Using the Weka Machine Learning toolkit

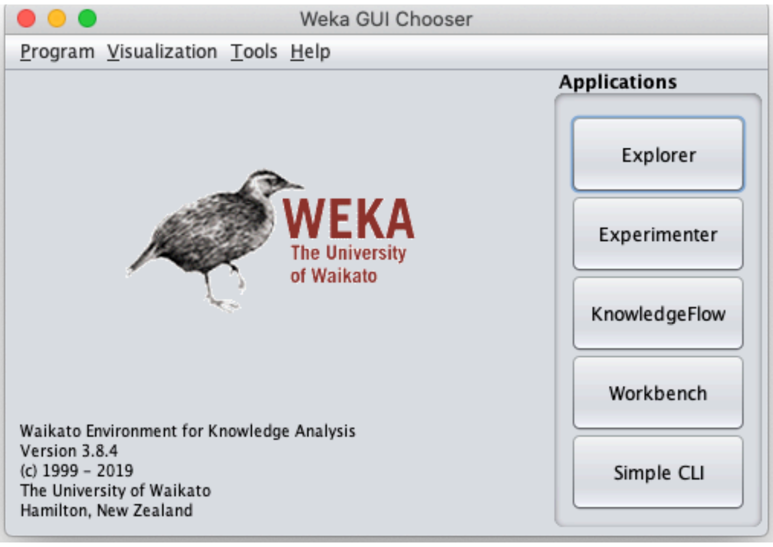
Weka is a Machine Learning toolkit written in Java. It was developed and is maintained by a team from Waikato University in New Zealand but it has packages contributed from many different sources.

Weka is reasonably easy to use as it has a GUI so you don't need to do any programming. It runs on Linux, macOS and Windows. You can download it from:

https://waikato.github.io/weka-wiki/downloading_weka/

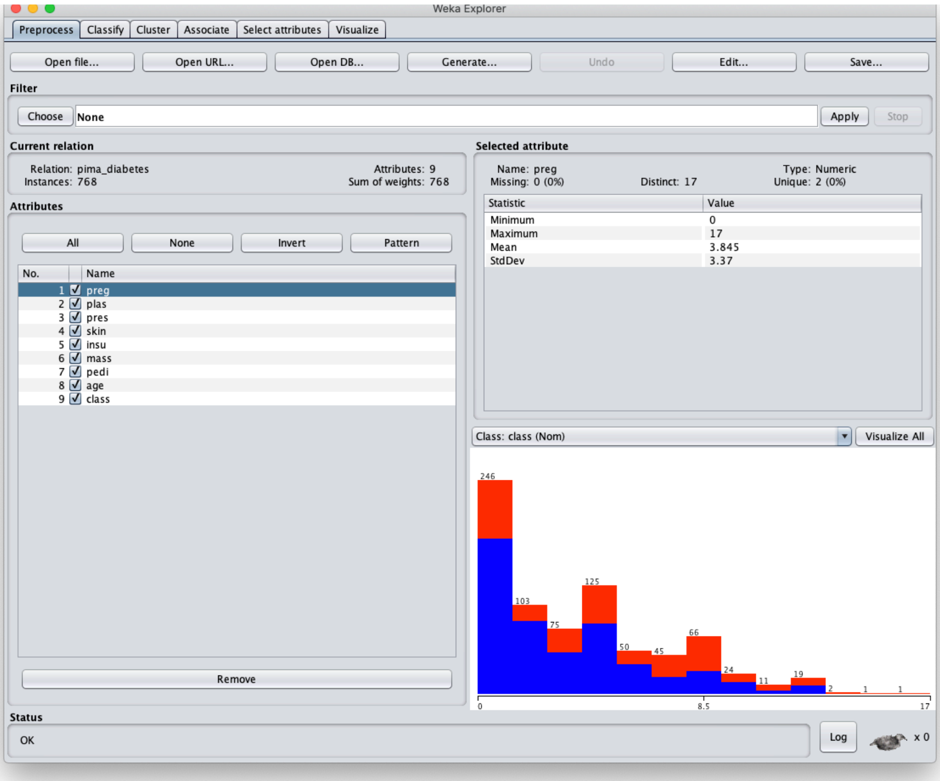
The download includes a Java JVM, so you shouldn't need to install Java, if you haven't got it already.

Once you have it installed, when you start Weka, you will see window like this:



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

1. Click on the explorer button and you will get a new window:



Filter

None

Current relation

Relation: pima_diabetes
Instances: 768
Attributes: 9
Sum of weights: 768

Attributes

No.		Name
1	☑	preg
2	☑	plas
3	☑	pres
4	☑	skin
5	☑	insu
6	☑	mass
7	☑	pedi
8	☑	age
9	☑	class

Selected attribute

Name: preg
Missing: 0 (0%)
Distinct: 17
Type: Numeric
Unique: 2 (0%)

Statistic	Value
Minimum	0
Maximum	17
Mean	3.845
StdDev	3.37

Class: class (Nom)

Status

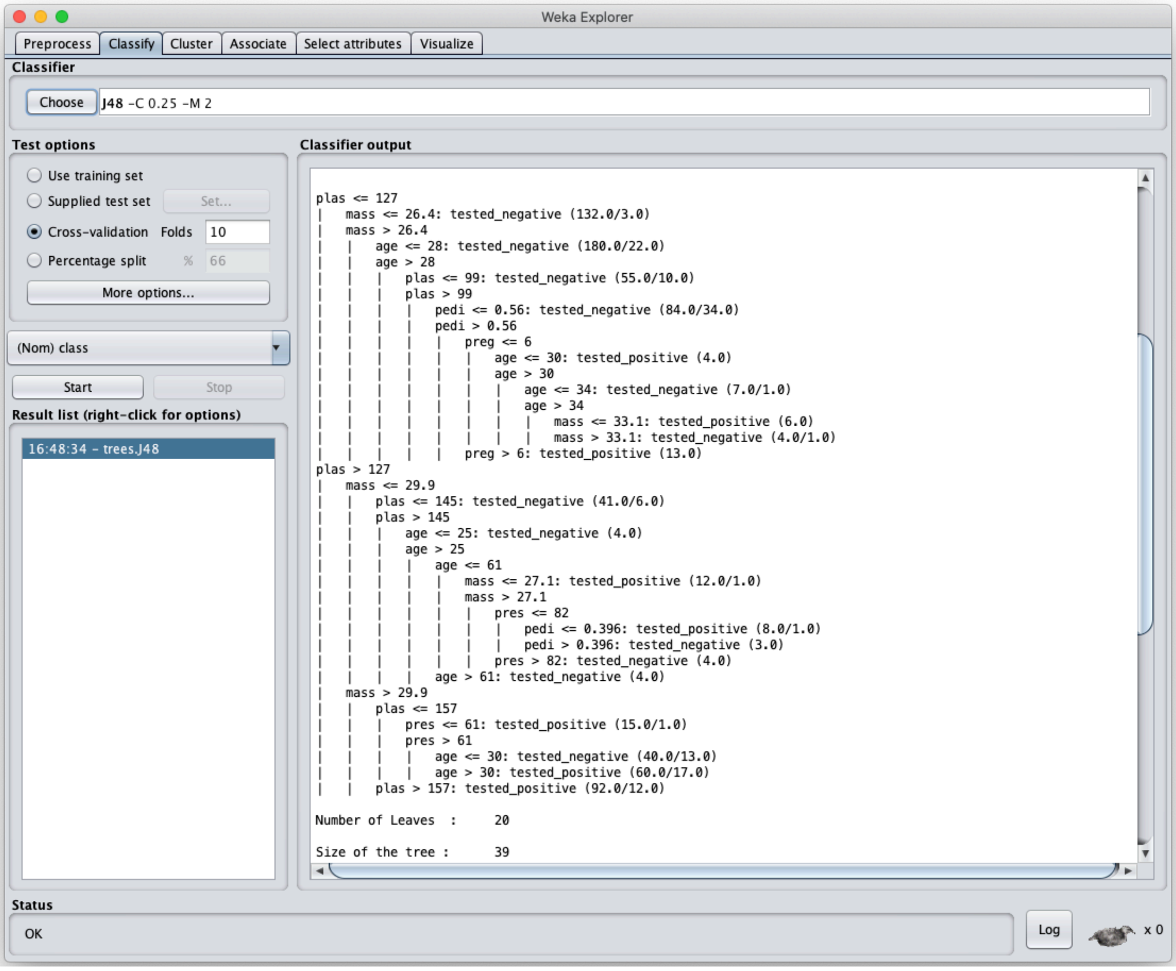
OK

This shows you the data panel.

2. Click “Open file …” to select the data file. This is in “ARFF” format. An example is appended at the end of this document. You can find example datasets in the ***data*** folder that is the installation folder, if you are using a Mac or Windows. If you use Linux, Weka will create a directory in your home directory.

3. By ticking the check boxes, you can see the distribution of objects with the ticked attribute values.

4. Next, click on the “classify” button (at the top of the window) to bring up the classification panel:



Classifier
J48 -C 0.25 -M 2

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

(Nom) class

Result list (right-click for options)
16:48:34 - trees.J48

Classifier output

```
plas <= 127
|   mass <= 26.4: tested_negative (132.0/3.0)
|   mass > 26.4
|   |   age <= 28: tested_negative (180.0/22.0)
|   |   age > 28
|   |   |   plas <= 99: tested_negative (55.0/10.0)
|   |   |   plas > 99
|   |   |   |   pedi <= 0.56: tested_negative (84.0/34.0)
|   |   |   |   pedi > 0.56
|   |   |   |   |   preg <= 6
|   |   |   |   |   |   age <= 30: tested_positive (4.0)
|   |   |   |   |   |   age > 30
|   |   |   |   |   |   |   age <= 34: tested_negative (7.0/1.0)
|   |   |   |   |   |   |   age > 34
|   |   |   |   |   |   |   |   mass <= 33.1: tested_positive (6.0)
|   |   |   |   |   |   |   |   mass > 33.1: tested_negative (4.0/1.0)
|   |   |   |   |   preg > 6: tested_positive (13.0)
plas > 127
|   mass <= 29.9
|   |   plas <= 145: tested_negative (41.0/6.0)
|   |   plas > 145
|   |   |   age <= 25: tested_negative (4.0)
|   |   |   age > 25
|   |   |   |   age <= 61
|   |   |   |   |   mass <= 27.1: tested_positive (12.0/1.0)
|   |   |   |   |   mass > 27.1
|   |   |   |   |   |   pres <= 82
|   |   |   |   |   |   |   pedi <= 0.396: tested_positive (8.0/1.0)
|   |   |   |   |   |   |   pedi > 0.396: tested_negative (3.0)
|   |   |   |   |   |   pres > 82: tested_negative (4.0)
|   |   |   |   age > 61: tested_negative (4.0)
|   mass > 29.9
|   |   plas <= 157
|   |   |   pres <= 61: tested_positive (15.0/1.0)
|   |   |   pres > 61
|   |   |   |   age <= 30: tested_negative (40.0/13.0)
|   |   |   |   age > 30: tested_positive (60.0/17.0)
|   |   plas > 157: tested_positive (92.0/12.0)

Number of Leaves  :     20

Size of the tree :      39
```

Status
OK

5. Clicking the “Choose” button will give you a list of different classifier learning algorithms. Choose ***J48*** under the ***Trees*** subcategory. This is Weka’s implementation of Quinlan’s C4.5 algorithm. You will find ***NaiveBayes*** under ***Bayes*** and ***MultilayerPerceptron*** under ***functions***. ***JRip*** is under ***rules***.

6. The default setting in for testing is 10-fold cross-validation. You can also click on “Percentage split” to get a default 66% training data 34% test data.

7. Clicking “Start” will run the learning algorithm on the data set and produce the output on the right. The panel contain error statistics and timing information.

Example ARFF File

The file needs an **@relation** line to tell it the name of the data table.

Each attribute is defined in an **@attribute** line, which also give the type of the attribute. A real valued attribute is ‘real’; discrete values attributes are given a list of the valid values. A question mark means missing value.

The example below is from the UCI repository. It is not the complete data file, as it is too big to list here. If you are interested, more example data sets are here:

https://waikato.github.io/weka-wiki/datasets/

```
@relation labor
@attribute 'duration' real
@attribute 'wage-increase-first-year' real
@attribute 'wage-increase-second-year' real
@attribute 'wage-increase-third-year' real
@attribute 'cost-of-living-adjustment' {'none','tcf','tc'}
@attribute 'working-hours' real
@attribute 'pension' {'none','ret_allw','empl_contr'}
@attribute 'standby-pay' real
@attribute 'shift-differential' real
@attribute 'education-allowance' {'yes','no'}
@attribute 'statutory-holidays' real
@attribute 'vacation' {'below_average','average','generous'}
@attribute 'longterm-disability-assistance' {'yes','no'}
@attribute 'contribution-to-dental-plan' {'none','half','full'}
@attribute 'bereavement-assistance' {'yes','no'}
@attribute 'contribution-to-health-plan' {'none','half','full'}
@attribute 'class' {'bad','good'}
@data
1,5,?,?,?,40,?,?,2,?,11,'average',?,?,'yes',?,'good'
2,4.5,5.8,?,?,35,'ret_allw',?,?,'yes',11,'below_average',?,'full',?,'full','good'
?,?,?,?,?,38,'empl_contr',?,5,?,11,'generous','yes','half','yes','half','good'
3,3.7,4,5,'tc',?,?,?,?,'yes',?,?,?,?,'yes',?,'good'
3,4.5,4.5,5,?,40,?,?,?,?,12,'average',?,'half','yes','half','good'
2,2,2.5,?,?,35,?,?,6,'yes',12,'average',?,?,?,?,'good'
3,4,5,5,'tc',?,'empl_contr',?,?,?,12,'generous','yes','none','yes','half','good'
3,6.9,4.8,2.3,?,40,?,?,3,?,12,'below_average',?,?,?,?,'good'
2,3,7,?,?,38,?,12,25,'yes',11,'below_average','yes','half','yes',?,'good'
1,5.7,?,?,'none',40,'empl_contr',?,4,?,11,'generous','yes','full',?,?,'good'
3,3.5,4,4.6,'none',36,?,?,3,?,13,'generous',?,?,'yes','full','good'
2,6.4,6.4,?,?,38,?,?,4,?,15,?,?,'full',?,?,'good'
2,3.5,4,?,'none',40,?,?,2,'no',10,'below_average','no','half',?,'half','bad'
3,3.5,4,5.1,'tcf',37,?,?,4,?,13,'generous',?,'full','yes','full','good'
1,3,?,?,'none',36,?,?,10,'no',11,'generous',?,?,?,?,'good'
2,4.5,4,?,'none',37,'empl_contr',?,?,?,11,'average',?,'full','yes',?,'good'
1,2.8,?,?,?,35,?,?,2,?,12,'below_average',?,?,?,?,'good'
1,2.1,?,?,'tc',40,'ret_allw',2,3,'no',9,'below_average','yes','half',?,'none','bad'
```