CAPSTONE PROJECT NO.2

SPOTIFY SONG RECOMMENDATION SYSTEM

SITARA ABRAHAM
SPRINGBOARD 2019

TABLE OF CONTENTS

PROJECT PROPOSAL

1. PROBLEM

One of the most satisfying feelings is finding a new song you can't stop listening to. However, more likely than not, most songs people come across don't align with their tastes. A problem every fan of music runs into is finding artists or songs similar to what s/he already loves. Now especially, when people have millions of songs waiting for them in an app, this problem turns what should be a relaxing/motivating/fun experience into a magnified inconvenience.

2. CLIENT

The client in this scenario would be anyone who has a Spotify account and would like to find new music recommendations based on what songs they've already listened to or have favorited. Spotify has a similar user taste-based playlist, but for the sake of this project's hypothetical situation the client would not have a music recommendation system in place in their app. Using this model, they'll be able to see what songs they might like without having to listen to them.

3. DATA

I will be pulling the data from the Spotify API with the help of the Python package made specifically for this, spotipy. With this API, I can view playlist and track features.

4. SOLUTION APPROACH

I will utilize two separate lists of songs to analyze: one list of songs I like, and the other list of songs I dislike. I will then extract the audio features of the songs (i.e. acousticness, liveness, loudness, tempo, etc.) and use these features to build a logistic regression model to classify a song as one I'd like or not. I will use 80% of the data as training data, while the remaining 20% will be for testing.

5. DELIVERABLES

The deliverables for this project will be the project code and a slide deck.

DATA WRANGLING

For this project, I chose to retrieve song data through Spotify using a Python library made specifically for this, Spotipy.

API

The first step in this API process was to execute initial commands in the command line, declaring the client ID, client secret, and the redirect URL. After retrieving a token for the session using these details in addition to declaring an API scope as a variable (in this case the scope was modifying a public playlist), it was time to actually retrieve the data.

The strategy for this predictive model included making two playlists, one with songs I personally like and another playlist with songs I dislike. To do this, I compiled public playlists from Spotify and transferred the songs into their respective Good or Bad playlists on my own account.

The playlist data was in the form of nested dictionaries; the main dictionary I needed was “tracks” (which were essentially pages of overall data) and under tracks, I needed “items” (the song data).

I looped through a list of other playlists of songs I liked, created an empty list of songs, appended the playlist songs to the empty list, and added the songs to my new playlists. I did this process separately for the Good and Bad playlists.

For each playlist of songs, I extracted the song ID’s. I then used these ID’s to extract the audio features and transform these features into the working dataframe to be used for the rest of the project.

NULL VALUES

Because I pulled this data through a well-documented API, there were no null values for each of the track items.

DATA STORY

PROBLEM

One of the most dynamic interests, throughout the ages and geographic location, is music. From being the voice behind the lyrics, to writing the lyrics, to producing the sounds, to blending songs together in a mix for energetic crowds, to simply listening--there is rarely a person you will meet who completely dislikes music altogether.

That being said, we all love the feeling of discovering a new song that we end up playing on repeat. But inevitably, we grow tired. Bored. And yearn for the novelty of a new song that aligns with our moods and tastes. For some, this search for new songs is thrilling, the backbone of their love of music even. For others, it's a time consuming process with usually little to yield as a result.

OVERVIEW

The aim of this project is to build a model that utilizes a user's current musical tastes to predict whether s/he will enjoy songs in new playlists. Currently, Spotify has a very intelligent recommendation system; it's "Discover Weekly" playlists are tailored to each user and provides new songs which align with the user's historic tastes.

DATA

A major barrier to this being a usable model amongst multiple Spotify users is the fact that I would not have access to anyone else's account data except my own. For the purposes of this project, these limitations are understandable.

The first hurdle was retrieving and wrangling the data of interest. Using Spotify's well-documented API and a Python library package made specifically for this purpose (Spotipy), I was able to use my API credentials to pull in my music data under the scope of modifying public playlists.

After creating two playlists in my account, one for songs I like and one for songs I dislike, I was able to extract the audio features of these songs and put all this data into a new dataframe.

To be able to recommend new songs to the user, the key features in this model are the following:

- Acousticness
- Danceability
- Duration in milliseconds
- Energy
- Instrumentalness
- Key
- Liveness
- Loudness
- Speechiness
- Tempo
- Time signature
- Valence

After confirming that there were no null values in the dataset and seeing the breakout between songs I like and songs I dislike (838 liked songs, 776 disliked songs), I used the describe method to take a preliminary look at the ranges of values among the features.

In terms of an initial hypothesis, it looked as though I seem to enjoy more lively songs. The majority of values of certain features such as danceability, tempo, and valence were on the higher end of the individual spectrums, indicating the majority of the liked songs had high values in these areas.

I then visualized these ranges for each of these key features. While most of the features had a minimal or reasonable amount of outliers, some of them were very skewed by outliers. Because the validity of the recommendation might be adversely affected by these skews, the features “time signature” and “instrumentalness” were dropped from the dataframe.

Next, I visualized the distribution of the remaining features. Again, while some features had reasonably normal distributions, others had blatant skews or even bimodal distributions. To achieve the maximum validity of the model, the features with the least normal distributions (“speechiness”,”liveness”,”key”,”acousticness”) were dropped.

HYPOTHESIS

Based on the range of values in the remaining features, my initial hypothesis is that the mean values for valence, tempo, loudness, danceability, and energy are different in songs that I like versus songs that I dislike.

EDA

EDA OVERVIEW

After using the Spotify API to retrieve my song data and cleaning up the dataset to features I will need, I then tested my preliminary hypotheses that the songs I like have overall different values of audio features than the songs I dislike.

Looking at the range of values for each of these features for each liked and disliked songs, it looked like the 75th percentile for each liked feature and its disliked counterpart were substantially different.

Before hypothesis testing, it was important to confirm a few assumptions about the data: normality, independence, and the dataset's relevance to the Central Limit Theorem.

NORMALITY

In the Data Story notebook, the normality of the continuous features was explored; features with distributions clearly deviating from being Gaussian were pruned from the dataset. However one feature, Mode, is binary and was not clumped with the continuous columns.

Using bootstrapping, ten thousand bootstrap replicates of the mean proportion of Mode were taken. Looking at the histogram of these replicates, the distribution was confirmed to be approximately normal.

INDEPENDENCE AND CLT

The audio features of each track in each playlist are not dependent on any other song in either playlist, so these observations can be classified as independent.

Since the general rule of thumb is that sample sizes of at least 30 are typically large enough for repeated sampled means to be normally distributed, this sample size (1,614 observations) is sufficiently large.

HYPOTHESIS TESTING

While there were multiple features which were possible to use for hypothesis testing, I used what I felt were the three most important features as a proxy for the rest. Based on the Spotify definitions, I determined the following features as being most indicative of my tastes:

- Tempo
  - "The overall estimated tempo of a track in beats per minute (BPM). In musical terminology, tempo is the speed or pace of a given piece and derives directly from the average beat duration."
- Danceability
  - "Danceability describes how suitable a track is for dancing based on a combination of musical elements including tempo, rhythm stability, beat strength, and overall regularity. A value of 0.0 is least danceable and 1.0 is most danceable."
- Valence
  - "A measure from 0.0 to 1.0 describing the musical positiveness conveyed by a track. Tracks with high valence sound more positive (e.g. happy, cheerful, euphoric), while tracks with low valence sound more negative (e.g. sad, depressed, angry)."

My null hypothesis for each of these features was that the mean value for said feature would be the same for both songs I liked and songs I disliked; the alternative hypothesis was that the mean values would be different.

I initially separated the dataset based on songs I liked (value of 1) and songs I disliked (value of 0). I then plotted the distributions and ECDF's of the feature values of the respective datasets. Looking at these plots, I could gauge a slight difference in values however further testing was needed to confirm the difference.

I then took replicates of permutated samples of the dataset for each feature, and recorded the replicated difference of means between the two samples. Using an alpha of 0.05, I calculated the probabilities of getting a difference of means higher than the observed difference.

After running this process for all three features, there was not sufficient evidence of a statistically significant difference of means of tempo, danceability, and valence between songs I liked and songs I disliked.

IN DEPTH ANALYSIS

OVERVIEW

After performing exploratory data analysis on the audio features of the Spotify dataset of songs I like and songs I dislike, I had initial thoughts on what values of features I would expect from songs I liked and disliked.

I then refined these hypotheses, and used inferential statistics and distribution visualizations to test the hypotheses. Because I aimed for the final product to be a holistic recommendation system for songs, I opted for multiple machine learning models to be worked in conjunction to predict whether I will like a song or not.

HYPOTHESIS

In the exploratory data analysis phase, I fleshed out the following hypotheses:

TEMPO
H0: The mean tempo is the same for songs I like and songs I dislike.

H1: The mean tempo is different for songs I like and songs I dislike.

DANCEABILITY
H0: The mean danceability is the same for songs I like and songs I dislike.

H1: The mean danceability is different for songs I like and songs I dislike.

VALENCE
H0: The mean valence is the same for songs I like and songs I dislike.

H1: The mean valence is different for songs I like and songs I dislike.

a = 0.05

With this stated alpha, there were surprisingly no statistically significant results. For each of these three features, the p-value was much higher than 0.05, indicating that my favor of a song isn’t dependent on a consistent range of values in tempo, danceability, or valence.

MACHINE LEARNING METHODOLOGY

Ideally, the end product for this project would be a comprehensive recommendation system, one that uses a mix of machine learning models to aid each other. For the sake of this project, I chose three ML models: a decision tree classifier, logistic regression, and KNeighbors classifier.

I wanted to include a decision tree in this mix because of relatively high number of dimensions; even though it is still separating the data set into two classes this model has its advantages because the boundary between songs I like and songs I dislike is not necessarily linear (as seen by the hypothesis testing).

However, a complication arises because of this same lack of linear boundary; because the classes are not well separated, a decision tree might be susceptible to over-fitting. Because of this, a logistic regression model might simplify the classes more effectively.

Finally, I wanted to round out the mix by using another supervised model that clusters data points: K Nearest Neighbors.

For each model, I started by instantiating a model with no specified optional parameters, and then looked at the accuracy score and classification report.

Then, I plotted a normalized confusion matrix for each model to evaluate the percentage of true and false positives. Finally, I conducted a cross validation grid search to find the best hyperparameters respective to each model and re-evaluated the accuracy score and classification report with the changes.

EVALUATION

For the sake of the context of predicting songs I like, using a metric as simple as accuracy score would suffice; the percent of songs I accurately enjoy out of a playlist recommended for me would be indicative of how successful the models are.

For the decision tree, out of curiosity more than anything, I plotted an ROC curve along with retrieving an accuracy score. The curve was far below the .5 threshold, indicating a low-performing model in terms of predictive quality.

Before any hyperparameter tuning, the accuracy scores were as follows:

- Decision Tree: 29.7%
- Logistic Regression: 49%
- K-Nearest Neighbors: 21.1%

Looking at these results, the logistic regression model would have the most predictive weight.

Once performing a CV Grid Search on the models, the accuracy scores were as follows:

- Decision Tree: 45.2%
- Logistic Regression: 49.2%
- K-Nearest Neighbors: 31%

The decision tree gained the most improvement with the parameter tuning, whereas the logistic regression model more or less stayed the same and continued to have the highest score.

CONCLUSION

While these scores can seem to be pretty low, in the context of Spotify's current recommendation system-fueled playlists I think these models are on par. For the Discover Weekly playlist Spotify tailors to each user, I don't like most of the songs recommended. That being said, I think the scores of these models when used in conjunction would perform decently in terms of predicting songs I'd probably enjoy.

At the risk of overfitting, I think the scores would definitely improve had I left the other audio features in the dataset for the models to train on. However, for the sake of this project and applying ideal practices, I opted to include only features with minimal outliers and with generally normal distributions.

In retrospect, there were further ways the model could have been improved:

- Tuning other Decision Tree Classifier parameters
    - Min_weight_fraction, max_features, min_impurity_split
- Adding a Random Forest Classifier
- Utilizing PCA to add vector features to make up for the reduction of other audio features

In terms of next steps, the whole exploration of the dataset and the resulting model has produced valuable insights in terms of my taste in music in the context of how lively or positive a song is. As it turns out, there is no significant distinction in these traits for songs I like and dislike. I can obviously confirm this conclusion, I enjoy both lively and more mellow songs, happy and perhaps more emotional.