TDDD41 - Lab 2

Martin Estgren, Alice Reinaudo

March 3, 2017

1 Data and binning

For this assignment, we use the iris dataset, which contains information about sizes of sepals and petals for three species of iris. With 3 bins, looking at the visualizations for the attribute values, it appears that the sepal width and length have moderate to low impact, while the petal width and length have high impact on the actual type of flower. This seems to occur because values for the sepal dimensions tend to overlap more between different species of iris while the petal dimensions do not. Similar information can be seen regardless of the number of bins, although the less bins we have the more information is lost.

Since we discretize and choose the number of bins, the more small bins we have the less often each of the corresponding discretized values will appear, and the smaller the support that each attribute-value set has on average. Vice versa, if the bins are few and wide then each value can repeat more often, and so the support will be on average higher. We keep this into account especially with regards to the minimum support.

2 Parameters

The following are the parameters we used for the functions, unless otherwise specified e.g. in the experiments:

- Minimum support = 0.1
- Metric type: confidence
- Minimum confidence = 0.9

3 k-Means

The results with k-means are quite good with 3 bins and 3 clusters. From the confusion matrix we can see that all the iris setosa items are classified correctly, while versicolor and virginica sometimes get swapped. If we increase the number of bins, we could expect our results to improve or at worst stay unchanged, since we would have a more precise measure of the distances of the data points.

```
Class attribute: class
Classes to Clusters:

  0   1   2  <-- assigned to cluster
  0   0  50 | Iris-setosa
 48   2   0 | Iris-versicolor
  7  43   0 | Iris-virginica

Cluster 0 <-- Iris-versicolor
Cluster 1 <-- Iris-virginica
Cluster 2 <-- Iris-setosa

Incorrectly clustered instances :        9.0        6        %
```

4 Association analysis

The frequent itemsets in this case correspond to the frequent attribute-value sets in our database, and a transaction corresponds to a single data point.

5 Clustering through association analysis

By using the default settings for the A priori algorithm, we do not get enough rules to cover all clusters. Instead we used $numRules = 1000$, $metricType = confidence$ and $minMetric = 0.9$. In this way we obtain 1000 rules and we keep what has more than 90% confidence. The following are the interesting rules that we got.

```
6. petallength='(-inf-2.966667]' 50
                                    ==> cluster=cluster3
   50          <conf:(1)>
10. petalwidth='(-inf-0.9]' 50
                                          ==> cluster=
   cluster3 50          <conf:(1)>
51. petallength='(2.966667-4.933333]' petalwidth
   ='(0.9-1.7]' 48 ==> cluster=cluster1 48          <conf
   :(1)>
277. sepallength='(5.5-6.7]' petallength
   ='(2.966667-4.933333]' petalwidth='(0.9-1.7]' 33 ==>
   cluster=cluster1 33    <conf:(1)>
106. petallength='(4.933333-inf)' petalwidth='(1.7-inf)'
   40     ==> cluster=cluster2 40     <conf:(1)>
284. sepalwidth='(2.8-3.6]' petalwidth='(1.7-inf)' 29
                  ==> cluster=cluster2 29          <conf
   :(1)>
```

The reason that we do not want to have the class attribute in the antecedent is that it is what we want to predict to begin with, so when we have a new flower to consider we do not have its class at our disposal. Moreover we only want the cluster attribute in the consequent, because the other attributes we want to use to predict, not to be predicted.

6 Experiments

6.1 Different number of bins

Increasing the number of bins can help take more information into account and calculating distances more precisely in k-means, and obtain more rules in association analysis. On the other hand, too many clusters could provide little extra information and possibly confuse the algorithms.

We tried using 6 bins. For k-means, we actually obtain a perfect result with no misclassified elements, which is consistent with our assumption in the previous paragraph.

We obtain the following rules.

```
3.        petallength='(-inf-1.983333]' 50
                                          ==> cluster=
    cluster1 50     <conf:(1)>
12. petalwidth='(-inf-0.5]' 49
                                          ==> cluster=
    cluster1 49     <conf:(1)>
61. sepallength='(5.5-6.1]' petallength='(3.95-4.933333]'
     21     ==> cluster=cluster2 21     <conf:(1)>
132.petallength='(3.95-4.933333]' petalwidth='(0.9-1.3]'
    18      ==> cluster=cluster2 18     <conf:(1)>
75. sepallength='(6.1-6.7]' petallength
    ='(4.933333-5.916667]' 20==> cluster=cluster3 20     <
    conf:(1)>
127.sepalwidth='(2.8-3.2]' petallength
    ='(4.933333-5.916667]' 18 ==> cluster=cluster3 18     <
    conf:(1)>
```

From these we can see that the first cluster is mostly characterized by the dimensions of the petals, while the second and third clusters are characterized by a mix of attribute values. The clustering however is not so good, because 31.3% of the flowers are in the incorrect cluster. It could be that, because we have a higher "resolution" in the possible values that the attributes can take, some sets of attribute values do not reach minimum support and don't give rise to the correct rules. This highlights that it may be important not to overdo the binning process but instead we try to make the discretization in a more significant way, for example by domain knowledge.

6.2 Different number of clusters

We do not necessarily need to have as many clusters as classes, but if we want to have a reasonable chance of correctly predicting class labels from clusters, we need at least as many clusters as classes. If we have more clusters, it is then not a problem to have more than one cluster map to the same class. These may for instance correspond to further subdivisions of flower types.

We experimented with 6 clusters, twice as many as the classes. What happened is that rules satisfying our previously stated constraints were only found for 3 of the clusters: 1, 2 and 5, whereas we do not find any rules for 3, 4 and 6, regardless of the minimum confidence we set. The latter clusters also contained points, but just a few so that the minimum confidence was never satisfied, because most other points belonging to the same classes were in the larger clusters. From this we can conclude that the classes are well separable and that 3 clusters are good enough to obtain an accurate classification. The following rules we found cover each of the clusters.

```
690. sepalwidth='(2.8-3.6]' petallength='(4.933333-inf)'
     28              ==> cluster=cluster2 26     <conf
    :(0.93)>
709. sepalwidth='(2.8-3.6]' petallength
    ='(2.966667-4.933333]' 24        ==> cluster=cluster1
    22     <conf:(0.92)>
726. sepalwidth='(2.8-3.6]' petallength
    ='(2.966667-4.933333]' petalwidth='(0.9-1.7]' 21 ==>
    cluster=cluster1 19     <conf:(0.9)>
```

Predictably, if we change the minimum support to 0.01, more of the clusters get rules, with 1, 2, 3, 5, 6 obtaining high confidence rules, as follows.

```
48.  sepallength='(5.5-6.7]' petallength
    ='(2.966667-4.933333]' 39 ==> cluster=cluster1 39
         <conf:(1)>
49.  sepallength='(5.5-6.7]' petalwidth='(0.9-1.7]' 38
                              ==> cluster=cluster1 38   <
    conf:(1)>
31.  petallength='(4.933333-inf)' petalwidth='(1.7-inf)'
     40       ==> cluster=cluster2 40   <conf:(1)>
551. sepalwidth='(3.6-inf)' petallength='(-inf
    -2.966667]' 13      ==> cluster=cluster3 13   <conf
    :(1)>
77.  sepalwidth='(2.8-3.6]' petallength='(-inf
    -2.966667]' 36      ==> cluster=cluster5 36   <conf
    :(1)>
83.  sepalwidth='(2.8-3.6]' petalwidth='(-inf-0.9]' 36
                              ==> cluster=cluster5 36   <
    conf:(1)>
```

656. sepallength='(-inf-5.5]' sepalwidth='(-inf-2.8]' petallength='(2.966667-4.933333]' 11 ==> cluster=cluster6 11 <conf:(1)>