Analyzing trends in Spotify listening data

Michael Maldonado

Abstract

With this project I attempted to analyze music listening data provided by Spotify containing every track played since account creation. The focus was on analyzing number of track plays per certain times and days. Spotify allows other programmers to access their API to search for specific track features that describe the mood and emotions of a specific track. With all these tools and data sources I discovered positive correlations related track energy/happiness and an emphasis on repeat plays for specific tracks and artists. With this report you will be able to see how to analyze and process Spotify listening data.

Introduction

Spotify is one of the most popular streaming services in the world and because of its size and userbase, it collects massive amounts of data about songs played. Every year Spotify gives a year-end report to its users and describes the top genres and artists that were played. Unfortunately, that is as far as the analysis goes when describing your listening data, and because of this, other programmers have made analysis tools that try to bridge the lack of information. Spotify allows users to request information about tracks, artists, and albums, though its API, returning information that Spotify itself uses in its end of year analysis. Since this information is freely available to other developers, it is easier to go further in the analysis. What I have tried to do in this project is analyze the listening data from my own Spotify account to find trends in all the songs that I have played since my account creation in late 2017, and the data was requested directly from Spotify.

Materials and Methods

The starting point for this project was the listening data requested from Spotify, Spotify categorizes listening data into 3 levels with levels 2 and 3 containing the most data per song play. I specifically asked a Spotify representative for level 2/3 data and weeks later received a zip archive with my data inside. The folder had various JSON files but the most important files, and the ones containing the actual song plays, were the endsong_X.JSON files, where X is a number that represents the position of the file out of all the endsong files. For the rest of this project I will be using Python with Jupyter

Notebook to manipulate and display the data. I loaded in these endsong JSON files into a pandas dataframe and began cleaning the data, starting with dropping unnecessary features in the dataset. Originally the data was represented with 21 features (Figure 1) and after dropping the unnecessary data it was reduced to 9.


```
"ts": "YYY-MM-DD 13:30:30",
"username": "_____",
"platform": "_____",
"ms_played": _____,
"conn_country": "_____",
"ip_addr_decrypted": "___.___.___.___",
"user_agent_decrypted": "_____",
"master_metadata_track_name": "_____,
“master_metadata_album_artist_name:_____”,
“master_metadata_album_album_name:_____",
“spotify_track_uri:_____”,
"episode_name": _____,
"episode_show_name": _____,
“spotify_episode_uri:_____”,
"reason_start": "_____",
"reason_end": "_____",
"shuffle": null/true/false,
"skipped": null/true/false,
"offline": null/true/false,
"offline_timestamp": _____,
"incognito_mode": null/true/false,
```


Figure 1:
an example of one endsong streaming entry before preprocessing

The changes were: changing the "ts" field to a Python datetime object (from UTC to US/Central), and dropping user information like IP addresses and Connection Country for the device. The final dataframe was reduced to a size of 151297 rows × 9 columns (Figure 2) meaning that there were a total of 151297 tracks recorded since my account creation. The most important features for this project were "ts" which represented when the track was played, and "uri" which is the tracks Spotify ID which allows you to access the Spotify API for information about its individual features. The next step was to get information about every single track based on its URI, the information Spotify provides is called "track_features" and includes values that Spotify uses to describe a tracks energy, accousticness and other features.

Figure 2:

an excerpt of 5 streaming entries after preprocessing

	ts	ms_played	track_name	artist_name	album_name	uri	reason_start	reason_end	shuffle
0	2018-12-31 08:22:13+00:00	6357	Electrify	Shiro Schwarz	Street Beat	spotify:track:0EujJrnoynjEagj1TWclkw	clickrow	fwdbtn	True
1	2021-03-30 00:27:25+00:00	11080	Oh My...	Toby Fox	UNDERTALE Soundtrack	spotify:track:6CBsGV6sv811sRb4sNBmcl	trackdone	fwdbtn	False
2	2018-06-05 02:52:25+00:00	232091	Orbit (feat. Richard Caddock)	WRLD	Orbit (feat. Richard Caddock)	spotify:track:3BkLpheTPPxLrAvi0dHId8	trackdone	logout	False
3	2021-04-25 21:27:05+00:00	2420	The Poetry Of August	Lamp	The Poetry Of August	spotify:track:38mF9ahDeZLbI1CTxbSo3V	clickrow	endplay	False
4	2019-09-12 14:59:47+00:00	25569	Fox	Shirobon	Dimensions	spotify:track:2KORn4JKTJvUMu8E3EnT5K	trackdone	backbtn	False

This "track_features" data was requested using the Spotify URI of all the tracks and appended into a new pandas dataframe (Figure 3). The process of querying Spotify required the use of the Spotipy Python package, and iterating through batches of 100 Track URIs and getting a row for the new Dataframe. This new dataframe of every single track's features had to have its null and Nan values filled in with zeros and standardized using ScikitLearn's StandardScaler (Figure 4) to allow the Principal Component analysis and Clustering to properly Converge.

Figure 3:

an excerpt of 5 track_features entries before Standardization

	danceability	energy	key	loudness	mode	speechiness	acousticness	instrumentalness	liveness	valence	tempo	duration_ms
0	0.908	0.763	10.0	-4.525	0.0	0.0430	0.060600	0.35500	0.0332	0.935	120.019	198426.0
1	0.000	0.400	9.0	-14.282	0.0	0.0000	0.991000	0.67800	0.3460	0.000	0.000	14483.0
2	0.571	0.620	6.0	-5.791	1.0	0.0267	0.005440	0.00000	0.0923	0.625	150.012	240000.0
3	0.356	0.414	9.0	-5.853	1.0	0.0293	0.893000	0.00678	0.1330	0.095	143.135	367760.0
4	0.579	0.971	10.0	-3.994	0.0	0.0633	0.000073	0.98000	0.2370	0.558	138.010	316522.0

Figure 4:

an excerpt of 5 track_features entries after Standardization

	danceability	energy	key	loudness	mode	speechiness	acousticness	instrumentalness	liveness	valence	tempo	duration_ms
0	2.098007	0.376665	1.251233	0.663106	-1.218532	-0.388433	-0.579291	0.314983	-1.036051	1.783132	-0.113029	-0.504763
1	-3.539666	-1.270408	0.972447	-1.378005	-1.218532	-0.966895	2.650331	1.241598	0.947208	-1.805114	-4.263473	-2.469016
2	0.005611	-0.272182	0.136092	0.398266	0.820660	-0.607710	-0.770763	-0.703434	-0.661337	0.593446	0.924176	-0.060811
3	-1.329301	-1.206884	0.972447	0.385296	0.820660	-0.572734	2.310152	-0.683984	-0.403285	-1.440533	0.686359	1.303487
4	0.055282	1.320442	1.251233	0.774188	-1.218532	-0.115345	-0.789393	2.107969	0.256111	0.336320	0.509128	0.756337

The two dataframes (all_songs and track_features) were used from now on to do analysis on each track.

Results and Discussion

The first part of the analysis will begin with looking at the distributions of tracks played per year cumulatively. This will show how many tracks were played on average and how they changed per each year.

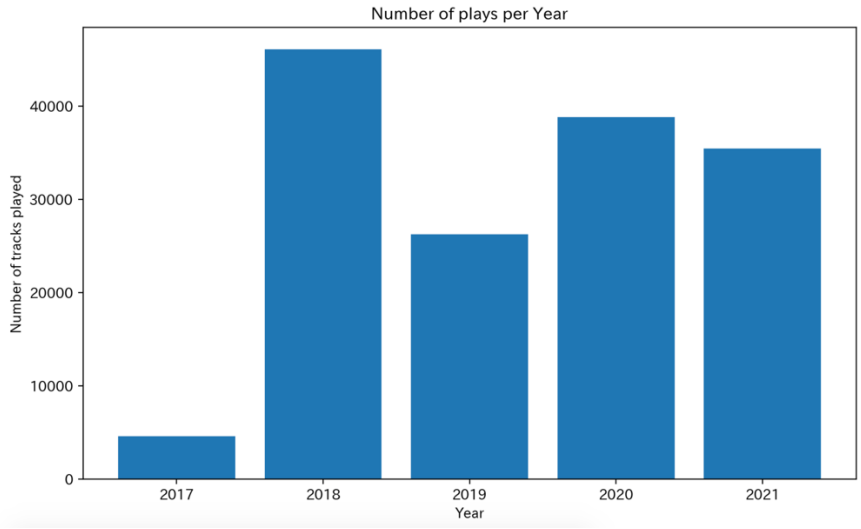

Figure 5:
Cumulative plays per year

We can see since the account's inception in 2017 (Figure 5) the Average of track plays per year equals out to 30259 plays. When we look closer at tracks played, especially the number of tracks played per time of day and week, there is a pattern in the heatmap that can show us the estimated work/sleep hours in a day that a user does not listen to tracks.

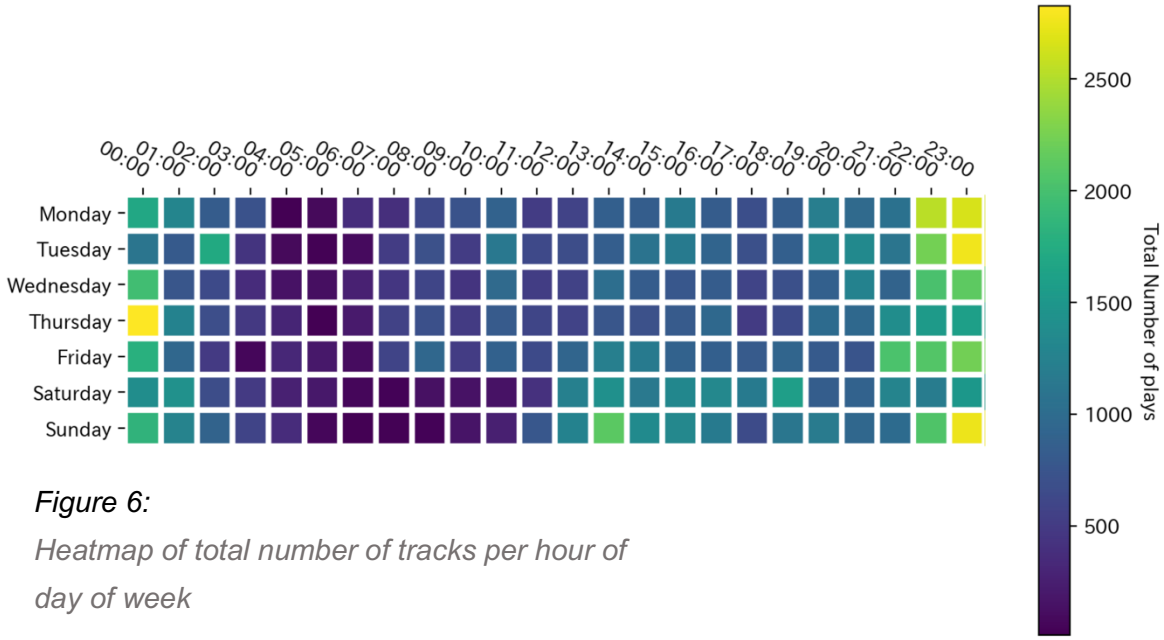

Figure 6:
Heatmap of total number of tracks per hour of day of week

In (Figure 6) we can see a dark patch around 03:00 and 07:00 during weekdays and 04:00 to 11:00 during weekends which can show how during weekends a bedtime can be later and the user returns to listening music later during the day. There are other hotspots during the late hours of 21:00 to 01:00 seeming to be the peak of listening time based on hour. For individual tracks there is a hierarchy of data related to each one, going from Artist to Album to Track, where Artist is the broadest. In (Figure 7) we can see the top artists played is dominated by small portion of the total artists tracked in the dataset (5155 artists).

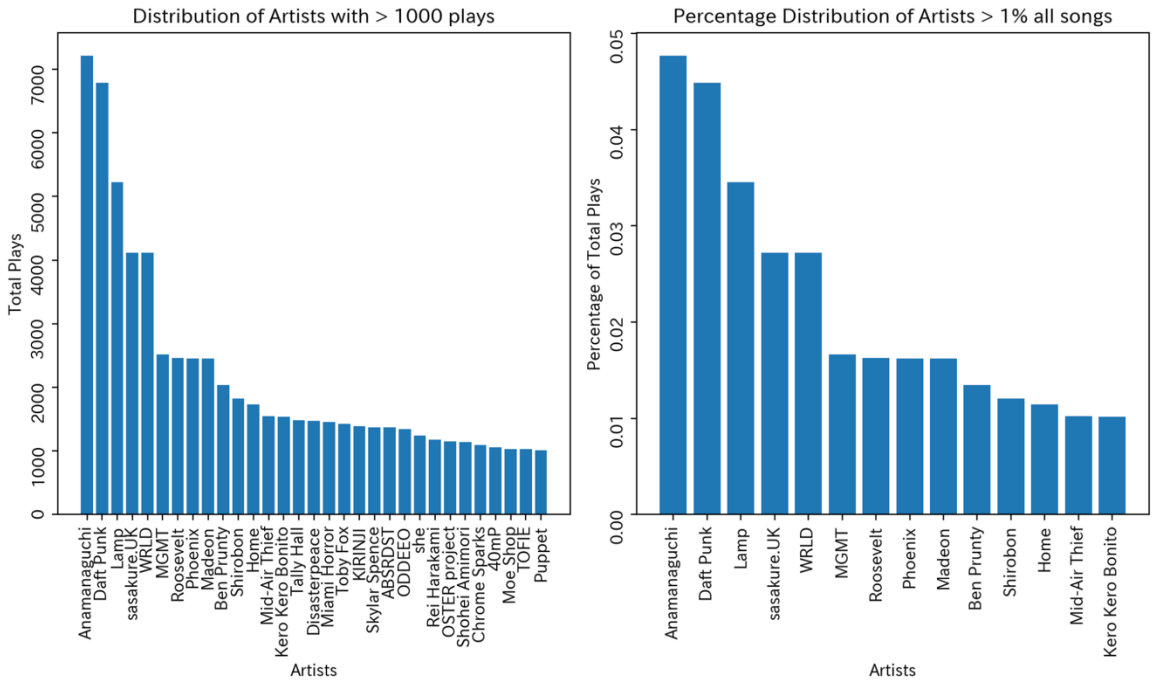

Figure 7:

Distribution of top artists played

Finally, for this section we will look at the time played per each track in dataset. In (Figure 8) there is a histogram with the cumulative tracks binned by number of minutes played, we can see that there are a large quantity tracks that get played only for a short amount of time (less than one minute). This makes sense since Spotify records every single action taken by the user in the data requested, that means even skipping or listening to a song and then changing your mind and moving to another. This does tell us another thing, a way to filter out tracks would be to see their percentage listened (out of their total playtime) and remove those that have low percentages.

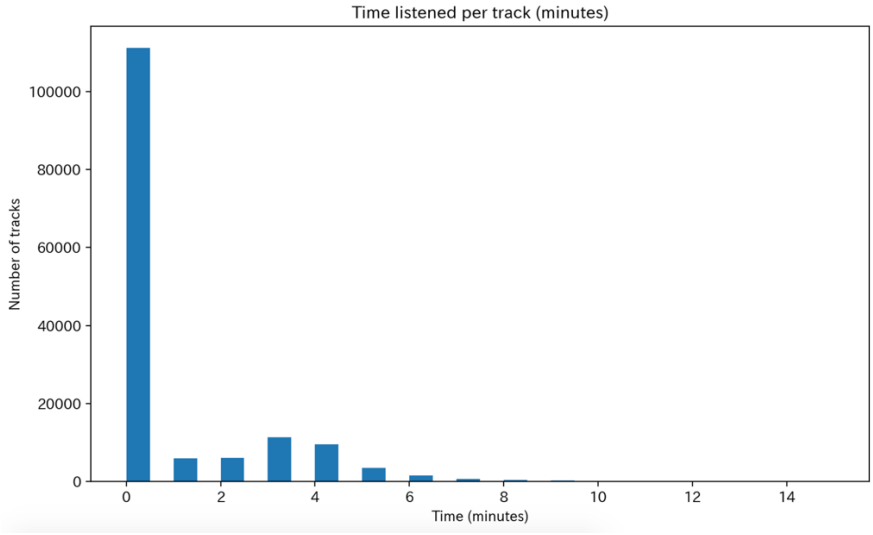

Figure 8:
Distribution of time played per track

For the second part of the analysis, I will look at the track_features dataframe to see any trends. The first analysis is a scatter plot matrix (Figure 8) of all features against each other to see any relationship and give us answer to the correlation heatmap (Figure 9).

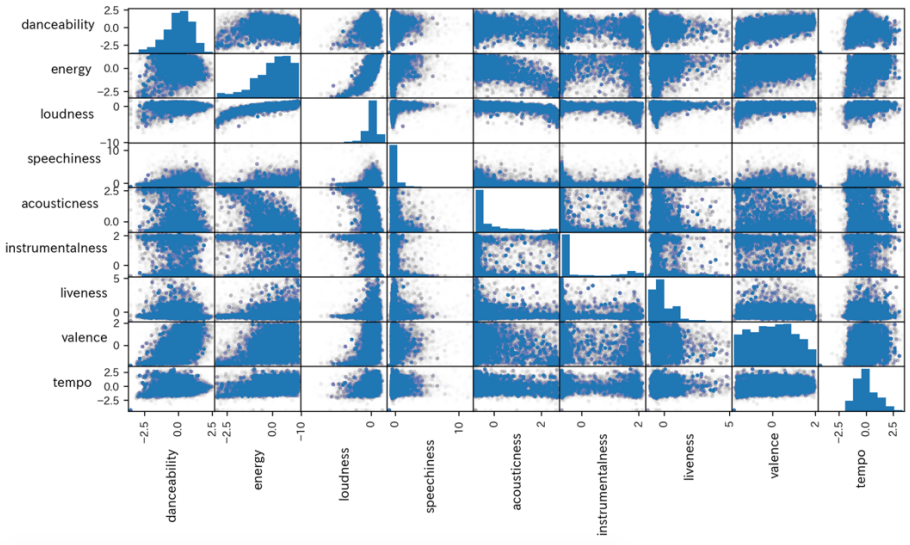

Figure 9:
Scatterplot matrix of all track_features

We can see that various features have high correlations, most prominently loudness with energy, and valence with danceability (Figure 10). These features mostly describe a tracks "happiness" and overall energy, which we can see in the histogram for danceability and energy are skewed right.

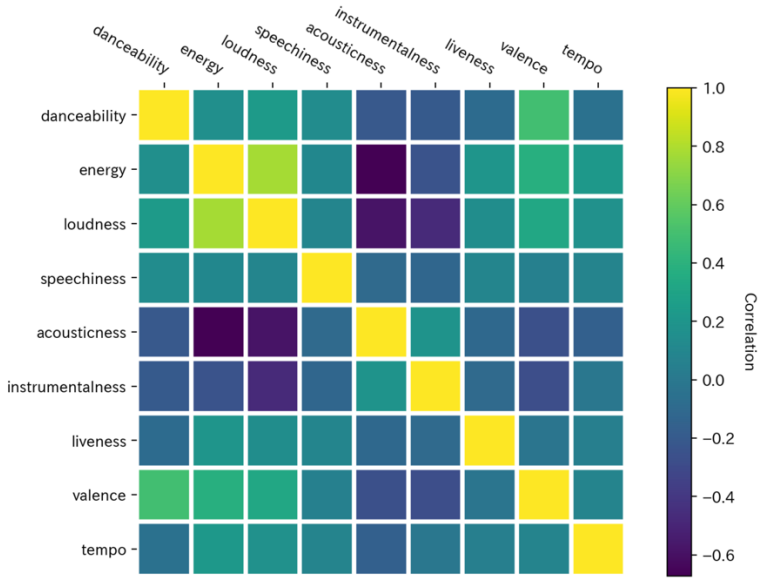

Figure 10:
Correlation Heatmap of all track_features

The last step was to perform Principal component analysis with the Standardized data. I decided on choosing 3 principal components to test out how the K means Clustering algorithm would classify the data. The PCA algorithm with 3 PCs had a total variance of 90%, meaning 90% of data was retained.

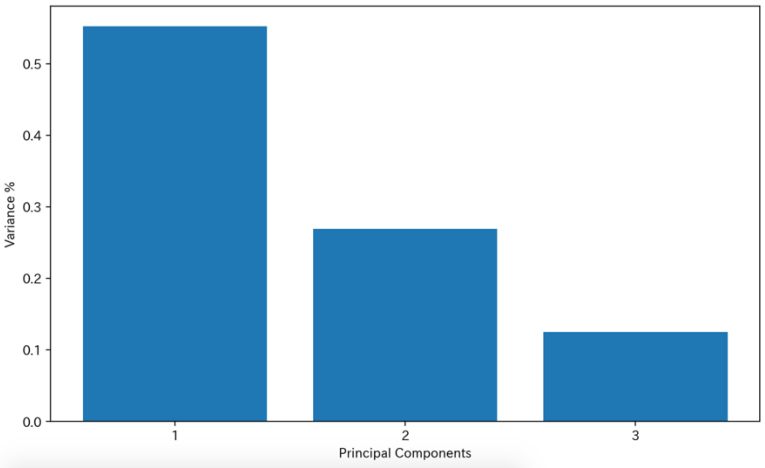

Figure 11:
Bar chart of Variance per Principal Component

Looking at (Figure 11), we can see that the majority of data is encoded in the first two principal components. With (Figure 12) these components are plotted in a 3-Dimensional Scatterplot and we can see a rough shape emerge.

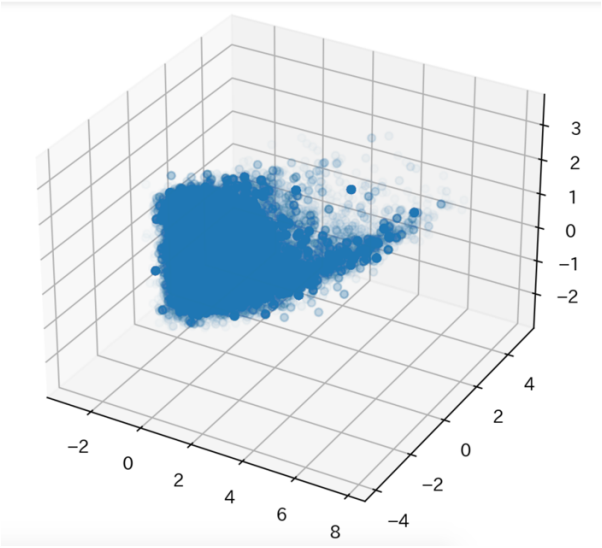

Figure 12:
3D representation of
Principal Components

Once these principal components were chosen the next step was to find the correct number of clusters for the K means algorithm. I plotted the Inertia per cluster from 1 to 9 to use the elbow method and choose what looked like the best number. I ended up choosing 4 clusters for the algorithm based on the graph returned.

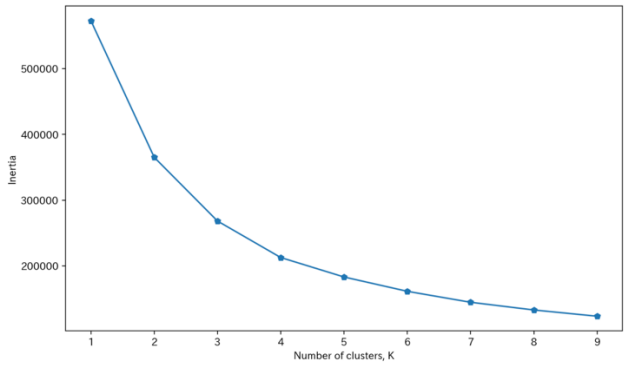

Figure 13:
Line Graph of K means
Inertia per cluster size

Finally, the model was used to predict with 4 clusters the Principal components and then plot in a 4-Dimensional representation in a 3-Dimensional Graph (Figure 14). The Final Cluster Scatterplot shows some boundaries between certain groups and we can assume that with these 3 principal components calculated, that these clusters might be some Larger groups of genres. My theory is that with these few clusters and the inertia still being extremely high as represented in (Figure 13) the clusters might contain tracks that are of different genres but still contain certain elements of acoustic features that might relate to larger genres.

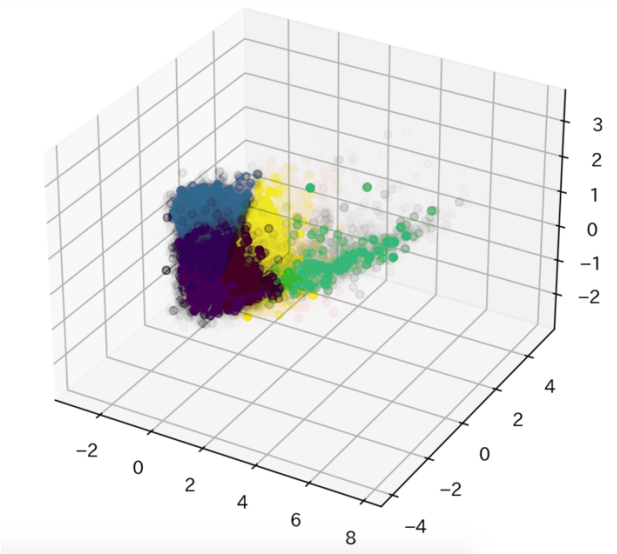

Figure 14:
3D Representation of Clusters

Conclusion

Overall, I discovered that with Spotify Listening Data, most of the time was spent preprocessing and cleaning the data, the API calls with track URIs was also very tedious. The data showed that for one person, their overall variety with artists can be very low especially when playing one singular artist and not discovering outside their preferred genres. Some things that I missed were, getting genre data for each track and creating a model that given a time and N number of previous tracks (with all information like track_genres and track_features) predict a new track with new_features and genres.