

# 732A75 Data Mining Lab-1

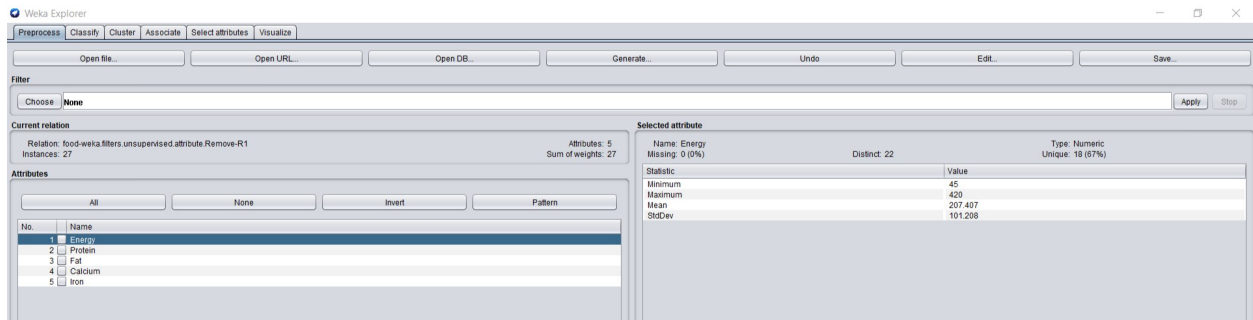
*Sridhar Adhikarla(sriad858) and Lakshidaa Saigiridharan(laksa656)*

*5 March 2019*

## SimpleKmeans:

### 1. Choose a set of attributes for clustering and give a motivation.

Usually in clustering operations we only use numeric attributes, any attributes of categorical type or having too many levels are excluded. It is due to the fact that clustering tends to be based on some distance measurement, and distance of attributes such as name, rownumbers are meaningless and may only come in the way of optimal solution.



## 2. Experiment with at least two different numbers of clusters.

**Weka Explorer**

Preprocess | Classify | **Cluster** | Associate | Select attributes | Visualize

**Clusterer**

Choose **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-slots 1 -S 10

**Cluster mode**

☒ Use training set  
☐ Supplied test set Set...  
☐ Percentage split % 66  
☐ Classes to clusters evaluation (Num) Iron  
☒ Store clusters for visualization

Ignore attributes

Start Stop

**Result list (right-click for options)**

09:31:42 - SimpleKMeans

**Clusterer output**

```
Fat
Calcium
Iron
Test mode: evaluate on training data

=== Clustering model (full training set) ===

kMeans
=====

Number of iterations: 2
Within cluster sum of squared errors: 5.069321339929419

Initial starting points (random):

Cluster 0: 340,20,28,9,2.6
Cluster 1: 170,25,7,12,1.5

Missing values globally replaced with mean/mode

Final cluster centroids:

Attribute    Full Data    Cluster#
              (27.0)      0          1
              (27.0)      (9.0)      (18.0)
=====
Energy       207.4074    331.1111    145.5556
Protein      19          19          19
Fat          13.4815     27.5556     6.4444
Calcium      43.963      8.7778     61.5556
Iron         2.3815      2.4667     2.3389

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

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

Clustered Instances

0      9 ( 33%)
1     18 ( 67%)
```

**Weka Explorer**

Preprocess | Classify | **Cluster** | Associate | Select attributes | Visualize

**Clusterer**

Choose **SimpleKMeans** -init 0 -max-candidates 100 -periodic-pruning 10000 -min-density 2.0 -t1 -1.25 -t2 -1.0 -N 5 -A "weka.core.EuclideanDistance -R first-last" -I 500 -num-slots 1 -S 10

**Cluster mode**

☒ Use training set  
☐ Supplied test set Set...  
☐ Percentage split % 66  
☐ Classes to clusters evaluation (Num) Iron  
☒ Store clusters for visualization

Ignore attributes

Start Stop

**Result list (right-click for options)**

09:31:42 - SimpleKMeans  
09:32:45 - SimpleKMeans

**Clusterer output**

```
=== Clustering model (full training set) ===

kMeans
=====

Number of iterations: 4
Within cluster sum of squared errors: 2.750432407251998

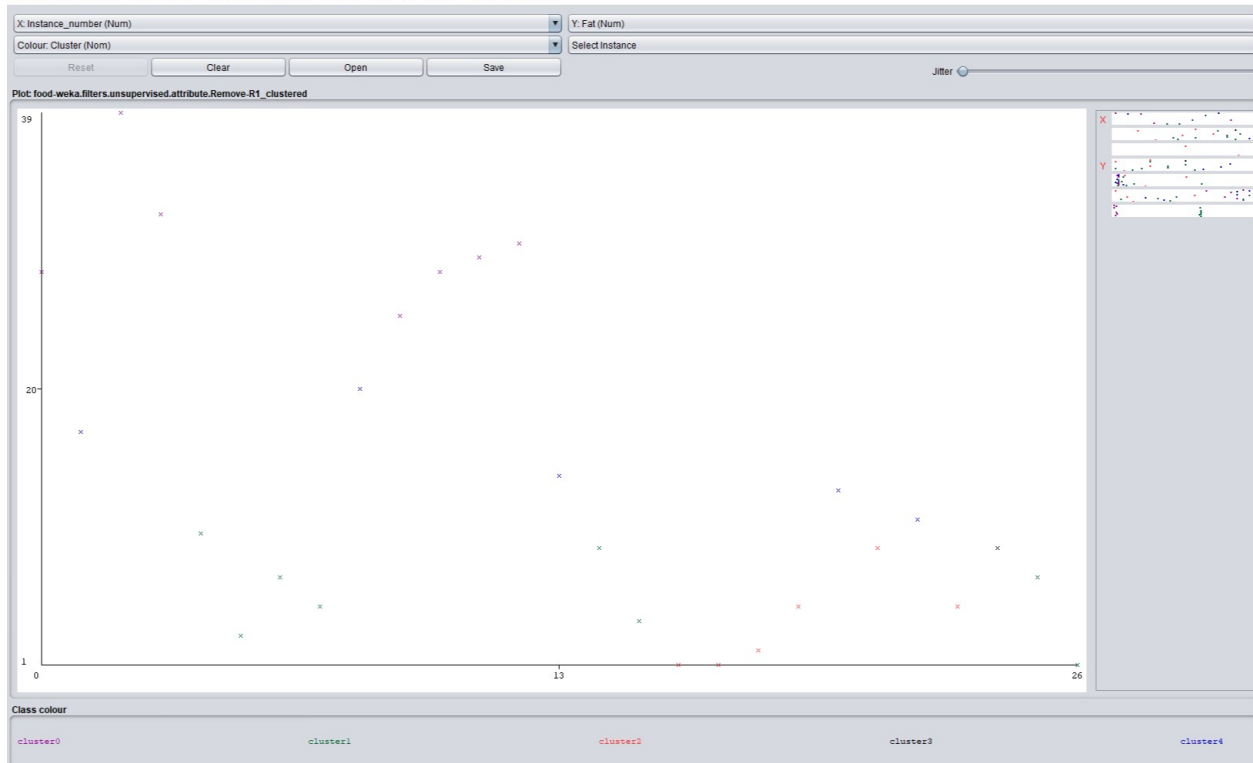
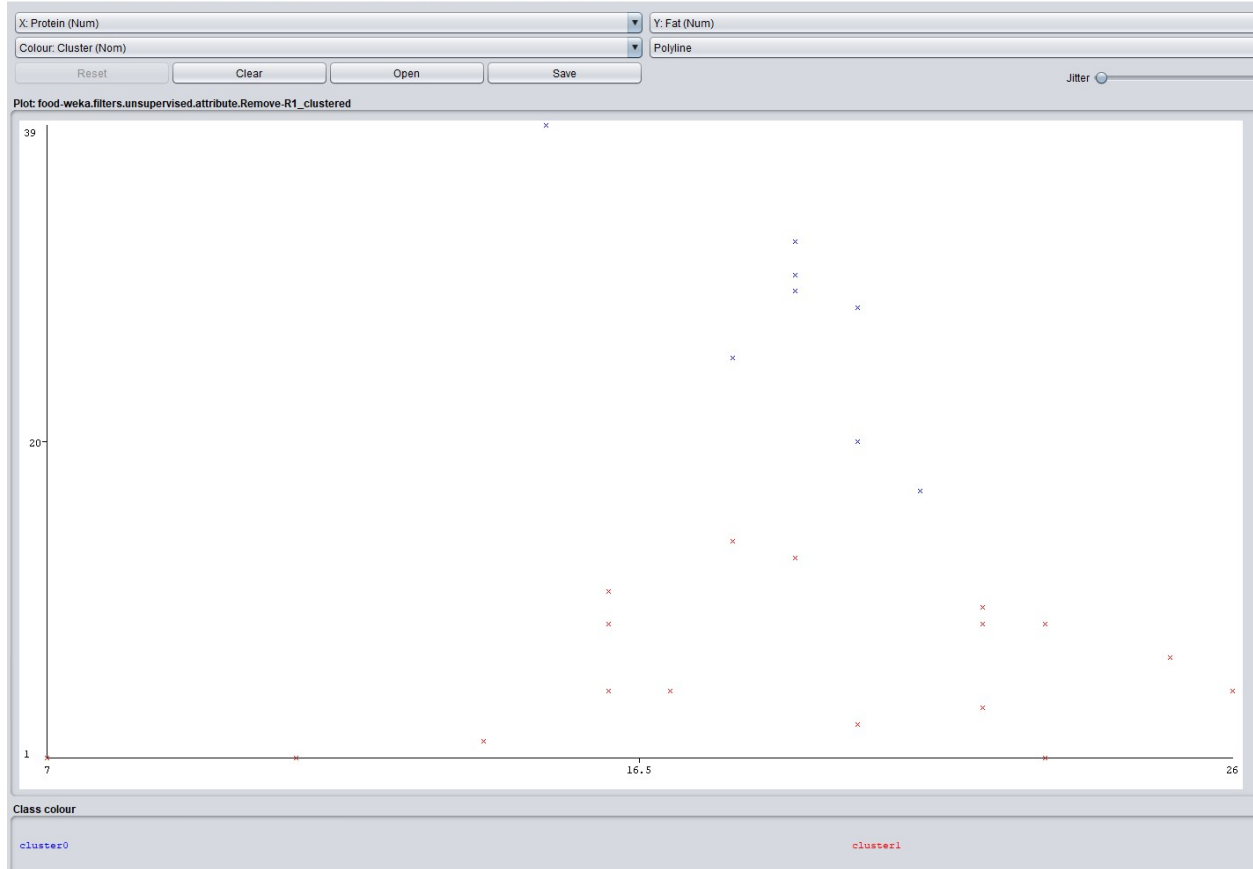
Initial starting points (random):

Cluster 0: 340,20,28,9,2.6
Cluster 1: 170,25,7,12,1.5
Cluster 2: 90,14,2,38,0.8
Cluster 3: 180,22,9,367,2.5
Cluster 4: 300,18,25,9,2.3

Missing values globally replaced with mean/mode

Final cluster centroids:

Attribute    Full Data    Cluster#
              (27.0)      0          1          2          3          4
              (27.0)      (7.0)      (8.0)      (6.0)      (1.0)      (5.0)
=====
Energy       207.4074    352.8571    153.125    102.5    180    222
Protein      19          18.5714     23.25     13.5     22    18.8
Fat          13.4815     30.1429     5.75      3.8333    9     15
Calcium      43.963      8.7143      23.75     87.5     367    8.8
Iron         2.3815      2.4143      2.45      2.5333    2.5    2.02
```



K-means is a partition based clustering algorithm in which as we increase the number of clusters, the number of subdivisions increases. It is due to the fact that none of the clusters can be empty or can overlap in a partition based clustering algorithm, and it covers all the data points. This can lead to the possibility that some clusters have just one or two points, which is meaningless. Thus care must be taken to ensure that we specify  $k$  carefully and try and inspect visually to see if the clusters make sense.

### 3. Try with a different seed value. Compare the results with the previous results. Explain what the seed value controls.

Weka Explorer

Preprocess Classify **Cluster** Associate Select attributes Visualize

Clusterer

Choose **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-slots 1 -S 999

Cluster mode

☒ Use training set  
☐ Supplied test set Set...  
☐ Percentage split % 66  
☐ Classes to clusters evaluation  
(Num) Iron  
☒ Store clusters for visualization

Ignore attributes

Start Stop

Result list (right-click for options)

09:32:45 - SimpleKMeans  
11:26:52 - SimpleKMeans

Clusterer output

Fat  
Calcium  
Iron

Test mode: evaluate on training data

=== Clustering model (full training set) ===

kMeans  
=====

Number of iterations: 6  
Within cluster sum of squared errors: 5.069321339929419

Initial starting points (random):

Cluster 0: 205,18,14,7,2.5  
Cluster 1: 135,22,4,25,0.6

Missing values globally replaced with mean/mode

Final cluster centroids:

Attribute	Full Data	Cluster# 0	Cluster# 1
	(27.0)	(9.0)	(18.0)
Energy	207.4074	331.1111	145.5556
Protein	19	19	19
Fat	13.4815	27.5556	6.4444
Calcium	43.963	8.7778	61.5556
Iron	2.3815	2.4667	2.3389

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

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

Clustered Instances

0	9 ( 33%)
1	18 ( 67%)

Weka Explorer

Preprocess Classify **Cluster** Associate Select attributes Visualize

Clusterer

Choose **SimpleKMeans** -init 0 -max-candidates 100 -periodic-pruning 10000 -min-density 2.0 -t1 -1.25 -t2 -1.0 -N 5 -A "weka.core.EuclideanDistance -R first-last" -I 500 -num-slots 1 -S 999

Cluster mode

☒ Use training set  
☐ Supplied test set Set...  
☐ Percentage split % 66  
☐ Classes to clusters evaluation  
(Num) Iron  
☒ Store clusters for visualization

Ignore attributes

Start Stop

Result list (right-click for options)

09:31:42 - SimpleKMeans  
09:32:45 - SimpleKMeans  
11:26:52 - SimpleKMeans  
11:27:37 - SimpleKMeans

Clusterer output

=== Clustering model (full training set) ===

kMeans  
=====

Number of iterations: 6  
Within cluster sum of squared errors: 1.8449230433405908

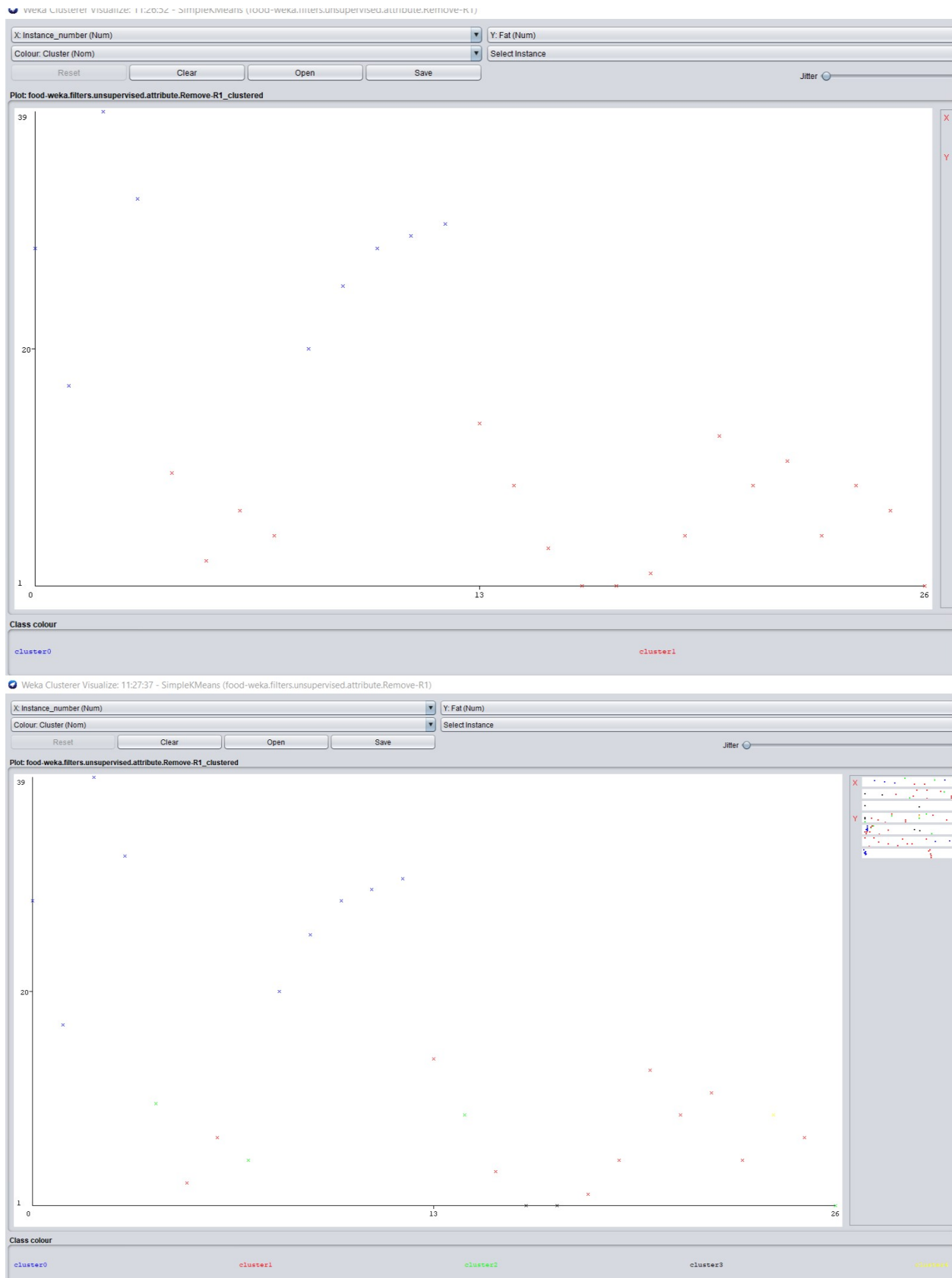
Initial starting points (random):

Cluster 0: 205,18,14,7,2.5  
Cluster 1: 135,22,4,25,0.6  
Cluster 2: 170,25,7,12,1.5  
Cluster 3: 135,16,5,15,0.5  
Cluster 4: 180,22,9,367,2.5

Missing values globally replaced with mean/mode

Final cluster centroids:

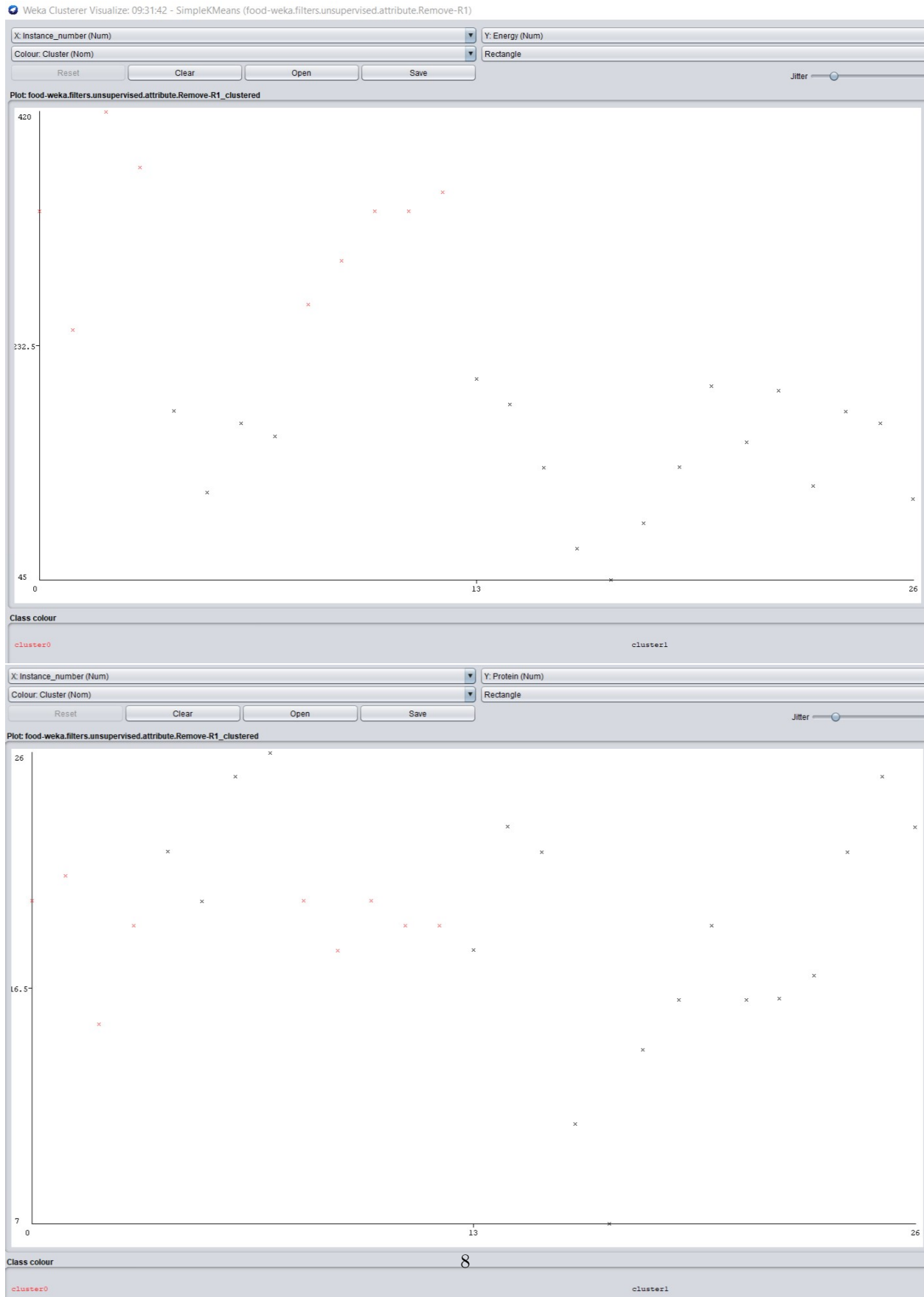
Attribute	Full Data	Cluster# 0	Cluster# 1	Cluster# 2	Cluster# 3	Cluster# 4
	(27.0)	5 (9.0)	(11.0)	(4.0)	(2.0)	(1.0)
Energy	207.4074	331.1111	153.6364	158.75	57.5	180
Protein	19	19	18.9091	23.5	9	22
Fat	13.4815	27.5556	7.2727	6.25	1	9
Calcium	43.963	8.7778	40.6364	34.5	78	367



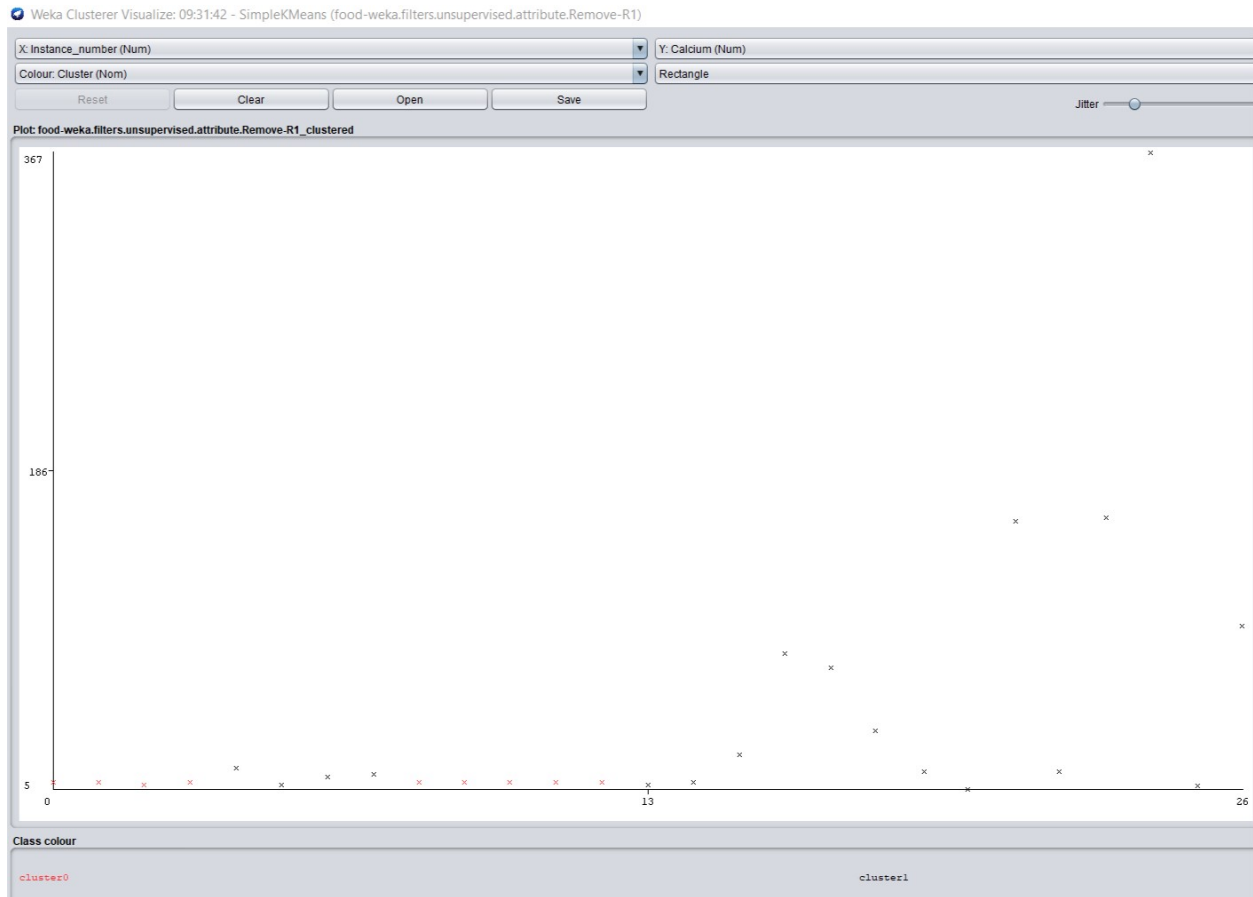
The seed value controls the starting points of the cluster centroids. If we change the value of the seed, the

algorithm seems much more unstable for larger values of  $k$ , producing quite different clusters. This could be due to the fact that we have very few data points. However, for smaller  $k$  values, the clusters produced are quite similar for different seeds. With  $k = 2$  for instance, the results are quite similar, evaluating by the centroids of the clusters, however for  $k = 5$ , much less so. We chose these values in order to compare how meaningful the clusters are when there are many vs when there are few. The number of data points in the clusters may vary too with the seed.

4. Do you think the clusters are “good” clusters? (Are all of its members “similar” to each other? Are members from different clusters dissimilar?)



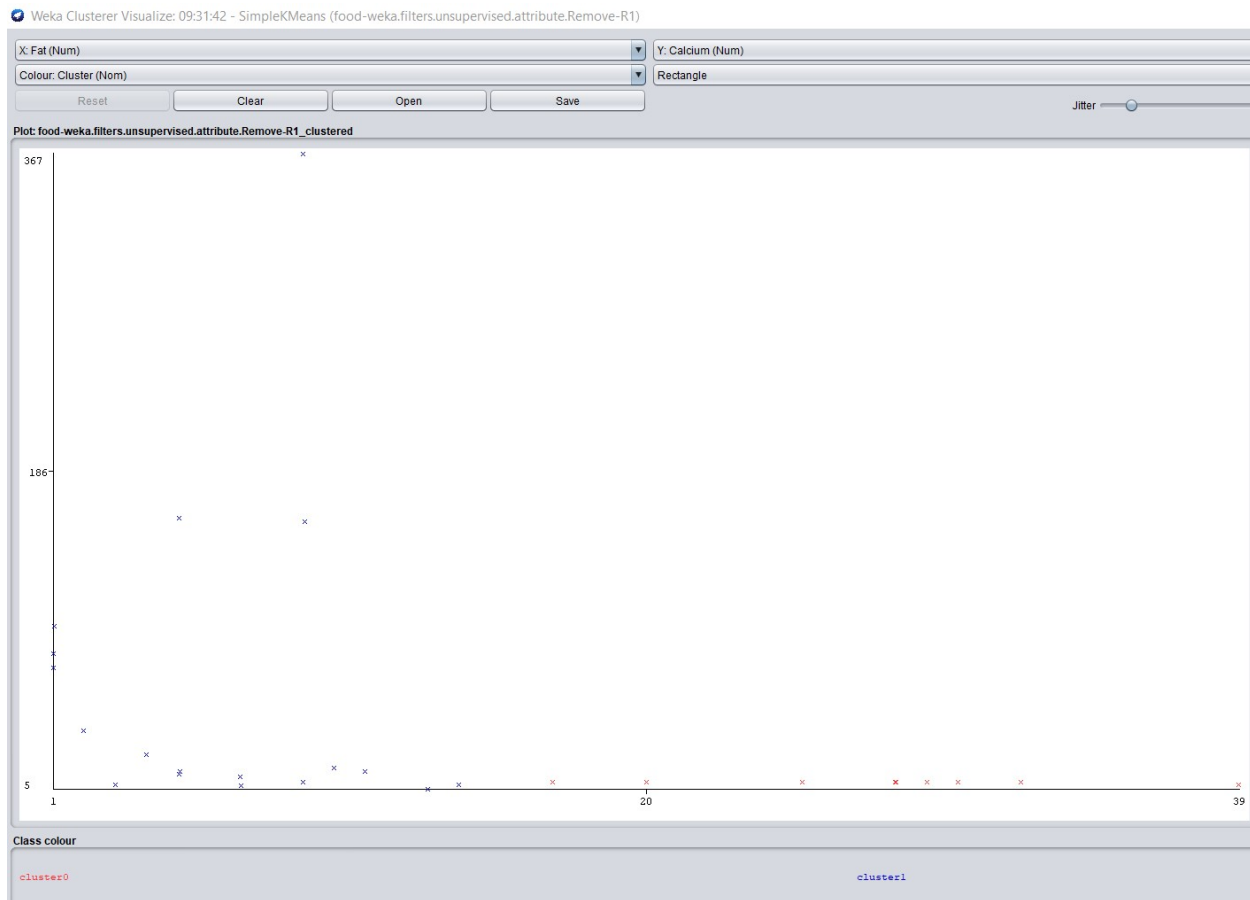




We evaluate the goodness of a cluster based on the attributes we are looking at. For more than two attributes it is hard to visualize them, this is the reason we visualize the cluster against each of the attributes to evaluate them.

The clusters separate some attributes (Energy) very well, but not so good with other attributes (Calcium). In a well done cluster the elements within a cluster are closer to each other and the clusters are separated (further the better). We can see that in the plot of cluster vs Energy. The red cluster elements are a little closely packed and well separated from the blue cluster.

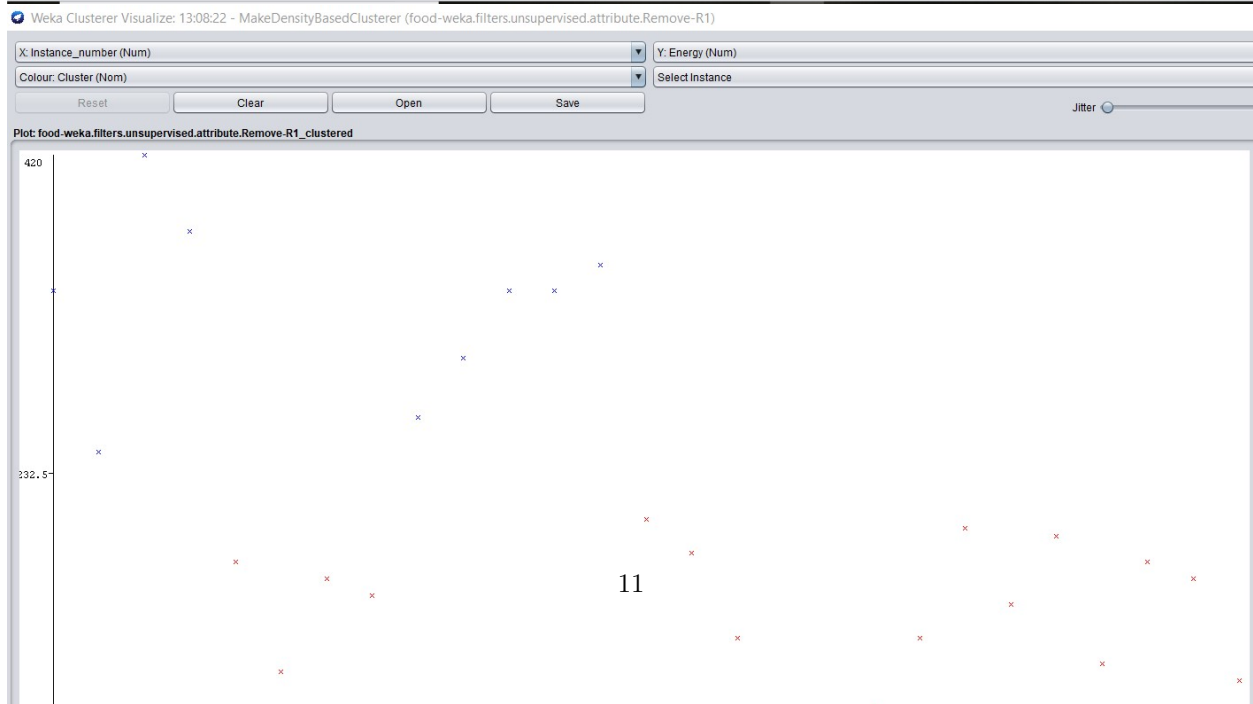
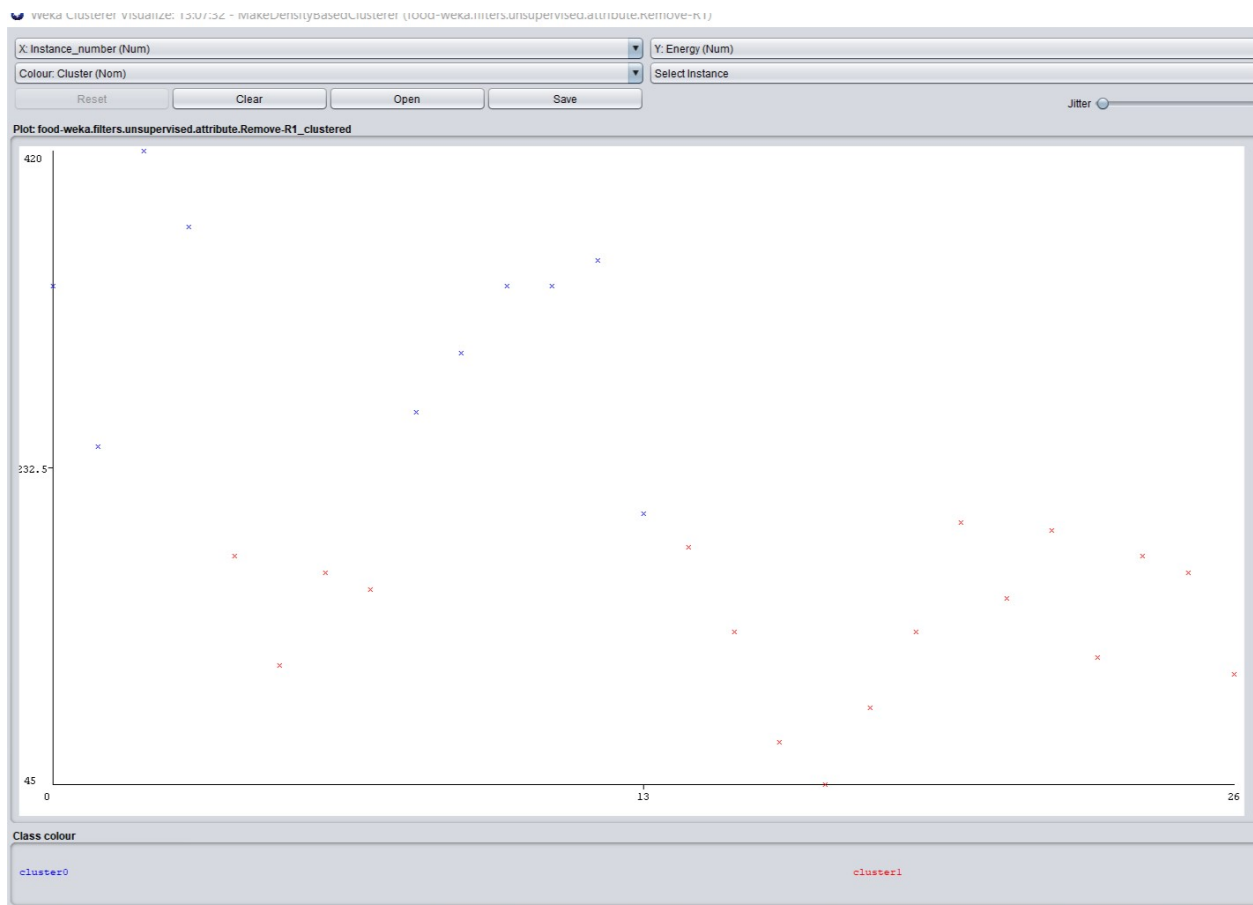
5. What does each cluster represent? Choose one of the results. Make up labels (words or phrases in English) which characterize each cluster.

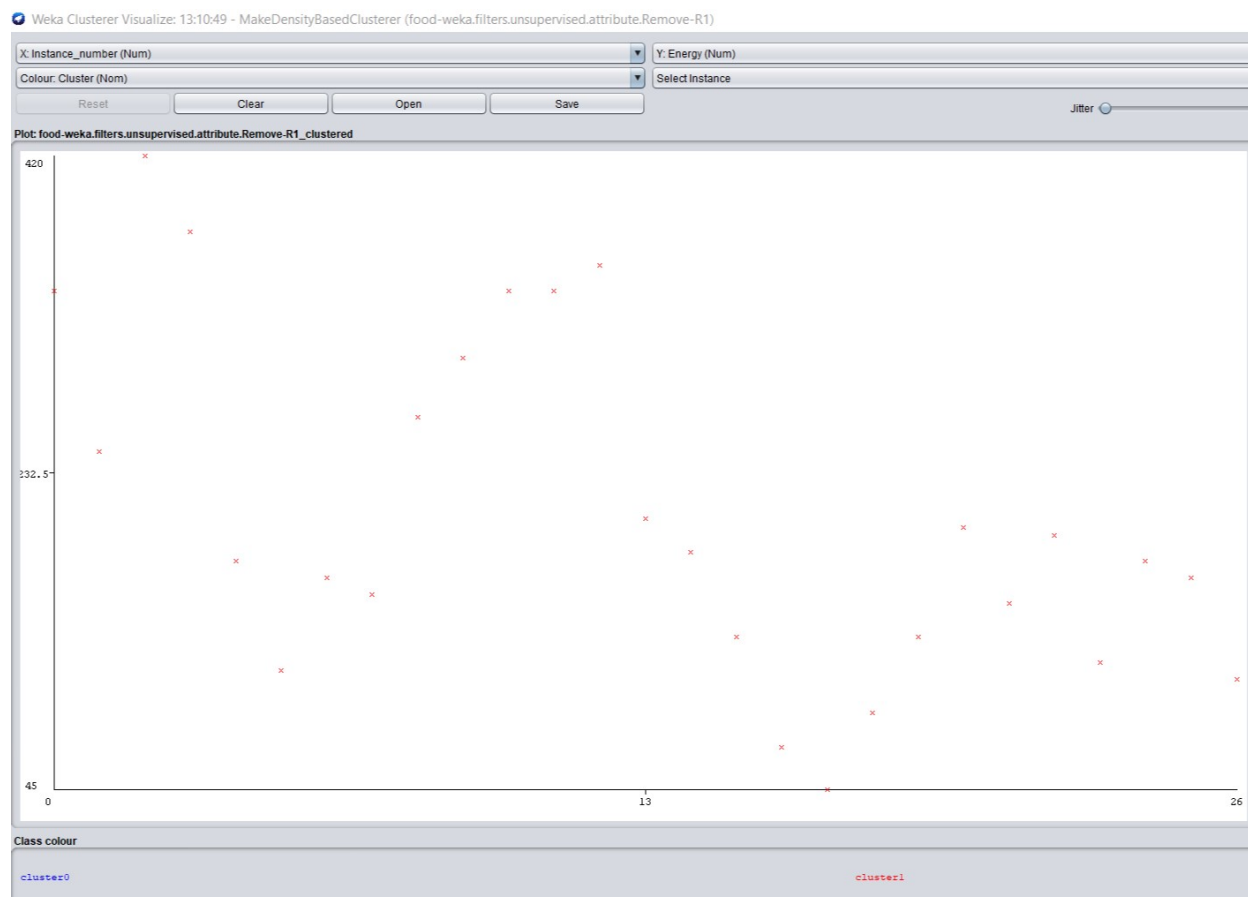


We chose  $k = 2$  with seed 10. As can be seen in the plots above, the clearest separation is in terms of fat content. In particular, we can see that very fatty foods contain very little calcium. The blue cluster contains meats that are high in fat and contain very little calcium, while the red cluster contains meats that are low in fat.

## MakeDensityBasedClusters:

1. Use the SimpleKMeans clusterer which gave the result you haven chosen in 5.
2. Experiment with at least two different standard deviations. Compare the results. (Hint: Increasing the standard deviation to higher values will make the differences in different runs more obvious and thus it will be easier to conclude what the parameter does)





In this method the algorithm creates clusters using K-means and then adjusts them with minimum standard deviation. The standard deviation value influences the size of the final cluster. For a very small min standard deviation value we are guaranteed to have as many clusters as we asked for (ie. equal to the value of  $k$  we specified, eg.  $k=4$ ). But for larger values of min standard deviation it starts merging the smaller clusters to satisfy the min standard deviation threshold, and so we end up with fewer number of clusters. So this also acts like a hyperparameter we use to control the cluster assignment, and the number of clusters.

With  $k = 2$  we first used the default minimum of  $sd = 0.000001$ , and we ended up with two clusters with the same data distribution as with the k-means algorithm. For clustering with  $sd = 10$  we got two clusters again, but two data points shifted cluster assignment. With  $sd = 1000$  all elements end up being in one cluster as a result of no data points being outside this constraint on the minimum standard deviation.