**Final Project - Milestone 3**

**Revised Project Statement & EDA**

**Team: Johannes Kolberg, Hardik Gupta, Will Seaton**



***Instructions***

*Wed, November 20: EDA and Revised Project Statement (15 points)*

On Canvas in the ’Final Project - Milestone 2’ assignment, submit a 2 - 3 page revised project statement and EDA (can be created using Latex, word processing software, etc.) and an accompanying Jupyter notebook (that was used to create the visuals). Your 2 - 3 page submission should include:

* A description of the data: what type of data are you dealing with? What methods have you used to explore the data (initial explorations, data cleaning and reconciliation, etc)?
* Visualizations and captions that summarize the noteworthy findings of the EDA.
* A revised project question based on the insights you gained through EDA.
* A baseline model.

*Note: the text here is 2-3 pages, as instructed, but the EDA plots make it longer.*

**Data Description**

The raw dataset contains 1,000 csv files that together comprise 1 million Spotify playlists, totaling 11.63GB. Each playlist contains an arbitrary number of songs, each of which is stored with various fields. Most files contain about 65,000 songs for roughly 65 million observations (or songs) across all the files. However, many songs appear in several playlists and are therefore repeated. Across the 1 million playlists, there are only about 2.5 million unique songs – far fewer than the 65 million songs contained in all the playlists taken together. The playlists vary in terms of the number of songs added to each, with the largest playlist comprising 341 songs and the smallest 3. From these files, we get information about the song title, song length, artist name, album name and the associated Spotify URI of the song, album, and artist.

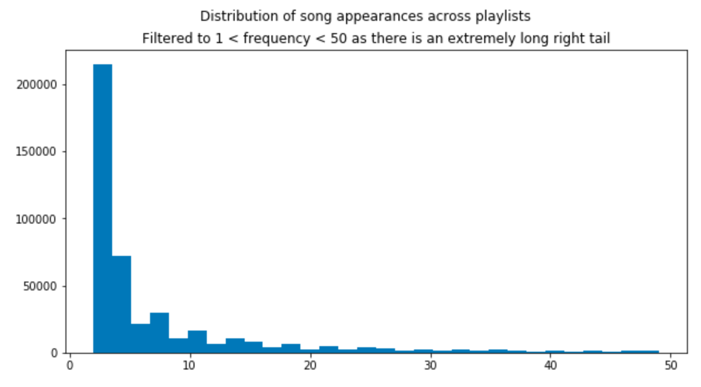
Since there is a lot of repeated information, we can drastically reduce the size of the dataset without losing any information. We do so by creating a master reference table of the unique songs, and store the playlists together as a list of vectors, where each vector is a playlist and each item in the vector the song ID (from the reference table we created) in the order they appear. To make this computationally tractable we leverage Pandas dataframes and the fact that their indices, if sorted and maintained properly, leverage hash tables for quick lookups.

This reduces the complete dataset to a songs master table of 0.42GB and a list of playlist vectors of 0.54GB. This is less than 10% the size of the original data format, with no information lost. However, when populating the songs master table with metadata, such as genre, we work on a subset of 200,000 playlists for this milestone, to avoid time-outs with Spotify’s API and to lighten the computational load of the EDA and baseline model. We enrich our data via the API using Spotipy, a lightweight Python library that allows us to authenticate to Spotify and query a large number of features on the song, artist, and album. We join in new features such as album and artist popularity, album release date, artist genres, and track features like danceability, energy, loudness and more. After enriching our song data with additional features directly from Spotify, we produce our final master pickle files for analysis.

**Exploratory Visualizations**

We explored a number of dimensions of songs and playlist data. For songs, only a few songs appear in an incredibly high number of playlists with rapid and significant drop off. Playlists have a peak at around 20 songs per playlist with a steady decline in distribution from there.

A screenshot of a cell phone

Description automatically generated

In looking at feature relationships, the majority appear uncorrelated with a few interesting exceptions. As danceability increases, we see a slight linear relationship with playlist inclusion. Loudness has a narrow band of inclusion around -7, showing that overly loud songs are not welcome. Perhaps surprisingly, higher energy songs appear to have no relationship with playlist inclusion, though this speaks more to the popularity of playlists focused on "Sleep" or "Classical", which have lower energy. Lastly, if your song is too long, above 5.5 minutes, it is going to be included in very few playlists.

We identified the preferred tempo for dance music to be about 125 bpms, which appears as a "tempo bump" on the plot of danceability against tempo. We wanted to see if there were certain musical keys that were more popular and found that First Key (C♯/D♭) and Seventh Key (G, or sol) are clear preferences for the top 100 most included songs in our dataset.

A screenshot of a map

Description automatically generatedA screenshot of a cell phone

Description automatically generated

In addition to 125 bpms for danceability, we found several other popular tempo peaks, including 80, 100, 120, 128, 140 and 170 bpms. Each of these likely represents the standard tempo for certain genres of music.

A screenshot of a cell phone

Description automatically generated

We investigated artist and album popularity and their relationship to playlist inclusion. Artist popularity shows an unsurprising relationship where the more popular an artist is, the more playlists it is likely to be included in. There is an interesting split of album popularity, where songs from top quartile popular albums and bottom quartile unpopular albums appear in playlists to a greater extent while songs from middle quartiles of average albums appear in very few. This may speak to the presence of "one hit wonders" on unpopular albums being included across playlists.

A screenshot of a computer

Description automatically generatedA screenshot of a computer

Description automatically generated

In looking at the distribution of album release year for songs in playlists, people primarily care about recently released music. Songs released even five years ago are included at a dramatically lower rate than songs from recent years.

A close up of a logo

Description automatically generated

From the Spotify Web API, we pulled and explored a number of song features, looking at how they have changed over time by album release date. The most dramatic cross over occurs with acousticness and energy, as the former declines severely starting in the 1950s and the latter rises steadily over time. The trend reverses itself temporarily during the 1980s but diverges again at a slower pace from 1990 to today. Valence, Spotify’s measure of positivity in a song, has also been declining at a slow but steady pace since about 1977. The most recent trend occurs with danceability, spiking to its highest ever levels starting in 2010.

A close up of a map

Description automatically generated

**Baseline Model**

For our baseline model, we deployed K-Means Clustering as an unsupervised method to group songs into clusters based on distance between their feature values. Conceptually, it works similarly to K-Nearest-Neighbors in that regard, except that the number of clusters is specified as a hyperparameter and their centers placed iteratively to minimize the summed distances between each song and its assigned cluster. The K-Means is fit on 95% of our 200,000 randomly chosen playlists. For new songs it can then assign them a cluster.

To generate a playlist, we then take some seed songs, use the fitted K-Means to predict their clusters, and use that information to pull new songs to populate the rest of the playlist, on the assumption that like songs have been grouped together and a user wants to hear similar songs within a single playlist. If several songs have been provided, we have experimented with two approaches. The first is to select the mode cluster from all provided songs and select further songs from there. The second is to randomly select new songs from all clusters predicted from our starter songs, but with the probability of each cluster depending on the number of starter songs from that cluster. Given a cluster from which to pull a song, it does so at random.

A downside to using an unsupervised model in this way is that we have no objective measure of song or subsequent playlist quality. We are likely adding songs that might be similar along multiple features but without a sense of if people enjoy the song. Spotify classifies tracks by popularity, but we saw from our EDA that popularity does not necessarily track with playlist inclusion. Furthermore, K-Means treat all features the same in terms of distance (especially after min-max scaling), but it is likely some are more important than others.

The K-means model only used the song metadata table, and therefore ignores playlist information. One driving factor is the high dimensionality of including co-occurrence across 200,000 playlists. For the baseline model we chose not to incorporate it, but we intend to pursue an approach that leverages the fact that songs co-occur in some playlists to make better-informed playlist suggestions.

**Updated Project Statement from EDA**

We will continue to focus on creating a playlist from a cold start with minor user input, such as the first five songs or a genre selection. Our current model successfully clusters like songs together but these clusters can be improved and narrowed for better song targeting.

We see from our feature exploration that song attributes have some relationship to playlist inclusion rate. Incorporating playlist inclusion as a measure of quality and weighting song selection from our derived clusters can serve as a form of quality by treating playlist inclusion as user-stated listening preferences. By improving cluster definition and weighting song selection based on playlist inclusion, we believe we can extend our playlist generation model to have improved results and more accurate song families.