**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.

**Data Description**

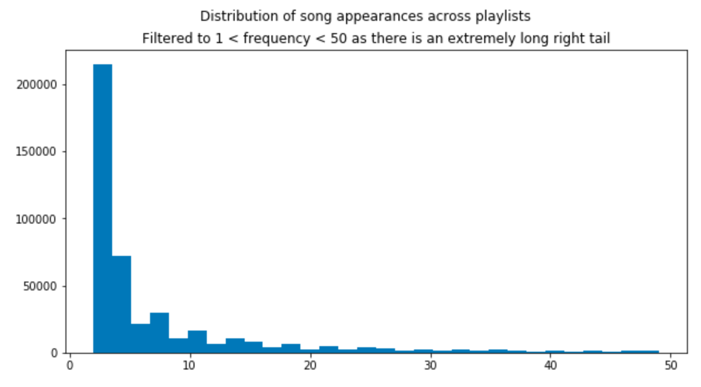
We began with 1,000 csv files, each containing about 65,000 songs and associated playlists, totaling potentially 65 million observations. A song appears in our data every time it is added to a playlist. The playlists vary in terms of the number of songs added to each, with the largest playlist comprising 341 songs and the smallest 3. From these files, we get information about the song title, song length, artist name, album name and associated 'uri', or Spotify unique identifier, for each. To handle the scale, we first create a reference table of all unique songs and their metadata found across all 1,000 files. From there, we create a smaller file representing playlists, which contains just the playlist id and a vector of associated song ids. We do this in a For loop, by enumerating the list of file names, reading them in one at a time, using Pandas groupby to calculate count appearances and unique track\_uri values and finally appending those to a DataFrame 'songs' and a Series 'playlists’, respectively. This step had a run-time of 2.2 hours. Over these optimized files, we perform replacement of some values, such as track\_uri with song\_id and select specific index columns, with a run-time of 7.7 hours.

Once our unique songs and playlists files are ready, we use the Spotify Web API to enrich our data. We used a package called 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 cluster songs into clusters, or families. To populate a playlist, we take a new song, predict its cluster and pull other songs from that cluster 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 select from all clusters predicted from our starter songs but with weights based on the starter songs.

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.

Our initial model has been run over a representative sample of 20% of all songs and playlists due to memory and computational costs. One driving factor is the high dimensionality of our data structure, where we maintain a binary indicator matrix saying whether each song (row) is in each playlist (column). We intend to pursue a reduction in dimensions by a method such as applying K-Means Clustering to the playlists as well, deriving a much smaller number of playlist families that can be used as features in our binary indicator matrix instead.

**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.