CS 464 Introduction to Machine Learning
Term Project Final Report

Recommendify

Track Recommendation System for Spotify Playlists

Group 23
Arda Barış Örtlek 21903472
Ayşın Tümay 21902058
Bahadır Aydoğan 21903064
Barış Tiftik 21702427

Department of Computer Science
Bilkent University
28.12.2022

1. Introduction

In this project, we aimed to code a recommendation system for Spotify playlists. The goal is to recommend 10 songs based on the technical properties of each track in the playlist using unsupervised learning algorithms. Since the playlists contain a set of songs within a similar fashion, the song which will be recommended will also have compatible features. Content-based approach was utilized. For this purpose, the tracks' features in the playlists are taken from Spotify Web API. Then, the average of these features forms a single vector close to the desired track. On the other hand, the unsupervised learning methods such as k-Means and DBSCAN perform clustering. The trained models give recommendations based on those clusters and the vectorized playlist. In this way, a matching track from a track pool will be selected in line with the feature vector. In order to evaluate the model, we calculate the Silhouette Score, Mean Square Error (MSE) and R2 Score. For this project we tried three different inputs by changing data reduction methods and two different clustering algorithms. The evaluation scores of those inputs and algorithms will be discussed in the following parts.

2. Problem Description

The problem we tackled is the song recommendation for spotify playlists. As we all witnessed in our daily life as Spotify users, Spotify recommends songs which are not necessarily related to our taste of music. The reason behind making irrelevant recommendations is using the features which do not really matter for us. In addition to the excess feature usage, its clustering might be majorly affected by the genre of the tracks which generally ignores some of the features like danceability, speachiness, and time signature. By considering these issues, Recommendify makes recommendations according to the features which really do matter for us and cluster the tracks by considering all of those features.

We researched and tried to tackle some main problems such as data reduction methods and their degree of effect on clustering algorithms, the effect of hyperparameters on clustering, evaluation metrics to test the validity of recommendations. While we were answering those questions, in this project we learned how to handle an unsupervised learning algorithm and the ways of evaluating it.

3. Methods

In this section, the unsupervised learning algorithms used in this project will be elaborated. Moreover, the different types of inputs used in experiments will be demonstrated.

3.1. PCA

After the required data is obtained, PCA is applied to compress the data and reduce the noise. By using PCA, the dimension of the data is reduced from 26 features to 15 principal components. To decide the number of principal components (PCs), PVE vs number of PCs plot is sketched as in Figure 1 and for each number of PCs the PVE values are displayed as in Figure 2. To apply the same reduction, the parameters, PCs are conserved for test preprocessing operation. The number of 15 is chosen with a PVE of 0.92.

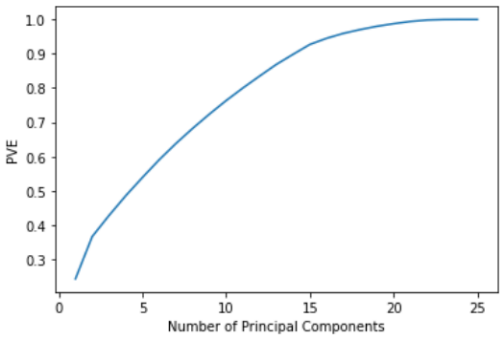


Fig.1: PVE vs Number of PCs plot

```
Number of components:1 -> PVE:0.24326171448094214
Number of components:2 -> PVE:0.3664227740584666
Number of components:3 -> PVE:0.42772897578057034
Number of components:4 -> PVE:0.4854238122029085
Number of components:5 -> PVE:0.5389116000133238
Number of components:6 -> PVE:0.590554006223087
Number of components:7 -> PVE:0.6381069099081874
Number of components:8 -> PVE:0.6822476624247833
Number of components:9 -> PVE:0.723848920590870 6
Number of components:10 -> PVE:0.7634003126814057
Number of components:11 -> PVE:0.8001213751094393
Number of components:12 -> PVE:0.8349905522028946
Number of components:13 -> PVE:0.8689735733525865
Number of components:14 -> PVE:0.8984491065831978
Number of components:15 -> PVE:0.9271934253856331
Number of components:16 -> PVE:0.9451263591921633
Number of components:17 -> PVE:0.9594644380247298
Number of components:18 -> PVE:0.9704866018618876
Number of components:19 -> PVE:0.9800929895571013
Number of components:20 -> PVE:0.9877245697981174
Number of components:21 -> PVE:0.9942298222534641
Number of components:22 -> PVE:0.9983439335499465
Number of components:23 -> PVE:0.9997655193001043
Number of components:24 -> PVE:1.0
Number of components:25 -> PVE:1.0
```

Fig.2: Number of PCs with PVE values

3.2. Autoencoder

Autoencoders are a type of artificial neural network used for dimensionality reduction and feature learning. They are composed of an encoder and a decoder, which are trained to reconstruct an input data sample from a lower-dimensional representation, known as the latent space or bottleneck. The encoder processes the input data and maps it to a lower-dimensional representation, which is then used by the decoder to reconstruct the original input. The encoding process is typically done by learning a set of weights that are used to transform the input data into a lower-dimensional space. This transformation is often achieved using a series of fully connected layers, with the number of units in the bottleneck layer being smaller than the input data. The goal of training an autoencoder is to learn a compressed representation of the input data that captures the most important features, while still allowing the decoder to accurately reconstruct the original input.

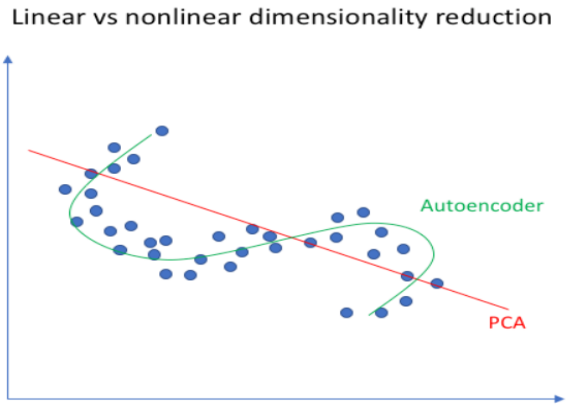


Fig.3: Autoencoder vs PCA [1]

```
Layer (type)                 Output Shape              Param #
=================================================================
 input_2 (InputLayer)        [(None, 26)]              0

 dense_6 (Dense)             (None, 52)                1404

 batch_normalization_4 (Batc  (None, 52)               208
 hNormalization)

 leaky_re_lu_4 (LeakyReLU)   (None, 52)                0

 dense_7 (Dense)             (None, 26)                1378

 batch_normalization_5 (Batc  (None, 26)               104
 hNormalization)

 leaky_re_lu_5 (LeakyReLU)   (None, 26)                0

 dense_8 (Dense)             (None, 15)                405

 dense_9 (Dense)             (None, 26)                416

 batch_normalization_6 (Batc  (None, 26)               104
 hNormalization)

 leaky_re_lu_6 (LeakyReLU)   (None, 26)                0

 dense_10 (Dense)            (None, 52)                1404

 batch_normalization_7 (Batc  (None, 52)               208
 hNormalization)

 leaky_re_lu_7 (LeakyReLU)   (None, 52)                0

 dense_11 (Dense)            (None, 26)                1378

=================================================================
Total params: 7,009
Trainable params: 6,697
Non-trainable params: 312
```

Fig. 4: Model Summary

2 hidden layers have been used before and after the bottleneck layer. Leaky ReLU has been selected as the non-linear activation function for hidden layers. Batch normalization has been applied to improve the stability of the network. MSE has been chosen for the loss function and Adam optimizer has been utilized. In order to tune the network, a set of different learning rates have been tried and the optimal one was found as 0.0063.

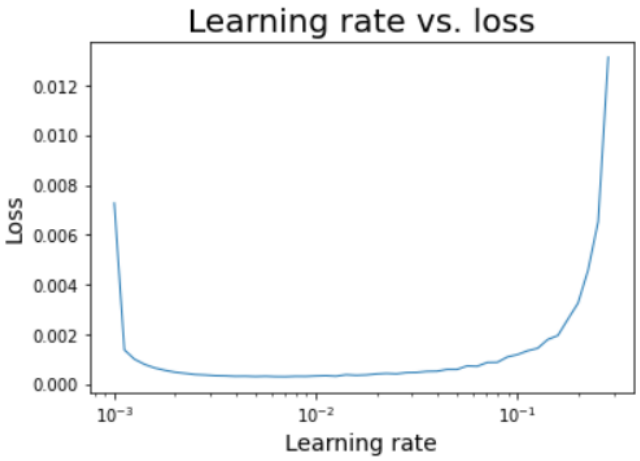


Fig.5: Learning rate vs loss

The network has been trained for 50 epochs with mini batch size 64.

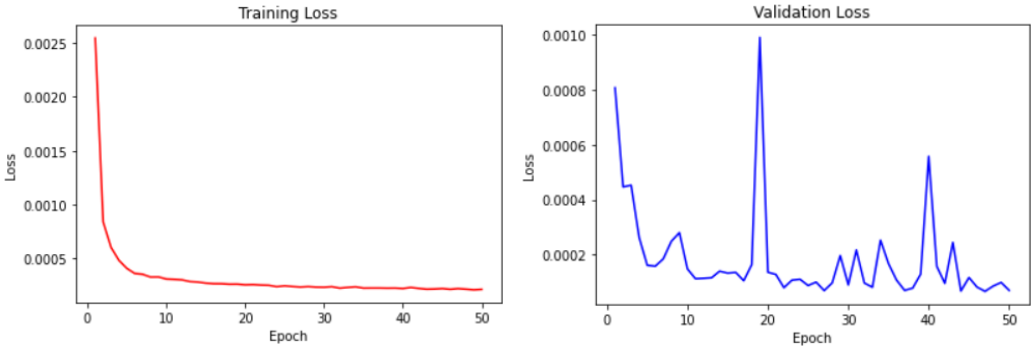


Fig. 6-7: Training and Validation loss over 50 epochs

3.3. k-Means

K-means is an unsupervised machine learning algorithm used for clustering. It is an iterative algorithm that divides a group of n data points into k clusters, where k is a predefined number specified by the user. The algorithm works by randomly selecting k data points as initial cluster centroids, and then iteratively assigning each data point to the cluster whose centroid is closest, based on some distance measure. The centroids are then updated to the mean of the data points assigned to each cluster. This process is repeated until the centroids stabilize or a predefined number of iterations is reached.

3.4. DBSCAN

DBCAN is a density based clustering algorithm with noise. It works efficiently for data with similar densities. It finds high density samples and generates clusters according to these samples. Hyperparameters that are tuned are epsilon and min_samples. Epsilon is the maximum euclidean distance for 2 samples to be considered as neighbors. Min_samples is the minimum number of samples in the neighbourhood to be considered as the core point. Below figure shows these terms visually. It is common sense to choose min_samples as twice the feature size [2].To determine the optimal epsilon, a search algorithm based on k-NN must be applied. The application difference between k-Means and DBSCAN is that DBSCAN cannot determine the exact number of clusters. It indirectly adjusts this parameter using epsilon and min_samples. In the below figure, the difference between k-Means and DBSCAN can be monitored.

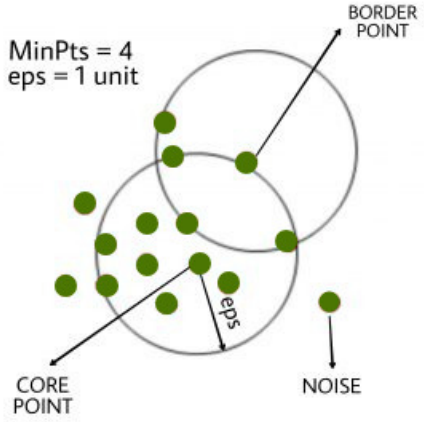


Fig.8 : DBSCAN working principle.

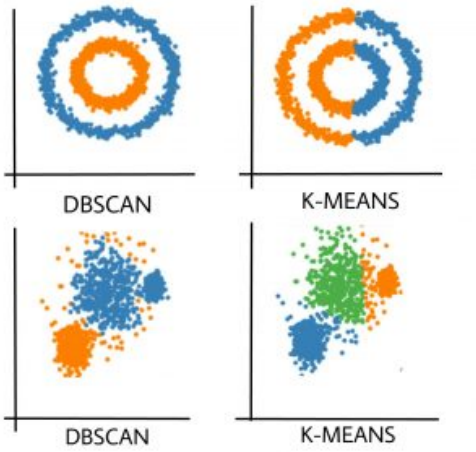


Fig.9: Comparison between k-Means and DBSCAN [3].

3.5 Evaluation Metrics

To evaluate the performance of clustering, Silhouette score is utilized. It uses the mean distance of the observation point with all other points in the same cluster (mean intra-cluster distance). Also, it uses the mean distance of the observation point with all other points in the cluster nearest (mean nearest-cluster distance). The score is between -1 and 1. Scores close to 1 is considered good while negatif scores means adverse clustering. If the score is close to 0, the distance between the clusters is negligible [4].

To evaluate the recommendation performance, MSE and r2 score is used since the problem will be reduced down to euclidean distance between 2 vectors by averaging test playlists for convenience. Basically, the distance between the recommendation and the average of the test playlist will be compared.

4. Results

In this section, the detailed results of each method with different input types will be provided.

4.1. k-Means

For 3 different cases, the model has been trained for cluster numbers which are in the range of [50,1000] with 50 step size. Inertia and Silhouette score have been calculated.

4.1.1 k-Means with No Feature Reduction

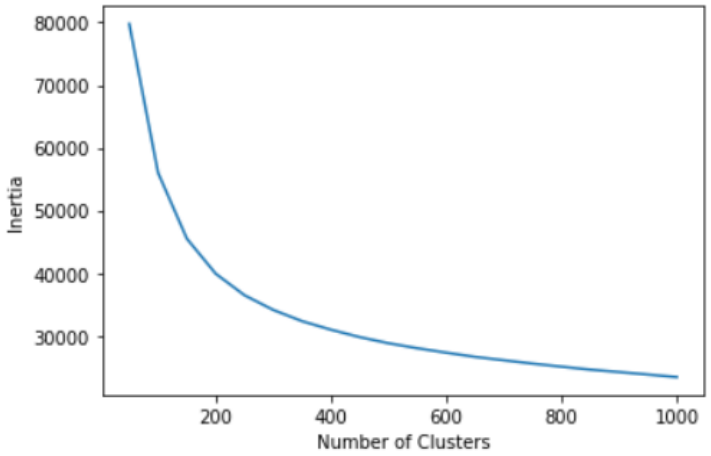


Fig. 10: Inertia vs number of clusters

Elbow method has been applied to the inertia graph in order to find optimal cluster number. The optimal one was 250.

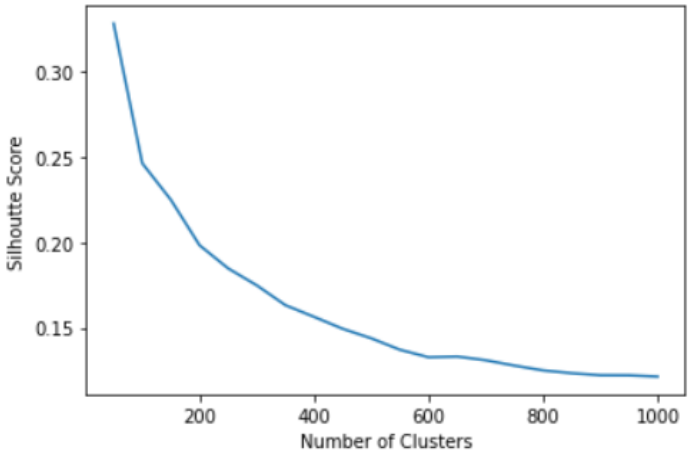


Fig. 11: Silhouette score vs number of clusters

	artist_name	track_name
54470	Chorus	Vishambhari Stuti
58593	Katherine Jenkins	Amazing Grace
102496	Radiohead	Desert Island Disk
73695	Kimbo Children's Music	Whirling Strawberries (Bright 3/4 Meter)
11467	Saba	FIGHTER
125225	Granville Bantock	Violin Sonata No.3 in C major
10711	Grizfolk	Interlude
231153	Kelela	Bluff
66342	Amigo the Devil	The Dreamer
230930	James Morrison	Just Like A Child

Fig. 12: Recommended songs for test_1

```
Mean squared error of a randomly selected recommended song and test_1
                    playlist: 0.10762547161809757

R2 score of a randomly selected recommended song and test_1 playlist:
                    -0.01328024830237684
```

4.1.2. k-Means with PCA

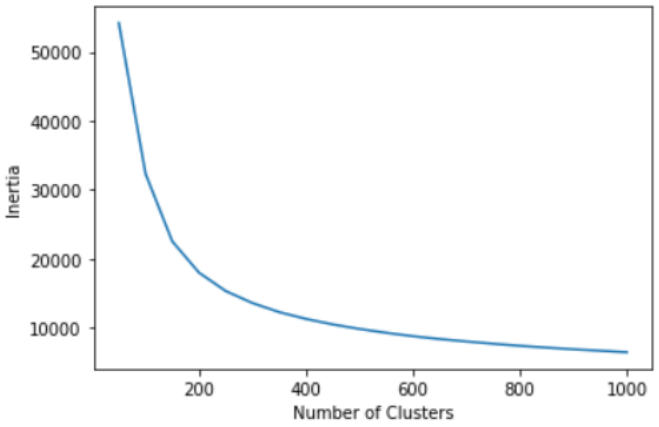


Fig. 13: Inertia vs number of clusters

Elbow method has been applied to the inertia graph in order to find optimal cluster number. The optimal one was 250.

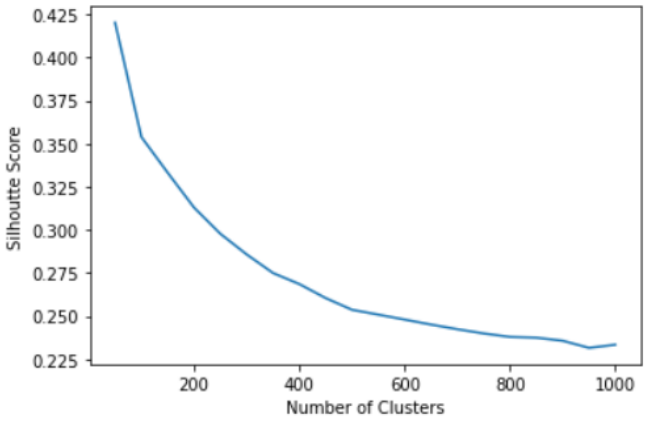


Fig. 14: Silhouette score vs number of clusters

The score was better than the case where any feature reduction operation has not been applied.

	artist_name	track_name
142391	Brent Faiyaz	Burn One (Interlude)
116183	Mac Miller	We (feat. CeeLo Green)
139602	Lenny Tavárez	Caviar (Remix) [feat. Kevin Roldan & Darell]
232126	Surfaces	Loving (Acoustic)
232112	Dwele	I'm Cheatin'
147269	Benny Sings	Everything I Know
109123	blackbear	Weak When Ur Around
156241	Belle & Sebastian	Piazza, New York Catcher
94996	Mac Ayres	Under
194825	Martin & les fées	La petite flamme (par Lorie Pester)

Fig. 15: Recommended songs for test_1

```
Mean squared error of a randomly selected recommended song and test_1
playlist: 0.037304920639516646

R2 score of a randomly selected recommended song and test_1 playlist:
0.5242376057642408
```

4.1.3. k-Means with Autoencoder

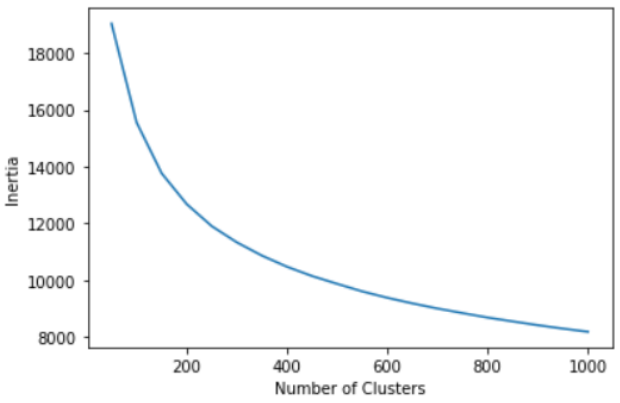


Fig. 16: Inertia vs number of clusters

Elbow method has been applied to the inertia graph in order to find optimal cluster number. The optimal one was 250.

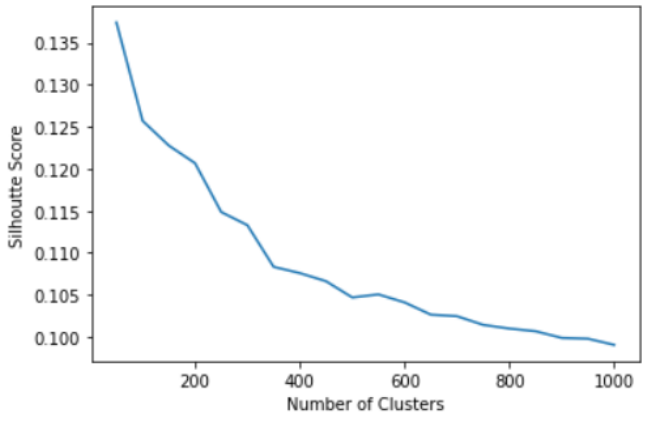


Fig. 17: Silhouette score vs number of clusters

Autoencoder model did not perform as expected. The score was worse than the PCA case.

	artist_name	track_name
124457	Gustavo Santaolalla	The Last of Us (You and Me)
232165	Sarah Vaughan	Make Yourself Comfortable
198998	Christophe Beck	Thin Air - Score Demo
81477	Giacomo Puccini	Tosca, SC 69, Act I: Ah! Finalmente! (Live)
128660	Jacques Offenbach	Les contes d'Hoffmann, Act II: Qu'as-tu donc? ...
58499	Ruggero Leoncavallo	Pagliacci: Vesti la giubba
42471	Henry Jamison	True North
44414	Sleeping At Last	Total Eclipse of the Heart
183048	Idina Menzel	This Day / Walking by a Wedding
53868	Omar Kent Dykes	Since I Met You Baby

Fig. 18: Recommended songs for test_1

```
Mean squared error of a randomly selected recommended song and
                test_1 playlist: 0.037186876

R2 score of a randomly selected recommended song and test_1
                playlist: 0.1461558123561445
```

4.2. DBSCAN

For all DBSCAN applications, the min_samples hyperparameter is determined as 2*#features. Epsilon is determined after a search algorithm by k-NN. When this algorithm is not enough, epsilon is chosen based on trial. The trials are done by considering the number of clusters and the amount of elements labelled as noise. As the epsilon and min_samples increases, cluster number reduces while the noise labels reduce.

4.2.1. DBSCAN with No Feature Reduction

In this experiment, the input data is only preprocessed. The number of features is 26. Therefore, min_samples is set to 52. The k-NN method with 52 nearest neighbors is run through. The below plot depicts knee epsilon value. The silhouette score increased as the epsilon value increased. In this sense, epsilon is set to 0.8 after some trials.

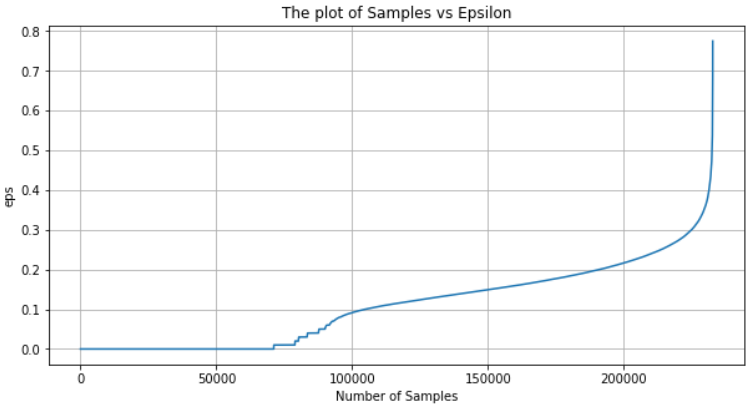


Fig.19: k-NN results to detect the knee for “epsilon”.

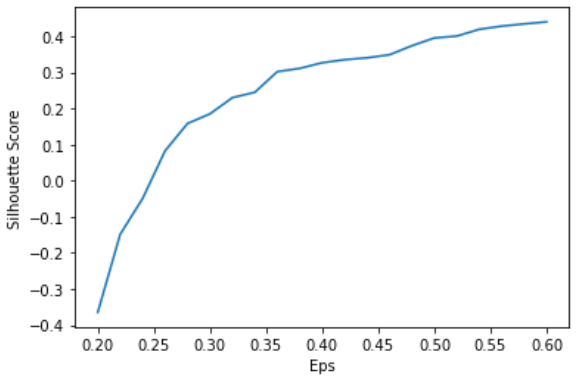


Fig.20: The plot of Silhouette score versus “epsilon” .

After the model is trained with the chosen parameters, the corresponding cluster names and sizes are displayed. A prediction algorithm based on euclidean distance is generated to take results from each test playlists. 10 samples for each playlist is taken just like Spotify. Test playlist 1 has the below recommended songs.

```
Mean squared error of recomended song and test_1 playlist: 0.0583795215643797
R2 score of recomended song and test_1 playlist: 0.5105692426491637
```

Fig. 21: Similarity between recommended song and test_1.

A randomly chosen 1 song has the above similarity between the vectorized test playlist 1, which is considerably high.

```
silhouette_score(df, dbscan_nopca.labels_)
```

```
0.4432927857125493
```

```
np.unique(dbscan_nopca.labels_, return_counts=True)
```

```
(array([-1,  0,  1,  2,  3,  4,  5,  6,  7,  8,  9, 10, 11, 12, 13, 1
        16, 17, 18, 19, 20, 21, 22, 23]),
 array([   18, 16245,  7124,  5638, 11758,  8097, 21943, 20028,  8722
         5105, 11329, 18466,  8665,  7442,  6361,  8795,  8084, 13874
```

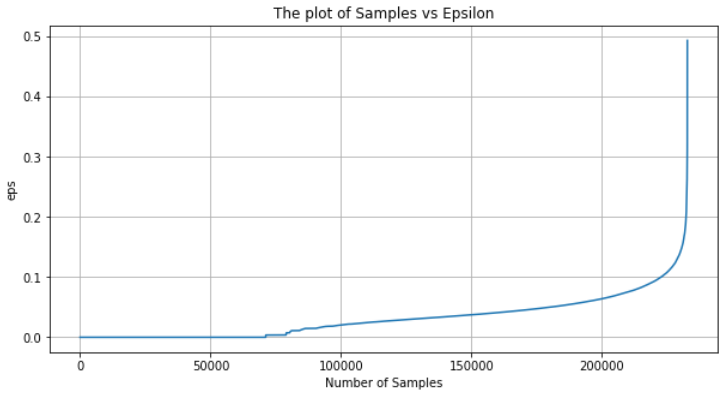
Fig.22 : The silhouette score of the final model, and distribution of clusters.

	artist_name	track name
11	Frank Churchill	3 Little Pigs: Who's Afraid of the Big Bad Wolf
8	Franco Escamilla	Series De La Infancia
13	La Mesa Reñoña	Episodio 14 (Machismo, Juanga, Bebé a Bordo)
14	La Mesa Reñoña	Episodio 15 (Lady Orinoco, Políticas De Youtub...
3	Pepper	Ufa Point Skit
17	Jon Hopkins	Late Night Tales: Jon Hopkins - Continuous Mix
5	Keem the Cipher	Alpha.
15	Jamie Llewellyn	Nature Sounds for Sleep: Crackling Log Fire wi...
10	Pepper	Ufa Point Skit
1	Bonobo	Late Night Tales: Bonobo

Fig.23: Recommended songs for test_1

4.2.2 DBSCAN with PCA

After the dataset is passed through PCA, the number of features reduces to 15. Therefore, min_samples is set to 30. k-NN algorithm with 30 nearest neighbours is applied to the data. The below figure displays the knee plot for epsilon values. Since the Silhouette score increases as epsilon increases, epsilon is taken as 0.8.



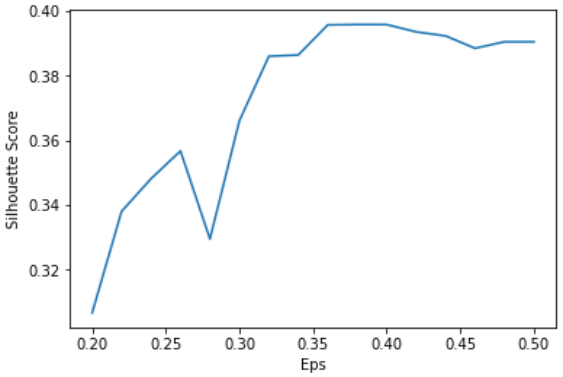


Fig.24: k-NN results to detect the knee for "epsilon". Fig.25: The plot of Silhouette score versus "epsilon"

```
silhouette_score(df, tuned_dbscan_08.labels_)
```

```
0.3958512606691428
```

```
np.unique(tuned_dbscan_08.labels_, return_counts=True)
```

```
(array([ 0,  1,  2,  3,  4,  5,  6,  7,  8,  9, 10, 11, 12, 13, 14, 15, 16,
        17, 18, 19, 20, 21, 22, 23]),
 array([16246,  7125,  5639, 11758,  8097, 21944, 20029,  8724,  5108,
        11329, 18466,  8666,  7442,  6361,  8795,  9520,  8084, 13876,
         8141,  8521,  5611,  6955,  3830,  2458]))
```

	artist_name	track name
781	H.E.R.	Pigment
1094	Nicola Porpora	Porpora: Cello Concerto in G Major: II. Largo
2167	Allan Rayman	Go My Way
273	Bonobo	Know You
2355	Rascal Flatts	Fast Cars And Freedom
2173	Abel Korzeniowski	Evgeni's Waltz
1754	Gaviria	Rico
2438	Allan Rayman	Faust Road
947	G Herbo	Boww
1680	Menor Menor	Bandida

Fig.26 : The silhouette score of the final model, and distribution of clusters.

Fig.27 : Recommended songs for test_1

The Silhouette score is less than the No Feature Reduction case. The cluster distribution is displayed above near 10 song recommendations. Moreover, the cosine similarity is applied to the desired cluster to take the most similar 10 songs. In terms of the quality of the recommendations, cosine similarity performed better.

	artist_name	track_name
0	Chorus	Mann Biachen
1	Glad	You Put This Love in My Heart
2	Hyannis Sound	Would You Go With Me
3	Brent Faiyaz	Burn One (Interlude)
4	Austin Wintory	Feasting on a Lord
5	Little Dragon	Best Friends - Christian Rich Rework
6	Belle & Sebastian	Piazza, New York Catcher
7	Billie Holiday	The Way You Look Tonight (with Teddy Wilson & ...
8	C.W. Stoneking	The Love Me or Die
9	Eric Bibb	With My Maker I Am One

Fig.28 : Recommended songs for test_1 with cosine similarity

```
Mean squared error of recomended song and test_1 playlist: 0.03561333989003694
R2 score of recomended song and test_1 playlist: 0.528985606383686
```

Fig.29: Similarity between recommended song and test_1.

The similarity of the recommended song after cosine similarity improved slightly. Considering the computationality, other methods did not go through cosine similarity filtering.

4.2.3. DBSCAN with Autoencoder

Finally, the output of the autoencoder is inputted to DBSCAN. Autoeencoder reduced the features to 15. Therefore, 30 features are used as well as PCA case. Epsilon is determined after a search loop of Silhouette score. The peak score is obtained at 0.26 epsilon value. Therefore necessary parameters are set for training.

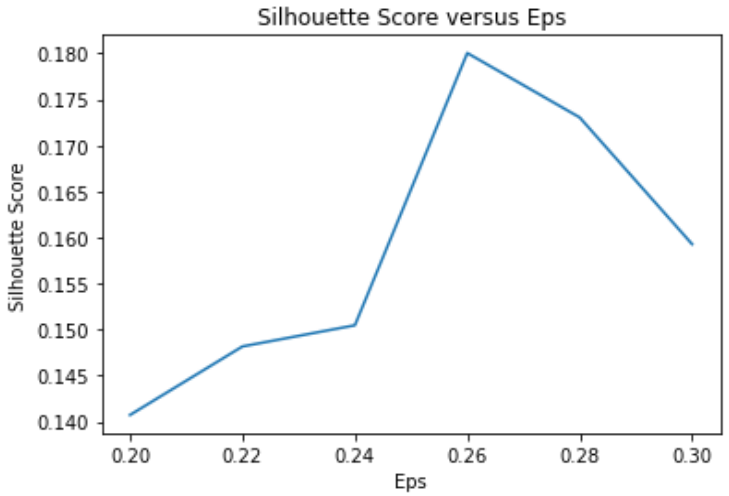


Fig.30: The plot of Silhouette score versus “epsilon”

After training the model, 23 clusters are generated including a noise cluster (-1). The song

```
silhouette_score(df, dbscan_ae.labels_)
```

```
0.18002737
```

```
np.unique(dbscan_ae.labels_, return_counts=True)
```

```
(array([-1,  0,  1,  2,  3,  4,  5,  6,  7,  8,  9, 10, 11, 12, 13, 14, 1
        16, 17, 18, 19, 20, 21]),
 array([ 2955, 14971, 22857, 19997, 27208, 26042, 17128,  7358, 14923,
        23729, 15259, 22404, 17413,    80,    42,    49,    61,    32,
           38,    46,    63,    37,    33]))
```

Fig.31: The silhouette score of the final model, and distribution of clusters.

	artist_name	track name
1885	Justin Hurwitz	Sextant
2453	grandson	Kiss Bang
1558	Lagwagon	Sleep
2305	Rodrigo y Gabriela	Tamacun - Remastered
1096	Franz Schubert	Introduction & Variations on "Trockne Blumen",...
628	Ibrary's Children's Music & Stories	The Walrus and the Carpenter
1983	Idina Menzel	Marvin's Long Lost Brother
2151	John Williams	Meeting Aragog
2019	Hayley Mills	Edward's Problem
1382	Ludwig van Beethoven	Fur Elise

Fig.32 : Recommended songs for test_1.

recommendations are displayed below.

To demonstrate the performance of the algorithm, MSE and r2 score is displayed for a random song of the recommendation.

```
Mean squared error of recomended song and test_1 playlist: 0.046455577
R2 score of recomended song and test_1 playlist: -0.06666187076917929
```

Fig.33: Similarity between recommended song and test_1.

Although the recommendations are verified to be good for the playlist by the users, the r2 score gave pretty bad results.

5. Discussion

In this section, the methods will be evaluated based on silhouette scores, runtime durations, and preciseness of the musical properties with mean squared error and r2 score.

Table 1 shows the Silhouette scores of the methods. The methods should not be compared with each other since each method uses different number of clusters, which would distort the assessment. Instead, one can compare each input's effect on the methods. k-Means scores the best with PCA while DBSCAN performs the best with no feature reduction. Autoencoder surprisingly scores the lowest in this assessment. The dataset may be more suitable for linearization instead of fitting with nonlinear activation functions of autoencoder. k-Means' performance on PCA is better than no feature reduction. This can be reasoned as follows. k-Means divides the cluster linearly. PCA also takes a linear projection of the dataset. Therefore, it works better with PCA. Moreover, the preprocessed data has many one-hot encoded columns which causes a lot of sparsity. Sparsity damages the model performance, and should be eliminated by a reduction algorithm. On the other hand,

DBSCAN takes density curves as clusters which is not a linear form. The reduction of the data might not be needed for DBSCAN. However, the final decision of the best algorithm will not solely depend on Silhouette score.

Table 1: Comparison of Silhouette scores of methods.

	No Feature Reduction	**PCA**	**Autoencoder**
k-Means	0.184	0.297	0.114
DBSCAN	0.444	0.396	0.180

Comparing the runtime durations of DBSCAN and k-Means, DBSCAN has taken a lot more time than k-Means. Also, higher epsilon values cause the RAM to be full, which means the computational expense is also higher than k-Means. To be time and computationally efficient, the chosen algorithm is k-Means. Moreover, comparing the recommended songs of the algorithms, k-Means made more relevant recommendations. Besides, the tunability of the algorithms constitute high importance. The new coming data should be easily tuned accordingly. Unfortunately, DBSCAN only indirectly manipulates cluster numbers. In many cases, it outputs noise if cluster number is desired to be increased. On the other hand, if noise is wanted to be decreased, it drastically reduces the number of clusters, which causes less relevant recommendations. However, k-Means has a parameter and a valid algorithm to tune the cluster numbers, which helped us comment on the diversity of the dataset. All in all, k-Means is the optimal method. Therefore, the most optimal input-algorithm couple is chosen to be k-Means with PCA.

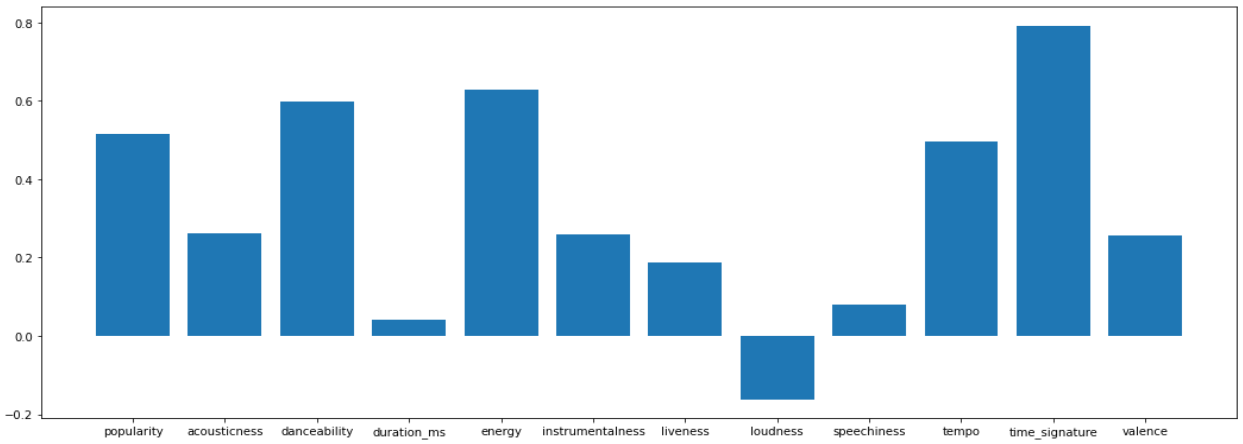


Fig.34: Musical properties of a successful recommendation: Soul-Dwele.

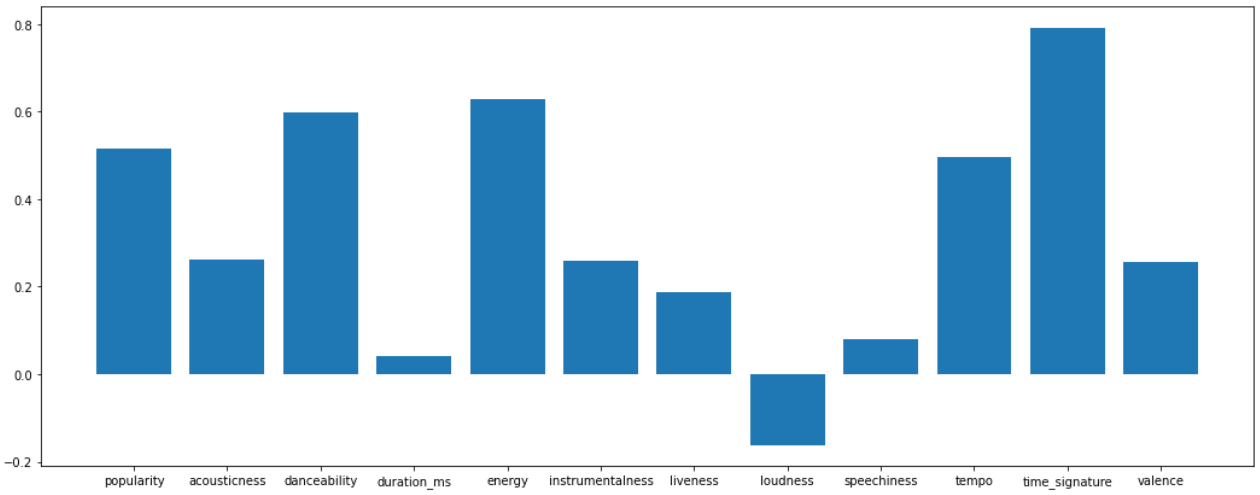


Fig.35: Vectorized musical properties of test playlist 1.

```
Mean squared error of recomended song and test_1 playlist: 0.03238653063179512
R2 score of recomended song and test_1 playlist: 0.6433988618926254
```

Fig.36: Similarity between recommended song and test_1.

To demonstrate the similarity between the test playlist 1 and the recommended song, the musical properties of the vectorized test playlist and recommended song is evaluated by mean squared error and r2 score. It can be deduced that the algorithm work satisfactorily, outputting songs with similar musical properties.

Although the results are quite satisfactory, there are still things to improve. The used dataset has somewhat unpopular songs and a questionable input column as genre. The recommended songs were not familiar to the listeners, which would decrease the reliability of the application. Moreover, the genre information is not provided by Spotify API for a single song. Rather, the artists have more than one genres that they are related to. Therefore, it is a slippery slope to randomly assign a song of its artist's genre. Since the train dataset cannot be matched to the test dataset because of the genre column, the genre information is taken out of the dataset.

6. Conclusions

In this term project, the aim was to generate song recommendations for Spotify playlists extracting data from Spotify API. The problem is unsupervised learning, namely clustering. The project focuses on 2 clustering algorithms with 3 different input combinations. k-Means and DBSCAN is used for clustering. As a helper method, k-NN is used for the elbow method and epsilon search method. The models are trained with only preprocessed data, PCA, and Autoencoder. The performance of clustering is evaluated by Silhouette score. The performance of the recommendation is evaluated by the similarity of the musical properties of the recommendation and test playlists. In this sense, mean squared error and r2 score is utilized. At the end of the algorithm development process, k-Means algorithm with PCA input is chosen as the optimal case. It has considerably low runtime duration, and more precise recommendations. From a developer perspective, the parameters of k-Means are easy to tune. It is observed that the input song dataset is more suitable for linearization. Moreover, k-Means works poorly with sparse data. To suggest improvements that could not be possible in this project, the used train dataset could include more popular and meaningful songs so that the recommended songs would address more people as a real life application. Another thing is that the correct genre information would plausibly improve our score even more. All in all, one can conclude that the chosen algorithm generates precise song recommendations as we wished.

7. Appendix

- Arda implemented PCA and Autoencoder models. Tried a set of cluster numbers in k-means algorithm for 3 different input types.
- Ayşın applied one-hot encoding as preprocessing, DBSCAN algorithm for 3 input types, and extracted song information from Spotify API for test playlists.
- Bahadır took responsibility in the preprocessing stage, data editing and feature reduction method autoencoder.
- Barış helped taking information from Spotify API.

8. Reference

[1] J. Jordan, “Introduction to autoencoders.,” *Jeremy Jordan*, 19-Mar-2018. [Online]. Available: https://www.jeremyjordan.me/autoencoders/. [Accessed: 27-Dec-2022].

[2] T. Mullin, “DBSCAN parameter estimation using Python,” *Medium*, 15-Jul-2020. [Online]. Available: https://medium.com/@tarammullin/dbscan-parameter-estimation-ff8330e3a3bd. [Accessed: 26-Dec-2022].

[3] “DBSCAN clustering in ML: Density based clustering,” GeeksforGeeks, 24-Aug-2022. [Online]. Available: https://www.geeksforgeeks.org/dbscan-clustering-in-ml-density-based-clustering/. [Accessed: 20-Dec-2022].

[4]A. Kumar, “Kmeans silhouette score with python examples - dzone,” *dzone.com*, 21-Jun-2021. [Online]. Available: https://dzone.com/articles/kmeans-silhouette-score-explained-with-python-exam#:~:text=Silhouette%20score%20is%20used%20to,each%20sample%20of%20different%20clusters. [Accessed: 26-Dec-2022].