RuleExplorer: A Scalable Matrix Visualization for Understanding Tree Ensemble Classifiers

Zhen Li, Weikai Yang, Jun Yuan, Jing Wu, Changjian Chen, Yao Ming, Fan Yang, Hui Zhang, Shixia Liu

Abstract—The high performance of tree ensemble classifiers benefits from a large set of rules, which, in turn, makes the models hard to understand. To improve interpretability, existing methods extract a subset of rules for approximation using model reduction techniques. However, by focusing on the reduced rule set, these methods often lose fidelity and ignore anomalous rules that, despite their infrequency, play crucial roles in real-world applications. This paper introduces a scalable visual analysis method to explain tree ensemble classifiers that contain tens of thousands of rules. The key idea is to address the issue of losing fidelity by adaptively organizing the rules as a hierarchy rather than reducing them. To ensure the inclusion of anomalous rules, we develop an anomaly-biased model reduction method to prioritize these rules at each hierarchical level. Synergized with this hierarchical organization of rules, we develop a matrix-based hierarchical visualization to support exploration at different levels of detail. Our quantitative experiments and case studies demonstrate how our method fosters a deeper understanding of both common and anomalous rules, thereby enhancing interpretability without sacrificing comprehensiveness.

Index Terms—Tree ensemble classifier, model reduction, hierarchical visualization

I. INTRODUCTION

Tree ensemble classifiers, such as random forests and boosted trees, are popular machine learning models [28], [36], [72]. These models make multi-class predictions using a large number of rules (Fig. 1), often exceeding tens of thousands [28]. On the one hand, the large set of rules makes these models powerful and has led them to become winning solutions for many machine learning competitions [70]. On the other hand, while individual rules are simple and easily understandable, the overall interpretability of these models is compromised by the overwhelming number of rules. Therefore, there is a trade-off between the performance and interpretability in these tree ensemble classifiers. Although performance is the main goal in machine learning competitions, the ability of domain experts to understand and explain decisions is crucial in high-stakes fields such as healthcare, law enforcement, and financial forecasting.

Several works have been proposed to enhance the interpretability of tree ensemble classifiers [43], [69]. However, the scalability issue becomes the main challenge when managing rule sets that extend to tens of thousands or even more. A widely adopted strategy to address the scalability issue of a complex model is model reduction [20], which simplifies the original model by using fewer rules to provide better interpretability. However, there are two technical challenges in using model reduction techniques directly.

First, in order to explain the overall behavior of the original model with fewer rules, model reduction methods usually fo-cus on extracting common rules that cover most of the samples while ignoring less frequent, anomalous ones. However, in real-world applications, these anomalous rules often expose potential flaws in the model and thus require further diagnosis for a complete understanding of the model behavior [34]. For example, a credit card application may be approved based on seemingly unrelated conditions, such as having a driver's license. However, the actual reason is that other applicants who meet these conditions typically have a good credit history and a certain income level. Previous research has shown that anomaly-biased sample selection methods effectively preserve the overall distribution and highlight anomalous samples for analysis [62], [65]. As a result, a promising solution is an anomaly-biased model reduction method that preserves both common rules and anomalous rules.

Second, simply using model reduction inevitably results in a loss of fidelity to the original model. This causes discrepancies between the behavior of the reduced model and that of the original one. Therefore, only examining the reduced rule set after model reduction may lead to problems in analyzing the whole model. For example, an abnormal behavior identified in the reduced model may be difficult to trace back to rules in the original model without access to the whole rule set. To address this problem, it is essential to find an effective way to examine all the rules of the original model. Hierarchical visualization has been shown effective in the exploration of large-scale data [42]. By re-organizing rather than discarding the rules that are not preserved in model reduction, the issue of losing fidelity can be effectively mitigated. However, the pre-built hierarchies commonly used in hierarchical visualization methods [11], [13], [37], [62], [67] lack the flexibility to accommodate different user preferences during exploratory analysis. This flexibility is often needed in real-world applications. For example, in the credit card approval process, users frequently examine different subsets of rules by constraining the values of different attributes (e.g., *Income*, *Years Employed*, etc.) in different ranges. As a result, there is a need to flexibly build the hierarchy for different analysis needs.

Based on the above analysis, we develop RuleExplorer, a scalable visual analysis tool designed to support interactive analysis of rule sets extracted from tree ensemble classifiers. The scalability to manage tens of thousands of rules is achieved by combining anomaly-biased model reduction and hierarchical visualization. Starting from all the rules, RuleExplorer initially utilizes model reduction to extract the rules for the first level in the hierarchy. Subsequently, it dynamically builds the hierarchy in a top-down manner based on user selections. At every level, a balance between
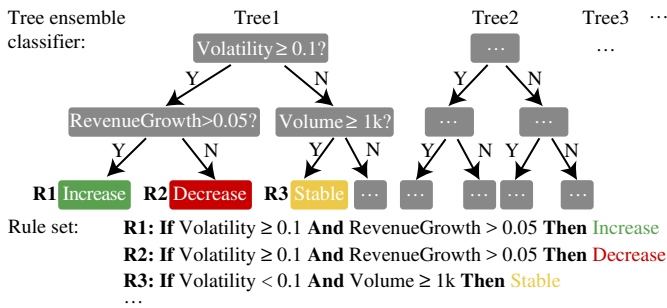
Fig. 1. An illustration of a tree ensemble classifier used to predict stock price movement with labels "increase", "decrease", and "stable."

representativeness and diversity is maintained by preserving both common and anomalous rules. Synergized with this dynamic rule hierarchy organization, a matrix-based hierarchical visualization is developed to support the exploration of rules at different detail levels. In this visualization, rows and columns correspond to rules and attributes, respectively. The confidence, coverage, and anomaly scores of each rule are also displayed. This setup allows users to quickly understand the common behavior of the model and identify rules that exhibit abnormal behavior. Finally, users can zoom in to explore more rules with similar behavior. Users can also examine the samples associated with certain rules to diagnose potential flaws in the model. The coordinated analysis between rules and samples further enhances the understanding of model behavior.

The effectiveness of the anomaly-biased model reduction is demonstrated through quantitative evaluation. Two case studies further demonstrate the effectiveness of RuleExplorer in analyzing the decision-making process of random forest models and gradient boosted tree models. Note that while we focus on tree ensemble classifiers, the methods are also applicable to other models through model surrogate techniques. In summary, the main contributions of this work are:

- An anomaly-biased model reduction method that preserves both common and anomalous rules.
- A matrix-based and dynamically constructed hierarchical visualization that supports the exploration of a large-scale rule set at different detail levels.
- A visual analysis tool that facilitates the understanding, exploration, and validation of tens of thousands of rules.

II. RELATED WORK

A. Model Surrogate

Our model reduction method relates to a category of explanation methods that utilize surrogate models. These methods use a simpler and more interpretable model to approximate and explain complex models [10], [17], [40], [49], [56], [66].

Most model surrogate methods are model-agnostic. These methods derive a surrogate model from any given machine learning model by approximating the input-output relationship of the model. While these methods have good generalizability, they inevitably neglect the inner decision logic in tree ensemble models, such as node splitting and majority voting. As a result, these methods cannot precisely reveal how predictions are made.

For tree ensemble models, the surrogate methods proposed in [38], [59] leverage the characteristics of tree ensembles for more accurate approximations. Meinshausen [38] selected representative rules from a random forest by solving a quadratic programming problem with linear inequality constraints. Vidal *et al.* [59] constructed an equivalent single decision tree from a given tree ensemble to improve model interpretability. Although these methods successfully preserve common rules that well summarize the behavior of the original model, they inevitably lose fidelity because other rules are discarded, especially those anomalous rules that are important for revealing potential flaws in the model. In contrast, our method reorganizes a tree ensemble into a hierarchy of rules to mitigate the issues of losing fidelity, and uses anomaly-biased model reduction to maintain a balance between the common rules and the anomalous rules at each hierarchical level.

B. Visualization for Model Analysis

The visualization community has developed a number of methods and systems to visually interpret, analyze, and diagnose machine learning models. Initial efforts and their subsequent variations aim to illustrate a variety of performance measures such as precision, recall, and between-class confusion [12], [26], [48]. Most recent efforts focus on explaining the inner workings of a machine learning model based on architecture exploration and the reasoning of its prediction.

Architecture visualization methods focus on explaining the inner workings of the model by analyzing specific components in the model [63], such as neurons and kernels in convolutional neural networks [5], [35], [57], hidden states in recurrent neural networks [8], [33], [39], [51], [54], attention heads in transformers and other attention-based models [19], [53], or tree structures in decision trees [36], [41], [58]. These visualizations are designed to help model developers analyze and diagnose machine learning models. However, for domain experts who have little knowledge of machine learning, such as financial analysts or doctors, the complex workflows and model-architecture-specific designs are of little help for their understanding and utilizing the models. To support these domain experts, our work focuses on explaining rules learned from tree ensemble classifiers, as rules are more understandable to them. If offered the right tool, they can make sense of the learned rules and validate the rules of interest.

C. Rule Visualization

Existing efforts to visualize rules can be categorized as single-rule-oriented or multiple-rule-oriented. Single-rule-oriented visualization [9], [27] focuses on visualizing samples covered by a single rule, while multiple-rule-oriented visualization seeks to visually illustrate interactions and relationships between multiple rules. Our work aligns with the latter. Common methods for multiple-rule-oriented visualization include the node-link diagram and the matrix diagram. Other compact representations such as treemaps [41], icicle plots [22], and sunburst diagrams [60] are also used to handle a larger number of rules. While these methods are effective at presenting an overview of the rules, they may not always provide the level of
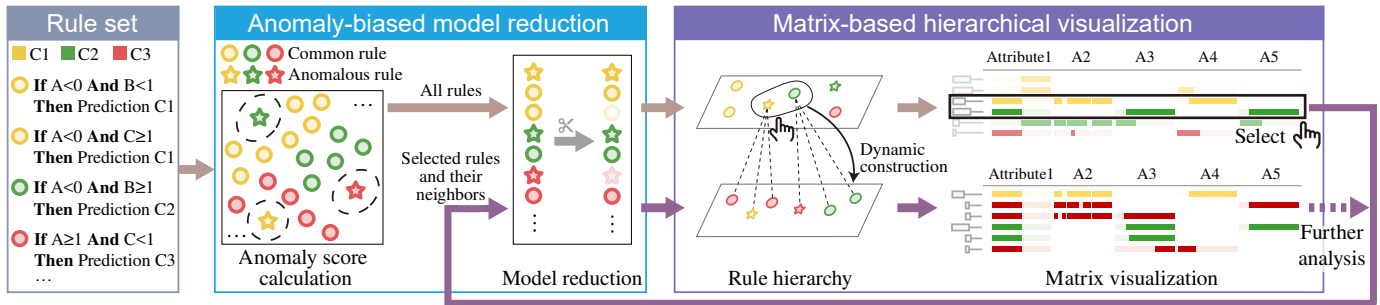
Fig. 2. RuleExplorer overview: Given a set of rules from a tree ensemble classifier, the anomaly-biased model reduction calculates the anomaly score for each rule and then extracts the representative rules. Next, the representative rules are fed into the matrix-based hierarchical visualization for exploration and analysis, where a rule hierarchy is dynamically built based on user selections. The brown color indicates the analysis of the representative rules at the top level, and the purple color indicates the iterative analysis subsequently.

details necessary for an in-depth analysis of individual rules. Therefore, we focus on introducing the first two methods.

The node-link diagram [31], [41], [55], [58] naturally represents the decision tree structure or summarizes multiple rules by aggregating the same conditional statement. For example, BaobabView [58] employs a node-link diagram to visualize the rules of decision trees and further encodes the split points and data flows in the tree. TreePOD [41] extends this design by utilizing a treemap to display the distribution of samples covered by different tree nodes. SuRE [66] summarizes the entire random forest into a node-link diagram while also retaining useful logical relationships between different rules. While the node-link diagrams effectively illustrate relationships between rules, they struggle to remain clear and manageable when displaying tens of thousands of rules. The matrix diagram [40], [43], [68] is a natural graphical representation of a list of rules. RuleMatrix [40] visualizes a falling rule list using a matrix diagram with rows as rules, columns as attributes, and cells as conditions using histogram encoding. DeforestVis [10] provides attribute-based explanations using simplified decision trees, allowing users to incrementally generate them to explore the trade-off between complexity and fidelity. ExplainableMatrix [43] also employs a matrix diagram, but distinguishes itself by using colored rectangles in matrix cells to encode the conditions for different classes. However, these methods show all the rules in one view without filtering, and still encounter the scalability issue in actual scenarios.

To address the scalability issue, we build a matrix-based hierarchical visualization to show representative rules at different levels to help practitioners understand, explore, and validate tree ensemble classifiers.

III. DESIGN OF RULEEXPLORER

A. Design Goals

We designed RuleExplorer in collaboration with eight domain experts ($E_1 - E_8$) from banks and quantitative trading companies. They utilize tree ensemble classifiers for credit card approval and stock trading. All of them are not co-authors of this work. We distilled the design requirements based on discussions with these domain experts and a literature review.
R1. Hierarchically organize large-scale rule sets. In real-world applications, a tree ensemble classifier often contains tens of thousands of rules, which makes it impractical to examine each rule individually. Although model reduction methods can extract representative rules to explain model behavior, they inevitably lose fidelity [38], [40], [66]. Domain experts have expressed the need to explore the entire rule set for analysis. Therefore, reorganizing rather than reducing the rules is preferred. A hierarchical reorganization is a promising solution due to its effectiveness in the analysis of large-scale data [52]. Moreover, during the reorganization, both common and anomalous rules should be preserved at each level to ensure a comprehensive understanding and diagnosis of potential problems.

R1.1 Preserve common rules to enable the understanding of the overall model behavior. All the experts emphasized the importance of examining the common rules learned by the model to establish trust. As E_2 said, once he understood the common rules used by the model in credit card approval, he could trust the model predictions. This would relieve his burden of examining each application individually. In addition, E_3 expressed concerns that model errors could result in financial losses. Thus, he would like to open the "black box" model and understand the overall model behavior from the common rules before trusting the decisions made by the model.

R1.2 Preserve anomalous rules to enable the diagnosis of potential model flaws. Anomalous rules indicate abnormal model behavior deviating from the common ones, which reflect potential model flaws. E_2 stated, "Anomalous rules usually cover only a small number of samples, which may be problematic samples and deserve further analysis." E_1 was also interested in examining anomalous rules, "If a rule approves applications, but with conditions suggesting low repayment ability and instability, it is likely flawed." All the experts believed that examining these rules helps uncover blind spots in the model and opens up opportunities for model improvement.

R2. Effectively explore and analyze the rules. The analysis of a large-scale rule set requires efficiently identifying the rules of interest and analyzing them in context. This leads to three major requirements. First, an intuitive and informative overview of the rule set is essential to help users quickly gain insights and start their analysis. Second, support for dynamic hierarchical exploration is necessary to enable detailed navigation through various levels of detail and facilitate the analysis

of different rule subsets. Third, rich interactions should be provided to allow for an in-depth analysis of rules concerning their utilized attributes and covered samples. To this end, a scalable and flexible interactive environment is crucial for effective exploration and analysis of rules.

R2.1 Present rule sets using an intuitive and informative visual representation. To effectively visualize a hierarchically organized rule set, a requirement is to both reveal the hierarchical structure and present detailed information on attributes within the rules. The commonly used node-link-based [64] or scatterplot-based [62] hierarchical visualizations focus on the former but require additional interactions to access detailed information, which is not ideal for analyzing rule sets. For example, E_1 expressed a desire for a quick overview that allows immediate identification of important rules and attributes. Moreover, simple and intuitive visual representations are always preferred because they facilitate quicker comprehension and reduce cognitive load. All these call for visual representations that are both intuitive and informative to meet the needs of domain experts.

R2.2 Cater to different analysis preferences. Users may have different analysis needs and focus on different subsets of rules during the exploration. For example, E_7 mentioned that in the overall analysis of stocks, he would pay more attention to the attributes directly related to price changes, such as moving average price and price volatility. When he focused on low-cap stocks, he would like to examine the attributes related to stock liquidity, such as turnover ratio and market depth. With the different analysis needs, he zoomed into different subsets of rules. Compared to a static hierarchy, which organizes the rules with a pre-defined criterion, a dynamic hierarchy is better at catering to users' different preferences [14], [23], [71].

R2.3 Locate rules of interest and relate them to training samples. When exploring the entire rule set or specific rule subsets, experts expressed their interest in analyzing rules based on various properties. E_1 was particularly interested in how the model would handle the borderline cases in the credit card application. Thus, he wanted to examine the low-confidence rules that were mostly relevant to these cases. To facilitate this, we provide a set of interactions, such as sorting rules by different properties, to help identify the rules of interest. Additionally, domain experts emphasized the importance of knowing the samples from which a rule was learned. For example, before making investment decisions based on some rules, E_7 wanted to validate these rules by examining the training samples that contributed to the learning of these rules. This highlights the need to relate rules to training samples.

B. System Overview

Guided by the aforementioned design requirements, we developed RuleExplorer to help domain experts explore and analyze the large-scale rule sets in tree ensemble classifiers. As illustrated in Fig. 2, RuleExplorer consists of two modules: anomaly-biased model reduction and matrix-based hierarchical visualization. The anomaly-biased model reduction module first calculates an anomaly score for each rule. Based on the anomaly scores, it then extracts representative rules through

model reduction for subsequent visualization. The extracted representative rules not only approximate the behavior of the input rule set but also preserve anomalous rules. The matrix-based visualization module visualizes the representative rules in a matrix and supports users in exploring the rules by dynamically constructing the rule hierarchy based on user selections. More specifically, users can select some rules of interest in the matrix for a more detailed examination. Upon selection, the selected rules and their neighborhood are input into model reduction to extract the representative rules at the next level in the hierarchy. The matrix visualization is updated accordingly with these new representative rules. Users can continue to select and analyze these rules in a similar way.

IV. ANOMALY-BIASED MODEL REDUCTION

The developed anomaly-biased model reduction method aims to enable both a comprehensive explanation of the overall model behavior and the identification of potential flaws (**R1.1**, **R1.2**). To this end, we first employ logistic regression to calculate the anomaly scores for all the rules in the model. Then, we formulate the anomaly-biased model reduction as a subset selection problem that extracts the rules to approximate the behavior of the given rule set on their covered samples, while prioritizing the anomalous rules with high scores.

A. Anomaly Score Calculation

To identify the anomalous rules, we need to calculate an anomaly score for each rule. This is achieved by converting the rules into feature representations and then employing logistic regression for anomaly detection.

Rule representation. In a tree ensemble classifier, each rule is represented by the conjunction of conditions on attributes and a corresponding prediction. Inspired by Zhao *et al.* [69], we vectorize the condition on each attribute and concatenate them to obtain the feature representation of a rule. There are two types of attributes. First, the condition on a categorical attribute is represented by a vector indicating the distribution of the categories that a sample can fall into. For example, assume that there are three categories F_1, F_2, and F_3 for a categorical attribute F, with 10, 20, and 30 samples in each category, respectively. Then, a condition $F \in \{F_1, F_2\}$ can be represented by $[10/(10 + 20 + 30), 20/(10 + 20 + 30), 0] = [1/6, 2/6, 0]$. Second, the condition on a numerical attribute is represented by a two-dimensional vector indicating the lower and upper bounds of the value range that satisfies the condition. To enable a fair comparison between conditions on different attributes with varying scales and distributions, the quantile normalization [6] is used to transform all the attribute distributions to the uniform distribution in the range $[0, 1]$.

Anomaly score calculation. With the feature representations and the corresponding predictions of the rules, we fit a logistic regression model. With this model, given a rule r_j, we obtain its probability p_j leading to its corresponding prediction. Accordingly, we calculate its anomaly score as $s_j = 1 - p_j$, which measures the deviation of r_j from the majority of rules [44]. As anomalous rules exhibit behavior deviating from the common rules, they have higher anomaly scores.
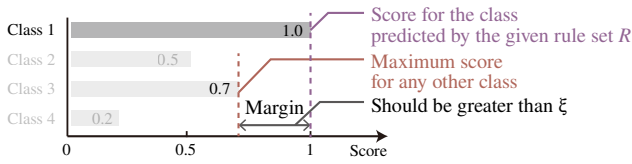
Fig. 3. An example illustration of multi-class hinge loss.

Fig. 4. Dynamic rule hierarchy construction between consecutive levels.

B. Model Reduction

Model reduction can be formulated as a subset selection. Given a rule set $R = \{r_j\}_{j=1}^m$ associated with their anomaly scores $\{s_j\}_{j=1}^m$ and a set of samples $X = \{x_i\}_{i=1}^n$ belonging to $C \geq 2$ classes, our method extracts a subset of representative rules R^* from R by solving the optimization problem:

$$\underset{\{z_j\}_{j=1}^m}{\text{argmin}} \quad \frac{1}{n} \sum_{i=1}^n \mathcal{L}(\hat{y}_i, \ell_i) - \lambda \frac{1}{M} \sum_{j=1}^m z_j s_j,$$

$$\text{s.t.} \quad z_j \in \{0,1\}, \ \sum_{j=1}^m z_j \leq M. \tag{1}$$

Here, $\hat{y}_i = [\hat{y}_{i,c}|c = 1, \cdots, C]$ is a vector of prediction scores of x_i by the extracted rule set R^*, where $\hat{y}_{i,c}$ is the prediction score of x_i to be of class c. ℓ_i is the prediction label of x_i by the given rule set R. $\mathcal{L}(\hat{y}_i, \ell_i)$ is the loss function that measures the prediction difference between the extracted rule set R^* and the original rule set R. z_j is the indicator of whether the rule r_j belongs to R^*. The first term ensures the consistency between the predictions of the given rules and the extracted rules. The second term prioritizes the selection of anomalous rules by maximizing the anomaly scores of the extracted rules. λ is a factor to balance the effect of the two terms. M is the maximum size of the extracted rule subset R^*.

Next, we introduce how to calculate $\mathcal{L}(\hat{y}_i, \ell_i)$. To encourage that R^* and R make consistent predictions, the idea is to increase $\hat{y}_{i,\ell_i}$, *i.e.*, the prediction score of x_i to be of class ℓ_i, while decreasing other $\hat{y}_{i,c}$ scores ($c \neq \ell_i$). Therefore, we adopt multi-class hinge loss [16]:

$$\mathcal{L}(\hat{y}_i, \ell_i) = max(\xi - (\hat{y}_{i,\ell_i} - \max_{c \neq \ell_i} \hat{y}_{i,c}), 0). \tag{2}$$

For a specific sample x_i, as shown in Fig. 3, the multi-class hinge loss encourages the score for class ℓ_i to be greater than any other class by a margin of at least ξ. This helps improve the generalization ability of the extracted rules [50]. To reduce the user burden of parameter adjustments and improve ease of use, we determine ξ and λ using a grid search. This search spans values from from 0.1 to 1.0, with a step size of 0.1, effectively balancing fidelity and anomaly scores. Specifically, we first set λ to 0 and searched for ξ that achieved the highest fidelity, then adjusted λ to improve anomaly scores while keeping fidelity loss within 1%. M is set to 80 based on the screen constraints and suggestions from the experts, as they prefer to examine as many rules as possible while maintaining visual clarity for rule analysis. We also conducted a sensitivity analysis of M in the supplemental materials to justify this choice.

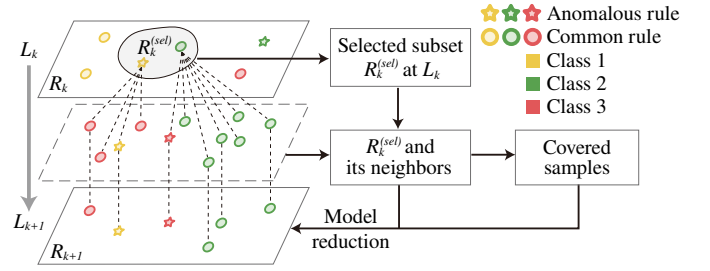Eq. (1) is an integer linear programming problem, which is NP-hard and time-consuming to solve. Even advanced com-mercial solvers cannot solve it within reasonable time limits due to the extremely large basis matrix and overwhelming memory consumption resulting from thousands of rules and samples. To solve it efficiently, we relax the z_j to continuous variables. The relaxed problem becomes a linear programming problem and can be solved in polynomial time [30]. Then, we adopt deterministic rounding to discretize z_j and determine which rules belong to R^*.

V. MATRIX-BASED HIERARCHICAL VISUALIZATION

In this section, we first introduce the dynamic construction of the rule hierarchy, and then present the design and implementation of the matrix-based hierarchical visualization that supports the exploration of rules.

A. Dynamic Hierarchy Construction

Given the large-scale rule set extracted from a tree ensemble classifier, we need to organize the rules into a hierarchy to support exploration at different detail levels. A simple way is to construct a static hierarchy with different numbers of rules extracted at each level to approximate the original model. However, such a practice lacks the flexibility to identify rules according to different analysis requirements. For example, in a stock price prediction model, most companies are predicted by the rules considering the historical price trend and volatility, while low-cap companies are more accurately predicted by the rules considering stock liquidity. Therefore, it is desirable to build the rule hierarchy dynamically to better accommodate users' preference during exploration (**R2.2**).

In particular, users start the exploration at the top level L_0. The representative rules R_0 are displayed, which are extracted from all rules to approximate the model predictions on all samples, while the other rules are hidden. Then, users can iteratively select rules of interest and zoom into their neighbor-hood for examination. During this process, the rule hierarchy is dynamically constructed. As illustrated in Fig. 4, the con-struction of level L_{k+1} from level L_k consists of three steps: **Selecting a rule subset $R_k^{(sel)}$ from R_k** and zooming in. More representative rules should be displayed to illustrate the model behavior in the neighborhood of the selected rule subset $R_k^{(sel)}$. **Determining the neighborhood of $R_k^{(sel)}$** is the key in this process. At the level L_k, the hidden rules are assigned to their nearest rules in R_k based on the weighted Euclidean distance between the feature representations of rules. Following the previous work [69], the weight of each attribute in the feature
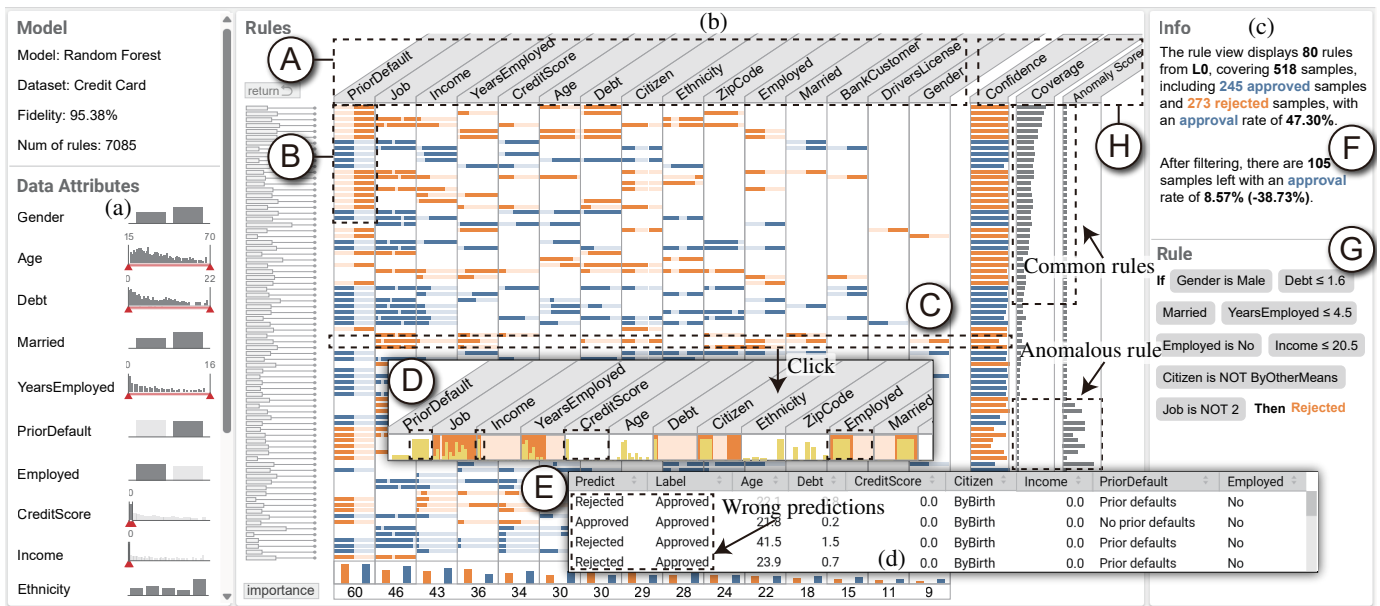
Fig. 5. RuleExplorer: (a) *attribute view* shows the attribute distribution and enables sample filtering; (b) *matrix view* shows the representative rules at a certain level of the rule hierarchy; (c) *info view* shows the overall statistics of the displayed rules and samples; (d) *data table* lists the samples covered by the displayed rules.

representation is calculated as the frequency of this attribute being used in the rules at this level, including representative and hidden ones. The neighborhood of $R_k^{(sel)}$ consists of the rules in $R_k^{(sel)}$ and the hidden rules assigned to them.

Extracting R_{k+1} using model reduction to approximate the model predictions on the samples covered by the neighborhood of $R_k^{(sel)}$. In order to keep the mental map between L_k and L_{k+1}, we ensure that the user-selected rule subset $R_k^{(sel)}$ is displayed at L_{k+1}.

B. Visualization Design

As shown in Fig. 5, the main component of RuleExplorer is the *matrix view* (Fig. 5(b)), which uses an enhanced matrix visualization to provide an intuitive and informative overview of representative rules (**R2.1**). We adopt the matrix visualization because of its effectiveness in comparing and analyzing multiple rules. Moreover, studies have shown that matrix visualizations outperform node-link diagrams and other techniques in terms of readability and scalability, particularly when dealing with large and complex datasets [1]. However, existing methods focus only on numerical attributes, so we extend the matrix visualization to accommodate categorical attributes. As shown in Fig. 6(a), a rule is an **If-Then** statement that includes 1) a conjunction of conditions on attributes (*e.g.*, Income $\leq$ 20 **AND** Years $\leq$ 4.5 **AND** Citizenship is not Other) and 2) a prediction label (*e.g.*, Rejected). In the matrix visualization, each row represents a rule, and its color hue encodes the prediction label of this rule. Each column represents an attribute. If an attribute is used in a condition of this rule, the corresponding cell uses dark-shaded and light-shaded horizontal areas to indicate the value ranges that satisfy and dissatisfy the condition, respectively (Fig. 6(b)). If an attribute is not used, the corresponding
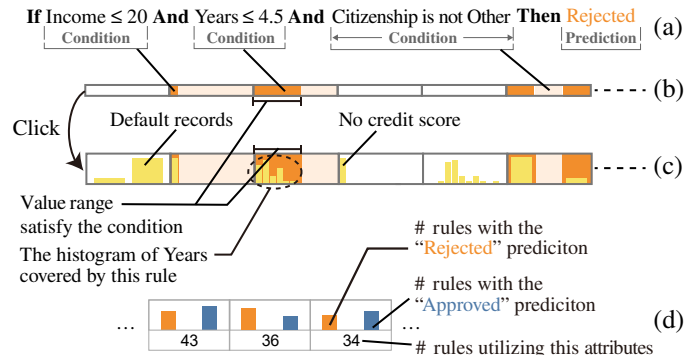

Fig. 6. The design of the matrix visualization. A rule (a) is visualized as a row (b) in the matrix, which can be expanded (c) to display sample distributions on different conditions. Additional rows (d) are appended at the bottom of the matrix to show cumulative statistics for each attribute.

cell is left blank. Such a design facilitates the comparison between the conditions of different rules. The detailed attribute distributions of the samples covered by a rule are displayed when clicking on the corresponding row. For example, in Fig. 6(c), the overlaid bar charts in each cell show that all the covered samples have no credit score, while most of them have default records. This information helps users verify whether a rule utilizes the appropriate attributes to make its predictions.

We also supplement the matrix with additional rows and columns that provide auxiliary information for analysis. At the bottom of the matrix, an additional row features cells that display the cumulative statistics for each attribute across the displayed rules. As shown in Fig. 6(d), each cell in this row shows the number of rules utilizing this attribute, and includes a bar chart 🔲 showing the prediction distribution of these rules. This helps users analyze the correlation of this attribute with different predictions. On the left of the matrix, a glyph

is placed next to each row. The bar length represents the number of its neighborhood at this level. When zooming in, the glyphs are also organized as an indented tree 𝄃 to show the hierarchical structure of the rules (Fig. 8D). On the right of the matrix, there are three additional columns(Fig. 5H) that show the confidence, coverage, and anomaly score of each rule. They represent the accuracy of rule prediction on its covered samples, the number of its covered samples, and the degree of its deviation from other rules, respectively.

In addition to the matrix view, RuleExplorer provides three supporting views: the *attribute view* (Fig. 5(a)), the *info view* (Fig. 5(c)), and the *data table* (Fig. 5(d)). These views are co-ordinated with each other to facilitate interactive exploration.

The *attribute view* shows the attribute distributions of data samples. To enhance data exploration, we integrate scented widgets 𝄃𝄃 [61], which provide visual cues for easy sample filtering (**R2.3**). We chose scented widgets for their intuitiveness and ease of use, making them ideal for our users.

The *info view* presents the overall statistics of the representative rules and their covered samples at the current hierarchical level. This information reduces the analysis burden on understanding model behavior during exploration (**R2.2**). By selecting a rule in the *matrix view* or applying filtering in the *attribute view*, it also presents the details of the selected rule (Fig. 5G) or the class distribution change of the filtered samples (Fig. 5F), respectively.

The *data table* lists the samples covered by the representative rules, which directly presents the relationships between rules and training samples (**R2.3**). The samples can also be sorted based on their attributes to facilitate analysis.

C. Interactive Exploration and Analysis

We provide a set of interactions to facilitate 1) navigating through the rule hierarchy; 2) analyzing rules of interest; and 3) analyzing attributes of interest (**R2.3**).

1) Navigating through the rule hierarchy: After identifying the rules of interest, users may want to compare them with their neighborhood for in-depth analysis. This is supported by a dynamically constructed rule hierarchy, within which we implement the zoom function for navigating this hierarchy. Specifically, users can click or brush the glyphs ⊏══•➝ of some rules. Then, RuleExplorer dynamically extracts their child rules in their neighborhood. Users can then zoom into the next level to examine these child rules. In the *matrix view*, the newly added rules are placed under their parent rules with their glyphs indented in comparison to those of their parent rules. After zooming, the order of attributes may change since the number of rules utilizing each attribute differs between the two levels. Some attributes may contribute more to predictions at this level. We mark the attributes with the top-3 largest increases in their orders using upward arrows ⬆ to inform users for further examination. Moreover, by clicking the "return" button ⤺ at the top-left corner of the *matrix view*, users can navigate back to the previous level and analyze other rules of interest at that level.

2) Analyzing rules of interest: At each level of the hierarchy, users first get an overall understanding of the general
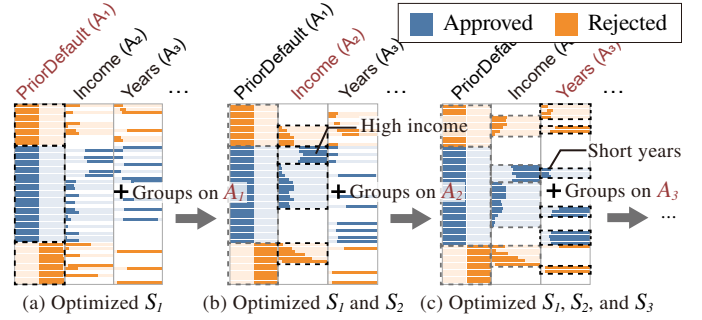

Fig. 7. An illustrative example of matrix reordering showing the formation of rule groups with (a) optimization on attribute A_1; (b) optimization on both A_1 and A_2; and (c) optimization on A_1, A_2 and A_3.

model behavior, and then identify and analyze the rules of interest through interactive exploration. By default, the rules are sorted in descending order of their coverage to highlight the common rules. They can also be sorted based on their confidence or anomaly scores by clicking on the corresponding column headers. Moreover, to compare rules with similar conditions, users can click on any attribute header to group the rules based on the condition on the attribute. For numerical attributes, the rules are sorted based on the lower and upper bounds of their value ranges. For categorical attributes, the rules with the same condition are grouped together. Rules that do not utilize this attribute are placed at the bottom.

In addition to grouping similar rules based on a single attribute, it is also desired to simultaneously consider multiple attributes. For example, compared to grouping rules only based on *PriorDefault* (Fig. 7 (a)), grouping rules with similar conditions on *PriorDefault* and *Income* together (Fig. 7 (b)) makes it more readily to discover the pattern that *"applicants with no prior default and high-income tend to be accepted"*. Therefore, to facilitate the discovery of rule patterns, we develop a matrix reordering algorithm to group rules based on multiple attributes.

The basic idea is to group rules with consistent predictions and similar conditions together. To achieve that, we maximize the number of similar conditions on the specific attributes between adjacent rules. Specifically, given n rules $R = [r_1, r_2, ..., r_n]$, and m attributes A_1, A_2, ..., A_m; we maximize S_1, S_2, ..., S_m, where S_j represents the number of similar conditions on attribute A_j between adjacent rules:

$$S_j = \sum_{i=1}^{n-1} \mathbb{I}(\text{pred}_{\pi(i)} = \text{pred}_{\pi(i+1)}) * \text{sim}_j(r_{\pi(i)}, r_{\pi(i+1)}). \quad (3)$$

Here, $\pi(i)$ is the index of the rule in R which is reordered to row i in the matrix. The first term is an indicator function, which equals 1 when the predictions of the two rules are consistent, and 0 otherwise. The second term $\text{sim}_j(r_{\pi(i)}, r_{\pi(i+1)})$ is also an indicator function that determines whether the two rules have similar conditions on the jth attribute. For categorical attributes, it is defined as whether the conditions are the same. For numerical attributes, it is defined as whether both the left and right endpoints between the two conditions

TABLE I
PERFORMANCE COMPARISON BETWEEN OUR METHOD AND THE BASELINE METHODS IN TERMS OF FIDELITY AND AVERAGE ANOMALY SCORE.

Dataset	#Classes	#Attributes	#Samples	Model	#Rules	Ours		RuleMatrix		HSR		Node harvest	
						Fidelity	AS	Fidelity	AS	Fidelity	AS	Fidelity	AS
Credit Card	2	14	690	RF	7,085	**0.9538**	**0.1309**	0.9503	0.0880	0.9264	0.0000	0.9300	0.0473
				GBT	2,126	**0.9249**	**0.1271**	0.9075	0.0735	0.8682	0.0846	0.9017	0.0042
Wine Quality	2	11	1,599	RF	28,116	**0.8725**	0.1432	0.8665	0.0689	0.7484	0.0000	0.5175	**0.1660**
				GBT	7,474	**0.8350**	**0.0974**	0.8065	0.0329	0.7368	0.0000	0.8275	0.0299
Crime	2	127	1,994	RF	12,576	**0.9178**	**0.1999**	0.9042	0.0203	0.8910	0.0000	0.8577	0.1602
				GBT	4,322	**0.9118**	**0.1152**	0.8986	0.0119	0.8368	0.0000	0.8978	0.0489
Abalone	4	8	4,177	RF	23,500	**0.8392**	**0.1247**	0.8243	0.0085	0.6695	0.0081	N/A	N/A
				GBT	11,810	**0.8967**	0.1263	0.8268	0.1231	0.5715	**0.3553**	N/A	N/A
Obesity	7	16	2,111	RF	19,490	**0.8561**	**0.1732**	0.7720	0.0138	0.7277	0.0683	N/A	N/A
				GBT	30,458	**0.9470**	**0.2309**	0.8394	0.0004	0.7233	0.0084	N/A	N/A
Dry Bean	7	16	13,611	RF	6,209	**0.9738**	**0.2829**	0.9702	0.0032	0.8726	0.0193	N/A	N/A
				GBT	52,455	**0.9151**	**0.3032**	0.9151	0.0946	0.8369	0.2360	N/A	N/A

RF: random forest. GBT: gradient boosted tree. AS: average anomaly score. Node harvest cannot handle multi-classification datasets.

differ within a user-specified threshold τ. We set $\tau = 0.1$ by default. This threshold controls whether two numerical conditions are similar enough to be placed adjacently. A smaller τ results in stricter matching criteria, while a larger τ value relaxes the criteria.

Maximizing S_j is equivalent to solving a traveling salesman problem [2], which will reorder the rules and form rule groups on the attribute A_j. Fig. 7 shows examples of rule groups formed on each attribute. However, reordering rules to optimize all S_j simultaneously is impossible as different optimization goals may conflict with each other. As experts are more concerned with patterns on important/preferred attributes, we adopt the idea of the constraint method [18], which maximizes $S_1, S_2, \ldots, S_m$ in turn. The constraints are that when maximizing S_j, the reordering of rules in a formed rule group can only be carried out within the group without crossing the group boundaries. This ensures minimal changes to $S_1, S_2, \ldots, S_{j-1}$ when maximizing S_j. We use the divide-and-conquer strategy proposed by Liu *et al.* [35] to solve this constrained traveling salesman problem.

3) Analyzing attributes of interest: As the number of attributes may exceed several hundred, it is impractical to directly display all attributes. To address this issue, we sort the attributes based on their importance and organize them into pages. On the front page, the most frequently used attributes are presented, which are important to the model prediction. Users can navigate to subsequent pages for attributes that are less frequently used according to their analysis needs. Moreover, users can pin specific attributes to the front page and adjust their positions by drag-and-drop. This facilitates the comparison of multiple attributes and understanding of their collaborative behavior. For example, Fig. 7(c) shows that by placing attributes *Income* and *Years* adjacently, users can identify patterns such as high income often leading to credit card approvals despite short working years.

VI. EVALUATION

We first conducted a quantitative experiment to evaluate the effectiveness of the anomaly-biased model reduction method. We then presented two case studies to showcase how RuleExplorer helps domain experts understand the decision logic of tree ensemble classifiers and improve them.

A. Quantitative Experiment

Datasets. We conducted the quantitative experiment on six datasets from the UCI Machine Learning Repository [3], as used in previous works [40], [66], including three binary classification datasets (Credit Card, Wine Quality, Crime) and three multiple classification ones (Abalone, Obesity, Dry Bean). The attribute numbers across these datasets range from 8 to 127, while the sample numbers range from 690 to 13,611. The diverse characteristics of these datasets enable a robust evaluation across various tasks and dataset scales.

Baseline methods. We selected three baseline methods, including two model surrogate methods, RuleMatrix [40] and HSR [66], and a state-of-the-art model reduction method, node harvest [38]. RuleMatrix generates a sequential rule list, while HSR generates a hierarchical rule set. Different from these two methods, node harvest directly extracts the rules from the original model rather than generating new ones. However, it cannot be used in multi-class classification.

Measures. We evaluated the quality of the methods from two aspects. First, *fidelity* measures the ability to preserve the behavior and predictive accuracy of the original model. It is defined as the ratio of the samples with the same predictions produced by the generated rule sets and the original model. Second, the *average anomaly score* evaluates the ability to preserve anomalous rules.

Experiment settings. To demonstrate the generalizability of our method to different tree ensemble classifiers, we conducted experiments on random forest models and gradient boosted tree models. Each dataset was partitioned into a 75% training set and a 25% test set. We first trained each model on the training set, resulting in models with rule counts ranging from 2,126 (LightGBM on credit card dataset) to 52,455 (LightGBM on drybean dataset). This wide range of rule numbers ensures a thorough evaluation across different levels of model complexity. Then, we performed model reduction to approximate the predictions on the training set. All methods are required to select 80 rules (consistent with Sec. IV) for this approximation. The fidelity was calculated on the test set, and the average anomaly score was calculated on the generated rule set. We ran five trials for each setting to reduce the effect of randomness.

Results. Our results are summarized in Table I. Our method

achieved the highest fidelity across all cases, and also achieved higher average anomaly scores than the baseline methods except in two cases. On the Wine Quality dataset, node harvest obtained a higher average anomaly score for the random forest model, while on the Abalone dataset, HSR obtained a higher average anomaly score for the gradient boosted model. However, the fidelity scores achieved in these two cases are significantly lower (0.5175 and 0.5715), indicating the corresponding rule sets failed to approximate the behavior of the original model. In contrast, our method achieved the highest fidelity and the second-highest average anomaly scores. This demonstrates that our method can strike a balance between fidelity and the preservation of anomalous rules.

B. Case Studies

We conducted two case studies to show the usefulness of RuleExplorer in explaining and diagnosing tree ensemble classifiers.

1) Credit Card Approval: In this case study, we collaborated with E_1 and E_2 to illustrate how RuleExplorer improves the understanding and diagnosis of the model to approve credit card applications. E_1 is a bank account manager with two years of work experience, who aims to increase credit card approval rates while minimizing the risk of defaults. E_2 is a data scientist responsible for developing the credit card approval model and improving its performance. Both E_1 and E_2 have participated in the interview for collecting design goals in Sec. III. During the study, E_1 interacted with the system to explore and identify potential model issues, while E_2 helped explain the underlying causes of these findings from a machine learning perspective and suggested ways to improve the model. The study used the Australian Credit Approval dataset [47], which consists of 690 samples, each representing a credit card applicant with 16 attributes and a label indicating approval or rejection. The dataset includes 307 "**approved**" samples and 383 "**rejected**" samples. These samples were randomly divided into training and test data with a ratio of 75% and 25%, respectively. To comprehensively evaluate the model, E_2 used Synthetic Data Vault [45] to augment the test data based on its distribution and labeled the new test samples in collaboration with E_1. The detailed augmentation process can be found in the supplemental material. After the augmentation, there are 500 samples in the test data. A random forest model with 200 trees and a maximum depth of 10 has been trained on this dataset with the scikit-learn library [46]. The trained model includes **7,085** rules and achieves an accuracy of 83.60%.

Overview. Initially, E_1 investigated the attributes frequently used in the extracted rules to see whether the decision-making process was reasonable. As shown in Fig. 5A, the attributes that describe income and credit status, such as *PriorDefault*, *Job*, *Income*, *Years Employed*, are frequently used and thus appear in the first few columns. This observation aligns with his experience, as these attributes are more important in credit card approval than those less frequently used attributes like *Gender* and *DriverLicenses*. Subsequently, E_1 examined the rules to see how these frequently used attributes influenced model predictions. He focused on two types of rules: common rules and anomalous rules.
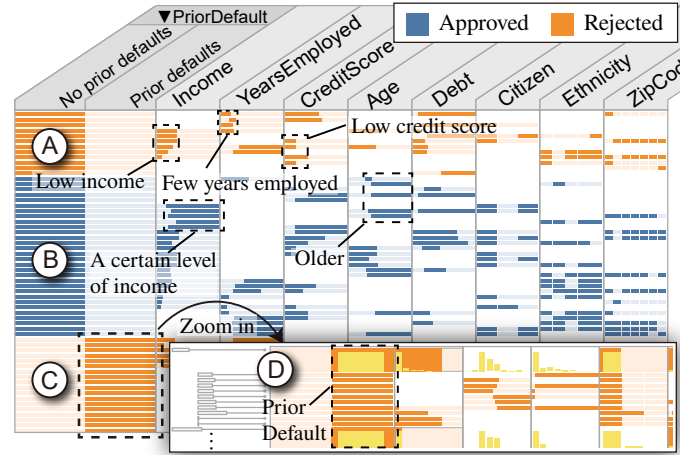

Fig. 8. Anayzing the rules that utilize the attribute *PriorDefault*: rules with no prior defaults may reject (A) or approve (B) credit card applications, while those with prior defaults consistently reject applications (C). Zoom in region C to examine the neighborhood (D).

Analyzing common rules. By default, the rules are sorted in descending order of their coverage, which facilitates the analysis of common rules (rules with higher coverage). E_1 found that most of the common rules used the attribute *PriorDefault* (Fig. 5B). He wanted to examine how this attribute influenced the model predictions. To this end, he clicked on the header of *PriorDefault*, and the matrix reordering algorithm grouped the rules based on their predictions and their conditions on this and subsequent attributes. There are three groups (Fig. 8). Rules with the condition of no prior defaults but predictions of "rejected" (Fig. 8A) linked to negative conditions such as low income, few years employed, or low credit scores. Rules with the condition of no prior defaults and predictions of "approved" (Fig. 8B) had additional positive conditions, such as a certain income level or being older, indicating a lower risk of default and a more stable status. E_1 regarded the behavior (Fig. 8A and B) as reasonable, since the condition of no prior defaults cannot directly lead to the approval, and it should be combined with additional conditions for further consideration. The third group included rules with the condition of prior defaults and predictions of "rejected" (Fig. 8C). Since there are some real-world cases where applicants with prior default records are still approved, E_1 wanted to see whether these were reflected. He selected the rules in (Fig. 8C) and zoomed into their neighborhood in the dynamic hierarchy, and found that all rules in the neighborhood had predictions of "rejected" (Fig. 8D). Consultation with E_2 clarified that a credit card approval model typically rejects all applicants with prior default records in realistic scenarios and leaves disputed cases for manual processing. To summarize, E_1 was satisfied with the behavior of the rules utilizing *PriorDefault*. Similarly, E_1 analyzed other frequently used attributes, including *Job*, *Income*, *Years Employed*, and acknowledged their proper behavior in making predictions.

E_1 continued to examine common rules and noticed a rule that had significantly lower confidence than the rules with similar coverage (Fig. 5C). The low confidence indicated that this rule covered many mispredicted samples. The *data table*
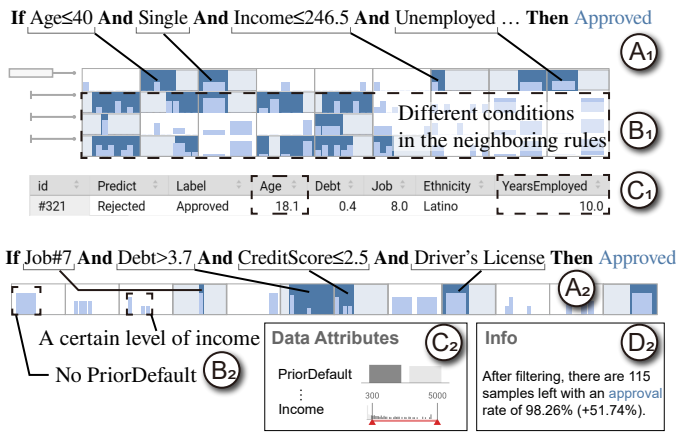
If Age≤40 **And** Single **And** Income≤246.5 **And** Unemployed … **Then** Approved

Ⓐ₁

Different conditions in the neighboring rules
Ⓑ₁

id	Predict	Label	Age	Debt	Job	Ethnicity	YearsEmployed	Ⓒ₁
#321	Rejected	Approved	18.1	0.4	8.0	Latino	10.0	

If Job#7 **And** Debt>3.7 **And** CreditScore≤2.5 **And** Driver's License **Then** Approved

Ⓐ₂

A certain level of income

No PriorDefault Ⓑ₂

Data Attributes Ⓒ₂

PriorDefault

Income   300   5000

Info Ⓓ₂

After filtering, there are 115 samples left with an approval rate of 98.26% (+51.74%).

Fig. 9. Analyzing the top-2 rules with the highest anomaly scores: 1) the analysis of the first rule (A₁), its neighbors (B₁), and the associated sample (C₁) reveals a labeling error; 2) the analysis of the second rule (A₂), its sample distribution (B₂), and information from the *attribute and info views* (C₂, D₂) confirms a spurious association.

enabled him to examine the samples covered by this rule. He found that these samples shared some negative conditions (Fig. 5D), including unemployment, missing credit scores, lack of income, and having prior default records, but were labeled as "approved" (Fig. 5E). E_1 suspected that they should be rejected given those negative conditions. He then used the *attribute view* to filter the samples with such negative conditions and found in the *info view* that they were mostly labeled as "rejected." This indicated that there were inconsistent approval criteria in the training data. He raised this issue with E_2, and E_2 commented that such inconsistency harmed the model training and led to performance degradation. Therefore, E_1 unified the approval criteria in these samples and corrected their labels to be "rejected." By retraining the model on the corrected data, the accuracy increased from 83.60% to 86.20%, and the prediction confidence increased from 0.7403 to 0.7423. **Analyzing anomalous rules**. Reordering rules according to their anomaly scores facilitates the analysis of anomalous rules. E_1 started his analysis by examining the rule with the highest anomaly score. As shown in Fig. 9A₁, this rule suggests approval for young, single, unemployed applicants with a medium-to-low income. These conditions were mostly neutral or negative, which should not lead to approval. He zoomed into its neighborhood and observed that few conditions were shared (Fig. 9B₁), which also indicated that its behavior differed a lot from other rules. He then examined its covered samples in the *data table*. As shown in Fig. 9C₁, it covered only an 18-year-old applicant with a 10-year work experience (#321), which was suspicious. E_1 confirmed that this applicant should not be approved and corrected its label to "rejected".

Subsequently, E_1 examined the rule with the second-highest anomaly score. This rule suggested approval for applicants with primarily negative or neutral conditions such as some debt, low credit score, having a driver's license (Fig. 9A₂). E_1 doubted the approval decision for these applicants, and checked the samples covered by this rule. He found most of the applicants had no prior default records and a certain level of income (Fig. 9B₂). E_1 believed these positive conditions were the real reason for the approval. By filtering samples

with such positive conditions using the scented widgets in the *attribute view* (Fig. 9C₂), he found that most of them (98.26%) are approved (Fig. 9D₂). Thus, he suspected the occurrence of this rule is because of the spurious association between the irrelevant conditions and the approval decision in these covered samples (Fig. 9A₂). Discussion with E_2 suggested adding more diverse samples, especially those that meet the conditions of this rule but get rejected, to mitigate this spurious association. E_2 then generated more samples using data augmentation, and E_1 validated these samples to ensure they accurately reflected real-world scenarios.

They analyzed other anomalous rules with the top-20 anomaly scores and found 1 more rule caused by label error and 15 more rules caused by spurious associations. Similarly, they corrected the label error and augmented the training samples. After retraining the model, the accuracy increased from 86.20% to 88.60%, and the average prediction confidence increased from 0.7423 to 0.7630. The average anomaly score of the rules also decreased from 0.0547 to 0.0493.

2) Quantitative Stock Trading: In this case study, we collaborated with E_3, a junior quantitative stock trader specializing in short-term trading in the U.S. stock market. He has also participated in the interview in Sec. III. His goal was to identify stocks with rising prices using a stock price prediction model. The training data consists of 119,416 samples collected from the U.S. stock market from April 2020 to April 2023, and the test data consists of 10,062 samples collected from May 2023 to August 2023. Each sample represents the information of a stock at a specific time point, including two types of attributes [21]: 158 *technical analysis attributes* and 99 *fundamental attributes*. The technical analysis attributes are derived from historical prices to evaluate and forecast financial market trends. For example, *STD60* measures the variance in stock prices over the preceding 60 days. The fundamental attributes provide insights into a company's financial health and operational performance. For example, *revenueGrowth* reflects the company's revenue growth rate. Based on the stock price fluctuation over the next seven days, the samples are categorized into three classes: "**increase**," "**decrease**," and "**stable**." Directly using all attributes to train the model can deteriorate its generalization ability due to attribute redundancy [29]. Following the common practice, E_3 selected attributes with an information coefficient larger than 0.05 [25], resulting in 140 attributes remaining. He then used these attributes to train a gradient boosted tree model with LightGBM [32] due to its superior performance and widespread adoption. The model was configured with 1000 trees and a maximum depth of 14. This model includes **81,000** rules but achieves an accuracy of only 56.26%, with class-specific accuracy of 26.69% for "increase," 41.53% for "decrease," and 66.56% for "stable." **Overview**. Initially, E_3 wanted to verify whether the model utilized appropriate attributes for prediction. He checked the most frequently used attributes in the *matrix view* and found the top-5 important attributes are technical analysis attributes (Fig. 10(a)). E_3 considered it reasonable as technical analysis attributes offered a more direct reflection of price fluctuations than fundamental attributes. Reordering rules based on the most important attribute *KLEN*, he found that the model tended
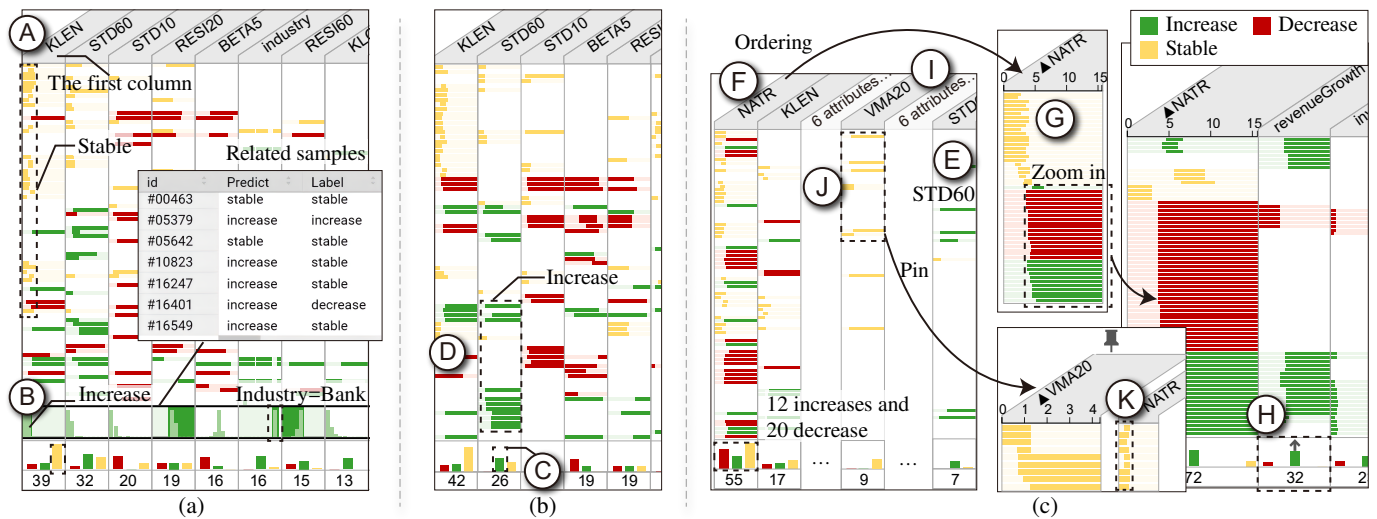
Fig. 10. Analyzing rules for quantitative stock trading: (a) in the initial model, rules with lower *KLEN* typically indicate "Stable" (A), except for one outlier (B); (b) after removing attributes that cause biased predictions, *STD60* was found to be crucial for predicting an "Increase" (C, D); (c) after adding more useful attributes for predicting "Increase", the importance of *STD60* decreased (E), while *NATR* became the most important attribute (F), which triggered further analysis (G, H, I).

to predict the stock price as "stable" when *KLEN* was low. This is reasonable because stocks with lower *KLEN* are more likely to maintain stable prices in the next few days.

Addressing biased predictions. However, there was an exception: when *KLEN* was low, one rule made an "increase" prediction (Fig. 10B). Examining the samples covered by this rule, E₃ found that these samples had small *KLEN*, medium *RESI20*, and medium *RESI60*, all of which indicated stable stock prices. However, they also shared the condition of *industry=Bank*, which may be the reason for the prediction of "increase." E₃ explained that the banks associated with these samples experienced a growth period when the training data was collected, so the trained model tended to predict them as "increase." However, such a pattern did not exist in the test samples and thus caused a performance drop. To address this issue, he removed the attribute *industry*. After retraining the model, he observed an improvement in accuracy from 56.26% to 56.52%. Specifically, on the samples with *industry=Bank*, the accuracy increased from 48.90% to 51.22%.

Improving model performance in "increase." As E₃'s primary goal was to find stocks with possibly rising prices to maximize profits, he focused on analyzing the attributes associated to the "increase" prediction in the retrained model. As shown in Fig. 10C, he observed that *STD60* was frequently used for the prediction of "increase". Reordering rules based on this attribute, he noticed that the rules with the conditions of larger *STD60* consistently made predictions of "increase" (Fig. 10D). However, as E₃ commented, a larger *STD60* just meant a large fluctuation in stock prices over the past 60 days, which did not necessarily indicate an increasing stock price. E₃ considered *STD60* was not a useful attribute for the "increase" prediction, and suspected the heavy reliance on it accounted for the low prediction accuracy in this class (26.35%). He then performed feature engineering by leveraging TA-lib [4], an open-source tool, to generate more technical analysis attributes. After adding these attributes and retraining the model,

the overall accuracy increased from 56.52% to 58.35%, and the class accuracy of "increase" increased from 26.35% to 28.24%.

To verify whether the newly added attributes were used correctly, E₃ examined the cumulative statistics for attributes at the bottom of the *matrix view*. The usage of *STD60* decreased from 26 to 7 (Fig. 10E), while a newly added attribute, *NATR*, became the most frequently used (Fig. 10F). *NATR* represents asset volatility. A high *NATR* indicates a large variation in stock prices. Reordering rules based on *NATR*, E₃ found that, indeed, high *NATR* values are associated with "increase" or "decrease" predictions, while low *NATR* values are associated with "stable" predictions (Fig. 10G). Examining the neighborhood of rules with high *NATR*, E₃ found that *revenueGrowth*, the growth of the profit rate, was frequently used together (Fig. 10H). Reordering rules based on *NATR* and *revenueGrowth* revealed a clear pattern that high *revenueGrowth* mostly led to an "increase" prediction, while low *revenueGrowth* mostly led to a "decrease" prediction. This observation was consistent with his experience.

Removing redundant attributes. During further exploration, E₃ identified some redundant attributes for prediction. For example, *VMA20*, which indicates the average trading volume in the preceding 20 days, was recognized as an important attribute (Fig. 10I). However, rules using this attribute made predictions of "stable" no matter low or high *VMA20* values, which seemed to suggest the value of VMA20 does not affect the prediction. Pinning *VMA20* to the first column and reordering rules based on it, E₃ found that all the rules using VMA20 also had low *NATR* values. As low NATR indicates a small variation in stock prices, E₃ believed it was the actual reason for the predictions of these rules, and *VMA20* was redundant. He proposed to remove *VMA20*. Similarly, he removed another eight redundant attributes. After retraining the model, its accuracy was improved to 59.20%.

Finally, E₃ employed the final model for simulated trading.

The trading strategy involved selecting 20 stocks predicted to be the most likely to increase in each 7-day trading cycle over a period of 3 months. Backtesting proved that this trading strategy produced an excess return of 48.7% with a maximum drawdown of 5.0%. E_3 was pleased with this performance and said the comprehensive understanding of this model made him more confident in employing the model for real-world trading.

C. Expert Feedback and Discussion

After the case studies, we conducted semi-structured interviews with each of the six experts (E_1-E_6) introduced in Sec. III-A. E_4-E_6 were not involved in the case studies. These newly involved experts included a fund manager with two years of experience (E_4), a vice president at a commercial bank branch (E_5), and a machine learning researcher with five years of experience (E_6). For E_4-E_6, we spent the initial 20 minutes introducing RuleExplorer and presenting the case studies. Each expert then used RuleExplorer for model understanding and verification tasks. It took 5-15 minutes for them to become familiar with the interactions and conduct analysis. We concluded with a discussion on RuleExplorer's advantages and limitations. Each interview lasted 35-60 minutes.

1) Usability: Overall, the experts are quite satisfied with RuleExplorer.

Learning curve. The experts consider RuleExplorer easy to use. They like that the matrix view takes a format similar to a data table and naturally supports table operations, such as sorting and swapping. This familiar design has enhanced their analysis efficiency. They also indicate that using RuleExplorer enables them to quickly understand the model, which has increased their trust in the model and their confidence in decision-making. Specifically, they expressed satisfaction with the dynamic hierarchy, which enables them to analyze different subsets of rules on demand. The experts also value the ease of finding anomalous rules using RuleExplorer. As E_2 commented, "Different from RuleMatrix [40] and ExplainableMatrix [43], this tool automatically recommends some unusual rules, which help me quickly identify potential issues in the model and make interventions accordingly." We believe that conducting a user study to compare RuleExplorer with these interactive systems would provide deeper insights into its usability and its effectiveness in facilitating model understanding. We consider this an important future work.

Scalability. Matrix visualizations often face the scalability issue due to the limited number of rows and columns that can be displayed [39], [65]. To tackle this issue, RuleExplorer 1) organizes a large rule set into a hierarchy and dynamically selects representative rules to display, and 2) sorts attributes by importance and divides them into pages. These two strategies are effective because tree ensemble models typically rely on a smaller set of key attributes for decision making [7], and users usually focus on a small subset of representative rules and key attributes for initial analysis. In addition, our tool can easily be extended to handle other scalability issues. For example, for attributes with thousands of categories, we can employ alternative feature transformation methods, such as Target Encoding [24], which replaces each category with a single numerical value representing its statistical correlation to the target variable. For classification tasks involving more than 10 classes, we can hierarchically organize classes and dynamically assign colors during user exploration [15].

Extensibility. While RuleExplorer is primarily designed for tree ensemble classifiers, it can be easily extended for regression models. To achieve this, users simply need to substitute the classification-specific loss with a regression-specific loss during model reduction and employ sequential color schemes to encode the regression values of each rule. In addition, E_6 also pointed out that RuleExplorer can be used to explain other models beyond tree ensemble models: "Similar to RuleMatrix [40], this tool supports the analysis of other machine learning models using model surrogate techniques. This greatly extends the applicability of the tool."

2) Limitations: In addition to the positive feedback, the experts also point out several limitations that offer directions for future research.

Online model update. In the current analysis, after the experts identify problems and make corrections, they need to retrain the model and load it back into the system. The experts highlighted that the ability to incrementally update parts of the model online to examine the result would help improve the analysis efficiency. Exploring methods for incrementally updating the model and effectively visualizing the changes presents an interesting research direction.

What-if analysis. Another interesting direction is the what-if analysis of rules. E_6 suggested allowing examination of how different rule conditions impact the output. However, further exploration is needed on how to use what-if analysis to more accurately guide experts in improving the model. This research will focus on understanding model rules and making corresponding modifications. We plan to gather more specific requirements from experts to guide future research.

Algorithm efficiency. The current efficiency of the model reduction algorithm is not as high as desired. The running time ranges from tens of seconds to a few hours, depending on the model complexity and data size. For example, it takes around 1 minute for the model in the first case study and takes 3-4 hours in the second case study. However, since model reduction is performed offline, experts find this time acceptable, as they are willing to trade off time for scalability in the big data era. Nevertheless, accelerating this process allows experts to start their analysis at the earliest opportunity. The bottlenecks here lie in the grid search for the algorithm parameters and linear programming in Sec. IV. In the future, we will design a method for quickly determining algorithm parameters, and simplify the linear programming by retaining more important constraints and variables to accelerate problem solving.

VII. CONCLUSION

In this paper, we have presented RuleExplorer, a visual analysis tool that helps domain experts understand the rules extracted from tree ensemble models. To address the scalability in handling large-scale rule sets, we combine an anomaly-biased model reduction method with a matrix-based hierarchical visualization. Organizing rules into a hierarchy achieves

scalability without sacrificing fidelity. We also propose a dynamic way to construct the rule hierarchy to accommodate different analysis preferences. The anomaly-biased model reduction preserves both common and anomalous rules at each level, and thus enables not only an understanding of the overall model behavior, but also the identification of potential flaws in the model. A quantitative evaluation and two case studies are conducted to demonstrate the effectiveness of RuleExplorer and its usefulness in real-world applications.

ACKNOWLEDGMENTS

REFERENCES

[1] B. Alper, B. Bach, N. Henry Riche, T. Isenberg, and J.-D. Fekete. Weighted graph comparison techniques for brain connectivity analysis. In *Proceedings of the SIGCHI conference on human factors in computing systems*, pp. 483–492, 2013. doi: 10.1145/2470654.2470724

[2] D. L. Applegate. *The traveling salesman problem: a computational study*, vol. 17. Princeton university press, 2006. doi: 10.5555/1374811

[3] A. Asuncion, D. Newman, et al. UCI machine learning repository, 2007.

[4] J. Benediktsson. TA-lib-python, 2001.

[5] A. Bilal, A. Jourabloo, M. Ye, X. Liu, and L. Ren. Do convolutional neural networks learn class hierarchy? *IEEE Transactions on Visualization and Computer Graphics*, 24(1):152–162, 2018. doi: 10.1109/TVCG.2017.2744683

[6] B. M. Bolstad, R. A. Irizarry, M. Åstrand, and T. P. Speed. A comparison of normalization methods for high density oligonucleotide array data based on variance and bias. *Bioinformatics*, 19(2):185–193, 2003. doi: 10.1093/bioinformatics/19.2.185

[7] L. Breiman. Random forests. *Machine learning*, 45(1):5–32, 2001. doi: 10.1023/A:1010933404324

[8] D. Cashman, G. Patterson, A. Mosca, N. Watts, S. Robinson, and R. Chang. RNNbow: Visualizing learning via backpropagation gradients in rnns. *IEEE Computer Graphics and Applications*, 38(6):39–50, 2018. doi: 10.1109/MCG.2018.2878902

[9] A. Chatzimparmpas, R. M. Martins, and A. Kerren. Visruler: Visual analytics for extracting decision rules from bagged and boosted decision trees. *Information Visualization*, 22(2):115–139, 2023. doi: 10.1177/14738716221142005

[10] A. Chatzimparmpas, R. M. Martins, A. C. Telea, and A. Kerren. Deforestvis: Behaviour analysis of machine learning models with surrogate decision stumps. In *Computer Graphics Forum*, vol. 43, p. e15004, 2024. doi: 10.1111/cgf.15004

[11] C. Chen, J. Chen, W. Yang, H. Wang, J. Knittel, X. Zhao, S. Koch, T. Ertl, and S. Liu. Enhancing single-frame supervision for better temporal action localization. *IEEE Transactions on Visualization and Computer Graphics*, 2024. doi: 10.1109/TVCG.2024.3388521

[12] C. Chen, Y. Guo, F. Tian, S. Liu, W. Yang, Z. Wang, J. Wu, H. Su, H. Pfister, and S. Liu. A unified interactive model evaluation for classification, object detection, and instance segmentation in computer vision. *IEEE Transactions on Visualization and Computer Graphics*, 30(1):76–86, 2024. doi: 10.1109/TVCG.2023.3326588

[13] C. Chen, J. Wu, X. Wang, S. Xiang, S.-H. Zhang, Q. Tang, and S. Liu. Towards better caption supervision for object detection. *IEEE Transactions on Visualization and Computer Graphics*, 28(4):1941–1954, 2022.

[14] C. Chen, J. Yuan, Y. Lu, Y. Liu, H. Su, S. Yuan, and S. Liu. OoDAnalyzer: Interactive analysis of out-of-distribution samples. *IEEE Transactions on Visualization and Computer Graphics*, 27(7):3335–3349, 2021. doi: 10.1109/TVCG.2020.2973258

[15] J. Chen, W. Yang, Z. Jia, L. Xiao, and S. Liu. Dynamic color assignment for hierarchical data. In *IEEE Transactions on Visualization and Computer Graphics*, pp. 1–11, 2024. doi: 10.1109/TVCG.2024.3456386

[16] K. Crammer and Y. Singer. On the algorithmic implementation of multiclass kernel-based vector machines. *Journal of Machine Learning Research*, 2(12):265–292, 2001. doi: 10.5555/944790.944813

[17] M. Craven and J. Shavlik. Extracting tree-structured representations of trained networks. In *Proceedings of the Advances in Neural Information Processing Systems*, pp. 24–30, 1995. doi: 10.5555/2998828.2998832

[18] K. Deb, K. Sindhya, and J. Hakanen. Multi-objective optimization. In *Decision sciences*, pp. 161–200. CRC Press, 2016. doi: 10.1007/0-387-28356-0_10

[19] J. F. DeRose, J. Wang, and M. Berger. Attention flows: Analyzing and comparing attention mechanisms in language models. *IEEE Transactions on Visualization and Computer Graphics*, 27(2):1160–1170, 2021. doi: 10.1109/TVCG.2020.3028976

[20] M. Du, N. Liu, and X. Hu. Techniques for interpretable machine learning. *Communications of the ACM*, 63(1):68–77, 2019. doi: 10.1145/3359786

[21] R. D. Edwards, J. Magee, and W. C. Bassetti. *Technical analysis of stock trends*. CRC press, 2018. doi: 10.1201/b14301

[22] J. Eirich, M. Münch, D. Jäckle, M. Sedlmair, J. Bonart, and T. Schreck. Rfx: a design study for the interactive exploration of a random forest to enhance testing procedures for electrical engines. In *Proceedings of Computer Graphics Forum*, pp. 302–315, 2022. doi: 10.1111/cgf.14452

[23] N. Elmqvist and J.-D. Fekete. Hierarchical aggregation for information visualization: Overview, techniques, and design guidelines. *IEEE Transactions on Visualization and Computer Graphics*, 16(3):439–454, 2009. doi: 10.1109/TVCG.2009.84

[24] A. Géron. *Hands-on machine learning with Scikit-Learn, Keras, and TensorFlow*. " O'Reilly Media, Inc.", 2022. doi: 10.5555/3378999

[25] T. H. Goodwin. The information ratio. *Financial Analysts Journal*, 54(4):34–43, 1998. doi: 10.2469/faj.v54.n4.2196

[26] J. Görtler, F. Hohman, D. Moritz, K. Wongsuphasawat, D. Ren, R. Nair, M. Kirchner, and K. Patel. Neo: Generalizing confusion matrix visualization to hierarchical and multi-output labels. In *Proceedings of the CHI Conference on Human Factors in Computing Systems*, pp. 1–13, 2022. doi: 10.1145/3491102.3501823

[27] J. Han and N. Cercone. Ruleviz: a model for visualizing knowledge discovery process. In *Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pp. 244–253, 2000. doi: 10.1145/347090.347139

[28] S. Hara and K. Hayashi. Making tree ensembles interpretable: A bayesian model selection approach. In *Proceedings of the International Conference on Artificial Intelligence and Statistics*, pp. 77–85, 2018.

[29] H. H. Htun, M. Biehl, and N. Petkov. Survey of feature selection and extraction techniques for stock market prediction. *Financial Innovation*, 9(1):26, 2023. doi: 10.5555/944790.944813

[30] J. P. Ignizio and T. M. Cavalier. *Linear programming*. Prentice-Hall, Inc., 1994.

[31] S. Jia, P. Lin, Z. Li, J. Zhang, and S. Liu. Visualizing surrogate decision trees of convolutional neural networks. *Journal of Visualization*, 23:141–156, 2020. doi: 10.1007/s12650-019-00607-z

[32] G. Ke, Q. Meng, T. Finley, T. Wang, W. Chen, W. Ma, Q. Ye, and T.-Y. Liu. LightGBM: A highly efficient gradient boosting decision tree. In *Proceedings of the Advances in Neural Information Processing Systems*, pp. 3146–3154, 2017. doi: 10.5555/3294996.3295074

[33] B. C. Kwon, M.-J. Choi, J. T. Kim, E. Choi, Y. B. Kim, S. Kwon, J. Sun, and J. Choo. RetainVis: Visual analytics with interpretable and interactive recurrent neural networks on electronic medical records. *IEEE Transactions on Visualization and Computer Graphics*, 25(1):299–309, 2019. doi: 10.1109/TVCG.2018.2865027

[34] D. Leman, A. Feelders, and A. Knobbe. Exceptional model mining. In *ECML PKDD*, pp. 1–16, 2008. doi: 10.1007/978-3-540-87481-2_1

[35] M. Liu, J. Shi, Z. Li, C. Li, J. Zhu, and S. Liu. Towards better analysis of deep convolutional neural networks. *IEEE Transactions on Visualization and Computer Graphics*, 23(1):91–100, 2017. doi: 10.1109/TVCG.2016.2598831

[36] S. Liu, J. Xiao, J. Liu, X. Wang, J. Wu, and J. Zhu. Visual diagnosis of tree boosting methods. *IEEE Transactions on Visualization and Computer Graphics*, 24(1):163–173, 2018. doi: 10.1109/TVCG.2017.2744378

[37] F. Lyu, C. Chen, J. Zhang, X. Feng, and Z. Tang. Visualization for supercomputer system: A survey. *Journal of Computer-Aided Design & Computer Graphics*, 36(3):321–335, 2024. doi: 10.3724/SP.J.1089.2024.2023-00791

[38] N. Meinshausen. Node harvest. *The Annals of Applied Statistics*, 4(4):2049–2072, 2010. doi: 10.1214/10-aoas367

[39] Y. Ming, S. Cao, R. Zhang, Z. Li, Y. Chen, Y. Song, and H. Qu. Understanding hidden memories of recurrent neural networks. In *Proceedings of the IEEE Conference on Visual Analytics Science and Technology*, pp. 13–24, 2017. doi: 10.1109/VAST.2017.8585721

[40] Y. Ming, H. Qu, and E. Bertini. RuleMatrix: Visualizing and understanding classifiers with rules. *IEEE Transactions on Visualization and Computer Graphics*, 25(1):342–352, 2019. doi: 10.1109/TVCG.2018.2864812

[41] T. Mühlbacher, L. Linhardt, T. Möller, and H. Piringer. TreePOD: Sensitivity-aware selection of pareto-optimal decision trees. *IEEE Transactions on Visualization and Computer Graphics*, 24(1):174–183, 2018. doi: 10.1109/TVCG.2017.2745158

[42] T. Munzner. *Visualization analysis and design*. CRC press, 2014. doi: 10.1201/b17511

[43] M. P. Neto and F. V. Paulovich. Explainable matrix-visualization for global and local interpretability of random forest classification ensembles. *IEEE Transactions on Visualization and Computer Graphics*, 27(2):1427–1437, 2021. doi: 10.1109/TVCG.2020.3030354

[44] A. Nurunnabi and G. West. Outlier detection in logistic regression: A quest for reliable knowledge from predictive modeling and classification. In *Proceedings of the IEEE International Conference on Data Mining Workshops*, pp. 643–652, 2012. doi: 10.1109/ICDMW.2012.107

[45] N. Patki, R. Wedge, and K. Veeramachaneni. The synthetic data vault. In *Proceedings of the IEEE International Conference on Data Science and Advanced Analytics*, pp. 399–410, 2016. doi: 10.1109/DSAA.2016.49

[46] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011. doi: 10.5555/1953048.2078195

[47] R. Quinlan. Statlog (Australian Credit Approval). UCI Machine Learning Repository. doi: 10.24432/C59012

[48] D. Ren, S. Amershi, B. Lee, J. Suh, and J. D. Williams. Squares: Supporting interactive performance analysis for multiclass classifiers. *IEEE Transactions on Visualization and Computer Graphics*, 23(1):61–70, 2017. doi: 10.1109/TVCG.2016.2598828

[49] M. T. Ribeiro, S. Singh, and C. Guestrin. "why should i trust you?" explaining the predictions of any classifier. In *Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pp. 1135–1144, 2016. doi: 10.1145/2939672.2939778

[50] L. Rosasco, E. De Vito, A. Caponnetto, M. Piana, and A. Verri. Are loss functions all the same? *Neural computation*, 16(5):1063–1076, 2004. doi: 10.1162/089976604773135104

[51] L. Sawatzky, S. Bergner, and F. Popowich. Visualizing rnn states with predictive semantic encodings. In *Proceedings of the IEEE Visualization Conference*, pp. 156–160, 2019. doi: 10.1109/VISUAL.2019.8933744

[52] B. Shneiderman. The eyes have it: a task by data type taxonomy for information visualizations. In *Proceedings of the IEEE Symposium on Visual Languages*, pp. 336–343. Elsevier, 1996. doi: 10.1109/VL.1996.545307

[53] H. Strobelt, S. Gehrmann, M. Behrisch, A. Perer, H. Pfister, and A. M. Rush. Seq2Seq-Vis: A visual debugging tool for sequence-to-sequence models. *IEEE Transactions on Visualization and Computer Graphics*, 25(1):353–363, 2019. doi: 10.1109/TVCG.2018.2865044

[54] H. Strobelt, S. Gehrmann, H. Pfister, and A. M. Rush. LSTMVis: A tool for visual analysis of hidden state dynamics in recurrent neural networks. *IEEE Transactions on Visualization and Computer Graphics*, 24(1):667–676, 2018. doi: 10.1109/TVCG.2017.2744158

[55] G. K. Tam, V. Kothari, and M. Chen. An analysis of machine-and human-analytics in classification. *IEEE Transactions on Visualization and Computer Graphics*, 23(1):71–80, 2016. doi: 10.1109/TVCG.2016.2598829

[56] S. Tan, R. Caruana, G. Hooker, and Y. Lou. Distill-and-Compare: Auditing black-box models using transparent model distillation. In *Proceedings of the AAAI/ACM Conference on AI, Ethics, and Society*, pp. 303–310, 2018. doi: 10.1145/3278721.3278725

[57] F. Y. Tzeng and K. L. Ma. Opening the black box - data driven visualization of neural networks. In *Proceedings of the IEEE Visualization Conference*, pp. 383–390, 2005. doi: 10.1109/VISUAL.2005.1532820

[58] S. Van Den Elzen and J. J. Van Wijk. BaobabView: Interactive construction and analysis of decision trees. In *Proceedings of the IEEE Conference on Visual Analytics Science and Technology*, pp. 151–160, 2011. doi: 10.1109/VAST.2011.6102453

[59] T. Vidal and M. Schiffer. Born-again tree ensembles. In *Proceedings of the International Conference on Machine Learning*, pp. 9743–9753, 2020. doi: 10.5555/3524938.3525841

[60] Z. J. Wang, C. Zhong, R. Xin, T. Takagi, Z. Chen, D. H. Chau, C. Rudin, and M. Seltzer. Timbertrek: exploring and curating sparse decision trees with interactive visualization. In *Proceedings of the IEEE Visualization Conference*, pp. 60–64, 2022. doi: 10.1109/VIS54862.2022.00021

[61] W. Willett, J. Heer, and M. Agrawala. Scented widgets: Improving navigation cues with embedded visualizations. *IEEE Transactions on Visualization and Computer Graphics*, 13(6):1129–1136, 2007. doi: 10.1109/TVCG.2007.70589

[62] S. Xiang, X. Ye, J. Xia, J. Wu, Y. Chen, and S. Liu. Interactive correction of mislabeled training data. In *Proceedings of the IEEE Conference on Visual Analytics Science and Technology*, pp. 57–68, 2019. doi: 10.1109/VAST47406.2019.8986943

[63] W. Yang, M. Liu, Z. Wang, and S. Liu. Foundation models meet visualizations: Challenges and opportunities. *Computational Visual Media*, 10(3):399–424, 2024. doi: 10.1007/s41095-023-0393-x

[64] W. Yang, X. Wang, J. Lu, W. Dou, and S. Liu. Interactive steering of hierarchical clustering. *IEEE Transactions on Visualization and Computer Graphics*, 27(10):3953–3967, 2021. doi: 10.1109/TVCG.2020.2995100

[65] W. Yang, X. Ye, X. Zhang, L. Xiao, J. Xia, Z. Wang, J. Zhu, H. Pfister, and S. Liu. Diagnosing ensemble few-shot classifiers. *IEEE Transactions on Visualization and Computer Graphics*, 28(9):3292–3306, 2022. doi: 10.1109/TVCG.2022.3182488

[66] J. Yuan, B. Barr, K. Overton, and E. Bertini. Visual exploration of machine learning model behavior with hierarchical surrogate rule sets. *IEEE Transactions on Visualization and Computer Graphics*, 30(2):1470–1488, 2024. doi: 10.1109/TVCG.2022.3219232

[67] J. Yuan, M. Liu, F. Tian, and S. Liu. Visual analysis of neural architecture spaces for summarizing design principles. *IEEE Transactions on Visualization and Computer Graphics*, 29(1):288–298, 2023. doi: 10.1109/TVCG.2022.3209404

[68] J. Yuan, O. Nov, and E. Bertini. An exploration and validation of visual factors in understanding classification rule sets. In *Proceedings of the IEEE Visualization Conference*, pp. 6–10, 2021. doi: 10.1109/VIS49827.2021.9623303

[69] X. Zhao, Y. Wu, D. L. Lee, and W. Cui. iForest: Interpreting random forests via visual analytics. *IEEE Transactions on Visualization and Computer Graphics*, 25(1):407–416, 2019. doi: 10.1109/TVCG.2018.2864475

[70] W. Zhou, T. D. Roy, and I. Skrypnyk. The KDD cup 2019 report. *ACM SIGKDD Explorations Newsletter*, 22(1):8–17, 2020. doi: 10.1145/3400051.3400056

[71] Y. Zhou, W. Yang, J. Chen, C. Chen, Z. Shen, X. Luo, L. Yu, and S. Liu. Cluster-aware grid layout. *IEEE Transactions on Visualization and Computer Graphics*, 30(01), 2024. doi: 10.1109/TVCG.2023.3326934

[72] Z.-H. Zhou and J. Feng. Deep forest. *National science review*, 6(1):74–86, 2019. doi: 10.1093/nsr/nwy108

Zhen Li is a fourth-year Ph.D. student of Software School, Tsinghua University. His research interest is explainable artificial intelligence. He received a B.S. degree from Tsinghua University and a M.Phil. degree from Hong Kong University of Science and Technology.

Weikai Yang is an assistant professor in Hong Kong University of Science and Technology (Guangzhou). His research interests lie in visual analytics, machine learning, and data quality improvement. He received a B.S. and a Ph.D from Tsinghua University.

Fan Yang is an algorithm engineer at Kuaishou. His research focuses on video understanding, large language model, and joint modeling of content and behavior. He received a M.S. from Tsinghua University and B.S. from Beijing University of Posts and Telecommunications.
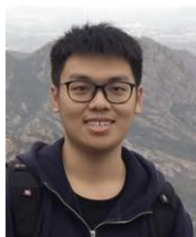
Jun Yuan is currently a Ph.D. student at Tsinghua University. His research interests are in explainable artificial intelligence. He received a B.S. degree from Tsinghua University.


Hui Zhang is an Associate Professor at School of Software, Tsinghua University, China. She received her B.Sc. and Ph.D. in Computer Science from Tsinghua University, in 1997 and 2003, respectively. Her research interests include computer aided design and computer graphics.


Jing Wu is a lecturer in computer science and informatics at Cardiff University, UK. Her research interests are in computer vision and graphics including image-based 3D reconstruction, face recognition, machine learning and visual analytics. She received BSc and MSc from Nanjing University, and Ph.D. from the University of York, UK. She serves as a PC member in CGVC, BMVC, etc.

Shixia Liu is a professor at Tsinghua University. Her research interests include explainable artificial intelligence, visual analytics for big data. She worked as a research staff member at IBM China Research Lab and a lead researcher at Microsoft Research Asia. She received a B.S. and M.S. from Harbin Institute of Technology, a Ph.D. from Tsinghua University. She is a fellow of IEEE and an associate editor-in-chief of IEEE Trans. Vis. Comput. Graph.
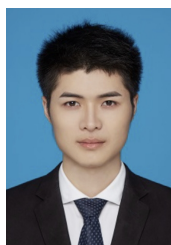
Changjian Chen is an assistant professor at Hunan University. He received a Ph.D. from Tsinghua University and a B.S. from University of Science and Technology of China. His research interests focus on visual analytics and machine learning, especially visual analysis methods to improve training data quality.
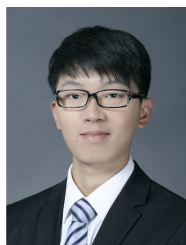
Yao Ming is a quantitative researcher at Citadel Securities. His research focus on visual analytics, explainable machine learning, and natural language processing. He received a Ph.D. in Computer Science from the Hong Kong University of Science and Technology and a B.S. from Tsinghua University. For more details please refer to https://www.myaooo.com.