Automatic Music Genre Classification with Random Forests

Jonathan Campbell

Columbia University

Abstract

Ever-expanding quantities of music data necessitate the development of algorithms to organize and locate audio samples. I test an audio classification application of Random Forests which predicts the genre of an audio sample based on Fourier Transform features. Results indicate that Random Forests' rate of successful classification compares favorably to that of other common classification algorithms: Linear Discriminant Analysis and Boosting. However, demonstrated accuracy rates are still not as high as accomplished in the literature. I suggest investigating dimensionality reduction algorithms such as Principal Component Analysis in order to increase successful classification rates for future Random Forests audio classification research.

Keywords: Genre Classification, Feature Extraction, Audio Features, Random Forests, Supervised Learning

Automatic Music Genre Classification with Random Forests

Introduction

The proliferation of databases storing digital music has necessitated research into methods which may organize ever-increasing amounts of data for quick querying and retrieval. The amount of digital data held worldwide has been increasing exponentially in recent years such that the amount of data created and copied in the year 2011 exceeded scientists' expectations by reaching a total sum greater than 1.8 zettabytes. This quantity is more than 10 times greater than the amount of data created just five years earlier in the year 2006 (Gantz et al., 2008; Gantz & Reinsel, 2011; Dannenberg, Foote, Tzanetakis, & Weare, 2011). As advancements in disk storage continually decrease costs and increase storage capacity, developments specific to music data generation are driving an audio data explosion. Pushes to digitize music libraries originally recorded on analog media and the development of sophisticated sharing platforms all contribute to the continually increasing volumes of music data being produced each year (Scaringella, Zoia, & Mlynek, 2006). With these developments in mind, it becomes clear that systematic organization protocols must evolve to suit ever-growing amounts of data in order to document audio samples for future retrieval.

Music genres traditionally constitute the mechanism by which music catalogues are organized, however these are far from absolute delineations. Genres may be defined by characteristics only indirectly related to musical content such as geographical origin, and even those directly related to aural characteristics rely on subjective judgments to differentiate one genre from another (Pachet & Cazaly, 2000). These judgments are far from unanimous;

researchers find that humans routinely characterize audio sample genres incorrectly at a rate of about one-in-four (Lippens et al., 2004).

Further, manual genre classification by humans is both time-consuming and demanding in terms of the labor required. Reports indicate that one such undertaking by Microsoft required 30 musicologists for one year in order to classify 100,000 music samples (Scaringella et al., 2006). Because of the labor intensiveness and fallibility associated with human manpower, machine learning algorithms must be implemented in order to feasibly organize ever-increasing sums of digital music data.

Computer analysis of music styles cannot be undertaken presently using audio samples containing direct digital representations of sound waveforms, as these representations convey relatively sparse levels of information (Scaringella et al., 2006). Instead, researchers use Fourier Transforms to translate audio waveforms into numerical features (Catalepe, Yaslan, & Sonmez, 2007). These features may be characterized as either short-term or long-term.

Short-term features are derived from an extremely brief portion of a music sample, such as a frame consisting of fractions of a second. Timbral features describing tonal qualities of sound are examples of short-term features computed with Fast Fourier Transforms, and have utility for applications in discerning music from environmental sounds, instrument classification, and speech recognition. Long-term features are sampled from a combination of short-term frame intervals and provide rhythmic information (Catalepe et al., 2007).

According to Banitalebi-Dehkordi & Banitalebi-Dehkordi (2014), a combination of short- and long-term audio features benefits classification accuracy. This principle can be applied to common pattern recognition algorithms researchers employ in order to discriminate between

different classes of music (Scaringella et al., 2006). Such techniques typically generate a similarity measure between audio samples based on the provided features. This distance measure then informs the placement of audio samples into clusters of observations with similar feature values.

K-means is the most common unsupervised clustering procedure used to classify audio samples, however, it is likely that further gains in accuracy may be achieved with other methods (Costa et al., 2012). McKay & Fujinaga (2004) report a remarkable 98% accuracy rate by coupling Fourier Transform-derived features with additional features discerned from MIDI files, a standard format for representing digitally-generated music, though these findings have not consistently been replicated by later studies using similar methods (Catalepe et al., 2007; McKay & Fujinaga, 2004). The use of supervised models such as linear discriminant analysis (LDA) for audio classification is well established in the literature. Supervised models typically have higher utility in this application when compared to unsupervised clustering approaches (McKay & Fujinaga, 2004).

One approach for audio classification applications which has not been investigated to its full extent is a design utilizing a Random Forest classification method. Previous research provides evidence that a Random Forest application may classify audio samples more successfully than logistic regression, Gaussian Mixture Models, and Support Vector Machines (Laurier & Grivolla, 2008; Saki & Kehtarnavaz, 2014; Kursa, Rudnicki, Wieczorkowska, Kubera, & Kubik-Komar, 2009). The present study employs a Random Forest classifier in order to determine whether such an approach compares favorably in classification accuracy to other techniques. To this end, I conduct exploratory analyses with the following methods: Random

Forest classification, unsupervised hierarchical clustering, as well as the supervised techniques Linear Discriminant Analysis and Boosting. In accordance with the previous studies cited above, I predict that the Random Forest classifier will produce more accurate results compared to each of the other supervised or unsupervised methods.

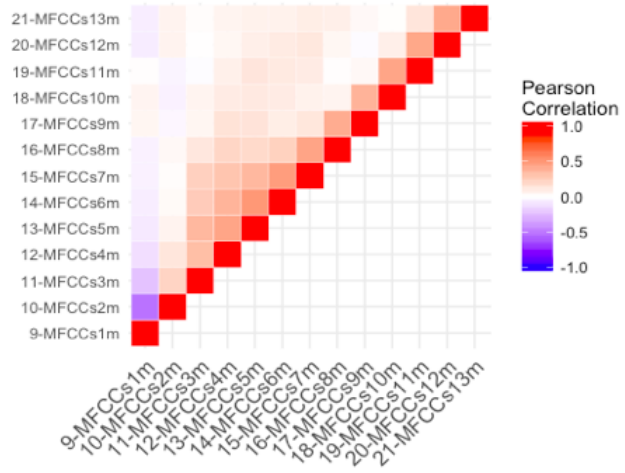
Data

This dataset contains 2300 observations of November 2016's top 100 songs across 23 electronic music genres. The columns of the dataset describe 71 distinct sound characteristics derived from the pyAudioAnalysis signal processing toolkit implemented in Python (Giannakopoulos, 2015). These audio features contain continuous data quantifying timbral and rhythmic information of a given song. MFCC features captured by Fast Fourier Transform are frequently used in sound classification literature and are represented as a 13-component feature vector in this data set. Features that are less commonly applied to audio classification research also appear in the data, such as the 12-component Chroma feature vector which measures the presence of each of the 12 pitches that make up typical Western scales. Further, information expressing characteristics of the frequency spectrum of the audio samples are also included in variables such as *Spectral Centroid* and *Spectral Flux* (Giannakopoulos, 2015).

Examination of summary statistics and correlation heatmaps seems to indicate considerable variation within individual components of the MFCC and Chroma feature vectors. Each MFCC feature seems to correlate moderately positively with the MFCC feature directly following it, but does not correlate strongly with others. The MFCCs1m component is the only one in the MFCC feature vector which shows a slightly different pattern, correlating negatively

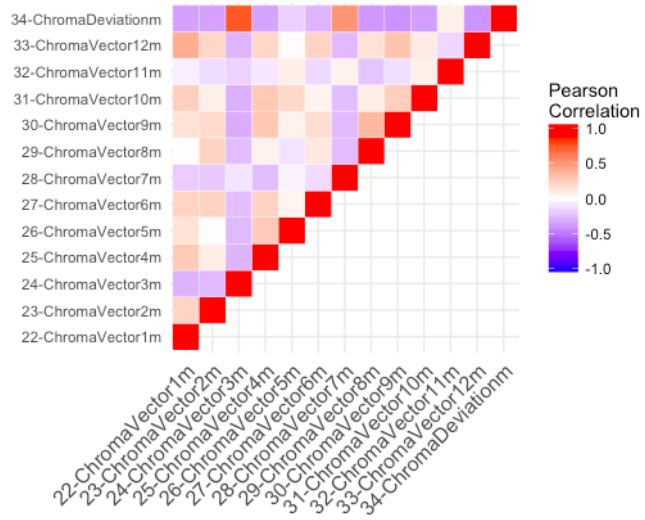
with the MFCC feature directly following it, but not correlating very strongly with other MFCC features.

Figure 1. *Correlations Between MFCC Features*



Correlations between components in the Chroma feature vector reveal slightly more variation than those between MFCC features. However, each one of the variables seems to generally relate to the rest in much the same way. For example, the ChromaVector1m and ChromaVector2m features are generally moderately positively related to each of the rest of Chroma components in the feature vector. Though patterns emerge, these correlation tables indicate that there is sufficient variation within these feature vectors, challenging the notion that the MFCC and Chrome feature vectors could be heavily reduced to only one or two features. Comprehensive descriptive statistics and correlations for each feature are provided in Appendix A, Tables 1&2, Figures 1-4.

Figure 2. *Correlations Between Chroma Features*



The dataset contains rhythmic information expressed in terms of several features relating to the tempo of a given audio sample. Beats per minute (BPM) is measured in two features, with one deriving values using an alternate algorithm from the Essentia C++ audio analysis library implemented in Python (Bogdanov et al., 2013). For the native pyAudioAnalysis algorithm, a confidence measure supplements the featureset, providing information concerning the "dominance" of a tempo within audio frame intervals, or certainty that the calculated BPM measure is representative of the true tempo (Giannakopoulos, 2015).

Perhaps most important for supervised machine learning techniques is the human-compiled variable describing the genre of music, in this case including categorical labels for sub-genres of electronic music such as "Techno," "Dance," and "Drum and Bass." This feature is intended for use with supervised classification procedures which train on these data before attempting to reconstruct these categories in the testing phase. Detailed description for each of

these features, including mathematical expressions for computation when applicable and/or available, is included in Appendix B.

A primary advantage to this dataset is that missing data is not a concern. Given that the featureset is generated by transformation of audio files into quantitative information by Python, data is complete so long as each desired song is readily available. Even if this were not the case, one could impute additional songs within the same genre for transformation by pyAudioAnalysis in order to replace missing values for any given observation.

Study 1

Method

Unsupervised Learning.

I start by conducting hierarchical clustering techniques to sort music pieces based on their extracted audio information. Since previous work has called into question the utility of k-means for audio classification, I choose instead to perform hierarchical clustering in exploratory analysis in order to examine its performance relative to supervised classification techniques for discerning between music genres (Costa et al., 2012).

This technique groups observations together by a computed measure of similarity. There are several algorithms which accomplish this, and I try three of them. The first is called single-linkage, which groups individual points with the smallest distances into like clusters:

$$D(X,Y) = \min_{x \in X, y \in Y} d(x,y),$$

where the distance between two sets of clusters X and Y is evaluated for $x \in X$ and $y \in Y$.

I determine the optimal quantity of clusters through cutting the dendrogram plotting the subsequent distances between points which form clusters at the height value for the longest tree. This tree represents the densest cluster configuration encompassing the greatest amount of observations. The dendrogram cuts are represented by blue, vertical lines in Appendix A, Figures 5-7 ("Single-linkage clustering," n.d.).

I repeat this process for the next two clustering applications using first a complete-linkage algorithm followed by an average-linkage algorithm. The distance function for complete-linkage is identical to that used in single-linkage, except that clusters are formed based on the farthest distance from other groups rather than the closest:

$$D(X,Y) = \max_{x \in X, y \in Y} d(x,y)$$

("Complete-linkage clustering," n.d.).

Finally, average-linkage clustering weights the distance between new clusters proportionally based on the distance between observations in existing clusters, where $\mathcal{A}$ and $\mathcal{B}$ are any two existing clusters and *X* is a new one:

$$d_{(\mathcal{A} \cup \mathcal{B}),X} = \frac{|\mathcal{A}| \cdot d_{\mathcal{A},X} + |\mathcal{B}| \cdot d_{\mathcal{B},X}}{|\mathcal{A}| + |\mathcal{B}|}$$

("UPGMA," n.d.).

Results

The outcomes of the hierarchical clustering models do not suggest that such unsupervised techniques effectively represent the data structure of this particular featureset (see Appendix A, Tables 3-13). Cutting along the densest trees on the dendrograms for any of the three distance algorithms results in many clusters which contain only one observation each.

Perhaps single-linkage algorithms model this dataset most effectively, as it is only with this algorithm in which cutting at the densest decision-tree results in a quantity of clusters that is theoretically relevant in that it is near the amount of unique values for pre-classified genre (23).

Study 2

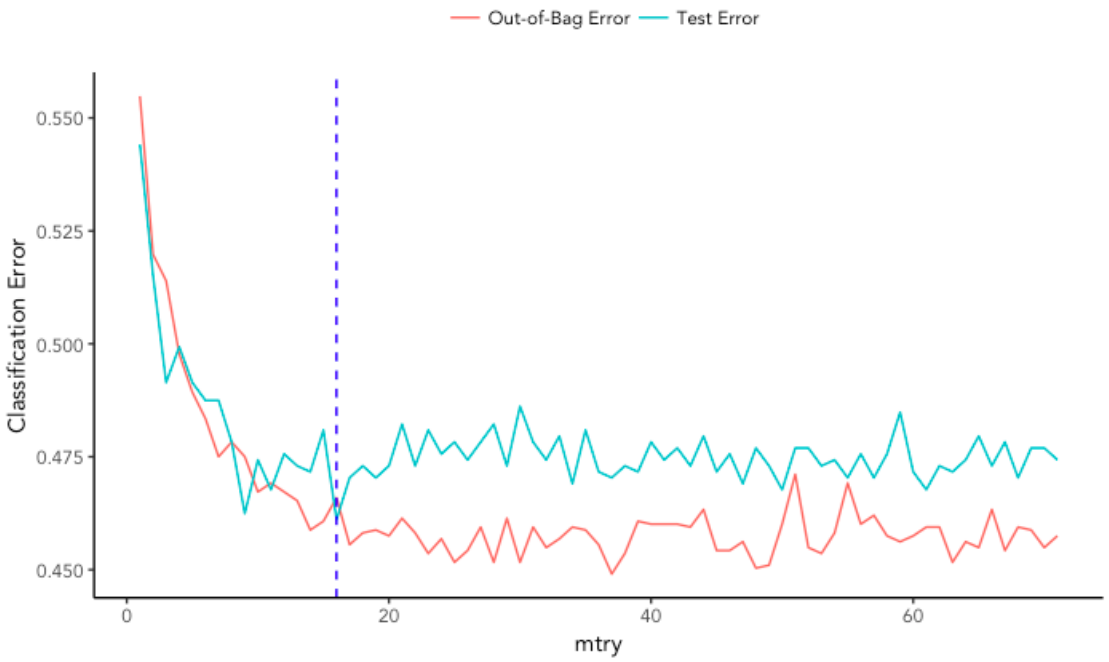
Method

Supervised Learning – Random Forests.

With the failure of unsupervised learning techniques to provide meaningful understanding of patterns underlying the data, it seems appropriate to shift to supervised learning techniques, which have demonstrated their utility in audio classification applications in previous literature (Catalepe et al., 2007). Random Forests, this study's primary technique of interest, form decision trees with bootstrapped samples and split nodes based on subsets of predictors randomly chosen for each tree. This technique predicts class for new data according to the aggregate of the outcomes of its trees (Liaw & Wiener, 2002).

I split my data with a large ratio of training to testing data such that two-thirds is used for training and one-third for testing. I then tune the optimal quantity of input features for the algorithm to choose for every tree by measuring the out of bag error, which is determined by measuring the error rate of predicting the data not included in the bootstrapped samples within each iteration. I compare the out of bag error with the test error for every possible value for the size of the feature subset from 1 to the full number of numerical features in the featureset, (71) and then choose the value which minimizes both. In this case, I determined the optimal number of decision tree features to be 16, as depicted below:

Figure 3. *Choosing Optimal Feature Subset Size*



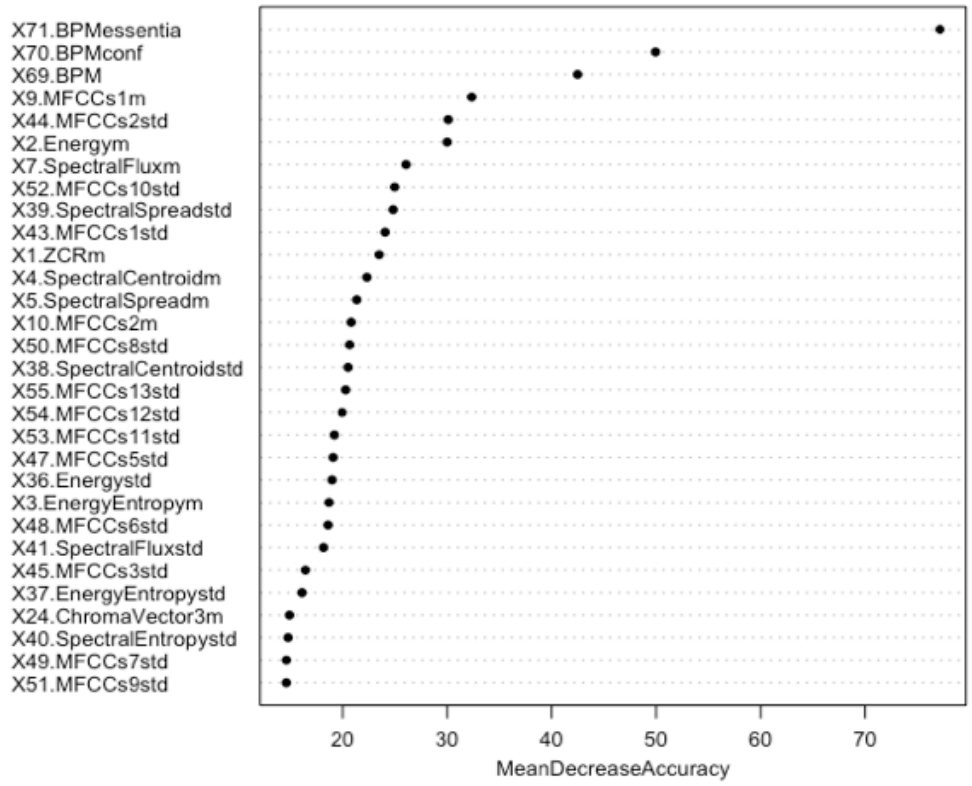
Further, I specify a large number of classification trees (1000) in order to prevent biased classification using the relatively large featureset as well as upward bias in the estimates of out of bag error. I also compute a measure of variable importance within a run of the algorithm, which also requires a large number of classification trees in order to produce results with less variance between runs. This is determined by measuring the change in classification error for each decision tree when out of bag information for a single feature is hidden from the model, while all others are retained. This process is repeated for every feature in the set, and subsequently each variable is ranked in importance based on the magnitude of the error rate change when it is excluded (Liaw & Wiener, 2002).

Results

Analysis of classification demonstrates that Random Forests predict genre of new data reasonably well, classifying about 52.3% of songs correctly according to their pre-coded genre. Examination of the classification rate shows considerable variability however, from 100% classification accuracy for PsyTrance to 15.15% for Techno.

Examination of variable importance reveals significant variation between variables as well. As demonstrated below, rhythmic features seem to be most important for audio classification by a wide margin, with tempo as computed with the Essentia algorithm decreasing a tree's classification accuracy by over 70% on average when information from that feature is masked from the algorithm. Leaving out other BPM variables can severely diminish classification accuracy as well, on average reducing the rate of accurate predictions within trees by over 40 percentage points. The timbral variables which trail the BPM variables most closely in importance decrease classification accuracy by over 30% and include MFCC and spectral energy features. Notably both the non-standardized and standardized representations of the first two MFCC variables appear in the top 15 most important variables.

Figure 4. *Study 2 Variable Importance*



Study 3

Method

Boosting

Given the large variation in correct classification rates between genres for the Random Rorest analysis, it is possible that some audio samples are particularly difficult to classify. Boosting is a technique which classifies via aggregation of decision trees, similarly to Random Forests, however the Boosting algorithm places special importance on the “hardest” samples within the training dataset (Schapire, Freund, Bartlett, & Lee, 1998). Boosting places a uniform

distribution over the subset used to train the first decision tree, and updates this distribution in order to reduce error for each subsequent tree:

$$D_{t+1}(i) = \frac{D_t(i)\exp(-y_i \alpha_t h_t(x_i))}{Z_t}$$

where $\alpha_t = \frac{1}{2}\ln((1 - \varepsilon_t)/\varepsilon_t)$ and Z_t is a scaling factor which transforms D_{t+1} into a simplex vector:

$$\begin{aligned} Z_t &= \sum_{i=1}^{m} D_t(i)\exp(-y_i \alpha_t h_t(x_i)) \\ &= \sum_{i:\, y_i = h_t(x_i)} D_t(i)\exp(-\alpha_t) + \sum_{i:\, y_i \neq h_t(x_i)} D_t(i)\exp(\alpha_t) \\ &= (1 - \varepsilon_t)\exp(-\alpha_t) + \varepsilon_t \exp(\alpha_t) \\ &= 2\sqrt{\varepsilon_t(1 - \varepsilon_t)}\,. \end{aligned}$$

The algorithm arrives at final classification by aggregating the decision of each tree:

$$f(x) = \frac{\sum_{t=1}^{T} \alpha_t h_t(x)}{\sum_{t=1}^{T} \alpha_t}.$$

The numerator of this function describes which variables are considered the "hardest." The boosting algorithm places the heaviest weights the samples for which the numerator returns the smallest values (Schapire et al., 1998). As in my procedure for Random Forests, I set aside two-thirds of data for training and specify 1000 classification trees for the boosting algorithm.

Results

Boosting performs markedly worse than the Random Forest algorithm, achieving a proper classification rate of 40%. As in the Random Forest results, boosting makes the best predictions for PsyTrance and Drum and Bass, and further Dance is again one of the two most

misclassified genres. Boosting classifies Techno music correctly at a much higher rate than did Random Forests, classifying 30.3% samples correctly to Random Forests' 9.1%. Similar variables are important to the Boosting analysis as were for the Random Forests (see Appendix A, Table 15.) The coefficient for the BPMEssentia feature again dwarfs the others, and the first and second components from the MFCC feature vector again appear within the upper 10 ranks for variable importance.

Study 4

Method

Supervised Learning-- Linear Discriminant Analysis.

Linear Discriminant Analysis (LDA) transforms data onto a one-dimensional projection which classifies according to distance derived from featureset data:

$$J(\mathbf{w}) = \frac{\mathbf{w}^T S_B \mathbf{w}}{\mathbf{w}^T S_W \mathbf{w}}$$

where $\boldsymbol{w}$ is the direction on the classification plane, S_B and S_W are between-class and within-class scatter matrices depending on data:

$$S_B = (\mathbf{m}_1 - \mathbf{m}_2)(\mathbf{m}_1 - \mathbf{m}_2)^T$$
$$S_W = \sum_{i\ 1,2} \sum_{\mathbf{x} \in C_i} (\mathbf{x} - \mathbf{m}_i)(\mathbf{x} - \mathbf{m}_i)^T$$

and $\boldsymbol{m_i}$ is the sample mean for each class:

$$\mathbf{m_i} - \frac{1}{n_i} \sum_{\mathbf{x} \in C_i} \mathbf{x}$$

The goal of LDA is to create classes with the greatest possible distance and least possible within-class variance, implemented with this expression:

$$\mathbf{w} = S_W^{-1}(\mathbf{m_1} - \mathbf{m_2})$$

I again set the proportion of training data at two-thirds and select the whole range of features for use with LDA (Alexandre-Cortizo, Rosa-Zurera, & Lopez-Ferreras, 2006).

Results

LDA performs slightly worse than the Random Forest model (52.3% classified correctly), predicting 48.6% of samples correctly. The range of classification error for each genre is considerably large, however the variance of error is smaller than that of either of the other two techniques. Unlike those techniques, LDA had no genre achieving a less than 10% proper classification rate, but it also did not classify any one genre at a rate of 90% or greater.

Study 5

Method

Random Forests.

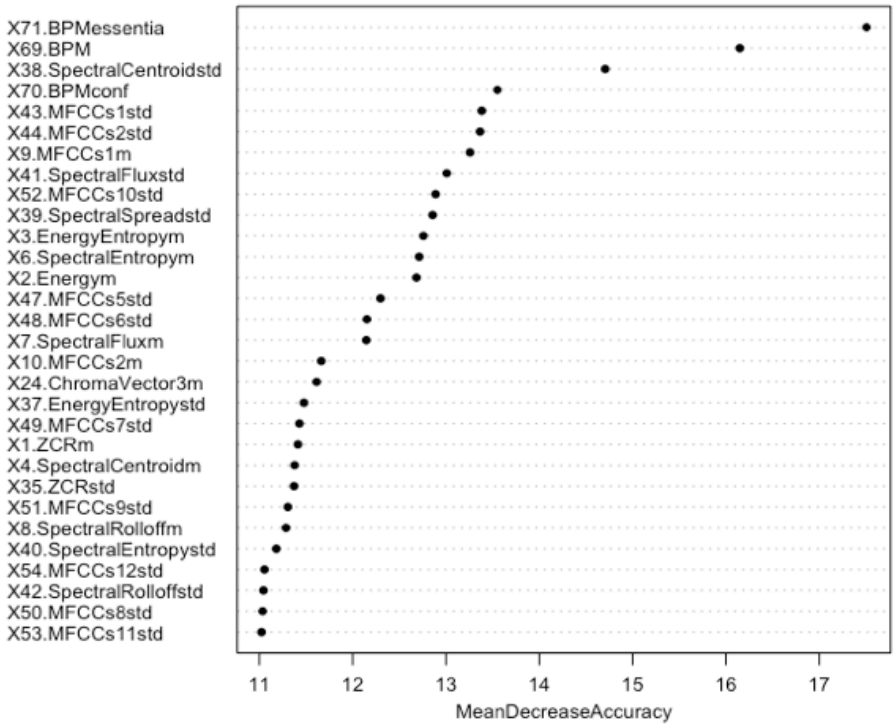
Out of the three supervised learning techniques, Random Forests perform the best. Perhaps it is possible to boost its correct classification rate even higher. Liaw & Wiener (2002) recommend trying different values for the number of decision trees in the model, such as one-

half the original amount. In this study, I set the number of trees at that value (500) and partition the data so that two-thirds of the data are trained on.

Results

Random Forests analysis with 500 trees classifies audio samples correctly 44.8% of the time, worse than LDA (48.6% correct) or Random Forests with 1000 trees (52.3% correct). Variation in error rates between genres follows patterns similar to those displayed in previous studies. Psytrance is among the best-predicted genres, with 90.9% classified correctly, while the correct classification rate for the worst-predicted genre, Techno, is 9.09%. Familiar patterns also emerge for variable importance, with BPM variables as well as first and second MFCC variables appearing in the top 10. The largest difference between variable importance for this study and that from Study 2 is that the scale of the changes are compressed here. In the present model, masking the most important feature's data results in a mean increase of 18 percentage points for classification error rate as compared to an increase of 70 percentage points in error rate for a comparably important variable in Study 2.

Figure 5. *Study 4 Variable Importance*



General Discussion

Overall, the findings of the five exploratory analyses indicate that Random Forest classification is relatively well-suited to predicting the genre of an audio sample represented by numerical features. Its rate of successful classification exceeds that of LDA, a method which has precedent in the literature for applications in audio classification, yet the features they weighted most heavily in analysis were largely the same, as was the case for all four supervised learning trials. In particular, BPM variables were highly influential to the classifying process, perhaps because their variance values are quite high in absolute terms when compared to other features in the dataset (SD_{BPM} = 46.64; $SD_{BPMEssentia}$ = 17.10; $SD_{MFCCs1m}$ = 1.18). This perspective would not explain however, the huge relevance of BPMEssentia in several models,

though its variance value is less that of BPM in absolute terms. Both of them however, are much larger than the variance of MFCCs1m, another consistently important variable for supervised learning models.

While the exceptional classification success rates of 80-90% found in the literature using LDA were not replicated, I can be reasonably certain that this does not reflect poorly on the utility of Random Forests. For this dataset, Random Forests outperformed the well-established LDA method in classification error rate. This comparison is more relevant than one judging the utility of Random Forests against an error rate obtained in another study primarily because of heterogeneity in data used between studies. Pre-classifying subjective music genre information can introduce noise if not accomplished in a systematic way,

Limitations

This highlights a potential limitation to my analysis in that the generative process for the pre-classified genre labels is not documented. This could pose a problem for the validity of the given pre-classified genre labels. Several of the genres listed in the values of the class feature are subgenres of one another or otherwise seem to be closely related, such as “HardDance” and “HardcoreHardTechno.” If it is the case that there is no systematic way to discern the difference between highly similar genre categories, then quantitative classification is unlikely to be accurate. Though this may explain some of the variance in error rate for all models between classes, the most egregious cases of genres with high rates of misclassification occur not for the more granular genres, but for the broadest. Throughout analyses, Dance and Techno repeatedly emerge as among the most difficult to classify. Without information about the classification process however, I cannot comment on whether this may represent a flaw of the

models, or if these are merely catch-all categories which store samples that do not fall in with one of the more popular, granular subgenres.

Future Research

Further study could illuminate these issues by examining datasets which include systematic information concerning their pre-classification decisions. That way, the researcher could determine whether a model needs additional tuning, or if certain classes are simply ill-suited for a classification framework. If the latter is the case, perhaps new classes within catch-all classes could be created through clustering. Since these clusters would not have meaningful names, as many of the pre-classified genres provide, the researcher would have to examine the underlying data in order to make judgments on the musical qualities of these clustered audio samples.

In addition, perhaps further gain in correct classification rates may be accomplished by employing feature reduction techniques. Though I concluded early in the design that sufficient variance between features of similar types, such as MFCC, precluded the use of dimensionality reduction techniques, examination of variable importance plots reveals that a small handful of features are extremely influential across analyses, namely BPM and some MFCC features. Further, many variables seemed to have very little effect on error rates in comparison, such as several standardized measures. It is likely that these added little with the original scale variables accounting for these components of the audio signal already, and could be removed from analysis with little negative impact on successful classification rate. PCA could prove useful in this regard for pruning and/or aggregating superfluous measures, or perhaps meaningful

variables could be chosen from the featureset based on their rank in a variable importance table.

Conclusion

Exploratory classification of audio samples with Random Forests demonstrates the utility of the algorithm for music genre classification. Random Forest-based analysis provided higher rates of correct classifications when compared to other supervised learning techniques such as LDA and boosting. Unsupervised clustering did not prove useful for this application. However, the question of whether other types of dimensionality reduction, such as PCA, may provide additional accuracy remains unanswered. Future study should examine the extent to which dimensionality reduction techniques can improve the predictions of Random Forest models.

Appendix A: Tables and Figures

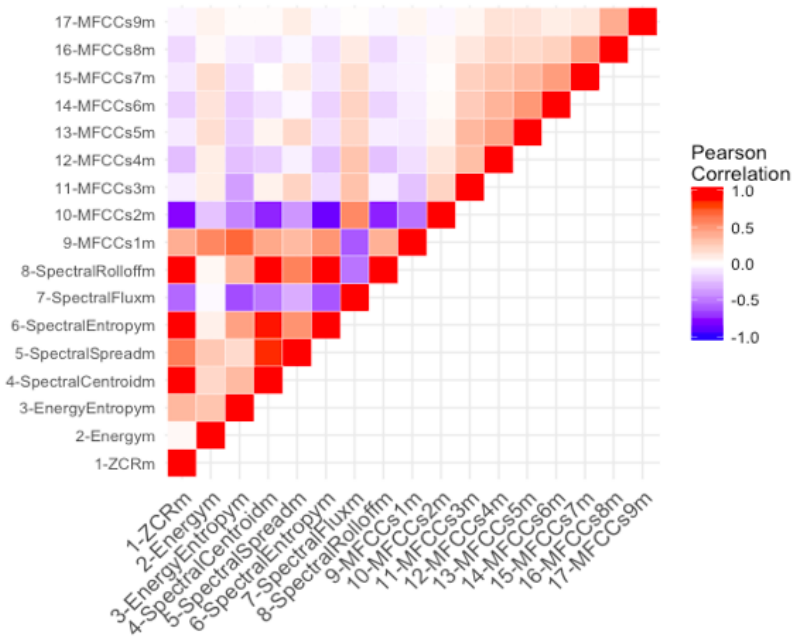
Table 1. *Summary Statistics for Features 1-36*

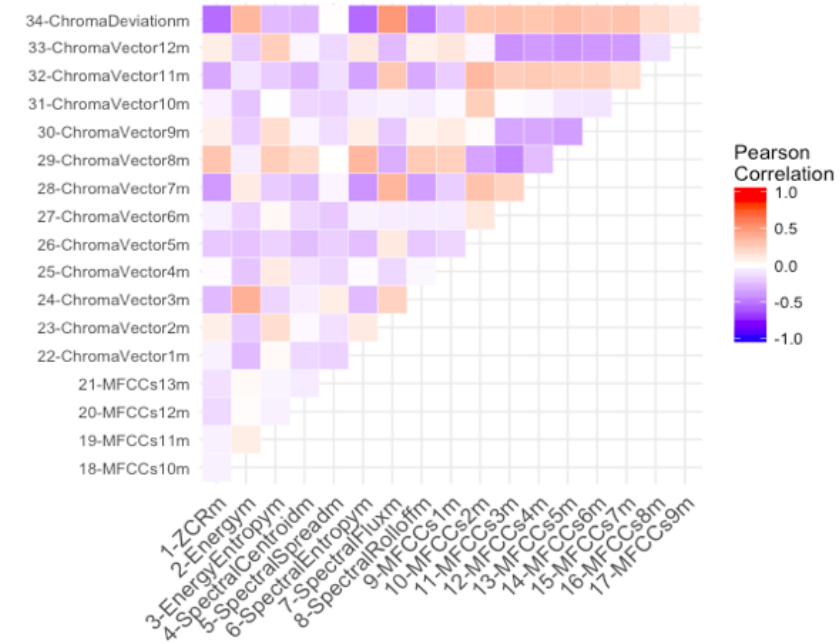
Statistic	N	Mean	St. Dev.	Min	Max
1-ZCRm	2,300	0.108	0.037	0.017	0.247
2-Energym	2,300	0.104	0.041	0.005	0.282
3-EnergyEntropym	2,300	3.111	0.071	2.746	3.254
4-SpectralCentroidm	2,300	0.236	0.044	0.083	0.374
5-SpectralSpreadm	2,300	0.241	0.020	0.152	0.301
6-SpectralEntropym	2,300	0.906	0.342	0.034	1.959
7-SpectralFluxm	2,300	0.011	0.005	0.003	0.053
8-SpectralRolloffm	2,300	0.191	0.080	0.012	0.484
9-MFCCs1m	2,300	-23.167	1.177	-30.379	-20.129
10-MFCCs2m	2,300	1.681	0.559	-0.019	4.163
11-MFCCs3m	2,300	0.108	0.288	-1.268	1.475
12-MFCCs4m	2,300	0.324	0.187	-0.446	0.976
13-MFCCs5m	2,300	0.102	0.146	-0.630	0.729
14-MFCCs6m	2,300	0.144	0.120	-0.431	0.642
15-MFCCs7m	2,300	0.057	0.108	-0.612	0.481
16-MFCCs8m	2,300	0.078	0.100	-0.329	0.474
17-MFCCs9m	2,300	0.040	0.096	-0.439	0.515
18-MFCCs10m	2,300	0.062	0.091	-0.414	0.486
19-MFCCs11m	2,300	0.031	0.083	-0.310	0.332
20-MFCCs12m	2,300	0.041	0.070	-0.333	0.356
21-MFCCs13m	2,300	0.018	0.061	-0.245	0.306
22-ChromaVector1m	2,300	0.010	0.005	0.001	0.052
23-ChromaVector2m	2,300	0.004	0.003	0.0002	0.046
24-ChromaVector3m	2,300	0.096	0.043	0.010	0.403
25-ChromaVector4m	2,300	0.008	0.004	0.001	0.050
26-ChromaVector5m	2,300	0.015	0.007	0.003	0.068
27-ChromaVector6m	2,300	0.008	0.004	0.002	0.041
28-ChromaVector7m	2,300	0.088	0.036	0.020	0.298
29-ChromaVector8m	2,300	0.002	0.001	0.0001	0.027
30-ChromaVector9m	2,300	0.005	0.002	0.001	0.033
31-ChromaVector10m	2,300	0.010	0.005	0.002	0.046
32-ChromaVector11m	2,300	0.034	0.016	0.006	0.169
33-ChromaVector12m	2,300	0.004	0.002	0.0002	0.027
34-ChromaDeviationm	2,300	0.046	0.011	0.017	0.112
35-ZCRstd	2,300	0.055	0.020	0.004	0.155
36-Energystd	2,300	0.058	0.020	0.003	0.158

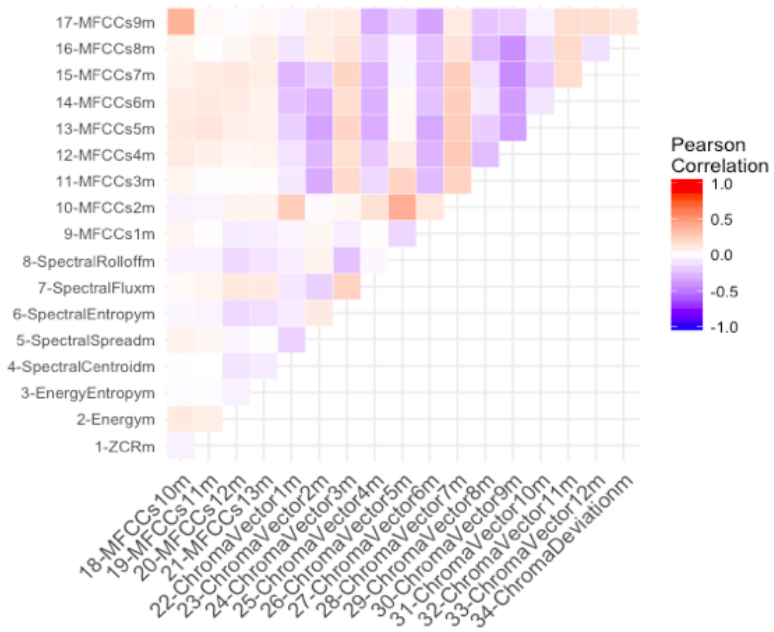
Table 2. *Summary Statistics for Features 37-71*

Statistic	N	Mean	St. Dev.	Min	Max
37-EnergyEntropystd	2,300	0.170	0.071	0.039	0.565
38-SpectralCentroidstd	2,300	0.059	0.019	0.017	0.129
39-SpectralSpreadstd	2,300	0.028	0.009	0.007	0.062
40-SpectralEntropystd	2,300	0.463	0.145	0.026	0.955
41-SpectralFluxstd	2,300	0.010	0.006	0.001	0.070
42-SpectralRolloffstd	2,300	0.136	0.052	0.006	0.299
43-MFCCs1std	2,300	1.167	0.400	0.327	4.550
44-MFCCs2std	2,300	0.580	0.154	0.241	1.270
45-MFCCs3std	2,300	0.411	0.090	0.190	0.848
46-MFCCs4std	2,300	0.324	0.061	0.171	0.639
47-MFCCs5std	2,300	0.280	0.049	0.166	0.528
48-MFCCs6std	2,300	0.254	0.041	0.153	0.446
49-MFCCs7std	2,300	0.240	0.037	0.141	0.411
50-MFCCs8std	2,300	0.228	0.034	0.136	0.384
51-MFCCs9std	2,300	0.221	0.034	0.128	0.385
52-MFCCs10std	2,300	0.214	0.033	0.124	0.338
53-MFCCs11std	2,300	0.207	0.031	0.109	0.333
54-MFCCs12std	2,300	0.195	0.027	0.108	0.336
55-MFCCs13std	2,300	0.181	0.024	0.103	0.303
56-ChromaVector1std	2,300	0.010	0.005	0.002	0.049
57-ChromaVector2std	2,300	0.004	0.003	0.0002	0.035
58-ChromaVector3std	2,300	0.064	0.023	0.010	0.173
59-ChromaVector4std	2,300	0.008	0.005	0.001	0.061
60-ChromaVector5std	2,300	0.014	0.006	0.004	0.071
61-ChromaVector6std	2,300	0.008	0.004	0.001	0.043
62-ChromaVector7std	2,300	0.068	0.020	0.018	0.143
63-ChromaVector8std	2,300	0.003	0.002	0.0002	0.027
64-ChromaVector9std	2,300	0.005	0.003	0.0005	0.028
65-ChromaVector10std	2,300	0.010	0.005	0.002	0.048
66-ChromaVector11std	2,300	0.031	0.012	0.004	0.096
67-ChromaVector12std	2,300	0.004	0.002	0.0003	0.029
68-ChromaDeviationstd	2,300	0.020	0.004	0.006	0.041
69-BPM	2,300	148.226	46.641	63.158	600.000
70-BPMconf	2,300	0.169	0.057	0.073	0.423
71-BPMessentia	2,300	120.949	17.102	61	188

Figures 1-4. *Heatmaps Displaying Correlation Coefficients for all Continuous Features, Excluding Standardized Features*






17-MFCCs9m
16-MFCCs8m
15-MFCCs7m
14-MFCCs6m
13-MFCCs5m
12-MFCCs4m
11-MFCCs3m
10-MFCCs2m
9-MFCCs1m
8-SpectralRolloffm
7-SpectralFluxm
6-SpectralEntropym
5-SpectralSpreadm
4-SpectralCentroidm
3-EnergyEntropym
2-Energym
1-ZCRm
18-MFCCs10m
19-MFCCs11m
20-MFCCs12m
21-MFCCs13m
22-ChromaVector1m
23-ChromaVector2m
24-ChromaVector3m
25-ChromaVector4m
26-ChromaVector5m
27-ChromaVector6m
28-ChromaVector7m
29-ChromaVector8m
30-ChromaVector9m
31-ChromaVector10m
32-ChromaVector11m
33-ChromaVector12m
34-ChromaDeviationm
Pearson Correlation
1.0
0.5
0.0
-0.5
-1.0

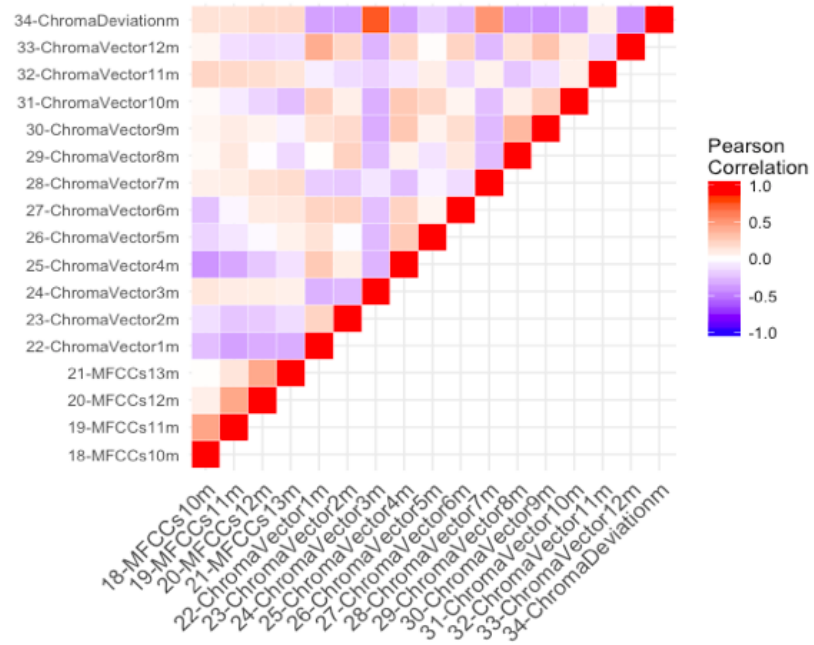

34-ChromaDeviationm
33-ChromaVector12m
32-ChromaVector11m
31-ChromaVector10m
30-ChromaVector9m
29-ChromaVector8m
28-ChromaVector7m
27-ChromaVector6m
26-ChromaVector5m
25-ChromaVector4m
24-ChromaVector3m
23-ChromaVector2m
22-ChromaVector1m
21-MFCCs13m
20-MFCCs12m
19-MFCCs11m
18-MFCCs10m
Pearson Correlation
1.0
0.5
0.0
-0.5
-1.0

Figure 5. *Single Linkage Dendrogram with Blue Line Cut*

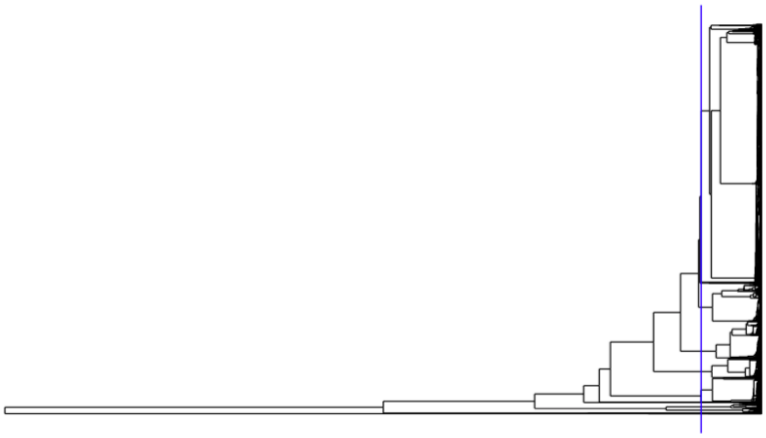


Figure 6. *Complete Linkage Dendrogram with Blue Line Cut*

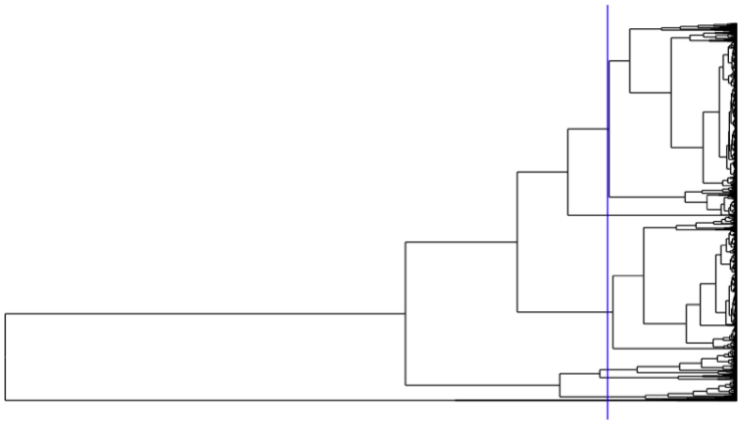


Figure 7. *Average Linkage Dendrogram with Blue Line Cut*

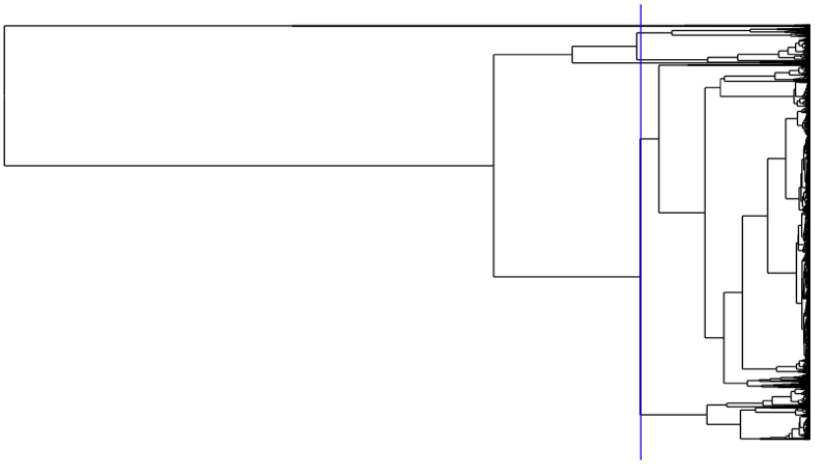


Table 3. *Results of Single Linkage Hierarchical Clustering*

	Cluster	Frequency
1	Cluster 1	1,522
2	Cluster 2	215
3	Cluster 3	144
4	Cluster 4	226
5	Cluster 5	116
6	Cluster 6	34
7	Cluster 7	2
8	Cluster 8	6
9	Cluster 9	23
10	Cluster 10	6
11	Cluster 11	1
12	Cluster 12	1
13	Cluster 13	1
14	Cluster 14	1
15	Cluster 15	1
16	Cluster 16	1

Tables 4-7. *Results of Average Linkage Hierarchical Clustering*

Cluster	Frequency
Cluster 1	414
Cluster 2	822
Cluster 3	51
Cluster 4	97
Cluster 5	165
Cluster 6	138
Cluster 7	1
Cluster 8	2
Cluster 9	4
Cluster 10	43
Cluster 11	119
Cluster 12	11
Cluster 13	1
Cluster 14	1
Cluster 15	8
Cluster 16	33

Cluster	Frequency
Cluster 17	7
Cluster 18	5
Cluster 19	50
Cluster 20	1
Cluster 21	40
Cluster 22	61
Cluster 23	13
Cluster 24	15
Cluster 25	24
Cluster 26	13
Cluster 27	5
Cluster 28	10
Cluster 29	8
Cluster 30	6
Cluster 31	6
Cluster 32	2

Cluster	Frequency
Cluster 33	2
Cluster 34	8
Cluster 35	29
Cluster 36	6
Cluster 37	15
Cluster 38	5
Cluster 39	2
Cluster 40	1
Cluster 41	1
Cluster 42	2
Cluster 43	2
Cluster 44	2
Cluster 45	1
Cluster 46	1
Cluster 47	14
Cluster 48	3

Cluster	Frequency
Cluster 49	1
Cluster 50	1
Cluster 51	1
Cluster 52	2
Cluster 53	1
Cluster 54	1
Cluster 55	2
Cluster 56	1
Cluster 57	1
Cluster 58	1
Cluster 59	1
Cluster 60	3
Cluster 61	4
Cluster 62	1
Cluster 63	6
Cluster 64	1
Cluster 65	1
Cluster 66	1

Tables 8-13. *Results of Complete Linkage Clustering*

Cluster	Frequency
Cluster 1	302
Cluster 2	103
Cluster 3	109
Cluster 4	443
Cluster 5	29
Cluster 6	48
Cluster 7	128
Cluster 8	37
Cluster 9	1
Cluster 10	6
Cluster 11	4
Cluster 12	32
Cluster 13	32
Cluster 14	97
Cluster 15	11
Cluster 16	1

Cluster	Frequency
Cluster 17	1
Cluster 18	45
Cluster 19	21
Cluster 20	8
Cluster 21	10
Cluster 22	18
Cluster 23	7
Cluster 24	5
Cluster 25	89
Cluster 26	41
Cluster 27	1
Cluster 28	27
Cluster 29	144
Cluster 30	33
Cluster 31	13
Cluster 32	6

Cluster	Frequency
Cluster 33	2
Cluster 34	13
Cluster 35	5
Cluster 36	15
Cluster 37	15
Cluster 38	13
Cluster 39	9
Cluster 40	10
Cluster 41	9
Cluster 42	8
Cluster 43	43
Cluster 44	23
Cluster 45	6
Cluster 46	6
Cluster 47	2
Cluster 48	2

Cluster	Frequency
Cluster 49	12
Cluster 50	87
Cluster 51	8
Cluster 52	9
Cluster 53	5
Cluster 54	1
Cluster 55	5
Cluster 56	5
Cluster 57	19
Cluster 58	10
Cluster 59	2
Cluster 60	1
Cluster 61	7
Cluster 62	3
Cluster 63	1
Cluster 64	8
Cluster 65	2

Cluster	Frequency
Cluster 66	2
Cluster 67	2
Cluster 68	1
Cluster 69	1
Cluster 70	8
Cluster 71	3
Cluster 72	1
Cluster 73	13
Cluster 74	1
Cluster 75	1
Cluster 76	2
Cluster 77	6
Cluster 78	11
Cluster 79	1
Cluster 80	1
Cluster 81	4

Cluster	Frequency
Cluster 82	2
Cluster 83	1
Cluster 84	1
Cluster 85	1
Cluster 86	1
Cluster 87	3
Cluster 88	1
Cluster 89	2
Cluster 90	1
Cluster 91	6
Cluster 92	1
Cluster 93	2
Cluster 94	1
Cluster 95	1

Table 14. *Correct Classification Rates by Genre for Studies 2-5*.

	RF 1	Boosting	LDA	RF 2
PsyTrance	96.97%	90.91%	87.88%	90.91%
DrumAndBass	90.91%	90.91%	87.88%	66.67%
HardDance	87.88%	69.7%	81.82%	78.79%
Trance	78.79%	63.64%	60.61%	81.82%
GlitchHop	75.76%	78.79%	57.58%	60.61%
HardcoreHardTechno	72.73%	30.3%	66.67%	66.67%
Breaks	69.7%	63.64%	45.45%	36.36%
Dubstep	66.67%	24.24%	48.48%	54.55%
Minimal	63.64%	45.45%	54.55%	63.64%
BigRoom	60.61%	45.45%	63.64%	57.58%
FutureHouse	45.45%	27.27%	24.24%	21.21%
ReggaeDub	45.45%	21.21%	57.58%	42.42%
ElectronicaDowntempo	42.42%	18.18%	36.36%	45.45%
FunkRAndB	42.42%	30.3%	54.55%	57.58%
DeepHouse	39.39%	24.24%	24.24%	21.21%
House	39.39%	36.36%	36.36%	27.27%
HipHop	36.36%	12.12%	42.42%	24.24%
ProgressiveHouse	36.36%	15.15%	45.45%	45.45%
IndieDanceNuDisco	30.3%	45.45%	42.42%	21.21%
ElectroHouse	27.27%	21.21%	27.27%	18.18%
TechHouse	27.27%	24.24%	33.33%	18.18%
Dance	18.18%	9.09%	18.18%	21.21%
Techno	9.09%	30.3%	21.21%	9.09%

Table 15. *Variable Importance for Study 3*

Variable	Influence
X71.BPMessentia	35.181
X4.SpectralCentroidm	6.177
X7.SpectralFluxm	6.018
X44.MFCCs2std	5.704
X10.MFCCs2m	5.144
X50.MFCCs8std	3.108
X47.MFCCs5std	3.036
X69.BPM	2.950
X9.MFCCs1m	2.627
X38.SpectralCentroidstd	2.479
X52.MFCCs10std	2.449
X41.SpectralFluxstd	2.131
X54.MFCCs12std	2.112
X42.SpectralRolloffstd	2.072
X43.MFCCs1std	2.029
X55.MFCCs13std	1.533
X70.BPMconf	1.496
X37.EnergyEntropystd	1.335
X2.Energym	1.134
X53.MFCCs11std	1.122
X34.ChromaDeviationm	1.020
X49.MFCCs7std	0.997
X39.SpectralSpreadstd	0.984
X56.ChromaVector1std	0.930
X35.ZCRstd	0.919
X51.MFCCs9std	0.828

Appendix B

Description of Features

1. Zero Crossing Rate

The rate of sign-changes of the values of the audio waveform signal during the duration of a particular frame, or the zero crossing rate, is calculated by normalizing the count of instances that the signal crosses the zero axis:

$$ZCR = \frac{1}{2}\left(\sum_{n=1}^{N-1} |sign(s(n)) - sign(s(n-1))|\right)\frac{F_s}{N},$$

where *s(n)* represents the signal, *N* describes the quantity of samples within that input signal, and F_s, the sampling frequency, is derived from:

$$sign(x) = \begin{cases} 1 & \text{if } x > 0 \\ 0 & \text{if } x = 0 \\ -1 & \text{if } x < 0. \end{cases}$$

The zero-crossing rate parameter has utility for discerning between speech and music, as well as classifying genres of music.

2. Energy

This measure is calculated by taking the sum of squares of the values of the audio waveform signal and normalizing these values by the duration of the frame(s) taken from the signal.

3. Entropy of Energy

Entropy of energy provides information concerning the presence of abrupt changes in

the audio signal waveform. The measure is derived from the entropy of the signal's normalized energy values, as computed with the normalization procedure as described above.

4. Audio Spectral Centroid

This feature gives insight about the presence of high- and low- frequencies within an audio signal. The center of gravity of power coefficients which make up the audio spectrum represent sum-weighted centroid frequencies and describe the degree to which listeners perceive a quality of sharpness in the audio sample. This measure rescales coefficients below 62.5 Hz into a single power coefficient so that these very low frequency coefficients do not inordinately influence calculation of the centroid.

The input spectrum *P(k)* is defined by:

$$f(k) = k\Delta F \quad (0 \le k \le N_{FT}/2),$$

where *k* represents the frequencies extracted from a given audio frame mapped into categorical values as defined by:

$$k = round(f/\Delta F) \ (0 \le f \le F_s/2),$$

in which ΔF, the range of frequencies within two particular Fast Fourier Transform discrete bins is given by F_S / N_{FT}, or the sampling frequency divided by the Fast Fourier Transform's size, where F_S is defined the same way as described for the Zero Crossing Rate.

Rescaling takes place by:

$$K_{low} = floor(62.5/\Delta F),$$

Here, *floor(x)* outputs the largest integer satisfying the condition that it is less than or equal to the input, and ΔF again represents an interval of frequencies between discrete Fast

Fourier Transform categories. The function creates a new spectrum of coefficients *P'(k')* related to the original:

$$P'(k') = \begin{cases} \sum_{k=0}^{K_{low}} P(k) & \text{for } k' = 0 \\ P(k' + K_{low}) & \text{for } 1 \le k' \le \frac{N_{FT}}{2} - K_{low}. \end{cases}$$

Further, the new frequencies *k* are given by:

$$f'(k') = \begin{cases} 31.25 & \text{for } k' = 0 \\ f(k' + K_{low}) & \text{for } 1 \le k' \le \frac{N_{FT}}{2} - K_{low}, \end{cases}$$

where *f(k)* is again given by:

$$f(k) = k\Delta F \quad (0 \le k \le N_{FT}/2),$$

and the 0th frequency of the spectrum is given by *f'*(0) = 31.25 Hz, where this value represents the middle of the low-frequency band.

After the spectrum of power coefficients has been transformed as to not over-weight the presence of very-low frequency components, the spectral centroid may be defined by the following, weighting the transformed frequencies *f'(k')* by their transformed power coefficients in *P'(k')*:

$$ASC = \frac{\sum_{k'=0}^{(N_{FT}/2)-K_{low}} \log_2\left(\frac{f'(k')}{1000}\right) P'(k')}{\sum_{k'=0}^{(N_{FT}/2)-K_{low}} P'(k')}$$

The centroid is scaled by log-frequency in a manner consistent with the range of frequencies audible to the human ear, in order to accurately measure the degree to which listeners perceive sharpness within a given audio sample.

5. Spectral Spread

This measure provides information on how the power spectrum is distributed surrounding the spectral centroid, such that a higher value reflects a wider or more diffuse distribution across the range of possible frequencies. This measure may inform differentiation between noise and tonal sounds. It is given by a function which calculates the spectrum's root-mean-square deviation from the spectral centroid within a given audio frame:

$$ASS = \sqrt{\frac{\sum_{k'=0}^{(N_{FT}/2)-K_{low}} \left[\log_2\left(\frac{f'(k')}{1000}\right) - ASC\right]^2 P'(k')}{\sum_{k'=0}^{(N_{FT}/2)-K_{low}} P'(k')}},$$

where power spectrum coefficients *P'(k')* and frequencies *f'(k')* are transformed in the same manner as in the calculation of the Spectral Centroid.

6. Spectral Entropy

This measure describes the entropy of the signal's spectral energies, normalized by weighting frequencies according to power coefficients, as described in section 4., "Audio Spectral Centroid".

7. Spectral Flux

This measure can lend insight to applications aiming to discern between speech and music. Spectral Flux values represent the average dispersion of the signal amplitude distribution between consecutive audio frames, calculated by taking the averaged squared difference between the variation of two adjacent spectral distributions:

$$SF = \frac{1}{LN_{FT}} \sum_{k=0}^{L-1} \sum_{k=0}^{N_{FT}-1} [\log(|S_l(k)| + \delta) - \log(|S_{l-1}(k)| + \delta)]^2,$$

where $S_l(k)$ refers to a Discrete Fourier Transform frame with index l, N_{FT} describes the size of the Discrete Fourier Transform, L gives the total count of frames in the signal, and δ is a parameter included to prevent calculation overflow.

Spectral flux values convey information concerning spectral change between audio frames. Speech samples display greater values than music samples on average, and further environmental sound samples result in the highest values for spectral change.

8. Spectral Rolloff

The spectral rolloff frequency is a feature which has utility in discerning between voiced and unvoiced speech. This frequency is the cutoff below which 85-95% of the spectrum's magnitude distribution is located, and is given by the following example function, where the threshold of spectrum magnitude distribution is chosen by the researcher to delineate 85% of the spectrum under the spectral rolloff frequency:

$$\sum_{k=0}^{K_{roll}} |S(k)| = 0.85 \sum_{k=0}^{N_{FT}/2} |S(k)|,$$

and K_{roll} is the discrete frequency category corresponding to the spectral rolloff frequency.

9. Mel-Frequency Cepstrum Coefficients

The vector of Mel-Frequency Cepstrum Coefficient (MFCC) features forms a popular, effective measure for speech recognition and other representations of speech and music signals. In this case, the MFCC vector is comprised of 12 component features.

A *cepstrum* is a transformation which allows an extraction signal *e(t)* and the impulse response of a filter *h(t)* to be separated from the model of a signal *s(t),* represented as *h(t)s(t) = e(t)h(t)* according to this framework. Using cepstrum transformation, signal energy may be computed in frequency bands centered according to the logarithmic mel scale, which is composed of pitches such that the human ear perceives each as equidistant from one another.

This process begins with the division of signal *s(n)* into frames of N_ω samples, which overlap surrounding frames by about 50%. Discontinuity between frames is then smoothed with the Hanning windowing function:

$$w(n) = \frac{1}{2}\left\{1 - \cos\left[\frac{2\pi}{N_w}\left(n + \frac{1}{2}\right)\right]\right\} \quad (0 \le n \le N_w - 1).$$

Then, Fast Fourier Transforms on each frame derive a magnitude spectrum from which a sparse representation is achieved through application of a mel-filter. This filter measures the log-energy of the spectrum, which serves as the input for a Discrete Cosine Transform function that derives cepstral coefficients:

$$c_i = \sum_{j=1}^{N_f}\left\{\log(E_j)\cos\left[i\left(j - \frac{1}{2}\right)\frac{\pi}{N_f}\right]\right\} \quad (1 \le i \le N_c).$$

In that function, c_i represents an MFCC with order *I*, E_j is the energy of the spectrum measured by the *j*th mel filter, N_f is the amount of mel filters, often $N_f = 24$, and N_c gives the quantity of MFCCs extracted from each frame, typically $N_c = 12$.

Linear regression over a small subset of audio frames captures information about variation in the spectrum across time:

$$\Delta c_i(l) = -c_i(l-2) - \frac{1}{2}c_i(l-1) + \frac{1}{2}c_i(l+1) + c_i(l+2)$$

and:

$$\Delta\Delta c_i(l) = c_i(l-2) - \frac{1}{2}c_i(l-1) - c_i(l) - \frac{1}{2}c_i(l+1) + c_i(l+2),$$

in which $c_i(l)$ is a coefficient with order *i* from an audio frame with index *l*, $\Delta\, c_i(l)$ the coefficient's derivative, and $\Delta\Delta\, c_i(l)$, the acceleration. The coefficients $c_i(l)$, coupled with the Δ and $\Delta\Delta$, comprise the vector of MFCCs.

10. Chroma Vector

The vector of Chromatic ("Chroma") features includes the power coefficients of the audio signal represented in discrete categories corresponding to the 12-note pitch scale typical of Western music.

11. Chroma Deviation

This feature is calculated by taking the standard deviation of all 12 Chroma coefficients which make up the Chroma Vector. It provides information concerning the distribution of pitch throughout the spectrum.

Note: Each of the features listed above are also represented within the dataset in standardized format.

12. BPM

The Beats per Minute (BPM) feature gives information about the tempo of a given audio sample. This measure is calculated by taking the time distances of maximum values of spectral energies over a sequence of audio frames. The most frequently occurring time distance within this sequence is then used to compute the dominant beats per minute value for the audio sample.

13. BPMconf

This feature describes the dominance of the calculated BPM value within the interval of audio frames as described above, providing information quantifying the certainty of the corresponding calculated BPM value. It is computed by taking the ratio of the frequency value for the most commonly occurring time distance between spectral energy peaks to the sum of the frequencies for all detected peaks (Giannakopoulos, 2015; Kim, Moreau, & Sikora, 2006).

14. BPMEssentia

This feature provides an alternative measure of Beats per Minute computed by the Essentia C++ audio extraction library. The Essentia BPM calculation algorithm computes a novelty curve by taking the discrete derivative of the log-compressed spectral energy values:

$$\text{novelty function } \Delta : [1 : T-1] \to \mathbb{R}:$$

$$\Delta(t) := \sum_{k=1}^{K} |Y(k, t+1) - Y(k, t)|_{\geq 0}$$

for $t \in [1 : T-1]$, where T indicates the number of frames, K is the quantity of power coefficients indexed by k, Y represents log-transformed spectral energy values, $|x|_{\geq 0} := x$ when x is a non-negative real number and $|x|_{\geq 0} := 0$ when x is a negative real number.

The output of this function is then passed through 5 filter bands and transformed by half-wave rectification, which subtracts the local average and retains only the positive components. The results of filtering are plotted in five novelty curves which are sum-weighted to form the final novelty curve. The curve represents the time intervals between spectral energy peaks, and the BPM measure is derived from the time interval with the greatest incidence (Bogdanov et al., 2013; Grosche & Müller, 2009).

15. Class

This feature contains genre information for each audio sample for use with supervised classification methods. Genre labels indicate one of the following 23 electronic music genres: BigRoom, Breaks, Dance, DeepHouse, DrumAndBass, Dubstep, ElectroHouse, ElectronicaDowntempo, FunkRAndB, FutureHouse, GlitchHop, HardcoreHardTechno, HardDance, HipHop, House, IndieDanceNuDisco, Minimal, ProgressiveHouse, PsyTrance, ReggaeDub, TechHouse, Techno, or Trance (Giannakopoulos, 2015; Kim, Moreau, & Sikora, 2006).

References

Alexandre-Cortizo, E., Rosa-Zurera, M., Lopez-Ferreras, F. (2006). Application of Fisher Linear Discriminant Analysis to Speech/Music Classification. In Computer as a Tool EUROCON (2005).

Banitalebi-Dehkordi, M., & Banitalebi-Dehkordi, A. (2014). Music genre classification using spectral analysis and sparse representation of the signals. Journal of Signal Processing Systems, 74(2), 273-280.

Bogdanov, D., Wack N., Gómez E., Gulati S., Herrera P., Mayor O., et al. (2013). ESSENTIA: an Audio Analysis Library for Music Information Retrieval. International Society for Music Information Retrieval Conference. In ISMIR'13 (pp. 493-498).

Cataltepe, Z., Yaslan, Y., & Sonmez, A. (2007). Music genre classification using MIDI and audio features. EURASIP Journal on Advances in Signal Processing, 2007(1), 036409.

Complete-linkage clustering. (n.d.) In *Wikipedia.* Retrieved from https://en.wikipedia.org/wiki/Complete-linkage_clustering

Costa, Y. M., Oliveira, L. S., Koerich, A. L., Gouyon, F., & Martins, J. G. (2012). Music genre classification using LBP textural features. Signal Processing, 92(11), 2723-2737.

Dannenberg, R. B., Foote, J., Tzanetakis, G., & Weare, C. (2001, September). Panel: New directions in Music Information Retrieval. In ICMC.

Gantz, J., Chute, C., Manfrediz, A., Minton, S., Reinsel, D., Schlichting, W., & Toncheva, A. (2008). The Diverse and Exploding Digital Universe. White Paper.

Gantz, J., & Reinsel, D. (2011). Extracting value from chaos. IDC iview, 1142(2011), 1-12.

Giannakopoulos, T. (2015). pyaudioanalysis: An open-source python library for audio signal analysis. PloS one, 10(12), e0144610.

Grosche, P., & Müller, M. (2009). A Mid-Level Representation for Capturing Dominant Tempo and Pulse Information in Music Recordings. In ISMIR (pp. 189-194).

Kursa, M., Rudnicki, W., Wieczorkowska, A., Kubera, E., & Kubik-Komar, A. (2009, September). Musical instruments in random forest. In International Symposium on Methodologies for Intelligent Systems (pp. 281-290). Springer, Berlin, Heidelberg.

Kim, H. G., Moreau, N., & Sikora, T. (2006). MPEG-7 audio and beyond: Audio content indexing and retrieval. John Wiley & Sons.

Laurier, C., Grivolla, J., & Herrera, P. (2008, December). Multimodal music mood classification using audio and lyrics. In Machine Learning and Applications, 2008. ICMLA'08. Seventh International Conference on (pp. 688-693). IEEE.

Liaw, A., & Wiener, M. (2002). Classification and regression by randomForest. R news, 2(3), 18-22.

Lippens, S., Martens, J. P., & De Mulder, T. (2004, May). A comparison of human and automatic musical genre classification. In Acoustics, Speech, and Signal Processing, 2004. Proceedings.(ICASSP'04). IEEE International Conference on (Vol. 4, pp. iv-iv). IEEE.

McKay, C., & Fujinaga, I. (2004, October). Automatic Genre Classification Using Large High-Level Musical Feature Sets. In ISMIR (Vol. 2004, pp. 525-530).

Pachet, F., & Cazaly, D. (2000, April). A taxonomy of musical genres. In Content-Based Multimedia Information Access-Volume 2 (pp. 1238-1245). LE CENTRE DE HAUTES ETUDES INTERNATIONALES D'INFORMATIQUE DOCUMENTAIRE.

Saki, F., & Kehtarnavaz, N. (2014, May). Background noise classification using random forest tree classifier for cochlear implant applications. In Acoustics, Speech and Signal Processing (ICASSP), 2014 IEEE International Conference on (pp. 3591-3595). IEEE.

Scaringella, N., Zoia, G., & Mlynek, D. (2006). Automatic genre classification of music content: a survey. IEEE Signal Processing Magazine, 23(2), 133-141.

Schapire, R., Freund, Y., Bartlett, P., & Lee, W. (1998). Boosting the Margin: A New Explanation for the Effectiveness of Voting Methods. The Annals of Statistics, 26(5), 1651-1686. Retrieved from http://www.jstor.org/stable/120016

Single-linkage clustering. (n.d.) In *Wikipedia.* Retrieved from https://en.wikipedia.org/wiki/Single-linkage_clustering

UPGMA. (n.d.) In *Wikipedia*. Retrieved from https://en.wikipedia.org/wiki/UPGMA