Anthony Lu ayl2063

FML Capstone: Music Classification

Preprocessing:

To clean and process this dataset, I first started by looking at how to handle missing data. To see if missing data could be dropped, I looked at what percentage of the dataset was missing. Since only 0.01% of the data was missing for all features, we can simply drop the rows with missing values without having a noticeable impact on the data itself. Some data in the tempo feature (after dropping NaN rows) were also labeled as "?" and some data in duration was labeled as "-1", which I decided to impute using the median, since the acoustic features weren't necessarily normally distributed.

Next, I needed to handle the categorical data. I coded the "key" feature into numbers based on the key in musical notation, with each half-step increase corresponding to an increase of 1 in the numerical code (coding "A" as 1, "A#" as 2, "B" as 3, etc). I then encoded "mode", with "major" corresponding to 0 and "minor" corresponding to 1. Finally, I encoded "genres". I encoded "Electronic" into 0, "Anime" into 1, "Jazz" into 2, "Alternative" into 3, "Country" into 4, "Rap" into 5, "Blues" into 6, "Rock" into 7, "Classical" into 8, and "Hip-Hop" into 9. I opted to not include the artist, song title, SpotifyID, and date obtained in my model since artist and song title don't provide insight into genre beyond linguistic properties and SpotifyID and the date obtained only tells us information about the Spotify database itself. Finally, I split the data into the training and test set using the recommended split (4500 from each genre for training, 500 from each genre for test) prior to dimensionality reduction, clustering, and classification, to prevent leakage. I then standardized all the features except the categorical features (key and mode).

Dimensionality Reduction:

Due to the high dimensionality of this dataset (14 features), we encounter issues such as interpretability and computation cost, which makes dimensionality reduction necessary. With high dimension data, the large number of features may also lead to overfitting, which could lead to poor performance on test data, and the large feature space may also make it difficult to find meaningful patterns. Dimensionality reduction helps us reveal hidden structures or patterns in the data, while removing redundant features and noise, making the classifier's job easier. The main candidates for this task are t-SNE, MDS, and PCA. t-SNE and MDS are both less efficient due to their poor time complexity, which leads to long computation time for larger datasets. PCA avoids this problem since it reduces dimensions linearly. I ended up using PCA to dimensionally reduce the data due its speed and scalability.

After standardizing the training set, I used it to fit the PCA. I then looked at the eigenvalues of each principal component (Figure 1), to see which principal components contributed the most variance and could be selected for the next step. I also inspected the loadings of each principal component to see how the features contribute to each component (Figure 2).

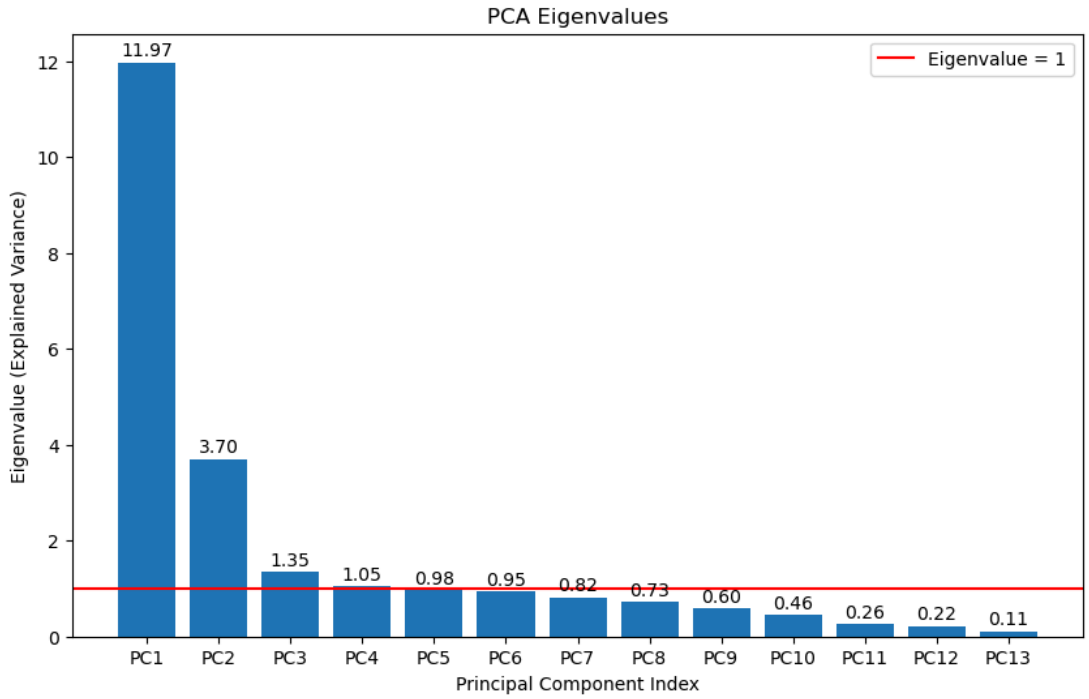

Figure 1: PCA Eigenvalues

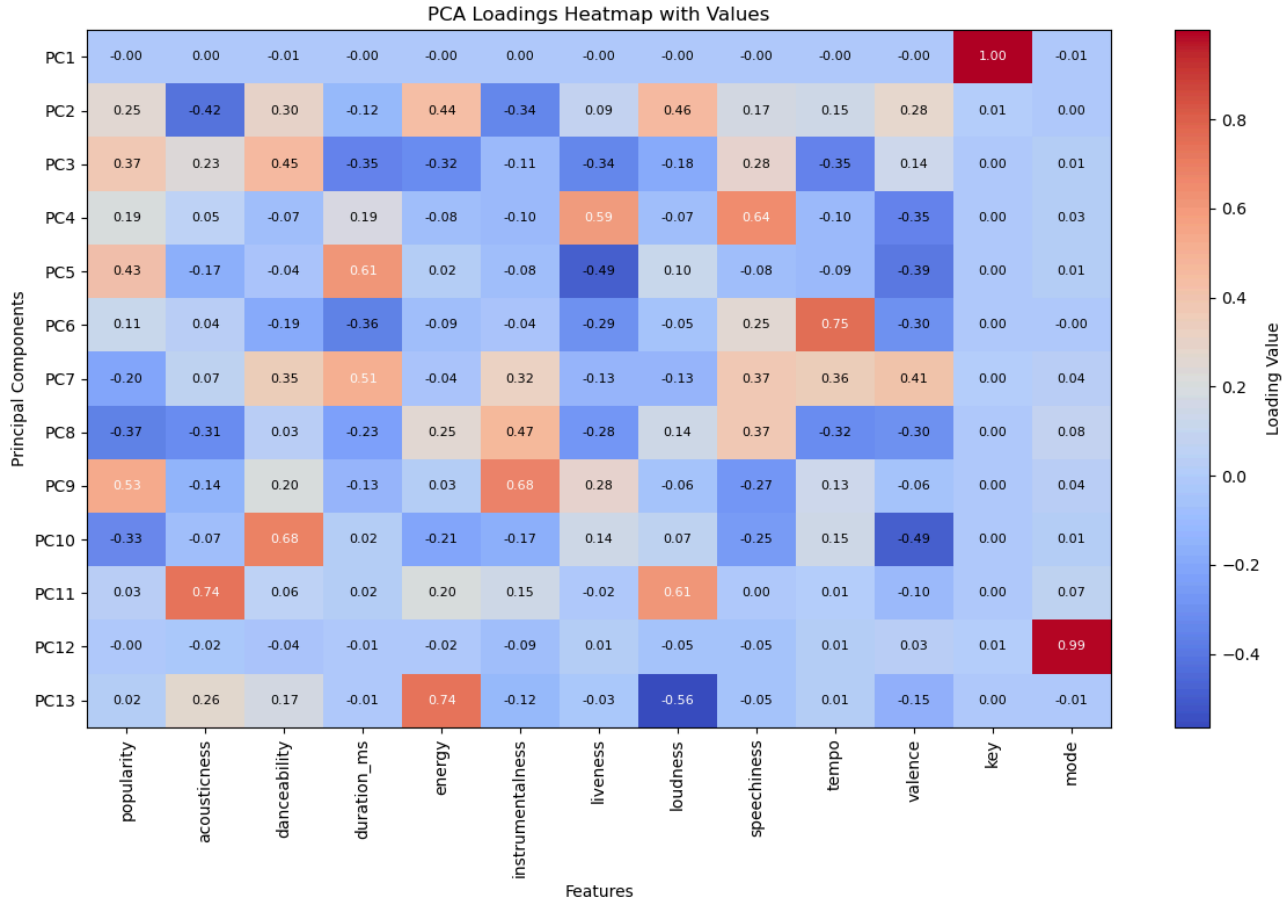

Figure 2: PCA Loadings

Using the Kaiser Criterion, I kept the first four PCs for the next steps (Figure 1). As we can see in Figure 2, PC 1 is dominated by "key", which could be explained by key's ordinal coding. PC 2 is dominated by features such as "accousticness", "energy", and "loudness", PC 3 is dominated by features such as "danceability", and PC 4 is dominated by features such as "liveness" and "speechiness". When the PCs are projected onto the first two components

we can also see how “key” dominates the component analysis, since the data points organize themselves into 12 stripes, which correspond to the 12 keys (Figure 3).

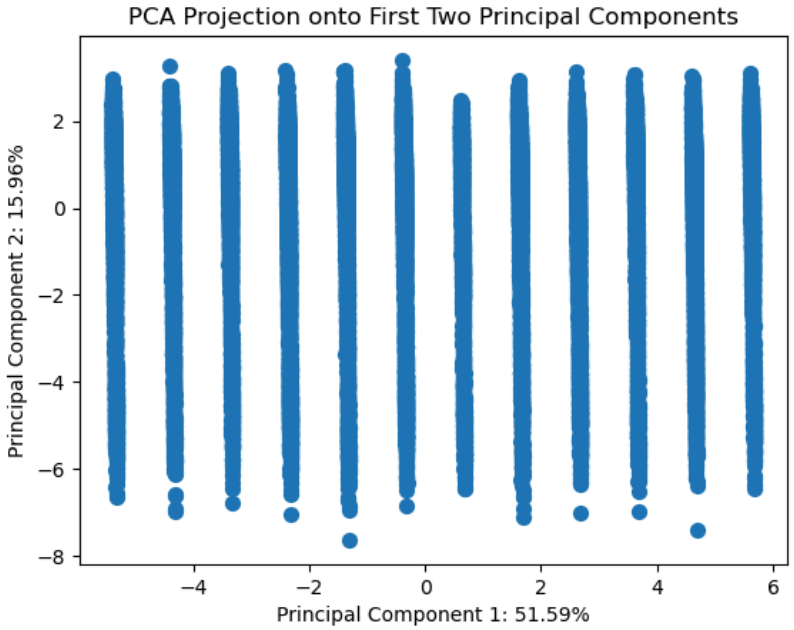

Figure 3: PCA Projection onto first two PCs

Clustering:

I made use of a clustering step to further uncover patterns or groupings. I chose to use k-Means due to its more efficient runtime, which scales better to larger datasets such as this one, as well as its simplicity in hyperparameter tuning (searching for optimal k using the Silhouette method vs arbitrarily setting epsilon and minimum samples for DBSCAN). Since PC 1 was so dominated by “key” (loading value of 1.00), I clustered using PCs 1-4 to look for patterns/groupings in the other auditory features, to prevent clustering to each key. Using the Silhouette method (Figure 4):

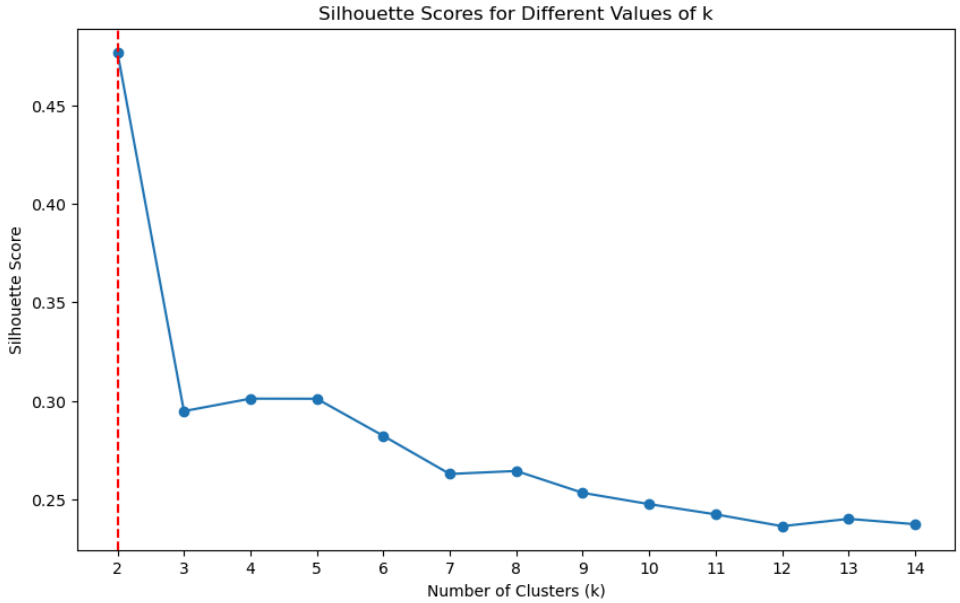

Figure 4: Silhouette scores for different k’s

As we can see in Figure 4, the optimal number of clusters was 2. I plotted the labels for the two clusters onto the 2-dimensional projection of PCA (Figure 5) as well as the 3-dimensional projection of PCA (Figure 6).

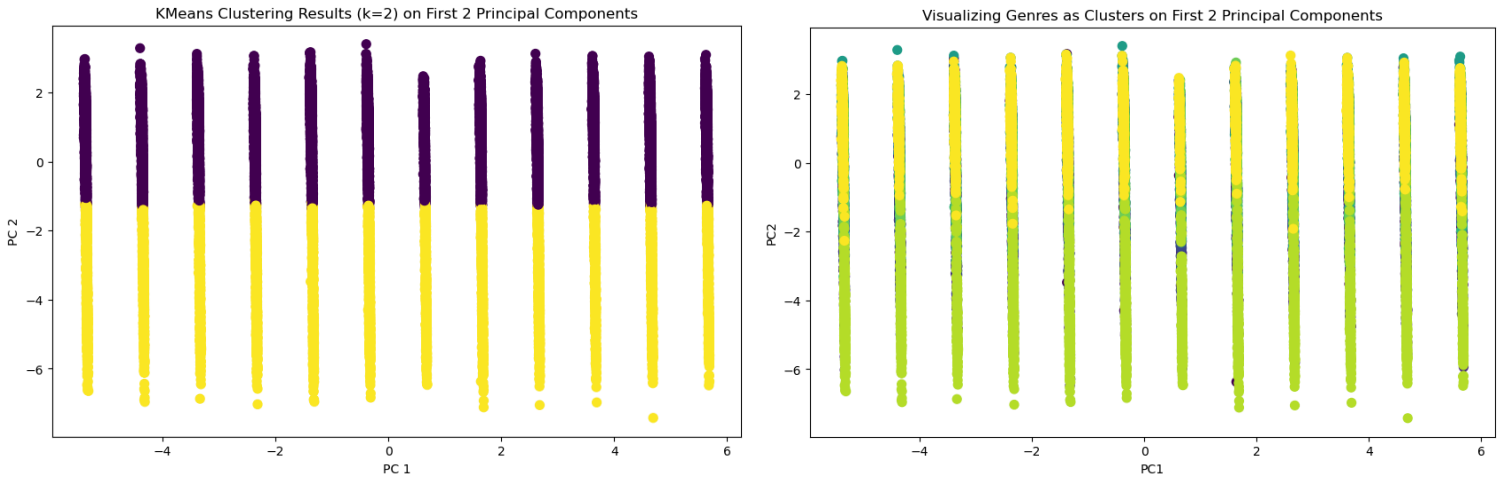

Figure 5: k-Means clustering compared to genres visualization (first two PCs)

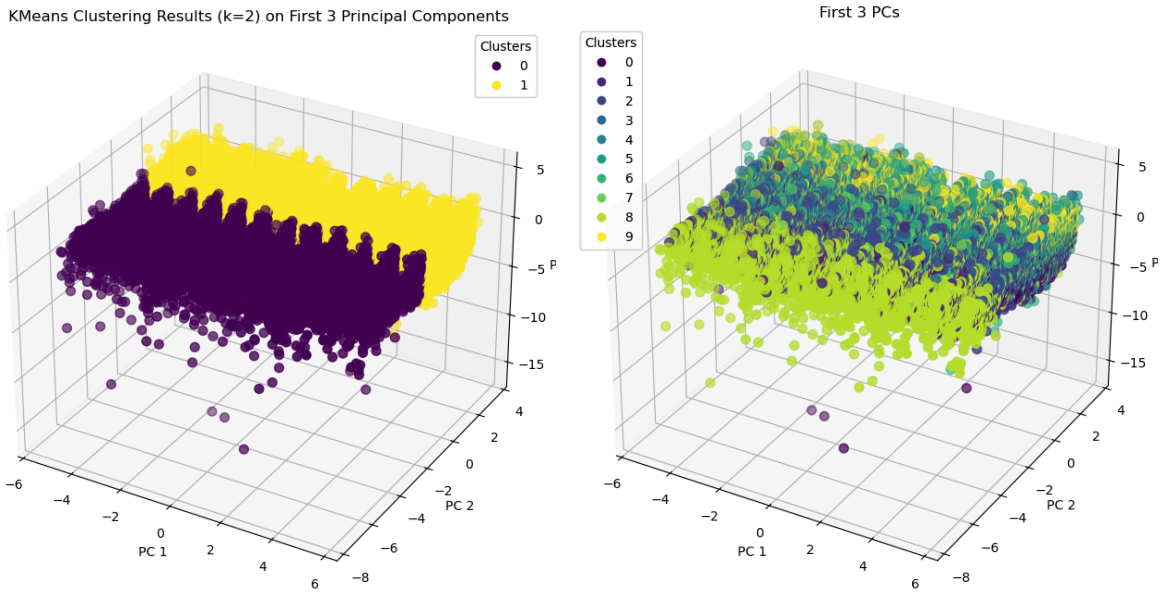

Figure 6: k-Means clustering compared to genres visualization (first three PCs)

As we can see in Figure 5, the labels clearly split up each stripe into two halves. Although there are two clusters rather than ten (ten genres), the clusters separate at a line parallel to PC1, which matches how the genres separate in Figure 3. This tells us that half of the genres could have distinct differences to the other half of the genres. Based on the PCA loadings, the yellow cluster seems to be made up of genres/songs with higher energy and loudness, while the purple cluster seems to have lower energy and loudness. In three dimensions, we can also see some separation between genres along the axis of PC3 (Figure 6).

Classification:

To choose a classification model, I first started with Logistic Regression, to build a baseline for performance. To gauge performance, I one-hot encoded the test outputs and plotted the ROC curves for each genre (Figure 7). ROC plots the True Positive Rate against the False positive rate for a class, so we need to use a one-vs-rest approach to measure how good the model is at predicting a class against all the other classes.

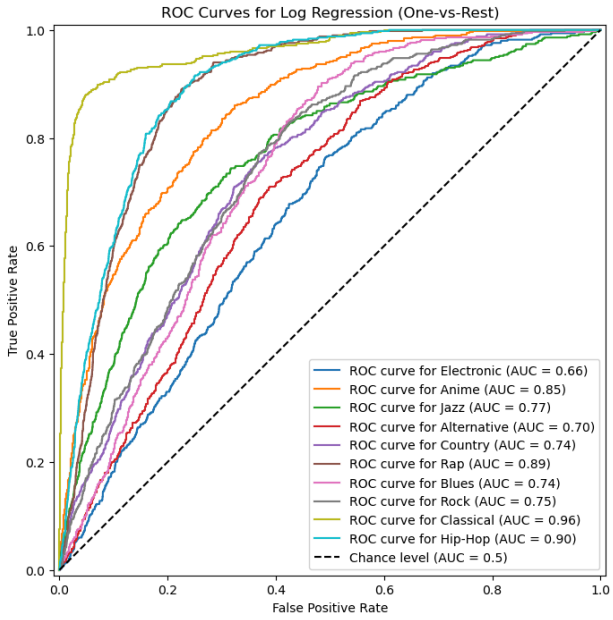

Figure 7: ROC Curves for logistic regression

Logistic regression already performs quite well in terms of AUC scores, as we can see in Figure 7. I then compared performance to a Support Vector Machine and decision tree based models. Looking at the average AUC score of each model, none of the four models (SVM, decision tree with adaBoost, decision tree bagged, and Random Forest) showed any improvement. With the SVM and Random Forest models,performance was very similar to the logistic regression, while performance dropped with the two single tree models (adaBoost and bagging). I then looked at the performances of a Multi-Layer Perceptron model with one hidden layer with 100 hidden units and a Fully Connected Neural Network with 200 hidden units and 2 hidden layers. The MLP showed a solid improvement in average AUC (0.80 to 0.83) while the Neural Network showed a loss in performance (Figure 8), even though the Neural Network was much higher in complexity.

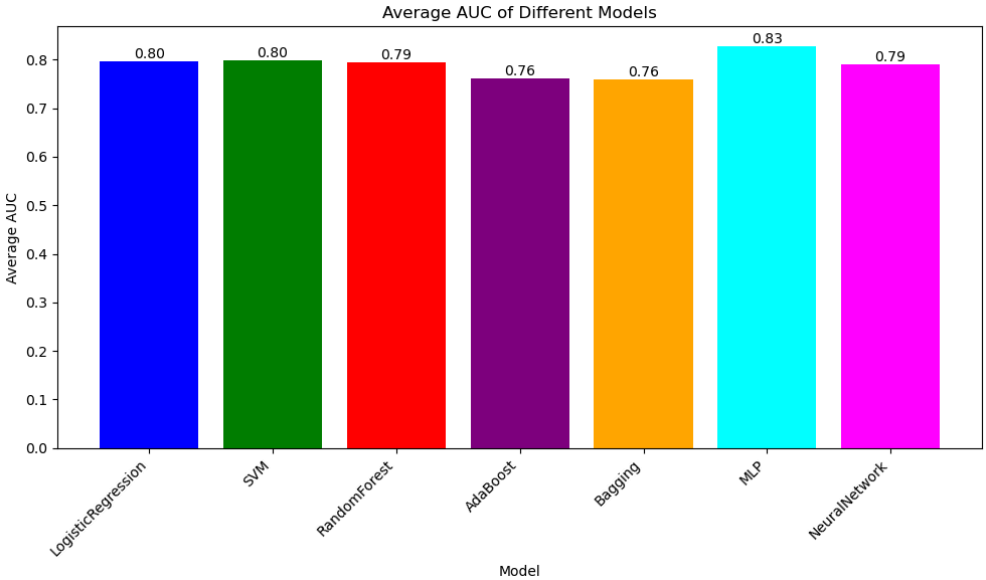

Figure 8: Models average AUC comparison

Conclusion:

The dimensionality reduction step seemed to have the best impact on performance gains. Reducing dimensions allowed less important features to be filtered out which led to improvements in both model performance and runtime. With more complex models such as the Multi-Layer Perceptron, the model was to train in a reasonable amount of time when attempted with the full dimensions of the original training set. Even though the k-Means clustering didn't result in clusters perfectly matching the genre classes, the clustering patterns did help us confirm that the structure in the PCs does correspond to the genres (Figure 5 and 6).

My final model consisted of a **Principal Component Analysis** for dimensionality reduction, **k-Means** for clustering, and a **Multi-Layer Perceptron** for classification, and an overall average AUC score of **0.83** (Figures 8 and 9).

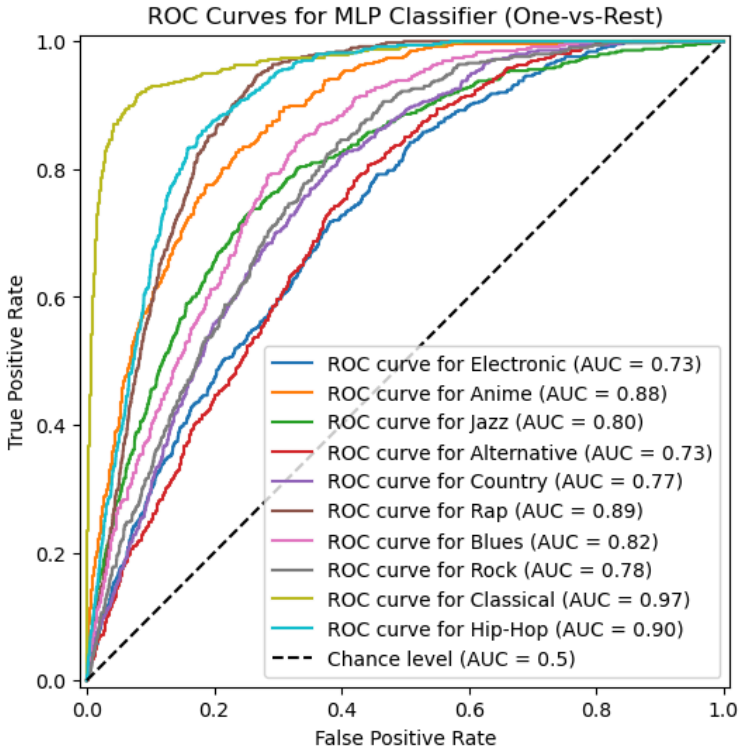

Figure 9: MLP Classifier performance

Additional Insights:

To gain additional insights on the data and how it impacted the dimensional reduction step, we can look at the correlations between the features and the labels (genres) (Figure 10).

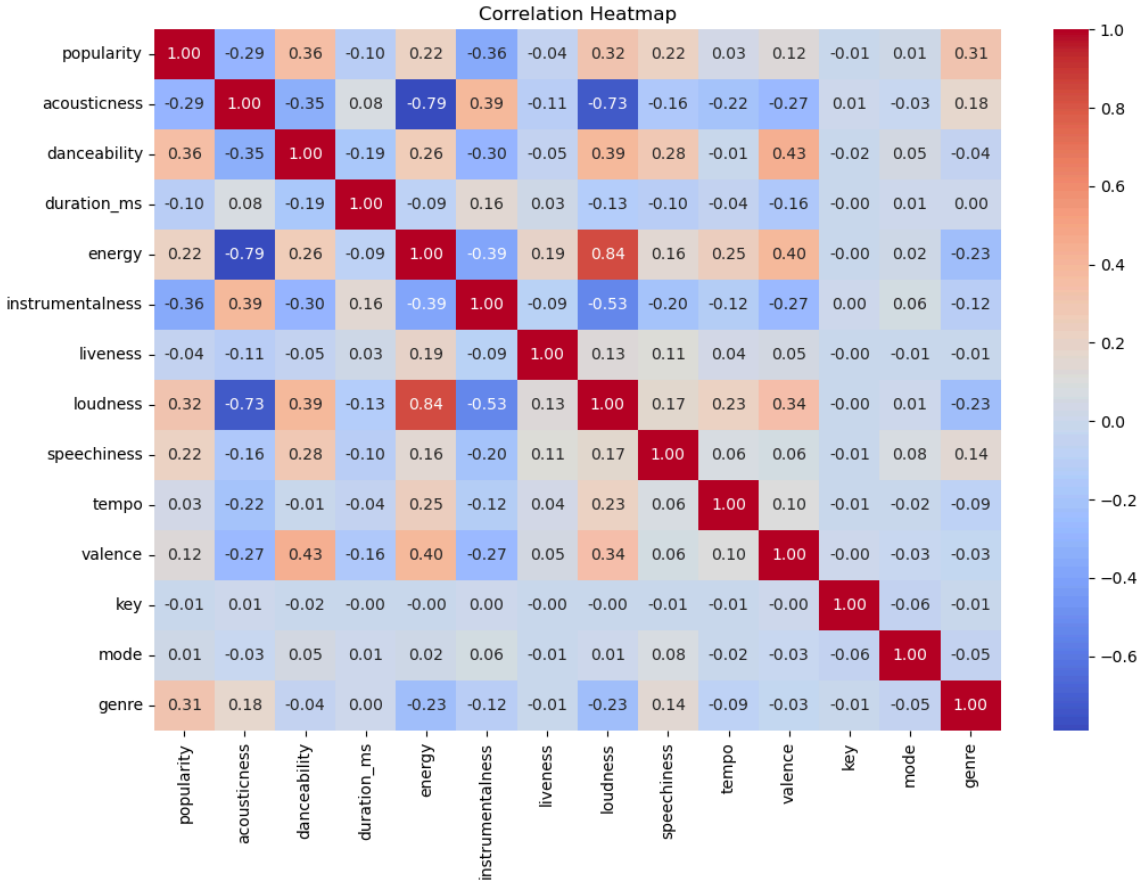

Figure 10: Correlation heatmap

As we can see in Figure 10, the acoustic features (acousticness, danceability, duration, energy, instrumentalness, liveness, loudness, speechiness, tempo, and valence) along with popularity, all have moderate to strong correlations with each other, which makes sense since different acoustic features contribute to each other. For example, loudness and energy have a high correlation of 0.84, which makes sense since high energy music tends to be louder. However, interestingly, key and mode both have little to no correlation with each other and the other variables. This tells us that even though the feature “key” explains a lot of the variance and structure in the data (which we know from the PCA), they have little to do with the relationships in the data itself.