iForest: Interpreting Random Forests via Visual Analytics

Xun Zhao, Yanhong Wu, Dik Lun Lee, and Weiwei Cui

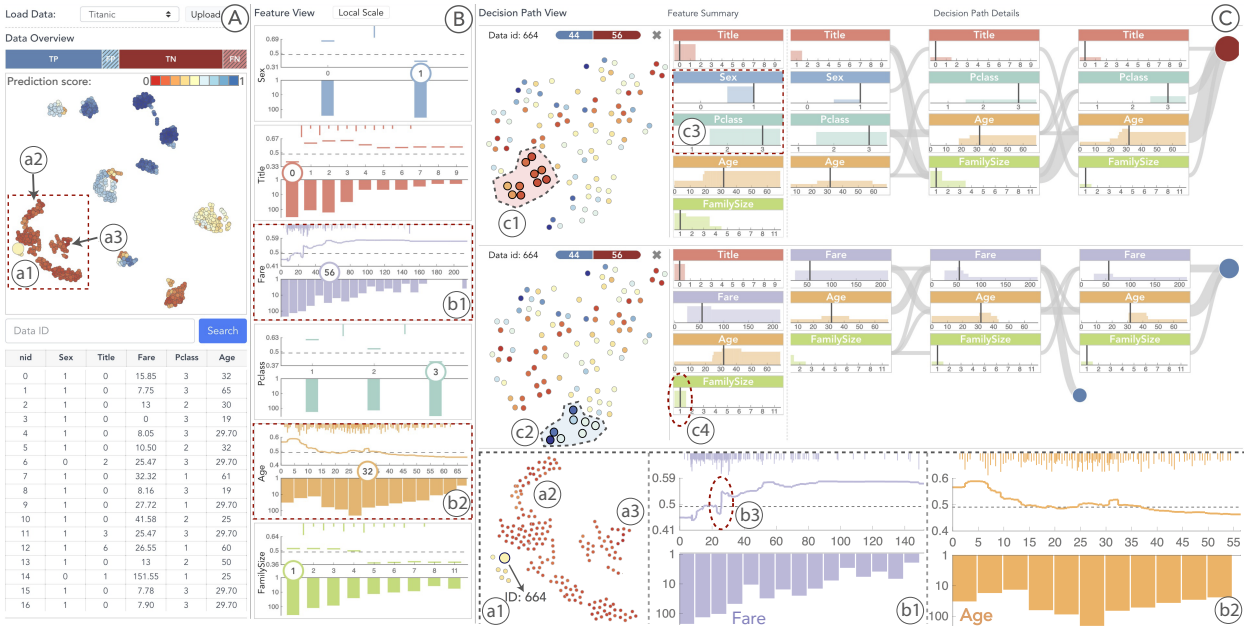

Figure 1. Using iForest to interpret random forests with Titanic dataset: (A) a Data Overview displaying an overview of how random forests classify data; (B) a Feature View depicting the relationships between features and predictions from various perspectives; (C) a Decision Path View revealing the underlying working mechanisms by enabling users to audit and compare different decision paths. iForest allows users to interpret random forests from various perspectives. For example, users can compare the negative decision paths (c1) against the positive ones (c2) to examine the most significant reasons for generating different results.

Abstract— As an ensemble model that consists of many independent decision trees, random forests generate predictions by feeding the input to internal trees and summarizing their outputs. The ensemble nature of the model helps random forests outperform any individual decision tree. However, it also leads to a poor model interpretability, which significantly hinders the model from being used in fields that require transparent and explainable predictions, such as medical diagnosis and financial fraud detection. The interpretation challenges stem from the variety and complexity of the contained decision trees. Each decision tree has its unique structure and properties, such as the features used in the tree and the feature threshold in each tree node. Thus, a data input may lead to a variety of decision paths. To understand how a final prediction is achieved, it is desired to understand and compare all decision paths in the context of all tree structures, which is a huge challenge for any users. In this paper, we propose a visual analytic system aiming at interpreting random forest models and predictions. In addition to providing users with all the tree information, we summarize the decision paths in random forests, which eventually reflects the working mechanism of the model and reduces users' mental burden of interpretation. To demonstrate the effectiveness of our system, two usage scenarios and a qualitative user study are conducted.

Index Terms—Interpretable Machine Learning, Random Forests, Random Forest Visualization, Visual Analytics.

1 Introduction

Random forests, an ensemble machine learning model that consists of many independent decision trees, are widely adopted for classification and regression tasks. Each tree in a random forest model is trained independently on a random subset of the training data and a random subset of features. Then, given a testing data input, random forests generate the final prediction by feeding the input to all decision trees and summarizing their results. Random forests have been extensively studied in the machine learning and data mining areas for many years [10] and proved to perform well in many domains, such as biomedical engineering [17] and traffic planning [25]. One exhaustive study [20] evaluates the performances of 179 classifiers stemming from different families on the UCI classification database, and shows that random forest classifiers overall outperform other classifier families, such as neural networks and support vector machines (SVM).

Despite their impressive prediction performance, one critical issue for random forests is interpretability. Although this issue is shared by most machine learning models, it is especially severe for random forests. According to Brieman et al. [11], "random forests are A+ predictors on performance" but "rate an F on interpretability". Thus, random forests are usually considered as black boxes [44], and the "F-grade" interpretability has prevented the model from being adopted in

- *X. Zhao and D. Lee are with the Hong Kong University of Science and Technology. E-mail: {xzhaoag, dlee}@ust.hk*
- *Y. Wu is with Visa Research. E-mail: yanwu@visa.com*
- *W. Cui is with Microsoft Research Asia and is the corresponding author. E-mail: weiwei.cui@microsoft.com*

Manuscript received xx xxx. 201x; accepted xx xxx. 201x. Date of Publication xx xxx. 201x; date of current version xx xxx. 201x. For information on obtaining reprints of this article, please send e-mail to: reprints@ieee.org. Digital Object Identifier: xx.xxxx/TVCG.201x.xxxxxxx

some domains that have little or zero tolerance of errors, such as financial lending, criminal justice, and medical diagnosis. As any machine learning models may provide erroneous predictions [15], data scientists in these domains need to understand how a particular prediction is reached and examine whether the model works properly. For instance, in medical diagnosis settings, data scientists usually develop machine learning models to assist doctors in making treatment decisions. However, a doctor can barely trust the model without understanding how it generates the prediction, as blind trust can be catastrophic for patients [15, 47]. Similarly, in the financial domain, failed loan applicants usually want to know the exact reasons why their loan applications have been rejected.

One popular approach to interpret classification models is to observe the relationships between features and predictions using methods like partial dependence plots (PDPs) [21, 22]. Although these methods can illustrate how feature values affect predictions, they fall short when it comes to random forests. In a random forest model, each tree represents a self-consistent prediction strategy, which can easily be interpreted using PDPs. However, a random forest model often contains tens or hundreds of independent decision trees. Since the same feature is likely being treated differently in different trees, forest-level analysis helps users better understand which range of feature values is strongly related to which predictions. However, it is difficult to extend the existing methods to provide an overview of a large number of trees.

Apart from revealing feature-prediction relationships, uncovering the underlying working mechanism is another critical perspective for interpreting machine learning models [34], including random forests. This helps users examine whether the trained model works properly, and hence gain confidence in the generated predictions. To fully understand the working mechanism of random forest models, users should be able to audit a prediction and track the decision process, so that they can determine if the prediction is reliable based on their domain knowledge [58]. For a single decision tree, users can easily audit its working mechanism by tracing the root-to-leaf decision paths. However, users need to compare hundreds of different decision paths for a particular prediction in random forests, which is a time-consuming and labor-intensive process.

Another important strategy for interpreting machine learning models is case-based reasoning. Based on the idea of solving problems by analogy [34], case-based reasoning facilitates users to determine their confidence in the prediction of a given data input by providing similar data examples as references [15, 28]. Popular case-based reasoning methods used in general machine learning models are usually based on model proximity [14] or feature similarity [57]. Unfortunately, neither of them incorporates the fact that random forests generate predictions through partitioning the training data. Since each partition corresponds to a path in the forest, similarities between decision paths may provide valuable insights to help users understand why a prediction is made. However, existing reasoning methods fail to provide data examples.

The above challenges hinder data scientists from various domains in using random forests to solve their problems due to the lack of interpretability. To tackle these challenges, we develop iForest, an interactive visualization system, to help users interpret random forests from different perspectives. First, we build a Feature View to illustrate the relationships between input features and outcome predictions. To uncover the underlying working mechanism of random forests, we propose a novel design that summarizes several decision paths based on feature appearances and ranges, which allows users to explore and understand the partition logics of these paths. We also support the case-based reasoning for random forest models and predictions from the perspective of both feature similarity and decision path similarity. Specifically, our contributions are summarized as follows:

- An interactive visualization system, iForest, that assists users in interpreting random forest models and predictions.
- A novel pixel-based bar chart design that summarizes the features and corresponding ranges in multiple decision paths to uncover the underlying working mechanism of random forests.
- Two usage scenarios and a qualitative user study that demonstrate the usefulness and effectiveness of iForest.

2 Background

Algorithm 1 Random Forest Construction

Input: $X = \{x_1, x_2, ..., x_N\}$, $F = \{f^1, f^2, ..., f^M\}$, $Y = \{y_1, y_2, ..., y_N\}$
Output: $RF = \{T_1, ..., T_K\}$
for `k= 1 to K` **do**
Draw a bootstrap sample D' of size N by sampling N times with replacement from the training data X and initializing a decision tree T_k with one root node.
repeat the following steps recursively for each terminal node v of tree T_k.
1. Select m features randomly from F, where $m < M$.
2. Pick the best splitting feature f_v from the m features.
3. Find the best splitting threshold θ_v for f_v and split v into two child nodes.

until the number of nodes in tree T_k reaches a certain threshold.
end for

Formally, given the input data $X = \{x_1, ..., x_N\}$ where N is the size of training data, we consider $x_i = \{x_i^1, ..., x_i^M\}$ as the input vector of the ith data item with M features $F = \{f^1, ..., f^M\}$. In this work, we focus on binary classifiers and consider the data label $Y = \{y_1, ..., y_N\}$ as binary categorical variables where $y_i \in \{0, 1\}$ is the label of the ith data item. A random forest model $RF = \{T_1, ..., T_K\}$ can be built using Algorithm 1 in which T_k represents a decision tree and K is the total number of trees. To assist the discussions in the rest sections, we introduce the following terms.

Split Point: For each node v in a decision tree, the split point s_v refers to a feature f_v and a threshold θ_v that are used to split node v into two child nodes. We pick the split point s_v that partitions the N_v training samples into left child v_L and right child v_R to maximize the decrease of node impurity. In our paper, we measure the node impurity using Gini impurity [24].

Decision Path: Given an input data item, each decision tree will identify a root-to-leaf path (decision path) accordingly, which leads to the prediction made by the tree. Assuming a decision path p contains H non-leaf nodes $V(p) = \{v_1, .., v_h, .., v_H\}$,
it can be represented as:

$$p = \{(x^{f_{v_1}} \bigotimes \theta_{v_1}), (x^{f_{v_2}} \bigotimes \theta_{v_2}), ..., (x^{f_{v_H}} \bigotimes \theta_{v_H})\} \quad (1)$$

where f_{v_h} is the feature, θ_{v_h} is the corresponding threshold, and $\bigotimes \in \{\text{“}\leq\text{”}, \text{“}>\text{”}\}$ represents the boolean condition on each node v_h.

Prediction Score: For one data instance x_i, we define the prediction score of the k th decision tree T_k as $T_k(x_i) \in [0, 1]$. A prediction score that is close to 0 or 1 indicates a high prediction confidence while a prediction score close to the threshold represents a low prediction confidence. The predicted label $\sigma(T_k(x_i), \theta) \in \{0, 1\}$ can be then obtained by setting certain predicted score threshold $\theta \in [0, 1]$. If the score is larger than θ (by default 0.5), the predicted label is positive, otherwise negative. A random forest generates the final prediction by either taking the majority vote of $\{\sigma(T_k(x_i), \theta) | 1 \leq k \leq K\}$ or averaging the prediction scores of all its internal trees, *i.e.*, $\sigma(\sum_{k=1}^{K} T_k(x_i)/K, \theta)$.

3 Related Work

3.1 Random Forest Interpretation

Random Forest interpretation methods can mainly be divided into 3 groups: feature analysis, model reduction, and case-based reasoning.

Feature Analysis. One simple but effective approach in this category is to calculate feature importance [10, 13, 24], which assigns each feature a score to indicate its impact on resulting predictions. The feature importance is often evaluated based on either the Mean Decrease Accuracy (MDA) [10] or the Mean Decrease Impurity (MDI) [13]. MDA is a model-agnostic method that randomly permutes the values of a feature, and then measures how much the prediction accuracy decreases. Unlike MDA that can be used for any model, MDI [37] is

specialized for tree-based models. It calculates the average decreased impurity (e.g., Gini impurity [24]), which determines the feature and its split point for each node in a decision tree [37]. Both MDA and MDI can be either applied globally on an entire dataset to reveal the overall feature importance [10] or calculated on a single prediction for a more detailed examination [43, 44]. Although they have been proven effective by various studies [37], MDI is more popular in random forest analysis [24]. Therefore, we choose MDI as our feature importance measure in this paper. Apart from feature importance, another powerful tool for illustrating the relationships between features and predictions is partial dependence plots (PDPs) [21, 22], which depict how feature value changes affect predictions. Typically, a PDP is visualized as a line chart, where the x-axis represents feature values and the y-axis shows prediction probabilities. Prospector et al. [30] recently introduces another heatmap-based design to represent PDPs, in which the color encodes prediction probabilities. This design is more space-efficient and allows the comparison of multiple features at the same time. However, one major drawback of the above methods is that they all ignore the split points in tree nodes. These thresholds depict random forests' partition criteria for each feature and can provide insights into which ranges of features are regarded critical by the trained model for predictions. We encode the feature threshold information which enables users to perceive these important feature thresholds.

Model Reduction. It is a post-hoc interpretation method that learns a surrogate model to approximate the original complex ensemble model. Surrogate models are often simple and interpretable, such as decision trees [48, 60], decision rules [16, 23], and decision sets [32]. For example, Schetinin et al. [48] propose a method that evaluates the uncertainty of all trees in a random forest model and then select the one that has the highest confidence and accuracy as the surrogate model. However, in this approach, the structure of the surrogate tree depends on the model parameters, which means it can still be too complex for humans to interpret. Apart from surrogate trees, some other work also learns other models, such as decision rules [16], which may have simpler structures. Though model-reduction methods provide a simplified overview of the working mechanism of random forests, when users need to examine a specific prediction, these methods are usually not accurate enough to reflect the model's actual behavior as they summarize the structures and properties of all the decision trees. When considering a specific prediction, only the decision paths that are decided by the input data item are related to the working logic. Compared to model reduction-based methods that approximate all the trees' structure and property information, showing decision paths for a particular prediction can accurately and flexibly reflect the working logic of random forests. Thus, we summarize the decision paths to uncover the underlying working mechanisms of random forests instead of model reduction-based methods.

Case-based Reasoning. This methodology relies on the intuition that a new problem can be solved by summarizing the solutions of similar problems. This aligns with the analogical decision making process of humans [29, 40], which has been widely adopted in various real-world applications, such as the medical [8] and the financial [33] domains. In the random forests scenario, case-based reasoning is often used to justify the reliability of predictions [14, 34]. Given a data item and its prediction, training data that are most similar to the input, along with their predictions, are collected first. Then, by comparing their feature and prediction differences, users can better understand and judge whether this prediction is reliable based on their knowledge. Common measures, such as Euclidean distance or Cosine distance in the feature space, are often adopted to compare different data items. In addition, specialized measures, such as the prediction score [14] or the number of common leaf nodes reached [51], can also be used as distance functions for calculating data similarities in random forests. These measures examine data similarity from the model's perspective, which helps inspect whether the trained model ignores some features or assign high weights on other features. Since the two types of measures evaluate the similarity from different perspectives, we utilize both of them in our system for users to better understand and judge a prediction.

3.2 Tree-based Model Visualization

In this section, we review existing work on decision tree and tree ensemble visualizations.

Decision trees can be visualized using different visualization techniques, such as node-link diagram [41, 56, 59], icicle plot [6, 36, 54], and Treemap [38, 49]. As a natural representation of tree structures, node-link diagrams are widely adopted in visualizing decision trees. In addition to tree structures, BaobabView [53] further encodes more information of models, such as the split point in each node and the training data volume that passes through each branch. Although node-link diagrams can clearly depict tree structures, they may not scale well when trees become deep and complex. A more space-efficient way to visualize decision trees is icicle plots [31], in which the nodes are encoded by multiple stacked bars. The length of each bar is proportional to the volume of data which passes through the corresponding node. To further encode the label distribution of each node, Ankerst et al. [7] adopt a pixel-based design that uses pixels to represent data items and pixel colors to represent the data labels. This pixel-based design enables users to evaluate whether each node can successfully partition data with different labels. Muhlbacher et al. [38] also use the pixel-based design on Treemap to help users estimate the complexity and performance of a decision tree. Although existing methods are effective in visualizing decision trees, they are not designed for random forests. In random forests, it is difficult or even impossible for humans to understand and compare the structures and properties of all decision trees. For each decision tree, only one decision path is related to a specific data item and showing the entire tree structures may confuse users as it includes too much unrelated information. In this paper, we choose to visualize the decision paths that are used in a specific prediction instead of the whole tree structures to illustrate the working mechanism of random forests and reduce users' mental burden.

Apart from visualizing individual decision trees, many methods also attempt to visualize different perspectives of tree-ensemble models. One perspective is the relationships between data and decision trees. Breiman and Wald [12] use random forest proximity as the similarity measure and adopt Multi-dimensional Scaling (MDS) to visualize training data, so that users can intuitively observe data clusters and outliers identified by the random forest model. Ploński et al. [45] use Self-Organizing Map (SOM) instead of MDS to achieve higher accuracy. Another perspective is the relationships between features and decision trees. Urbanek et al. [52] adopt a matrix-based visualization that illustrates the feature importances on each tree, which enables users to compare feature similarities in a fine granularity. One drawback of these methods is that they still consider random forests as a black box and ignore the structures and properties of decision paths that reflect the model's working mechanism. Analyzing the similarities between decision paths can answer many questions that are critical for prediction interpretation. For example, among all the decision paths for a data item, how many of them generate positive predictions and what are the prediction scores of these decision paths? To overcome this drawback, in addition to exploring the data similarity and feature correlations, we visualize different properties of decision paths to reveal how random forests generate predictions. Liu et al. [35] propose an interactive visual diagnosis tool to analyze the performance of boosting trees, which are a tree ensemble model similar to random forests. They also design a temporal confusion matrix that visualizes the class confusions of all trees to help users analyze and diagnose boosting tree models. However, they do not help users understand how the model makes predictions and inspect a specific prediction. In our work, we focus more on understanding how random forests make predictions and observing prediction quality rather than model diagnosis such as comparing the effects of different model parameter settings.

4 Design Goals

Based on a thorough literature review of 35 papers collected from the machine learning, visualization and human-computer interaction fields, we distilled the following design goals to guide the system development. Further details are provided in supplementary materials.

G1: Reveal the relationships between features and predictions. To make random forests transparent, users first need to understand what the model has learned in general and be able to evaluate the model's predictions [34]. In training stages, random forests basically learn the mappings between input features and outcome predictions. Thus, these mappings reflect model behaviors and can facilitate users in understanding the characteristics of random forest models. For example, users may want to know which features are considered important by models and measure their influences (positive or negative) on predictions. Users can then understand whether a small change in feature values can alter the prediction [30]. From the features' influences on predictions, users can further infer the feature correlation and remove unnecessary features. Thus, revealing the relationships between features and predictions is beneficial in interpreting random forests.

G2: Uncover the underlying working mechanisms. Opening the black box of random forests not only requires revealing the relationships between features and predictions, but also needs to uncover the underlying working mechanisms [34]. Users should be able to audit the decision process of a prediction and make sure they agree before making a decision [9]. For example, doctors may need to understand the model's prediction on a specific patient to evaluate whether this prediction is appropriate so that they can take further actions. This helps users examine whether the model works properly and understand why a specific prediction has been reached [58]. For random forests, the overall working logic can be described by the structures and attributes of individual decision trees. For example, the split point of each tree node depicts the feature threshold of different predictions. Analyzing the root-to-leaf decision paths can help answer many questions in random forest interpretation. For instance, what are the similarities among the decision paths that generate the same prediction? What are the major differences for two decision paths to generate different results? For each decision path, what is the prediction and how certain is the prediction? By summarizing the structure and attributes of different decision trees, we aim to uncover the underlying working mechanisms of random forests.

G3: Provide case-based reasoning. Case-based reasoning is a crucial part of the most effective strategies in decision making scenarios. [28]. It relies on the idea that a new problem can be solved based on the summarized solution of similar problems [29]. These similar problems can serve as a scaffold for understanding and solving a new problem. This helps provide a holistic picture of random forests such as observing the model performance, examining which types of data the model tends to predict incorrectly, or identifying feature importance and impact. Similarly, when interpreting random forests, users can evaluate the prediction of a new case by comparing it with similar examples from the training data [14]. However, providing case-based reasoning for random forests is more complex than other models, since the random forests can have various similarity metrics. For example, the similarity metric for random forests can be calculated based not only on feature value distance, but also on how many common leaf nodes are reached [13, 51]. Thus, the similar cases of a testing case can vary according to different similarity metrics. Providing different sets of similar cases is beneficial as it helps users evaluate the prediction from various perspectives.

5 Analytical Tasks

To fulfill the design goals, we have distilled the following design tasks.

T1: Encode feature importance and partial dependence information. According to the definition, feature importance reflects the depth of a feature in decision trees and the amount of data partitioned by the feature threshold. Thus, it is a popular measure to help users understand which features are decisive to predictions [24]. However, such information is not sufficient to reveal how feature values and predications are correlated (**G1**), because feature importances cannot demonstrate how feature values affect final predictions. To address this issue, the partial dependence information should also be examined, since it demonstrates how predictions respond to feature changes. Through the partial dependence information, users can judge whether changing a feature value slightly would strongly affect the prediction so that they can evaluate the robustness of the model or a specific prediction. Therefore, the system should encode both the feature importance and the partial dependence information of features.

T2: Encode the split point distribution of each feature. As discussed above, the split point distribution is another piece of critical information to reveal the relationships between input features and output predictions (**G1**). A feature range with dense split points suggests the prediction is sensitive to values in this range (i.e., this range has a high impact on predictions). On the other hand, if a feature range only has a few and sparse split points, this may indicate that the feature has little influence on predictions. Therefore, visualizing the density information helps reveal different ranges' effects on predictions.

T3: Encode the prediction results and summarizing the similarities of decision paths. To understand how a prediction is generated from random forests (**G2**), users often need to dig into all decision paths that jointly produce the final prediction. Showing the diversity of interim predictions helps users evaluate the uncertainty of the final prediction. In addition, the similarities of decision paths can provide a good overview of their relationships. For example, users may want to know which groups of decision paths produce positive results and which decision paths have similar structures in general. Hence, to deepen the understanding of the working mechanism of random forests, it is important to expose how interim predictions are generated and the final consensus is reached when given a data input.

T4: Review structures of decision paths. Each decision path has a unique structure, including the path length, the features appeared on the path, the order of the appeared features, and the split threshold in each node. Although two decision paths may yield the same prediction, their structures are likely to be very different. These structures can provide deeper insights into the underlying working mechanisms of random forests (**G2**) [26, 34]. For example, users may be interested in whether there are any common feature thresholds among the decision paths that generate the same prediction. Users may also want to understand what the major differences between the paths that lead to opposite predictions. Thus, our system should summarize the structures of decision paths and enable users to examine their differences.

T5: Identify training data clusters and outliers. To support case-based reasoning (**G3**), users should be able to identify clusters and outliers from the training data. This can provide concrete examples for users to analyze the behaviors of models. For examples, users can observe whether certain data clusters are likely to have the same prediction result. Some data instances with high prediction confidences may also be incorrectly predicted by models. In addition, data outliers may indicate data noises, which can be removed from the training dataset to improve model performances. We aim to provide users with an overview of the training data and enable them to compare ground truth labels and prediction results.

T6: Encode training data value distribution. This distribution is useful for both feature analysis (**G1**) and case-based reasoning (**G3**). On one hand, it can reveal the relationships between features and predictions in a finer granularity (**G1**). For example, comparing the predictions of data in different feature ranges can help users analyze which feature ranges have stronger impact to predictions. On the other hand, the training data value distribution can also serve as evidence when observing partial dependence information and split point distribution. For example, if a numerical feature range has a low split point density, users need analyze whether this is because the data are sparse or this range of the feature is not considered important by the model based on the feature value distribution. Thus, encoding the distribution is useful for both feature analysis (**G1**) and case-based reasoning (**G3**).

T7: Support interactive model inspection. All three design goals listed above require our system to provide interactive model inspection. During the inspection process, users may want to examine why a group of data is incorrectly predicted. For example, they may examine the feature value distributions of the data group to check which features are responsible for the errors (**G1**). Sometimes users are more interested in exploring a single prediction interactively, rather than understanding the random forest model as a whole. When inspecting a single prediction, users may want to compare several deci-

sion paths for a deeper understanding of the model's working mechanism (**G2**), or examine the predictions of similar examples in the training data (**G3**). Therefore, the system should support users to interactively inspect the model and individual predictions.

6 System Overview

Motivated by the above design goals and tasks, we designed iForest, an interactive visual analytics system, to allow users to interpret random forest models. Our system consists of three major components: data processing, model building, and visual analysis. In the first module, we clean the raw datasets, conduct feature engineering, and store the processed data. The model building module trains a random forest model based on the processed data and calculates the similarity matrix for training data. These two modules are developed using Python. Specifically, we use scikit-learn [4] to build random forests and leverage a Python web framework, Flask [2], to build the back-end. The visual analysis module, which is the front-end of our system, consists of three major views to support different design tasks with rich interactions. We utilize D3 [1] in this visual analysis module.

7 Visual Design

As shown in Fig. 1, iForest contains three major views: 1) the *Data Overview* which enables users to identify clusters and outliers from the training data and summarizes the model's performance on these data; 2) the *Feature View* which depicts the relationships between features and predictions from various perspectives; 3) and the *Decision Path View* which aims to help users inspect a single prediction by summarizing and comparing all the corresponding decision paths. We also provide a *Control Panel* to let users customize their own testing data and feed them into iForest for understanding and evaluating the predictions. A rich set of interactions is also provided to link these views together to allow a dynamic exploration.

7.1 Data Overview

The Data Overview allows users to explore training data and their prediction results, so that users can have concrete examples to understand what the model has learned and how the model performs on the dataset. Common multi-dimensional data visualization techniques include parallel coordinate plots (PCP), scatterplot matrices (SPM), and dimension reduction techniques. PCP and SPM can accurately display multi-dimensional data and reveal dimension correlation while they are not space efficient when the number of dimensions is large. In contrast, dimension reduction methods have good scalability but one drawback is that the distances between points in the projected space may not accurately represent the distances in the original space. As we design the Data Overview mainly for users to obtain an overall picture on data clusters and outliers (**T5**), we select dimension reduction methods to display the training data for a better scalability. Then, we draw a confusion matrix to help users observe the model's performance on these data and enable users to filter certain data subsets (**T7**). Finally, we display a data table that allows users to easily browse the whole training dataset.

This view may have the visual clutter problem, which is a common drawback for many dimension reduction-based visualizations. To alleviate this issue, we adopt an overlap removal algorithm [18] to increase the distances between overlapping circles. We further support zooming and panning to help users focus on a specific region in the view.

The confusion matrix is illustrated as four horizontal rectangles at the top. The rectangle width encodes the size of the corresponding category, while the color and texture represent different prediction categories. For example, true positives are illustrated in deep blue, and false positives are represented in light blue with stroke texture (Fig. 1A). Similarly, true negatives and false negatives are visualized in red. Compared with the traditional 2×2 matrix-based design, our design is more space-efficient and enables users to quickly compare the sizes of different prediction categories. Users can click on a rectangle to filter a category of data instances. The opacity of the corresponding circles below is set to zero, while we keep the strokes

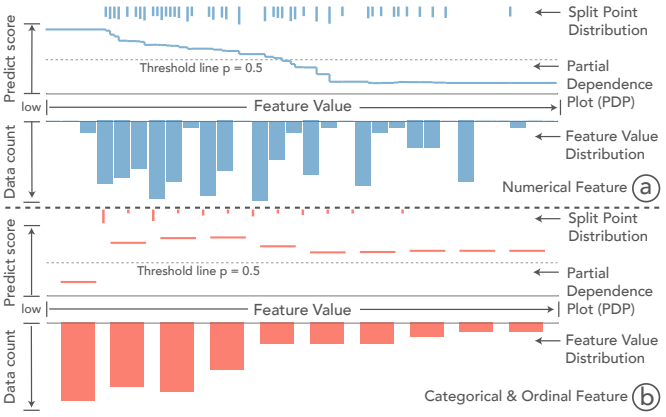

Figure 2. Feature View's visual encoding for (a) numerical and (b) categorical features. The upper parts show the split point density and partial dependence information and the lower parts illustrate the feature value distribution.

of them to provide contextual information. In the lower part, actual feature values are shown in a table to allow users to quickly browse and select a data instance for inspection.

7.2 Feature View

Since random forest models are a mapping from input features and output predictions, revealing the relationships between features and predictions can help analyze what models have learned from training data and facilitate the understanding of models. For example, users may want to know which features are most important for a model and how feature values affect predictions (**T1**). In addition, the split point distribution of each feature (**T2**) and the training data value distribution (**T6**) also reflect the relationships between features and predictions. Therefore, we design the Feature View to reveal such information and facilitate users' understanding of random forest models.

Visual Encoding. The Feature View visualizes feature information as a list (Fig. 2), in which each row represents a feature. Features are sorted based on the Gini importance (Sec. 2). The design of each feature row contains two parts as shown in Fig. 2a. The upper part displays the partial dependence information and split point distribution, while the bottom part shows the training data distribution. These two parts share an x-axis, which represents the current feature values in ascending order of Gini importance (from left to right).

In the upper part, we use a line chart to represent the partial dependence plots and a pixel-based bar chart to encode the split point distribution. For a feature $f^m \in F = \{f^1, f^2, ..., f^M\}$, let $C = F - \{f\}$ to be the complement set of f, the partial dependence is calculated as:

$$PDP_{f^m}(\alpha) = \frac{1}{N}\sum_{i=1}^{N} RF(x_i^C, x_i^m = \alpha) \quad (2)$$

where N denotes the size of training data, function $RF(x)$ represents random forest models that take the training data as input and output the prediction in the form of probability. The partial dependence calculates the average value of prediction f of all the training data, while x_i^C is fixed but setting the value of feature m to α for each data item. The y-axis represents the predictions of the model and the values that range from 0 to 1 are in ascending order from bottom to top. Since we focus on binary classification in this paper, we set the threshold to 0.5 so that the prediction is positive when value is larger than 0.5 and negative otherwise. Although we support both numerical and ordinal data, the line chart has different visualization formats for these two types of features (Fig. 2b). For numerical data, the partial dependence information is illustrated as a continuous line chart; for ordinal data, it is displayed as discrete horizontal lines. To help users better evaluate predictions, we use a gray dashed horizontal line to highlight the threshold (0.5 by default). Thus, if the curve is above the dashed line, it indicates that the prediction is positive, otherwise negative. To

allow users explore the effect of features on the predictions at different scales, we support switching between two scales of y-axis. The default scale of y-axis is ranging from 0 to 1, which is useful for users to directly evaluate which features have large effect on the prediction. Another scale of y-axis ranges from the minimum to the maximum prediction of each particular feature, which allows users to examine the relations between a specific feature and the predictions in details. In addition, we draw a pixel-based bar chart at the top of the line chart to represent the split point density in which the bar height encodes the number of split points at this position. In the bottom part, we use the bar charts to represent the feature value distributions and the inverted y-axis represents the feature value, where the feature value increases from top to bottom. The length of each bar represents the number of data items with the corresponding feature value.

Design Alternatives. When designing the Feature View, we considered several alternatives. We considered heatmap and line charts in designing the partial dependence plots. We chose line charts over heatmap, because users may feel difficult to perceive small differences in heatmap [39]. In order to maintain a consistent design, we chose histogram to obtain the coherent visualization style. In addition, we have considered to overlay the partial dependence plot on the value and split point distributions. Though this design is space-efficient, we abandoned it for several reasons. First, the design would cause occlusion problems that users may have difficulty in perceiving the partial dependence information and the size of training data in the same feature value range. In addition, the overlapping of histograms and the plots may lead to interpretation errors. Therefore, based on the above reasons, we select our design as the final design choice.

7.3 Decision Path View

Users usually need to flexibly audit the decision process of a prediction before making final decisions (**T7**). When auditing a prediction, the interim predictions and the corresponding prediction scores (**T3**) are an important aspect to examine. Moreover, each interim prediction has a unique decision path, thus summarizing and comparing the structures of various decision paths can provide deeper insights into the underlying working mechanisms (**T4**). Thus, this view is designed to help users audit, summarize, and compare decision paths. The Decision Path View (Fig. 1C) is a list in which all decision paths of a prediction is represented as a row. Specifically, each row contains three major visual components: 1) the *Decision Path Projection* that provides an overview of all decision paths in a single prediction based on their similarities, 2) the *Feature Summary* that summarizes the critical feature ranges of decision paths, and 3) the *Decision Path Flow* that provides the detailed information of decision paths layer by layer.

Algorithm 2 Decision Path Distance

Input: Decision Paths p_i, p_j, Feature list $F = \{f^1, f^2, ..., f^M\}$
Output: Distance $dist_{ij}$

```
for each feature f^m in F = {f^1, f^2, ..., f^M} do
    if f^m not in F_{p_i} and f^m not in F_{p_j} then
        continue
    else if f^m not in F_{p_i} or f^m not in F_{p_j} then
        dist_ij = dist_ij + 1
        feature_cnt = feature_cnt + 1
    else
        dist_ij = 1/2 (|θ^{f^m,l}_{p_i} − θ^{f^m,l}_{p_j}| + |θ^{f^m,u}_{p_i} − θ^{f^m,u}_{p_j}|)
        feature_cnt = feature_cnt + 1
    end if
end for
dist_ij = dist_ij / feature_cnt
```

Decision Path Projection. Since a single prediction in random forests is generated based on the results of many individual decision paths, we aim to provide users with an overview of these decision paths (**T3**). Similar to the Data Overview, we use t-SNE to project all the decision paths onto a 2D plane (Fig. 3a) as circles so that users can easily observe their similarities. The pairwise distances of decision paths can be calculated according to the feature ranges. Specifically, for a root-to-leaf decision path p (Eq. 1) of a data instance $x = \{x^1, x^2, ..., x^M\}$, a feature may occur multiple times in the decision path. Let the set of features used in path p be $F_p \subseteq F = \{f^1, f^2, ..., f^M\}$. For each $f^n \in F_p$, we can merge the ranges of f^n on the path to $[\theta_p^{f^n,l}, \theta_p^{f^n,u}]$, where $\theta_p^{f^n,l} = Max\{\theta_{t_h} \mid f_{t_h} = f^n, \otimes = \text{“}>\text{”}, t_h \in t(p)\}$, and $\theta_p^{f^n,u} = Min\{\theta_{t_h} \mid f_{t_h} = f^n, \otimes = \text{“}\leq\text{”}, t_h \in t(p)\}$.

To generate the t-SNE embedding, we normalize the feature range to $[0, 1]$. The normalized feature range of the path can be written as:

$$p = \{[\theta_p^{f^n,l}, \theta_p^{f^n,u}] \mid f^n \in F_p\} \tag{3}$$

We can calculate the pairwise decision path distance in Algorithm 2.

We also considered including feature orders when measuring decision path distances. The advantage of including feature orders is that the calculated distance can better represent the dissimilarities between decision paths with less information loss. However, this approach may position two decision paths with similar features apart from each other due to the order effect. As the Decision Path Projection mainly aims at providing an overview of decision path similarity to guide users for further exploration, we exclude feature orders when calculating decision path distances to better cluster the paths that share a common feature set. Users can further compare the feature orders of different decision paths using the Decision Path Flow component.

As in the Data Overview, we use a red-to-blue divergent color scheme to encode the prediction score, from negative to positive. On top of the Decision Path Projection, we also draw a divergent bar and use the bar width to indicate the number of decision paths with positive or negative labels. User can click on a circle or draw a lasso (Fig. 3f) to add the corresponding decision paths to the Feature Summary and Decision Path Flow for further exploration.

Feature Summary. For the decision paths that have been selected in the Decision Path Projection, their features and corresponding ranges are summarized in Feature Summary. Specifically, each feature occurred in the selected paths is represented using a *Feature Cell* in a list format (Fig. 3b). Users can scroll the list when the number of features listed is large. Feature Cell is a pixel-based bar chart design (Fig. 3d) that summarizes the feature ranges of multiple decision paths on feature f. We use the same categorical color scheme of the Feature View to represent different features. The x-axis represents feature values in ascending order from left to right, and the y-axis encodes the number of decision paths in which their feature ranges covers this value. We also draw a vertical gray bar to encode the current data instance's value x^f. From the height of the bar chart, we can estimate which ranges of the feature are considered critical.

Decision Path Flow. The Decision Path Flow aims at revealing the structures and properties of multiple decision paths at the layer level. This enables users to examine the orders of the features appeared in different decision paths, which is critical in measuring feature importance. As shown in Fig. 3c, each column represents a layer, in which the layer depth increases from left to right. Similar to the Feature Summary, we use Feature Cells to summarize features and the corresponding ranges of nodes from the same layer. This helps users examine how the feature ranges of different decision paths evolve from the root layer to the leaf layer. If a layer contains leaf nodes, we append a pie chart in the corresponding column, where the red sector represents negative labels and the blue sector represents positive labels (Fig. 3e). The pie chart radius encodes the number of decision paths that have leaf nodes at this layer and the sector angle encodes the ratio of the paths of the corresponding label. We draw curves to connect features from different layers. The curve width encodes the number of decision paths that have the corresponding feature pair in adjacent layers.

8 Evaluation

In this section, we describe two usage scenarios and a qualitative user study that demonstrate the effectiveness of iForest. The dataset descriptions and user study materials, such as task description, questionnaires, and study results, can be found in our supplemental materials.

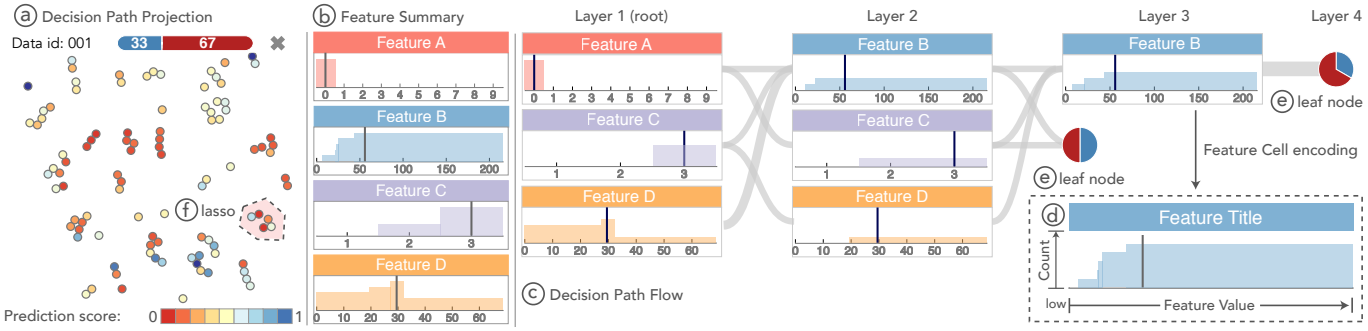

Figure 3. Visual components of the Decision Path View. (a) The Decision Path Projection provides an overview of decision path similarities. (b) The Feature Summary shows the summarized feature ranges for multiple selected decision paths. (c) The Decision Path Flow visualizes the detailed structures and feature ranges layer by layer. (d) The Feature Cell depicts the summarized feature range. (e) The pie chart displays the label distribution on the leaf node. (f) Lasso can be drawn around multiple circles to examine the details of the corresponding decision paths.

8.1 Usage Scenario I

In this usage scenario, we describe Daniel, a machine learning beginner, utilizes iForest to investigate which groups of people were likely to survive in the Titanic shipwreck and how random forests make the corresponding predictions. The label to predict is *Survival*, a binary variable in which survival is marked as positive. Daniel first performs feature engineering on the training data [5] which contains 891 passengers, to combine similar or redundant features together for model simplicity and robustness. He generates six features, namely, *Sex*, *Title*, *Fare*, *PClass*, *Age*, and *Family Size*, to train a random forest model.

Daniel loads the trained model into iForest and he first examines the Feature View to investigate which features are more critical for survival. As *Sex* ranks top in the Feature View (Fig. 1B), Daniel recognizes that *Sex* is the most important feature related to predictions (**T1**). From the partial dependence plots, he also observes that passengers tend to survive when $Sex = 0$ (female) but not to survive when $Sex = 1$ (male). This suggests that the model's learned pattern aligns with the fact that female passengers indeed are more likely to survive in the Titanic shipwreck [27].

Apart from categorical features like *Sex*, Daniel likes to further understand the relationships between prediction labels and numerical features, such as *Fare* and *Age*. Comparing *Fare* with *Age*, *Fare*'s split points are mainly located at the beginning of the x-axis (Fig. $1b_1$) while the split points of *Age* are distributed more evenly (Fig. $1b_2$) (**T2**). This suggests when examining *Fare*, the random forest model mainly checks if the values are smaller than a certain threshold. In contrast, there are no significant thresholds for predictions when examining *Age*. To examine the relationships between features and predictions, Daniel switches the y-axis of partial dependence plot from global scale to local scale. He finds that there is a steep curve in *Fare*'s partial dependence plot around the value 25 (Fig. $1b_3$). To discover the differences between the passengers divided by the threshold, Daniel hovers on the corresponding bar charts to highlight and compare the feature value distribution of the passengers in different Fare ranges (**T7**). He identifies that passengers with $Fare > 25$ are usually from the first class and are more likely to survive than passengers with $Fare < 25$ from the lower classes.

After examining the feature-prediction relationships learned by the random forest model, Daniel explores the Data Overview to analyze whether the random forest model fits the training data well and captures the characteristics of the training data. Daniel identifies several clusters (Fig. 1A) with different colors (**T5**). For example, cluster a_2 and a_3 are both encoded with dark red color, which indicates these passengers are predicted as non-survival by the model with high confidences. By hovering over the circles and observing their feature values, Daniel finds that these passengers are mainly male from the third class. Apart from cluster a_2, Daniel also identifies six outliers (Fig. $1a_1$) with light yellow color (**T5**). He discovers that these outliers are also third class male passengers but their *Fares* are 56. To examine whether the model produces correct predictions on these six passengers, Daniel filters out *True Negatives* and discovers that these six passengers are *False Negatives*, which indicates that they survived.

To analyze the reasons why the model generates predictions on these passengers with a low confidence, Daniel clicks one of the six circles to add this passenger (*ID = 664*) to the Decision Path View for further examination. Fig. $1c_2$ shows that, apart from the many red circles that represent the decision paths with negative predictions, there are also some blue circles that represent positive predictions in the Decision Path Projection (**T3**). To explore the working mechanism differences between the positive and negative predictions of this passenger, Daniel first draws a lasso around the red circles to explore the decision paths with negative predictions (Fig. $1c_1$). He also adds this passenger's account data copy to the Decision Path View and draws a lasso around the blue circles (Fig. $1c_2$) for comparison. Comparing the Feature Summary of the positive and negative decision paths, Daniel finds that the negative decision paths examine *Sex* and *Class* (Fig. $1c_3$) while the positive decision paths check *Fare*, in which this passenger has abnormal large values (**T4**). In addition, Daniel identifies that most of the negative decision paths use 3 as the threshold for feature *Family Size*. However, for the decision paths with positive predictions, the *Family Size* threshold is 1 (Fig. $1c_4$). This suggests that in addition to feature types, these two groups of decision paths may even have different thresholds of the same feature in predictions (**T4**).

8.2 Usage Scenario II

We use the *German Credit Data* [3] to demonstrate this usage scenario. This dataset contains 1000 bank accounts with 9 features, each having a prediction label of good credit (positive) or bad credit (negative). Some of the features are ordinal (e.g., *Account Balance*'s range is $[1,4]$, in which larger number indicates larger amount of balance). In this usage scenario, we describe Emma, a data scientist working at a bank to help her manager in analyzing whether a loan applicant has a good credit. She utilizes a random forest model built on this dataset to predict the credit status.

Emma first uploads a loan applicant's information to iForest and observes that the model predicts that the applicant has a bad credit. From the Decision Path View's prediction label distribution (Fig.4a), she notices that the number of positive and negative decisions is similar, which indicates this is likely to be a marginal case. To ensure the prediction is reliable, Emma examines the Decision Path Projection in the Decision Path View to check the result and the corresponding confidence of each decision path. From the color saturations of circles, she observes that though negative decision paths outnumber positive decision paths, several positive decision paths have high confidence (Fig. 4b) (**T3**). To examine why these positive decision paths achieve high confidence, she draws a lasso around these circles to examine them in detail. From the Feature Summary in the Decision Path View, she observes that the highly adopted ranges in these decision paths are *Account Balance* $\in [3.5,4]$, *Duration of Credit* ≥ 30, and *Length of Current Employment* ≥ 3.5 (**T4**). Since the corresponding three feature values of the applicant belong to these three feature ranges, these decision paths generate positive predictions with high

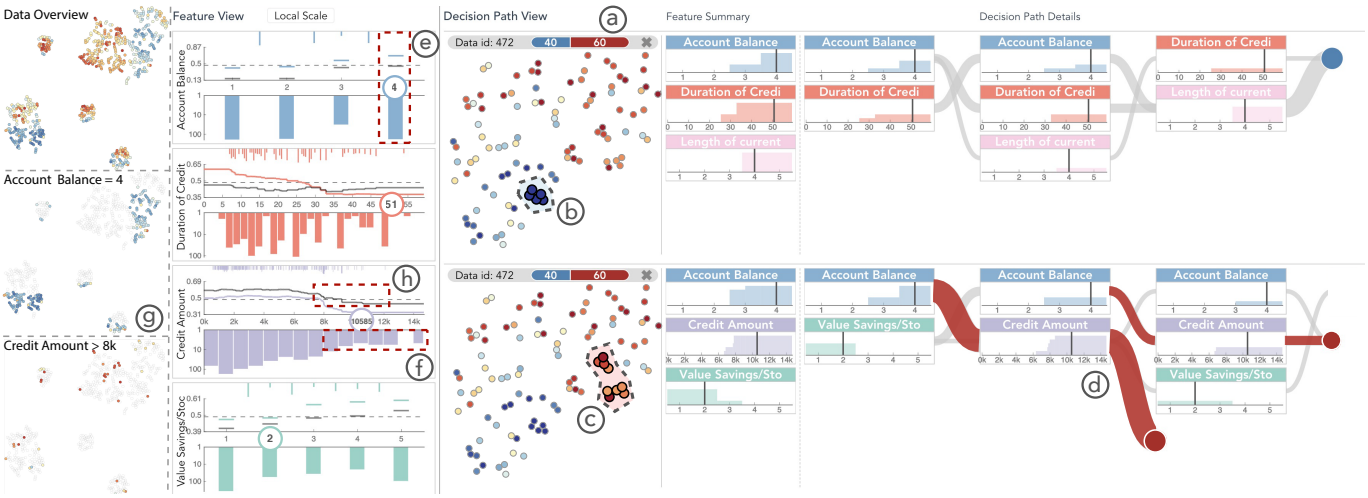

Figure 4. Auditing a prediction with German Credit Data using iForest: (a) the prediction distribution of all decision paths; (b) the positive decision paths with high confidence; (c) the negative decision paths with high confidence; (d) the summarized decision rule from *Account Balance* $\in [2.5, 4]$ to *Credit Amount* $\geq 8k$; (e) the number of accounts with *Account Balance* $= 4$; (f) the number of accounts with *Credit Amount* $\geq 8k$; (g) the accounts with *Account Balance = 4*; (h) the local partial dependence plot of *Credit Amount* $\in [8000, 10144]$.

confidence. Then, Emma needs to investigate why other decision paths produce negative results so that she can better evaluate the strengths and weaknesses of this application. To find the strongest reason behind the rejection, Emma draws a lasso around the circles with dark red color (Fig. 4c), which indicates that these negative decision paths have high confidence. She observes that the important ranges for rejecting the application are *Account Balance* $\in [2.5, 4]$, *Credit Amount* $\geq 8k$, and *Value Savings / Stocks* ≤ 2.5. It is strange that *Account Balance* $\in [2.5, 4]$ is similar to the feature range used in those positive decision paths of high confidence. In order to understand why a feature range is critical in both positive and negative decision paths with high prediction confidence, Emma further examines the Decision Path Flow to check the details of path structures. She observes that though *Account Balance* occurs in most of the decision paths, it appears mostly in the first and second layers while *Credit Amount* $\geq 8k$ appears mostly in leaf layers (Fig. 4d). To investigate these two ranges regarding case-based reasoning, Emma switches to the Feature View to examine account distributions (**T6**). From the feature distribution histograms, Emma identifies that a great number of accounts (Fig. 4e) have an *Account Balance* within $[2.5, 4]$, which indicates large balance amounts. Meanwhile, there are only a few people whose *Credit Amount* are larger than $8k$ (Fig. 4f). Furthermore, by hovering over the histogram bars ranging from $[2.5, 4]$ for *Account Balance*, she observes that many blue circles are highlighted in the Data Overview (Fig. 4g), which indicates many accounts have good credit (**T5**). However, many accounts are in bad credits with *Credit Amount* larger than $8k$.

Linking this observation to the decision paths in the Decision Path View, Emma realizes that *Account Balance* $\in [2.5, 4]$ can separate the dataset into two coarse-grained large groups based on labels. However, for the data in the group of *Account Balance* $\in [2.5, 4]$, the *Credit Amount* $\geq 8k$ actually is the main reason for these decision paths generating negative prediction results with high confidence as it partitions the data in a finer granularity. As *Credit Amount* is a very critical factor in approving loan application according to the bank policy, Emma decides to suggest rejecting this application to avoid risk.

To provide explanations and suggestions, Emma decides to investigate what actions can be taken to change this loan application from negative to positive. She first examines the local partial dependence plots of features in the Feature View (**T1**) to inspect how the predictions are affected by the feature values. As *Credit Amount* is a strong factor, she first examines the column representing *Credit Amount*. As shown in Fig. 4h, from the local partial dependence plot, when the value of *Credit Amount* decreases from the current value $10,144$ to $8,000$, the prediction score keeps increasing. In the Feature View, Emma drags the circular label mark of *Credit Amount*, which represents the current value of *Credit Amount*, to inspect the impact of *Credit Amount* on the prediction so that she can suggest a feasible value to the applicant (**T7**). When Emma reduces the *Credit Amount* value, she observes the minimum change to turn the prediction into positive is by setting *Credit Amount* to $8,035$. Therefore, based on the above analysis, Emma reports to reject this application and suggests reducing the *Credit Amount*.

8.3 Qualitative User Study

A formal comparison between iForest and a baseline visualization system is not applicable because existing techniques mainly focus on a single decision tree or the diagnosis of tree-ensemble models, which are different from the goal of iForest. In addition, random forest interpretation includes a series of complex tasks that may require users to navigate between the data, feature, and model perspectives, and link them together to understand how random forests generate certain predictions, which is not a simple yes or no question. Therefore, we choose to conduct a qualitative user study instead of a controlled quantitative experiments.

Design and Procedure. From a local university and an industry research lab, we recruited 10 participants (7 males, aged 21 to 28 years (mean = 24.4, SD = 2.5)) including undergraduate students, graduate students, and research scientists. Four of them have knowledge in information visualization; two of them have experience in machine learning but not familiar with random forests. The entire study took approximately 45 minutes to finish on average.

We began each study with a tutorial on random forests and an introduction to iForest's interface. The participants were asked to think aloud. When introducing iForest, we used the Titanic dataset to familiarize the participants with iForest. Then, we encouraged the participants to freely explore our system and try different interactions.

In the formal study, we adopted the German Credit dataset for users to complete tasks to avoid the memorization effects. For each task, we recorded the task completion time and the problems encountered by the participants. After finishing all the ten tasks, we asked the participants to complete a questionnaire with 12 questions to evaluate the effectiveness and aesthetics of iForest. Each question was designed using a 7-point Likert scale from strongly disagree (1) to strongly agree (7). We also conducted a post-study interview to collect user feedback on iForest, such as which parts need to be improved.

Results and Discussion. According to our design goals (Sec. 4), we designed 10 tasks to cover different perspectives of random forest interpretation. Tasks 1–3 pertain to case-based reasoning for gaining a holistic picture of random forests. Tasks 4–10 pertain to auditing a prediction to uncover the underlying working mechanisms and exploring

the relationships between features and predictions. Successfully completing all the tasks requires the participants to utilize all the views in iForest. In general, the participants had completed the tasks successfully within a short period of time. However, two tasks (Task 9 and Task 10) took longer time than other tasks. This may be because that Task 9 requires the participants to examine different Feature Cells for summarization and Task 10 requires participants to manually tweak feature values. For the questionnaire, the majority of participants declared that the interactions are easy (6.3), the system is visually pleasing (6.5) and useful for interpret random forest models (6.7).

In the post-study interview session, most participants valued the effectiveness of iForest in helping them understand random forests and how a specific prediction is made. Particularly, they appreciated the usefulness of the Decision Path View and the Feature View. For the Decision Path View, a participant remarked that the feature ranges "can help me examine why a point is false positive. It is because some of its feature values are close to the decision thresholds." For the Feature View, one participant commented that "the Feature View is very intuitive, I can see all the feature and their distributions. It is easy to understand which features are important for decision making." Besides the positive feedbacks, the participants also provided many suggestions on improving iForest. For the Feature Summary of the Decision Path View, some participants mentioned that they need time to understand the meaning of the visual encodings. This indicates that for some people, the learning curve of iForest might be steep.

Apart from the visual design, most participants valued that iForest can be helpful for random forest interpretation in many domains. Some participants commented that "it can be useful for me to sell my used car at a good price. I can see which features are important for the prices and which parts I should fix." Some other participants also appreciated the usefulness of iForest for education and learning.

9 Discussion

Generalization. Although iForest mainly focuses on interpreting random forests, it can be applied to other tree-based ensemble models, such as boosting trees [22]. These ensemble models all consist of many decision trees and generate final predictions by summarizing the output of all internal trees. Unlike random forests, in which decision trees are independently trained, decision trees in boosting tree models are trained in a sequential order. One potential enhancement to support boosting trees in iForest is encoding the temporal information. For example, we can enable users to switch the color encoding of the scatter plot so that they can compare decision paths from different time steps and understand how boosting trees evolve along time.

In addition, iForest can be easily extended to support multi-class classification and regression tasks. For classification tasks, one potential improvement is to apply the one-vs-rest strategy. For example, we can label one class as the positive class and all other classes as negative classes. For regression tasks, we can change the confusion matrix in the Data Overview to a violin diagram or bar charts to represent the prediction error distribution. Then, users can brush to select a specific group of data items for further examination.

Lastly, although the focus on iForest is to interpret random forests, it can also be extended to diagnose and debug random forests. For example, we can visualize and compare the feature importances on different trees to help feature engineering. From the evaluation perspective, we can further enable users to switch between different model evaluation measures, such as ROC-AUC [19] or training loss [46].

Scalability. Similar to many visualization systems, one critical issue we considered when designing iForest is scalability.

In the Data Overview, circles may overlap when the number is large. To address this issue, we first adopt an overlap removal algorithm to enlarge the distances between overlapped circles. We also support users to focus on a specific region of circles via interactions, such as panning and zooming. In general, we support hundreds to thousands of data items in the Data Overview. When the data size becomes even larger, we can sample the data similar to Liu et al.'s approach [35].

In the Feature View, we use categorical colors to represent different features. Since humans may have problems in perceiving more than ten categorical colors [55] at the same time, iForest colors the top ten features with largest importance. Other less important features are colored in gray. We support to visualize six to ten features in the same window depending on the display size, which is usually sufficient since previous research indicates that human has a limited visual capacity that around three to seven objects [50]. When the number of features increases, users can use a scroll bar to examine them and we do not limit the number of total features in datasets.

In the Decision Path View, iForest can support hundreds of paths from different trees in the projection, which is usually adequate for training a good random forest model [42]. When the number of decision paths increases, we use similar approaches adopted by the Data Overview to alleviate the clutter problems. Depending on different screen ratios, the Feature Summary and Decision Path Flow components generally support to visualize four consecutive decision nodes in a screen (Fig. 1C). Users can scroll these two components to explore longer decision paths and we do not limit the maximum decision path length. Besides, as we use curves to link the nodes from different layers, these curves may have overlapping problems. However, this view mainly focuses on exploring a few similar paths. Thus, the edge number between each layer is limited, which makes the view less cluttered. In case more decision paths are investigated, users can view their feature summaries instead of the detailed layer information.

We conducted our evaluation using a MacBook Pro with 2.9GHz Intel Core i9 CPU with 16GB memory as the web server and Chrome (version 63) as the browser. The system can perform interactively after the model training stage, which may take a few minutes for the datasets used in the usage scenarios.

Targeted users. Our targeted users are the data scientists from various domains who apply random forests in solving their problems and demand interpretable predictions. We assume that they have basic knowledge on random forests but encounter difficulty in interpreting them. Note that the machine learning experts who have rich knowledge in machine learning may also benefit from using iForest in understanding the predictions made by random forests. Though they may know how random forests are constructed, the variety and complexity of the many trees still hinder them from understanding what a trained model has learned and how the decisions are made in a prediction.

10 Conclusion

We have presented iForest, an interactive visualization system that helps users interpret random forest models from various perspectives. The system reveals relations between input features and output predictions, hence enabling users to flexibly tweak feature values to monitor prediction changes. It also helps users audit the decision process of predictions to explore the underlying working mechanisms. Our evaluation results show that iForest can effectively assist users in understanding random forest models and their predictions.

iForest has several promising directions for future research. First, instead of feature importance and partial dependence information, we aim to further analyze feature correlations from the model perspective so that users can conduct feature engineering to improve model performance. Second, we plan to conduct more comprehensive user studies to further evaluate the effectiveness of iForest, such as recruiting more participants, conducting a quantitative study and comparing with other interpretation tools. In addition, we like to improve the scalability of our system for larger dataset, such as sampling the most representative data items instead of random sampling in the Data Overview.

11 Acknowledgments

The authors would like to thank Prof. Huamin Qu from HKUST and Dr. Enrico Bertini from NYU for their constructive suggestions of revising the paper and the anonymous reviewers for their valuable comments. This research was supported in part by Hong Kong Theme-based Research Scheme grant T41-709/17N.

References

[1] D3. https://d3js.org/.
[2] Flask. http://flask.pocoo.org/.

[3] German credit data. https://archive.ics.uci.edu/ml/datasets/Statlog+%28German+Credit+Data%29.

[4] Scikit-learn. http://scikit-learn.org/stable/.

[5] Titanic. https://www.kaggle.com/c/titanic.

[6] M. Ankerst. Visual data mining with pixel-oriented visualization techniques. In *Proc. of the SIGKDD Workshop on Visual Data Mining*, 2001.

[7] M. Ankerst, M. Ester, and H.-P. Kriegel. Towards an effective cooperation of the user and the computer for classification. In *Proc. of the International Conference on Knowledge Discovery and Data Mining*, pages 179–188, 2000.

[8] I. Bichindaritz and C. Marling. Case-based reasoning in the health sciences: What's next? *Artificial Intelligence in Medicine*, 36(2):127–135, 2006.

[9] O. Biran and K. McKeown. Human-centric justification of machine learning predictions. In *Proc. of the International Joint Conference on Artificial Intelligence, 2017*, pages 1461–1467, 2017.

[10] L. Breiman. Random forests. *Machine learning*, 45(1):5–32, 2001.

[11] L. Breiman. Statistical modeling: The two cultures. *Statistical Science*, 16(3):199–231, 2001.

[12] L. Breiman. Looking inside the black box. *Wald Lecture II, Department of Statistics, California University*, 2002.

[13] L. Breiman. Manual on setting up, using, and understanding random forests v3.1. *Statistics Department Univ. of California Berkeley*, 1, 2002.

[14] R. Caruana, H. Kangarloo, J. Dionisio, U. Sinha, and D. Johnson. Case-based explanation of non-case-based learning methods. In *Proc. of the AMIA Symposium*, page 212, 1999.

[15] R. Caruana, Y. Lou, J. Gehrke, P. Koch, M. Sturm, and N. Elhadad. Intelligible models for healthcare: Predicting pneumonia risk and hospital 30-day readmission. In *Proc. of the International Conference on Knowledge Discovery and Data Mining*, pages 1721–1730, 2015.

[16] H. Deng. Interpreting tree ensembles with intrees. *arXiv preprint arXiv:1408.5456*, 2014.

[17] R. Díaz-Uriarte and S. A. De Andres. Gene selection and classification of microarray data using random forest. *BMC Bioinformatics*, 7(1):3, 2006.

[18] T. Dwyer, K. Marriott, and P. J. Stuckey. Fast node overlap removal. In *International Symposium on Graph Drawing*, pages 153–164, 2005.

[19] T. Fawcett. An introduction to roc analysis. *Pattern Recognition Letters*, 27(8):861–874, 2006.

[20] M. Fernández-Delgado, E. Cernadas, S. Barro, and D. Amorim. Do we need hundreds of classifiers to solve real world classification problems. *Journal of Machine Learning Research*, 15(1):3133–3181, 2014.

[21] J. Friedman. Greedy function approximation: a gradient boosting machine. *Annals of Statistics*, pages 1189–1232, 2001.

[22] J. Friedman, T. Hastie, and R. Tibshirani. *The elements of statistical learning*, volume 1. Springer Series in Statistics New York, 2001.

[23] C. Gallego-Ortiz and A. L. Martel. Interpreting extracted rules from ensemble of trees: Application to computer-aided diagnosis of breast mri. *arXiv preprint arXiv:1606.08288*, 2016.

[24] R. Genuer, J.-M. Poggi, and C. Tuleau-Malot. Variable selection using random forests. *Pattern Recognition Letters*, 31(14):2225–2236, 2010.

[25] T. Gindele, S. Brechtel, and R. Dillmann. Learning driver behavior models from traffic observations for decision making and planning. *IEEE Intelligent Transportation Systems Magazine*, 7(1):69–79, 2015.

[26] R. Guidotti, A. Monreale, F. Turini, D. Pedreschi, and F. Giannotti. A survey of methods for explaining black box models. *arXiv preprint arXiv:1802.01933*, 2018.

[27] W. Hall. Social class and survival on the ss titanic. *Social Science & Medicine*, 22(6):687–690, 1986.

[28] B. Kim, C. Rudin, and J. A. Shah. The bayesian case model: A generative approach for case-based reasoning and prototype classification. In *Advances in Neural Information Processing Systems*, pages 1952–1960, 2014.

[29] J. Kolodner. *Case-based reasoning*. Morgan Kaufmann, 2014.

[30] J. Krause, A. Perer, and K. Ng. Interacting with predictions: Visual inspection of black-box machine learning models. In *Proc. of the CHI Conference on Human Factors in Computing Systems*, pages 5686–5697, 2016.

[31] J. B. Kruskal and J. M. Landwehr. Icicle plots: Better displays for hierarchical clustering. *The American Statistician*, 37(2):162–168, 1983.

[32] H. Lakkaraju, E. Kamar, R. Caruana, and J. Leskovec. Interpretable & explorable approximations of black box models. *arXiv preprint arXiv:1707.01154*, 2017.

[33] H. Li and J. Sun. Ranking-order case-based reasoning for financial distress prediction. *Knowledge-Based systems*, 21(8):868–878, 2008.

[34] Z. C. Lipton. The mythos of model interpretability. *arXiv preprint arXiv:1606.03490*, 2016.

[35] S. Liu, J. Xiao, J. Liu, X. Wang, J. Wu, and J. Zhu. Visual diagnosis of tree boosting methods. *IEEE Trans. on Visualization and Computer Graphics*, 24(1):163–173, 2018.

[36] Y. Liu and G. Salvendy. Design and evaluation of visualization support to facilitate decision trees classification. *International Journal of Human-computer Studies*, 65(2):95–110, 2007.

[37] G. Louppe, L. Wehenkel, A. Sutera, and P. Geurts. Understanding variable importances in forests of randomized trees. In *Advances in Neural Information Processing Systems*, pages 431–439, 2013.

[38] T. Mühlbacher, L. Linhardt, T. Möller, and H. Piringer. Treepod: Sensitivity-aware selection of pareto-optimal decision trees. *IEEE Trans. on Visualization and Computer Graphics*, 24(1):174–183, 2018.

[39] T. Munzner. *Visualization analysis and design*. CRC press, 2014.

[40] A. Newell and H. A. Simon. *Human problem solving*, volume 104. Prentice-Hall Englewood Cliffs, NJ, 1972.

[41] T. D. Nguyen, T. B. Ho, and H. Shimodaira. Interactive visualization in mining large decision trees. In *Pacific-Asia Conference on Knowledge Discovery and Data Mining*, pages 345–348. Springer, 2000.

[42] T. M. Oshiro, P. S. Perez, and J. A. Baranauskas. How many trees in a random forest? In *International Workshop on Machine Learning and Data Mining in Pattern Recognition*, pages 154–168. Springer, 2012.

[43] A. Palczewska, J. Palczewski, R. M. Robinson, and D. Neagu. Interpreting random forest classification models using a feature contribution method. In *Integration of Reusable Systems*, pages 193–218, 2013.

[44] A. Palczewska, J. Palczewski, R. M. Robinson, and D. Neagu. Interpreting random forest classification models using a feature contribution method. In *Integration of Reusable Systems*, pages 193–218. 2014.

[45] P. Płoński and K. Zaremba. Visualizing random forest with self-organising map. In *International Conference on Artificial Intelligence and Soft Computing*, pages 63–71. Springer, 2014.

[46] S. Ren, X. Cao, Y. Wei, and J. Sun. Global refinement of random forest. In *IEEE Conference on Computer Vision and Pattern Recognition*, pages 723–730, 2015.

[47] M. T. Ribeiro, S. Singh, and C. Guestrin. "why should I trust you?": Explaining the predictions of any classifier. In *Proc. of the International Conference on Knowledge Discovery and Data Mining*, pages 1135–1144, 2016.

[48] V. Schetinin, J. E. Fieldsend, D. Partridge, T. J. Coats, W. J. Krzanowski, R. M. Everson, T. C. Bailey, and A. Hernandez. Confident interpretation of bayesian decision tree ensembles for clinical applications. *IEEE Trans. on Information Technology in Biomedicine*, 11(3):312–319, 2007.

[49] B. Shneiderman. Tree visualization with tree-maps: 2-d space-filling approach. *ACM Trans. on Graphics*, 11(1):92–99, 1992.

[50] T. A. Sørensen and S. Kyllingsbæk. Short-term storage capacity for visual objects depends on expertise. *Acta Psychologica*, 140(2):158–163, 2012.

[51] H. F. Tan, G. Hooker, and M. T. Wells. Tree space prototypes: Another look at making tree ensembles interpretable. *arXiv preprint arXiv:1611.07115*, 2016.

[52] S. Urbanek. Exploring statistical forests. *Proceeding of the Joint Statistical Meeting, Springer*, 2002.

[53] S. Van Den Elzen and J. J. van Wijk. Baobabview: Interactive construction and analysis of decision trees. In *IEEE Conference on Visual Analytics Science and Technology*, pages 151–160, 2011.

[54] J. Wang, B. Yu, and L. Gasser. Concept tree based clustering visualization with shaded similarity matrices. In *Proc. of the IEEE International Conference on Data Mining*, pages 697–700, 2002.

[55] C. Ware. *Information visualization: perception for design*. Elsevier, 2012.

[56] M. Ware, E. Frank, G. Holmes, M. Hall, and I. H. Witten. Interactive machine learning: letting users build classifiers. *International Journal of Human-Computer Studies*, 55(3):281–292, 2001.

[57] I. Watson and F. Marir. Case-based reasoning: A review. *The Knowledge Engineering Review*, 9(4):327–354, 1994.

[58] A. Weller. Challenges for transparency. *arXiv preprint arXiv:1708.01870*, 2017.

[59] J. Zhang, L. Gruenwald, and M. Gertz. Vdm-rs: A visual data mining system for exploring and classifying remotely sensed images. *Computers & Geosciences*, 35(9):1827–1836, 2009.

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