Elements of Music: An Exploration into Music Affinities with Essence as Data

Introduction

Can we find elements of a song to determine its likability for a listener? Or put another way, can metrics derived from the music itself classify its affinity?

I sat around my firepit upstairs on the roof under a summer night's sky listening to music, much of which was suggested to me by a friend. Most of which I thoroughly enjoy. The human still beating the computer. I watched the dancing flames fueled by the propane and wondered. What is it about a song that allures me? Why is it that most of the time I can tell within a few seconds of listening whether I will like this ensemble of sounds? There's something in the music. Some element(s) towards which I am naturally drawn. Can we capture that? Can we deconstruct those aural tones into number where they can be measured and distinguished? Can we us that to build a computer model capable of suggesting likable music with a success rate as high as a human recommender?

I explored Spotify's API for features to use in an ML model or analysis back in 2023. I played around with their audio features and audio analysis APIs which have since been deprecated (as of November 27, 2024) with a warning on their legal terms under Section IV Restrictions, 2. General Restrictions, a. Misuse of the Spotify Platform, i. "Do not misuse the Spotify Platform, including by using the Spotify Platform or any Spotify Content to train a machine learning or AI model or otherwise ingesting Spotify Content into a machine learning or AI model;".

I found this a little funny. That note did not appear the first time around, though the features I did gather the first time around also did not prove to be what I was looking for. Things like acousticness, danceability, energy, instrumentalness, etc. from this API weren't able to differentiate music like I had hoped, so when I came back to explore this hypothesis I wasn't too bummed that these features were now locked away.

The other source of musical extraction into data that did prove more interesting the first time around was from librosa. This is where things got interesting for me. Things like Mel-frequency cepstral coefficients (MFCCs), mel-scaled spectrograms, and chromagrams emerged as infinite representations of music as number. Features I couldn't quite understand despite seeking further explanation. But a funny thing about machine learning is that it doesn't quite matter if the composer understands the numbers or the patterns of their construction. If the model picks up signal and the results can be replicated on examples never seen before by the trained model, there is little merit in comprehending *exactly* why.

I keep referencing “the first time around”, but that’s not what will be analyzed here. That has been referenced to provide some context for the inspiration of the second go-around, which will be the focus of this brief exposition. With motivation to return to the premise I sought to explore the elements of music at its essence which might differentiate a song as desirable for the listener, as opposed to methodologies like collaborative filtering or more baked up algorithms like Spotify’s content-based filtering. The latter of which seems to flirt with the elements of a song but upon further examination appear more to file tracks into pre-constructed bins that miss the essence. Spotify’s algorithms can be further explored in their own words (a detailed explanation remains elusive), or by others who have extensively written about their own understandings.

Exploration

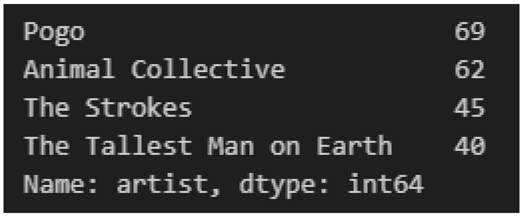
I began with a motive to first differentiate what in my mind is disparate music by using such elements as previewed above from librosa. To do this I used music files on my local computer as a launching point. I picked four bands in my library whom I consider quite different sounding who also have a large enough body of work to feed as much data to the model while the discography itself is fairly homogenous. My goal here is to train a model to be able to definitively tell me This is Artist X, this is Y, etc., as I can do almost immediately upon hearing a song from one of these artists. A model that can accomplish this level of distinction about music using only number is a signal that the parent objective is at least approachable. I know such research has been done at the genre level, but I believe for many audiophiles the genre abstraction is not a narrow enough field of view.

So I used the following four bands (and the albums that follow) as I have at my disposal at least four albums for each of them:

- Animal Collective
    - Fall Be Kind
    - Feels
    - Meeting of the Waters
    - Merriweather Post Pavillion
    - Painting With
    - Strawberry Jam
    - Sung Tongs
- The Tallest Man on Earth
    - The Wild Hunt
    - Shallow Grave
    - There’s No Leaving Now
    - Dark Bird Is Home
- The Strokes

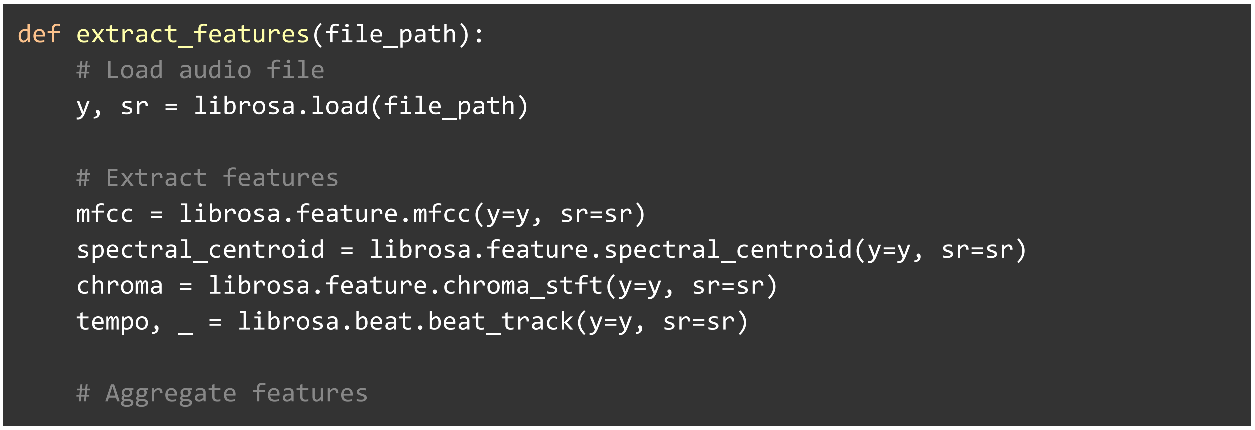
    - Angles
    - First Impressions of Earth
    - Is This It
    - Room on Fire
- Pogo[1]
    - 2016 miscellaneous tracks (not an album)
    - Kindred Shadows
    - Star Charts
    - Miscellaneous tracks (not an album)

The resulting track count per artist is as follows:



```
Pogo                        69
Animal Collective           62
The Strokes                 45
The Tallest Man on Earth    40
Name: artist, dtype: int64
```

With these tracks using librosa I extracted musical features. I used several user-defined helper functions to accomplish my specific goal and shape of the data here. Though the following functions were used, I have already thought about other ways of extracting the data here, specifically obtaining a wider range of the MFCCs, Spectral Centroids, and Chroma than just the means. Again, the objective of this initial exploration was to see if even the first-go around provided enough signal for artist-level distinction. The primary functions I used are outlined below for the trained eye to understand at a deeper level. If the code is illegible to the reader, do not fear, the point of the analysis will not be overshot.



```
def extract_features(file_path):
    # Load audio file
    y, sr = librosa.load(file_path)

    # Extract features
    mfcc = librosa.feature.mfcc(y=y, sr=sr)
    spectral_centroid = librosa.feature.spectral_centroid(y=y, sr=sr)
    chroma = librosa.feature.chroma_stft(y=y, sr=sr)
    tempo, _ = librosa.beat.beat_track(y=y, sr=sr)

    # Aggregate features
```

[1] *NB: My collection of Pogo's discography is a bit scattered but I don't think this detracts from the goal or will distract the model. His body of work is quite homogenous.*

```
    features = np.hstack((np.mean(mfcc, axis=1),
                          np.mean(spectral_centroid),
                          np.mean(chroma, axis=1),
                          tempo))
    return features
```

```
# Generate list of full file paths for music files in a given folder
def get_music_files(folder_path):
    music_extensions = ['.mp3', '.wav', '.flac', '.m4a', '.aac']
    music_files = [os.path.join(folder_path, file) for file in
os.listdir(folder_path)
                   if os.path.splitext(file)[1].lower() in music_extensions]
    return music_files
```

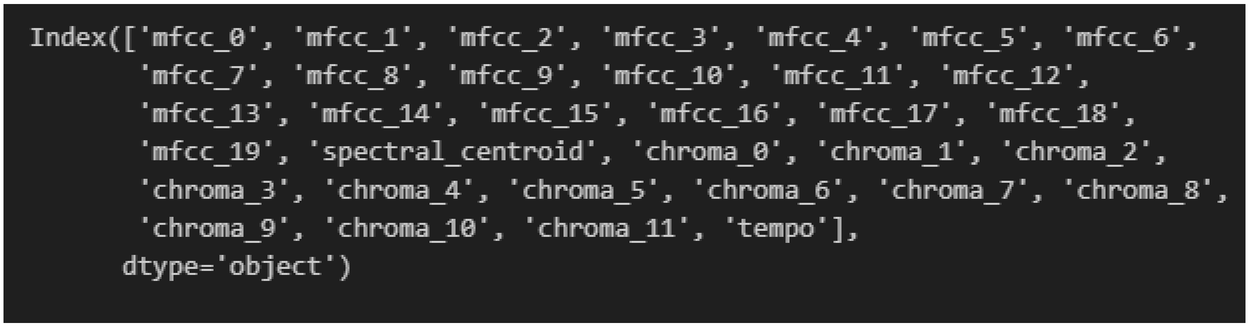
```
## DF fn
def create_music_dataframe(artist_album_dict):
    data = []
    for artist, albums in artist_album_dict.items():
        for album, folder_path in albums.items():
            files = get_music_files(folder_path)
            for file_path in files:
                data.append({
                    'artist': artist,
                    'album': album,
                    'filepath': file_path,
                    'track_name': os.path.basename(file_path)
                })
    return pd.DataFrame(data)
```

```
# Add librosa features to the df
def add_features_to_df(df):
    feature_names = ['mfcc_' + str(i) for i in range(20)] + ['spectral_centroid'] +
['chroma_' + str(i) for i in range(12)] + ['tempo']

    def extract_and_add_features(row):
        features = extract_features(row['filepath'])
        for i, feature in enumerate(features):
            row[feature_names[i]] = feature
        return row

    return df.apply(extract_and_add_features, axis=1)
```

To summarize the above, I target the desired music files on my computer then extract the music features from them and finally collect them all into a single dataset. Now that I have a collection of songs and data about those songs, I can see if a computer is able to differentiate the things that I can differentiate with my own ears.

Modeling

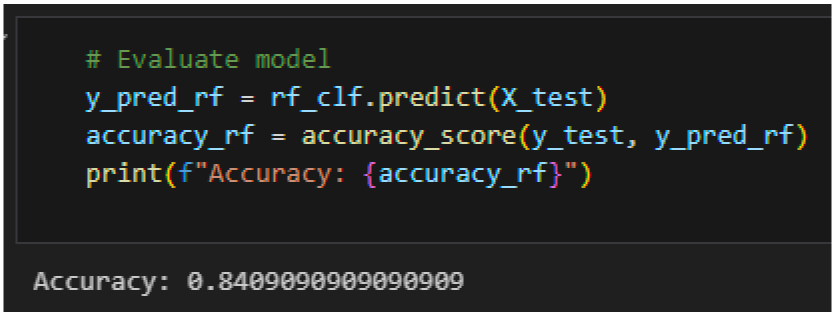
I wanted to start with very simple models to keep the point broad and to seek signal without overturning. I organized the data into features and the artist as the target variable. An output of the features, the same as previously discussed, can be seen below:



```
Index(['mfcc_0', 'mfcc_1', 'mfcc_2', 'mfcc_3', 'mfcc_4', 'mfcc_5', 'mfcc_6',
       'mfcc_7', 'mfcc_8', 'mfcc_9', 'mfcc_10', 'mfcc_11', 'mfcc_12',
       'mfcc_13', 'mfcc_14', 'mfcc_15', 'mfcc_16', 'mfcc_17', 'mfcc_18',
       'mfcc_19', 'spectral_centroid', 'chroma_0', 'chroma_1', 'chroma_2',
       'chroma_3', 'chroma_4', 'chroma_5', 'chroma_6', 'chroma_7', 'chroma_8',
       'chroma_9', 'chroma_10', 'chroma_11', 'tempo'],
      dtype='object')
```

I then randomly split the data into training and test sets, the former used for the model to learn which songs are associated with which artists, and the latter to be used for evaluation of the model with 20% of the data withheld. Again, I wanted this first run to be elementary while still being able to gauge efficacy so I did not introduce any cross validation or grid search for model hyper parameters. As such, the first model I fit to these data was a random forest with 100 estimators and a fixed random state. Being well aware of the many different evaluation metrics like precision, recall, and a confusion matrix, I started with a simple accuracy score[2] to see if this model picked up any signal whatsoever. Interestingly, the model highly exceeded randomness:

[2] The accuracy_score for a multi-class random forest classifier measures the proportion of correctly classified instances out of the total number of instances in the dataset. It is calculated by dividing the number of correct predictions by the total number of predictions made by the model.
For example, if a model correctly classifies 90 out of 100 samples, the accuracy would be 90%. This metric provides an overall estimate of the model's performance across all classes.

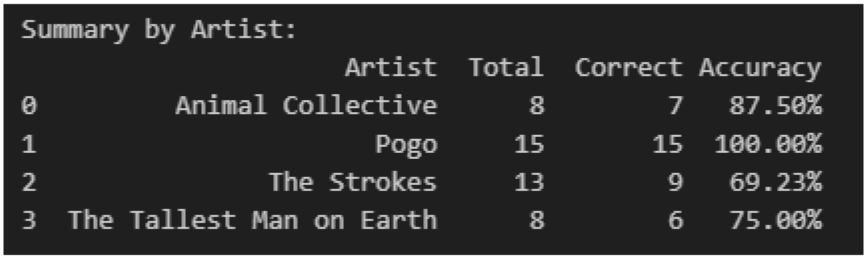


```python
# Evaluate model
y_pred_rf = rf_clf.predict(X_test)
accuracy_rf = accuracy_score(y_test, y_pred_rf)
print(f"Accuracy: {accuracy_rf}")
```

```
Accuracy: 0.8409090909090909
```

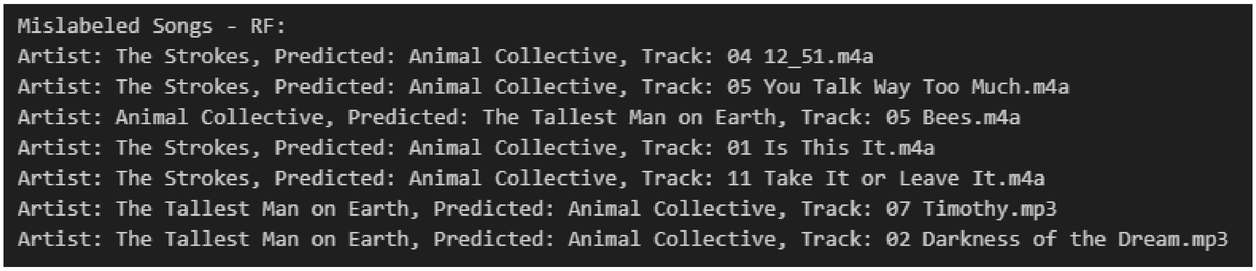
This result alone suggests there is plenty of signal for a model to distinguish artists, even with only 216 rows of data (and fewer than that to be trained on for this first model, per the test holdout set).

I was interested in the accuracy per artist, trying to piece together where it struggled and succeeded:



```
Summary by Artist:
                  Artist  Total  Correct Accuracy
0      Animal Collective      8        7   87.50%
1                   Pogo     15       15  100.00%
2            The Strokes     13        9   69.23%
3  The Tallest Man on Earth   8        6   75.00%
```

And an even deeper dive into the misses:



```
Mislabeled Songs - RF:
Artist: The Strokes, Predicted: Animal Collective, Track: 04 12_51.m4a
Artist: The Strokes, Predicted: Animal Collective, Track: 05 You Talk Way Too Much.m4a
Artist: Animal Collective, Predicted: The Tallest Man on Earth, Track: 05 Bees.m4a
Artist: The Strokes, Predicted: Animal Collective, Track: 01 Is This It.m4a
Artist: The Strokes, Predicted: Animal Collective, Track: 11 Take It or Leave It.m4a
Artist: The Tallest Man on Earth, Predicted: Animal Collective, Track: 07 Timothy.mp3
Artist: The Tallest Man on Earth, Predicted: Animal Collective, Track: 02 Darkness of the Dream.mp3
```

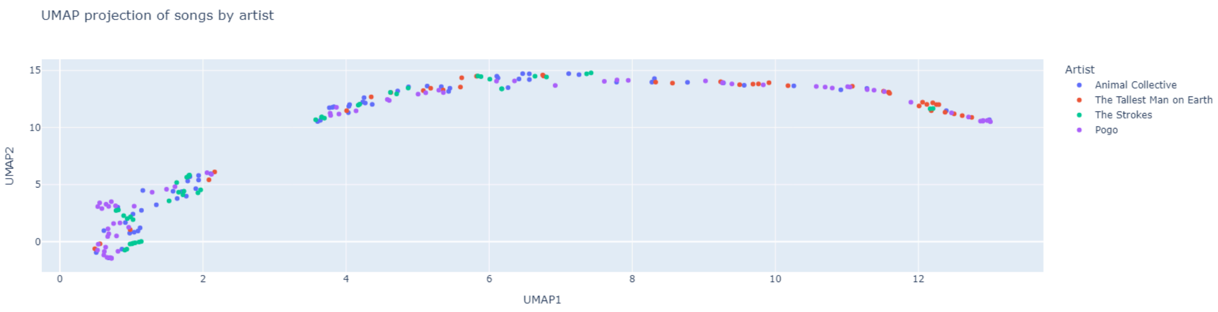
These results are highly curious to me. I personally would never mistake Bees by Animal Collective to be TMOE, or confuse those Strokes songs as AC, etc.. Prima facie this tells me that there are elements to these artists that the model is not seeing that I can clearly hear. However, it is a great first run through.

I also tried an un-tuned KNN but the results were not nearly as good, though still better than random (overall 45% accuracy). My summary written in a comment below the code for this notebook was as follows:

> *“Without any model tuning the RF does very well, 85%, without much data to work with. The KNN did not do very well, but better than a random guess.*
>
> *These results, to me, suggest there is signal in all of these librosa features to distinguish one artist from another.*
> *I believe that's enough to support the original hypothesis:*
> *- It's possible to build a recommender based on librosa extracted features rather than relying on Collaborative Based Filtering.*
> *- i.e. it's possible to suggest songs to a listener based on what they like without relying on what other people like.*
> *- - there's signal in the music itself for affinity”*

Plotting

I next wanted to plot these data to see if a feature reduction algorithm would be able to distinguish these artists on a two-dimensional plane (I used the same features and used UMAP as a function for dimensionality reduction). Unfortunately, it did not:



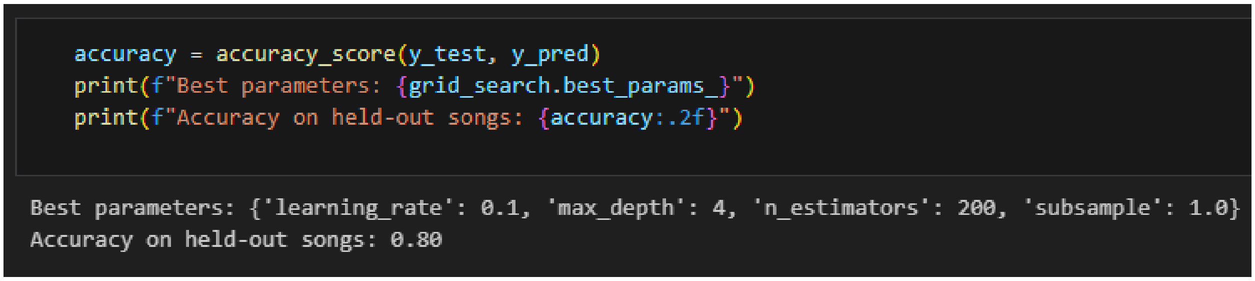
While the plot is difficult to examine here it does not require intense interrogation. There are small pockets where the artist’s music are collected together but plenty more examples of the entirety of the library of songs being scattered seemingly at random. I was hoping for something like a cluster of red in the upper right hand corner, blue in the lower left, etc. with maybe a few dots overlapped out of place. This attempt alongside the above modeling example suggests that a scrappy go at feature reduction to 2 dimensions is not sufficient for this analysis, and that’s okay.

Modeling v2 - This sample of music vs. 2024 Top 150 Country Playlist

I found enough signal in the investigation so far to try and take a step further: Could a machine learning algorithm learn the difference between music that I like and a top 150 country music playlist using only these librosa features, and only a sample of 216 songs of one class and 150 of the other?

I would also take a step further here and introduce XGBoost. If random forest classifier can find signal, I was curious how well an XGB would do. I also introduced a cross-validated (5 fold) grid search for model hyperparameters (though only one iteration of values) and again stayed broad by optimizing on accuracy. I used that best version of the model to fit on the training set and evaluate on the test set (10 randomly holdout songs).

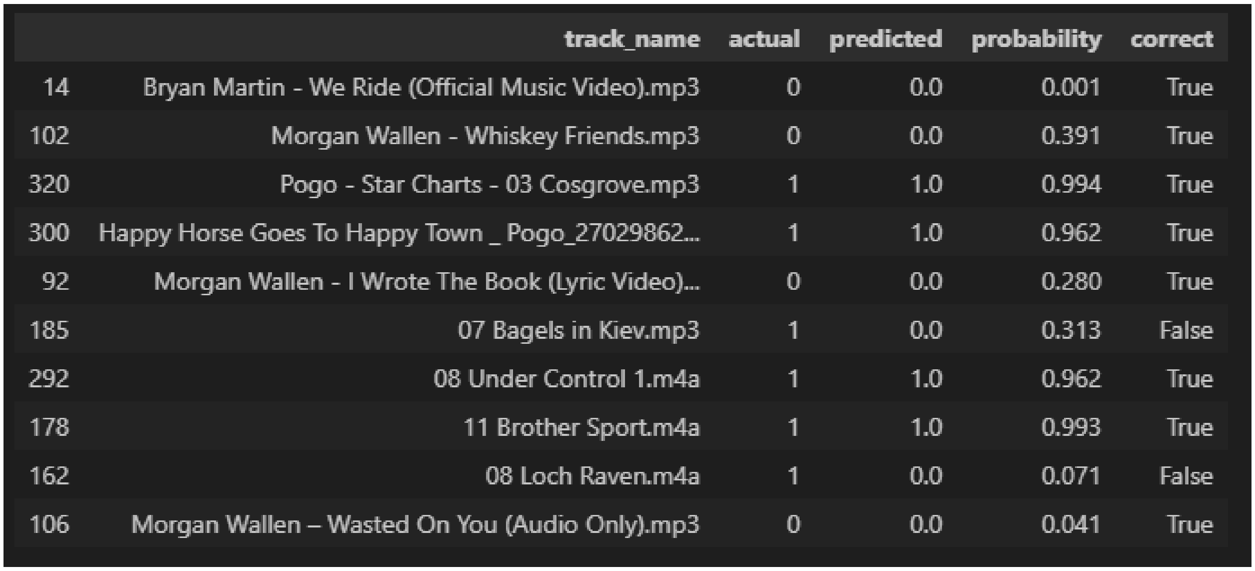
The result was an accuracy of 80%, or 8 out of the 10 songs were correctly labeled as from my music library or from the country music library. More testing is necessary here with different values being held out and of differing quantities, but again this is far exceeding a random guess.



```
accuracy = accuracy_score(y_test, y_pred)
print(f"Best parameters: {grid_search.best_params_}")
print(f"Accuracy on held-out songs: {accuracy:.2f}")
```

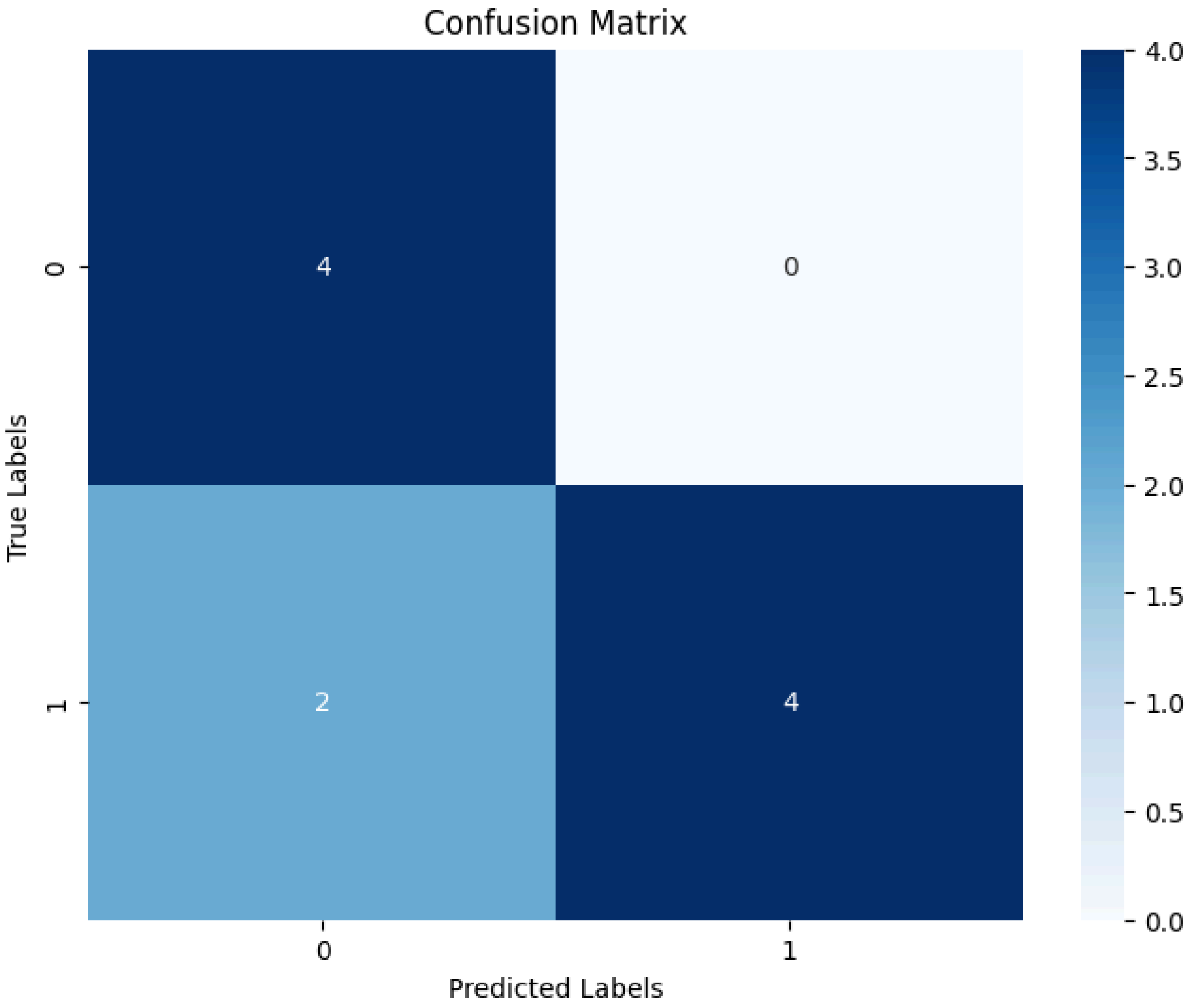
```
Best parameters: {'learning_rate': 0.1, 'max_depth': 4, 'n_estimators': 200, 'subsample': 1.0}
Accuracy on held-out songs: 0.80
```

I wanted to examine the probability outputs of the classes for the test set:



	track_name	actual	predicted	probability	correct
14	Bryan Martin - We Ride (Official Music Video).mp3	0	0.0	0.001	True
102	Morgan Wallen - Whiskey Friends.mp3	0	0.0	0.391	True
320	Pogo - Star Charts - 03 Cosgrove.mp3	1	1.0	0.994	True
300	Happy Horse Goes To Happy Town _ Pogo_27029862...	1	1.0	0.962	True
92	Morgan Wallen - I Wrote The Book (Lyric Video)...	0	0.0	0.280	True
185	07 Bagels in Kiev.mp3	1	0.0	0.313	False
292	08 Under Control 1.m4a	1	1.0	0.962	True
178	11 Brother Sport.m4a	1	1.0	0.993	True
162	08 Loch Raven.m4a	1	0.0	0.071	False
106	Morgan Wallen – Wasted On You (Audio Only).mp3	0	0.0	0.041	True

The misses are interesting to me. Loch Raven and Bagels in Kiev are so obviously not country music to my ears that I’m curious what features the model is missing to come up with that guess at those confidence levels. I also included a confusion matrix this time for easier interpretation of precision, recall, etc.:



My two high level conclusions from these results are as follows:

1. Some re-shuffling of the test set would be highly insightful.
2. There's something here.

My extrapolation or projection of application for this formulation is to train a model on every individual's music preferences. The positive class would be a song the listener enjoys, and the negative class the opposite. Neutrals could be included if so desired. After training on the listener's affinities the model could the infer on every song the listener has never heard before. Those songs with a high probability towards the positive class should be included for the listener's recommendation. Re-training could be done every month. A listener's affinities are malleable and perhaps a higher weight should be added towards what they're most recently enjoying.

Nevertheless I believe such a shallow and introductory investigation into the matter proves a computer capable of distinguishing a listener's affinities towards music using only the music. Perhaps not with 100% success, but I believe the current bar for a computer's ability to suggest likable music to be quite low.