

Lab 3

732A61 Data Mining - Clustering and Association Analysis

Carles Sans Fuentes

March 6, 2017

Assignment 1

In this assignment I am going to work with the data set monk1.arff. This is an artificial dataset with 124 instances, each described by 6 discrete attributes and a binary class attribute.

Clustering

First, I am asked to cluster the data with different algorithms and number of clusters. The ones that are going to be used are the (1) Simplekmeans and (2) the XXX with (3) 3 Clusters and (4) 4 Cluster such that this are 4 options. I am going to use the clusters to class evaluation model to see whether the clustering algorithm is able to discover the class division existing in the data.

Exercise 1 Clustering

3 Clusters k-means

The algorithm "SimpleKMeans" has been applied to your data. In Weka euclidian distance is implemented in SimpleKmeans and the seed chosen for this case (ignoring the class attribute) is 10. The results are provided here below for three clusters:

```
1 === Run information ===
2
3 Scheme:          weka.clusterers.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 first-last" -I
   500 -num-slots 1 -S 10
4 Relation:        monk1
5 Instances:        124
6 Attributes:       7
7                   attribute#1
8                   attribute#2
9                   attribute#3
10                  attribute#4
11                  attribute#5
12                  attribute#6
13 Ignored:         class
14 Test mode:       evaluate on training data
15
16
17
18 === Clustering model (full training set) ===
19
20
21 kMeans
22 =====
23
24 Number of iterations: 3
25 Within cluster sum of squared errors: 314.0
26
27 Initial starting points (random):
28
29 Cluster 0: 1,2,1,1,1,2
```

```

30 Cluster 1: 3,2,2,3,2,1
31 Cluster 2: 2,3,1,2,1,1
32
33 Missing values globally replaced with mean/mode
34
35 Final cluster centroids:
36
37 Attribute      Full Data      Cluster#
38                (124)        (59)        (38)        (27)
39 =====
40 attribute#1      1            1            3            2
41 attribute#2      3            2            1            3
42 attribute#3      1            1            2            1
43 attribute#4      3            1            3            2
44 attribute#5      4            3            2            1
45 attribute#6      2            2            1            1
46
47
48
49
50 Time taken to build model (full training data) : 0.03 seconds
51
52 === Model and evaluation on training set ===
53
54 Clustered Instances
55
56 0          59 ( 48%)
57 1          38 ( 31%)
58 2          27 ( 22%)

```

4 Clusters k-means

The algorithm "SimpleKMeans" has been applied to your data for 4 clusters. The seed chosen for this case (ignoring the class attribute) is 10. The results are provided here below for four clusters:

```

1  === Run information ===
2
3  Scheme:      weka.clusterers.SimpleKMeans -init 0 -max-candidates 100 -periodic-pruning 10000
               -min-density 2.0 -t1 -1.25 -t2 -1.0 -N 4 -A "weka.core.EuclideanDistance -R first-last" -I
               500 -num-slots 1 -S 10
4  Relation:    monk1
5  Instances:   124
6  Attributes:  7
7               attribute#1
8               attribute#2
9               attribute#3
10              attribute#4
11              attribute#5
12              attribute#6
13 Ignored:
14              class
15 Test mode:    evaluate on training data
16
17
18 === Clustering model (full training set) ===
19
20
21 kMeans
22 =====
23
24 Number of iterations: 3
25 Within cluster sum of squared errors: 293.0
26
27 Initial starting points (random):
28
29 Cluster 0: 1,2,1,1,1,2
30 Cluster 1: 3,2,2,3,2,1
31 Cluster 2: 2,3,1,2,1,1
32 Cluster 3: 2,3,1,3,1,2
33
34 Missing values globally replaced with mean/mode
35
36 Final cluster centroids:
37
38 Attribute      Full Data      Cluster#
39                (124)        (50)        (36)        (24)        (14)
40 =====
41 attribute#1      1            1            3            2            2
42 attribute#2      3            2            1            3            3
43 attribute#3      1            1            2            1            1
44 attribute#4      3            1            3            2            3

```

```

45 attribute#5          4          3          2          1          1
46 attribute#6          2          2          1          1          2
47
48
49
50
51 Time taken to build model (full training data) : 0.01 seconds
52
53 === Model and evaluation on training set ===
54
55 Clustered Instances
56
57 0          50 ( 40%)
58 1          36 ( 29%)
59 2          24 ( 19%)
60 3          14 ( 11%)

```

Makedensitybasedclusterer with clusterer kmeans with 3 clusters

The algorithm "makedensitybasedclusterer" has been applied to the data with default min standard deviation equals to 1.0E-6. As a clusterer for the algorithm, the SimpleKmeans with 3 clusters has been used as well as a seed chosen for this case (ignoring the class attribute) which is 10. The results are provided here below for three clusters:

```

1 === Run information ===
2
3 Scheme:          weka.clusterers.MakeDensityBasedClusterer -M 1.0E-6 -W weka.clusterers.
                  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 first-last" -I 500 -num-slots 1 -S
                  10
4 Relation:        monk1-weka.filters.unsupervised.attribute.AddCluster-Wweka.clusterers.
                  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 first-last" -I 500 -num-slots 1 -S 10-I7
5 Instances:       124
6 Attributes:      8
7                  attribute#1
8                  attribute#2
9                  attribute#3
10                 attribute#4
11                 attribute#5
12                 attribute#6
13 Ignored:
14                 class
15                 cluster
16 Test mode:       evaluate on training data
17
18
19 === Clustering model (full training set) ===
20
21 MakeDensityBasedClusterer:
22
23 Wrapped clusterer:
24 kMeans
25 =====
26
27 Number of iterations: 3
28 Within cluster sum of squared errors: 314.0
29
30 Initial staring points (random):
31
32 Cluster 0: 1,2,1,1,1,2
33 Cluster 1: 3,2,2,3,2,1
34 Cluster 2: 2,3,1,2,1,1
35
36 Missing values globally replaced with mean/mode
37
38 Final cluster centroids:
39
40 Attribute          Full Data          Cluster#
41                   (124)          0          1          2
42                   (124)          (59)          (38)          (27)
43 =====
44 attribute#1          1          1          3          2
45 attribute#2          3          2          1          3
46 attribute#3          1          1          2          1
47 attribute#4          3          1          3          2
48 attribute#5          4          3          2          1
49 attribute#6          2          2          1          1
50
51

```

```

52 Fitted estimators (with ML estimates of variance):
53
54 Cluster: 0 Prior probability: 0.4724
55
56 Attribute: attribute#1
57 Discrete Estimator. Counts = 32 17 13 (Total = 62)
58 Attribute: attribute#2
59 Discrete Estimator. Counts = 9 34 19 (Total = 62)
60 Attribute: attribute#3
61 Discrete Estimator. Counts = 37 24 (Total = 61)
62 Attribute: attribute#4
63 Discrete Estimator. Counts = 30 14 18 (Total = 62)
64 Attribute: attribute#5
65 Discrete Estimator. Counts = 8 13 23 19 (Total = 63)
66 Attribute: attribute#6
67 Discrete Estimator. Counts = 16 45 (Total = 61)
68
69 Cluster: 1 Prior probability: 0.3071
70
71 Attribute: attribute#1
72 Discrete Estimator. Counts = 9 10 22 (Total = 41)
73 Attribute: attribute#2
74 Discrete Estimator. Counts = 21 7 13 (Total = 41)
75 Attribute: attribute#3
76 Discrete Estimator. Counts = 9 31 (Total = 40)
77 Attribute: attribute#4
78 Discrete Estimator. Counts = 11 9 21 (Total = 41)
79 Attribute: attribute#5
80 Discrete Estimator. Counts = 9 17 6 10 (Total = 42)
81 Attribute: attribute#6
82 Discrete Estimator. Counts = 24 16 (Total = 40)
83
84 Cluster: 2 Prior probability: 0.2205
85
86 Attribute: attribute#1
87 Discrete Estimator. Counts = 7 18 5 (Total = 30)
88 Attribute: attribute#2
89 Discrete Estimator. Counts = 8 4 18 (Total = 30)
90 Attribute: attribute#3
91 Discrete Estimator. Counts = 22 7 (Total = 29)
92 Attribute: attribute#4
93 Discrete Estimator. Counts = 4 19 7 (Total = 30)
94 Attribute: attribute#5
95 Discrete Estimator. Counts = 15 4 4 8 (Total = 31)
96 Attribute: attribute#6
97 Discrete Estimator. Counts = 19 10 (Total = 29)
98
99
100 Time taken to build model (full training data) : 0.09 seconds
101
102 === Model and evaluation on training set ===
103
104 Clustered Instances
105
106 0      60 ( 48%)
107 1      39 ( 31%)
108 2      25 ( 20%)
109
110
111 Log likelihood: -6.09108

```

Makedensitybasedclusterer with clusterer kmeans with 4 clusters

The algorithm "makedensitybasedclusterer" has been applied to the data with default min standard deviation equals to 1.0E-6. As a clusterer for the algorithm, the SimpleKmeans with 4 clusters has been used as well as a seed chosen for this case (ignoring the class attribute) which is 10. The results are provided here below for four clusters:

```

1 === Run information ===
2
3 Scheme:      weka.clusterers.MakeDensityBasedClusterer -M 1.0E-6 -W weka.clusterers.
               SimpleKMeans -- -init 0 -max-candidates 100 -periodic-pruning 10000 -min-density 2.0 -t1
               -1.25 -t2 -1.0 -N 4 -A "weka.core.EuclideanDistance -R first-last" -I 500 -num-slots 1 -S
               10
4 Relation:    monki-weka.filters.unsupervised.attribute.AddCluster-Wweka.clusterers.
               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 first-last" -I 500 -num-slots 1 -S 10-I7
5 Instances:   124
6 Attributes:  8
7              attribute#1

```

```

8         attribute#2
9         attribute#3
10        attribute#4
11        attribute#5
12        attribute#6
13 Ignored:
14         class
15         cluster
16 Test mode: evaluate on training data
17
18
19 === Clustering model (full training set) ===
20
21 MakeDensityBasedClusterer:
22
23 Wrapped clusterer:
24 kMeans
25 =====
26
27 Number of iterations: 3
28 Within cluster sum of squared errors: 293.0
29
30 Initial starting points (random):
31
32 Cluster 0: 1,2,1,1,1,2
33 Cluster 1: 3,2,2,3,2,1
34 Cluster 2: 2,3,1,2,1,1
35 Cluster 3: 2,3,1,3,1,2
36
37 Missing values globally replaced with mean/mode
38
39 Final cluster centroids:
40
41 Attribute      Full Data      Cluster#
42                (124)          0          1          2          3
43                (124)          (50)         (36)         (24)         (14)
44 =====
45 attribute#1      1            1            3            2            2
46 attribute#2      3            2            1            3            3
47 attribute#3      1            1            2            1            1
48 attribute#4      3            1            3            2            3
49 attribute#5      4            3            2            1            1
50 attribute#6      2            2            1            1            2
51
52
53 Fitted estimators (with ML estimates of variance):
54
55 Cluster: 0 Prior probability: 0.3984
56
57 Attribute: attribute#1
58 Discrete Estimator. Counts = 28 13 12 (Total = 53)
59 Attribute: attribute#2
60 Discrete Estimator. Counts = 9 31 13 (Total = 53)
61 Attribute: attribute#3
62 Discrete Estimator. Counts = 30 22 (Total = 52)
63 Attribute: attribute#4
64 Discrete Estimator. Counts = 30 14 9 (Total = 53)
65 Attribute: attribute#5
66 Discrete Estimator. Counts = 6 11 22 15 (Total = 54)
67 Attribute: attribute#6
68 Discrete Estimator. Counts = 16 36 (Total = 52)
69
70 Cluster: 1 Prior probability: 0.2891
71
72 Attribute: attribute#1
73 Discrete Estimator. Counts = 9 10 20 (Total = 39)
74 Attribute: attribute#2
75 Discrete Estimator. Counts = 21 7 11 (Total = 39)
76 Attribute: attribute#3
77 Discrete Estimator. Counts = 8 30 (Total = 38)
78 Attribute: attribute#4
79 Discrete Estimator. Counts = 11 9 19 (Total = 39)
80 Attribute: attribute#5
81 Discrete Estimator. Counts = 8 16 6 10 (Total = 40)
82 Attribute: attribute#6
83 Discrete Estimator. Counts = 24 14 (Total = 38)
84
85 Cluster: 2 Prior probability: 0.1953
86
87 Attribute: attribute#1
88 Discrete Estimator. Counts = 7 16 4 (Total = 27)
89 Attribute: attribute#2
90 Discrete Estimator. Counts = 8 4 15 (Total = 27)
91 Attribute: attribute#3
92 Discrete Estimator. Counts = 19 7 (Total = 26)
93 Attribute: attribute#4
94 Discrete Estimator. Counts = 4 19 4 (Total = 27)

```

```

95 Attribute: attribute#5
96 Discrete Estimator. Counts = 13 4 4 7 (Total = 28)
97 Attribute: attribute#6
98 Discrete Estimator. Counts = 19 7 (Total = 26)
99
100 Cluster: 3 Prior probability: 0.1172
101
102 Attribute: attribute#1
103 Discrete Estimator. Counts = 5 7 5 (Total = 17)
104 Attribute: attribute#2
105 Discrete Estimator. Counts = 1 4 12 (Total = 17)
106 Attribute: attribute#3
107 Discrete Estimator. Counts = 12 4 (Total = 16)
108 Attribute: attribute#4
109 Discrete Estimator. Counts = 1 1 15 (Total = 17)
110 Attribute: attribute#5
111 Discrete Estimator. Counts = 6 4 2 6 (Total = 18)
112 Attribute: attribute#6
113 Discrete Estimator. Counts = 1 15 (Total = 16)
114
115
116 Time taken to build model (full training data) : 0.03 seconds
117
118 === Model and evaluation on training set ===
119
120 Clustered Instances
121
122 0          51 ( 41%)
123 1          35 ( 28%)
124 2          23 ( 19%)
125 3          15 ( 12%)
126
127
128 Log likelihood: -6.06035

```

Explanation of the clustering part

It can be seen that three and 4 clusters have been created using the algorithm K-means and the Makedensitybasedclusterer with the clusterer kmeans with 3 and 4 clusters. Still, not a clear cluster result has been reached. This makes sense somehow because I have ignored the class label which classifies data in two clusters. Thus, classifying for 3 or 4 clusters using algorithms does not provide with much information (at least in this case) neither with a good clustering. Also, initial classification is almost exactly the same (it always changed by one in both algorithms) for the different algorithms and clusters. This might mean that the algorithm does not work good in order to be able to change from one state to another. Here below I will try just before starting with the association to check for the kmeans algorithm with two clusters and compare it as well with the class classification.

```

1 === Run information ===
2
3 Scheme:      weka.clusterers.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
4 Relation:    monk1-weka.filters.unsupervised.attribute.AddCluster-Wweka.clusterers.
              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 first-last" -I 500 -num-slots 1 -S 10-I7-
              weka.filters.unsupervised.attribute.Remove-R8
5 Instances:   124
6 Attributes:  7
7              attribute#1
8              attribute#2
9              attribute#3
10             attribute#4
11             attribute#5
12             attribute#6
13 Ignored:
14             class
15 Test mode:   evaluate on training data
16
17
18 === Clustering model (full training set) ===
19
20
21 kMeans
22 =====
23
24 Number of iterations: 3

```

```

25 Within cluster sum of squared errors: 358.0
26
27 Initial starting points (random):
28
29 Cluster 0: 1,2,1,1,1,2
30 Cluster 1: 3,2,2,3,2,1
31
32 Missing values globally replaced with mean/mode
33
34 Final cluster centroids:
35
36 Attribute      Full Data      Cluster#
37                (124)      (77)      (47)
38 =====
39 attribute#1      1          1          3
40 attribute#2      3          2          3
41 attribute#3      1          1          2
42 attribute#4      3          1          3
43 attribute#5      4          4          2
44 attribute#6      2          2          1
45
46
47
48
49 Time taken to build model (full training data) : 0.01 seconds
50
51 === Model and evaluation on training set ===
52
53 Clustered Instances
54
55 0          77 ( 62%)
56 1          47 ( 38%)

```

It still happens the same as before, leading to almost no changes from the initial points and to a non-clear clustering. The clustering algorithm cannot find a good cluster class division because the data is classified according to rules and not according to distances between points. For this reason, trying to predict clusters using distances to create the centroid does not give a good classification, because the classification is not based on distances (discrete data), but on categorical data. Therefore, the algorithms used are not good for our case.

Exercise 2

In this part I am going to perform association analysis to find a set of rules that are able to accurately predict the class label from the rest of the attributes. It is recommended by the lab to use a minimum support of 0.05 and a maximum number of rules of 19, which is the information that is going to be used for that. The apriori algorithm is used and it finds the following results:

```

1 === Run information ===
2
3 Scheme:      weka.associations.Apriori -N 19 -T 0 -C 0.9 -D 0.05 -U 1.0 -M 0.05 -S -1.0 -c -1
4 Relation:    monk1-weka.filters.unsupervised.attribute.AddCluster-Wweka.clusterers.
               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 first-last" -I 500 -num-slots 1 -S 10-I7-
               weka.filters.unsupervised.attribute.Remove-R8
5 Instances:   124
6 Attributes:  7
7               attribute#1
8               attribute#2
9               attribute#3
10              attribute#4
11              attribute#5
12              attribute#6
13              class
14 === Associator model (full training set) ===
15
16
17 Apriori
18 =====
19
20 Minimum support: 0.05 (6 instances)
21 Minimum metric <confidence>: 0.9
22 Number of cycles performed: 19
23
24 Generated sets of large itemsets:
25
26 Size of set of large itemsets L(1): 19
27

```

```

28 Size of set of large itemsets L(2): 151
29
30 Size of set of large itemsets L(3): 378
31
32 Size of set of large itemsets L(4): 125
33
34 Size of set of large itemsets L(5): 6
35
36 Best rules found:
37
38 1. attribute#5=1 29 ==> class=1 29 <conf:(1)> lift:(2) lev:(0.12) [14] conv:(14.5)
39 2. attribute#1=3 attribute#2=3 17 ==> class=1 17 <conf:(1)> lift:(2) lev:(0.07) [8] conv
40 : (8.5)
41 3. attribute#3=1 attribute#5=1 17 ==> class=1 17 <conf:(1)> lift:(2) lev:(0.07) [8] conv
42 : (8.5)
43 4. attribute#5=1 attribute#6=1 16 ==> class=1 16 <conf:(1)> lift:(2) lev:(0.06) [8] conv
44 : (8)
45 5. attribute#1=2 attribute#2=2 15 ==> class=1 15 <conf:(1)> lift:(2) lev:(0.06) [7] conv
46 : (7.5)
47 6. attribute#1=3 attribute#5=1 13 ==> class=1 13 <conf:(1)> lift:(2) lev:(0.05) [6] conv
48 : (6.5)
49 7. attribute#5=1 attribute#6=2 13 ==> class=1 13 <conf:(1)> lift:(2) lev:(0.05) [6] conv
50 : (6.5)
51 8. attribute#2=3 attribute#5=1 12 ==> class=1 12 <conf:(1)> lift:(2) lev:(0.05) [6] conv
52 : (6)
53 9. attribute#3=2 attribute#5=1 12 ==> class=1 12 <conf:(1)> lift:(2) lev:(0.05) [6] conv
54 : (6)
55 10. attribute#1=3 attribute#2=3 attribute#6=2 12 ==> class=1 12 <conf:(1)> lift:(2) lev
56 : (0.05) [6] conv:(6)
57 11. attribute#4=1 attribute#5=1 11 ==> class=1 11 <conf:(1)> lift:(2) lev:(0.04) [5] conv
58 : (5.5)
59 12. attribute#1=2 attribute#5=1 10 ==> class=1 10 <conf:(1)> lift:(2) lev:(0.04) [5] conv
60 : (5)
61 13. attribute#2=2 attribute#5=1 10 ==> class=1 10 <conf:(1)> lift:(2) lev:(0.04) [5] conv
62 : (5)
63 14. attribute#1=1 attribute#2=1 9 ==> class=1 9 <conf:(1)> lift:(2) lev:(0.04) [4] conv
64 : (4.5)
65 15. attribute#4=2 attribute#5=1 9 ==> class=1 9 <conf:(1)> lift:(2) lev:(0.04) [4] conv
66 : (4.5)
67 16. attribute#4=3 attribute#5=1 9 ==> class=1 9 <conf:(1)> lift:(2) lev:(0.04) [4] conv
68 : (4.5)
69 17. attribute#1=2 attribute#2=2 attribute#3=1 9 ==> class=1 9 <conf:(1)> lift:(2) lev:(0.04)
70 [4] conv:(4.5)
71 18. attribute#1=3 attribute#2=3 attribute#3=1 9 ==> class=1 9 <conf:(1)> lift:(2) lev:(0.04)
72 [4] conv:(4.5)
73 19. attribute#3=1 attribute#5=1 attribute#6=1 9 ==> class=1 9 <conf:(1)> lift:(2) lev:(0.04)
74 [4] conv:(4.5)

```

Now I am going to set the variable "car" in the a priori algorithm to true setting as well the index to the class variable to get the rules directly:

```

1 === Run information ===
2
3 Scheme:          weka.associations.Apriori -N 19 -T 0 -C 0.9 -D 0.05 -U 1.0 -M 0.05 -S -1.0 -A -c
4 Relation:        monk1-weka.filters.unsupervised.attribute.AddCluster-Wweka.clusterers.
5                   SimpleKMeans -init 0 -max-candidates 100 -periodic-pruning 10000 -min-density 2.0 -t1 -1.25
6                   -t2 -1.0 -N 3 -A "weka.core.EuclideanDistance -R first-last" -I 500 -num-slots 1 -S 10-17-
7                   weka.filters.unsupervised.attribute.Remove-R8
8 Instances:       124
9 Attributes:      7
10                attribute#1
11                attribute#2
12                attribute#3
13                attribute#4
14                attribute#5
15                attribute#6
16                class
17 === Associator model (full training set) ===
18
19 Apriori
20 =====
21 Minimum support: 0.05 (6 instances)
22 Minimum metric <confidence>: 0.9
23 Number of cycles performed: 19
24
25 Generated sets of large itemsets:
26 Size of set of large itemsets L(1): 33
27
28 Size of set of large itemsets L(2): 141
29

```



```

30 Size of set of large itemsets L(3): 59
31
32 Size of set of large itemsets L(4): 1
33
34 Best rules found:
35
36 1. attribute#5=1 29 ==> class=1 29      conf:(1)
37 2. attribute#1=3 attribute#2=3 17 ==> class=1 17      conf:(1)
38 3. attribute#3=1 attribute#5=1 17 ==> class=1 17      conf:(1)
39 4. attribute#5=1 attribute#6=1 16 ==> class=1 16      conf:(1)
40 5. attribute#1=2 attribute#2=2 15 ==> class=1 15      conf:(1)
41 6. attribute#1=3 attribute#5=1 13 ==> class=1 13      conf:(1)
42 7. attribute#5=1 attribute#6=2 13 ==> class=1 13      conf:(1)
43 8. attribute#2=3 attribute#5=1 12 ==> class=1 12      conf:(1)
44 9. attribute#3=2 attribute#5=1 12 ==> class=1 12      conf:(1)
45 10. attribute#1=3 attribute#2=3 attribute#6=2 12 ==> class=1 12      conf:(1)
46 11. attribute#4=1 attribute#5=1 11 ==> class=1 11      conf:(1)
47 12. attribute#1=2 attribute#5=1 10 ==> class=1 10      conf:(1)
48 13. attribute#2=2 attribute#5=1 10 ==> class=1 10      conf:(1)
49 14. attribute#1=1 attribute#2=1 9 ==> class=1 9      conf:(1)
50 15. attribute#4=2 attribute#5=1 9 ==> class=1 9      conf:(1)
51 16. attribute#4=3 attribute#5=1 9 ==> class=1 9      conf:(1)
52 17. attribute#1=2 attribute#2=2 attribute#3=1 9 ==> class=1 9      conf:(1)
53 18. attribute#1=3 attribute#2=3 attribute#3=1 9 ==> class=1 9      conf:(1)
54 19. attribute#3=1 attribute#5=1 attribute#6=1 9 ==> class=1 9      conf:(1)

```

Eliminating association rules

Giving the fact that the rules in our case are monotonous, e.g if the set A is true, for any superset of A called B, then the rule for A still holds, we can eliminate from the first rule all other supersets getting the following 4 set of rules:

```

1 1. attribute#5=1 29 ==> class=1 29      conf:(1)
2 2. attribute#1=3 attribute#2=3 17 ==> class=1 17      conf:(1)
3 5. attribute#1=2 attribute#2=2 15 ==> class=1 15      conf:(1)
4 14. attribute#1=1 attribute#2=1 9 ==> class=1 9      conf:(1)

```

Explanation of classification

This tell us that (1) whenever attribute5 is classified as 1 or (2) attribute1 and attribute2 are classified as 3 or (3) attribute1 and attribute2 are classified as 2 or (4) attribute1 and attribute2 are classified as 1, the classification for the data will be 1. I can simply it even more to 2 rules: (1) whenever attribute5 is classified as 1 or (2) whenever attribute1 and attribute2 are classified equally the the same number.

Conclusion

As previously said, trying to predict clusters using distances to create the centroid does not give a good classification, because the classification of the data is not based on distances (discrete data), but on categorical data. Therefore, the algorithms used are not good for our case, unless this data is treated somehow such that the algorithm does not look for the centroid in the way the kmeans or the makedensitybasedclustered is used but taking into account how cluster from categorical data. For this reason, I will say that the clustering algorithms fail or perform poorly for the monk1 dataset.