Lab Exercise Four

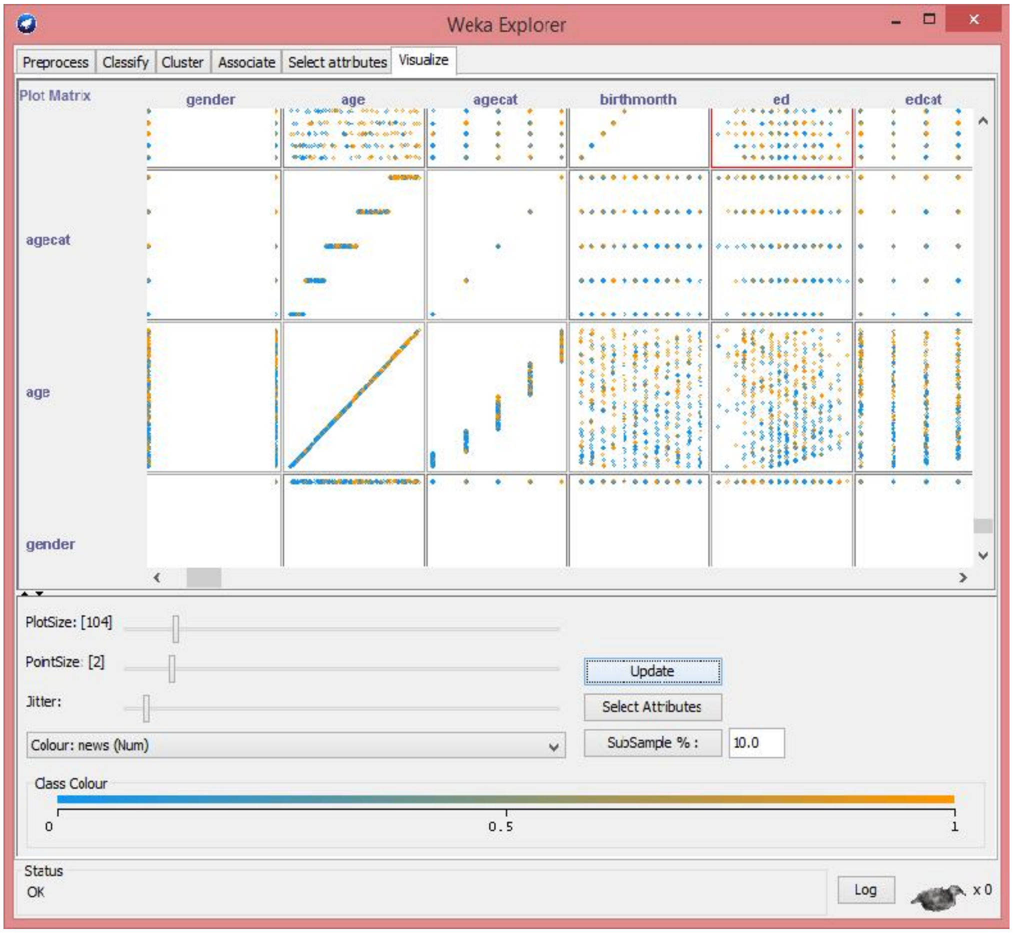
Clustering with WEKA Explorer

1. Open a terminal window from the left bar. Go to directory /opt/weka-3-6-13, then type command :
   java –jar weka.jar.

2. Fire up WEKA to get the GUI Chooser panel. Select Explorer from the four choices on the right side.

3. We are on ***Preprocess*** now. Click the ***Open file*** button to bring up a standard dialog through which you can select a file. Choose the **telco_labFour.csv** file.

4. You could ignore irrelevant attributes during the clustering process, like **custIds**. To identify redundant attributes, we could check the correlation from Visualization of the data set under Visualize Tab. ***age*** and ***agecat*** are correlated. One of them should be ignored. We keep ***age*** for clustering purpose; also ***ed*** (removing ***edcat***), then we have 8 attributes left for clustering (we will ignore ***custIds***, ***agecat*** and ***edcat*** when we perform clustering.



PlotSize: [104]

PointSize: [2]

Jitter:

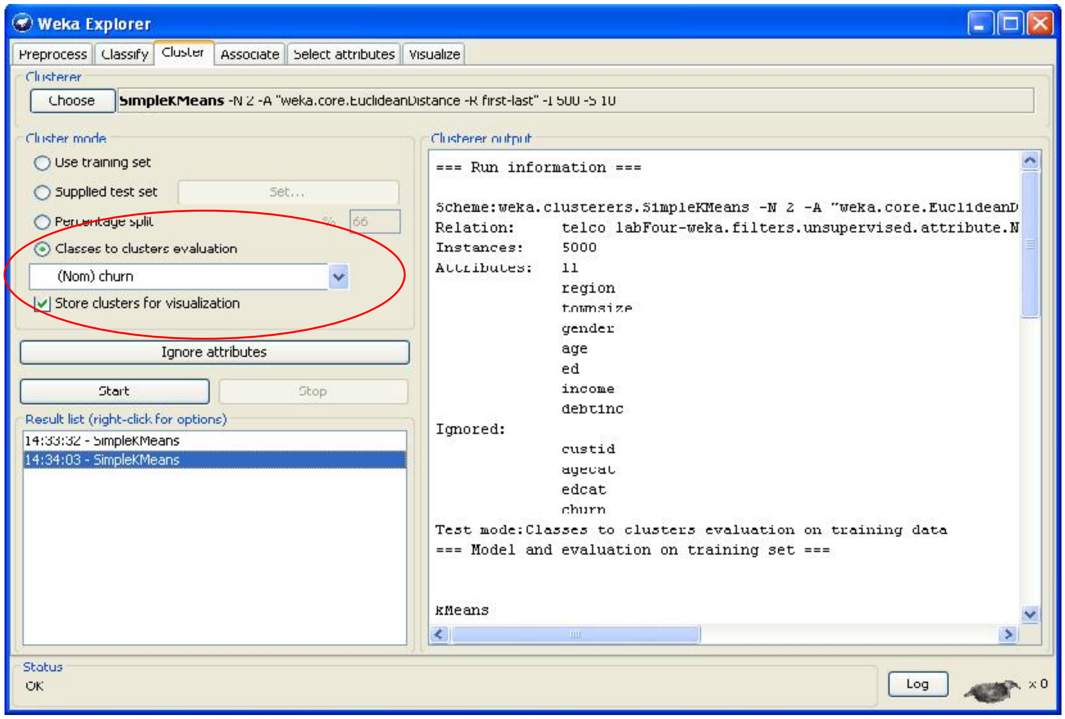
Colour: news (Num)

SubSample % : 10.0

OK

5. Before we do clustering with **Weka**, we need to normalize your numeric data values (use **Normalize** filter). Since we have the class label, we would like to set it to nominal before normalization. This information will be used to evaluate the clustering performance.

6. To perform clustering on the data set, click ***Cluster*** tab and choose ***SimpleKMeans*** algorithm. We set **k = 2** for this data set. Choose ***Classes to clusters evaluation*** and select the last attribute as class label. Check *Store clusters for visualization*. Click ***Ignore attributes*** and select ***custIds, agecat, edcat***, and the last attribute ***churn***. Then click **Start**.



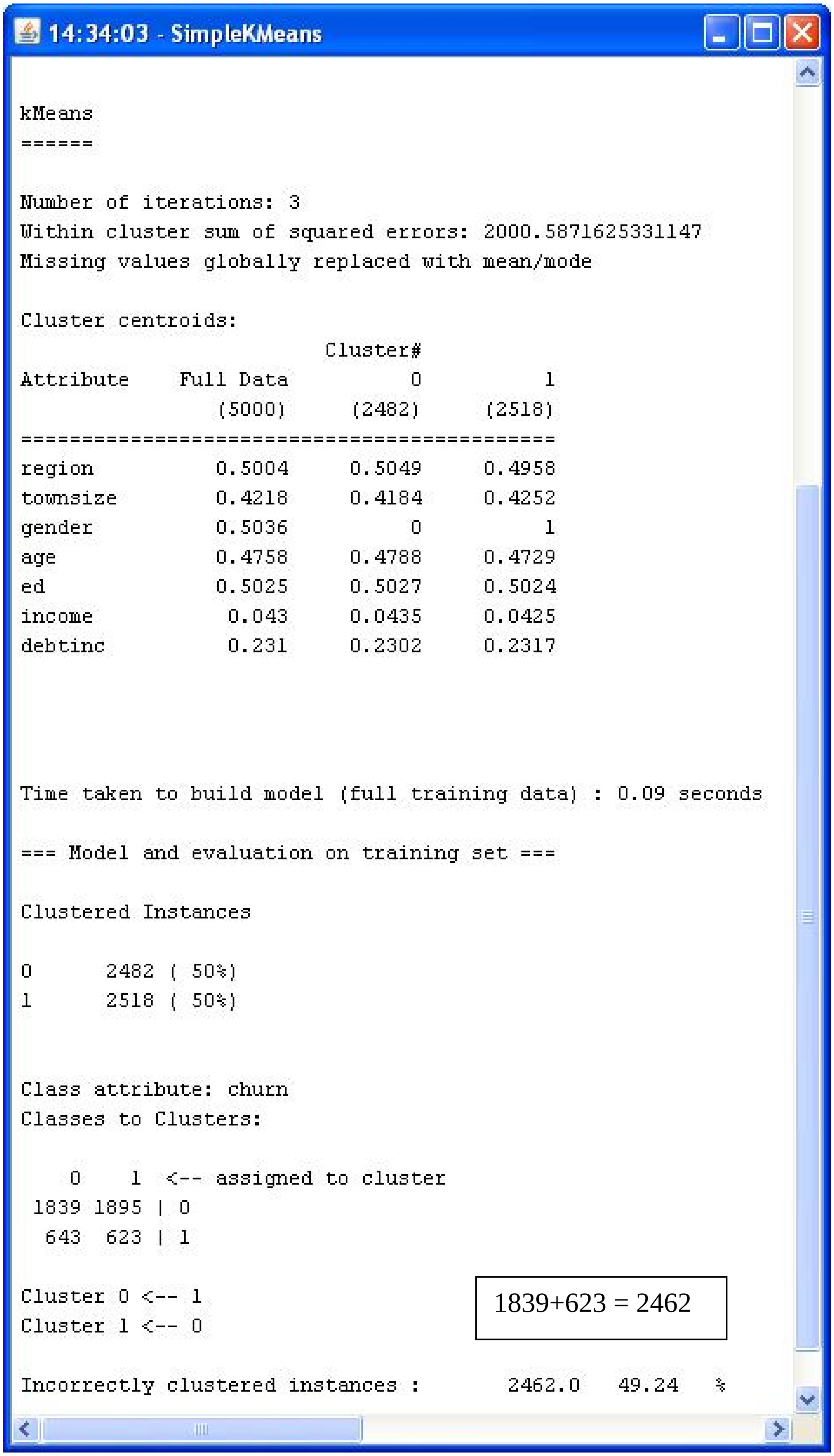
Clusterer: **SimpleKMeans** -N 2 -A "weka.core.EuclideanDistance -R first-last" -I 500 -S 10

Cluster mode:
- ○ Use training set
- ○ Supplied test set
- ○ Percentage split % 66
- ● Classes to clusters evaluation
- (Nom) churn
- ☑ Store clusters for visualization

Result list (right-click for options)
- 14:33:32 - SimpleKMeans
- 14:34:03 - SimpleKMeans

```
=== Run information ===

Scheme:weka.clusterers.SimpleKMeans -N 2 -A "weka.core.EuclideanD
Relation:     telco labFour-weka.filters.unsupervised.attribute.N
Instances:    5000
Attributes:   11
              region
              townsize
              gender
              age
              ed
              income
              debtinc
Ignored:
              custid
              agecat
              edcat
              churn
Test mode:Classes to clusters evaluation on training data
=== Model and evaluation on training set ===


kMeans
```

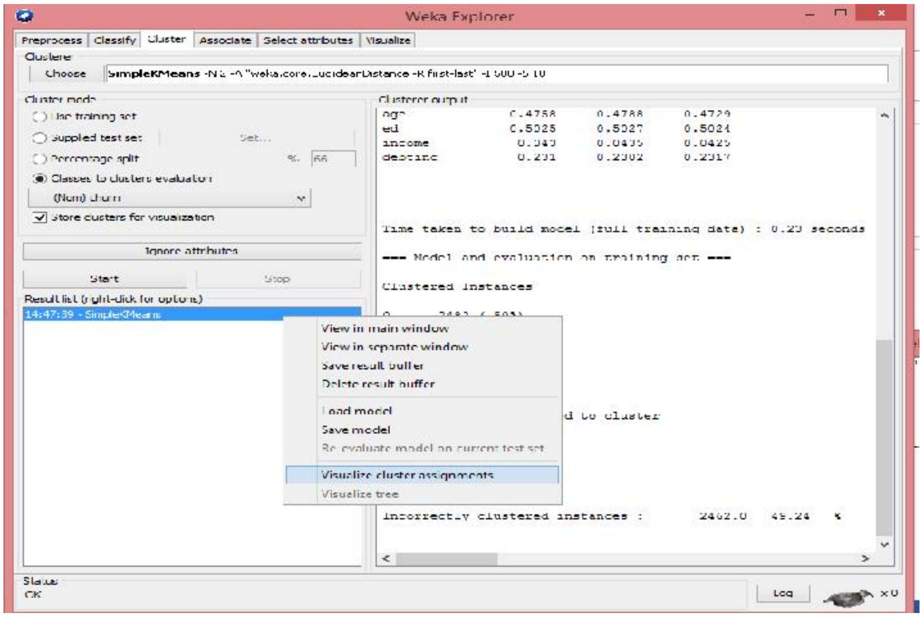
Status
OK



```
kMeans
======

Number of iterations: 3
Within cluster sum of squared errors: 2000.5871625331147
Missing values globally replaced with mean/mode

Cluster centroids:
                         Cluster#
Attribute    Full Data          0          1
                (5000)     (2482)     (2518)
============================================
region          0.5004     0.5049     0.4958
townsize        0.4218     0.4184     0.4252
gender          0.5036          0          1
age             0.4758     0.4788     0.4729
ed              0.5025     0.5027     0.5024
income           0.043     0.0435     0.0425
debtinc          0.231     0.2302     0.2317




Time taken to build model (full training data) : 0.09 seconds

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

Clustered Instances

0      2482 ( 50%)
1      2518 ( 50%)


Class attribute: churn
Classes to Clusters:

    0    1  <-- assigned to cluster
 1839 1895 | 0
  643  623 | 1

Cluster 0 <-- 1
Cluster 1 <-- 0

Incorrectly clustered instances :       2462.0   49.24   %
```

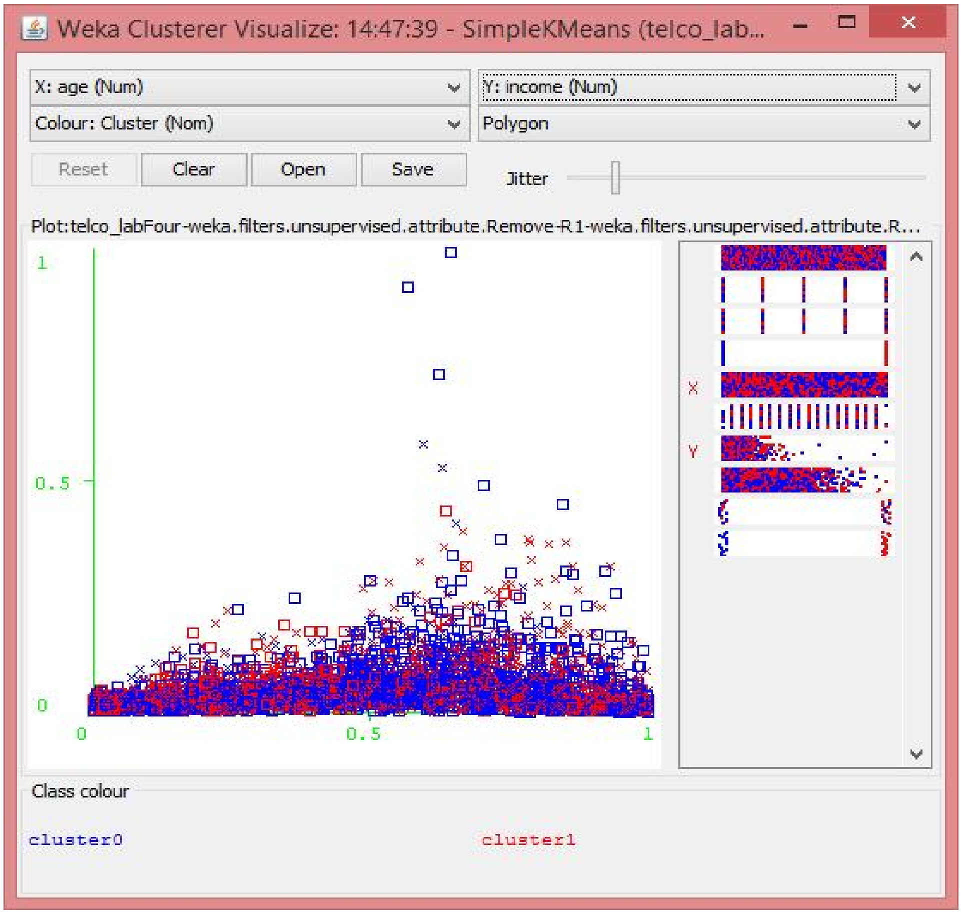
1839+623 = 2462

7. You could visualize the clustering results by right-clicking the result list and choose visualize clusters assignments. You could select different combination of two attributes as X and Y.



SimpleKMeans -N 2 -A "weka.core.EuclideanDistance -R first-last" -I 500 -S 10

```
age                0.4758      0.4788      0.4729
ed                 0.5025      0.5027      0.5024
income              0.049      0.0435      0.0425
debtinc             0.231      0.2302      0.2317



Time taken to build model (full training data) : 0.23 seconds

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

Clustered Instances

0       2482 ( 50%)

d to cluster

Incorrectly clustered instances :       2462.0   49.24   %
```



X: age (Num)

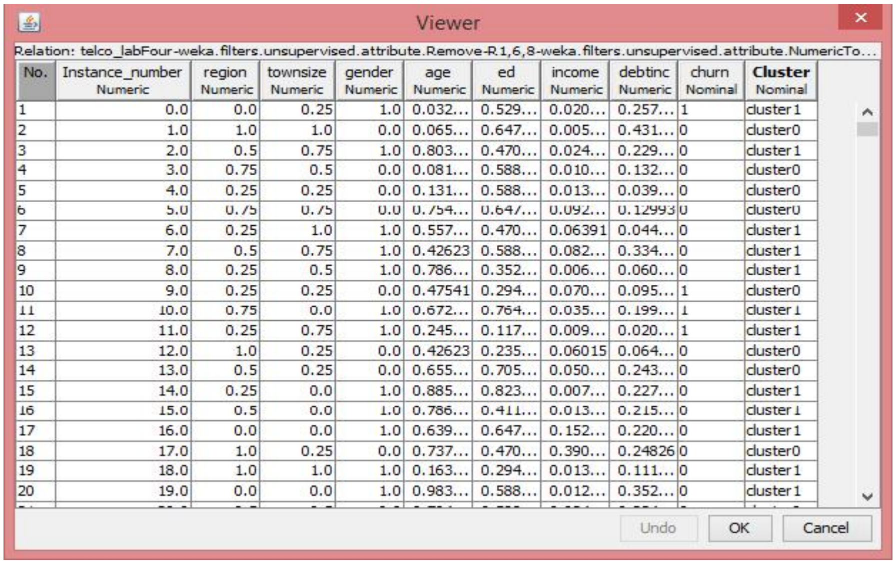
Y: income (Num)

Colour: Cluster (Nom)

Polygon

Plot:telco_labFour-weka.filters.unsupervised.attribute.Remove-R1-weka.filters.unsupervised.attribute.R...

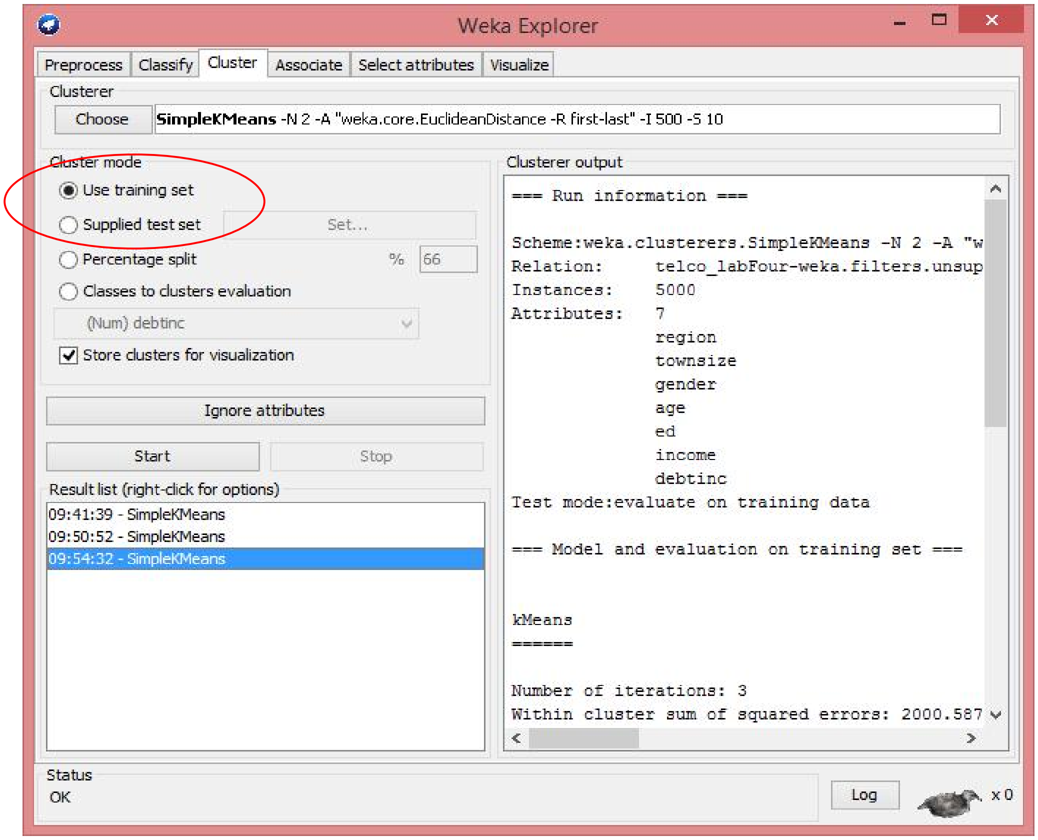
8. You could save the clustering results by clicking Save button on the Visualization panel. The results are saved in a **.arff** file. You could use Weka to open it and view the results.



Relation: telco_labFour-weka.filters.unsupervised.attribute.Remove-R1,6,8-weka.filters.unsupervised.attribute.NumericTo...

No.	Instance_number Numeric	region Numeric	townsize Numeric	gender Numeric	age Numeric	ed Numeric	income Numeric	debtinc Numeric	churn Nominal	**Cluster** Nominal
1	0.0	0.0	0.25	1.0	0.032...	0.529...	0.020...	0.257...	1	cluster1
2	1.0	1.0	1.0	0.0	0.065...	0.647...	0.005...	0.431...	0	cluster0
3	2.0	0.5	0.75	1.0	0.803...	0.470...	0.024...	0.229...	0	cluster1
4	3.0	0.75	0.5	0.0	0.081...	0.588...	0.010...	0.132...	0	cluster0
5	4.0	0.25	0.25	0.0	0.131...	0.588...	0.013...	0.039...	0	cluster0
6	5.0	0.75	0.75	0.0	0.754...	0.647...	0.092...	0.12993	0	cluster0
7	6.0	0.25	1.0	1.0	0.557...	0.470...	0.06391	0.044...	0	cluster1
8	7.0	0.5	0.75	1.0	0.42623	0.588...	0.082...	0.334...	0	cluster1
9	8.0	0.25	0.5	1.0	0.786...	0.352...	0.006...	0.060...	0	cluster1
10	9.0	0.25	0.25	0.0	0.47541	0.294...	0.070...	0.095...	1	cluster0
11	10.0	0.75	0.0	1.0	0.672...	0.764...	0.035...	0.199...	1	cluster1
12	11.0	0.25	0.75	1.0	0.245...	0.117...	0.009...	0.020...	1	cluster1
13	12.0	1.0	0.25	0.0	0.42623	0.235...	0.06015	0.064...	0	cluster0
14	13.0	0.5	0.25	0.0	0.655...	0.705...	0.050...	0.243...	0	cluster0
15	14.0	0.25	0.0	1.0	0.885...	0.823...	0.007...	0.227...	0	cluster1
16	15.0	0.5	0.0	1.0	0.786...	0.411...	0.013...	0.215...	0	cluster1
17	16.0	0.0	0.0	1.0	0.639...	0.647...	0.152...	0.220...	0	cluster1
18	17.0	1.0	0.25	0.0	0.737...	0.470...	0.390...	0.24826	0	cluster0
19	18.0	1.0	1.0	1.0	0.163...	0.294...	0.013...	0.111...	0	cluster1
20	19.0	0.0	0.0	1.0	0.983...	0.588...	0.012...	0.352...	0	cluster1

9. If the data set has no class labels, then when you perform clustering on the data set, choose Use Training Dataset as Cluster mode.



SimpleKMeans -N 2 -A "weka.core.EuclideanDistance -R first-last" -I 500 -S 10

Cluster mode

● Use training set

○ Supplied test set

○ Percentage split % 66

○ Classes to clusters evaluation

(Num) debtinc

☑ Store clusters for visualization

Result list (right-click for options)

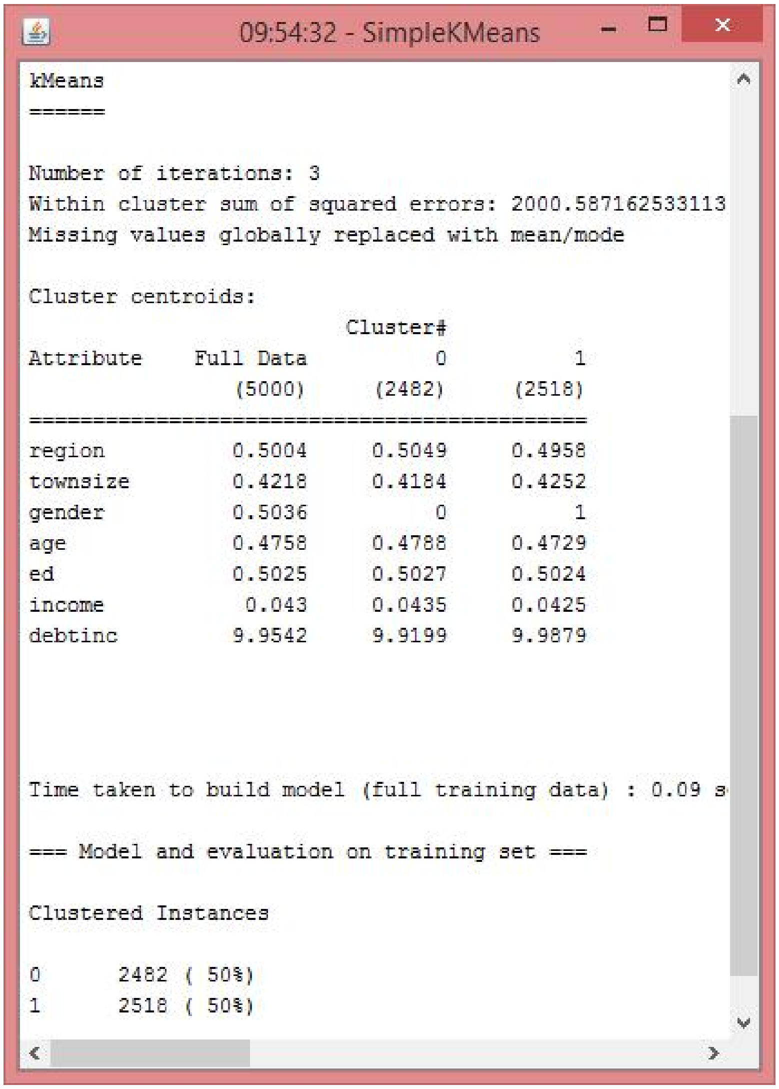
09:41:39 - SimpleKMeans
09:50:52 - SimpleKMeans
09:54:32 - SimpleKMeans

Clusterer output

```
=== Run information ===

Scheme:weka.clusterers.SimpleKMeans -N 2 -A "w
Relation:     telco_labFour-weka.filters.unsup
Instances:    5000
Attributes:   7
              region
              townsize
              gender
              age
              ed
              income
              debtinc
Test mode:evaluate on training data

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


kMeans
======

Number of iterations: 3
Within cluster sum of squared errors: 2000.587
```

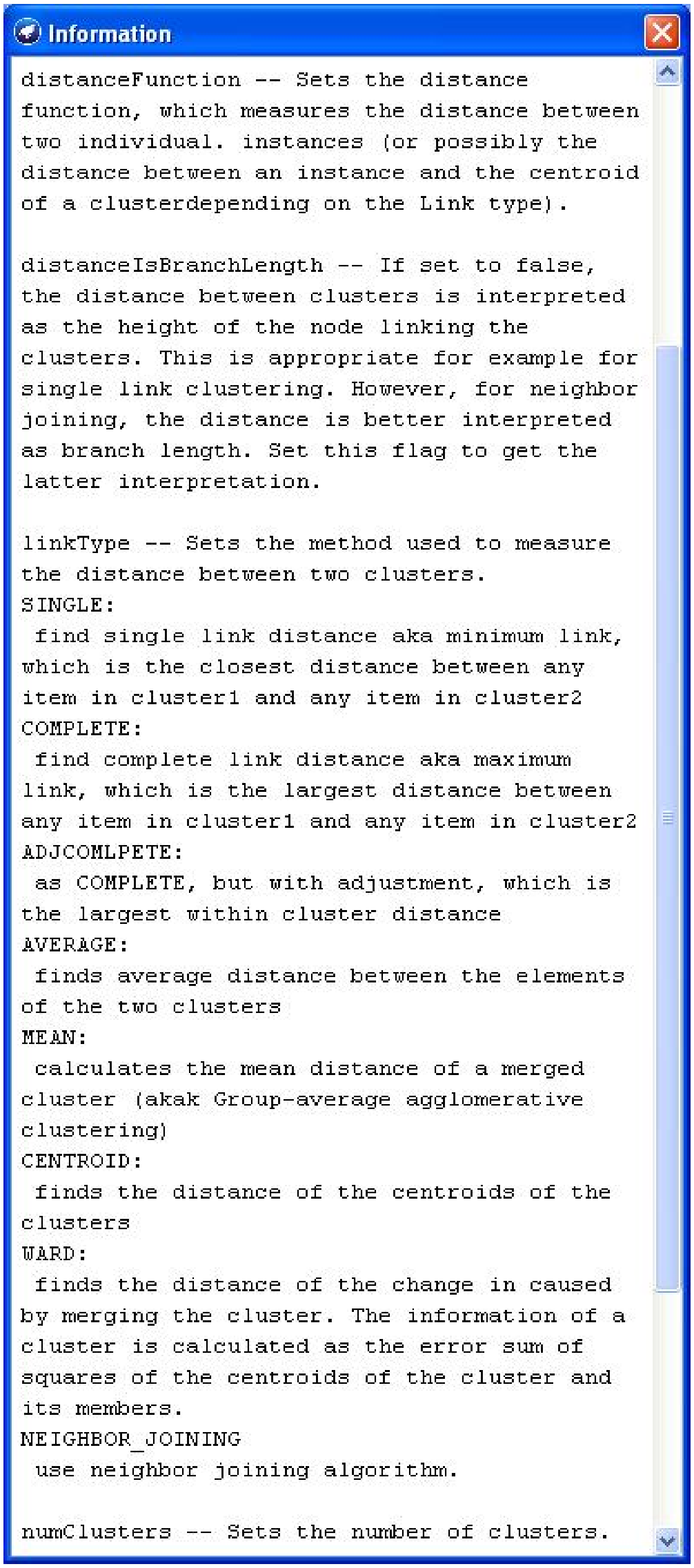
Status

OK



```
kMeans
======

Number of iterations: 3
Within cluster sum of squared errors: 2000.587162533113
Missing values globally replaced with mean/mode

Cluster centroids:
                          Cluster#
Attribute    Full Data           0           1
                (5000)      (2482)      (2518)
=============================================
region          0.5004      0.5049      0.4958
townsize        0.4218      0.4184      0.4252
gender          0.5036           0           1
age             0.4758      0.4788      0.4729
ed              0.5025      0.5027      0.5024
income           0.043      0.0435      0.0425
debtinc         9.9542      9.9199      9.9879




Time taken to build model (full training data) : 0.09 s

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

Clustered Instances

0      2482 ( 50%)
1      2518 ( 50%)
```

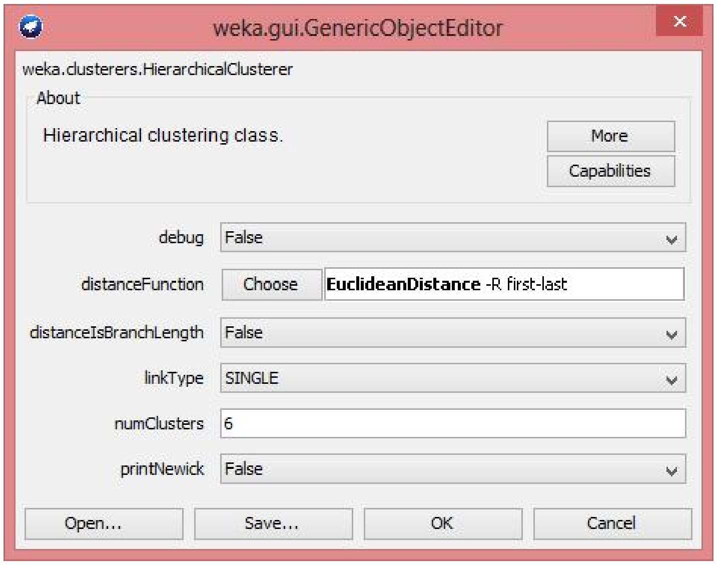
10. ***HierarchicalClusterer*** implements agglomerative (bottom-up) generation of hierarchical clusters. Several different link types, which are ways of measuring the distance between clusters, are available as options.



```
distanceFunction -- Sets the distance
function, which measures the distance between
two individual. instances (or possibly the
distance between an instance and the centroid
of a clusterdepending on the Link type).

distanceIsBranchLength -- If set to false,
the distance between clusters is interpreted
as the height of the node linking the
clusters. This is appropriate for example for
single link clustering. However, for neighbor
joining, the distance is better interpreted
as branch length. Set this flag to get the
latter interpretation.

linkType -- Sets the method used to measure
the distance between two clusters.
SINGLE:
 find single link distance aka minimum link,
which is the closest distance between any
item in cluster1 and any item in cluster2
COMPLETE:
 find complete link distance aka maximum
link, which is the largest distance between
any item in cluster1 and any item in cluster2
ADJCOMLPETE:
 as COMPLETE, but with adjustment, which is
the largest within cluster distance
AVERAGE:
 finds average distance between the elements
of the two clusters
MEAN:
 calculates the mean distance of a merged
cluster (akak Group-average agglomerative
clustering)
CENTROID:
 finds the distance of the centroids of the
clusters
WARD:
 finds the distance of the change in caused
by merging the cluster. The information of a
cluster is calculated as the error sum of
squares of the centroids of the cluster and
its members.
NEIGHBOR_JOINING
 use neighbor joining algorithm.

numClusters -- Sets the number of clusters.
```

11. Since the Hierarchical Clustering algorithm builds a tree for the whole dataset, let’s practice this algorithm on a smaller dataset due to the memory space

limitation. Open ***glass.arff*** dataset used in the previous lab, first normalize all the numeric values in the dataset into [0,1]. Then chose ***HierarchicalCluster*** cluster. Since this datset has 6 classes, we set ***numClusters*** as 6. To save time, we set ***printNewick*** as **False**.



weka.clusterers.HierarchicalClusterer

About

Hierarchical clustering class.

debug: False

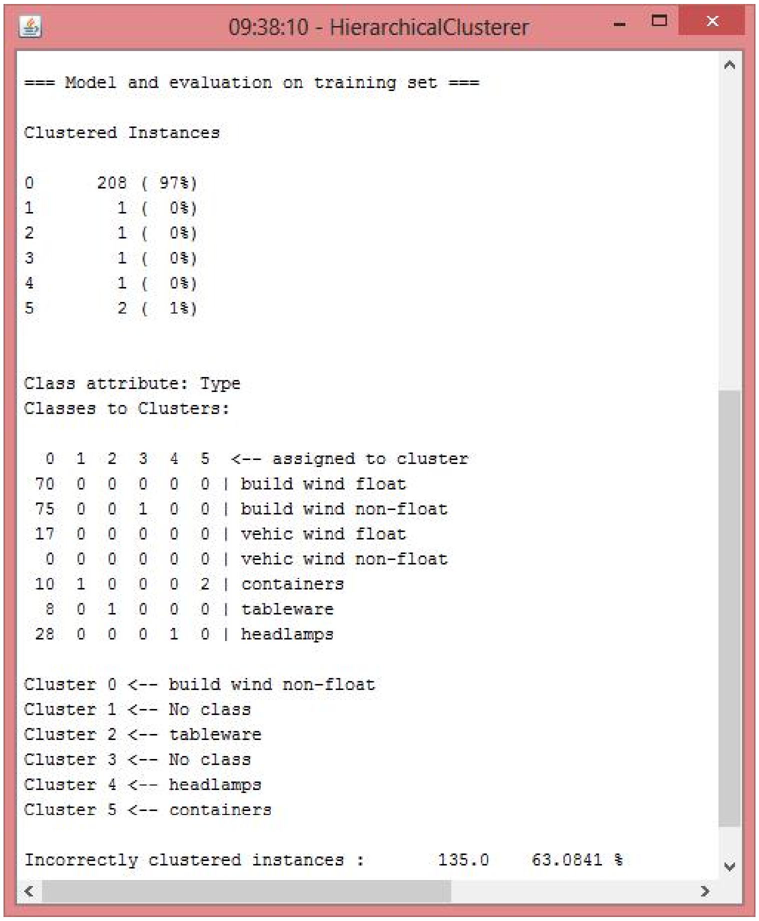
distanceFunction: **EuclideanDistance** -R first-last

distanceIsBranchLength: False

linkType: SINGLE

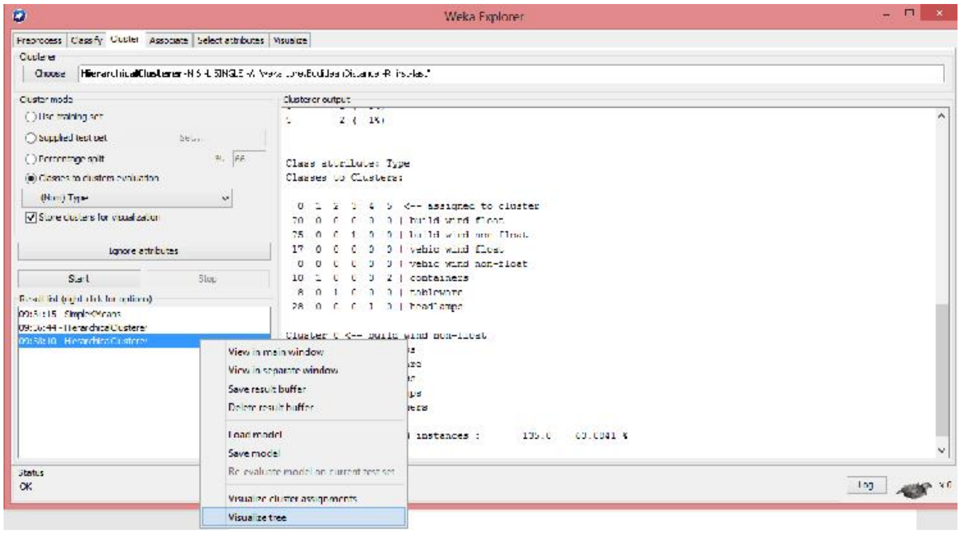
numClusters: 6

printNewick: False



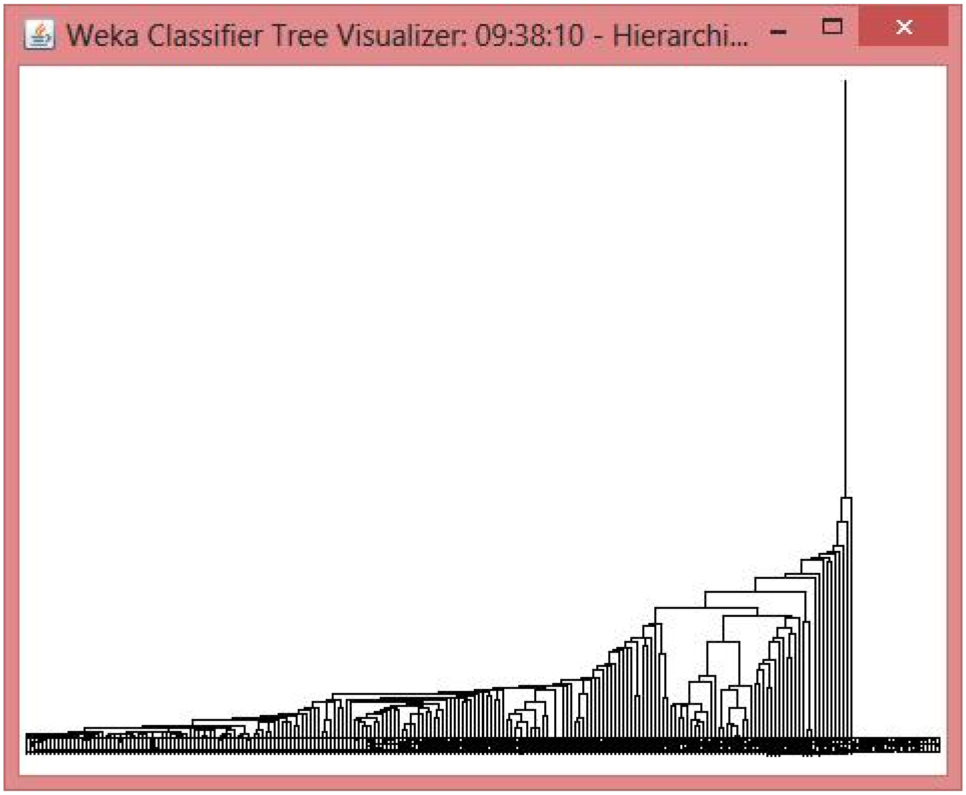
```
=== Model and evaluation on training set ===

Clustered Instances

0      208 ( 97%)
1        1 (  0%)
2        1 (  0%)
3        1 (  0%)
4        1 (  0%)
5        2 (  1%)


Class attribute: Type
Classes to Clusters:

  0  1  2  3  4  5  <-- assigned to cluster
 70  0  0  0  0  0 | build wind float
 75  0  0  1  0  0 | build wind non-float
 17  0  0  0  0  0 | vehic wind float
  0  0  0  0  0  0 | vehic wind non-float
 10  1  0  0  0  2 | containers
  8  0  1  0  0  0 | tableware
 28  0  0  0  1  0 | headlamps

Cluster 0 <-- build wind non-float
Cluster 1 <-- No class
Cluster 2 <-- tableware
Cluster 3 <-- No class
Cluster 4 <-- headlamps
Cluster 5 <-- containers

Incorrectly clustered instances :       135.0    63.0841 %
```

12. After you run the clustering algorithm, you could right click the clustering result and check its hierarchical tree by clicking Visualize Tree.

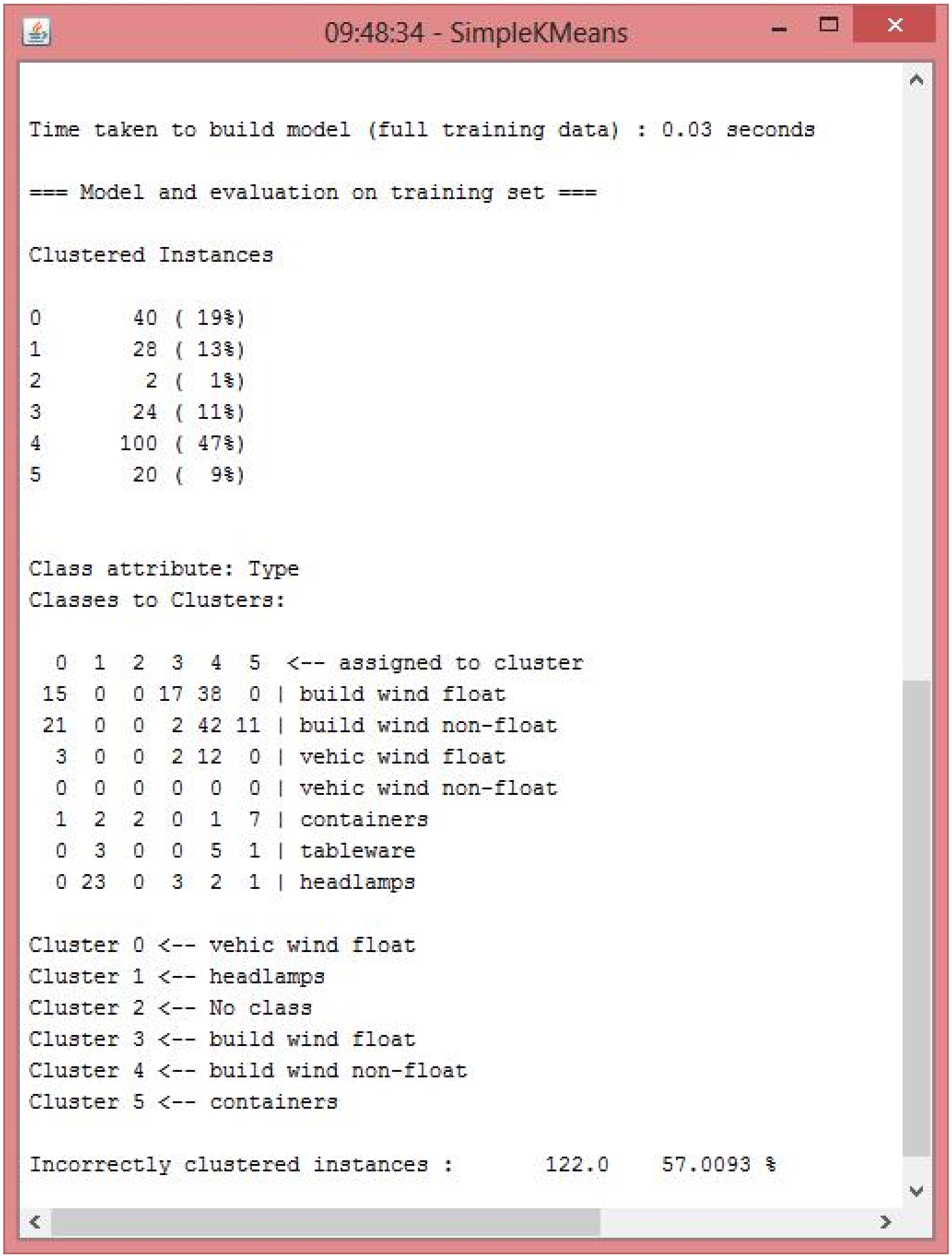


HierarchicalClusterer -N 6 -L SINGLE -A "weka.core.EuclideanDistance -R first-last"

```
5       2 (  1%)


Class attribute: Type
Classes to Clusters:

  0  1  2  3  4  5  <-- assigned to cluster
 70  0  0  0  0  0 | build wind float
 75  0  0  1  0  0 | build wind non-float
 17  0  0  0  0  0 | vehic wind float
  0  0  0  0  0  0 | vehic wind non-float
 10  1  0  0  0  2 | containers
  8  0  1  0  0  0 | tableware
 28  0  0  0  1  0 | headlamps

Cluster 0 <-- build wind non-float

 instances :      135.0     63.0841 %
```

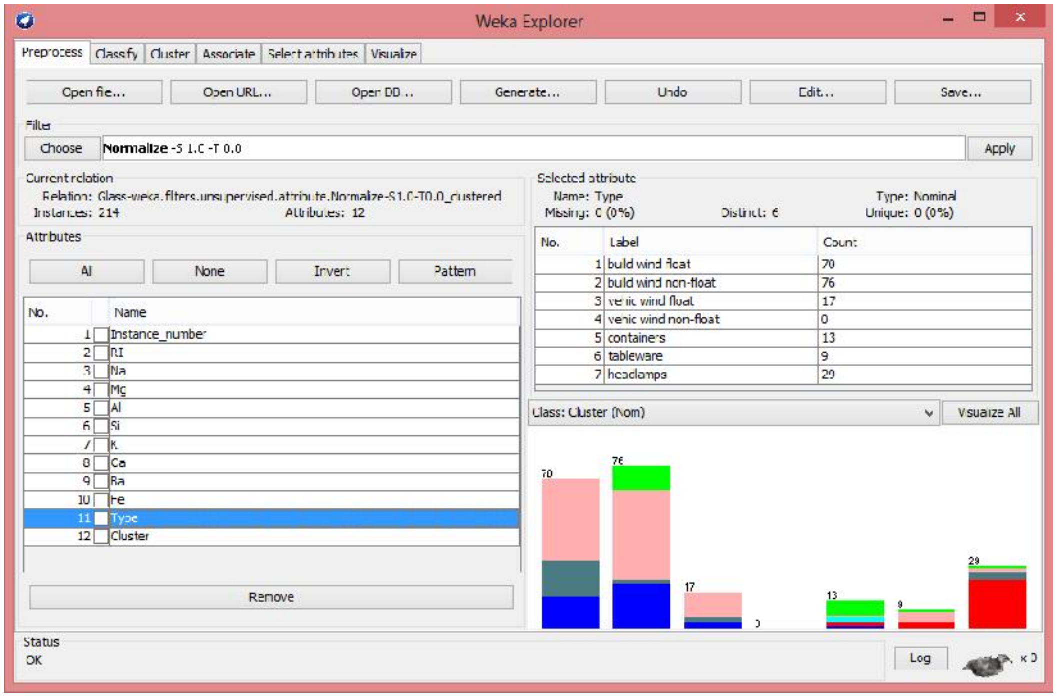


13. Since the performance of HierarchicalClustering is not good, we could run K-means algorithm on the same dataset and compare their performance. Do not forget to set the number of clusters to 6.



```
Time taken to build model (full training data) : 0.03 seconds

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

Clustered Instances

0        40 ( 19%)
1        28 ( 13%)
2         2 (  1%)
3        24 ( 11%)
4       100 ( 47%)
5        20 (  9%)


Class attribute: Type
Classes to Clusters:

  0  1  2  3  4  5  <-- assigned to cluster
 15  0  0 17 38  0 | build wind float
 21  0  0  2 42 11 | build wind non-float
  3  0  0  2 12  0 | vehic wind float
  0  0  0  0  0  0 | vehic wind non-float
  1  2  2  0  1  7 | containers
  0  3  0  0  5  1 | tableware
  0 23  0  3  2  1 | headlamps

Cluster 0 <-- vehic wind float
Cluster 1 <-- headlamps
Cluster 2 <-- No class
Cluster 3 <-- build wind float
Cluster 4 <-- build wind non-float
Cluster 5 <-- containers

Incorrectly clustered instances :      122.0    57.0093 %
```

14. Save the clustering results as ***glass_kmeans_result.arff*** file and reopen it with Weka. Since clustering results are saved as the last column, they are considered as class labels for the datset. The original class labels are considered as a feature of the dataset, you could click the Type attribute to see the overlapping among clusters vs. classes.



Filter

Normalize -S 1.0 -T 0.0

Current relation

Relation: Glass-weka.filters.unsupervised.attribute.Normalize-S1.0-T0.0_clustered

Instances: 214 Attributes: 12

Attributes

No.		Name
1	□	Instance_number
2	□	RI
3	□	Na
4	□	Mg
5	□	Al
6	□	Si
7	□	K
8	□	Ca
9	□	Ba
10	□	Fe
11	□	Type
12	□	Cluster

Selected attribute

Name: Type Type: Nominal

Missing: 0 (0%) Distinct: 6 Unique: 0 (0%)

No.	Label	Count
1	build wind float	70
2	build wind non-float	76
3	vehic wind float	17
4	vehic wind non-float	0
5	containers	13
6	tableware	9
7	headlamps	29

Class: Cluster (Nom)

Status

OK