PREDICTING SONG POPULARITY BASED ON AUDIO FEATURES

Anmol Anand

anmolanand@utexas.edu

Abstract

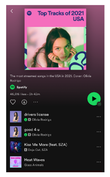
Artists are constantly putting out new songs, trying to develop the next big hit that will blow up and top the charts. What if we could help artists predict whether their newly-released songs will be a hit? We propose a solution using Spotify's API and the audio features to indicate a song's popularity index. We will show the accuracy of different models such as Linear Regression, Logistic Regression, and K-Nearest Neighbors.

Data Collection

To make the dataset, we decided that the best course of action would be to pull a series of tracks using the Spotify API and various public playlists. First, we added the Spotify "top hits" playlists for 2000-2021. Then, after concluding that the dataset wasn't diverse enough, we decided to add a series of playlists for different genres to pad out the dataset and make it more robust. Then, taking a look at the dataset's features, we noticed that popularity was an updating value, meaning that it was a metric that measured the current popularity. In addition to this, we also used the audio features provided by Spotify for each track.

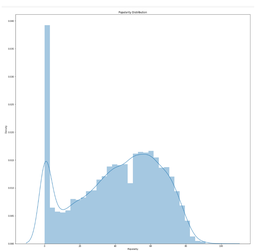
Acousticness	Danceability	Energy	Instrumentalness
Key	Liveness	Loudness	Mode
Speechiness	Tempo	Valence	Duration

To create the dataset, we developed a Python script that fetched tracks in a playlist and then, for each song in the playlist, used its ID to associate with the audio features.

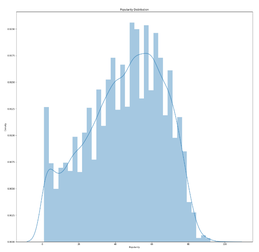


	track_uri	popularity	danceability	energy	speechiness	instrumentalness
0	5wANPM4fQCJwkGd4rN57mH	90	0.561	0.431	0.0578	0.000014

Data Distribution

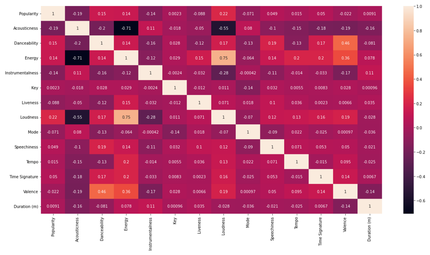


An analysis of 15,000 songs from 37 Spotify playlists shows that the popularity follows a normal distribution, except for tracks that don't have a popularity score. The Spotify Popularity Index is a 0-to-100 score that ranks how many times users play a track relative to other tracks on Spotify. The more popular a track, the higher its popularity score would be.



The data was processed to remove all tracks that don't have a popularity score assigned by Spotify.

Feature Selection



To reduce the features used in the model and only focus on the features that were important to the model, we made a correlation map. While no values were highly correlated, some had more correlations than others.

Out of all the present features, we focused on Loudness, Energy, Danceability, Valence, Speechiness, and Instrumentalness.

Results

Model	Training Accuracy	Testing Accuracy
Linear	13.26%	12.37%
Logistic (with Lasso)	90.05%	88.89%
KNN	73.47%	61.70%

Linear Regression:

After cleaning the dataset with feature selection, we tried to use Linear Regression. We first tried a basic linear regression model and got an accuracy of below 10%, and a root mean square error value of 22.60.

$$RMSE = \sqrt{\frac{\sum_{i=1}^{N} \|y(i) - \hat{y}(i)\|^2}{N}},$$

This value was a high number for RMSE, so we filtered out the songs with a popularity index of '0'. It was unclear precisely what Spotify meant by a popularity value of zero. We still got an accuracy of below 10% when we ran this, and the RMSE was 19.88. Through this, we concluded that linear regression wasn't going to be able to fit our data well, as no matter what features we chose to focus on, the results didn't change much.

Logistic Regression:

For Logistic Regression, we split the data into two sets: popular and unpopular. We labeled any values with a popularity index of 70 as popular and the others as unpopular.

Let y(x) = 0/1 [Binary classification - C0 popular songs, C1 unpopular songs]

$$\mu = E(y \mid \mathbf{x}) = P(C_1|\mathbf{x})$$

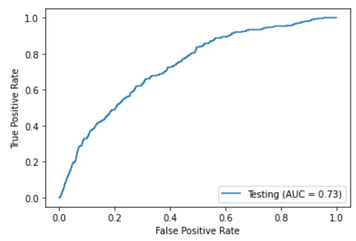
$$\text{Model: } \ln [\mu/(1-\mu)] = \beta_0 + \beta_1 x_1 + \ldots + \beta_k x_k = \beta.\,\mathbf{x}$$

$$\text{Equivalently: model } P(C_1|\mathbf{x}) = \mu = 1/(1+\exp(-\beta.\,\mathbf{x})) = \sigma(\beta.\,\mathbf{x})$$

We obtained the best results when using the uncleaned data set with lasso regularisation to prevent overfitting.

Results (cont.)

ROC/AUC curve:



Using a cutoff of 70, meaning <70 unpopular, >70 popular, we got an accuracy rate of
Training data Accuracy: 90.05%
Testing data Accuracy: 88.89%

The ROC_AUC_SCORE was:
ROC_AUC_SCORE on Training Data: 75.29%
ROC_AUC_SCORE on Testing Data: 75.91%

While these numbers looked good, our confidence in them was low. Upon examining the values of our predictions, we noticed our model predicted that every track was unpopular. First, we believed this was due to the dataset's creation, as most of the "hit" songs were on the top of the set while the "non-hit" songs were on the bottom of the dataset. Shuffling the dataset and rerunning it did not yield much better results. We then concluded that this was because the class was imbalanced. This result makes sense as most songs are more likely not to be a hit rather than be a hit, so our model decided that everything would not be a hit.

We then set class_weight to "balanced" in our Logistic Regression model to combat this and calculated the values of the class weights using: n_samples / (n_classes * np.bincount[y])

Our changes yielded the following results:
Training data Accuracy: 62.01%
Testing data Accuracy: 61.57%
ROC_AUC_SCORE on Training Data: 72.82%
ROC_AUC_SCORE on Testing Data: 73.23%

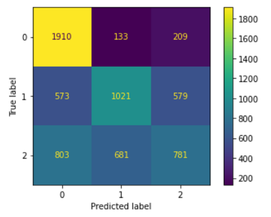
While at first glance, this seems worse than our previous model, when we appended the predicted values to the dataset, we were able to see that our model was predicting both 0s and 1s. When we analyzed the returned values, we got some fascinating results:

KNN:

Because of the distribution of our data, we decided to try using classification to be able to section the data off into certain boxes. Then, we could use k-Nearest Neighbours to try to predict probability. we created low, neutral, and high probability bins and got the best results from using the cleaned data. Then we could use KNN to predict the likelihood that a given point will fall into one of those three categories based on what the points around that data fall into.

Song Name	Actual Value	Predicted Value
Umbrella - Rihanna	0 (not a hit)	1 (hit)
Observation: This song was super popular when released and currently has 800M+ plays on Spotify. The model did this a lot, where it would predict that songs that were popular at one point were popular. Because Spotify bases the popularity value on current plays, it makes sense why our model would be off on this.		
All For Us - from the HBO Original Series Euphoria	1 (hit)	0 (not a hit)
Observation: This showcases another exciting aspect of our model. Our model predicted this wouldn't be a hit, strictly using audio features. However, due to social media (and the fact it is from a TV show), this song blew up and became super popular. This result shows that while our model can predict some values, it is still impossible to confidently call a song a hit because of aspects outside of musicality.		
High Hopes - Panic! At the Disco	1 (hit)	1 (hit)
Observation: An instance of our model successfully predicting a hit. This particular one was interesting because it is considered one of the most popular songs by this artist, and our model was able to discover that.		
Heaven - Beyonce	0 (not a hit)	0 (not a hit)
Observation: An instance of our model successfully predicting a nonhit. When looking at the streams from this album, while Beyonce is a famous artist, this is one of the lower stream counts on the album.		

Confusion Matrix: Using a n_neighbors value of 1, we wanted to be able to use different values, but it seemed like our model was suffering from overfitting in this case.



TP - True Negative 1403
FP - False Negative 21
FN - False Negative 81
TP - True Positive 1099
Accuracy Score: 72.71%
Misclassification Rate: 27.29%