Milestone 3 Aileen Villalpando, Colin Harvey, and Clara Cousins
Group 52 TF: Matteo Zhang

Part I. Description of data and models

Our goal is to automatically generate a thematically-focused playlist for a user once provided with user-selected mood and songs. We will predict this using the audio features of the user's songs and the lyrical content of songs in our database. This implementation will identify a cluster of other songs with similar sound qualities to the user's selected songs. Then we will predict the mood of the clustered similar songs according to *their* lyrical content and recommend to the user any songs that match the selected mood consistent with the user's specified mood. We chose 10 moods to identify from the lyrical content: loving (n=436 songs), sad (n=447), angry (n=521), happy (n=686), hype (n=1089), empowered (n=946), party (n=600), ignant (n=2083), chill (n=503), and sexy (n=1049). These moods are the key terms that we fed to the Spotify search engine to choose 3-7 playlists that fit into each of these classifications. To determine the subject matter of each song, we scraped Genius.com for the lyrics for each song. The Genius lyrics database relies on reputable users to transcribe the lyrics, who also update the database regularly with new songs.

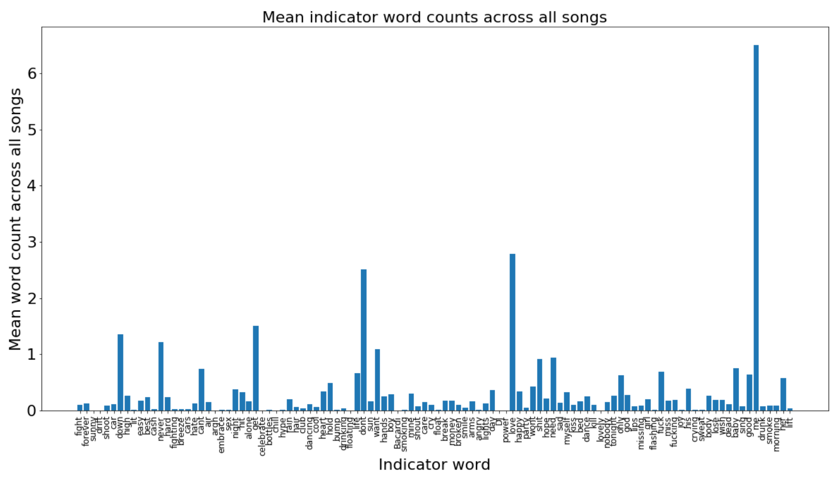
For this milestone, we focused on 4 out of the 10 specified moods. From each mood list, we chose one of playlists and used the Spotipy Python library and the playlists' uniform resource identifier (URI) to query the URIs, audio features, song titles, and artist name(s) for each song in the playlist. We used string formatting in Python to construct the appropriate URL to query the lyrics given the song's title and artist's name. In this initial phase, we are only including songs that have a single artist and have lyrics stored at the URL formatted as in the following example, where "Us" is the song title and "Clara Mae" is the artist: "https://genius.com/Clara-mae-us-lyrics". We used BeautifulSoup to extract the song lyrics from Genius.

Model 1 predicts the playlist mood using 100 indicator words (intuitively selected; indicator word counts shown in **Figure 1**) that describe the 10 moods as features via a logistic regression model. We constructed a dictionary for each song where the keys are words in the song and values are the number of appearances of each word. Then we trained a multinomial logistic regression classifier (Model 1) on these data to model the relationship between the predictors (frequency of each indicator word) and the response variable (mood) by splitting the data into an 85:15 train-test split and then fitting the logistic regressor to the training data using ten-fold cross-validation. In our first iteration, we have only included songs from 4 moods: "loving", "happy", "sad", and "ignant." For this subset of data, our model yielded a training accuracy of 0.764 and a testing accuracy of 0.583. The coefficients of each predictor in Model 1 are shown in **Figure 2**. Model 2 uses a k-Nearest Neighbors (kNN) approach to generate a clustering model to group songs based on their audio features, listed in **Table 1** (two audio features that appear to differ across moods are shown in **Figure 3**). In the accompanying notebook, we use sklearn's "NearestNeighbors" to find the 10 closest neighbors of each song in the user's base playlist (simulated as 5% of the songs in our database) after training clusters from the remaining 95% of the songs in our playlists. We use Model 1 to identify the predicted mood of each song and make playlist recommendations.

In summary, we can use Model 1 and 2 to develop a program for automatic playlist generation. The user will specify a song theme and supply a short list of songs that, according to them, fit this theme. We will pass their short list of songs into Model 2 and determine the cluster of songs that best matches the short list of songs in terms of audio features. In the coming weeks, we will add songs to our training database and will take the predicted cluster from Model 2 and for every song in the cluster, we will pass the frequencies for each indicator word into the Model 1 classifier. This program will then suggest songs for the user when predicted subject content labels from the Model 1 are consistent with the user's specified mood.

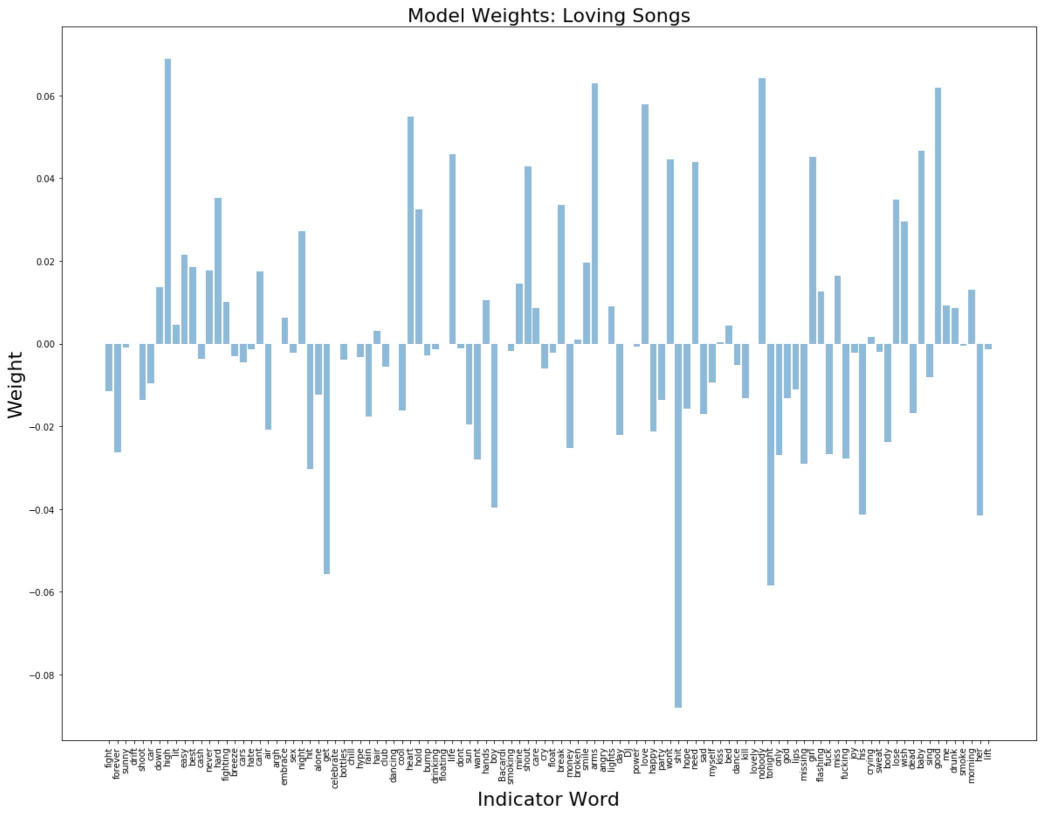
Part II. Visualizations and captions that summarize the noteworthy findings of the EDA plus accompanying Jupyter notebook

Figure 1. Appearance of indicator words in tracks used to train Model 1.



Caption: “Me” was used most frequently, while many words were used less than once per song on average.

Figure 2. Bar plots of the model coefficients for the mood “Love” from Model 1.



Caption: We observe that “shit” was the most influential predictor in classifying songs into the “love” class. A higher frequency of this word had a negative association with being classified as a love song. More instances of the word “high” conferred greater chance of classification as a love song.

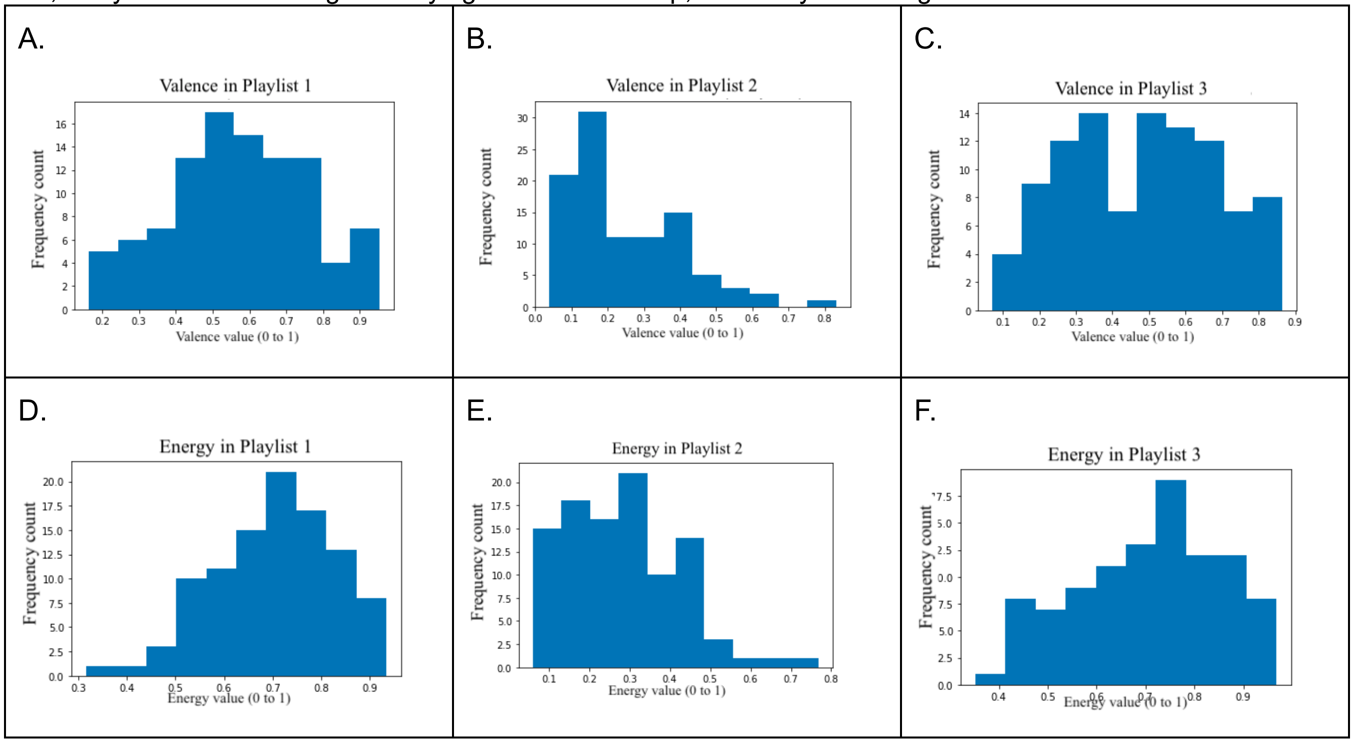
Table 1. Descriptions and expected values of audio features for each song.[1]

Audio feature	**Description**	**Values**
Danceability	Determined by tempo and rhythm	Continuous floats from 0-1 (0=low)
Energy	A measure of musical intensity	Continuous floats from 0-1 (0=low, e.g., classical)
Key	Key of track	Integers from -1 to 11 (0=C, 1=C♯/D♭, 2=D, 2.5= D¼#)
Loudness	Correlated with amplitude of the sounds	Continuous negative floats -60 to 0 decibels (dB)

[1] "Get Audio Features for a Track | Spotify for Developers." https://developer.spotify.com/web-api/get-audio-features/. Accessed 17 Nov. 2019.

Mode	Major or minor of a track	0 (major) or 1 (minor)
Speechiness	Percentage of speech to music in a track	Continuous floats 0-1 (1=mostly speech-like)
Acousticness	A prediction of whether the track is acoustic	Continuous floats 0-1 (1=acoustic)
Instrumentalness	A prediction of the presence of vocals	Continuous floats 0-1 (1=instrumental, less vocal)
Liveness	A prediction of whether the track was recorded live	Continuous floats 0-1 (1=live)
Valence	A measure of the emotional positivity	Continuous floats 0-1 (1=positive sounding)
Tempo	Track speed, related to average beat length	Nonzero integers from 0 to 250
Duration (ms)	Duration of track	Milliseconds
Time Signature	Measure of beats in each bar	Integer values, typically less than 5

Figure 3. Histograms of audio features valence and energy in three playlists, where Playlist 1 is called "Happy Hits," Playlist 2 is "Sad Songs for Crying Yourself to Sleep," and Playlist 3 is "Ignant Shit."



<u>Caption:</u> Amongst the 13 available quantitative audio features, two features were explicitly related to emotions: valence and energy. Valence describes the positivity of a track with continuous floats from 0 to 1. A valence of 0 describes a less positive-sounding track; a valence of 1 describes a more positive-sounding track. Visual analysis shows that Playlist 1 and Playlist 3 both had high modes, although Playlist 1 had a smaller distribution. Playlist 2, titled "Sad Songs for Crying Yourself to Sleep," exhibited a much lower mode valence than either Playlists 1 or 3. From this exploratory analysis, it seems that playlists which mention specific emotions in the title show narrower distributions than those with less explicitly-mentioned emotions; "sad" and "happy" are emotions that can be identified easily with valence, whereas "ignant" is not as well discerned. The next step would be to continue this analysis by reviewing histograms for the same feature for all playlists listed for a certain mood. This can confirm whether there is a trend in the distributions for our chosen happy playlists.

Energy showed similar patterns between the three playlists chosen. Low energy songs score around 0 while high energy songs score towards 1. Playlist 1 and 3 show high modes, where 1 has a smaller distribution than Playlist 3; once again, this may be attributed to the playlists' focus on a specific emotion ("happy"), rather than a general mood ("ignant"). Playlist 2 exhibited lower energy values.

<u>Part III. A revised project question based on the insights you gained through EDA</u>

Can we accurately predict the mood of a song from a subset of its lyrical content in order to match the playlist-maker's mood? Can we use Spotify-derived sound qualities to determine the similarity between potential suggested songs and user-input songs before giving playlist suggestions with the appropriate predicted mood?