MINISTRY OF EDUCATION AND SCIENCE OF RUSSIA

SAINT PETERSBURG STATE

ELECTRICAL ENGINEERING UNIVERSITY

"LETI" NAMED AFTER V.I. ULYANOV (LENIN)

Department of Computer Engineering

REPORT

for laboratory work #2

in the discipline "Machine Learning"

Topic: Clustering

Student group 0308	___________________	Dashkin D.Sh.
Teacher	___________________	Alhasan A.

Saint Petersburg

2024

Purpose of the work.

Serves to acquire and consolidate skills in data preprocessing and application of machine learning methods to solve clustering problems.

Exercise.

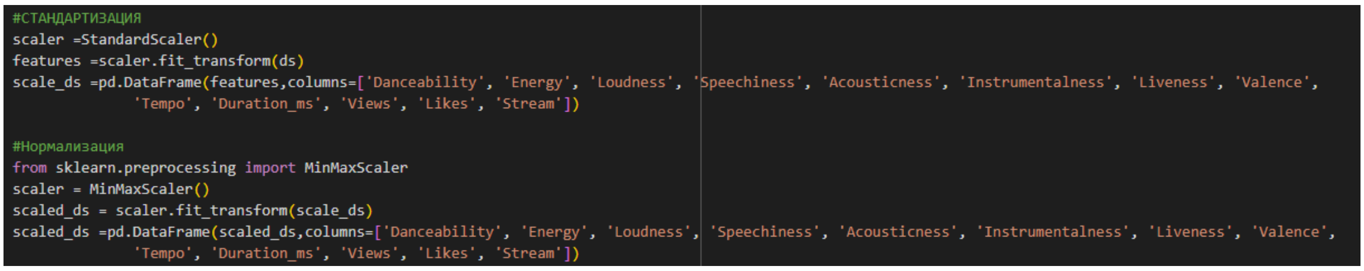
During the laboratory work the following steps must be completed:

1. Model training and parameter selection (where applicable):
    a. K-means method
    b. DBSCAN
    c. Hierarchical clustering
2. Evaluation of models
    a. Expert assessment
    b. Comparison of class division using clustering with real ones.
    c. Visualization of predicted values

Completing the work.

1. Preparing data for training.

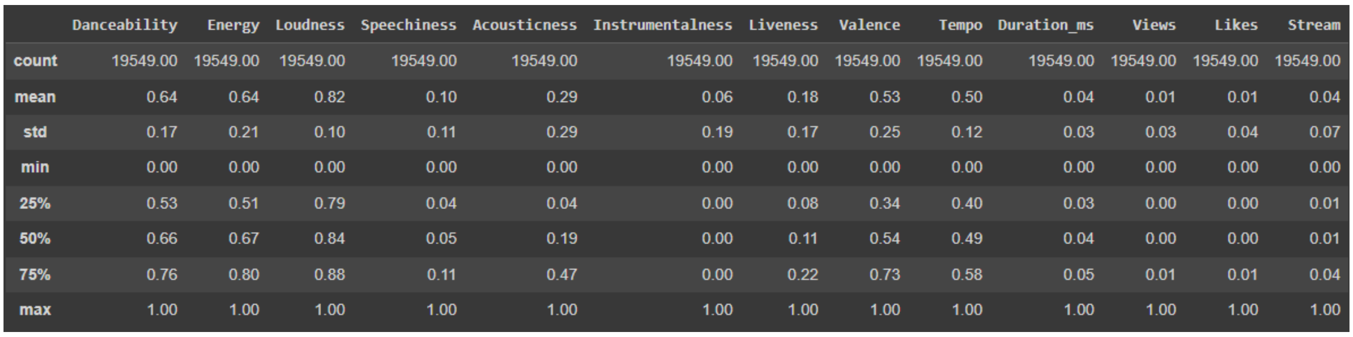
Before we begin, we need to standardize and normalize our data (Figure 1). The post-processed data is shown in Figure 2.



```
#СТАНДАРТИЗАЦИЯ
scaler =StandardScaler()
features =scaler.fit_transform(ds)
scale_ds =pd.DataFrame(features,columns=['Danceability', 'Energy', 'Loudness', 'Speechiness', 'Acousticness', 'Instrumentalness', 'Liveness', 'Valence',
              'Tempo', 'Duration_ms', 'Views', 'Likes', 'Stream'])

#Нормализация
from sklearn.preprocessing import MinMaxScaler
scaler = MinMaxScaler()
scaled_ds = scaler.fit_transform(scale_ds)
scaled_ds =pd.DataFrame(scaled_ds,columns=['Danceability', 'Energy', 'Loudness', 'Speechiness', 'Acousticness', 'Instrumentalness', 'Liveness', 'Valence',
              'Tempo', 'Duration_ms', 'Views', 'Likes', 'Stream'])
```

Figure 1 – Standardization and normalization.

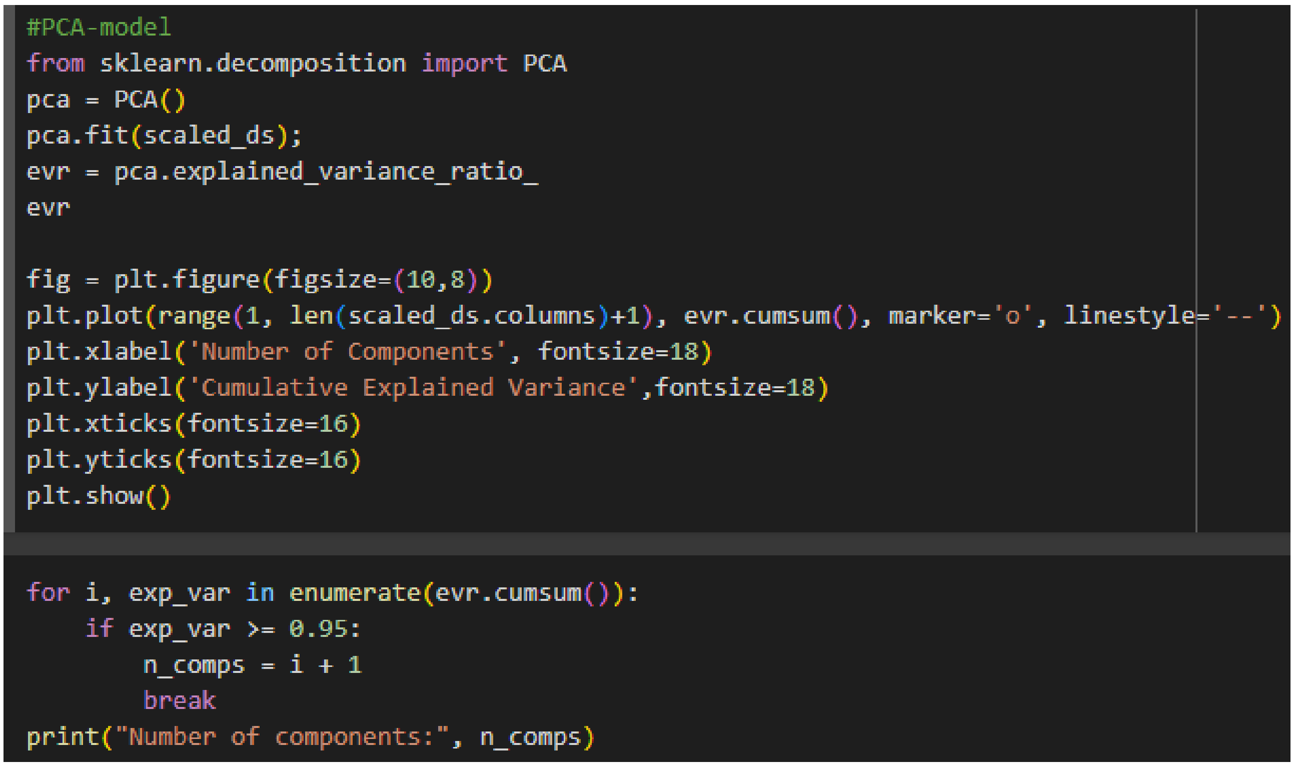


	Danceability	Energy	Loudness	Speechiness	Acousticness	Instrumentalness	Liveness	Valence	Tempo	Duration_ms	Views	Likes	Stream
count	19549.00	19549.00	19549.00	19549.00	19549.00	19549.00	19549.00	19549.00	19549.00	19549.00	19549.00	19549.00	19549.00
mean	0.64	0.64	0.82	0.10	0.29	0.06	0.18	0.53	0.50	0.04	0.01	0.01	0.04
std	0.17	0.21	0.10	0.11	0.29	0.19	0.17	0.25	0.12	0.03	0.03	0.04	0.07
min	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
25%	0.53	0.51	0.79	0.04	0.04	0.00	0.08	0.34	0.40	0.03	0.00	0.00	0.01
50%	0.66	0.67	0.84	0.05	0.19	0.00	0.11	0.54	0.49	0.04	0.00	0.00	0.01
75%	0.76	0.80	0.88	0.11	0.47	0.00	0.22	0.73	0.58	0.05	0.01	0.01	0.04
max	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00

Figure 2 – Data after processing.

2. PCA model

I decided to apply the PCA algorithm to reduce the number of parameters. A graph of the total accumulated variance versus the number of parameters was drawn. For the accuracy of further clustering, the number of parameters was chosen with a total accumulated variance value greater than 0.95. n_components = 8. The results are presented in Figures 3.1 and 3.2.

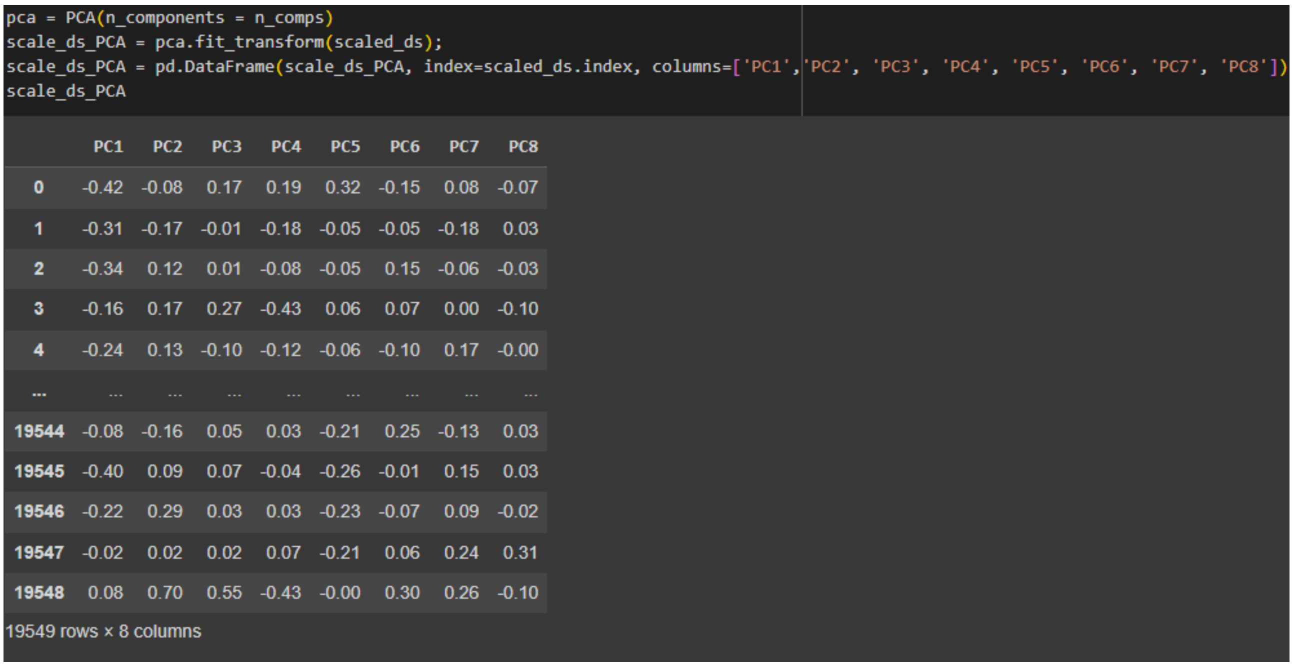


```python
#PCA-model
from sklearn.decomposition import PCA
pca = PCA()
pca.fit(scaled_ds);
evr = pca.explained_variance_ratio_
evr

fig = plt.figure(figsize=(10,8))
plt.plot(range(1, len(scaled_ds.columns)+1), evr.cumsum(), marker='o', linestyle='--')
plt.xlabel('Number of Components', fontsize=18)
plt.ylabel('Cumulative Explained Variance',fontsize=18)
plt.xticks(fontsize=16)
plt.yticks(fontsize=16)
plt.show()
```

```python
for i, exp_var in enumerate(evr.cumsum()):
    if exp_var >= 0.95:
        n_comps = i + 1
        break
print("Number of components:", n_comps)
```

Figure 3.1 – Selecting n_components



```python
pca = PCA(n_components = n_comps)
scale_ds_PCA = pca.fit_transform(scaled_ds);
scale_ds_PCA = pd.DataFrame(scale_ds_PCA, index=scaled_ds.index, columns=['PC1','PC2', 'PC3', 'PC4', 'PC5', 'PC6', 'PC7', 'PC8'])
scale_ds_PCA
```

	PC1	PC2	PC3	PC4	PC5	PC6	PC7	PC8
0	-0.42	-0.08	0.17	0.19	0.32	-0.15	0.08	-0.07
1	-0.31	-0.17	-0.01	-0.18	-0.05	-0.05	-0.18	0.03
2	-0.34	0.12	0.01	-0.08	-0.05	0.15	-0.06	-0.03
3	-0.16	0.17	0.27	-0.43	0.06	0.07	0.00	-0.10
4	-0.24	0.13	-0.10	-0.12	-0.06	-0.10	0.17	-0.00
...	...	...	...	...	...	...	...	...
19544	-0.08	-0.16	0.05	0.03	-0.21	0.25	-0.13	0.03
19545	-0.40	0.09	0.07	-0.04	-0.26	-0.01	0.15	0.03
19546	-0.22	0.29	0.03	0.03	-0.23	-0.07	0.09	-0.02
19547	-0.02	0.02	0.02	0.07	-0.21	0.06	0.24	0.31
19548	0.08	0.70	0.55	-0.43	-0.00	0.30	0.26	-0.10

19549 rows × 8 columns

Figure 3.2 – Application of PCA model

3. K-Means

Before applying the average algorithm, I decided to use the elbow method to determine the optimal number of clusters into which to divide our data. The result is shown in Figure 4.

```
#МЕТОД ЛОКТЯ ДЛЯ ОПРЕДЕЛЕНИЯ ОПТИМАЛЬНОГО K

wcss = {}
for i in range(1, 11):
    kmeans = KMeans(n_clusters = i, init = 'random', random_state = 44)
    kmeans.fit(scale_ds_PCA.values)
    wcss[i] = kmeans.inertia_

plt.plot(wcss.keys(), wcss.values(), 'gs-')
plt.xlabel("k")
plt.ylabel('WCSS')
plt.show()
```

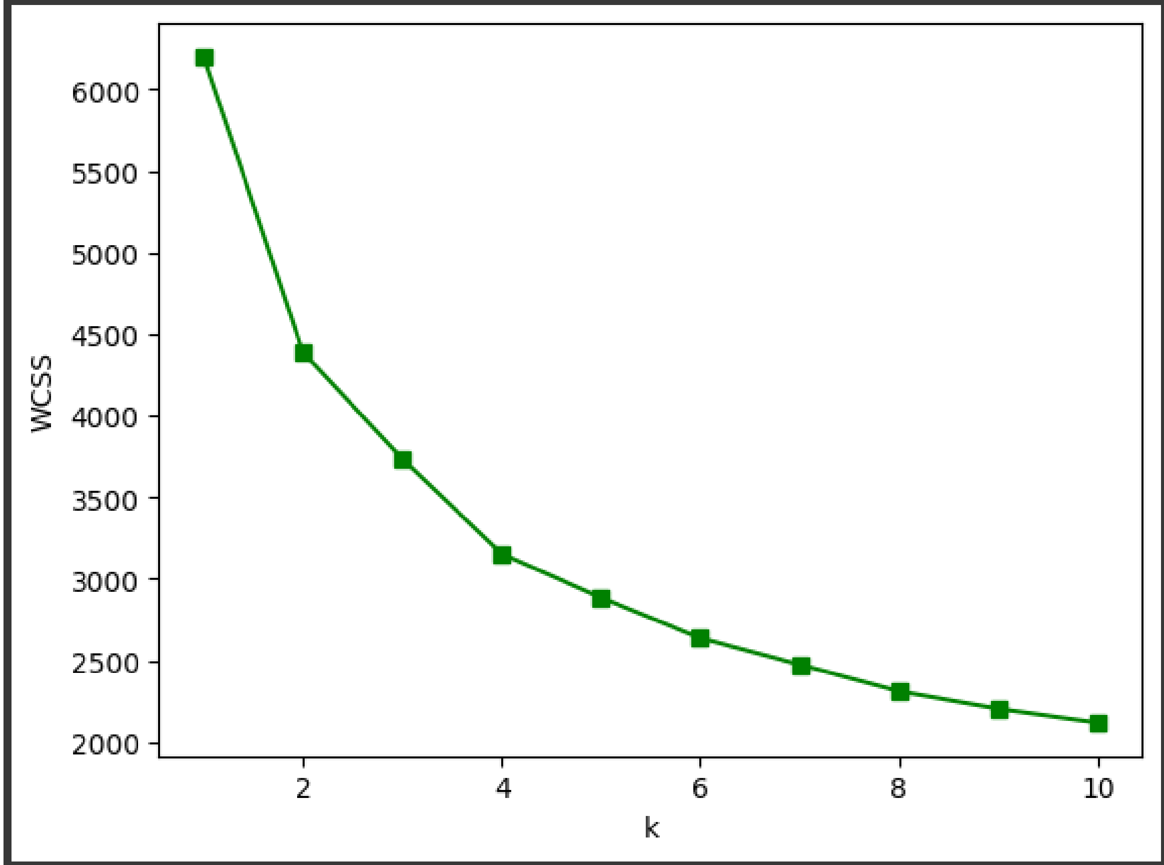
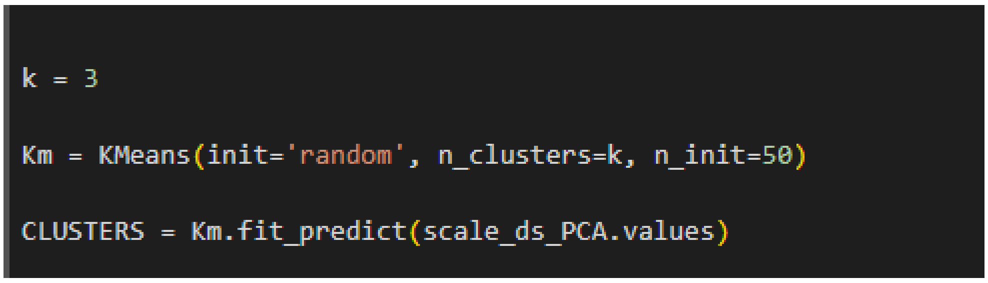


Figure 4 – Elbow Method

At the moment where the graph bends (like the bend of the elbow) – the optimal k clusters. K = 3.

Next we apply k-means. The code is shown in Figure 5.



```
k = 3

Km = KMeans(init='random', n_clusters=k, n_init=50)

CLUSTERS = Km.fit_predict(scale_ds_PCA.values)
```

Figure 5 – K-means

The results are presented in Figure 6.

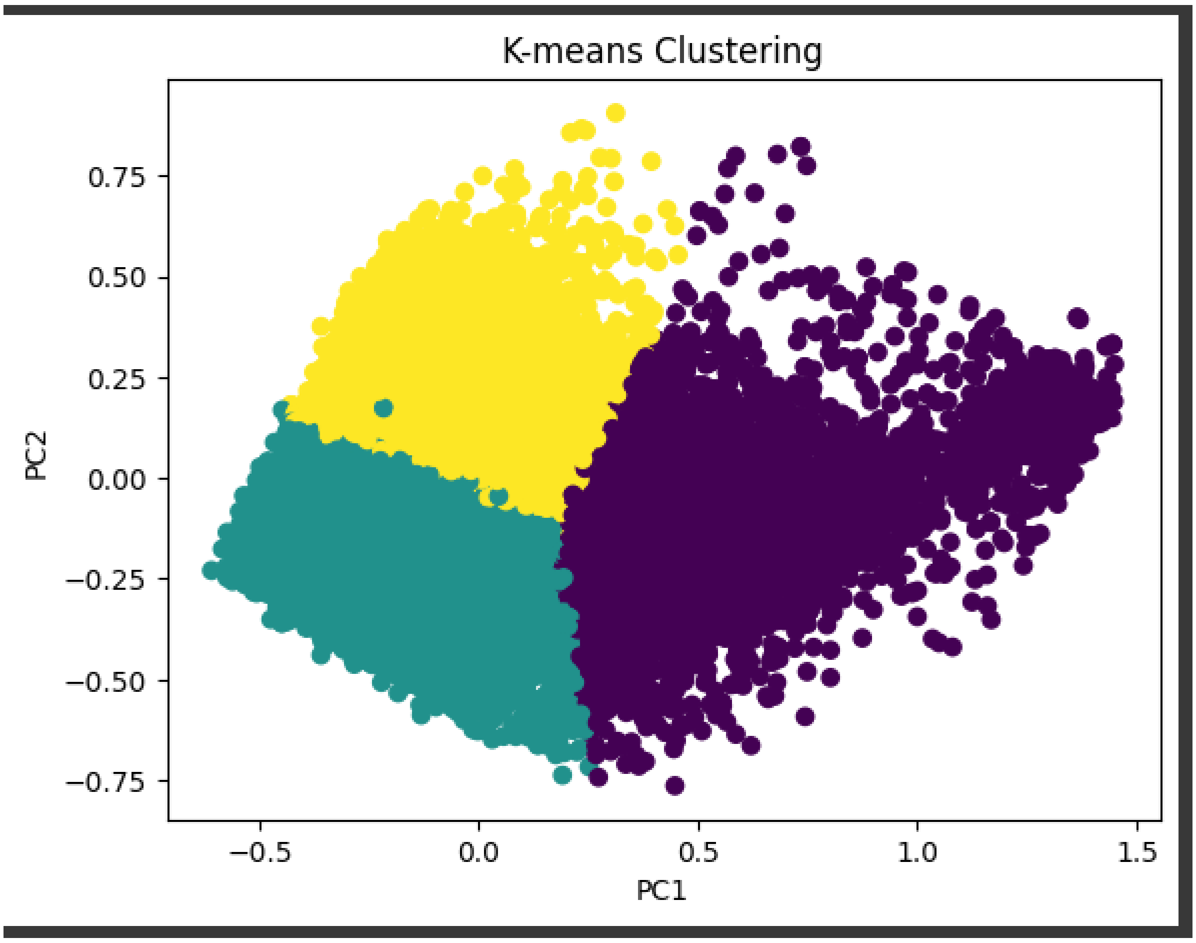
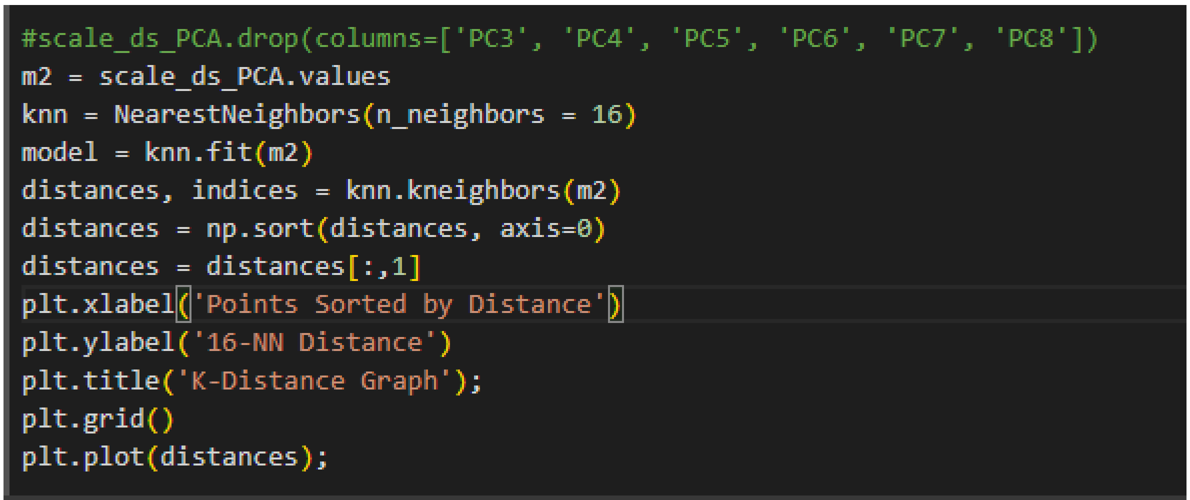


Figure 6 – K-means results

4. DBSCAN.

This method is density-based. Given a set of points in some space, the algorithm groups together points that are closely spaced (points with many close neighbors), while marking as outliers points that are alone in low-density areas (whose nearest neighbors are far away).

This method uses two important characteristics, such as**min_samples - the number of samples in the vicinity of a point to consider it as a principal point and eps - the maximum distance between two points to consider one as a neighbor of the other.**
For the algorithm to work correctly**min_samples >= 2*D, where D is the data dimension. That is, in our case, D = 8.**
In order to correctly determine these two parameters, the nearest neighbors method (k-nearest neighbors method) was used. This is an algorithm for automatic classification of objects or regression. In the case of using the method for classification, the object is assigned to the class that is the most common among the k neighbors of this element, the classes of which are already known.
Main parameter –**n_neighbors, which by rules is equal to min_samples -1.**
Thanks to this algorithm we will be able to find**eps parameter for our DBSCAN algorithm. The result is shown in Figure 7.**



```
#scale_ds_PCA.drop(columns=['PC3', 'PC4', 'PC5', 'PC6', 'PC7', 'PC8'])
m2 = scale_ds_PCA.values
knn = NearestNeighbors(n_neighbors = 16)
model = knn.fit(m2)
distances, indices = knn.kneighbors(m2)
distances = np.sort(distances, axis=0)
distances = distances[:,1]
plt.xlabel('Points Sorted by Distance')
plt.ylabel('16-NN Distance')
plt.title('K-Distance Graph');
plt.grid()
plt.plot(distances);
```

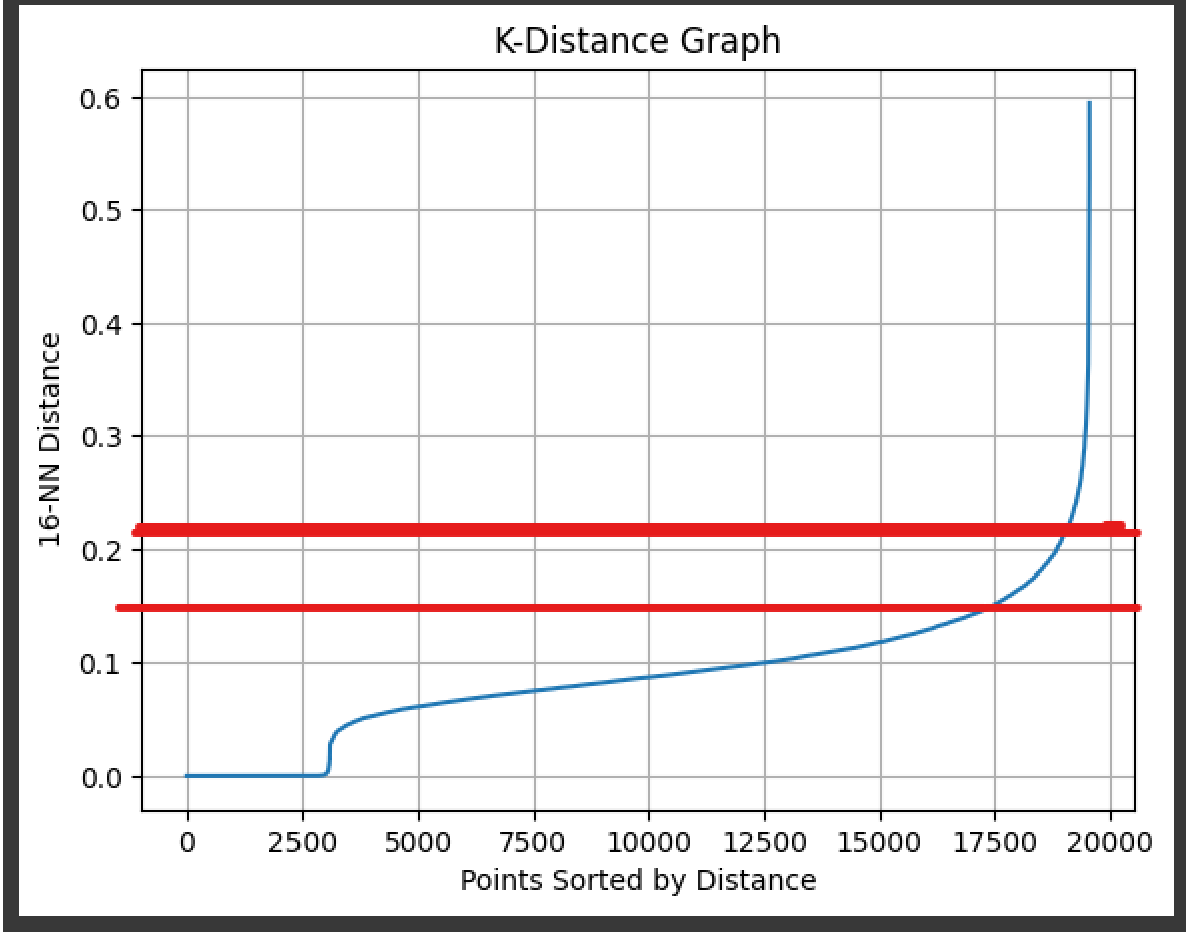
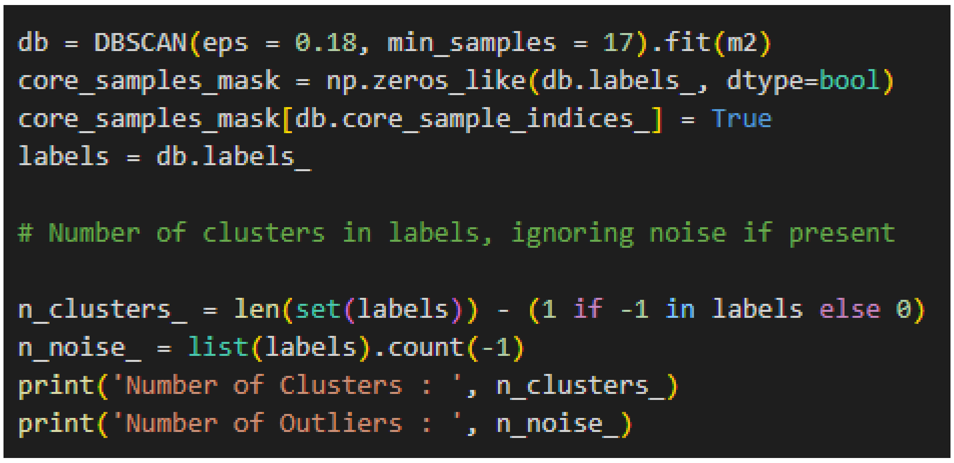


Figure 7 – k-nearest neighbors method

The parameter should be taken from the area where the "fold" begins. Let's choose eps = 0.18 and apply our DBSCAN (Figure 8).



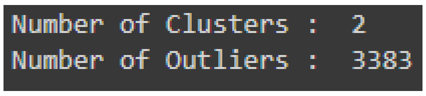
```
db = DBSCAN(eps = 0.18, min_samples = 17).fit(m2)
core_samples_mask = np.zeros_like(db.labels_, dtype=bool)
core_samples_mask[db.core_sample_indices_] = True
labels = db.labels_

# Number of clusters in labels, ignoring noise if present

n_clusters_ = len(set(labels)) - (1 if -1 in labels else 0)
n_noise_ = list(labels).count(-1)
print('Number of Clusters : ', n_clusters_)
print('Number of Outliers : ', n_noise_)
```

Figure 8 – DBSCAN

Our algorithm divided the data into two clusters with 3383 outliers, i.e. points that did not fall into either cluster (Figures 9-10).



```
Number of Clusters :  2
Number of Outliers :  3383
```

Figure 9 – DBSCAN Result

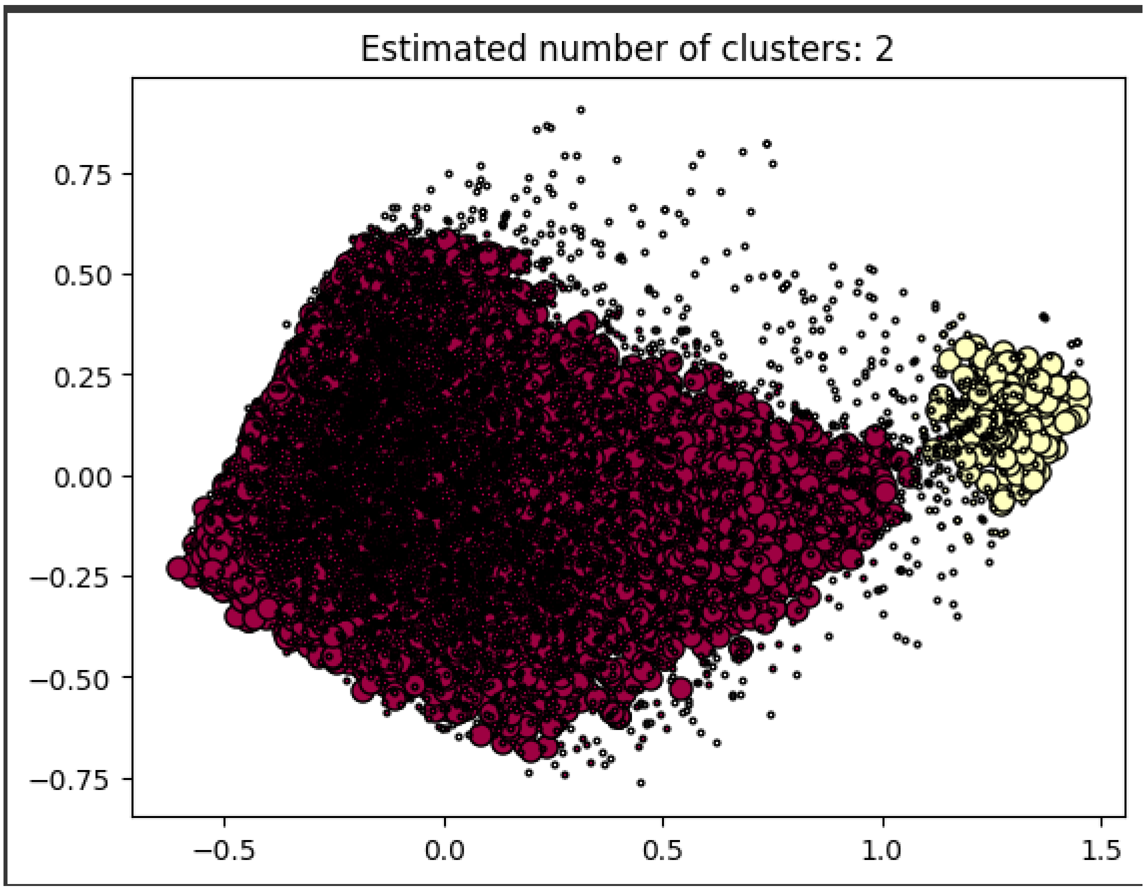


Figure 10 – DBSCAN visualization

5. Hierarchy.

A hierarchical data model is a data model that uses a representation of the database as a tree structure consisting of objects of different levels. There are relationships between objects, each object can include several lower-level objects.

To begin with, I built a dendrogram (Figure 11).

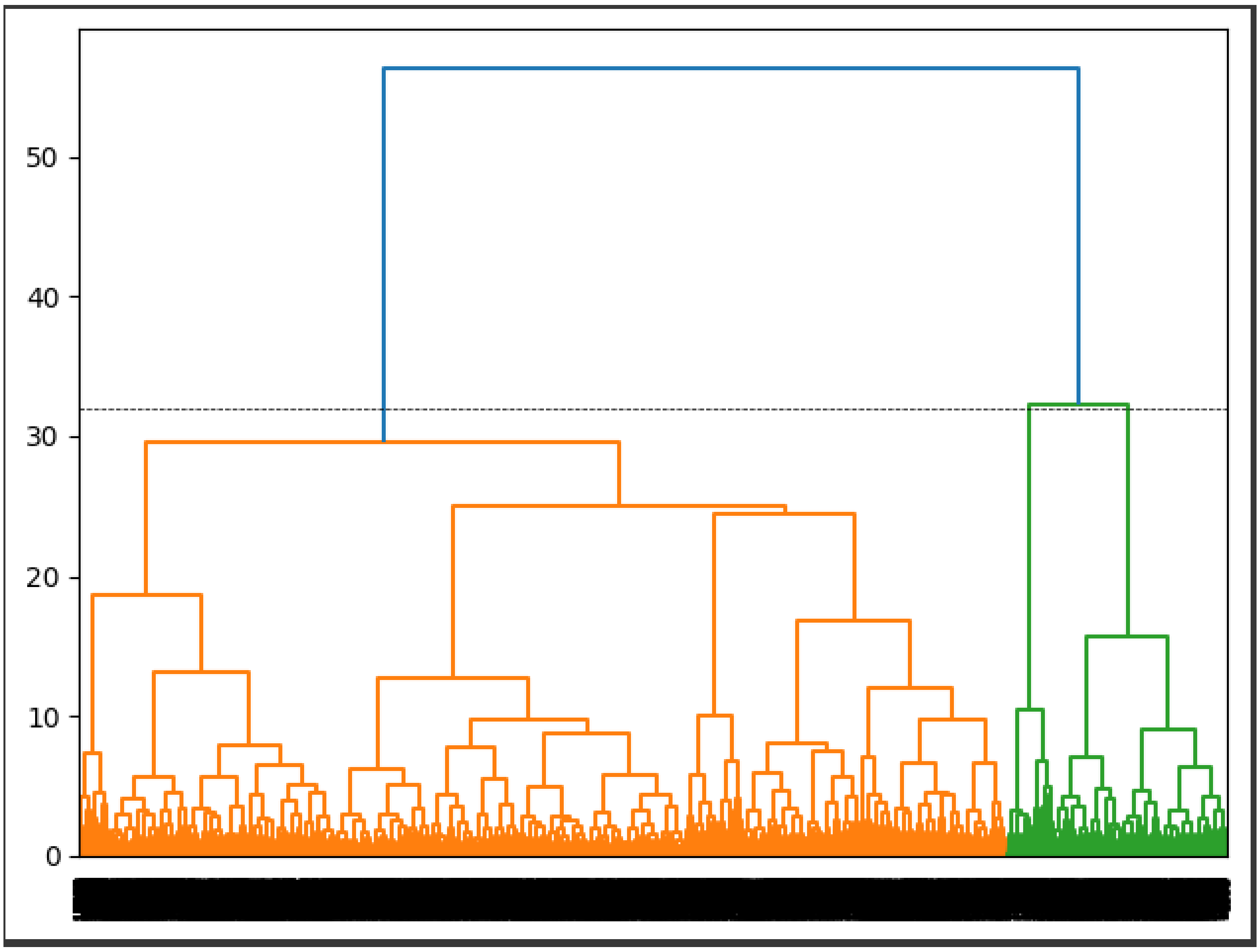
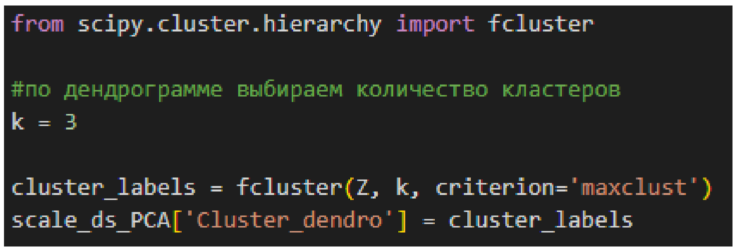


Figure 11 – Dendrogram

It can be seen that the three branches at the top level are combined into one, and therefore the selected number of clusters is 3.

Let's apply our hierarchical clustering algorithm and look at the results (Figures 12-13).



```python
from scipy.cluster.hierarchy import fcluster

#по дендрограмме выбираем количество кластеров
k = 3

cluster_labels = fcluster(Z, k, criterion='maxclust')
scale_ds_PCA['Cluster_dendro'] = cluster_labels
```

Figure 12 – Application of hierarchical clustering

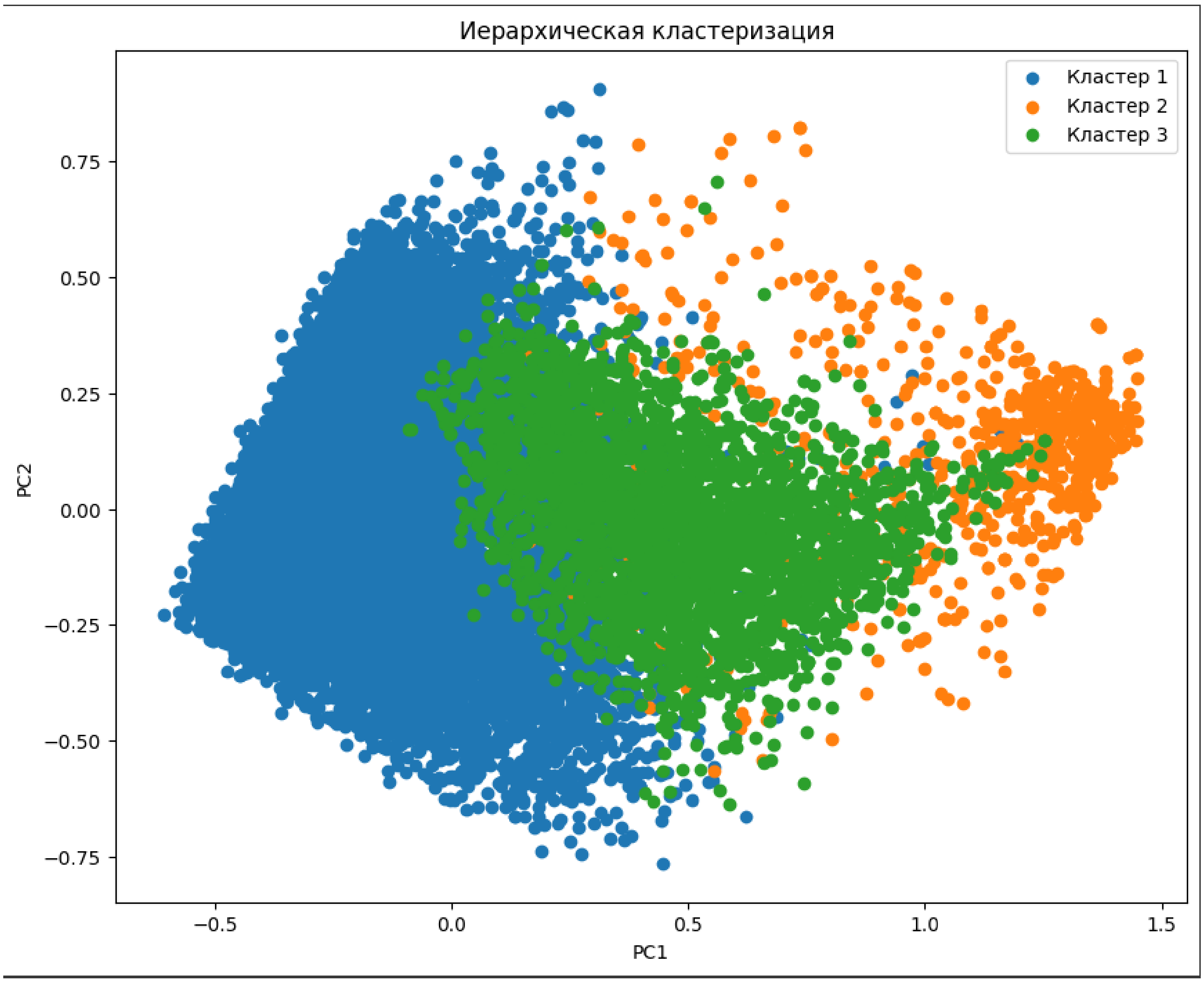
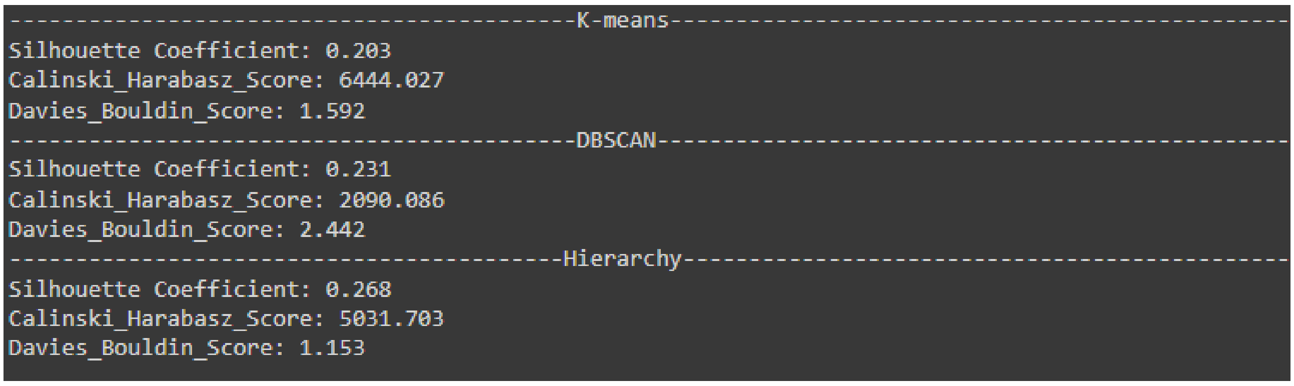


Figure 13 – Results of hierarchical clustering

6. Evaluation of clustering performance

Evaluating the performance of an algorithm is not as trivial as calculating the number of errors or the accuracy and recall of a supervised classification algorithm. If the underlying true labels are unknown, the evaluation must be performed using the model itself. Since in our work we do not know these labels, we can limit ourselves to only a few methods of clustering evaluation, namely the “silhouette” label, the variance ratio criterion, and the Davis-Bouldin index. The first two criteria are evaluated according to the principle: the higher the better the algorithm, while the latter index is the opposite: a lower Davis-Bouldin index refers to a model with better separation between clusters. The results of these metrics are presented in Figure 14.



```
------------------------------------------K-means-------------------------------------------
Silhouette Coefficient: 0.203
Calinski_Harabasz_Score: 6444.027
Davies_Bouldin_Score: 1.592
------------------------------------------DBSCAN--------------------------------------------
Silhouette Coefficient: 0.231
Calinski_Harabasz_Score: 2090.086
Davies_Bouldin_Score: 2.442
-----------------------------------------Hierarchy------------------------------------------
Silhouette Coefficient: 0.268
Calinski_Harabasz_Score: 5031.703
Davies_Bouldin_Score: 1.153
```

Figure 14 – Evaluation of clustering methods

Conclusion:

I'm byacquired and consolidated skills in data preprocessing and application of machine learning methods to solve clustering problems.

Link to code