Predicting Song Popularity Using Spotify Data

Sarah Prakriti Peters
Brown University - Data Science Initiative
Github Link

INTRODUCTION

Spotify is a global audio streaming and media services provider, founded by Daniel Ek and Martin Lorentzon. As of June 2022, they have 433 million active monthly subscribers. According to an article published in Variety, 100,000 songs are being uploaded on a daily basis to digital service providers. According to Business Insider, Spotify is the top music streaming app in the market. With such a large consumer base, Spotify frequently releases the top 50/100 popular song playlists - both globally and regionally. Using Spotify data, the prediction of a song's popularity is particularly important, as those scores can be used in several use cases. For instance, a popularity score can be calculated for new releases from both upcoming and established artists. A higher score means higher chances of being featured on the Release Radar and Discover Weekly playlists. Further, music labels can predict future music trends using these scores.

The aim of this project is to develop a model that predicts the popularity metric from a scale of 0 to 100 (0 being least popular to 100 being most popular) for unreleased or new music using regression. For this project, data has been collected from seven genres of music - Alternative, Blues, Hip-Hop, Indie Alternative, Metal, Popular and Rock music. Most features in the dataset focus on the quality of music through audio features which are defined below:

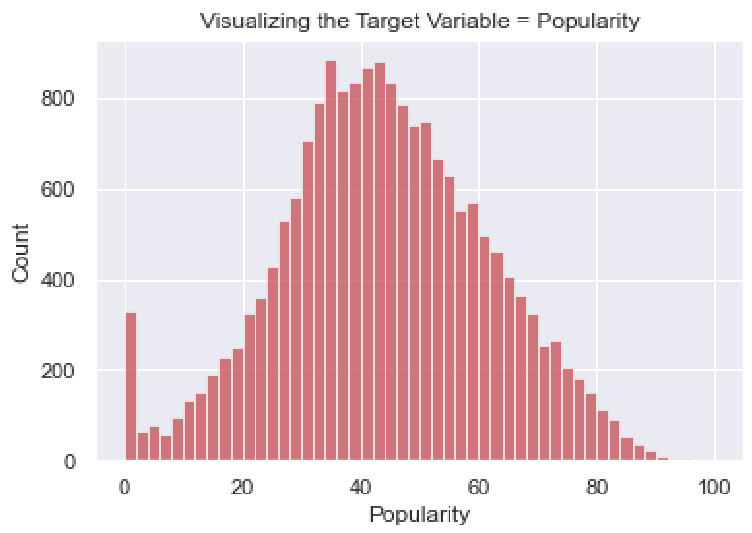
Danceability refers to how suitable a track is for dancing based on a combination of musical elements including tempo, rhythm stability, and overall regularity. A value of 0.0 is least danceable and 1.0 is most danceable. **Energy** is a measure from 0.0 to 1.0 and represents a measure of intensity and activity. Energetic tracks feel fast and loud. **Key** is the estimated overall key of the track. The mapping starts at 0 = C, 1 = C♯/D♭, 2 = D, and so on. **Loudness** refers to the overall loudness of a track in decibels (dB). **Mode** indicates the modality (major or minor) of a track. Major is represented by 1 and minor is 0. **Speechiness** detects the presence of spoken words in a track. **Acousticness** is a confidence measure where 1.0 represents high confidence the track is acoustic **Instrumentalness** predicts whether a track contains no vocals. The closer the instrumentalness: value is to 1.0, the greater likelihood the track contains no vocal content. **Liveness** detects the presence of an audience in the recording. A value above 0.8 provides strong likelihood that the track is live. **Valence** is 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). **Tempo** is the overall estimated tempo of a track in beats per minute (BPM). **Time**

Signature refers to an estimated overall time signature of a track. The time signature (meter) is a notational convention to specify how many beats are in each bar (or measure). **Duration milliseconds** is the duration of the track in milliseconds. The remaining features in the dataset are identification features such as Artist and Track names, URI, Track URL, Genres and Playlists. The final dataset has 26,752 records and 22 features, which was downloaded from Kaggle (with no missing values), but originally extracted from the Spotify API.

An interesting take on this project was documented by Philip Peker on Medium [5], where he discussed predicting popularity through techniques such as Linear regression, Decision Trees and Random Forest. The common evaluation metric used was Root Mean Squared Error (RMSE). Of these three, Random Forest performed the best. A particularly interesting discovery was that the dependent variable (Popularity) did not have a strong correlation with other independent variables, which is similar to the correlation results obtained from this dataset.

EXPLORATORY DATA ANALYSIS

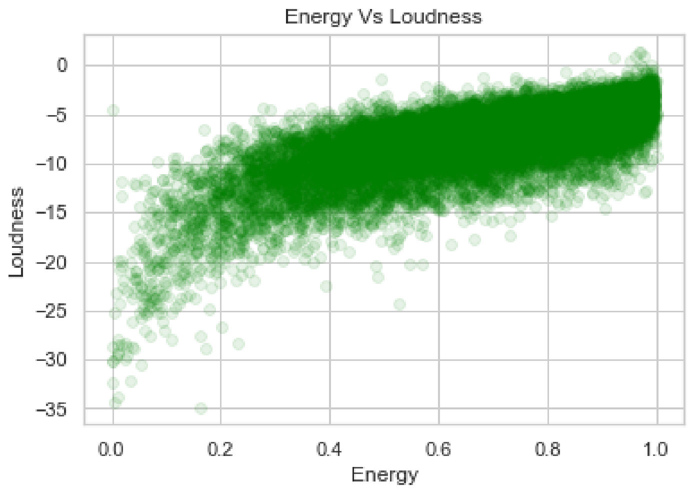
The target variable here is Popularity. Popularity scores are bound between 0 and 100, with 0 being least popular and 100 being most popular. As seen in the graph below, the distribution of popularity scores looks like a normal distribution, with a spike at 0. Most popularity scores lie between 40 and 60.



Popularity scores lie between 0 to 100. This histogram showcases the distribution of scores across the entire dataset

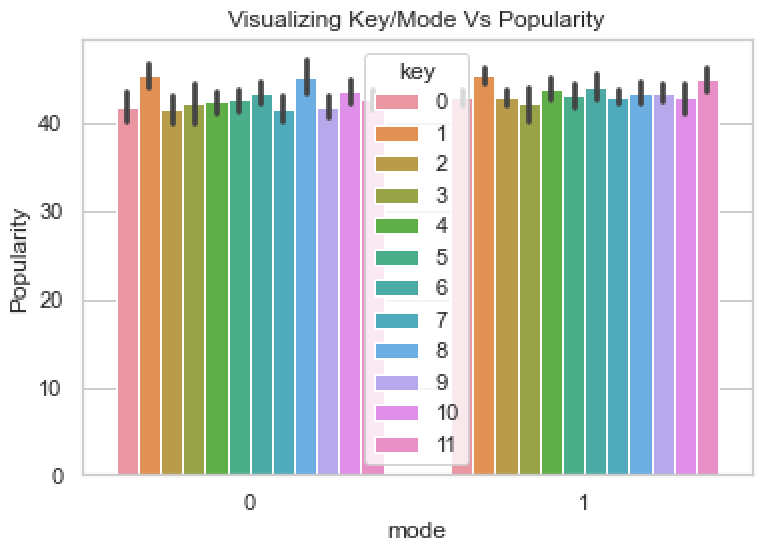
One interesting finding was that musicians who release more tracks do not necessarily score higher on the popularity scale. For example, we can see that the artist with the most songs (Omer Adam) has a lower popularity score than, say, The Rolling Stones, who landed number 4 on the list of artists with the most songs. This is a key finding because we cannot assume that an artist's popularity is directly related to the number of released songs.

On the other hand, there were some obvious connections between certain features. For example, there is a clear positive correlation between Energy and Loudness with correlation coefficient of 0.73. We can observe that the higher the Energy, the louder a song becomes.

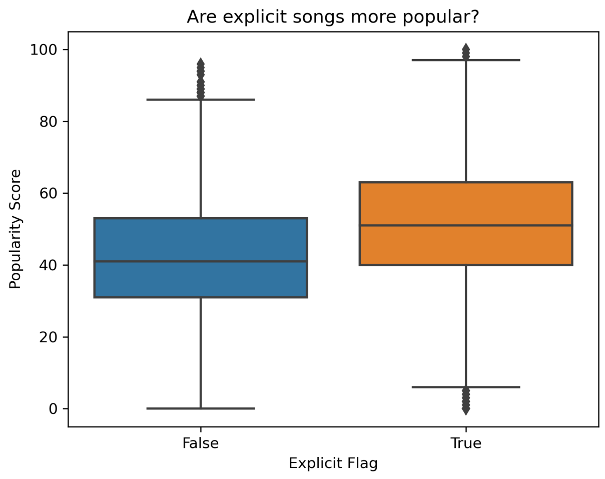


Songs with a higher energy value tend to be louder

Another interesting discovery was the effect of mode of the key used on Popularity values. For the same key, the popularity score associated with the major and minor scale varied. For example, we can see that G-major is less popular than G-minor (Key number = 7). In this dataset, key values are not evenly distributed.

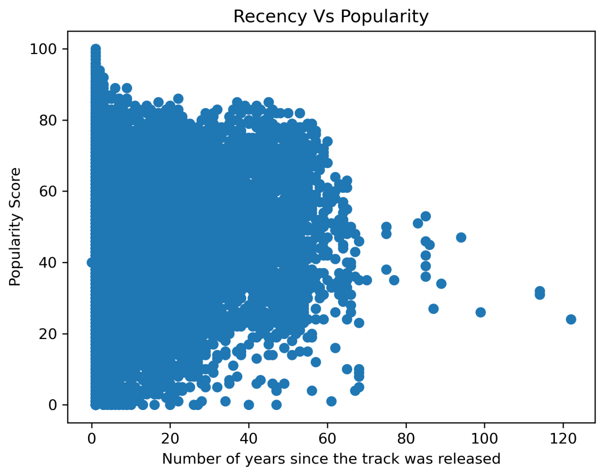


Based on the mode, the same key can produce different popularity scores



Popularity scores were generally higher for songs that had explicit content.

We also looked at how recently a track was released, and we can see that the popularity of a track decreases as time increases.



We can see that as time increases, popularity scores decrease

METHODS

The initial dataset had 26,752 records with 22 features. Within these records, some songs were repeated by the same artist and had the same popularity score. On further analysis, it was found that one artist can release multiple songs, but some songs were repeated in the dataset. The only column with varying values was "Playlist". However, this column has no effect on the popularity of the song. Hence, the original dataset was reduced to 18,551 rows with unique track names for each artist.

SPLITTING STRATEGY

Since the data was independent and identically distributed, without any group structure or time series data, the test set was created using a basic split method, and the train and validation sets were created using K-fold validation with 4 folds. The data was split as 80% other and 20% test, and then the 80% was divided using K-fold split.

DATA PREPROCESSING

Two features (Key, Time Signature) had numeric values with no inherent ordering. For example, even though Key has an order from 0 to 11, it does not mean that Key 1 is greater than Key 0. Hence, these two features were one-hot encoded. Mode and Explicit_flag were already binary-valued (0/1) features. Duration_ms represents the duration of the song in milliseconds. A better way to represent the duration of a song would be to use minutes as it is easier to interpret. Except for loudness, the remaining audio features had values between 0 and 1. Initially, loudness values ranged from -34.8 to 1.3 dB. As there was no clear minimum or maximum value, using the MinMaxScaler would not have been the best choice. So, all the continuous features were standard scaled.

PIPELINE

A column transformer was used to preprocess the data. The features that were not altered were allowed to pass through the transformer. This preprocessor was passed through a pipeline object, along with the supervised machine learning algorithm. Five different algorithms were tried - Ridge regression, K-nearest neighbors regression, Support Vector Machines Regression, Random Forest regression and XGBoost regression. The following hyperparameters were tuned for each algorithm -

S.No.	Algorithm Used	Hyper-parameters Tuned

1.	Ridge Regression	Solver = [sag, saga]
		Alpha=[1,0.1,0.01,0.001]
		Fit_Intercept=[True, False]

2.	KNN Regression	N_Neighbors = [1,3,5,10,50]
		Weights = [Uniform, Distance]

3.	SVM Regression	Gamma = [1e-3, 1e-2, 1e-1, 1e1, 1e3, 1e5]
		C = [1e-3, 1e-2,1e-1, 1e0, 1e1,1e2, 1e3]

4.	Random Forest Regression	N_estimators = [100, 150, 200, 250, 300],
		Max_depth = [3,5,7,11],

		Max_features = [sqrt, log2, none]

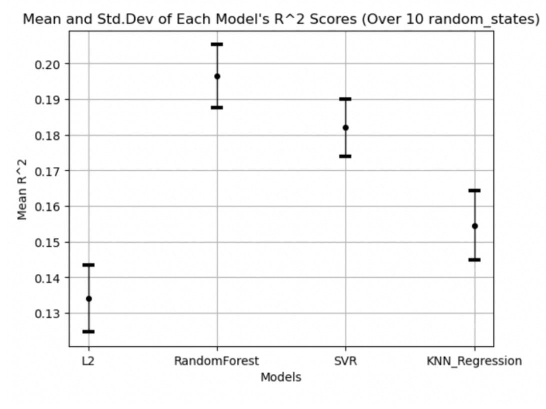
5.	XGBoost Regression	Learning_Rate = [0.001,0.01, 0.03, 0.05, 0.1]
		Max_depth = [1,5,10,30,50]

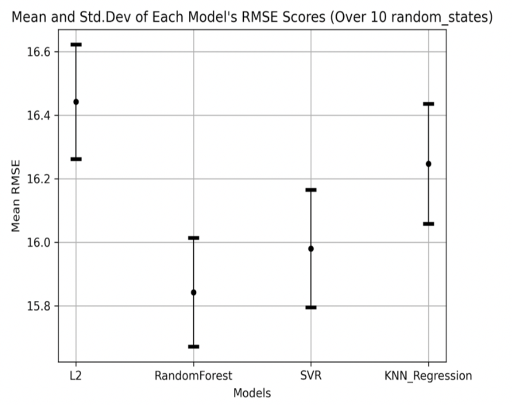
EVALUATION METRICS

Both R^2 and Root Mean Squared Error were calculated and compared for each model. The baseline R^2 score was 0.0, and the baseline RMSE score was 17.67.

Apart from hyperparameter tuning, each model was tested using different random state values. Each random state was used in the splitting strategy and while initializing the regressor. The final test scores were an average of all the test scores over multiple random states.

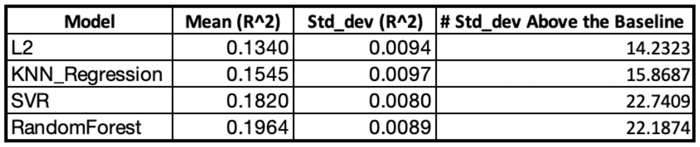
RESULTS



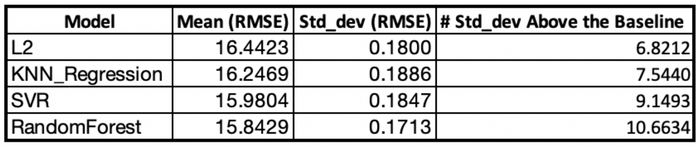


We can see that Random Forest Regressor performed the best based on both scores

XGBoost regressor was not included in the results because it did not perform better than the baseline. Even though random forest performed the best, the results were still only slightly better than the baseline which is why we see the following trend in true versus predicted values.

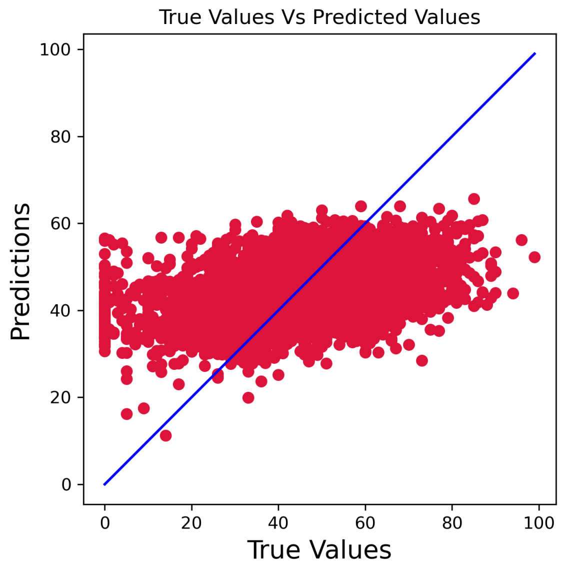


Model	Mean (R^2)	Std_dev (R^2)	# Std_dev Above the Baseline
L2	0.1340	0.0094	14.2323
KNN_Regression	0.1545	0.0097	15.8687
SVR	0.1820	0.0080	22.7409
RandomForest	0.1964	0.0089	22.1874



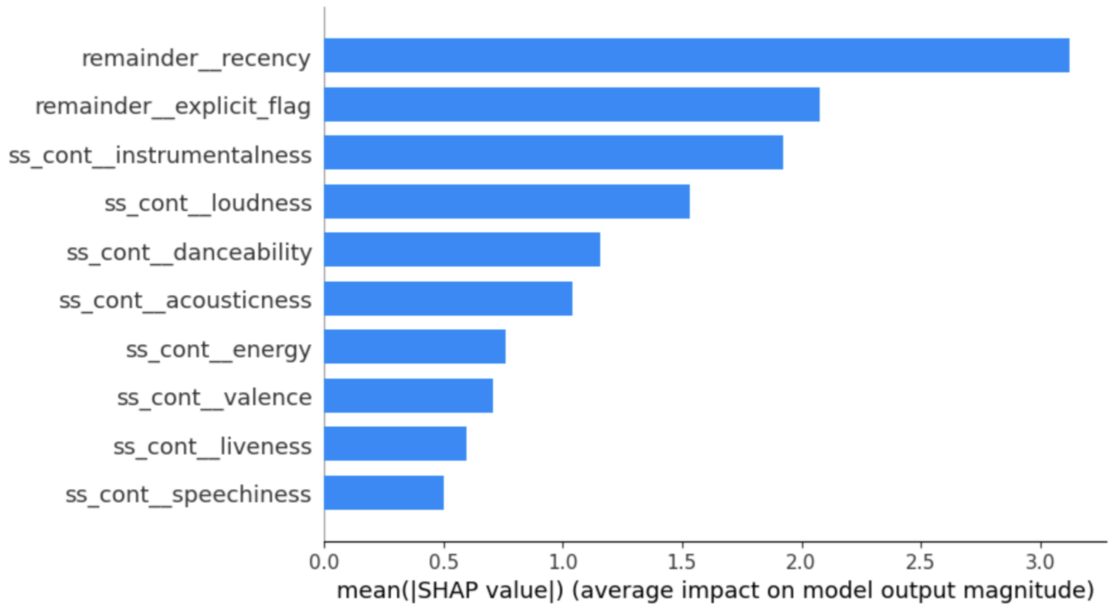
Model	Mean (RMSE)	Std_dev (RMSE)	# Std_dev Above the Baseline
L2	16.4423	0.1800	6.8212
KNN_Regression	16.2469	0.1886	7.5440
SVR	15.9804	0.1847	9.1493
RandomForest	15.8429	0.1713	10.6634

Table 1&2: Results observed on the test set

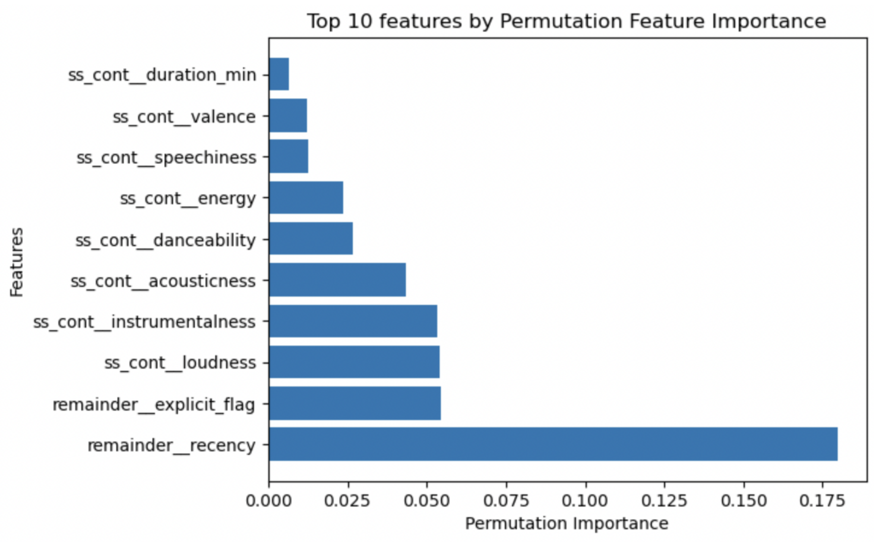


The blue line represents a perfect regressor. The closer the points are to the line, the more accurate the model

The top two features were recency and explicit flag, which were expected. The top audio feature was instrumentalness, which was also expected due to its high correlation with the target variable.



Top 10 features based on SHAP values

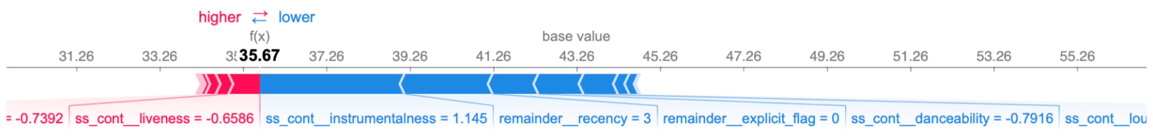


Top 10 features observed using Permutation Importance

In both the importance graphs, we can see that recency and explicit_flag are the top two features. The predictions were also analyzed using local SHAP values.

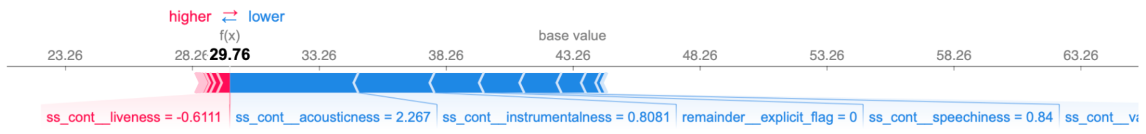
```
Test set prediction for index[200] =  35.668447178865485
```

Out[71]:



```
Test set prediction for index[100] =  29.76481041281493
```

Out[72]:



```
Test set prediction for index[0] =  50.573757389956945
```

Out[60]:

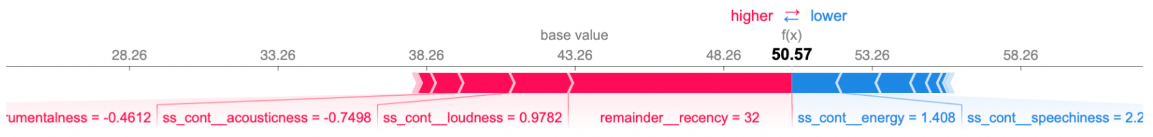


Fig 1,2,3: The red features contribute to higher predictions and the blue features contribute to lower values

As seen in the prediction above, the explicit flag is 0, which means False. Since we saw earlier that more explicit songs are more popular, we expect that this feature will reduce the popularity of the track. This is the exact trend that is followed here.

<u>OUTLOOK</u>

While these models provide a good starting point in predicting the popularity of a song, we can improve these results in the following ways -

Further feature engineering -
Since each variable had a low correlation with the target variable, further feature engineering or feature extraction would need to be done to extract more predictive power. Each feature could be integrated with others in certain ways.
<u>For example</u>**:** loudness and energy could be multiplied, acousticness and speechiness could be combined, etc.

In-depth EDA -
Instead of just looking at the number of years since release, we can look at whether the track was released over the holidays, or over the weekend.
<u>For example</u>**:** Is a song more popular if it is released on a Friday compared to a Monday?

Modelling Approach -
Could this problem be converted into a classification problem compared to a regression problem? The aim would be to predict a suitable threshold that defines popularity.
<u>For example</u>**:** Songs that have popularity scores > 50 are popular. Will classification models yield better results?

REFERENCES

1. Music Streaming Hits Major Milestone as 100,000 Songs are Uploaded Daily to Spotify and Other DSPs, William, C (October 2022) *https://variety.com/2022/music/news/new-songs-100000-being-released-every-day-dsps-1235395788/*
2. The 5 Best Music Streaming Services you can subscribe to in 2022, Cohen, S, Tricarico, A, and Blanchet, B, (August 2022) *https://www.businessinsider.com/guides/tech/best-music-streaming-service-subscription*
3. Spotify: Predicting Popularity of a Track, Azene, D (March 2022) *https://medium.com/@daazene/spotify-predicting-popularity-of-a-track-44c8da3daa34*
4. Predicting Song Popularity on Spotify!, Aali, A, (December 2021) *https://medium.com/@asad.aali/predicting-song-popularity-on-spotify-e61074865251*
5. Predicting Popularity on Spotify — When Data Needs Culture More than Culture Needs Data, Peker, P (June 2021) *https://towardsdatascience.com/predicting-popularity-on-spotify-when-data-needs-culture-more-than-culture-needs-data-2ed3661f75f1*
6. Lior, I. (2021) Spotify Multi-Genre Playlists Data. Retrieved October 2022 from *https://www.kaggle.com/code/siropo/using-spotify-s-audio-features*