Predicting Song Release Date Using Billboard and Spotify Data

Neal Manahan
General Assembly Chicago Data Science Immersive
May 15, 2019

As a major label artist, planning and preparing a successful release goes further than finding a great single or recording a critically acclaimed album. A strategic rollout is essential to standing out and cutting through noise in this highly competitive industry. The target of this project is to pinpoint an effective time to release new music. In order to achieve this, I decided to gather data from the largest music streaming service, Spotify, along with chart data from the industry publication Billboard to analyze the songs that see the most success while on the best selling/most listened to charts.

I started this projected by pulling in chart data using the billboard web api wrapper. I created a function that would return the track name, performing artists, and weekly chart position of the track. I nested this in a loop that would iterate over the last 3 years of charts and add each week's chart to a row of a pandas dataframe labeled 'hot_100'. I saved this dataframe as a CSV for use in the cleaning/organization notebook.

I then collected pre-formatted CSV files from the Spotify Charts website. These CSV files contained weekly reporting for the top 200 songs in the United States with track name, performing artists, and total stream numbers for the week. I used the glob module to call in all the files at once into the jupyter notebook and save each track to a row in 2 pandas dataframes with a loop. One of the pandas dataframe (labeled 'position') is used to find the peak position of each track on the Spotify 200 chart, and the other dataframe is used to sum total streams for each track during the time period the track is on the charts (labeled 'frame'). I kept these two dataframes separate for organization - these could be saved to one if preferred. At the completion of these steps, I saved the dataframe to a CSV file for use in the cleaning/organization notebook.

After data gathering was complete, I moved to the second notebook in this repository to clean and organize the data prior to modeling. Before beginning this process, I applied for a Spotify API key at Spotify for Developers and got a personal key and secret key to authorize calls to Spotify's API. I called in the CSV files that were saved via the data gathering notebook. In order to get the peak position of a track from billboard, I created a loop that iterates over each row of the pandas dataframe. This loop contained both a set and a list - the set ensured that if a track had been seen already at a higher chart position, it would not be added to the list of unique tracks. I also saved the column location of the track to a list, which corresponds to the highest position on the weekly chart.

Next, the chart data from Spotify needed to be organized in two different ways from its original format. The data must be first organized by the chart position each week in the 'position' dataframe. Once ordered, the peak position was sourced from this data - I used the same loop as above for the Billboard chart data here. Next, the number of streams for each time the song appears on the top 200 chart needed to be totaled from the 'spotify_200' dataframe. Once organized, I combined these two unique data lists into one dataframe by joining on the track name.

Unfortunately, Billboard does not disclose sales data compiled from Nielson Soundscan reports that are used to create their charts - which are considered the industry standards for reporting. This meant I would only be able to use Spotify streaming data for the project. Future iterations of this model and project would include access to Nielson Soundscan's proprietary data to create a more accurate model.

Another issue that arose dealt with the format in which Billboard and Spotify listed song titles. Guest artists are included in the track name on Billboard, while on Spotify all artists are listed together. To combine the Billboard chart data with the spotify total streams and peak position, I created a loop that

would take each track name from the Spotify 200 dataframe and look for matches in the Billboard Hot 100 dataframe. If the track name existed in both charts, they were saved to a new dataframe for combination later.

Next, I used the track name from the new collection of songs in both Billboard and Spotify charts to search for song information using the Spotify API. I created a loop that took each name from the list and searched the Spotify database and returned the information from a dictionary that I saved to a list. I chose this method to limit the number of calls to the API and collect all the information at once. Now that the information was saved locally to the list, I pulled out specific values from the track's dictionary using a loop to gather release date and the performing artists' names. Using the artist name pulled from the track information, I then searched the Spotify API for information on the artist. Using the same loop as I did song information (with minor tweaks), I saved the artist information dictionaries to a list. I then pulled out the artist genres, artist popularity metric, and number of followers from the artist information list. Artist's genres were saved as a nested list, so I used the join method to combine these values into a single string. Once this information was collected, I added it all to the dataframe used for modeling.

I had some null values as a result of missing information from the Spotify API, so I determined that I would drop these items as there were very few. Finally, I used the release date information to create a column with the release month, which I anticipated being my target variable. I then saved this information as a CSV for use in my exploratory data analysis and predictive model.

Moving into the EDA and modeling notebook, I began by using count vectorizer to transform qualitative 'genre' column for modeling. I hypothesized that genre may have an impact on release time frame and song performance. Through some initial data investigation, I decided to collect a max features count of 10, since there were many variations on the major genres that kept appearing. I set up the countvectorizer with an n_gram range of (1, 2) to vectorize the most commonly used single word and 2-word phrases in the corpus. Once the 10 features were transformed, I split combinations of the word vectors into 4 genre categories: Pop (1), Hip-Hop (2), Other(Rock, Country, Holiday, World) (3), Pop Rap (4).

I began some data analysis by looking at the relationship between the target variable (month released) and the features I would be using for my model (number of streams, artist popularity, artist followers, genre category). Using a heatmap, I identified that the top three most correlated features were the vector representing the word 'trap' (a subgenre of rap/hip-hop), the 'music' word vector, and the genre category feature. The three most negatively correlated features are artist followers, the 'rap rap' word vector, and 'pop rap'. An interesting initial finding here is that based on this scale, as the 'value of the months' (1 = January, 2 = February, etc.) increases, the number of artist followers decreases ever so slightly. Unfortunately, based on the results of this heatmap, there are no significant linear relationships in this data. While the lack of correlation is troubling, further EDA showed that songs released in Q2 (Late Spring) and Q4 (Winter/Holiday Season) spend time on the Billboard and Spotify Charts. Songs released in April and June have substantially more plays than those released any other month of the year. Finally, I wanted to explore my hypothesis of genre factoring into release date. Measuring the most popular artists based on genre, artists with roots in the hip-hop and rap genres have the highest popularity, and those with pop crossover characteristics sit at the top.

At this point, I decided to prepare my model for predicting the release date of a track. Due to my findings during EDA, I decided that I would first try to predict the Quarter/Season of the release. Without strong relationships between my target and the features, I was unsure a classification model would accurately predict 12 discrete classes, instead opting for 4 seasonal classes. I split my data into 4 classes by numerically representing each quarter based on month of release. January-March were labeled Q1 (Late Winter/Early Spring) , April-June Q2 (Spring), July-September Q3 (Summer), October-December Q4 (Fall/Holiday). Classes were not perfectly balanced over the 4 quarters, so I stratified the classes when creating the training and testing the model.

For features, I used number of streams, artist popularity, artist followers, and the genre category from the Spotify API, peak position on the Billboard Hot 100 Chart, and peak position on the Spotify United States Top 200 weekly chart. I split this data into a training set of 70% of the data, leaving 30% of the data unseen by the model for testing. I chose these variables after careful consideration of the available data and in order to answer my question based solely on listening habits. After splitting the data from training and testing, I scaled my data because of the drastically different distances of each feature.

I then turned my attention to outfitting and running the model. I chose to use a Random Forest Classifier to determine the season based on the interpretability of feature importance and the tendency of the model to reduce overfitting, which I anticipate being an issue based on data quality/availability. Because I was unsure of how exactly I wanted to tune the parameters of my model, I chose to implement a randomized search cross validation to key in on some values to set my parameters. I chose to search over number of trees (n_estimators), the purity of each decision leaf (min_samples_leaf), number of samples required to split (min_samples_split), how many levels the trees would contain (max_depth), and whether or not to sample with replacement (bootstrap). Ultimately, the best parameters were a 1666 tree forest, a max depth of 100, splitting at a minimum 5 samples, a purity of 1 sample per split, and sampling without replacement. The training data resulted in a perfect accuracy score over 3 sample folds of validation with the best parameters, but the testing data accuracy was only 52.7%. Because of the model was overfit, I decided to hone in on ranges of the best parameters from the randomized search. The best parameters changed slightly, but there was no change in accuracy score.

The model was the most accurate at predicting releases from Q2 correctly, and had the hardest time classifying songs from Q3, which was the class with the fewest songs represented. Upon analysis of the feature importances, artist followers was the most important feature in determining class. This was not surprising since artist followers had the strongest negative correlation with the target variable. The least important feature turned out to be genre category, which disproved my hypothesis about genre leading to release date prediction.

Based on the performance of the model, I looked into prediction and probabilities of each track and class selection to make some conclusions. While the model was overfit, it was still better than the baseline of the majority class. I also noticed that the probability of the correct predictions of the test data was higher by roughly 12 percentage points, leading me to believe there is a trend that can be uncovered with more (and better) data. As I think about next steps for this project, I would look outside of Spotify for additional data points. The most easily accessible data that could add value would be through social media platforms - analyzing artist follower counts and interactions. Additionally I believe the model would perform considerably better with access to the full breadth of data available from Nielsen Soundscan, and other streaming platforms that are not as transparent as Spotify.

Music consumption trends will continue to morph as technology continues to improve, and at any moment the next number one can be released online. Inversely, a great song can end up lost in the shuffle of the the countless other songs released every day and never receive its due attention. With the breadth of data available thanks to consumer streaming technology, planning song releases and album rollouts can evolve into an analytic and strategic process.