THE UNIVERSITY OF
WAIKATO
Te Whare Wānanga o Waikato

WEKA KnowledgeFlow Tutorial for Version 3-5-8

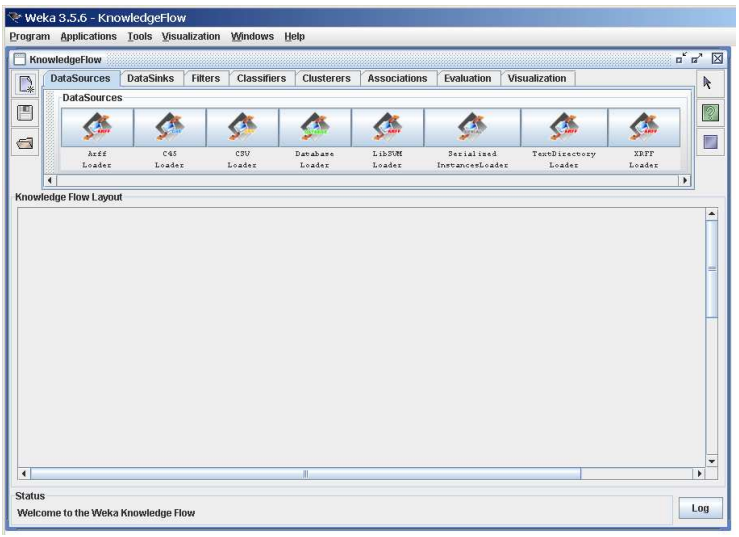
Mark Hall
Peter Reutemann

May 21, 2008

Contents

1 Introduction

The KnowledgeFlow provides an alternative to the Explorer as a graphical front end to WEKA's core algorithms. The KnowledgeFlow is a work in progress so some of the functionality from the Explorer is not yet available. On the other hand, there are things that can be done in the KnowledgeFlow but not in the Explorer.



The KnowledgeFlow presents a *data-flow* inspired interface to WEKA. The user can select WEKA components from a tool bar, place them on a layout canvas and connect them together in order to form a *knowledge flow* for processing and analyzing data. At present, all of WEKA's classifiers, filters, clusterers, loaders and savers are available in the KnowledgeFlow along with some extra tools.

The KnowledgeFlow can handle data either incrementally or in batches (the Explorer handles batch data only). Of course learning from data incrementally requires a classifier that can be updated on an instance by instance basis. Currently in WEKA there are ten classifiers that can handle data incrementally:

- AODE
- IB1
- IBk
- KStar
- NaiveBayesMultinomialUpdateable
- NaiveBayesUpdateable
- NNge
- Winnow

And two of them are meta classifiers:

- *RacedIncrementalLogitBoost* - that can use of any regression base learner to learn from discrete class data incrementally.
- *LWL* - locally weighted learning.

This manual is also available online on the *WekaDoc Wiki* [2].

2 Features

The KnowledgeFlow offers the following features:

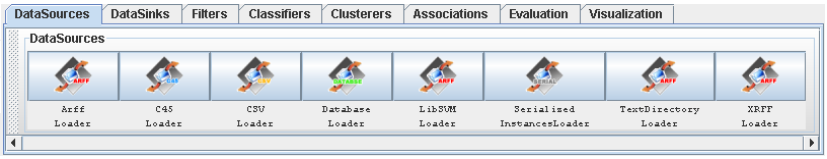
- intuitive data flow style layout
- process data in batches or incrementally
- process multiple batches or streams in parallel (each separate flow executes in its own thread)
- chain filters together
- view models produced by classifiers for each fold in a cross validation
- visualize performance of incremental classifiers during processing (scrolling plots of classification accuracy, RMS error, predictions etc.)
- plugin facility for allowing easy addition of new components to the KnowledgeFlow

3 Components

Components available in the KnowledgeFlow:

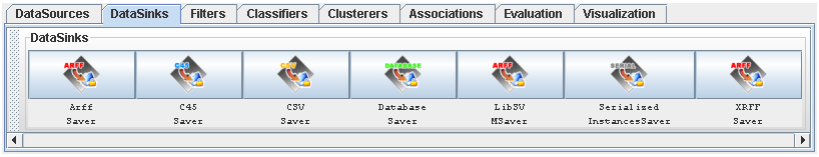
3.1 DataSources

All of WEKA's loaders are available.



3.2 DataSinks

All of WEKA's savers are available.

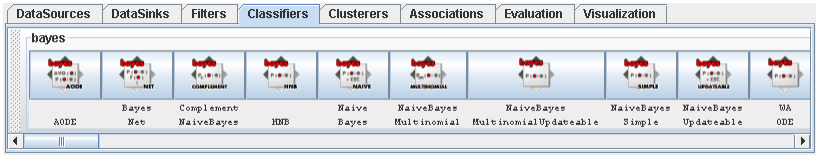


3.3 Filters

All of WEKA's filters are available.

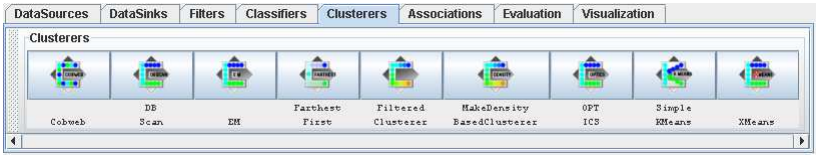
3.4 Classifiers

All of WEKA's classifiers are available.

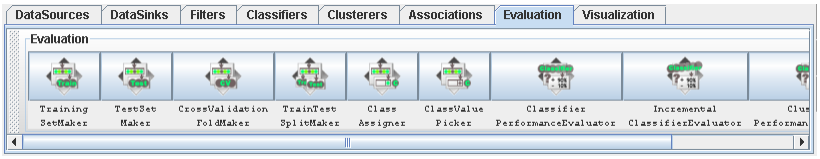


3.5 Clusterers

All of WEKA's clusterers are available.



3.6 Evaluation



- *TrainingSetMaker* - make a data set into a training set.
- *TestSetMaker* - make a data set into a test set.
- *CrossValidationFoldMaker* - split any data set, training set or test set into folds.
- *TrainTestSplitMaker* - split any data set, training set or test set into a training set and a test set.
- *ClassAssigner* - assign a column to be the class for any data set, training set or test set.
- *ClassValuePicker* - choose a class value to be considered as the "positive" class. This is useful when generating data for ROC style curves (see *ModelPerformanceChart* below and example 4.2).
- *ClassifierPerformanceEvaluator* - evaluate the performance of batch trained/tested classifiers.
- *IncrementalClassifierEvaluator* - evaluate the performance of incrementally trained classifiers.
- *ClustererPerformanceEvaluator* - evaluate the performance of batch trained/tested clusterers.
- *PredictionAppender* - append classifier predictions to a test set. For discrete class problems, can either append predicted class labels or probability distributions.

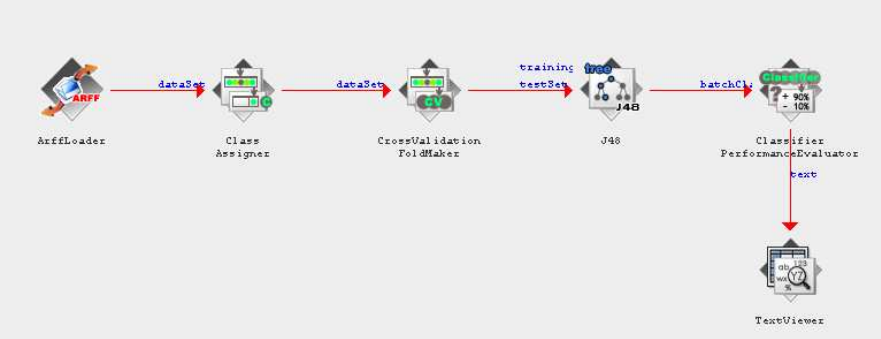
3.7 Visualization

- *DataVisualizer* - component that can pop up a panel for visualizing data in a single large 2D scatter plot.
- *ScatterPlotMatrix* - component that can pop up a panel containing a matrix of small scatter plots (clicking on a small plot pops up a large scatter plot).
- *AttributeSummarizer* - component that can pop up a panel containing a matrix of histogram plots - one for each of the attributes in the input data.
- *ModelPerformanceChart* - component that can pop up a panel for visualizing threshold (i.e. ROC style) curves.
- *TextViewer* - component for showing textual data. Can show data sets, classification performance statistics etc.
- *GraphViewer* - component that can pop up a panel for visualizing tree based models.
- *StripChart* - component that can pop up a panel that displays a scrolling plot of data (used for viewing the online performance of incremental classifiers).

4 Examples

4.1 Cross-validated J48

Setting up a flow to load an ARFF file (batch mode) and perform a cross-validation using J48 (WEKA's C4.5 implementation).



- Click on the DataSources tab and choose *ArffLoader* from the toolbar (the mouse pointer will change to a *cross hairs*).
- Next place the ArffLoader component on the layout area by clicking somewhere on the layout (a copy of the ArffLoader icon will appear on the layout area).
- Next specify an ARFF file to load by first right clicking the mouse over the ArffLoader icon on the layout. A pop-up menu will appear. Select *Configure* under *Edit* in the list from this menu and browse to the location of your ARFF file.
- Next click the *Evaluation* tab at the top of the window and choose the *ClassAssigner* (allows you to choose which column to be the class) component from the toolbar. Place this on the layout.
- Now connect the ArffLoader to the ClassAssigner: first right click over the ArffLoader and select the *dataSet* under *Connections* in the menu. A *rubber band* line will appear. Move the mouse over the ClassAssigner component and left click - a red line labeled *dataSet* will connect the two components.
- Next right click over the ClassAssigner and choose *Configure* from the menu. This will pop up a window from which you can specify which column is the class in your data (last is the default).
- Next grab a *CrossValidationFoldMaker* component from the Evaluation toolbar and place it on the layout. Connect the ClassAssigner to the CrossValidationFoldMaker by right clicking over *ClassAssigner* and selecting *dataSet* from under *Connections* in the menu.
- Next click on the *Classifiers* tab at the top of the window and scroll along the toolbar until you reach the *J48* component in the *trees* section. Place a J48 component on the layout.

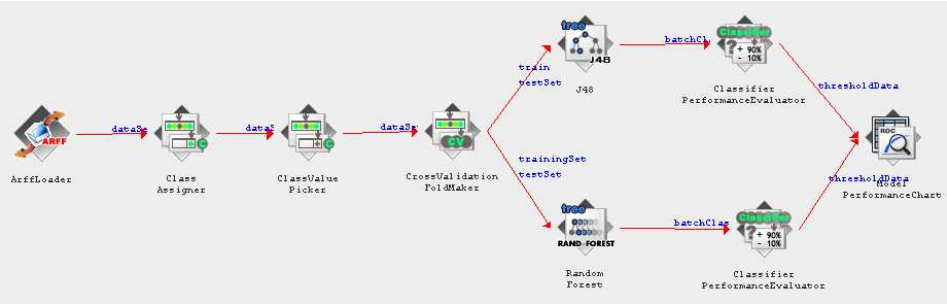
- Connect the CrossValidationFoldMaker to J48 TWICE by first choosing *trainingSet* and then *testSet* from the pop-up menu for the CrossValidationFoldMaker.
- Next go back to the *Evaluation* tab and place a *ClassifierPerformanceEvaluator* component on the layout. Connect J48 to this component by selecting the *batchClassifier* entry from the pop-up menu for J48.
- Next go to the *Visualization* toolbar and place a *TextViewer* component on the layout. Connect the ClassifierPerformanceEvaluator to the TextViewer by selecting the *text* entry from the pop-up menu for ClassifierPerformanceEvaluator.
- Now start the flow executing by selecting *Start loading* from the pop-up menu for ArffLoader. Depending on how big the data set is and how long cross-validation takes you will see some animation from some of the icons in the layout (J48's tree will *grow* in the icon and the ticks will animate on the ClassifierPerformanceEvaluator). You will also see some progress information in the *Status* bar and *Log* at the bottom of the window.

When finished you can view the results by choosing *Show results* from the pop-up menu for the *TextViewer* component.

Other cool things to add to this flow: connect a *TextViewer* and/or a *GraphViewer* to J48 in order to view the textual or graphical representations of the trees produced for each fold of the cross validation (this is something that is not possible in the Explorer).

4.2 Plotting multiple ROC curves

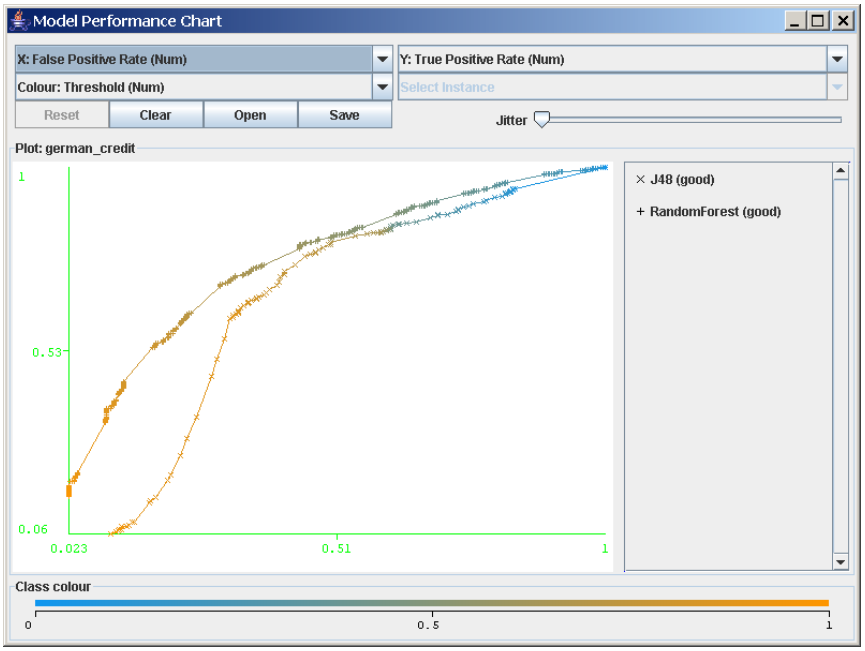
The KnowledgeFlow can draw multiple ROC curves in the same plot window, something that the Explorer cannot do. In this example we use *J48* and *RandomForest* as classifiers. This example can be found on the *WekaWiki* as well [4].



- Click on the DataSources tab and choose *ArffLoader* from the toolbar (the mouse pointer will change to a *cross hairs*).
- Next place the ArffLoader component on the layout area by clicking somewhere on the layout (a copy of the ArffLoader icon will appear on the layout area).
- Next specify an ARFF file to load by first right clicking the mouse over the ArffLoader icon on the layout. A pop-up menu will appear. Select *Configure* under *Edit* in the list from this menu and browse to the location of your ARFF file.
- Next click the *Evaluation* tab at the top of the window and choose the *ClassAssigner* (allows you to choose which column to be the class) component from the toolbar. Place this on the layout.
- Now connect the ArffLoader to the ClassAssigner: first right click over the ArffLoader and select the *dataSet* under *Connections* in the menu. A *rubber band* line will appear. Move the mouse over the ClassAssigner component and left click - a red line labeled *dataSet* will connect the two components.
- Next right click over the ClassAssigner and choose *Configure* from the menu. This will pop up a window from which you can specify which column is the class in your data (last is the default).
- Next choose the *ClassValuePicker* (allows you to choose which class label to be evaluated in the ROC) component from the toolbar. Place this on the layout and right click over *ClassAssigner* and select *dataSet* from under *Connections* in the menu and connect it with the *ClassValuePicker*.
- Next grab a *CrossValidationFoldMaker* component from the Evaluation toolbar and place it on the layout. Connect the ClassAssigner to the CrossValidationFoldMaker by right clicking over *ClassAssigner* and selecting *dataSet* from under *Connections* in the menu.

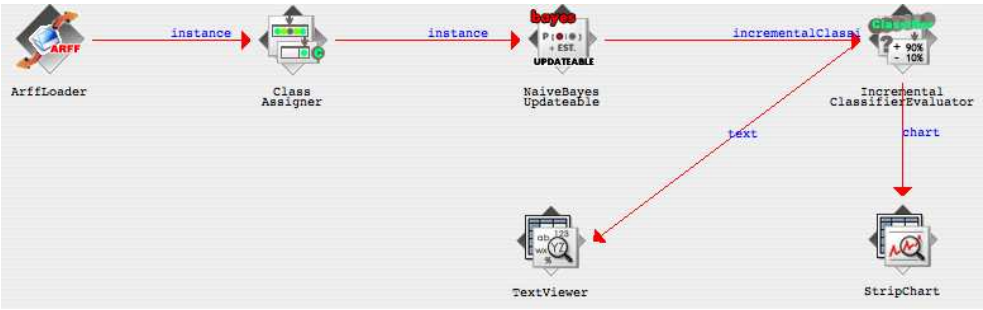
- Next click on the *Classifiers* tab at the top of the window and scroll along the toolbar until you reach the *J48* component in the *trees* section. Place a J48 component on the layout.
- Connect the CrossValidationFoldMaker to J48 TWICE by first choosing *trainingSet* and then *testSet* from the pop-up menu for the CrossValidationFoldMaker.
- Repeat these two steps with the RandomForest classifier.
- Next go back to the *Evaluation* tab and place a *ClassifierPerformanceEvaluator* component on the layout. Connect J48 to this component by selecting the *batchClassifier* entry from the pop-up menu for J48. Add another *ClassifierPerformanceEvaluator* for RandomForest and connect them via *batchClassifier* as well.
- Next go to the *Visualization* toolbar and place a *ModelPerformanceChart* component on the layout. Connect both ClassifierPerformanceEvaluators to the ModelPerformanceChart by selecting the *thresholdData* entry from the pop-up menu for ClassifierPerformanceEvaluator.
- Now start the flow executing by selecting *Start loading* from the pop-up menu for ArffLoader. Depending on how big the data set is and how long cross validation takes you will see some animation from some of the icons in the layout. You will also see some progress information in the *Status* bar and *Log* at the bottom of the window.
- Select *Show plot* from the popup-menu of the *ModelPerformanceChart* under the *Actions* section.

Here are the two ROC curves generated from the UCI dataset *credit-g*, evaluated on the class label *good*:



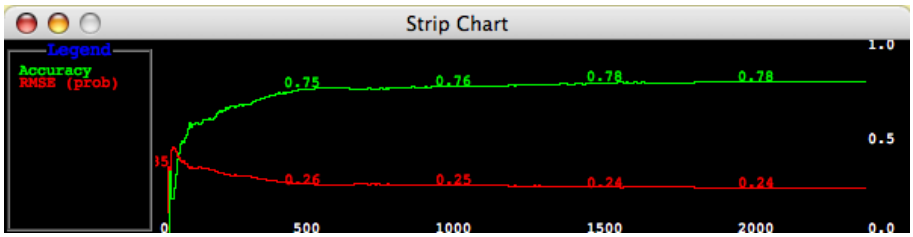
4.3 Processing data incrementally

Some classifiers, clusterers and filters in Weka can handle data incrementally in a streaming fashion. Here is an example of training and testing *naive Bayes* incrementally. The results are sent to a *TextViewer* and predictions are plotted by a *StripChart* component.

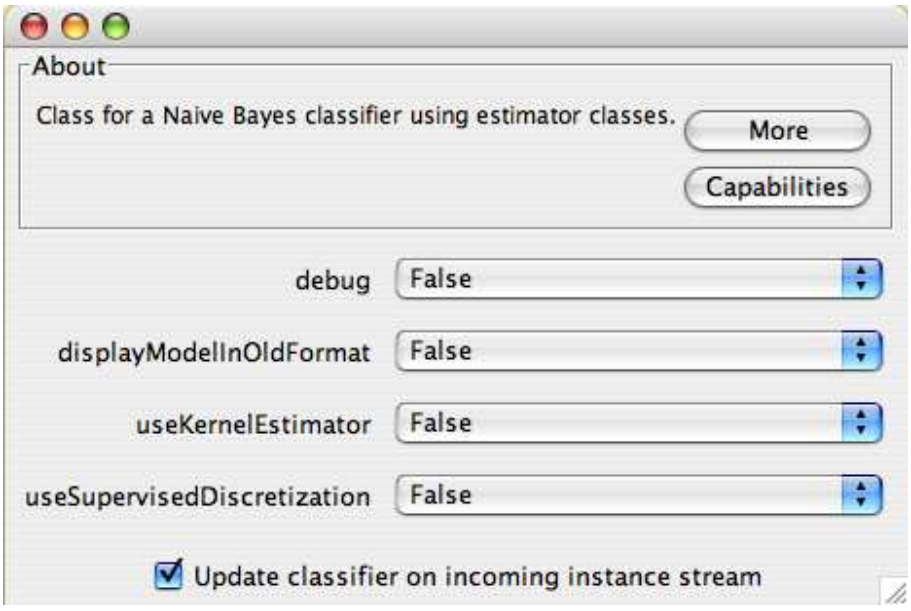


- Click on the DataSources tab and choose *ArffLoader* from the toolbar (the mouse pointer will change to a *cross hairs*).
- Next place the ArffLoader component on the layout area by clicking somewhere on the layout (a copy of the ArffLoader icon will appear on the layout area).
- Next specify an ARFF file to load by first right clicking the mouse over the ArffLoader icon on the layout. A pop-up menu will appear. Select *Configure* under *Edit* in the list from this menu and browse to the location of your ARFF file.
- Next click the *Evaluation* tab at the top of the window and choose the *ClassAssigner* (allows you to choose which column to be the class) component from the toolbar. Place this on the layout.
- Now connect the ArffLoader to the ClassAssigner: first right click over the ArffLoader and select the *dataSet* under *Connections* in the menu. A *rubber band* line will appear. Move the mouse over the ClassAssigner component and left click - a red line labeled *dataSet* will connect the two components.
- Next right click over the *ClassAssigner* and choose *Configure* from the menu. This will pop up a window from which you can specify which column is the class in your data (last is the default).
- Now grab a *NaiveBayesUpdateable* component from the *bayes* section of the *Classifiers* panel and place it on the layout.
- Next connect the *ClassAssigner* to *NaiveBayesUpdateable* using a *instance* connection.
- Next place an *IncrementalClassiferEvaluator* from the *Evaluation* panel onto the layout and connect *NaiveBayesUpdateable* to it using a *incrementalClassifier* connection.

- Next place a *TextViewer* component from the *Visualization* panel on the Layout. Connect the *IncrementalClassifierEvaluator* to it using a *text* connection.
- Next place a *StripChart* component from the *Visualization* panel on the layout and connect *IncrementalClassifierEvaluator* to it using a *chart* connection.
- Display the *StripChart's* chart by right-clicking over it and choosing *Show chart* from the pop-up menu. Note: the *StripChart* can be configured with options that control how often data points and labels are displayed.
- Finally, start the flow by right-clicking over the *ArffLoader* and selecting *Start loading* from the pop-up menu.



Note that, in this example, a prediction is obtained from naive Bayes for each incoming instance **before** the classifier is trained (updated) with the instance. If you have a pre-trained classifier, you can specify that the classifier **not** be updated on incoming instances by unselecting the check box in the configuration dialog for the classifier. If the pre-trained classifier is a **batch** classifier (i.e. it is not capable of incremental training) then you will only be able to test it in an incremental fashion.



5 Plugin Facility

The KnowledgeFlow offers the ability to easily add new components via a plugin mechanism. Plugins are installed in a directory called `.knowledgeFlow/plugins` in the user's home directory. If this directory does not exist you must create it in order to install plugins. Plugins are installed in subdirectories of the `.knowledgeFlow/plugins` directory. More than one plugin component may reside in the same subdirectory. Each subdirectory should contain jar file(s) that contain and support the plugin components. The KnowledgeFlow will dynamically load jar files and add them to the classpath. In order to tell the KnowledgeFlow which classes in the jar files to instantiate as components, a second file called `Beans.props` needs to be created and placed into each plugin subdirectory. This file contains a list of fully qualified class names to be instantiated. Successfully instantiated components will appear in a "Plugins" tab in the KnowledgeFlow user interface.Below is an example plugin directory listing, the listing of the contents of the jar file and the contents of the associated Beans.props file:

```
cygnus:~ mhall$ ls -l $HOME/.knowledgeFlow/plugins/kettle/
total 24
-rw-r--r--   1 mhall  mhall   117 20 Feb 10:56 Beans.props
-rw-r--r--   1 mhall  mhall  8047 20 Feb 14:01 kettleKF.jar

cygnus:~ mhall$ jar tvf /Users/mhall/.knowledgeFlow/plugins/kettle/kettleKF.jar
     0 Wed Feb 20 14:01:34 NZDT 2008 META-INF/
    70 Wed Feb 20 14:01:34 NZDT 2008 META-INF/MANIFEST.MF
     0 Tue Feb 19 14:59:08 NZDT 2008 weka/
     0 Tue Feb 19 14:59:08 NZDT 2008 weka/gui/
     0 Wed Feb 20 13:55:52 NZDT 2008 weka/gui/beans/
     0 Wed Feb 20 13:56:36 NZDT 2008 weka/gui/beans/icons/
  2812 Wed Feb 20 14:01:20 NZDT 2008 weka/gui/beans/icons/KettleInput.gif
  2812 Wed Feb 20 14:01:18 NZDT 2008 weka/gui/beans/icons/KettleInput_animated.gif
  1839 Wed Feb 20 13:59:08 NZDT 2008 weka/gui/beans/KettleInput.class
   174 Tue Feb 19 15:27:24 NZDT 2008 weka/gui/beans/KettleInputBeanInfo.class

cygnus:~ mhall$ more /Users/mhall/.knowledgeFlow/plugins/kettle/Beans.props
# Specifies the tools to go into the Plugins toolbar
weka.gui.beans.KnowledgeFlow.Plugins=weka.gui.beans.KettleInput
```

References

[1] Witten, I.H. and Frank, E. (2005) *Data Mining: Practical machine learning tools and techniques. 2nd edition* Morgan Kaufmann, San Francisco.

[2] *WekaDoc* – `http://weka.sourceforge.net/wekadoc/`

[3] *WekaWiki* – `http://weka.sourceforge.net/wekawiki/`

[4] *Plotting multiple ROC curves* on *WekaWiki* – `http://weka.sourceforge.net/wiki/index.php/Plotting_multiple_ROC_curves`