Song Affinity Prediction

Mary Neuburger & Mason Heaman

What we set out to do

Background

Large companies such as Spotify often use algorithms to suggest songs based on songs a user has already liked. However, these algorithms are often not made available to the public, and as such are not well understood.

What we're doing

Our goal is to attempt to create a basic model that emulates the function of the algorithms large companies use. It predicts whether we would like a given song based on whether we liked other similar songs in the dataset we trained it on. We also wanted to find trends in the dataset using general popularity as reported by Spotify.

Why it's interesting

Music is a large part of our lives. We often use it as background noise while working on other activities, to set a mood, or to reflect our own mood. Music platforms such as Spotify, Pandora, and many others take on the task of reccomending songs based on user info. However, their methods of prediction are kept from users. We want to see if we can create a reliable model for predicting if a song will be liked by a user.

The dataset

We wanted to acquire a large set of songs that models our individual taste. Therefore, we hoped to acquire an even number of liked and disliked songs for both members. To do this, we pulled from three Spotify playlists: Spotify Top 100[1] , 2021 Wrapped (Mason)[2], and 2021 Wrapped (Mary)[3]. This provided us with a 300 song set that was then rated as liked or disliked by both members. We used the Organize Your Music webtool to retrieve metadata for each song. This tool utilizes the Spotify Web API and returns the following features:

> Genre - the genre of the track
> Year - the release year of the recording. Note that due to vagaries of releases, re-releases, re-issues and general madness, sometimes the release years are not what you'd expect.
> Beats Per Minute (BPM) - The tempo of the song.
> Energy - The energy of a song - the higher the value, the more energetic song

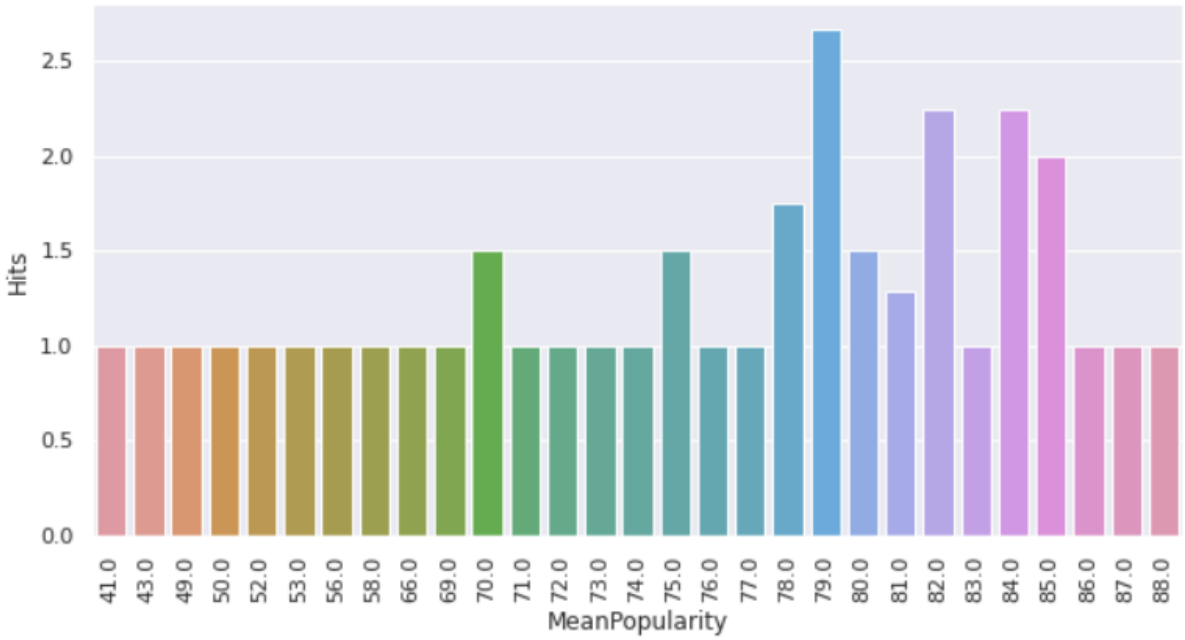
> Danceability - The higher the value, the easier it is to dance to this song.
> Loudness (dB) - The higher the value, the louder the song.
> Liveness - The higher the value, the more likely the song is a live recording.
> Valence - The higher the value, the more positive mood for the song.
> Acousticness - The higher the value the more acoustic the song is.
> Speechiness - The higher the value the more spoken word the song contains.
> Popularity - The higher the value the more popular the song is.
> Duration - The length of the song.[4]

Along with our affinity scores, this provided a set of 14 features for our analysis. Out of the 300 songs, Mason liked 168 while Marv liked 132.

Exploratory Analysis

Artists and Popularity

We wanted to look for trends in our data by comparing popularity to a couple of different factors. First, we wanted to see if artists appear more often in the top 100 songs if their popularity was higher. We calculated the artists' popularities by adding up the popularity score of an artist's hits on the list and then dividing that number by the amount of times that artist appears on the list, creating their mean popularity. We then split the data down the middle and calculated the mean number of hits in each half to see if there was a difference in hits by each half, which there was. The bottom half of the popularity dataset had 1.37 hits per artist and the top half had 1.69 hits per artist, meaning that higher popularity artists had, on average, more hits in the top 100. We wanted to visualise that, so we created a bar chart of popularities.

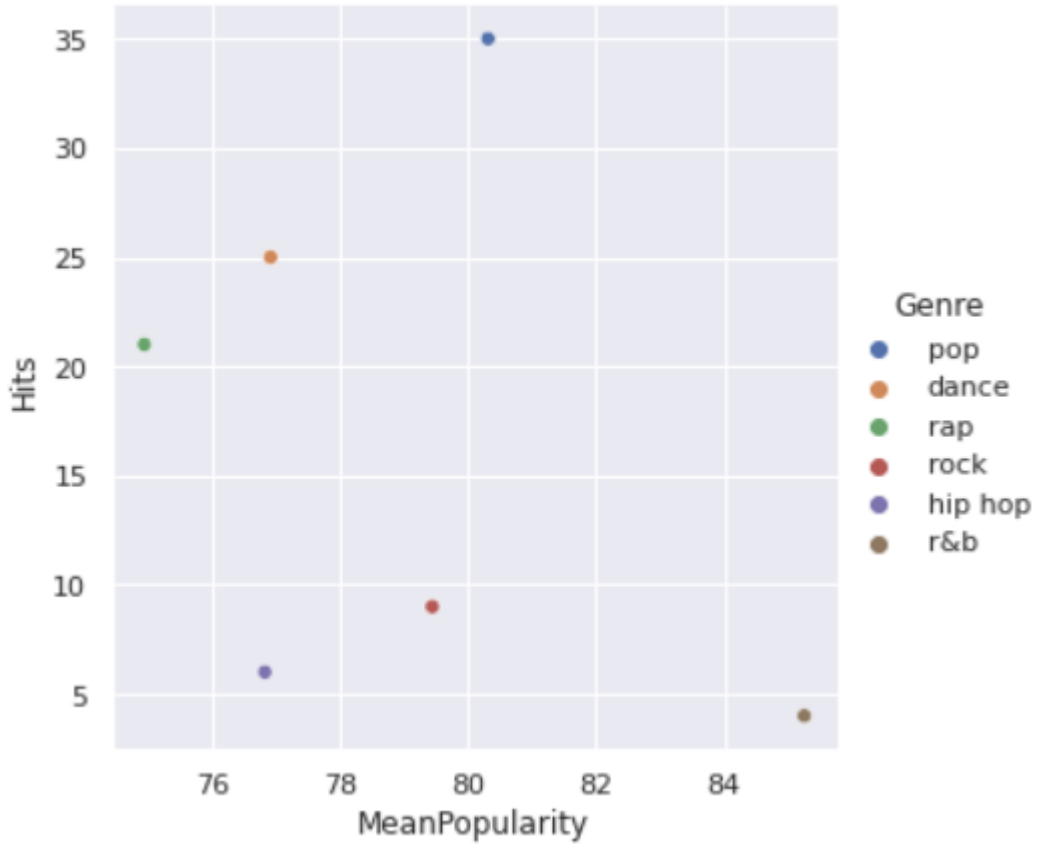


As shown above, most of the artists only have one hit in the top 100. Most of the artists who have more than one hit fall into the higher popularities, so we think it's likely that artists who have high popularities will have more hits in the top 100. One of our concerns is that we calculated artist

popularity by the number of songs already in the top 100, so true artist popularity is probably different from the number we calculated. The artists who are most popular on average have only 1 hit, so it is possible these artists are one hit wonders rather than truly popular artists.

Genre and Popularity

Our second interest in the dataset was whether any given genre was more popular, and if that meant it had more hits in the top 100. We calculated the mean popularity for each genre the same way we calculated the mean popularity for each artist, by adding up the popularity of each song in the genre and dividing by the total number of songs in the genre. The results confused us a little, as the most popular genre only had 4 hits, while the least popular genre had 21 hits. As visualisations had helped present the findings a little clearer previously, we decided to make a scatter plot of the data we had just gathered.

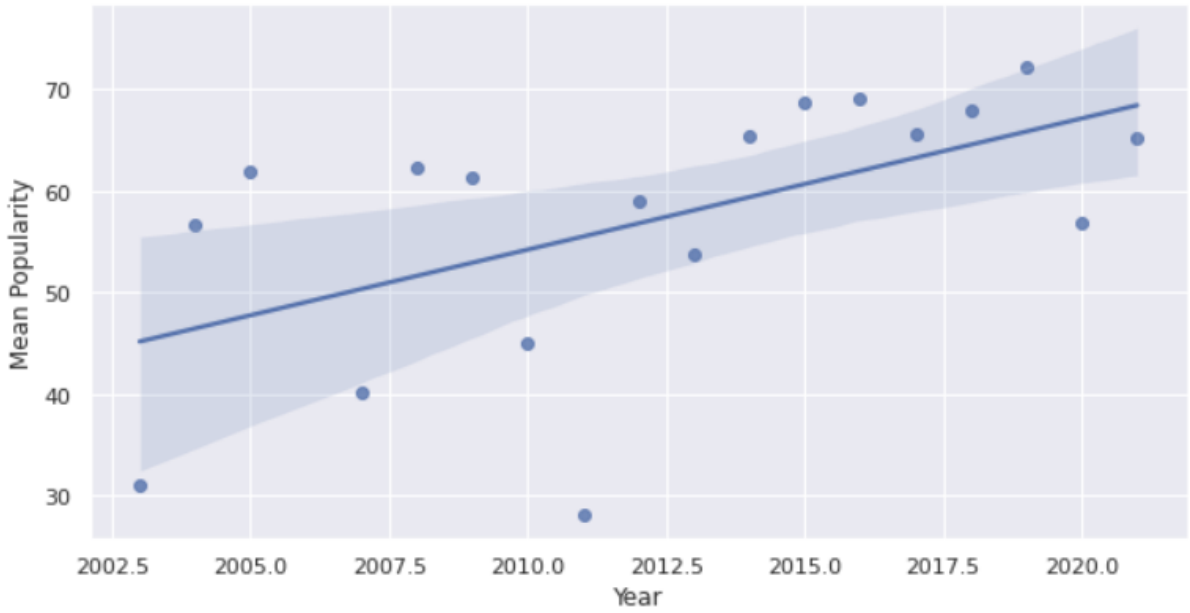


The scatter plot made it clearer to us that there wasn't much of a relationship between popularity and hits of a genre in the top 100 songs. The genre with the most hits had average popularity, while the most popular genre had the fewest hits. Once again, we were concerned that the conclusions we came to were inaccurate because of how we calculated the mean popularity of a genre, given that we only based that number off of the data from the top 100. We felt that the mean popularity of the genres with more hits in them were probably closer to the genre's actual mean popularity; however, we were unsure of how close the calculated mean popularity of the genres with fewer hits were to the genre's actual mean popularity.

Release Year and Popularity

Our last interest was to see if there was a relationship between the year a song was released and its popularity. For this, we could use our entire dataset instead of just the top 100 songs. Just like in our previous explorations, we calculated the mean popularity of each year by adding up the popularities of the songs released in a given year and dividing that number by the number of songs released in that year. Because the data contains songs released in a lot of years, we decided to graph the data first before drawing our initial conclusions.

We drew a line of best fit on this graph to better visualise the trend in popularity as the songs get more recent. What surprised us was that the trend was negative, and songs released before 2000 were extremely popular. When we looked closer at the years, we realised each point before 2000 was only based on 1 song, so it was possible that these points were actually outliers. Our reasoning for why these particular songs were so popular was that they were old enough for the listener to experience nostalgia when listening to them. Because of that, we decided to exclude them from the graph to see if we could isolate a more accurate trend based on years with more hits in them.



Once the points in question had been removed, we could see a clearer trend in the modern data. The trend in popularity by year seemed to be positive rather than negative, so we felt that the removal of the years before 2000 was justified. Our biggest concern with this analysis was with the shaded area around the line of best fit, which represents the certainty that the regression algorithm has that the line of best fit actually fits the data. When the shaded area is larger, there is less certainty. Because of the size of the shaded area, we believe that there is a relationship between release year and popularity, but the relationship is not strong.

Predicitons

Splits

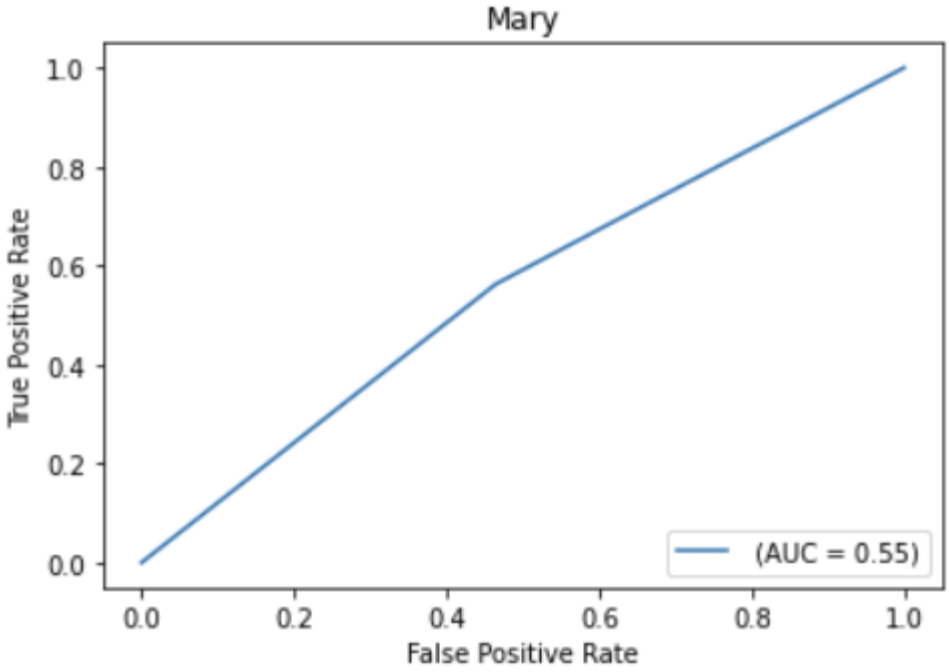
The 300 x 14 dataset of song features was split into three randomly sampled sets: train, validataion, and test (180, 60, and 60 songs respectively)

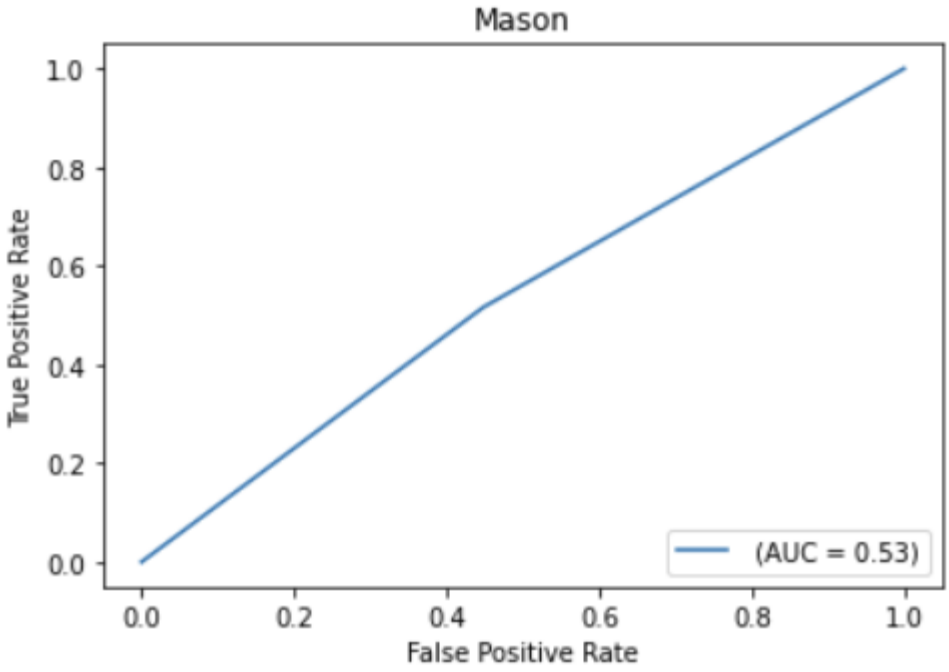
Evaluation Metrics

The models will be evaluated using the Area Under the Receiver Operating Characteristics or Area Under Curve (AOC). This is a performance metric for classification problems that uses true positive rate and false positive rate to measure how well a model can distinguish between classes. A higher AUC score represents a model that can better predict targets correctly.

Baseline

A probability based baseline was used to demonstrate our models validity. This baseline assumes that ratio of songs liked in the dataset will generalize (165/300 or 132/300) to other songs, and uses this as its only feature. On a test set of 60 songs, the probability baseline will assign songs a 0 (dislike) or a 1(like) as if it were flipping a coin with weighted probabilities.





The ROC curve and AUC metric displayed above show that this baseline is slightly more effective at predicting affinity than a random guess.

Preprocessing

In the raw dataset, the genre of songs is reported as a string and is far too descriptive to be reliably used as a categorical feature. Genres such as canadian contemporary r&b and 10 different sub categories of hip hop made up the 41 unique genres in the set. To solve this, we fit each of the subcategories into a set of 8 base genres that made up the majority of songs: country, rap, pop, rock, hip hop, r&b, dance, and metal. We then took two different aproaches to incorporate this feature: ordinal encoding and one-hot vectors.

Ordinal encoding

The genre feature remains a single column in our dataset, where each genre string is converted to a numerical value (1 for country, 2 for rap, etc.)

One-Hot vectors

The genre feature is expanded to 8 seprate features in separate columns. Each row will have a 1 in the column that corresponds to its genre, while all other columns will be blank.

The separate methods of genre preprocessing resulted in two separate data sets to be passed into our models.

Models

Three models were trained for this predicion task: Random Forest, Logistic Regression, and KNeighbors. Each model was trained seprately on the ordinal and one hot datasets. Below is the reported AUC scores on the validation sets for each model.

Random Forest

- One-hot
  - Mary: 0.81
  - Mason: 0.89
- Ordinal
  - Mary: 0.81
  - Mason: 0.83

Logistic Regression

- One-hot
  - Mary: 0.72
  - Mason: 0.79
- Ordinal
  - Mary: 0.75
  - Mason: 0.78

KNeighbors

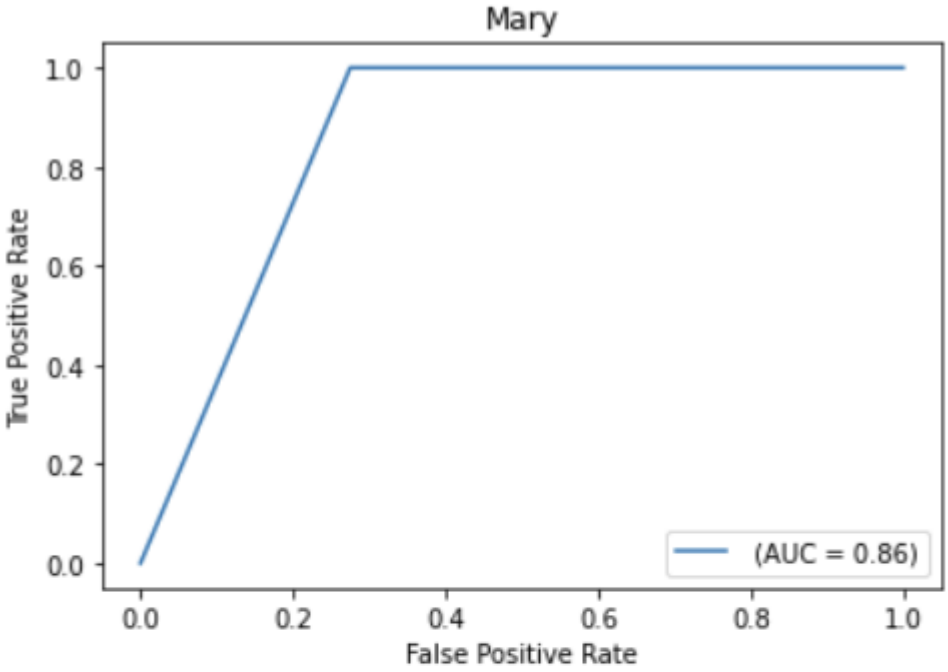
- One-hot
  - Mary: 0.71
  - Mason: 0.75
- Ordinal
  - Mary: 0.71
  - Mason: 0.75

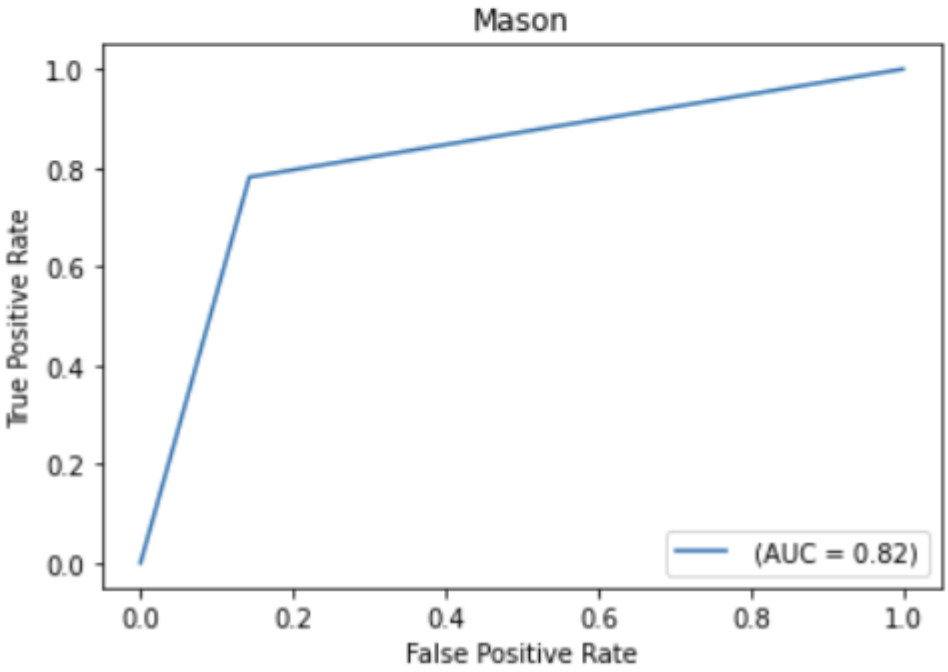
Our two highest performing models were Random Forest and Logistic Regression. At an earlier state in the project, we had a smaller dataset that only pulled from the Spotify Top 100. Under these conditions, we saw that the Random Forest model drastically overfit to the training data and was unable to generalize out to new songs in the test set. This caused us to believe that Logistic Regression was the ideal model for our prediciton task.

However, increasing the set to 300 songs with a more even ratio of likes to dislikes solved this issue. On the improved dataset, Random Forest outperformed Logistic Regression on the validation set and was our chosen model going into testing.

Results

On the test set, we were able to achieve good AUC scores with the Random Forest model. See





Notebooks

Exploratory Analysis

Song Affinity Prediction

Citations

1: https://open.spotify.com/playlist/5ABHKGoOzxkaa28ttQV9sE?si=64e1b98de40a4fcf

2: https://open.spotify.com/playlist/0Ae3qo4E8cZ44yJXnLw75M?si=c343ba54dc414fbc

3: https://open.spotify.com/playlist/3RTZnQ47H8NRmZd650Yfee?si=639946eabca04223

4: Lamere. (2016, August 6). Organize Your Music. Organize Your Music. Retrieved December 6, 2021, from http://organizeyourmusic.playlistmachinery.com/