Discover the Best Weekly

Catherine Bourque
College of William Mary
Williamsburg, Virginia
chbourque@email.wm.edu

Abstract

This paper will show that a multi-layer perceptron classification model can be used to predict songs that a user is more likely to enjoy from a specific playlist. The music streaming platform, Spotify, currently uses three different recommendation models to build user-specific playlists each week based on their music taste. The idea is to expose the user to new music based on songs they have previously liked. This supervised learning model takes a user's Spotify-recommended playlist, called "Discover Weekly", and classifies each song into one of two categories; likes or dislikes. The classification is based on the audio features that make up a track, with the goal of understanding which features are most important in determining whether or not a user will like a song. The model serves as an extra filtering layer to improve Spotify's current system of recommending new music.

Keywords: classification model, neural networks, perceptron, audio features

ACM Reference Format:
Catherine Bourque. 2020. Discover the Best Weekly. In *Proceedings of ACM Conference (Conference'17).* ACM, New York, NY, USA, 5 pages. https://doi.org/10.1145/nnnnnnn.nnnnnnn

1 Introduction

Spotify, a digital audio service provider, allows users to access millions of songs and to create, edit, and collaborate on playlists with other users. One of Spotify's most-loved features is their recommendation system that builds playlists uniquely suited to each user's music taste. Every Monday, a new playlist of 30 songs, called "Discover Weekly", is released to over 200 million users. This playlist serves as a method of recommending a set of songs that the user has not heard before that they are likely to enjoy based on their listening history.

Conference'17, July 2017, Washington, DC, USA

ACM ISBN 978-x-xxxx-xxxx-x/YY/MM...$15.00
https://doi.org/10.1145/nnnnnnn.nnnnnnn

The algorithm that determines which songs are selected for a user's Discover Weekly playlist is based on three recommendation techniques [7]. Collaborative filtering is used to collect and analyze a user's listening behavior. For around 200 million users and 40 million songs, Spotify stores a matrix in which each row represents a user and each column represents a song. If a user has heard a song, that song's column in their row is marked with a 1, otherwise it is marked with a 0. The algorithm then identifies groups of users with similar row vectors and generates Discover Weekly playlists by picking songs that a certain user has not heard, but that others in the group have heard [1]. There are several drawbacks to this method, the most important of which is that it does not utilize any information about the tracks themselves. Recommendations are made solely based on how popular a song is, making it difficult for less popular artists to gain traction, and leading to recommendations that may not fit a user's taste.

In order to combat this issue, Spotify also uses content-based filtering and audio feature analysis to make recommendations. Content-based filtering creates profiles for each user with descriptor tags of their consumption patterns, such as the genres they regularly listen to. Songs and artists have similar profiles, built from Natural Language Processing techniques that collect and analyze information from around the internet that describe the artist or the song. These descriptor tags on song and artist profiles are then matched to tags on user profiles to recommend new music. The major drawback of this method is that the NLP algorithms used to create tags for songs or artists are entirely based upon the information they collect from websites, such as music review articles or blogs. Again, the sounds of the tracks are not taken into consideration with this method, and online reviews can be subjective from author to author.

Audio feature analysis is essential in creating Discover Weekly playlists that are highly personal to a user's taste. Spotify extracts a number of audio features from tracks using convolutional neural networks. Employing four convolutional layers and three fully-connected layers, these neural networks produce a sonic profile, which is a collection of audio features that make up a song, such as tempo, key, and time signature. The sonic profile is then compared to other similar profiles in Spotify's database to recommend songs with similar audio features.

Although the combination of these techniques should theoretically produce excellent Discover Weekly recommendations, I have found that my recommendations are hit or miss from week to week. I typically listen to different genres for a week or two at a time and I have noticed that my recommendations often lag behind by several weeks. Another issue that I have with this system, as well as many of my friends, is that in attempting to achieve a high level of similarity between songs, Spotify often compromises the diversity of their recommendations. I have received a fair number of Discover Weekly playlists that contain songs that all sound very similar, and none of which I absolutely love. My idea for this project was to train a model to pick out songs from my Discover Weekly that I am more likely to enjoy, based on the audio features from a set of songs that I really love. I wanted to understand what audio features contribute most to music that I like and music that I do not like, in order to filter the best and worst recommendations from my Discover Weekly each week. The ultimate goal of this project is to minimize the amount of time spent listening to songs that I do not enjoy by identifying the best recommendations for my taste.

2 The Data Set

Acquiring a large dataset for this project was quite simple. I already had a collection of around 5000 songs saved, in addition to the millions of other songs at my disposal via the entire Spotify library. The most difficult part of this task was to find a way to categorize the data into a "likes" playlist and a "dislikes" playlist, so as to maximize the effectiveness of the classifier model.

My personal library is comprised of both songs that I love and listen to regularly, and songs that I heard once, saved, and never listened to again, or rarely listen to. The author of the project that I took inspiration from used his entire library of saved songs for his likes playlist and a collection of songs taken from friends that he judged to have "god-awful music tastes" for his dislikes playlist [4]. While I considered this idea, I thought a more effective route would be to split my saved songs into two categories; songs that I regularly enjoy and would like to find other songs with similar sounds, and songs that I liked enough to save but not enough to continually listen to. It would have been more simple to use the original author's method rather than combing through my entire library, however I felt that my method would create a model much more fine-tuned to my specific taste. My Discover Weekly recommendations are typically well-suited to my taste and rarely fall into the category of being "god-awful". I felt that a dislikes playlist of songs that I truly do not enjoy would be ineffective in trying to filter my Discover Weekly because I usually enjoy, to a certain degree, most or all of the songs recommended to me. The most logical course of action then, to fine-tune a playlist that

acousticness	danceability	energy	instrumentalness	key	liveness	loudness	mode	speechiness	tempo	time_signature	valence
0.04670	0.892	0.633	0.000189	8	0.1390	-7.471	1	0.0902	137.994	4	0.4590
0.21900	0.701	0.624	0.000000	10	0.1070	-11.115	1	0.3060	93.024	4	0.3830
0.03950	0.837	0.636	0.001250	1	0.3420	-7.643	1	0.0860	145.972	4	0.2740
0.23100	0.647	0.667	0.000000	5	0.1330	-5.563	1	0.3040	172.080	4	0.7040
0.34900	0.825	0.733	0.000000	1	0.1120	-5.625	0	0.0969	97.974	4	0.6200
0.15900	0.800	0.467	0.000000	0	0.6530	-9.974	1	0.4500	132.054	4	0.3070
0.31200	0.709	0.503	0.000000	7	0.2100	-5.762	1	0.5380	87.673	4	0.2600
0.03120	0.822	0.502	0.000887	7	0.1140	-7.380	1	0.1480	73.003	4	0.5250

Figure 1. Audio features of the data set.

was already decently good, was to fill the dislikes playlist with songs that I do not necessarily dislike, but would not listen to often, and to fill the likes playlist with songs that I love and would listen to at any time.

Both playlists that I created for the training stage contained 500 songs each, sorted from my saved library using the method described above. Both playlists covered a wide range of genres, including rock, funk, disco, soul, hip-hop, electronic, and indie. The test set comprised of 90 songs, in which I simply added my entire 30-song Discover Weekly playlist to this set over the course of 3 weeks. In order to validate the results, I created another likes playlist and a dislikes playlist, in which I categorized each song in the test set in the same manner as the training set.

3 Exploratory Data Analysis

Once the playlists were created, I used Spotify's API service to extract the audio features for each song (fig. 1) [3]. These features include acousticness, danceability, energy, instrumentalness, key, liveness, loudness, mode, spechiness, tempo, time signature, and valence. It also included a duration and popularity measure for each track, however I felt that because my library contains a wide variety of both long and short songs, and popular and underground artists, these features would be unimportant in deciding whether or not I would like a song. I wanted to focus solely on the audio features that make up a track, so I dropped these two features from the data set entirely after seeing how widely they varied.

In order to understand which features were most informative in predicting whether or not a song would fall into my likes category, I began by plotting the mean values (fig. 2) and the distributions (fig. 3) of each audio feature [2]. The slight distinctions between the distributions of danceability, energy, and loudness features, in particular, showed that I prefer tracks that are more suited for dancing, slower, less energetic tracks, and tracks with a lower overall loudness. The distribution of the valence feature was particularly informative, showing that I prefer more positive, upbeat tracks over more negative, sad tracks. Overall, this analysis told me that the distinction between tracks that I love and do not love is small and likely involves a combination of different features depending on the song.

4 Training

After seeing only minimal differences between all of the audio features, I decided it would be in the best interest of performance to train the model using all of the audio features shown in the graphs above. There were no features that stood out as being particularly good indicators of whether or not I would like a song, so I kept all of them, with the hope that this would yield more accurate results than only using a select few features.

I combined both the likes playlist and the dislikes playlist into a single set, assigning a 1 to my liked tracks, and a 0 to my disliked tracks. I then split the data into a training set and a testing set that comprised of 80% and 20% of the original set, respectively. I then used a multi-layer perceptron learning algorithm with a cross-entropy error function and backpropagation to train the model over 500 epochs [5]. I used 2 hidden layers for the network, with 12 nodes in the input layer, 2 nodes in the output layer, and 8 and 6 nodes in the hidden layers.

5 Testing

I tested the out-of-sample performance of the model on the playlist I created from my Discover Weekly recommendations. I verified these results using the likes and dislikes playlists that I separated each of these recommendations into. Again, I did this by assigning a 1 to songs from the likes playlist and a 0 to songs from the dislikes playlist on the entire set of recommendations.

6 Experimental Results

In order to evaluate the performance of the model, I used precision and accuracy metrics. Each song was identified as either a true positive or a true negative, i.e. correctly predicted to be a like or dislike, or a false positive or false negative, i.e. a true dislike incorrectly predicted to be a like or a true like incorrectly predicted to be a dislike. The precision measure defines the percentage of all positives that were actually correct, whereas the recall measure defines the percentage of actual positives that were correctly identified.

From the set of songs used for training, the model achieved average scores of 54% for both precision and recall. In other words, when the model predicts that a song falls into the likes category, it is right 54% of the time, and it correctly identifies 54% of all liked songs. Out of the 800 songs used for training, the model correctly predicted 173 songs to be true positives and 261 songs to be true negatives, and incorrectly predicted 225 songs to be false positives and 141 songs to be false negatives. The test set achieved average scores of 59% for both precision and recall. Out of these 200 songs, 46 were true positives, 71 were true negatives, 56 were false

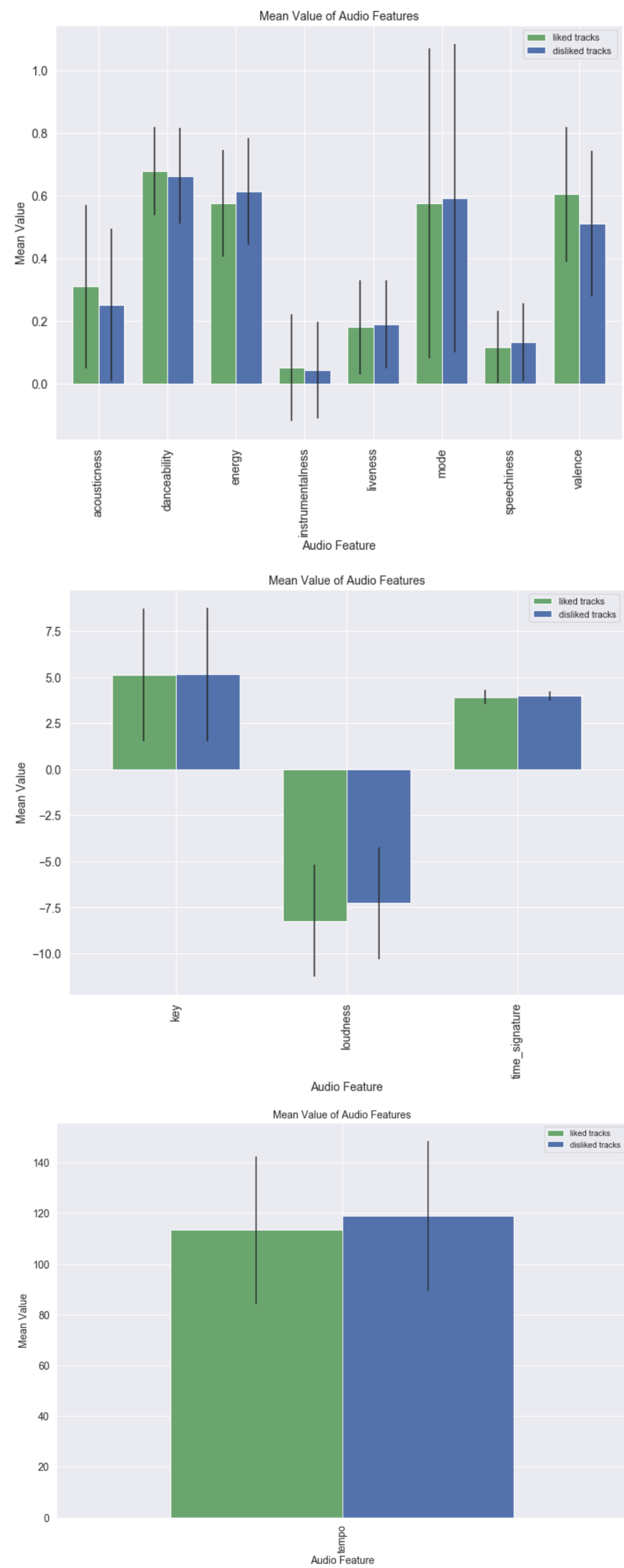

Figure 2. Mean Values of each audio feature for liked and disliked tracks.

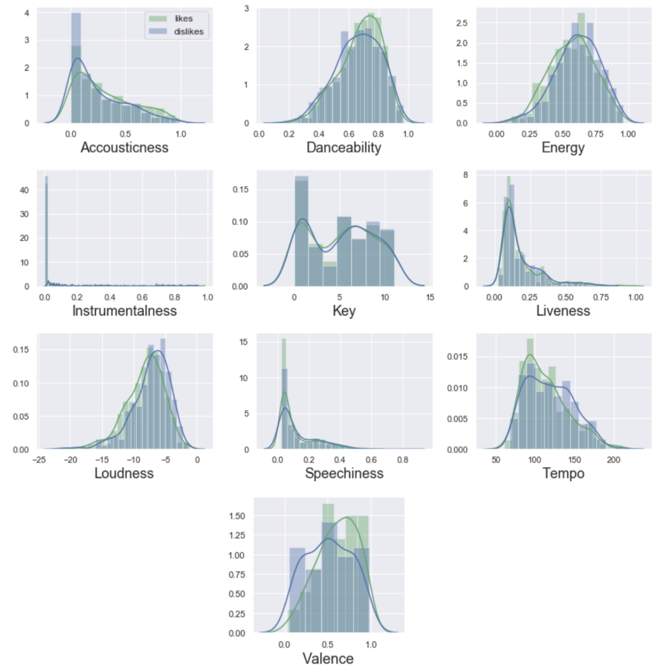

Figure 3. Distributions of each audio feature for liked and disliked tracks.

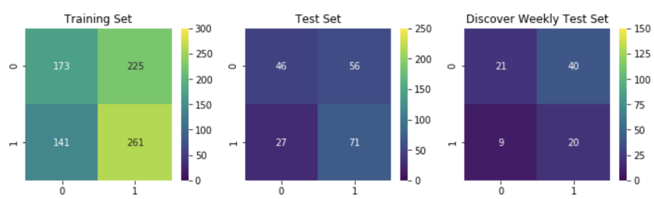

Figure 4. Confusion matrix visualization showing the number of songs in each category ((0,0) = true positive, (0,1) = false positive, (1,0) = false negative, (1,1) = true negative).

positives, and 27 were false negatives. The precision and recall suffered slightly when testing the model on the test set of my Discover Weekly songs, with a score of 52% for both precision and recall. From this set of 90 songs, the model correctly predicted 21 true positives, 20 true negatives, 40 false positives, and 9 false negatives. In order to visualize the results, I used a confusion matrix in which the top left box corresponds to true positives, the top right corresponds to false positives, the bottom left corresponds to false negatives, and the bottom right corresponds to true negatives (fig. 4) [6].

7 Future Research

After seeing the results of both the exploratory data analysis and the model's performance out of sample, it became apparent that building the training set was an extremely important step of this project. In order to get a better sense of the audio features that are common among my favorite songs, I would like to build a series of smaller playlists, each made up of songs I love that sound very similar. Analyzing the features of two separate playlists that sound very different could provide more insight into why I enjoy certain songs. The original project that I based this project on [4] used a logistic regression algorithm rather than a perceptron algorithm for training, which achieved precision and accuracy scores extremely similar to mine. I would like to explore whether or not my method of building the training data set, in combination with a logistic regression algorithm, could achieve better scores than the original project.

Another point of interest that I would like to look into is analyzing how covers of the same song across several genres differ in terms of audio features. While my data set did not have any duplicates of the exact same song by a single artist, it did include covers of the same song by several artists in different genres. It would be interesting to analyze which audio features are preserved and which are different between these covers and then to build a model that will predict whether or not I will like a cover, depending on the presence of a certain feature.

8 Conclusion

In this paper, I described a method of collecting a set of songs to train a multi-layer perceptron to predict whether or not I would like a song recommended to me. The perceptron was then used to filter my Discover Weekly playlist into two sets; songs that the model predicts that I will like and songs that the model predicts that I will not like. This model can be used to decrease the amount of time a user will spend combing through their recommended playlist each week by isolating a set of songs that they are more likely to enjoy based on their previous listening patterns.

References

[1] Umash A. Bhat. 2017. *Spotify's Discover Weekly: How machine learning finds your new music.* Retrieved November 30, 2020 from https://hackernoon.com/spotifys-discover-weekly-how-machine-learning-finds-your-new-music-19a41ab76efe

[2] Abigail Hipolito. 2017. *Spotify: Analyzing and Predicting Songs.* Retrieved November 30, 2020 from https://medium.com/mlreview/spotify-analyzing-and-predicting-songs-58827a0fa42b

[3] Sander Horpestad. 2020. *How to Extract Data using Spotify's API, Python and Spotipy.* Retrieved November 30, 2020 from https://morioh.com/p/31b8a607b2b0

[4] George McIntire. 2017. *A Machine Learning Deep Dive Into My Spotify Data.* Retrieved November 30, 2020 from https://opendatascience.com/a-machine-learning-deep-dive-into-my-spotify-data/

[5] Deepika Singh. 2019. *Machine Learning with Neural Networks Using scikit-learn.* Retrieved November 30, 2020 from https://www.pluralsight.com/guides/machine-learning-neural-networks-scikit-learn

[6] Dennis T. 2019. *Confusion Matrix Visualization.* Retrieved November 30, 2020 from https://medium.com/@dtuk81/confusion-matrix-visualization-fc31e3f30fea

[7] Valerio Velardo. 2019. *Spotify's Discover Weekly explained- Breaking from your music bubble or, maybe not?* Retrieved November 30,

2020 from https://medium.com/the-sound-of-ai/spotifys-discover-weekly-explained-breaking-from-your-music-bubble-or-maybe-not-b506da144123