Data Mining Assignment 4:

Due on Monday, April 9, 2018

Dr. Predrag Radivojac

Arnav Arnav (aarnav)

February 3, 2018

Contents

Problem 1

The main problem faced here is to get figure out the cluster memberships.
Assuming that the clusters are globular, to find cluster memberships we can find the pairwise distance between centroids. Next, we can find out all the points that are further away than twice the maximum pairwise distance between centroids. We can be certain that these points must be in different clusters. We can say that the points that are closer than half the minimum pairwise distance between centroids can be in the same cluster. With this information, we can get the cluster memberships.
Once the cluster memberships are known, we can find the silhouette coefficient for all the points and find the weighted average of the silhouette coefficients of all the clusters to get the silhouette coefficient of the entire clustering. Then, two clusters can be merged that maximize the overall silhouette coefficient of the clustering.
Another approach can be to use Ward's method and merge the clusters that minimize the sum of squared errors in the clustering

Problem 2

a. The code can be found in the file `code/q2/apriori.py`

   The code contains two classes Apriori1, which contains the $F_{k-1} \times F_{k-1}$ implementation of the apriori algorithm and the class Apriori2 which uses $F_{k-1} \times F_1$ implementation to generate candidates.

b. The three datasets used for the comparison are

   - UCI cars dataset available at `https://archive.ics.uci.edu/ml/datasets/car+evaluation`
   - UCI mushroom dataset available at`https://archive.ics.uci.edu/ml/datasets/mushroom`
   - UCI nursery dataset available at `https://archive.ics.uci.edu/ml/datasets/nursery`

   Table 1 contains the results of the comparison for different values of minimum support implementation

Table 1: Number of candidates generated upto level 4

Dataset	**Minimum support**	$F_{K-1} \times F_{k-1}$	$F_{k-1} \times F_1$
$Cars$	0.05	1669	552
$Cars$	0.1	375	2477
$Cars$	0.2	278	1133
$Nursery$	0.05	3647	13439
$Nuresry$	0.1	958	6244
$Nursery$	0.2	332	1053
$Mushroom$	0.05	100951	61021
$Mushrom$	0.1	34481	15641
$Mushroom$	0.2	8564	9017

c. The table 2 enumerates the number of closed frequent and maximal frequent item-sets

d. The number of rules generated are listed in table 3

e. the top 5 rules generated for different combination of support and confidence are listed below

   - For min_sup = 0.05 and min_conf = 0.5 and cars dataset the rules generated are: (rules follow the pattern: variable_value and ...) here 5 is the safety variable, 3 is the number of persons, 4 represents the luggage boot space, 6 is the acceptability

Problem 2 continued on next page...

Table 2: Frequent closed, Maximal frequent and frequent item sets for $F_{k-1} \times F_{k-1}$

Dataset	Minimum support	Frequent itemsets	Closed Frequent itemsets	Maximal Frequent itemse
Cars	0.05	349	181	3
Cars	0.1	846	40	0
Cars	0.2	31	20	3
Nursery	0.05	623	442	24
Nuresry	0.1	169	119	8
Nursery	0.2	26	23	19
Mushroom	0.05	60246	2945	0
Mushrom	0.1	21562	1299	0
Mushroom	0.2	5450	402	0

Table 3: Frequent closed, Maximal frequent and frequent item sets for $F_{k-1} \times F_{k-1}$

Dataset	Minimum support	Minimum confidence	number of rules
Cars	0.05	0.5	140
Cars	0.0.5	0.6	112
Cars	0.0.05	0.7	83
Cars	0.1	0.5	39
Cars	0.1	0.6	30
Cars	0.1	0.7	19
Cars	0.2	0.5	8
Cars	0.2	0.6	8
Cars	0.2	0.7	5
Nursery	0.05	0.5	445
Nuresry	0.05	0.6	117
Nursery	0.05	0.7	96
Nursery	0.1	0.5	85
Nuresry	0.1	0.6	26
Nursery	0.1	0.7	24
Nursery	0.2	0.5	2
Nuresry	0.2	0.6	2
Nursery	0.2	0.7	2
Mushroom	0.05	0.5	8
Mushroom	0.05	0.6	8
Mushroom	0.05	0.7	5
Mushroom	0.05	0.5	8
Mushroom	0.05	0.6	8
Mushroom	0.05	0.7	5
Mushroom	0.05	0.5	8
Mushroom	0.05	0.6	8
Mushroom	0.05	0.7	5

```
if 3_2 and 4_big ===> 6_unacc  confidence = 1.0
```

```
if 3_2 and 4_med ===> 6_unacc  confidence = 1.0

if 3_2 and 5_high ===> 6_unacc  confidence = 1.0

if 0_vhigh and 3_2 ===> 6_unacc  confidence = 1.0

if 1_med and 3_2 ===> 6_unacc  confidence = 1.0
```

- For min_sup = 0.05 and min_conf = 0.6 and cars dataset the rules generated are

```
if 0_low and 3_2 ===> 6_unacc  confidence = 1.0

if 2_3 and 5_low ===> 6_unacc  confidence = 1.0

if 0_high and 5_low ===> 6_unacc  confidence = 1.0

if 1_high and 3_2 ===> 6_unacc  confidence = 1.0

if 2_2 and 5_low ===> 6_unacc  confidence = 1.0
```

- For min_sup = 0.05 and min_conf = 0.7 and cars dataset the rules generated are

```
if 2_2 and 3_2 ===> 6_unacc  confidence = 1.0

if 0_high and 3_2 ===> 6_unacc  confidence = 1.0

if 0_vhigh and 1_vhigh ===> 6_unacc  confidence = 1.0

if 2_3 and 3_2 ===> 6_unacc  confidence = 1.0

if 1_med and 3_2 ===> 6_unacc  confidence = 1.0
```

- For min_sup = 0.1 and min_conf = 0.5 and cars dataset the rules generated are

```
if 3_2 and 4_small ===> 6_unacc  confidence = 1.0

if 3_2 and 4_med ===> 6_unacc  confidence = 1.0

if 3_2 and 4_big ===> 6_unacc  confidence = 1.0

if 5_low ===> 6_unacc  confidence = 1.0

if 4_small and 5_low ===> 6_unacc  confidence = 1.0
```

- For min_sup = 0.1 and min_conf = 0.6 and cars dataset the rules generated are

```
if 3_4 and 5_low ===> 6_unacc  confidence = 1.0
```

```
if 3_more and 5_low ===> 6_unacc  confidence = 1.0

if 4_big and 5_low ===> 6_unacc  confidence = 1.0

if 4_med and 5_low ===> 6_unacc  confidence = 1.0

if 4_small and 5_low ===> 6_unacc  confidence = 1.0
```

- For min_sup = 0.1 and min_conf = 0.7 and cars dataset the rules generated are

```
if 3_2 and 4_big ===> 6_unacc  confidence = 1.0

if 5_low ===> 6_unacc  confidence = 1.0

if 3_2 ===> 6_unacc  confidence = 1.0

if 3_2 and 5_high ===> 6_unacc  confidence = 1.0

if 4_small and 5_low ===> 6_unacc  confidence = 1.0
```

- For min_sup = 0.2 and min_conf = 0.5 and cars dataset the rules generated are

```
if 4_small ===> 6_unacc  confidence = 0.78125

if 0_vhigh ===> 6_unacc  confidence = 0.833333333333

if 1_vhigh ===> 6_unacc  confidence = 0.833333333333

if 3_2 ===> 6_unacc  confidence = 1.0

if 5_low ===> 6_unacc  confidence = 1.0
```

- For min_sup = 0.2 and min_conf = 0.6 and cars dataset the rules generated are

```
if 4_small ===> 6_unacc  confidence = 0.78125

if 0_vhigh ===> 6_unacc  confidence = 0.833333333333

if 1_vhigh ===> 6_unacc  confidence = 0.833333333333

if 3_2 ===> 6_unacc  confidence = 1.0

if 5_low ===> 6_unacc  confidence = 1.0
```

- For min_sup = 0.2 and min_conf = 0.7 and cars dataset the rules generated are

```
if 4_small ===> 6_unacc  confidence = 0.78125
```

```
if 0_vhigh ===> 6_unacc  confidence = 0.833333333333

if 1_vhigh ===> 6_unacc  confidence = 0.833333333333

if 3_2 ===> 6_unacc  confidence = 1.0

if 5_low ===> 6_unacc  confidence = 1.0
```

- For min_sup = 0.05 and min_conf = 0.5 and nursery dataset the rules generated are

```
if 0_pretentious and 5_convenient and 8_not_recom ===> 7_not_recom  confidence = 1.0

if 3_3 and 7_not_recom ===> 8_not_recom  confidence = 1.0

if 0_great_pret and 5_inconv and 7_not_recom ===> 8_not_recom  confidence = 1.0

if 2_complete and 8_not_recom ===> 7_not_recom  confidence = 1.0

if 5_inconv and 6_slightly_prob and 7_not_recom ===> 8_not_recom  confidence = 1.0
```

- For min_sup = 0.05 and min_conf = 0.6 and nursery dataset the rules generated are

```
if 3_1 and 8_not_recom ===> 7_not_recom  confidence = 1.0

if 2_incomplete and 7_not_recom ===> 8_not_recom  confidence = 1.0

if 2_incomplete and 8_not_recom ===> 7_not_recom  confidence = 1.0

if 2_foster and 8_not_recom ===> 7_not_recom  confidence = 1.0

if 5_inconv and 6_slightly_prob and 7_not_recom ===> 8_not_recom  confidence = 1.0
```

- For min_sup = 0.05 and min_conf = 0.7 and nursery dataset the rules generated are

```
if 2_incomplete and 7_not_recom ===> 8_not_recom  confidence = 1.0

if 2_incomplete and 8_not_recom ===> 7_not_recom  confidence = 1.0

if 2_foster and 7_not_recom ===> 8_not_recom  confidence = 1.0

if 3_3 and 7_not_recom ===> 8_not_recom  confidence = 1.0

if 5_inconv and 6_slightly_prob and 7_not_recom ===> 8_not_recom  confidence = 1.0
```

- For min_sup = 0.1 and min_conf = 0.5 and nursery dataset the rules generated are

```
if 7_not_recom ===> 8_not_recom  confidence = 1.0
```

Problem 2 continued on next page...

```
if 8_not_recom ===> 7_not_recom  confidence = 1.0

if 6_slightly_prob and 8_not_recom ===> 7_not_recom  confidence = 1.0

if 4_less_conv and 7_not_recom ===> 8_not_recom  confidence = 1.0

if 6_slightly_prob and 7_not_recom ===> 8_not_recom  confidence = 1.0
```

- For min_sup = 0.1 and min_conf = 0.6 and nursery dataset the rules generated are

```
if 0_great_pret and 7_not_recom ===> 8_not_recom  confidence = 1.0

if 0_great_pret and 8_not_recom ===> 7_not_recom  confidence = 1.0

if 7_not_recom ===> 8_not_recom  confidence = 1.0

if 6_slightly_prob and 8_not_recom ===> 7_not_recom  confidence = 1.0

if 6_slightly_prob and 7_not_recom ===> 8_not_recom  confidence = 1.0
```

- For min_sup = 0.1 and min_conf = 0.7 and nursery dataset the rules generated are

```
if 0_great_pret and 7_not_recom ===> 8_not_recom  confidence = 1.0

if 0_great_pret and 8_not_recom ===> 7_not_recom  confidence = 1.0

if 7_not_recom ===> 8_not_recom  confidence = 1.0

if 6_slightly_prob and 8_not_recom ===> 7_not_recom  confidence = 1.0

if 6_slightly_prob and 7_not_recom ===> 8_not_recom  confidence = 1.0
```

- For min_sup = 0.2 and min_conf = 0.5 and nursery dataset the rules generated are

```
if 8_not_recom ===> 7_not_recom  confidence = 1.0

if 7_not_recom ===> 8_not_recom  confidence = 1.0
```

- For min_sup = 0.2 and min_conf = 0.6 and nursery dataset the rules generated are

```
if 8_not_recom ===> 7_not_recom  confidence = 1.0

if 7_not_recom ===> 8_not_recom  confidence = 1.0
```

- For min_sup = 0.2 and min_conf = 0.7 and nursery dataset the rules generated are

```
if 8_not_recom ===> 7_not_recom  confidence = 1.0

if 7_not_recom ===> 8_not_recom  confidence = 1.0
```

f. The top 5 rules generated using lift for various combinations of support and confidence are listed below

g. For min_sup = 0.05 and min_conf = 0.5 and cars dataset the rules generated are

```
if 1_vhigh and 2_2 ===> 6_unacc  confidence = 13.9259259259

if 0_vhigh and 2_2 ===> 6_unacc  confidence = 13.9259259259

if 0_vhigh and 1_high ===> 6_unacc  confidence = 16.0

if 0_high and 1_vhigh ===> 6_unacc  confidence = 16.0

if 0_vhigh and 1_vhigh ===> 6_unacc  confidence = 16.0
```

h. For min_sup = 0.05 and min_conf = 0.6 and cars dataset the rules generated are

```
if 1_vhigh and 2_2 ===> 6_unacc  confidence = 13.9259259259

if 0_vhigh and 2_2 ===> 6_unacc  confidence = 13.9259259259

if 0_vhigh and 1_vhigh ===> 6_unacc  confidence = 16.0

if 0_high and 1_vhigh ===> 6_unacc  confidence = 16.0

if 0_vhigh and 1_high ===> 6_unacc  confidence = 16.0
```

i. For min_sup = 0.05 and min_conf = 0.7 and cars dataset the rules generated are

```
if 1_vhigh and 2_2 ===> 6_unacc  confidence = 13.9259259259

if 0_vhigh and 2_2 ===> 6_unacc  confidence = 13.9259259259

if 0_vhigh and 1_vhigh ===> 6_unacc  confidence = 16.0

if 0_vhigh and 1_high ===> 6_unacc  confidence = 16.0

if 0_high and 1_vhigh ===> 6_unacc  confidence = 16.0
```

j. For min_sup = 0.1 and min_conf = 0.5 and cars dataset the rules generated are

```
if 3_more and 5_low ===> 6_unacc  confidence = 9.0

if 3_2 and 4_big ===> 6_unacc  confidence = 9.0
```

```
if 4_big and 5_low ===> 6_unacc  confidence = 9.0

if 3_2 and 4_small ===> 6_unacc  confidence = 9.0

if 4_small and 5_low ===> 6_unacc  confidence = 9.0
```

k. For min_sup = 0.1 and min_conf = 0.6 and cars dataset the rules generated are

```
if 3_2 and 4_med ===> 6_unacc  confidence = 9.0

if 3_2 and 4_big ===> 6_unacc  confidence = 9.0

if 4_med and 5_low ===> 6_unacc  confidence = 9.0

if 3_2 and 5_low ===> 6_unacc  confidence = 9.0

if 4_small and 5_low ===> 6_unacc  confidence = 9.0
```

l. For min_sup = 0.1 and min_conf = 0.7 and cars dataset the rules generated are

```
if 3_2 and 4_small ===> 6_unacc  confidence = 9.0

if 4_med and 5_low ===> 6_unacc  confidence = 9.0

if 3_2 and 4_big ===> 6_unacc  confidence = 9.0

if 3_2 and 5_low ===> 6_unacc  confidence = 9.0

if 4_small and 5_low ===> 6_unacc  confidence = 9.0
```

m. For min_sup = 0.2 and min_conf = 0.5 and cars dataset the rules generated are

```
if 4_small ===> 6_unacc  confidence = 2.34375

if 3_2 ===> 6_unacc  confidence = 3.0

if 5_low ===> 6_unacc  confidence = 3.0

if 0_vhigh ===> 6_unacc  confidence = 3.33333333333

if 1_vhigh ===> 6_unacc  confidence = 3.33333333333
```

n. For min_sup = 0.2 and min_conf = 0.6 and cars dataset the rules generated are

```
if 4_small ===> 6_unacc  confidence = 2.34375

if 3_2 ===> 6_unacc  confidence = 3.0

if 5_low ===> 6_unacc  confidence = 3.0

if 0_vhigh ===> 6_unacc  confidence = 3.33333333333

if 1_vhigh ===> 6_unacc  confidence = 3.33333333333
```

o. For min_sup = 0.2 and min_conf = 0.7 and cars dataset the rules generated are

```
if 4_small ===> 6_unacc  confidence = 2.34375

if 3_2 ===> 6_unacc  confidence = 3.0

if 5_low ===> 6_unacc  confidence = 3.0

if 0_vhigh ===> 6_unacc  confidence = 3.33333333333

if 1_vhigh ===> 6_unacc  confidence = 3.33333333333
```

It can be seen that the rules generated by using lift are not common. Lift weights the confidence of the rule by dividing it by support of the antecedent in the data. This helps us avoid generating rules about items that always occur together and may not be interesting. The rules with an antecedent that occurs less get a higher confidence value than those with frequently occurring antecedents.

Problem 3

a. The code can be found in the directory `code/q3/recommender.py`.

The table 4 shows the Mean Absolute Rating (MAD) for different values of K for different distance metrics, on the datasets.

b. To include information about the movie genres and users, the algorithm was altered in the following way. Given a user and a movie whose rating has to be predicted, First we find the K closest movies on the basis of genres. Then, we find all users who have rated those movies. Next we find the K nearest users to the current user using the user rating information along with the age and gender information of the user. Finally we predict movie's rating as the mean rating of all movies by these k users.

The movie genre information was used in the first step because the 10M dataset does not have user information, and this algorithm could not be used for the 10M dataset using the user information. When finding the closest movies, the genre information alone was used because, the users ratings of a movie can differ widely across users and two movies with the same genre can have a high cosine distance than two movies with different genres but same ratings.

The table 5 lists the MAD for different values of K for the above algorithm on the datasets.

Table 4: Performance evaluation of on 100k dataset

K	Distance	u1	u2	u3	u4	u5
5	*euclidean*	0.919	0.915	0.906	0.908	0.906
10	*euclidean*	0.874	0.872	0.861	0.861	0.861
20	*euclidean*	0.848	0.847	0.838	0.836	0.839
30	*euclidean*	0.841	0.837	0.828	0.828	0.830
5	*manhattan*	0.905	0.884	0.775	0.854	0.892
10	*manhattan*	1.193	1.223	1.191	1.192	1.193
20	*manhattan*	1.053	1.060	1.072	1.107	1.151
5	*cosine*	0.846	0.840	0.840	0.839	0.845
10	*cosine*	0.823	0.818	0.815	0.813	0.821
20	*cosine*	0.818	0.809	0.804	0.802	0.808
30	*cosine*	0.819	0.808	0.803	0.801	0.807

Table 5: Performance evaluation on 100k dataset

K	Distance	u1	u2	u3	u4	u5
5	*euclidean*	0.841	0.854	0.778	0.848	0.893
10	*euclidean*	1.192	1.223	1.198	1.192	1.105
20	*euclidean*	1.053	1.106	1.107	1.107	1.151
30	*euclidean*	1.181	1.162	1.118	1.180	1.156
5	*manhattan*	0.850	0.854	0.775	0.854	0.892
10	*manhattan*	1.193	1.223	1.191	1.192	1.193
20	*manhattan*	1.053	1.060	1.072	1.107	1.151
30	*manhattan*	1.181	1.162	1.118	1.180	1.156
5	*cosine*	0.096	1.168	1.038	1.056	0.806
10	*cosine*	1.143	0.891	1.136	1.181	1.024
20	*cosine*	1.123	1.120	1.058	1.082	1.220
30	*cosine*	1.166	1.169	1.072	1.185	1.194

c. The above algorithm was run on the 10M dataset. Since the dataset is very large it takes a long time to execute the algorithm on the dataset and all 2 million samples could not be tested. The number of test samples that were encountered were 500,000 out of 2 million.

The table 6 shows the MAD ratings for 3 different values of K for the algorithm used in part b

Table 6: Performance evaluation on 10M dataset

K	Distance	u1	u2	u3	u4	u5
5	*cosine*	0.847	0.865	0.852	0.854	0.836
10	*cosine*	0.822	0.824	0.832	0.830	0.831
30	*cosine*	0.821	0.820	0.830	0.833	0.821

d. the results may be improved by trying different distance functions such as Hamming distance or Jaccard distance. Further, a weighted mean of ratings of k nearest neighbors can be used instead of simply using the mean. There are many ways to use movie information and other methods can be used which take into account other factors such as the year of release of the movie and the actors in the movie. The data of user reviews is timestamped which can be utilized to capture trends by other users to predict a user's

Problem 3 continued on next page...

rating using time series analysis. A user's known movie reviews and predicted movie reviews can be used to predict the next movie the user will see with the help of these timestamps.