**Spotting The Hits**

**CoNVO**

**Context:** Spotify is one of the world’s leading music streaming services. Beyond providing a large library of artists, albums, and songs, they are well known for their collection of data-generated and editor-curated playlists. Some reach out to individual users’ tastes, while others present a set of tracks that will be of broad interest to a variety of users, based either on the playlist’s content (e.g., global hits, or ‘country music’) or context (e.g., a certain mood).

**Need:** The goal of Spotify’s playlists is to please current users (reducing churn) and to attract new customers (increasing growth). They want as many $10 monthly subscriptions as they can get. Thus, they are always looking for new ways to create playlists that will be attractive, novel, and pleasurable for subscribers. Currently, they are lacking a playlist which predicts which songs are about to take off as hits (i.e., pre-viral tracks).

**Vision:** A web app, with the click of a button, generates a playlist that predicts which songs recently added to the Spotify library are likely going to be hits in the coming months. It will include a link to access this playlist via Spotify, where the user can choose to add the playlist to their personal account. It will be particularly enjoyed by those music fanatics who enjoy responding to the question “Have you heard\_\_\_\_ by \_\_\_\_\_? I just heard it on the radio and it was awesome!” with “Um… duh Janet… that’s been on my running playlist for months.”

**Outcome:** This playlist is eventually made public and promoted on Spotify’s “Discover” feature. Spotify takes ownership of the playlist and has editors which curate the automatically generated set of tracks in whatever way they wish. The success of the playlist’s *model* will be measured by how many of the songs featured on this playlist go on to become chart-topping hits. The success of the *playlist* will be determined by the relative amount of user-base growth after its introduction.

**The Ideal Process:**

* Three data sources will be used for this project: Spotify data, Twitter data, and billboard chart data. All have something unique to offer and can be queried through python. The primary novelty of this playlist is the utilization of twitter content to predict what is pre-viral on Spotify.
* Getting the data:
  + Pull top 100 charting songs every \_\_\_ days for \_\_\_\_ years from now.
  + Clean the data so that every song is only present once, but retain the following information for each song: date the song first entered the charts, highest position on the charts, and duration it remained on the charts. Create some metric of “song impact” using highest position and duration on chart. Also, create a categorical variable indicating if the song was top 100, top 50, top 25, top 10 at its peak
  + For each track, pull Spotify data for the track (e.g., album, danceability, tempo, genre, etc.)
  + For each track, using the date it first entered the charts, query twitter to get every tweet about the track for the \_\_\_ days preceding.
  + Extract some quantitative metrics about those tweets (# of tweets, # of retweets/replies/favorites/etc.) and some NLP data from the content of the tweets.
  + In addition, select some number of songs spanning “this period of time” that did NOT make it onto the billboard charts, and repeat the process. (I need to flesh out how I’m getting these songs, and what date I’ll be using as a reference for each to pull twitter date)
  + Split the data into train/test subsets
* Training the model(s):
  + Model 1 – fit a logistic regression to determine what features predict if a song will make it onto the charts or not, using a binary variable that differs between the songs in the training dataset which did/not make the billboard charts
  + Model 2 – fit a linear regression (for only songs that were on the charts), to determine which features predict overall chart popularity (using either the track highest position feature or the “song impact” feature I constructed)
  + Model 3 – fit a decision tree (e.g., random forest, extra trees) that classifies whether a song will be in the top 100, top 50, top 25, top 10, or not make the charts, using our constructed class feature.
  + Use the test subset of the data to asses the performance of each model, and see which bits are performing the best
* The product:
  + A web app homepage has a single button that you click to run the model. Upon click, the web app:
  + (in the background)
  + 1. Pulls the newest tracks from the last \_\_\_\_ days from Spotify (maybe I filter for ones that have been around for at least \_\_\_ days or have \_\_\_\_ listens already), and the associated Spotify data fields
  + 2. Pulls twitter data for the last \_\_\_\_ days preceding the current date for tweets that contain the track
  + 3. Cleans the data in the same manner as was done in pre-processing for model fitting
  + 4. Feeds all tracks into model 1, and determines which ones are likely to make the top 100
  + 5. For tracks that are likely to make top 100, feeds them into model 2 to determine predicted impact
  + 6. Alternatively (or in addition), feeds all of the tracks into model 3
  + 7. Produces a list of \_\_\_ songs that are predicted to have a high impact in the charts in the future and adds these to a new playlist on my personal Spotify account.
  + (on the page)
  + 8. Provides (graphically) the track-listing on the page, displays the playlist’s “album art”, and creates a link to open that playlist in their Spotify client

**Possible Considerations**

1. I could add a component to the algorithm that sorts the playlist so that it is most aesthetically pleasing for the listener instead of ordering it from descending predicted impact
2. I could make separate playlists that are split by genre

OR

1. I could download billboard-top-100 by genre, and then separately train models for each genre’s top 100. Then I would perform a genre-split before running Spotify’s new tracks through their respective models and generate a playlist for each genre.

**Concerns**

1. It doesn’t look straightforward to get historical data from twitter. I’m still looking at some different python packages and communicating with someone who has the last 2 years of data.
2. It may be hard to get historical Spotify data (e.g., for a song that was on the billboards top 100 in 2016, how many listens did it have in 2016?) This isn’t essential to the project, however.
3. I have no experience in NLP which I intend to use for the twitter data. I am mostly confident that I can pick it up quickly over the weekend at least enough to run “data.fit” with it.
4. I have no experience creating web apps. I worry this will be quite time consuming.
5. When (i.e., relative to when) and for what duration am I pulling tweets for a track that is in my training dataset? 2 weeks prior to its first appearance on the charts? For 2 weeks a month before its first appearance? Unclear.
6. I need for my training dataset to include tracks which were not in the billboard top 100 to inform models 1 and 3. How do I randomly select these? Concern (5) is even more difficult here: what relative benchmark do I have to pull twitter data from?