Predicting Spotify Popularity

Christopher Goodwin

Bellevue University - DSC680

Abstract

Spotify is the world's biggest music streaming platform, with over 286 million monthly active users. Users of the service have access to one of the biggest-ever collections of music in history, plus podcasts, and other audio content. While Spotify has had steady growth since its creation in 2008, there was a noticeable sharp uptick in both users and subscribers around 2015. This could be attributed to the introduction of new Spotify-curated playlists. (Iqbal, 2020) These playlists, as a matter of fact, have great influence on the popularity of songs. Playlists in particular are instrumental in guiding consumers to specific content, according to a new working paper by researchers at the European Commission's Joint Research Center and the University of Minnesota. (Passy, 2018) In this paper, we will look at what other factors make a song popular, and see if we can predict an individual song's popularity on Spotify.

Predicting Spotify Popularity

What makes a song popular on Spotify? There may be obvious answers, like if an artist is already popular, their new song will likely be popular. Or if a song is constantly played on Top 40 radio, it will likely be streamed more at home as well. Or if a record label pours money into advertising and promotion. We want to look deeper than that, though.

Instead of looking at industry factors like producer, artist, or record label, we will be looking at information about the song itself. What year did it come out? Does it have a fast or slow tempo? How loud is it? Is it explicit or clean? Incorporating these factors into a regression model, we hope to predict a song's popularity.

The Data

We need to start by examining the data we will be using in our analysis. The *Spotify Dataset* was created by Kaggle user Yamac Eren Ay, and contains information from over 175,000 songs, ranging from the 1920s to present day. Each song contains the following numerical features that we will use in our analysis: acousticness (Ranges from 0 to 1), danceability (Ranges from 0 to 1), energy (Ranges from 0 to 1), duration_ms (Integer typically ranging from 200k to 300k), instrumentalness (Ranges from 0 to 1), valence (Ranges from 0 to 1), popularity (Ranges from 0 to 100), tempo (Float typically ranging from 50 to 150), liveness (Ranges from 0 to 1), loudness (Float typically ranging from -60 to 0), speechiness (Ranges from 0 to 1), year (Ranges from 1921 to 2020), mode (0 = Minor, 1 = Major), explicit (0 = No explicit content, 1 = Explicit content), and key (All keys on octave encoded as values ranging from 0 to 11, starting on C as 0, C# as 1 and so on). (Ay, 2021)

The data is in great shape, so we don't have to do much manipulation whatsoever. My main task with the data was to gain a better understanding of some of the terms. Below is an outline of what each term is evaluating: (Tiffany, 2018)

- Acousticness: A confidence measure from 0.0 to 1.0 of whether the track is acoustic. 1.0 represents high confidence the track is acoustic.
- 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.
- Energy: Energy is a measure from 0.0 to 1.0 and represents a perceptual measure of intensity and activity. Typically, energetic tracks feel fast, loud, and noisy.
- Instrumentalness: Predicts whether a track contains no vocals. "Ooh" and "aah" sounds are treated as instrumental in this context. Rap or spoken word tracks are clearly "vocal." The closer the instrumentalness value is to 1.0, the greater likelihood the track contains no vocal content. Values above 0.5 are intended to represent instrumental tracks, but confidence is higher as the value approaches 1.0.
- Liveness: Detects the presence of an audience in the recording. Higher liveness values represent an increased probability that the track was performed live. A value above 0.8 provides strong likelihood that the track is live.
- Speechiness: Speechiness detects the presence of spoken words in a track. The more exclusively speech-like the recording (e.g. talk show, audio book, poetry), the closer to 1.0 the attribute value. Values above 0.66 describe tracks that are probably made entirely of spoken words. Values between 0.33 and 0.66 describe

tracks that may contain both music and speech, either in sections or layered, including such cases as rap music. Values below 0.33 most likely represent music and other non-speech-like tracks.

- 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.
- 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).

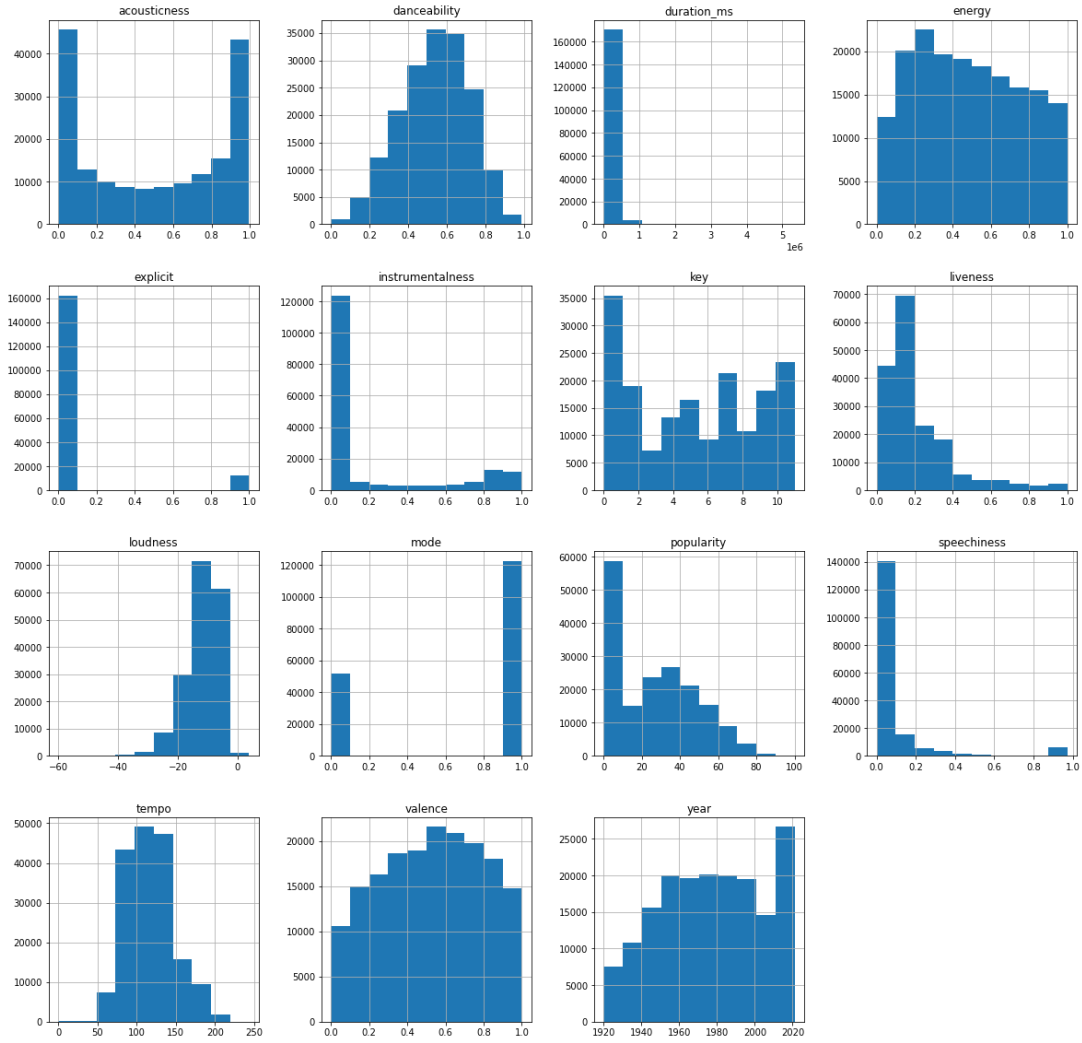
With all of this information, we can begin to perform our analysis.

Initial Analysis

Our first step was to really take a look at the data, and see if we could gain anything just from its general distributions. I created histograms for each of the variables, seen below:

Figure 1

Distributions of variables of interest

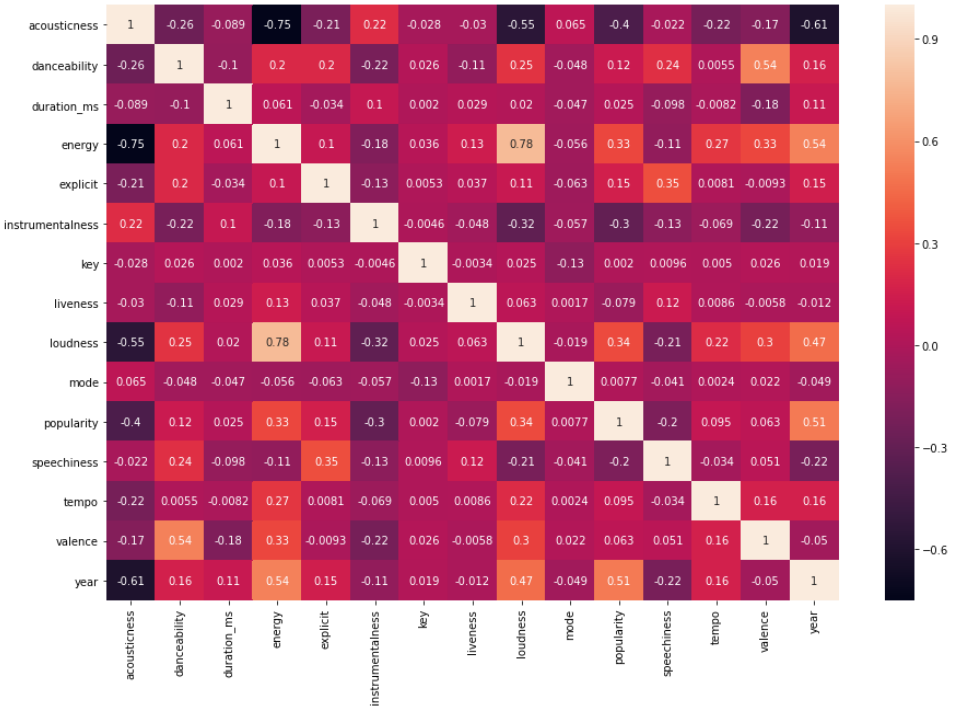


As we can see, based on the definitions of the different variables, nothing is really out of order here. A few surprises might be the number of explicit songs being so much lower, and the fact that a good chunk of songs have a low popularity rating. This concerns me a little bit at first glance, but I think it should be fine in the long run.

From here, I wanted to look at a correlation matrix of the data. Which variables have a strong relationship with popularity?

Figure 2

Correlation Matrix



If we use a cutoff of +/- .2, we see that there are a number of variables that have a pronounced impact on popularity: acousticness, energy, instrumentalness, loudness, speechiness, and year. We are going to keen in on these variables in our models and see if they have a large impact on their own.

Model Setup

I wanted to take a quick moment to describe how we set up our data in order to perform our models. We need to set up our features variable X, as well as our determining variable y. We will have two different examples of X- one with only the 6 features we determined to have a strong correlation, and then another one that contains every variable in the dataset. Our y variable is obviously the popularity scores for the songs.

The next step is to split our data into training and test datasets. This was done using the *scikit-learn* `train_test_split` function, with an 80/20 split. This yielded 139,511 songs in the training set, and 34,878 songs in the test set. One final step we took was to standardize the data, which often makes training more efficient. Standardization rescales our features so they have a mean of 0 and standard deviation of 1. This was done using the `StandardScaler()` from *scikit-learn*.

Random Forest Regressor

Now that we have our datasets, we can begin modeling. The first model I chose was a Random Forest Regressor. Decision trees learn how to best split the dataset into separate branches, allowing it to learn non-linear relationships. Random Forests build many individual trees, pooling their predictions. As it uses a collection of results to make a final decision, Random Forest regression is often referred to as an "Ensemble technique". (Ronaghan, 2018)

Once again, we utilized the *scikit-learn* package, using the `RandomForestRegressor()` technique. We fit the model to our training data, and then evaluated the accuracy on our test data. Using only the six variables of interest, we received an accuracy score of 61.7923, with an average error of 8.9386. At the surface, these values aren't great. However, when you think about the fact that popularity is on a scale from 1-100, an average error of ~9 points isn't horrible.

Since the numbers weren't very impressive, I wanted to try again using all of the features we have available to us. Does including everything muddy the waters even more, or does it help keen in on the popularity scores? Fitting the model to our new training set, I then calculated the accuracy score and average error: 65.8914 and 8.4287 respectively. Progress! Our accuracy score

increased by just over 4%, and the average error went down. Let's take a look at which variables had the most importance in this model:

Table 1

Importance Ratings for Features in Random Forest

Feature	Importance
Year	0.537230
Instrumentalness	0.077244
Loudness	0.051899
Duration_ms	0.046652
Energy	0.039294
Liveness	0.037326
Acousticness	0.037317
Danceability	0.035588
Speechiness	0.035446
Valence	0.035059
Tempo	0.034130
Key	0.014823
Explicit	0.014612
Mode	0.003382

The most striking thing from this chart is that year truly dominates the model, accounting for over 50% of it. This would make sense, since you would expect the recent songs to be more popular. However, we also see that some variables we excluding originally were actually significant to the model. It would appear from this that including every variable enhances the power of our model, so I will include them in all the models moving forward.

K-Nearest Neighbors Regressor

The next model we will use is the K-Nearest Neighbors (KNN) Regressor. see on the page that follows. K-Nearest Neighbors (KNN) makes a prediction for a new observation by searching for the most similar training observations and pooling their values. (Ronaghan, 2018) I felt like this would be a great option, since we could pool like songs together and hopefully determine their popularity.

I once again used *scikit-learn*, this time using the `KNeighborsRegressor()` option. I fit the model to our same standardized training data set, and calculated the accuracy and average error. For the KNN model, we only had an accuracy score of 53.1978 and av average error of 10.3205. We took a step back with this model, dropping almost 10% in accuracy and increasing our average error to over 10. Once again, in context, I think being within 10 points from 1-100 is not absolutely horrible. But it would appear this model type is not great for our dataset.

Neural Network

The final model we will look at is a simple neural network. Neural networks can learn complex patterns using layers of neurons which mathematically transform the data. The layers between the input and output are referred to as "hidden layers". A neural network can learn relationships between the features that other algorithms cannot easily discover. (Ronaghan, 2018) In our example, we are using the *scikit-learn* MLP Classifier which is a simple type of neural network called a multi-layer perceptron (MLP). This is represented as `MLPClassifier()`. Following suit in our previous examples, we fit this model to our standardized training data. This model was by far the worst performing, with an accuracy score of 26.1569, and an average error of 10.3313. Unfortunately, with an accuracy score this low, we don't gain much insight into our data here.

Next Steps

Where do we go from here? It is obvious that none of our models were very impressive. We were not able to achieve an accuracy score of higher than 65, or an average error of less than 8. Is this a data issue? Did we make a wrong decision along the way? Is popularity just

impossible to predict? I believe the answer to all of those questions is no, but how do we prove it?

I think our biggest issue is the year. Year alone accounted for over 50% of our models. It makes sense, because the popularity rating is a current/active variable, meaning that is the popularity of the song today. New songs definitely get more air play, especially in the Spotify playlists that we mentioned earlier. This is going to sway popularity. But how do we combat that? Should we remove the year altogether? Should we try to find a way to bin some of the years together? Should we adjust the popularity ratings to account for the year? We are on the right track, but I think finding the right answer to those questions will greatly improve the accuracy of our predictions.

Conclusion

Spotify is truly a beast in the audio streaming world. In 2018, it launched across 13 countries in the Middle East and North Africa, and in 2019, launched in India. Spotify remains comfortably in the lead over other streaming services, with 108 million subscribers to Apple Music's 60 million, as of June 2019. Spotify increased subscribers to 130 million in Q1 2020, while Apple Music is estimated to be somewhere near the 70 million mark as of February 2020. (Iqbal, 2020)

As an avid Spotify user myself, I thought it would be interesting to take a closer look at some of their data. I sought to determine if it was possible to predict the popularity of a song. While I would not say I failed in that respect, I don't know that I succeeded either. With the Random Forest Regressor, we were able to achieve an accuracy score of roughly 65%, and an average error of a little over 8. That meant on our test data set, we were able to predict the

popularity score within 8 points on average. Being on a scale from 1-100, this definitely could be worse. But it also could be a lot better, and I look forward to continuing my research.

References

Ay, Y.E. (2021). *Spotify Dataset 1921-2020, 160k+ Tracks* https://www.kaggle.com/yamaerenay/spotify-dataset-19212020-160k-tracks

Iqbal, M. (2020). *Spotify Usage and Revenue Statistics (2020)* https://www.businessofapps.com/data/spotify-statistics/

Passy, J. (2018). *How Spotify influences what songs become popular (or not)* https://www.marketwatch.com/story/how-spotify-influences-what-songs-become-popular-or-not-2018-06-18

Ronaghan, S. (2018). *Machine Learning: Trying to predict a numerical value* https://srnghn.medium.com/machine-learning-trying-to-predict-a-numerical-value-8aafb9ad4d36

Tiffany, K. (2018). *You can now play with Spotify's recommendation algorithm in your browser* https://www.theverge.com/tldr/2018/2/5/16974194/spotify-recommendation-algorithm-playlist-hack-nelson