Data Wrangling Project Report

(Inter-)relationships Between the Audio Features of Spotify Playlists and/or their Metadata

1 Project Group Members

Name	VUnetID
Dovydas Vadišius	dvs254
Laura I.M. Stampf	lsf270
Lennart K.M. Schulz	lsz480

2 Research Questions

Main Research Question (RQ0):

Are there (inter-)relationships between the audio features of Spotify playlists and/or their metadata[1]? If so, what relationships exist?

Research Question 1 (RQ1):

Is the popularity (measured in number of followers) of a Spotify playlist related to its tracks' audio features (tempo, loudness, danceability, ...)? If so, what relations exist?

Research Question 2 (RQ2):

Are there relations of audio features within playlists (e.g., faster playlists are more energetic)? If so, what relationships exist?

Research Question 3 (RQ3):

How do the relations of audio features within playlists, as identified in RQ2, compare to those within individual tracks?

RQ0 covers the abstract topic this research project is aiming to investigate. The subsequent research questions (RQ1, RQ2, RQ3) are specified instances of RQ0, allowing for concrete analysis.

RQ1 was the initial focus of this research project. However, during the development of the data-wrangling mechanisms it became apparent that the dataset facilitates the analysis of many types of relations. Consequently, RQ2 was identified and specified and, to allow putting its findings into a grander perspective, RQ3 was further added.

3 Data Sources

The project relies on two data sources, both last accessed on 1 Feb 2024:

Spotify Million Playlist Dataset (MPD) [1]:

This official dataset consists of 1 million public Spotify playlists, sampled from the over 4 billion public Spotify playlists at the time (2018). It not only includes information about the contents (tracks) of each playlist but also about its metadata (e.g., the number of followers).

Spotify Web API [2]:

The MPD only includes a list of tracks for each playlist. To investigate the research questions of this project, however, information about the audio features is needed. The Spotify Web API offers an endpoint for exactly this: querying the audio features for a given song.

Audio feature information is not available for all songs. However, out of the 1,205,287 queries, only 44 did not return any audio feature data.

[1] The term **metadata** in the context of a playlist encompasses a range of essential information, such as its name, last modification date, and the number of followers.

4 Data Wrangling Methods

The following illustration of the applied data wrangling techniques is structured into the different stages of the process.

4.1 Data Acquisition

The data acquisition stage is further constituted of multiple parts: An initial data visualization aimed at understanding the MPD for subsequent selection decisions, a data construction and cleaning process to transform the MPD into a more workable format, and the acquisition of audio features for all needed tracks.

4.1.1 Initial Data Visualization

The MPD contains information about 1 million playlists. This information is stored in 1,000 JSON files, each containing information about 1,000 playlists. In total, the dataset consists of 33.54 GB worth of JSON files.

Given the substantial volume of data, it was anticipated that processing would demand considerable computational resources. Thus, the first consideration was whether there was a way to reasonably reduce the number of playlists considered in the further steps without affecting the validity of the results.

Generally, playlists with very few followers may not be of as much interest to this study compared to playlists with higher follower counts. Especially when a playlist only has one follower, being the playlist creators themselves, it may be likely that the playlist was never intended to be a public playlist for other people to listen to as well.

To investigate whether removing all playlists with a follower count of 1 would significantly reduce the size of the dataset, the frequency of follower counts in the playlist dataset was analyzed, the results of which can be seen in Figure 1.

From this figure, it is observable that there are many playlists with very few followers. Exclusion of playlists with only a few followers could drastically reduce the size of the dataset. A further investigation on the concrete effect of a minimum follower threshold on the number of playlists included in further analyses showed that even a threshold of 2 (so only considering playlists with at least two followers) would decrease the size of the dataset by approximately 75%, from 1,000,000 to 245,781 playlists. Thereby, the subsequent processing and analyses were performed with a reduced dataset only considering those playlists that have at least two followers.

4.1.2 Playlist Data Construction and Cleaning

To construct a DataFrame containing the complete playlist dataset, the contents of the individual JSON files had to be joined. Due to the format of these files, it is not

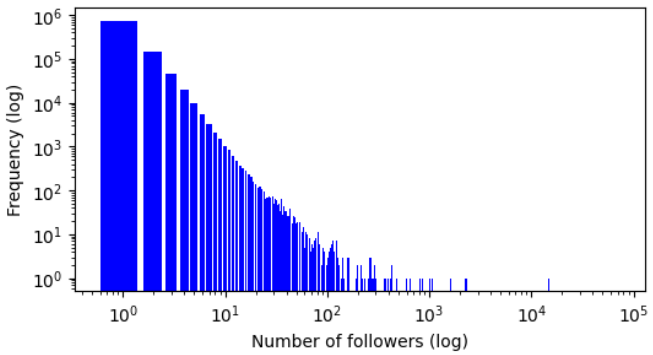


Figure 1: Frequencies of Playlist Follower Counts

possible to simply call `pandas.read_json` for producing a DataFrame of the data. Instead, `json.load` was used to produce a dictionary of the file's contents. The two higher-level keys in this dictionary are "info" and "playlists", where only the latter is of interest. To reduce the size of the dataset, some (for this study unnecessary) metadata was removed entirely while the list containing information about tracks was replaced by a simpler list of track IDs.

As this process turned out to be rather time-consuming (taking over 2 minutes) and unnecessary repetitions of the process should be avoided, the intermediate results are serialized into a Python pickle file (which can then be *unpickled* from within other notebooks in further processing).

4.1.3 Acquisition of Track Audio Features

A set of unique track IDs compositing the entirety of the selected playlists was produced as part of the previous step, posing as the input for this (second) step of data acquisition.

The *Get Several Audio Features* Spotify Web API endpoint [2] takes a list of (up to 100) track IDs and returns an array of audio feature information encoded in the JSON format. To use the API endpoint, however, an authentication token is needed. Such a token can be obtained by creating a Spotify App and requesting a (temporary) token for that App. As the token expiration times are shorter than the expected time needed for all API requests, the mechanism to request a new token upon expiration of the previous one is built into the script.

One, not initially expected, issue with the Web API is that it is not intended for large data collection purposes. Thereby, each App is limited to some number of requests per some amount of time until some cool-down period is enforced. The exact numbers for this, however, are nowhere specified. To allow a fully automated process for the data acquisition nonetheless, the script reads a list of credentials for various Apps and, upon restriction of one App, simply advances to the next in the list. Experiments showed that, with an interval of 0.75 seconds between requests, about 2,000 requests were possible from one App before it got suspended from the API endpoint. Ultimately, 7 different App credentials were needed to obtain the whole of the desired dataset.

4.2 Data Processing

Once all available audio features data was acquired, the data could be processed and merged with the playlist data, computing statistical metrics for all (selected) playlists. The (simplified) process of this step is:

1. Select a playlist from the list of all playlists.
2. Gather the audio features of all songs that are part of the playlist.
3. Calculate statistical metrics of the audio features across all collected songs.
4. Merge the metadata with the calculated metrics and insert the resulting row into the results table.

When reading the data of extracted audio features, it is crucial to instruct pandas to use track IDs as indices to reduce complexity. This allows locating the audio features for a given track in $O(1)$ time rather than by iterating through all tracks and searching for an entry of a desired track ID in $O(n)$.

The initial approach to the processing step was a *naive* single-threaded implementation. However, due to the size of the dataset, this resulted in a runtime of over 2 hours. As the first run missed audio features for a significant portion of tracks (due to network issues in the initial Web API data acquisition run) a re-run was necessary after the missing data was acquired.

Even though time was not necessarily an issue, it was still desired to improve the implementation of the processing step to run (much) faster. As the task is *embarrassingly parallel* in the sense that the list of playlists can be split into chunks that are processed in parallel, the implementation was promptly extended for parallel processing with multiple threads.

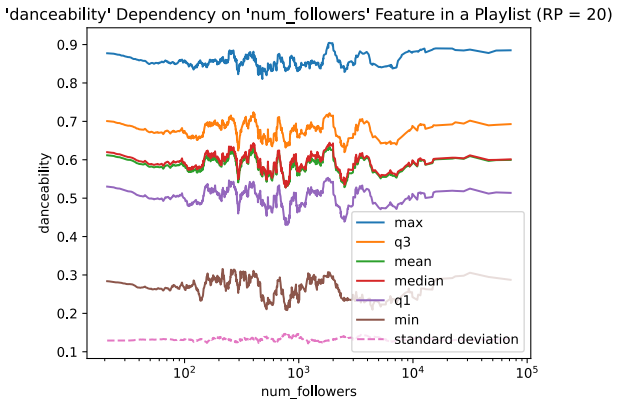
The projected running time of this implementation (based on the progress after a few minutes) was around 8 hours, so about 4 times slower than the single-threaded implementation... Some research quickly uncovered the issue: Python's `threading` library does not allow true concurrency - only one thread can execute code at a time. The `multiprocessing` library does not have this limitation and allows for true concurrency, utilizing multiple CPU cores. This library, however, had a different limitation: the worker function(s) cannot be defined in a notebook but need to be defined in a regular Python file.

After the necessary code modifications, the new implementation was able to complete the task that initially took over 2 hours as a single process in under 23 minutes using 8 processes (on a 10-core Apple M1 Max CPU).

4.3 Data Visualization

Once the table with metrics for all playlists had been generated, the next step was to plot trends for all possible pairs of features: song features (acousticness, danceability, duration, energy, instrumentalness, key, liveness, loudness, mode, speechiness, tempo, time signature) against the metadata (12 * 6 = 72 graphs), and song features against all possible song features (12 * 12 = 144 graphs). Since the data of the playlist features varies significantly, in order to see the trends clearly, a rolling period (RP) in a range of between 5 and 2000 (depending on the graph) was applied to the data. Additionally, the function that generated graphs had visualization options to increase the readability of the graphs.

A total of 228 graphs were plotted in the task of finding noticeable (inter-)relations between various audio features and/or playlist metadata.



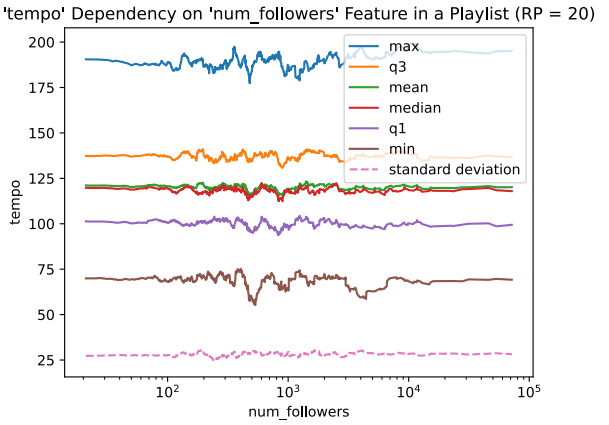
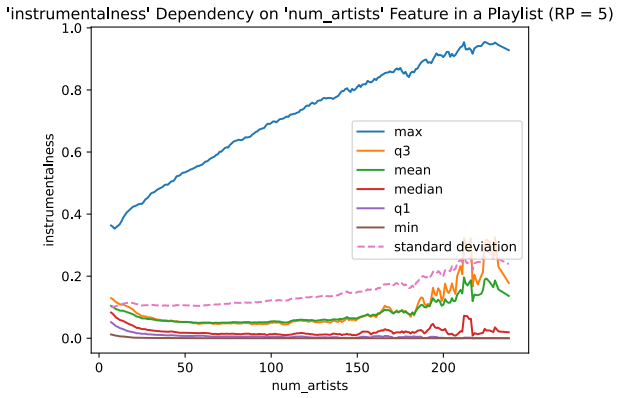


Figure 2: Feature Dependency on Number of Followers



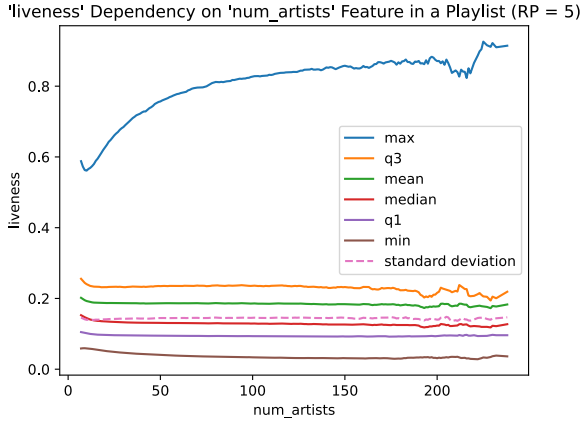


Figure 3: Feature Dependency on Number of Artists

5 Results & Discussion

To answer RQ1, the playlist statistics (min, Q1, mean, median, Q3, max, standard deviation) of the aforementioned features were plotted against the number of followers. However, contrary to our initial assumptions, **none of the results showed a significant relation between audio features and playlist popularity**. An example can be seen in Figure 2 which shows the tempo and danceability feature dependency on the number of followers. While the feature values fluctuate with varying follower counts, the overall trend is stable in both numerical values and variance.

On the other hand, there exist some trends between other values of playlist metadata and its features. Playlists that have more artists tend to have a higher variation in instrumentalness of the songs (sd of ∼0.1 with artist, while sd is over 0.2 with $\geq$ 200 artists), while the variation of other features (such as liveness in Figure 3) remains constant. Additionally, higher values of certain metadata (num_artists, num_tracks, num_albums, num_edits) tend to increase the span of many track features. This is expected, as having more (and more diverse) tracks in a playlist results in a higher probability of having a track with a uniquely high or low value for some feature. This can be seen in Figure 3, where the (blue) line representing the max value has an increasing trend.

For RQ2, **several relations between different features of tracks can be observed**. For instance, acousticness is related to the energy seeming linearly with a negative correlation (Figure 4), while acousticness and instrumentalness have a positive correlation. Moreover, tempo (measured in beats per minute (bpm)) appears to be an important factor for multiple features. A higher tempo of a song can be related to a lower acousticness, but also higher values of energy (first plot of Figure 5) and loudness. Other features tend to increase or decrease in certain tempo ranges: danceability values are highest at a tempo of 95-125 bpm but decrease with higher values, while speechiness has a valley of lowest values at around 115-120 bpm (second plot of Figure 5).

When comparing the feature statistics of the playlists against features of individual tracks for RQ3, certain tendencies arise. **Some pairs of features correlate quite closely**, such as acousticness and energy (Figure 4), mode and speechiness, instrumentalness and loudness. Other features, however, **exhibit relations in the playlists that are not found in individual songs**, average energy feature values of the playlists seem to increase significantly faster than energy of individual tracks with a higher tempo. The dependency on instrumentalness and tempo is the opposite, as this playlist feature tends to sharply decrease when tempo increases from 100 to 110 bpm, while the feature levels of individual songs stay constant. Lastly, the speechiness feature has several spikes at around 95-100 bpm and 130 bpm, which are the popular ranges for tracks with lyrics (e.g, hip-hop and rap for songs under 100 bpm).

6 Conclusion

6.1 Addressing the Research Questions

Based on the data visualizations (some of which presented in Section 5) the following conclusions are drawn for the

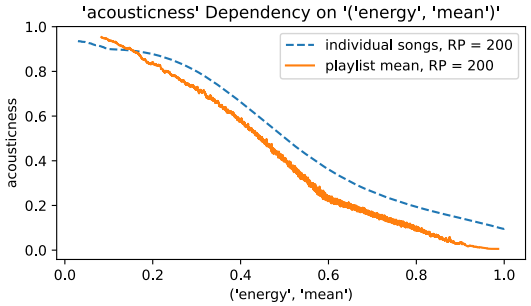


Figure 4: Acousticness Dependency on Energy

research questions as defined in Section 2:

RQ1: There are no significant relations between playlist popularity and audio features.

RQ2: There are multiple positive and negative correlations of two features of the songs within a playlist. Notably, tempo acts as an important factor for various features.

RQ3: While some relations between audio features within playlists are reflective of the general relations between those features in all individual songs, certain relations can be observed that are not present in the set of all individual songs.

Even though the answer to the initial research question (RQ1) was, contrary to the early expectations, negative, RQ3 lead to some interesting findings. Apparently, playlist creators tend to build playlists from songs such that relations between some audio features are prominent in the playlist even though they cannot be generally found in the set of all songs. For instance, slower playlists tend to have a much higher instrumentalness value even though slower songs are not generally more instrumental.

6.2 Limitations

There are several limitations to the approach taken in this project.

Playlist selection: The MPD consists of 1 million playlists drawn from a pool of over 4 billion public playlists on Spotify. However, the selection process for these playlists remains unclear. Since the MPD represents only a subset of the total playlists available, its validity for analysis relies heavily on the sampling method employed. Without a clear understanding of how the playlists were sampled, there exists a risk of introducing bias into any data analysis conducted using the MPD.

Data for popular playlists: The distribution of follower counts among playlists reveals a pronounced right skew, as illustrated in Figure 1. Notably, there is a scarcity of playlists with high follower counts. This lack of data points for more popular playlists reveals a notable limitation in our analysis, as it limits our ability to draw comprehensive conclusions across the entire spectrum of playlist popularity.

Rolling period in plots: The visualizations of trends discussed in Section 4.3 have been produced by taking

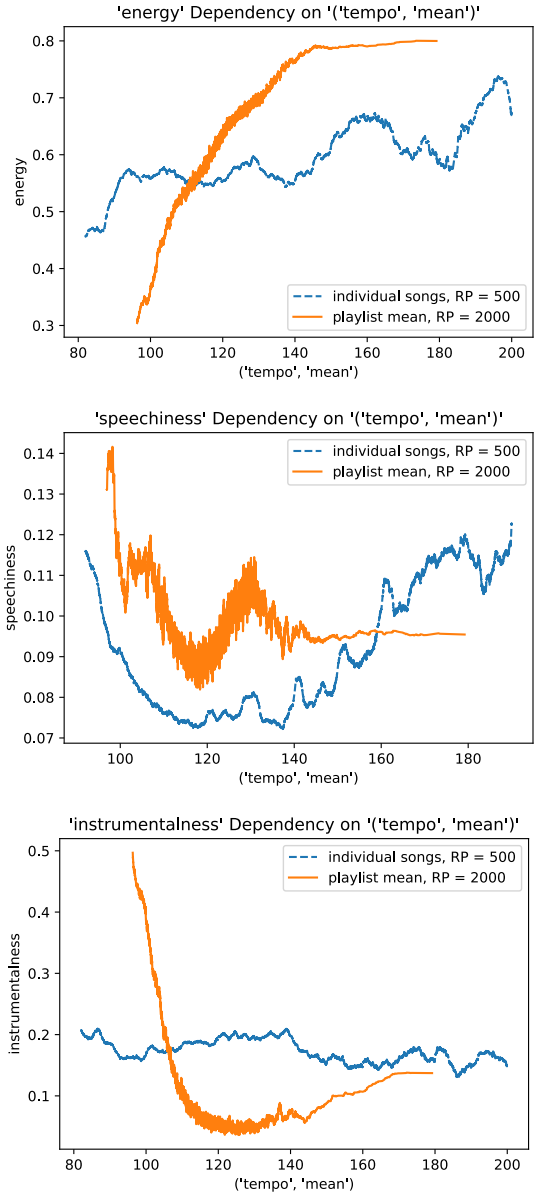


Figure 5: Feature Dependency on Tempo

a rolling period of between 5 and 2000 due to their otherwise highly volatile and fluctuating nature. By doing so, a number of initial values have not been directly included in the diagram. However, since the number of missing values is relatively small compared to the number of data points in the graphs ($< 1\%$), this did not suppress the observed trends.

References

[1] AIcrowd. "Spotify Million Playlist Dataset Challenge", [Online]. Available: `https://www.aicrowd.com/challenges/spotify-million-playlist-dataset-challenge/dataset_files` (visited on 1 Feb. 2024).

[2] Spotify. "Web API Documentation: Get Several Audio Features", [Online]. Available: `https://developer.spotify.com/documentation/web-api/reference/get-several-audio-features/` (visited on 1 Feb. 2024).