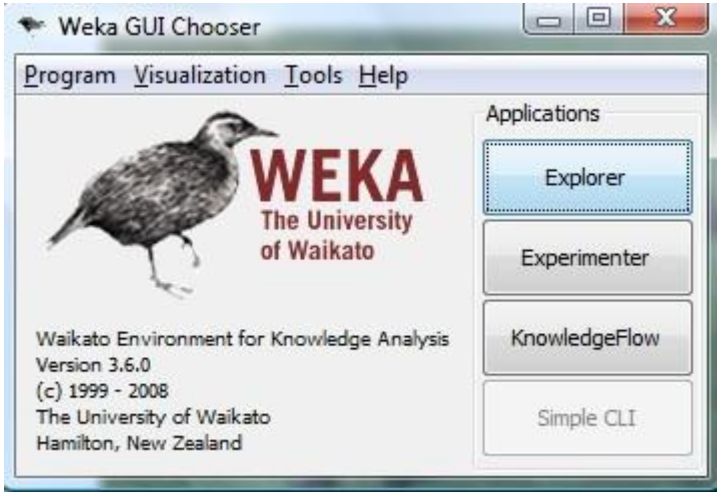
Introduction to Weka

WEKA
The University
of Waikato



Waikato Environment for Knowledge Analysis
Version 3.6.0
(c) 1999 - 2008
The University of Waikato
Hamilton, New Zealand

WEKA:: Introduction

A collection of open source ML algorithms

- pre-processing
- classifiers
- clustering
- association rule

Created by researchers at the University of Waikato in New Zealand

Java based

WEKA:: Installation

Download software from http://www.cs.waikato.ac.nz/ml/weka/

If you are interested in modifying/extending weka there is a developer version that includes the source code

Set the weka environment variable for java

```
setenv WEKAHOME /usr/local/weka/weka-3-0-2
setenv CLASSPATH $WEKAHOME/weka.jar:$CLASSPATH
```

Download some ML data from http://mlearn.ics.uci.edu/MLRepository.html

WEKA:: Introduction .contd

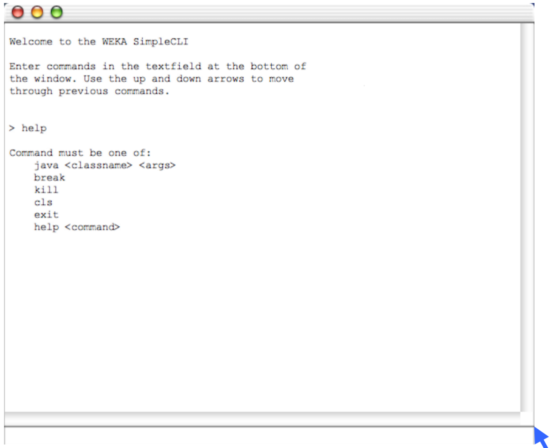
Routines are implemented as classes and logically arranged in packages

Comes with an extensive GUI interface

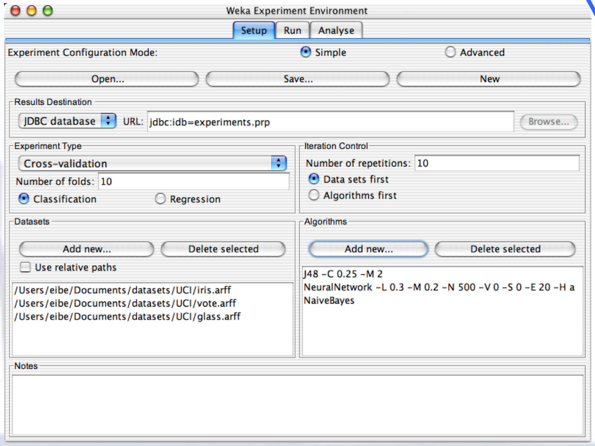
Weka routines can be used stand alone via the command line

```
Eg. java weka.classifiers.j48.J48 -t $WEKAHOME/data/iris.arff
```

WEKA:: Interface



```
Welcome to the WEKA SimpleCLI

Enter commands in the textfield at the bottom of
the window. Use the up and down arrows to move
through previous commands.


> help

Command must be one of:
    java <classname> <args>
    break
    kill
    cls
    exit
    help <command>
```



Experiment Configuration Mode: ● Simple ○ Advanced

Results Destination

JDBC database URL: jdbc:idb=experiments.prp

Experiment Type

Cross-validation

Number of folds: 10

● Classification ○ Regression

Iteration Control

Number of repetitions: 10

● Data sets first

○ Algorithms first

Datasets

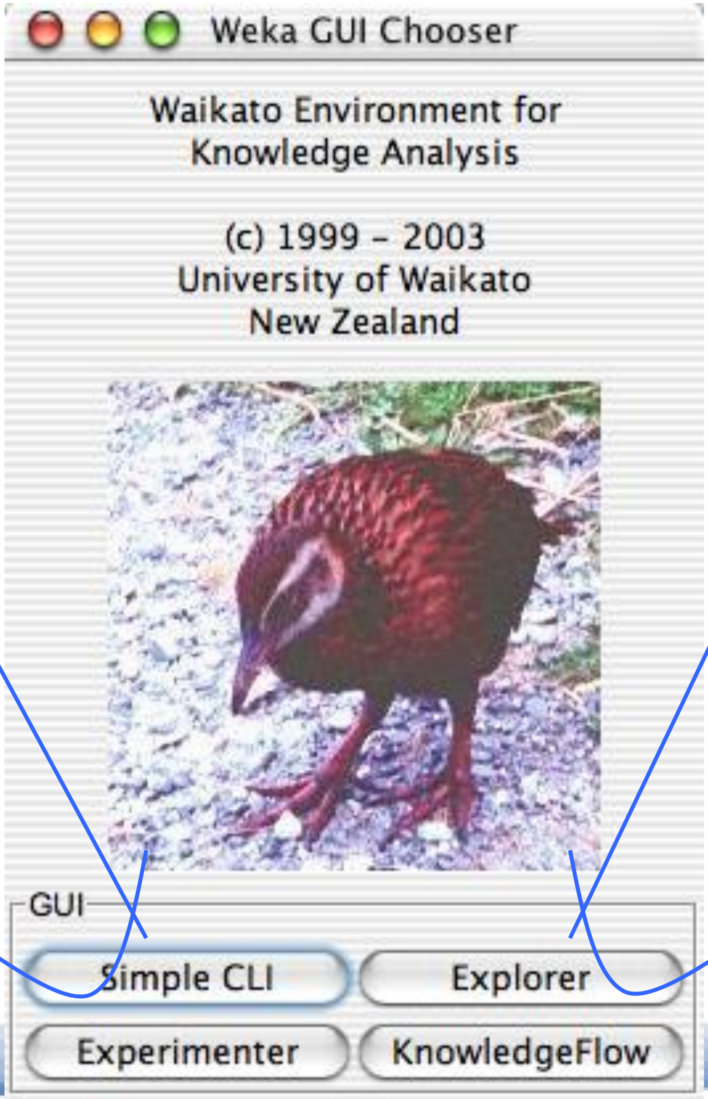
□ Use relative paths

/Users/eibe/Documents/datasets/UCI/iris.arff
/Users/eibe/Documents/datasets/UCI/vote.arff
/Users/eibe/Documents/datasets/UCI/glass.arff

Algorithms

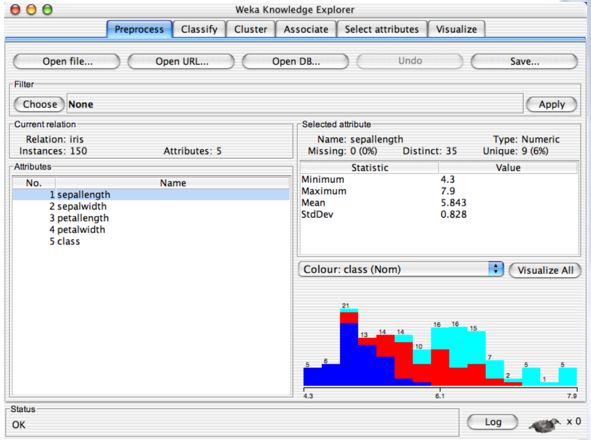
J48 -C 0.25 -M 2
NeuralNetwork -L 0.3 -M 0.2 -N 500 -V 0 -S 0 -E 20 -H a
NaiveBayes

Notes



Waikato Environment for
Knowledge Analysis

(c) 1999 – 2003
University of Waikato
New Zealand



Filter

None

Current relation

Relation: iris
Instances: 150   Attributes: 5

Attributes

No.	Name
1	sepallength
2	sepalwidth
3	petallength
4	petalwidth
5	class

Selected attribute

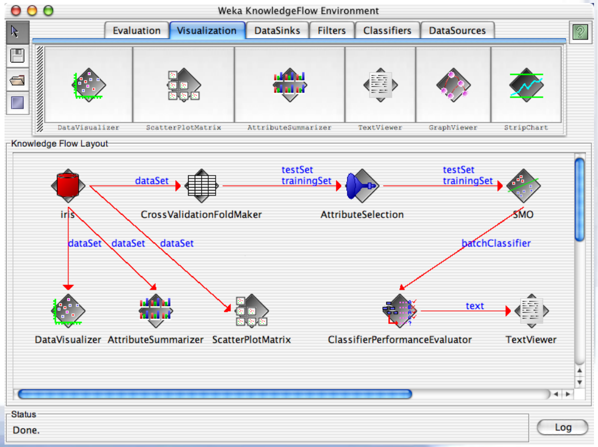
Name: sepallength   Type: Numeric
Missing: 0 (0%)   Distinct: 35   Unique: 9 (6%)

Statistic	Value
Minimum	4.3
Maximum	7.9
Mean	5.843
StdDev	0.828

Colour: class (Nom)

Status

OK



Status

Done.

WEKA:: Data format

Uses flat text files to describe the data

Can work with a wide variety of data files including its own ".arff" format and C4.5 file formats

Data can be imported from a file in various formats:

- ARFF, CSV, C4.5, binary

Data can also be read from a URL or from an SQL database (using JDBC)

WEKA:: ARRF file format

numeric attribute

nominal attribute

```
@relation heart-disease-simplified

@attribute age numeric
@attribute sex { female, male}
@attribute chest_pain_type { typ_angina, asympt, non_anginal, atyp_angina}
@attribute cholesterol numeric
@attribute exercise_induced_angina { no, yes}
@attribute class { present, not_present}

@data
63,male,typ_angina,233,no,not_present
67,male,asympt,286,yes,present
67,male,asympt,229,yes,present
38,female,non_anginal,?,no,not_present
...
```

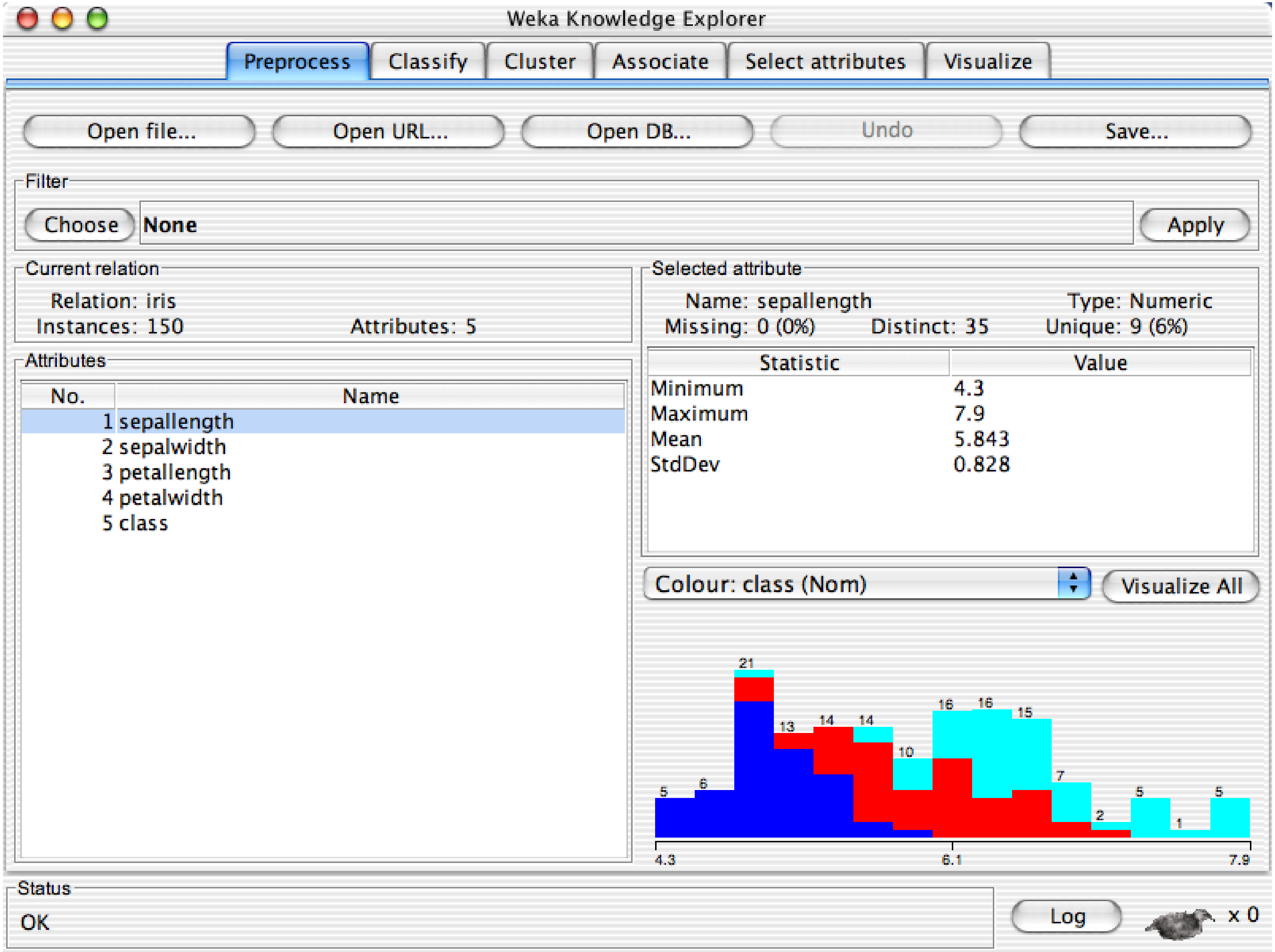
A more thorough description is available here
http://www.cs.waikato.ac.nz/~ml/weka/arff.html

WEKA:: Explorer: Preprocessing

Pre-processing tools in WEKA are called “filters”

WEKA contains filters for:

- Discretization, normalization, resampling, attribute selection, transforming, combining attributes, etc



Filter

None

Current relation

Relation: iris
Instances: 150 Attributes: 5

Attributes

No.	Name
1	sepallength
2	sepalwidth
3	petallength
4	petalwidth
5	class

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

Status

OK

WEKA:: Explorer: building “classifiers”

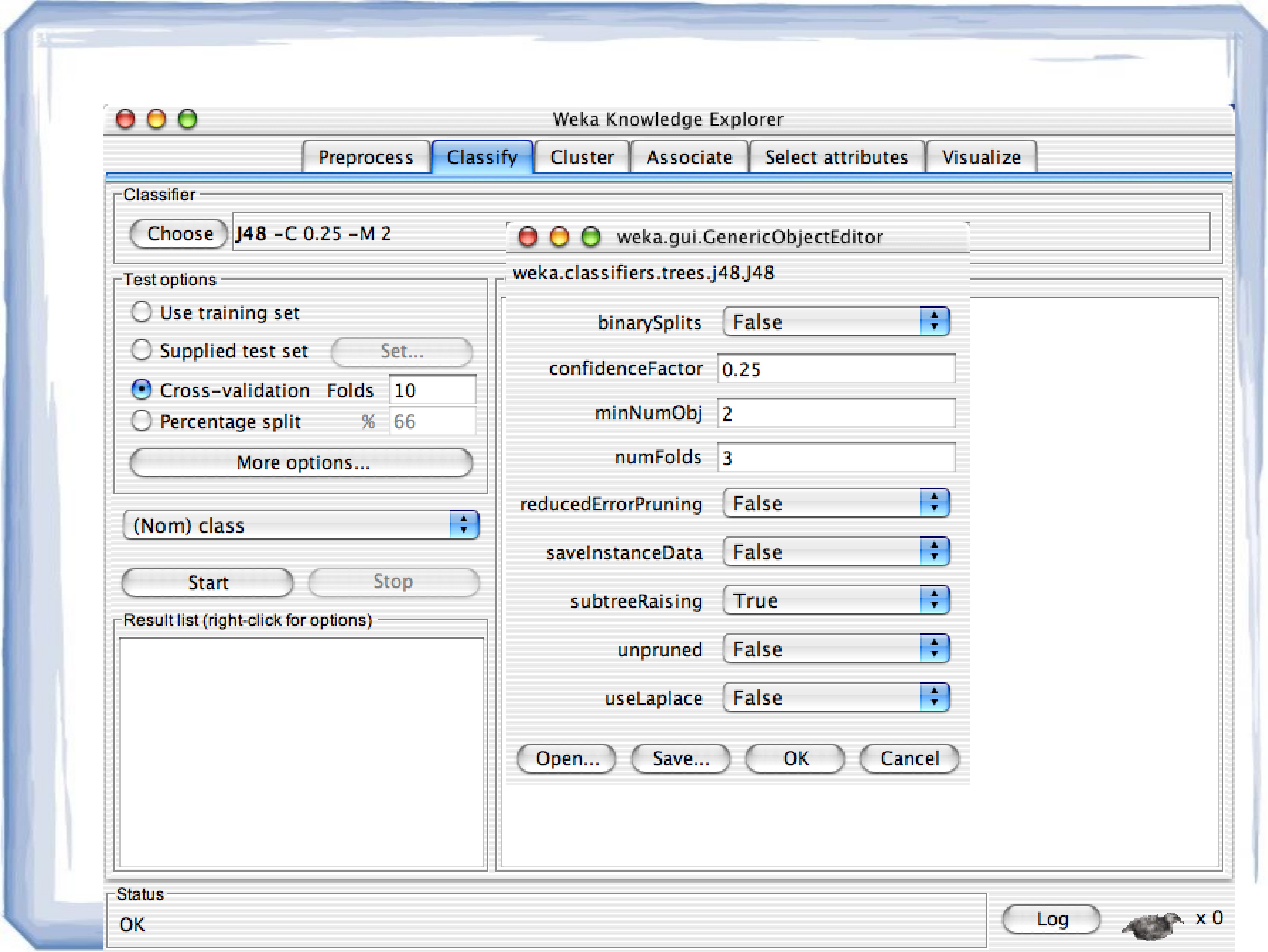
Classifiers in WEKA are models for predicting nominal or numeric quantities

Implemented learning schemes include:

- Decision trees and lists, instance-based classifiers, support vector machines, multi-layer perceptrons, logistic regression, Bayes’ nets, …

“Meta”-classifiers include:

- Bagging, boosting, stacking, error-correcting output codes, locally weighted learning, …



J48 -C 0.25 -M 2

○ Use training set

○ Supplied test set

● Cross-validation Folds 10

○ Percentage split % 66

(Nom) class

weka.classifiers.trees.j48.J48

binarySplits	False
confidenceFactor	0.25
minNumObj	2
numFolds	3
reducedErrorPruning	False
saveInstanceData	False
subtreeRaising	True
unpruned	False
useLaplace	False

OK

WEKA:: Explorer: Clustering

- Example showing simple K-means on the Iris dataset

Overview

- What is Weka?
- Where to find Weka?
- Command Line Vs GUI
- Datasets in Weka
- ARFF Files
- Classifiers in Weka
- Filters

What is Weka?

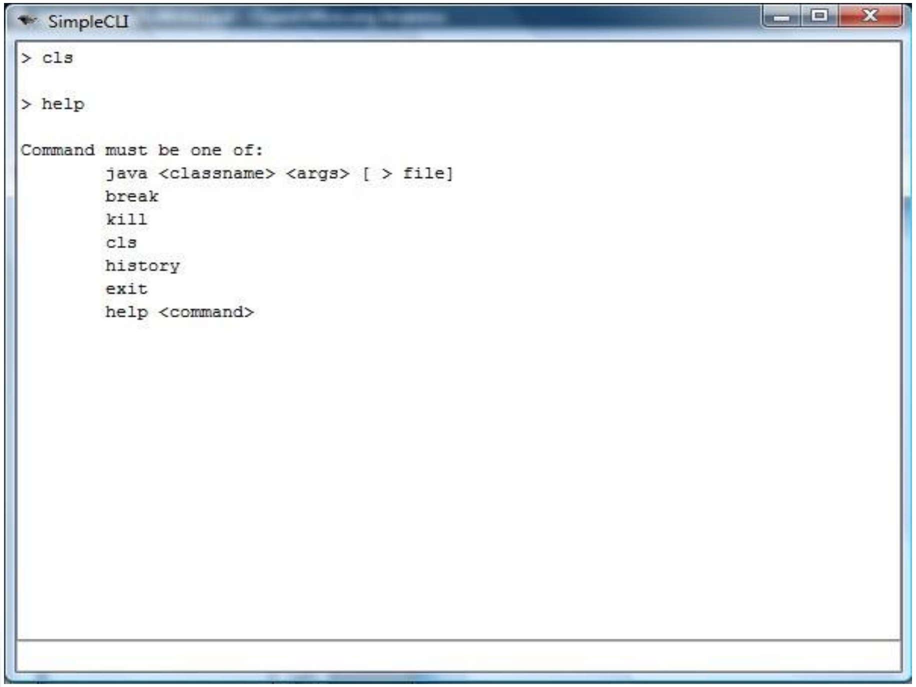
- Weka is a collection of machine learning algorithms for data mining tasks. The algorithms can either be applied directly to a dataset or called from your own Java code. Weka contains tools for data pre-processing, classification, regression, clustering, association rules, and visualization. It is also well-suited for developing new machine learning schemes.

Where to find Weka

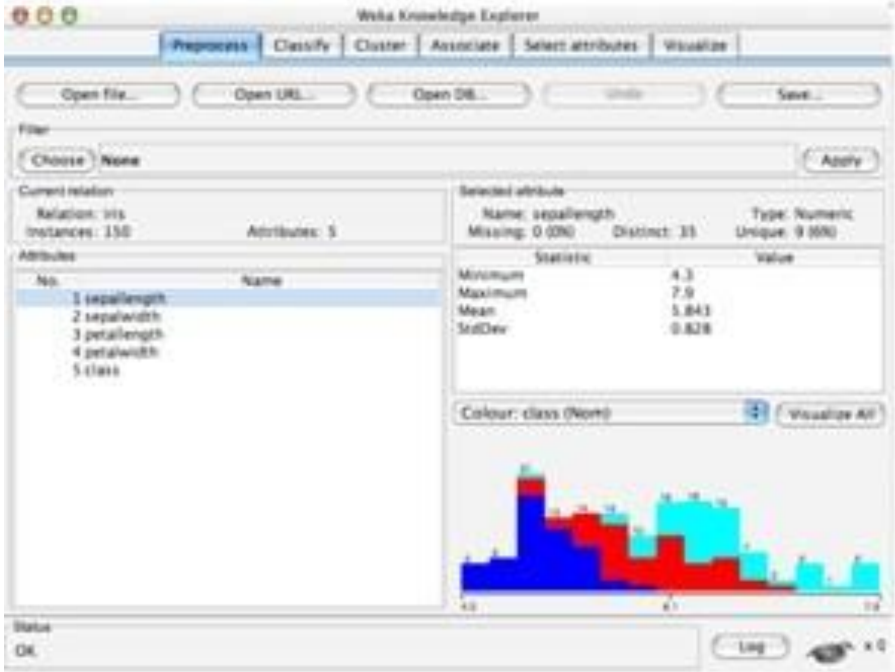
- Weka website (Latest version 3.6):
  - http://www.cs.waikato.ac.nz/ml/weka/

- Weka Manual:
  - http://transact.dl.sourceforge.net/sourceforge/weka/WekaManual-3.6.0.pdf

CLI Vs GUI



```
> cls

> help

Command must be one of:
        java <classname> <args> [ > file]
        break
        kill
        cls
        history
        exit
        help <command>
```



None

Current relation

Relation: iris
Instances: 150 Attributes: 5

Attributes

No.	Name
1	sepallength
2	sepalwidth
3	petallength
4	petalwidth
5	class

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

Colour: class (Nom)

Status

OK

- Recommended for in-depth usage
- Offers some functionality not available via the GUI

- Explorer
- Experimenter
- Knowledge Flow

Datasets in Weka

- Each entry in a dataset is an instance of the java class:
  - weka.core.Instance
- Each instance consists of a number of attributes

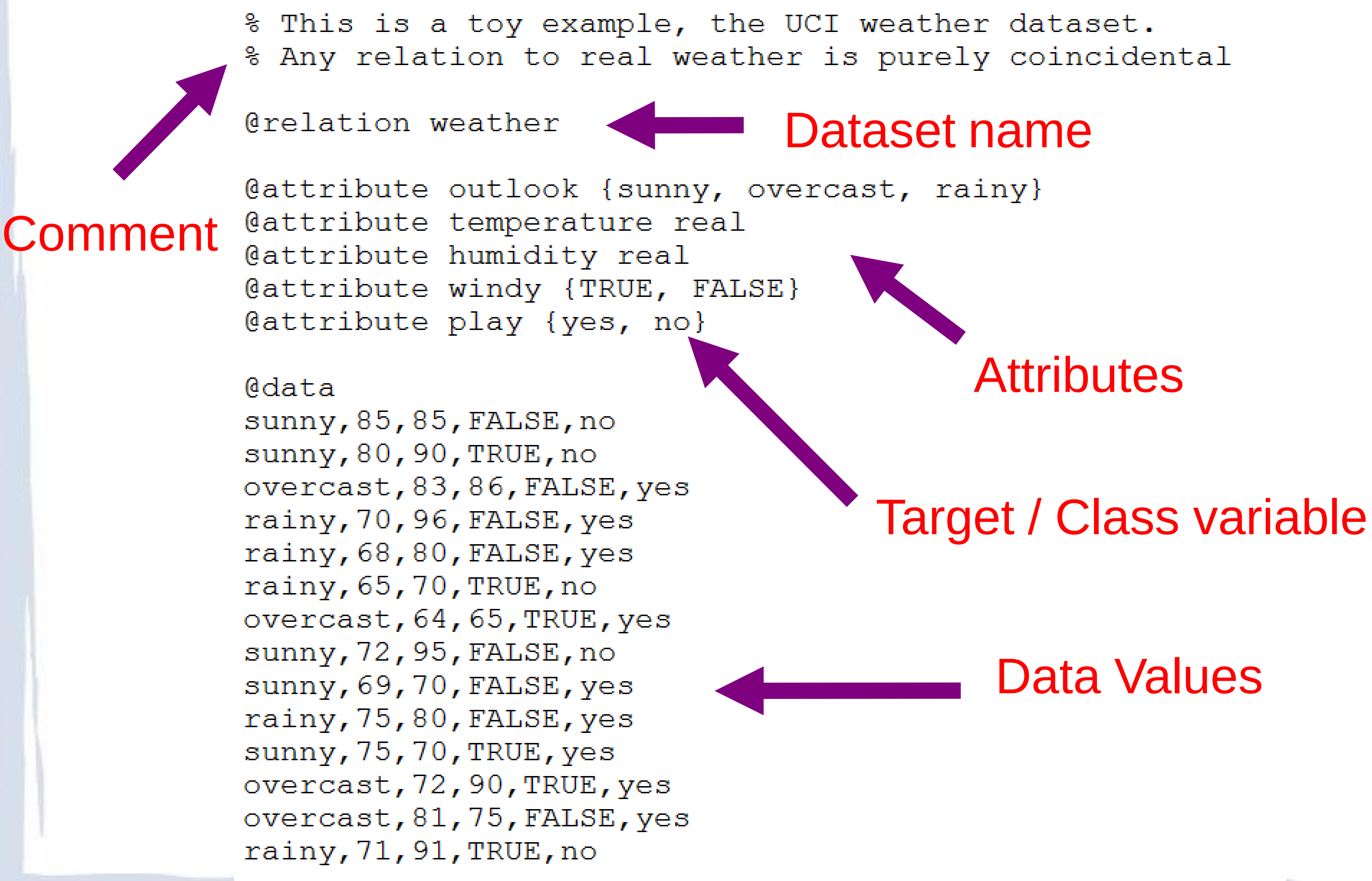
Attributes

- *Nominal*: one of a predefined list of values
  - e.g. red, green, blue
- *Numeric*: A real or integer number
- *String:* Enclosed in “double quotes”
- *Date*
- *Relational*

ARFF Files

- The external representation of an Instances class
- Consists of:
  - A header: Describes the attribute types
  - Data section: Comma separated list of data

ARFF File Example



```
% This is a toy example, the UCI weather dataset.
% Any relation to real weather is purely coincidental

@relation weather

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

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

Comment

Dataset name

Attributes

Target / Class variable

Data Values

Assignment ARFF Files

- Credit-g
- Heart-c
- Hepatitis
- Vowel
- Zoo

- http://www.cs.auckland.ac.nz/~pat/weka/

ARFF Files

- Basic statistics and validation by running:
  - java weka.core.Instances data/soybean.arff

Classifiers in Weka

- Learning algorithms in Weka are derived from the abstract class:
  - weka.classifiers.Classifier
- Simple classifier: ZeroR
  - Just determines the most common class
  - Or the median (in the case of numeric values)
  - Tests how well the class can be predicted without considering other attributes
  - Can be used as a Lower Bound on Performance.

Classifiers in Weka

- Simple Classifier Example
  - java weka.classifiers.rules.ZeroR -t data/weather.arff
  - java weka.classifiers.trees.J48 -t data/weather.arff
- Help Command
  - java weka.classifiers.trees.J48 -h

Classifiers in Weka

- **Soybean.arff** split into train and test set
  - Soybean-train.arff
  - Soybean-test.arff
- Input command:

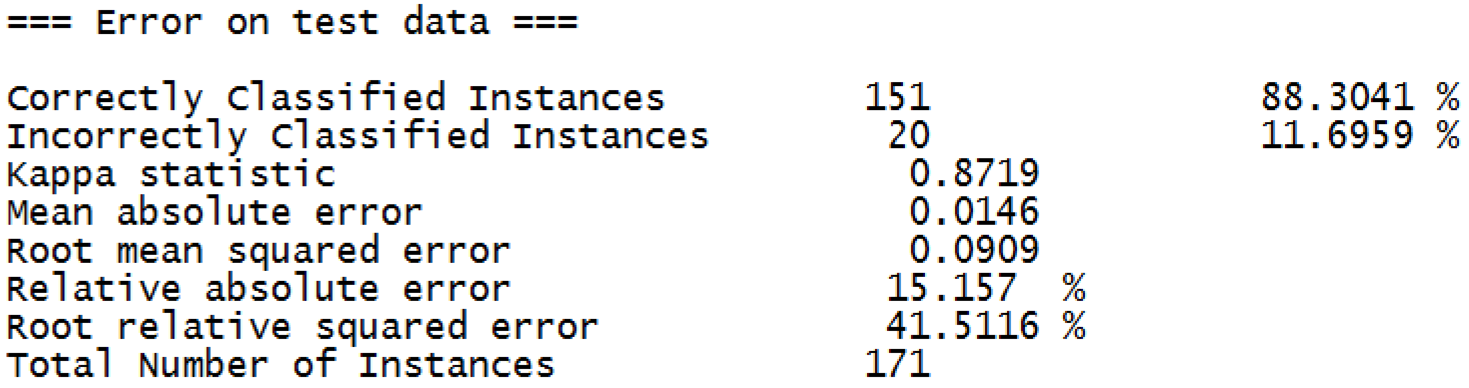
  -
    ```
    java weka.classifiers.trees.J48 -t soybean-train.arff -T soybean-test.arff -i
    ```

Training data

Test data

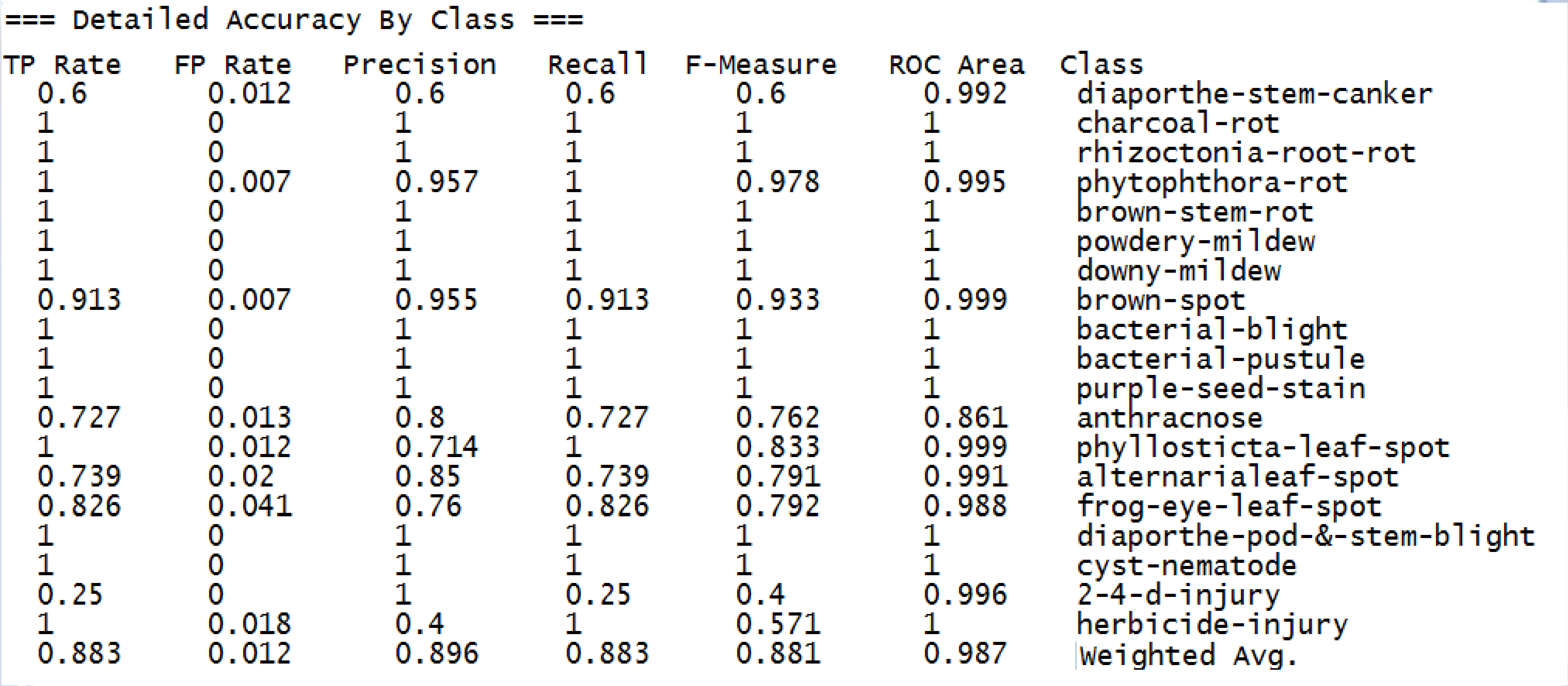
Provides more detailed output

Soybean Results



```
=== Error on test data ===

Correctly Classified Instances         151               88.3041 %
Incorrectly Classified Instances        20               11.6959 %
Kappa statistic                          0.8719
Mean absolute error                      0.0146
Root mean squared error                  0.0909
Relative absolute error                 15.157  %
Root relative squared error             41.5116 %
Total Number of Instances              171
```

Soybean Results (cont...)



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

TP Rate   FP Rate   Precision   Recall  F-Measure   ROC Area  Class
  0.6       0.012     0.6         0.6     0.6         0.992     diaporthe-stem-canker
  1         0         1           1       1           1         charcoal-rot
  1         0         1           1       1           1         rhizoctonia-root-rot
  1         0.007     0.957       1       0.978       0.995     phytophthora-rot
  1         0         1           1       1           1         brown-stem-rot
  1         0         1           1       1           1         powdery-mildew
  1         0         1           1       1           1         downy-mildew
  0.913     0.007     0.955       0.913   0.933       0.999     brown-spot
  1         0         1           1       1           1         bacterial-blight
  1         0         1           1       1           1         bacterial-pustule
  1         0         1           1       1           1         purple-seed-stain
  0.727     0.013     0.8         0.727   0.762       0.861     anthracnose
  1         0.012     0.714       1       0.833       0.999     phyllosticta-leaf-spot
  0.739     0.02      0.85        0.739   0.791       0.991     alternarialeaf-spot
  0.826     0.041     0.76        0.826   0.792       0.988     frog-eye-leaf-spot
  1         0         1           1       1           1         diaporthe-pod-&-stem-blight
  1         0         1           1       1           1         cyst-nematode
  0.25      0         1           0.25    0.4         0.996     2-4-d-injury
  1         0.018     0.4         1       0.571       1         herbicide-injury
  0.883     0.012     0.896       0.883   0.881       0.987     Weighted Avg.
```

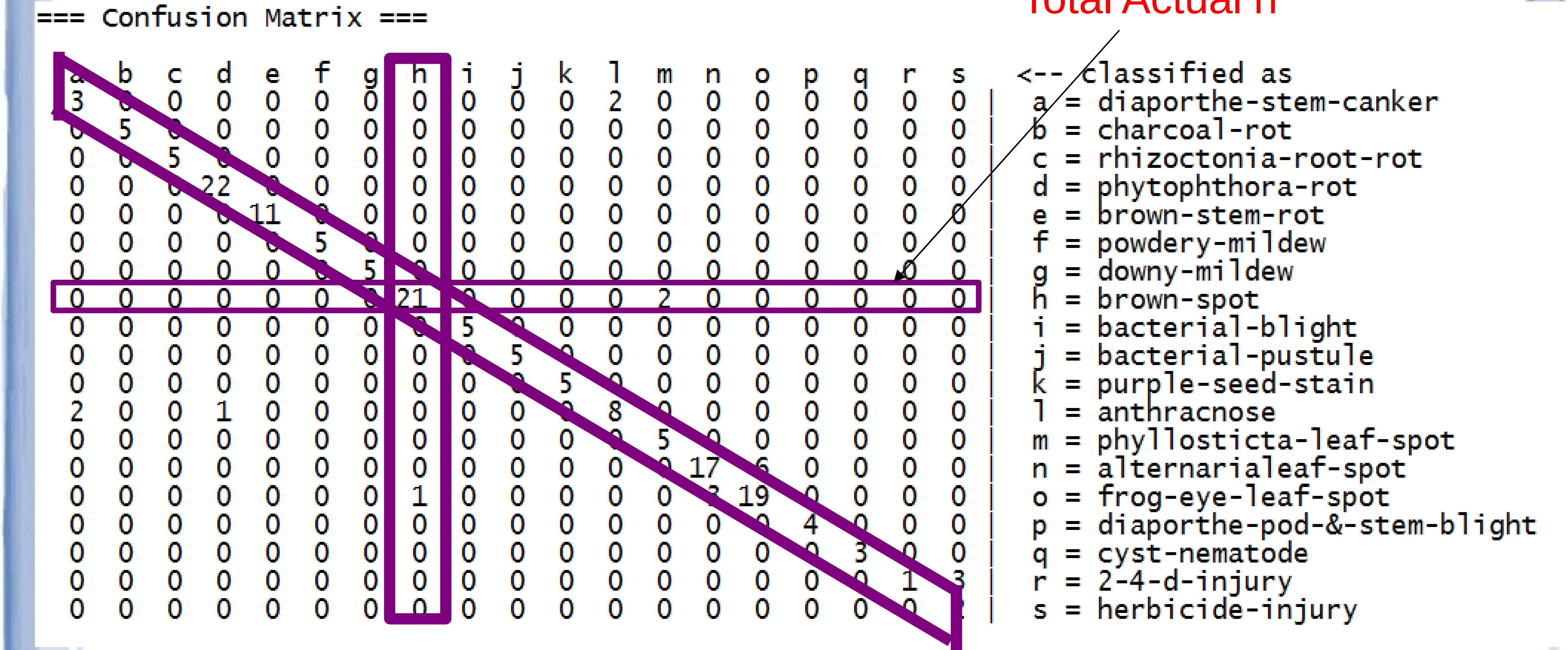
Soybean Results (cont...)

- True Positive (*TP*)
  - Proportion classified as class x / Actual total in class x
  - Equivalent to Recall
- False Positive (*FP*)
  - Proportion incorrectly classified as class x / Actual total of all classes, except x

Soybean Results (cont...)

- Precision:
  - Proportion of the examples which truly have class x / Total classified as class x
- F-measure:
  - 2*Precision*Recall / (Precision + Recall)
  - i.e. A combined measure for precision and recall

Soybean Results (cont...)



```
=== Confusion Matrix ===

  a  b  c  d  e  f  g  h  i  j  k  l  m  n  o  p  q  r  s   <-- classified as
  3  0  0  0  0  0  0  0  0  0  0  2  0  0  0  0  0  0  0 |  a = diaporthe-stem-canker
  0  5  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0 |  b = charcoal-rot
  0  0  5  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0 |  c = rhizoctonia-root-rot
  0  0  0 22  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0 |  d = phytophthora-rot
  0  0  0  0 11  0  0  0  0  0  0  0  0  0  0  0  0  0  0 |  e = brown-stem-rot
  0  0  0  0  0  5  0  0  0  0  0  0  0  0  0  0  0  0  0 |  f = powdery-mildew
  0  0  0  0  0  0  5  0  0  0  0  0  0  0  0  0  0  0  0 |  g = downy-mildew
  0  0  0  0  0  0  0 21  0  0  0  0  2  0  0  0  0  0  0 |  h = brown-spot
  0  0  0  0  0  0  0  0  5  0  0  0  0  0  0  0  0  0  0 |  i = bacterial-blight
  0  0  0  0  0  0  0  0  0  5  0  0  0  0  0  0  0  0  0 |  j = bacterial-pustule
  0  0  0  0  0  0  0  0  0  0  5  0  0  0  0  0  0  0  0 |  k = purple-seed-stain
  2  0  0  1  0  0  0  0  0  0  0  8  0  0  0  0  0  0  0 |  l = anthracnose
  0  0  0  0  0  0  0  0  0  0  0  0  5  0  0  0  0  0  0 |  m = phyllosticta-leaf-spot
  0  0  0  0  0  0  0  0  0  0  0  0  0 17  6  0  0  0  0 |  n = alternarialeaf-spot
  0  0  0  0  0  0  0  1  0  0  0  0  0  3 19  0  0  0  0 |  o = frog-eye-leaf-spot
  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  4  0  0  0 |  p = diaporthe-pod-&-stem-blight
  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  3  0  0 |  q = cyst-nematode
  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  1  3 |  r = 2-4-d-injury
  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  2 |  s = herbicide-injury
```

Total Actual h

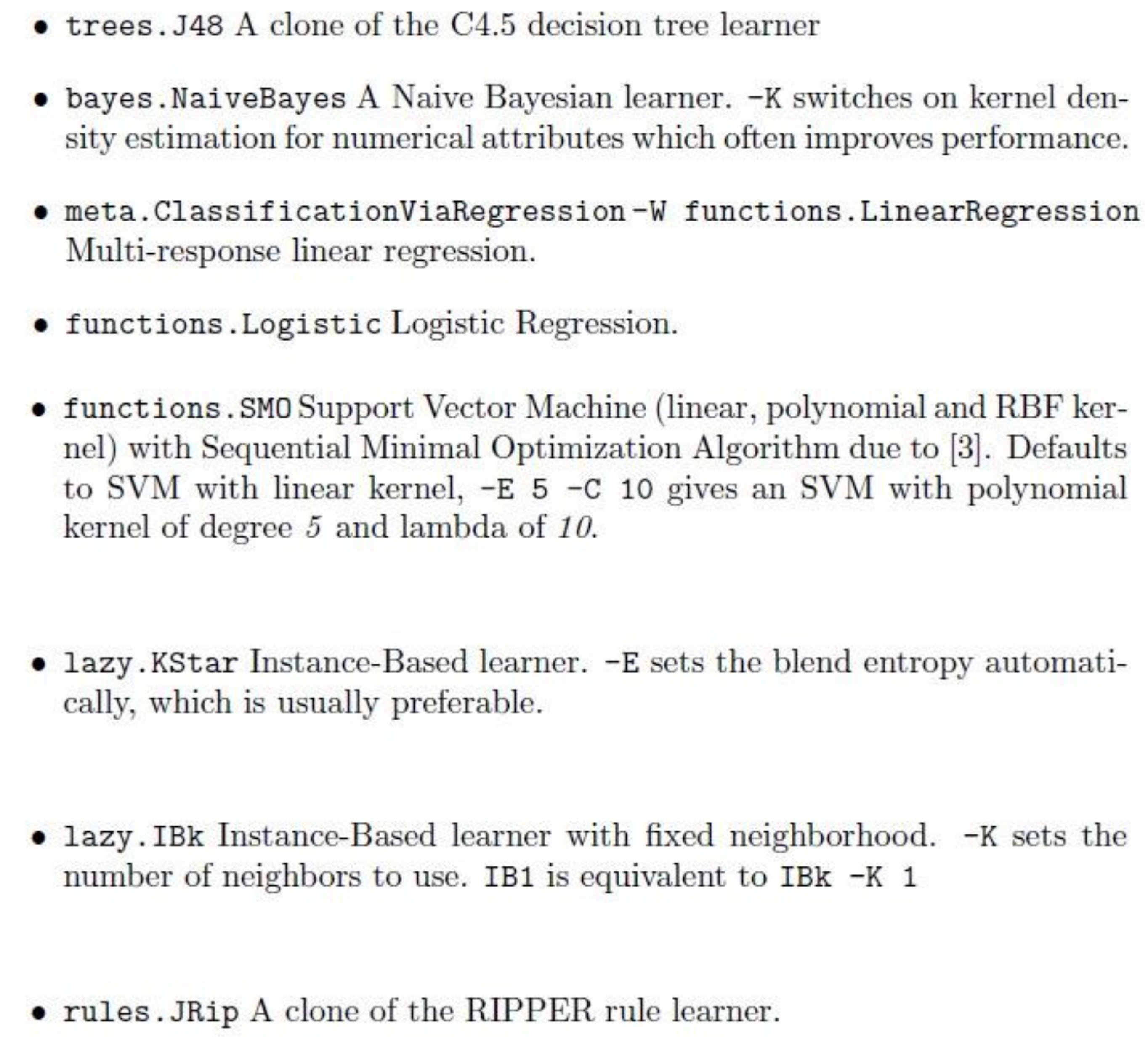
Total Classified as h

Total Correct

Filters

- weka.filters package
- Transform datasets
- Support for data preprocessing
  - e.g. Removing/Adding Attributes
  - e.g. Discretize numeric attributes into nominal ones
- More info in Weka Manual p. 15 & 16.

More Classifiers



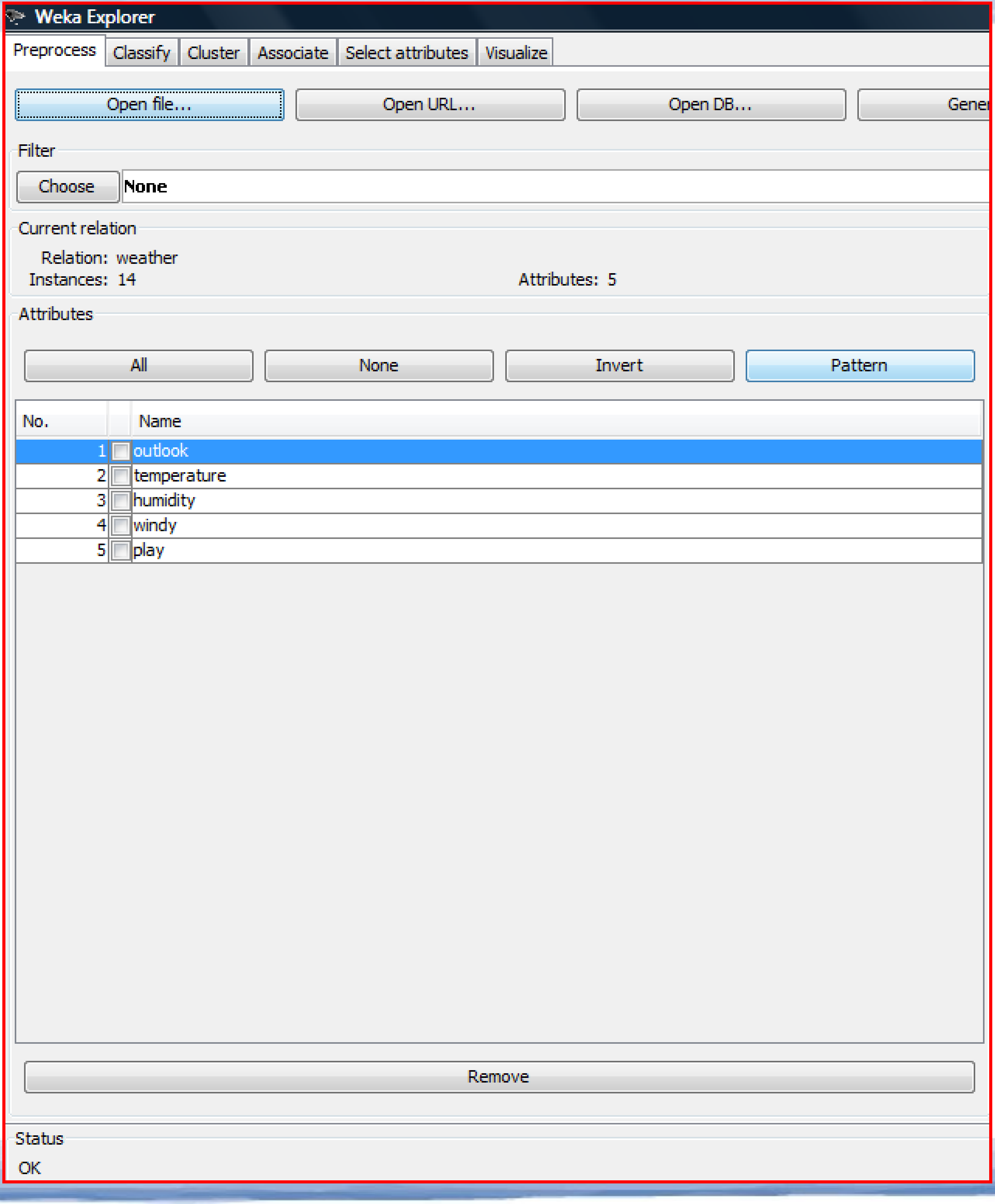
- `trees.J48` A clone of the C4.5 decision tree learner
- `bayes.NaiveBayes` A Naive Bayesian learner. `-K` switches on kernel density estimation for numerical attributes which often improves performance.
- `meta.ClassificationViaRegression -W functions.LinearRegression` Multi-response linear regression.
- `functions.Logistic` Logistic Regression.
- `functions.SMO` Support Vector Machine (linear, polynomial and RBF kernel) with Sequential Minimal Optimization Algorithm due to [3]. Defaults to SVM with linear kernel, `-E 5 -C 10` gives an SVM with polynomial kernel of degree *5* and lambda of *10*.
- `lazy.KStar` Instance-Based learner. `-E` sets the blend entropy automatically, which is usually preferable.
- `lazy.IBk` Instance-Based learner with fixed neighborhood. `-K` sets the number of neighbors to use. `IB1` is equivalent to `IBk -K 1`
- `rules.JRip` A clone of the RIPPER rule learner.

Explorer

- Preprocess
- Classify
- Cluster
- Associate
- Select attributes
- Visualize

Preprocess

- Load Data
- Preprocess Data
- Analyse Attributes



Filter

None

Current relation

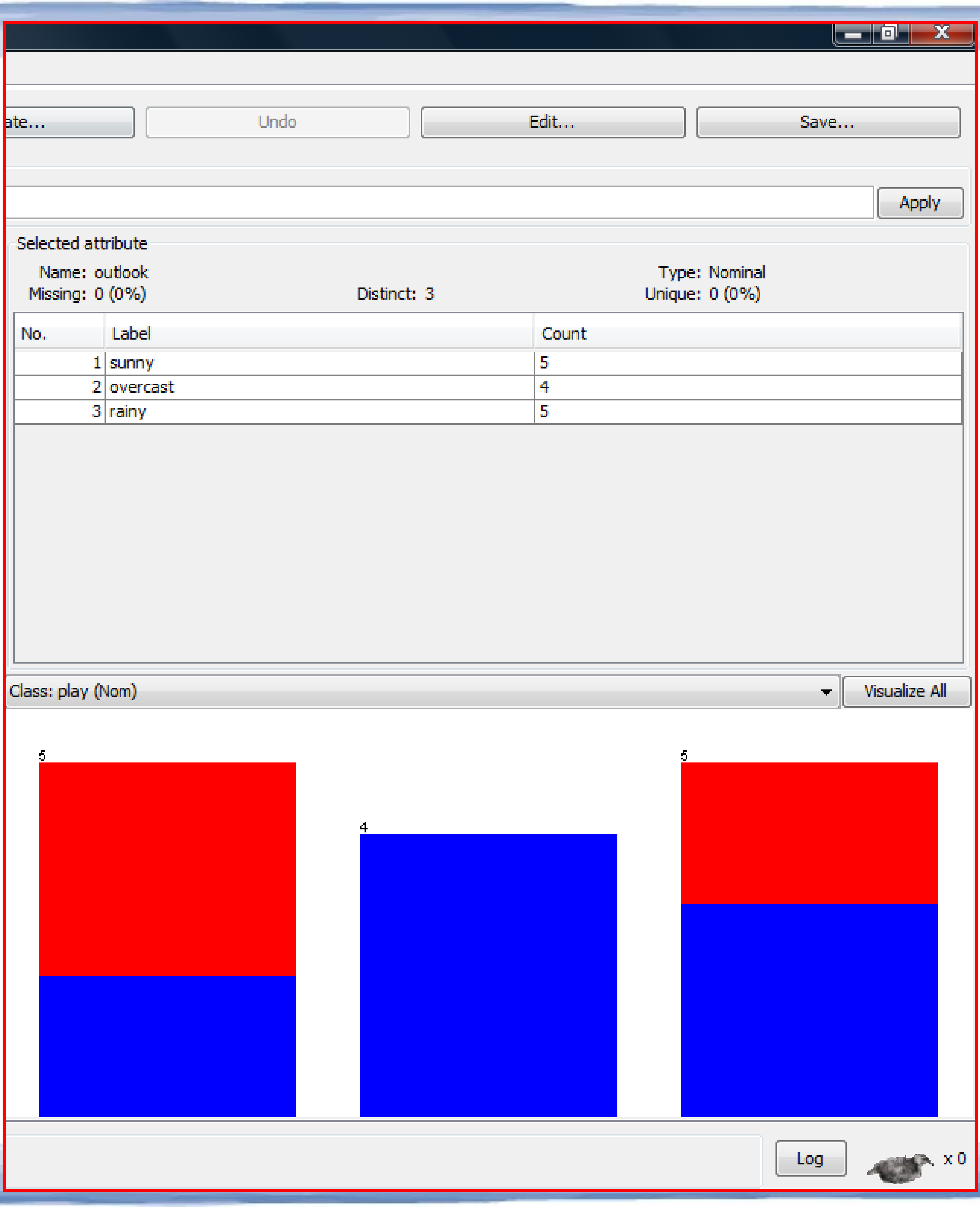
Relation: weather
Instances: 14
Attributes: 5

Attributes

No.		Name
1	□	outlook
2	□	temperature
3	□	humidity
4	□	windy
5	□	play

Status

OK



Selected attribute

Name: outlook Type: Nominal

Missing: 0 (0%) Distinct: 3 Unique: 0 (0%)

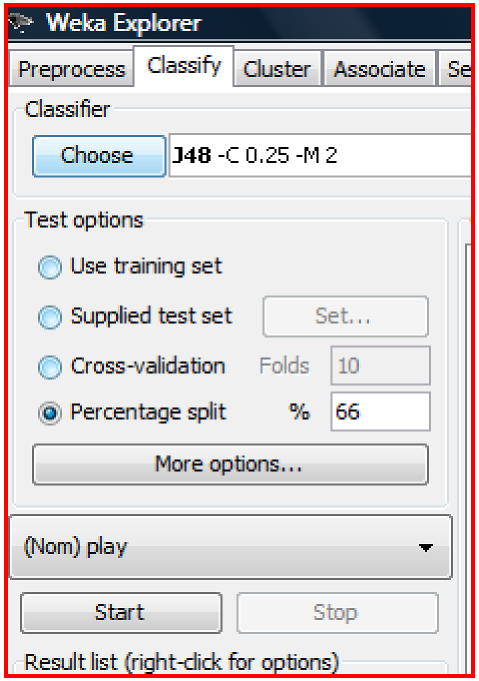
No.	Label	Count
1	sunny	5
2	overcast	4
3	rainy	5

Class: play (Nom)

Classify

- Select Test Options e.g:
  - Use Training Set
  - % Split,
  - Cross Validation...
- Run classifiers
- View results

Classify



Weka Explorer

Classifier

J48 -C 0.25 -M 2

Test options

○ Use training set

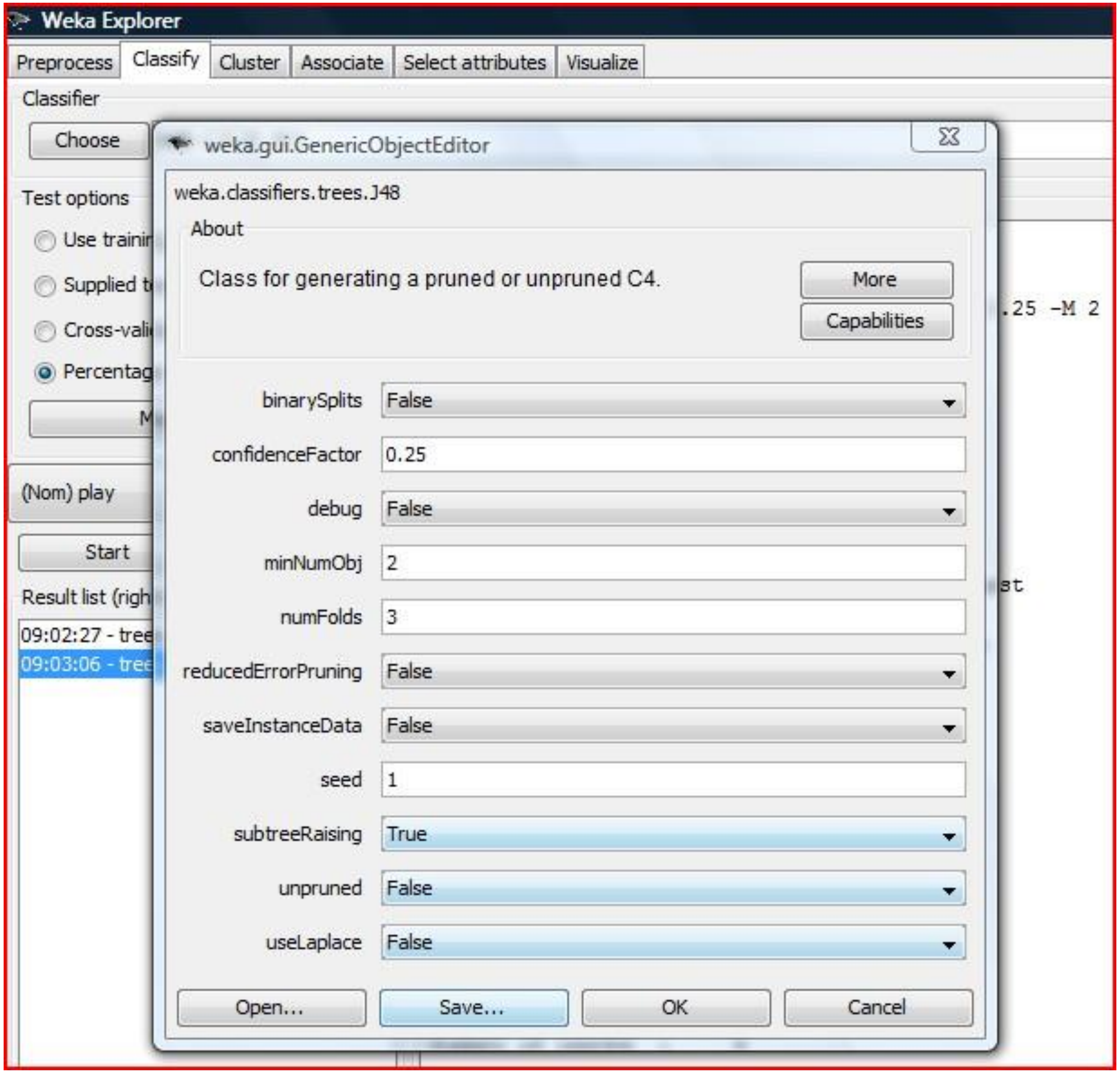
○ Supplied test set

○ Cross-validation Folds 10

● Percentage split % 66

(Nom) play

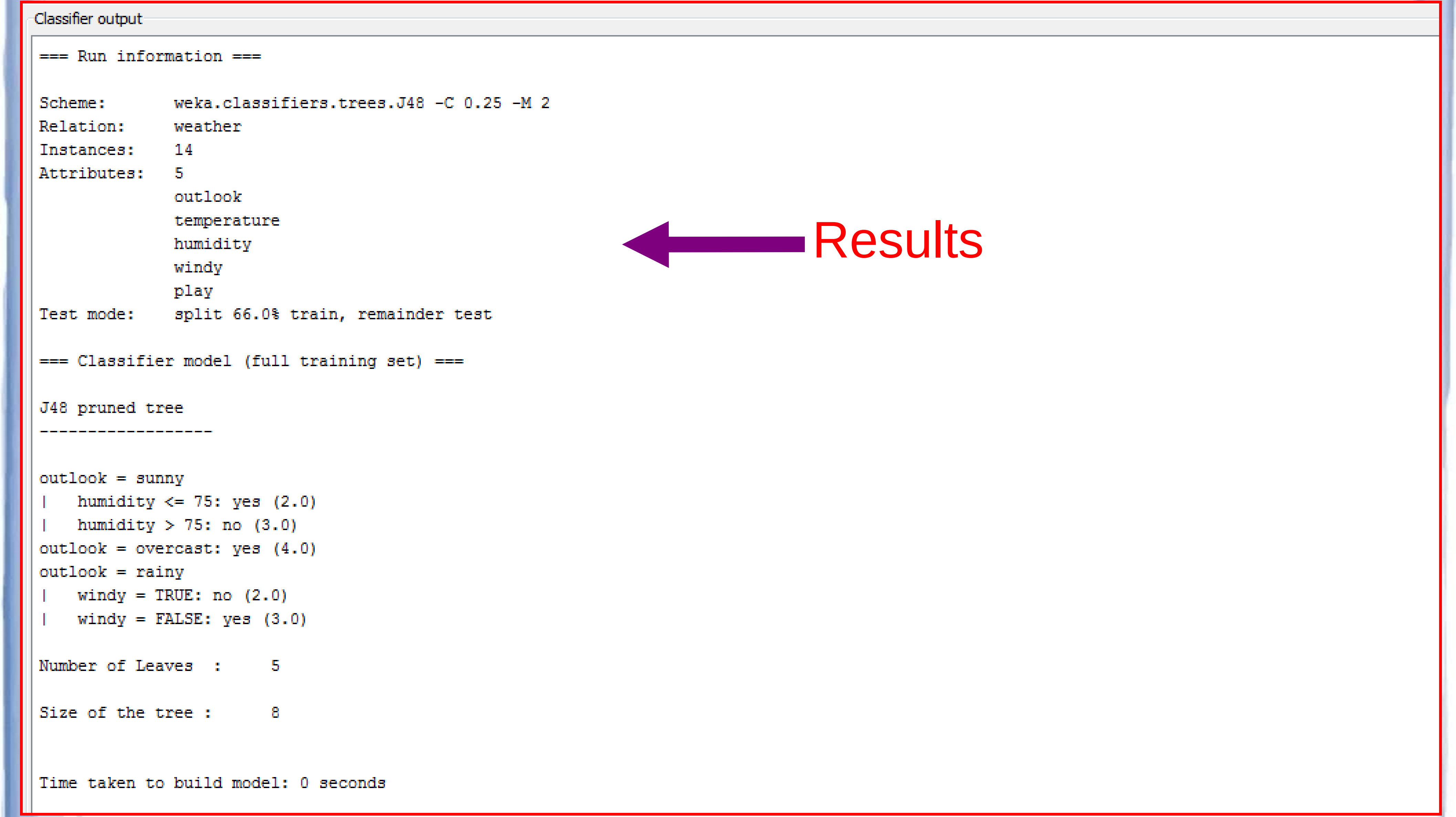
Result list (right-click for options)



weka.classifiers.trees.J48

Class for generating a pruned or unpruned C4.

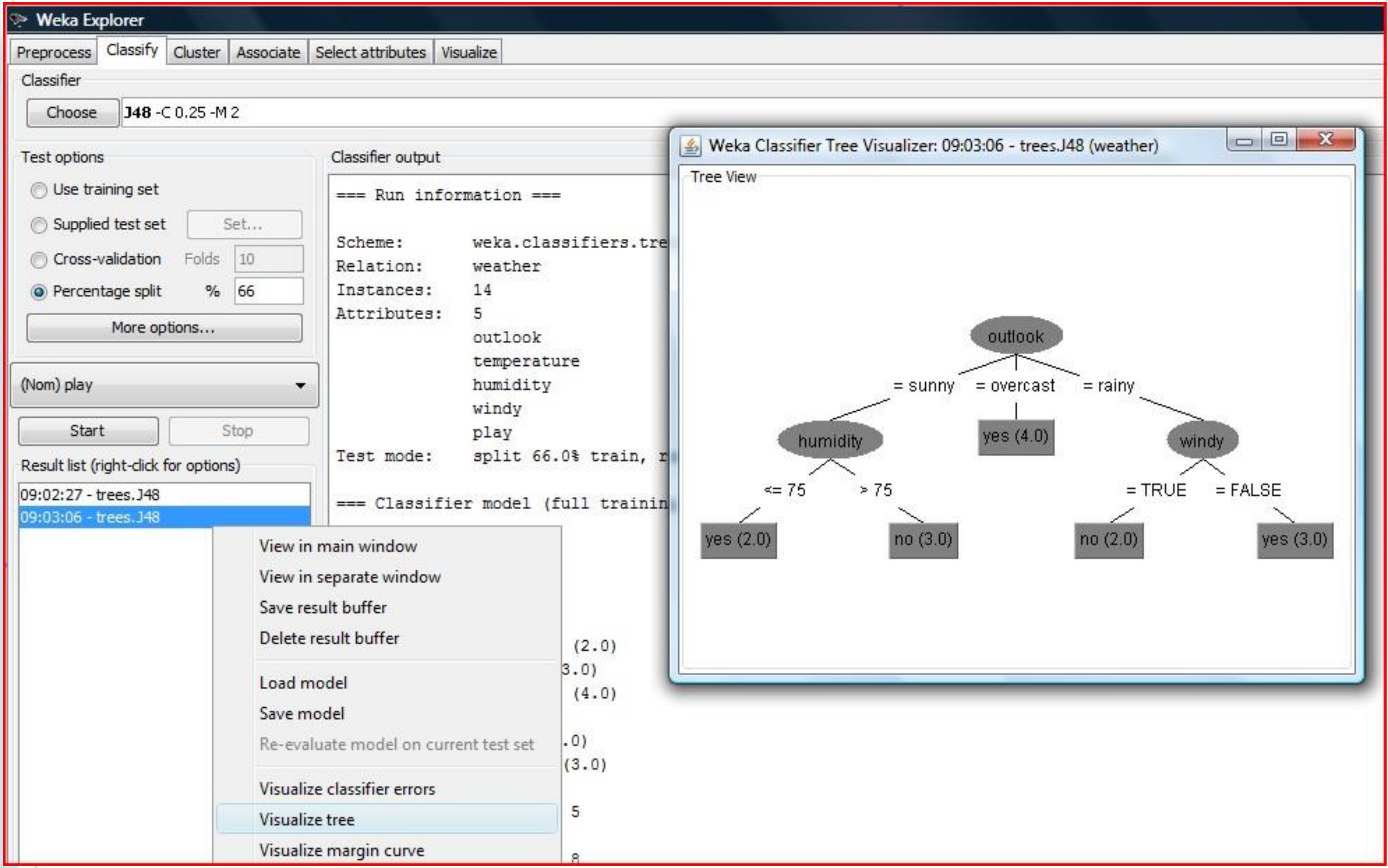
binarySplits	False
confidenceFactor	0.25
debug	False
minNumObj	2
numFolds	3
reducedErrorPruning	False
saveInstanceData	False
seed	1
subtreeRaising	True
unpruned	False
useLaplace	False



Classifier output

```
=== Run information ===

Scheme:       weka.classifiers.trees.J48 -C 0.25 -M 2
Relation:     weather
Instances:    14
Attributes:   5
              outlook
              temperature
              humidity
              windy
              play
Test mode:    split 66.0% train, remainder test

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

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

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

Number of Leaves  :     5

Size of the tree :      8


Time taken to build model: 0 seconds
```

Results



Classifier: J48 -C 0.25 -M 2

Test options:
- ○ Use training set
- ○ Supplied test set
- ○ Cross-validation Folds 10
- ● Percentage split % 66

(Nom) play

Result list (right-click for options)
- 09:02:27 - trees.J48
- 09:03:06 - trees.J48

Classifier output

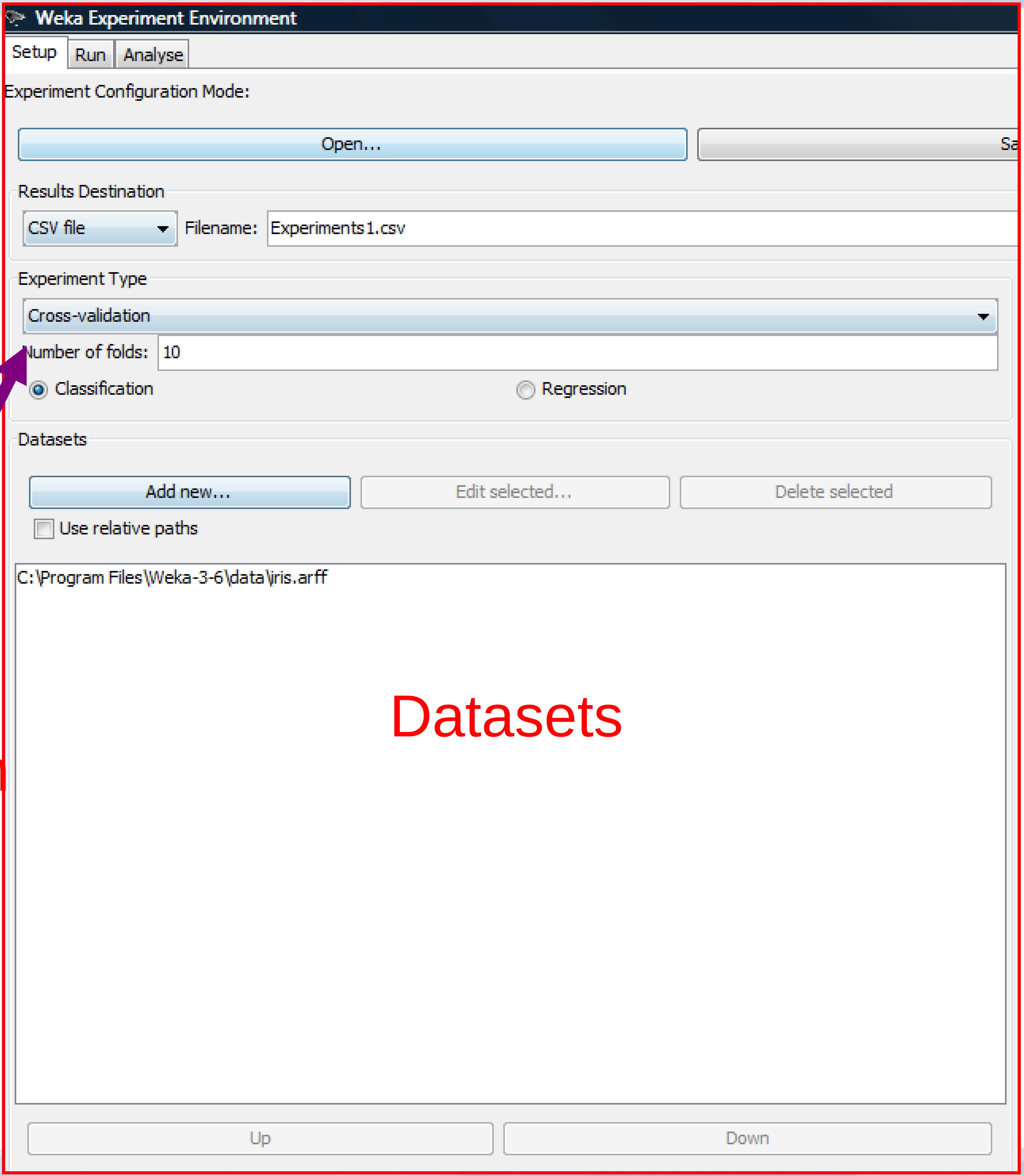
```
=== Run information ===

Scheme:       weka.classifiers.tre
Relation:     weather
Instances:    14
Attributes:   5
              outlook
              temperature
              humidity
              windy
              play
Test mode:    split 66.0% train, r

=== Classifier model (full trainin

 (2.0)
3.0)
 (4.0)

.0)
(3.0)

5

8
```

Experimenter

- Allows users to create, run, modify and analyse experiments in a more convenient manner than when processing individually.
  - Setup
  - Run
  - Analyse

Experimenter: Setup

- Simple/Advanced
- Results Destinations
  - ARFF
  - CSV
  - JDBC Database



Weka Experiment Environment

Setup | Run | Analyse

Experiment Configuration Mode:

Open... | Sa

Results Destination

CSV file ▾ Filename: Experiments1.csv

Experiment Type

Cross-validation ▾

Number of folds: 10

● Classification ○ Regression

Datasets

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

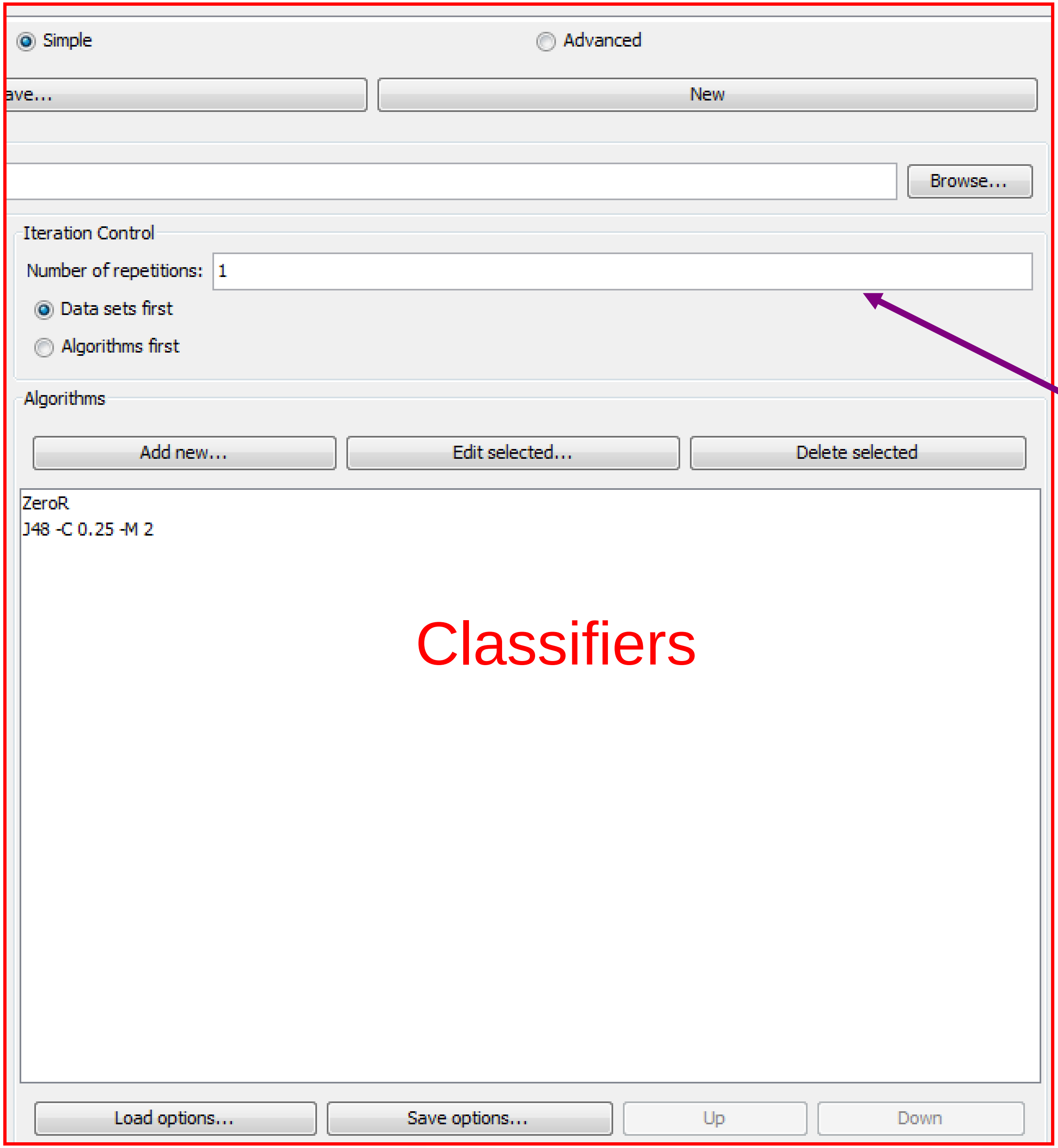
□ Use relative paths

C:\Program Files\Weka-3-6\data\iris.arff

Up | Down

10-fold Cross Validation

Datasets



● Simple ○ Advanced

Iteration Control

Number of repetitions: 1

● Data sets first

○ Algorithms first

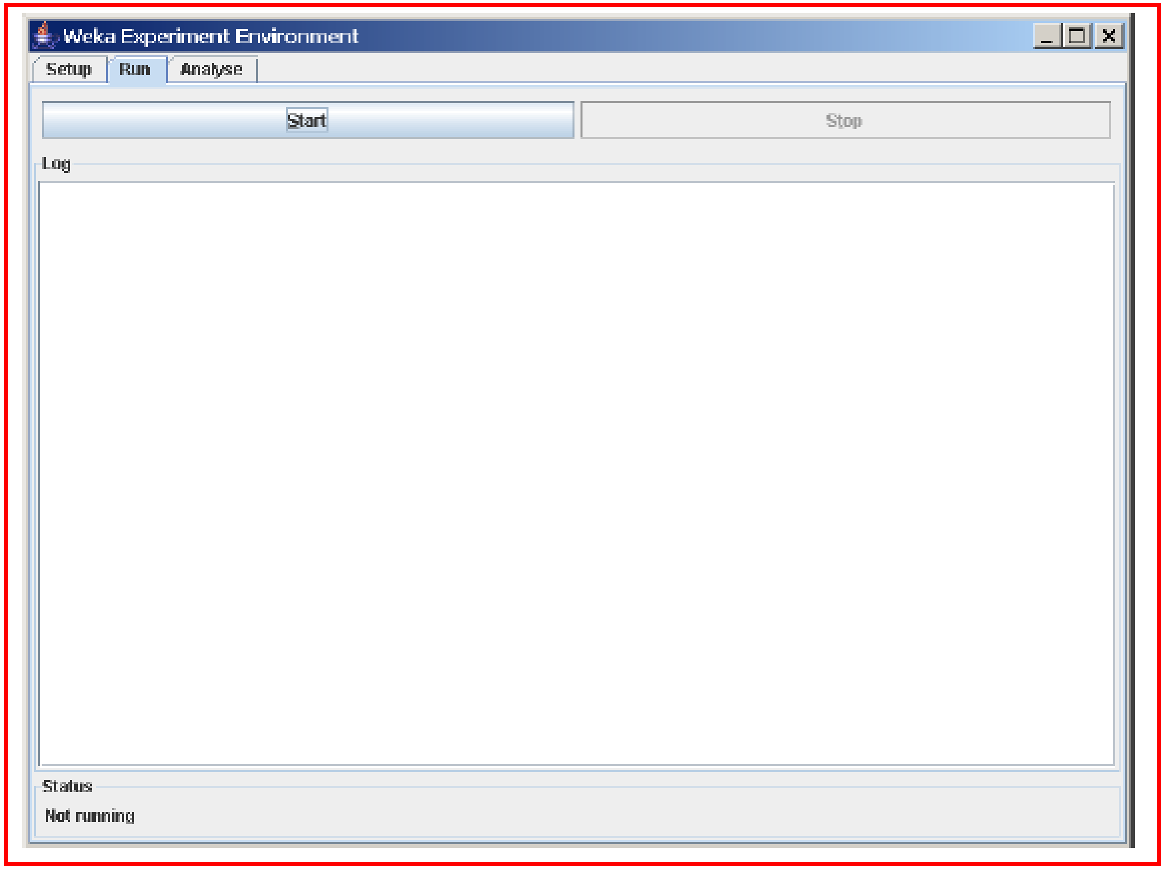
Algorithms

ZeroR
J48 -C 0.25 -M 2

Num of runs

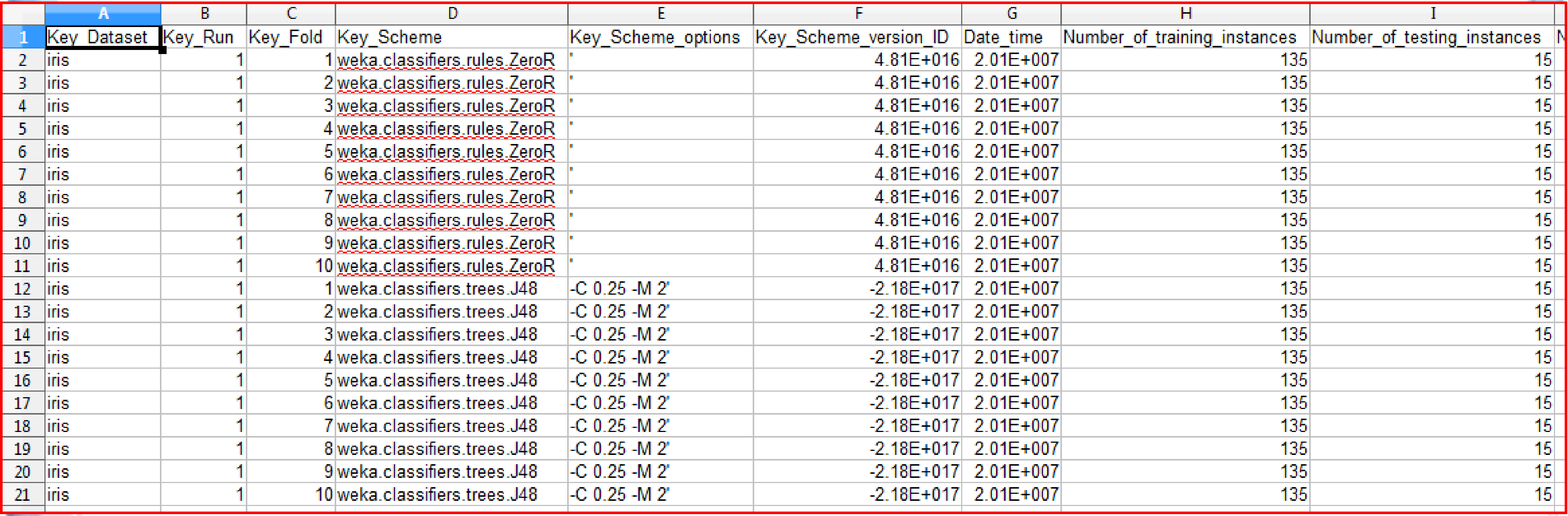
Classifiers

Run Simple Experiment



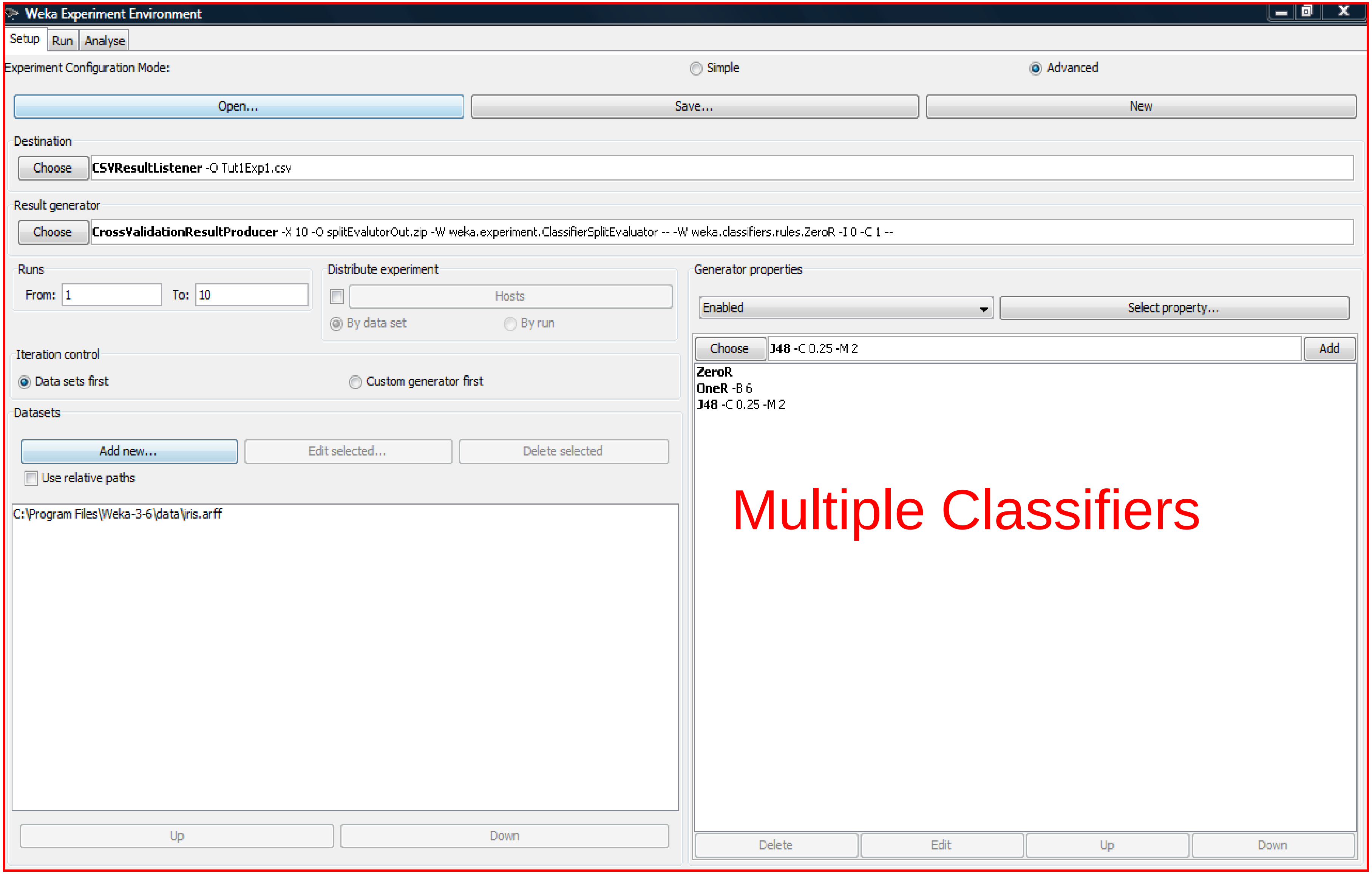
Not running

Results



	A	B	C	D	E	F	G	H	I	
1	Key_Dataset	Key_Run	Key_Fold	Key_Scheme	Key_Scheme_options	Key_Scheme_version_ID	Date_time	Number_of_training_instances	Number_of_testing_instances	N
2	iris	1	1	weka.classifiers.rules.ZeroR	'	4.81E+016	2.01E+007	135	15	
3	iris	1	2	weka.classifiers.rules.ZeroR	'	4.81E+016	2.01E+007	135	15	
4	iris	1	3	weka.classifiers.rules.ZeroR	'	4.81E+016	2.01E+007	135	15	
5	iris	1	4	weka.classifiers.rules.ZeroR	'	4.81E+016	2.01E+007	135	15	
6	iris	1	5	weka.classifiers.rules.ZeroR	'	4.81E+016	2.01E+007	135	15	
7	iris	1	6	weka.classifiers.rules.ZeroR	'	4.81E+016	2.01E+007	135	15	
8	iris	1	7	weka.classifiers.rules.ZeroR	'	4.81E+016	2.01E+007	135	15	
9	iris	1	8	weka.classifiers.rules.ZeroR	'	4.81E+016	2.01E+007	135	15	
10	iris	1	9	weka.classifiers.rules.ZeroR	'	4.81E+016	2.01E+007	135	15	
11	iris	1	10	weka.classifiers.rules.ZeroR	'	4.81E+016	2.01E+007	135	15	
12	iris	1	1	weka.classifiers.trees.J48	-C 0.25 -M 2'	-2.18E+017	2.01E+007	135	15	
13	iris	1	2	weka.classifiers.trees.J48	-C 0.25 -M 2'	-2.18E+017	2.01E+007	135	15	
14	iris	1	3	weka.classifiers.trees.J48	-C 0.25 -M 2'	-2.18E+017	2.01E+007	135	15	
15	iris	1	4	weka.classifiers.trees.J48	-C 0.25 -M 2'	-2.18E+017	2.01E+007	135	15	
16	iris	1	5	weka.classifiers.trees.J48	-C 0.25 -M 2'	-2.18E+017	2.01E+007	135	15	
17	iris	1	6	weka.classifiers.trees.J48	-C 0.25 -M 2'	-2.18E+017	2.01E+007	135	15	
18	iris	1	7	weka.classifiers.trees.J48	-C 0.25 -M 2'	-2.18E+017	2.01E+007	135	15	
19	iris	1	8	weka.classifiers.trees.J48	-C 0.25 -M 2'	-2.18E+017	2.01E+007	135	15	
20	iris	1	9	weka.classifiers.trees.J48	-C 0.25 -M 2'	-2.18E+017	2.01E+007	135	15	
21	iris	1	10	weka.classifiers.trees.J48	-C 0.25 -M 2'	-2.18E+017	2.01E+007	135	15	

Advanced Example



Weka Experiment Environment

Experiment Configuration Mode: ○ Simple ● Advanced

Destination: CSVResultListener -O Tut1Exp1.csv

Result generator: CrossValidationResultProducer -X 10 -O splitEvalutorOut.zip -W weka.experiment.ClassifierSplitEvaluator -- -W weka.classifiers.rules.ZeroR -I 0 -C 1 --

Runs: From: 1 To: 10

Distribute experiment: □ Hosts ● By data set ○ By run

Iteration control: ● Data sets first ○ Custom generator first

Datasets: □ Use relative paths

C:\Program Files\Weka-3-6\data\iris.arff

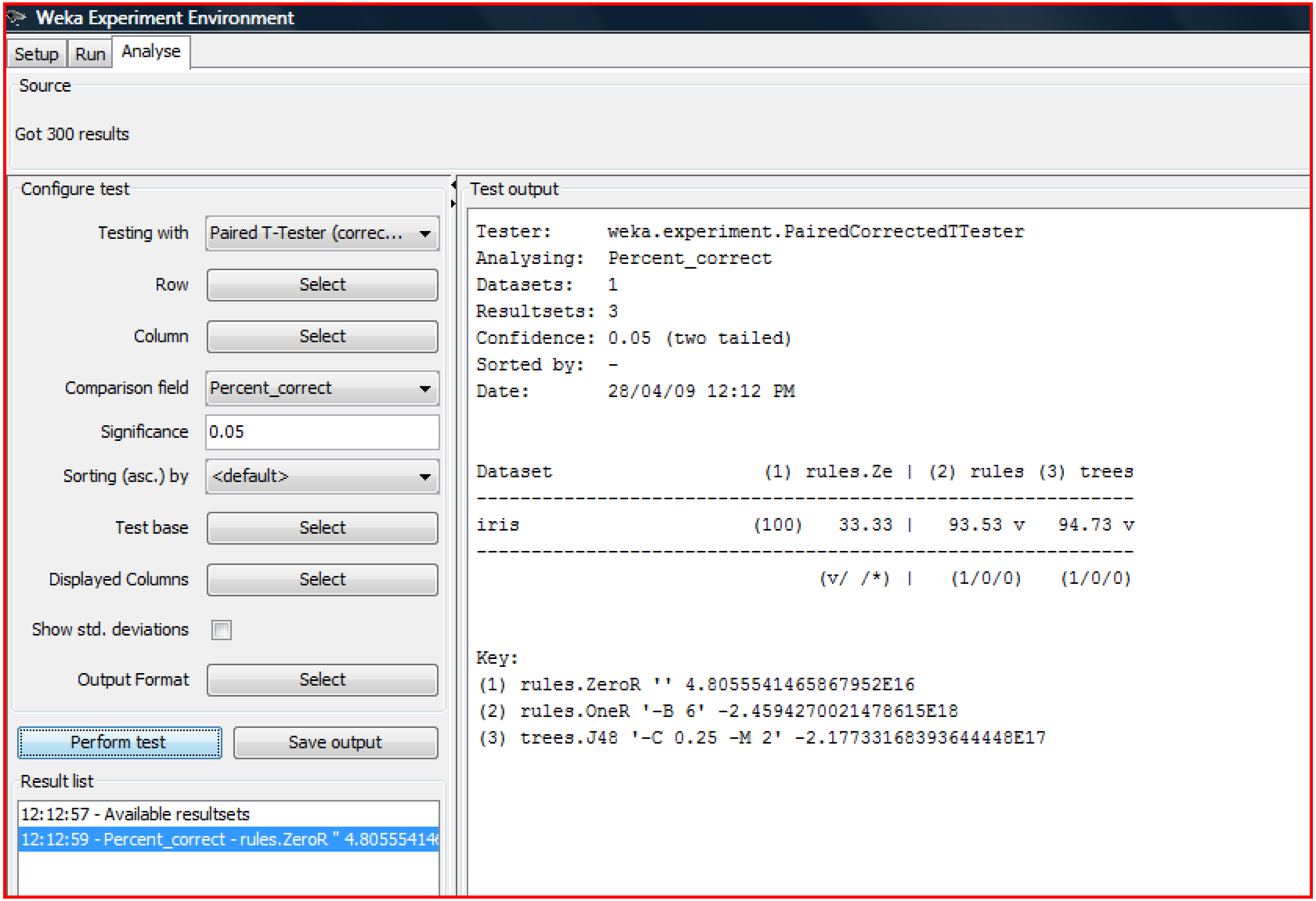
Generator properties: Enabled

J48 -C 0.25 -M 2

ZeroR
OneR -B 6
J48 -C 0.25 -M 2

Multiple Classifiers

Advanced Example



Got 300 results

Testing with: Paired T-Tester (correc...

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

12:12:57 - Available resultsets

12:12:59 - Percent_correct - rules.ZeroR '' 4.80555414

```
Tester:     weka.experiment.PairedCorrectedTTester
Analysing:  Percent_correct
Datasets:   1
Resultsets: 3
Confidence: 0.05 (two tailed)
Sorted by:  -
Date:       28/04/09 12:12 PM


Dataset                   (1) rules.Ze | (2) rules (3) trees
------------------------------------------------------------
iris                     (100)   33.33 |   93.53 v   94.73 v
------------------------------------------------------------
                               (v/ /*) |   (1/0/0)   (1/0/0)


Key:
(1) rules.ZeroR '' 4.8055541465867952E16
(2) rules.OneR '-B 6' -2.4594270021478615E18
(3) trees.J48 '-C 0.25 -M 2' -2.1773316839364448E17
```

RAPID Miner

RapidMiner:: Introduction

A very comprehensive open-source software implementing tools for

intelligent data analysis, data mining, knowledge discovery, machine learning, predictive analytics, forecasting, and analytics in business intelligence (BI).

Is implemented in Java and available under GPL among other licenses

Available from http://rapid-i.com

RapidMiner:: Intro. Contd.

- Is similar in spirit to Weka’s Knowledge flow
- Data mining processes/routines are views as sequential operators
  - Knowledge discovery process are modeled as operator chains/trees
    - Operators define their expected inputs and delivered outputs as well as their parameters
- Has over 400 data mining operators

Rapid Miner

- RapidMiner is an integrated enterprise artificial intelligence framework that offers AI solutions to positively impact businesses.
- It is used as a data science software platform for data extraction, data mining, deep learning, machine learning, and predictive analytics.
- RapidMiner offers a free trial so that users can assess its capabilities.

RapidMiner:: Intro. Contd.

Uses XML for describing operator trees in the KD process

Alternatively can be started through the command line and passed the XML process file

Rapid Miner

Working-Attached with Other doc-How to Work?

Thank you!!!