Music Genre Prediction

Thomas Currie

Project URL: <https://github.com/Kraftcur/Project1>

**ABSTRACT**

Music and their genres are an ever-changing part of culture in every country. The Music Genre Prediction apps goal is to help to categorize the music genres based on real data from the song. I formulated the problem as a binary classification problem and applied a decision tree classifier to solve it. The model is made up of 8 determining factors: energy, liveness, tempo, speechiness, acousticness, instrumentalness, danceability, and valence. After completing this app, I have found it has around 43% accuracy for the test data I have given, and it can be improved with a larger test set with more diverse genres.

# INTRODUCTION

1. **Background**

Knowing the genres of songs is an important part of the music industry. It helps to organize and identify artists in order to accurately display their creativity to the world. There is much debate on what makes a song one genre but not another. Knowing this, how do we as a culture classify the genre of a song? Is there a way to specifically determine the genre through data rather than opinion? Every song has audio features associated with it. Each feature is a numeric value which classifies how strong or weak the song is in that area. Using this data, we can try to determine the genre based on facts.

1. **Goal**

Can the genre of a song be determined from audio features of a song alone? I will be applying a classification tree to see if the result can give an accurate representation of the genre of a song.

1. **Method**

The data collected is from a Spotify API dataset as well as the Genius API.

1. **Challenges**

The project posed many challenges. Early challenges included the way to search a song and always get a result and letting the user know that the application is finding the correct song. To overcome this challenge, I had to find a way to search for a song just like you would search on google. To do this I used the Genius API and obtained the top results for a search. With that request, I can get the top result along with the album cover and artist. I can then show this information on screen to confirm to the user the song that was found is the one they are looking for. Another challenge was getting the audio features from Spotify. Spotify’s API for python, Spotipy, has a feature to search a song by track name, but that will not always match the genius search. To solve this problem, I pass the artists name from the Genius search to the Spotipy search function. I then loop through the results of the track search to find the correct track title and artist combination. When the correct combination is found I can use another Spotipy function to search for audio features based on a unique trackID. As the project came together, one large challenge presented itself. The challenge was with the actual training data I found with genres matched to the audio attributes. I found that the training data is not diverse in the types of genres it had making the model not as accurate as it could be. To help solve this problem I looked into clustering, but there were so many different types of genres that the clusters did not always contain similar genres. Instead, I did some more preprocessing on the original dataset to limit the number of genres to only eleven main categories. This greatly improved the accuracy of the model.

1. **Results**

I found that the model does work well for some tracks, and the accuracy of my model is 43%. With the current model my hypothesis is not proved, but is still feasible.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.

*CSE881-2015*, Month 1–2, 2004, City, State, Country.

Copyright 2004 ACM 1-58113-000-0/00/0004…$5.00.

# DATA

The only raw dataset I used was the Spotify Hot 100 dataset. This dataset contains song information from the top charts for the past 20 years. It puts genre with the audio features making it good for applying a classification model on. The file is in CSV format and was downloaded from data.world: [Spotify Billboard Hot 100 2000-2018](https://data.world/typhon/billboard-hot-100-songs-2000-2018-w-spotify-data-lyrics). The data contains 31 columns specifying song title and artist along with the genres and audio features. It also includes some other information not needed for this app that must be discarded. There are some missing values that need to be addressed as well. The file is 18.5 MB.

In the preprocessing stages I discarded missing values, did some string manipulation and spread out the data to account for multiple genres for one song. Filtering out the empty rows, and deleting columns I don’t need. The following is a raw row of data I used:

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Genre | energy | liveness | tempo | speechiness | acousticness | instrumentalness | danceability | valence |
| [u'dance pop', u'pop', u'post-teen pop'] | 0.724 | 0.12 | 102.061 | 0.0486 | 0.0945 | 0.00000168 | 0.601 | 0.508 |

**Table 1: Raw genre data row example with unnecessary columns removed**

I then spread out the genres into multiple rows for classification:

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Genre | energy | liveness | tempo | speechiness | acousticness | instrumentalness | danceability | valence |
| dance pop | 0.724 | 0.12 | 102.061 | 0.0486 | 0.0945 | 0.00000168 | 0.601 | 0.508 |
| pop | 0.724 | 0.12 | 102.061 | 0.0486 | 0.0945 | 0.00000168 | 0.601 | 0.508 |
| post-teen pop | 0.724 | 0.12 | 102.061 | 0.0486 | 0.0945 | 0.00000168 | 0.601 | 0.508 |

**Table 2: First structure of data for prediction**

After finding that too many genres were hindering the performance, I condensed every song to one of my eleven main genre categories: rap, pop, hip hop, country, r&b, rock, folk, metal, jazz, classical, or latin. The previous genre table would be condensed to:

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Genre | energy | liveness | tempo | speechiness | acousticness | instrumentalness | danceability | valence |
| pop | 0.724 | 0.12 | 102.061 | 0.0486 | 0.0945 | 0.00000168 | 0.601 | 0.508 |

**Table 3: Final structure of data**

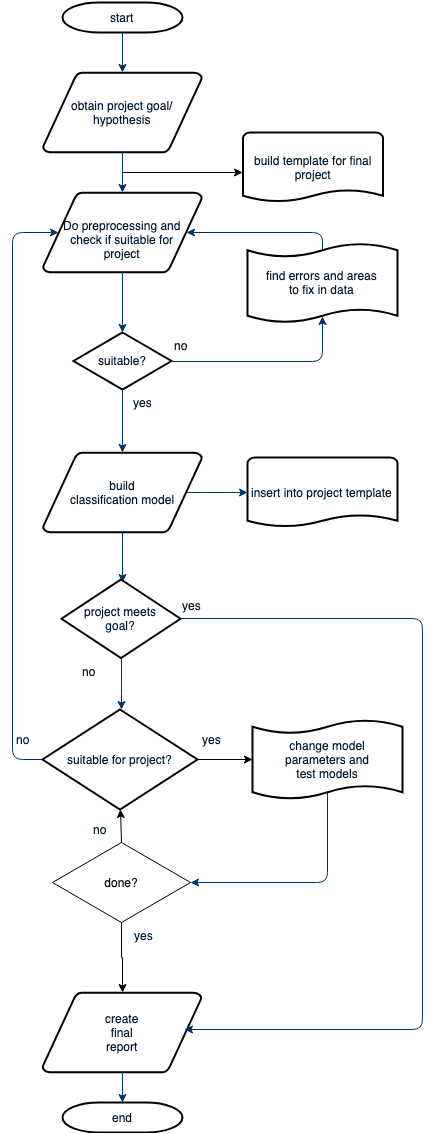
Therefor all the songs were more of a generalized category and the model would be better fit with the target attribute being the genre and the predictor attributes being the rest. The final csv is a 723 KB file and contains 6815 rows each with the structure as shown in Table 3. The preprocessing files as well as testing files I created to do these steps are in the folder Preprocessing on the GitHub repository. Each python file has a description at the top to describe what it does.

# METHODOLOGY

I started this project by creating a template to work on. I used Pythons library Django to create a web app that can be deployed to localhost. For html styling, I used Bootstrap. I then moved to a testing folder where all of my preprocessing and testing was done. I used Pythons scikit-learn toolkit to perform many different classifications, I ended up using the decision tree classifier to perform my classification. I tested different combinations of tree depth and other parameters to decide which one worked best for my application. I landed on a max depth of 7. Also in this testing folder, I learned how to use my Spotify and Genius API’s in the way I needed them. This was very helpful. Once I found how to get the data I wanted, I put it into the project template and didn’t have to change it.

Here is a brief summary of the code I have written for this project.

* Project1 folder: this is the folder that contains the entire Django project. From this folder use terminal command ‘python manage.py runserver’ to start server on local host (API keys needed).
* Project1>delta>data: This is a folder that contains the three iterations of preprocessing that I tested. ‘Hot100OneGenre.csv’ is the one I used.
* Project1>delta>templates>delta: This folder contains three html files I created.
* Project1>delta>views.py: this folder contains all final functions to connect to API’s, train model, and return the results.
* Project1>Testing: this folder contains all the preprocessing/testing files I used to build the project.

Below is a general flowchart of how I created the project.

# Software used:

* Scikit-learn: <https://scikit-learn.org/stable/index.html>
* Django[1]: <https://www.djangoproject.com/>
* Spotipy API: <https://spotipy.readthedocs.io/en/latest/>
* Genius API: <https://docs.genius.com/>
* Pandas: <https://pandas.pydata.org/>

# EXPERIMENTAL EVALUATION

This section describes the experimental setup and results.

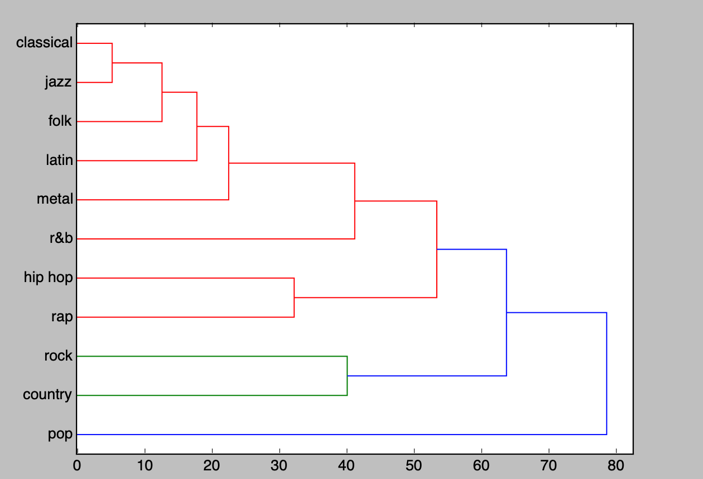
## Experimental Setup

This section should include:

1. Built on MacOS Mojave
2. I compared my results to what the popular opinion was about the songs I would test as well as another study using frequency to predict genres [2]. The results of my test was a maximium of 43% accurate making it not a good substitute to a popular opinion of the genre of a song and not as good as frequency testing study.
3. I used accuracy of test to measure my results based on the test data.

## Experimental Results

My first attempt at classifying songs by genre included logistic regression and a test set that had more than 100 genres in it. This test provided low accuracy score due to the great amount of different genres present. I then turned to lowering the number of genres to 11 main genres as well as testing decision tree classifier as well. This greatly improved performance, but since there were multiple songs having the same attributes but different genre names, the accuracy was still not good enough. I then decided to match each individual song to one main genre and saw the accuracy go up another 10% to 43%. Looking for solutions I turned to hierarchical clustering shown in the graph below. The results from this cluster showed that my testing data was not well diverse therefore I couldn’t get the results any higher with the test data I had.



This project was not as successful as I would have liked it to be, but I learned a lot about classification and how important a good test set is. I believe this will work for some genres of songs that were better represented in the test data. If I had looked into my test data sooner, I could have made my own testing set through Spotify and try to gather a wider range of genres so each is better represented in the model.

# CONCLUSIONS

A song can be classified by its audio features. In my test, the accuracy level was 43% with a non-diverse group. In the future, a more diverse training set would server a great benefit to the classification models and their accuracy.

# REFERENCES (at least 3 references)

1. <https://code.djangoproject.com/wiki/Tutorials>
2. Ram, Jayen, and Daniel Salz. *Using Song Lyrics and Frequency to Predict Genre*. Stanford, 2017, cs229.stanford.edu/proj2017/final-reports/5241796.pdf.
3. <http://pandas.pydata.org/pandas-docs/stable/>
4. <https://spotipy.readthedocs.io/en/latest/>
5. <http://cse.msu.edu/~ptan/CSE482/>