CMSC 491 Data Science Project Report

Rohan Ahuja, Daniyal Khan

UMBC

**Abstract**

The goal of this project was to analyze Spotify music data and use various techniques to predict the popularity score of a song, and then analyze popularity growth over time of a song. First, a dataset of Spotify Audio Features from April 2019, which consisted of data from the official Spotify Web API for 130,000 songs, was used to predict the popularity of a song. Then, two versions of the Spotify Audio Features dataset, one from April 2019 and the other from November 2018, were integrated together to predict growth or decline of a song popularity over time.

Note: TwoJupyter notebooks were used to analyze these separately for clarity.

# Datasets

The dataset used for the project was Spotify Audio Features, which contains audio features for 130,000 tracks collected from the official Spotify Web API. The data within the dataset was obtained through Spotify for Developers, which gives users access to the extensive catalog of Spotify data.

Two versions of this dataset were used. The first version consists of Spotify song data from April 2019, and the second version consists of Spotify song data from November 2018. The dataset from April 2019 was used to predict song popularity, and the dataset from November 2018 was integrated with the April 2019 dataset to predict growth or decline of song popularity over time.

The features in the dataset consist of the artist name, track ID, track name, and numerous audio features of the song. The following is a description taken directly from Spotify of each audio feature.

**Acousticness:**

A confidence measure from 0.0 to 1.0 of whether the track is acoustic. 1.0 represents high confidence the track is acoustic.

**Danceability:**

Danceability describes how suitable a track is for dancing based on a combination of musical elements including tempo, rhythm stability, beat strength, and overall regularity. A value of 0.0 is least danceable and 1.0 is most danceable.

**Duration\_ms:**

It is the duration of the track in milliseconds. We might want to convert this to seconds later.

**Energy:**

Energy is a measure from 0.0 to 1.0 and represents a perceptual measure of intensity and activity. Typically, energetic tracks feel fast, loud, and noisy. Loudness, timbre, onset rate and general entropy are some of the features contributing to this.

**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.

**Key:**

The estimated overall key of the track. Integers map to pitches using standard Pitch Class notation. If there was no key detected, a value of -1 was assigned.

**Liveness:**

Detects whether the song was played live by looking for audience in the background. Higher liveness values represent an increased probability that the track was performed live. A value above 0.8 provides strong likelihood that the track is live.

**Loudness:**

The overall loudness of a track in decibels (dB), averaged across the entire track. Loudness directly relates to amplitude. Values typical range between -60 and 0 db.

**Mode:**

Mode indicates the modality (major or minor) of a track, the type of scale from which its melodic content is derived. Major is represented by 1 and minor is 0.

**Speechiness:**

Speechiness detects the presence of spoken words in a track. Values > 0.66 indicate that the song pretty much contains spoken words everywhere. Values < 0.33 indicate that the track is mostly music and represent non-speech like sounds.

**Tempo:**

The overall estimated tempo of a track in beats per minute (BPM). Tempo is the speed or pace of a given piece and derives directly from the average beat duration.

**Time\_Signature:**

The time signature is a notational convention used in Western musical notation to specify how many beats are contained in each measure, and which note value is equivalent to a beat.

**Valence:**

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), while tracks with low valence sound more negative (eg. sad)

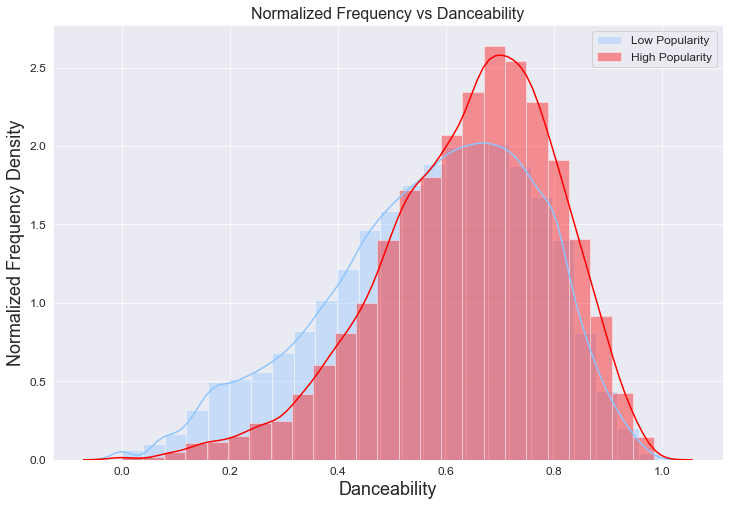
**Popularity:**

A popularity score between 0 to 100, describing how popular a track is.

# Analysis of Dataset

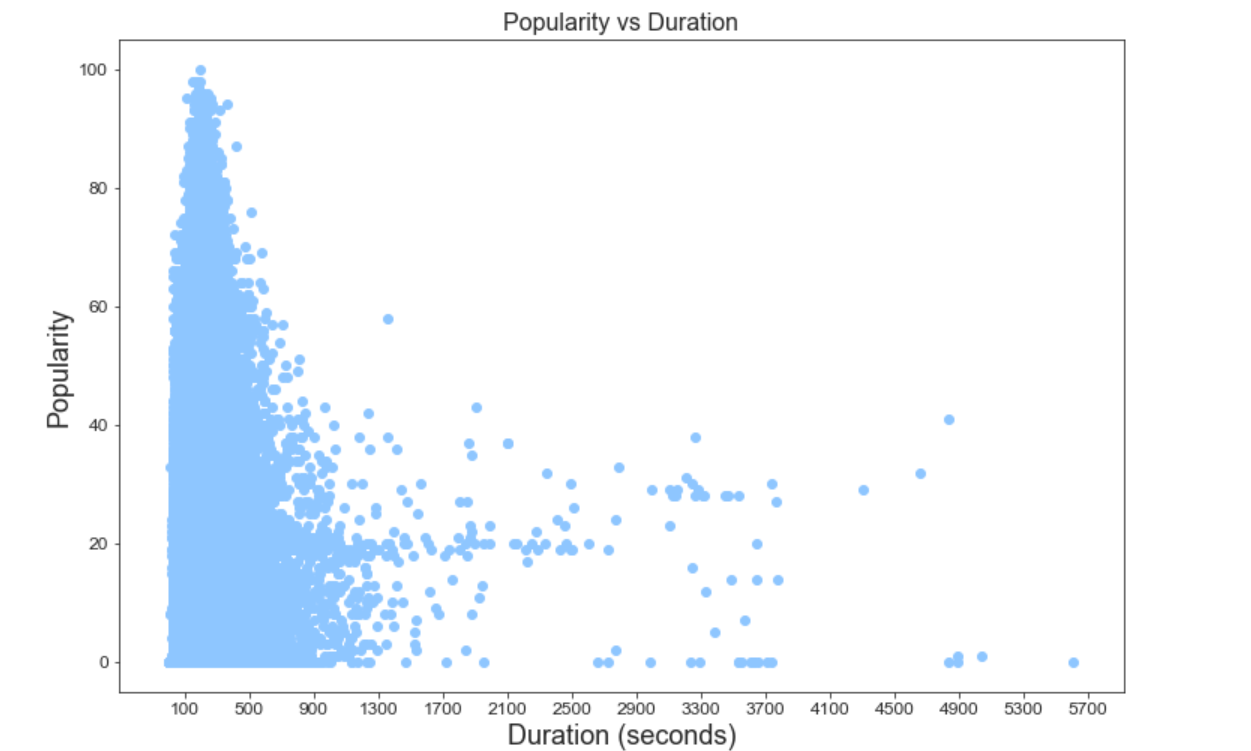
The first dataset analyzed was the Spotify Audio Features from April 2019. When cleaning the dataset, the time unit for the duration of a song was converted from milliseconds to seconds for readability. There were no duplicates in the data, so no rows needed to be dropped.

Popularity was examined to see how it related to the other audio features. First, popularity was related to danceability in the form of a histogram. Danceability is how suitable a song is for dancing with 0.0 being the least danceable and 1.0 being the most danceable.



Songs were categorized to have high popularity if their popularity score was greater than 50, and low popularity if their popularity score was less than or equal to 50. There was not a clear correlation between popularity and danceability, but the number of popular songs was greatest when they had a danceability of around 0.7. It is important to note that for the graph above, and many graphs later in this report, a normalized frequency was used instead of the actual number of songs, since the number of songs with low popularity is just too many.

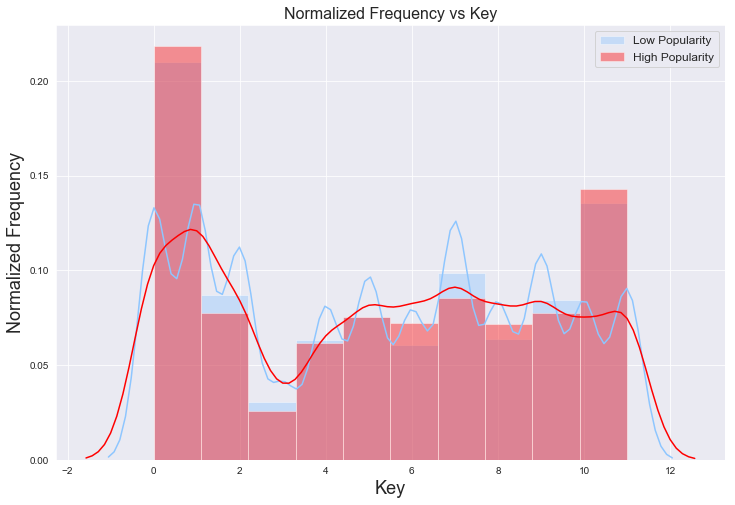
Next, popularity vs duration was observed in the form of a scatter plot. Duration is the length of the song in seconds.



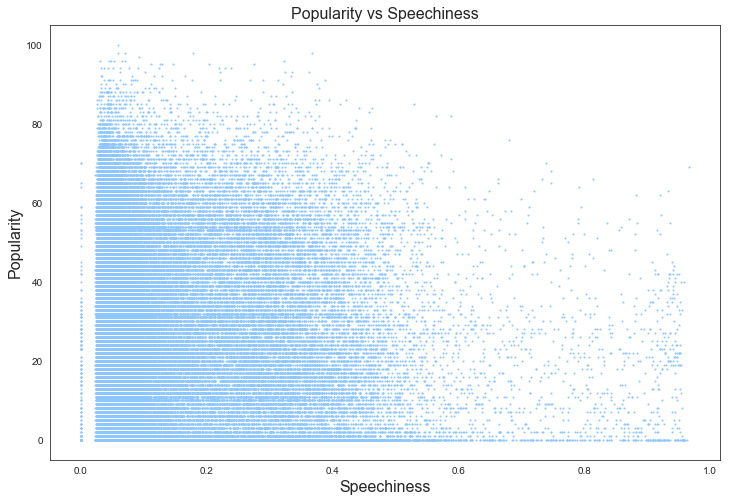
Songs which lasted between 200 to 300 seconds (3.5 to 5 minutes) were the only ones capable of being extremely popular. Songs that took too little or too much time don’t have a lot of popular songs. However, there was a sudden surge around the 2000 to 3000 second mark, which could be podcasts.

The key of a song could influence the song’s popularity, so a histogram was created for it. Songs were categorized to have high popularity if their popularity score was greater than 50, and low popularity if their popularity score was less than or equal to 50.

In the figure below, the extremes, 0 and 10, are best for highly popular values. Once again, a normalized song frequency was used. A key of 6 or 8 also seems to give reasonable results as compared to songs with low popularity.

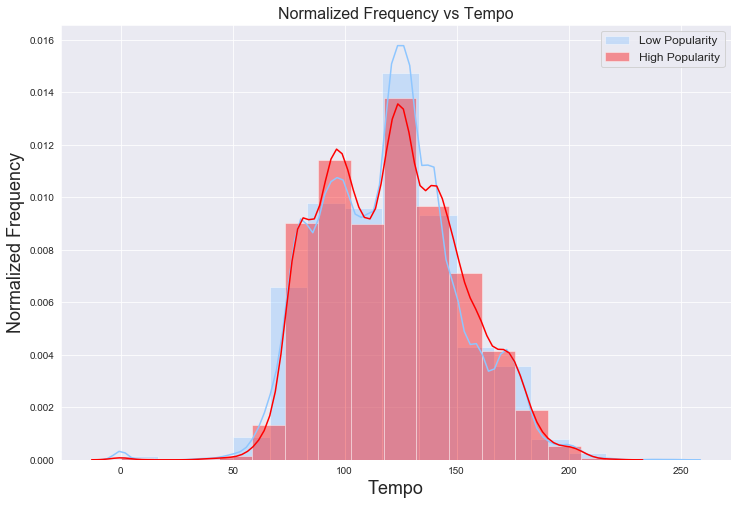


Popularity was then compared with the “Speechiness” of a song through a scatter plot. Speechiness detects spoken words in a song, with values greater than 0.66 meaning the song contains spoken words everywhere, while values less than 0.33 mean the song is mostly music.



Extremely high speechiness was not seen to be very popular. For the most popular songs, a speechiness around 0.10 appeared to be a good amount.

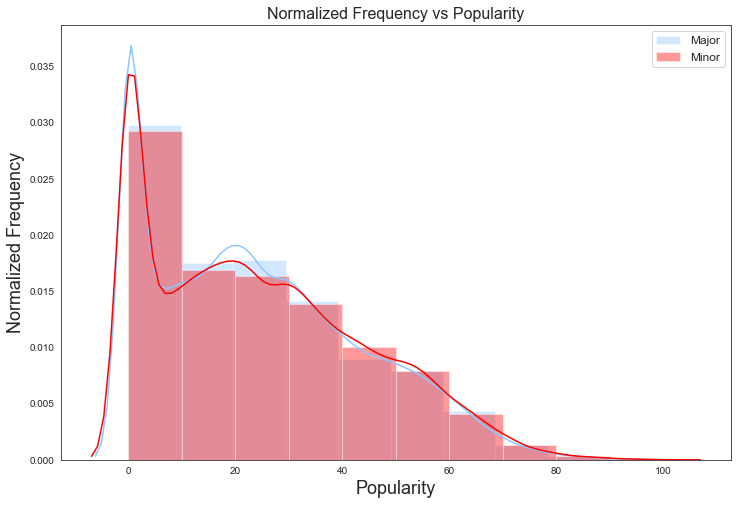
Next, tempo and popularity were analyzed through a histogram. Tempo is the speed of a song in beats per minute. Songs were categorized to have high popularity if their popularity score was greater than 50, and low popularity if their popularity score was less than or equal to 50.



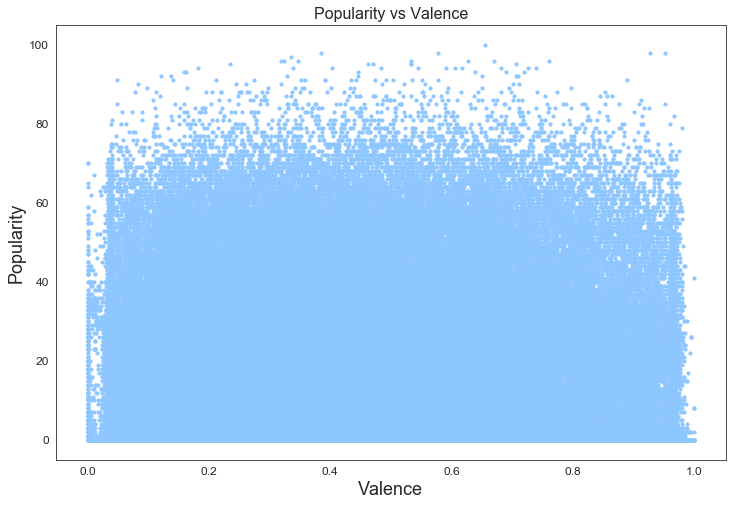
From the histogram, a clear correlation can be deduced, that a tempo range of around 70-160 results in more popularity.

Modality was then compared to popularity using a histogram. The mode of a song can be major or minor, from which the melody of the song is derived.

There was not a clear correlation able to be observed from the graph, but the most popular songs were minor. (See figure below)

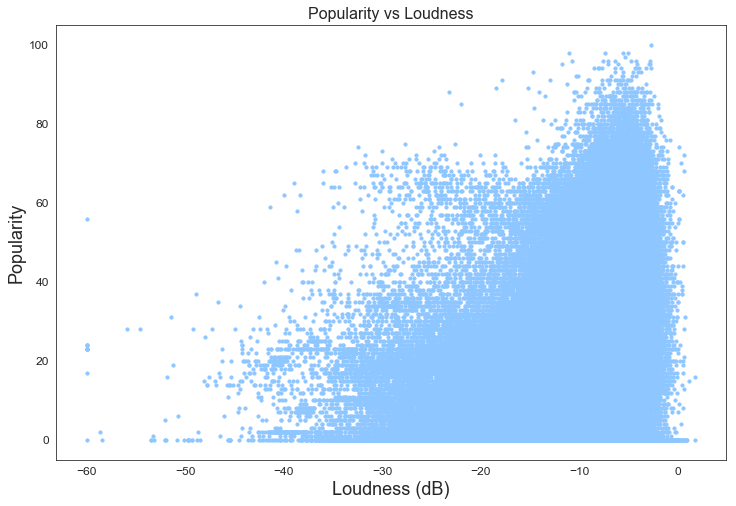


Next, a scatter plot was made to determine any possible trends between valence and popularity. Valence is the measure of positivity in the song, with higher values meaning the song is happier, while lower values indicate the song is sadder.



Based on the graph, all types of songs are popular, regardless of whether the song is happy/positive or sad/negative.

Lastly, a scatter graph was created to compare loudness with popularity. Loudness is the overall loudness of the song in decibels. Loudness also relates to energy, so we are also trying to analyze whether people prefer high energy songs or low energy songs.



There is a clear correlation that the louder the song, the more popular it is.

A correlation matrix was then created to analyze the relationships between all the features of songs. From the figure below,

**Most prominent correlations**

Positive:

Loudness and Energy

Danceability and Loudness

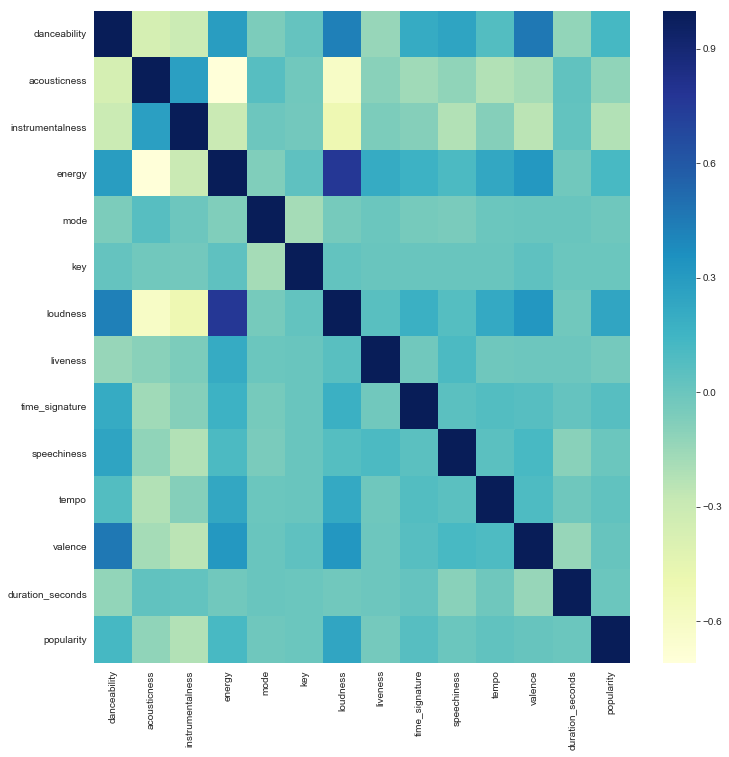
Danceability and Valence

Negative:

Acousticness and Loudness

Acousticness and Energy

Loudness and Instrumentalness



When looking at popularity, there aren't many correlations, however loudness is the most positively correlated feature, followed by danceability and then energy.

After observing the data and how the audio features related to popularity, Machine Learning algorithms were applied to the dataset to predict a song’s popularity. For all the algorithms, the data was split into training and testing datasets, with 20% of the data being used for testing and the rest for training. The audio features selected to be independent variables for training were: "danceability", "acousticness", "instrumentalness", "key", "loudness", "speechiness", "tempo", "valence", and "duration\_seconds". The dependent variable to be predicted was popularity.

**Machine Learning**

First a baseline was calculated. The baseline is the error if all the popularity values were predicted as the average value. The mean absolute error of the baseline was 16.3799. This is what the Machine Learning algorithms should beat and get less error than.

The first Machine Learning algorithm applied was Linear Regression, which predicts a dependent variable’s value based on its linear relationship with the independent variables. The data was fitted to a linear function, and the mean absolute error after testing it was 15.5612, which is already an improvement over the baseline approach.

The second Machine Learning algorithm used to predict popularity was a Random Forest regressor, which builds multiple decision trees during training time and then merges them together randomly for accurate and stable predictions. The random forest for the dataset consisted of 100 trees, and after testing and fitting the data, the mean absolute error was 14.6057, which is less than the error for Linear Aggression.

The next Machine Learning algorithm tried on the data was K-Nearest Neighbors, which uses the average of the values of the *k* nearest neighbors to predict the target variable. To predict popularity, *k* was set to 100, and the Euclidean metric, which is simply the straight-line distance between two points, was used to determine the closest neighbors. The mean absolute error after testing using the regressor was 15.5631, which is similar to Linear Regression, however Random Forests still performed better than both.

Then, Gradient Boosting Regressors were applied to the dataset. This algorithm uses decision trees as weak learners to make predictions, which are added together one at a time during which loss is calculated and minimized using a loss function. The algorithm was built using a least squares regression as the loss function and performed 500 boosting stages. The resulting mean absolute error was 14.8080, which was almost as good as Random Forests.

Lastly, a Multilayer Perceptron Neural Network was built. It is composed of perceptron, which produce output based on the linear and non-linear combination of inputs with weights. The multilayer perceptron consists of three or more layers: an input layer to receive data, an arbitrary number of hidden layers which act as the computation engine of the neural network, and an output layer to make a prediction regarding the input. The Multilayer Perceptron Neural Network modeled for the data consisted of four hidden layers, used the relu activation function for the hidden layers, used a stochastic gradient-based optimizer for weight optimization with a momentum of 0.9, and a maximum of 5000 iterations. After testing the model on the data, a mean absolute error of 14.9616 was obtained, which is a reasonable performance that is close but not better than Random Forests. Perhaps with more parameter tuning, Neural Networks would be the best approach. It is also very import to take note of the fact that the MLP was from the Scikit-learn library, which isn’t as popular for deep learning purposes.

# Integration of Datasets

Two versions of the Spotify Audio Features dataset, one from April 2019 and the other from November 2018, were integrated together to predict growth or decline of a song’s popularity over time.

The two datasets were integrated using an outer join. After integrating, the data was cleaned. First, the time unit for the duration of a song was converted from milliseconds to seconds for readability. There were a few rows that included null values, and this is because some songs didn’t exist back in 2018, so we had to drop these rows because they did not have any 2018 popularity. The “track\_id” feature was also dropped since it will not be used.

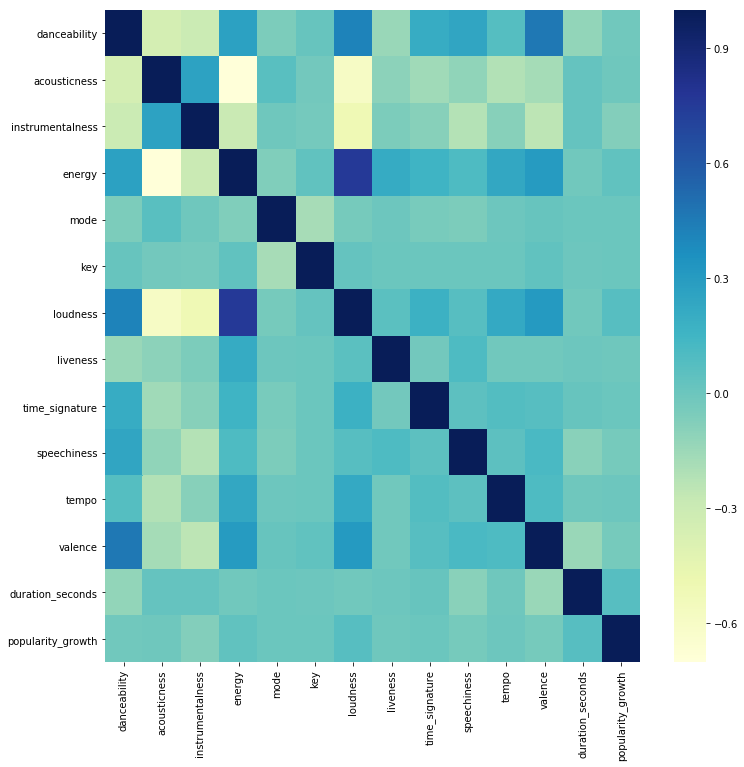
The popularity features in the integrated dataset became “popularity\_april\_2019” and “popularity\_nov\_2018”. A third feature was added, “popularity\_growth”, which is positive if the song became more popular and negative if the song became less popular. Popularity growth was simply the difference between popularity in April and popularity in November.

**Statistical Analysis**

A correlation matrix was then created to reveal any relationships between audio features and popularity growth (figure on next page).

However, there were no clear correlations between popularity growth and any other audio features, however popularity growth did have a little correlation with duration and loudness.

Since no meaningful information was obtained from the correlation matrix, a new column was created for binary classification. If a song maintained its popularity or grew in popularity, a value of “1” is assigned. If the song decreased in popularity, it is assigned a “0”.



Now that the binary classification column had been created, Machine Learning Algorithms were used to determine popularity growth/maintenance. For all the algorithms, the data was split into training and testing datasets, with 20% of the data being used for testing and the rest for training. The audio features selected to be independent variables for training were: "danceability", "acousticness", "instrumentalness", "key", "loudness", "speechiness", "tempo", "valence", and "duration\_seconds". The dependent variable to be predicted was “popularity\_maintain/growth”.

First, a baseline score was established by guessing all the values to be 1, which resulted in an accuracy rate of 62.83%. The goal is for this score to be beaten by the Machine Learning algorithms.

The first Machine Learning algorithm applied was Logistic Regression, which analyzes the relationship between one dependent variable and one or more independent variables to estimate the parameters of a logistic model. The Logistic Regression model applied to the data used the “lbfgs” algorithm for the optimization problem, and had a maximum of 1500 iterations for the algorithm to converge. The resulting accuracy score after testing was 62.94%, which is almost the same as the baseline score, only slightly better.

The next models built were Support Vector Machines, which predicts and assigns the test example to one of two categories. All the training example points are represented in space such that the two categories are divided by a gap as wide as possible. When given testing examples, they are mapped to the same space and their category is predicted based on which side of the gap the fall on. The Support Vector Machines built on the data had a maximum of 10,000 iterations and solved the dual optimization problem. The accuracy score was 62.94%, the same result as logistic regression.

The next Machine Learning model applied was a Multilayer Perceptron Neural Network. The model built for the data consisted of three hidden layers of sizes 15, 20, and 15, used the relu activation function for the hidden layers, used a stochastic gradient-based optimizer for weight optimization with a momentum of 0.9, and a maximum of 20,000 iterations. The resulting accuracy score from testing was 65.62%, giving a decent improvement over the baseline score.

The last Machine Learning model built was a Random Forest Classifier. The random forest for the dataset consisted of 200 trees with a maximum depth of 20, and after training and testing the data, the accuracy score was 65.45%. Although the Random Forest Classifier performed well, Neural Networks performed the best of all the algorithms.

# Insights

Information obtained from analyzing Spotify data has the potential to be extremely valuable. The music industry is worth billions of dollars, so any information regarding songs on the most popular streaming service, Spotify, could be highly beneficial to record labels and artists. In particular, popularity is the most critical aspect of a song, since hit songs lead to more money for record labels and greater success for artists.

Our data analysis showed correlations between certain audio features of songs and a song’s popularity. We were also able to build Machine Learning models to accurately predict a song’s popularity based on its audio features. This information could be used by people in the music industry, and most likely already is, to formulate how to create a hit song. They could create songs with the perfect audio features and guarantee a song’s popularity. The predictive models could also be used for existing songs, in the case that an artist wants to stay true to themselves and not use algorithms to write their music, the record label could still use a predictive model to see which songs from an album have the most potential to be a hit, and then make those songs the commercial singles. Perhaps in the future, machine learning models could actually create songs based on preferred audio features and actually write songs that become popular.

In the case that there was more time, more effort would be put into the Neural Network. Although the best model for predicting a song’s popularity is not too far off from the actual value, it can still be improved by tuning the parameters, resulting in better predictions. Also, instead of scikit-learn’s Neural Network, a library for deep learning such as PyTorch could have been used, if we had a little more time to apply this. Lastly, for predicting popularity growth, an older dataset could be found to integrate, which could result in better results and clearer correlations.