MASTER

ObliqueTreeVis
Visual Analytics for Interpreting Oblique Decision Trees

Liu, Chicheng

Award date:
2022

EINDHOVEN UNIVERSITY OF TECHNOLOGY

Department of Mathematics and Computer Science
Visualization Research Group

ObliqueTreeVis: Visual Analytics for Interpreting Oblique Decision Trees

Master Thesis

Chicheng Liu

Supervisors:
Prof. dr. Anna Vilanova
dr. Dennis Collaris
dr. Sibylle Hess
Linhao Meng

Final version

Eindhoven, October 2022

Abstract

Due to their ease of interpretation, univariate decision tree induction techniques for classification problems, such as CART, C4.5, and ID3, are widely used in machine learning. However, because these trees use axis-parallel splits, which only look at one attribute of each node, they tend to get very big, which makes it harder to figure out what the tree means. Oblique decision tree induction algorithms, which divide the feature space in a multidimensional hyperplane using oblique splits, usually produce much smaller trees. However, individual oblique splits are difficult to interpret. In this research, we want to utilize visualization to improve the interpretability of oblique decision trees and to evaluate the interpretability of oblique decision trees in the context of visual analysis in comparison to univariate decision trees.

Preface

This thesis is my last project at TU/e and as a student. I'm delighted I spent two years of my Master's degree at Eindhoven; it was an interesting and unforgettable experience. I'm delighted I picked the Visualization Research Group as the origin of my graduation project, since I learned a great deal there. After completing this capstone, I will transition from student to visualization development engineer. I will use all I have learnt at TU/e in my new line of work.

I'd want to thank everyone who helped me with my graduation project. I'd like to start by thanking my supervisor, Professor Anna Vilanova, for all the help and direction she has given me. She has provided me with unselfish guidance throughout my graduation project and has resolved any issues that have arisen. The present shape of my thesis would not exist without her coaching. Second, I would like to extend my heartfelt appreciation to Dennis Collaris and Linhao Meng. They have provided me with substantial assistance in both the creation and design of the visualization system and the writing of the thesis. In addition, I would like to express my appreciation to Professor Sibylle Hess of the Data Mining Research Group for agreeing to serve on my examination committee.

I am also deeply grateful to Dr. Alfred Truong. His Ph.D. thesis [32], which summarized the oblique decision tree in detail, provided significant inspiration for this study.

Finally, I would want to thank my loving parents for their many years of support and trust. I would also want to thank my friends and classmates who have offered me assistance, listened to my thoughts, and assisted me in resolving my issues throughout the complex process of writing my thesis.

Contents

Contents **vii**

1 Introduction **1**
1.1 Motivation 1
1.2 Problem Definition 3
1.3 Contribution 3
1.4 Organization of the Report 3

2 Preliminaries **5**
2.1 Classification 5
2.2 Tree-structured Classification 6
2.2.1 Tree Construction 6
2.2.2 Oblique Decision Trees 7
2.3 Feature Contribution 9

3 Task Analysis **11**

4 Related Work **13**
4.1 Visualization for Data Understanding 13
4.2 Visualization for Model Understanding 16
4.2.1 Tree-based Visualization 16
4.2.2 Visualizing Feature Space Split in Decision Nodes 17
4.2.3 Visualizing Tree Interpretation of Decision Paths 19
4.3 Model Explanation 20

5 Data Processing **23**
5.1 Original Data 23
5.2 Trained Oblique Decision Tree 23
5.3 Data Manipulation 24

6 ObliqueTreeVis Visual Design **27**
6.1 ObliqueTreeVis Overview 27
6.2 Oblique Tree View 28
6.3 Feature View 31
6.4 Projection View 34

7 Evaluation **37**
7.1 Use Case: Palmer Penguin Species Classification 37
7.2 Use Case: Horseshoe Crab Species Classification 38
7.3 User Study 40

8 Conclusions and Future Work **43**
8.1 Conclusions 43
8.2 Future Work 43

Bibliography **45**

Appendix **47**

A User Study Questionnaire **49**

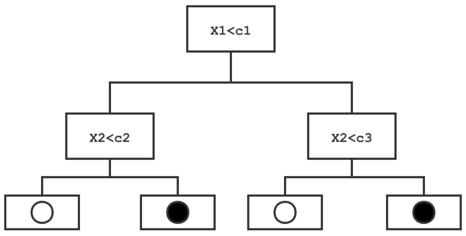
B User Study Results **51**

Chapter 1

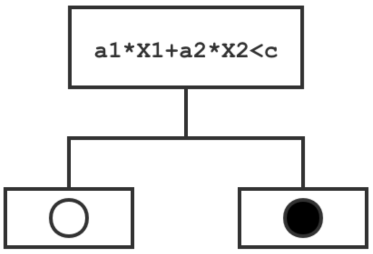
Introduction

1.1 Motivation

Decision trees are a popular machine learning method because they are easy to interpret. This type of model is used for classification tasks, which assign points to pre-specified categories called classes. It can be shown via a node-link diagram, as shown in Fig 1.1a. Each internal node in the node-link diagram represents a test in a feature space. Each test can be thought of as a function that divides the feature space into sub-spaces. Each link represents the result of the current parent node test, whereas each leaf node has a class label. By repeatedly applying the splitting function to the input, decision tree induction methods generate a top-down decision tree. The optimal choice rule is obtained by applying the splitting function to the remaining data at each node. The procedure terminates when all the training data points are divided into one of the classes. The primary difference between decision tree algorithms is how the splitting function works.



(a) A traditional decision tree requires more than one test to achieve the same classification of observations.



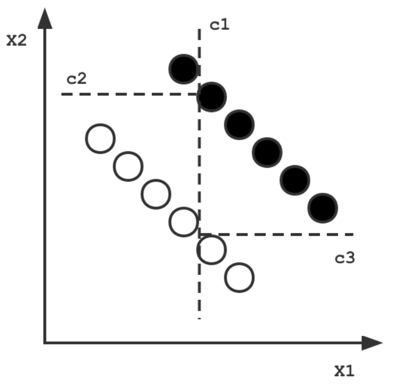
(b) An oblique decision tree is able to perfectly partition these observations.

Figure 1.1: This artificial classification problem can easily be solved with a single oblique split. However, axis-parallel splits cannot achieve the same purpose with a single split, *i.e.* oblique splits can segment data points better, so oblique decision trees are naturally smaller.

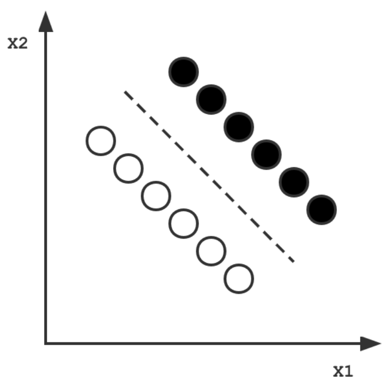
There are mainly two kinds of splitting functions. Axis-parallel splits are often applied in traditional decision trees such as CART and C4.5 [8, 28], while oblique decision trees like OC1 [23] use oblique splits. An axis-parallel split divides on a single feature variable, producing a hyperplane that is orthogonal to one axis and parallel to all others, as shown in Figure 1.2a. If the feature is a known quantity, a univariate split for that feature is easy to interpret. However, since only one feature variable is involved in each splitting function, such an approach may generate large tree sizes and thus increase the difficulty of understanding.

In contrast, an oblique split partitions on the linear combination of feature variables, generating an oblique hyperplane. The difference in the visual representation of the decision tree using different splitting functions can be seen in Figure 1.1. It is worth noting that for decision trees using axis-parallel split, some trees are larger than others simply because different classes of observations are not distributed in the same way in the feature space. This type of problem is not encountered for decision trees using oblique splits. As can be seen from Fig 1.2b, oblique splits are better able to partition observations; thus, oblique decision trees naturally use fewer tests.

On the other hand, univariate decision trees require a step-wise approximation of this rule. Therefore, multivariate split algorithms can achieve similar accuracy to parallel ones while producing smaller trees. However, while oblique decision trees may be smaller, their oblique splits are more challenging to interpret.



(a) Axis-parallel Split



(b) Bivariate Split

Figure 1.2: Comparison of Axis-parallel Split and Oblique Split in Bivariate Case

The size of the tree and splitting function understanding have a significant impact on the interpretability of decision trees. The decision tree gets increasingly challenging to understand as the tree grows in size. Similarly, suppose the splitting function is a linear equation with several feature variables. In that case, it is more difficult to understand than if just one feature variable is present. Because of the different splitting functions used, traditional and oblique decision trees have opposite characteristics in terms of internal node count and interpretability.

Visualization techniques have been demonstrated to be a practical approach for enhancing the interpretability of classifiers (*i.e.*, machine learning models targeting classification problems such as decision trees). Both traditional and oblique decision trees are interpretable classifiers and can be considered algorithms that learn from data. For such classification systems, visualization and visual analysis techniques have been used in different life cycles of the systems to enhance their interpretability. For instance, visualization techniques are applied to explore data during the data engineering stage or graphically represent the results of univariate decision tree algorithms in the model operation stage [20]. In addition, there is also work on visualization to support the decision tree in the stage of model development [33]. Nevertheless, little work has been done to support understanding oblique decision trees through visualization.

1.2 Problem Definition

In this research, we want to utilize visualization to improve the interpretability of oblique decision trees and evaluate the interpretability of oblique decision trees in the context of visual analysis compared to traditional decision trees. To achieve the research goal, we need to accomplish several tasks. First, we need visualization techniques to help people understand the input space. Even though tree structure models are usually considered interpretable, considering that our target users are domain experts, we should present the whole structure of the tree model, including decision nodes, leaf nodes, and decision paths. Besides, since the decision rules in each node of the oblique decision tree are equations consisting of multiple features, we also need to show the decision boundaries reflected by each rule with the help of visualization techniques. Also, our system should let users explore the links between features and decision nodes or decision paths in an interactive way. To help domain experts build trust in the model and make decisions in their daily work, we need to provide additional interpretation of the classification results. At the same time, our visual analytics system should work with traditional decision tree models so that it can be evaluated during the user study phase.

1.3 Contribution

ObliqueTreeVis is an interactive visualization system designed to assist users in interpreting oblique decision trees from various angles. To achieve a good balance between interpretability and model size, we only choose to focus on oblique trees with bivariate splits. This decision also enables us to use proven two-dimensional visualization techniques. First, we make an oblique tree view to show how the tree model is put together as a whole and to show how the oblique decision tree works. We give a summary view and a detailed view of the decision nodes. This lets users explore and understand how these nodes decide how to split up the data. We make a path summary view in each leaf node so that feature-relevance explanations of the prediction results can help us understand how these paths are split up. In addition, we build a feature view to show the distribution of different features. This view shows the user how much each feature contributes to the features of different decision paths. This helps the user figure out the relationship between the features and the results of the prediction. In addition, we present a projection view to show the data similarity of the original data set. This component also allows users to explore interactively across views. We also let users change the oblique decision tree system by using the control panel view. This component enables the user to select different data sets and switch between models. Lastly, we use a user study and a use case to show that users of a visual analytics system that uses an oblique decision tree as a model can understand the model faster and make more accurate predictions about classification than users of a visual analytics system that uses a traditional decision tree as a model. With the same use study and another use case, we show that ObliqueTreeVis can be used to understand oblique decision trees.

1.4 Organization of the Report

The remainder of the thesis is organized as follows: In Chapter 2, we provide preliminary knowledge about the problem domain. In Chapter 3, we analyze research problems in detail and sort out the analytical tasks. Next, we present related work for the visualization of oblique decision trees in Chapter 4. Chapter 5 shows the processing of the data. In Chapter 6, we introduce the design of our visualization. In Chapter 7, we present the evaluation of our solution. And finally, in Chapter 8, we conclude our work and discuss possible directions for future research.

Chapter 2

Preliminaries

In this chapter, the basic background concepts and models mentioned in this thesis are introduced and explained. In Section 2.1, we begin by quickly introducing the classification problem as an example of supervised learning, as well as a few common classification problems. To explain the scope of this study, Section 2.2 discusses decision tree concepts and illustrates the process of tree development. The content of the introduction to decision trees and oblique decision trees is largely inspired by Truong [32]. In addition, we present the computation procedure of a local interpretation method feature contribution used for decision trees.

2.1 Classification

Given a feature space $\mathbf{X}$ and a response space $\mathbf{C} = \{1, ..., k\}$ with k classes, *classification* is a task that requires the use of machine learning algorithms that learn how to assign a class label $C_i \in \mathbf{C}$ to observation $\mathbf{x}^i \in \mathbf{X}$ from the problem domain where $i \in \{1, ..., N\}$. There are many different types of classification tasks, including:

Binary Classification: contains two class labels;

Multi-Class Classification: contains more than two class labels;

Multi-Label Classification: contains many class labels, one or more of which may be predicted for each case.

Due to the fact that the classification algorithms used for binary or multi-level classification cannot be directly used for multi-label classification and the focus of this study is the visual exploration of oblique decision trees, only classification algorithms for binary or multi-level classification are considered.

A *classifier* is a function f that sorts unlabeled data into labeled classes, or categories of information, *i.e.*, $\hat{C}_i = f(\mathbf{x}^i)$. Practically, people use *probabilistic classifiers* to generalize the notion of *classifiers*: instead of functions, they are conditional distributions $\Pr(C_i|\mathbf{x}^i, \mathbf{T})$ over all possible classes in $\mathbf{C}$. $\mathbf{T}$ denotes the training set, which is $\mathbf{T} = \{\left(X_1^i, ..., X_p^i, C_i\right)_{i=1}^N\} = \left(\mathbf{x}^i, C_i\right)_{i=1}^N$. Therefore, classification can be done through finding the class having the highest probability, as Equation 2.1 shows to us.

$$\hat{C}_i = \arg\max_{\mathbf{C}} \Pr(C_i|\mathbf{x}^i, \mathbf{T}) \tag{2.1}$$

2.2 Tree-structured Classification

Decision trees are used a lot because they are easy to understand. They are one of the most popular machine learning models. It can be traced back to a tree-structured classification method as early as 1959. A British researcher, William Belson, proposed a tree-like classification method for matching population samples [4].

> In this article, Dr. Belson describes a technique for matching population samples. This depends on the combination of empirically developed predictors to give the best available predictive, or matching, composite. The underlying principle is quite distinct from that inherent in the multiple correlation method.

Before decision trees were formally proposed, tree-structured classifiers were a natural way to interpret stratified populations and have long been used by practitioners in a variety of fields. Although creating such classifiers often requires a great deal of thought, researchers have sought ways to automate this process. It was not until 1984 when Breiman *et al*. [8] formally proposed one of the famous decision tree algorithms, CART. In their understanding of the purpose for which decision trees were created [8, p.7], "An important criterion for a good classification procedure is that it not only produces accurate classifiers (within the limits of the data) but that it also provides insight and understanding into the predictive structure of the data.", we can also get a glimpse of how important interpretability is for decision trees.

As an interpretable classifier, decision trees should develop interpretable models with an easy-to-understand structure. Therefore, in the next sections, we use the CART [8] as an example to introduce the process of decision tree construction in detail.

2.2.1 Tree Construction

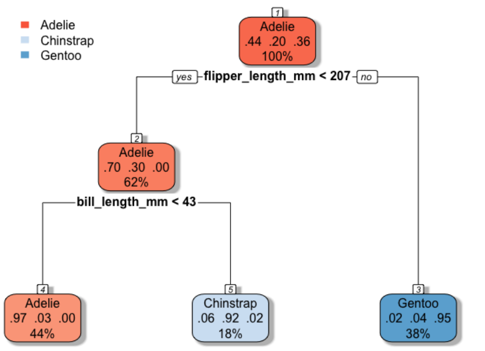
A decision tree is grown as a tree containing a root node, several internal nodes and several leaf nodes. The leaf nodes correspond to decision outcomes at the bottom[1] of a tree. Each other node corresponds to a test; each node contains observations that are divided into sub-nodes based on the test results; the root node contains the entire dataset. The path from the root node to each leaf node corresponds to a sequence of tests. An example of a typical tree is given in Figure 2.1. Two tests divide the training set observations into two groups that are well separated from each other.

In order to produce "purer" child nodes at each stage of the tree growth, CART performs the best partitioning of the instances for each class. This was achieved by applying a so-called *impurity measure* that can quantify the "purity" of each node. Common measures are: information gain, gain ratio, gini index, etc. However, it has long been recognized that it is difficult to create a decision tree that is both accurate and understandable.

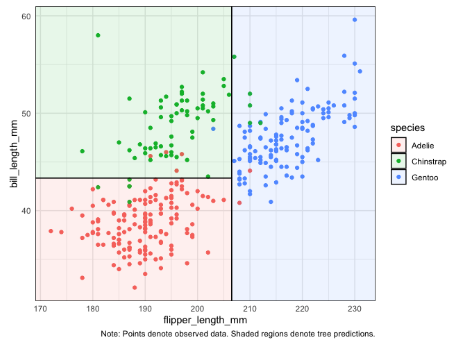
The most severe problem of decision trees with easy-to-understand structures is their scalability. A decision tree classifier uses internal nodes and branches to show how rules are combined to decide how to classify a data instance. Thus, humans can trace a specific classification from a leaf to the root to understand the final prediction. to understand the predictions of the classifier. As the number of rules or nonlinearity rises, so does the performance of decision trees. Although they are simple to learn and follow at first glance, it becomes impossible to understand the classifier as a whole when the number of rule nodes get very large. An example is shown in Figure 2.2.

Therefore, the researchers tried to use oblique splitting to generate decision trees in order to reduce the size of the generated trees in order to improve the tree interpretability.

[1]The roots of the tree are at the top of the viewport and grow downward.



(a) A fitted decision tree using axis-parallel splits.



(b) The corresponding decision boundaries in the feature space specified by the tree.

Figure 2.1: The above tree is grown from Palmer Penguin Data. The decision tree only considers two continuous attributes (flipper_length_mm, bill_length_mm). The dataset has three classes ($k = 3$). This tree contains two binary tests and generates three leaf nodes. Since there are only two attributes involved in the classification problem, it is possible to view the decision boundaries defined by this tree. Due to the use of axis-parallel splits, the feature space is partitioned into rectangular blocks.

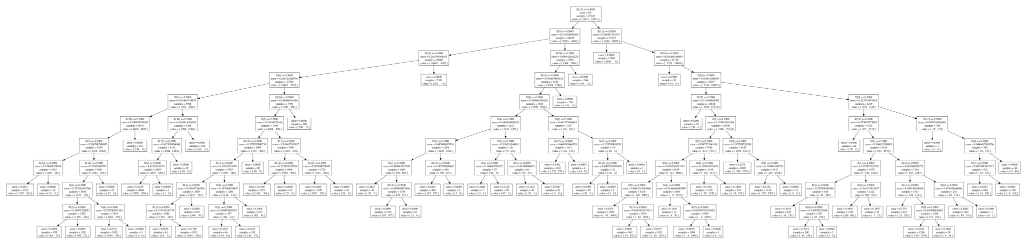


Figure 2.2: A decision tree has too many nodes and its meaning is difficult to understand.

2.2.2 Oblique Decision Trees

In Chapter 1 we have argued that oblique splits can partition the training data set better, so the oblique tree is naturally smaller. However, the application of oblique splits introduces a more difficult splitting function to understand. To introduce the properties of the oblique decision tree, I use the classification of oblique decision trees by Truong [32]. Where there are q continuous attributes in the data set, different oblique splits are based on:

Full Oblique Split: all q attributes;

Concise Oblique Split: fewer than q attributes.

Concise oblique splits are more interpretable than full oblique splits [6]. For example, $X_2 > 0.75X_3$ is more interpretable than $X_1 + 0.5X_2 - 0.1X_3 + X_4 < 0$. Although a decision tree with full oblique split can better partition observations, full oblique partitioning is less interpretable than a concise oblique tree. This stems from the fact that each oblique split of an oblique decision tree contains information about decision boundaries that is difficult for people to understand. Such a problem becomes more severe as the number of features involved in the oblique split increases. Meanwhile, researchers have demonstrated that decision trees can achieve sufficiently accurate results and generate modest tree sizes when the number of attributes involved in a single decision node is not greater than two [6, 7]. For our study, it is reasonable to use decision trees using concise oblique split as a starting point to improve the interpretability of oblique decision trees through visualization techniques. Next, we consider the problem of finding oblique splits.

We have argued that decision trees applying concise oblique splits are better in terms of interpretability than those using full oblique splits. We decided to focus our study on the case when $q = 2$, considering that it still maintains a decent predictive performance.

$$a_m X_m + a_n X_n \leq T \tag{2.2}$$

When using oblique splits, ideally, the best oblique split is used at each stage of generating the decision tree, which means that each potential oblique split needs to be considered. Each node includes a test according to Equation 2.2. To find the optimal oblique split, most known algorithms employ a deterministic hill climbing algorithm [23, 8, 13]. Among them, CART with linear combination (CART-LC) proposed by Breiman *et al*. [8] is the easiest to implement. Considering the time constraint, this study decides to implement the oblique decision tree induction algorithm with $q = 2$ based on CART-LC.

This procedure is initiated by assigning the parameters values that correspond to the univariate split for attribute X_m. For instance, if the best univariate split for X_1 is $X_1 \leq 9$, then one should combine X_1 and X_2 with parameters $a_m = 1$ and $a_n = 0$, and $T = 9$ as the start point of the algorithm.

Considering that the variables included in the test contain ordering information, each decision node ideally needs to consider $q(q-1)$ cases as shown in Algorithm 1.

Algorithm 1 Pseudocode for Bivariate split to find the optimal split

```
1. find best univariate split $X_m \leq T$
2. find best bivariate split:
   for $m = 1$ to $q$ do
       for $n = 1$ to $q$ and $n \neq m$ do
           find $a_m X_m + a_n X_n \leq T$
       end for
   end for
```

The method of finding the optimal split in the univariate split case has been described in detail in Section 2.2.1, and next we discuss in detail how to find the optimal split in the oblique split case.

The purpose of this procedure is to optimize a given split sp $= a_m X_m + a_n X_n \leq T$ by iteratively evaluating splits of the form:

$$\text{sp} - \delta(X_m + \gamma) \leq T. \tag{2.3}$$

For each $m \in \{1, \ldots, q\}$, the optimal δ may be determined through calculating

$$u^j = \frac{\text{sp}^j - T}{X_m^j + \gamma}. \tag{2.4}$$

For $j = 1, \ldots, N$, take γ as a constant. It is recommended that this technique be executed for three distinct γ values, namely $-0.25, 0, 0.25$. Then, δ and γ are derived from the best of these three scenarios and utilized to update the current split sp to a new split sp′ as follows:

$$\text{sp}' = a'_m X_m + a_n X_n \tag{2.5}$$

where a'_m becomes $(a_m - \delta)$ and $T' = T + \delta\gamma$.

This process is repeated for the remaining features $X2, \ldots, Xq$. When the cycle of updating the variable coefficients concludes, we focus our attention on this threshold T, leaving all coefficients fixed and modifying just the threshold T in order to get the best split. We cease cycling when the decrease in impurity from one cycle to the next falls below a predetermined minimal threshold ϵ. This approach finds a global minimum for the impurity of the considered linear splits. However, the algorithm might get locked into a local minimum.

2.3 Feature Contribution

By examining which features are important, we may infer that the model made judgments based on these features, thus explaining the decision-making process. Feature contribution metrics have been made so that features can be compared and ranked in a useful way. These metrics give a feature a single score based on how much it contributed to the outcome. By determining local feature contributions, we may identify how the features in each decision path impact the outcomes.

Palczewska et al. [25] proposed a robust calculation of the local feature contribution, either for discrete or continuous features, in a decision tree model. To describe her method in detail, first understand the definition of the variable Y^n_{mean}. It is the vector of fractions of the training instances in a decision node n belonging to class $C_k, k \in \{1, 2, ..., K\}$. w_m and w_n denote normalized absolute coefficients of a_m and a_n in Equation 2.2. This gives us:

$$w_m = \frac{|a_m|}{|a_m| + |a_n|}; w_n = \frac{|a_n|}{|a_m| + |a_n|}. \tag{2.6}$$

The calculation of local feature contribution is based on the concept of *local increments*. A local increment is about the change in the distribution of a particular feature between a parent node p and a child node c. It is defined as follows:

$$LI^p_f = \begin{cases} w^p_f(Y^c_{\text{mean}} - Y^p_{\text{mean}}), & \text{if the split in the parent is performed over the feature } f, \\ \underbrace{(0, \ldots, 0)}_{K \text{ times}}, & \text{otherwise.} \end{cases} \tag{2.7}$$

where the difference is computed coordinate-wise and w^p_f is the normalized absolute coefficient of feature f in decision p. Then, a series of local incremental accumulations on a decision path becomes a local feature contribution instance under the corresponding class. The contribution FCi^f of a feature f in a tree for an instance i is equal to the total of LIf across all nodes along the route of i from the root node to a leaf node.

$$FC^f_i = \sum_{p \in DP_i} LI^p_f, \tag{2.8}$$

where DP_i is the set of decision nodes involved in the decision path for instance i. Therefore, the feature contributions vecetor FC_i for an instance i is a 2D vector consisting of contributions FC^f_i of all features f. Finally, Palczewska *et al*. [25] did an estimation of local feature contribution under the corresponding class over all the FC^f_i. The median in statistics is a measure of the expected value that is less impacted by outliers than the sample mean. This attribute of the median may be used to determine a "typical level" of feature contributions for instances of a certain class. We choose those cases from the training data set (containing of w occurrences) that are appropriately identified as DC. The following are the ordered contributions for feature f to class k:

$$FC^{f,k}_{(1)}, ..., FC^{f,k}_{(q)}, FC^{f,k}_{(q+1)}, ..., FC^{f,k}_{(w)} \tag{2.9}$$

One may independently determine the median contribution of each feature for each class.

$$FC^{f,k} = \begin{cases} \frac{FC^{f,k}_{(q)} + FC^{f,k}_{(q+1)}}{2}, w \text{ is even} \\ FC^{f,k}_{(q+1)}, w \text{ is odd} \end{cases}, \tag{2.10}$$

where $FC^{f,k}$ is the local feature contribution of feature f for class k.

Chapter 3

Task Analysis

In this chapter, we outline the various user tasks for our system. These will serve as design criteria for the visualization.

In this study, our approach targets many potential domain experts in various fields influenced by emerging machine learning techniques. As machine learning tools like tree-structured models are used increasingly in these fields, experts may not know much about machine learning algorithms. However, they may want to or be required to use them to help them make decisions at work.

The primary goal of our target users is to fully understand the behaviors of an oblique decision tree. As discussed in Chapter 1, we explore the idea of incorporating an extra explanatory interface between humans and the oblique decision tree to provide interpretability. Oblique decision trees as tree classifiers to solve classification problems can be viewed as a machine that has extracted the information in the data set and abstracted the information as different partition processes. Therefore, understanding the underlying structure of the data is a prerequisite for gaining model insights.

T1 Gain insight in the underlying structure of the data. In this task, visualization techniques should be applied to help users understand the data. Since oblique decision trees deal with classification problems, the data involved in visualization are multi-dimensional data sets with categorical classes. To help the target user understand the data, we can summarize the statistical features of the data set and provide users with a statistical understanding. Furthermore, visualization techniques can be used to present the results of dimensionality reduction to help users understand how the data is distributed in the input space.

Once the user understands the distribution and characteristics of the data set, they can formally begin to understand how the model behaves. The next task is to present the structure of the tree to the user from a global perspective.

T2 Understand the distribution of classes across decision rules. As a rule-based model, any decision tree can be thought of as a collection of decision rules (*i.e.* **IF-THEN** statements). We propose to use the node-link diagram representation to reduce the difficulty of user understanding. With the help of visualization techniques, each decision rule is presented in the form of a decision path. It is represented by multiple decision nodes strung together in a link. In this way, users can naturally read the entire tree, following different data classes to explore decision nodes or decision paths, and find leaf nodes for specific classes of predictions more efficiently.

To understand which decisions the model makes for particular instances, target users may want to inspect individual nodes. For traditional decision trees and other rule-based classifiers such as decision sets, each rule is easy to understand, and the user can read it without extra tools or mathematical knowledge. However, this is not the case for each decision node in an oblique decision tree. In an oblique split, linear combinations of attributes shall be jointly evaluated as rule conditions instead of simple attribute pairs. They are usually considered more difficult to understand without additional tools. Hence, we propose the following user task:

T3 Understand how an oblique split works in each decision node. Understanding the inner workings of each oblique split is equivalent to understanding the decision boundaries presented by that decision node in the corresponding feature space. Since our study limits the oblique decision tree to no more than two features in each oblique split, traditional two dimensional visualization techniques can be used to show how a particular decision node in the oblique decision tree works.

However, understanding the inner workings of each decision node is not enough to understand how the model makes a prediction for a specific instance. To understand why the model makes a prediction, users can follow the decision path from the root node of the model to a leaf node. When users are faced with a classification problem, it is necessary to analyze a decision path or even multiple decision paths. In addition, when domain experts want to understand the prediction for one particular instance, finding the corresponding decision path is also important to improve the understanding of the model by the user.

T4 Locate decision paths that contain decisions on a particular feature or that apply to particular instances of interest. Each decision path is a rule set of multiple oblique splits. Since there are many paths, it may take the target user a long time to search through all the decision paths. Visualization techniques can support the user in finding specific decision paths effectively (such as ones that contain particular features or apply to certain instances). With the help of visualization techniques, users reduce the time required to find decision paths and can greatly improve their understanding of the model.

In addition to improving the efficiency of users in finding target decision paths to solve classification problems, users are often faced with the challenge of understanding the decision-making process based on a particular classification result. Classification results are presented as leaf nodes in a tree, which can be easily located by the user from the oblique decision tree. Immediately after locating the leaf nodes, the corresponding decision paths can be found by tracing the path from the leaf node back to the root node. To help users understand which features act in these decision paths and build trust in predictions made by the model, we propose the following user task:

T5 Identify which features contribute to a particular decision path and the decision nodes. The decision-making process of the model involves multiple features. To explain such a decision process, the user needs to understand which features are relevant to the output and how they contribute to such an outcome. Therefore, we need to provide the user with feature-relevance explanations for different decision paths. In addition to this, since at most two features may be involved in each oblique split, we also need to present the contribution of the involved features to a single decision node.

In this chapter, we discuss the tasks that our study needs to accomplish to improve the interpretability of oblique decision trees. We start from the understanding of the data by the user. Then explain that the user needs to understand the working mechanism of decision nodes inside the oblique decision tree to analyze one or more decision paths. In addition to this, we emphasize the task of helping the user locate the decision paths involved in certain features. This helps to improve the efficiency of the user in analyzing the model. It is also important to help users explain the contribution of features in some specific decision-making processes.

Chapter 4

Related Work

In this chapter, we discuss related work in the areas of visualization that is relevant to our work.

4.1 Visualization for Data Understanding

Extensive studies have been conducted on the visualization of exploratory data analysis. The data used in classification problems is multidimensional and has classes. This section discusses relevant visualization studies used for statistical analysis and dimension reduction.

The box plot is one of the most commonly used methods for presenting statistical properties of data. Potter *et al*. [27] present a survey on box plot design. As seen in Figure 4.1, (a) depicts the key features of a box plot. Box plots that include additional information, such as confidence intervals in (e) and distribution densities in (f-i), provide a more thorough overview than basic box plots. Nonetheless, the data sets used by categorization algorithms have gotten more complicated and dimensionally dense. Exploring hundreds of box plots does not always provide information about the data set.

In helping users to understand data, dimensionality reduction is often used to understand the distribution of data in the input space. Therefore, projections, including principle component analysis (PCA) [40], multidimensional scaling (MDS) [15], t-distributed stochastic neighborhood embedding (t-SNE) [34], and uniform manifold approximation and projection (UMAP) [21], are the most commonly used dimensionality reduction techniques at this stage. The role of visualization is to present and enhance the results of dimensionality reduction. A well-known example is the Embedding Projector [30], which embeds human-interpretable classes (e.g., images with categorical colors) into scatterplots implemented by dimensionality reduction. By browsing the projection map, users can better understand the relationship between different instances and identify clusters and outliers, as shown in Figure 4.2.

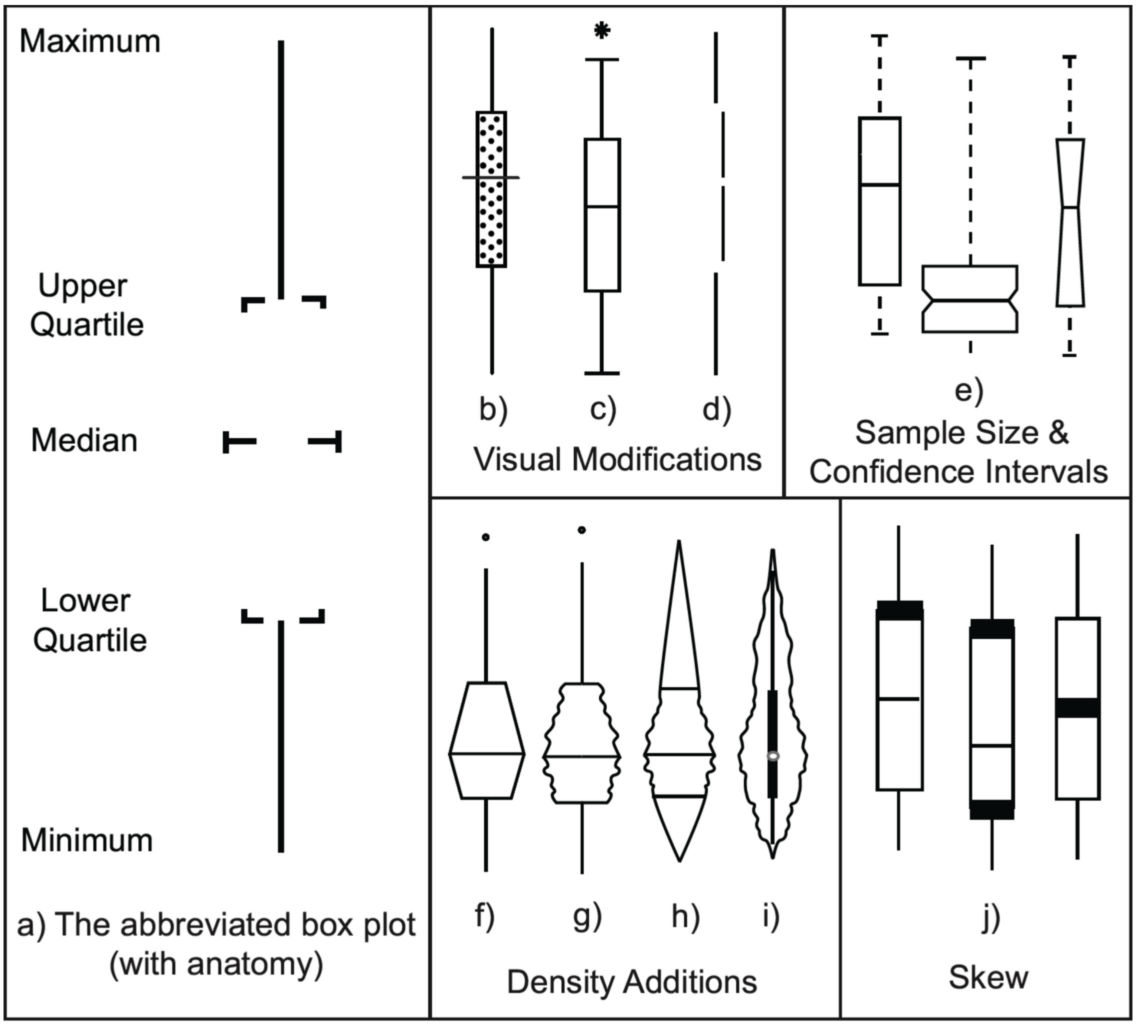
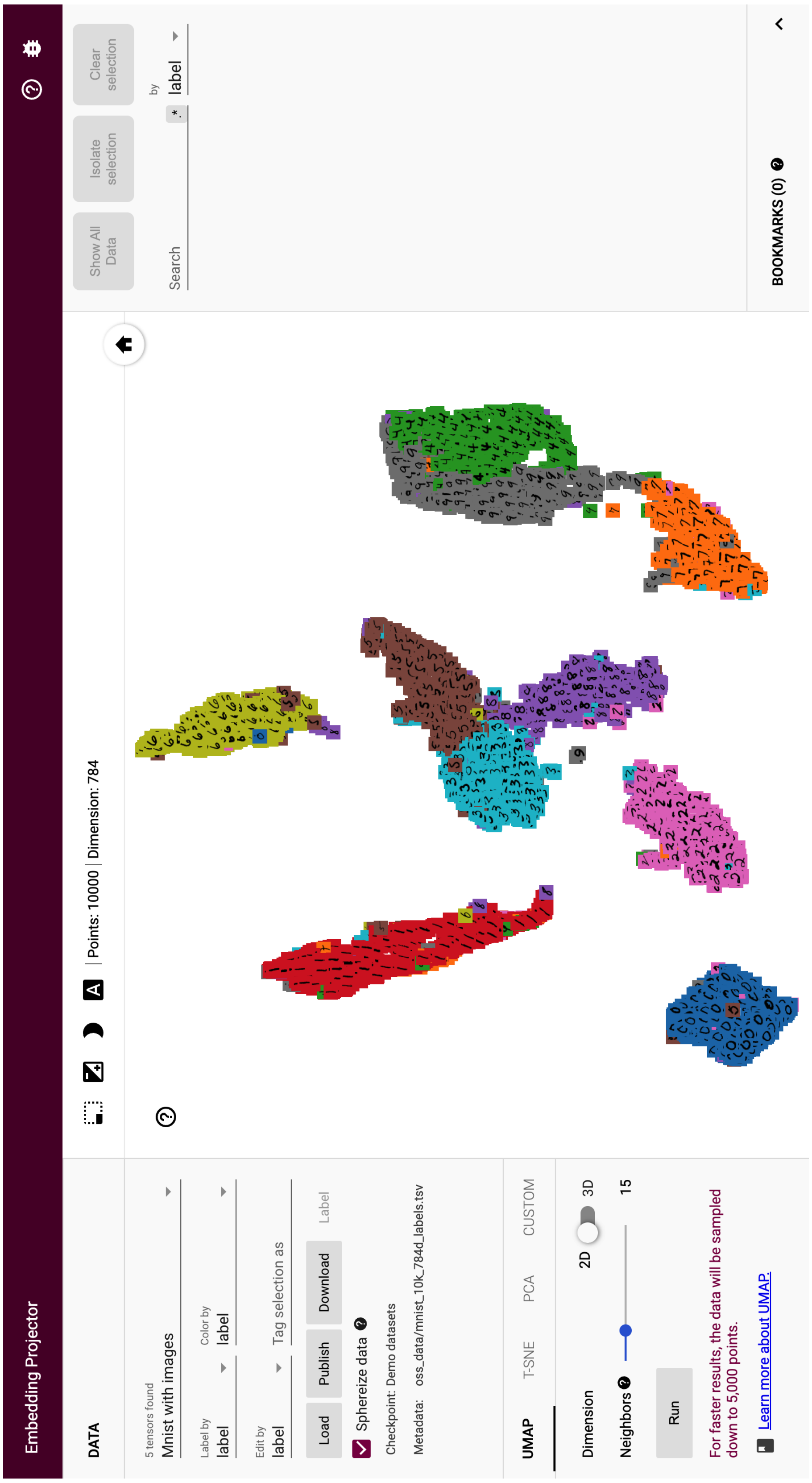


Figure 4.1: Variations on the box plot. a) Abbreviated box plot. b) Range plot. c) Box plot. d) Interquartile plot. e) Variable width and notched box plots expressing sample sizes and confidence levels. f) Hist plot. g) Vase plot. h) Boxpercentile plot. i) Violin plot. j) Skew and modality plots. [27]



5 tensors found
Mnist with images

Label by
label

Color by
label

Edit by
label

Tag selection as
Label

Sphereize data

Checkpoint: Demo datasets

Metadata: oss_data/mnist_10k_784d_labels.tsv

Dimension 2D 3D

Neighbors 15

For faster results, the data will be sampled down to 5,000 points.

Learn more about UMAP.

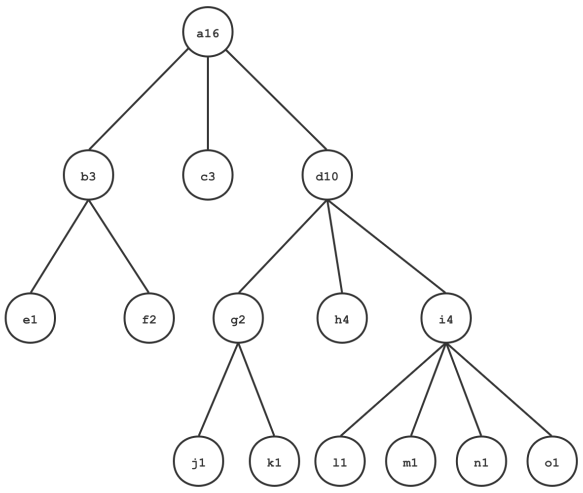
| Points: 10000 | Dimension: 784

.* by label

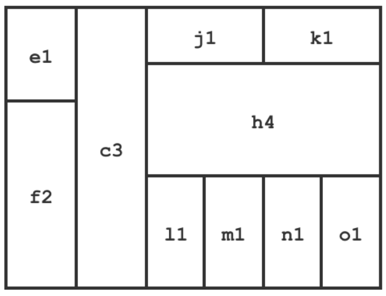
Figure 4.2: A UMAP projection of Mnist data with images.

4.2 Visualization for Model Understanding

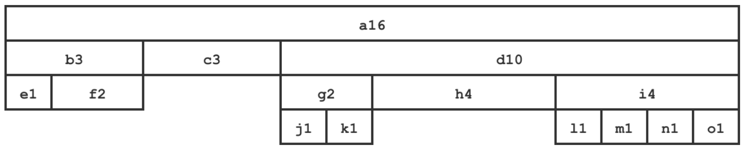
One of our tasks is to help researchers better understand the characteristics and working mechanisms of oblique decision trees. For classifiers with a special hierarchical tree-like structure like oblique decision trees, they use internal nodes and branches to represent their classification inference as the connection points of rules. Humans can trace specific classifications from a leaf to the root to understand the predictions. Thus, most of the related work for understanding tree-structured classifiers focuses on visualizing specific components of a classifier. There are mainly three aspects: tree-based visualization (see Section 4.2.1); visualization of feature space splits in decision nodes (see Section 4.2.2); and visualization of decision tree path interpretation (see Section 4.2.3).



(a) Node-link diagram



(b) Treemap



(c) Icicle plot

Figure 4.3: Common tree representations, each showing the same tree in a different way.

4.2.1 Tree-based Visualization

One can visualize decision trees using different visualization techniques, mainly including node-link diagram [24, 38, 41], icicle plot [20, 37], and treemap [22]. Node-link diagrams are extensively used to visualize decision trees because they naturally depict tree structures, as shown in Figure 4.3a. One problem with node-linked diagrams is that they do not work well for large data sets. In contrast, icicle plots [14] is a more space-efficient technique to depict decision trees. As shown in Figure 4.3c, nodes are represented by several stacked bars, and the length of each bar is related to the volume of data that travels through it. Ankerst *et al*. [2] employ a pixel-based approach to encode the class distribution of each node. They use pixels to represent observations and pixel colors to represent data classes, as shown in

Figure 4.4. This pixel-based design enables users to determine if each node partitions data correctly using distinct classes.

Besides, the treemap technique is another visualization scheme used for decision trees. As shown in Figure 4.3b, they are space efficient, scaling up to thousands of nodes. Muhlbacher *et al*. [22] take advantage of the treemap pixel-based architecture to assist users in estimating the complexity and performance of a decision tree in Figure 4.5. But at the cost of making the different levels within the tree harder to perceive and distinguish. Because of this flaw, it is not only hard to see the internal decision nodes of the oblique decision tree, but it is also hard to show each decision path clearly.

Overall, considering the hierarchical structure of the oblique decision tree and its own small tree size, the node-link diagram is one of the visualization techniques that can be borrowed. On the one hand, it can provide enough space to combine with other potential visualization techniques to present the information of internal decision nodes; on the other hand, based on its own tree structure, it can clearly present each decision path from the root node to the leaf node. Such a property assists the user in understanding how specific predictions are made, which is not possible with the other two options.

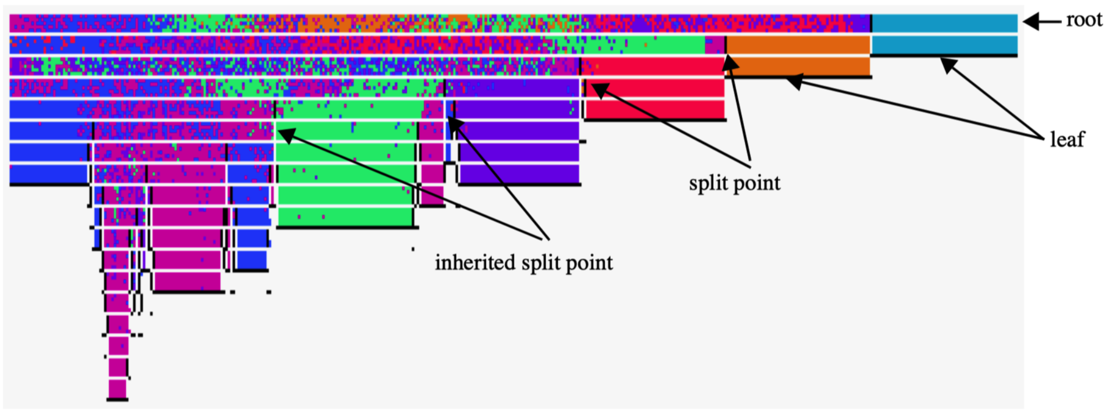


Figure 4.4: Icicle tree visualization combined with pixel-based observations and encoding with colors as data classes proposed by Ankerst *et al*. [2]

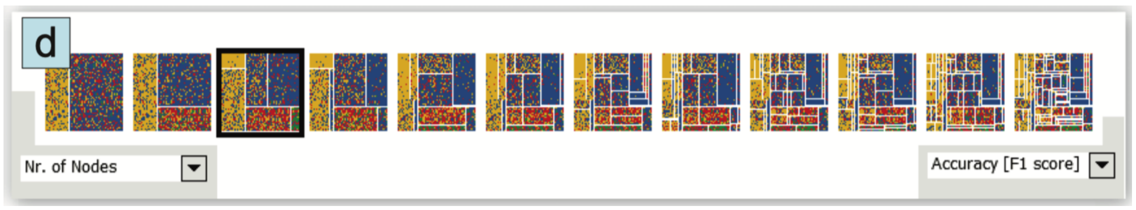
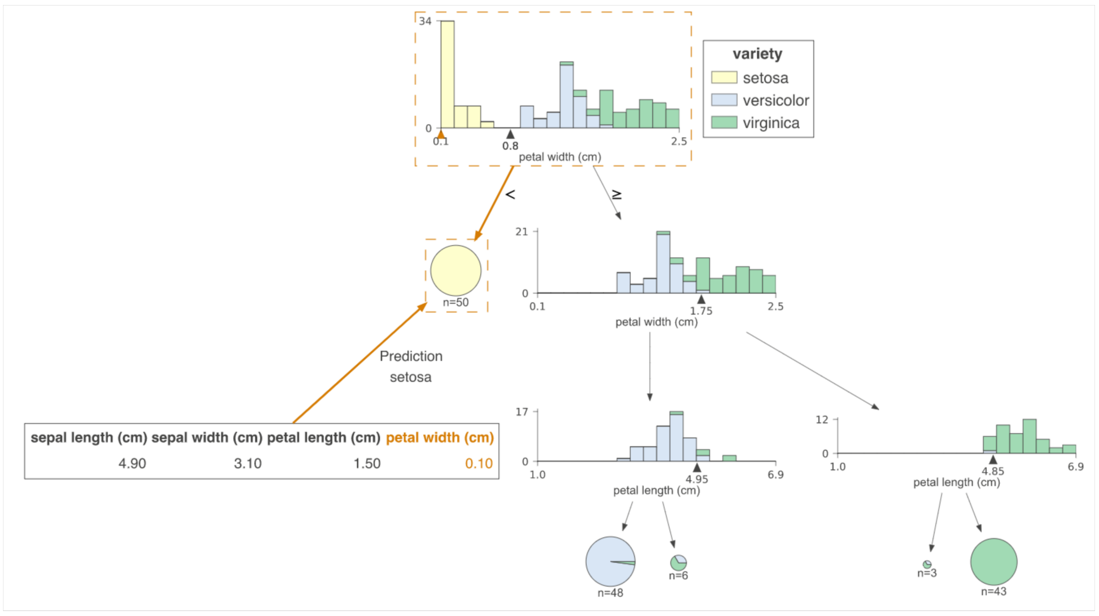


Figure 4.5: Treemap visualization combined with pixel-based observations and encoding with colors as data classes proposed by Muhlbacher *et al*. [22]

4.2.2 Visualizing Feature Space Split in Decision Nodes

Although the node-link diagram is a good choice to show the global tree structure and provide clear decision paths in oblique decision trees, it is difficult for users to understand how internal decision-making works without the help of other tools. Currently, many decision tree visualizations have been developed using node-link diagrams to learn how to visualize the split of feature space for each decision node. Such previous studies have positive implications for our research. We therefore discuss work on feature-target space visualization in this section.



sepal length (cm)	sepal width (cm)	petal length (cm)	petal width (cm)
4.90	3.10	1.50	0.10

Figure 4.6: A typical decision tree visualization with the Iris data proposed by Parr *et al*. [26]

Parr *et al*. came up with dtreeviz, a python library with a node-link diagram layout for decision tree visualization and model interpretation [26]. The decision tree example with the Iris data set is shown in Figure 4.6. Each decision node presents the class distribution of the subset of the point on the one hand, and the position of the split point in the class distribution on the other hand, and thus the class distribution of each of the two subsets divided by the current decision node. The visual encoding for class distribution is a stacked histogram. They used stacked bar colors to represent data classes. This visualization has some nice properties. It not only presents how many classes the current node contains, but then presents the distribution of these classes using a histogram. Often, the count of different classes and their distribution are of interest to the user in addition to what classes are available.

Elzen *et al*. presented an alternative method to visualize how observations in the current decision node are split. When there are more than four or five classes, the stacked histograms are difficult to read. But streamgraph and horizontal class histogram enable us to visualize data sets with more classes. Although the streamgraph does not present the class distribution precisely, the absolute value is displayed inside each color-coded histogram bar. Each color represents one of the data classes. Also, the splitpoint in this case is a draggable interactive part that lets the user see how the split works in different cases while they are interacting with it. Even though building models is not the focus of our research, the streamgraph can be used to help visualize feature space.

Truong [32] uses multiple line segments in the scatterplot to present all the splits in the two-dimensional feature space, as shown in Figure 4.8. With each line segment, the user can clearly see how the oblique split makes decisions in a two-dimensional feature space. All the observations are encoded with textual glyphs, colored corresponding to their respective class in the scatterplot, allowing the user to see the general class distribution and the effect of each split. But the problem is that the user cannot tell from the associated decision boundaries which line segment corresponds to the specific oblique split. On the other

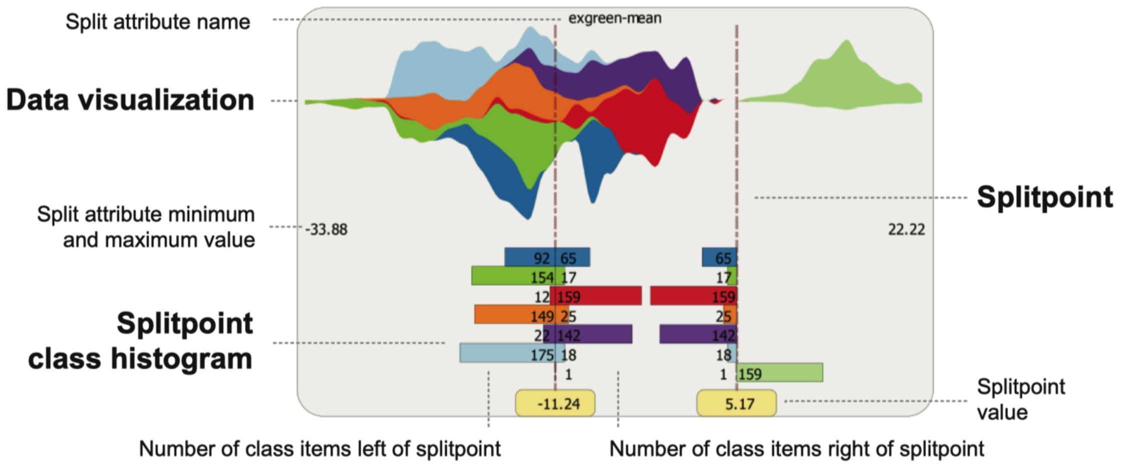

Split attribute name
exgreen-mean
Data visualization
Split attribute minimum and maximum value
-33.88
22.22
Splitpoint
Splitpoint class histogram
-11.24
5.17
Splitpoint value
Number of class items left of splitpoint
Number of class items right of splitpoint

Figure 4.7: A decision node to visualize the split feature space proposed by Elzen *et al*. [33]

hand, the user cannot see the order of these oblique splits in the decision path itself, which makes it harder for the user to understand how the model works. The implications of this example for this study are that the oblique split is presented as a line segment in a scatterplot to help the user understand how the split is worked in a two-dimensional feature space, and then the technique is introduced to each decision node to present each oblique split independently rather than as a whole in a scatterplot.

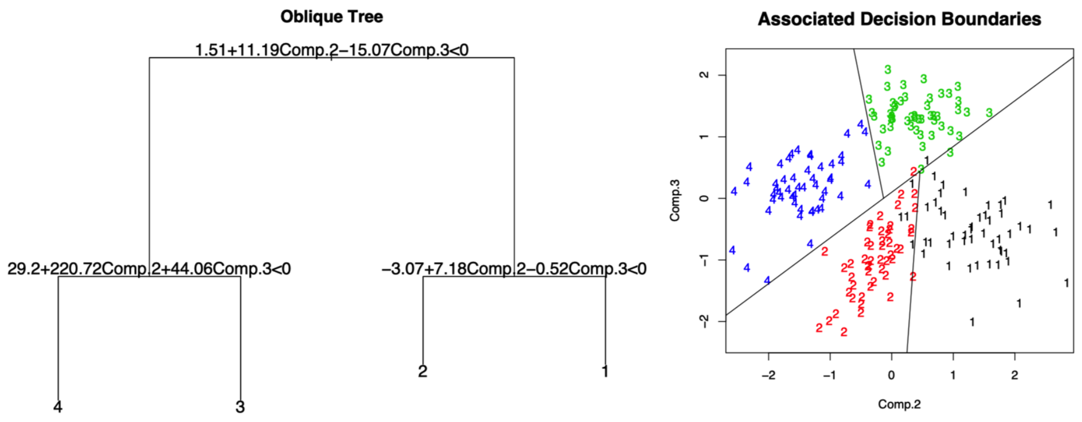


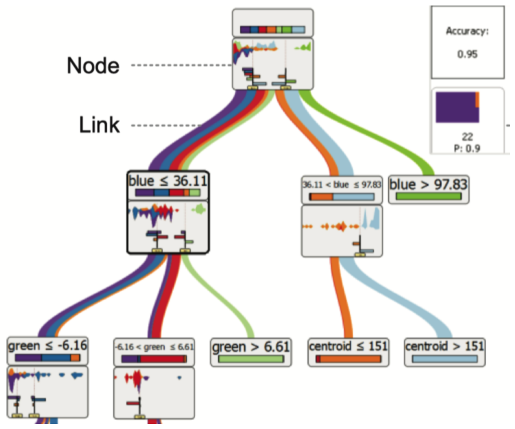
Figure 4.8: An oblique decision tree and associated decision boundaries proposed by Truong [32]

4.2.3 Visualizing Tree Interpretation of Decision Paths

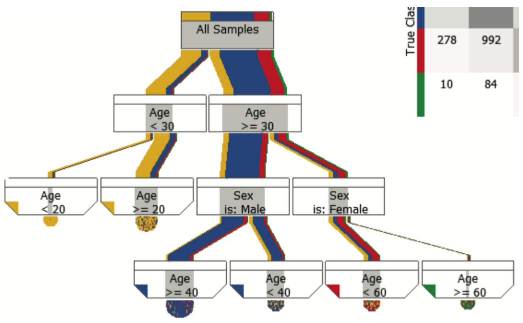
Based on the hierarchical tree structure of the oblique decision tree itself, it can clearly present each decision path from the root node to the leaf nodes. The decision path serves as an important component that helps the user understand how a particular prediction is obtained by this classifier.

As Figure 4.6 shows, Parr *et al*. used some techniques to highlight the decision-making process. They showed an orange arrow in the horizontal feature space for decision nodes along the path to the leaf predictor node. This makes the decision easy to track; if the orange wedge is to the left of the black wedge, go left, else go right. Decision nodes that are part of the decision path are surrounded by boxes with dashed lines, and the edges of their children are thicker and colored orange. Although the decision path here contains only one class, when a decision path contains more than one class, the information about the class flow distribution contained within the decision path is not presented to the user.

Since the node-link diagram is a convenient technique for presenting decision paths in tree-based visualization, many researchers decided to work on the width and color of the links to convey node size and class distribution as shown in Figure 4.9. The width of each link in BaobabView [33] and TreePOD [22] is proportional to the number of observations going from the parent node to the child node. If the inbound link at a node is dense, then this node contains many instances. If the inbound link is very thin, there are few instances present. To display the distribution of classes for a link of two nodes, the broad link is partitioned into bands, with each band's color and width according to the class. In terms of other link details, Baobab has performed better, as seen in Figure 4.9a, where each link is curved to make each decision route easier to follow while displaying node size and class distribution.



(a) Decision tree layout of BaobabView [33]



(b) Decision tree layout of TreePOD [22]

Figure 4.9: Comparison of BaobabView and TreePOD in decision tree layout. Both use the width of the link to convey the node size and the color of the embedded link to characterize the class, but the difference is that the former link has the B´ezier curve feature, while the latter does not.

However, when the oblique decision tree model is deployed, another important challenge is how to get users to trust the model. Using Explainable AI (XAI) techniques, we discuss how to assist users in the local interpretation of specific predictions in the next section.

4.3 Model Explanation

Ribeiro *et al*. [29] proposed a process of explaining specific predictions in LIME. It comprises feature relevance techniques and visual explanation techniques. As Figure 4.10 shows, the local explanation is a small list of symptoms with relative weights; these symptoms either help the prediction (green) or are evidence against the prediction (red). Users usually have a priori knowledge about the application domain, which they can use to accept (trust) or reject the prediction if they understand the reasoning behind the prediction in order to monitor the model. Feature relevance explanation aims to depict the functioning of a classifier by ranking or measuring the contribution each feature has in the prediction output by the classifier to be explained.

Many tree ensembles and multiple classifier systems seek to improve model-specific interpretability through feature relevance explanation [9, 16, 25, 39, 31]. Breiman [9] was the first to analyze the feature contribution within tree ensembles. His approach is based on evaluating Mean Decrease Accuracy (MDA) or Mean Increase Error (MIE) when a variable is randomly permuted in out-of-bag samples. However, this ranking is too general and simplistic and does not provide a local explanation. Palczewska *et al*. [25] proposed a local feature contribution metric. They cleverly use the property of decision trees as hierarchical structures to calculate the local contribution of features in decision paths by using the difference in class distribution between parent and child nodes. The details are discussed in Chapter 2.3.

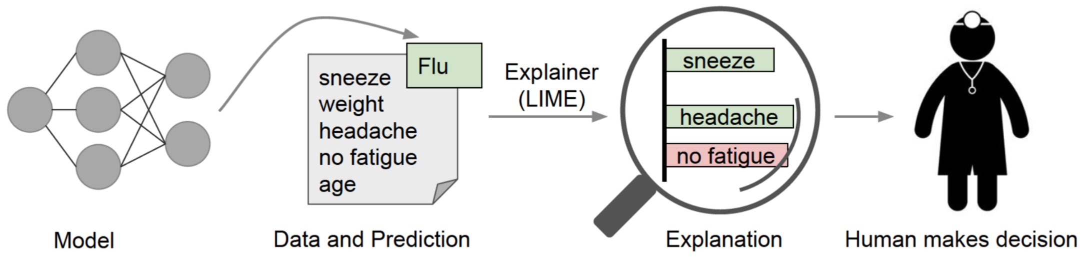


Figure 4.10: Explaining individual predictions. A model predicts that a patient has the flu, and LIME highlights the symptoms in the patient's history that led to the prediction. Sneeze and headache are portrayed as contributing to the "flu" prediction, while "no fatigue" is evidence against it. With these, a doctor can make an informed decision about whether to trust the model's prediction. [29]

The literature we present in this chapter shows that many visualization techniques are suitable for visualizing certain parts of our tasks. We can use box plots to assist users in statistical understanding of the data sets. There are several dimensionaliy reduction techniques to show how the input space is distributed. In the field of tree-based visualization, we see that visualization techniques are mainly based on the presentation of node-link diagrams. As for the visualization of the splitting of the feature space for each decision node, we see that the research focuses on how to present the class distribution of the sub data set after being split and how the split makes the classification decision. For visualizing the tree interpretation of each decision path, people use visualization techniques to highlight a certain decision path in the whole decision tree; they also focus on how to present the class distribution in each decision path. In addition to this, feature relevance explanation provides faithful local explanation for specific predictions. In the next chapter, we present how to combine multiple visualization techniques to accomplish tasks defined in Chapter 3.

Chapter 5

Data Processing

In Section 4.2.1 we explained why we chose a node-link diagram to present the oblique decision tree. Moreover, to accomplish tasks (**T2-T5**) outlined in Chapter 3, we introduce a certain data structure to communicate the relevant information from the Python backend to the Javascript frontend. D3, which is the frontend visualization library for this study, accepts only binary trees as input to generate node-link diagrams. In addition, in order to visualize each node, we need to prepare the required data to be stored in the binary tree in advance. In this chapter, we discuss the original data, the trained oblique decision tree, and explain how we manipulate the results of the trained classifier to generate a binary tree structure data, to effectively communicate the model information from backend to frontend.

5.1 Original Data

The data sets used to train our oblique decision trees are Palmer Penguin [12], Iris [1, 11], Wine [36], and Synthetic data [1]. All of these are multivariate data sets for classification problems. Taking Palmer Penguin data set as an example, it contains 7 different attributes, of which 3 are categorical and 4 are continuous. An example of a row is given in Table 5.1.

Parameter	**Value**
Species	Adelie
Island	Torgersen
Sex	Male
culmen_length_mm	39.1
culmen_depth_mm	18.7
flipper_length_mm	181.0
body_mass_g	3750.0

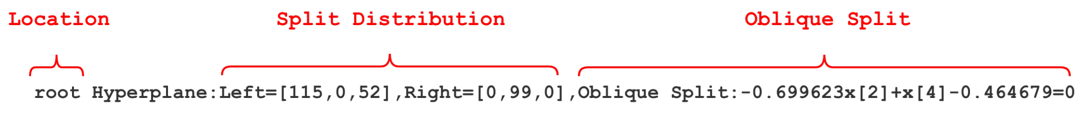
Table 5.1: An example of a penguin observation.

5.2 Trained Oblique Decision Tree

The oblique decision tree is actually composed of a series of rules. They can be represented in textual form as Figure 5.1 shows. A decision rule is unique and represents a decision path. It consists of multiple predicates. A predicate in a decision rule represents a decision node

[1]This synthetic data set is described in detail in Section 7.2.

separately. It contains three kinds of information, location of the current node, split distribution after the current split, and the oblique split. For example, `lrl` denotes the decision path from the root node to the current decision node. From the root node, it goes to the left child node, then proceeds to the right child node, and the immediately following left child node is the location of the current decision node. The split distribution is class distributions of the left and right subsets after the current split. Most importantly, the oblique split is represented in mathematical form.



```
root Hyperplane:Left=[115,0,52],Right=[0,99,0],Oblique Split:-0.699623x[2]+x[4]-0.464679=0
```

Figure 5.1: A decision node example for Palmer Peguins data.

```
0.  root Hyperplane:Left=[115,0,52],Right=[0,99,0],Oblique Split:-0.699623x[2]+x[4]-0.464679=0
1a. l    Hyperplane:Left=[0,0,49],Right=[115,0,3], Oblique Split:-3.413128x[1]+x[5]+0.388552=0
1b. Class 2
2a. Class 3
2b. lr   Hyperplane:Left=[1,0,3],Right=[114,0,0],  Oblique Split:-1.185092x[1]+x[5]+0.296273=0
3a. lrl  Hyperplane:Left=[0,0,3],Right=[1,0,0],    Oblique Split:x[2]-0.145013=0
3b. Class 1
4a. Class 3
4b. Class 1
```

Figure 5.2: An example of decision path consisting of a set of decision nodes for Palmer Penguins data.

A trained oblique decision tree consists of decision nodes and predicted outcomes in leaf nodes, as shown in Figure 5.2. A series of decision rules are strung together based on the corresponding location information to derive a specific predicted classification result. To achieve task **T2** formulated in Chapter 3, such information can be visualized in the form of a node-link diagram. In addition to this, to accomplish task **T3**, we want to generate data visualization in each decision node. To achieve this, we generate specific data for each decision node. In the next section, we further introduce data manipulation to assist the visualization in decision nodes.

5.3 Data Manipulation

To assist users in understanding the inner workings of a single oblique split (task **T3**), we visualize the decision boundary in the corresponding feature space reflected by the oblique split. We record which observations are involved and how they are divided into children nodes. In addition, considering that task **T5** is to help users explain how features contribute to different decision paths, we compute the feature contribution per class and store it in each leaf node. It is worth noting that the class distribution in each node is a prerequisite to obtain feature contribution values, as described in the feature contribution method presented in Chapter 2.

To obtain the data structure shown in Figure 5.3, data manipulation consists of the following three steps:

1. **Build a binary tree.** Based on the location information in each decision rule shown in Figure 5.1, a corresponding binary tree is generated, and when traversing each node,

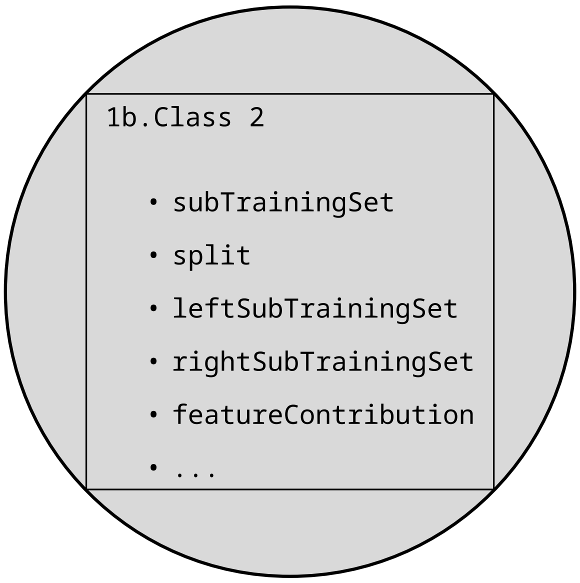


Figure 5.3: An example of target data structure for a leaf node.

the oblique split information of that node is inserted, on the one hand, to facilitate the subsequent re-classifcation of the data set.

2. **Reclassify the data set.** Each point in the data set starts from the root node of the binary tree generated in the previous step and is determined to be divided into left or right child nodes by the current oblique split based on its feature value. At the same time, the data point is recorded to the current decision node. The class corresponding to the data point is also recorded and used to count the class distribution of the decision node.

3. **Compute feature contribution values.** Finally, based on the class distribution obtained in the previous step, the feature contribution values per class are stored in the corresponding leaf nodes with the help of the robust calculation of the local feature contribution proposed by Palczewska *et al*. [25].

Once we have the data needed for the visualization, we can proceed to the visual design phase, which is described in Chapter 6.

Chapter 6

ObliqueTreeVis Visual Design

In this chapter we present visualization techniques to facilitate the user tasks mentioned in Chapter 3. First we describe each of the ObliqueTreeVis [19] components and what tasks they each accomplish. Then we describe, in detail, the visual encoding of each individual component and justify our choices.

6.1 ObliqueTreeVis Overview

This section presents an overview of ObliqueTreeVis in which we relate the views to the tasks we defined in Chapter 3.

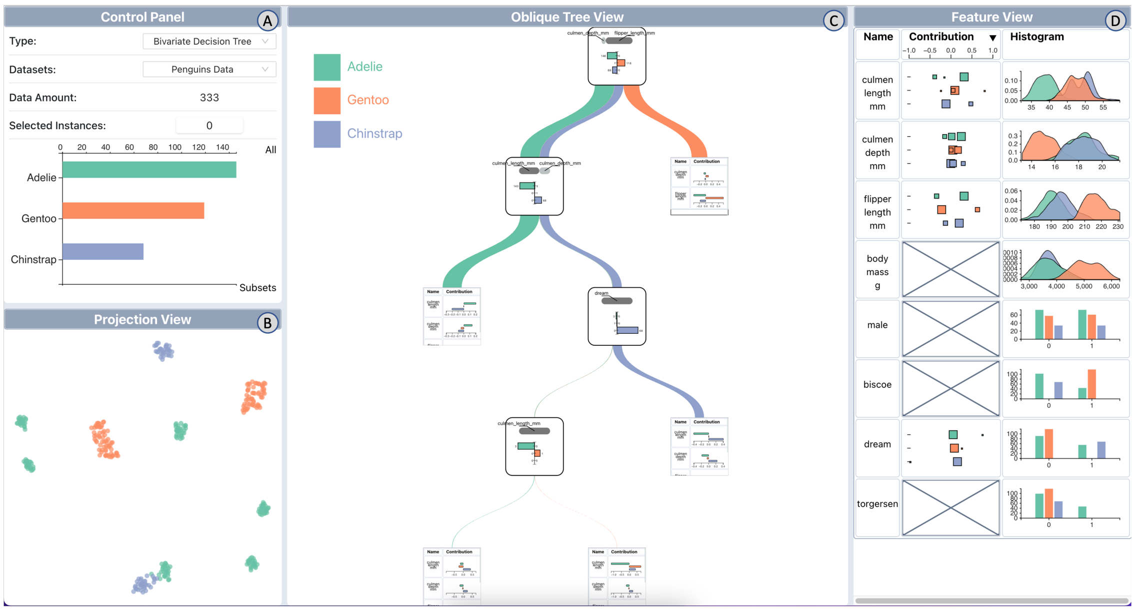


Figure 6.1: The interface of ObliqueTreeVis. (A) Control panel component; (B) Projection view component; (C) Oblique tree view component; (D) Feature view component.

The interface of ObliqueTreeVis is shown in Figure 6.1. To interpret the oblique decision tree from multiple aspects, we have chosen to place all components on one page. All four components above are implemented with dynamic linkage to help users understand the oblique decision tree.

Before we elaborate on the visual design of the components, we first recall tasks defined in Chapter 3:

T1 Gain insight in the underlying structure of the data.

T2 Understand the distribution of classes across decision rules.

T3 Understand how an oblique split works in each decision node.

T4 Locate decision paths involved in particular features and certain instances of interest.

T5 Identify which features contribute to a particular decision path and the decision nodes.

These four tasks can be summarized as exploration from three aspects: *Oblique Decision Tree*, *Explanation*, and *Feature*. The corresponding three kinds of data, binary tree (*Oblique Decision Tree*), feature contribution values (*Explanation*), and feature space (*Feature*), are preprocessed inputs to our visual analytics system. Because all three aspects relate to our tasks, they guide the visual design of ObliqueTreeVis. The remaining three components address one to two of the above three aspects, except for *component A* which provides a summary of the classification tasks and enables the user to select models for further analysis. Specifically, the projection view (Figure 6.1B) at the bottom left as *component B* presents the user with how the data set is distributed in the input space, addressing *Feature*. The oblique tree view (Figure 6.1C) in the middle serves as *component C* to present the user with a tree model in the form of a node-link diagram, addressing *Oblique Decision Tree*. Furthermore, *component C* visualizes feature-relevance *Explanation* in each leaf node. The feature view (Figure 6.1D) on the far right contains not only information about feature contribution but also the distribution of features per class, addressing *Explanation* and *Feature*.

Four tasks can be accomplished with the help of ObliqueTreeVis by practicing different ways to interact with each of these four components. In the next few sections, we present the visual design of each component of ObliqueTreeVis. We describe the visual encodings and interactions of the individual components in detail and justify our choices.

6.2 Oblique Tree View

The oblique tree view (see Figure 6.1C) concerns the visualization of the tree structure to assist users in understanding the working mechanisms of oblique splits and interpreting the decision-making process (**T2**,**T3**, and **T5**). There are various ways to visualize the tree structure. Barlow *et al*. [3] examined node-link diagrams, treemaps, tree-rings, and icicle plots with regards to their ease of identifying tree topology, ease of identifying node links, ease of identifying leaf sizes, and user preference. The icicle plot and node-link diagram performed the best in this investigation. Considering that we want to allow the user to understand the mechanisms operating inside each decision node as they read the tree (task **T3**), the tree and the data visualization shall be tightly integrated. We need to show the data visualization on the decision nodes, which is hard to do with icicle plots because the decision nodes further up the tree get smaller and smaller until they can no longer show the data visualization in a meaningful way.

Another reason for using node-link diagrams instead of icicle plots is that we get an additional component, namely links. Links can present the user with a distribution of classes along the decision path (task **T2**). The user understands the whole model by reading and understanding the decision boundaries made by each decision node and organizing them into a decision-making process with logical relationships. This requires us to present the distribution of classes along the decision path to the user with the help of visualization tools. It represents the size of the subset that the current node contains. The width of each link is determined by the number of instances that pass from the parent node to the child node. For instance, if the inbound connection of a node is dense, this node includes many instances.

If the inbound connection is thin, then there are a small number of instances. To display the class distribution of the links between two nodes, the broad links are divided into many bands. According to the class, bands are colored and given a comparable scale.

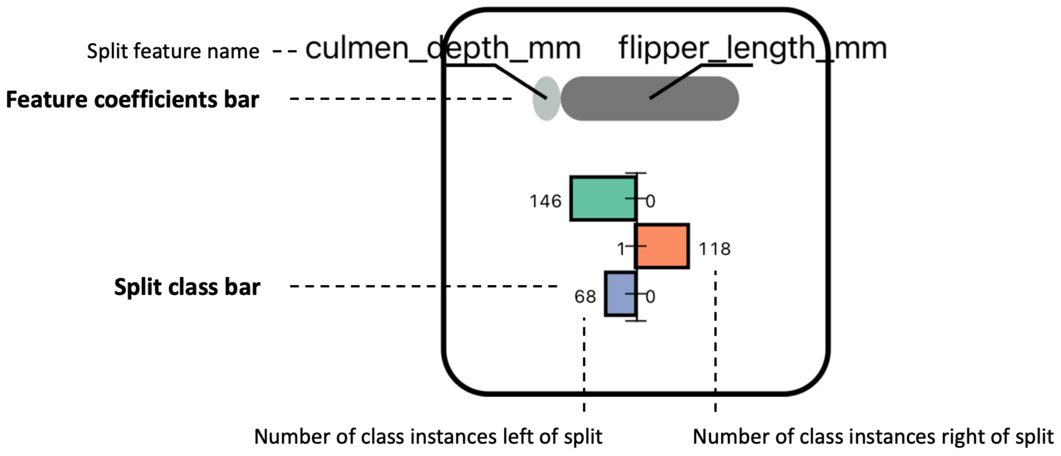


Figure 6.2: Summary view for decision nodes.

The nodes are visualized as rectangles, and the nodes are divided into decision nodes and leaf nodes. The former contains two sub-views for presenting relevant information to solve task **T3**. The latter presents the feature-relevance explanation related to the decision path for solving task **T5**. For decision nodes, they contain a summary view (see Figure 6.2) as well as a detailed view (see Figure 6.3). For leaf nodes, they contain a path summery view (see Figure 6.5).

The summary view provides the user with the weight of the feature coefficients involved in the current oblique split. This weighting presents how each involved feature contributes to the decision boundary. This information is visualized through **feature coefficents bar**. The length of each bar depicts the corresponding feature coefficient weight where the bar with a greater weight is filled with a darker color. **feature coefficents bar** are linked with the feature name column (see Figure 6.1) in the feature view to solve task T3. When a specific feature name is selected in the feature view, the bars in the summary view that are identical to the selected feature are highlighted. This interaction reduces the time spent by the user in locating the decision path involving certain features.

In addition, the summary view also provides the user with the class distribution of the subset after the oblique split, which is visualized in a **split class bar**. The bars of the histogram are colored according to the class. The values are displayed beside the bars and indicate the actual number of items on the left and right sides of the oblique split. With this histogram, the user can know exactly the results of the current oblique split on the class distribution. It helps the user to understand how oblique split works (task **T3**).

With the help of the summary view, the user not only understands which features act on the current decision node and the extent of their contribution to the split, but also the effect of the current decision node on the class distribution of the corresponding subset. However, the understanding of the decision boundaries requires the help of the detailed view.

The detailed view provides users with a detailed oblique split visualization to solve task **T3**. **Scatterplot** encodes two split features using point marks and both vertical and horizontal spatial position. The class is encoded by adding color to each point mark. **Oblique split** is visualized through a solid black line to depict the decision boundary. This visual design reduces the extra mathematical knowledge required by the user to read formulas made up of a combination of features. We enable the user to select points of interest in **Scatterplot** to

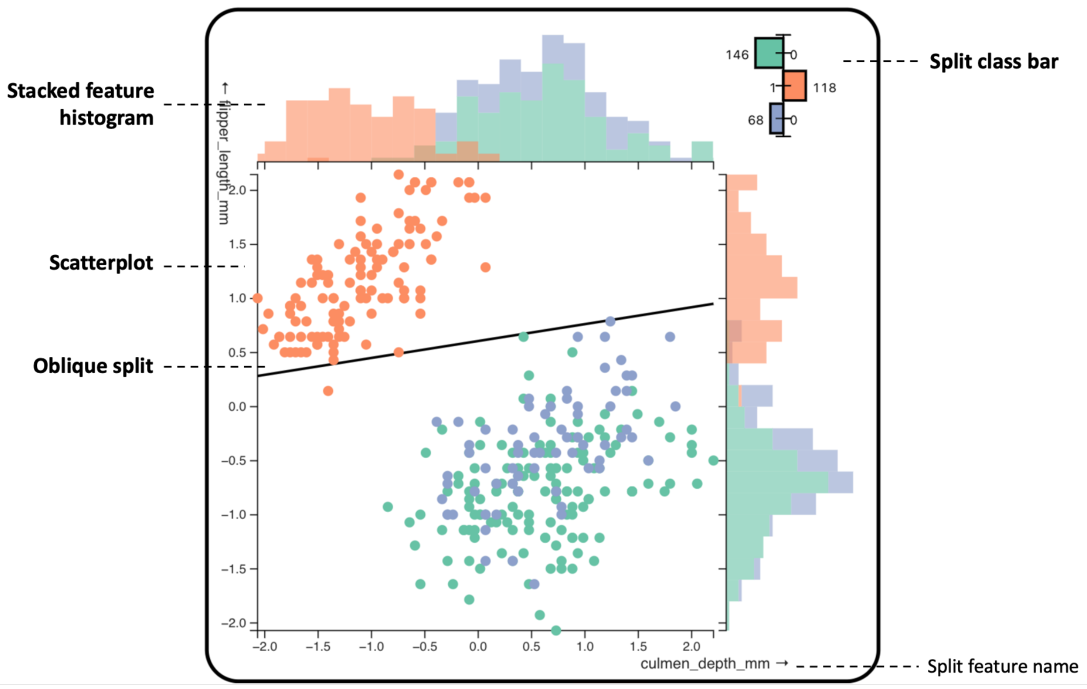


Figure 6.3: Detailed view of the decision nodes containing the oblique split.

gain insight in the underlying structure of the data set. **Split class bar** is also included in the detailed view to provide a precise class distribution under the effect of the current oblique split. When only one feature is involved in a decision node, a **stacked feature histogram** and a split line constitute the detailed view, as shown in Figure 6.4.

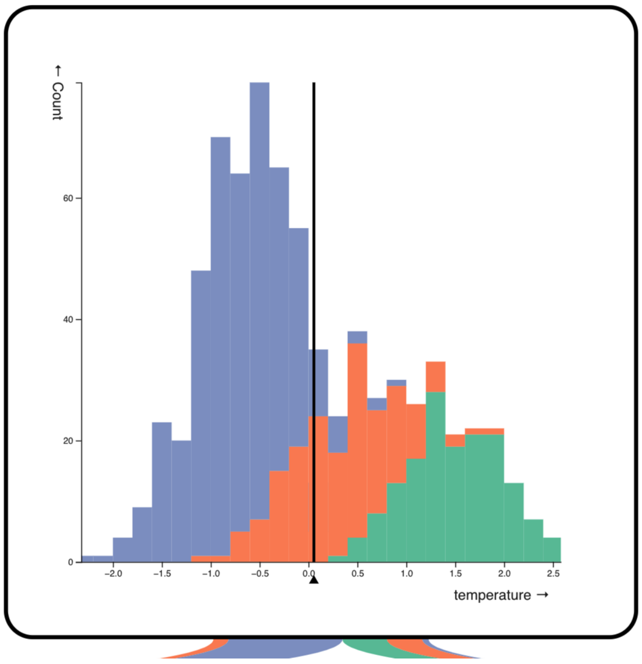


Figure 6.4: Detailed view of the decision nodes containing the axis-parallel split.

In addition, **stacked feature histogram** shows the class distribution of the two subsets separated by the oblique split. This is meant to help users see the effective distribution interval of the oblique split in the feature space. The effective distribution interval of the feature space is presented to the user through the overlapping area of the two histograms. The bar chart shows the contribution of features to each class, and the direction of each bar represents whether the contribution is positive or negative. The color of the bar indicates the corresponding class.

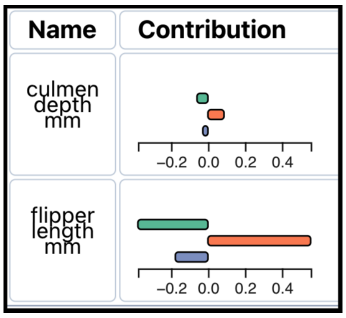
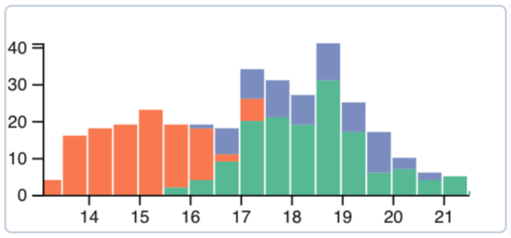


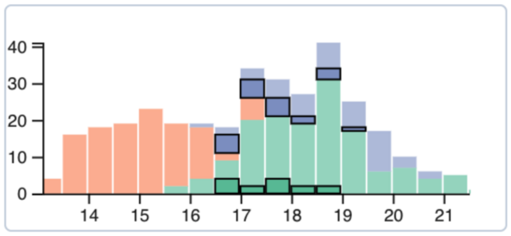
Figure 6.5: Path summary view for leaf nodes.

The path summary view shows a data visualization of the feature contribution values for each class, which contains features that all contribute to the leaf node results to address task **T5**. The bars indicate the contribution value of the feature on each class, and the direction of the bars refers to whether the corresponding contribution value is positive or negative. Each bar is filled with color according to the corresponding class.

6.3 Feature View



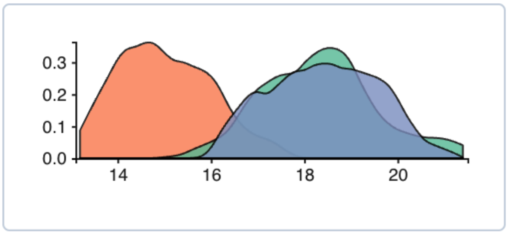
(a) Stacked histogram for continuous feature `culmen_depth_mm` in Palmer Penguin data.



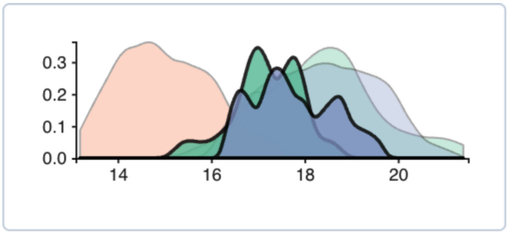
(b) Highlighted stacked histogram for continuous feature `culmen_depth_mm` in Palmer Penguin data after the user selected a subset

Figure 6.6: Stacked histogram as one of the visual designs for continuous feature distribution visualization.

In the feature view (see Figure 6.1D), the feature name, feature contribution value, and feature distribution each occupy a separate column to form a feature table. To support task **T1**, we use density plots to show the distribution of continuous features and apply bar charts for categorical features. The stacked histogram was considered in our preliminary visual design to show the distribution of continuous features, as shown in Figure 6.6a. When a subset is selected, black-bordered boxes are highlighted bars in the stacked histogram. Since they are not on the same line, this makes it difficult for the user to visually compare feature distributions between different classes, as shown in Figure 6.6b. The advantage of using probability distributions as our display option is that it avoids the obscurity of some classes in the case of unbalanced data. The visual design using probability distributions as the presented feature distributions allows the use of an additional density plot of the selected subset to overlay the original plots without affecting the view of the overall feature distributions. This also allows the user to clearly compare feature distributions across classes, as Figure 6.7 shows.



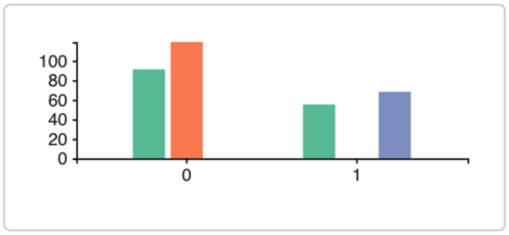
(a) Density plot for continuous feature culmen depth in Palmer Penguin data.



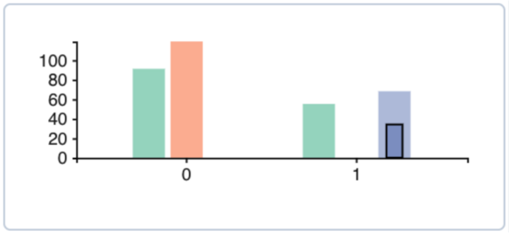
(b) Highlighted density plot for continuous feature culmen depth in Palmer Penguin data after the user selected a subset

Figure 6.7: Density plot as one of the visual designs for continuous feature distribution visualization.

When a subset is selected, the probability distribution of the subset is overlaid on the original plot with a lower opacity degree. For categorical features as shown in Figure 6.8, a new bar is overlaid on the original chart, while the original bar is pulled up for transparency and the width of the new bar is shortened to achieve visual contrast.



(a) Bar chart for categorical feature dream island in Palmer Penguin data.



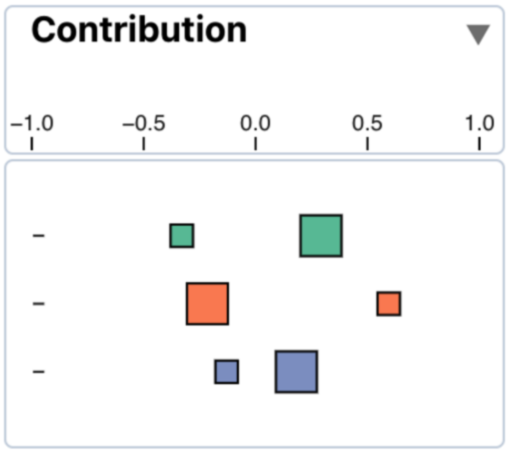
(b) Highlighted bar chart for categorical feature dream island in Palmer Penguin data after the user selected a subset

Figure 6.8: Bar chart as one of the visual designs for categorical feature distribution visualization.

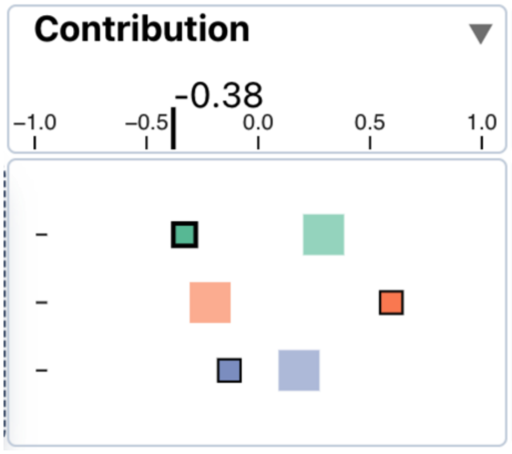
To support task **T5**, the feature view also presents feature-relevance explanations in the middle column of the feature table. Feature contribution values per class are visualized in rect plots (see Figure 6.9) to assist users in comparing explanations cross features. Each cell summarizes the feature contribution values of different decision paths for the corresponding feature. The cell is empty if the selected tree classifier does not use the feature.

In our study, we use local feature contribution values as feature-relevance explanation (Section 2.3) proposed by Palczewska *et al*. [25]. Feature contribution values are computed per instance. Instances have the same feature contribution values if they all go through the same decision path. This allows each decision path in the whole tree classifier to have an independent feature contribution value. However, the number of decision paths in a classifier can be as low as a few or as many as several dozens, so it is difficult to present the feature contribution value to the user with the help of statistical analysis-type visualization tools.

As Figure 6.9a shows, each rectangle represent a feature contribution value for the corresponding class, feature, and decision path. The size of each rectangle represents how



(a) Rect plot of feature contribution visualization for continuous feature culmen length in Palmer Penguin data.



(b) Highlighted rect plot of feature contribution visualization for continuous feature culmen length in Palmer Penguin data after the user selected a subset

Figure 6.9: Rect plot as one of the visual designs for feature contribution visualization.

many instances go through the corresponding decision path. The position of each rectangle on the x-axis represents its corresponding feature contribution value. Such a visual design helps the user to clearly see how the corresponding features act on the whole tree classifier. In addition, when a subset is selected, the feature contribution value that has an impact on the instances in the subset is highlighted, and the rest of the rectangles are made more transparent to provide visual contrast in the Figure 6.9b shown.

In addition, with such an aligned layout, we decide to place the x-axis in the header of the column as shown in Figure 6.9a. On the one hand this avoids duplicate presentation of the x-axis in each cell, and on the other hand the uniform x-scale helps the user compare the contribution of different features to the tree classifier. Since such a visual design only provides an approximate value to the user, for each rectangle, we add a hover effect on each rectangle. As shown in Figure 6.9b, when a rectangle is hovered over by the cursor, the x-axis in the header above appears with the corresponding exact feature contribution value to the user.

Apart from feature contribution visualization, another concern is which features are more important. This requires us to rank feature contribution values. Since there is no unique way to compute this ranking, we decide to use average absolute feature contribution value as the metric to sort feature cells. The user can click on the downward-flipping triangle in the header to sort the feature cells as Figure 6.9a shown.

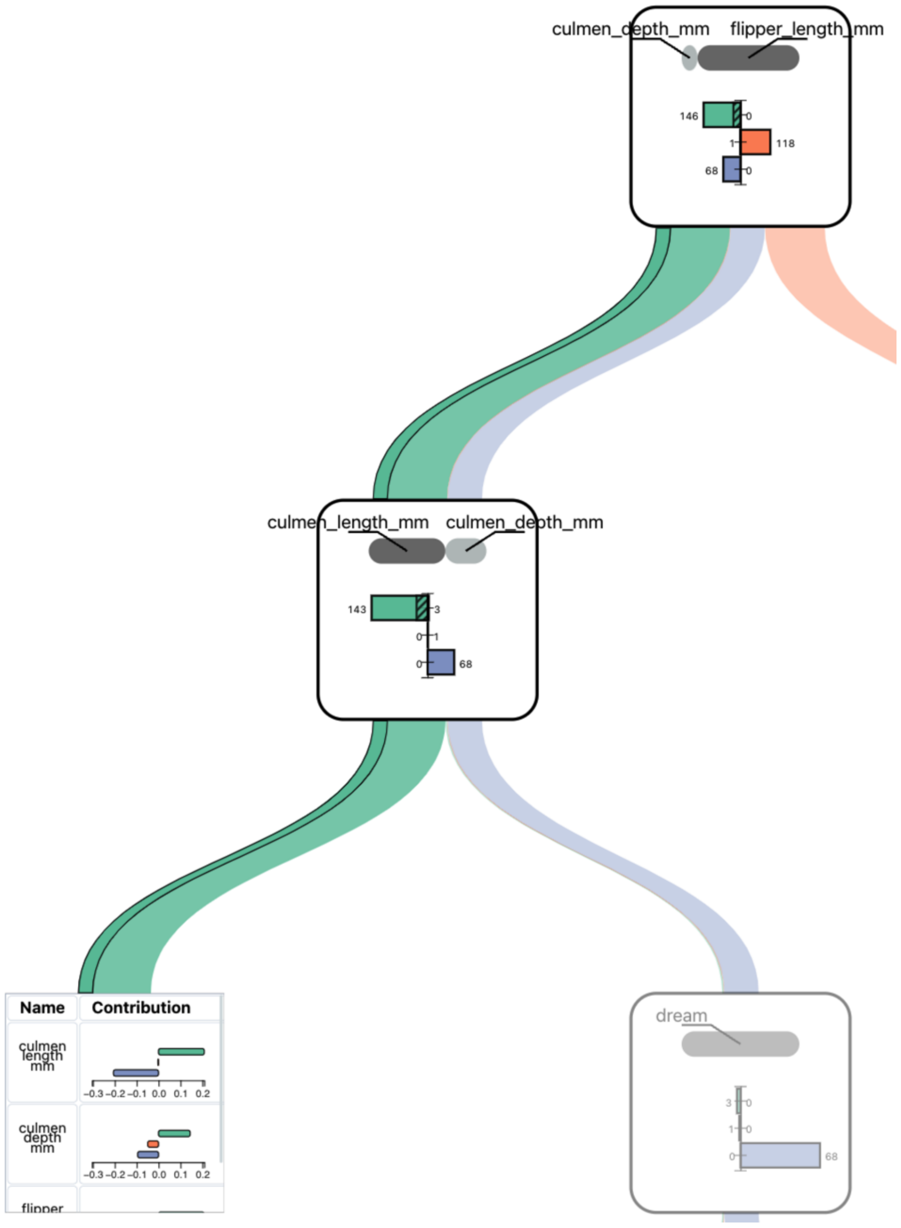


Figure 6.10: Highlighted decision path in the oblique tree view.

6.4 Projection View

The projection view applies t-Distributed Stochastic Neighbor Embedding (t-SNE) [35] with Euclidean distance to map the feature space into two dimensions. Instances are shown as point marks colored according to their classes. The projection view assists in exploring data similarity, addressing task T1.

When a subset is selected in the projection view, its corresponding decision path is highlighted in the oblique tree view as shown in Figure 6.10. Nodes not related to the highlighted decision path are made more transparent, and secondly, the links of the highlighted decision path are also re-rendered, and the width of the new links is scaled based on the number of instances of the selected subset. Meanwhile, those features involved in the decision path of the selected subset are highlighted in the feature view, as shown in Figure 6.11. There-

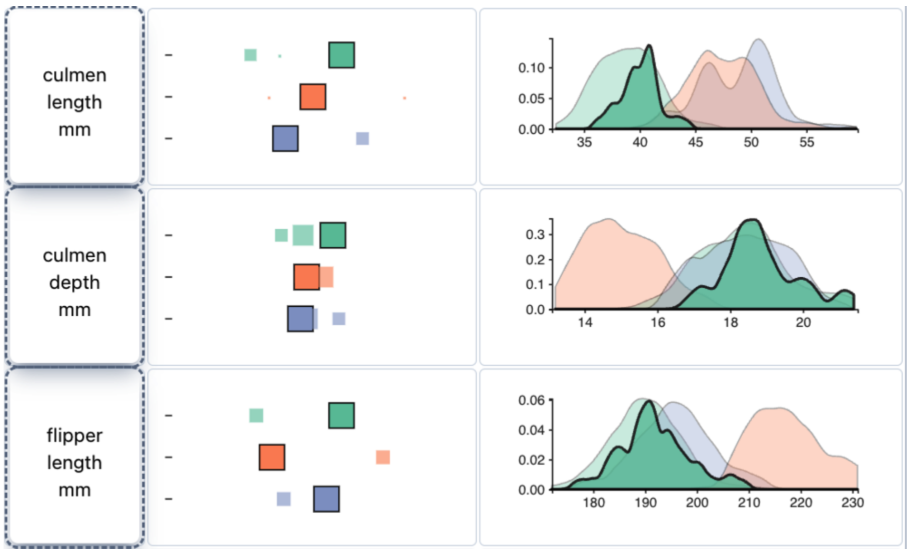


Figure 6.11: Highlighted feature rows in the feature view.

fore, the projection view helps the user to build a bridge between the oblique tree view and the feature view regarding the decision path. This allows the user to select the subset of interest and then explore the feature characteristics and feature-relevance explanation of the decision path under the dynamic interaction of the three major views.

Chapter 7

Evaluation

In this chapter, we present two use cases and a quantitative user study to demonstrate how ObliqueTreeVis effectively helps users understand the behavior of an oblique decision tree in the context of visual analysis in comparison to traditional decision trees.

7.1 Use Case: Palmer Penguin Species Classification

In this use case, we demonstrated how ObliqueTreeVis helps people understand an oblique decision tree. Palmer Penguin Data comprises various measurements of 333 penguins. The task is to classify the different penguin species, which are Adelie, Gentoo, and Chinstap. Each data instance contains eight features: weight, culmen length, culmen depth, filpper length, sex, and island of residence (three features processed with one-hot encoding).

In this case, we show how to use ObliqueTreeVis to investigate which features are critical for a certain prediction and how these features contribute to the corresponding prediction. First, we focus on investigating features that are critical to Gentoo penguins, addressing task **T5**. First of all, through the oblique tree view, we can clearly see that the classification result of the current two decision paths is the Gentoo penguin species, as shown in Figure 7.1A. Next, we consult the path summary view (see Figure 7.1B) in the leaf node of the two decision paths and find out that the features with the highest contribution are flipper length and culmen length (**T4**). To further understand why these two features are critical to the prediction of Gentoo penguins, we next look at the decision boundary involving these features. Therefore, we use the feature selection in the feature view (see Figure 7.1C) to quickly locate decision nodes that involve these features. Take flipper length as an example. It successfully divides 118 penguins of Gentoo species in the root node of the tree under the joint effect of culmen depth. By observing the summary view of the decision node as shown in Figure 7.1D, we can find that the filpper length does contribute to the main partition effect in the oblique split, which can be seen from the feature coefficient bars (**T3**). Secondly, through the stacked feature histogram corresponding to the flipper length in the detailed view (see Figure 7.1E), the current oblique decision tree tells us that Gentoo penguins have longer flippers than the other two species. (**T1**).

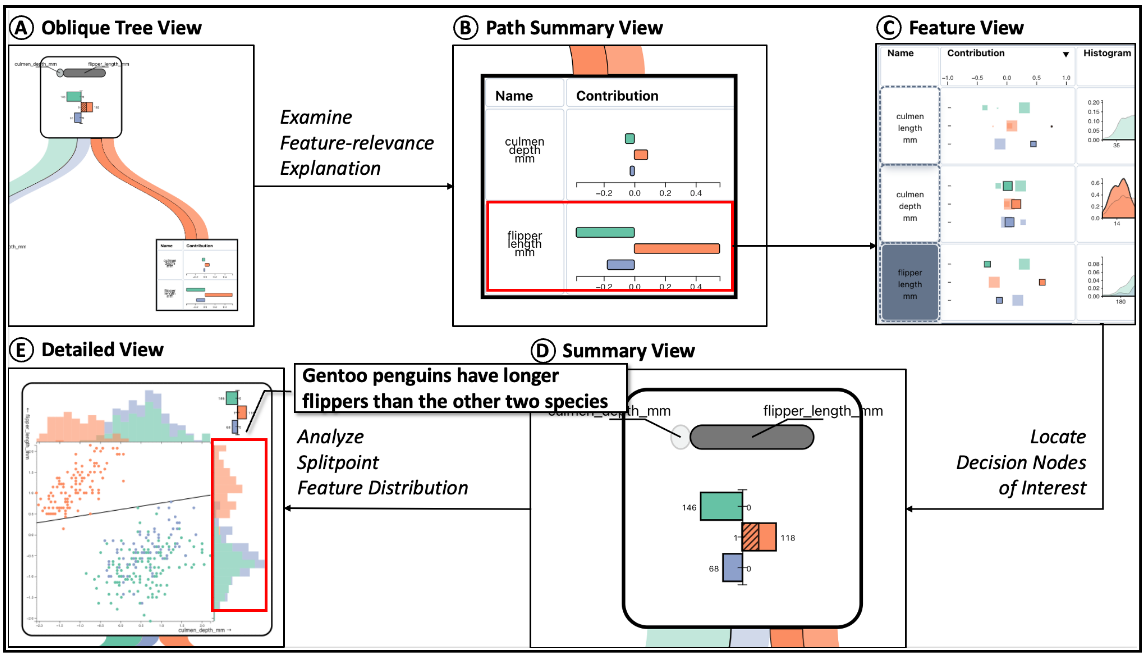


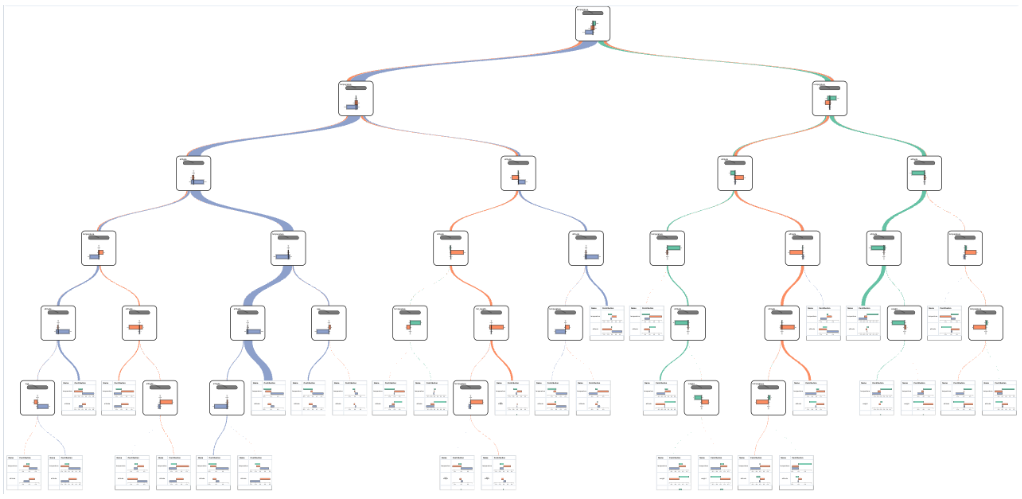
Figure 7.1: Using ObliqueTreeVis to interpret oblique decision trees with Palmer Penguin Data: (A) One highlighted decision path in the oblique tree view when the Bivariate Decision Tree model is selected. (B) Flipper length contributes the most in the path summary view. (C) Select filpper length in the feature view. Decision nodes involving the selected feature are highlighted. (D) Investigate the highlighted decision node in the summary view. (E) Analyze splitpoint feature distribution from the stacked feature histogram in the detailed view.

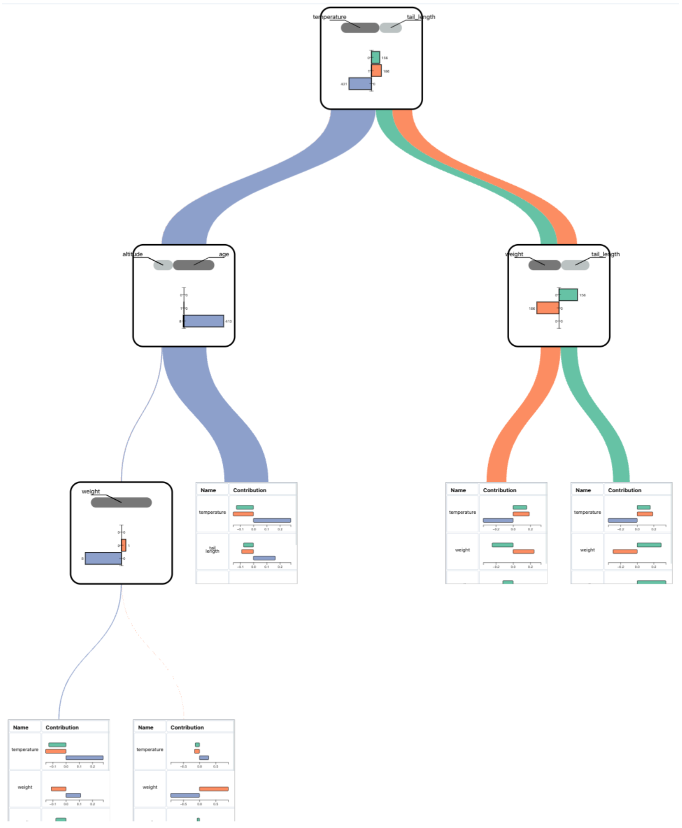
7.2 Use Case: Horseshoe Crab Species Classification

In this use case, we use an artificially synthesized data set to evaluate the interpretability of oblique decision trees and traditional decision trees in the context of visual analytics.

Horseshoe Crab Data. This is a data set for classification problems of an artificially created fictional species, the horseshoe crab. The data set contains five continuous features, namely, temperature, weight, tail length, altitude, and age. It contains three species, Rotund, Gigas, and Trident horseshoe crabs. Through matrix operations, the original isotropic data set becomes anisotropically distributed. This results in positive or negative correlations between the five features. In addition, the fifth feature, age, is added to the feature space as noise, which obeys a Gaussian distribution and is a redundant feature. This data set contains 764 instances, of which the distribution of classes (Rotund, Gigas, and Trident) is 156, 187, and 421.

Comparing the interpretability between oblique decision trees and traditional decision trees. At the beginning, we trained the data set using oblique decision trees and decision trees, respectively. The pruning algorithm is not used in either model. The final oblique decision tree generated a tree with 3 levels of depth and a total of 4 decision nodes (internal nodes that are not leaf nodes). In contrast, the traditional decision tree CART produced a tree with a depth of 6 levels and a staggering 32 decision nodes, as Figure 7.2 shows. From these two sets of data, it can be seen that the oblique decision tree generates smaller tree sizes when there is a significant correlation between the features. Then, we followed the structure of the tree model to explore decision nodes and decision paths. When exploring the oblique decision tree model, we only needed to look at the detailed view in the root node alone to understand the characteristics of the data set in terms of temperature and tail length (the tail length of horseshoe crab is shorter when the living environment is warmer),


(a) An example of a traditional decision tree, trained with Horseshoe Crab Data.



(b) An example of an oblique decision tree, trained with Horseshoe Crab Data.

Figure 7.2: Comparison of two decision tree models trained from Horseshoe Crab Data. The oblique decision tree produces a tree with a depth of 3 levels and a total of 4 decision nodes. In contrast, the traditional decision tree CART produced a tree with a depth of 6 levels and 32 decision nodes.

and the combination of these two features is able to classify three different species of horseshoe crab. However, when we explored the traditional decision tree, to get insight in the underlying structure of the data conveyed by the model, we were required to explore several decision nodes. Before we can make valid conclusions on the relationship between altitude and temperature, we must investigate the whole left side of the decision tree. Under the combined influence of the two features, temperature and altitude, the traditional decision tree was able to classify the species Gigas and Trident. However, the traditional decision tree was unable to provide more specific information, including a positive correlation between these two features. This tells us that oblique decision trees have better interpretability in the context of visual analytics compared to traditional decision trees.

7.3 User Study

To illustrate how ObliqueTreeVis assists users in comprehending the behavior of an oblique decision tree in the context of visual analysis as compared to traditional decision trees, we perform a two-section online quantitative study [18]. Detailed user study questions can be found in Appendix A. With the help of ObliqueTreeVis, questions are made to find out how well ObliqueTreeVis helps users understand oblique decision trees and how easy it is to understand oblique decision trees compared to traditional decision trees. In two sections, we ask participants to answer relevant questions. After participants complete all the questions, we ask them to complete a questionnaire to understand user preferences and directions for improvement.

Study Design. According to research, age, past experience, and education may operate as moderators that confound individual impacts on experimental results [5]. We recruited 10 master students studying data science. These students all have the basic understanding of machine learning related knowledge.

The research consists of three phases. Each participant receives a 10-minute lesson to begin. In the second stage, respondents are first asked to answer a list of questions to solve tasks (**T2**,**T4**), in section I. Respondents are then asked to answer a list of problem-solving questions to solve tasks (**T3**,**T5**), and the questions are used to evaluate the interpretability of oblique decision trees and traditional decision trees with the assistance of the visual analysis system. In the experiment, half of the respondents are told to use the decision tree to answer the questions in Section II, and the other half are told to use the oblique decision tree to answer the Section II questions. In the formal user study, we use Horseshoe Crab Data and ObliqueTreeVis to interpret a traditional and oblique decision tree.

After the second part of the user study is done, we ask participants to fill out a three-question questionnaire to see how well ObliqueTree works and to get user feedback on ObliqueTreeVis. As Table 7.3 shows, some of the questions are designed using a 5-point Likert scale, ranging from strongly disagree (1) to strongly agree (5). Some of the questions are descriptive questions.

Question	**Task**
List all the leaf nodes of the prediction class i?	[**T2**]
Which of the following decision paths has feature x involved?	[**T4**]

Table 7.1: Questions of section I in the quantitative experiment.

Questions in Section I. Two questions (see Table 7.1) are created to validate the ability of respondents to accomplish tasks (**T2**,**T4**) using ObliqueTreeVis in section I.

Questions in Section II. There is little consensus on what constitutes interpretability in machine learning and how to evaluate it for a benchmark. Lakkaraju *et al*. argued that

[1]Comparison denotes the research of evaluating the interpretability of oblique decision trees and traditional decision trees with the assistance of the visual analysis system.

Question	Task
Which class would the model most likely predict for instance j?	[**T3**,**T5**,**Comparison**[1]]
For the introduced instance j, what features are involved in making the prediction?	[**T3**,**T5**,**Comparison**]
For the introduced instance j, is feature X_u contributing more than feature X_v?	[**T3**,**T5**,**Comparison**]

Table 7.2: Questions of section II in the quantitative experiment.

if humans can understand the decision boundary of a model by looking at the rules, we call a rule-based model an interpretable model [17]. Considering that our model is mainly oriented towards domain experts to help them make task-oriented decisions in their daily work, we can assess the interpretability of the model by analyzing how accurately users understand the decision boundaries conveyed by the model. Moreover, both our third and fifth tasks (**T3**,**T5**) are directly related to the user understanding of the decision boundary conveyed by the model. Therefore, we design three questions examining the understanding of the decision boundary depicted by the model in section II. These three questions are classification-oriented questions, as shown in Table 7.2. The first question requires users to follow decision paths in the model to solve a classification problem. The second question examines whether the respondent can correctly identify which features act on a certain prediction. Also, the third question checks how well the respondent understands feature-relevance explanations for a certain decision path. We prepare two different data instances to be classified for the respondents in section II. In each case, respondents were asked to answer these three questions.

Question: System Usability	**Result**
Do you find this visualization useful for understanding the behavior of an oblique decision tree	4.50/5
Do you think this visualization easy to use?	3.80/5
Please evaluate the ease of use of the different components of the system.	
Oblique Tree View	4.40/5
Feature View	3.80/5
Projection View	3.50/5

Table 7.3: Post-experiment questionnaire.

Measures. For the aforementioned questions, we assess them in three aspects. The first is the accuracy of the answer by the respondent. Because each question is made to be objective and have the best answer, accuracy gives us an objective measure of how well the respondent understands the different tasks. The second perspective used for evaluation is the time spent. This is because for domain experts, the ability to answer questions quickly is critical in their work. Moreover, the dimensions previously proposed by Doshi-Velez *et al*. to evaluate the interpretability of task-oriented machine learning models include time constraints. [10]. The third thing each question looks at is how sure the respondent is about his or her answer. Confidence is measured on a scale of 1 to 5 (Totally Not Confident $= 1, \ldots,$ Very Confident $= 5$).

Results. Table 7.4 presents the results of the user study. Detailed user study results can be found in Appendix B. The average time that respondents took to complete questions in section I is 210.2 seconds. The accuracy of the performed tasks is about 78 % in section I. The average level of confidence in the answers they gave ranges between confident (4) and totally confident (5). All the respondents answered the required questions correctly and confidently most of the time. Combined with the experiment results (see Table 7.4) given by respondents using oblique decision trees in section II, this suggests that the basic usability of our approach is validated.

For the results in section II, we found that the accuracy was higher when respondents were dealing with the oblique decision tree, compared to the case of the decision tree in section II. For respondents using Decision Tree, the most common mistakes they made were: (1) ignoring some features involved in a particular data instance prediction; and (2) giving the wrong feature importance explanation for a particular data instance prediction. By interviewing the respondents who answered incorrectly, they gave the explanation that although they could accurately predict the data instance by the decision tree model, they were prone to forget some predicates due to the long decision path. In addition to that, when they wanted to analyze the feature-relevance explanations for the predicted results, they tended to get the wrong leaf nodes. Because there were many leaf nodes that gave the same result, they could easily locate the wrong leaf node. Although there were also one or two incorrect results for respondents using the oblique decision tree, they were mostly able to avoid the mentioned situations. Respondents explained that they were able to find the corresponding decision path very quickly. Moreover, they could answer the questions with confidence based on the feature-relevance explanations given by the path summary view in the leaf node.

The average time taken to complete questions in section II based on the oblique decision tree is roughly 70% of the time taken to complete the same questions in section II based on the decision tree. In addition, answers based on the oblique decision tree had about 20% higher confidence compared to the answers based on the decision tree. This indicates that respondents find it easier to understand decision rules based on the oblique decision tree.

Section	Metric	ODT	DT
section I	Accuracy	0.78	—
	Time Spent	210.2	
	Confidence	4.25	
section II: instance A	Accuracy	0.87	0.67
	Time Spent	315.0	418.6
	Confidence	4.07	3.47
section II: instance B	Accuracy	0.91	0.67
	Time Spent	236.4	354.6
	Confidence	3.93	3.07

Table 7.4: Results of the user study. Section I evaluated how ObliqueTreeVis accomplished task **T2** and **T4**. Section II accessed how ObliqueTreeVis accomplished task **T3** and **T5**, and compared interpretability of Oblique Decision Trees (ODT) and Decision Trees (DT).

In the post-experiment questionnaire section, most participants valued the effectiveness of ObliqueTreeVis in helping them understand oblique decision trees and how to make specific predictions. In particular, they appreciated the usefulness of Oblique Tree View. For the Oblique Tree View, one respondent commented that he liked that this view "helped him present decision boundaries within a single split, which a textual data equation could not provide"; and that he was able to "interactively explore data points of interest through the scatterplot". But one respondent mentioned that he wanted to "explore the projection view under certain features involved" in the process of user study. However, ObliqueTreeVis does not currently satisfy his needs.

Chapter 8

Conclusions and Future Work

The key findings are presented in this chapter. The limits of ObliqueTreeVis and future research are then discussed.

8.1 Conclusions

This study aims to develop a visual analytics system to enhance the interpretability of oblique decision trees and evaluate the interpretability of oblique decision trees *vs*. traditional decision trees in the context of visual analysis. To achieve the research goal, we first give an overview of tree-structured methods and then analyze the situations of our domain experts. This arrives at five user tasks (**T1**-**T5**). Based on the given tasks, we offer relevant research. In the related work, we analyze the effectiveness of these methods and different visualization techniques, serving as the inspiration for our visual design choices.

We focus on oblique trees with bivariate splits to achieve a good balance between interpretability and model size. This decision also enables us to use proven two-dimensional visualization techniques. To help users understand the distribution of classes across decision rules, we build an oblique tree view to show the global structure of the tree model. To reveal the primary working mechanism of the oblique decision tree, we present a summary view and a detailed view of the decision nodes. These two views allow users to explore and understand the partitioning logic of these nodes. To understand the partitioning logic for each decision path, we create a path summary view in each leaf node to support feature-relevance explanations of the prediction results. Furthermore, we build a feature view to show the distribution of different features. This view also provides the user with a global view of how much each feature contributes to the features of different decision paths, further helping to establish the relationship between the features and the prediction results. To show the similarity of the original data set, we provide a projection view. This also allows users to explore interactively across views.

In our evaluation, we provide two use cases and a user study. Through a use case and a user study, we preliminarily validate the effectiveness of the visual analytics system, ObliqueTreeVis. In addition, we demonstrate that users of a visual analytics system using a bivariate decision tree can interpret the model faster and give more accurate classification predictions than a visual analytics system using a traditional decision tree as a model.

8.2 Future Work

Below, we discuss the future work of this visual analytics system.

Scalability of the Visualization. While the current implementation of ObliqueTreeVis can visualize tree models with more than 100 decision nodes, its interpretability has only been validated on tree models with fewer than 40 decision nodes. It is unclear whether users can still gain an overall understanding of more complex tree models. In the future, we may consider using pruning algorithms to reduce the size of the tree model.

Scalability of the Bivariate Decision Tree Induction Method. The oblique decision tree induction algorithm currently used in the study only supports a maximum number of three classes. On the one hand, this affects the usefulness of ObliqueTreeVis; on the other hand, it is not possible to assess whether the number of classes has an impact on the interpretability of the current visual analytics system. Therefore, the next priority is to make the oblique decision tree induction algorithm support data sets with a larger number of classes.

Number of Features in Each Split. Our study builds on the fact that there are readily available traditional two-dimensional visualization techniques to depict the decision boundaries of bivariate splits. This comes at the cost of compromising the model in terms of performance. Instead of using new visualization techniques to describe the hyperplane of an oblique split, we can also consider using algorithms to summarize a complex oblique split. For example, can we express a linear combination of two features with a newly created feature? This gives us the opportunity to describe high-dimensional decision boundaries in a new feature space with the help of visualization techniques. At the same time, we use other methods to help users interpret the meaning of each new feature.

Bibliography

[1] Edgar Anderson. The species problem in iris. *Annals of the Missouri Botanical Garden*, 23(3):457–509, 1936. 23

[2] Mihael Ankerst, Martin Ester, and Hans-Peter Kriegel. Towards an effective cooperation of the user and the computer for classification. In *KDD '00*, 2000. 16, 17

[3] Todd Barlow and Padraic Neville. A comparison of 2-d visualizations of hierarchies. In *Proceedings of the IEEE Symposium on Information Visualization 2001 (INFOVIS'01)*, INFOVIS '01, page 131, USA, 2001. IEEE Computer Society. 28

[4] William A Belson. Matching and prediction on the principle of biological classification. *Journal of the Royal Statistical Society: Series C (Applied Statistics)*, 8(2):65–75, 1959. 6

[5] Izak Benbasat and Ronald N. Taylor. Behavioral aspects of information processing for the design of management information systems. *IEEE Transactions on Systems, Man, and Cybernetics*, 12:439–450, 1982. 40

[6] Jan C Bioch, Onno van der Meer, and Rob Potharst. Bivariate decision trees. In *European Symposium on Principles of Data Mining and Knowledge Discovery*, pages 232–242. Springer, 1997. 7

[7] Ferdinand Bollwein and Stephan Westphal. A branch & bound algorithm to determine optimal bivariate splits for oblique decision tree induction. *Appl. Intell.*, 51:7552–7572, 2021. 7

[8] Leo Breiman, Jerome Friedman, Charles J. Stone, and R.A. Olshen. *Classification and Regression Trees*. Chapman and Hall/CRC, 1984. 1, 6, 8

[9] Leo Breiman, Jerome H Friedman, Richard A Olshen, and Charles J Stone. *Classification and regression trees*. Routledge, 2017. 21

[10] Finale Doshi-Velez and Been Kim. Towards a rigorous science of interpretable machine learning. *arXiv: Machine Learning*, 2017. 41

[11] Rory A. Fisher. The use of multiple measurements in taxonomic problems. *Annals of Human Genetics*, 7:179–188, 1936. 23

[12] Kristen B. Gorman, Tony D. Williams, and William R. Fraser. Ecological sexual dimorphism and environmental variability within a community of antarctic penguins (genus pygoscelis). *PLoS ONE*, 9, 2014. 23

[13] David Heath, Simon Kasif, and Steven Salzberg. Induction of oblique decision trees. In *IJCAI*, volume 1993, pages 1002–1007. Citeseer, 1993. 8

[14] Joseph B. Kruskal and James M. Landwehr. Icicle plots: Better displays for hierarchical clustering. *The American Statistician*, 37:162–168, 1983. 16

[15] Joseph B Kruskal and Myron Wish. *Multidimensional scaling*. Number 11. Sage, 1978. 13

[16] Victor E Kuz'min, Pavel G Polishchuk, Anatoly G Artemenko, and Sergey A Andronati. Interpretation of qsar models based on random forest methods. *Molecular informatics*, 30(6-7):593–603, 2011. 21

[17] Himabindu Lakkaraju, Stephen H Bach, and Jure Leskovec. Interpretable decision sets: A joint framework for description and prediction. In *Proceedings of the 22nd ACM SIGKDD international conference on knowledge discovery and data mining*, pages 1675–1684, 2016. 41

[18] Chicheng Liu. oblique-tree-user-study. `https://github.com/TbabmBarry/oblique-tree-user-study`, 2022. 40

[19] Chicheng Liu. Obliquetreevis: Oblique decision tree visualization. `https://github.com/TbabmBarry/ObliqueTreeVis`, 2022. 27

[20] Yan Liu and Gavriel Salvendy. Design and evaluation of visualization support to facilitate decision trees classification. *Int. J. Hum. Comput. Stud.*, 65:95–110, 2007. 2, 16

[21] Leland McInnes, John Healy, and James Melville. Umap: Uniform manifold approximation and projection for dimension reduction. *arXiv preprint arXiv:1802.03426*, 2018. 13

[22] Thomas Mühlbacher, Lorenz Linhardt, Torsten Möller, and Harald Piringer. Treepod: Sensitivity-aware selection of pareto-optimal decision trees. *IEEE Transactions on Visualization and Computer Graphics*, 24:174–183, 2018. 16, 17, 20

[23] Sreerama K. Murthy, Simon Kasif, and Steven Salzberg. A system for induction of oblique decision trees. *J. Artif. Int. Res.*, 2(1):1–32, aug 1994. 1, 8

[24] Trong Dung Nguyen, Tu Bao Ho, and Hiroshi Shimodaira. Interactive visualization in mining large decision trees. In *Proceedings of the 4th Pacific-Asia Conference on Knowledge Discovery and Data Mining, Current Issues and New Applications*, PADKK '00, page 345–348, Berlin, Heidelberg, 2000. Springer-Verlag. 16

[25] Anna Palczewska, Jan Palczewski, Richard Marchese Robinson, and Daniel Neagu. Interpreting random forest classification models using a feature contribution method. *ArXiv*, abs/1312.1121, 2013. 9, 21, 25, 32

[26] Terence Parr, Tudor Lapusan, and Prince Grover. Decision tree visualization. `https://github.com/parrt/dtreeviz`, 2022. 18

[27] Kristin Potter, Joe Kniss, Richard Riesenfeld, and Chris R. Johnson. Visualizing Summary Statistics and Uncertainty. *Computer Graphics Forum*, 2010. 13, 14

[28] J. Ross Quinlan. Improved use of continuous attributes in c4.5. *J. Artif. Intell. Res.*, 4:77–90, 1996. 1

[29] Marco Tulio Ribeiro, Sameer Singh, and Carlos Guestrin. "why should i trust you?": Explaining the predictions of any classifier. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '16, page 1135–1144, New York, NY, USA, 2016. Association for Computing Machinery. 20, 21

[30] Daniel Smilkov, Nikhil Thorat, Charles Nicholson, Emily Reif, Fernanda B. Viégas, and Martin Wattenberg. Embedding projector: Interactive visualization and interpretation of embeddings. *ArXiv*, abs/1611.05469, 2016. 13

[31] Gabriele Tolomei, Fabrizio Silvestri, Andrew Haines, and Mounia Lalmas. Interpretable predictions of tree-based ensembles via actionable feature tweaking. In *Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '17, page 465–474, New York, NY, USA, 2017. Association for Computing Machinery. 21

[32] Alfred Kar Yin Truong. Fast growing and interpretable oblique trees via logistic regression models. 2009. v, 5, 7, 18, 19

[33] Stef van den Elzen and Jarke J. van Wijk. Baobabview: Interactive construction and analysis of decision trees. *2011 IEEE Conference on Visual Analytics Science and Technology (VAST)*, pages 151–160, 2011. 2, 19, 20

[34] Laurens Van der Maaten and Geoffrey Hinton. Visualizing data using t-sne. *Journal of machine learning research*, 9(11), 2008. 13

[35] Laurens van der Maaten and Geoffrey E. Hinton. Visualizing data using t-sne. *Journal of Machine Learning Research*, 9:2579–2605, 2008. 34

[36] B. G. M. Vandeginste. Parvus: An extendable package of programs for data exploration, classification and correlation, m. forina, r. leardi, c. armanino and s. lanteri, elsevier, amsterdam, 1988, price: Us $645 isbn 0-444-43012-1. *Journal of Chemometrics*, 4, 1990. 23

[37] Jun Wang, Bei Yu, and Les Gasser. Concept tree based clustering visualization with shaded similarity matrices. *2002 IEEE International Conference on Data Mining, 2002. Proceedings.*, pages 697–700, 2002. 16

[38] Malcolm Ware, Eibe Frank, Geoffrey Holmes, Mark Hall, and Ian H. Witten. Interactive machine learning: Letting users build classifiers. *Int. J. Hum.-Comput. Stud.*, 56(3):281–292, mar 2002. 16

[39] Soeren H Welling, Hanne HF Refsgaard, Per B Brockhoff, and Line H Clemmensen. Forest floor visualizations of random forests. *arXiv preprint arXiv:1605.09196*, 2016. 21

[40] Svante Wold, Kim Esbensen, and Paul Geladi. Principal component analysis. *Chemometrics and intelligent laboratory systems*, 2(1-3):37–52, 1987. 13

[41] Jianting Zhang, Le Gruenwald, and Michael Gertz. Vdm-rs: A visual data mining system for exploring and classifying remotely sensed images. *Comput. Geosci.*, 35(9):1827–1836, sep 2009. 16

Appendix A

User Study Questionnaire

Welcome to the ObliqueTreeVis visual analytics system user study!

Next

1. What is your user id? *

Previous

Next

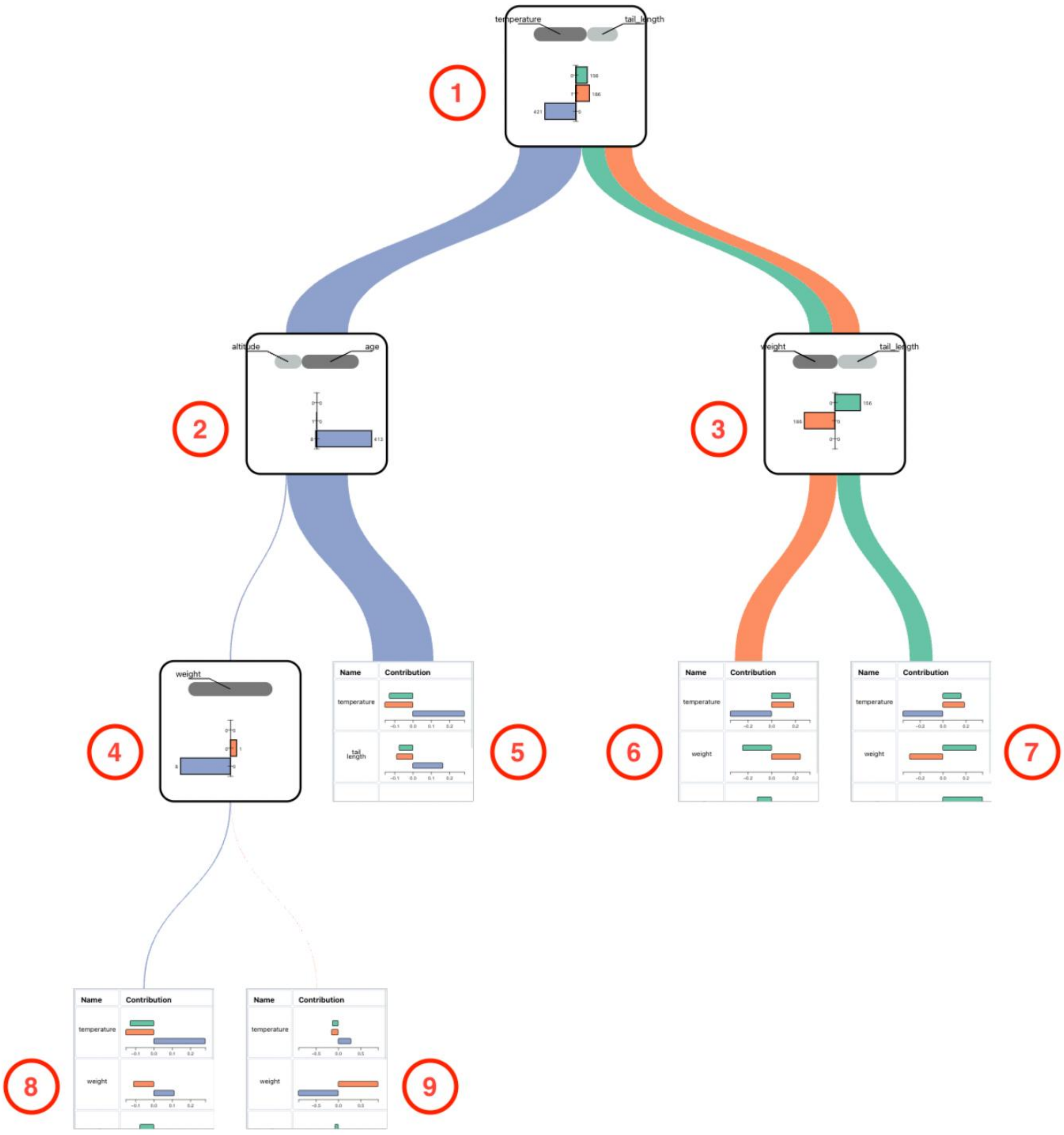
Section I

Experiment Setup

- Please enter the system (ObliqueTreeVis entrance is provided in the document)
- Load Houseshoe Crab Data
- Choose Bivariate Decision Tree

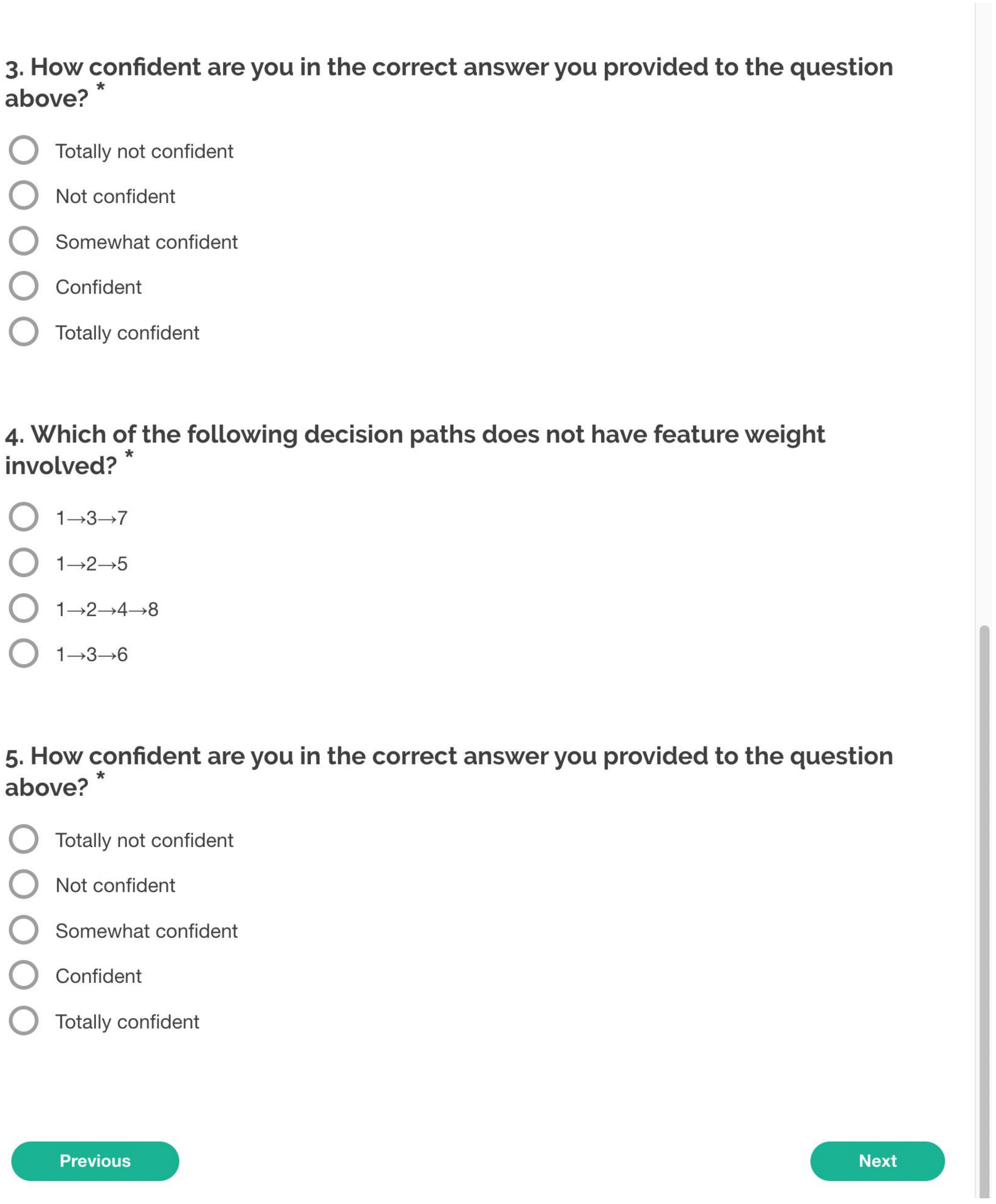
Previous

Next



The purpose of the above image is to provide information on the labeled serial number of each node, please answer the questions in the context of the system.

2. List all the leaf nodes of the prediction class Gigas (Answer with the node number in the diagram, using a comma as a separator). *



3. **How confident are you in the correct answer you provided to the question above? ***

- ○ Totally not confident
- ○ Not confident
- ○ Somewhat confident
- ○ Confident
- ○ Totally confident

4. **Which of the following decision paths does not have feature weight involved? ***

- ○ 1→3→7
- ○ 1→2→5
- ○ 1→2→4→8
- ○ 1→3→6

5. **How confident are you in the correct answer you provided to the question above? ***

- ○ Totally not confident
- ○ Not confident
- ○ Somewhat confident
- ○ Confident
- ○ Totally confident

Section II

Experiment Setup

- Please enter the system (ObliqueTreeVis entrance is provided in the document)
- Load Houseshoe Crab Data
- Choose Bivariate Decision Tree

Previous Next

Suppose the information and measurements of a houseshoe crab instance A:

Feature	Scaled Value	Unscaled Value
Temperature	-1.30	-240.75 °C
Weight	1.50	1000.27 g
Tail Length	1.54	628.44 mm
Altitude	-1.35	-140.53 m
Age	1.36	384.91 day

6. What would the model most likely predict for this crab instance A? *

○ Gigas

○ Rotund

○ Trident

7. How confident are you in the correct answer you provided to the question above? *

○ Totally not confident

○ Not confident

○ Somewhat confident

○ Confident

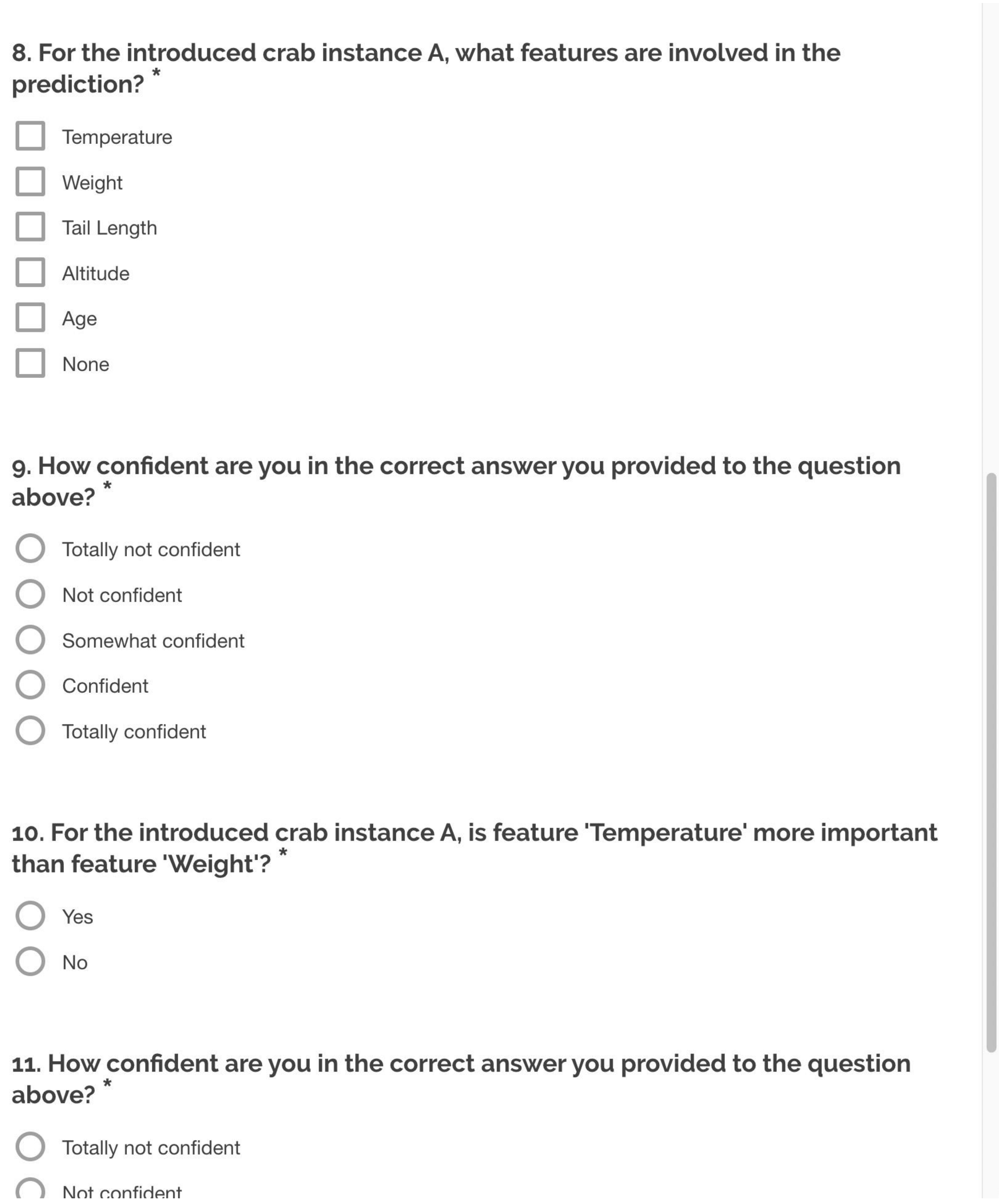
○ Totally confident

8. For the introduced crab instance A, what features are involved in the prediction? *

☐ Temperature

☐ Weight

☐ Tail Length



8. For the introduced crab instance A, what features are involved in the prediction? *

- ☐ Temperature
- ☐ Weight
- ☐ Tail Length
- ☐ Altitude
- ☐ Age
- ☐ None

9. How confident are you in the correct answer you provided to the question above? *

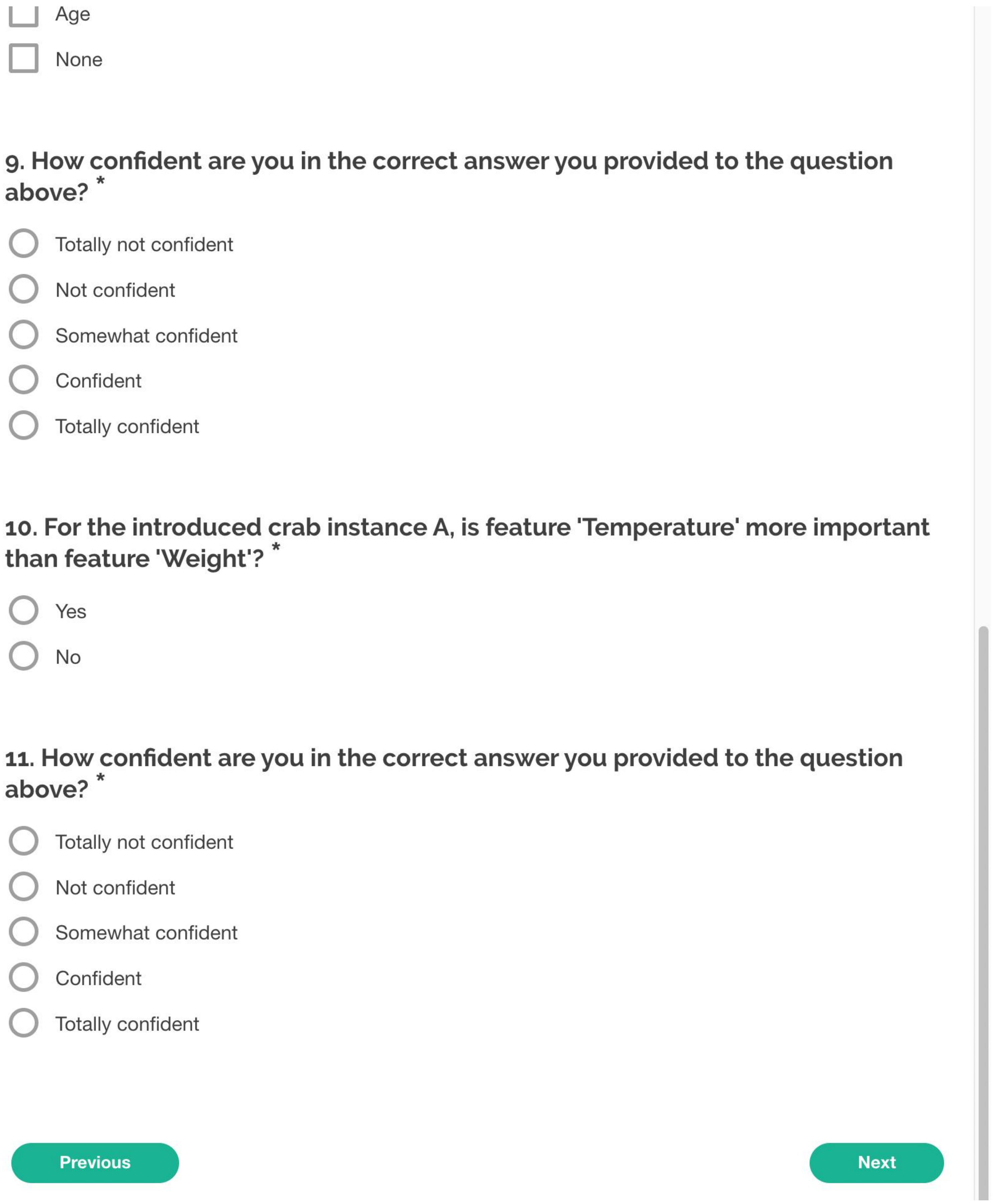
- ○ Totally not confident
- ○ Not confident
- ○ Somewhat confident
- ○ Confident
- ○ Totally confident

10. For the introduced crab instance A, is feature 'Temperature' more important than feature 'Weight'? *

- ○ Yes
- ○ No

11. How confident are you in the correct answer you provided to the question above? *

- ○ Totally not confident
- ○ Not confident



☐ Age

☐ None

9. How confident are you in the correct answer you provided to the question above? *

○ Totally not confident

○ Not confident

○ Somewhat confident

○ Confident

○ Totally confident

10. For the introduced crab instance A, is feature 'Temperature' more important than feature 'Weight'? *

○ Yes

○ No

11. How confident are you in the correct answer you provided to the question above? *

○ Totally not confident

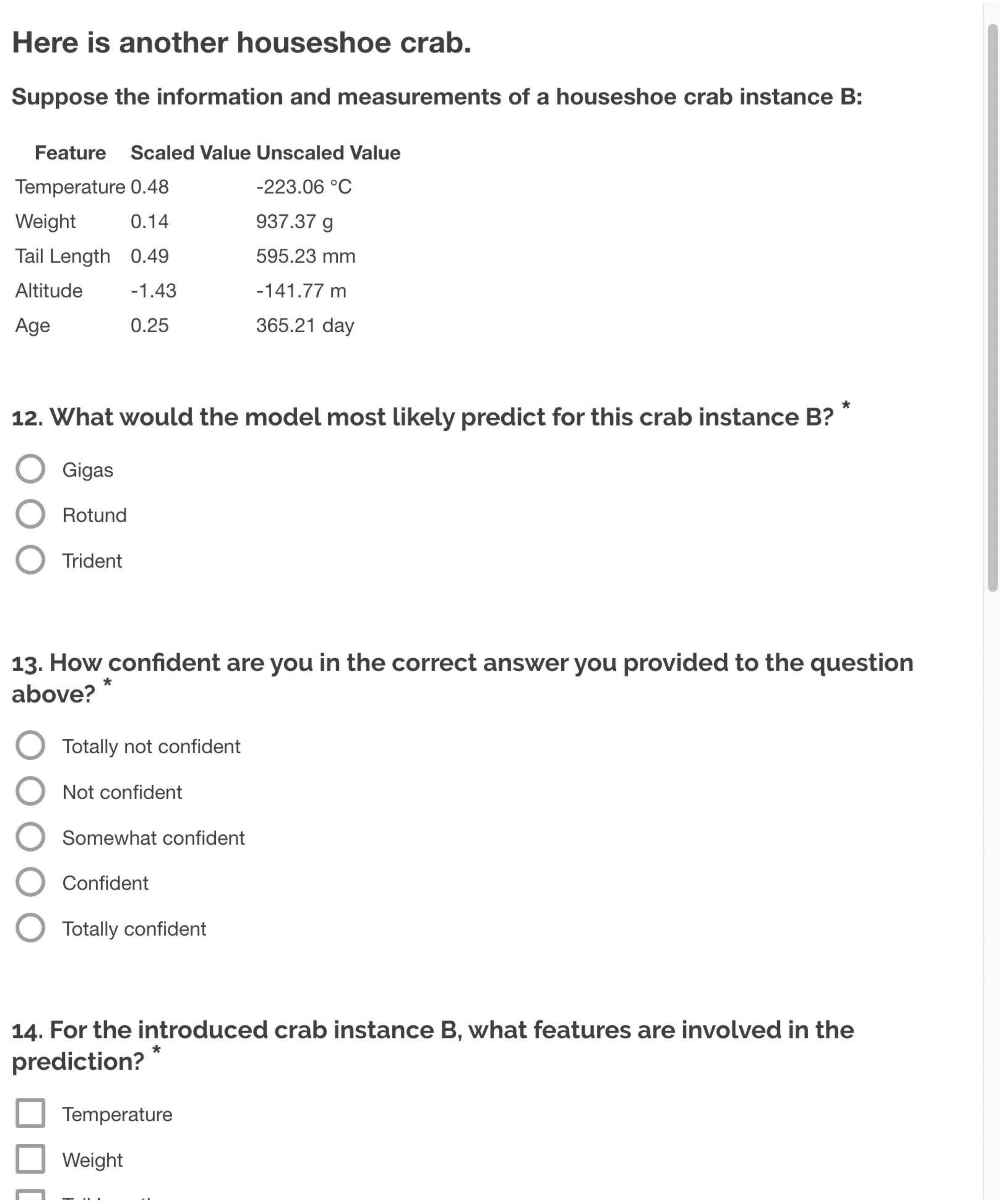
○ Not confident

○ Somewhat confident

○ Confident

○ Totally confident

Previous Next



Here is another houseshoe crab.

Suppose the information and measurements of a houseshoe crab instance B:

Feature	Scaled Value	Unscaled Value
Temperature	0.48	-223.06 °C
Weight	0.14	937.37 g
Tail Length	0.49	595.23 mm
Altitude	-1.43	-141.77 m
Age	0.25	365.21 day

12. What would the model most likely predict for this crab instance B? *

- ○ Gigas
- ○ Rotund
- ○ Trident

13. How confident are you in the correct answer you provided to the question above? *

- ○ Totally not confident
- ○ Not confident
- ○ Somewhat confident
- ○ Confident
- ○ Totally confident

14. For the introduced crab instance B, what features are involved in the prediction? *

- ☐ Temperature
- ☐ Weight

14. For the introduced crab instance B, what features are involved in the prediction? *

- ☐ Temperature
- ☐ Weight
- ☐ Tail Length
- ☐ Altitude
- ☐ Age
- ☐ None

15. How confident are you in the correct answer you provided to the question above? *

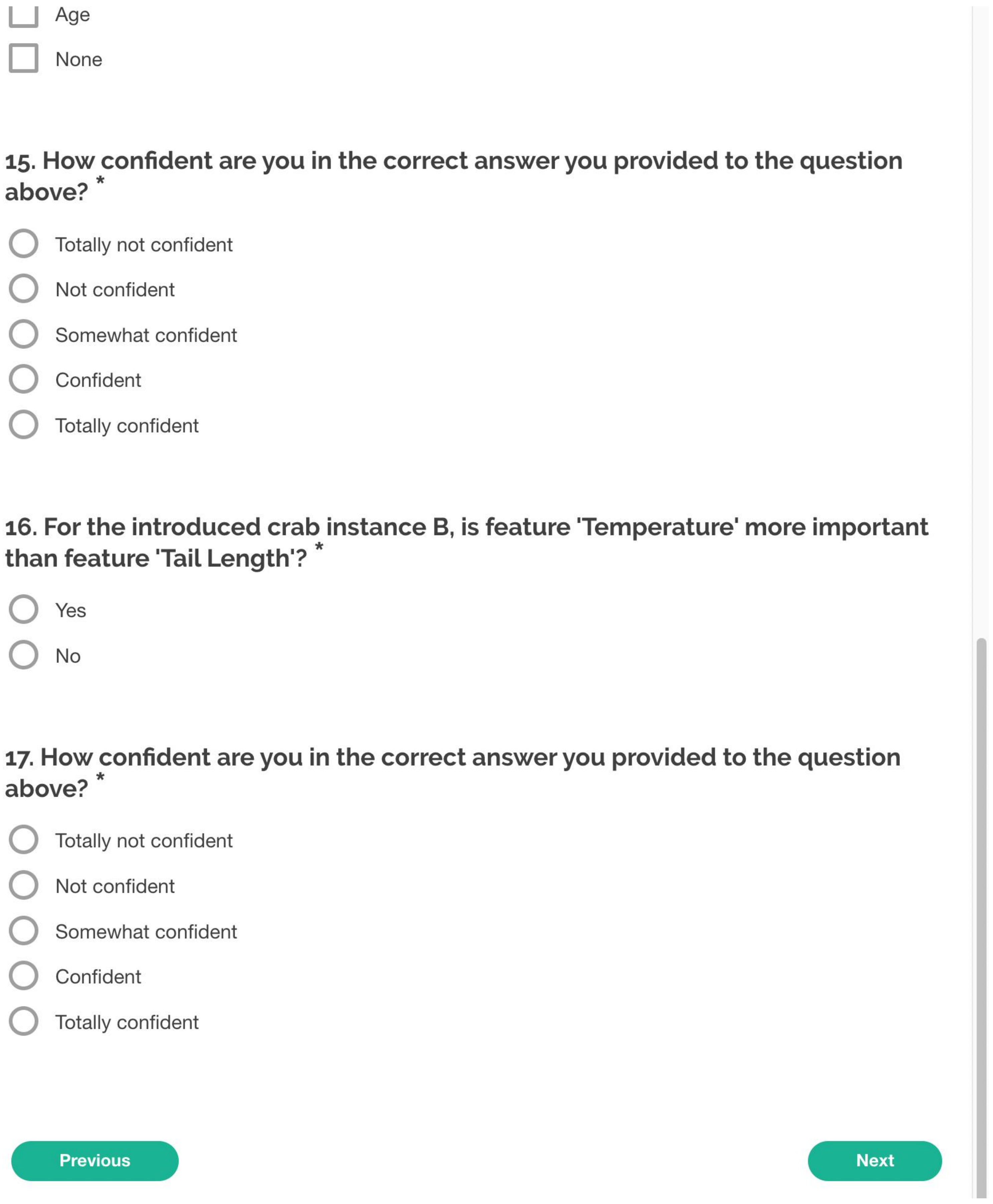
- ○ Totally not confident
- ○ Not confident
- ○ Somewhat confident
- ○ Confident
- ○ Totally confident

16. For the introduced crab instance B, is feature 'Temperature' more important than feature 'Tail Length'? *

- ○ Yes
- ○ No

17. How confident are you in the correct answer you provided to the question above? *

- ○ Totally not confident
- ○ Not confident



☐ Age

☐ None

15. How confident are you in the correct answer you provided to the question above? *

○ Totally not confident

○ Not confident

○ Somewhat confident

○ Confident

○ Totally confident

16. For the introduced crab instance B, is feature 'Temperature' more important than feature 'Tail Length'? *

○ Yes

○ No

17. How confident are you in the correct answer you provided to the question above? *

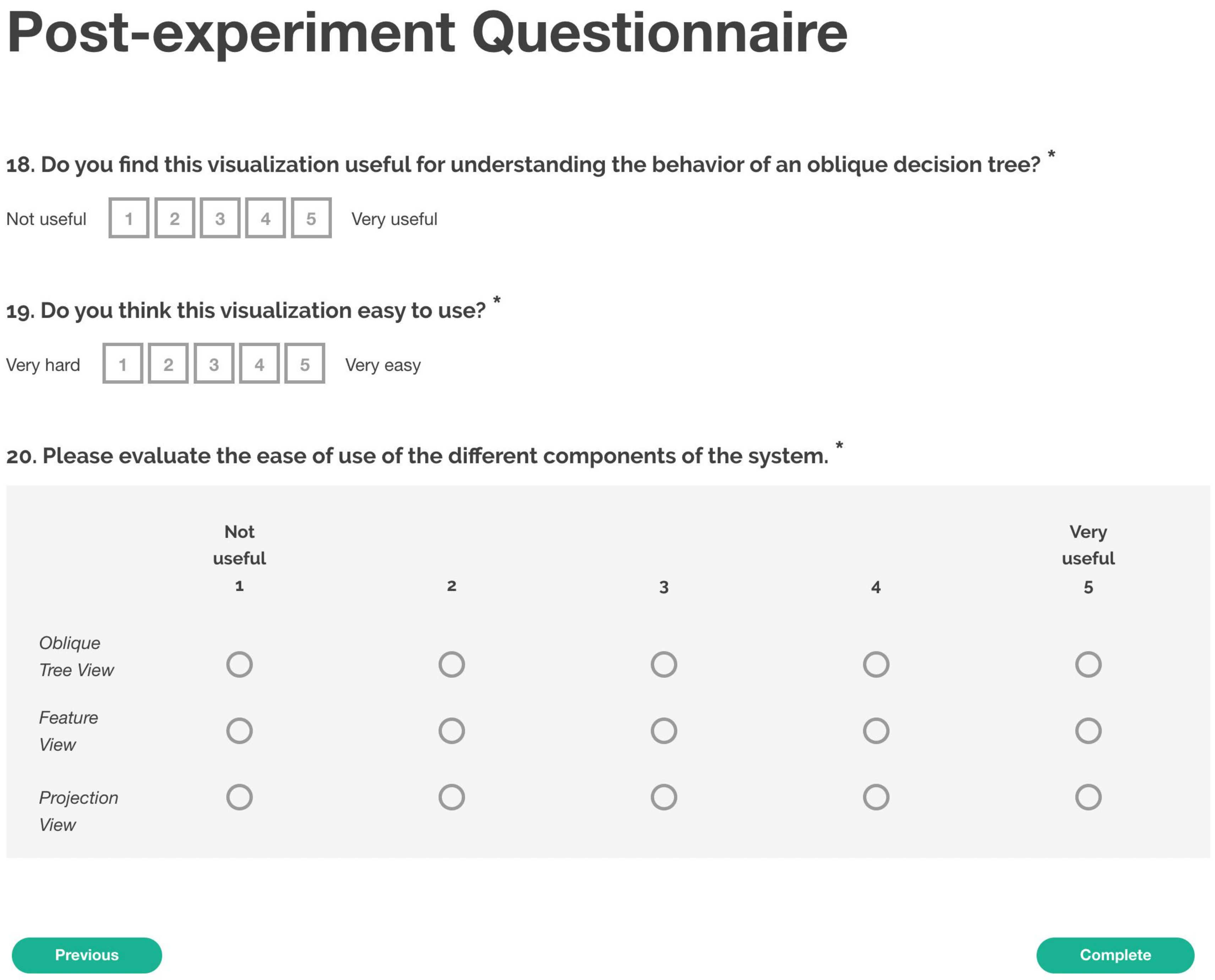
○ Totally not confident

○ Not confident

○ Somewhat confident

○ Confident

○ Totally confident



Post-experiment Questionnaire

18. Do you find this visualization useful for understanding the behavior of an oblique decision tree? *

Not useful 1 2 3 4 5 Very useful

19. Do you think this visualization easy to use? *

Very hard 1 2 3 4 5 Very easy

20. Please evaluate the ease of use of the different components of the system. *

	Not useful 1	2	3	4	Very useful 5
Oblique Tree View	○	○	○	○	○
Feature View	○	○	○	○	○
Projection View	○	○	○	○	○

Appendix B

User Study Results

user id	section two - instance a								
	q1		q2		q3		accuracy	confidence	time spent (s)
	accuracy	confidence	accuracy	confidence	accuracy	confidence			
c4ca4238a0b923820dcc509a6f75849b	1	4	0.75	4	0	2	0.583	3.333	249
dd41c6da7897cd248698870811d89ed6	1	4	1	4	1	4	1	4	410
71c1e8a22dbe10daae0743e628bb991c	1	5	0.5	4	1	3	0.833	4	441
c6f057b86584942e415435ffb1fa93d4	1	5	1	5	1	5	1	5	124
c81e728d9d4c2f636f067f89cc14862c	1	4	0.75	4	1	4	0.917	4	351
						Average	0.87	4.07	315

user id	section two - instance b								
	q1		q2		q3		accuracy	confidence	time spent (s)
	accuracy	confidence	accuracy	confidence	accuracy	confidence			
c4ca4238a0b923820dcc509a6f75849b	1	4	0.667	3	1	3	0.889	3.333	225
dd41c6da7897cd248698870811d89ed6	1	4	1	4	1	4	1	4	391
71c1e8a22dbe10daae0743e628bb991c	0	4	1	4	1	3	0.667	3.667	231
c6f057b86584942e415435ffb1fa93d4	1	4	1	5	1	4	1	4.333	114
c81e728d9d4c2f636f067f89cc14862c	1	5	1	4	1	4	1	4.333	221
						Average	0.91	3.93	236.4

user id	section two - instance a								
	q1		q2		q3		accuracy	confidence	time spent (s)
	accuracy	confidence	accuracy	confidence	accuracy	confidence			
99891000461f5b4e63817d95ab5749c4	0	3	0	3	1	4	0,333	3,333	468
d229a067f09c9a27d0f40eb02e91462c	1	4	1	4	1	3	1	3,667	483
eccbc87e4b5ce2fe28308fd9f2a7baf3	1	4	1	4	0	3	0,667	3,667	432
ec76ac6c764e20643208b88d111e3b24	1	4	0	4	1	4	0,667	4	404
381ea59648fd3e1fa64a05d567b43726	1	2	0	3	1	3	0,667	2,667	306
						Average	0,67	3,47	418,6

user id	section two - instance b								
	q1		q2		q3		accuracy	confidence	time spent (s)
	accuracy	confidence	accuracy	confidence	accuracy	confidence			
99891000461f5b4e63817d95ab5749c4	1	3	0	2	0	3	0,333	2,667	580
d229a067f09c9a27d0f40eb02e91462c	1	4	1	4	1	2	1	3,333	180
eccbc87e4b5ce2fe28308fd9f2a7baf3	1	4	1	4	0	0	0,667	2,667	277
ec76ac6c764e20643208b88d111e3b24	0	4	1	4	1	3	0,667	3,667	364
381ea59648fd3e1fa64a05d567b43726	1	3	1	3	0	3	0,667	3	372
						Average	0,67	3,07	354,6

user id	section one						
	q1		q2		accuracy	confidence	time spent
	accuracy	confidence	accuracy	confidence			
c4ca4238a0b923820dcc509a6f75849b	1	5	0,5	5	0,75	5	261
dd41c6da7897cd248698870811d89ed6	1	4	0	3	0,5	3,5	174
71c1e8a22dbe10daae0743e628bb991c	1	5	1	4	1	4,5	120
c6f057b86584942e415435ffb1fa93d4	1	5	1	5	1	5	75
c81e728d9d4c2f636f067f89cc14862c	1	4	1	5	1	4,5	133
99891000461f5b4e63817d95ab5749c4	1	4	0	2	0,5	3	272
d229a067f09c9a27d0f40eb02e91462c	0	3	1	4	0,5	3,5	343
eccbc87e4b5ce2fe28308fd9f2a7baf3	1	5	1	5	1	5	153
ec76ac6c764e20643208b88d111e3b24	1	5	1	5	1	5	348
381ea59648fd3e1fa64a05d567b43726	0	3	1	4	0,5	3,5	223
				Average	0,78	4,25	210,2

user id	visualization interpretability	visualization usability	components usability		
			oblique tree view		projection view
c4ca4238a0b923820dcc509a6f75849b	5	3	5	5	
dd41c6da7897cd248698870811d89ed6	4	4	4	4	4
71c1e8a22dbe10daae0743e628bb991c	5	3	5	5	3
c6f057b86584942e415435ffb1fa93d4	5	4	4	4	3
c81e728d9d4c2f636f067f89cc14862c	5	4	4	4	3
eccbc87e4b5ce2fe28308fd9f2a7baf3	4	5	5	2	2
ec76ac6c764e20643208b88d111e3b24	4	4	4	3	3
d229a067f09c9a27d0f40eb02e91462c	5	4	5	3	4
99891000461f5b4e63817d95ab5749c4	4	4	4	5	5
381ea59648fd3e1fa64a05d567b43726	4	3	4	3	3
Average	4,50	3,80	4,40	3,80	3,50