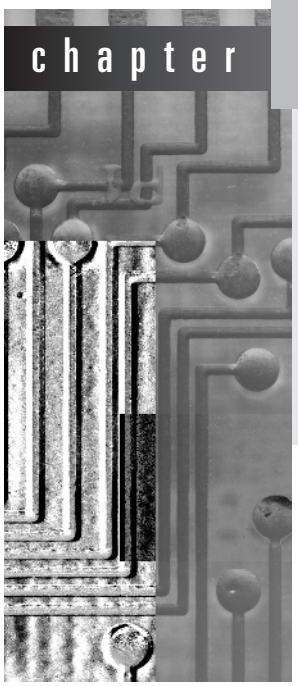
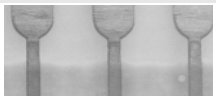


chapter 10

The Explorer



Weka's main graphical user interface, the Explorer, gives access to all its facilities using menu selection and form filling. It is illustrated in Figure 10.1. There are six different panels, selected by the tabs at the top, corresponding to the various data mining tasks that Weka supports.

10.1 Getting started

Suppose you have some data and you want to build a decision tree from it. First, you need to prepare the data then fire up the Explorer and load in the data. Next you select a decision tree construction method, build a tree, and interpret the output. It's easy to do it again with a different tree construction algorithm or a different evaluation method. In the Explorer you can flip back and forth between the results you have obtained, evaluate the models that have been built on different datasets, and visualize graphically both the models and the datasets themselves—including any classification errors the models make.

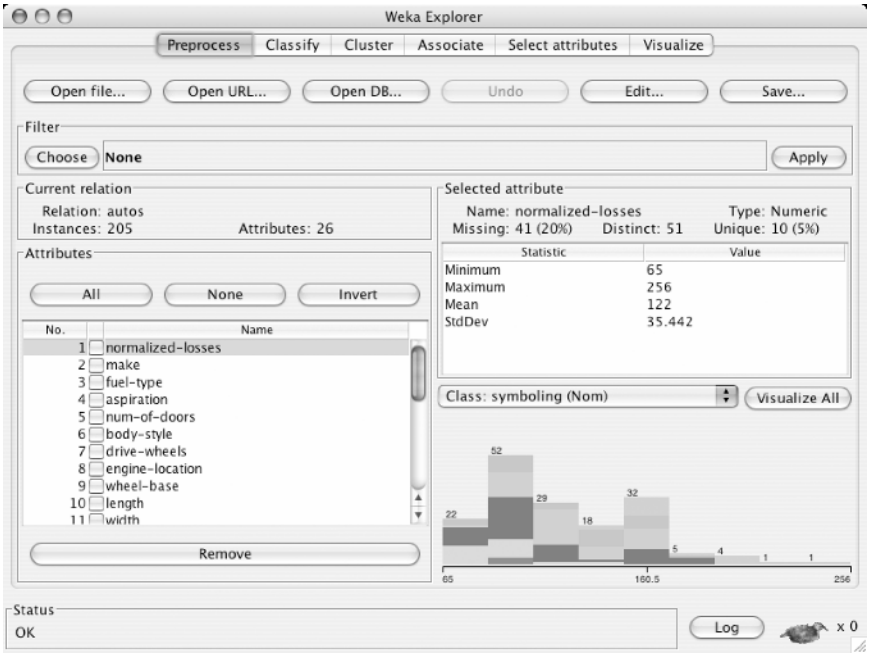


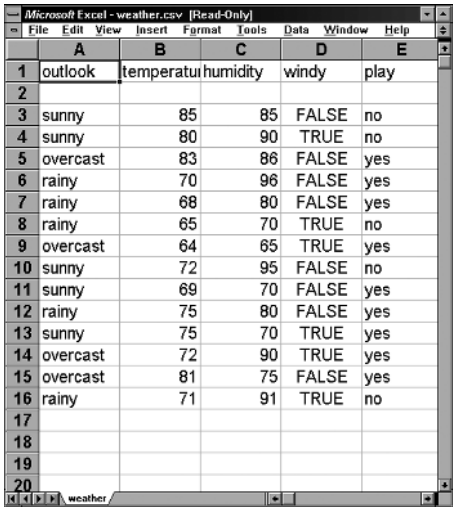
Figure 10.1 The Explorer interface.

Preparing the data

The data is often presented in a spreadsheet or database. However, Weka's native data storage method is ARFF format (Section 2.4). You can easily convert from a spreadsheet to ARFF. The bulk of an ARFF file consists of a list of the instances, and the attribute values for each instance are separated by commas (Figure 2.2). Most spreadsheet and database programs allow you to export data into a file in comma-separated value (CSV) format as a list of records with commas between items. Having done this, you need only load the file into a text editor or word processor; add the dataset's name using the *@relation* tag, the attribute information using *@attribute,* and a *@data* line; and save the file as raw text. For example, Figure 10.2 shows an Excel spreadsheet containing the weather data from Section 1.2, the data in CSV form loaded into Microsoft Word, and the result of converting it manually into an ARFF file. However, you don't actually have to go through these steps to create the ARFF file yourself, because the Explorer can read CSV spreadsheet files directly, as described later.

Loading the data into the Explorer

Let's load this data into the Explorer and start analyzing it. Fire up Weka to get the panel shown in Figure 10.3(a). Select *Explorer* from the four graphical user



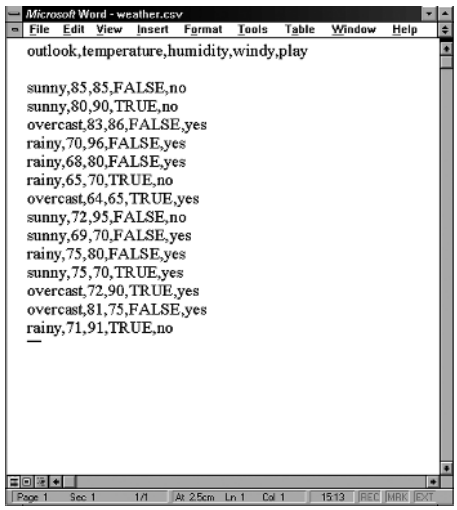
Microsoft Excel - weather.csv [Read-Only]

File Edit View Insert Format Tools Data Window Help

	A	B	C	D	E
1	outlook	temperatu	humidity	windy	play
2					
3	sunny	85	85	FALSE	no
4	sunny	80	90	TRUE	no
5	overcast	83	86	FALSE	yes
6	rainy	70	96	FALSE	yes
7	rainy	68	80	FALSE	yes
8	rainy	65	70	TRUE	no
9	overcast	64	65	TRUE	yes
10	sunny	72	95	FALSE	no
11	sunny	69	70	FALSE	yes
12	rainy	75	80	FALSE	yes
13	sunny	75	70	TRUE	yes
14	overcast	72	90	TRUE	yes
15	overcast	81	75	FALSE	yes
16	rainy	71	91	TRUE	no
17					
18					
19					
20					

weather

(a)



Microsoft Word - weather.csv

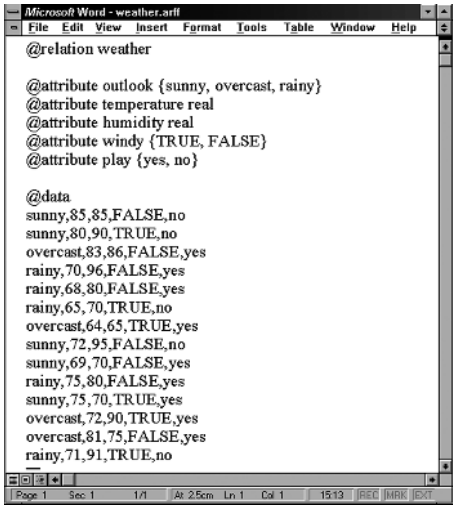
File Edit View Insert Format Tools Table Window Help

```
outlook,temperature,humidity,windy,play

sunny,85,85,FALSE,no
sunny,80,90,TRUE,no
overcast,83,86,FALSE,yes
rainy,70,96,FALSE,yes
rainy,68,80,FALSE,yes
rainy,65,70,TRUE,no
overcast,64,65,TRUE,yes
sunny,72,95,FALSE,no
sunny,69,70,FALSE,yes
rainy,75,80,FALSE,yes
sunny,75,70,TRUE,yes
overcast,72,90,TRUE,yes
overcast,81,75,FALSE,yes
rainy,71,91,TRUE,no
```

Page 1 Sec 1 1/1 At 2.5cm Ln 1 Col 1 15:13 REC MRK EXT

(b)



Microsoft Word - weather.arff

File Edit View Insert Format Tools Table Window Help

```
@relation weather

@attribute outlook {sunny, overcast, rainy}
@attribute temperature real
@attribute humidity real
@attribute windy {TRUE, FALSE}
@attribute play {yes, no}

@data
sunny,85,85,FALSE,no
sunny,80,90,TRUE,no
overcast,83,86,FALSE,yes
rainy,70,96,FALSE,yes
rainy,68,80,FALSE,yes
rainy,65,70,TRUE,no
overcast,64,65,TRUE,yes
sunny,72,95,FALSE,no
sunny,69,70,FALSE,yes
rainy,75,80,FALSE,yes
sunny,75,70,TRUE,yes
overcast,72,90,TRUE,yes
overcast,81,75,FALSE,yes
rainy,71,91,TRUE,no
```

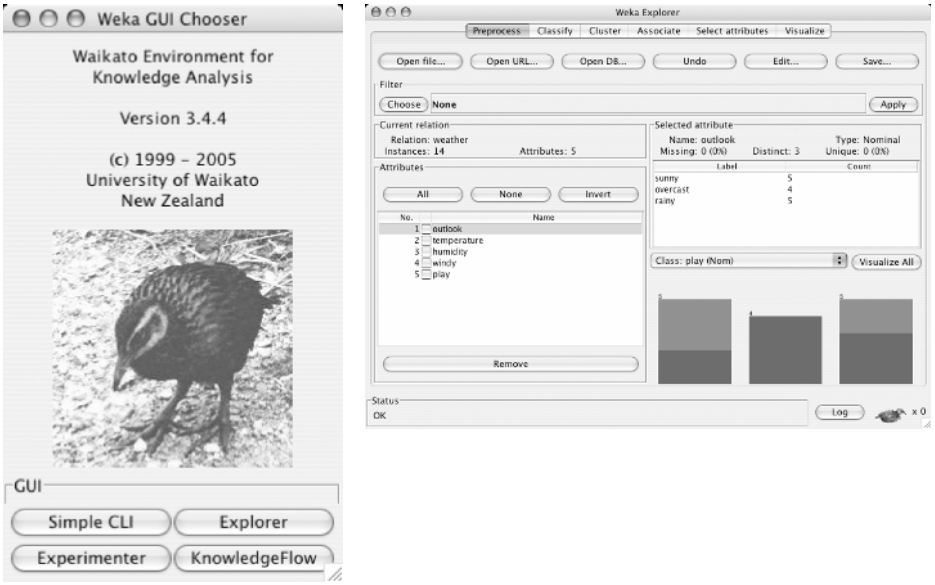
Page 1 Sec 1 1/1 At 2.5cm Ln 1 Col 1 15:13 REC MRK EXT

(c)

Figure 10.2 Weather data: (a) spreadsheet, (b) CSV format, and (c) ARFF.

interface choices at the bottom. (The others were mentioned earlier: *Simple CLI* is the old-fashioned command-line interface.)

What you see next is the main Explorer screen, shown in Figure 10.3(b). Actually, the figure shows what it will look like *after* you have loaded in the weather data. The six tabs along the top are the basic operations that the Explorer supports: right now we are on *Preprocess*. Click the *Open file* button to



(a) (b)

Figure 10.3 The Weka Explorer: (a) choosing the Explorer interface and (b) reading in the weather data.

bring up a standard dialog through which you can select a file. Choose the *weather.arff* file. If you have it in CSV format, change from *ARFF data files* to *CSV data files*. When you specify a *.csv* file it is automatically converted into ARFF format.

Having loaded the file, the screen will be as shown in Figure 10.3(b). This tells you about the dataset: it has 14 instances and five attributes (center left); the attributes are called *outlook, temperature, humidity, windy,* and *play* (lower left). The first attribute, *outlook,* is selected by default (you can choose others by clicking them) and has no missing values, three distinct values, and no unique values; the actual values are *sunny, overcast,* and *rainy,* and they occur five, four, and five times, respectively (center right). A histogram at the lower right shows how often each of the two values of the class, *play,* occurs for each value of the *outlook* attribute. The attribute *outlook* is used because it appears in the box above the histogram, but you can draw a histogram of any other attribute instead. Here *play* is selected as the class attribute; it is used to color the histogram, and any filters that require a class value use it too.

The *outlook* attribute in Figure 10.3(b) is nominal. If you select a numeric attribute, you see its minimum and maximum values, mean, and standard

deviation. In this case the histogram will show the distribution of the class as a function of this attribute (an example appears in Figure 10.9 on page 384).

You can delete an attribute by clicking its checkbox and using the *Remove* button. *All* selects all the attributes, *None* selects none, and *Invert* inverts the current selection. You can undo a change by clicking the *Undo* button. The *Edit* button brings up an editor that allows you to inspect the data, search for particular values and edit them, and delete instances and attributes. Right-clicking on values and column headers brings up corresponding context menus.

Building a decision tree

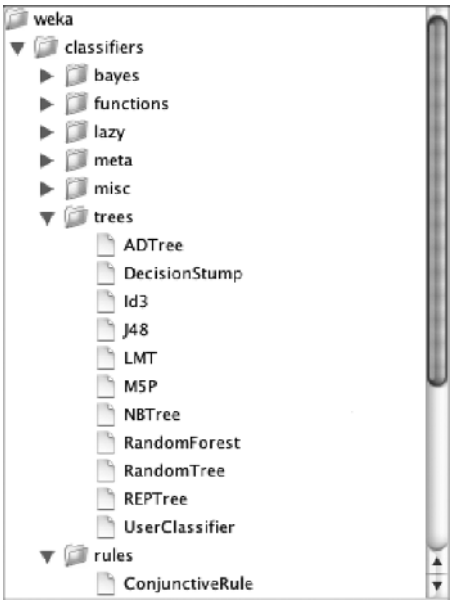
To see what the C4.5 decision tree learner described in Section 6.1 does with this dataset, use the J4.8 algorithm, which is Weka's implementation of this decision tree learner. (J4.8 actually implements a later and slightly improved version called C4.5 revision 8, which was the last public version of this family of algorithms before the commercial implementation C5.0 was released.) Click the *Classify* tab to get a screen that looks like Figure 10.4(b). Actually, the figure shows what it will look like *after* you have analyzed the weather data.

First select the classifier by clicking the *Choose* button at the top left, opening up the *trees* section of the hierarchical menu in Figure 10.4(a), and finding *J48*. The menu structure represents the organization of the Weka code into modules, which will be described in Chapter 13. For now, just open up the hierarchy as necessary—the items you need to select are always at the lowest level. Once selected, *J48* appears in the line beside the *Choose* button as shown in Figure 10.4(b), along with its default parameter values. If you click that line, the J4.8 classifier's object editor opens up and you can see what the parameters mean and alter their values if you wish. The Explorer generally chooses sensible defaults.

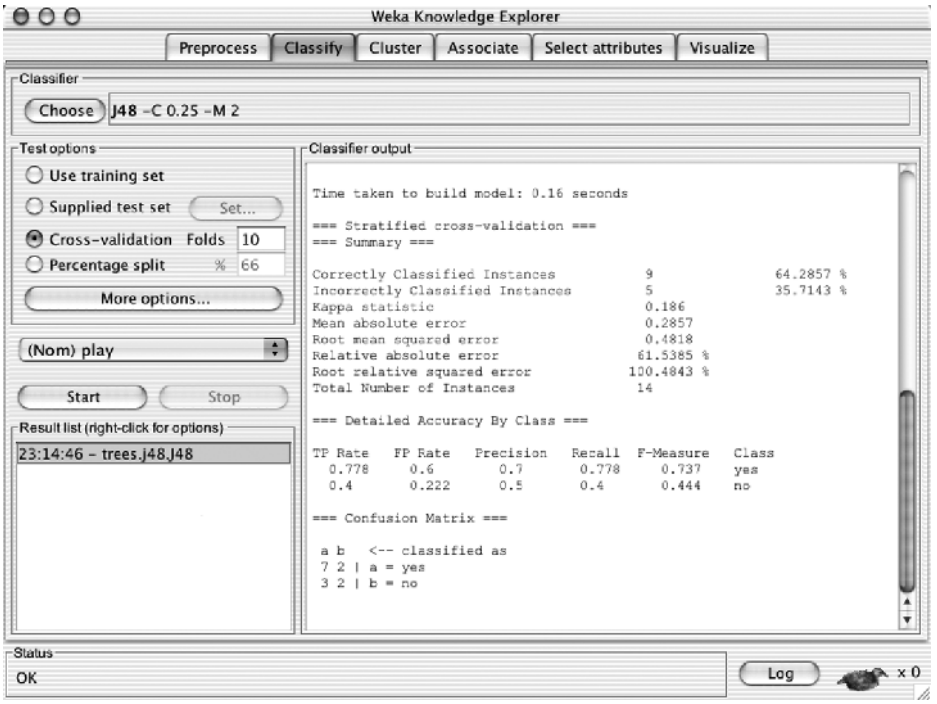
Having chosen the classifier, invoke it by clicking the *Start* button. Weka works for a brief period—when it is working, the little bird at the lower right of Figure 10.4(b) jumps up and dances—and then produces the output shown in the main panel of Figure 10.4(b).

Examining the output

Figure 10.5 shows the full output (Figure 10.4(b) only gives the lower half). At the beginning is a summary of the dataset, and the fact that tenfold cross-validation was used to evaluate it. That is the default, and if you look closely at Figure 10.4(b) you will see that the *Cross-validation* box at the left is checked. Then comes a pruned decision tree in textual form. The first split is on the *outlook* attribute, and then, at the second level, the splits are on *humidity* and *windy,* respectively. In the tree structure, a colon introduces the class label that



(a)



(b)

Figure 10.4 Using J4.8: (a) finding it in the classifiers list and (b) the *Classify* tab.

```
=== Run information ===

Scheme:       weka.classifiers.trees.J48 -C 0.25 -M 2
Relation:     weather
Instances:    14
Attributes:   5
              outlook
              temperature
              humidity
              windy
              play
Test mode:    10-fold cross-validation

=== Classifier model (full training set) ===

J48 pruned tree
------------------

outlook = sunny
|   humidity <= 75: yes (2.0)
|   humidity > 75: no (3.0)
outlook = overcast: yes (4.0)
outlook = rainy
|   windy = TRUE: no (2.0)
|   windy = FALSE: yes (3.0)

Number of Leaves  :  5

Size of the tree :   8

Time taken to build model: 0.27 seconds

=== Stratified cross-validation ===
=== Summary ===

Correctly Classified Instances           9               64.2857 %
Incorrectly Classified Instances         5               35.7143 %
Kappa statistic                          0.186
Mean absolute error                      0.2857
Root mean squared error                  0.4818
Relative absolute error                 60      %
Root relative squared error             97.6586 %
Total Number of Instances               14
```

Figure 10.5 Output from the J4.8 decision tree learner.

```
=== Detailed Accuracy By Class ===

TP Rate   FP Rate   Precision   Recall  F-Measure   Class
  0.778     0.6        0.7       0.778     0.737    yes
  0.4       0.222      0.5       0.4       0.444    no

=== Confusion Matrix ===

 a b   <-- classified as
 7 2 | a = yes
 3 2 | b = no
```

Figure 10.5 (continued)

has been assigned to a particular leaf, followed by the number of instances that reach that leaf, expressed as a decimal number because of the way the algorithm uses fractional instances to handle missing values. If there were incorrectly classified instances (there aren't in this example) their number would appear, too: thus *2.0/1.0* means that two instances reached that leaf, of which one is classified incorrectly. Beneath the tree structure the number of leaves is printed; then the total number of nodes (*Size of the tree*). There is a way to view decision trees more graphically: see pages 378–379 later in this chapter.

The next part of the output gives estimates of the tree's predictive performance. In this case they are obtained using stratified cross-validation with 10 folds, the default in Figure 10.4(b). As you can see, more than 30% of the instances (5 out of 14) have been misclassified in the cross-validation. This indicates that the results obtained from the training data are optimistic compared with what might be obtained from an independent test set from the same source. From the confusion matrix at the end (described in Section 5.7) observe that 2 instances of class *yes* have been assigned to class *no* and 3 of class *no* are assigned to class *yes*.

As well as the classification error, the evaluation module also outputs the Kappa statistic (Section 5.7), the mean absolute error, and the root mean-squared error of the class probability estimates assigned by the tree. The root mean-squared error is the square root of the average quadratic loss (Section 5.6). The mean absolute error is calculated in a similar way using the absolute instead of the squared difference. It also outputs relative errors, which are based on the prior probabilities (i.e., those obtained by the *ZeroR* learning scheme described later). Finally, for each class it also outputs some statistics from page 172.

Doing it again

You can easily run J4.8 again with a different evaluation method. Select *Use training set* (near the top left in Figure 10.4(b)) and click *Start* again. The classifier output is quickly replaced to show how well the derived model performs on the training set, instead of showing the cross-validation results. This evaluation is highly optimistic (Section 5.1). It may still be useful, because it generally represents an upper bound to the model's performance on fresh data. In this case, all 14 training instances are classified correctly. In some cases a classifier may decide to leave some instances unclassified, in which case these will be listed as *Unclassified Instances*. This does not happen for most learning schemes in Weka.

The panel in Figure 10.4(b) has further test options: *Supplied test set,* in which you specify a separate file containing the test set, and *Percentage split,* with which you can hold out a certain percentage of the data for testing. You can output the predictions for each instance by clicking the *More options* button and checking the appropriate entry. There are other useful options, such as suppressing some output and including other statistics such as entropy evaluation measures and cost-sensitive evaluation. For the latter you must enter a cost matrix: type the number of classes into the *Classes* box (and terminate it with the *Enter* or *Return* key) to get a default cost matrix (Section 5.7), then edit the values as required.

The small pane at the lower left of Figure 10.4(b), which contains one highlighted line, is a history list of the results. The Explorer adds a new line whenever you run a classifier. Because you have now run the classifier twice, the list will contain two items. To return to a previous result set, click the corresponding line and the output for that run will appear in the classifier output pane. This makes it easy to explore different classifiers or evaluation schemes and revisit the results to compare them.

Working with models

The result history list is the entry point to some powerful features of the Explorer. When you *right*-click an entry a menu appears that allows you to view the results in a separate window, or save the result buffer. More importantly, you can save the model that Weka has generated in the form of a Java object file. You can reload a model that was saved previously, which generates a new entry in the result list. If you now supply a test set, you can reevaluate the old model on that new set.

Several items in the right-click menu allow you to visualize the results in various ways. At the top of the Explorer interface is a separate *Visualize* tab, but that is different: it shows the dataset, not the results for a particular model. By

right-clicking an entry in the history list you can see the classifier errors. If the model is a tree or a Bayesian network you can see its structure. You can also view the margin curve (page 324) and various cost and threshold curves (Section 5.7). For cost and threshold curves you must choose a class value from a submenu. The *Visualize threshold curve* menu item allows you to see the effect of varying the probability threshold above which an instance is assigned to that class. You can select from a wide variety of curves that include the ROC and recall–precision curves (Table 5.7). To see these, choose the X- and Y-axes appropriately from the menus given. For example, set X to *False positive rate* and Y to *True positive rate* for an ROC curve or X to *Recall* and Y to *Precision* for a recall–precision curve.

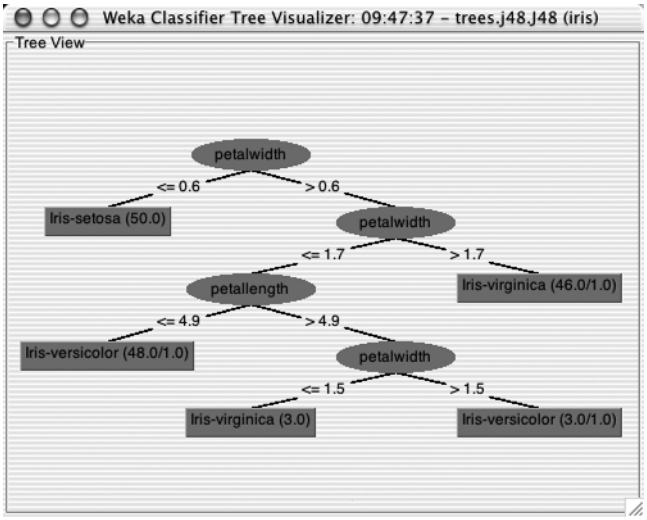
Figure 10.6 shows two ways of looking at the result of using J4.8 to classify the Iris dataset (Section 1.2)—we use this rather than the weather data because it produces more interesting pictures. Figure 10.6(a) shows the tree. Right-click a blank space in this window to bring up a menu enabling you to automatically scale the view or force the tree into the window. Drag the mouse to pan around the space. It's also possible to visualize the instance data at any node, if it has been saved by the learning algorithm.

Figure 10.6(b) shows the classifier errors on a two-dimensional plot. You can choose which attributes to use for X and Y using the selection boxes at the top. Alternatively, click one of the speckled horizontal strips to the right of the plot: left-click for X and right-click for Y. Each strip shows the spread of instances along that attribute. X and Y appear beside the ones you have chosen for the axes.

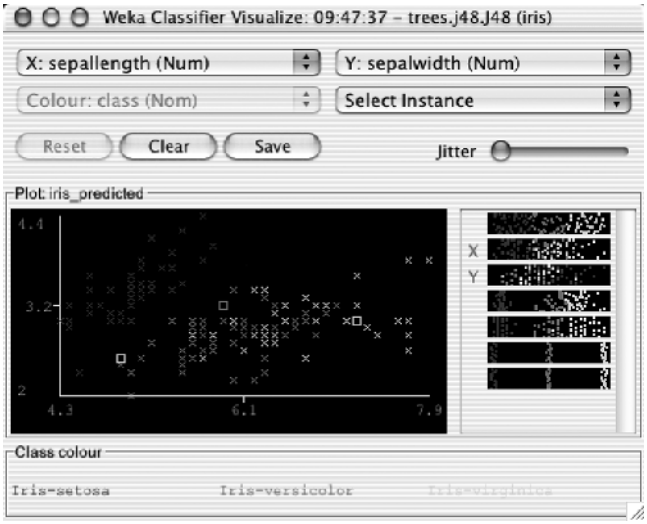
The data points are colored according to their class: blue, red, and green for *Iris setosa, Iris versicolor,* and *Iris virginica,* respectively (there is a key at the bottom of the screen). Correctly classified instances are shown as crosses; incorrectly classified ones appear as boxes (of which there are three in Figure 10.6(b)). You can click on an instance to bring up relevant details: its instance number, the values of the attributes, its class, and the predicted class.

When things go wrong

Beneath the result history list, at the bottom of Figure 10.4(b), is a status line that says, simply, *OK*. Occasionally, this changes to *See error log,* an indication that something has gone wrong. For example, there may be constraints among the various different selections you can make in a panel. Most of the time the interface grays out inappropriate selections and refuses to let you choose them. But occasionally the interactions are more complex, and you can end up selecting an incompatible set of options. In this case, the status line changes when Weka discovers the incompatibility—typically when you press *Start*. To see the error, click the *Log* button to the left of the weka in the lower right-hand corner of the interface.



(a)



(b)

Figure 10.6 Visualizing the result of J4.8 on the iris dataset: (a) the tree and (b) the classifier errors.

10.2 Exploring the Explorer

We have briefly investigated two of the six tabs at the top of the Explorer window in Figure 10.3(b) and Figure 10.4(b). In summary, here's what all of the tabs do:

1. *Preprocess:* Choose the dataset and modify it in various ways.
2. *Classify:* Train learning schemes that perform classification or regression and evaluate them.
3. *Cluster:* Learn clusters for the dataset.
4. *Associate:* Learn association rules for the data and evaluate them.
5. *Select attributes:* Select the most relevant aspects in the dataset.
6. *Visualize:* View different two-dimensional plots of the data and interact with them.

Each tab gives access to a whole range of facilities. In our tour so far, we have barely scratched the surface of the *Preprocess* and *Classify* panels.

At the bottom of every panel is a *Status* box and a *Log* button. The status box displays messages that keep you informed about what's going on. For example, if the Explorer is busy loading a file, the status box will say so. Right-clicking anywhere inside this box brings up a little menu with two options: display the amount of memory available to Weka, and run the Java garbage collector. Note that the garbage collector runs constantly as a background task anyway.

Clicking the *Log* button opens a textual log of the actions that Weka has performed in this session, with timestamps.

As noted earlier, the little bird at the lower right of the window jumps up and dances when Weka is active. The number beside the × shows how many concurrent processes that are running. If the bird is standing but stops moving, it's sick! Something has gone wrong, and you should restart the Explorer.

Loading and filtering files

Along the top of the *Preprocess* panel in Figure 10.3(b) are buttons for opening files, URLs, and databases. Initially, only files whose names end in *.arff* appear in the file browser; to see others, change the *Format* item in the file selection box.

Converting files to ARFF

Weka has three file format converters: for spreadsheet files with the extension *.csv,* for C4.5's native file format with the extensions *.names* and *.data,* and for serialized instances with the extension *.bsi*. The appropriate converter is used based on the extension. If Weka cannot load the data, it tries to interpret it as ARFF. If that fails, it pops up the box shown in Figure 10.7(a).

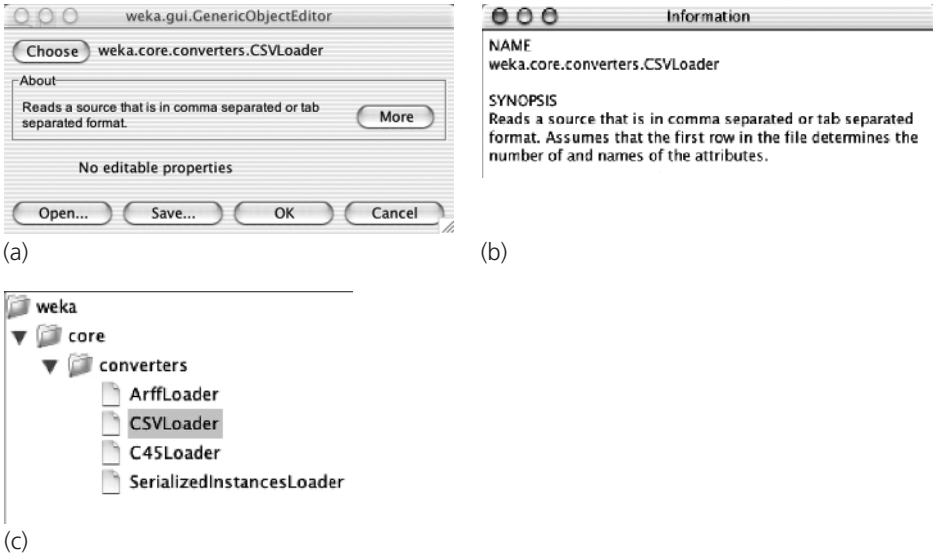


Figure 10.7 Generic object editor: (a) the editor, (b) more information (click *More*), and (c) choosing a converter (click *Choose*).

This is a generic object editor, used throughout Weka for selecting and configuring objects. For example, when you set parameters for a classifier, you use the same kind of box. The *CSVLoader* for *.csv* files is selected by default, and the *More* button gives you more information about it, shown in Figure 10.7(b). It is always worth looking at the documentation! In this case, it explains that the spreadsheet's first row determines the attribute names. Click *OK* to use this converter. For a different one, click *Choose* to select from the list in Figure 10.7(c).

The *ArffLoader* is the first option, and we reached this point only because it failed. The *CSVLoader* is the default, and we clicked *Choose* because we want a different one. The third option is for the C4.5 format, in which there are two files for a dataset, one giving field names and the other giving the actual data. The fourth, for serialized instances, is for reloading a dataset that has been saved as a Java serialized object. Any Java object can be saved in this form and reloaded. As a native Java format, it is quicker to load than an ARFF file, which must be parsed and checked. When repeatedly reloading a large dataset it may be worth saving it in this form.

Further features of the generic object editor in Figure 10.7(a) are *Save,* which saves a configured object, and *Open,* which opens a previously saved one. These are not useful for this particular kind of object. But other generic object editor panels have many editable properties, and having gone to some trouble to set them up you may want to save the configured object to reuse later.

Files on your computer are not the only source of datasets for Weka. You can open a URL, and Weka will use the hypertext transfer protocol (HTTP) to download an ARFF file from the Web. Or you can open a database (*Open DB*)—any database that has a Java database connectivity (JDBC) driver—and retrieve instances using the SQL *Select* statement. This returns a relation that Weka reads in as an ARFF file. To make this work with your database, you may need to modify the file *weka/experiment/DatabaseUtils.props* in the Weka distribution by adding your database driver to it. (To access this file, expand the *weka.jar* file in the Weka distribution.)

Data can be saved in all these formats using the *Save* button in Figure 10.3(b). Apart from loading and saving datasets, the *Preprocess* panel also allows you to filter them. Filters are an important component of Weka.

Using filters

Clicking *Choose* (near the top left) in Figure 10.3(b) gives a list of filters like that in Figure 10.8(a). Actually, you get a collapsed version: click on an arrow to open up its contents. We will describe how to use a simple filter to delete specified attributes from a dataset, in other words, to perform manual attribute selection. The same effect can be achieved more easily by selecting the relevant attributes using the tick boxes and pressing the *Remove* button. Nevertheless, we describe the equivalent filtering operation explicitly, as an example.

Remove is an unsupervised attribute filter, and to see it you must scroll further down the list. When selected, it appears in the line beside the *Choose* button, along with its parameter values—in this case the line reads simply "Remove." Click that line to bring up a generic object editor with which you can examine and alter the filter's properties. (You did the same thing earlier by clicking the *J48* line in Figure 10.4(b) to open the J4.8 classifier's object editor.) The object editor for the *Remove* filter is shown in Figure 10.8(b). To learn about it, click *More* to show the information in Figure 10.8(c). This explains that the filter removes a range of attributes from the dataset. It has an option, *attributeIndices,* that specifies the range to act on and another called *invertSelection* that determines whether the filter selects attributes or deletes them. There are boxes for both of these in the object editor shown in Figure 10.8(b), and in fact we have already set them to *1,2* (to affect attributes 1 and 2, namely, *outlook* and *temperature*) and *False* (to remove rather than retain them). Click *OK* to set these properties and close the box. Notice that the line beside the *Choose* button now reads *Remove –R 1,2*. In the command-line version of the *Remove* filter, the option *–R* is used to specify which attributes to remove. After configuring an object it's often worth glancing at the resulting command-line formulation that the Explorer sets up.

Apply the filter by clicking *Apply* (at the right-hand side of Figure 10.3(b)). Immediately the screen in Figure 10.9 appears—just like the one in Figure