Classifying Electronic Music Subgenres Using Audio Features

Project Report
April 2023
Kevin Cho

1) Problem Statement

Music consists of a large variety of genres ranging anywhere from Ambient to Death Metal. I expect that machine learning techniques can easily distinguish songs when looking at broad genres, but I would like to see how well they can distinguish subgenres of music where the differences are more subtle. I will specifically focus on Electronic Dance Music (EDM) where over 100 subgenres exist. Many subgenres are heavily influenced by others, so they contain many similar audio features. I plan to use supervised classification techniques such as Logistic Regression, SVM, Neural Network, Random Forest, and KNN to see if the audio features of a song can be used to classify its subgenre.

The research questions to be answered are:

1. What is the best prediction accuracy that can be achieved after tuning hyperparameters?
2. Can performing dimensionality reduction with PCA improve prediction accuracy?
3. Do linear or non-linear classification methods perform better for this problem?

If a model with high accuracy is achieved, it could be used to help DJs sort their music libraries. Oftentimes DJ's will download songs in large batches from record pools that are unlabeled or labeled with only broad genres. Sometimes they may even come across songs without artist or track names. Being able to determine the subgenre of a song just by analyzing its audio features could be immensely useful in this scenario. It is critical for DJs to sort to specific subgenres so they can set the intended energy and mood. For example, Deep House and Electro House have a similar tempo and contain many of the same instrumental elements, but they have subtle differences that drastically change the "feel" of the songs and the audience for which a DJ would want to play them. This type of classification could also be useful for music streaming companies to suggest curated playlists to very specific taste.

2) Data Source

The "Electronic Music Features" dataset from Kaggle was used for this analysis. This dataset contains 2,300 rows x 93 columns. Each row represents a song; there are 100 of the top songs for each of 23 distinct subgenres (classes) according to Beatport.com in November 2016. The last column of the data represents the subgenre classifications. It includes 7 variations of House, 2 variations of Techno, and 2 variations of Trance amongst others; so there will be an opportunity to try to distinguish some very similar subgenres. The other 92 columns represent the audio features of each song. This includes information such as beats per minute (BPM aka tempo), loudness, energy, entropy, along with many other features extracted and calculated from the audio file of each song. The features were extracted using pyAudioAnalysis on a random two-minute sample of each song.

Due to the large number of features in this dataset, I expect that the non-linear methods will generally perform better as there is likely a complex relationship between variables. I expect that the benefit from PCA could end up being minimal. Since there are over 20 classes to distinguish, a large number of features is likely needed to separate them. However, it will be interesting to see if similar performance can be achieved while using less dimensions since this would have the benefit of reducing computational complexity.

I expect that no model will be able to achieve 100% accuracy since EDM subgenres are fluid and the labeling can be somewhat subjective. The labels used in this dataset are the subgenres judged by Beatport.com; the labeling could be different if done by a different listener or even the original artists.

3) Methodology

The analysis was performed using numpy, pandas, and sklearn in a Python Jupyter Notebook and the results were visualized with matplotlib. As part of data exploration, a correlation matrix was created using all 92 features and is displayed in Figure 1 below. Some of the features show high correlation to each other indicating possible multicollinearity (the yellow blocks), but the majority do not. Multicollinearity would have the largest effect on linear methods such as Logistic Regression and Support Vector Machine (SVM). This will be addressed by L2 regularization in the linear methods. Principal Component Analysis (PCA) will be explored for all methods and should also help address multicollinearity.

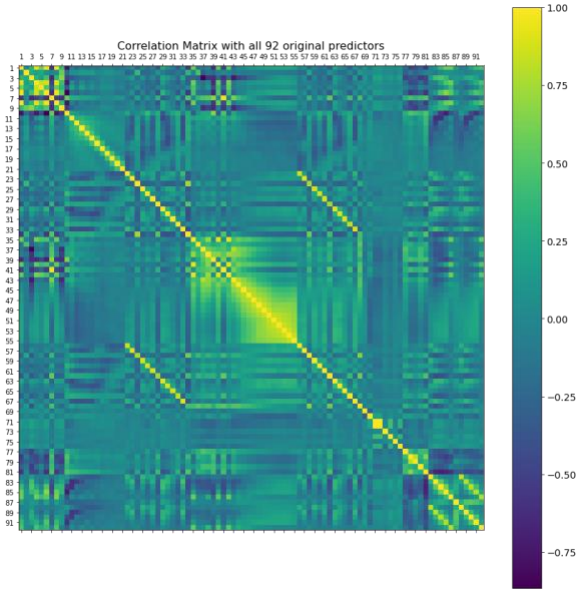

Figure 1 – Correlation Matrix of all 92 Original Features

The data was randomly split to 80% training and 20% test while keeping all the classes balanced. Therefore, there are 80 songs in the training set and 20 in the test set for each subgenre; the total training set contains 1840 songs while the test set contains 460. The features of training set were scaled and normalized using its mean and standard deviation, then the same transformation was applied to the test set. Scaling helps the distanced based methods (KNN and SVM) and the gradient based methods (Logistic Regression and Neural Network) perform better. The hyperparameters of each model were tuned using 3-fold CV on the training data.

The proposed methods are described below:

1. Logistic Regression
    a. Parametric and linear classifier that fits to a sigmoid function then classifies new datapoints based on their probability
    b. Hyperparameter tuning: regularization coefficient value for L2 penalty term
2. Linear SVM
    a. Uses a linear kernel
    b. Parametric classifier that separates classes by drawing linear decision boundaries balancing maximizing margin and minimizing misclassification
    c. Hyperparameter tuning: regularization coefficient value for L2 penalty term
3. RBF (Radial Basis Function) SVM
    a. Uses an RBF kernel
    b. Parametric classifer that separates classes by drawing non-linear decision boundaries balancing maximizing margin and minimizing misclassification
    c. Hyperparameter tuning: regularization coefficient value for L2 penalty term
4. Neural Network
    a. Parametric and non-linear classifier that uses a network of weighted and interconnected perceptrons to classify new datapoints
    b. Hyperparameter tuning: solver for weight optimization and hidden layer sizes
5. Random Forest
    a. Non-parametric and non-linear classifier that classifies new datapoints based on the majority classification of an ensemble of decision trees
    b. Hyperparameter tuning: number of trees, number of features to randomly consider at each split, criterion to measure split quality
6. K-Nearest Neighbors (KNN)
    a. Non-parametric and non-linear classifier that classifies new datapoints based on the majority classification of their *k* nearest neighbors
    b. Hyperparameter tuning: *k*-value, distance metric, and weight function

Each of the 6 models described above were trained with two separate datasets (original and PCA), resulting in 12 total models for comparison. The original dataset was scaled and contains the 92 original features. Principal Component Analysis was used to apply dimensionality reduction and create the PCA dataset. The principal components were fit to the features of the training set (post-scaling), then the same transformation was applied to the test set. Figure 2 below shows that 46 principal components capture 95% of the variance, so 46 PCs were used for the PCA dataset. The hyperparameters were tuned using 3-fold CV on the PCA training data.

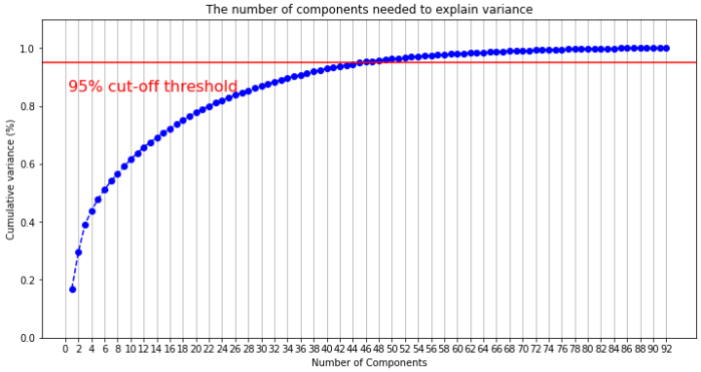

Figure 2 – Explained Variance vs. Number of Principal Components

The tuned hyperparameters for each model are summarized in Table 1 below. The hyperparameters for most methods are different when using the original dataset vs. the PCA dataset. The regularization coefficients for the linear methods (Logistic Regression and Linear SVM) are greater than 1, suggesting that they are addressing multicollinearity. The hidden layer sizes of the Neural Network models may not be completely optimized; only a few different sizes were evaluated due to long runtime. Random Forest considers less features at each split to keep the randomness high when using the PCA dataset since it contains half as many features as the original dataset. The optimal *k*-value and distance metric both change for KNN when using the PCA dataset vs. original, suggesting a significant difference in the spatial arrangement of the datasets.

	92 Original Features	**46 Principal Components**
Logistic Regression	Regularization coeff. = 10	Regularization coeff. = 10
Linear SVM	Regularization coeff. = 100	Regularization coeff. = 10
RBF SVM	Regularization coeff. = 1	Regularization coeff. = 1
Neural Network	Solver = Stochastic Gradient Descent; Hidden Layer Sizes = (1000,500)	Solver = Stochastic Gradient Descent; Hidden Layer Sizes = (500,500)
Random Forest	Num. trees = 1000; Num. features to split = 10; Split quality criterion = Gini impurity	Num. trees = 1000; Num. features to split = 4; Split quality criterion = Gini impurity
KNN	*k*-value = 9; Distance metric = Manhattan (L1) Weight function = Inverse of distance	*k*-value = 13; Distance metric = Euclidian (L2) Weight function = Inverse of distance

Table 1 – Tuned hyperparameters of all 12 models

4) Final Results

Figure 3 below shows the classification performance metrics for each of the 12 models on the test data. The precision, recall, f1-score, and accuracy end up being similar within each model; therefore, I will focus on accuracy to compare the models to each other.

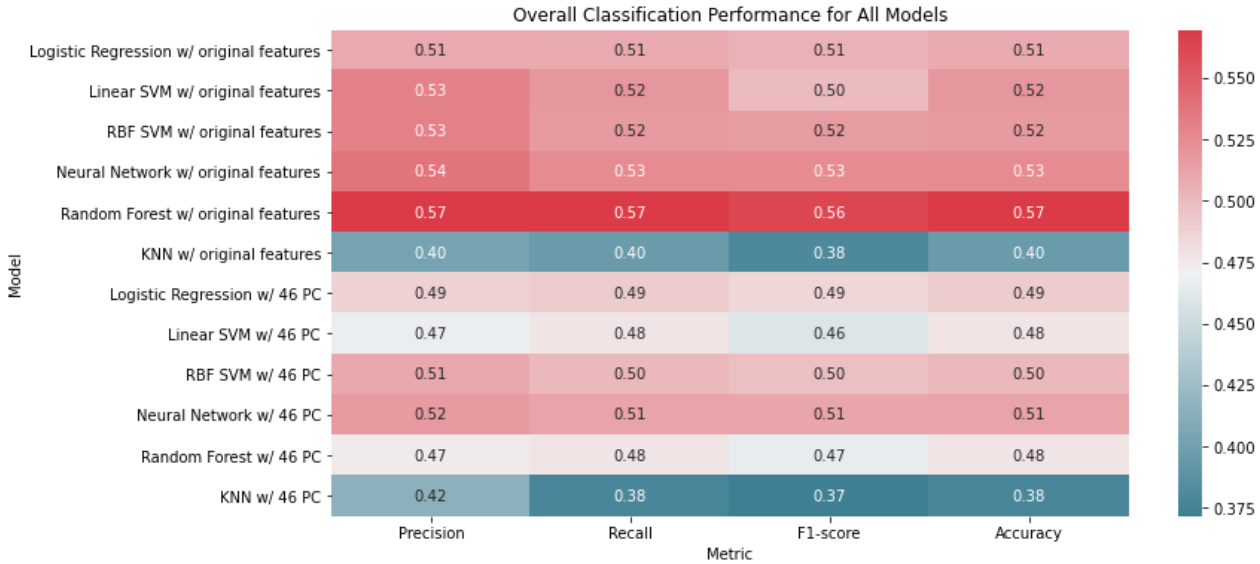

Figure 3 – Classification Performance Metrics for All Models on Test Data

Random Forest w/ all 92 original features has the highest overall accuracy at 57%; this is 4 percentage points higher than the next best model which is Neural Network w/ original features at 53%. Although 57% accuracy is not as high as I hoped for, it is much better than randomly guessing one of the 23 subgenres (1/23 = 4.3%). This suggests that Random Forest is a feasible method for this classification problem and could possibly be improved further with additional hyperparameter tuning, data preprocessing, and/or in conjunction with other methods such as boosting.

Each method performs slightly worse with the PCA data compared to the original data, but most are within 1-2 percentage points. The only exception being Random Forest which experiences a larger drop-off in accuracy from 57% to 48% when applying PCA, making it perform around the same level as the linear methods with PCA. Although the 46 principal components capture 95% of the variance of the original dataset, it seems Random Forest needs all the variance to improve prediction performance and is most sensitive to the loss in information. I would conclude that PCA is generally effective in reducing model complexity since uses half the number of features with minimal drop in accuracy for most methods, but it is not desirable for achieving the absolute best accuracy.

In general, the non-linear models performed better than the linear ones (with a few exceptions); this suggests that the decision boundaries are complex and better fitted by the non-linear methods. An exception is KNN; both KNN models had 10+ percentage points lower accuracy than all other models. I suspect that KNN is suffering from the high dimensionality of the datasets; the feature spaces are very sparse and the test points may be very far from the training points. The other exception is Random Forest after PCA; it has one of the lowest accuracies of the non-KNN methods as explained in the prior paragraph.

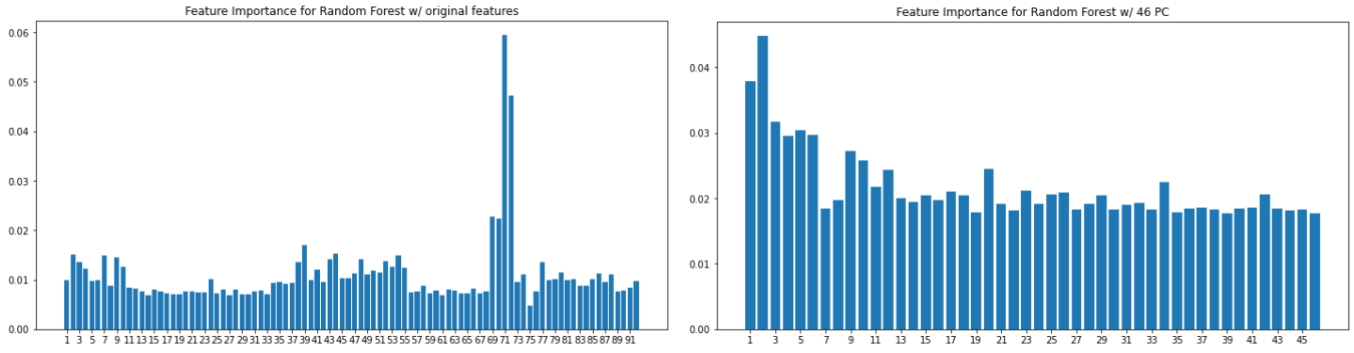

Figure 4 – Feature Importance for Random Forest with (a) 92 Original Features (b) 46 Principal Components

Analysis for the rest of this section will focus on Random Forest, the best performing method. Figure 4 above shows the Gini importance of each feature for the Random Forest models. The left plot represents the original 92 features and shows that features 71 and 72 have the highest importance. These two features are related to BPM (beat per minute); this sounds reasonable since BPM is typically one of the most defining characteristics of a subgenre. The right plot represents the 46 principal components and shows that the first 6 PCs have slightly higher importance than the subsequent PCs. This is expected since each subsequent PC contains less explained variance.

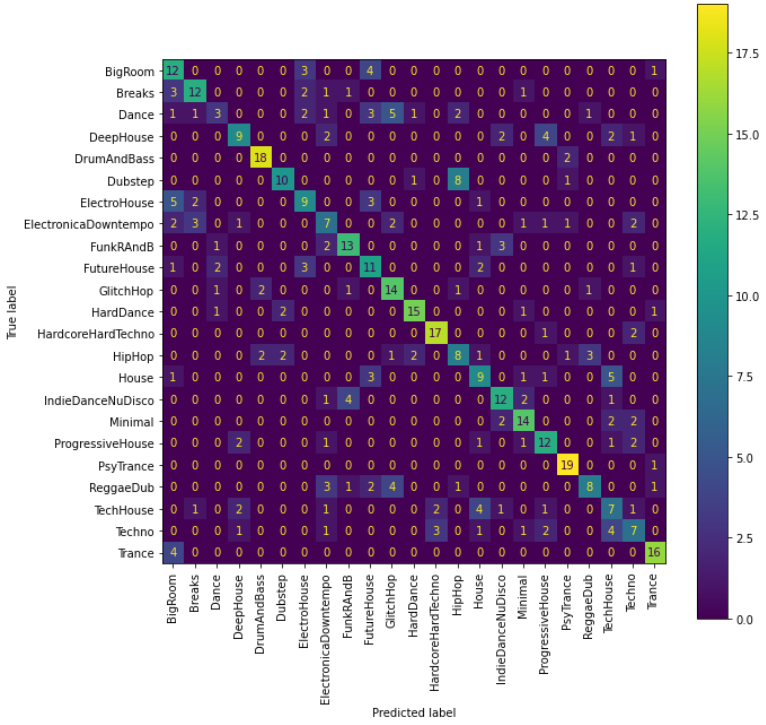

Figure 5 – Confusion Matrix for Random Forest with all 92 Original Features

Figure 5 above shows the confusion matrix for the Random Forest model with all 92 original features. Each subgenre has 20 test data points; the number of correctly classified test data points for each subgenre will be denoted in parentheses in the following sentences. The model is best at classifying Psytrance (19), DrumAndBass (18), and HardcoreHardTechno (17). The most difficult subgenres to classify were Dance (3), ElectronicaDowntempo (7), TechHouse (7), and Techno (7). I suspect that Dance could be a generic subgenre used by Beatport when they are unsure of the exact subgenre, hence why its test predictions were scattered amongst 9 other subgenres. TechHouse and Techno were confused with each other and many other variations of House and Techno; this is not surprising since they are derivatives of each other and contain many similar musical elements. Many of the 7 variations of House were confused with each other; out of these subgenres the model was best at classifying BigRoomHouse (12), ProgressiveHouse (12), and FutureHouse (11). I would assess that this is reasonably good and a promising result given the number of subgenres and their similarities; the predictions are still much better than randomly guessing. It is also encouraging to see that PsyTrance (19) and Trance (16) were barely confused with each other and both were classified with relatively high accuracy.

5) Conclusion and Future Work

The best classification accuracy of 57% was achieved with a Random Forest model using all 92 original features. Most methods had slightly worse accuracy (1-2 pp) when applying PCA with 46 principal components, while Random Forest had a larger drop-off (9 pp). In general, the non-linear methods had higher accuracy than the linear, the exceptions being Random Forest with PCA and KNN. It appears feasible to use audio features to classify songs to EDM subgenres based on the results of this analysis, however this is just a starting point and more work could be done to improve classification performance.

Ideas for future work include:

1. Training with additional data
    a. This dataset was not very large given the number of classes
2. Variable selection prior to model training
    a. Although L2 regularization was used for the linear methods, they may have still suffered from some multicollinearity
3. Additional tuning of Neural Network
    a. NN was the second-best performing method but only a small number of hidden layer sizes were tested due to long runtime
4. Additional tuning of Random Forest
    a. Other hyperparameters such as max depth and minimum leaf size could be tuned
5. Explore Boosting
    a. A boosting algorithm such as AdaBoost could possibly reduce bias and improve prediction performance
    b. Could also be applied in conjunction with Random Forest
6. Explore Clustering
    a. Clustering may reveal more natural grouping than the subgenres assigned by Beatport

6) Resources

1. Electronic Music Features – 201611 BeatportTop100 in *Kaggle*
    a. https://www.kaggle.com/datasets/caparrini/electronic-music-features-201611-beatporttop100
2. pyAudioAnalysis
    a. https://github.com/tyiannak/pyAudioAnalysis