Group 8 Project Proposal

Alex Qian, Dmitry Shlykov, Patrick Scerri, Jason Chin, Joe Diaz, Porfirio Mohabir

Problem Statement

We will be exploring a Spotify dataset which includes information on artists and songs. We found the information in this dataset, including artist and song popularity, danceability, tempo and instrumentality to be very fascinating and we believe it can reveal interesting insights. We chose this topic because we found it interesting on a personal level and because we use the Spotify software ourselves. We're approaching the project as though we're analysts at Spotify who are using this data to enhance the customer experience, look for new opportunities for the company, and reveal trends about the music industry. We'd also like to create a system with which we can better recommend new artists and songs to our customers.

Dataset, Description, and Proposed Analyses

Our data can be found on Kaggle here.

The dataset, data.csv, is the full Spotify dataset with a total of 19 columns. It contains song data like artist, release year, popularity, and several different song metrics such as energy, liveness, and tempo. The datasets, data_by_artist.csv and data_by_year.csv, contain most of the same data but examined in different ways.

The data_by_artist.csv file aggregates all of the song data for a particular artist and provides a composite score so you can determine that particular artist's popularity, liveness, loudness, etc. The data_by_year.csv file aggregates all of the song data for a calendar year and provides composite scores for songs from that year. What we plan to do is use a combination of all three datasets to generate song/artist recommendations based on what a customer is in the mood for. We will also perform some analytics on the 3 datasets in order to gain novel insights such as:

-Generating a line chart to show how the tempo, danceability, liveness, etc. of music has evolved over a given time period.
-Generating a scatter plot to examine the relationship between different song metrics. For example: Do songs with higher tempos have shorter durations? What's the relationship between the song's energy level vs loudness?

When starting the program users will be provided instructions and have the option to select between an analytics mode and a recommendation mode. By selecting the "analytics" mode, the user will be able to input certain parameters that output different data visualizations like the ones listed above. If the user selects the "recommendation" mode, the user will be asked if they would like to search by song, artist, or year. They will then be required to input answers to a few questions that will output a recommendation based on what they were looking for.

Methods

In order to find songs and artists most similar to what the user would like, we will consider two methods - first considering the “angles” between songs and then the “distance” between two songs. Each song and or artist has features as listed by Spotify - including danceability, energy, beats per minute, etc. Mathematically, we can treat these values as found by Spotify as entries in a vector and each vector lies in the dimension space.

It is known that the dot product of any two vectors (multiplying the first entries of the vector, then the second entries, and so on and then summing all the products) divided by the lengths of each vector (found by distance formula) results in the cosine of the angle between the vectors. We can say songs/artists are similar if their vectors are pointing in similar directions - this would imply that certain proportions of the songs are similar. Pointing in similar directions means that the angle between the vectors is small, thus the cosine is close to 1. Given a song/artist a user wants to find similarity to and the features the user prioritizes, we can find the cosine of each other item with respect to the chosen song/artist and then rank them. One concern that we have with this is that this does not account for the length of the vectors, that is, even if two vectors may point in the same direction, there is a difference between those with large magnitudes versus those with small magnitudes.

Our second method also treats our songs in a vector space and finds the distance between each other item and our inputs using the distance formula. Items are then ranked with lowest distances being preferable; this can be thought of as a scatterplot in which we find the nearest neighbors. Two concerns with this method are that angles are not accounted for as accurately as the first method and that it’s possible for one dimension/feature to be extremely influential in determining the ranking based on the scale. We will need to find a way to account for the scale, possibly by using z-scores instead.

In Python, we can run these algorithms by creating a class of songs/artists and then generating a list of all songs/artists as objects. When given an input of a song/artist/features, we can then find the cosine and distance of each item with respect to our inputs using a map function and then construct our rankings for both methods. Songs/artists that are near the top of the rankings in both lists should be those that are most similar to what is desired.

<u>Rough Framework</u>

# read file

# let user choose analysis vs algorithm

# analysis

    # by year?
    # BPM, year, popularity?
    # time series?
    # histograms?
    # key/tempo

# rank variables for popularity, then explore the top 3 variables and how they correlate with other variables and years/genres/.. --- tell an interesting story - presentation
# regressions -
# stacked barcharts - expl energy level & count vs popularity

#recommendation algorithm

# ask user for artist
# ask user for which features they care about
# run algorithm, display top x