Spotify Song Popoularity Prediction Project

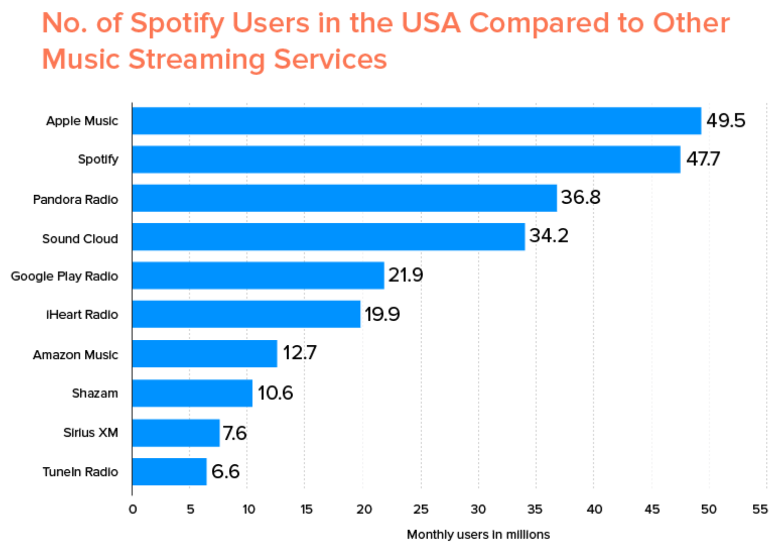
Author: Dan Kang
Course: Springboard
Cohort: August 2020
Assignment: Final Capstone

Summary

In this Capstone Project, we attempt a solution at the Spotify Song Popularity Problem using a dataset from Kaggle that contains over 170,000 songs from Spotify spanning the years 1921-Present. The data was collected via the Spotify Web API and contains a number of numeric and categorical features for each song, which can be used to predict the song's Popularity (scored on a scale of 0 to 100). We attempted 3 different models before settling on a Random Forest Regressor Model with leading performance of 88.5% on our test set (91.5% on train set). Special thanks to Devin Cavagnaro for his mentorship on this project.

Introduction

Spotify is one of the largest streaming services in the world, along with Apple Music, Amazon Music, Pandora, and Tidal. It's home to 345 million Monthly Active Users and the company has a market capitalization of $58Bn, making it one of the leading music and technology companies in the world. One of Spotify's competitive moats is its Discovery feature, which makes recommendations to users based off of their tastes/preferences. As such, predicting which songs a user may like is at the core of Spotify's long-term strategy, and key to fending off competitive pressure.



Source: https://appinventiv.com/blog/spotify-statistics-facts/

Clearly, Spotify and the artists on the platform both have a vested interest in determining whether or not a song will have a high popularity score (i.e. appeal to Spotify's user base more broadly). More popular songs generate more streams, driving more revenues to both Spotify and the artist: artists get paid $.0003 per stream, and higher engagement on the platform leads to higher user retention for Spotify. Additionally, labels have a vested interest in investing in songs they think will be more popular (via promotional activity) in order to drive more efficient margins/bottom line results. We outline our methodologies below.

Data Wrangling and Features:

The raw dataset began with 170,653 songs and 19 features. We outline the features below, but a more detailed summary of each feature can be found here:

Primary:
- id (Id of the track/song generated by Spotify) – We drop this feature as it doesn't help our model

Numerical:
- acousticness (Ranges from 0 to 1): Measure of electric amplification (0 being lots of electric amplification and 1 being none)
- danceability (Ranges from 0 to 1): How danceable a song is
- energy (Ranges from 0 to 1): Mellow vs. dynamic
- duration_ms (Integer typically ranging from 200k to 300k): Length of song
- instrumentalness (Ranges from 0 to 1): Measure of vocals in a song (0 having more vocals 1 having more instruments)
- valence (Ranges from 0 to 1): Measure of mood, with 0 being sad and 1 being happy.
- popularity (Ranges from 0 to 100): The metric we are trying to predict
- tempo (Float typically ranging from 50 to 150): How fast a song is
- liveness (Ranges from 0 to 1): Probability that a song was recorded at a live concert
- loudness (Float typically ranging from -60 to 0): Self explanatory.
- speechiness (Ranges from 0 to 1): Measure of spoken words in a track
- year (Ranges from 1921 to 2020)

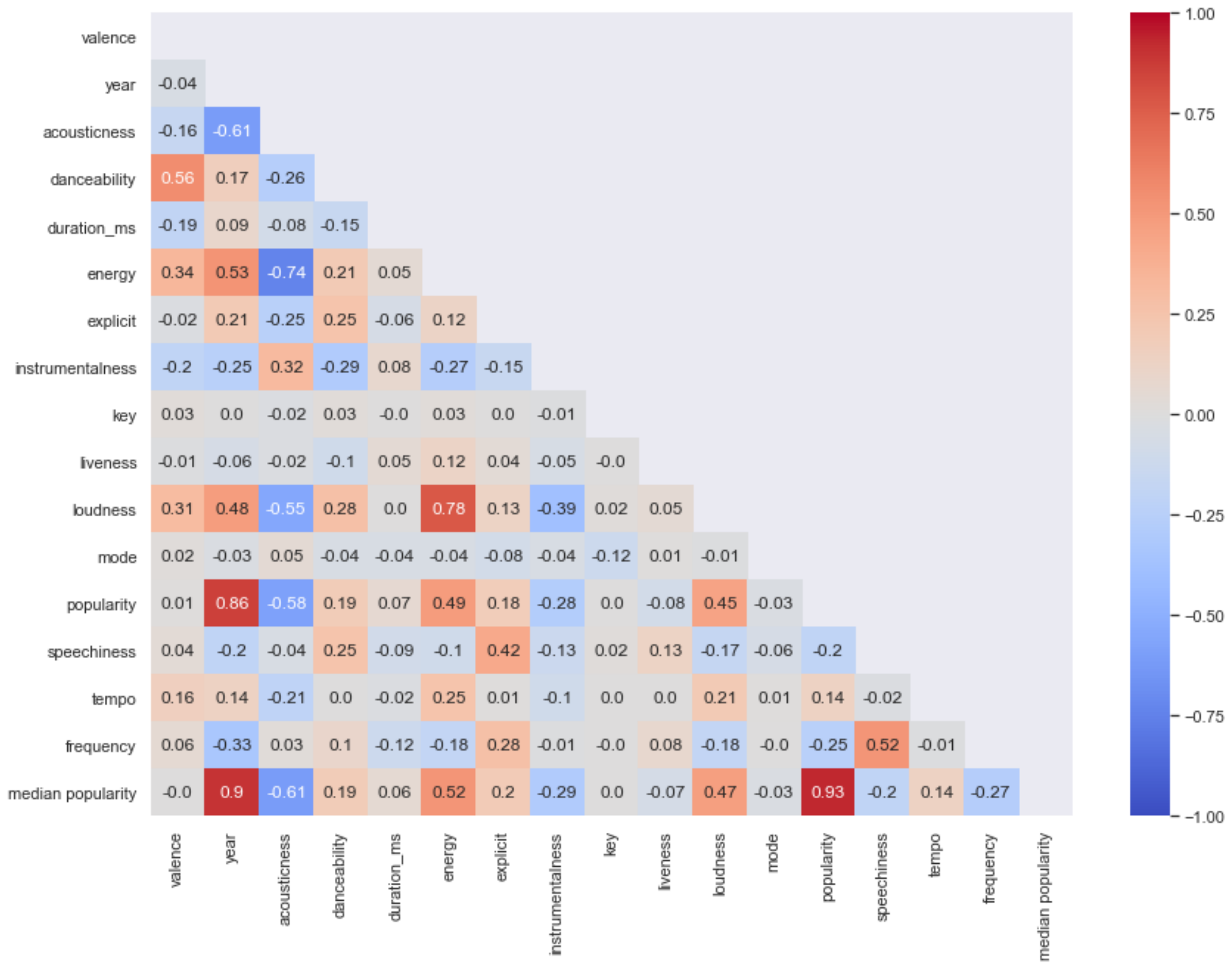
Dummy:
- mode (0 = Minor, 1 = Major)
- explicit (0 = No explicit content, 1 = Explicit content)

Categorical:
- key (All keys on octave encoded as values ranging from 0 to 11, starting on C as 0, C# as 1 and so on…)
- artists (List of artists mentioned): We eventually replace this feature with the median popularity of the artist; it is an important feature used in our model, but needs to be represented numerically.
- release_date (Date of release mostly in yyyy-mm-dd format, however precision of date may vary) – We drop this feature as it doesn't help determine popularity
- name (Name of the song)

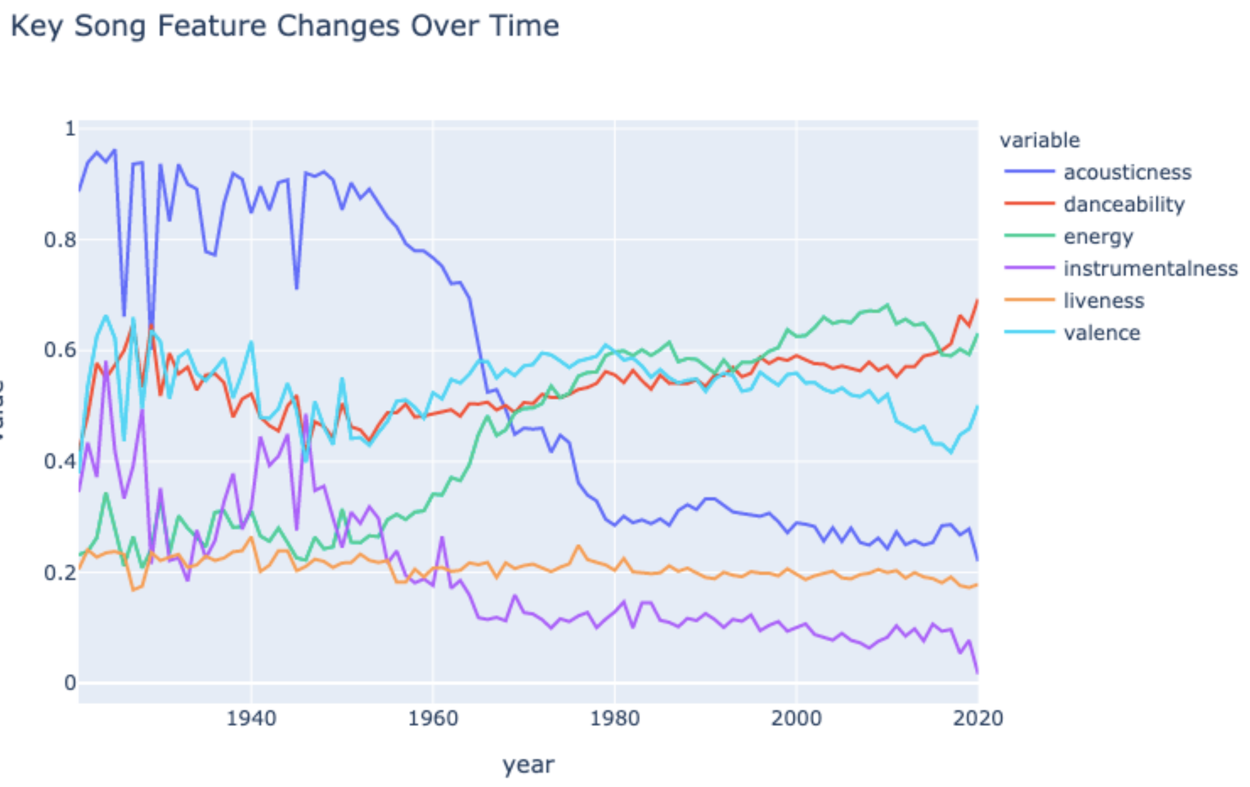
After dropping tracks with duplicate names, we end up with 133,638 songs.

Exploratory Data Analysis

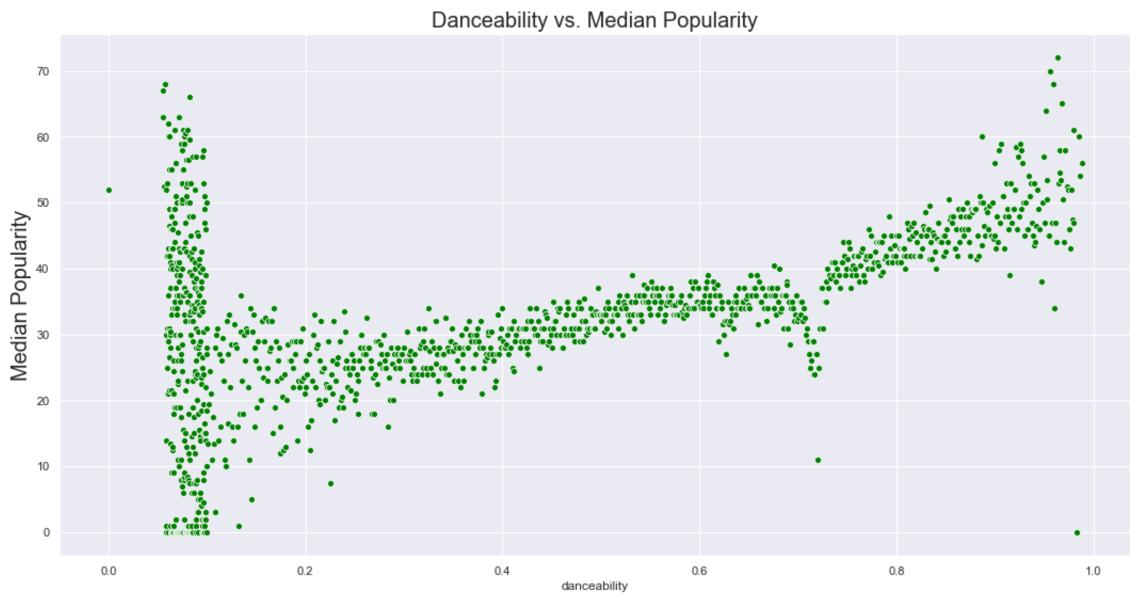
After exploring the top songs by feature (most popular, danceable, live, etc,), we produced a heatmap to see the correlation between the features.



We find that year, acousticness, and energy are most correlated with popularity. Intuitively, it makes sense that more recent songs are more popular on the platform. Below we outline how several key features have changed over time.



Finally, we proceeded by grouping by each feature of interest and then plotting each feature vs. the median popularity measure of that group. See below, where we find that danceability has a generally positive correlation with median popularity:



Based off our EDA, we decided to isolate the following key features:

key_features = ['popularity', 'acousticness', 'danceability', 'energy', 'instrumentalness', 'liveness', 'valence', 'year', ‘artist’]

Pre-Processing / Training

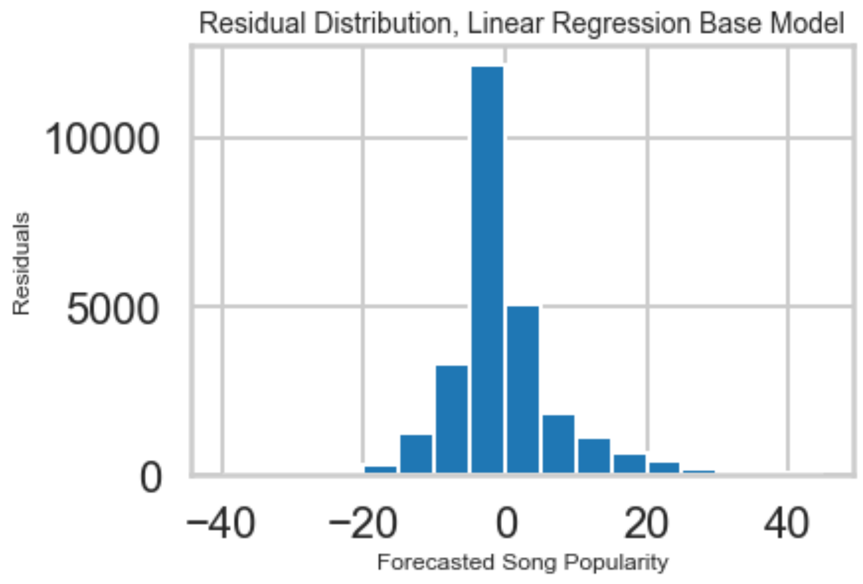
We replaced artist with median popularity so we have a numerical way of predicting popularity – we also make sure to adjust to account for artists who may have been one-hit wonders, so the artist must have released at least 2 songs in order to qualify for the data we use in preprocessing. We used an 80/20 train test split and ran some baseline linear regression models to get a general sense of how well linear regression would perform without any hyperparameter tuning. We scaled with StandardScaler() and then used GridSearchCV() for hyperparameter tuning. Not surprisingly, our preprocessing also highlighted Median Popularity (i.e Artist) as a strong factor in determining popularity.

Modeling

We ran 3 different modeling scenarios: standard linear regression, a random forest, and KNN, and assessed performance via Correlation Coefficient, Mean Absolute Error, and Root Mean Squared Error.

Based on the work conducted during pre processing, we expected a standard linear regression model could/should serve just fine given the reduced feature set we decided to use, so further work could be done to assess the impact of including all other features; however, given the other features' relatively low correlation with popularity (most have R^2 < 0.3), for simplicity we excluded those features.

```
Base Linear Regression Train Score:       0.8696329040638551
Base Linear Regression Test Score:  0.8732499569027268
--------------------------------------------------------------------------
------
Ridge Regression Train Score:             0.869632903836639
Ridge Regression Test Score:        0.8732500098793184
--------------------------------------------------------------------------
------
Lasso Regression Train Score:             0.8696329040598028
Lasso Regression Test Score:         0.8732499674113304
```

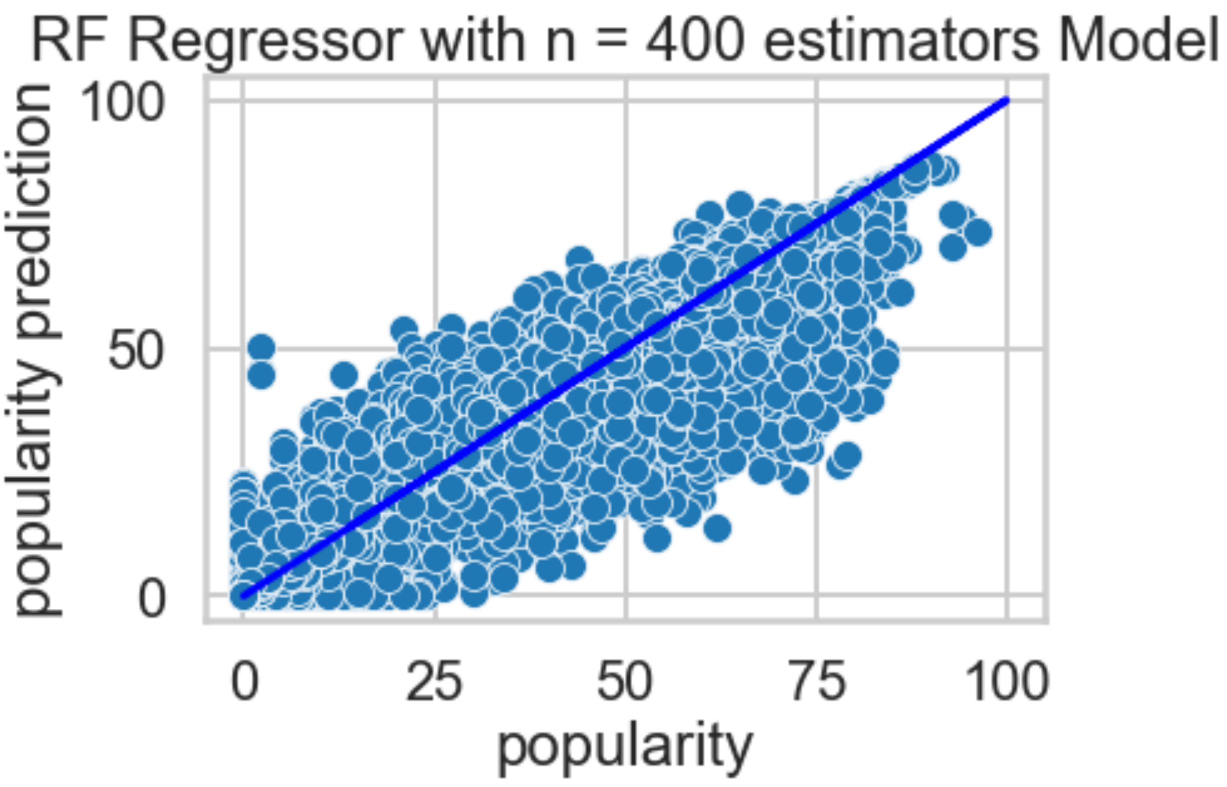


We then used a RandomForestRegressor() baseline model—for context, this model fits a number of classifying decision trees on various sub-samples of the dataset, and uses averaging to improved predictive accuracy and control overfitting.

```
Base Random Forest Train Score:       0.9829180789475191
Base Random Forest Test Score:  0.8811599967332122
Base Random Forest MAE:               5.0065475136352955
Base Random Forest RMSE:              7.700652819235559
```

Our baseline random forest accuracy score was extremely high at .9829 on our training set, with a test score at .8812 suggesting overfitting on the baseline model. We thus adjusted for max_depth and found m = 14 to be the optimal value to prevent overfitting, resulting in a much lower training score of 0.9154 but a better test score of 0.8851 (with the two scores converging meaningfully). We used this tuned model as it outperformed our Ridge and Lasso regressions and did not overfit the way our baseline model did.

```
n=400 Random Forest Train Score:       0.9154081545055371
n=400 Random Forest Test Score:  0.8851086097219257
n=400 Random Forest MAE:               4.879277125416826
n=400 Random Forest RMSE:              7.5716400316387364
```



Business Recommendation

An actionable business recommendation can be taken from three different perspectives with this exercise:

1) Spotify: From Spotify's perspective, they are in the best position to make a highly accurate and useful ML model to predict popularity given the billions of streams they see every day. Again, better Discovery improves the value of the product, driving higher user retention and long-term pricing power. Our recommendation would be to use this model to predict popularity of new songs on the platform so that they can power their Discover offering.
2) Artist: From an artist's perspective, a strong popularity prediction model could mean that an artist could "backwards engineer " a popular song. Clearly the notion is a bit silly, but in actuality an artist who is releasing an album with multiple songs could predict which song on the album is going to be most popular and therefore invest more time into promoting that song (via social media, paid/organic digital campaigns, etc.).
3) Label: The 3 major music labels control ~70% of the world's music supply, and that dynamic looks unlikely to change in the near term. From their perspective, a model like this can help them select artists that have the highest probability of producing continuous hits over time, and perhaps reduce investments in artists that are less likely to produce popular streams.

Additional Work

From our perspective, this project will be ongoing and perhaps the biggest change will be on feature engineering. Rather than apply StandardScaler(), we may want to try other scaling techniques. Additionally, further work can be done so that our model not only predict popularity, but actually makes song or artist recommendations based on what’s trending. Lastly, an interesting exercise would be to incorporate decade into the input of the model, so that our model automatically adjusts which features are most important based on time era.

Works Cited

1) https://www.kaggle.com/yamaerenay/spotify-dataset-19212020-160k-tracks
2) https://appinventiv.com/blog/spotify-statistics-facts/
3) http://scikit-learn.org/stable/auto_examples/ensemble/plot_ensemble_oob.html#sphx-glr-auto-examples-ensemble-plot-ensemble-oob-py