WEKA DATA MINING SYSTEM
Weka Experiment Environment

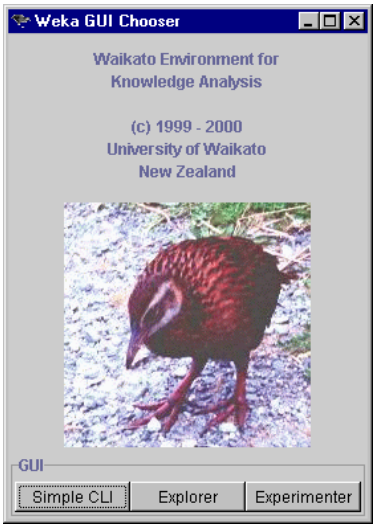
Introduction

The Weka Experiment Environment enables the user to create, run, modify, and analyse experiments in a more convenient manner than is possible when processing the schemes individually. For example, the user can create an experiment that runs several schemes against a series of datasets and then analyse the results to determine if one of the schemes is (statistically) better than the other schemes.

The Experiment Environment can be run from the command line using the Simple CLI. For example, the following commands could be typed into the CLI to run the OneR scheme on the Iris dataset using a basic train and test process. (Note that the commands would be typed on one line into the CLI.)

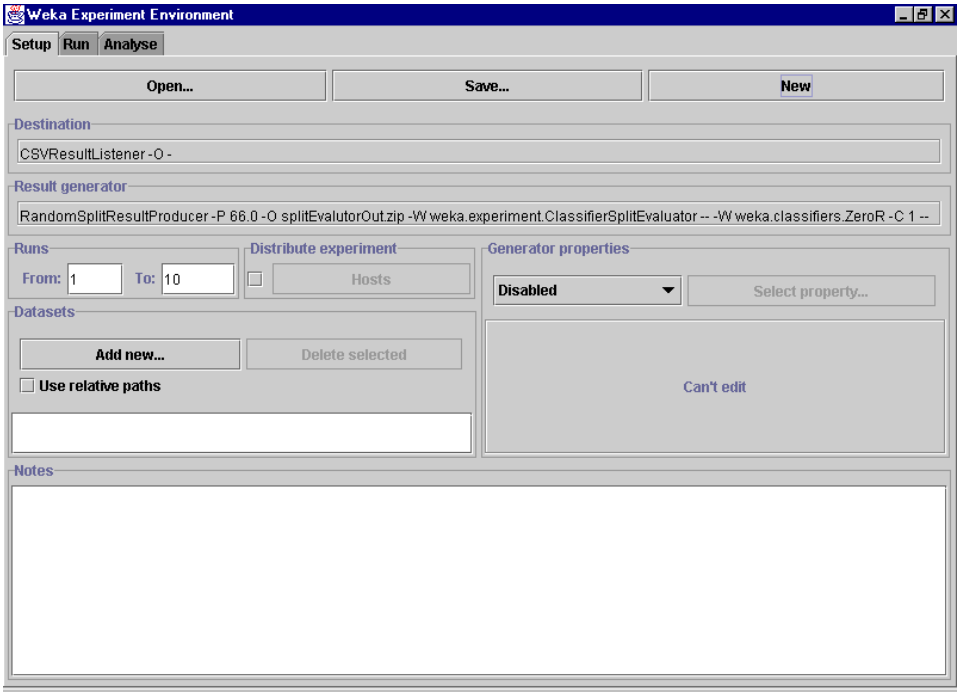
```
java weka.experiment.Experiment -r -T data/iris.arff
  -D weka.experiment.InstancesResultListener
  -P weka.experiment.RandomSplitResultProducer --
  -W weka.experiment.ClassifierSplitEvaluator --
  -W weka.classifiers.OneR
```

While commands can be typed directly into the CLI, this technique is not particularly convenient and the experiments are not easy to modify. The Weka-3-1-9 system includes a GUI that provides the user with more flexibility when developing experiments than is possible by typing commands into the CLI. Some basic documentation is included in the README_Experiment_Gui file in the Weka folder. To begin the Experiment Environment GUI, start Weka and click on Experimenter in the Weka GUI Chooser window.

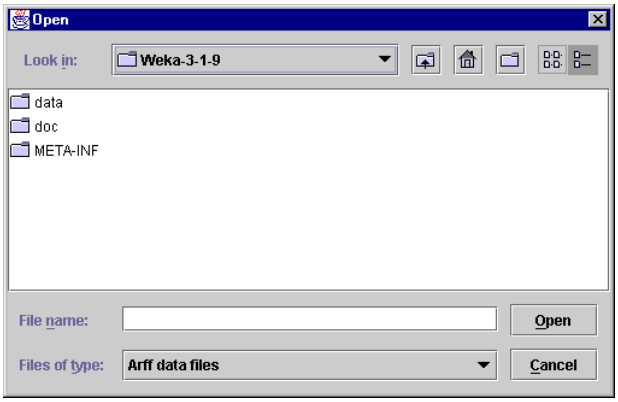


Defining an Experiment

When the Experimenter is started, the Setup window (actually a pane) is displayed. Click New to initialize an experiment. This causes default parameters to be defined for the experiment.

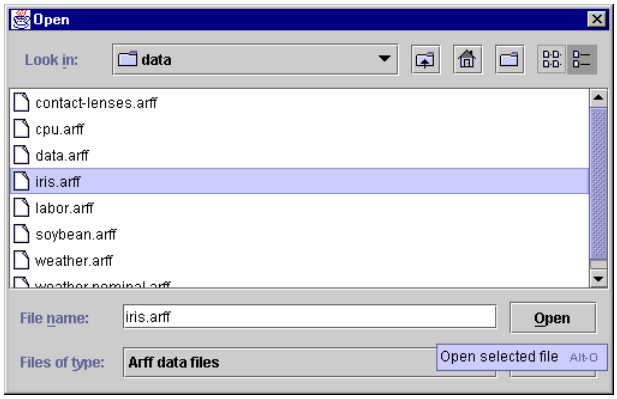


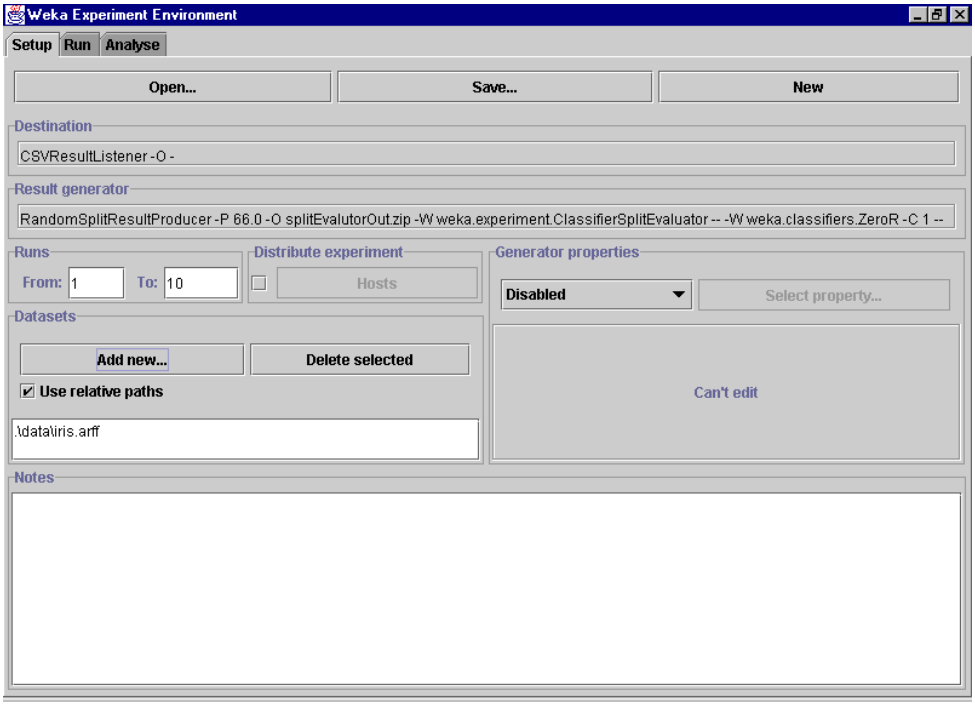
To define the dataset to be processed by a scheme, first select "Use relative paths" in the Datasets panel of the Setup window and then click on "Add new …" to open a dialog window.



Double click on the "data" folder to view the available datasets or navigate to an alternate location.

Select iris.arff and click Open to select the Iris dataset.

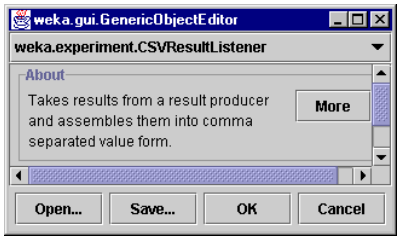




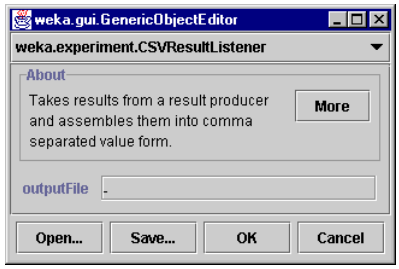
The dataset name is now displayed in the Datasets panel of the Setup window.

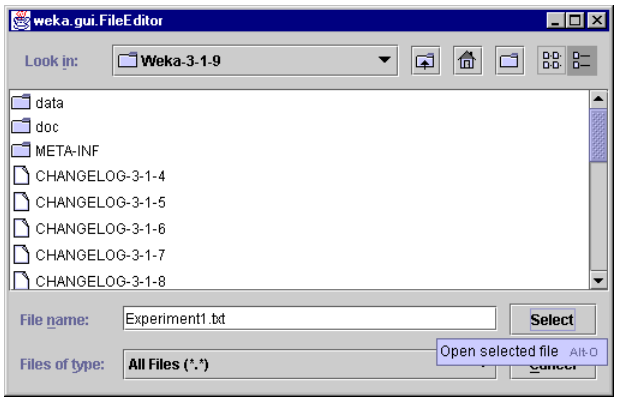
Saving the Results of the Experiment

To identify a dataset to which the results are to be sent, click on the "CSVResultListener" entry in the Destination panel. Note that this window (and other similar windows in Weka) is not initially expanded and some of the information in the window is not visible. Drag the bottom right-hand corner of the window to resize the window until the scroll bars disappear.

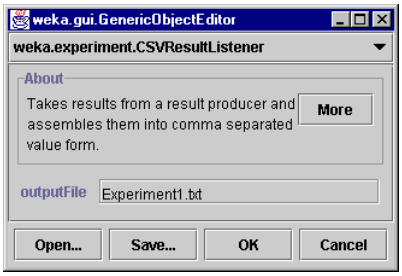


The output file parameter is near the bottom of the window, beside the text "outputFile". Click on this parameter to display a file selection window.

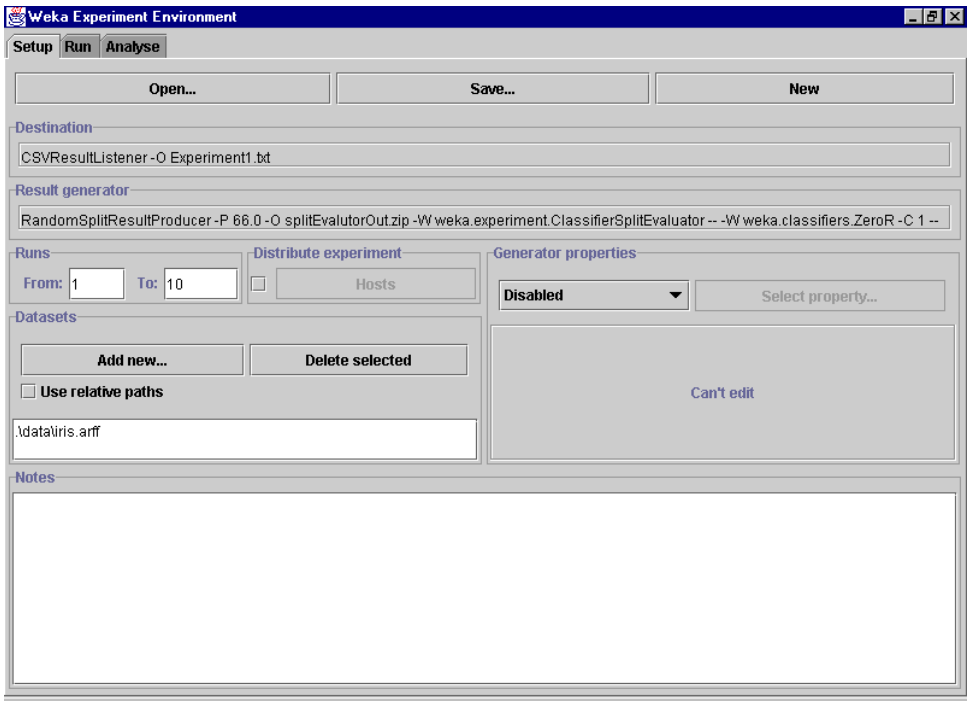




Type the name of the output file, click Select, and then click close (x). The file name is displayed in the outputFile panel. Click on OK to close the window.

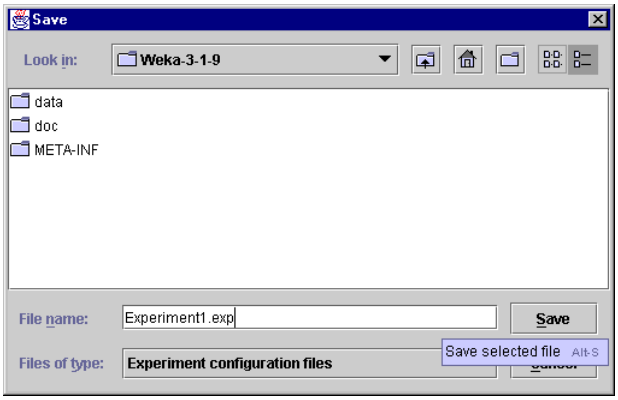


The dataset name is displayed in the Destination panel of the Setup window.



Saving the Experiment Definition

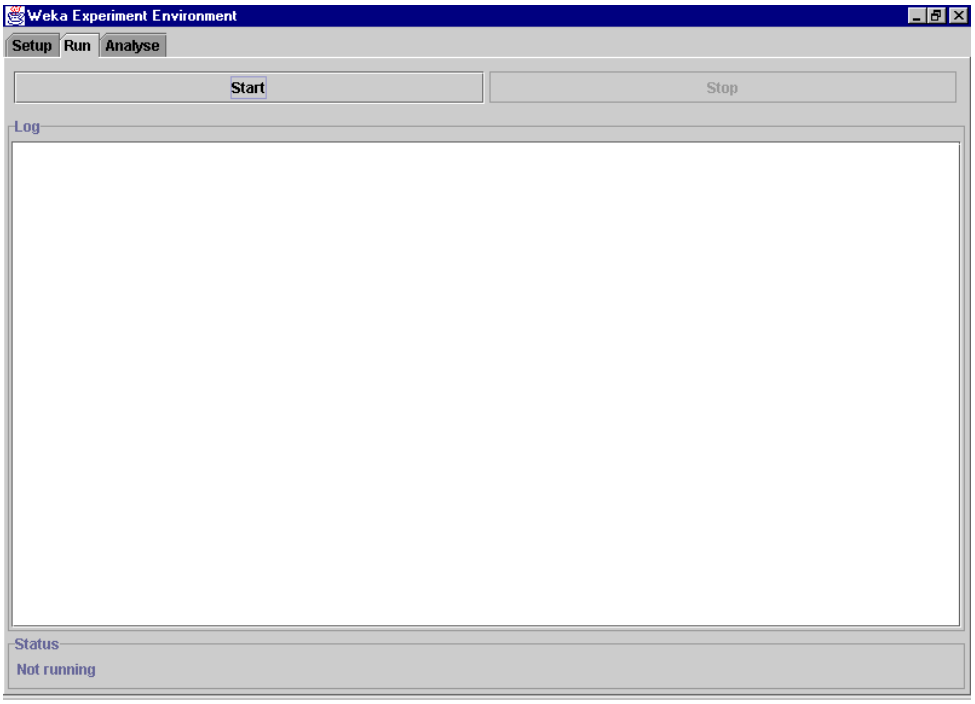
The experiment definition can be saved at any time. Select "Save …" at the top of the Setup window. Type the dataset name with the extension "exp" (or select the dataset name if the experiment definition dataset already exists).



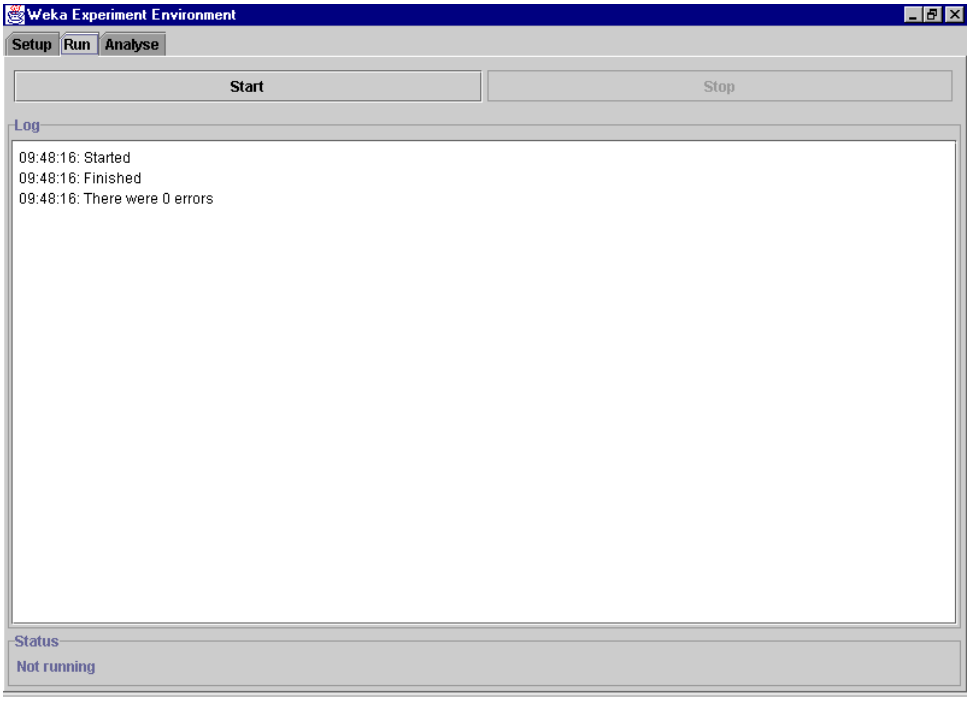
The experiment can be restored by selecting Open in the Setup window and then selecting Experiment1.exp in the dialog window.

Running an Experiment

To run the current experiment, click the Run tab at the top of the Experiment Environment window. The current experiment performs 10 randomized train and test runs on the Iris dataset, using 66% of the patterns for training and 34% for testing, and using the ZeroR scheme.



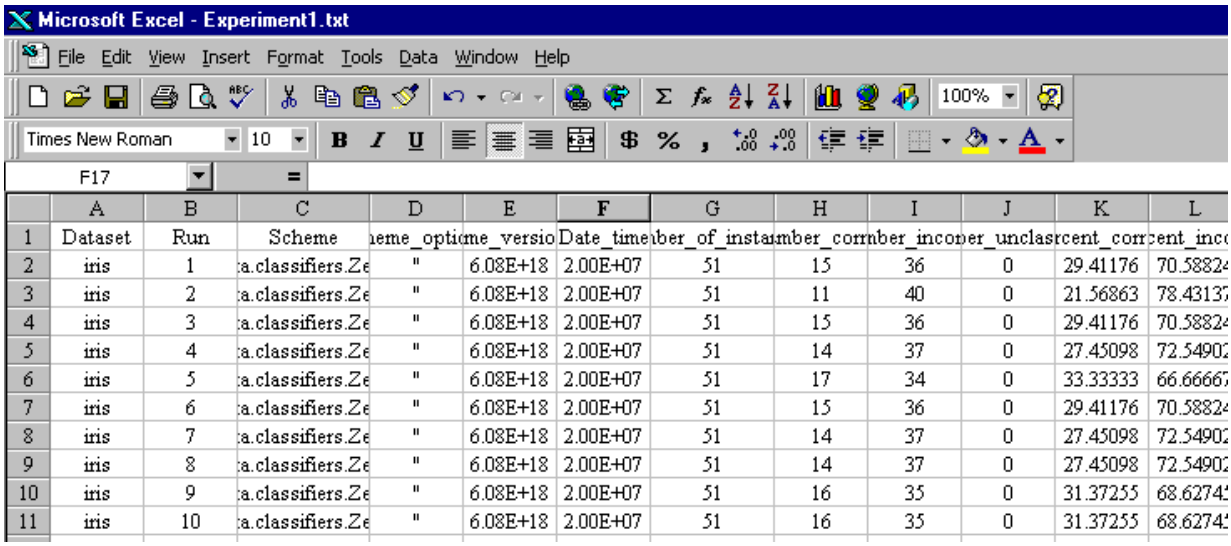
Click Start to run the experiment.



If the experiment was defined correctly, the 3 messages shown above will be displayed in the Log panel. The results of the experiment are saved to the dataset Experiment1.txt. The first two lines in this dataset are shown below.

```
Dataset,Run,Scheme,Scheme_options,Scheme_version_ID,Date_time,Number_of_instances,Number
_correct,Number_incorrect,Number_unclassified,Percent_correct,Percent_incorrect,Percent_
unclassified,Mean_absolute_error,Root_mean_squared_error,Relative_absolute_error,Root_re
lative_squared_error,SF_prior_entropy,SF_scheme_entropy,SF_entropy_gain,SF_mean_prior_en
tropy,SF_mean_scheme_entropy,SF_mean_entropy_gain,KB_information,KB_mean_information,KB_
relative_information,True_positive_rate,Num_true_positives,False_positive_rate,Num_false
_positives,True_negative_rate,Num_true_negatives,False_negative_rate,Num_false_negatives
,IR_precision,IR_recall,F_measure,Summary

iris,1,weka.classifiers.ZeroR,'',6077547173920530258,2.00102021558E7,51.0,15.0,36.0,0.0,
29.41176470588235,70.58823529411765,0.0,0.4462386261694216,0.47377732045597576,100.0,100
.0,81.5923629400546,81.5923629400546,0.0,1.5998502537265609,1.5998502537265609,0.0,0.0,0
.0,0.0,0.0,0.0,0.0,0.0,1.0,31.0,1.0,20.0,0.0,0.0,0.0,0.0,?
iris,2,weka.classifiers.ZeroR,'',6077547173920530258,2.00102021558E7,51.0,11.0,40.0,0.0,
21.568627450980394,78.43137254901961,0.0,0.4513648596693575,0.48049218646442554,100.0,10
0.0,83.58463098131035,83.58463098131035,0.0,1.6389143329668696,1.6389143329668696,0.0,0.
0,0.0,0.0,0.0,0.0,0.0,0.0,0.0,1.0,31.0,1.0,20.0,0.0,0.0,0.0,0.0,?
```

The results are generated in comma-separated value (CSV) form and can be loaded into a spreadsheet for analysis. (Note that the results generated by the Weka system are in Unix format with only a linefeed at the end of each line. An MS Windows program that can convert from Unix format to DOS/Windows format is Textpad at www.textpad.com. This program contains a variety of other useful features, including find and replace with regular expressions.)



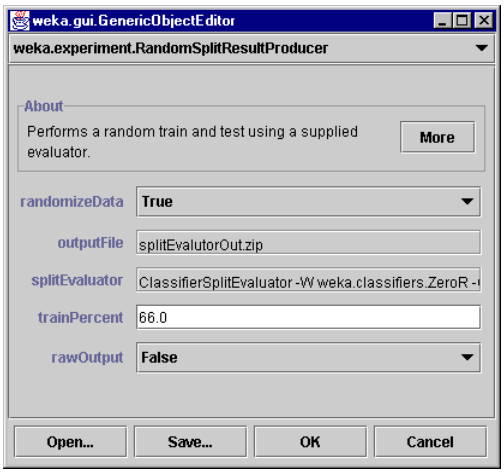
	A	B	C	D	E	F	G	H	I	J	K	L
1	Dataset	Run	Scheme	eme_optic	me_versio	Date_time	ber_of_insta	mber_corr	mber_inco	er_unclas	rcent_cor	cent_inc
2	iris	1	a.classifiers.Ze	"	6.08E+18	2.00E+07	51	15	36	0	29.41176	70.5882
3	iris	2	a.classifiers.Ze	"	6.08E+18	2.00E+07	51	11	40	0	21.56863	78.4313
4	iris	3	a.classifiers.Ze	"	6.08E+18	2.00E+07	51	15	36	0	29.41176	70.5882
5	iris	4	a.classifiers.Ze	"	6.08E+18	2.00E+07	51	14	37	0	27.45098	72.5490
6	iris	5	a.classifiers.Ze	"	6.08E+18	2.00E+07	51	17	34	0	33.33333	66.6666
7	iris	6	a.classifiers.Ze	"	6.08E+18	2.00E+07	51	15	36	0	29.41176	70.5882
8	iris	7	a.classifiers.Ze	"	6.08E+18	2.00E+07	51	14	37	0	27.45098	72.5490
9	iris	8	a.classifiers.Ze	"	6.08E+18	2.00E+07	51	14	37	0	27.45098	72.5490
10	iris	9	a.classifiers.Ze	"	6.08E+18	2.00E+07	51	16	35	0	31.37255	68.6274
11	iris	10	a.classifiers.Ze	"	6.08E+18	2.00E+07	51	16	35	0	31.37255	68.6274

Line 1 is a header that identifies the columns of output. Each subsequent line defines one train and test run. Line 2 of the spreadsheet indicates that for the first run of the experiment, the Iris dataset was used with the ZeroR scheme and that 51 instances were tested by the scheme: 15 instances were classified correctly, 36 instances were classified incorrectly, and 0 instances could not be classified.

As will be discussed later, experiments can also be analysed directly by the Weka system and do not need to be processed in a spreadsheet.

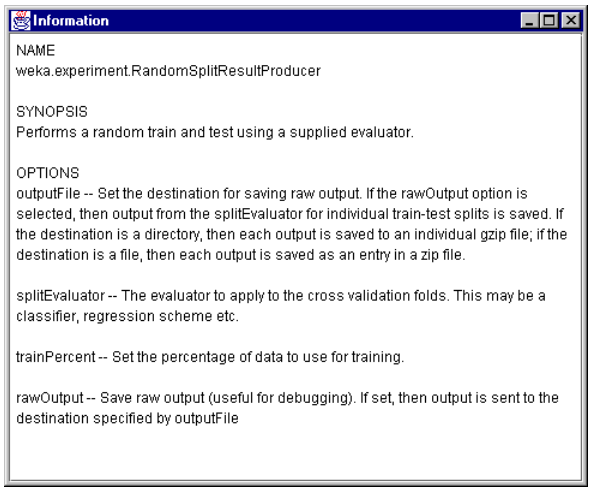
Changing the Experiment Parameters

The parameters of an experiment can be changed by clicking on the ResultGenerator panel. Drag the corner of the window to expand the window until all parameters are visible.

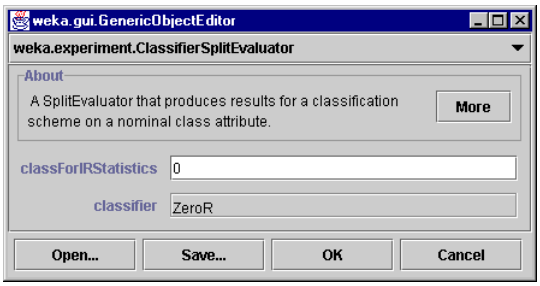


The RandomSplitResultProducer performs repeated train/test runs. The number of patterns (expressed as a percentage) used for training is given in the trainPercent box. (The number of runs is specified in the Runs parameter in the Setup window.)

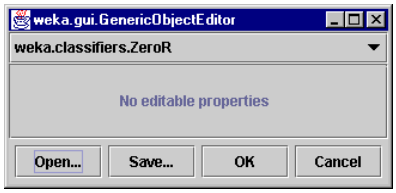
A small help file can be displayed by clicking More in the About panel.



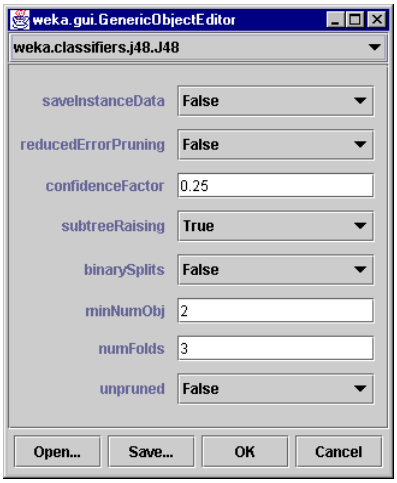
Click on the splitEvaluator entry to display the SplitEvaluator properties.



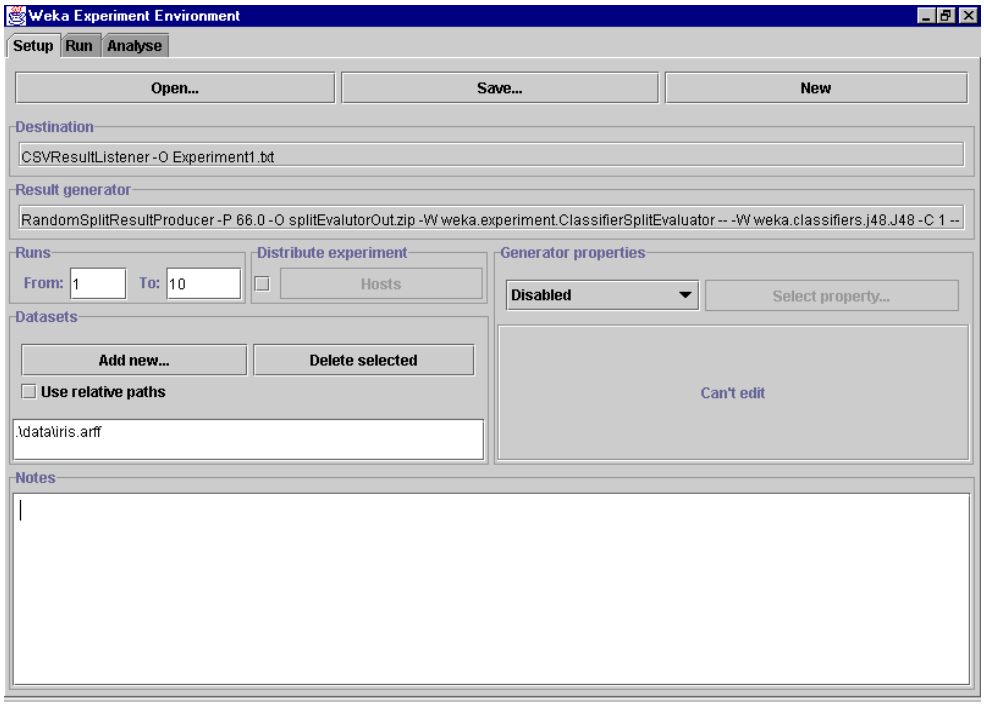
Click on the classifier entry (ZeroR) to display the scheme properties.



This scheme has no modifiable properties but most other schemes do have properties that can be modified by the user. Click on the drop-down list for the scheme (ZeroR) to select a different scheme. The window below shows the parameters available for the j48.J48 decision-tree scheme. If desired, modify the parameters and then click OK to close the window.

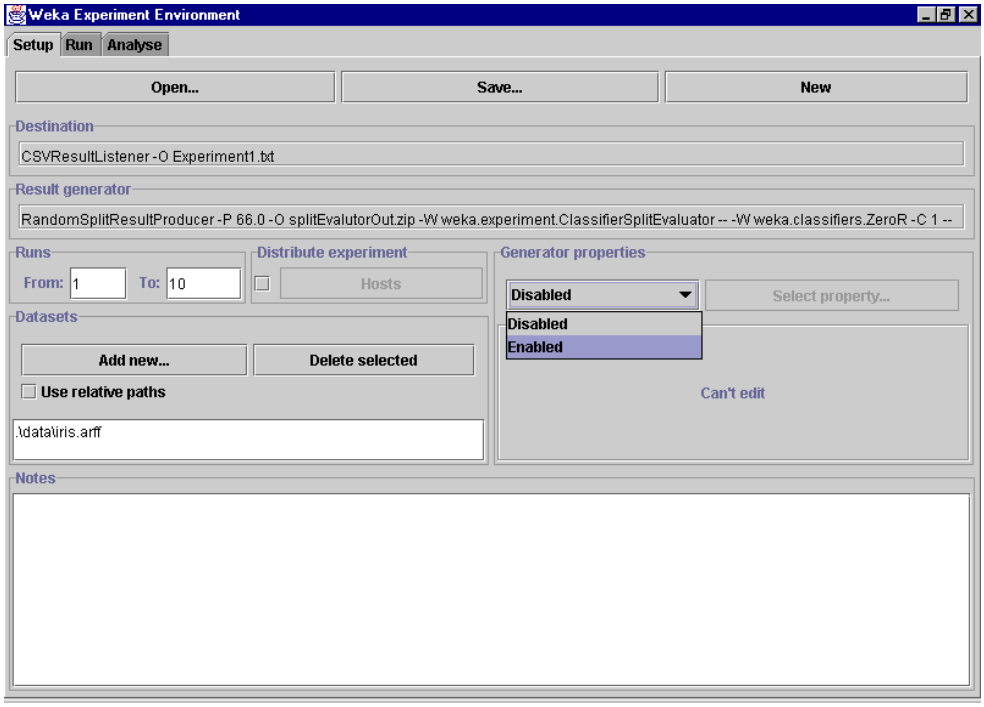


The name of the new scheme is displayed in the Result generator panel.

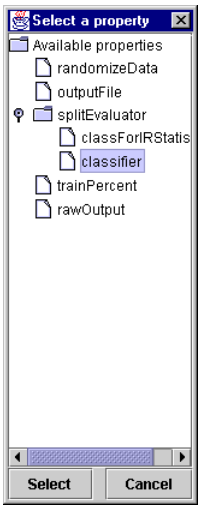


Adding Additional Schemes

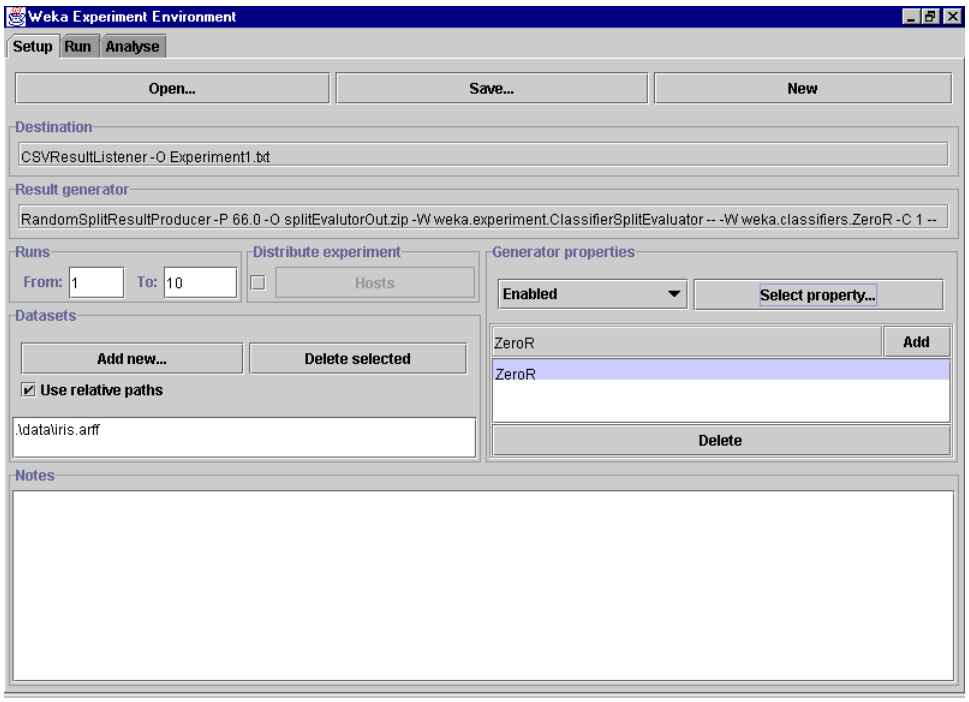
Additional Schemes can be added in the Generator properties panel. To begin, change the drop-down list entry from Disabled to Enabled in the Generator properties panel.



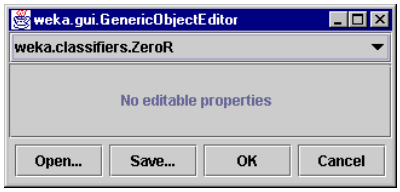
Click Select property and expand splitEvaluator so that the classifier entry is visible in the property list; click Select.



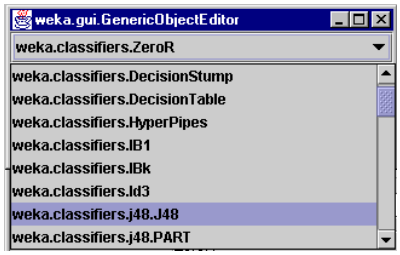
The scheme name is displayed in the Generator properties panel.



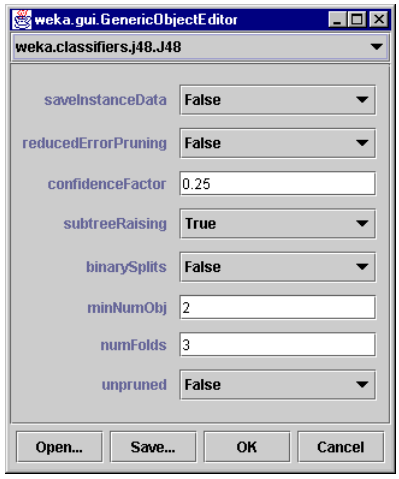
To add another scheme, click on the scheme name to display the scheme properties window.



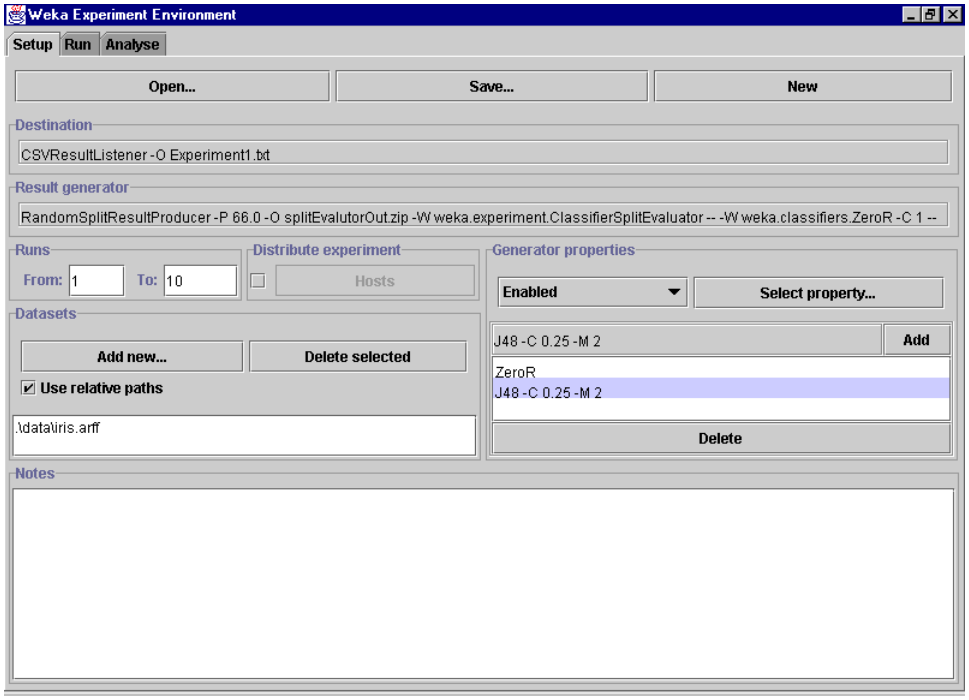
Click on the drop-down list to select another scheme.



To change to a decision-tree scheme, select j48.J48.



The new scheme is added to the Generator properties panel. Click Add to add the new scheme.



Now when the experiment is run, results are generated for both schemes.

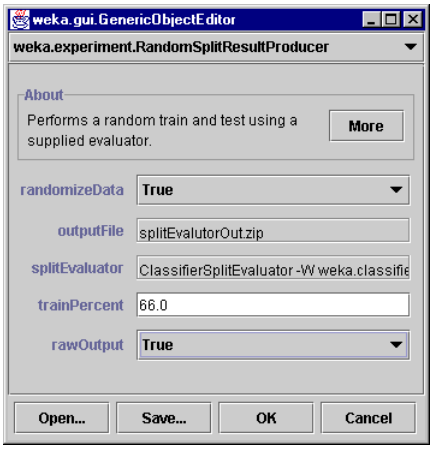
To add additional schemes, repeat this process. To remove a scheme, select the scheme by clicking on it and then click Delete.

Adding Additional Datasets

The scheme(s) may be run on any number of datasets at a time. Additional datasets are added by clicking "Add new …" in the Datasets panel. Datasets are deleted from the experiment by selecting the dataset and then clicking Delete Selected.

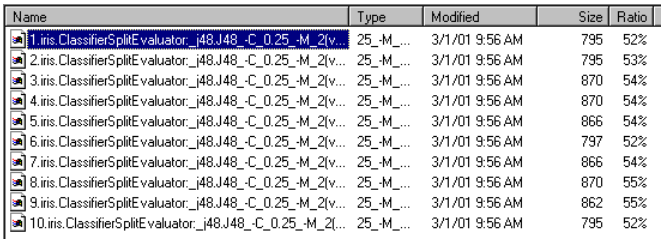
Raw Output:

The output generated by a scheme can be saved to a file and then examined at a later time. Open the Result Producer window by clicking on the Result Generator panel in the Setup window.



Click on rawOutput and select the True entry from the drop-down list. By default, the output is sent to the file splitEvaluatorOut.zip. The output file can be changed by clicking on the outputFile panel in the window.

Now when the experiment is run, the result of each processing run is archived, as shown below.

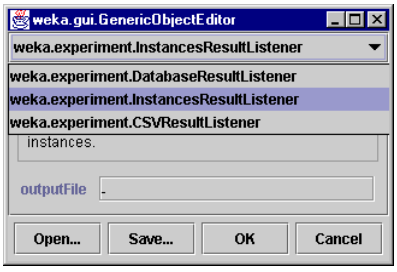


The contents of the first run are:

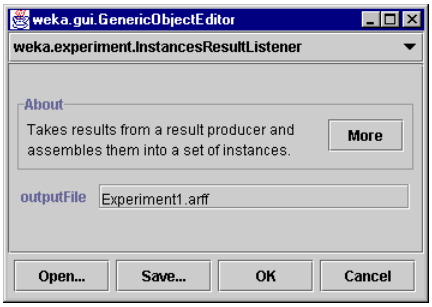
```
ClassifierSplitEvaluator: weka.classifiers.j48.J48 -C 0.25 -M 2(version -
      8930283471623955722)Classifier model:

J48 pruned tree
------------------
petalwidth <= 0.4: Iris-setosa (30.0)
petalwidth > 0.4
|   petalwidth <= 1.7: Iris-versicolor (37.0/3.0)
|   petalwidth > 1.7: Iris-virginica (32.0/1.0)

Number of Leaves  :     3
Size of the tree :      5

Correctly Classified Instances          47               92.1569 %
Incorrectly Classified Instances         4                7.8431 %
Mean absolute error                      0.0718
Root mean squared error                  0.2185
Relative absolute error                 16.0867 %
Root relative squared error             46.1175 %
Total Number of Instances               51
measureTreeSize : 5.0
measureNumLeaves : 3.0
measureNumRules : 3.0
```

Instances Result Producer

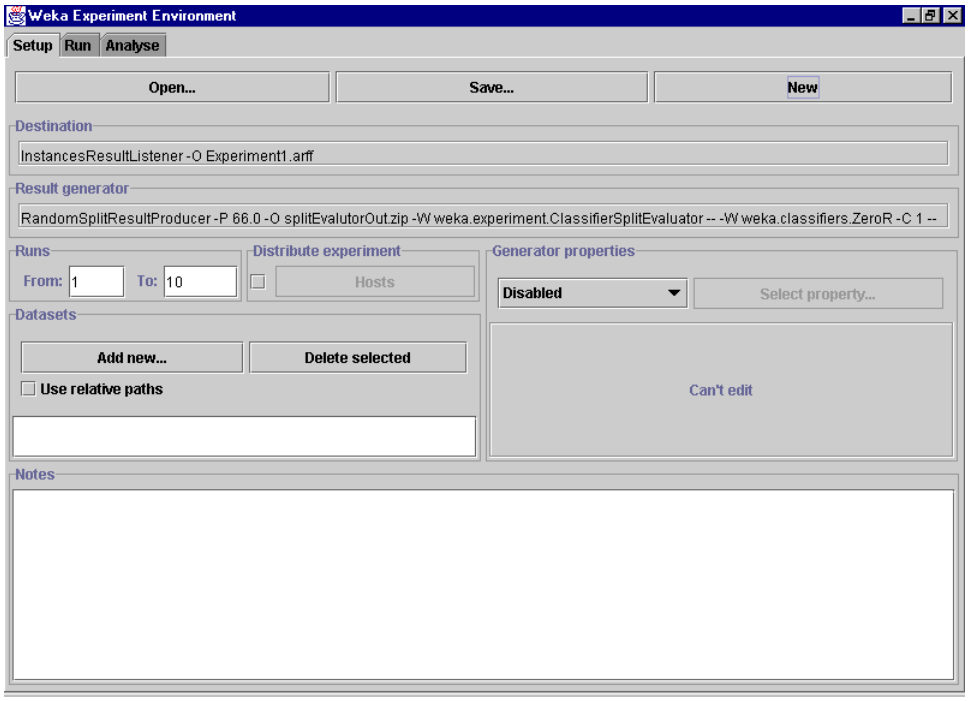
In addition to sending results of an experiment to a CSV Result Listener, results can also be sent to an Instances Result Listener and then analysed by the Weka Experiment Analyser. Click on the result listener portion of the Destination panel and then select Instances Result Listener.



Then select the output dataset. The dataset extension should be “arff”.



The new parameters are now displayed in the Destination panel.



When this experiment is run, results are generated in "arff" format (as illustrated below).

```
@relation InstanceResultListener
@attribute Key_Dataset {iris}
@attribute Key_Run {1,2,3,4,5,6,7,8,9,10}
@attribute Key_Scheme {weka.classifiers.ZeroR}
@attribute Key_Scheme_options {''}
@attribute Key_Scheme_version_ID {6077547173920530258}
@attribute Date_time numeric
@attribute Number_of_instances numeric
@attribute Number_correct numeric
@attribute Number_incorrect numeric
@attribute Number_unclassified numeric
@attribute Percent_correct numeric
```

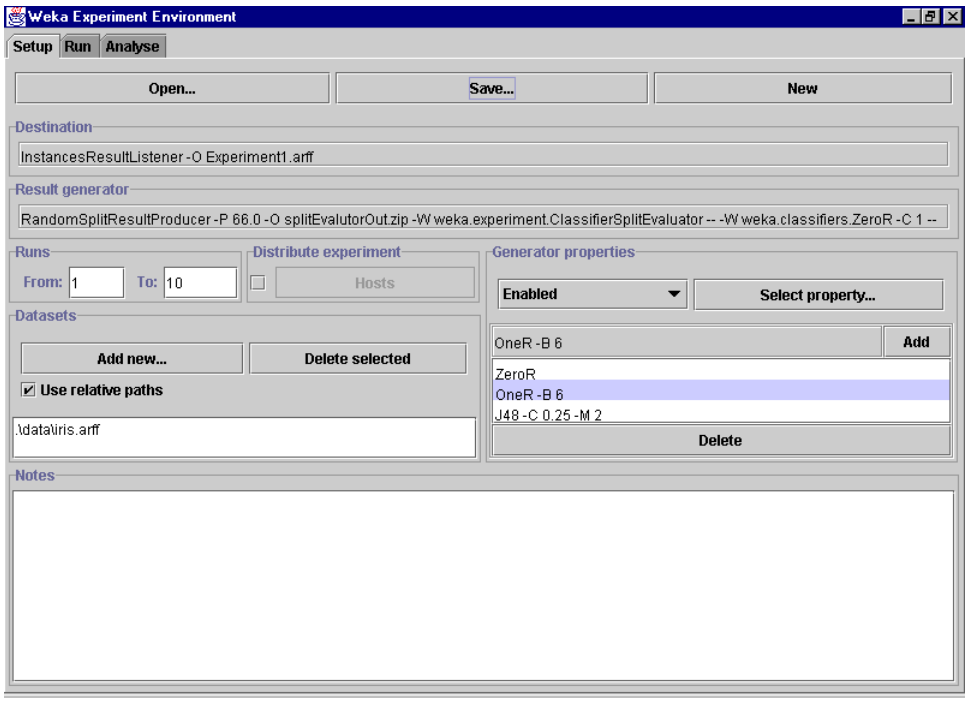
•••

```
@data
iris,1,weka.classifiers.ZeroR,'',6077547173920530258,20010205.1546,51,15,36,0,29.411765,
70.588235,0,0.446239,0.473777,100,100,81.592363,81.592363,0,1.59985,1.59985,0,0,0,0,0,0,
0,0,1,31,1,20,0,0,0,?
iris,2,weka.classifiers.ZeroR,'',6077547173920530258,20010205.1546,51,11,40,0,21.568627,
78.431373,0,0.451365,0.480492,100,100,83.584631,83.584631,0,1.638914,1.638914,0,0,0,0,0,
0,0,0,1,31,1,20,0,0,0,?
iris,3,weka.classifiers.ZeroR,'',6077547173920530258,20010205.1546,51,15,36,0,29.411765,
70.588235,0,0.446239,0.473777,100,100,81.592363,81.592363,0,1.59985,1.59985,0,0,0,0,0,0,
0,0,1,35,1,16,0,0,0,?
```

•••

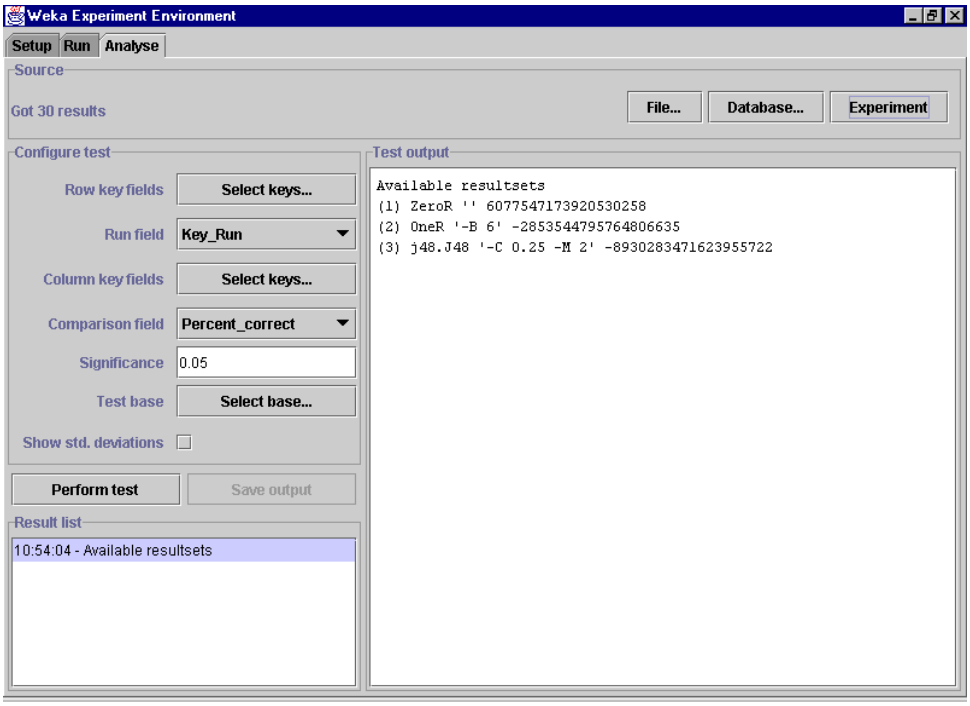
Experiment Analyser

Weka includes an experiment analyzer that can be used to analyse the results of experiments that were sent to an Instances Result Listener. The experiment shown below uses 3 schemes, ZeroR, OneR, and j48.J48, to classify the Iris data in an experiment using 10 train and test runs, with 66% of the data used for training and 34% used for testing.



After the experiment setup is complete, run the experiment. Then, to analyse the results, select the Analyse tab at the top of the Experiment Environment window. Note that the results must be in arff format, as generated by the Instances Result Listener.

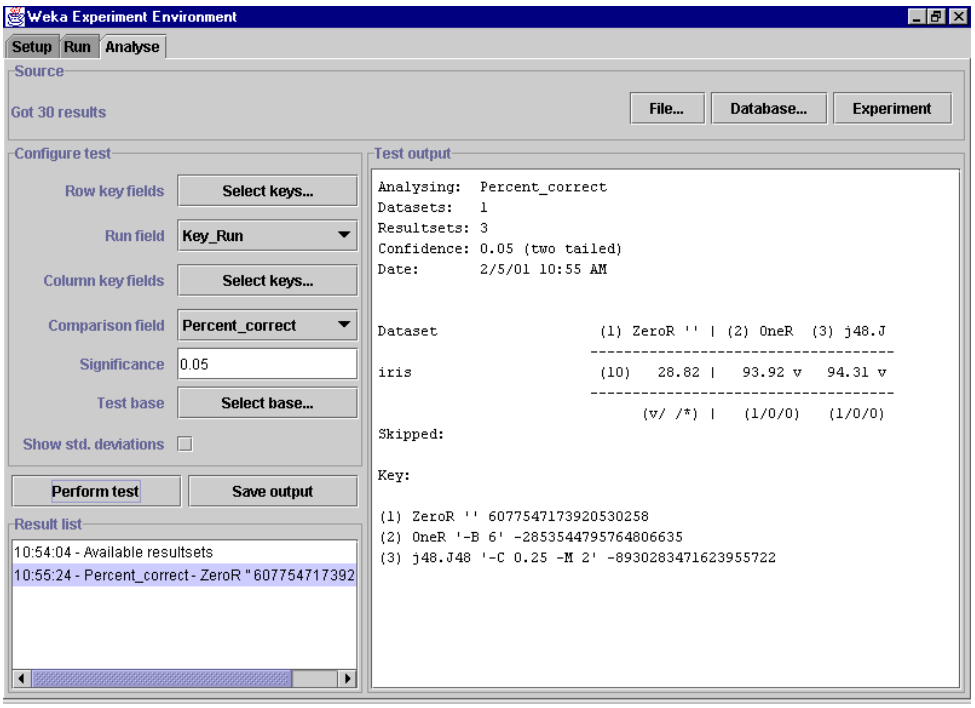
Click on Experiment to analyse the results of the current experiment.



The number of result lines available ("Got 30 results") is shown in the Source panel. This experiment consisted of 10 runs, for 3 schemes, for 1 dataset, for a total of 30 result lines.

Results can also be loaded from an earlier experiment file by clicking File and loading the appropriate .arff results file. Similarly, results sent to a database (using the Database Result Listener) can be loaded from the database.

Select the Percent_correct attribute from the Comparison field and click Perform test to generate a comparison of the 3 schemes.

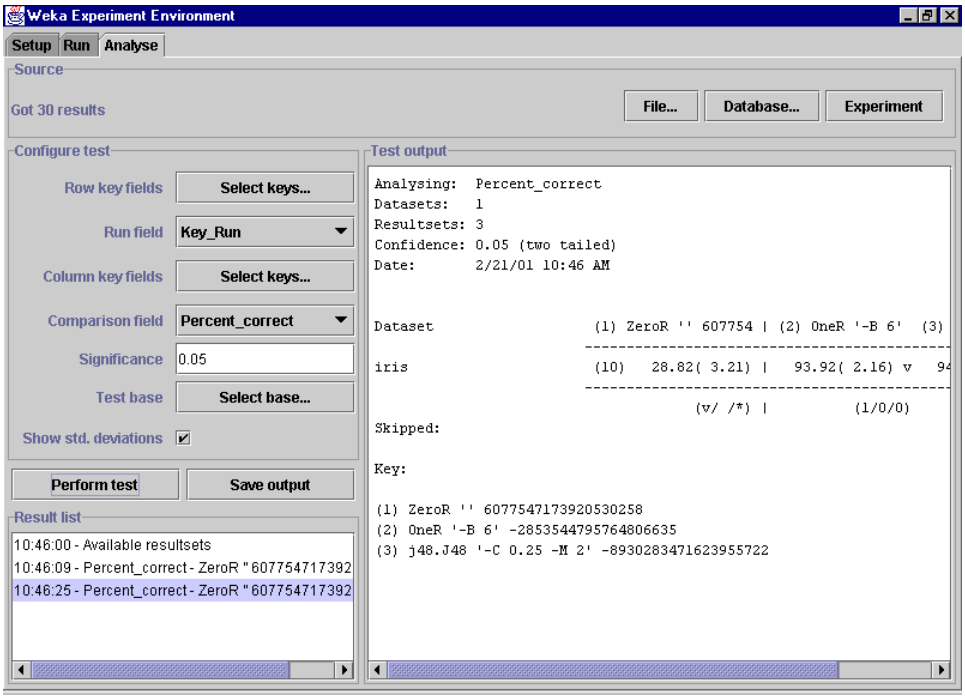


The schemes used in the experiment are shown in the columns and the datasets used are shown in the rows.

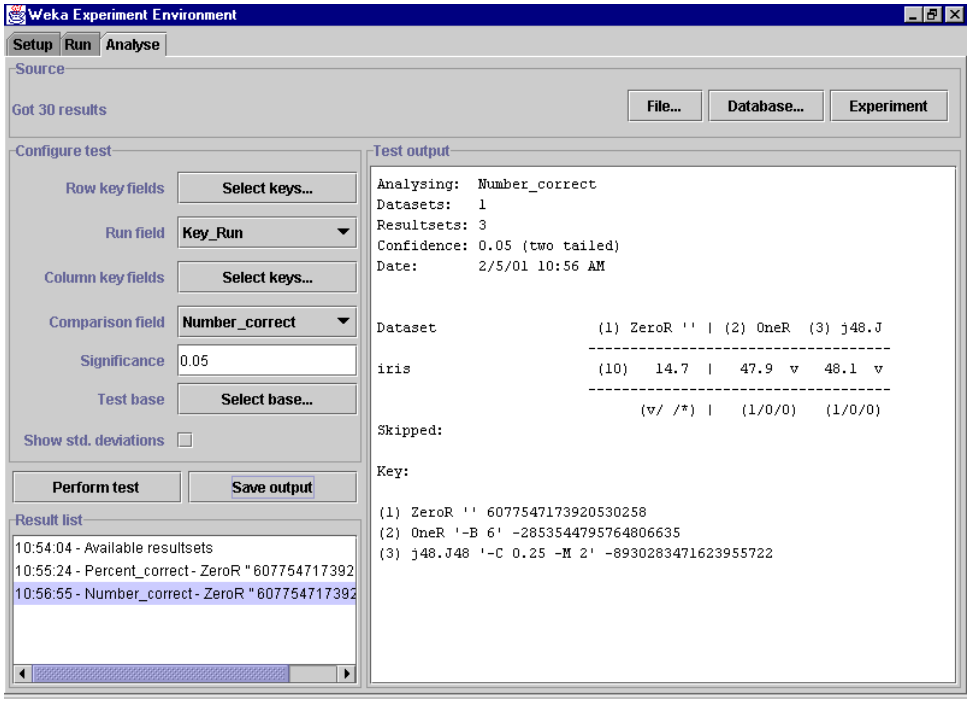
The percentage correct for each of the 3 schemes is shown in each dataset row: 28.82% for ZeroR, 93.92% for OneR, and 94.31% for j48.J48. The annotation "v" or "*" indicates that a specific result is statistically better (v) or worse (*) than the baseline scheme (in this case, ZeroR) at the significance level specified (currently 0.05). The results of both OneR and j48.J48 are statistically better than the baseline established by ZeroR. At the bottom of each column after the first column is a count (xx/ yy/ zz) of the number of times that the scheme was better than (xx), the same as (yy), or worse than (zz) the baseline scheme on the datasets used in the experiment. In this example, there was only one dataset and OneR was better than ZeroR once and never equivalent to or worse than ZeroR (1/0/0); j48.J48 was also better than ZeroR on the dataset.

The value "(10)" at the beginning of the "iris" row defines the number of runs of the experiment.

The standard deviation of the attribute being evaluated can be generated by selecting the Show std. deviations check box.

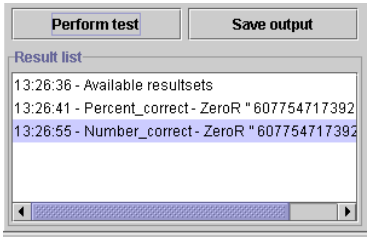


Selecting Number_correct as the comparison field and clicking Perform test generates the average number correct (out of a maximum of 51 test patterns – 34% of 150 patterns in the Iris dataset).

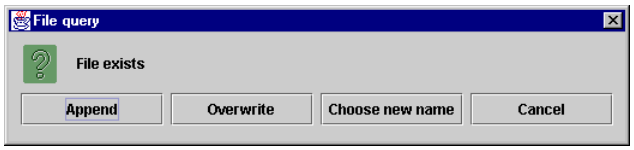


Saving the Results

The information displayed in the Test output panel is controlled by the currently-selected entry in the Result list panel. Clicking on an entry causes the results corresponding to that entry to be displayed.

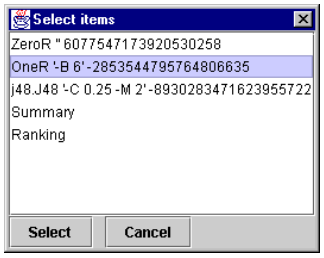


The results shown in the Test output panel can be saved to a file by clicking Save output. Only one set of results can be saved at a time but Weka permits the user to save all results to the same dataset by saving them one at a time and using the Append option instead of the Overwrite option for the second and subsequent saves.

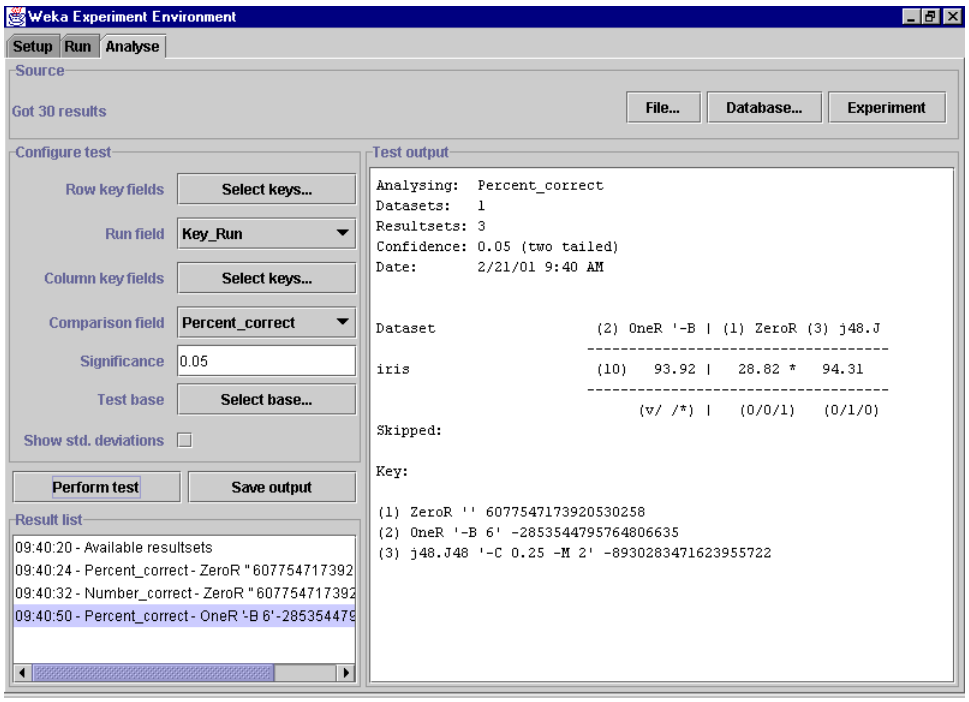


Changing the Baseline Scheme

The baseline scheme can be changed by clicking Select base… and then selecting the desired scheme. Selecting the OneR scheme causes the other schemes to be compared individually with the OneR scheme.



If the test is performed on the Percent_correct field with OneR as the base scheme, the system indicates that there is no statistical difference between the results for OneR and j48.J48. There is however a statistically significant difference between OneR and ZeroR.



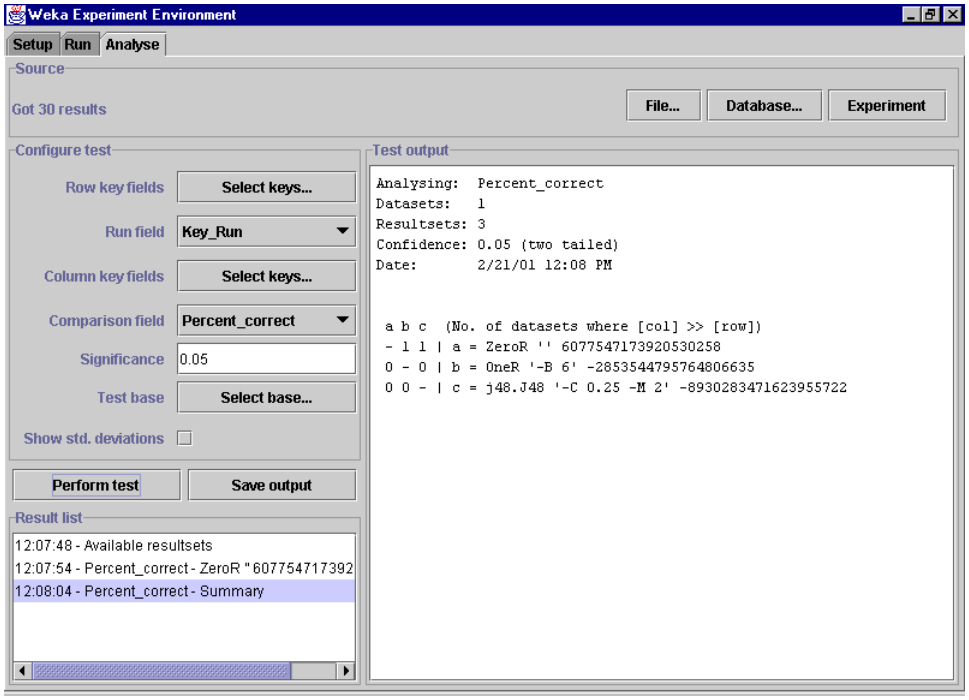
Statistical Significance

The term "statistical significance" used in the previous section refers to the result of a pair-wise comparison of schemes using a "t-test". For more information on the t-test, consult the Weka text (*Data Mining* by I. Witten and E. Frank) or an introductory statistics text. As the significance level is decreased, the confidence in the conclusion increases.

In the current experiment, there is not a statistically significant difference between the OneR and j48.J48 schemes.

Summary Test

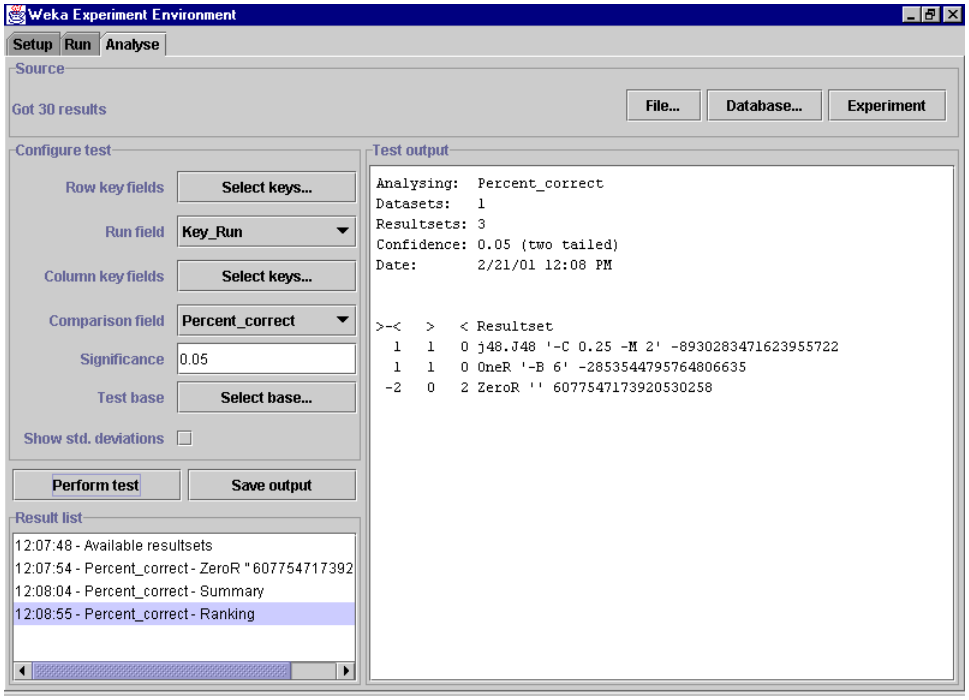
Selecting Summary from Test base and performing a test causes the following information to be generated.



In this experiment, the first row (- 1 1) indicates that column “b” (OneR) is better than row “a” (ZeroR) and that column “c” (j48.J48) is also better than row “a”. The remaining entries are 0 because there is no significant difference between OneR and j48.J48 on the dataset that was used in the experiment.

Ranking Test

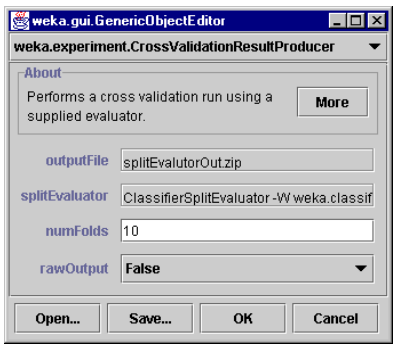
Selecting Ranking from Test base causes the following information to be generated.



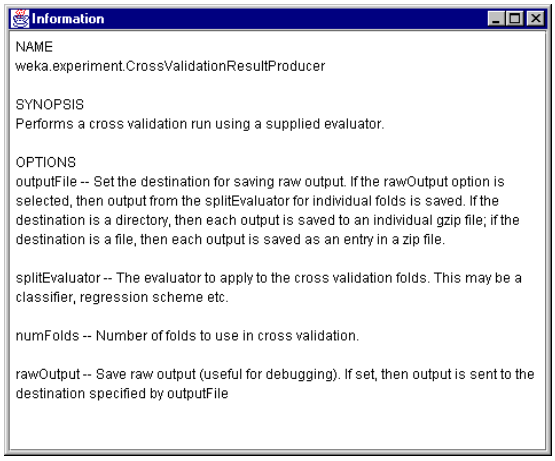
The ranking test ranks the schemes according to the total wins (“>”) and losses (“<”) against the other schemes. The first column (“>-<”) is the difference between the number of wins and the number of losses.

Cross-Validation Result Producer

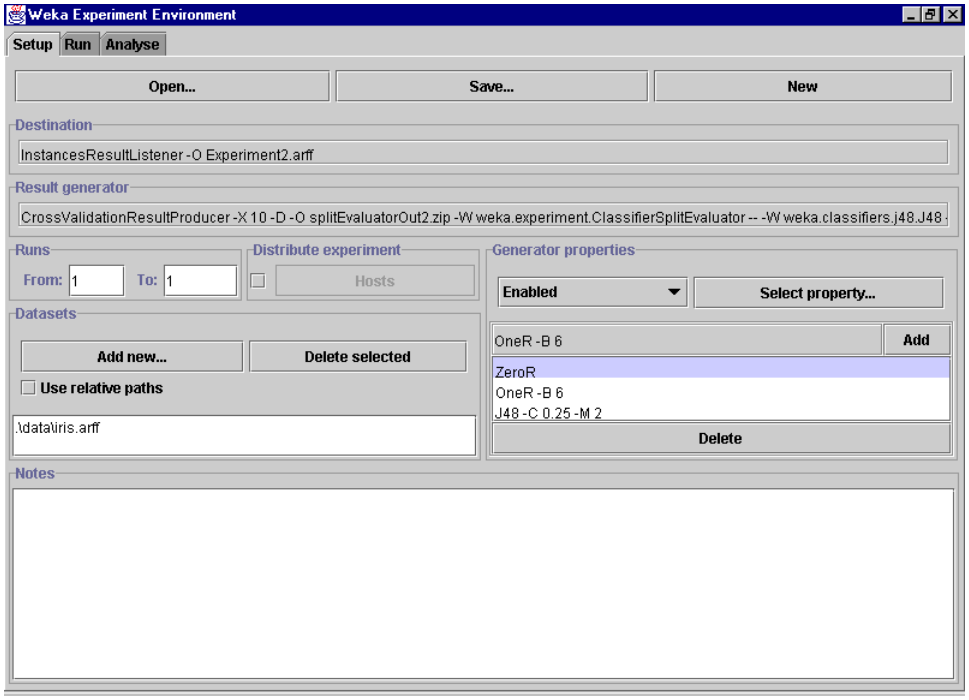
To change from random train and test experiments to cross-validation experiments, click on the Result generator entry. At the top of the window, click on the drop-down list and select CrossValidationResultProducer. The window now contains parameters specific to cross validation such as the number of partitions/folds. The experiment performs 10-fold cross-validation instead of train and test.



The result generator panel now indicates that cross validation will be performed. Click on More to generate a brief description of the cross-validation result producer.

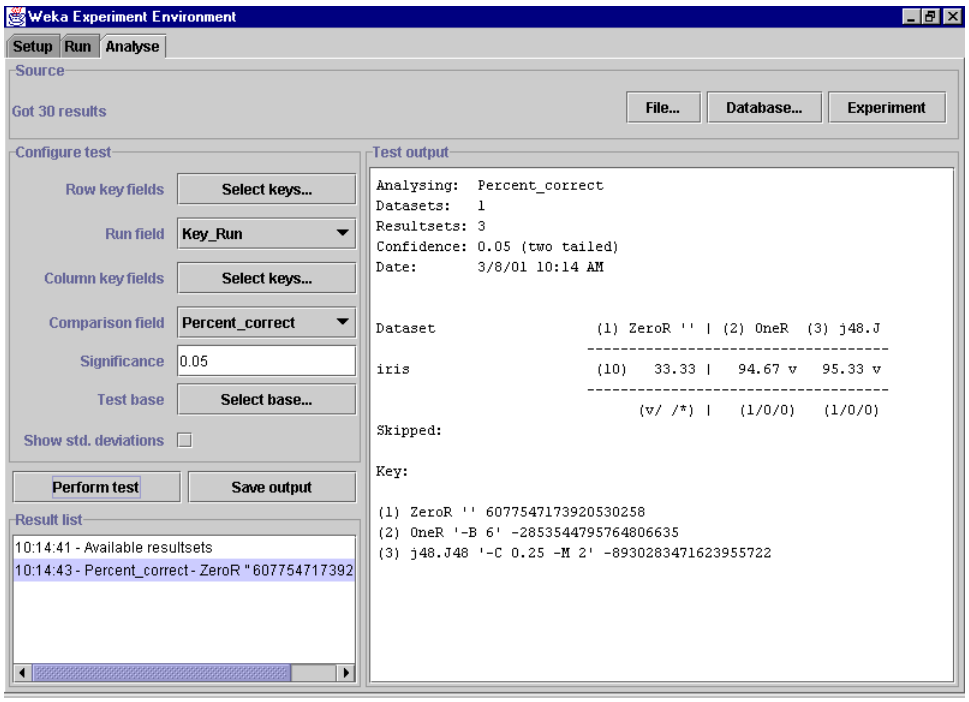


As with the Random Split Result Producer, multiple schemes can be run during cross validation by adding them to the Generator properties panel.



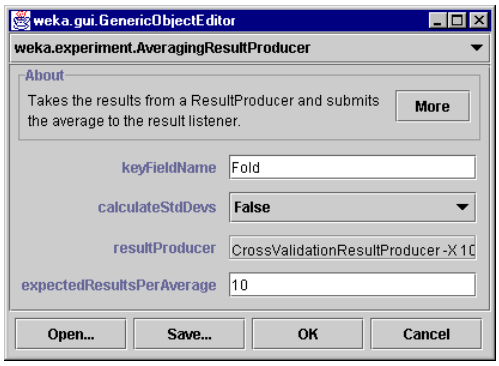
The number of runs is set to 1 in the Setup window.

When this experiment is analysed, the following results are generated. Note that there are 30 (1 run times 10 folds times 3 schemes) result lines processed.

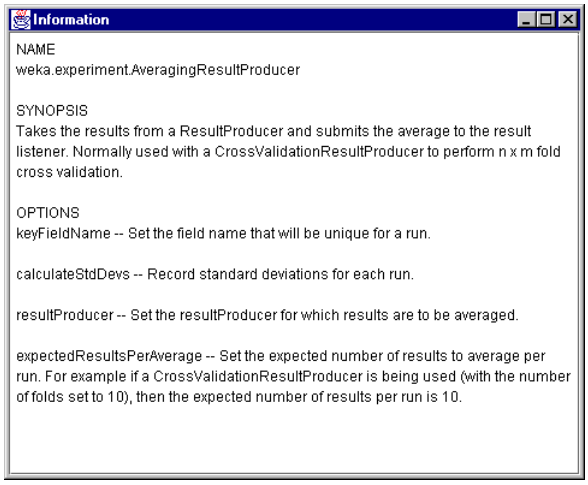


Averaging Result Producer:

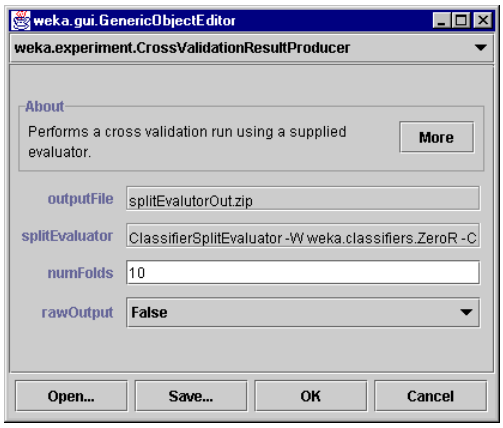
An alternative to the CrossValidation Result Producer is the Averaging Result Producer. This result producer takes the average of a set of runs (which are typically cross-validation runs). This result producer is identified by clicking the Result Generator panel and then selecting AveragingResultProducer from the drop-down list.



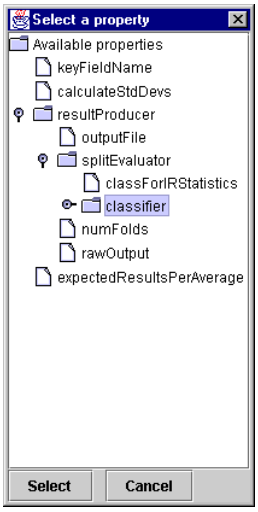
The associated help file is shown below.

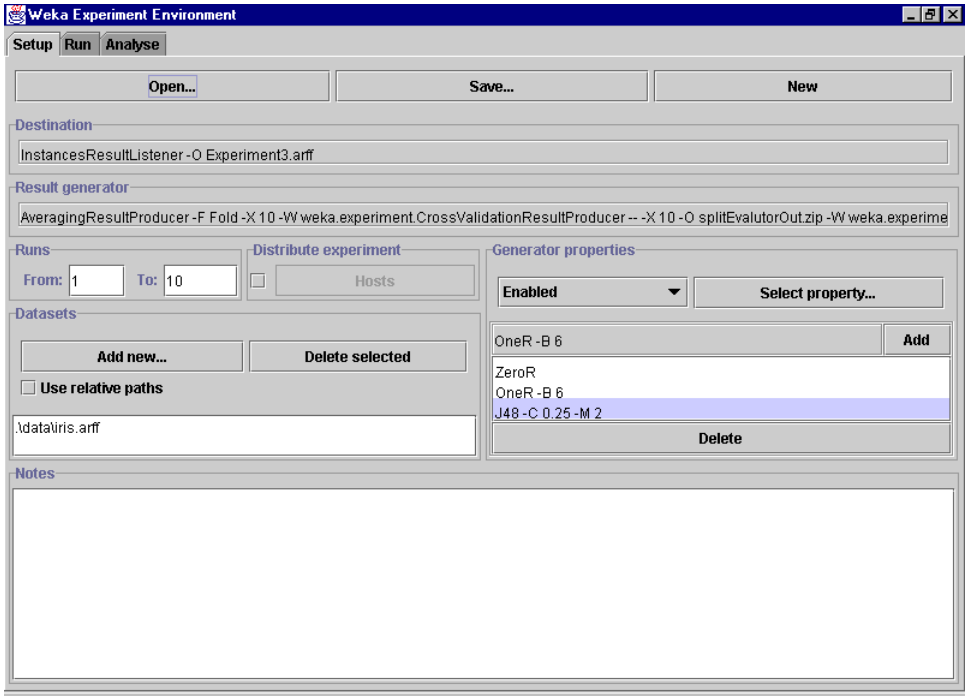


Clicking the resultProducer panel brings up the following window.

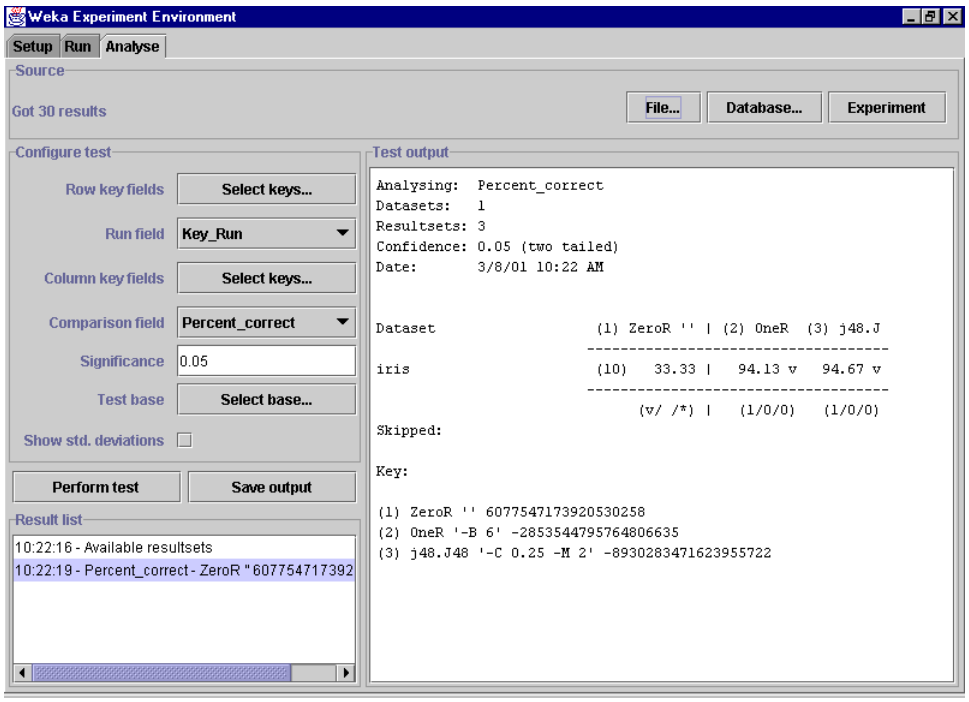


As with the other Result Producers, additional schemes can be defined. When the AveragingResultProducer is used, the classifier property is located deeper in the Generator properties hierarchy.





In this experiment, the ZeroR, OneR, and j48.J48 schemes are run 10 times with 10-fold cross validation. Each run of 10 cross-validation folds is then averaged, producing one result line for each run (instead of one result line for each fold as in the previous example using the cross-validation result producer) for a total of 30 result lines. If the raw output is saved, all 300 results are sent to the archive.



It should be noted that while the results generated by the averaging result producer are slightly worse than those generated by the cross-validation result producer, the standard deviations are significantly smaller with the averaging result producer (as can be seen below).

Cross-validation Results

```
Analysing:  Percent_correct
Datasets:   1
Resultsets: 3
Confidence: 0.05 (two tailed)
Date:       3/8/01 1:39 PM

Dataset                   (1) ZeroR '' 607754 | (2) OneR '-B 6'   (3) j48.J48 '-C
                          ------------------------------------------------------------
iris                      (10)   33.33( 0    ) |   94.67( 4.22) v   95.33( 4.5 ) v
                          ------------------------------------------------------------
                                        (v/ /*) |          (1/0/0)          (1/0/0)
```

Averaging Result Producer Results

```
Analysing:  Percent_correct
Datasets:   1
Resultsets: 3
Confidence: 0.05 (two tailed)
Date:       3/8/01 1:40 PM

Dataset                   (1) ZeroR '' 607754 | (2) OneR '-B 6'   (3) j48.J48 '-C
                          ------------------------------------------------------------
iris                      (10)   33.33( 0    ) |   94.13( 0.76) v   94.67( 0.7 ) v
                          ------------------------------------------------------------
                                        (v/ /*) |          (1/0/0)          (1/0/0)
```

Distributing Processing over Several Machines

For experiments that require significant computational resources, the processing can be distributed over a series of machines by defining the appropriate information in the Distribute experiment panel of the Setup window. The following information has been copied from the README_Experiment_Gui in the Weka folder.

"This is very much experimental (no pun intended). The Experimenter in Weka-3-1-9 includes the ability to split an experiment up and distribute it to multiple hosts. This works best when all results are being sent to a central data base, although you could have each host save its results to a distinct arff file and then merge the files afterwards.

"Distributed experiments have been tested using InstantDB (with the RMI bridge) and MySQL under Linux.

"Each host *must* have Java and Weka installed, access to whatever data sets you are using, and an experiment server running (weka.experiment.RemoteEngine).

"If results are being sent to a central data base, then the appropriate JDBC data base drivers must also be installed on each host and be listed in a DatabaseUtils.props (this file can be found in the experiment directory in either the weka.jar or weka-src.jar file) file which is accessable to the RemoteEngine running on that host.

"To start a RemoteEngine experiment server on a host--- first extract the remote.policy file from either weka.jar or weka-src.jar (its in the experiment subdirectory) and place this somewhere accessable. Then type:

java -Djava.security.policy=path to remote.policy/remote.policy weka.experiment.RemoteEngine (Java 1.2; for Java 1.1 leave out the "-Djava.security..." part))

"If all goes well there should be a message saying that the RemoteEngine has been bound in the RMI registry. Repeat this process on all hosts that you want to use.

"The SetUp panel of the Experimenter works exactly as before, but there is now a small panel next to the Runs panel which controls whether an experiment will be distributed or not. By default, this panel is inactive indicating that the experiment is a default (single machine) experiment. Clicking the checkbox will enable a remote (distributed) experiment and activates the "Hosts" button. Clicking the Hosts button will popup a window into which you can enter the names of the machines that you want to distribute the experiment to. Enter fully qualified names here, e.g. blackbird.cs.waikato.ac.nz.

"Once host names have been entered configure the rest of the experiment as you would normally. When you go to the Run panel and start the experiment progress on sub-experiments running on the different hosts will be displayed along with any error messages.

"Remote experiments work by making each run into a separate sub-experiment that is then sent by RMI to a remote host for execution. So for a standard 10 fold cross-validation style of experiment the maximum practical number of remote hosts is 10 (i.e. One for each sub-experiment); you can of course have fewer than 10 remote hosts."

Experimenter Parameters

The following information describes the parameters of the Experimenter.

```
> java weka.experiment.Experiment -h

Usage:
-l <exp file>
       Load experiment from file (default use cli options)
-s <exp file>
       Save experiment to file after setting other options
       (default don't save)
-r
       Run experiment (default don't run)

-L <num>
       The lower run number to start the experiment from.
       (default 1)
-U <num>
       The upper run number to end the experiment at (inclusive).
       (default 10)
-T <arff file>
       The dataset to run the experiment on.
       (required, may be specified multiple times)
-P <class name>
       The full class name of a ResultProducer (required).
       eg: weka.experiment.RandomSplitResultProducer
-D <class name>
       The full class name of a ResultListener (required).
       eg: weka.experiment.CSVResultListener
-N <string>
       A string containing any notes about the experiment.
       (default none)

Options specific to result producer weka.experiment.RandomSplitResultProducer:

-P <percent>
       The percentage of instances to use for training.
       (default 66)
-D
Save raw split evaluator output.
-O <file/directory name/path>
       The filename where raw output will be stored.
       If a directory name is specified then then individual
       outputs will be gzipped, otherwise all output will be
       zipped to the named file. Use in conjuction with -D. (default splitEvalutorOut.zip)
-W <class name>
       The full class name of a SplitEvaluator.
       eg: weka.experiment.ClassifierSplitEvaluator
-R
Set when data is not to be randomized.

Options specific to split evaluator weka.experiment.ClassifierSplitEvaluator:

-W <class name>
       The full class name of the classifier.
       eg: weka.classifiers.NaiveBayes
-C <index>
       The index of the class for which IR statistics
       are to be output. (default 1)
Required: -T <arff file name>
```