Music Genre Classification

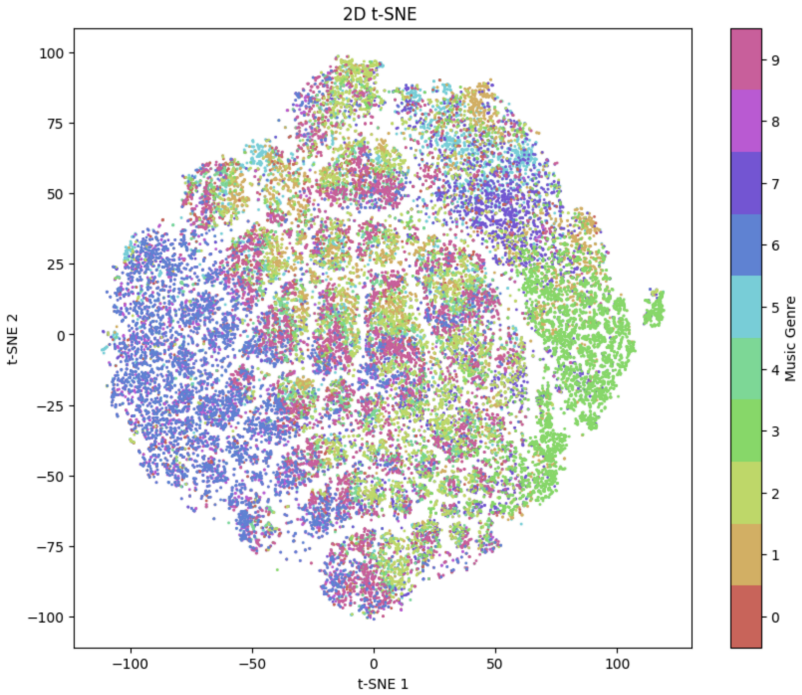
Overview:

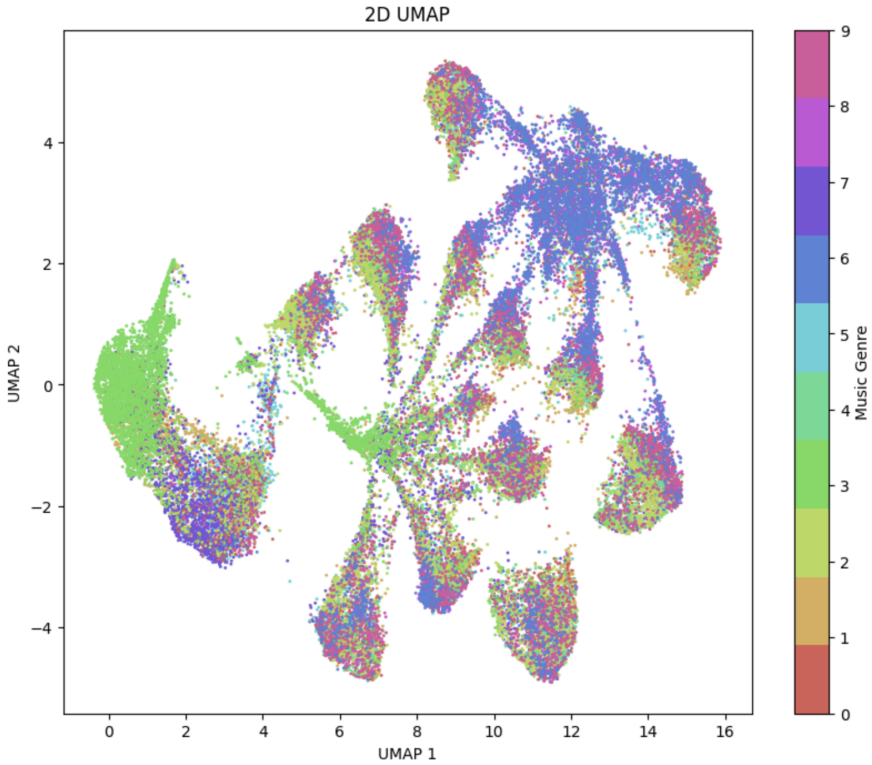
In this project, I built a classification model to predict the genre of a song based on its features.

Process:

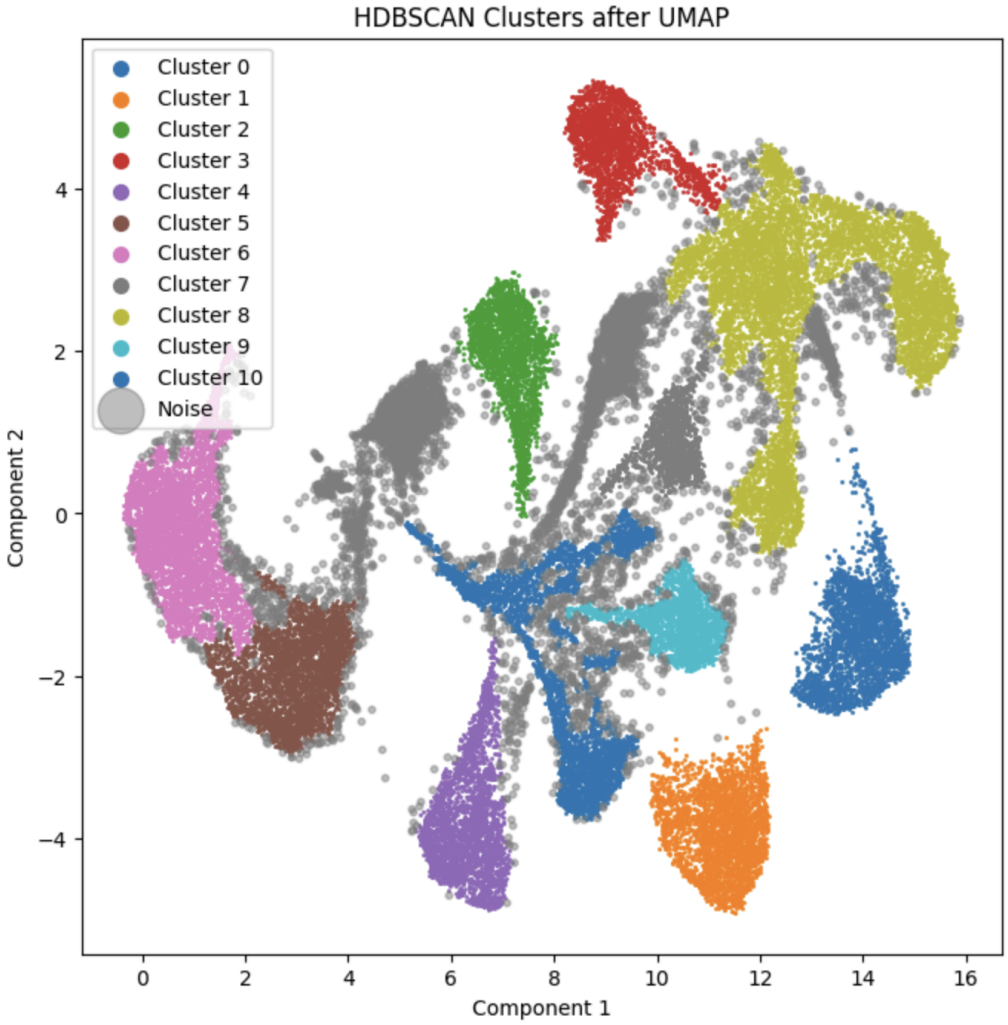
- Data Preprocessing
    - Rows with missing instance_id, artist_name, and track_name were removed as they did not provide any value.
    - Missing values in duration_ms and tempo were replaced with their respective medians.
    - The key column was one-hot encoded. The mode and music_genre columns were label encoded. I did not encode the mode similarly to the key column because mode doesn't play a critical role in genre (that is, even a sad song can have a major mode).
    - Numerical features were standardized using StandardScaler().
- Train/Test Split
    - As per requirements, I split the dataset such that 500 randomly selected songs from each genre formed the test set, and the remaining songs were used for training.
- Dimensionality Reduction and Visualization
    - I used t-SNE and UMAP to reduce the feature space to 2 dimensions for visualization. While t-SNE did not provide any particularly distinct clusters for different genres or other features, the UMAP plot has shown a bit better clustering for different genres (with many genres overlapping, which makes sense in the context of music since music genres cannot be clearly defined - that is, for example, jazz elements are sometimes used for pop music, alternative or hip-hop).
    - I chose to do t-SNE and UMAP (and not PCA) because the dataset that was given very likely contains non-linear relationships that PCA (a linear method), might not capture effectively (I also attempted doing it with this dataset and I didn't find any particular insights from PCA alone). t-SNE is good at visualizing high-dimensional data, but it is computationally very intensive - therefore in the resulting notebook only t-SNE with perplexity set to 30 and number of components set to 2 is displayed. UMAP, compared to t-SNE, allows to preserve more of the global structure and also has faster computation - I chose

n-neighbors set to 30 and number of components set to 2 as the best parameters.





- Clustering Analysis
    - To get a better clustering from UMAP results, I also did HDBSCAN. This clustering helps to understand the inherent grouping of songs based on their features a bit better. While this clustering doesn't represent genres exactly, it is a good visualization and idea of how the data can be even grouped for genre classification. The hyperparameters for HDBSCAN were the following: in_cluster_size=1500, min_samples=10
    - Note: I chose HDBSCAN because it can handle clusters of varying densities, which I thought would be a good fit for the music genre dataset. Also compared to DBSCAN, HDBSCAN can handle noise.



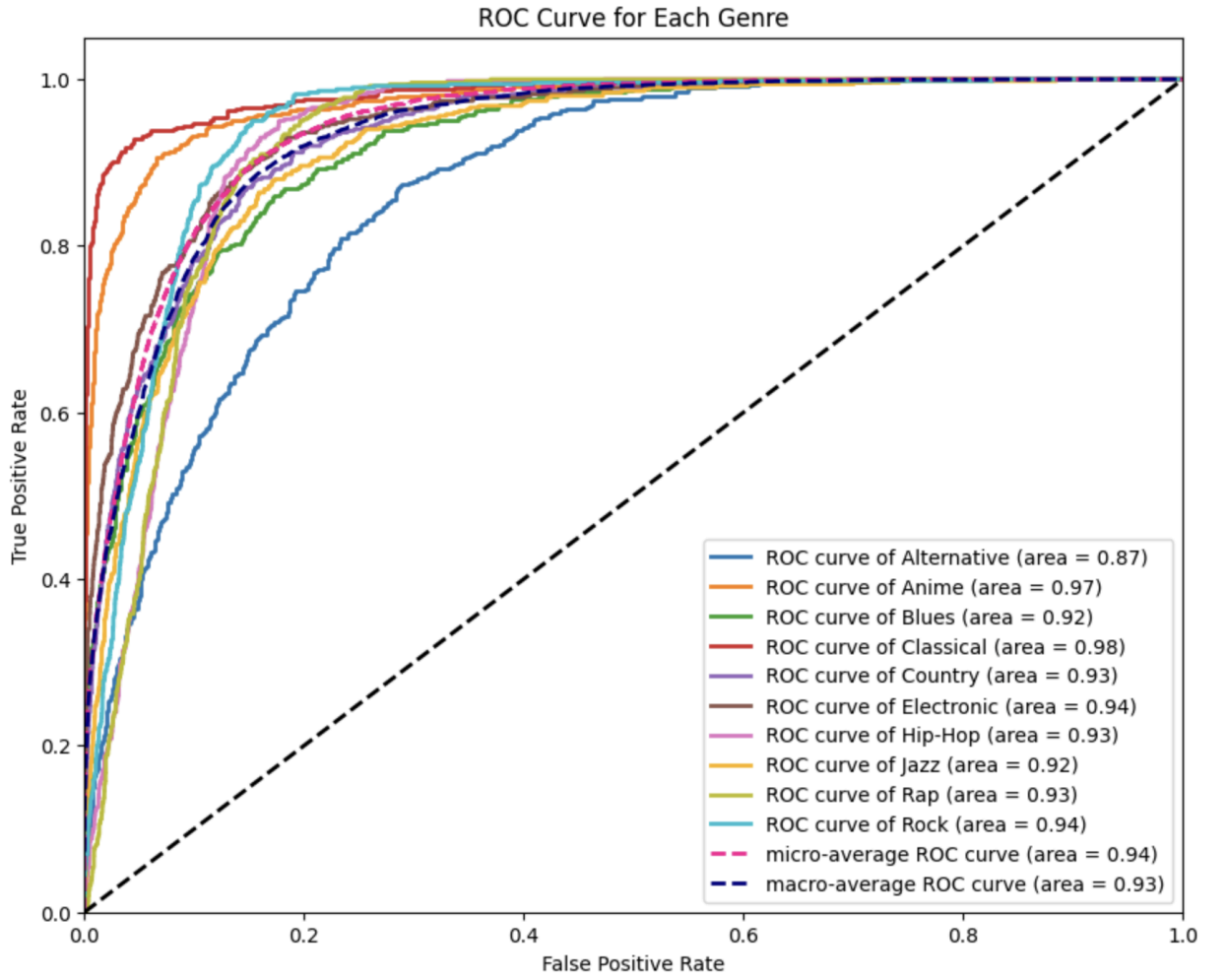
- Model Training
    - After performing UMAP for dimensionality reduction and HDBSCAN for clustering, I added the cluster labels as features to the training and test sets. This

was done in hopes to increase model accuracy for classification by expanding the feature space further.

- Then, I used XGBoost for classification. I chose XGBoost because it is fast and can handle a variety of data types and complexities.
- Then, I calculated both ROC AUC and accuracy metrics.

Results:

The mean ROC AUC for the test set was 0.9345. The train ROC AUC was 0.9496 and the test ROC AUC was 0.9322, which shows that the model can distinguish between classes pretty well.



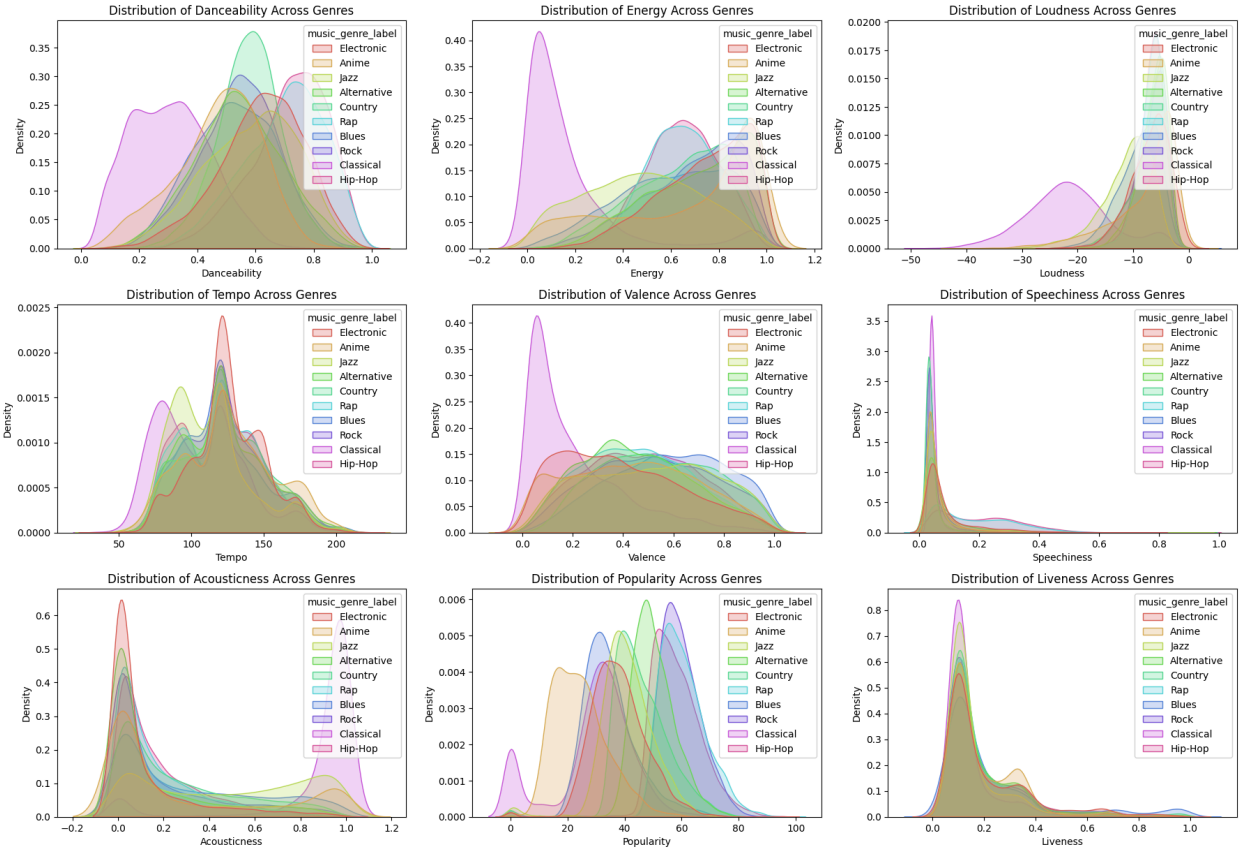
In terms of accuracy, 64.59% of the training samples were correctly classified and 58.82% of the test samples were correctly classified by the model. The model's predictive power is quite low. And it has a slightly lower accuracy on the test set compared to the training set, which was expected as the model was exposed to unseen data in the test set, yet this suggests some level of overfitting, which perhaps could be mediated by having cross-validation, regularization, as well as proper hyperparameter tuning and further feature engineering.

It is also important to note that while AUC is high, the accuracy is pretty low. This might be because the model is better at ranking the correct class higher, even if it's not always predicting the exact class correctly. I am not entirely sure, maybe I also should have set the threshold more properly for this classification task (as described here). Overall, in the real world, I think the model would probably not predict genre very well, which is understandable since the idea of genre is a human-made concept and sometimes songs that have features of other genres fall into a completely different genre.

```
Train Accuracy: 0.6461
Test Accuracy: 0.5906
Train ROC AUC: 0.9495
Test ROC AUC: 0.9341

Classification Report (Train):
              precision    recall  f1-score   support

           0       0.54      0.41      0.47      4500
           1       0.84      0.78      0.81      4500
           2       0.69      0.60      0.64      4500
           3       0.87      0.86      0.87      4500
           4       0.63      0.64      0.63      4500
           5       0.72      0.68      0.70      4500
           6       0.53      0.57      0.55      4500
           7       0.62      0.61      0.61      4500
           8       0.54      0.53      0.53      4500
           9       0.54      0.78      0.64      4500

    accuracy                           0.65     45000
   macro avg       0.65      0.65      0.65     45000
weighted avg       0.65      0.65      0.65     45000

Classification Report (Test):
              precision    recall  f1-score   support

           0       0.44      0.38      0.41       500
           1       0.82      0.75      0.79       500
           2       0.63      0.55      0.58       500
           3       0.86      0.88      0.87       500
           4       0.61      0.56      0.59       500
           5       0.65      0.62      0.64       500
           6       0.43      0.45      0.44       500
           7       0.56      0.54      0.55       500
           8       0.43      0.45      0.44       500
           9       0.51      0.73      0.60       500

    accuracy                           0.59      5000
   macro avg       0.59      0.59      0.59      5000
weighted avg       0.59      0.59      0.59      5000
```

The most important factor in my classification success was probably feature engineering in my opinion, as well as data preprocessing.

As an extra, I also visualized different audio features with respect to their genre.



From this we can see that a lot of genres do share very similar distributions - which does confirm that music genre classification is a complex task. Popularity, acousticness, danceability and energy seem to have the most variability among genres which makes sense as music can be either calm or very energetic / aggressive.

Visualization source: https://medium.com/@navodas88/music-genre-prediction-using-audio-features-96dea4e71ad3