Music Genre Classification: Ensemble of Classifiers

EEE498/591: Machine Learning
Team: Naive Boyes

Jonah Yi, Dale Mondejar, Zachery Garibay, Adan Partida

Abstract

The overall purpose of this project was to explore different machine learning algorithms to classify songs to their corresponding musical genre. A GTZAN containing 1000 musical tracks was used as the dataset. Features were extracted from the dataset such as MFCCs and zero-crossing rate. Eight different machine learning models were then used to train on the dataset. The models of each algorithm were visualized and compared for their accuracy. It was found that Neural Network was the best algorithm to classify the dataset. Due to a small dataset, the extracted models were susceptible to overfitting.

Introduction

Music streaming services like Spotify, Apple Music, and others have over millions of tracks. Those streaming services attempt to and successfully classify each track to certain genres and recommend them to their users. There exists an incentive to classify the music tracks accurately and recommend them to its users to encourage them to subscribe to their service. We have explored different Machine Learning algorithms and compared their performance to each of them for the best accuracy and training time

Theory

Any sound can be represented as an audio signal with acoustic features such as frequency, amplitude, and frequency [1]. These features are distinguishable enough so that patterns are prevalent for analysis. Because music genres are by definition audio compositions characterized by similar form and content, it stands to reason that machine learning algorithms are capable of detecting patterns for audio tracks of the same genre. This idea forms the fundamental basis for music genre classification.

There exists multiple methods for classifying music genres; our approach focuses on time domain and frequency domain features extracted from a given input audio signal [4]. A list of each feature extracted and a brief description are as follows:

Mel Frequency Cepstral Coefficients: Power spectrum of a sound representation derived from the inverse Fourier transform.

Zero-Crossing Rate: The rate at which a signal is changing sign.

Spectral Centroid: The location of the center of mass of an audio signal's spectrum determined by the Fourier transform of the signal with the following weights:

$$\text{Centroid} = \frac{\sum_{n=0}^{N-1} f(n)x(n)}{\sum_{n=0}^{N-1} x(n)}$$

Spectral Rolloff: The point in an audio signal's power spectrum where 85% of the power is at the lower frequencies.

Spectral Bandwidth: The difference between the upper and lower frequencies in a band of frequencies.

Chroma Frequencies: Measurement of an audio signal's sounds and subsequent classification of said sound as higher, lower, and medium.

Root-Mean-Square: The root mean square value for an interval of an audio signal.

These features were chosen because of their proven usefulness in classifying musical genres [5][6].

Methodology

The GTZAN dataset was used for our comparison. The dataset consists of 1000 audio tracks each 30 seconds long. It contains 10 genres, each represented by 100 tracks. The tracks are all 22050 Hz Mono 16-bit audio files in .wav format. The genres are blues, classical, country, disco, hip hop, jazz, metal, pop, reggae, and rock.

The raw data embedded within the GTZAN dataset must be transformed into more useful representations. The data was read into a CSV file to allow for easier feature extraction. The features were extracted using the Librosa package. All of the features were standardized using the Scikit-learn StandardScaler. Scikit-learn's principal component analysis was used to remove unuseful features. The data set was split into test and train datasets with 30% of the data being used as the test data.

With our data successfully standardized and split, we trained multiple classifiers and extracted our models for performance comparison. The trained classifiers were KNN, SVM, Logistic Regression, Random Forest, MLP, Naive Bayes, Decision trees, and a neural network with dropout and L2 regularization. These models were chosen for their feasibility of implementation given the deadline for the project as well as their proven accuracy in machine learning literature [3][7].

Each of the models was exhibiting overfitting in the training dataset causing each of the testing accuracies to be lowered. The initial proposed solution to the overfitting was to optimize each of the models using parameter turning, principal component analysis, and dropping some of the MFCCs. The models that required optimization, due to their near 100% training accuracy, were the SVM, Random Forest, and neural network models [2].

Optimization of the SVM and Random Forest included using cross-validation to search for the optimal parameters for the models. The SVM model implemented GridSearchCV with 'cv=5' in an attempt to reduce the overfitting and improve the accuracy of the model which successfully reduced the training accuracy to 86% and testing to 60%, reducing the overfitting in the model. Random Forest implemented RandomSearchCV to cross-validate the dataset randomly in order to tune all of the parameters in the trees and nodes which proved to be a crutch on the training accuracy [9]. While tuning the parameters improved the training accuracy, the processing time and training accuracy were significantly worse as overfitting was heavily present with a training accuracy of 100% and the highest training time of around 75s. This would be improved or solved by using a larger dataset and adding regularization. The Neural Network was optimized to reduce the overfitting, by adding in regularization, to punish the over-training, and dropout to reduce the training accuracy which proved to be successful. [8].

Adding in these optimizations was done in an attempt to combat or reduce the overfitting which was exhibited by a handful of the models and to improve the training accuracy of the models. Both of the models that implement CV or parameter tuning, used relatively more time in order to train the models due to the time taken to tune each of the parameters which reduced how well they performed overall since time was taken into account but negligible since the training time for each of the models was less than 2 min each but may increase as data scales.

After testing some of the models and observing overfitting, a correlation matrix was applied in order to potentially drop features to improve dimensionality and reduce complexity to reduce overfitting. The matrix showed that while some of the parameters had relatively high correlation values, it was not worth getting rid of them in order to improve accuracy as they contained highly important features as shown in figure 1.

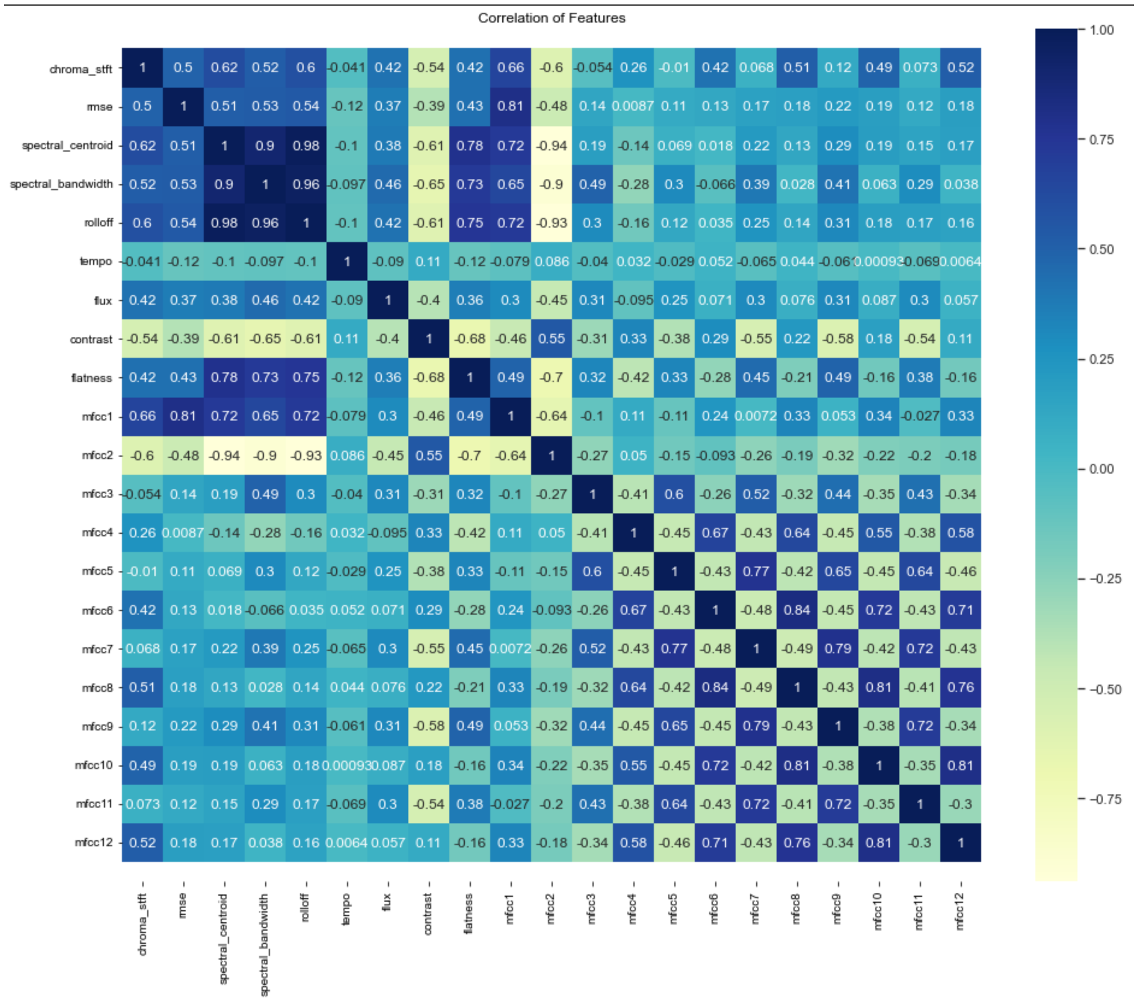

Figure 1: Correlation between features to visually show the importance

Initially, MFCCs 13-20 were dropped due to prior knowledge obtained from research [reference research] and to reduce complexity. With investigation, the dropped MFCCs did not positively contribute to the accuracies as well. A PCA was then applied to the data to validate our hypothesis which proved to only drop 1 or 2 features that were not deemed important which was usually the last 1 or 2 MFCCs and did not necessarily have an effect on the training or testing accuracies. While PCA is often used for unsupervised learning, we thought it would not hurt or decrease the accuracies and could potentially provide a benefit that was minimal and did not require much computing power or time.

Results

The performance metrics of note were test accuracy, train accuracy, and training time. We collected these metrics in table 1 below.

Algorithm	Test Accuracy	Train Accuracy	Training Time
KNN	64%	78%	1.39s
SVM	60%	86%	6.30s
Logistic Regression	64%	78%	.045s
Random Forest w/ parameter tuning	67%	100%	73.32s
MLP	59%	100%	.22s
Naive Bayes	57%	70%	.002s
Decision Tree	48%	100%	.008s
NN w/ DO	72%	93%	22.66s

Table 1: Test accuracy, train accuracy, and training time of each algorithm.

There are a few standout results in the table. The first is the abysmal training time for the Random Forest algorithm relative to the other models. This result was validated with multiple training runs on different machines. On the opposite end of the spectrum, the Naive Bayes algorithm trained exceptionally fast which is not surprising given the fact that it does not build an entire model but instead makes estimations and classifies according to those estimates. The Random Forest, multilayer perceptron, and Decision Tree models all had a large disparity in their respective test and training accuracies which indicated the presence of strong overfitting. The models that had the best performance, in terms of accuracy, contained relatively less overfitting than the other models which means that the models were being trained too well and needed to be trained differently. The overfitting comes from the complexity of some of the models specifically, Random Forest, Decision Tree, and the multilayer perceptron. While some of this overfitting can be attributed to the relatively small dataset of 1000 songs, the data is still well balanced with having 100 songs from each of the 10 genres. Even with parameter tuning of the Random Forest, the overfitting becomes even more prevalent with hyper-tuned parameters that yield the best performance which leads to the conclusion that the dataset is not large enough for Random Forest, Decision Tree, and MLP to avoid overfitting the training dataset.

The neural network proved to have the best performance throughout each of the different classifiers used. While this neural network was given the same features as the other models, the improvement comes in the optimization of each of the layers and neurons through dropout and regularization. Before implementing the NN, the other models showcased overfitting, which meant that the neural network would most likely showcase the same behavior as seen in figure 2.

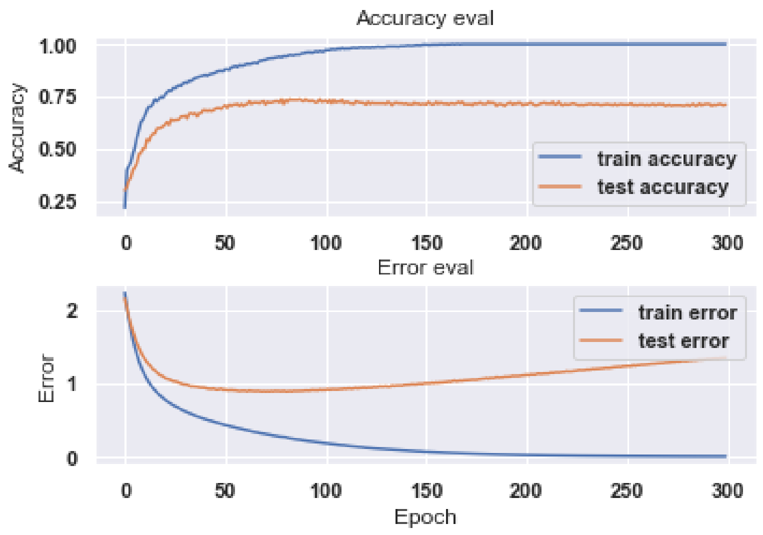

Figure 2: Train and test accuracy with error for the Neural Network plotted over 300 epochs.

With a dropout rate of 20% in each of the 3 implemented layers, along with regularization, the training and testing accuracies had much less divergence over the 300 epochs. The train and test accuracies and errors for the optimized neural network produced results that showed overfitting was mitigated with the training accuracy approaching 92% and testing accuracy 72%-74%.

The key results in the project can be observed in each of the confusion matrices which visually exhibit the performance of each of the models utilized and how each of the 10 genres was misclassified. While looking at the individual matrices may show how each of the models performs on the separate genres, the most important point or trend is seen when observing the models together. Each model tends to misclassify genres that are the most similar in terms of instrumental elements, spectral elements in frequency, and power levels. This adds a level of complexity to the problem and classifiers as it is often difficult for models, and even humans, to make that distinction between highly similar genres and discern the nuanced differences.

The genres that were often misclassified as one another were Jazz as Blues, Jazz as Classical, Hip Hop as Reggae, and Rock as Metal which is seen in figure 3. The distinction between some of the songs for each of these similar genres is nearly blended because songs may take influence from certain musical elements such as the instrumentals, melodies, and rhythms, from other genres which causes a grey area between the songs for similar genres. The misclassifications between each of the similar genres detract from each of the models' capability to correctly classify the genre. This misclassification could also be from human error in misclassifying the genre as songs often blend multiple genres.

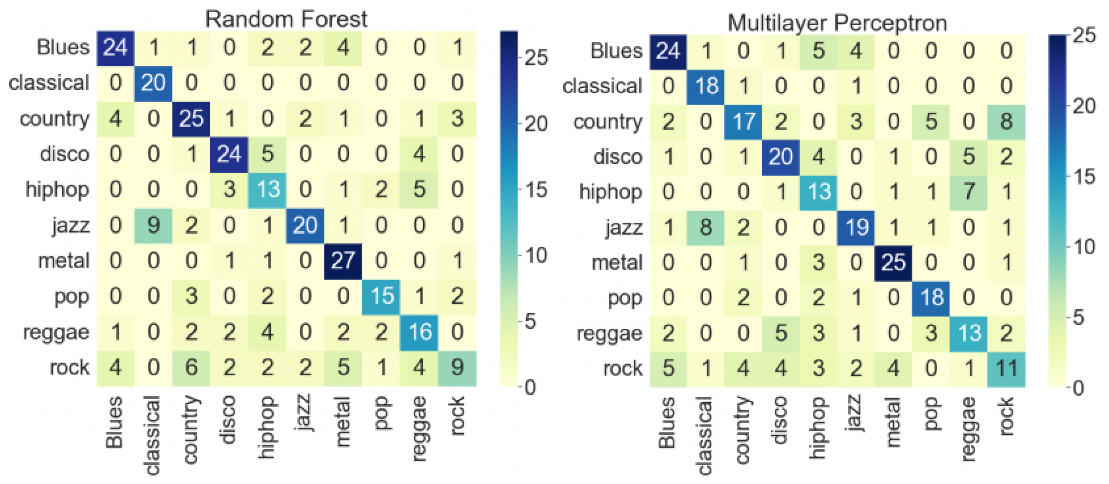

Figure 3: Random Forest and MLP confusion matrices showcasing the misclassifications of each genre.

An interesting trend that is worth mentioning is seen in the worst classifiers of the bunch which were the decision tree model and Naive Bayes. Both of the classifiers have poor performance with each having less than 60% test accuracy but the confusion matrices of each show how correlated each of the genres is with one another. Naive Bayes is particularly interesting as it showcases this trend the most due to its underlying assumption of independence. The confusion matrix, shown below in figure 4, shows how the misclassifications of the genres, make the matrix a correlation of the most similar genres i.e. Jazz and Blues.

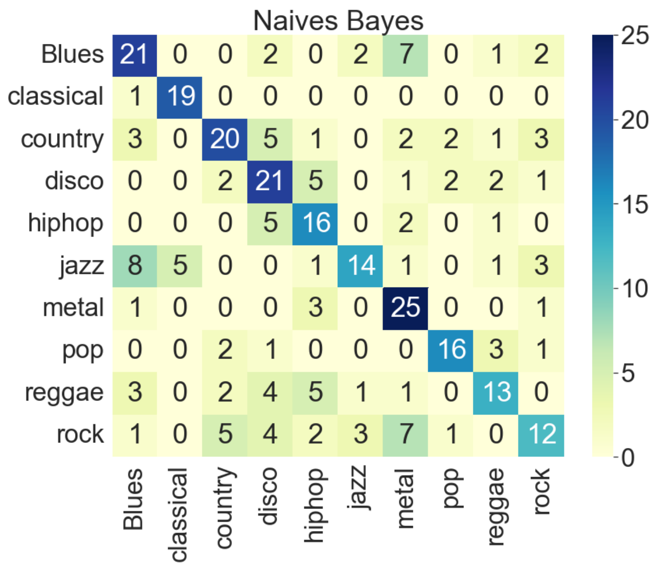

Figure 4: Naive Bayes confusion matrix shows a correlation between similar genres with a high amount of misclassifications.

Conclusion

By using Neural Networks for classifying genres, we could remove user feedback in determining the genres. We encountered overfitting with high variance in our models, we reduced overfitting by adding regularization, dropout layer, and principal component analysis to our model which improved our training and testing accuracy.

Our accuracy could be further improved by adding more data to our model or by using extracted spectrogram images and implementing Convolutional Neural Networks, the accuracy could surpass our chosen algorithm.

References

[1] A. Elbir and N. Aydin, “Music genre classification and music recommendation by using Deep Learning,” *Institution of Engineering and Technology*, 01-Jun-2020. [Online]. Available: https://ietresearch.onlinelibrary.wiley.com/doi/10.1049/el.2019.4202. [Accessed: 03-Dec-2021].

[2] B. Carremans, “Handling overfitting in deep learning models,” *Medium*, 08-Jan-2019. [Online]. Available: https://towardsdatascience.com/handling-overfitting-in-deep-learning-models-c760ee047c6e. [Accessed: 03-Dec-2021].

[3] D. A. Huang, A. A. Serafini, and E. J. Pugh, “Music genre classification - stanford university,” *https://github.com/derekahuang/Music-Classification*, 2018. [Online]. Available: http://cs229.stanford.edu/proj2018/report/21.pdf. [Accessed: 03-Dec-2021].

[4] L. wadhwa, “Understanding music genre classification‐A Multi-Modal Fusion Approach,” *Medium*, 08-Jul-2020. [Online]. Available: https://medium.com/swlh/understanding-music-genre-classification-a-multi-modal-fusion-approach-6989caa87803. [Accessed: 03-Dec-2021].

[5] R. Thiruvengatanadhan, “Music genre classification using MFCC and AANN - IRJET,” *International Research Journal of Engineering and Technology (IRJET)*, Oct-2018. [Online]. Available: https://www.irjet.net/archives/V5/i10/IRJET-V5I10199.pdf. [Accessed: 03-Dec-2021].

[6] R. Thiruvengatanadhan, “Speech/music classification using MFCC and KNN,” *International Journal of Computational Intelligence Research* , 2017. [Online]. Available: https://www.ripublication.com/Openaccess/ijcirv13n10_14.pdf. [Accessed: 03-Dec-2021].

[7] S. Chillara, K. A. S, S. A. Neginhal, S. Haldia, and V. k S, “Music genre classification using machine learning ...,” *https://www.irjet.net/*, May-2019. [Online]. Available: https://www.irjet.net/archives/V6/i5/IRJET-V6I5174.pdf. [Accessed: 03-Dec-2021].

[8] S. Loukas, “PCA clearly explained‐how, when, why to use it and feature importance: A guide in python,” *Medium*, 08-Oct-2021. [Online]. Available: https://towardsdatascience.com/pca-clearly-explained-how-when-why-to-use-it-and-feature-importance-a-guide-in-python-7c274582c37e. [Accessed: 03-Dec-2021].

[9] W. Koehrsen, “Hyperparameter tuning the random forest in python,” *Medium*, 10-Jan-2018. [Online]. Available: https://towardsdatascience.com/hyperparameter-tuning-the-random-forest-in-python-using-scikit-learn-28d2aa77dd74. [Accessed: 03-Dec-2021].