Music Genre Classification

Griffin Holt, Kavindu Kusal
C S 472 - Machine Learning
Brigham Young University

Abstract

The objective of the experiments is to develop a machine-learning model to classify the genre of a 10-second clip of a song. The possible genres are: Blues, Classical, Country, Disco, Hip-hop, Jazz, Metal, Pop, Reggae, and Rock. The dataset is pulled from the GTZAN public music repository; in total, 73 different input features were extracted from 3000 music samples. Hyperparameter searches, feature reduction experiments, forward selection wrapper experiments, and additional fine-tuning were conducted to improve the classification accuracies of nine different models. The top three performing models were, in order: a Gradient Boost ensemble (86.20%), a Random Forest ensemble (85.70%), and a Multi-layer Perceptron (85.17%). The 'Classical' genre was the easiest to classify; the 'Rock' genre was the most difficult to classify.

1 Introduction

The primary objective of our project is to develop a machine learning model to classify the genre of a 10-second clip of a song. The model will be fed certain audio features extracted from the clip, after which it will classify the song as one of ten possible genres: Blues, Classical, Country, Disco, Hip-hop, Jazz, Metal, Pop, Reggae, and Rock.

2 Methods

2.1 Data Source

We utilized the GTZAN open-source music library which consists of the 10 different genres (identified previously) with 100 different 30-second samples each, for a total of 1000 samples. This GTZAN dataset is made available by G. Tzanetakis and P. Cook [Tzanetakis and Cook, 2002] and is hosted on the Music Analysis, Retrieval and Synthesis for Audio Signals website [MARSYAS, 2015].

To augment this dataset, we split each 30-second clip into three 10-second clips, thereby increasing the total number of samples to 3000. We considered this a reasonable data augmentation method on the assumption that none of the music samples are likely to have an exactly repeated 10-second refrain.

2.2 Feature Extraction

The tempo of each 10-second clip was estimated using Librosa. Thirty-six additional audio features were extracted from rolling-windowed frames of each 10-second clip (with each frame being 1024 samples and the hop length of the rolling windows being 512 frames) using either the Librosa audio processing library or through a function we wrote ourselves. We then took the mean and standard deviation of each of these thirty-six features in order to aggregate them from vector form, resulting in a total of 73 input features extracted from each 10-second audio clip.

These thirty-six extracted audio features are listed in Table 1 as features 2-37. Precise mathematical definitions of each feature are not included for brevity's sake, but can be found in [Peeters, 2004].

The output label ('Genre') was included with the GTZAN dataset [MARSYAS, 2015].

Unknown Values

In the case that one of the 1024-sample frames was "silent" (i.e., no sound occurred during the frame), then the computations of Energy Entropy (Feature No. 4), Band Energy Ratio (Feature No. 6), and Spectral Flux (Feature No. 12) resulted in "divide-by-zero" errors. Only 11 instances (out of the 3000 total instances in our augmented dataset) had such an error occur, 10 of which were labeled as from the "Hiphop" genre. Thus, we decided, in such a case, to assign the mean and standard deviation values for these features to be 0. A zero value for both the mean and standard deviation of these three features does not occur naturally. Thus, the use of 0 for these values gives the machine learning models extra information about these music clips (i.e., that they contain silence is some part of the song which is evidently more common in hip-hop music).

2.3 Selected Models

We used Scikit-Learn as our machine learning framework. We selected nine models with which to experiment, each of which are listed in Table 2.

No.	Feature	Abbrev.	Type	Description
1	Tempo	–	Time	The estimated tempo of the music (in beats per minute, bpm)
2	Amplitude Envelope	AE	Time	The maximum amplitude (infinity-norm) of the signal at each time step
3	Root Mean Square Energy	RMSE	Time	The square root of the mean of then squared energy (two-norm) of the signal at each time step
4	Energy Entropy	EE	Time	The entropy of the energy of the signal at each time step
5	Zero-Crossing Rate (ZCR)	ZCR	Time	The number of times that the signal crosses the zero-amplitude line in a given time step
6	Band Energy Ratio	BER	Frequency	The energy ratio of the different frequency bands
7	Spectral Centroid	SCe	Frequency	A measure of the "center of mass" of the spectrum (calculated via the Fourier Transform) - it is roughly connected to the "brightness" of a sound
8	Spectral Bandwidth	SB	Frequency	An indication of how spread a frequency in the music clip is across the spectrum
9	Spectral Rolloff	SR	Frequency	The frequency below which 99% of the energy of the spectrum is contained
10	Spectral Flatness	SFlat	Frequency	The flatness of the spectrum. It is computed using the ratio between the geometric and arithmetic means
11	Spectral Contrast	SCo	Frequency	The decibel difference between peaks and valleys in the spectrum
12	Spectral Flux	SFlux	Frequency	A measure of how quickly the spectrum of a signal is changing. It is calculated by computing the difference between the current spectrum and that of the previous frame.
13-25	Mel Frequency Cepstral Coefficients	MFCC (1-13)	Frequency-Time	Coefficients that describe the Mel-Frequency Spectrogram
26-37	Chroma Vector	Chroma (1-12)	Frequency-Time	Coefficients that describe the chromagram (a description of pitch) of a sound clip.

Table 1: Complete List of Extracted Audio Features

Model	Scikit-Learn Class
Stochastic Gradient Descent	'SGDClassifier'
Logistic Regression	'LogisticRegression'
Perceptron	'Perceptron'
Multi-layer Perceptron	'MLPClassifier'
Decision Tree	'DecisionTreeClassifier'
Random Forest	'RandomForestClassifier'
Gradient Boosting (Decision Trees)	'GradientBoostingClassifier'
Linear Support Vector Machine	'LinearSVC'
Gaussian Naive Bayes Estimator	'GaussianNB'

Table 2: Selected Machine Learning Models

2.4 Measuring Accuracy

Due to the limited size of our dataset (3000 samples), we used 10-fold cross validation (through Scikit-Learn's 'KFold' function and shuffled with a seed of 42) to measure classification accuracy rather than separating out a single test set.

Nested Cross-Validation

When conducting hyperparameter search or feature selection experiments, we utilized a machine learning technique known as "nested cross-validation". The nested cross-validation procedure is as follows: First, the entire dataset is split using 10-fold cross validation; this is the "outer" cross-validation. For each iteration of the outer cross-validation, the test set is a held-out fold and the training set is the union of the other 9 folds. Then, the hyperparameter search (grid or random) or the feature selection search is run on the training set using 4- or 5-fold cross-validation as an accuracy measurement; this is the "inner" cross-validation. The model that performed the best on the "inner" cross-validation (i.e., the model with the best hyperparameters or the model trained on the selected features) is then evaluated against the test set from the outer cross-validation (i.e., the held-out fold). Performances and models are reported from each of the 10 fold of the outer cross-validation.

In terms of its usefulness, nested cross-validation provides a method by which to reduce bias in hyperparameter tuning and model selection. The search does not have the opportunity to overfit the entire dataset–it is only exposed to a subset of the dataset provided by outer cross-validation [Brownlee, 2021].

3 Initial Results

For each of the nine models, we used 10-cross validation with default hyperparameters in order to get a baseline performance measure. We repeated this procedure three times: once with unnormalized data, once with min-max normalized data, and once with max-absolute normalized data. The results of these experiments are in Table 3.

From these initial results, we can gather a few key observations. First, notice that the two ensemble approaches–Random Forest and Gradient Boost–already achieved classification accuracies of more than 80%. Second, notice that–excluding the tree-based models–normalization drastically improves the performance of the models. Third, we see that *all* of the models (even on unnormalized data) perform well above the baseline accuracy (10%, as the data is evenly balanced across all 10 genres).

Models	No Normalization	Min-Max	Max-Absolute
Baseline Accuracy		10.00%	
SGD	27.17%	56.23%	55.63%
Logistic Regression	46.10%	72.70%	73.10%
Perceptron	27.40%	56.97%	59.70%
MLP	49.13%	69.37%	70.70%
Decision Tree	60.23%	60.90%	60.40%
Random Forest	83.80%	83.73%	83.73%
Gradient Boost	81.30%	81.80%	81.63%
Linear SVM	41.47%	73.97%	73.90%
Gaussian NB	48.53%	56.73%	56.73%

Table 3: Initial model performances

4 Model Experiments

To improve upon the initial results, we conducted four successive sets of experiments: hyperparameter search for all models using the full feature set; hyperparameter search for all models using five different PCA-reduced feature sets; forward feature selection; and fine-tuning of the best models.

4.1 Hyperparameter Search with the Full Feature Set

For each model, we conducted either a grid or random search on a specified hyperparameter space with nested cross-validation using the full feature set (i.e., all 73 features). The hyperparameter spaces (excluding the topologies for the multi-layer perceptron) for each of our nine models are specified in Table 4. The tested topologies for the multi-layer perceptron were: $[146]$, $[292]$, $[584]$, $[730]$, $[146, 146]$, $[292, 292]$, $[584, 584]$, and $[730, 730]$. Each experiment was conducted three times: once with unnormalized data, once with min-max normalized data, and once with max-absolute normalized data.

From this experiment, we had a few important observations. First, the multi-layer perceptron and the stochastic gradient descent both showed a large improvement from their initial experiment by 13.47 percentage points and 15.67 percentage points, respectively. Second, the multi-layer perceptron was able to match the performance of the two ensemble methods–Random Forest and Gradient Boost–joining the group of models that performed with above 80% classification accuracy

Model	Hyperparameter Space
SGD	loss': ['hinge', 'modified_huber', 'squared_hinge', 'squared_loss', 'huber', 'epsilon_insensitive', 'squared_epsilon_insensitive'], 'penalty': ['l2', 'l1'], 'alpha': loguniform(0.0001, 0.1), 'learning_rate': ['constant', 'optimal', 'invscaling', 'adaptive'], 'eta0': loguniform(0.0001, 1)
Logistic Regression	penalty': ['l2', 'l1'], 'C': loguniform(0.01, 100), 'solver': ['newton-cg', 'lbfgs', 'liblinear', 'sag', 'saga'], 'multi_class': ['ovr', 'multinomial'],
Perceptron	penalty': ['l2', 'l1'], 'alpha': loguniform(0.0001, 0.1), 'eta0': loguniform(0.0001, 1)
MLP	activation': ['logistic', 'tanh', 'relu'], 'solver': ['lbfgs', 'sgd', 'adam'], 'alpha': [0.00001, 0.0001, 0.001, 0.01, 0.1], 'learning_rate': ['constant', 'invscaling', 'adaptive'], 'learning_rate_init': [0.00001, 0.0001, 0.001, 0.01, 0.05, 0.1, 0.5, 1, 10], 'max_iter': [100, 200, 500, 1000, 2000], 'momentum': [0.01, 0.05, 0.1, 0.3, 0.5, 0.7, 0.9]
Decision Tree	criterion': ['gini', 'entropy'], 'splitter': ['best', 'random'], 'max_depth': [3, 5, 10, 25, None], 'max_features': ['sqrt', 'log2', None]
Random Forest	n_estimators': [50, 100, 150], 'criterion': ['gini', 'entropy'], 'max_depth': [5, 10, 25, None], 'max_features': ['sqrt', 'log2', None]
Gradient Boost	learning_rate': [0.001, 0.01, 0.1], 'max_depth': [3, 5, 7], 'max_features': ['sqrt', 'log2', None]
Linear SVM	penalty': ['l2', 'l1'], 'loss': ['hinge', 'squared_hinge'], 'max_iter': [1000, 2000], 'C': [0.1, 1, 10]
Gaussian NB	var_smoothing': loguniform(1e-10, 1e-5)"

Table 4: Defined Hyperparameter Spaces (excluding MLP Topology)

4.2 Feature Reduction

For each model, we conducted either a grid or random search on a specified hyperparameter space with nested cross-validation using a PCA-reduced feature set ("PCA" being Principal Component Analysis). The hyperparameter spaces for each of our nine models are the same as those from the full feature set experiment (see Table 4). The tested topologies for the multi-layer perceptron varied depending on the number of principal components. Each experiment was conducted fifteen times, once for each combination of the three normalization techniques (no normalization, min-max, max-absolute) and five different principal component sizes ($n = 1, 2, 5, 10, 15$).

As one might expect, all of the models performed best with $n = 15$ principal components for the PCA-reduced feature set. Interestingly enough, all of the models also performed best with max-absolute normalization.

Although many of the PCA-reduced models did not perform as well as their full feature set counterparts, the PCA-reduced multi-layer perceptron model, the decision tree model, and the random forest model all were remarkably close and the PCA-reduced linear Gaussian Naive Bayes estimator actually performed better (59.77% versus 56.93%).

4.3 Feature Selection: Wrappers

For each model, a forward-selection wrapper was run on each fold of 10-Fold Cross Validation, giving us 10 different sets of 15 selected features for that model. The 15 features with the highest mode (in being selected across all 10-folds) were then selected. 10-Fold Cross Validation was run once more with these selected 15 features in order to obtain a final accuracy measurement. For each model, the normalization technique and hyperparameters used were those with which the model performed the best on the full feature set.

Similar to the PCA-reduction experiments, although many of the wrapper-selected models did not perform as well as their full feature set counterparts, the multi-layer perceptron model and the gradient boost model all were remarkably close. In addition, the Decision Tree, Gradient Boost, and Gaussian Naive Bayes estimator all performed better with their respective wrapper-selected feature sets than they did with the full feature set.

4.4 Fine-tuning

Once the three previous sets of experiments were completed, three models stood out above the rest in terms of performance:

1. the Gradient Boost model, optimized on the full dataset without normalization;
2. the Random Forest model, optimized with the forward-selection wrapper and min-max normalization; and
3. the Multi-Layer Perceptron, optimized on the full dataset with min-max normalization.

We took these three models and attempted to fine-tune them to increase their mean accuracy by whatever amount we could. For the Gradient Boost Model, we played with the maximum depth of the tree. For the Random Forest model, we did this by increasing and decreasing the number of estimators in the ensemble and the number of features considered at each node split until maximum accuracy was achieved. For the Multi-layer Perceptron, we experimented with various topologies (including three hidden layers, which we had excluded from the previous hyperparameter searches).

We also experimented slightly with the single Perceptron model by training it without early stopping.

For all four of these fine-tuning experiments, we were able to increase the classification accuracy (although only by a few percentage points or, in some cases, even less than a percentage point).

5 Final Results

The best performances of each model for each of the five sets of experiments are presented in Figure 1.

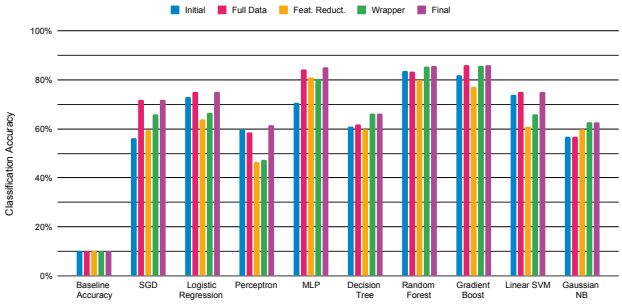

Figure 1: Best model performances across all five sets of experiments

The two models that showed the most improvement from their initial experiment to their final experiment were Stochastic Gradient Descent and the Multi-layer Perceptron. The three best models overall were (in order from highest accuracy to lowest) the Gradient Boost ensemble, the Random Forest ensemble, and the Multi-layer Perceptron; each of these models scored above 85% accuracy. We will expand on each of their final models below:

5.1 Multi-layer Perceptron

The Multi-layer Perceptron achieved a mean accuracy (across 10-fold cross-validation) of 85.17%. This was accomplished by training a multi-layer perceptron with 2 hidden layers (584 nodes each) with a learning rate of $c = 0.005$, an $l2$-regularization parameter of $\lambda = 0.005$, and a momentum of $\alpha = 0.3$ on the full feature set with min-max normalization.

To help us understand the successes and failures of this model, we averaged out the confusion matrices across 10-fold cross validation of a similar high-performing multi-layer perceptron; this mean confusion matrix is displayed in Figure 2.

As we can see from the confusion matrix, the multi-layer perceptron had the highest accuracy rate for the 'Classical' genre and the lowest accuracy rate for the 'Rock' genre. The multi-layer perceptron often confused 'Classical' and 'Jazz'; 'Rock' and 'Country'; 'Blues' and 'Country'; 'Rock' and 'Disco'; and 'Hiphop' and 'Disco'.

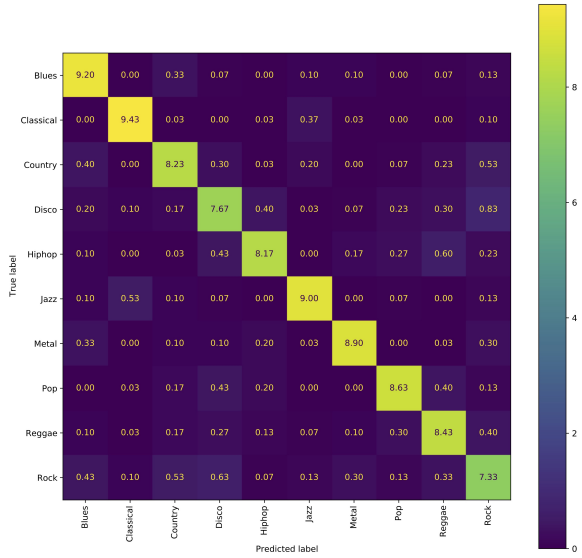

Figure 2: A confusion matrix (averaged over 10-fold cross-validation) for a high-performing MLP model

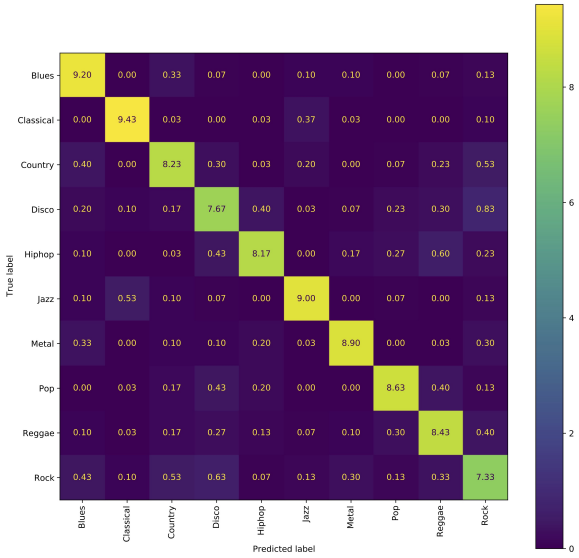

Figure 3: A confusion matrix (averaged over 10-fold cross-validation) for a high-performing Random Forest model

5.2 Random Forest

The Random Forest achieved a mean accuracy (across 10-fold cross-validation) of 85.70%. This was accomplished by training the random forest on its min-max normalized wrapper-selected feature set with $n = 150$ estimators, no maximum tree depth, a maximum of $f = 6$ features to be considered at each node split, and with the Gini Impurity measure for determining when to split.

The mean confusion matrix for the random forest model is displayed in Figure 3.

As was with the MLP, the random forest had the highest accuracy rate for the ‘Classical‘ genre and the lowest accuracy rate for the ‘Rock‘ genre. In addition, the random forest often confused ‘Classical‘ and ‘Jazz‘; ‘Rock‘ for ‘Country‘, ‘Metal‘, and ‘Disco‘; ‘Country‘ for ‘Blues‘ and ‘Jazz‘; and ‘Disco‘ and ‘Hiphop‘.

5.3 Gradient Boost

The Gradient Boost model achieved a mean accuracy (across 10-fold cross-validation) of 86.20%. This was accomplished by training the gradient boost ensemble on the unnormalized full feature set with $n = 100$ estimators, a learning rate of $c = 0.1$, a maximum tree depth of $m = 7$, and a maximum of $f = 8$ features to be considered at each node split.

The mean confusion matrix for the gradient boost model is displayed in Figure 4.

As was with the MLP and the random forest, the gradient boost model had the highest accuracy rate for the ‘Classical‘ genre and the lowest accuracy rate for the ‘Rock‘ genre. In addition, the gradient boost often confused ‘Classical‘ and ‘Jazz‘; ‘Rock‘ and ‘Disco‘; ‘Rock‘ and ‘Metal‘; ‘Rock‘ for ‘Blues‘ and ‘Country‘; and ‘Reggae‘ and ‘Disco‘.

6 Conclusions

Across the five sets of experiments that we ran with the nine different models on our expanded GTZAN dataset, we were able to achieve astonishingly high accuracies with three of the models: the Multi-layer Perceptron model (85.17%), the Random Forest ensemble (85.70%), and the Gradient Boost Decision Tree ensemble (86.20%). The other six models also did remarkably well, all scoring above 60% accuracy by the final experiment.

Also notable is the fact that several of the models were able to achieve near-maximum accuracy with a PCA-reduced feature set or with their respective wrapper-selected feature sets. Thus, if one wanted to opt for a simpler model, there exist models that could perform at a high level with only 15 features (instead of the full 73).

As we look at the confusion matrices of the top three models, we see several commonalities between them. All three of the models had the highest accuracy score for the ‘Classical‘ genre. When one considers the style of classical music in comparison with the other nine genres, this is not too surprising: the other nine genres are all much newer in origin than and collectively much different in style from classical music.

All three of the models also had the lowest accuracy score for the ‘Rock‘ genre; in fact, all three models scored so low on the ‘Rock‘ genre that if it had not been included in the dataset, it is entirely possible that mean classification accuracies above 90% could have been achieved by all three models. This is not entirely surprising either; the genre of rock music is often considered a conglomerate of multiple genres and–in its early years–was heavily influenced by blues and country [Rock and of Fame, 2021]. Interestingly enough, all the three models confused ‘Rock‘ with at least one of these two genres, if not both.

When we consider the other genres that the top three mod-

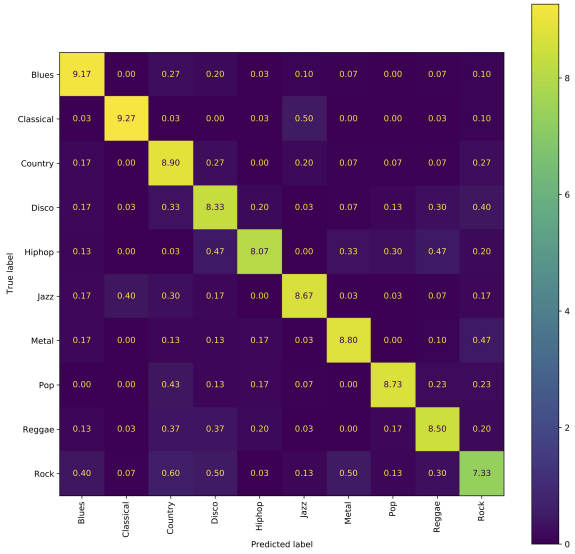

Figure 4: A confusion matrix (averaged over 10-fold cross-validation) for a high-performing Gradient Boost model

els often confused, we also should not be surprised. 'Classical' and 'Jazz' were confused by all three models; these two genres share many similar instruments (e.g., more traditional instruments such as the piano, the clarinet, etc.) and also are the only two genres that may not have a vocal performance in a song. As noted previously, 'Rock', 'Country', and 'Blues' were often confused by the three models; these three genres share similar roots or, in the case of 'Rock', originated from the other two. Note also that 'Rock' and 'Metal' were confused by the Gradient Boost model; the metal genre is sometimes seen as a subset of the rock genre. Many of these genres may possibly be confused by humans as well.

In summary, we know that high accuracy can be achieved and that the mistakes made by the top-performing models are similar (although perhaps slightly more numerous) to the mistakes that might be made by human listeners.

6.1 Future Work

Expanded Dataset

One of the largest drawbacks of our experiments was the limited amount of data with which we could train the models. Even after augmenting the GTZAN dataset by splitting each 30-second clip into three 10-second clips, we still only had 3000 samples.

In addition, it is entirely possible that we introduced significant bias into the models by splitting the clips in this way. Although none of the instances were identical, each instance in our augmented set had two other instances that could have been very similar to it; this could have resulted in our models simply memorizing these similarities instead of memorizing similarities within the genres themselves.

The Google AudioSet has 60,960 10-second audio samples for each of the ten genres we experimented with for this project; adding these 10-second audio samples to our augmented dataset could allow our models to generalize better to new data.

Eliminate Unnecessary Features

Although we did feature selection experiments for each individual model, we did not conduct any experiments to eliminate unnecessary features from the dataset as a whole. It is possible that there are features that were not significantly used by any of the models with which we experimented. By identifying and removing such features, additional noise could be removed from the data and the model performances might increase.

Improving 'Rock' Genre Classification

As noted previously, the top three models had the most difficulty in correctly identifying music samples from the 'Rock' genre. We have supposed that this is likely due to the similarity of 'Rock' to some of the other genres, namely 'Blues', 'Country', 'Metal', and 'Disco'.

We believe that it would be worthwhile to, first, research which audio features are particular pertinent to identifying the 'Rock' genre. It is likely that other researchers have written papers on this particular question. We could then include these features (if they were not already included in our feature set) and perhaps weight them more heavily when attempting to classify the 'Rock' genre.

We also believe it would be worthwhile to train a number of models to specialize in identifying 'Rock' music. These specialized models could then be combined in an ensemble with our top-performing models from this current project.

Nonhomogeneous Ensemble Approaches

Although we utilized two ensemble approaches in our project–Random Forest and Gradient Boost–both of these approaches were homogeneous in the type of model of which the ensembles were composed; that is, both ensemble approaches were only composed of decision tree models. It might be worth exploring the combination of different types of models; for example, if we combined our top three-performing models–the Multi-layer Perceptron, the Random Forest, and the Gradient Boost ensemble–we might be able to achieve an even higher accuracy than our current maximum 86.20%.

References

[Brownlee, 2021] Jason Brownlee. Nested cross-validation for machine learning with python. *Machine Learning Mastery*, Jan 2021.

[MARSYAS, 2015] MARSYAS. Data sets, 2015.

[Peeters, 2004] Geoffroy Peeters. A large set of audio features for sound description (similarity and classification) in the cuidado project. 2004.

[Rock and of Fame, 2021] Rock and Roll Hall of Fame. Roots of rock, 2021.

[Tzanetakis and Cook, 2002] G. Tzanetakis and P. Cook. Musical genre classification of audio signals. *IEEE Transactions on Speech and Audio Processing*, 10(5):293–302, 2002.