

# Introduction to Weka

*Daniel Rodríguez, University of Alcalá*



Universidad  
de Alcalá

OXFORD  
BROOKES  
UNIVERSITY

# Weka

Weka Toolkit

- <http://www.cs.waikato.ac.nz/ml/weka/>

Developed in Java and Open source

- GNU Licence

Well supported by the community through mailing lists and Wiki:

- <http://weka.wikispaces.com/>

Also, supported commercially by Pentaho

- <http://weka.pentaho.com/>
- Acknowledgement: J Hernandez & C Ferri (UPV) for some examples

# Weka Toolkit

Composed of several tools

- GUI Chooser
  - Package manager, Visualisation and Viewers
- Explorer (the one to use most of the time)
- Experimenter
- KnowledgeFlow
- SimpleCLI
  - (Command Line Interface) - Obsolete



# ARFF, XRFF, and sparse formats

ARFF (Attribute-Relation File Format) is Weka's native file format

- Composed of Header and Data
  - Header of the ARFF file contains the name of the relation, a list of the attributes (the columns in the data), and their types (numeric, nominal, date).
  - Data is composed of comma separated values after @data

**XRFF** (eXtensible attribute-Relation File Format) is an XML-based extension of the ARFF format. It has the advantage of the following additional capabilities:

- class attribute specification
- attribute weights (So far, only Naïve Bayes can handle attribute weights)  
and instance weights are also supported in ARFF

- It can be compressed using the xrff.gz extension

Both ARFF and XRFF formats support sparse data when most data values are zeros

- (e.g. supermarket dataset)

# ARFF Example

```
@relation weather % Comment  
  
@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
```

ARFF-Viewer - /home/drg/weka/data/weather.arff

File Edit View

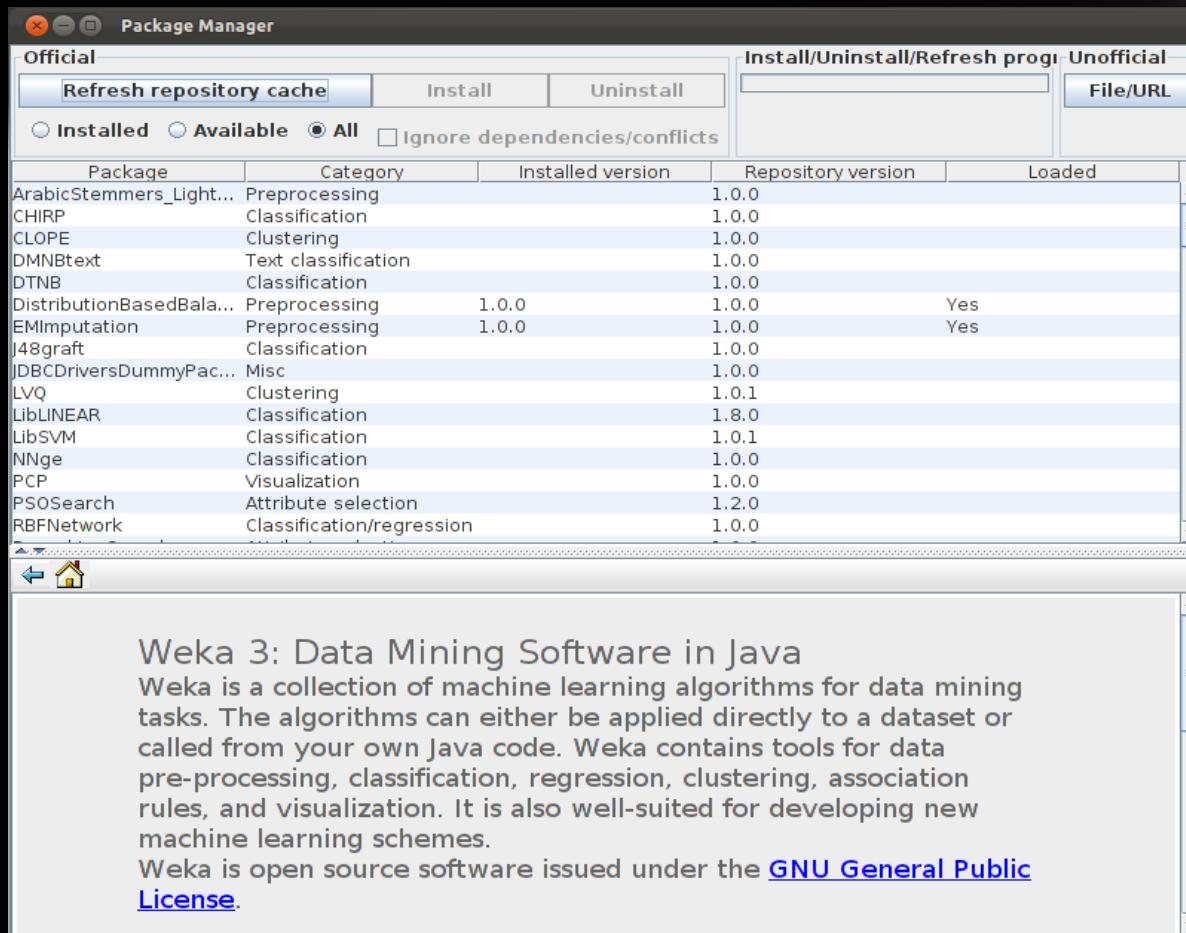
weather.arff

Relation: weather

No.	1: outlook Nominal	2: temperature Numeric	3: humidity Numeric	4: windy Nominal	5: play Nominal
1	sunny	85.0	85.0	FALSE	no
2	sunny	80.0	90.0	TRUE	no
3	overcast	83.0	86.0	FALSE	yes
4	rainy	70.0	96.0	FALSE	yes
5	rainy	68.0	80.0	FALSE	yes
6	rainy	65.0	70.0	TRUE	no
7	overcast	64.0	65.0	TRUE	yes
8	sunny	72.0	95.0	FALSE	no
9	sunny	69.0	70.0	FALSE	yes
10	rainy	75.0	80.0	FALSE	yes
11	sunny	75.0	70.0	TRUE	yes
12	overcast	72.0	90.0	TRUE	yes
13	overcast	81.0	75.0	FALSE	yes
14	rainy	71.0	91.0	TRUE	no

# Package Manager (version > 3.7.2)

Just the most common algorithms are included with the download but others can be installed via the Package Manager.



# Explorer

The one to use most of the time. It can be used to find the most appropriate algorithm and parameters for a given dataset (usually trial & error approach)

Weka Explorer

Preprocess Classify Cluster Associate Select attributes Visualize

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

Filter

Choose None Apply

Current relation

Relation: weather Instances: 14 Attributes: 5 Sum of weights: 14

Attributes

All None Invert Pattern

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

Remove

Selected attribute

Name: outlook Type: Nominal  
Missing: 0 (0%) Distinct: 3 Unique: 0 (0%)

No.	Label	Count	Weight
1	sunny	5	5.0
2	overcast	4	4.0
3	rainy	5	5.0

Class: play (Nom) Visualize All

5 4 5

Remove

Status OK Log x 0

# Explorer's panels

Explorer has 6 panels to analyse and prepare data:

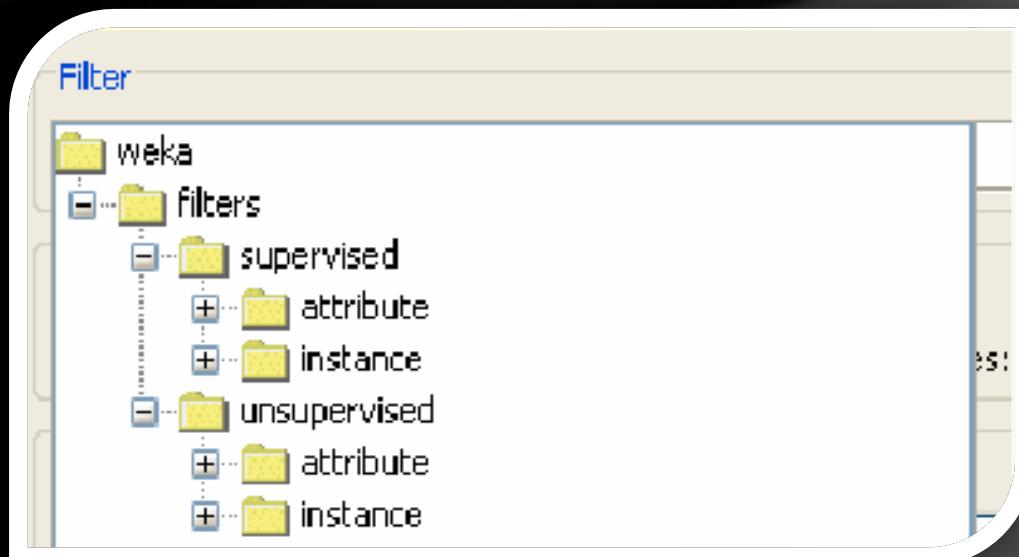
- **Preprocess:** Tools and filters for data manipulation
- **Classification:** Classification and regresión techniques
- **Cluster:** Clustering techniques
- **Associate:** Association techniques
- **Select Attributes:** Permite aplicar diversas técnicas para la reducción del número de atributos
- **Visualize:** Visualisation techniques
- **Other panels** can be added (e.g., Time series analysis)

# Preprocess Panel

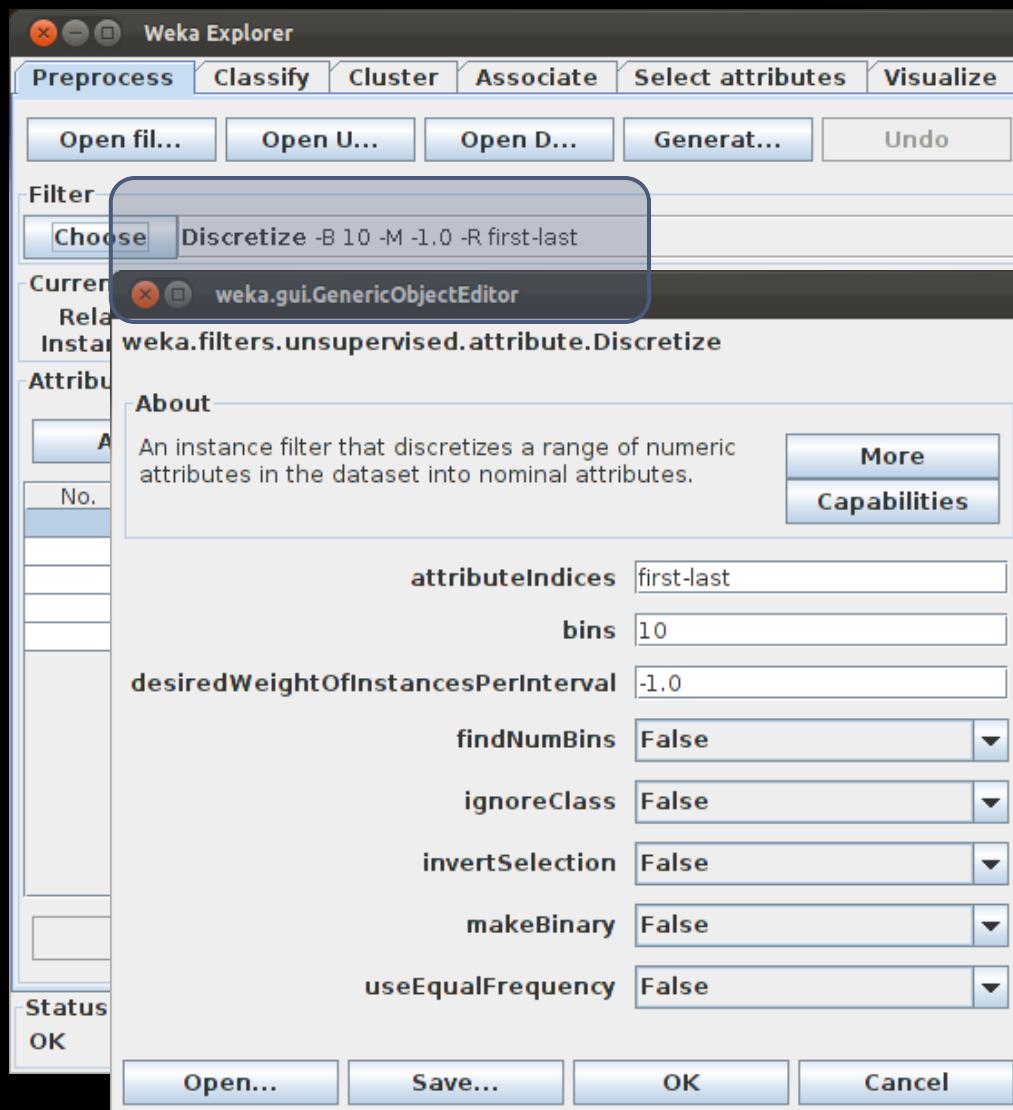
**Filter** algorithms are classified as

- Supervised – if the filter takes into account the class
- Unsuervised – otherwise
- An in turn they are also classified according to whether the filter applies to instances or attributes

Filter examples include discretisation (can be supervised or unsupervised, attribute selection (we will cover this one later in detail), etc.



# Unsupervised Discretize Filter Example



**Information**

**NAME**  
weka.filters.unsupervised.attribute.Discretize

**SYNOPSIS**  
An instance filter that discretizes a range of numeric attributes in the dataset into nominal attributes. Discretization is by simple binning. Skips the class attribute if set.

**OPTIONS**

attributeIndices -- Specify range of attributes to act on. This is a comma separated list of attribute indices, with "first" and "last" valid values. Specify an inclusive range with "-". E.g: "first-3,5,6-10,last".

bins -- Number of bins.

desiredWeightOfInstancesPerInterval -- Sets the desired weight of instances per interval for equal-frequency binning.

findNumBins -- Optimize number of equal-width bins using leave-one-out. Doesn't work for equal-frequency binning.

ignoreClass -- The class index will be unset temporarily before the filter is applied.

invertSelection -- Set attribute selection mode. If false, only selected (numeric) attributes in the range will be discretized; if true, only non-selected attributes will be discretized.

makeBinary -- Make resulting attributes binary.

useEqualFrequency -- If set to true, equal-frequency binning will be used instead of equal-width binning.

# Discretize Supervised Filter Example

**Weka Explorer**

Preprocess Classify Cluster Associate Select attribute

Open fi... Open U... Open D... Gener...

**weka.gui.GenericObjectEditor**

**weka.filters.supervised.attribute.Discretize**

**About**

An instance filter that discretizes a range of numeric attributes in the dataset into nominal attributes.

**attributeIndices** first-last

**invertSelection** False

**makeBinary** False

**useBetterEncoding** False

**useKononenko** False

Open... Save... OK Cancel

**Status**  
OK

**Information**

NAME  
weka.filters.supervised.attribute.Discretize

SYNOPSIS  
An instance filter that discretizes a range of numeric attributes in the dataset into nominal attributes. Discretization is by Fayyad & Irani's MDL method (the default).

For more information, see:

Usama M. Fayyad, Keki B. Irani: Multi-interval discretization of continuousvalued attributes for classification learning. In: Thirteenth International Joint Conference on Artificial Intelligence, 1022-1027, 1993.

Igor Kononenko: On Biases in Estimating Multi-Valued Attributes. In: 14th International Joint Conference on Artificial Intelligence, 1034-1040, 1995.

OPTIONS

attributeIndices -- Specify range of attributes to act on. This is a comma separated list of attribute indices, with "first" and "last" valid values. Specify an inclusive range with "-". E.g: "first-3,5,6-10,last".

invertSelection -- Set attribute selection mode. If false, only selected (numeric) attributes in the range will be discretized; if true, only non-selected attributes will be discretized.

makeBinary -- Make resulting attributes binary.

useBetterEncoding -- Uses a more efficient split point encoding.

useKononenko -- Use Kononenko's MDL criterion. If set to false uses the Fayyad & Irani criterion.

# 'Classify' Panel

Classification and regression Techniques:

Decission Trees: ID3, C4.5 (J48), ...

Regresssion Trees: LMT (M5), ...

Rules: PART, CN2, ...

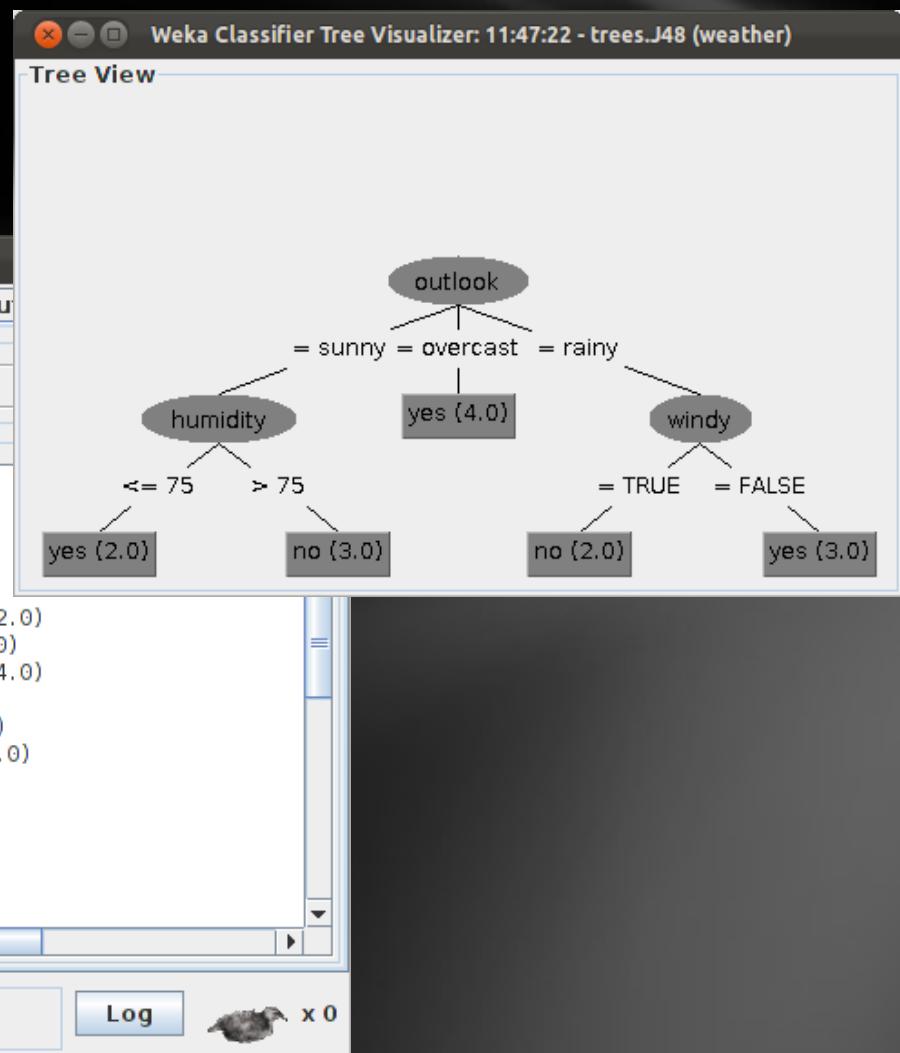
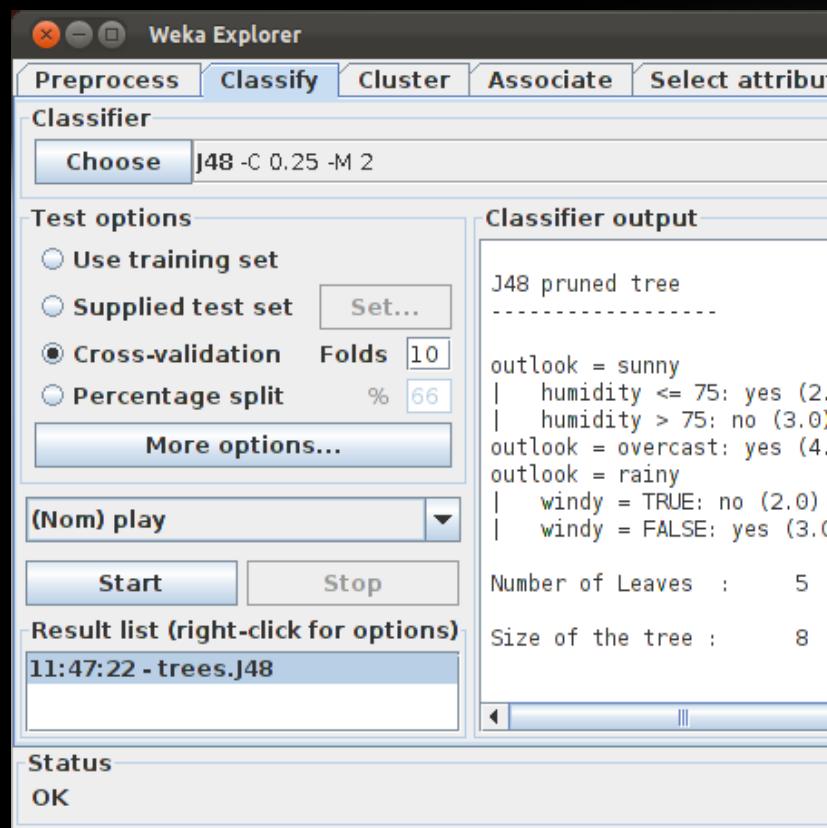
Functions: Regression, Neural Networks, logistic regression, Support Vector Machines (SMO), ...

Lazy Techniques: IB1, IBK, ...

Bayesian Techniques: Naive Bayes

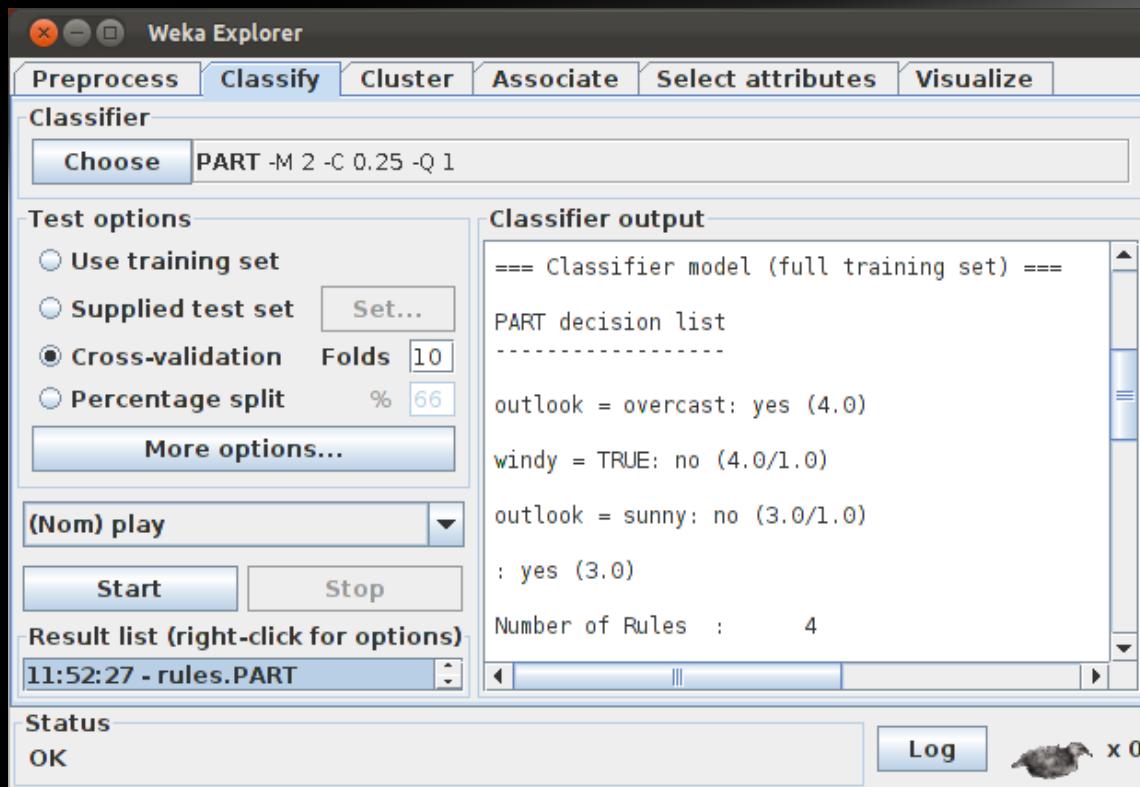
Meta-techniques

# 'Classify' Panel – Trees (C4.5 – J48)



# 'Classify' Panel – Rules

We can generate rules from trees or there are other algorithms to generate rules directly



# 'Classify Panel – Regression trees (M5P)

The image shows two windows from the Weka interface illustrating the M5P regression tree model.

**Weka Explorer Window:**

- Panel A:** Shows a data table with columns A, B, and C. Row 1 is selected, showing values 1, 48000, yes.
- Panel B:** Shows the M5P - M 4.0 configuration. It includes:
  - Training set
  - Test set
  - Validation
  - Leave one out split
  - More options...
  - Folds: 10
  - %: 66
- Classifier output:** Displays the M5 pruned model tree and the corresponding linear models (LM1 and LM2) for PhD=yes and PhD>yes.
- Number of Rules:** 2

**Weka Explorer: Visualizing wishfulthinking2 Window:**

- X:** YoExp (Num)
- Y:** Salary (Num)
- Colour:** PhD (Nom)
- Plot:** wishfulthinking2
- Class colour:** yes (blue), no (red)
- Tree View:** Shows the regression tree structure:

```
graph TD; PhD[PhD=yes] --> LM1["LM 1 (10/3.179%)"]; PhD --> LM2["LM 2 (10/11.537%)"]
```

# Lazy Techniques

Lazy techniques (also known as Collaborative Filtering or Instance-based Learning) do not build models, just retain instances.

Once a new instance needs to be classified, these techniques search for 'similar' instances in the repository

**Weka's IBk** implements k-NN.

- Weka's IBk uses the Euclidean distance by default.
- The sum of the squared differences between normalized attribute values is computed; this is then normalized by the number of attributes in the data; finally the square root is taken.
- Following this, weighting can be applied to the distances (if selected).
  - Normalising distances for all attributes so that attributes have the same impact on the distance function.
- It may return  $k$  neighbours. If there are ties in the distance, neighbours are voted to form the final classification.

# IBk

Weka Explorer

Preprocess Classify Clusters

Classifier

Choose IBk -K 1 -W 0 -A "weka.classifiers.lazy.IBk"

Test options

Use training set

Supplied test set Set

Cross-validation Folds 1

Percentage split %

More options...

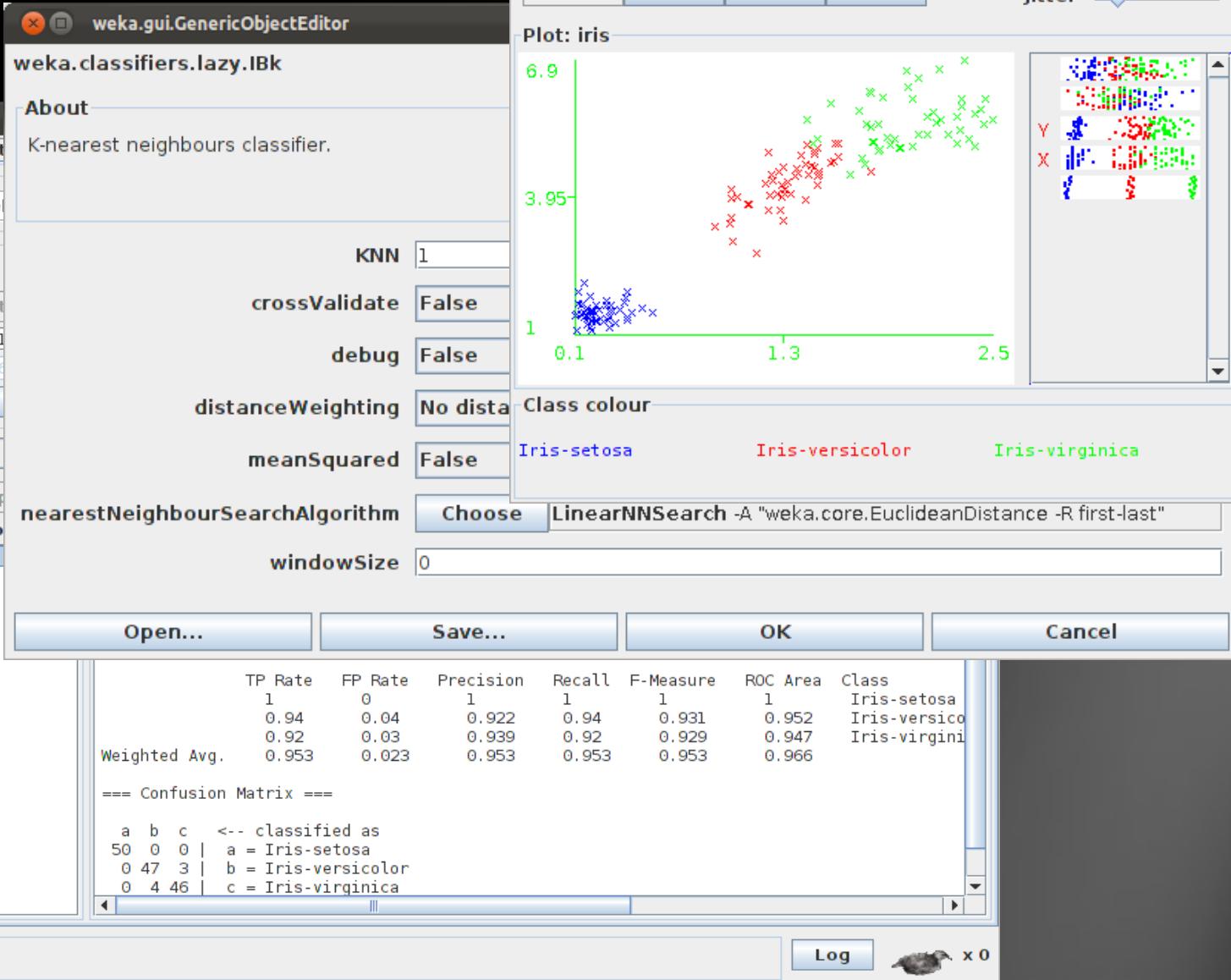
(Nom) class

Start Stop

Result list (right-click for options)

13:45:44 - lazy.IBk

Status OK



# Cluster Tab

**Weka Explorer**

Preprocess Classify Cluster **Cluster** Associate Select attributes Visualize

**Clusterer**

Choose SimpleKMeans -N 3 -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)**

15:13:30 - SimpleKMeans

**Clusterer output**

kMeans  
=====

Number of iterations: 3  
Within cluster sum of squared errors: 7.817456892309574  
Missing values globally replaced with mean/mode

Cluster centroids:

Attribute	Full Data (150)	Cluster# 0 (50)	1 (50)	2 (50)
sepallength	5.8433	5.936	5.006	6.588
sepalwidth	3.054	2.77	3.418	2.974
petallength	3.7587	4.26	1.464	5.552
petalwidth	1.1987	1.326	0.244	2.026
class	Iris-setosa Iris-versicolor	Iris-setosa	Iris-setosa	Iris-virginica

Time taken to build model (full training data) : 0.01 seconds

== Model and evaluation on training set ==

Clustered Instances

0	50 ( 33%)
1	50 ( 33%)
2	50 ( 33%)

Status OK Log x 0

# Association

Weka Explorer

Preprocess Classify Cluster Associate **Select attributes** Visualize

Associator

Choose Apriori -N 10 -T 0 -C 0.9 -D 0.05 -U 1.0 -M 0.1 -S -1.0 <-1

Start Stop

Result list (right-click to edit)

15:16:18 - Apriori

Apriori

=====

Minimum support: 0.15 (694 instances)  
Minimum metric <confidence>: 0.9  
Number of cycles performed: 17

Generated sets of large itemsets:

Size of set of large itemsets L(1): 44  
Size of set of large itemsets L(2): 380  
Size of set of large itemsets L(3): 910  
Size of set of large itemsets L(4): 633  
Size of set of large itemsets L(5): 105  
Size of set of large itemsets L(6): 1

Best rules found:

1. biscuits=t frozen foods=t fruit=t total=high 788 => bread and cake=t 723 <conf: (0.92)> lift: (1.27) lev: (0.03) [155] conv: (3.35)
2. baking needs=t biscuits=t fruit=t total=high 760 => bread and cake=t 696 <conf: (0.92)> lift: (1.27) lev: (0.03) [149] conv: (3.28)
3. baking needs=t frozen foods=t fruit=t total=high 770 => bread and cake=t 705 <conf: (0.92)> lift: (1.27) lev: (0.03) [150] conv: (3.27)
4. biscuits=t fruit=t vegetables=t total=high 815 => bread and cake=t 746 <conf: (0.92)> lift: (1.27) lev: (0.03) [159] conv: (3.26)
5. party snack foods=t fruit=t total=high 854 => bread and cake=t 779 <conf: (0.91)> lift: (1.27) lev: (0.04) [164] conv: (3.15)
6. biscuits=t frozen foods=t vegetables=t total=high 797 => bread and cake=t 725 <conf: (0.91)> lift: (1.26) lev: (0.03) [151] conv: (3.06)
7. baking needs=t biscuits=t vegetables=t total=high 772 => bread and cake=t 701 <conf: (0.91)> lift: (1.26) lev: (0.03) [145] conv: (3.01)
8. biscuits=t fruit=t total=high 954 => bread and cake=t 866 <conf: (0.91)> lift: (1.26) lev: (0.04) [179] conv: (3)
9. frozen foods=t fruit=t vegetables=t total=high 834 => bread and cake=t 757 <conf: (0.91)> lift: (1.26) lev: (0.03) [156] conv: (3)
10. frozen foods=t fruit=t total=high 969 => bread and cake=t 877 <conf: (0.91)> lift: (1.26) lev: (0.04) [179] conv: (2.92)

Status OK Log x 0

# Association

Weka Explorer

Preprocess Classify Cluster Associate **Select attributes** Visualize

Associator

Choose Apriori -N 10 -T 0 -C 0.9 -D 0.05 -U 1.0 -M 0.1 -S -1.0 <-1

Start Stop

Result list (right-click to edit)

15:16:18 - Apriori

Apriori

=====

Minimum support: 0.15 (694 instances)  
Minimum metric <confidence>: 0.9  
Number of cycles performed: 17

Generated sets of large itemsets:

Size of set of large itemsets L(1): 44  
Size of set of large itemsets L(2): 380  
Size of set of large itemsets L(3): 910  
Size of set of large itemsets L(4): 633  
Size of set of large itemsets L(5): 105  
Size of set of large itemsets L(6): 1

Best rules found:

1. biscuits=t frozen foods=t fruit=t total=high 788 => bread and cake=t 723 <conf: (0.92)> lift: (1.27) lev: (0.03) [155] conv: (3.35)
2. baking needs=t biscuits=t fruit=t total=high 760 => bread and cake=t 696 <conf: (0.92)> lift: (1.27) lev: (0.03) [149] conv: (3.28)
3. baking needs=t frozen foods=t fruit=t total=high 770 => bread and cake=t 705 <conf: (0.92)> lift: (1.27) lev: (0.03) [150] conv: (3.27)
4. biscuits=t fruit=t vegetables=t total=high 815 => bread and cake=t 746 <conf: (0.92)> lift: (1.27) lev: (0.03) [159] conv: (3.26)
5. party snack foods=t fruit=t total=high 854 => bread and cake=t 779 <conf: (0.91)> lift: (1.27) lev: (0.04) [164] conv: (3.15)
6. biscuits=t frozen foods=t vegetables=t total=high 797 => bread and cake=t 725 <conf: (0.91)> lift: (1.26) lev: (0.03) [151] conv: (3.06)
7. baking needs=t biscuits=t vegetables=t total=high 772 => bread and cake=t 701 <conf: (0.91)> lift: (1.26) lev: (0.03) [145] conv: (3.01)
8. biscuits=t fruit=t total=high 954 => bread and cake=t 866 <conf: (0.91)> lift: (1.26) lev: (0.04) [179] conv: (3)
9. frozen foods=t fruit=t vegetables=t total=high 834 => bread and cake=t 757 <conf: (0.91)> lift: (1.26) lev: (0.03) [156] conv: (3)
10. frozen foods=t fruit=t total=high 969 => bread and cake=t 877 <conf: (0.91)> lift: (1.26) lev: (0.04) [179] conv: (2.92)

Status OK Log x 0

# Attribute Selection

Feature selection is important in different ways:

- A reduced volume of data allows different data mining or searching techniques to be applied.
- Irrelevant and redundant attributes can generate less accurate and more complex models. Furthermore, data mining algorithms can be executed faster.
- We can avoid the collection of data for those irrelevant and redundant attributes in the future.

There exist two major approach in feature selection from the method's output point of view depending on the way that features are evaluated:

- **Feature ranking**, a.k.a. feature weighting, assesses individual features and assigns them weights according to their degrees of relevance
- **Feature subset selection (FSS)** evaluate the goodness of each found feature subset (Unusually, some search strategies in combination with subset evaluation can provide a ranked list).

# Feature Ranking

Feature ranking algorithms category, one can expect a ranked list of features which are ordered according to evaluation measures

Each attribute correlation with the class is evaluated independently of other attributes according to a statistical test (e.g.:  $\chi$ -squared, Infogain, etc.)

- A subset of features is often selected from the top of a ranking list. A feature is good and thus will be selected if its weight of relevance is greater than a user-specified threshold value,
- or we can simply select the first  $k$  features from the ranked list.

This approach is efficient for high-dimensional data due to its linear time complexity in terms of dimensionality

Very fast method.

However, it cannot detect redundant attributes

# Feature Subset Selection (FSS)

In the FSS category, candidate subsets are generated based on a certain search strategy:

- **Exhaustive, heuristic and random** search

Each candidate subset is evaluated by a certain **evaluation measure**. If a new subset turns out to be better, it replaces the previous best subset. The process of subset generation and evaluation is repeated until a given stopping criterion is satisfied. Existing evaluation measures include:

- Consistency measure attempts to find a minimum number of features that separate classes as consistently as the full set of features can. An inconsistency is defined as to instances having the same feature values but different class labels.
- Correlation measure evaluates the goodness of feature subsets based on the hypothesis that good feature subsets contain features highly correlated to the class, yet uncorrelated to each other.
- Accuracy of a learning algorithm (Wrapper-based attribute selection) uses the target learning algorithm to estimate the worth of attribute subsets. The feature subset selection algorithm conducts a search for a good subset using the induction algorithm itself as part of the evaluation function.

The time complexity in terms of data dimensionality is, exponential for exhaustive search, quadratic for heuristic search and linear to the number of iterations in a random search

# FSS – CFS (Correlation Based FS)

Correlation measure is applied in an algorithm called CFS that exploit heuristic search (best first) to search for candidate feature subsets.

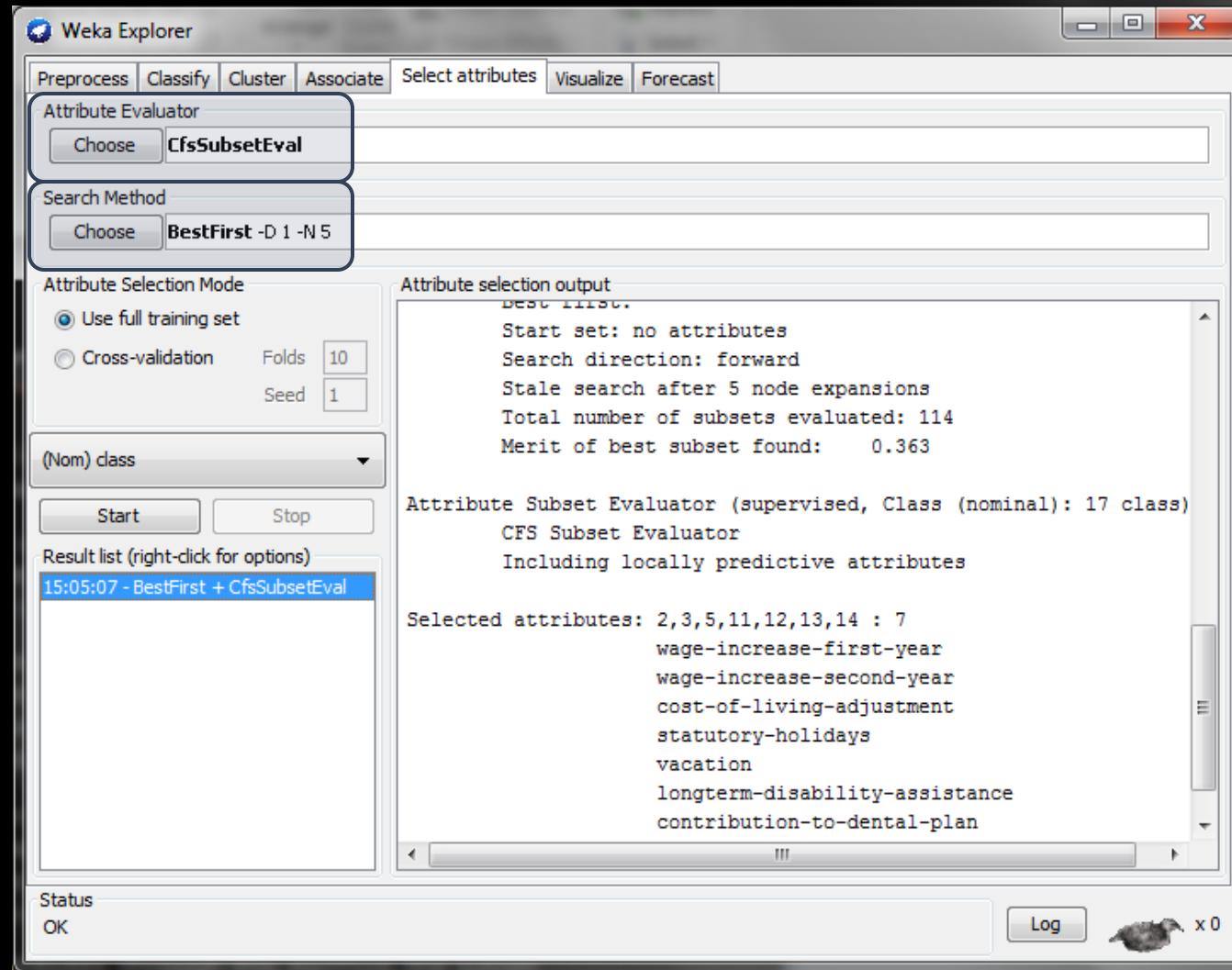
- One of the most frequently used search techniques is hill-climbing (greedy). It starts with an empty set and evaluates each attribute individually to find the best single attribute.

It then tries each of the remaining attributes in conjunction with the best to find the most suited pair of attributes. In the next iteration, each of the remaining attributes are tried in conjunction with the best pair to find the most suited group of three attributes.

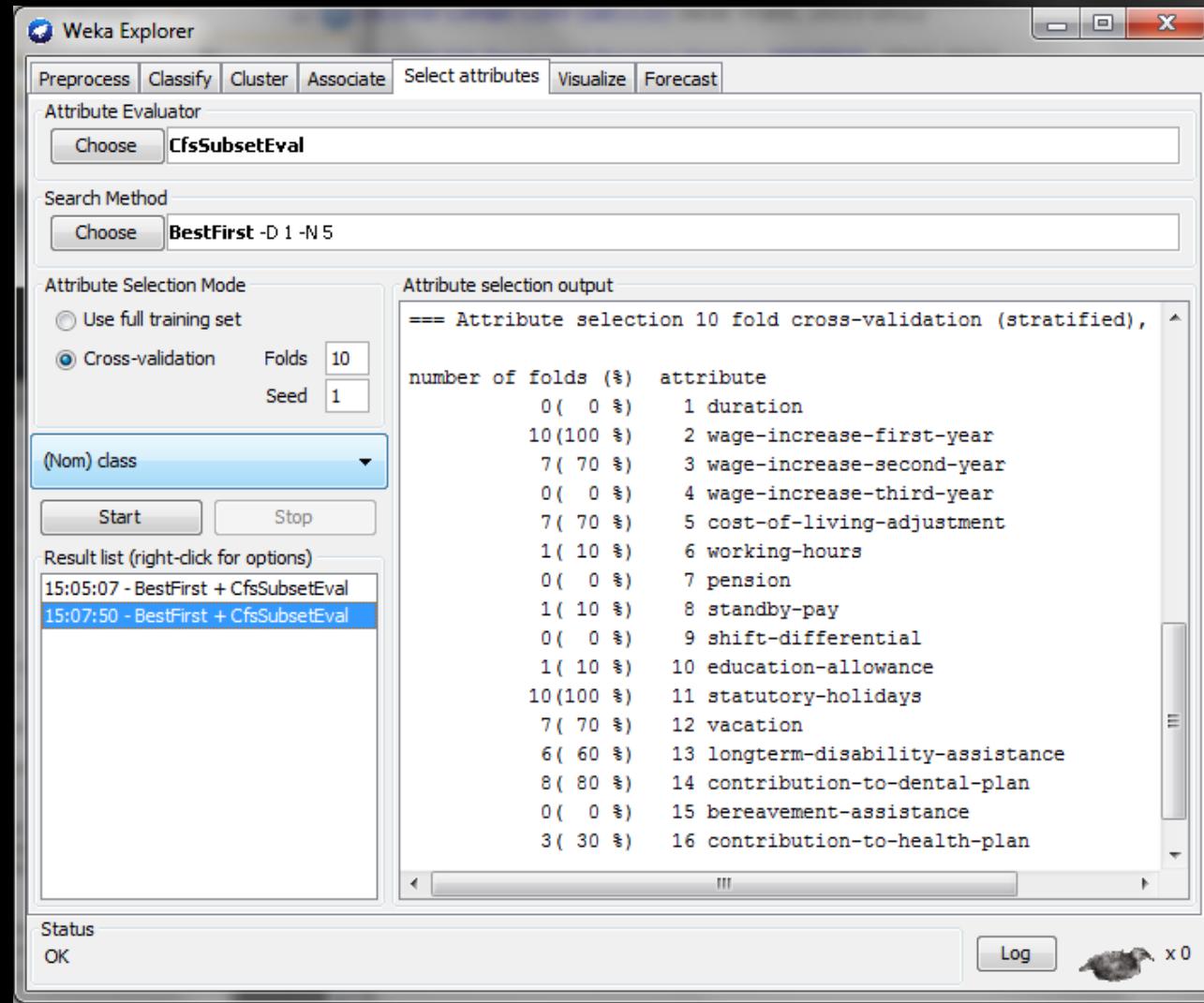
This process continues until no single attribute addition improves the evaluation of the subset; i.e., subset evaluator is run  $M$  times to choose the best single attribute,  $M-1$  times to find the best pair of attributes,  $M-2$  times the best group of three, and so on.

- E.g., if we have chosen 5 attributes through this method, the subset evaluator has been run  $M+(M-1)+(M-2)+(M-3)+(M-4)$  times.

# CFS Example



# CFS – 10 CV



# Exercise - Hay fever

Find the best possible model to recommend the type of drug for hay fever depending on the patient. The attributes collected from historical patients include:

- Age
- Sex
- BP Blood Pressure
- Cholesterol level
- Na: Blood sodium level
- K: Blood potassium level
- There are 5 possible drugs: DrugA, DrugB, DrugC, DrugX, DrugY

The dataset can be downloaded from:

- <http://www.cc.uah.es/drg/courses/datamining/datasets.zip>

Hints:

- Try C4.5, can you simplify the generated tree combining attributes?
- Naive Bayes: can you improve the 'default' results with Feature Selection?

# Exercise – Cost Sensitive Example

Using the German credit dataset:

<http://www.cc.uah.es/drg/courses/datamining/datasets.zip>

This dataset is composed of 20 attributes (7 numeric and 13 nominal) of clients of a bank requesting a credit.

Please, check for details of the in the file itself, including the provided cost matrix:

		actual	
		good	bad
	good	0	1
	bad	5	0

This cost matrix means that it is 5 times more costly to give credit to a person that won't pay back than the other way around.

Check what happens with ZeroR, Naïve Bayes, Bagging

# Association Rules and dependencies

The Titanic dataset (titanic.arff) is composed of 4 attributes

- Class (1st, 2nd, 3rd)
- Age (adult, child)
- Sex (male, female)
- Survived (yes, no)

describing the actual characteristics of the 2,201 passengers of The Titanic  
("Report on the Loss of the 'Titanic' (S.S.)" (1990), British Board of Trade  
Inquiry Report\_ (reprint), Gloucester, UK: Allan Sutton Publishing)

Using the Apriori algorithm extract information contained in this dataset

- Modify the default parameters to obtain rules that consider 'child' in the Age attribute. As there is a small number of values containing samples of age = 0, those rules are filtered out with due to their low coverage

# Exercise – Clustering

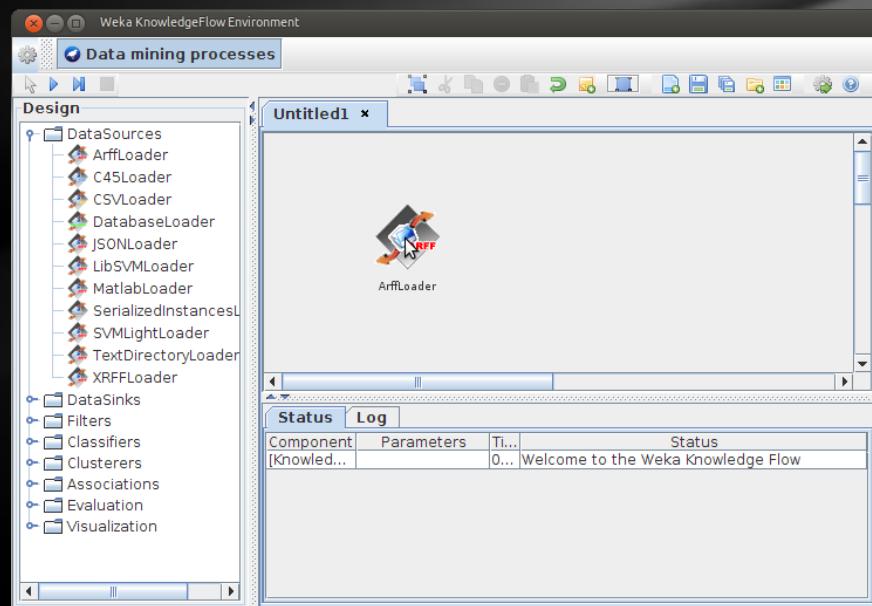
Using the employees dataset:

<http://www.cc.uah.es/drg/courses/datamining/datasets.zip>

Checks the results for k Means usign 3 clusters.

# KnowlegeFlow

Visually



## Example of Running J4.8 (C4.5) using KnowledgeFlow

The screenshot shows the Weka KnowledgeFlow Environment interface. The main window displays a flow diagram titled "Untitled1" for running a J4.8 classifier. The flow consists of the following components and connections:

- ArffLoader** (ARFF input) connects to **Class Assigner**.
- Class Assigner** connects to **J48**.
- J48** connects to **tree**.
- tree** connects to **CrossValidation**.
- CrossValidation** has two outputs: **trainingSet** and **testSet**.
- trainingSet** connects to **J48**.
- testSet** connects to **Classifier**.
- Classifier** connects to **Classifier Performance Evaluator**.
- Classifier Performance Evaluator** connects to **TextViewer**.

The **Text Viewer** window on the left shows the execution log and results for the J4.8 run. Key output details include:

- Evaluation result**:
  - Scheme: J48
  - Options: -C 0.25 -M 2
  - Relation: CML
- Correctly Classified Instances**: 438 (87.9518 %)
- Incorrectly Classified Instances**: 60 (12.0482 %)
- Kappa statistic**: 0.0424
- Mean absolute error**: 0.1757
- Root mean squared error**: 0.3301
- Relative absolute error**: 98.2132 %
- Root relative squared error**: 110.822 %
- Coverage of cases (0.95 level)**: 95.7831 %
- Mean rel. region size (0.95 level)**: 83.1325 %
- Total Number of Instances**: 498

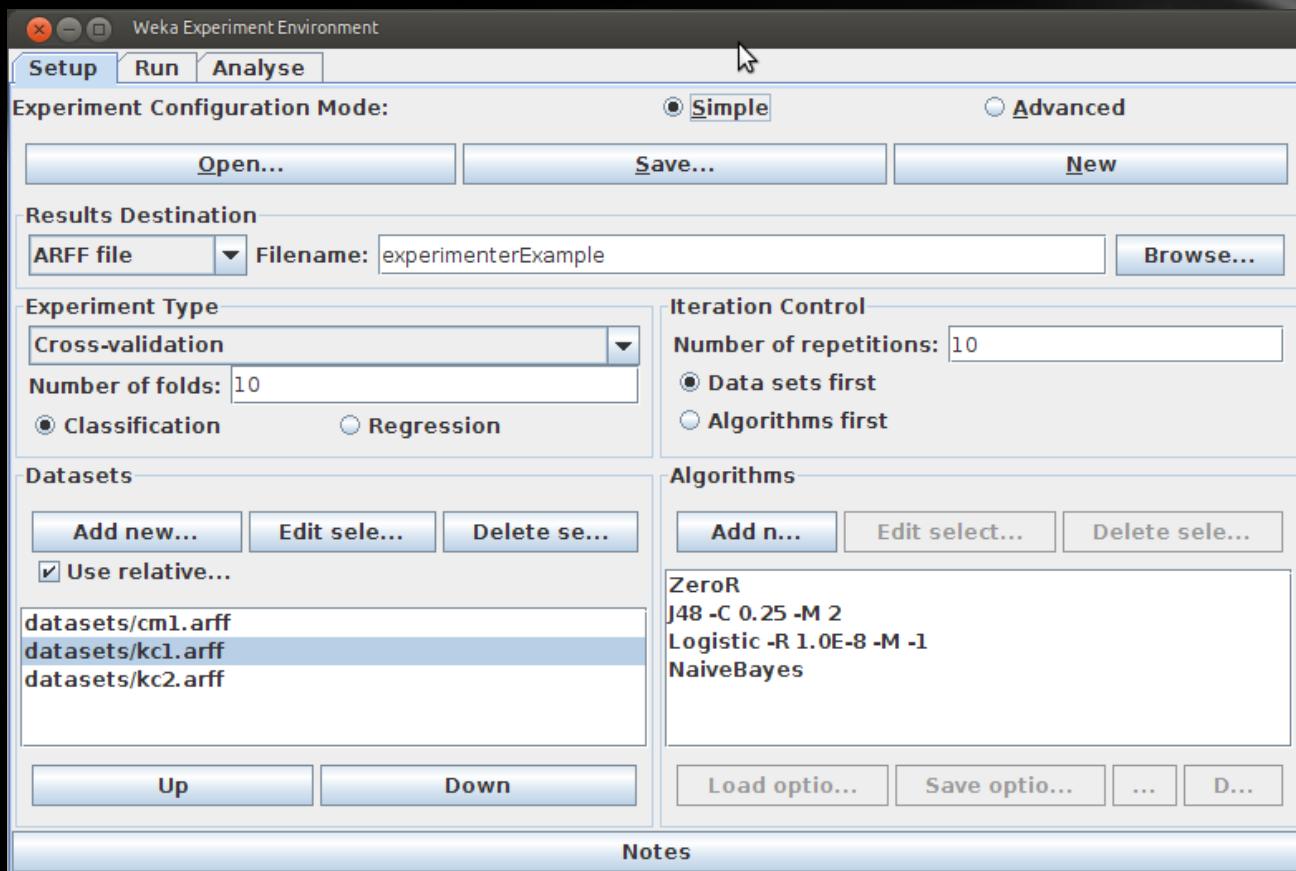
The **Log** table at the bottom shows the status of various components:

Component	Parameters	Status
Arff...	-O:... OK.	OK
Class...	-	Finished.
CV...	-	Finished.
J48...	-C 0.25 -M 2	Finished.
Perf...	-	Finished.

# Experimenter

Allow us to compare multiple algorithms

Typically, 10 times 10-CV



# Experimenter – Output

After Running the experiment (Run tab) and loading the generated file (Analyze tab)

Weka Experiment Environment

Setup Run Analyse **Analyse**

Source

Got 1200 results

File... Database... Experiment

Configure test

Testing with Paired T-Tester (corrected)

Select rows and cols Ro... Cols Sw...

Comparison field Area\_under\_ROC

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

21:06:31 - Area\_under\_ROC - rules.ZeroR '' 48055541  
21:06:40 - Available resultsets

Test output

Tester: weka.experiment.PairedCorrectedTTester  
Analysing: Area\_under\_ROC  
Datasets: 3  
Resultsets: 4  
Confidence: 0.05 (two tailed)  
Sorted by: -  
Date: 02/05/12 21:06

Dataset (1) rules.ZeroR | (2) trees.J48 (3) functions. (4) bayes.Naiv

	(1) rules.ZeroR	(2) trees.J48	(3) functions.	(4) bayes.Naiv
CM1	(100) 0.50(0.00)	0.56(0.13)	0.78(0.11) v	0.74(0.10) v
KC1	(100) 0.50(0.00)	0.70(0.07) v	0.80(0.04) v	0.79(0.04) v
KC2	(100) 0.50(0.00)	0.69(0.10) v	0.82(0.08) v	0.84(0.06) v

(v/ /\*) | (2/1/0) (3/0/0) (3/0/0)

Key:

(1) rules.ZeroR '' 48055541465867954  
(2) trees.J48 '-C 0.25 -M 2' -217733168393644444  
(3) functions.Logistic '-R 1.0E-8 -M -1' 3932117032546553727  
(4) bayes.NaiveBayes '' 5995231201785697655