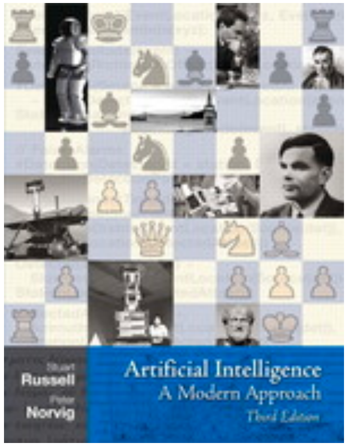
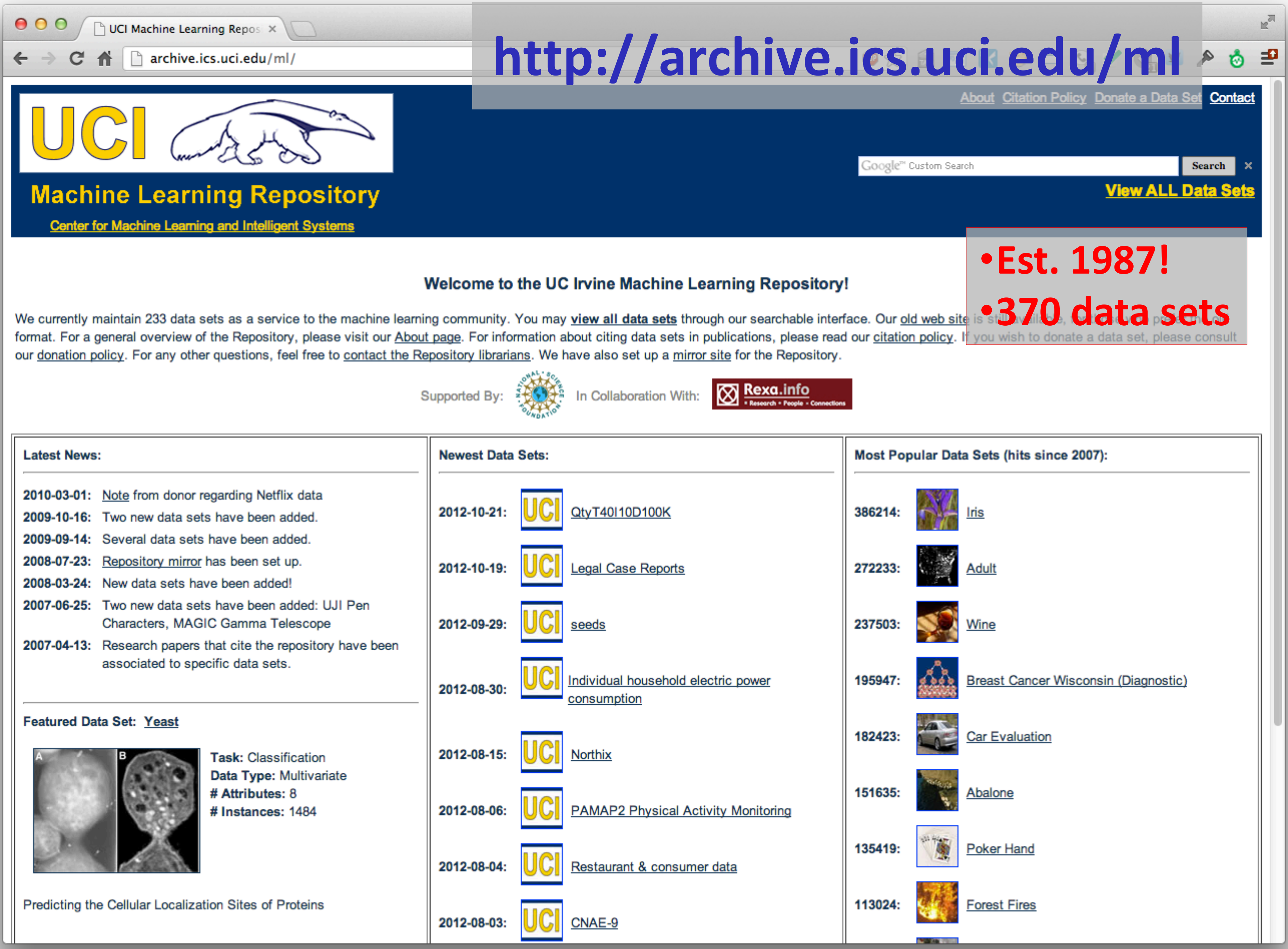
14_2_dt_examples

Decision Trees in AIMA, WEKA, and SCIKIT-LEARN



http://archive.ics.uci.edu/ml

- Est. 1987!
- 370 data sets


UCI Machine Learning Repos ×
archive.ics.uci.edu/ml/
About Citation Policy Donate a Data Set Contact
UCI
Machine Learning Repository
Center for Machine Learning and Intelligent Systems
Google™ Custom Search
Search
View ALL Data Sets

Welcome to the UC Irvine Machine Learning Repository!

We currently maintain 233 data sets as a service to the machine learning community. You may view all data sets through our searchable interface. Our old web site is still available, for those who prefer the old format. For a general overview of the Repository, please visit our About page. For information about citing data sets in publications, please read our citation policy. If you wish to donate a data set, please consult our donation policy. For any other questions, feel free to contact the Repository librarians. We have also set up a mirror site for the Repository.

Supported By: In Collaboration With: Rexa.info • Research • People • Connections

Latest News:

2010-03-01:	Note from donor regarding Netflix data
2009-10-16:	Two new data sets have been added.
2009-09-14:	Several data sets have been added.
2008-07-23:	Repository mirror has been set up.
2008-03-24:	New data sets have been added!
2007-06-25:	Two new data sets have been added: UJI Pen Characters, MAGIC Gamma Telescope
2007-04-13:	Research papers that cite the repository have been associated to specific data sets.

Featured Data Set: Yeast

Task: Classification
Data Type: Multivariate
# Attributes: 8
# Instances: 1484

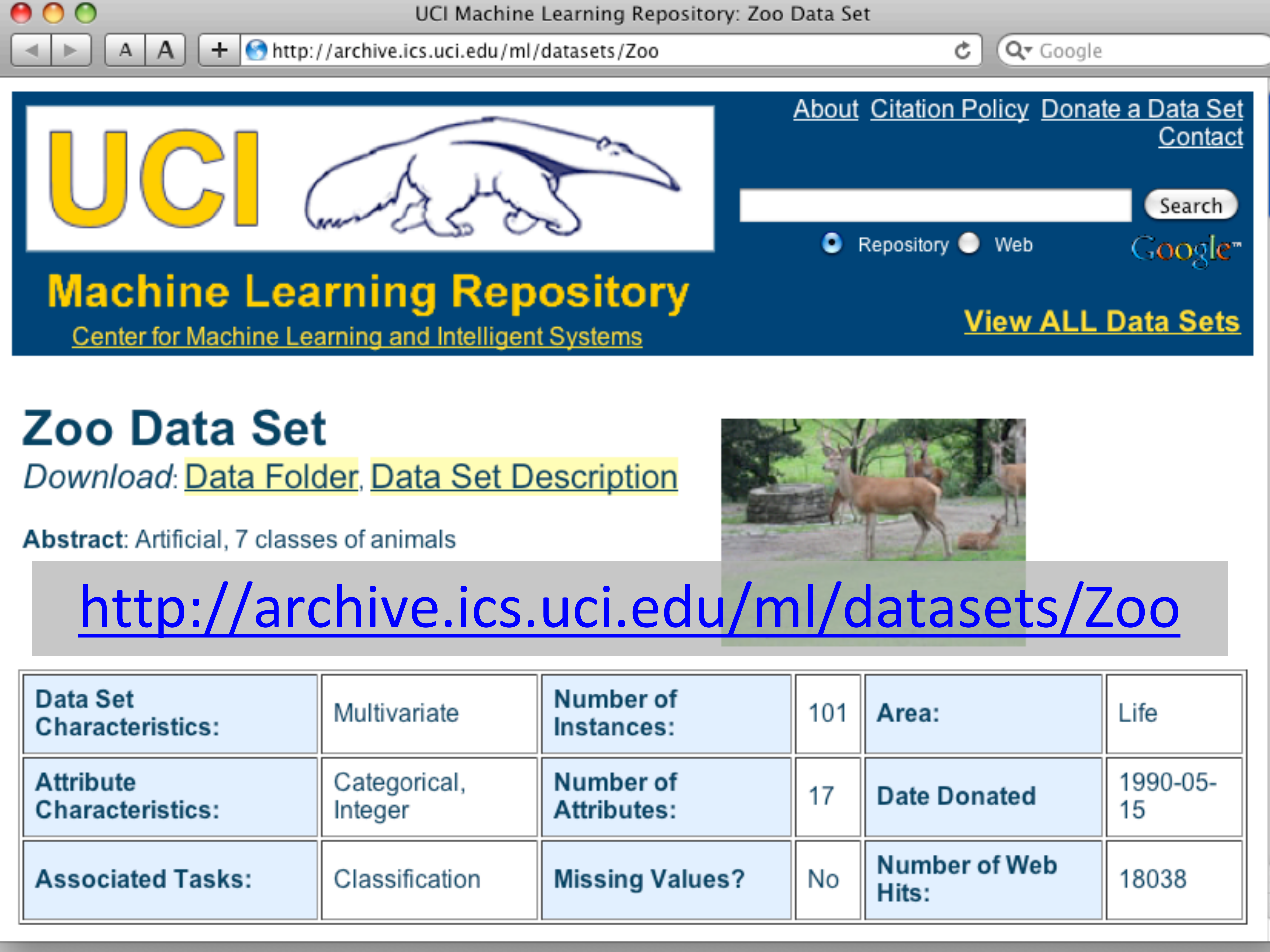
Predicting the Cellular Localization Sites of Proteins

Newest Data Sets:

2012-10-21:	QtyT40I10D100K
2012-10-19:	Legal Case Reports
2012-09-29:	seeds
2012-08-30:	Individual household electric power consumption
2012-08-15:	Northix
2012-08-06:	PAMAP2 Physical Activity Monitoring
2012-08-04:	Restaurant & consumer data
2012-08-03:	CNAE-9

Most Popular Data Sets (hits since 2007):

386214:	Iris
272233:	Adult
237503:	Wine
195947:	Breast Cancer Wisconsin (Diagnostic)
182423:	Car Evaluation
151635:	Abalone
135419:	Poker Hand
113024:	Forest Fires



UCI Machine Learning Repository: Zoo Data Set

http://archive.ics.uci.edu/ml/datasets/Zoo

UCI

Machine Learning Repository

Center for Machine Learning and Intelligent Systems

View ALL Data Sets

Zoo Data Set

Download: Data Folder, Data Set Description

Abstract: Artificial, 7 classes of animals

http://archive.ics.uci.edu/ml/datasets/Zoo

Data Set Characteristics:	Multivariate	**Number of Instances:**	101	**Area:**	Life
Attribute Characteristics:	Categorical, Integer	**Number of Attributes:**	17	**Date Donated**	1990-05-15
Associated Tasks:	Classification	**Missing Values?**	No	**Number of Web Hits:**	18038

1) animal name: string
2) hair: Boolean
3) feathers: Boolean
4) eggs: Boolean
5) milk: Boolean
6) airborne: Boolean
7) aquatic: Boolean
8) predator: Boolean
9) toothed: Boolean
10) backbone: Boolean
11) breathes: Boolean
12) venomous: Boolean
13) fins: Boolean
14) legs: {0,2,4,5,6,8}
15) tail: Boolean
16) domestic: Boolean
17) catsize: Boolean
18) type: {mammal, fish, bird, shellfish, insect, reptile, amphibian}

Zoo training data

category label

101 Instances

```
aardvark,1,0,0,1,0,0,1,1,1,1,0,0,4,0,0,1,mammal
antelope,1,0,0,1,0,0,0,1,1,1,0,0,4,1,0,1,mammal
bass,0,0,1,0,0,1,1,1,1,0,0,1,0,1,0,0,fish
bear,1,0,0,1,0,0,1,1,1,1,0,0,4,0,0,1,mammal
boar,1,0,0,1,0,0,1,1,1,1,0,0,4,1,0,1,mammal
buffalo,1,0,0,1,0,0,0,1,1,1,0,0,4,1,0,1,mammal
calf,1,0,0,1,0,0,0,1,1,1,0,0,4,1,1,1,mammal
carp,0,0,1,0,0,1,0,1,1,0,0,1,0,1,1,0,fish
catfish,0,0,1,0,0,1,1,1,1,0,0,1,0,1,0,0,fish
cavy,1,0,0,1,0,0,0,1,1,1,0,0,4,0,1,0,mammal
cheetah,1,0,0,1,0,0,1,1,1,1,0,0,4,1,0,1,mammal
chicken,0,1,1,0,1,0,0,0,1,1,0,0,2,1,1,0,bird
chub,0,0,1,0,0,1,1,1,1,0,0,1,0,1,0,0,fish
clam,0,0,1,0,0,0,1,0,0,0,0,0,0,0,0,0,shellfish
crab,0,0,1,0,0,1,1,0,0,0,0,0,4,0,0,0,shellfish
…
```

Zoo example

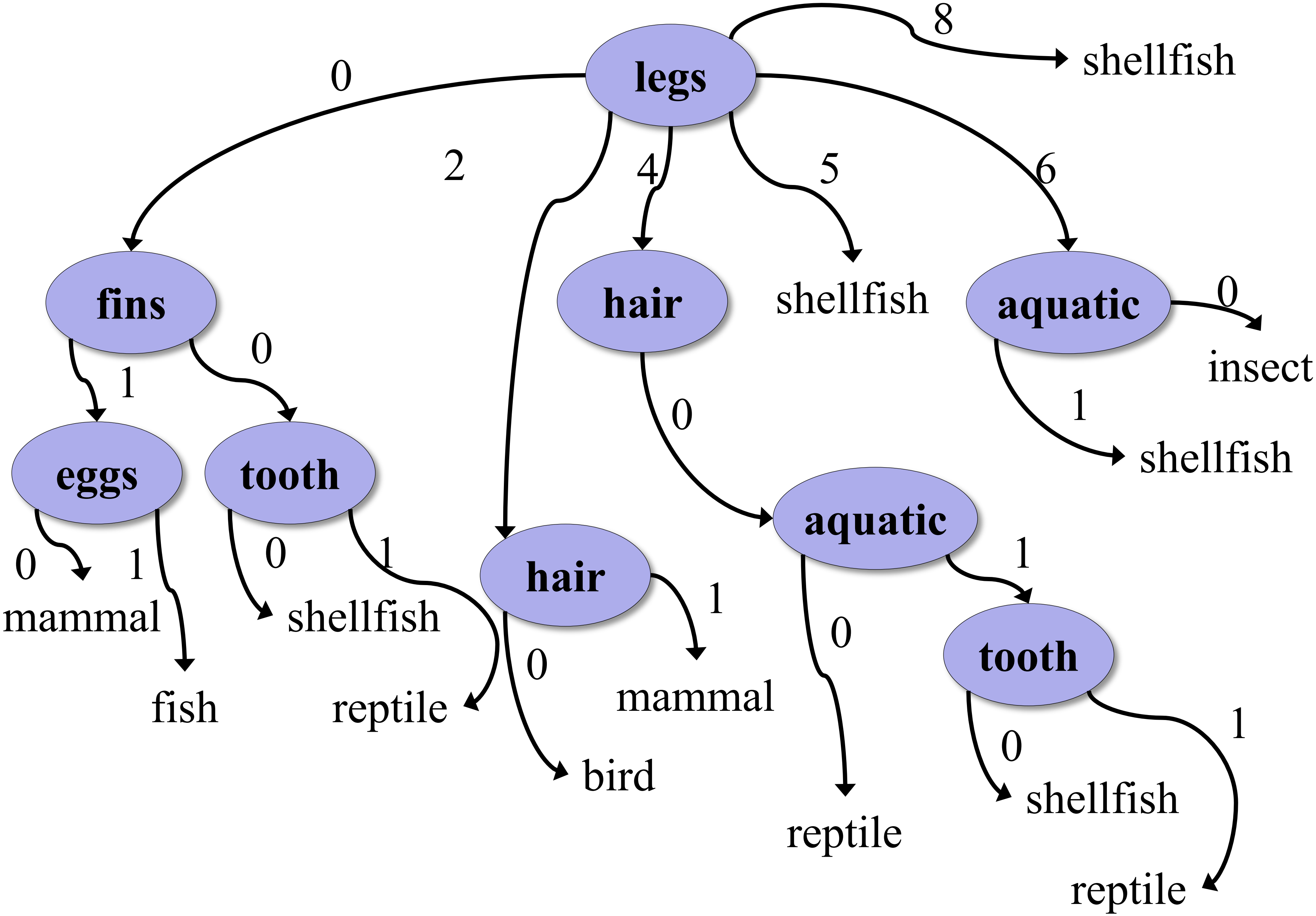
```
aima-python> python
>>> from learning import *
>>> zoo
<DataSet(zoo): 101 examples, 18 attributes>
>>> dt = DecisionTreeLearner()
>>> dt.train(zoo)
>>> dt.predict(['shark',0,0,1,0,0,1,1,1,1,0,0,1,0,1,0,0]) #eggs=1
'fish'
>>> dt.predict(['shark',0,0,0,0,0,1,1,1,1,0,0,1,0,1,0,0]) #eggs=0
'mammal’
```

Zoo example

```
>> dt.dt
DecisionTree(13, 'legs', {0: DecisionTree(12, 'fins', {0:
DecisionTree(8, 'toothed', {0: 'shellfish', 1: 'reptile'}), 1:
DecisionTree(3, 'eggs', {0: 'mammal', 1: 'fish'})}), 2:
DecisionTree(1, 'hair', {0: 'bird', 1: 'mammal'}), 4:
DecisionTree(1, 'hair', {0: DecisionTree(6, 'aquatic', {0:
'reptile', 1: DecisionTree(8, 'toothed', {0: 'shellfish', 1:
'amphibian'})}), 1: 'mammal'}), 5: 'shellfish', 6:
DecisionTree(6, 'aquatic', {0: 'insect', 1: 'shellfish'}), 8:
'shellfish'})
```

Zoo example

```
>>> dt.dt.display()
Test legs
 legs = 0 ==> Test fins
    fins = 0 ==> Test toothed
       toothed = 0 ==> RESULT =  shellfish
       toothed = 1 ==> RESULT =  reptile
    fins = 1 ==> Test eggs
       eggs = 0 ==> RESULT =  mammal
       eggs = 1 ==> RESULT =  fish
 legs = 2 ==> Test hair
    hair = 0 ==> RESULT =  bird
    hair = 1 ==> RESULT =  mammal
 legs = 4 ==> Test hair
    hair = 0 ==> Test aquatic
       aquatic = 0 ==> RESULT =  reptile
       aquatic = 1 ==> Test toothed
          toothed = 0 ==> RESULT =  shellfish
          toothed = 1 ==> RESULT =  amphibian
    hair = 1 ==> RESULT =  mammal
 legs = 5 ==> RESULT =  shellfish
 legs = 6 ==> Test aquatic
    aquatic = 0 ==> RESULT =  insect
    aquatic = 1 ==> RESULT =  shellfish
 legs = 8 ==> RESULT =  shellfish
```


legs
0
2
4
5
6
8
shellfish
fins
hair
shellfish
aquatic
0
insect
1
shellfish
1
0
eggs
tooth
0
mammal
1
fish
0
shellfish
1
reptile
hair
0
bird
1
mammal
0
aquatic
0
reptile
1
tooth
0
shellfish
1
reptile

Zoo example

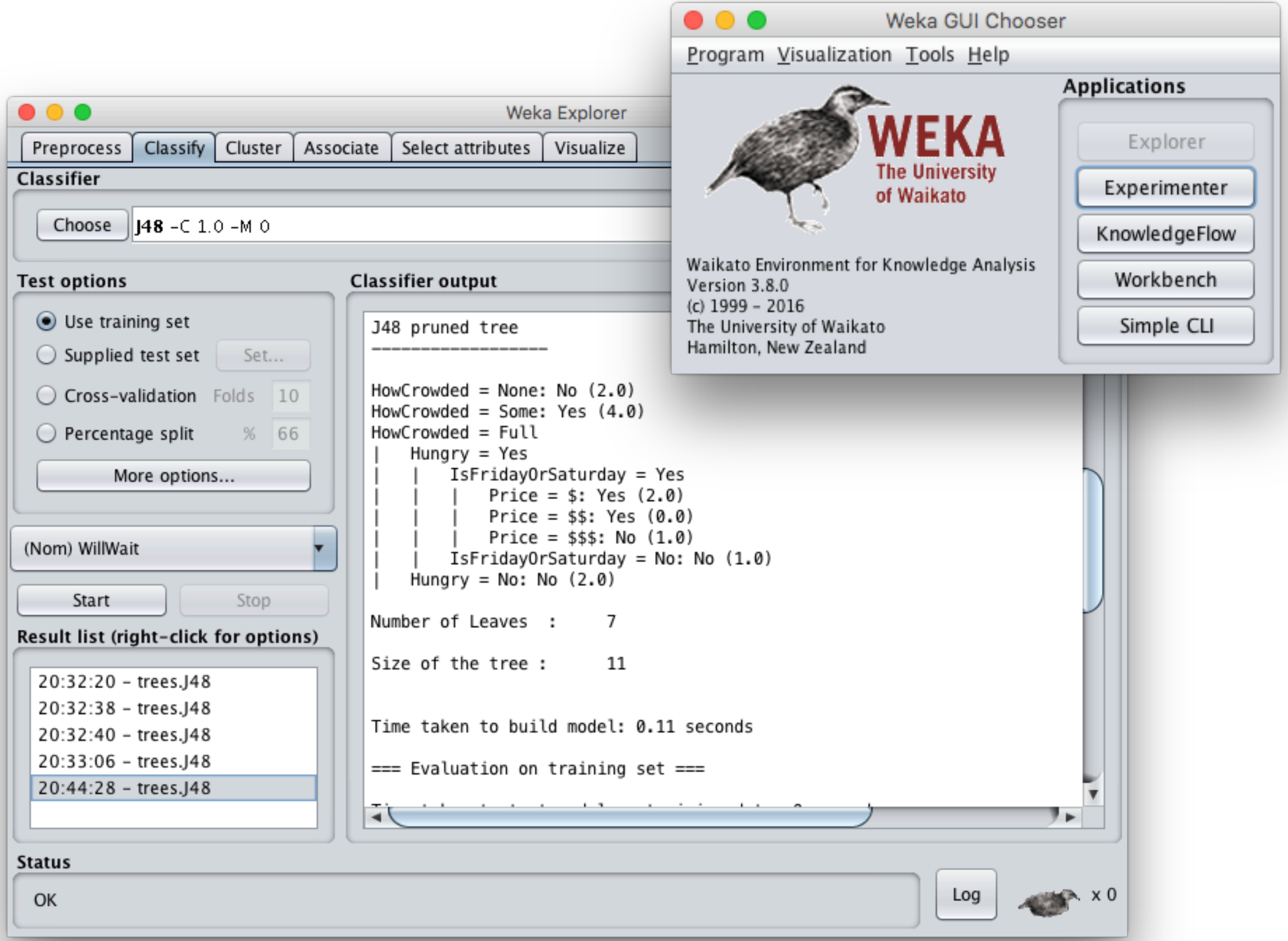
```
>>> dt.dt.display()
Test legs
 legs = 0 ==> Test fins
     fins = 0 ==> Test toothed
         toothed = 0 ==> RESULT =  shellfish
         toothed = 1 ==> RESULT =  reptile
     fins = 1 ==> Test milk
         milk = 0 ==> RESULT =  fish
         milk = 1 ==> RESULT =  mammal
 legs = 2 ==> Test hair
     hair = 0 ==> RESULT =  bird
     hair = 1 ==> RESULT =  mammal
 legs = 4 ==> Test hair
     hair = 0 ==> Test aquatic
         aquatic = 0 ==> RESULT =  reptile
         aquatic = 1 ==> Test toothed
             toothed = 0 ==> RESULT =  shellfish
             toothed = 1 ==> RESULT =  amphibian
     hair = 1 ==> RESULT =  mammal
 legs = 5 ==> RESULT =  shellfish
 legs = 6 ==> Test aquatic
     aquatic = 0 ==> RESULT =  insect
     aquatic = 1 ==> RESULT =  shellfish
 legs = 8 ==> RESULT =  shellfish
```

After adding the shark example to the training data & retraining

Weka

Waikato Environment for Knowledge Analysis
Version 3.7.4
(c) 1999 - 2011
The University of Waikato
Hamilton, New Zealand

- Open-source Java machine learning tool
- http://www.cs.waikato.ac.nz/ml/weka/
- Implements many classifiers & ML algorithms
- Uses common data representation format; easy to try different ML algorithms and compare results
- Comprehensive set of data pre-processing tools and evaluation methods
- Three modes of operation: GUI, command line, Java API



Waikato Environment for Knowledge Analysis
Version 3.8.0
(c) 1999 - 2016
The University of Waikato
Hamilton, New Zealand

Classifier: J48 -C 1.0 -M 0

Test options:

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

(Nom) WillWait

Result list (right-click for options):

- 20:32:20 - trees.J48
- 20:32:38 - trees.J48
- 20:32:40 - trees.J48
- 20:33:06 - trees.J48
- 20:44:28 - trees.J48

Classifier output:

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

HowCrowded = None: No (2.0)
HowCrowded = Some: Yes (4.0)
HowCrowded = Full
|   Hungry = Yes
|   |   IsFridayOrSaturday = Yes
|   |   |   Price = $: Yes (2.0)
|   |   |   Price = $$: Yes (0.0)
|   |   |   Price = $$$: No (1.0)
|   |   IsFridayOrSaturday = No: No (1.0)
|   Hungry = No: No (2.0)

Number of Leaves  :     7

Size of the tree :      11


Time taken to build model: 0.11 seconds

=== Evaluation on training set ===
```

Status: OK

x 0

Common .arff* data format

```
% Simplified data for predicting heart disease with just six variables
% Comments begin with a % allowed at the top
@relation heart-disease-simplified

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

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

***age** is a numeric attribute*

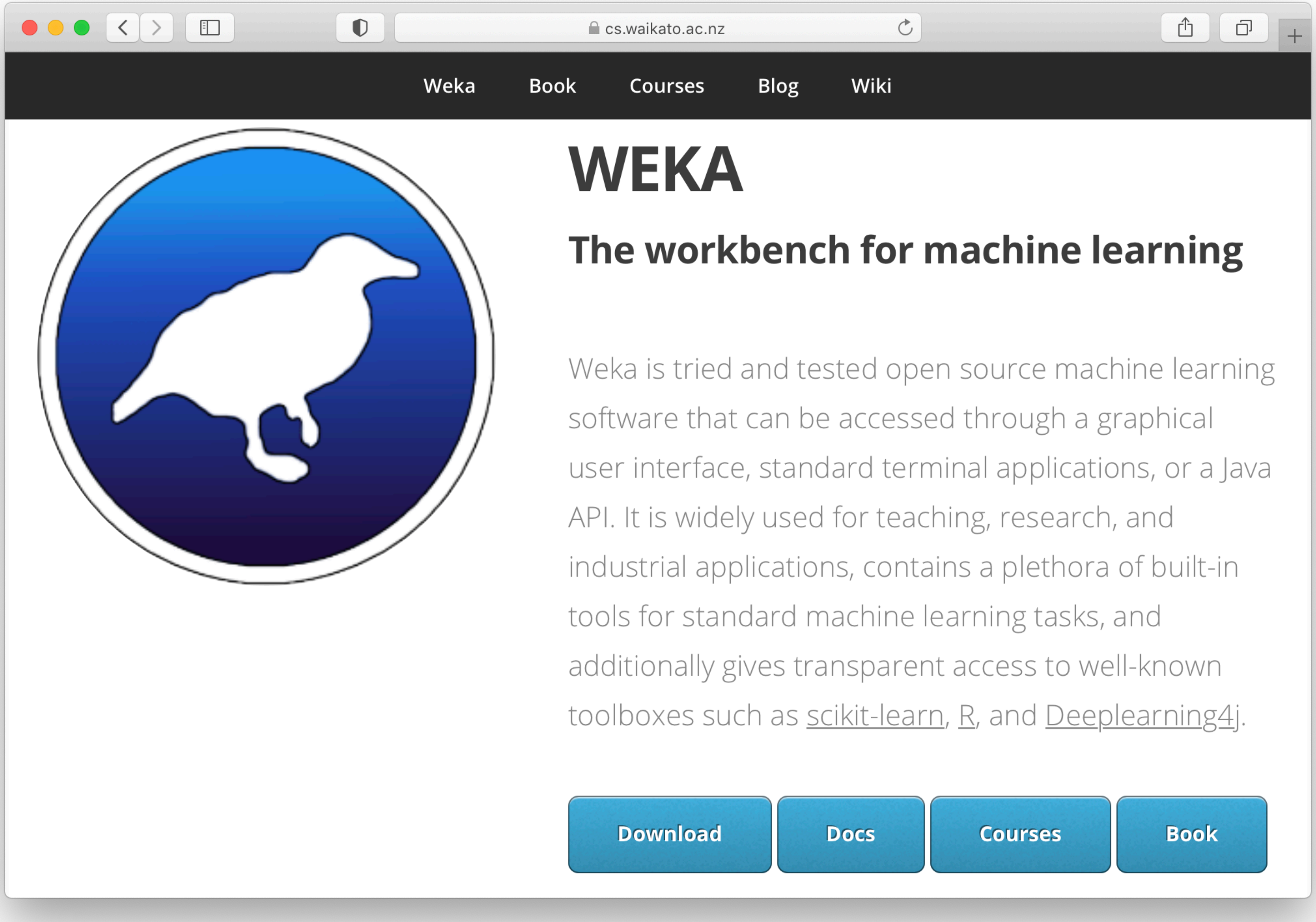
***sex** is a nominal attribute*

***class** is target variable*

Training data

*ARFF = Attribute-Relation File Format

Weka demo



cs.waikato.ac.nz

WEKA

The workbench for machine learning

Weka is tried and tested open source machine learning software that can be accessed through a graphical user interface, standard terminal applications, or a Java API. It is widely used for teaching, research, and industrial applications, contains a plethora of built-in tools for standard machine learning tasks, and additionally gives transparent access to well-known toolboxes such as scikit-learn, R, and Deeplearning4j.

https://cs.waikato.ac.nz/ml/weka/

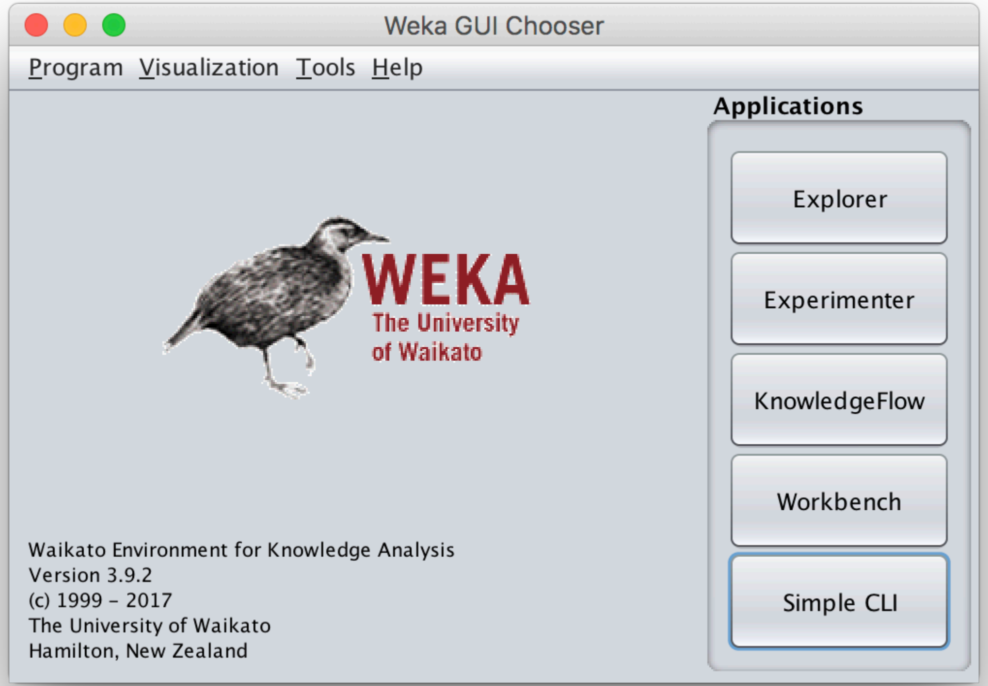
Install Weka

- Download and install Weka
- cd to your weka directory
- Invoke the GUI interface or call components from the command line
  - You may want to set environment variables (e.g., CLASSPATH) or aliases (e.g., weka)

Getting your data ready

- Our class code repo's ML directory has several data files for the restaurant example
    1. **restaurant.csv:** original data in simple text format
    2. **restaurant.arff:** data put in Weka's **arff** format
    3. **restaurant_test.arff:** more data for test/evaluation
    4. **restaurant_predict.arff:** new data we want predictions for using a saved model
- #1 is the raw training data we're given
- We'll train and save a model with #2
- Test it with #3
- Predict target on new data with #4

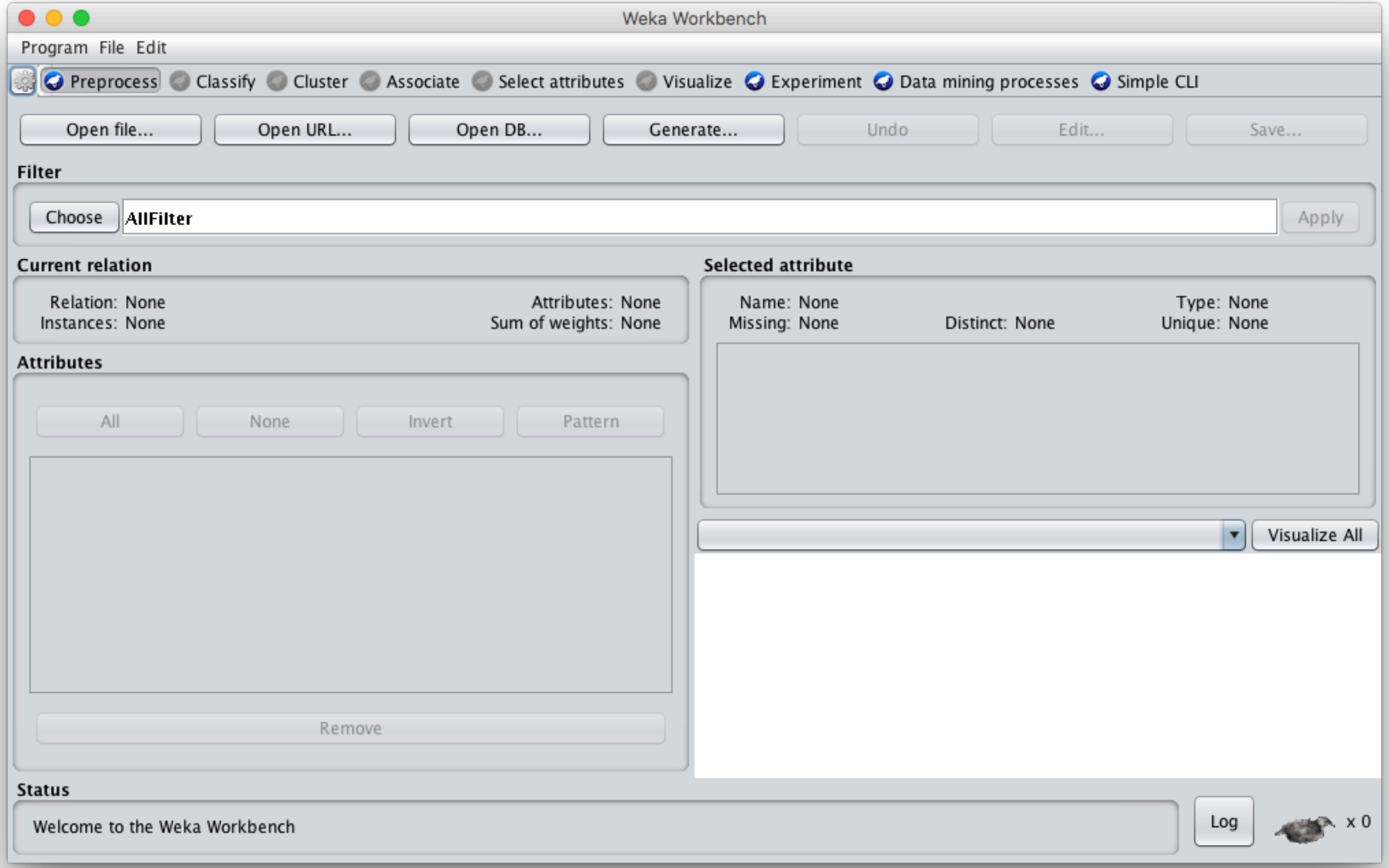
Open Weka app



Waikato Environment for Knowledge Analysis
Version 3.9.2
(c) 1999 - 2017
The University of Waikato
Hamilton, New Zealand

- cd /Applications/weka
- java -jar weka.jar
- Apps optimized for different tasks
- Start with Explorer

Explorer Interface



Filter

AllFilter

Current relation

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

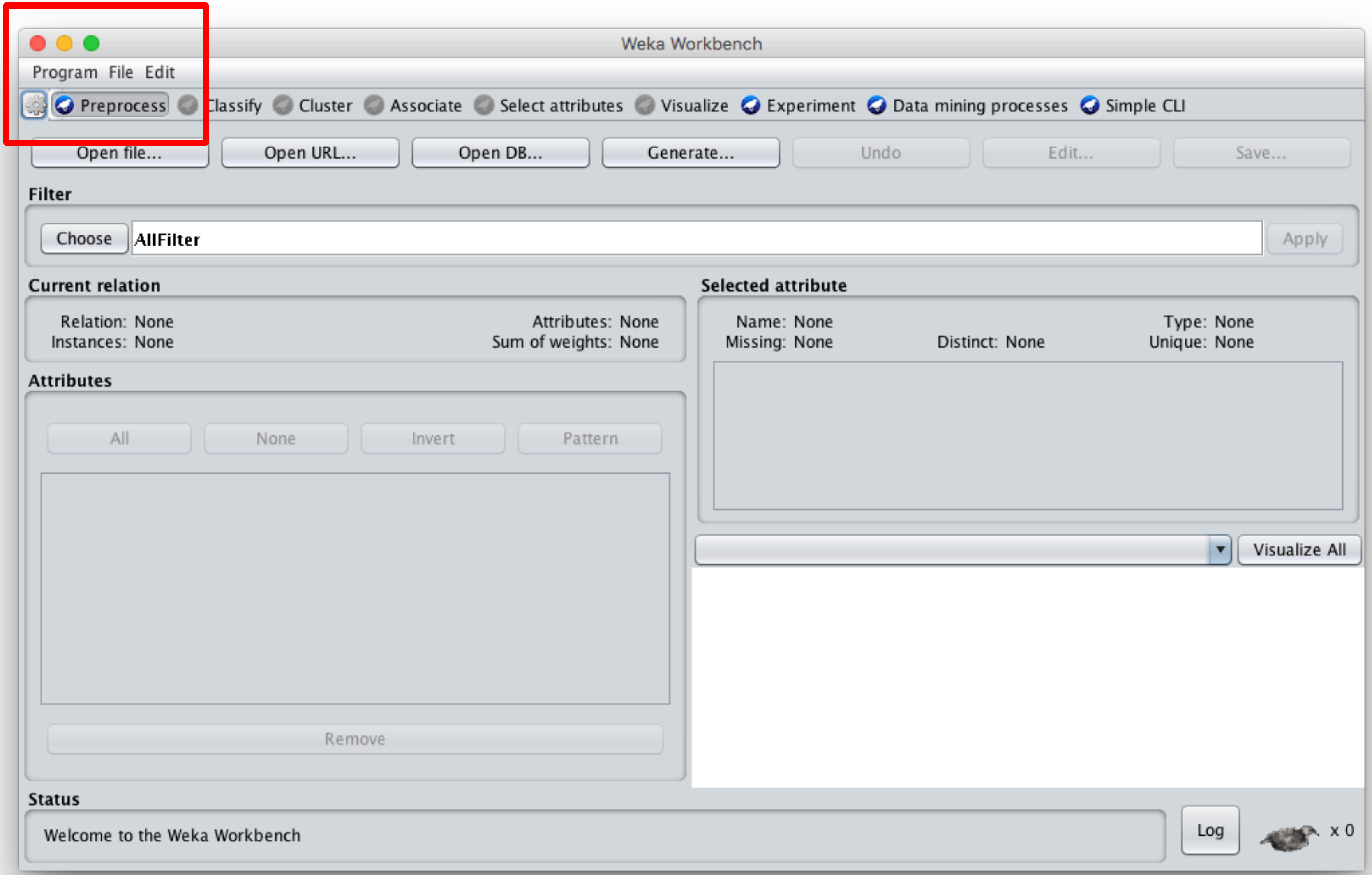
Selected attribute

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

Status

Welcome to the Weka Workbench

Starts with Data Preprocessing; open file to load data



Filter

AllFilter

Current relation

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

Attributes

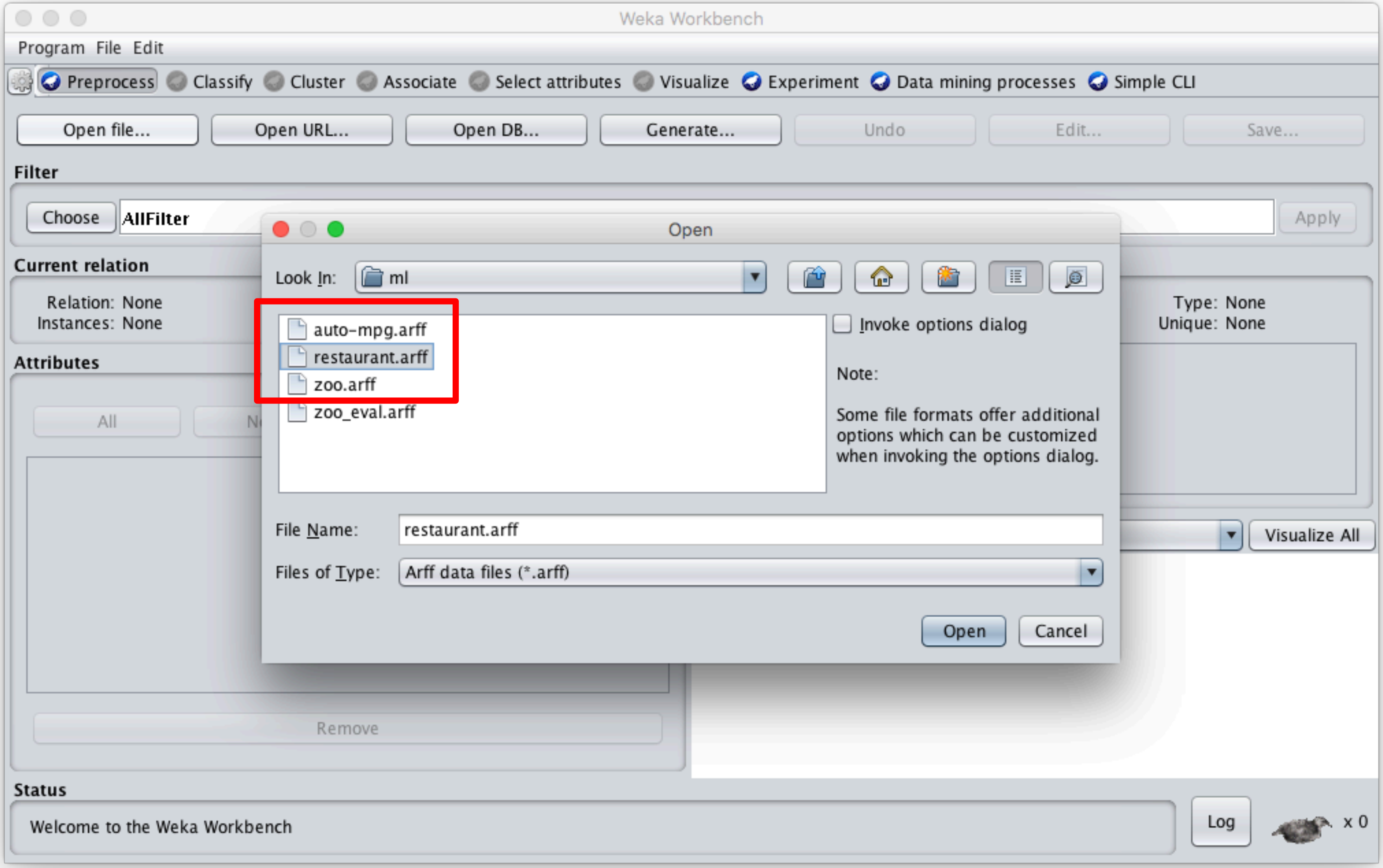
Selected attribute

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

Status

Welcome to the Weka Workbench

Load restaurant.arff training data



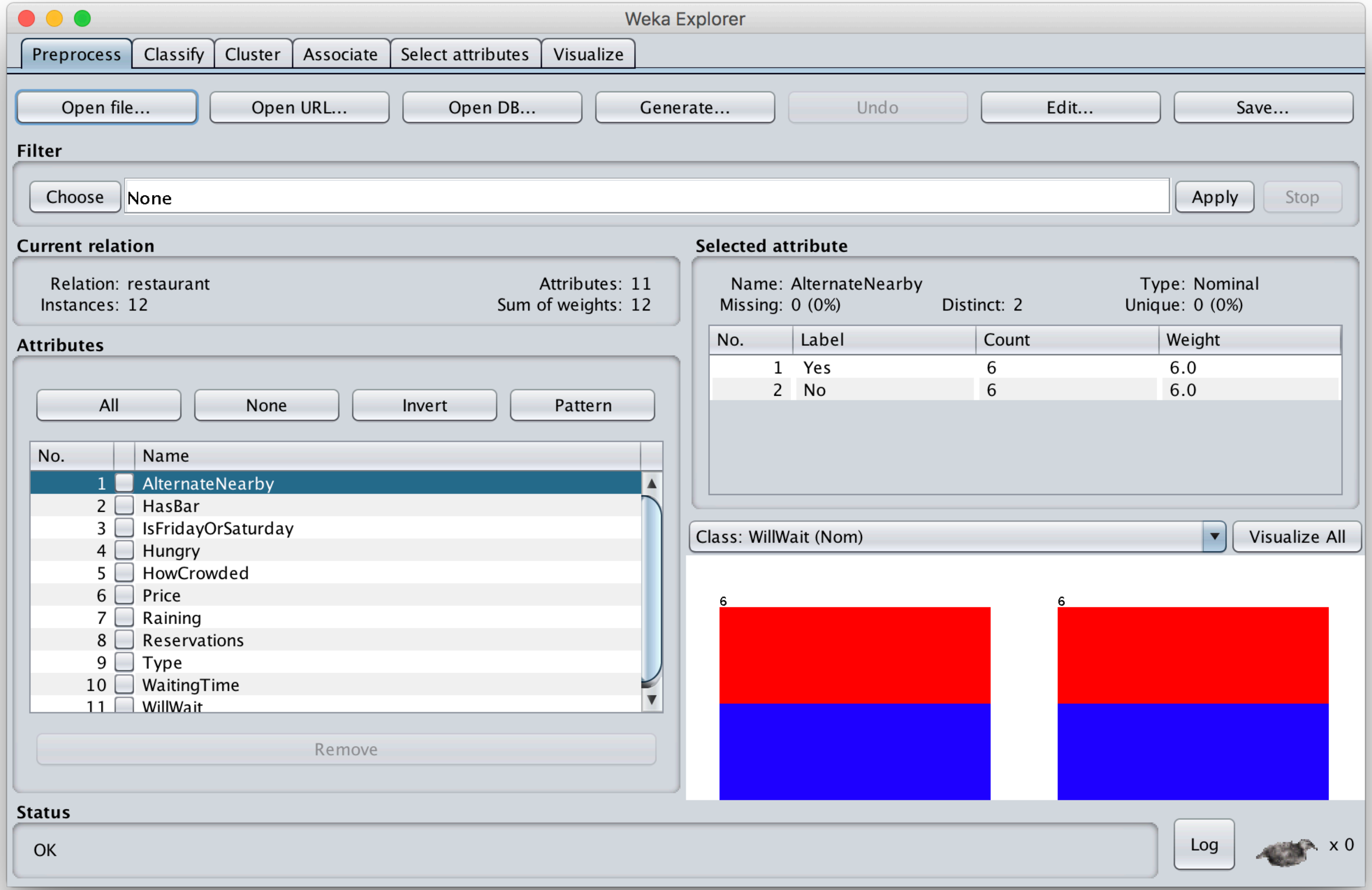
Note:

Some file formats offer additional options which can be customized when invoking the options dialog.

File Name: restaurant.arff

Files of Type: Arff data files (*.arff)

We can inspect/remove features



Filter

None

Current relation

Relation: restaurant
Instances: 12
Attributes: 11
Sum of weights: 12

Attributes

No.		Name
1	□	AlternateNearby
2	□	HasBar
3	□	IsFridayOrSaturday
4	□	Hungry
5	□	HowCrowded
6	□	Price
7	□	Raining
8	□	Reservations
9	□	Type
10	□	WaitingTime
11	□	WillWait

Selected attribute

Name: AlternateNearby
Type: Nominal
Missing: 0 (0%)
Distinct: 2
Unique: 0 (0%)

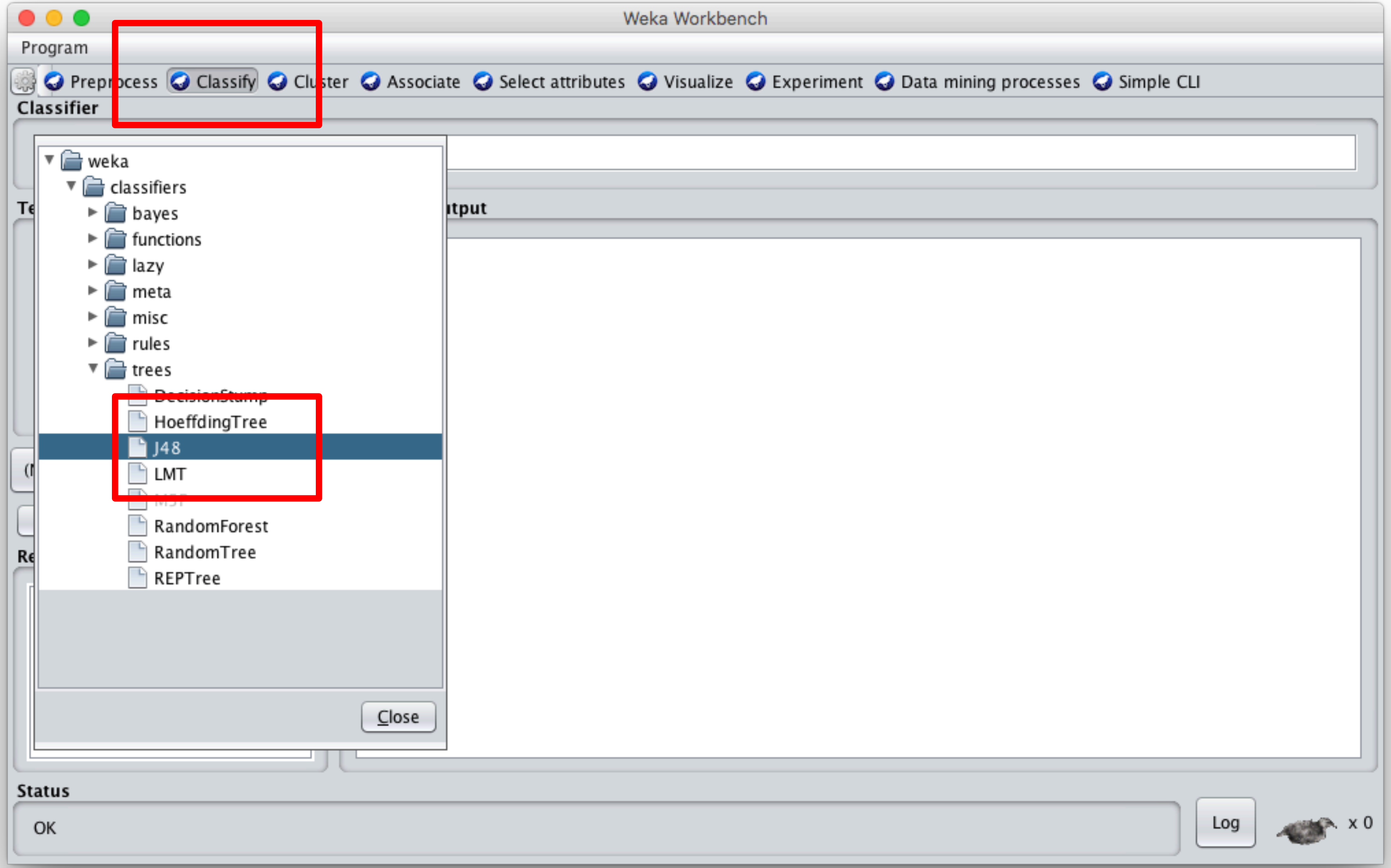
No.	Label	Count	Weight
1	Yes	6	6.0
2	No	6	6.0

Class: WillWait (Nom)

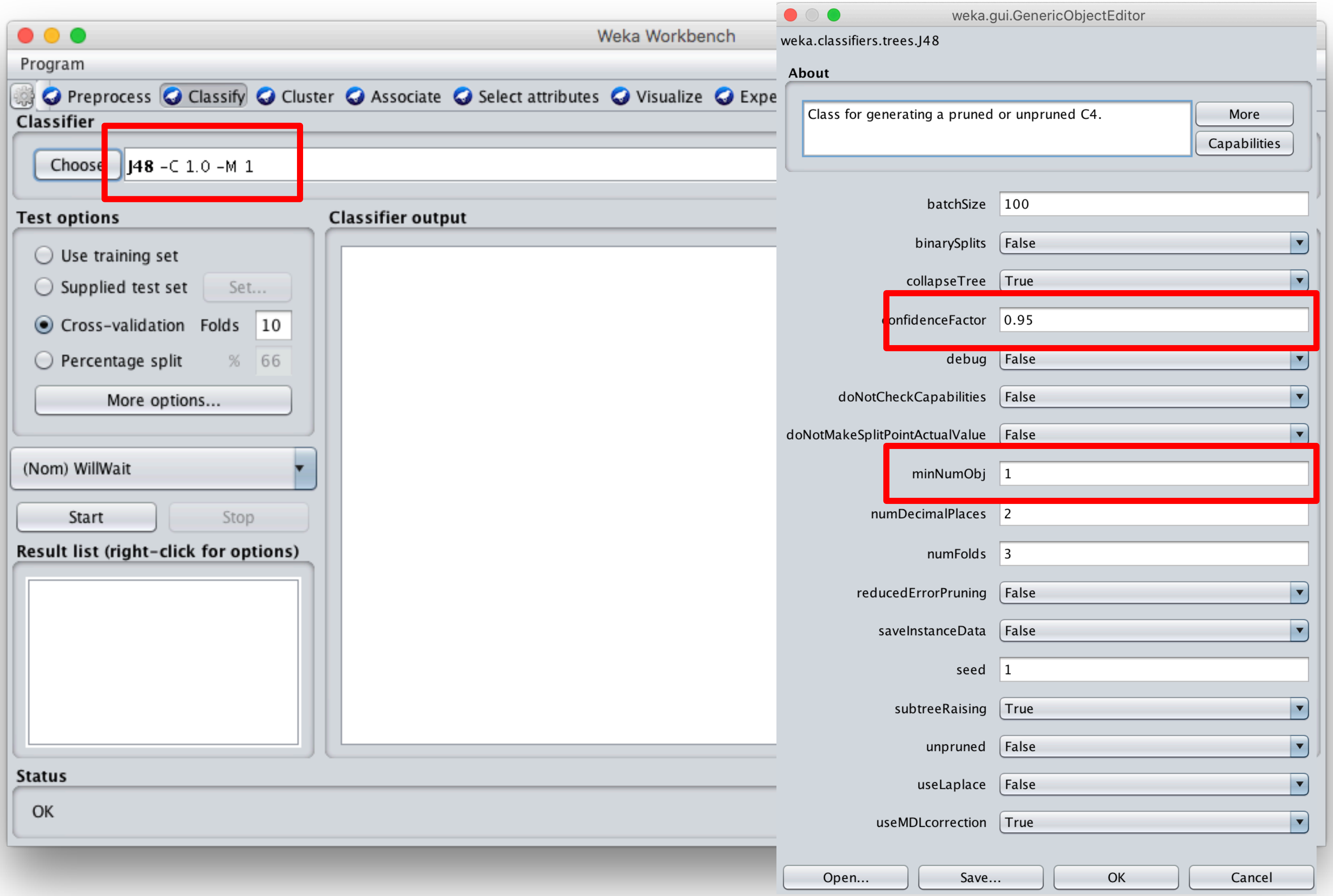
Status

OK

Select: classify > choose > trees > J48



Adjust parameters



Classifier

J48 -C 1.0 -M 1

Test options

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

(Nom) WillWait

Status

OK

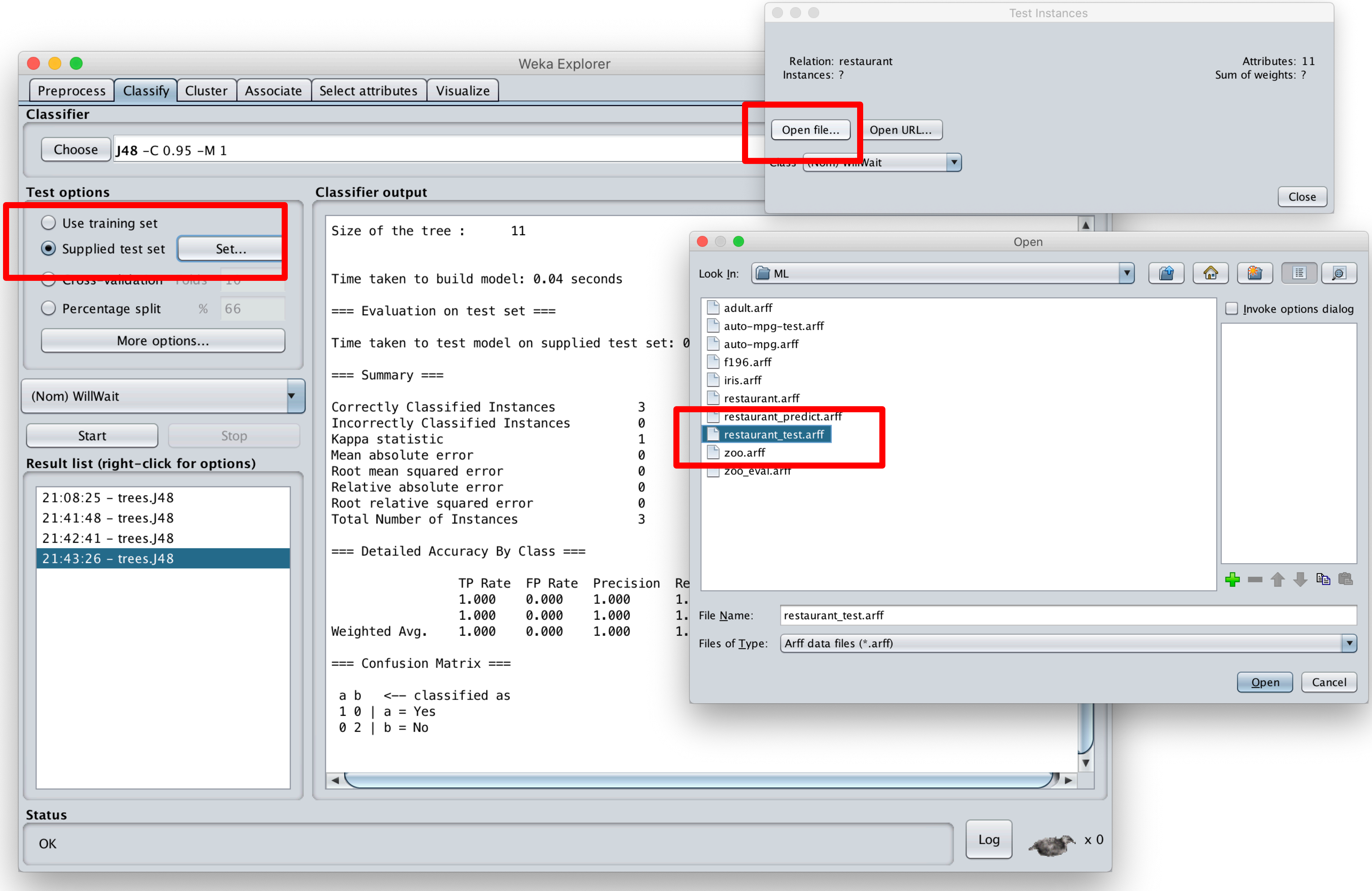
weka.classifiers.trees.J48

About

Class for generating a pruned or unpruned C4.

batchSize	100
binarySplits	False
collapseTree	True
confidenceFactor	0.95
debug	False
doNotCheckCapabilities	False
doNotMakeSplitPointActualValue	False
minNumObj	1
numDecimalPlaces	2
numFolds	3
reducedErrorPruning	False
saveInstanceData	False
seed	1
subtreeRaising	True
unpruned	False
useLaplace	False
useMDLcorrection	True

Select the testing procedure



Classifier: J48 -C 0.95 -M 1

Test options:

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

(Nom) WillWait

Result list:

- 21:08:25 - trees.J48
- 21:41:48 - trees.J48
- 21:42:41 - trees.J48
- 21:43:26 - trees.J48

```
Size of the tree :      11


Time taken to build model: 0.04 seconds

=== Evaluation on test set ===

Time taken to test model on supplied test set: 0

=== Summary ===

Correctly Classified Instances           3
Incorrectly Classified Instances         0
Kappa statistic                          1
Mean absolute error                      0
Root mean squared error                  0
Relative absolute error                  0
Root relative squared error              0
Total Number of Instances                3

=== Detailed Accuracy By Class ===

                 TP Rate  FP Rate  Precision  Re
                 1.000    0.000    1.000      1.
                 1.000    0.000    1.000      1.
Weighted Avg.    1.000    0.000    1.000      1.

=== Confusion Matrix ===

 a b   <-- classified as
 1 0 | a = Yes
 0 2 | b = No
```

Status: OK

Test Instances:

Relation: restaurant
Instances: ?
Attributes: 11
Sum of weights: ?
Class (Nom) WillWait

Open:

Look In: ML

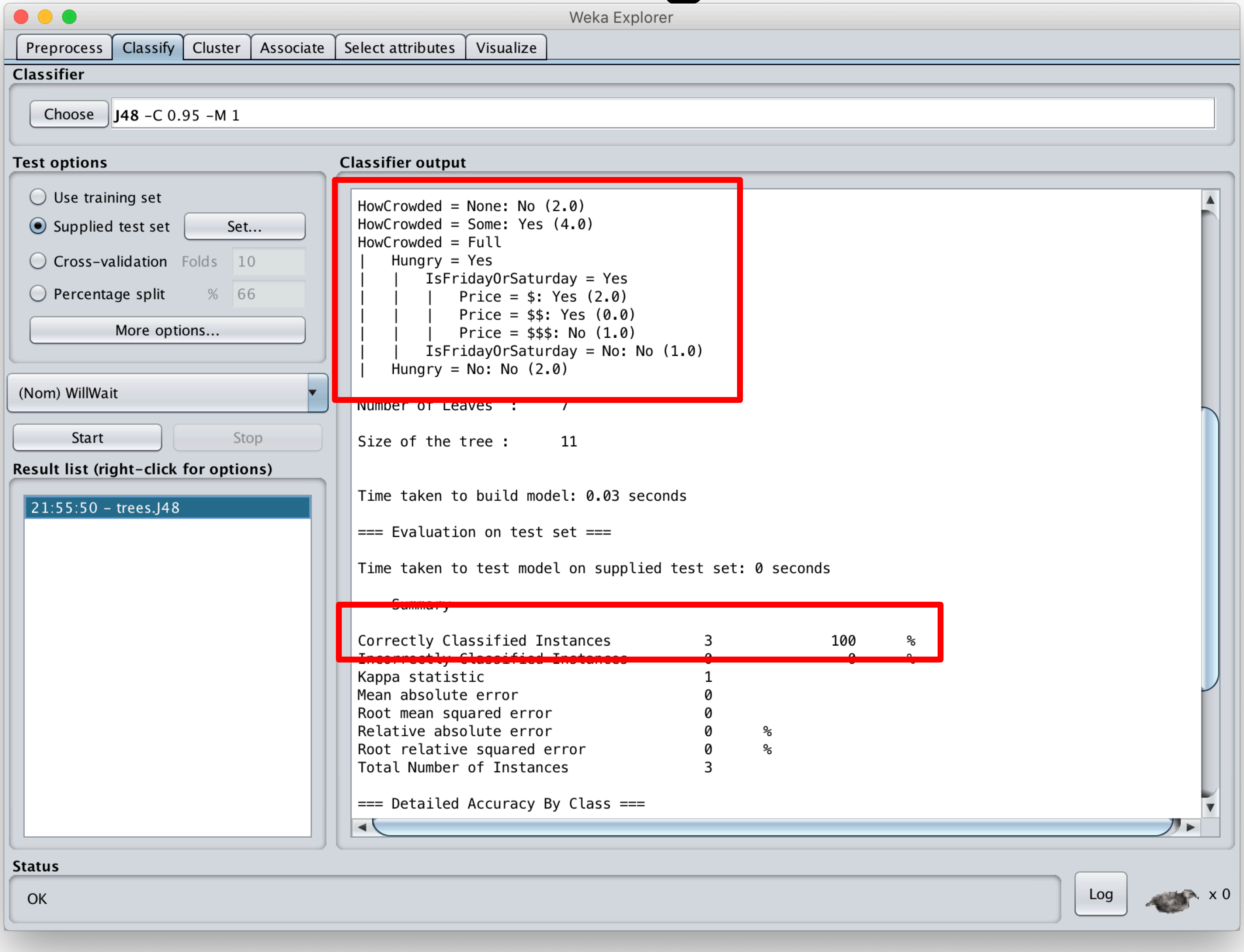
- adult.arff
- auto-mpg-test.arff
- auto-mpg.arff
- f196.arff
- iris.arff
- restaurant.arff
- restaurant_predict.arff
- restaurant_test.arff
- zoo.arff
- zoo_eval.arff

□ Invoke options dialog

File Name: restaurant_test.arff

Files of Type: Arff data files (*.arff)

See training results



Classifier

Choose J48 -C 0.95 -M 1

Test options

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

(Nom) WillWait

Result list (right-click for options)

21:55:50 - trees.J48

Status

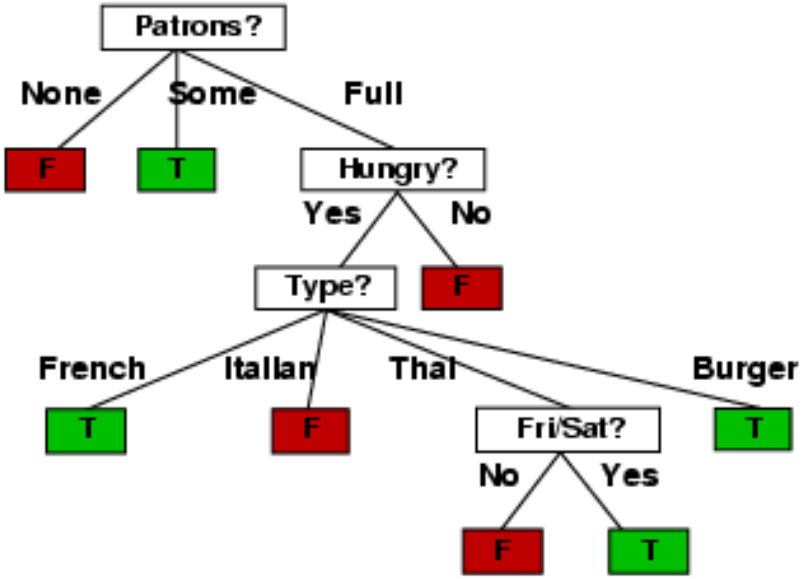
OK

Classifier output

```
HowCrowded = None: No (2.0)
HowCrowded = Some: Yes (4.0)
HowCrowded = Full
|   Hungry = Yes
|   |   IsFridayOrSaturday = Yes
|   |   |   Price = $: Yes (2.0)
|   |   |   Price = $$: Yes (0.0)
|   |   |   Price = $$$: No (1.0)
|   |   IsFridayOrSaturday = No: No (1.0)
|   Hungry = No: No (2.0)

Number of Leaves  :      7

Size of the tree :      11


Time taken to build model: 0.03 seconds

=== Evaluation on test set ===

Time taken to test model on supplied test set: 0 seconds

=== Summary ===

Correctly Classified Instances           3              100      %
Incorrectly Classified Instances         0                0      %
Kappa statistic                          1
Mean absolute error                      0
Root mean squared error                  0
Relative absolute error                  0      %
Root relative squared error              0      %
Total Number of Instances                3

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

Compare results

```
HowCrowded = None: No (2.0)
HowCrowded = Some: Yes (4.0)
HowCrowded = Full
|   Hungry = Yes
|   |   IsFridayOrSaturday = Yes
|   |   |   Price = $: Yes (2.0)
|   |   |   Price = $$: Yes (0.0)
|   |   |   Price = $$$: No (1.0)
|   |   IsFridayOrSaturday = No: No (1.0)
|   Hungry = No: No (2.0)
```



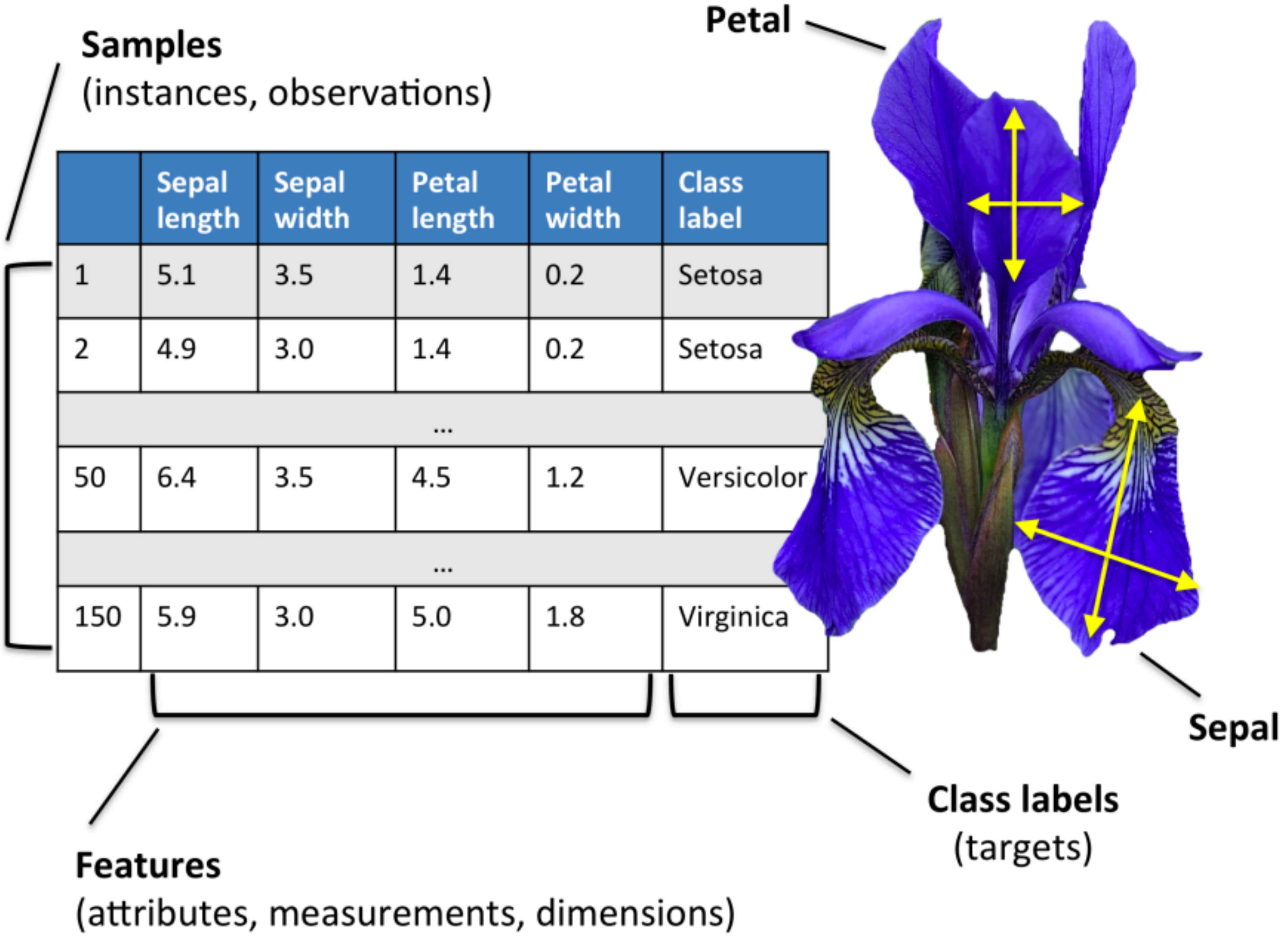
J48 pruned tree: nodes:11; leaves:7, max depth:4

ID3 tree: nodes:12; leaves:8, max depth:4

The two decision trees are equally good

scikit-learn

- Popular open source ML and data analysis tools for Python
- Built on NumPy, SciPy, and matplotlib for efficiency
- However decision tree tools are a weak area
  - E.g., data features must be numeric, so working with restaurant example requires conversion
  - Perhaps because DTs not used for large problems
- We'll look at using it to learn a DT for the classic iris flower dataset



Samples
(instances, observations)

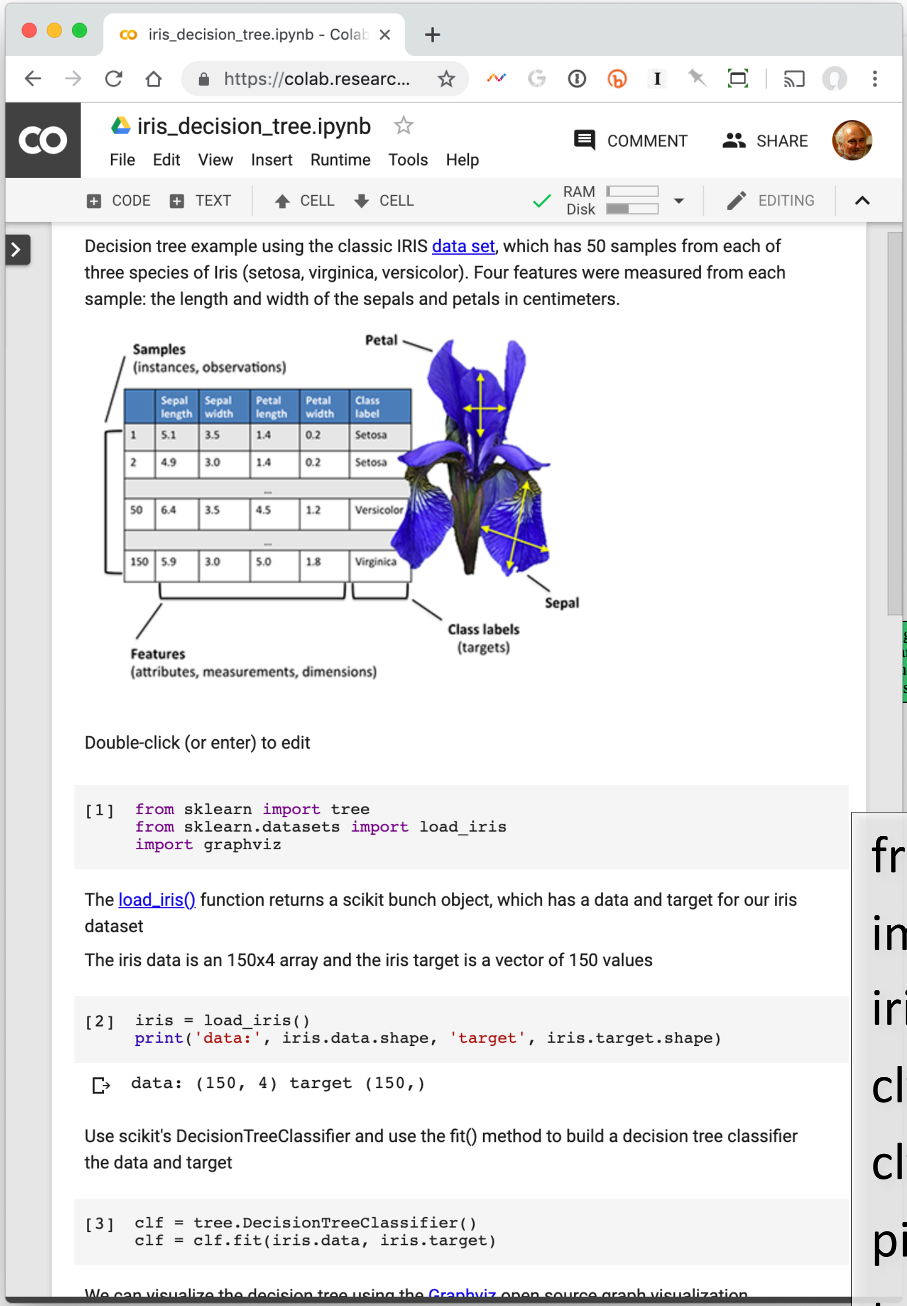
	Sepal length	Sepal width	Petal length	Petal width	Class label
1	5.1	3.5	1.4	0.2	Setosa
2	4.9	3.0	1.4	0.2	Setosa
...					
50	6.4	3.5	4.5	1.2	Versicolor
...					
150	5.9	3.0	5.0	1.8	Virginica

Features
(attributes, measurements, dimensions)

Class labels
(targets)

50 samples from each of three species of Iris (setosa, virginica, versicolor) with four data features: length and width of the sepals and petals in centimeters

Scikit DT



Decision tree example using the classic IRIS data set, which has 50 samples from each of three species of Iris (setosa, virginica, versicolor). Four features were measured from each sample: the length and width of the sepals and petals in centimeters.

	Sepal length	Sepal width	Petal length	Petal width	Class label
1	5.1	3.5	1.4	0.2	Setosa
2	4.9	3.0	1.4	0.2	Setosa
			...		
50	6.4	3.5	4.5	1.2	Versicolor
			...		
150	5.9	3.0	5.0	1.8	Virginica

Double-click (or enter) to edit

```
[1]  from sklearn import tree
     from sklearn.datasets import load_iris
     import graphviz
```

The load_iris() function returns a scikit bunch object, which has a data and target for our iris dataset

The iris data is an 150x4 array and the iris target is a vector of 150 values

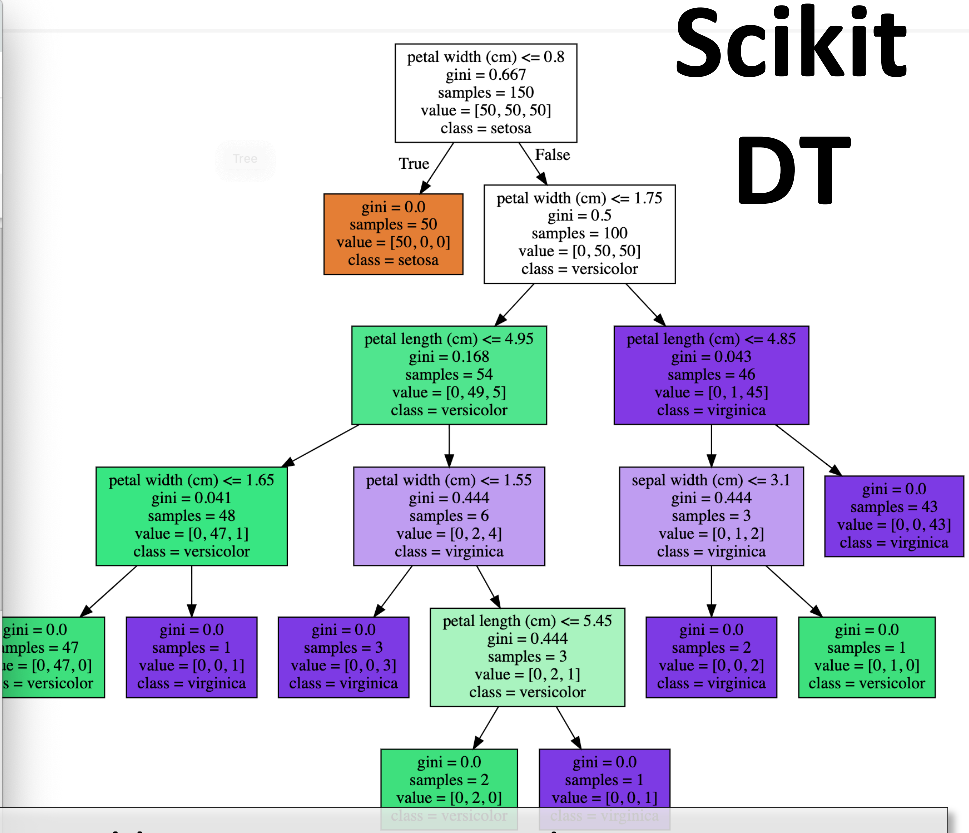
```
[2]  iris = load_iris()
     print('data:', iris.data.shape, 'target', iris.target.shape)
```

```
data: (150, 4) target (150,)
```

Use scikit's DecisionTreeClassifier and use the fit() method to build a decision tree classifier the data and target

```
[3]  clf = tree.DecisionTreeClassifier()
     clf = clf.fit(iris.data, iris.target)
```

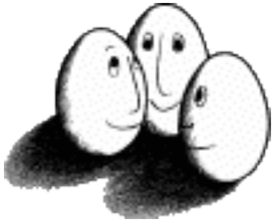
We can visualize the decision tree using the Graphviz open source graph visualization



```
from sklearn import tree, datasets
import graphviz, pickle
iris = datasets.load_iris()
clf = tree.DecisionTreeClassifier()
clf = clf.fit(iris.data, iris.target)
pickle.dump(clf, open(‘iris.p’, ‘wb’))
tree.export_graphviz(clf, out_file=“iris.pdf”)
```

http://bit.ly/iris671

Weka vs. scikit-learn vs. …



- Weka: good for experimenting with many ML algorithms
  - Other tools are more efficient & scalable
- Scikit-learn: popular and efficient suite of open-source machine-learning tools in Python
  - Uses NumPy, SciPy, matplotlib for efficiency
  - Preloaded into Google’s Colaboratory
- Custom apps for a specific ML algorithm are often preferred for speed or features