# Weka Tutorial

Weka is an open source collection of data mining tasks which you can utilize in a number of different ways. It comes with a Graphical User Interface (GUI), but can also be called from your own Java code. You can even write your own batch files for tasks that you need to execute more than once, possibly with slightly different parameters each time.

## 1 Downloading and installing Weka

If you want to use Weka on your own machine, follow the instructions on Weka's Homepage. Weka is installed on all the CS machines.

If you are using Weka on your own machine, you should download the folder containing Weka data files. These can also be found in the data directory of your Weka installation.

For those using the CS machines, the data files are in the folder

```
http://www.cs.vassar.edu/~cs366/data/weka_files
```

## 2 Starting up the Weka Explorer

From the CS machines: Open a command window and type

```
weka
```

On your own computer: Either double-click on the weka-3-8-2-oracle-jvm icon in your weka installation folder or open a command window and type:

```
java -Xmx500M weka.gui.explorer.Explorer
```

You will see the Weka GUI chooser. You'll have a choice between the command line interface (CLI), the Experimenter, the Explorer, Workbench, and Knowledge flow. Initially, we'll stick with the Explorer, so click on **Explorer**.

## 3 Introduction to the Weka Explorer

The **Preprocess** panel is the panel opened after starting the WEKA Explorer. Before changing to any of the other panels the Explorer must have a data set to work with.

Load a dataset by clicking the *Open file* button in the top left corner of the panel. Choose the file named *weather.nominal.arff*.

After you have done this you will see:

As the result shows, the weather data has 14 instances, and 5 attributes called *outlook, temperature, humidity, windy,* and *play.* Click on the name of an attribute in the left subpanel to see information about the selected attribute on the right, such as its values and how many times an instance in the dataset has a particular value. This information is also shown in the form of a histogram. All attributes in this dataset are "nominal"that is, they have a predefined finite set of values. The last attribute, *play*, is the "class" attribute; its value can be *yes* or *no*.

A note: Weka's native data storage method is the ARFF format. The bulk of an ARFF file consists of a list of the instances, and the attribute values for each instance are separated by commas. You can see what this looks like by opening one of the data files directly, but this is not necessary now.

# 4 The Dataset Editor

It is possible to view and edit an entire dataset from within Weka. To do this, click the *Edit* button from the row of buttons at the top of the Preprocess panel. This opens a new window called Viewer, which lists all instances of the weather data:

| No. | 1: outlook Nominal | 2: temperature Nominal | 3: humidity Nominal | 4: windy Nominal | 5: play Nominal |
|---|---|---|---|---|---|
| 1 | sunny | hot | high | FALSE | no |
| 2 | sunny | hot | high | TRUE | no |
| 3 | overcast | hot | high | FALSE | yes |
| 4 | rainy | mild | high | FALSE | yes |
| 5 | rainy | cool | normal | FALSE | yes |
| 6 | rainy | cool | normal | TRUE | no |
| 7 | overcast | cool | normal | TRUE | yes |
| 8 | sunny | mild | high | FALSE | no |
| 9 | sunny | cool | normal | FALSE | yes |
| ... | rainy | mild | normal | FALSE | yes |
| ... | sunny | mild | normal | TRUE | yes |
| ... | overcast | mild | high | TRUE | yes |
| ... | overcast | hot | normal | FALSE | yes |
| ... | rainy | mild | high | TRUE | no |

Now load the file *iris.arff* and open it in the editor. How many numeric and how many nominal attributes does this dataset have?

## 4.1 Applying a Filter

Weka "filters" can be used to modify datasets in a systematic fashionthat is, they are data preprocessing tools. To remove an attribute from the weather dataset, use a filter called *Remove*; its full name is

```
weka. lters.unsupervised.attribute.Remove
```

Examine this name carefully. Filters are organized into a hierarchical structure of which the root is weka. Those in the unsupervised category don't require a class attribute to be set; those in the supervised category do. Filters are further divided into ones that operate primarily on attributes (the attribute category) and ones that operate primarily on instances (the instance category). Click the *Choose* button in the Preprocess panel to open a hierarchical menu from which you select a filter by following the path corresponding to its full name. Use the path given in the full name above to select the *Remove* filter. The text "Remove" will appear in the field next to the *Choose* button.

Click on the field containing this text. The Generic Object Editor window, which is used throughout Weka to set parameter values for all of the tools, opens. In this case it contains a short explanation of the *Remove* filter click *More* to get a fuller description. Enter 3 into the *attributeIndices* field and click the *OK* button. The window with the filter options closes. Now click the *Apply* button on the right, which runs the data through the filter. The filter removes the attribute with index 3 from the dataset, and you can see that this has happened. This change does not affect the dataset in the file; it only applies to the data held in memory. The changed dataset can be saved to a new ARFF file by pressing the *Save* button and entering a file name. The action of the filter can be undone by pressing the *Undo* button. Again, this applies to the version of the data held in memory.

What we have described illustrates how filters are applied to data. However, in the particular case of *Remove*, there is a simpler way of achieving the same effect. Instead of invoking a filter, attributes can be selected using the small boxes in the Attributes subpanel and removed using the *Remove* button that appears at the bottom, below the list of attributes.

**Do the following exercises:**

1. Load the *weather.nominal* dataset. Use the filter *weka.unsupervised.instance.RemoveWithValues* to remove all instances in which the humidity attribute has the value high. To do this, first make the field next to the Choose button show the text *RemoveWithValues*. Then click on it to get the Generic Object Editor window, and figure out how to change the filter settings appropriately.

2. Undo the change to the dataset that you just performed, and verify that the data has reverted to its original state.

## Tutorial Part II

**The immediately following pages are taken from the Weka tutorial in the book *Data Mining: Practical Machine Learning Tools and Techniques*, Third Edition, by Ian H. Witten, Eibe Frank, and Mark A. Hall. The full text of the book is available in pdf form here.**

Following that (material after the decorative page saying "Weka") is an extended tutorial for using Weka, consisting of a subset of the slides that are available on the course wiki page for this week's lecture. This is a far more in-depth tutorial in terms of providing multiple examples for you to work through. Also, there are screen-by-screen examples of how to use the basic functions in Weka that you can refer to even when doing the tutorial in the next section.

**Please do the following:**

- **Complete up to and including section 17.3 of the next group of pages (from the *Data Mining* book). You do not need to do the exercises, especially the ones that consist of open-ended questions about interpreting results etc. Note that the tutorial references other chapters in the book, often to provide illustrations. You can refer to them by accessing the pdf of the book at the link above.**

- **Go through the appended lecture slides tutorial and work through all the examples there. *Anything you do not finish in class should be done as homework due on Wednesday, February 21.***

## The Visualize Panel

Now take a look at Weka's data visualization facilities. These work best with numeric data, so we use the iris data. Load *iris.arff*, which contains the iris dataset of Table 1.4 containing 50 examples of three types of Iris: *Iris setosa*, *Iris versicolor*, and *Iris virginica*.

Click the *Visualize* tab to bring up the Visualize panel (shown in Figure 11.17). Click the first plot in the second row to open up a window showing an enlarged plot using the selected axes. Instances are shown as little crosses, the color of which depends on the instance's class. The *x*-axis shows the *sepallength* attribute, and the *y*-axis shows *petalwidth*.

Clicking on one of the crosses opens up an Instance Info window, which lists the values of all attributes for the selected instance. Close the Instance Info window again.

The selection fields at the top of the window containing the scatter plot determine which attributes are used for the *x*- and *y*-axes. Change the *x*-axis to *petalwidth* and the *y*-axis to *petallength*. The field showing *Color: class (Num)* can be used to change the color coding.

Each of the barlike plots to the right of the scatter plot window represents a single attribute. In each bar, instances are placed at the appropriate horizontal position and scattered randomly in the vertical direction. Clicking a bar uses that attribute for the *x*-axis of the scatter plot. Right-clicking a bar does the same for the *y*-axis. Use these bars to change the *x*- and *y*-axes back to *sepallength* and *petalwidth*.

The *Jitter* slider displaces the cross for each instance randomly from its true position, and can reveal situations where instances lie on top of one another. Experiment a little by moving the slider.

The *Select Instance* button and the *Reset*, *Clear*, and *Save* buttons let you modify the dataset. Certain instances can be selected and the others removed. Try the Rectangle option: Select an area by left-clicking and dragging the mouse. The *Reset* button changes into a *Submit* button. Click it, and all instances outside the rectangle are deleted. You could use *Save* to save the modified dataset to a file. *Reset* restores the original dataset.

## The Classify Panel

Now we apply a classifier to the weather data. Load the weather data again. Go to the Preprocess panel, click the *Open file* button, and select *weather.nominal.arff* from the data directory. Then switch to the Classify panel (shown in Figure 11.4(b)) by clicking the *Classify* tab at the top of the window.

### *Using the C4.5 Classifier*

As you learned in Chapter 11 (page 410), the C4.5 algorithm for building decision trees is implemented in Weka as a classifier called *J48*. Select it by clicking the *Choose*

button near the top of the *Classify* tab. A dialog window appears showing various types of classifier. Click the *trees* entry to reveal its subentries, and click *J48* to choose that classifier. Classifiers, like filters, are organized in a hierarchy: *J48* has the full name *weka.classifiers.trees.J48*.

The classifier is shown in the text box next to the *Choose* button: It now reads *J48 –C 0.25 –M 2*. This text gives the default parameter settings for this classifier, which in this case rarely require changing to obtain good performance.

For illustrative purposes we evaluate the performance using the training data, which has been loaded in the Preprocess panel—this is *not* generally a good idea because it leads to unrealistically optimistic performance estimates. Choose *Use training set* from the *Test options* part of the Classify panel. Once the test strategy has been set, the classifier is built and evaluated by pressing the *Start* button. This processes the training set using the currently selected learning algorithm, C4.5 in this case. Then it classifies all the instances in the training data and outputs performance statistics. These are shown in Figure 17.2(a).

### *Interpreting the Output*

The outcome of training and testing appears in the Classifier Output box on the right. Scroll through the text and examine it. First, look at the part that describes the decision tree, reproduced in Figure 17.2(b). This represents the decision tree that was built, including the number of instances that fall under each leaf. The textual representation is clumsy to interpret, but Weka can generate an equivalent graphical version.

Here's how to get the graphical tree. Each time the *Start* button is pressed and a new classifier is built and evaluated, a new entry appears in the Result List panel in the lower left corner of Figure 17.2(a). To see the tree, right-click on the entry *trees.J48* that has just been added to the result list and choose *Visualize tree*. A window pops up that shows the decision tree in the form illustrated in Figure 17.3. Right-click a blank spot in this window to bring up a new menu enabling you to auto-scale the view. You can pan around by dragging the mouse.

Now look at the rest of the information in the Classifier Output area. The next two parts of the output report on the quality of the classification model based on the chosen test option.

This text states how many and what proportion of test instances have been correctly classified:

```
Correctly Classified Instances   14   100%
```

This is the accuracy of the model on the data used for testing. In this case it is completely accurate (100%), which is often the case when the training set is used for testing.

At the bottom of the output is the confusion matrix:

```
=== Confusion Matrix ===

 a b   <- classified as
 9 0 | a = yes
 0 5 | b = no
```

**(a)**

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

outlook = sunny
|   humidity = high: no (3.0)
|   humidity = normal: yes (2.0)
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
```

**(b)**

**FIGURE 17.2**

Output after building and testing the classifier: (a) screenshot and (b) decision tree.

Each element in the matrix is a count of instances. Rows represent the true classes, and columns represent the predicted classes. As you can see, all 9 *yes* instances have been predicted as *yes*, and all 5 *no* instances as *no*.

**Exercise 17.1.8.** How would this instance be classified using the decision tree?

outlook = sunny, temperature = cool, humidity = high, windy = TRUE

**FIGURE 17.3**

The decision tree that has been built.

### *Setting the Test Method*

When the *Start* button is pressed, the selected learning algorithm is run and the dataset that was loaded in the Preprocess panel is used with the selected test protocol. For example, in the case of tenfold cross-validation this involves running the learning algorithm 10 times to build and evaluate 10 classifiers. A model built from the *full* training set is then printed into the Classifier Output area: This may involve running the learning algorithm one final time. The remainder of the output depends on the test protocol that was chosen using test options; these options were discussed in Section 11.1.

**Exercise 17.1.9.** Load the iris data using the Preprocess panel. Evaluate C4.5 on this data using (a) the training set and (b) cross-validation. What is the estimated percentage of correct classifications for (a) and (b)? Which estimate is more realistic?

### *Visualizing Classification Errors*

Right-click the *trees.J48* entry in the result list and choose *Visualize classifier errors*. A scatter plot window pops up. Instances that have been classified correctly are marked by little crosses; ones that are incorrect are marked by little squares.

**Exercise 17.1.10.** Use the *Visualize classifier errors* function to find the wrongly classified test instances for the cross-validation performed in Exercise 17.1.9. What can you say about the location of the errors?

## 17.2 NEAREST-NEIGHBOR LEARNING AND DECISION TREES

In this section you will experiment with nearest-neighbor classification and decision tree learning. For most of it, a real-world forensic glass classification dataset is used.

We begin by taking a preliminary look at the dataset. Then we examine the effect of selecting different attributes for nearest-neighbor classification. Next we study class noise and its impact on predictive performance for the nearest-neighbor method. Following that we vary the training set size, both for nearest-neighbor classification and for decision tree learning. Finally, you are asked to interactively construct a decision tree for an image segmentation dataset.

Before continuing you should review in your mind some aspects of the classification task:

- How is the accuracy of a classifier measured?
- To make a good classifier, are all the attributes necessary?
- What is class noise, and how would you measure its effect on learning?
- What is a learning curve?
- If you, personally, had to invent a decision tree classifier for a particular dataset, how would you go about it?

### The Glass Dataset

The glass dataset *glass.arff* from the U.S. Forensic Science Service contains data on six types of glass. Glass is described by its refractive index and the chemical elements that it contains; the the aim is to classify different types of glass based on these features. This dataset is taken from the UCI datasets, which have been collected by the University of California at Irvine and are freely available on the Web. They are often used as a benchmark for comparing data mining algorithms.

Find the dataset *glass.arff* and load it into the Explorer interface. For your own information, answer the following exercises, which review material covered in the previous section.

**Exercise 17.2.1.** How many attributes are there in the dataset? What are their names? What is the class attribute? Run the classification algorithm *IBk* (*weka.classifiers.lazy.IBk*). Use cross-validation to test its performance, leaving the number of folds at the default value of 10. Recall that you can examine the classifier options in the Generic Object Editor window that pops up when you click the text beside the *Choose* button. The default value of the KNN field is 1: This sets the number of neighboring instances to use when classifying.

**Exercise 17.2.2.** What is the accuracy of *IBk* (given in the Classifier Output box)? Run *IBk* again, but increase the number of neighboring instances to $k = 5$ by entering this value in the KNN field. Here and throughout this section, continue to use cross-validation as the evaluation method.

**Exercise 17.2.3**. What is the accuracy of *IBk* with five neighboring instances ($k = 5$)?

## Attribute Selection

Now we investigate which subset of attributes produces the best cross-validated classification accuracy for the *IBk* algorithm on the glass dataset. Weka contains automated attribute selection facilities, which are examined in a later section, but it is instructive to do this manually.

Performing an exhaustive search over all possible subsets of the attributes is infeasible (why?), so we apply the backward elimination procedure described in Section 7.1 (page 311). To do this, first consider dropping each attribute individually from the full dataset, and run a cross-validation for each reduced version. Once you have determined the best eight-attribute dataset, repeat the procedure with this reduced dataset to find the best seven-attribute dataset, and so on.

**Exercise 17.2.4.** Record in Table 17.1 the best attribute set and the greatest accuracy obtained in each iteration. The best accuracy obtained in this process is quite a bit higher than the accuracy obtained on the full dataset.

**Exercise 17.2.5.** Is this best accuracy an unbiased estimate of accuracy on future data? Be sure to explain your answer. (*Hint:* To obtain an unbiased estimate of accuracy on future data, we must not look at the test data *at all*

**Table 17.1** Accuracy Obtained Using *IBk*, for Different Attribute Subsets

| Subset Size (No. of Attributes) | Attributes in "Best" Subset | Classification Accuracy |
|---|---|---|
| 9 | | |
| 8 | | |
| 7 | | |
| 6 | | |
| 5 | | |
| 4 | | |
| 3 | | |
| 2 | | |
| 1 | | |
| 0 | | |

when producing the classification model for which the estimate is being obtained.)

## Class Noise and Nearest-Neighbor Learning

Nearest-neighbor learning, like other techniques, is sensitive to noise in the training data. In this section we inject varying amounts of class noise into the data and observe the effect on classification performance.

You can flip a certain percentage of class labels in the data to a randomly chosen other value using an unsupervised attribute filter called *AddNoise*, in *weka.filters.unsupervised.attribute*. However, for this experiment it is important that the test data remains unaffected by class noise. Filtering the training data without filtering the test data is a common requirement, and is achieved using a metalearner called *FilteredClassifier*, in *weka.classifiers.meta*, as described near the end of Section 11.3 (page 444). This metalearner should be configured to use *IBk* as the classifier and *AddNoise* as the filter. *FilteredClassifier* applies the filter to the data before running the learning algorithm. This is done in two batches: first the training data and then the test data. The *AddNoise* filter only adds noise to the first batch of data it encounters, which means that the test data passes through unchanged.

**Exercise 17.2.6.** Record in Table 17.2 the cross-validated accuracy estimate of *IBk* for 10 different percentages of class noise and neighborhood sizes $k = 1$, $k = 3$, $k = 5$ (determined by the value of $k$ in the $k$-nearest-neighbor classifier).
**Exercise 17.2.7.** What is the effect of increasing the amount of class noise?
**Exercise 17.2.8.** What is the effect of altering the value of $k$?

**Table 17.2** Effect of Class Noise on *IBk*, for Different Neighborhood Sizes

| Percentage Noise | $k = 1$ | $k = 3$ | $k = 5$ |
|---|---|---|---|
| 0% | | | |
| 10% | | | |
| 20% | | | |
| 30% | | | |
| 40% | | | |
| 50% | | | |
| 60% | | | |
| 70% | | | |
| 80% | | | |
| 90% | | | |
| 100% | | | |

**Table 17.3** Effect of Training Set Size on *IBk* and *J48*

| Percentage of Training Set | *IBk* | *J48* |
|---|---|---|
| 10% | | |
| 20% | | |
| 30% | | |
| 40% | | |
| 50% | | |
| 60% | | |
| 70% | | |
| 80% | | |
| 90% | | |
| 100% | | |

## Varying the Amount of Training Data

This section examines learning curves, which show the effect of gradually increasing the amount of training data. Again, we use the glass data, but this time with both *IBk* and the C4.5 decision tree learners, implemented in Weka as *J48*.

To obtain learning curves, use *FilteredClassifier* again, this time in conjunction with *weka.filters.unsupervised.instance.Resample*, which extracts a certain specified percentage of a given dataset and returns the reduced dataset.[1] Again, this is done only for the first batch to which the filter is applied, so the test data passes unmodified through the *FilteredClassifier* before it reaches the classifier.

**Exercise 17.2.9.** Record in Table 17.3 the data for learning curves for both the one-nearest-neighbor classifier (i.e., *IBk* with $k = 1$) and *J48*.
**Exercise 17.2.10.** What is the effect of increasing the amount of training data?
**Exercise 17.2.11.** Is this effect more pronounced for *IBk* or *J48*?

## Interactive Decision Tree Construction

One of Weka's classifiers is interactive: It lets the user—you!—construct your own classifier. Here's a competition: Who can build a classifier with the highest predictive accuracy?

Follow the procedure described in Section 11.2 (page 424). Load the file *segment-challenge.arff* (in the data folder that comes with the Weka distribution). This dataset

[1]This filter performs sampling with replacement, rather than sampling without replacement, but the effect is minor and we will ignore it here.

has 20 attributes and 7 classes. It is an image segmentation problem, and the task is to classify images into seven different groups based on properties of the pixels.

Set the classifier to *UserClassifier*, in the *weka.classifiers.trees* package. We use a separate test set (performing cross-validation with *UserClassifier* is incredibly tedious!), so in the *Test options* box choose the *Supplied test set* option and click the *Set* button. A small window appears in which you choose the test set. Click *Open file* and browse to the file *segment-test.arff* (also in the Weka distribution's data folder). On clicking *Open*, the small window updates to show the number of attributes (20) in the data. The number of instances is not displayed because test instances are read incrementally (so that the Explorer interface can process larger test files than can be accommodated in main memory).

Click *Start*. *UserClassifier* differs from all other classifiers: It opens a special window and waits for you to build your own classifier in it. The tabs at the top of the window switch between two views of the classifier. The *Tree visualizer* shows the current state of your tree, and the nodes give the number of class values there. The aim is to come up with a tree of which the leaf nodes are as pure as possible. To begin with, the tree has just one node—the root node—containing all the data. More nodes will appear when you proceed to split the data in the *Data visualizer*.

Click the *Data visualizer* tab to see a two-dimensional plot in which the data points are color-coded by class, with the same facilities as the Visualize panel discussed in Section 17.1. Try different combinations of *x*- and *y*-axes to get the clearest separation you can find between the colors. Having found a good separation, you then need to select a region in the plot: This will create a branch in the tree. Here's a hint to get you started: Plot *region-centroid-row* on the *x*-axis and *intensity-mean* on the *y*-axis (the display is shown in Figure 11.14(a)); you can see that the red class (*sky*) is nicely separated from the rest of the classes at the top of the plot.

There are four tools for selecting regions in the graph, chosen using the dropdown menu below the *y*-axis selector. *Select Instance* identifies a particular instance. *Rectangle* (shown in Figure 11.14(a)) allows you to drag out a rectangle on the graph. With *Polygon* and *Polyline* you build a free-form polygon or draw a free-form polyline (left-click to add a vertex and right-click to complete the operation).

When you have selected an area using any of these tools, it turns gray. (In Figure 11.14(a) the user has defined a rectangle.) Clicking the *Clear* button cancels the selection without affecting the classifier. When you are happy with the selection, click *Submit*. This creates two new nodes in the tree, one holding all the instances covered by the selection and the other holding all remaining instances. These nodes correspond to a binary split that performs the chosen geometric test.

Switch back to the *Tree visualizer* view to examine the change in the tree. Clicking on different nodes alters the subset of data that is shown in the *Data visualizer* section. Continue adding nodes until you obtain a good separation of the classes—that is, the leaf nodes in the tree are mostly pure. Remember, however, that you should not overfit the data because your tree will be evaluated on a separate test set.

When you are satisfied with the tree, right-click any blank space in the *Tree visualizer* view and choose *Accept The Tree*. Weka evaluates the tree against the test set and outputs statistics that show how well you did.

**Exercise 17.2.12.** You are competing for the best accuracy score of a hand-built *UserClassifier* produced on the *segment-challenge* dataset and tested on the *segment-test* set. Try as many times as you like. When you have a good score (anything close to 90% correct or better), right-click the corresponding entry in the Result list, save the output using *Save result buffer*, and copy it into your answer for this exercise. Then run *J48* on the data to see how well an automatic decision tree learner performs on the task.

## 17.3 CLASSIFICATION BOUNDARIES

In this section we examine the classification boundaries that are produced by different types of models. To do this, we use Weka's *Boundary Visualizer*, which is not part of the Explorer interface. To find it, start up the Weka GUI Chooser as usual from the Windows Start menu (on Linux or the Mac, double-click *weka.jar* or *weka.app*, respectively) and select *BoundaryVisualizer* from the *Visualization* menu at the top.

The boundary visualizer shows a two-dimensional plot of the data and is most appropriate for datasets with two numeric attributes. We will use a version of the iris data without the first two attributes. To create this, start up the Explorer interface, load *iris.arff* using the *Open file* button, and remove the first two attributes (*sepallength* and *sepalwidth*) by selecting them and clicking the *Remove* button that appears at the bottom. Then save the modified dataset to a file (using *Save*) called, say, *iris.2D.arff*.

Now leave the Explorer interface and open this file for visualization using the boundary visualizer's *Open file* button. Initially, the plot just shows the data in the dataset.

### Visualizing 1R

The purpose of the boundary visualizer is to show the predictions of a given model for every possible combination of attribute values—that is, for every point in the two-dimensional space. The points are color-coded according to the prediction the model generates. We will use this to investigate the decision boundaries that different classifiers generate for the reduced iris dataset.

Start with the 1R rule learner. Use the *Choose* button of the boundary visualizer to select *weka.classifiers.rules.OneR*. Make sure you tick *Plot training data*; otherwise, only the predictions will be plotted. Then click the *Start* button. The program starts plotting predictions in successive scan lines. Click the *Stop* button once the

plot has stabilized—as soon as you like, in this case—and the training data will be superimposed on the boundary visualization.

**Exercise 17.3.1.** Explain the plot based on what you know about 1R. (*Hint:* Use the Explorer interface to look at the rule set that 1R generates for this data.)
**Exercise 17.3.2.** Study the effect of the *minBucketSize* parameter on the classifier by regenerating the plot with values of 1, and then 20, and then some critical values in between. Describe what you see, and explain it. (*Hint:* You could speed things up by using the Explorer interface to look at the rule sets.)

Now answer the following questions by thinking about the internal workings of 1R. (*Hint:* It will probably be fastest to use the Explorer interface to look at the rule sets.)

**Exercise 17.3.3.** You saw earlier that when visualizing 1R the plot always has three regions. But why aren't there more for small bucket sizes (e.g., 1)? Use what you know about 1R to explain this apparent anomaly.
**Exercise 17.3.4.** Can you set *minBucketSize* to a value that results in less than three regions? What is the smallest possible number of regions? What is the smallest value for *minBucketSize* that gives this number of regions? Explain the result based on what you know about the iris data.

## Visualizing Nearest-Neighbor Learning

Now let's examine the classification boundaries created by the nearest-neighbor method. Use the boundary visualizer's *Choose* button to select the *IBk* classifier (*weka.classifiers.lazy.IBk*) and plot its decision boundaries for the reduced iris data.

*OneR*'s predictions are categorical: For each instance, they predict one of the three classes. In contrast, *IBk* outputs probability estimates for each class, and the boundary visualizer uses them to mix the colors red, green, and blue that correspond to the three classes. *IBk* estimates class probabilities by looking at the set of $k$-nearest neighbors of a test instance and counting the number in each class.

**Exercise 17.3.5.** With $k = 1$, which is the default value, it seems that the set of $k$-nearest neighbors could have only one member and therefore the color will always be pure red, green, or blue. Looking at the plot, this is indeed almost always the case: There is no mixing of colors because one class gets a probability of 1 and the others a probability of 0. Nevertheless, there is a small area in the plot where two colors are in fact mixed. Explain this. (*Hint:* Examine the data carefully using the Explorer interface's Visualize panel.)
**Exercise 17.3.6.** Experiment with different values of $k$, say 5 and 10. Describe what happens as $k$ increases.

## Visualizing *NaïveBayes*

Turn now to the *NaïveBayes* classifier. Its assumption that attributes are conditionally independent given a particular class value means that the overall class probability is obtained by simply multiplying the per-attribute conditional probabilities together (and taking into account the class prior probabilities as well). In other words, with two attributes, if you know the class probabilities along the *x*- and *y*-axes (and the class prior), you can calculate the value for any point in space by multiplying them together (and then normalizing). This is easy to understand if you visualize it as a boundary plot.

Plot the predictions of *NaïveBayes*. But first discretize the attribute values. By default, Weka's *NaiveBayes* classifier assumes that the attributes are normally distributed given the class. You should override this by setting *useSupervisedDiscretization* to *true* using the Generic Object Editor window. This will cause *NaïveBayes* to discretize the numeric attributes in the data with a supervised discretization technique. In most practical applications of *NaïveBayes*, supervised discretization works better than the default method. It also produces a more comprehensible visualization, which is why we use it here.

**Exercise 17.3.7.** The plot that is generated by visualizing the predicted class probabilities of *NaïveBayes* for each pixel location is quite different from anything we have seen so far. Explain the patterns in it.

## Visualizing Decision Trees and Rule Sets

Decision trees and rule sets are similar to nearest-neighbor learning in the sense that they are quasi-universal: In principle, they can approximate any decision boundary arbitrarily closely. In this section, we look at the boundaries generated by *JRip* and *J48*.

Generate a plot for *JRip*, with default options.

**Exercise 17.3.8.** What do you see? Relate the plot to the output of the rules that you get by processing the data in the Explorer.

**Exercise 17.3.9.** The *JRip* output assumes that the rules will be executed in the correct sequence. Write down an equivalent set of rules that achieves the same effect regardless of the order in which they are executed. Generate a plot for *J48*, with default options.

**Exercise 17.3.10.** What do you see? Again, relate the plot to the output that you get by processing the data in the Explorer interface. One way to control how much pruning *J48* performs is to adjust the minimum number of instances required in a leaf, *minNumObj*.

**Exercise 17.3.11.** Suppose you want to generate trees with 3, 2, and 1 leaf node, respectively. What are the exact ranges of values for *minNumObj* that achieve this, given default values for the other parameters?

### Messing with the Data

With the Boundary Visualizer you can modify the data by adding or removing points.

**Exercise 17.3.12.** Introduce some noise into the data and study the effect on the learning algorithms we looked at above. What kind of behavior do you observe for each algorithm as you introduce more noise?

## 17.4 PREPROCESSING AND PARAMETER TUNING

Now we look at some useful preprocessing techniques, which are implemented as filters, as well as a method for automatic parameter tuning.

### Discretization

As we know, there are two types of discretization techniques: *unsupervised* ones, which are "class blind," and *supervised* ones, which take the class value of the instances into account when creating intervals. Weka's main unsupervised method for discretizing numeric attributes is *weka.filters.unsupervised.attribute.Discretize*. It implements these two methods: equal-width (the default) and equal-frequency discretization.

Find the glass dataset *glass.arff* and load it into the Explorer interface. Apply the unsupervised discretization filter in the two different modes explained previously.

**Exercise 17.4.1.** What do you observe when you compare the histograms obtained? The one for equal-frequency discretization is quite skewed for some attributes. Why?

The main *supervised* technique for discretizing numeric attributes is *weka.filters.supervised.attribute.Discretize*. Locate the iris data, load it, apply the supervised discretization scheme, and look at the histograms obtained. Supervised discretization strives to create intervals within which the class distribution is consistent, although the distributions vary from one interval to the next.

**Exercise 17.4.2.** Based on the histograms obtained, which of the discretized attributes would you consider to be most predictive? Reload the glass data and apply supervised discretization to it.

**Exercise 17.4.3.** For some attributes there is only a single bar in the histogram. What does that mean?

Discretized attributes are normally coded as nominal attributes, with one value per range. However, because the ranges are ordered, a discretized attribute is actually on an ordinal scale. Both filters have the ability to create binary attributes rather than multivalued ones, by setting the option *makeBinary* to *true*.

**Exercise 17.4.4.** Choose one of the filters and use it to create binary attributes. Compare the result with the output generated when *makeBinary* is *false*. What do the binary attributes represent?

## More on Discretization

Here we examine the effect of discretization when building a *J48* decision tree for the data in *ionosphere.arff*. This dataset contains information about radar signals returned from the ionosphere. "Good" samples are those showing evidence of some type of structure in the ionosphere, while for "bad" ones the signals pass directly through the ionosphere. For more details, take a look at the comments in the ARFF file. Begin with unsupervised discretization.

**Exercise 17.4.5.** For *J48*, compare cross-validated accuracy and the size of the trees generated for (1) the raw data, (2) data discretized by the unsupervised discretization method in default mode, and (3) data discretized by the same method with binary attributes.

Now turn to supervised discretization. Here a subtle issue arises, discussed near the end of Section 11.3 (page 432). If Exercise 17.4.5 were simply repeated using a supervised discretization method, the result would be overoptimistic. In effect, because cross-validation is used for evaluation, *the data in the test set has been taken into account when determining the discretization intervals*. This does not give a fair estimate of performance on fresh data.

To evaluate supervised discretization fairly, use *FilteredClassifier* from Weka's metalearners. This builds the filter using the training data only, and then evaluates it on the test data using the discretization intervals computed for the training data. After all, that is how you would have to process fresh data in practice.

**Exercise 17.4.6.** Using *FilteredClassifier* and *J48*, compare cross-validated accuracy and the size of the trees generated for (4) supervised discretization in default mode, and (5) supervised discretization with binary attributes.
**Exercise 17.4.7.** Compare these with the results for the raw data from Exercise 17.4.5. How can decision trees generated from discretized data possibly be better predictors than ones built from raw numeric data?

## Automatic Attribute Selection

In most practical applications of supervised learning not all attributes are equally useful for predicting the target. For some learning schemes, redundant and/or irrelevant attributes can result in less accurate models. As you found in Section 17.2, it is tedious to identify useful attributes in a dataset manually; automatic attribute selection methods are usually more appropriate.

Attribute selection methods can be divided into filter and wrapper methods (see Section 7.1, page 308). The former apply a computationally efficient heuristic to measure the quality of a subset of attributes; the latter measure the quality of an attribute subset by building and evaluating an actual classification model, which is more expensive but often delivers superior performance.

The Explorer interface's Select attributes panel applies attribute selection methods to datasets. The default is to use *CfsSubsetEval*, described in Section 11.8 (page 488), which evaluates subsets of attributes. An alternative is to evaluate attributes individually using an evaluator like *InfoGainAttributeEval* (see Section 11.8, page 491) and then rank them by applying a special "search" method, namely the *Ranker*, as described Section 11.8 (page 490).

**Exercise 17.4.8.** Apply the ranking technique to the labor negotiations data in *labor.arff* to determine the four most important attributes based on information gain.[2]

*CfsSubsetEval* aims to identify a subset of attributes that are highly correlated with the target while not being strongly correlated with one another. It searches through the space of possible attribute subsets for the "best" one using the *BestFirst* search method by default, although other methods can be chosen. In fact, choosing *GreedyStepwise* and setting *searchBackwards* to *true* gives backward elimination, the search method you used manually in Section 17.2.

To use the wrapper method rather than a filter method, such as *CfsSubsetEval*, first select *WrapperSubsetEval* and then configure it by choosing a learning algorithm to apply and setting the number of cross-validation folds to use when evaluating it on each attribute subset.

**Exercise 17.4.9.** On the same data, run *CfsSubsetEval* for correlation-based selection, using the *BestFirst* search. Then run the wrapper method with *J48* as the base learner, again using the *BestFirst* search. Examine the attribute subsets that are output. Which attributes are selected by both methods? How do they relate to the output generated by ranking using information gain?

## More on Automatic Attribute Selection

The *Select attributes* panel allows us to gain insight into a dataset by applying attribute selection methods to it. However, as with supervised discretization, using this information to reduce a dataset becomes problematic if some of the reduced data is used for testing the model (as in cross-validation). Again, the reason is that we have

[2]Note that most evaluators, including *InfoGainAttributeEval* and *CfsSubsetEval*, discretize numeric attributes using Weka's supervised discretization method before evaluating them.

looked at the class labels in the test data while selecting attributes, and using the test data to influence the construction of a model biases the accuracy estimates obtained.

This can be avoided by dividing the data into training and test sets and applying attribute selection to the training set only. However, it is usually more convenient to use *AttributeSelectedClassifer*, one of Weka's metalearners, which allows an attribute selection method and a learning algorithm to be specified as part of a classification scheme. *AttributeSelectedClassifier* ensures that the chosen set of attributes is selected based on the training data only.

Now we test the three attribute selection methods from above in conjunction with *NaïveBayes*. *NaïveBayes* assumes independence of attributes, so attribute selection can be very helpful. You can see the effect of redundant attributes by adding multiple copies of an attribute using the filter *weka.filters.unsupervised.attribute.Copy* in the Preprocess panel. Each copy is obviously perfectly correlated with the original.

**Exercise 17.4.10.** Load the diabetes classification data in *diabetes.arff* and add copies of the first attribute. Measure the performance of *NaïveBayes* (with *useSupervisedDiscretization* turned on) using cross-validation after you have added each one. What do you observe?

Do the above three attribute selection methods, used in conjunction with *AttributeSelectedClassifier* and *NaïveBayes*, successfully eliminate the redundant attributes? Run each method from within *AttributeSelectedClassifier* to see the effect on cross-validated accuracy and check the attribute subset selected by each method. Note that you need to specify the number of ranked attributes to use for the *Ranker* method. Set this to 8 because the original diabetes data contains 8 attributes (excluding the class). Specify *NaïveBayes* as the classifier to be used inside the wrapper method because this is the classifier for which we want to select a subset.

**Exercise 17.4.11.** What can you say regarding the performance of the three attribute selection methods? Do they succeed in eliminating redundant copies? If not, why?

## Automatic Parameter Tuning

Many learning algorithms have parameters that can affect the outcome of learning. For example, the decision tree learner C4.5 has two parameters that influence the amount of pruning (we saw one, the minimum number of instances required in a leaf, in Section 17.3). The *k*-nearest-neighbor classifier *IBk* has a parameter (*k*) that sets the neighborhood size. But manually tweaking parameter settings is tedious, just like manually selecting attributes, and presents the same problem: The test data must not be used when selecting parameters; otherwise, the performance estimate will be biased.

Weka's metalearner *CVParameterSelection* searches for the best parameter settings by optimizing cross-validated accuracy on the training data. By default, each setting is evaluated using tenfold cross-validation. The parameters to optimize are specified using the *CVParameters* field in the Generic Object Editor window. For each parameter, three pieces of information must be supplied: (1) a string that names it using its letter code (which can be found in the Javadoc for the corresponding classifier—see Section 14.2, page 525); (2) a numeric range of values to evaluate; and (3) the number of steps to try in this range (note that the parameter is assumed to be numeric). Click on the *More* button in the Generic Object Editor window for more information and an example.

For the diabetes data used in the previous section, use *CVParameterSelection* in conjunction with *IBk* to select the best value for the neighborhood size, ranging from 1 to 10 in 10 steps. The letter code for the neighborhood size is *K*. The cross-validated accuracy of the parameter-tuned version of *IBk* is directly comparable with its accuracy using default settings because tuning is performed by applying inner cross-validation runs to find the best parameter value for each training set occurring in the outer cross-validation—and the latter yields the final performance estimate.

**Exercise 17.4.12.** What accuracy is obtained in each case? What value is selected for the parameter-tuned version based on cross-validation on the full data set? (*Note:* This value is output in the Classifier Output text area because, as mentioned earlier, the model that is output is the one built from the full dataset.)

Now consider parameter tuning for *J48*. If there is more than one parameter string in the *CVParameters* field, *CVParameterSelection* performs a grid search on the parameters simultaneously. The letter code for the pruning confidence parameter is *C*, and you should evaluate values from 0.1 to 0.5 in five steps. The letter code for the minimum leaf size parameter is *M*, and you should evaluate values from 1 to 10 in 10 steps.

**Exercise 17.4.13.** Run *CVParameterSelection* to find the best parameter value setting. Compare the output you get to that obtained from *J48* with default parameters. Has accuracy changed? What about tree size? What parameter values were selected by *CVParameterSelection* for the model built from the full training set?

## 17.5 DOCUMENT CLASSIFICATION

Next we perform some experiments in document classification. The raw data is text, and this is first converted into a form suitable for learning by creating a dictionary of terms from all the documents in the training corpus and making a numeric

attribute for each term using Weka's unsupervised attribute filter *StringToWordVector*. There is also the class attribute, which gives the document's label.

## Data with String Attributes

The *StringToWordVector* filter assumes that the document text is stored in an attribute of type *String*—a nominal attribute without a prespecified set of values. In the filtered data, this is replaced by a fixed set of numeric attributes, and the class attribute is put at the beginning, as the first attribute.

To perform document classification, first create an ARFF file with a string attribute that holds the document's text—declared in the header of the ARFF file using *@attribute document string*, where *document* is the name of the attribute. A nominal attribute is also needed to hold the document's classification.

**Exercise 17.5.1.** Make an ARFF file from the labeled mini-documents in Table 17.4 and run *StringToWordVector* with default options on this data. How many attributes are generated? Now change the value of the option *minTermFreq* to 2. What attributes are generated now?

**Exercise 17.5.2.** Build a *J48* decision tree from the last version of the data you generated.

**Exercise 17.5.3.** Classify the new documents in Table 17.5 based on the decision tree generated from the documents in Table 17.4. To apply the same

**Table 17.4** Training Documents

| Document Text | Classification |
|---|---|
| The price of crude oil has increased significantly | yes |
| Demand for crude oil outstrips supply | yes |
| Some people do not like the flavor of olive oil | no |
| The food was very oily | no |
| Crude oil is in short supply | yes |
| Use a bit of cooking oil in the frying pan | no |

**Table 17.5** Test Documents

| Document Text | Classification |
|---|---|
| Oil platforms extract crude oil | unknown |
| Canola oil is supposed to be healthy | unknown |
| Iraq has significant oil reserves | unknown |
| There are different types of cooking oil | unknown |

filter to both training and test documents, use *FilteredClassifier*, specifying the *StringToWordVector* filter and *J48* as the base classifier. Create an ARFF file from Table 17.5, using question marks for the missing class labels. Configure *FilteredClassifier* using default options for *StringToWordVector* and *J48*, and specify your new ARFF file as the test set. Make sure that you select *Output predictions* under *More options* in the Classify panel. Look at the model and the predictions it generates, and verify that they are consistent. What are the predictions?

## Classifying Actual Documents

A standard collection of newswire articles is widely used for evaluating document classifiers. *ReutersCorn-train.arff* and *ReutersGrain-train.arff* are training sets derived from this collection; *ReutersCorn-test.arff* and *ReutersGrain-test.arff* are corresponding test sets. The actual documents in the corn and grain data are the same; only the labels differ. In the first dataset, articles concerning corn-related issues have a class value of 1 and the others have 0; the aim is to build a classifier that identifies "corny" articles. In the second, the labeling is performed with respect to grain-related issues; the aim is to identify "grainy" articles.

**Exercise 17.5.4.** Build classifiers for the two training sets by applying *FilteredClassifier* with *StringToWordVector* using (1) *J48* and (2) *NaiveBayesMultinomial*, evaluating them on the corresponding test set in each case. What percentage of correct classifications is obtained in the four scenarios? Based on the results, which classifier would you choose?

Other evaluation metrics are used for document classification besides the percentage of correct classifications: They are tabulated under *Detailed Accuracy By Class* in the Classifier Output area—the number of true positives (TP), false positives (FP), true negatives (TN), and false negatives (FN). The statistics output by Weka are computed as specified in Table 5.7; the *F*-measure is mentioned in Section 5.7 (page 175).

**Exercise 17.5.5.** Based on the formulas in Table 5.7, what are the best possible values for each of the output statistics? Describe when these values are attained.

The Classifier Output also gives the ROC area (also known as AUC), which, as explained in Section 5.7 (page 177), is the probability that a randomly chosen positive instance in the test data is ranked above a randomly chosen negative instance, based on the ranking produced by the classifier. The best outcome is that all positive examples are ranked above all negative examples, in which case the AUC is 1. In the worst case it is 0. In the case where the ranking is essentially random, the AUC is 0.5, and if it is significantly less than this the classifier has performed anti-learning!

**Exercise 17.5.6.** Which of the two classifiers used above produces the best AUC for the two Reuters datasets? Compare this to the outcome for percent correct. What do the different outcomes mean?

The ROC curves discussed in Section 5.7 (page 172) can be generated by right-clicking on an entry in the result list and selecting *Visualize threshold curve*. This gives a plot with FP Rate on the *x*-axis and TP Rate on the *y*-axis. Depending on the classifier used, this plot can be quite smooth or it can be fairly irregular.

**Exercise 17.5.7.** For the Reuters dataset that produced the most extreme difference in Exercise 17.5.6, look at the ROC curves for class 1. Make a very rough estimate of the area under each curve, and explain it in words.
**Exercise 17.5.8.** What does the ideal ROC curve corresponding to perfect performance look like?

Other types of threshold curves can be plotted, such as a precision–recall curve with Recall on the *x*-axis and Precision on the *y*-axis.

**Exercise 17.5.9.** Change the axes to obtain a precision–recall curve. What is the shape of the ideal precision–recall curve, corresponding to perfect performance?

## Exploring the *StringToWordVector* Filter

By default, the *StringToWordVector* filter simply makes the attribute value in the transformed dataset 1 or 0 for all single-word terms, depending on whether the word appears in the document or not. However, as mentioned in Section 11.3 (page 439), there are many options:

- *outputWordCounts* causes actual word counts to be output.
- *IDFTransform* and *TFTransform*: When both are set to *true*, term frequencies are transformed into TF × IDF values.
- *stemmer* gives a choice of different word-stemming algorithms.
- *useStopList* lets you determine whether or not stopwords are deleted.
- *tokenizer* allows different tokenizers for generating terms, such as one that produces word *n*-grams instead of single words.

There are several other useful options. For more information, click on *More* in the Generic Object Editor window.

**Exercise 17.5.10.** Experiment with the options that are available. What options give a good AUC value for the two datasets above, using *NaiveBayesMultinomial* as the classifier?

Not all of the attributes (i.e., terms) are important when classifying documents. The reason is that many words are irrelevant for determining an article's topic. Weka's *AttributeSelectedClassifier*, using ranking with *InfoGainAttributeEval* and the *Ranker* search, can eliminate less useful attributes. As before, *FilteredClassifier* should be used to transform the data before passing it to *AttributeSelectedClassifier*.

**Exercise 17.5.11.** Experiment with this, using default options for *StringToWordVector* and *NaiveBayesMultinomial* as the classifier. Vary the number of the most informative attributes that are selected from the information gain–based ranking by changing the value of the *numToSelect* field in the *Ranker*. Record the AUC values you obtain. How many attributes give the best AUC for the two datasets discussed before? What are the best AUC values you managed to obtain?

## 17.6 MINING ASSOCIATION RULES

In order to get some experience with association rules, we work with *Apriori*, the algorithm described in Section 4.5 (page 144). As you will discover, it can be challenging to extract useful information using this algorithm.

### Association-Rule Mining

To get a feel for how to apply *Apriori*, start by mining rules from the *weather.nominal.arff* data that was used in Section 17.1. Note that this algorithm expects data that is purely nominal: If present, numeric attributes must be discretized first. After loading the data in the Preprocess panel, click the *Start* button in the Associate panel to run *Apriori* with default options. It outputs 10 rules, ranked according to the confidence measure given in parentheses after each one (they are listed in Figure 11.16). As we explained in Chapter 11 (page 430), the number following a rule's antecedent shows how many instances satisfy the antecedent; the number following the conclusion shows how many instances satisfy the entire rule (this is the rule's "support"). Because both numbers are equal for all 10 rules, the confidence of every rule is exactly 1.

In practice, it can be tedious to find minimum support and confidence values that give satisfactory results. Consequently, as explained in Chapter 11, Weka's *Apriori* runs the basic algorithm several times. It uses the same user-specified minimum confidence value throughout, given by the *minMetric* parameter. The support level is expressed as a proportion of the total number of instances (14 in the case of the weather data), as a ratio between 0 and 1. The minimum support level starts at a certain value (*upperBoundMinSupport*, default 1.0). In each iteration the support is decreased by a fixed amount (*delta*, default 0.05, 5% of the instances) until either a certain number of rules has been generated (*numRules*, default 10 rules) or the support reaches a certain "minimum minimum" level (*lowerBoundMinSupport*,

default 0.1)—because rules are generally uninteresting if they apply to less than 10% of the dataset. These four values can all be specified by the user.

This sounds pretty complicated, so we will examine what happens on the weather data. The Associator output text area shows that the algorithm managed to generate 10 rules. This is based on a minimum confidence level of 0.9, which is the default and is also shown in the output. The *Number of cycles performed*, which is shown as 17, indicates that *Apriori* was actually run 17 times to generate these rules, with 17 different values for the minimum support. The final value, which corresponds to the output that was generated, is 0.15 (corresponding to $0.15 \times 14 \approx 2$ instances).

By looking at the options in the Generic Object Editor window, you can see that the initial value for the minimum support (*upperBoundMinSupport*) is 1 by default, and that delta is 0.05. Now, $1 - 17 \times 0.05 = 0.15$, so this explains why a minimum support value of 0.15 is reached after 17 iterations. Note that *upperBoundMinSupport* is decreased by delta *before* the basic *Apriori* algorithm is run for the first time.

The Associator output text area also shows how many frequent item sets were found, based on the last value of the minimum support that was tried (0.15 in this example). In this case, given a minimum support of two instances, there are 12 item sets of size 1, 47 item sets of size 2, 39 item sets of size 3, and six item sets of size 4. By setting *outputItemSets* to *true* before running the algorithm, all those different item sets and the number of instances that support them are shown. Try it out!

**Exercise 17.6.1.** Based on the output, what is the support for this item set?

outlook = rainy humidity = normal windy = FALSE play = yes

**Exercise 17.6.2.** Suppose you want to generate all rules with a certain confidence and minimum support. This can be done by choosing appropriate values for *minMetric*, *lowerBoundMinSupport*, and *numRules*. What is the total number of possible rules for the weather data for each combination of values in Table 17.6?

**Table 17.6** Number of Rules for Different Values of Minimum Confidence and Support

| Minimum Confidence | Minimum Support | Number of Rules |
|---|---|---|
| 0.9 | 0.3 | |
| 0.9 | 0.2 | |
| 0.9 | 0.1 | |
| 0.8 | 0.3 | |
| 0.8 | 0.2 | |
| 0.8 | 0.1 | |
| 0.7 | 0.3 | |
| 0.7 | 0.2 | |
| 0.7 | 0.1 | |

*Apriori* has some further parameters. If *significanceLevel* is set to a value between 0 and 1, the association rules are filtered based on a $\chi^2$ test with the chosen significance level. However, applying a significance test in this context is problematic because of the multiple comparison problem: If a test is performed hundreds of times for hundreds of association rules, it is likely that significant effects will be found just by chance—that is, an association seems to be statistically significant when really it is not. Also, the $\chi^2$ test is inaccurate for small sample sizes (in this context, small support values).

There are alternative measures for ranking rules. As well as confidence, *Apriori* supports *lift*, *leverage*, and *conviction*, which can be selected using *metricType*. More information is available by clicking *More* in the Generic Object Editor window.

**Exercise 17.6.3.** Run *Apriori* on the weather data with each of the four rule-ranking metrics, and default settings otherwise. What is the top-ranked rule that is output for each metric?

## Mining a Real-World Dataset

Now consider a real-world dataset, *vote.arff*, which gives the votes of 435 U.S. congressmen on 16 key issues gathered in the mid-1980s, and also includes their party affiliation as a binary attribute. This is a purely nominal dataset with some missing values (corresponding to abstentions). It is normally treated as a classification problem, the task being to predict party affiliation based on voting patterns. However, association-rule mining can also be applied to this data to seek interesting associations. More information on the data appears in the comments in the ARFF file.

**Exercise 17.6.4.** Run *Apriori* on this data with default settings. Comment on the rules that are generated. Several of them are quite similar. How are their support and confidence values related?

**Exercise 17.6.5.** It is interesting to see that none of the rules in the default output involve *Class = republican*. Why do you think that is?

## Market Basket Analysis

In Section 1.3 (page 26) we introduced market basket analysis—analyzing customer purchasing habits by seeking associations in the items they buy when visiting a store. To do market basket analysis in Weka, each transaction is coded as an instance of which the attributes represent the items in the store. Each attribute has only one value: If a particular transaction does not contain it (i.e., the customer did not buy that item), this is coded as a missing value.

Your job is to mine supermarket checkout data for associations. The data in *supermarket.arff* was collected from an actual New Zealand supermarket. Take a look at this file using a text editor to verify that you understand the structure. The

main point of this exercise is to show you how difficult it is to find any interesting patterns in this type of data!

**Exercise 17.6.6.** Experiment with *Apriori* and investigate the effect of the various parameters described before. Write a brief report on the main findings of your investigation.

# WEKA

Computational Linguistics

# A Toy Example

- Demonstrate WEKA on a simple example:
  - 3 categories from "Newsgroups":
    - » misc.forsale
    - » rec.sport.hockey
    - » comp.graphics
  - 20 documents per category
  - Features:
    - » words converted to lowercase
    - » frequency 2 or more required
    - » stopwords removed

http://www.cs.vassar.edu/~cs366/data/weka_files/example_train.aarf

http://www.cs.vassar.edu/~cs366/data/weka_files/example_test.aarf

# 20 Newsgroups Data Set

- Content and structure:
  - approximately 20,000 newsgroup documents
    - » 19,997 originally
    - » 18,828 without duplicates
  - partitioned evenly across 20 different newsgroups
  - we are only using a subset
  - Some categories are strongly related (and thus hard to discriminate):

computers

| | | |
|---|---|---|
| comp.graphics<br>comp.os.ms-windows.misc<br>comp.sys.ibm.pc.hardware<br>comp.sys.mac.hardware<br>comp.windows.x | rec.autos<br>rec.motorcycles<br>rec.sport.baseball<br>rec.sport.hockey | sci.crypt<br>sci.electronics<br>sci.med<br>sci.space |
| misc.forsale | talk.politics.misc<br>talk.politics.guns<br>talk.politics.mideast | talk.religion.misc<br>alt.atheism<br>soc.religion.christian |

# Sample Posting: "talk.politics.guns"

from

subject

… writes:

From: cdt@sw . D. Tavares)
Subject: Re: Congress to review ATF's status

In article <C5vzHF.D5K@cbnews.cb.att.com>, lvc@cbnews.cb.att.com (Larry Cipriani) writes:

> WASHINGTON (UPI) -- As part of its investigation of the deadly
> confrontation with a Texas cult, Congress will consider whether the
> Bureau of Alcohol, Tobacco and Firearms should be moved from the
> Treasury Department to the Justice Department, senators said Wednesday.
> The idea will be considered because of the violent and fatal events
> at the beginning and end of the agency's confrontation with the Branch
> Davidian

reply

Need special handling during feature extraction…

Of course. When the catbox begines to smell, simply transfer its contents into the potted plant in the foyer.

"Why Hillary! Your government smells so... FRESH!"
--

signature

cdt@rocket.sw.stratus.com --If you believe that I speak for my company,
OR cdt@vos.stratus.com write today for my special Investors' Packet...

# Explorer: Pre-Processing The Data

- WEKA can import data from:
  - files: ARFF, CSV, binary
  - URL
  - SQL database
- Pre-processing tools (filters) are used for:
  - Discretization, normalization, resampling, attribute selection, transforming and combining attributes, etc.

Weka Explorer
Preprocessing
Classify
Cluster
Associate
Select attributes
Visualize
Open file...
Open URL...
Open DB...
Generate...
Undo
Save...
Classification
Statistical attribute selection
Filter
Choose
None
Filter selection
Apply
Current relation
Relation: diverse
Instances: 75
Attributes: 101
Characteristics
Selected attribute
Name: edu
Missing: 0 (0%)
Distinct: 7
Type: Numeric
Unique: 1 (1%)
Statistic
Value
Minimum
0
Maximum
6
Mean
1.4
StdDev
1.489
Statistics about the values of the selected attribute
Attributes
All
None
Invert
Pattern
No.
Name
1 edu
2 subject
3 lines
4 organization
5 period
6 pp
7 play
8 ca
9 power
10 university
11 host
12 nntp-posting
13 d
14 re
15 team
16 g
17 me
18 scorer
19 pts
20 does
21 selanne
22 writes
23 article-i
24 second
25 go
26 lemieux
27 biggest
28 pittsburgh
29 sale
30 distribution
31 year
32 article
Manual attribute selection
List of attributes (last: class variable)
Remove
Class: newsgroup_class (Nom)
Visualize All
38
20
12
2
3
0
3
6
Frequency and categories for the selected attribute
Status
OK
Log
x 0

# Explorer: Building Classifiers

- Classifiers in WEKA are models for:
  - classification (predict a nominal class)
  - regression (predict a numerical quantity)
- Learning algorithms:
  - Naïve Bayes, decision trees, kNN, support vector machines, multi-layer perceptron, logistic regression, etc.
- Meta-classifiers:
  - cannot be used alone
  - always combined with a learning algorithm
  - examples: boosting, bagging etc.

Weka Explorer
Preprocess
Classify
Cluster
Associate
Select attributes
Visualize
Classifier
Choose
ZeroR
Choice of classifier
Test options
Use training set
Supplied test set
Set...
Cross-validation
Folds
10
Percentage split
%
66
More options...
(Nom) class
Start
Stop
Result list (right-click for options)
Classifier output
Cross-validation: split the data into e.g. 10 folds and 10 times train on 9 folds and test on the remaining one
The attribute whose value is to be predicted from the values of the remaining ones.
Default is the last attribute.
Here (in our toy example) it is named "class".
Status
OK
Log
x 0

Weka Explorer
Preprocess
Classify
Cluster
Associate
Select attributes
Visualize
Classifier
weka
classifiers
bayes
AODE
AODEsr
BayesianLogisticRegression
BayesNet
ComplementNaiveBayes
DMNBtext
HNB
NaiveBayes
NaiveBayesMultinomial
NaiveBayesMultinomialUpdateable
NaiveBayesSimple
NaiveBayesUpdateable
WAODE
functions
lazy
meta
mi
misc
Filter...
Remove filter
Close
Choosing a classifier
Status
OK
Log
x 0

Weka Explorer
Preprocess
Classify
Cluster
Associate
Select attributes
Visualize
Classifier
Choose
NaiveBayes
Test options
Use training set
Supplied test set
Set...
Cross-validation
Folds
10
Percentage split
%
66
More options...
(Nom) class
Start
Stop
Result list (right-click for options)
Classifier output
Status
OK
Log
x 0

Weka Explorer
Preprocess
Classify
Cluster
Associate
Select attributes
Visualize
Classifier
Choose
weka.gui.GenericObjectEditor
weka.classifiers.bayes.NaiveBayes
About
Class for a Naive Bayes classifier using estimator classes.
More
Capabilities
debug
False
displayModelInOldFormat
False
useKernelEstimator
False
useSupervisedDiscretization
False
Open...
Save...
OK
Cancel
Test options
Use trainin
Supplied t
Cross-vali
Percentage
(Nom) class
Start
Result list (righ
Status
OK
Log
x 0
displays synopsis and options
Outputs additional information
False: Gaussian
True: kernels (better)
numerical to nominal conversion by discretization

Weka Explorer

Preprocess | Classify | Cluster | Associate | Select attributes | Visualize

Classifier

Choose

weka.gui.GenericObjectEditor

weka.classifiers.bayes.NaiveBayes

About

Class for a Naive Bayes classifier using estimator classes.

More

Capabilities

| | |
|---|---|
| debug | False |
| displayModelInOldFormat | False |
| useKernelEstimator | True |
| useSupervisedDiscretization | False |

Open... | Save... | OK | Cancel

Test options

Use trainin

Supplied t

Cross-vali

Percentage

(Nom) class

Start

Result list (righ

Information

NAME
weka.classifiers.bayes.NaiveBayes

SYNOPSIS
Class for a Naive Bayes classifier using estimator classes. Numeric estimator precision values are chosen based on analysis of the training data. For this reason, the classifier is not an UpdateableClassifier (which in typical usage are initialized with zero training instances) -- if you need the UpdateableClassifier functionality, use the NaiveBayesUpdateable classifier. The NaiveBayesUpdateable classifier will use a default precision of 0.1 for numeric attributes when buildClassifier is called with zero training instances.

For more information on Naive Bayes classifiers, see

George H. John, Pat Langley: Estimating Continuous Distributions in Bayesian Classifiers. In: Eleventh Conference on Uncertainty in Artificial Intelligence, San Mateo, 338-345, 1995.

OPTIONS
debug -- If set to true, classifier may output additional info to the console.

displayModelInOldFormat -- Use old format for model output. The old format is better when there are many class values. The new format is better when there are fewer classes and many attributes.

useKernelEstimator -- Use a kernel estimator for numeric attributes rather than a normal distribution.

useSupervisedDiscretization -- Use supervised discretization to convert numeric attributes to nominal ones.

Status
OK

Log

x 0

Weka Explorer
Preprocess
Classify
Cluster
Associate
Select attributes
Visualize
Classifier
Choose
NaiveBayes -K
Test options
Use training set
Supplied test set
Set...
Cross-validation
Folds
10
Percentage split
%
66
More options...
(Nom) class
Start
Stop
Result list (right-click for options)
Classifier output
Status
OK
Log
x 0

Weka Explorer

Preprocess | Classify | Cluster | Associate | Select attributes | Visualize

**Classifier**

Choose **NaiveBayes**

**Test options**

- Use training set
- Supplied test set — Set...
- Cross-validation — Folds 10
- Percentage split — % 66

More options...

(Nom) class

Start | Stop

**Result list (right-click for options)**

13:50:15 - bayes.NaiveBayes
13:53:50 - bayes.NaiveBayes

**Classifier output**

```
  precision                    1                1                1

driver
  mean                         0              0.1                0
  std. dev.               0.3333           0.4359           0.3333
  weight sum                  20               20               20
  precision                    2                2                2

dropped
  mean                         0              0.1                0
  std. dev.               0.1667              0.3           0.1667
  weight sum                  20               20               20
  precision                    1                1                1

druce
  mean                         0              0.1                0
  std. dev.               0.3333           0.4359           0.3333
  weight sum                  20               20               20
  precision                    2                2                2

dsp
  mean                      0.05                0             0.05
  std. dev.               0.2179           0.1667           0.2179
  weight sum                  20               20               20
  precision                    1                1                1

dta
  mean                         0                0              0.1
  std. dev.               0.3333           0.3333           0.4359
  weight sum                  20               20               20
  precision                    2                2                2


Time taken to build model: 0.04 seconds

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

Correctly Classified Instances          43               71.6667 %
Incorrectly Classified Instances        17               28.3333 %
Kappa statistic                          0.575
Mean absolute error                      0.188
Root mean squared error                  0.4326
Relative absolute error                 42.3007 %
Root relative squared error             91.7785 %
Total Number of Instances               60

=== Detailed Accuracy By Class ===

               TP Rate   FP Rate   Precision   Recall  F-Measure   ROC Area  Class
                 0.65     0.225       0.591     0.65     0.619      0.829    misc.forsale
                 0.8      0.1         0.8       0.8      0.8        0.904    rec.sport.hockey
                 0.7      0.1         0.778     0.7      0.737      0.904    comp.graphics
Weighted Avg.    0.717    0.142       0.723     0.717    0.719      0.879

=== Confusion Matrix ===

  a  b  c   <-- classified as
 13  4  3 |  a = misc.forsale
  3 16  1 |  b = rec.sport.hockey
  6  0 14 |  c = comp.graphics
```

accuracy

different/easy class

Status
OK

Log x 0

# Confusion matrix

- Contains information about the actual and the predicted classification
- All measures can be derived from it:
  - accuracy: (a+d)/(a+b+c+d)
  - recall: d/(c+d) => R
  - precision: d/(b+d) => P
  - F-measure: 2PR/(P+R)
  - false positive (FP) rate: b/(a+b)
  - true negative (TN) rate: a/(a+b)
  - false negative (FN) rate: c/(c+d)
- These extend for more than 2 classes

| | | predicted | |
|---|---|---|---|
| | | – | + |
| true | – | a | b |
| | + | c | d |

# Predictions Output

Weka Explorer

Preprocess | Classify | Cluster | Associate | Select attributes | Visualize

**Classifier**

Choose — NaiveBayes

**Test options**

- Use training set
- Supplied test set — Set...
- Cross-validation — Folds 10
- Percentage split — % 66

More options...

(Nom) class

Start — Stop

**Result list (right-click for options)**

13:50:15 - bayes.NaiveBayes
13:53:50 - bayes.NaiveBayes

**Classifier output**

```
  precision                    1                1                1

driver
  mean                         0              0.1                0
  std. dev.               0.3333           0.4359           0.3333
  weight sum                  20               20               20
  precision                    2                2                2

dropped
  mean                         0              0.1                0
                                              0.3           0.1667
                                               20               20
                                                1                1

                                              0.1                0
                                            .4359           0.3333
                                               20               20
                                                2                2

                                                0             0.05
                                            .1667           0.2179
                                               20               20
                                                1                1

                                                0              0.1
                                            .3333           0.4359
                                               20               20

                                                          71.6667 %
                                                          28.3333 %

                                                5
                                                7 %
                                                5 %

                  Recall   F-Measure   ROC Area  Class
                  0.65       0.619       0.829    misc.forsale
                  0.8        0.904       0.904    rec.sport.hockey
                  0.7        0.737       0.904    comp.graphics
Weighted Avg.     0.717      0.719       0.879

=== Confusion Matrix ===

  a  b  c   <-- classified as
 13  4  3 |  a = misc.forsale
  3 16  1 |  b = rec.sport.hockey
  6  0 14 |  c = comp.graphics
```

**Classifier evaluation options**

- [x] Output model
- [x] Output per-class stats
- [ ] Output entropy evaluation measures
- [x] Output confusion matrix
- [x] Store predictions for visualization
- [x] Output predictions
- Output additional attributes
- [ ] Cost-sensitive evaluation
- Random seed for XVal / % Split: 1
- [ ] Preserve order for % Split
- [ ] Output source code — WekaClassifier

OK

Outputs the probability distribution for each example

Status: OK — Log — x 0

Weka Explorer

Preprocess | Classify | Cluster | Associate | Select attributes | Visualize

# Predictions Output

**Classifier**

Choose **NaiveBayes**

**Test options**

- Use training set
- Supplied test set — Set...
- Cross-validation — Folds 10
- Percentage split — % 66

More options...

(Nom) class

Start | Stop

**Result list (right-click for options)**

13:50:15 - bayes.NaiveBayes
13:53:50 - bayes.NaiveBayes
14:01:58 - bayes.NaiveBayes

**Classifier output**

```
=== Predictions on test data ===

inst#,     actual, predicted, error, probability distribution
     1 3:comp.gra 3:comp.gra          0       0      *1
     2 3:comp.gra 3:comp.gra          0       0      *1
     3 2:rec.spor 2:rec.spor          0      *1       0
     4 2:rec.spor 2:rec.spor          0      *1       0
     5 1:misc.for 1:misc.for         *1       0       0
     6 1:misc.for 1:misc.for         *1       0       0
     1 3:comp.gra 3:comp.gra          0       0      *1
     2 3:comp.gra 3:comp.gra          0       0      *1
     3 2:rec.spor 2:rec.spor          0      *1       0
     4 2:rec.spor 1:misc.for      +  *1       0       0
     5 1:misc.for 1:misc.for         *1       0       0
     6 1:misc.for 1:misc.for         *1       0       0
     1 3:comp.gra 3:comp.gra          0       0      *1
     2 3:comp.gra 3:comp.gra          0       0      *1
     3 2:rec.spor 1:misc.for      +  *1       0       0
     4 2:rec.spor 2:rec.spor          0      *1       0
     5 1:misc.for 1:misc.for         *1       0       0
     6 1:misc.for 1:misc.for         *1       0       0
     1 3:comp.gra 3:comp.gra          0       0      *1
     2 3:comp.gra 3:comp.gra          0       0      *1
     3 2:rec.spor 2:rec.spor          0      *1       0
     4 2:rec.spor 2:rec.spor          0      *1       0
     5 1:misc.for 2:rec.spor      +   0      *1       0
     6 1:misc.for 3:comp.gra      +   0       0      *1
     1 3:comp.gra 1:misc.for      +  *1       0       0
     2 3:comp.gra 1:misc.for      +  *1       0       0
     3 2:rec.spor 2:rec.spor          0      *1       0
     4 2:rec.spor 3:comp.gra      +   0       0      *1
     5 1:misc.for 1:misc.for         *1       0       0
     6 1:misc.for 1:misc.for         *1       0       0
     1 3:comp.gra 1:misc.for      +  *0.921   0       0.079
     2 3:comp.gra 1:misc.for      +  *1       0       0
     3 2:rec.spor 2:rec.spor          0      *1       0
     4 2:rec.spor 2:rec.spor          0      *1       0
     5 1:misc.for 1:misc.for         *1       0       0
     6 1:misc.for 2:rec.spor      +   0      *1       0
     1 3:comp.gra 3:comp.gra          0       0      *1
     2 3:comp.gra 3:comp.gra          0       0      *1
     3 2:rec.spor 2:rec.spor          0      *1       0
     4 2:rec.spor 2:rec.spor          0      *1       0
     5 1:misc.for 3:comp.gra      +   0       0      *1
     5 1:misc.for 3:comp.gra      +  *3       0      -2
     6 3:comp.gra 1:misc.for      +  *3.999   0       0.001
     2 3:comp.gra 3:comp.gra          0       0      *1
     3 2:rec.spor 1:misc.for      +  *1       0       0
     3 2:rec.spor 2:misc.for      +  -2      *3       0
     3 2:misc.for 2:misc.for         -2      *3       0
     5 1:misc.for 2:misc.for      +  -2      *3       0
     6 3:comp.gra 3:comp.gra      +   0      *3      -2
     2 3:comp.gra 1:misc.for      +  *1       0       0
     3 2:rec.spor 2:rec.spor          0      *1       0
     4 2:rec.spor 2:rec.spor          0      *1       0
     5 1:misc.for 2:rec.spor      +   0      *1       0
     5 1:misc.for 2:rec.spor      +   0      -2      *3
     6 3:comp.gra 3:comp.gra      +   0       0      *1
     2 3:comp.gra 3:comp.gra          0       0      *1
     2 3:comp.gra 3:comp.gra          0      *3      -2
     3 2:rec.spor 2:rec.spor          0      -2      *3
     3 2:misc.for 2:misc.for         -2      *3       0
     6 1:misc.for 1:misc.for         *1       0       0
```

Probability distribution for a wrong example: predicted 1 instead of 2

Naïve Bayes makes incorrect conditional independence assumptions and typically is over-confident in its prediction regardless of whether it is correct or not.

**Status**

OK

Log | x 0

Weka Explorer

Preprocess | Classify | Cluster | Associate | Select attributes | Visualize

**Classifier**

Choose — NaiveBayes

**Test options**

- Use training set
- Supplied test set — Set...
- Cross-validation — Folds 10
- Percentage split — % 66

More options...

(Nom) class

Start — Stop

**Result list (right-click for options)**

13:50:15 - bayes.NaiveBayes
13:53:50 - bayes.NaiveBayes
14:0[illegible] - bayes.NaiveBayes

(right click)

- View in main window
- View in separate window
- Save result buffer
- Delete result buffer
- Load model
- Save model
- Re-evaluate model on current test set
- Visualize classifier errors
- Visualize tree
- Visualize margin curve
- Visualize threshold curve
- Cost/Benefit analysis
- Visualize cost curve

**Classifier output**

```
 1 3:comp.gra 1:misc.for    +  *1      0      0
 2 3:comp.gra 1:misc.for    +  *1      0      0
 3 2:rec.spor 2:rec.spor       0     *1      0
 4 2:rec.spor 3:comp.gra    +  0      0     *1
 5 1:misc.for 1:misc.for      *1      0      0
 6 1:misc.for 1:misc.for      *1      0      0
 1 3:comp.gra 1:misc.for    +  *0.921  0      0.079
 2 3:comp.gra 1:misc.for    +  *1      0      0
 3 2:rec.spor 2:rec.spor       0     *1      0
 4 2:rec.spor 2:rec.spor       0     *1      0
 5 1:misc.for 1:misc.for      *1      0      0
 6 1:misc.for 2:rec.spor    +  0     *1      0
 1 3:comp.gra 3:comp.gra       0      0     *1
 2 3:comp.gra 3:comp.gra       0      0     *1
 3 2:rec.spor 2:rec.spor       0     *1      0
 4 2:rec.spor 2:rec.spor       0     *1      0
 5 1:misc.for 3:comp.gra    +  0      0     *1
 6 1:misc.for 1:misc.for      *1      0      0
 1 3:comp.gra 1:misc.for    +  *0.999  0      0.001
 2 3:comp.gra 3:comp.gra       0      0     *1
 3 2:rec.spor 1:misc.for    +  *1      0      0
 4 2:rec.spor 2:rec.spor       0     *1      0
 5 1:misc.for 1:misc.for      *1      0      0
 6 1:misc.for 2:rec.spor    +  0     *1      0
 1 3:comp.gra 3:comp.gra       0      0     *1
 [illegible] 1:misc.for    +  *1      0      0
 [illegible] 2:rec.spor        0     *1      0
 [illegible] 2:rec.spor        0     *1      0
 [illegible] 2:rec.spor    +  0     *1      0
 [illegible] 3:comp.gra    +  0      0     *1
 [illegible] 3:comp.gra        0      0     *1
 [illegible] 3:comp.gra        0      0     *1
 [illegible] 2:rec.spor        0     *1      0
 [illegible] 2:rec.spor        0     *1      0
 [illegible] 1:misc.for       *1      0      0
 [illegible] 1:misc.for       *1      0      0

[illegible]ross-validation ===

[illegible]ied Instances          43               71.6667 %
[illegible]ified Instances        17               28.3333 %
[illegible]                        0.575
[illegible]or                      0.188
[illegible] error                  0.4326
[illegible] error                 42.3007 %
[illegible]ared error             91.7785 %
[illegible]nstances               60

=== Detailed Accuracy By Class ===

               TP Rate   FP Rate   Precision   Recall   F-Measure   ROC Area  Class
                 0.65      0.225      0.591     0.65      0.619      0.829    misc.forsale
                 0.8       0.1        0.8       0.8       0.8        0.904    rec.sport.hockey
                 0.7       0.1        0.778     0.7       0.737      0.904    comp.graphics
Weighted Avg.    0.717     0.142      0.723     0.717     0.719      0.879

=== Confusion Matrix ===

  a  b  c   <-- classified as
 13  4  3 |  a = misc.forsale
  3 16  1 |  b = rec.sport.hockey
  6  0 14 |  c = comp.graphics
```

**Status**

OK

Log — x 0

Weka Explorer
Preprocess
Classify
Cluster
Associate
Select attributes
Visualize
Classifier
Choose
NaiveBayes
Weka Classifier Visualize: 14:01:58 - bayes.NaiveBayes (example)
X: predictedclass (Nom)
Y: class (Nom)
Colour: class (Nom)
Select
Reset
Clear
Open
Save
Plot:example_predicted
Little squares designate errors
comp.graphics
rec.sport.hockey
misc.forsale
misc.forsale
rec.sport.hockey
comp.graphics
Class colour
misc.forsale rec.sport.hockey comp.graphics
Weka : Instance info
deadline: 0.0
dean: 0.0
decvax: 0.0
defenseman: 0.0
definitely: 0.0
deleted: 0.0
description: 0.0
development: 0.0
dialup-slip: 0.0
diego: 0.0
dipietro: 2.0
disappointment: 0.0
displaying: 0.0
dmorfx: 0.0
dmorph: 0.0
dollar: 0.0
donato: 2.0
done: 0.0
dot: 0.0
dots: 0.0
doubling: 0.0
drive: 0.0
driver: 0.0
dropped: 0.0
druce: 0.0
dsp: 0.0
dta: 0.0
predictedclass: rec.sport.hockey
class: rec.sport.hockey
Log
x 0

# Running on Test Set

Weka Explorer

Preprocess | Classify | Cluster | Associate | Select attributes | Visualize

**Classifier**

Choose **NaiveBayes**

**Test options**

- Use training set
- Supplied test set — Set...
- Cross-validation — Folds 10
- Percentage split — % 66

More options...

(Nom) newsgroup_class

Start | Stop

**Result list (right-click for options)**

13:50:15 - bayes.NaiveBayes
13:53:50 - bayes.NaiveBayes
14:01:58 - bayes.NaiveBayes

**Test Instances**

Relation: example
Instances: ? Attributes: 1001

Open file... | Open URL... | Close

**Open**

newsgroups

| Name | Date Modified |
|---|---|
| example_test.arff | Tuesday, March 4, 2014 3:28 PM |
| example_train.arff | Tuesday, March 4, 2014 3:28 PM |
| newsgroups_test | Thursday, November 1, 2007 5:27 PM |
| newsgroups_train | Thursday, November 1, 2007 5:27 PM |
| test | Monday, March 3, 2014 3:00 PM |
| weka.py | Tuesday, March 4, 2014 3:27 PM |

File Format: Arff data files (*.arff)

Cancel | Open

```
=== Stratified cross-valida
=== Summary ===

Correctly Classified Instan
Incorrectly Classified Inst
Kappa statistic
Mean absolute error
Root mean squared error
Relative absolute error
Root relative squared error
Total Number of Instances

=== Detailed Accuracy By Class ===

                 TP Rate   FP Rate   Precision   Recall   F-Measure   ROC Area  Class
                   0.65      0.225      0.591      0.65      0.619      0.829    misc.forsale
                   0.8       0.1        0.8        0.8       0.8        0.904    rec.sport.hockey
                   0.7       0.1        0.778      0.7       0.737      0.904    comp.graphics
Weighted Avg.      0.717     0.142      0.723      0.717     0.719      0.879

=== Confusion Matrix ===

   a  b  c   <-- classified as
  13  4  3 |  a = misc.forsale
   3 16  1 |  b = rec.sport.hockey
   6  0 14 |  c = comp.graphics
```

Status: OK

Log | x 0

Weka Explorer

Preprocess | Classify | Cluster | Associate | Select attributes | Visualize

**Classifier**

Choose **NaiveBayes**

**Test options**

- Use training set
- Supplied test set — Set...
- Cross-validation — Folds 10
- Percentage split — % 66

More options...

(Nom) class

Start | Stop

**Result list (right-click for options)**

13:50:15 - bayes.NaiveBayes
13:53:50 - bayes.NaiveBayes

**Classifier output**

```
  precision                    1                1                1

driver
  mean                         0              0.1                0
  std. dev.               0.3333           0.4359           0.3333
  weight sum                  20               20               20
  precision                    2                2                2

dropped
  mean                         0              0.1                0
  std. dev.               0.1667              0.3           0.1667
  weight sum                  20               20               20
  precision                    1                1                1

druce
  mean                         0              0.1                0
  std. dev.               0.3333           0.4359           0.3333
  weight sum                  20               20               20
  precision                    2                2                2

dsp
  mean                      0.05                0             0.05
  std. dev.               0.2179           0.1667           0.2179
  weight sum                  20               20               20
  precision                    1                1                1

dta
  mean                         0                0              0.1
  std. dev.               0.3333           0.3333           0.4359
  weight sum                  20               20               20
  precision                    2                2                2


Time taken to build model: 0.04 seconds

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

Correctly Classified Instances          43               71.6667 %
Incorrectly Classified Instances        17               28.3333 %
Kappa statistic                          0.575
Mean absolute error                      0.188
Root mean squared error                  0.4326
Relative absolute error                 42.3007 %
Root relative squared error             91.7785 %
Total Number of Instances               60

=== Detailed Accuracy By Class ===

               TP Rate   FP Rate   Precision   Recall  F-Measure   ROC Area  Class
                 0.65      0.225      0.591     0.65      0.619      0.829    misc.forsale
                 0.8       0.1        0.8       0.8       0.8        0.904    rec.sport.hockey
                 0.7       0.1        0.778     0.7       0.737      0.904    comp.graphics
Weighted Avg.    0.717     0.142      0.723     0.717     0.719      0.879

=== Confusion Matrix ===

  a  b  c   <-- classified as
 13  4  3 |  a = misc.forsale
  3 16  1 |  b = rec.sport.hockey
  6  0 14 |  c = comp.graphics
```

accuracy

**Status**

OK

Log x 0

# Explorer: Attribute Selection

- Find which attributes are the most predictive ones
- Two parts:
  - Search method:
    - » best-first, forward selection, random, exhaustive, genetic algorithm, ranking
  - Evaluation method:
    - » information gain, chi-squared, etc.
- Very flexible: WEKA allows (almost) arbitrary combinations of these two

Weka Explorer
Preprocess
Classify
Cluster
Associate
Select attributes
Visualize
Individual Features Ranking
Attribute Evaluator
Choose
ChiSquaredAttributeEval
Search Method
Choose
Attribute Sele
Use full t
Cross-va
(Nom) new
Start
Result list (rig
weka.gui.GenericObjectEditor
weka.attributeSelection.ChiSquaredAttributeEval
About
ChiSquaredAttributeEval :
More
Evaluates the worth of an attribute by computing the value of the chi-squared statistic with respect to the class.
Capabilities
binarizeNumericAttributes
False
missingMerge
True
Open...
Save...
OK
Cancel
Information
NAME
weka.attributeSelection.ChiSquaredAttributeEval
SYNOPSIS
ChiSquaredAttributeEval :
Evaluates the worth of an attribute by computing the value of the chi-squared statistic with respect to the class.
OPTIONS
binarizeNumericAttributes -- Just binarize numeric attributes instead of properly discretizing them.
missingMerge -- Distribute counts for missing values. Counts are distributed across other values in proportion to their frequency. Otherwise, missing is treated as a separate value.
Status
OK
Log
x 0

Weka Explorer

Preprocess | Classify | Cluster | Associate | Select attributes | Visualize

**Attribute Evaluator**

Choose: **ChiSquaredAttributeEval**

**Search Method**

Choose: **Ranker -T -1.7976931348623157E308 -N -1**

**Attribute Selection Mode**

- (•) Use full training set
- ( ) Cross-validation — Folds 10, Seed 1

(Nom) class

Start | Stop

**Result list (right-click for options)**

15:42:22 - BestFirst + CfsSubsetEval
15:51:23 - BestFirst + CfsSubsetEval
15:51:42 - Ranker + ChiSquaredAttributeEva

**Attribute selection output**

```
=== Run information ===

Evaluator:    weka.attributeSelection.ChiSquaredAttributeEval
Search:weka.attributeSelection.Ranker -T -1.7976931348623157E308 -N -1
Relation:     example
Instances:    60
Attributes:   1001
[list of attributes omitted]
Evaluation mode:evaluate on all training data


=== Attribute Selection on all input data ===

Search Method:
        Attribute ranking.

Attribute Evaluator (supervised, Class (nominal): 1001 class):
        Chi-squared Ranking Filter

Ranked attributes:
24          118 nhl
24           52 sale
17.52        11 ca
15.8491      81 toronto
15.8491      27 lemieux
14.902       63 offer
14.902       89 games
14.902       26 go
13.3333       7 play
13.3333      21 team
13.3333      61 game
13.3333      72 goals
13.3333      91 hockey
13.3333     173 win
13.3333      10 first
13.3333      28 selanne
13.3333     145 work
12.4038     159 interested
12.4038      54 asking
12.3529      23 re
10.9091     252 program
10.9091      39 winnipeg
10.9091      41 mario
10.9091      29 pittsburgh
10.9091      53 screen
10.9091     140 season
10.9091     324 might
10.9091      49 ottawa
10.9091     196 library
10.9091     216 teams
10.9091     156 give
10.9091     365 version
```

misc.forsale

rec.sport.hockey

comp.graphics

Status: OK

Log | x 0

Weka Explorer

Preprocess | Classify | Cluster | Associate | Select attributes | Visualize

**Attribute Evaluator**

Choose: ChiSquaredAttributeEval

**Search Method**

Choose: Ranker -T -1.7976931348623157E308 -N -1

**Attribute Selection Mode**

- Use full training set
- Cross-validation — Folds 10, Seed 1

(Nom) class

Start | Stop

random number seed

**Result list (right-click for options)**

15:42:22 - BestFirst + CfsSubsetEval
15:51:23 - BestFirst + CfsSubsetEval
15:51:42 - Ranker + ChiSquaredAttributeEva
15:55:05 - BestFirst + CfsSubsetEval
15:56:24 - Ranker + ChiSquaredAttributeEva
15:57:03 - BestFirst + CfsSubsetEval
15:58:07 - Ranker + ChiSquaredAttributeEva

**Attribute selection output**

```
=== Run information ===

Evaluator:    weka.attributeSelection.ChiSquaredAttributeEval
Search:weka.attributeSelection.Ranker -T -1.7976931348623157E308 -N -1
Relation:     example
Instances:    60
Attributes:   1001
[list of attributes omitted]
                                    cross-validation


=== Attribute selection 10 fold cross-validation (stratified), seed: 1 ===

average merit      average rank  attribute
21.638 +- 2.232         1.4 +- 0.49       52 sale
21.638 +- 2.232         1.6 +- 0.49      118 nhl
16.604 +- 1.875         4   +- 1.55       11 ca
14.297 +- 1.984         7.5 +- 5.94       81 toronto
14.297 +- 1.984         8.4 +- 7.14       27 lemieux
13.482 +- 1.337         8.5 +- 2.84       89 games
13.682 +- 1.915        10.3 +- 7.04       26 go
13.496 +- 1.74         11.3 +- 8.65       63 offer
12.064 +- 1.392        15.1 +- 8.34       10 first
12.012 +- 1.215        16.1 +- 6.53       61 game
12.012 +- 1.215        16.3 +- 7.38       21 team
12.022 +- 1.624        16.6 +- 8.53       72 goals
12.022 +- 1.624        16.6 +-13.19      145 work
12.022 +- 1.624        17.4 +- 8.15       91 hockey
12.012 +- 1.215        17.5 +- 7.83      173 win
12.022 +- 1.624        17.7 +- 8.19       28 selanne
12.022 +- 1.624        18.5 +- 8.64        7 play
11.242 +- 1.327        21.8 +-10.01      159 interested
11.256 +- 1.69         22.9 +-12.12       54 asking
 9.83  +- 1.19         29   +- 8.94      216 teams
 9.83  +- 1.19         29   +- 7.09       82 winning
 9.83  +- 1.19         29.2 +- 8.21      140 season
 9.83  +- 1.19         29.6 +- 6.93      324 might
 9.83  +- 1.19         30   +-10.26       39 winnipeg
 9.83  +- 1.19         30.8 +-10.49       41 mario
 9.83  +- 1.19         31   +- 9.42      209 record
 9.84  +- 1.576        32.6 +-22.02      252 program
 9.83  +- 1.19         33   +-19.39      365 version
 9.83  +- 1.19         33.6 +- 8.03      266 teemu
 7.725 +- 1.12         45.8 +- 7.43      410 deepak
 7.725 +- 1.12         47   +- 9.23      400 chhabra
 7.725 +- 1.12         47   +-10.29       74 ists
 7.725 +- 1.12         47.1 +- 8.24      409 dchhabra
 7.725 +- 1.12         50.6 +-11.45      381 another
 9.204 +- 3.248        53   +-80.19      104 winner
 7.725 +- 1.12         53   +-11.88      485 played
 7.725 +- 1.12         53.1 +- 4.78      462 leaders
 5.819 +- 3.868        58.8 +-24.81      344 say
 9.204 +- 3.248        60.7 +-95.18       29 pittsburgh
 9.204 +- 3.248        63.2 +-111.11      49 ottawa
```

misc.forsale

rec.sport.hockey

comp.graphics

Not in original list

Status: OK

Log

x 0

# Features Subsets Selection

Weka Explorer

Preprocess | Classify | Cluster | Associate | Select attributes | Visualize

Attribute Evaluator

Choose **CfsSubsetEval**

Search Method

Choose **BestFirst -D 1 -N 5**

Attribute Selection Mod

Use full training se

Cross-validation

(Nom) newsgroup_c

Start

Result list (right-click f

weka.gui.GenericObjectEditor

weka.attributeSelection.CfsSubsetEval

About

CfsSubsetEval :

Evaluates the worth of a subset of attributes by considering the individual predictive ability of each feature along with the degree of redundancy between them.

More

Capabilities

locallyPredictive True

missingSeparate False

Open... Save... OK Cancel

Information

NAME
weka.attributeSelection.CfsSubsetEval

SYNOPSIS
CfsSubsetEval :

Evaluates the worth of a subset of attributes by considering the individual predictive ability of each feature along with the degree of redundancy between them.

Subsets of features that are highly correlated with the class while having low intercorrelation are preferred.

For more information see:

M. A. Hall (1998). Correlation-based Feature Subset Selection for Machine Learning. Hamilton, New Zealand.

OPTIONS
locallyPredictive -- Identify locally predictive attributes. Iteratively adds attributes with the highest correlation with the class as long as there is not already an attribute in the subset that has a higher correlation with the attribute in question

missingSeparate -- Treat missing as a separate value. Otherwise, counts for missing values are distributed across other values in proportion to their frequency.

Status
OK

Log

Weka Explorer

Preprocess | Classify | Cluster | Associate | Select attributes | Visualize

**Attribute Evaluator**

Choose **CfsSubsetEval**

**Search Method**

Choose **GreedyStepwise** -T -1.7976931348623157E308 -N -1

**Attribute Selection Mode**

- (●) Use full training set
- ( ) Cross-validation  Folds 10  Seed 1

(Nom) class

Start  Stop

Result list (right-click for options)

- 15:42:22 - BestFirst + CfsSubsetEval
- 15:51:23 - BestFirst + CfsSubsetEval
- 15:51:42 - Ranker + ChiSquaredAttributeEva
- 15:55:05 - BestFirst + CfsSubsetEval
- 15:56:24 - Ranker + ChiSquaredAttributeEva
- 15:57:03 - BestFirst + CfsSubsetEval
- 15:58:07 - Ranker + ChiSquaredAttributeEva
- 16:01:20 - GreedyStepwise + CfsSubsetEval

**Attribute selection output**

```
=== Run information ===

Evaluator:    weka.attributeSelection.CfsSubsetEval
Search:weka.attributeSelection.GreedyStepwise -T -1.7976931348623157E308 -N -1
Relation:     example
Instances:    60
Attributes:   1001
[list of attributes omitted]
Evaluation mode:evaluate on all training data


=== Attribute Selection on all input data ===

Search Method:
        Greedy Stepwise (forwards).
        Start set: no attributes
        Merit of best subset found:    0.544

Attribute Subset Evaluator (supervised, Class (nominal): 1001 class):
        CFS Subset Evaluator
        Including locally predictive attributes

Selected attributes: 10,11,23,26,27,49,52,53,54,63,72,81,89,91,104,118,133,145,150,152,156,159,174,190,196,252,365,571,629 : 29
                     first
                     ca
                     re
                     go
                     lemieux
                     ottawa
                     sale
                     screen
                     asking
                     offer
                     goals
                     toronto
                     games
                     hockey
                     winner
                     nhl
                     image
                     work
                     available
                     cup
                     give
                     interested
                     world
                     forsale
                     library
                     program
                     version
                     brand
                     fact
```

29 attributes selected

misc.forsale

comp.graphics

rec.sport.hockey

Status: OK

Log  x 0

Weka Explorer

Preprocess | Classify | Cluster | Associate | Select attributes | Visualize

# Saving the Selected Features

**Attribute Evaluator**

Choose: **CfsSubsetEval**

**Search Method**

Choose: **GreedyStepwise** -T -1.7976931348623157E308 -N -1

**Attribute Selection Mode**

- Use full training set
- Cross-validation — Folds 10, Seed 1

(Nom) class

Start | Stop

**Result list (right-click for options)**

```
15:42:22 - BestFirst + CfsSubsetEval
15:51:23 - BestFirst + CfsSubsetEval
15:51:42 - Ranker + ChiSquaredAttributeEva
15:55:05 - BestFirst + CfsSubsetEval
15:56:24 - Ranker + ChiSquaredAttributeEva
15:57:03 - BestFirst + CfsSubsetEval
15:58:07 - Ranker + ChiSquaredAttributeEva
16:01:20 - G
```

- View in main window
- View in separate window
- Save result buffer
- Delete result buffer
- Visualize reduced data
- Save reduced data...

**Attribute selection output**

```
=== Run information ===

Evaluator:    weka.attributeSelection.CfsSubsetEval
Search:weka.attributeSelection.GreedyStepwise -T -1.7976931348623157E308 -N -1
Relation:     example
Instances:    60
Attributes:   1001
[list of attributes omitted]
Evaluation mode:evaluate on all training data


=== Attribute Selection on all input data ===

Search Method:
        Greedy Stepwise (forwards).
        Start set: no attributes
        Merit of best subset found:    0.544

Attribute Subset Evaluator (supervised, Class (nominal): 1001 class):
        CFS Subset Evaluator
        Including locally predictive attributes

Selected attributes: 10,11,23,26,27,49,52,53,54,63,72,81,89,91,104,118,133,145,150,152,156,159,174,190,196,252,365,571,629 : 29
                     first
                     ca
                     re
                     go
                     lemieux
                     ottawa
                     sale
                     screen
                     asking
                     offer
                     goals
                     toronto
                     games
                     hockey
                     winner
                     nhl
                     image
                     work
                     available
                     cup
                     give
                     interested
                     world
                     forsale
                     library
                     program
                     version
                     brand
                     fact
```

All we can do from this tab is to save the buffer in a text file. Not very useful...

But we can also perform feature selection during the pre-processing step...
(the following slides)

Status: OK

Log | x 0

Weka Explorer
Preprocess
Classify
Cluster
Associate
Select attributes
Visualize
Features Selection in Preprocessing
Open file...
Open URL...
Open DB...
Generate...
Undo
Filter
uteSelection.CfsSubsetEval " -S "weka.attributeSelection.BestFirst -D 1 -N 5"
Apply
weka
filters
AllFilter
MultiFilter
supervised
attribute
AddClassification
AttributeSelection
ClassOrder
Discretize
NominalToBinary
PLSFilter
instance
unsupervised
Filter...
Remove filter
Close
ttributes: 1001
Invert
Pattern
Selected attribute
Name: think
Missing: 0 (0%)
Distinct: 5
Type: Numeric
Unique: 2 (3%)
Statistic
Value
Minimum
0
Maximum
10
Mean
0.433
StdDev
1.489
Class: class (Nom)
Visualize All
18 d
19 pts
20 one
21 team
22 article-i
23 re
24 second
25 think
26 go
27 lemieux
28 selanne
29 pittsburgh
30 don't
31 writes
32 new
33 like
34 scoring
35 third
Remove
Status
OK
Log
x 0

Weka Explorer
Preprocess
Classify
Cluster
Associate
Select attributes
Visualize
Open file...
Open URL...
Open DB...
Generate...
Undo
Edit...
Save...
Filter
Choose
AttributeSelection -E "weka.attributeSelection.CfsSubsetEval -L" -S "weka.attributeSelection.BestFirst -D 1 -N 5"
Apply
Current relation
Relation: example
Instances: 60
Attributes: 1001
Selected attribute
Name: ca
Missing: 0 (0%)
Distinct: 4
Type: Numeric
Unique: 0 (0%)
Statistic
Value
Attributes
All
No.
weka.gui.GenericObjectEditor
weka.filters.supervised.attribute.AttributeSelection
About
A supervised attribute filter that can be used to select attributes.
More
Capabilities
evaluator
Choose
CfsSubsetEval -L
search
Choose
BestFirst -D 1 -N 5
Open...
Save...
OK
Cancel
Information
NAME
weka.filters.supervised.attribute.AttributeSelection
SYNOPSIS
A supervised attribute filter that can be used to select attributes. It is very flexible and allows various search and evaluation methods to be combined.
OPTIONS
evaluator -- Determines how attributes/attribute subsets are evaluated.
search -- Determines the search method.
weka.gui.GenericObjectEditor
weka.attributeSelection.CfsSubsetEval
About
CfsSubsetEval :
Evaluates the worth of a subset of attributes by considering the individual predictive ability of each feature along with the degree of redundancy between them.
More
Capabilities
locallyPredictive
False
missingSeparate
False
Open...
Save...
OK
Cancel
Class: class (Nom)
Visualize All
50
7
0
3
0
2
4
Remove
Status
OK
Log
x 0

Weka Explorer

Preprocess | Classify | Cluster | Associate | Select attributes | Visualize

Open file... | Open URL... | Open DB... | Generate... | Undo | Edit... | Save...

Filter

Choose | **AttributeSelection** -E "weka.attributeSelection.CfsSubsetEval -L" -S "weka.attributeSelection.BestFirst -D 1 -N 5" | Apply

Current relation

Relation: example
Instances: 60 Attributes: 1001

Selected attribute

Name: ca Type: Numeric
Missing: 0 (0%) Distinct: 4 Unique: 0 (0%)

| Statistic | Value |
|---|---|
| Minimum | 0 |

Attributes

All

| No. | Name |
|---|---|
| 1 | edu |
| 2 | subj |
| 3 | lines |
| 4 | orga |
| 5 | peri |
| 6 | pp |
| 7 | play |
| 8 | pow |
| 9 | com |
| 10 | first |
| 11 | ca |
| 12 | univ |
| 13 | woul |
| 14 | g |
| 15 | host |
| 16 | nntp-posting |
| 17 | scorer |
| 18 | d |
| 19 | pts |
| 20 | one |
| 21 | team |
| 22 | article-i |
| 23 | re |
| 24 | second |
| 25 | think |
| 26 | go |
| 27 | lemieux |
| 28 | selanne |
| 29 | pittsburgh |
| 30 | don't |
| 31 | writes |
| 32 | new |
| 33 | like |
| 34 | scoring |
| 35 | third |
| 36 | see |

Remove

weka.gui.GenericObjectEditor

weka.filters.supervised.attribute.AttributeSelection

About

A supervised attribute filter that can be used to select attributes.

More | Capabilities

evaluator | Choose | **CfsSubsetEval** -L

search | Choose | **BestFirst** -D 1 -N 5

Open... | Save... | OK | Cancel

Information

NAME
weka.filters.supervised.attribute.AttributeSelection

SYNOPSIS
A supervised attribute filter that can be used to select attributes. It is very flexible and allows various search and evaluation methods to be combined.

OPTIONS
evaluator -- Determines how attributes/attribute subsets are evaluated.

search -- Determines the search method.

weka.gui.GenericObjectEditor

weka.attributeSelection.BestFirst

About

BestFirst:

More

Searches the space of attribute subsets by greedy hillclimbing augmented with a backtracking facility.

direction | Forward

lookupCacheSize | 1

searchTermination | 5

startSet

Open... | Save... | OK | Cancel

Class (Nom) | Visualize All

Status
OK

Log | x 0

Weka Explorer

Preprocess | Classify | Cluster | Associate | Select attributes | Visualize

Open file... | Open URL... | Open DB... | Generate... | Undo | Edit... | Save...

Filter

Choose **AttributeSelection** -E "weka.attributeSelection.CfsSubsetEval " -S "weka.attributeSelection.BestFirst -D 1 -N 5" Apply

Current relation

Relation: example
Instances: 60
Attributes: 1001

Attributes

All | None | Invert | Pattern

| No. | Name |
|---|---|
| 1 | edu |
| 2 | subject |
| 3 | lines |
| 4 | organization |
| 5 | period |
| 6 | pp |
| 7 | play |
| 8 | power |
| 9 | com |
| 10 | first |
| 11 | ca |
| 12 | university |
| 13 | would |
| 14 | g |
| 15 | host |
| 16 | nntp-posting |
| 17 | scorer |
| 18 | d |
| 19 | pts |
| 20 | one |
| 21 | team |
| 22 | article-i |
| 23 | re |
| 24 | second |
| 25 | think |
| 26 | go |
| 27 | lemieux |
| 28 | selanne |
| 29 | pittsburgh |
| 30 | don't |
| 31 | writes |
| 32 | new |
| 33 | like |
| 34 | scoring |
| 35 | third |
| 36 | see |

1001 attributes:
1000 + 1 (for the class)

Remove

Selected attribute

Name: edu
Missing: 0 (0%)
Distinct: 7
Type: Numeric
Unique: 2 (3%)

| Statistic | Value |
|---|---|
| Minimum | 0 |
| Maximum | 6 |
| Mean | 1.533 |
| StdDev | 1.467 |

Class: class (Nom) | Visualize All

Status
OK

Log x 0

Weka Explorer

Preprocess | Classify | Cluster | Associate | Select attributes | Visualize

Open file... | Open URL... | Open DB... | Generate... | Undo | Edit... | Save...

Filter

Choose | **AttributeSelection** -E "weka.attributeSelection.CfsSubsetEval -L" -S "weka.attributeSelection.BestFirst -D 1 -N 5" | Apply

Current relation

Relation: example-weka.filters.supervised.attribute.AttributeSelection-Eweka.attributeSele...
Instances: 60 Attributes: 30

Attributes

All | None | Invert | Pattern

| No. | Name |
|---|---|
| 1 | first |
| 2 | ca |
| 3 | re |
| 4 | go |
| 5 | lemieux |
| 6 | ottawa |
| 7 | sale |
| 8 | screen |
| 9 | asking |
| 10 | offer |
| 11 | goals |
| 12 | toronto |
| 13 | games |
| 14 | hockey |
| 15 | winner |
| 16 | nhl |
| 17 | image |
| 18 | work |
| 19 | available |
| 20 | cup |
| 21 | give |
| 22 | interested |
| 23 | world |
| 24 | forsale |
| 25 | library |
| 26 | program |
| 27 | version |
| 28 | brand |
| 29 | fact |
| 30 | class |

Just 30 attributes remain:
29 + 1 (for the class)

Remove

Selected attribute

Name: first Type: Numeric
Missing: 0 (0%) Distinct: 6 Unique: 3 (5%)

| Statistic | Value |
|---|---|
| Minimum | 0 |
| Maximum | 11 |
| Mean | 0.617 |
| StdDev | 1.767 |

Class: class (Nom) | Visualize All

Status
OK

Log x 0

# Run Naïve Bayes With the 30 Features

Weka Explorer

Preprocess | Classify | Cluster | Associate | Select attributes | Visualize

**Classifier**

Choose: NaiveBayes

**Test options**

- Use training set
- Supplied test set — Set...
- Cross-validation — Folds 10
- Percentage split — % 66

More options...

(Nom) class

Start | Stop

**Result list (right-click for options)**

```
15:35:44 - bayes.NaiveBayes
15:36:29 - bayes.NaiveBayes
15:49:22 - bayes.NaiveBayes
15:56:10 - bayes.NaiveBayes
17:26:09 - bayes.NaiveBayes
```

**Classifier output**

```
  precision                    1                1                1

library
  mean                         0                0             0.35
  std. dev.               0.1667           0.1667           0.6538
  weight sum                  20               20               20
  precision                    1                1                1

program
  mean                         0                0              0.3
  std. dev.               0.1667           0.1667           0.5568
  weight sum                  20               20               20
  precision                    1                1                1

version
  mean                         0                0             0.25
  std. dev.               0.1667           0.1667            0.433
  weight sum                  20               20               20
  precision                    1                1                1

brand
  mean                      0.15                0                0
  std. dev.               0.3571           0.1667           0.1667
  weight sum                  20               20               20
  precision                    1                1                1

fact
  mean                         0             0.15                0
  std. dev.               0.1667           0.3571           0.1667
  weight sum                  20               20               20
  precision                    1                1                1


Time taken to build model: 0 seconds

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

Correctly Classified Instances          55               91.6667 %
Incorrectly Classified Instances         5                8.3333 %
Kappa statistic                          0.875
Mean absolute error                      0.0661
Root mean squared error                  0.2104
Relative absolute error                 14.8701 %
Root relative squared error             44.6383 %
Total Number of Instances               60

=== Detailed Accuracy By Class ===

                 TP Rate   FP Rate   Precision   Recall   F-Measure   ROC Area  Class
                   0.95      0.1       0.826      0.95      0.884      0.977    misc.forsale
                   0.9       0         1          0.9       0.947      0.991    rec.sport.hockey
                   0.9       0.025     0.947      0.9       0.923      0.982    comp.graphics
Weighted Avg.      0.917     0.042     0.924      0.917     0.918      0.983

=== Confusion Matrix ===

  a  b  c   <-- classified as
 19  0  1 |  a = misc.forsale
  2 18  0 |  b = rec.sport.hockey
  2  0 18 |  c = comp.graphics
```

higher accuracy

Status: OK

Log

x 0

Weka Explorer

Preprocess | Classify | Cluster | Associate | Select attributes | Visualize

**Attribute Evaluator**

Choose **InfoGainAttributeEval**

**Search Method**

Choose **Ranker** -T -1.7976931348623157E308 -N -1

**Attribute Selection Mode**

- Use full training set
- Cross-validation  Folds 10  Seed 1

(Nom) class

Start  Stop

**Result list (right-click for options)**

11:35:08 - Ranker + InfoGainAttributeEval

**Attribute selection output**

```
                work
                available
                cup
                give
                interested
                world
                forsale
                library
                program
                version
                brand
                fact
                class
Evaluation mode:evaluate on all training data


=== Attribute Selection on all input data ===

Search Method:
        Attribute ranking.

Attribute Evaluator (supervised, Class (nominal): 30 class):
        Information Gain Ranking Filter

Ranked attributes:
 0.3167    7 sale
 0.3167   16 nhl
 0.2236    2 ca
 0.2083    5 lemieux
 0.2083   12 toronto
 0.2075    1 first
 0.1907   13 games
 0.1907   10 offer
 0.1907    4 go
 0.1752   14 hockey
 0.1752   11 goals
 0.1752   18 work
 0.1597   22 interested
 0.1597    9 asking
 0.1592    3 re
 0.1434    8 screen
 0.1434    6 ottawa
 0.1434   15 winner
 0.1434   26 program
 0.1434   27 version
 0.1434   25 library
 0.1434   21 give
 0.1434   20 cup
 0.1305   23 world
 0.1305   19 available
 0.1127   17 image
 0.1127   24 forsale
 0.0831   28 brand
 0.0831   29 fact

Selected attributes: 7,16,2,5,12,1,13,10,4,14,11,18,22,9,3,8,6,15,26,27,25,21,20,23,19,17,24,28,29 : 29
```

**Status**

OK

Log  x 0

# Some Important Algorithms

- WEKA has weird naming for some algorithms
- Here are some translations:
  - Naïve Bayes: `weka.classifiers.bayes.NaiveBayes`
  - Perceptron: `weka.classifiers.functions.VotedPerceptron`
  - Decision tree: `weka.classifiers.trees.J48`
  - Support vector machines: `weka.classifiers.functions.SMO`
  - *k* nearest neighbor: `weka.classifiers.lazy.IBk`
- Some of these are more sophisticated versions of the classic algorithms
  - e.g. the classic Naïve Bayes seems to be missing
  - A good alternative is the Multinomial Naïve Bayes model

# Performing Experiments

- Experimenter makes it easy to compare the performance of different learning schemes
- Problems:
  - classification
  - regression
- Results: written into file or database
- Evaluation options:
  - cross-validation
  - learning curve
  - hold-out
- Can also iterate over different parameter settings
- Significance-testing built in!

# Iris data set

```
@RELATION iris

@ATTRIBUTE sepallength       REAL
@ATTRIBUTE sepalwidth        REAL
@ATTRIBUTE petallength       REAL
@ATTRIBUTE petalwidth        REAL
@ATTRIBUTE class    {Iris-setosa,Iris-versicolor,Iris-virginica}

@DATA
5.1,3.5,1.4,0.2,Iris-setosa
4.9,3.0,1.4,0.2,Iris-setosa
4.7,3.2,1.3,0.2,Iris-setosa
4.6,3.1,1.5,0.2,Iris-setosa
. . .
7.0,3.2,4.7,1.4,Iris-versicolor
6.4,3.2,4.5,1.5,Iris-versicolor
6.9,3.1,4.9,1.5,Iris-versicolor
5.5,2.3,4.0,1.3,Iris-versicolor
. . .
6.3,3.3,6.0,2.5,Iris-virginica
5.8,2.7,5.1,1.9,Iris-virginica
7.1,3.0,5.9,2.1,Iris-virginica
6.3,2.9,5.6,1.8,Iris-virginica
. . .
```

# Experiments Setup

Open http://www.cs.vassar.edu/~cs366/data/weka_files/iris.arff

Enter filename for results if desired

Choose CVS

Experiment Configuration Mode: Simple Advanced

Open... New

Results Destination

CSV file Filename: Browse...

Experiment Type

Cross-validation

Number of folds: 10

Classification Regression

Iteration Control

Number of repetitions: 10

Data sets first

Algorithms first

Datasets

Add new... Edit selected... Delete selected

Use relative paths

Up Down

Algorithms

Add new... Edit selected... Delete selected

Load options... Save options... Up Down

weka.gui.GenericObjectEditor

weka
classifiers
bayes
AODE
AODEsr
BayesianLogisticRegression
BayesNet
ComplementNaiveBayes
DMNBtext
HNB
NaiveBayes
NaiveBayesMultinomial
NaiveBayesMultinomialUpdateable
NaiveBayesSimple
NaiveBayesUpdateable
WAODE
functions
lazy
meta
mi
misc

Filter... Remove filter Close

More

Capabilities

Cancel

Notes

Weka Experiment Environment

Setup | Run | Analyse

Experiment Configuration Mode: Simple Advanced

Open... Save... New

Results Destination

CSV file Filename: /Users/ide/Dropbox/Vassar/Courses/366/ml/results/experiment.csv Browse...

CSV file: can be open in Excel

Experiment Type

Cross-validation

Number of folds: 10

Classification Regression

Iteration Control

Number of repetitions: 10

Data sets first

Algorithms first

Datasets

Add new... Edit selected... Delete selected

Use relative paths

/Users/ide/Dropbox/Vassar/Courses/366/data/weka_files/iris.arff

datasets

Up Down

Algorithms

Add new... Edit selected... Delete selected

J48 -C 0.25 -M 2

IBk -K 1 -W 0 -A "weka.core.neighboursearch.LinearNNSearch -A \"weka.core.EuclideanDistance -R

SMO -C 1.0 -L 0.0010 -P 1.0E-12 -N 0 -V -1 -W 1 -K "weka.classifiers.functions.supportVector.P

NaiveBayes

algorithms

Load options... Save options... Up Down

Notes

Weka Experiment Environment
Setup
Run
Analyse
Start
Stop
Log
17:58:15: Started
17:58:16: Finished
17:58:16: There were 0 errors
Status
Not running

Weka Experiment Environment

Setup | Run | Analyse

**Source**

Got 400results

File... | Database... | Experiment

**Configure test**

| | |
|---|---|
| Testing with | Paired T-Test... |
| Row | Select |
| Column | Select |
| Comparison field | Percent_correct |
| Significance | 0.05 |
| Sorting (asc.) by | <default> |
| Test base | Select |
| Displayed Columns | Select |
| Show std. deviations | ☐ |
| Output Format | Select |

Perform test | Save output

**Result list**

```
18:00:54 - Available resultsets
18:10:07 - Available resultsets
18:10:27 - Percent_correct - bayes.NaiveBayes '' !
18:11:38 - Available resultsets
18:11:44 - Percent_correct - bayes.NaiveBayes '' !
18:23:15 - Available resultsets
18:23:19 - Percent_correct - trees.J48 '-C 0.25 -I
18:24:00 - Available resultsets
18:24:04 - Percent_correct - trees.J48 '-C 0.25 -I
```

**Test output**

```
Available resultsets
(1) trees.J48 '-C 0.25 -M 2' -2.17733168393644448E17
(2) lazy.IBk '-K 1 -W 0 -A \"weka.core.neighboursearch.LinearNNSearch -A \\\"weka.core.EuclideanDistance -R first-last\\\
(3) functions.SMO '-C 1.0 -L 0.0010 -P 1.0E-12 -N 0 -V -1 -W 1 -K \"functions.supportVector.PolyKernel -C 250007 -E 1.0\"
(4) bayes.NaiveBayes '' 5.9952312017856973E18
```

Weka Experiment Environment

Setup | Run | Analyse

**Source**

Got 400results

File... | Database... | Experiment

**Configure test**

Testing with: Paired T-Test...

Row: Select

Column: Select

Comparison field: Percent_correct

Significance: 0.05

Sorting (asc.) by: <default>

Test base: Select

Displayed Columns: Select

Show std. deviations: ☐

Output Format: Select

Perform test | Save output

**Result list**

```
18:00:54 - Available resultsets
18:10:07 - Available resultsets
18:10:27 - Percent_correct - bayes.NaiveBayes '' !
18:11:38 - Available resultsets
18:11:44 - Percent_correct - bayes.NaiveBayes '' !
18:23:15 - Available resultsets
18:23:19 - Percent_correct - trees.J48 '-C 0.25 -I
18:24:00 - Available resultsets
18:24:04 - Percent_correct - trees.J48 '-C 0.25 -
```

**Test output**

```
Tester:     weka.experiment.PairedCorrectedTTester
Analysing:  Percent_correct
Datasets:   1
Resultsets: 4
Confidence: 0.05 (two tailed)
Sorted by:  -
Date:       2/21/16 6:24 PM

Dataset                   (1) trees.J4 | (2) lazy. (3) funct (4) bayes
------------------------------------------------------------------------
iris                    (100)    94.73 |   95.40    96.27    95.53
------------------------------------------------------------------------
                                 (v/ /*) |  (0/1/0)  (0/1/0)  (0/1/0)

Key:
(1) trees.J48 '-C 0.25 -M 2' -9.1773316839364448E17
(2) lazy.IBk '-K 1 -W 0 -A \"weka.core.neighboursearch.LinearNNSearch -A \\\"weka.core.EuclideanDistance -R first-last\\\
(3) functions.SMO '-C 1.0 -L 0.0010 -P 1.0E-12 -N 0 -V -1 -W  -K \"functions.supportVector.PolyKernel -C 250007 -E 1.0\"
(4) bayes.NaiveBayes '' 5.9952312017856973E18
```

accuracy

Decision tree is the worst

SVM is the best

Results are output into an CSV file, which can be read in Excel!

experiment.csv

| | Key_Dataset | Key_Run | Key_Fold | Key_Scheme | Key_Scheme | Key_Scheme | Date_time | Number_of_ | Number_of_ | Number_cor | Number_inc | Number_unc | Percent_corr | Percent_incc | Percent_unc | Kappa_statis | Mean_absol | Root_mean_ | Relative_abs | Root_relativ | SF_prior_ent | SF_scheme_ | SF_entropy_ | SF_mean_pr | SF_mean_scl |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 2 | iris | 1 | 1 | weka.classifi | '-C 0.25 -M 2 | -2.177E+17 | 2.02E+07 | 135 | 15 | 14 | 1 | 0 | 93.3333333 | 6.66666667 | 0 | 0.9 | 0.04501601 | 0.16931765 | 10.1286031 | 35.9176986 | 23.7744375 | 2.6327151 | 21.1417224 | 1.5849625 | 0.17551434 |
| 3 | iris | 1 | 2 | weka.classifi | '-C 0.25 -M 2 | -2.177E+17 | 2.02E+07 | 135 | 15 | 15 | 0 | 0 | 100 | 0 | 0 | 1 | 0.01058801 | 0.01588652 | 2.38230289 | 3.37003988 | 23.7744375 | 0.34785621 | 23.4265813 | 1.5849625 | 0.02319041 |
| 4 | iris | 1 | 3 | weka.classifi | '-C 0.25 -M 2 | -2.177E+17 | 2.02E+07 | 135 | 15 | 15 | 0 | 0 | 100 | 0 | 0 | 1 | 0.01058801 | 0.01588652 | 2.38230289 | 3.37003988 | 23.7744375 | 0.34785621 | 23.4265813 | 1.5849625 | 0.02319041 |
| 5 | iris | 1 | 4 | weka.classifi | '-C 0.25 -M 2 | -2.177E+17 | 2.02E+07 | 135 | 15 | 15 | 0 | 0 | 100 | 0 | 0 | 1 | 0.01058801 | 0.01588652 | 2.38230289 | 3.37003988 | 23.7744375 | 0.34785621 | 23.4265813 | 1.5849625 | 0.02319041 |
| 6 | iris | 1 | 5 | weka.classifi | '-C 0.25 -M 2 | -2.177E+17 | 2.02E+07 | 135 | 15 | 14 | 1 | 0 | 93.3333333 | 6.66666667 | 0 | 0.9 | 0.05865633 | 0.21277975 | 13.1976744 | 45.1374022 | 23.7744375 | 5.97709158 | 17.7973459 | 1.5849625 | 0.39847277 |
| 7 | iris | 1 | 6 | weka.classifi | '-C 0.25 -M 2 | -2.177E+17 | 2.02E+07 | 135 | 15 | 15 | 0 | 0 | 100 | 0 | 0 | 1 | 0.00940076 | 0.01486493 | 2.11517165 | 3.1533278 | 23.7744375 | 0.30879833 | 23.4656392 | 1.5849625 | 0.02058656 |
| 8 | iris | 1 | 7 | weka.classifi | '-C 0.25 -M 2 | -2.177E+17 | 2.02E+07 | 135 | 15 | 13 | 2 | 0 | 86.6666667 | 13.3333333 | 0 | 0.8 | 0.09312169 | 0.29831136 | 20.952381 | 63.2813965 | 23.7744375 | 2148.13906 | -2124.3646 | 1.5849625 | 143.209271 |
| 9 | iris | 1 | 8 | weka.classifi | '-C 0.25 -M 2 | -2.177E+17 | 2.02E+07 | 135 | 15 | 13 | 2 | 0 | 86.6666667 | 13.3333333 | 0 | 0.8 | 0.09185185 | 0.29499715 | 20.6666667 | 62.578346 | 23.7744375 | 1079.62154 | -1055.8471 | 1.5849625 | 71.9747693 |
| 10 | iris | 1 | 9 | weka.classifi | '-C 0.25 -M 2 | -2.177E+17 | 2.02E+07 | 135 | 15 | 15 | 0 | 0 | 100 | 0 | 0 | 1 | 0.00940076 | 0.01486493 | 2.11517165 | 3.1533278 | 23.7744375 | 0.30879833 | 23.4656392 | 1.5849625 | 0.02058656 |
| 11 | iris | 1 | 10 | weka.classifi | '-C 0.25 -M 2 | -2.177E+17 | 2.02E+07 | 135 | 15 | 15 | 0 | 0 | 100 | 0 | 0 | 1 | 0.01058801 | 0.01588652 | 2.38230289 | 3.37003988 | 23.7744375 | 0.34785621 | 23.4265813 | 1.5849625 | 0.02319041 |
| 12 | iris | 2 | 1 | weka.classifi | '-C 0.25 -M 2 | -2.177E+17 | 2.02E+07 | 135 | 15 | 14 | 1 | 0 | 93.3333333 | 6.66666667 | 0 | 0.9 | 0.05271765 | 0.20654967 | 11.8614719 | 43.8158008 | 23.7744375 | 5.76432761 | 18.0101099 | 1.5849625 | 0.38428851 |
| 13 | iris | 2 | 2 | weka.classifi | '-C 0.25 -M 2 | -2.177E+17 | 2.02E+07 | 135 | 15 | 13 | 2 | 0 | 86.6666667 | 13.3333333 | 0 | 0.8 | 0.09198966 | 0.2948591 | 20.6976744 | 62.5490613 | 23.7744375 | 1079.56205 | -1055.7876 | 1.5849625 | 71.9708036 |
| 14 | iris | 2 | 3 | weka.classifi | '-C 0.25 -M 2 | -2.177E+17 | 2.02E+07 | 135 | 15 | 14 | 1 | 0 | 93.3333333 | 6.66666667 | 0 | 0.9 | 0.06795491 | 0.21102321 | 15.2898551 | 44.7647823 | 23.7744375 | 4.91275076 | 18.8616867 | 1.5849625 | 0.32751672 |
| 15 | iris | 2 | 4 | weka.classifi | '-C 0.25 -M 2 | -2.177E+17 | 2.02E+07 | 135 | 15 | 15 | 0 | 0 | 100 | 0 | 0 | 1 | 0.01058801 | 0.01588652 | 2.38230289 | 3.37003988 | 23.7744375 | 0.34785621 | 23.4265813 | 1.5849625 | 0.02319041 |
| 16 | iris | 2 | 5 | weka.classifi | '-C 0.25 -M 2 | -2.177E+17 | 2.02E+07 | 135 | 15 | 15 | 0 | 0 | 100 | 0 | 0 | 1 | 0.01058801 | 0.01588652 | 2.38230289 | 3.37003988 | 23.7744375 | 0.34785621 | 23.4265813 | 1.5849625 | 0.02319041 |
| 17 | iris | 2 | 6 | weka.classifi | '-C 0.25 -M 2 | -2.177E+17 | 2.02E+07 | 135 | 15 | 13 | 2 | 0 | 86.6666667 | 13.3333333 | 0 | 0.8 | 0.09292929 | 0.29663711 | 20.9090909 | 62.9262326 | 23.7744375 | 2148.13267 | -2124.3582 | 1.5849625 | 143.208844 |
| 18 | iris | 2 | 7 | weka.classifi | '-C 0.25 -M 2 | -2.177E+17 | 2.02E+07 | 135 | 15 | 15 | 0 | 0 | 100 | 0 | 0 | 1 | 0.01058801 | 0.01588652 | 2.38230289 | 3.37003988 | 23.7744375 | 0.34785621 | 23.4265813 | 1.5849625 | 0.02319041 |
| 19 | iris | 2 | 8 | weka.classifi | '-C 0.25 -M 2 | -2.177E+17 | 2.02E+07 | 135 | 15 | 14 | 1 | 0 | 93.3333333 | 6.66666667 | 0 | 0.9 | 0.04867725 | 0.2110574 | 10.952381 | 44.7720353 | 23.7744375 | 1074.13906 | -1050.3646 | 1.5849625 | 71.6092708 |
| 20 | iris | 2 | 9 | weka.classifi | '-C 0.25 -M 2 | -2.177E+17 | 2.02E+07 | 135 | 15 | 13 | 2 | 0 | 86.6666667 | 13.3333333 | 0 | 0.8 | 0.1005003 | 0.28085766 | 22.612568 | 59.5789064 | 23.7744375 | 8.51688086 | 15.2575566 | 1.5849625 | 0.56779206 |
| 21 | iris | 2 | 10 | weka.classifi | '-C 0.25 -M 2 | -2.177E+17 | 2.02E+07 | 135 | 15 | 15 | 0 | 0 | 100 | 0 | 0 | 1 | 0.01058801 | 0.01588652 | 2.38230289 | 3.37003988 | 23.7744375 | 0.34785621 | 23.4265813 | 1.5849625 | 0.02319041 |
| 22 | iris | 3 | 1 | weka.classifi | '-C 0.25 -M 2 | -2.177E+17 | 2.02E+07 | 135 | 15 | 15 | 0 | 0 | 100 | 0 | 0 | 1 | 0.01058801 | 0.01588652 | 2.38230289 | 3.37003988 | 23.7744375 | 0.34785621 | 23.4265813 | 1.5849625 | 0.02319041 |
| 23 | iris | 3 | 2 | weka.classifi | '-C 0.25 -M 2 | -2.177E+17 | 2.02E+07 | 135 | 15 | 14 | 1 | 0 | 93.3333333 | 6.66666667 | 0 | 0.9 | 0.05759397 | 0.21192618 | 12.9586426 | 44.9563312 | 23.7744375 | 1074.43464 | -1050.6602 | 1.5849625 | 71.6289761 |
| 24 | iris | 3 | 3 | weka.classifi | '-C 0.25 -M 2 | -2.177E+17 | 2.02E+07 | 135 | 15 | 14 | 1 | 0 | 93.3333333 | 6.66666667 | 0 | 0.9 | 0.04867725 | 0.20610486 | 10.952381 | 43.7214435 | 23.7744375 | 5.56614451 | 18.208293 | 1.5849625 | 0.3710763 |
| 25 | iris | 3 | 4 | weka.classifi | '-C 0.25 -M 2 | -2.177E+17 | 2.02E+07 | 135 | 15 | 15 | 0 | 0 | 100 | 0 | 0 | 1 | 0.01058801 | 0.01588652 | 2.38230289 | 3.37003988 | 23.7744375 | 0.34785621 | 23.4265813 | 1.5849625 | 0.02319041 |
| 26 | iris | 3 | 5 | weka.classifi | '-C 0.25 -M 2 | -2.177E+17 | 2.02E+07 | 135 | 15 | 13 | 2 | 0 | 86.6666667 | 13.3333333 | 0 | 0.8 | 0.09198966 | 0.29826331 | 20.6976744 | 63.271202 | 23.7744375 | 2148.10184 | -2124.3274 | 1.5849625 | 143.206789 |
| 27 | iris | 3 | 6 | weka.classifi | '-C 0.25 -M 2 | -2.177E+17 | 2.02E+07 | 135 | 15 | 14 | 1 | 0 | 93.3333333 | 6.66666667 | 0 | 0.9 | 0.05503246 | 0.209015 | 12.3823029 | 44.3387777 | 23.7744375 | 1074.34786 | -1050.5734 | 1.5849625 | 71.6231904 |
| 28 | iris | 3 | 7 | weka.classifi | '-C 0.25 -M 2 | -2.177E+17 | 2.02E+07 | 135 | 15 | 13 | 2 | 0 | 86.6666667 | 13.3333333 | 0 | 0.8 | 0.09292929 | 0.29829636 | 20.9090909 | 63.2782129 | 23.7744375 | 2148.13267 | -2124.3582 | 1.5849625 | 143.208844 |
| 29 | iris | 3 | 8 | weka.classifi | '-C 0.25 -M 2 | -2.177E+17 | 2.02E+07 | 135 | 15 | 13 | 2 | 0 | 86.6666667 | 13.3333333 | 0 | 0.8 | 0.1031746 | 0.27624293 | 23.2142857 | 58.5999737 | 23.7744375 | 7.90187567 | 15.8725618 | 1.5849625 | 0.52679171 |
| 30 | iris | 3 | 9 | weka.classifi | '-C 0.25 -M 2 | -2.177E+17 | 2.02E+07 | 135 | 15 | 15 | 0 | 0 | 100 | 0 | 0 | 1 | 0.00940076 | 0.01486493 | 2.11517165 | 3.1533278 | 23.7744375 | 0.30879833 | 23.4656392 | 1.5849625 | 0.02058656 |
| 31 | iris | 3 | 10 | weka.classifi | '-C 0.25 -M 2 | -2.177E+17 | 2.02E+07 | 135 | 15 | 15 | 0 | 0 | 100 | 0 | 0 | 1 | 0.00940076 | 0.01486493 | 2.11517165 | 3.1533278 | 23.7744375 | 0.30879833 | 23.4656392 | 1.5849625 | 0.02058656 |
| 32 | iris | 4 | 1 | weka.classifi | '-C 0.25 -M 2 | -2.177E+17 | 2.02E+07 | 135 | 15 | 13 | 2 | 0 | 86.6666667 | 13.3333333 | 0 | 0.8 | 0.09198966 | 0.2948591 | 20.6976744 | 62.5490613 | 23.7744375 | 1079.56205 | -1055.7876 | 1.5849625 | 71.9708036 |
| 33 | iris | 4 | 2 | weka.classifi | '-C 0.25 -M 2 | -2.177E+17 | 2.02E+07 | 135 | 15 | 15 | 0 | 0 | 100 | 0 | 0 | 1 | 0.01058801 | 0.01588652 | 2.38230289 | 3.37003988 | 23.7744375 | 0.34785621 | 23.4265813 | 1.5849625 | 0.02319041 |
| 34 | iris | 4 | 3 | weka.classifi | '-C 0.25 -M 2 | -2.177E+17 | 2.02E+07 | 135 | 15 | 13 | 2 | 0 | 86.6666667 | 13.3333333 | 0 | 0.8 | 0.10666667 | 0.3032517 | 24 | 64.3293992 | 23.7744375 | 1080.2065 | -1056.4321 | 1.5849625 | 72.0137668 |
| 35 | iris | 4 | 4 | weka.classifi | '-C 0.25 -M 2 | -2.177E+17 | 2.02E+07 | 135 | 15 | 13 | 2 | 0 | 86.6666667 | 13.3333333 | 0 | 0.8 | 0.11239936 | 0.29828717 | 25.2898551 | 63.2762652 | 23.7744375 | 1078.91275 | -1055.1383 | 1.5849625 | 71.9275167 |
| 36 | iris | 4 | 5 | weka.classifi | '-C 0.25 -M 2 | -2.177E+17 | 2.02E+07 | 135 | 15 | 15 | 0 | 0 | 100 | 0 | 0 | 1 | 0.01058801 | 0.01588652 | 2.38230289 | 3.37003988 | 23.7744375 | 0.34785621 | 23.4265813 | 1.5849625 | 0.02319041 |
| 37 | iris | 4 | 6 | weka.classifi | '-C 0.25 -M 2 | -2.177E+17 | 2.02E+07 | 135 | 15 | 15 | 0 | 0 | 100 | 0 | 0 | 1 | 0.00940076 | 0.01486493 | 2.11517165 | 3.1533278 | 23.7744375 | 0.30879833 | 23.4656392 | 1.5849625 | 0.02058656 |
| 38 | iris | 4 | 7 | weka.classifi | '-C 0.25 -M 2 | -2.177E+17 | 2.02E+07 | 135 | 15 | 15 | 0 | 0 | 100 | 0 | 0 | 1 | 0.01058801 | 0.01588652 | 2.38230289 | 3.37003988 | 23.7744375 | 0.34785621 | 23.4265813 | 1.5849625 | 0.02319041 |
| 39 | iris | 4 | 8 | weka.classifi | '-C 0.25 -M 2 | -2.177E+17 | 2.02E+07 | 135 | 15 | 15 | 0 | 0 | 100 | 0 | 0 | 1 | 0.01058801 | 0.01588652 | 2.38230289 | 3.37003988 | 23.7744375 | 0.34785621 | 23.4265813 | 1.5849625 | 0.02319041 |
| 40 | iris | 4 | 9 | weka.classifi | '-C 0.25 -M 2 | -2.177E+17 | 2.02E+07 | 135 | 15 | 14 | 1 | 0 | 93.3333333 | 6.66666667 | 0 | 0.9 | 0.04867725 | 0.2110574 | 10.952381 | 44.7720353 | 23.7744375 | 1074.13906 | -1050.3646 | 1.5849625 | 71.6092708 |
| 41 | iris | 4 | 10 | weka.classifi | '-C 0.25 -M 2 | -2.177E+17 | 2.02E+07 | 135 | 15 | 15 | 0 | 0 | 100 | 0 | 0 | 1 | 0.01058801 | 0.01588652 | 2.38230289 | 3.37003988 | 23.7744375 | 0.34785621 | 23.4265813 | 1.5849625 | 0.02319041 |
| 42 | iris | 5 | 1 | weka.classifi | '-C 0.25 -M 2 | -2.177E+17 | 2.02E+07 | 135 | 15 | 15 | 0 | 0 | 100 | 0 | 0 | 1 | 0.01058801 | 0.01588652 | 2.38230289 | 3.37003988 | 23.7744375 | 0.34785621 | 23.4265813 | 1.5849625 | 0.02319041 |
| 43 | iris | 5 | 2 | weka.classifi | '-C 0.25 -M 2 | -2.177E+17 | 2.02E+07 | 135 | 15 | 13 | 2 | 0 | 86.6666667 | 13.3333333 | 0 | 0.8 | 0.10572391 | 0.30314813 | 23.7878788 | 64.3074285 | 23.7744375 | 1080.14389 | -1056.3695 | 1.5849625 | 72.009593 |
| 44 | iris | 5 | 3 | weka.classifi | '-C 0.25 -M 2 | -2.177E+17 | 2.02E+07 | 135 | 15 | 14 | 1 | 0 | 93.3333333 | 6.66666667 | 0 | 0.9 | 0.05384521 | 0.21134193 | 12.1151717 | 44.8323932 | 23.7744375 | 1074.3088 | -1050.5344 | 1.5849625 | 71.6205866 |
| 45 | iris | 5 | 4 | weka.classifi | '-C 0.25 -M 2 | -2.177E+17 | 2.02E+07 | 135 | 15 | 15 | 0 | 0 | 100 | 0 | 0 | 1 | 0.01058801 | 0.01588652 | 2.38230289 | 3.37003988 | 23.7744375 | 0.34785621 | 23.4265813 | 1.5849625 | 0.02319041 |
| 46 | iris | 5 | 5 | weka.classifi | '-C 0.25 -M 2 | -2.177E+17 | 2.02E+07 | 135 | 15 | 14 | 1 | 0 | 93.3333333 | 6.66666667 | 0 | 0.9 | 0.04501601 | 0.16931765 | 10.1286031 | 35.9176986 | 23.7744375 | 2.6327151 | 21.1417224 | 1.5849625 | 0.17551434 |
| 47 | iris | 5 | 6 | weka.classifi | '-C 0.25 -M 2 | -2.177E+17 | 2.02E+07 | 135 | 15 | 14 | 1 | 0 | 93.3333333 | 6.66666667 | 0 | 0.9 | 0.05503246 | 0.209015 | 12.3823029 | 44.3387777 | 23.7744375 | 1074.34786 | -1050.5734 | 1.5849625 | 71.6231904 |
| 48 | iris | 5 | 7 | weka.classifi | '-C 0.25 -M 2 | -2.177E+17 | 2.02E+07 | 135 | 15 | 15 | 0 | 0 | 100 | 0 | 0 | 1 | 0.01058801 | 0.01588652 | 2.38230289 | 3.37003988 | 23.7744375 | 0.34785621 | 23.4265813 | 1.5849625 | 0.02319041 |
| 49 | iris | 5 | 8 | weka.classifi | '-C 0.25 -M 2 | -2.177E+17 | 2.02E+07 | 135 | 15 | 14 | 1 | 0 | 93.3333333 | 6.66666667 | 0 | 0.9 | 0.04848485 | 0.21103619 | 10.9090909 | 44.7675355 | 23.7744375 | 1074.13267 | -1050.3582 | 1.5849625 | 71.6088445 |
| 50 | iris | 5 | 9 | weka.classifi | '-C 0.25 -M 2 | -2.177E+17 | 2.02E+07 | 135 | 15 | 14 | 1 | 0 | 93.3333333 | 6.66666667 | 0 | 0.9 | 0.04867725 | 0.2110574 | 10.952381 | 44.7720353 | 23.7744375 | 1074.13906 | -1050.3646 | 1.5849625 | 71.6092708 |
| 51 | iris | 5 | 10 | weka.classifi | '-C 0.25 -M 2 | -2.177E+17 | 2.02E+07 | 135 | 15 | 15 | 0 | 0 | 100 | 0 | 0 | 1 | 0.00940076 | 0.01486493 | 2.11517165 | 3.1533278 | 23.7744375 | 0.30879833 | 23.4656392 | 1.5849625 | 0.02058656 |
| 52 | iris | 6 | 1 | weka.classifi | '-C 0.25 -M 2 | -2.177E+17 | 2.02E+07 | 135 | 15 | 14 | 1 | 0 | 93.3333333 | 6.66666667 | 0 | 0.9 | 0.05384521 | 0.21134193 | 12.1151717 | 44.8323932 | 23.7744375 | 1074.3088 | -1050.5344 | 1.5849625 | 71.6205866 |
| 53 | iris | 6 | 2 | weka.classifi | '-C 0.25 -M 2 | -2.177E+17 | 2.02E+07 | 135 | 15 | 14 | 1 | 0 | 93.3333333 | 6.66666667 | 0 | 0.9 | 0.05503246 | 0.209015 | 12.3823029 | 44.3387777 | 23.7744375 | 1074.34786 | -1050.5734 | 1.5849625 | 71.6231904 |
| 54 | iris | 6 | 3 | weka.classifi | '-C 0.25 -M 2 | -2.177E+17 | 2.02E+07 | 135 | 15 | 14 | 1 | 0 | 93.3333333 | 6.66666667 | 0 | 0.9 | 0.04867725 | 0.2110574 | 10.952381 | 44.7720353 | 23.7744375 | 1074.13906 | -1050.3646 | 1.5849625 | 71.6092708 |
| 55 | iris | 6 | 4 | weka.classifi | '-C 0.25 -M 2 | -2.177E+17 | 2.02E+07 | 135 | 15 | 13 | 2 | 0 | 86.6666667 | 13.3333333 | 0 | 0.8 | 0.08888889 | 0.2981424 | 20 | 63.2455532 | 23.7744375 | 2148 | -2124.2256 | 1.5849625 | 143.2 |
| 56 | iris | 6 | 5 | weka.classifi | '-C 0.25 -M 2 | -2.177E+17 | 2.02E+07 | 135 | 15 | 15 | 0 | 0 | 100 | 0 | 0 | 1 | 0.01058801 | 0.01588652 | 2.38230289 | 3.37003988 | 23.7744375 | 0.34785621 | 23.4265813 | 1.5849625 | 0.02319041 |
| 57 | iris | 6 | 6 | weka.classifi | '-C 0.25 -M 2 | -2.177E+17 | 2.02E+07 | 135 | 15 | 14 | 1 | 0 | 93.3333333 | 6.66666667 | 0 | 0.9 | 0.05271765 | 0.16724349 | 11.8614719 | 35.4777026 | 23.7744375 | 2.68676663 | 21.0876709 | 1.5849625 | 0.17911778 |
| 58 | iris | 6 | 7 | weka.classifi | '-C 0.25 -M 2 | -2.177E+17 | 2.02E+07 | 135 | 15 | 15 | 0 | 0 | 100 | 0 | 0 | 1 | 0.00940076 | 0.01486493 | 2.11517165 | 3.1533278 | 23.7744375 | 0.30879833 | 23.4656392 | 1.5849625 | 0.02058656 |
| 59 | iris | 6 | 8 | weka.classifi | '-C 0.25 -M 2 | -2.177E+17 | 2.02E+07 | 135 | 15 | 14 | 1 | 0 | 93.3333333 | 6.66666667 | 0 | 0.9 | 0.05271765 | 0.20654967 | 11.8614719 | 43.8158008 | 23.7744375 | 5.76432761 | 18.0101099 | 1.5849625 | 0.38428851 |
| 60 | iris | 6 | 9 | weka.classifi | '-C 0.25 -M 2 | -2.177E+17 | 2.02E+07 | 135 | 15 | 15 | 0 | 0 | 100 | 0 | 0 | 1 | 0.01058801 | 0.01588652 | 2.38230289 | 3.37003988 | 23.7744375 | 0.34785621 | 23.4265813 | 1.5849625 | 0.02319041 |

Weka Experiment Environment

Setup | Run | Analyse

Source

Got 400results

File... | Database... | Experiment

Configure test

Testing with: Paired T-Test...

Row: Select

Column: Select

Comparison field: Percent_correct

Significance: 0.05

Sorting (asc.) by: <default>

Test base: Select

Displayed Columns: Select

Show std. deviations

Output Format: Select

Perform test | Save output

Result list

```
18:00:54 - Available resultsets
18:10:07 - Available resultsets
18:10:27 - Percent_correct - bayes.NaiveBayes '' !
18:11:38 - Available resultsets
18:11:44 - Percent_correct - bayes.NaiveBayes '' !
18:23:15 - Available resultsets
18:23:19 - Percent_correct - trees.J48 '-C 0.25 -l
18:24:00 - Available resultsets
18:24:04 - Percent_correct - trees.J48 '-C 0.25 -l
18:46:24 - Percent_correct - trees.J48 '-C 0.25 -l
18:46:27 - Percent_correct - trees.J48 '-C 0.25 -l
18:46:54 - Available resultsets
18:46:57 - Percent_correct - trees.J48 '-C 0.25 -l
18:47:39 - Available resultsets
18:47:42 - Percent_correct - trees.J48 '-C 0.25 -l
```

Test output

```
Tester:     weka.experiment.PairedCorrectedTTester
Analysing:  Percent_correct
Datasets:   1
Resultsets: 4
Confidence: 0.05 (two tailed)
Sorted by:  -
Date:       2/21/16 6:47 PM


Dataset                   (1) trees.J4 | (2) lazy. (3) funct (4) bayes
-----------------------------------------------------------------------
example                  (100)   62.67 |   46.00 *   73.17    74.17
-----------------------------------------------------------------------
                               (v/ /*) |  (0/0/1)   (0/1/0)  (0/1/0)


Key:
(1) trees.J48 '-C 0.25 -M 2' -2.1773316839364448E17
(2) lazy.IBk '-K 1 -W 0 -A \"weka.core.neighboursearch.LinearNNSearch -A \\\"weka.core.EuclideanDistance -R first-last\\\
(3) functions.SMO '-C 1.0 -L 0.0010 -P 1.0E-12 -N 0 -V -1 -W 1 -K \"functions.supportVector.PolyKernel -C 250007 -E 1.0\"
(4) bayes.NaiveBayes '' 5.9952312017856973E18
```

K-means clustering loses big

Bayes wins!

Results on the example dataset

# DECISION TREES

# Sample Input Data

| age | income | student | credit_rating | buys_computer |
|---|---|---|---|---|
| <=30 | high | no | fair | no |
| <=30 | high | no | excellent | no |
| 31…40 | high | no | fair | yes |
| >40 | medium | no | fair | yes |
| >40 | low | yes | fair | yes |
| >40 | low | yes | excellent | no |
| 31…40 | low | yes | excellent | yes |
| <=30 | medium | no | fair | no |
| <=30 | low | yes | fair | yes |
| >40 | medium | yes | fair | yes |
| <=30 | medium | yes | excellent | yes |
| 31…40 | medium | no | excellent | yes |
| 31…40 | high | yes | fair | yes |
| >40 | medium | no | excellent | no |

# Output: A Decision Tree for “buys_computer”

# Algorithm for Decision Tree Induction

- Basic algorithm (a greedy algorithm)
  - Tree is constructed in a top-down recursive divide-and-conquer manner
  - At start, all the training examples are at the root
  - Attributes are categorical (if continuous-valued, they are discretized in advance)
  - Examples are partitioned recursively based on selected attributes
  - Test attributes are selected on the basis of a heuristic or statistical measure (e.g., information gain)

Load http://www.cs.vassar.edu/~cs366/data/weka_files/iris.arff

Weka Explorer

Open File... | Open URL... | Open DB... | Generate... | Undo | Edit... | Save...

**Filter**

Choose | None | Apply

**Current relation**

Relation: iris
Instances: 150 Attributes: 5

**Attributes**

All | None | Invert | Pattern

| No. | | Name |
|---|---|---|
| 1 | ☑ | sepallength |
| 2 | ☐ | sepalwidth |
| 3 | ☐ | petallength |
| 4 | ☐ | petalwidth |
| 5 | ☐ | class |

Remove

**Selected attribute**

Name: sepallength Type: Numeric
Missing: 0 (0%) Distinct: 35 Unique: 9 (6%)

| Statistic | Value |
|---|---|
| Minimum | 4.3 |
| Maximum | 7.9 |
| Mean | 5.843 |
| StdDev | 0.828 |

Class: class (Nom) | Visualize All

**Status**

OK

Log | x 0

Weka Explorer
Preprocess
Classify
Cluster
Associate
Select attributes
Visualize
Classifier
weka
classifiers
bayes
functions
lazy
meta
mi
misc
rules
trees
ADTree
BFTree
DecisionStump
FT
Id3
J48
J48graft
LADTree
LMT
M5P
NBTree
Filter...
Remove filter
Close
Decision tree algorithm
Status
OK
Log
x 0

Weka Explorer
Preprocess
Classify
Cluster
Associate
Select attributes
Visualize
Classifier
Choose
J48 -C 0.25 -M 2
Test options
Use training set
Supplied test set
Set...
Cross-validation
Folds
10
Percentage split
%
66
More options...
(Nom) class
Start
Stop
Result list (right-click for options)
Classifier output
Train on 66%, test on 33%
Status
OK
Log
x 0

Weka Explorer

Preprocess | Classify | Cluster | Associate | Select attributes | Visualize

**Classifier**

Choose: J48 -C 0.25 -M 2

**Test options**

- Use training set
- Supplied test set — Set...
- Cross-validation — Folds 10
- Percentage split — % 66

More options...

(Nom) class

Start | Stop

**Result list (right-click for options)**

09:45:13 - trees.J48

**Classifier output**

```
=== Run information ===

Scheme:weka.classifiers.trees.J48 -C 0.25 -M 2
Relation:     iris
Instances:    150
Attributes:   5
              sepallength
              sepalwidth
              petallength
              petalwidth
              class
Test mode:split 66.0% train, remainder test

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

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

petalwidth <= 0.6: Iris-setosa (50.0)
petalwidth > 0.6
|   petalwidth <= 1.7
|   |   petallength <= 4.9: Iris-versicolor (48.0/1.0)
|   |   petallength > 4.9
|   |   |   petalwidth <= 1.5: Iris-virginica (3.0)
|   |   |   petalwidth > 1.5: Iris-versicolor (3.0/1.0)
|   petalwidth > 1.7: Iris-virginica (46.0/1.0)

Number of Leaves  :     5

Size of the tree :      9


Time taken to build model: 0.01 seconds

=== Evaluation on test split ===
=== Summary ===

Correctly Classified Instances          49               96.0784 %
Incorrectly Classified Instances         2                3.9216 %
Kappa statistic                          0.9408
Mean absolute error                      0.0396
Root mean squared error                  0.1579
Relative absolute error                  8.8979 %
Root relative squared error             33.4091 %
Total Number of Instances               51

=== Detailed Accuracy By Class ===

                 TP Rate   FP Rate   Precision   Recall  F-Measure   ROC Area  Class
                   1         0          1         1         1          1        Iris-setosa
                   1         0.063      0.905     1         0.95       0.969    Iris-versicolor
                   0.882     0          1         0.882     0.938      0.967    Iris-virginica
Weighted Avg.      0.961     0.023      0.965     0.961     0.961      0.977

=== Confusion Matrix ===

  a  b  c   <-- classified as
 15  0  0 |  a = Iris-setosa
  0 19  0 |  b = Iris-versicolor
  0  2 15 |  c = Iris-virginica
```

**Status**

OK

Log | x 0

Weka Explorer

Preprocess | Classify | Cluster | Associate | Select attributes | Visualize

**Classifier**

Choose: J48 -C 0.25 -M 2

**Test options**

- Use training set
- Supplied test set — Set...
- Cross-validation — Folds 10
- Percentage split — % 66

More options...

(Nom) class

Start | Stop

**Result list (right-click for options)**

09:45:12 - trees.J48

- View in main window
- View in separate window
- Save result buffer
- Delete result buffer
- Load model
- Save model
- Re-evaluate model on current test set
- Visualize classifier errors
- Visualize tree
- Visualize margin curve
- Visualize threshold curve ▶
- Cost/Benefit analysis ▶
- Visualize cost curve ▶

**Classifier output**

```
=== Run information ===

Scheme:weka.classifiers.trees.J48 -C 0.25 -M 2
Relation:     iris
Instances:    150
Attributes:   5
              sepallength
              sepalwidth
              petallength
              petalwidth
              class
Test mode:split 66.0% train, remainder test

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

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

petalwidth <= 0.6: Iris-setosa (50.0)
petalwidth > 0.6
|   petalwidth <= 1.7
|   |   petallength <= 4.9: Iris-versicolor (48.0/1.0)
|   |   petallength > 4.9
|   |   |   petalwidth <= 1.5: Iris-virginica (3.0)
|   |   |   petalwidth > 1.5: Iris-versicolor (3.0/1.0)
|   petalwidth > 1.7: Iris-virginica (46.0/1.0)

Number of Leaves  :     5

Size of the tree :      9

Time taken to build model: 0.01 seconds

=== Evaluation on test split ===
=== Summary ===

Correctly Classified Instances          49               96.0784 %
Incorrectly Classified Instances         2                3.9216 %
Kappa statistic                          0.9408
Mean absolute error                      0.0396
Root mean squared error                  0.1579
Relative absolute error                  8.8979 %
Root relative squared error             33.4091 %
Total Number of Instances               51

=== Detailed Accuracy By Class ===

               TP Rate   FP Rate   Precision   Recall  F-Measure   ROC Area  Class
                 1         0          1         1         1          1        Iris-setosa
                 1         0.063      0.905     1         0.95       0.969    Iris-versicolor
                 0.882     0          1         0.882     0.938      0.967    Iris-virginica
Weighted Avg.    0.961     0.023      0.965     0.961     0.961      0.977

=== Confusion Matrix ===

  a  b  c   <-- classified as
 15  0  0 |  a = Iris-setosa
  0 19  0 |  b = Iris-versicolor
  0  2 15 |  c = Iris-virginica
```

Status: OK

Log | x 0

Weka Classifier Tree Visualizer: 09:45:13 - trees.J48 (iris)
Tree View
petalwidth
'<= 0.6'
'> 0.6'
'Iris-setosa (50.0)'
petalwidth
'<= 1.7'
'> 1.7'
petallength
'Iris-virginica (46.0/1.0)'
'<= 4.9'
'> 4.9'
'Iris-versicolor (48.0/1.0)'
petalwidth
'<= 1.5'
'> 1.5'
'Iris-virginica (3.0)'
'Iris-versicolor (3.0/1.0)'

# CLUSTERING IN WEKA

# BMW Browsers Data

```
@relation car-browsers
@attribute CustomerID numeric
@attribute Dealership numeric
@attribute Showroom numeric
@attribute ComputerSearch numeric
@attribute M5 numeric
@attribute 3Series numeric
@attribute Z4 numeric
@attribute Financing numeric
@attribute Purchase numeric

@data
1,1,0,0,0,0,0,0,0
2,1,1,1,0,0,0,1,0
3,1,0,0,0,0,0,0,0
. . .
```

Data on

- how people walk through the dealership and showroom
- what cars they look at
- how often they ultimately make purchases

Mine this data to find patterns, using clusters to determine if certain behaviors emerge

- 100 rows (samples)
- Each column describes the steps that the customers reached in their BMW experience
  - 1 = made it to this step or looked at this car
  - 0 = they didn't reach this step

Weka Explorer

Preprocess | Classify | Cluster | Associate | Select attributes | Visualize

Open file... | Save...

http://www.cs.vassar.edu/~cs366/data/weka_files/bmw-browsers.arff

Filter

Choose | None | Apply

Current relation

Relation: car-browsers
Instances: 100 Attributes: 9

Attributes

All | None | Invert | Pattern

| No. | Name |
|---|---|
| 1 | CustomerID |
| 2 | Dealership |
| 3 | Showroom |
| 4 | ComputerSearch |
| 5 | M5 |
| 6 | 3Series |
| 7 | Z4 |
| 8 | Financing |
| 9 | Purchase |

Delete Customer ID

Remove

Selected attribute

Name: CustomerID Type: Numeric
Missing: 0 (0%) Distinct: 100 Unique: 100 (100%)

| Statistic | Value |
|---|---|
| Minimum | 1 |
| Maximum | 100 |
| Mean | 50.5 |
| StdDev | 29.011 |

Class: Purchase (Num) | Visualize All

Status
OK

Log | x 0

Weka Explorer
Preprocess
Classify
Cluster
Associate
Select attributes
Visualize
Clusterer
Choose
SimpleKMeans -N 2 -A "weka.core.EuclideanDistance -R first-last" -I 500 -S 10
Cluster mode
Use training set
Supplied test set
Set...
Percentage split
%
66
Classes to clusters evaluation
(Nom) class
Store clusters for visualization
Ignore attributes
Start
Stop
Result list (right-click for options)
Clusterer output
Status
OK
Log
x 0

Weka Explorer
Preprocess
Classify
Cluster
Associate
Select attributes
Visualize
Clusterer
Choose
SimpleKMeans -N 2
ast" -I 500 -S 10
Show properties...
Copy configuration to clipboard
Enter configuration...
(right click)
Cluster mode
Use trainin
Supplied te
Percentage split
%
66
Classes to clusters evaluation
(Num) Purchase
Store clusters for visualization
Ignore attributes
Start
Stop
Result list (right-click for options)
Status
OK
Log
x 0

Weka Explorer
Preprocess
Classify
Cluster
Associate
Select attributes
Visualize
Clusterer
Choose
SimpleKMeans -N 5
Cluster mode
Use training set
Supplied test set
Percentage split
Classes to clusters evaluation
(Num) Purchase
Store clusters for visualization
Ignore attributes
Start
Result list (right-click for options)
weka.gui.GenericObjectEditor
weka.clusterers.SimpleKMeans
About
Cluster data using the k means algorithm.
More
Capabilities
displayStdDevs
False
distanceFunction
Choose
EuclideanDistance -R first-l
dontReplaceMissingValues
False
maxIterations
500
numClusters
5
preserveInstancesOrder
False
seed
10
Open...
Save...
OK
Cancel
Defaults to 2.
Change to 5 for now…
Status
OK
Log
x 0

Weka Explorer

Preprocess | Classify | Cluster | Associate | Select attributes | Visualize

**Clusterer**

Choose: **SimpleKMeans** -N 5 -A "weka.core.EuclideanDistance -R first-last" -I 500 -S 10

**Cluster mode**

- Use training set
- Supplied test set — Set...
- Percentage split — % 66
- Classes to clusters evaluation — (Num) Purchase
- Store clusters for visualization

Ignore attributes

Start | Stop

**Result list (right-click for options)**

- 13:02:35 - SimpleKMeans
- 13:26:31 - SimpleKMeans
- 13:50:09 - SimpleKMeans

**Clusterer output**

```
=== Run information ===

Scheme:weka.clusterers.SimpleKMeans -N 5 -A "weka.core.EuclideanDistance -R first-last" -I 500 -S 10
Relation:     car-browsers-weka.filters.unsupervised.attribute.Remove-R1
Instances:    100
Attributes:   8
              Dealership
              Showroom
              ComputerSearch
              M5
              3Series
              Z4
              Financing
              Purchase
Test mode:evaluate on training data

=== Model and evaluation on training set ===


kMeans
======

Number of iterations: 8
Within cluster sum of squared errors: 113.58260073260074
Missing values globally replaced with mean/mode

Cluster centroids:
                                Cluster#
Attribute         Full Data          0          1          2          3          4
                     (100)       (26)       (27)        (5)       (14)       (28)
==============================================================================
Dealership              0.6     0.9615     0.6667          1     0.8571          0
Showroom               0.72     0.6923     0.6667          0     0.5714          1
ComputerSearch         0.43     0.6538          0          1     0.8571     0.3214
M5                     0.53     0.4615      0.963          1     0.7143          0
3Series                0.55     0.3846     0.4444        0.8     0.0714          1
Z4                     0.45     0.5385          0        0.8     0.5714     0.6786
Financing              0.61     0.4615     0.6296        0.8          1        0.5
Purchase               0.39          0     0.5185        0.4          1     0.3214



Time taken to build model (full training data) : 0 seconds

=== Model and evaluation on training set ===

Clustered Instances

0       26 ( 26%)
1       27 ( 27%)
2        5 (  5%)
3       14 ( 14%)
4       28 ( 28%)
```

**Status**

OK

Log | x 0

# How to Interpret?

| Attribute | Full Data (100) | 0 (26) | 1 (27) | 2 (5) | 3 (14) | 4 (28) |
|---|---|---|---|---|---|---|
| Dealership | 0.6 | 0.9615 | 0.6667 | 1 | 0.8571 | 0 |
| Showroom | 0.72 | 0.6923 | 0.6667 | 0 | 0.5714 | 1 |
| ComputerSearch | 0.43 | 0.6538 | 0 | 1 | 0.8571 | 0.3214 |
| M5 | 0.53 | 0.4615 | 0.963 | 1 | 0.7143 | 0 |
| 3Series | 0.55 | 0.3846 | 0.4444 | 0.8 | 0.0714 | 1 |
| Z4 | 0.45 | 0.5385 | 0 | 0.8 | 0.5714 | 0.6786 |
| Financing | 0.61 | 0.4615 | 0.6296 | 0.8 | 1 | 0.5 |
| Purchase | 0.39 | 0 | 0.5185 | 0.4 | 1 | 0.3214 |

1 = everyone in that cluster shares the same value of one
0 = everyone in that cluster has a value of zero for that attribute
Numbers = average value of everyone in the cluster

```
Time taken to build model (full training data) : 0 seconds

=== Model and evaluation on training set ===

Clustered Instances

0       16 ( 16%)
1       32 ( 32%)
2        6 (  6%)
3       17 ( 17%)
4       29 ( 29%)
```

# How to Interpret?

| Attribute | Full Data (100) | 0 (26) | 1 (27) | 2 (5) | 3 (14) | 4 (28) |
|---|---|---|---|---|---|---|
| Dealership | 0.6 | 0.9615 | 0.6667 | 1 | 0.8571 | 0 |
| Showroom | 0.72 | 0.6923 | 0.6667 | 0 | 0.5714 | 1 |
| ComputerSearch | 0.43 | 0.6538 | 0 | 1 | 0.8571 | 0.3214 |
| M5 | 0.53 | 0.4615 | 0.963 | 1 | 0.7143 | 0 |
| 3Series | 0.55 | 0.3846 | 0.4444 | 0.8 | 0.0714 | 1 |
| Z4 | 0.45 | 0.5385 | 0 | 0.8 | 0.5714 | 0.6786 |
| Financing | 0.61 | 0.4615 | 0.6296 | 0.8 | 1 | 0.5 |
| Purchase | 0.39 | 0 | 0.5185 | 0.4 | 1 | 0.3214 |

## BMW Babies

Always end up purchasing a car and financing it
Walk around the lot looking at cars, then turn to the computer search available at the dealership
Buy M5s or Z4, never 3-series

## Starting Out With BMW

Always look at the 3-series, never look at the much more expensive M5
Walk right into the showroom, choosing not to walk around the lot
Ignore the computer search terminals
Nearly 50 percent get to the financing stage, but only 32 percent ultimately finish the transaction

## The Dreamers

Wander around the dealership, looking at cars parked outside on the lots, but trail off when it comes to coming into the dealership. Don't purchase anything.

## M5 Lovers

Walk straight to the M5s, ignoring the 3-series cars and the Z4
Have a relatively high purchase rate

## Throw-Aways

Aren't statistically relevent, can't draw any good conclusions from their behavior (Reduce the number of clusters?)

Weka Explorer

Preprocess | Classify | Cluster | Associate | Select attributes | Visualize

**Clusterer**

Choose: **SimpleKMeans** -N 5 -A "weka.core.EuclideanDistance -R first-last" -I 500 -S 10

**Cluster mode**

- Use training set
- Supplied test set — Set...
- Percentage split — % 66
- Classes to clusters evaluation — (Num) Purchase
- Store clusters for visualization

Ignore attributes

Start | Stop

**Result list (right-click for options)**

13:02:35 - SimpleKMeans
13:26:31 - SimpleKMeans

- View in main window
- View in separate window
- Save result buffer
- Delete result buffer
- Load model
- Save model
- Re-evaluate model on current test set
- Visualize cluster assignments
- Visualize tree

**Clusterer output**

```
=== Run information ===

Scheme:weka.clusterers.SimpleKMeans -N 5 -A "weka.core.EuclideanDistance -R first-last" -I 500 -S 10
Relation:     car-browsers
Instances:    100
Attributes:   9
              CustomerID
              Dealership
              Showroom
              ComputerSearch
              M5
              3Series
              Z4
              Financing
              Purchase
Test mode:evaluate on training data

=== Model and evaluation on training set ===


kMeans
======

Number of iterations: 7
Within cluster sum of squared errors: 119.97220997072645
Missing values globally replaced with mean/mode

Cluster centroids:
                                   Cluster#
Attribute                Full Data          0          1          2          3          4
                           (100)       (16)       (32)        (6)       (17)       (29)
=================================================================================================
CustomerID                   50.5     29.625      60.25    10.8333    26.8235    73.3448
Dealership                    0.6     0.9375     0.6875          1          1          0
Showroom                     0.72          1     0.7188     0.1667     0.1765          1
ComputerSearch               0.43     0.5625     0.2188          1     0.7059     0.3103
M5                           0.53      0.625     0.9688     0.8333     0.3529     0.0345
3Series                      0.55      0.625     0.3125          1          0          1
Z4                           0.45     0.8125     0.1875        0.5     0.2353     0.6552
Financing                    0.61        0.5     0.8125     0.6667     0.5294     0.4828
Purchase                     0.39          0     0.6875        0.5     0.2941     0.3103


Time taken to build model (full training data) : 0 seconds

=== Model and evaluation on training set ===

Clustered Instances

0       16 ( 16%)
1       32 ( 32%)
2        6 (  6%)
3       17 ( 17%)
4       29 ( 29%)
```

**Status**

OK

Log | x 0

Weka Clusterer Visualize: 13:26:31 - SimpleKMeans (car-browsers)
X: Instance_number (Num)
Y: CustomerID (Num)
Colour: Cluster (Nom)
Select Instance
Reset
Clear
Open
Save
Jitter
Plot:car-browsers_clustered
100
50.5
1
0
49.5
99
X
Y
Class colour
cluster0 cluster1 cluster2 cluster3 cluster4

Weka Clusterer Visualize: 13:50:09 - SimpleKMeans (car-browsers-weka.filters.unsupervised.attribute.Remove-R1)
X: M5 (Num)
Y: Purchase (Num)
Colour: Cluster (Nom)
Select Instance
Reset
Clear
Open
Save
Jitter
Plot:car-browsers-weka.filters.unsupervised.attribute.Remove-R1_clustered
X=1, Y=1 (looked at M5s and financed)
Only clusters 1 and 3 represented
X=0, Y=0 (looked at M5s and financed)
Only clusters 4 and 0 represented
1
0.5
0
0
0.5
1
X
Y
Class colour
cluster0 cluster1 cluster2 cluster3 cluster4

# EM Algorithm

- Expectation Maximization (EM)
  - Start with initial values for the parameters
  - Calculate the cluster probabilities for each instance
  - Re-estimate the values for the parameters
  - Repeat
- General purpose maximum likelihood estimate algorithm for missing data
  - Can also be used to train Bayesian networks (later)

# Beyond Normal Models

- More than one class:
  - Straightforward
- More than one numeric attribute
  - Easy if assume attributes independent
  - If dependent attributes, treat them jointly using the bivariate normal
- Nominal attributes
  - No more normal distribution!

# EM using Weka

- Options
  - numClusters : set number of clusters.
    - » Default = -1 selects it automatically
  - maxIterations :  maximum number of iterations
  - seed : random number seed
  - minStdDev : set minimum allowable standard deviation

Weka Explorer

Preprocess | Classify | Cluster | Associate | Select attributes | Visualize

**Clusterer**

Choose: EM -I 100 -N -1 -M 1.0E-6 -S 100

**Cluster mode**

- Use training set
- Supplied test set
- Percentage split
- Classes to clusters evaluati
  - (Nom) class
- Store clusters for visualizat

Ignore attribute

Start

**Result list (right-click for optio**

```
11:26:23 - SimpleKMeans
11:27:00 - SimpleKMeans
11:28:04 - SimpleKMeans
11:31:44 - SimpleKMeans
11:33:12 - HierarchicalClusterer
11:33:47 - SimpleKMeans
11:36:50 - SimpleKMeans
11:42:09 - EM
11:52:23 - EM
11:53:04 - EM
11:55:21 - Cobweb
11:58:13 - Cobweb
```

**weka.gui.GenericObjectEditor**

weka.clusterers.EM

About

Simple EM (expectation maximisation) class.

More | Capabilities

| | |
|---|---|
| debug | False |
| displayModelInOldFormat | False |
| maxIterations | 100 |
| minStdDev | 1.0E-6 |
| numClusters | -1 |
| seed | 100 |

Open... | Save... | OK | Cancel

```
|   node 24 [20]
|   |   node 25 [19]
|   |   |   leaf 26 [18]
|   |   node 25 [19]
|   |   |   leaf 27 [1]
|   node 24 [20]
|   |   leaf 28 [1]


Time taken to build model (full training data) : 0.03 seconds

=== Model and evaluation on training set ===

Clustered Instances

 2       2 (  3%)
 4       1 (  2%)
 5       1 (  2%)
 8       9 ( 15%)
10       1 (  2%)
11       1 (  2%)
12       1 (  2%)
13       1 (  2%)
15       1 (  2%)
16       1 (  2%)
17       1 (  2%)
19       1 (  2%)
20      17 ( 28%)
22       1 (  2%)
23       1 (  2%)
26      18 ( 30%)
28       2 (  3%)
```

Status: OK

Log | x 0

Weka Explorer

Preprocess | Classify | Cluster | Associate | Select attributes | Visualize

**Clusterer**

Choose: EM -I 100 -N -1 -M 1.0E-6 -S 100

**Cluster mode**

- Use training set
- Supplied test set — Set...
- Percentage split — % 66
- Classes to clusters evaluation — (Nom) class
- Store clusters for visualization

Ignore attributes

Start | Stop

**Result list (right-click for options)**

```
11:26:23 - SimpleKMeans
11:27:00 - SimpleKMeans
11:28:04 - SimpleKMeans
11:31:44 - SimpleKMeans
11:33:12 - HierarchicalClusterer
11:33:47 - SimpleKMeans
11:36:50 - SimpleKMeans
11:42:09 - EM
```

**Clusterer output**

```
  mean                     0       0  0.0625       0  0.3889
  std. dev.           0.3891  0.3891  0.2421  0.3891  0.5906

cup
  mean                     0  0.4375       0  0.1667       0
  std. dev.           0.5031  0.8638  0.5031  0.3727  0.5031

give
  mean                     0  0.4375       0  0.1667       0
  std. dev.           0.5031  0.8638  0.5031  0.3727  0.5031

interested
  mean                  0.25       0   0.375       0  0.0556
  std. dev.            0.433  0.3428  0.4841  0.3428  0.2291

world
  mean                  0.25  0.4375       0       0       0
  std. dev.            0.433  0.6092  0.3891  0.3891  0.3891

forsale
  mean                  1.75       0       0       0       0
  std. dev.           0.8292  0.4903  0.4903  0.4903  0.4903

library
  mean                     0       0       0       0  0.3889
  std. dev.           0.4155  0.4155  0.4155  0.4155  0.6781

program
  mean                     0       0       0       0  0.3333
  std. dev.           0.3542  0.3542  0.3542  0.3542  0.5774

version
  mean                     0       0       0       0  0.2778
  std. dev.           0.2787  0.2787  0.2787  0.2787  0.4479

brand
  mean                     0       0  0.1875       0       0
  std. dev.           0.2198  0.2198  0.3903       0  0.2198

fact
  mean                     0  0.1875       0       0       0
  std. dev.           0.2198  0.3903  0.2198  0.2198  0.2198

class
  misc.forsale             5       2      15       2       1
  rec.sport.hockey         1      16       1       6       1
  comp.graphics            1       1  2.9999  1.0001      19
  [total]                  7      19 18.9999  9.0001      21


Time taken to build model (full training data) : 1.06 seconds

=== Model and evaluation on training set ===

Clustered Instances

0       4 (  7%)
1      12 ( 20%)
2      12 ( 20%)
3      11 ( 18%)
4      21 ( 35%)


Log likelihood: -18.17848
```

**Status**

OK

Log | x 0

Weka Explorer

Preprocess | Classify | Cluster | Associate | Select attributes | Visualize

**Clusterer**

Choose: **Cobweb** -A 1.0 -C 0.0028209479177387815 -S 42

**Cluster mode**

- Use training set
- Supplied test set — Set...
- Percentage split — % 66
- Classes to clusters evaluation
  - (Nom) play
- Store clusters for visualization

Ignore attributes

Start | Stop

**Result list (right-click for options)**

```
11:26:23 - SimpleKMeans
11:27:00 - SimpleKMeans
11:28:04 - SimpleKMeans
11:31:44 - SimpleKMeans
11:33:12 - HierarchicalClusterer
11:33:47 - SimpleKMeans
11:36:50 - SimpleKMeans
11:42:09 - EM
11:52:23 - EM
11:53:04 - EM
11:55:21 - Cobweb
```

**Clusterer output**

```
                play
Test mode:evaluate on training data

=== Model and evaluation on training set ===

Number of merges: 1
Number of splits: 0
Number of clusters: 21

node 0 [14]
|   node 1 [5]
|   |   leaf 2 [1]
|   node 1 [5]
|   |   leaf 3 [1]
|   node 1 [5]
|   |   node 4 [2]
|   |   |   leaf 5 [1]
|   |   node 4 [2]
|   |   |   leaf 6 [1]
|   node 1 [5]
|   |   leaf 7 [1]
node 0 [14]
|   node 8 [6]
|   |   node 9 [2]
|   |   |   leaf 10 [1]
|   |   node 9 [2]
|   |   |   leaf 11 [1]
|   node 8 [6]
|   |   leaf 12 [1]
|   node 8 [6]
|   |   node 13 [3]
|   |   |   leaf 14 [1]
|   |   node 13 [3]
|   |   |   leaf 15 [1]
|   |   node 13 [3]
|   |   |   leaf 16 [1]
node 0 [14]
|   node 17 [3]
|   |   leaf 18 [1]
|   node 17 [3]
|   |   leaf 19 [1]
|   node 17 [3]
|   |   leaf 20 [1]


Time taken to build model (ful

=== Model and evaluation on tr

Clustered Instances

 2       1 (  7%)
 3       1 (  7%)
 5       1 (  7%)
 6       1 (  7%)
 7       1 (  7%)
10       1 (  7%)
11       1 (  7%)
12       1 (  7%)
14       1 (  7%)
15       1 (  7%)
16       1 (  7%)
18       1 (  7%)
19       1 (  7%)
20       1 (  7%)
```

weather.arff

Weka Classifier Tree Visualizer: 11:58:13 - Cobweb (example-weka....

Tree View

Status: OK

Log

x 0