 shows a sample of such a rule set. In contrast to the usual rule-set solutions that identify different situations and develop rules for each of them, this set is remarkably simple. It is based on a relationship between pole angle and cart velocity that applies across situations. In this manner, EVOTER can discover and take advantage of physical relationships, which is, in general, a powerful ability of evolutionary optimization [Schmidt and Lipson 2009].

---

**5.2.2 Cart-Pole.**

Experiment. Cart-pole is one of the standard reinforcement learning benchmarks. In the popular CartPole-v0 implementation in the OpenAI Gym platform [Brockman et al. 2016], there is a single pole on a cart that moves left and right depending on the force applied to it. The controller inputs are the position of the cart, the velocity of the cart, the angle of the pole, and the rotation rate of the pole. A reward is given for each time step that the pole stays near vertical, and the cart stays near the center of the track; otherwise, the episode ends [Francon et al. 2020].

Results. EVOTER finds solutions reliably within five generations. These rule sets will keep the pole from falling indefinitely when started with situations in the standard validation set. Figure 6 shows a sample of such a rule set. In contrast to the usual rule-set solutions that identify different situations and develop rules for each of them, this set is remarkably simple. It is based on a relationship between pole angle and cart velocity that applies across situations. In this manner, EVOTER can discover and take advantage of physical relationships, which is, in general, a powerful ability of evolutionary optimization [Schmidt and Lipson 2009].
The cart-pole domain can be used to illustrate the value of the second and third extensions of EVOTER, i.e. the ability to compare features and to exponentiate them. First, consider the discovered rule shown in Figure 6 but without the exponent, thus reducing it to a straightforward comparison between the two features. This rule establishes a linear decision boundary, as shown in Figure 7a. Naive Bayes largely misses it, making decisions based on pole angle only (Figure 7b). The decision tree captures this relationship, but struggles to represent it accurately (Figure 7c, even with the relatively complex model consisting of four levels (shown in Figure 7e). In effect, it can only do it by adding many decision nodes, like the random forest does Figure 7e). This illustration emphasizes an important point: Performance of these other transparent models comes with a significant cost in complexity, making them less explainable. Section 4 demonstrated this conclusion quantitatively; the visualization of the decision tree in Figure 7e makes it concrete: it is much more complex and difficult to understand than the simple linear rule it approximates, i.e. ‘IF (0.11 * velocity.of.cart < 0.87 * angle.of.pole) THEN Left; ELSE Right’.

Second, consider the actual nonlinear rule shown in Figure 6. The decision boundary is now nonlinear, as seen in Figure 8a. Naive Bayes approximates it with a straight line and the four-level deep decision tree with a series of blocks. Again, with more levels and parameters, as in the random forest, the approximation becomes better, with the cost of increased complexity and decreased explainability. Since many physical systems are nonlinear, the ability to represent exponents is crucial in uncovering such relationships. For instance, consider Kepler’s Third Law stating that the squares of the orbital periods of planets are directly proportional to the cubes of the semi-major axes of their orbits. Like many

Fig. 7. EVOTER versus other transparent models representing the linear rule ‘IF (0.11 * velocity.of.cart < 0.87 * angle.of.pole) THEN Left; ELSE Right’ (a) The decision boundary of this rule in EVOTER is a linear line. (b) The decision boundary of the naive-Bayes model largely misses the linear relationship and instead keys on the pole angle only. (c) The decision boundary of the decision tree with four levels struggles to approximate the relationship, even though it is quite complex, as shown in e (with colors representing increased certainty from red to green). It would need many more levels to be as accurate as the simple rule in EVOTER. (d) The decision boundary of the random-forest model with ten estimators is better, but this model also has many more parameters. Thus, the other models are less accurate or more complex (i.e., have more parameters) than the explanation discovered by EVOTER through its feature comparison extension.
similar laws of nature, this law was initially discovered as an underlying principle of numerous observations. Yet linear models such as decision trees cannot learn it: It is a power law that requires the representation of exponents. This level of complexity exceeds the capabilities of the other transparent systems, highlighting the importance of the nonlinearity extension in EVOTER in developing insights in scientific domains.

5.3 Evolving Rule Sets with a Surrogate

Many real-world domains are too costly for direct evolution, and the evaluation of candidates needs to be done more economically against a surrogate model [Francon et al. 2020; Miikkulainen et al. 2021]. The next experiment demonstrates that solutions evolved with a surrogate can be remarkably similar to those evolved directly in the domain.

Experiment. ESP (Evolutionary Surrogate-assisted Prescription; [Francon et al. 2020]) is a general approach to decision making where the decision policy is discovered through evolution and the surrogate is constructed through machine learning. The surrogate, or the predictor, can be an opaque model such as a random forest or neural network trained with gradient descent. Similarly, the decision policy, or the prescriptor, is often a neural network; however, evolving a rule set to represent the strategy results in an explainable solution instead.

In this experiment, ESP was set up to solve the same cart-pole domain as in the previous section. The prescriptor population was initially random and therefore generated a diverse set of actions. The predictor was trained with the outcome of these actions. The trained predictor was then used to evaluate prescriptor candidates through evolution.
Results. The predictor trained reliably with 100 action examples from the random initial population. Evolution then took only three generations to find agents that solved the examples in the validation set. Interestingly, the resulting rules (Figure 9) are logically almost identical to those resulting from direct evolution (Figure 6). While this validity of the surrogate modeling approach has been demonstrated before in terms of performance [Francon et al. 2020], rule-set evolution makes this conclusion explicit.

5.4 Obtaining Insights and Avoiding Biases

In this experiment, rule-based ESP was applied to the task of recommending medication for hospitalized diabetes patients that would both minimize hospital readmissions and lead to good discharge outcomes. The results were evaluated together with physicians, giving them a plain-text interpretation and indicating insights that are possible to obtain with this approach.

Experiment. The Diabetes data set [Strack et al. 2014] represents ten years (1999-2008) of clinical care at 130 US hospitals and integrated delivery networks. Information was extracted from the database for encounters that satisfied the following criteria:

- It is an inpatient encounter (a hospital admission).
- It is a diabetic encounter, i.e., one with any kind of diabetes was entered into the system as a diagnosis.
- The length of stay was at least one day and at most 14 days.
- Laboratory tests were performed during the encounter.
- Medications were administered during the encounter.

The data set includes over 50 features characterizing the patient and the hospital context, including patient number, race, gender, age, admission type, time in hospital, the medical specialty of admitting physician, number of lab tests performed, HbA1c test results, and diagnosis, as well as the number of outpatients, inpatients, and emergency visits by the patient in the year before the hospitalization. As actions, 21 different treatments can be prescribed, i.e., different diabetic and other medications. Two objectives are optimized, readmission rate and discharge disposition. The three possible readmission categories are: "no readmission" (+1 point), "readmission in less than 30 days" (0), and "readmission in more than 30 days" (-1). The discharge-disposition categories were: "sent back home" (+2 points), "remained in the hospital" (-1), "left with advised medical attention" (-1), "sent to another hospital" (-1), and "died" (-4). The goal is to maximize the readmission and discharge-disposition scores simultaneously.

A neural-network predictor was trained with ~66k samples in the historical data, with ~33k reserved for validation and ~33k for testing. A population of 100 rule sets was then evolved for 40 generations to generate a Pareto front: Given the patient and hospital contexts, they represent different tradeoffs between reducing readmissions and improving discharge dispositions. Note that these objectives are somewhat at odds: e.g., sending the patient home too early may result in early readmission.

Manuscript submitted to ACM
EVOTER: Evolution of Transparent Explainable Rule sets

1. (diag1.injury ≤ 2.2 + diag1.respiratory) & (diag2.neoplasms > 33) −→ Metformin-Pioglitazone
2. (age(60 – 70) ≥ 1.2 + age(10 – 20)) −→ Glyburide
3. (diag2.diabetes ≥ 0.82 + diag1.digestive) −→ Glipizide-Metformin
4. (admission.Court/Law ≥ 1.045 + diag3.circulatory) −→ Pioglitazone
5. (age(80 – 90) ≥ 2.29 + admission.type.id.Newborn) −→ Tolazamide
6. (age(30 – 40) ≤ 0.47 + age(50 – 60)) & (admission.type.id.Newborn ≤ 0.47 + age(50 – 60)) & (diag2.respiratory ≥ 1.86 + diag3.musculoskeletal) & (diag1.diabetes ≥ 1.42 + age(30 – 40)) & (0.18 + gender ≥ 0.38 + race.Asian) & (0.12 + race.AfricanAmerican ≥ 0.15 + admission.Court/Law) & (0.03 + race.Hispanic ≤ 0.76 + admission.Emergency.Room) −→ Insulin
7. Default −→ Glipizide

Fig. 10. A sample rule set discovered through ESP to reduce hospital readmissions and improve discharge disposition for diabetes. The rules are transparent but contain redundancies that make them sometimes hard to read. A domain expert can verify that the set is meaningful and translate them into plain English where the insights and biases are easier to see (Figure 11).

Results. The predictor training achieved a test accuracy of ~80% (of correct classifications into the three categories). Rule set evolution started with a no-readmission rate of ~29%, which improved to ~49% over the 40 generations; a similar improvement was observed along the discharge-disposition objective. Remarkable tradeoffs were discovered that performed much better than the actual prescriptions in the data set. Whereas 78% of patients were sent home and the no-readmission rate was 60% in the data set, the most dominant solution recommended treatments that were predicted to result in sending home as well as no readmission 99% of the time.

1. If the patient has a respiratory problem but not a neoplasms problem prescribe Metformin-Pioglitazone
2. If the patient is any age except 60 to 70 prescribe Glyburide
3. If the patient has Diabetic Diagnostic prescribe Glipizide-Metformin
4. If the patient has circulatory diagnosis prescribe Pioglitazone
5. If the patient has musculoskeletal diagnosis prescribe Tolazamide
6. If the patient’s age is between 30 to 60 & the patient has musculoskeletal diagnosis & is not Asian & was not sent by court/law & was admitted in emergency prescribe Insulin
7. If none of the above prescribe Glipizide

Fig. 11. Expert interpretation of the diabetes treatment rule set. In particular, Rule 6 is now easier to read. It also makes it possible to identify a potential racial bias; in this case, the bias is medically indicated, but in other applications, it may be due to inaccurate training data. Such biases can be identified and edited out when using rule sets, which would be difficult to do with other machine-learning approaches.

Figure 10 shows this highly effective rule set discovered by EVOTER. The rules are relatively transparent but again contain redundancies that make them somewhat difficult to read. To clarify and evaluate them, these rules were given to a domain expert who removed the redundancies and translated them into plain text (Figure 11). For instance, given that their features take on binary values, the last three clauses of Rule 6 can be simplified to “not Asian”, “not court/law”, and emergency. The expert verified that, indeed, this set constitutes a meaningful treatment policy.

This example also illustrates the potential of the rule set evolution to make biases in machine learning systems explicit. For instance, rule 6 is conditioned upon the race of the patients being Asian. In this case, race is indeed a medically valid consideration, but it is easy to see that in other domains with incomplete data, such constraints could be
Fig. 12. Evolution of neural-network and rule-set prescriptors in the single-objective domain of Heart-Failure Prevention. The plots are averaged over 10 runs and shading indicates 95% confidence intervals. The rule set population varies more throughout evolution, but the final best solutions perform almost the same on average, with a death rate of slightly less than 0.11. The differences are not statistically significant ($p = 0.76$). Thus, explainability is achieved with more diversity and practically no cost in performance.

simply due to biases in the data. With rule sets, it is possible to identify such biases. Furthermore, it is possible to edit them out when they are not warranted, which would be very difficult to do with other machine-learning approaches.

5.5 Performance of Evolved Rule Set vs. Neural-Network Prescriptors

The standard approach in ESP is to evolve neural networks as the prescriptor. While neural networks are powerful and flexible, they are opaque. On the other hand, rule sets are transparent and interpretable. An essential question is whether rule sets can achieve performance comparable to neural networks in this role. In this section, this question is answered in two different tasks, one (heart-failure prevention) involving a single objective and the other (diabetes treatment) two objectives. In both cases, the performance of rule sets is found only slightly short of that of the neural networks, demonstrating that explainability can be achieved with a minimal cost.

5.5.1 Heart-Failure Prevention Recommendations (Single Objective).

Experiment. A random-forest predictor was trained on the Heart-Failure data set [Chicco and Jurman 2020] to predict the probability of death for a patient given two possible interventions, i.e., ejection fraction and serum creatinine. The predictor achieved an out-of-sample Mathews correlation coefficient of 0.33.

The prescriptor receives the patient’s condition as its input and prescribes one of the two interventions. The predictor is queried to evaluate its fitness in minimizing deaths. The neural network prescriptor’s input nodes correspond to the patient’s condition and output nodes to the two possible treatments; it has a hidden layer of 16 nodes. Its weights were evolved over 100 generations using a population size of 100. Similarly, the rule-set prescriptor includes left-hand side clauses referring to the patient’s condition and right-hand side actions specifying interventions. Evolution was similarly set to run for 100 generations with a population of 100.

Results. The effectiveness of the two prescriptor models in preventing death is shown over evolution time in Figure 12, averaged over ten runs. It took more generations for rule-set evolution to reach the same level as the neural network evolution, and there is more diversity and variance in the rule-set populations. The likely reason is that even small changes in rules (e.g., adding or removing a clause) can have a large effect on performance. However, on average, the final results are very similar: The best rule set performs at 10.8% and the best neural network at 10.7%. Thus, explainability is achieved with practically no cost in performance. With such variability, an interesting question is:
EVOTER: Evolution of Transparent Explainable Rule sets

| Rule | Description |
|------|-------------|
| 1. &lt; 0.08 + anaemia ≤ 0.44 + platelets | 0.21 + ejection fraction |
| 2. &lt; 0.08 + platelets ≤ 0.45[0.1.0] | 0.59 + ejection fraction |
| 3. &lt; 0.84 + anaemia ≤ 0.40 + age | 0.04 + serum.createine |
| 4. &lt; 0.89 + serum sodium &gt; 0.97 + smoking | 0.32 + ejection fraction |
| 5. &lt; 1.00 + age &gt; 0.97 + platelets | 0.49 + ejection fraction |
| 6. &lt; 0.39 + creatinine.phosphokinase ≤ 0.92 + anaemia | 0.05 + serum.createine |
| 7. &lt; 0.94 + platelets ≤ 0.02 + anaemia | 0.65 + ejection fraction |
| 8. &lt; 0.46 + smoking ≤ 0.50 + diabetes | 0.51 + ejection fraction |
| 9. &lt; 0.49 + anaemia ≤ 0.92[0.1.0] & 0.44 + smoking ≤ 0.50 + diabetes & 0.09 + platelets ≤ 0.78[0.1.0] | 0.36 + ejection fraction |
| 10. &lt; 0.08 + high.blood.pressure ≤ 0.85 + diabetes | 0.21 + ejection fraction |
| 11. &lt; 0.91 + platelets ≤ 0.82 + high.blood.pressure | 0.32 + ejection fraction |
| 12. &lt; 0.63 + age &lt; 0.88 + high.blood.pressure | 0.51 + ejection fraction |
| 13. &lt; 0.10 + creatinine.phosphokinase ≤ 0.43 + sex | 0.46 + ejection fraction |
| 14. &lt; 0.47 + serum sodium &gt; 0.09 + creatinine.phosphokinase | 0.32 + ejection fraction |
| 15. &lt; 0.46 + smoking ≤ 0.50 + diabetes | 0.21 + ejection fraction |
| 16. &lt; 0.08 + high.blood.pressure ≤ 0.85 + diabetes | 0.45 + ejection fraction |
| 17. &lt; 0.94 + platelets ≤ 0.02 + anaemia | 0.65 + ejection fraction |
| 18. &lt; 1.00 + age &gt; 0.97 + platelets | 0.49 + ejection fraction |
| 19. &lt; 0.46 + smoking ≤ 0.50 + diabetes | 0.32 + ejection fraction |
| 20. &lt; 0.08 + high.blood.pressure &lt; 0.21 + sex | 0.51 + ejection fraction |
| 21. &lt; 0.46 + smoking ≤ 0.50 + diabetes | 0.46 + ejection fraction |
| 22. &lt; 0.06 + anaemia ≤ 0.82 + creatinine.phosphokinase | 0.51 + ejection fraction |
| 23. &lt; 0.84 + creatinine.phosphokinase ≤ 0.40 + age & 0.44 + sex ≤ 0.65[0.1.0] | 0.04 + serum.createine |
| 24. &lt; 0.46 + smoking ≤ 0.50 + diabetes | 0.21 + ejection fraction |
| 25. &lt; 0.08 + age ≤ 0.85 + diabetes | 0.21 + ejection fraction |
| 26. &lt; 0.09 + platelets ≤ 0.78[0.1.0] & 0.08 + age ≤ 0.85 + diabetes | 0.36 + ejection fraction |
| 27. &lt; 0.93 + platelets ≤ 0.82 + high.blood.pressure & 0.24 + serum sodium ≤ 0.29 + diabetes | 0.46 + ejection fraction |
| 28. &lt; 0.10 + creatinine.phosphokinase ≤ 0.43 + sex | 0.46 + ejection fraction |
| 29. &lt; 0.46 + smoking ≤ 0.50 + diabetes | 0.46 + ejection fraction |
| 30. &lt; 0.40 + diabetes ≤ 0.33 + serum sodium | 0.32 + ejection fraction |
| 31. &lt; 0.31 + creatinine.phosphokinase ≤ 0.02 + high.blood.pressure | 0.48 + ejection fraction |
| 32. &lt; 0.84 + creatinine.phosphokinase ≤ 0.40 + age | 0.04 + serum.createine |
| 33. &lt; 0.94 + platelets ≤ 0.02 + anaemia | 0.59 + ejection fraction |
| 34. &lt; 0.08 + age ≤ 0.85 + diabetes | 0.21 + ejection fraction |
| 35. &lt; 0.47 + serum sodium &gt; 0.09 + creatinine.phosphokinase | 0.32 + ejection fraction |
| 36. &lt; 0.84 + creatinine.phosphokinase ≤ 0.43 + sex | 0.32 + ejection fraction |
| 37. &lt; 0.46 + smoking ≤ 0.50 + diabetes | 0.21 + ejection fraction |
| 38. &lt; 0.08 + age ≤ 0.85 + diabetes | 0.59 + ejection fraction |
| 39. &lt; 0.46 + smoking ≤ 0.50 + diabetes & 0.08 + platelets ≤ 0.45[0.1.0] | 0.59 + ejection fraction |
| 40. &lt; 0.08 + age ≤ 0.85 + diabetes | 0.21 + ejection fraction |
| 41. &lt; 0.46 + smoking ≤ 0.50 + diabetes | 0.21 + ejection fraction |
| 42. &lt; 0.09 + platelets ≤ 0.78[0.1.0] & 0.08 + age ≤ 0.85 + diabetes | 0.36 + ejection fraction |
| 43. &lt; 0.21 + high.blood.pressure ≤ 0.85 + diabetes | 0.36 + ejection fraction |
| 44. &lt; 0.46 + smoking ≤ 0.50 + diabetes | 0.21 + ejection fraction |
| 45. &lt; 0.08 + smoking ≤ 0.30 + high.blood.pressure & 0.08 + age ≤ 0.85 + diabetes | 0.21 + ejection fraction |
| 46. &lt; 0.21 + high.blood.pressure ≤ 0.85 + diabetes | 0.21 + ejection fraction |
| 47. &lt; 0.08 + age ≤ 0.85 + diabetes | 0.21 + ejection fraction |

Fig. 13. A sample solution rule set for the Heart-Failure Prevention domain. The default rule indicates serum, and the others mostly identify situations where ejection fraction is preferable. Many of the rules are redundant, which is a common result of the search process. Based on the times applied counters, those rules can be removed, resulting in a compact set shown in Figure 14.

How can one candidate be chosen in the end such that it most likely generalizes to unseen data, given the multiple hypothesis problem [Austin et al. 2014; Avdorova 1966]? It turns out that under a smoothness assumption that similar candidates perform similarly, picking a candidate based on the average performance of its neighbors provides a viable strategy [Miikkulainen et al. 2017].
was run on the same Diabetes Treatment recommendation task as in Section 5.4. Again, the goal was to minimize slightly higher compared to the neural networks across the ten runs. This variation can be attributed to the discrete population of 100 rule sets and 100 neural networks over 100 generations, and evolution repeated ten times.

A sample rule-set solution is shown in Figure 13. This rule set prioritizes scenarios where the ejection fraction is favorable and utilizes serum creatinine as the default prescription. As discussed in Section 5.2.1, there are many redundant rules as well. To improve interpretability, rules with times_applied = 0 were pruned, resulting in a much smaller set with 29, as shown in Figure 14. Each of these rules participates in making decisions on this dataset, however, only four of them are activated for a given input on average. Such conciseness makes interpretation and explanation easier, supporting applications in safety-critical fields such as healthcare, finance, and law.

Fig. 14. Solution rule set from Figure 13 compressed by pruning inactive rules. The original rule number is in parenthesis, followed by the number of times the rule was applied (out of 300 samples). This set is functionally equivalent but much smaller. Also, on average only four rules are active for each case in the dataset. Such conciseness makes interpretation and explanation easier, supporting applications in safety-critical fields such as healthcare, finance, and law.

5.5.2 Diabetes Treatment Recommendations (Multiobjective).

Experiment. In order to assess the cost of explainability in a domain with multiple objectives, a second comparison was run on the same Diabetes Treatment recommendation task as in Section 5.4. Again, the goal was to minimize hospital readmissions while simultaneously improving discharge dispositions. Rule-based ESP was again used to evolve a population of 100 rule sets and 100 neural networks over 100 generations, and evolution repeated ten times.

Results. Figure 15 provides an overview of the results obtained. Again, the variation in the evolved rule sets is slightly higher compared to the neural networks across the ten runs. This variation can be attributed to the discrete
nature of the rule set components in contrast to the continuous values of the weights in neural networks. Although the average performance curves are noticeably different, their confidence intervals still overlap. This result suggests that the performance cost associated with the explainability provided by the rule sets is small (i.e. barely significant).

In sum, the experiments in both single and multiobjective domains suggest that there may be a small cost in performance when evolving rule sets vs. neural network prescriptors. Depending on the domain, such cost may be well worth it in order to achieve transparent, explainable performance.

6 Discussion and Future Work

The experiments presented in this paper suggest that rule-set evolution in general and its EVOTER implementation in particular is a practical and effective approach to machine learning. EVOTER applies to a wide range of tasks, including prediction, classification, prescription, and policy search, both in static and time-series contexts. It is possible to discover solutions directly in the domain, or through surrogate models that represent the domain safely and cost-effectively. Most importantly, compared to black-box models such as neural networks, rule-set solutions are explainable, i.e. both interpretable and transparent. They can provide insight into the domain, uncovering fundamental principles as well as hidden biases. While there may be a slight performance cost compared to black-box models, it may be well worth it in many safety-critical domains.

There are several compelling directions for future work. In terms of technical extensions, simplicity could be further encouraged through an explicit secondary evolution objective to favor smaller rule sets, incorporated e.g. through NSGA-II [Deb et al. 2002]. Similarly, exploration and creativity can be encouraged through a secondary novelty objective [Shahrzad et al. 2018, 2020]. To encourage general solutions, the times\_applied counter can be useful: A very low count may indicate overfitting or even a corner case bias, and evolution can be directed to avoid such cases. Further, the rule grammar could be expanded during evolution gradually. The process could be started with a restricted set of operators that could then be expanded as evolution progresses, thus implementing a form of curricular learning. In cases where the data set may be excessively large, partial and incremental evaluation through age-layering can be used to make the process more efficient [Hemberg et al. 2014; Hodjat and Shahrzad 2013; Shahrzad et al. 2016]. EVOTER also lends
itself well to parallelization and federation, both at the level of evaluating individuals and at the level of evaluating conditions. Further, condition evaluation can be tensorized and evaluated in parallel on different data points, taking advantage of GPUs.

A useful practical extension to EVOTER would be to include a facility for the expert to edit the discovered rule sets. First, such a facility could be used to identify and remove redundancies, making the rules easier to read. Although much of such simplification can be done automatically, there is still room for human insight. Second, the facility can provide a means for removing biases and making sure that the models are fair. Unlike with opaque models, the learned biases are explicit and can be evaluated and edited out if desired. Third, it may be possible to edit the rules to add further knowledge that may be well understood but challenging to learn, such as safety limits and other real-world constraints.

More generally, it may be possible to build rule-set twins for existing SoTA black-box models. In this manner, converting an opaque but well-performing model into a transparent rule-set equivalent may be possible. This model can then be used to explain the learned behavior and identify its biases and potential weaknesses. Such rule-set models are executable, and if they replicate the performance of the black-box model accurately enough, they could be deployed instead of the black box in explanation-critical applications. In this manner, the process of rule set distillation can play a vital role in deploying machine learning systems in the real world.

It may also be possible to create hybrid systems that take advantage of both black-box and explainable elements. Such a system could e.g. use deep neural networks to extract visual features from medical images, such as the position, dimension, and texture of lumps in X-rays, followed by EVOTER to derive transparent, verifiable rules based on these features. This methodology would not only make it possible to audit the system but also allow an expert to verify the features extracted by the deep network, thus resulting in a more robust and trustworthy machine-learning application.

To encourage adoption further, Large Language Models (LLMs) [Bubeck et al. 2023; et al. 2023; Touvron et al. 2023] could be used to translate EVOTER rules into plain language. Given the propositional rules as input, LLMs should be able to generate natural language explanations, descriptions, and interpretations. This approach would enable users with varying levels of expertise to comprehend the underlying concepts and their functions. It could thus potentially bridge the gap between intricate machine learning techniques and a broader audience, making it adoption of such systems easier.

7 Conclusion

This study demonstrated rule-set evolution, implemented in EVOTER, as a promising approach to developing inherently transparent and explainable models. The approach contrasts with the dominant black-box models, such as neural networks, in machine learning. Rule-set evolution can be applied to various tasks, including prediction, classification, prescription, and policy search, to both static and time-series problems, and with and without surrogate models of the domain. While a slight performance cost can be expected compared to black-box models, it may be acceptable in many cases considering the benefits of explainability. Rule-set evolution can thus enhance the trust, understanding, and adoption of AI systems in practical applications.
EVOTER: Evolution of Transparent Explainable Rule sets

References

Amita Adadi and Mohammed Berrada. 2018. Peeking Inside the Black-Box: A Survey on Explainable Artificial Intelligence (XAI). *IEEE Access* 6 (2018), 52138–52160.

H. Akaike. 1974. A new look at the statistical model identification. *IEEE Trans. Automat. Control* 19, 6 (1974), 716–723.

RS Austin, Isaac Dalsingh, and NS Altman. 2014. Multiple Hypothesis Testing: A Review. *Journal of the Indian Society of Agricultural Statistics* 68 (01 2014), 303–314.

Ligia Igorevna Avdeeva. 1966. *Simultaneous statistical inference*. Springer.

Vaishak Belle and Ioannis Papantonis. 2021. Principles and Practice of Explainable Machine Learning. *Handbook of Evolutionary Machine Learning*. Risto Miikkulainen. 2023. *Evolutionary Supervised Machine Learning*. In Scott M. Lundberg and Su-In Lee. 2017. A Unified Approach to Interpreting Model Predictions. In *Proceedings of Isaac Lage, Emily Chen, Jeffrey He, Menaka Narayan,Been Kim, Sam Gershman, and Finale Doshi-Velez. 2019. An evaluation of the human-interpretability of explanation. arXiv:1902.00906 (2019).

Himabindu Lakkaraju, Ece Kamar, Rich Caruana, and Jure Leskovec. 2019. Faithful and Customizable Explanations of Black Box Models. In *Proceedings of the 2019 AAAI/ACM Conference on AI, Ethics, and Society (Honolulu, HI, USA) (AIES ’19)*. Association for Computing Machinery, New York, NY, USA, 131–138.

Pantelis Linardatos, Vassilis Papastefanopoulos, and Sotiris Kotsiantis. 2020. Explainable ai: A review of machine learning interpretability methods. *Entropy* 23, 1 (2020), 18.

Scott M. Lundberg and Su-In Lee. 2017. A Unified Approach to Interpreting Model Predictions. In *Proceedings of the 31st International Conference on Neural Information Processing Systems*. Curran Associates Inc.

Risto Miikkulainen. 2023. Evolutionary Supervised Machine Learning. In *Handbook of Evolutionary Machine Learning*, W. Banzhaf, P. Machado, and M. Zhang (Eds.). Springer, New York.

Manuscript submitted to ACM
Risto Miikkulainen, Olivier Francon, Elliot Meyerson, Xin Qiu, Darren Sargent, Elisa Canzani, and Babak Hodjat. 2021. From Prediction to Prescription: Evolutionary Optimization of Non-Pharmaceutical Interventions in the COVID-19 Pandemic. *IEEE Transactions on Evolutionary Computation* 25 (2021), 386–401.

Risto Miikkulainen, Hormoz Shahrzad, Nigel Duffy, and Phil Long. 2017. How to Select a Winner in Evolutionary Optimization?. In *Proceedings of the IEEE Symposium Series in Computational Intelligence*. IEEE. http://www.cs.utexas.edu/users/ai-lab?miikkulainen:ssci17

Tim Miller. 2019. Explanation in artificial intelligence: Insights from the social sciences. *Artificial intelligence* 267 (2019), 1–38.

Meike Nauta, Jan Trienes, Shreyasi Pathak, Elisa Nguyen, Michelle Peters, Yasmín Schmitt, Maurice van Keulen, and Christin Seifert. 2022. From Anecdotal Evidence to Quantitative Evaluation Methods: A Systematic Review on Evaluating Explainable AI. *Comput. Surveys* 55 (2022), 1–42. https://api.semanticscholar.org/CorpusID:246063780

Michael O’Neill and Conor Ryan. 1998. Grammatical evolution: Evolving programs for an arbitrary language. In *Genetic Programming*. Springer, 83–96.

Jaume Bacardit Pèlarroya. 2004. Pittsburgh genetic-based machine learning in the data mining era: representations, generalization, and run-time.

Arnak Poghosyan, Asht Harutyunyan, Naira Grigoryan, and Nicholas Kushmerick. 2021. Incident Management for Explainable and Automated Root Cause Analysis in Cloud Data Centers. *Journal of Universal Computer Science* 27, 11 (2021), 1152–1173.

Marco Tulio Ribeiro, Sameer Singh, and Carlos Guerzin. 2016. “Why Should I Trust You?”. Explaining the Predictions of Any Classifier. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining* (San Francisco, California, USA) (KDD ’16) Association for Computing Machinery, New York, NY, USA, 1135–1144.

Michael Schmidt and Hod Lipson. 2009. Distilling Free-Form Natural Laws from Experimental Data. *Science* 324, 5923 (2009), 81–85.

Gesina Schwalbe and Bettina Finzel. 2021. A comprehensive taxonomy for explainable artificial intelligence: A systematic survey of surveys on methods and concepts. *Data Mining and Knowledge Discovery* (2021), 1–59.

G. Schwarz. 1978. Estimating the dimension of a model. *The Annals of Statistics* 6, 2 (1978), 461–464.

Hormoz Shahrzad, Danial Funk, and Risto Miikkulainen. 2018. Enhanced Optimization with Composite Objectives and Novelty Selection. In *Proceedings of the 2018 Conference on Artificial Life*. Tokyo, Japan. http://www.cs.utexas.edu/users/ai-lab?shahrzad:alife18

Hormoz Shahrzad, Babak Hodjat, Camille Dolle, Andrej Denisov, Simon Lau, Donn Goodhew, Justin Dyer, and Risto Miikkulainen. 2020. Enhanced Optimization with Composite Objectives and Novelty Pulsation. In *Genetic Programming Theory and Practice XVII*, Wolfgang Banzhaf, Erik Goodman, Leigh Shenenman, Leonardo Trujillo, and Bill Wozel (Eds.). Springer, New York, 275–293.

Hormoz Shahrzad, Babak Hodjat, and Risto Miikkulainen. 2016. Estimating the Advantage of Age-Layering in Evolutionary Algorithms. *arXiv:1602.06911*.

Sujatha Srinivasan and Sivakumar Ramakrishnan. 2011. Evolutionary multi objective optimization for rule mining: A review. *Artificial Intelligence Review* 36 (10 2011), 205–248. https://doi.org/10.1007/s10462-011-9212-3

Beata Strack, Jonathan Deshazo, Chris Gennings, Juan Luis Olmo Ortiz, Sebastian Ventura, Krzysztof Cios, and John Clore. 2014. Impact of HbA1c Measurement on Hospital Readmission Rates: Analysis of 70,000 Clinical Database Patient Records. *BioMed Research International* 2014 (2014), 781670.

Norman Tasfi. 2016. PyGame Learning Environment. https://github.com/ntasfi/PyGame-Learning-Environment.

Erico Tjoa and Cuntai Guan. 2021. A Survey on Explainable Artificial Intelligence (XAI): Toward Medical XAI. *IEEE Transactions on Neural Networks and Learning Systems* 32 (2021), 4793–4813.

Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal Azhar, Aurelien Rodriguez, Armand Joulin, Edouard Grave, and Guillaume Lample. 2023. LLaMA: Open and Efficient Foundation Language Models. *arXiv:2302.13971* (2023).

Joseph Townsend, Thomas Chaton, and Joao M Monteiro. 2019. Extracting relational explanations from deep neural networks: A survey from a neural-symbolic perspective. *IEEE transactions on neural networks and learning systems* 31, 9 (2019), 3456–3470.

Manuscript submitted to ACM
A Implementation of initialization, mutation, crossover, and pruning

Algorithm 1 describes how rule sets are randomly initialized in the beginning of evolution; Algorithms 2 through 5 describe how rule sets are crossed over; Algorithms 6 through 8 specify how individual rules are changed through mutation; Algorithm 5 implements the novel method of combining conditions and actions between two rule sets. Algorithm 9 illustrates how the rule sets are simplified through pruning based on the times_applied counters.

Algorithm 1 Generate Random Rule Set

Require: max_rules: Maximum number of building block rules
Require: max_conditions: Maximum number of building block conditions per rule
Require: action_set: Set of possible domain actions
Ensure: Returns a randomly generated RuleSet

1: function GENERATERANDOMRULESET(max_rules, max_conditions, action_set)
2:   ruleSet ← new RuleSet()
3:   numRules ← RANDOM(1, max_rules)
4:   for i ← 1 to numRules do
5:     rule ← new Rule()
6:       numConditions ← RANDOM(1, max_conditions)
7:       for j ← 1 to numConditions do
8:         condition ← new Condition()
9:         Create a condition with random parameters
10:        ADDCONDITION(rule, condition)
11:       rule.action ← RANDOMCHOICE(action_set)
12:       ADDRULE(ruleSet, rule)
13:   return ruleSet

Algorithm 2 Rule-Set Crossover

Require: parent1, parent2: The two-parent RuleSets
Require: config: Hyper-parameters for crossover
Ensure: Returns an offspring RuleSet derived from two parents

1: function RULESETCROSSOVER(parent1, parent2)
2:   Create an empty offspring RuleSet
3:   Filter inactive rules from parent1 and parent2  ▶ Keep only rules with times_applied > 0
4:   crossoverStrategy ← RANDOMCHOICE(\{'single_point', 'uniform', 'combine_rules'\})
5:   if crossoverStrategy == 'single_point' then
6:     offspring ← SINGLEPOINTCROSSOVER(parent1, parent2)
7:   else if crossoverStrategy == 'uniform' then
8:     offspring ← UNIFORMCROSSOVER(parent1, parent2)
9:   else
10:     offspring ← COMBINERULES(parent1, parent2)  ▶ crossoverStrategy is 'combine_rules'
11:   if offspring is invalid or empty then
12:     offspring ← randomly generated RuleSet
13:   offspring ← PRUNERULES(offspring)  ▶ Remove redundant rules from offspring
14: return offspring
Algorithm 3 Single-Point Crossover

Require: \(parent_1, parent_2\): The two-parent RuleSets
Require: config: Hyper-parameters for crossover
Ensure: Returns an offspring RuleSet derived through single-point crossover

1: function SINGLEPOINTCROSSOVER(parent\(_1\), parent\(_2\))
2: \(crossoverIndex \leftarrow \) A random number between 1 and \(\min(\text{Size}(parent_1), \text{Size}(parent_2))\)
3: maxLength \(\leftarrow\)
4: Create an empty offspring RuleSet
5: for \(i \leftarrow 1\) to \(crossoverIndex\) do
6: Append \(parent_1[i]\) to offspring
7: for \(i \leftarrow crossoverIndex + 1\) to \(\text{Size}(parent_2)\) do
8: Append \(parent_2[i]\) to offspring
9: return offspring

Algorithm 4 Uniform Crossover

Require: \(parent_1, parent_2\): The two-parent RuleSets
Require: config: Hyper-parameters for crossover
Ensure: Returns an offspring RuleSet derived through uniform crossover

1: function UNIFORMCROSSOVER(parent\(_1\), parent\(_2\))
2: maxLength \(\leftarrow \max(\text{Size}(parent_1), \text{Size}(parent_2))\)
3: Create an empty offspring RuleSet
4: for \(i \leftarrow 1\) to \(\text{maxLength}\) do
5: if \(i \leq \text{Size}(parent_1)\) and \(i \leq \text{Size}(parent_2)\) then
6: offspringRule \(\leftarrow \) RANDOMCHOICE(\([\text{parent}_1[i], \text{parent}_2[i]]\))
7: else if \(i \leq \text{Size}(parent_1)\) then
8: offspringRule \(\leftarrow\) \(parent_1[i]\)
9: else
10: offspringRule \(\leftarrow\) \(parent_2[i]\)
11: Append offspringRule to offspring
12: return offspring

Algorithm 5 Combine-Rules Crossover

Require: \(parent_1, parent_2\): The two-parent RuleSets
Ensure: Returns an offspring RuleSet, enhancing \(parent_2\) with conditions from \(parent_1\)

1: function COMBinerules(parent\(_1\), parent\(_2\))
2: offspring \(\leftarrow\) Clone(parent\(_2\)) \(\triangleright\) Initialize offspring as a copy of parent\(_2\)
3: selectedRule \(\leftarrow\) RANDOMSELECT(parent\(_1\)) \(\triangleright\) Randomly select a rule from parent\(_1\)
4: selectedAction \(\leftarrow\) GETACTION(selectedRule) \(\triangleright\) Extract action of the selected rule
5: for all rule in offspring do
6: if GETACTION(rule) \(\equiv\) selectedAction then
7: APPENDCONDITIONSfrom(rule, selectedRule) \(\triangleright\) Append conditions to rules in offspring with the same action
8: return offspring
Algorithm 6 Mutate Conditions

Require: condition: The Condition object to be mutated
Require: config: Hyper-parameters for condition mutation
Require: states: Dictionary of possible input features
Ensure: Returns a mutated Condition object

1. function MutateCondition(condition)
2. Randomly determine mutation type based on config
3. if selected mutation is "perturbation" then
4.    Perturb the value of condition
5. else if selected mutation is changing "operator" then
6.    Randomly change the 'operator' of condition
7. else if selected mutation is modifying the "lag" then
8.    Randomly modify the lag of condition
9. else if selected mutation is changing states then
10.   Randomly change the states involved in condition
11. Apply any necessary adjustments to maintain logical consistency
12. return the mutated condition

Algorithm 7 Mutate Rule

Require: rule: The Rule object to be mutated
Require: config: Hyper-parameters for rule mutation
Require: actions: Dictionary of possible actions or classes
Ensure: Returns a mutated Rule object

1. function MutateRule(rule)
2. Randomly select mutation aspect based on config
3. if selected mutation is changing "action" then
4.    rule.action ← MutateAction(rule.action)
5. else if selected mutation is changing "certainty" then
6.    rule ← MutateCertainty(rule)
7. else if selected mutation of a "condition" then
8.    Randomly determine specific condition mutation strategy based on config
9. if selected condition strategy is in place mutation then
10.   condition ← PickRandomCondition(rule) ▷ Picks a random condition from rule
11.   rule ← MutateCondition(rule, condition)
12. else if selected condition strategy is removal & multiple conditions exist then
13.   condition ← PickRandomCondition(rule) ▷ Picks a random condition from rule
14.   rule ← RemoveCondition(rule, condition)
15. else
16.   condition ← CreateRandomCondition()
17.   rule ← AddCondition(condition)
18. Apply any necessary adjustments to maintain logical consistency
19. return rule
Algorithm 8 Mutate Rule Set

Require: ruleset: The RuleSet object to be mutated
Require: config: Hyper-parameters for ruleset mutation
Require: actions: Dictionary of possible actions or classes
Ensure: Returns a mutated RuleSet object

1. function MutateRuleSet(ruleset)
2. Randomly select mutation type based on config
3. if selected mutation is “addition” then
4. Create a random new rule with actions from actions
5. Append the new rule to ruleset
6. else if selected mutation is “removal” then
7. if ruleset has more than one rule then
8. Randomly remove an existing rule from ruleset
9. else if selected mutation is “modify default action” then
10. Randomly select a new action from actions
11. Set this action as the new default action for ruleset
12. else if selected mutation is “reorder” then
13. Randomly shuffle the order of rules in ruleset
14. return ruleset

Algorithm 9 Prune Redundant Rules and Conditions

Require: rules: A list of Rule objects
Ensure: rules is pruned of redundant rules, redundant conditions, tautologies, and falsehoods

1. function PruneRules(rules)
2. for each rule in rules do
3. Identify and remove any tautological or false conditions within rule
4. for each condition in rule.conditions do
5. if condition is redundant or a tautology then
6. Remove condition from rule
7. if rule is now redundant or contains only tautologies or falsehoods then
8. Remove rule from rule_set
9. for each pair of rule_i, rule_j in rules do
10. if rule_i and rule_j are logically equivalent then
11. Remove rule_j from rules
12. return rules ★ Return the pruned set of rules
B Implementation of the AIC and BIC metrics

Algorithm 10 specifies the simplified calculation of AIC and BIC for the special conditions of the experiments: The classifiers always predict one class with certainty (probability of one) and assign a probability of zero to all other classes. Under these conditions, AIC and BIC can be calculated using the model’s accuracy. Further, to make the method practical, a small value (1e-15) is used for the logarithm of zero (which is mathematically undefined), similarly to the Scikit-learn library’s log_loss function (found in sklearn.metrics.log_loss).

Algorithm 10 Calculate AIC and BIC under the special conditions of the experiments

1: procedure CalculateAICBIC(accuracy, n_parameters, n_data_points)
2: n_incorrect ← (1 − accuracy) × n_data_points
3: small_prob ← 1 × 10^-15
4: log_likelihood ← n_incorrect × log(small_prob)
5: AIC ← 2 × n_parameters − 2 × log_likelihood
6: BIC ← log(n_data_points) × n_parameters − 2 × log_likelihood
7: return AIC, BIC
C Balancing accuracy vs. complexity

Accuracy and complexity are conflicting properties in that accuracy can often be improved by making the model more complex. It is therefore illuminating to show the trade-offs that the different methods pick along these metrics. As illustrated in Figure 16, EVOTER discovers non-dominated solutions in every domain. Although in these four datasets non-dominance follows immediately because EVOTER is no less accurate and always much simpler, this result suggests even in domains where accuracy may be lower when the models are simpler, EVOTER’s solutions may provide a useful and practical tradeoff.

Fig. 16. Comparison of accuracy vs. complexity trade-offs. Each point represents the average of ten runs as shown in Tables 1 – 4. In each dataset, the EVOTER solution is non-dominated by the other solutions, that is, EVOTER is either to the right or above the others, or both. It therefore finds a superior balance between accuracy and simplicity.