Creating a Spotify Music Recommender System

Ananya Garg, Tanya Beri, Aida Mohasesi, Angelique Rubio, Diane Jang, Valeria Galvan

March 2023

Abstract

Spotify is known as the best music streaming application for several reasons, but one of the main reasons is their music recommendation system. We are constantly searching for new music, and one of the reasons Spotify's recommendation algorithm is so successful is because it gives you new music based on the music you already listen to. Therefore, our group decided to create our own music recommendation system that can take a playlist as input and output five recommended songs based on the songs already in the playlist. We used the LastFM data set which consisted of around 1.6 million songs including their genres and combined with data using the Spotify Web API which contained information about audio features which limited our final dataset to around 100,000 songs. We created our recommender system using cosine similarity matrices, calculating the similarity between audio features in songs in the inputted playlist, and songs in our dataset. Our algorithm compares the features of each input song from the playlist to each song in our database and creates a matrix with cosine similarity values. Outputting the top five songs with the highest cosine similarity scores (which indicate better recommendations), we were then able to successfully give music recommendations. We tested the system on 15 Spotify generated playlists and each of our group members playlists too, and rated the results outputted by our algorithm – yielding an average rating of 3.83 out of 5.

1 Introduction & Research Questions

User-focused platforms are constantly looking for ways to individualize and improve the user experience. Spotify stands out among these platforms, specifically because of their music recommendation system. One of the reasons Spotify is so popular is because it recommends music based on music the user already enjoys listening to, giving users the opportunity to expand their music library without the extra work of looking for new music. Users are given individually curated playlists they enjoy, and Spotify stays among the top music streaming services. As the demand for creating systems that center the user experience grows, we wanted to see if we could apply statistics to develop a Spotify recommender system .

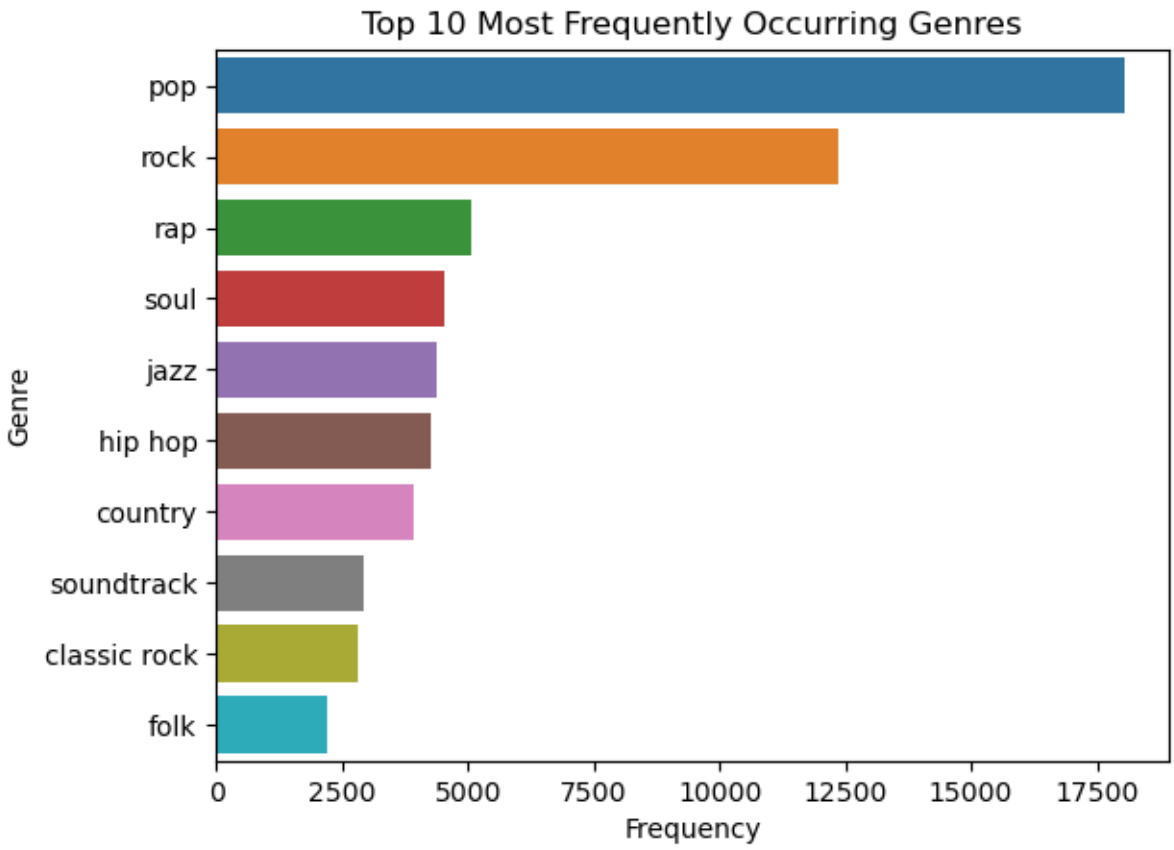
2 Data Cleaning & Preparation

To begin, we were inspired to begin this project after observing the Last FM dataset. This dataset included tracks and genres of millions of songs from Spotify up to the year 2020. Since we were working with a substantially large dataset, we encountered a few issues when attempting to run our exploratory data analysis. To remedy this, we chose to filter the Last FM dataset by a different dataset containing top 10,000 artists from the year 2020. Doing so allowed us to more efficiently analyze the data, as it reduced the size of the data from millions of songs to around 100,000. We also reduced the dataset by removing columns with irrelevant information, such as timestamp and user id. After reducing the size of our dataset, we merged it with information from Spotify Web API. Spotify Web API allowed us to access relevant metadata *(Web API | Spotify for Developers, 2024)* including audio features for each

song, such as: danceability, energy, acousticness, instrumentalness, liveness and valence. These audio features formed the basis for how our program assessed which songs to recommend to the user.

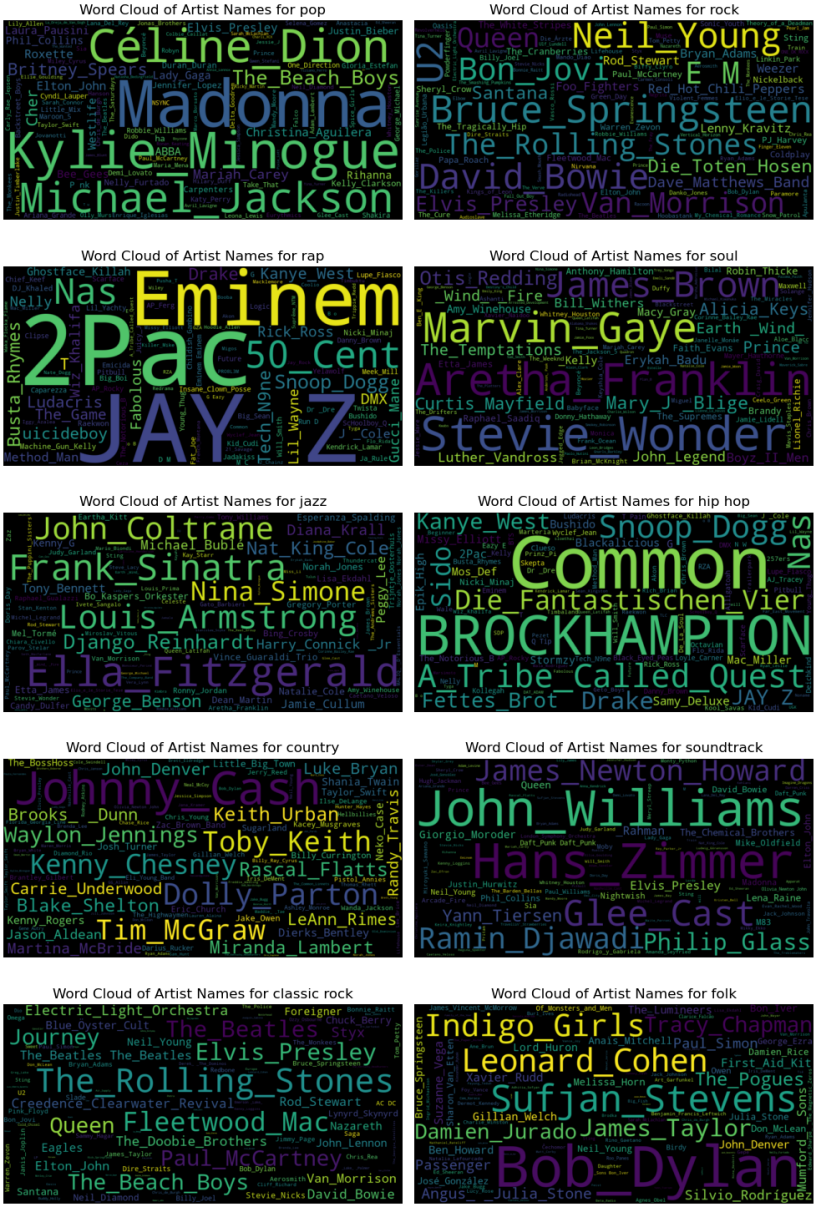
3 Exploratory Data Analysis

3.1 Top Genres by Frequency



Above, we see that Pop, Rock, and Rap are the most frequently occuring genres for songs in our database, with Pop being most frequent by a large margin.

3.2 Word Cloud for Top Artists in Each Genre



Above is a word cloud of the top artist names for each genre. Some big names for the 'Rap' word cloud are 2Pac, Jay Z, and Eminem - which are the most popular artists in the 'Rap' genre around the year 2020. This visualization provides insights into which artists are most likely to appear in recommendations (since they appear most frequently). Since different genres showed the highest difference in audio features, we produced separate word clouds for artists in each genre.

3.3 Summary Statistics for Audio Features

Summary Statistics of Audio Features

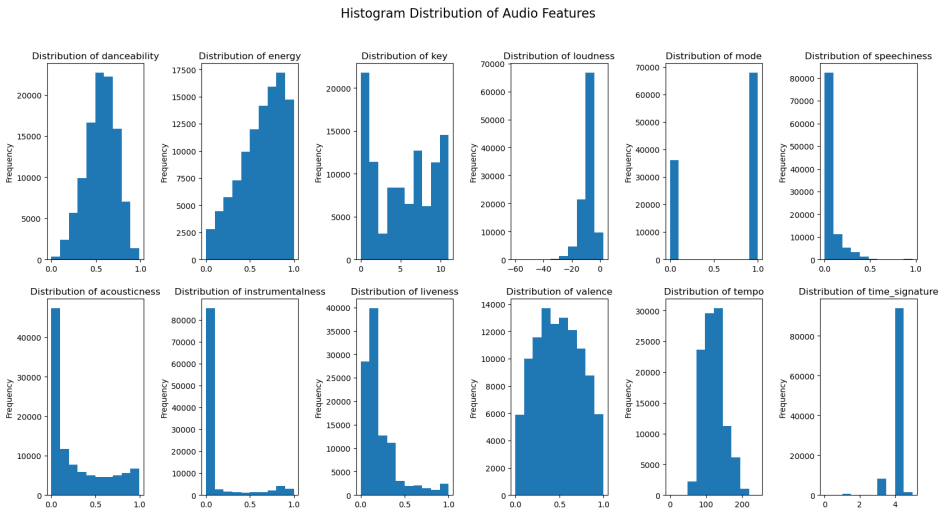
	danceability	energy	key	loudness	mode	speechiness	acousticness	instrumentalness	liveness	valence	tempo	time_signature
mean	0.557711	0.629843	5.267932	-8.437435	0.652408	0.083725	0.282203	0.112518	0.216792	0.490136	120.666692	3.912886
std	0.171585	0.246407	3.56234	4.612785	0.476208	0.099081	0.315891	0.260392	0.197679	0.251721	29.253511	0.387838
min	0.0	0.0	0.0	-60.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
25%	0.446	0.46	2.0	-10.459	0.0	0.0343	0.0169	0.0	0.0969	0.287	97.7855	4.0
50%	0.569	0.672	5.0	-7.255	1.0	0.0459	0.134	4.9e-05	0.133	0.485	119.997	4.0
75%	0.682	0.834	8.0	-5.259	1.0	0.0818	0.508	0.0147	0.276	0.692	138.4335	4.0
max	0.987	1.0	11.0	2.309	1.0	0.964	0.996	0.999	1.0	0.996	244.08	5.0

Above are summary statistics for the different audio features we looked at.

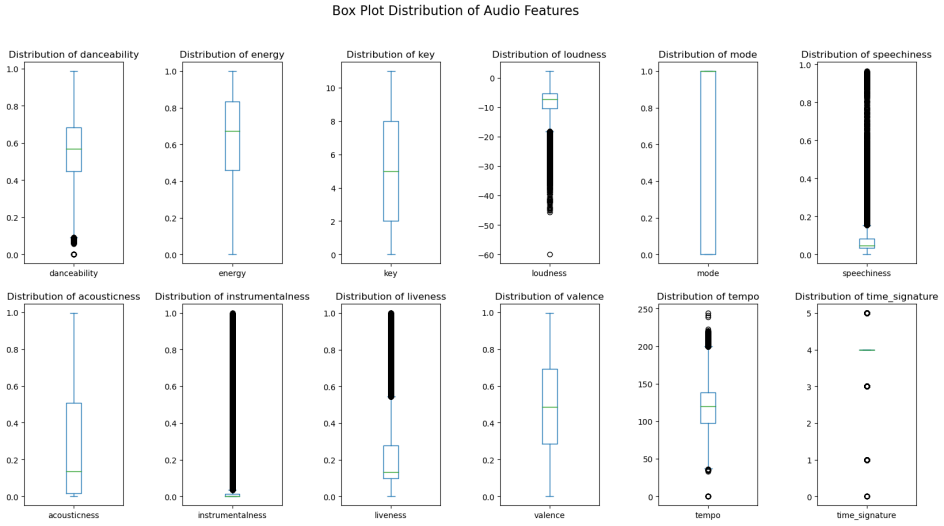
Features Description:

- 'Danceability' is a measure of how suitable a track is for dancing based on a combination of musical elements, with a value of 0.0 being least danceable and 1.0 being most danceable.
- 'Energy' is measured from 0 to 1 and is a measure of intensity and activity (where energetic tracks feel fast and loud).
- 'Key' represents the key the songs is in, where 0 = C, 1 = C#, 2 = D, and so on.
- 'Loudness' is the average decibels of a track, typically ranging between -60db and 0 db.
- 'Mode' indicates whether a track is in major or minor where major is represented by 1 and minor represented by 0.
- 'Speechiness' is a value between 0 and 1 and detects the presence of spoken words in a track, where if a track is more speech-like (talk show/audio book), the value is closer to 1.
- 'Acousticness' is a confidence measure between 0 and 1 of whether the track is acoustic.
- 'Instrumentalness' is a value between 0 and 1 which predicts whether a track contains no vocals, where songs with values closer to 1 represent have greater likelihood of containing no vocal content
- 'Liveness' detects presence of an audience in the recording, with higher values representing increased probability that the track was performed live.
- 'Valence' is a measure from 0 to 1 describing the positiveness of a track, with higher values representing songs that sound more positive
- 'Tempo' is overall estimated tempo of a track in beats per minute (BPM), giving the speed of a song.
- 'Time_signature' specifies how many beats are in each bar, ranging from 3 to 7 which indicates time signatures of 3/4 to 7/4.

Looking at the summary statistics, we see that tempo, loudness, and key have the largest standard deviations, which makes sense since they have the largest range of values possible. Between the audio features that are valued between 0 and 1, we see that acousticness, instrumentalness, and valence are the features with the largest standard deviations.

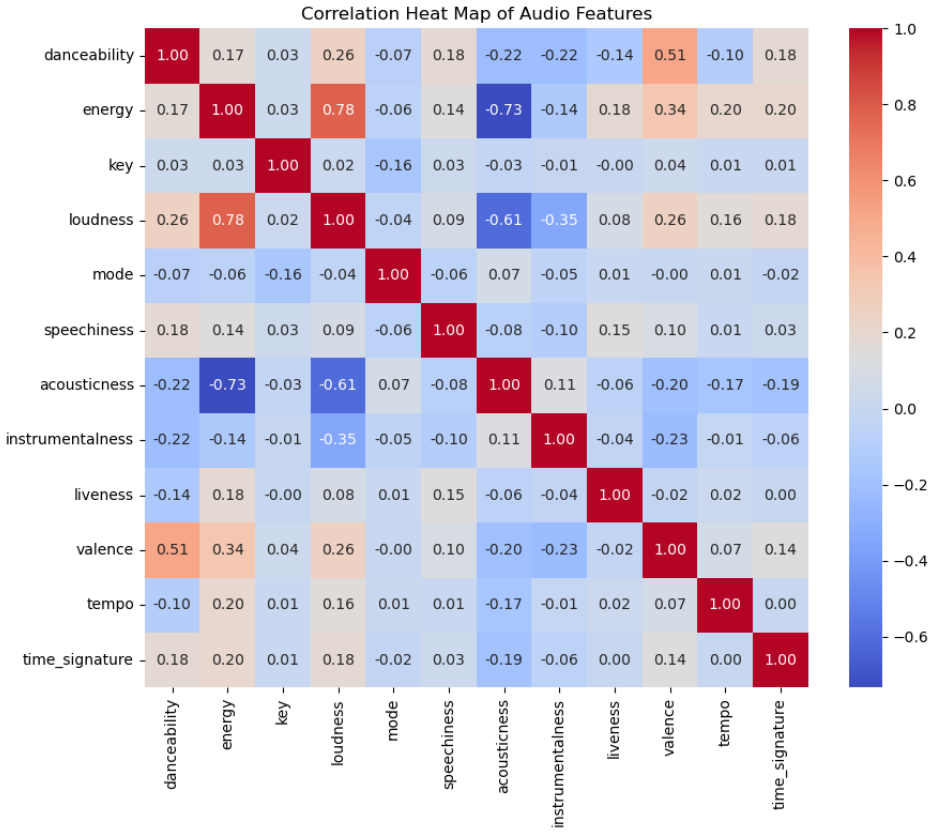


Further looking at the histogram distributions of individual audio features above, we see that energy is skewed to the left, with a higher frequency of songs in our database having higher energy. Similarily, loudness and frequency is also skewed to the left, where more songs are louder and at a faster time signature. Looking at the speechiness, it is extremely right-skewed, which indicates that the majority of songs are less speech-like (actual songs instead of podcasts or talk shows). This is similar to the histograms for 'instrumentalness', 'liveness' and 'acousticness', which are also skewed right, and this indicates that more songs tend to be less acoustic, lesser probability of being performed live, and have higher likelihood of having vocal content. Finally, the distribution for 'valence', 'tempo', and 'danceability' are near normal, which indicate that the songs in our database have a range of values of positivity, speed, and danceability.



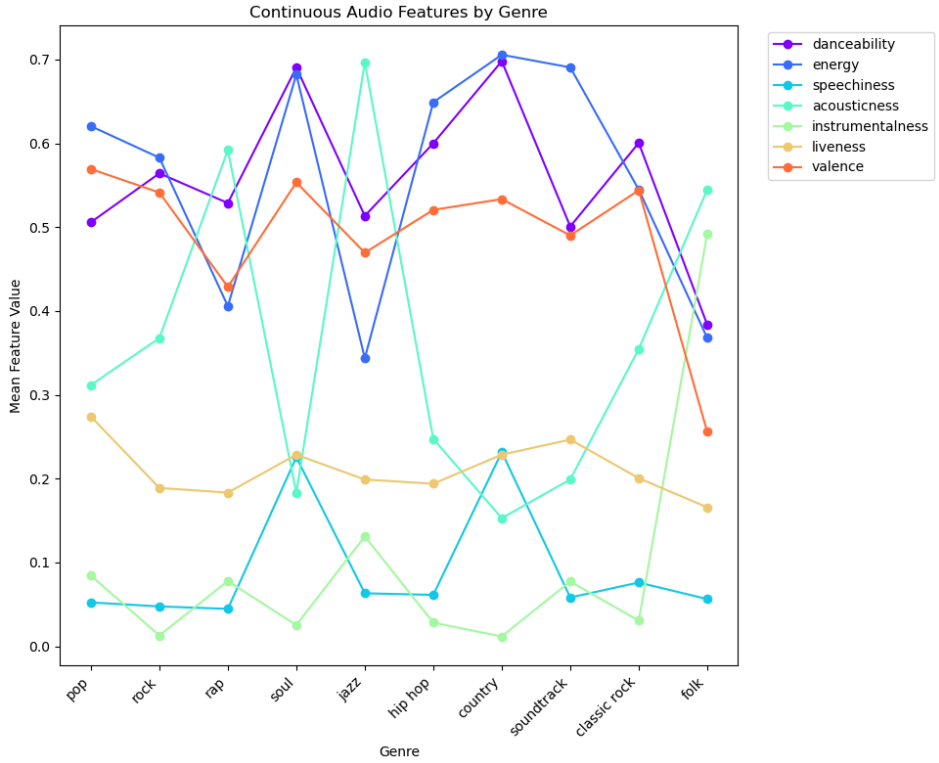
If we look at the boxplot distributions for audio features, we see that the values for instrumentalness, loudness, speechiness, and liveness all have many outliers. For instrumentalness, speechiness, and loudness, it looks like the majority of songs have low values for all of these features. Corresponding to the histogram distribution above, for loudness, it seems that the majority of songs tend to be louder. It is interesting to note that the distribution of valence looks to be very evenly spread, with a median value occuring somewhere close to 0.5, which shows the variety in positivity in songs and artists in our database. The distributions of danceability and energy are also skewed towards higher values, which can be compared to the skewed distribution of loudness values - this indicates that the majority of songs in our database (being made right now by the most popular artists) tend to have higher energy and loudness, which appeal more to listeners.

3.4 Correlation between Audio Features



We see highest correlations between acousticness and energy where songs with higher energy tend to be less acoustic - which makes sense. We also see a high correlation between loudness and energy. This also makes sense given that energetic songs tend to be louder. Interestingly, we don't see a high correlation between danceability and energy or loudness, given that highly danceable songs tend to be more lively. It is also interesting that we don't see a significant correlation between tempo and energy, given that more energetic songs would theoretically be faster. We see the highest correlation of valence with danceability with a correlation of 0.51, indicating that more positive songs tend to be more danceable, but this is not a strong correlation value.

3.5 Mean Continuous Audio Feature Values

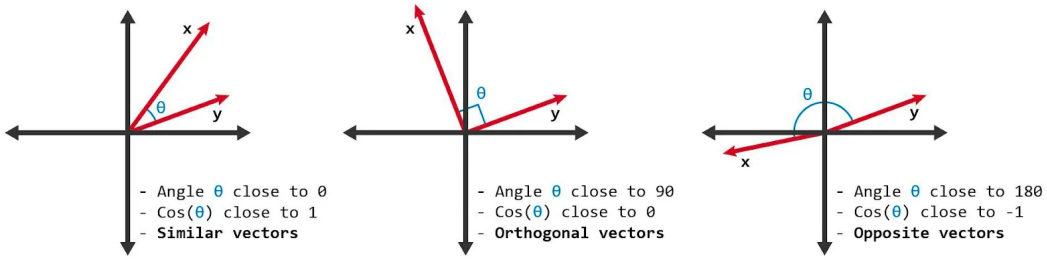


The plot above shows the mean audio feature values for the top 10 most frequently occuring genres in our database. We see highest danceability, energy, instrumentalness and speechiness scores for the Soul and Country genres. The Folk genre had lowest mean valence (positivity), danceability, and energy values.

4 Methods

After our initial data cleaning and merging, we created a multi-argument recommender function that assesses the similarity between different audio features of songs on the input and output playlists. The first component of the recommender function is a pre-processing function that extracts attributes from each of the songs in the playlist you input. Relevant components such as track URI, track name, artist name, and all audio features are then stored in a data frame and categorized as our input argument. Next, there is a final data argument. This final data argument contains information on over 104,000 song tracks with the same previously mentioned song components and is filtered to ensure the user does not receive a recommendation for a song that is already on their input playlist.

The backbone of our recommendation system comes from a function that calculates the cosine similarity matrix between the audio features for the input playlist songs and final data songs. Cosine similarity is a measure used to determine the similarity between vectors (Chang, 2023), which is calculated by taking the dot product of each vector and dividing it by the product of their magnitudes. Essentially, the smaller the angle between two vectors, the closer the cosine of that angle is to 1 and the higher the cosine similarity.



We applied this calculation to our project by assigning a vector to each audio feature of each track from both the input and final playlist. For example, a song with high danceability, instrumentalism, and tempo is given a vector that reflects those features, while a song with low danceability, instrumentalism, and tempo is given another vector to reflect those features. Once each song is assigned to corresponding vectors, the function begins to compute the cosine similarity matrix. The matrix is created by comparing the vector of the i-th input playlist song and the j-th final playlist song and storing that value in a dataframe (Chang, 2023). The cosine similarity matrix is very similar to a correlation matrix in format and understanding. Lower values indicate less similarity and therefore a bad recommendation, and higher values indicate higher similarity, and therefore a good recommendation, For our recommender system, we took in a user's playlist and outputted the 5 songs with the highest cosine similarity score.

5 Results

We successfully applied statistics to construct a Spotify recommender system. Our evaluation involved testing our recommender system on 15 Spotify generated playlists followed by individual playlists from each team member. Following this, each team member rated the recommendations given to them on a scale of 1 to 5, with 1 indicating the lowest rating and 5 indicating the highest. The obtained ratings varied, with the lowest rating being a 2, and the highest rating being a 5, all of which averaged our recommender system to rating of 3.83. It is important to acknowledge the limitations of our project. While the average rating suggests a moderate level of success, we need to further explore the factors that contributed to the variation in ratings and address any potential improvements to our recommender system.

6 Conclusions

In conclusion, our project aimed to explore the application of statistics in developing a Spotify recommender system. Through our efforts, we successfully developed a system capable of recommending songs based on audio features similar to the inputted playlist. Following rigorous testing on 15 Spotify-generated playlists and subsequently on each team member's personal playlists, we obtained promising results, with an average user rating of 3.83 with the lowest user rating being a 2 and the highest user rating being a 5. While our system shows potential, there are areas for enhancement, including expanding the song database and integrating natural language processing for lyrics analysis, as suggested earlier. Nonetheless, our project serves as a valuable foundation, fostering a deeper understanding of Spotify's mechanisms and providing the opportunity to build our own recommendation systems.

7 Discussion & Limitations

One of the main limitations for this project is that we only considered the effect of audio features when generating song recommendations. Thus, the cosine similarity matrix, used in our recommender system, fails to account for additional pertinent features such as lyrics and overall tone of songs. For future improvements, it would be useful to take into account lyrics using natural language processing

methods to be able to make more accurate recommendations for more niche playlists. Additionally, in the process of collecting our song tracks catalog data, we decided to limit the number of songs collected to only songs produced by the top 10,000 artists in the United States in order to decrease the dimensionality of the dataset. Therefore, the bank of songs that we were recommending from was limited, meaning that not all genres and types of songs were available for recommendations. As a result, some specific types of songs may not have had good similarity matches from the song bank due to the limited songs that were queried.

It is also important to note that in our catalog dataset, there were several versions of the same song such as dance mixes, live versions, acoustic versions, and remixes to name a few. To prevent our recommender system from recommending different versions of the same song, we attempted to keep only the original version for each song by keeping only the songs with unique URIs. However, due to time constraints, not all types of versions in the catalog dataset could be accounted for and sometimes our recommender does indeed recommend multiple versions of the same song.

Finally, working with the Spotify Web API presented its own challenges. Because our web app was filed under development mode, when making requests to the Spotify Web API, our accounts would often get rate-limited, meaning that we had made too many requests to the API within a given time frame. To bypass this issue, we either had to create new Spotify accounts or wait for an extended period of time in order to reset our access to the API along with our requests history.