Using ML to Predict Song Popularity

Samir Kaddoura, Simran Mallik, Niner Wilson

Motivation / Background

As students all interested in music, we wanted to investigate music data and see what interesting predictions we could make. After thinking about what problems we could investigate with this data, we landed on one that we were all curious to know more about: how can we predict the popularity of a song? As avid listeners of different genres, we were very curious to learn about what specific musical elements of a song makes a song popular. Are key signatures and tempo the main predictor of popularity? How important are the specific chords that are incorporated in the song? Or, do the energy and valence of the song matter more than other elements? These were questions that intrigued us all and we wanted to find a way to predict popularity based off of these musical elements of a song.

One important thing to note is that we were not looking at how popular the artist is or what genre the song is categorized as. We were purely interested in determining what musical elements are most important when predicting song popularity, and are not focused on other factors that don't influence how the song sounds like.

Data

We created our full dataset from a Spotify dataset found on Kaggle and joined this data with data pulled from the Spotify API.

The Kaggle dataset included all the songs that were on Spotify's Top 200 Weekly (Global) charts for 2020 and 2021. This dataset will be referred to as the "Spotify Top 200 Charts" dataset. The important features that were in this Spotify Top 200 Charts dataset included:

1. Highest Charting Position
2. Number of Times Charted
3. Song Name
4. Number of Streams (up until when the data was collected)
5. Artist
6. Song ID
7. Danceability

8. Energy
9. Loudness
10. Speechiness
11. Acousticness
12. Liveness
13. Tempo
14. Duration (ms)
15. Valence
16. Chord

Then, we joined this dataset with data from the Spotify API. The features from the Spotify API include:

1. danceability
2. energy
3. key
4. loudness
5. mode
6. speechiness
7. acousticness
8. instrumentalness
9. liveness
10. valence
11. tempo
12. id
13. uri
14. track_href
15. duration_ms
16. time_signature
17. title
18. artist

Initially, we wanted to use the Spotify Top 200 Charts and Spotify joined dataset as the training set, and find another Kaggle dataset for our testing set.

We found this new dataset called “Spotify Top Chart Songs 2022”, however, we realized that there was no stream metric, which was a component of our popularity score heuristic (made from the Spotify Top 200 Charts dataset). We decided to temporarily remove the stream metric from our score heuristic so we could use the Spotify Top Chart Songs 2022 dataset, however, we saw that without the streams metric, the popularity score heuristic was not very informative. We ended up reinstating the stream

metric in our popularity score heuristic, dropping this test dataset (the Spotify Top Chart Songs 2022 dataset), and splitting the original dataset (Spotify Top 200 Charts) into a training and testing dataset.

We also used a Kaggle dataset called “Charts” to extract the Spotify URL for each song.

In summary: The “Spotify Top 200 Charts” dataset was initially used as our training dataset, but then was used as the entire dataset that was split into the training and testing datasets. The Spotify Top Chart Songs 2022 dataset was initially used as our testing dataset, but was then dropped completely. The “Charts” dataset was used to retrieve the Spotify URL for each song so we could extract API information.

Methods

Preparing the Data

1. First, we investigated the features of the Spotify Top 200 Charts dataset (training), the Spotify Top Chart Songs 2022 dataset (testing), and the Charts dataset.
2. We then extracted the year from the Charts dataset and subsetted the region to only include music data from the U.S.
3. We checked which songs and artists from the Charts dataset were in the Spotify Top Chart Songs 2022 (testing) dataset, and included these songs in a new dataframe called bigdata_test_2.
4. Since we only needed information of the elements of each song and not daily or weekly information, we dropped duplicate data from bigdata_test_2 by dropping other rows with the same artist and track name (since there could be numerous rows for each song due to daily and weekly information). We named this dataframe as usedata.
5. We then extracted the Spotify API information for each song in usedata by using the Spotify URL. We put this information into a new dataframe called finaldata.
6. We then joined the Spotify Top Chart Songs 2022 dataset and the finaldata dataset into a new dataset called testingdata.
7. We repeated steps 3-6, but this time using the Spotify Top 200 Charts dataset (training) instead of the Spotify Top Chart Songs 2022 dataset (testing). The final dataset created was called trainingdata.
8. Finally, we saved the testingdata and trainingdata.

Note: Since “testingdata” didn’t have the stream metric, we ended up only using the dataset called “trainingdata” for our analysis as our full dataset, and then splitted this

into a training and testing dataset. The stream metric was important for our popularity score heuristic, so this is why we only used the "trainingdata" dataset.

EDA and Popularity Score Heuristic

1. We used the trainingdata.csv dataset and labeled this as "data".
2. We used histograms to look at the distribution of each feature to get a sense of what the data looks like.
3. We used scatterplots to look at the relationship between each feature and "Streams, "Number of Times Charted", and "Highest Charting Position" to determine if there were any trends or stark relationships.
4. We created the popularity score heuristic for each song using this formula: (Number of Times Charted / Highest Charting Position) + (Number of Streams / 1000000). In order to balance out the highest charting position and number of times charted we decided to divide the latter by the former. This way, songs that charted for a long time and achieved a high level of fame will have a higher score than songs that charted high but only for a short time, or songs that charted relatively low for a long time. To factor in streams, we just added a bonus of one point per million streams. We believe that this score provides a good heuristic to estimate a song's true popularity based on its musical elements, instead of using just one of the possible variables.

Model Creation and Hypertuning

We included valence, danceability, energy, key, loudness, mode, speechiness, acousticness, instrumentalness, liveness, tempo, duration_ms, time_signature in our models. Given that many of the most popular songs use a similar key signature (usually C major) and time signature (usually 4/4), we hypothesized that these two elements would be included in the best predictor model.

We tested linear regression models, GAMs, Lasso models, Ridge models, KNN models, and Random Forest models.

Linear Regression

We used forward stepwise selection to determine which features we should include in the linear models. After looking at the MSE for 12 linear models, each with a different combination of features, we found that the best linear model had an MSE of 51.98.

GAM

We used forward stepwise selection to determine which features we should include in the GAM models. After looking at the AIC for 12 linear models, each with a different combination of features, we found that the best GAM model had an MSE of 53.67.

Lasso
We included all the features in a LASSO model which outputted an MSE of 52.68.

Ridge
We included all the features in a Ridge model which outputted an MSE of 52.65.

KNN
We included all the features in a KNN model which outputted an MSE of 90.42.

Random Forest
We hypertuned our random forest in three ways. Because our dataset didn't contain too many independent variables, we decided that iterating through all possible models was a reasonable time commitment and would ensure that we selected the best features for our final model. To implement this iteration we use Forward Stepwise selection. We would iterated through the parameters adding them to our model one at a time. Then we would select the parameter that minimized MSE during the past iteration, remove that parameter from our list of variables, and iterate again. Eventually, we finished with a model that contained acousticness, loudness, time signature, mode, key, and instrumentalness.

Before we could implement that however, we ran a full model to hypertune the 'Max Depth' parameter. We found that a max depth of 2 generated the lowest testing error on our full model. This became our max depth parameter going forward.
Once we iterated through the variables a few times and had a model with more than 3 parameters, we decided to hypertune the max features parameter in sklearn, which affects how many features are looked at before splitting each node in the tree. We ended up with a max features value of n/3 with n being the number of parameters in our model.

Results

Our lowest MSE we encountered was 36.8, but this was with a two variable model. We opted for a 6 variable model with an MSE of 37.1 for our final model.

We think the model performed well based on the MSE's and also our general knowledge - for example, "Blinding Lights" ranked very high in our model. This song

also was very popular and broke multiple records. From our general knowledge, the song rankings made sense.

Challenges and Further Work

Though there are many Spotify datasets, there were few that had streaming data, which we needed to create a good popularity score heuristic. For the future, we could pull data on our own to get the number of streams so that we could have more data to work with.

We are confident that the algorithm can be used for predicting popularity in 2020-2021, as this is what our model was trained on. We also believe the model can be extrapolated to other years that are close to 2020 and 2021, such as 2000 and onwards. However, as time progresses, people's taste in music evolves, and so this model would need to be trained on fresher data. Adding a time variable would be good so that one can see how specific songs would rank in specific time periods.