Data Science Institute
COLUMBIA UNIVERSITY

Project Report

Daniel Gomez, Maxime Kaniewicz, Srinidhi Kunigal, Marianne Sorba

GENRE CLASSIFICATION

Projects for Data Science, Patrick Houlihan

Contents

Introduction

Being able to extract information out of music files is becoming more and more important as the music market is getting predominantly digital. Genre recognition is one of the key aspects of music analysis, as it can be used to classify very large sets of music tracks, but also serve as a basic tool for music recommendations online for example.

Based on extracted numerical features, we first performed a multiclass classification to identify the main genres. We built and trained several models whose performances we then compared. We then tried to adapt our algorithms to the case where tracks have several genre labels instead of one only. Eventually, we also added text analysis based on the track titles to improve the accuracy of our predictions.

Daniel took care of the multiclass classification, the MP3 feature extraction and the feature analysis, Maxime of the multiclass classification and the multilabel classification, Srinidhi of the title analysis and the feature description, Marianne of the description of the data and the multiclass classification.

Description of the data

1 Our Database

The data we used in this project comes from the FMA (Free Music Archive) that can be found on GitHub [1]. This giant database contains 917 GiB of audio from 106,574 tracks, 16,341 artists and 161 genres. It also provides audio features for each mp3 file that were pre-computed using LibROSA [2], a python package for music and audio analysis.

For our projects, we mainly used the database containing the features. To get a clean dataframe, we merged the dataframe containing the description of the tracks with the one containing the features. Our final dataframe contains 19034 songs. Each song has a specific id and title and has 523 features. There is a total of 163 genres in our database, with 15 main genres. Below is an example of the subgenres of "Rock", which is one of the main genres.

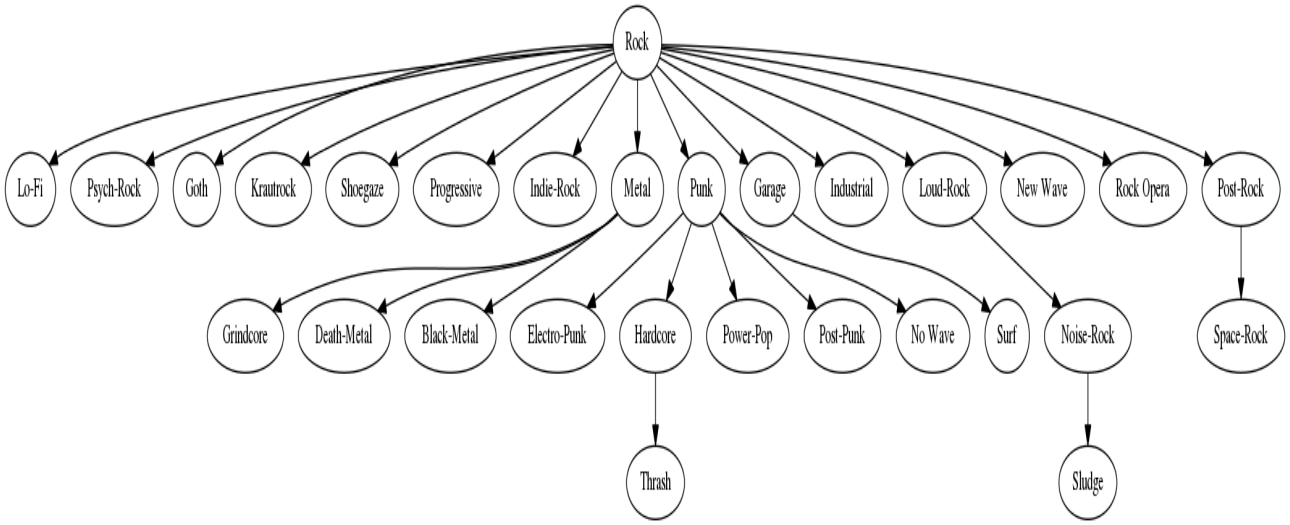

Figure 1: Subgenres of Rock

There is three variables that describe the genre of each tracks: one for the main genre, one for multiple genres, and one for the subgenres of the track. We can see in the figure below that the genres are not evenly distributed. We kept the four biggest genres for first multiclass classifier to have a better accuracy and not give importance to the underrepresented genres without loosing too much information.

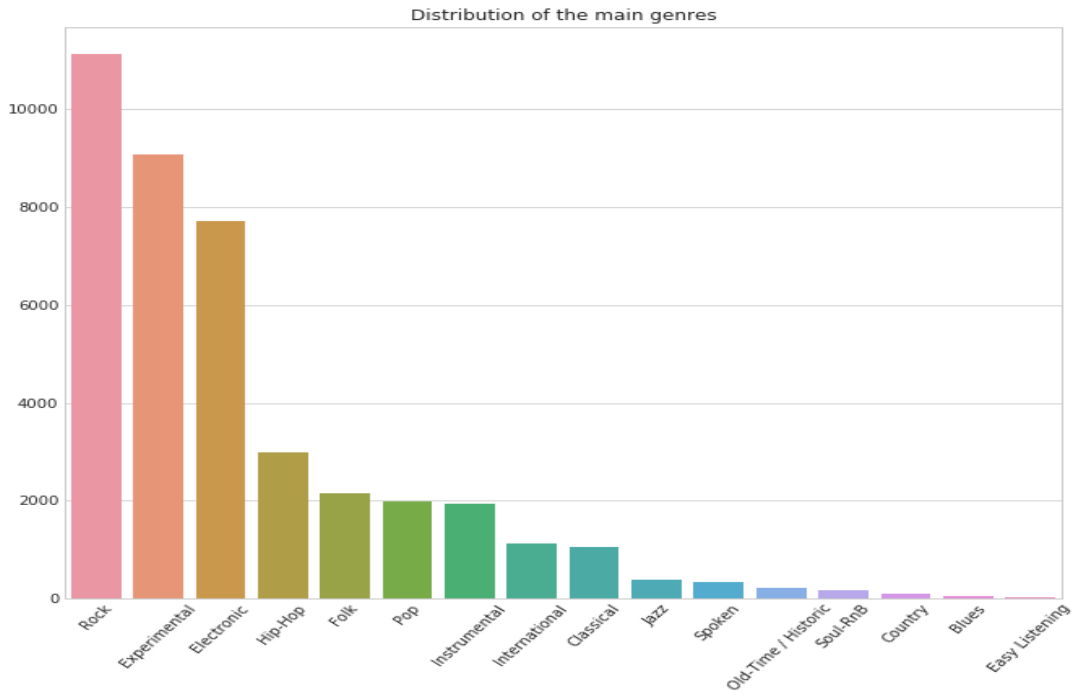

Figure 2: Distribution of the different genres

2 An overview of the features

Electronically, a sound signal is a function of (Frequency & Amplitude) vs Time. Whereas, Humans perceive music as an ensemble of various entities: Pitch, Rhythm, Loudness & Timbre and their variations. The exact mathematical representations of them are still in debate, but, pretty accurate estimates are available. The features are derived as shown in the image below by first taking the Fast Fourier Transform on the given music signal. Each of the features, in-turn have 7 derived features: Mean, Median, Min, Max, Standard Deviation (related to 2nd moment), Skew (3rd moment) and Kurtosis (4th moment).

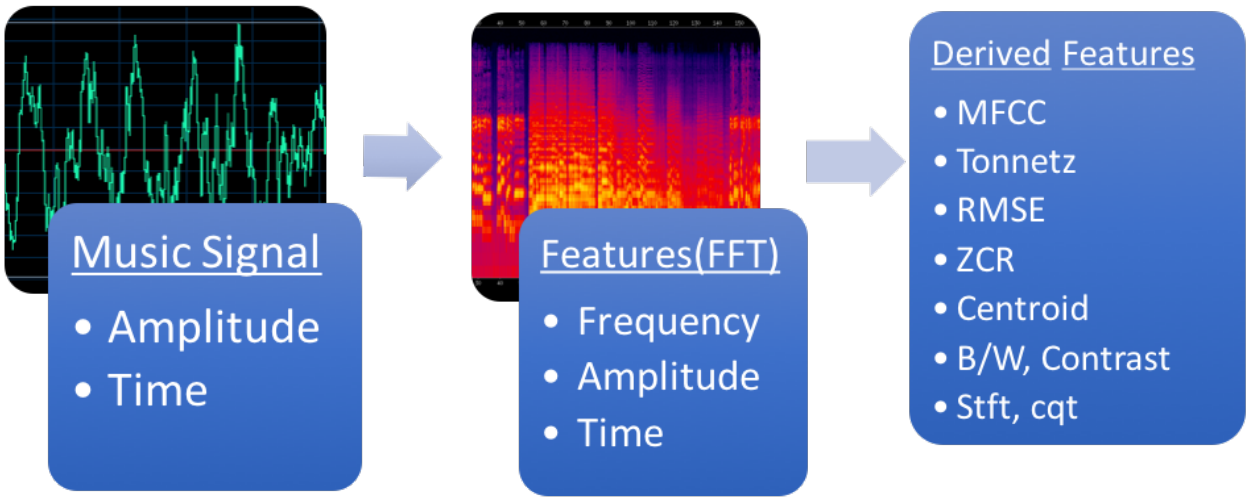

Figure 3: Features

Features related to the Mel Frequency Cepstrum(MFCC), Tonnetz and Spectral Contrast were found to be the most important. They make intuitive sense, as, Mel frequency is closely related to the human perception of pitch, tonnetz to the timber and the (octave based) spectral contrast is the difference between the peak and the low frequency in a sample. Some of the features, like those related to the 'chroma' representation- related to the intensity of each note in the octave, were not found to be important and hence were removed.

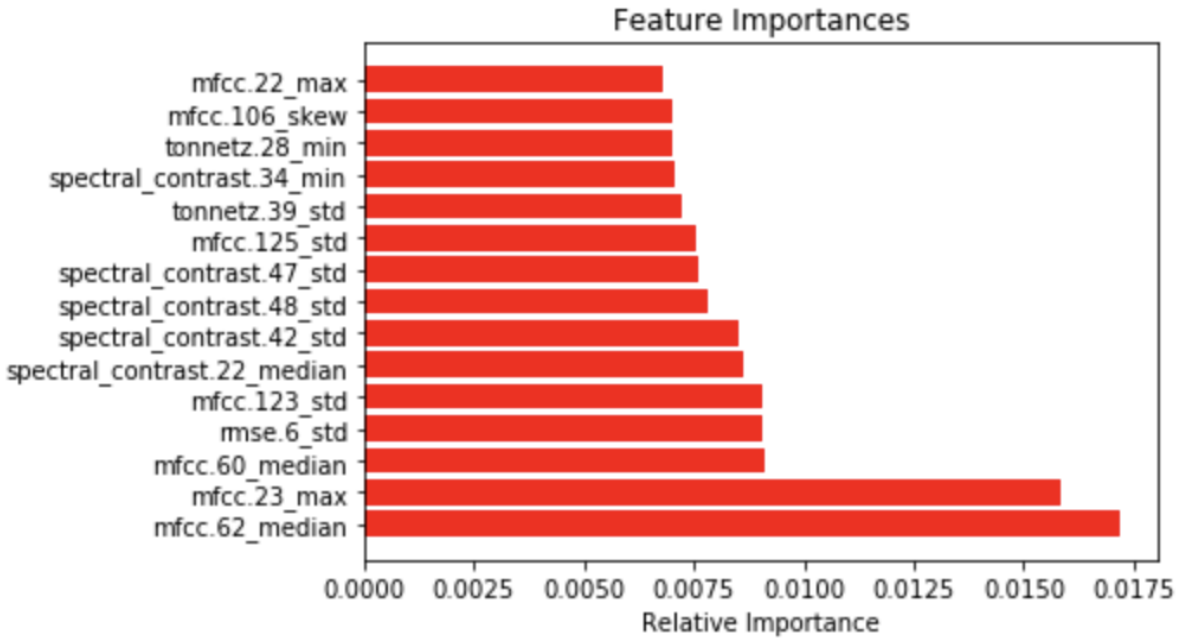

Figure 4: Feature Importance from the Random Forest Model

Multiclass classification

Our first goal was to be able to predict the 4 main genres given the features of our tracks. For this, we trained different models and tuned them using our dataset. We split our data in two: we used two third of our data to train our model, one third will be used to test our model.

1 The models

The first model that we used is a k-nearest neighbors classifier. For each song in the test data set we locate the k closest members of the training set, and predict the genre that has the biggest weight within those k closest members. Euclidean Distance measure is used to calculate how close each member of the Training Set is to the target row that is being examined. We used a gridsearch to tune the parameter k.

The second model that we used is a random forest. To tune our random forest, we first identified and chose the parameters that control the speed/accuracy trade-off. For example, intuitively, the more trees we have, the more accurate our model, but the longer it is to fit our model to our data. We then tuned the parameters that control over fitting by analyzing the performance of our model on both the training and testing set. Below is an example of the graphs that we used to tune our parameters.

Figure 5: Tuning of the maximum depth of the trees

We can see in this graph that our model is respectively under fitting and over fitting for a maximum depth that is lower or higher than 12. The ideal depth of our trees is therefore 12.

The third and last model that we used is a Gradient Boosting Classifier. We tuned its parameters using Gridsearch.

2 The results

The final accuracy scores that we got for our KNN Classifier, GBC Classifier and RF Classifier were respectively 0.715, 0.7842 and 0.730. In view of the confusion matrices below, it is clear that the KNN classifier is not as good as the other models. And while the random forest is also not quite as good as the Gradient Boosting Classifier, it is important to note that it was significantly longer to train. Depending on how costly we consider time, the random forest might therefore be a better model.

Real \ Prediction	Electronic	Experimental	Hip-Hop	Rock
Electronic	2287	496	88	228
Experimental	360	2693	48	395
Hip-Hop	276	78	735	4046
Rock	213	406	26	4046

Table 1: Confusion matrix of the Gradient Boosting Classifier on the testing set

Real \ Prediction	Electronic	Experimental	Hip-Hop	Rock
Electronic	2118	615	57	309
Experimental	454	2504	28	510
Hip-Hop	399	102	539	118
Rock	259	502	26	3906

Table 2: Confusion matrix of the Random-Forest Classifier on the testing set

Real \ Prediction	Electronic	Experimental	Hip-Hop	Rock
Electronic	1601	316	229	348
Experimental	461	1800	107	590
Hip-Hop	230	50	678	82
Rock	180	240	51	3237

Table 3: Confusion matrix of the KNN Classifier on the testing set

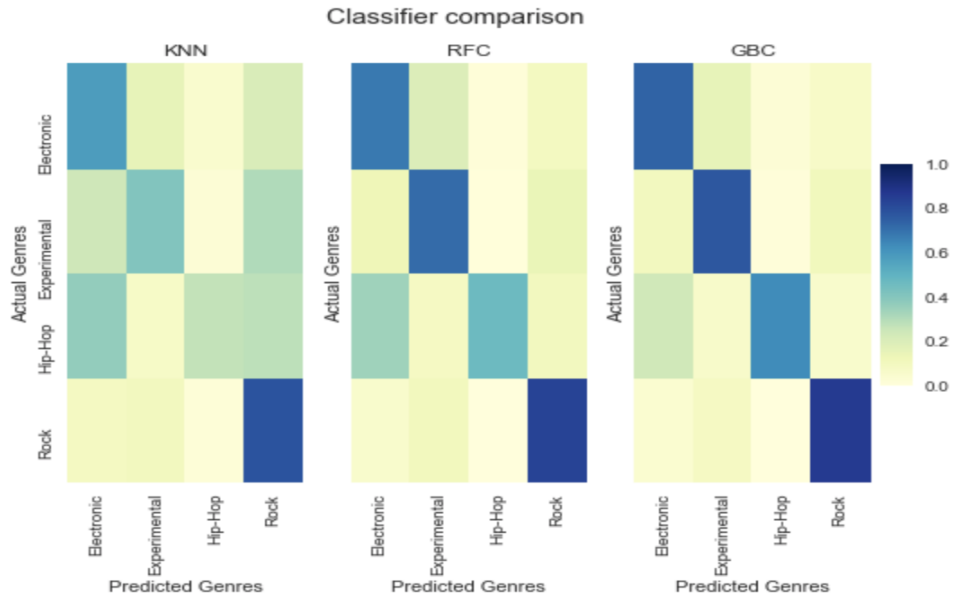

Figure 6: Color Matrices based on the confusion matrices

3 Expanding to the Full Set of Genres

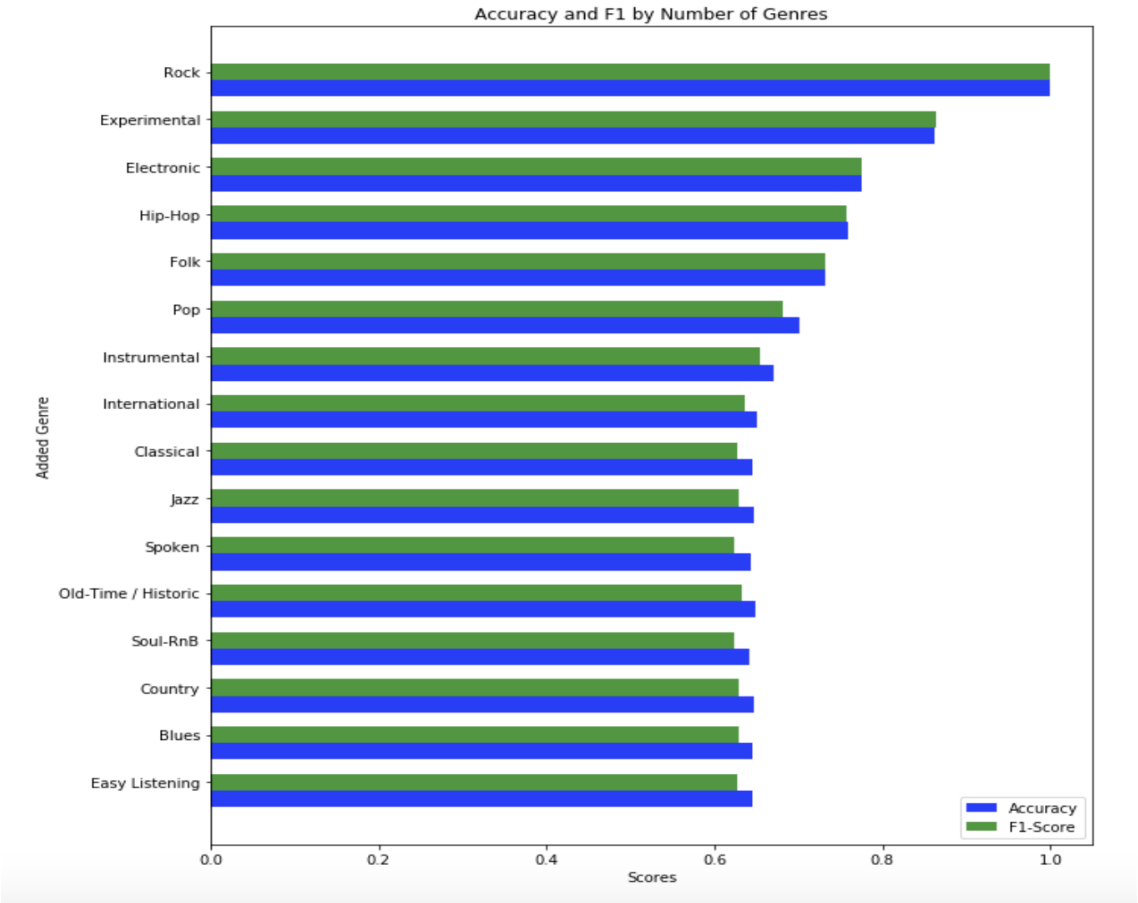

Figure 7: Accuracy of Neural Network depending on amount of genres

Running a Shallow Neural Network on four genres was found to be near competitive with our top result and much, much faster. This led us to decide to expand this result to choose the top n genres from n = 1 to n = 16 and run the NN on all of those. Obviously, the accuracy will decrease as we add more and more genres, but we found that it stayed quite good even up to 16 genres, as shown in Figure 7. The plot is set up to show the accuracy and f1 score after the genre on the y axis was added. This means it's cumulative and as you go down the graph, there are more and more genres. We made the neural network have more hidden nodes linearly to the amount of genres as this was found to work best. We made the neural network have as many hidden nodes as the amount of genres, as this was found to work best.

Multilabel classification

The data we have does not only provide the main music genre, but also lists of genres in case where several genres can apply. For instance, it is quite common that a song could be called Pop - Rock, instead of simply Rock or Pop. After managing to successfully classify the main music genre in the previous step, we want to push the analysis further and see if it is possible to allow tracks to belong to several music categories. For this purpose, we will use multilabel classification.

1 The models

We chose to keep the 9 most represented labels in our data set : ['Avant-Garde', 'Electronic', 'Experimental', 'Experimental Pop', 'Folk', 'Lo-Fi', 'Noise', 'Pop', 'Rock']. In the following models, we consider y as an array where the i-th element is equal to 1 if the track belongs to the i-th genre and 0 otherwise. For instance, a track with [1,1,0,0,0,0,0,0,0] is Electronic and Experimental. For each track, the goal is thus to predict an array with possibly several ones in it. We chose two approaches for this.

- The first approach we can use is the "One vs Rest" approach, where one classification is computed for each label, and all the remaining labels are merged together as the alternative value.
- The second approach is the chain classifier approach, where we predict one label after another just like previously, but then push the predicted label to the explanatory variables set for the next labels. This approach allows to take into account some correlations that could exist within the labels. For instance, it makes sense to think that once a song has been identified as Pop, it is less likely to also be Electronic, but more likely to also be Rock or Experimental Pop.

Intuitively, we can think that for the second approach, the order in which the labels are predicted will affect the accuracy of the prediction. Obviously, we could not try all the combinations, so we tried one where we put the labels that are most likely to be paired first, one where these labels were put last, and one random distribution of the labels. However, it seems like they all are somewhat equivalent on our data set.

2 The Results

For each of the two approaches, we computed three models : K Nearest Neighbors, Random Forest Classifier and a Neural Network Classifier. We chose to use the f1-weighted-score. f1 can be interpreted as an average between precision and recall. In our case, the score will compute an average of the f1-scores for each label, weighted by the size of each label. Here are the results of our models :

	f1-Score
KNN One vs Rest	0.46
KNN Chain	0.46
RF One vs Rest	0.31
RF Chain	0.33
NN One vs Rest	0.41
NN Chain	0.44

Table 4: F1 scores of our models

Recall/Precision / Labels	RFC Recall	NN Recall	RFC Precision	NN Precision
Avant-Garde	0.051	0.3156	0.392	0.288
Electronic	0.429	0.570	0.709	0.596
Experimental	0.341	0.488	0.540	0.484
Experimental Pop	0.041	0.320	0.414	0.280
Folk	0.148	0.445	0.624	0.405
Lo-Fi	0.210	0.432	0.835	0.381
Noise	0.132	0.324	0.612	0.331
Pop	0.064	0.326	0.608	0.302
Rock	0.209	0.514	0.725	0.507

Table 5: Precision / recall table of our models

We can see from these tables that the f1 score can be increased up to three points using a Classifier Chain, as we could expect. Thus, there is a clear advantage to use a Chain Classifier instead of a One Vs Rest Classifier, which is the default setting in scikit-learn. We can also see that from the three algorithms used, we get the best results with KNN, reaching a f1 score of 0.46. The Neural Network Classifier reaches a f1 score of 0.44, but is much faster than the KNN procedure. However, the Random Forest Classifier is far behind these two algorithms in terms of f1, as it only reaches 0.33.

However, only considering the f1 score is restrictive. A closer look at the recall and precision values of each of the label predictions indicates that all the models are not strong on the same prediction aspects. The comparison of the Random Forest and the Neural Network classifier shows that the former has a larger precision on all labels and the latter has a larger recall on all labels. This means that the Random Forest classifier doesn't often predict 1 for a label, but when it does, the prediction is more reliable. We can conclude that the selection of the model should not only depend on the f1 score performance, but rather on the priority of the algorithm user, be it precision or recall.

Text Analysis: Improving the accuracy using NLP

It is often seen that a particular genre of songs uses certain words frequently. In the hip-hop domain, for example, the words 'Yo', 'Homey', and 'Swag' are ubiquitous.

1 The Model

We built a LDA-Topic model with 5 topics on Song titles to see if the text clusters (Topics) were intuitively differentiating Genres. The LDA model, basically does a sort of unsupervised clustering and hence identifies hidden clusters the given text, if any.
We created the corpus dictionary by removing stop-words, punctuations and then lemmatizing the remaining words. Then, using the LDA module(from gensim), we generated the Word-Topic probability matrix and thus, the Document-Topic probability matrix. We used the Document-Topic probability values thus generated (5 probabilities per track, summing to 1) as features to improve the accuracy in previous classification models.

2 Topic Model: Results

	Average Probability				
Genre	Topic 1	Topic 2	Topic 3	Topic 4	Topic 5
Country	23%	26%	21%	16%	14%
Electronic	20%	20%	19%	20%	21%
Jazz	16%	23%	19%	21%	21%
Old-Time / Historic	17%	20%	23%	20%	21%

Figure 8: Features

We found that each Genre is roughly, favoring two-three particular Topics. For Ex., 'Country' songs are dense in Topics 1&2, but not so much in others. Some of the performances of the genres are represented in the feature table. As we can see, the procedure does not discriminate between Electronic tracks, but does somewhat differenciate some of the less represented genres. The range of variability for the other genres does not exceed the range presented in this tables.

The probability features, when used to predict Genres, improve accuracy by 1 percent point. Better clustering can be obtained by doing the same analysis on Song lyrics, but, they are not easily available and running NLP models on them is computationally heavy.

Extracting the mp3 features

We extracted the features in a very similar way to the way the dataset originally created the features, using much of their code and adjusting it to our process. We created a function that uses the librosa python library to compute the Cepstrum Coefficients, as well as the spectral properties and other useful features of an inputted mp3 file. From here, we used numpy to calculate seven statistical moments from each of these time series and used those as our columns for classification. We then created a function that takes in an mp3 file, computes these features, and runs the result through our full 16-genre classifying neural network. That way, we were able to predict the genres of our own favorite MP3 files.

Conclusion

We based our project on a dataset of more than 100,000 songs found in the Free Music Archive (FMA). We trained models on this data to predict the genre. We compared the performance of several models, and achieved around 80% accuracy with the Gradient Boosting Classifier when predicting the four main genres. When working on the 16 main genres, the accuracy goes down to around 65%, which is still fairly satisfactory. The multilabel classification, where we allowed the tracks to belong to several genres, proved to be more challenging, as we only reached a 46% accuracy using K Nearest Neighbors. We then tried to improve the performances of our algorithms by including a text analysis of the track titles. However, this did not significantly improve the accuracy, as the titles are not a large enough text sample for this purpose. Extending the analysis to the lyrics would have been an interesting, though costly next step. Finally, we were able to predict the genre of any of our own MP3 files using the functions and models we have built.

References

[1] Kirell Benzi, Michaël Defferrard, Pierre Vandergheynst, and Xavier Bresson. FMA: A dataset for music analysis. *CoRR*, abs/1612.01840, 2016.

[2] Brian McFeek, Colin Raffel, Dawen Liang, Daniel P.W. Ellis, Matt McVicar, Eric Battenberg, and Oriol Nietok. librosa: Audio and music signal analysis in python, 2015.

[3] George Tzanetakis and Georg Essl. Automatic musical genre classification of audio signals. In *IEEE Transactions on Speech and Audio Processing*, pages 293–302, 2001.

[4] Beth Logan. Mel frequency cepstral coefficients for music modeling. In *In International Symposium on Music Information Retrieval*, 2000.

[5] Shubham Jain. Solving multi-label classification problems (case studies included), 2017.