SINGAPORE UNIVERSITY OF
TECHNOLOGY AND DESIGN

Established in collaboration with MIT

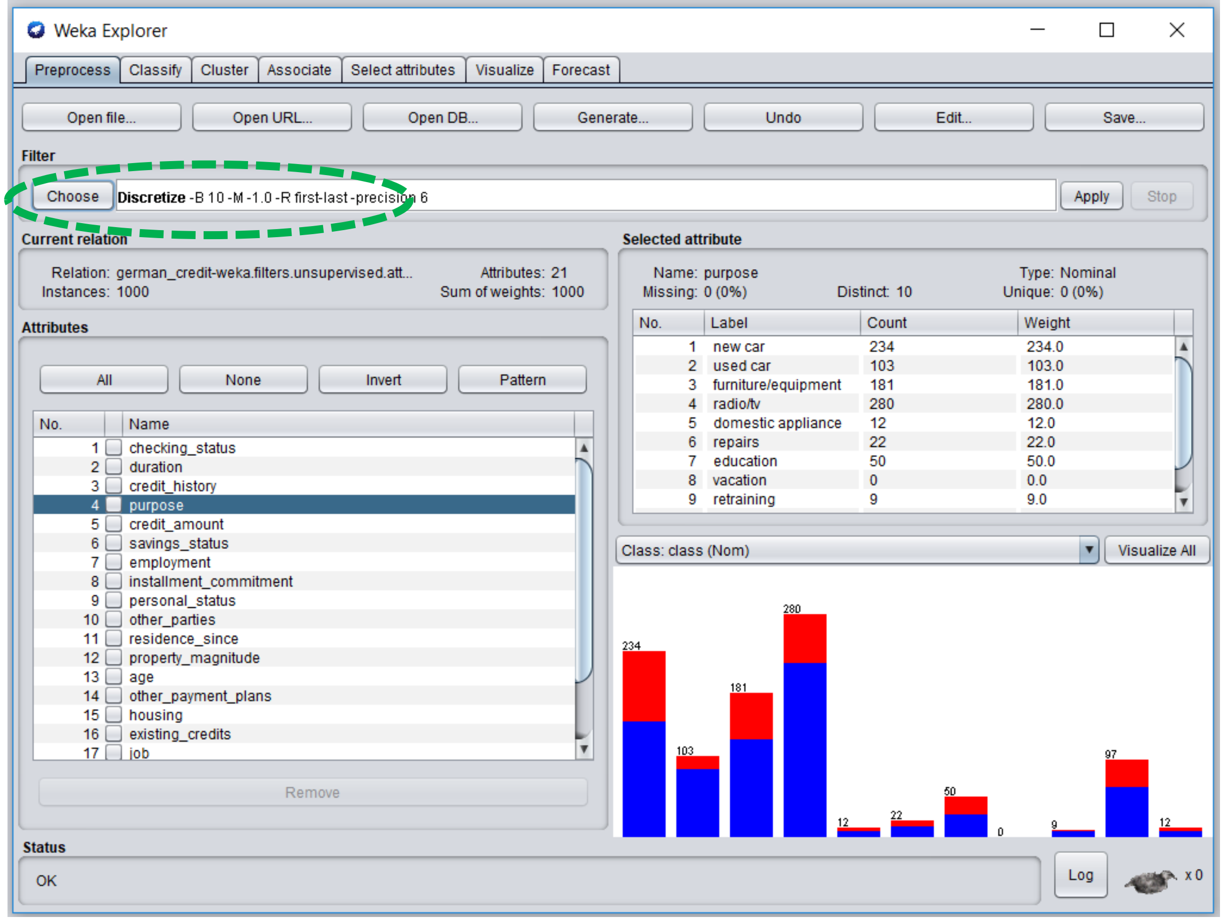
Associations and Clustering

PROF. K. H. LIM

50.038 Computational data science

Pre-processing

- Various pre-processing steps to handle feature
- Look at Filter→Unsupervised→Attribute
- Which will you need for association rule mining?



Filter

Discretize -B 10 -M -1.0 -R first-last -precision 6

Current relation

Relation: german_credit-weka.filters.unsupervised.att... Attributes: 21
Instances: 1000 Sum of weights: 1000

Selected attribute

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

No.	Label	Count	Weight
1	new car	234	234.0
2	used car	103	103.0
3	furniture/equipment	181	181.0
4	radio/tv	280	280.0
5	domestic appliance	12	12.0
6	repairs	22	22.0
7	education	50	50.0
8	vacation	0	0.0
9	retraining	9	9.0

Attributes

No.	Name
1	☐ checking_status
2	☐ duration
3	☐ credit_history
4	☐ purpose
5	☐ credit_amount
6	☐ savings_status
7	☐ employment
8	☐ installment_commitment
9	☐ personal_status
10	☐ other_parties
11	☐ residence_since
12	☐ property_magnitude
13	☐ age
14	☐ other_payment_plans
15	☐ housing
16	☐ existing_credits
17	☐ job

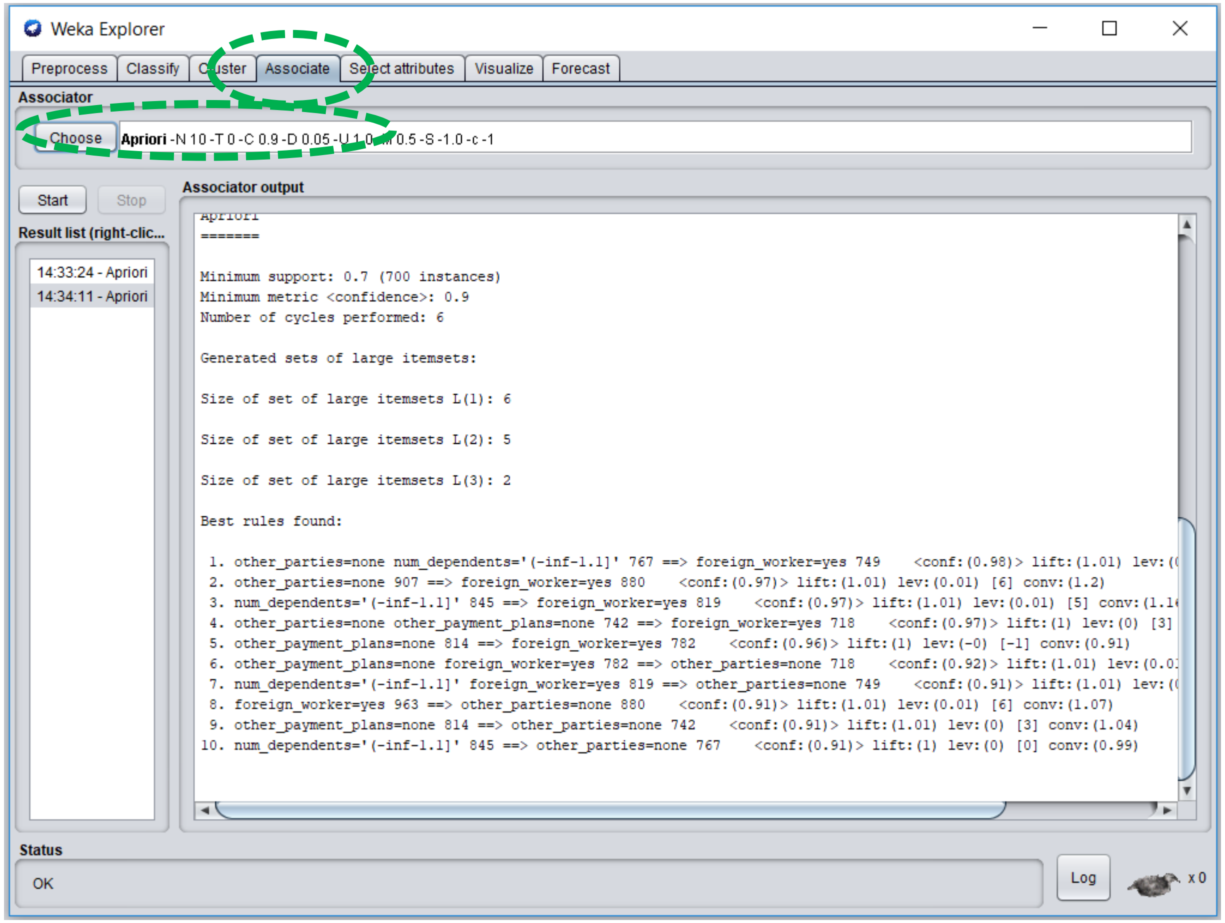
Class: class (Nom)

Status

OK

Association Rule Mining

- Weka provides an implementation of the Apriori algorithm



Associator

Apriori -N 10 -T 0 -C 0.9 -D 0.05 -U 1.0 -M 0.5 -S -1.0 -c -1

Result list (right-clic...

14:33:24 - Apriori
14:34:11 - Apriori

Associator output

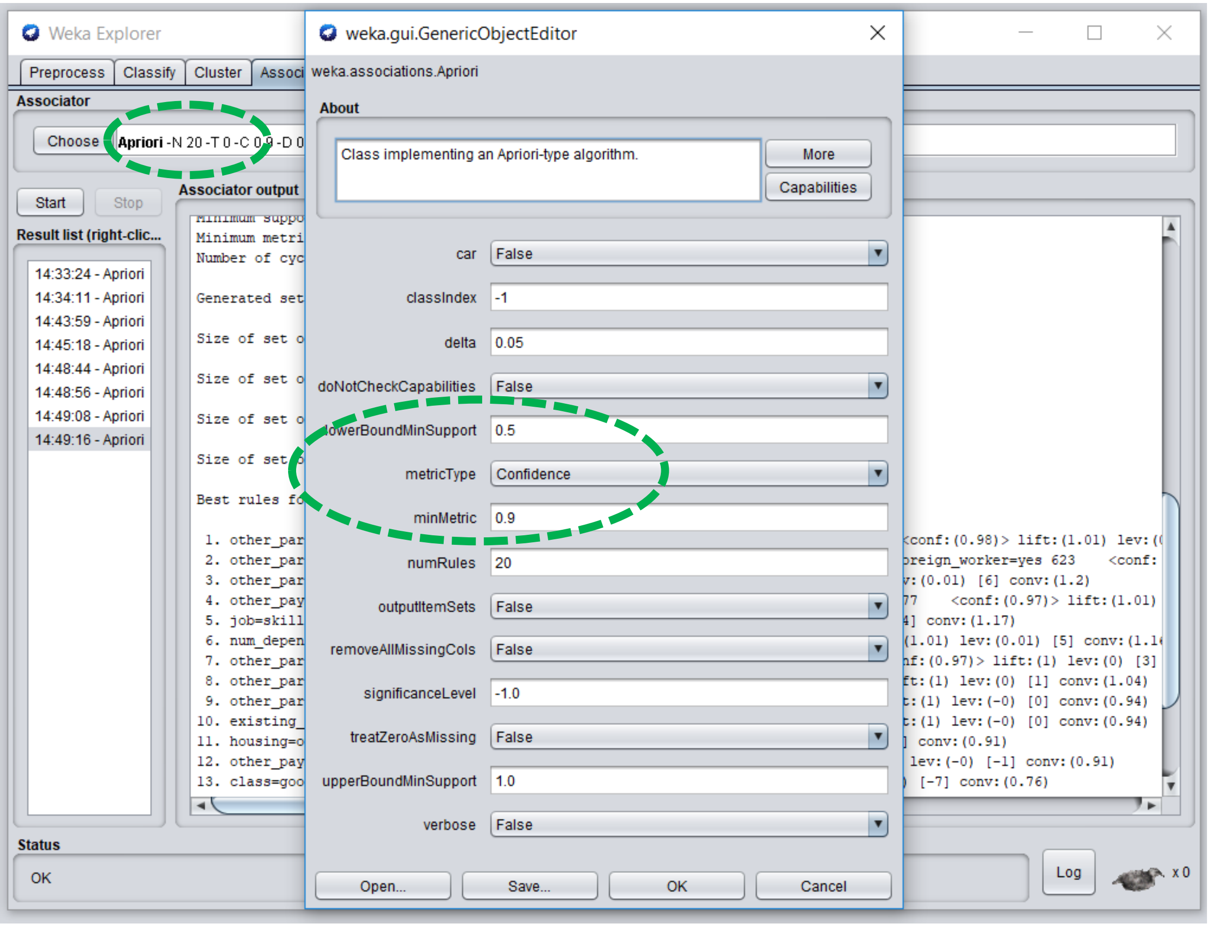
```
Apriori
=======

Minimum support: 0.7 (700 instances)
Minimum metric <confidence>: 0.9
Number of cycles performed: 6

Generated sets of large itemsets:

Size of set of large itemsets L(1): 6

Size of set of large itemsets L(2): 5

Size of set of large itemsets L(3): 2

Best rules found:

 1. other_parties=none num_dependents='(-inf-1.1]' 767 ==> foreign_worker=yes 749    <conf:(0.98)> lift:(1.01) lev:(
 2. other_parties=none 907 ==> foreign_worker=yes 880    <conf:(0.97)> lift:(1.01) lev:(0.01) [6] conv:(1.2)
 3. num_dependents='(-inf-1.1]' 845 ==> foreign_worker=yes 819    <conf:(0.97)> lift:(1.01) lev:(0.01) [5] conv:(1.1
 4. other_parties=none other_payment_plans=none 742 ==> foreign_worker=yes 718    <conf:(0.97)> lift:(1) lev:(0) [3]
 5. other_payment_plans=none 814 ==> foreign_worker=yes 782    <conf:(0.96)> lift:(1) lev:(-0) [-1] conv:(0.91)
 6. other_payment_plans=none foreign_worker=yes 782 ==> other_parties=none 718    <conf:(0.92)> lift:(1.01) lev:(0.0
 7. num_dependents='(-inf-1.1]' foreign_worker=yes 819 ==> other_parties=none 749    <conf:(0.91)> lift:(1.01) lev:(
 8. foreign_worker=yes 963 ==> other_parties=none 880    <conf:(0.91)> lift:(1.01) lev:(0.01) [6] conv:(1.07)
 9. other_payment_plans=none 814 ==> other_parties=none 742    <conf:(0.91)> lift:(1.01) lev:(0) [3] conv:(1.04)
10. num_dependents='(-inf-1.1]' 845 ==> other_parties=none 767    <conf:(0.91)> lift:(1) lev:(0) [0] conv:(0.99)
```

Status

OK

Association Rule Mining

- Weka provides an implementation of the Apriori algorithm
    - Options to set various parameters, e.g., minSup and minConf thresholds



Apriori -N 20 -T 0 -C 0.9 -D 0

14:33:24 - Apriori
14:34:11 - Apriori
14:43:59 - Apriori
14:45:18 - Apriori
14:48:44 - Apriori
14:48:56 - Apriori
14:49:08 - Apriori
14:49:16 - Apriori

```
Minimum suppo
Minimum metri
Number of cyc

Generated set

Size of set o

Size of set o

Size of set o

Size of set o

Best rules fo

 1. other_par
 2. other_par
 3. other_par
 4. other_pay
 5. job=skill
 6. num_depen
 7. other_par
 8. other_par
 9. other_par
10. existing_
11. housing=o
12. other_pay
13. class=goo
```

```
<conf:(0.98)> lift:(1.01) lev:(
oreign_worker=yes 623    <conf:
v:(0.01) [6] conv:(1.2)
77    <conf:(0.97)> lift:(1.01)
4] conv:(1.17)
(1.01) lev:(0.01) [5] conv:(1.1
nf:(0.97)> lift:(1) lev:(0) [3]
ft:(1) lev:(0) [1] conv:(1.04)
t:(1) lev:(-0) [0] conv:(0.94)
t:(1) lev:(-0) [0] conv:(0.94)
] conv:(0.91)
 lev:(-0) [-1] conv:(0.91)
) [-7] conv:(0.76)
```

Status: OK

weka.associations.Apriori

About: Class implementing an Apriori-type algorithm.

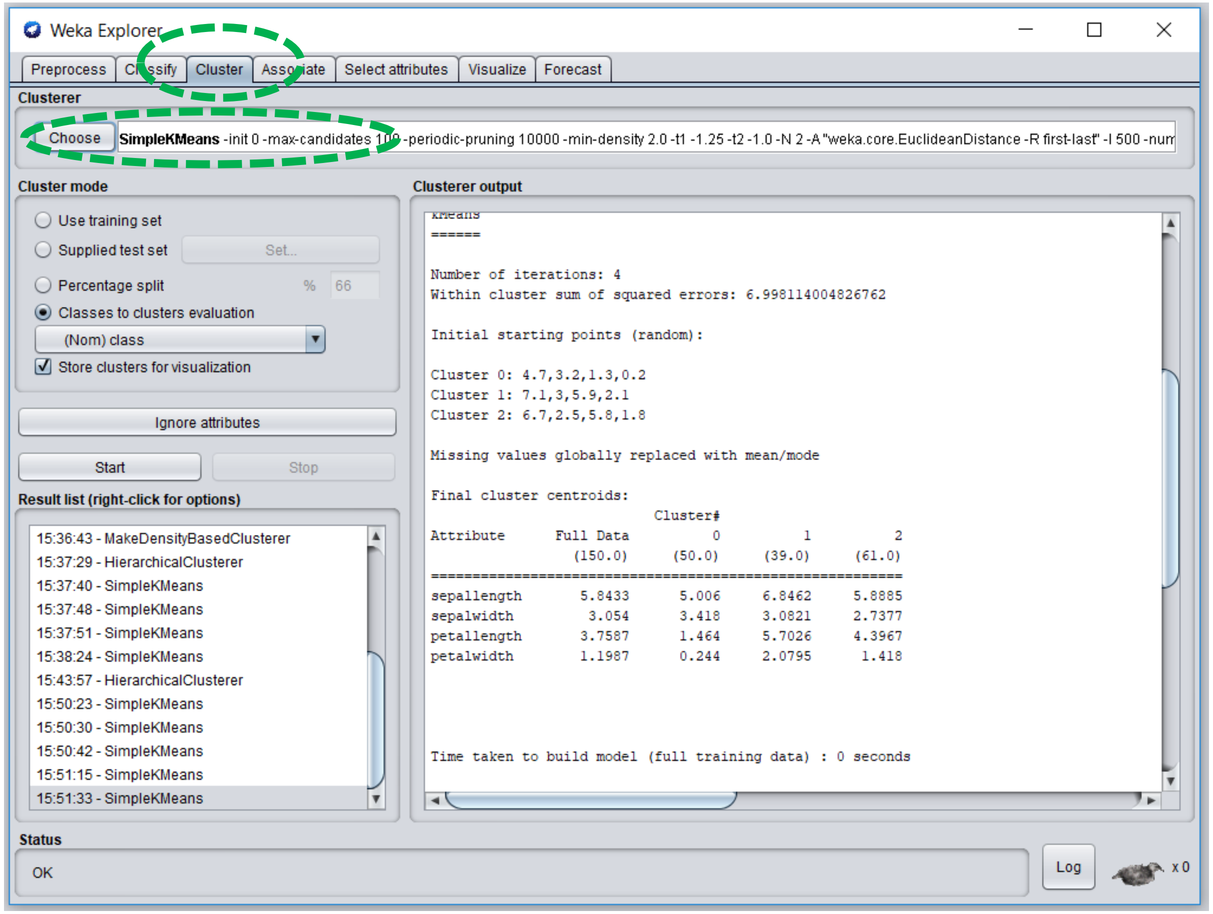
car: False
classIndex: -1
delta: 0.05
doNotCheckCapabilities: False
lowerBoundMinSupport: 0.5
metricType: Confidence
minMetric: 0.9
numRules: 20
outputItemSets: False
removeAllMissingCols: False
significanceLevel: -1.0
treatZeroAsMissing: False
upperBoundMinSupport: 1.0
verbose: False

Exercise

1. Load in the ./weka-3.8/data/credit-g.arff dataset
2. What types of features are in the dataset?
   norminal, numeric
3. How should you pre-process the dataset before applying association rule mining? discretize
4. With a minSup=0.8 threshold, identify the top 10 association rules (based on confidence scores)?
   - What do you observe? How can you obtain the top 10 rules?
5. Now load in the ./weka-3.8/data/supermarket.arff dataset
   - With a minSup=0.5 threshold, what are the frequent 2-itemsets?
   - What are the top 3 association rules based on confidence?

Clustering

- Weka provides implementations of various clustering algorithms, including k-means and hierarchical



Clusterer

SimpleKMeans -init 0 -max-candidates 100 -periodic-pruning 10000 -min-density 2.0 -t1 -1.25 -t2 -1.0 -N 2 -A "weka.core.EuclideanDistance -R first-last" -I 500 -num

Cluster mode

○ Use training set

○ Supplied test set

○ Percentage split % 66

● Classes to clusters evaluation

(Nom) class

☑ Store clusters for visualization

Result list (right-click for options)

15:36:43 - MakeDensityBasedClusterer
15:37:29 - HierarchicalClusterer
15:37:40 - SimpleKMeans
15:37:48 - SimpleKMeans
15:37:51 - SimpleKMeans
15:38:24 - SimpleKMeans
15:43:57 - HierarchicalClusterer
15:50:23 - SimpleKMeans
15:50:30 - SimpleKMeans
15:50:42 - SimpleKMeans
15:51:15 - SimpleKMeans
15:51:33 - SimpleKMeans

Clusterer output

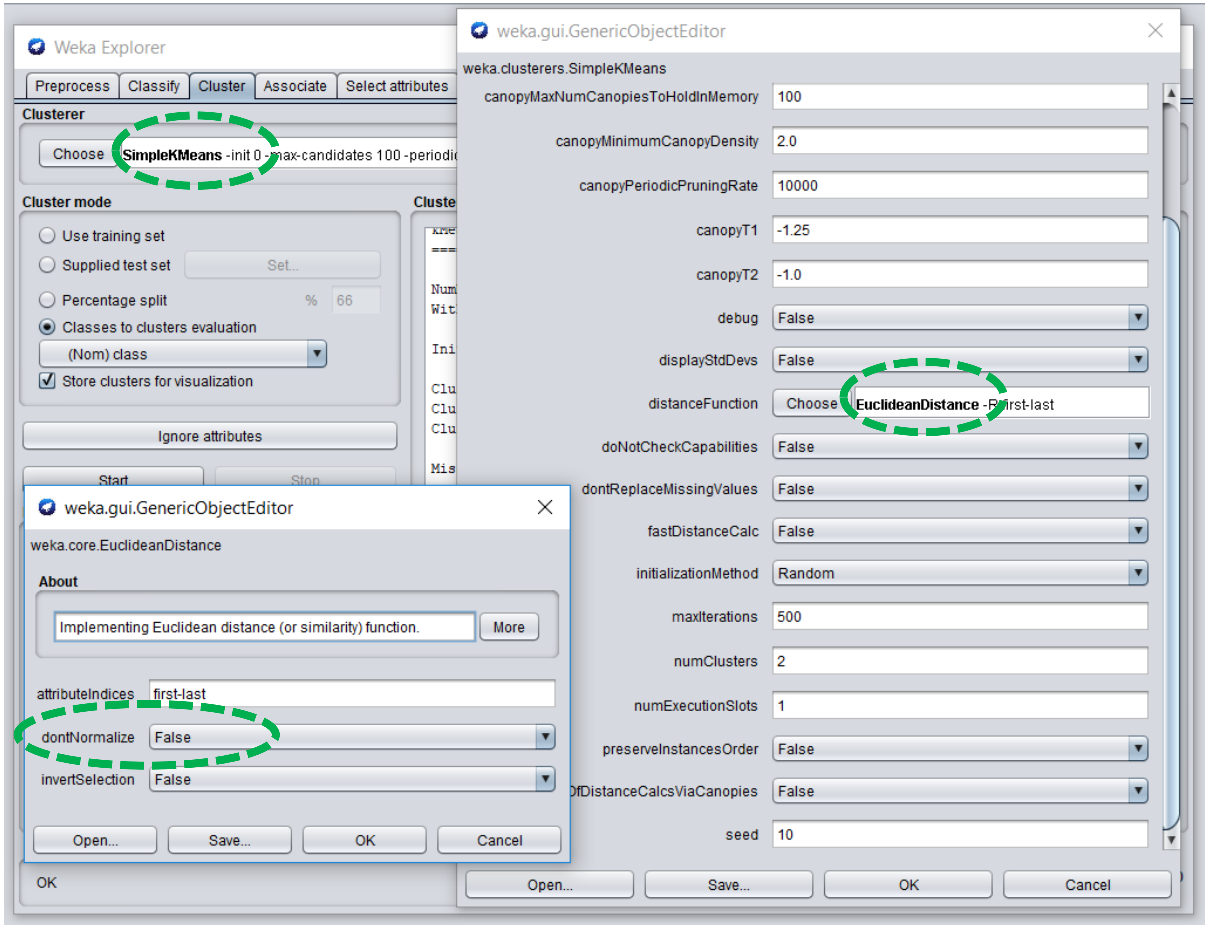
```
kMeans
======

Number of iterations: 4
Within cluster sum of squared errors: 6.998114004826762

Initial starting points (random):

Cluster 0: 4.7,3.2,1.3,0.2
Cluster 1: 7.1,3,5.9,2.1
Cluster 2: 6.7,2.5,5.8,1.8

Missing values globally replaced with mean/mode

Final cluster centroids:
                         Cluster#
Attribute    Full Data          0          1          2
               (150.0)     (50.0)     (39.0)     (61.0)
=========================================================
sepallength     5.8433      5.006     6.8462     5.8885
sepalwidth       3.054      3.418     3.0821     2.7377
petallength     3.7587      1.464     5.7026     4.3967
petalwidth      1.1987      0.244     2.0795      1.418




Time taken to build model (full training data) : 0 seconds
```

Status

OK

Clustering

- Weka provides implementations of various clustering algorithms, including k-means and hierarchical
  - Able to fine-tune k-means in various ways, e.g., select distance measure, set seeds, feature normalization, set k-value, etc



Weka Explorer

Clusterer

Choose | **SimpleKMeans** -init 0 -max-candidates 100 -periodi

Cluster mode

○ Use training set
○ Supplied test set Set...
○ Percentage split % 66
● Classes to clusters evaluation
(Nom) class
☑ Store clusters for visualization

OK

weka.gui.GenericObjectEditor

weka.clusterers.SimpleKMeans

canopyMaxNumCanopiesToHoldInMemory	100
canopyMinimumCanopyDensity	2.0
canopyPeriodicPruningRate	10000
canopyT1	-1.25
canopyT2	-1.0
debug	False
displayStdDevs	False
distanceFunction	Choose **EuclideanDistance** -R first-last
doNotCheckCapabilities	False
dontReplaceMissingValues	False
fastDistanceCalc	False
initializationMethod	Random
maxIterations	500
numClusters	2
numExecutionSlots	1
preserveInstancesOrder	False
fDistanceCalcsViaCanopies	False
seed	10

weka.gui.GenericObjectEditor

weka.core.EuclideanDistance

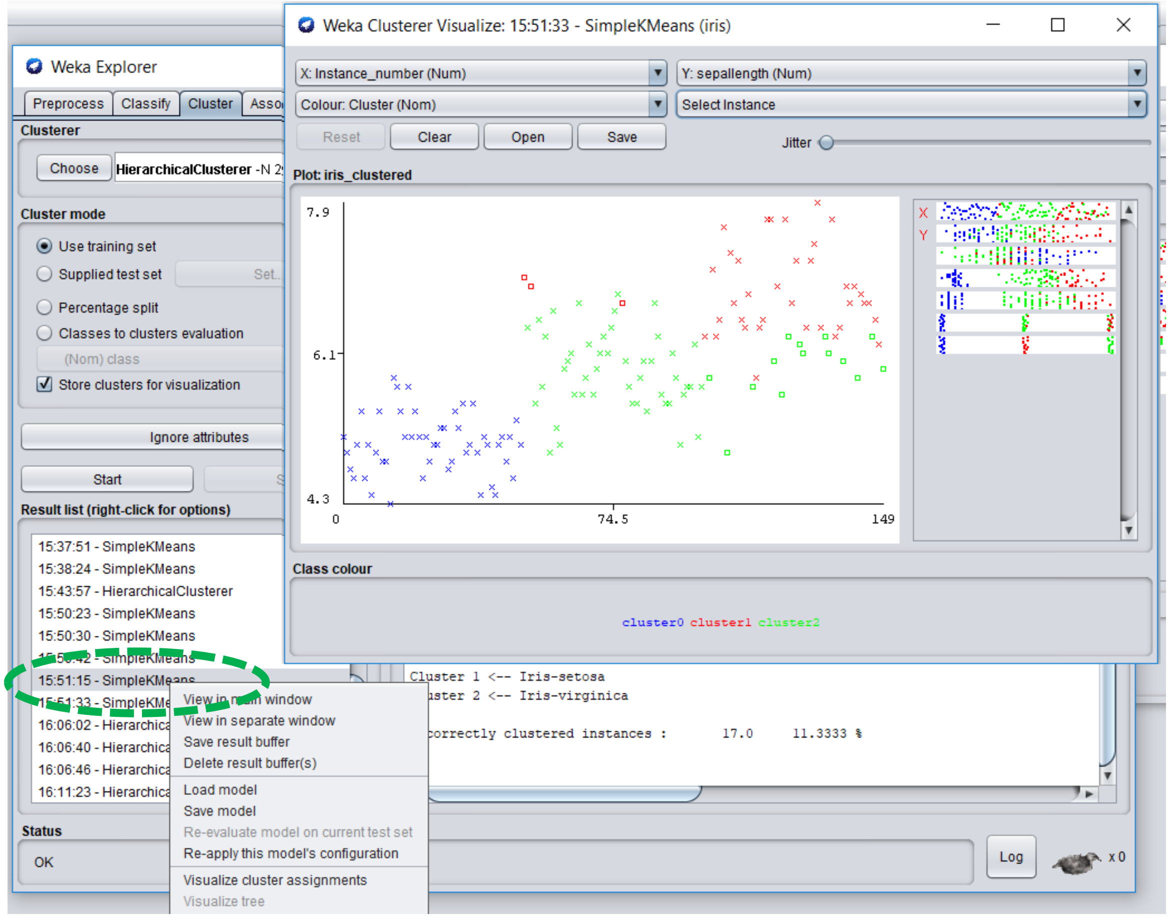
About

Implementing Euclidean distance (or similarity) function.

attributeIndices	first-last
dontNormalize	False
invertSelection	False

Clustering

- Able to visualize cluster assignments based on different features
  - Right-click on “result list”, and select “visualize cluster assignment”



```
Cluster 1 <-- Iris-setosa
uster 2 <-- Iris-virginica

correctly clustered instances :      17.0     11.3333 %
```

Exercise

1. Load in the ./weka-3.8/data/iris.arff dataset
2. Run the k-means (SimpleKMeans) algorithm multiple times with k=3 and observe the sum of squared errors (SSE) values.
   - K-means typically return different clusters with each run, why do you observe in terms of SSE and why is this so?
3. Run k-means again, with feature normalization and without.
   - What do you observe now in terms of SSE?

Project

- Use the rest of the lab to work on your projects

- Presentation during Week 8
  - Allocated time of 10min per group
  - Details on presentation slots available from next week
  - Please ***sign up for a presentation slot***