MEF UNIVERSITY

Playlist Recommendation with Similarity Based Approach using Spotify Million Playlist Dataset

Capstone Project

Murat Can Taşar

Advisor: Dr. Öğr. Üyesi Utku Koç

ISTANBUL, 2022

Academic Honesty Pledge

I promise not to collaborate with anyone, not to seek or accept any outside help, and not to give any help to others.

I understand that all resources in print or on the web must be explicitly cited.

In keeping with MEF University’s ideals, I pledge that this work is my own and that I have neither given nor received inappropriate assistance in preparing it.

Name: Murat Can Taşar

Date: 02/09/2019

Signature:

TABLE OF CONTENTS

1. SPOTIFY MPD CHALLENGE

Spotify Million Playlist Dataset (MPD) challenge has been running since 2018 as a continuation of another challenge called “RecSys Challenge 2018”. The dataset contains 1 million unique playlists with playlist titles created by Spotify users between 2010 and 2017. The main evaluation criteria are based on playlist continuation, and the evaluation metrics have been created by Spotify to best measure the resulting playlists (Spotify, aicrowd.com, n.d.).

The dataset first introduced in RecSys Challenge in 2018. It was sampled over 4 billion public playlists on Spotify created by US users. According to Spotify, it is the largest playlist dataset available to public on the internet (Spotify, aicrowd.com, n.d.).

The main challenge was to create and fill 10000 playlists given as challenge. Unfortunately, the scoring system on AICrowd was not functioning properly, so I turned my attention to create custom playlists instead of optimizing submission results.

2. RELATED WORKS

There are many who have come up with a unique solution of their own to compete in the Spotify MPD or RecSys 2018 Challenge. Some of the solutions and approaches of those entrants are available online in an article format.

One of these solutions created by Volkovs et al. uses a two-stage approach to create an automatic playlist continuation process (Volkovs, et al., 2018). The first stage is to retrieve 20 thousand songs for each playlist using Collaborative Filtering model with a twist on Matrix Factorization approach. Retrieved songs then gets scored using CNN as well as neighbor-based similarity methods, which are namely item-item and user-user. After weighing each playlist recommendations, using the track features available on Spotify, they used XGBoost to rerank all the tracks to come up with a final recommendation list.

Wu, Anand and Sydykov state that they tried to solve this problem by applying an autoencoder to the playlists (Wu, Anand, & Sydykov, 2018). By using an autoencoder they reduced the available dimensions and applying a collaborative filtering method by using matrix factorization let them solve this problem. They also compared the performances of denoising autoencoders with variational autoencoders during the span of this challenge.

Another approach leveraged nearest neighbor method to come up with a solution. Ludewig et al., used an unconventional method of finding the similarities between playlist names and tracks to recommend to different challenge scenarios (Ludewig, Kamehkhosh, Landia, & Jannach, 2018).

3. DATA PREPARATION

3.1. Dataset Overview

Spotify MPD challenge provides two different datasets; one contains playlists with track ids and several other information to work on, the other one contains playlists ready to be filled using the work set provided. Both these datasets are in the same format, and contain yet another dataset inside a column, named "tracks". A general overview of these said datasets can be found in the table below.

Column Name	**Type**	**Example**	**Column Name**	**Type**	**Example**
name	string	Workout	num_followers	int	1
collaborative	bool	False	tracks	Dictionary	see Table 2
pid	int	100003	num_edits	int	26
modified_at	int	1509321600	duration_ms	int	14989941
num_tracks	int	61	num_artists	int	36
num_albums	int	49	description	string	gymlist

Table 1. MPD Raw Dataset Overview

As mentioned above, column "tracks" contain a huge dictionary inside. When unpacked from the original dataset, its general overview can be found below.

Column Name	**Type**	**Example**	**Column Name**	**Type**	**Example**
pos	int	2	track_name	string	What
artist_name	string	BRONCHO	album_uri	string	spotify:album:7enH7cYcQOGSyR8qCN39MB
track_uri	string	spotify:track:5xhukedm2mj6DcfL76dxni	duration_ms	int	136280
artist_uri	string	spotify:artist:6Lll1MPPak4m4vZKuJB264	album_name	string	Just Enough Hip to be Woman

Table 2. Inside of column name "tracks" available on the workset.

My main aim was to come up with a dynamic solution to fill each challenge playlist using the dataset at hand, but last-minute complications caused by the challenge scoring system prevented the project to reach that point. Instead, it seemed fit to turn this challenge into a small product.

3. 2. Data Preparation Stages and Challenges Faced during Preprocessing

The dataset provided consisted of 1 million unique playlists, with around 2.6 million unique tracks in total. These playlists were provided with 1000 json files, each containing 1000 unique playlists. Since the goal was to create a recommendation system, musical information about each track was necessary. Luckily, a Python library called Spotipy, which is a user-friendly wrapper of Spotify Developer API, came in very handy at this stage. At this point, it was unavoidable to retrieve each track's musical features, such as its danceability, valence, tempo, energy etc. using the Spotipy library. The methodology and its stages are summarized in the table below.

Stage	Method
1	Unpacking playlists from json files
2	Unpacking unique tracks from playlists
3	Creating Spotify id and secret
4	Retrieving track features using Spotipy wrapper

Table 3. Stages of Preprocessing

There were many challenges faced during this stage. Below you can find a table stating the challenges and how they were resolved. Each problem stated represented alongside with a stage number refers to the problems risen under that specific stage given in Table 3.

Stage	Problem	Solution
1	Reading all json files	Converted json files into csv's so that they hold almost half the space on disk. Each json file was around 40MB and a converted csv file was around 20MB.
1	Reading all csv files	Since there were 1000 csv files each with a size of around 20MB, a single dataframe containing all the playlists would hold around 20*1000 = 20000MB = 20GB space. To overcome this issue, I created 4 different 5 GB buckets of the dataset to make it relatively easy to process.
2	Unpacking tracks in 4 different playlist buckets	Had to create a notebook specifically designed to unpack, retrieve, hold, and append each track in playlist for 4 buckets of data to be able to retrieve each track's feature from Spotify.
4	Finding optimum delay to retrieve each track's feature from Spotify	This was the step that took most of the project time. Spotify API has two main bottlenecks if you want to retrieve track features. One can only request the features for 100 tracks at once, also there must be an undefined amount of delay between each query. A delay of 0.05 seconds was to be found as the most valid option on Stackoverflow. Retrieval of all track features took around 2 days after finding the optimum delay and number of tracks to be sent at once.

Table 4. Challenges Faced at each Preprocessing Stage

After implementing the solutions stated in Table 4, the final dataset's metadata can be found in the table below.

Column Name	Type	Explanation
uri	string	A special string used to describe a track's id on Spotify servers.
danceability	float	Danceability describes how suitable a track is for dancing based on a combination of musical elements including tempo, rhythm stability, beat strength, and overall regularity. A value of 0.0 is least danceable and 1.0 is most danceable.

energy	float	Energy is a measure from 0.0 to 1.0 and represents a perceptual measure of intensity and activity. Typically, energetic tracks feel fast, loud, and noisy. For example, death metal has high energy, while a Bach prelude scores low on the scale. Perceptual features contributing to this attribute include dynamic range, perceived loudness, timbre, onset rate, and general entropy.
key	integer	The key the track is in. Integers map to pitches using standard Pitch Class notation. E.g. 0 = C, 1 = C♯/D♭, 2 = D, and so on. If no key was detected, the value is -1.
loudness	float	The overall loudness of a track in decibels (dB). Loudness values are averaged across the entire track and are useful for comparing relative loudness of tracks. Loudness is the quality of a sound that is the primary psychological correlate of physical strength (amplitude). Values typically range between -60 and 0 db.
mode	integer	Mode indicates the modality (major or minor) of a track, the type of scale from which its melodic content is derived. Major is represented by 1 and minor is 0.
speechiness	float	Speechiness detects the presence of spoken words in a track. The more exclusively speech-like the recording (e.g. talk show, audio book, poetry), the closer to 1.0 the attribute value. Values above 0.66 describe tracks that are probably made entirely of spoken words. Values between 0.33 and 0.66 describe tracks that may contain both music and speech, either in sections or layered, including such cases as rap music. Values below 0.33 most likely represent music and other non-speech-like tracks.
acousticness	float	A confidence measure from 0.0 to 1.0 of whether the track is acoustic. 1.0 represents high confidence the track is acoustic.
instrumentalness	float	Predicts whether a track contains no vocals. "Ooh" and "aah" sounds are treated as instrumental in this context. Rap or spoken word tracks are clearly "vocal". The closer the instrumentalness value is to 1.0, the greater likelihood the track contains no vocal

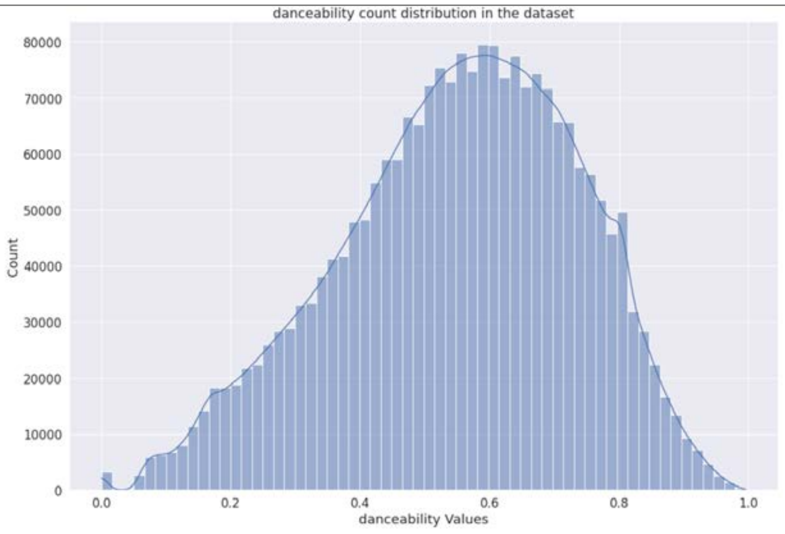
		content. Values above 0.5 are intended to represent instrumental tracks, but confidence is higher as the value approaches 1.0.
liveness	float	Detects the presence of an audience in the recording. Higher liveness values represent an increased probability that the track was performed live. A value above 0.8 provides strong likelihood that the track is live.
valence	float	A measure from 0.0 to 1.0 describing the musical positiveness conveyed by a track. Tracks with high valence sound more positive (e.g. happy, cheerful, euphoric), while tracks with low valence sound more negative (e.g. sad, depressed, angry).
tempo	float	The overall estimated tempo of a track in beats per minute (BPM). In musical terminology, tempo is the speed or pace of a given piece and derives directly from the average beat duration.
duration_ms	integer	The duration of the track in milliseconds.
time_signature	integer	An estimated time signature. The time signature (meter) is a notational convention to specify how many beats are in each bar (or measure). The time signature ranges from 3 to 7 indicating time signatures of "3/4", to "7/4".

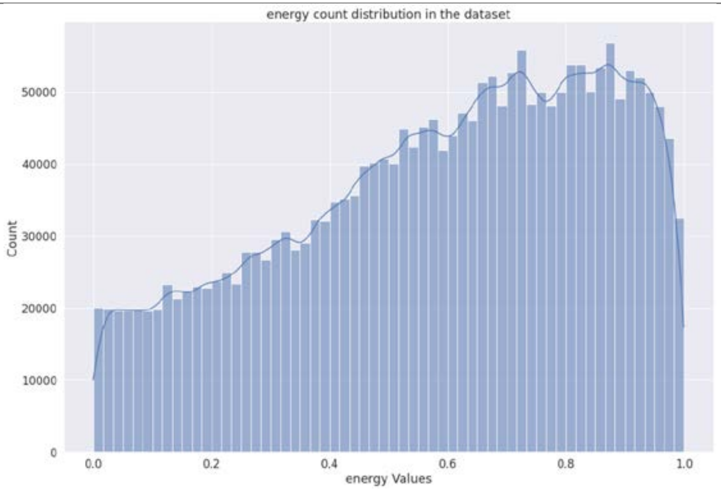
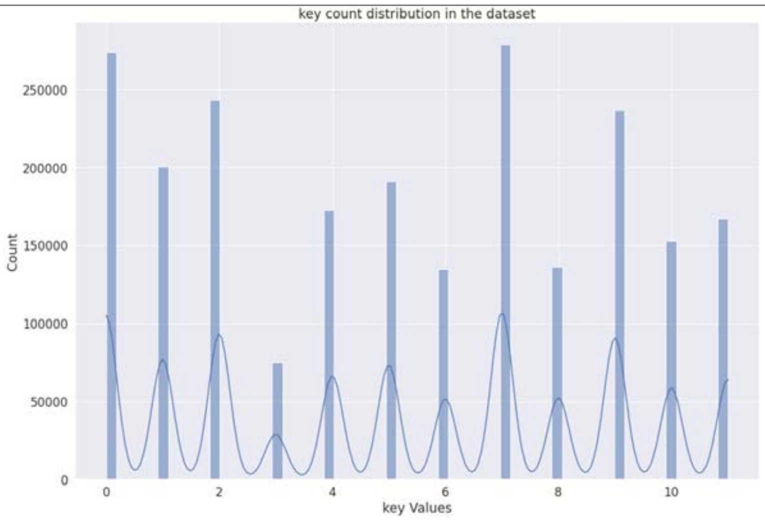
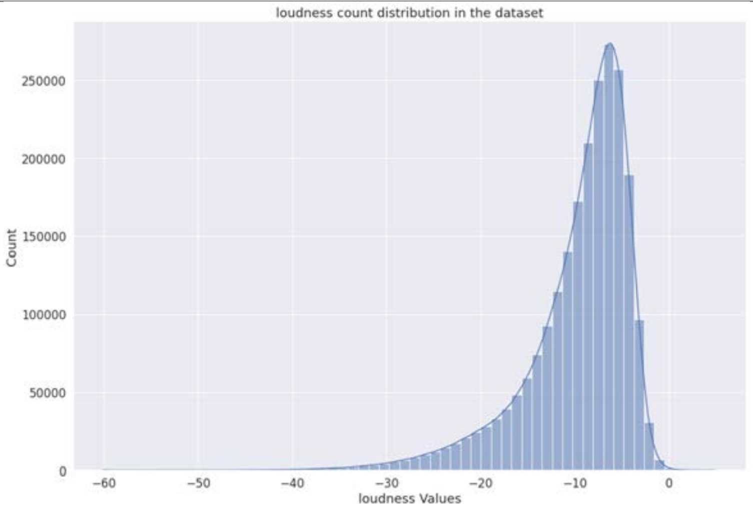
Table 5. Final Dataset Column Meanings (Spotify, 2022).

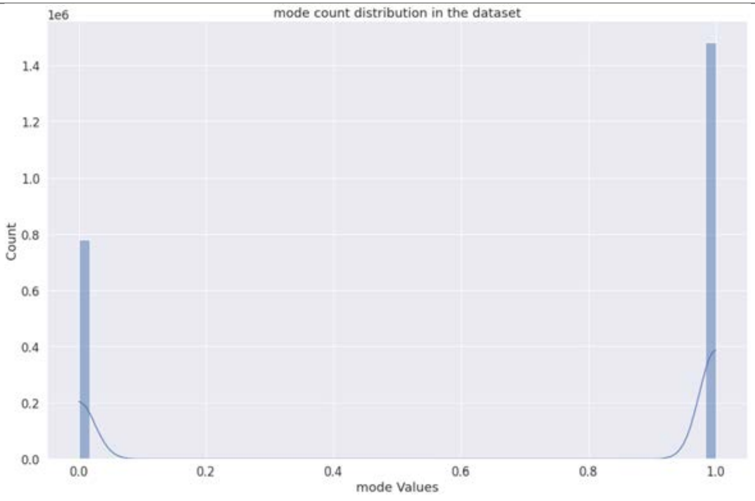
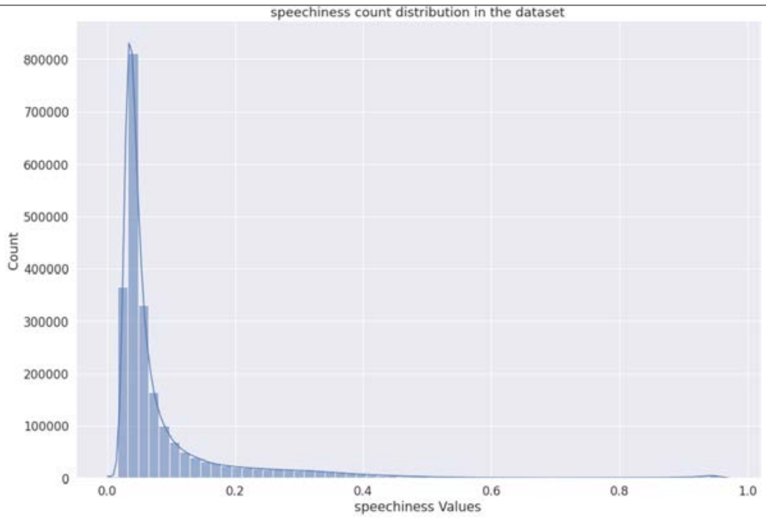
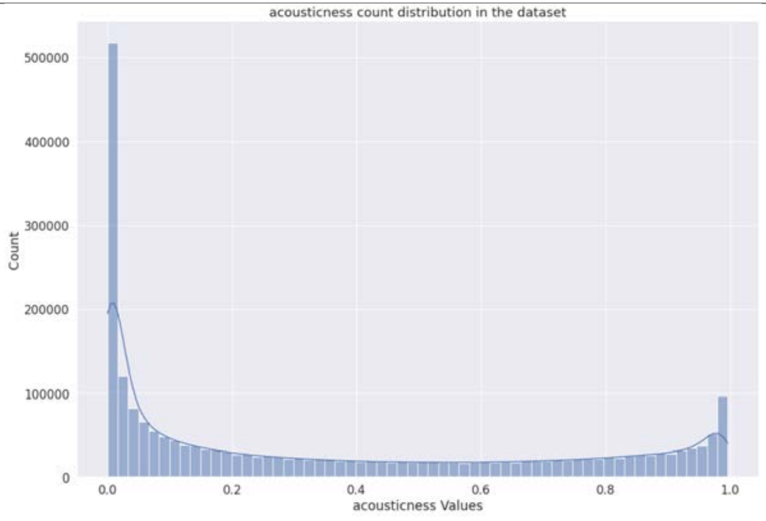
4. EXPLORATORY DATA ANALYSIS OF THE PREPROCESSED DATASET

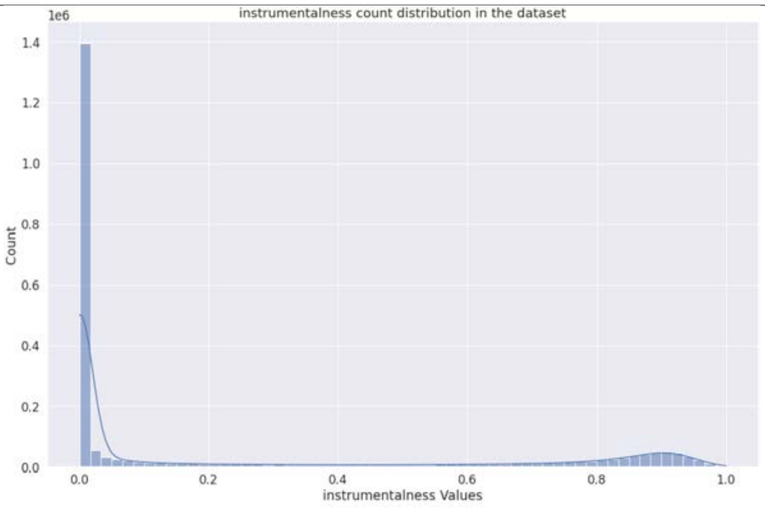
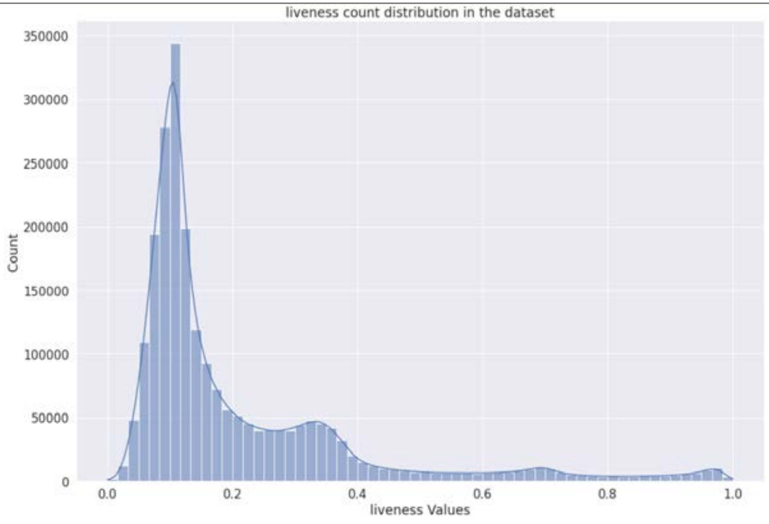
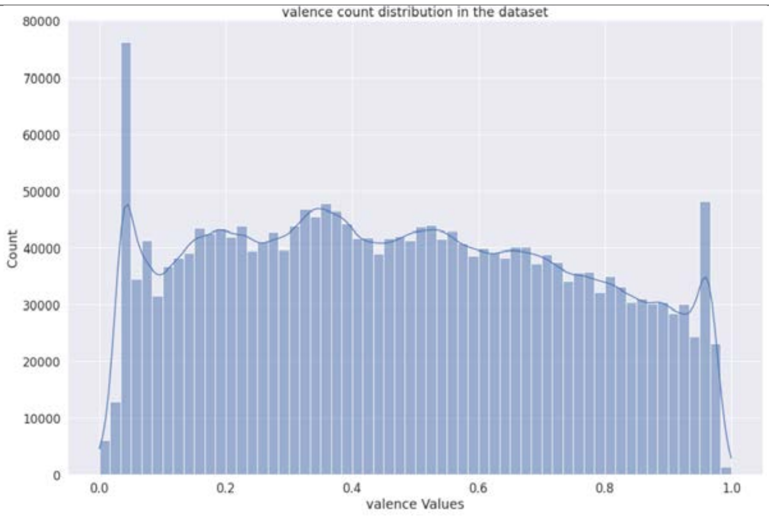
4.1. Histograms of each Track Feature

After being able to create the final dataset that was going to be used during the recommendation stage, it was complimentary to analyze the data and know what it contained. Looking at each feature's histogram proved useful, as it showed that there were no clear outliers as well as some of them seemed to have a bell-shaped distribution going on. Some features like mode, key and time time_signature seemed like categorical variables. Also as expected, duration_ms column had its values gathered around almost at the same area, since track duration for each song was to be expected to be more or less the same. In the table below, you can find the distribution of each track feature used in the recommendation system.

Feature	Graph	Comment
danceability		It resembles almost a normal distribution curve, which tells us that it probably will not cause any bias during recommendation phase.

energy		Same as danceability, its distribution gathers slightly around the middle.
key		It seems like this feature is a categorical feature, since it contains 12 different key values in it.
loudness		Most of the tracks at hand seem to gather around -10 dB, and forming a left skewed distribution.

mode		Mode seems to be a categorical feature as well. Almost 1/3 tracks have a mode of 0 (minor), and the remaining have a mode of 1 (major).
speechiness		According to Spotify documentation (Spotify, 2022), tracks that have less than 0.66 speechiness are probably songs, and our dataset is gathered around 0.1 speechiness.
acousticness		Seems like most of the tracks have no acoustic element in them, but remaining tracks follow a uniform distribution up to 1, which states that a track is acoustic.

instrumentalness		Using the information on Spotify documentation (Spotify, 2022), almost all tracks at hand are not instrumental tracks, but some are showing signs of instrumentalness.
liveness		Most tracks are gathered around 0.1, which states that almost all tracks are not live performances.
valence		Valence values are almost uniformly distributed, this creates a nice balance on overall feeling of the tracks at hand.

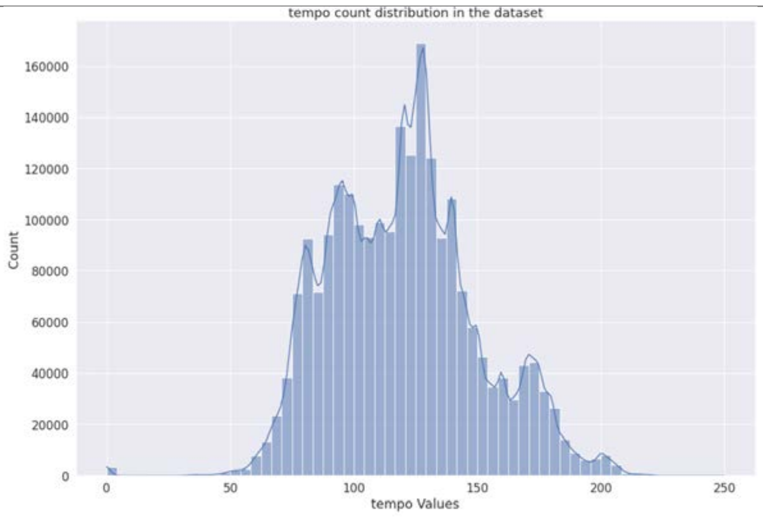
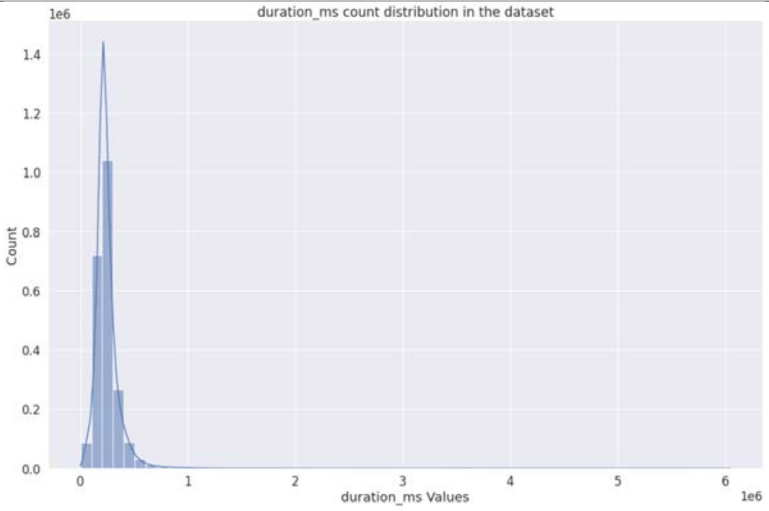
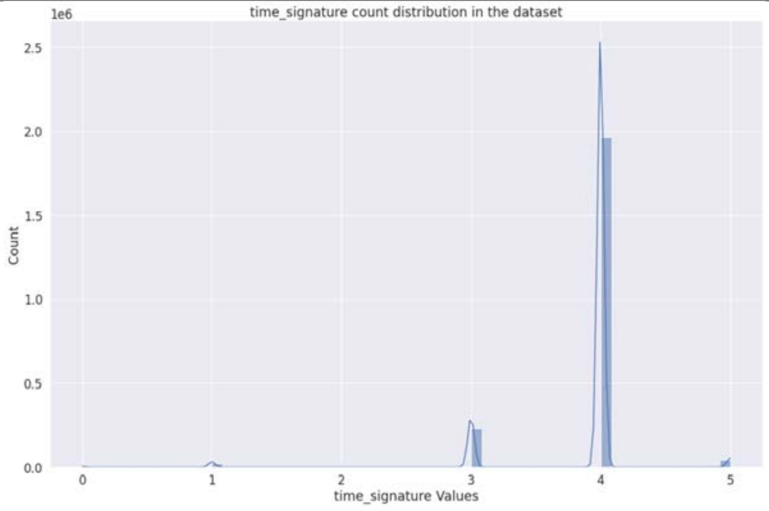
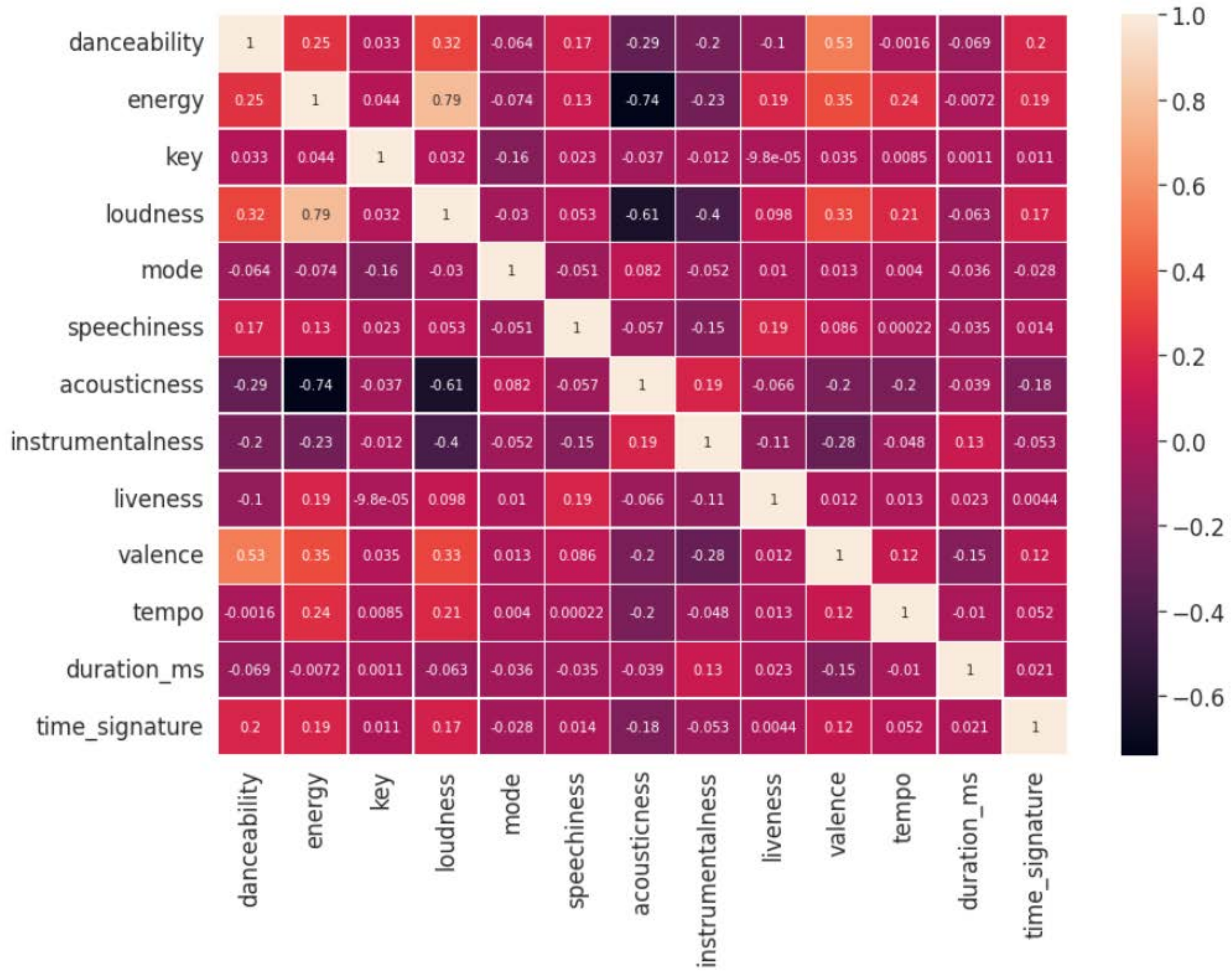
tempo		Seems like tempo values are located around 100 and 150 bpms, and cover a range of 50 to 200.
duration_ms		As expected, most of the tracks are around 500000 ms, which converts roughly to 8 minutes of track length.
time_signature		This feature also seems like a categorical feature, with most of it sitting on a time signature of 4. It shows that there are not so many complicated tracks available tempo-wise.

Table 6. Histograms of each Feature

4.2. Correlation Matrix of Track Features

It was necessary to remove highly correlated features before feeding the dataset into the recommendation system. Although it contained some correlated features such as acousticness-energy and loudness-energy, correlation was not high enough to make it justifiable to remove either of these features. All other features seemingly have almost no correlation between them, which provides a better explanability to the results. So, all features were considered without applying any feature elimination process during the similarity calculation phase.



	danceability	energy	key	loudness	mode	speechiness	acousticness	instrumentalness	liveness	valence	tempo	duration_ms	time_signature
danceability	1	0.25	0.033	0.32	-0.064	0.17	-0.29	-0.2	-0.1	0.53	-0.0016	-0.069	0.2
energy	0.25	1	0.044	0.79	-0.074	0.13	-0.74	-0.23	0.19	0.35	0.24	-0.0072	0.19
key	0.033	0.044	1	0.032	-0.16	0.023	-0.037	-0.012	-9.8e-05	0.035	0.0085	0.0011	0.011
loudness	0.32	0.79	0.032	1	-0.03	0.053	-0.61	-0.4	0.098	0.33	0.21	-0.063	0.17
mode	-0.064	-0.074	-0.16	-0.03	1	-0.051	0.082	-0.052	0.01	0.013	0.004	-0.036	-0.028
speechiness	0.17	0.13	0.023	0.053	-0.051	1	-0.057	-0.15	0.19	0.086	0.00022	-0.035	0.014
acousticness	-0.29	-0.74	-0.037	-0.61	0.082	-0.057	1	0.19	-0.066	-0.2	-0.2	-0.039	-0.18
instrumentalness	-0.2	-0.23	-0.012	-0.4	-0.052	-0.15	0.19	1	-0.11	-0.28	-0.048	0.13	-0.053
liveness	-0.1	0.19	-9.8e-05	0.098	0.01	0.19	-0.066	-0.11	1	0.012	0.013	0.023	0.0044
valence	0.53	0.35	0.035	0.33	0.013	0.086	-0.2	-0.28	0.012	1	0.12	-0.15	0.12
tempo	-0.0016	0.24	0.0085	0.21	0.004	0.00022	-0.2	-0.048	0.013	0.12	1	-0.01	0.052
duration_ms	-0.069	-0.0072	0.0011	-0.063	-0.036	-0.035	-0.039	0.13	0.023	-0.15	-0.01	1	0.021
time_signature	0.2	0.19	0.011	0.17	-0.028	0.014	-0.18	-0.053	0.0044	0.12	0.052	0.021	1

Table 7. Correlation Matrix of Features

4.3. Overall Description and Overview of the Dataset

By applying a simple describe function from the Pandas library to the dataset stated above, it is convenient to achieve a numerical description of each feature and their percentile analysis. Results can be found in the table below. Note that they are only a numerical representation of the histograms given in section 5.2.

	Count	Mean	Std	Min	25%	Median	75%	max
danceability	2262190	0.55	0.18	0	0.43	0.56	0.69	1
energy	2262190	0.58	0.27	0	0.39	0.62	0.81	1
key	2262190	5.26	3.56	0	2	5	8	11
loudness	2262190	-9.66	5.63	-60	-11.91	-8.18	-5.83	4.92
mode	2262190	0.65	0.48	0	0	1	1	1
speechiness	2262190	0.09	0.12	0	0.04	0.05	0.08	0.97
acousticness	2262190	0.35	0.35	0	0.02	0.21	0.69	1
instrumentalness	2262190	0.22	0.35	0	0	0	0.42	1
liveness	2262190	0.21	0.19	0	0.1	0.13	0.26	1
valence	2262190	0.48	0.27	0	0.25	0.47	0.7	1
tempo	2262190	119.99	29.92	0	96.94	120.01	138.05	249.99
Duration_ms	2262190	247437.74	156204.03	1000	184291	225280	278295.5	6047705
Time_signature	2262190	3.88	0.47	0	4	4	4	5

Table 8. Description of the Dataset

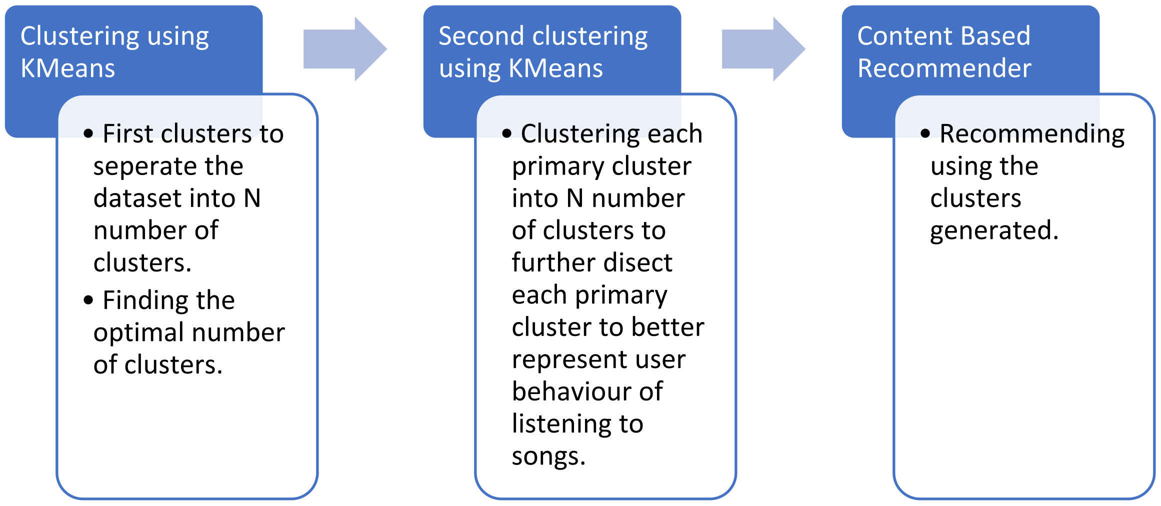
Additionally, table below represents an overview of counts of tracks, albums, artists etc. Note that this list represents the raw numbers of the dataset. While track features were being retrieved from Spotify, there are exactly 102 tracks in the dataset that Spotify could not provide a set of features to, which bring the total of unique tracks to 2,262,190.

Property	**Value**
Number of unique playlists	1,000,000
Number of tracks (duplicated)	66,346,428
Number of unique tracks	2,262,292
Number of unique albums	734,684
Number of unique artists	295,860
Number of unique playlist titles	92,944
Average playlist length (tracks)	66.35

Table 9. Overall Statistics of the Dataset

5. USED METHODOLOGY

Firstly, to generate a playlist using similarities between each track, there were basically two main school of thoughts to follow. One was to simply use the whole dataset and find the most similar tracks to a vector and recommend. This was the simplest approach that let us complete the task successfully. On the other hand, we had to think about how realistic this approach would be if it were to be scaled to many different users. Clustering people's music likings into rigid groups would certainly eliminate some of the unseen elements and turn the problem into a basic one. But on the other hand, according to Dannenberg the term "genre" is a category of music defined by a specific style and can be influenced by social interactions (Dannenberg, 2010). Since recommending a song to a person should consider their favorite tunes, artists, albums, or in general, genres, it would be best to cluster the songs at hand before proceeding with recommendations. However, I have implemented both approaches into a single function to compare results from both sides. In the upcoming sections, the results will be explained in a subjective manner since the better approach can only be determined using a personal touch in the end. A simple process line below explains the steps involving the whole project also covers the methodologies used during each stage.


Clustering using KMeans
• First clusters to seperate the dataset into N number of clusters.
• Finding the optimal number of clusters.
Second clustering using KMeans
• Clustering each primary cluster into N number of clusters to further disect each primary cluster to better represent user behaviour of listening to songs.
Content Based Recommender
• Recommending using the clusters generated.

5.1. Unsupervised Clustering using KMeans Algorithm

Since our dataset contains no labels or even potential candidates for labels, the main approach used for this project was to use unsupervised machine learning algorithms, specifically KMeans, to better understand and manage tracks at hand. An intuitive approach instead of using clustering would be to separate tracks using genres. Since Spotify only releases genre information for artist and albums, the idea of using this information thought to be not explain user playlist creation behavior very well.

KMeans according to Piench is one of the most used clustering algorithms. The main objective of this algorithm is to find clusters that contain the minimal Euclidean distance from the center of the cluster, which is referred to as centroid (Piech, 2013).

Just like using any other algorithm, KMeans also has its own advantages and disadvantages depending on the problem. Its two main positives are that it is very easy to implement and comfortable to scale to larger datasets. The clusters found are the results of the convergence, so it is safe to assume that the end clusters are converged. One main downside of this algorithm is that the number of clusters should be initialized at the beginning. This means that optimum number of clusters can and probably will be different than the initialized number. Thus, certain validation methods had to be applied alongside it, such as magnitude and cardinality analysis as well as silhouette score analysis. Also, another complication could be caused by the dimensions available on the dataset (Google, 2022). Since we have 13 different dimensions ready-to-be-used, this can further cause our clusters to not be as separate or identifying as we want to be. This problem will be talked about in the future works section.

5.2. Finding the Optimum Number of Clusters

Since initializing the number of clusters in KMeans is said and accepted as to be a disadvantage of the algorithm, finding the optimum number of clusters requires additional analysis. These analyses are very simple in logic, and yet provide enough explainability to back up the reasoning behind the cluster selection process. Google Machine Learning Course Hub page provides in depth informative article about how one of these analyses can be conducted on a dataset. In the next section quality of clusters and finding the optimum number will be discussed.

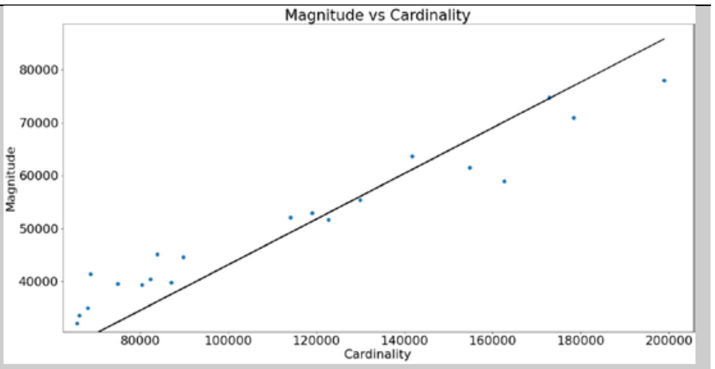
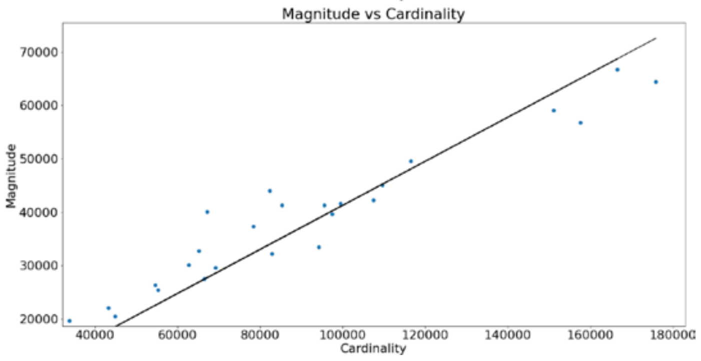
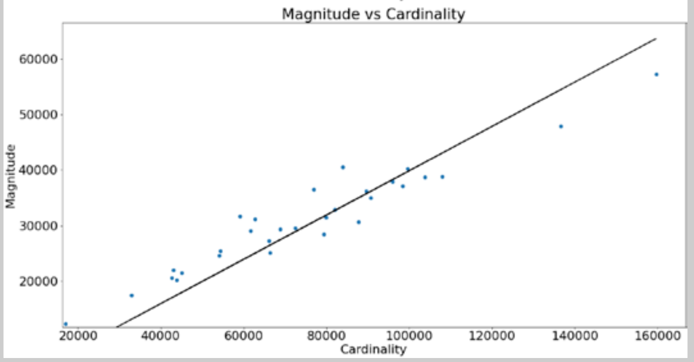
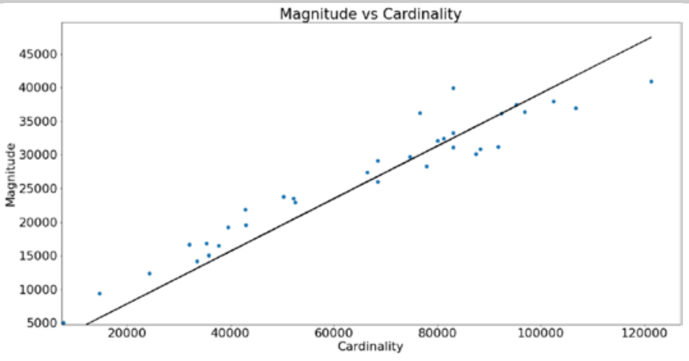
5.2.1. Quality of clusters

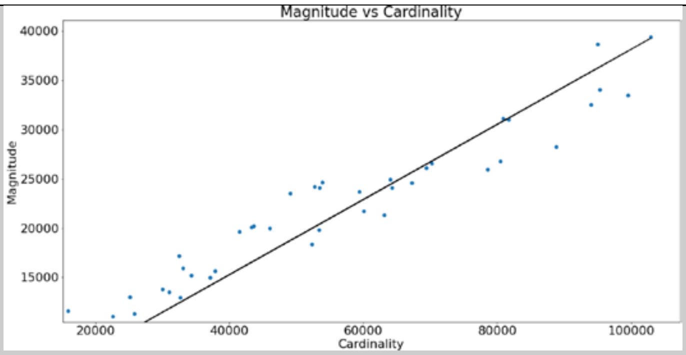
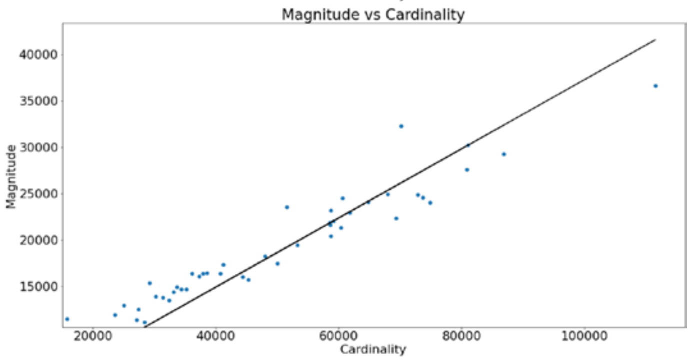
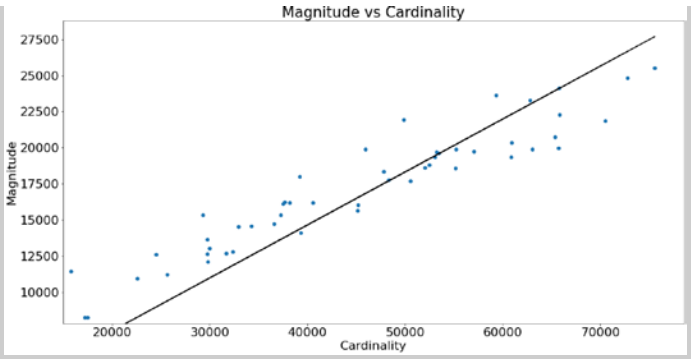
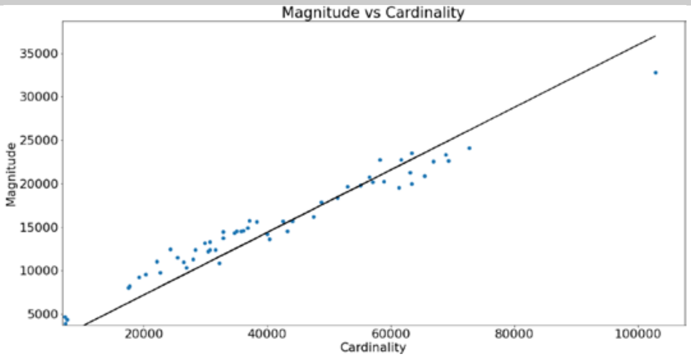
Before discussing the results, it is necessary to define two import subjects at this moment, cluster cardinality and cluster magnitude.

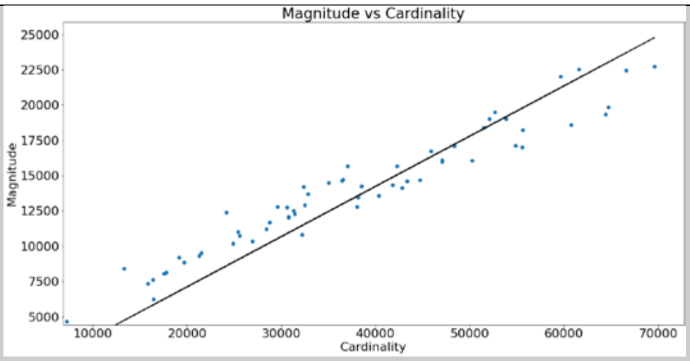
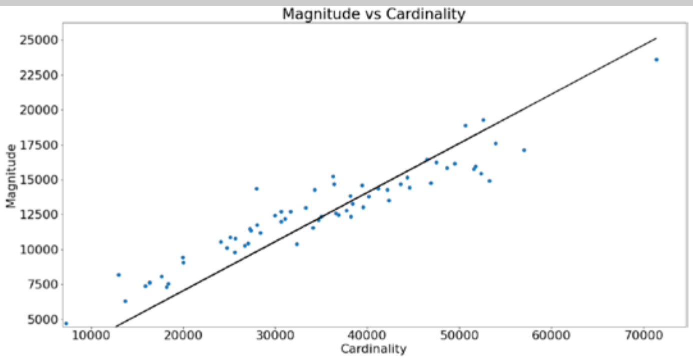
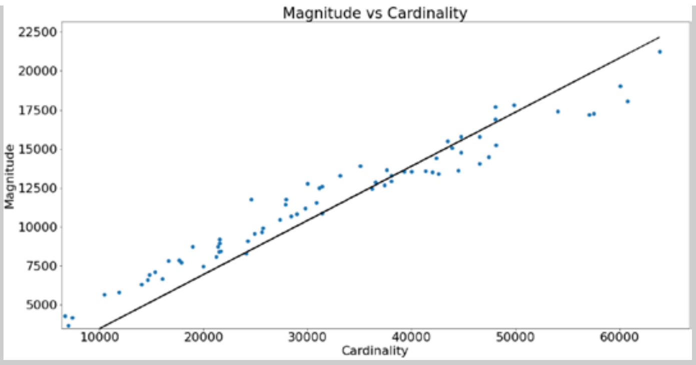
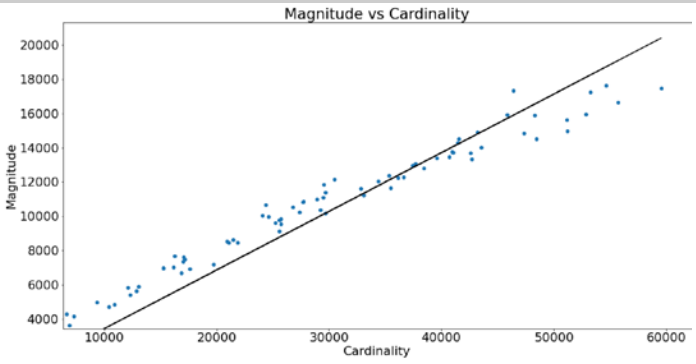
Cluster cardinality is the count of each observation under a cluster. It shows how big, or how populated a cluster is. It can be calculated by simply counting the number of data points in each cluster (Google-ML, 2022).

Cluster magnitude on the other hand, represents the total Euclidean distance from cluster centroids to each data point in the cluster. The distance for each point in the cluster is measured against the final cluster centroid and gets summed up to generate one single cluster magnitude value (Google-ML, 2022).

Using these two concepts, a simple analysis is conducted on the clusters to be able to determine whether the generated clusters follow a certain trend. According to Google Machine Learning Team, intuitively if a cluster has a high cardinality, meaning that it contains lots of data points inside, it should have a high magnitude as well when compared to other clusters (Google-ML, 2022). Thus, if cluster vs magnitude graph were to be plotted for each cluster, it should have a positive slope trend going on. Using this approach, the dataset at hand was clustered into several different number of clusters. KMeans algorithm has been applied to the dataset starting from 20 clusters with a step size of 5 until 80 clusters, and the cardinality and magnitude analyses have been stored for each clustering procedure. Below, you can find the resulting cluster vs magnitude graphs for each said cluster sizes.

#of Clusters	Cluster Cardinality vs Cluster Magnitude Graph
20	
25	
30	
35	

40	Magnitude vs Cardinality Magnitude Cardinality 40000 35000 30000 25000 20000 15000 20000 40000 60000 80000 100000
45	Magnitude vs Cardinality Magnitude Cardinality 40000 35000 30000 25000 20000 15000 20000 40000 60000 80000 100000
50	Magnitude vs Cardinality Magnitude Cardinality 27500 25000 22500 20000 17500 15000 12500 10000 20000 30000 40000 50000 60000 70000
55	Magnitude vs Cardinality Magnitude Cardinality 35000 30000 25000 20000 15000 10000 5000 20000 40000 60000 80000 100000

60	 Magnitude vs Cardinality Magnitude Cardinality 25000 22500 20000 17500 15000 12500 10000 7500 5000 10000 20000 30000 40000 50000 60000 70000
65	 Magnitude vs Cardinality Magnitude Cardinality 25000 22500 20000 17500 15000 12500 10000 7500 5000 10000 20000 30000 40000 50000 60000 70000
70	 Magnitude vs Cardinality Magnitude Cardinality 22500 20000 17500 15000 12500 10000 7500 5000 10000 20000 30000 40000 50000 60000
75	 Magnitude vs Cardinality Magnitude Cardinality 20000 18000 16000 14000 12000 10000 8000 6000 4000 10000 20000 30000 40000 50000 60000

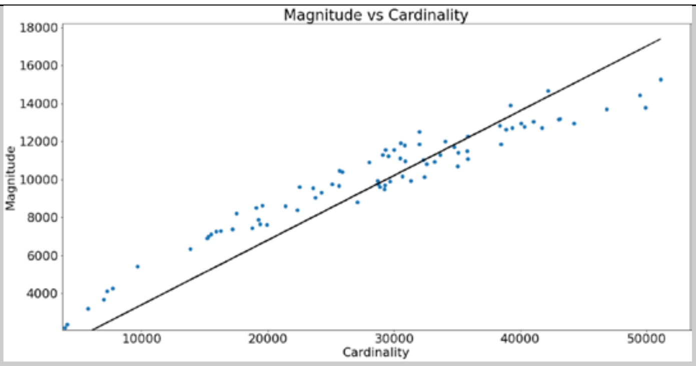
80	

Table 10. Cluster Cardinality vs Magnitude using 20 to 80 Clusters.

As can be seen from the table above, all N number of clusters used on the dataset proved to be satisfactory. To better distinguish and select the best number of clusters, an additional couple of metrics were used. Each cardinality vs magnitude graph has a linear black line fitted for each available data point on the graph. Using this fitted line, we can retrieve its R2, MAPE and MAE values and use them how good of a fit that line is. Below is the table that represents these values for each fitted black line.

# of Clusters	R2	MAPE	MAE
20	0.808313	0.53534	62564.18
25	0.844317	0.553557	51726.85
30	0.805116	0.563098	44181.69
35	0.877439	0.569334	38477.19
40	0.844687	0.5826	34180.59
45	0.820298	0.593695	30830.28
50	0.697554	0.601144	28067.03
55	0.894337	0.599864	25710.29
60	0.828177	0.609021	23769.44
65	0.743415	0.615159	22104.96
70	0.865745	0.614138	20634.93
75	0.874668	0.616866	19364.97
80	0.785674	0.619909	18264.23

Table 11. Cluster Cardinality vs Magnitude Fitted Line Metrics

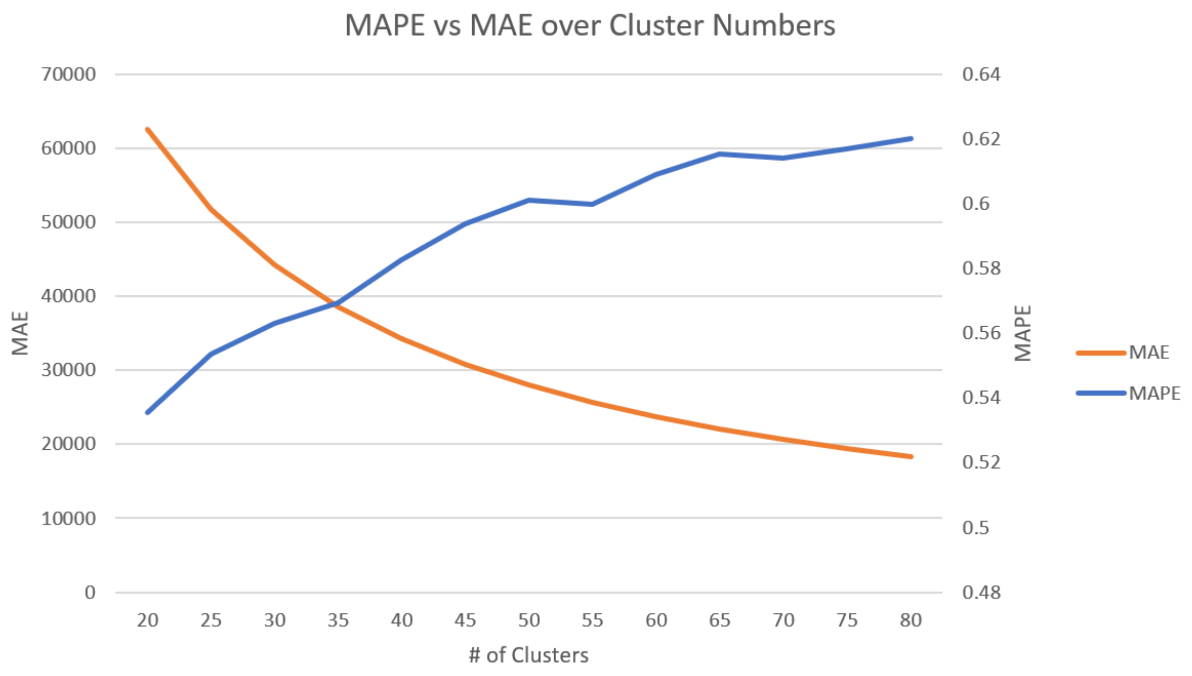


Table 12. MAPE vs MAE over different number of clusters.

By using two of the tables given above, an optimum number of clusters can be achieved using a simple elbow method. Around 50 clusters, MAE and MAPE values seem to reach their saturation points. If we were to choose a higher number of clusters, MAE value was going to drop slightly and MAPE would also increase slightly. Selecting the average of each end would prove to be satisfactory, thus 50 clusters were chosen to be the first number of clusters.

Since 50 clusters would not be nearly enough to define user behavior of listening to music, or even replicating the idea of a genre, a secondary clustering approach was used. Each of the 50 clusters the further clustered into 50 clusters, yielding a total of 2500 clusters. This way each abstract genre could better represent their sub-abstract genres better and more accurately. Since it was found out to be that 50 clusters were the optimal number of clusters for primary clustering, considering the time and manhour to find the secondary optimal clusters would cause the project to derail, it was convenient to choose 50 clusters as for the second clustering stage.

5.3. Content Based Recommender

According to Lops, main objective of content-based recommendation systems is to try to find similar items to a given item. It can be a user's movie likings, shopping habits, favorite books, or in this case, "contents" of a particular track (Lops, de Gemmis, & Semeraro, 2010). One of the processes used by a content-based recommender is to find these similarities using a vector space to calculate the shortest selected type of distances (Mishra, 2021).

There are lots of different similarity measures to determine whether two points in space are close to one another. The distances used will be explained in the next section in detail.

5.3.1. Similarity measures used

There were 3 different distance calculations used to find the closest, or "most similar" item to a given vector. The function that calculates and outputs the most similar items leverages sklearn's pairwise distances. This means that the 3 distance measures used can be changed in the future by just typing any similarity that is available in pairwise_distances method in sklearn. In the next section, the distances used in the recommendation system is explained in detail.

5.3.1.1. Cosine similarity

According to Xia, Euclidean distance is the most common metric used because of its intuitiveness and easiness. Sadly, it is highly susceptible to the magnitudes of each vector (Xia, Zhang, & Li, 2015). Finding out the cosine angle between the two points in space eliminates this disadvantage. Below is the formula for how to calculate this similarity measure between two points in space (Han, Kamber, & Pei, 2012).

$$similarity(x,y) = \cos(\theta) = \frac{\vec{x} \cdot \vec{y}}{||\vec{x}||||\vec{y}||}$$

$$||\vec{x}|| = \sqrt{x_1^2 + x_2^2 + x_3^2 + \cdots + x_n^2}$$

$$||\vec{y}|| = \sqrt{y_1^2 + y_2^2 + y_3^2 + \cdots + y_n^2}$$

Similarity is the measurement of how similar two points in space are to each other. A similarity value of 1 means that the points are almost identical. The distance, however, is calculated by the following formula, and has an intuitive explainability (Seelam, 2021).

$$Cosine\ distance = 1 - Cosine\ similarity$$

If similarity is close to 1, that means that two points are very similar. Distance on the other hand, should be close to zero if these two points are very similar to each other. Thus, 1 minus cosine similarity gives us the cosine distance.

5.3.1.2. L1 similarity

L1 similarity, or also known as L1 norm, Manhattan distance, Taxicab norm, Cityblock distance, is a simple distance measurement that originates from finding the block distance between two points on a city map (Harmouch, 2021). It is calculated by using the following formula (Opengenus, n.d.).

$$L1\ Norm = \|\vec{w}\| = |x_1 - y_1| + |x_2 - y_2| + \cdots + |x_n - y_n|$$

5.3.1.3. L2 similarity

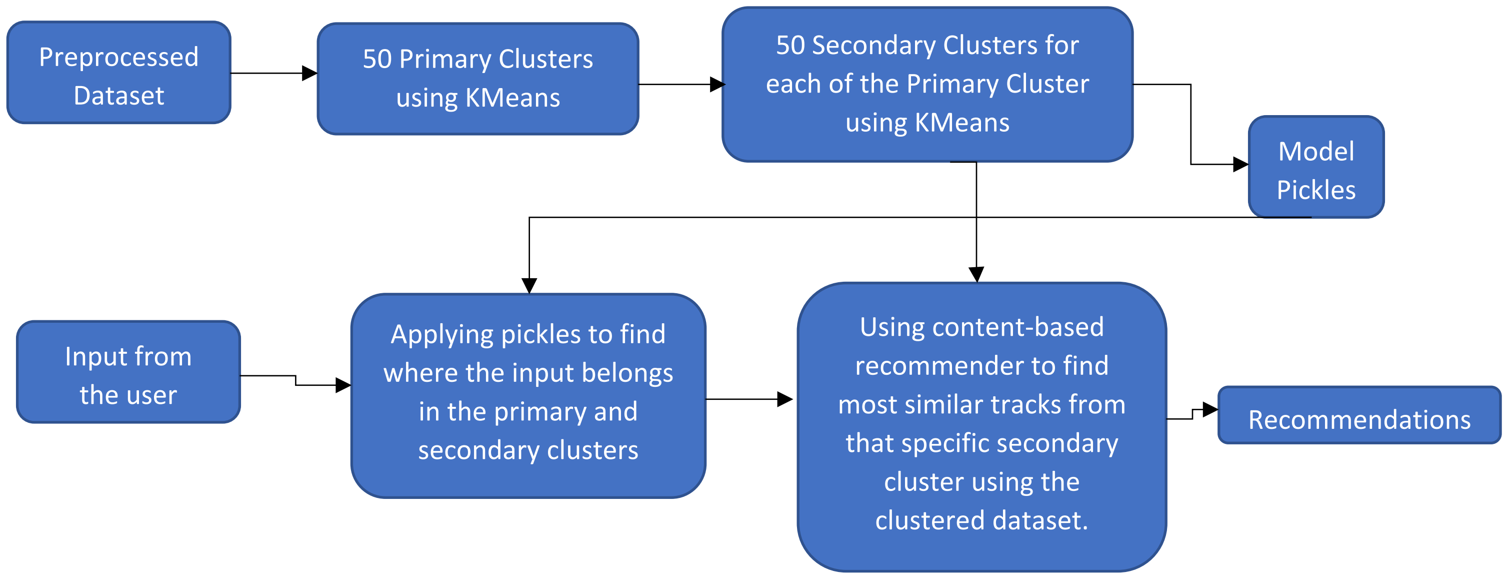
It is widely known as the Euclidean distance. To find the L2 distance between two points in space, the following formula is used (Wang, Zhang, & Feng, 2005),

$$distance(p, q) = \sqrt{\sum_{i=1}^{n} (q_i - p_i)^2}$$

Where p and q are two points in Euclidean n-space, qi and pi are Euclidean vectors initialized from the origin point, and n is the dimensions of the Euclidean space.

5.4. Recommending Using Primary and Secondary Clusters with Content Based Recommender

The process line below explains how a recommendation was generated using primary and secondary clusters with content-based recommender.


Preprocessed Dataset
50 Primary Clusters using KMeans
50 Secondary Clusters for each of the Primary Cluster using KMeans
Model Pickles
Input from the user
Applying pickles to find where the input belongs in the primary and secondary clusters
Using content-based recommender to find most similar tracks from that specific secondary cluster using the clustered dataset.
Recommendations

To summarize what the schema above explains is that, once we cluster the dataset into primary and secondary clusters, we later use these model pickles to find out where the input from the user (artist, track, or album) fits best. Using the tracks that are in the user's input cluster, we run the content-based recommender to find the most similar tracks in that specific cluster and rank them using cosine, l1 and l2 norms. Top N tracks become the final recommendations. If it is an album or top N tracks by an artist though, we find each tracks cluster separately and take the first m number of tracks that would return the desired number of unique recommended songs. For example, if the user wants to get recommendations based on their liking of Beethoven, we take the first – let us say- 10 tracks of the artist Beethoven available on Spotify, use model pickles to find their clusters, recommend each track from their respective cluster m tracks, and return -let's say- 20 tracks combined, which would -optimally- let us take 2 track recommendations from each top Beethoven tracks.

6. FINAL PRODUCT DEMO

The demo notebook can be found under “Final Product Demo.ipynb”. To give a quick oversight of what is inside the notebook, the most crucial cells have been shown below.

Using the connect_Spotify function available in the bdacapstone library, user can enter their client id and client secret to connect to their Spotify account. For security reasons, I did not show my client id and secret in this cell.

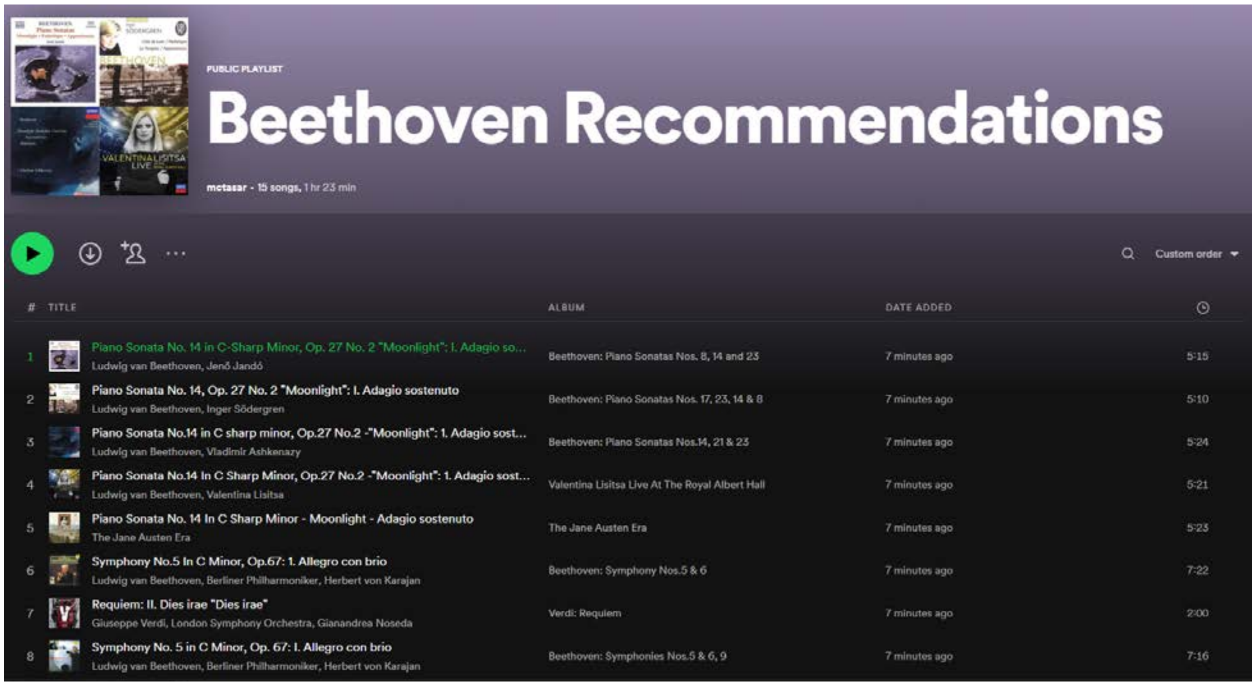
```
connection = bda.connect_Spotify(playlist_creator_mode=True)
```

The cell below is the heart of the recommendation. Using the create_playlist function, user can enter a desired playlist name to fill it with the search term recommendations.

```
bda.create_playlist(sp=connection,
                    playlist_name='Beethoven Recommendations',
                    search_term='Beethoven',
                    search_term_type='artist',
                    search_term_limit=5,
                    use_clusters=True,
                    return_n_recommendations=15,
                    return_qrcode=True)
```

At the end of this function, a QR code appears to take the user to the playlist generated on their Spotify account.

The recommendations for the artist “Beethoven” returned a recommendation list below.

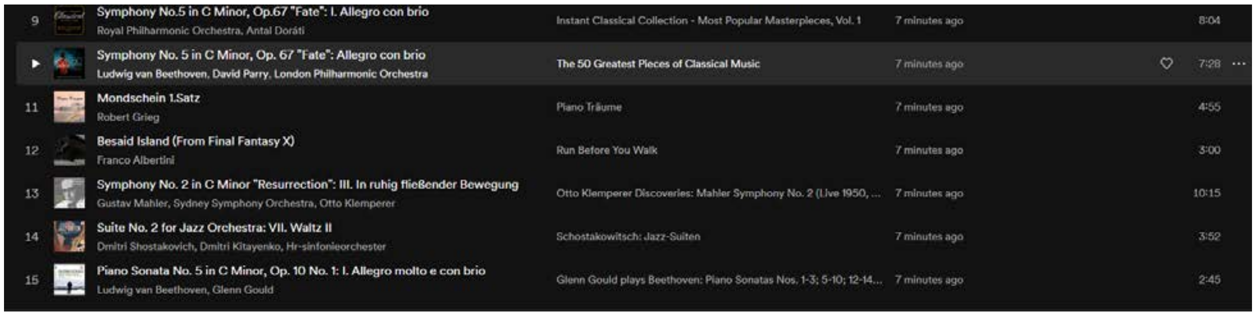


PUBLIC PLAYLIST

Beethoven Recommendations

metasar • 15 songs, 1 hr 23 min

Custom order



#	TITLE	ALBUM	DATE ADDED	🕒
1	Piano Sonata No. 14 in C-Sharp Minor, Op. 27 No. 2 "Moonlight": I. Adagio so... Ludwig van Beethoven, Jenő Jandó	Beethoven: Piano Sonatas Nos. 8, 14 and 23	7 minutes ago	5:15
2	Piano Sonata No. 14, Op. 27 No. 2 "Moonlight": I. Adagio sostenuto Ludwig van Beethoven, Inger Södergren	Beethoven: Piano Sonatas Nos. 17, 23, 14 & 8	7 minutes ago	5:10
3	Piano Sonata No.14 in C sharp minor, Op.27 No.2 -"Moonlight": 1. Adagio sost... Ludwig van Beethoven, Vladimir Ashkenazy	Beethoven: Piano Sonatas Nos.14, 21 & 23	7 minutes ago	5:24
4	Piano Sonata No.14 In C Sharp Minor, Op.27 No.2 -"Moonlight": 1. Adagio sost... Ludwig van Beethoven, Valentina Lisitsa	Valentina Lisitsa Live At The Royal Albert Hall	7 minutes ago	5:21
5	Piano Sonata No. 14 In C Sharp Minor - Moonlight - Adagio sostenuto The Jane Austen Era	The Jane Austen Era	7 minutes ago	5:23
6	Symphony No.5 In C Minor, Op.67: 1. Allegro con brio Ludwig van Beethoven, Berliner Philharmoniker, Herbert von Karajan	Beethoven: Symphony Nos.5 & 6	7 minutes ago	7:22
7	Requiem: II. Dies irae "Dies irae" Giuseppe Verdi, London Symphony Orchestra, Gianandrea Noseda	Verdi: Requiem	7 minutes ago	2:00
8	Symphony No. 5 in C Minor, Op. 67: I. Allegro con brio Ludwig van Beethoven, Berliner Philharmoniker, Herbert von Karajan	Beethoven: Symphonies Nos.5 & 6, 9	7 minutes ago	7:16
9	Symphony No.5 in C Minor, Op.67 "Fate": I. Allegro con brio Royal Philharmonic Orchestra, Antal Doráti	Instant Classical Collection - Most Popular Masterpieces, Vol. 1	7 minutes ago	8:04
▶	Symphony No. 5 in C Minor, Op. 67 "Fate": Allegro con brio Ludwig van Beethoven, David Parry, London Philharmonic Orchestra	The 50 Greatest Pieces of Classical Music	7 minutes ago	7:28
11	Mondschein 1.Satz Robert Grieg	Piano Träume	7 minutes ago	4:55
12	Besaid Island (From Final Fantasy X) Franco Albertini	Run Before You Walk	7 minutes ago	3:00
13	Symphony No. 2 in C Minor "Resurrection": III. In ruhig fließender Bewegung Gustav Mahler, Sydney Symphony Orchestra, Otto Klemperer	Otto Klemperer Discoveries: Mahler Symphony No. 2 (Live 1950, ...	7 minutes ago	10:15
14	Suite No. 2 for Jazz Orchestra: VII. Waltz II Dmitri Shostakovich, Dmitri Kitayenko, Hr-sinfonieorchester	Schostakowitsch: Jazz-Suiten	7 minutes ago	3:52
15	Piano Sonata No. 5 in C Minor, Op. 10 No. 1: I. Allegro molto e con brio Ludwig van Beethoven, Glenn Gould	Glenn Gould plays Beethoven: Piano Sonatas Nos. 1-3; 5-10; 12-14...	7 minutes ago	2:45

7. FUTURE WORKS

According to Google ML Team, the curse of dimensionality is one of the main disadvantages when it comes to similarity-based recommendation systems (Google, 2022). To increase the overall performance of the resulting recommendations, a PCA or Autoencoder can be applied to reduce the dimensions at hand. After reducing the number of features to 2 or 3, clustering process can be applied again and using the dimension reduction decoder and encoder.

Another improvement to the system at hand could be implementing a collaborative approach to the playlists themselves. Finding similar playlists with each other can prove useful when another album or top 10 songs of an artist enters as an input. This way it could better represent user listening behavior by making a connection between each playlist and their track content.

8. REFERENCES

Dannenberg, R. B. (2010). Style in Music. In S. Argamon, K. Burns, & D. Scholomo, *The Structure of Style: Algorithmic Approaches to Understanding Manner and Meaning* (pp. 45-58). Berlin: Springer-Verlag.

Google. (2022). *developers.google.com*. Retrieved from https://developers.google.com/machine-learning/clustering/algorithm/advantages-disadvantages

Google-ML. (2022). *developers.google.com*. Retrieved from https://developers.google.com/machine-learning/clustering/interpret

Han, J., Kamber, M., & Pei, J. (2012). Getting to Know Your Data. *Data Mining (Third Edition)*, 29. doi:https://doi.org/10.1016/B978-0-12-381479-1.00002-2

Harmouch, M. (2021, March 14). *towardsdatascience.com*. Retrieved from 17 types of similarity and dissimilarity measures used in data science: https://towardsdatascience.com/17-types-of-similarity-and-dissimilarity-measures-used-in-data-science-3eb914d2681

Lops, P., de Gemmis, M., & Semeraro, G. (2010). Content-based Recommender Systems: State of the Art and Trends. *Recommender Systems Handbook*, 73. doi:DOI: 10.1007/978-0-387-85820-3_3

Ludewig, M., Kamehkhosh, I., Landia, N., & Jannach, D. (2018). Effective Nearest-Neighbor Music Recommendations. *In Proceedings of Proceedings of the ACM Recommender Systems Challenge 2018 (RecSys Challenge '18)*, 1-6. doi:https://doi.org/10.1145/3267471.3267474

Mishra, U. (2021, May 29). *What is a Content-based Recommendation System in Machine Learning?* Retrieved from analyticssteps.com: https://www.analyticssteps.com/blogs/what-content-based-recommendation-system-machine-learning

Opengenus. (n.d.). *iq.opengenus.org*. Retrieved from Manhattan distance: https://iq.opengenus.org/manhattan-distance/

Piech, C. (2013). *www.stanford.edu*. Retrieved from https://stanford.edu/~cpiech/cs221/handouts/kmeans.html

Seelam, S. (2021, May 26). *medium.com*. Retrieved from Machine Learning Fundamentals: Cosine Similarity and Cosine Distance: https://medium.com/geekculture/cosine-similarity-and-cosine-distance-48eed889a5c4

Spotify. (2022). Retrieved from developer.spotify.com: https://developer.spotify.com/documentation/web-api/reference/#/operations/get-several-audio-features

Spotify. (n.d.). *aicrowd.com*. Retrieved from Spotify Million Playlist Dataset Challenge: https://www.aicrowd.com/challenges/spotify-million-playlist-dataset-challenge

Volkovs, M., Rai, H., Cheng, Z., Wu, G., Lu, Y., & Sanner, S. (2018). Two-stage Model for Automatic Playlist Continuation at Scale. *RecSys Challenge '18: Proceedings of the ACM Recommender Systems Challenge 2018*, 1-6. doi:https://doi.org/10.1145/3267471.3267480

Wang, L., Zhang, Y., & Feng, J. (2005, June 20). On the Euclidean distance of images. *IEEE Transactions on Pattern Analysis and Machine Intelligence, 27*(8), 1334-1339. doi:10.1109/TPAMI.2005.165

Wu, K., Anand, V., & Sydykov, K. (2018). Don’t Stop the Music: Playlist Continuation with Autoencoders.

Xia, P., Zhang, L., & Li, F. (2015). Learning similarity with cosine similarity ensemble. *Information Sciences*, 40. doi:https://doi.org/10.1016/j.ins.2015.02.024