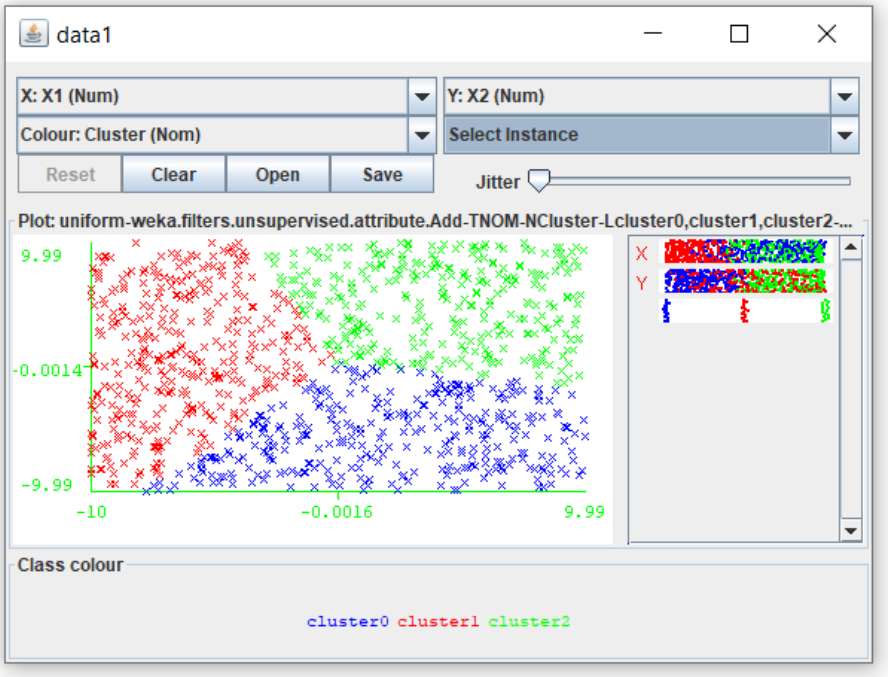
SPRAWOZDANIE – LABORATORIUM 12

Karolina Kotłowska, 19 maja 2023

12.1

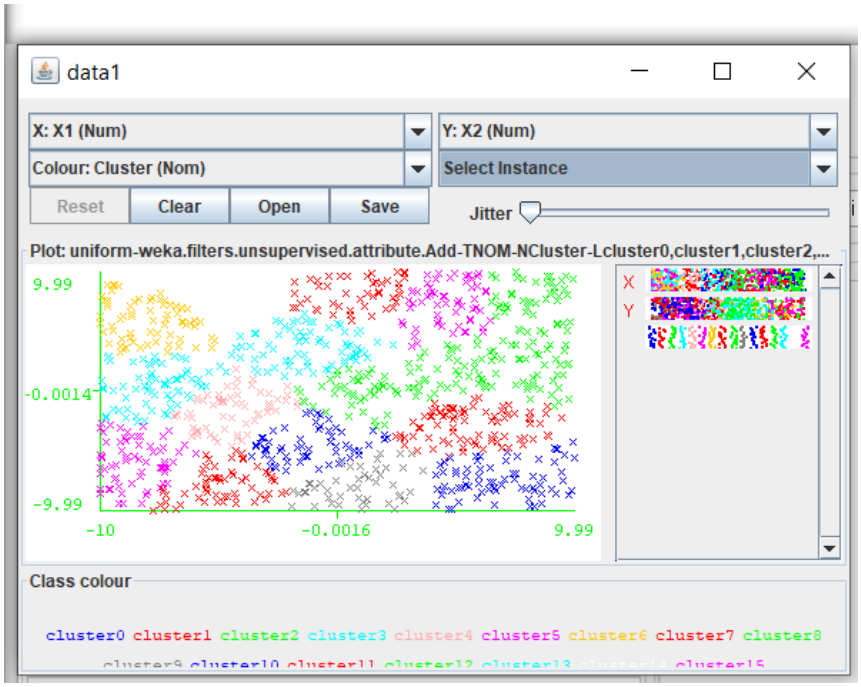
```java
public static void main(String[] args) {
    try{
        ConverterUtils.DataSource source = new ConverterUtils.DataSource( location: "cl-001.arff");
        Instances data = source.getDataSet();

        SimpleKMeans cls = new SimpleKMeans();
        cls.setNumClusters(3);
        cls.setPreserveInstancesOrder(true);
        cls.buildClusterer(data);

        Add filter = new Add();
        filter.setAttributeIndex("last");
        int num  = cls.numberOfClusters();
        String labels = "cluster0";
        for(int i=1;i<num;i++){
            labels+=", cluster";
            labels+=i;
        }
        filter.setNominalLabels(labels);
        filter.setAttributeName("Cluster");
        filter.setInputFormat(data);
        Instances newData = Filter.useFilter(data, filter);

        visualize(newData,  title: "data1");
```



Dla kodu:

```
public static void main(String[] args) {
    try{
        ConverterUtils.DataSource source = new ConverterUtils.DataSource( location: "cl-001.arff");
        Instances data = source.getDataSet();

        SimpleKMeans cls = new SimpleKMeans();
        String stringFromClipboard="-init 0 -max-candidates 100 -periodic-pruning 10000 -min-density 2.0 -t1 " +
                "-1.25 -t2 -1.0 -N 16 -A \"weka.core.EuclideanDistance -R first-last\" -I 500 -num-slots 1 -S 10";
        cls.setOptions(Utils.splitOptions(stringFromClipboard));
        cls.buildClusterer(data);

        Add filter = new Add();
```

Wynik (dla ilości klastrów równej 16):

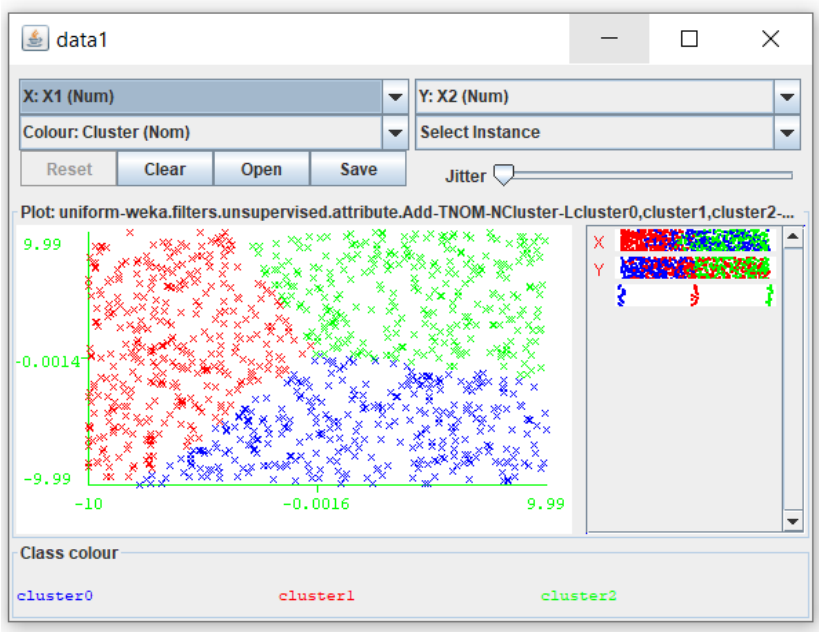


12.2

```
256
257         int idx = newData.numAttributes()-1;
258         for(int i=0;i<newData.numInstances();i++){
259             newData.get(i).setValue(idx, cls.clusterInstance(data.get(i)));
260         }
261
262         Instances centroids = cls.getClusterCentroids();
263         for(int i=0;i<centroids.numAttributes();i++){
264             System.out.print(centroids.attribute(i));
265             System.out.print(",");
266         }
267
268         for(int i=0;i<centroids.numInstances();i++){
269             System.out.println(centroids.get(i));
270         }
271         System.out.printf(Locale.US,  s: "Error: %f",cls.getSquaredError());
272
```

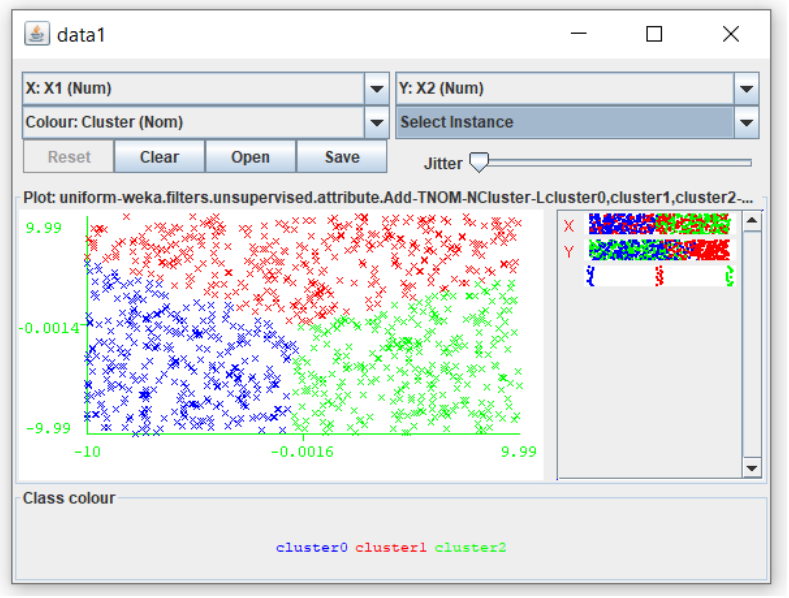
Testowanie wpływu parametrów:

Seed = 10, k = 3



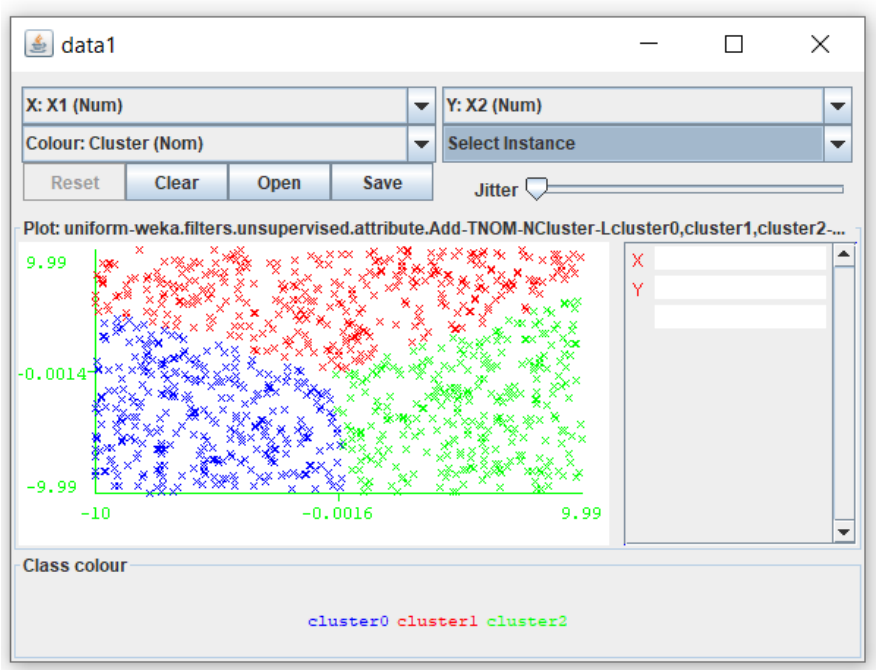
```
"C:\Program Files\Java\jdk1.8.0_111\bin\java.exe" ...
@attribute X1 numeric,@attribute X2 numeric,2.157022,-5.633541
-6.377598,0.868186
4.233062,5.053473
Error: 67.271506DBSCAN clustering results
================================================================================

Clustered DataObjects: 1000
Number of attributes: 2
Epsilon: 0.1; minPoints: 3
Distance-type:
Number of generated clusters: 1
Elapsed time: .07

(   0.) -9.51123,-6.341656                                          -->  0
(   1.) 1.999615,-2.929536                                          -->  0
(   2.) 2.546304,9.941458                                           -->  0
(   3.) -4.040784,2.769457                                          -->  0
(   4.) -9.160898,-9.430981                                         -->  0
(   5.) 5.384602,4.879228                                           -->  0
(   6.) -9.64885,2.807276                                           -->  0
(   7.) 3.43867,0.654373                                            -->  0
(   8.) -2.538701,-6.455795                                         -->  0
(   9.) -2.823715,-1.385636                                         -->  0
(  10.) -6.68313,3.695279                                           -->  0
(  11.) -7.326282,-7.741354                                         -->  0
(  12.) 8.302096,0.350346                                           -->  0
(  13.) 0.21075,-5.874326                                           -->  0
(  14.) 3.13912,-5.315312                                           -->  0
(  15.) 0.990428,7.681875                                           -->  0
(  16.) 6.710052,1.489361                                           -->  0
(  17.) 9.24473,2.671012                                            -->  0
(  18.) -4.135635,9.632951                                          -->  0
(  19.) 6.562687,3.323982                                           -->  0
(  20.) 8.276695,-8.151093                                          -->  0
(  21.) -1.10919,-1.776405                                          -->  0
(  22.) 5.84589,6.949657                                            -->  0
(  23.) -9.106099,-4.36156                                          -->  0
(  24.) -6.823067,-2.099277                                         -->  0
(  25.) 7.227673,-1.215572                                          -->  0
(  26.) 3.363145,5.145105                                           -->  0
(  27.) -4.237179,1.742185                                          -->  0
(  28.) -8.413141,-3.849789                                         -->  0
(  29.) 3.913523,8.652976                                           -->  0
(  30.) 5.982538,-0.643333                                          -->  0
(  31.) 5.961594,-3.664659                                          -->  0
```

Seed = 15, k = 3


data1
X: X1 (Num)
Y: X2 (Num)
Colour: Cluster (Nom)
Select Instance
Reset
Clear
Open
Save
Jitter
Plot: uniform-weka.filters.unsupervised.attribute.Add-TNOM-NCluster-Lcluster0,cluster1,cluster2-...
9.99
-0.0014
-9.99
-10
-0.0016
9.99
X
Y
Class colour
cluster0 cluster1 cluster2

```
"C:\Program Files\Java\jdk1.8.0_111\bin\java.exe" ...
@attribute X1 numeric,@attribute X2 numeric,-5.779038,-3.392108
0.339427,6.223505
5.081385,-3.683625
Error: 65.585522DBSCAN clustering results
================================================================================

Clustered DataObjects: 1000
Number of attributes: 2
Epsilon: 0.1; minPoints: 3
Distance-type:
Number of generated clusters: 1
Elapsed time: .07

(   0.) -9.51123,-6.341656                                          -->  0
(   1.) 1.999615,-2.929536                                          -->  0
(   2.) 2.546304,9.941458                                           -->  0
(   3.) -4.040784,2.769457                                          -->  0
(   4.) -9.160898,-9.430981                                         -->  0
(   5.) 5.384602,4.879228                                           -->  0
(   6.) -9.64885,2.807276                                           -->  0
(   7.) 3.43867,0.654373                                            -->  0
(   8.) -2.538701,-6.455795                                         -->  0
(   9.) -2.823715,-1.385636                                         -->  0
(  10.) -6.68313,3.695279                                           -->  0
(  11.) -7.326282,-7.741354                                         -->  0
(  12.) 8.302096,0.350346                                           -->  0
(  13.) 0.21075,-5.874326                                           -->  0
(  14.) 3.13912,-5.315312                                           -->  0
(  15.) 0.990428,7.681875                                           -->  0
(  16.) 6.710052,1.489361                                           -->  0
(  17.) 9.24473,2.671012                                            -->  0
(  18.) -4.135635,9.632951                                          -->  0
(  19.) 6.562687,3.323982                                           -->  0
(  20.) 8.276695,-8.151093                                          -->  0
(  21.) -1.10919,-1.776405                                          -->  0
(  22.) 5.84589,6.949657                                            -->  0
(  23.) -9.106099,-4.36156                                          -->  0
(  24.) -6.823067,-2.099277                                         -->  0
(  25.) 7.227673,-1.215572                                          -->  0
(  26.) 3.363145,5.145105                                           -->  0
(  27.) -4.237179,1.742185                                          -->  0
(  28.) -8.413141,-3.849789                                         -->  0
(  29.) 3.913523,8.652976                                           -->  0
(  30.) 5.982538,-0.643333                                          -->  0
(  31.) 5.961594,-3.664659                                          -->  0
```

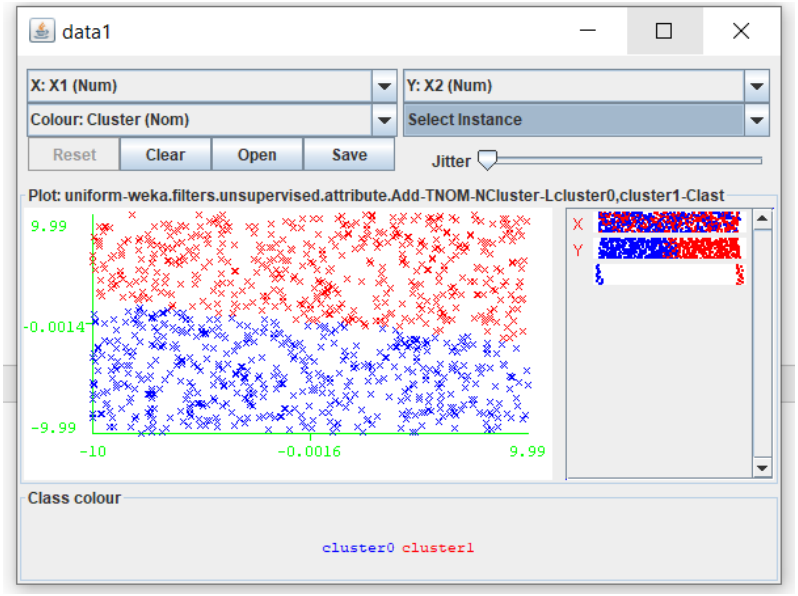
Seed = 130, k = 3


data1
X: X1 (Num)
Y: X2 (Num)
Colour: Cluster (Nom)
Select Instance
Reset
Clear
Open
Save
Jitter
Plot: uniform-weka.filters.unsupervised.attribute.Add-TNOM-NCluster-Lcluster0,cluster1,cluster2-...
9.99
-0.0014
-9.99
-10
-0.0016
9.99
X
Y
Class colour
cluster0 cluster1 cluster2

```
"C:\Program Files\Java\jdk1.8.0_111\bin\java.exe" ...
@attribute X1 numeric,@attribute X2 numeric,-5.502327,-3.759238
-0.271195,6.279937
5.42712,-3.087031
Error: 65.693046DBSCAN clustering results
========================================================================================

Clustered DataObjects: 1000
Number of attributes: 2
Epsilon: 0.1; minPoints: 3
Distance-type:
Number of generated clusters: 1
Elapsed time: .06

(   0.) -9.51123,-6.341656                                          -->  0
(   1.) 1.999615,-2.929536                                          -->  0
(   2.) 2.546304,9.941458                                           -->  0
(   3.) -4.040784,2.769457                                          -->  0
(   4.) -9.160898,-9.430981                                         -->  0
(   5.) 5.384602,4.879228                                           -->  0
(   6.) -9.64885,2.807276                                           -->  0
(   7.) 3.43867,0.654373                                            -->  0
(   8.) -2.538701,-6.455795                                         -->  0
(   9.) -2.823715,-1.385636                                         -->  0
(  10.) -6.68313,3.695279                                           -->  0
(  11.) -7.326282,-7.741354                                         -->  0
(  12.) 8.302096,0.350346                                           -->  0
(  13.) 0.21075,-5.874326                                           -->  0
(  14.) 3.13912,-5.315312                                           -->  0
(  15.) 0.990428,7.681875                                           -->  0
(  16.) 6.710052,1.489361                                           -->  0
(  17.) 9.24473,2.671012                                            -->  0
(  18.) -4.135635,9.632951                                          -->  0
(  19.) 6.562687,3.323982                                           -->  0
(  20.) 8.276695,-8.151093                                          -->  0
(  21.) -1.10919,-1.776405                                          -->  0
(  22.) 5.84589,6.949657                                            -->  0
(  23.) -9.106099,-4.36156                                          -->  0
(  24.) -6.823067,-2.099277                                         -->  0
(  25.) 7.227673,-1.215572                                          -->  0
(  26.) 3.363145,5.145105                                           -->  0
(  27.) -4.237179,1.742185                                          -->  0
(  28.) -8.413141,-3.849789                                         -->  0
(  29.) 3.913523,8.652976                                           -->  0
(  30.) 5.982538,-0.643333                                          -->  0
(  31.) 5.961594,-3.664659                                          -->  0
```

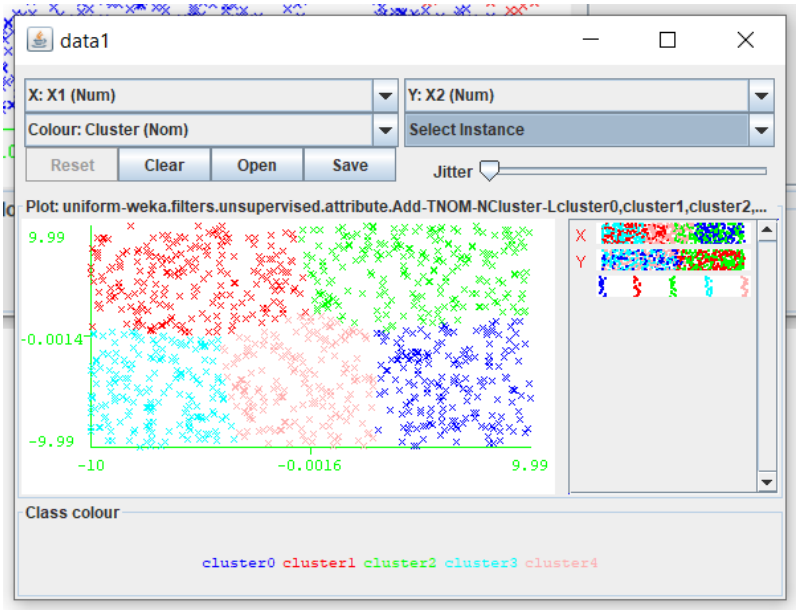
Testowanie dla różnych wartości k.

Seed = 10, k= 2



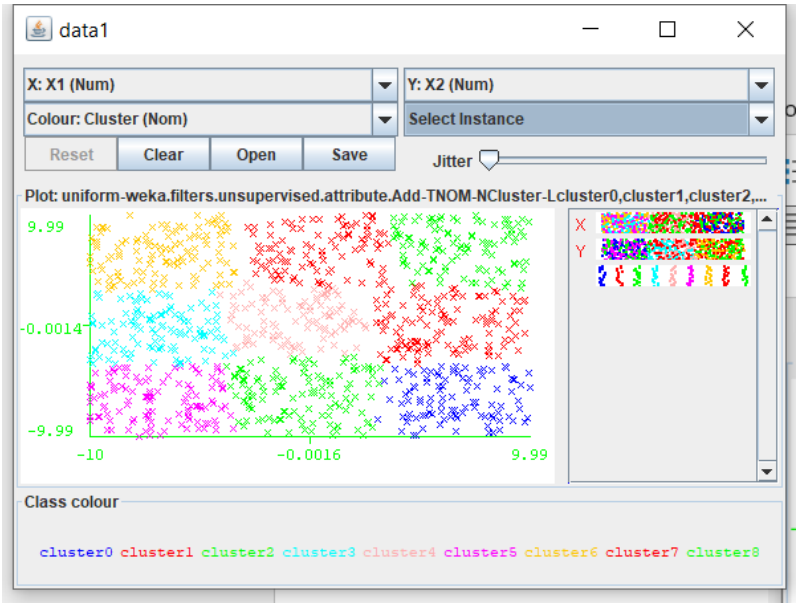
Error: 105.429206

Seed = 10, k = 5



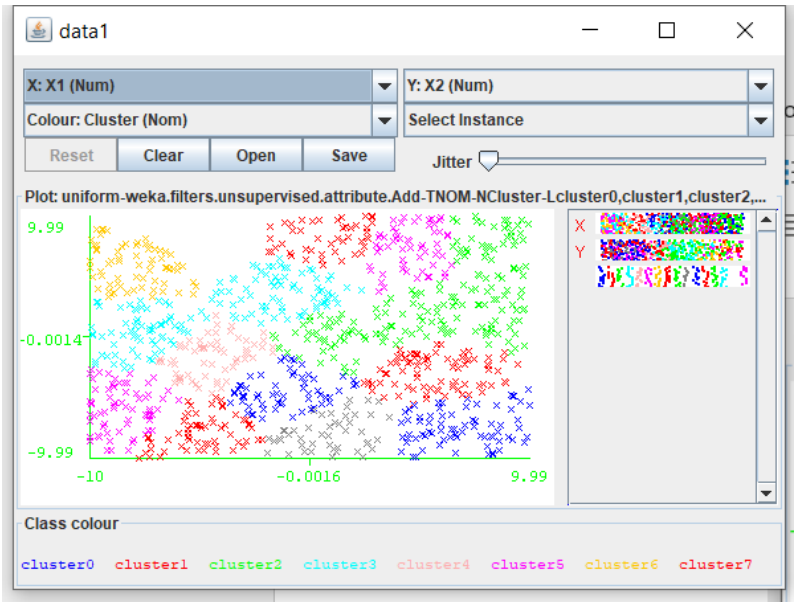
Error: 35.345713

Seed = 10, k = 9



Error: 17.300230

Seed = 10, k = 16

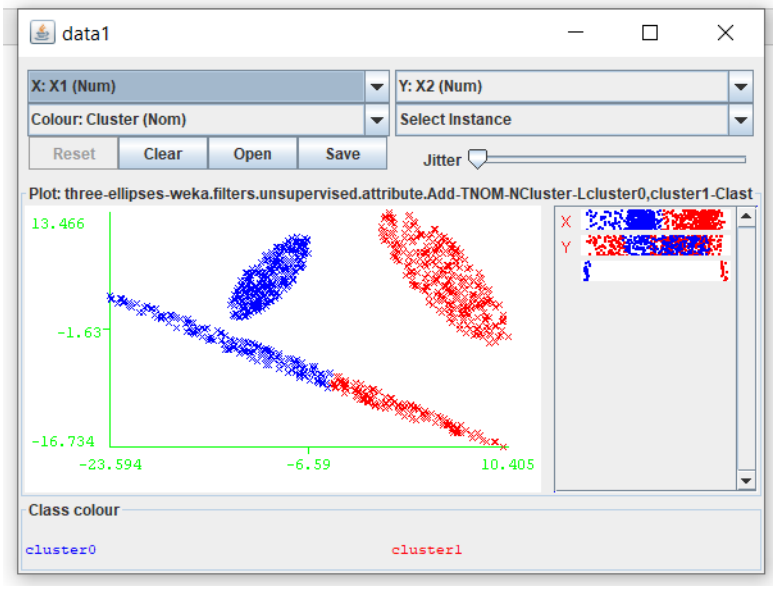


Error: 10.389671

12.3

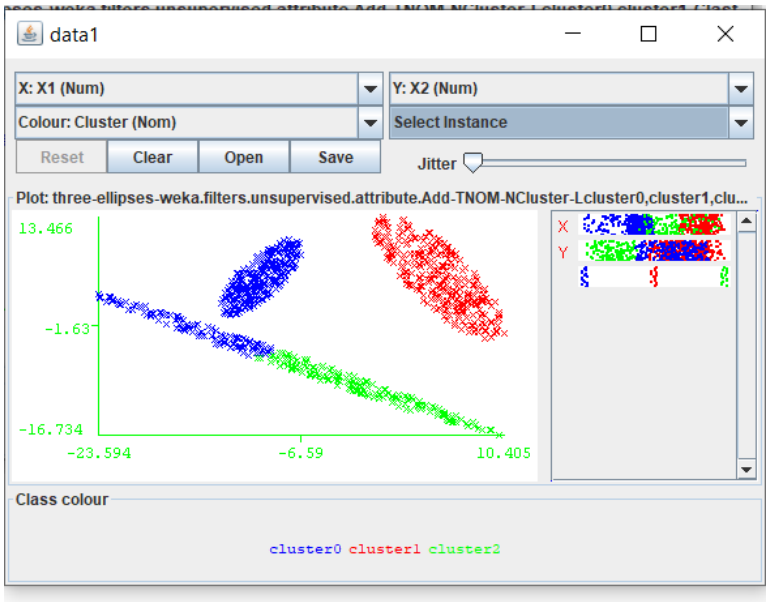
12.3.1 Plik cl-002.arff -> analiza dla różnych wartości k.

K = 2



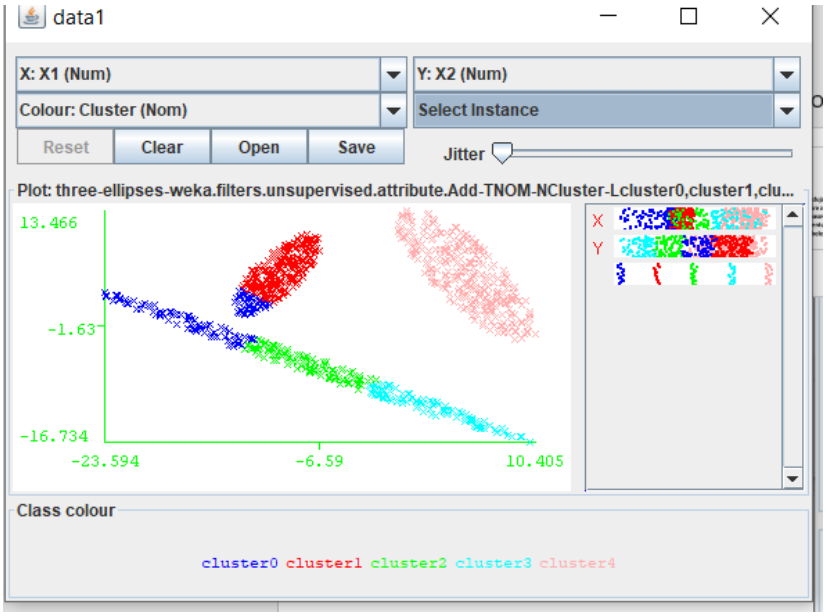
Error: 56.508511

K = 3



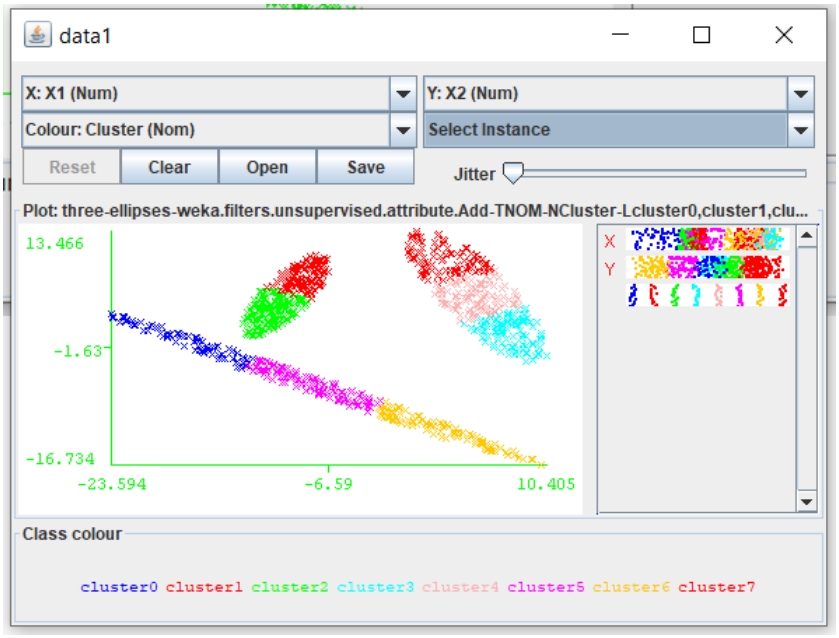
Error: 25.898306

K = 5



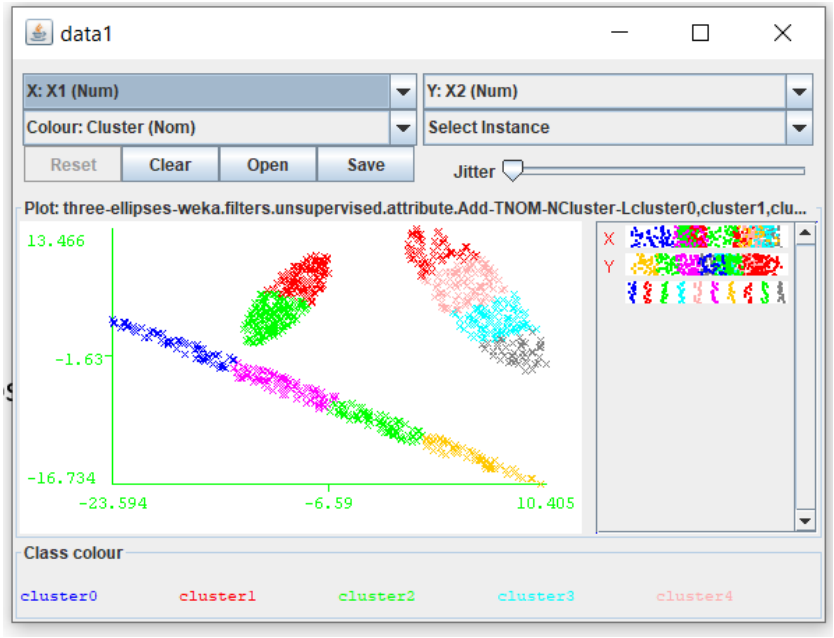
Error: 14.194627

K = 8



Error: 6.245052

K = 10

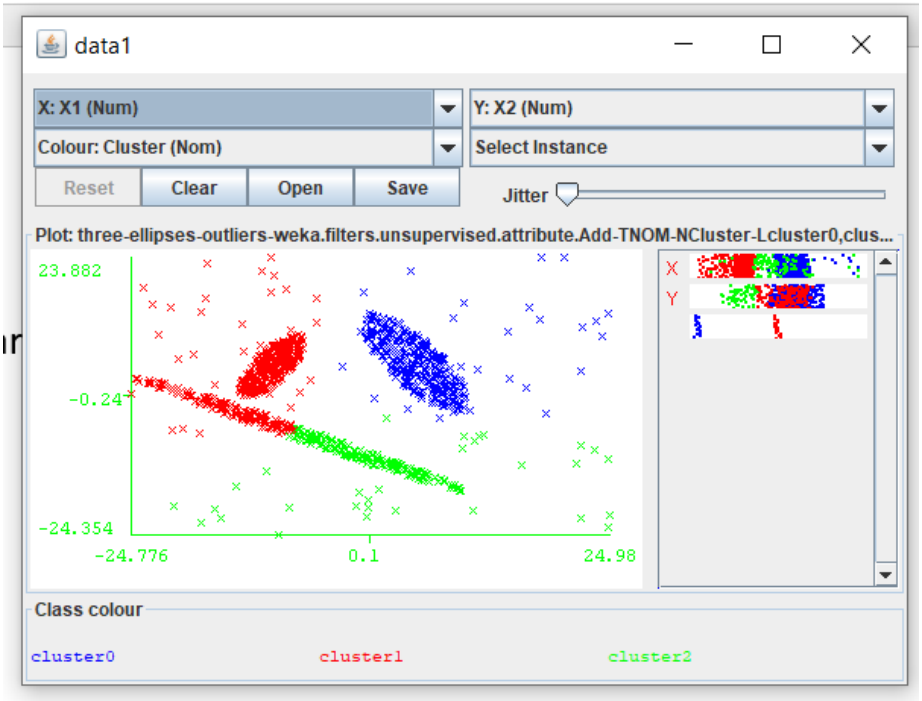


Error: 4.205950

Co się dzieje z funkcją kosztu w miarę wzrostu k?

- Wraz ze wzrostem liczby klastrów, wartość funkcji kosztu maleje, ale nie zawsze oznacza to optymalne rozwiązanie. W ostatniej wizualizacji, gdzie liczba klastrów jest znacznie większa, klasyfikacja nie jest jednoznaczna i elipsy są nieprecyzyjne. Jednak w przykładach dla k=2 i k=3 widać, że te wartości są najlepszym wyborem dla tego rodzaju danych.

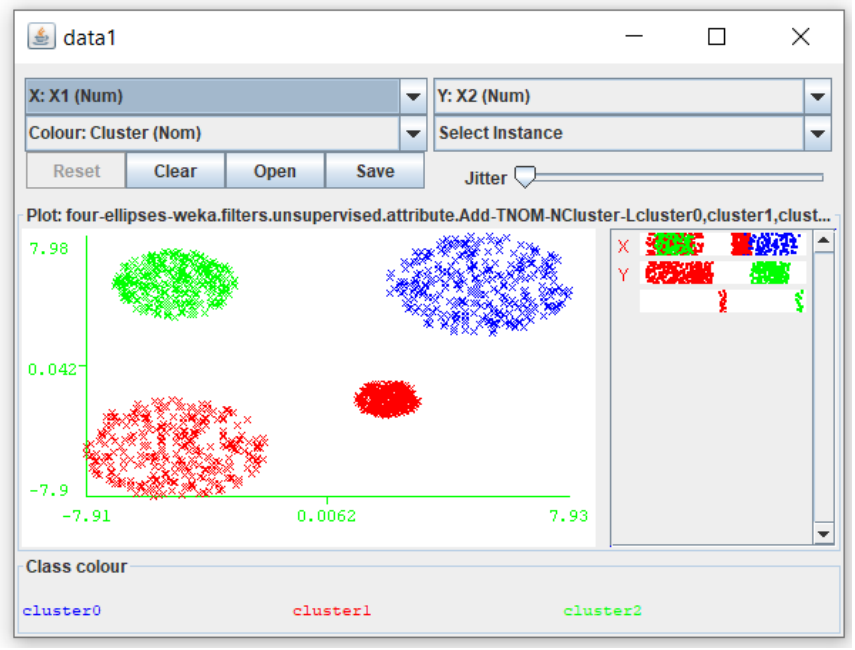
12.3.2 Plik cl-003.arff k=3



Error: 18.767046

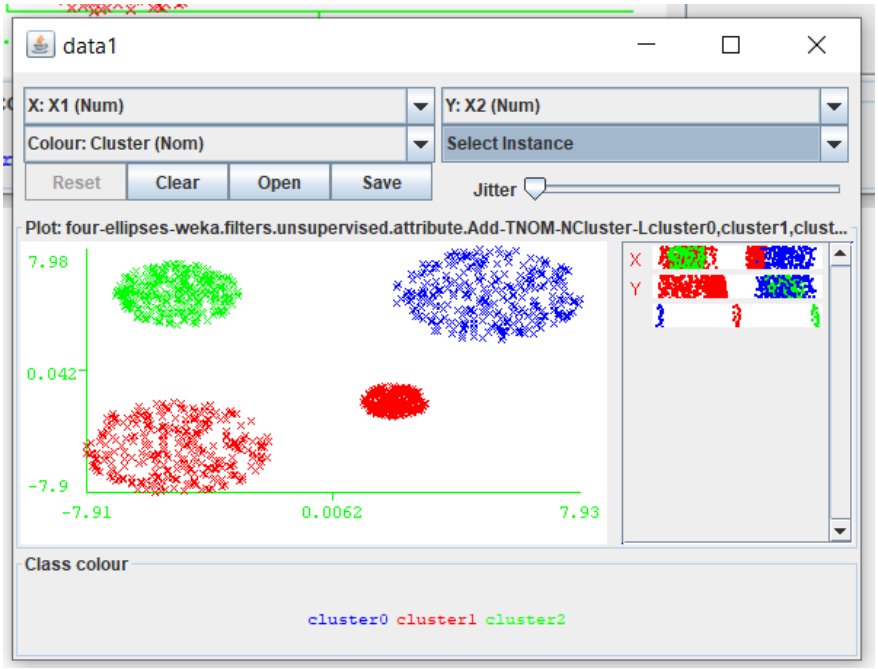
12.3.3 Plik cl-004.arff k=2, 3, 4

K = 2



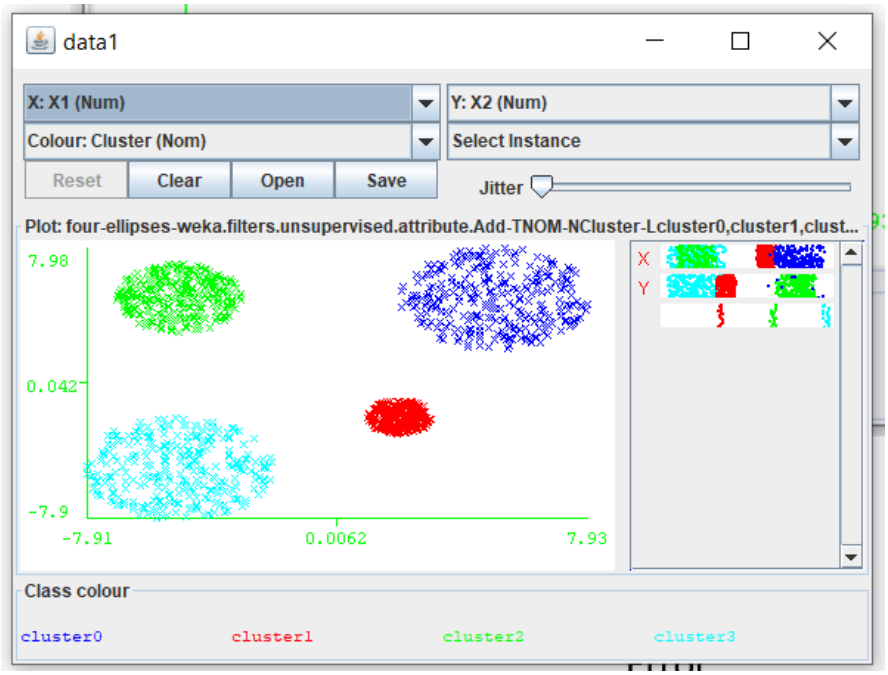
Error: 48.670713

K = 3



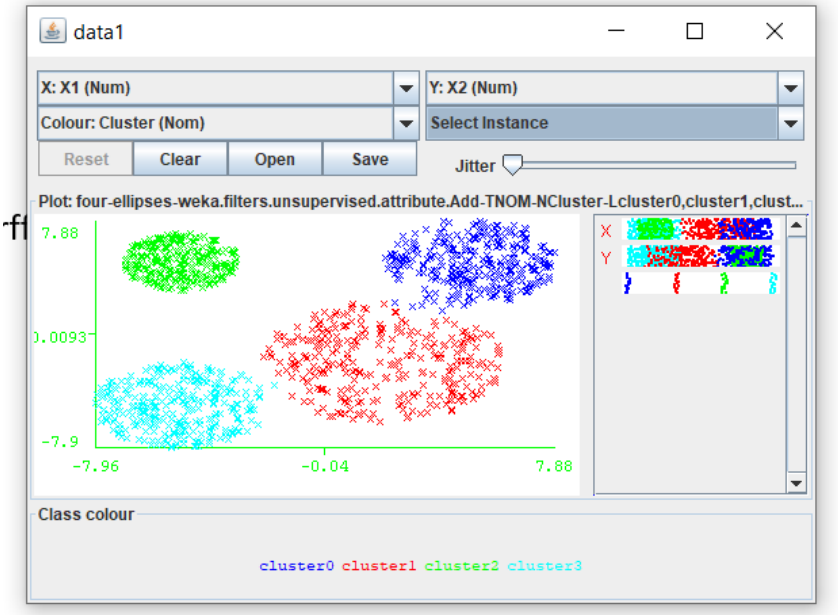
Error: 48.670713

K = 4



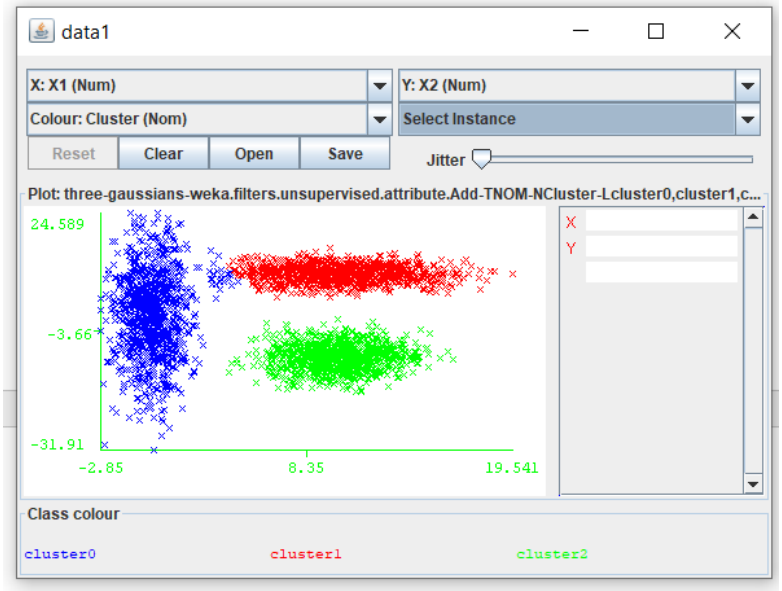
Error: 14.069859

12.3.4 Plik cl-005.arff, dobrane k = 4



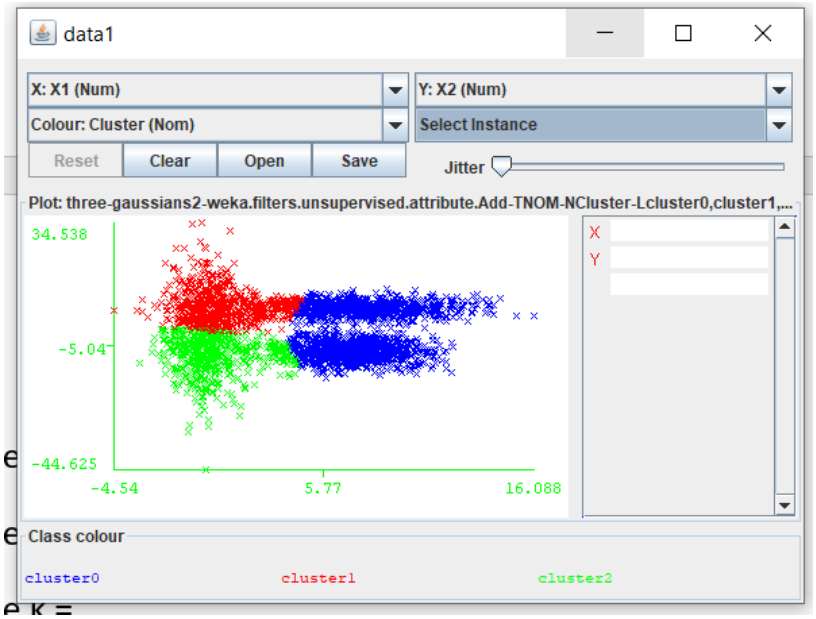
Error: 14.069859

12.3.5 Plik cl-006.arff, dobrane k = 3



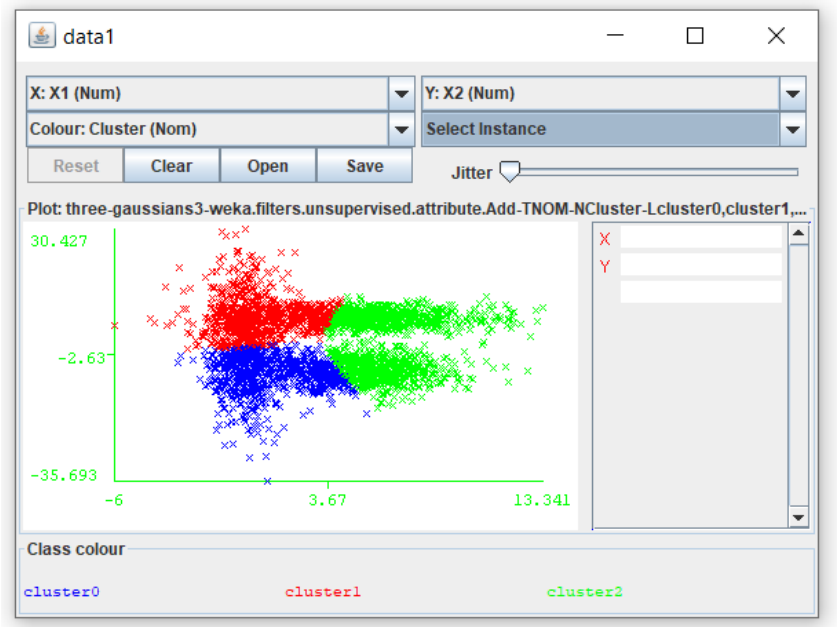
Error: 59.439722

12.3.6 Plik cl-007.arff, dobrane k = 3



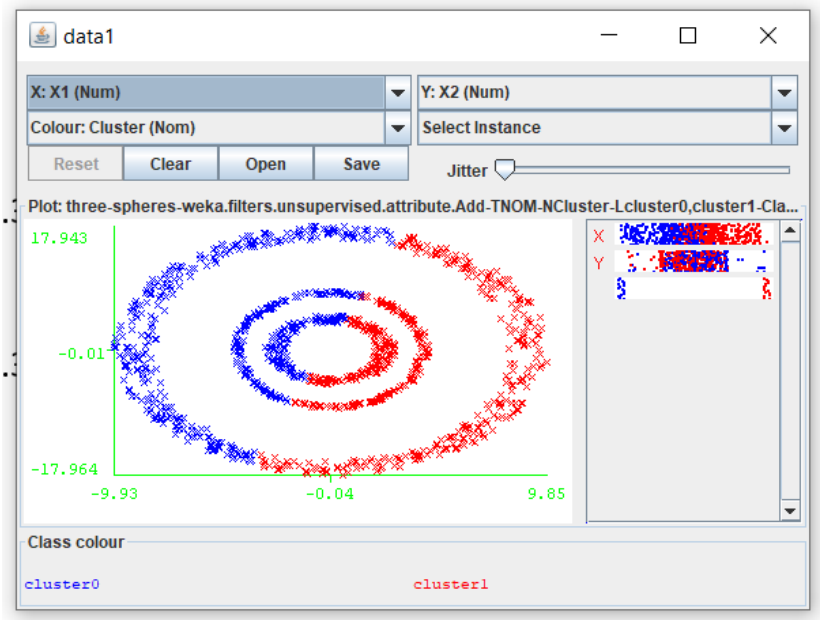
Error: 45.169816

12.3.7 Plik cl-008.arff, dobrane k = 3



Error: 47.943523

12.3.8 Plik cl-009.arff, dobrane k = 2



Error: 101.991525

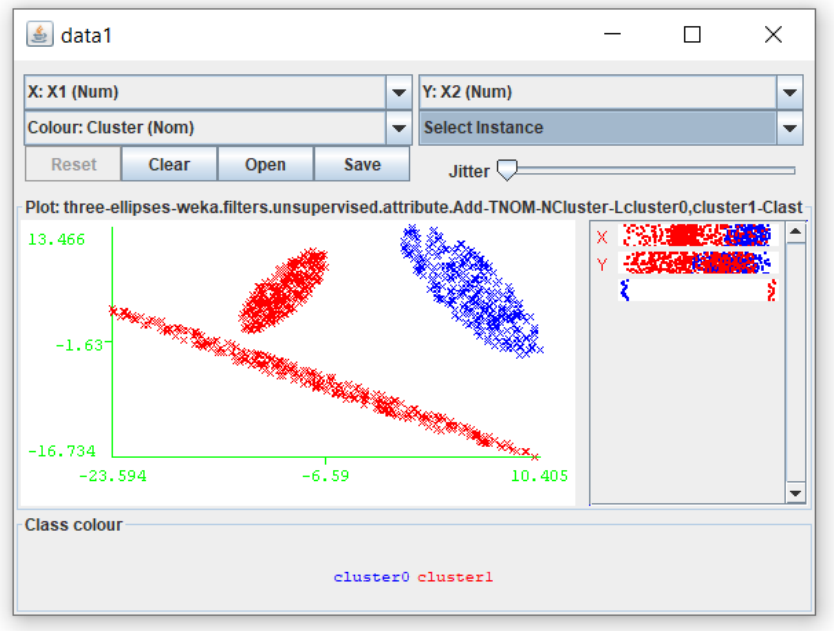
12.3.9 Plik cl-010.arff, dobrane k = 4

Error: 40.452607

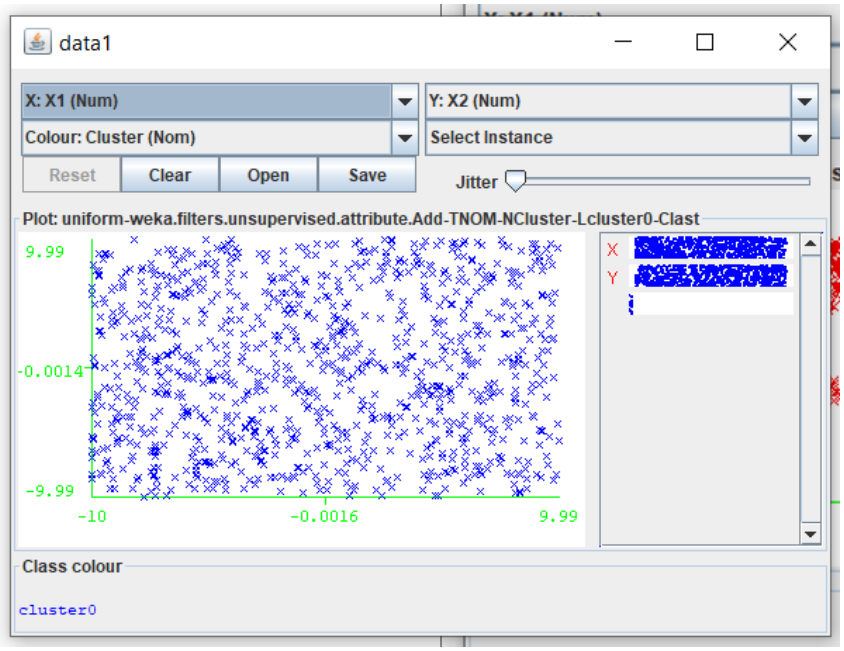
12.4

```
public static void main(String[] args) {
    try{
        ConverterUtils.DataSource source = new ConverterUtils.DataSource( location: "cl-001.arff");
        Instances data = source.getDataSet();

        DBSCAN cls = new DBSCAN();
        cls.setMinPoints(4);
        cls.setEpsilon(0.03);
        cls.buildClusterer(data);

        Add filter = new Add();
        filter.setAttributeIndex("last");
        int num  = cls.numberOfClusters();
        String labels = "cluster0";
        for(int i=1;i<num;i++){
            labels+=", cluster";
            labels+=i;
        }
        filter.setNominalLabels(labels);
        filter.setAttributeName("Cluster");
        filter.setInputFormat(data);
        Instances newData = Filter.useFilter(data, filter);


        int idx = newData.numAttributes()-1;
        for(int i=0;i<newData.numInstances();i++){
            try{
                int val = cls.clusterInstance(data.get(i));
                if(val==-1)continue;
                newData.get(i).setValue(idx, val);
            }
            catch(Exception e){}
        }

        visualize(newData,  title: "data1");
```

12.4.1 cl-002.arff



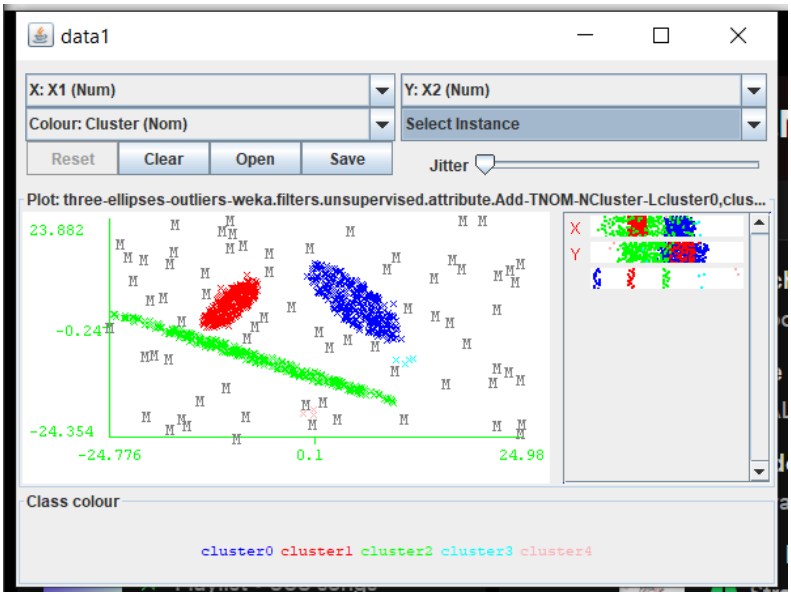
12.4.2 cl-001.arff



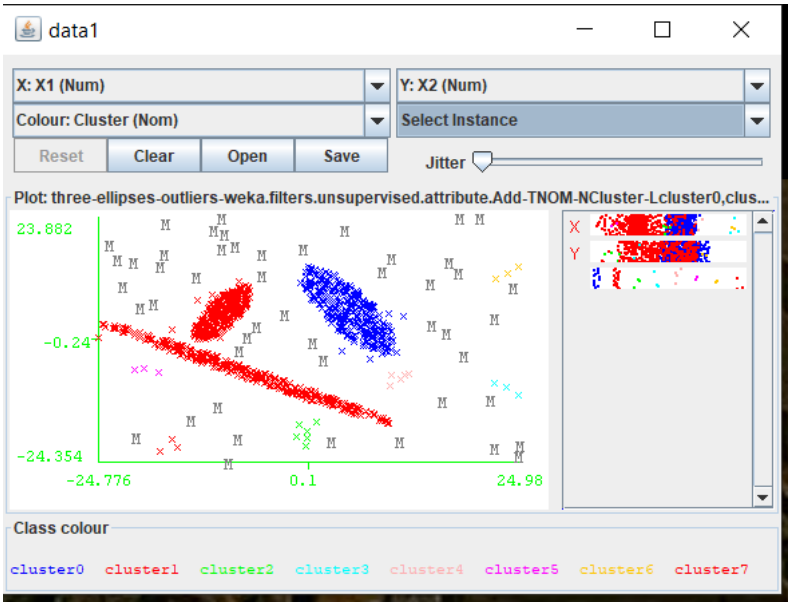
Został wykryty 1 klaster. Wynika to z szumu, który jest jednorodny.

12.4.3 cl-003.arff

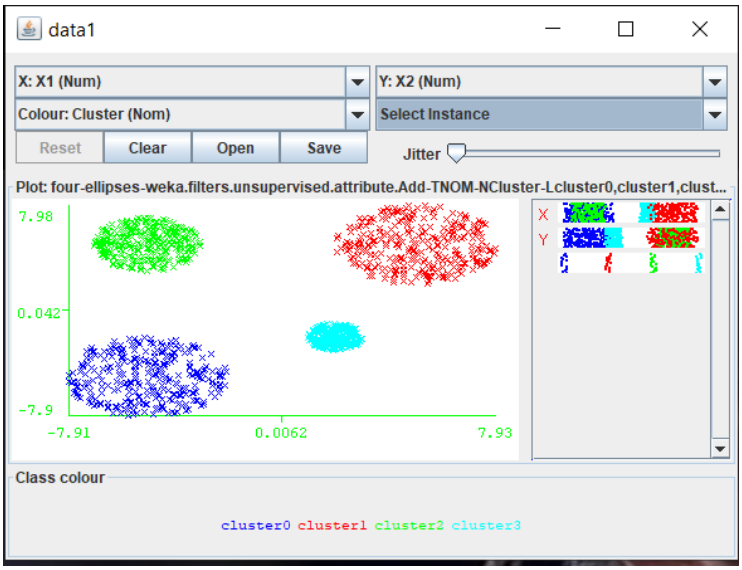
Epsilon = 0.03



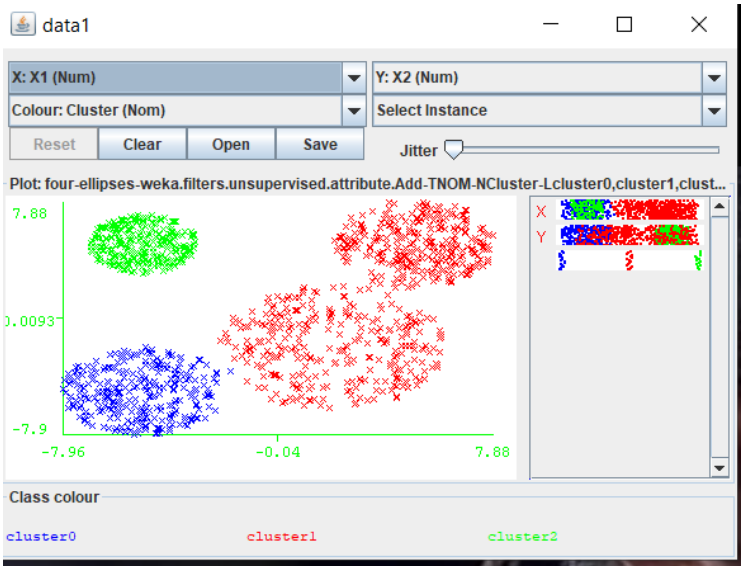
Epsilon = 0.05



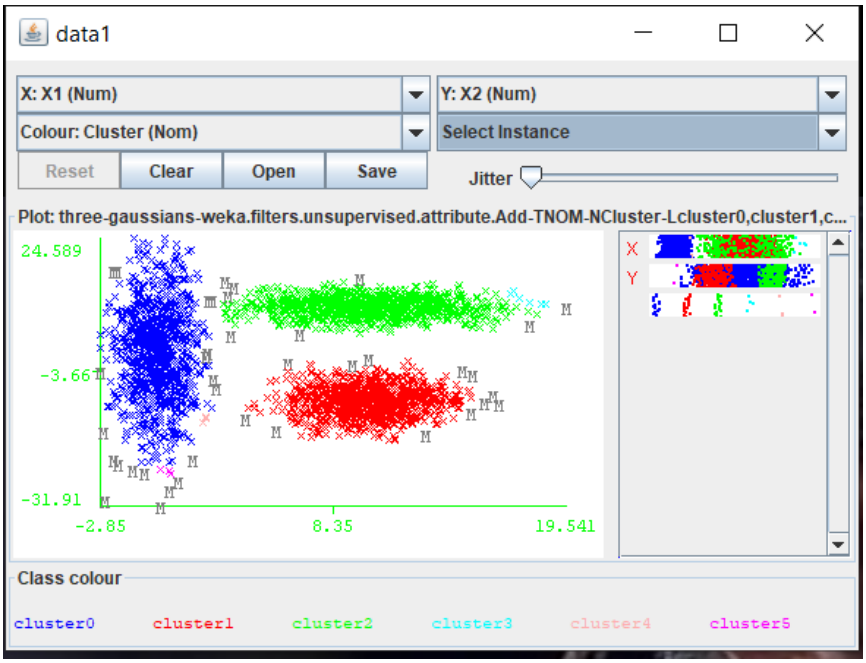
12.4.4 cl-0004.arff z dobranym epsilon = 0.2



12.4.5 cl-0005.arff z dobranym epsilon = 0.05

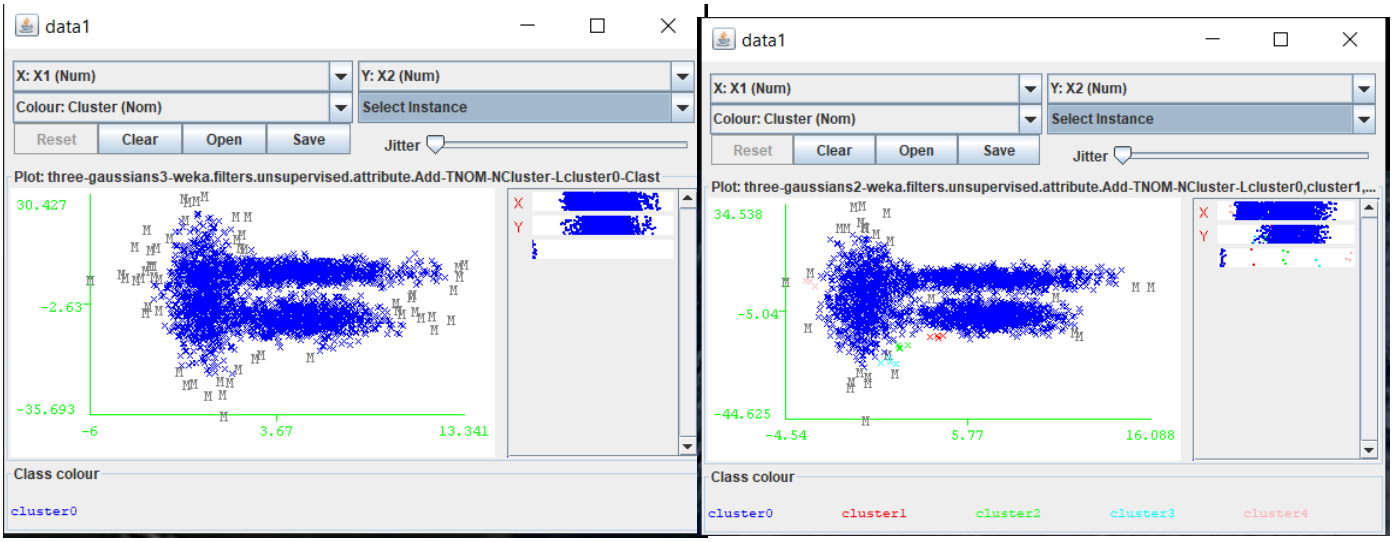


12.4.6 cl-0006.arff z dobranym epsilon = 0.03



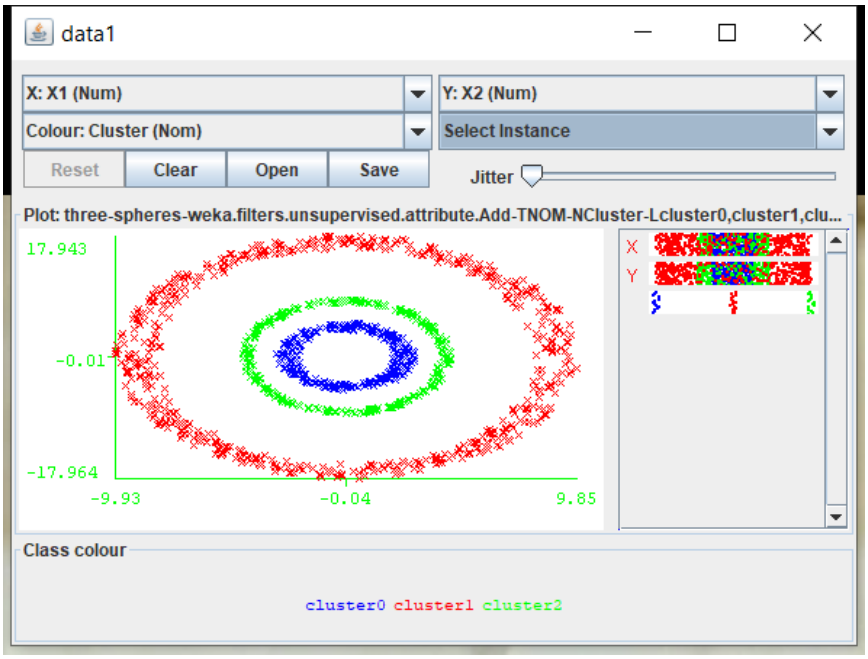
12.4.7 cl-0007.arff i cl-0008.arff z dobranym epsilon = 0.01

Dla tych zestawów danych nie udało mi się dobrać odpowiednich parametrów. Być może wynika to z bliskiego rozmieszczenia danych na płaszczyźnie co czyni te dane trudno rozróżnialne dla algorytmu DBSCAN.



cl-007.arff i cl-008.arff

12.4.8 cl-0009.arff z dobranym epsilon = 0.04



12.4.9 cl-0010.arff z dobranym epsilon = 0.03

12.5

12.5.1

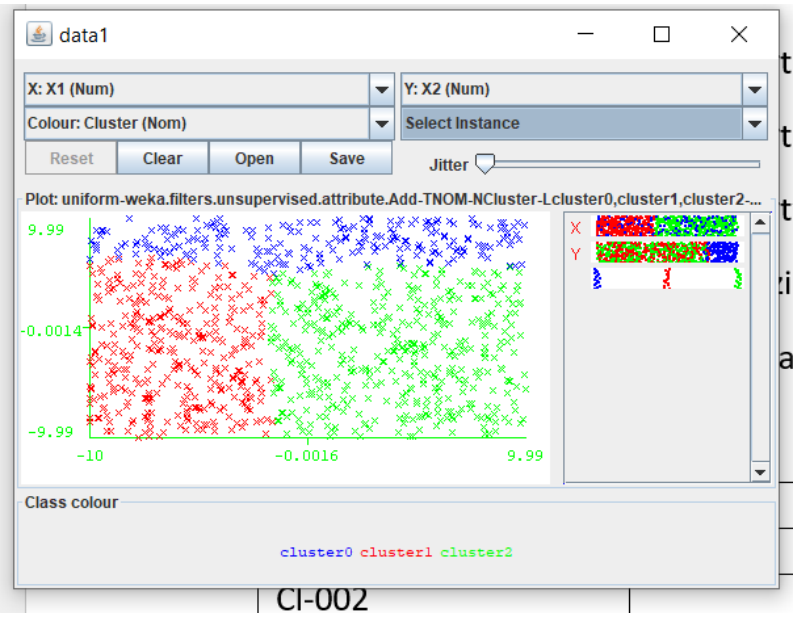
```
for(int i=0;i<data.numInstances();i++){
    double density = cls.logDensityForInstance(data.get(i));
    logLikehood+=density;
}
System.out.printf(Locale.US,  s: "LL: %f", ...objects: logLikehood/data.numInstances());
```

12.5.2

```
int k =3;

EM cls = new EM();
cls.setNumClusters(k);
cls.setSeed(10);
cls.buildClusterer(data);
```

12.5.3

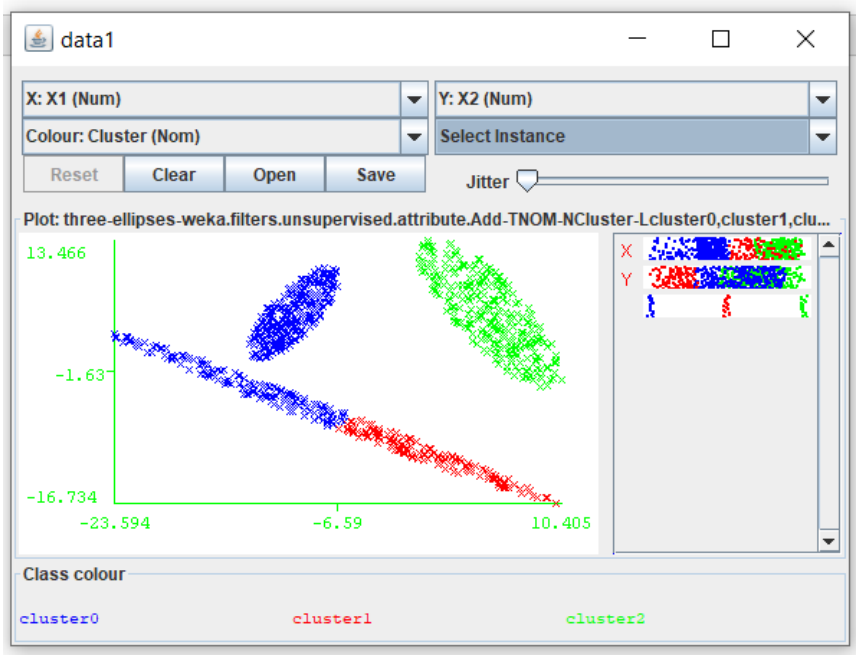
Cl-001.arff



Cl-002

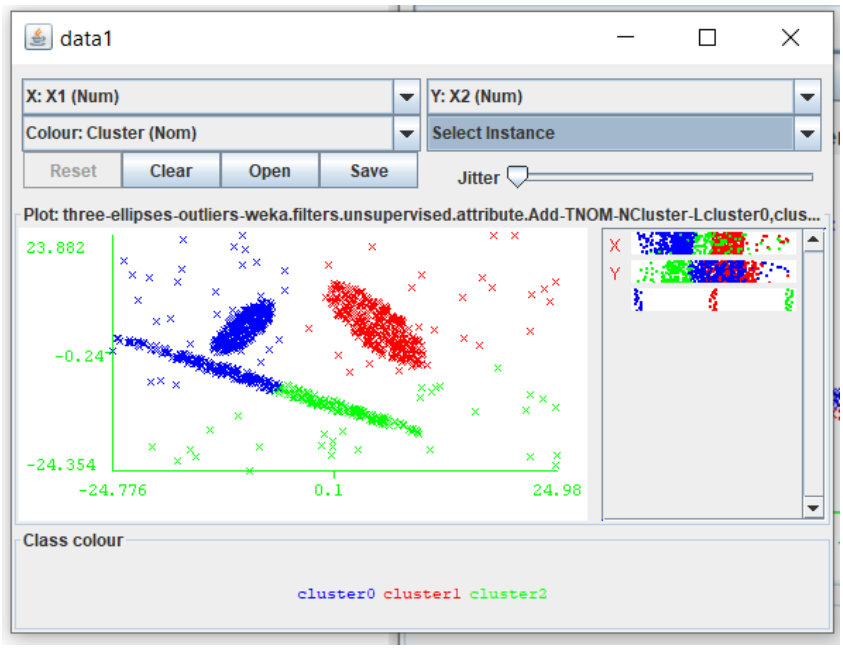
LL: -6.194026

Cl-002.arff



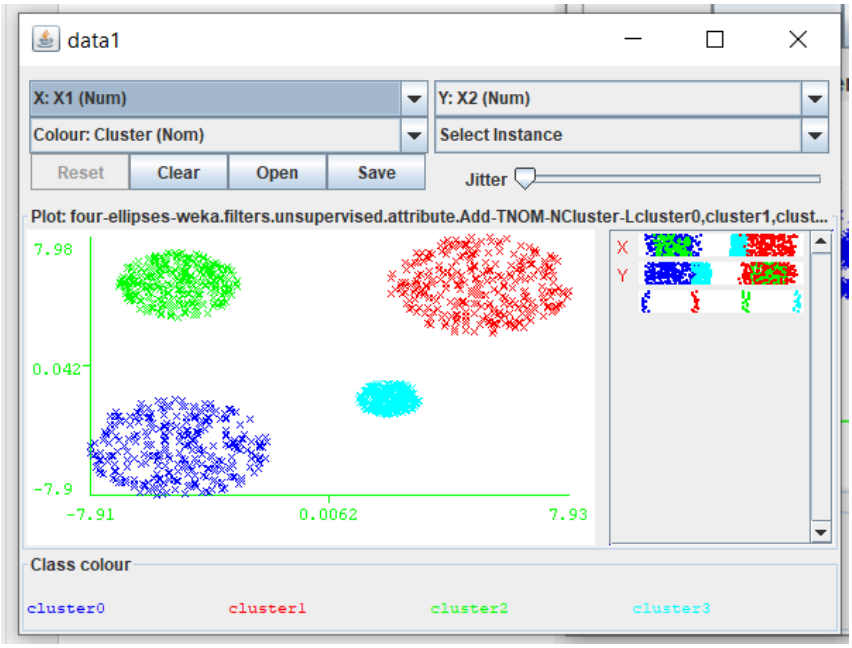
LL: -6.435150

Cl-003.arff



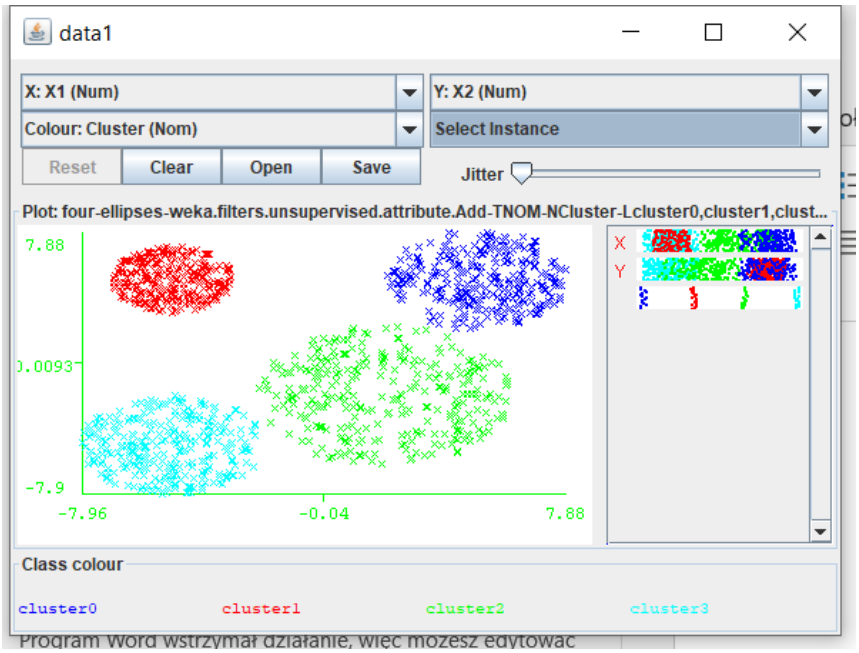
LL: -6.856417

Cl-004.arff k =4



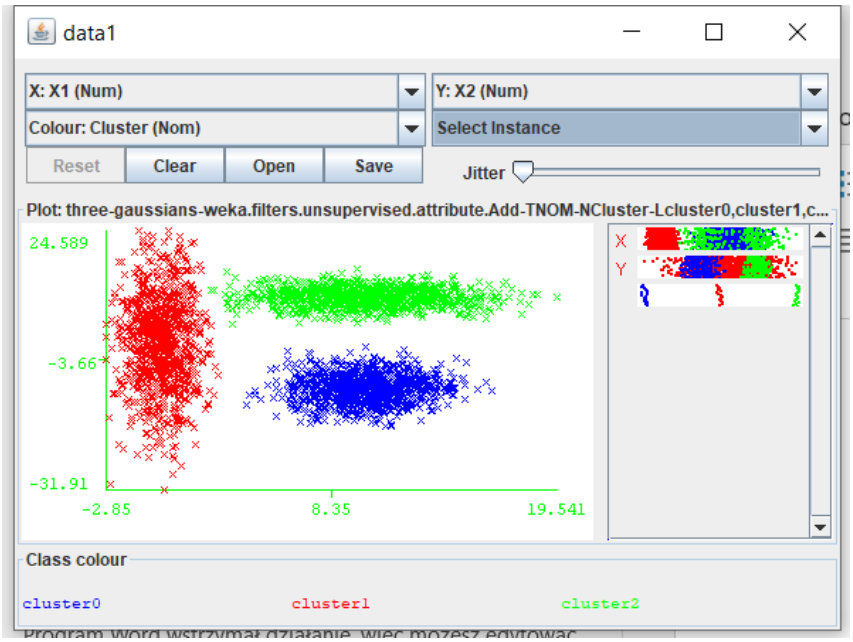
LL: -4.306211

Cl-005.arff k = 4



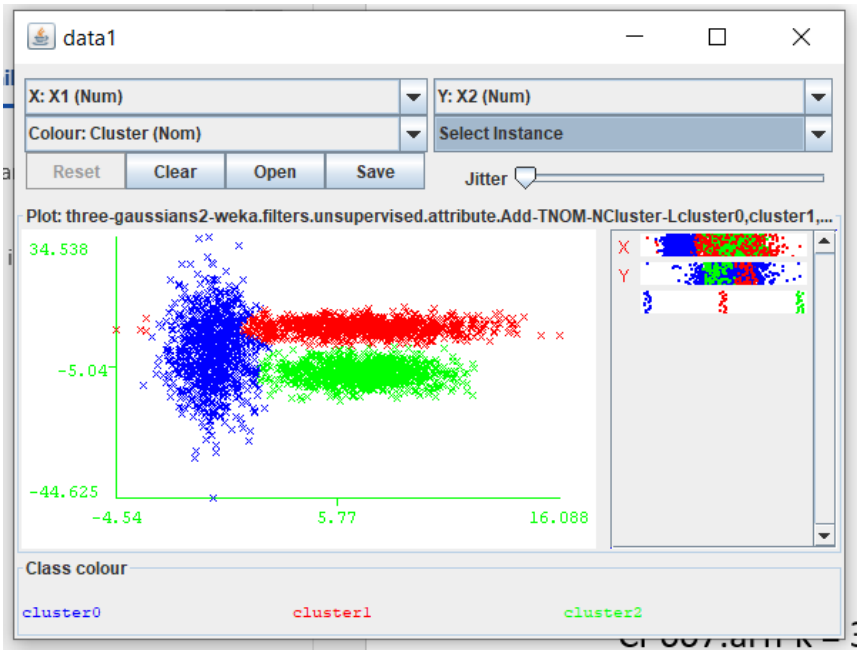
LL: -4.974675

Cl-006.arff k =3



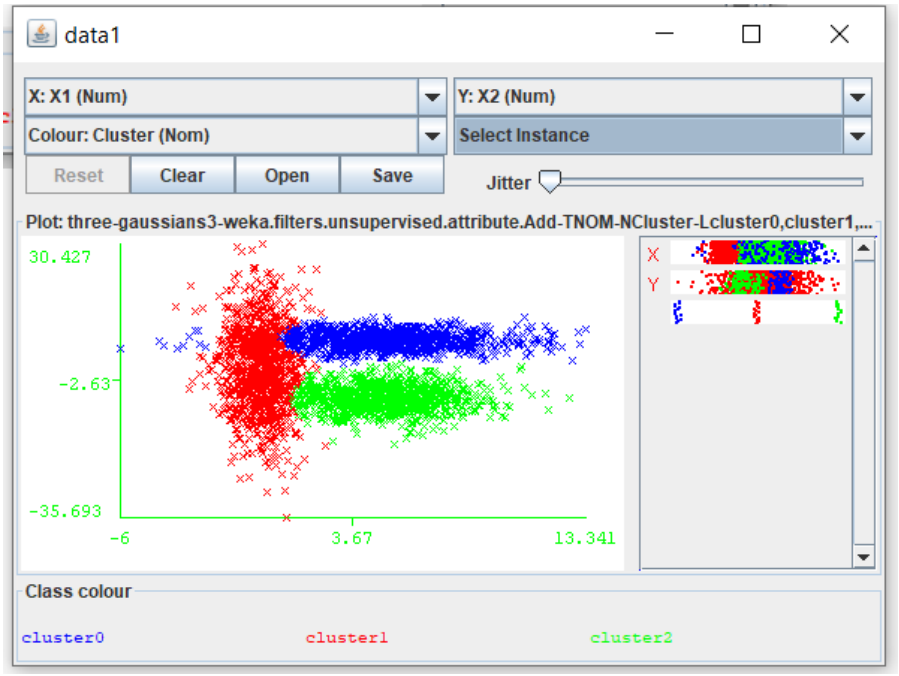
LL: -5.876975

Cl-007.arff k = 3



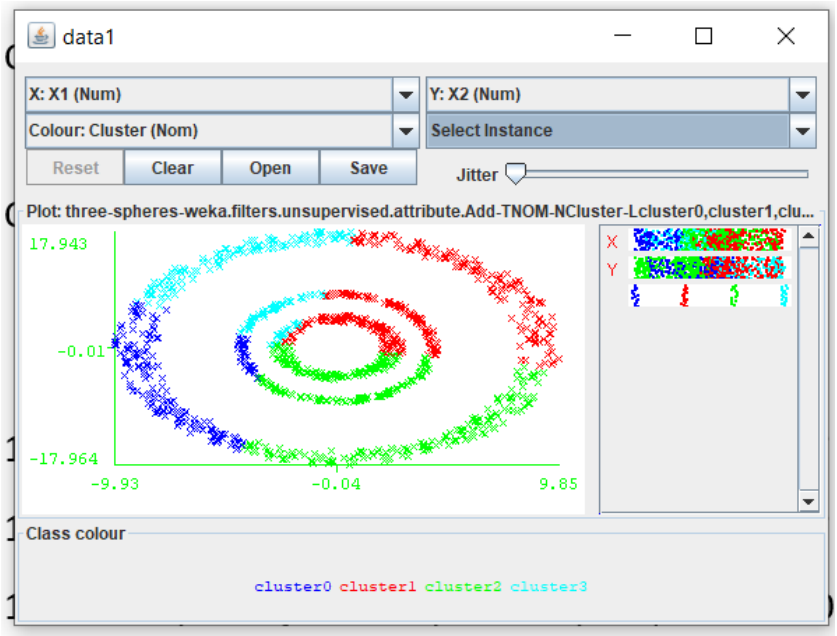
LL: -5.842824

Cl-008.arff k = 3



LL: -5.727114

Cl-009.arff k = 4



LL: -6.607904

Cl-010.arff k = 4

LL: -6.581815

12.5.3.1 Czy da się odtworzyć kształt elips dla cl-002 i cl-003? Wyjaśnij.

Wystąpią problemy z odtworzeniem elips dla tych zestawów danych. Niezależnie od wartości k, nie jest możliwe otrzymanie rezultatu, który pozwoliłby odtworzyć elipsy.

12.5.3.2 Czy da się odtworzyć kształt elips dla cl-004 i cl-005?

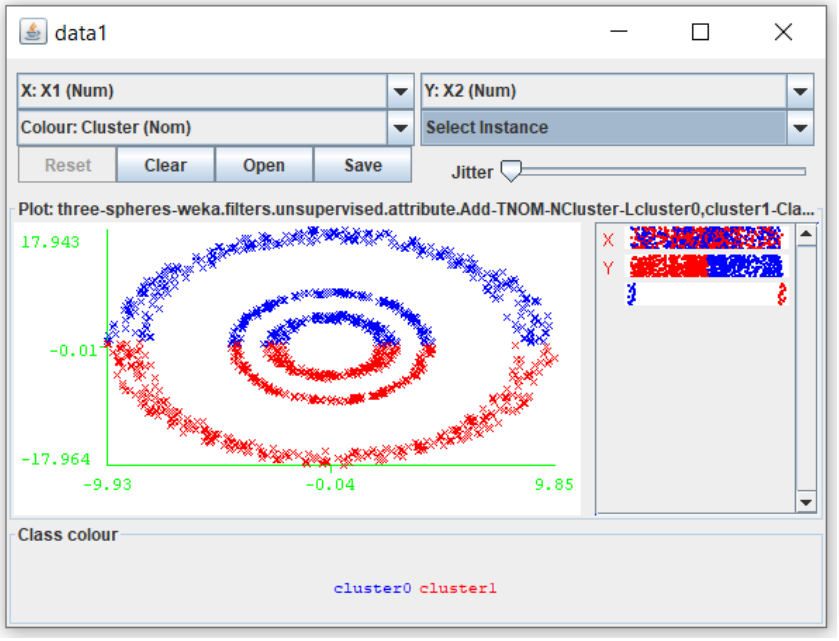
Odtworzenie elips dla tych zestawów danych jest możliwe (dla parametru k = 4).

12.5.3.3 Czy da się odtworzyć kształty skupisk dla cl-006, cl-007 i cl008?

Tak, da się odtworzyć kształty skupisk dla tych zestawów danych.

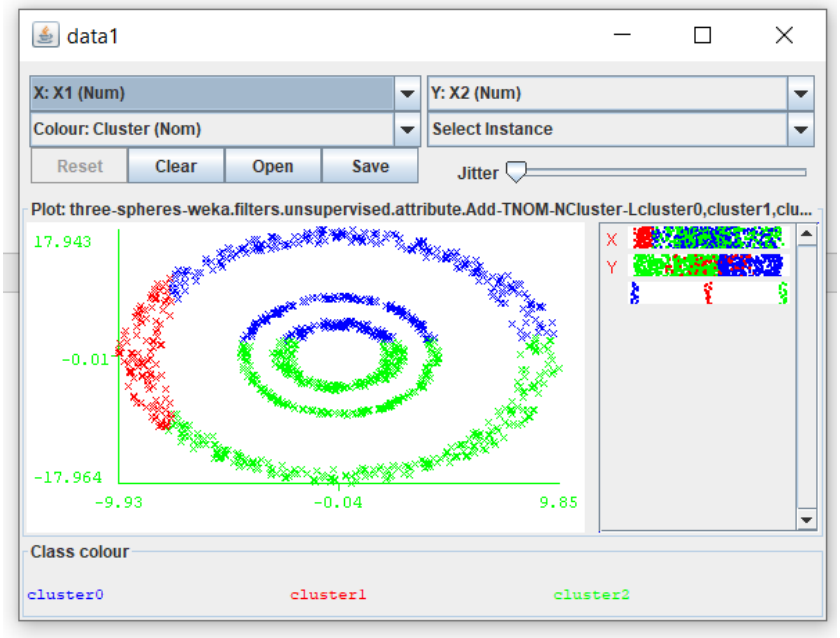
12.5.3.4 Zbiór: cl-009 - przetestuj działanie algorytmu dla k=2,3,4

K = 2



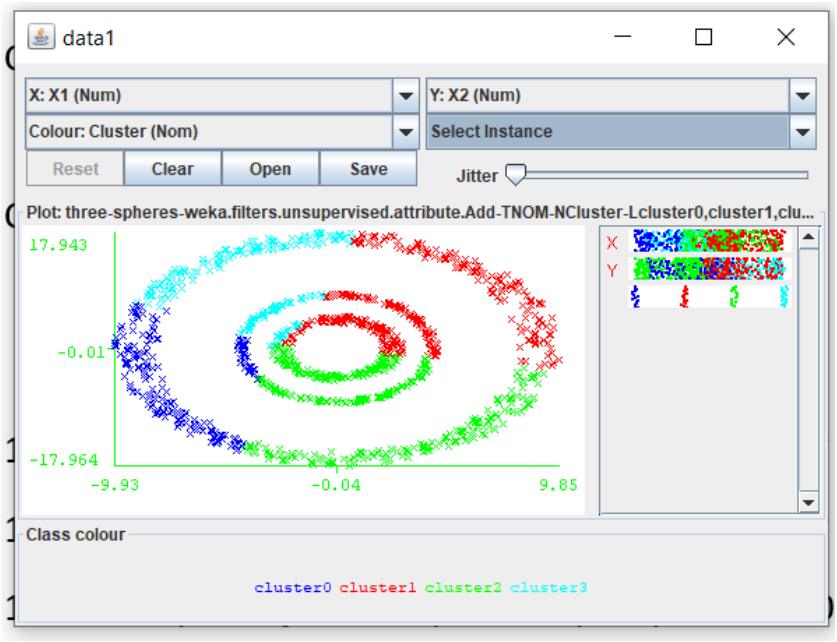
LL: -6.620152

K = 3



LL: -6.559911

K = 4



LL: -6.607904

12.5.3.5 Zbiór: cl-010 - przetestuj dla k=4,8

K = 4

K = 8

12.6

Zbiór danych	K-means	DBSCAN	EM
Cl-001	słaby	słaby	średni
Cl-002	średni	słaby	średni
Cl-003	średni	słaby	średni
Cl-004	dobry	dobry	dobry
Cl-005	dobry	słaby	dobry
Cl-006	dobry	średni	dobry
Cl-007	średni	słaby	dobry
Cl-008	średni	słaby	dobry
Cl-009	słaby	dobry	słaby
Cl-010	dobry	dobry	dobry