Appendix

i. Our exploratory data analysis involved visualizing feature distributions and pairwise correlations to understand the relationships within the data. We split the dataset into training and testing sets using an 80/20 split to ensure robust model evaluation.

ii. During preprocessing, categorical variables such as explicit were encoded as binary values, and numerical features were standardized to align their ranges. We first dropped entries in the dataframe with the same track_id and track_name, but different genres to avoid confusing the model. Since the dataset was alphabetical based on genre, we first shuffled the rows and then kept the first entry for each duplicate song. We also filtered the dataset to include only the top 40 genres, reducing class imbalance and simplifying the classification task.

iii. Linear regression was not directly applicable to the main objective of our project, which focused on classification. However, in an earlier project check-in, we explored linear regression to predict the popularity of songs using audio features such as danceability, energy, and valence. This analysis provided insights into which features were most strongly associated with song popularity. Although this analysis helped us understand relationships within the dataset, it was not leveraged for feature importance in the classification task. Our code was already turned in as part of Project Check-in 2 and the code has not changed

iv. Regression analysis was applied using Logistic Regression to classify tracks by genre. We first applied logistic regression to all the audio features and all the genres, but was only able to get a 25% accuracy rate. We then examined the feature importance using the magnitude of coefficients since the coefficients were standardized already, which helped identify influential features such as instrumentalness, acousticness, and energy. Running logistic regression with only these important features actually dropped the accuracy to 21%. Finally, we decided to cut down the number of genres to 40. This got us to an accuracy of 40%, which is a big improvement from our baseline, but it did require us to cut our dataset by a significant amount. Adding regularization to all of this did not seem to change anything. Overall, the linear nature of Logistic Regression limited its ability to handle non-linear relationships, highlighting its constraints for this dataset.

v. KNN was applied using the KNearestNeighborsClassifier from Scikit-learn. By tuning the neighbor hyperparameter, we achieved a 38% accuracy. We did this by trying different values starting from 5 and going up to 40 in increments of 5 (10 was the ideal value). We used the DecisionTreeClassifier from Scikit-learn to implement decision trees, and by using GridSearchCV, which is also from Scikit-learn, we were able to find the ideal values of the “max_depth,” “min_samples_split,” and “min_samples_leaf” parameters. We were able to achieve a 48% accuracy with this method. Finally, random forest is the key method we chose to go with and the process and results are written on the main document.

vi. PCA and clustering were tricky for the purpose of our project. We initially reduced dimensionality by cutting down genres to 40 to make our model more computationally efficient. We thought this would make more intuitive sense for this project because it didn’t seem ideal to combine features like

speechiness into a select few principle components. We wanted to cut down on dimensionality on our own instead of utilizing dimensionality reduction techniques such as PCA and clustering because we did not want to lose interpretability of our data and features. This initial reduction was enough, as our model accuracy for random forest was not any better with PCA. This tells us that the initial reduction was enough to leave us with the most meaningful features.

vii. We decided to use neural networks as our key methodology. We explained our methods for this and the results from it in our main document. The same inputs were used as the other models. Similar to logistic regression, accuracy was higher when the number of classes were reduced. For the neural network, only the genres that had at least 900 observations after removing duplicates were used, resulting in a total of 41 genres. A neural network consisting of a total of 3 layers, ReLU activation layers, and a dropout layer was used. The output layer has an output size of 41, corresponding to the number of different classes, or genres in our case, when considering genres that have at least 900 observations. On the final model, the neural network obtained training losses ranging from 2.3 to 1.5, validation losses ranging from 2.0 to 1.5, training accuracy ranging from 34% to 53%, and validation accuracy ranging from 41% to 54%. Validation accuracy seemed consistently greater, suggesting that overfitting was not an issue. On the testing set, the model achieved an accuracy of 54.24%, which was similar to the accuracy during the final epochs. Based on the classification report, it was noticeable that the neural network performed better for certain genres, while it did not perform great on other genres. For example, the model had precision = 0.90, recall = 0.86, and f1-score = 0.88 for one class, while it had precision = 0.33, recall = 0.34, f1-score = 0.33 for another class. There are great distinctions in the results of the model evaluation, suggesting that the neural network performs better for specific genres. However, the ROC AUC in terms of one vs. rest was also plotted, and the results displayed very high AUC, with most classes having AUC over 0.90.

viii. Hyperparameter tuning was primarily applied to neural networks, KNN, decision trees, and random forests. For neural networks, we focused on tuning the learning rate and batch size to improve training stability and performance. For KNN, we experimented with different values of k to identify the optimal number of neighbors for classification. For decision trees we experimented with different values of the maximum depth, the minimum number of samples to split, and the minimum number of samples in the leaf nodes. For random forests the values we tuned were the number of trees, maximum depth of the trees, minimum number of samples to split on in the trees, and the minimum number of samples in the leaves of the trees. We evaluated which hyperparameter was most effective through evaluation metrics, such as accuracy. These tuning efforts were crucial in improving model performance and ensuring the selected parameters aligned with the dataset's characteristics. For neural networks, hyperparameter tuning involved changing the learning rate, batch size, number of neurons. We initially started off with a learning rate of 0.1, but after multiple iterations, we found that a learning rate of 0.001 was best for our model. We initially used smaller batch sizes of 16 and continued to increase the batch size, resulting in a final batch size of 128. Finally, the number of neurons in each layer was changed multiple times, using values ranging from 256 neurons to 16 neurons. However, the combination of 128 neurons in the first layer, and 64 in the second hidden layer seemed to work best.