Music Genre Classification System Final Report

Team Members

Adithi Suresh, Annie Pang, Joshua Shin, Millie Kobayashi

Problem Statement

With the growing expansion of the digital music landscape, automating music genre classification has the potential to change the way we discover and recommend music. Current existing methods fall short in providing users with an easier way to discover music that aligns with the listeners' preferences. Therefore, this project addresses this barrier by introducing an automated music genre classification system. By leveraging different markers, including acousticness, danceability, and energy, we aim to create a user-friendly solution that allows listeners to explore and discover new music within their preferred genres effortlessly.

Objective

The primary objective of our project is to develop a robust music genre classification system that effectively categorizes songs based on their inherent features. By leveraging the distinctive markers identified in our methodology, we aim to create a classification framework wherein each song can be accurately described and categorized into its respective genre. Ultimately, the goal is to enhance the music discovery experience for listeners by enabling them to easily explore and find new music that aligns with their tastes and preferences. Through this classification system, users will have access to a curated selection of songs tailored to their individual interests, facilitating a more enjoyable and personalized music listening experience.

Approach

In addressing the challenge of automating music genre classification to enhance the music discovery experience, we propose leveraging a combination of machine learning models tailored to handle the complexities of music data. Each model offers unique advantages and capabilities, which collectively contribute to the development of a robust classification system. Here's why we chose to incorporate the following models:

KNN (K-Nearest Neighbors):

- KNN is an effective algorithm for classification tasks. Its ability to classify data based on similarity makes it suitable for music genre classification, where songs with similar acoustic features are likely to belong to the same genre.

Logistic Regression:

- Logistic Regression is a well-established algorithm for binary classification tasks. By transforming the output into probabilities, Logistic Regression can provide insights into the likelihood of a song belonging to a particular genre, making it valuable in a multi-class classification scenario like music genre classification.

Random Forest:

- Random Forest is an ensemble learning method that combines multiple decision trees to improve classification accuracy. Its ability to handle high-dimensional data and capture complex relationships between features makes it an ideal candidate for music genre classification, where songs may exhibit diverse acoustic characteristics.

ANN (Artificial Neural Network):

- ANN is a versatile deep-learning model capable of learning intricate patterns from data. With multiple hidden layers, ANN can extract abstract features and capture the nuanced characteristics associated with different music genres.

CNN (Convolutional Neural Network):

- CNN is well-suited for processing structured data. Its ability to learn hierarchical features from input data makes it effective for music genre classification.

LSTM (Long Short-Term Memory):

- In the context of music genre classification, LSTM can learn variations, contributing to more accurate genre predictions.

CNN-LSTM:

- By incorporating CNN for feature extraction and LSTM for sequential modeling, CNN-LSTM offers a comprehensive approach to music genre classification.

XG Boost:

- XG Boost is a powerful gradient-boosting algorithm known for its speed and performance. By iteratively improving the model's predictions through boosting, XG Boost can effectively handle imbalanced data and improve classification accuracy, making it a valuable addition to our ensemble of models.

XG Boost and CNN:

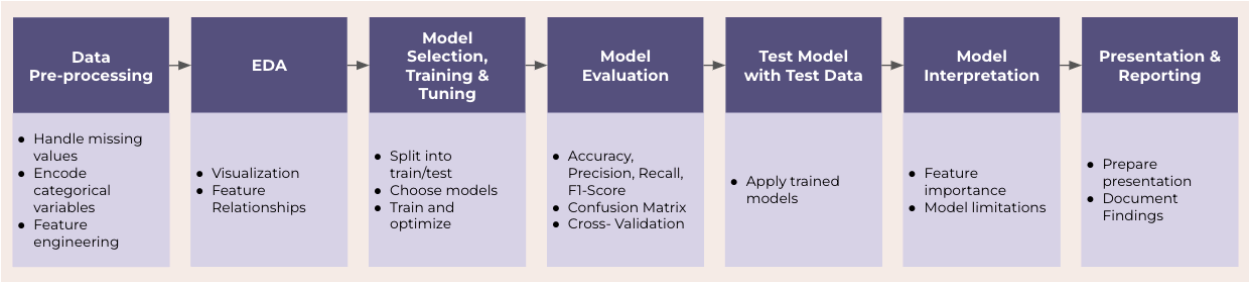
- Powerful model where XGBoost is combined with Convolutional Neural Network model to utilize the strengths of both models in arriving at higher classification accuracy levels.

Transformer Model:

- By adapting transformer architecture to music data, we can leverage its attention mechanism to capture more nuanced relationships between music features, enhancing the classification performance.

For each model, a rigorous process of model selection, training, and tuning is undertaken to optimize performance and ensure generalizability to unseen data. This involves experimenting with various hyperparameters, cross-validating the models, and evaluating their performance metrics on validation datasets. Furthermore, model interpretation techniques are employed to gain insights into the underlying decision-making process and to identify key features driving genre classification decisions.

Block Diagram



Datasets

For this project, the models are built using two datasets. The first dataset is “Prediction of music genre” from Kaggle. This data includes several features, such as acousticness, danceability, energy, etc., for 50,000 songs and categorizes the music genre into ten categories. The second dataset, “Spotify Tracks Dataset,” is also from Kaggle. This dataset contains similar features as the first dataset for 89,741 songs and categorizes the music into 114 genres.

Datasets (Links)

- Prediction of Music Genre
- Spotify Tracks Dataset

Success/Failure

Our success criteria are defined by specific performance benchmarks that indicate the effectiveness and reliability of our music genre classification system:

- Accuracy Threshold: Achieving an accuracy of at least 80% on the test dataset signifies a strong overall performance of the classification models.

In contrast, certain metrics indicate potential shortcomings or deficiencies in our classification system:

- F1-Score: If the F1-Score falls below 0.5, it suggests an imbalance between precision and recall, indicating suboptimal performance in correctly classifying music genres.
- Cross-Validation Performance: A standard deviation in accuracy above 10% during cross-validation indicates instability in model performance and may signal overfitting or insufficient generalization.

Evaluation Parameters

To comprehensively evaluate the performance of our classification models, we consider a range of evaluation parameters:

- Accuracy: Measures the proportion of correctly classified instances, providing an overall assessment of model performance.
- Confusion Matrix: Offers insights into the distribution of true positive, true negative, false positive, and false negative predictions, enabling a deeper understanding of classification errors.
- ROC (Receiver Operating Characteristic): Graphically illustrates the trade-off between true positive rate and false positive rate across different threshold values, helping assess the models' discrimination ability.
- AUC (Area Under the Curve): Quantifies the overall performance of the classification models by computing the area under the ROC curve, with higher values indicating better performance.
- Comparing Loss to Baseline Model: By comparing metrics such as log loss, cross-entropy, or mean squared error (MSE) to those of a baseline model, we can assess the relative improvement or degradation in model performance achieved through our classification system. This comparison provides valuable insights into the efficacy of our approach compared to existing methods.

Experiments and Results

1) **KNN**
2) **Logistic Regression**
3) **Random Forest**
4) **ANN**
5) **LSTM**

6) **CNN**
7) **CNN-LSTM**
8) **XG Boost**
9) **XG Boost Combined with CNN (to make hybrid model)**
10) **Transformer Model**

Model 1: KNN (Classical ML)

The first model we built as the baseline was the K Nearest-Neighbors (KNN) algorithm. While this algorithm is effective for classification problems like ours, it can be computationally costly, especially with larger datasets like the one used in this project. This is because it needs to calculate the distance between a new point and all existing points. However, the simplicity of this model makes it a good starting point.

As a baseline, we created a KNN model with no normalization or feature selection using n_neighbors equal to the square root of the training dataset that comprises 80% of our overall dataset. For data 1, we used n_neighbors equal to 200 and included 24 features: “popularity”, “acousticness”, “danceability”, “duration_ms”, “energy”, “instrumentalness”, “liveness”, “loudness”, “speechiness”, “valence”, “key_A”, “key_A#”, “key_B”, “key_C”, “key_C#”, “key_D”, “key_D#”, “key_E”, “key_F”, “key_F#”, “key_G”, “key_G#”, “mode_Major”, and “mode_Minor.” Although we have a feature for “tempo,” we did not include this because there are 4980 null values, which is around 10% of our data. The training accuracy for our baseline is 0.20555, and the test accuracy is 0.1858. For data 2, we used n_neighbors equal to 301 and included fourteen features: "popularity", "duration_ms", "danceability", "energy", "key", "loudness", "mode", "speechiness", "acousticness", "instrumentalness", "liveness", "valence", "tempo", and "time_signature." The training accuracy for this algorithm is 0.050066, and the test accuracy is 0.038158.

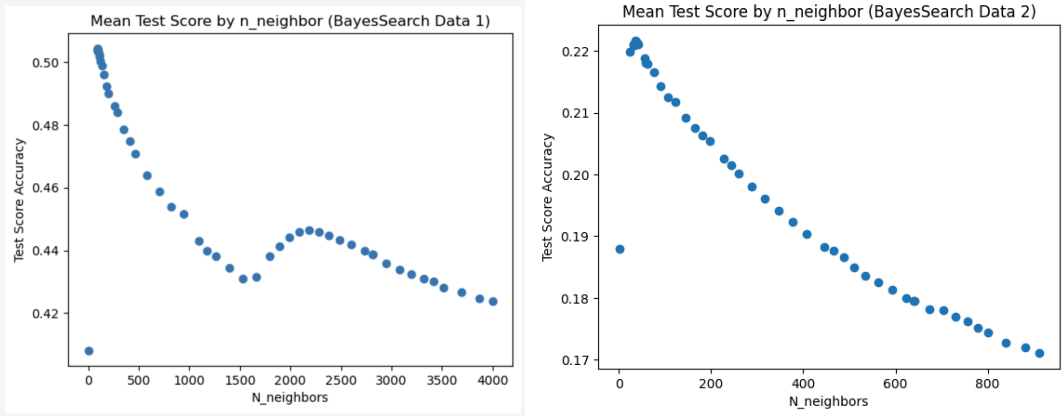
As our first step to improve our baseline, we re-scaled the features using the min-max normalization. We used the min-max normalization over the standardization because not all the features follow the Gaussian distribution. Similar to the baseline, we split 80% of our normalized dataset as training and 20% as test data and created a KNN algorithm using our training data with n_neighbors equal to the square root of the training dataset. We included the same features as our baseline KNN model. Re-scaling the data before running the model significantly improved the accuracy of both the training and test accuracy for data 1 and data 2. The training accuracy for the algorithm using data 1 is 0.4287, and the test accuracy is 0.4082. For data 2, our training accuracy is 0.184816 and our test accuracy is 0.169386.

We were able to further improve the algorithm’s accuracy by conducting feature selection. Unlike complex models, such as neural networks, KNN does not learn to put less weight on features that are not as significant. Therefore, feature selection is crucial to remove noise and

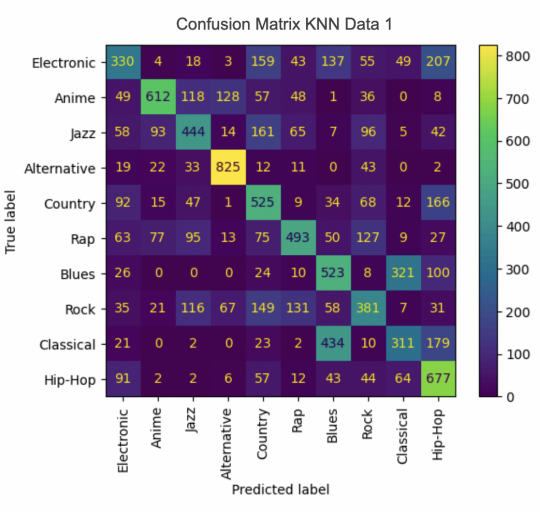
build a more accurate model. To determine the features that are important, we used SequentialFeatureSelector on the KNN algorithm with n_neighbors equal to the square root of the training dataset. For data1, we found that “popularity”, “acousticness”, “danceability”, “duration_ms”, “energy”, “instrumentalness”, “loudness”, “speechiness”, “valence”, “key_D#”, “key_F#”, ad “key_G#” are the important features. The accuracy of the model using just those features for the training dataset was 0.50715 and the test set was 0.4936. For data 2, we found that the most important features are “popularity”, “danceability”, “loudness”, “speechiness”, “acousticness”, “instrumentalness”, and “tempo.” Using these features, our KNN model gave us a training accuracy of 0.21168 and a test accuracy of 0.20105. As the difference between the test and training accuracy is small, in addition to the increased accuracy, this model will generalize well to unseen data. It is important to note that similar features were chosen for both datasets using the SequentialFeatureSelector.

To enhance our KNN algorithm even further, we conducted k-fold cross-validation and Bayesian Search on the data we conducted feature selection on to determine the most appropriate n_neighbors value. For data 1, the best n_neighbors parameter size with BayesSearchCV is 87. This improved the accuracy of the model using data 1 to 0.527775 for the training data and 0.5102 for the test data. For data 2, the best n_neighbors parameter size selected with BayesSearchCV is 34. This also increased the accuracy of the model using data 2 to 0.28239 for the training data and 0.22864 for the test data.

The figure below shows the average accuracy calculated for the test dataset by n_neighbors using Bayes Search for Data 1 and 2.



The figure below shows the confusion matrix for the KNN model built on data 1 that is re-scaled, feature selected, and parameter selected.



The four images below are to compare the classification report of the baseline and the final model for data 1 and 2.

Data 1 Baseline:

	precision	recall	f1-score	support
Alternative	0.15	0.15	0.15	1005
Anime	0.21	0.33	0.25	1057
Blues	0.17	0.10	0.13	985
Classical	0.33	0.36	0.34	967
Country	0.16	0.40	0.23	969
Electronic	0.17	0.10	0.12	1029
Hip-Hop	0.14	0.08	0.10	1012
Jazz	0.17	0.14	0.15	996
Rap	0.16	0.11	0.13	982
Rock	0.18	0.10	0.13	998
accuracy			0.19	10000
macro avg	0.18	0.19	0.17	10000
weighted avg	0.18	0.19	0.17	10000

Data 1 Final Model:

	precision	recall	f1-score	support
Alternative	0.40	0.33	0.36	1005
Anime	0.72	0.55	0.62	1057
Blues	0.55	0.45	0.49	985
Classical	0.77	0.85	0.81	967
Country	0.38	0.58	0.46	969
Electronic	0.62	0.54	0.58	1029
Hip-Hop	0.41	0.51	0.46	1012
Jazz	0.46	0.39	0.42	996
Rap	0.41	0.32	0.36	982
Rock	0.48	0.58	0.52	998
accuracy			0.51	10000
macro avg	0.52	0.51	0.51	10000
weighted avg	0.52	0.51	0.51	10000

Data 2 Baseline:

	precision	recall	f1-score	support
acoustic	0.00	0.00	0.00	214
afrobeat	0.02	0.01	0.01	202
alt-rock	0.01	0.01	0.01	206
alternative	0.03	0.05	0.04	201
ambient	0.00	0.00	0.00	192
anime	0.03	0.01	0.01	199
black-metal	0.02	0.01	0.01	195
bluegrass	0.00	0.00	0.00	178
blues	0.09	0.10	0.09	229
brazil	0.00	0.00	0.00	206
breakbeat	0.01	0.01	0.01	200
british	0.00	0.00	0.00	187
cantopop	0.01	0.00	0.01	222
chicago-house	0.05	0.15	0.07	203
children	0.05	0.08	0.06	201
chill	0.02	0.01	0.02	208
classical	0.13	0.05	0.08	201
club	0.04	0.01	0.02	209
comedy	0.00	0.00	0.00	216
country	0.03	0.09	0.04	209
dance	0.05	0.18	0.07	214
dancehall	0.04	0.08	0.05	192
death-metal	0.00	0.00	0.00	185
accuracy			0.04	22800
macro avg	0.03	0.04	0.03	22800
weighted avg	0.03	0.04	0.03	22800

Data 2 Final Model:

	precision	recall	f1-score	support
acoustic	0.10	0.13	0.11	214
afrobeat	0.22	0.15	0.18	202
alt-rock	0.05	0.05	0.05	206
alternative	0.09	0.10	0.10	201
ambient	0.25	0.27	0.26	192
anime	0.09	0.08	0.08	199
black-metal	0.30	0.43	0.36	195
bluegrass	0.17	0.44	0.25	178
blues	0.18	0.08	0.11	229
brazil	0.09	0.12	0.11	206
breakbeat	0.37	0.34	0.35	200
british	0.08	0.05	0.06	187
cantopop	0.12	0.09	0.10	222
chicago-house	0.39	0.48	0.43	203
children	0.20	0.17	0.18	201
chill	0.16	0.13	0.15	208
classical	0.43	0.52	0.47	201
club	0.32	0.16	0.22	209
comedy	0.89	0.83	0.86	216
country	0.17	0.24	0.20	209
dance	0.21	0.29	0.24	214
dancehall	0.16	0.20	0.18	192
death-metal	0.20	0.25	0.22	185
accuracy			0.23	22800
macro avg	0.22	0.23	0.21	22800
weighted avg	0.22	0.23	0.21	22800

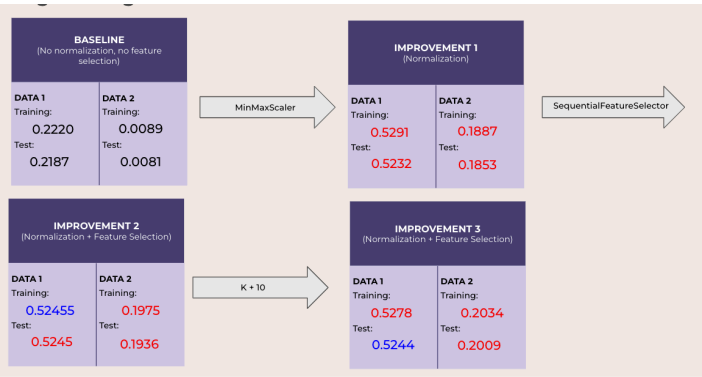
Model 2: Logistic Regression (Classical ML)

For our next model, we utilized logistic regression. This algorithm is widely used for binary classification tasks and is applicable to multi-class scenarios like music genre classification. After preprocessing both datasets by handling missing values, we applied one-hot encoding for data 1 and label encoding for data 2.

The initial test accuracies were 21.9% for data 1 and 0.81% for data 2, which were quite low. To enhance the model's performance, we employed various techniques. Firstly, we applied min-max scaling to normalize the data. Additionally, we implemented feature selection to reduce dimensionality and focus on the most relevant features.

After applying these techniques, there was a significant improvement in accuracy. For data 1, there was a notable increase from the baseline accuracy. However, for data 2, the improvement was minimal.

Furthermore, we experimented with increasing the value of k by 10. While this adjustment had little impact on the accuracy of data 1, data 2 showed only marginal progress toward higher accuracy.



Classification Report for Data 1:

	precision	recall	f1-score	support
Alternative	0.38	0.29	0.33	1005
Anime	0.62	0.59	0.61	1057
Blues	0.50	0.48	0.49	985
Classical	0.76	0.80	0.78	967
Country	0.44	0.56	0.49	969
Electronic	0.61	0.59	0.60	1029
Hip-Hop	0.46	0.47	0.46	1012
Jazz	0.49	0.41	0.45	996
Rap	0.44	0.40	0.42	982
Rock	0.52	0.65	0.57	998
accuracy			0.52	10000
macro avg	0.52	0.52	0.52	10000
weighted avg	0.52	0.52	0.52	10000

Classification Report for Data 2:

	precision	recall	f1-score	support
acoustic	0.10	0.10	0.10	214
afrobeat	0.30	0.15	0.20	202
alt-rock	0.00	0.00	0.00	206
alternative	0.01	0.00	0.01	201
ambient	0.28	0.36	0.32	192
anime	0.02	0.01	0.01	199
black-metal	0.34	0.44	0.38	195
bluegrass	0.23	0.20	0.21	178
blues	0.00	0.00	0.00	229
brazil	0.06	0.01	0.02	206
breakbeat	0.18	0.18	0.18	200
british	0.25	0.01	0.02	187
cantopop	0.15	0.14	0.15	222
chicago-house	0.30	0.42	0.35	203
children	0.25	0.35	0.29	201
chill	0.15	0.14	0.14	208
classical	0.43	0.43	0.43	201
club	0.15	0.05	0.07	209
comedy	0.85	0.84	0.85	216
country	0.13	0.25	0.17	209
dance	0.09	0.07	0.08	214
dancehall	0.11	0.13	0.12	192
death-metal	0.22	0.23	0.22	185
deep-house	0.10	0.19	0.13	208
detroit-techno	0.36	0.45	0.40	201
disco	0.12	0.08	0.10	224
disney	0.30	0.18	0.22	203
reggae	0.09	0.04	0.06	180
reggaeton	0.07	0.06	0.07	213
rock	0.03	0.02	0.02	192
rock-n-roll	0.12	0.13	0.13	183
rockabilly	0.12	0.11	0.11	188
romance	0.37	0.62	0.47	191
sad	0.18	0.35	0.24	187
salsa	0.24	0.51	0.32	198
samba	0.15	0.14	0.15	195
sertanejo	0.28	0.44	0.34	227
show-tunes	0.20	0.13	0.16	216
singer-songwriter	0.08	0.04	0.05	187
ska	0.11	0.13	0.12	194
sleep	0.67	0.67	0.67	215
songwriter	0.09	0.04	0.05	197
soul	0.07	0.05	0.06	172
spanish	0.00	0.00	0.00	230
study	0.50	0.80	0.62	202
swedish	0.00	0.00	0.00	188
synth-pop	0.09	0.15	0.11	179
tango	0.39	0.62	0.48	210
techno	0.18	0.09	0.12	197
trance	0.11	0.13	0.12	166
trip-hop	0.19	0.09	0.13	191
turkish	0.09	0.07	0.08	193
world-music	0.13	0.25	0.17	207
accuracy			0.20	22800
macro avg	0.17	0.20	0.17	22800
weighted avg	0.17	0.20	0.17	22800

By running a classification report function on both datasets, we obtain precision, recall, f1-score, and support values for each class in the dataset. These metrics provide insights into our model's performance across different classes. Our accuracy aligns with what we achieved in our last model improvement.

To provide clarity on these metrics, let's take precision as an example. Precision measures the accuracy of positive predictions for a specific class. For instance, in data 1, let's consider the "country" genre. The model correctly predicts a song to belong to the "country" genre approximately 44% of the time. This indicates how well the model performs in correctly identifying songs belonging to the "country" genre.

Model 3: Random Forest (Ensemble)

For the third model, we employed the Random Forest Classifier, a robust ensemble learning method known for its flexibility and effectiveness in handling various types of data.

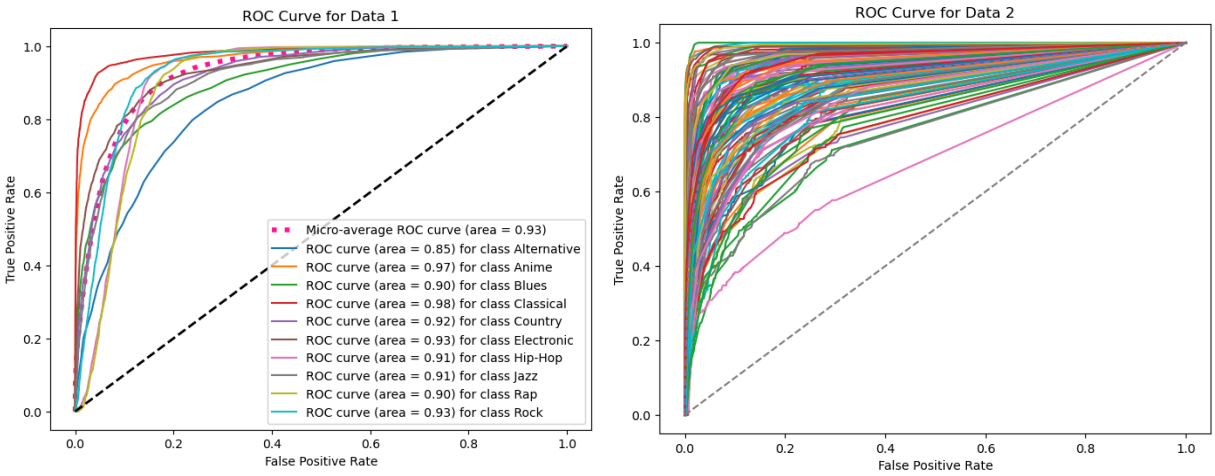
For the baseline model, we utilized a basic Random Forest classifier without any special preprocessing or feature engineering. This baseline model was trained on both datasets independently. To enhance the performance of the model, we experimented with feature scaling using MinMaxScaler in a pipeline. Normalization helps in bringing all feature values to a

common scale, which can be beneficial, especially for distance-based algorithms like Random Forest. By introducing MinMaxScaler to normalize the feature values, we aimed to observe its impact on the model's performance. This experiment was conducted separately for each dataset to discern if normalization provides consistent improvements across different data distributions.

For Data 1, the baseline Random Forest model achieved an accuracy of 56%, with notable variations in precision, recall, and F1-score across different music genres. Similarly, for Data 2, the baseline model attained an accuracy of 34%.

After incorporating feature normalization into the pipeline, the model's performance saw a slight improvement. In Data 1, the accuracy remained unchanged at 56%, suggesting that the impact of normalization was not substantial, indicating that further exploration might be warranted. For Data 2, the accuracy increased to 82%, implying that normalization had a significant effect on the model's performance for this dataset. Looking at the average ROC curve graph, we observed that the baseline model for data 1 has an area under the curve (AUC) of 0.93; while the baseline model for data 2 has 0.91. After improvement, we observed that the improved data 1 model has an area under the curve (AUC) of 0.93; while the improved data 2 model has 1.00. It suggests that the classifier's ability to distinguish between the positive and negative instances for the "acoustic" class is quite good, as the AUC is close to 1.

The experiments demonstrate that while normalization had a minor positive effect on the model's performance for Data 1, it significantly increased the accuracy for Data 2. These findings highlight the importance of considering dataset characteristics and the suitability of preprocessing techniques for optimal model performance. Further investigations, including additional feature engineering and hyperparameter tuning, may be necessary to unlock the full potential of the Random Forest classifier for these datasets.



Figures Above: ROC Curve of Random Forest Classifier Baseline Model Data 1 and Data 2 (Baseline Model 2 with Complete Legend in Appendix)

Accuracy for Data 1: 0.56
Classification Report for Data 1:

	precision	recall	f1-score	support
Alternative	0.44	0.36	0.40	1027
Anime	0.79	0.74	0.77	1032
Blues	0.61	0.55	0.58	1013
Classical	0.82	0.86	0.84	947
Country	0.56	0.58	0.57	995
Electronic	0.65	0.62	0.64	1007
Hip-Hop	0.35	0.39	0.37	1005
Jazz	0.55	0.53	0.54	972
Rap	0.34	0.32	0.33	997
Rock	0.49	0.65	0.56	1004
accuracy			0.56	9999
macro avg	0.56	0.56	0.56	9999
weighted avg	0.56	0.56	0.56	9999

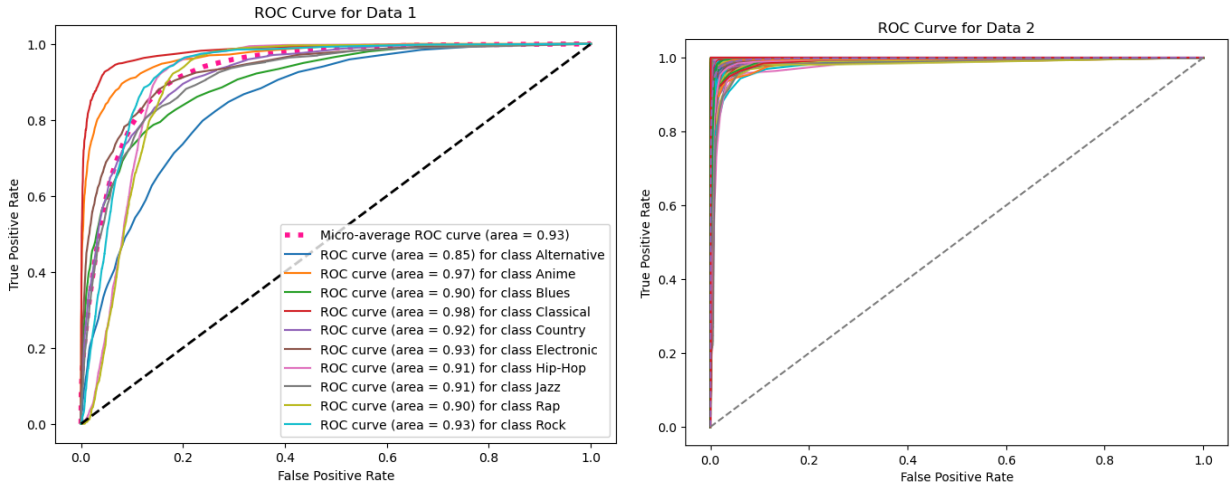
Figure Above: Classification Report of Random Forest Classifier Baseline Model Data 1

Accuracy for Data 2: 0.34

Classification Report for Data 2:

	precision	recall	f1-score	support
acoustic	0.28	0.24	0.26	213
afrobeat	0.36	0.37	0.37	203
alt-rock	0.06	0.06	0.06	215
alternative	0.11	0.13	0.12	184
ambient	0.35	0.35	0.35	197
anime	0.19	0.16	0.17	193
black-metal	0.58	0.65	0.61	210
bluegrass	0.41	0.60	0.49	205
blues	0.17	0.13	0.15	214
brazil	0.04	0.04	0.04	197
breakbeat	0.57	0.50	0.53	199
british	0.17	0.09	0.12	214
cantopop	0.32	0.33	0.32	193
chicago-house	0.57	0.58	0.57	206
children	0.53	0.51	0.52	214
chill	0.27	0.27	0.27	198
classical	0.56	0.56	0.56	198
club	0.37	0.24	0.29	191
comedy	0.89	0.85	0.87	191
country	0.61	0.61	0.61	208
dance	0.37	0.44	0.40	206
dancehall	0.27	0.27	0.27	204
death-metal	0.29	0.40	0.33	191
deep-house	0.17	0.24	0.20	186
detroit-techno	0.58	0.56	0.57	191
disco	0.27	0.21	0.23	189
disney	0.52	0.42	0.46	197
drum-and-bass	0.69	0.69	0.69	199
dub	0.10	0.09	0.10	214
dubstep	0.12	0.15	0.14	212
edm	0.06	0.06	0.06	196
electro	0.26	0.22	0.24	204
electronic	0.08	0.04	0.05	210
emo	0.26	0.28	0.27	197
folk	0.16	0.12	0.14	224
forro	0.39	0.60	0.48	194
french	0.34	0.26	0.30	193
funk	0.33	0.33	0.33	214
garage	0.21	0.16	0.18	201
german	0.31	0.19	0.23	204
gospel	0.30	0.41	0.34	198
goth	0.15	0.08	0.11	215
grindcore	0.86	0.87	0.86	225
groove	0.11	0.07	0.09	213
grunge	0.21	0.22	0.22	188
guitar	0.34	0.30	0.32	191
happy	0.45	0.47	0.46	211
hard-rock	0.11	0.10	0.11	206
hardcore	0.32	0.33	0.32	203
hardstyle	0.50	0.50	0.50	199
heavy-metal	0.33	0.40	0.36	198
hip-hop	0.30	0.33	0.32	186
honky-tonk	0.80	0.81	0.81	203
house	0.14	0.13	0.13	220
idm	0.68	0.48	0.56	213
indian	0.17	0.16	0.16	196
indie	0.09	0.07	0.08	227
indie-pop	0.13	0.11	0.11	208
industrial	0.22	0.19	0.20	194
iranian	0.71	0.85	0.77	194
j-dance	0.41	0.48	0.44	203
j-idol	0.51	0.62	0.56	194
j-pop	0.12	0.10	0.11	212
j-rock	0.10	0.09	0.10	207
jazz	0.45	0.41	0.43	224
k-pop	0.35	0.39	0.37	204
kids	0.62	0.67	0.65	202
latin	0.21	0.22	0.21	204
latino	0.12	0.08	0.10	213
malay	0.26	0.19	0.22	205
mandopop	0.25	0.30	0.27	193
metal	0.14	0.14	0.14	192
metalcore	0.32	0.37	0.34	188
minimal-techno	0.36	0.41	0.38	227
mpb	0.10	0.13	0.11	190
new-age	0.55	0.49	0.52	229
opera	0.40	0.49	0.44	185
pagode	0.43	0.57	0.49	186
party	0.49	0.59	0.54	198
piano	0.45	0.42	0.43	199
pop	0.28	0.30	0.29	179
pop-film	0.30	0.47	0.37	204
power-pop	0.42	0.46	0.44	222
progressive-house	0.25	0.30	0.27	171
psych-rock	0.22	0.17	0.19	199
punk	0.09	0.10	0.09	196
punk-rock	0.07	0.07	0.07	194
r-n-b	0.14	0.10	0.12	194
reggae	0.09	0.11	0.10	188
reggaeton	0.12	0.12	0.12	212
rock	0.31	0.28	0.29	199
rock-n-roll	0.28	0.39	0.33	193
rockabilly	0.29	0.22	0.25	184
romance	0.64	0.85	0.73	201
sad	0.29	0.37	0.32	191
salsa	0.51	0.78	0.62	190
samba	0.37	0.42	0.39	209
sertanejo	0.45	0.57	0.50	196
show-tunes	0.36	0.29	0.32	196
singer-songwriter	0.04	0.05	0.05	179
ska	0.21	0.25	0.23	212
sleep	0.90	0.85	0.88	174
songwriter	0.04	0.04	0.04	196
soul	0.43	0.37	0.40	197
spanish	0.20	0.09	0.12	188
study	0.60	0.88	0.71	182
swedish	0.20	0.12	0.15	176
synth-pop	0.35	0.23	0.27	200
tango	0.70	0.86	0.77	207
techno	0.15	0.13	0.14	197
trance	0.41	0.42	0.42	184
trip-hop	0.29	0.24	0.26	189
turkish	0.26	0.27	0.27	212
world-music	0.37	0.44	0.40	177
accuracy			0.34	22800
macro avg	0.33	0.34	0.33	22800
weighted avg	0.33	0.34	0.33	22800

Figures Above: Classification Report of Random Forest Classifier Baseline Model Data 2



Figures Above: ROC Curve of Random Forest Classifier Improved Model Data 1 and Data 2 (Improved Model 2 with Complete Legend in Appendix)

Accuracy for Data 1: 0.56
Classification Report for Data 1:

	precision	recall	f1-score	support
Alternative	0.40	0.38	0.39	856
Anime	0.78	0.79	0.78	919
Blues	0.60	0.54	0.57	873
Classical	0.86	0.86	0.86	881
Country	0.60	0.59	0.59	921
Electronic	0.67	0.59	0.62	931
Hip-Hop	0.35	0.36	0.35	905
Jazz	0.55	0.53	0.54	888
Rap	0.34	0.33	0.33	914
Rock	0.46	0.61	0.53	916
accuracy			0.56	9004
macro avg	0.56	0.56	0.56	9004
weighted avg	0.56	0.56	0.56	9004

Figure Above: Classification Report of Random Forest Classifier Improved Model Data 1

Accuracy for Data 2: 0.82
Classification Report for Data 2:

	precision	recall	f1-score	support
acoustic	0.99	1.00	0.99	213
afrobeat	0.95	0.97	0.96	203
alt-rock	0.73	0.79	0.76	215
alternative	0.70	0.74	0.72	184
ambient	0.93	0.93	0.93	197
anime	0.91	0.84	0.87	193
black-metal	0.95	0.92	0.93	210
bluegrass	0.93	0.97	0.95	205
blues	0.75	0.71	0.73	214
brazil	0.76	0.87	0.82	197
breakbeat	0.94	0.94	0.94	199
british	0.78	0.68	0.73	214
cantopop	0.82	0.93	0.87	193
chicago-house	0.95	0.95	0.95	206
children	0.87	0.87	0.87	214
chill	0.85	0.85	0.85	198
classical	0.92	0.85	0.88	198
club	0.91	0.76	0.83	191
comedy	0.97	0.90	0.93	191
country	0.91	0.83	0.87	208
dance	0.87	0.90	0.88	206
dancehall	0.87	0.92	0.89	204
death-metal	0.86	0.95	0.90	192
deep-house	0.82	0.88	0.85	186
detroit-techno	0.97	0.98	0.98	194
disco	0.75	0.78	0.76	194
disney	0.87	0.90	0.88	199
drum-and-bass	0.90	0.91	0.90	194
dub	0.50	0.48	0.49	204
dubstep	0.59	0.64	0.61	226
edm	0.71	0.67	0.69	200
electro	0.62	0.73	0.67	207
electronic	0.65	0.55	0.60	200
emo	0.74	0.78	0.76	193
folk	0.53	0.56	0.55	212
forro	0.79	0.97	0.87	205
french	0.68	0.60	0.64	194
funk	0.78	0.68	0.73	218
garage	0.66	0.64	0.65	192
german	0.79	0.63	0.70	196
gospel	0.79	0.92	0.85	212
goth	0.78	0.70	0.74	223
grindcore	0.97	0.99	0.98	226
groove	0.74	0.67	0.70	212
grunge	0.74	0.77	0.75	202
guitar	0.76	0.88	0.82	189
happy	0.85	0.85	0.85	199
hard-rock	0.60	0.55	0.57	185
hardcore	0.72	0.73	0.72	205
hardstyle	0.85	0.89	0.87	210
heavy-metal	0.92	0.95	0.94	198
hip-hop	0.86	0.78	0.82	202
honky-tonk	0.94	0.96	0.95	205
house	0.75	0.76	0.76	230
idm	0.95	0.87	0.91	217
indian	0.51	0.58	0.54	202
indie	0.36	0.33	0.34	212
indie-pop	0.36	0.36	0.36	209
industrial	0.86	0.79	0.82	211
iranian	0.96	0.97	0.96	181
j-dance	0.92	0.83	0.87	213
j-idol	0.95	0.91	0.93	190
j-pop	0.66	0.62	0.64	224
j-rock	0.60	0.66	0.63	196
jazz	0.84	0.74	0.79	213
k-pop	0.89	0.84	0.86	213
kids	0.89	0.91	0.90	198
latin	0.82	0.85	0.84	210
latino	0.81	0.81	0.81	193
malay	0.77	0.79	0.78	189
mandopop	0.86	0.85	0.86	187
metal	0.69	0.78	0.73	186
metalcore	0.93	0.90	0.91	206
minimal-techno	0.95	0.95	0.95	208
mpb	0.71	0.84	0.77	191
new-age	0.93	0.89	0.91	199
opera	0.87	0.88	0.87	189
pagode	0.90	0.93	0.91	192
party	0.90	0.91	0.91	205
piano	0.95	0.78	0.85	197
pop	0.78	0.83	0.80	166
pop-film	0.81	0.86	0.83	213
power-pop	0.90	0.85	0.88	223
progressive-house	0.85	0.89	0.87	191
psych-rock	0.85	0.75	0.80	196
punk	0.61	0.59	0.60	221
punk-rock	0.54	0.61	0.57	192
r-n-b	0.71	0.65	0.68	198
reggae	0.79	0.78	0.79	194
reggaeton	0.84	0.83	0.84	199
rock	0.82	0.86	0.84	205
rock-n-roll	0.79	0.81	0.80	189
rockabilly	0.83	0.82	0.83	200
romance	0.98	0.96	0.97	213
sad	0.96	0.99	0.97	177
salsa	0.92	0.98	0.95	174
samba	0.93	0.91	0.92	189
sertanejo	0.95	0.99	0.97	188
show-tunes	0.96	0.91	0.94	198
singer-songwriter	0.60	0.66	0.63	194
ska	0.89	0.86	0.87	209
sleep	0.99	0.99	0.99	197
songwriter	0.61	0.64	0.62	187
soul	0.89	0.88	0.89	210
spanish	0.95	0.85	0.90	193
study	0.97	1.00	0.99	192
swedish	0.87	0.83	0.85	188
synth-pop	0.92	0.91	0.91	185
tango	0.97	0.99	0.98	184
techno	0.97	0.97	0.97	200
trance	0.97	0.98	0.98	191
trip-hop	0.95	0.98	0.96	183
turkish	0.99	0.95	0.97	185
world-music	0.99	0.99	0.99	188
accuracy			0.82	22800
macro avg	0.82	0.82	0.82	22800
weighted avg	0.82	0.82	0.82	22800

Figure Above: Classification Report of Random Forest Classifier Improved Model Data 2

Model 4: – Artificial Neural Network

For the fourth model, we implemented an Artificial Neural Network (ANN) with Keras. We first created a baseline model with no data rescaling and one Dense hidden layer of dimension 256. The activation function we used for our hidden layer is the Rectified linear unit (ReLU) function because it is the most commonly used activation function. We used the softmax activation for our output layer because it is commonly used for multiclass classification problems. We used the sparse_categorical_crossentropy for our loss function because our output labels are integers and

not vectors. We tried both stochastic gradient descent (SGD) and adam for our optimizer and ran our model for 100 epochs with batch size 64. For data 1, we had a training accuracy of 0.1277 and test accuracy of 0.1244 with SGD. For adam as our optimizer, the training accuracy and test accuracy both increased to 0.1468 and 0.1375, respectively. For data 2, our training accuracy was 0.0091, and test accuracy was 0.0073 for SGD. With adam, our training and test accuracy remained the same. When updating our model, we will continue using adam over SGD because adam increased accuracy for data 1.

To improve our model, we rescaled our data to input into the model using MinMaxScaler. This significantly increased the accuracy for data 1 - 0.6057 for training and 0.5726 for test. For data 2, our training accuracy also significantly increased to 0.3333 and our test accuracy became 0.3075.

To further improve our model, we increased a hidden layer of dimension 128 with an activation function using ReLU. This significantly increased our training accuracy for data 1 to 0.6531, but our test accuracy decreased to 0.5447. For data 2, our training and test accuracy increased to 0.3671 and 0.3152. By increasing a hidden layer, the model decreased the ability to classify new, unseen data for both data 1 and 2.

As the generalizability of our model decreased due to the increase in hidden layers, we halved the dimensions of each hidden layer to 128 for the first and 64 for the second. While this decreased our training accuracy for data 1 to 0.6152, it improved our test accuracy to 0.5724. For data 2, both training and test accuracy decreased to 0.3224 and 0.3046. By halving the dimensions of the hidden layers, the model became better at generalizing to new data.

By adding another hidden layer of dimension 32 and activation function of ReLU, we improved our training accuracy for data 1 to 0.6229, although slightly decreased test accuracy to 0.5654. For data 2, our training and test accuracy decreased to 0.3186 and 0.3029.

The four images below are to compare the classification report of the baseline and the final model for data 1 and 2.

Data 1 Baseline:

Data 1 Final Model:

	precision	recall	f1-score	support
0	0.44	0.02	0.04	1005
1	0.64	0.06	0.12	1057
2	0.52	0.05	0.08	985
3	0.79	0.07	0.12	967
4	0.33	0.06	0.10	969
5	0.69	0.04	0.08	1029
6	0.42	0.02	0.03	1012
7	0.10	0.93	0.18	996
8	0.36	0.06	0.10	982
9	0.52	0.08	0.14	998
accuracy			0.14	10000
macro avg	0.48	0.14	0.10	10000
weighted avg	0.48	0.14	0.10	10000

	precision	recall	f1-score	support
0	0.44	0.36	0.39	1005
1	0.83	0.63	0.71	1057
2	0.54	0.59	0.56	985
3	0.82	0.83	0.83	967
4	0.55	0.51	0.53	969
5	0.64	0.61	0.63	1029
6	0.46	0.38	0.42	1012
7	0.53	0.48	0.51	996
8	0.44	0.55	0.49	982
9	0.49	0.73	0.58	998
accuracy			0.57	10000
macro avg	0.57	0.57	0.56	10000
weighted avg	0.57	0.57	0.56	10000

Data 2 Baseline:

	precision	recall	f1-score	support
0	0.00	0.00	0.00	214
1	0.00	0.00	0.00	202
2	0.00	0.00	0.00	206
3	0.00	0.00	0.00	201
4	0.00	0.00	0.00	192
5	0.00	0.00	0.00	199
6	0.00	0.00	0.00	195
7	0.00	0.00	0.00	178
8	0.00	0.00	0.00	229
9	0.00	0.00	0.00	206
10	0.00	0.00	0.00	200
11	0.00	0.00	0.00	187
12	0.00	0.00	0.00	222
13	0.00	0.00	0.00	203
14	0.00	0.00	0.00	201
15	0.00	0.00	0.00	208
16	0.00	0.00	0.00	201
17	0.00	0.00	0.00	209
18	0.00	0.00	0.00	216
19	0.00	0.00	0.00	209
20	0.00	0.00	0.00	214
21	0.00	0.00	0.00	192
...				
accuracy			0.01	22800
macro avg	0.00	0.01	0.00	22800
weighted avg	0.00	0.01	0.00	22800

Data 2 Final Model:

	precision	recall	f1-score	support
0	0.15	0.21	0.18	214
1	0.44	0.24	0.31	202
2	0.02	0.01	0.01	206
3	0.14	0.03	0.06	201
4	0.31	0.21	0.25	192
5	0.26	0.09	0.13	199
6	0.50	0.48	0.49	195
7	0.38	0.53	0.44	178
8	0.17	0.09	0.12	229
9	0.15	0.16	0.15	206
10	0.49	0.44	0.46	200
11	0.06	0.02	0.03	187
12	0.24	0.23	0.23	222
13	0.51	0.46	0.49	203
14	0.33	0.41	0.36	201
15	0.23	0.40	0.29	208
16	0.58	0.56	0.57	201
17	0.28	0.11	0.15	209
18	0.87	0.84	0.85	216
19	0.22	0.39	0.28	209
20	0.16	0.24	0.20	214
21	0.25	0.17	0.20	192
...				
accuracy			0.30	22800
macro avg	0.29	0.30	0.28	22800
weighted avg	0.29	0.30	0.28	22800

Model 5: – Long Short-Term Model

For our fifth model, we built a long short-term model (LSTM). We started by creating a baseline model with an LSTM layer of dimension 256 and a Dense layer of 10 using activation function of softmax. Similar to the previous models, we used Softmax because it is a commonly used function for multiclass classification problems and sparse categorical cross entropy because our labels are integers and not vectors. As previous models showed that Adam was the better optimizer, we used Adam for our baseline optimizer. For our baseline model, we had a training accuracy of 0.9367 and test accuracy of 0.4794 for data 1. Our training accuracy was 0.7860 and test accuracy was 0.2413 for data 2.

Our baseline model shows high training accuracy, but very low test accuracy for both data 1 and 2, which means that the model is overfitting to the training dataset. Therefore, we halved the dimension of our LSTM layer to 128. While our training accuracy decreased to 0.7712 with this change, our test accuracy increased to 0.5023 for data 1. Similarly, for data 2, our training accuracy decreased to 0.4958, but our test accuracy increased to 0.2646.

As our next attempt to improve our model, we applied the MinMaxScaler to our data before running it through our model. With this step, our training and test accuracy increased to 0.7856 and 0.5095 for data 1. For data 2, our training accuracy decreased to 0.4836, but our test accuracy increased to 0.2733.

To further improve our model by incorporating past and future data, we implemented bidirectional LSTM with the same parameters as the previous model. With this implementation, our training accuracy decreased to 0.6003, but our test accuracy increased even further to 0.5665 for data 1. Similar to data 1, using data 2, our training accuracy also decreased and became 0.3472, but our test accuracy increased and became 0.3065. With the bidirectional LSTM, we were able to improve the generalizability of our models.

As an attempt to further improve our model and avoid overfitting, we included Dropout of 0.1 for random neurons to be dropped out. By including dropout, our training accuracy and test accuracy became lower to 0.5916 and 0.5655 for data 1. The training and test also lowered for data 2 to 0.3381 and 0.3028. Therefore, our final model will be the Bidirectional LSTM without the dropout.

The four images below are to compare the classification report of the baseline and the final model for data 1 and 2.

Data 1 Baseline:

	precision	recall	f1-score	support
0	0.32	0.33	0.32	1005
1	0.70	0.65	0.68	1057
2	0.47	0.43	0.45	985
3	0.77	0.78	0.77	967
4	0.41	0.46	0.44	969
5	0.54	0.54	0.54	1029
6	0.37	0.33	0.35	1012
7	0.43	0.41	0.42	996
8	0.34	0.37	0.35	982
9	0.45	0.48	0.47	998
accuracy			0.48	10000
macro avg	0.48	0.48	0.48	10000
weighted avg	0.48	0.48	0.48	10000

Data 1 Final Model Without Dropout:

	precision	recall	f1-score	support
0	0.42	0.40	0.41	1005
1	0.73	0.71	0.72	1057
2	0.51	0.59	0.55	985
3	0.83	0.86	0.84	967
4	0.56	0.46	0.51	969
5	0.70	0.54	0.61	1029
6	0.45	0.44	0.45	1012
7	0.53	0.52	0.53	996
8	0.44	0.49	0.47	982
9	0.53	0.65	0.59	998
accuracy			0.57	10000
macro avg	0.57	0.57	0.57	10000
weighted avg	0.57	0.57	0.57	10000

Data 2 Baseline:

Data 2 Final Model Without Dropout:

	precision	recall	f1-score	support
0	0.19	0.15	0.17	214
1	0.32	0.31	0.31	202
2	0.12	0.04	0.06	206
3	0.12	0.07	0.09	201
4	0.31	0.40	0.35	192
5	0.14	0.10	0.12	199
6	0.59	0.51	0.55	195
7	0.44	0.53	0.48	178
8	0.11	0.02	0.04	229
9	0.16	0.29	0.21	206
10	0.37	0.46	0.41	200
11	0.07	0.03	0.04	187
12	0.25	0.20	0.22	222
13	0.49	0.52	0.50	203
14	0.34	0.38	0.36	201
15	0.23	0.19	0.21	208
16	0.51	0.59	0.55	201
17	0.33	0.18	0.23	209
18	0.95	0.84	0.89	216
19	0.21	0.28	0.24	209
20	0.21	0.21	0.21	214
21	0.28	0.26	0.27	192
...				
accuracy			0.30	22800
macro avg	0.29	0.30	0.28	22800
weighted avg	0.29	0.30	0.28	22800

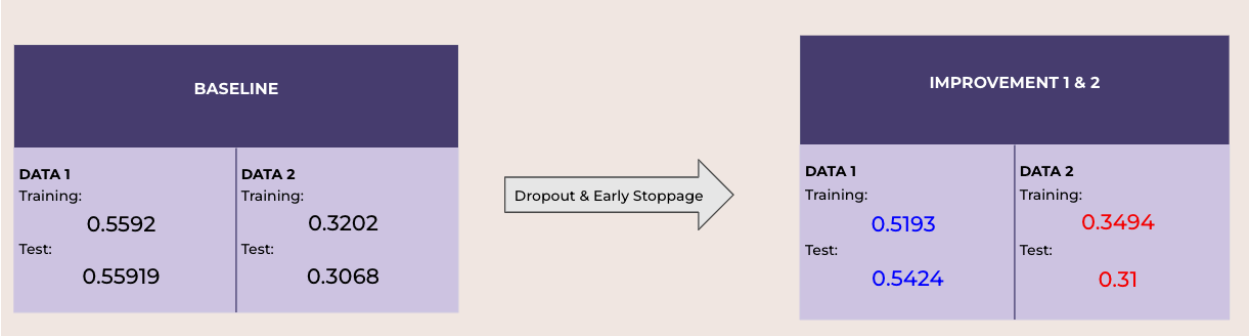
	precision	recall	f1-score	support
0	0.11	0.10	0.10	214
1	0.24	0.22	0.23	202
2	0.07	0.08	0.07	206
3	0.12	0.14	0.13	201
4	0.23	0.26	0.24	192
5	0.11	0.09	0.10	199
6	0.45	0.48	0.47	195
7	0.31	0.37	0.33	178
8	0.15	0.14	0.14	229
9	0.11	0.09	0.10	206
10	0.32	0.40	0.36	200
11	0.07	0.07	0.07	187
12	0.20	0.15	0.17	222
13	0.41	0.36	0.39	203
14	0.29	0.29	0.29	201
15	0.10	0.09	0.09	208
16	0.44	0.55	0.49	201
17	0.19	0.14	0.17	209
18	0.86	0.82	0.84	216
19	0.41	0.36	0.38	209
20	0.33	0.29	0.31	214
21	0.20	0.20	0.20	192
...				
accuracy			0.24	22800
macro avg	0.24	0.24	0.24	22800
weighted avg	0.24	0.24	0.24	22800

Model 6: Convolutional Neural Network

We then explored the use of Convolutional Neural Networks (CNN) for music genre classification, given its suitability for processing structured data. CNNs excel at learning hierarchical features from input data, making them well-suited for this task.

After cleaning and preprocessing both data 1 and data 2 by handling missing values, we applied one-hot encoding for data 1 and label encoding for data 2. With a basic CNN architecture, we achieved test accuracies of 55.9% and 30.7% for data 1 and data 2, respectively. Given the modest accuracies, we implemented several techniques to enhance the model's performance. This included adding dropout layers to introduce regularization and prevent overfitting, as well as employing early stopping to monitor validation loss and mitigate overfitting.

Upon reviewing the next improvement table, we observed a slight decrease in accuracy for data 1, highlighted in blue text, and a slight increase for data 2, indicated in red text. These adjustments aimed to fine-tune the model and optimize its performance on both datasets.



In the CNN section, we examined the precision, recall, f1-score, and support values for each class in the dataset, along with the corresponding accuracy.

To clarify, precision measures the accuracy of positive predictions for a specific class. For instance, consider the "anime" class in data 1. If the model correctly identifies 66% of all instances that truly belong to the "anime" class among all instances predicted as "anime," it achieves a precision of 0.66 for that class.

In practical terms, if there were 100 instances of "anime" in the dataset, and the classifier correctly identified 66 of them as "anime" (true positives), while incorrectly classifying 34 of them as not "anime" (false negatives), then the recall for "anime" would be 0.66. This metric provides insights into the model's ability to capture all instances of a particular class within the dataset.

Classification Report for Data 1:

	precision	recall	f1-score	support
Alternative	0.46	0.27	0.34	1005
Anime	0.74	0.66	0.70	1057
Blues	0.58	0.44	0.50	985
Classical	0.79	0.85	0.82	967
Country	0.53	0.47	0.50	969
Electronic	0.68	0.57	0.62	1029
Hip-Hop	0.39	0.30	0.34	1012
Jazz	0.46	0.52	0.48	996
Rap	0.42	0.63	0.51	982
Rock	0.46	0.73	0.56	998
accuracy			0.54	10000
macro avg	0.55	0.54	0.54	10000
weighted avg	0.55	0.54	0.54	10000

Classification Report for Data 2:

	precision	recall	f1-score	support
0	0.21	0.14	0.17	214
1	0.36	0.14	0.20	202
2	0.15	0.10	0.12	206
3	0.12	0.08	0.09	201
4	0.32	0.40	0.35	192
5	0.21	0.24	0.23	199
6	0.50	0.55	0.52	195
7	0.35	0.56	0.43	178
8	0.16	0.07	0.09	229
9	0.18	0.24	0.21	206
10	0.55	0.42	0.48	200
11	0.10	0.03	0.04	187
12	0.31	0.22	0.26	222
13	0.40	0.60	0.48	203
14	0.38	0.40	0.39	201
15	0.27	0.27	0.27	208
16	0.58	0.56	0.57	201
17	0.40	0.21	0.27	209
18	0.92	0.83	0.88	216
19	0.37	0.33	0.35	209
20	0.29	0.20	0.24	214
21	0.26	0.26	0.26	192
22	0.28	0.32	0.30	185
23	0.21	0.18	0.19	208
24	0.49	0.29	0.36	201
25	0.15	0.08	0.10	224
26	0.32	0.25	0.28	203

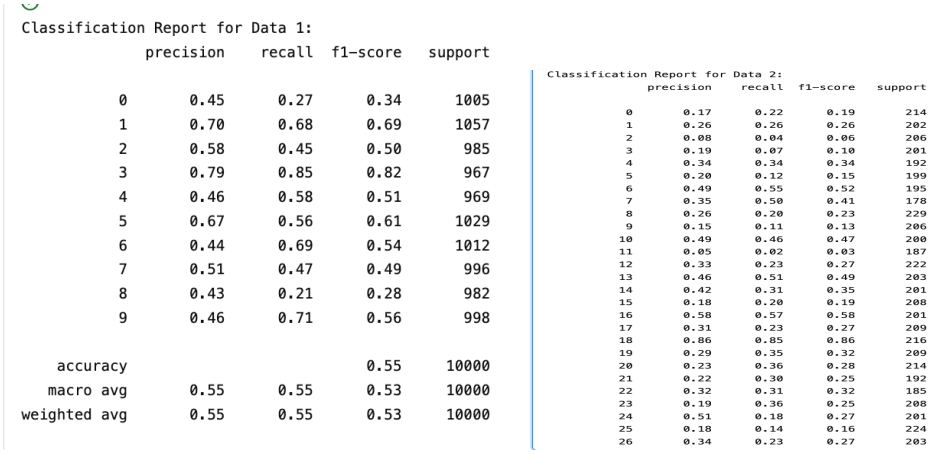
88	0.09	0.07	0.08	180
89	0.18	0.44	0.25	213
90	0.13	0.23	0.17	192
91	0.26	0.40	0.32	183
92	0.27	0.30	0.28	188
93	0.62	0.70	0.65	191
94	0.24	0.28	0.25	187
95	0.49	0.67	0.56	198
96	0.36	0.31	0.34	195
97	0.36	0.68	0.47	227
98	0.26	0.18	0.21	216
99	0.13	0.12	0.12	187
100	0.16	0.21	0.18	194
101	0.86	0.70	0.77	215
102	0.07	0.02	0.03	197
103	0.23	0.34	0.28	172
104	0.20	0.09	0.12	230
105	0.62	0.73	0.67	202
106	0.15	0.06	0.09	188
107	0.19	0.19	0.19	179
108	0.60	0.83	0.69	210
109	0.16	0.05	0.07	197
110	0.27	0.28	0.28	166
111	0.20	0.18	0.19	191
112	0.22	0.44	0.29	193
113	0.23	0.42	0.30	207
accuracy			0.30	22800
macro avg	0.29	0.30	0.29	22800
weighted avg	0.29	0.30	0.29	22800

Model 7: Convolutional Neural Network-Long Short-Term Model

For our next model, we explored a hybrid CNN-LSTM architecture, offering a holistic approach to music genre classification.

After cleaning and preprocessing both data 1 and data 2 by addressing missing values and employing one-hot encoding for data 1 and label encoding for data 2, we attained test accuracies of 54% and 25.9%, respectively. Despite starting with similarly low baseline accuracies as CNN, we sought to determine if there were any notable differences.

As illustrated in the subsequent improvement table, both data 1 and data 2 experienced slight improvements, contrasting with the minor decline observed in data 1 when compared to the original CNN model.



Classification Report for Data 1:

	precision	recall	f1-score	support
0	0.45	0.27	0.34	1005
1	0.70	0.68	0.69	1057
2	0.58	0.45	0.50	985
3	0.79	0.85	0.82	967
4	0.46	0.58	0.51	969
5	0.67	0.56	0.61	1029
6	0.44	0.69	0.54	1012
7	0.51	0.47	0.49	996
8	0.43	0.21	0.28	982
9	0.46	0.71	0.56	998
accuracy			0.55	10000
macro avg	0.55	0.55	0.53	10000
weighted avg	0.55	0.55	0.53	10000

Classification Report for Data 2:

	precision	recall	f1-score	support
0	0.17	0.22	0.19	214
1	0.26	0.26	0.26	202
2	0.08	0.04	0.06	206
3	0.19	0.07	0.10	201
4	0.34	0.34	0.34	192
5	0.20	0.12	0.15	199
6	0.49	0.55	0.52	195
7	0.35	0.50	0.41	178
8	0.26	0.20	0.23	229
9	0.15	0.11	0.13	206
10	0.49	0.46	0.47	200
11	0.05	0.02	0.03	187
12	0.33	0.23	0.27	222
13	0.46	0.51	0.49	203
14	0.42	0.31	0.35	201
15	0.18	0.20	0.19	208
16	0.58	0.57	0.58	201
17	0.31	0.23	0.27	209
18	0.86	0.85	0.86	216
19	0.29	0.35	0.32	209
20	0.23	0.36	0.28	214
21	0.22	0.30	0.25	192
22	0.32	0.31	0.32	185
23	0.19	0.36	0.25	208
24	0.51	0.18	0.27	201
25	0.18	0.14	0.16	224
26	0.34	0.23	0.27	203
88	0.09	0.07	0.08	180
89	0.18	0.44	0.25	213
90	0.13	0.23	0.17	192
91	0.26	0.40	0.32	183
92	0.27	0.30	0.28	188
93	0.62	0.70	0.65	191
94	0.24	0.28	0.25	187
95	0.49	0.67	0.56	198
96	0.36	0.31	0.34	195
97	0.36	0.68	0.47	227
98	0.26	0.18	0.21	216
99	0.13	0.12	0.12	187
100	0.16	0.21	0.18	194
101	0.86	0.70	0.77	215
102	0.07	0.02	0.03	197
103	0.23	0.34	0.28	172
104	0.20	0.09	0.12	230
105	0.62	0.73	0.67	202
106	0.15	0.06	0.09	188
107	0.19	0.19	0.19	179
108	0.60	0.83	0.69	210
109	0.16	0.05	0.07	197
110	0.27	0.28	0.28	166
111	0.20	0.18	0.19	191
112	0.22	0.44	0.29	193
113	0.23	0.42	0.30	207
accuracy			0.30	22800
macro avg	0.29	0.30	0.29	22800
weighted avg	0.29	0.30	0.29	22800

Model 8: XGBoost Model

For our eighth model in this music genre classification system, we created an XGBoost (Extreme Gradient Boosting) model. This model was used as a means of developing a more thorough and robust music classification system. The XGBoost model is valued primarily for its extreme efficiency and effectiveness when dealing with structured data. Some of the key advantages of using this model include its precision and accuracy, which is crucial when combating the classification tasks associated with our music genre classification system. Some of the primary features and techniques used in our XGBoost implementation include feature importance analysis and handling missing values. Feature importance analysis allows us to identify which features are most significant when determining music genre. This then provides us with insights into the dataset and the model's decision-making process. Another feature used is handling missing values. XGBoost can handle missing data and values, which makes the model effective at training without additional steps to manage the missing values.

Our baseline model for XGBoost was set up with default parameters in order to establish a performance benchmark. Based on the results we derived from this baseline model, we arrived at an accuracy rate of 58.1% for data 1 and an accuracy rate of 32.4% for data 2. When we created our improved model, we utilized GridSearchCV as a means of optimizing the XGBoost model. Here we applied GridSearchCV to fine-tune parameters such as 'learning_rate', 'max_depth', and 'n_estimators'. The improvements proved to be significant, as we derived new accuracy rates that were slightly higher than the ones we derived from the baseline model. Here we arrived at an accuracy rate of 59.6% for data 1 and 62.2% for data 2. The hyperparameters we utilized for the improved model include 'learning_rate' of 0.05, 'max_depth' of 5 and 'n_estimators' at 300.

While our XGBoost model didn't meet the accuracy threshold we defined for success, 80%, our model did showcase its ability to handle the complex task of music genre classification as well as its ability to use hyperparameter tuning as a means of improving accuracy rates.

Accuracy for Data 1: 0.5807580758075808
Classification Report for Data 1:

	precision	recall	f1-score	support
0	0.46	0.41	0.43	1027
1	0.83	0.74	0.78	1032
2	0.63	0.57	0.60	1013
3	0.85	0.86	0.85	947
4	0.58	0.60	0.59	995
5	0.69	0.65	0.67	1007
6	0.38	0.38	0.38	1005
7	0.55	0.56	0.55	972
8	0.37	0.38	0.37	997
9	0.52	0.69	0.60	1004
accuracy			0.58	9999
macro avg	0.59	0.58	0.58	9999
weighted avg	0.59	0.58	0.58	9999

Baseline Model Data 1

Accuracy for Data 2: 0.32394736842105265
Classification Report for Data 2:

	precision	recall	f1-score	support
0	0.26	0.20	0.23	213
1	0.35	0.33	0.34	203
2	0.08	0.07	0.08	215
3	0.13	0.15	0.14	184
4	0.31	0.34	0.33	197
5	0.19	0.18	0.19	193
6	0.59	0.58	0.58	210
7	0.47	0.53	0.50	205
8	0.15	0.14	0.15	214
9	0.09	0.11	0.10	197
10	0.51	0.44	0.47	199
11	0.10	0.05	0.07	214
12	0.28	0.26	0.27	193
13	0.57	0.59	0.58	206
14	0.58	0.46	0.51	214
15	0.22	0.22	0.22	198
16	0.51	0.55	0.53	198
17	0.31	0.27	0.29	191
18	0.90	0.84	0.87	191
19	0.64	0.57	0.60	208
20	0.38	0.44	0.41	206

accuracy			0.32	22800
macro avg	0.32	0.33	0.32	22800
weighted avg	0.32	0.32	0.32	22800

Baseline Model Data 2

```
Best parameters for Data 1: {'learning_rate': 0.05, 'max_depth': 5, 'n_estimators': 300}
Accuracy for Data 1: 0.5960596059605960596
Classification Report for Data 1:
              precision    recall  f1-score   support

           0       0.50      0.39      0.44      1027
           1       0.83      0.74      0.78      1032
           2       0.64      0.56      0.60      1013
           3       0.85      0.86      0.85       947
           4       0.59      0.62      0.61       995
           5       0.68      0.65      0.67      1007
           6       0.42      0.43      0.43      1005
           7       0.56      0.55      0.55       972
           8       0.42      0.42      0.42       997
           9       0.52      0.76      0.62      1004

    accuracy                           0.60      9999
   macro avg       0.60      0.60      0.60      9999
weighted avg       0.60      0.60      0.60      9999
```

Improved Model Data 1

```
Accuracy for Data 2 subset: 0.622
Classification Report for Data 2 subset:
              precision    recall  f1-score   support

           0       0.46      0.46      0.46       173
           1       0.51      0.54      0.52       179
           2       0.62      0.62      0.62       226
           3       0.54      0.53      0.53       203
           4       0.55      0.62      0.58       214
           5       0.54      0.61      0.57       199
           6       0.93      0.90      0.92       211
           7       0.67      0.76      0.71       202
           8       0.48      0.41      0.44       208
           9       0.57      0.57      0.57       204
          10       0.65      0.56      0.60       194
          11       0.74      0.70      0.72       203
          12       0.86      0.79      0.83       204
          13       0.45      0.37      0.41       209
          14       0.44      0.43      0.44       194
          15       0.76      0.81      0.79       193
          16       0.71      0.78      0.75       202
          17       0.92      0.94      0.93       200
          18       0.46      0.49      0.47       194
          19       0.52      0.48      0.50       188

    accuracy                           0.62      4000
   macro avg       0.62      0.62      0.62      4000
weighted avg       0.62      0.62      0.62      4000
```

Improved Model Data 2

Model 9: XGBoost and CNN Hybrid Model

Given that we were able to successfully create both CNN and XGBoost models, we also decided to create a hybrid model, combining XGBoost with a Convolutional Neural Network (CNN).This combined model was aimed at leveraging CNN's most attractive feature: detailed feature

extraction, along with XGBoost's useful classification abilities. Feature extraction using CNN is performed as the CNN model extracts detailed features from the data. This can include patterns, for example, that may not have been apparent otherwise. To further the power of this combined model, the features extracted from the CNN model are then used as inputs into XGBoost for further classification. This method of utilizing two different models simultaneously is aimed at enhancing the classification accuracy overall by bringing in the strengths of both models.

Upon setting up the environment with all the necessary libraries for data manipulation, we loaded the dataset using pandas. The rows with missing values were dropped and the features were standardized with the use of 'StandardScaler'. Next we defined the Convolutional Neural Network using 'tensorflow.keras' and then compiled the model using the Adam optimizer and sparse categorical cross entropy loss function. We trained the model for five epochs for both data 1 as well as data 2, where validation was performed in the test data.

```
Epoch 1/5
1250/1250 [==============================] - 15s 11ms/step - loss: 1.4313 -
accuracy: 0.4546 - val_loss: 1.3242 - val_accuracy: 0.5007
Epoch 2/5
1250/1250 [==============================] - 13s 11ms/step - loss: 1.2329 -
accuracy: 0.5270 - val_loss: 1.2126 - val_accuracy: 0.5350
Epoch 3/5
1250/1250 [==============================] - 13s 11ms/step - loss: 1.1916 -
accuracy: 0.5404 - val_loss: 1.1792 - val_accuracy: 0.5455
Epoch 4/5
1250/1250 [==============================] - 13s 11ms/step - loss: 1.1626 -
accuracy: 0.5486 - val_loss: 1.1621 - val_accuracy: 0.5488
Epoch 5/5
1250/1250 [==============================] - 13s 10ms/step - loss: 1.1382 -
accuracy: 0.5576 - val_loss: 1.1429 - val_accuracy: 0.5537
```

Data 1 Training and Validation Accuracy per Epoch and Cross Entropy Loss

```
Epoch 1/5
2850/2850 [==============================] - 40s 14ms/step - loss: 3.3245 -
accuracy: 0.1913 - val_loss: 2.9657 - val_accuracy: 0.2463
Epoch 2/5
2850/2850 [==============================] - 39s 14ms/step - loss: 2.8602 -
accuracy: 0.2641 - val_loss: 2.7793 - val_accuracy: 0.2746
Epoch 3/5
2850/2850 [==============================] - 39s 14ms/step - loss: 2.7075 -
accuracy: 0.2913 - val_loss: 2.6889 - val_accuracy: 0.2948
Epoch 4/5
2850/2850 [==============================] - 39s 14ms/step - loss: 2.6173 -
accuracy: 0.3080 - val_loss: 2.6574 - val_accuracy: 0.2981
Epoch 5/5
2850/2850 [==============================] - 39s 14ms/step - loss: 2.5495 -
accuracy: 0.3185 - val_loss: 2.6054 - val_accuracy: 0.3116
```

Data 2 Training and Validation Accuracy per Epoch and Cross Entropy Loss

In combining CNN with XGBoost, we initialized an XGBClassifier from the ‘xgboost’ library, where the classifier was trained using the CNN features that result from the training data. Based on this, we are able to make predictions regarding the CNN features from the test data. For data 1, the accuracy rate we derived was 61.70%, which is the same or lower than those derived from both the CNN and XGBoost standalone models, indicating that creating this combined model didn’t result in a higher accuracy that would meet our threshold for success.

```
Accuracy for Data 1: 0.6170212765957447
Classification Report for Data 1:
              precision    recall  f1-score   support

           0       0.46      0.50      0.48       962
           1       0.80      0.77      0.78      1033
           2       0.55      0.52      0.54       960
           3       0.56      0.62      0.59      1021
           4       0.65      0.58      0.61      1018
           5       0.57      0.57      0.57      1019
           6       0.75      0.82      0.78       977
           7       0.00      0.00      0.00       107

    accuracy                           0.62      7097
   macro avg       0.54      0.55      0.54      7097
weighted avg       0.61      0.62      0.61      7097
```

Data 1 Results

Model 10: Transformer Model

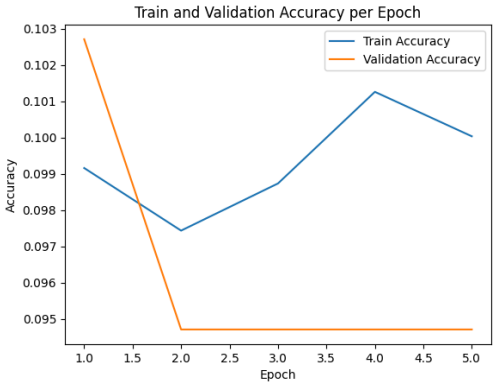
For our tenth model, we built a transformer model. For both “data1” and “data2”, we started by building a baseline model using a Transformer neural network architecture. Initially, we split the data into features and the target variable, removing the target variable 'music_genre_num' from the feature set. The model architecture included an embedding layer, a transformer encoder layer, and a fully connected layer. The model was trained using the Adam optimizer and Cross Entropy Loss criterion for 5 epochs. During training, we calculated both training and validation accuracies to evaluate the model's performance.

For data 1, the baseline model results showed limited accuracy, with training accuracy ranging from approximately 9.9% to 10.1% and validation accuracy around 9.5% to 10.3% across 5 epochs. For data 2, the baseline model results also showed limited accuracy, with training

accuracy ranging from approximately 0.82% to 0.95% and validation accuracy around 0.75% to 0.93% across different epochs.

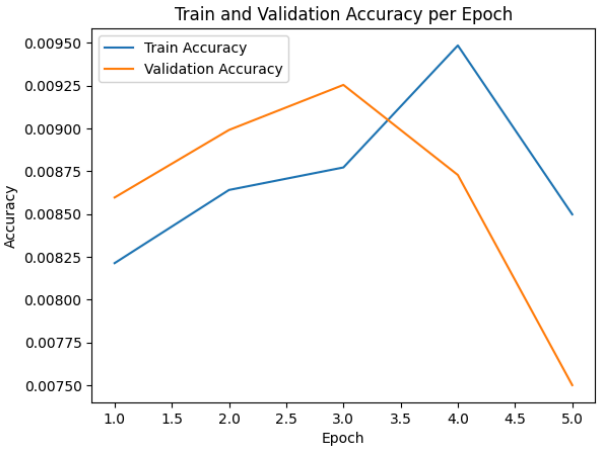
To improve the model's performance, several enhancements were made. First, we selected a subset of relevant features based on their correlation with the target variable. Then, we applied feature scaling using StandardScaler to normalize the selected features. Additionally, we introduced L2 regularization to the model by adding weight decay to the optimizer to prevent overfitting. Hyperparameters such as learning rate, dropout rate, and weight decay were fine-tuned to optimize the model's performance. Class weights were computed and applied to handle class imbalance, ensuring that the model is not biased towards majority classes. Finally, we extended the training duration by increasing the number of epochs to 10 epochs to allow the model more time to learn complex patterns in the data.

The improved model demonstrated significant enhancements in accuracy compared to the baseline. For data 1, training accuracy improved to approximately 44.6% to 48.1%, and validation accuracy increased to about 49.9% to 50.8% across 10 epochs. For data 2, training accuracy improved to approximately 1.05% to 3.19%, and validation accuracy increased to about 0.93% to 2.87% across 10 epochs. These results indicate that the improved model performs substantially better than the baseline model, showcasing the effectiveness of the implemented enhancements in enhancing the model's predictive capability for music genre classification.



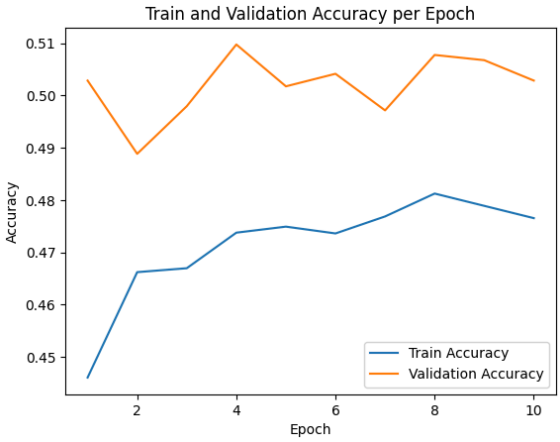

```
Epoch 1/5
Train loss 2.3578872 accuracy 0.0991599
Val   loss 2.3179360 accuracy 0.1027103
Epoch 2/5
Train loss 2.3236991 accuracy 0.0974347
Val   loss 2.3144543 accuracy 0.0947095
Epoch 3/5
Train loss 2.3120693 accuracy 0.0987349
Val   loss 2.3149541 accuracy 0.0947095
Epoch 4/5
Train loss 2.3081996 accuracy 0.1012601
Val   loss 2.3076936 accuracy 0.0947095
Epoch 5/5
Train loss 2.3066860 accuracy 0.1000350
Val   loss 2.3057200 accuracy 0.0947095
```

Figure Above: Training and Validation Accuracy per Epoch Graph and Cross Entropy Loss Table– Transformer Model Baseline Model Data 1



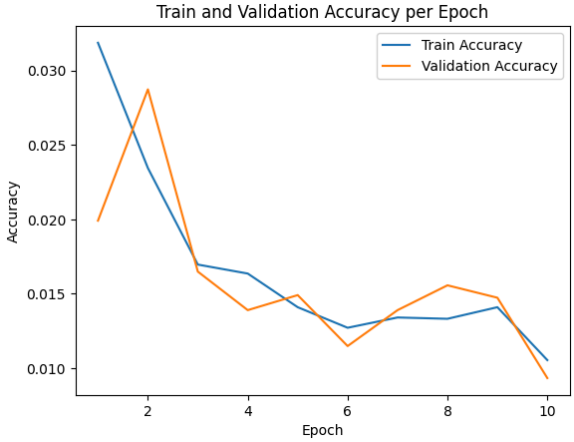

```
Epoch 1/5
Train loss 4.7895087 accuracy 0.0082129
Val   loss 4.7574063 accuracy 0.0085965
Epoch 2/5
Train loss 4.7480230 accuracy 0.0086405
Val   loss 4.7440438 accuracy 0.0089912
Epoch 3/5
Train loss 4.7420022 accuracy 0.0087721
Val   loss 4.7409478 accuracy 0.0092544
Epoch 4/5
Train loss 4.7407748 accuracy 0.0094849
Val   loss 4.7398903 accuracy 0.0087281
Epoch 5/5
Train loss 4.7399979 accuracy 0.0084980
Val   loss 4.7385654 accuracy 0.0075000
```

Figure Above: Training and Validation Accuracy per Epoch Graph and Cross Entropy Loss Table– Transformer Model Baseline Model Data 2




```
Epoch 1/10
Train loss 1.4850932 accuracy 0.4460696
Val   loss 1.3236436 accuracy 0.5028503
Epoch 2/10
Train loss 1.4211544 accuracy 0.4662216
Val   loss 1.3353931 accuracy 0.4888489
Epoch 3/10
Train loss 1.4232043 accuracy 0.4669717
Val   loss 1.3145529 accuracy 0.4979498
Epoch 4/10
Train loss 1.4013639 accuracy 0.4737724
Val   loss 1.3002996 accuracy 0.5097510
Epoch 5/10
Train loss 1.3974236 accuracy 0.4749225
Val   loss 1.3099071 accuracy 0.5017502
Epoch 6/10
Train loss 1.3937568 accuracy 0.4736224
Val   loss 1.2879273 accuracy 0.5041504
Epoch 7/10
Train loss 1.3877345 accuracy 0.4768727
Val   loss 1.3003910 accuracy 0.4971497
Epoch 8/10
Train loss 1.3813889 accuracy 0.4812481
Val   loss 1.2822335 accuracy 0.5077508
Epoch 9/10
Train loss 1.3817808 accuracy 0.4788979
Val   loss 1.2897220 accuracy 0.5067507
Epoch 10/10
Train loss 1.3806878 accuracy 0.4765477
Val   loss 1.2831718 accuracy 0.5028503
```

Figure Above: Training and Validation Accuracy per Epoch Graph and Cross Entropy Loss Table– Transformer Model Improved Model Data 1




```
Epoch 1/10
Train loss 4.5035145 accuracy 0.0318538
Val   loss 4.5287349 accuracy 0.0199123
Epoch 2/10
Train loss 4.5048833 accuracy 0.0234435
Val   loss 4.4010076 accuracy 0.0287281
Epoch 3/10
Train loss 4.6100597 accuracy 0.0169631
Val   loss 4.6313468 accuracy 0.0164912
Epoch 4/10
Train loss 4.6489790 accuracy 0.0163600
Val   loss 4.6766808 accuracy 0.0139035
Epoch 5/10
Train loss 4.6848995 accuracy 0.0141012
Val   loss 4.6744840 accuracy 0.0149123
Epoch 6/10
Train loss 4.7032847 accuracy 0.0127196
Val   loss 4.6955508 accuracy 0.0114912
Epoch 7/10
Train loss 4.7070285 accuracy 0.0134104
Val   loss 4.6934801 accuracy 0.0139035
Epoch 8/10
Train loss 4.7007198 accuracy 0.0133227
Val   loss 4.6777616 accuracy 0.0155702
Epoch 9/10
Train loss 4.6933939 accuracy 0.0141012
Val   loss 4.6809181 accuracy 0.0147368
Epoch 10/10
Train loss 4.7226529 accuracy 0.0105485
Val   loss 4.7374747 accuracy 0.0093421
```

Figure Above: Training and Validation Accuracy per Epoch Graph and Cross Entropy Loss Table– Transformer Model Improved Model Data 2

Discussions (Possible Bias/Fairness):

Based on the models we created as part of this music genre classification system, we learned that there is much potential for biases and issues regarding fairness. Bias can often manifest itself in many different forms, including selection bias for example. Selection bias refers to research bias that is caused by the various methods utilized to select a population of interest as well as sampling methods. This can occur as the datasets used and the model bias that is caused by the training algorithms are breeding grounds for potential selection bias. Given that our models are trained based on datasets that we sourced from platforms such as Kaggle, this can result in selection bias. Kaggle predominantly only includes data regarding Western music genres, for example, which can result in an underrepresentation of non-Western music genres. This limitation can serve as a hindrance, as the model's predictions may be skewed. This ultimately can result in making the models less effective at classifying genres that are not well represented within the training data.

Constraints

In regards to our work, the main constraints we encountered include a limited timeframe to create models with more hyperparameter choices. In the short amount of time we had to complete our project, we weren't able to expand our models even further as much as we wanted to. In addition, another constraint we encountered was limited hardware capabilities. We switched between different platforms to work on our models, including Deepnote, Google Colab, Jupyter Notebook, and VScode. All of these platforms experienced a 'Kernel Timeout Error' when we were attempting to run certain models. This setback caused us to have to switch between different platforms in order to successfully execute our models. Another constraint we encountered was the limited monetary funds we had as a team. Given that access to machines with higher capacities and larger datasets cost more, we weren't able to explore those options and had to make do with the resources we had available to us.

Furthermore, some overall constraints that we have encountered as part of our study have to do with computational resources and data availability. When building such an extensive music genre classification system, the process can be computationally intensive, as it requires processing very large datasets with complex features. This can ultimately serve as a hindrance when evaluating our models and their efficiency. In addition, another constraint we encountered would be limited access to diverse and extensive datasets. Given that our data is sourced from Kaggle, with predominantly Western music genres, this lack of extensive datasets would have likely hindered our models' ability to learn and make generalizations regarding a larger variety of music genres across the globe.

Standards

The main standards by which my team adhered to for the purposes of this project include the use of Python 3.10.9 and the use of features such as sklearn, keras, matplotlib, and pandas. By using these libraries, we were able to effectively create machine learning models in Python. In addition to these standards, our project also adheres to general machine learning standard practices used for classification tasks, such as cross-validation for model assessment, as well as the use of standard metrics within machine learning such as F1-score, accuracy, and AUC used for performance evaluation. While we adhere to these standards, these standards can be considered as ever-evolving, given that music genres are constantly evolving and the fact that guidelines for determining a genre are heavily subjective as well. Due to these factors, standards are relatively flexible.

Comparisons

When comparing our models to existing solutions that are currently present in the market, our approach combines a variety of different machine learning techniques together, ranging from classical algorithms such as KNN and Logistic Regression, as well as advanced deep learning models such as CNNs and LSTMs. This diverse approach towards creating a thorough music genre classification system has proven to be more effective in comparison to that of single-model systems. In addition to these models, we have also leveraged techniques such as Transformers and hybrid models as part of our study. These methods have allowed us to create a path that would aim to surpass the performance of more traditional music classification systems used in the industry.

Limitations of the Study

Outside of the limitations previously mentioned including those in relation to limited time, limited hardware capacities, and limited funds to execute the project, there are some additional limitations posed on our study as well. One such limitation would be data quality and diversity. Presently, while extensive, our current datasets don't fully encompass the entirety of the global music scene. The diversity and quality of the datasets being used directly impact the capabilities of the classification models, which is a current limitation we face as part of this project. Additionally, generalization serves as another limitation. As we know, broader generalizations cannot be made using models that are trained on specific datasets. So, our existing models may not be able to accurately generalize well with other types of music or datasets from sources outside of Kaggle for example. Lastly, interpretability is another limitation of our study. Deep learning models, especially those like CNNs and LSTMs are often hard to interpret, which can serve as a limitation as being able to interpret them is important for most real-world applications.

Future Work

While our music genre classification system is extensive with the use of many different types of models, there are many ways to expand on them as part of future work. One good enhancement would be incorporating more diverse data, for example. This can be achieved through the use of a broader array of music genres from around the world, resulting in a greater amount of training data for the models which would then yield more inclusive and representative models. Model explainability is also a great way to expand our existing models. By developing techniques that make the models more interpretable, we can allow users to understand the classification decisions and the basis on which they were determined more in-depth. Additionally, optimization of the models to be real-time classification systems would also be beneficial in the future. This could include enhancing user experience as well as creating options for use within streaming

services such as Spotify or Apple Music for example. Lastly, in the future, models can be customized to utilize a more user-centric approach. This could look like options to receive user feedback. Based on user feedback, for example, additional features to set preferences that allow for music recommendations and personalization would be extremely beneficial for users.

Appendix

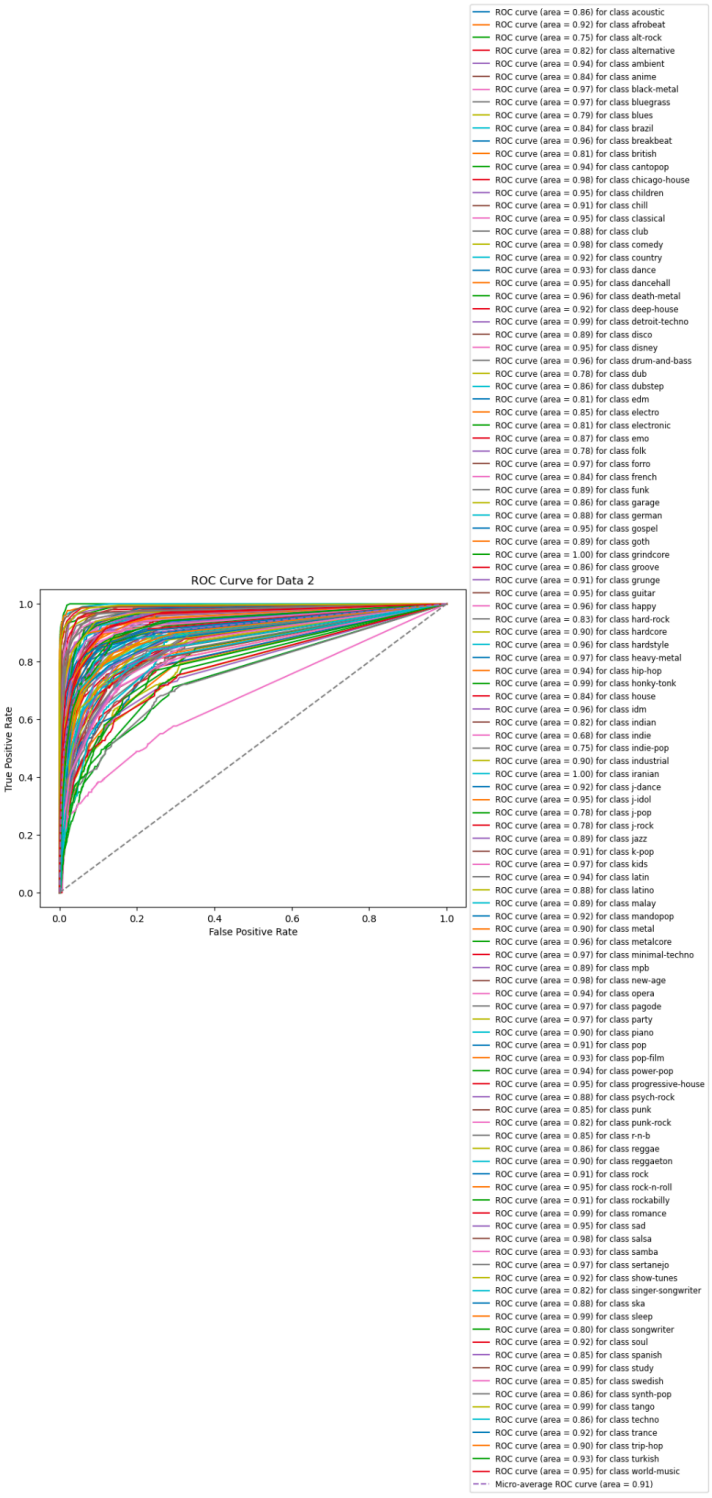


Figure X: ROC Curve of Random Forest Classifier Baseline Model Data 2

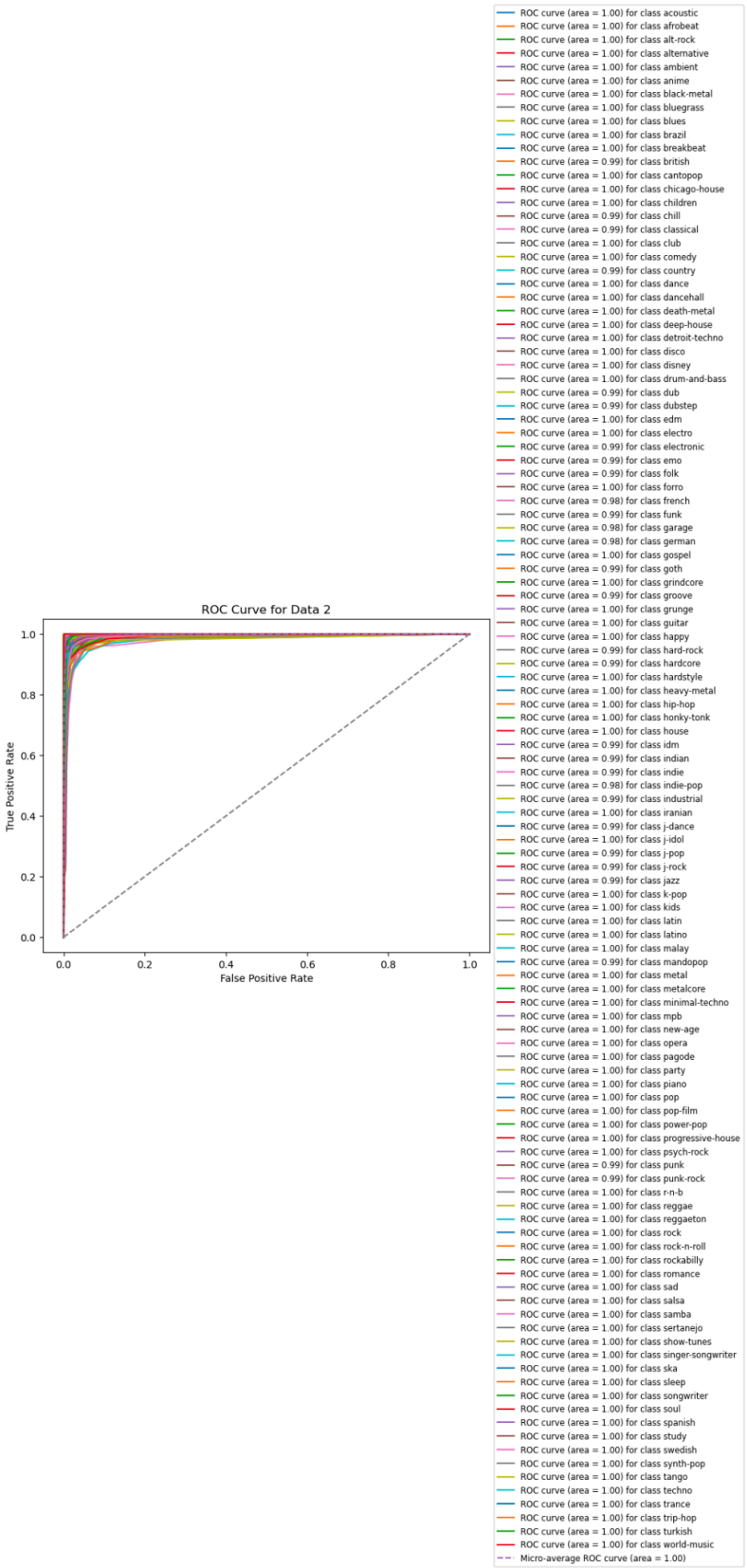


Figure X: ROC Curve of Random Forest Classifier Improved Model Data 2

Accuracy for Data 2: 0.34

Classification Report for Data 2:

	precision	recall	f1-score	support
acoustic	0.28	0.24	0.26	213
afrobeat	0.36	0.37	0.37	203
alt-rock	0.06	0.06	0.06	215
alternative	0.11	0.13	0.12	184
ambient	0.35	0.35	0.35	197
anime	0.19	0.16	0.17	193
black-metal	0.58	0.65	0.61	210
bluegrass	0.41	0.60	0.49	205
blues	0.17	0.13	0.15	214
brazil	0.04	0.04	0.04	197
breakbeat	0.57	0.50	0.53	199
british	0.17	0.09	0.12	214
cantopop	0.32	0.33	0.32	193
chicago-house	0.57	0.58	0.57	206
children	0.53	0.51	0.52	214
chill	0.27	0.27	0.27	198
classical	0.56	0.56	0.56	198
club	0.37	0.24	0.29	191
comedy	0.89	0.85	0.87	191
country	0.61	0.61	0.61	208
dance	0.37	0.44	0.40	206
dancehall	0.27	0.27	0.27	204
death-metal	0.29	0.40	0.33	191
deep-house	0.17	0.24	0.20	186
detroit-techno	0.58	0.56	0.57	191
disco	0.27	0.21	0.23	189
disney	0.52	0.42	0.46	197
drum-and-bass	0.69	0.69	0.69	199
dub	0.10	0.09	0.10	214
dubstep	0.12	0.15	0.14	212
edm	0.06	0.06	0.06	196
electro	0.26	0.22	0.24	204
electronic	0.08	0.04	0.05	210
emo	0.26	0.28	0.27	197
folk	0.16	0.12	0.14	224
forro	0.39	0.60	0.48	194
french	0.34	0.26	0.30	193
funk	0.33	0.33	0.33	214
garage	0.21	0.16	0.18	201
german	0.31	0.19	0.23	204
gospel	0.30	0.41	0.34	198
goth	0.15	0.08	0.11	215
grindcore	0.86	0.87	0.86	225
groove	0.11	0.07	0.09	213
grunge	0.21	0.22	0.22	188
guitar	0.34	0.30	0.32	191
happy	0.45	0.47	0.46	211
hard-rock	0.11	0.10	0.11	206
hardcore	0.32	0.33	0.32	203
hardstyle	0.50	0.50	0.50	199
heavy-metal	0.33	0.40	0.36	198
hip-hop	0.30	0.33	0.32	186
honky-tonk	0.80	0.81	0.81	203
house	0.14	0.13	0.13	220
idm	0.68	0.48	0.56	213
indian	0.17	0.16	0.16	196
indie	0.09	0.07	0.08	227
indie-pop	0.13	0.11	0.11	208
industrial	0.22	0.19	0.20	194
iranian	0.71	0.85	0.77	194
j-dance	0.41	0.48	0.44	203
j-idol	0.51	0.62	0.56	194
j-pop	0.12	0.10	0.11	212
j-rock	0.10	0.09	0.10	207
jazz	0.45	0.41	0.43	224
k-pop	0.35	0.39	0.37	204
kids	0.62	0.67	0.65	202
latin	0.21	0.22	0.21	204
latino	0.12	0.08	0.10	213
malay	0.26	0.19	0.22	205
mandopop	0.25	0.30	0.27	193
metal	0.14	0.14	0.14	192
metalcore	0.32	0.37	0.34	188
minimal-techno	0.36	0.41	0.38	227
mpb	0.10	0.13	0.11	190
new-age	0.55	0.49	0.52	229
opera	0.40	0.49	0.44	185
pagode	0.43	0.57	0.49	186
party	0.49	0.59	0.54	198
piano	0.45	0.42	0.43	199
pop	0.28	0.30	0.29	179
pop-film	0.30	0.47	0.37	204
power-pop	0.42	0.46	0.44	222
progressive-house	0.25	0.30	0.27	171
psych-rock	0.22	0.17	0.19	199
punk	0.09	0.10	0.09	196
punk-rock	0.07	0.07	0.07	194
r-n-b	0.14	0.10	0.12	194
reggae	0.09	0.11	0.10	188
reggaeton	0.12	0.12	0.12	212
rock	0.31	0.28	0.29	199
rock-n-roll	0.28	0.39	0.33	193
rockabilly	0.29	0.22	0.25	184
romance	0.64	0.85	0.73	201
sad	0.29	0.37	0.32	191
salsa	0.51	0.78	0.62	190
samba	0.37	0.42	0.39	209
sertanejo	0.45	0.57	0.50	196
show-tunes	0.36	0.29	0.32	196
singer-songwriter	0.04	0.05	0.05	179
ska	0.21	0.25	0.23	212
sleep	0.90	0.85	0.88	174
songwriter	0.04	0.04	0.04	196
soul	0.43	0.37	0.40	197
spanish	0.20	0.09	0.12	188
study	0.60	0.88	0.71	182
swedish	0.20	0.12	0.15	176
synth-pop	0.35	0.23	0.27	200
tango	0.70	0.86	0.77	207
techno	0.15	0.13	0.14	197
trance	0.41	0.42	0.42	184
trip-hop	0.29	0.24	0.26	189
turkish	0.26	0.27	0.27	212
world-music	0.37	0.44	0.40	177
accuracy			0.34	22800
macro avg	0.33	0.34	0.33	22800
weighted avg	0.33	0.34	0.33	22800

Figure X: Random Forest Classifier Classification Report Baseline Model Data 2

Accuracy for Data 2: 0.82

Classification Report for Data 2:

	precision	recall	f1-score	support
acoustic	0.99	1.00	0.99	213
afrobeat	0.95	0.97	0.96	203
alt-rock	0.73	0.79	0.76	215
alternative	0.70	0.74	0.72	184
ambient	0.93	0.93	0.93	197
anime	0.91	0.84	0.87	193
black-metal	0.95	0.92	0.93	210
bluegrass	0.93	0.97	0.95	205
blues	0.75	0.71	0.73	214
brazil	0.76	0.87	0.82	197
breakbeat	0.94	0.94	0.94	199
british	0.78	0.68	0.73	214
cantopop	0.82	0.93	0.87	193
chicago-house	0.95	0.95	0.95	206
children	0.87	0.87	0.87	214
chill	0.85	0.85	0.85	198
classical	0.92	0.85	0.88	198
club	0.91	0.76	0.83	191
comedy	0.97	0.90	0.93	191
country	0.91	0.83	0.87	208
dance	0.87	0.90	0.88	206
dancehall	0.87	0.92	0.89	204
death-metal	0.86	0.95	0.90	192
deep-house	0.82	0.88	0.85	186
detroit-techno	0.97	0.98	0.98	194
disco	0.75	0.78	0.76	194
disney	0.87	0.90	0.88	199
drum-and-bass	0.90	0.91	0.90	194
dub	0.50	0.48	0.49	204
dubstep	0.59	0.64	0.61	226
edm	0.71	0.67	0.69	200
electro	0.62	0.73	0.67	207
electronic	0.65	0.55	0.60	200
emo	0.74	0.78	0.76	193
folk	0.53	0.56	0.55	212
forro	0.79	0.97	0.87	205
french	0.68	0.60	0.64	194
funk	0.78	0.68	0.73	218
garage	0.66	0.64	0.65	192
german	0.79	0.63	0.70	196
gospel	0.79	0.92	0.85	212
goth	0.78	0.70	0.74	223
grindcore	0.97	0.99	0.98	226
groove	0.74	0.67	0.70	212
grunge	0.74	0.77	0.75	202
guitar	0.76	0.88	0.82	189
happy	0.85	0.85	0.85	199
hard-rock	0.60	0.55	0.57	185
hardcore	0.72	0.73	0.72	205
hardstyle	0.85	0.89	0.87	210
heavy-metal	0.92	0.95	0.94	198
hip-hop	0.86	0.78	0.82	202
honky-tonk	0.94	0.96	0.95	205
house	0.75	0.76	0.76	230
idm	0.95	0.87	0.91	217
indian	0.51	0.58	0.54	202
indie	0.36	0.33	0.34	212
indie-pop	0.36	0.36	0.36	209
industrial	0.86	0.79	0.82	211
iranian	0.96	0.97	0.96	181
j-dance	0.92	0.83	0.87	213
j-idol	0.95	0.91	0.93	190
j-pop	0.66	0.62	0.64	224
j-rock	0.60	0.66	0.63	196
jazz	0.84	0.74	0.79	213
k-pop	0.89	0.84	0.86	213
kids	0.89	0.91	0.90	198
latin	0.82	0.85	0.84	210
latino	0.81	0.81	0.81	193
malay	0.77	0.79	0.78	189
mandopop	0.86	0.85	0.86	187
metal	0.69	0.78	0.73	186
metalcore	0.93	0.90	0.91	206
minimal-techno	0.95	0.95	0.95	208
mpb	0.71	0.84	0.77	191
new-age	0.93	0.89	0.91	199
opera	0.87	0.88	0.87	189
pagode	0.90	0.93	0.91	192
party	0.90	0.91	0.91	205
piano	0.95	0.78	0.85	197
pop	0.78	0.83	0.80	166
pop-film	0.81	0.86	0.83	213
power-pop	0.90	0.85	0.88	223
progressive-house	0.85	0.89	0.87	191
psych-rock	0.85	0.75	0.80	196
punk	0.61	0.59	0.60	221
punk-rock	0.54	0.61	0.57	192
r-n-b	0.71	0.65	0.68	198
reggae	0.79	0.78	0.79	194
reggaeton	0.84	0.83	0.84	199
rock	0.82	0.86	0.84	205
rock-n-roll	0.79	0.81	0.80	189
rockabilly	0.83	0.82	0.83	200
romance	0.98	0.96	0.97	213
sad	0.96	0.99	0.97	177
salsa	0.92	0.98	0.95	174
samba	0.93	0.91	0.92	189
sertanejo	0.95	0.99	0.97	188
show-tunes	0.96	0.91	0.94	198
singer-songwriter	0.60	0.66	0.63	194
ska	0.89	0.86	0.87	209
sleep	0.99	0.99	0.99	197
songwriter	0.61	0.64	0.62	187
soul	0.89	0.88	0.89	210
spanish	0.95	0.85	0.90	193
study	0.97	1.00	0.99	192
swedish	0.87	0.83	0.85	188
synth-pop	0.92	0.91	0.91	185
tango	0.97	0.99	0.98	184
techno	0.97	0.97	0.97	200
trance	0.97	0.98	0.98	191
trip-hop	0.95	0.98	0.96	183
turkish	0.99	0.95	0.97	185
world-music	0.99	0.99	0.99	188
accuracy			0.82	22800
macro avg	0.82	0.82	0.82	22800
weighted avg	0.82	0.82	0.82	22800

Figure X: Random Forest Classifier Classification Report Improved Model Data 2