Working Paper Series
ISSN 1170-487X

Interactive Machine Learning—Letting Users Build Classifiers

by **Malcolm Ware, Eibe Frank, Geoffrey Holmes, Mark Hall and Ian H Witten**

Working Paper 00/4
March 2000

Department of Computer Science
The University of Waikato
Private Bag 3105
Hamilton, New Zealand

Interactive Machine Learning—Letting Users Build Classifiers

Malcolm Ware MFW4@CS.WAIKATO.AC.NZ
Eibe Frank EIBE@CS.WAIKATO.AC.NZ
Geoffrey Holmes GEOFF@CS.WAIKATO.AC.NZ
Mark Hall MHALL@CS.WAIKATO.AC.NZ
Ian H. Witten IHW@CS.WAIKATO.AC.NZ
Department of Computer Science, University of Waikato, Hamilton, New Zealand

Abstract

According to standard procedure, building a classifier is a fully automated process that follows data preparation by a domain expert. In contrast, *interactive* machine learning engages users in actually generating the classifier themselves. This offers a natural way of integrating background knowledge into the modeling stage—so long as interactive tools can be designed that support efficient and effective communication. This paper shows that appropriate techniques can empower users to create models that compete with classifiers built by state-of-the-art learning algorithms. It demonstrates that users—even users who are not domain experts—can often construct good classifiers, without any help from a learning algorithm, using a simple two-dimensional visual interface. Experiments demonstrate that, not surprisingly, success hinges on the domain: if a few attributes can support good predictions, users generate accurate classifiers, whereas domains with many high-order attribute interactions favor standard machine learning techniques. The future challenge is to achieve a symbiosis between human user and machine learning algorithm.

1. Introduction

Standard machine learning algorithms are non-interactive: they input training data and output a model. Usually, their behavior is controlled by parameters that let the user tweak the algorithm to match the properties of the domain—for example, the amount of noise in the data. In practice, even users familiar with how the algorithm works must resort to trial and error to find optimal parameter settings. Most users have limited understanding of the underlying techniques and this makes it even harder for them to apply learning schemes effectively.

The same problem arises when choosing a technique for a particular data mining problem at hand. The best choice generally depends on the properties of the domain, yet there is no standard recipe for selecting a suitable scheme. The problem is compounded by the fact that users are often unaware of the strengths and weaknesses of the individual learning schemes. Parameter and scheme selection are the only mechanisms through which users can affect how the model is generated, and (at least in propositional machine learning) there is no other way for domain knowledge to enter the inductive process beyond the data preparation stage.

This paper presents a graphical interactive approach to machine learning that makes the learning process explicit by visualizing the data and letting the user "draw" decision boundaries in a simple but flexible manner. Because the user controls every step of the inductive process, parameter and scheme selection are no longer required. When used by a domain expert, background knowledge is automatically exploited because the user is involved in every decision that leads to the induced model. The scheme works most naturally with numeric attributes, although the interface does accommodate nominal attributes.

The material presented in this paper builds on an earlier approach to interactive classifier construction by Ankerst et al. (1999), whose system lets users generate an univariate decision tree by placing split points on numeric attributes using an interactive visualization. There are two main differences with the work presented here. First, our system allows complex interactions between any pair of attributes to be captured using a two-dimensional visualization. Second, we present results of an empirical evaluation that in-

volves several novice users and standard datasets, enabling us to identify strengths and weaknesses of this interactive visual approach to machine learning.

The paper is organized as follows. The next section describes our method of interactive classifier construction; Section 3 gives a detailed example. Next we apply the system to the problem of classifying Kiwifruit and compare the resulting classifier's performance to a decision tree automatically generated by C4.5 (Quinlan, 1993). Section 5 presents an empirical comparison involving five benchmark datasets and classifiers constructed by five novice users. Section 6 discusses related work, while Section 7 summarizes the main results and future directions.

2. Visual Decision Tree Construction Using 2D Polygons

Flexibility and simplicity are the key elements in the design of our system. The goal is to make the process of building a classification model as intuitive as possible. The system enables the user to construct a decision tree graphically with bivariate splits on attributes (Lubinsky, 1994). Bivariate splits were chosen for three reasons. First, they are representationally more powerful than univariate splits. Second, it is difficult to construct splits visually on more than two attributes. Third, all users are familiar with two-dimensional data representation from drawing programs and graphical packages.

Each bivariate split is represented as a polygon or set of polygons. Polygons are easy to draw and can approximate arbitrarily complex two-dimensional shapes. In conjunction with the standard recursive "divide and conquer" decision tree procedure, they enable users to approximate the target concept to any degree of accuracy while minimizing the number of splits that must be generated to identify "pure" regions of the instance space.

Figure 1 illustrates the user interface. There are two kinds of panel: tree visualizers (Figure 1a, d, and f) and data visualizers (Figure 1b, c, and e). At the top of each screen is a selector that indicates which kind of panel is currently being displayed; users can click this to switch between panels at any stage of the construction process. The tree visualizer displays the structure of the decision tree in its current state: Figure 1a, d, and f shows trees with one, two, and three leaves respectively. The user can select any node by left-clicking on it, which highlights the node and loads the data at that node into the data visualizer. The data visualizer contains a two-dimensional visualization of the instances that reach the selected node: Figure 1b, c, and e show the data for the root node of Figure 1a, the root node of Figure 1d, and the right child of the root of Figure 1f respectively. The data visualizer allows the user to define a split by drawing polygons in the visualization. Once a split has been generated, the resulting nodes are appended to the tree structure in the tree visualizer.

2.1 Basic functionality

The data visualizer is divided into three areas: controls (at the top), a two-dimensional scatter plot (on the left), and one-dimensional bar graphs (on the right). The controls allow the user to select attributes and control other aspects of the display. The scatter plot displays the instances on a plane whose axes are defined by the two attributes currently selected by the user. The color of each data point indicates the class value of the instance corresponding to that point, and a key to the color coding, giving each attribute name in the color that represents it, is displayed below the scatter plot. (Three colors are used in Figure 1, and they appear as barely-distinguishable shades of gray; of course the actual color display is far more striking.)

The bar graphs, one for each attribute in the dataset, provide a compact one-dimensional visualization of each attribute in isolation. The array of bar graphs scrolls to accommodate more attributes than will fit in the space provided (although this is not necessary with the dataset of Figure 1). These bars provide a convenient way of visualizing the discriminatory power of individual attributes. The horizontal axis of an attribute's bar spans the range of the attribute it represents. Data points are randomly distributed along the short vertical axis to provide an indication of the distribution of class values at any given point in the attribute's range.

There are two ways in which the user can select attributes for display in the scatter plot. First, pull-down menus are provided in the control area at the top of the data visualizer that allow the user to choose the attribute for the X and Y axes by selecting the name of an attribute from the appropriate drop-down list. Second, attributes can be selected from the attribute bars displayed in the right area of the data visualizer: clicking on a bar with the left or right mouse button chooses that attribute for the scatter plot's X and Y axis respectively. Nominal attributes can be chosen; the different attribute values are displayed along the axis in a discrete manner.

Once the user is satisfied with their choice of attributes, a split can be drawn interactively in the scat-

ter plot area of the data visualizer. This is accomplished by enclosing data points within one or more polygons. A pull-down menu at the top of the panel lets the user choose from a list of shapes that can be drawn. The shapes range from a simple rectangle or polygon to a "polyline" or open-sided polygon (as shown in Figure 1c and e). They are drawn by left-clicking a series of points in the scatter plot. In the case of a polyline, a final click (with the right mouse button) on one side of the line determines which data points are enclosed; the end-points of the line segment at either end of the polyline are extended to infinity.

A split is defined by the area enclosed within the polygon that has been drawn, or the union of these areas if there is more than one polygon. When satisfied with the result, the user inserts it into the tree structure by clicking the *Submit* button at the top of the data visualizer. This appends two new nodes, the left containing all instances enclosed by the polygons, the right receiving all remaining instances. The modified tree can be viewed by switching to the tree visualizer. If, on the other hand, the user is not satisfied with the split they have drawn, it can be removed by clicking the *Clear* button.

The process of defining splits and appending them to the tree continues until the user is satisfied with the resulting classifier. At any stage the data at any given node in the tree can be visualized by left-clicking on that node. If the user decides to redefine a split at an existing interior node, the subtree below that node will be replaced by the nodes corresponding to the new split. The user also has the option of simply removing an existing subtree without defining a new split by right-clicking on a node in the tree visualizer.

2.2 Other features

Users can adjust how the tree structure is displayed in the tree visualizer. A right-click *outside* a node generates a pop-up menu from which one can select different options to rescale the tree. In addition, it is possible to move the tree by dragging it with the left mouse button.

The data visualizer offers additional options that alter the appearance of the data to accommodate preferences of individual users. The color assigned to each class can be changed using a pop-up color selector. The *jitter* slider is useful if several instances occupy exactly the same coordinates in the scatter plot. Depending on the level of jitter, all data points are randomly perturbed by a small amount.

The data visualizer also allows the user to examine properties of individual data points by left-clicking on any point in the scatter plot (so long as "select instance" is chosen in the shape-selection pull-down menu near the top—as it is in Figure 1b). This brings up a text window summarizing all attribute values for the instances (possibly more than one) located at that point in the plot.

3. An example

Here is a detailed walk through the process of building a decision tree for the well-known Iris data (Blake, Keogh & Merz, 1998). This dataset has a simple structure that lends itself naturally to interactive classifier construction. It consists of four numeric attributes that measure properties of Iris flowers. There are three classes, each representing a different variety of Iris.

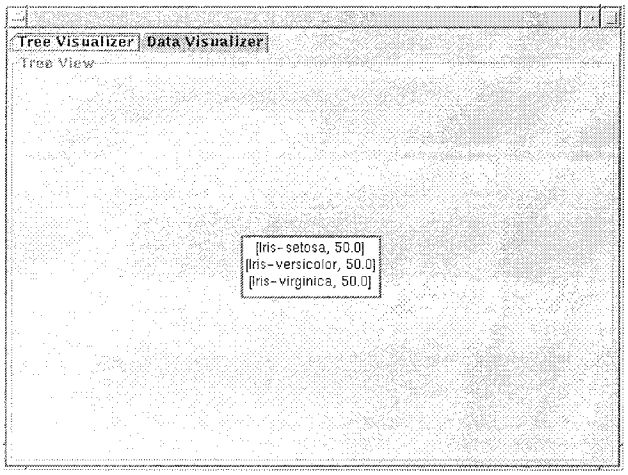
Before any splits are made, the tree visualizer displays a single node that corresponds to the root of the tree (Figure 1a). Inside the node is shown the number of instances belonging to it, broken down by class. In this case there are 50 instances of each class. The node is automatically selected: this is indicated by a highlighted border (cf. the borderless unselected nodes in Figure 1d and f).

To generate a split, the user switches to the data visualizer, which at this juncture displays the data points at the root node. Figure 1b shows the situation after the user has chosen the third and fourth attributes (*petallength* and *petalwidth*) for the X and Y axes respectively; both the selection controls and the attribute bars are updated accordingly.

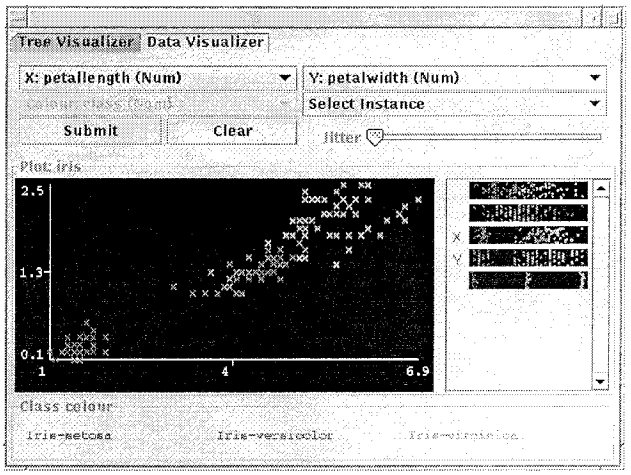
Next, the user draws a split in Figure 1c, in this case by choosing the polyline option to generate an open-sided polygon and splitting off the instances belonging to the Iris-setosa variety (located in the lower left corner of the display, and easily distinguished by color in the actual interface). The "enclosed" area of the polyline is shown in light gray.

Figure 1d shows how the tree is altered as a consequence of submitting the split. Two new nodes are attached to the root. The left one corresponds to the light gray area in the data visualizer, the right one to the remaining (black) region of the instance space. The right node is automatically highlighted for further processing, because users generally work by splitting off "easy" regions and leaving the rest for later refinement. The instances at this new node are automatically displayed in the data visualizer.

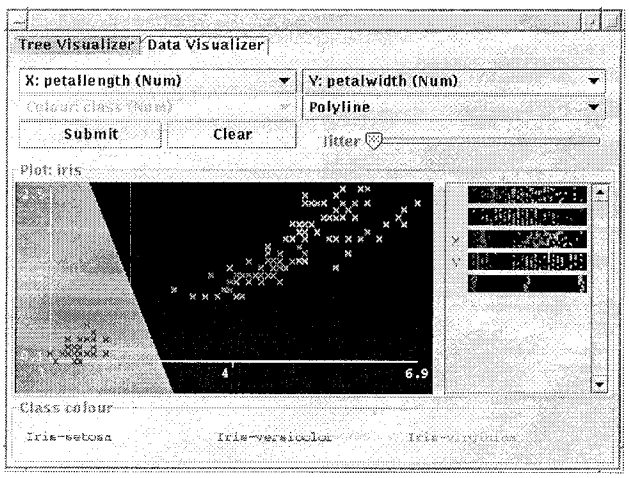
The illustration shows one further split being made, again using the polyline primitive, which divides the



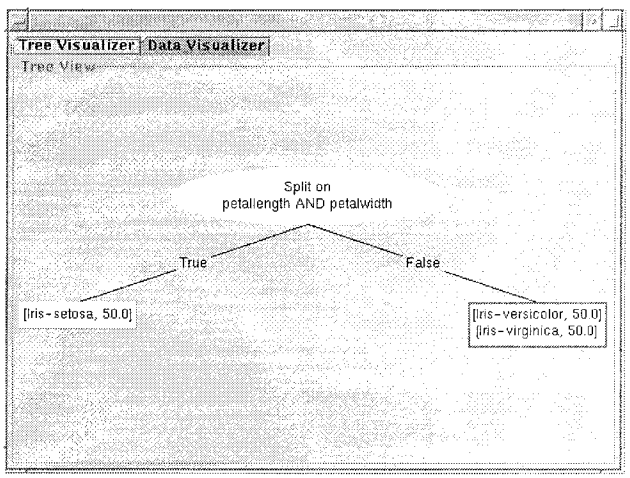
(a)



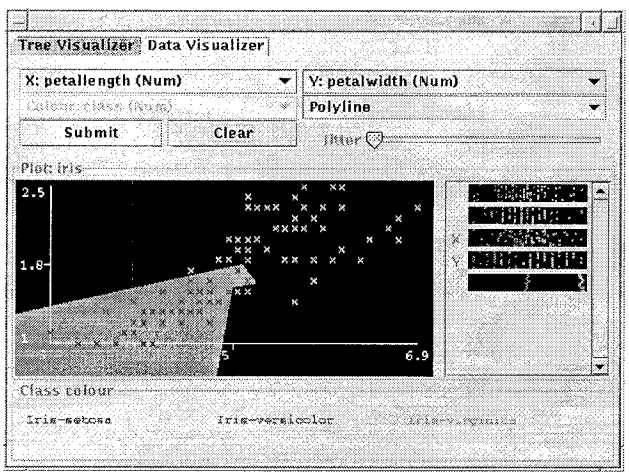
(b)



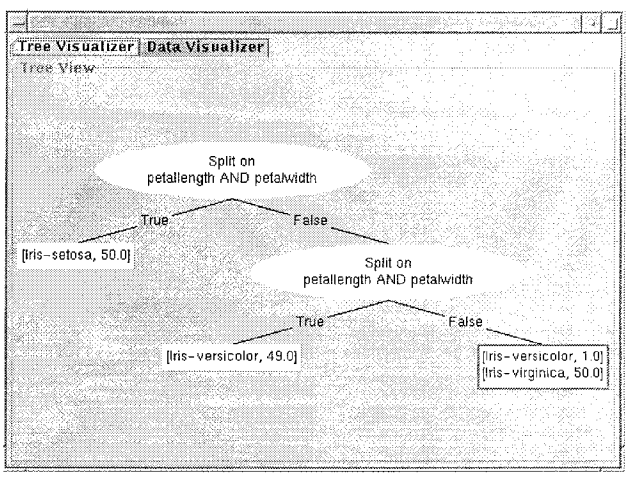
(c)



(d)

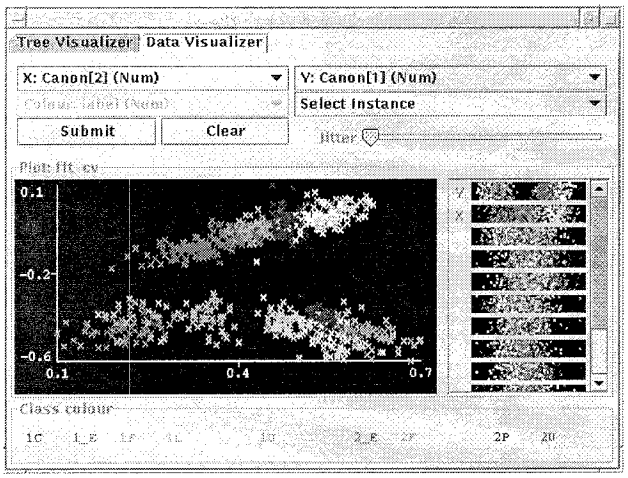


(e)

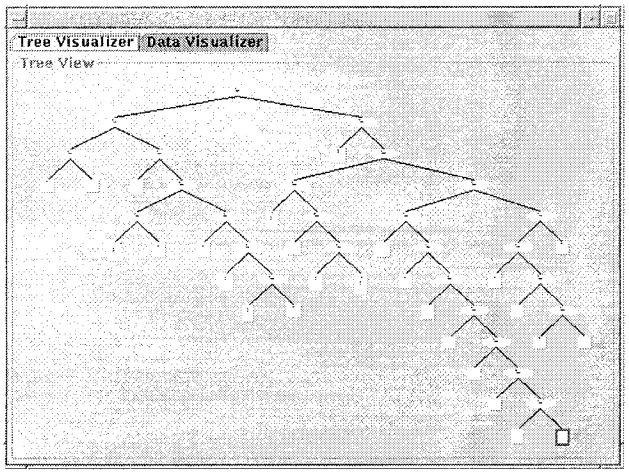


(f)

Figure 1. Constructing a classifier for the Iris data



(a)



(b)

Figure 2. Classifying Kiwifruit vines

remaining instances into two almost pure subsets in Figure 1e. The resulting decision tree is shown in Figure 1f. It contains a total of five nodes and classifies all but one of the training instances correctly.

4. Classifying Kiwifruit

Our interest in interactive machine learning derives from the observation that several datasets from our applied data mining projects appear to lend themselves naturally to manual classification. We first noticed this when displaying the datasets using the XGobi data visualization tool (Swayne, Cook & Buja, 1998) so that our clients could see what was going on.

A particular example of this type of problem involves classifying Kiwifruit vines into twelve classes. The task is to determine which pre-harvest fruit management

Table 1. Datasets.

Dataset	Train	Test	Attributes	Classes
waveform	500	4500	40	3
shuttle	43500	14500	9	7
segment	210	2100	19	7
sat	4435	2000	36	6
letter	15000	5000	16	26

treatment had been applied to the vines, on the basis of visible-NIR spectra collected at harvest and after storage (Kim, Mowat, Poole & Kasabov, 1999). The training and test data contain 879 instances and 929 instances respectively.

The training data, visualized using the first two attributes, is shown in Figure 2a. One author, with no prior knowledge of the domain and no previous attempts at generating a classifier for this problem, created a decision tree manually from the training data using the procedure described in the last section. The resulting tree contained 53 nodes and achieved an accuracy of 85.8% on the test data: Figure 2b shows a miniature view. For comparison, we ran the decision tree inducer C4.5 (revision 8) using the same training and test data. It produced a tree containing 93 nodes with an accuracy of 83.2% on the test data. The difference in accuracy is statistically significant at the 80% level according to a two-sided paired t-test.

This result encouraged us to perform a controlled experiment involving more subjects and standard benchmark datasets.

5. Experiments on Benchmark Datasets

In order to test whether users can, in general, construct accurate models, we performed an experiment using a selection of standard numeric-attribute datasets from the UCI repository (Blake, Keogh & Merz, 1998). The datasets were chosen to present subjects with a range of predictive tasks of varying difficulty, and to be large enough to obtain reliable accuracy estimates (since cross-validation would be tedious, to say the least!—and it would be impossible to prevent the user from transferring knowledge from one fold to another). Each dataset was divided into training and test sets. Table 1 summarizes the characteristics of the datasets.

Five novice users used our system to construct a model for each dataset. They were allowed to familiarize themselves with the software by practicing on the Iris dataset before building classifiers for the benchmark datasets. Table 2 shows the accuracy on the test set and the size of each decision tree produced by the

Table 2. Accuracy and size of the generated decision trees.

	Accuracy						Tree size					
Dataset	C4.5	A	B	C	D	E	C4.5	A	B	C	D	E
waveform	72.47	72.53	71.58	78.13	66.33	64.53	73	21	61	23	13	19
shuttle	99.95	99.83	99.95	99.97	99.93	99.76	53	19	29	27	23	29
segment	88.90	88.71	86.52	89.67	90.52	74.95	35	25	31	25	17	57
sat	85.45	78.10	78.90	83.85	80.80	81.95	431	37	71	39	23	241
letter	87.70	42.00	33.86	63.86	43.94	38.54	2105	121	73	171	79	137

users; it also contains corresponding figures for the trees generated by C4.5.

Table 2 shows that for three of the five datasets—*waveform*, *shuttle* and *segment*—users are able to equal or better C4.5's performance. For *sat*, three users were able to get within 5% of C4.5's performance. On the *letter* dataset, all users were roundly outperformed by C4.5, although user C put in a sterling effort and achieved nearly 64% accuracy.

The user-generated trees are almost always smaller than those produced by C4.5, the only exception being the tree that user E produced for the *segment* dataset. This is no surprise because bivariate splits are inherently more powerful than the univariate splits that C4.5 uses. Although univariate splits have the advantage of being easily interpretable when printed in textual form, bivariate splits can be analyzed simply by visualizing them—and our experience with clients is that they find such visualizations extremely enlightening.

The difference in size between the manually-generated trees and those produced by C4.5 is particularly striking for the *letter* dataset. It is clear from the magnitude of the difference that users were overwhelmed by the task of generating an accurate classifier for this domain. The problem is that the data cannot be separated into clearly defined clusters by looking at just two attributes at a time. High-dimensional attribute interactions need to be modeled to obtain an accurate classifier for this domain.

Table 3 provides insight into the complexity of the tasks from a user's perspective. It shows the time spent constructing models, in minutes for the users and minutes and seconds for C4.5. For the *letter* dataset, the problem is that two attribute dimensions are insufficient to separate off substantial numbers of instances of the same class. This leads to a seemingly endless interaction where the human desire to finish the task outweighs the need to build an accurate model.

Table 3. Time to construct tree (seconds for C4.5; minutes for users A, B, C, D and E).

Dataset	C4.5	A	B	C	D	E
waveform	0:11	28	34	18	20	11
shuttle	0:50	22	18	34	33	25
segment	0:03	22	10	15	30	24
sat	0:22	47	32	40	52	91
letter	0:47	107	67	207	182	56

6. Related Work

Ankerst et al. (1999) pioneered interactive decision tree construction. Their system, which they called "perception-based classification" (PBC), allows users to generate splits from a visualization of a dataset. The dataset is shown as a set of circle segments, one for each attribute, and concentric arcs are associated with attribute values. In other words, instances are sorted from the center of the circle to its outer boundary according to their value for that attribute; along the arc, instances are distributed randomly. The color of an instance indicates its class. The user can define univariate multiway splits on an attribute by choosing a circle segment and inserting arcs into it. As in our approach, the data is then divided into subsets using this split and corresponding nodes are added to the tree structure.

PBC is closely related to our approach to interactive machine learning. Instead of simple univariate splits we allow more complex bivariate splits, exploiting the user's ability to identify clusters in two-dimensional scatter plots. In our system, the functionality of the circle segments is achieved by the attribute bars on the right side of the data visualizer. In fact, it is not clear to us why PBC is based on circle segments instead of linear attribute bars: due to their shape, circle segments may give the user the misleading impression that there is a qualitative difference between instances close to the center of the circle and those close to its boundary.

Ankerst et al. (1999) present experimental results for PBC, obtained from a single user, for three of the five datasets employed in Section 5: *sat*, *segment*, and

shuttle. Unfortunately they state neither the user's experience nor how many trials were involved in obtaining the results. In their experiments, user-generated decision trees were less accurate than an implementation of C4.5 on *sat* and *segment*, and equally accurate on *shuttle*. However, the difference in accuracy was never large and the user-generated trees had the advantage of being much smaller.

The experimental results presented in Section 5 add two important new findings. First, manual classifier construction is not likely to be successful for large datasets with high-dimensional attribute interactions. Second, the accuracy of a user-generated classifier depends strongly on the person who produced it. We might expect even better results from professionally-motivated users who were intimately familiar with the datasets.

7. Conclusions

This paper presents a novel interactive method for constructing decision tree classifiers. The system is easy and intuitive to use. With it, people can build accurate classifiers after very little practice, shortly after encountering the system for the first time. An interactive approach to building classifiers allows users who are familiar with the data to exercise effective use of domain knowledge. It also has the advantage of demonstrating the inductive process in a concrete way, thereby educating users about what the results mean.

Experiments on standard numeric-attribute datasets involving several users show that for some datasets manually-constructed models can be smaller and as accurate as those produced by the decision tree inducer C4.5. Users build good models when clusters are visually apparent in two dimensions. For large datasets involving high-dimensional interactions, manual classifier construction is too tedious to be worthwhile.

The new challenge is to create a symbiotic relationship that combines the skills of human user and machine learning algorithm. Situations in which manual decision-tree construction will fail can be identified by visualizing the data, and in such cases the user may want to invoke a learning algorithm to take over the induction process. The latest version of our system has this capability: its empirical evaluation is next on our research agenda.

References

Ankerst, M., Elsen, C., Ester, M. & Kriegel, H.-P. (1999). Visual classification: An interactive approach to decision tree construction. In *Proceedings of the 5th International Conference on Knowledge Discovery and Data Mining* (pp. 392–397). ACM Press.

Blake, C., Keogh, E. & Merz, C. J. (1998). *UCI Repository of Machine Learning Data-Bases.* Irvine, CA: University of California, Department of Information and Computer Science. [http://www.ics.uci.edu/~mlearn/MLRepository.html].

Kim, J., Mowat, A., Poole, P. & Kasabov, N. (1999). Applications of connectionism to the classification of kiwifruit berries from visible-near infrared spectral data. In *Proceedings of the ICONIP99 International Workshop* (pp. 213–218). University of Otago.

Lubinsky, D. (1994). Classification trees with bivariate splits. *Applied Intelligence, 4*, 283–296.

Quinlan, J. R. (1993). *C4.5: Programs for Machine Learning.* San Mateo, CA: Morgan Kaufmann.

Swayne, D. F., Cook, D. & Buja, A. (1998). XGobi: Interactive dynamic data visualization in the X window system. *Computational Graphical Statistics, 7(1)*, 113–130.