Music Recommendation

Using Spotify API Data

Irpan Yierpan
Computer Science
Rice University
Houston, Texas USA
ay32@rice.edu

Keifer Convertino
Computer Science
Rice University
Houston, Texas USA
kjc6@rice.edu

ABSTRACT

Using audio feature data publicly available via Spotify's Web API, we identified 5 clusters within the data using a Gaussian Mixture Model and were able to assign them with qualitative labels ("chill", "dance", etc.). Using these clusters and the audio feature data, we also employed a k-Nearest-Neighbors algorithm for recommending songs that were similar to a given input track based on both audio features and Gaussian Mixture Model generated cluster probability vectors, allowing users to input their favorite songs and receive recommendations that could also be granulated by genre or by our previously computed clusters. Lastly, we produced a polynomial regression model for predicting the popularity of a song based on its audio features. Ultimately, however, we were only able to achieve a testing accuracy(R^2 score) of 0.264 with this model, suggesting that achieving a highly accurate prediction for a song's popularity on Spotify would not be possible using solely audio features.

1 Introduction

1.1 Motivation

Music plays a big role in most people's lives. Whether through consuming or producing, music is a highly important creative outlet and form of entertainment. For this reason, we wished to create a tool that could help improve the way people are able to interact with music, both from a listener's perspective and from an artist's. By utilizing machine learning techniques and access to large datasets of songs and their features, we can provide just that.

1.2 Related Works

Spotify themselves provide the largest example of a music recommendation tool built upon data and machine learning [7]. They utilize a mixture of several strategies to provide a rather famous recommendation engine. They employ collaborative filtering, which utilizes listener connections and user feedback in order to recommend songs that user's who are similar to you enjoy. This type of system is only possible with access to large amounts of user data—things like time spent listening to certain songs, song likes, and visits to different artist's pages. Additionally, they employ natural language processing across the internet to form clusters and networks of artists. They can use this information to recommend artists or songs that are within the network of an artist that you already enjoy.

Lastly, they employ audio models that take raw audio data in order to find songs that are similar in things like time signature, key, mode, tempo, and loudness. Spotify uses a Convolutional Neural Network for their audio models. This last type of model in particular is of interest to us, as it largely uses data that Spotify makes publicly available through an API. Several other individuals have used this same data to make popularity predictions and analyze the importance of certain audio features, with some mixed results [1,6].

1.3 Our Goal

We are hoping to utilize publicly available audio features in order to do the following:

a. identify clusters in the dataset and assign them as qualitative "categories" (ex: "chill", "aggressive", etc.);
b. allow listeners to select song(s) that they like and receive recommendations in a category or genre;
c. allow an artist to input a song and receive a prediction for its popularity on Spotify, and then compare it to other popular songs (in terms of audio features);
d. package these services into a website that will be publicly accessible.

2 Methods

2.1 Data

Our dataset [9] was collected from Spotify's public API [2]. It contains over 228,159 entries (tracks) with 18 features each. Each song can be identified by a track ID, which can then be used in the Spotify API to access more information about that song. Outside of the features used for identification (track ID, track name, artist name), the dataset also contained audio features such as acousticness,

danceability, duration, time signature, and genre, etc. These were the primary features of our interests. Lastly, for each track, there was also a popularity score calculated by the Spotify API (discussed in detail in section 2.5).

2.2 Data Preprocessing

2.2.1 Filtering Tracks. Once we identified the goals of our project, we performed some exploratory data analysis with them in mind. During the analysis, we noticed a few problems with our dataset.

First, we found that our dataset contains duplicated tracks. There are multiple occurrences of a track with identical track ID, and the only feature that is different is the popularity score of the track. This is because popularity is calculated and updated periodically, therefore, a track will appear multiple times in the dataset with a different popularity score over time. Thus, to not introduce unwanted noise to our models later, we decided to combine the duplicated tracks by averaging their popularity score.

Next, when we looked through different genres of tracks in our dataset, we noticed that our dataset contained tracks from the “Comedy” genre, after checking a few “Comedy” tracks on Spotify, we found that these are not songs, they are actually stand-up comedy snips. Since we are only interested in songs, we decided to drop all the tracks from the “Comedy” genre.

After looking at the different genres, we plotted the distribution of each audio feature. However, we noticed that the duration feature is heavily skewed. For instance, its maximum value was 5.552917e+06 millisecond, which means the track is about 92.5 minutes long. We looked at the tracks with large duration values and found that these are actually podcasts rather than songs. We wanted to filter out the podcasts, but there was not a feature that could help us to identify them, so as a workaround, we only kept the tracks within 3 standard deviations of the mean of the duration feature as it yielded tracks with durations between about 15 seconds and about 9 minutes 47 seconds, which is a reasonable duration range for a song.

After combining duplicated tracks and removing all the unneeded ones, we are left with 142,683 tracks(about 62.5% of the original size).

2.2.2 Feature Selection. During our exploratory data analysis, we also analyzed each feature individually and decided on which features to keep.

For instance, while looking at the distribution of each audio feature, we noticed that the instrumentalness feature is heavily skewed, with about 85.96% of the tracks having a value close to 0, meaning that most songs are not instrumental in the dataset. Thus, we decided to drop this feature. Similarly, about 86.95% of the tracks have a time signature value of “4/4”, so we also dropped it as there are not many time signature variations between tracks.

After looking at the distribution of each feature, we wanted to see if there were any features that captured redundant information. For instance, we have the mode, the key, and the valence features. The mode of a track usually impacts the “feeling” of the track, that is, a track with major mode usually gives the listener a more “happy” and “positive” feeling and minor mode usually gives the listener a more “sad” and “negative” feeling [5]. Similarly, different keys also produce different feelings. In general, just like the modes, major keys tend to result in a more positive feeling and minor keys tend to result in a more negative feeling [8]. On the other hand, the valence feature also captures the positive and the negative feeling that a track produces, with a higher valence value meaning a more positive feeling. Therefore, since these features provided some redundant information, we decided to drop the key and the mode and only keep valence as keeping the numerical feature rather than the categorical ones allowed us to make the final features in our dataset all numeric.

Moreover, we also calculated the correlations between each of the numerical features in our dataset (Figure 1). We found that the energy feature is highly correlated with the loudness feature with a correlation coefficient of +0.81. Thus, we decided to drop the energy feature. We kept the loudness feature because compared to the energy feature, it is more correlated with the popularity feature which is what we are interested in predicting. In contrast, we also have the acousticness feature which is highly correlated with both the energy and the loudness feature with a correlation coefficient of -0.72 and -0.7 respectively. However, we decided to keep the acousticness feature because, despite a high correlation with loudness, acousticness is still a very unique feature of a song, energy and loudness are highly correlated because they capture some redundant information as energetic songs tend to be louder to give the listener the energetic feeling, but acousticness is a very different kind of feature compared to energy and loudness.

After performing all the feature analysis and dropping unneeded features, we are left with 8 numerical features

(acousticness, danceability, liveness, loudness, speechiness, tempo, valence, popularity) and 4 categorical features (track ID, artist name, track name, genre).

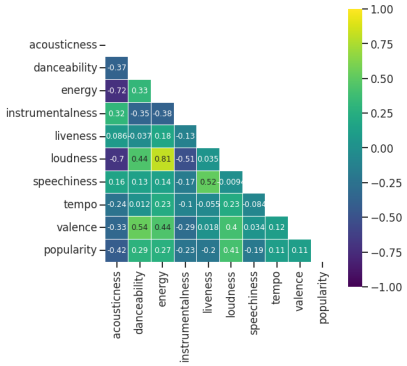

Figure 1: **Feature Correlations**

2.2.3 Normalization. The final numerical features were all on very different scales, so as the final step of the preprocessing, we wanted to normalize all the numerical audio features (that is, numerical feature except for popularity as we want to later predict this value and will not include it in clustering analysis nor in the recommendation modeling). We decided to use min-max scaling to normalize these features because all the features have an upper and lower bound.

2.3 Cluster Analysis

As we mentioned before, the first goal of this project is to identify clusters within our dataset, analyze the distinct features of each cluster and come up with qualitative categories to label each cluster. To achieve this, we needed a clustering algorithm.

We found previous work has been done in a Medium article using the K-Means algorithm [4]. The author of this article performed K-Means clustering on a very similar dataset where he used the Elbow method on the sum of squared distances of samples to their closest cluster center to find an optimal number of clusters. Then he analyzed the audio features of each cluster to give each of them a label such as “Dance” and “Wind Down”. We wanted to do a similar analysis, however, we want to change the clustering algorithm and the way to find the best number of clusters.

2.3.1 K-Means vs GMM. In our analysis, instead of using the K-Means algorithm, we decided to use the Gaussian Mixture Models(GMM) to cluster our dataset. This is because K-Means classifies each song to one and only one cluster, however, a song can belong to multiple different categories (for example, a song can be “happy” and “chill” at the same time). On the other hand, GMM gives a probability of the songs being in a cluster for every cluster. In other words, with GMM we can classify a song to be in multiple clusters rather than one, so we decided to use GMM instead.

2.3.2 Finding the Optimal Number of Clusters. In the article with K-Means clustering, the author used the Elbow method on the sum of squared distances of samples to their closest cluster center to find the optimal number of clusters. The Elbow method yielded k=4 for the optimal number of clusters, however, he used k=5 because it seemed to show better results for labeling the clusters with distinctive categories. Thus, this method does not necessarily give the best number of clusters, so for GMM we are using a different metric to find the optimal number of clusters.

We decided to use the silhouette score, which is calculated using not only the distance between samples to their closest cluster center but also the distance between samples to their next closest cluster center. In other words, it is a score used to measure the separation distance between clusters, so the higher the score, the more distinct each cluster is. Our goal is to find the number of clusters that maximize the silhouette score so that the clusters we ended up with have very distinct features.

A problem we encountered was that with the size of our dataset, it may take days to compute the silhouette scores, so instead of clustering the whole dataset with GMM and computing the silhouette scores, we randomly sampled 5% and 10% of our data, and tried to cluster the subsets into between 2 and 20 clusters, and computed the silhouette scores after each clustering step. As we can see from the figure(Figure 2), the 5% and the 10% subset of the data yielded a very similar trend, which gave us confidence that the subsets are reflecting the whole dataset. We also see that when the number of clusters is 3, we achieve the maximum silhouette score. However, we wanted more than 3 clusters, so we picked the second maximum silhouette score where the number of clusters is 5.

2.3.3 Labeling the Clusters. After running GMM with 5 clusters, we wanted to find what features made these clusters different from one another. Therefore, as shown in the figure (Figure 3), we plotted the distribution of each audio feature for each cluster on top of the distribution of these audio features in the whole dataset, and we labeled if any of the audio features for each cluster has significantly higher or lower values compared to the distribution of the

whole dataset. Finally, we came up with the category label for each of the clusters based on their unique audio features (Table 1).

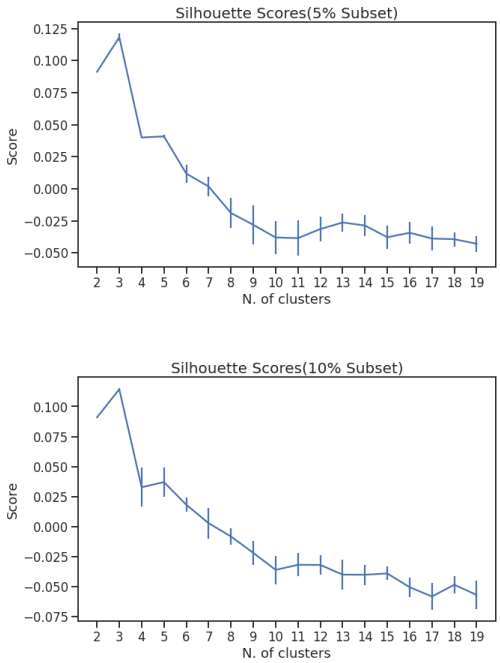

Figure 2: **GMM Silhouette Scores**

2.4 Song Recommendations

Making song recommendations is a key part of our project. However, since we do not have access to the wealth of user data and metrics that Spotify does, we can only rely on the available audio feature data to make recommendations.

With these features in mind, we decided that a distance comparison (via k-Nearest-Neighbors) between the audio features of tracks could be a good way to find songs that are similar to an input song. Furthermore, we can granulate these recommendations by allowing a user to specify a genre or category of particular interest. For example, a user could specify that they want to get Pop song recommendations, or use our previously computed song "categories" ("chill", "loud", etc). K-Nearest Neighbors also lends itself well to a web application, as a user can easily specify how many songs they want to get recommendations for, and a kNN model can return this easily (just modify k).

We also experimented with using a Gaussian Mixture Model to produce inputs for our kNN model. The idea here is that we can cluster our songs with a GMM model, then produce the probabilities that a song is in each cluster (a vector) for

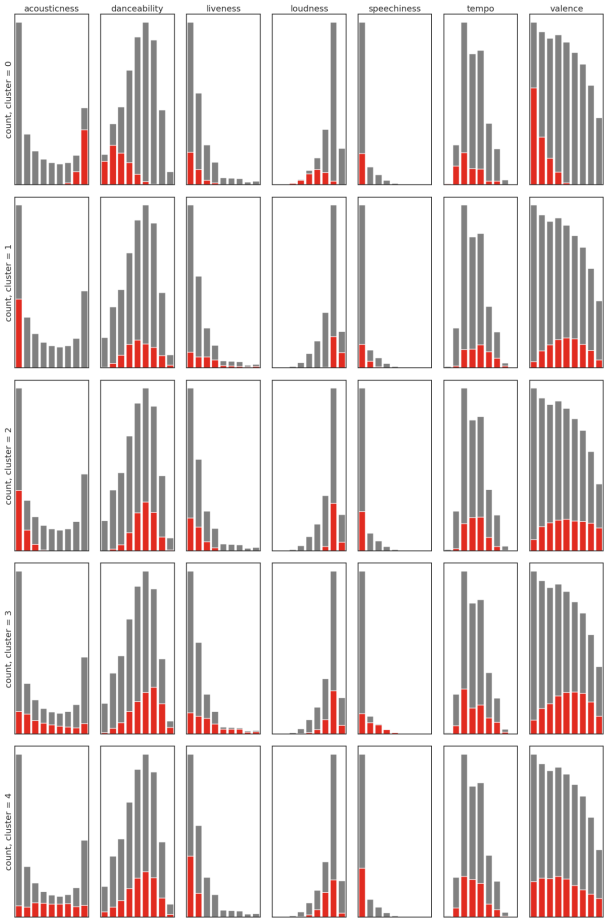

Figure 3: **Distribution of Audio Features for Each Cluster**

Culster	acousticness	danceablity	liveness	loudness	speechinessss	tempo	valence	Label
0	Low	High			Low			Chill
1	Low		High	High				Loud
2		High	High		High		High	Happy
3	High	Low	Low	Low	Low	Low	Low	Acoustic
4		High	Low		Low			Dance

Table 1: **Cluster Labelling**

each of the songs in the dataset, and then ask the model do the same for an input song. We can then run kNN on this vector generated from the input song against the vectors for all the songs in the database. In doing so, we are essentially getting the songs that GMM thinks are the most similar in terms of cluster assignment, which could indirectly tell us that the songs are similar. However, a few issues arise with this method. For one, it does not lend itself to utilizing the categories we previously computed in our clustering. We will have to rerun GMM clustering on our previously computed clusters to generate the probability

vectors. Additionally the issue of selecting the number of components arises. When examining all the data (getting a recommendation irrespective of genre or category), it makes sense to use 5 as our number of components, as we had previously settled on this number during our cluster analysis. However, if granulated by genre or category, the number of components becomes more ambiguous, as we have not done cluster analysis on the individual datasets for genre or category. Ultimately we decided on utilizing 5 as our number of components irrespective of what a user selects for genre or category for simplicity.

2.5 **Popularity**

Another major part of our project is to help musical artists gain insight into the audio features that may make their tracks more popular. Currently, Spotify calculates a popularity score for each track that has been uploaded to the platform. According to the Spotify Web API Reference [3], Spotify uses real-time user traffic data to calculate the popularity of each track, and a higher popularity score indicates that the listeners have been listening to this track a lot recently. Therefore, being able to predict the popularity of a track will be very valuable to an artist. Thus, we wanted to help the artist to predict the popularity of their tracks using the audio features so that they will be able to compare their tracks with the other most popular tracks, and it can help them improve their tracks to gain more popularity.

2.5.1 Popularity Analysis. Before trying to come up with a model to predict the popularity, we wanted to do some analysis to see how the different audio features correlate to the popularity of the tracks.

We looked at a previous work where others tried to analyze what features made the top 100 songs on Spotify from 2017 popular, and in this article, the author found that most popular songs had high danceability, energy, and valence, and low liveness, speechiness, and acousticness [1]. We performed a similar analysis on our dataset and got similar results. As we can see from the figure (Figure 4), comparing the top 100, 1000, and 10000 songs with the highest popularity score in our dataset, they all have high danceability and loudness, medium valence and tempo, low liveness, speechiness, and acousticness. Moreover, we also compared the mean audio features of the top 1000 songs across different genres, and as we can see from the few examples in the figure (Figure 5), popular songs from different genres have very different mean audio features, which assured us that in our final model, we must let the artist choose a specific genre (it can be an “All” genre as well) that they want to predict in.

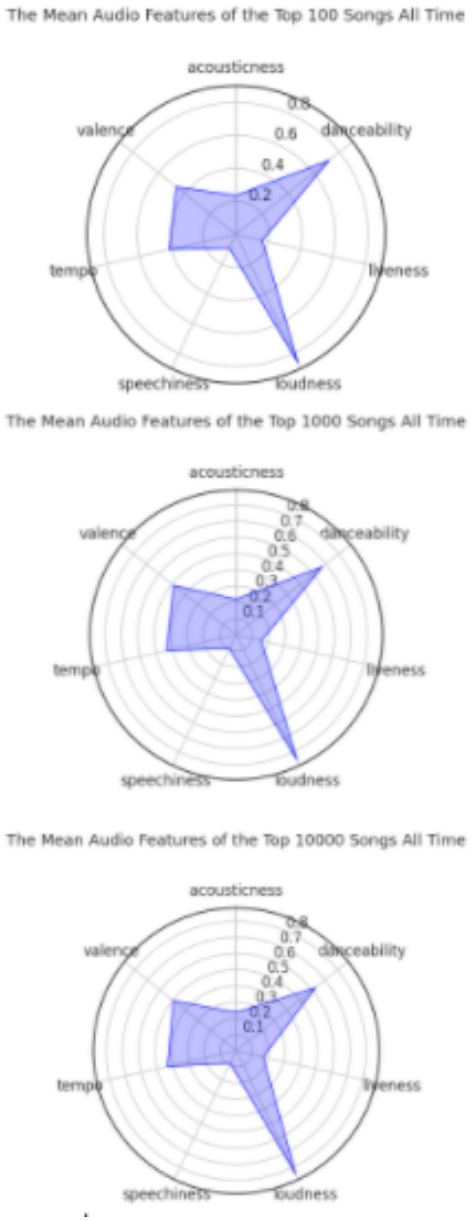

Figure 4: **Mean Audio Features of Top Songs**

2.5.2 Popularity Prediction. After analyzing the audio features of the most popular songs, we started to come up with a popularity prediction model.

First of all, we had a few requirements for our ideal model, that is the final model must have a high cross-validation and testing accuracy because we wanted to provide reliable

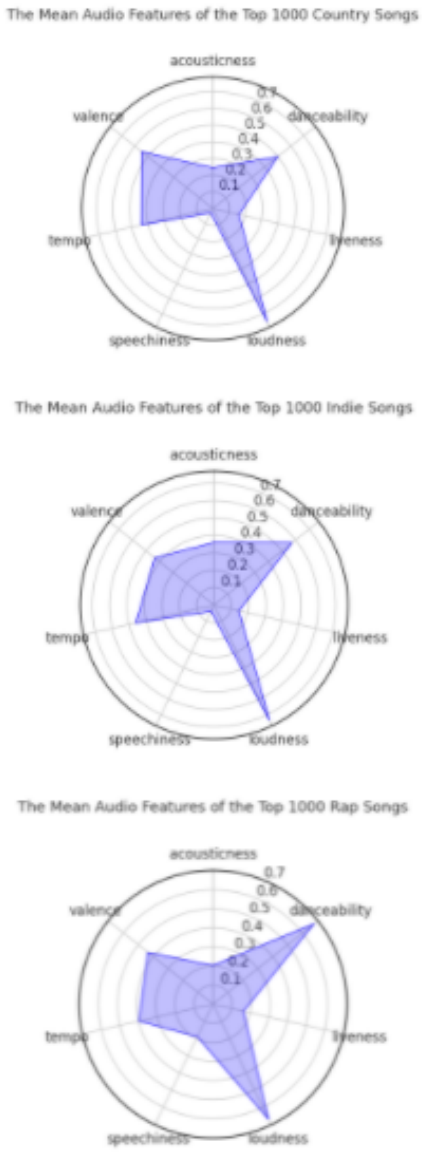

Figure 5: **Mean Audio Features of Top Songs Different Genre**

predictions for the artists; it also has to have a reasonably high training accuracy without overfitting the training data; and it must be able to run in a reasonable amount of time because we want our final product to be able to serve the artists without frustrations. With these requirements in mind, we split our data into 75% training data and 25% testing data with all the numeric audio features as input features, and the popularity score as target variables.

All the models we used to train and test our data are shown in the table (Table 2). We started with a simple linear regression model, then we tried polynomial regression models for degrees from 2 to 6. We did not try any polynomial regression models for degrees greater than 6 because our training data only had 7 input features, thus, a polynomial regression model of degrees greater than 6 will overfit our data. Moreover, we also tried a support vector regressor, however, due to the size of our data, it took way too long to run, so we did not obtain any results. Lastly, we tried XGBoost Regressor, Decision Tree Regressor, and Random Forest Regressor, for all of these models, we tuned all of their hyperparameters with 5 fold cross-validated grid search, and the best results were recorded in the table. As for the results, all the accuracies were measured in terms of R^2 scores, as it helps us explain how well the regression predictions approximate the real popularity scores of the tracks. Lastly, the fitting time of each model is also measured as we want a model that fits the data in a short enough time(under 10 seconds).

We compared all of the modeling results, and arrived at the conclusion that the polynomial regression model with degree = 4 was the one that best fits our requirement. First of all, it yielded the highest testing accuracy. Other models that yielded similar testing accuracies were the random forest regressor and the polynomial regression model with degree = 5. However, the random forest regressor had a training accuracy of 0.896, which we suspected is evidence of overfitting, and the fitting time was about 81 seconds, which is too high for an interactive application. On the other hand, the polynomial regression model with degree = 5 had a higher training accuracy, but a slightly lower cross-validation and testing accuracy and a longer fitting time. This again made us suspect that this was the point where the polynomial regression models started to overfit, thus, we decided that a polynomial regression model with degree = 4 was the best choice.

Model(Tuned)	Training Accuracy	Cross Validation Accuracy(cv=10)	Testing Accuracy	Fitting Time
Linear Regression	0.196	0.196	0.197	<1s
Poly Regression (d=2)	0.234	0.233	0.233	<1s
Poly Regression (d=3)	0.252	0.250	0.251	1s
Poly Regression (d=4)	0.269	0.262	0.264	3s
Poly Regression (d=5)	0.281	0.263	0.263	11s
Poly Regression (d=6)	0.294	0.237	0.240	38s
Support Vector Regressor	na	na	na	>1hour
XGBoost Regressor	0.268	0.259	0.259	11s
Desicion Tree Regressor	0.185	0.183	0.183	<1s
Random Forest Regressor	0.896	0.264	0.263	81s

Table 2: **Model Comparison**

3 Discussion

When it came to song recommendations, the kNN model built on audio features appears to perform better than the one built on Gaussian Mixture Model generated probability vectors. While this is of course to personal taste, the recommendations supplied by the model built on audio features were empirically of more similar genres and "sounded" more like the songs inputted by a user. Additionally, the model built on audio features has the added benefit of recommending the same songs every time you call for recommendations based on an input song, while the GMM model will recommend a set of different songs each time (as the model must be regenerated on each prediction).

Our popularity model, like the one produced in a related work [6], was somewhat middling in performance. It seems as though solely audio features are not a strong indicator of what the popularity of a song will be. This makes some sense, as even though there are some genres that dominate the charts (pop, hip-hop, etc.), there are significant differences in their sounds both between and within genres themselves. Especially considering how modern music is consumed today, where personalities and social media have a large impact on which music becomes popular, it would probably require different data, or at least additional data, to make a more accurate popularity prediction. Things like an artist's number of followers across different social medias, historical data for their popularity, or how well connected they are to other artists based on social media follower overlap or discussion (consider smaller artists who are associated with much larger ones, and therefore receive a boost in popularity) would all likely play a big role in the amount of traffic a song receives. Using just audio features would likely only be able to predict a metric deriving the song's "listenability" or "pleasantness" —completely agnostic of context or who the artist behind the song actually is. This might certainly be a factor in what makes a song popular, but it is not the end all be all.

4 Conclusion

Empirically, the kNN song recommender built on audio features performs well. A next step would be to add some form of feedback for the recommendations on our frontend so that we can gather user data to determine whether kNN on audio feature or the kNN on audio feature GMM probabilities generate better recommendations. Moreover, perhaps we could further tailor these recommendations to a user (for example, by weighting features that are deemed more important to that individual more heavily). On the other hand, our popularity predictor was only somewhat accurate, suggesting that to predict the popularity of a track more successfully, one would need to incorporate more and different features, such as social media numbers or levels of connections to other artists.

CONTRIBUTIONS

Both individuals performed preliminary data analysis and exploration. Irpan was responsible for the data pre-processing, clustering analysis, and popularity predictions. Keifer was responsible for the song recommendations and creating the website for users to interact with the models. All of the source code for this project can be found at https://github.com/ixxan/Spotify-Music-Recommendation, and the website we built for this project can be found at https://dsci303-music-recommender.herokuapp.com/.

REFERENCES

[1] Ashrith. 2019. What makes a song likeable? (March 2019). Retrieved from https://towardsdatascience.com/what-makes-a-song-likeable-dbfdb7abe404 [Accessed 11 December 2021].

[2] Developer.spotify.com. 2021. Web API | Spotify for Developers. [online] Available at: <https://developer.spotify.com/documentation/web-api/> [Accessed 11 December 2021].

[3] Developer.spotify.com. 2021. Web API Reference | Spotify for Developers. [online] Available at: <https://developer.spotify.com/documentation/web-api/reference/#/operations/get-several-tracks> [Accessed 11 December 2021].

[4] Koh, J., 2019. Profiling my Favorite Songs on Spotify through clustering. [online] Medium. Available at: <https://towardsdatascience.com/profiling-my-favorite-songs-on-spotify-through-clustering-33fee591783d> [Accessed 11 December 2021].

[5] M4arts.org. 2016. [online] Available at: <https://www.m4arts.org/post/2016-06-10-why-is-major-happy-and-minor-sad> [Accessed 11 December 2021].

[6] Philip Peker. 2021. Predicting popularity on Spotify-when data needs culture more than culture needs data. (November 2021). Retrieved from https://towardsdatascience.com/predicting-popularity-on-spotify-when-data-needs-culture-more-than-culture-needs-data-2ed3661f75f1 [Accessed 11 December 2021].

[7] Sophia Ciocca. 2020. How does Spotify know you so well? (April 2020). Retrieved from https://medium.com/s/story/spotifys-discover-weekly-how-machine-learning-finds-your-new-music-19a41ab76efe

[8] The Music Studio. 2020. How Chords and Key Impact Emotion in Music - The Music Studio. [online] Available at: <https://www.themusicstudio.ca/blog/2020/08/how-chords-and-key-impact-emotion-in-music/> [Accessed 11 December 2021].

[9] Tran, S., 2021. Spotify Song Popularity Prediction. [online] Kaggle.com. Available at: <https://www.kaggle.com/huanntran100/spotify-song-popularity-prediction/data> [Accessed 11 December 2021].