



# DeforestVis: Behaviour Analysis of Machine Learning Models with Surrogate Decision Stumps

Angelos Chatzimpampas,<sup>1</sup> Rafeal M. Martins,<sup>2</sup> Alexandru C. Telea<sup>3</sup> and Andreas Kerren<sup>2,4</sup>

<sup>1</sup>Department of Computer Science, Northwestern University, Evanston, USA  
angelos.chatzimpampas@northwestern.edu

<sup>2</sup>Department of Computer Science and Media Technology, Linnaeus University, Växjö, Sweden  
rafael.martins@lnu.se, kerren@acm.org

<sup>3</sup>Department of Information and Computing Sciences, Utrecht University, Utrecht, The Netherlands  
a.c.telea@uu.nl

<sup>4</sup>Department of Science and Technology, Linköping University, Norrköping, Sweden

## Abstract

As the complexity of machine learning (ML) models increases and their application in different (and critical) domains grows, there is a strong demand for more interpretable and trustworthy ML. A direct, model-agnostic, way to interpret such models is to train surrogate models—such as rule sets and decision trees—that sufficiently approximate the original ones while being simpler and easier-to-explain. Yet, rule sets can become very lengthy, with many if-else statements, and decision tree depth grows rapidly when accurately emulating complex ML models. In such cases, both approaches can fail to meet their core goal—providing users with model interpretability. To tackle this, we propose DeforestVis, a visual analytics tool that offers summarization of the behaviour of complex ML models by providing surrogate decision stumps (one-level decision trees) generated with the Adaptive Boosting (AdaBoost) technique. DeforestVis helps users to explore the complexity versus fidelity trade-off by incrementally generating more stumps, creating attribute-based explanations with weighted stumps to justify decision making, and analysing the impact of rule overriding on training instance allocation between one or more stumps. An independent test set allows users to monitor the effectiveness of manual rule changes and form hypotheses based on case-by-case analyses. We show the applicability and usefulness of DeforestVis with two use cases and expert interviews with data analysts and model developers.

**Keywords:** surrogate model, model understanding, adaptive boosting, machine learning, visual analytics, visualization

**CCS Concepts:** • Human-centred computing → Visualization; Visual analytics; • Machine learning → Supervised learning

## 1. Introduction

In machine learning (ML), surrogate models (also called *metamodels* or *emulators*) are interpretable models trained to approximate the predictions of a typically a black box, more complex, so-called *target model* [SFK08, Mol20]. Such simpler, more *transparent*, surrogate models (or surrogates in brief) are more easy to examine and interpret than the original more complex models. For example, a hard-to-understand Convolutional Neural Network (CNN) can be approximated with a surrogate decision tree trained on the CNN's output predictions [JLL\*20]. Surrogates can also describe the behaviour of target models by summarizing their predictions in terms of the features of a given data set and associating misclassifications in a test set with particular subgroups of training samples. Training a surrogate is model-agnostic—it uses no explicit knowledge

of the target model, only access to its input data and predictions [Mol20].

Surrogates offer both local and global explanations [EAMS19, Mol20]. Local models, such as LIME [RSG16], aim to explain and reason about specific predictions of a target model. Global surrogates, the focus of our work, are simpler models that try to explain the overall behaviour and global predictions of target models [DCB19]. While any ML model can be used as a surrogate, rule sets (or lists) [MQB19, FW98], decision trees [DCB19, SL91], and Generalized Additive Models (GAMs) [HHC\*19, CLG\*15, NJKC19] are three particularly effective strategies. GAMs provide interpretable model coefficients that capture nonlinear relationships between input features [HT86]. Yet, they do not show relationships between the input and output features. Tree-based rule extraction

methods [ST01] are a universal and mature technique, especially since surrogate decision trees reflect well the human decision-making process. These solutions are typically used to explain deep learning or ensemble learning algorithms with state-of-the-art predictive performance but poor decision accountability [JLL\*20]. Rule extraction is a generalizable method in theory because its surrogate modelling does not consider the inner workings of black boxes. Still, applying it directly to neural networks is impractical and may result in suboptimal performance due to the latter's inherent complexity and nonlinearity [HKPC19, CMJ\*20]. Surrogate decision trees generated from end-to-end CNNs are too vast to parse, even with a modified gradient-based approach resulting in hundreds of levels [ZYMW19]. Yuan et al. [YBOB22] found that domain experts only examined one or two features at a time via a system similar to ours designed for exploring hierarchical surrogate rule-sets. They found that shorter, larger, rulesets outperformed lengthy, uninterpretable rules and fully-grown decision trees in terms of model interpretability. Given these, an open question is: **(RQ1) How to summarize the behaviour of large-scale ML models while providing detailed but ultra-compact explanations on demand?**

The challenge of building accurate surrogate models can be addressed from two perspectives. *Top-down* approaches aim to fit the surrogate model to the whole target model to emulate its behaviour [MQB19]. *Bottom-up* approaches aggregate the results of local surrogates tailored for individual data instances [CvW22]. Hybrid approaches that first train a global surrogate model (top-down) and then allow reasoning about specific cases (bottom-up) can be better for expert users. Model explanation systems should be designed so that users can quickly understand how models predict and be able to tune their output [YBOB22]. Such systems should also enable the exploration of more precise surrogate models. These, however, come with larger, more complex, decision trees which take more effort to understand and thus reduce the surrogate's added-value [YBOB22]. Ideally, we want to retain as much information as possible while reducing the number of decision trees of the surrogate—a complexity-fidelity trade-off [VOMS21]. Separately, to achieve good generalizability for the surrogate model, one needs to override a rule only after one examines the data distribution for each feature and understands the implications of their changes locally (for a specific decision) and globally (for all decisions). This leads to our second question: **(RQ2) How to effectively help users to inject their domain knowledge into machine-produced rules while monitoring the local and global impact of their adjustments?**

We present DEFORESTVIS (see Figure 1), a Visual Analytics (VA) tool for the exploratory analysis of *decision stumps*—one-level decision trees [IL92]. DEFORESTVIS creates such stumps using the Adaptive Boosting (AdaBoost) method [Sch99]. Our tool allows users to trade off complexity of the visual explanation (number of stumps) versus fidelity of the surrogate (accuracy score). DEFORESTVIS summarizes the decision boundaries and the contribution of each feature to a separate test set. An in-depth analysis is possible by observing the influence of individual decision stumps. Also, users can visually inspect both the local and the global impact of a change in a rule. In summary, our contributions are as follows:

- a visual analytic workflow that simplifies the behaviour analysis of complex ML models via surrogate models;

- an implementation of this workflow in a VA tool via multiple linked views for selecting accurate and simple surrogate models, summarizing the behaviour of complex models while also explaining how the aggregated information was computed, and formulating what-if hypotheses when overriding particular rules extracted from decision stumps;
- a proof-of-concept use case and a usage scenario with real-world healthcare data that highlight the efficiency and effectiveness of our approach in forming compact rule sets; and
- the evaluation of our proposal via interviews with data analysts and model developers.

We organize the rest of this paper as follows. Section 2 introduces the necessary background information for making this paper self-sustained. Section 3 discusses related work on surrogate models for model interpretation and approaches for visualizing rules and decision trees. Section 4 describes the user goals, analytical tasks, and user types, of VA tools using surrogate models for behaviour analysis of complex ML models. Section 5 presents our tool. Section 6 describes a use case for examining alternative decisions and their combinatorial effect while manually adjusting a decision rule. It also shows the applicability and usefulness of DEFORESTVIS with a real-world data set for a binary classification problem. Section 7 presents feedback obtained from expert interview sessions and reports the limitations identified by the experts. Section 8 reflects further on the targeted users and the limitations that may lead to improvements for our tool. Finally, Section 9 concludes the paper.

## 2. Background

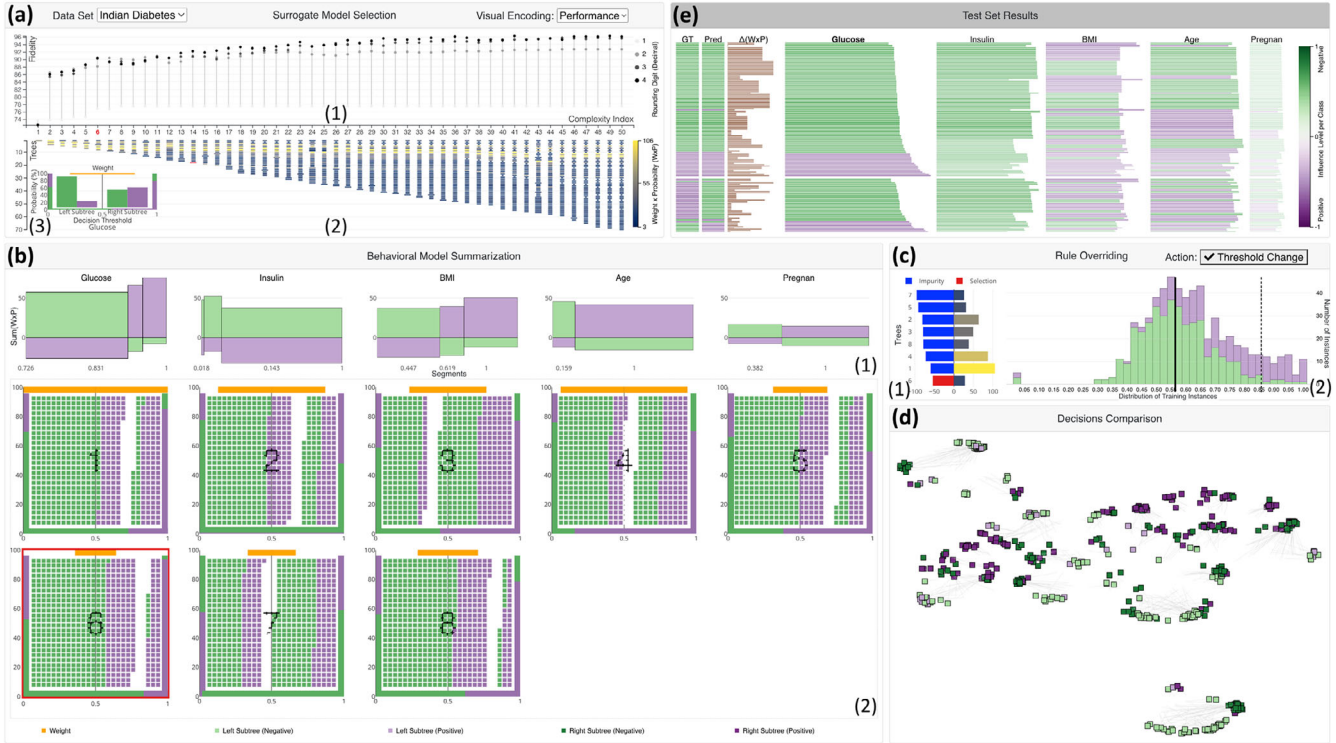
We next briefly overview the algorithmic steps of the AdaBoost surrogate model used in DEFORESTVIS, so as to introduce the reader to the utilized methods and measures. For more details, we refer to the original algorithm's paper from Freund et al. [FSA99].

**Data set:** For a binary classification problem, let  $T = \{x_i\}$ ,  $1 \leq i \leq N$ ,  $T \subset \mathbb{R}^n$ , be a training set of  $n$ -dimensional instances  $x_i$ , each having a label  $y_i \in C$ , where  $C = \{C_1, C_2\}$  for the binary case.

**Target model:** A (complex) model  $f$  is trained on  $T$  to predict variables  $y = f(x)$ .

**Surrogate model:** AdaBoost successively fits many decision stumps (*maximum depth* = 1) to weighted training samples from  $T$  and next tries to predict  $y$ . Initially, all weights are uniformly set to  $w_i = 1/N$ . Should the initial stump's prediction prove inaccurate, the samples  $x_i$  that were incorrectly predicted receive a higher weight  $w_i$ . Throughout the process, we keep the constraint  $\sum_i w_i = 1$  with  $w_i \in [0, 1]$ . The above process iterates from  $m = 1$  to  $M$  and is regulated by a *learning rate* hyperparameter. Adding decision stumps continues until reaching a user-defined limit set by a *number of trees/estimators* hyperparameter.

**Choosing features and split points for a stump:** To build individual decision stumps, a node  $v$  is selected for dividing data into two child nodes. For every  $v$  in a stump, the best *decision threshold*  $p_v$  among the  $n$  is chosen according to the *Gini impurity* [GPTM10] metric  $GI(v) = \sum_{c \in C} P_{vc}(1 - P_{vc})$ , where  $P_{vc}$  is the likelihood of a specific classification outcome  $c$  (in our case  $C_1$  and  $C_2$ ).



**Figure 1:** Components of DEFORESTVIS: (a.1) lollipop plot shows data-rounding effects in the fidelity score for four different decimal digit precisions; (a.2) dot plot with lines of various widths (unique rules/stumps  $>$  original  $>$  duplicated) and colors (visual encoding: performance, that is, weighted probability ( $W \times P$ )) that explains complexity increase as more decision stumps get added and (a.3) selective stump-based explanation; (b.1) segmented bar chart tells the predictive outcome and power of each segment based on automatically computed thresholds and (b.2) detailed stump-based explanation grid; (c.1) bar chart shows the impurity and weighted probability of each decision stump; (c.2) histogram shows the active rule's threshold and distribution of training instances; (d) projection aggregates the global behaviour of instances; colour shows the local behaviour according to the currently selected decision stump; and (e) fragmented bar chart shows the per-feature contribution and influence level for each test case. The main questions that the five views of DEFORESTVIS address are: (a) Which surrogate model gives the user the desired fidelity/complexity trade-off (Section 5.1)? (b) How do feature thresholds affect the selected surrogate model, summarizing the behaviour of the complex model (Section 5.2)? (c) How can specific rules be overridden (Section 5.3)? (d) What changes due to such user actions (Section 5.4)? (e) How does the new surrogate model perform on unseen test data (Section 5.5)?

**Impact of weights:** Within the  $m = 1$  to  $M$  loop, a model  $G_m(x)$  is fitted to the training set  $T$  using the weights  $w_i$ , leading to the concept of *weighted probability* ( $W \times P$ ) that plays a central role in DEFORESTVIS. Subsequently, AdaBoost uses this method to measure the training performance of the classifier given by  $\alpha_m = \log((1 - \text{err}_m)/\text{err}_m)$ , where  $\text{err}_m$  is the misclassification rate for the training set at iteration  $m$ . Therefore,  $\alpha_m$  gives the influence a particular stump will exert in the classification. When a stump classifies correctly, producing zero misclassifications, its error rate is 0 and its  $\alpha$  value becomes a large positive number. Conversely, a stump with a 50% correct classification will have an  $\alpha$  value of 0. If a stump predominantly misclassifies, then  $\alpha$  would become a large negative value.

**Final ensemble of weak learners:** After evaluating the error values for every stump, the sample weights are revised by  $w_i^{m+1} = w_i^m \cdot e^{\alpha_m}$ . As described earlier,  $\alpha$  is (1) positive when the prediction aligns with the actual output or (2) negative when there is a discrepancy. In case (1), the sample weight diminishes compared

to its previous value due to the algorithm already performing well. In case (2), the sample weight rises to prevent a similar misclassification by the next stump. This mechanism ensures subsequent stumps are influenced by their predecessors. Finally, AdaBoost's output aggregates all the stumps as  $G(x) = \pm[\sum_{m=1}^M \alpha_m G_m(x)]$ .

**Training multiple surrogate models:** In DEFORESTVIS, we train multiple surrogate models with incrementally more stumps to explore the trade-off between *complexity* (measured as the number of decision stumps) and *fidelity* (measured as how accurately the surrogate model fits the target model's predictions) [DCB19]). This leads to stumps that were found initially in the least complex surrogate model (*unique stumps or rules*) to be repeated either in the subsequent surrogates (*original stumps*) or the surrogate itself (*duplicated stumps*). Following the algorithmic process explained above, *duplicated stumps* use the same *feature and decision threshold value* as the *original stump*, but with different weights/voting power while possibly predicting the opposite class for the same segment. Hence, they aim to fix the error (i.e. smooth the effect) of an *original stump*.

In contrast, *unique decision stumps* most likely have greater weight, thus contributing significantly to the final ensemble of stumps.

### 3. Related Work

We next review related work on visualizing the internal structures and outputs of surrogate models (Section 3.1) and broader visualization techniques for decision trees and rules (Section 3.2). We also highlight our contributions in relation to existing work.

#### 3.1. Surrogate model visualization

Recent work has used surrogate models to approximate the behaviour of complex ML models locally [DB21, RSG16, RSG18, LL17, TDB21, YCB\*22, EAM14], globally [CB20, YBOB22, DCB19, MQB19], or on all scales [CvW22, JLL\*20]; all of these examples provide visual exploration of such surrogates as well. Closer to our work, SuRE [YBOB22] uses hierarchical rules to describe the decision space of a given ML model and explore its results by an interactive hierarchical visualization of the extracted rules. However, when checking multiple intertwined rules in the form of if-else statements, participants evaluating SuRE almost always examines at most two conditions at a time—a limitation we overcome with DEFORESTVIS due to the simple nature of one-level decision trees. Another interesting finding is that these users analyze thoroughly the effect of predictions based on *individual features*, thus matching well the main design concept of our proposed tool. Di Castro and Bertini [DCB19] use a single surrogate decision tree to replicate a classification model’s prediction and visualize it to propose simple yet effective explanations for the original model. RuleMatrix [MQB19] uses a matrix design and Sankey diagram visualization for the content of a rule list showing how data flows through the list. The problem with the above two VA tools is that they use a flat tabular layout [VCP22] which disregards the rules’ hierarchical structure and the important feature-ordering information captured by the hierarchical structure of decision trees. In our approach, this is not a problem since AdaBoost produces one-level decision trees (stumps), and we sort stumps for the same feature on the “importance” extracted directly from the AdaBoost algorithm. DRIL [CB20] presents a rule list for adjusting thresholds and examining relationships between rules and data. Our VA tool focuses on both the summarized rules and the decision stumps that serve as an extra explanation of how the aggregation of information occurs.

StrategyAtlas [CvW22], a hybrid approach, aims to explain individual data instances by aggregating multiple local surrogates [CvW22]. The method employs well-known explanation techniques, such as LIME [RSG16] and SHAP [LL17], to obtain feature-vector contributions. Points with similar feature contributions are grouped together via dimensionality reduction. However, the final visualization produced by StrategyAtlas is a surrogate decision tree, which suffers from interpretability issues due to its if-else structure. Since we use shallow decision trees, our approach suffers far less from this problem. CNN2DT shows the data flow through the surrogate decision tree of a CNN by a collapsible tree [JLL\*20]. In contrast, our approach is not specific to a single model (e.g. CNNs) and can be adapted to suit a range of domains based on the data sets and the expertise of the expert user.

Among local surrogate approaches [DB21, RSG16, RSG18, LL17, TDB21, YCB\*22, EAM14], SUBPLEX [YCB\*22], provides a visual explanation for interpreting sub-populations of local explanations. It uses clustering and projection visualization techniques to help users better understand these explanations. Yet, this approach trains a local surrogate, whereas ours aggregates the result with a global surrogate and provides explanations of individual data instances of interest to the user.

#### 3.2. Tree- and rule-based model visualization

Many VA tools have been created to examine decision trees stemming from bagging [ZWLC19, NP21, EMJ\*22, NWWH19, NP22], boosting [LXL\*18, HLLW19, WZWY21, XCC\*21], or both ensemble learning methods [CMK23]. Most relevant to our work, VisRuler [CMK23] is a VA tool that assists users in making decisions based on Random Forest (RF) and AdaBoost models. The tool’s VA workflow involves selecting a diverse set of robust models, identifying important features, and determining essential decisions for global or local explanations. While our tool partially addresses the aforementioned challenges, our key focus is to obtain decision stumps with their assigned *weights* and to enable *rule overriding* of the resulting decision stumps extracted from an accurate and simpler AdaBoost model that approximates the behaviour of a complex target model. These functionalities are both unsupported by VisRuler.

Several VA tools assist with the interpretation or diagnosis of the training process of gradient boosting models [Fri01]. GBMVis [XCC\*21] reveals the technical properties of gradient boosting, allowing the assessment of feature significance and decision-making tracking. BOOSTVIS [LXL\*18] offers views such as a temporal confusion matrix, t-SNE projection [vdMH08], and node-link diagram to monitor performance and examine rules. GBRTVIS [HLLW19] uses continuous loss function monitoring to explore gradient boosting and visualizes the process with a node-link diagram and a treemap. VISTB [WZWY21] provides a redesigned temporal confusion matrix and feature impact comparison for per-instance prediction tracking, feature selection, and hyperparameter tuning. In contrast to DEFORESTVIS, its node-link diagram designed for deep decision trees can limit the users’ ability to evaluate many decisions simultaneously. Also, our choice of the AdaBoost algorithm (simpler than gradient boosting or other ensemble learning algorithms [Bre01, Wol92, CMKK21]) to generate decision trees in combination with a simple bar chart visualization allows users to instantly explore rules and compare the confidence of the surrogate model for each rule and feature.

Several VA tools have been developed to aid in the interpretation of RF models. iForest [ZWLC19] shows the hierarchical structure of decision paths generated by RF. ExMatrix [NP21] uses a matrix-like visualization to analyse RF models and connect rules to classification results. Neto and Paulovich [NP22] propose a tool for extracting and explaining patterns in high-dimensional data sets from random decision trees. Colourful trees [NWWH19] uses a botanical metaphor to interactively explain the core parameters of RF models and allows for customized mappings of RF components to visual attributes. Finally, RfX [EMJ\*22] enables users to compare multiple decision trees from an RF model and manually adjust single trees using overlapping histograms and dissimilarity projections.

In contrast, *DEFORESTVIS* helps the mining of rules with a focus on class outcomes for all and/or specific cases, provides a simple visual representation of the logic behind the produced rules, and retains the hierarchy of decision stumps due to the intrinsic AdaBoost's weighting system. In our work, users can explore the local and global impact of a manually overridden rule before confirming their action.

The visualization of single decision trees has been previously attempted through various methods such as node-link diagrams [vdEvW11, NHS00, LJC16, CD19, BN01, PNWG17, BvLH\*11, SCS04, MGT\*03, BKSS14, WFH\*01, HC00], treemaps [MLMP18, GGPPS13], icicle plots [PSMD14, AEK00], star coordinates [TM03b, TM03a], parallel coordinates [TKC17], scatterplots [MJEP\*21], and scatterplot matrices [Do07]. However, these techniques do not work well when exploring multiple decision trees, which is important for understanding what individual trees have learned. Current visualizations of decision trees are not designed to explore model behaviour. To address this, we propose a feature-aligned tree visualization that helps to understand and analyse rules across multiple one-level decision trees. Additionally, we summarize in segmented bar charts all decision stumps' predictive power and the final predictive outcome collectively.

Finally, to the best of our knowledge, no work in the literature describes the use of VA in conjunction with the AdaBoost ensemble learning algorithm [Sch99] to generate interpretable decision stumps that aggregate the behaviour of complex ML models.

#### 4. User Goals and Analytical Tasks

We outline five User Goals (**UG1–UG5**) our and similar tools aim to achieve to extract easily comprehensible decision rules (Section 4.1). Next, we identify five Analytical Tasks (**AT1–AT5**) that *DEFORESTVIS* aims to help its users to complete (Section 4.2), following the guidelines from Munzner [Mun09]. These goals and tasks guide the design decisions made when developing *DEFORESTVIS*. Throughout the process, our target users are model developers and domain experts with a basic understanding of information visualization, a reasonable familiarity with the fundamental ML concepts, and a good understanding of their data. Our tool focuses on binary classification problems and tabular data with a limited number of meaningful features for the targeted users. We discuss these aspects further in Sections 7.7 and 8.1.

##### 4.1. User goals

Our five user goals (described below) are more general and have been extracted from other relevant works which, overall, target similar users having similar goals. Specifically, our five user goals were influenced by the research discussed in Section 3, the guidelines from Zhao et al. [ZWLC19], and our own experiences with interpretable/explainable ML [CMJK20, CMK20, CMK23] and trustworthy ML [CMJ\*20]. In particular, we took into account user goals and tasks outlined by Collaris and van Wijk [CvW22] in their study that involved interviewing six data science teams with an interest in explaining ML. Additionally, we considered the four user goals proposed by Antweiler and Fuchs [AF22] based on their collaboration with healthcare professionals, which is also relevant for our usage scenario described in Section 6.

**UG1: Replace an unintuitive ML model with an interpretable surrogate model for making decisions.** As already outlined, our idea is to replace a complex and unintuitive model with an interpretable surrogate model that can approximate the original model's behaviour while providing more transparent and understandable decision-making [CMJK20, CMJ\*20]. The surrogate model is a one-level surrogate decision tree or a rule-based system that will offer insights into how the model arrived at its predictions. When doing this, we also want that the used surrogate models should be easy to explore and understand by users (e.g. by avoiding deep decision trees—issues not addressed by prior works [DCB19, MQB19, YBOB22]).

**UG2: Identify good solutions for the trade-off between complexity and fidelity in approximation models.** To do this, it is necessary to carefully consider the specific problem at hand and the available resources, such as the time users are willing to spend and the free screen space [CMK23]. In some cases, a simple model with lower fidelity may be sufficient; in other cases, a more complex model with higher fidelity may be needed. Also, for surrogate decision trees, for each threshold value that decides if one instance falls into the left or right subtree, only a limited precision can be achieved [CMK23]. In contrast to previous works [DCB19, MQB19, CB20, YBOB22], VA tools should communicate the impact of decimal precision to choose the appropriate rounding approach for the given problem. Finally, VA tools should support humans in reducing their cognitive load—as much as possible—while retaining high accuracy.

**UG3: Analysis of the extracted decision rules individually and jointly to understand the behaviour of complex models.** Decision rules are a powerful tool to explain the behaviour of complex ML models. By analysing decision rules individually, we can identify which *features are most important* in the model's decision-making process and how they are *weighted* [AF22]. Conversely, by examining decision rules jointly, we can gain a better understanding of how the model as a whole makes decisions and find any potential biases or limitations [AF22]. Unlike RuleMatrix [MQB19], the voting power of each weighted decision stump plays a vital role in surrogate models based on boosting algorithms, as is the AdaBoost method in our case. To sum up, VA tools enabling this exploration of standalone surrogate models can entirely replace a complex model if no further retraining processes are involved.

**UG4: Comparison of similarities/disparities over groups of training samples when changing a single decision rule.** The thresholds used for splitting in a decision tree can be adjusted based on prior knowledge of the problem domain [AF22]. That is, a decision tree rule can be improved by repeatedly refining the split thresholds until they are properly adapted. However, optimizing decision trees in this way for each rule can result in models that are too specialized and may overfit the training data. Hence, it is important to test each rule with new training data to ensure that it is not overfitted. Similar to RfX [EMJ\*22]—but for surrogate models as is our case—we deem the adaptation of rules according to experts' prior knowledge an entry point for what-if hypotheses testing (see also **UG5**) and building their own trustful model [YBOB22]. VA tools should show how the adjustment of a particular threshold will influence the training instances that belong to different sub-branches of the decision stump.

**UG5: Multiple hypotheses about reversing the prediction for specific test cases.** Similar to iForest [ZWLC19]—but for surrogate models as is our case—we want to assess the accuracy of a surrogate model’s prediction *versus* the original model for case-based reasoning [CvW22]. When the surrogate makes incorrect predictions for some test cases, it is crucial to find whether we can adjust a wrong prediction in the right direction. Formulating multiple hypotheses about reversing the prediction means considering several possible feature-based explanations of why the model wrongly predicted and assessing whether one can adjust it to make a correct prediction. VA tools should support this process by explaining to users what should be changed to get a correct prediction or, more generally, adjust the surrogate model according to the domain expert’s prior knowledge (as described in UG4).

## 4.2. Analytical tasks

Considering the guidelines from Munzner [Mun09], we found five analytical tasks concerning concrete operations that users should be able to perform using DEFORESTVIS to achieve our user goals.

**AT1: Use shallow decision trees to split the complex model and see the impact of information reduction in faithfulness.** Users should be able to see how the precision value picked for the threshold in each shallow decision tree affects the accuracy of the whole surrogate model (UG1). Rounding up a few threshold-decimal values for each decision stump will reduce the time required for users to grasp the value of deciding in favour of one or the other class.

**AT2: Find the “optimal” number of decision trees needed to retain high-enough model performance.** Following AT1, users should be guided through the process of selecting the appropriate number of decision trees that preserves high enough fidelity for their given problem (UG2). One drawback of global surrogate models is that they cannot confidently tell how close to the target model is ‘enough’ for a selected surrogate to be. This is a decision that ultimately is to be taken by users who explore such surrogates for a given concrete problem [Mol20, DCB19]. Thus, it is crucial to enable the comparison of the predictive accuracy of surrogate models having different numbers of decision trees. Few trees are easier to understand but likely less precise. Many trees are arguably problematic since users are unlikely to engage meaningfully with hundreds of rules.

**AT3: Examine the summarized thresholds for each feature that lead to different predictions and drill down to investigate single decision rules.** The summarization of the per-feature decisions in a single view that combines the decisions sorted from the most to the least contributing features allows users to assess the influence of each feature (UG3). Understanding the thresholds and decision rules can help users to explain the model’s predictions. Once we have a clear understanding of these thresholds, we can drill down to investigate the underlying single decision trees/rules that the model combines to arrive at its final prediction. This involves enabling users to examine each individual feature and the threshold value associated with it.

**AT4: Assist users in manually adjusting decision rules and provide visual feedback about the impact of their actions.** Man-

ually adjusting rules is useful when the default rules do not fit users’ needs or when the system’s performance needs to be optimized based on expert decisions (cf. UG4). By allowing users to adjust each decision tree/rule, they can customize the application to their specific use case or preferences. The impact of such changes locally for a specific rule should be juxtaposed to the global influence on all decision rules and for an entire training and/or test set.

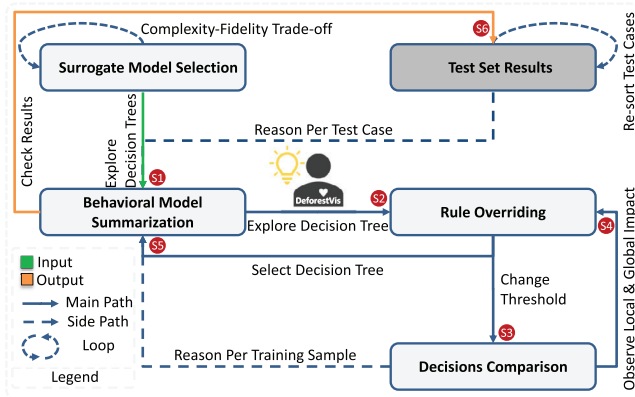
**AT5: Experiment with what-if scenarios to predict particular test instances in a different class.** An aggregated explanation of why specific test cases were misclassified should be highlighted for users (UG5). They should be capable to explore borderline cases that are close to the decision boundary of a class and form new hypotheses about which threshold one should change to classify such cases into another class. The influence level of each feature and the ease of manipulating a feature should both be visualized for users to evaluate.

## 5. DeforestVis: System Overview

We have developed DEFORESTVIS, an interactive web-based VA tool that allows users to explore the behaviour of complex ML models with feature-based explanations from surrogate decision stumps to meet our user goals and analysis tasks (Section 4). The frontend of DEFORESTVIS is developed in JavaScript using Vue.js [vue14], D3.js [D311], and Plotly.js [plo10]; the backend is written in Python using Flask [Fla10] and Scikit-Learn [PVG\*11].

**Views and workflow:** To keep—as small as possible—the number of views needed for a system that visualizes surrogate models, DEFORESTVIS has five main views (Figure 1): (a) surrogate model selection (→ AT1 and AT2), (b) behavioural model summarization (→ AT3), (c) rule overriding, (d) comparing decisions (→ AT4), and (e) test set results (→ AT5). These views support our workflow in Figure 2: (i) build several surrogate models with increasing complexity by including more decision stumps in architectures (Figure 1(a)); (ii) select a surrogate model with low-complexity and high-fidelity to fit the one’s desired precision (Figure 3(a)); (iii) analyse the behaviour of the target model by exploring the summarized *decision threshold* per feature from the weighted decision stumps (Figure 1(b)); (iv) examine both local and global impact of overriding an automatically produced rule by comparing decisions while adjusting the threshold value (Figure 1(c) and (d)); and (v) observe the influence of the final surrogate model on unseen data and optionally reason why a test case was classified as a given class based on the contribution of each feature (Figure 1(e)). By repeating the workflow in Figure 2, the user gains knowledge about what the target model has learned from the data and, in addition, can fine-tune the target model’s decision-making via the surrogate model.

**Implementation details:** DEFORESTVIS uses the state-of-the-art ensemble learning Explainable Boosting Machine (EBM) approach [NJKC19]. Yet, our workflow is model-agnostic since AdaBoost-based surrogate models can approximate the behaviour of any ML model. We chose EBM intentionally because this algorithm produces systematically fewer decision rules compared to other ensemble learning methods that we have experimented with, for example, XGBoost [CG16] or Random Forest [Bre01]. For all our experiments (including the use cases in Section 6), we use EBM



**Figure 2:** The DEFORESTVIS workflow enables users to choose the preferable surrogate model according to their willingness to sacrifice fidelity in favor of less complexity, explore the decisions extracted from the surrogate model (which serves as a simplified representation of the complex ML model), and manipulate individual rules based on the visual feedback and their prior experiences. To close the loop, users can reason about specific test cases iteratively while exploring the already-existing exported explanations for the training data. Steps 51–56 showcase a simple single-iteration loop.

with the default hyperparameters as the target model, and we further split data into 80% training and 20% testing with a stratified strategy (i.e. keeping the same balance in all classes for both sets). We randomly sample the hyperparameter space (50 Random Search iterations) in order to visualize each AdaBoost model with an increasingly larger number of decision stumps ( $n\_estimators$  hyperparameter). We used the default AdaBoost hyperparameters [Sch99] except for the maximum number of features ( $max\_features$ ) to use when looking for the best split, which we set to the square root of the number of features. For more technical implementation details, we refer the reader to our source code repository [Def23]. We describe DEFORESTVIS by an example with the *breast cancer (Wisconsin)* data set [DG17] (699 samples, 9 features, 2-class classification task: *benign, malignant*). All our data sets are normalized to [0, 1].

### 5.1. Surrogate model selection

After creating 50 surrogate models by gradually increasing complexity, we use DEFORESTVIS to show their training-prediction accuracy with a lollipop plot (y-axis: fidelity; x-axis: complexity; see Figure 1(a.1); AT1). In DEFORESTVIS, *fidelity* is defined as the accuracy with which every surrogate model can simulate the target model [DCB19], while *complexity* is the number of decision stumps in each surrogate model (AT2). The top circles in the plot show threshold precision—defined as how much rounding decimal digits affects the surrogate’s fidelity—via a greyscale colourmap (light grey: less precision; black: maximum precision; AT1).

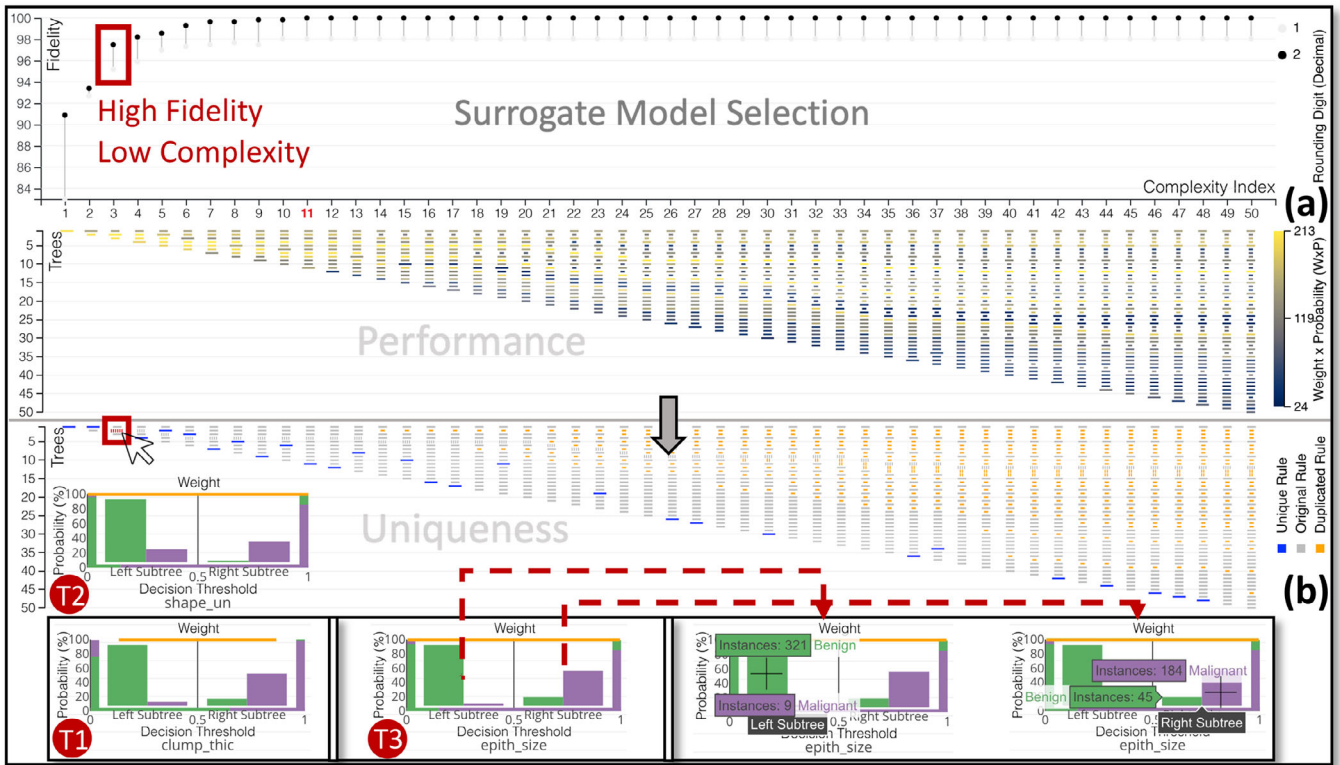
The dot plot with lines in Figure 1(a.2) below the lollipop plot shares the same x-axis as the lollipop plot. Its lines show decision stumps included in every surrogate model (AT2). These are coloured to show either (1) performance (using the colourblind-friendly version of Viridis colourmap, see Figure 1(a.2)) or (2) the

uniqueness of a decision stump (see Figure 3(b)). The first option encodes the weight  $W$  of each rule multiplied by the predicted probability  $P$  of all instances to belong to the Ground Truth (GT) class. The second option scans the space of surrogate models from fewer to more decision stumps being produced. Longest, blue lines are unique decision stumps (or rules); “original” rules found in an earlier smaller surrogate model which exist just once in the current surrogate model are grey; and duplicated rules found twice or more in the same surrogate model are orange, narrow lines. Next, users can choose between the two visual encoding modes and click on a line to bring up a pop-up with the decision stump/rule it encodes on the left-hand side of the dot plot (Figure 1(a.3) and Figure 3(b)). When doing this, the solid lines encoding the same (in terms of feature and threshold) decision stump in all surrogate models change to dashed lines, with the most-common decision stumps globally located at the top of each stack of decision stumps. This lets users quickly explore *unique and influential* decision stumps that *should* be part of their surrogate model, and choose the appropriate stumps for their surrogate model. Another use case is to get inspiration from a more complex surrogate model on how to modify a decision stump of a less complex surrogate model. Users can next select another surrogate model than the default (highest-fidelity, lowest-complexity) one to analyse. The currently selected surrogate model (Figure 3(a), complexity index #11) gets its index label marked red.

We compactly visualize a decision stump (Figure 3(b), 12) by a four-component visual design that shows all relevant information of AdaBoost’s one-level decision trees. Each stump uses a single feature to cut the training instances into two subtrees. The top orange bar expands left and right from the middle to show the weight (influence) of a single AdaBoost stump. In the remaining visual components, green maps one class and purple the other (in our case benign and malignant cancer, respectively). The bottom bar shows the *decision threshold value* (range 0 to 1) that separates training instances below that threshold to either class; in our case, this threshold is around 0.15—below this value, 12 predicts the green class. The left stacked bar on the boundary shows the predicted probability (or confidence) with which the decision tree suggests that the training samples of the left subtree are in one or the other class. For 12, almost 100% probability is in favor of the green class; for the right subtree, approx. 80% suggests the purple class and the remainder the green class. The bar chart in the middle shows the distribution of training samples (i.e. the exact number of instances) that fall into the left or right subtree (colour encodes GT class, see above). In 12, most training samples of the left and right subtrees are correctly predicted as the green, respectively purple, classes.

**Design discussion.** An alternative to the lollipop and dot plots shown in Figure 1(a) is using a scatterplot to visualize the balance between fidelity and complexity, as suggested in [MLMP18]. This method could also help users to seek solutions on the Pareto frontier. A scatterplot might also scale better with the sample count. However, a scatterplot has a few drawbacks: (1) it can complicate the selection and exploration of *individual* decision stumps; (2) it could obscure the sequential relationship between stumps; and (3) it may not effectively show the rounding effect on the surrogate model’s accuracy and require additional designs to show such information.

**Design scalability.** To improve the scalability of DEFORESTVIS, a solution is to separate the visualizations in Figure 1(a) into a



**Figure 3:** Exploration of the complexity-fidelity trade-off with DEFORESTVIS. The lollipop plot (a) shows the fidelity score against the complexity index for the incremental increase in number of estimators hyperparameter of the surrogate models created by AdaBoost. Rounding decimal digits that could result in information loss are also visible here. The highlighted surrogate model has only three one-level decision trees but can emulate over 97% the target model. The dot plot with lines in the same view shows the contribution of each decision to the final result. The higher the value of the weighted probability (weight  $W \times$  probability  $P$ ), the more important a rule is. View (b) shows an alternative encoding with newly discovered decision stumps having larger width and coloured blue. Already found rules which are still part of the next model are in grey; after they are included multiple times, they become duplicated rules (orange, smallest width). The user clicks a rule of interest to inspect (red).

different tab specifically conceived for selecting the optimal surrogate model. Additionally, our choice of representing a decision stump by a four-component visual design is to save space and encode diverse information compared to, for example, more traditional node-link diagrams.

### 5.2. Behavioural model summarization

This view includes a grid of stumps at the bottom that is summarized with a segmented bar chart on top, and it is the most important view that intentionally has the largest size because the other views are connected with this behavioural model summarization central view.

The decision stumps in Figure 1(b.2) follow the same design as the active stumps explained before, except we replace the bar chart with a colour-coded grid showing training samples. The goal behind the active stumps is to get inspired from other more complex surrogate models to override some of the decision stumps of the selected surrogate model accordingly. In contrast, the grid we use here enables the in-depth exploration of which subtree each sample falls into (AT3). Every grid contains two cell groups, one to the

left, one to the right, ordered top-to-bottom by the predicted probability of each sample belonging to the GT class, both groups separated by some whitespace in the middle of the grid. Misclassified samples are shown as cells closer to the grid middle, with colours mapping the GT of each training sample. Each feature is encoded by a series of decision stumps, with the most important features (having a higher sum of weighted probability for all decision stumps jointly influencing them) listed first, for example, *Glucose* and *Insulin* in Figure 1(b.1).

For each feature, the top part of Figure 1(b.1) shows a segmented bar chart summarizing all its decision stumps, with one segment (rectangle) added per threshold value (AT3). For example, if two decision stumps exist with different thresholds, the segmented bar chart will be split into three rectangular segments, while for one stump, the information represented is equivalent to the decision stump itself. Therefore, the chart's x-axis encodes the all available thresholds that separate predictions into different classes; the y-axis shows weighted probability for the most probable class above the zero line, and weighted probability for the other class on the negative side (i.e. rectangles below zero). When hovering over a segment,

the corresponding decision stump is highlighted to show how, and with what magnitude, the stump votes. The most impure decision stump (based on the *Gini impurity measurement* [GPTM10]) is selected by default and marked in red. High impurity is problematic because it indicates a poor separation between classes.

**Design discussion.** Although the detailed stump-based explanation grid may initially seem complex, it is essentially a compact visualization for each decision stump, depicting the (1) threshold for each decision, (2) predicted probability for the left condition, (3) predicted probability for the right condition, (4) weight of each stump, and (5) training instances with the GT class. Despite its complexity, the grid can be easily understood by breaking it down into five *design zones*. Also, the visual encoding and colour scales shared by this grid and the projection view help users decode the grid visual representation.

**Design scalability.** While our current grid design is quite functional, its scalability to large data sets might be challenging. To overcome scalability issues regarding the display of many instances in the decision stumps (Figure 1(b.2)), DEFORESTVIS reduces each box to plot each training sample by a single pixel. A benefit of this chosen fine-grained grid design (of raw data) is the sorting of instances according to how easy or hard it is for an instance to be classified in the opposite class. However, if more samples exist than available pixels, we could aggregate or group similar samples into larger boxes [EF10, Mun14]. While fully scalable to any instance count, this solution would make it harder to drill down to single problematic training instances. In the extreme case, when having thousands of instances and features, this view could be completely replaced by the segmented bar chart. We suggest further solutions to this problem and other potential limitations found by the experts in Section 7.

### 5.3. Rule overriding

The bar chart (Figure 1(c.1)) shows impurities with the weighted probability score for each stump on a colourblind-friendly Viridis colourmap once more (Section 5.1; AT4). The y-axis shows the identification index of the stumps with rules, highest impurity at the top. The currently selected stump is marked red (Section 5.2). In Figure 1(c.1), for instance, the 6<sup>th</sup> stump has the lowest impurity value (close to -50) but its weighted probability score is low, see its dark blue colour and short bar length.

The histogram in Figure 1(c.2) groups training instances into 10 or 20 bins, depending on the user-chosen decimal precision. When more precision is desired, for example, Figure 1(c.2), the highest value among the bins is used (AT4). The dashed black vertical lines show the threshold value before, respectively after, the user's interaction with the currently active decision stump (marked red). GT classes are colour-coded in green and purple, like in the other views.

### 5.4. Comparing decisions

Figure 1(d) helps to study further one's change of a rule's threshold value. We use here a UMAP projection [MHM18] of all training samples (AT4). Samples are coloured by class with brightness mapping the local relationship of these samples with the 6<sup>th</sup> decision stump that is the purest one and relates to the most important

feature, that is, *Glucose*. Bright and dark colours show samples in the left, respectively right, subtree of the selected decision stump only (i.e. local investigation). Finally, each input dimension taken for UMAP encodes one decision stump of the investigated surrogate model and equals 0 for samples in the left subtree of the stump, else 1. In Figure 1(b.2), we thus have eight dimensions since the studied surrogate model has eight stumps that are being reduced to two dimensions (i.e. global investigation). As such, samples identically classified by all stumps will be positioned close to each other.

When users change the threshold value, the projection updates. We show how points move in the projection (before *versus* after the change) by lines, so users can understand how their changes in one rule impact specific samples *locally* for a specific stump and *globally* for all sample pairs falling into different subtrees of all stumps (AT4).

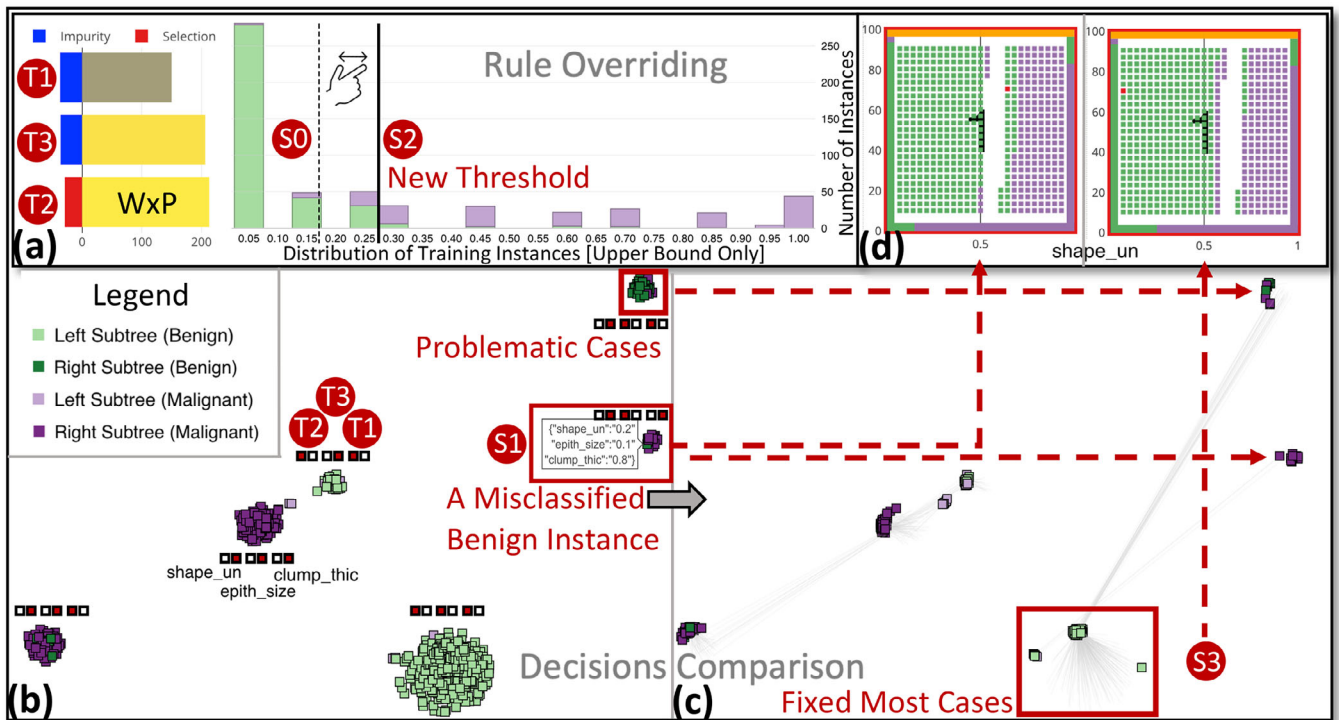
**Design discussion.** For the UMAP projection, we have tried using shape instead of colour for visualizing if a sample belongs to the left or right subtree of the selected decision stump in earlier design iterations of DEFORESTVIS. But, with that design, it was almost impossible to correctly see if instances belong to the left or right subtree.

**Design scalability.** To limit overplotting in a projection, Collaris and van Wijk [CvW22] recommended a restriction to a maximum of 5000 instances. This indication can also be applied to our UMAP projection view. We relax this limit by allowing zooming and panning our projection view. The use of techniques to reduce clutter in projections and edge bundling for the lines showing the trajectory of points can further improve overplotting [HMJE\*19, YXX\*21].

### 5.5. Test set results

To test if the user's changes do not overfit the surrogate model, our VA tool shows the ground truth (*GT*) and predicted (*Pred*) results for every test sample (Figure 1(e) and Figure 5(a); AT5), with each table row being a test case. The brown  $\Delta(W \times P)$  column shows the difference in weighted probability needed to switch the prediction from one class to the other. A low  $\Delta(W \times P)$  value means low classification confidence for that particular test sample. The bar charts to the right of  $\Delta(W \times P)$  show the contributions of each feature proportional to all features (summing up to 100% pixel length of the view; see the values in Figure 5(b)) for the prediction of each test sample. Colour encodes the predicted class (green or purple). Colour saturation double-encodes the *influence level per class* to highlight the most impactful features for a given test instance, with fully opaque bars displaying high confidence, suggesting that the weighted probability is higher for those features than others (AT5).

**Design discussion.** If perceptual complexity was not an issue for the bar charts to the right of  $\Delta(W \times P)$ , we could alternatively use colour saturation to map the prediction boundary difference. For example, a fully opaque bar would mean small adjustments to the decision threshold of the decision stump(s) relevant for this feature will lead to a change in the predicted class outcome (serving a similar purpose as the red lines in Figure 5(c)). In summary, our VA tool leverages simple yet effective visualizations that ML experts are already accustomed to, such as bar charts, histograms, and projections. This was also confirmed by the ML experts from our interview



**Figure 4:** Impact of manually overriding the most-contributing rule in a low-complexity (index #3) but high enough fidelity scenario (shown in Figure 3(b)). In (a), the user clicks on rule  $\textcircled{2}$  (cf. Figure 3) and, after studying the distribution of training samples on each side of the subtree, decides to adjust the threshold (from  $\textcircled{0}$  to  $\textcircled{2}$ ). The initial state (b) has some problematic cases, marked  $\textcircled{1}$ ,  $\textcircled{2}$ ,  $\textcircled{3}$ , belonging to the right subtree of  $\textcircled{2}$ , left subtree of  $\textcircled{3}$ , and right subtree of  $\textcircled{1}$  based on the manual inspection by hovering over each of them (see small red boxes). These are benign samples mixed with malignant ones. Even though the threshold change  $\textcircled{2}$  adversely affects the classification of malignant cases, its benefit overcomes the default suggestion. View (c) confirms this: the trajectory of points shows a better separation of the previously highlighted sample groups. After the change  $\textcircled{2}$ , the hovered point (among others) has moved from the right subtree to the left, which is correct (see (d)).

sessions, as demonstrated in Section 7. By using such familiar techniques, we aim to minimize the learning curve for new users.

**Design scalability.** If the table’s scalability becomes an issue, a solution is to group similar samples and visualize only the misclassified test instances [EF10, Mun14]. However, the current granularity level of DEFORESTVIS has many benefits, including being able to explain *specific test instances* and trace back *raw data instances* to the surrogate model’s internal parts.

## 6. Use Cases

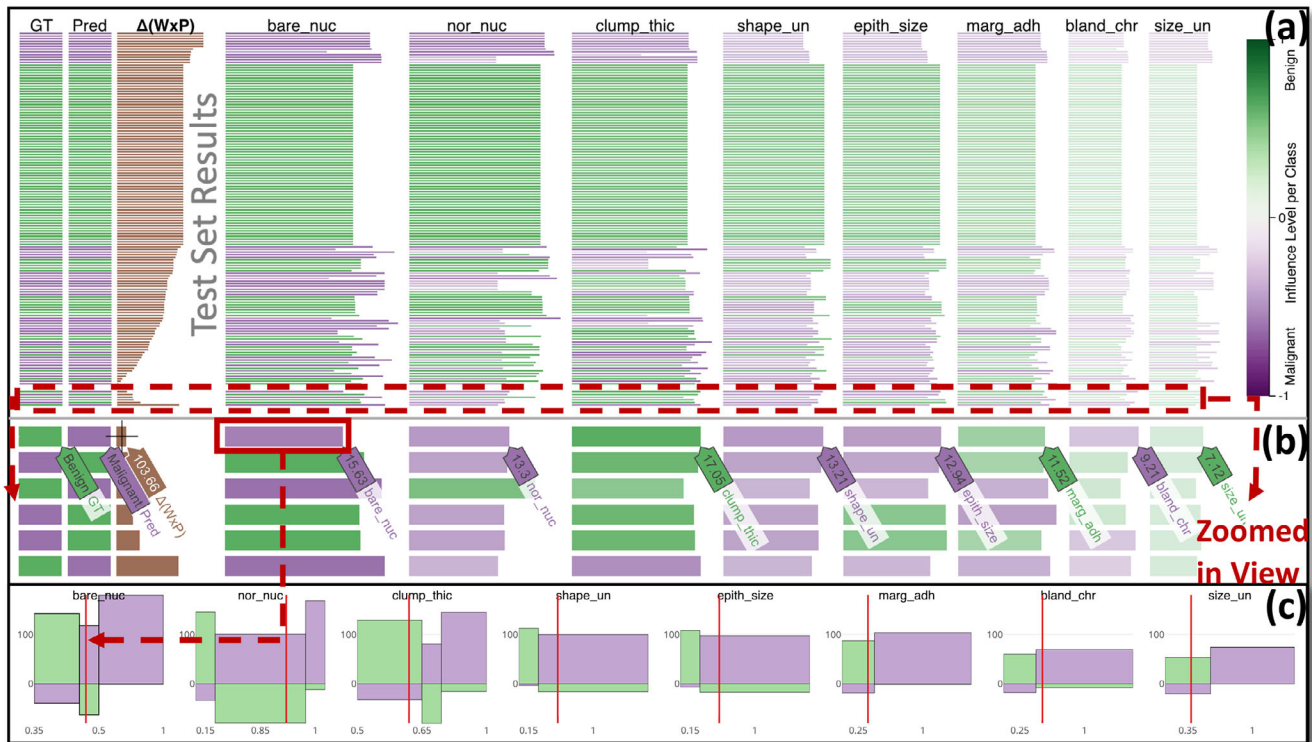
We next present a use case and a usage scenario showing how DEFORESTVIS evaluates the behaviour and summarizes the knowledge generated from a complex ML model (see also Supplemental Material).

### 6.1. Use case: Additional analysis support

We next discuss additional exploration features of our VA tool, using the same data set as in Section 5. Although tools like RuleMatrix [MQB19], DRIL [CB20], and the one designed by Di Castro

and Bertini [DCB19] (see Section 3) provide feature importance and tree exploration insights similar to DEFORESTVIS, our proposed tool allows users to vary the complexity level of the surrogate model in different ways to explain with only a few trees an entire complex model, resulting in an adaptive design that may be more practical to real-world scenarios, as motivated in Section 1, supported by Yuan et al. [YBOB22], and the ML experts in Section 7.

**Exploring dynamics of different surrogate models.** In the scenario presented so far to illustrate the tool’s views, we observe from Figure 3(a) that rounding the thresholds of the decision stumps to two decimals leads to the highest possible fidelity in all 50 surrogate models, which already to a certain extent minimizes the users’ cognitive load (AT1). The surrogate model with complexity index #11 (Figure 3(a), marked red), containing 11 decision stumps, already explains 100% of the original model’s behaviour. In the dot plot with lines of Figure 3(a), we see that the weighted probability value for many newly produced/unique rules (for surrogate models after model #11) substantially decreases to  $\approx 24$  (indicated by dark blue colours appearing for the unique rules with lengthy bars in Figure 3(a)). We next choose to explore a simplified surrogate model (the selection is marked with the blue box in Figure 3(b)) composed of only three decision stumps since this reduces



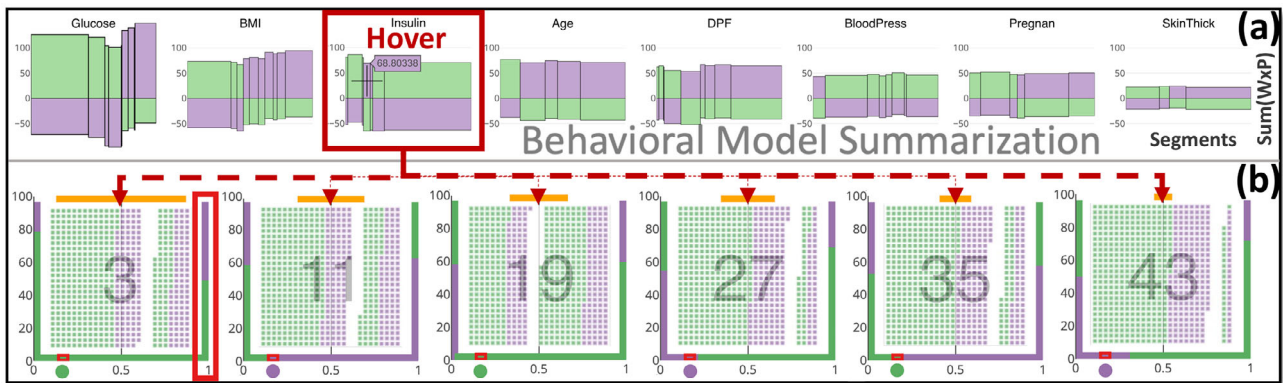
**Figure 5:** Local analysis uses the most accurate (and as simple as possible) surrogate model (complexity index #11, Figure 3(a)) to predict a test case. (a) shows the certainty of the surrogate model classifying the test set. Test cases at the bottom are misclassified, with the easiest-to-swap class instance drawing the user’s focus, see (b). We want to find the feature that contributes the most to this case being misclassified as malignant. After *clump\_thic*, which moves the final prediction toward the benign class, the next most contributing feature is *bare\_nuc* based on the length of the bars and the strong opaque colour revealing the influence level of the features. A threshold increase from  $\approx 0.35$  to  $\approx 0.45$  would swap the prediction for this test case, as shown in (c). Yet, this fluctuation can harm generalizability since some malignant cases would fall in the left/wrong subtree.

complexity drastically (by almost 73% if we count the number of decision stumps in each surrogate model) and fidelity remains above 97% (AT2). We select the unique decision stump (identifiable by its larger width) introduced in this specific surrogate model—marked with dashed red lines in Figure 3(b) (AT3). This stump has the maximum possible weight (see the lengthy orange bar on top of ⑫), thus it is a *core rule* for the *shape\_un* feature. The pop-up ⑫ tells that, when the value is lower than 0.15, the probability of samples being classified as benign is very high according to the left subtree. ⑪ has less impact due to its smaller weight, but it classifies malignant (purple) samples with very high probability when *clump\_thic* is above  $\approx 0.6$  (indicated by the purple colour at the bottom bar of the ). Another interesting decision stump is ⑬, with 321 training samples belonging to the benign class and 9 misclassified in this subtree with extremely high predicted probability and confidence (see the tree weight). The right subtree of ⑬ has 184 malignant instances and 45 benign instances with a probability above 80% (compare the purple bar to the green bar on the right-hand side).

**Analysing and overriding a decision rule.** We move on to a deeper exploration of the surrogate model with three decision stumps. In Figure 4(a), we selected the purest, most impactful, decision stump (AT4). In step ⑩, we see that the automatically-

generated stump finds values below 0.15 as benign while the GT in the histogram tells us that the bin between 0.2 and 0.25 has more benign than malignant cases. To confirm this, we study the projection of training samples forming different groups (Figure 4(b)), looking whether the samples belong to the three decision stumps (⑫, ⑬, and ⑪). The problematic (confused) cases are marked by red boxes, with many benign cases being left out in the right subtree instead of the left. One such benign case is marked in ⑪ with *shape\_un* being 0.2. To handle such cases, we change the threshold to a new value of 0.25, step ⑪. This makes the benign (dark green) cases light green and move to the benign cluster (⑬, Figure 4(c) bottom). The highlighted case visible in Figure 4(d) has been resolved by this action and moved from the right subtree to the left one (AT4).

**Testing a borderline test case hypothesis.** In Figure 5(a), we spot test cases that are misclassified by the surrogate model (AT5). How easy it is to change the prediction for borderline cases with the lowest  $\Delta(W \times P)$  visible in Figure 5(b)? The highest-contribution feature is *clump\_thic* (longest bar in Figure 5(b) for the hovered test case) which influences the result in the correct direction (green colour). The second most-contributing feature is *bare\_nuc* 15.63% of the sum of weighted probability for all features. By hovering over this test case, we see that its *bare\_nuc* value is marginally in favour



**Figure 6:** Analysis of decision stumps explaining how slight insulin amounts classify patients as diabetic. Hovering over this feature (a) allows Amy to follow the path of how this aggregated rule was created. To Amy’s surprise, the model appears confused in (b) as there is a balance between the six present decision stumps—three suggest the positive class, the rest suggest the opposite. The largest-weight rule seems divided since the probability of its right subtree is 50% (red box), so it eliminates itself from the prediction outcome. The remaining rules make the decision. The second more impactful rule favours the negative class. Given this analysis, Amy could modify the rules to change this strange behaviour (see Section 6).

of the purple (malignant) class with a 0.35 threshold (Figure 5(c)). Adjusting the rule to 0.45 would lead to this test case falling into the green class but could negatively affect other training or test samples. To solve this, we can repeat the analysing and overriding of a decision rule procedure described earlier above.

## 6.2. Usage scenario: Behavioural summary of the target model

Amy—a data analyst in a hospital—got a labelled data set on *diabetes* [SED\*88] with 8 features, 768 samples. She splits the data set into 80% training and 20% test samples. Amy aims to fit a highly accurate (but complex) ML model to the training data and check its prediction ability on the test set. Yet, from her experience, she knows that it is hard to check what complex ML models do learn. She uses DEFORESTVIS to analyse if such a model performs well and also presents to the doctors the main findings using a simpler surrogate model that facilitates their domain expertise injection on the decision rules. Her end goal is to *fully replace* the target model.

**Inspecting an unusual feature of interest.** Amy begins her exploration with the default surrogate model created by DEFORESTVIS, which achieves the highest possible fidelity of approx. 96% (AT1). The surrogate model index #41 contains many decision stumps (Figure 1(a.1)), so interpreting the model’s behaviour is hard (AT2). However, DEFORESTVIS provides a summary of the predictions and confidence levels for all decision stumps extracted for each feature, see Figure 6(a) (AT3). Amy quickly notices that *Glucose*, *BMI*, and *Insulin* are the most important features, and the AdaBoost model has produced many decision stumps for these features (Figure 6(a)). What catches Amy’s attention is the behaviour of the stumps related to the *Insulin* feature: From the six stumps related to this feature, three suggest that the prediction should be positive for diabetes, while the other three suggest the opposite. The 3<sup>rd</sup> decision stump, which has the highest weight value, appears to be divided between the two classes with a probability of about 50% in the right subtree (Figure 6(b)). Due to this strange behaviour, this stump is ruled out, and the 11<sup>th</sup> decision stump provides the next most impactful rule in

favour of the positive class prediction. After investigating the classification rules, Amy can easily override the problematic stump and adjust the model’s behaviour accordingly.

**Improving the surrogate model and communicating the results to domain experts.** Amy wants to communicate her findings to the doctors. Since the currently active surrogate model has many stumps to analyse, which may overwhelm the doctors, she selects the one with complexity index #6 instead (Figure 1(a.1); AT2). This model has only eight stumps, has a fidelity of over 90%. Interestingly, DEFORESTVIS demonstrates that the next five surrogates use 2 decimal digits instead of 4 (grey dots being on top of black) for achieving higher fidelity. To ensure that she has learned the most from other pre-trained surrogate models with above 92% fidelity, Amy checks for unique rules occurring after this AdaBoost model (i.e. having longer bars, see Figure 1(a.2); AT3) and selects the one from model #14 with a moderate weighted probability value (red coloured and dashed lines, see Figure 1(a.2); AT3) because the remaining unique stumps have very low weighted probability (dark blue colour). The decision stump suggests that, for *Glucose*, the threshold should be set to  $\approx 0.55$  so that training samples below get classified as negative (Figure 1(a.3)). This suggestion makes Amy think about the most impactful feature, *Glucose*, which makes the model predict one class or the other, as suggested by the test samples in Figure 1(e) (AT5). Another finding is that *Insulin* in this surrogate model is only positive, leading to fewer negative diabetes cases (fully green for all decisions). The selected 6<sup>th</sup> decision stump (Figure 1(c.1)) is much more in favour of the negative class with a very high threshold for proposing the positive class (Figure 1(b.2)). With this knowledge, Amy decides to decrease the threshold for that stump to make the prediction more flexible (Figure 1(c.2); AT4). The impact is visible in Figure 1(d) (AT4). Amy decides to present this finding to the experienced doctors to get their opinion about this manual threshold change and the behaviour summary (Figure 1(b.1)), as well as to verify the hypothesis that the updated surrogate model performs as they expect (or better than before) and potentially insert their knowledge into it.

## 7. Evaluation

We gathered more feedback on the effectiveness of *DEFORESTVIS* by conducting online semi-structured interview sessions with five experts (**E1–E5**), along the same procedure as in [MXLM20, XXM\*19, CMKK22, CPK23].

### 7.1. Participants

**E1** is an assistant professor with a PhD in mathematics and 7 years of experience with ML, currently developing ML models for reinforcement. **E2** is a senior researcher in a governmental research institute, working with applied ML projects, with a PhD in software engineering, and 6 years of experience with ML. **E3** is a data analyst in a large multinational company working with data engineering, a PhD in informatics, and 6 years of ML experience. **E4** is a data analyst and PhD candidate working with time-series data and anomaly detection with 5 years of ML experience. **E5** is a PhD candidate in deep learning with 5 years of experience in deploying ML models in a large multinational company. **E5** was the only one who reported a colourblindness issue (deuteranomaly), but mentioned having no problem perceiving correctly the colours used in *DEFORESTVIS*. **All experts** were knowledgeable in data visualization and had encountered visual tools at some point during their professional careers.

### 7.2. Methodology

We conducted individual interview sessions online via Zoom using a large PC screen in full-screen mode, with the experts' participation being completely voluntary. Each interview session lasted about 1 h and 30 min and was structured as follows: (1) present the core research goals of *DEFORESTVIS*, its analytical tasks (Section 4) and workflow (Section 5); (2) explain the functionality of every view and the steps taken to arrive at the results in Section 6.1; and (3) interact with the tool on a newly-introduced data set (heart disease diagnosis [DG17]), similarly to the demo video accompanying this paper. The goal of phase (3) was not to accomplish a specific task, but rather to elucidate the interconnections between each view and to explore the potential capabilities of our system with this simplistic healthcare data set. In this formative evaluation, experts were asked to provide their opinions on the four aspects summarized in Sections 7.4–7.7 by following a think-aloud protocol.

### 7.3. Overview

The feedback we received was positive and supported the use of *DEFORESTVIS* for surrogate modelling. **E4** highlighted the issue of adding more and more rules when using tree-based surrogates, making it rather quickly almost impossible to examine each one. A summarization of which trees are more interesting in the behaviour model summarization and decisions comparison are great additions of *DEFORESTVIS*, offering users further guidance (**RQ1**). **E1** and **E3** deemed our VA tool suitable for real-world applications and also educational purposes (the latter since we visually explain how Adaboost works). **All experts** were impressed and expressed confidence in the advantages of using *DEFORESTVIS*, especially praised for the transparency our VA tool offers from multiple levels (top-down, in-between, and bottom-up) and for various users (**RQ2**).

### 7.4. Workflow

**Somewhat intensive to learn but usable after training.** **E1** and **E4** praised the conceptual workflow of our tool, going from a broader view (their favourite panel is shown in Figure 1(a)) to more fine-grained views. They both mentioned that the learning curve was steep. Interestingly, **E1** felt confident that he could understand the tool without a training session. **E4** suggested that some indicators could guide users on where to look first, but the training we provided was sufficient to make him understand how the tool works.

**Diverse workflow steps for different users.** **E2** and **E4** recommended having a model developer work together with a domain expert to enhance collaboration, which is an important aspect for visualization tools in general beyond *DEFORESTVIS*. The top-down approach of selecting and tuning the surrogate model was more appropriate for model developers and data analysts (see Section 6); the bottom-up approach was found relevant for domain experts (i.e. starting from Figure 1(e)). The rule-overriding related views (Figure 1(c) and (d)) serve as the middle ground, enabling developers and experts to collaborate (**AT4**). **E4** suggested that domain experts could have better understood the overriding rule process if they had focused on feature-based modifications rather than decision stumps—a hypothesis that should be tested in the future. Still, the benefit of working with decision stumps is that it allows for micromanagement for each stump, which sometimes gets combined with other decision stumps (targeted toward data analysts and model developers). Here, **E4** pointed out that domain experts might need to change decision stumps based on their previous knowledge, that is, achievable by rule overriding and decisions comparison views in *DEFORESTVIS*; and **E2** recommended that prior knowledge of Adaboost might be required to explore the whole process (e.g. the background information provided in Section 2).

**Extra verification loop.** Finally, **E2** proposed comparing the model-based extracted distributions against the actual data distribution to ensure that the model behaves as expected to reassure users to trust the results of the selected surrogate model. This topic is timely and requires further investigation by the visualization community (e.g. see Kale et al. [KGQ\*24]).

### 7.5. Visualization and coordination

**Encouraging results about the novelty of DEFORESTVIS.** **E4** stated that the interaction between performance and uniqueness (cf. Figure 1(a), top-right toggle) is powerful as it helps find decision stumps that are less influential and duplicated, showing they are uninteresting for users and guiding them to select a surrogate model (**AT1** and **AT2**). “The model is simplified by reducing the number of one-level decision trees and rounding the threshold values to fewer decimals, which is incredible!”, said **E3** (**AT1**). **All experts** clicked on a decision tree in the dot plot with lines (Figure 1(a)) to propagate to the following surrogate models, giving an idea of when this tree was re-weighted due to a duplicate decision stump, as mentioned by **E1**. **E1** followed up by saying: “This view answers the question of how much complexity should be added before making a simple rule much more complex!” (**AT2**). As an analogy, *DEFORESTVIS* could be viewed as facilitating users in inspecting diverse systematically-reduced dimensionalities, and effectively deciding the “optimal” one for the given problem [JLJ23].

**Improving visualizations.** E1, E2, and E5 liked the fragmented bar chart in Figure 1(e) and found it useful for visualizing data and hypothesizing changes using a bottom-up approach (especially the  $\Delta(W \times P)$  column; AT5). Additionally, E5 stated that, for binary classification, ground truth covers the prediction, so the *Pred* column is not necessary unless it is a multi-class problem. He suggested replacing the column with the predictions made by the targeted model to enable a direct comparison of whether the surrogate and target models agree or not. This idea led us to think about a future research opportunity where the input data are compared against the surrogate model's predictions except solely for the target model to improve directly the former by studying the misclassifications of the latter.

**Familiarity with the tool and coordination between views.** In the projection-based view, two potential drawbacks were found by E3 and E5 (AT4). The first issue is the abstract nature of the UMAP projection, which can make it hard to see prominent clusters of data points as the number of decision stumps increases (E3). A solution to this would be to highlight a specific training instance and see where it belongs to in all decision stumps at once. The second issue is the difficulty of keeping track of left *versus* right subtrees, which can predict different classes in different decision trees (E5). This issue may improve with practice and familiarity with the data set, but also with the hovering functionality that can help explaining in which subtree an instance belongs to for every decision stump.

## 7.6. Interaction

**Interactivity improvements.** E1 said that a lasso selection could be used in the UMAP view to find samples classified similarly in the same subtrees by decision stumps because analysing samples individually in the training set is helpful but fails to directly explain sample clusters (AT3 and AT4). However, he also said that the projection view gives insights into why samples are close and how confused they are. Also, trajectories (gray lines) are useful to show the global effect of local changes if clear clusters are formed by UMAP (see Figure 1(d); T4).

**Details on demand.** E2 said that comparing stump-based weights is easy in pairs but by what exact amount is challenging, which could be improved by showing the precise value on hovering. These are all minor issues to be addressed in a future version of DEFORESTVIS.

**Alternative colour mapping.** Finally, E3 suggested that the colour gradient in Figure 1(a.2) could have been reversed from yellow to blue as yellow was perceived as stronger, but there is a design trade-off with consistency as the same colour gradient is used in Figure 1(c.1), right-hand side (found well designed by them).

## 7.7. Limitations identified by the experts

**Scalability issues.** E2 and E5 were concerned about our tool's (1) *scalability versus the number of features*; E3 pointed the issue of (2) *adding more training and testing samples*; E4 pointed the issue of incrementally exploring (3) *additional decision stumps*. For (1), E2 and E5 agreed with using AdaBoost as a surrogate model to fit target-model predictions as it automatically generates decision stumps only for important features. This is affected by the number

of decision stumps of the selected surrogate model. For (2), E3 proposed to aggregate similar training samples to scale the decision-comparison view and a bar chart similar to Figure 1(a.3) instead of the sample grids (Figure 1(b.2)), and then doing the same for testing samples with binning test cases according to how they were predicted and how hard it is to force them to swap classes. E5 suggested the progressive visual exploration of the data and features to overcome such issues. For (3), E4 said that, with the surrogate model selection panel (Figure 1(a)), users would typically select fewer decision stumps that are easily explorable, and worst case, explanations using decision stumps could be replaced by more segmented bar charts to show more features (also solving the scalability issue).

Overall, DEFORESTVIS has limited scalability concerning the instance and feature count, that is, it can handle, conservatively, up to several thousands of instances having several tens of features. However, similar to other VA tools, DEFORESTVIS does not aim to offer hundreds or thousands of individual features to the user to construct an explanation. It thus falls in the group of tools that address so-called tabular data, which consists of a relatively small number of features that have a clear meaning and identity that experts want to see and reason about. The other group of tools address inherently high-dimensional data, for example, with thousands of dimensions that do not all have a clear meaning, and which one does not typically want to reason about explicitly, thus visualize explicitly, as common in deep-learning contexts.

**Beyond binary classification.** E1 noted that multi-class classification is perhaps the most limiting factor when using AdaBoost as a surrogate model (simple if-else decision stumps), but a binary one-versus-rest strategy could be used. Despite that, E1 thought that the proposed visual designs are not bounded by this same restriction, especially if users focus on the segmented bar charts that summarize the predictive power and thresholds for each feature (Figure 1(b.1)). However, in general, many related works focus only on binary classification problems according to Chatzimpampas et al. [CMJ\*20], that is, the same category DEFORESTVIS falls into.

**Held-out test set.** E5 mentioned that there is a danger of bias if users tune the model to improve against the test cases presented in our tool. E1 also mentioned that users should only use the bottom-up approach for explanation purposes and not for tampering with the training process. An external data set may be needed if our tool is used to tune the training of a surrogate model to fit well a test set, according to E5.

## 8. Discussion

In this section, we describe the intended target users and the general limitations of our VA tool beyond the ones identified by the experts.

### 8.1. Targeted users

DEFORESTVIS aims to assist model developers in understanding the behaviour of their models in order to further optimize them, and also help domain experts to use simplified and tunable surrogate models for prediction instead of a more complex ML model. By visually examining surrogate models, developers can gain insights into how these surrogates represent the original model, which can help them

improve both models. Visual examination of surrogate models can also help developers to identify influential features, detect interactions between features, and make other adjustments to improve the accuracy and reliability of the complex ML model, for example, by informing future data collection cycles. For data analysts and domain experts, visualizations of surrogate models can provide an intuitive understanding of how these models behave and the relationships between the input variables and the output predictions. This can help to identify patterns, anomalies, and other important features in the data that may not be immediately apparent from numerical summaries or other types of analyses. Additionally, tree- and rule-based visualizations can assist data analysts in communicating their findings to domain experts in a clear and compelling way [YNB21]. As highlighted in Section 7, the five ML experts who engaged in our interview sessions, each lasting 1 h and 30 min, successfully comprehended the core principles and managed to use DEFORESTVIS. An additional opportunity that emerges from this is the development of a simplified version of our tool, specifically designed for novice ML users with limited visualization literacy.

Concretely, from our overall observations during the expert interviews (see Section 7), it is arguable that DEFORESTVIS has a relatively steep learning curve—at least if one aims to master all its offered features. Indeed, our tool consists of five views (see Figure 1 and Figure 2), each of them offering internally interactive sorting and/or selection mechanisms which, when activated, affect all the other linked views. Apart from the limitations exposed by our expert interviews, including the users' suggestions for improvement (Section 7), we see further possibilities to improve DEFORESTVIS to reduce its learning curve:

- Add *explanatory tooltips* on the interactable elements in each view to tell users which are the available controls therein and what is their purpose. This should remind users what they can do within each view.
- Use a *recommender system* that analyses the last actions of the user and next recommends other views/controls in the tool that would naturally follow the given actions. For example, when a user has selected a surrogate model (Figure 1(a)), the tool can suggest, by means of a highlight, the examination of other surrogate models with unique decision stumps. This should guide users through a typical workflow based on their previous actions.
- Offer the use of *presets* during a tool start-up dialog allowing setting problem-specific constraints. For example, users may never want to have a fidelity of less than 90% of the targeted model or to explore more than a certain number of decision stumps. These constraints can next be used by DEFORESTVIS to filter out, from all views, the data that does not fit them, leading to simpler, more targeted visualizations.

## 8.2. Limitations and potential extensions

We next discuss several limitations of our proposal as well as potential improvements to alleviate these.

**Different fidelity measures.** Fidelity in eXplainable AI (XAI) refers to how well a surrogate model can emulate a complex, black-box model and aims to evaluate the surrogate's prediction ability as a whole [VOMS21]. A prevalent measure for global fidelity (also

used by us) is the overall prediction accuracy of the surrogate model. Other global fidelity measures include the R-squared measure for regression tasks [CW97]. While intuitive and straightforward, such measures may fall short in cases with imbalanced data or varying misclassification costs. Measures can also be internal, for example, assessing the surrogate's representation of the target model's decision boundaries [VOMS21]. However, such measures are relatively more complicated to compute and visualize. We next consider extending our approach to incorporate such more complex measures in a visually intuitive way.

**Beyond weak models.** Although the benefit of using decision stumps is their simplicity, this forces each model to be a weak predictor that may not capture all relations in the data. As each stump only considers one feature at a time, the interaction between features might be ignored, leading to less accurate predictions in situations where feature interactions are critical. Hence, the effectiveness of the explanations might be limited, as is the case with GAMs [HT86] and neural additive models [AMF\*21] (a variant of GAMs for neural networks). Since the boosting procedure tries incrementally improving the model by focusing on instances misclassified in previous iterations, users can more easily intervene by adapting the individual stumps and observing their impact on the entire surrogate model [WKN\*22]. Ways to visualize more complex decision trees while the experts still wish to explore and modify them are yet to be found.

**Backpropagating the surrogate's changes to the target model.** While, in some cases, the surrogate model may completely *replace* the target model (if accurate enough and since easier to explain and tune), in other cases one would like to incorporate the users' manual adjustments and insights into the target model. It is important to note that our current approach does not include this feedback mechanism. Rather, our emphasis has been on developing a more accessible *alternative* to a complex model, that is, a surrogate model that domain experts can better comprehend and manipulate. Incorporating a feedback loop that can help refine the target model based on surrogate threshold adjustments is, to our knowledge, a generally unaddressed, but very important, challenge for future research.

## 9. Conclusion

In this paper, we present DEFORESTVIS, a VA tool that helps the analysis of black box, complex, ML models. We use decision stumps (one-level decision trees) from the AdaBoost algorithm, which are easily interpretable, to produce surrogate models that approximate the behaviour of a target model. Our tool's linked views help users iteratively find a balance between complexity and fidelity while minimizing the precision (measured as number of decimals) used in the decision thresholds without sacrificing accuracy. Additionally, DEFORESTVIS enables users to explore decision stumps in many ways, including their purity and impactfulness, and summarizes the behaviour of the surrogate model by aggregating the predictive outcomes and power of all decision stumps related to each data feature. Users can override rules and compare them on local and global scales. Also, users can focus on particular test cases and use the extracted rules to understand why these cases got their particular classifications. We evaluated the applicability and effectiveness of DEFORESTVIS using real-world data sets and by conducting expert

interviews with five expert data analysts and model developers. Our results show the benefits of using DEFORESTVIS for ML model analysis but also highlight potential limitations that we aim to address in future work.

## Acknowledgements

This work was partially supported through the ELLIIT environment for strategic research in Sweden.

## References

- [AEK00] ANKERST M., ESTER M., KRIEDEL H.-P.: Towards an effective cooperation of the user and the computer for classification. In *Proceedings of the Sixth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining* (2000), KDD '00, ACM, pp. 179–188. <https://doi.org/10.1145/347090.347124>
- [AF22] ANTWEILER D., FUCHS G.: Visualizing rule-based classifiers for clinical risk prognosis. In *Proceedings of the IEEE Visualization and Visual Analytics* (2022), VIS '22, IEEE, pp. 55–59. <https://doi.org/10.1109/VIS54862.2022.00020>
- [AMF\*21] AGARWAL R., MELNICK L., FROSST N., ZHANG X., LINGERICH B., CARUANA R., HINTON G. E.: Neural Additive Models: Interpretable machine learning with neural nets. In *Advances in Neural Information Processing Systems* (2021), vol. 34, Curran Associates, Inc., pp. 4699–4711.
- [BKSS14] BEHRISCH M., KORKMAZ F., SHAO L., SCHRECK T.: Feedback-driven interactive exploration of large multidimensional data supported by visual classifier. In *Proceedings of the IEEE Conference on Visual Analytics Science and Technology* (2014), VAST '14, pp. 43–52. <https://doi.org/10.1109/VAST.2014.7042480>
- [BN01] BARLOW T., NEVILLE P.: Case study: Visualization for decision tree analysis in data mining. In *Proceedings of the IEEE Symposium on Information Visualization*, (2001), INFOVIS '01, IEEE, pp. 149–152. <https://doi.org/10.1109/INFVIS.2001.963292>
- [Bre01] BREIMAN L.: Random forests. *Machine Learning* 45, 1 (Oct. 2001), 5–32.
- [BvLH\*11] BREMM S., VON LANDESBERGER T., HEB M., SCHRECK T., WEIL P., HAMACHER K.: Interactive visual comparison of multiple trees. In *Proceedings of the IEEE Conference on Visual Analytics Science and Technology* (2011), VAST '11, IEEE, pp. 31–40. <https://doi.org/10.1109/VAST.2011.6102439>
- [CB20] CAO F., BROWN E. T.: DRIL: Descriptive rules by interactive learning. In *Proceedings of the IEEE Visualization Conference* (2020), VIS '20, pp. 256–260. <https://doi.org/10.1109/VIS47514.2020.00058>
- [CD19] CAVALLO M., DEMIRALP C.: Clustrophile 2: Guided visual clustering analysis. *IEEE TVCG* 25, 1 (2019), 267–276.
- [CG16] CHEN T., GUESTRIN C.: XGBoost: A scalable tree boosting system. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining* (2016), KDD '16, ACM, pp. 785–794. <https://doi.org/10.1145/2939672.2939785>
- [CLG\*15] CARUANA R., LOU Y., GEHRKE J., KOCH P., STURM M., ELHADAD N.: Intelligible models for healthcare: Predicting pneumonia risk and hospital 30-day readmission. In *Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining* (2015), KDD '15, ACM, pp. 1721–1730. <https://doi.org/10.1145/2783258.2788613>
- [CMJ\*20] CHATZIMPARPAS A., MARTINS R. M., JUSUFI I., KUCHER K., ROSSI F., KERREN A.: The state of the art in enhancing trust in machine learning models with the use of visualizations. *Computer Graphics Forum* 39, 3 (June 2020), 713–756.
- [CMJK20] CHATZIMPARPAS A., MARTINS R. M., JUSUFI I., KERREN A.: A survey of surveys on the use of visualization for interpreting machine learning models. *Information Visualization* 19, 3 (July 2020), 207–233.
- [CMK20] CHATZIMPARPAS A., MARTINS R. M., KERREN A.: t-viSNE: Interactive assessment and interpretation of t-SNE projections. *IEEE TVCG* 26, 8 (Aug. 2020), 2696–2714.
- [CMK23] CHATZIMPARPAS A., MARTINS R. M., KERREN A.: VisRuler: Visual analytics for extracting decision rules from bagged and boosted decision trees. *Information Visualization* 22, 2 (2023), 115–139.
- [CMKK21] CHATZIMPARPAS A., MARTINS R. M., KUCHER K., KERREN A.: StackGenVis: Alignment of data, algorithms, and models for stacking ensemble learning using performance metrics. *IEEE TVCG* 27, 2 (2021), 1547–1557.
- [CMKK22] CHATZIMPARPAS A., MARTINS R. M., KUCHER K., KERREN A.: FeatureEnVi: Visual analytics for feature engineering using stepwise selection and semi-automatic extraction approaches. *IEEE TVCG* 28, 4 (2022), 1773–1791.
- [CPK23] CHATZIMPARPAS A., PAULOVICH F. V., KERREN A.: HardVis: Visual analytics to handle instance hardness using undersampling and oversampling techniques. *Computer Graphics Forum* 42, 1 (2023), 135–154.
- [CvW22] COLLARIS D., VAN WIJK J. J.: StrategyAtlas: Strategy analysis for machine learning interpretability. *IEEE Transactions on Visualization and Computer Graphics* 29, 6 (2023), 2996–3008. <https://doi.org/10.1109/TVCG.2022.3146806>
- [CW97] COLIN CAMERON A., WINDMEIJER F. A.: An R-squared measure of goodness of fit for some common nonlinear regression models. *Journal of Econometrics* 77, 2 (1997), 329–342.
- [D311] D3 — Data-driven documents, 2011. Accessed January 24, 2024. URL: <https://d3js.org/>
- [DB21] DENG J., BROWN E. T.: RISSAD: Rule-based interactive semi-supervised anomaly detection. In *Proceedings of the*

- EuroVis 2021 – Short Papers* (2021), The Eurographics Association. <https://doi.org/10.2312/evs.20211050>
- [DCB19] DI CASTRO F., BERTINI E.: Surrogate decision tree visualization. In *Proceedings of the CEUR Workshop* (2019), vol. 2327, CEUR-WS.
- [Def23] DeforestVis Code, 2023. Accessed January 24, 2024. URL: <https://github.com/angeloschatzimparmpas/DeforestVis>
- [DG17] DUA D., GRAFF C.: UCI machine learning repository, 2017. Accessed January 24, 2024. URL: <http://archive.ics.uci.edu/ml>
- [Do07] DO T.-N.: Towards simple, easy to understand, an interactive decision tree algorithm. *College Information Technology Can Tho University, Can Tho, Vietnam, Technology Report* (2007), 06–01.
- [EAM14] EISEMANN M., ALBUQUERQUE G., MAGNOR M.: A nested hierarchy of localized scatterplots. In *Proceedings of the 27th SIBGRAPI Conference on Graphics, Patterns and Images* (2014), pp. 80–86. <https://doi.org/10.1109/SIBGRAPI.2014.14>
- [EAMS19] ELSHAWI R., AL-MALLAH M. H., SAKR S.: On the interpretability of machine learning-based model for predicting hypertension. *BMC Medical Informatics and Decision Making* 19, 1 (2019), 1–32.
- [EF10] ELMQVIST N., FEKETE J.-D.: Hierarchical aggregation for information visualization: Overview, techniques, and design guidelines. *IEEE Transactions on Visualization and Computer Graphics* 16, 3 (2010), 439–454.
- [EMJ\*22] EIRICH J., MÜNCH M., JÄCKLE D., SEDLMAIR M., BONART J., SCHRECK T.: RfX: A design study for the interactive exploration of a random forest to enhance testing procedures for electrical engines. *Computer Graphics Forum* 41, 6 (2022), 302–315.
- [Fla10] Flask — A micro web framework written in Python, 2010. Accessed January 24, 2024. URL: <https://flask.palletsprojects.com/>
- [Fri01] FRIEDMAN J. H.: Greedy function approximation: A gradient boosting machine. *Annals of Statistics* (2001), 1189–1232.
- [FSA99] FREUND Y., SCHAPIRE R., ABE N.: A short introduction to boosting. *Journal of Japanese Society for Artificial Intelligence* 14, 5 (Sept. 1999), 771–780.
- [FW98] FRANK E., WITTEN I. H.: Generating accurate rule sets without global optimization. In *Proceedings of the Fifteenth International Conference on Machine Learning* (1998), ICML '98, Morgan Kaufmann Publishers Inc., pp. 144–151.
- [GGPPS13] GUERRA-ÓMEZ J., PACK M. L., PLAISANT C., SHNEIDERMAN B.: Visualizing change over time using dynamic hierarchies: TreeVersity2 and the StemView. *IEEE TVCG* 19, 12 (2013), 2566–2575.
- [GPTM10] GENUER R., POGGI J.-M., TULEAU-MALOT C.: Variable selection using random forests. *Pattern Recognition Letters* 31, 14 (2010), 2225–2236.
- [HC00] HAN J., CERCONE N.: RuleViz: A model for visualizing knowledge discovery process. In *Proceedings of the sixth ACM SIGKDD international conference on Knowledge discovery and data mining* (2000), KDD '00, ACM, pp. 244–253. <https://doi.org/10.1145/347090.347139>
- [HHC\*19] HOHMAN F., HEAD A., CARUANA R., DELINE R., DRUCKER S. M.: Gamut: A design probe to understand how data scientists understand machine learning models. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems* (2019), ACM. <https://doi.org/10.1145/3290605.3300809>
- [HKPC19] HOHMAN F., KAHNG M., PIANTA R., CHAU D. H.: Visual analytics in deep learning: An interrogative survey for the next frontiers. *IEEE TVCG* 25, 8 (2019), 2674–2693.
- [HLLW19] HUANG Y., LIU Y., LI C., WANG C.: GBRTVis: Online analysis of gradient boosting regression tree. *Journal of Visualization* 22, 1 (Feb. 2019), 125–140.
- [HMJE\*19] HILASACA G. M., MARCÍLIO-JR W. E., ELER D. M., MARTINS R. M., PAULOVIK F. V.: Overlap removal of dimensionality reduction scatterplot layouts. *ArXiv e-prints 1903.06262* (2019). arXiv:1903.06262.
- [HT86] HASTIE T., TIBSHIRANI R.: Generalized additive models. *Statistical Science* 1, 3 (1986), 297–310.
- [IL92] IBA W., LANGLEY P.: Induction of one-level decision trees. In *Machine Learning Proceedings 1992*. Morgan Kaufmann, San Francisco (CA), 1992, pp. 233–240. <https://doi.org/10.1016/B978-1-55860-247-2.50035-8>
- [JLJ23] JEONG H., JEONG H.-O., LEE S., JEONG W.-K.: Dimensionality explorer for single-cell analysis. In *2023 IEEE 16th Pacific Visualization Symposium (PacificVis)* (2023), pp. 51–60. <https://doi.org/10.1109/PacificVis56936.2023.00013>
- [JLL\*20] JIA S., LIN P., LI Z., ZHANG J., LIU S.: Visualizing surrogate decision trees of convolutional neural networks. *Journal of Visualization* 23, 1 (2020), 141–156.
- [KGQ\*24] KALE A., GUO Z., QIAO X. L., HEER J., HULLMAN J.: EVM: Incorporating model checking into exploratory visual analysis. *IEEE Transactions on Visualization and Computer Graphics* 30, 1 (2024), 208–218. <https://doi.org/10.1109/TVCG.2023.3326516>
- [LJC16] LEE T., JOHNSON J., CHENG S.: An interactive machine learning framework, 2016. arXiv:1610.05463.
- [LL17] LUNDBERG S. M., LEE S.-I.: A unified approach to interpreting model predictions. In *Proceedings of the Advances in Neural Information Processing Systems* (2017), vol. 30, Curran Associates, Inc.

- [LXL\*18] LIU S., XIAO J., LIU J., WANG X., WU J., ZHU J.: Visual diagnosis of tree boosting methods. *IEEE TVCG* 24, 1 (Jan. 2018), 163–173.
- [MGT\*03] MUNZNER T., GUIMBRETIERE F., TASIRAN S., ZHANG L., ZHOU Y.: TreeJuxtaposer: Scalable tree comparison using focus+context with guaranteed visibility. *ACM Transactions on Graphics* 22, 3 (July 2003), 453–462.
- [MHM18] MCINNES L., HEALY J., MELVILLE J.: UMAP: Uniform manifold approximation and projection for dimension reduction. *ArXiv e-prints 1802.03426* (Feb. 2018). arXiv:1802.03426.
- [MJEP\*21] MARCÍLIO-JR W. E., ELER D. M., PAULOVICH F. V., RODRIGUES-JR J. F., ARTERO A. O.: ExplorerTree: A focus+context exploration approach for 2D embeddings. *Big Data Research* 25 (2021), 100239.
- [MLMP18] MÜHLBACHER T., LINHARDT L., MÖLLER T., PIRINGER H.: TreePOD: Sensitivity-aware selection of pareto-optimal decision trees. *IEEE TVCG* 24, 1 (2018), 174–183.
- [Mol20] MOLNAR C.: *Interpretable Machine Learning*. Independently Published, 2020.
- [MQB19] MING Y., QU H., BERTINI E.: RuleMatrix: Visualizing and understanding classifiers with rules. *IEEE TVCG* 25, 1 (2019), 342–352.
- [Mun09] MUNZNER T.: A nested model for visualization design and validation. *IEEE TVCG* 15, 6 (2009), 921–928.
- [Mun14] MUNZNER T.: *Visualization Analysis and Design*. CRC Press, 2014.
- [MXLM20] MA Y., XIE T., LI J., MACIEJEWSKI R.: Explaining vulnerabilities to adversarial machine learning through visual analytics. *IEEE TVCG* 26, 1 (Jan. 2020), 1075–1085.
- [NHS00] NGUYEN T., HO T., SHIMODAIRA H.: A visualization tool for interactive learning of large decision trees. In *Proceedings of the 12th IEEE International Conference on Tools with Artificial Intelligence* (2000), ICTAI '00, IEEE, pp. 28–35. <https://doi.org/10.1109/TAI.2000.889842>
- [NJKC19] NORI H., JENKINS S., KOCH P., CARUANA R.: InterpretML: A unified framework for machine learning interpretability. *ArXiv e-prints 1909.09223* (Sep. 2019).
- [NP21] NETO M. P., PAULOVICH F. V.: Explainable Matrix - Visualization for global and local interpretability of random forest classification ensembles. *IEEE TVCG* 27, 2 (2021), 1427–1437.
- [NP22] POPOLIN NETO M., PAULOVICH F. V.: Multivariate data explanation by Jumping Emerging Patterns visualization. *IEEE Transactions on Visualization and Computer Graphics* 30, 2 (2024), 1549–1563. <https://doi.org/10.1109/TVCG.2022.3223529>
- [NWWH19] NSCH R. H., WIESNER P., WENDLER S., HELLWICH O.: Colorful Trees: Visualizing random forests for analysis and interpretation. In *Proceedings of the IEEE Winter Conference on Applications of Computer Vision* (2019), WACV '19, IEEE, pp. 294–302. <https://doi.org/10.1109/WACV.2019.00037>
- [plo10] Plotly — JavaScript open source graphing library, 2010. Accessed January 24, 2024. URL: <https://plotly.com>
- [PNWG17] PHILLIPS N. D., NETH H., WOIKE J. K., GAISSMAIER W.: FFTrees: A toolbox to create, visualize, and evaluate fast-and-frugal decision trees. *Judgment and Decision making* 12, 4 (2017), 344–368.
- [PSMD14] PADUA L., SCHULZE H., MATKOVIĆ K., DELRIEU C.: Interactive exploration of parameter space in data mining: Comprehending the predictive quality of large decision tree collections. *Computers & Graphics* 41 (2014), 99–113.
- [PVG\*11] PEDREGOSA F., VAROQUAUX G., GRAMFORT A., MICHEL V., THIRION B., GRISEL O., BLONDEL M., PRETTENHOFER P., WEISS R., DUBOURG V., VANDERPLAS J., PASSOS A., COURNAPÉAU D., BRUCHER M., PERROT M., DUCHESNAY E.: Scikit-Learn: Machine learning in Python. *Journal of Machine Learning Research* 12 (Nov. 2011), 2825–2830.
- [RSG16] RIBEIRO M. T., SINGH S., GUESTRIN C.: “Why should I trust you?”: Explaining the predictions of any classifier. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining* (2016), KDD '16, ACM, pp. 1135–1144. <https://doi.org/10.1145/2939672.2939778>
- [RSG18] RIBEIRO M. T., SINGH S., GUESTRIN C.: Anchors: High-precision model-agnostic explanations. In *Proceedings of the AAAI Conference on Artificial Intelligence* (Apr. 2018), vol. 32. <https://doi.org/10.1609/aaai.v32i1.11491>
- [Sch99] SCHAPIRE R. E.: A brief introduction to boosting. In *Proceedings of the 16th International Joint Conference on Artificial Intelligence - Volume 2* (1999), IJCAI'99, Morgan Kaufmann Publishers Inc., pp. 1401–1406.
- [SCS04] SONG H., CURRAN E. P., STERRITT R.: Multiple foci visualisation of large hierarchies with FlexTree. *Information Visualization* 3, 1 (2004), 19–35.
- [SED\*88] SMITH J., EVERHART J., DICKSON W., KNOWLER W., JOHANNES R.: Using the ADAP learning algorithm to forecast the onset of diabetes mellitus. In *Proceedings of the Annual Symposium Computer Application in Medical Care* (1988), American Medical Informatics Association, pp. 261–265.
- [SFK08] SOBESTER A., FORRESTER A., KEANE A.: *Engineering Design via Surrogate Modelling: A Practical Guide*. John Wiley & Sons, 2008.
- [SL91] SAFAVIAN S., LANDGREBE D.: A survey of decision tree classifier methodology. *IEEE Transactions on Systems, Man, and Cybernetics* 21, 3 (1991), 660–674.
- [ST01] SATO M., TSUKIMOTO H.: Rule extraction from neural networks via decision tree induction. In *Proceedings of the International Joint Conference on Neural Networks* (2001), vol. 3

- of *IJCNN '01*, pp. 1870–1875 vol. 3. <https://doi.org/10.1109/IJCNN.2001.938448>
- [TDB21] THOMAS L. V., DENG J., BROWN E. T.: FacetRules: Discovering and describing related groups. In *Proceedings of the IEEE Workshop on Machine Learning from User Interactions* (2021), MLUI '21, pp. 21–26. <https://doi.org/10.1109/MLUI54255.2021.00008>
- [TKC17] TAM G. K. L., KOTHARI V., CHEN M.: An analysis of machine- and human-analytics in classification. *IEEE TVCG* 23, 1 (2017), 71–80.
- [TM03a] TEOH S. T., MA K.-L.: PaintingClass: Interactive construction, visualization and exploration of decision trees. In *Proceedings of the ninth ACM SIGKDD international conference on Knowledge discovery and data mining* (2003), KDD '03, ACM, pp. 667–672. <https://doi.org/10.1145/956750.956837>
- [TM03b] TEOH S. T., MA K.-L.: StarClass: Interactive visual classification using star coordinates. In *Proceedings of the 2003 SIAM International Conference on Data Mining* (2003), SIAM, pp. 178–185. <https://doi.org/10.1137/1.9781611972733.16>
- [VCP22] VARU R., CHRISTINO L., PAULOVICH F. V.: ARMATRIX: An interactive item-to-rule matrix for association rules visual analytics. *Electronics* 11, 9 (2022). <https://doi.org/10.3390/electronics11091344>
- [vdEvW11] VAN DEN ELZEN S., VAN WIJK J. J.: BaobabView: Interactive construction and analysis of decision trees. In *Proceedings of the IEEE Conference on Visual Analytics Science and Technology* (2011), VAST '11, IEEE, pp. 151–160. <https://doi.org/10.1109/VAST.2011.6102453>
- [vdMH08] VAN DER MAATEN L., HINTON G.: Visualizing data using t-SNE. *Journal of Machine Learning Research* 9 (2008), 2579–2605.
- [VOMS21] VELMURUGAN M., OUYANG C., MOREIRA C., SINDHGATTA R.: Evaluating fidelity of explainable methods for predictive process analytics. In *Proceedings of the International Conference on Advanced Information Systems Engineering* (2021), Springer, pp. 64–72.
- [vue14] Vue.js — The progressive JavaScript framework, 2014. Accessed January 24, 2024. URL: <https://vuejs.org/>
- [WFH\*01] WARE M., FRANK E., HOLMES G., HALL M., WITTEN I. H.: Interactive machine learning: Letting users build classifiers. *International Journal of Human-Computer Studies* 55, 3 (2001), 281–292.
- [WKN\*22] WANG Z. J., KALE A., NORI H., STELLA P., NUNNALLY M. E., CHAU D. H., VORVOREANU M., WORTMAN VAUGHAN J., CARUANA R.: Interpretability, then what? Editing machine learning models to reflect human knowledge and values. In *Proceedings of the 28th ACM SIGKDD Conference on Knowledge Discovery and Data Mining* (New York, NY, USA, 2022), KDD '22, Association for Computing Machinery, pp. 4132–4142. <https://doi.org/10.1145/3534678.3539074>
- [Wol92] WOLPERT D. H.: Stacked generalization. *Neural Networks* 5, 2 (1992), 241–259.
- [WZwy21] WANG J., ZHANG W., WANG L., YANG H.: Investigating the evolution of tree boosting models with visual analytics. In *Proceedings of the 14th IEEE Pacific Visualization Symposium* (2021), PacificVis '21, IEEE, pp. 186–195. <https://doi.org/10.1109/PacificVis52677.2021.00032>
- [XCC\*21] XIA Y., CHENG K., CHENG Z., RAO Y., PU J.: GBMVis: Visual analytics for interpreting gradient boosting machine. In *Proceedings of the Cooperative Design, Visualization, and Engineering: 18th International Conference* (2021), CDVE '21, Springer, pp. 63–72. [https://doi.org/10.1007/978-3-030-88207-5\\_7](https://doi.org/10.1007/978-3-030-88207-5_7)
- [XXM\*19] XU K., XIA M., MU X., WANG Y., CAO N.: EnsembleLens: Ensemble-based visual exploration of anomaly detection algorithms with multidimensional data. *IEEE TVCG* 25, 1 (Jan. 2019), 109–119.
- [YBOB22] YUAN J., BARR B., OVERTON K., BERTINI E.: Visual exploration of machine learning model behavior with hierarchical surrogate rule sets. *IEEE Transactions on Visualization and Computer Graphics* 30, 2 (2024), 1470–1488. <https://doi.org/10.1109/TVCG.2022.3219232>
- [YCB\*22] YUAN J., CHAN G. Y.-Y., BARR B., OVERTON K., REES K., NONATO L. G., BERTINI E., SILVA C. T.: SUBPLEX: A visual analytics approach to understand local model explanations at the subpopulation level. *IEEE Computer Graphics and Applications* 42, 6 (2022), 24–36.
- [YNB21] YUAN J., NOV O., BERTINI E.: An exploration and validation of visual factors in understanding classification rule sets. In *Proceedings of the IEEE Visualization Conference* (2021), VIS '21, pp. 6–10. <https://doi.org/10.1109/VIS49827.2021.9623303>
- [YXX\*21] YUAN J., XIANG S., XIA J., YU L., LIU S.: Evaluation of sampling methods for scatterplots. *IEEE Transactions on Visualization and Computer Graphics* 27, 2 (2021), 1720–1730.
- [ZWLC19] ZHAO X., WU Y., LEE D. L., CUI W.: iForest: Interpreting random forests via visual analytics. *IEEE TVCG* 25, 1 (Jan. 2019), 407–416.
- [ZYMW19] ZHANG Q., YANG Y., MA H., WU Y.: Interpreting CNNs via decision trees. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition* (June 2019), CVPR '19, IEEE Computer Society, pp. 6254–6263. <https://doi.org/10.1109/CVPR.2019.00642>

### Supporting Information

Additional supporting information may be found online in the Supporting Information section at the end of the article.

Supporting Information