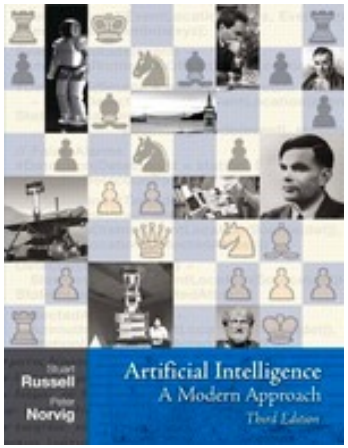
Decision Trees in AIMA, WEKA, and SCIKIT-LEARN



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

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

- Est. 1987!
- 370 data sets

Welcome to the UC Irvine Machine Learning Repository!

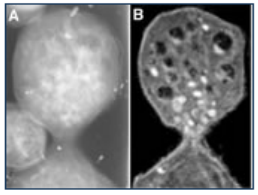
We currently maintain 233 data sets as a service to the machine learning community. You may **view all data sets** through our searchable interface. Our old web site is still available, [illegible] format. For a general overview of the Repository, please visit our About page. For information about citing 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: National Science Foundation 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

About Citation Policy Donate a Data Set Contact

Search

Repository Web

Google™

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

```
> aipython
>>> from learning4e import *
>>> zoo
<DataSet(zoo): 101 examples, 18 attributes>
>>> zdt = DecisionTreeLearner(zoo)
>>> zdt(['shark',0,0,1,0,0,1,1,1,1,0,0,1,0,1,0,0]) #eggs=1
'fish'
>>> zdt(['shark',0,0,0,0,0,1,1,1,1,0,0,1,0,1,0,0]) #eggs=0
‘mammal’
```

Zoo example

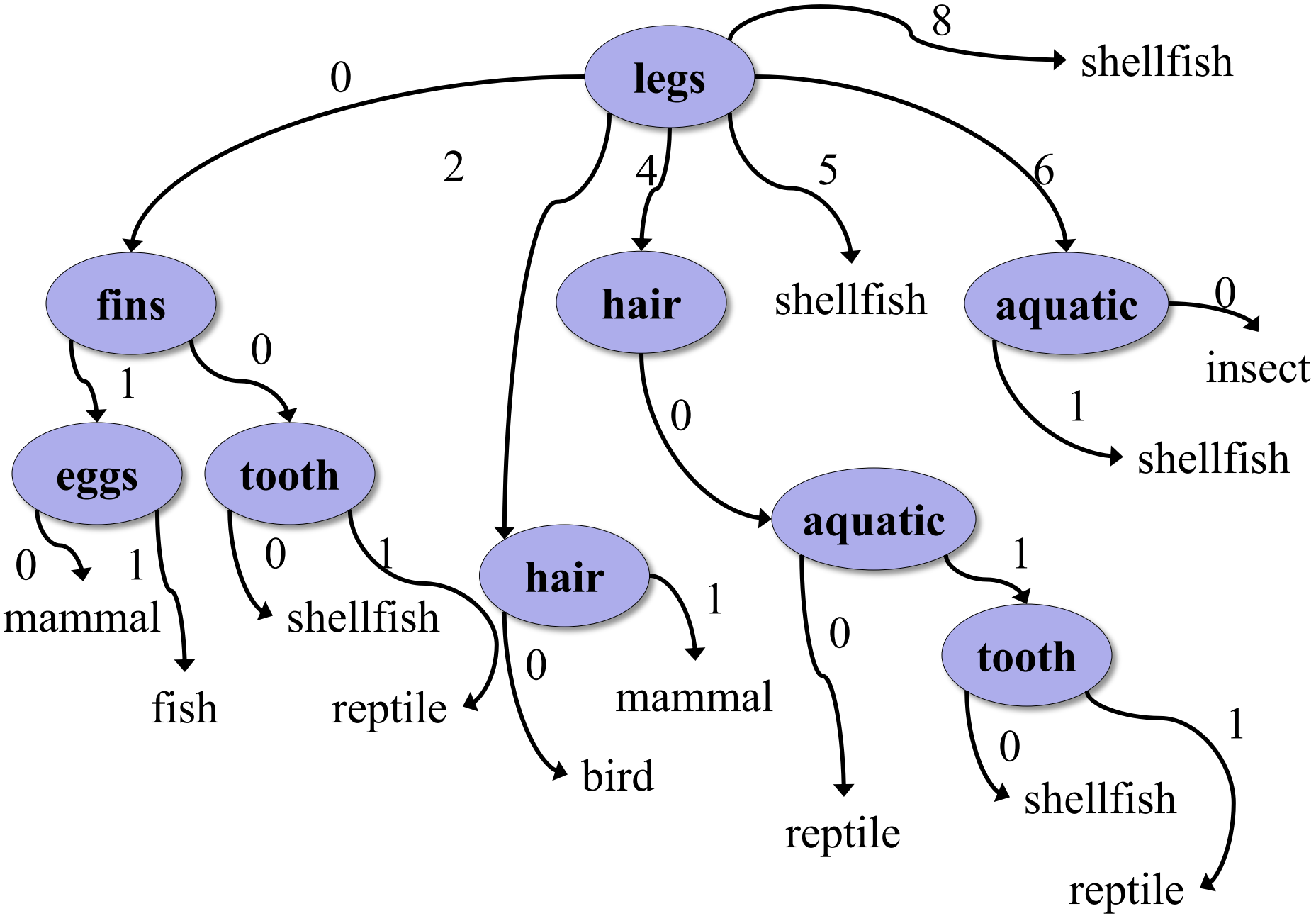
```
>> zdt
DecisionTree(13, 'legs', {0: DecisionTree(12, 'fins', {0:
DecisionTree(8, 'toothed', {0: 'shellfish', 1: 'reptile'}), 1:
DecisionTree(3, 'eggs', {0: 'mammal', 1: 'fish'})}), 2:
DecisionTree(1, 'hair', {0: 'bird', 1: 'mammal'}), 4:
DecisionTree(1, 'hair', {0: DecisionTree(6, 'aquatic', {0: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

AIMA's decision tree representation difficult for people to interpret

Zoo example

```
>>> zt.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
```

Better, but still difficult for people to understand


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

```
>>> 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
```

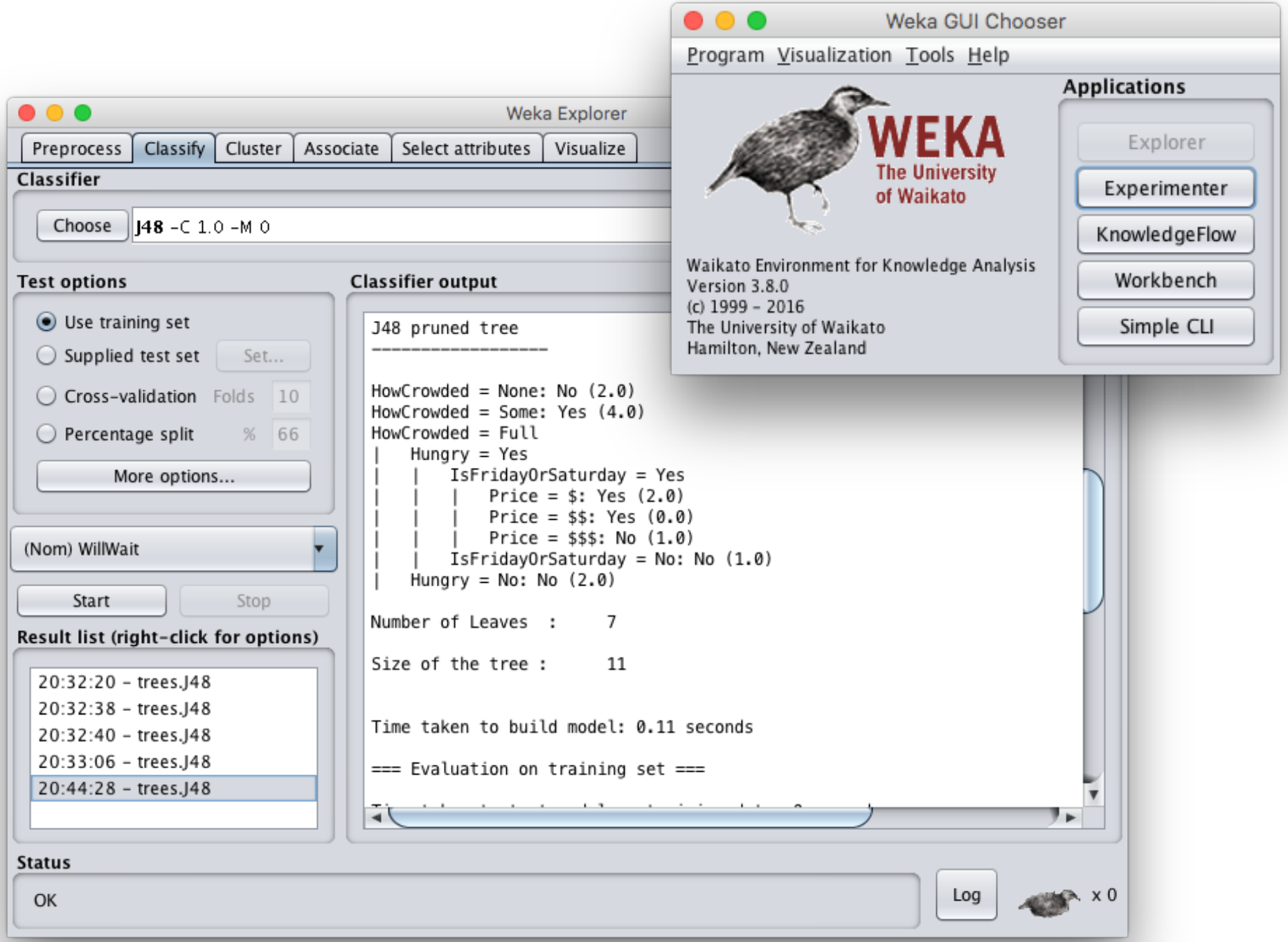
Zoo example

After adding the shark example to the training data & retraining

['shark',0,0,0,0,0,1,1,1,1,0,0,1,0,1,0,0, 'fish']

Weka

- 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


Weka GUI Chooser
Program Visualization Tools Help
WEKA
The University of Waikato
Waikato Environment for Knowledge Analysis
Version 3.8.0
(c) 1999 - 2016
The University of Waikato
Hamilton, New Zealand
Applications
Explorer
Experimenter
KnowledgeFlow
Workbench
Simple CLI
Weka Explorer
Preprocess Classify Cluster Associate Select attributes Visualize
Classifier
Choose
J48 -C 1.0 -M 0
Test options
Use training set
Supplied test set
Set...
Cross-validation Folds 10
Percentage split % 66
More options...
(Nom) WillWait
Start
Stop
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
Log
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*

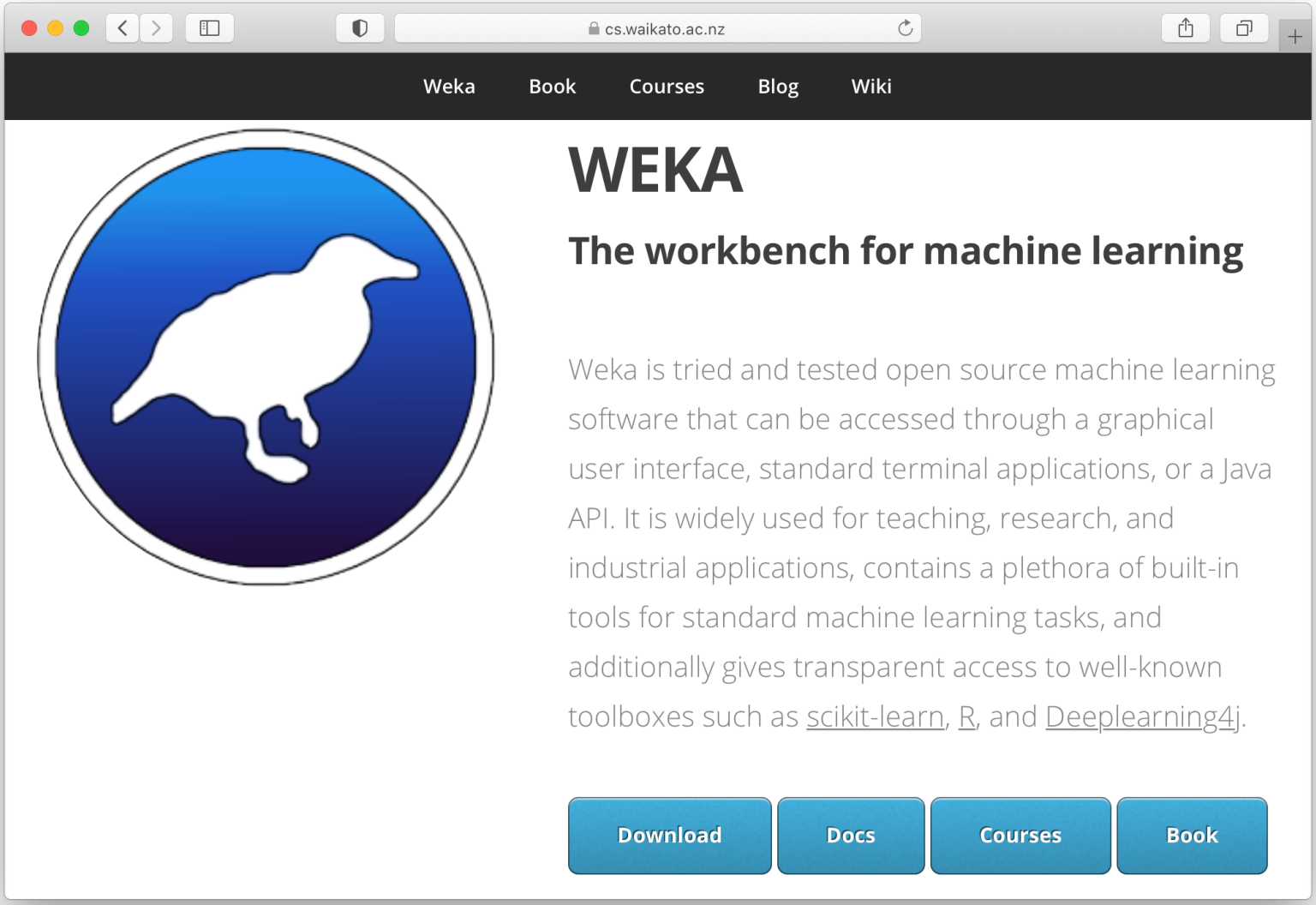
sex *is a nominal attribute*

class *is target variable*

Training data

*ARFF = Attribute-Relation File Format

Weka demo



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

Install Weka

- Download and install Weka
  - Requires Java
- 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)

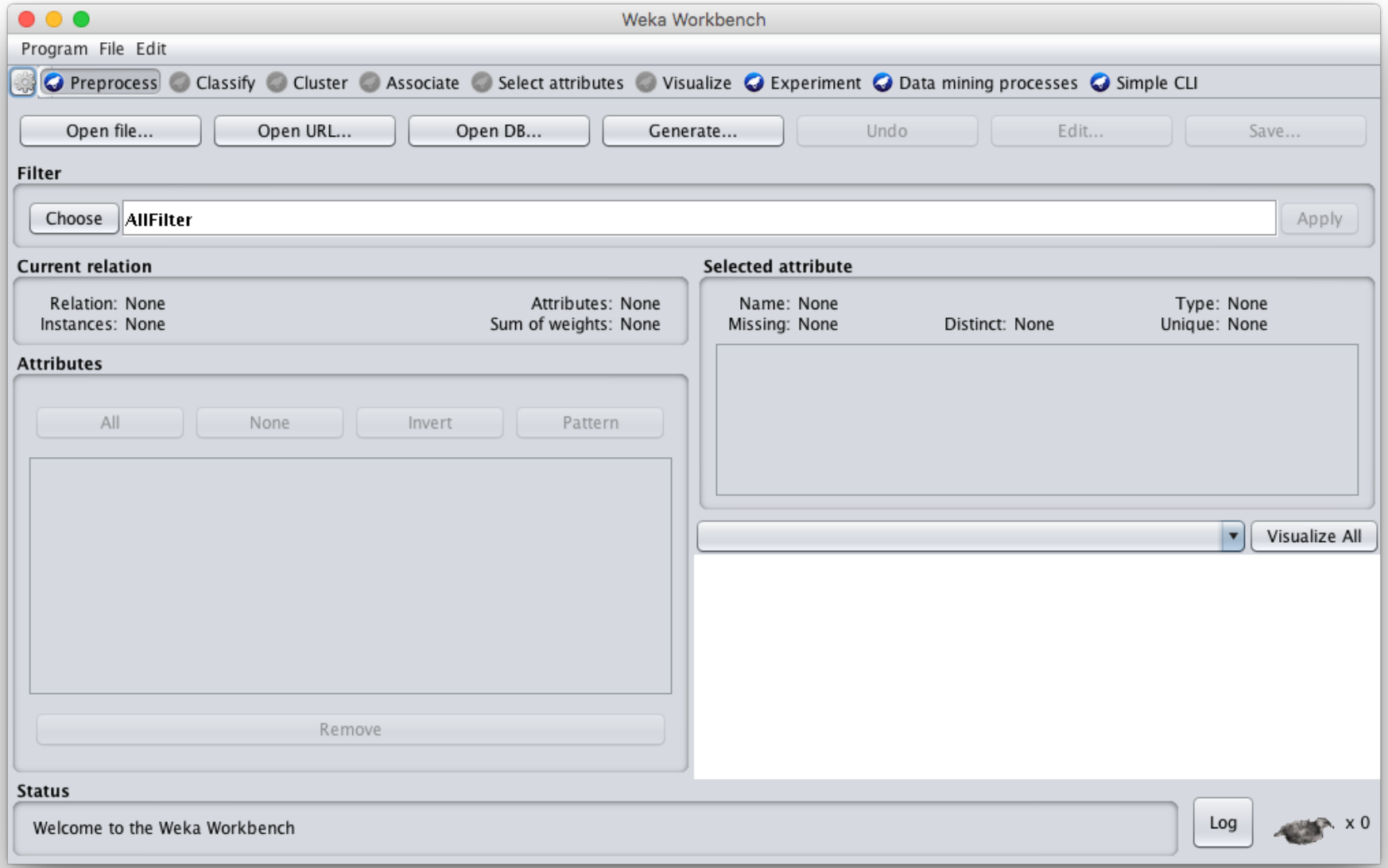
Getting your data ready

- 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
- #2 is an arff version of #1
- We'll train and save a model with #2
- Test it with #3
- Predict target on new data with #4

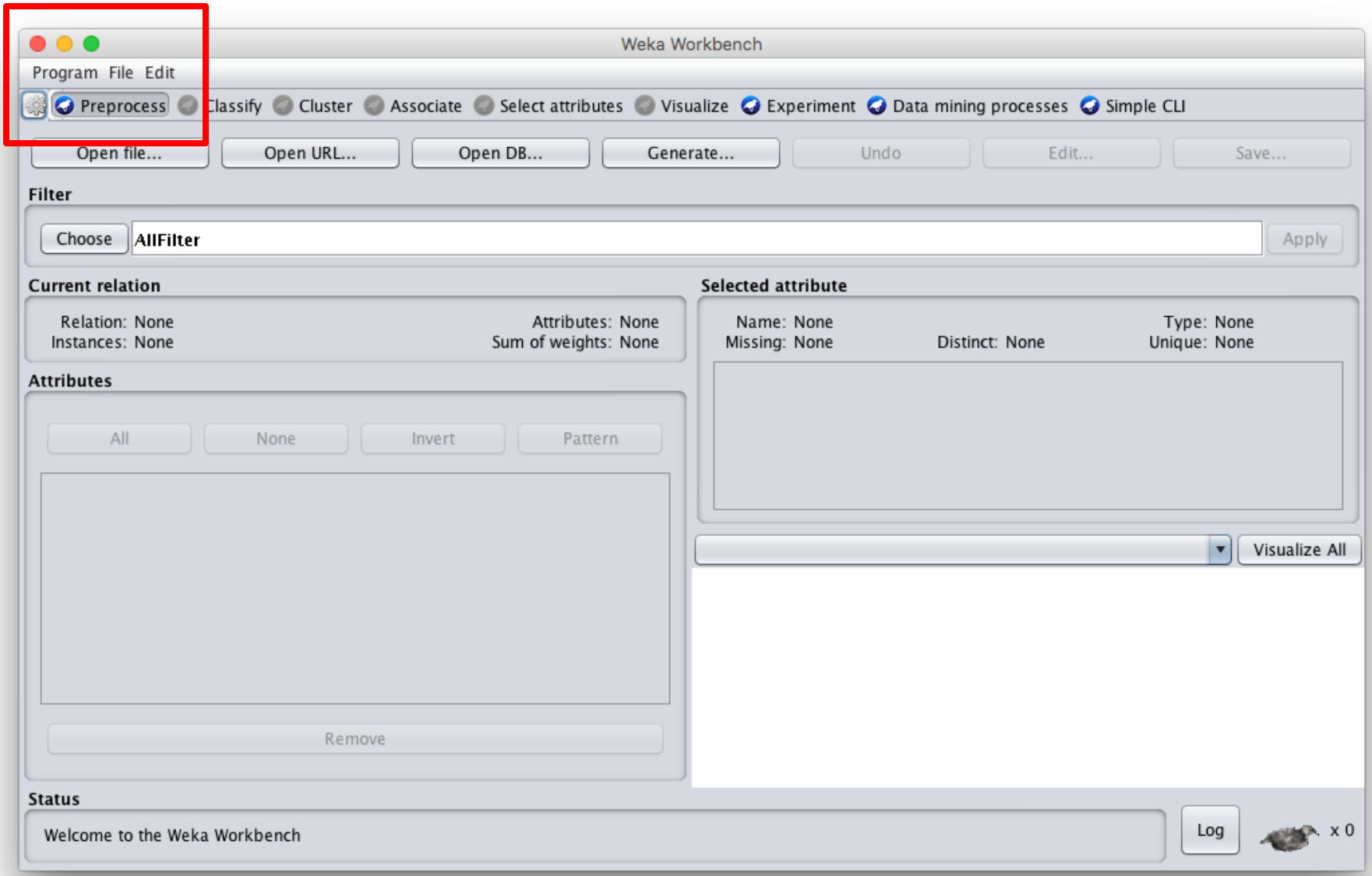
Open Weka app

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

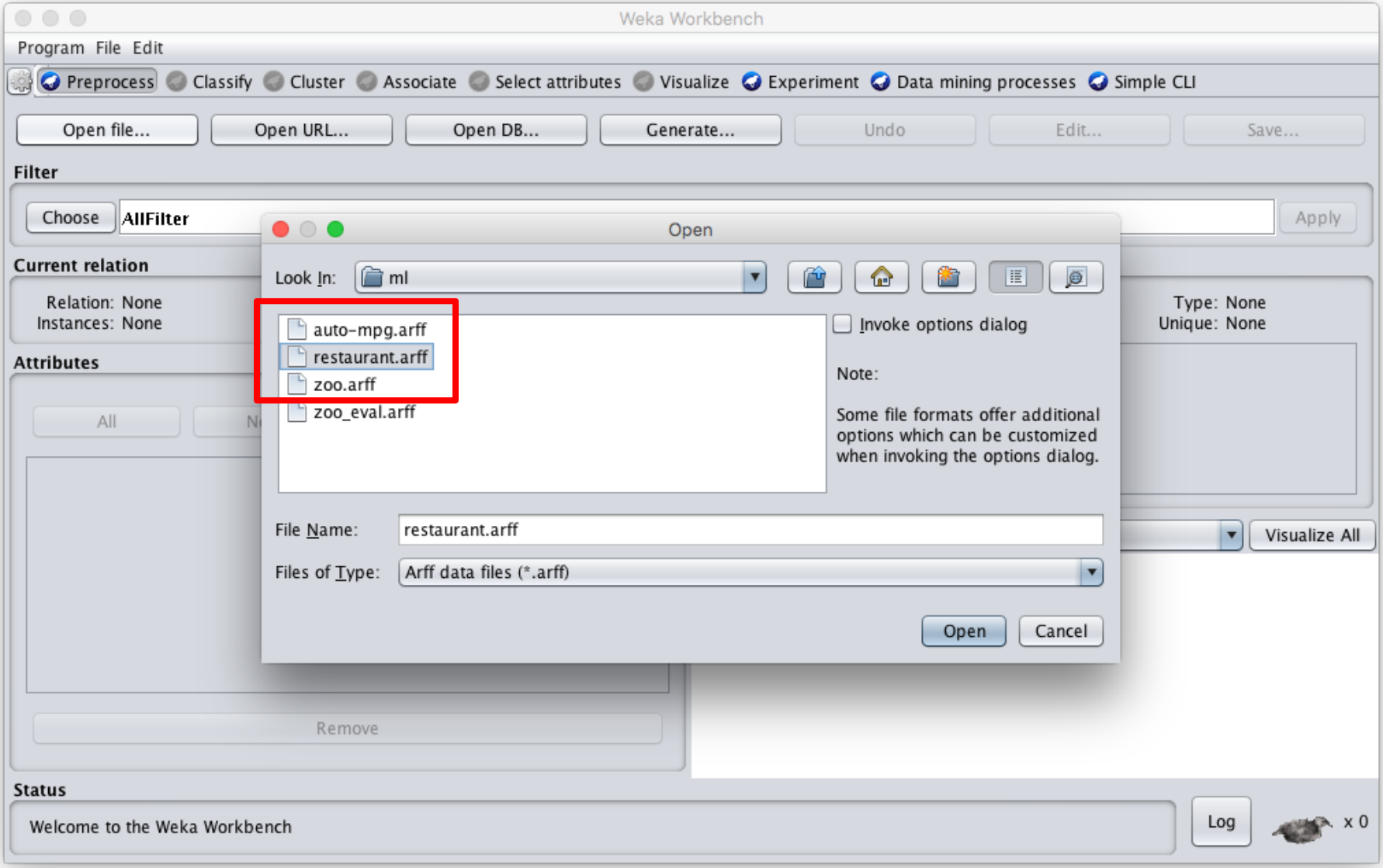
Explorer Interface



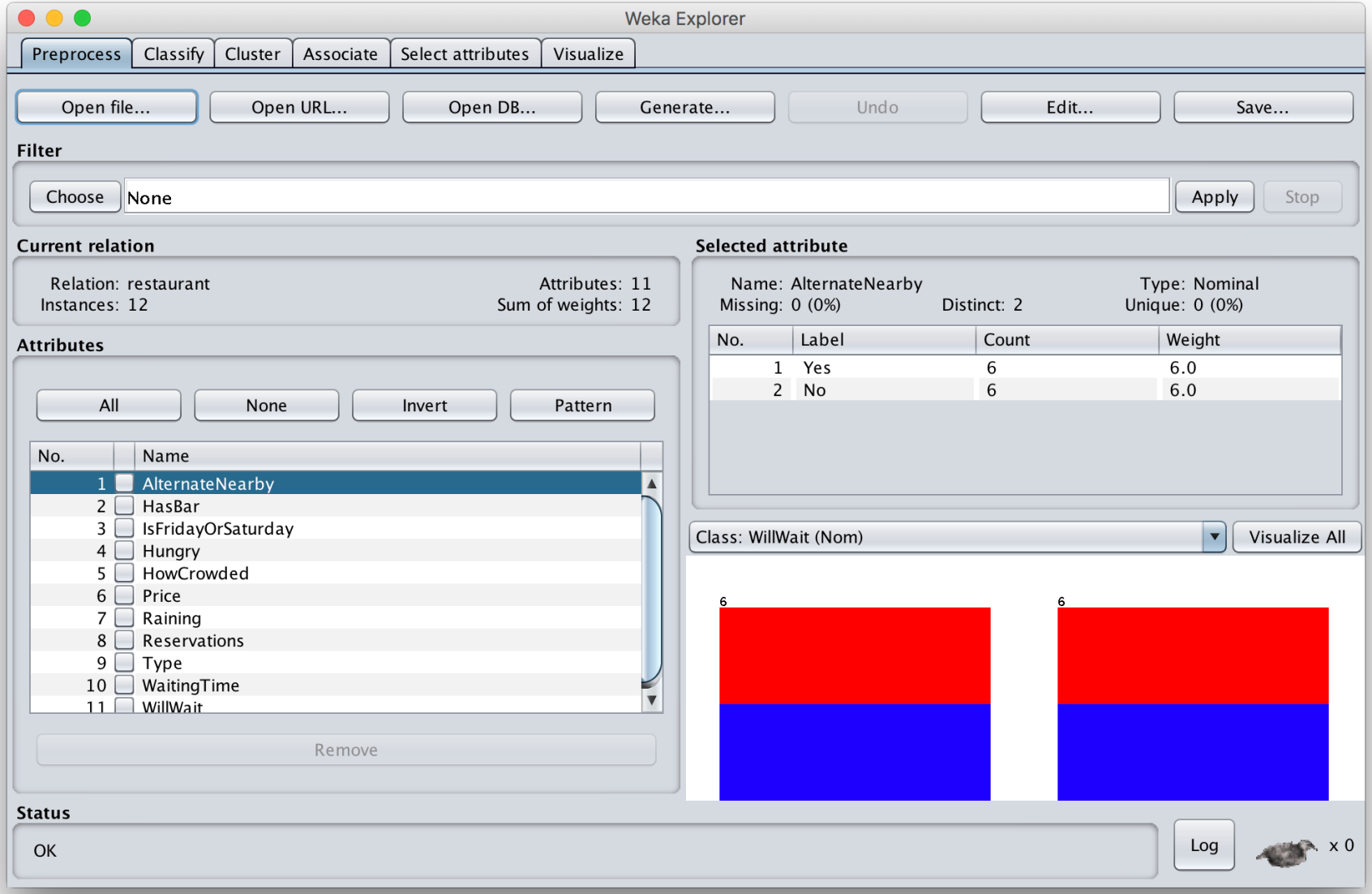
Starts with Data Preprocessing; open file to load data



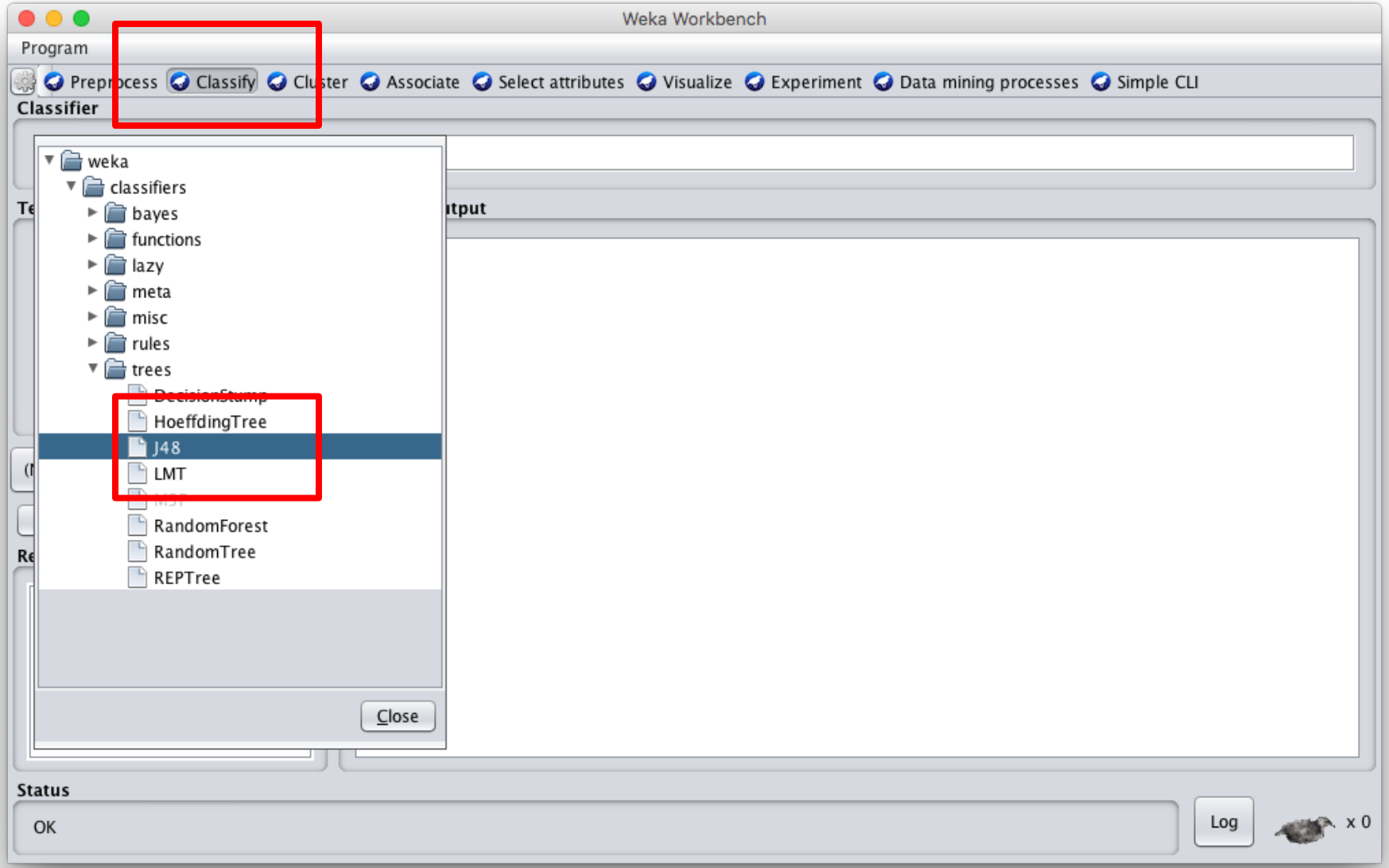
Load restaurant.arff training data



We can inspect/remove features



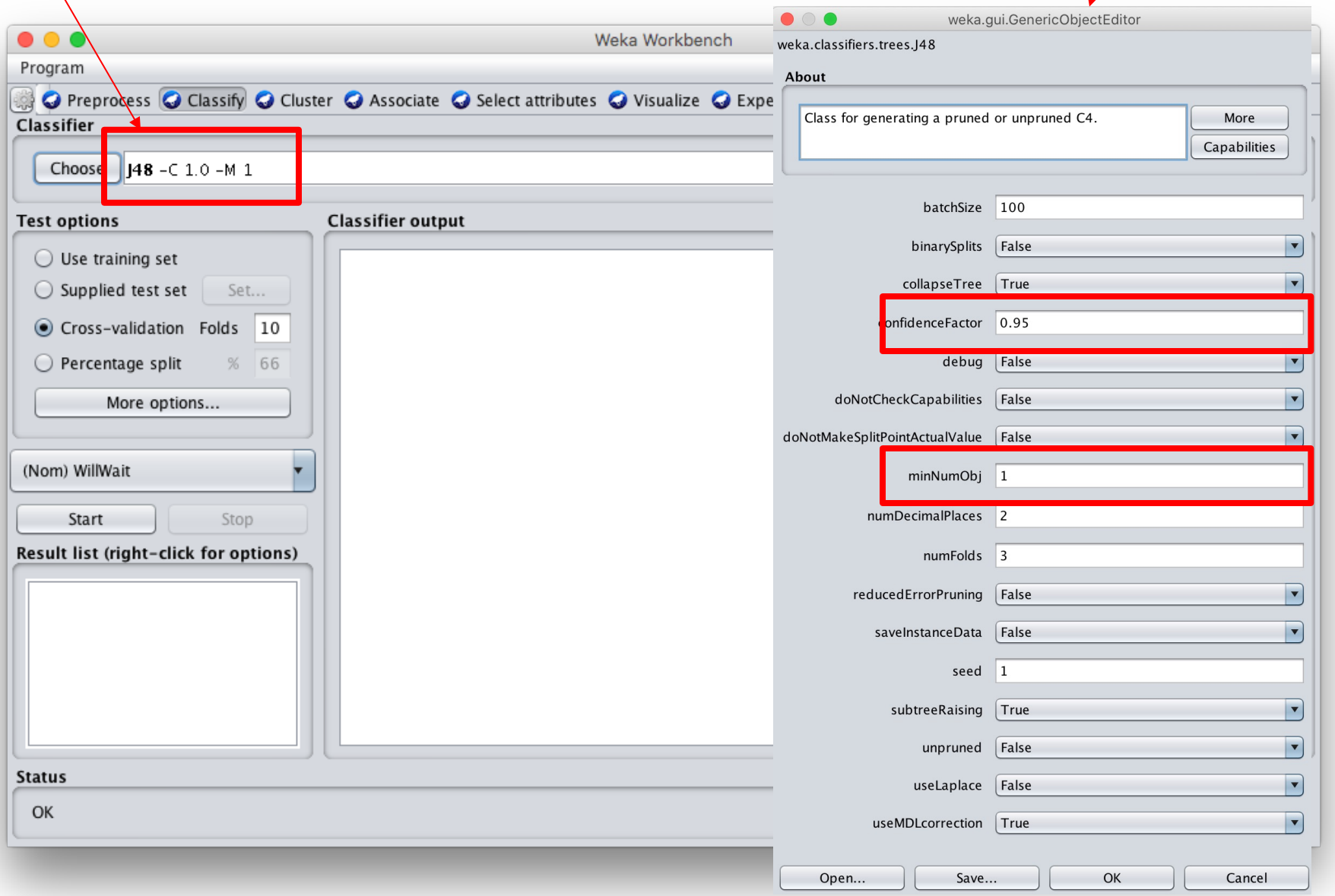
Select: classify > choose > trees > J48



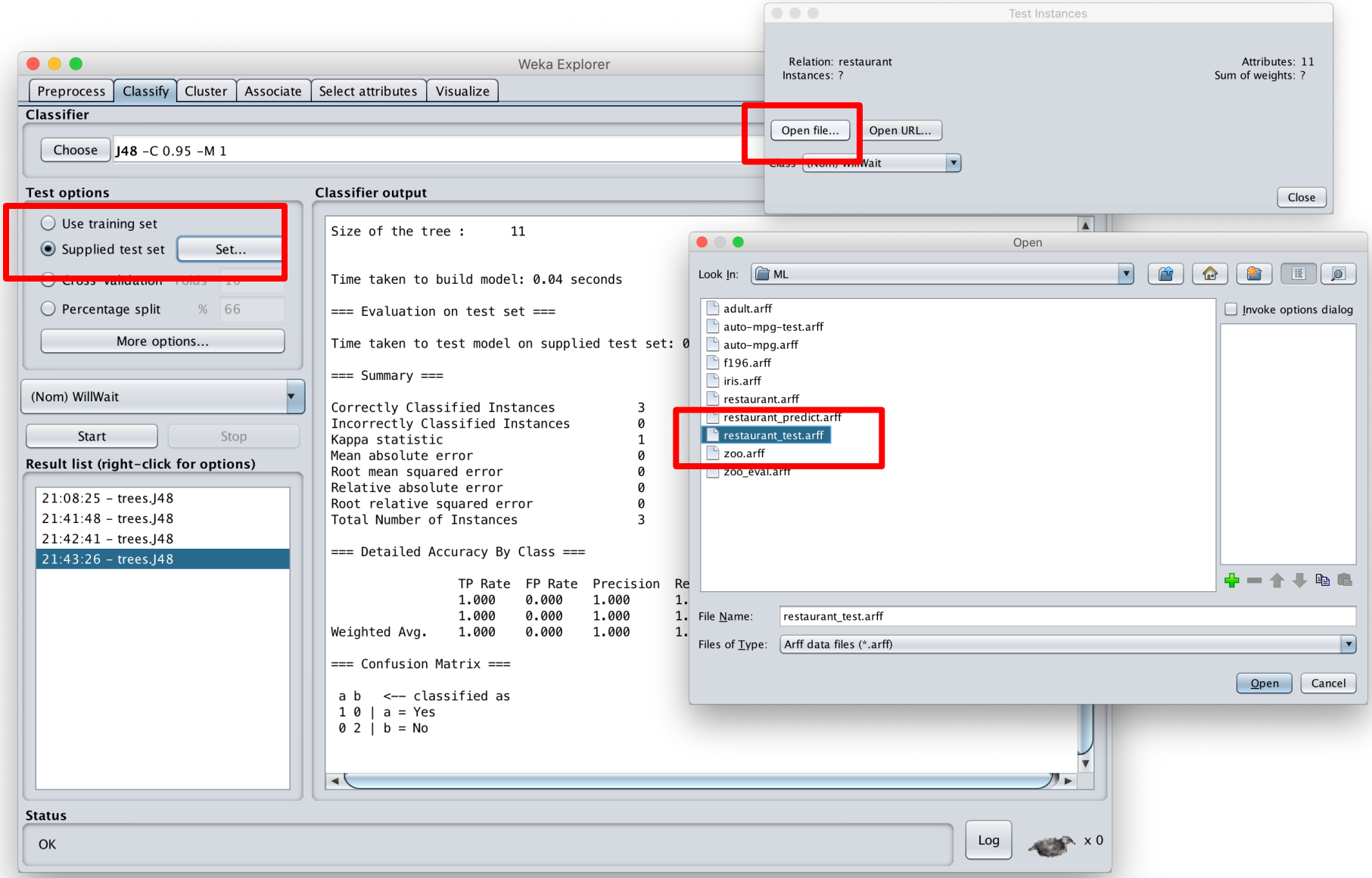
Adjust parameters

Note command line like syntax

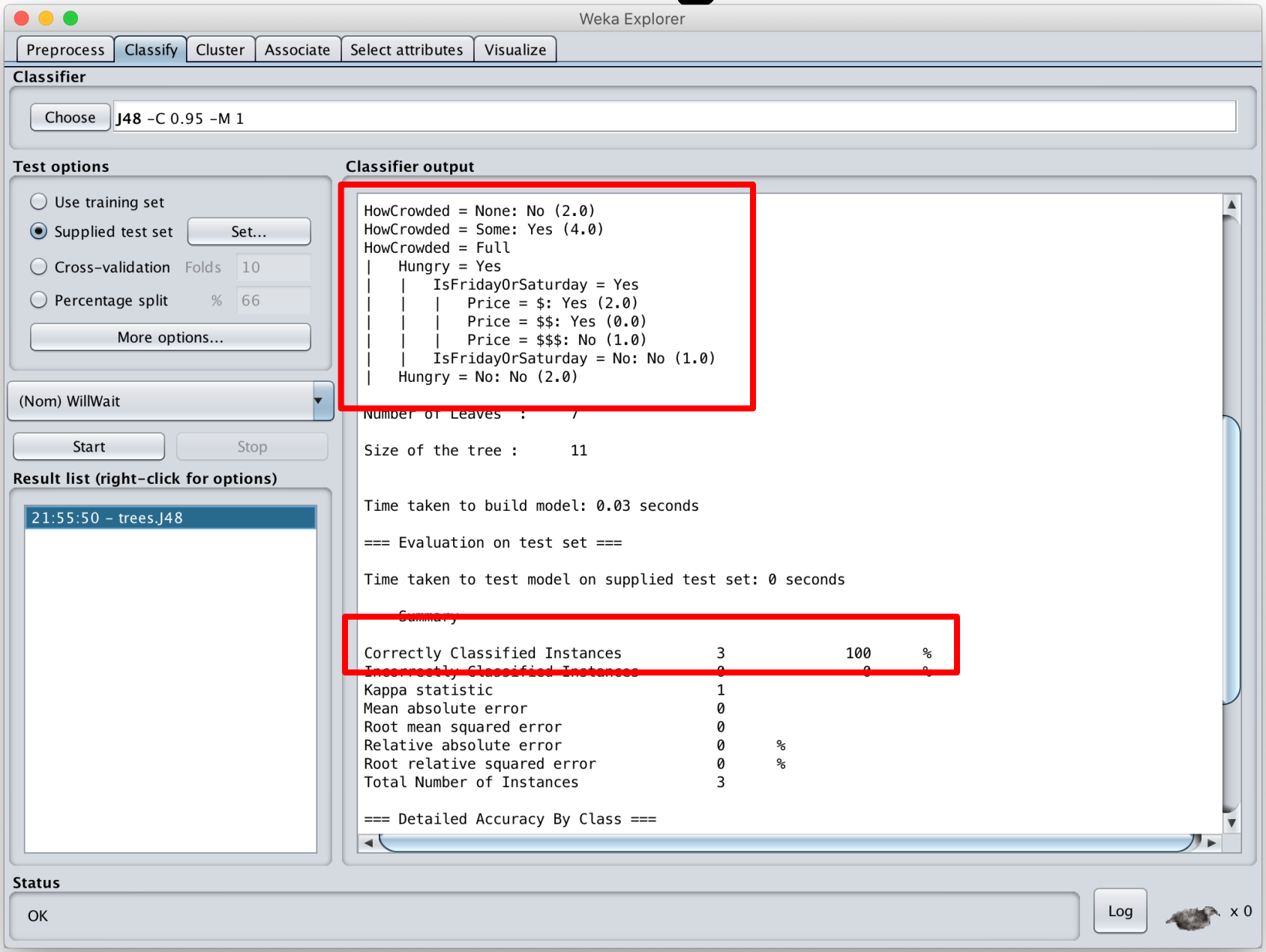
Change parameters here



Select the testing procedure



See training results



Compare results

HowCrowded = None: No (2.0)

HowCrowded = Some: Yes (4.0)

HowCrowded = Full

| Hungry = Yes

| | IsFridayOrSaturday = Yes

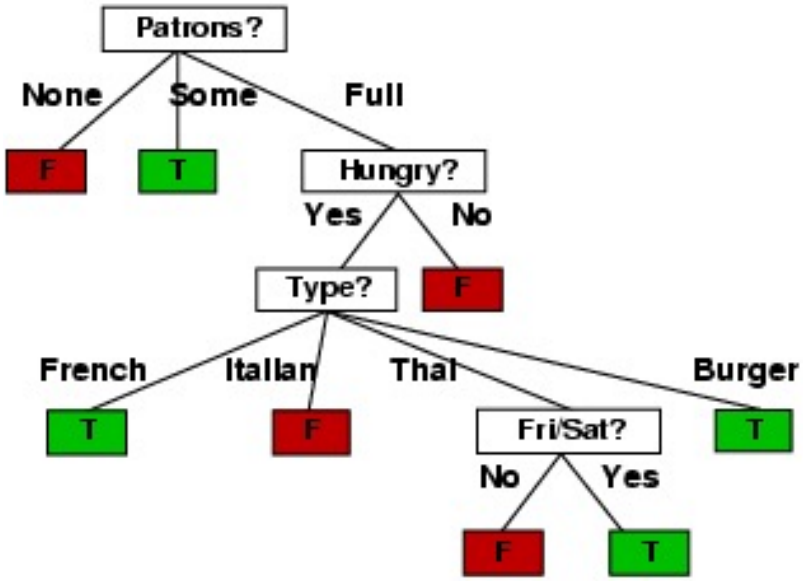
| | | Price = $: Yes (2.0)

| | | Price = $$: Yes (0.0)

| | | Price = $$$: No (1.0)

| | IsFridayOrSaturday = No: No (1.0)

| Hungry = No: No (2.0)



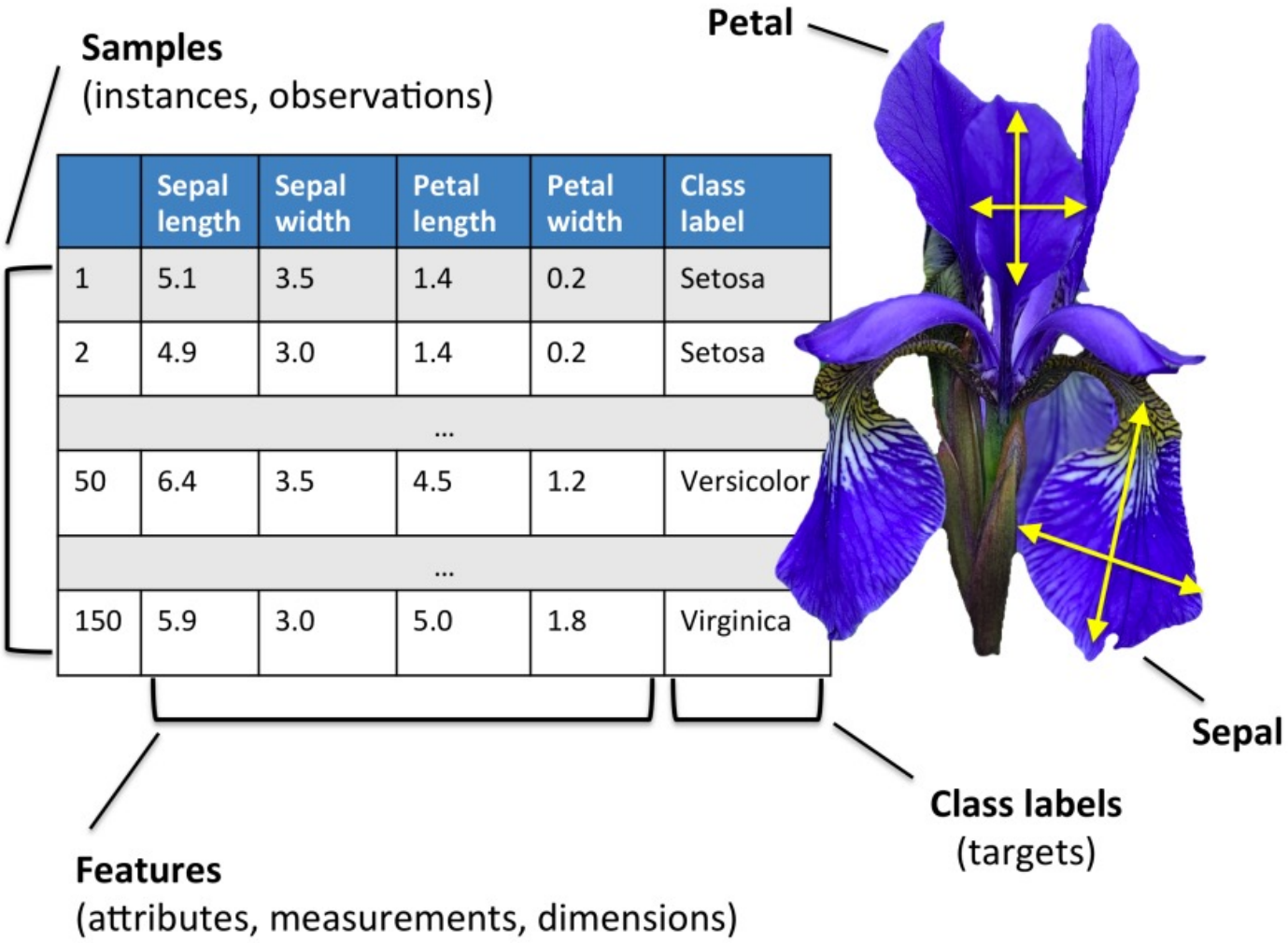
J48 pruned tree: nodes:11; leaves:7, max depth:4

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

The two decision trees are equally good

scikit-learn

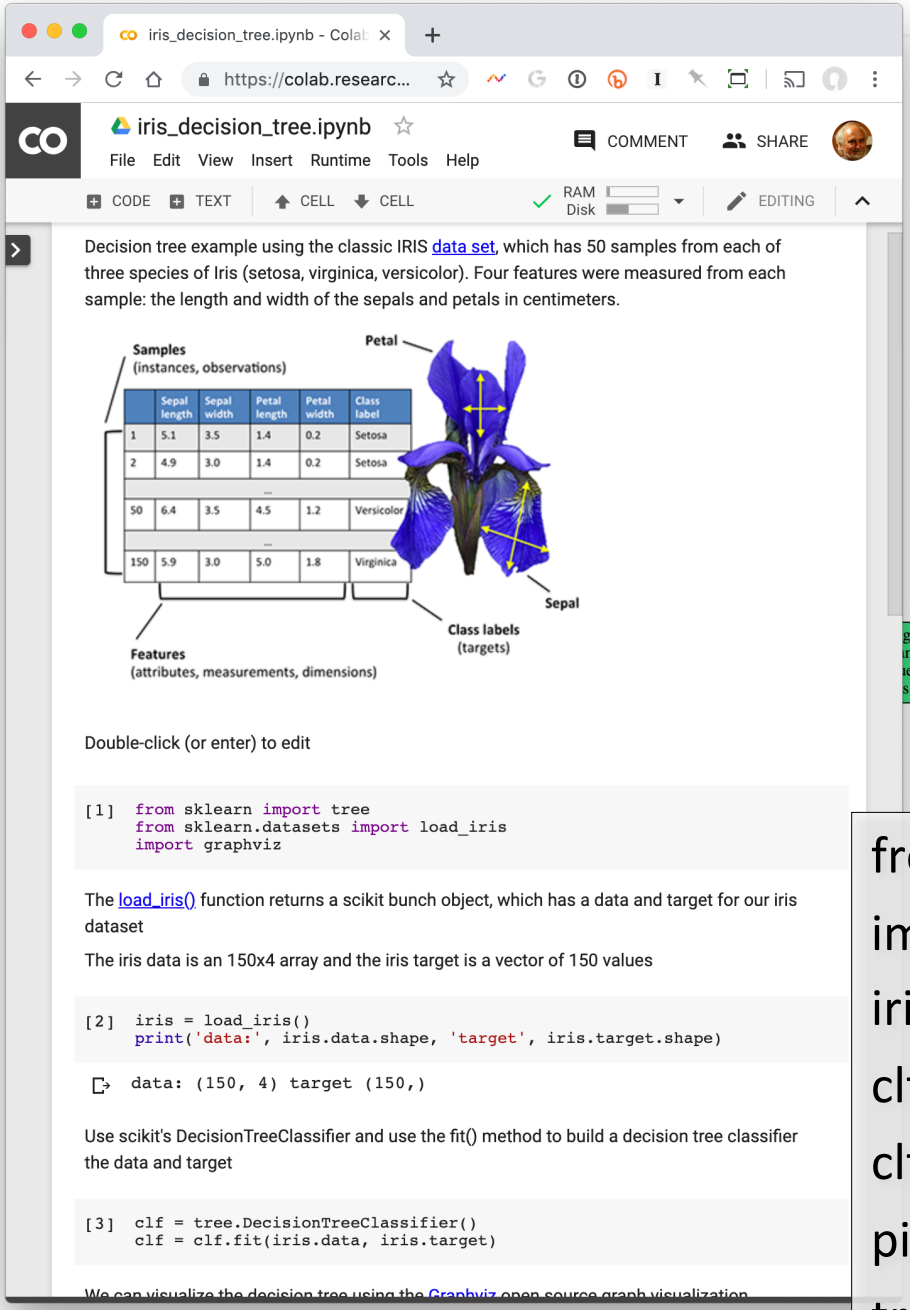
- Popular open source ML and data analysis tools for Python
- Built on NumPy, SciPy, and matplotlib for efficiency
- However, decision tree tools are a weak area
  - E.g., data features must be numeric, so working with restaurant example requires conversion
  - Perhaps because DTs not used for large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

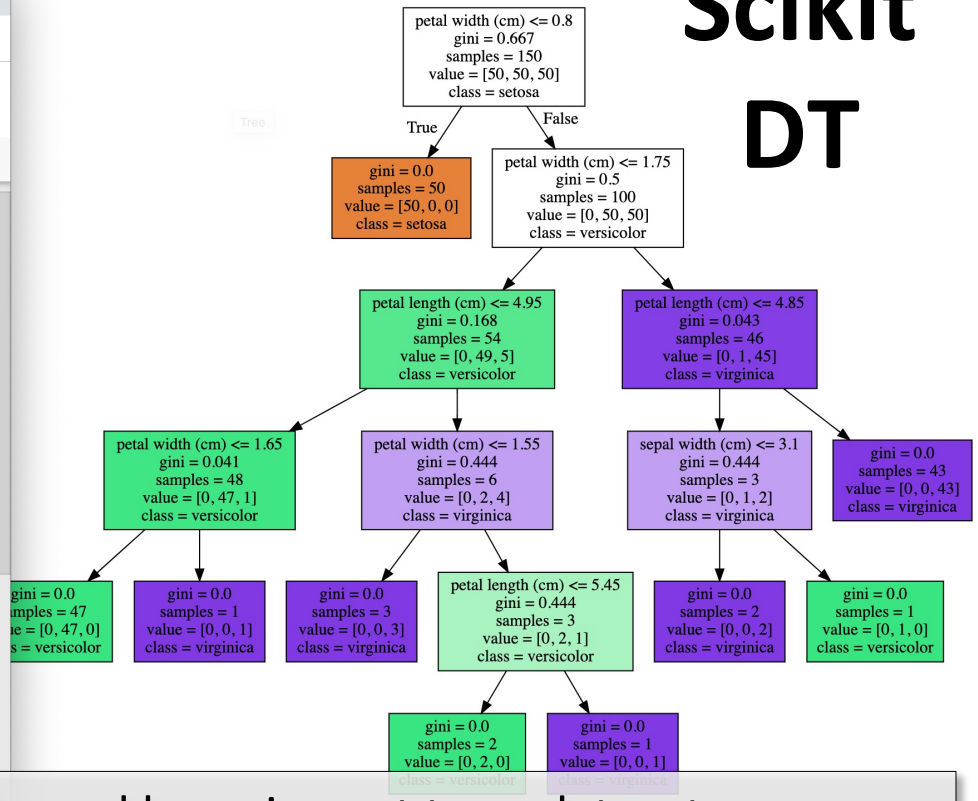


	Sepal length	Sepal width	Petal length	Petal width	Class label
1	5.1	3.5	1.4	0.2	Setosa
2	4.9	3.0	1.4	0.2	Setosa
			...		
50	6.4	3.5	4.5	1.2	Versicolor
			...		
150	5.9	3.0	5.0	1.8	Virginica

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

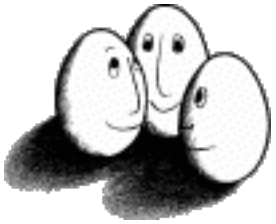





```
from sklearn import tree, datasets
import graphviz, pickle
iris = datasets.load_iris()
clf = tree.DecisionTreeClassifier()
clf = clf.fit(iris.data, iris.target)
pickle.dump(clf, open('iris.p', 'wb'))
tree.export_graphviz(clf, out_file="iris.pdf")
```

Try it on colab

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