Lab Assignment – Unsupervised Learning

Task 1: Dimensionality reduction, Latent semantic indexing

1b) For querying I have randomly chosen document 643. I implemented three different transfer parameters, namely words_bow, doc_bow and texts_bow, in order to demonstrate the different outcomings resulting from varying these. For the sake of readability, I only print the best ten results, of the query. In the following I will briefly summarize my findings and interpretations of the results:

First query based upon manually picked words from document 643
After reading through the document I chose the first three words ‘peir’, ‘yuan’ and ‘yeh’

```
Place        Doc#       Similarity score
0           (9009,      0.85382783)
1           (3260,      0.49242395)
2           (6015,      0.4506209)
3           (10011,     0.42895377)
4           (5074,      0.40640038)
5           (4235,      0.40590054)
6           (7780,      0.3908772)
7           (266,       0.3877284)
8           (138,       0.37643796)
9           (4162,      0.37420386)
```

As we can see the result is rather disappointing. First of all, and this might be the biggest disenchantment, the chosen document (643) is not even in the list of the best ten results. The reason for this lies in two aspects; first, the number of chosen words is utterly low, only three words seem to be too little in order to achieve pattern identification well. Second, the chosen words might not be special, deterministic enough in order to differentiate from other documents. Due to the weak performance I decided to enrich the list of words in hopes of a better result. The second version of the ‘words_bow’ was: ‘Old’, ‘Testament’, ‘Moslem’, ‘God’, ‘manuscript’, ‘Isaiah’ and provided the following results:

```
0 (559, 0.88805586)
1 (10579, 0.83293283)
2 (11004, 0.8318608)
3 (10761, 0.8186557)
4 (1566, 0.8129226)
5 (4932, 0.80008)
6 (3258, 0.78843707)
7 (10734, 0.78539664)
8 (8643, 0.7836587)
9 (11095, 0.7683506)
```

Unfortunately, still no sight of document 643. It seems like there are still not enough key words and/or these are not indicative of the text in order to find the desired document. Let us move on to the next query.

Second query based upon a bag of words (bow) of the non-preprocessed document 643

```
0 (643, 0.64683616)
1 (1998, 0.51491475)
2 (6781, 0.49600124)
3 (11095, 0.48540366)
4 (3973, 0.48340416)
5 (11300, 0.47347844)
6 (8051, 0.46233577)
7 (8444, 0.45994574)
8 (3086, 0.45978504)
9 (3577, 0.4591573)
```

Finally, but not quite striking, the upfront chosen document is found and even at the first place with the best similarity score of all documents within the data. Although our chosen document is in the first place the similarity score of only 0.6468 is relatively low. This is due to the noise in the newly created bow. By passing every word of the document a lot of irrelevant words such as ‘it’, ‘is’, ‘and’, ‘but’ etc. are subject to the matching between deterministic words of the LSI model, resulting in a quite ok but far from good similarity score. The gap between our desired document 643 and the next best match is significant with a score reduction of 0.1319. We can clearly see how the first reduction in similarity between best and second-best matching result is quite large and exponentially decreases afterwards. The difference in score between the ninth and tenth best result is with only 0.000628 nearly non-existent.

Third query based upon a bag of words (bow) of the preprocessed document 643

```
0 (643, 0.96114063)
1 (3960, 0.6998903)
2 (11095, 0.6980866)
3 (6467, 0.68601227)
4 (6763, 0.67640543)
5 (9637, 0.6748852)
6 (7654, 0.6700207)
7 (2904, 0.65322024)
8 (10579, 0.6481948)
9 (2529, 0.6425407)
```

In the end we finally made it. We found document 643 at the first place with an excellent similarity score of 0.9611. As mentioned above in the second query the reason for the low score was the attached noise, because not a relevant subset of words was chosen, but every single word of the text was part of the bow. Within this query we removed the noise, thus leading to the best query result.
We additionally see that the gap in score for document 3960 is even bigger as in our second query. With a score difference of 0.2612 (almost double of the first gap-analysis of query number two) we can argue that the first result is with high probability by far the best matching document to our input.

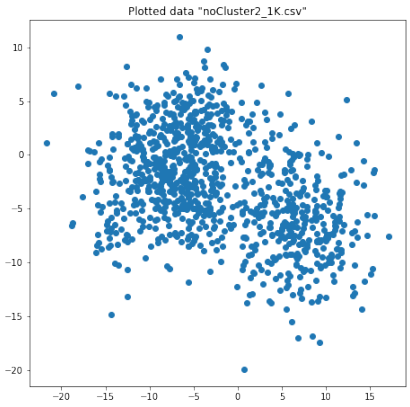
Lukas Sorer a01250600
Foundation of Data Analysis SS2020
Lab Assignment – Unsupervised Learning

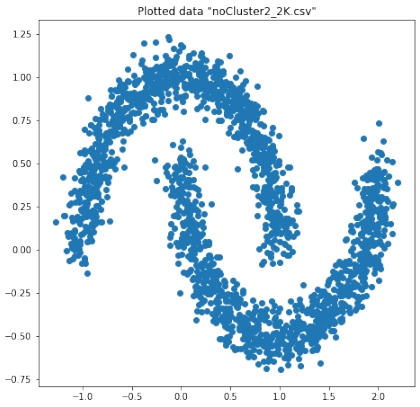
Task 2: Clustering

2a) Import data:

```
data1 = np.loadtxt('noCluster2_1K.csv', skiprows=1, delimiter=',')
data2 = np.loadtxt('noCluster3_1K.csv', skiprows=1, delimiter=',')
data3 = np.loadtxt('noCluster2_2K.csv', skiprows=1, delimiter=',')
```

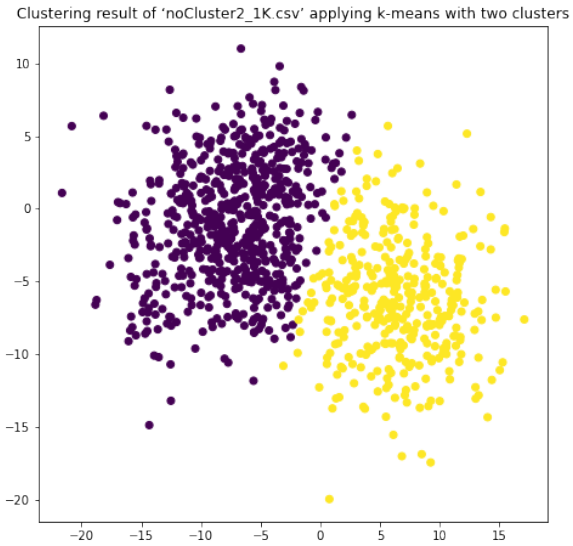
Plot data:



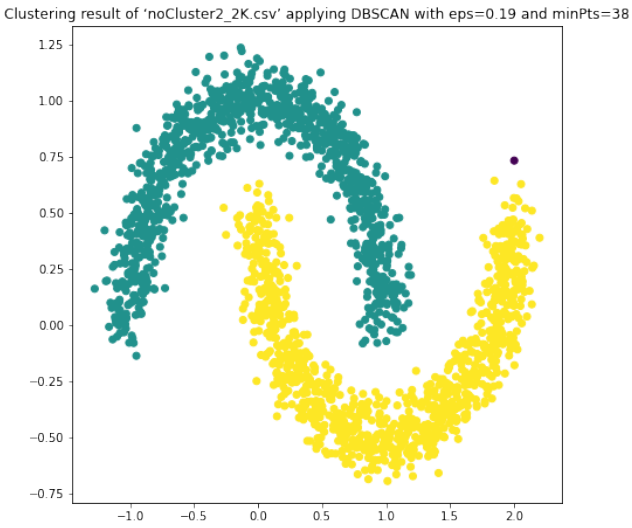


2b) I tried several clustering algorithms, in the following I will present only the best performing ones according to the NMI and Silhouette score and visualize them. Additionally, I will provide which parameters I used achieving this. The last question of task 2b I'd like to answer visually in task 3.

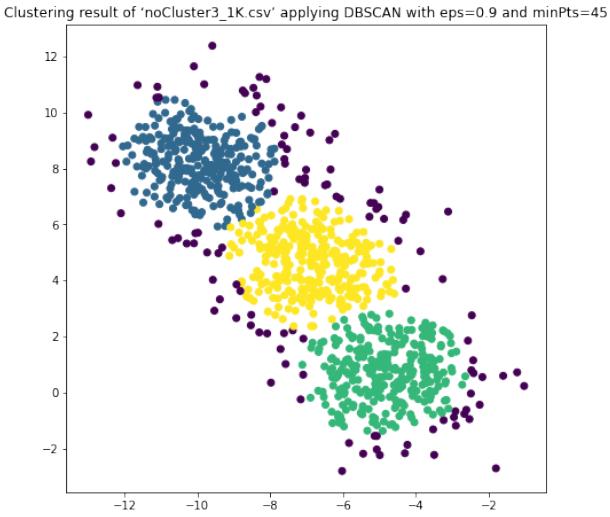
According to my evaluation the best clustering results for the dataset 'noCluster2_1K.csv' is provided by the k-means algorithm when the parameter n_clusters is set to two. The NMI delivers a score of 0.7233 and the Silhouette score is 0.5286. (Although this result is head to head with the score learned from Average-Linkage clustering when the number of clusters is also two)

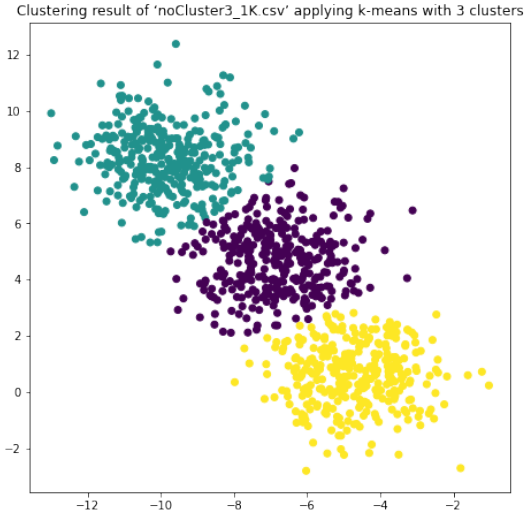


Evaluating the second dataset i.e. ‘noCluster2_2K.csv’ I found that the DBSCAN clustering algorithm delivers the best results. With an amazing NMI score of 0.9971 and a moderate Silhouette score of 0.1987 it performs best by choosing eps=0.19 and minPts=38. Additionally, we can see that it almost perfectly labels the datapoints according to the cluster they belong. Only one datapoint is considerate as outlier, here we see the strength of density-based clustering algorithms.

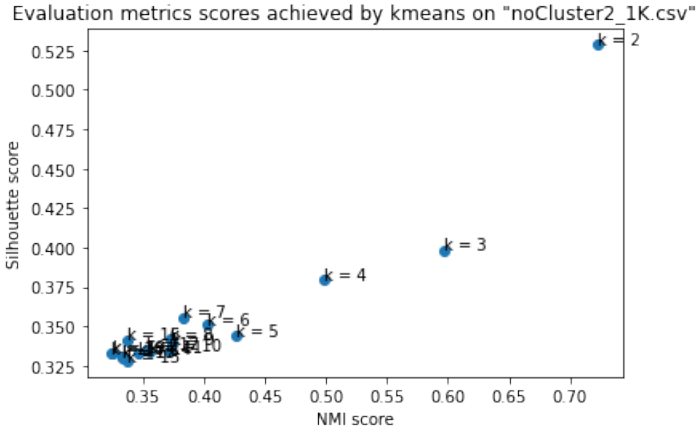


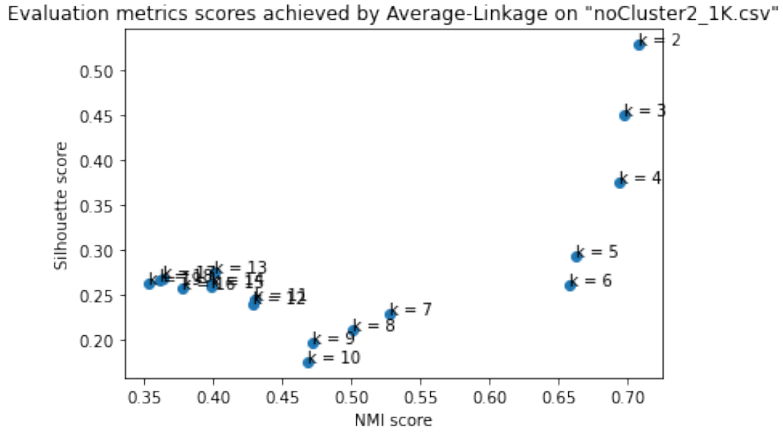
Finally, we analyze dataset number three ‘noCluster3_1K.csv’. Here I’d like to present two results, one that is based upon the evaluations scores and one, that is intuitively a result that I personally would prefer. The result I like the most, but has a strikingly low evaluation scoring, is based on the DBSCAN clustering using eps=0.9 and minPts=45. If we believe in the evaluation scores the best results are delivered using k-means with three clusters.

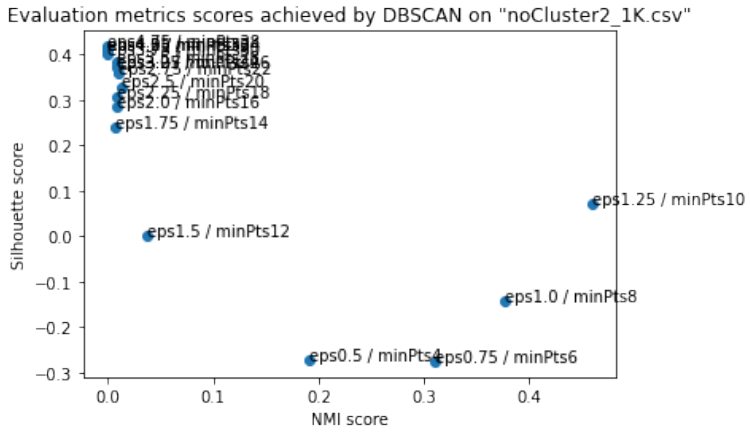


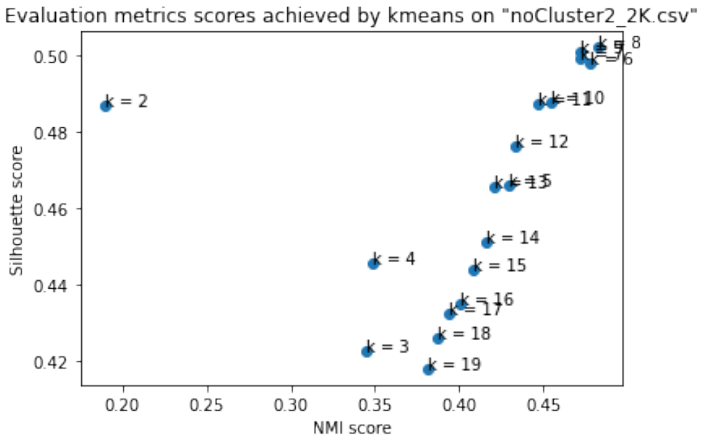


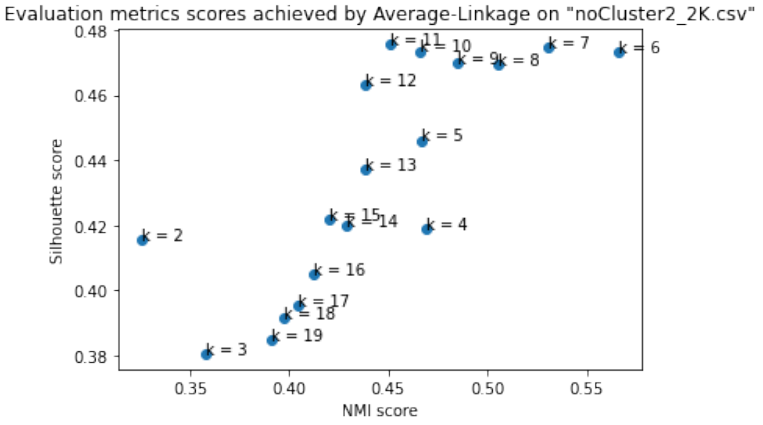
3c) Evaluation of 3 different datasets with each 3 different clustering algorithms with respect to the evaluation metrics Normalized Mutual Information and Silhouette Score.

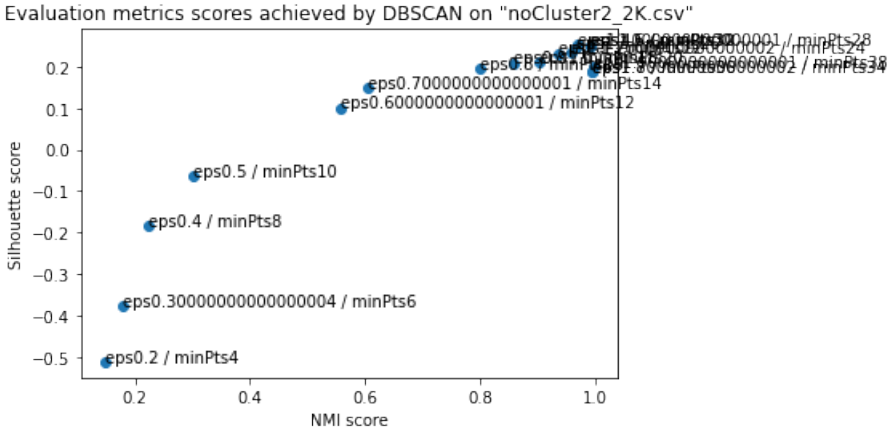


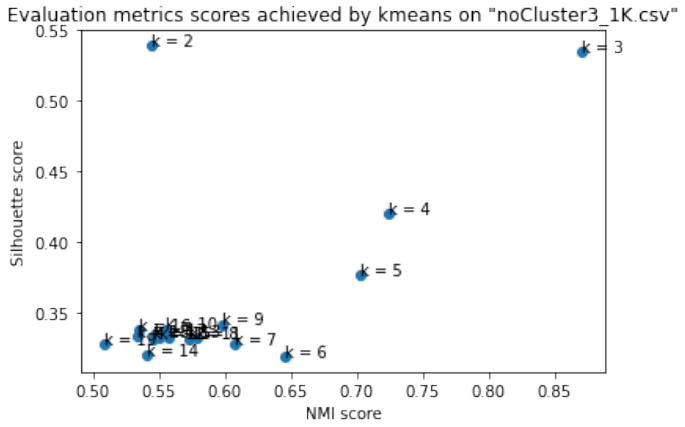


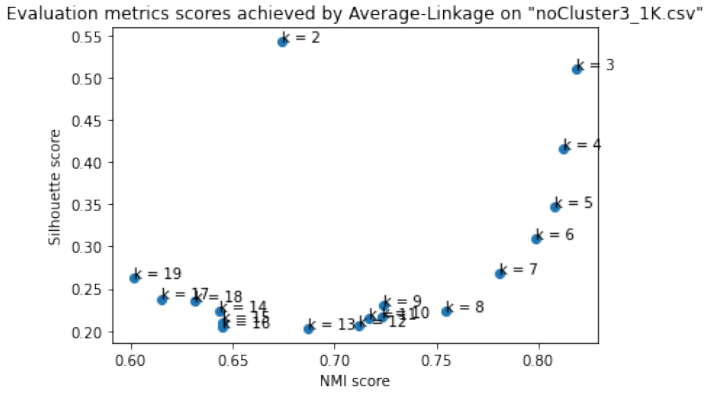


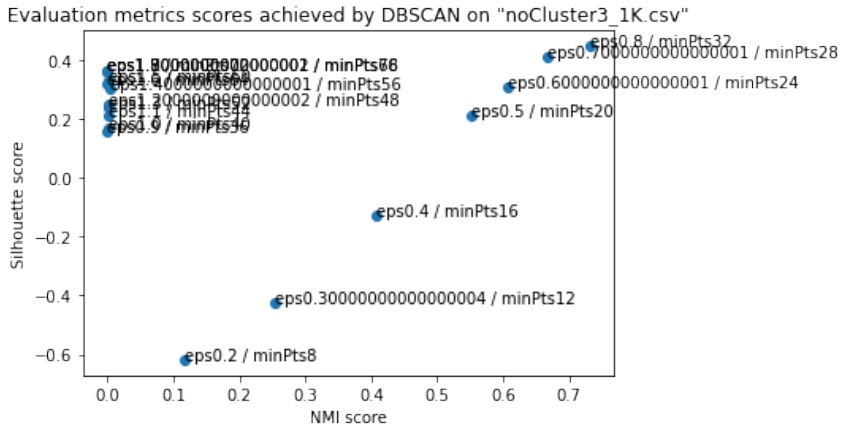

Evaluation metrics scores achieved by kmeans on "noCluster2_2K.csv"
Silhouette score
NMI score
k = 2
k = 3
k = 4
k = 12
k = 14
k = 15
k = 16
k = 17
k = 18
k = 19


Evaluation metrics scores achieved by Average-Linkage on "noCluster2_2K.csv"
Silhouette score
NMI score
k = 2
k = 3
k = 4
k = 5
k = 6
k = 7
k = 8
k = 12
k = 13
k = 16
k = 19


Evaluation metrics scores achieved by DBSCAN on "noCluster2_2K.csv"
Silhouette score
NMI score
eps0.7000000000000001 / minPts14
eps0.6000000000000001 / minPts12
eps0.5 / minPts10
eps0.4 / minPts8
eps0.30000000000000004 / minPts6
eps0.2 / minPts4







4d) Brief explanation of NMI and Silhouette Score

NMI is an external measure for determining the quality of clustering. Generally speaking, Mutual Information tells us about the reduction in entropy of class labels that we get if the labels are known. Because it's normalized, we can compare the NMI between different clustering result which have different number of clusters.

The Silhouette Score is an internal measure of how similar an object is to its own cluster compared to other clusters. The large advantage to NMI for example is that we don't need the actual labels as it only compares distances between datapoints.

Task 3: Apriori Algorithm for Recommender System

3a) see oneItem.txt in folder

3b) see patterns.txt in folder

3c) Unfortunately there is a problem with the code based recommender, so I manually load the data from patterns.txt into a spreadsheet tool and filtered for frequent item sets that contain both movies "Ant-Man and the Wasp" and "Spider-Man: Far from Home" the only result, and therefore the best recommendation was "Spider-Man: Into the Spider-Verse" with a confidence of 465.