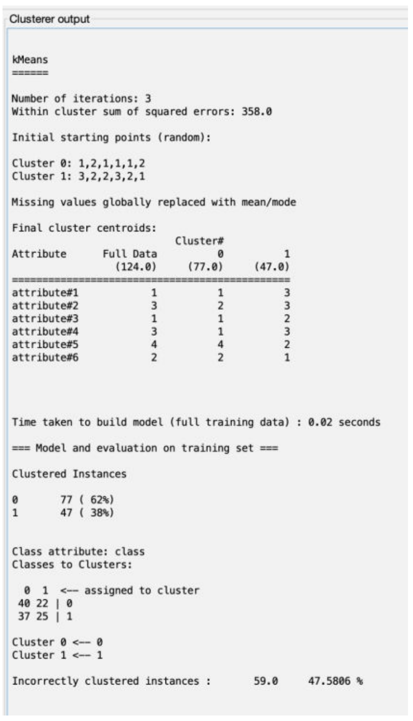
Clustering Attempts

SimpleKMeans - two clusters

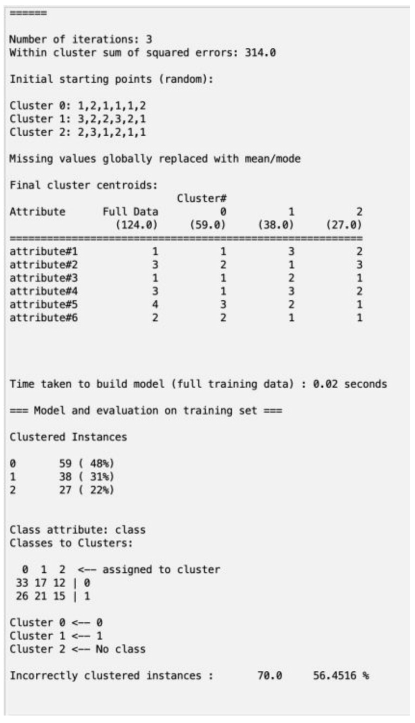
We started using the SimpleKMeans algorithm with two clusters as we know there are two classes in the data. A little bit less than 50% of the data was misclassified using this setup, which is hardly better than taking a random guess.



```
kMeans
======

Number of iterations: 3
Within cluster sum of squared errors: 358.0

Initial starting points (random):

Cluster 0: 1,2,1,1,1,2
Cluster 1: 3,2,2,3,2,1

Missing values globally replaced with mean/mode

Final cluster centroids:
                         Cluster#
Attribute      Full Data        0          1
                 (124.0)   (77.0)     (47.0)
=============================================
attribute#1            1        1          3
attribute#2            3        2          3
attribute#3            1        1          2
attribute#4            3        1          3
attribute#5            4        4          2
attribute#6            2        2          1




Time taken to build model (full training data) : 0.02 seconds

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

Clustered Instances

0       77 ( 62%)
1       47 ( 38%)


Class attribute: class
Classes to Clusters:

  0  1  <-- assigned to cluster
 40 22 | 0
 37 25 | 1

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

Incorrectly clustered instances :       59.0     47.5806 %
```

SimpleKMeans - three clusters

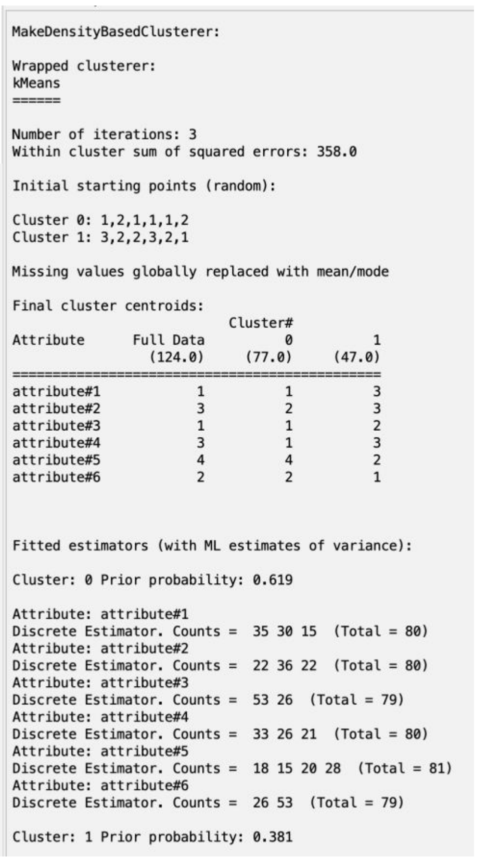
Increasing the number of clusters by one results in even worse results, incorrectly classifying over 50% of the data. It makes sense that having more clusters than classes will give poor results when comparing to the true classes, since we know that there are only supposed to be two.



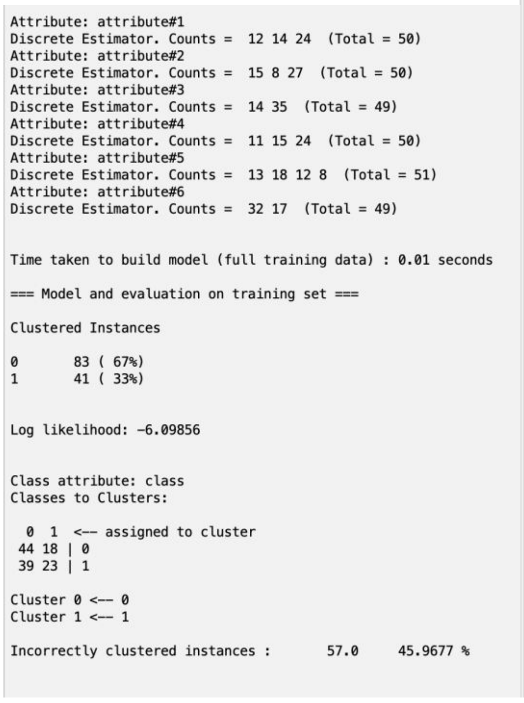
```
======

Number of iterations: 3
Within cluster sum of squared errors: 314.0

Initial starting points (random):

Cluster 0: 1,2,1,1,1,2
Cluster 1: 3,2,2,3,2,1
Cluster 2: 2,3,1,2,1,1

Missing values globally replaced with mean/mode

Final cluster centroids:
                         Cluster#
Attribute      Full Data        0          1          2
                 (124.0)   (59.0)     (38.0)     (27.0)
========================================================
attribute#1            1        1          3          2
attribute#2            3        2          1          3
attribute#3            1        1          2          1
attribute#4            3        1          3          2
attribute#5            4        3          2          1
attribute#6            2        2          1          1




Time taken to build model (full training data) : 0.02 seconds

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

Clustered Instances

0       59 ( 48%)
1       38 ( 31%)
2       27 ( 22%)


Class attribute: class
Classes to Clusters:

  0  1  2  <-- assigned to cluster
 33 17 12 | 0
 26 21 15 | 1

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

Incorrectly clustered instances :       70.0     56.4516 %
```

Density-Based - two clusters

The density-based algorithm also produces poor results when using two clusters, only being slightly better than a random guess.



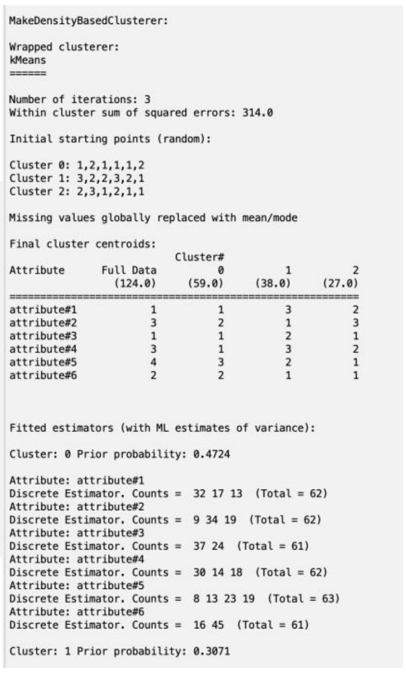
```
MakeDensityBasedClusterer:

Wrapped clusterer:
kMeans
======

Number of iterations: 3
Within cluster sum of squared errors: 358.0

Initial starting points (random):

Cluster 0: 1,2,1,1,1,2
Cluster 1: 3,2,2,3,2,1

Missing values globally replaced with mean/mode

Final cluster centroids:
                           Cluster#
Attribute      Full Data          0          1
                 (124.0)     (77.0)     (47.0)
==============================================
attribute#1            1          1          3
attribute#2            3          2          3
attribute#3            1          1          2
attribute#4            3          1          3
attribute#5            4          4          2
attribute#6            2          2          1



Fitted estimators (with ML estimates of variance):

Cluster: 0 Prior probability: 0.619

Attribute: attribute#1
Discrete Estimator. Counts =  35 30 15  (Total = 80)
Attribute: attribute#2
Discrete Estimator. Counts =  22 36 22  (Total = 80)
Attribute: attribute#3
Discrete Estimator. Counts =  53 26  (Total = 79)
Attribute: attribute#4
Discrete Estimator. Counts =  33 26 21  (Total = 80)
Attribute: attribute#5
Discrete Estimator. Counts =  18 15 20 28  (Total = 81)
Attribute: attribute#6
Discrete Estimator. Counts =  26 53  (Total = 79)

Cluster: 1 Prior probability: 0.381
```



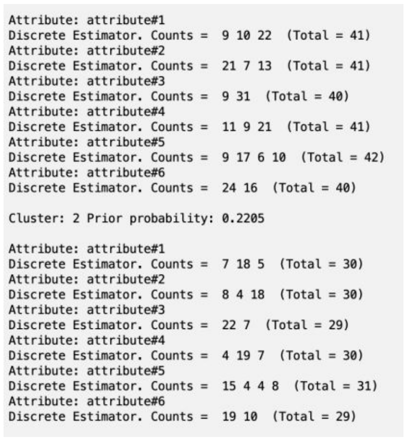
```
Attribute: attribute#1
Discrete Estimator. Counts =  12 14 24  (Total = 50)
Attribute: attribute#2
Discrete Estimator. Counts =  15 8 27  (Total = 50)
Attribute: attribute#3
Discrete Estimator. Counts =  14 35  (Total = 49)
Attribute: attribute#4
Discrete Estimator. Counts =  11 15 24  (Total = 50)
Attribute: attribute#5
Discrete Estimator. Counts =  13 18 12 8  (Total = 51)
Attribute: attribute#6
Discrete Estimator. Counts =  32 17  (Total = 49)


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

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

Clustered Instances

0       83 ( 67%)
1       41 ( 33%)


Log likelihood: -6.09856


Class attribute: class
Classes to Clusters:

  0  1  <-- assigned to cluster
 44 18 | 0
 39 23 | 1

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

Incorrectly clustered instances :       57.0     45.9677 %
```

Density-Based - three clusters

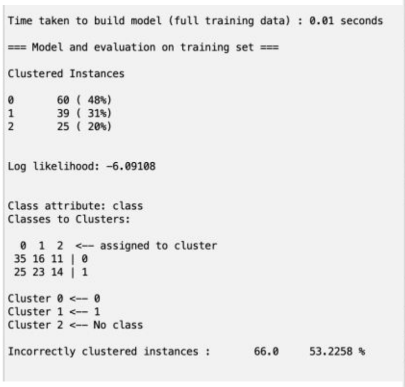
Again, increasing the number of clusters for the density-based algorithm further decreases the performance of the clustering.



```
MakeDensityBasedClusterer:

Wrapped clusterer:
kMeans
======

Number of iterations: 3
Within cluster sum of squared errors: 314.0

Initial starting points (random):

Cluster 0: 1,2,1,1,1,2
Cluster 1: 3,2,2,3,2,1
Cluster 2: 2,3,1,2,1,1

Missing values globally replaced with mean/mode

Final cluster centroids:
                           Cluster#
Attribute      Full Data          0          1          2
                 (124.0)     (59.0)     (38.0)     (27.0)
=========================================================
attribute#1            1          1          3          2
attribute#2            3          2          1          3
attribute#3            1          1          2          1
attribute#4            3          1          3          2
attribute#5            4          3          2          1
attribute#6            2          2          1          1



Fitted estimators (with ML estimates of variance):

Cluster: 0 Prior probability: 0.4724

Attribute: attribute#1
Discrete Estimator. Counts =  32 17 13  (Total = 62)
Attribute: attribute#2
Discrete Estimator. Counts =  9 34 19  (Total = 62)
Attribute: attribute#3
Discrete Estimator. Counts =  37 24  (Total = 61)
Attribute: attribute#4
Discrete Estimator. Counts =  30 14 18  (Total = 62)
Attribute: attribute#5
Discrete Estimator. Counts =  8 13 23 19  (Total = 63)
Attribute: attribute#6
Discrete Estimator. Counts =  16 45  (Total = 61)

Cluster: 1 Prior probability: 0.3071
```



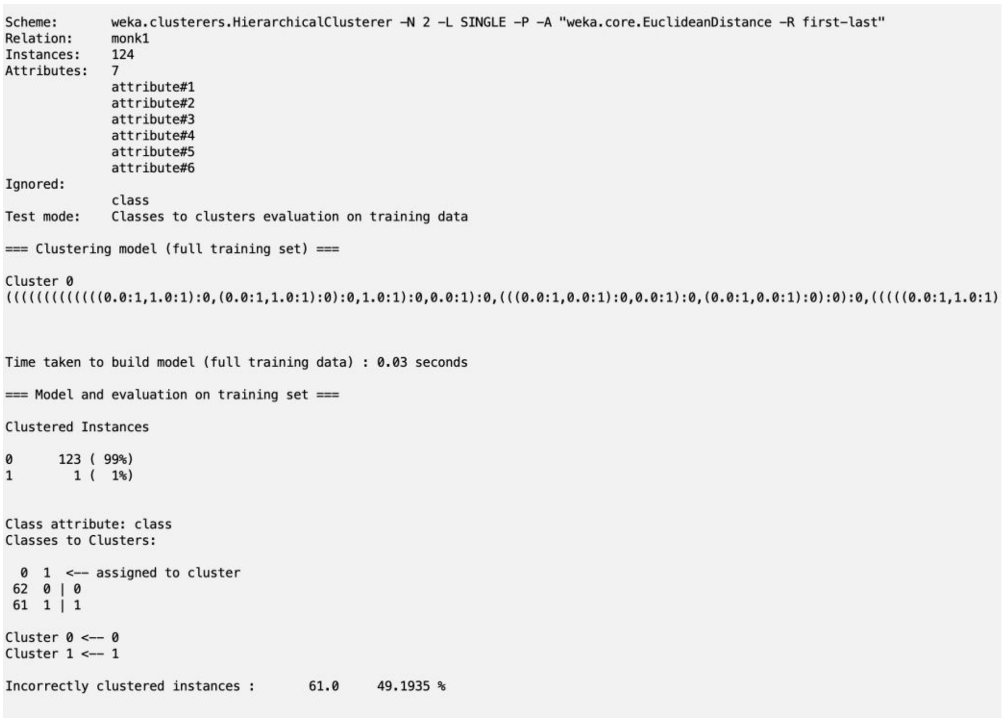
```
Attribute: attribute#1
Discrete Estimator. Counts =  9 10 22  (Total = 41)
Attribute: attribute#2
Discrete Estimator. Counts =  21 7 13  (Total = 41)
Attribute: attribute#3
Discrete Estimator. Counts =  9 31  (Total = 40)
Attribute: attribute#4
Discrete Estimator. Counts =  11 9 21  (Total = 41)
Attribute: attribute#5
Discrete Estimator. Counts =  9 17 6 10  (Total = 42)
Attribute: attribute#6
Discrete Estimator. Counts =  24 16  (Total = 40)

Cluster: 2 Prior probability: 0.2205

Attribute: attribute#1
Discrete Estimator. Counts =  7 18 5  (Total = 30)
Attribute: attribute#2
Discrete Estimator. Counts =  8 4 18  (Total = 30)
Attribute: attribute#3
Discrete Estimator. Counts =  22 7  (Total = 29)
Attribute: attribute#4
Discrete Estimator. Counts =  4 19 7  (Total = 30)
Attribute: attribute#5
Discrete Estimator. Counts =  15 4 4 8  (Total = 31)
Attribute: attribute#6
Discrete Estimator. Counts =  19 10  (Total = 29)
```



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

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

Clustered Instances

0       60 ( 48%)
1       39 ( 31%)
2       25 ( 20%)


Log likelihood: -6.09108


Class attribute: class
Classes to Clusters:

  0  1  2  <-- assigned to cluster
 35 16 11 | 0
 25 23 14 | 1

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

Incorrectly clustered instances :       66.0     53.2258 %
```

Hierarchical Clustering - two clusters

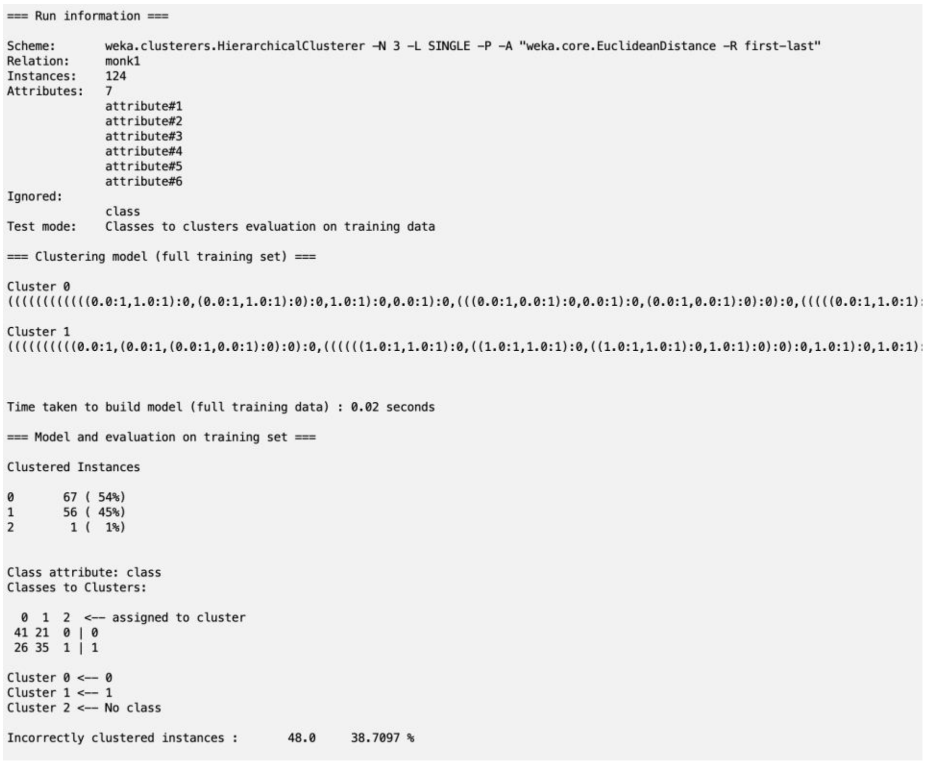
Using two clusters on the hierarchical clustering algorithm gives the same accuracy as a random guess.



```
Scheme:       weka.clusterers.HierarchicalClusterer -N 2 -L SINGLE -P -A "weka.core.EuclideanDistance -R first-last"
Relation:     monk1
Instances:    124
Attributes:   7
              attribute#1
              attribute#2
              attribute#3
              attribute#4
              attribute#5
              attribute#6
Ignored:
              class
Test mode:    Classes to clusters evaluation on training data

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

Cluster 0
(((((((((((((0.0:1,1.0:1):0,(0.0:1,1.0:1):0):0,1.0:1):0,0.0:1):0,(((0.0:1,0.0:1):0,0.0:1):0,(0.0:1,0.0:1):0):0):0,(((((0.0:1,1.0:1)



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

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

Clustered Instances

0      123 ( 99%)
1        1 (  1%)


Class attribute: class
Classes to Clusters:

  0  1  <-- assigned to cluster
 62  0 | 0
 61  1 | 1

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

Incorrectly clustered instances :       61.0     49.1935 %
```

Hierarchical Clustering - three clusters

Relatively, this performs surprisingly well with only 38.7% of the data assigned to the wrong cluster. Still, it’s not very good.



```
=== Run information ===

Scheme:       weka.clusterers.HierarchicalClusterer -N 3 -L SINGLE -P -A "weka.core.EuclideanDistance -R first-last"
Relation:     monk1
Instances:    124
Attributes:   7
              attribute#1
              attribute#2
              attribute#3
              attribute#4
              attribute#5
              attribute#6
Ignored:
              class
Test mode:    Classes to clusters evaluation on training data

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

Cluster 0
((((((((((((0.0:1,1.0:1):0,(0.0:1,1.0:1):0):0,1.0:1):0,0.0:1):0,(((0.0:1,0.0:1):0,0.0:1):0,(0.0:1,0.0:1):0):0):0,(((((0.0:1,1.0:1):

Cluster 1
((((((((((0.0:1,(0.0:1,(0.0:1,0.0:1):0):0):0,((((((1.0:1,1.0:1):0,((1.0:1,1.0:1):0,((1.0:1,1.0:1):0,1.0:1):0):0):0,1.0:1):0,1.0:1):


Time taken to build model (full training data) : 0.02 seconds

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

Clustered Instances

0       67 ( 54%)
1       56 ( 45%)
2        1 (  1%)


Class attribute: class
Classes to Clusters:

  0  1  2  <-- assigned to cluster
 41 21  0 | 0
 26 35  1 | 1

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

Incorrectly clustered instances :       48.0     38.7097 %
```

Association

We wrote a small Python script to evaluate our rules, avoiding redundancy to find some different options, and wound up with the four following rules as the best:

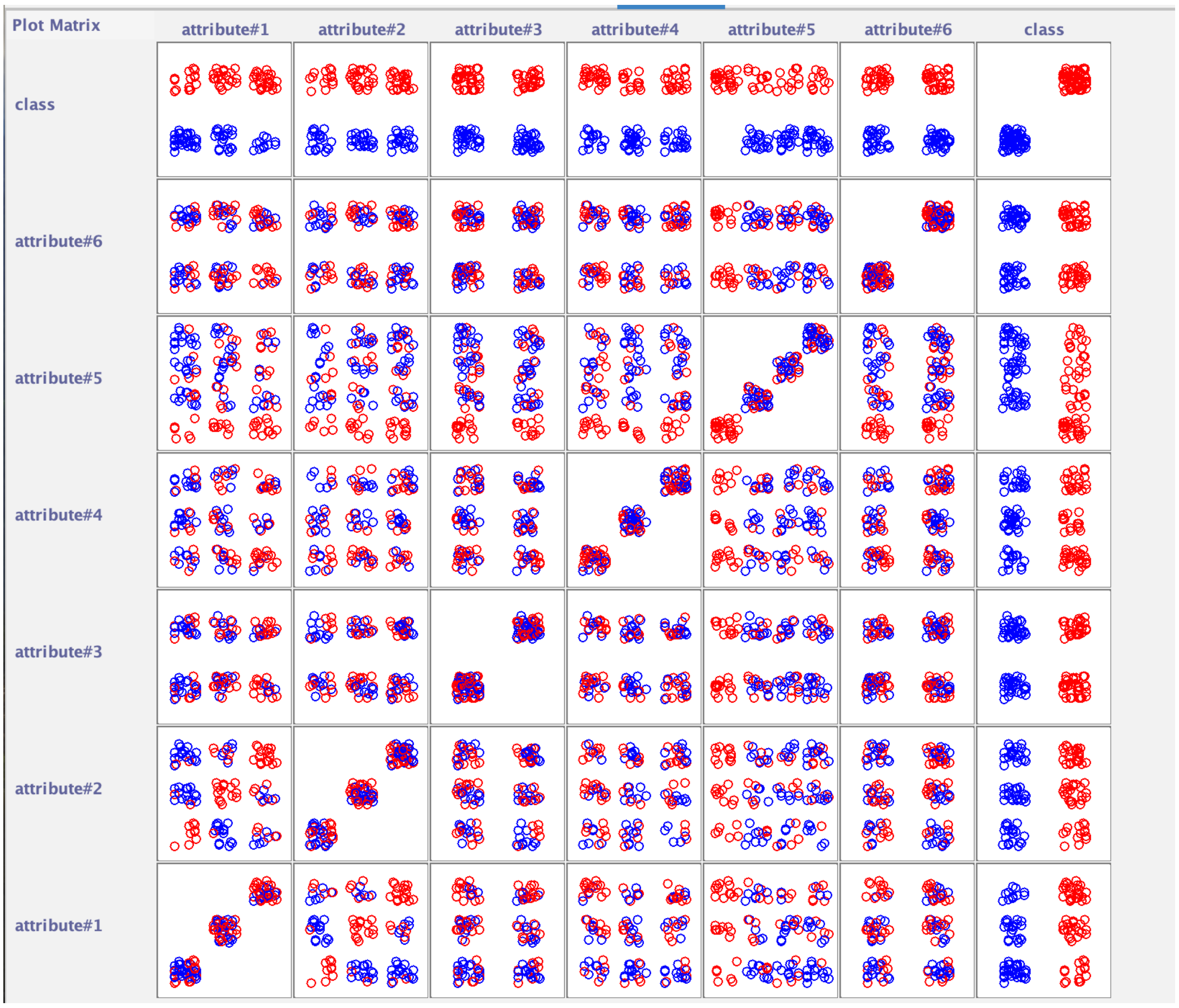
1. Attribute 5 = 1 → class = 1
2. Attribute 3 = 3 and attribute 2 = 3 → class = 1
3. Attribute 1 = 2 and attribute 2 = 2 → class = 1
4. Attribute 1 = 3 and attribute 2 = 3 and attribute 6 = 2 → class = 1

The Python script:

```python
import pandas as pd
from scipy.io import arff

data = arff.loadarff(r"C:\Users\marij\Desktop\monk1.arff")[0]
df = pd.DataFrame(data)
df = df.map(lambda x: int(x))

df.loc[df["attribute#5"] == 1, "assoc"] = 1
df.loc[(df["attribute#3"] == 3) & (df["attribute#2"] == 3), "assoc"] = 1
df.loc[(df["attribute#1"] == 2) & (df["attribute#2"] == 2), "assoc"] = 1
df.loc[(df["attribute#1"] == 3) & (df["attribute#2"] == 3) &
(df["attribute#6"] == 2), "assoc"] = 1
df.loc[df["assoc"].isna(), "assoc"] = 0
```

```python
# df.loc[df["attribute#5"] == 1, "assoc"] = 1
# df.loc[(df["attribute#3"] == 3) & (df["attribute#2"] == 3), "assoc"] = 1
# df.loc[(df["attribute#1"] == 2) & (df["attribute#2"] == 2), "assoc"] = 1
# df.loc[(df["attribute#1"] == 3) & (df["attribute#2"] == 3) &
(df["attribute#3"] == 1), "assoc"] == 1
# df.loc[df["assoc"].isna(), "assoc"] = 0

print(f"{len(df.loc[df["class"] != df["assoc"]].index)} misclassified")
```

Visualization



The Question

Why can the clustering algorithms not find a clustering that matches the class division in the database?

As can be seen from the visual, the data is mostly very mixed between all the attributes. The data is discrete, and there are no clear clusters that show up across all attributes. Association allows us to look at a subset of the attributes at a time and draw conclusions from this. For example, when attribute five has value one this leads to class one 100% of the time. When doing clustering, the clusters have to have centroids in all the attributes. This is perhaps also why the hierarchical clustering algorithm performed slightly better, as it does not rely on these centroids.

Would you say that the clustering algorithms fail or perform poorly for the monk1 dataset?

The clustering algorithms perform very poorly on this dataset, often misassigning around 50% of the data points. With accuracies like these on a binary-classed dataset, we may as well randomly assign classes. In this case, association would be a better choice for the reasons explained above.