Clustering using WEKA

Dataset

- 3D_spatiao_network.arff
- au1.csv, au2.csv, and au3.csv
- BlackFriday.csv
- highdimensional.csv

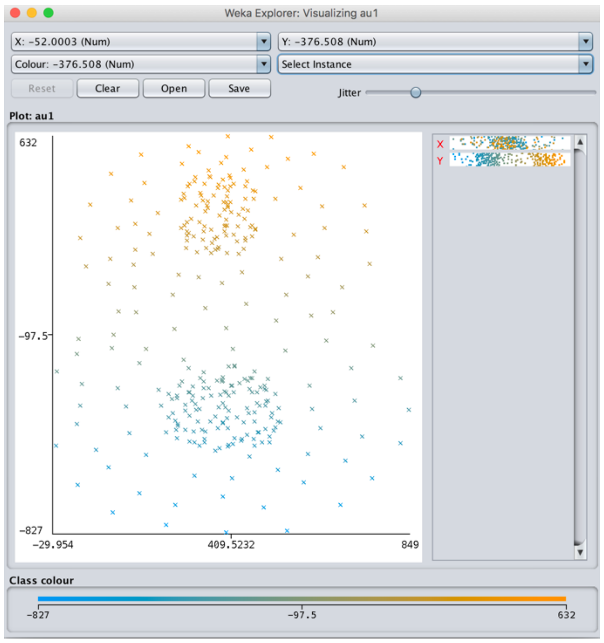
Task

1. Clustering with DBSCAN (au1.csv and au2.csv)
2. Clustering result comparison (au1.csv)
3. DBSCAN with high dimensional data (highdimensional.csv)
4. Hierarchical clustering
5. More hierarchical clustering practice

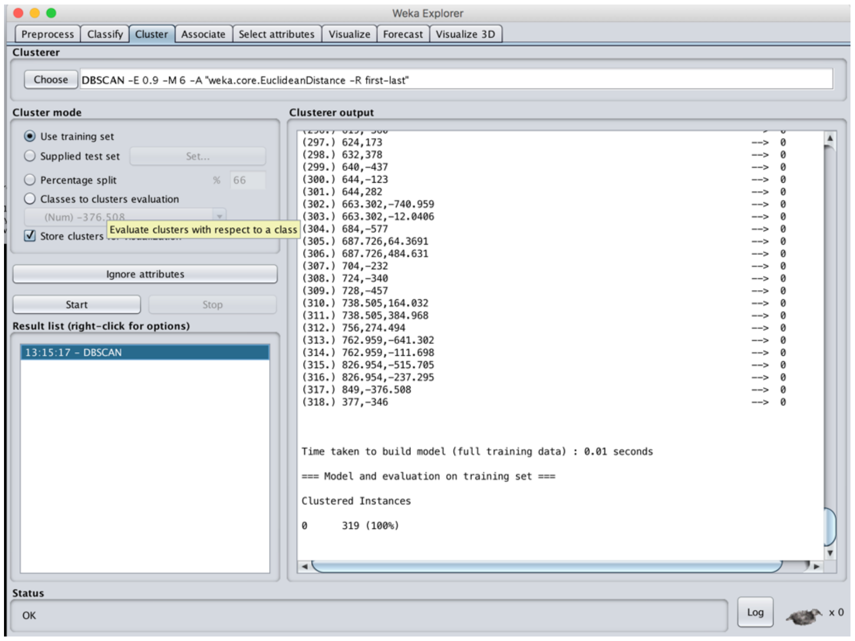
Task 1: Clustering with DBSCAN

Step 1: if you do not have DBSCAN and OPTICS installed in your Weka, then please go to **Weka GUI Choose->Tools->Package Manager** and search and install **optics_dbScan**

Step 2: Load au1.csv data into WEKA.
The data has two dense clusters and a sparse cluster (could be seen as noise) around these two dense clusters as below.



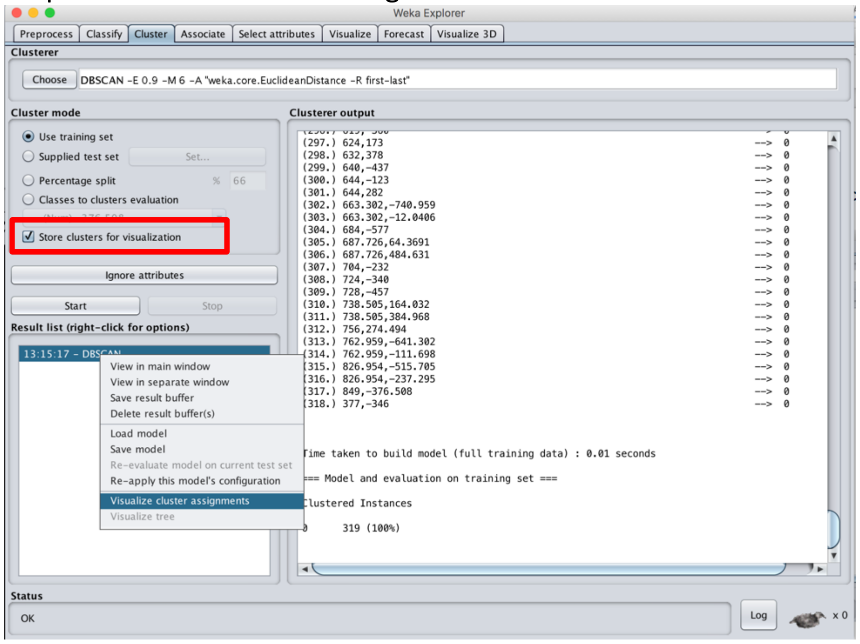
Step 3: Run DBSCAN with a default value. Check 'Store clusters for visualization' to see the clustering result.



DBSCAN -E 0.9 -M 6 -A "weka.core.EuclideanDistance -R first-last"

```
(297.) 624,173                                                              -->  0
(298.) 632,378                                                              -->  0
(299.) 640,-437                                                             -->  0
(300.) 644,-123                                                             -->  0
(301.) 644,282                                                              -->  0
(302.) 663.302,-740.959                                                     -->  0
(303.) 663.302,-12.0406                                                     -->  0
(304.) 684,-577                                                             -->  0
(305.) 687.726,64.3691                                                      -->  0
(306.) 687.726,484.631                                                      -->  0
(307.) 704,-232                                                             -->  0
(308.) 724,-340                                                             -->  0
(309.) 728,-457                                                             -->  0
(310.) 738.505,164.032                                                      -->  0
(311.) 738.505,384.968                                                      -->  0
(312.) 756,274.494                                                          -->  0
(313.) 762.959,-641.302                                                     -->  0
(314.) 762.959,-111.698                                                     -->  0
(315.) 826.954,-515.705                                                     -->  0
(316.) 826.954,-237.295                                                     -->  0
(317.) 849,-376.508                                                         -->  0
(318.) 377,-346                                                             -->  0


Time taken to build model (full training data) : 0.01 seconds

=== Model and evaluation on training set ===

Clustered Instances

0      319 (100%)
```

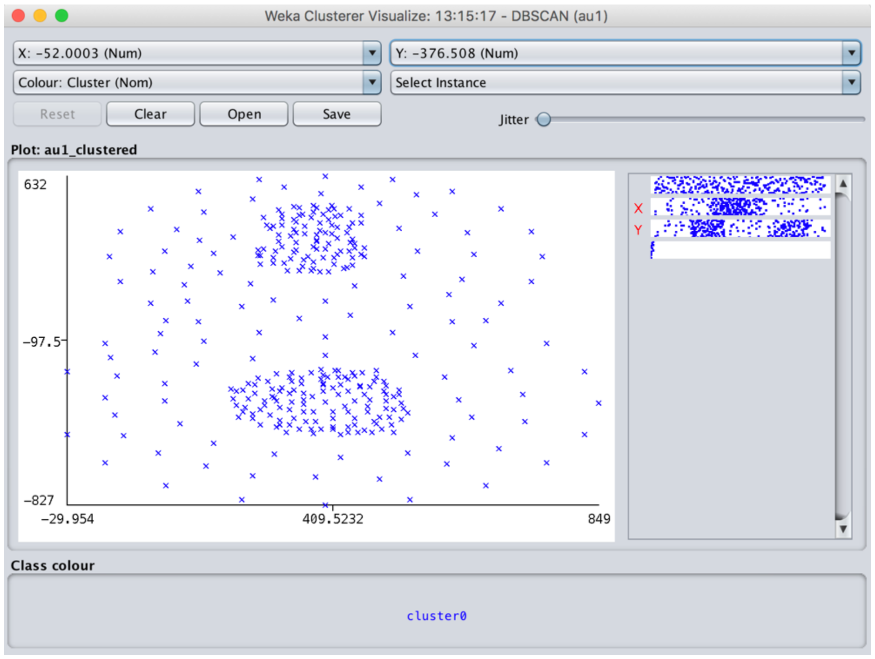
Step 4: Visualise the clustering result.



DBSCAN -E 0.9 -M 6 -A "weka.core.EuclideanDistance -R first-last"

```
(297.) 624,173                                                              -->  0
(298.) 632,378                                                              -->  0
(299.) 640,-437                                                             -->  0
(300.) 644,-123                                                             -->  0
(301.) 644,282                                                              -->  0
(302.) 663.302,-740.959                                                     -->  0
(303.) 663.302,-12.0406                                                     -->  0
(304.) 684,-577                                                             -->  0
(305.) 687.726,64.3691                                                      -->  0
(306.) 687.726,484.631                                                      -->  0
(307.) 704,-232                                                             -->  0
(308.) 724,-340                                                             -->  0
(309.) 728,-457                                                             -->  0
(310.) 738.505,164.032                                                      -->  0
(311.) 738.505,384.968                                                      -->  0
(312.) 756,274.494                                                          -->  0
(313.) 762.959,-641.302                                                     -->  0
(314.) 762.959,-111.698                                                     -->  0
(315.) 826.954,-515.705                                                     -->  0
(316.) 826.954,-237.295                                                     -->  0
(317.) 849,-376.508                                                         -->  0
(318.) 377,-346                                                             -->  0


Time taken to build model (full training data) : 0.01 seconds

=== Model and evaluation on training set ===

Clustered Instances

0      319 (100%)
```

Visualisation shows one cluster with everything as below.


Weka Clusterer Visualize: 13:15:17 - DBSCAN (au1)

X: -52.0003 (Num)

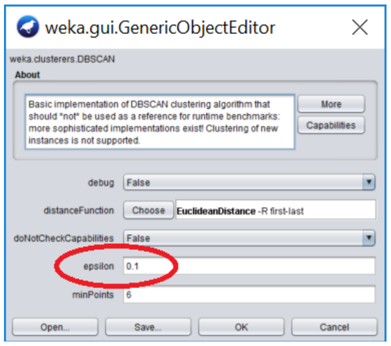
Y: -376.508 (Num)

Colour: Cluster (Nom)

Plot: au1_clustered

Class colour: cluster0

DBScan has clustered everything together into one cluster due to epsilon being too large. Change epsilon distance to 0.1 and run again.


weka.gui.GenericObjectEditor

weka.clusterers.DBSCAN

About

Basic implementation of DBSCAN clustering algorithm that should *not* be used as a reference for runtime benchmarks: more sophisticated implementations exist! Clustering of new instances is not supported.

debug: False

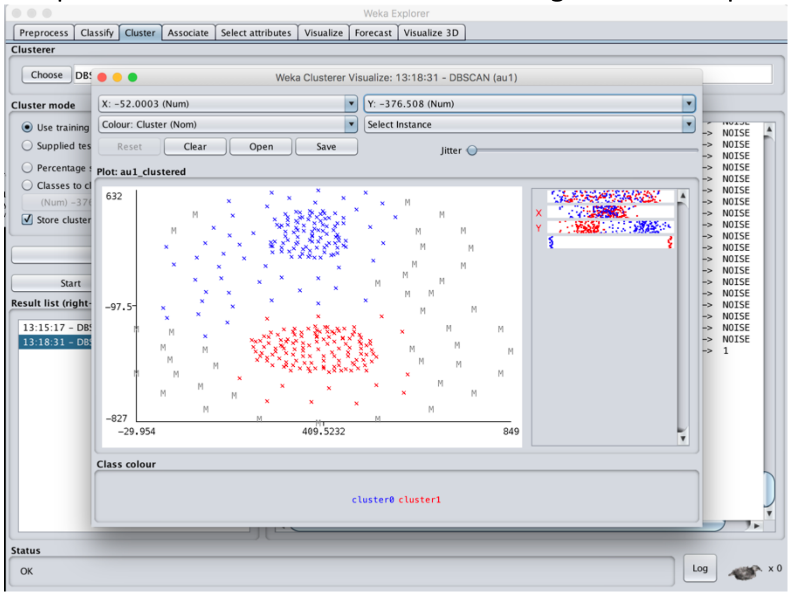
distanceFunction: **EuclideanDistance** -R first-last

doNotCheckCapabilities: False

epsilon: 0.1

minPoints: 6

Now the data has been split into 2 classes and we can see the ignored noise points marked as M.


Weka Explorer

Weka Clusterer Visualize: 13:18:31 - DBSCAN (au1)

X: -52.0003 (Num)

Y: -376.508 (Num)

Colour: Cluster (Nom)

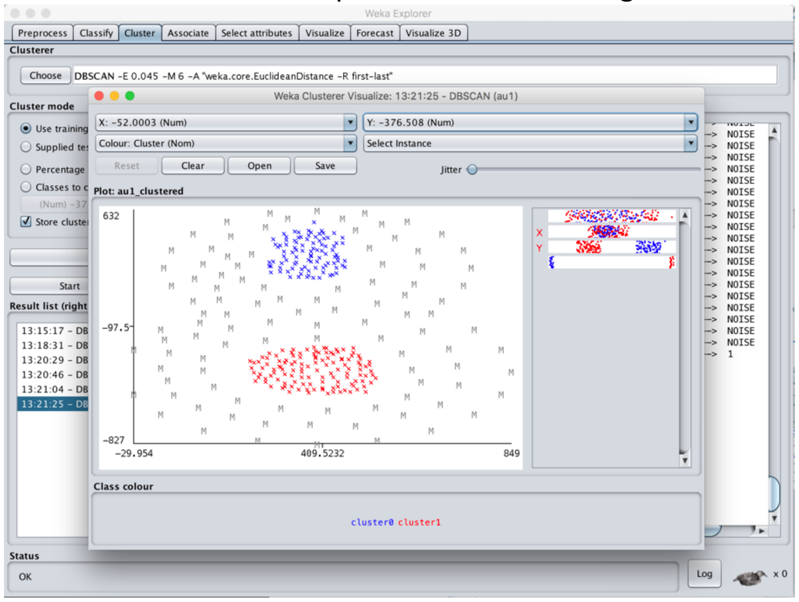
Plot: au1_clustered

Class colour: cluster0 cluster1

```
-> NOISE
-> NOISE
-> NOISE
-> NOISE
-> NOISE
-> NOISE
-> NOISE
-> NOISE
-> NOISE
-> NOISE
-> NOISE
-> NOISE
-> NOISE
-> NOISE
-> NOISE
-> NOISE
-> NOISE
-> NOISE
-> NOISE
-> 1
```

Status: OK

Step 5: But still the clustering is not perfect. Try tweaking the epsilon parameter to detect better clusters. Epsilon value = 0.045 and minPts = 6 produces the following result.



DBSCAN -E 0.045 -M 6 -A "weka.core.EuclideanDistance -R first-last"

Weka Clusterer Visualize: 13:21:25 - DBSCAN (au1)

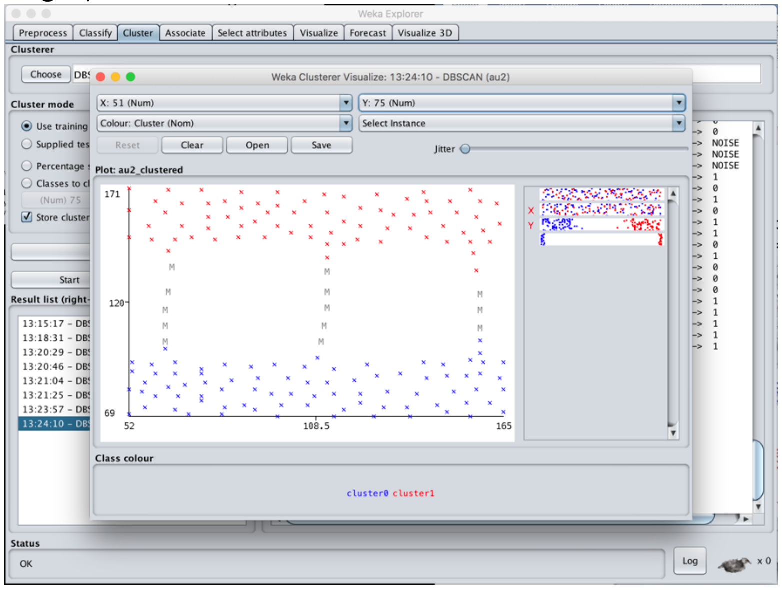
X: -52.0003 (Num)
Y: -376.508 (Num)
Colour: Cluster (Nom)
Select Instance

Plot: au1_clustered

Result list: 13:15:17 - DB, 13:18:31 - DB, 13:20:29 - DB, 13:20:46 - DB, 13:21:04 - DB, 13:21:25 - DB

Status: OK

Step 6: load au2.csv, and try to tweak the epsilon parameter to find the following two clusters connected by links (bridges).



Weka Clusterer Visualize: 13:24:10 - DBSCAN (au2)

X: 51 (Num)
Y: 75 (Num)
Colour: Cluster (Nom)
Select Instance

Plot: au2_clustered

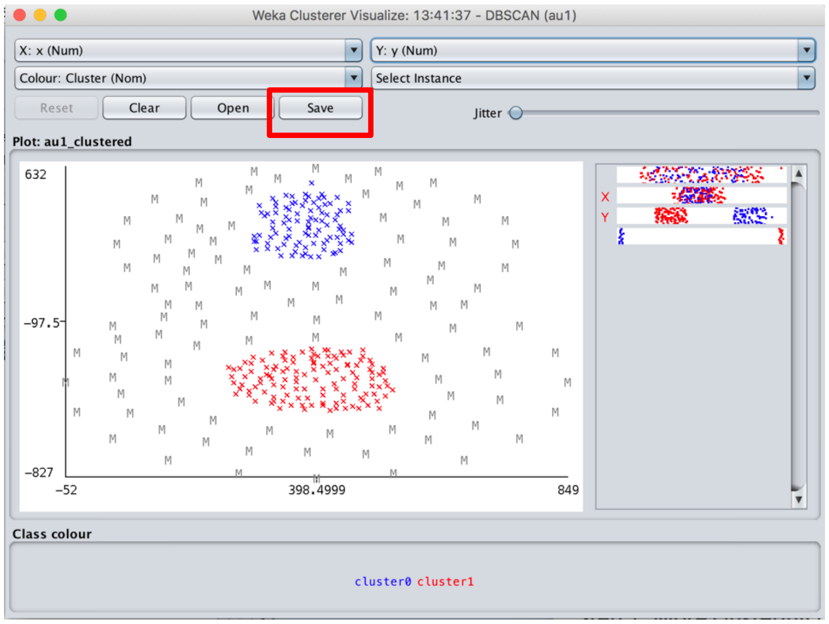
Result list: 13:15:17 - DB, 13:18:31 - DB, 13:20:29 - DB, 13:20:46 - DB, 13:21:04 - DB, 13:21:25 - DB, 13:23:57 - DB, 13:24:10 - DB

Status: OK

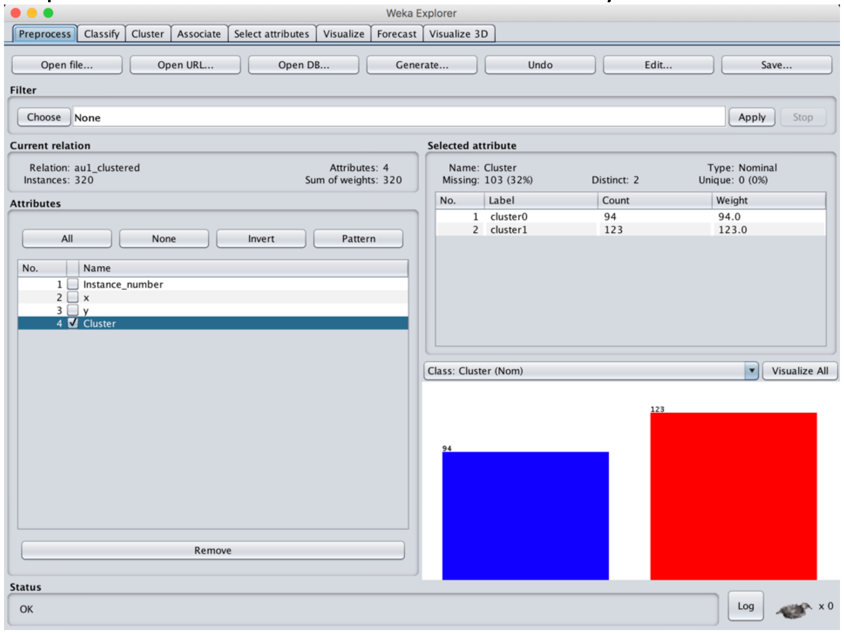
Task 2: Clustering result comparison

You may wish to store the clustering result to the file.

Step 1: You can save it when the clustering result is visualised.



Step 2: Name the file au1-dbscan.arff.
Step 3: Load au1-dbscan.arff into WEKA. Now you will see the data with 4 attributes (ID, x, y, Cluster).



Filter: None

Current relation
Relation: au1_clustered
Instances: 320
Attributes: 4
Sum of weights: 320

Selected attribute
Name: Cluster
Missing: 103 (32%)
Distinct: 2
Type: Nominal
Unique: 0 (0%)

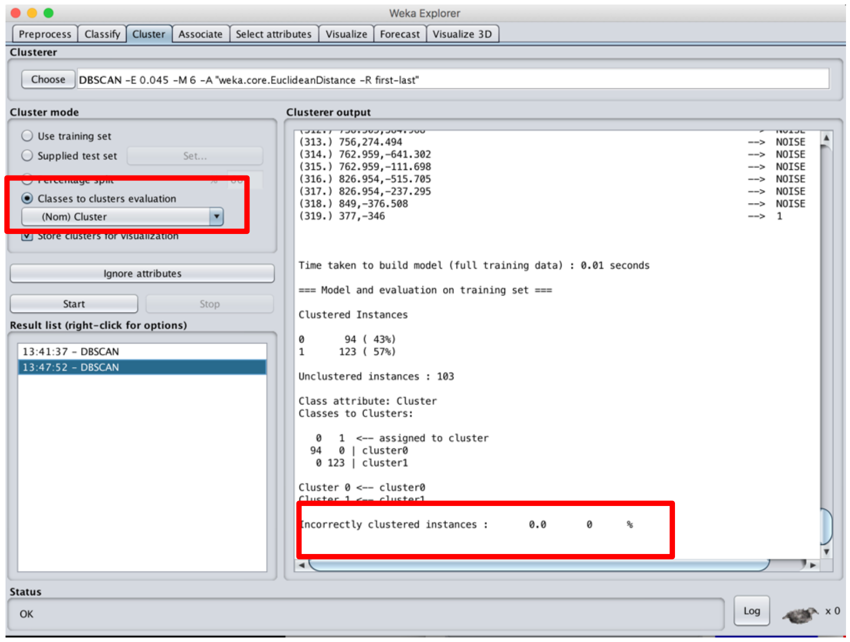
No.	Label	Count	Weight
1	cluster0	94	94.0
2	cluster1	123	123.0

Attributes

No.		Name
1	☐	Instance_number
2	☐	x
3	☐	y
4	☑	Cluster

Status
OK

Step 5: remove ‘Instance_number’ attribute.
Step 6: Cluster using DBSCAN (epison = 0.045 and minPts = 6), and compare this clustering result with ‘Cluster’ attribute in the file. Then not surprisingly, you will see it is 100% accurate.



DBSCAN -E 0.045 -M 6 -A "weka.core.EuclideanDistance -R first-last"

Cluster mode: Classes to clusters evaluation; (Nom) Cluster; Store clusters for visualization

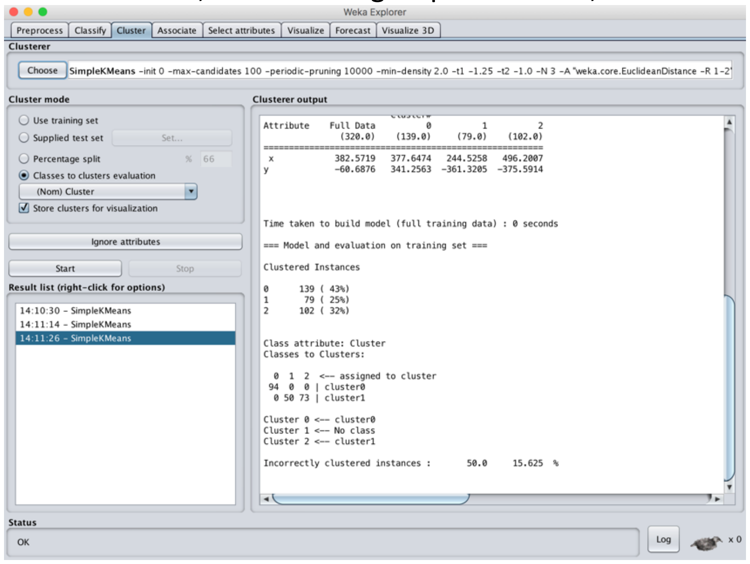
Result list (right-click for options)

13:41:37 - DBSCAN
13:47:52 - DBSCAN

```
(313.) 756,274.494                                              -->  NOISE
(314.) 762.959,-641.302                                         -->  NOISE
(315.) 762.959,-111.698                                         -->  NOISE
(316.) 826.954,-515.705                                         -->  NOISE
(317.) 826.954,-237.295                                         -->  NOISE
(318.) 849,-376.508                                             -->  NOISE
(319.) 377,-346                                                 -->  1


Time taken to build model (full training data) : 0.01 seconds

=== Model and evaluation on training set ===

Clustered Instances

0       94 ( 43%)
1      123 ( 57%)

Unclustered instances : 103

Class attribute: Cluster
Classes to Clusters:

   0   1  <-- assigned to cluster
  94   0 | cluster0
   0 123 | cluster1

Cluster 0 <-- cluster0
Cluster 1 <-- cluster1

Incorrectly clustered instances :       0.0       0      %
```

Status: OK

Step 7: now compare this DBSCAN result with k-means clustering with k=3. Please note that DBSCAN finds 2 clusters, and 1 noise group. Therefore, we can compare it with k-means k=3.



SimpleKMeans -init 0 -max-candidates 100 -periodic-pruning 10000 -min-density 2.0 -t1 -1.25 -t2 -1.0 -N 3 -A "weka.core.EuclideanDistance -R 1-2"

Cluster mode: Classes to clusters evaluation; (Nom) Cluster; Store clusters for visualization

Result list (right-click for options)

14:10:30 - SimpleKMeans
14:11:14 - SimpleKMeans
14:11:26 - SimpleKMeans

```
Attribute    Full Data          0          1          2
               (320.0)    (139.0)     (79.0)    (102.0)
========================================================
 x            382.5719   377.6474   244.5258   496.2007
y             -60.6876   341.2563  -361.3205  -375.5914



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

=== Model and evaluation on training set ===

Clustered Instances

0      139 ( 43%)
1       79 ( 25%)
2      102 ( 32%)


Class attribute: Cluster
Classes to Clusters:

  0  1  2  <-- assigned to cluster
 94  0  0 | cluster0
  0 50 73 | cluster1

Cluster 0 <-- cluster0
Cluster 1 <-- No class
Cluster 2 <-- cluster1

Incorrectly clustered instances :      50.0     15.625  %
```

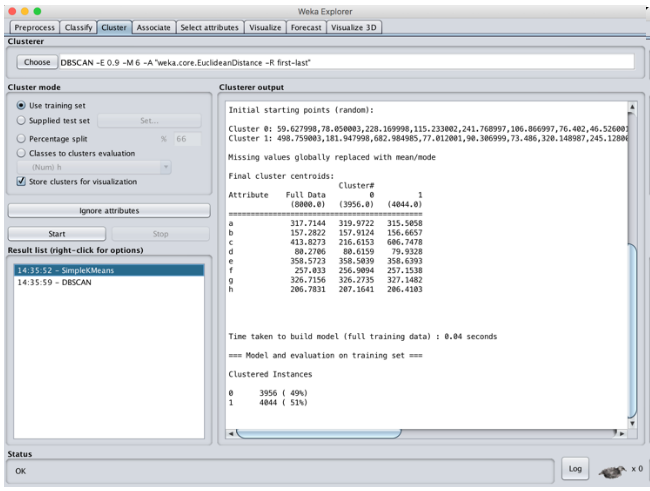
Status: OK

You can see that there is around 15% misclassification. It appears that all those noise points are misclassified in k-means.

Task 3: DBSCAN with high dimensional data

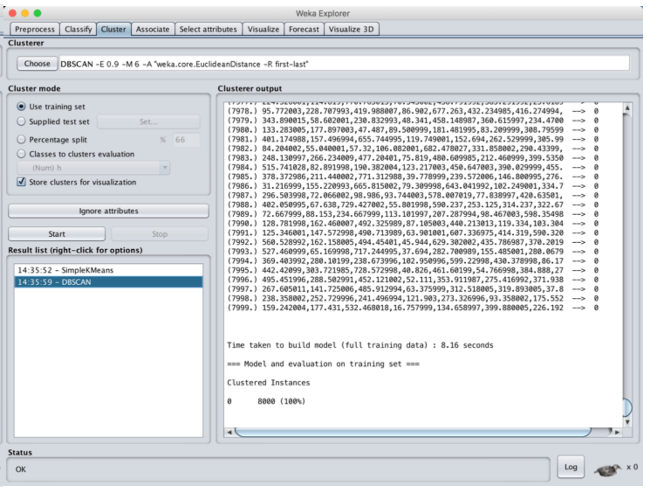
k-means clustering is fast and works well for high dimensional data, however DBSCAN is able to detect quality clusters (arbitrary shapes) but computationally very expensive (slow) for high dimensional data.

Step 1: open highdimensional.csv. The data set has 8000 instances with 8 dimensions.
Step 2: Run k-means and DBSCAN with default values and measure time taken to calculate. As you can see below, it takes 0.04 sec and 8.16 sec in my machine (this would vary with your machine).



DBSCAN -E 0.9 -M 6 -A "weka.core.EuclideanDistance -R first-last"

```
Initial starting points (random):

Cluster 0: 59.627998,78.050003,228.169998,115.233002,241.768997,106.866997,76.402,46.526001
Cluster 1: 498.759003,181.947998,682.984985,77.012001,90.306999,73.486,320.148987,245.12800

Missing values globally replaced with mean/mode

Final cluster centroids:
                         Cluster#
Attribute    Full Data          0          1
              (8000.0)   (3956.0)   (4044.0)
============================================
a             317.7144   319.9722   315.5058
b             157.2822   157.9124   156.6657
c             413.8273   216.6153   606.7478
d              80.2706    80.6159    79.9328
e             358.5723   358.5039   358.6393
f              257.033   256.9094   257.1538
g             326.7156   326.2735   327.1482
h             206.7831   207.1641   206.4103




Time taken to build model (full training data) : 0.04 seconds

=== Model and evaluation on training set ===

Clustered Instances

0      3956 ( 49%)
1      4044 ( 51%)
```



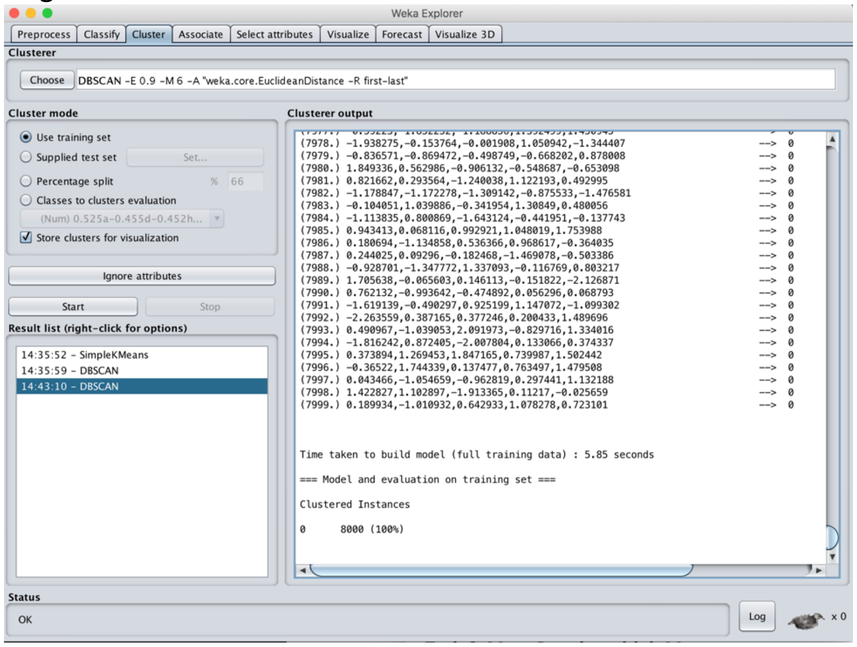
DBSCAN -E 0.9 -M 6 -A "weka.core.EuclideanDistance -R first-last"

```
(7978.) 95.772003,228.707993,419.988007,86.902,677.263,432.234985,416.274994,  --> 0
(7979.) 343.890015,58.602001,230.832993,48.341,458.148987,360.615997,234.4700  --> 0
(7980.) 133.283005,177.897003,47.487,89.500999,181.481995,83.209999,308.79599  --> 0
(7981.) 401.174988,157.496994,655.744995,119.749001,152.694,262.529999,305.99  --> 0
(7982.) 84.204002,55.040001,57.32,106.082001,682.478027,331.858002,290.43399,  --> 0
(7983.) 248.130997,266.234009,477.20401,75.819,480.609985,212.460999,399.5350  --> 0
(7984.) 515.741028,82.891998,190.382004,123.217003,450.647003,390.029999,455.  --> 0
(7985.) 378.372986,211.440002,771.312988,39.778999,239.572006,146.800995,276.  --> 0
(7986.) 31.216999,155.220993,665.815002,79.309998,643.041992,102.249001,334.7  --> 0
(7987.) 296.503998,72.066002,98.986,93.744003,578.007019,77.838997,420.63501,  --> 0
(7988.) 402.050995,67.638,729.427002,55.801998,590.237,253.125,314.237,322.67  --> 0
(7989.) 72.667999,88.153,234.667999,113.101997,207.287994,98.467003,598.35498  --> 0
(7990.) 128.781998,162.460007,492.325989,87.105003,440.213013,119.334,103.304  --> 0
(7991.) 125.346001,147.572998,490.713989,63.901001,607.336975,414.319,590.320  --> 0
(7992.) 560.528992,162.158005,494.45401,45.944,629.302002,435.786987,370.2019  --> 0
(7993.) 527.460999,65.169998,717.244995,37.694,282.700989,155.485001,280.0679  --> 0
(7994.) 369.403992,280.10199,238.673996,102.950996,599.22998,430.378998,86.17  --> 0
(7995.) 442.42099,303.721985,728.572998,40.826,461.60199,54.766998,384.888,27  --> 0
(7996.) 495.451996,288.502991,452.121002,52.111,353.911987,275.416992,371.938  --> 0
(7997.) 267.605011,141.725006,485.912994,63.375999,312.518005,319.893005,37.8  --> 0
(7998.) 238.358002,252.729996,241.496994,121.903,273.326996,93.358002,175.552  --> 0
(7999.) 159.242004,177.431,532.468018,16.757999,134.658997,399.880005,226.192  --> 0


Time taken to build model (full training data) : 8.16 seconds

=== Model and evaluation on training set ===

Clustered Instances

0      8000 (100%)
```

Step 3: Once possible way to handle high dimensional data with DBSCAN, is to use Feature Selection or Feature Reduction (Dimension Reduction) approach. Let's use PCA for this data.

Step 4: with 60% of variable captured, PCA produces 5 principle components.

Step 5: Now run DBSCAN with the same default value. Now it takes 5.85 sec which is faster than the original 8 attributes.



DBSCAN -E 0.9 -M 6 -A "weka.core.EuclideanDistance -R first-last"

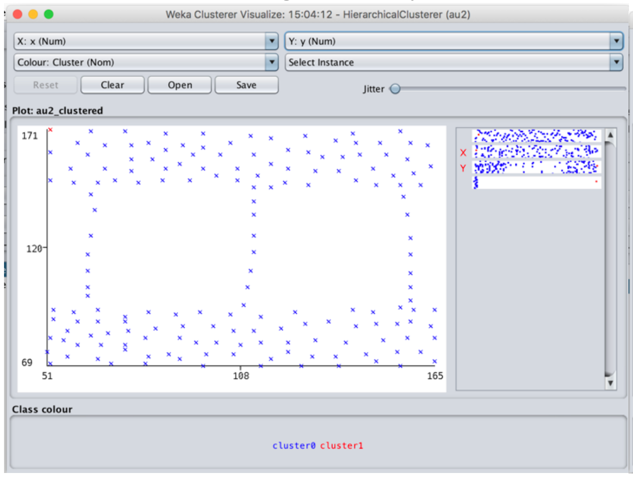
```
(7978.) -1.938275,-0.153764,-0.001908,1.050942,-1.344407        --> 0
(7979.) -0.836571,-0.869472,-0.498749,-0.668202,0.878008        --> 0
(7980.) 1.849336,0.562986,-0.906132,-0.548687,-0.653098         --> 0
(7981.) 0.821662,0.293564,-1.240038,1.122193,0.492995           --> 0
(7982.) -1.178847,-1.172278,-1.309142,-0.875533,-1.476581       --> 0
(7983.) -0.104051,1.039886,-0.341954,1.30849,0.480056           --> 0
(7984.) -1.113835,0.800869,-1.643124,-0.441951,-0.137743        --> 0
(7985.) 0.943413,0.068116,0.992921,1.048019,1.753988            --> 0
(7986.) 0.180694,-1.134858,0.536366,0.968617,-0.364035          --> 0
(7987.) 0.244025,0.09296,-0.182468,-1.469078,-0.503386          --> 0
(7988.) -0.928701,-1.347772,1.337093,-0.116769,0.803217         --> 0
(7989.) 1.705638,-0.065603,0.146113,-0.151822,-2.126871         --> 0
(7990.) 0.762132,-0.993642,-0.474892,0.056296,0.068793          --> 0
(7991.) -1.619139,-0.490297,0.925199,1.147072,-1.099302         --> 0
(7992.) -2.263559,0.387165,0.377246,0.200433,1.489696           --> 0
(7993.) 0.490967,-1.039053,2.091973,-0.829716,1.334016          --> 0
(7994.) -1.816242,0.872405,-2.007804,0.133066,0.374337          --> 0
(7995.) 0.373894,1.269453,1.847165,0.739987,1.502442            --> 0
(7996.) -0.36522,1.744339,0.137477,0.763497,1.479508            --> 0
(7997.) 0.043466,-1.054659,-0.962819,0.297441,1.132188          --> 0
(7998.) 1.422827,1.102897,-1.913365,0.11217,-0.025659           --> 0
(7999.) 0.189934,-1.010932,0.642933,1.078278,0.723101           --> 0


Time taken to build model (full training data) : 5.85 seconds

=== Model and evaluation on training set ===

Clustered Instances

0      8000 (100%)
```

Task 4: Hierarchical clustering with WEKA

Bottom-up, agglomerative, hierarchical clustering is a popular approach to generate a clustering hierarchy from the dataset. There are many different types of merge techniques including single-linkage, complete-linkage, mean-linkage, and average-linkage. As you can easily notice, hierarchical clustering is more computationally expensive (slower) than partitioning clustering, so it would take longer time with larger data.

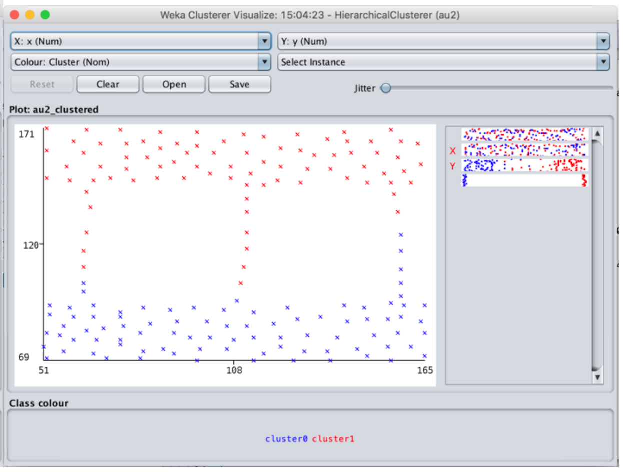
Step 1: load au2.csv into WEKA.

Step 2: Run Hierarchical clustering with the number of clusters = 2. Try to observe the result with various different merge techniques. Which one is able to detect the two clusters?


Weka Clusterer Visualize: 15:04:12 - HierarchicalClusterer (au2)

X: x (Num)
Y: y (Num)
Colour: Cluster (Nom)
Select Instance

Plot: au2_clustered


Weka Clusterer Visualize: 15:04:23 - HierarchicalClusterer (au2)

X: x (Num)
Y: y (Num)
Colour: Cluster (Nom)
Select Instance

Plot: au2_clustered

Task 5: More Practice with DBSCAN and Hierarchical clustering

More clustering practice with 3D_spatial_network and BlackFriday.csv data with DBSCAN and hierarchical clustering. You might need to do preprocessing before you are able to use them.

This is the end of this week’s prac.