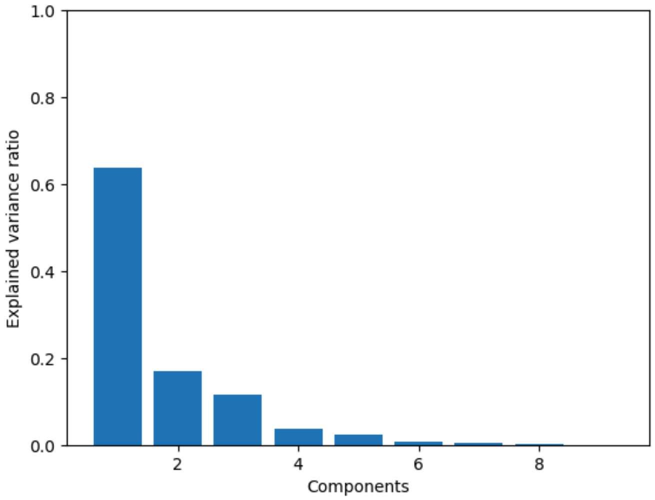
Data Cleaning:

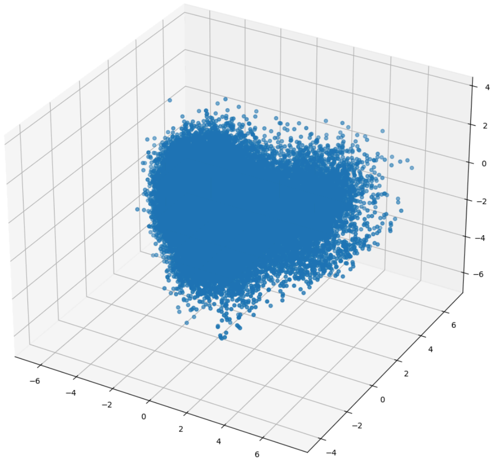
First, I imputed the most common value in each column in place of missing values (including “nan”, “?”) in order to preserve the size of the dataset. I replaced duration values of -1.0 with the median duration. I then one-hot encoded the key and genre of the music, as they did not make sense to interpret numerically. I dropped the first one-hot encoded column for both. I did the same with the mode of the music, with 1 representing “minor” and 0 representing “major”. I did not use instance ID, obtained date, artist name, or track name, as they are difficult to predict using or are not related to characteristics of the song. I then standardized the columns containing quantitative data using z-scores.

Dimension Reduction:

I used LDA for feature extraction for the purpose of classification due to its better performance on labeled data. This produced the following graph of explained variance ratios for each component.



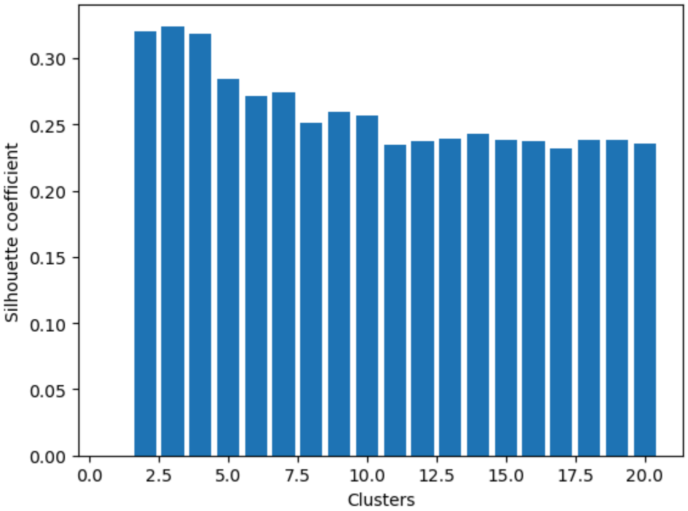
Using the elbow method, I decided to use only the first 3 components, which produced the following graph.



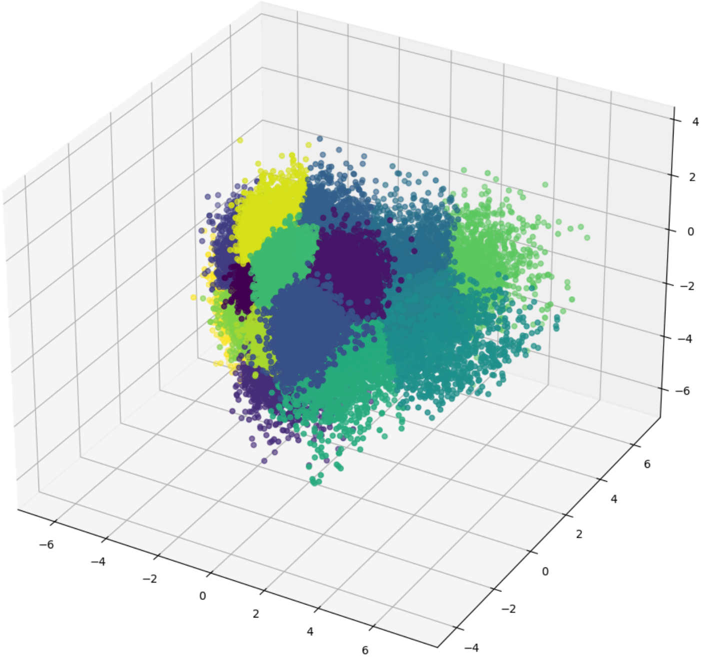
I would have liked to use a nonlinear dimension reduction method as well in order to make the clustering step after this easier, and to create better looking graphs. However, my laptop was unable to do this as the runtime was very high.

Clustering:

I chose to use kMeans as I lack the domain knowledge to properly set the hyperparameters of DBSCAN. Also, kMeans can run in a reasonable amount of time on my laptop. Using the silhouette method on a range of 2-20 clusters resulted in the following graph.



This shows that 17 is the correct amount of clusters, because it minimized the silhouette coefficient (0.232306). This is notably different from the 10 clusters from the genres named in the dataset that I would expect to see.

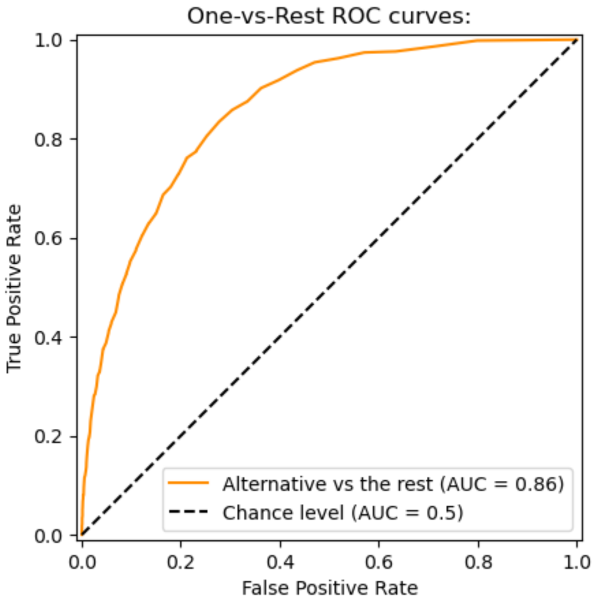


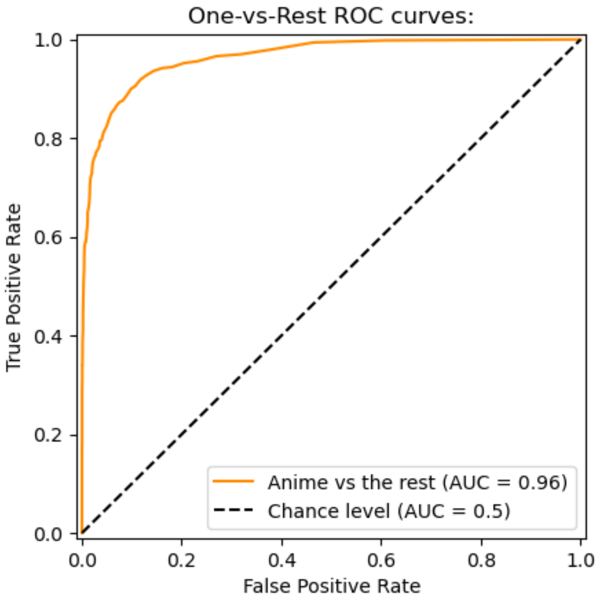
This graph is fairly hard to interpret as it is 3D, giving a total sum of the distance of all points to their respective clusters' center of 50813.33856.

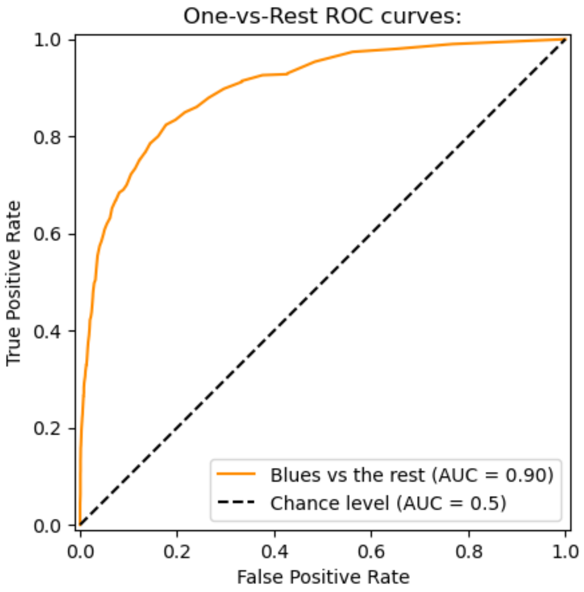
Fitting the Classification Model:

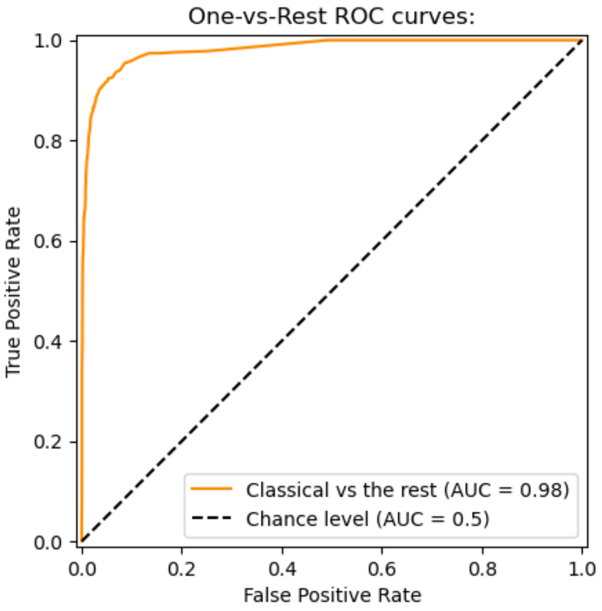
Because there was an equal amount of each genre in the dataset, I was able to simply select 10% of each genre for the test set using Scikit-learn’s train test split method, avoiding leakage.

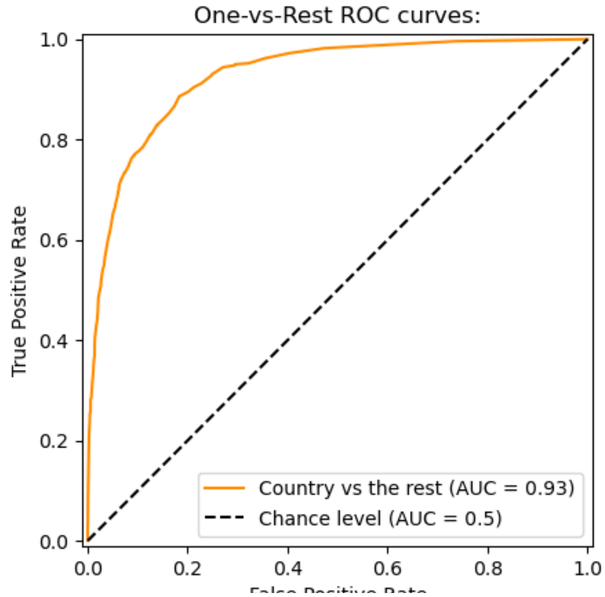
I chose to use a random forest with Gini as a criterion, as the time complexity of a neural network was too large. This produced the following one-vs-rest ROC curves.

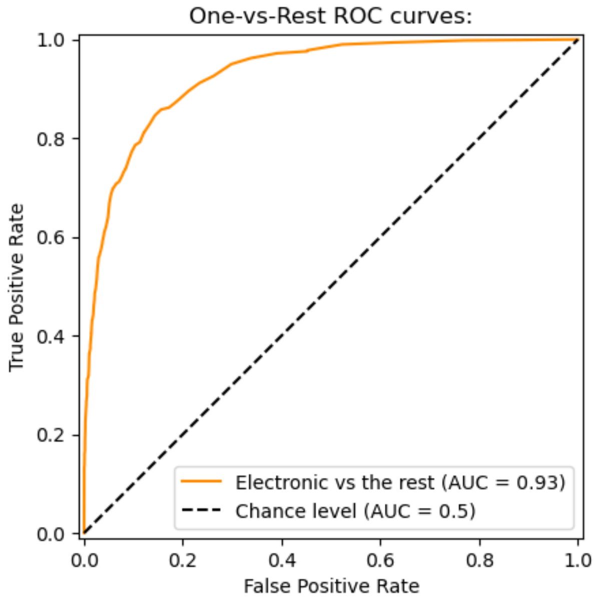


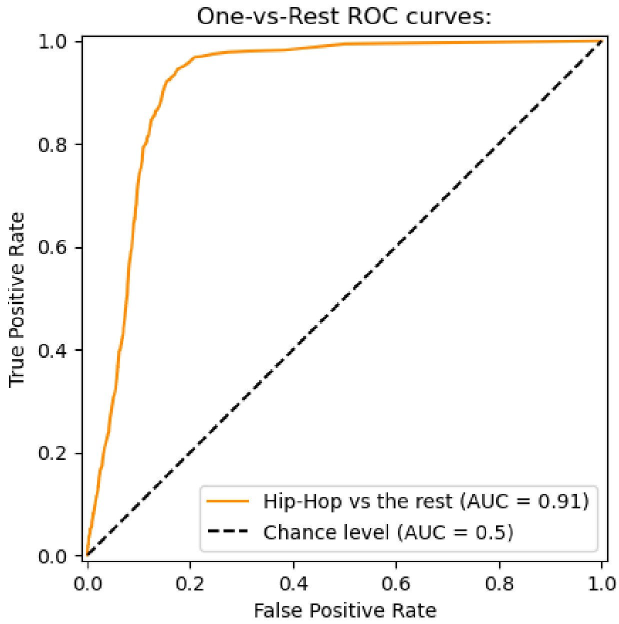


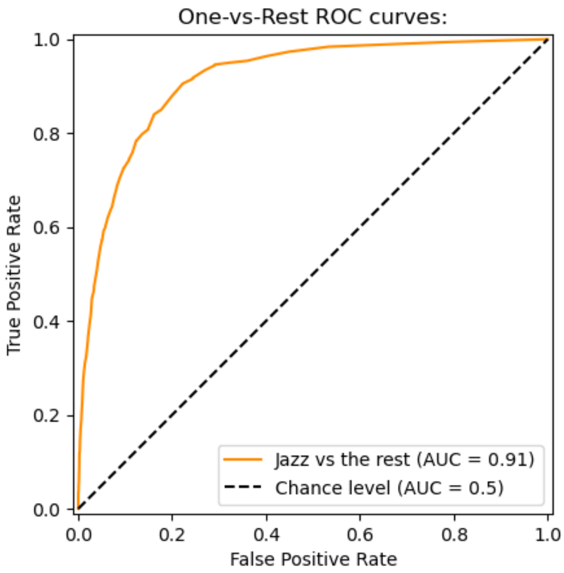


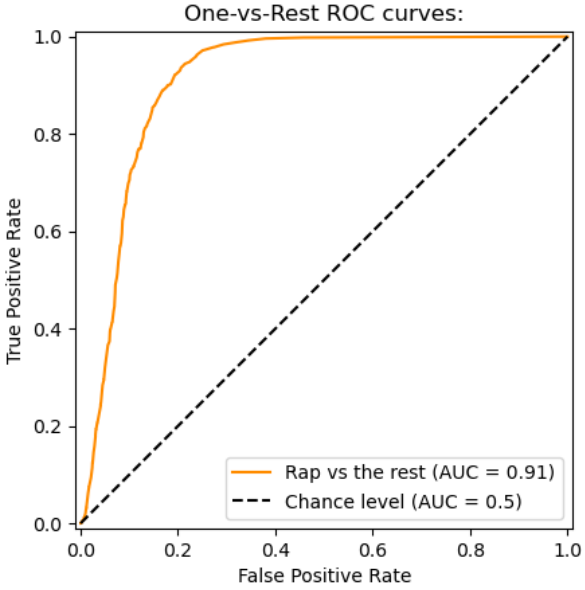


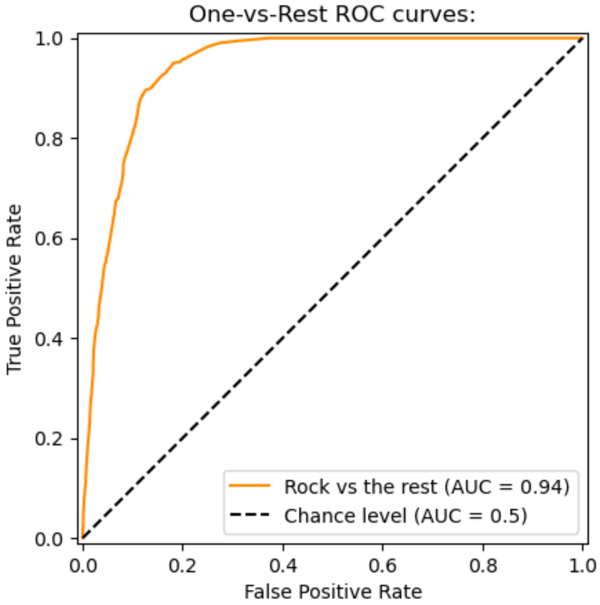

One-vs-Rest ROC curves:
True Positive Rate
False Positive Rate
0.0
0.2
0.4
0.6
0.8
1.0
Country vs the rest (AUC = 0.93)
Chance level (AUC = 0.5)


One-vs-Rest ROC curves:
True Positive Rate
False Positive Rate
0.0
0.2
0.4
0.6
0.8
1.0
Electronic vs the rest (AUC = 0.93)
Chance level (AUC = 0.5)


One-vs-Rest ROC curves:
True Positive Rate
False Positive Rate
0.0
0.2
0.4
0.6
0.8
1.0
Hip-Hop vs the rest (AUC = 0.91)
Chance level (AUC = 0.5)


One-vs-Rest ROC curves:
True Positive Rate
False Positive Rate
0.0
0.2
0.4
0.6
0.8
1.0
Jazz vs the rest (AUC = 0.91)
Chance level (AUC = 0.5)


One-vs-Rest ROC curves:
True Positive Rate
False Positive Rate
0.0
0.2
0.4
0.6
0.8
1.0
Rap vs the rest (AUC = 0.91)
Chance level (AUC = 0.5)


One-vs-Rest ROC curves:
True Positive Rate
False Positive Rate
0.0
0.2
0.4
0.6
0.8
1.0
Rock vs the rest (AUC = 0.94)
Chance level (AUC = 0.5)

These are good AUCs and show that the model is good at predicting the genre from the features extracted using LDA. It is best at identifying classical music and the worst at identifying alternative music. This makes sense, as alternative is a very broad genre of music in terms of the features given and classical is fairly specific.