Ethan Elizar

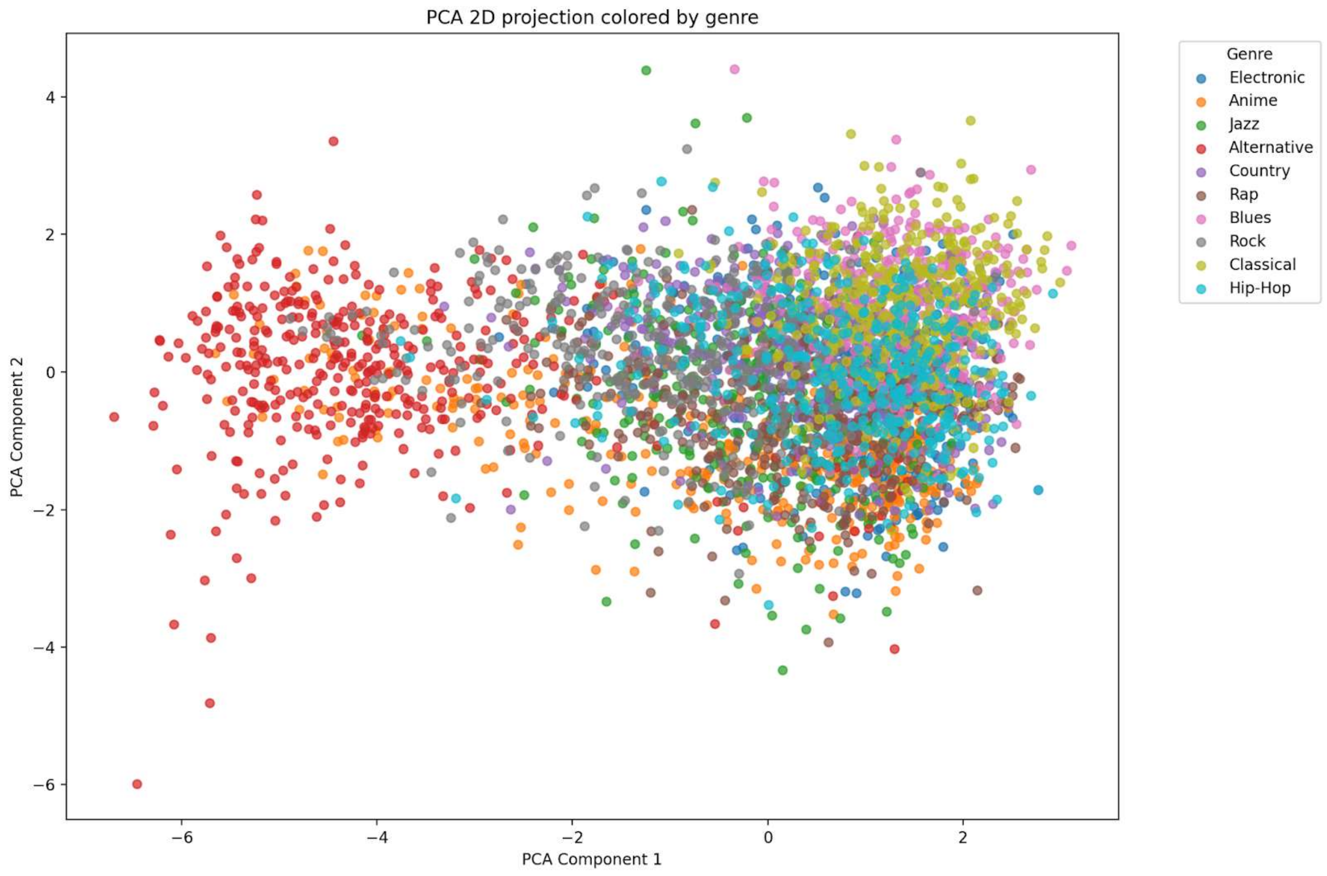
Spotify Classification Project

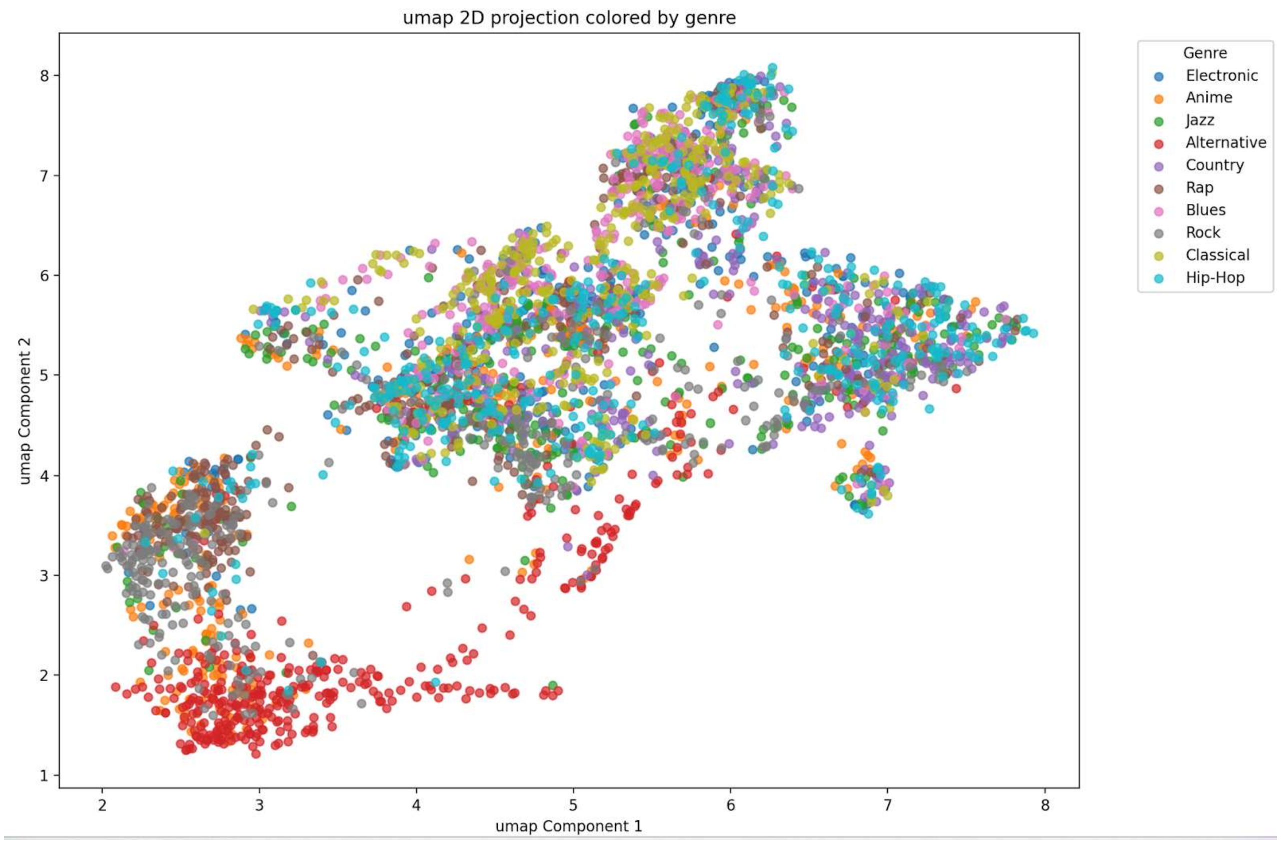
May 10, 2025

To create a classification model, some steps needed to be completed first, including cleaning of the data, dimensionality reduction, and clustering. When cleaning the data, I first had to address the randomly missing data, which included random question marks and -1 values that appeared in columns like tempo and 'duration_ms'. I replaced these question marks and -1 values with NaN values, and dropped using dropna all rows that contained NaNs. Next, I needed to convert key into numerical data, which I did by one-hot encoding the key into variables 'key_A' through 'key_G#', so these values become only 0's and 1's. I also one-hot encoded the mode and music_genre variables, so these values also became just 0 and 1 values to represent a song representing a given mode or given music genre.

For performing the train/test split, I used my random seed and did a 90/10 train test split so that 10% randomly picked songs from the genre were in the test set and the rest in the training set. The last data cleaning procedure I did was to separate numeric columns and categorical columns, only normalize the numeric columns, and then concatenate the normalized numeric and unchanged categorical columns together. This process also led to me not using variables that aren't useful for classification, like unique Spotify ID, artist name, song name, and obtained date.

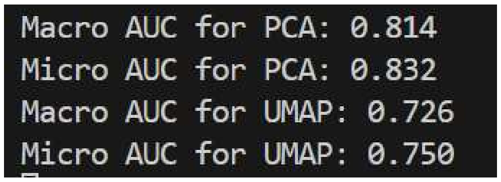
The next step was dimensionality reduction. I tried two methods for my dimensionality reduction, specifically PCA and UMAP, each with 6 principal components. I wanted to reduce the dimensions, but not oversimplify the data to the point where classification failed. I tried PCA as it gives very interpretable/easy to use principal components, and I tried UMAP because I wanted to see if a non-linear dimensionality reduction performed well on this data. I also wanted to see if UMAP provided some interesting visualization that PCA couldn't, since UMAP can capture complex patterns. Below I have provided the visualization of genres as clusters in the lower dimensional space.





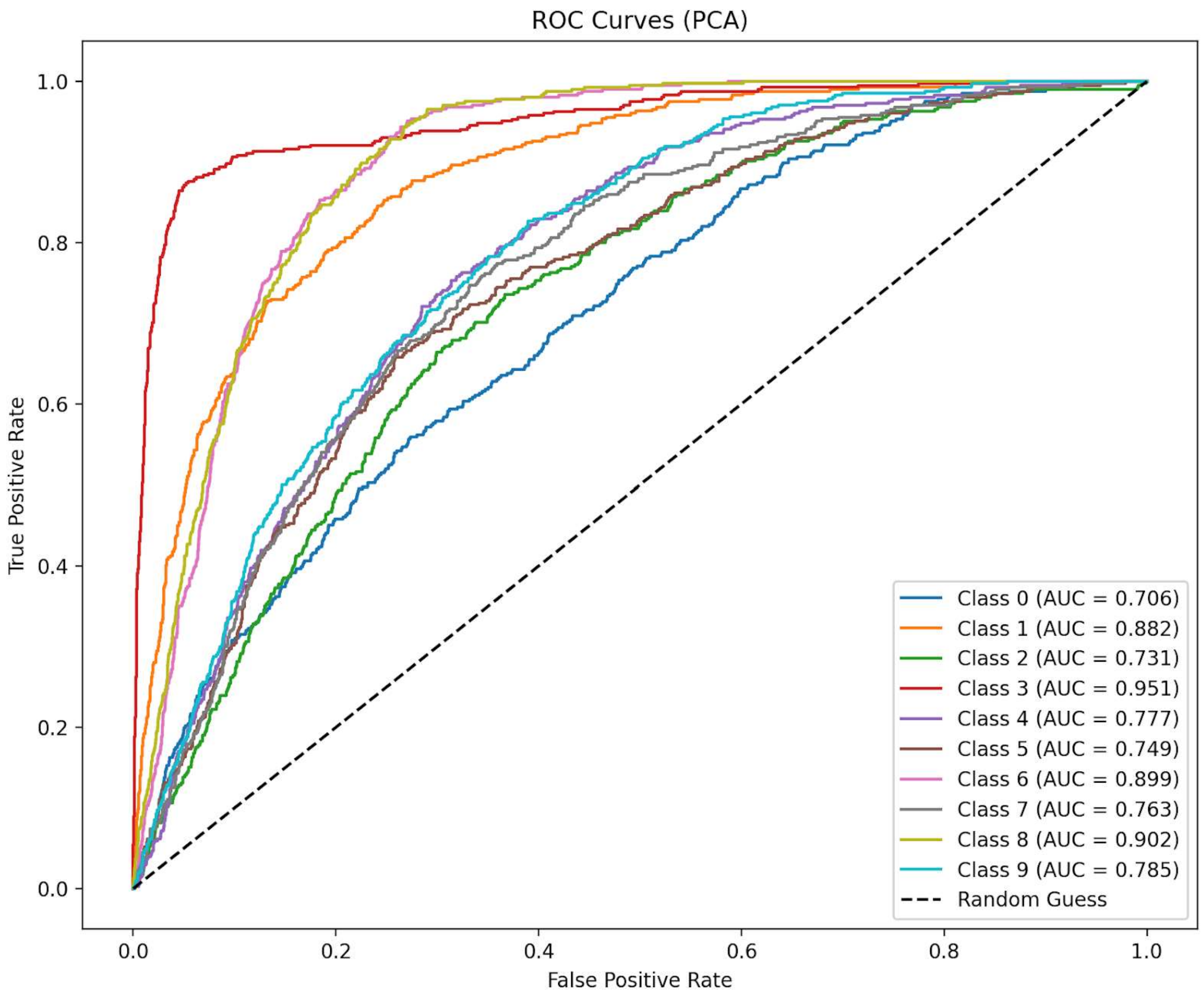
As I expected, the UMAP visualization provides much more “separation” between what could be clusters than the PCA. However, when putting the true genre labels over each point in the 2D prediction, the UMAP actually provides large separation to make apparent triangle shaped clusters, almost like a starfish, but then the genre labels are all mixed up even in these separated segments. The PCA has less clear separations, and especially for the positive components of PCA Component 1, the genres seem to be very mixed up. One thing I can see in common with both predictions with genres as clusters is that they both seem to be fairly good at separating “alternative” songs, which are in red. Additionally, there are zones in both projections where “classical” or light green seems to be somewhat clustering together. These genres may have more unique groupings of characteristics than the others. I also tried seeing how Kmeans clustering with 10 clusters compared to these projections above, and it confirmed my suspicion that especially for UMAP, it created some separations for theoretical “clusters” that weren’t actually based on genre differences.

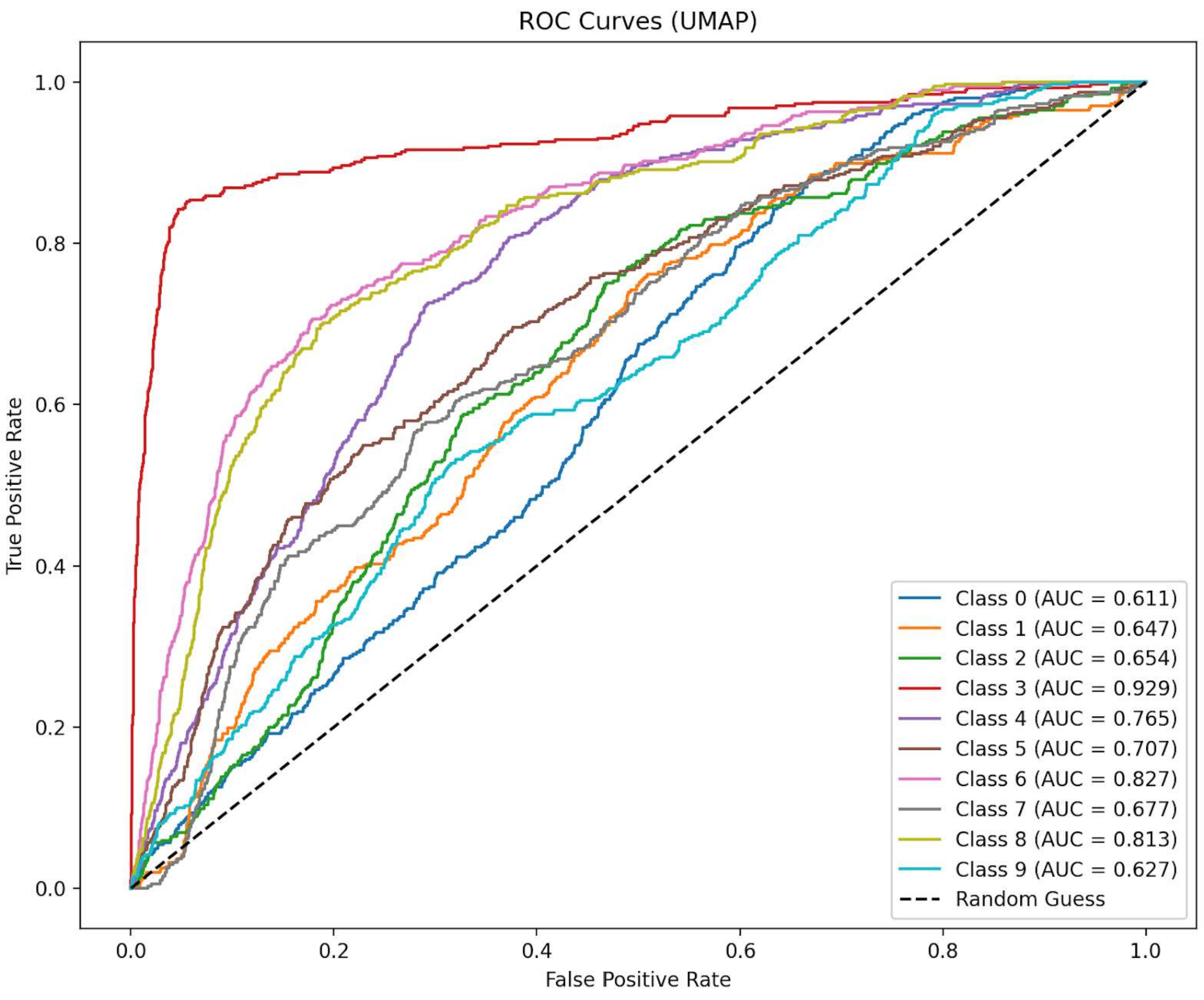
Lastly, I needed to actually make and visualize the classification model. I used Logistic Regression, and since this is multi-class classification instead of binary, I made sure to use a MultiOutputClassifier from sklearn. To ensure no data leakage between the train and test sets, for both the PCA based and UMAP based on models, I fit the multinomial logistic regression on the training components, and got predicted probabilities from the test components in order to calculate the ROC. The Macro AUC (unweighted average of class AUCs) and Micro AUC (global AUC for all classes) are below for PCA and UMAP. The plots of the AUC curves, since this was multi classification, ended up being a curve for each class, which confirmed for both PCA and UMAP that class 3, “alternative” was the easiest to classify. So ultimately, the highest AUC I could get was with PCA, where I got an AUC of **0.832,** which is solid performance!



```
Macro AUC for PCA: 0.814
Micro AUC for PCA: 0.832
Macro AUC for UMAP: 0.726
Micro AUC for UMAP: 0.750
```

However, when looking at the precision and recall scores from the classification report, they were lower, suggesting the model does have some issues. The most important factor that I think underlined my classification success was making sure to clean the data correctly by not normalizing categorical variables and not turning a variable like key into a numerical value from 0 to 11. Making errors like this in the preprocessing of the data could cause a PCA projection that does not actually truly represent the data, and could introduce unnecessary noise in the classification process later. Also, making sure to not use too few principal components, which would oversimplify the data, also could have contributed to the score being above 0.8.


ROC Curves (PCA)
True Positive Rate
False Positive Rate
0.0
0.2
0.4
0.6
0.8
1.0
Class 0 (AUC = 0.706)
Class 1 (AUC = 0.882)
Class 2 (AUC = 0.731)
Class 3 (AUC = 0.951)
Class 4 (AUC = 0.777)
Class 5 (AUC = 0.749)
Class 6 (AUC = 0.899)
Class 7 (AUC = 0.763)
Class 8 (AUC = 0.902)
Class 9 (AUC = 0.785)
Random Guess


ROC Curves (UMAP)
True Positive Rate
False Positive Rate
0.0
0.2
0.4
0.6
0.8
1.0
Class 0 (AUC = 0.611)
Class 1 (AUC = 0.647)
Class 2 (AUC = 0.654)
Class 3 (AUC = 0.929)
Class 4 (AUC = 0.765)
Class 5 (AUC = 0.707)
Class 6 (AUC = 0.827)
Class 7 (AUC = 0.677)
Class 8 (AUC = 0.813)
Class 9 (AUC = 0.627)
Random Guess