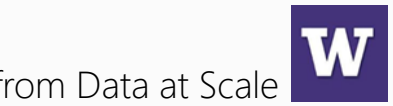
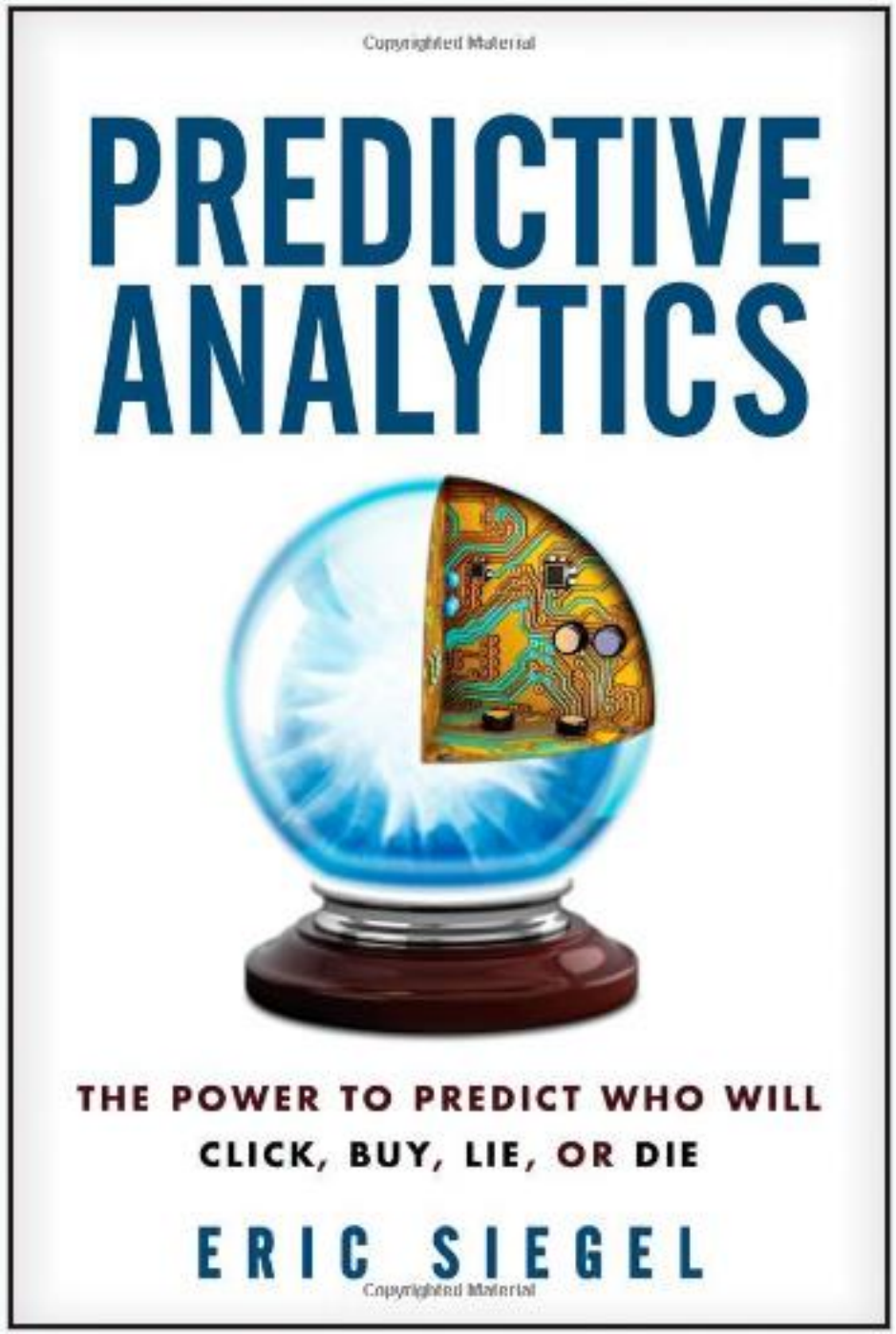
Data Science

Deriving Knowledge from Data at Scale

Valentine N. Fontama

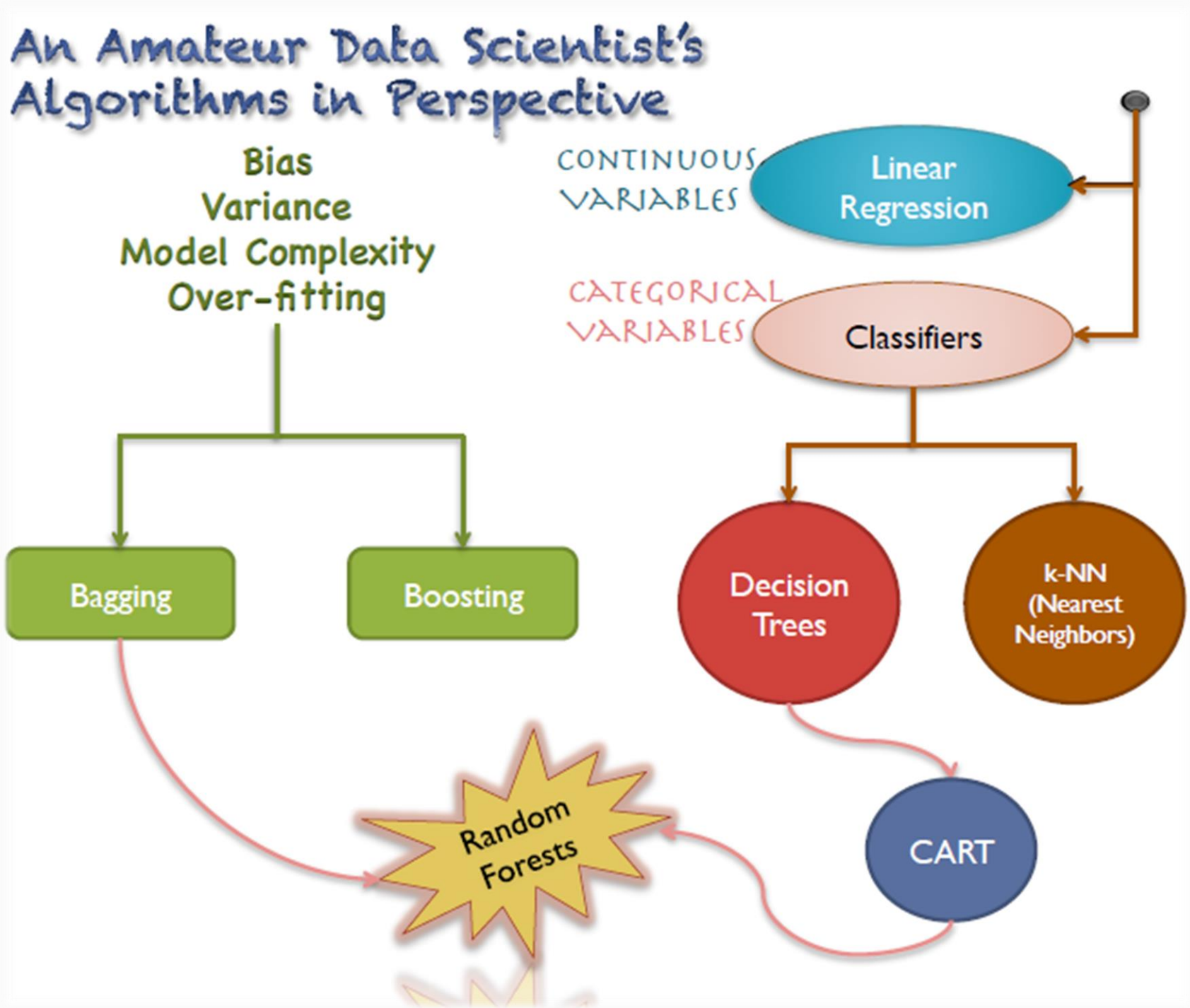
July 30th, 2015





Read this… A brilliant read that offers an accessible overview of predictive analytics, technical but at the same time a recreational read with ample practical examples, and it provides footnotes for further study…

I highly recommend it…



Review of Course Plan...

W5: Clustering Review
Clustering Assignment

W6: Feature Select/Create
SVMs & Regression
Data Prep Assignment
Kaggle Contest HW

W7: SVMs Cont'd

Lecture Outline

• Opening Discussion *Review Discussion…*	30 minutes
• Data ScienceHands On	60 minutes
• Break	5 minutes
• Data Science Modelling *Model performance evaluation…*	30 minutes
• Machine Learning Boot Camp *Clustering, k-Means…*	~60 minutes
• Close	

Lecture 4 Homework

Assignment: Decisions Trees and Classification

Due Date: July 30, 2015

1. Targeted Marketing Campaign

In this problem we will use historical data from past customer responses to build a classification model. In class next week we will apply the trained model to a new set of prospects to whom we may want extend an offer for a PEP. Rather than doing a mass marketing campaign to all new prospects, we would like to target those that are likely to respond positively to our offer (according to our classification model).

There is one data sets available, comma delimited, and the first row contains the field names):

- bank-data.csv - Preclassified training data Set for Building a Model

Clustering

Fundamental Concepts: Calculating similarity of objects described by data; Using similarity for prediction; Clustering as similarity-based segmentation.

Exemplary Techniques: Searching for similar entities; Nearest neighbor methods; Clustering methods; Distance metrics for calculating similarity.

Clustering

- Cluster images based on key features most useful for identification
- Use Cluster ID as index over images for exact match & retrieval

Clustering

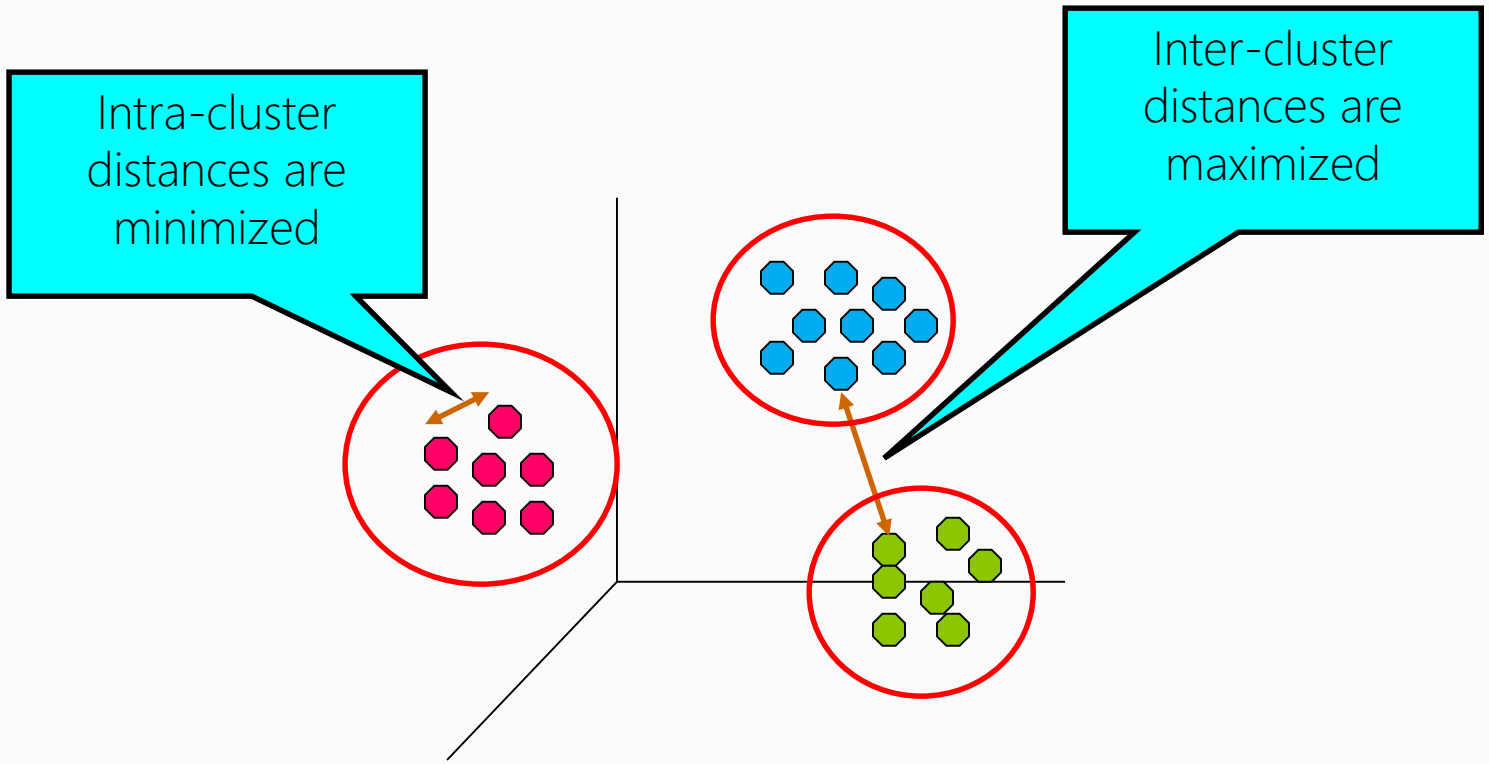
We may want to ***retrieve*** similar things directly. For example, IBM wants to find companies that are *similar to their best business customers*, in order to have sales staff look at them as prospects. Hewlett-Packard maintains many high performance servers for clients; this maintenance is aided by a tool that, given a server configuration, *retrieves information on other similarly configured servers*.

We may want to group similar items together into clusters, for example to see whether our *customer base contains groups of similar customers* and what these groups have in common.

Reasoning from similar cases of course extends beyond business applications; it is natural to fields such as medicine and law. A doctor may *reason about a new difficult case by recalling a similar case and its diagnosis*. A lawyer often argues cases by citing legal precedents, which are *similar historical cases whose dispositions were previously judged and entered into the legal casebook*.

What is clustering?

A grouping of data objects such that the objects within a group are similar (or related) to one another and different from (or unrelated to) the objects in other groups

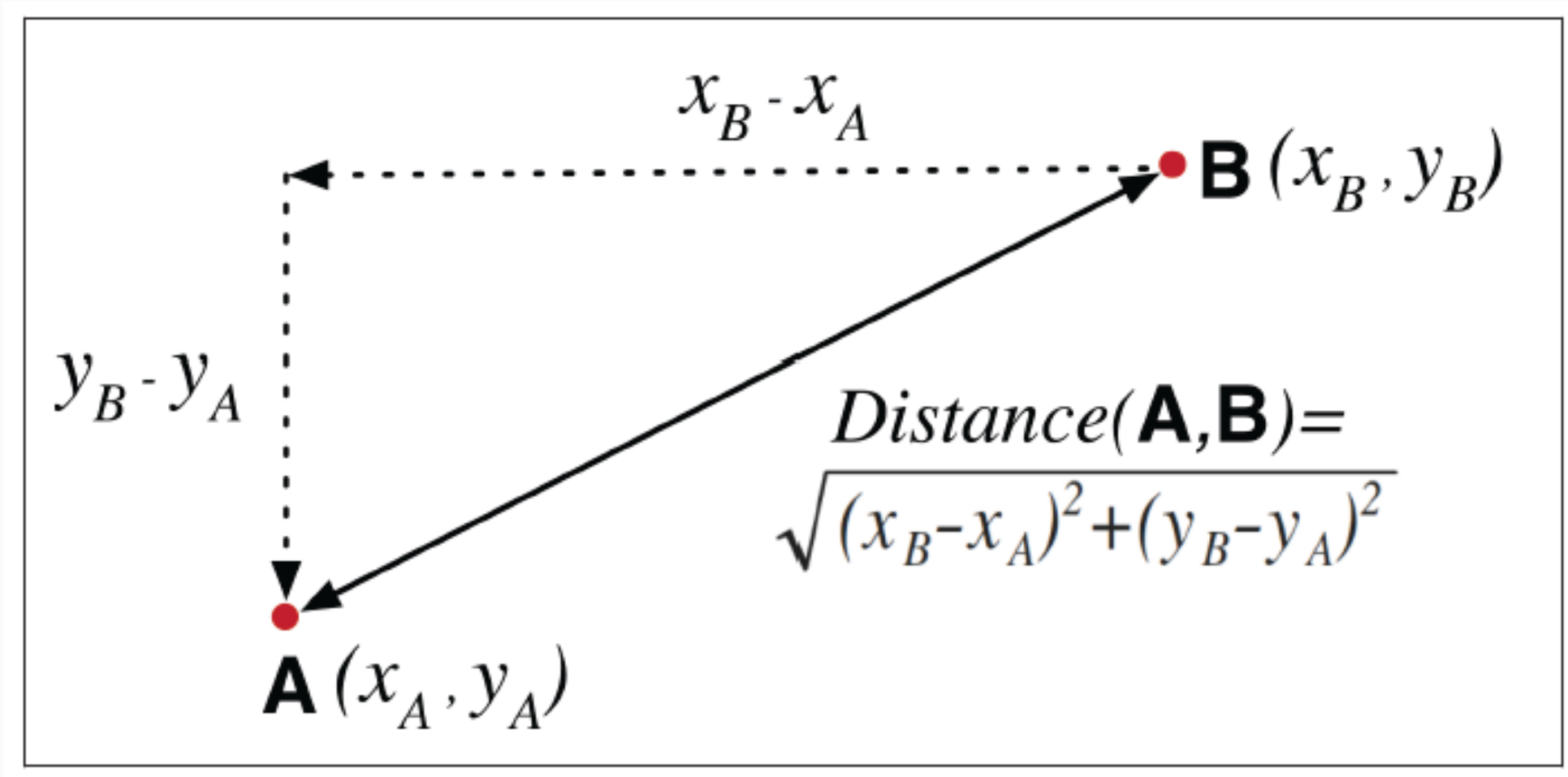


How do we define “similarity”?

- Goal is to group together “similar” data – but what does this mean?
- No single answer, it depends on what we want to find or emphasize in the data; this is one reason why clustering is an “art”
- The similarity measure is often more important than the clustering algorithm used – don’t overlook this choice!
- There is no golden standard, depends on goal: data reduction, “natural clusters”, “useful” clusters, outlier detection, etc.

Similarity Measures

- Euclidean Distance



Similarity Measures

Manhattan Distance

$$d_{\text{Manhattan}}(\mathbf{X},\mathbf{Y}) = \| \mathbf{X} - \mathbf{Y} \|_1 = | x_1 - y_1 | + | x_2 - y_2 | + \cdots$$

Jaccard Distance

$$d_{\text{Jaccard}}(X, Y) = 1 - \frac{| X \cap Y |}{| X \cup Y |}$$

Edit Distance

1. `1113 Bleaker St.`
2. `113 Bleecker St.`

1. Delete a 1,
2. Insert a c, and
3. Replace an a with an e.

Lecture Outline

- Opening Discussion — 30 minutes
  Review Discussion…
- **Data ScienceHands On — 60 minutes**
- Break — 5 minutes
- Data Science Modelling — 30 minutes
  Model performance evaluation…
- Machine Learning Boot Camp — ~60 minutes
  Clustering, k-Means…
- Close

WEKA Explorer: clustering data

WEKA contains "clusterers" for finding groups of similar instances in a dataset

Implemented schemes are:

- *k*-Means, EM, Cobweb, *X*-means, FarthestFirst;

Clusters can be visualized and compared to "true" clusters (if given);

WEKA Explorer: clustering data (optional reading)

International Journal of Emerging Technology and Advanced Engineering
Website: www.ijetae.com (ISSN 2250-2459, Volume 2, Issue 5, May 2012)

Comparison the various clustering algorithms of weka tools

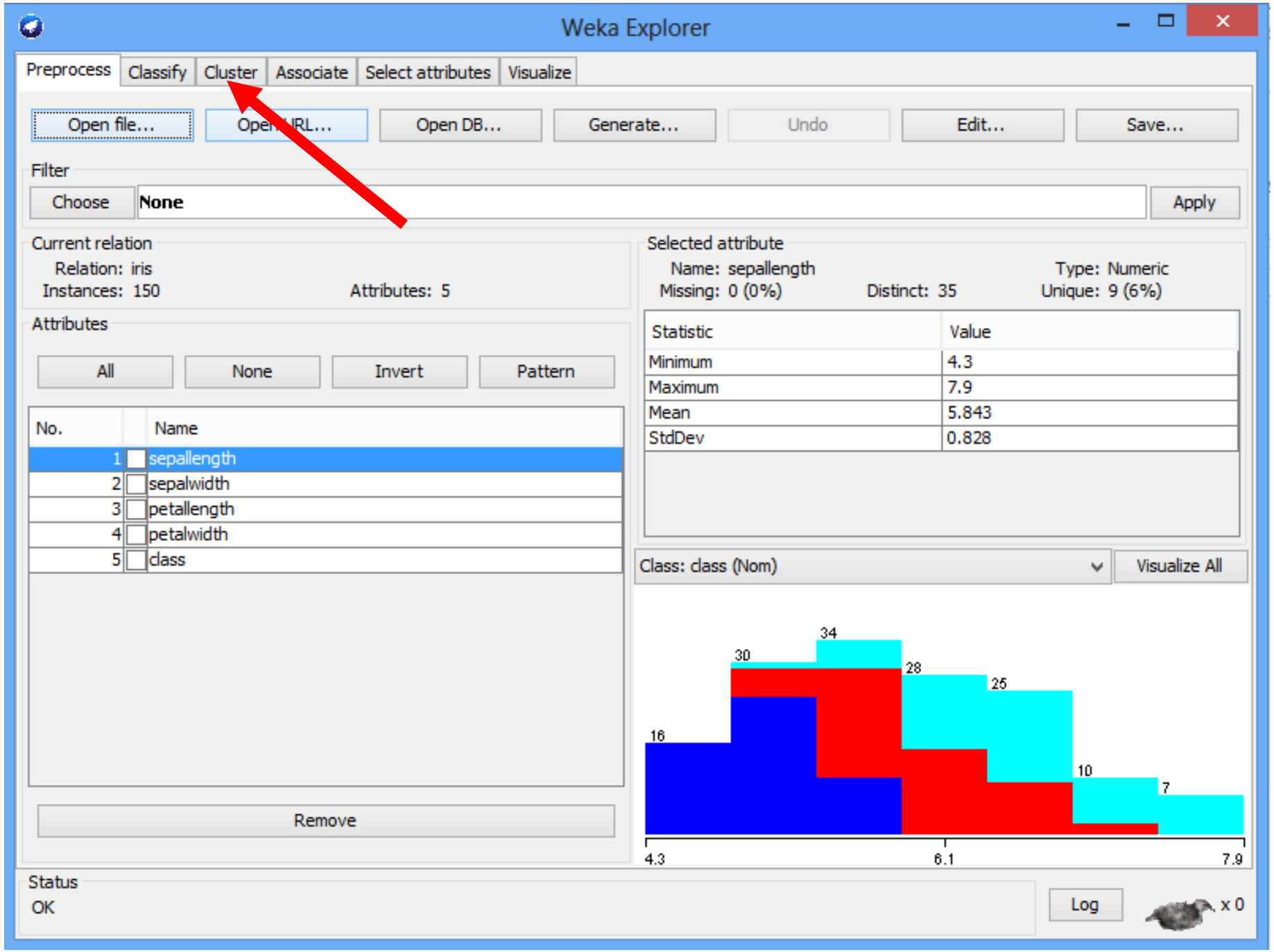
Narendra Sharma [1], Aman Bajpai [2], Mr. Ratnesh Litoriya [3]
[1,2,3] *Department of computer science, Jaypee University of Engg. & Technology*
[1] narendra_sharma88@yahoo.com
[2] amanbajpai97@gmail.com
[3] ratneshlitoriya@yahoo.com

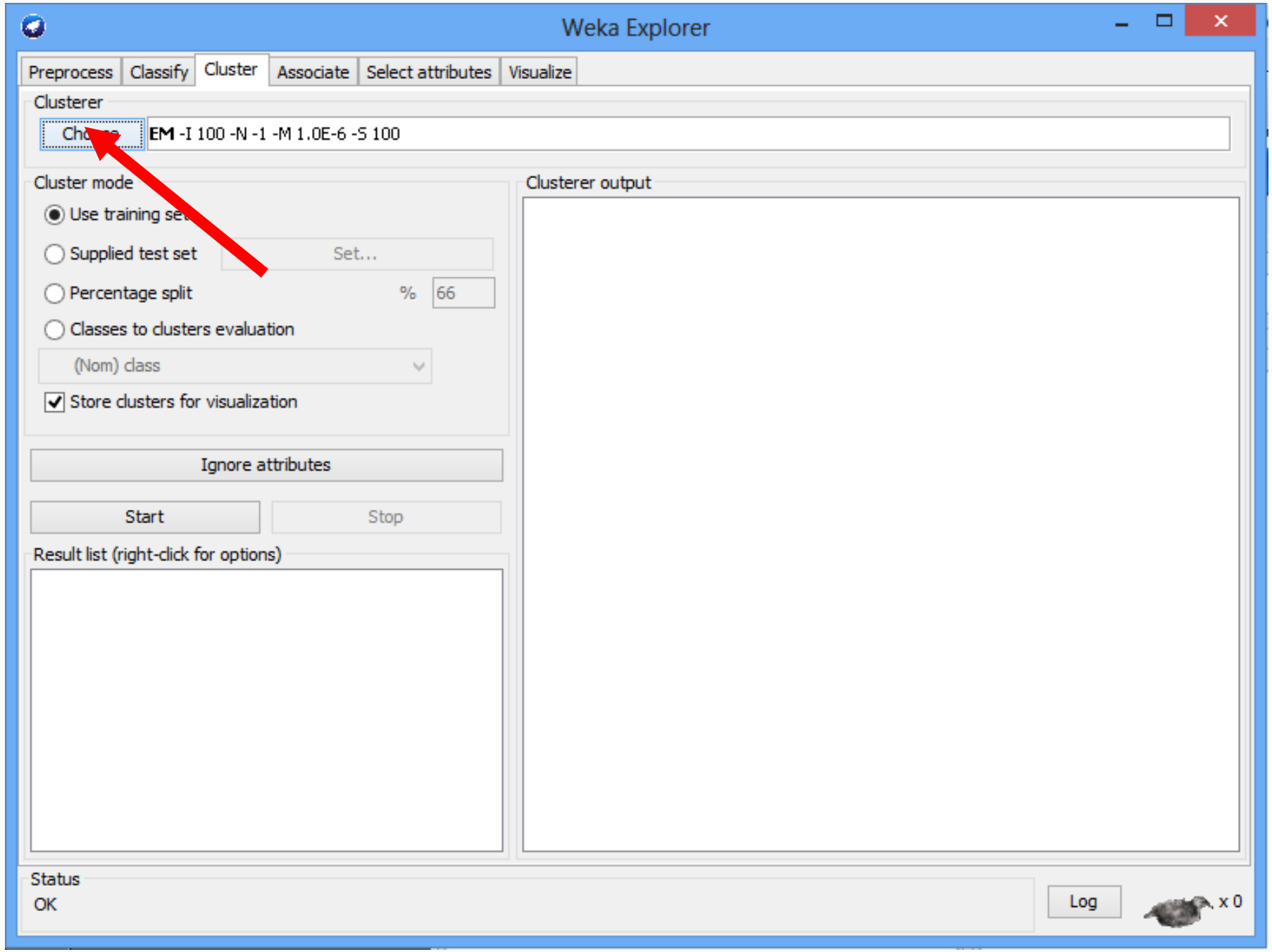
Abstract— **Generally, data mining (sometimes called data or knowledge discovery) is the process of analyzing data from different perspectives and summarizing it into useful information. Data mining software is one of a number of analytical tools for analyzing data. It allows users to analyze data from many different dimensions or angles, categorize it, and summarize the relationships identified. Weka is a data mining tools. It is contain the many machine leaning algorithms. It is provide the facility to classify our data through various algorithms. In this paper we are studying the various clustering algorithms. Cluster analysis or clustering is the task of assigning a set of objects into groups (called clusters) so that the objects in the same cluster are more similar (in some sense or another) to each other than to**

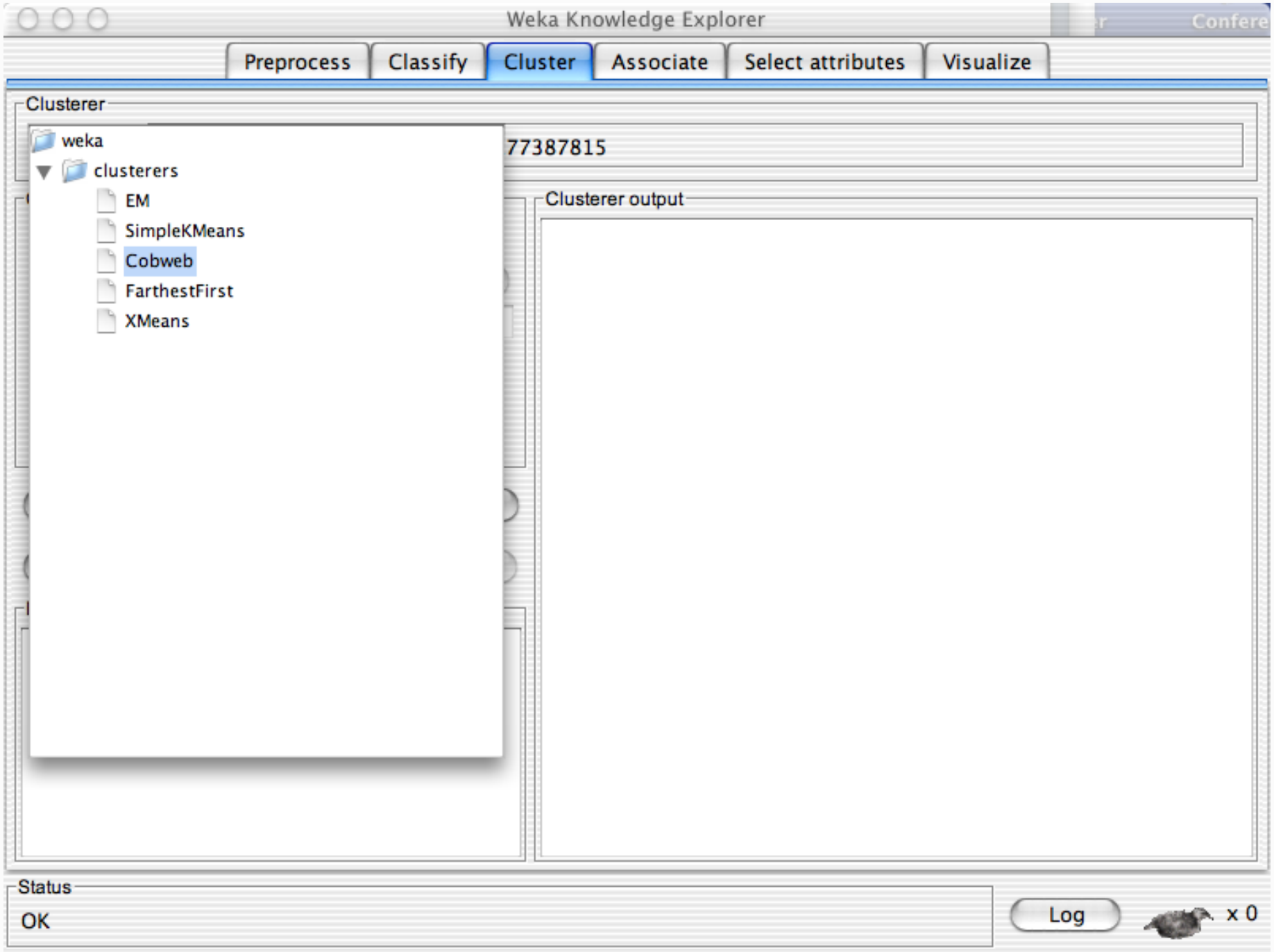
The main thing, why I am chooses WEKA, because we can work in weka easily without having the deep knowledge of data mining techniques.

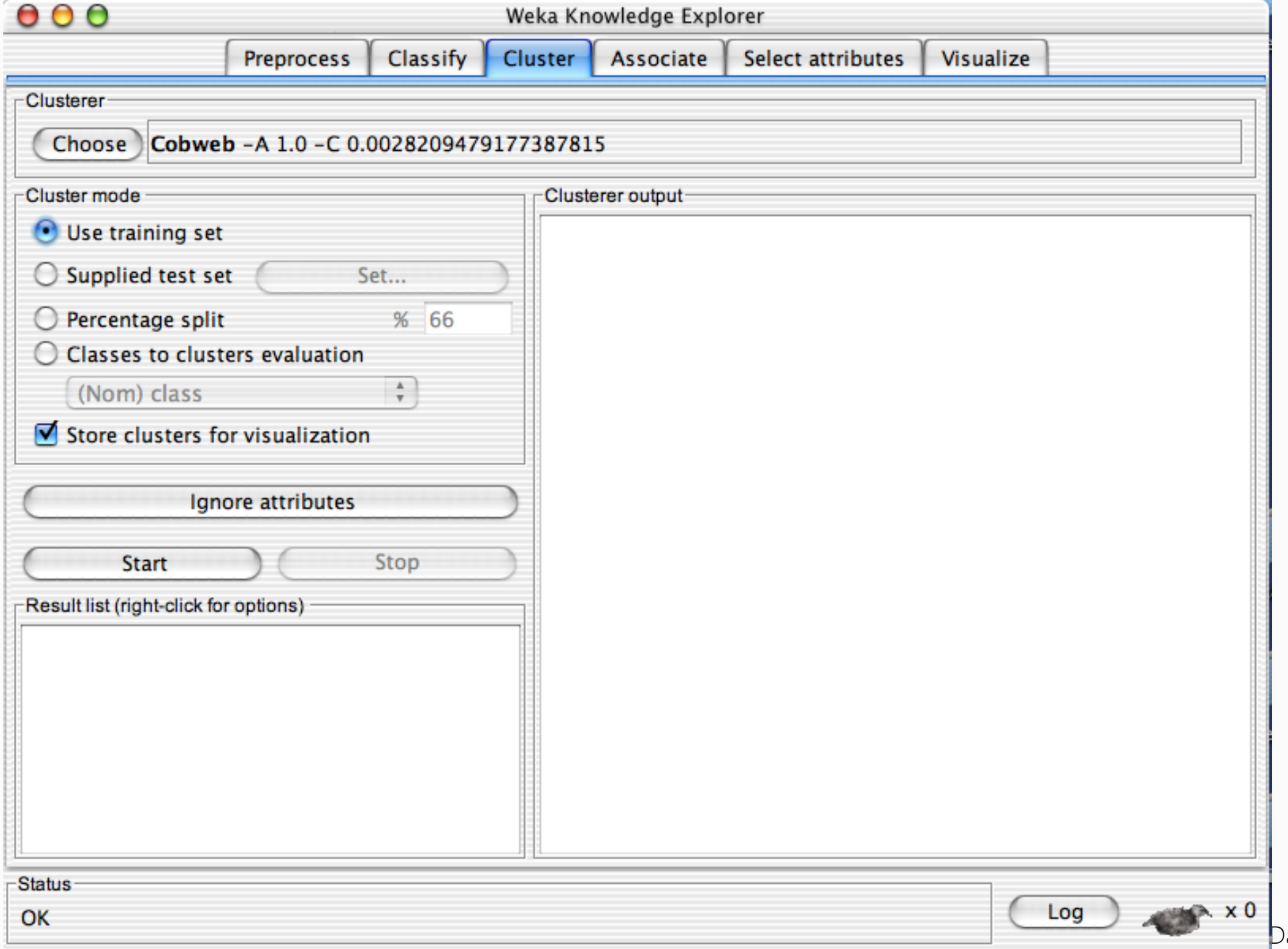
II. What Is Cluster Analysis?

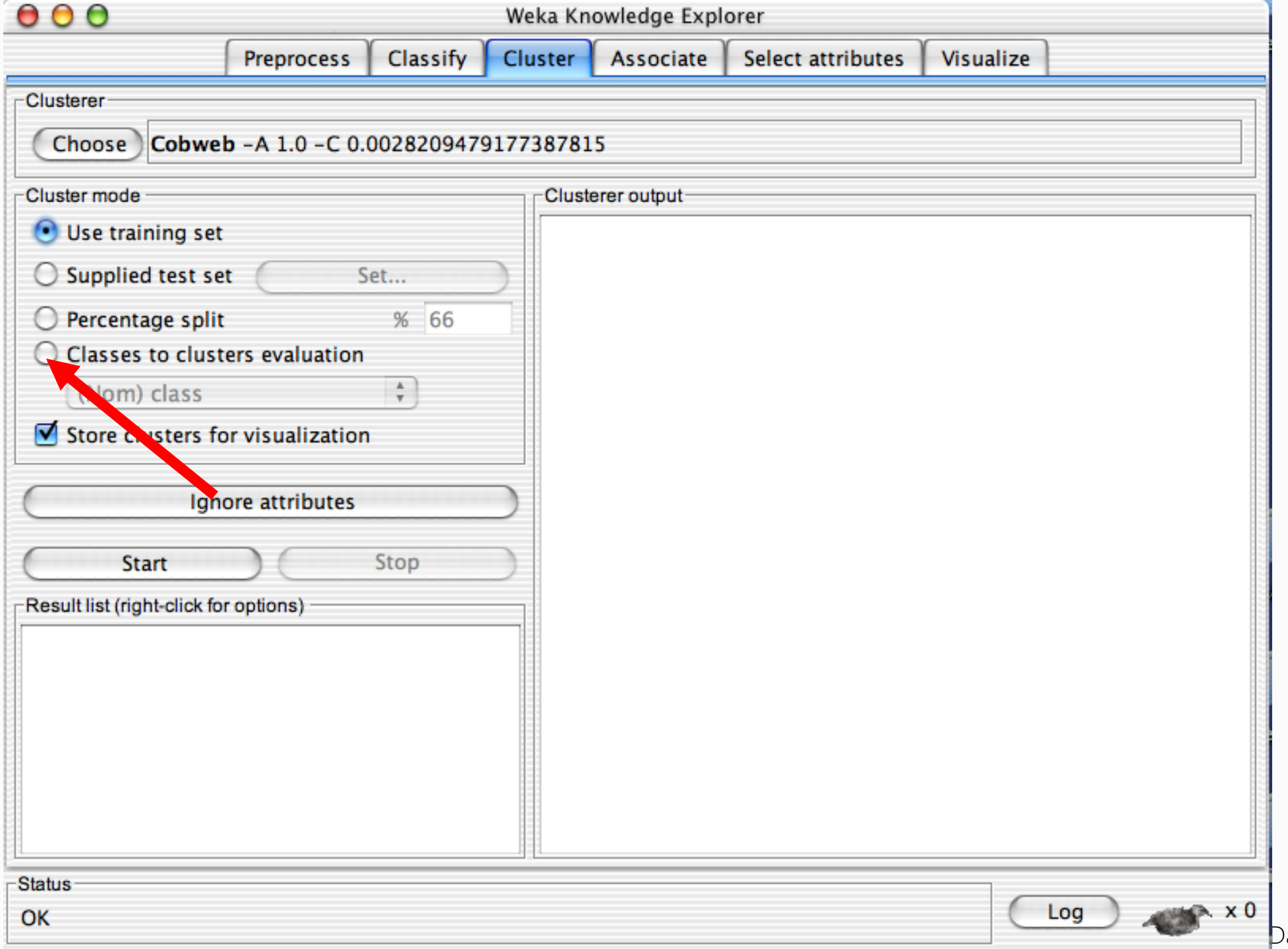
Cluster analysis[1] groups objects (observations, events) based on the information found in the data describing the objects or their relationships. The goal is that the objects in a group will be similar (or related) to one other and different from (or unrelated to) the objects in other groups. The greater the likeness (or homogeneity) within a group, and the greater the disparity between groups, the "better" or


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: iris
Instances: 150
Attributes: 5
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
Attributes
All
None
Invert
Pattern
No.
Name
1 sepallength
2 sepalwidth
3 petallength
4 petalwidth
5 class
Remove
Class: class (Nom)
Visualize All
34
30
28
25
16
10
7
4.3
6.1
7.9
Status
OK
Log
x 0


Weka Explorer
Preprocess
Classify
Cluster
Associate
Select attributes
Visualize
Clusterer
EM -I 100 -N -1 -M 1.0E-6 -S 100
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)
Clusterer output
Status
OK
Log
x 0


Weka Knowledge Explorer
Preprocess
Classify
Cluster
Associate
Select attributes
Visualize
Clusterer
weka
clusterers
EM
SimpleKMeans
Cobweb
FarthestFirst
XMeans
77387815
Clusterer output
Status
OK
Log
x 0


Weka Knowledge Explorer
Preprocess
Classify
Cluster
Associate
Select attributes
Visualize
Clusterer
Choose
Cobweb -A 1.0 -C 0.0028209479177387815
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)
Clusterer output
Status
OK
Log
x 0


Weka Knowledge Explorer
Preprocess
Classify
Cluster
Associate
Select attributes
Visualize
Clusterer
Choose
Cobweb -A 1.0 -C 0.0028209479177387815
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)
Clusterer output
Status
OK
Log
x 0

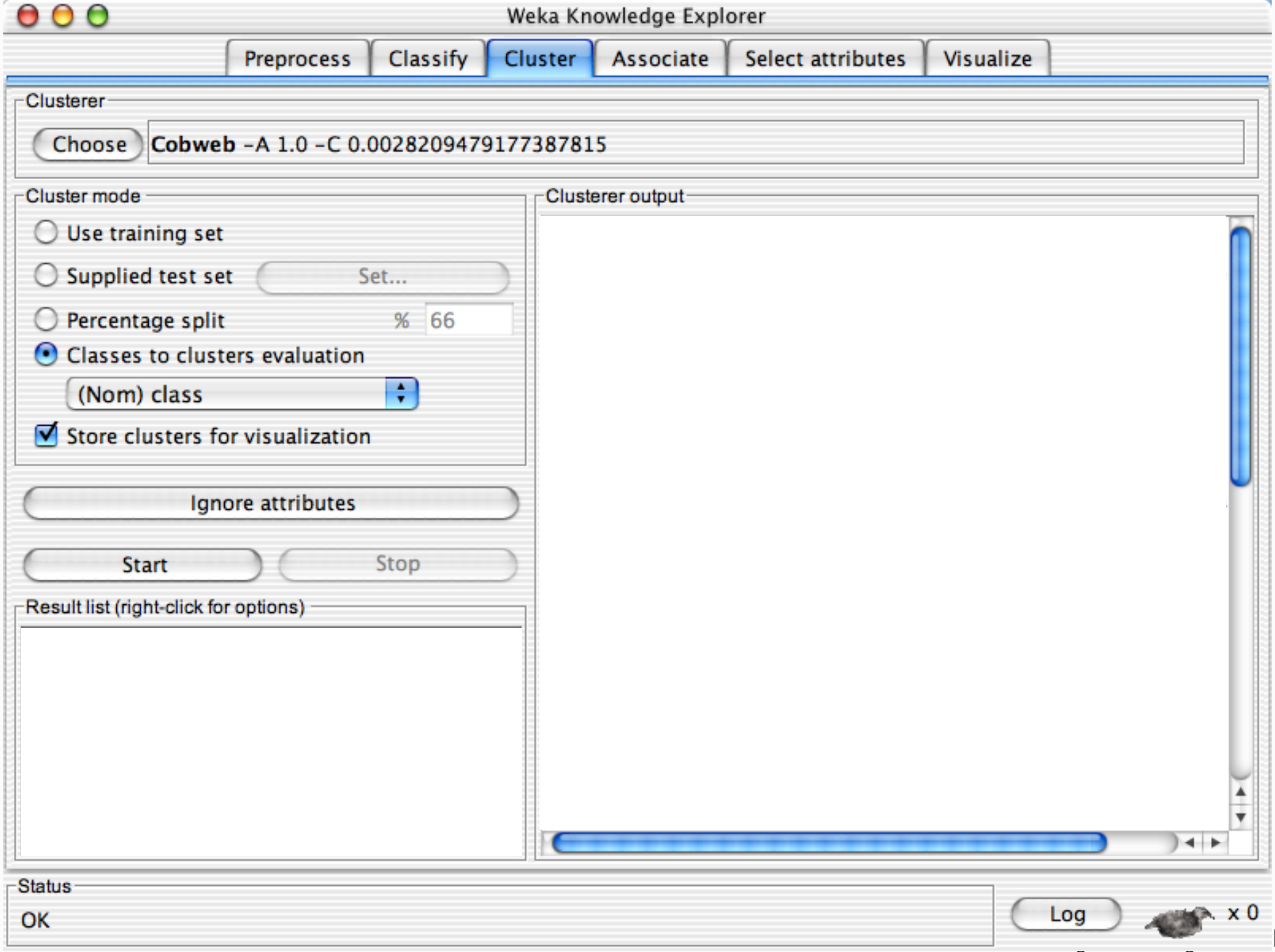
Evaluation in WEKA

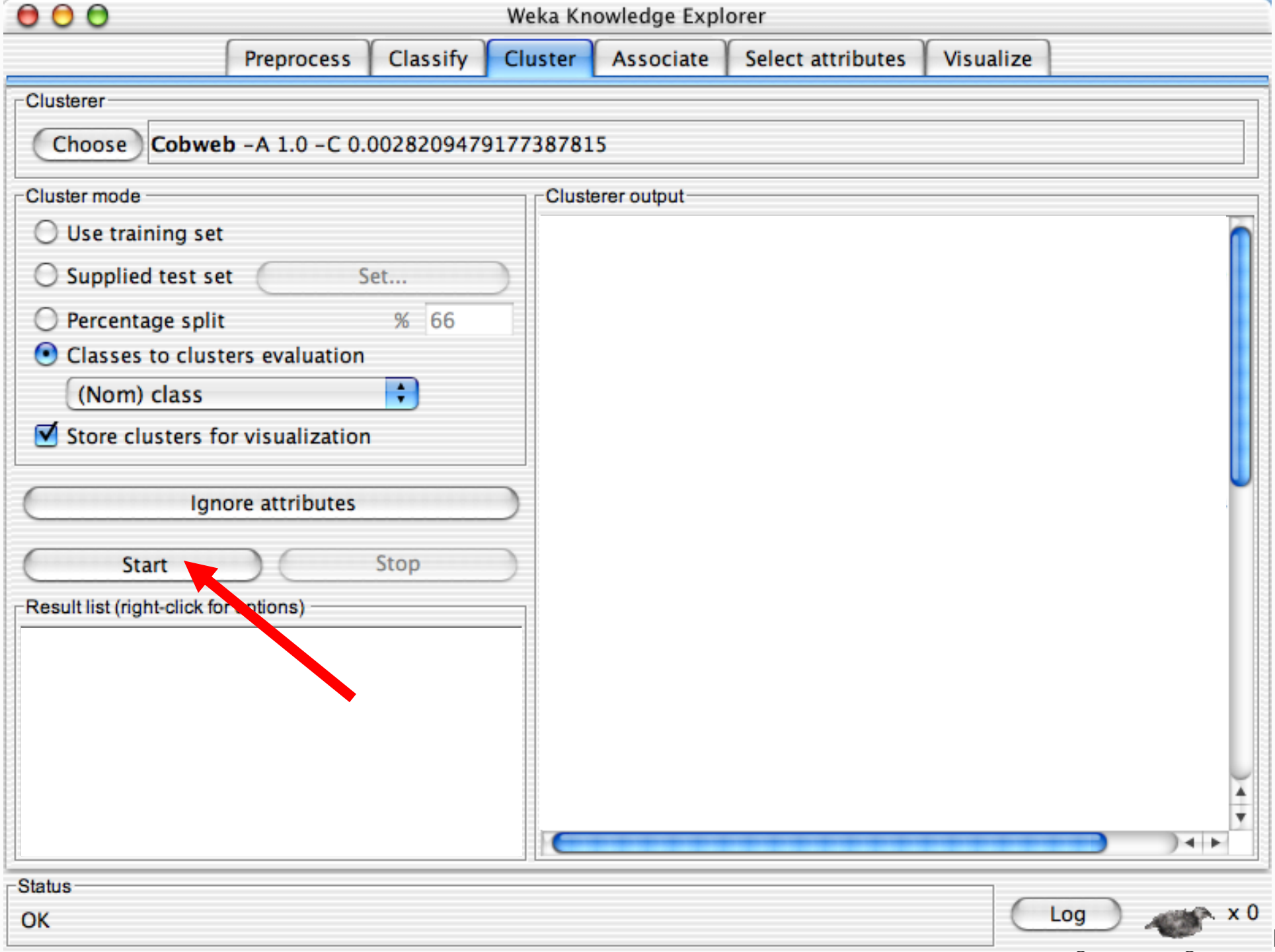
The way Weka **evaluates** clusterings depends on the cluster mode you select. Four different cluster modes are available (as buttons in the Cluster mode panel):

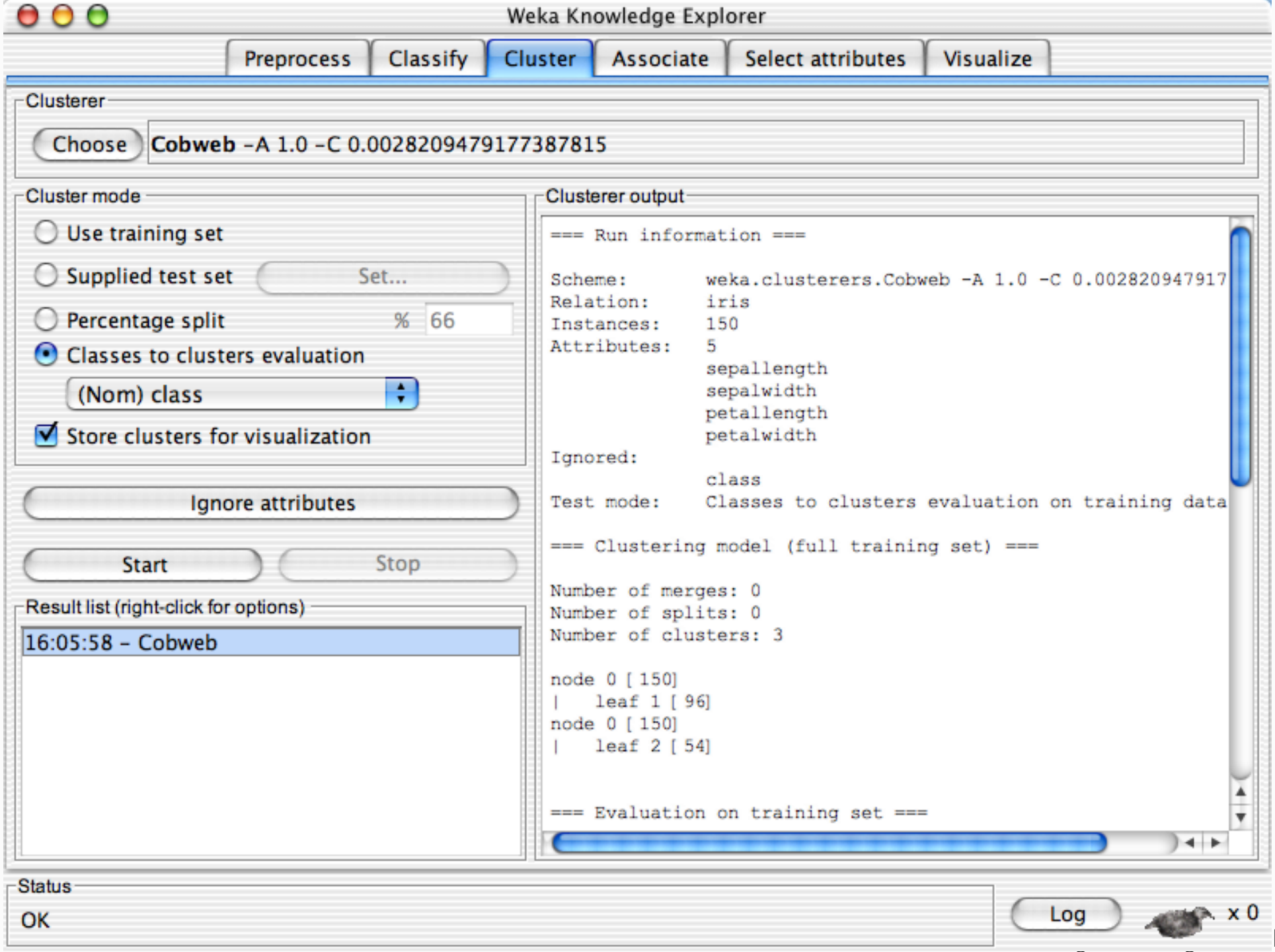
- **Use training set** (default). After generating the clustering Weka classifies the training instances into clusters according to the cluster representation and computes the percentage of instances falling in each cluster.

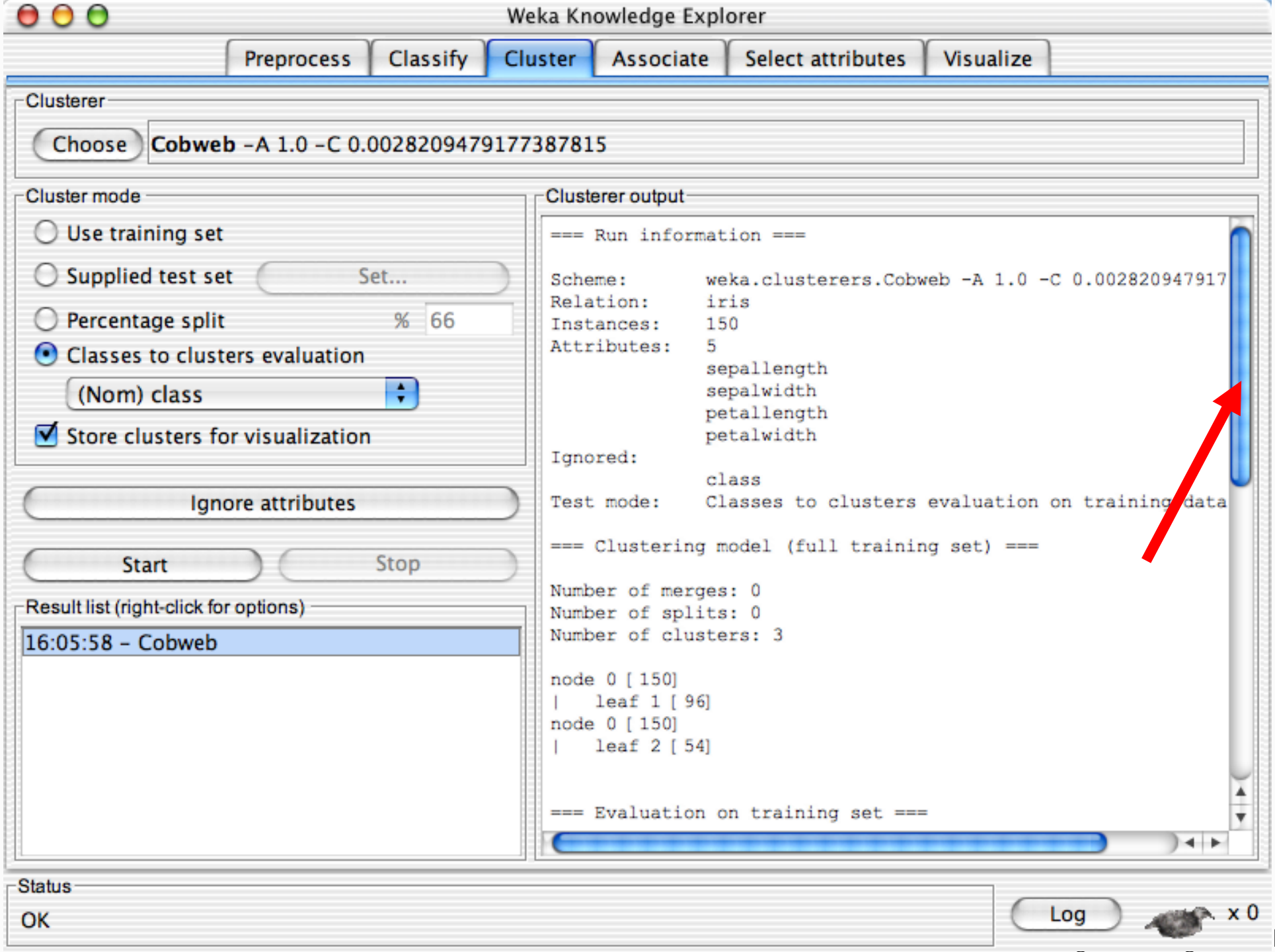
- In **Supplied test** set or **Percentage split** Weka can evaluate clusterings on separate test data.

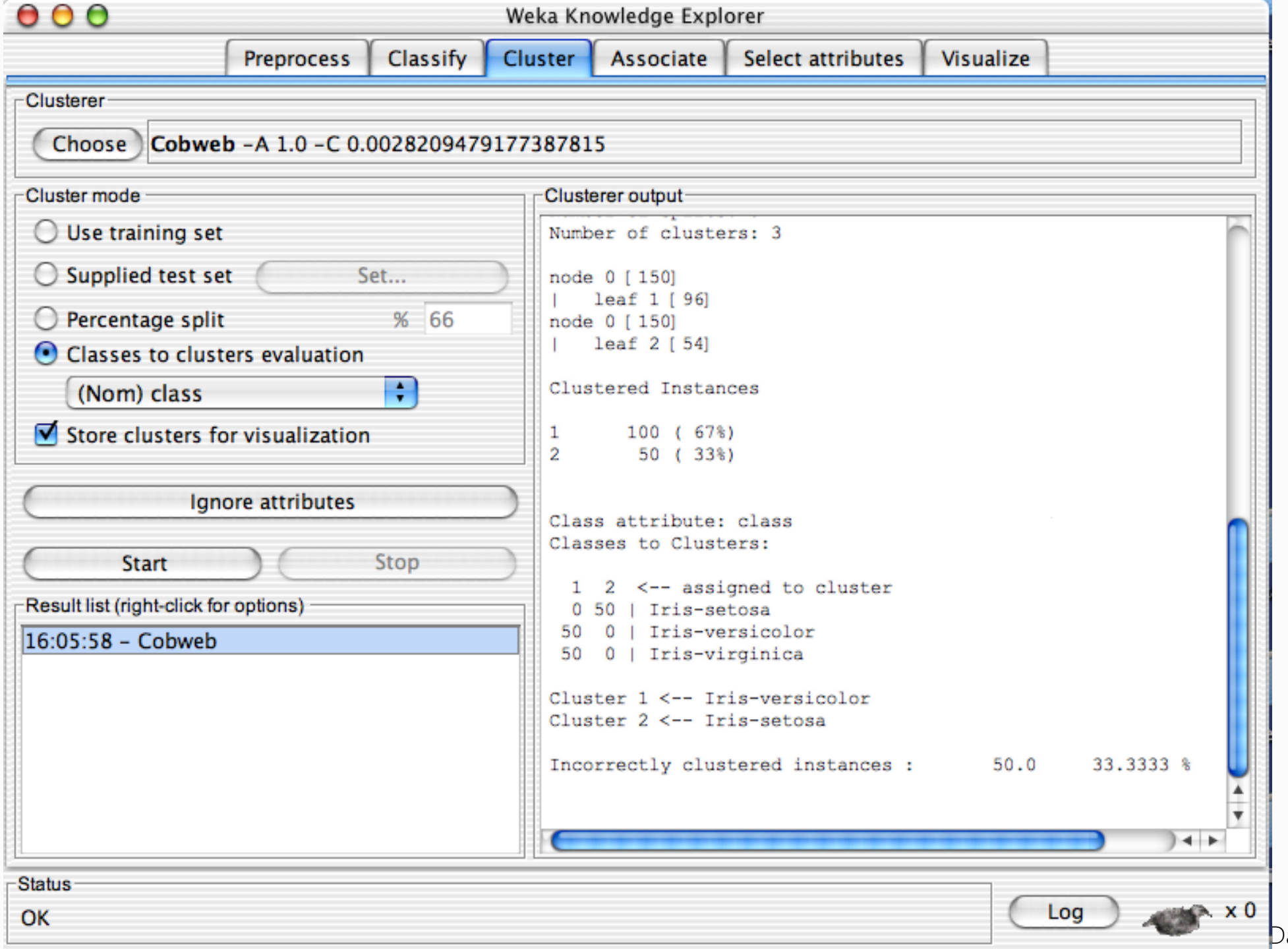
- **Classes to clusters** evaluation. In this mode Weka first ignores the class attribute and generates the clustering. Then during the test phase it assigns classes to the clusters, based on the majority value of the class attribute within each cluster. Then it computes the classification error, based on this assignment and also shows the corresponding confusion matrix.

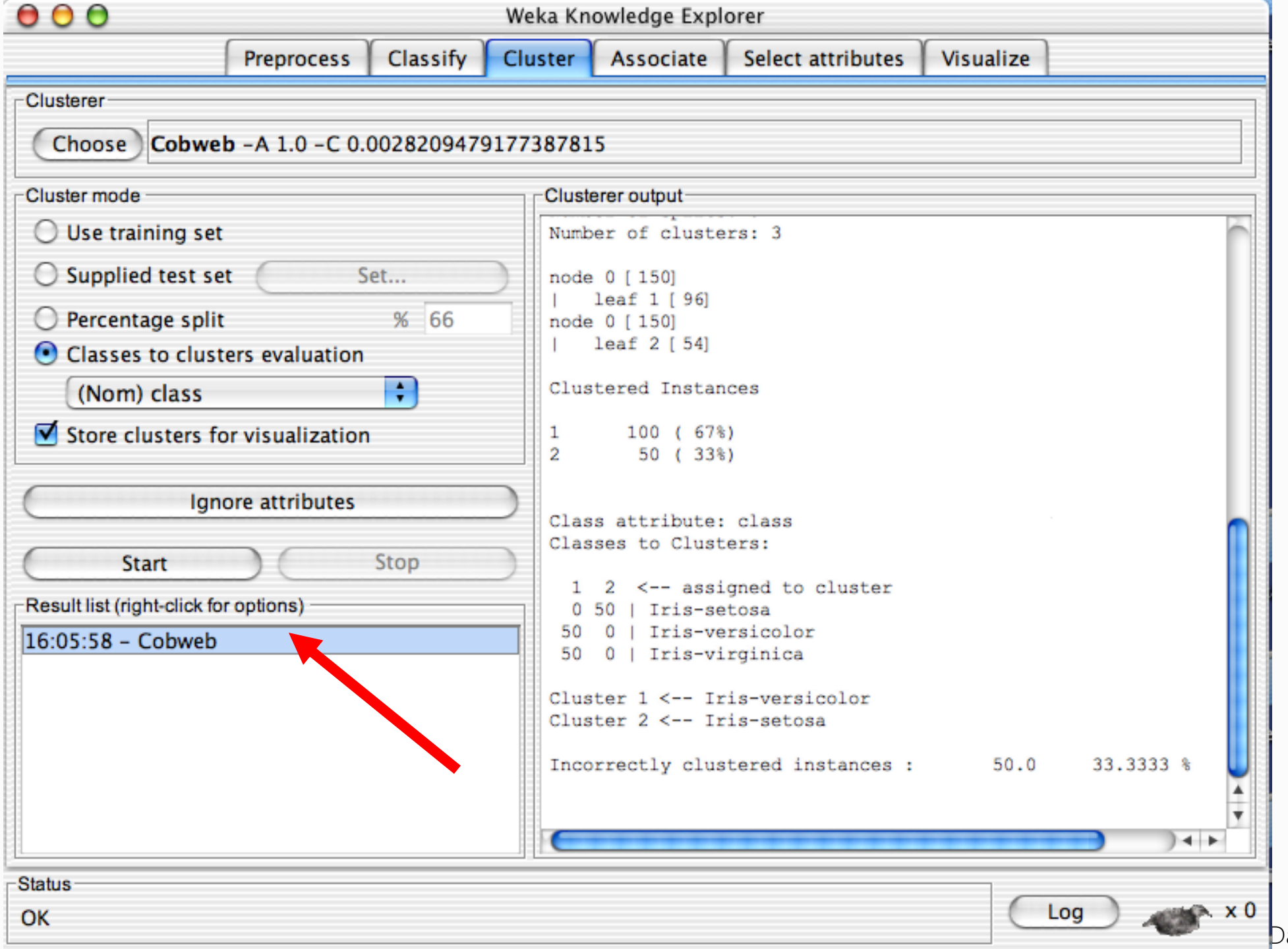

Weka Knowledge Explorer
Preprocess
Classify
Cluster
Associate
Select attributes
Visualize
Clusterer
Choose
Cobweb -A 1.0 -C 0.0028209479177387815
Cluster mode
Use training set
Supplied test set
Set...
Percentage split
%
66
Classes to clusters evaluation
(Nom) class
Store clusters for visualization
Ignore attributes
Start
Stop
Result list (right-click for options)
Clusterer output
Status
OK
Log
x 0

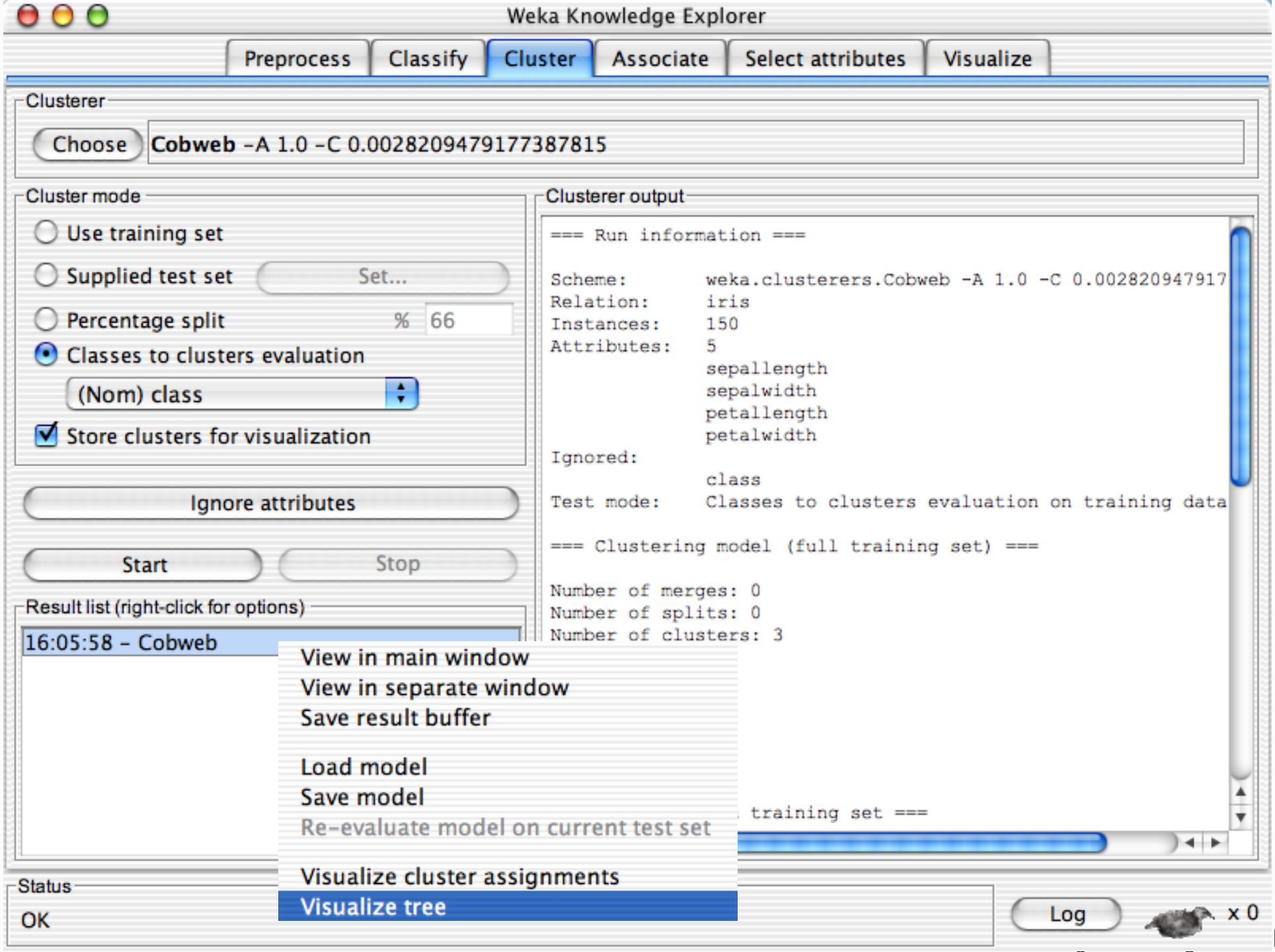

Weka Knowledge Explorer
Preprocess
Classify
Cluster
Associate
Select attributes
Visualize
Clusterer
Choose
Cobweb -A 1.0 -C 0.0028209479177387815
Cluster mode
Use training set
Supplied test set
Set...
Percentage split
%
66
Classes to clusters evaluation
(Nom) class
Store clusters for visualization
Ignore attributes
Start
Stop
Result list (right-click for options)
Clusterer output
Status
OK
Log
x 0

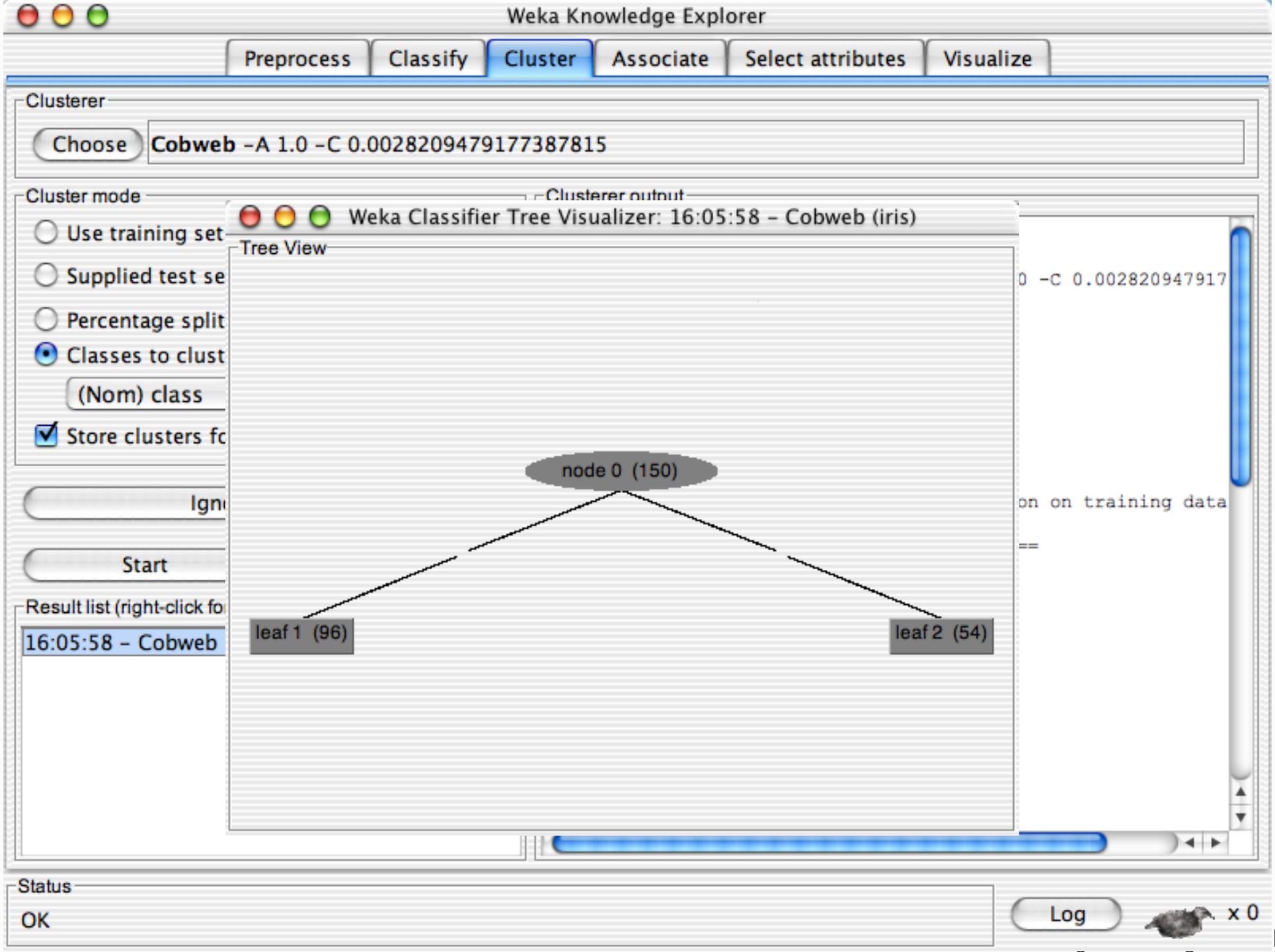

Weka Knowledge Explorer
Preprocess
Classify
Cluster
Associate
Select attributes
Visualize
Clusterer
Choose
Cobweb -A 1.0 -C 0.0028209479177387815
Cluster mode
Use training set
Supplied test set
Set...
Percentage split
% 66
Classes to clusters evaluation
(Nom) class
Store clusters for visualization
Ignore attributes
Start
Stop
Result list (right-click for options)
16:05:58 - Cobweb
Clusterer output
=== Run information ===
Scheme: weka.clusterers.Cobweb -A 1.0 -C 0.002820947917
Relation: iris
Instances: 150
Attributes: 5
sepallength
sepalwidth
petallength
petalwidth
Ignored:
class
Test mode: Classes to clusters evaluation on training data
=== Clustering model (full training set) ===
Number of merges: 0
Number of splits: 0
Number of clusters: 3
node 0 [150]
| leaf 1 [96]
node 0 [150]
| leaf 2 [54]
=== Evaluation on training set ===
Status
OK
Log
x 0

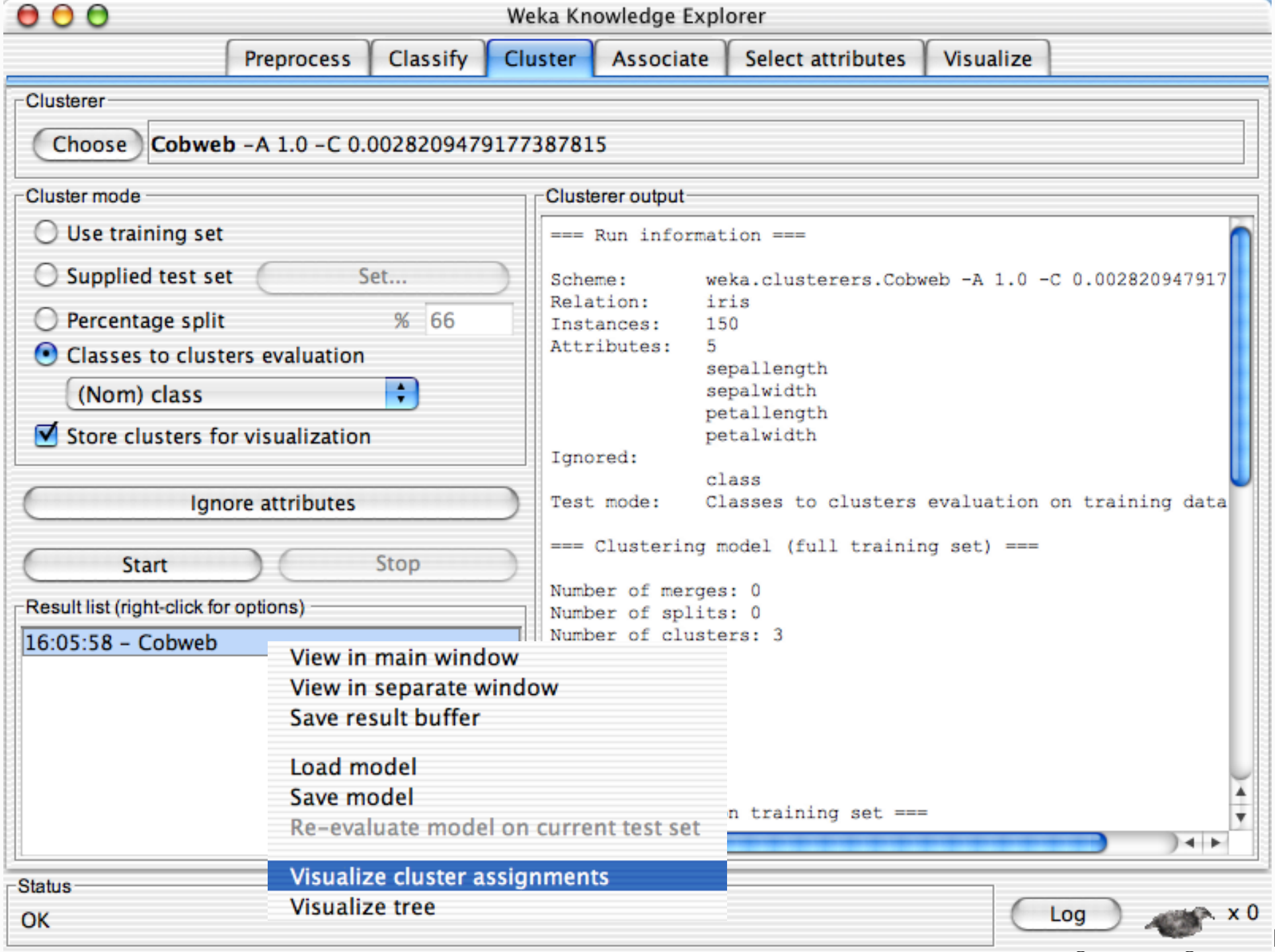

Weka Knowledge Explorer
Preprocess
Classify
Cluster
Associate
Select attributes
Visualize
Clusterer
Choose
Cobweb -A 1.0 -C 0.0028209479177387815
Cluster mode
Use training set
Supplied test set
Set...
Percentage split
% 66
Classes to clusters evaluation
(Nom) class
Store clusters for visualization
Ignore attributes
Start
Stop
Result list (right-click for options)
16:05:58 - Cobweb
Clusterer output
=== Run information ===
Scheme: weka.clusterers.Cobweb -A 1.0 -C 0.002820947917
Relation: iris
Instances: 150
Attributes: 5
sepallength
sepalwidth
petallength
petalwidth
Ignored:
class
Test mode: Classes to clusters evaluation on training data
=== Clustering model (full training set) ===
Number of merges: 0
Number of splits: 0
Number of clusters: 3
node 0 [150]
| leaf 1 [96]
node 0 [150]
| leaf 2 [54]
=== Evaluation on training set ===
Status
OK
Log
x 0

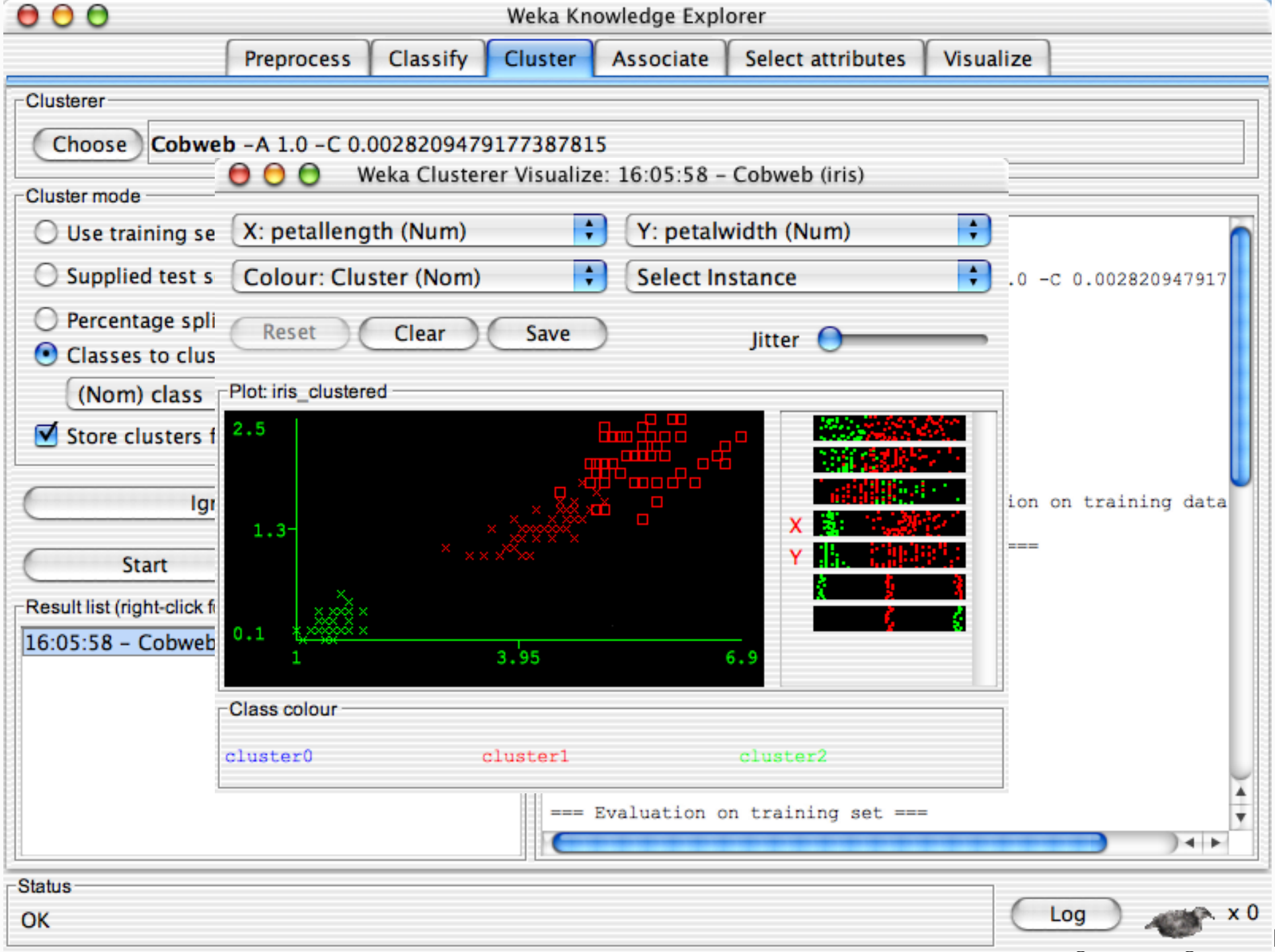

Weka Knowledge Explorer
Preprocess
Classify
Cluster
Associate
Select attributes
Visualize
Clusterer
Choose
Cobweb -A 1.0 -C 0.0028209479177387815
Cluster mode
Use training set
Supplied test set
Set...
Percentage split
% 66
Classes to clusters evaluation
(Nom) class
Store clusters for visualization
Ignore attributes
Start
Stop
Result list (right-click for options)
16:05:58 - Cobweb
Clusterer output
Number of clusters: 3
node 0 [150]
| leaf 1 [96]
node 0 [150]
| leaf 2 [54]
Clustered Instances
1 100 (67%)
2 50 (33%)
Class attribute: class
Classes to Clusters:
1 2 <-- assigned to cluster
0 50 | Iris-setosa
50 0 | Iris-versicolor
50 0 | Iris-virginica
Cluster 1 <-- Iris-versicolor
Cluster 2 <-- Iris-setosa
Incorrectly clustered instances : 50.0 33.3333 %
Status
OK
Log
x 0


Weka Knowledge Explorer
Preprocess
Classify
Cluster
Associate
Select attributes
Visualize
Clusterer
Choose
Cobweb -A 1.0 -C 0.0028209479177387815
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)
16:05:58 - Cobweb
Clusterer output
Number of clusters: 3
node 0 [150]
| leaf 1 [96]
node 0 [150]
| leaf 2 [54]
Clustered Instances
1 100 (67%)
2 50 (33%)
Class attribute: class
Classes to Clusters:
1 2 <-- assigned to cluster
0 50 | Iris-setosa
50 0 | Iris-versicolor
50 0 | Iris-virginica
Cluster 1 <-- Iris-versicolor
Cluster 2 <-- Iris-setosa
Incorrectly clustered instances : 50.0 33.3333 %
Status
OK
Log
x 0


Weka Knowledge Explorer
Preprocess
Classify
Cluster
Associate
Select attributes
Visualize
Clusterer
Choose
Cobweb -A 1.0 -C 0.0028209479177387815
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)
16:05:58 - Cobweb
Clusterer output
=== Run information ===
Scheme: weka.clusterers.Cobweb -A 1.0 -C 0.002820947917
Relation: iris
Instances: 150
Attributes: 5
sepallength
sepalwidth
petallength
petalwidth
Ignored:
class
Test mode: Classes to clusters evaluation on training data
=== Clustering model (full training set) ===
Number of merges: 0
Number of splits: 0
Number of clusters: 3
training set ===
View in main window
View in separate window
Save result buffer
Load model
Save model
Re-evaluate model on current test set
Visualize cluster assignments
Visualize tree
Status
OK
Log
x 0

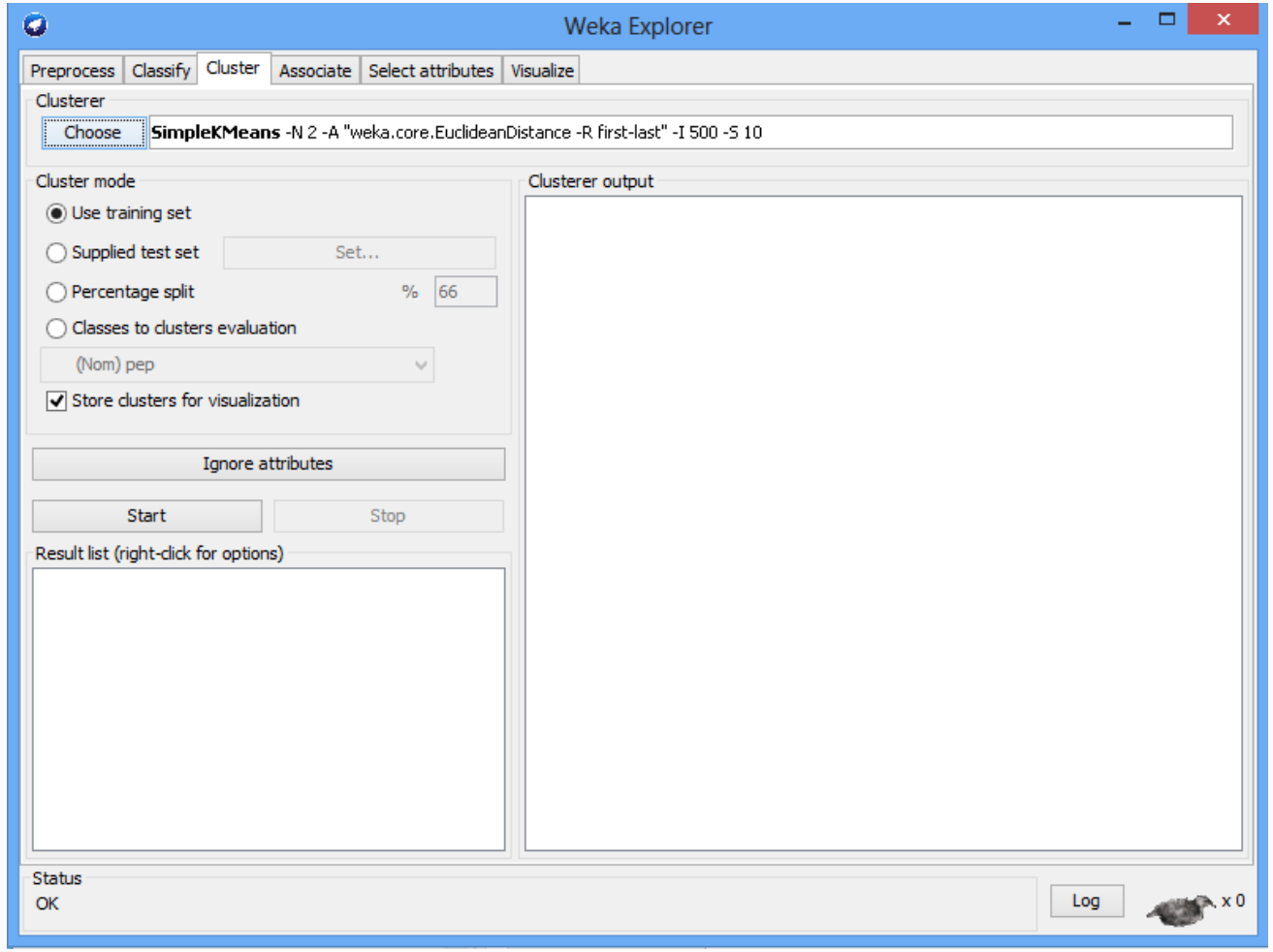

Weka Knowledge Explorer
Preprocess
Classify
Cluster
Associate
Select attributes
Visualize
Clusterer
Choose
Cobweb -A 1.0 -C 0.0028209479177387815
Cluster mode
Use training set
Supplied test se
Percentage split
Classes to clust
(Nom) class
Store clusters fo
Ign
Start
Result list (right-click fo
16:05:58 - Cobweb
Clusterer output
Weka Classifier Tree Visualizer: 16:05:58 - Cobweb (iris)
Tree View
node 0 (150)
leaf 1 (96)
leaf 2 (54)
0 -C 0.002820947917
on on training data
==
Status
OK
Log
x 0


Weka Knowledge Explorer
Preprocess
Classify
Cluster
Associate
Select attributes
Visualize
Clusterer
Choose
Cobweb -A 1.0 -C 0.0028209479177387815
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)
16:05:58 - Cobweb
Clusterer output
=== Run information ===
Scheme: weka.clusterers.Cobweb -A 1.0 -C 0.002820947917
Relation: iris
Instances: 150
Attributes: 5
sepallength
sepalwidth
petallength
petalwidth
Ignored:
class
Test mode: Classes to clusters evaluation on training data
=== Clustering model (full training set) ===
Number of merges: 0
Number of splits: 0
Number of clusters: 3
n training set ===
View in main window
View in separate window
Save result buffer
Load model
Save model
Re-evaluate model on current test set
Visualize cluster assignments
Visualize tree
Status
OK
Log
x 0

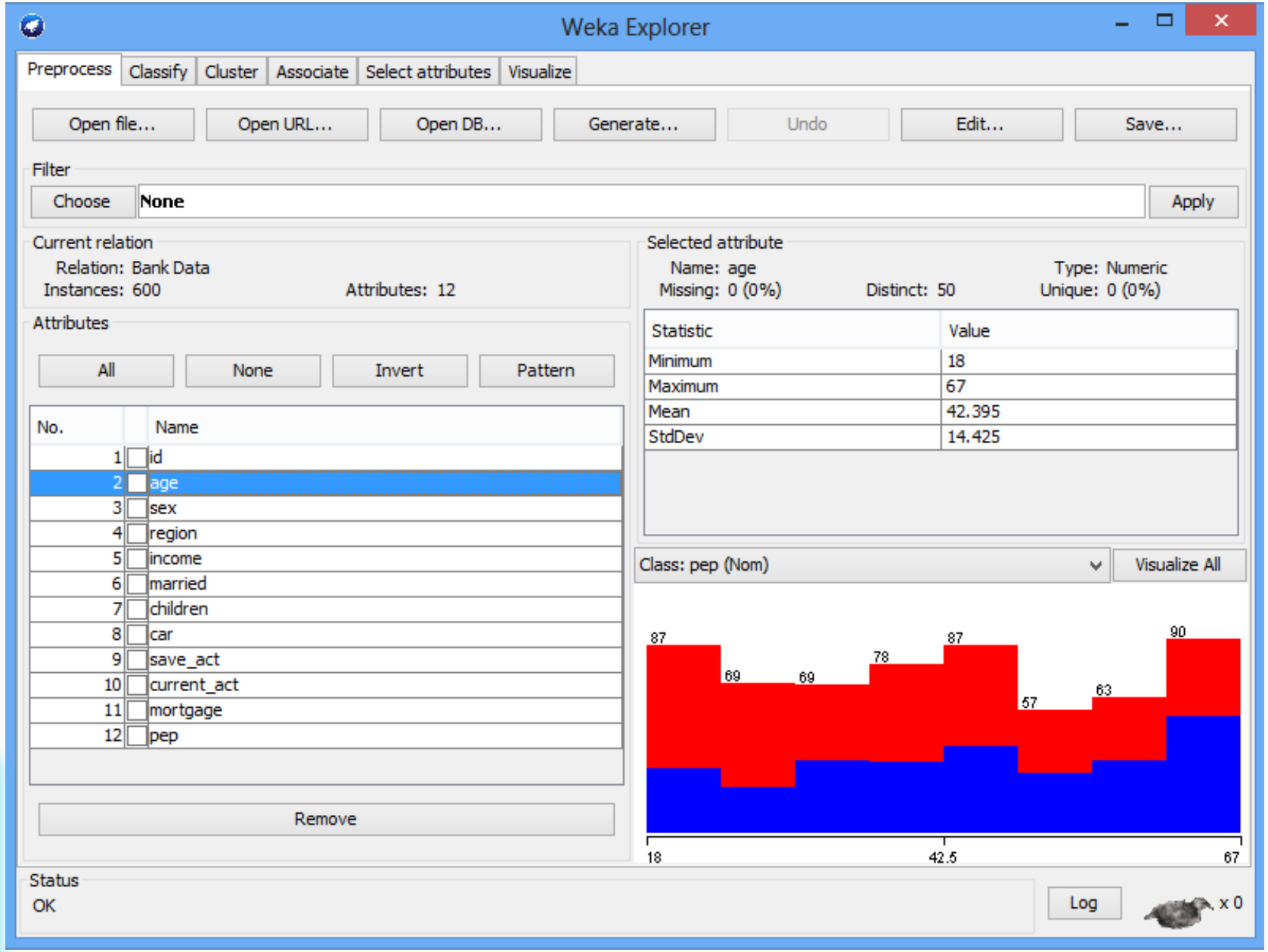

Weka Knowledge Explorer
Preprocess
Classify
Cluster
Associate
Select attributes
Visualize
Clusterer
Choose
Cobweb -A 1.0 -C 0.0028209479177387815
Weka Clusterer Visualize: 16:05:58 - Cobweb (iris)
Cluster mode
Use training se
Supplied test s
Percentage spli
Classes to clus
(Nom) class
Store clusters f
X: petallength (Num)
Y: petalwidth (Num)
Colour: Cluster (Nom)
Select Instance
Reset
Clear
Save
Jitter
Plot: iris_clustered
2.5
1.3
0.1
1
3.95
6.9
X
Y
Start
Result list (right-click f
16:05:58 - Cobweb
Class colour
cluster0
cluster1
cluster2
.0 -C 0.002820947917
ion on training data
===
=== Evaluation on training set ===
Status
OK
Log
x 0

K-Means in WEKA

- WEKA automatically handles mixture of categorical and numerical variables;
- Algorithm automatically normalizes numerical attributes, replaces missing;
- Uses Euclidean distance to compute distance between instance and cluster;



K-Means in WEKA
Bank Data

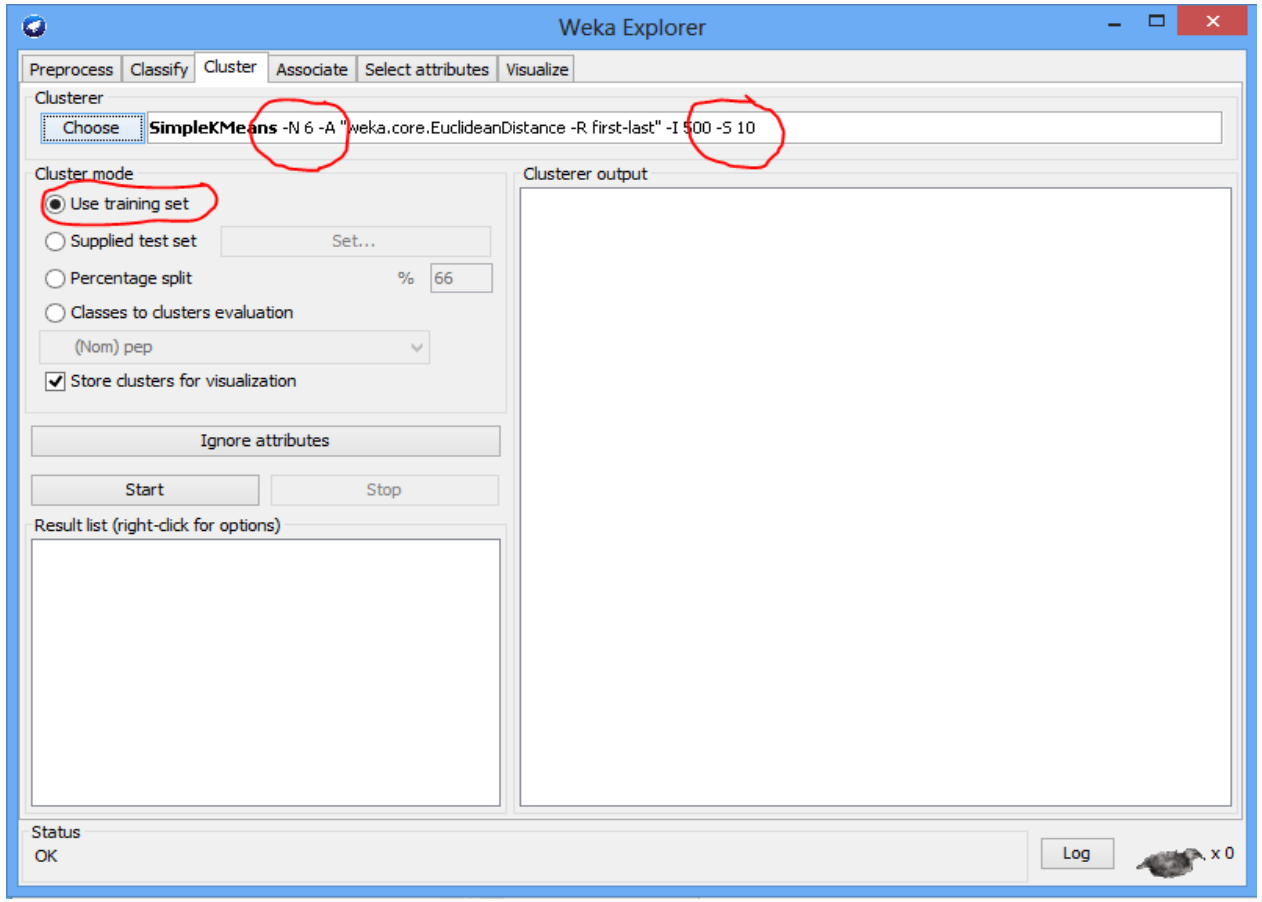


Note, some implementations of K-means only allow numerical values so it may be necessary to convert categorical to binary.

Also, normalize attributes on very differently scales (age and income).

K-Means in WEKA

- Select ‘cluster’ tab, click on ‘choose’ and select ‘SimpleKMeans’ from dropdown.
- Enter 6 as the number of clusters, leave seed ‘as is’;
- Note Kmeans can be sensitive to initial placement of clusters, play with seed;
- Select ‘Use training set’ option, click start…



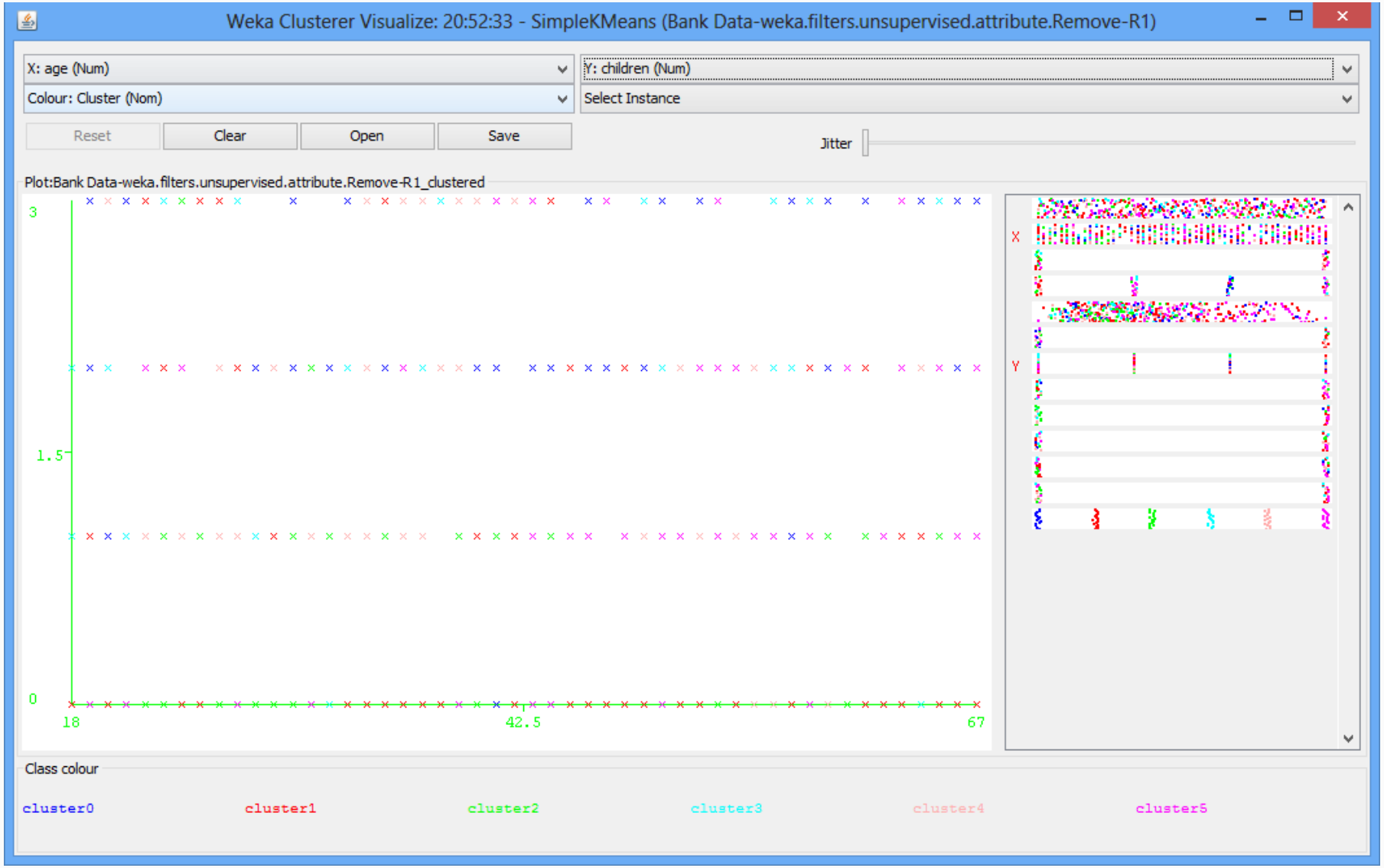
K-Means in WEKA, let's analyze the results…

- Right click the result set in the "Results List" panel

```
Number of iterations: 18
Within cluster sum of squared errors: 1955.4146634784236
Missing values globally replaced with mean/mode

Cluster centroids:
                         Cluster#
Attribute      Full Data         0          1          2          3          4          5
                   (600)      (74)      (164)      (71)       (58)       (99)      (134)
=============================================================================================
id               ID12101    ID12107    ID12103    ID12101    ID12104    ID12102    ID12108
age                42.395    42.9324    43.7744    39.0282    37.3103     38.404    47.3433
sex                FEMALE     FEMALE     FEMALE     FEMALE     FEMALE       MALE       MALE
region         INNER_CITY      RURAL INNER_CITY INNER_CITY      TOWN INNER_CITY      TOWN
income        27524.0312 28838.7605 28586.4063 20463.1273 20600.8528  25720.037 33568.3929
married              YES         NO        YES        YES        YES        YES         NO
children          1.0117      1.973      0.628     0.6901     1.6207      0.899     0.9403
car                  NO         NO         NO         NO         NO        YES        YES
save_act            YES        YES        YES         NO         NO         NO        YES
current_act         YES        YES        YES        YES        YES        YES        YES
mortgage             NO         NO         NO         NO         NO        YES         NO
pep                  NO         NO         NO        YES         NO        YES        YES



Time taken to build model (full training data) : 0.14 seconds
```

K-Means in WEKA

- Right click ‘visualize cluster assignments’ in the “Results List” panel



Clustering

Some final takeaways from this model: The power of clustering and Nearest Neighbor becomes obvious when we talk about data sets like Netflix and Amazon. Amazon with their ~100 million users and Netflix with their 4 Billion streamed moves, their algorithms are very accurate since there are likely many potential customers in their databases with similar buying/viewing habits to you. Thus, the nearest neighbor to yourself is likely very similar. This creates an accurate and effective model.

Contrarily, the model breaks down quickly and becomes inaccurate when you have few data points for comparison. In the early stages of an online e-commerce store for example, when there are only 50 customers, a product recommendation feature will likely not be accurate at all, as the nearest neighbor may in fact be very distant from yourself.

Lecture 5 Homework 5b

Homework Lecture 5, mandatory

An online shopping site has the following primary pages or sections: **Home, Products, Search, Prod_A, Prod_B, Prod_C, Cart, Purchase**. A user may browse from "Home" to "Products" and then to one of the individual products. The user may also search for a specific product by using the "Search" function. A visit to "Cart" implies that the user has placed an item in the shopping cart, and "Purchase" indicates the user completed the purchase of items in the shopping cart. The site has collect the hypothetical session data for 100 sessions. This data is available in **CSV** format, Sessions.CSV, on course website.

Use WEKA's K-means clustering algorithm to cluster these user sessions into segments. Try different clustering runs with various numbers of clusters (e.g., between 4 and 8), and select the result set(s) that seem to best answer as many of the following questions as possible.

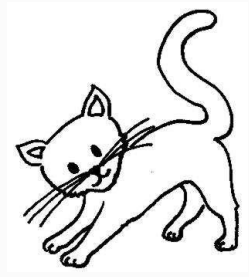
- If a new user is observed to access the following pages: **Home => Search => Prod_B**, according to your clusters, what other product should be recommended to this user? Explain your answer based on your clustering results. What if the new user has accessed the following sequence instead: **Products => Prod_C**?
- Can clustering help us identify casual browsers ("window shoppers"), focused browsers (those who seem to know what products they are looking for), and searchers (those using the search function to find items they want)? If so, are any of these groups show a higher or lower propensity to make a purchase?
- Do any of the segments show particular interest in one or more products, and if so, can we identify any special characteristics about their navigational behavior or their purchase propensity?
- If we know that, during the time of data collection, independent banner ads had been placed on some popular sites pointing to products A and B, can we identify segments corresponding to visitors that respond to the ads? If so, can we determine if either of these promotional campaigns are having any success?

Distributions Matter…

Sometimes two wrongs do make a right…

Distributions Matter…

Because the Internet is all about cute kittens



Resulting in highly skewed distribution in training set…

The Class Imbalance Problem I

- Data sets are said to be balanced if there are, approximately, as many positive examples of the concept as there are negative ones.
- Many domains that do not have a balanced data set.

Examples:

- Helicopter Gearbox Fault Monitoring
- Discrimination between Earthquakes and Nuclear Explosions
- Document Filtering
- Detection of Oil Spills
- Detection of Fraudulent Telephone Calls

The Class Imbalance Problem II

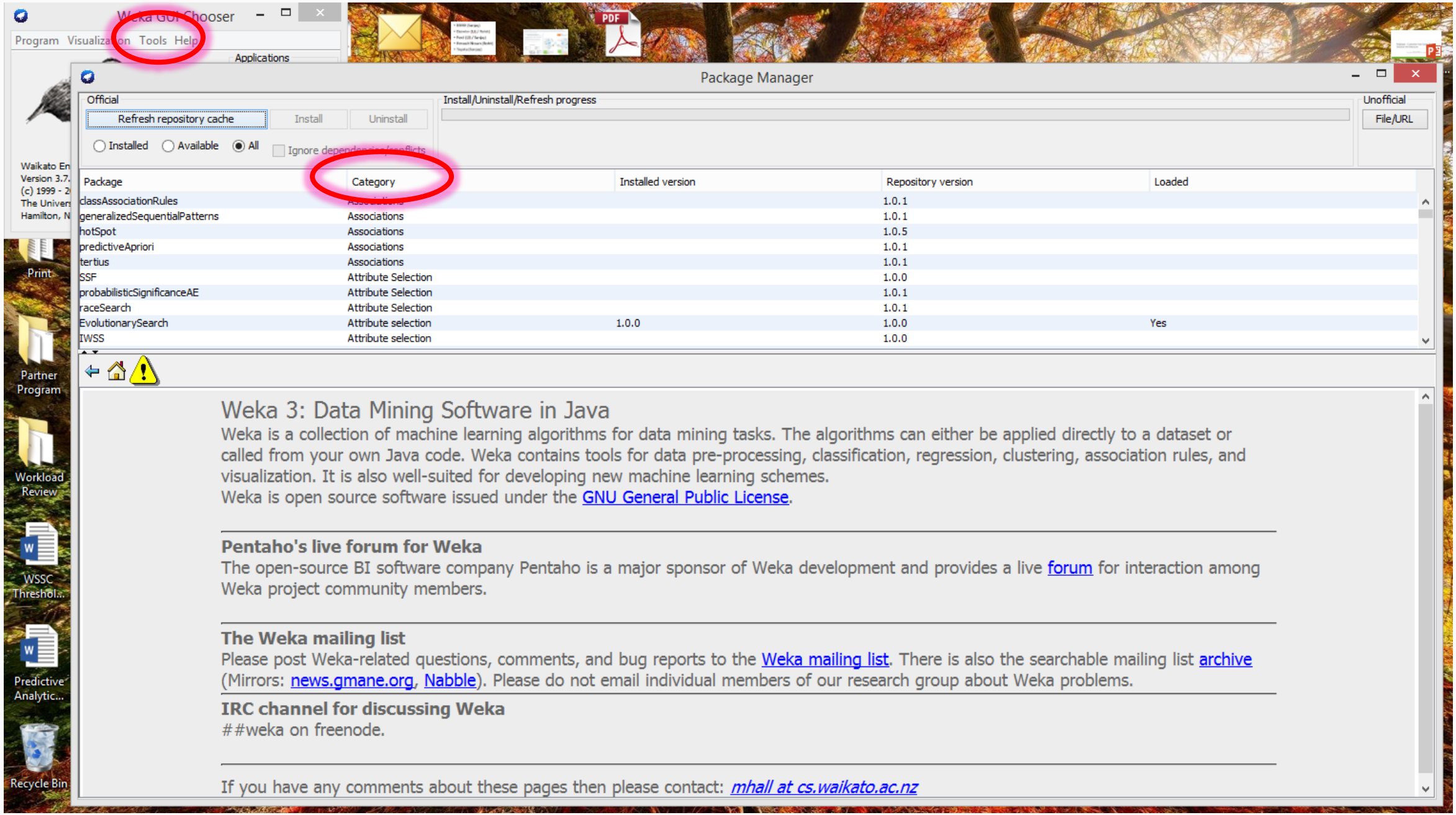
- The problem with class imbalances is that standard learners are often biased towards the majority class.
- That is because these classifiers attempt to reduce global quantities such as the error rate, not taking the data distribution into consideration.
- As a result examples from the overwhelming class are well-classified whereas examples from the minority class tend to be misclassified.

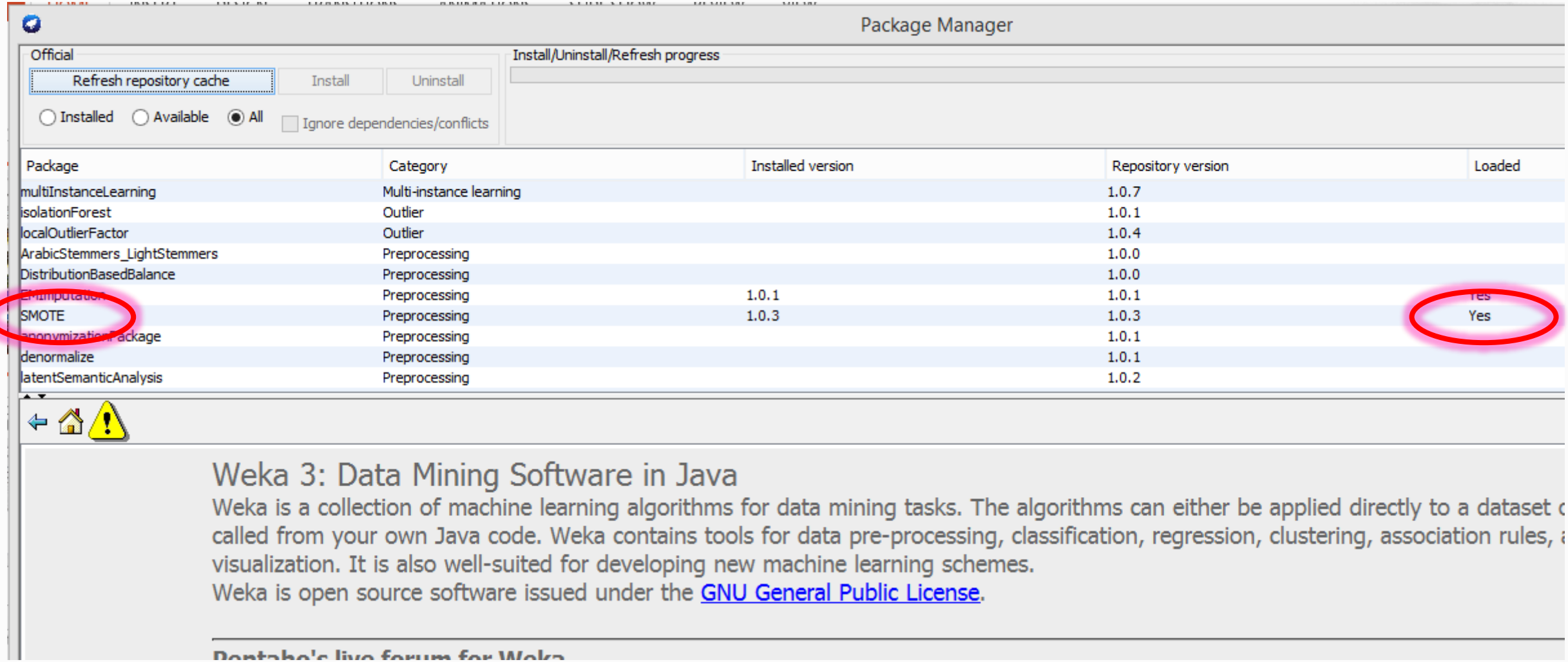
Some Generalities

- Evaluating performance of a model on a class imbalance problem is not done appropriately with standard accuracy/error rate.
  - ROC Analysis is typically used, instead.

- There are three main ways to deal with class imbalances: re-sampling, re-weighing, and one-class learning (cover in SVMs)

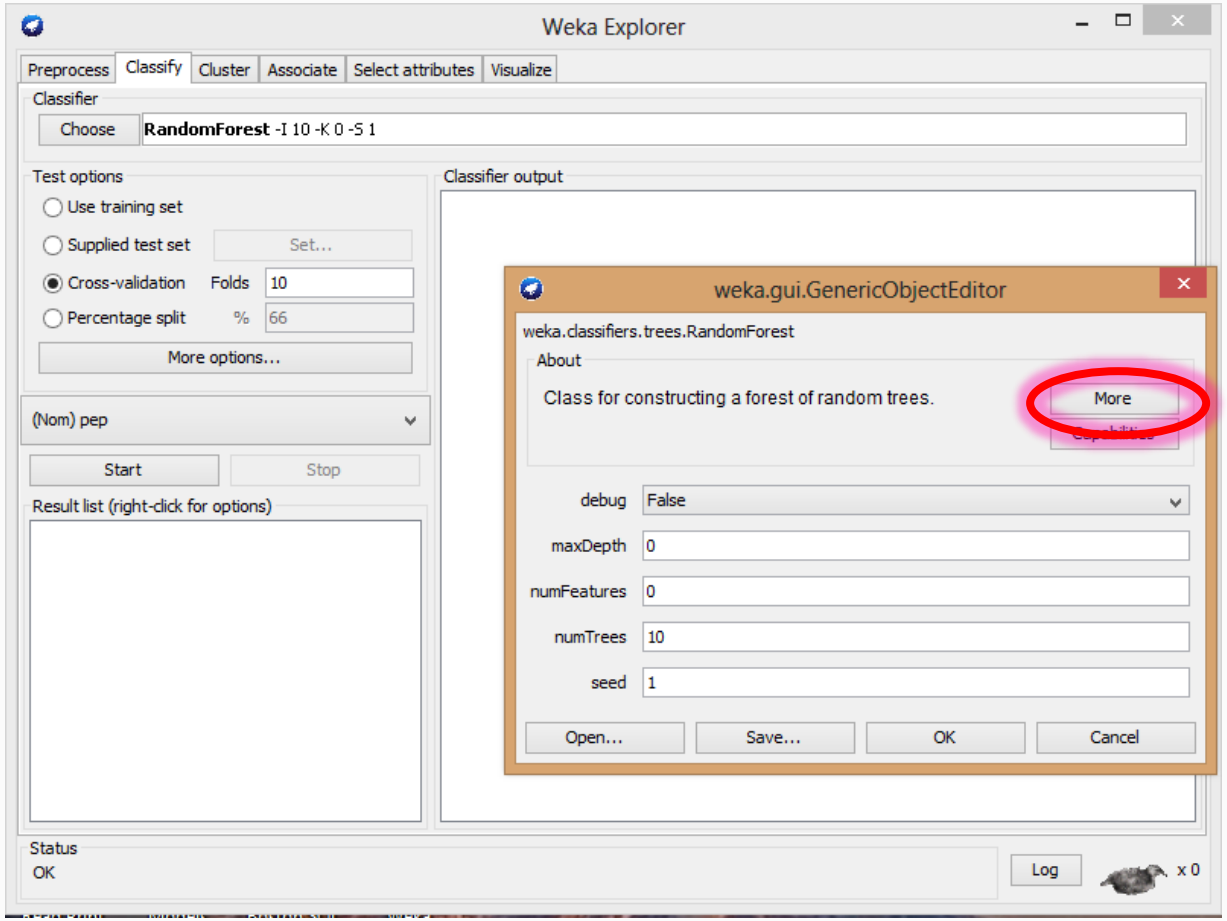
- Re-sampling provides a simple way of biasing generalization process.
- It can do so by:
  - Generating synthetic samples accordingly biased
  - Controlling the amount and placement of the new samples


Weka GUI Chooser
Program Visualization Tools Help
Applications
Package Manager
Official
Refresh repository cache
Install
Uninstall
Installed
Available
All
Ignore dependencies/conflicts
Install/Uninstall/Refresh progress
Unofficial
File/URL
Package
Category
Installed version
Repository version
Loaded
classAssociationRules
generalizedSequentialPatterns
hotSpot
predictiveApriori
tertius
SSF
probabilisticSignificanceAE
raceSearch
EvolutionarySearch
IWSS
Associations
Attribute Selection
Attribute selection
1.0.0
Yes
Weka 3: Data Mining Software in Java
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.
Weka is open source software issued under the GNU General Public License.
Pentaho's live forum for Weka
The open-source BI software company Pentaho is a major sponsor of Weka development and provides a live forum for interaction among Weka project community members.
The Weka mailing list
Please post Weka-related questions, comments, and bug reports to the Weka mailing list. There is also the searchable mailing list archive (Mirrors: news.gmane.org, Nabble). Please do not email individual members of our research group about Weka problems.
IRC channel for discussing Weka
##weka on freenode.
If you have any comments about these pages then please contact: mhall at cs.waikato.ac.nz
Print
Partner Program
Workload Review
WSSC Threshol...
Predictive Analytic...
Recycle Bin


Package Manager
Official
Refresh repository cache
Install
Uninstall
Installed
Available
All
Ignore dependencies/conflicts
Install/Uninstall/Refresh progress
Package
Category
Installed version
Repository version
Loaded
multiInstanceLearning
Multi-instance learning
1.0.7
isolationForest
Outlier
1.0.1
localOutlierFactor
Outlier
1.0.4
ArabicStemmers_LightStemmers
Preprocessing
1.0.0
DistributionBasedBalance
Preprocessing
1.0.0
Preprocessing
1.0.1
1.0.1
SMOTE
Preprocessing
1.0.3
1.0.3
Yes
Preprocessing
1.0.1
denormalize
Preprocessing
1.0.1
latentSemanticAnalysis
Preprocessing
1.0.2
Weka 3: Data Mining Software in Java
Weka is a collection of machine learning algorithms for data mining tasks. The algorithms can either be applied directly to a dataset
called from your own Java code. Weka contains tools for data pre-processing, classification, regression, clustering, association rules,
visualization. It is also well-suited for developing new machine learning schemes.
Weka is open source software issued under the GNU General Public License.

Improve Your Modelling Skills

Explore a bit, both in WEKA and online (references in WEKA notes)



Machine Learning, 54, 255–273, 2004
© 2004 Kluwer Academic Publishers. Manufactured in The Netherlands.

Is Combining Classifiers with Stacking Better than Selecting the Best One?

SAŠO DŽEROSKI saso.dzeroski@ijs.si

How to Make Stacking Better and Faster While Also Taking Care of an Unknown Weakness

Alexander K. Seewald ALEX@SEEWALD.AT; ALEXSEE@AI.UNIVIE.AC.AT
Austrian Research Institute for Artificial Intelligence, Schottengasse 3, A-1010 Wien, Austria

Abstract

We investigated performance differences between multi-class and two-class datasets for ensemble learning schemes. We were surprised to find that Stacking, the best-known such scheme, performs worse on multi-class datasets. In this paper we will present results concerning this heretofore unknown weakness of Stacking. In addition we will present a new variant of Stacking which is able to compensate for this weakness, improving Stacking significantly on some multi-class datasets. The dimensionality of the meta-data set is reduced by a factor equal to the number of classes, which leads to faster learning. In comparison to other ensemble learning methods this improves Stacking's lead further, making it the most successful system by a variety of measures.

1. Introduction

When faced with the decision "Which algorithm will be most accurate on my classification problem?", the method or meta-classification scheme is the family of *stacking* algorithms (Wolpert, 1992). The basic idea of Stacking is to use the predictions of the original classifiers as attributes in a new training set that keeps the original class labels.

A straightforward extension of this approach is using class probability distributions of the original classifiers[1] which convey not only prediction information, but also confidence for all classes. This approach was evaluated and found to be superior to Stacking with predictions in (Ting & Witten, 1999), provided *multi-response linear regression* (MLR) is used as level-1 classifier. Our variant StackingC is based on this extension.

We have investigated the performance of four common meta-classification schemes, including the mentioned Stacking variant, relative to multi-class vs. two-class datasets. We found no significant differences for Grading and Voting[2] – however, the mentioned Stacking variant showed a significant performance degradation for multi-class datasets. We will present these results and a variant of stacking which does not show this performance degradation.

First, we will present the basic concept behind Stack-

SMOTE: A State-of-the-Art Resampling Approach

- SMOTE stands for Synthetic Minority Oversampling Technique.
  - Technique designed by Chawla, Hall, & Kegelmeyer in 2002.
- It combines Informed oversampling of the minority class with random undersampling of the majority class.
- SMOTE currently yields the best results as far as re-sampling and modifying the probabilistic estimate techniques go (Chawla, 2003).

SMOTE's Informed Oversampling Procedure II

For each minority Sample

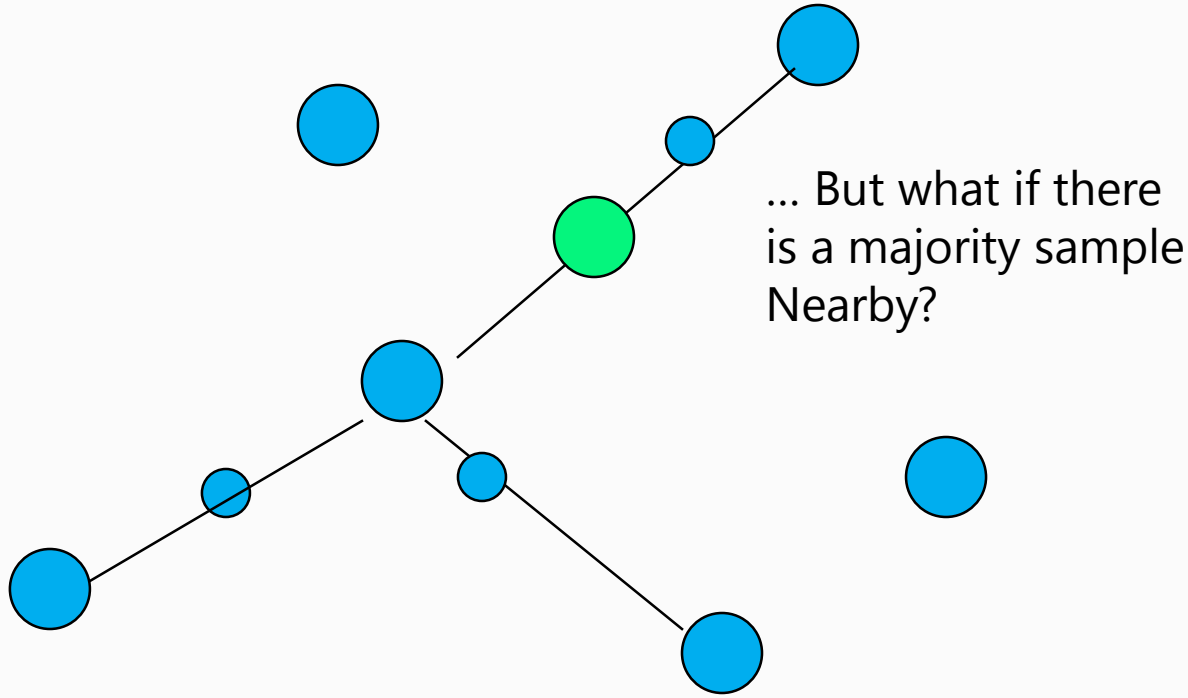
- Find its k-nearest minority neighbors
- Randomly select j of these neighbors
- Randomly generate synthetic samples along the lines joining the minority sample and its j selected neighbors

  (j depends on the amount of oversampling desired)

SMOTE's Informed vs. Random Oversampling

- Random Oversampling (with replacement) of the minority class has the effect of making the decision region for the minority class very specific.
- In a decision tree, it would cause a new split and often leads to overfitting.
- SMOTE's informed oversampling generalizes the decision region for the minority class.
- As a result, larger and less specific regions are learned, thus, paying attention to minority class samples without causing overfitting.

SMOTE's Informed Oversampling Procedure I



… But what if there is a majority sample Nearby?

: Minority sample

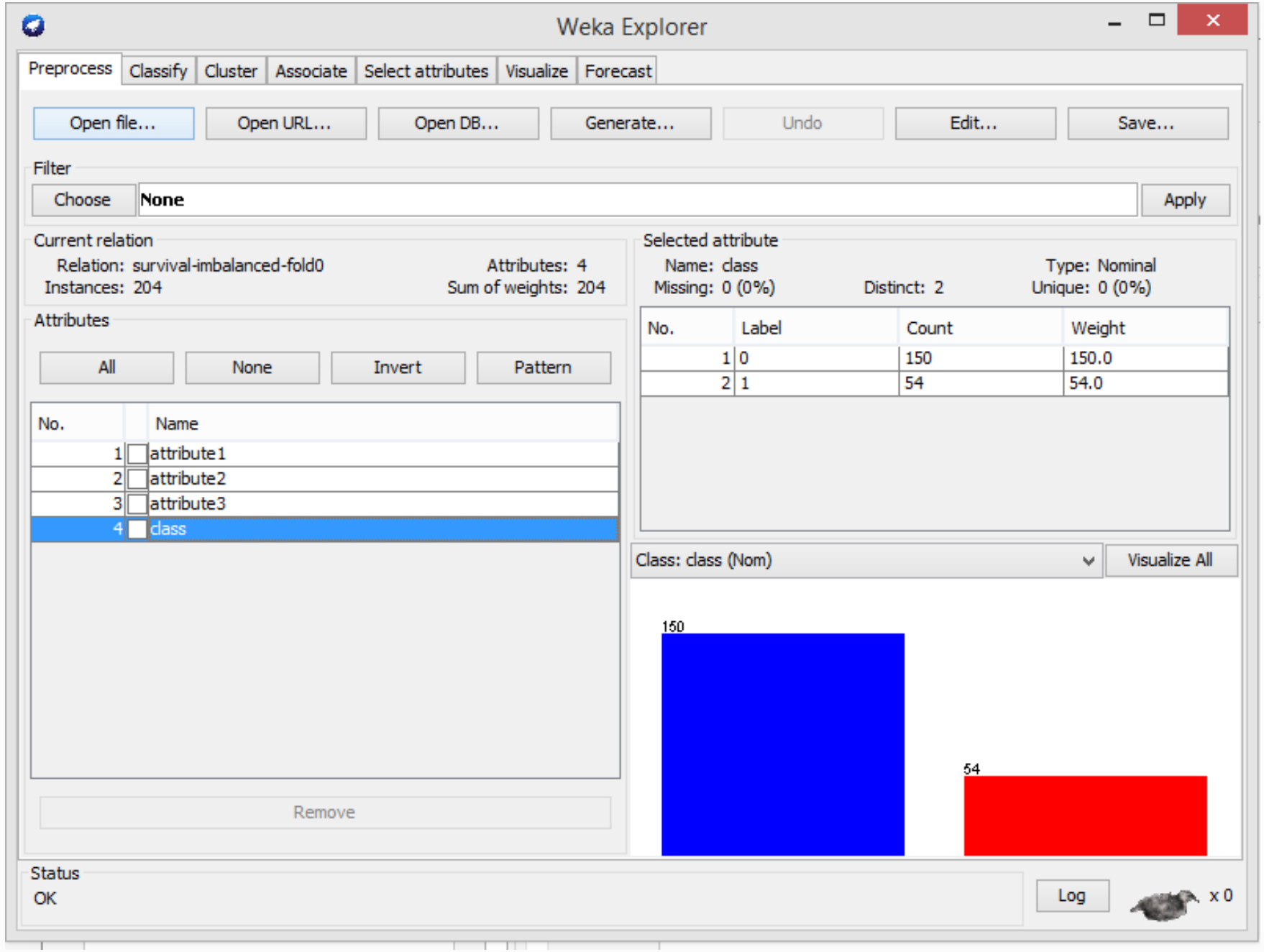
: Synthetic sample

: Majority sample

SMOTE's Shortcomings

- Overgeneralization
  - SMOTE's procedure can be dangerous since it blindly generalizes the minority area without regard to the majority class.
  - This strategy is problematic in the case of highly skewed class distributions since, in such cases, the minority class is very sparse with respect to the majority class, thus resulting in a greater chance of class mixture.

- Lack of Flexibility
  - The number of synthetic samples generated by SMOTE is fixed in advance, thus not allowing for any flexibility in the re-balancing rate.



Let's try it: *open the file example_imbalanced.arff...*

If your model is 98% accurate, so what?

Or right 1 time in 35

Cross Validation

- In cross validation, you break your training data up into K equally-sized partitions. You train a learning algorithm on K-1 of them and test it on the remaining 1. You do this K times, each time holding out a different partition as the "development" part. You can then average your performance over K runs.
- Suppose that you've presented a machine learning solution to your boss that achieves 7% error on cross validation. Your nemesis, Gabe, gives a solution to your boss that achieves 6.9% error on cross validation. How impressed should your boss be?
- It depends. If this 0.1% improvement was measured over 1000 examples, perhaps not too impressed. It would mean that Gabe got exactly *one more example right than you did*. (In fact, he probably got 15 more right and 14 more wrong.) If this 0.1% improvement was measured over 1,000,000 examples, perhaps this is more impressive.

How can you evaluate models?

Type I and Type II errors

- Different types of error losing different types of money…
- What is the cost (opportunity and actual) of a false positive?
- What is the cost of a false negative?

Gain Curves

Don’t forget the user

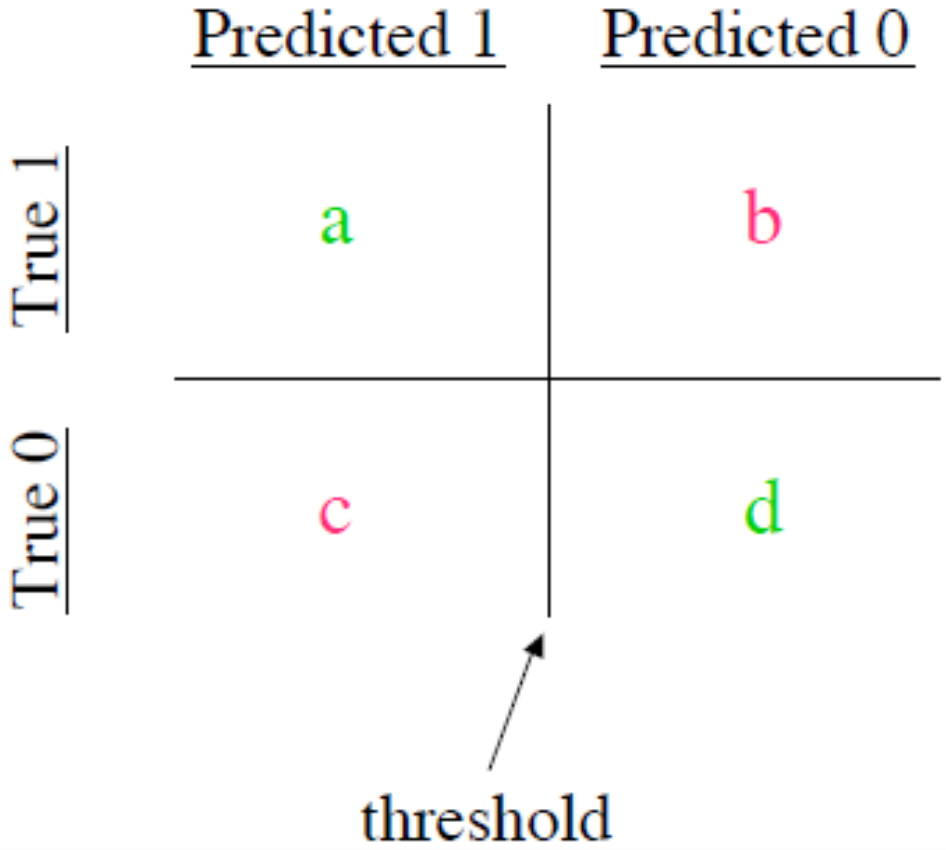
- They may need to be able to explain (or justify) their decision
- Decision trees fight back

Cost Considerations

- Cost / Profit Matrix
- Example: decide whether to offer Personal Equity Plan (PEP)
  - True Positive: Payoff is $1000/customer
  - True Negative: $0
  - False Positive: -$10/customer
  - False negative: -$1000/customer (opp. Cost)

Precision/Recall Curves

Precision and Recall



$$PRECISION = a/(a+c)$$

$$RECALL = a/(a+b)$$

Precision/Recall Curves

Once you can compute precision and recall, you are often able to produce precision/recall curves. Suppose that you are attempting to identify spam. You run a learning algorithm to make predictions on a test set. But instead of just taking a "yes/no" answer, you allow your algorithm to produce its confidence. For instance, using a perceptron, you might use the distance from the hyperplane as a confidence measure. You can then sort all of your test emails according to this ranking. You may put the most spam-like emails at the top and the least spam-like emails at the bottom

Precision/Recall Curves

Once you can compute precision and recall, you are often able to produce precision/recall curves. Suppose that you are attempting to identify spam. You run a learning algorithm to make predictions on a test set. But instead of just taking a "yes/no" answer, you allow your algorithm to produce its confidence. For instance, using a perceptron, you might use the distance from the hyperplane as a confidence measure. You can then sort all of your test emails according to this ranking. You may put the most spam-like emails at the top and the least spam-like emails at the bottom

Once you have this sorted list, you can choose how aggressively you want your spam filter to be by setting a threshold anywhere on this list. One would hope that if you set the threshold very high, you are likely to have high precision (but low recall). If you set the threshold very low, you'll have high recall (but low precision). By considering every possible place you could put this threshold, you can trace out a curve of precision/recall values. This allows us to ask the question: for some fixed precision, what sort of recall can I get…

F Score

Sometimes we want a single number that informs us of the quality of the solution. A popular way to combine precision and recall into a single number is by taking their harmonic mean. This is known as the balanced f-measure:

$$F = \frac{2 \times P \times R}{P + R}$$

The reason to use a harmonic mean rather than an arithmetic mean is that it favors systems that achieve roughly equal precision and recall. In the extreme case where P = R, then F = P = R. But in the imbalanced case, for instance P = 0.1 and R = 0.9, the overall f-measure is a modest 0.18.

	0.0	0.2	0.4	0.6	0.8	1.0
0.0	0.00	0.00	0.00	0.00	0.00	0.00
0.2	0.00	0.20	0.26	0.30	0.32	0.33
0.4	0.00	0.26	0.40	0.48	0.53	0.57
0.6	0.00	0.30	0.48	0.60	0.68	0.74
0.8	0.00	0.32	0.53	0.68	0.80	0.88
1.0	0.00	0.33	0.57	0.74	0.88	1.00

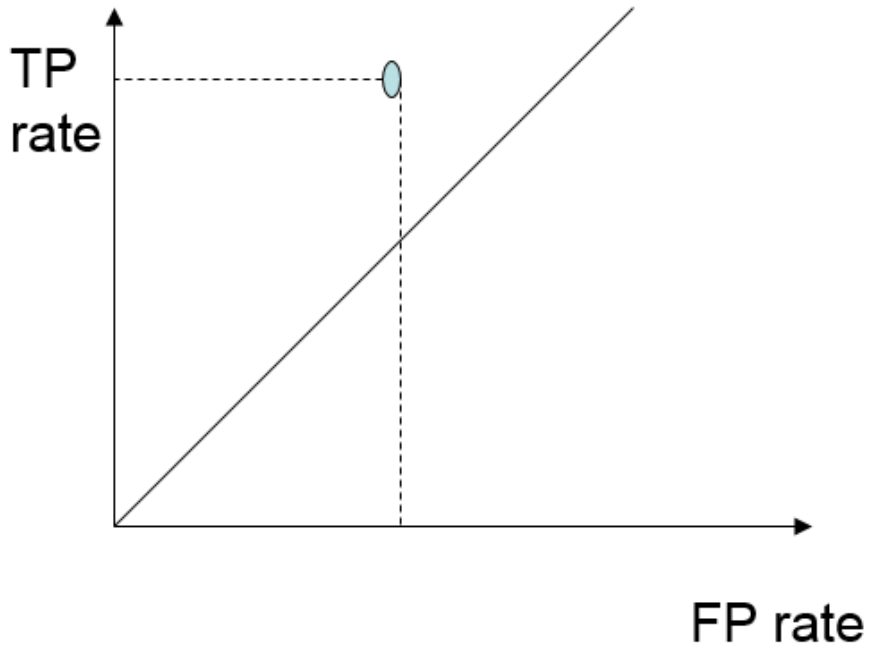
Note…

One thing to keep in mind is that precision and recall (and hence f-measure) *depend crucially on which class is considered* the thing you wish to find. In particular, if you take a binary data set and flip what it means to be a positive or negative example, you will end up with completely different precision and recall values.

It is *not the case that precision on the flipped task is equal to recall on the original task* (nor vice versa). Consequently, f-measure is also not the same. For some tasks where you are less sure about what you want, report two sets of precision/recall/f-measure numbers, which vary based on which class is considered the thing to spot.

Point in ROC Space

		TRUE CLASS	
		YES	NO
PREDICTED CLASS	YES	TP	FP
	NO	FN	TN
Total:		P	N



FP rate: FP/N TP rate: TP/P (recall)

FN rate: FN/N TN rate: TN/P

Classifier accuracy: (TP+TN)/(P+N)

Shows how good is classifier in discriminating positive instances from negative ones

ROC Curve of a Probabilistic Classifier

Naïve Bayes, for example, outputs the probability of an instance in a testing set to be classified as YES

Outlook	Temp	Windy	**P(Y|E)**	Real class
overcast	mild	yes	**0.95**	YES
rainy	mild	no	**0.80**	YES
rainy	cool	yes	**0.60**	NO
sunny	mild	no	**0.45**	YES
sunny	cool	no	**0.40**	NO
sunny	hot	no	**0.35**	NO
sunny	hot	yes	**0.25**	NO

ROC Curve of a Probabilistic Classifier

In a general case, we classify an instance as YES if the probability is more than 50%

Outlook	Temp	Windy	P(Y|E)	Real class
overcast	mild	yes	**0.95**	YES
rainy	mild	no	**0.80**	YES
rainy	cool	yes	**0.60**	NO
sunny	mild	no	**0.45**	YES
sunny	cool	no	**0.40**	NO
sunny	hot	no	**0.35**	NO
sunny	hot	yes	**0.25**	NO

Classified as YES

Classified as NO

Operating threshold

ROC Curve of a Probabilistic Classifier

We compute the confusion matrix

		TRUE CLASS	
		YES	NO
PREDICTED CLASS	YES	2 (TP)	1 (FP)
	NO	1 (FN)	3 (TN)
Total:		3 (P)	4 (N)

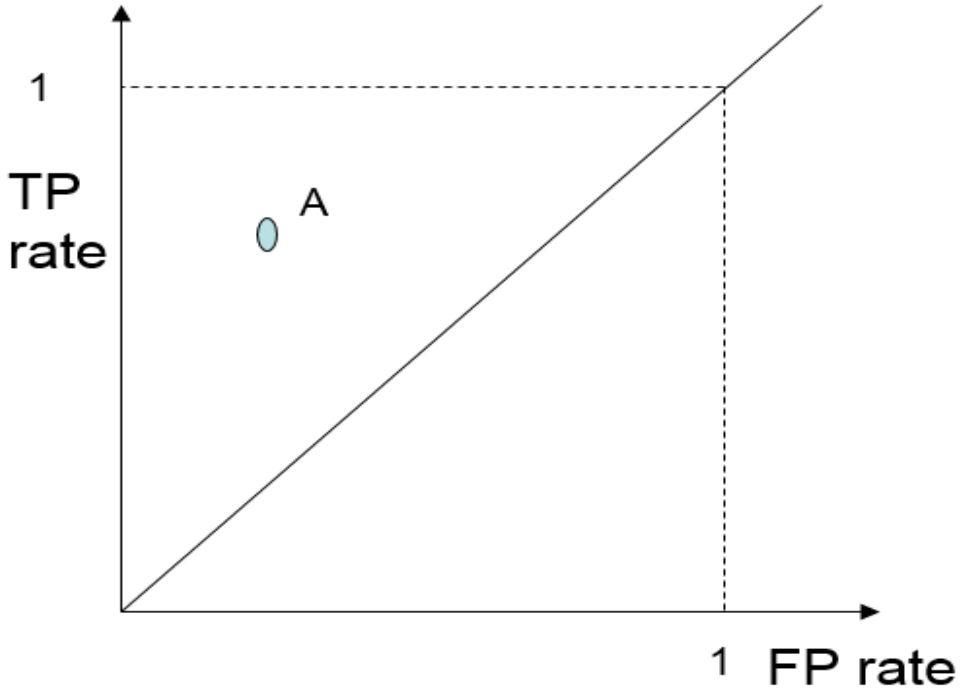
Outlook	Temp	Windy	P(Y|E)	Predicted class	Real class
overcast	mild	yes	0.95	YES	YES
rainy	mild	no	0.80	YES	YES
rainy	cool	yes	0.60	YES	NO
sunny	mild	no	0.45	NO	YES
sunny	cool	no	0.40	NO	NO
sunny	hot	no	0.35	NO	NO
sunny	hot	yes	0.25	NO	NO

And the TP and FP rates:

TP rate: TP/P=2/3≈0.7

FP rate: FP/N=1/4=0.25

ROC Curve of a Probabilistic Classifier



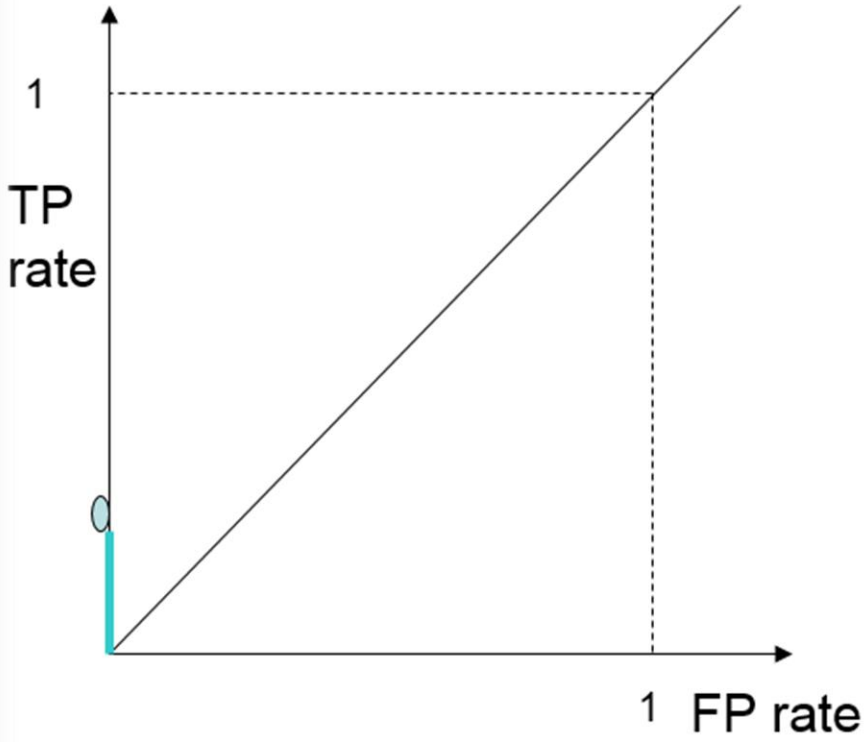
This corresponds to point A in a ROC space

FP rate: FP/N=1/4=0.25

TP rate: TP/P=2/3≈0.7

Outlook	Temp	Windy	P(Y|E)	Predicted class	Real class
overcast	mild	yes	0.95	YES	YES
rainy	mild	no	0.80	YES	YES
rainy	cool	yes	0.60	YES	NO
sunny	mild	no	0.45	NO	YES
sunny	cool	no	0.40	NO	NO
sunny	hot	no	0.35	NO	NO
sunny	hot	yes	0.25	NO	NO

ROC Curve of a Probabilistic Classifier



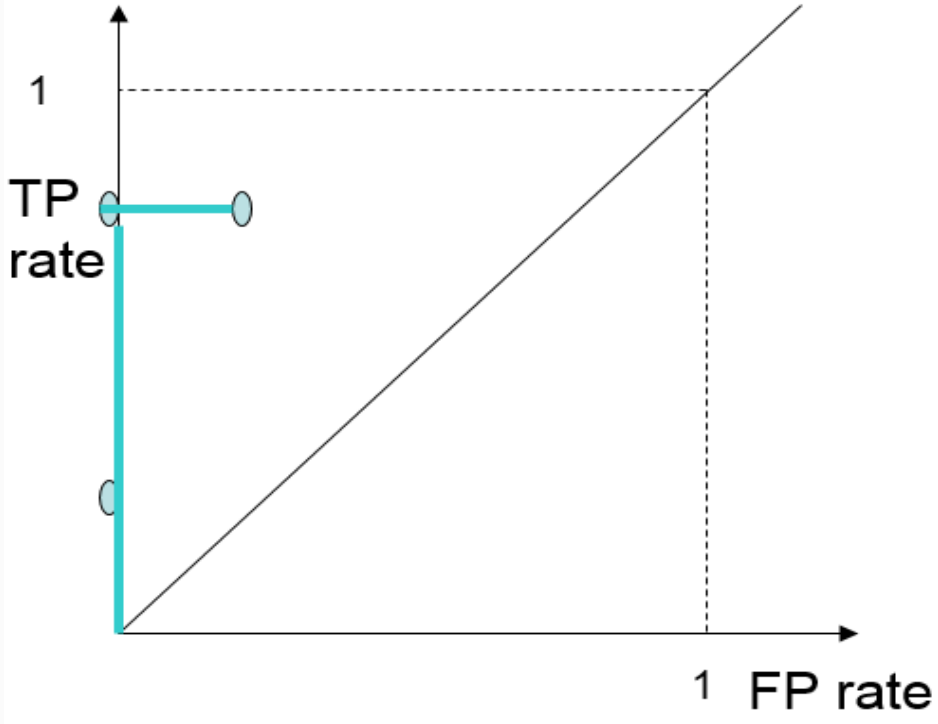
For different threshold values we get different points in a ROC space

Outlook	Temp	Windy	P(Y|E)	Predicted class	Real class
overcast	mild	yes	0.95	YES	YES
rainy	mild	no	0.80	YES	YES
rainy	cool	yes	0.60	YES	NO
sunny	mild	no	0.45	NO	YES
sunny	cool	no	0.40	NO	NO
sunny	hot	no	0.35	NO	NO
sunny	hot	yes	0.25	NO	NO

FP rate: FP/N=0/4=0

TP rate: TP/P=1/3≈0.3

ROC Curve of a Probabilistic Classifier



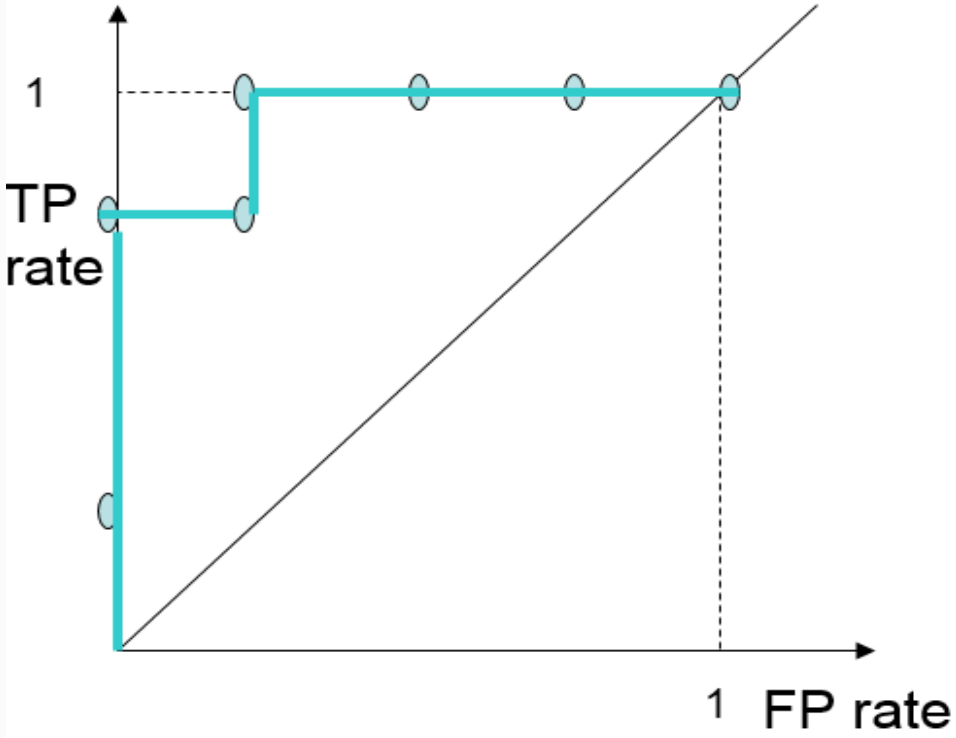
For different threshold values we get different points in a ROC space

Outlook	Temp	Windy	P(Y|E)	Predicted class	Real class
overcast	mild	yes	0.95	YES	YES
rainy	mild	no	0.80	YES	YES
rainy	cool	yes	0.60	YES	NO
sunny	mild	no	0.45	NO	YES
sunny	cool	no	0.40	NO	NO
sunny	hot	no	0.35	NO	NO
sunny	hot	yes	0.25	NO	NO

FP rate: FP/N=1/4=0.25

TP rate: TP/P=2/3≈0.7

ROC Curve of a Probabilistic Classifier



For different threshold values we get different points in a ROC space

Outlook	Temp	Windy	P(Y|E)	Predicted class	Real class
overcast	mild	yes	0.95	YES	YES
rainy	mild	no	0.80	YES	YES
rainy	cool	yes	0.60	YES	NO
sunny	mild	no	0.45	YES	YES
sunny	cool	no	0.40	NO	NO
sunny	hot	no	0.35	NO	NO
sunny	hot	yes	0.25	NO	NO

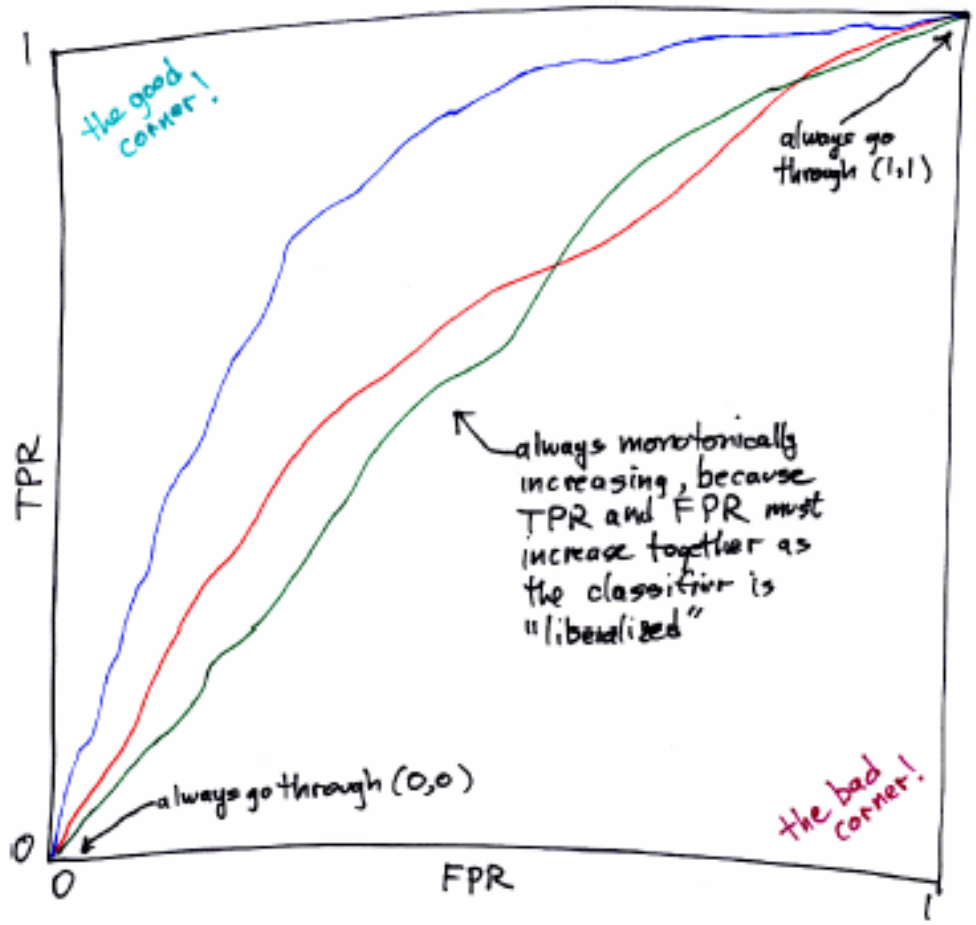
FP rate: FP/N=1/4=0.25

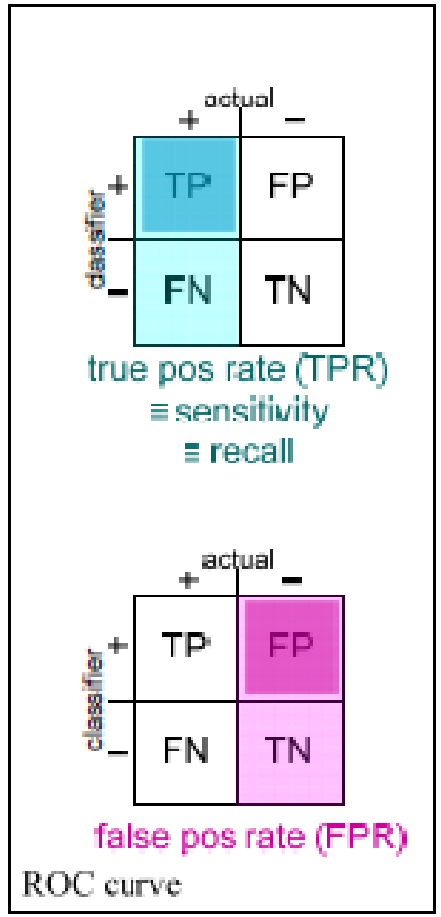
TP rate: TP/P=3/3=1.0, etc…

Performance Metrics

Receiver operating characteristic (ROC) curve

- True positive rate vs. False positive rate for various operating points, as the classifier goes from "conservative" to "liberal"

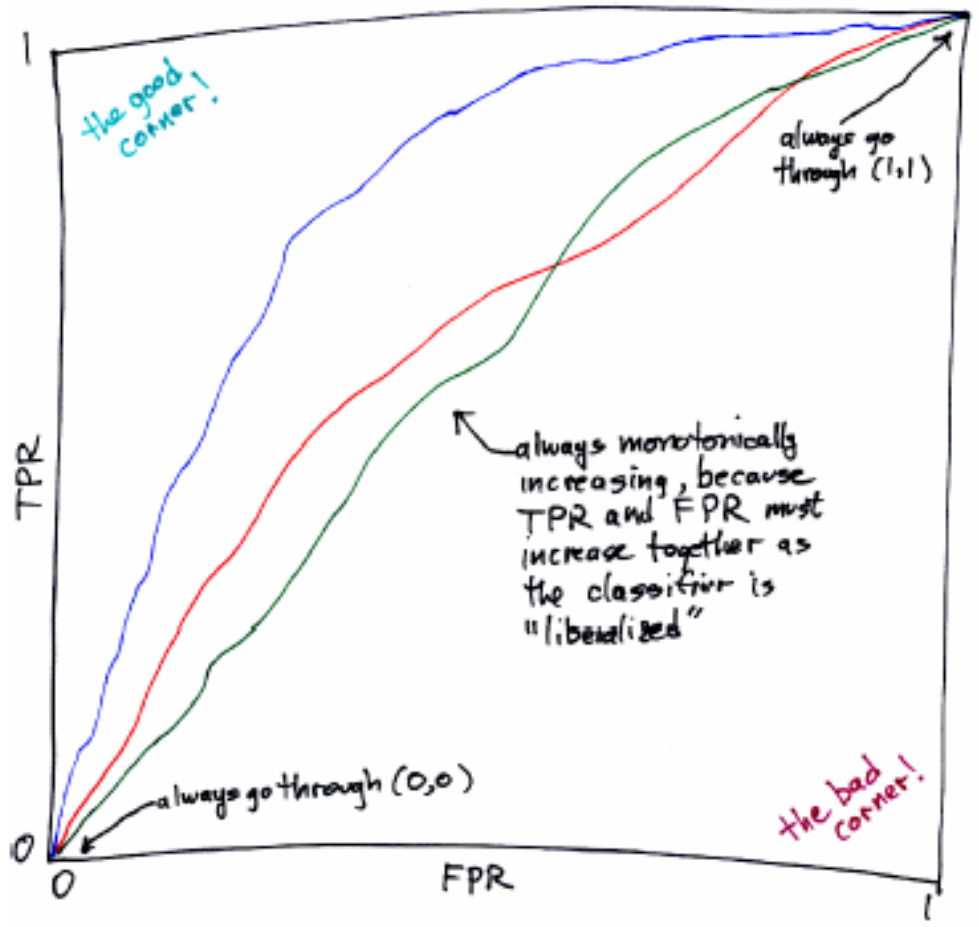




Performance Metrics

Receiver operating characteristic (ROC) curve

- True positive rate vs. False positive rate for various operating points, as the classifier goes from "conservative" to "liberal"



blue dominates red and green

neither red nor green dominate the other

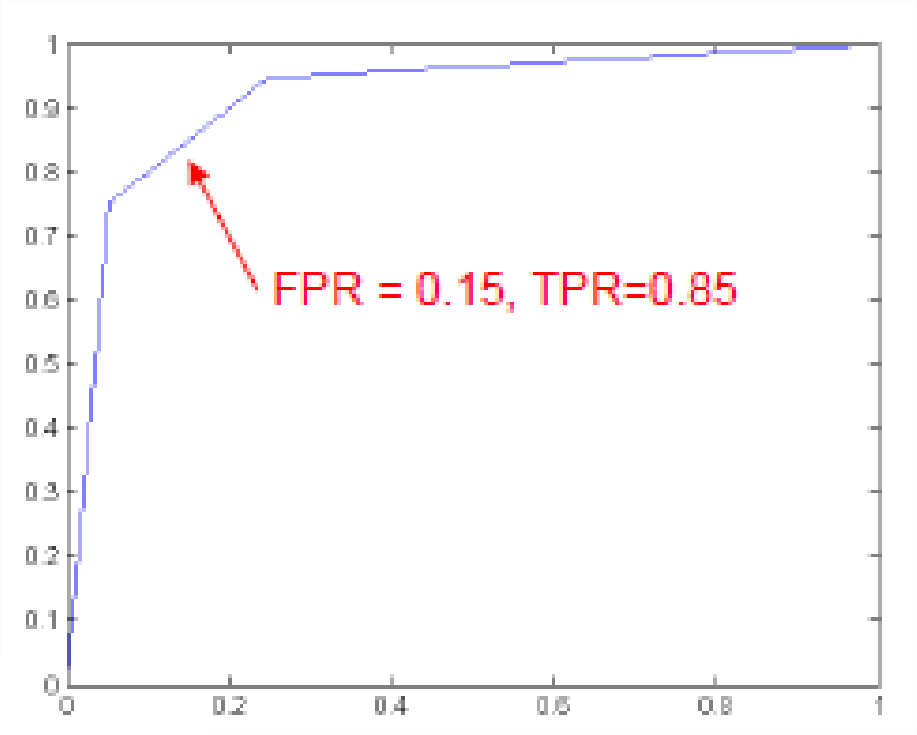
You could get the best of the red and green curves by making a hybrid classifier that switches between strategies at the cross-over points.

Performance Metrics

Since ROC curves don't explicitly show any dependence on the constant P/N (ratio of actual + to – in the sample) they can be misleading if you care about FP versus TP.

Suppose you have a test for Alzheimer's whose false positive rate can be varied from 5% to 25% as the false negative rate varies from 25% to 5% (suppose linear dependences on both):

You try the test on a population of 10,000 people, 1% of whom actually are Alzheimer's positive:



classifier \ actual	+	–
+	85	1485
–	15	8415

FP swamps TP by ~17:1. You'll be telling 17 people that they might have Alzheimer's for every one who actually does. It is unlikely that your test will be used.

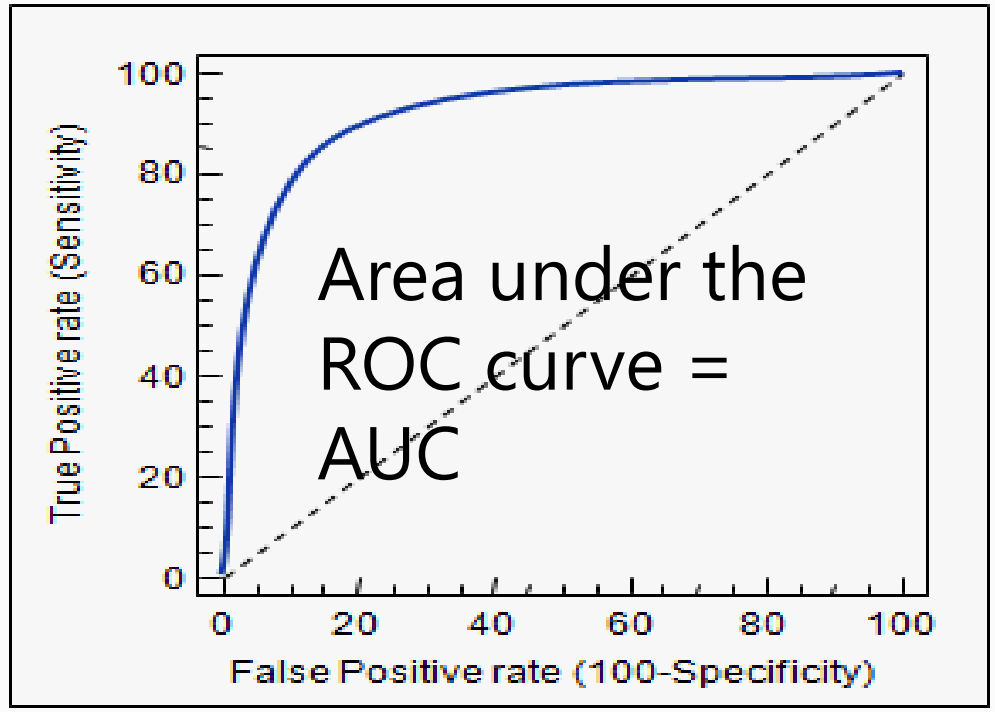
In a case like this, ROC, while correct, somewhat misses the point.

ROC Curve

- Area under the ROC curve (AUC) is a measure of the model performance

$$0.5\ (random\ model) < AUC < 1\ (perfect\ model)$$

- Larger the AUC, better is the model



Performance Metrics

Receiver operating characteristic (ROC) curve, AUC metric…

- 1.0: perfect prediction
- 0.9: excellent prediction
- 0.8: good prediction
- 0.7: mediocre prediction
- 0.6: poor prediction
- 0.5: random prediction
- <0.5: something wrong!

Lecture 5 Homework 5a

Targeted Marketing Campaign

In this problem we will use historical data from past customer responses to build a classification model. The model will then be applied to a new set of prospects to whom we may want extend an offer for a PEP. Rather than doing a mass marketing campaign to all new prospects, we would like to target those that are likely to respond positively to our offer (according to our classification model).

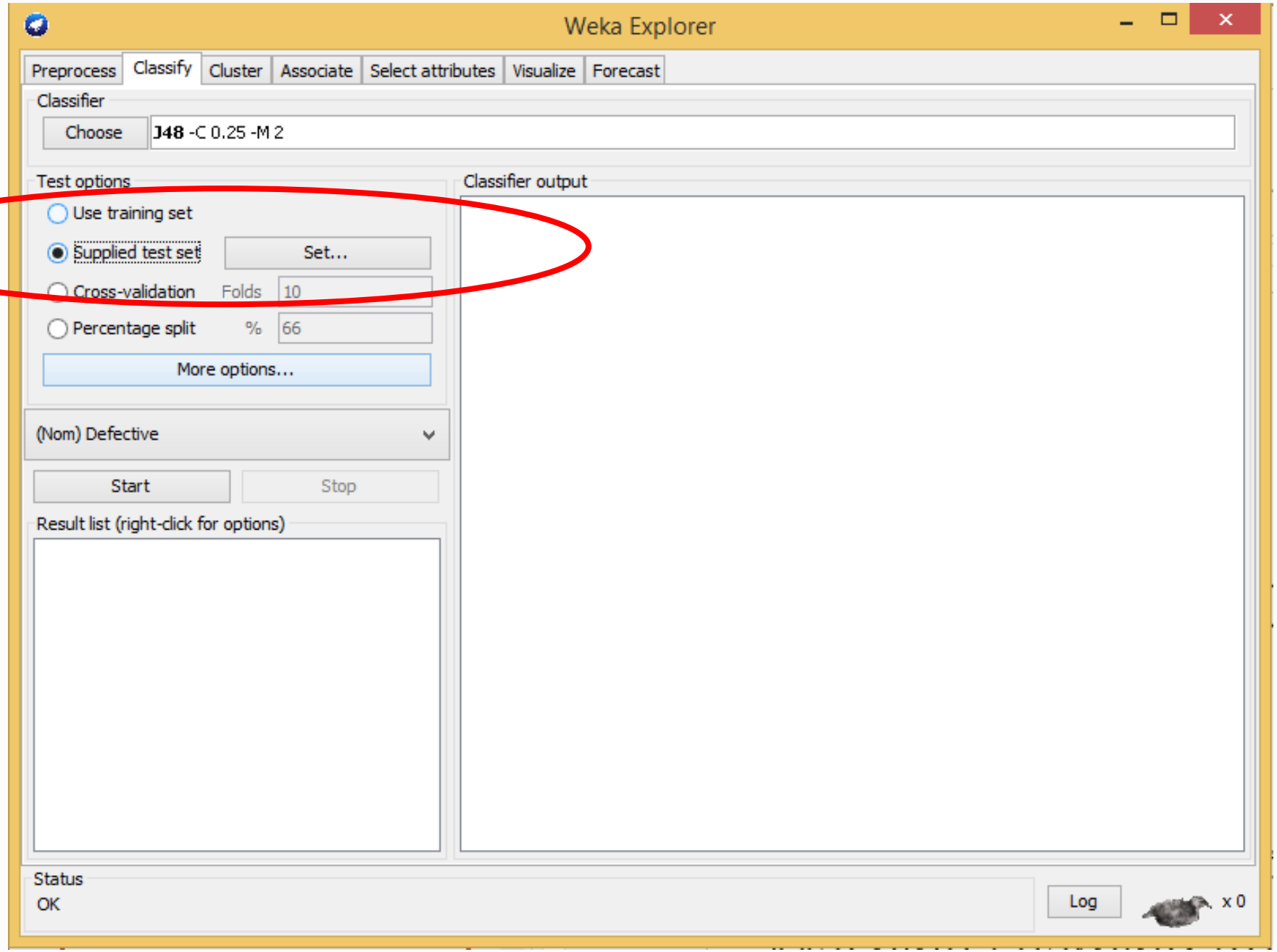
There are two data sets available (the data sets are comma delimited, and the first row contains the field names):

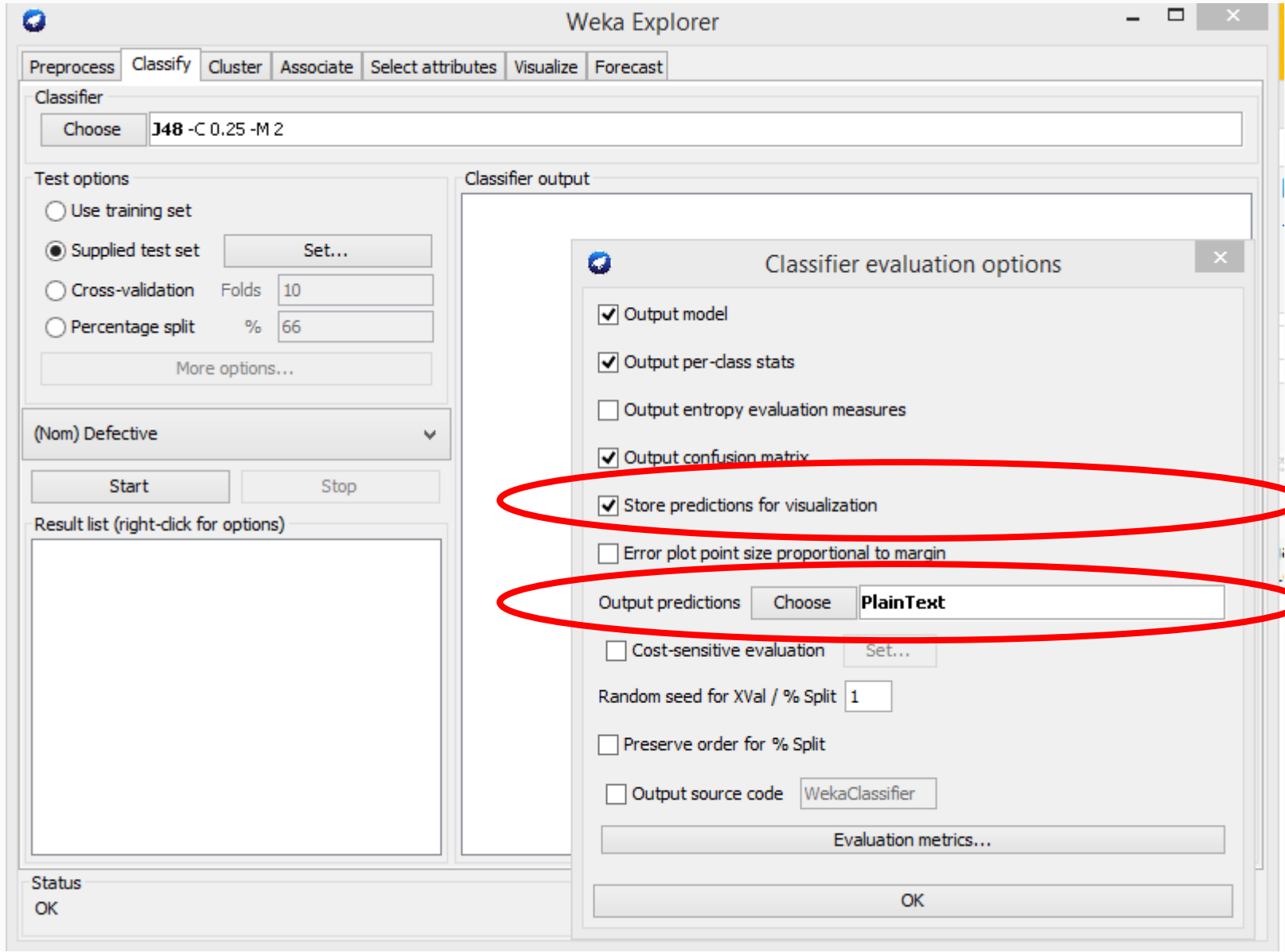
- bank-data.csv - Labelled training data set for Building a Model
- bank-new.csv - A set of new customers from which to find the "hot prospects" for the next mailing, using the profiles built from the training set.

- Decision Tree, 10-fold cross validation;
- Make predictions with the model against test data set;
- False Positive costs $10, False Negative results in a $1000 loss;
- Lift charts over predictive data;

From model construction to prediction to impact…

1. Build our predictive model in WEKA Explorer;
2. Use our model to score (predict) which new customers to target in our upcoming advertising campaign;
   - ARFF file manipulation (hacking), all too common pita…
   - Excel manipulation to join model output with our customers list
3. Compute the lift chart to assess business impact of our predictive model on the advertising campaign
   - How are Lift charts built, of all the charts and/or performance measures from a model this one is 'on you' to construct;
   - Where is the business 'bang for the buck'?


Weka Explorer
Preprocess
Classify
Cluster
Associate
Select attributes
Visualize
Forecast
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...
Classifier output
(Nom) Defective
Start
Stop
Result list (right-click for options)
Status
OK
Log
x 0


Weka Explorer
Preprocess
Classify
Cluster
Associate
Select attributes
Visualize
Forecast
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...
Classifier output
(Nom) Defective
Start
Stop
Result list (right-click for options)
Status
OK
Classifier evaluation options
Output model
Output per-class stats
Output entropy evaluation measures
Output confusion matrix
Store predictions for visualization
Error plot point size proportional to margin
Output predictions
Choose
PlainText
Cost-sensitive evaluation
Set...
Random seed for XVal / % Split
1
Preserve order for % Split
Output source code
WekaClassifier
Evaluation metrics...
OK

How to avoid a time suck…

Ensemble Methods

Bagging

- Select many independent samples from training data, *with replacement…*
- Learn one model on each sample (training set), can be done in parallel
- Classifier: Majority voting of the various models

Boosting

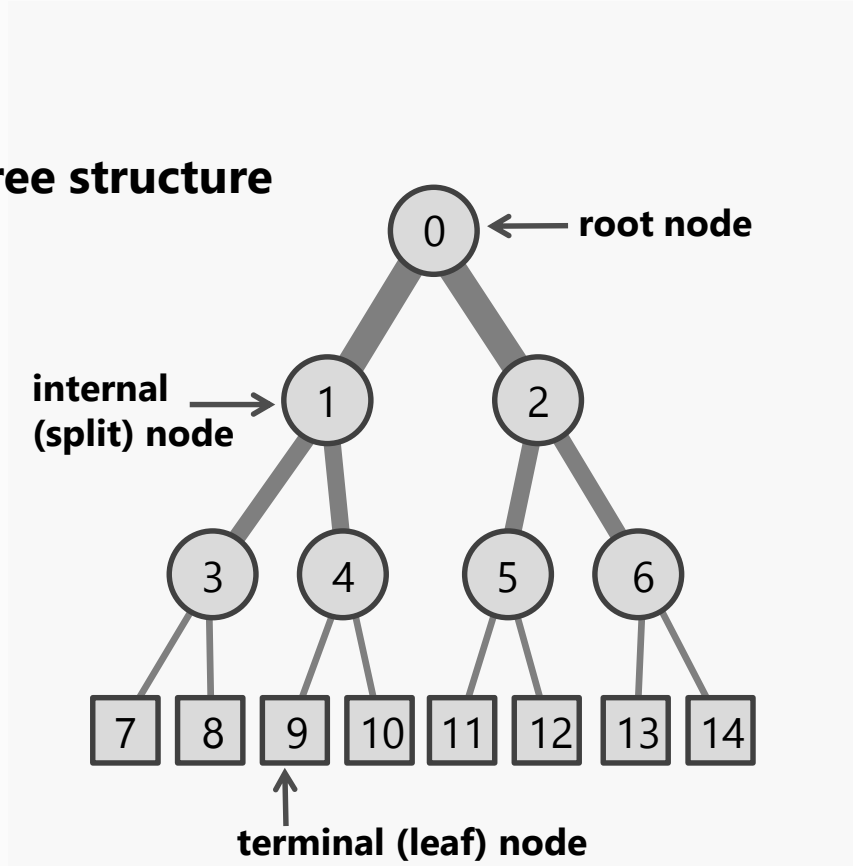
- Learns a sequence of models
- Each model focuses on examples that previous models did badly on
- Learns a weighting of the models

Decision Trees:

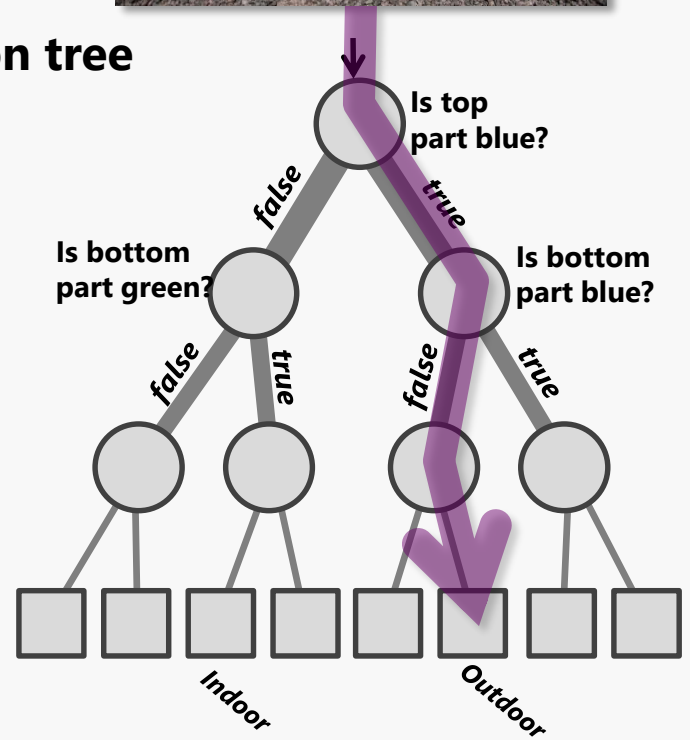
- With **bagging** (+random selection of m features to use with each node): *Random Forest*
- With **boosting**: *Boosted Decision Trees* (Bing ranking of search results)

Decision Trees and Decision Forests

A general tree structure



A decision tree



A forest is an ensemble of trees. The trees are all slightly different from one another.

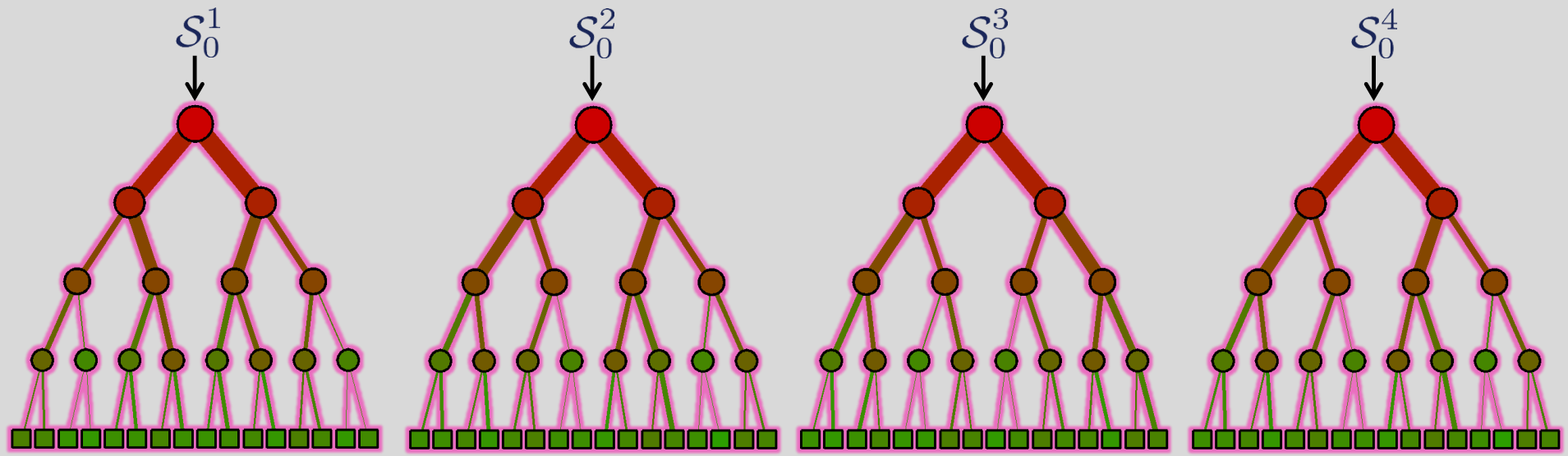
Decision Forest Model: the randomness model

1) Bagging (randomizing the training set)

$\mathcal{S}_0$ The full training set

$\mathcal{S}_0^t \subset \mathcal{S}_0$ The randomly sampled subset of training data made available for the tree *t*

Forest training



Decision Forest Model: the randomness model

2) Randomized node optimization (RNO)

$\mathcal{T}$ The full set of all possible node test parameters

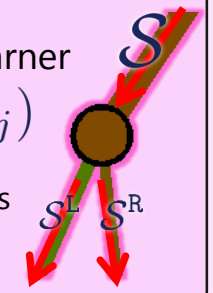
$\mathcal{T}_j \subset \mathcal{T}$ For each node the set of randomly sampled features

$\rho = |\mathcal{T}_j|$ Randomness control parameter.
For $\rho = |\mathcal{T}|$ no randomness and maximum tree correlation.
For $\rho = 1$ max randomness and minimum tree correlation.

Node training

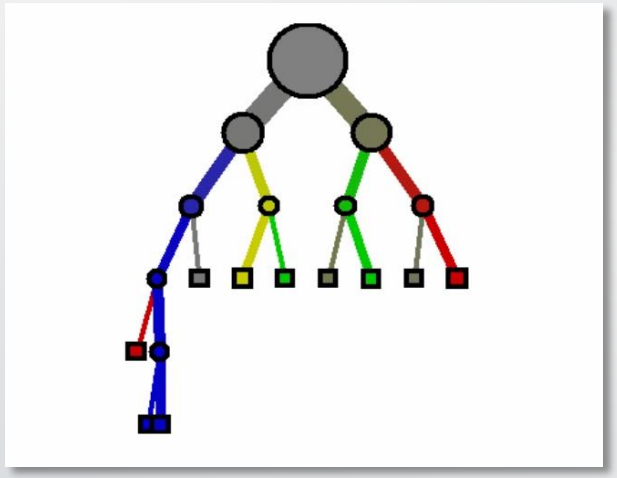
Node weak learner $h(\mathbf{v}, \boldsymbol{\theta}_j)$

Node test params $\boldsymbol{\theta} \in \mathcal{T}_j$

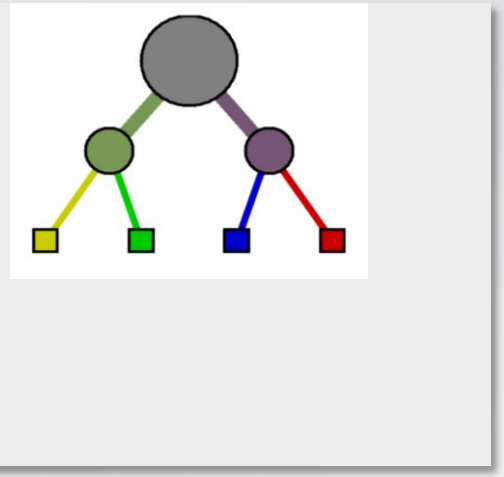


The effect of ρ

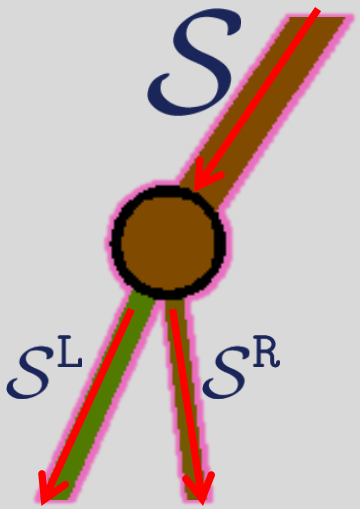
Small value of ρ; little tree correlation.



Large value of ρ; large tree correlation.



Decision Forest Model: training and information gain



Information gain

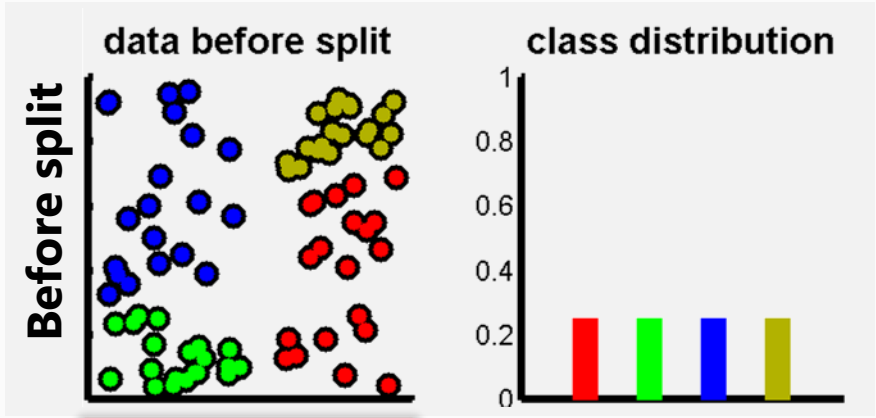
$$I(\mathcal{S}, \boldsymbol{\theta}) = H(\mathcal{S}) - \sum_{i \in \{\mathrm{L},\mathrm{R}\}} \frac{|\mathcal{S}^i|}{|\mathcal{S}|} H(\mathcal{S}^i)$$

Shannon's entropy

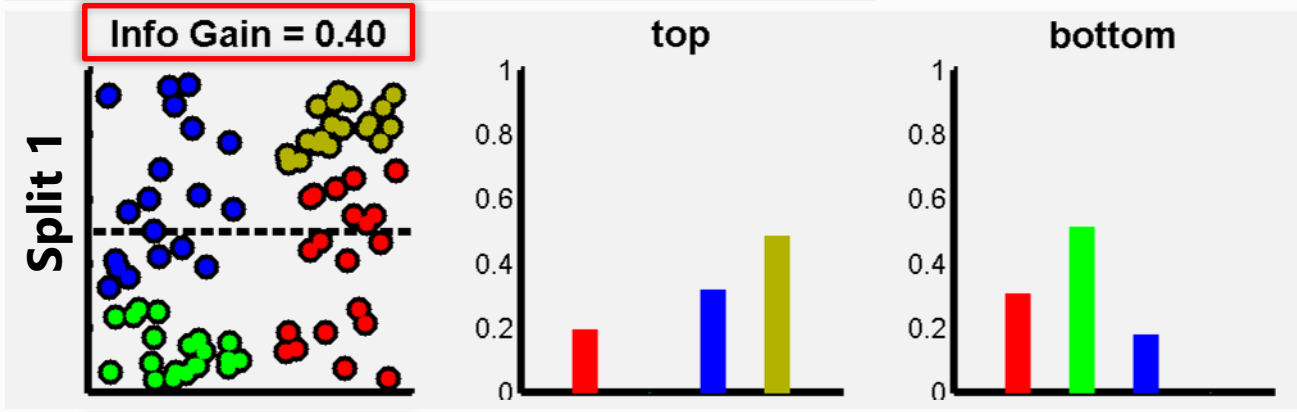
$$H(\mathcal{S}) = -\sum_{c \in \mathcal{C}} p(c) \log(p(c))$$

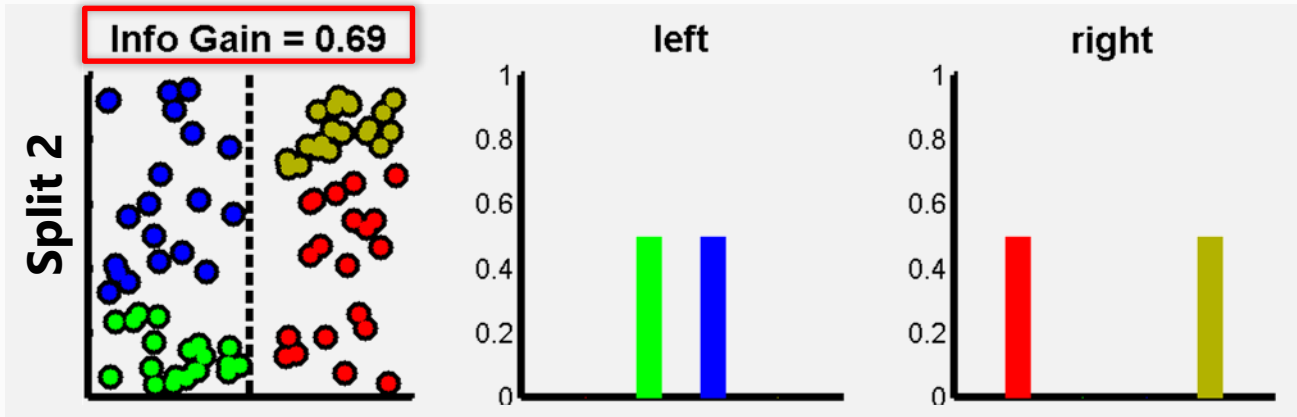
Node training

$$\boldsymbol{\theta} = \arg\max_{\boldsymbol{\theta} \in \mathcal{T}_j} I(\mathcal{S}_j, \boldsymbol{\theta})$$

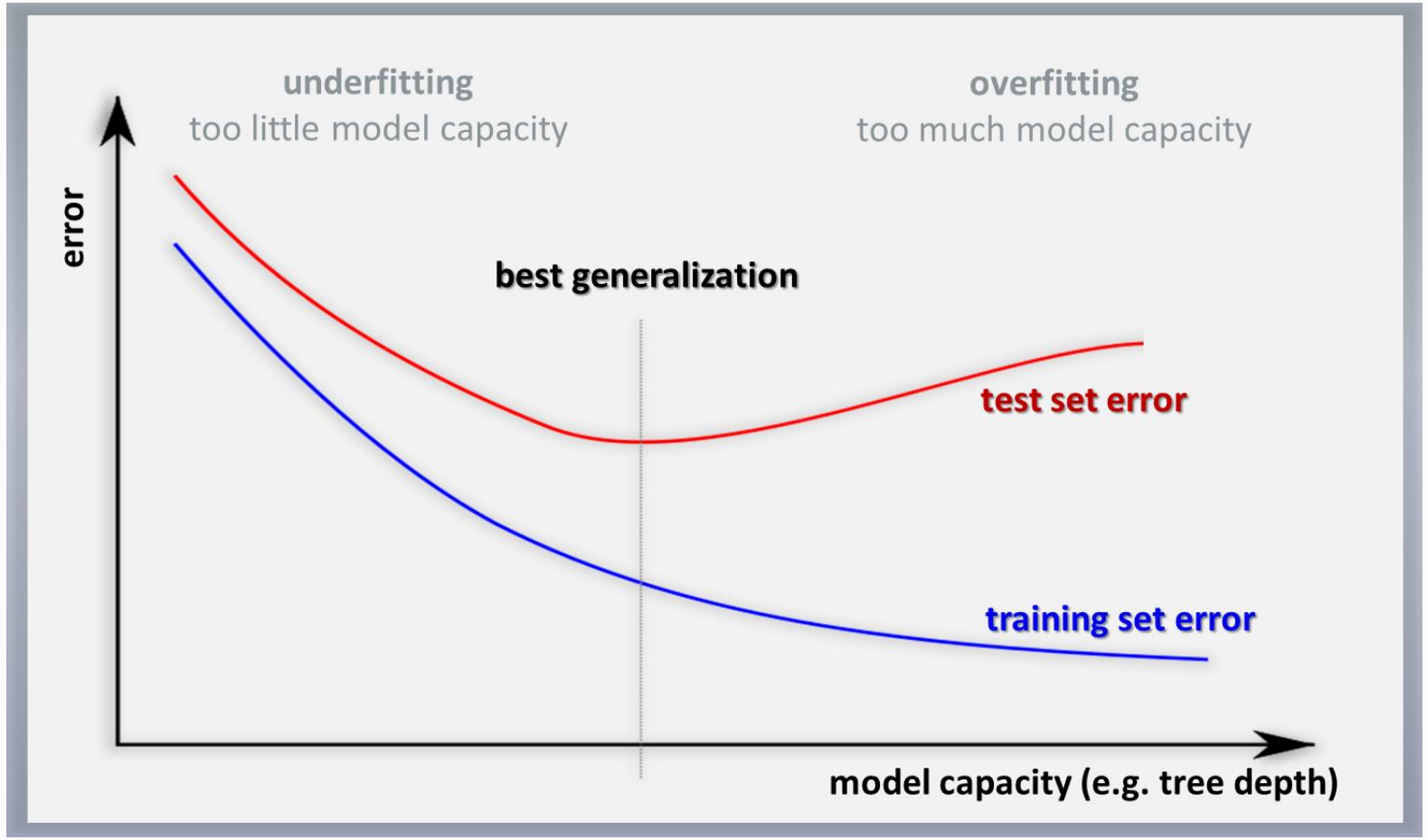


(for categorical, non-parametric distributions)

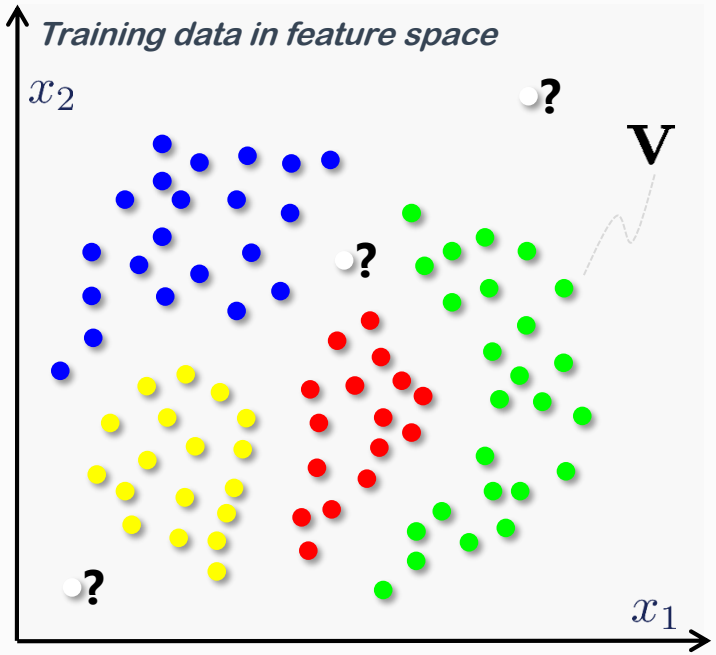


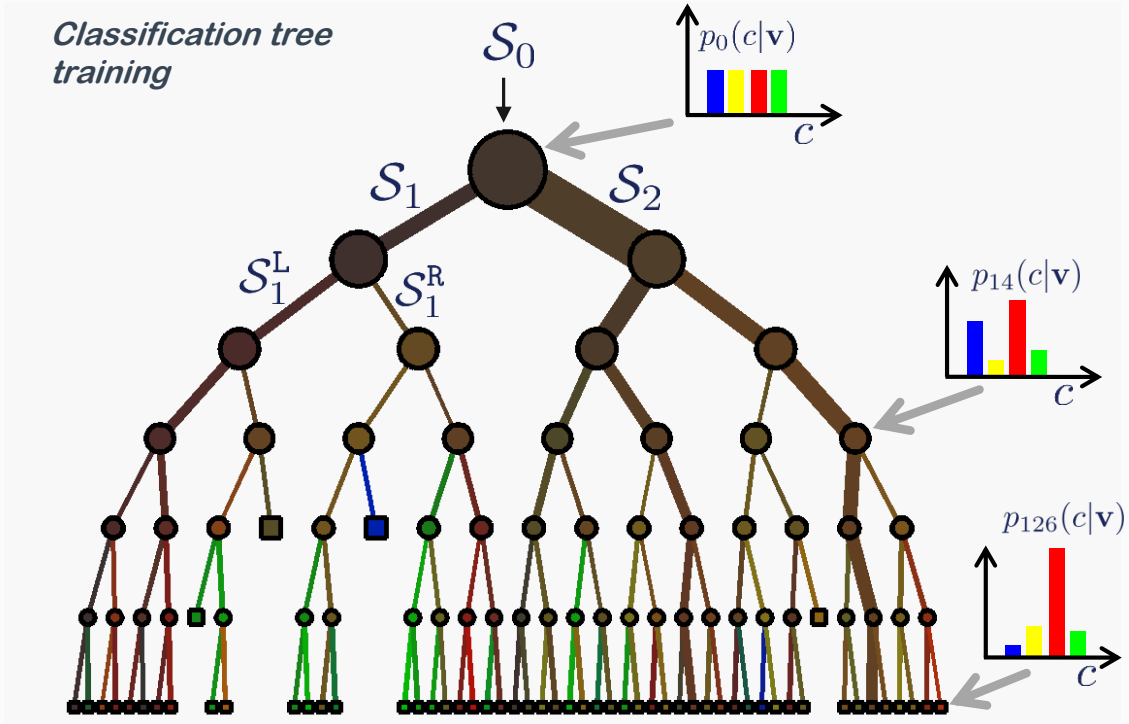


Why we prune…


underfitting
too little model capacity
overfitting
too much model capacity
error
best generalization
test set error
training set error
model capacity (e.g. tree depth)

Classification Forest





Model specialization for classification

Input data point	$\mathbf{v} = (x_1, \cdots, x_d) \in \mathbb{R}^d$	*(x_i is feature response)*
Output is categorical	$c \in \mathcal{C}$ with $\mathcal{C} = \{c_k\}$	*(discrete set)*
Node weak learner	$h(\mathbf{v}, \boldsymbol{\theta}) \in \{\mathtt{true}, \mathtt{false}\}$	
Obj. funct. for node j	$I = H(\mathcal{S}_j) - \sum_{i=\mathrm{L},\mathrm{R}} \frac{\lvert\mathcal{S}_j^i\rvert}{\lvert\mathcal{S}_j\rvert} H(\mathcal{S}_j^i)$	*(information gain)*
Training node j	$\boldsymbol{\theta}_j = \arg\max_{\boldsymbol{\theta} \in \mathcal{T}_j} I(\mathcal{S}_j, \boldsymbol{\theta})$	
Predictor model	$p(c\|\mathbf{v}) = \sum_j p(c\|j)p(j\|\mathbf{v})$	*(class posterior)*

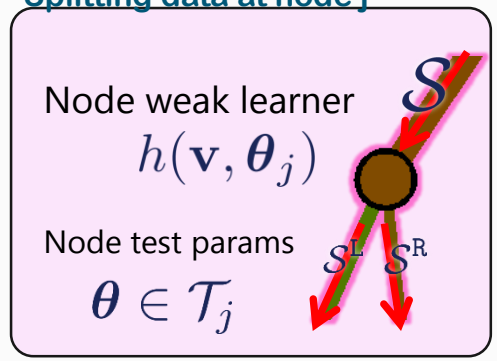
Entropy of a discrete distribution

$$H(\mathcal{S}) = -\sum_{c \in \mathcal{C}} p(c) \log(p(c))$$

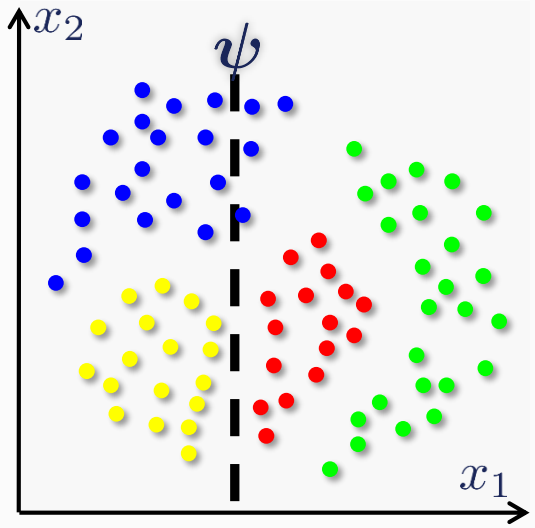
with $c(\mathbf{v}) \; : \; \mathbb{R}^d \rightarrow \mathcal{C}$

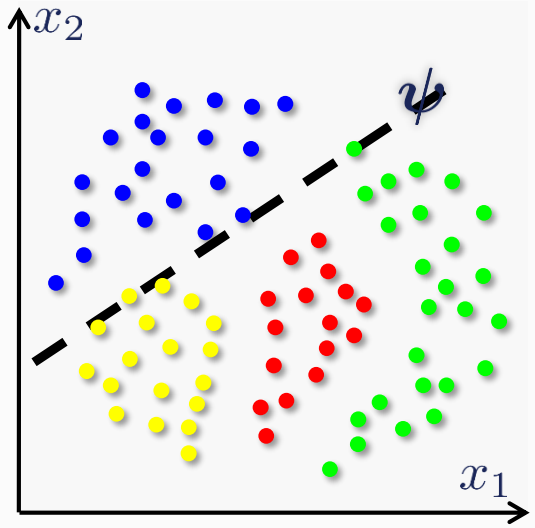
Classification Forest: the weak learner model

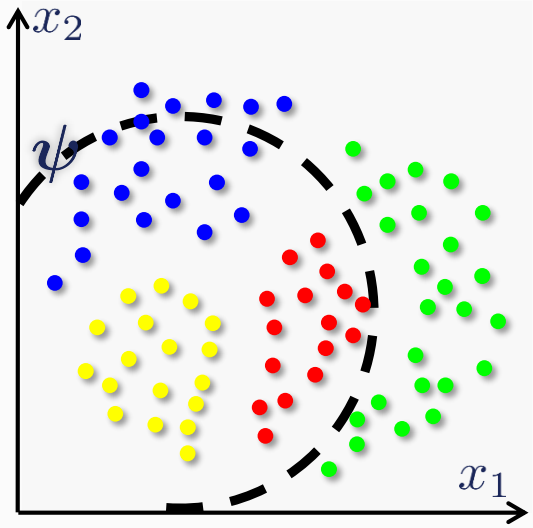
Splitting data at node j



Examples of weak learners







Weak learner: axis aligned

$$h(\mathbf{v}, \boldsymbol{\theta}) = [\tau_1 > \boldsymbol{\phi}(\mathbf{v}) \cdot \boldsymbol{\psi} > \tau_2]$$

Feature response for 2D example. $\boldsymbol{\phi}(\mathbf{v}) = (x_1 \; x_2 \; 1)^\top$

With $\boldsymbol{\psi} = (1 \; 0 \; \psi_3)$ or $\boldsymbol{\psi} = (0 \; 1 \; \psi_3)$

Weak learner: oriented line

$$h(\mathbf{v}, \boldsymbol{\theta}) = [\tau_1 > \boldsymbol{\phi}(\mathbf{v}) \cdot \boldsymbol{\psi} > \tau_2]$$

Feature response for 2D example. $\boldsymbol{\phi}(\mathbf{v}) = (x_1 \; x_2 \; 1)^\top$

With $\boldsymbol{\psi} \in \mathbb{R}^3$ a generic line in homog. coordinates.

Weak learner: conic section

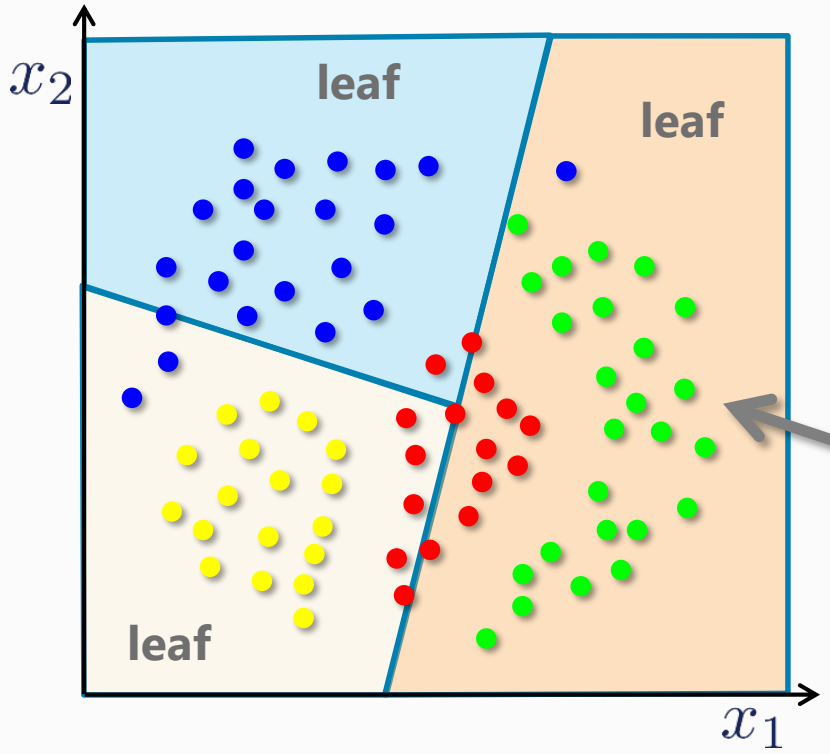
$$h(\mathbf{v}, \boldsymbol{\theta}) = \left[\tau_1 > \boldsymbol{\phi}^\top(\mathbf{v}) \; \boldsymbol{\psi} \; \boldsymbol{\phi}(\mathbf{v}) > \tau_2\right]$$

Feature response for 2D example. $\boldsymbol{\phi}(\mathbf{v}) = (x_1 \; x_2 \; 1)^\top$

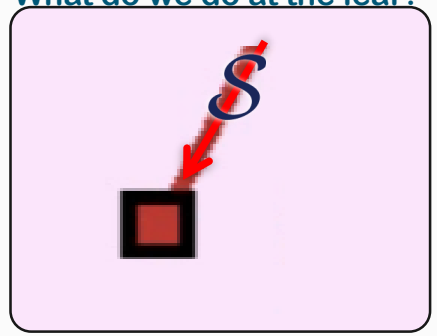
With $\boldsymbol{\psi} \in \mathbb{R}^{3\times 3}$ a matrix representing a conic.

In general $\boldsymbol{\phi}$ may select only a very small subset of features $\boldsymbol{\phi}(\mathbf{v}) : \mathbb{R}^d \rightarrow \mathbb{R}^{d'+1}, \quad d' << d$

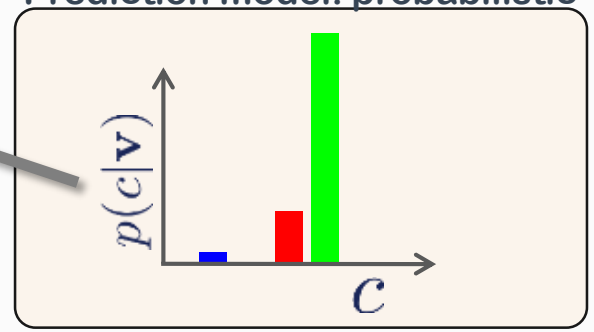
Classification Forest: the prediction model



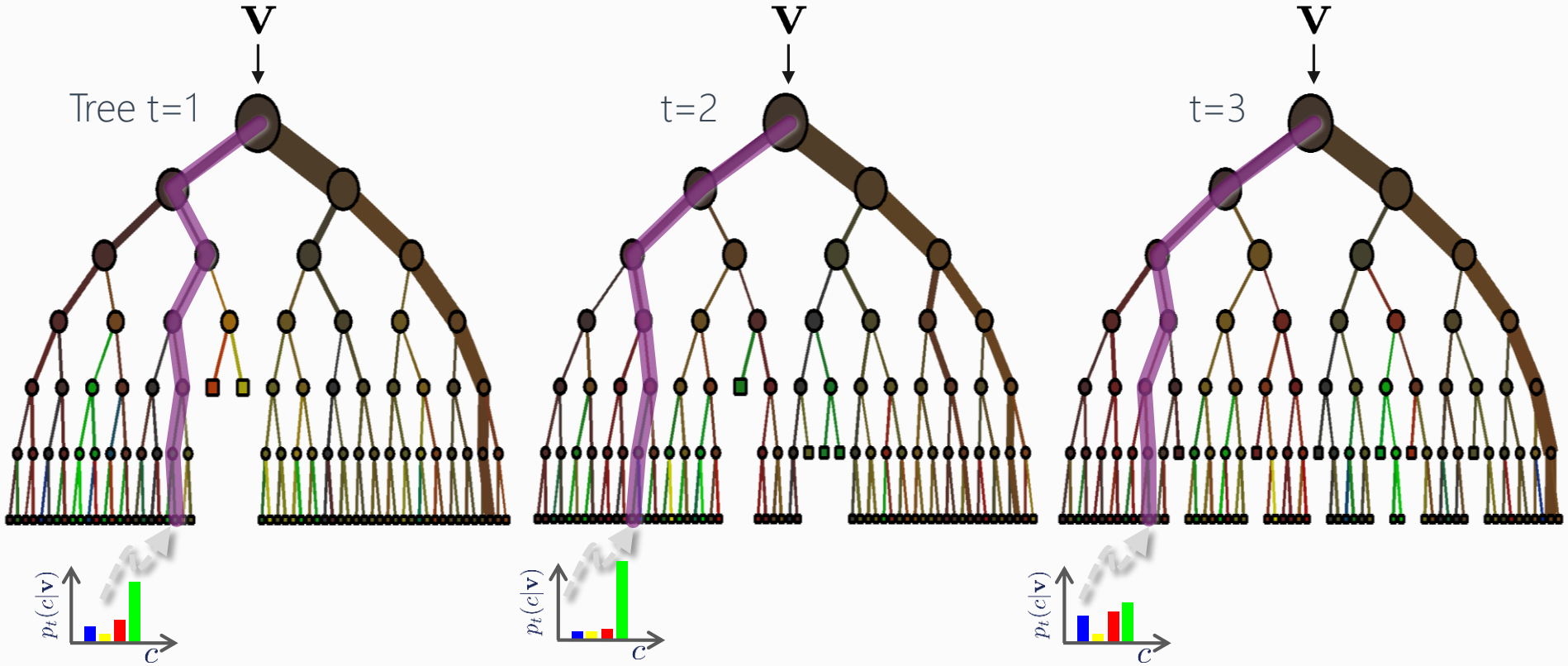
What do we do at the leaf?



Prediction model: probabilistic

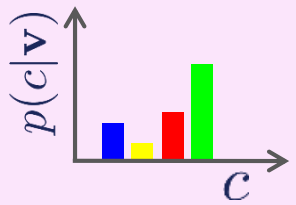


Classification Forest: the ensemble model

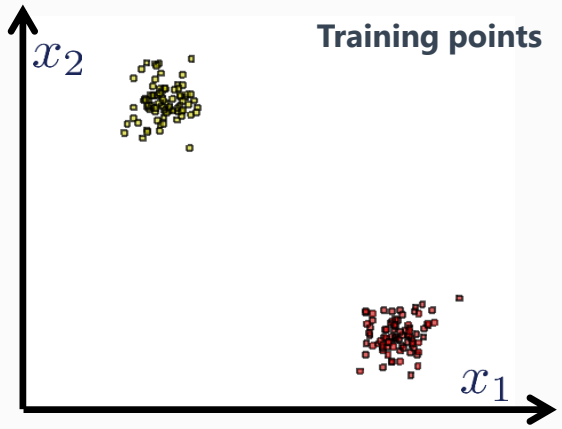


The ensemble model

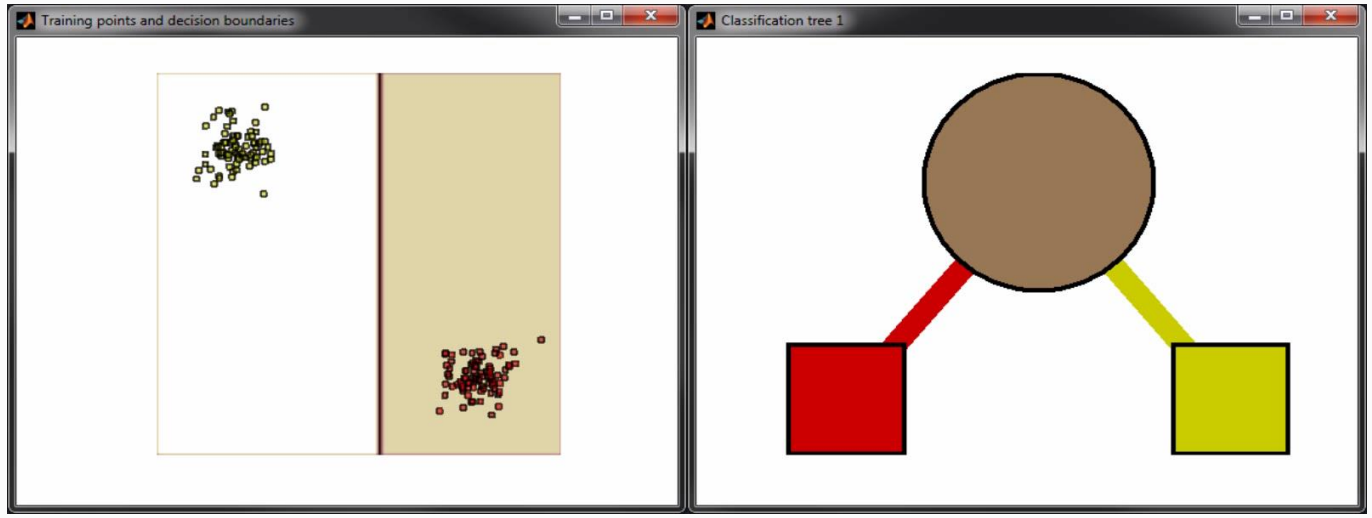
Forest output probability $p(c|\mathbf{v}) = \frac{1}{T}\sum_{t}^{T} p_t(c|\mathbf{v})$



Classification Forest: effect of the weak learner model



Training different trees in the forest



Testing different trees in the forest

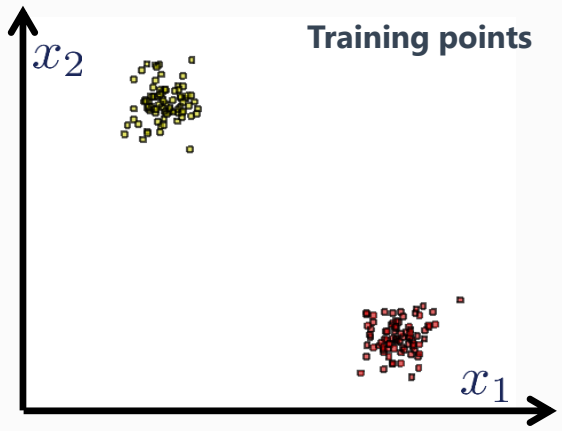
Three concepts to keep in mind:

- "Accuracy of prediction"
- "Quality of confidence"
- "Generalization"

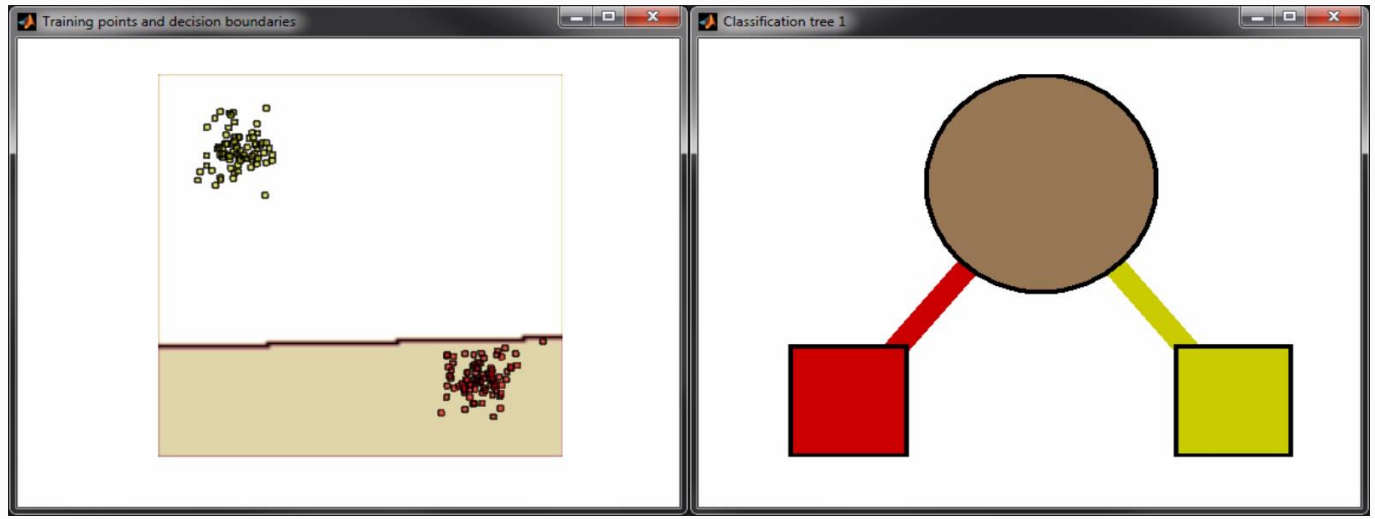
(2 videos in this page)

Parameters: T=200, D=2, weak learner = aligned, leaf model = probabilistic

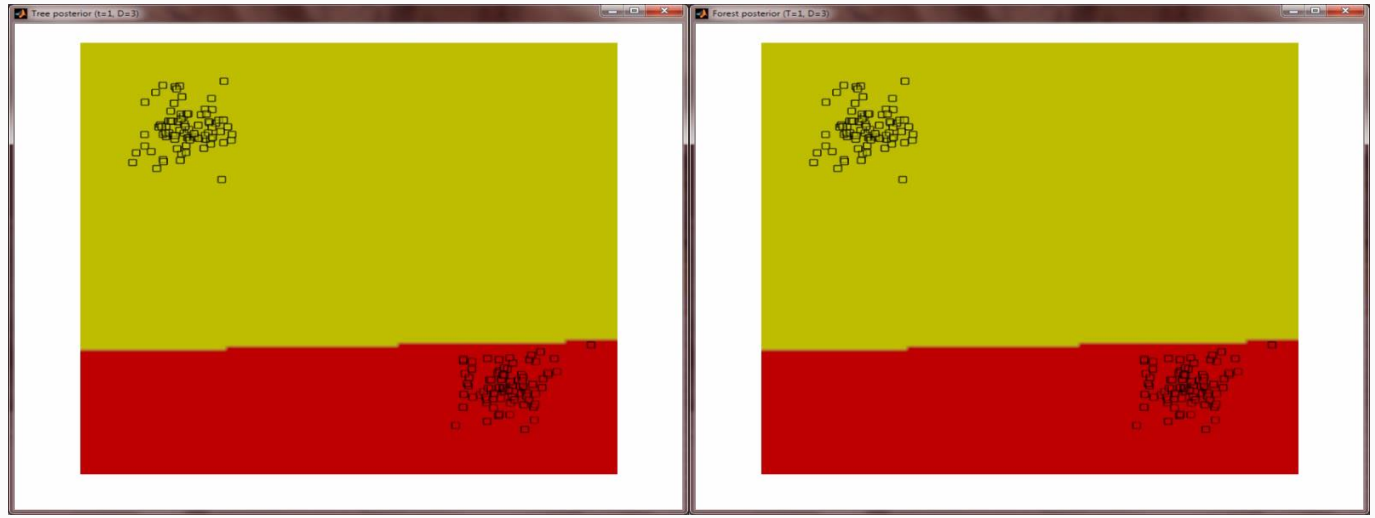
Classification Forest: effect of the weak learner model



Training different trees in the forest

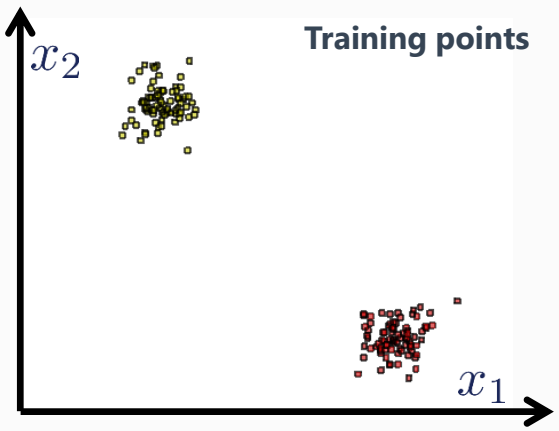


Testing different trees in the forest

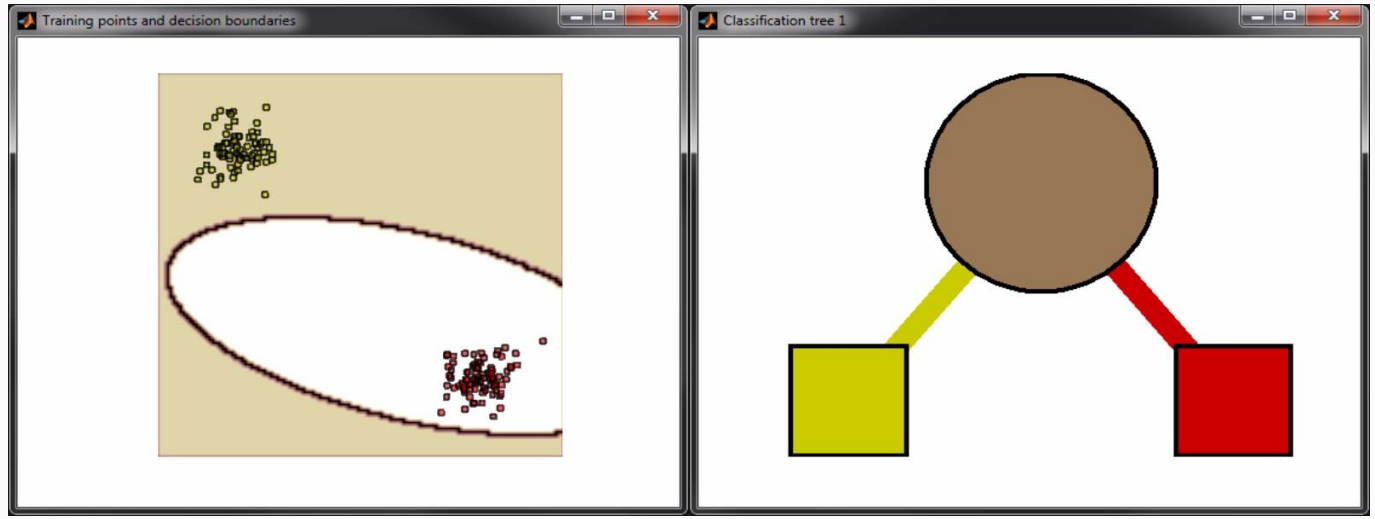


Parameters: T=200, D=2, weak learner = linear, leaf model = probabilistic

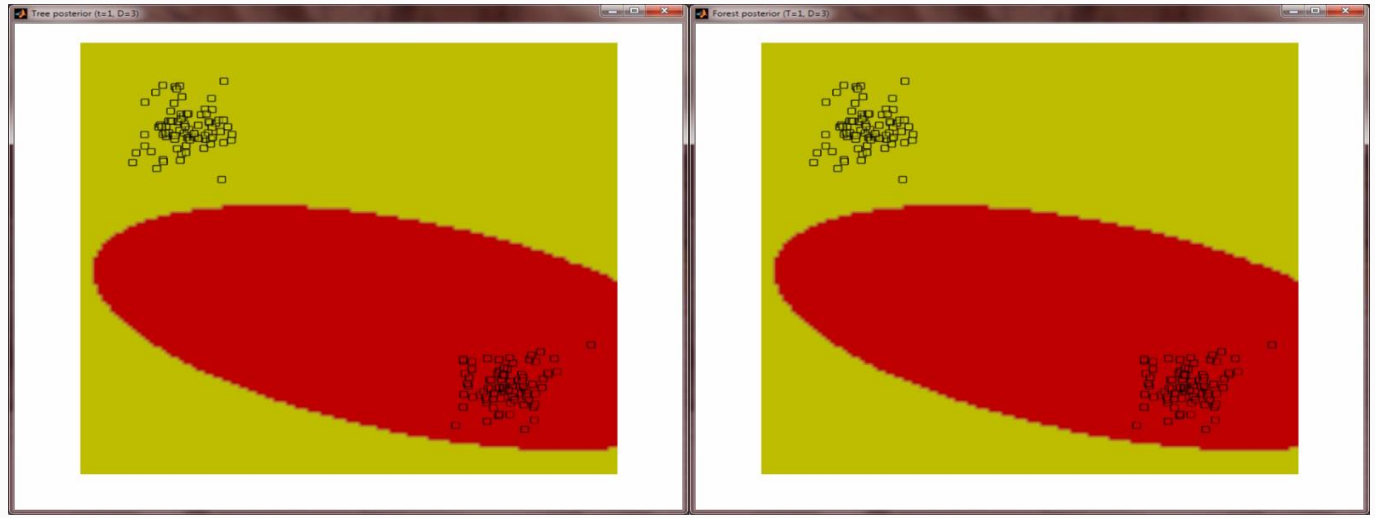
Classification Forest: effect of the weak learner model



Training different trees in the forest



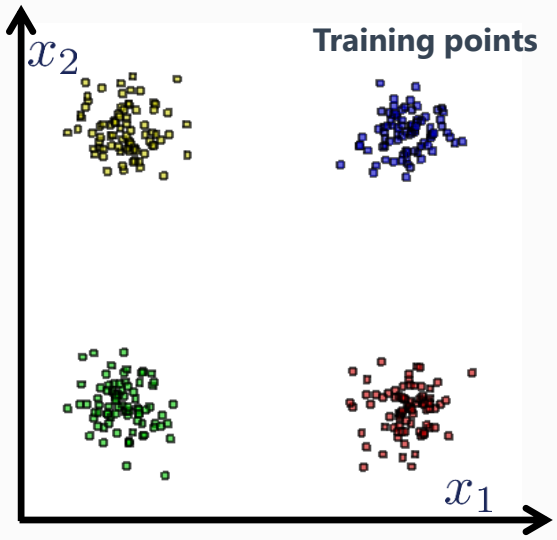
Testing different trees in the forest



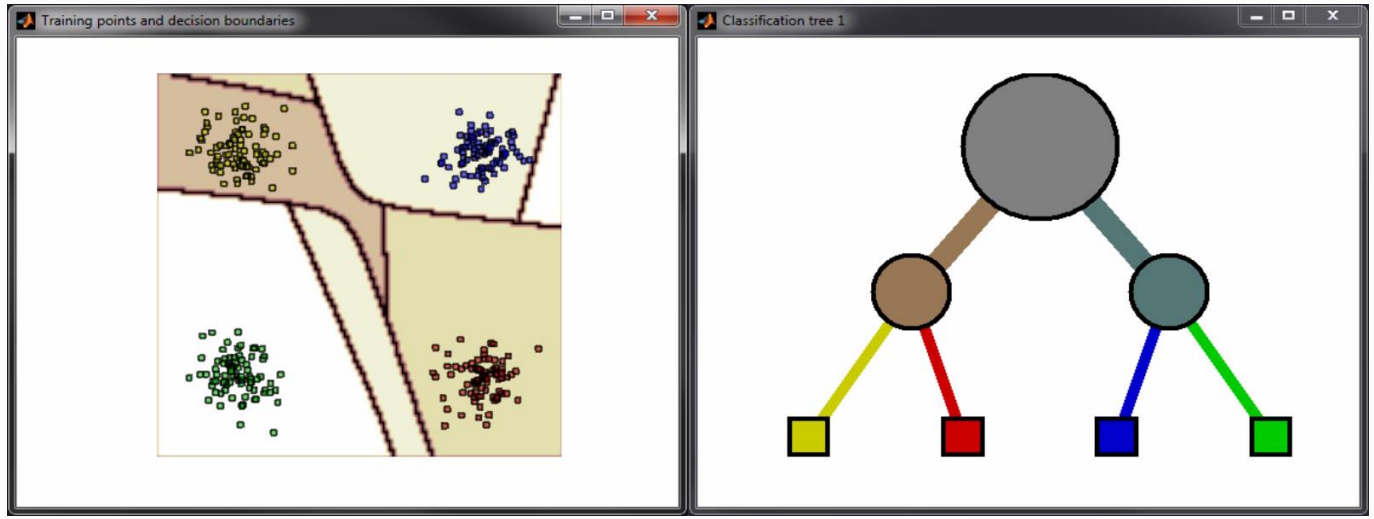
Parameters: T=200, D=2, weak learner = conic, leaf model = probabilistic

(2 videos in this page)

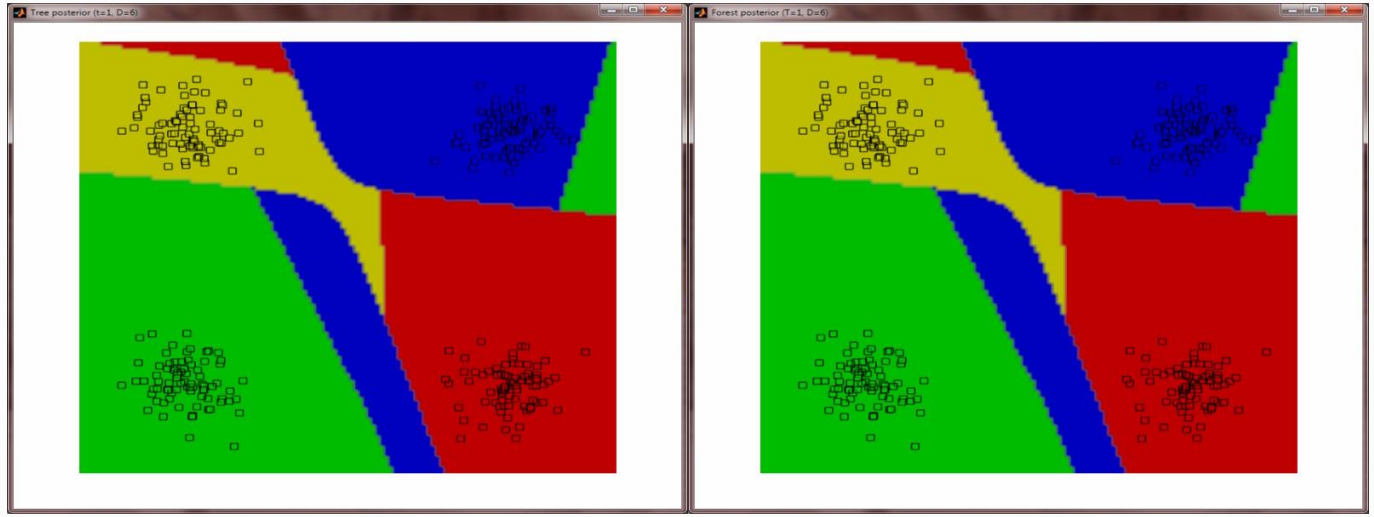
Classification Forest: with >2 classes



Training different trees in the forest



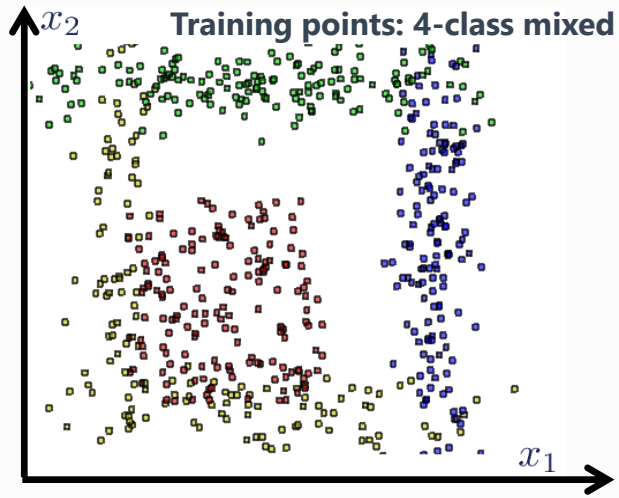
Testing different trees in the forest

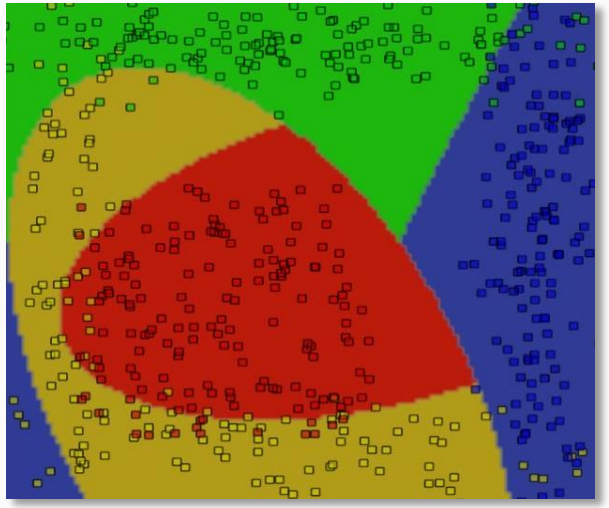


Parameters: T=200, D=3, weak learner = conic, leaf model = probabilistic

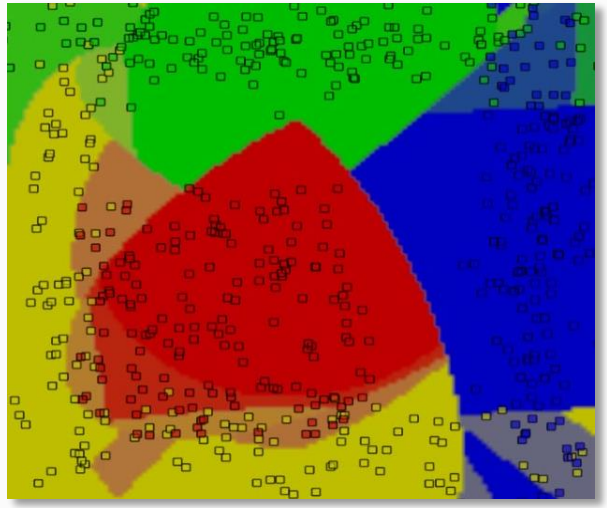
(2 videos in this page)

Classification Forest: effect of tree depth

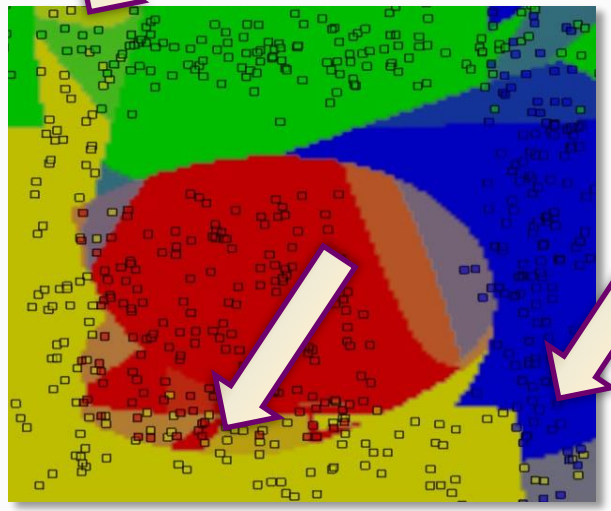




T=200, D=3, w. l. = conic



T=200, D=6, w. l. = conic



T=200, D=15, w. l. = conic

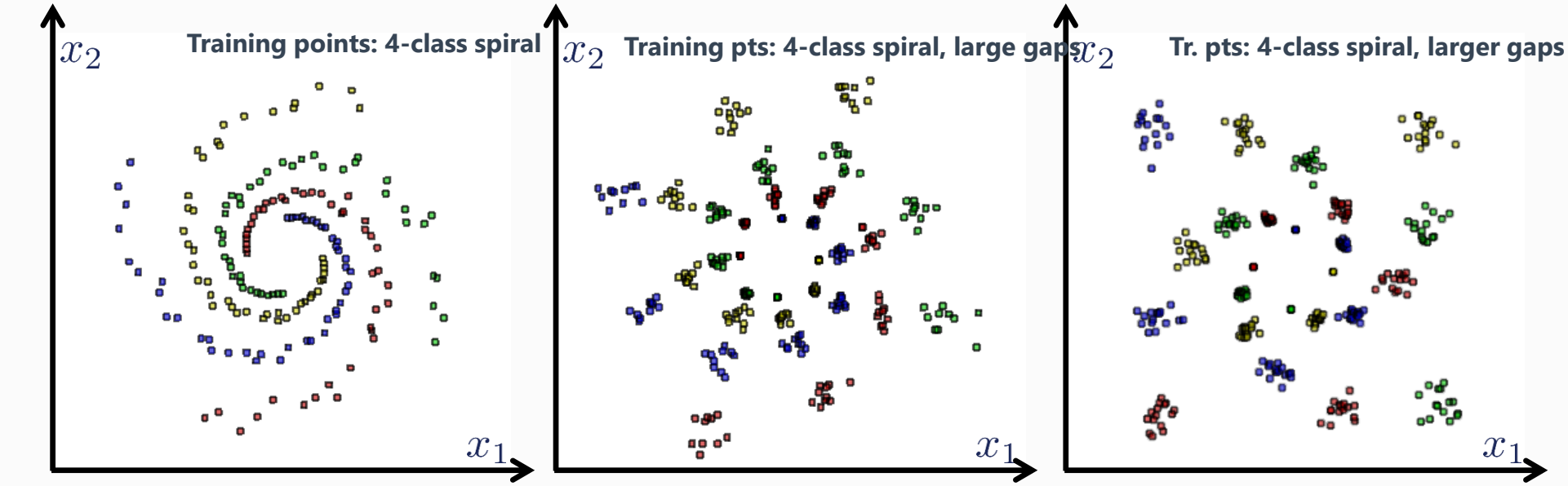
max tree depth, D

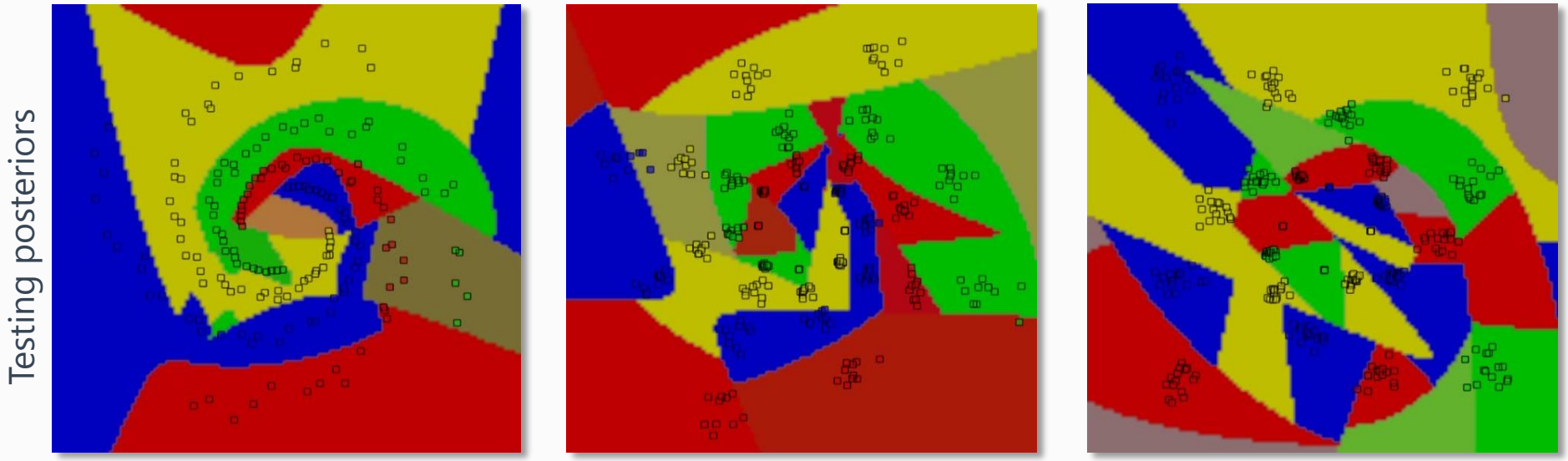
underfitting → overfitting

(3 videos in this page)

Predictor model = prob.

Classification Forest: analysing generalization





Parameters: T=200, D=13, w. l. = conic, predictor = prob.

(3 videos in this page)

Data Science

Deriving Knowledge from Data at Scale

*Feature extraction and selection are the **most important** but underrated step of machine learning. Better features are better than better algorithms…*

Data Science

Deriving Knowledge from Data at Scale

That's all for tonight….

Valentine N. Fontama

July 30th, 2015