Working Paper Series
ISSN 1170-487X

# A Development Environment for Predictive Modelling in Foods

by G Holmes and M Hall

Working Paper 00/9
July 2000


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

# A DEVELOPMENT ENVIRONMENT FOR PREDICTIVE MODELLING IN FOODS

G. Holmes[1] and M. A. Hall[2]

1, 2 Department of Computer Science, University of Waikato, PB 3105, Hamilton, New Zealand, {geoff,mhall}@cs.waikato.ac.nz

## ABSTRACT

WEKA (Waikato Environment for Knowledge Analysis) is a comprehensive suite of Java class libraries that implement many state-of-the-art machine learning/data mining algorithms. Non-programmers interact with the software via a user interface component called the Knowledge Explorer.

Applications constructed from the WEKA class libraries can be run on any computer with a web browsing capability, allowing users to apply machine learning techniques to their own data regardless of computer platform. This paper describes the user interface component of the WEKA system in reference to previous applications in the predictive modeling of foods.

## 1. INTRODUCTION

Inductive modelling is one of the tools that can be successfully employed in predictive modelling for agricultural applications (Holmes *et al*, 1998). With careful data preparation and sound experimental techniques it is possible to induce models with high predictive accuracy. If the model can be visualized (not all inductive techniques have this capability), then it is possible to gain valuable insights into the data such as the relative importance and relevance of attributes and the quantification of key values that "split" numeric attributes to give good differentiation within the data.

In the food sciences, most potential users of inductive modeling are computer literate non-programmers working on a variety of computer platforms who are comfortable with modern computer applications. These users need a tool that will assist them through all phases of the process of developing a sound model of their data. It is widely recognized that the key to success comes from users bringing their domain knowledge to the process of application development.

This process of developing an application is iterative and, in our experience, crucially dependent on using methods that are capable of producing an interpretable model of the data. Users require support for data cleansing and selection of a subset of attributes for model construction, since these are crucial phases of the data mining process. To illustrate the pitfalls that can be encountered in these stages, suppose, by way of an example, that two hundred instances of some kind of fruit have been grown on four different orchards. Environmental data (rainfall, sunshine hours, humidity etc.) has been recorded, and each fruit has been assigned a unique identifier in the range 1-200. Given environmental data, the task is to predict which orchard an instance of the fruit came from. If the identifier information is included in the data set then a smart algorithm might induce the following model:

If **fruit_number <= 50** Then **Class = Orchard_1**
ElseIf **fruit_number <= 100** Then **Class = Orchard_2**
ElseIf **fruit_number <=150** Then **Class = Orchard_3**
Else **Class = Orchard_4**

This model is 100% accurate but 0% useful—the model is simply noting that fruit from Orchard 1 were assigned identifiers in the range 1-50, fruit from Orchard 2 were assigned identifiers in the range 51-100, etc. If the model had been produced by a "black box" learner (such as a neural network), the naïve scientist might have been mislead. Interpretable output provides a reality check for induced models. The Knowledge Explorer reduces the risk of adopting misleading models by making it easy for a user to try a variety of learning paradigms on the same data and to visualize the resulting models. It also provides tools for the inclusion or exclusion of attributes or data values that are inappropriate and has mechanisms for discovering the most useful attributes to use in an application.

Model evaluation is another critical phase in application development. Typically, users view a model constructed from their entire data set, and cross-validation is the method of choice for evaluation. Error rate (100 – percentage correct) is used as the main means of comparing models on the assumption that classification accuracy is of primary concern.

The emerging standard in machine learning for estimating the error rate is to use stratified ten-fold cross-validation. Data is divided randomly into ten parts, in each of which the class is represented in roughly the same proportion as in the entire data set. Each of the ten parts is held out in turn while the learning scheme builds a model from the remaining nine parts. The holdout part is used for testing, and an error estimate is calculated. The ten error estimates are averaged to produce an overall estimate of error. The Knowledge Explorer uses this model of evaluation, thereby reducing the risk of end-users' drawing unsubstantiated conclusions. A separate utility (not described in this paper) called the experiment editor allows users to go beyond a single ten-fold cross-validation to obtain a better error estimate by reducing the random variation that can occur in choosing the parts. Although there is no consensus in the field, a common practice is to repeat this single ten-fold cross-validation ten times, giving rise to the construction and evaluation of one hundred models

This paper is organized as follows. The next section describes the WEKA Knowledge Explorer—in particular, its support for data preparation, the inclusion of domain knowledge and the application of a variety of inductive learning paradigms to the same task. This section also contains a case study of a mushroom data set to show how various aspects of the system are applied. Section 3 presents a novel interactive interface to the Knowledge Explorer that encourages users to build their own models of data. This section also contains a case study on a kiwifruit data set. Section 4 summarizes the contributions that the system makes to end-user development of predictive models and indicates future directions for the WEKA system.

## 2. THE WEKA KNOWLEDGE EXPLORER

The Knowledge Explorer (Figure 1) contains a number of state-of-the-art learning algorithms called *classifiers* (algorithms for both classification and regression), *meta-classifiers* that can improve the performance of the base classifiers, association rule learners, unsupervised learning methods (clustering) and a number of methods for pre-processing data called *filters*. Filters enable data to be processed at the instance and attribute value levels. Feature selection (six methods) and data visualization tools are also included in the interface. A summary of the Knowledge Explorer's capability is given in Table 1; for a thorough description of the system see (Witten and Frank, 1999).

The interface has the look and feel of a modern computer application and allows data to be entered from either a file (for example, a spreadsheet), URL or database connection. The interface has a number of "states" through which a user works, corresponding to phases in the process of developing a data mining application. The state shown in Figure 1 is *Preprocess*, where data is entered and then possibly modified by filters. The other states are indicated along the bar to the right of the Preprocess button.

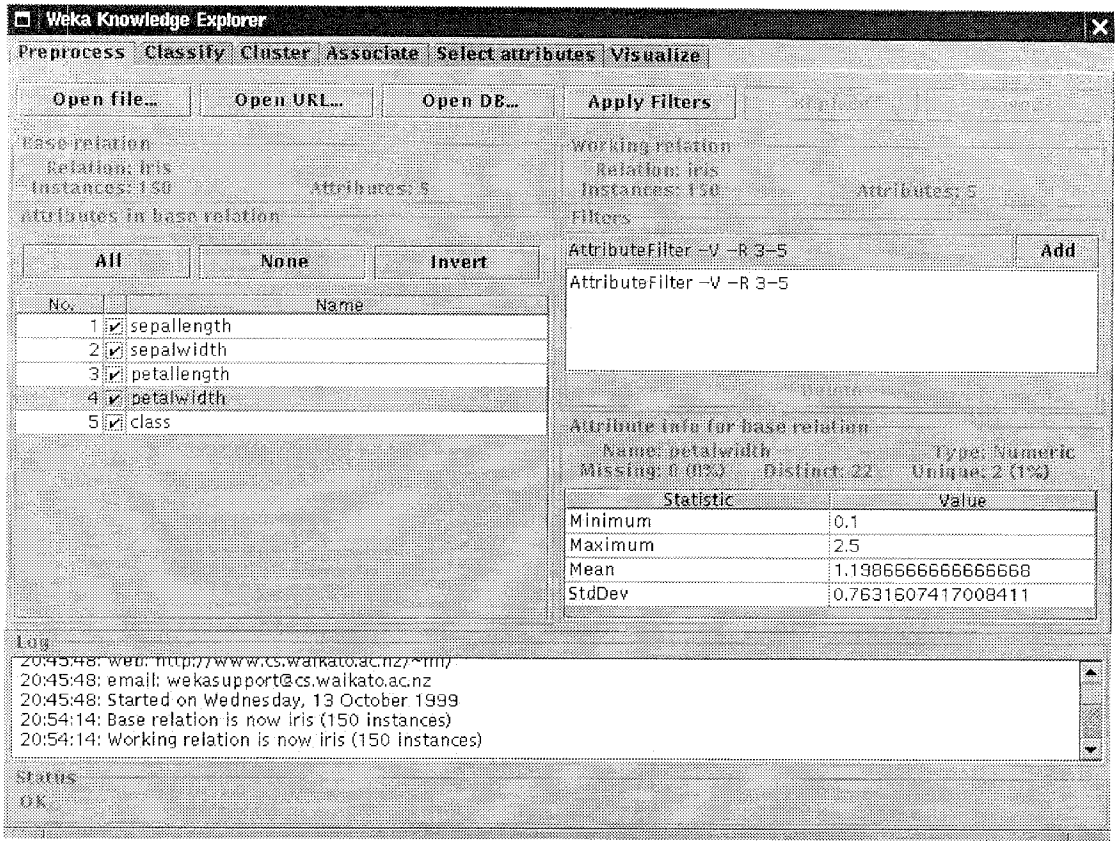



*Figure 1.* The WEKA Knowledge Explorer Interface

The *Preprocess* screen—where a dataset is loaded into the system—is the logical entry point for WEKA analysis. At this point the user can access simple statistics for each attribute (range, mean, and standard deviation) and apply simple transformations to attributes (for example, normalizing a column of values). Next, a user is likely to visit the *Visualize* screen, to explore the relative discriminatory power of different attributes through one- and two-dimensional plots of attribute values. The user can *Select* subsets of attributes to analyze with *Classify* (classification), *Cluster* (clustering), or *Associate* (association rule) model induction algorithms. Table 1 summarizes the facilities currently available in the Knowledge Explorer.

The analysis process is not necessarily linear—results from one induced model are likely to suggest further preprocessing transformations, users may experiment with more than one subset of attributes for induction, and more than one type of induction algorithm may be applied to the data set. The simple yet powerful interface encourages experimentation with analysis options. As its name implies, the Knowledge Explorer supports exploration of the implicit structure in a data set, guided by an evolving understanding of the data.

The following case study (Section 2.1) illustrates the support that the Knowledge Explorer offers in developing a data mining application.

| Learning Scheme | Meta Scheme | Filter |
|---|---|---|
| ZeroR | Bagging | Add attribute |
| OneR | AdaBoost | Delete attribute |
| Naïve Bayes | Logit Boost | Discretize attribute |
| Decision Tables | | Select attribute |
| Ibk (instance-based) | **Feature Selection** | Convert from nominal to binary |
| J4.8 (decision tree) | Wrappers | Merge values |
| Support Vector Machines | Relief | Replace missing values |
| Linear Regression | InfoGain | Swap attribute values |
| Locally Weighted Regression | Racing | Transform numeric values |
| Decision Stump | Correlation based | |
| Model Tree | Principal Components | |

*Table 1.* Summary of Knowledge Explorer capability

## 2.1 CASE STUDY 1 – MUSHROOM DATA

The marketing of mushrooms is a highly subjective exercise. In New Zealand there are no detailed grading specifications for marketing fresh mushrooms. The product tends to be graded based on the experience of farmers. These "experts" gain most of their external feedback on quality matters from the wholesaler or auction floor. The real requirements of consumers are therefore three or four steps removed from the initial quality decisions which must be made at the production end of the supply chain.

This case study uses the Knowledge Explorer to try to undercover the criteria used by consumers in reaching a decision on quality. Over a four week period a group of 10 consumers was asked to segregate groups of mushrooms into three separate categories based on their understanding of "good" and "bad" mushrooms. At each assessment the consumers were presented with a set of 60 mushrooms.

A range of quality was created by presenting each consumer with a set of mushrooms made up of freshly purchased and stored product. On the day of each assessment a sample of mushrooms was purchased at a local retail outlet. This sample was divided into six subsets. Two subsets were used for immediate assessment, while two were placed in a coolstore at 10°C and two placed in a 2°C store. At the next assessment time (four days later) one of the 2°C subsets and one of the 10°C subsets were used. The second pair of subsets were used eight days after they had been purchased.

The mushrooms were then assessed for several objective measures of quality. First a set of digital images was captured (Kusabs et. al., 2000) using a lighting chamber with a 22w annular florescent light. The mushroom was viewed against a matt black background. The images were captured using a Kodak DC120 digital camera as 1280 x 960 pixel bitmaps. These images represent each pixel in the RGB (red, green, blue) colour space with intensities of 0 to 255. This image data was also transformed into the HSV (hue, saturation, value) colour space according to Tao et. al. (1995). This colour space is supposed to most closely replicate the human perception of colour.

For the machine learning analysis the image had to be summarized as a smaller number of attributes. The histograms were normalized to remove size effects so that they describe relative intensity. For the normalized image the background was removed and the each of the six histograms (R, G, B, H, S and V) for the remaining image were averaged into five bins. Each bin was used as an attribute for the machine learning scheme. With both top and bottom views of the mushroom this produced a total of 60 visual attributes. The weight of the

mushrooms and their firmness were also included as attributes. A total of 1320 mushrooms were evaluated.

The 62 quality attributes were then analysed using J4.8, from the *Classify* panel of the Knowledge Explorer. J4.8 is a Java implementation (Witten and Frank, 1999) of the C4.5 learning scheme (Quinlan, 1993) which builds a model of the data as a decision tree. Stratified ten-fold cross validation was used to estimate a measure for the likely future performance of the model. A separate model was produced for each of the ten consumers, and all data for all consumers was then combined to generate a global consumer model.

Two machine learning techniques were used to enhance the classification—feature selection and boosting. Feature selection estimates the contribution of the different quality attributes to determining the grade, by testing the effects of addition and removal of individual attributes. This enables the identification of the most significant and influential attributes and combinations, so that extraneous or marginal attributes can be de-selected from model construction.

Feature selection is particularly important when dealing with large numbers of attributes. The Knowledge Explorer encourages an experimental approach by providing six distinct state-of-the-art approaches to the task, each of which can be selected for comparison with relative ease.

In this case study, two variants of the "wrapper" (Kohavi and John, 1995) feature selection technique were used. The first, standard wrapper, uses a full run of five fold cross-validation to estimate the impact of adding and removing attributes. The second method, called "racing" (Moore and Lee, 1994), reduces the computation required by evaluating subsets of features in parallel. If a subset is deemed unlikely to have the lowest estimated error, at some stage during a cross validation, then it is dropped from the race. Similarly, subsets that generate near identical classifications are also discarded. This has the effect of reducing the percentage of training examples used during the evaluation and reduces computational time. The relative performance of the two techniques (standard wrapper and raced wrapper) can be seen in Table 2.

The second method used to improve the classification accuracy of the induced models was a "boosting" technique. Boosting builds a series of models in a sequence, such that each new model in the sequence is designed to improve on the errors made by its predecessor. It achieves this by initially weighting the data equally when building and testing a model and then re-weighting specific data instances to focus attention on those instances that were previously incorrectly classified. A new model is constructed based on these re-weighted values and the process continues, typically for a fixed number of iterations. Instances are classified by majority vote. Each model decides on a classification and a vote is taken.

In this study, fifty models were built and classified by Adaboost (Freund and Schapire, 1996). While the technique improves prediction and reduces misclassifications, it has the disadvantage that it does not produce a single interpretable decision tree. In this study, boosting resulted in marginal to moderate increases in classification accuracy—approximately 0.5% to 11.2% (Table 2).

In order to compare the perceptions of quality at the producer and consumer ends of the supply chain, both producers and consumers must assess the same mushrooms. A producer model was generated in a previous study (Kusabs et. al., 2000), also using the Knowledge Explorer). The mushrooms were graded by the ten consumers and the induced producer model.

The degree of similarity between consumers was assessed by calculating a correlation matrix. Each consumer was compared with each of the other nine consumers, the 'global' consumer (an combination of all the consumers) and the producer model. The highest agreement between consumers was approximately 70% and the lowest 51%. Agreement with the global consumer ranged from 80% to 67% which indicates that these consumers relate better to an average of the whole group, than they do to any individual consumer.

| Consumer | ZeroR | Standard Wrapper Accuracy | Raced Wrapper Accuracy | Boosted Accuracy | Features selected (by racing) |
|---|---|---|---|---|---|
| 1 | 42.2% | 63.2% | 63.1% | 67.3% | Rt4, Gb1, Hb3, St3, Sb3, Vt2 |
| 2 | 49.7% | 66.9% | 63.1% | 68.2% | Ht4, St1, Vt1 |
| 3 | 40.1% | 62.5% | 57.5% | 68.7% | Gb1, Bt3, Ht0, Hb0, Hb4, Vt4 |
| 4 | 58.9% | 71.0% | 71.4% | 74.9% | Bt1, Bb0, Bb2, Ht2, St1, St2 |
| 5 | 55.1% | 77.1% | 75.1% | 80.1% | Gt0, Bb4, Hb4, St1, St3, Vt2 |
| 6 | 46.5% | 69.4% | 64.0% | 71.0% | Wt, Gt2, Gb0, Gb3, Bt4, Bb2 |
| 7 | 62.6% | 67.5% | 67.5% | 70.3% | Rt2, Rb0, Rb3, Bt1, Bt2, Ht2 |
| 8 | 44.6% | 63.3% | 63.9% | 64.4% | Gt2, Gb3, St1, Sb4 |
| 9 | 38.0% | 61.8% | 59.7% | 64.9% | Wt, Rt2, Hb1, Sb2 |
| 10 | 35.5% | 69.9% | 69.8% | 70.9% | Rb0, Rb1 |
| Global | | 70.1% | 69.0% | | Rt3, Ht0 |
| Producer | | 82.1% | | | Rt1, Rb1, Bb1, Sb1, Vt3 |

*Table 2.* Accuracy of different explorer techniques for consumer grading decisions

The zeroR accuracy figure in Table 2 is produced by choosing the majority class for a given data set. It is a good estimate of "default accuracy"; a learning scheme should provide significant improvement on this value.

Using the racing wrapper, user model accuracy ranged from 60% to 75%. The producer model accuracy, when grading mushrooms in the previous study, was 82% (Kusabs et. al., 2000). This suggests, as expected, that the consumers' grading criteria are not as well defined as the producer.

Only two of the consumers (6 and 9) used weight as a major determinant. One consumer (2) used only visual attributes relating to the top of the mushroom, and consumer 10 used only the bottom of the mushroom. The attributes used by the producer are also shown in Table 2. It is noteworthy that of the attributes used by the producer only one is used by a (single) consumer. This was consumer 10, who graded entirely using the bottom view of the mushrooms. Visual attributes dominate in the consumer decision, with only two consumers using weight as a primary attribute.

The J4.8 machine learning scheme produced a useful set of quality description models for the 10 consumers. These models have easily interpretable decision trees which are built on the objective attributes measured. An analysis of the consumer decision trees show that different consumers use different combinations of attributes to classify their mushrooms (final column of Table 2). A full tree for consumer 10 after racing feature selection has been applied is shown in Figure 2. The tree is interpreted as follows: grade 1 mushrooms have Rb1 <= 498; grade 2 mushrooms have either Rb1 > 498 and Rb0 <= 990 or (908 < Rb1 <= 2005) and Rb0 <= 1028; while grade 3 mushrooms have Rb1 > 908 and Rb0 > 1028 or Rb1 > 2005 and Rb0 <= 1028 or 498 < Rb1 <= 908 and Rb0 > 990. The values in parentheses at the leaves of the tree give an estimate of how well the rule leading to that leaf will differentiate as yet unclassified examples. For example, the grade 1 leaf has correctly classified 402 of the 511 (402 + 109) grade 1 mushroom examples in the training data.

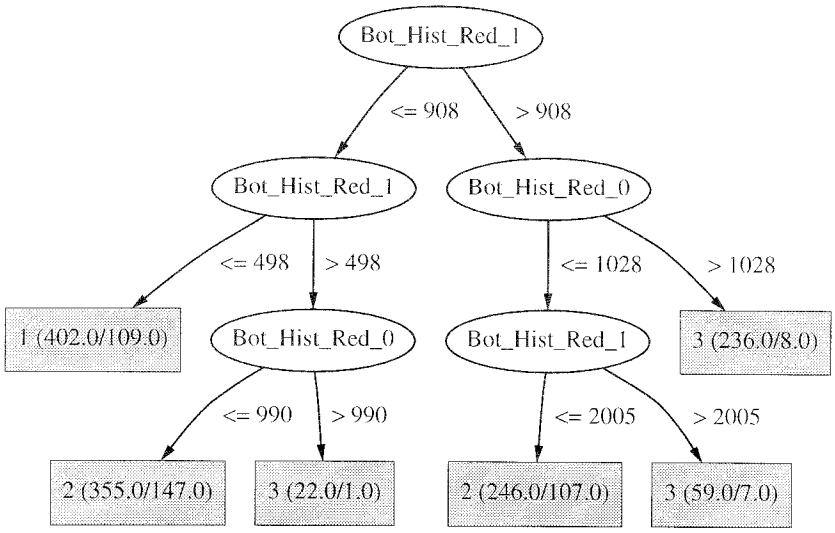



*Figure 2.* Example Consumer Decision Tree (consumer 10)

Feature selection and inductive learning provide an insight into the complex relationships which exist where a set of inexperienced consumers are confronted with a highly variable biological product. Boosting can then be applied to the induced models to enhance classification accuracy.

## 3. INTERACTIVE MACHINE LEARNING

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 modify the algorithm to match the properties of the domain or data set—for example, the amount of noise in the data. Users who are familiar with the workings of an algorithm still have to resort to trial and error to find optimal parameter settings for a particular application. Most users have a 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 learning technique for a 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 affect the model generated, and there is no other way for domain knowledge to enter the inductive process beyond the data preparation stage.

An option in the *Classify* screen of the Knowledge Explorer supports a graphical, interactive approach to machine learning. Model construction is guided by the user, who "draws" 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.

Figure 3 illustrates the user interface. There are two kinds of panel: tree visualizers (Figure 3a, 3d, and 3f) and data visualizers (Figure 3b, 3c, and 3e). 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 3a, 3d, and 3f 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 3b, 3c, and 3e show the data for the root node of Figure 3a, the root node of Figure 3d, and the right child of the root of Figure 3f 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.

### 3.1 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 2, 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 data set, 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 data set of Figure 3). 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 scatter 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 3c and 3e). 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.

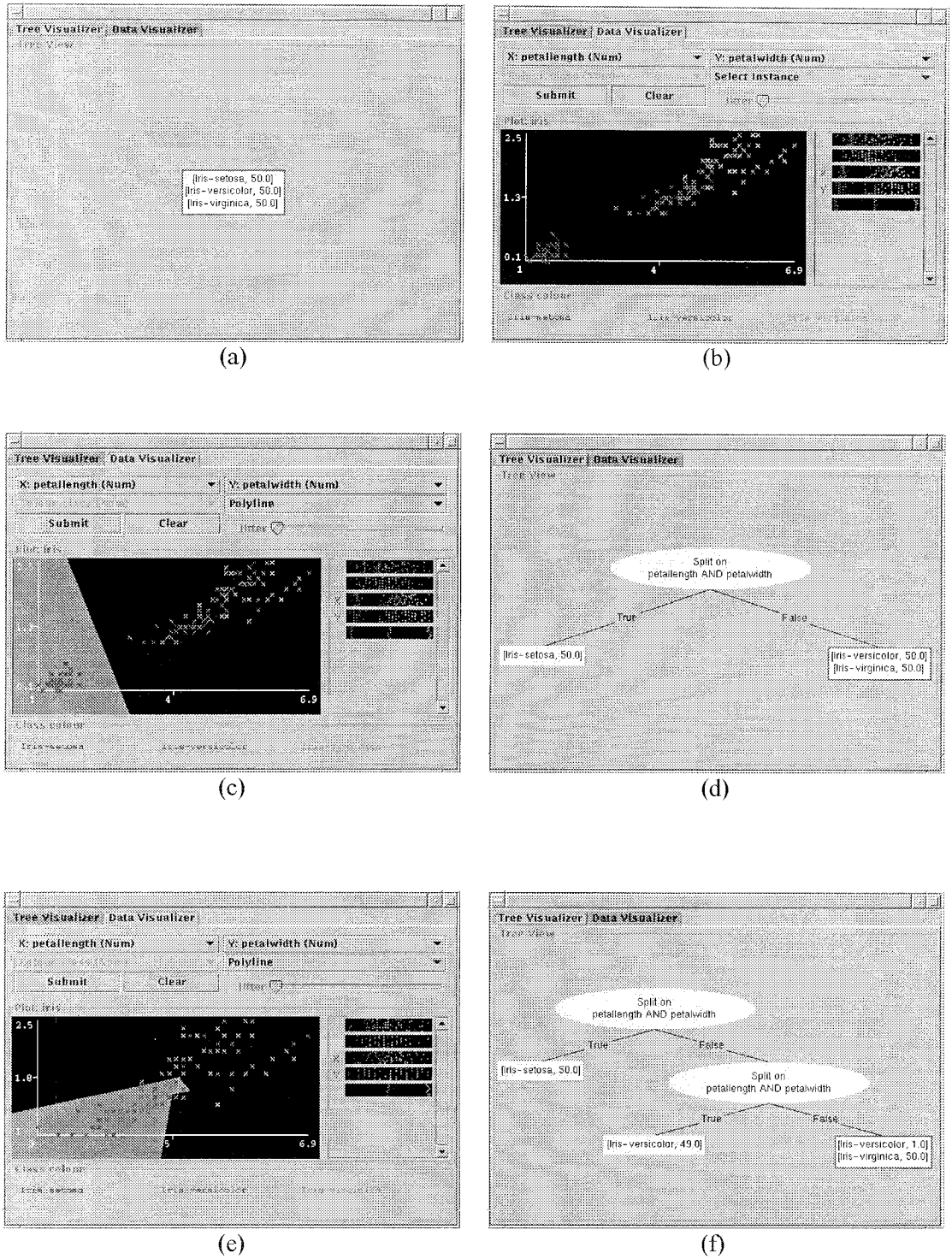



*Figure 3.* Constructing a classifier for the Iris data

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.

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 3b). This brings up a text window summarizing all attribute values for the instances (possibly more than one) located at that point in the plot.

Here is a detailed walk through the process of building a decision tree for the well-known (in the statistics and data mining community) Iris data (Fisher, 1936; data set obtained from (Blake et al, 1998)). This data set has a simple structure that lends itself naturally to interactive classifier construction. It consists of four numeric attributes that measure properties of Iris flowers. The learning task is to construct a decision tree classifier based on these four attributes that will correctly assign test data to the three classes, each representing a different variety of Iris (*virginica*, *versicolor* and *setosa*).

Before any splits are made, the tree visualizer displays a single node that corresponds to the root of the tree (Figure 3a). 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 3d and 3f).

To generate a split, the user switches to the data visualizer, which at this juncture displays the data points at the root node. Figure 3b 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.

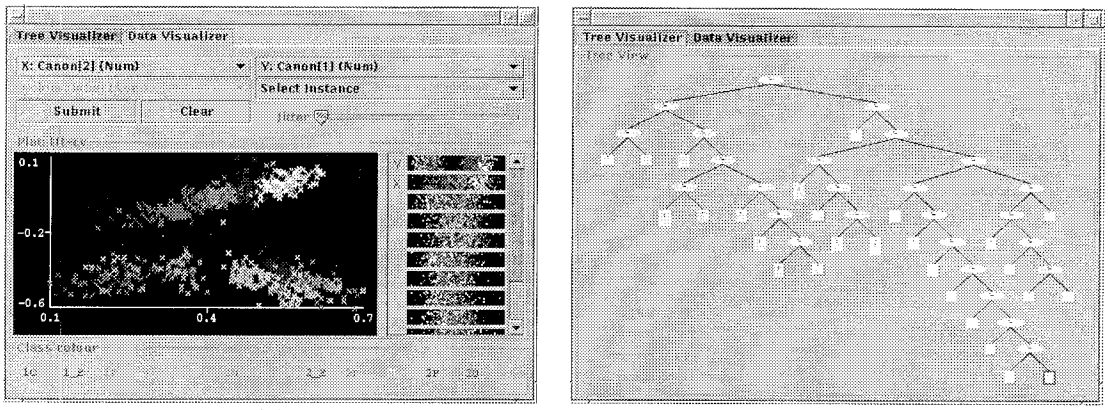
Next, the user draws a split in Figure 3c, in this case by choosing the polyline option to generate an open-sided polygon and splitting off the instances belonging to the Iris-setosa variety (located in the lower left corner of the display, and easily distinguished by color in the actual interface). The "enclosed" area of the polyline is shown in light gray.

Figure 3d 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 remaining instances into two almost pure subsets in Figure 3e. The resulting decision tree is shown in Figure 3f. It contains a total of five nodes and classifies all but one of the training instances correctly.

### 3.1 CASE STUDY 2 – KIWIFRUIT DATA

Our interest in interactive machine learning derives from the observation that several data sets from our applied data mining projects appear to lend themselves naturally to manual classification. An example of this type of problem involves classifying kiwifruit vines into twelve classes. The task is to determine which one of twelve pre-harvest fruit management treatments 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.




(a) (b)

*Figure 4.* Classifying Kiwifruit Vines

The training data, visualized using the first two of eleven attributes, is shown in Figure 4a. A user, 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 (using only four attributes) and achieved an accuracy of 85.8% on the test data: Figure 4b shows a miniature view. For comparison, we ran the decision tree inducer J4.8 (also used in the first case study) over the same training and test data. It produced a tree containing 93 nodes (using nine attributes) 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 is preliminary but very encouraging. This part of the Knowledge Explorer allows users to become involved in model construction, taking advantage of the innate human ability to visually identify boundaries between classes in the data.

## 4. CONCLUSION

In this paper we have described the WEKA Knowledge Explorer, a tool for exploring data using inductive learning. We have given a brief description of its features and demonstrated its use in two case studies related to food process technology. The tool can be used to uncover previously unknown aspects of the data, and is particularly useful in discovering the relative relevance of attributes in the classification task (Section 2). The user classifier described in Section 3 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 Knowledge Explorer has this capability, but this feature and the impact of domain knowledge on model construction have not been empirically evaluated. This type of empirical study is next on our research agenda.

WEKA is proving to be an extremely popular piece of software—since its release there have been over 100 downloads per week. Through the Knowledge Explorer interface it is possible for non-programmers, on any computer platform, to produce sound inductive models of their data.

To date, this interface has been used in several applications in the food sciences: determining the factors that lead to apple bruising (Holmes *et al*, 1998), identifying the factors that contribute to a degradation in the moisture content of milk powder, modeling mushroom grading criteria (Kusabs *et al*, 2000), and discovering the treatment that has been applied to a kiwifruit vine. WEKA has the potential to become a valuable part of a food scientist's set of modeling tools.

## ACKNOWLEDGEMENTS

This work was supported by the New Zealand Foundation for Research, Science and Technology (FRST). We would like to thank Sally Jo Cunningham for her helpful comments on an earlier draft of this paper.

## REFERENCES

Blake, C., Keogh, E. and 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].

Fisher,R.A (1936). The use of multiple measurements in taxonomic problems. *Annual Eugenics*, 7, Part II, 179-188 (1936); also in Contributions to Mathematical Statistics, John Wiley 1950.

Freund, Y. and R. E. Schapire. 1996. "Experiments with a new boosting algorithm," in *Proceedings of the 13th International Conference on Machine Learning*. Morgan Kaufmann Publishers, San Mateo, California, pp. 148-156.

Holmes, G., S. J. Cunningham, B. T. Dela Rue and A. F. Bollen. 1998. "Predicting apple bruising using machine learning," *Acta Hort*, 476:289-296.

Kim, J., Mowat, A., Poole, P. and 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.* University of Otago, (pp. 213-218)..

Kohavi, R., and John, G. H. 1995. "Wrappers for feature subset selection." *Artificial Intelligence*, 97:273-324.

Kusabs, N. J., Bollen, A.F., Trigg L. and Holmes G. 2000. "Objective measurement of mushroom quality relative to industry inspectors." In preparation.

Moore, A. W. and M. S. Lee. 1994. "Efficient algorithms for minimising cross validation error," in *Proceedings of the 11th International Conference on Machine Learning.* Morgan Kaufmann Publishers, New Brunswick, NJ, pp. 190-198.

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

Tao, Y., P. H. Heinemann, Z. Varghese, C. T. Morrow and H. J. Sommer III. 1995. "Machine vision for colour inspection of potatoes and apples." *Transactions of the ASAE*, 38(5):1555-1561.

Witten, I. H., and Frank E. (1999) *Data Mining: Practical Machine Learning Tools and Techniques with Java Implementations*. Morgan Kaufmann, San Francisco.