BaobabView: Interactive Construction and Analysis of Decision Trees

Stef van den Elzen* Jarke J. van Wijk†

Eindhoven University of Technology

ABSTRACT

We present a system for the interactive construction and analysis of decision trees that enables domain experts to bring in domain specific knowledge. We identify different user tasks and corresponding requirements, and develop a system incorporating a tight integration of visualization, interaction and algorithmic support. Domain experts are supported in growing, pruning, optimizing and analysing decision trees. Furthermore, we present a scalable decision tree visualization optimized for exploration. We show the effectiveness of our approach by applying the methods to two use cases. The first case illustrates the advantages of interactive construction, the second case demonstrates the effectiveness of analysis of decision trees and exploration of the structure of the data.

Index Terms: H.1.2 [Information Systems]: User/Machine Systems—Human Information Processing; H.2.8 [Database Management]: Database Applications—Data Mining; I.3.6 [Computer Graphics]: Methodology and Techniques—Interaction Techniques

1 INTRODUCTION

One of the main research areas in machine learning is classification, a supervised learning method. In a classification problem each item is defined by attribute values and a class label. The goal is to construct a model that predicts the target class label for an item given the attribute values. The model is constructed from a set of records, the training set, for which the attribute values and according class labels are known, such as historic data.

A decision tree is a model used for classification. It can be represented by a node-link diagram in which each internal node represents a test on an attribute, each link represents the according test value, or range of values, and each leaf node contains a class label. A decision tree is typically constructed by a recursive top-down divide-and-conquer algorithm [9, 32, 33]. The construction starts by considering the entire dataset, which is recursively partitioned into mutually exclusive subsets. The main decisions to be made during the construction are deciding on which attribute and test values to separate the dataset and when to stop splitting. The attribute choice is based on an impurity measure such as *Gini Gain* [9], *Information Gain* [33] and *Gain Ratio* [34].

A decision tree is sensitive to noise and outliers in the dataset, causing the model to overfit the data. Two methods are used to overcome this; stop growing the tree early [33] or fully grow the tree and then prune the tree by deleting subtrees [35]. Pruning is typically performed using a proportion of the data, the prune set, which is not used for growing the tree. After construction, the prune set is evaluated on the decision tree and nodes are deleted until accuracy of the prune set no longer increases.

A decision tree is typically evaluated on a separate test or evaluation set containing records for which the class labels are known. The records are evaluated on the constructed decision tree, probably leading to misclassifications. The accuracy of the decision tree is defined as the proportion of misclassifications on this test set. However, accuracy is not the only quality measure. For a decision tree to be understandable, its complexity should be low, which can be measured by the following metrics [36]: (1) the total number of nodes; (2) total number of leaves; (3) tree depth; (4) number of attributes used. Small trees containing few attributes are therefore preferred.

Traditionally a decision tree is created by setting parameters, running an algorithm and evaluating the constructed tree. The parameters are tweaked and the algorithm is run again. This process is repeated until users are satisfied with the constructed decision tree. Once the decision tree is constructed, it is used to classify data for which the class labels are unknown.

Often, users constructing a decision tree are very knowledgeable in their field, but possess little knowledge of decision tree construction algorithms. Therefore, users do not know the exact meaning of all parameters and their influence on the constructed decision tree. Furthermore, they have little to no knowledge about the algorithms' inner workings. Because of this lack of knowledge, the decision tree construction is often a trial-and-error process and may be very time-consuming. Even worse, domain experts are not enabled to apply their domain specific knowledge to optimize the decision tree, because the algorithm is used as a black box and does not allow them to steer it. Ankerst et al. [4] identify three important reasons to include domain knowledge and use visualization:

- by providing adequate data and knowledge visualizations, the pattern recognition capabilities of the human can be used to increase the effectiveness of decision tree construction;
- due to their active involvement, users have a deeper understanding of the resulting decision tree; and
- when obtaining intermediate results from the algorithm, users can provide domain knowledge to focus the further search of the algorithm. Using domain knowledge has been recognized as a promising approach for constraining knowledge discovery and for avoiding overfitting.

Liu and Salvendy [23] extend this list further; the interactive construction process: improves the effectiveness of modeling; enhances users understanding of the algorithm; and gives them greater satisfaction with the task.

Recently the inability to incorporate domain knowledge in automatic decision tree construction is acknowledged in industry, where it is believed that interactivity can close the gap [8].

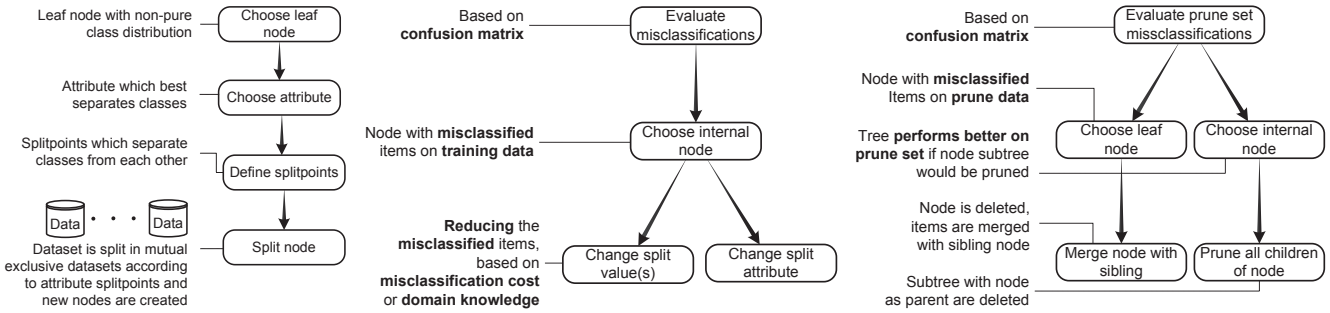
In this paper we present BaobabView, a tool for interactive construction and analysis of decision trees that enables domain experts to apply their domain specific knowledge. We think our tool provides a double example of a visual analytics approach. We show how a machine learning method can be enhanced using interaction and visualization; we also show how manual construction and analysis can be supported by algorithmic and automated support.

In Section 2 we identify user tasks and corresponding interaction, visualization and algorithmic requirements. Next we discuss related work in Section 3 and introduce our approach in Section 4. To show the effectiveness of our method, two use cases are discussed in Section 5. We conclude with a summary of our results and identify future work in Section 6.

*e-mail: s.j.v.d.elzen@tue.nl

†e-mail: vanwijk@win.tue.nl

IEEE Symposium on Visual Analytics Science and Technology
October 23 - 28, Providence, RI, USA
978-1-4673-0014-8/11/$26.00 ©2011 IEEE



(a) Grow. (b) Optimize. (c) Prune.

Figure 1: Steps of the (a) grow process; (b) optimize process; and (c) prune process.

2 User Tasks and Requirements

Users either want to *edit* the tree (grow, prune or optimize), *use* the tree (classification) or *analyse* it (data exploration). Typically, users often switch between the edit and analysis process. For each task we identify important elements and extract requirements. In our research, we mainly focus on the *edit* and *analysis* process.

2.1 Edit

The edit process can be divided in three subprocesses; **grow, prune** and **optimize**. New leaf nodes in the decision tree are created by splitting already existing leaf nodes. Pruning is achieved by deleting subtrees or by merging two sibling nodes into one node. Optimization is achieved by changing split attributes and tweaking split values.

Grow process In the grow process users construct a decision tree. First a leaf node is chosen based on the class distribution and node size. Next a split attribute is chosen based on an impurity measure or on the domain knowledge of the user. Splitpoints are defined such that misclassifications are minimized, again based on domain knowledge. Finally, the node is split and the process starts over (see Figure 1(a)).

Optimize process In the optimization process the decision tree is tweaked based on users' domain knowledge. Misclassifications on the training dataset are identified based on the confusion matrix [19]. Next the involved internal nodes are identified and either the split attribute is changed or the split values are adjusted such that the cost of misclassifications is minimized (see Figure 1(b)). Often misclassifications of one class are considered worse than misclassifications of another class. Users tweak the split points based on this domain knowledge. Also, acquiring values for one attribute can be more costly than for other attributes [12, 39, 21].

Prune process In the pruning process users generalize the constructed decision tree to prevent overfitting. First misclassifications on the prune set are evaluated with support of the confusion matrix. Next the user merges the involved node or deletes the entire subtree if this increases the accuracy on the prune set (see Figure 1(c)).

2.2 Analysis

Analysis on the decision tree is performed to gain insight. By having a thorough understanding of the decision tree users also gained insight in the underlying structure of their data. The analysis is furthermore useful to determine whether the decision tree is reliable or not, or to see whether classification of a certain class is likely to be correct or not; trees in which one class label is scattered among many leaves are likely to misclassify this particular class. Furthermore leaves that contain few records are also likely to overfit the data and / or misclassify it.

2.3 Requirements

We argue that a tight integration of visualization, interaction and algorithmic support is the key to a powerful system to support all different tasks. For each component, we define requirements in Table 1. A decision tree visualized using a node-link diagram with involved process elements is shown in Figure 2.

3 Related Work

First a short overview of the state of the art in decision tree visualization is given. Next interactive construction of decision trees is reviewed.

3.1 Decision Tree Visualization

The most important visualizations that potentially could be used to represent decision trees are the indentation diagram, node-link diagram, treemap, treering and icicle plot. Treemaps and treerings are never proposed in literature and will not be discussed.

In the **indentation diagram** each internal and leaf node are shown textually. The parent-child relation is conveyed by indenting the child with respect to the parent. Ankerst et al. [3], Do [11] and Poulet et al. [31, 29, 30] use indentation diagrams to visualize decision trees. While this diagram displays each of the decision tree components (internal nodes, leaves, split predicates and class labels), it is poor in conveying the overall structure of the decision tree. If the tree is deep and has many nodes, it is hard to see which nodes are on the same level. Furthermore it is difficult to see the number of leaves, and the context may get lost if users have to scroll because of limited screen space.

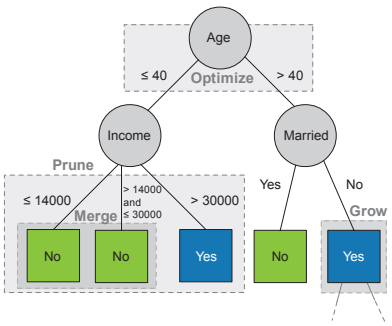


Figure 2: Decision tree node-link representation with annotated user tasks *grow*, *prune* and *optimize*.

Table 1: Requirements to support users in interactive decision tree construction tasks.

	Interaction *Users need to be enabled to*	**Visualization** *Show*	**Algorithmic support** *The system should*
Grow	choose leaf node	class distribution at each leaf node	suggest a leaf node based on class distribution
	choose split attribute	class distribution per attribute	suggest a split attribute based on impurity measure
	define split points	node size	suggest split points for the chosen attribute
	let system grow subtree		Automatically grow a subtree from a chosen node
Optimize	change split attribute	confusion matrix of training set	determine misclassifications on training set
	change split points	alternative attributes	
Prune	delete subtrees	confusion matrix of prune set	determine misclassifications on prune set
	merge two sibling nodes	prune set accuracy at each node	determine prune set accuracy at each node that would be the result of pruning the subtree
Analyse	highlight class(es)	whether a class is easy to separate from other classes	
	highlight attribute(s)	what attributes are involved in the separation process	
	choose alternative datasets	whether classes are difficult to separate from each other	
		effect on decision tree using alternative datasets	

The most well known tree visualization is the **node-link diagram**, used by Han and Cerone [17], Ware et al. [41] and Zhang et al. [44]. Internal and leaf nodes are represented by node glyphs and each parent-child relationship is represented by a link from parent to child node. In the node-link diagram, compared to the indentation diagram, it is easier to see the number of leafs and also to determine which nodes are on the same level. The split points are displayed on the links, the split attribute on the nodes and each leaf contains a class label (see Figure 2). Nguyen et al. [27] use a 2.5D variation on the node-link diagram. Nodes of interest are drawn on top of blurred nodes which are not of interest. However, none of the proposed methods show the full class distribution at each leaf, and also node size is not conveyed. Barlow and Neville [6] use multiple views. One view contains a node-link graph and another view shows an icicle plot. Wlodyka et al. [42] show the percentage of the majority class at each leaf, still no full class distribution is presented. Pham et al. [28] use a radial node-link diagram with fish-eye zoom interaction method to explore decision trees. None of the node-link variations integrate the data visualization with the tree visualization.

Ankerst et al. [4] and Liu and Salvendy [23] use **icicle plots** to visualize decision trees. The icicle plots naturally convey node size, which is a big advantage, and in both tree- and data visualization are integrated. Ankerst et al. use pixel-based bar charts [2]. This is a very compact data visualization and scales well, up to the point where one record is smaller than a pixel, however one cannot see if there are (large) gaps between classes in terms of attribute values. The class distribution can be derived from each bar if the user is capable of estimating the number of records from one class by adding all pixels of one color. This task becomes difficult if class values are scattered among the node and there is no clear separation. Furthermore, this pixel-based technique is mainly applicable to continuous valued attributes. Split values as well as split attributes are not shown in the tree visualization. Liu and Salvendy use mosaic plots [14] as data visualization, only targeted at categorical data. From these plots, the class distribution at each node can be derived. Also this visualization runs into scaling problems. From smaller nodes the mosaic plot is not visible any more. This problem gets worse as node sizes tend to get smaller and smaller, up to the point where a node glyph is smaller than a pixel.

Wang et al. [40] propose to visualize decision trees using a combined icicle plot and matrix view. Each of the individual items are displayed in a similarity matrix view below the icicle plot. From this visualization the node distribution can be derived however, gaps between attribute values are not displayed because records are placed in sequence independent of value. Furthermore, the icicle plot tree visualization does not scale well. Teoh and Ma [38] propose a variation on the icicle plot in which the current node of interest is shown as a large rectangle. The parent of each node is shown as a smaller rectangle left of each node and children are shown as smaller rectangles below the node. This tree visualization is integrated with data visualization by visualizing the individual records using either star- [26] or parallel coordinates plots, however, nodes topologically far away from the focus node are displayed too small to meaningfully convey the data visualization.

Xu et al. [43] propose to integrate decision trees with parallel coordinates plots, but due to the nature of parallel coordinates plots class distributions are not conveyed clearly, split attributes and split values are implicit.

Most proposed methods from the literature do not integrate the tree visualization with the data visualization. This has the advantage that both views can be optimized for their sole purpose and one view does not conflict with the other view or clutter it. However, this method also suffers from drawbacks. The first obvious drawback is the need to switch between the views, which may cause users to lose their context. Furthermore, only one node from the tree can be selected to show the data visualization for the records of that node. No overview is present in which both the overall tree structure is conveyed and the according data at the nodes. The few attempts in which the data visualization is well integrated with the tree visualization, are either aimed at continuous attributes or categorical attributes and, except for the method of Ankerst et al., do not seem to scale well.

Visualization of alternative datasets allows users to see the impact on the performance of the derived model [18], however, none of the described methods allow for this [25].

All current approaches are mainly targeted at supporting the grow process. As we argued earlier, the prune, optimize and analyse process are equally important. Some methods allow to prune the decision tree but no visual support is offered. To support users in all tasks, appropriate visualization techniques are developed and described in this paper.

3.2 Interactive Construction

Ankerst et al. [3] enable the interactive construction of decision trees via the use of circle segments [5]. Later, the circle segments were replaced by pixel-oriented bar charts [4] and the interaction was expanded in three ways: by letting the computer propose a split to the user, by letting the computer expanding a subtree and by automatically growing the tree. Y. Liu and Salvendy [23] provide similar interaction mechanics. However, no support for optimizing, pruning or analyzing the decision tree is provided. In Y. Liu and Salvendy [24] (an extension on [23]), users are enabled to automatically prune the decision tree after construction. However, the user is again not involved in the pruning process, making it impossible to incorporate domain knowledge. Han and Cerone [17] support the construction of decision trees in five ways: splitting a node (growing), labelling a leaf node, unlabelling a leaf node, deleting all

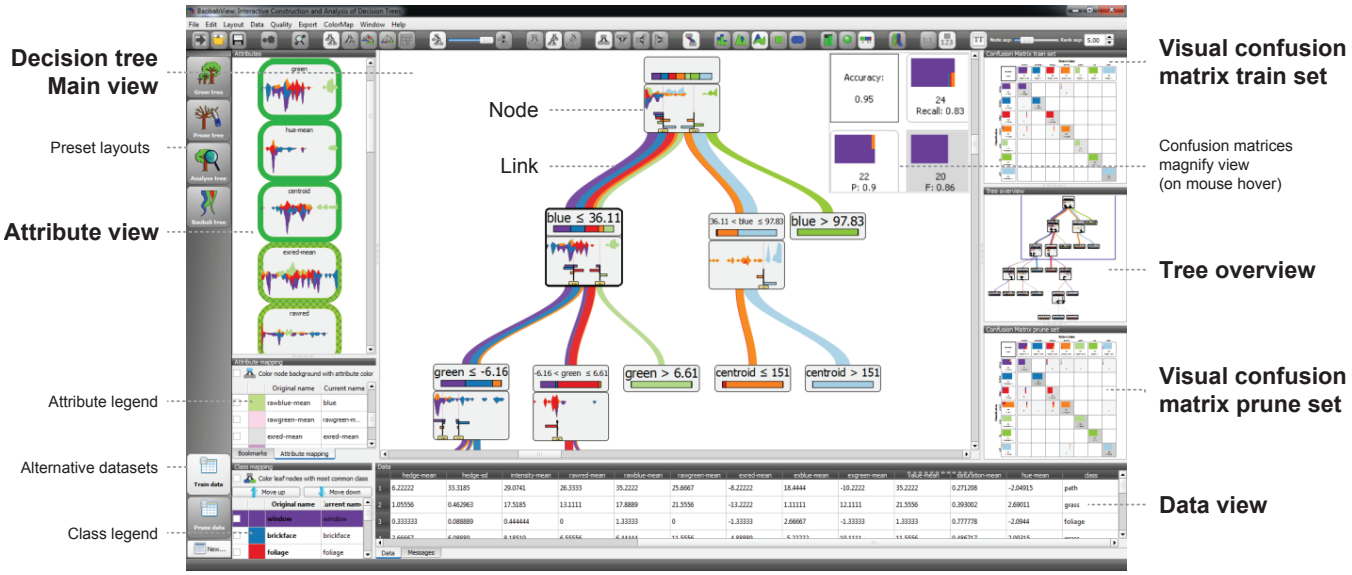


Figure 3: Interface of the interactive decision tree construction software with according proposed decision tree visualization; Based on adapted node-link diagram. Nodes contain important decision tree components. Links are visualized as a stream of items.

children of a node (pruning), and inspecting a node. However, this method is completely manual, no algorithmic support is provided.

Teoh and Ma [38, 26] allow for interactive construction of decision trees by painting an area of a star coordinate plot and assigning an according class label to it. This procedure is completely manual without algorithmic support, as well as the method of D. Liu et al. [22].

Ware et al. [41], Poulet [31, 29, 30] and Do [11] allow for bivariate decision trees by enabling users to respectively draw a split polygon and split line in a scatter plot visualization. In Ware et al. users are not algorithmically supported. Poulet and Do both support users algorithmically by suggesting the best split line.

Most systems partially meet the defined requirements on interaction, visualization and algorithmic support. However, no algorithmic support is provided to support the pruning process. In addition, algorithmic support on decision tree quality is limited and no system is capable of highlighting classes, highlighting attributes and visualizing alternative datasets to support the analysis process. Also dynamically changing split values or attributes to test alternative models is not supported, once a split attribute and value are chosen they can not be changed without first deleting all child nodes. We believe dynamically changing split points, attributes and datasets to see the effect on the constructed decision tree are essential in the grow, prune, optimize and analysis process.

4 BaobabView

In this section we present the solution we have developed to meet the requirements given and to improve on previous work. We named our tool BaobabView, because some of our tree visualizations (see Figures 10(b), 13 and 14) resemble the characteristic shape of the baobab, an African tree. Also, in ancient times kings, elders and leaders would hold meetings under huge baobabs to discuss important matters. The trees provided shelter, and they believed that the spirit of the baobab would always help them make wise decisions [1].

We argue that the key for a successful system is tight integration of visualization, interaction and algorithmic support. **Visualization** is needed to provide insight in *data, tree* and *classifier behavior*. **Interaction** is needed to *incorporate domain knowledge* in the decision tree, when choosing split attributes and split values, and pruning the tree. Furthermore, interaction is needed to *enable hypothesis testing*, by changing split attributes and values. Finally, **algorithmic support** is needed to *support users in construction* and *enable decision tree evaluation*.

In BaobabView users are enabled to manually construct a decision tree, automatically grow a selected node or import a complete decision tree, generated by the J48 algorithm of Weka [16] for exploration or optimization. Users are enabled to optimize and prune a decision tree. Exploration is supported in analysing both the decision tree and underlying structure of the data. Users are enabled to easily switch between tasks. For each task, users are supported by linked views, preset layouts and according interaction techniques. Figure 3 shows the complete user interface of BaobabView.

4.1 Decision Tree View

The first design decision concerns the visualization of the tree structure. Barlow and Neville [7] compared node-link diagrams, treemaps, tree-rings and icicle plots on their ease of identification of tree topology; ease of identification of node relationships; ease of identification of leaf sizes; and user preference. From this study the icicle plot and node-link diagram were the most favorable. The node-link diagram has the disadvantage that node size is not demonstrated, therefore an icicle plot would be preferred. However, because we want a tight integration of tree and data visualization, we need the data visualizations to be displayed on the nodes, which inevitable leads to aspect ratio problems when using an icicle plot; nodes deeper in the tree get smaller and smaller until they no longer convey the data visualizations in a meaningful manner.

By using a node-link diagram as opposed to the icicle plot we are given an extra component, namely the link, which we use to convey node size. The width of each link is set proportional to the number of items that is flowing from the parent node to the child node, e.g., if the incoming link at a node is thick then many items are contained in this node, if the incoming link is very thin then only a few items are present. To visualize the distribution of classes for a link between two nodes, the wide link is split into bands, each colored according to the class and given the according proportional width. To convey the link better and make it aesthetically more

[1] http://www.golimpopo.com/activity-detail_baobab-tree_15.html

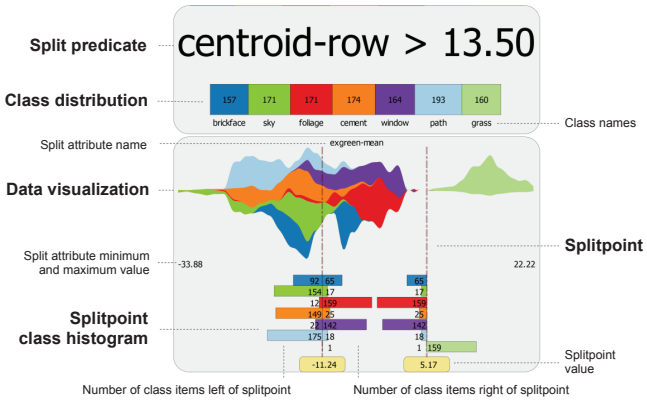


Figure 4: Proposed solution to visualize a node.

pleasing, the link is drawn as a Bézier curve. Finally, we choose for a top-down node-link diagram as opposed to a radial node-link diagram, because here the positions of the nodes have a semantic meaning, i.e., child nodes can be horizontally ordered on split predicates; child nodes containing records with lower values of the split attribute are positioned to the left of nodes with higher values of the split attribute.

As a default layout algorithm to position the nodes of the decision tree we use the Sugiyama algorithm [37] as part of the graphviz [13] implementation, which performs the steps described by Gansner et al. [15].

Nodes are visualized as rectangles, showing relevant information as stacked elements to users for performing their tasks (see Figure 4). The first two elements are relevant for all nodes, the others primarily for interior nodes. All elements can optionally be hidden, if an aspect is not relevant or if an overview is needed.

The **split predicate** is displayed using plain text. Because the predicates are based here on a single attribute, they are easy to understand and we do not need a sophisticated visualization metaphor.

The **class distribution** of the set of items at a node is visualized using a horizontal bar. Each class is given an area of the bar, proportional to the class quantity. This visualization has some nice properties. It is space-filling, and in our opinion, from the horizontal bar visualization it is easy to perceive if a node contains multiple classes or consists of only one class. This information is important to decide if we need to split the node further.

The attribute class values and distributions (**data visualization**) for a prospective split attribute are visualized using a Streamgraph [10]. Each class is visualized using a different colored stack in the Streamgraph. In our prototype we experimented with four other data visualizations: dot plots, boxplots, stacked histogram and smoothed stacked histogram (see Figure 5). We enable users

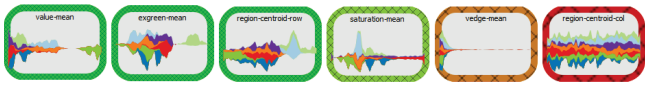


Figure 6: Ranked and sorted attributes based on gain ratio for the segment dataset from the UCI repository [1].

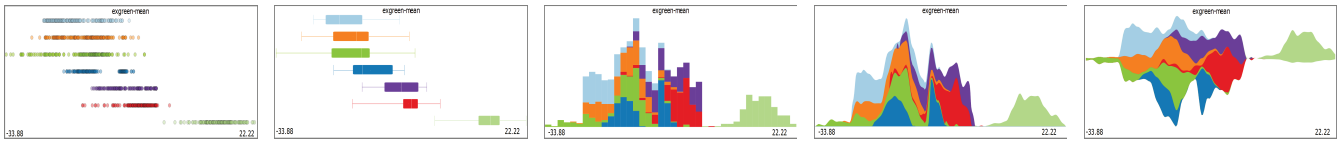
to visualize any combination of data visualizations, for example, a Streamgraph with overlayed dot plots to identify individual values. We found that the Streamgraph yields the most effective perception of class distribution and quantities. For nominal attributes, we present a stacked histogram to users in which each bar represents one enumerated value.

Splitpoints are visualized using a dashed line, the split value is displayed in a yellow box at the lower end of the splitline. The Streamgraph data visualization is not very precise, and while the relative data quantities can be perceived (colored areas), the absolute values are not present. We solve this problem by introducing (movable) splitpoints with **class histograms** at both sides of the splitpoint. The histogram bars are colored according to the classes. The value is displayed inside the bar, denoting the actual number of items at the left and right side of the splitpoint. The true quantities are perceived and the user is enabled to see the effect of the split by moving the splitpoint. Note that when the splitpoint is dragged, the sum of the number of elements in the left and right interval remains constant, and hence the total width of the histogram bars remains constant. This gives a stable and smooth impression; the user can drag the splitpoint and watch the bars move.

The **tree overview** provides a contextual overview of the main view and navigation support. Both the main view and the overview support zooming, panning and filtering (collapsing and expanding subtrees).

4.2 Attribute View

The attribute view shows the data visualization of each attribute for a selected node. The attributes are sorted based on a user selected impurity measure (Gini Gain [9], Information Gain [33], Gain Ratio [34]). As default impurity measure, Gain Ratio [34] is used, such that users can directly see the most appropriate candidate attributes for splitting the node. The gain ratio is a value between zero and one; a value of one is assigned if a class can be perfectly separated from the other classes, a value of zero is assigned if the attribute does not allow separating one class from the other classes. The Streamgraphs show the class distributions, and show users which classes can be split off. If we rank and sort the attributes horizontally with highest gain ratio value at the left and worst gain ratio value at the right end, we obtain a visualization as shown in Figure 6. Each attribute is given a thick colored border (user-defined colormap) with hatch pattern indicating the goodness of split for the according attribute; in the example, green and fine-



(a) Dot plots. (b) Boxplots. (c) Stacked histogram. (d) Smoothed histogram. (e) Streamgraph.

Figure 5: Different data visualization methods (individual or combination) provided to users. (a) Dot plots: Over-plotting is solved by transparency. Outliers can be detected. (b) Boxplots: Showing individual class distributions. However, area of the boxplots is misleading because it does not represent quantity. For example quantities for red and blue only differ 3 items. (c) Stacked histogram: Showing individual class distributions as well as quantities. Interpretation is difficult due to discontinuities in colors. (d) Smoothed histogram: Discontinuities are prevented for easier interpretation. (e) Streamgraph: Reducing wiggles and spikes. In our opinion, the Streamgraph is best suited to see individual class distributions as well as quantities.

grained hatch pattern indicates a gain ratio value of 1 whereas red and a coarse-grained hatch pattern represents a gain ratio value of 0. The sorted list in combination with the domain knowledge of users allows them to choose the appropriate attribute for splitting the nodes. Furthermore, users are enabled to inspect a table showing correlation values of all attribute pairs.

4.3 Visual Confusion Matrix

A decision tree can be evaluated on misclassifications by inspecting the confusion matrix. A confusion matrix M is an $n \times n$ matrix, where n is the number of classes. It displays the number of correct and incorrect classifications. The horizontal axis denotes the known class labels for each record, the vertical axis denotes the class labels as classified by the decision tree. On the intersection of a row i and a column j the number of classifications $M_{i,j}$ is displayed. The diagonal axis shows all correctly classified items. From this diagonal axis, we can calculate the accuracy of the tree which is displayed in the upper left corner of the matrix. Items that are off diagonal are misclassifications. Figure 7 shows our visual confusion matrix.

Each cell in the visual confusion matrix is given a varying grayscale color according to the quantity $M_{i,j}$. This color encoding allows one to quickly see if there are cells that contain many misclassified items. Each non-empty cell in the matrix contains two stacked rectangles. The width of the rectangles is proportional to $M_{i,j}$. The upper rectangle is colored to the known class label and the lower rectangle to the given class label (see Figure 8). If the item is misclassified then the stacked rectangles both contain colors from the given class and the known class, enabling users to see if a class is misclassified. Each item on the horizontal axis shows a summary of the according column by a rectangle made up of placing each item in the column as a sequence. This rectangle represents the recall [19] of a class. If the recall is high, then the rectangle is uniformly colored; if it is low, it is made up of many different class colors. Similarly, the vertical axis items represent a summary of each row, showing the precision [19] of a class. Finally, each diagonal item shows the harmonic mean of the precision and recall, known as the f-measure.

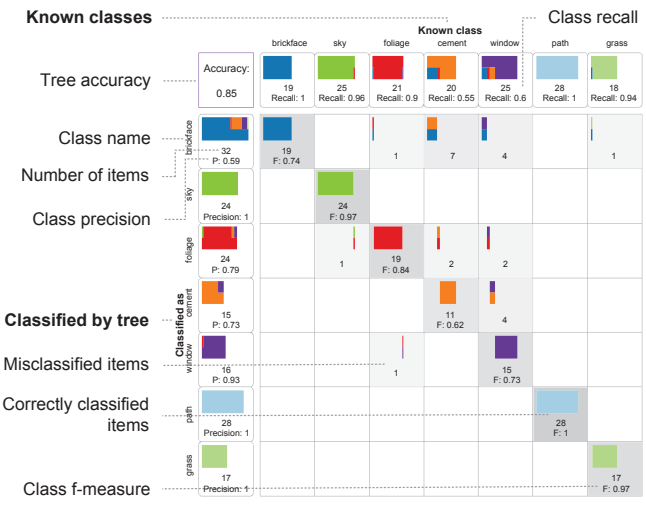


Figure 7: Visual confusion matrix.

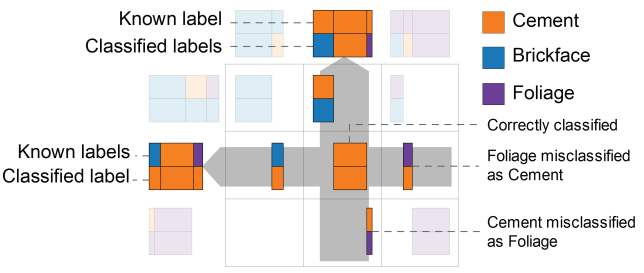


Figure 8: Visual confusion matrix cells explained.

The visual confusion matrices in the user interface (see Figure 3) show the classification results for both the train and prune set of the current decision tree and are linked to the main- and the data view. If the mouse is hovered over a cell in the confusion matrix the according nodes in the decision tree, which misclassify these items, are highlighted. If a cell of the confusion matrix is clicked, the actual misclassified records are highlighted in the data view.

4.4 Interaction

For each of the user tasks interaction techniques are implemented according to the requirements. Dependent on operations, different tree layouts are useful, which are discussed in the next sections. We implemented many options to control the layout of the tree and the visualization of the nodes, such that a wide variety of designs can be produced and evaluated. For each user task an according preset layout is defined. Users are enabled to easily switch between the different preset layouts. We use animation techniques to guide the user in the transition. Close-up inspection of the tree is facilitated via zoom-pan options and a separate navigation window.

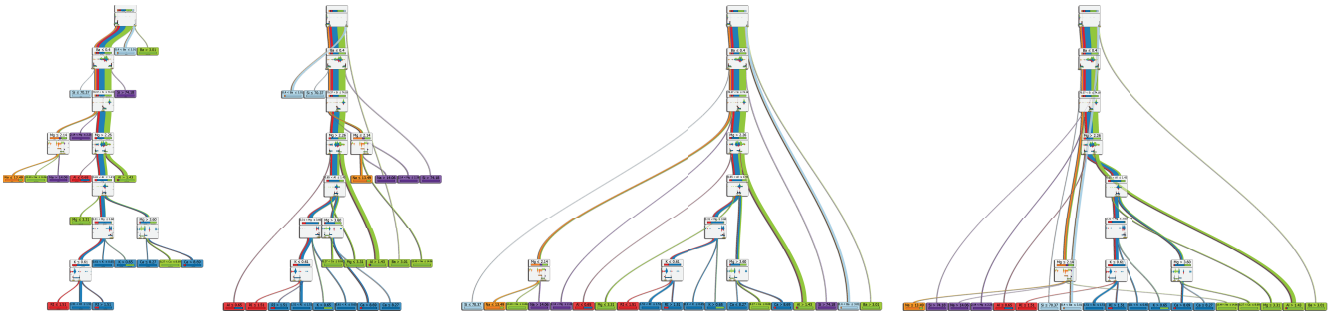
Grow Users are enabled to first choose a leaf node, next choose the split attribute and finally define split point values for this attribute. If splitpoints are added to a node, the splitpoints are positioned such that class separation according to gain ratio is as high as possible. The user is then enabled to fine tune the splitpoint value. We enable users to select a node and grow the tree automatically further if manual splitting is too challenging.

In the growing process users quickly need to identify leaf nodes that need to be split further, therefore they are enabled to lay out all leaf nodes on the same level. As an additional feature, users can see the leaf distributions by grouping leaf nodes based on majority class, or use a combination of both layouts. Figure 9 shows the default and optional layouts.

Prune At each internal node, the accuracy of the train and prune set that would be the result of pruning the subtree is computed. If the prune set accuracy of the entire tree would be better when the subtree of a node is pruned, the node is increased in size and colored (user-defined colormap) according to which node could be pruned best first, in line with the cost complexity pruning method [9]. We enable users to prune a subtree or to merge a leaf node with a sibling node. Furthermore, users are enabled to inspect outliers and noise through the inspection of dot plots at each node. By hovering over a dot, the according attribute values are highlighted in the data view.

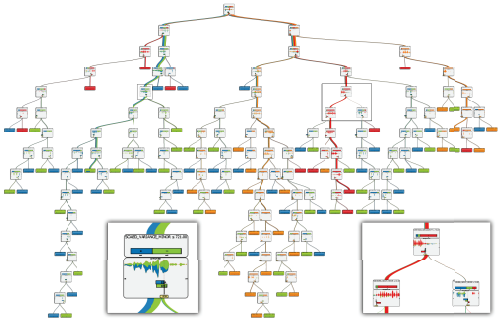
For the layout of the nodes in the decision tree, we adapted the Sugiyama algorithm such that edges with greater weight are drawn shorter and more straight. Each edge is given a weight according to the number of items. Nodes with fewer items, likely overfitting the data, are therefore pushed to the side of the tree and are easily identified.

Optimize Optimization of a decision tree is achieved by changing split attributes or split point values on any node in the decision tree. Nodes that are subject to optimization can be identified using both visual confusion matrices. Furthermore, users are enabled to use their domain knowledge to change the split attribute at a node or fine-tune the split value(s) such that misclassification cost is minimized. Also, users are enabled to apply domain knowledge in terms of attribute cost to optimize the decision tree. Costly attributes used

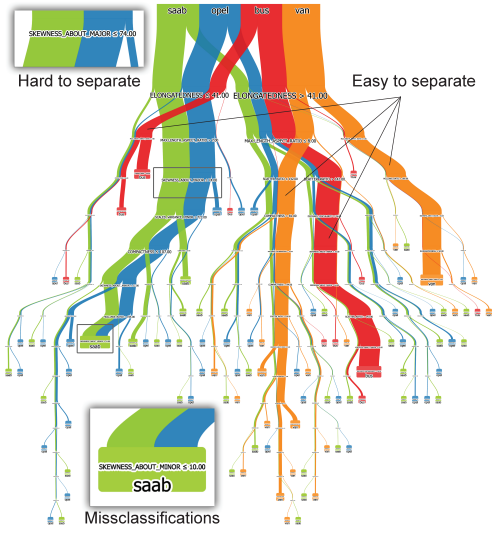


(a) Default layout. (b) Leaves grouped. (c) Leaves on same level. (d) Leaves grouped on same level.

Figure 9: Additional layouts providing users more insight. (a) Default grow process layout. (b) All leaves are grouped based on majority class, enabling users to see the deepest level of a class in the tree and the number of involved leaves. (c) All leaves on same level. Enabling users to identify which nodes need to be split further, based on the class distribution bars. (d) All leaves grouped on majority class and positioned on the same level. Providing users with insight on leaf distributions.



(a) Standard node-link diagram, as preset layout for grow process, showing data visualizations and class distributions at each node.



(b) Same tree as Figure 10(a) optimized for analysis. Note that, complementary to the standard node-link diagram, much insight becomes available by the use of color-banded links: bus and van class are easy to separate from the rest of classes. Opel and Saab class are difficult to separate. Opel is misclassified often as Saab using this decision tree (Blue stream ending in green node).

Figure 10: Tree layouts for editing and analysis processes.

in the decison tree can be changed to alternative attributes. All optimizations made to the decision tree are propagated to the involved subtrees which are updated in realtime. Additionally, all linked views such as the confusion matrices and attribute views are updated to reflect all changes. This enables users to directly see the effect of different hypotheses.

Finally, users can load different datasets and visualize them by the constructed decision tree. This visually shows the effect the decision tree has on different datasets and misclassifications, next to that, overfitting can be detected.

Analysis When analyzing the overall tree and data, users are mainly interested in the structure. Therefore, we put emphasis on the links and do not show the Streamgraph and splitpoints at each node in the analysis preset layout mode. Furthermore, the class distribution is not shown because it can be derived from the links. Additionally, we do not show the node glyphs but only the split predicate. Each link is drawn as a continuous stream of items, from root to leaf node. Figure 10 shows the decision tree preset layouts for the grow and analyse process.

For the preset layout, we use the adapted Sugiyama algorithm with weighted edges. The layer separation height depends on the number of items in the nodes. Links containing more items are given a greater height. In addition, the width of the links and nodes are proportional to their size. To optimize the readability and interpretation of the decision tree we minimize the number of crossing edges by sorting the class sequence at each node. The class ordering is determined by taking the weighted x-position of the leafs for one class and then sorting these.

Additionally we allow users to focus on interesting classes by highlighting of the involved paths. Important attributes used in the decision tree can be explored by giving each node the according split attribute color.

5 Use Cases

In the following sections, two case studies are presented that demonstrate the edit and analyse process. The first use case shows the interactive construction process, including growing, pruning and optimization. The second use case shows the effectiveness and scalability of the continuous color-banded links on complex decision trees for analysis purposes.

5.1 Interactive Construction

In this use case, we construct a decision tree for image segmentation data [1]. The instances, regions of 3×3 pixels, are drawn randomly from a database of 7 outdoor images. The images were next hand-segmented to create a classification for every pixel. The dataset

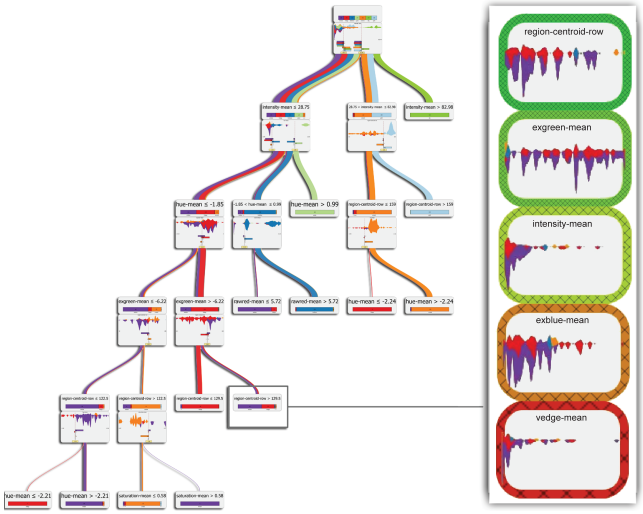


Figure 11: Decision tree after 10 splits. Highlighted node is difficult to split manually. Right image shows some Streamgraphs of the alternative attributes that can be used to split the node. Note there is no attribute that can be used to separate the involved classes easily.

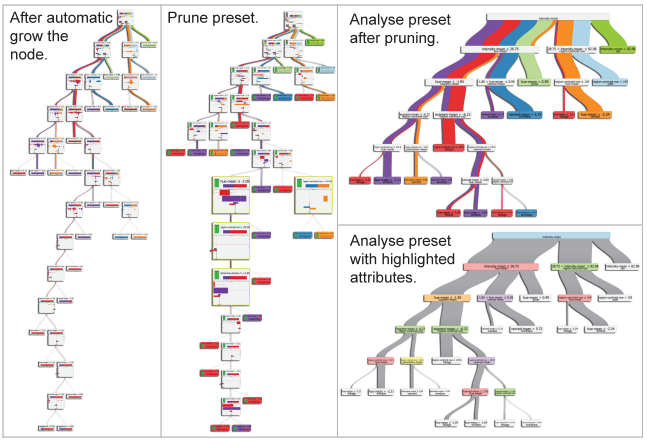


Figure 12: Decision tree in grow mode after automatic growing of hard to separate node (left). Decision tree in prune mode showing nodes bigger that can be pruned according to the prune set accuracy (middle). Final decision tree after pruning in analyse mode (upper right), also shown with nodes colored according to split attributes (lower right).

contains 2310 instances, 19 continuous attributes and 7 classes. We divide the dataset into 3 subsets; the first, 40 percent of total dataset, to use as train set in BaobabView; the second, 30 percent, to use as prune set; the third, 30 percent of total dataset, to evaluate the constructed decision tree.

Within a few minutes we make the first 10 splits, which are fairly straightforward choices. In the splitting process we choose for different attributes and splitpoints than presented to us by the system (the choices the classifier algorithm would take). Often the algorithm tries to split off only a few items, ignoring the main classes to be separated. For example in a node with three classes and distribution $D = \{2,28,25\}$ the algorithm would split off the class with two items, while we decide it is better to separate the classes with 28 and 25 items. Splitting of the two items does not contribute much to the decision tree and is likely to overfit it.

Next we are confronted with a node that does not lend itself for separation easily (see Figure 11). After inspection of the alternatives in the attribute view, we decide manual splitting this node further is too hard. We therefore choose to automatically grow this node further. Next we decide we are done growing the tree, there are no leaf nodes left that have a highly impure class distribution. Now we want to prune the tree based on the accuracy of the prune set. We select the prune mode and are presented with the nodes that have a higher or equal accuracy on the prune set, if this node's subtree would be pruned. We see that many of the automatically generated branches can be pruned. We prune the tree until there is no accuracy improvement on the prune set. Now we inspect the confusion matrices whether one class is misclassified a lot. This is not the case and we conclude we are done constructing the decision tree, presented in Figure 12.

Our resulting tree has 29 nodes in total and uses only 6 of the 19 attributes. Now we evaluate the constructed decision tree on the evaluation dataset. The accuracy of our constructed tree on the evaluation set is 93.2%. To compare our method to a traditional decision tree construction we construct a decision tree with the J48 algorithm of Weka (including pruning with default settings) on the same datasets and evaluate on the same evaluation dataset. The tree has an accuracy of 94.5% on the evaluation dataset, slightly better than our approach, however the constructed decision tree contains 75 nodes, more than 2.5 times our number of nodes. Furthermore, there are 12 involved attributes in the decision tree, twice as much as compared to our number of attributes used. Our tree is therefore easier to interpret and because we were actively involved in the construction process, we gained additional knowledge about the data. For example, we observed that the grass, path, cement and brickface classes are relatively easy to separate from the rest of the classes and that the foliage and window classes are difficult to separate from each other. Furthermore, we were enabled to use our domain knowledge by choosing appropriate attributes, tweaking splitpoints and decide which subtrees to prune.

5.2 Data Exploration

For the exploration use case we use medical data to determine the location of primary tumors. The data is provided by the Institute of Oncology of the University Medical Centre in Ljubljana, Yugoslavia [20] and obtained from the UCI repository [1]. The data consists of 22 classes and 17 attributes. We choose to import the decision tree, generated by the Weka J48 algorithm (Weka implementation of the C4.5 algorithm of Quinlan [33]), and use this to explore the underlying structure of the data. The imported tree consists of 154 nodes; 66 internal nodes and 88 leaf nodes.

We switch to analyse mode such that paths are visualized as continuous streams of items from root to leaf and no node glyphs are present, as described in Section 4.4 (see Figure 13). From the visualization we directly see that lung, breast and, head and neck are easy to distinguish from the rest because they lead to thick streams and do not split much. We see that rectum and stomach tumors are very similar and hard to separate, because the two streams are entangled. The same thing is true for pancreas and gallbladder tumors. Furthermore we notice from both the tree and the visual confusion matrix that stomach tumors are often misclassified as ovary tumors when using this decision tree, therefore extra care should be taken if ovary is determined.

Next we are interested whether there are any differences in diagnoses for male and female. We prune the entire tree and split the root node on the sex attribute. Subsequently we choose to automatically grow the decision tree from both the constructed male and female child nodes. We see (Figure 14) that head and neck, colon, testis, liver and prostate tumors are more diagnosed in males; females are diagnosed with breast, gallbladder, and ovary.

By highlighting individual class paths we observe that both sexes

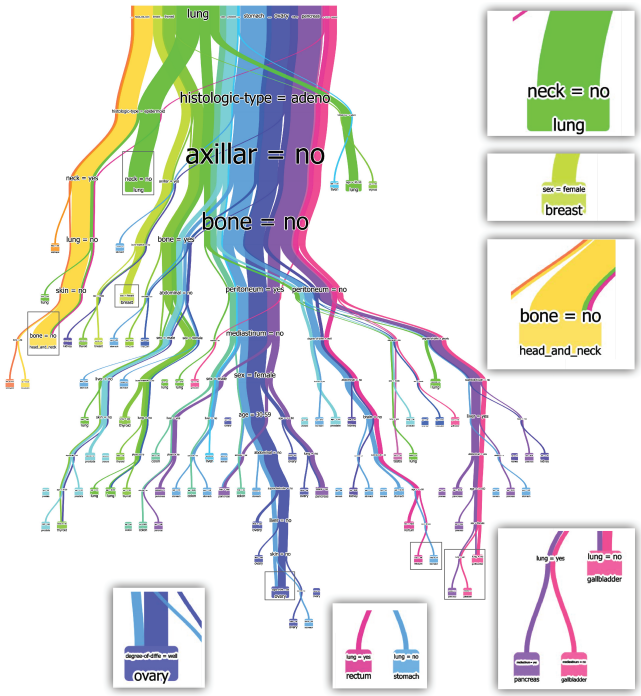


Figure 13: Primary tumor location decision tree imported from Weka, showing easy to separate classes (upper right), hard to separate classes (lower right) and misclassifications (lower left).

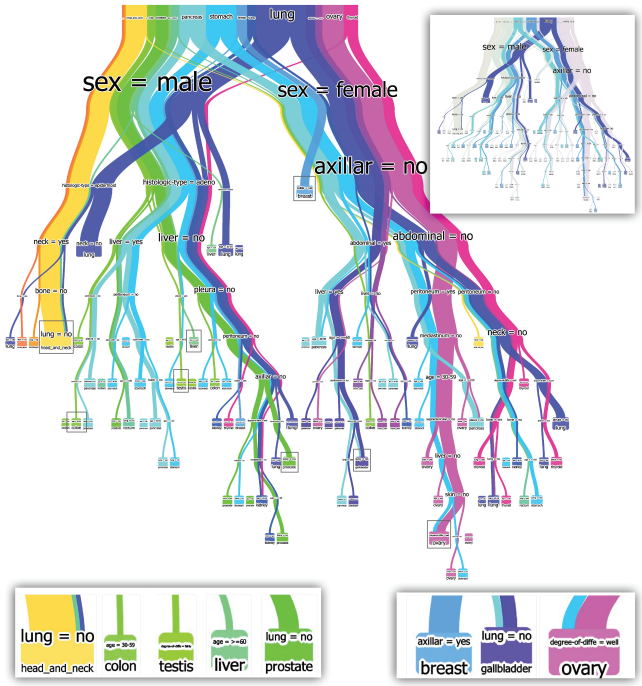


Figure 14: Primary tumor location decision tree constructed by manual first-split on sex, next automatically constructed. Showing typical diagnoses for males (lower left) diagnoses for females (lower right) and diagnoses of both sexes (upper right).

are diagnosed with pancreas, stomach and lung tumors. The paths of these classes split high in the tree and continue their paths to both the male and female branches.

6 Conclusions

Our aim was to develop a tool for the interactive construction and analysis of decision trees that supports all aspects of this process, and offers a combination of visualization and automated support, such that domain experts are enabled to bring in their domain knowledge and to perform their tasks effectively and efficiently.

We approached this as a design problem. Rather than inventing just one new visualization and building on top of that, we first analyzed the activities in detail, derived requirements from these, and next generated and selected solutions for all aspects. A simple but important lesson we learned was that different activities require different visualizations, each showing the most relevant information as clearly as possible. We used a multiple view approach, but also found that a parametrized visualization of the tree itself was useful and effective. This enables the user to select different layouts and different information to be shown, where smooth, animated transitions help to maintain situational awareness.

We think our approach is an advancement over previous solutions, especially concerning the span of supported tasks (construction, pruning, analysis), the tight integration of automated methods, and the rich variety of visualizations offered. As far as we know, none of the systems proposed in literature offers this set of features. As an example, the work of Ankerst et al. does not support users in the pruning and analysing tasks, does not allow for dynamic changes and visualization of alternative datasets and also, we think that pixel-based bar charts are less clear than Streamgraphs. We have built upon a number of existing methods, such as a node-link diagram to visualize the tree, Streamgraphs to visualize distributions, and the confusion matrix to show information. Our particular combination and integration is novel as such, but also we provide new solutions for a variety of issues. Specifically, we developed a flexible layout method for trees; visual annotations to the confusion matrix; a widget to enable the user to modify split points while giving detailed feedback on the effect of changes; and the use of color-banded edges with variable width to visualize the flow of the data through the tree.

Finally, we have shown the effectiveness of the tool for different types of use by presenting two use cases.

6.1 Future Work

In our research, we mainly focused on the *edit* and *analyse* user tasks. We think that interactive *use* of decision trees can be valuable and should be explored in future research, for instance to show users why certain classifications were made. We have shown our tool to three experts in decision trees, and they were very positive and appreciated the wide range of functionality offered and the clear visual feedback. However, the application should be tested more thoroughly by domain experts for real-world cases to identify the value in practice together with a formal user study evaluation. Setting up a controlled, quantitative experiment will not be easy, as we do not aim just for improved accuracy, but also hope to provide users with more insight and confidence in the results, which aspects are difficult to quantify. Finally, formal comparisons of our approach with other decision tree visualization methods should be conducted.

References

[1] D. N. A. Asuncion. UCI machine learning repository. http://www.ics.uci.edu/~mlearn/MLRepository.html, 2007.

[2] M. Ankerst. Visual data mining with pixel-oriented visualization techniques. Technical report, The Boeing Company P.O. Box 3707 MC 7L-70, Seattle, WA 98124, 2001.

[3] M. Ankerst, C. Elsen, M. Ester, and H.-P. Kriegel. Visual classification: an interactive approach to decision tree construction. In *KDD*

'99: Proceedings of the fifth ACM SIGKDD international conference on Knowledge discovery and data mining, pages 392–396, New York, NY, USA, 1999. ACM.

[4] M. Ankerst, M. Ester, and H.-P. Kriegel. Towards an effective cooperation of the user and the computer for classification. In *Proceedings of the sixth ACM SIGKDD international conference on Knowledge discovery and data mining*, KDD '00, pages 179–188, New York, NY, USA, 2000. ACM.

[5] M. Ankerst, D. A. Keim, and H.-P. Kriegel. Circle segments: A technique for visually exploring large multidimensional data sets. In *Proceedings on Visualization, Hot Topic Session, San Francisco, CA, United States*, 1996.

[6] T. Barlow and P. Neville. Case study: Visualization for decision tree analysis in data mining. In *Proceedings of the IEEE Symposium on Information Visualization 2001 (INFOVIS'01)*, pages 149–152, Washington, DC, USA, 2001. IEEE Computer Society.

[7] T. Barlow and P. Neville. A comparison of 2-d visualizations of hierarchies. In *IEEE Symposium on Information Visualization*, page 131, Los Alamitos, CA, USA, 2001. IEEE Computer Society.

[8] A. Blumenstock, M. Mueller, C. Lanquillon, S. Kempe, J. Hipp, and R. Wirth. Interactivity closes the gap: Lessons learned in an automotive industry application. In *Proceeding of the 2010 conference on Data Mining for Business Applications*, pages 17–34, Amsterdam, The Netherlands, 2010. IOS Press.

[9] L. Breiman, J. Friedman, R. Olshen, and C. Stone. *Classification and Regression Trees*. Wadsworth and Brooks, Monterey, CA, 1984.

[10] L. Byron and M. Wattenberg. Stacked Graphs – Geometry & Aesthetics. *IEEE Transactions on Visualization and Computer Graphics*, 14(6):1245–1252, 2008.

[11] T.-N. Do. Towards simple, easy to understand, and interactive decision tree algorithm. Technical report, College of Information Technology, Cantho University, 2007.

[12] P. Domingos. Metacost: a general method for making classifiers cost-sensitive. In *Proceedings of the fifth ACM SIGKDD international conference on Knowledge discovery and data mining*, KDD '99, pages 155–164, New York, NY, USA, 1999. ACM.

[13] J. Ellson, E. R. Gansner, E. Koutsofios, S. C. North, and G. Woodhull. Graphviz - Open Source Graph Drawing Tools. *Graph Drawing*, pages 483–484, 2001.

[14] M. Friendly. *Visualizing Categorical Data*. SAS Publishing, 1st edition, 2001.

[15] E. Gansner, E. Koutsofios, S. North, and K.-P. Vo. A technique for drawing directed graphs. *IEEE Transactions on Software Engineering*, 19(3):214 –230, March 1993.

[16] M. Hall, E. Frank, G. Holmes, B. Pfahringer, P. Reutemann, and I. H. Witten. The weka data mining software: an update. *SIGKDD Explor. Newsl.*, 11:10–18, November 2009.

[17] J. Han and N. Cercone. Interactive construction of decision trees. In *Proceedings of the 5th Pacific-Asia Conference on Knowledge Discovery and Data Mining*, PAKDD '01, pages 575–580, London, UK, 2001. Springer-Verlag.

[18] W. Johnston. *Model visualization*, pages 223–227. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 2002.

[19] R. Kohavi and F. Provost. Glossary of terms. *Mach. Learn.*, 30:271–274, February 1998.

[20] I. Kononenko, I. Bratko, and R. Roskar. Experiments in automatic learning of medical diagnostic rules. Technical report, Faculty of Electrical Engineering, E. Kardelj University, Ljubljana, 1984.

[21] C. Ling, V. Sheng, and Q. Yang. Test strategies for cost-sensitive decision trees. *IEEE Transactions on Knowledge and Data Engineering*, 18(8):1055–1067, August 2006.

[22] D. Liu, A. Sprague, and J. Gray. Polycluster: an interactive visualization approach to construct classification rules. In *Proceedings of International Conference on Machine Learning and Applications*, pages 280 – 287, December 2004.

[23] Y. Liu and G. Salvendy. Design and evaluation of visualization support to facilitate decision trees classification. *Int. J. Hum.-Comput. Stud.*, 65:95–110, February 2007.

[24] Y. Liu and G. Salvendy. Interactive visual decision tree classification. In *Proceedings of the 12th international conference on Human-computer interaction: interaction platforms and techniques*, HCI'07, pages 92–105, Berlin, Heidelberg, 2007. Springer-Verlag.

[25] Y. Liu and G. Salvendy. Visualization support to better comprehend and improve decision tree classification modelling process: a survey and appraisal. *Theoretical Issues in Ergonomics Science*, 8(1):63–92, 2007.

[26] K.-L. Ma and S. T. Teoh. Starclass: Interactive visual classification using star coordinates. In *Proceedings of the 3rd SIAM International Conference on Data Mining*, pages 178–185. The 3rd SIAM International Conference on Data Mining, 2003.

[27] T. Nguyen, T. Ho, and H. Shimodaira. Interactive visualization in mining large decision trees. *Knowledge Discovery and Data Mining. Current Issues and New Applications*, 1805:345–348, 2000.

[28] N.-K. Pham, T.-N. Do, F. Poulet, and A. Morin. Interactive exploration of decision tree results. In *International Symposium on Applied Stochastic Models and Data Analysis, ASMDA'07*, pages 152–160, La Cane, Grce, 2007.

[29] F. Poulet. High dimensional visual data classification. *Pixelization Paradigm*, 4370:25–34, 2007.

[30] F. Poulet. Visual data mining. *Towards Effective Visual Data Mining with Cooperative Approaches*, pages 389–406, 2008.

[31] F. Poulet and E. Recherche. Cooperation between automatic algorithms, interactive algorithms and visualization tools for visual data mining. In *Proceedings of Visual Data Mining @ ECML, the 2nd Int. Workshop on Visual Data Mining*, PAKDD '02, 2002.

[32] J. R. Quinlan. Induction of decision trees. *Mach. Learn.*, 1:81–106, March 1986.

[33] J. R. Quinlan. *C4.5: programs for machine learning*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 1993.

[34] J. R. Quinlan. Improved use of continuous attributes in c4.5. *J. Artif. Int. Res.*, 4:77–90, March 1996.

[35] J. R. Quinlan. Simplifying decision trees. *Int. J. Hum.-Comput. Stud.*, 51(2):497–510, 1999.

[36] L. Rokach and O. Maimon. Top-down induction of decision trees classifiers - a survey. *IEEE Transactions on Systems, Man, and Cybernetics, Part C: Applications and Reviews*, 35(4):476–487, November 2005.

[37] K. Sugiyama, S. Tagawa, and M. Toda. Methods for visual understanding of hierarchical system structures. *IEEE Transactions on Systems, Man and Cybernetics*, 11(2):109–125, February 1981.

[38] S. T. Teoh and K.-L. Ma. Paintingclass: interactive construction, visualization and exploration of decision trees. In *Proceedings of the ninth ACM SIGKDD international conference on Knowledge discovery and data mining*, KDD '03, pages 667–672, New York, NY, USA, 2003. ACM.

[39] P. Turney. Types of cost in inductive concept learning. 2000.

[40] J. Wang, B. Yu, and L. Gasser. Concept tree based clustering visualization with shaded similarity matrices. In *IEEE International Conference on Data Mining*, volume 0, page 697, Los Alamitos, CA, USA, 2002. IEEE Computer Society.

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

[42] A. Wlodyka, R. Mlynarski, G. Ilczuk, E. Pilat, and W. Kargul. Visualization of decision rules - from the cardiologists point of view. *Computers in Cardiology, 2008*, pages 645–648, September 2008.

[43] Y. Xu, W. Hong, N. Chen, X. Li, W. Liu, and T. Zhang. Parallel filter: A visual classifier based on parallel coordinates and multivariate data analysis. *Advanced Intelligent Computing Theories and Applications. With Aspects of Artificial Intelligence*, 4682:1172–1183, 2007.

[44] J. Zhang, L. Gruenwald, and M. Gertz. Vdm-rs: A visual data mining system for exploring and classifying remotely sensed images. *Comput. Geosci.*, 35:1827–1836, September 2009.