CS 254 Machine Learning

Music Recommendation System (MRS) Final Report

John Urbani (UG), Alex Silence (UG), Robby Beattie (UG)

Introduction

We have created a Music Recommendation System (MRS) that categorizes music and determines if a new song is similar to others. This is a common problem in the music streaming industry, used to keep a user engaged and listening for a long period of time, as such many of these companies look for ways to find new songs for the user to listen to and guarantee that they will enjoy them. A common way to do this is to categorize music by genre using prelabeled tracks, determining which genre the user enjoys and giving them similar music. We approached this problem in a similar fashion, attempting to classify songs by genre using machine learning on several features contained in the songs themselves, such as tempo, acousticness, and loudness. By using machine learning as opposed to manually categorizing music, we can not only classify large numbers of songs by genre, but also more broadly fit the model towards a particular user's tastes. We are able to train our model on a large list of songs labelled by category then accurately determine the genre of a new song, however, there is room for error as many songs may be categorized in more than one genre or not fit a specific type of music.

Problem Definition and Algorithm

Task Definition

In our project we have taken data directly from Spotify using their web API. By doing this we are able to read in individual songs and playlists along with metadata features for each track. Our output should tell us with high accuracy which songs fit well together, such as being in the same genre of music. As we will be able to categorize any individual song, this could be used by many music streaming services to help their users listen to music they haven't heard before, keeping them engaged and coming back to their platform for much longer than a traditional recommendation system would.

Datasets

We created our own datasets through the Spotify web API, reading in hundreds of songs automatically using Python, then labelling songs by genre manually to check the accuracy of our model. The Spotify API is available for anyone to use but requires an account and verification from Spotify before use. We gathered our songs by looking at user-made playlists for different genres of music then labelling each song in this playlist as part of that category so we have less manual labelling to perform. To keep our dataset fair we tried to use a similar number of tracks for each genre so it will not overfit to any one type of music. We do not need any special hardware to process our data, however we do require an internet connection to access the web API when collecting data for the first time.

We have created a total of three different datasets, one containing 15 different genres of music to show the generality that can be achieved with machine learning, one with a few genres that are very similar sounding, with a large number of songs from each, and lastly one with music curated by a user alongside many songs that they do not enjoy to show possibilities outside of genre classification.

The Spotify API gives us the following metadata features for each track:
[beats_per_minute, acousticness, artist, danceability, duration_ms, energy, explicit, instrumentalness, key, liveness, loudness, valence, popularity]

Algorithm Definition

During our preliminary testing we tested models on two different datasets, the first containing 3 genres of music with 100 songs each and the second containing 8 genres with 85-100 songs each. We then ran these two datasets through several classifiers to determine which may give us a good baseline to work with. Some of the models tested included the Random Forest, AdaBoost, MLP, KNeighbors, and Decision Tree Classifiers. We found high success with Random Forest, AdaBoost, Quadratic Discriminant Analysis, and SVC, so we decided to use these classifiers to train our models to get a baseline before using neural networks.

During our testing we also determined that the ‘time_signature’ feature was detrimental to our training, especially after being regulated from 0 to 1, so this was removed before testing. Regularization was performed on all other features to aid the classifiers. The hyperparameters for each of the classifiers were also tweaked to better fit the dataset being tested, with Random Forest generally being kept at 100 trees.

Afterwards we tested all of our datasets on three different neural networks, network 1 included 1 hidden layer, with 256 nodes, and used an Adam optimizer, network 2 used 2 hidden layers, one with 256 nodes and the other with 512 nodes, with an Adam optimizer, and finally network 3 used 2 hidden layers, also with 256 and 512 nodes, with a SGD optimizer. We determined that these networks would suffice to give us a good range of results for our datasets by testing them rigorously with many different hyperparameters, some of which were changed to better fit the different datasets.

Each of these neural networks included only densely connected layers using relu activation, with the exception of the output layer which used softmax activation. We felt that testing different optimizers and the number of nonlinear layers would help us understand our results better.

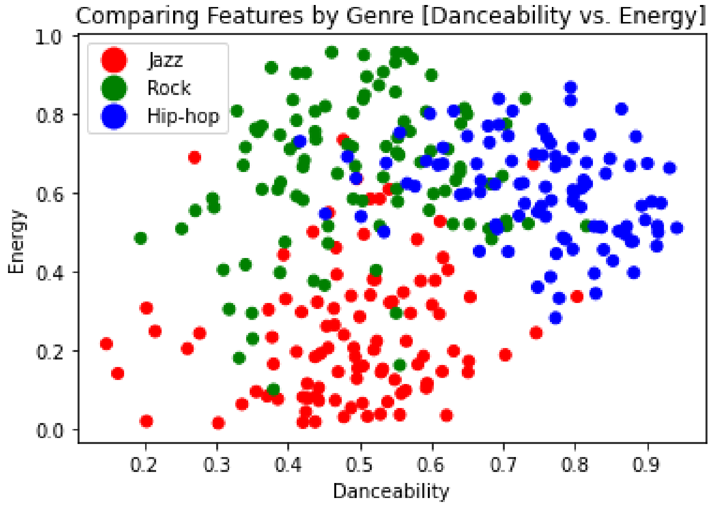
Experimental Evaluation

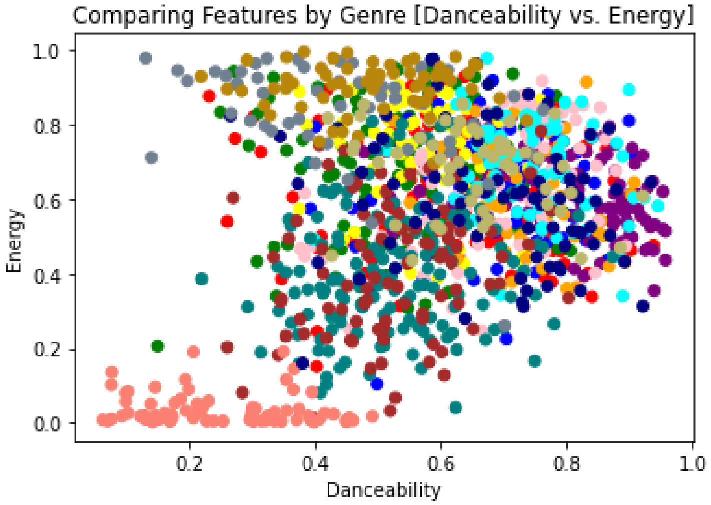
Methodology

We are determining the success of our model by looking at the overall accuracy of predictions and the precision for each genre of music classified. We expect to see high accuracy values as many of the chosen music genres are different enough from one another that many of their features will relate well within their group while being distinctly not a member of other groups. We believe that the accuracy of the predictions, with a large testing dataset, will provide ample evidence of the model’s performance, wanting to significantly beat a random guess, which we can see with the example of the 15 genre dataset being about 6.7%.

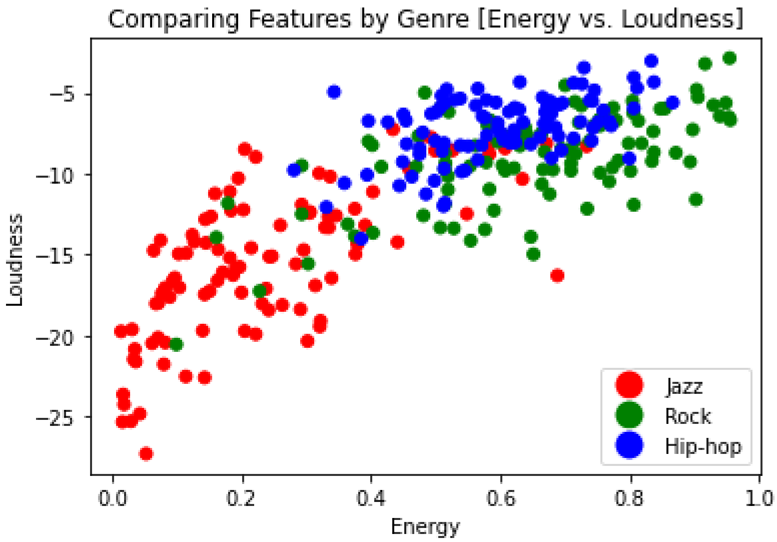
We can also use the individual precisions to help determine issues in our model, in particular areas where some genres of music may overlap, causing one group to accept songs from another unexpectedly. This issue will be especially prevalent in the 15 genre dataset as there are many genres there that sound similar, however, we hope to show that this can be avoided with our dataset made entirely of similar sounding genres to test if the differentiation is possible.

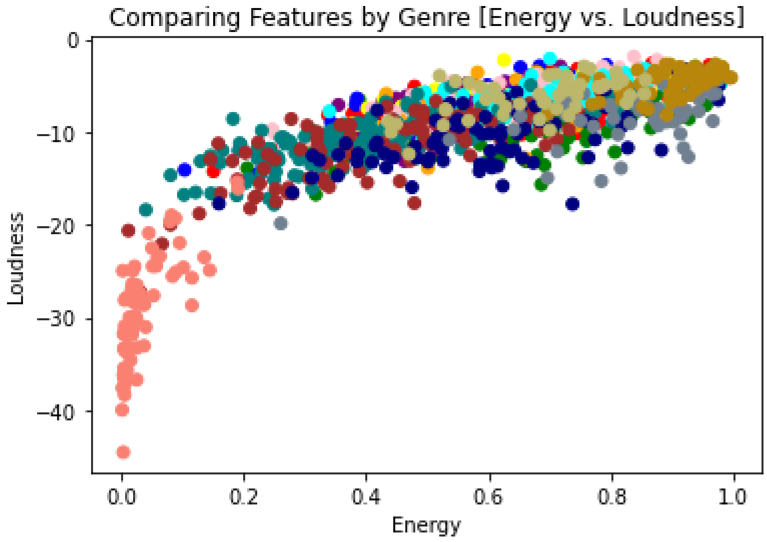
When comparing some features of our genres we can already see some key differences between them that will help our model to classify them. For instance we can look at some results found during our preliminary testing that show the clustering of genres very clearly, along with results taken from our 15 genre dataset for comparison:



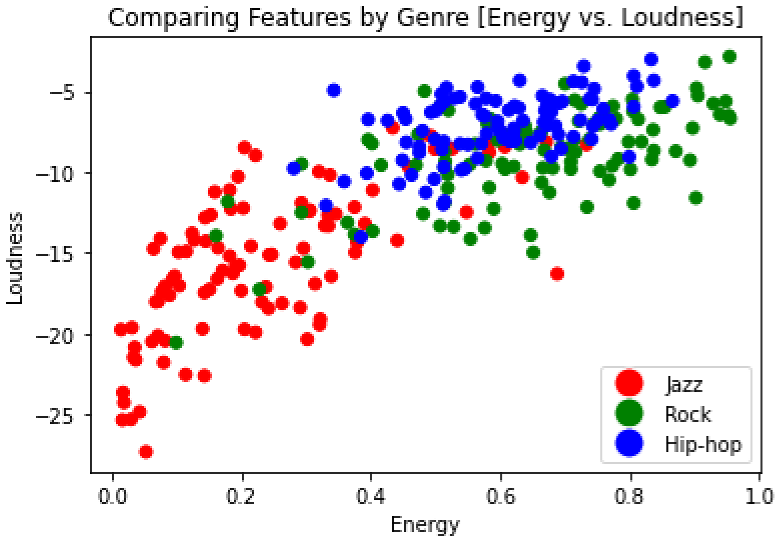


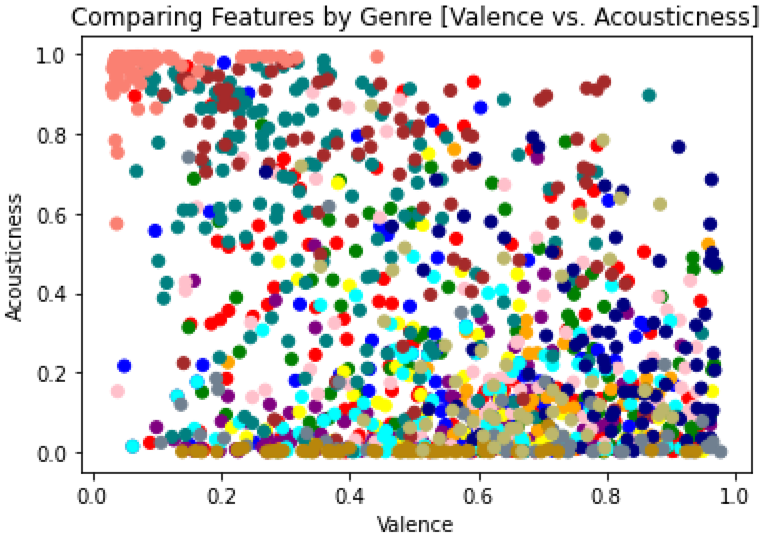
Here we can see obvious clustering between three different genres as we had suspected. Each of these clusters make sense when looking at the music itself, with our classic jazz songs being generally low energy songs and modern hip-hop being a type of high-energy dance music. The 15 genre dataset is harder to understand, but the general clustering of colors remains true, although there is clearly more overlap here, so it may not be as useful during classification.



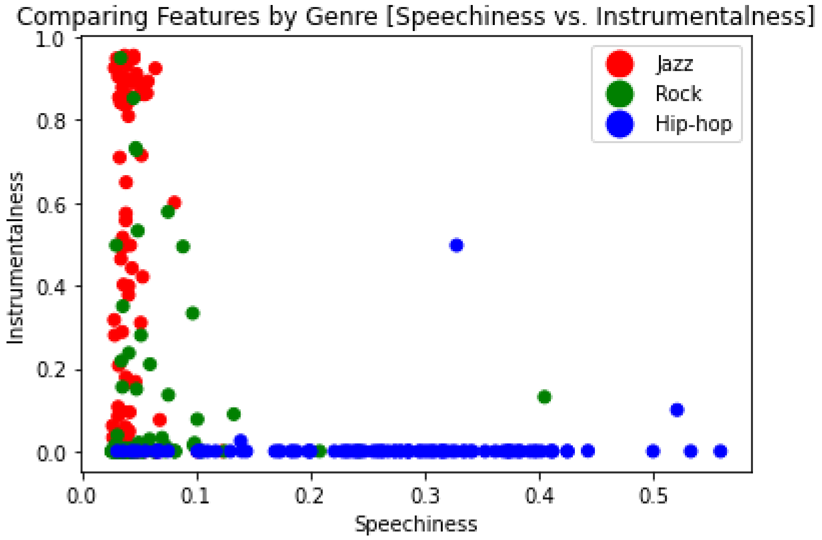


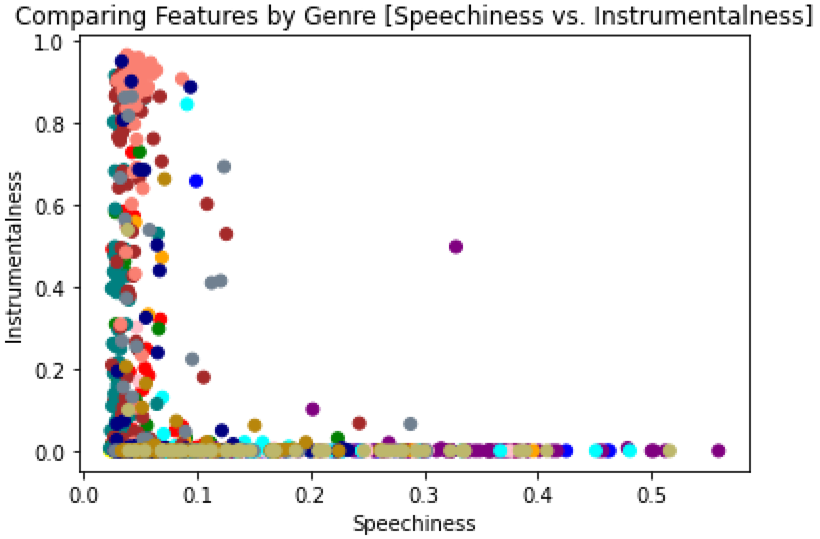
We see some clustering here as well along with some overlap between rock and hip-hop, which is expected as some high-energy rock music can sound very similar to hip-hop. Jazz is still very low energy in comparison and has a very low loudness. This loudness value is referring to the decibels Spotify had to change the master recording of a song to normalize the volumes across the platform. In this case it makes sense that our classic jazz music would have such a low loudness as many master recordings made at the time were using lower-quality recording equipment compared to modern standards, leading to an overall drop in loudness. As with the previous example we see that this holds true for the 15 genre dataset as well.





Valence here is a measurement of the cheeriness or happiness of a song. Rock is once again all over the board, although this was expected as it is a very broad genre that makes use of both acoustic and non-acoustic instruments and has both happy and not happy songs. Electronic instruments are not heavily used in classic jazz so the high acoustic value was expected. Similarly hip-hop makes almost no use of acoustic instruments, preferring modern electronic sounds instead. Hip-hop also makes very few sad songs, unlike jazz where this is much more common, explaining the difference in valence values. The 15 genre dataset is much less clear here, although we do still see some color clustering, especially near the very high and low ends of the acoustic feature.





Here we can see a very distinct difference between these music genres, that being jazz's preference for instruments over vocals, or speechiness. Rock is in a similar place here, although having some songs with higher vocal scores. Hip-hop however is very different, having almost no songs that don't contain some vocals.

Overall when looking at these comparisons, we hypothesize that these differences between genres will be enough to categorize individual songs into their respective genre. This is another good comparison for the 15 genre dataset as many genres are either lyric-heavy or instrument-heavy, giving us some clustering similar to what we found in our preliminary testing.

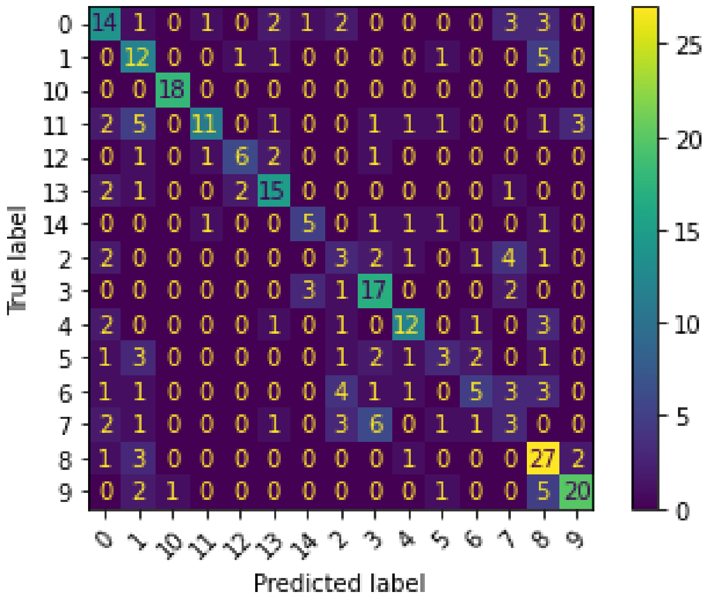
Results

General 15 Genre Dataset Results

This dataset consisted of the following genres, labeled by number:

0 : indie, 1 : rock, 2 : pop, 3 : hiphop
4 : country, 5 : r&b, 6 : latin, 7 : dance
8 : folk, 9 : jazz, 10 : classical, 11 : soul
12 : punk, 13 : metal, 14 : reggae

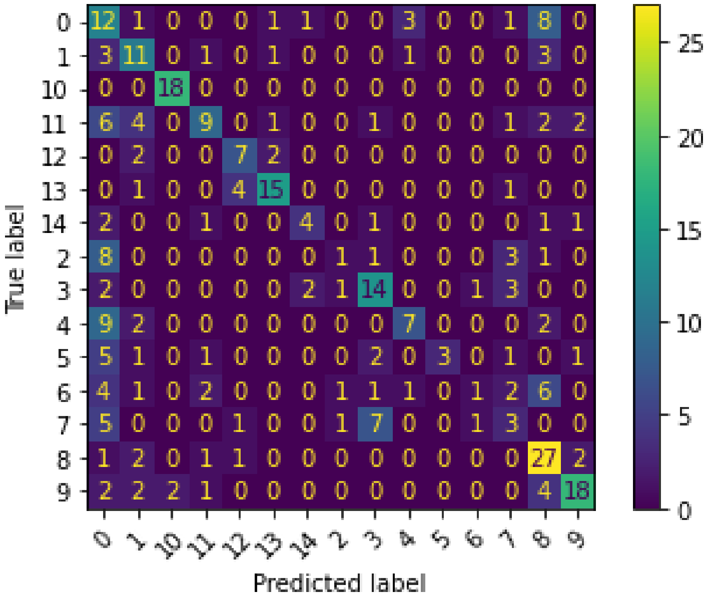
Random Forest Classification



	0	1	10	11	12	13	14	2	3	4	5	6	7	8	9
0	14	1	0	1	0	2	1	2	0	0	0	0	3	3	0
1	0	12	0	0	1	1	0	0	0	0	1	0	0	5	0
10	0	0	18	0	0	0	0	0	0	0	0	0	0	0	0
11	2	5	0	11	0	1	0	0	1	1	1	0	0	1	3
12	0	1	0	1	6	2	0	0	1	0	0	0	0	0	0
13	2	1	0	0	2	15	0	0	0	0	0	0	1	0	0
14	0	0	0	1	0	0	5	0	1	1	1	0	0	1	0
2	2	0	0	0	0	0	0	3	2	1	0	1	4	1	0
3	0	0	0	0	0	0	3	1	17	0	0	0	2	0	0
4	2	0	0	0	0	1	0	1	0	12	0	1	0	3	0
5	1	3	0	0	0	0	0	1	2	1	3	2	0	1	0
6	1	1	0	0	0	0	0	4	1	1	0	5	3	3	0
7	2	1	0	0	0	1	0	3	6	0	1	1	3	0	0
8	1	3	0	0	0	0	0	0	0	1	0	0	0	27	2
9	0	2	1	0	0	0	0	0	0	0	1	0	0	5	20

Our random forest classifier was highly successful with an accuracy of 56%, with the lowest precision being 19% for one genre, but having high precisions for most genres. This accuracy far exceeds our random-guessing accuracy of 6%, making it clear that these genres can be differentiated using their metadata, although we see some genres struggling, being mapped into the wrong genre. This is an issue that we continued to see with all of our classifiers for this dataset and was expected as many songs would realistically fit into multiple genres and may even give humans trouble deciding where they should be grouped. Overall we expect some songs to fall very close to the edge of multiple classes as they may share many features with songs from many genres.

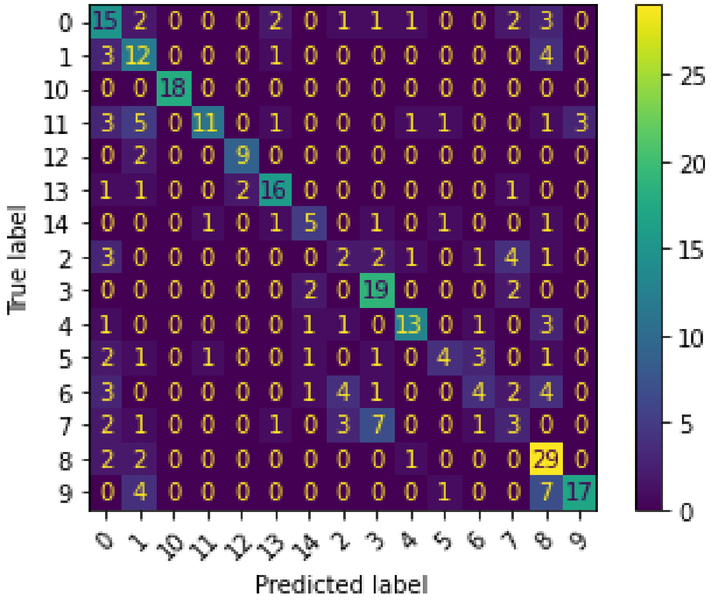
SVC Classification



	0	1	10	11	12	13	14	2	3	4	5	6	7	8	9
0	12	1	0	0	0	1	1	0	0	3	0	0	1	8	0
1	3	11	0	1	0	1	0	0	0	1	0	0	0	3	0
10	0	0	18	0	0	0	0	0	0	0	0	0	0	0	0
11	6	4	0	9	0	1	0	0	1	0	0	0	1	2	2
12	0	2	0	0	7	2	0	0	0	0	0	0	0	0	0
13	0	1	0	0	4	15	0	0	0	0	0	0	1	0	0
14	2	0	0	1	0	0	4	0	1	0	0	0	0	1	1
2	8	0	0	0	0	0	0	1	1	0	0	0	3	1	0
3	2	0	0	0	0	0	2	1	14	0	0	1	3	0	0
4	9	2	0	0	0	0	0	0	0	7	0	0	0	2	0
5	5	1	0	1	0	0	0	0	2	0	3	0	1	0	1
6	4	1	0	2	0	0	0	1	1	1	0	1	2	6	0
7	5	0	0	0	1	0	0	1	7	0	0	1	3	0	0
8	1	2	0	1	1	0	0	0	0	0	0	0	0	27	2
9	2	2	2	1	0	0	0	0	0	0	0	0	0	4	18

SVC classification appears to run into the same issues as the random forest, however, it has less success, only giving us an accuracy of 53%. While this is still highly successful, we see about the same precision levels, with no class having a precision below 20% but only a few classes above 60% precision.

Adaboost, using Random Forest base estimator

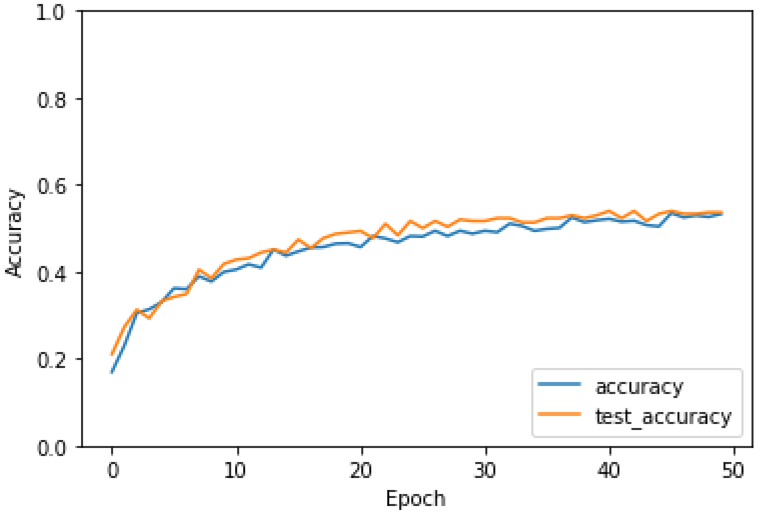


	0	1	10	11	12	13	14	2	3	4	5	6	7	8	9
0	15	2	0	0	0	2	0	1	1	1	0	0	2	3	0
1	3	12	0	0	0	1	0	0	0	0	0	0	0	4	0
10	0	0	18	0	0	0	0	0	0	0	0	0	0	0	0
11	3	5	0	11	0	1	0	0	0	1	1	0	0	1	3
12	0	2	0	0	9	0	0	0	0	0	0	0	0	0	0
13	1	1	0	0	2	16	0	0	0	0	0	0	1	0	0
14	0	0	0	1	0	1	5	0	1	0	1	0	0	1	0
2	3	0	0	0	0	0	0	2	2	1	0	1	4	1	0
3	0	0	0	0	0	0	2	0	19	0	0	0	2	0	0
4	1	0	0	0	0	0	1	1	0	13	0	1	0	3	0
5	2	1	0	1	0	0	1	0	1	0	4	3	0	1	0
6	3	0	0	0	0	0	1	4	1	0	0	4	2	4	0
7	2	1	0	0	0	1	0	3	7	0	0	1	3	0	0
8	2	2	0	0	0	0	0	0	0	1	0	0	0	29	0
9	0	4	0	0	0	0	0	0	0	0	1	0	0	7	17

As expected, Adaboost improved on our original random forest model, bringing us from 56% accuracy all the way to 60% accuracy. While this classifier still suffers from many of the same problems as the others, this was our most successful model for this dataset.

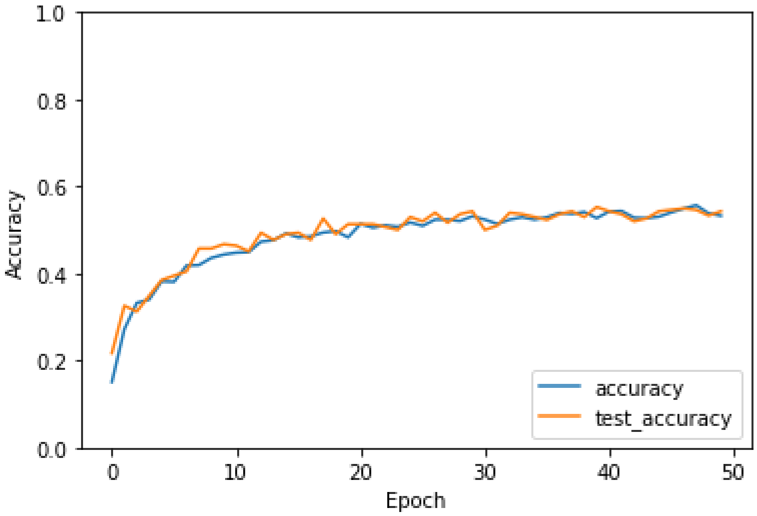
Quadratic Discriminant Analysis

1 Hidden Layer, Adam Optimizer



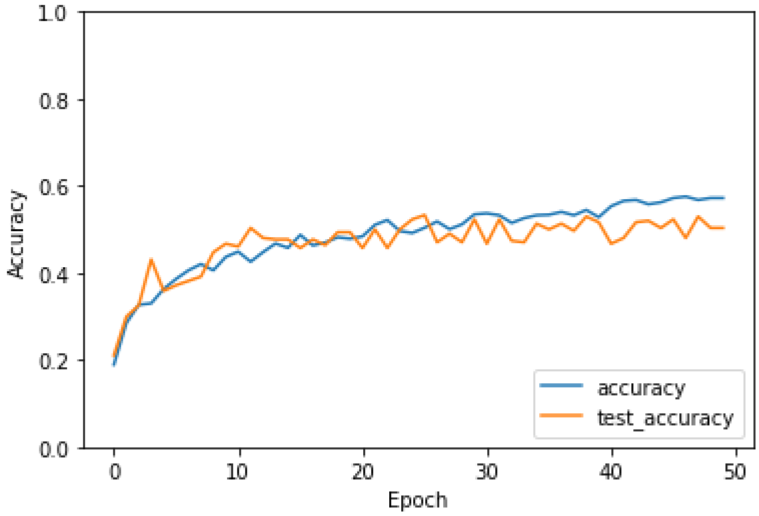
Moving to our neural networks, our 1 hidden layer model showed results very similar to our traditional classifiers, giving us an accuracy of 53.6% on our testing data after training. We can see the accuracy of the training and testing sets are very similar throughout the process so we did not experience much in terms of overfitting, but we do not expect to see significantly improved results by training the model for more time. The Adam optimizer used for this and the following model are using a learning rate of 0.00025 and a decay of 0.95. As mentioned previously this model's hidden layer has 256 nodes. We also used a batch size of 16 with 50 epochs for these three networks.

2 Hidden Layers, Adam Optimizer



Our 2 hidden layer model came out almost identically to the 1 layer model, although they both used nonlinear activation functions. The final accuracy for this model on our testing set was 54.3%, a slight improvement over the previous model, although still not an improvement over the Adaboost classifier. As mentioned previously this model has one hidden layer with 256 nodes and a second with 512 nodes.

2 Hidden Layers, SGD Optimizer



In this final neural network we tried to use a different optimizer, however, this appears to have only hurt our results, giving us a testing accuracy of only 50.3%, with clear differences between the training and testing accuracy as the model continues to train. As such we decided to only use our 2 hidden layer Adam optimizer model for our other datasets as it was the most successful of the networks and did not have this accuracy issue.

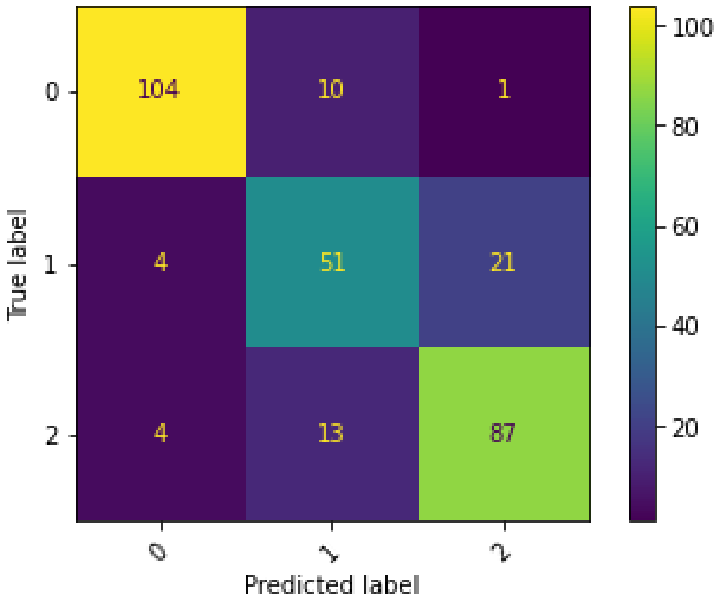
Similar 3 Genre Dataset Results

This dataset consisted of the following genres, labeled by number:

0 : jazz, 1 : blues, 2 : soul

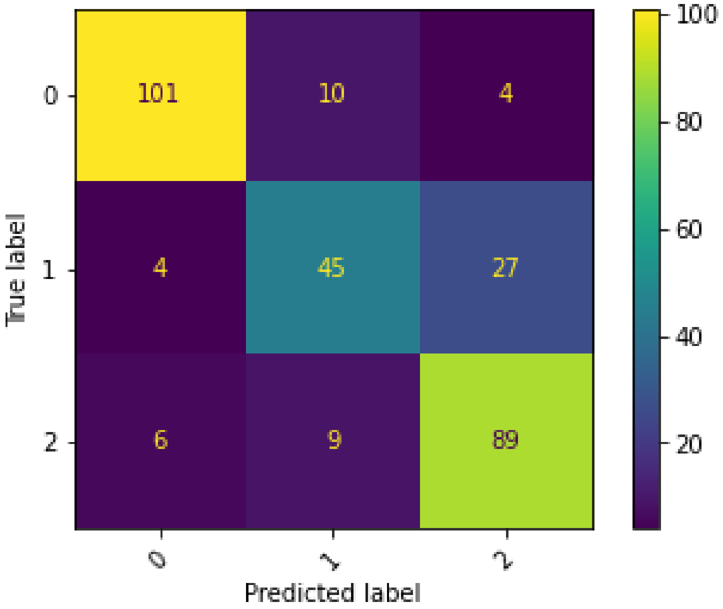
All of these genres share many traits with each other, including instruments used, amount of vocals, and energy of the songs. As such we tried to see if our model could differentiate between these very similar music genres if given enough data.

Random Forest Classification



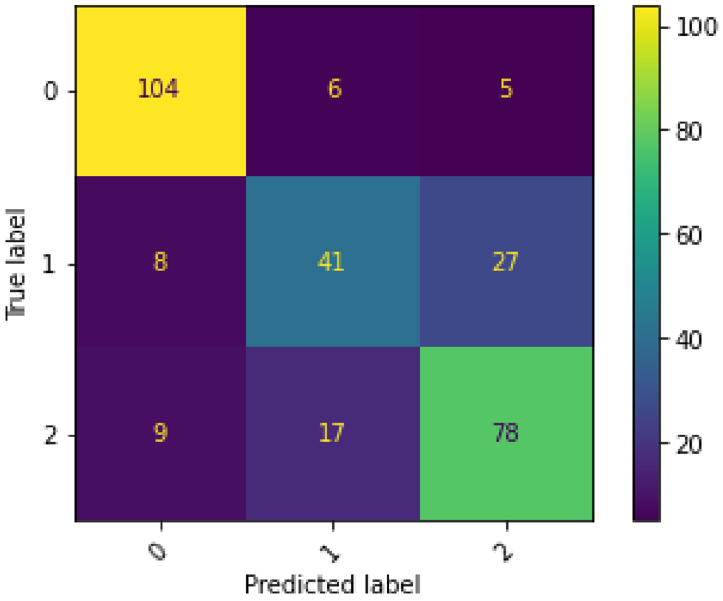
In this dataset we hoped to significantly beat the random guess accuracy of 33%. In our first test we tried a random forest classifier with 100 trees, giving us an accuracy of 82%, which shows that we can differentiate between similar genres as long as we have enough data to work with, which is not shown as well in the more general 15 genre dataset. Our precisions were all over 73%, but we can still see some struggling between the blues and soul genres (labels 1 and 2) which may be caused by a bias in our training data that included more jazz songs (label 0) than the other two.

SVC Classification



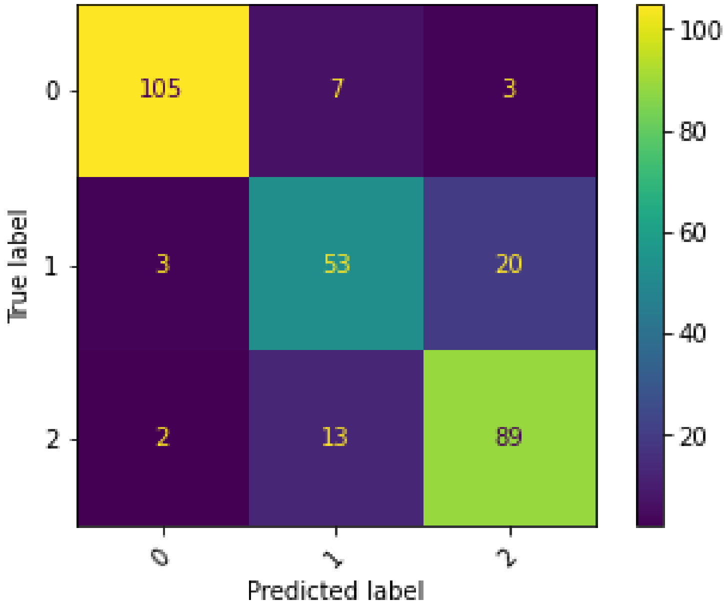
Similar to our previous dataset, our SVC is less successful than our random forest, only giving us an accuracy of 78%. We can see many more mistakes between labels 1 and 2 as well, with the lowest precision being 70% for label 1.

Quadratic Discriminant Analysis



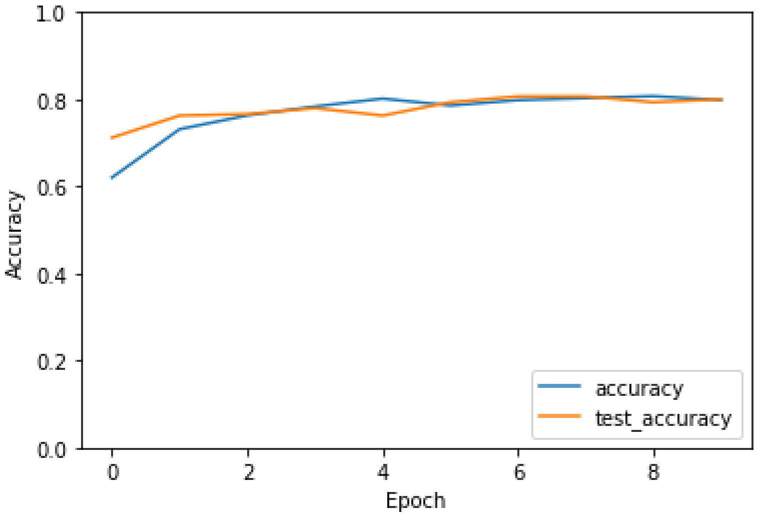
Quadratic discriminant analysis suffers even more with the similar features found between these genres, giving us an accuracy of 74% and a lowest precision of 64% for label 1.

Adaboost, using Random Forest base estimator



Adaboost found very good success here, as it did in the previous dataset, improving our accuracy from 82% to 84%. This increase was not very drastic and we did not see a large improvement between labels 1 and 2.

2 Hidden Layers, Adam Optimizer

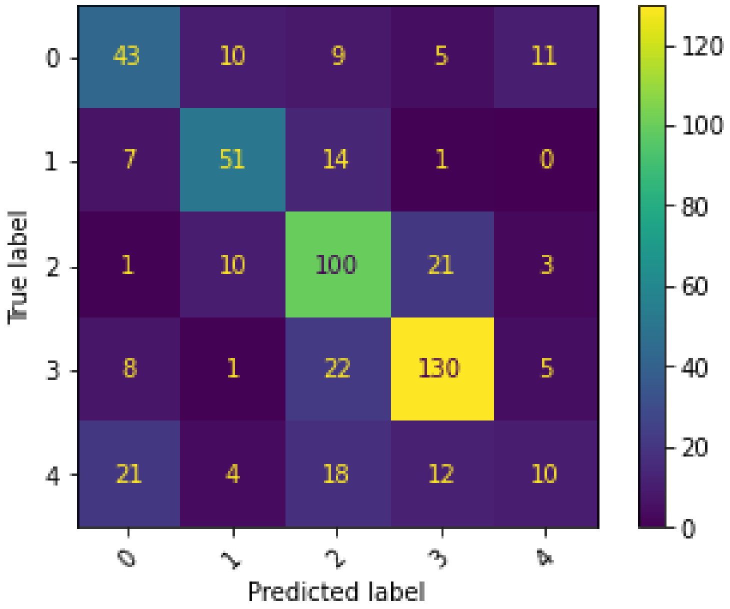


Our best neural network, running through 10 epochs with a batch size of 16, a learning rate of 0.00025, and a decay of 0.95, only ever achieved a test accuracy of 80%. While this is a good result, we once again find better results in our Adaboost models. We believe that this may be due to the simplicity of our datasets, only comparing 13 features, all of which are flat, meaning we likely do not require the complexity offered by a neural network.

Personal Playlist (non-genre specific) Dataset Results

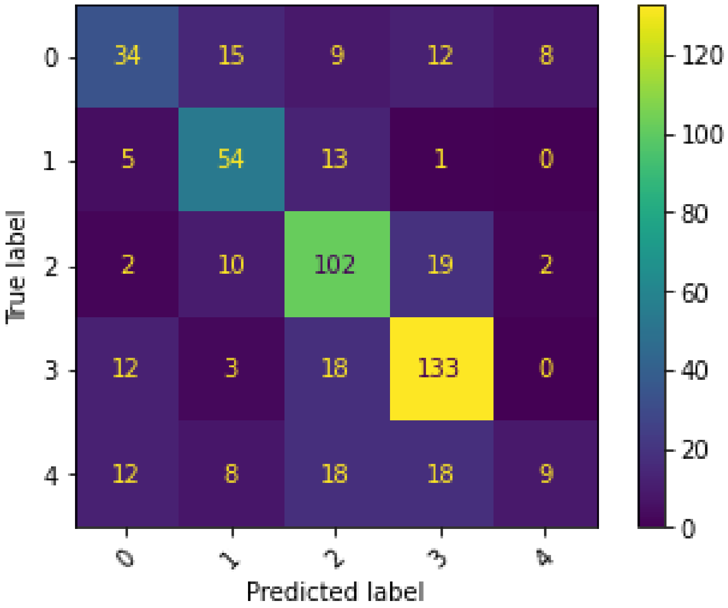
This dataset consisted of several large genre-diverse user-curated playlists. As many of these playlists contain songs from the same genres, we hope to find differences in each user’s music preferences that helps to differentiate these playlists The dataset contains five user-created playlists, labeled by number.

Random Forest Classification



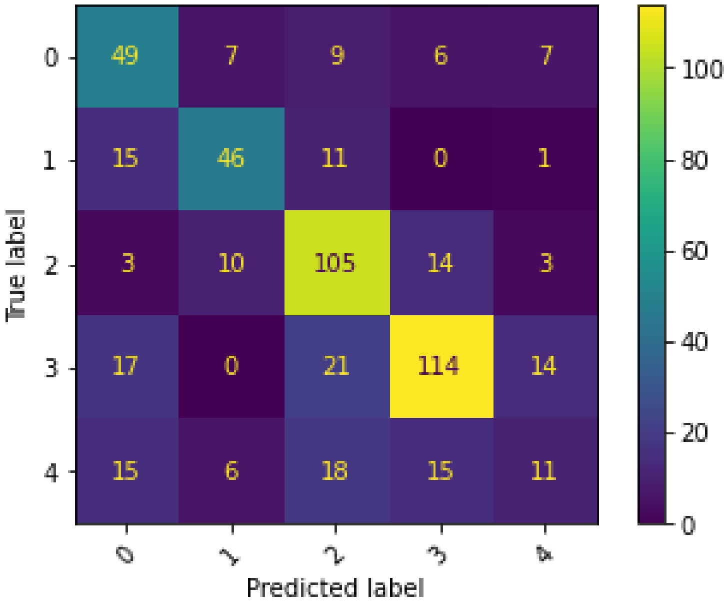
With five different playlists we had hoped to beat a random guess accuracy of 20%. As we can see our random forest classifier easily beats this with an accuracy of 59% and fairly high precisions with the exception of playlist 4. Overall we are happy with these results and expected higher amounts of overlap between playlists, but only playlist 4 had significant trouble, likely caused by a low amount of training data. This shows us that we will be able to classify music that is not obviously similar to each other, with these playlists having a wide variety of genres inside of them.

SVC Classification



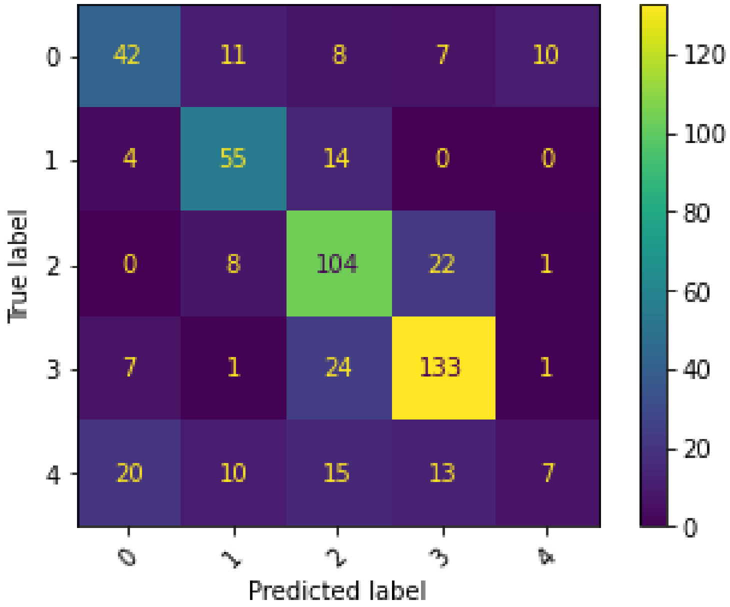
Our SVC gave us a similar accuracy of 59%, although with slightly lower precisions overall, which was expected based on the results of the previous datasets.

Quadratic Discriminant Analysis



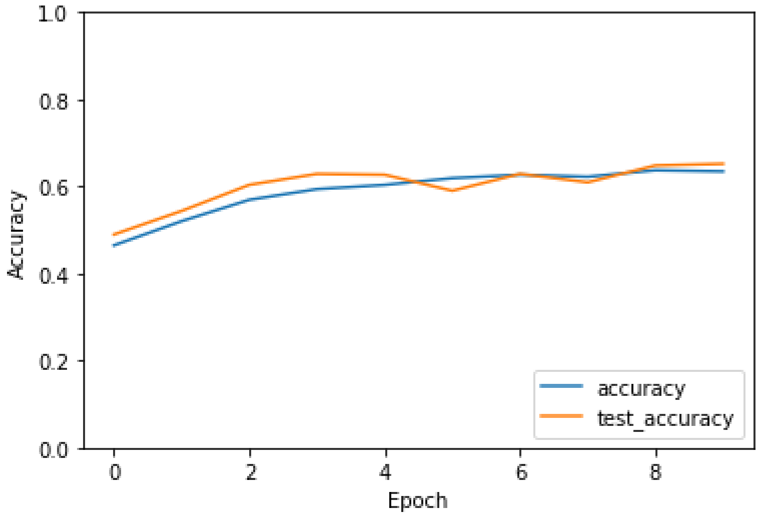
Similar to the previous datasets, quadratic discrimination struggled to differentiate the songs compared to the other models, giving us only 57% accuracy. However, the difference between this and other models is not very large.

Adaboost, using Random Forest base estimator



Once again Adaboost gave us the highest accuracy among traditional classifiers, reaching 60%. This improvement is once again very small, but will make a good comparison for our neural network.

2 Hidden Layers, Adam Optimizer



In this case our neural network gave us an accuracy of 65.2%. This was a huge improvement compared to the traditional models and is the first time we've seen the neural network surpassing Adaboost. We believe this is due to the added complexity of this dataset as there are fewer clusters between the features of each playlist due to their genre diversity, giving the neural network an edge as it could better find connections that the traditional classifiers may have missed.

Discussion

Our hypothesis that groups of music must have some features that keep them distinct from other groups is certainly true given our results, being able to categorize several genres and non-genre specific playlists with our models, having accuracies significantly higher than random guesses. We have discovered that the two best classification techniques for this problem would be either the Adaboost classifier or creating a deep neural network, giving very similar results, with our Adaboost models being superior for datasets that classify by genre and our neural network being superior for the more complex problem that is classifying non-genre specific songs into groups.

Overall we were happy with these results, however, we believe some of the models could be improved further if given larger and more evenly distributed training sets, with some bias being shown, especially in our traditional classifiers for our non-genre specific dataset.

Related Work

https://developer.spotify.com/documentation/web-api/reference/#endpoint-get-recommendations

Spotify itself has a recommendation system for a user, however they do not explain their exact methods. This is due to the fact that there is still ongoing research into designing the best algorithms to find users new music. What we do know is that spotify has more of a bias towards recommending similar artists and songs for the same user. Our approach is more agnostic towards the user, as it trains off of just songs from specific playlists we give it. This means that the recommendations that our program will give a user will be more generic and not based off of a specific user.

https://arxiv.org/pdf/1601.01892.pdf
There is also work to create song recommendation systems based on entirely different parameters. This paper describes a way to recommend songs using a matrix factorization and graph total variation. The paper proposes a way to find similar songs based on more features than we use in specific; it encodes playlist proximity information, and social features such as artist popularity, and how recently the artist has been popular.

Most music recommendation systems use similar methods and similar high level features for the music. Features such as acousticness and valence are pretty standardized forms of comparing songs. However the bulk of the other recommendation systems use more information on artist and song title. Conversely we have no information on artists, we believe that it will encourage our system to recommend a wider variety of songs and artists. We will likely find songs by the same artist will be classified similarly, but it is possible that our system will be less biased towards recommending the same artist repeatedly.

Code and Datasets

https://github.com/JohnUrbani/SpotifyMusicRecommender

Our code and datasets are available online on Github, although our Python scripts will not run without a private key provided by Spotify. As we already have our datasets created, they can be accessed and used without the private key, with all of our training and visualizations made in Jupyter Notebook files. Our training is laid out similarly to our homeworks, however we make an attempt to train several models with different methods, including classifiers not used in class and several neural networks, each displaying its accuracy and other statistics for each dataset.

Our process starts with gathering several Spotify playlists containing songs of a specific genre and adding their ID along with the name of the genre to our get_data_and_label.py script. This will gather the metadata for every song in the given playlists and add them, along with their target genre, to data_categorized.csv. In our Jupyter Notebook we finish cleaning the data, removing excess features that will not be used during training, such as the ID of the song or the name of the artist. To make this process simpler for collecting 3 different datasets we have separate python scripts and csv files for each.

Conclusion

We have found that it is possible to classify music, both by genre and other non-genre specific groupings, based purely on metadata taken from each song. This is an incredibly useful discovery that could be used by many companies to determine the music tastes of individual users and recommend music to keep users engaged with very high success.

We were interested to find the lack of improvement when using deep neural networks for our two genre datasets, instead finding better results with our Adaboost model. This was interesting as we had originally assumed that neural networks would easily outcompete traditional classifiers, however, these two datasets had obvious clusters in their features that aided the traditional models more than the neural networks which did not train for an incredibly high number of epochs. We saw also that the neural network was able to handle the more complex user-curated dataset much better than the Adaboost classifier was, which may come from the lack of obvious clusters found in the data, with the neural network having been able to find the connections between features that weren't obvious to us.

Overall this experiment was highly successful and we have shown that our hypothesis was correct, that music can be classified using metadata instead of using traditional music analysis methods while retaining very high accuracies.

Bibliography

Benzi, Kirell, et al. “Song Recommendation with Non-Negative Matrix Factorization and Graph Total Variation.” *2016 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2016, doi:10.1109/icassp.2016.7472115.

Spotify Web API, developer.spotify.com/documentation/web-api/reference/#endpoint-get-recommendations