Identifying Features that make a song popular on Spotify

Kevin Van Vorst (kpv23), Manya Walia (mtw62), Saaqeb Siddiqi (ss3759)

December 13, 2020

Table of Contents

1 Introduction

In a world where topping the trending popularity charts and becoming a viral sensation defines the success of a song and its artist, it has become increasingly important to crack the code behind what makes a song popular on Spotify - the major audio streaming service of the modern day. As of October 2020, Spotify boasts of almost 320 million active users all over the world. Through a platform following the freemium structure, it provides access to over 60 million songs [6].

Because of the rise in its popularity, Spotify also has the means to monetarily compensate artists for their contributions to the service according to the success of their music. Owing to this, it would be very beneficial to assess what features and/or combination of audio features determine the popularity of a particular song to what extent. We would like to use data from the Spotify API and perform extensive supervised learning analyses on it in order to create scalable models that accurately predict whether or not it will be popular on Spotify.

2 Data Processing

2.1 Data Cleaning

The dataset we will analyze consists of over 160,000 songs throughout 1921 to 2020 [1]. All the tracks and its information were originally collected directly from the Spotify Web API. Every song has a name, list of artists, release date, key, and is identified by a unique ID generated by Spotify. Each song has 19 features associated with it - 1 of which is a primary id key, 14 of which are numeric audio features, and 4 are categorical. Below is a table of all the features and their corresponding type:

Audio Feature	Feature Value Type
id	int
acousticness	float (Ranges from 0 to 1)
danceability	float (Ranges from 0 to 1)
energy	float (Ranges from 0 to 1)
duration_ms	int (typically ranges from 200k to 300k)
instrumentalness	float (Ranges from 0 to 1)
valence	float (Ranges from 0 to 1)
popularity	int (Ranges from 0 to 100)
tempo	float (typically ranges from 50 to 150)
liveness	float (Ranges from 0 to 1)
loudness	float (typically ranges from -60 to 0)
speechiness	float (Ranges from 0 to 1)
year	Ranges from 1921 to 2020
mode	0 = Minor, 1 = Major
explicit	0 = No explicit content, 1 = Explicit content
key	int
artists	string
release_date	date in yyyy-mm-dd
name	string

2.2 Data Preparation and Cleaning

First to clean the dataset all rows with missing values were dropped. Surprisingly, the dataset was complete and no rows were dropped. Next, two columns were dropped before the preliminary analysis was performed. The first dropped column was the “id” column which contained for each song, a unique string of integers and letters generated by Spotify to identify the track. That means with 169,909 unique identifiers, this column is irrelevant for our analysis and must be dropped. The other dropped column was the “release_date” column which contained the release date of the song. Compared to the “year” column, this data is too specific for our analysis as we question trends across years and not monthly or daily. Only the “year” column will suffice with our time related analyses and therefore the “release_date” column was dropped. Additionally, the “name” and “artists” columns were also dropped since our models will only depend on numeric audio characteristic inputs.

Finally, feature transformation is carried out in order to achieve an output classification status of popular/not popular, as there is not much inference to be derived from a popularity difference of a few points (example: not much difference between a song with popularity 24 or 28). Hence, a popularity binary classification column is initialized as y. Each song receives either a 1 if the popularity is greater than 0.80 and 0 if popularity is less than 0.80. This will serve as the value our models must predict (y_vector). After cleaning and preparation, the resulting dataset was 169,909 x 15 with a predicting vector of 169,909 x 1.

2.3 Data Visualization

Before performing our preliminary analyses, our data was visualized through a few scatter plots and histograms:

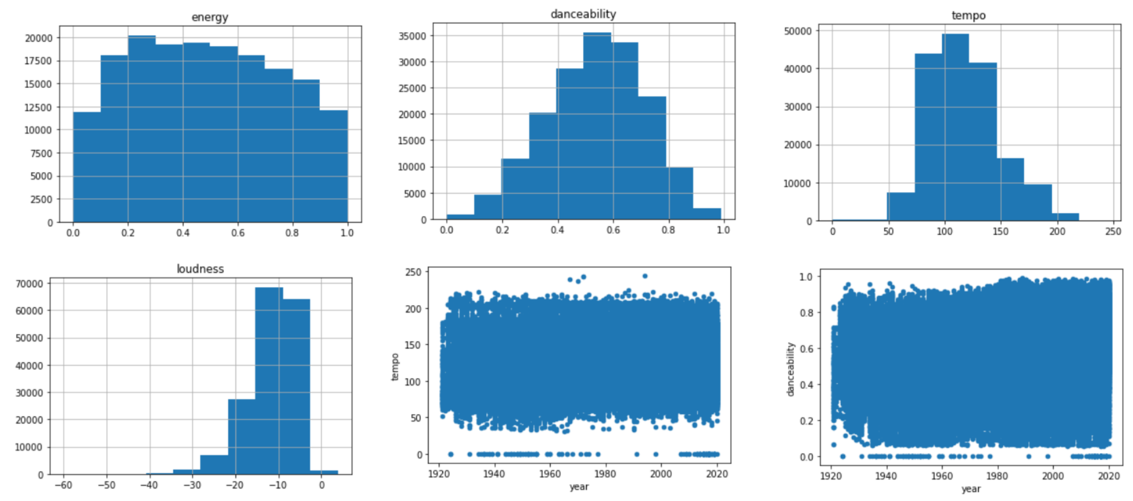

Figure 1: Various scatterplots and histograms of features.

3 Data Modeling and Analysis

3.1 Least Squares Regression

As a preliminary analysis, an ordinary least-squares error regression model was fit to 70% of the real-valued data and a summary of the results was generated. The adjusted R^2 of the model is 0.837, which indicates that our model fits the data decently well. The F-statistic probability is 0.00, which proves that all regression parameters are non-zero and that our regression model has good validity in fitting and predicting our data. The Mean Squared Error of the model on our training set is 100.52677995484363, and the Mean Squared Error of the model on our testing set is 101.03827809516288, which once again, are decent MSE values for such a large dataset. Based on the figure below and the Mean Squared Errors of the training and test set, using least-squares is not a sufficient model to fit our data. This may be because of the complexity of the dataset since it contains various different types such as large ranges of continuous values and boolean entries.

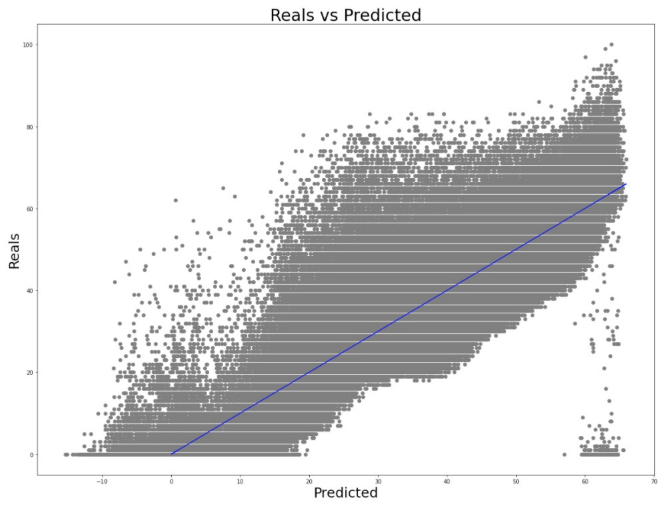

Figure 2: All column values incorporated into regression analysis.

3.2 K-Nearest Neighbors (KNN)

Another rather useful unsupervised learning tool of analysis is the K-nearest neighbors (K-NN) algorithm, which calculates the Euclidean distance between the new data point (which is to be classified) to all the training points. The new data point is classified as the same class to which majority of the K (most optimal) number of nearest data points belong [5]. K-NN is not a parametric model, making it steer clear of any underlying assumptions about our data. This quality proves to be extremely useful and advantageous as real world song data from Spotify is very unpredictable and fast changing, and not having to assume its linear separability or uniform distribution can help us stay away from any extremely harsh theoretical limitations. Even intuitively, it makes realistic sense for an artist who is seeking a breakthrough to try and replicate the audio features of a successful song which has already proven itself to be popular in the past.

In our analysis, a new song will be classified as popular (1) or not (0) after our K-NN algorithm with the optimal K value of 5 calculates all the distances from this song to the other songs in the training set, selects 5 nearest songs to it (in terms of features), and finally makes a decision influenced by the majority of these songs [3]. We split our dataset into a training set (80% of the data) and a testing set (20% of the data), and then perform a 5-NN classification analysis using KNeighborsClassifier from the Python package SciKitLearn. Upon comparing our predictions to that of the test set, we obtain very promising results. Our confusion matrix is:

$$\begin{pmatrix} 33861 & 1 \\ 120 & 0 \end{pmatrix}$$

Indicating that our model has good accuracy. Upon obtaining the metrics accuracy test score, we see a score of 0.9964392913895592, which is approximately 99.644%. Hence, our K-NN learning algorithm has done a very satisfactory job at predicting the popularity of a given Spotify song.

3.3 Support Vector Machine (SVM)

A Support Vector Machine (SVM) is a classification/regression method that outputs a predicted binary classification answer. Since we initialized the popularity_bool vector, we will use SVM as a supervised learning tool to predict the binary value {0,1} indicating if a song will be popular or not. Our dataset and desired results are a perfect candidate for performing a support vector regression with. Also considering the large amount of input features, a linear SVM was chosen. For this model, the revised dataset is once again split into two subsets: the training dataset and the test dataset. These datasets are divided into an 80%-20% split respectively. Then, utilizing the function LinearSVR() provided by the package Scikit-Learn, a SVM model was fit to the training data. Following this is applying this fitted model to the test dataset and the evaluation of the performance of the model. The model's confusion matrix is:

$$\begin{pmatrix} 84732 & 1 \\ 223 & 0 \end{pmatrix}$$

The model yields a score of 0.9973633099876406 which approximately 99.74%. This is an extremely high score for a model to receive especially since only a SVM was fit to the data. Therefore, we can conclude that applying SVM is acceptable to fit on our data and is a valid model.

3.4 Logistic Regression

The Logistic Regression model is another way to analyze and predict what evidently affects the popularity of a song. The logistic regression is evidently more effective on predicting over binary values, leading to the simplification of division of the dataset's popularity having a specific cutoff at the value of 70% popularity. This simplifies the regression model, allowing us to analyze the trends over the years more specifically rather than focusing on trying to predict the exact floating point value of the popularity.

After dividing the dataset into a training set and testing set with an 80/20 split, we used Scikit-Learn's LogisticRegression() function to evaluate the predictions of the popularity of a song. Over the entire dataset, the model appeared to get extremely accurate results with an accuracy of 97%. The confusion matrix was:

$$\begin{pmatrix} 33054 & 928 \\ 0 & 0 \end{pmatrix}$$

What was evermore interesting is how the model was ineffective over a yearly subset of the dataset in 2019. The naive model received a mere score of 61% over the subset, indicating the overreliance of a year to year data to predict whether a song is popular or not. What is a significant discovery is how the overall popularity of songs evolved over the course of years. It is extremely easy to predict how popular a song that came out before 2006 (the year Spotify was founded) rather than predicting whether songs that are more recent in creation.

Over the course of changing the subset of the dataset to include more recent years, it's pretty evident that year is the biggest influence to a song's popularity. The in depth analysis that immensely reduced the overall accuracy requires an in depth dive into the independent features that affect its popularity. To do that, some features were excluded from the dataset when implementing the logistic regression. At the end of the experimentation component, when only these features (danceability, energy, explicit, liveness, speechiness, tempo) were included, the accuracy of the model jumped to about 87%. These features overall show the power of how popular a song is in the past 10 years.

4 Discussion

4.1 Results

Four unique modeling techiniques were performed on a Spotify dataset containing a song's audio characteristics. Out of all of them, the least squares model performed the worst returning and R^2 of 0.837. A simple linear fit was not sufficient enough for this massive dataset. The other three models, KNN, SVM, and logisitc regression, performed very well with respective scores of 0.9964392913895592, 0.9973633099876406, and 0.97. These methods proved to be reliable and accurately predict whether a song is popular or not given it's audio qualities.

4.2 Weapons of Math Destruction

In Cathy O'Niel's book "Weapons of Math Destruction", O'Niel describes the dangers of the manipulation of predictive models and the interpretation of their results. She explains a model is a weapon of math destruction if the outcome is not easily measurable, has predictions with negative consequences, or creates self-fulfilling feedback loops [4]. Our model and it's purpose is only to inform us of trends in musical audio features. Our results or model do not dictate people to like a certain artist or genre based on our results.

The features we use provide sufficient enough information to get extremely accurate results for a few of our models.

5 Conclusion

Overall, the effectiveness of this analysis displayed how historic data that is evaluated on 'popularity' could be skewed to focus on things that involve the most clicks rather than focusing on normalizing the popularity with the decade to which the song came out. The most effective analysis appeared by creating a hardlined binary definition of what makes a song popular. Even then, the song's year affected each model's overall utility since most songs in the past were correctly classified as not popular. It took to excluding and including very specific features to truly correctly predict whether a song is popular or not. Overall, it appears that the 5NN implementation yielded the best results especially since it found songs that were most similar in framework to itself in order to evaluate whether or not it was popular. Since much of the features were similar, it was able to predict its popularity with relative ease. In fact, the 5NN method is much similar to the very early implementation of Spotify's recommendation algorithm, explaining how this algorithm yield such excellent results [2].

References

[1] Yamacc Eren Ay. *Spotify Dataset 1921-2020, 160k+ Tracks.* URL: `https://www.kaggle.com/yamaerenay/spotify-dataset-19212020-160k-tracks`.

[2] Malte Ludewig. *Empirical analysis of session-based recommendation algorithms.* URL: `https://link.springer.com/article/10.1007/s11257-020-09277-1`.

[3] Ritchie Ng. *K-nearest Neighbors (KNN) Classification Model.* URL: `https://www.ritchieng.com/machine-learning-k-nearest-neighbors-knn/`.

[4] Cathy O'Niel. *Weapons of Math Destruction.* Penguin Books, 2016.

[5] Scott Robinson. *K-Nearest Neighbors Algorithm in Python and Scikit-Learn.* URL: `https://stackabuse.com/k-nearest-neighbors-algorithm-in-python-and-scikit-learn/`.

[6] Spotify. *Company Info.* URL: `https://newsroom.spotify.com/company-info/`.