Andrew Yan 5/8/25

Capstone Project: Classification Report

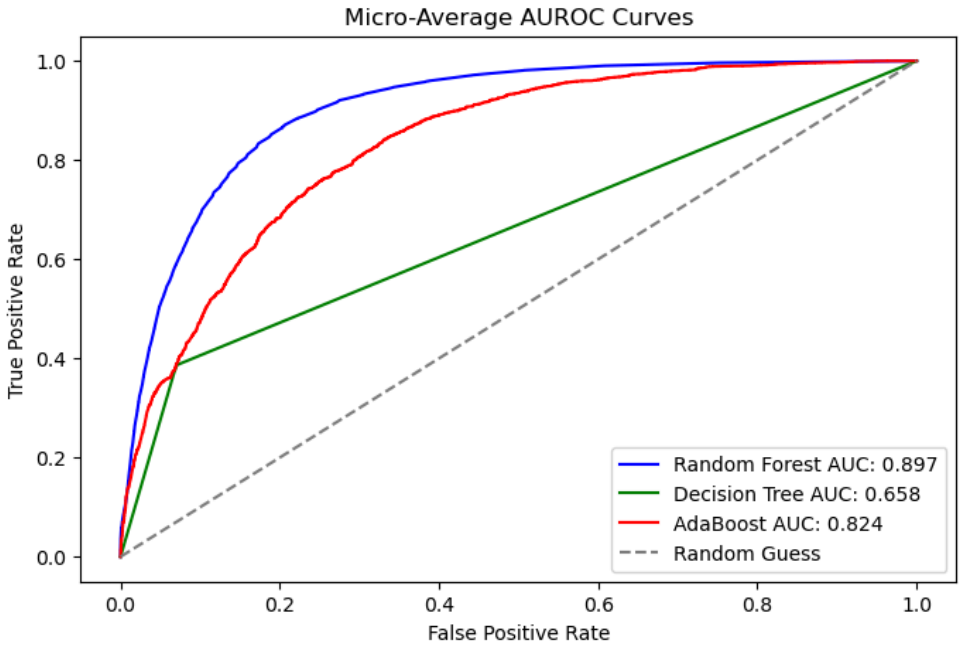
Preprocessing

I started by setting the random seed to my N-number so the train-test split, models, and results are unique. Then I opened the *musicData.csv* file and set a *df_filtered* variable and dropped the rows with explicit missing data. There were 50005 rows in the file, then dropping all NaNs resulted in 50000 rows. Then, I noticed there were random missing data, but they were marked with a *?* symbol, so Python wouldn't recognize those as NaNs. Then I converted them to NaNs, so I can handle them later without dropping those rows. After, I encoded the *music_genre* variable, to extract 500 rows for each genre and determine how many genres there are, so I can appropriately run a stratified train-test split such that 500 of each genre is in the test set and 4500 of the remaining data from that genre is in the training set. Then I set the *genre_encoded* variable for the y train and test variables and all other variables, except those with information about artist or track, as the X train and test variables. Then I set the numerical and categorical variables from X appropriately and used Pipeline to do a multi-step preprocessing where I use SimpleImputer to fill in any NaNs with either the median of the column (for numerical variables) or most frequent categorical input (for categorical variables), then scaled the numerical variables or One-Hot Encode the categorical variables. I set the *preprocessor* with ColumnTransformer to combine the numerical and categorical preprocessing. I also set the PCA with 0.95 as n_components as I would have enough variance and the models can handle a balanced 500-per-genre setup without serious overfitting. These are set up for the models later on. Finally, since the outcome variable is multi-class with 10 different genres, I set a binarized *y_test* variable for later on to show each model's correct and incorrect predictions.

Running 3 Supervised Models and Plotting AUROC Figure

After running the processing, I decided to run 3 classification models: Random Forest, Decision Trees, and AdaBoost, so I can evaluate and compare each model's performance and determine which model performs better on this dataset. For each model, I built them using the Pipeline module to run the preprocessor, the PCA as my dimensionality reduction, then set the respective models in one piece of code. For Random Forest, I set n_estimators to 100 since I have 50,000 total data and a relatively higher value can reduce the risk of overfitting. After building the model, I fit them using the training set, made prediction probabilities with the *X_test* set, and computed each of their AUC scores using the prediction probabilities and binary *y_test_bin*. After running all the models, I flattened each AUC score and *y_test_bin* as the actual *y_test* is a multi-class variable. This provides the micro-average AUC scores to treat all classes equally. Then plotted each model's micro-average AUC curve in one AUROC figure to visualize each curve and compare them with each other.

For the Random Forest Model, I got an AUC score of about 0.897. This means that the model has a strong predictive performance to predict the music genre using Random Forest. For the Decision Tree Model, I got an AUC score of about 0.658. This means that the model's predictive performance to predict the music genre is moderate using Decision Tree. For AdaBoost, I got an AUC score of about 0.824, meaning that this model's predictive performance is strong. The figure that shows each AUROC curve visualizes these outcomes compared to Random Guess. The Random Forest curve is smooth and nearly perfect. The Decision Tree curve is a linear line that goes to about (0.08, 0.4), then suddenly turns right to (1, 1). The AdaBoost curve starts curving right, then turns left and continues curving. Overall, the Random Forest model has a better predictive performance with **Micro-Average AUC = 0.897**.



AUROC Figure showing each model's Micro-Average AUC Curves

Dimensionality Reduction Clustering Plot per Genre

I used the full filtered dataset for this clustering plot to illustrate the distribution for each genre. While training the models, I used a train-test split to avoid any leakage and overfitting, but visualizing all samples together in a 2D Dimensionality Reduction Clustering Plot has no impact on model evaluation. In this case, I applied the same preprocessing steps as I did with pipelines and PCA, except using the full dataset variables X and y. Then I projected the resulting PCA feature into a 2D Dimensionality Reduction Plot and color coded each point by genre. The plot shows that most genres form tight clusters along PC2 between approximately -2.5 and 2.5, with a few outliers above and below that band. Along PC1, Classical Music forms a tight cluster that extends to the left – up to -6 – and the other genres have narrower clusters between -2 and 2. I think that this clustering shows how PCA captures distinct audio feature differences like tempo and energy, such that it separates low-energy genres like Classical from high-energy genres. The most important factor that underlies this classification's success is the ability for non-linear

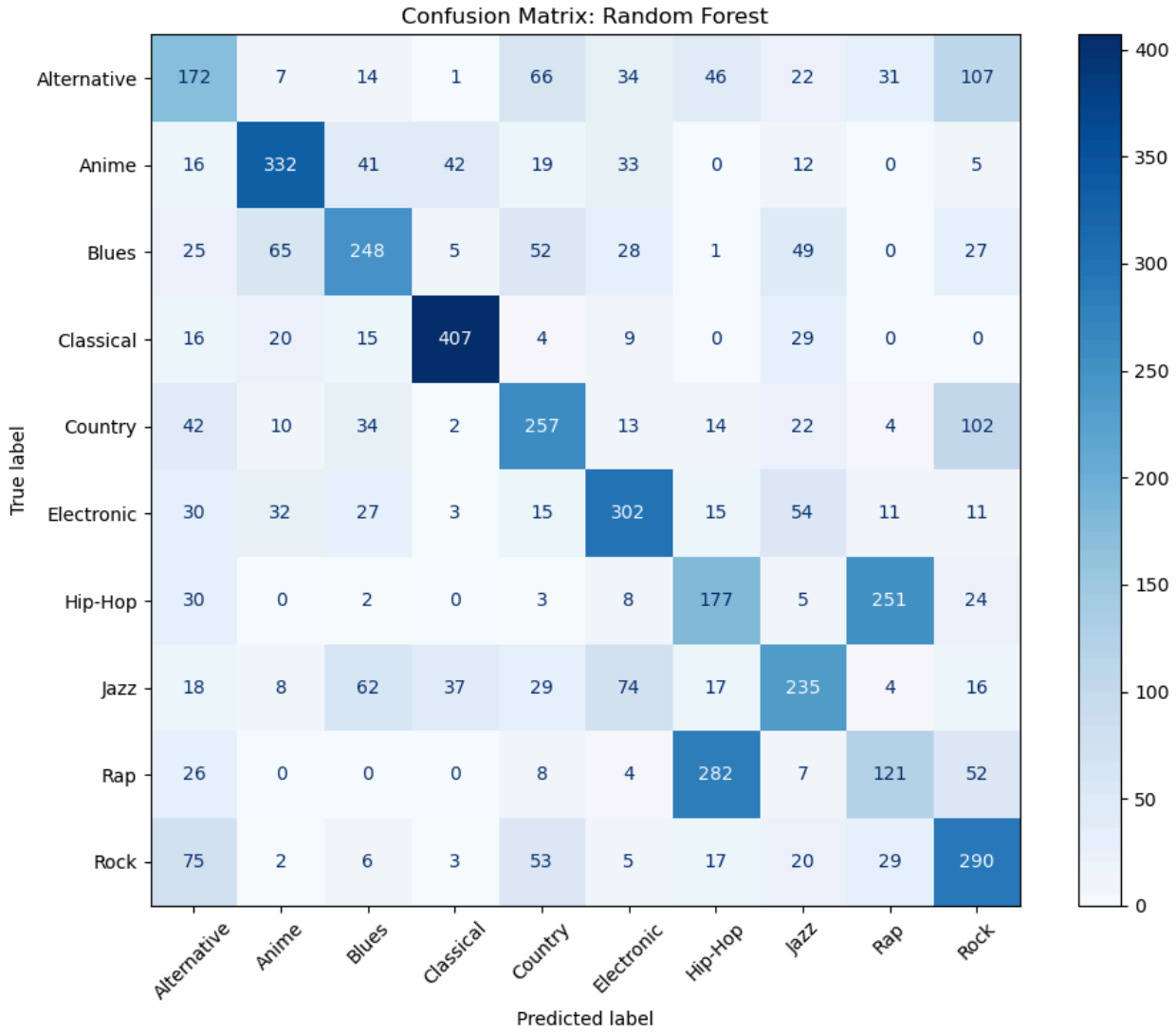
dimensionality reduction models, Random Forest and AdaBoost, to exploit these axes and features in a lower dimensionality space such that they can figure out genres whose clusters overlap in 2D space.

Dimensionality Reduction Cluster Plot of First 2 Principal Components per Genre

Extra Credit

I made predictions using the test data and created a Confusion Matrix for the best model, Random Forest, to compare the predicted genres with true labels. This visualization is interesting in a 10-class setting where there are more incorrect choices than in binary classification. From the matrix, the model correctly classifies 2541 of 5000 samples: 172 Alternative, 332 Anime, 248 Blues, 407 Classical, 257 Country, 302 Electronic, 177 Hip-Hop, 235 Jazz, 121 Rap, and 290 Rock. The model still has confusions. 282 Rap labels were predicted as Hip-Hop, 251 Hip-Hop labels predicted as Rap, 107 Alternative predicted as Rock, 102 Country predicted as

Rock. The other incorrect combinations have smaller numbers. While just over half of the test sample is predicted correctly, the micro-averaged AUC score is still high. AUC measures the model's ability to rank correct genres higher than incorrect ones, not only exact matches. In a multi-class problem, a model can perform well in terms of the AUC score, even if its top-1 prediction is occasionally wrong. These results suggest that despite some genre overlap, Random Forest still captures the overall structure of the data effectively and remains a strong predictive model — especially when evaluated with ranking-based metrics like AUC.



Confusion Matrix for Random Forest Model showing Number of Correct vs. Incorrect Predictions per Genre