Samuel Pederson

Final Report:

Spotify Music Data Analysis

Introduction / Problem Statement

Music is something that we all enjoy in our daily lives. Whether we listen on the road or when getting ready in the morning, music has become a part of our daily lives. This is even more evident when you consider that the music industry made over 28 billion dollars in 2021 according to Statista. This is all to say that having a song be even slightly more famous can be a great boon to everyone involved. So then the question becomes how can we increase the popularity of music outside of the old, conventional ways of doing so like hiring a bigger-name artist? This was the question I was asking myself when I first found the Spotify song popularity dataset on Kaggle. And the results of my research were a bit mixed so to speak, but interesting nonetheless, so without further ado let's get into it.

Data Wrangling

The dataset that I decided to use for this was pulled from Kaggle and came mostly cleaned with upwards of 18,000 songs represented in the dataset (specifically 18,835 songs). The data had already been mostly cleaned as there were no null entries and it was already organized in such a way that doing analysis, and while there may have been some outliers none of them were too extreme. The biggest problem that I ran into during the data wrangling process is that

there were a lot of duplicates of the data. When I got rid of all duplicate songs, there were only 13,068 songs left in my dataset. This means that I had nearly 6,000 duplicates in my dataset when I started. I also tried to include some additional data from other datasets I found relating to the top 50 songs of each year, but as the songs that I had in the initial dataset were not pulled specifically from those years I decided to drop this exportation pretty early on. Additionally, there were some features that I decided were not doing all that much, and I eventually dropped them, but that was not until I moved into my exploratory data analysis. If I could do one thing differently regarding my data wrangling, I think I would have just gotten my data directly from Spotify's API. I'm not sure if this would have had much effect on the results at the end, but I could have gotten more information on each song (say for example the artist who made it), and I think that would have helped the accuracy of the model in the end.

Exploratory Data Analysis

To talk about the features that I decided to drop in my dataset, I feel like it is necessary to first give a brief description of each of them before I explain why I decided to keep some of them and get rid of others. The features were as follows:

- Song_popularity: A measure of how popular the song was on a scale from 1-100 and the target feature I was trying to predict during this study.
- Song_duration_ms: A measure of how long the song was in milliseconds.
- Acousticness: A measure of how acoustic a song is measured on a scale from 0-1 with a score of 1 indicating the song is likely acoustic.

- Danceability: A measure of how good a song is for dancing based on a variety of other metrics measured on a scale from 0-1 where a score of 1 means the song is likely good for dancing.
- Energy: A measure of how energetic a song is (a fast-paced or loud song would be considered energetic) measured on a scale from 0-1 where a score of 1 would mean the song is very energetic.
- Instrumentals: A measure of how instrumental (non-vocal) a song is measured on a scale from 0-1 where a score of 1 would mean there are little to no vocals on the track.
- Key: A measure of the key signature of the song measured on a scale from 0-11 using the Pitch Class Integer Notation (a good explanation is available here).
- Liveness: A measure of the likelihood that a song was recorded live measured on a scale from 0-1 where a score of 1 would mean it's likely the song was recorded in front of a live audience.
- Loudness: A measure of how loud a song is before being adjusted by Spotify, measured by LUFS (an article here can describe how those are measured). Scores range from -38 to 1, but all are normalized by Spotify to ~-14 when the song is played.
- Audio_mode: A measure of the modality of the song on a binary scale where 0 means the song is in a minor mode and 1 means the song is in a major mode.
- Speeches: A measure of how many spoken words are in a track, measured on a scale from 0-1 where a score of 1 would mean the whole track is spoken words.
- Tempo: A measure of the tempo of a song, measured in BPM (beats per minute).
- Time_signature: A measure of the time signature of a piece, measured on a scale from 0-7 where all are in a /4-time signature.

- Audio_valience: A measure of how positive sounding the given track is, measured on a scale from 0-1 where a score of 1 would indicate that the song is happy sounding.

I'll look over the top track category I tried to makin as it did not accomplish what I had wanted, leaving me with these categories. That said, not all of these were very useful in describing the popularity of the song. Based on the initial look, there were very few things that correlated well with the song's popularity and the few things that did correlate were with very weak correlations as seen in the heatmap below.

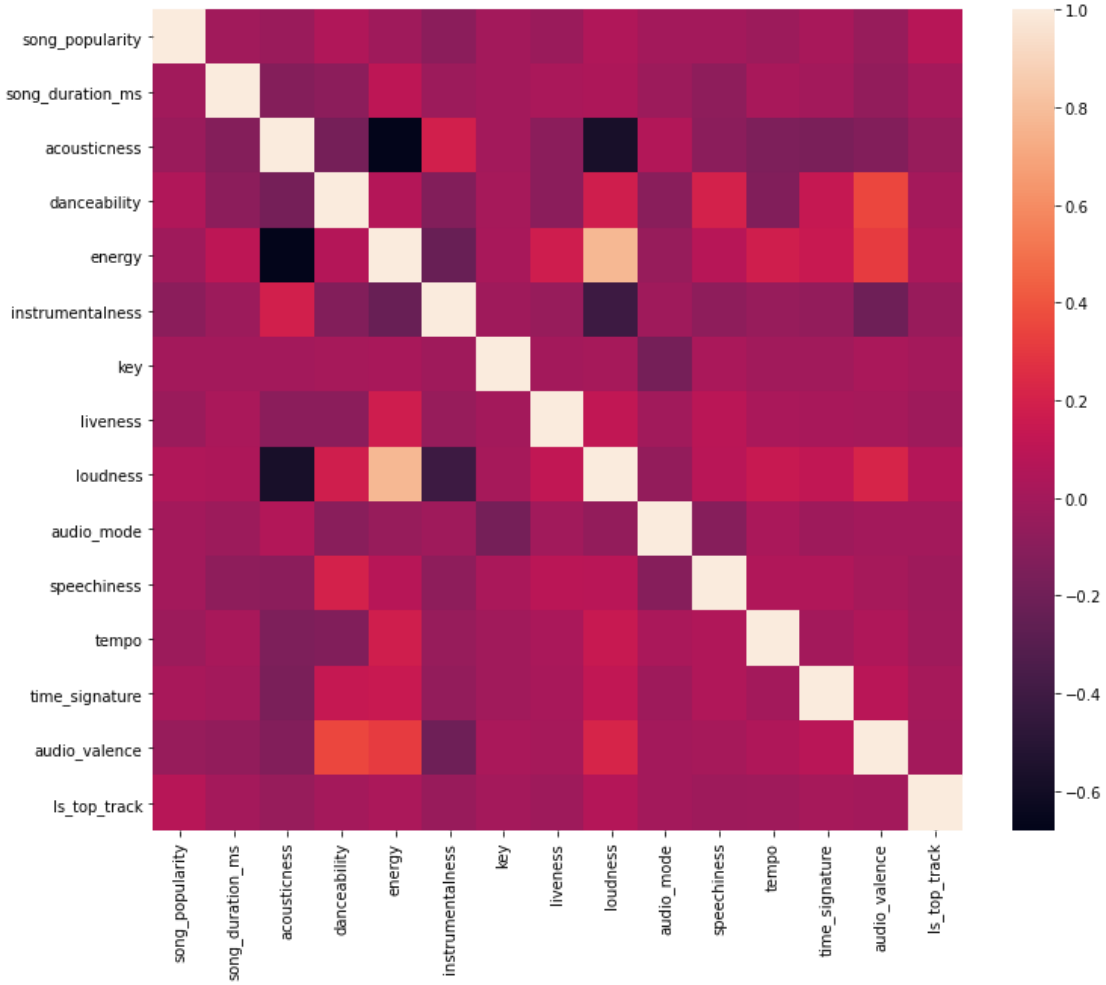

Figure 1: Correlation Heat Map of Song Features

As you can see, the song's popularity loosely correlates with all the features. That does not mean that there is no possible correlation, but it was at this point that I began to get nervous that I wouldn't be able to model popularity well with the features I had. I'll talk a bit more on that later, but for now I'll go over the features that I decided were too uncorrelated to make a difference when it came to modeling. The features that I deemed were not useful for modeling were audio_mode and time_signature (of course in addition to the is_top_track category that I talked about previously). The reason for me to pick these two columns out in perticular was due to them seemingly having no correlation at all with the song's popularity. This is, in my opinion, due to what these categories represent. Audio mode shows whether a song is in a major or minor mode, which is just a little above a 60-30 split (~36.7% minor mode and ~63.3% major mode) as seen below in figure 2-a. Additionally it had the lowest correlation score when compared to all the other features, leading me to drop it to help reduce dimensionality. Time signature was much the same, albeit with a higher correlation than some of the categories that I did end up keeping. The main reason I decided to get rid of this feature was because most of the values were 4 (indicating a 4/4 time signature). This shouldn't be of much surprise as it is by far the most popular time signature when writing music (~93.4% of the songs in the dataset were this time signature), but I felt like the extra dimensionality that would have come along from using one hot encoding would have caused further problems while not leading to any new knowledge. Below in figure 2-b you can see this distribution for yourself.

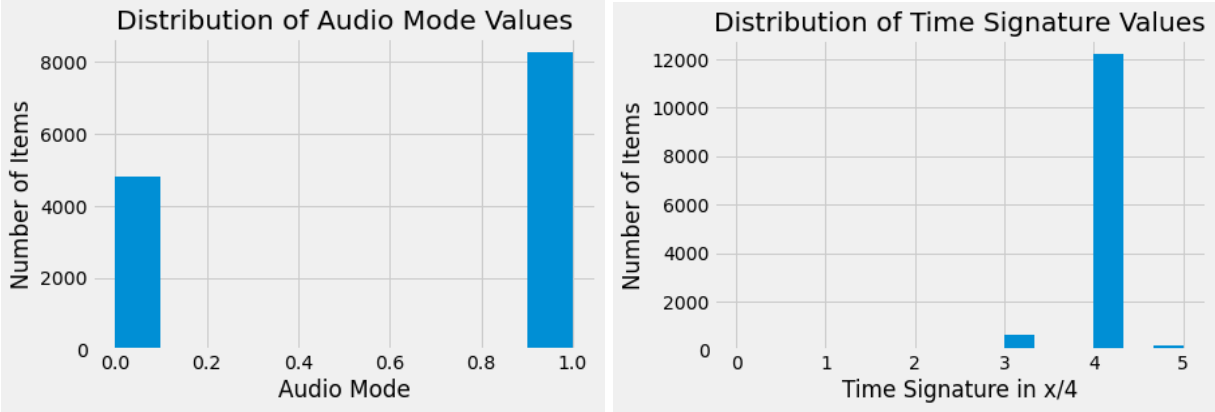

Figure 2: Distribution of Values for Audio Mode (2-a) and Time Signature (2-b)

Pre-Processing and Training

I'll attempt to keep this section short as I only made a couple of transformations as most of the features were not made in such a way that would be beneficial for merging. To solve this, I could have done some additional feature engineering with something like the featuretools library, but both of my own discretion and that of my mentor I decided that this problem would not have been overly helped by this decision. Instead I used two main transformations to the data at this stage. I first got dummy features for the categorical columns (at this point, I had reduced it down to just the key as a categorical column), doing this so that the modeling would run properly later on. The second transformation that I did was scaling all the non-categorical columns using StandardScaler from scikit-learn, once again doing so to ensure later modeling would go smoothly. One thing I just have to say about this dataset is just how clean it was in it's initial state, as there were relatively few procedures that I had to do to get to the modeling of my problem.

Model Selection

For the model selection I made the decision to try out 5 different models as opposed to the 3 that was required for the assignment. The reasoning for this decision was because I knew going into the modeling that I may have a hard time geting a model with high accuracy just based off the known correlation (or rather lack therof). This also led me to taking the additional step of using a Grid Search cross validation for all of the models in order to see how good of a model I could make with the given features. Below in figure 3, you can see the mean absolute errors (MAE), as that was the first metric I was using to make a decision on the efficacy of the models.

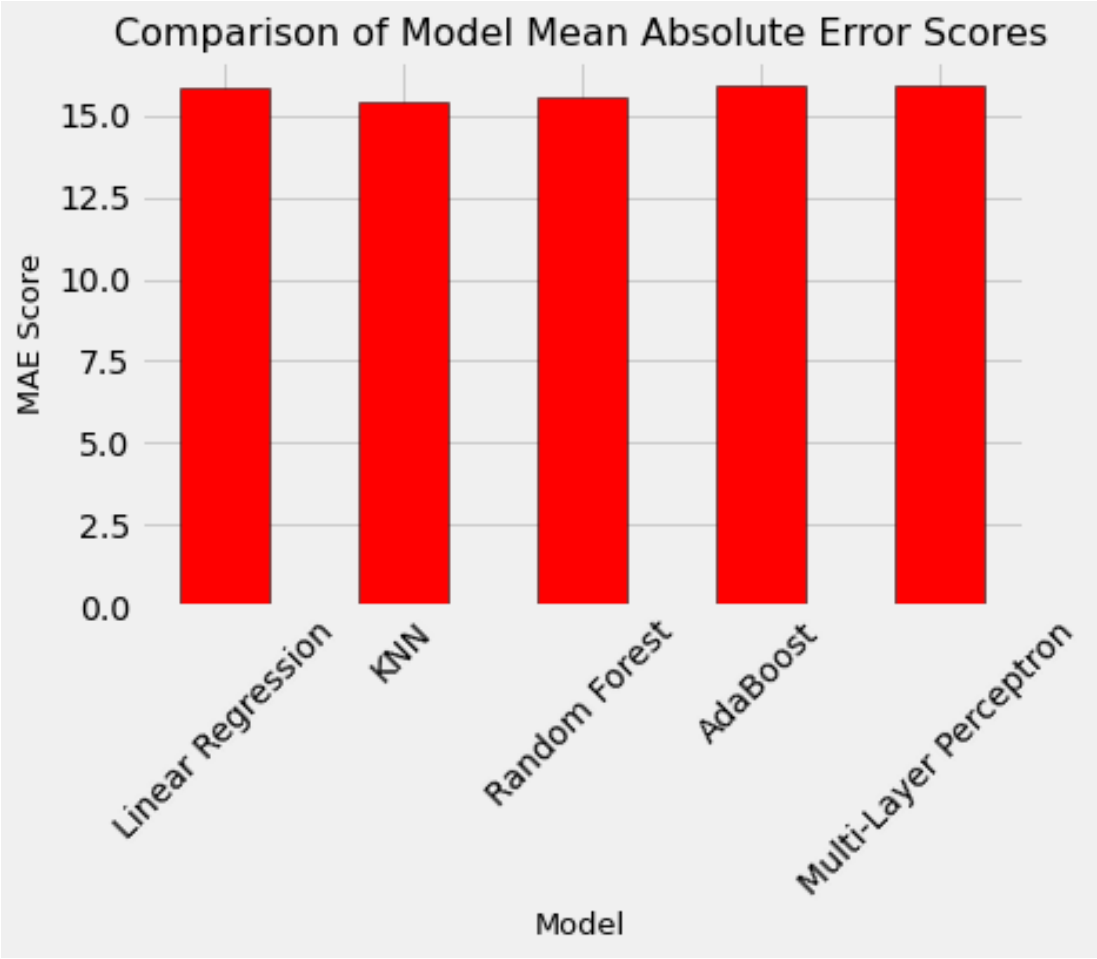

Figure 3: Mean Absolute Error (MAE) Results for All 5 Tested Models

As you can see, all the models had roughly the same mean absolute errors, all being in the 15-16 range (remember on a scale from 1-100). Because of this, I decided that to make an accurate judgement on which model performed the best I would need to also consider another metric, and as it is a problem relating to regression I decided to go with an R^2 test, the results of which are in figure 4.

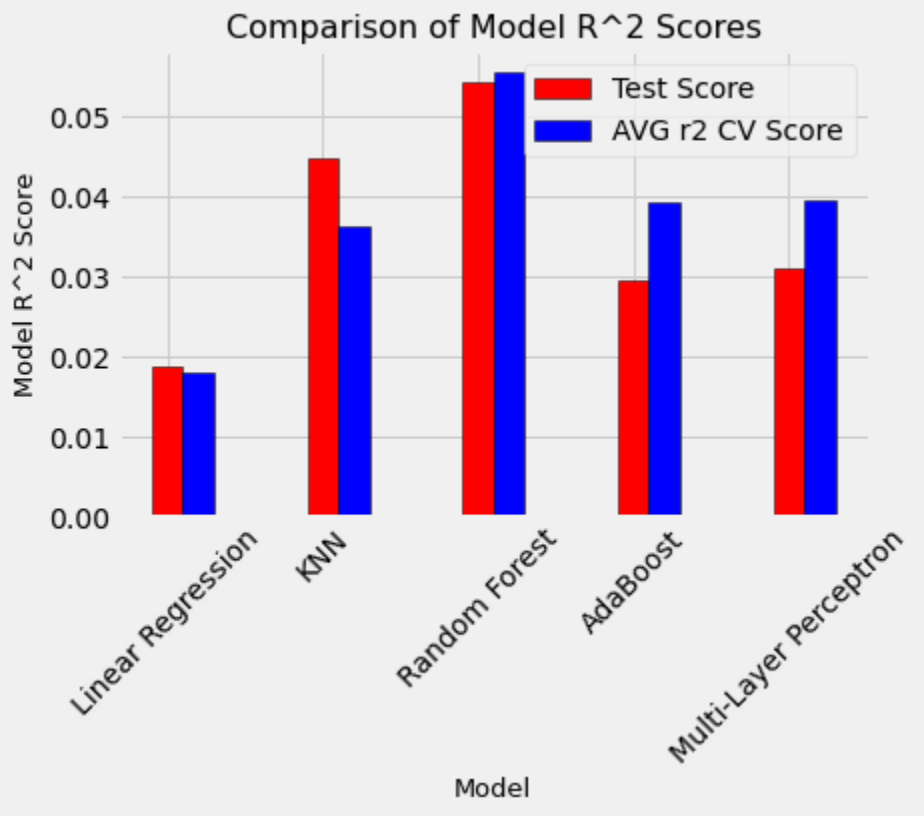

Figure 4: Test and Average Cross Validation R-Squared Scores for All Models

When I got these results, I had an initially panicked response. For a quick explanation of how to interperate R^2 score, it (generally) is on a scale from 0-1 and acts similar to a percentage. In this case, the Random Forest model having a score of 0.05 would mean that the model can explain about 5% of the variance of the target feature (song_popularity). In other words, popularity is more a factor of outside, non-tested features than the ones that were used. From here I tried another Grid Search on the Random Forest model allowing for more estimators

among trying out some different parameters in the grid. That said the additional time it took for the new models to run and the inconsistency in the quality of these models led me to drop further hyperparameter tuning as I saw not much benefit from this. I ended up going for the Random Forest model in the end still as the best model as it had roughly equivalent MAE scores and had by far the best R^2 scores.

Key Takeaways

From this project I learned quite a few things, the biggest being that it is very hard to properly model the popularity of a song based off the musical features of the song. However, that isn't to say that there was no discovery of any kind made during the process. The foremost discovery I made was that the best song element to look at for discerning the songs popularity is the key it is in. As seen below in figure 5-a, all the most important features were related to the key the song was in. This led me to making testing the difference key would have on the outcome of the popularity test assuming all other elements were the same besides key, the results of which are in figure 5-b.

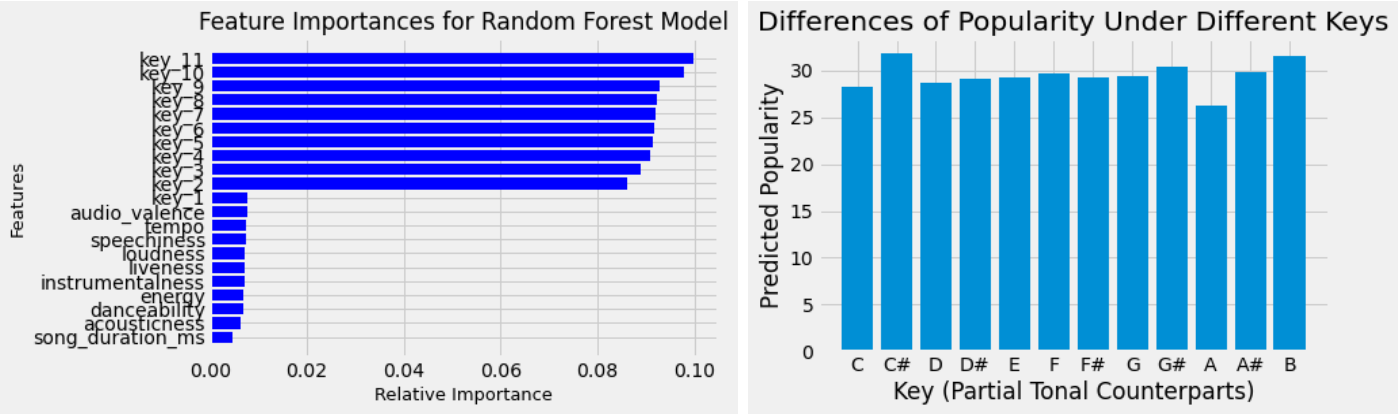

Figure 5: Feature Importances in Main Model (5-a)

and Popularity Difference under Every Key (5-b)

I want to preface any suggestions I am about to make in terms of key signature with a disclaimer that all of these values are within the margin of error. That said, we can see from the best performing model that the most important feature (or rather features) is the key of the song, making up a majority of the projected popularity of the song. As stated earlier, I then used this information to test the different values of popularity by just changing the key. From that, we can determine that you are slightly more likely to have a higher popularity song if you have something in the key of C# (or B ♭ as they are the same) versus something in the key of A. In fact, the expected difference in popularity outcomes between these two groups would be just above 5% (assuning the 1-100 scale was as percent). From this, we can determine that key has the most impact on the outcome of a songs popularity, at least when only considering the inherent aspects of the music. So the two key takeaways I want to leave you with are that you cannot accurately predict the popularity of a song based off the elements of the song and if you try to maximize the potential for popularity by choosing specific elements, choosing the right key signature (mainly C# and B) is the best approach.

Future Research

Reguardless of what I have said, the results of my tests were of course less than ideal, after all I could only describe 5% of the variability in popularity with my best performing model. That said, I don't think this means you cannot figure out what leads to a popular song, instead it just means that the aspects of the song I was looking at were not the best for modeling. One thing I wish I could have had was some kind of artist ID code for the problem, as I personally think much of what makes a song popular is the person who is singing it. Another factor that I think could have helped would have been something to figure out how much the song was being

promoted either on radio or on social media apps like Instagram and TicTok (albeit there wouldn't have been any easy way to get this). I think both of those would have had much more impact on the popularity of any given song.

If I were to do anything different in my process, I think it would have to be not getting rid of the time signature column. While it was mostly the same value, it still could have possibly led to slightly better accuracy, but I showed things as I did them and I still don't totally regret my decision. Finally, I think I should have maybe tried more or better hyperparameter tuning for some of the models that I tested out, as for some of them such as the neural net that I tried out I just kind of threw in a variety of things into the grid for the grid search and I maybe could have been a bit more thorough. That being said, I am confident in the results that song popularity is not overly effected by the aspects that make up the song.