Predicting Song Genre Using Machine Learning and Spotify API

Ethan David Rayala, Bodhi Harmony, Kevin Dai, Ahmadou Bamba Diouf

1 Introduction

The aim of this machine learning project is to develop a model that can predict the genre of a song using audio features obtained from the Spotify API. The project involves collecting audio features for a number of songs, including features related to rhythm, melody, and timbre. These features are then used to train our machine learning algorithm, which is capable of classifying songs into five different genres. The project also involves evaluating the performance of the model using an accuracy score. The results of the project have potential applications in music recommendation systems, music streaming services, and music analysis.

2 Previous Work

There are a few professional studies that have used machine learning to predict a song's genre based on audio features obtained from music platforms such as Spotify and Last.fm. One study by Yang et al. (2016) used a dataset of over 500,000 songs and extracted audio features using the Spotify API. The authors then used various machine learning algorithms, including logistic regression and neural networks, to predict the genre of the songs. They achieved an accuracy of over 70% using a logistic regression model. Another study by Lee et al. (2018) used a similar approach but also incorporated user listening history and demographic information to improve the accuracy of the model. Through the use of a deep learning model, they achieved an accuracy of over 80%. Another project by Kumar (2022) used the K-Nearest Neighbors classification algorithm to build a music genre classification program that has approximately 70% accuracy. The last project we surveyed was done by Ranganath (2023) and used the GTZAN Genre Classification dataset to train their data, which consists of 1,000 audio tracks of 10 genres. They used a variety of features to visualize the audio aspects of their dataset. For the actual machine learning algorithm, which achieved 92.93% accuracy, the authors chose to use the Convolutional Neural Networks method to train their model.

3 Methodology

3.1 Locations

In our Github, (`https://github.com/erayala15/COMP-562-Final-Project-Spotify`), we have a file called 'forfun.py' which runs the Spotify API to get the set of songs and write them to a file called 'bigfile.json'. Our model is in 'maquinaprender.ipynb' which reads the content of 'bigfile.json'.

3.2 Intro

Data collection was the bulk of our work. All of our data collection was using the Spotify API, whose documentation is provided to us through Spotify's website. Spotify provides an API endpoint that, when given a set of song ids, returns attributes about that song. Those attributes (all set in a scale from 0.0 to 1.0) include:

Acousticness: A confidence measure from 0.0 to 1.0 of whether 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.

Energy: Represents a perceptual measure of intensity and activity. Typically, energetic tracks feel fast, loud, and noisy. For example, death metal has high energy, while a Bach prelude scores low on the scale.

Instrumentalness: Predicts whether a track contains no vocals. "Ooh" and "aah" sounds are treated as instrumental in this context. Rap or spoken word tracks are clearly "vocal".

Liveness: Detects the presence of an audience in the recording. Higher liveness values represent an increased probability that the track was performed live.

Mode: Indicaties 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. The more exclusively speech-like the recording (e.g. talk show, audio book, poetry), the closer to 1.0 the attribute value.

Tempo: The overall estimated tempo of a track in beats per minute (BPM).

Valence: 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).

Our goal is to use these specific attributes about a song to train a model to predict the genre.

3.3 Challenges

The first challenge was to find a set of songs of a given genre. When using Spotify's search API to find a song, one may enumerate a particular genre as part of the query parameter. However, Spotify does not attribute any genre to a particular song - rather, Spotify assigns genres to artists and albums. Our workaround was to use Spotify's search API, note which song ids were returned, and retroactively add the genre tag to those searched songs to use in our ML model. Unfortunately, when we used Spotify's search API, we were returned a set of songs that, while tangentially related to the requested genre, often did not pass the 'ear test' for that genre. Additionally, Spotify returned an inconsistent number of songs per each request. Preliminary testing with sets of songs retrieved this way yielded an accuracy ranging from .5-.7 with our model. However, we note that if we are extremely generous with our search parameters, creating a training set of 'classical piano', 'electronica' and 'screamo' music exclusively, we could reach an accuracy of .96!

3.4 The Playlist Solution

After spending time to pursue finding songs with Spotify's search API, we decided to use official Spotify playlists to find songs of a certain genre (we still used our approach of retroactively adding a genre tag). Of course, this incurs two main negatives: the set is likely biased, and the sample size is small. However, we believe that this approach is optimal given our time constraints. While the set of returned data may be biased, the playlists we found were popular, meaning the songs in those playlists are likely to be indicative, or defining of a certain genre. Additionally, manually running the 'ear test' for songs returned this way - we found that these songs were much more 'obviously' part of the genre, where songs found through the search API were questionable at best from the 'ear test'.

3.5 Model

We decided to use scikit-learn and Python to create our model. Firstly, we created a dataframe with the data to be used in our model. Next the data was split into training and testing sets - the testing set was used to teach the model to predict a genre based on provided input and expected output, and the testing set was used to evaluate the performance of the model. We then called the LabelBinarizer() function of the scikit-learn library to convert our categorical variables to a binary form because most machine learning algorithms require numerical inputs. Afterwards, we created a matrix to train with the aforementioned attributes. With this matrix, we ran scikit-learn's RandomForestClassifier() function to make 100 decision trees to be used to classify our data when training. The fit() function was then used to fit the forest model to the training data. Lastly, the predictions were made on the testing set utilizing the predict() function on the first model.

3.6 Evaluation

We used sklearn's 'score' method to analyze the accuracy of our model. This method returns the mean accuracy of the test data on our genre label; the accuracy is between 0.0 to 1.0. We observed a few key findings. When using just two genres, the accuracy was very high. The accuracy decreased as we added more sets of genres. We also were able to improve the accuracy of our model by picking which attributes to include in the model. Danceability, speechiness, and tempo seemed to be good predictors of genre. However, we noticed that mode - the key of a song - was only 1 or 0 representing a major or minor key for a given song. When we removed this attribute from our model, our accuracy tended to improve by a few hundredths.

4 Conclusion

4.1 Results

Ultimately, we picked 50 songs from 5 genres: Metal, Salsa, Jazz, Country, and Reggaeton. Our model yielded an accuracy of .94! With this configuration of songs, we noticed that our model had great results for Metal and Jazz...

4.2 Limitations

Our largest limitation is the relatively small sample size of this work. We had 250 total songs, 50 of which were used for testing. In the future, we would have found a way to either improve our implementation using the spotty Spotify search API, or to rewrite our implementation to include more songs from the Spotify official playlists. One potential way to counteract this is to take the Spotify curated playlists of new music in a genre over a larger period of time, as the playlists update weekly.

4.3 Future Work / Impact

Our work is important because this work can be used to, in a more empirical way, observe the most defining characteristics (from the attributes defined by Spotify) of a given genre. More analysis to this end could be done (aside from increasing the sample size) in algorithmically tweaking both the genres evaluated, and the attributes of a given song that are fed into a model. Our model was able to accomplish its task without referring to existing known databases of songs. There are many avenues which could still be explored, such as how important certain attributes are in differentiating between two specific genres: For example, when experimenting with the classical genre vs rap, using only the two attributes acousticness and speechiness yielded 100% accuracy; however, when using the two attributes valence and liveness for the same genres yielded only 80% accuracy. It is completely possible that certain attributes are key for

discerning between specific genres, and also possible that certain attributes are universally good predictors.

References

1. C. Ranganath, "Music Genre Classification using CNN," Clairvoyant Blog, 11-Apr-2019. [Online]. Available: `https://www.clairvoyant.ai/blog/music-genre-classification-using-cnn/`. [Accessed: 08-May-2023].

2. D. Lee, Y. Lee, H. Lee, and J. Lee, "Using Deep Learning and User Data to Predict Music Genres," in Proceedings of the 2018 International Conference on Information and Communication Technology Convergence (ICTC), Jeju, Korea (South), 2018, pp. 447–449.

3. scikit-learn. "RandomForestClassifier - scikit-learn 0.24.2 documentation," [Online]. Available: `https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.RandomForestClassifier.html`. [Accessed: 08-May-2023].

4. S. Kumar, "Music Genre Classification Project using Machine Learning Techniques," Analytics Vidhya, 29-Mar-2022. [Online]. Available: `https://www.analyticsvidhya.com/blog/2022/03/music-genre-class`