Task -1

(a) Task (a) was also executed via Python. The documented code can be found in the zip file.
Building on the instruction of Latent Semantic Indexing (LSI) in the lecture, tf-idf measure stands for: "Term Frequency – Inverse Document Frequency".
In general, this technique determines the quantification of a word (phrase) in a document (similar to the introduced cosine similarity in the lecture), assigns a weight to each word and thereby sets the significance of the word in the document.
As we use the fetch_20newsgroups feature of sklearn, the documents (corpus) have benn read in as a matrix and by using the .TfidModel() and tfidf() function to create the relevant word (-combinations) to check similarity with one example query.

Simplified, the tf-idf measure can be seen as:
tfidf = term frequency * inverse document frequency
where term frequency is individual for each document (counting occurrence in specific document), whereas in document frequency the occurrences of in all documents (all matrix columns) is determined. After the inversion the document frequency can be used to measure the informativeness on the present terms.

(b) Task (b) was also executed via Python. The documented code can be found in the zip file.
I chose the 502th document of the generated dataset by random and run the query with the python implementation.
As set by default, the number of topics (num_topics) was 200, relevant for the latent dimensions.
After creating the query via doc2bow.lower().split() (rest of the documentation can be found in the .py file) and converting the query to the LSI space a similarity query was run against the corpus.

As the selected 502th document consists of:
I was recently thumbing through the 1993 Lemon-Aid New\nCar Guide. What I found was a car would be given a 'Recommended'\nunder the picture while a few sentences later noting how a\ndriver and passenger were virtually guaranteed to be killed\nin a front end collision. The most highly recommended small\ncar (The Civic) has the worst crash rating of all of the small\ncars listed. There were many such cases of 'great' vehicles\nwhere you wouldn't survive an accident. Is it only me, or is\nsafety not one of the most important factors when buying a car?\n\n\n"
As can be seen the document mainly describes topics around a car (yellow) and car related topics (red).
The most similar documents found via LSI have been document 11063, which consists of the phrase "A million posters to call "you car drivers" and he chooses me, a non car owner", which is really adequate as it describes a car related topic.

The second most similar document was the document 10521 which describes the relevant criteria for a customer to buy a car, indicating a proper implementation of the LSI in python.

Task -2

(a) Plot of noCluster2_1K

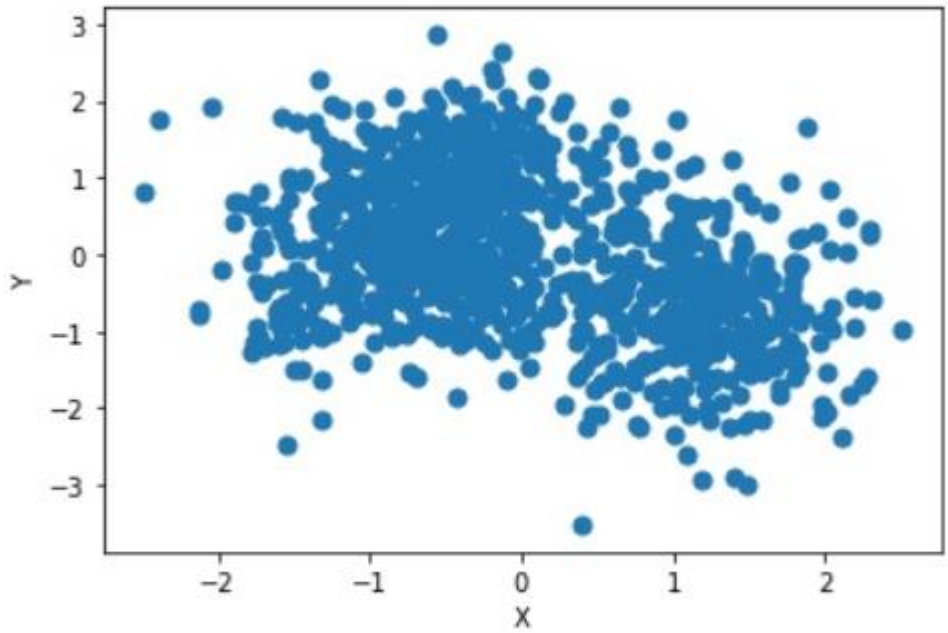


Figure 1 Plot of noCluster2_1K

Plot of noCluster2_2K

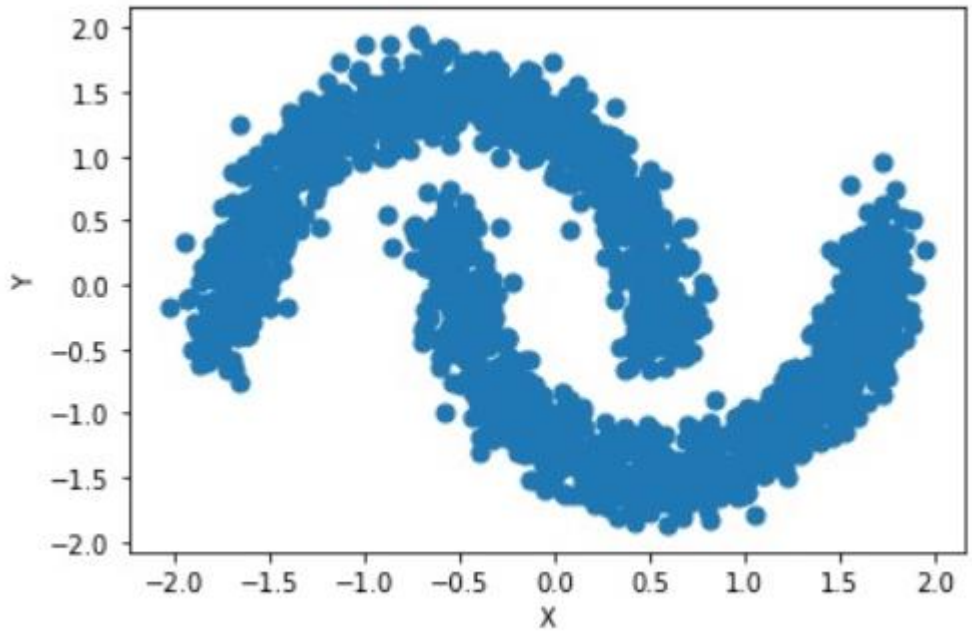


Figure 2 Plot of noCluster2_2K

Plot of noCluster3_1K

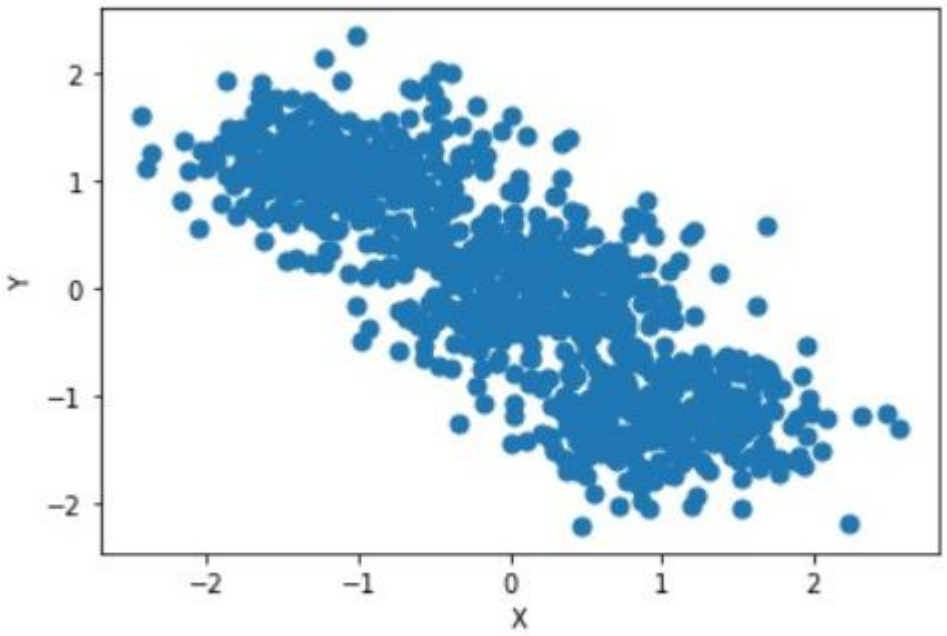


Figure 3 Plot of noCluster3_1K

The data of X and Y was standardized with help of the sklearn.preprocessing StandardScaler for all imported Clusters

(b) DBSCAN:

noCluster2_1K

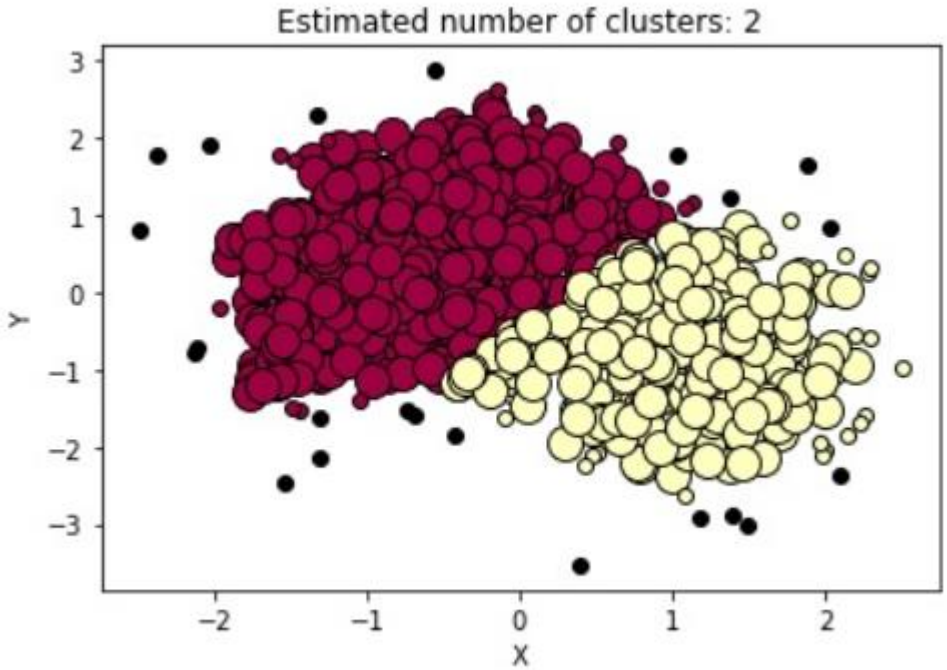


Figure 4 DSBCAN on noCluster2_1K

noCluster2_2K

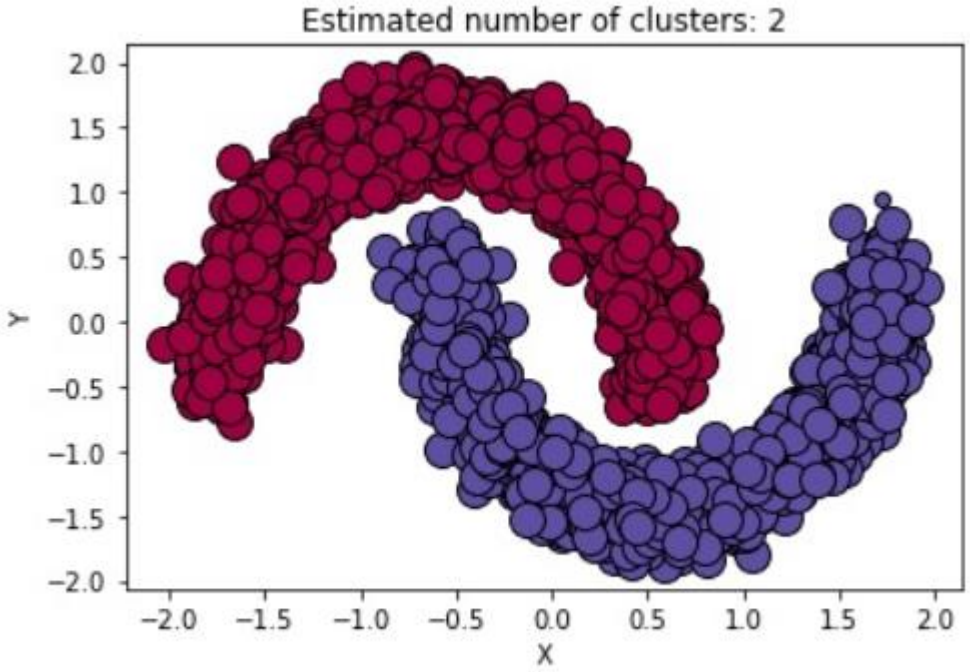


Figure 5 DSBCAN on noCluster2_2K

noCluster3_1K

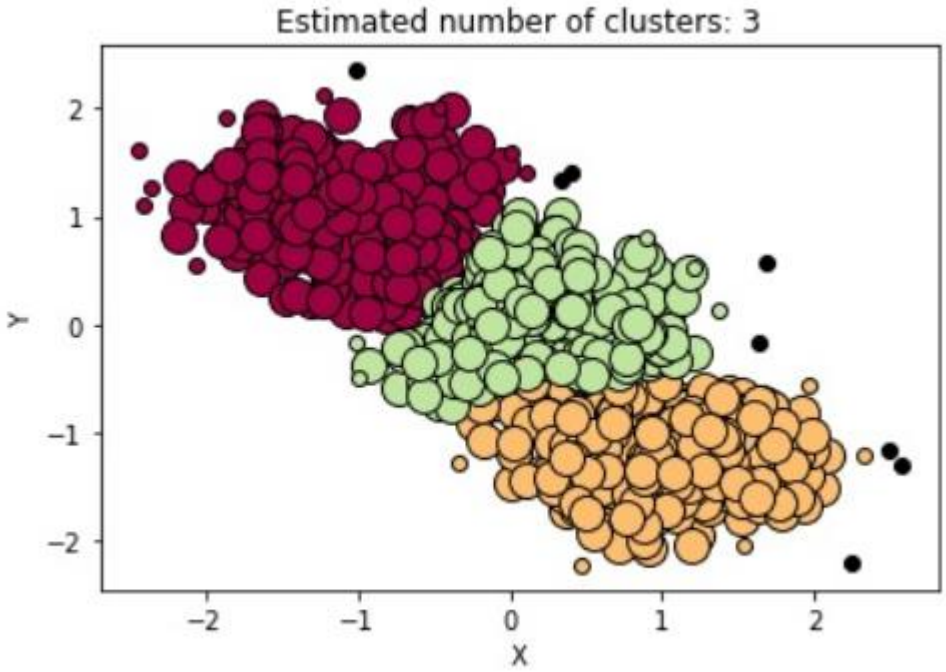


Figure 6 DSBCAN on noCluster3_1K

DSBCAN was executed using sklearns DSBCAN and metrics. For all three clusters the function DSBCAN(eps=0.4, min_samples = 9, s=50). Therefore the maximum distance between two samples to be considered as in the neighborhood of the other was set to 0.4 and the number of samples in the neighborhood for a point to be considered a core point to 9. The distance was calculated on default settings, meaning with Euclidean distance.

As can be seen, for the clusters noCluster3_1K and noCluster2_1K some noise points are present, while 3 and 2 number of clusters have been calculated.

NoCluster2_2K shows no noise points and was split into 2 clusters.

KMeans:

noCluster2_1K

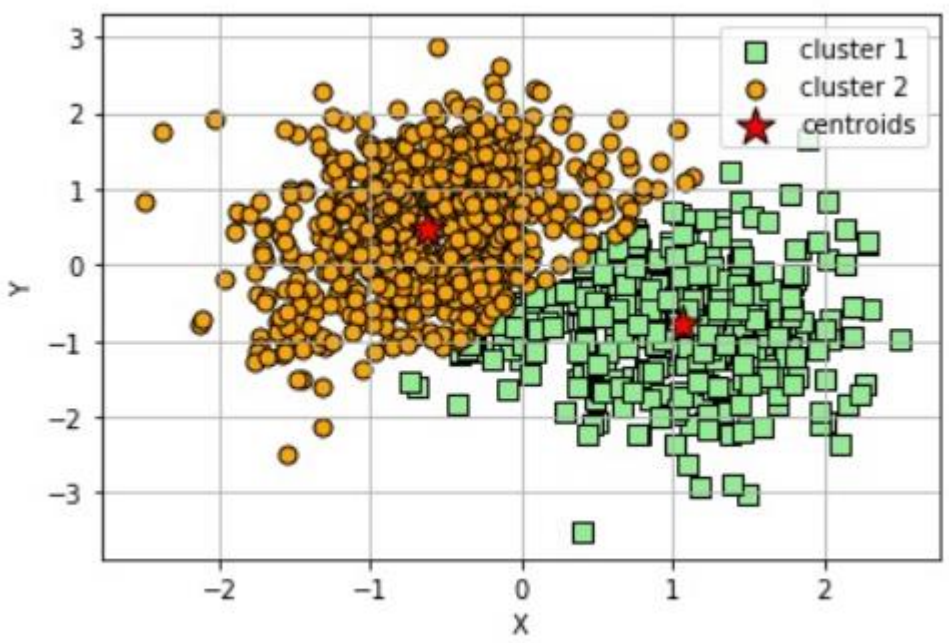


Figure 7 KMeans on noCluster2_1K

noCluster2_2K

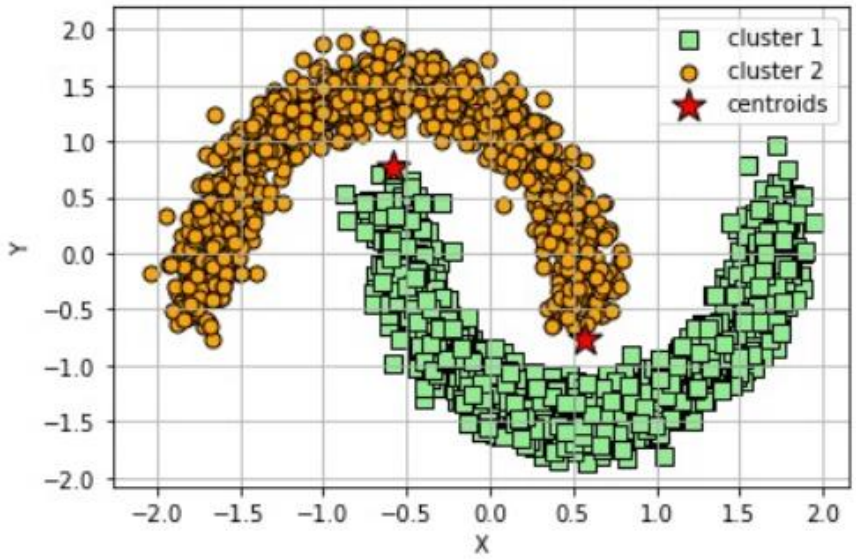


Figure 8 KMeans on noCluster2_2K

noCluster3_1K

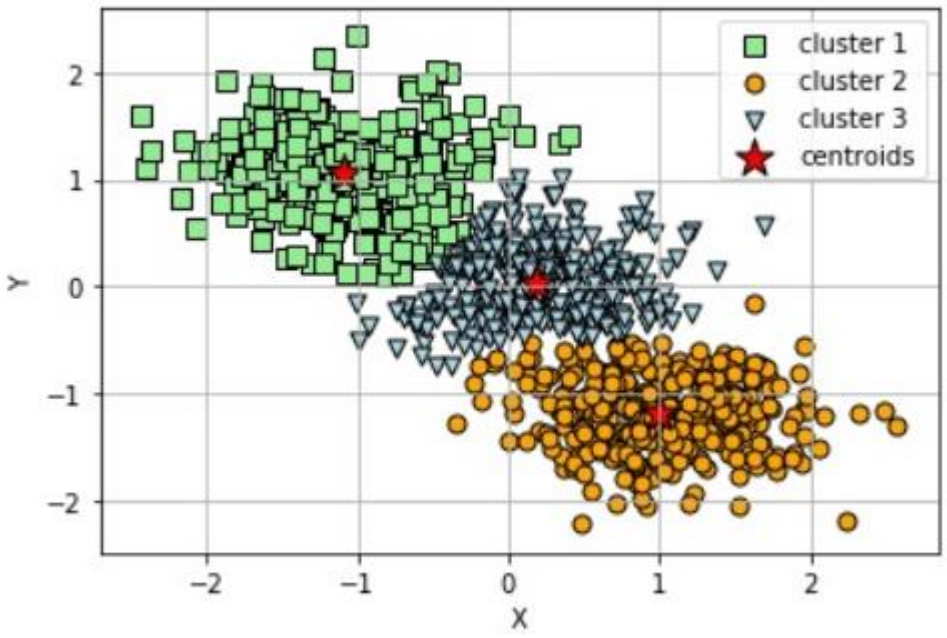


Figure 9 KMeans on noCluster3_1K

The KMeans clustering technique was executed using sklearns KMeans. The parameters of the function have not been chosen equivalently, therefore all 3 functions are listed here:

For noCluster2_1K: KMeans(n_clusters = 2, init='k-means++', n_init=10, max_iter=300, tol=1e-04, random_state=0)

For noCluster2_2K: KMeans(n_clusters = 2, init='k-means++', n_init=10, max_iter=300, tol=1e-04, random_state=0)

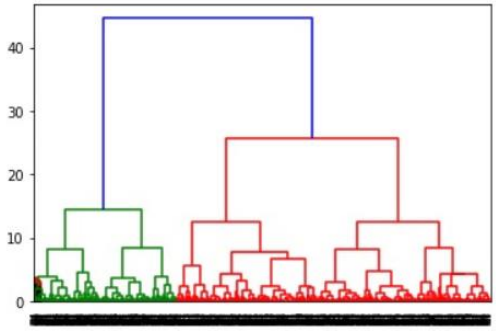
For noCluster3_1K: KMeans(n_clusters = 3, init='k-means++', n_init=10, max_iter=300, tol=1e-04, random_state=0)

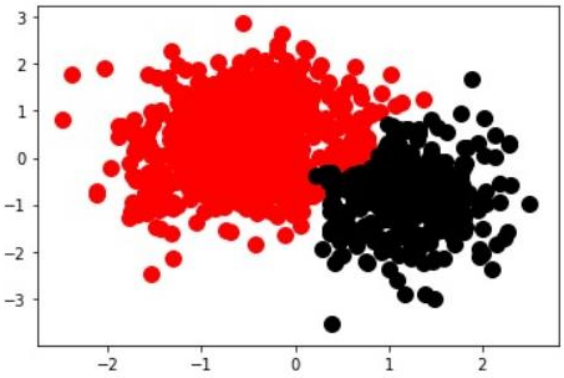
As can be seen above the only difference in the parameters is the deviation for noCluster3_1K in n_clusters=3. The 3 was chosen simply because the human intuition approximates 3 clusters for this data, n_clusters = 2 for the remaining clusters have been chosen for the same reason.

The init parameter was set to 'k-means++' to ensure that the algorithm is guaranteed to find a solution that is O(log(k)) competitive to the optimal Kmeans solution and therefore avoids the problem of suboptimal clustering.

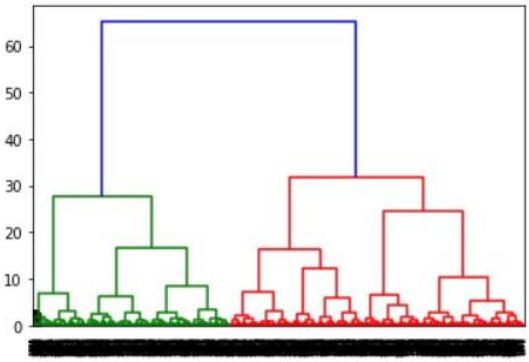
Average Link:

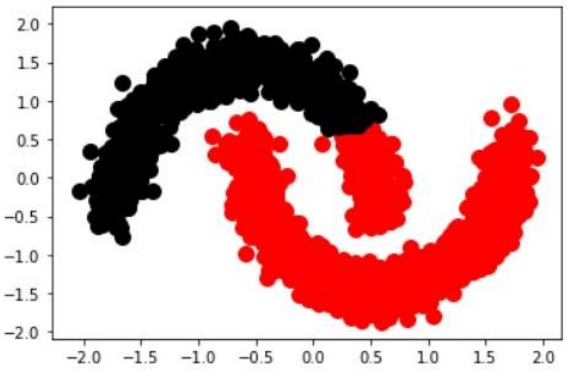
noCluster2_1K



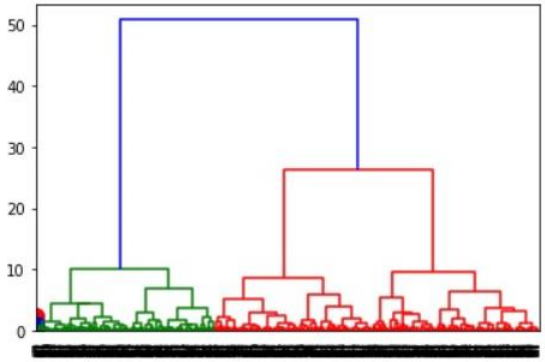


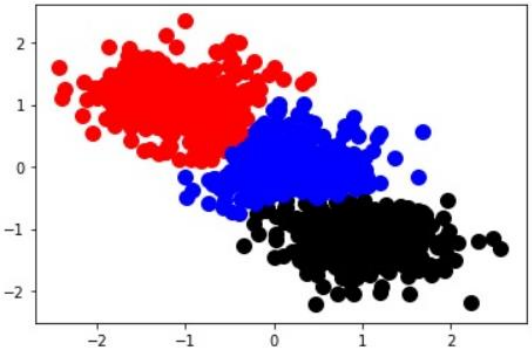
noCluster2_2K





noCluster3_1K





(c) Evaluate

DBSCAN:

Silhouette Score: 0.4249

Silhouette Score: 0.3827

Silhouette Score: 0.4896

NMI: 0.8186

NMI: 1.0

NMI: 0.9488

KMeans:

Silhouette Score: 0.6160

Silhouette Score: 0.5688

Silhouette Score: 0.6821

NMI: 1.0

NMI: 1.0

NMI: 1.0

Average Link:

Silhouette Score: 0.4638

Silhouette Score: 0.4562

Silhouette Score: 0.5389

NMI: 0.7357

NMI: 0.6239

NMI: 0.7449

!Be aware that NMI only indicate how much is learned about classlabels when cluster ID is known. In other words meaningfulness of both scores depends, already assumed by the fact that one is an external evaluation (NMI) and one an internal evaluation (SC), see (d).

(d)

The Silhouette Score uses the distance within a cluster and the mean nearest cluster distance to calculate the value for each sample. The mean intra-cluster distance and the mean nearest-cluster distance are calculated for each sample and the resulting mean Silhouette Score lies between -1 and 1. 1 is representing the optimum score, while values around 0 indicate overlapping clusters and negative values even indicate a wrong assignment to a cluster. By this the degree of separation between clusters is determined.
It is an internal evaluation measure meaning that it is only meaningful if the clustering technique is principally appropriate for the problem.
For all 3 techniques the silhouette score lies between 0 and 1, showing the best values parallel to NMI for the KMeans method. Values between 0,57 and 0,68 indicate well separated clusters where most samples are for away from the neighboring clusters. In difference to the NMI results the second best values for the Silhouette Score are achieved through the average link technique. As can be seen in the images above, the average link technique does not provide the same clusters as KMeans ("splitting the upper bow in two clusters"), but accepting this assignment the Silhouette Score shows good values.
DBSCAN presents the lowest Silhouette score around 0,4, which is acceptable by looking at the plotting of the data, where a strict border of the clusters is hard to identify/set.

Normalized Mutual Information (NMI) is calculated by using the class labels, cluster labels, entropy and the mutual information between the class and cluster labels. It represents a normalization of the mutual information score and shows results between 0 and 1, where 0 inidcates no mutual information and 1 a perfect correlation. External Evaluation measure to map the clustering result and given classes.

Based on the provided real class labels as third column for the noCluster_ datasets, the NMI was calculated. From the calculated values of (c) especially the results of the KMeans technique are interesting as all 3 clusters result in a score of 1, meaning that every class label was assigned correctly. As the Kmeans algorithm is good in detecting structure of the data if the clusters have a spherical shape, the result is understandable. The geometric shapes are not too complicated and it was easy for a human to estimate the number of clusters by examining the ploted data.
The results for DBSCAN are also good and could have even been higher if the parameters (minPoints, distance) would have defined by lower values.
The lowest NMI values are provided by the average link method, although with still good values.

Task -3 Apriori Algorithm for Recommender Systems

(a) Task a was executed via Python. The documented code and the resulting frequent item set "oneItems.txt" can be found in the zip file.

(b) As in part (a) the documented code and the resulting frequent itemsets "patterns.txt" can be found in the zip file.

(c) User likes: "Ant-Man and the Wasp", "Spider-Man: For From Home".
Based on this knowledge the confidence of the rule
{"Ant-Man and the Wasp", "Spider-Man: Far from Home"} → {Y}
was evaluated, by searching for the movie X which maximizes the confidence.

The calculation was done and is documented in the python file. To calculate the confidence, the following formula has been used:

$$conf(X \rightarrow Y) = \frac{Supp(X \cup Y)}{Supp(X)}$$

The maximum confidence was found for X = {"Avengers: Infinity War – Part II"} with a confidence score of 96,34% or:

$$conf(\{\text{“Ant} - \text{Man and the Wasp”, “Spider} - \text{Man: Far from Home”}\} \rightarrow \{\text{“Avengers: Infinity War} - \text{Part II”}\})$$

$$= \frac{Supp(\{\text{“Ant} - \text{Man and the Wasp”, “Spider} - \text{Man: Far from Home”}\} \cup \{\text{“Avengers: Infinity War} - \text{Part II”}\})}{Supp(\{\text{“Ant} - \text{Man and the Wasp”, “Spider} - \text{Man: Far from Home”}\})}$$

$$= \frac{579}{601} = 0{,}9633943428$$