MACHINE LEARNING

TASK 2 - REPORT

Karolina Czerczak & Bartosz Lewandowski

04.05.2022

PART A

In the first part of this task we will present the operation of clustering algorithms on a data set VideoGamesSales.csv which describes data on the sales of games. Our goal is to identify certain groups of products so as to help publishers plan their release and promotion schedules for next year.

Due to the nature of the data, we will split clustering into two parts. Firstly, we will cluster the data in general, with the best possible outcome. Secondly, we will cluster the data in search of outliers. For this purpose, we will use hierarchical clustering with the average measure and DBSCAN-type clustering with the Gower measure.

Shortly about two methods:

1. "Hierarchical clustering is a general family of clustering algorithms that build nested clusters by merging or splitting them successively. This hierarchy of clusters is represented as a tree (or dendrogram). The root of the tree is the unique cluster that gathers all the samples, the leaves being the clusters with only one sample. See the Wikipedia page for more details."

   - **Ward** minimizes the sum of squared differences within all clusters. It is a variance-minimizing approach and in this sense is similar to the k-means objective function but tackled with an agglomerative hierarchical approach.
   - **Maximum** or **complete linkage** minimizes the maximum distance between observations of pairs of clusters.
   - **Average linkage** minimizes the average of the distances between all observations of pairs of clusters.
   - **Single linkage** minimizes the distance between the closest observations of pairs of clusters.

   Scikit-learn.org

2. "The DBSCAN algorithm views clusters as areas of high density separated by areas of low density. Due to this rather generic view, clusters found by DBSCAN can be any shape, as opposed to k-means which assumes that clusters are convex shaped. The central component to the DBSCAN is the concept of *core samples*, which are samples that are in areas of high density. A cluster is therefore a set of core samples, each close to each other (measured by some distance measure) and a set of non-core samples that are close to a core sample (but are not themselves core samples)."

   Scikit-learn.org

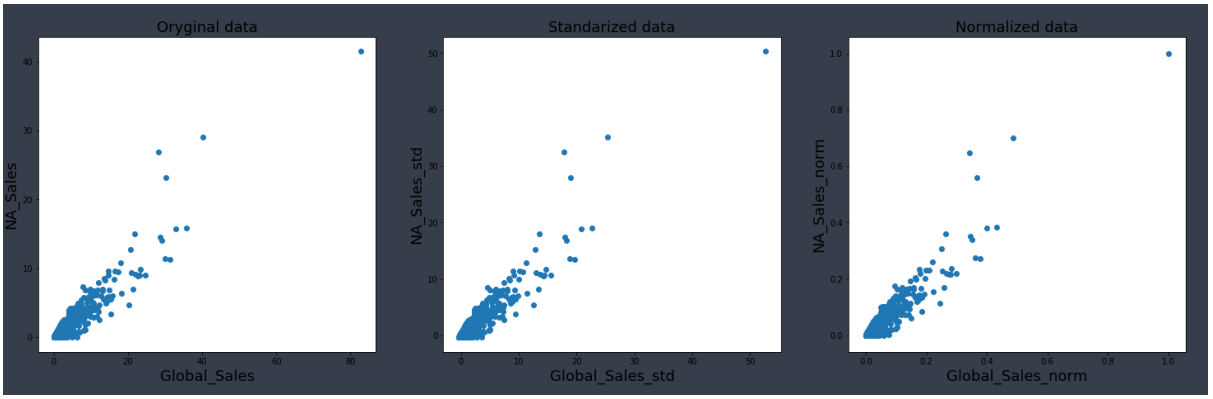
3. “Gower Similarity (GS) was first defined by J. C. Gower in 1971 [2]. To calculate the similarity between observations i and j (e.g., two customers), GS is computed as the average of partial similarities (ps) across the m features of the observation.”

$$GS_{ij} = \frac{1}{m}\sum_{f=1}^{m} ps_{ij}^{(f)}$$

(Similarity between observations i and j. Having each observation m different features, either numerical, categorical or mixed.)

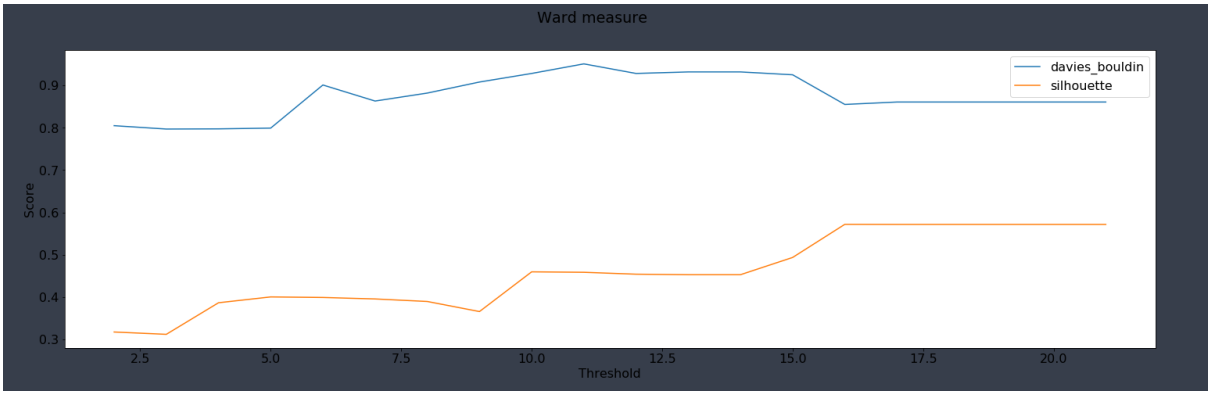
Towardsdatascience.com

Note. There was no need to standardize or normalize data due to the same scale of “Sales”.

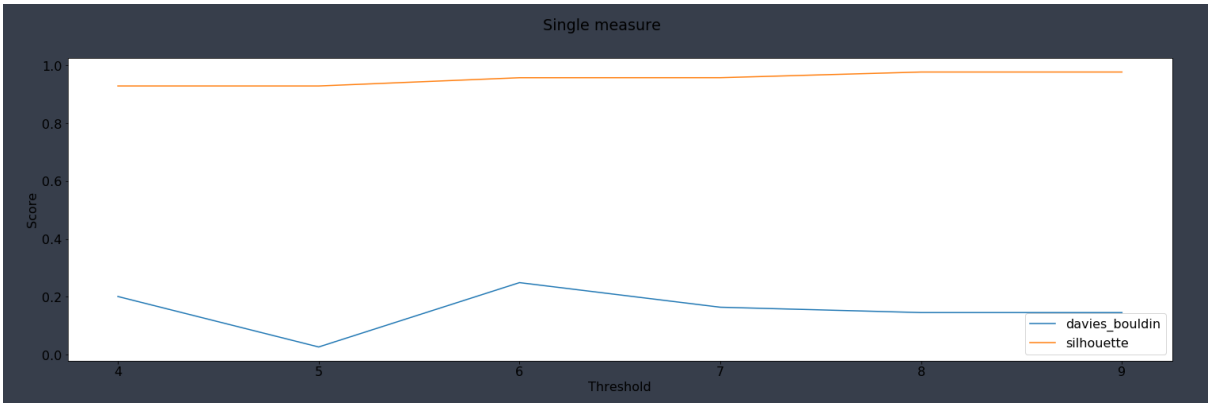


(Comparing data before and after standardization/normalization.)

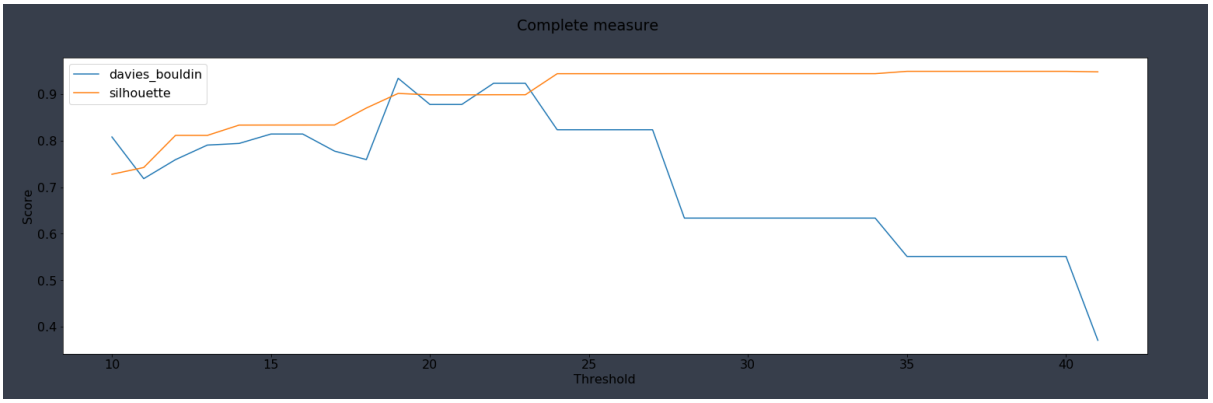
We start by comparing clustering performance with different parameters in hierarchical clustering. For performance measure we will use *Davies-Bouldin score (the minimum score is zero, with lower values indicating better clustering)* and *mean Silhouette Coefficient (the best value is 1 and the worst value is -1. Values near 0 indicate overlapping clusters).*



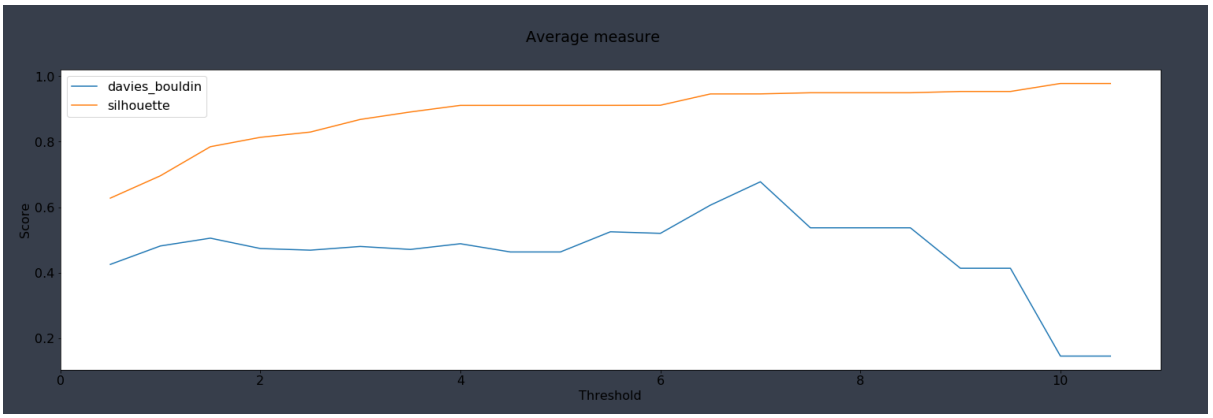
(method='ward' ; metric='euclidean')



(method=’single' ; metric='minkowski')

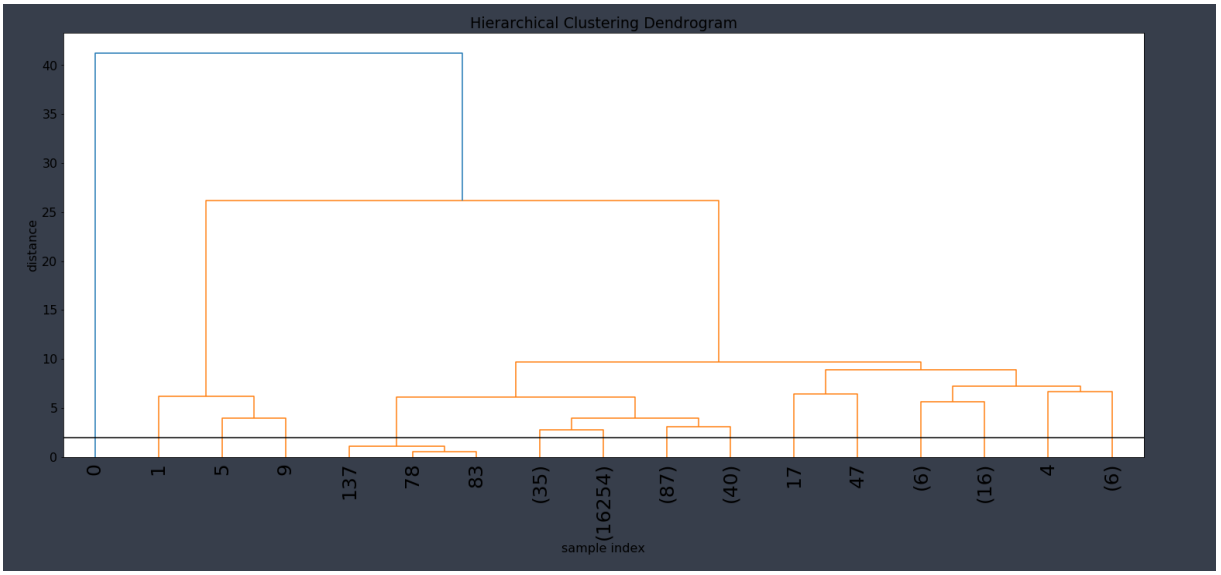


(method='complete' ; metric='seuclidean')

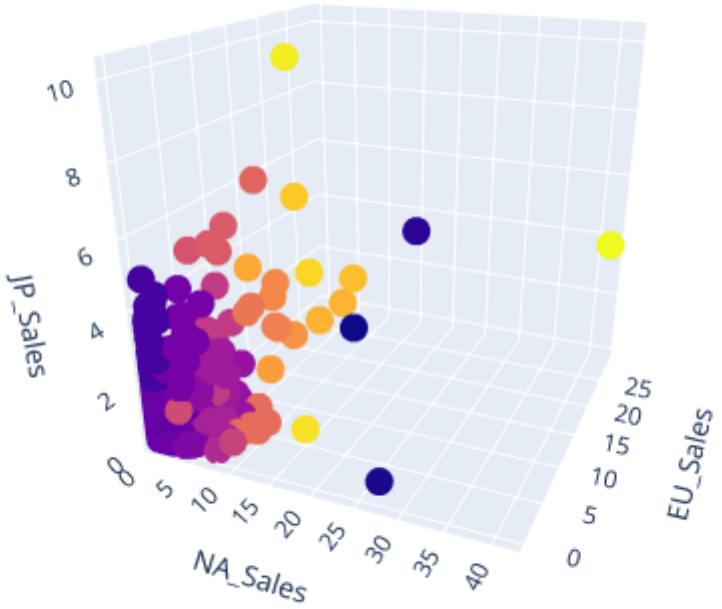


(method='average’ ; metric='chebyshev')

We will choose a method ‘average’ with ‘chebyshev’ metric and a threshold of 2, which resulted in the following:



(dendrogram)




```
davies_bouldin_score 0.4739599609326761
silhouette_score 0.8133689121758959
```

(Hierarchical clustering with average method and chebyshev measure)

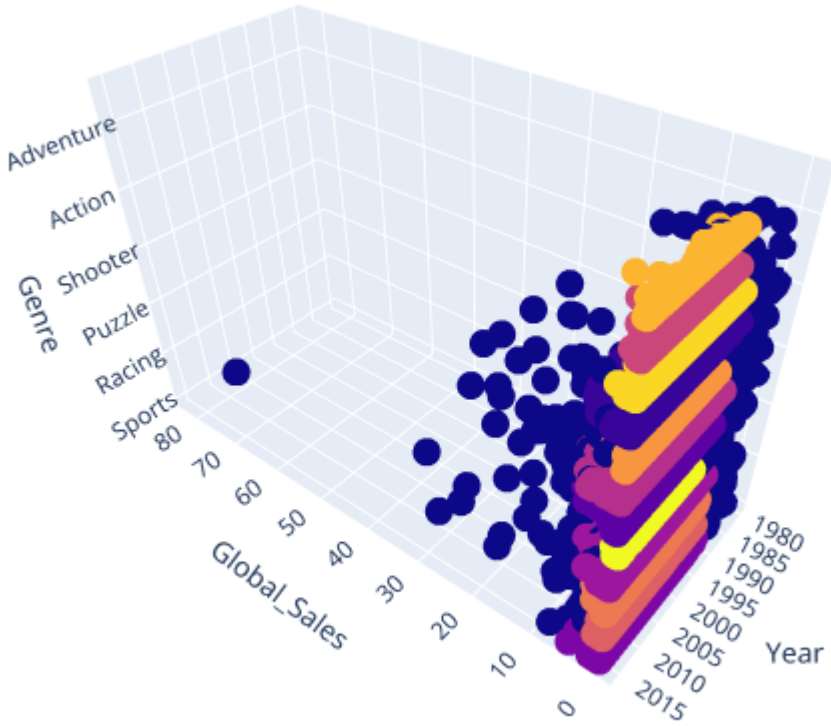
In the next step, we will use *the Gower Similarity method* to eliminate outliers from our clasters. First, we create a distance matrix.
Note. In our case we use Gower Dissimilarity.

```
df = df.sort_values(['Global_Sales'], ascending = False)
cols = ['NA_Sales','EU_Sales','JP_Sales','Other_Sales','Genre','Year']
distance_matrix = gower.gower_matrix(df[cols], cat_features = [False,False,False,False,True,False])
distance_matrix
```

```
array([[0.        , 0.6306776 , 0.45314816, ..., 0.72249186, 0.70870864,
        0.71796787],
       [0.6306776 , 0.        , 0.46900526, ..., 0.5520775 , 0.53829426,
        0.54755354],
       [0.45314816, 0.46900526, 0.        , ..., 0.4366627 , 0.25621277,
        0.4321387 ],
       ...,
       [0.72249186, 0.5520775 , 0.4366627 , ..., 0.        , 0.18077607,
        0.00485014],
       [0.70870864, 0.53829426, 0.25621277, ..., 0.18077607, 0.        ,
        0.17592593],
       [0.71796787, 0.54755354, 0.4321387 , ..., 0.00485014, 0.17592593,
        0.        ]], dtype=float32)
```

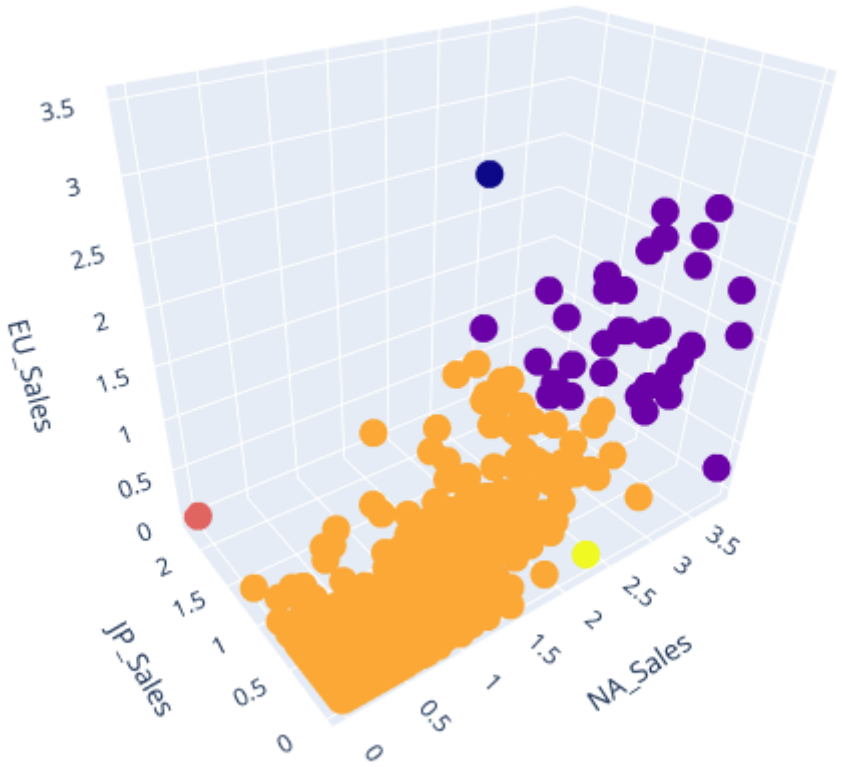
(distance matrix)

Then, from distance matrix, with epsilon = 0.3 (the maximum distance between two samples for one to be considered as in the neighborhood of the other) and n_0 = 325 (number of samples in a neighborhood for a point to be considered as a core point) we get following:



(DBSCAN clustering with Gower Similarity measure)

Note. We switched to ‘Global_Sales’, ‘Genre’ and ‘Year’ axes due to different task of this particular clustering i.e. finding connected groups with respect to categorical values.
We successively exclude outliers. Now we connect this to approaches together, that is, visualization by clusters. As example, we will take a look at observations from cluster 0.



(visualization of cluster 0 i.e. action games)

To summarize, we were able to split the data into certain groups by the average of the distances between all observations of pairs of each cluster which resulted in first clusterization. In the next step we splitted the data with respect to categorical value (genre) and year of game release (different scale). Lastly, we can offer a visualization of each target group separated with general clustering that shows success of certain games with respect to region.

PART B

In the second part of this task, data which will be clustered describe exam results of American students. Using a clustering algorithm, we need to identify groups of students who need social service and to find groups of students with above-average performance. For data clustering we use two algorithms: DBSCAN and K-means.

First, we have to process data so that the clustering algorithms can be easily applied. We start with changing all missing data values to values from the previous record. There is little data missing thus this process has almost no impact on the dataset. Then we change the type of data in ‘reading_score’ column to numeric and we create new columns with dummies for each column with qualitative data. Next step is data standardization and normalization. There are different ranges of data in this dataset thus we want to standardize and normalize them using StandardScaler and normalize methods from scikit learn package. ANd the last step of preprocessing data is dimensionality reduction using PCA (Principal component analysis). We reduce the number of dimensions to two.

We will describe applied clustering methods:

1. DBSCAN – this algorithm finds clusters based on the number of neighbors and the radius of its neighborhood. These two values are parameters of this method. If a record has the predefined number of neighbors in its neighborhood (which is a ball with the predefined radius), it is a core sample. Each cluster contains core samples which are in a dense area. These clusters are created as follows: a core sample is selected, then all its neighbors that are also core samples are added and so on. There are also records that are not core samples and we call them outliers which create a noise. They do not belong to any cluster. We implemented this method from scikit learn package.
2. K-means – this algorithm starts with choosing n centroids of the clusters, where n is the predefined number of clusters. They are being chosen randomly from the dataset. Then for each sample we calculate distances from the centroids of the clusters and assign the sample to the cluster with the smallest distance. Next, we recalculate centroids to be the center of the cluster in an average sense, so the new centroids are defined as the average of all samples in the cluster. Algorithm stops after a predefined number of iterations.

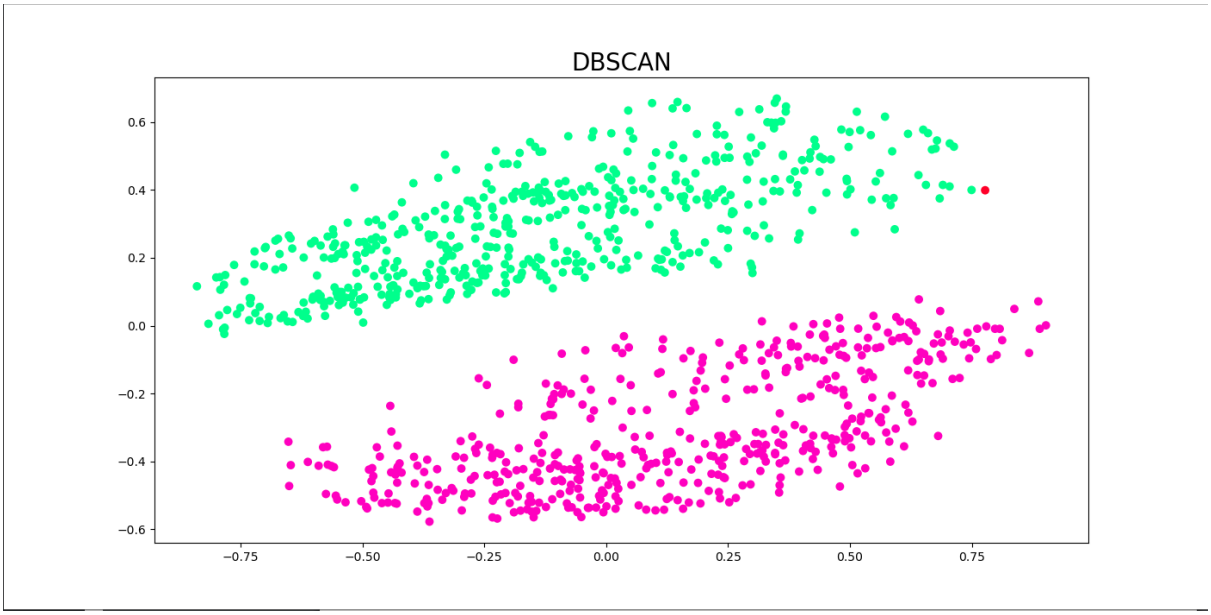
To measure the quality of the clusters we use two indexes: Davies-Bouldin index and Silhouette index. Both are imported from scikit learn package. Results of clustering are shown on the two-dimensional graph and for each cluster average scores and percentage of the specific group included in the cluster are shown in the table.

Comparison of DBSCAN method with different parameters:

epsilon	min_samples	metric	Silhouette index	Davies-Bouldin index
0.05	5	euclidean	-0.4248	4.2605
0.05	10	euclidean	-0.2812	5.3358
0.05	15	euclidean	-0.3772	4.2217
0.05	5	manhattan	-0.4260	4.5189
0.05	10	manhattan	-0.4337	6.3088

0.05	15	manhattan	-0.2185	2.3241
0.08	5	euclidean	0.0391	3.1715
0.08	10	euclidean	0.0826	3.1183
0.08	15	euclidean	-0.2639	4.3304
0.08	5	manhattan	-0.1227	3.6412
0.08	10	manhattan	-0.3503	5.6263
0.08	15	manhattan	-0.2675	3.6471
0.12	5	euclidean	0.0895	3.3011
0.12	10	euclidean	-0.0940	2.4033
0.12	15	euclidean	0.0740	2.2629
0.12	5	manhattan	-0.1303	2.5435
0.12	10	manhattan	0.0872	3.1867
0.12	15	manhattan	0.0818	2.6738

We can see that the 0.05 radius was too small to cluster data properly and we can conclude that the euclidean metric was better than the Manhattan one. The best choice of the above was the following: epsilon = 0.12, min_samples = 15, metric = ‘euclidean’. The graph for this clustering seting:



But when we look at the exam results of people clustered into 2 groups, we can see that they are similar. Thus, we want to cluster data using only the exam results data:

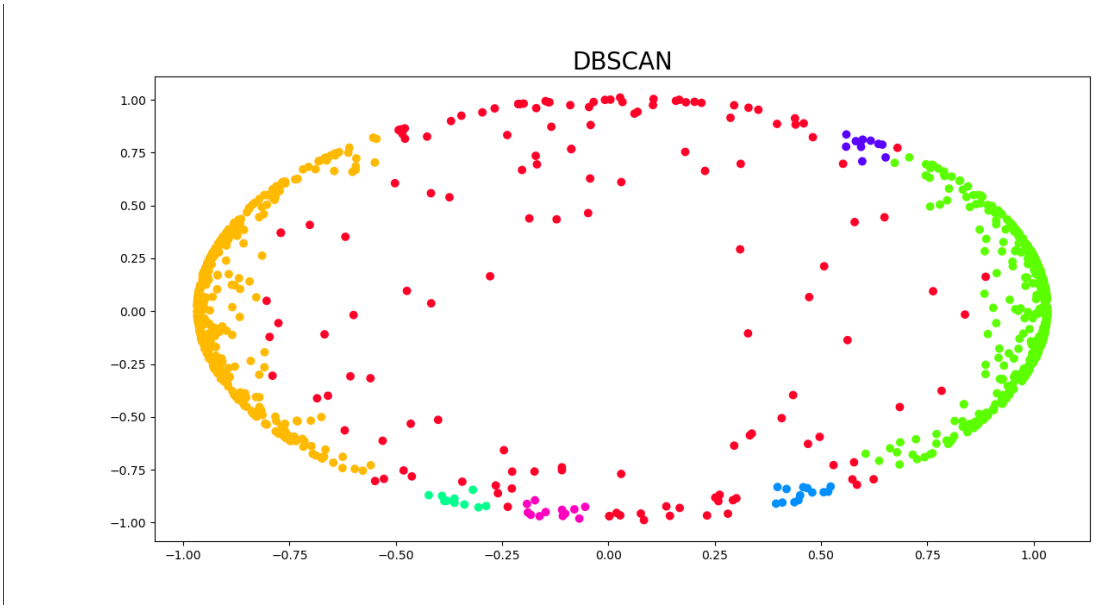
epsilon	min_samples	metric	Silhouette index	Davies-Bouldin index
0.05	5	euclidean	-0.0698	2.3318
0.05	10	euclidean	0.0706	1.0581
0.05	15	euclidean	-0.0189	1.8984
0.05	5	manhattan	-0.1337	1.7265
0.05	10	manhattan	-0.0283	2.0543
0.05	15	manhattan	0.1130	1.0583
0.08	5	euclidean	0.0491	1.4201
0.08	10	euclidean	0.0086	1.5668
0.08	15	euclidean	0.2649	0.9958
0.08	5	manhattan	-0.0192	2.1737
0.08	10	manhattan	-0.0133	1.1641
0.08	15	manhattan	0.1295	1.6572
0.12	5	euclidean	-0.2485	14.5097
0.12	10	euclidean	0.0684	1.1908
0.12	15	euclidean	0.0871	1.1314
0.12	5	manhattan	0.0181	1.5200
0.12	10	manhattan	0.0507	1.2049
0.12	15	manhattan	0.2500	1.0132

Now, we can see that two optimal options are the following:

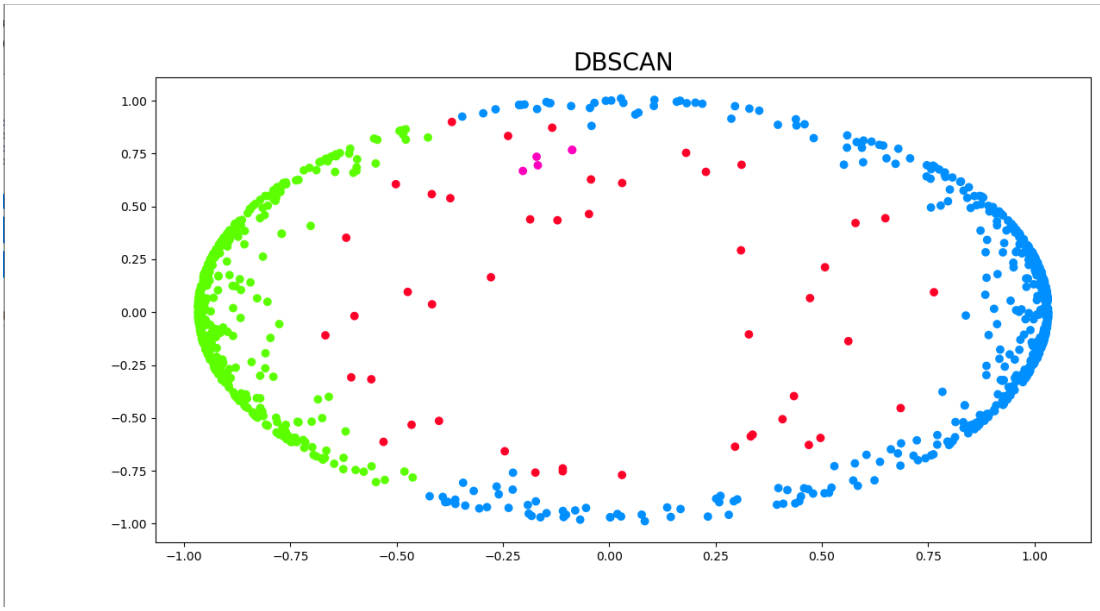
- epsilon = 0.08, min_samples = 10, metric = ‘euclidean’,
- epsilon = 0.12, min_smaples = 5, metric = ‘manhattan’

Let’s see the graph for these two clustering settings:

- For the first one:



- And for the second one:

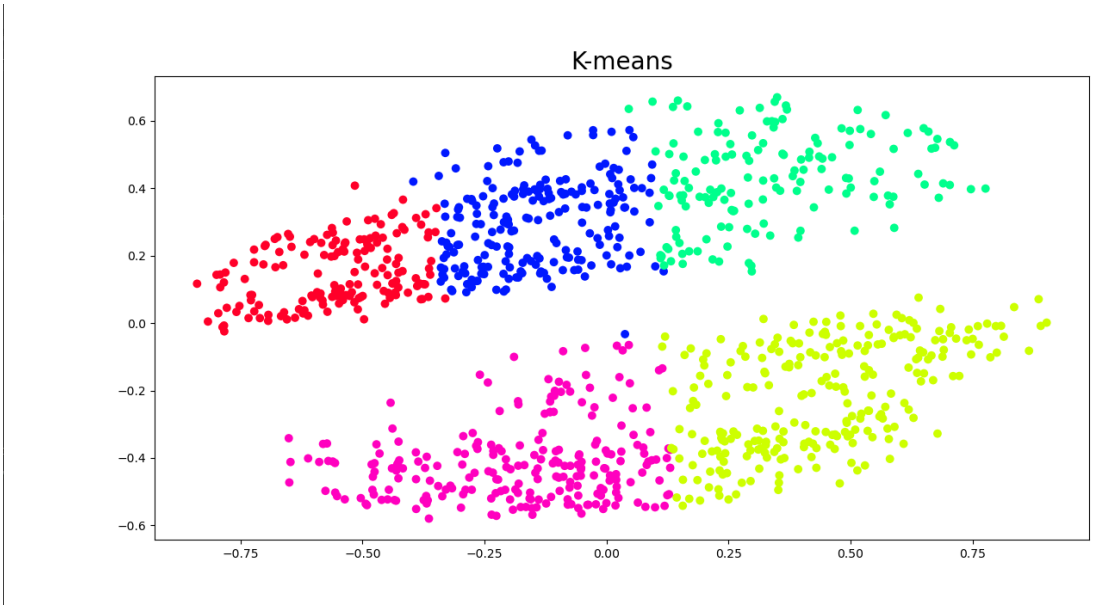


Now, let's look on the results of clustering using K-means method:

Number of clusters	Metric	Number of iterations	Silhouette index	Davies-Bouldin index
2	euclidean	100	0.0895	3.3011
2	euclidean	1000	0.0895	3.3011
2	manhattan	100	0.0899	3.2818
2	manhattan	1000	0.3635	0.8992
2	czebyszew	100	0.3236	0.9847

2	czebyszew	1000	0.1216	2.1387
3	euclidean	100	0.1572	2.3435
3	euclidean	1000	0.1403	2.2906
3	manhattan	100	0.1493	1.9607
3	manhattan	1000	0.1199	2.3862
3	czebyszew	100	0.1572	1.9214
3	czebyszew	1000	0.1559	1.9074
5	euclidean	100	0.0480	2.0419
5	euclidean	1000	0.0252	2.1828
5	manhattan	100	0.0395	2.1499
5	manhattan	1000	0.0128	2.3655
5	czebyszew	100	0.0543	2.0715
5	czebyszew	1000	0.0339	2.1487

Based on this results, it is clear that Manhattan metric is better in this case than euclidean metric and Czebyszew metric yields similar results as euclidean metric. The best results were obtained while there were 5 clusters predefined. The best clustering setting from the above for this method is the following: clusters_number = 5, metric = ‘czebyszew’, iterations = 1000. The graph for this setting:

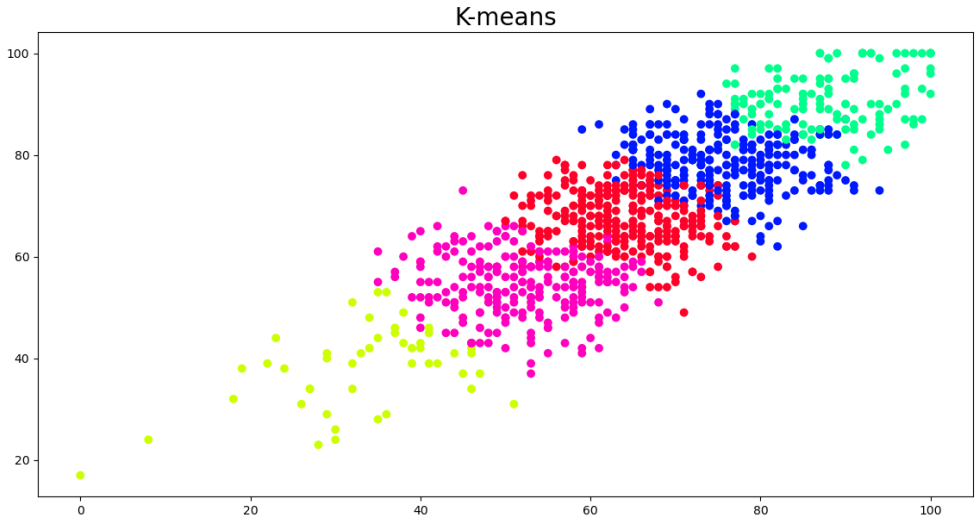


We will also check results of clustering using K-means method on the exam results data:

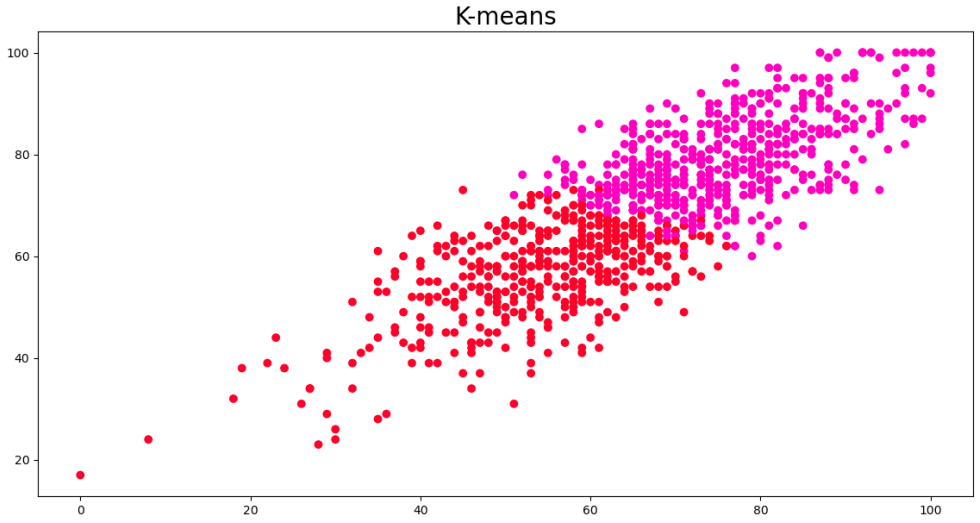
Number of clusters	Metric	Number of iterations	Silhouette index	Davies-Bouldin index
2	euclidean	100	0.4700	0.7495
2	euclidean	1000	0.4700	0.7495
2	manhattan	100	0.4700	0.7493
2	manhattan	1000	0.4700	0.7493
2	czebyszew	100	0.4643	0.7560
2	czebyszew	1000	0.4643	0.7560
3	euclidean	100	0.4007	0.8034
3	euclidean	1000	0.4007	0.8034
3	manhattan	100	0.3972	0.8066
3	manhattan	1000	0.3972	0.8066
3	czebyszew	100	0.3942	0.8096
3	czebyszew	1000	0.3942	0.8096
5	euclidean	100	0.3220	0.9464
5	euclidean	1000	0.3220	0.9464
5	manhattan	100	0.3232	0.9410
5	manhattan	1000	0.3148	0.9580
5	czebyszew	100	0.2795	1.0085
5	czebyszew	1000	0.3078	0.9744

It is clear that number of iterations which is over 100 has no impact or even no impact on the effectiveness of the method, so the algorithm converges quickly. We can also notice that manhattan metric is better than euclidean one, but czebyszew metric is the best one. Greater number of clusters means better clustering in sense of Silhouette index and worse clustering in sense of Davies-Bouldin index,, thus the best option from the above are the following:

- clusters_number = 5, metric = ‘czebyszew’, iterations = 1000. The graph for this option is:



- clusters_number = 2, metric = ‘manhattan’, iterations = 100. The graph for this option is:



To summarize, student who need social service are these with free/reduced lunch, parental level of education - high school and without test preparation, but student who should be granted are these with standard lunch and generally females with test preparation.