Evaluating Genre Classification Methods

Christian Taruc and Michael Motyka

December 17, 2017

Abstract

Our goal is to be able to build a functional predictive model using several supervised learning methods to try to classify musical genres. Later, we experiment with combining unsupervised dimensionality reduction and resampling methods on a lyrical corpus.

Introduction

Musical genres and its classifcation tend to be one of the more interesting problems in musical information retrieval. Its applications are numerous (recommendation systems, track recognition, and categorization). Despite its ubiquitious nature in music, there are no strict standards for many musical genres in the world. Much of genre is based on historical and cultural practices. For this reason, genres are tough to classify.

Throughout our initial research, we found that the standard method for music information retrieval applications, which includes genre classification, was Convolutional Neural Nets on features collected with MFC. [1]

MFC (mel-frequency cepstrum) consists of MFCCs (mel-spectrograms ceptral coefficients), a sort of adjusted time-frequency representation on each individual music file. These are based on the mel scale, which approximates the human perception's sense of pitch.

This seems to be the standard method for these types of problems throughout the literature we encountered. However, with little knowledge on MFC methods nor a deep understanding of neural nets, we opted for a simpler approach.

Our Approach

We found that the Spotify API had several out of the box features, useful in analysis:

> Instrumentalness: This value represents the amount of vocals in the song. The closer it is to 1.0, the more instrumental the song is.
>
> Acousticness: This value describes how acoustic a song is. A score of 1.0 means the song is most likely to be an acoustic one.
>
> Liveness: This value describes the probability that the song was recorded with a live audience. According to the official documentation "a value above 0.8 provides strong likelihood that the track is live".
>
> Speechiness: "Speechiness detects the presence of spoken words in a track". If the speechiness of a song is above 0.66, it is probably made of spoken words, a score between 0.33 and 0.66 is a song that may contain both music and words, and a score below 0.33 means the song does not have any speech.
>
> Energy: "(energy) represents a perceptual measure of intensity and activity. Typically, energetic tracks feel fast, loud, and noisy".
>
> Danceability: "Danceability describes how suitable a track is for dancing based on a combination of musical elements including tempo, rhythm stability, beat strength, and overall regularity. A value of 0.0 is least danceable and 1.0 is most danceable".

Valence: "A measure from 0.0 to 1.0 describing the musical positiveness conveyed by a track. Tracks with high valence sound more positive (e.g. happy, cheerful, euphoric), while tracks with low valence sound more negative (e.g. sad, depressed, angry)".

[2]

There is no public documentation on the exact methodology to determine these features. However we believe that the features were (at least slightly) computed using the MFCC to Neural Net method mentioned above.[3]

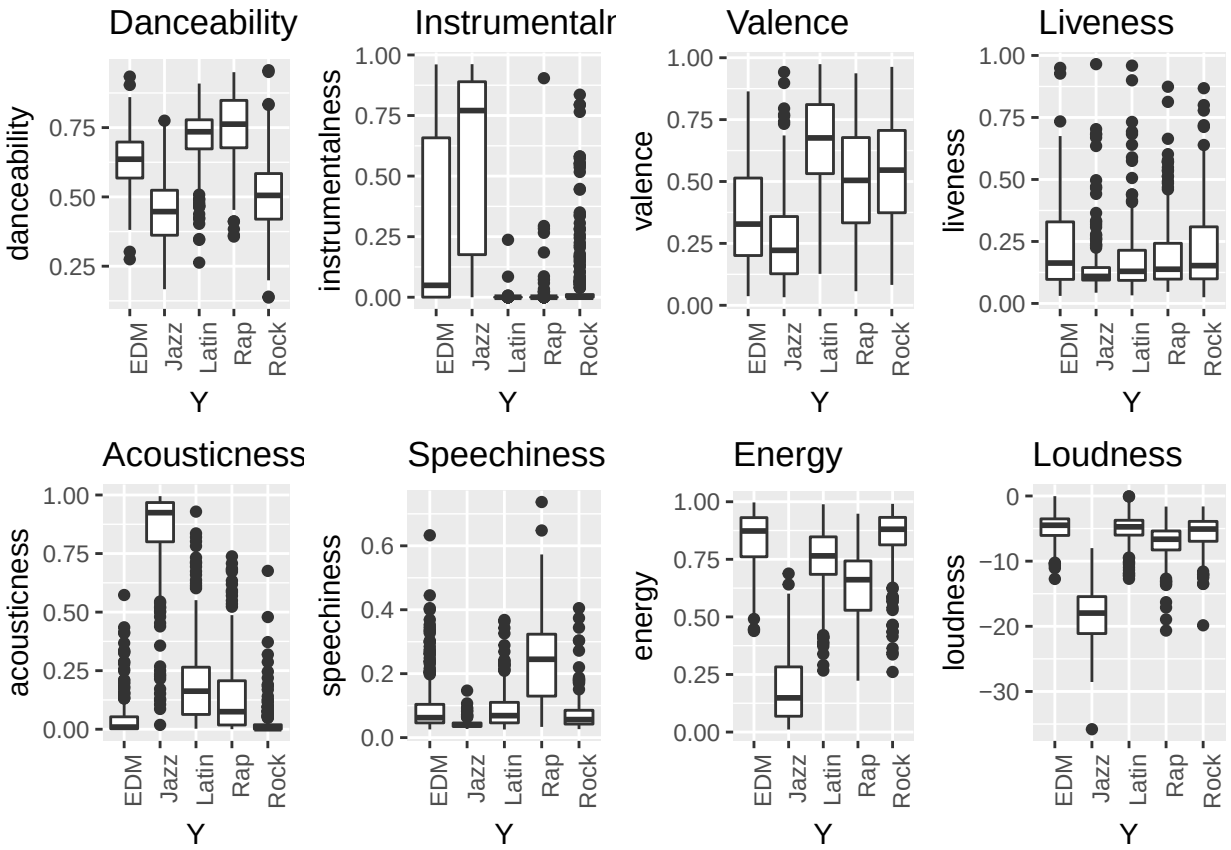
We believe 5 different genres should be able to encompass popular western music: rap, rock, latin, EDM (electronic dance music), and jazz. These genres are chosen because they are distinct enough to not encourage overlap (i.e. cross genre music).

Our method starts with data collection. We gather hundreds of songs from each genre with a given genre classification. Then, using a correlation matrix we remove highly correlated features. We classify with the resulting features using multinomial logistic regression and random forests.

Data Retrieval

Since the Spotify API itself does not give 'genre' as a feature, we created 'master' playlists and gathered songs for each genre. Each playlist ranged from 220 - 270 tracks. This allowed us to get an accurate representation of genre usable in our dataset. For consistency reasons we limited the songs to be from playlists generated by Spotify. This seemed to be the largest source of a variety of songs that were reasonably representative of their genre. In total, we have a combined playlist of 1227 observations.

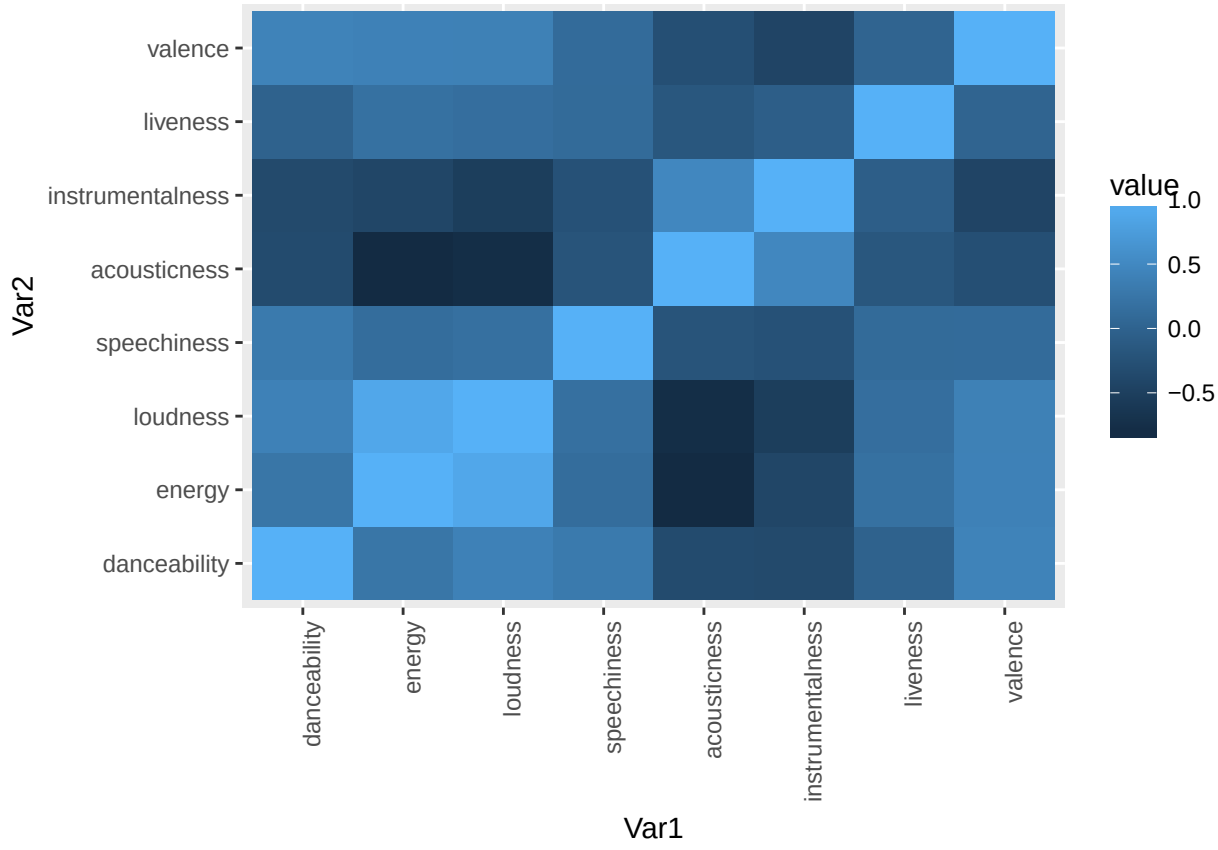
Using spotifyr, an R package, we queried Spotify for audio features and other track metadata for each song from these playlists, such as tempo and key, joining all of the information in a tibble. (The following boxplots describe each individiual feature according to genre.)



There are a couple of things to note. Jazz has a very low energy, loudness, valence, danceability, and speechiness scores. Rock is very similar to EDM according to these features, with similar energy, danceability, liveness, speechiness and acousticness scores. Rap and latin also are similar in danceability, energy acousticness, and liveness. We see there are large variations in the instrumentalness of rock, with a very low mean but very high amounts of variance.

Feature Selection

Still wary of these out of the box features, we attempt to find correlations. Using the caret library, we find that the energy feature is highly correlated to the loudness feature (.88) and thus remove loudness from the feature set. In general, the feature set is decorrelated, which will be useful for classification.



Methods and Results

We split the data randomly on a 70/30 split: 70% for training, 30% for testing. We classify using two techniques: multinomial logistic regression and random forests.

Multinomial logistic regression

Since our dependent variable has 5 different factors, we extend logistic regression to multinomial regression. Multinomial logistic regression is a multi-equation model, similar to multiple linear regression. When using multinomial regression we focus on pairwise relations with a "base" class. For 5 genres we need 4 sets of logit equations and coefficients, since we are looking at 4 comparisons between classes. In essence, we take several logistic regressions and take the probability of the highest outcome.

We use the nnet package to run the model. From the output we see that the rock genre was the base class. For jazz, the biggest difference is in the energy of the music. For latin music 'instrumentalness' was the most important differentiator . For rap, 'speechiness' is the biggest difference between it and rock. For EDM, 'danceability' was a big factor.

```
##        truth
## pred    EDM Jazz Latin Rap Rock
##   EDM    53    1    10   2   10
##   Jazz    1   68     1   1    1
##   Latin   6    2    67   8    6
##   Rap     4    0    10  42    1
##   Rock   12    0     6   2   55
```

```
## [1] 0.2276423
```

We find the test error to be ~22%. Jazz was nearly perfect in classification. However, latin and rap seem to misclassify often. This is probably because there are a lot of cross genre songs in the latin category (i.e. rap verses on latin music). We also see a lot of misclassification with EDM.

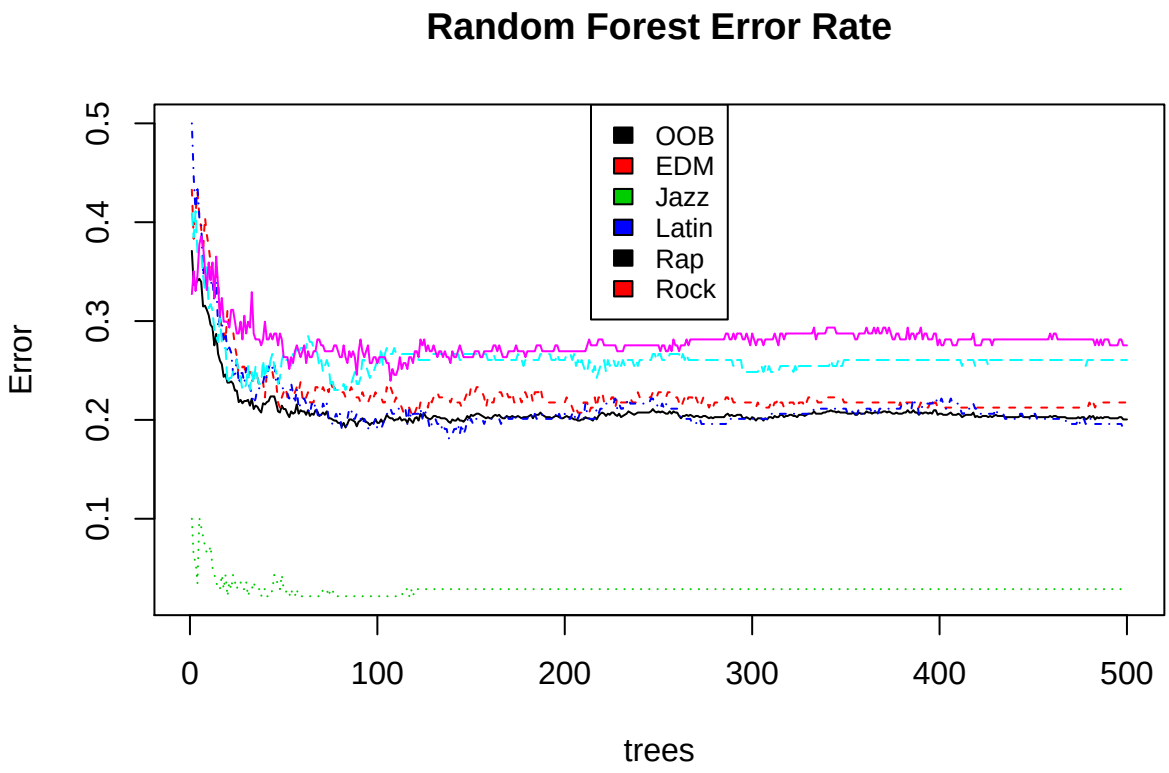
Random Forest

The class notes on random forest gave a succinct outline of the algorithm which served as our reference. Random forest creates several bagged decision trees. Also to note is that at every individual split, the trees choose from a random subset of the features.

We use the randomForest R library. We chose our 'ntry' (the number of features in the subset) to be sqrt(7) and ask for 500 trees.

Random forest provided a 4% increase in accuracy compared the multinomial logistic regression model!

We see from the plot that the model most frequently misidentifies rock, and similar to multinomial logistic jazz is classified nearly perfect. We see from the confusion matrix that EDM is the least accurate, followed by latin.




```
##         truth
## pred     EDM Jazz Latin Rap Rock
##   EDM     53    1     4   2   11
##   Jazz     1   69     0   1    0
##   Latin    6    0    76   3    2
##   Rap      4    0    10  47    2
##   Rock    12    1     4   2   58
```

```
## [1] 0.1788618
```

Lyric Based Classification

Not satisfied with using Spotify's out of the box features, we shift our methodology to modeling with text-based data. We believe that lyric-based genre classification may be useful because of the inherent differences in word choices between the genres. For example, latin music is the only genre with spanish words. Rap is the genre most commonly associated with slang, however rap can also be very lexically dense. EDM is very repetive, etc.

This second method required scraping lyrical data from the Genius API. We then take this data into a lyrical corpus using the tm package in R. This creates a neat document term matrix, which we further transform using PCA. We believed PCA would be the most efficient way to approximate variance in the data. Finally, we use different classification methods to create a predictive model.

Our Approach

Data Retrieval

Lyrical analysis has always been hard because of copyright issues. Most research surrounding song lyrics use a bag of words model. Unable to find an appropriate dataset that included the music in our original dataset we attempted to scrape lyrics from genius.com. Using the geniusr package, we are able to gather the lyrics from the same set of songs used in the original dataset. We tidy this data set, (remove punctuation, set all characters to lowercase, attempt to remove any combined words, and remove any stopwords) for tokenization.

However, it became clear that the Genius website did not have all the songs in our original playlist dataset. In the end, only 30% of the songs from the original playlist were scraped. Many of the rock and rap songs were scraped, but there were a clear lack in the 'latin','edm', and 'jazz' genre. It was clear why there were no jazz songs: they did not have lyrics. This was also partly true for 'edm' and 'latin' music, although there were some music with lyrics. For our case, we dropped the 'jazz' genre completely. We keep the EDM and Latin music.

In search of ways to recover our dataset, we came across a resampling technique called SMOTE (synthetic minority over-sampling technique).

SMOTE

SMOTE is very useful when there are rare cases in a dataset. It takes advantage of over-sampling the minority class, while under-sampling the majority class thus leading to a better overall dataset. The technique creates synthetic examples of a given class with similar features to other examples in the class.

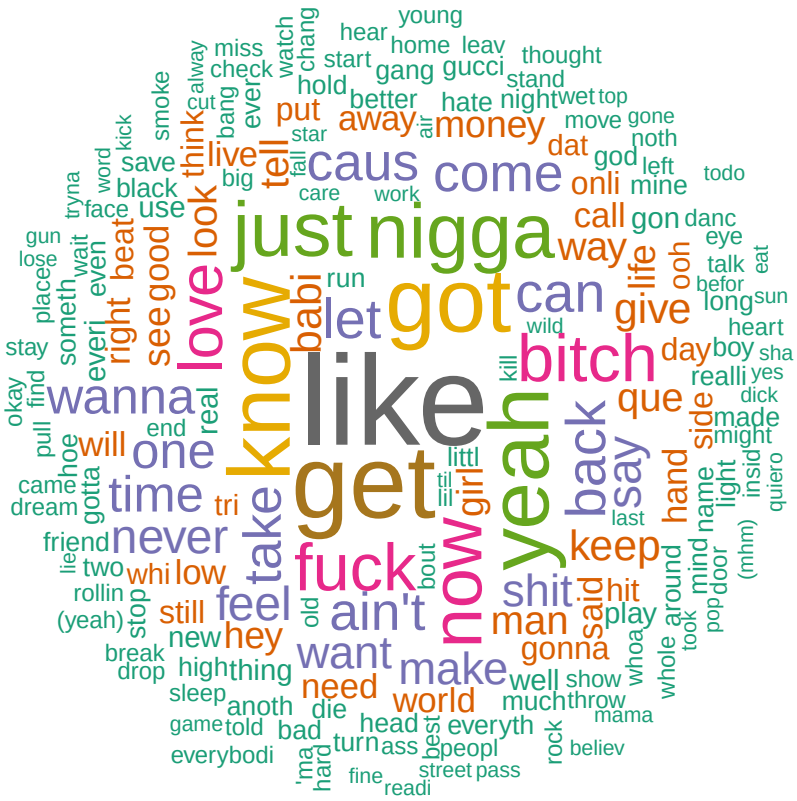
These synthetic examples are generated using k nearest neighbors. Given a random data point, SMOTE takes the vector between any k neighbors (the norm), and multiplies it by some magnitude between 0 and 1. This is added to the given data point to create a 'new' data point. [4]

We use the SMOTE technique to repopulate the dataset with 'latin' and 'edm' genres. However, it is not possible to use SMOTE on character datatypes. Further transformations are necessary before we use SMOTE. The PCA transformation followed by SMOTE resampling has shown to be successful in increasing classification rate. [5]

Tokenization/Document Term Matrix

We use the tm package to create a corpus file using the current dataset. Tokenization is a process in which we take the lyrical content of each song and break it up into units called 'tokens'. These tokens are usually words in the context of corpora. We further stem these tokens, a process which reduces the word to its root form. We do this to further reduce the feature space without reducing explainable variance.

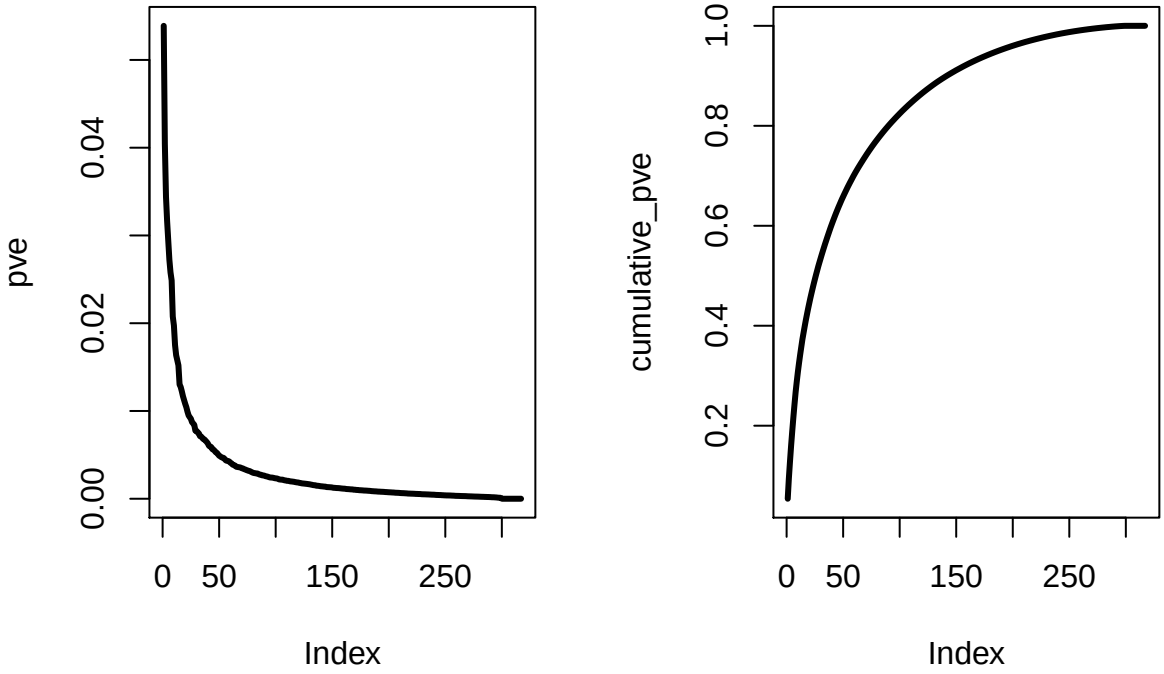
This leads to the creation of a Document Term Matrix, which gives us the frequency of each token (column) for each song (row). However, with nearly ~8000 features, applying SMOTE or attempting to classify would be very computationally intensive. We then create a wordcloud to see the most important words with regards to variation (excuse the language).



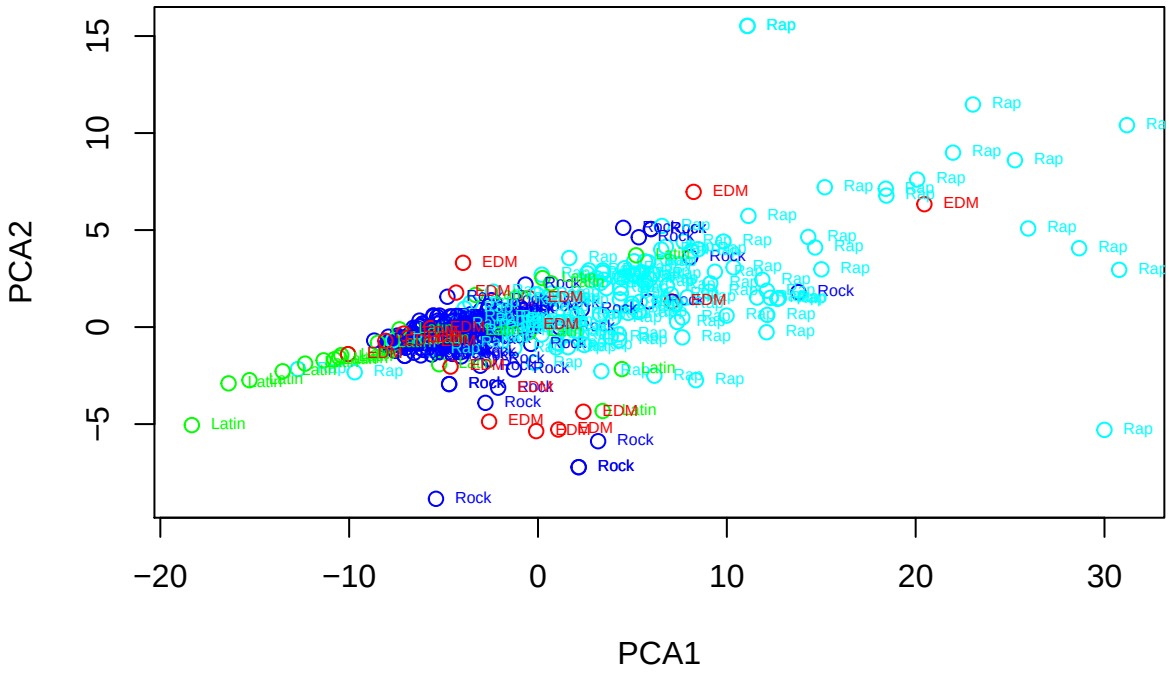
Principle Components Analysis

We use PCA to reduce this problem to a more suitable lower dimension. PCA does this by creating 'principle component' features that aims to show the most variation between observations in a lower dimensional space.

PCA finds a new orthogonal coordinate system, given by eigenvectors of its covariance matrix, and diagonalizes the covariance matrix. Then the direction of the first principle component is the first eigenvector of the covariance matrix. We can see from the following plots the proportion of variance explained by each component. Around ~50 components there is a dropoff in proportion of variance explained.



Plotting the first two principle components and labeling according to genre, we see clear distinctions in the classes (especially between rap and latin genres). There are outliers (Get Low - Dillon Francis, Get Money - Junior MAFIA, Low Life - Future) that we leave in the dataset but remove for the sake of the plot.



SMOTE(continued)

The PCA technique reduced the feature space to 318 variables. We split the dataset again on a 70/30 split and then do SMOTE on the training set. We see in the original training set, there is a large imbalance between genres. (The following table is of the probability of each label).

```
##
##        EDM       Jazz      Latin        Rap       Rock
## 0.04977376 0.00000000 0.05429864 0.39366516 0.50226244
```

Since SMOTE is usually for two-class factors, we have to do multiple passes across the dataset. We create a 'majority' set containing the union of the 'rap' and 'rock' observations and then create two individual 'minority' sets, containing the 'edm' or 'latin' observations. We pass the SMOTE function twice, using each minority class against the majority class. We then add the synthetic data points to the training set.

The resulting dataset is closer to the expected distribution.

```
##
##       EDM      Jazz     Latin       Rap      Rock
## 0.1833333 0.0000000 0.2666667 0.2416667 0.3083333
```

Methods and Results

kNN

We then use the kNN algorithm to classify. First we use 5-fold cross validation to determine the best choice of k. Then we model and create a confusion matrix.

We compare kNN between the non-SMOTE dataset and the SMOTE dataset, using confusion matrices. The first matrix is without SMOTE, the second is with SMOTE.

```
##           true
## predicted EDM Latin Rap Rock
##     EDM     0     0   1    0
##     Latin   0     0   0    0
##     Rap     2     0  10    1
##     Rock    6    10  34   32
```

```
## [1] 0.5625
```

```
##           true
## predicted EDM Latin Rap Rock
##     EDM     2     0   4    3
##     Latin   0     4   0    0
##     Rap     1     0   7    1
##     Rock    5     6  34   29
```

```
## [1] 0.5625
```

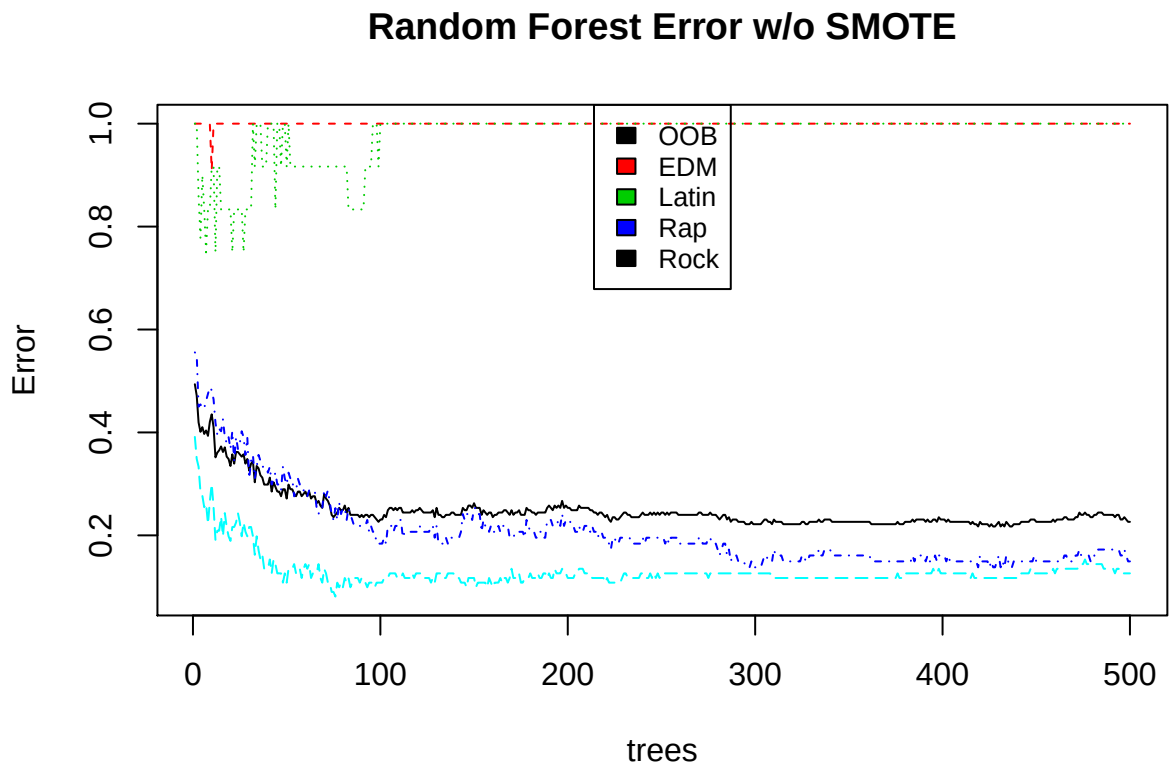
We see that SMOTE did not provide a change accuracy. It may have not been a useful addition after all. We see that it was able to increase the accuracy in the prediction for the Latin genre, however it decreased in its Rap classification. We may have over trained the classifier with SMOTE to be very sensitive to the 'EDM' class and thus has a large EDM false positive rate. For future tuning, a lower EDM proportion would be useful.

Random Forest

Next, we use the random forest method to classify. Similar to random forest used for the spotify feature dataset, we try 500 trees and plot the error rates.

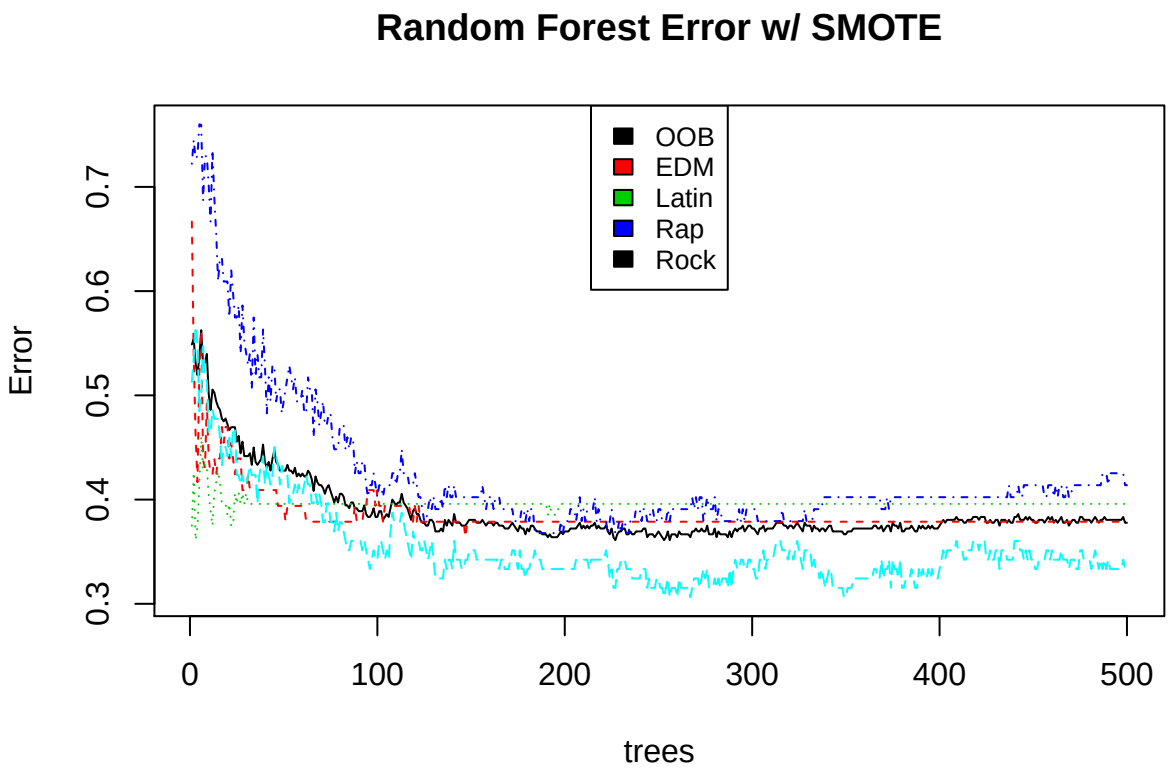
SMOTE was successful in increasing accuracy! The model had a very hard time classifying latin music, without SMOTE. However, when we included SMOTE the latin classifier increased from a 10% TPR to a 80% TPR. The downside was that the model included more false positives.

Also interesting to note was that the model did not predict for EDM at all, regardless of SMOTE. This might be because there were no features in which EDM was a good split.




```
##         truth
## pred     EDM Latin Rap Rock
##   EDM      0     0   0    0
##   Latin    0     1   0    0
##   Rap      5     5  39    6
##   Rock     3     4   6   27
```

```
## [1] 0.3020833
```




```
##         truth
## pred     EDM Latin Rap Rock
##   EDM      0     0   0    0
##   Latin    0     8   0    0
##   Rap      3     0  37    4
##   Rock     5     2   8   29
```

```
## [1] 0.2291667
```

Conclusion

Discussion

We are content with how our models performed. Considering the Spotify dataset had features extracted from a black box (which we assume to be a Neural Network/MFCC method) we expected that methodology to perform better as a predictive model. We also see that both the Multinomial Logistic Regression and kNN predictably did worse than the ensemble method of random forest. We see that, although SMOTE underperformed for the kNN classifier, it did improve the random forest model. The decreased performance in the kNN method is attributable to oversampling a specific label. This would have been best solved with some sort of cross-validation on the SMOTE levels.

We also wanted to perform logistic regression using the text data, but this was too computationally intensive and thus we opted for a more naive kNN approach.

In general, the hardest step of the process was data collection. Scraping, even with R wrappers, was a time-intensive process. We question how each of our models would classify music such as classical (we assume jazz) or metal (we assume rock). In the case of the lyric model, it would not even be able to classify between classical and jazz!

Future Work

Our project is a very simple approach on the music genre classification problem. Fully understanding the MFC to Neural Network pipeline and implementing it would be the next step in improving our model. A more diverse and complex dataset, one which includes more genre labels and more songs to each genre would also greatly decrease model variance while maintaining model bias. Our text methodology can be extended in several ways. Including much more data is a simple way to increase classifier score. We believe an SVM classifier would also be useful to complement PCA in our pipeline.

Genre is inherently ambiguous. However, it is critically important in categorising music. Genre classification by lyrics is inherently flawed by vague genre boundaries and by the fact that every genre borrows lyrical themes and styles.

Our project shows that lyrical data performs weaker in genre classification compared to other forms of data. However, it is very useful in learning the 'sentiment' of a song and in that has merit in mood classification. A pipeline that includes a combination of both lyric-based and audio-based techniques and weighs them accordingly might be useful to shore up their individual weaknesses.

References

[1] *Tom LH. Li and Antoni B. Chan* "Genre Classification and the invariance of MFCC to key and Tempo" https://pdfs.semanticscholar.org/5767/70606d61a746412c39c39a839bbb5228b9ac.pdf

[2] https://developer.spotify.com/web-api/get-audio-features/

[3] *Ben Anne* "Recommending Music on Spotify with Deep Learning" http://benanne.github.io/2014/08/05/spotify-cnns.html

[4] "SMOTE - Supersampling Rare Events in R" http://amunategui.github.io/smote/

[5] *Mehdi Naseriparsa and Mohammad Mansour Riahi Kashani* "Combination of PCA with SMOTE Resampling to Boost the Prediction Rate in Lung Cancer Dataset" https://arxiv.org/ftp/arxiv/papers/1403/1403.1949.pdf