Data Mining

Module:
Data Mining with Weka

Last updated 9/20/2020

Overview of DM with Weka Module

- Weka-1: Weka Basics
- Weka-2: Preprocessing
- Weka-3: Decision Trees
- Weka-4: Other Classifiers & Feature Selection
- Weka-5: KnowledgeFlow Interface

Weka-1: Weka Basics

- What is Weka?
- Why use Weka (and why not)
- Weka versions and downloading the software
- Weka ARFF file format
- Starting Weka and the 5 Weka Interfaces

What is Weka?

- Weka is a data mining suite developed at University of Waikato
- Weka stands for Waikato Environment for Knowledge Analysis
- Weka includes everything necessary to generate and apply data mining models
  - Covers all major data mining tasks
  - Includes tools to preprocess and visualize data
  - Includes multiple (5) interfaces
    - We will focus on the explorer interface
    - Briefly discuss the knowledgeflow
    - Does not require any programming

WEKA is also a bird

Copyright: Martin Kramer (mkramer@wxs.nl)

Why Weka?

- There are many options on what data mining suite or toolkit one can use
- Weka has the following advantages:
  - Easy to learn
  - Free and open-source
  - Easy to download and runs on many platforms
  - Does not require programming knowledge
    - This is important since a few students may not have much programming experience

Weka Disadvantages

- Weka is not as flexible as other tools that are programming based
  - If you are programming anyway you can combine capabilities more flexibly and write code to do things not supported by Weka
  - Tools based on Python or R have large ecosystems

DM Tools for This Course

- For your project you may use Weka, Python, or even another data mining toolkit
- I do provide snippets of Python code and Weka examples in some lectures
  - But everyone should learn WEKA and complete the tutorial/exercises in this module
- We will not cover the details of how to use Python in this course

WEKA: the software

- Data mining software written in Java
  - distributed under the GNU Public License
- Used for research, education, and applications
- Main features:
  - Comprehensive set of data pre-processing tools, learning algorithms and evaluation methods
  - Multiple interfaces that do not require programming
  - Experimenter can compare learning algorithms

Weka: Versions

- There are several versions of WEKA:
  - As of Feb 2020 the stable version is 3.8.4 and that is the one you should be using
- Many of these slides are based on older version and will look different from what you see
  - Dr. Weiss has added notes for significant differences
- Class WEKA page provides relevant info and links
  - https://storm.cis.fordham.edu/~gweiss/data-mining/weka.html
- Can also use following link to download software and documentation
  - https://www.cs.waikato.ac.nz/ml/weka/

Downloading Weka

- Weka runs on all major platforms
- It is free and easy to download
- Download Weka from here:
  - https://waikato.github.io/weka-wiki/downloading_weka/
  - Use the latest stable version (3.8.4 as of Aug 2020)
- There is plenty of documentation
  - https://waikato.github.io/weka-wiki/documentation/

Weka ARFF File Format

- ARFF= Attribute Relation File Format
- Weka ARFF files include two main parts:
  - Specification of the features
  - The actual data
- Files are “flat files” usually comma separated
- Other tools use a separate file for each part
  - C4.5 decision tree tool, which was once very popular, had a “names” and “data” file
  - I find the single file format odd, since specification is short but actual data can be millions of lines

Weka ARFF Example

This just defines dataset name

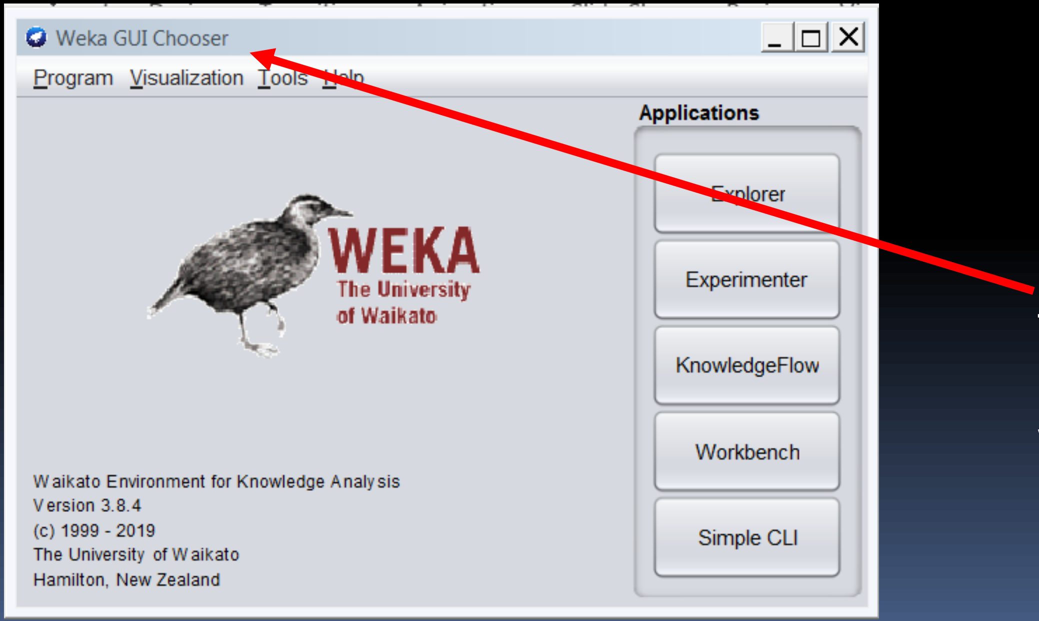
```
@relation heart-disease-simplified
```

Categorical feature must list values

```
@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
...
```

Starting Weka

- Run Weka 3.8.4 (with console)
  - Different version if a new stable version released
  - May want to create a shortcut on your desktop



Waikato Environment for Knowledge Analysis
Version 3.8.4
(c) 1999 - 2019
The University of Waikato
Hamilton, New Zealand

This is the GUI Chooser Window

The Weka Interfaces

- Explorer:
  - We cover in detail
  - Enables you to apply multiple actions but does not explicitly model them
- Experimenter
  - Run many experiments in controlled manner and compare results
- KnowledgeFlow:
  - Like Explorer but each step is represented as a node in a graph, so flow explicitly represented
- Workbench
  - Combines all GUI interfaces into one
- Simple CLI (command line interface)
  - No GUI so can use shell scripts to control experiments
  - Similar to using Python where each command is a function

Weka-1 Review: Weka Basics

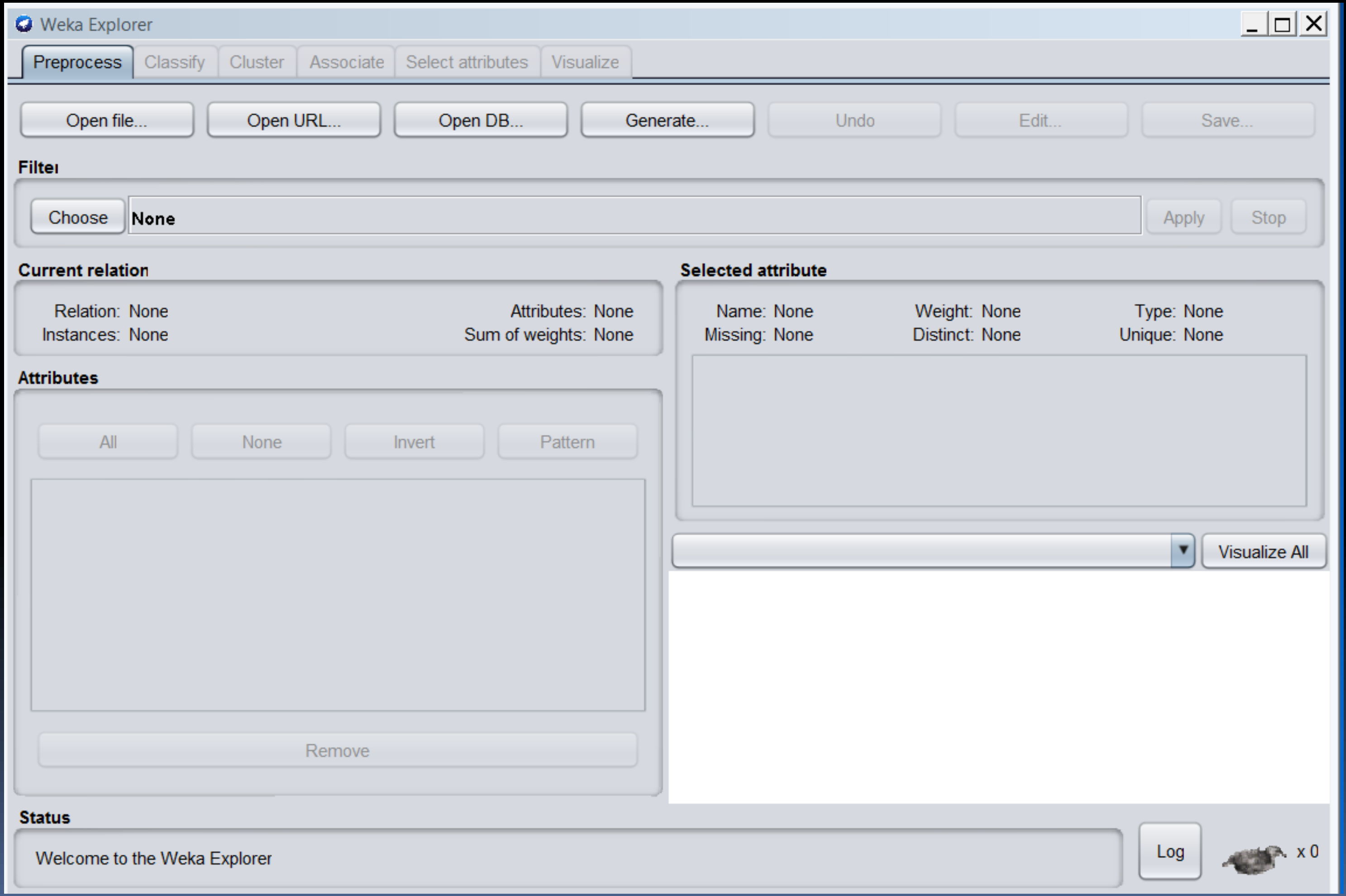
- You should know the following:
  - What Weka is
  - The benefits & drawbacks of WEKA
  - How to download WEKA
    - Ideally you should have installed it
  - The ARFF format and understand it
  - The 5 WEKA interfaces and what they are used for

Weka-2: Preprocessing

- How to import data
- Visualizing feature value distribution and relationships to class values
- Preprocessing filters
  - Example using “Discretize” filter
- It is strongly recommended that you follow along on your own installation of Weka
  - Reproduce each step on your computer

Start the Explorer

Click on "Explorer " Button in the GUI Chooser Window



Filter: None

Relation: None
Instances: None
Attributes: None
Sum of weights: None

Name: None
Missing: None
Weight: None
Distinct: None
Type: None
Unique: None

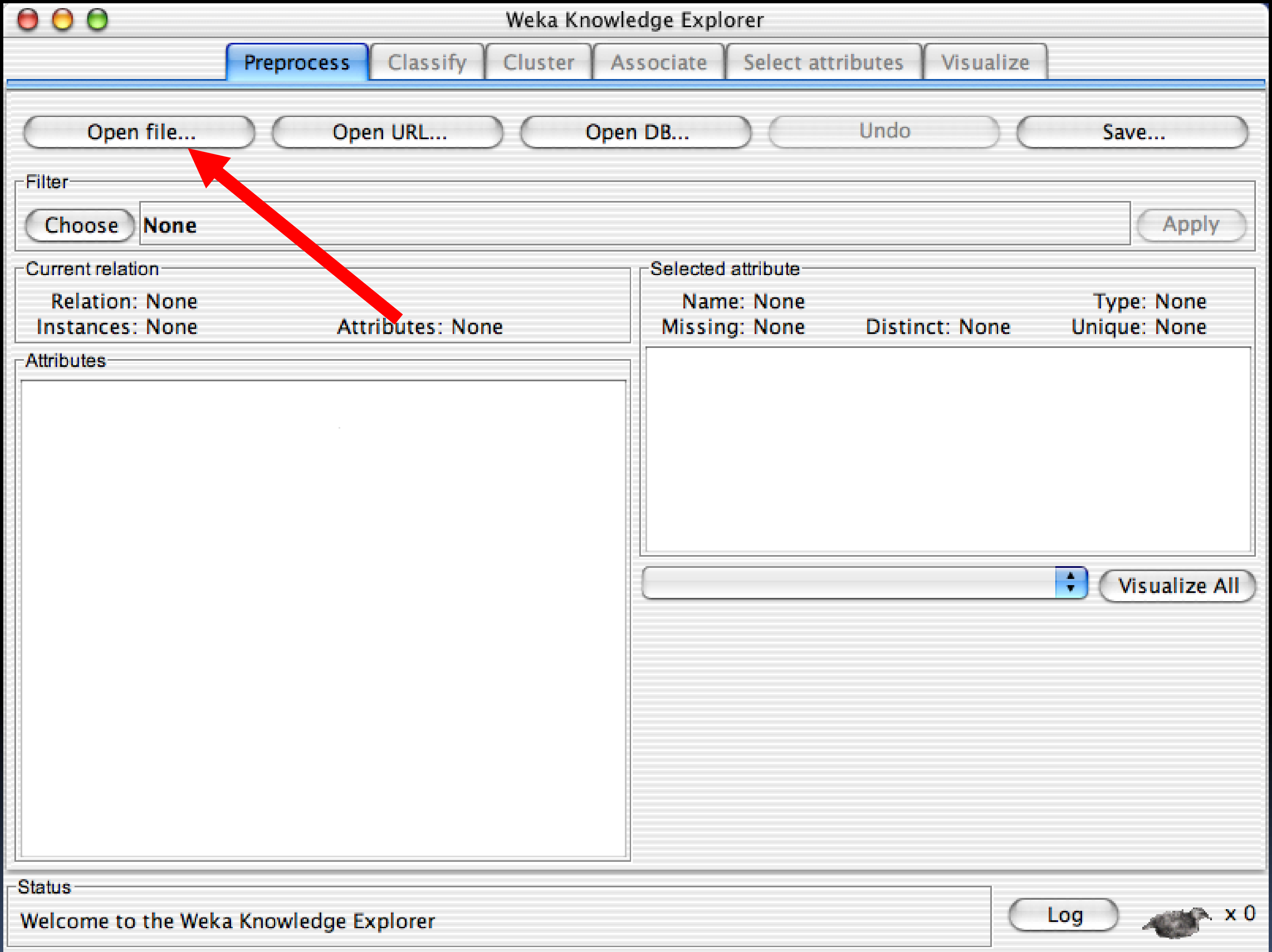
Welcome to the Weka Explorer

Importing Data

- Data can be imported from various formats:
  - ARFF, CSV, C4.5
  - You will most often import data in csv if not already prepared for WEKA
- Weka steps
  - Make sure Explorer "PreProcess" tab is selected
  - Click "Open File…" or "Open URL…"
  - Set the extension appropriately (default is .arff) and navigate to the file and select it
- If first line has feature names they are used
  - If no feature names then generic names are generated
  - Add feature names if missing or models won't be interpretable

Reading in the Iris Dataset

- We start with the well-known Iris data set
  - Weka probably installed it so search for "iris".
  - Otherwise download it from my copy on internet:
    - https://storm.cis.fordham.edu/~gweiss/data-mining/weka-data/iris.arff
- Open it using "Open File" or "Open URL"
- Weka immediately shows stats about the features and how they are distributed
- Other data sets it you want to play around:
    - https://storm.cis.fordham.edu/~gweiss/data-mining/datasets.html
  - Hundreds of others available from UCI Machine Learning Repository
    - https://archive.ics.uci.edu/ml/

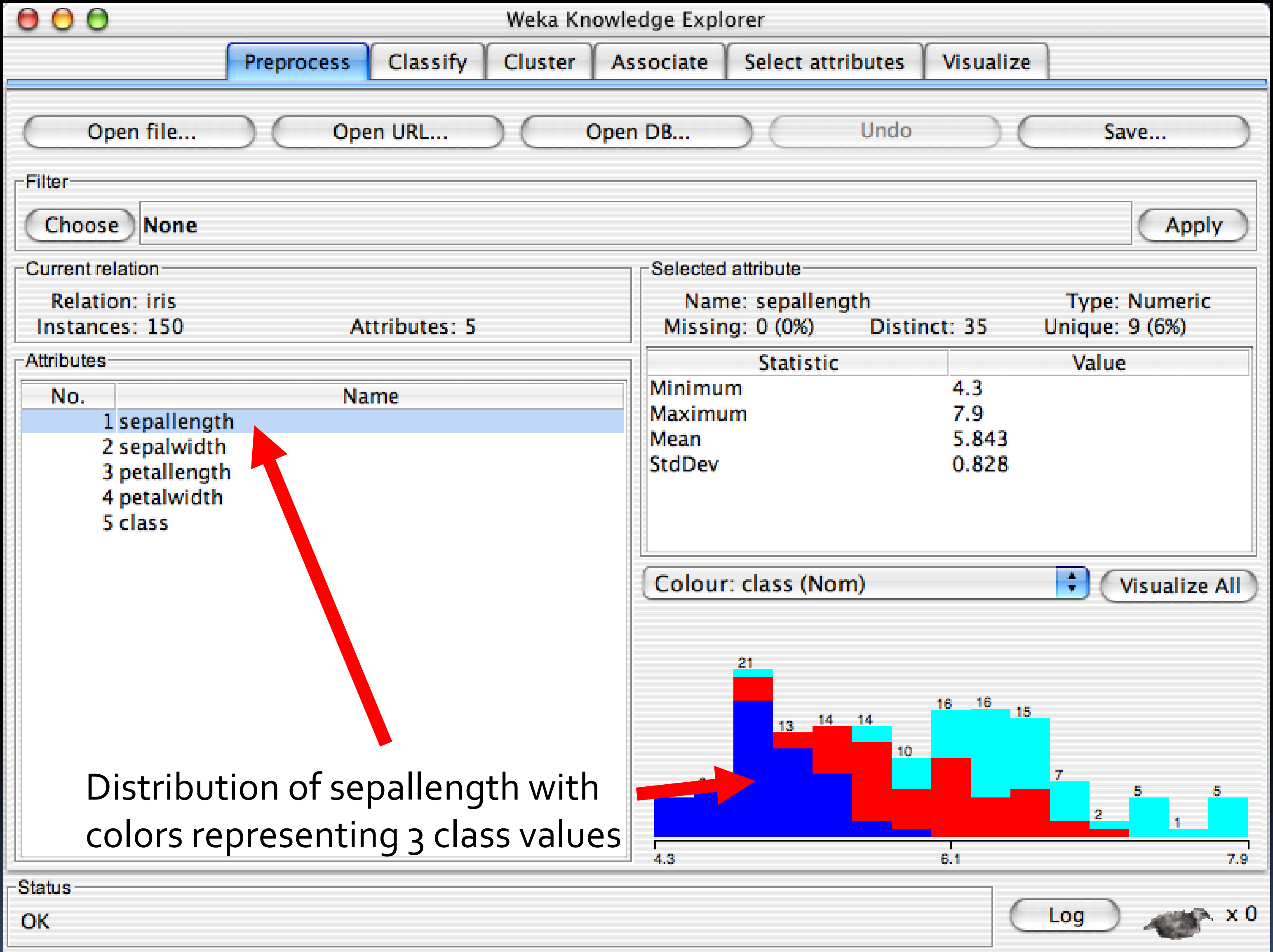


Filter: None

Relation: None
Instances: None
Attributes: None

Name: None
Type: None
Missing: None
Distinct: None
Unique: None

Welcome to the Weka Knowledge Explorer



None

Relation: iris
Instances: 150 Attributes: 5

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

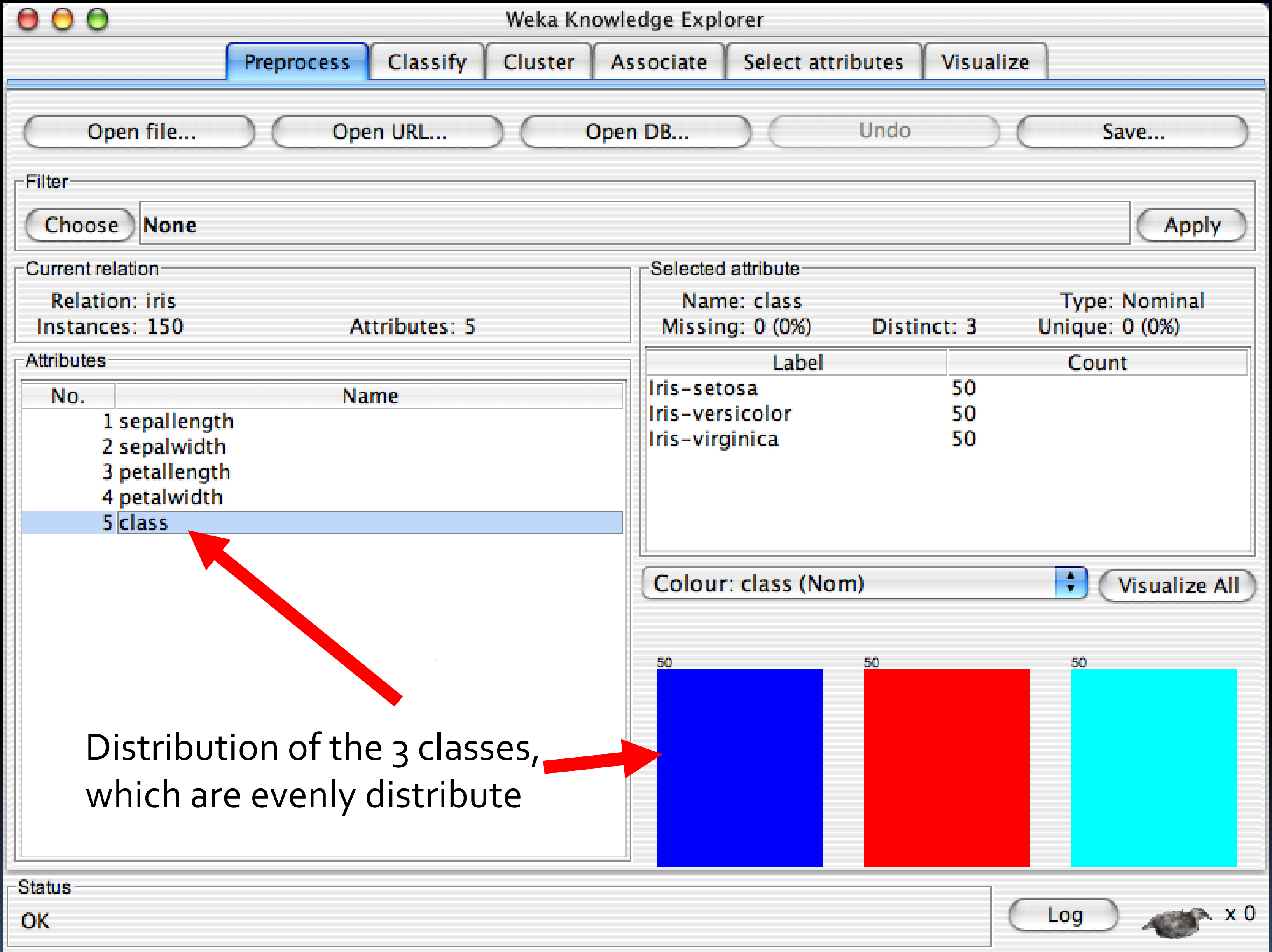
Name: sepallength Type: Numeric
Missing: 0 (0%) Distinct: 35 Unique: 9 (6%)

Statistic	Value
Minimum	4.3
Maximum	7.9
Mean	5.843
StdDev	0.828

Colour: class (Nom)

OK

Distribution of sepallength with colors representing 3 class values



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: class Type: Nominal
Missing: 0 (0%) Distinct: 3 Unique: 0 (0%)

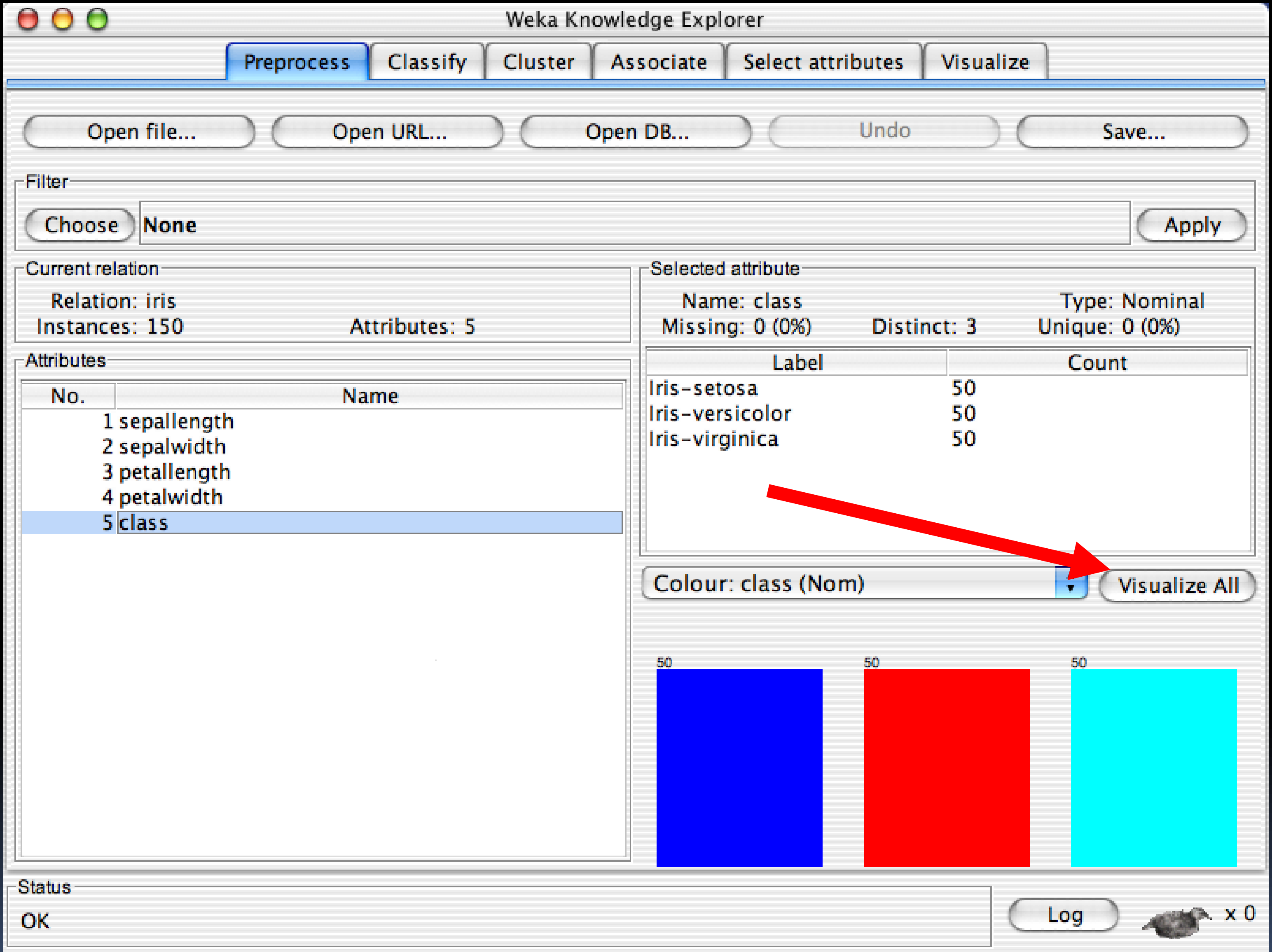
Label	Count
Iris-setosa	50
Iris-versicolor	50
Iris-virginica	50

Colour: class (Nom)

Status

OK

Distribution of the 3 classes, which are evenly distribute



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

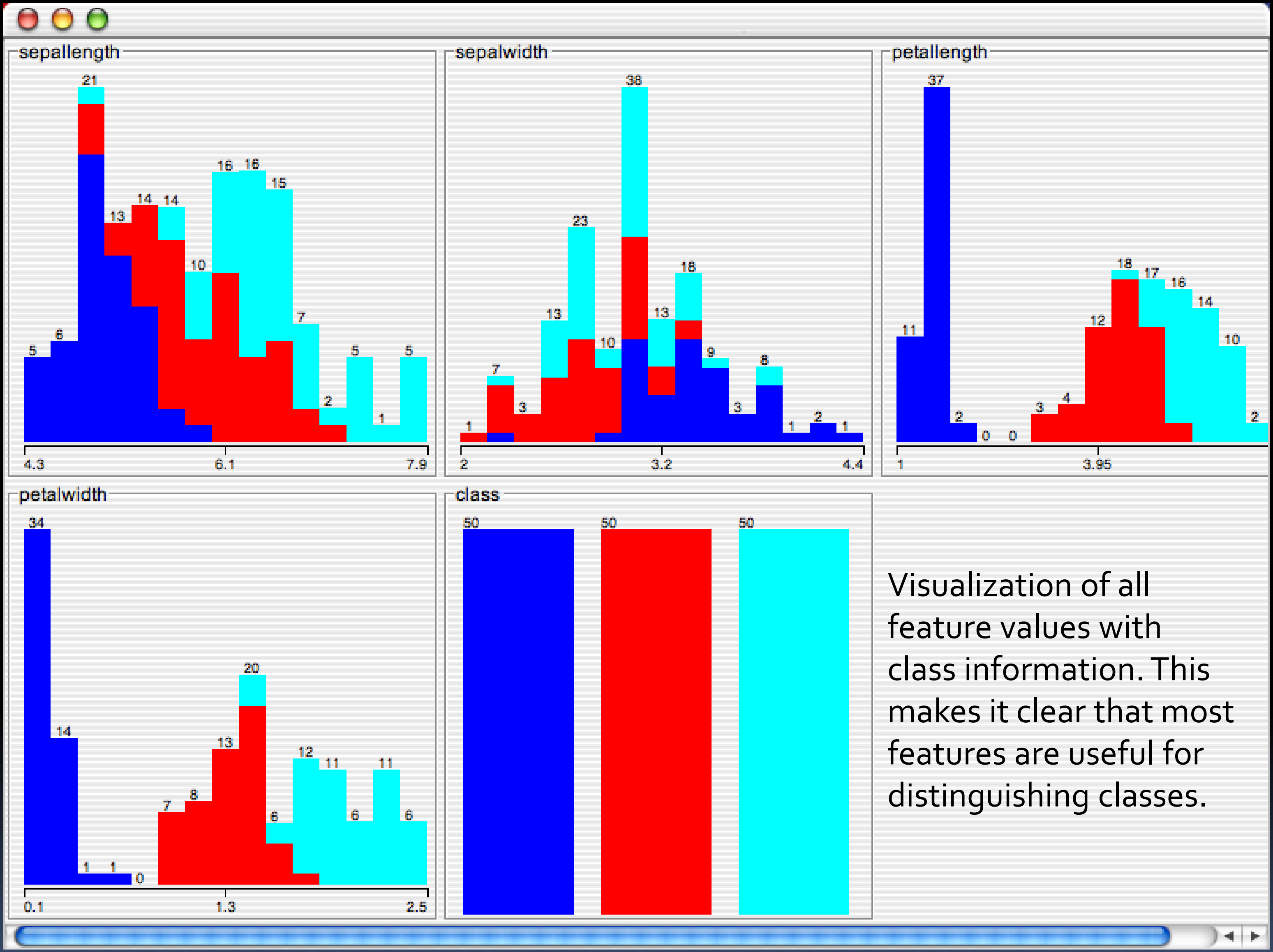
Name: class Type: Nominal
Missing: 0 (0%) Distinct: 3 Unique: 0 (0%)

Label	Count
Iris-setosa	50
Iris-versicolor	50
Iris-virginica	50

Colour: class (Nom)

Status

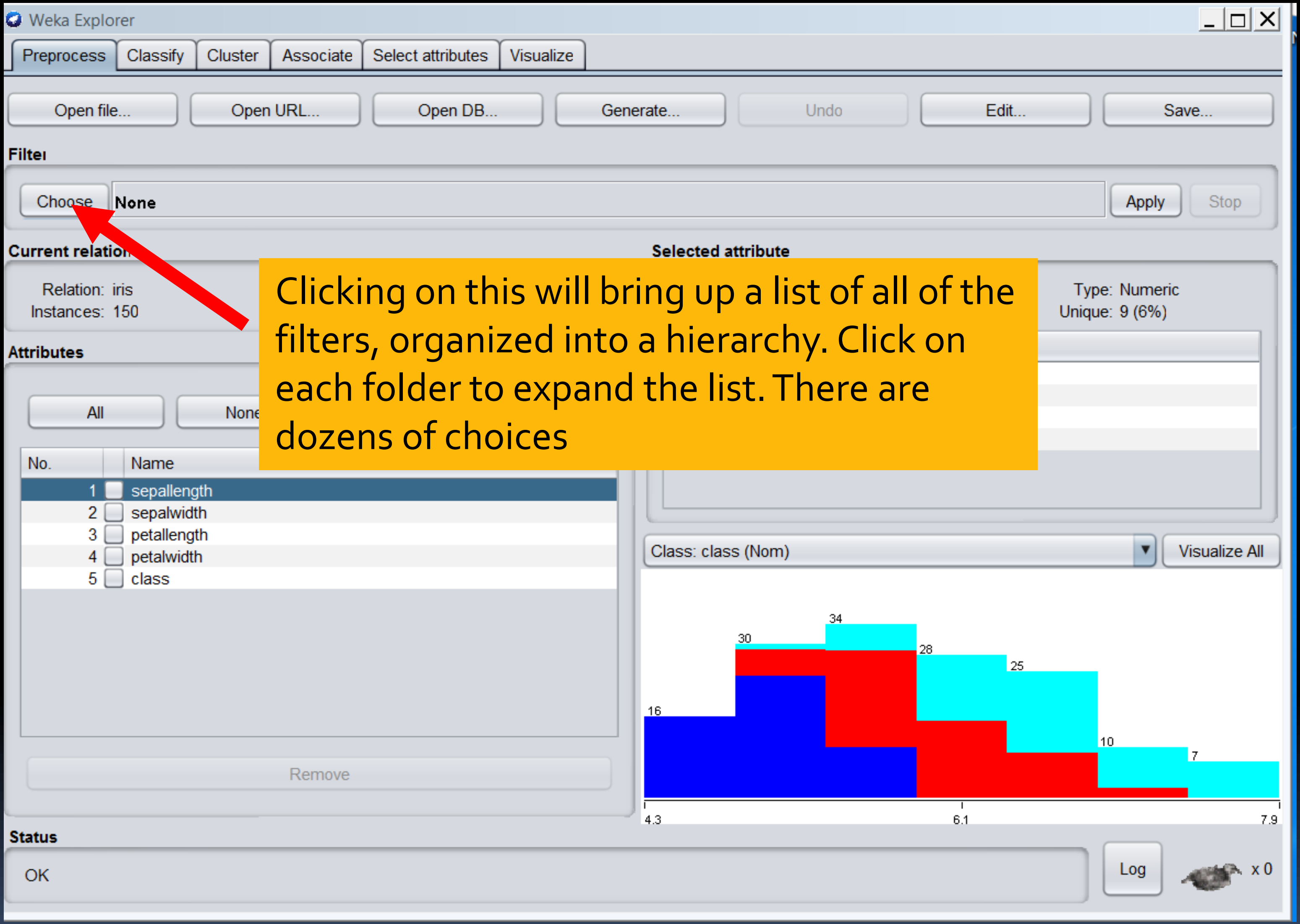
OK



Visualization of all feature values with class information. This makes it clear that most features are useful for distinguishing classes.

Preprocessing Filters

- Pre-processing tools in WEKA called "filters"
- Many preprocessing filters available:
  - Discretization
  - Normalization
  - Resampling
  - Feature selection
  - Feature transformation
  - Many more
- Know what available so can use it when needed
  - We will focus on Discretization



Relation: iris
Instances: 150

Type: Numeric
Unique: 9 (6%)

Attributes

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

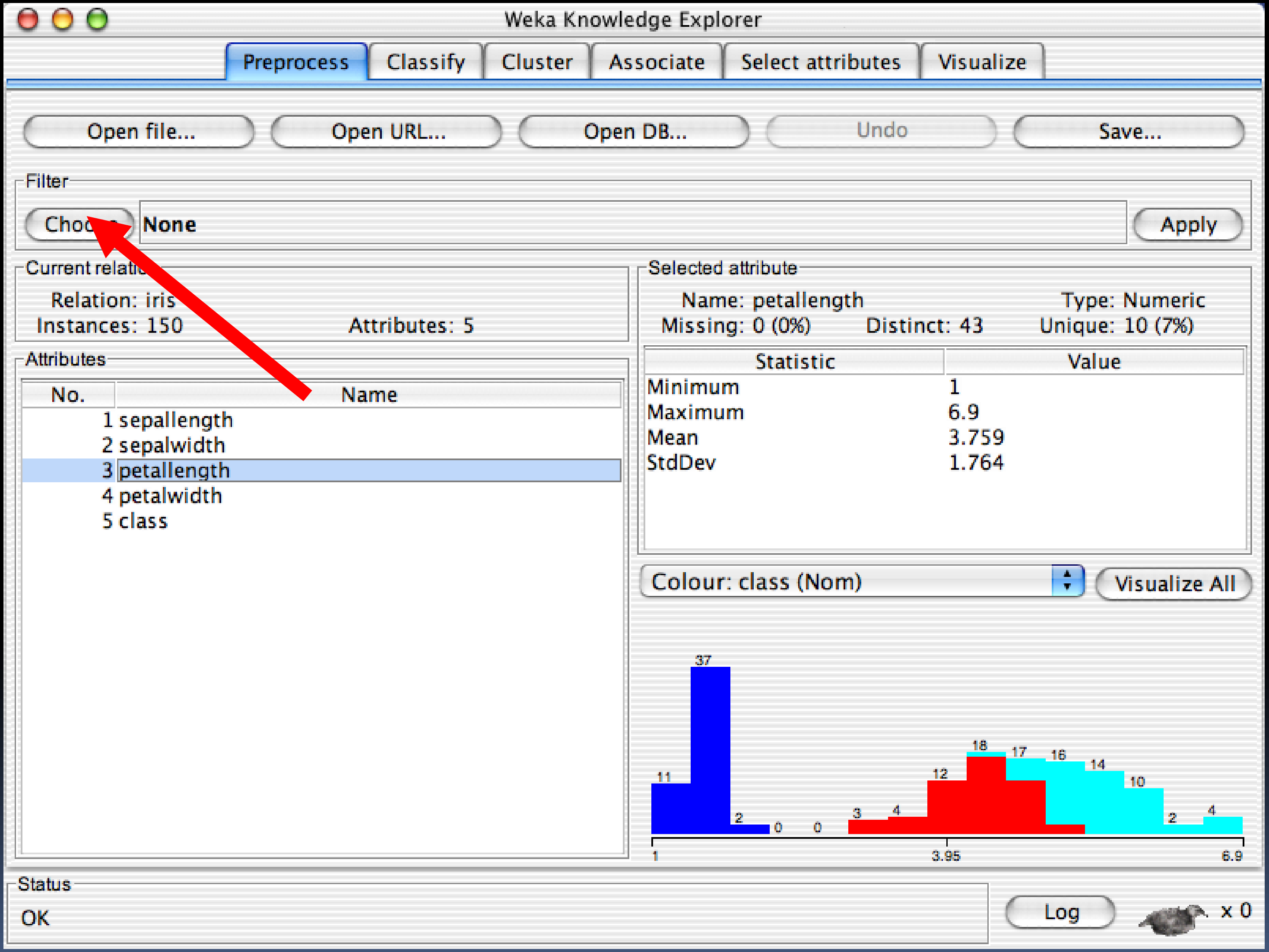
Class: class (Nom)

OK

Clicking on this will bring up a list of all of the filters, organized into a hierarchy. Click on each folder to expand the list. There are dozens of choices

Discretization

- Discretization: numerical feature → categorical
  - In preprocess tab select “Choose”, expand “unsupervised” then “attribute”
    - Select “Discretize” filter
    - Note discretize applies to attributes not instances
    - The following few slides are from an earlier version and may look just a little bit different
    - Note: there is a version under the “supervised” folder but it works differently and has different options



Filter: None

Relation: iris
Instances: 150
Attributes: 5

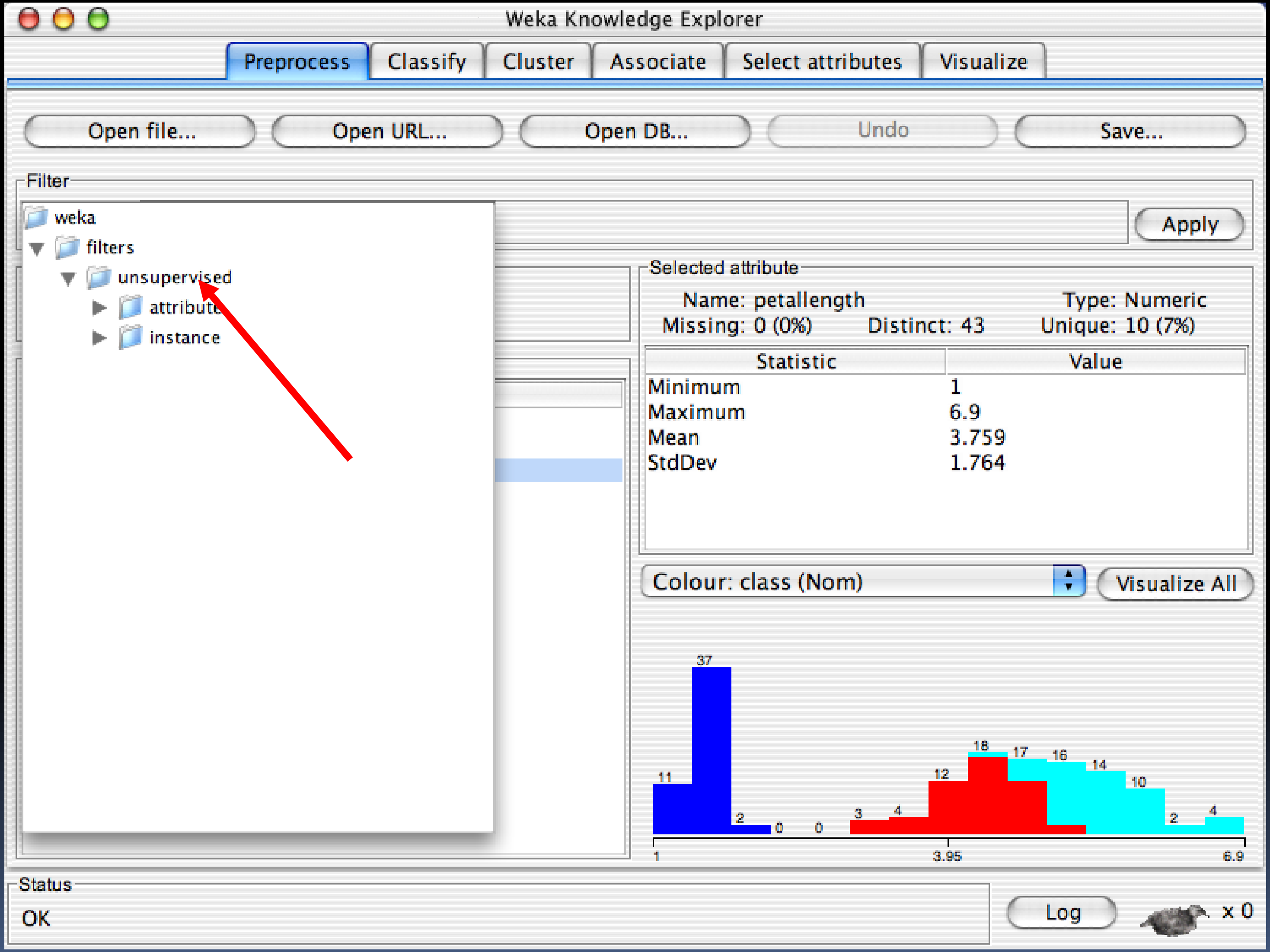
No.	Name
1	sepallength
2	sepalwidth
3	petallength
4	petalwidth
5	class

Name: petallength
Type: Numeric
Missing: 0 (0%)
Distinct: 43
Unique: 10 (7%)

Statistic	Value
Minimum	1
Maximum	6.9
Mean	3.759
StdDev	1.764

Colour: class (Nom)

Status: OK



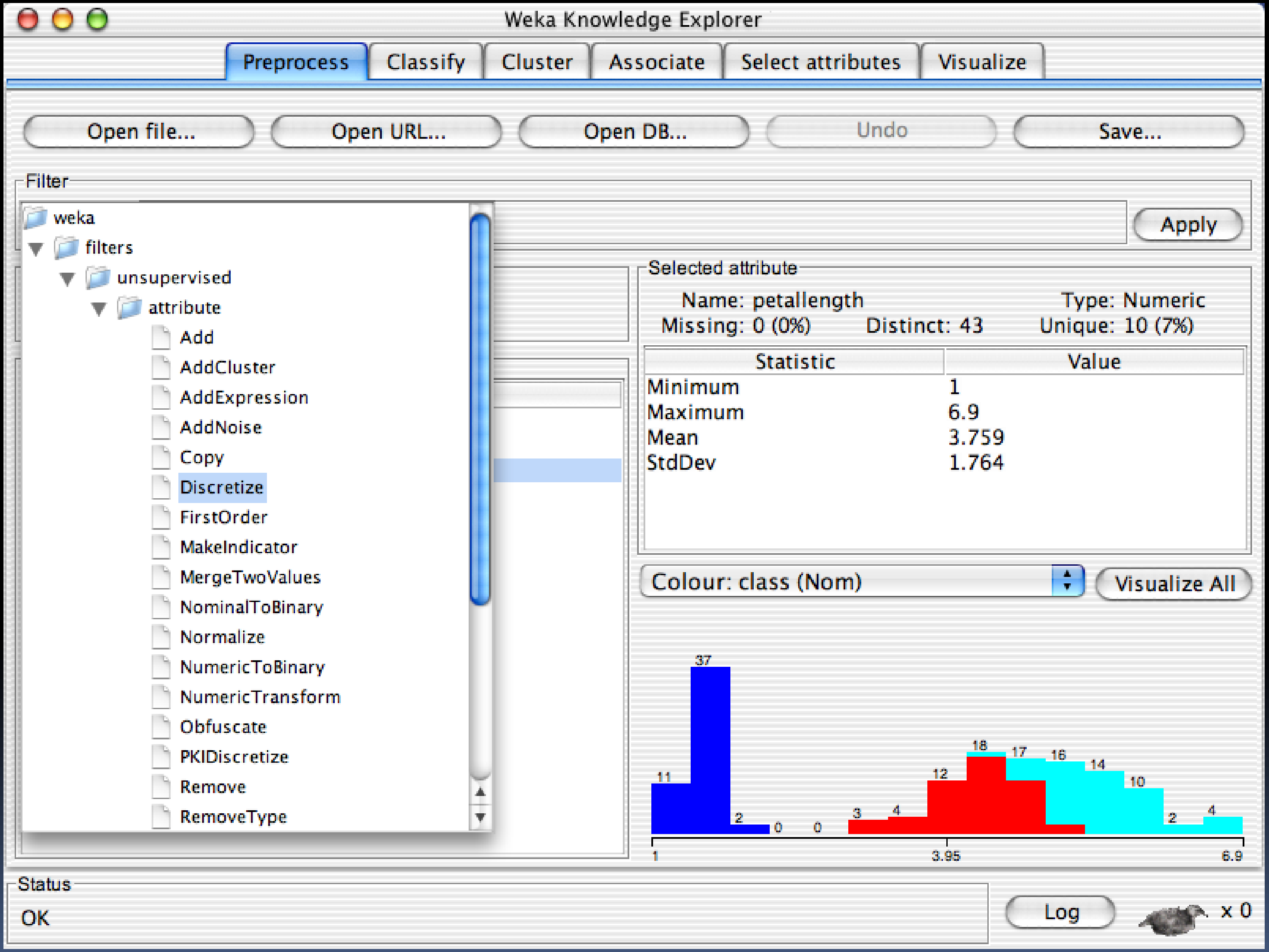
Selected attribute

Name: petallength
Type: Numeric
Missing: 0 (0%)
Distinct: 43
Unique: 10 (7%)

Statistic	Value
Minimum	1
Maximum	6.9
Mean	3.759
StdDev	1.764

Status

OK

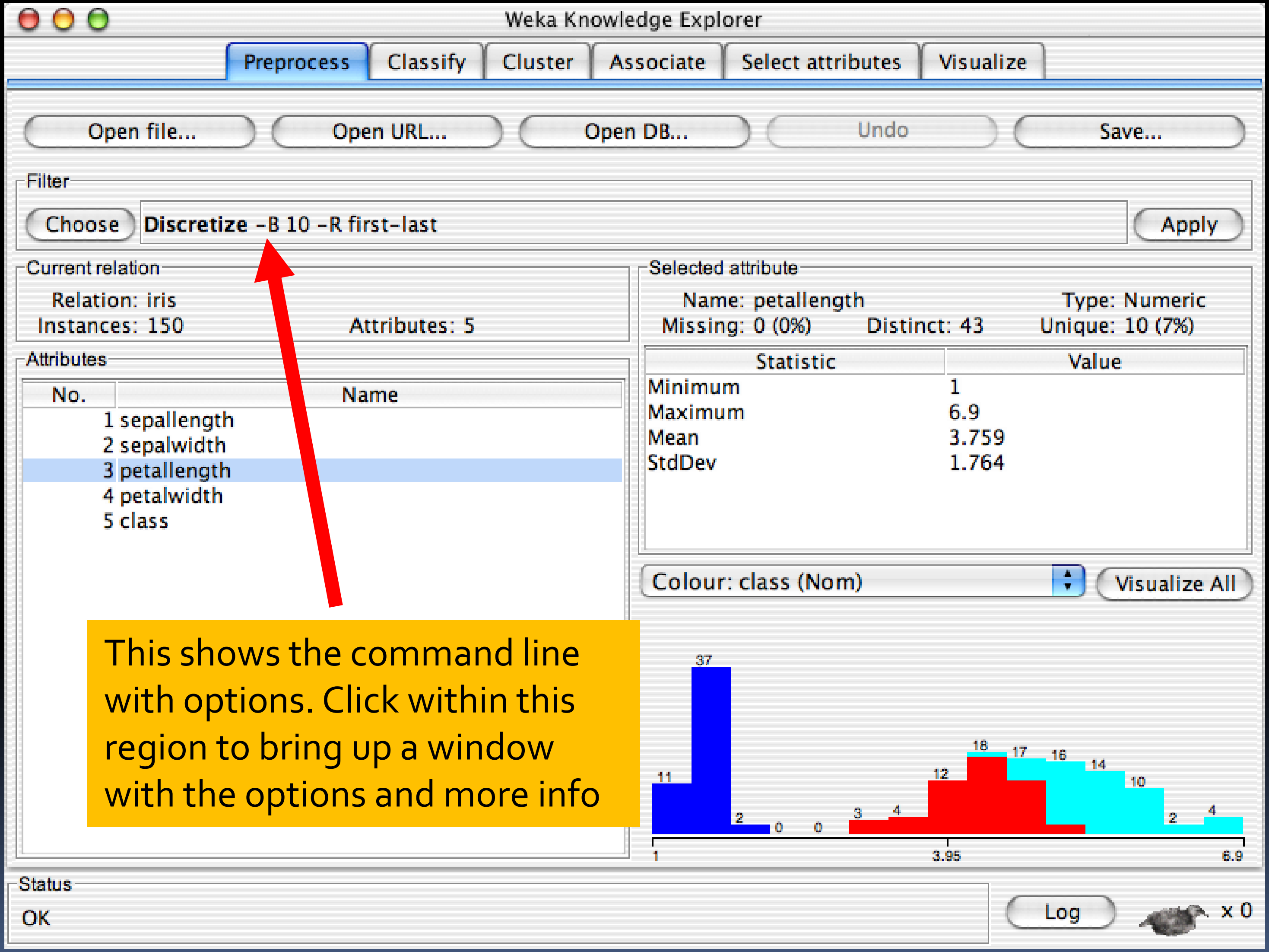


Selected attribute

Name: petallength Type: Numeric
Missing: 0 (0%) Distinct: 43 Unique: 10 (7%)

Statistic	Value
Minimum	1
Maximum	6.9
Mean	3.759
StdDev	1.764

Status
OK



Filter

Discretize -B 10 -R first-last

Current relation

Relation: iris
Instances: 150
Attributes: 5

Attributes

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

Selected attribute

Name: petallength
Type: Numeric
Missing: 0 (0%)
Distinct: 43
Unique: 10 (7%)

Statistic	Value
Minimum	1
Maximum	6.9
Mean	3.759
StdDev	1.764

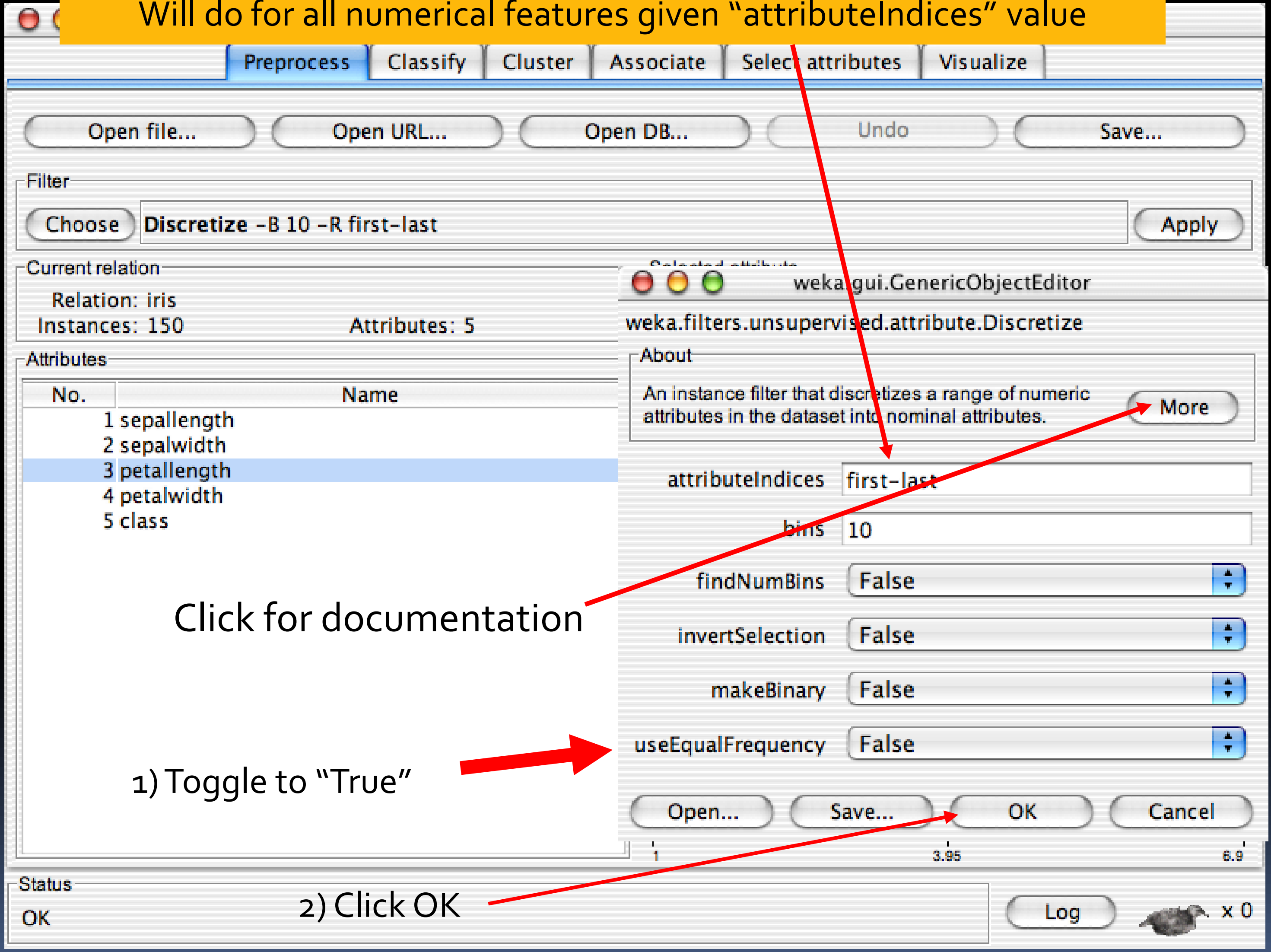
Status

OK

This shows the command line with options. Click within this region to bring up a window with the options and more info

Create 10 equal frequency bins

Will do for all numerical features given “attributeIndices” value



Filter

Choose **Discretize** -B 10 -R first-last Apply

Current relation

Relation: iris
Instances: 150 Attributes: 5

Attributes

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

Status

OK

weka.gui.GenericObjectEditor

weka.filters.unsupervised.attribute.Discretize

About

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

More

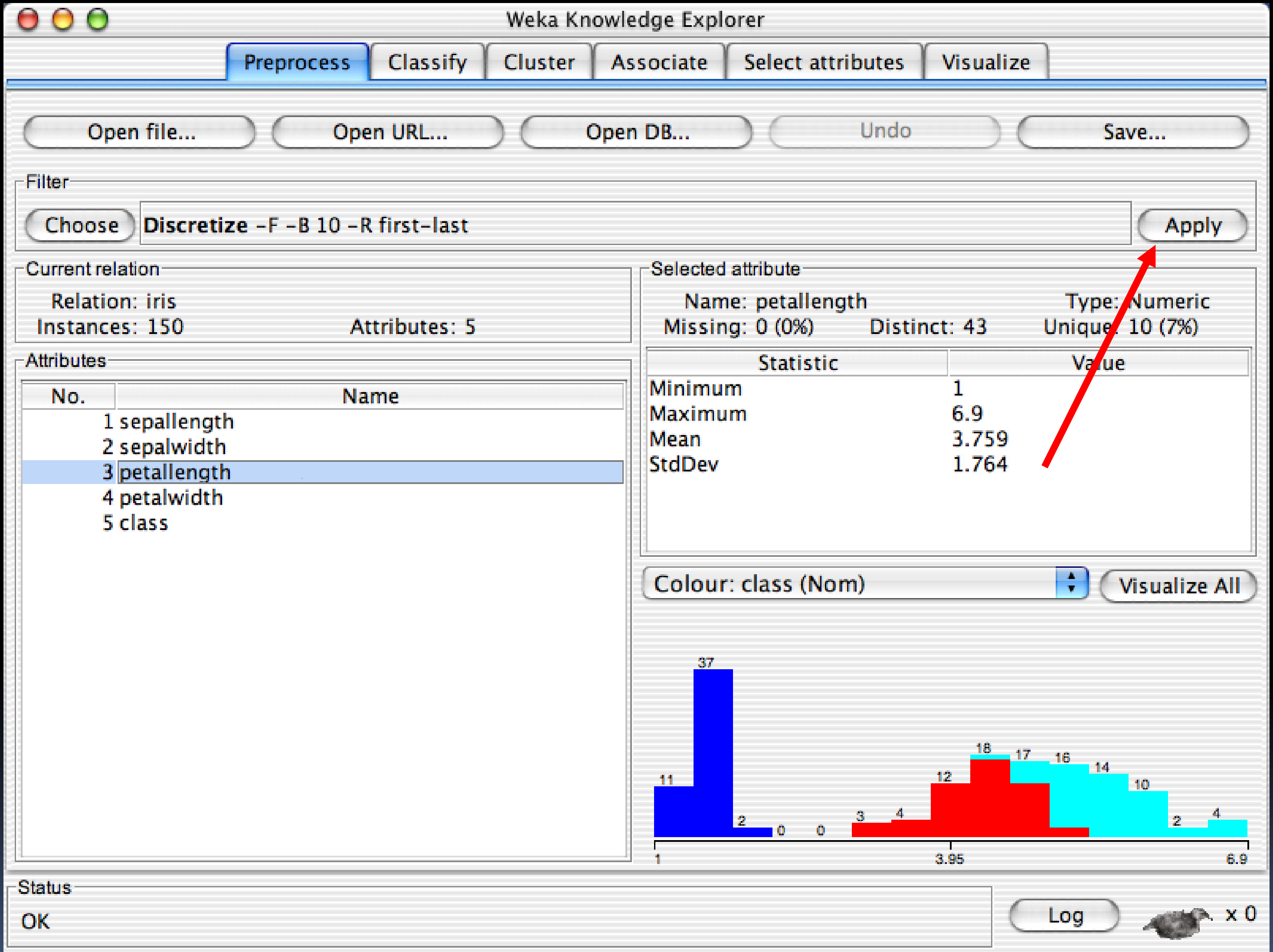
attributeIndices	first-last
bins	10
findNumBins	False
invertSelection	False
makeBinary	False
useEqualFrequency	False

Open... Save... OK Cancel

Click for documentation

1) Toggle to “True”

2) Click OK



Filter

Discretize -F -B 10 -R first-last

Current relation

Relation: iris
Instances: 150 Attributes: 5

Attributes

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

Selected attribute

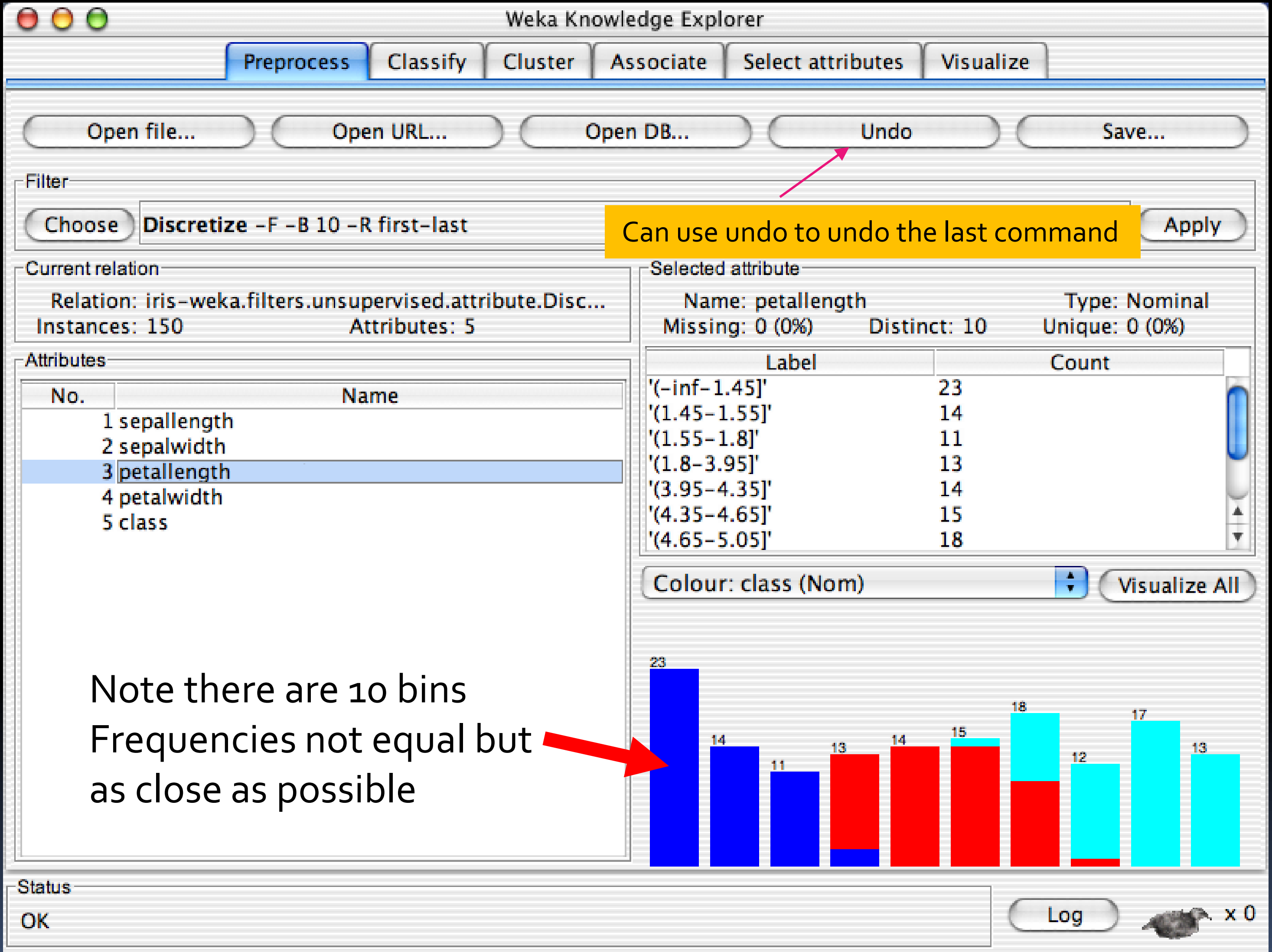
Name: petallength Type: Numeric
Missing: 0 (0%) Distinct: 43 Unique: 10 (7%)

Statistic	Value
Minimum	1
Maximum	6.9
Mean	3.759
StdDev	1.764

Colour: class (Nom)

Status

OK



Filter: Discretize -F -B 10 -R first-last

Current relation

Relation: iris-weka.filters.unsupervised.attribute.Disc...
Instances: 150 Attributes: 5

Attributes

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

Selected attribute

Name: petallength Type: Nominal
Missing: 0 (0%) Distinct: 10 Unique: 0 (0%)

Label	Count
'(-inf-1.45]'	23
'(1.45-1.55]'	14
'(1.55-1.8]'	11
'(1.8-3.95]'	13
'(3.95-4.35]'	14
'(4.35-4.65]'	15
'(4.65-5.05]'	18

Status

OK

Can use undo to undo the last command

Note there are 10 bins
Frequencies not equal but
as close as possible

Weka-2 Review: Preprocessing

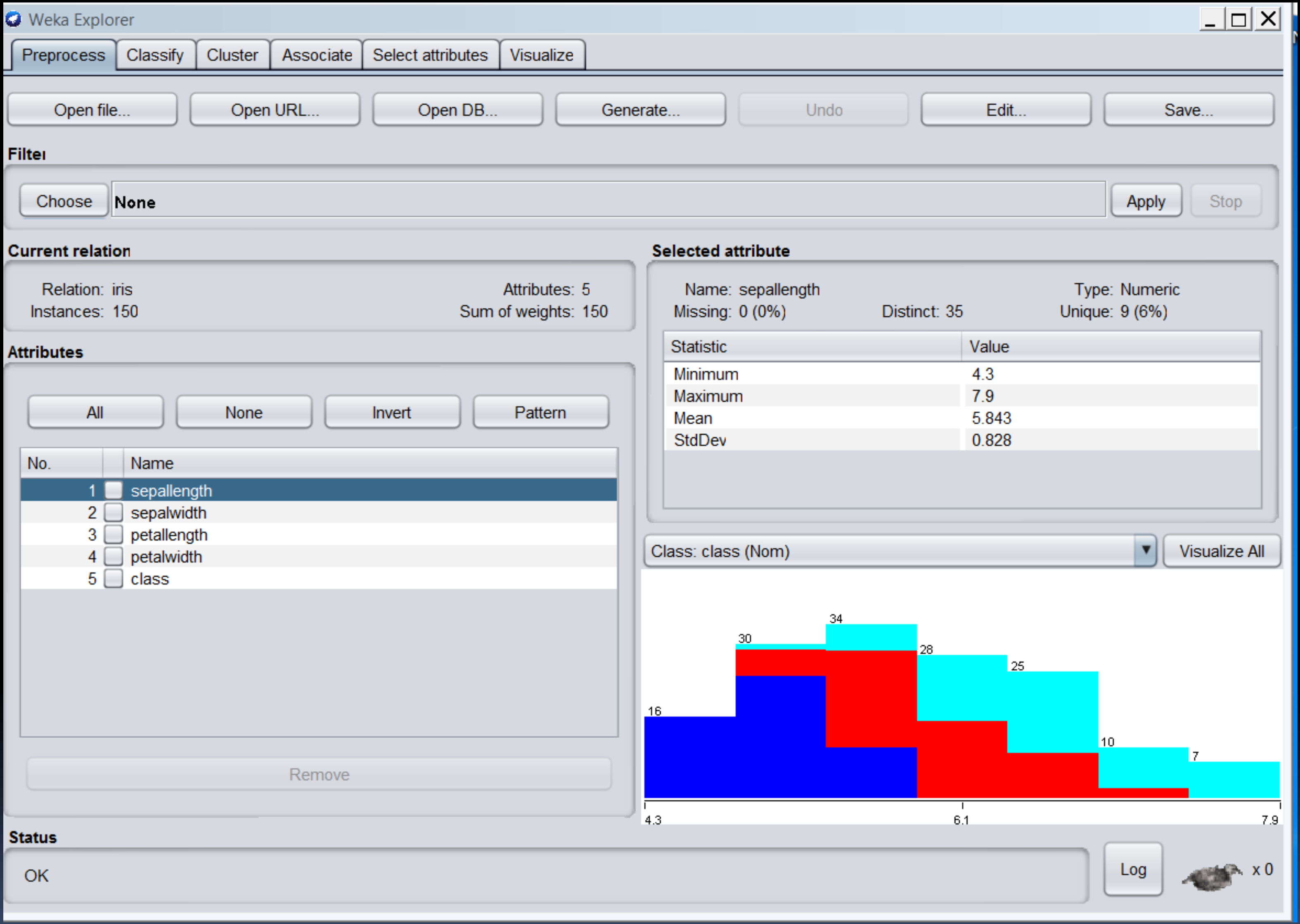
- You should now be able to:
  - Import data into Weka
  - Visualize the features and how they relate to class
  - Have an idea of the preprocessing facilities (filters) in Weka and be able to apply them

Weka-3: Decision Trees

- Weka supports many classification and regression algorithms
- We will go over a DT example is some detail
  - Build a decision tree to classify Iris data set
  - Examine the results
  - Visualize the decision tree
  - Visualize the errors

Weka-3: Prediction Algorithms

- Weka supports all major classification and regression methods
  - Decision Trees, Rule learners, Nearest Neighbor, Naïve Bayes, Neural Networks, etc.
- Also support ensemble classifiers:
  - Will learn about these later, but combine classifiers
- We first focus on decision trees
- Let's start fresh without the discretization
  - Undo last step or close window and reload "Iris"



Filter: None

Current relation

Relation: iris
Attributes: 5
Instances: 150
Sum of weights: 150

Attributes

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

Selected attribute

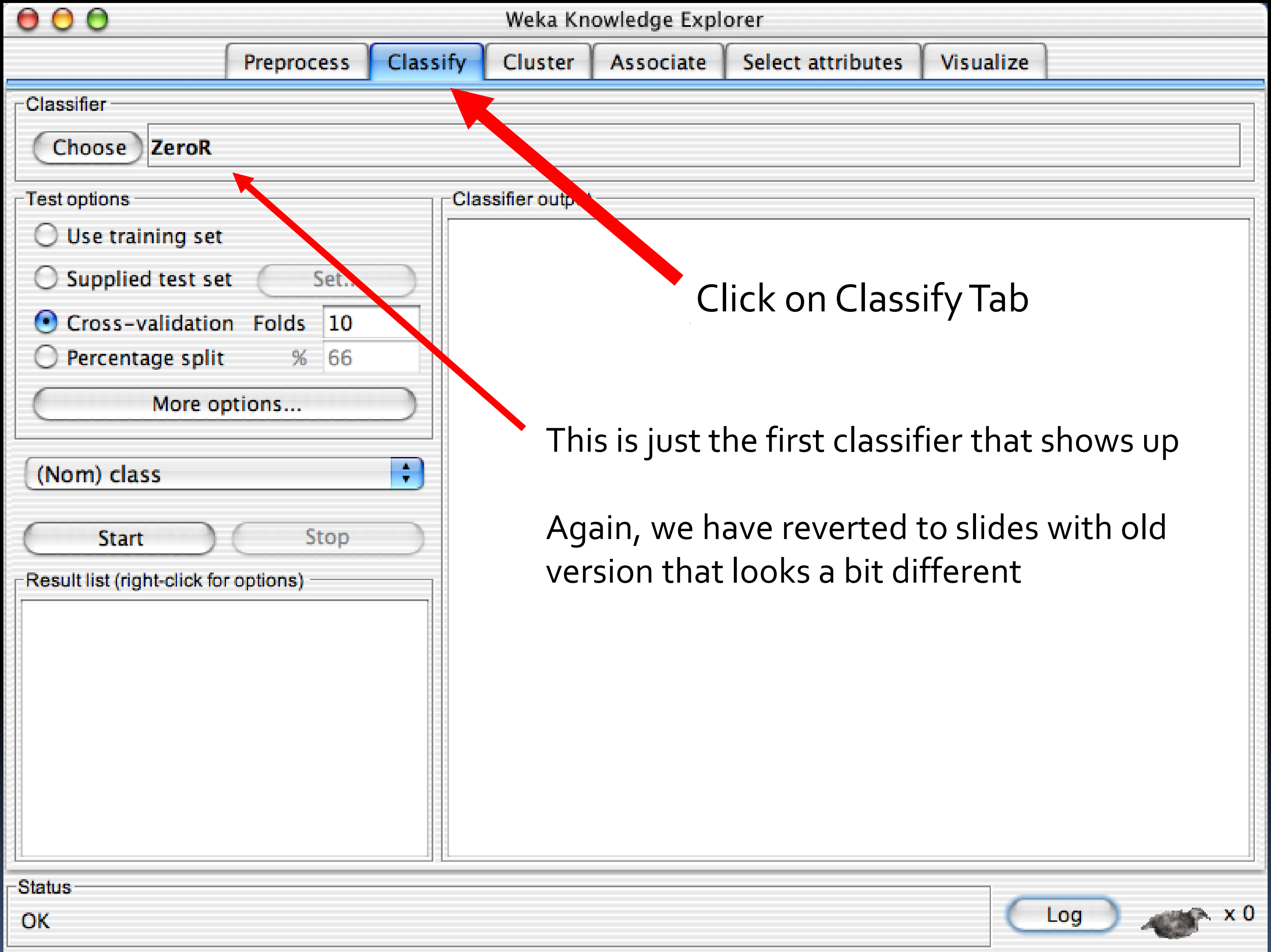
Name: sepallength
Type: Numeric
Missing: 0 (0%)
Distinct: 35
Unique: 9 (6%)

Statistic	Value
Minimum	4.3
Maximum	7.9
Mean	5.843
StdDev	0.828

Class: class (Nom)

Status

OK



Classifier: ZeroR

Test options

○ Use training set

○ Supplied test set

● Cross-validation Folds 10

○ Percentage split % 66

(Nom) class

Classifier output

Result list (right-click for options)

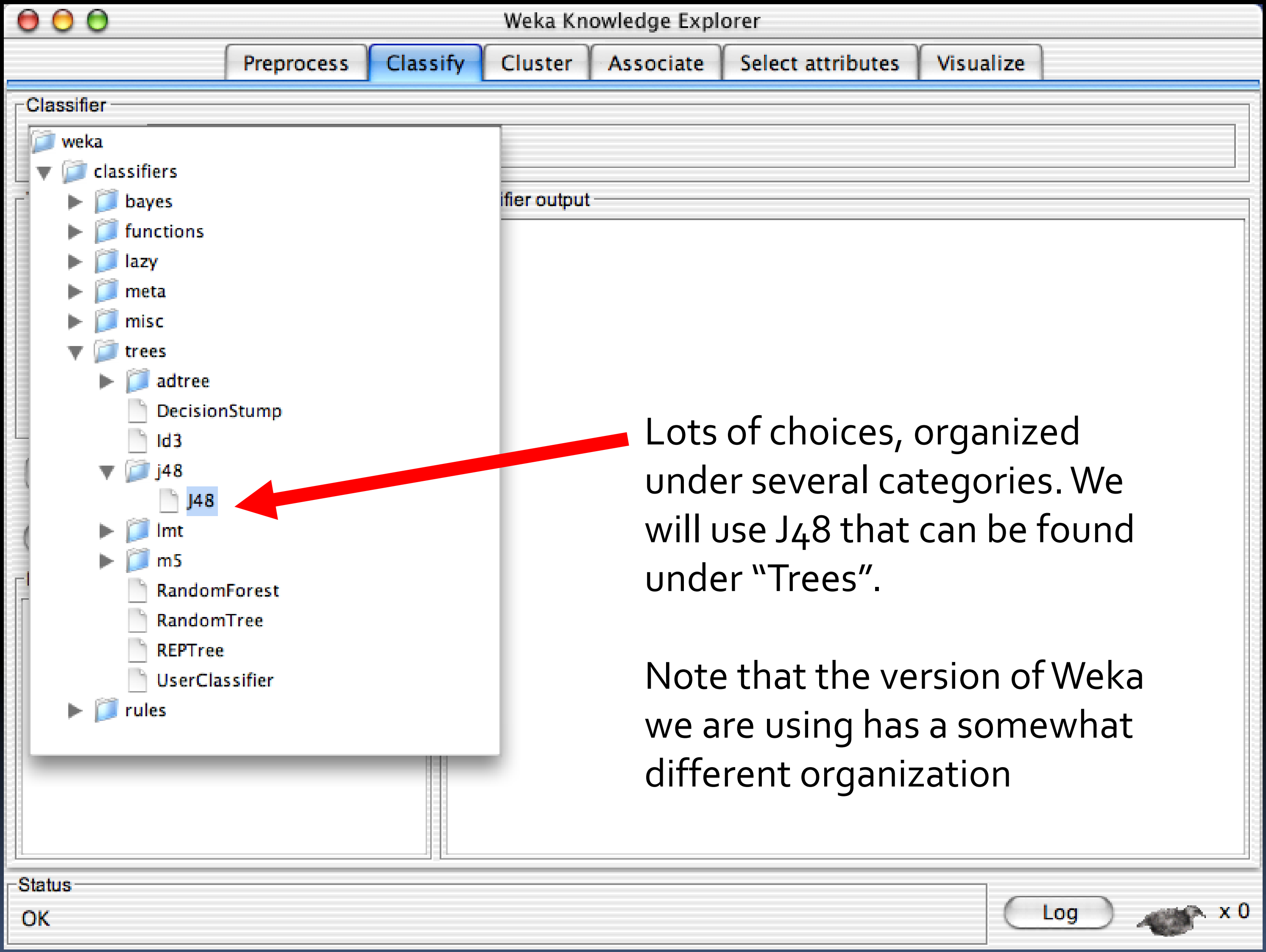
Status

OK

Click on Classify Tab

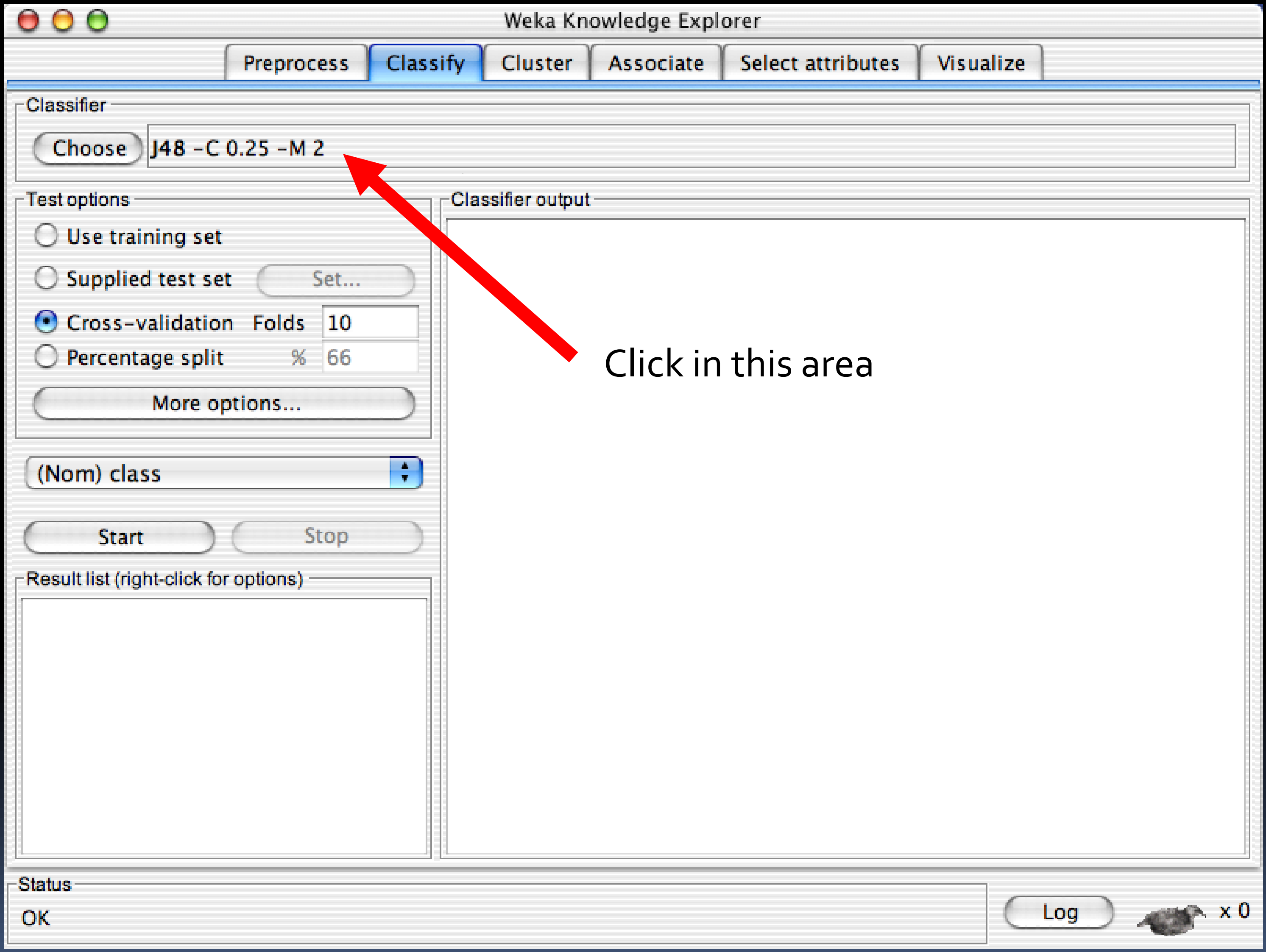
This is just the first classifier that shows up

Again, we have reverted to slides with old version that looks a bit different



Lots of choices, organized under several categories. We will use J48 that can be found under "Trees".

Note that the version of Weka we are using has a somewhat different organization



Weka Knowledge Explorer

Classifier

J48 -C 0.25 -M 2

Test options

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

(Nom) class

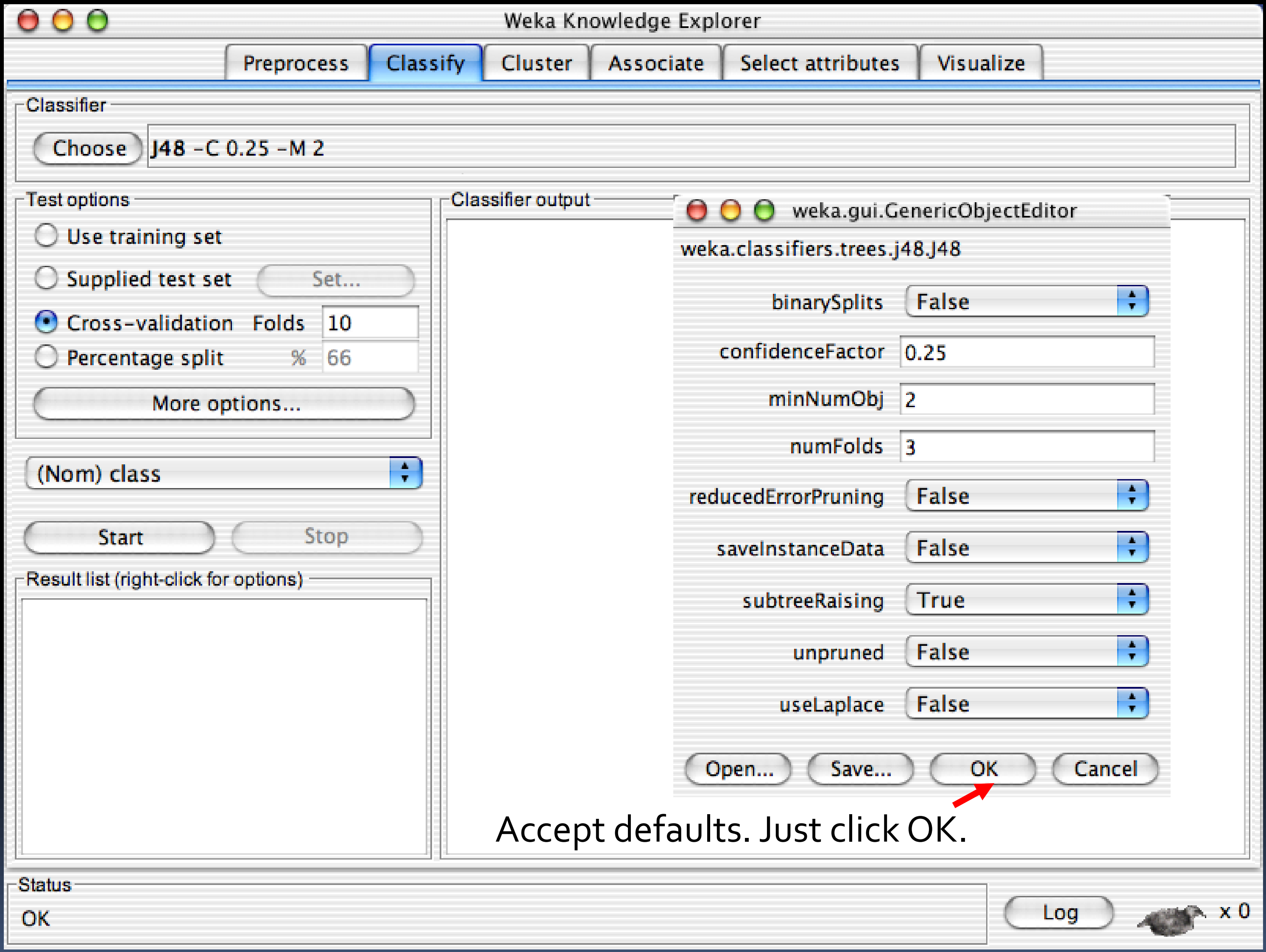
Result list (right-click for options)

Classifier output

Status

OK

Click in this area



Weka Knowledge Explorer

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

Classifier

Choose J48 -C 0.25 -M 2

Test options

○ Use training set

○ Supplied test set Set...

● Cross-validation Folds 10

○ Percentage split % 66

More options...

(Nom) class

Start Stop

Result list (right-click for options)

Classifier output

weka.gui.GenericObjectEditor

weka.classifiers.trees.j48.J48

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

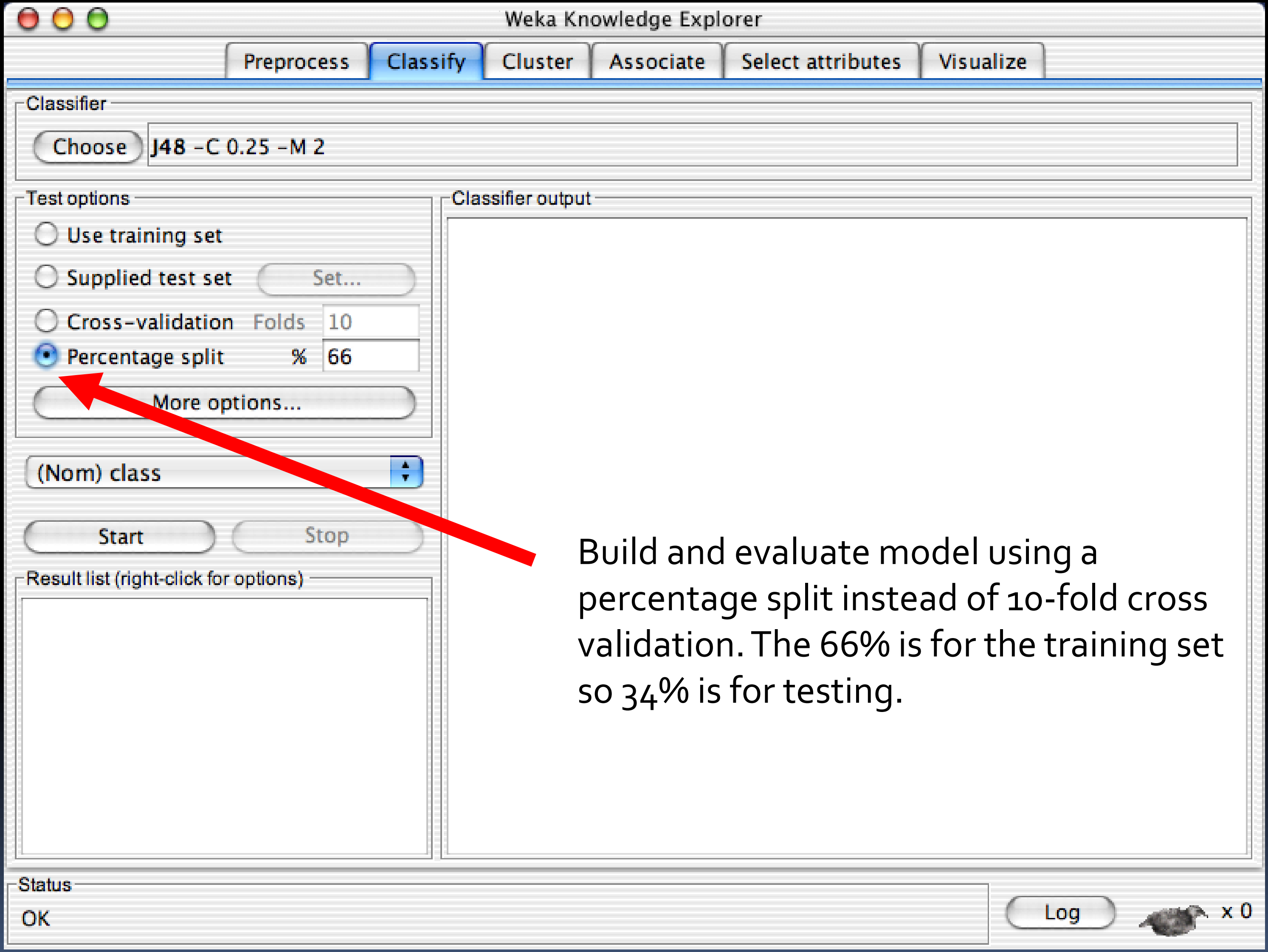
Open... Save... OK Cancel

Accept defaults. Just click OK.

Status

OK

Log x 0



Classifier

J48 -C 0.25 -M 2

Test options

○ Use training set

○ Supplied test set

○ Cross-validation Folds 10

● Percentage split % 66

(Nom) class

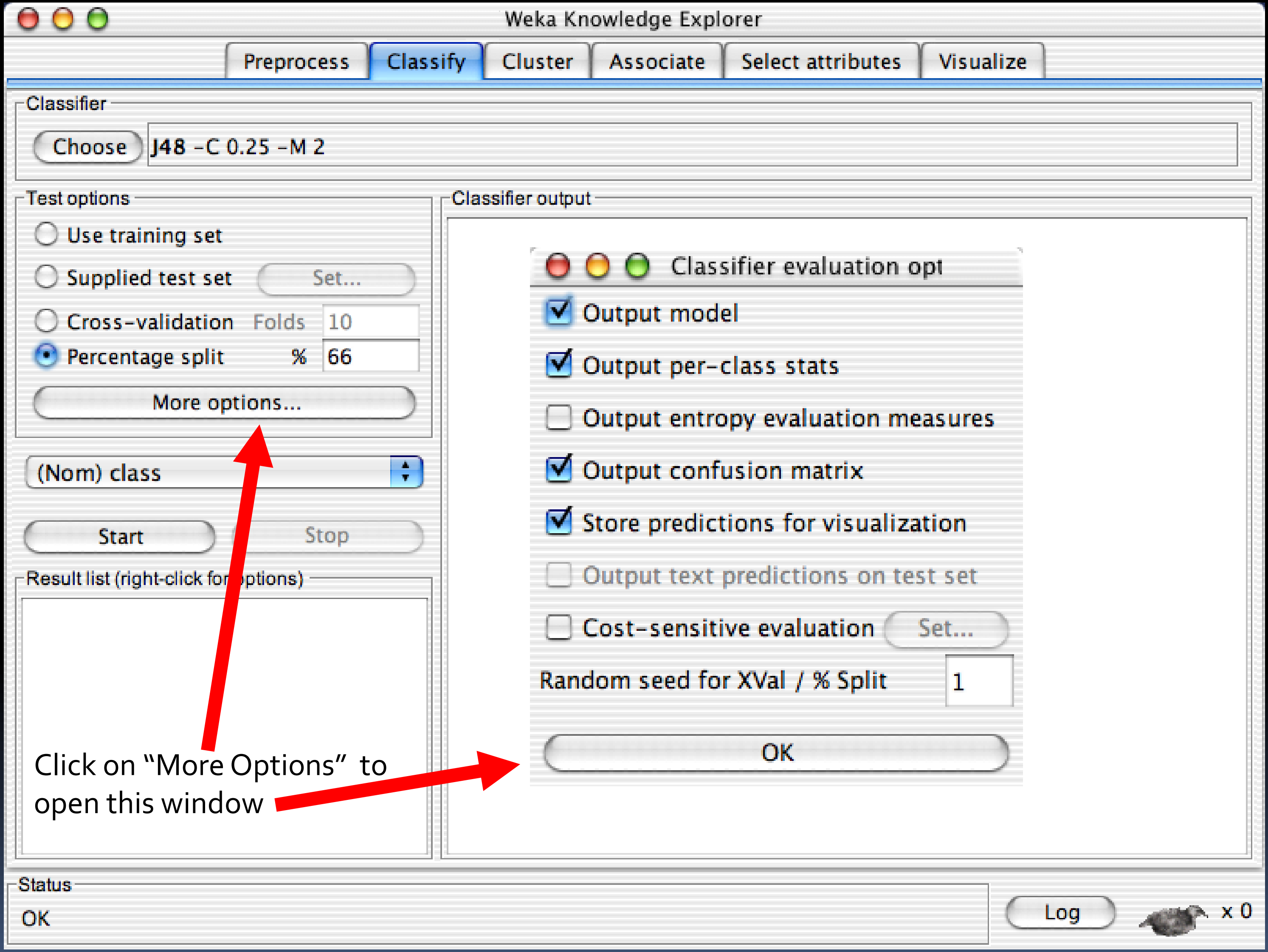
Classifier output

Result list (right-click for options)

Status

OK

Build and evaluate model using a percentage split instead of 10-fold cross validation. The 66% is for the training set so 34% is for testing.



Weka Knowledge Explorer

Classifier

Choose J48 -C 0.25 -M 2

Test options

○ Use training set

○ Supplied test set Set...

○ Cross-validation Folds 10

● Percentage split % 66

More options...

(Nom) class

Start Stop

Result list (right-click for options)

Classifier output

Classifier evaluation opt

☑ Output model

☑ Output per-class stats

□ Output entropy evaluation measures

☑ Output confusion matrix

☑ Store predictions for visualization

□ Output text predictions on test set

□ Cost-sensitive evaluation Set...

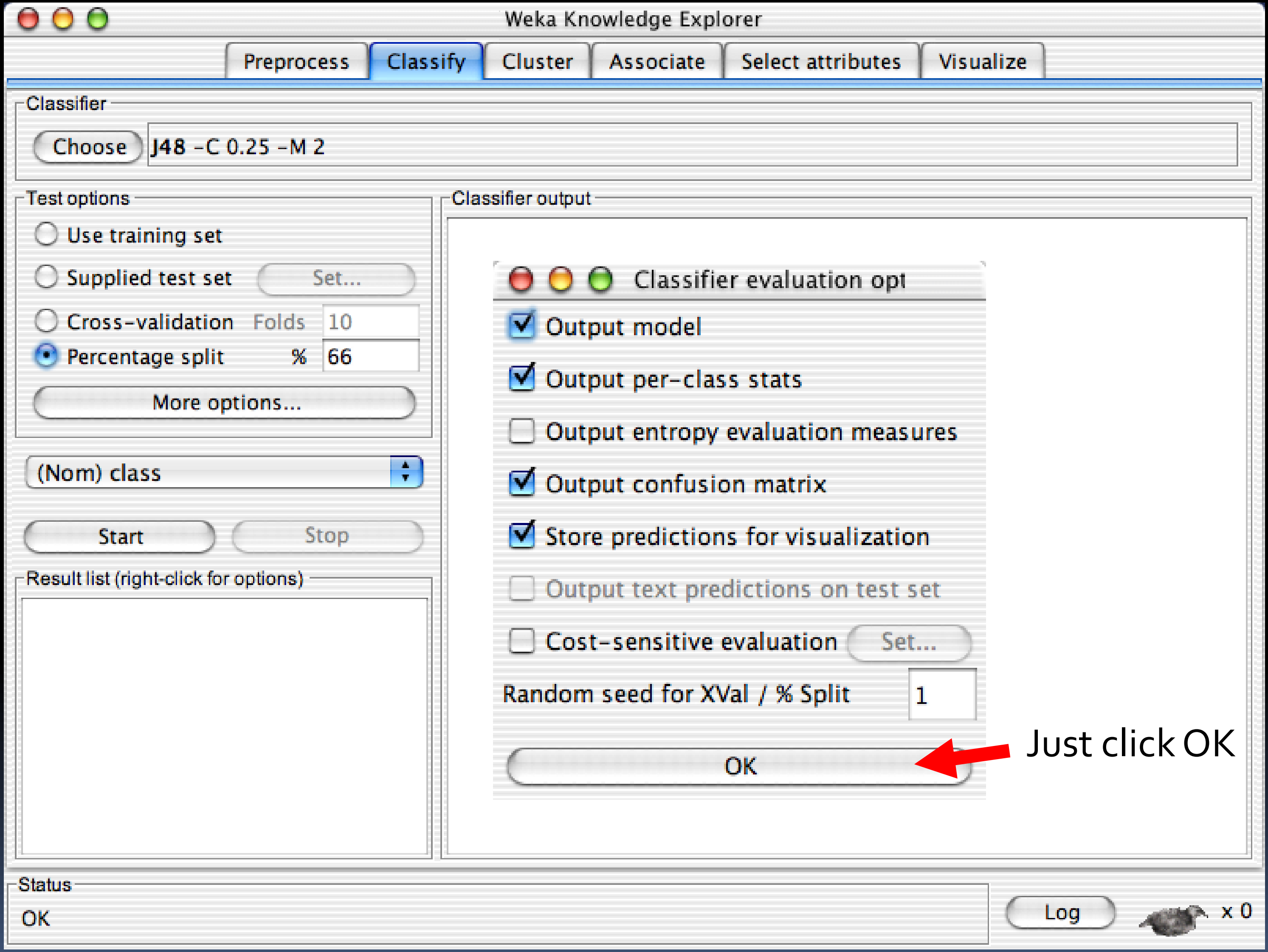
Random seed for XVal / % Split 1

OK

Status

OK

Click on "More Options" to open this window



J48 -C 0.25 -M 2

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

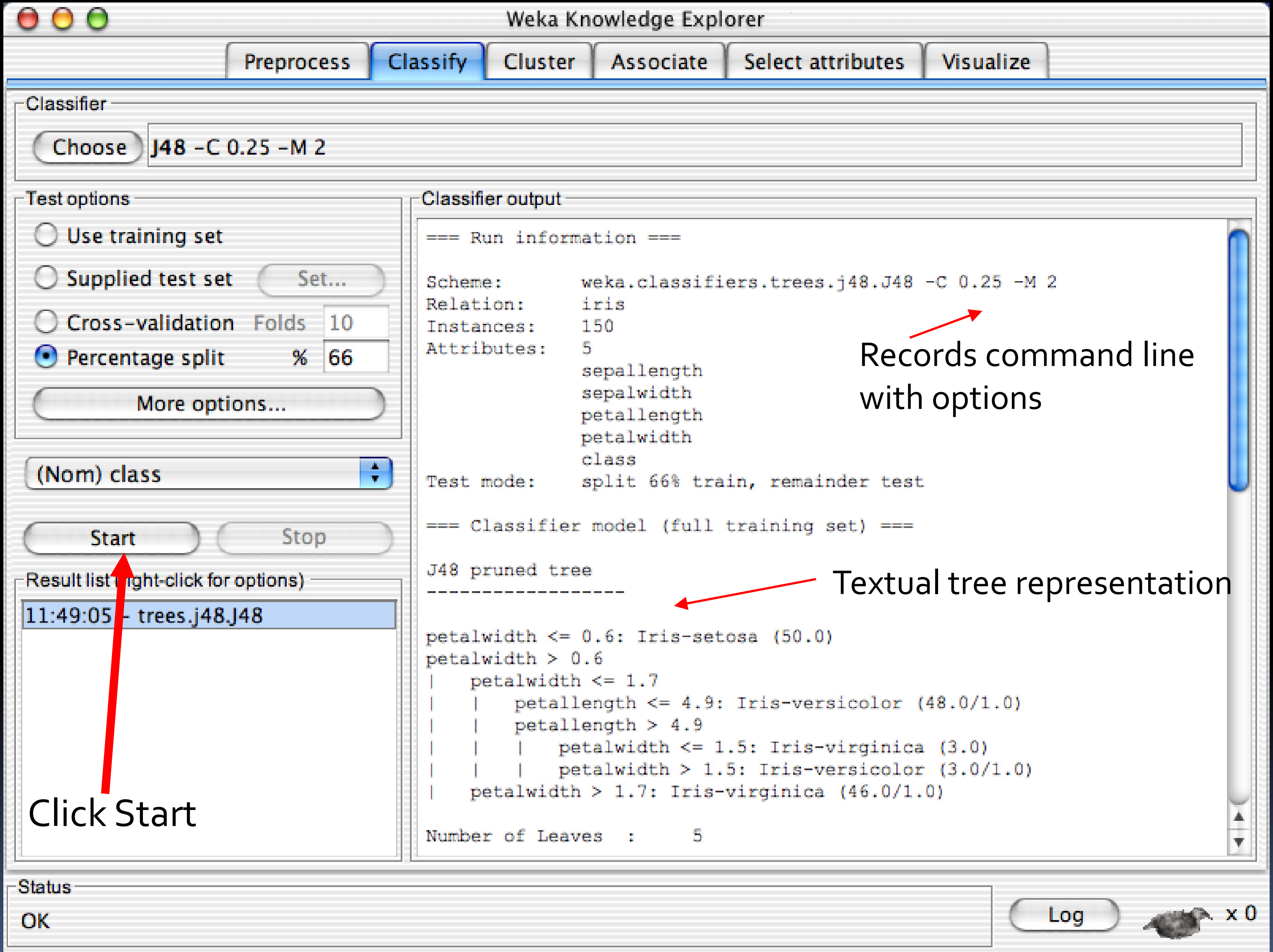
(Nom) class

☑ Output model
☑ Output per-class stats
□ Output entropy evaluation measures
☑ Output confusion matrix
☑ Store predictions for visualization
□ Output text predictions on test set
□ Cost-sensitive evaluation
Random seed for XVal / % Split 1

Status: OK

Just click OK

Note: Do not worry if your results are slightly different



Classifier

Choose J48 -C 0.25 -M 2

Test options

○ Use training set

○ Supplied test set

○ Cross-validation Folds 10

● Percentage split % 66

(Nom) class

Result list (right-click for options)

11:49:05 - trees.j48.J48

Classifier output

```
=== Run information ===

Scheme:       weka.classifiers.trees.j48.J48 -C 0.25 -M 2
Relation:     iris
Instances:    150
Attributes:   5
              sepallength
              sepalwidth
              petallength
              petalwidth
              class
Test mode:    split 66% train, remainder test

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

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

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

Number of Leaves  :     5
```

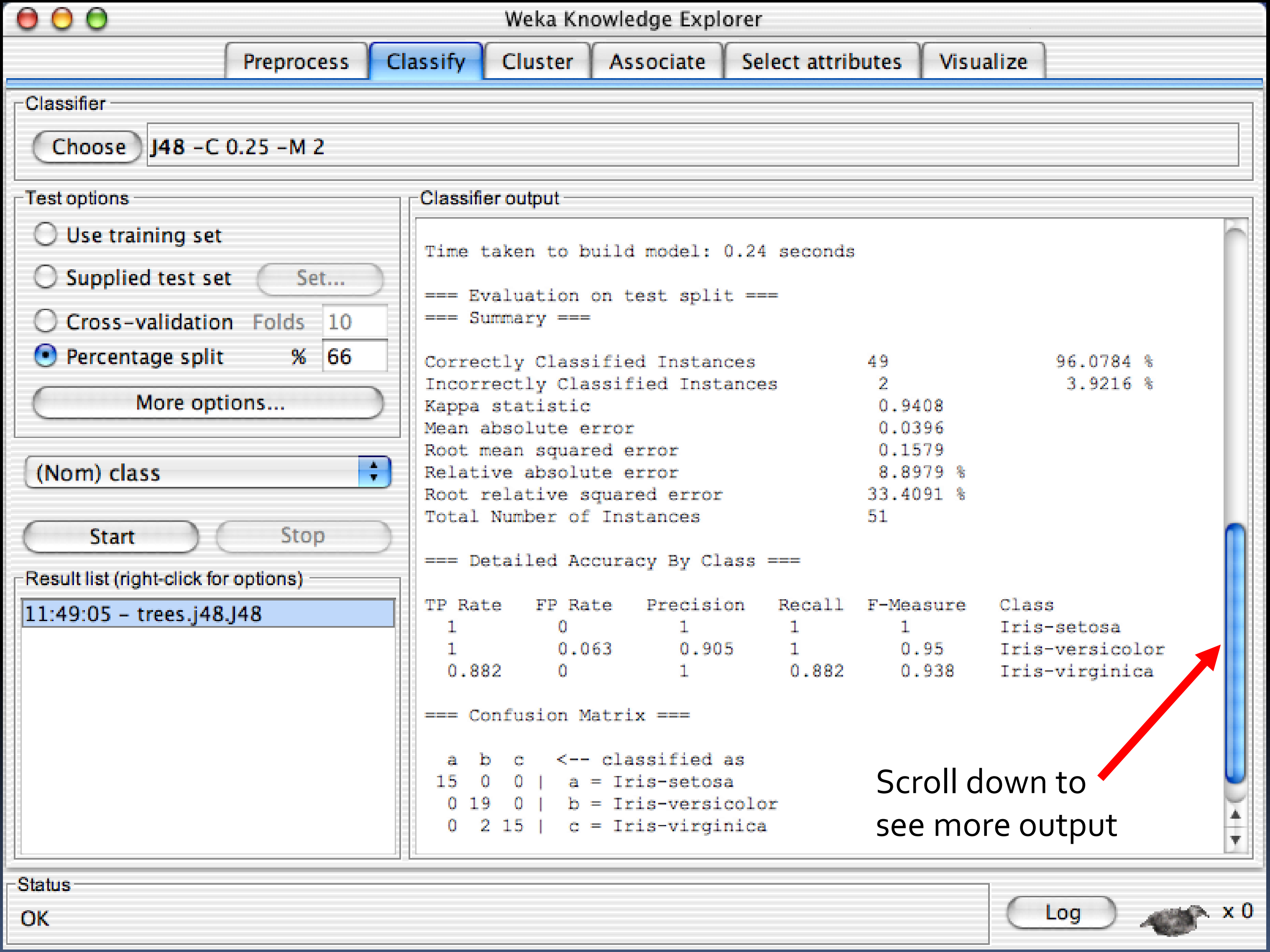
Status

OK

Records command line with options

Textual tree representation

Click Start



Classifier: J48 -C 0.25 -M 2

Test options

○ Use training set

○ Supplied test set

○ Cross-validation Folds 10

● Percentage split % 66

(Nom) class

Result list (right-click for options)

11:49:05 - trees.j48.J48

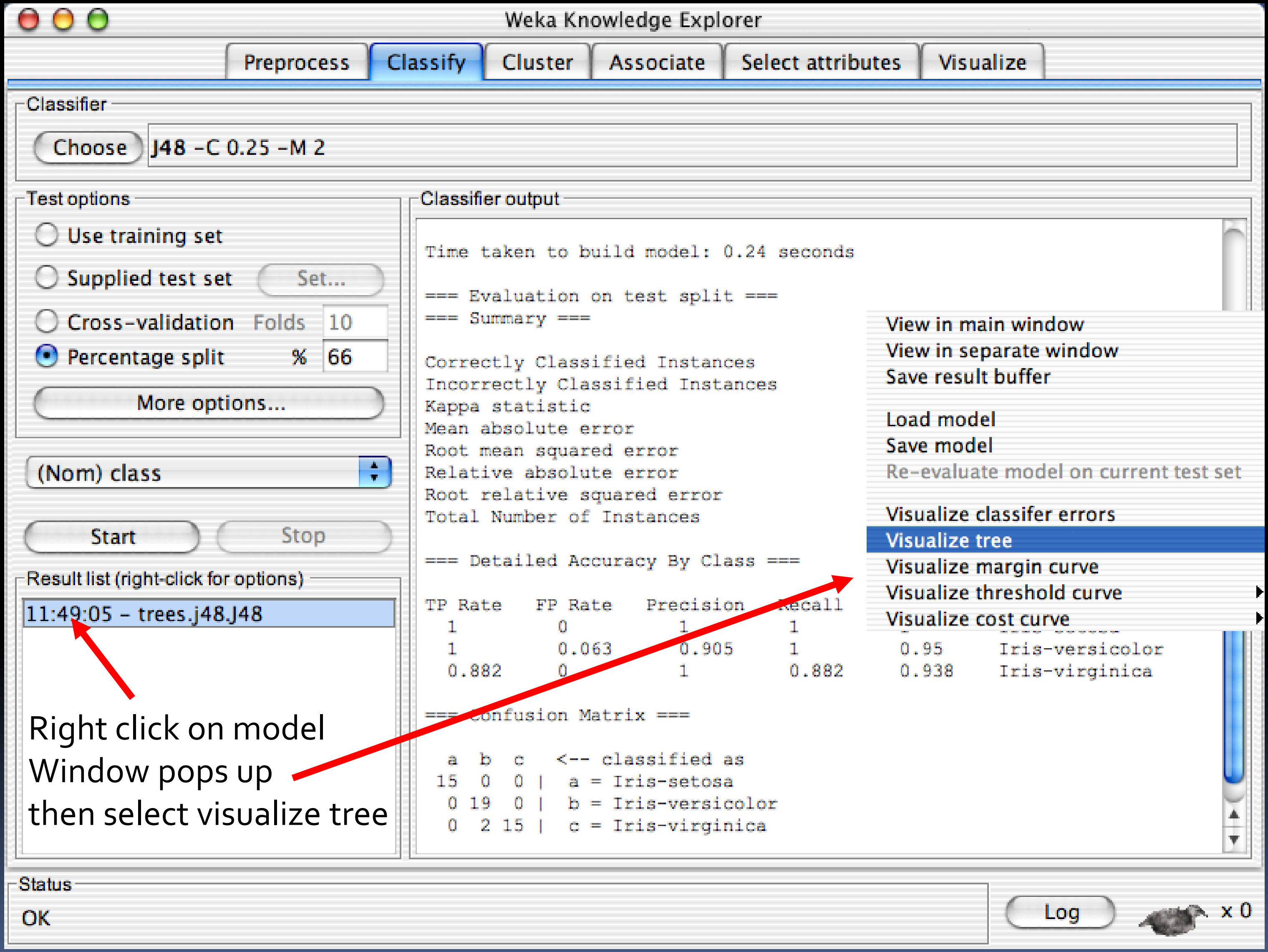
Classifier output

```
Time taken to build model: 0.24 seconds

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

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

=== Detailed Accuracy By Class ===

TP Rate   FP Rate   Precision   Recall  F-Measure   Class
  1         0          1         1         1        Iris-setosa
  1         0.063      0.905     1         0.95     Iris-versicolor
  0.882     0          1         0.882     0.938    Iris-virginica

=== Confusion Matrix ===

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

Scroll down to see more output

Status

OK



Classifier: J48 -C 0.25 -M 2

Test options

○ Use training set

○ Supplied test set

○ Cross-validation Folds 10

● Percentage split % 66

(Nom) class

Result list (right-click for options)

11:49:05 - trees.j48.J48

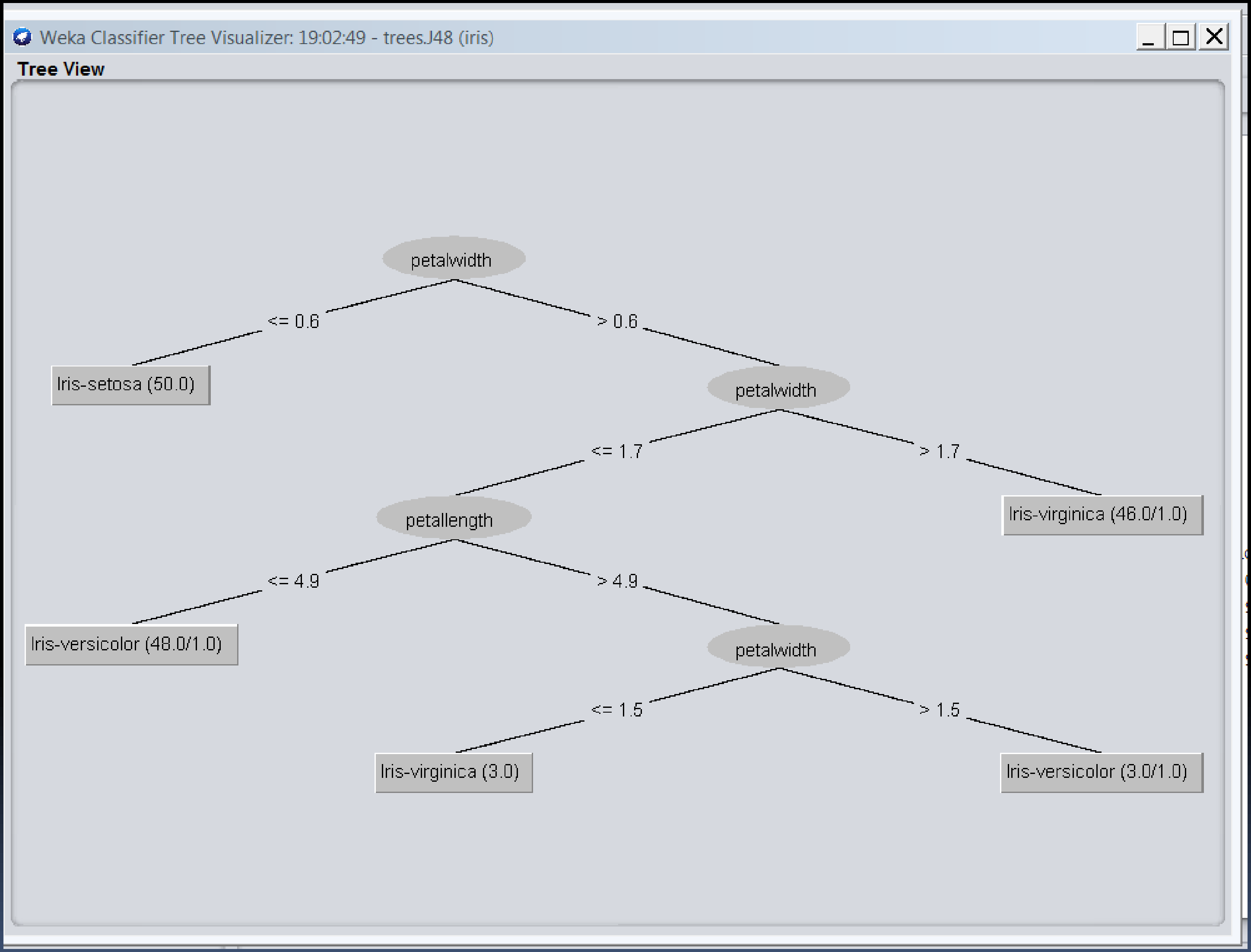
Classifier output

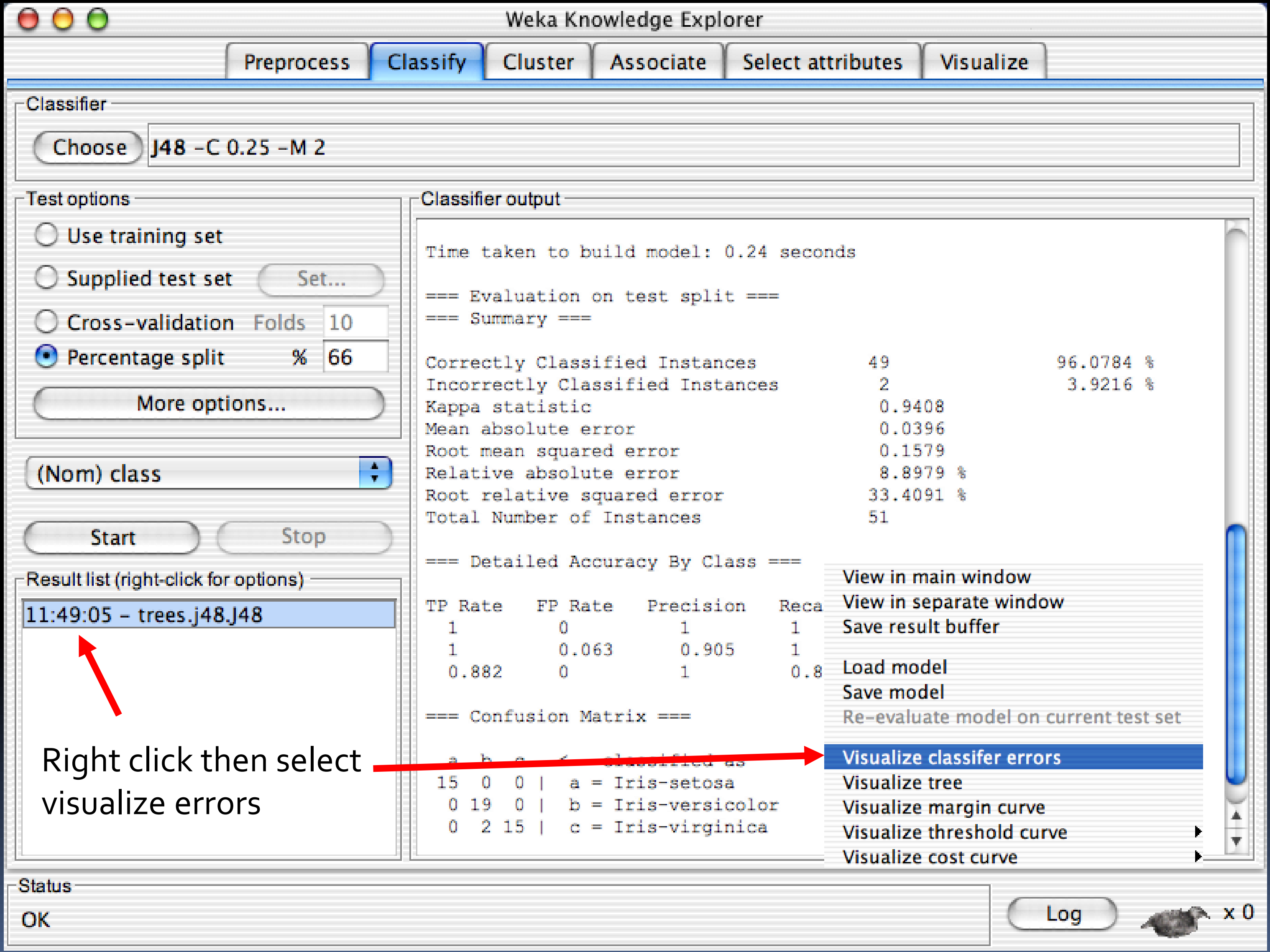
```
Time taken to build model: 0.24 seconds

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

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

=== Detailed Accuracy By Class ===

TP Rate   FP Rate   Precision   Recall
  1         0         1           1
  1         0.063     0.905       1          0.95      Iris-versicolor
  0.882     0         1           0.882      0.938     Iris-virginica

=== Confusion Matrix ===

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

Status

OK

Right click on model
Window pops up
then select visualize tree


Weka Classifier Tree Visualizer: 19:02:49 - trees.J48 (iris)
Tree View
petalwidth
<= 0.6
> 0.6
Iris-setosa (50.0)
petalwidth
<= 1.7
> 1.7
Iris-virginica (46.0/1.0)
petallength
<= 4.9
> 4.9
Iris-versicolor (48.0/1.0)
petalwidth
<= 1.5
> 1.5
Iris-virginica (3.0)
Iris-versicolor (3.0/1.0)



Classifier: J48 -C 0.25 -M 2

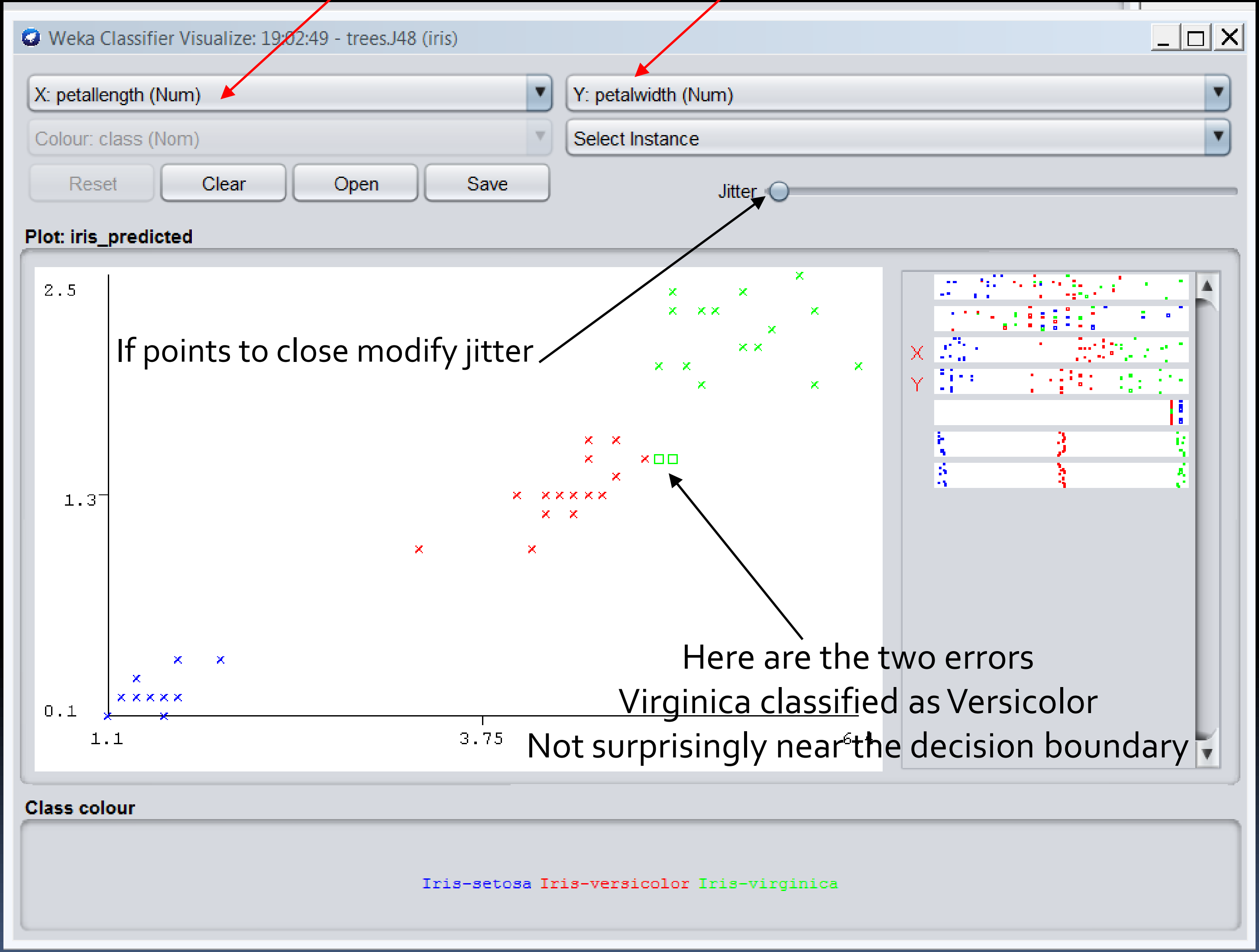
Cross-validation Folds: 10

Percentage split %: 66

```
Time taken to build model: 0.24 seconds

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

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

=== Detailed Accuracy By Class ===

TP Rate   FP Rate   Precision   Reca
  1         0          1          1
  1         0.063      0.905      1
  0.882     0          1          0.8

=== Confusion Matrix ===

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

Right click then select visualize errors

Set X to petallength and Y to petalwidth



If points to close modify jitter

Here are the two errors
Virginica classified as Versicolor
Not surprisingly near the decision boundary

Weka-3 Review: Decision Trees

- Covered an example in detail
- You should be able to build a decision tree classifier and understand the results
- You should be able to find the necessary documentation to understand and set the various parameters

Weka-4: Other Classifiers and Feature Selection

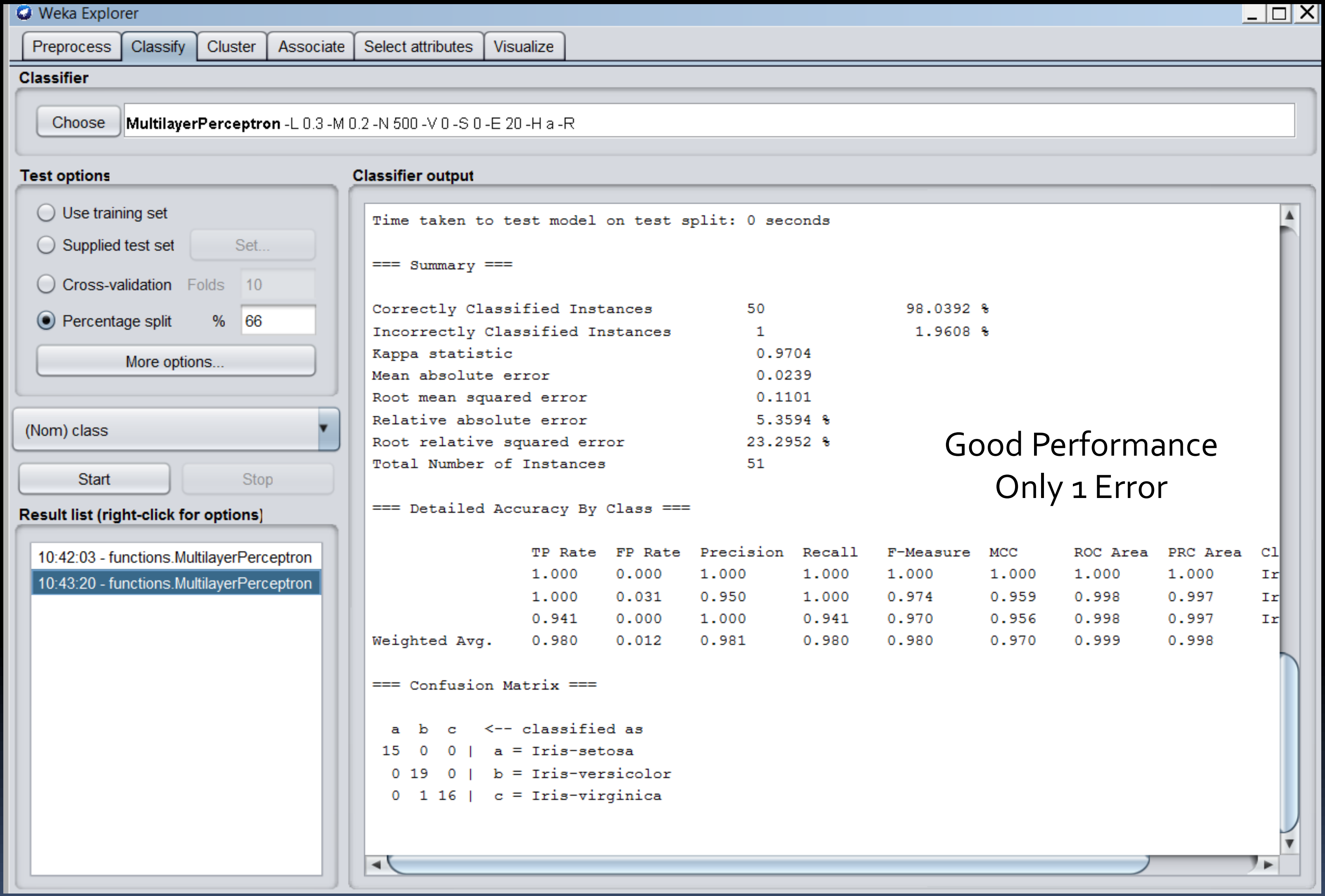
- Briefly explore two more classifiers
  - Artificial Neural Networks
  - Nearest Neighbor
- Demonstrate that with Weka it is easy to run any classifier and can treat like “Black Box”
- Briefly introduce WEKA feature selection

Artificial Neural Networks (ANNs)

- ANNs covered later in course
- Inspired by the brain
  - Simple processing nodes connected together
  - Each node has a weight and learning occurs by adjusting the weights to get desired output
- ANNs also called Multilayer Perceptrons
- Examples of universal function approximators

Creating ANN in WEKA for Iris

- While processing Iris data set, choose ANN:
  - In Classify tab click "Choose" button
  - Navigate to "Classifiers" and then "functions"
  - Select "Multilayer Perceptron"
  - Make sure the training/test is still set 66% training
  - Click "Start" to run ANN (use default options)



Classifier

MultilayerPerceptron -L 0.3 -M 0.2 -N 500 -V 0 -S 0 -E 20 -H a -R

Test options

○ Use training set

○ Supplied test set

○ Cross-validation Folds 10

● Percentage split % 66

(Nom) class

Result list (right-click for options)

10:42:03 - functions.MultilayerPerceptron

10:43:20 - functions.MultilayerPerceptron

Classifier output

```
Time taken to test model on test split: 0 seconds

=== Summary ===

Correctly Classified Instances          50               98.0392 %
Incorrectly Classified Instances         1                1.9608 %
Kappa statistic                          0.9704
Mean absolute error                      0.0239
Root mean squared error                  0.1101
Relative absolute error                  5.3594 %
Root relative squared error             23.2952 %
Total Number of Instances               51

=== Detailed Accuracy By Class ===

                 TP Rate  FP Rate  Precision  Recall   F-Measure  MCC      ROC Area  PRC Area  Cl
                 1.000    0.000    1.000      1.000    1.000      1.000    1.000     1.000     Ir
                 1.000    0.031    0.950      1.000    0.974      0.959    0.998     0.997     Ir
                 0.941    0.000    1.000      0.941    0.970      0.956    0.998     0.997     Ir
Weighted Avg.    0.980    0.012    0.981      0.980    0.980      0.970    0.999     0.998

=== Confusion Matrix ===

  a  b  c   <-- classified as
 15  0  0 |  a = Iris-setosa
  0 19  0 |  b = Iris-versicolor
  0  1 16 |  c = Iris-virginica
```

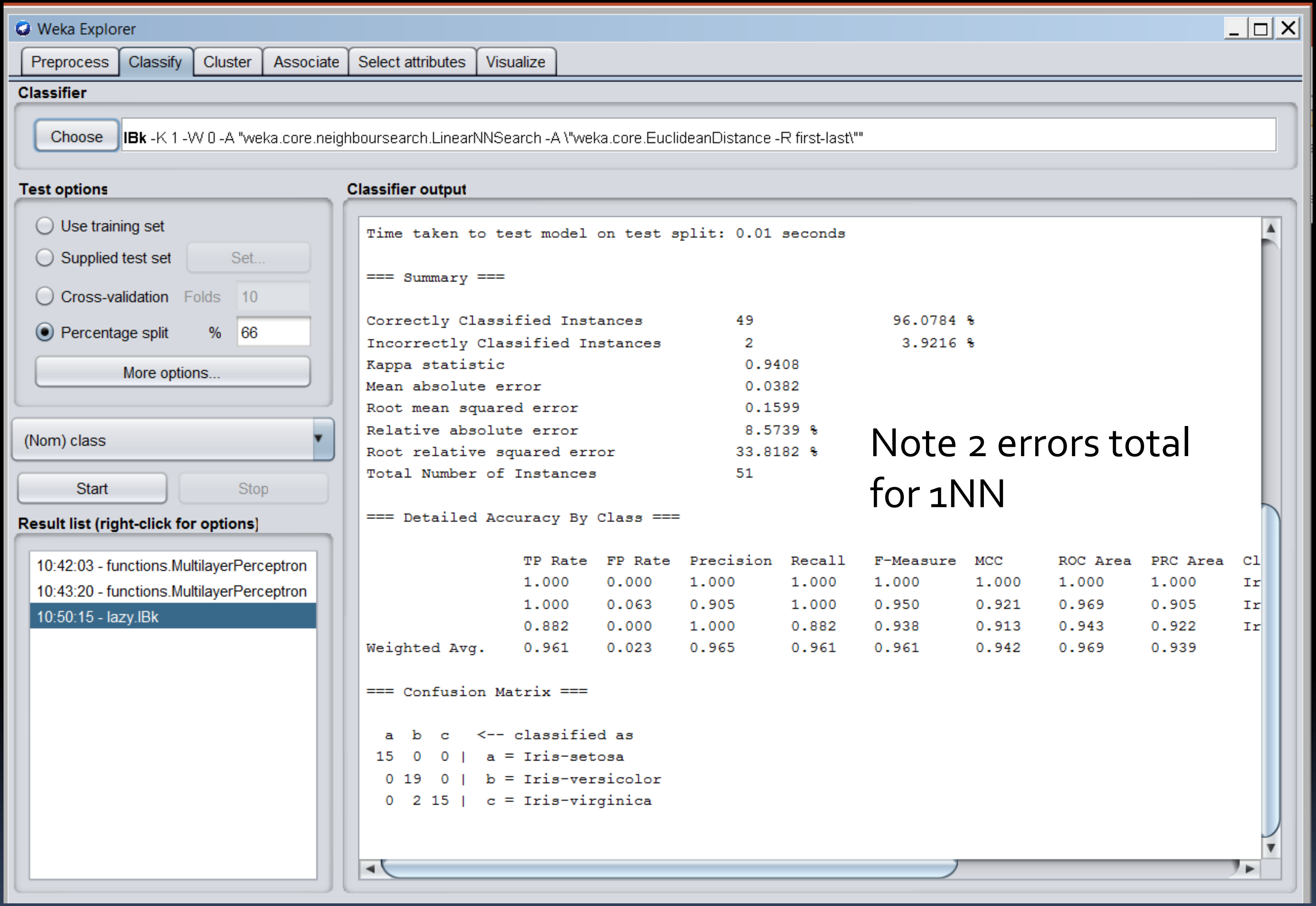
Good Performance
Only 1 Error

Nearest Neighbor Classification

- Nearest Neighbor classifies example based on most similar instance(s)
  - Relators use this method to price your home
  - Also called instance-based learning
  - *k* used to specify number of nearest neighbors
    - Common values 1, 3, 5
    - Method sometimes referred to as *k*NN
    - Weka uses the IB*k* algorithm (IB=Instance Based)
  - Also called "Lazy Learning"
    - No work is done up front to build model
    - All work done at classification time

Creating IBk Model with Weka

- While processing Iris data set, choose IBk:
    - In Classify tab, click “Choose” button
    - Navigate to “Classifiers” and then “Lazy”
    - Select “IBk”
        - Keep all of the default options
    - Make sure the training/test is still set 66% training
    - Click “Start” to run method
        - Results on next slide (uses default k=1)
        - Results for k=5 on slide after that: try it by modifying the option



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

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

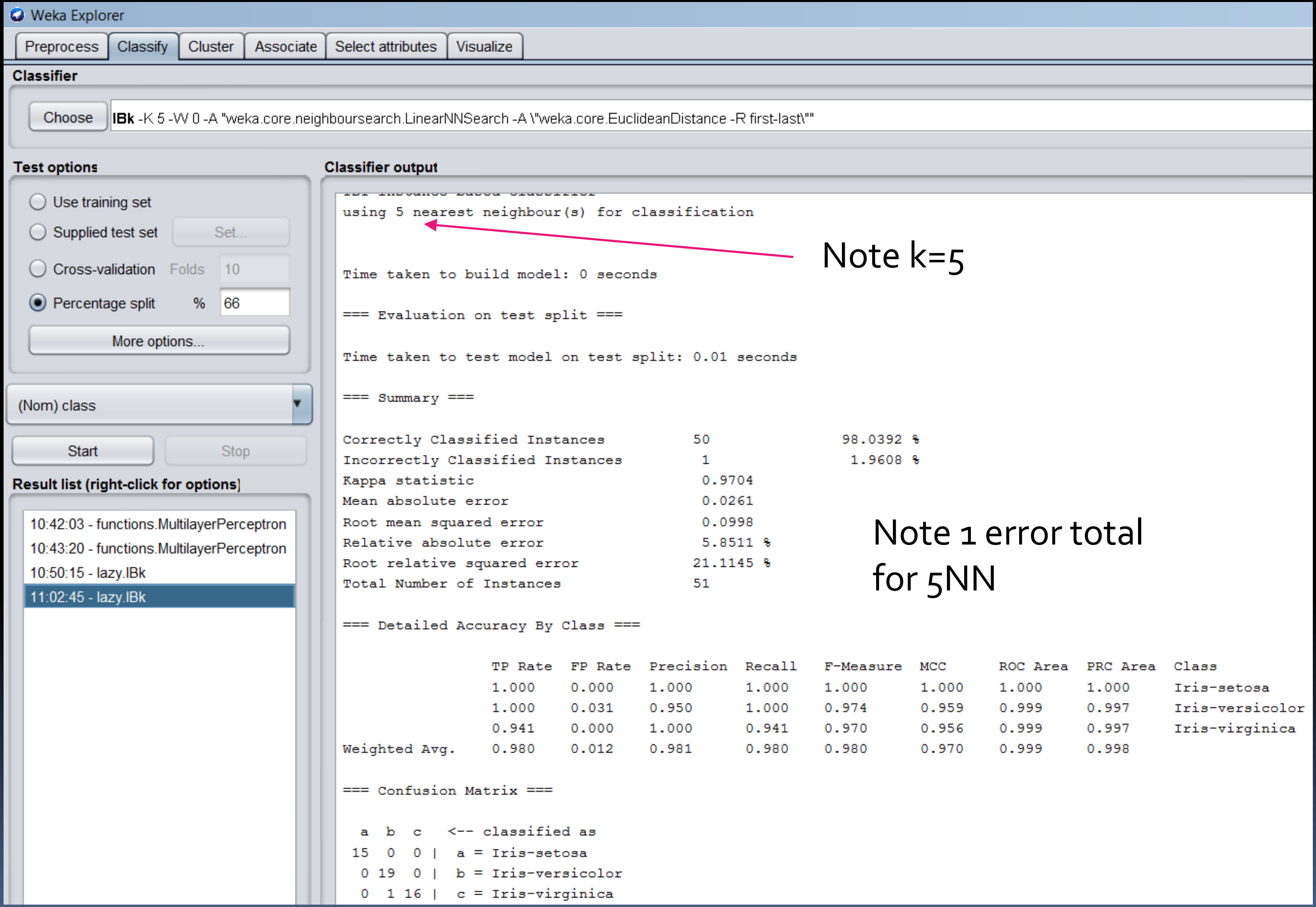
(Nom) class

Result list (right-click for options)

- 10:42:03 - functions.MultilayerPerceptron
- 10:43:20 - functions.MultilayerPerceptron
- 10:50:15 - lazy.IBk

```
Time taken to test model on test split: 0.01 seconds

=== Summary ===

Correctly Classified Instances          49               96.0784 %
Incorrectly Classified Instances         2                3.9216 %
Kappa statistic                          0.9408
Mean absolute error                      0.0382
Root mean squared error                  0.1599
Relative absolute error                  8.5739 %
Root relative squared error             33.8182 %
Total Number of Instances               51

=== Detailed Accuracy By Class ===

                 TP Rate  FP Rate  Precision  Recall   F-Measure  MCC      ROC Area  PRC Area  Cl
                 1.000    0.000    1.000      1.000    1.000      1.000    1.000     1.000     Ir
                 1.000    0.063    0.905      1.000    0.950      0.921    0.969     0.905     Ir
                 0.882    0.000    1.000      0.882    0.938      0.913    0.943     0.922     Ir
Weighted Avg.    0.961    0.023    0.965      0.961    0.961      0.942    0.969     0.939

=== Confusion Matrix ===

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

Note 2 errors total for 1NN



Classifier

IBk -K 5 -W 0 -A "weka.core.neighboursearch.LinearNNSearch -A \"weka.core.EuclideanDistance -R first-last\""

Test options

○ Use training set

○ Supplied test set

○ Cross-validation Folds 10

● Percentage split % 66

(Nom) class

Result list (right-click for options)

10:42:03 - functions.MultilayerPerceptron
10:43:20 - functions.MultilayerPerceptron
10:50:15 - lazy.IBk
11:02:45 - lazy.IBk

Classifier output

```
IB1 instance-based classifier
using 5 nearest neighbour(s) for classification


Time taken to build model: 0 seconds

=== Evaluation on test split ===

Time taken to test model on test split: 0.01 seconds

=== Summary ===

Correctly Classified Instances          50               98.0392 %
Incorrectly Classified Instances         1                1.9608 %
Kappa statistic                          0.9704
Mean absolute error                      0.0261
Root mean squared error                  0.0998
Relative absolute error                  5.8511 %
Root relative squared error             21.1145 %
Total Number of Instances               51

=== Detailed Accuracy By Class ===

                 TP Rate  FP Rate  Precision  Recall   F-Measure  MCC      ROC Area  PRC Area  Class
                 1.000    0.000    1.000      1.000    1.000      1.000    1.000     1.000     Iris-setosa
                 1.000    0.031    0.950      1.000    0.974      0.959    0.999     0.997     Iris-versicolor
                 0.941    0.000    1.000      0.941    0.970      0.956    0.999     0.997     Iris-virginica
Weighted Avg.    0.980    0.012    0.981      0.980    0.980      0.970    0.999     0.998

=== Confusion Matrix ===

  a  b  c   <-- classified as
 15  0  0 |  a = Iris-setosa
  0 19  0 |  b = Iris-versicolor
  0  1 16 |  c = Iris-virginica
```

Note k=5

Note 1 error total for 5NN

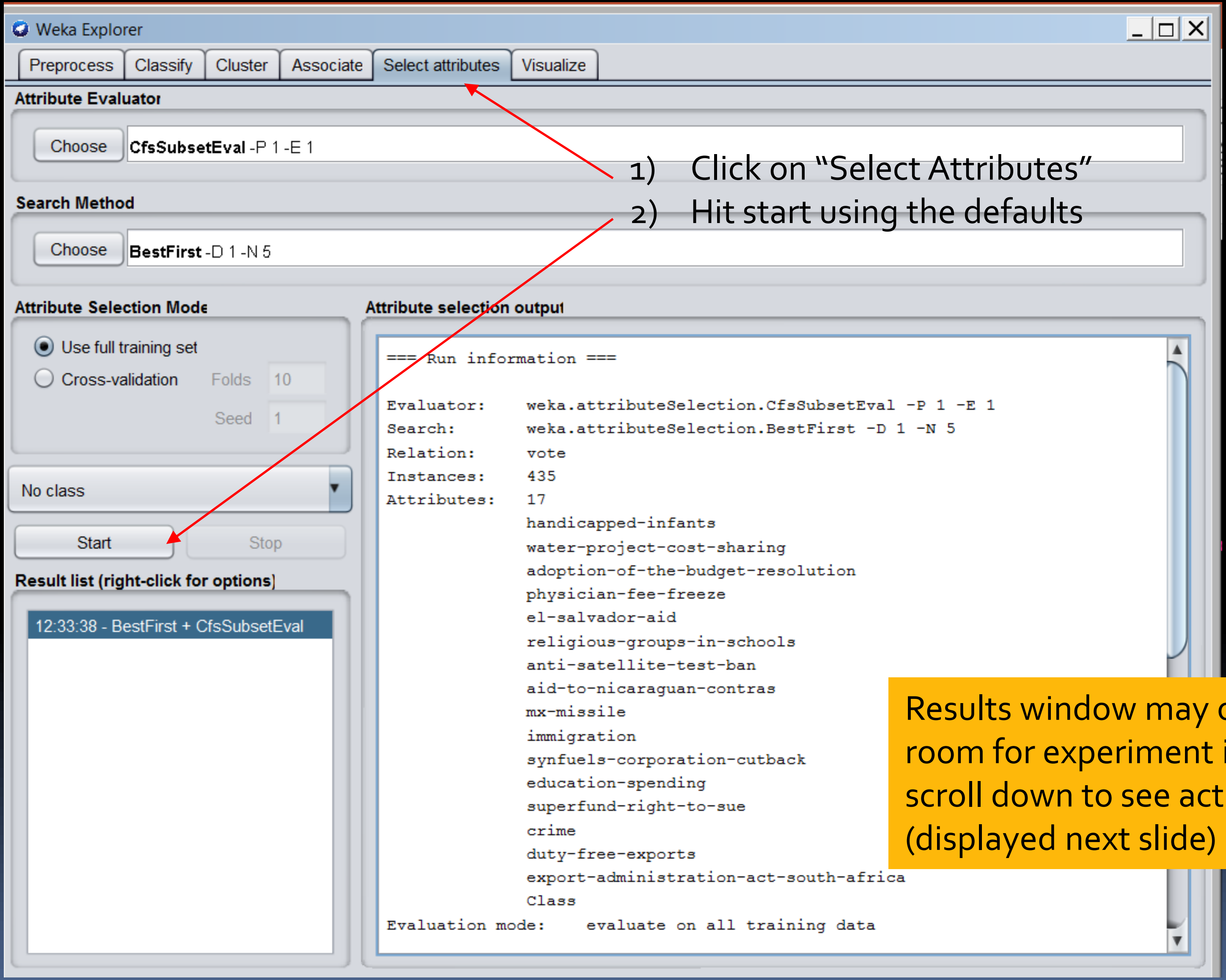
Run a Few Classifiers on Own

- Run Random Forest
  - A tree ensemble method that is under Trees
    - It builds lots of trees using slightly different training examples and features and then combines them
    - We will study them and other ensembles later in course
  - The *numIterations* parameter controls # of trees
- Run Bagging
  - An ensemble (meta learning) method, so find it under classifiers→meta
    - Bagging runs base classifier repeatedly with different training data
    - Run with default classifier and then change classifier to J48

Weka Feature Selection

- Explore simple feature selection methods
- Use Vote Dataset that has more features
  - Go to Preprocess tab and load using url:
    - https://storm.cis.fordham.edu/~gweiss/data-mining/weka-data/vote.arff
    - Feature selection is needed for methods that cannot handle irrelevant or redundant features
- Experiments:
  - Best first forward search
    - Pick best feature and then add more, but with backtracking to explore more of the space
  - Info gain with ranking
    - Rank the features based on their information gain

Best First Forward Search for feature subset



CfsSubsetEval -P 1 -E 1

BestFirst -D 1 -N 5

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

No class

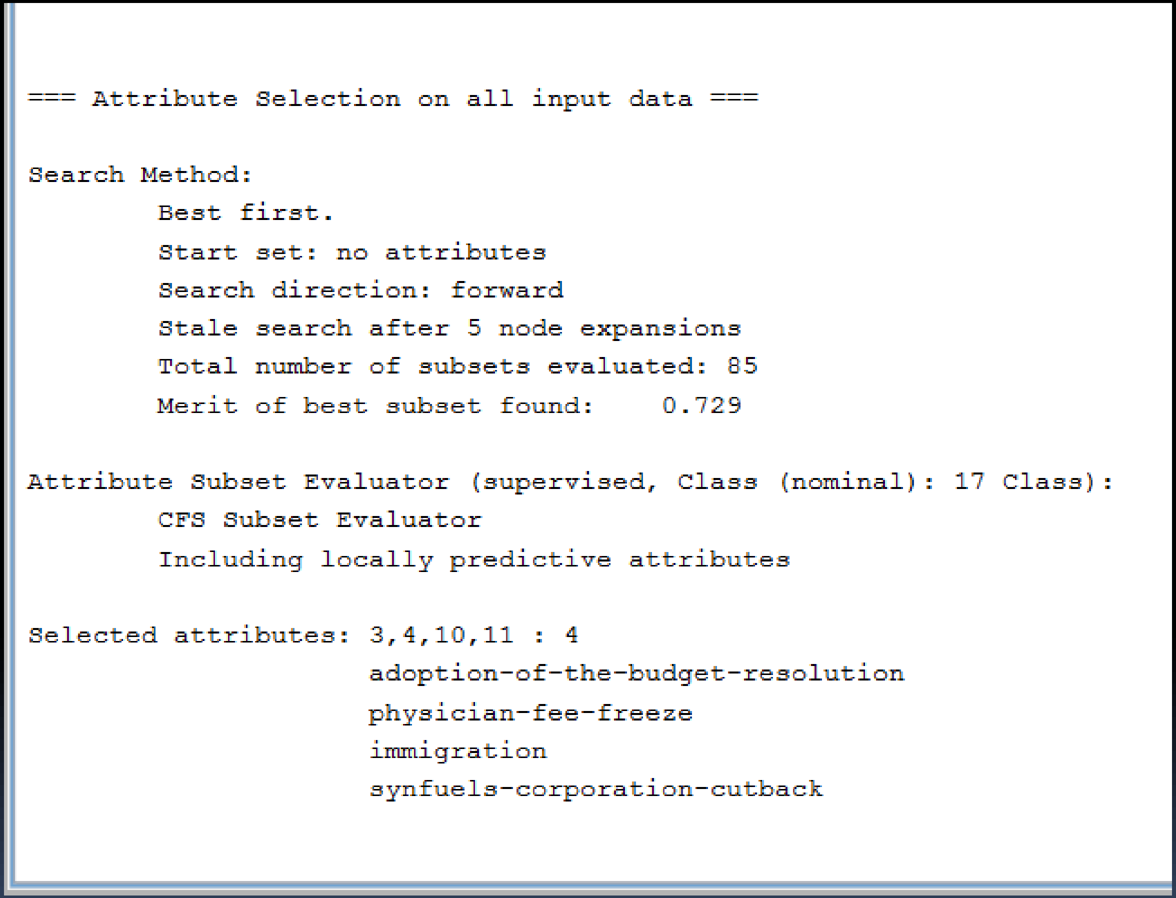
12:33:38 - BestFirst + CfsSubsetEval

```
=== Run information ===

Evaluator:    weka.attributeSelection.CfsSubsetEval -P 1 -E 1
Search:       weka.attributeSelection.BestFirst -D 1 -N 5
Relation:     vote
Instances:    435
Attributes:   17
              handicapped-infants
              water-project-cost-sharing
              adoption-of-the-budget-resolution
              physician-fee-freeze
              el-salvador-aid
              religious-groups-in-schools
              anti-satellite-test-ban
              aid-to-nicaraguan-contras
              mx-missile
              immigration
              synfuels-corporation-cutback
              education-spending
              superfund-right-to-sue
              crime
              duty-free-exports
              export-administration-act-south-africa
              Class
Evaluation mode:    evaluate on all training data
```

1) Click on "Select Attributes"
2) Hit start using the defaults

Results window may only have room for experiment info so scroll down to see actual results (displayed next slide)

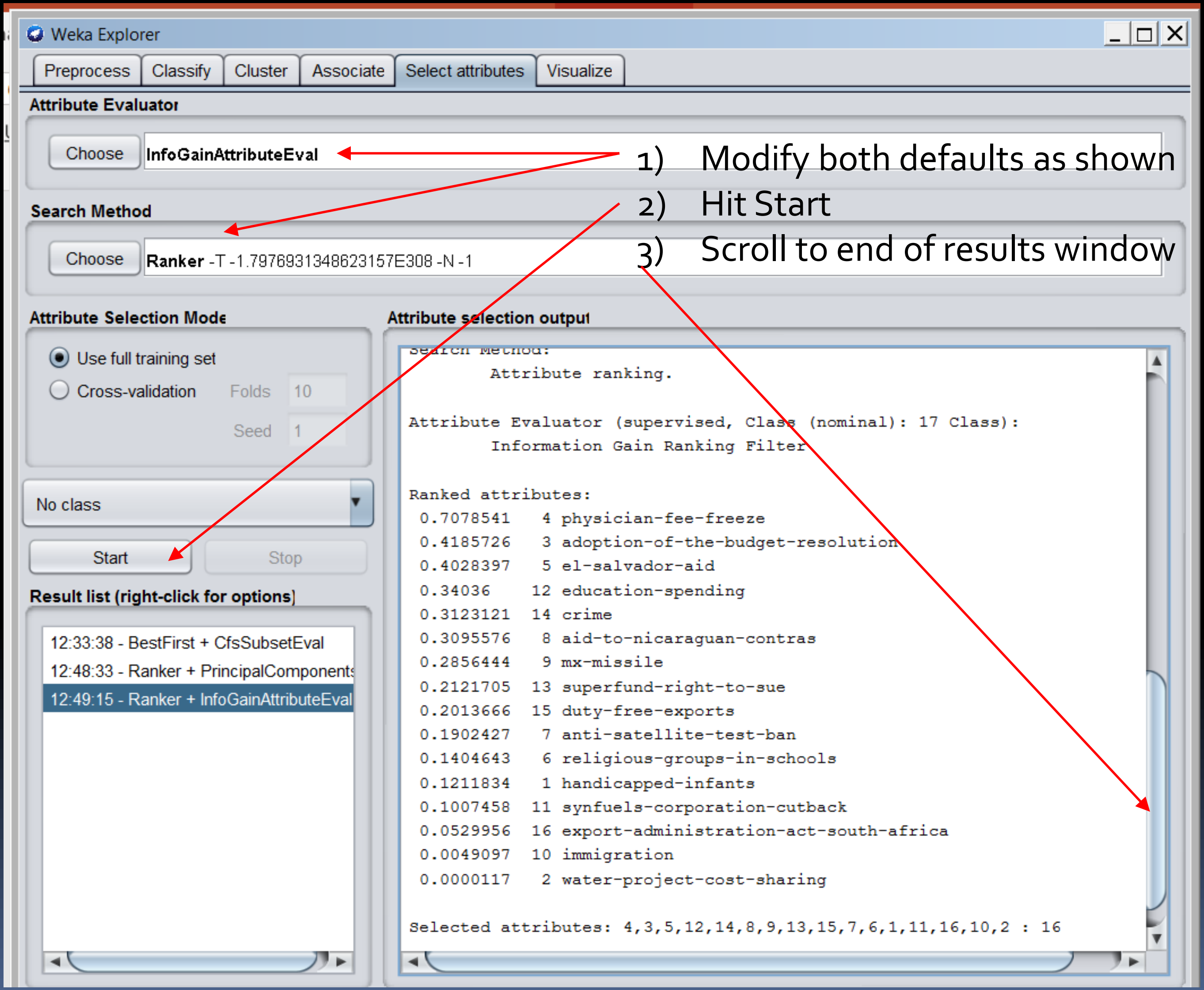
Best First Search Subset Selection



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

Search Method:
        Best first.
        Start set: no attributes
        Search direction: forward
        Stale search after 5 node expansions
        Total number of subsets evaluated: 85
        Merit of best subset found:    0.729

Attribute Subset Evaluator (supervised, Class (nominal): 17 Class):
        CFS Subset Evaluator
        Including locally predictive attributes

Selected attributes: 3,4,10,11 : 4
                     adoption-of-the-budget-resolution
                     physician-fee-freeze
                     immigration
                     synfuels-corporation-cutback
```

4 attributes selected of the 17

Rank Features by Info Gain



1) Modify both defaults as shown
2) Hit Start
3) Scroll to end of results window

Attribute Evaluator: InfoGainAttributeEval

Search Method: Ranker -T -1.7976931348623157E308 -N -1

Attribute Selection Mode:
● Use full training set
○ Cross-validation Folds 10 Seed 1

No class

Result list (right-click for options)
- 12:33:38 - BestFirst + CfsSubsetEval
- 12:48:33 - Ranker + PrincipalComponents
- 12:49:15 - Ranker + InfoGainAttributeEval

Attribute selection output

```
Search Method:
        Attribute ranking.

Attribute Evaluator (supervised, Class (nominal): 17 Class):
        Information Gain Ranking Filter

Ranked attributes:
 0.7078541    4 physician-fee-freeze
 0.4185726    3 adoption-of-the-budget-resolution
 0.4028397    5 el-salvador-aid
 0.34036     12 education-spending
 0.3123121   14 crime
 0.3095576    8 aid-to-nicaraguan-contras
 0.2856444    9 mx-missile
 0.2121705   13 superfund-right-to-sue
 0.2013666   15 duty-free-exports
 0.1902427    7 anti-satellite-test-ban
 0.1404643    6 religious-groups-in-schools
 0.1211834    1 handicapped-infants
 0.1007458   11 synfuels-corporation-cutback
 0.0529956   16 export-administration-act-south-africa
 0.0049097   10 immigration
 0.0000117    2 water-project-cost-sharing

Selected attributes: 4,3,5,12,14,8,9,13,15,7,6,1,11,16,10,2 : 16
```

Review Weka-4: Other Classifiers and Feature Selection

- Built and evaluated neural network and nearest neighbor models
  - You should be able to use Weka to run any classification model
- You should understand how to run feature selection using Weka
  - It is okay if you do not understand the details of each method
    - Can look them up in the documentation

Weka-5: KnowledgeFlow Interface

- Provides a drag and drop interface
  - Each step is a node that you drop onto workspace
  - You connect nodes to create a process flow
- Advantage: supports multiple process flows
  - Can have a path with steps to preprocess data etc.
  - Then branch to build different classifiers
- Other DM tools provide similar interfaces
  - My early DM tools in industry had this interface
    - SAS Enterprise Miner
    - Clementine (now IBM SPSS Modeler)

Video Demonstrating KnowledgeFlow

- You are not required to try KnowledgeFlow
- Watch first 4 minutes of 7-minute video
  - It does not show how to use mouse to connect nodes, which can be a bit tricky
  - It conveys the basic ideas and power of interface
  - Link to Ian Witten's Youtube video:
    - https://youtu.be/sHSg0VX9z-8
    - Also listed under my Youtube playlist:
      - https://www.youtube.com/playlist?list=PLg2TXNAfR8_ukJY1fEPDMuXV43ZrBBpUh
        - Select "More Data Mining with Weka (1.4 .."

Review of DM with Weka Module

- You should have installed Weka
- Understand the different interfaces
- Using Explorer interface, be able to:
  - Load any data set
  - Visualize the features
  - Build any classifier and know how to learn about parameters and change them
  - Examine and interpret the results
  - Know how to perform feature selection
- Understand basics of the knowledgeFlow