Music Genre Recognition

Mortiniera Thevie, Gomez Antoine, Hajri Skander, Wagner Patrik

***Abstract*—At the time of digitalization, the use of coousmamputers to study, store, listen or do anything else related to music became obvious. With the increasing amount of data at our disposal and the unprecedented computational power of today's computers, the demand of automated tools suitable for MIR and genre recognition tasks keeps increasing. In this paper, we used various Machine Learning models to classify music genres. Our best model achieved 76% for a 4-genre recognition with Logit Model, and 61.4% for a 10-genre recognition task using Extra Trees.**

I. Introduction

Music can be found in all the human societies since the prehistoric age and has always been a way for human beings to express themselves, to socially gather and also a way to symbolize cultures. Music is essential and universal. Friedrich Nietzsche said : "A life without music is simply a mistake, a tiredness, an exile". Therefore, at the time of digitalization, the use of computers to study, store, listen or do anything else related to music became obvious. Since 2017, the free music archive[1] (FMA), a freely available and easily accessible data set has been released, suitable for MIR tasks, and evaluate some baselines for genre recognition.

We used this data set to solve a classification problem to detect music genres directly by analyzing the features of the music tracks. As a first visualization step, we created a graph where nodes represent the tracks and where the latter are connected by edges in function of the similarity of their features to visualize clusters of music genres. Secondly, given acoustic feature of tracks we classified them according to their genre as Rock, Hip-hop, Electronic and Experimental, using several machine learning models.

II. Data Extraction

A. Data description and features selection

The given FMA data set was composed of :

- 106,574 tracks with metadata such as ID, title, artist, genres, tags and play counts
- 163 genre IDs with their name and parent
- common features extracted with librosa
- audio features provided by Spotify for a smaller subset

In our study, we only used the tracks, genres and features data extracted from librosa. Multiple choices and operations have been made in order to clean and present the data in the most suitable way. First of all, for the tracks, we decided to keep only the relevant information for our task. We noticed that some tracks'genre were mixtures of various genres, hence we decided to only keep the tracks who did not have various ones. Thus, the only possible genres remaining were the one considered as genres top, counting up to 16. In preparation for the subsequent prediction task and optimize our recognition algorithm, we have decided to only keep 4 different genres, since the performances of machine learning models drops when the number of classifications is above 4, according to previous works in the area. [2] As for the features, for the prediction task we decided to keep the whole over filtering them due to performance reasons, i.e, the classifier performed slightly better using all of them.

However, for the visualization part, the choice of features subset was decided using Principal Components Analysis (PCA) choosing the features with the highest variations and the ones which gave us better clustering visualizations. Hence, we decided to keep the Mel Frequency cepstrum Coefficients (MFCCs).

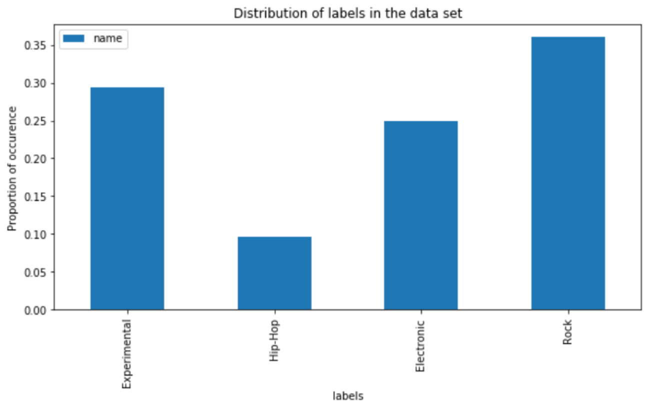

Figure 1. Genre Distribution

As we can see in the figure above, the distribution of the labels is quite unbalanced. In order to tackle this and remove the bias, we chose to balance the labels equally among the 4 genres. We split the data set into training and test data, with test set being 10% of it.

III. Visualization

to visualize clusters of music genres, we created a graph where nodes represented the tracks and where the latter were connected by edges in function of the similarity of their features. As a first trial, we modelized using a 2 dimensions approach, however it gave poor results as the visualization was noisy, we could not see the separation of genres since we used 4 and more of them (we tried also a 10-approach). Therefore, we used a 3 dimension layout for visualization purpose which allowed us not to loose too much information. We used laplacian eigenmaps[3] to project on the 3 principal eigenvectors constituings our axis.

A. *Genre Clustering*

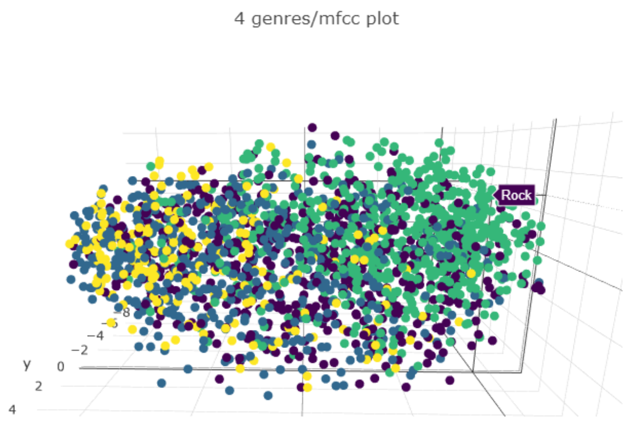

Figure 2. 4-Genre Clustering using MFCCs

We observed 4 clusters, however the separation was not that clear as mentionned before, it looked like a noisy cloud. In particular, Electronic and Hip-Hop clusters overlapped.

B. *Artist Clustering*

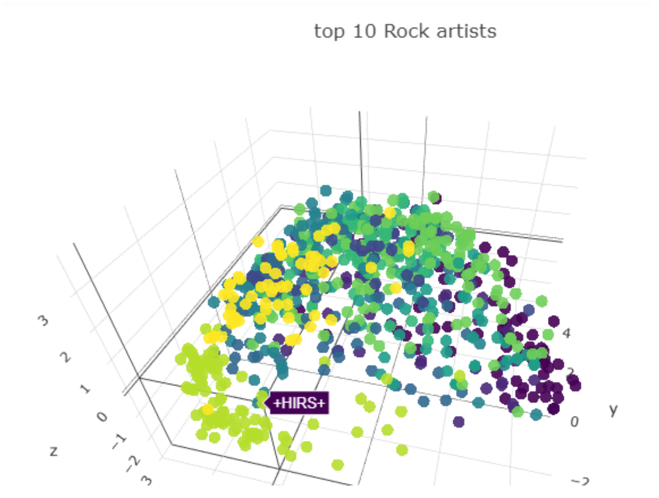

Figure 3. Top 10 rock artist clustering

Here, we used again laplacian eigenmaps but within the top 10 artists Rock-labelled tracks. We can observe, even inside the same group of music genre, clustering of music, implying that songs from one artist to another, even from the same genre, differs from their acoustic characteristics.

C. *Artist-Genre based Clustering*

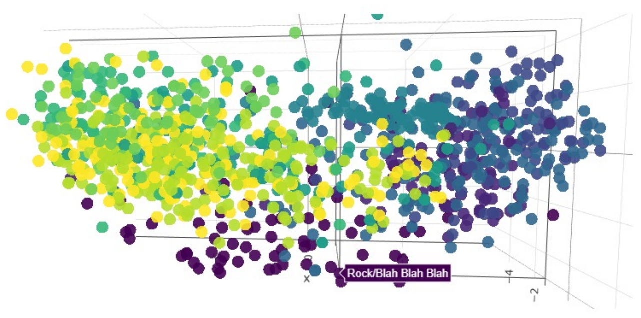

Figure 4. Top 5 rock and hip-hop artists clustering

We used once more laplacian eigenmaps within the top 5 artists in Rock and Hip-hop labelled tracks. Two distinct genres clusters appeared : Green/yellow for Hip-Hop and Blue/Purple for ROck. Although, we observed differences within the same genre groups above, when plotted against each other, the differences within the same genres group are weaker than the one with other genres. Furthermore, we noticed that some ”in-between” songs appears. They revealed that genre classification is not strict, some artists can make heterogeneous genres songs and that some music belongs to different genres. In the second part, with more advanced machine learning techniques, we wondered if our results reflected the analysis done in this part.

IV. Learning Models

We experiment several machines learning techniques for our classification task. Hyperparameters were tuned running a grid search along with 5-fold cross validation to avoid over-fitting in a reduced sample data set which consisted of 10% of the actual transformed data. The following models were used.

A. *Logistic regression*

Namely for Logit Model, the inverse of regularization strength C was chosen among $[0.001, 0.001, 0.01, 0.1, 1, 10, 100, 1000]$ and we tried between $L1$ (Lasso regularization) and $L2$ (ridge regression) penalties. In our settings, $L1$ penalty and $C = 1$ were chosen, to reach an accuracy of **83%** on the training set and **63%** on the test set.

B. *K-nearest neighbors*

Regarding K-Nearest Neighbors model, which was strongly suggested during class for graph-based inference models[4], the number of neighbors to considered was set to bet odd and choosen between 1 and 11. GridSearch found that the best settings was the one consisting of 7 numbers to consider, using euclidean distance as metric with uniform weighting, reaching an accuracy of **68%** on the training set and **61%** on the test set.

C. *OneVsRestClassifier with Linear SVC base*

Another widely used popular strategy for multiclass classification is OneVsRestClassifier. In this setting, we used Support Vector Machines (SVMs) with a linear Kernel as the base estimator, and the inverse of regularization strength C was chosen among $[0.001, 0.001, 0.01, 0.1, 1, 10, 100, 1000]$ as in Logit Model. We reached an accuracy of **79%** on the training set and **62%** on the test set, with $C = 0.1$.

D. *SVC with infinite feature MAP (rbf)*

Keeping up with SVMs, we tried here with the radial basis function kernel for an infinite map feature. The regularization strength C was searched among the same set as before, but we tried now with the parameter $\gamma \in [10e-3, 10e-4]$. The best parameters set found was $C = 100, \gamma = 0.001$ which an accuracy of **83%** on the training set and **68%** on the test set.

E. *Multi-layer perceptron*

Finally, we tried using Neural networks which are known to give better results than classic methods. Here, we used Multi-layer perceptron. The best setting was with 2 hidden layers with 200 hidden nodes on each, hyperbolic tangent as activation functions, penalty parameter $\alpha = 0.1$ and ”adam” a stochastic gradient-based optimizer; which gave an accuracy of **81%** on the training set and **60%** on the test set.

V. 10 Genres Attempt

As an attempt to have more realistic results with a wider range of genres we initially tested the Extra Trees Classifier model, used in [1] to recognize tracks between the top 10 genres.

A. *MFCCs*

As a first intuition we trained our model using the MFCC features only, MFCCs being used in music information retrieval applications such as genre classification and audio similarity measures.
The main parameters of the Extra Trees Classifier are the number of estimators(depth of the trees) and the maximum number of features. First of all, we tried to find the right range of estimators to be used. To this end we chose a low number of maximum features, here 5, and got the following results displayed in Fig. 5.

We can see that the precision gain after n_estimators = 100 is very small, we will hence not go further than a depth of 150 and the 140 MFCC features for the rest of our tests. To ensure the validity of our models we had to do cross-validation and choose the number of folds to use. We therefore tested different sizes and chose a number of 10 folds,

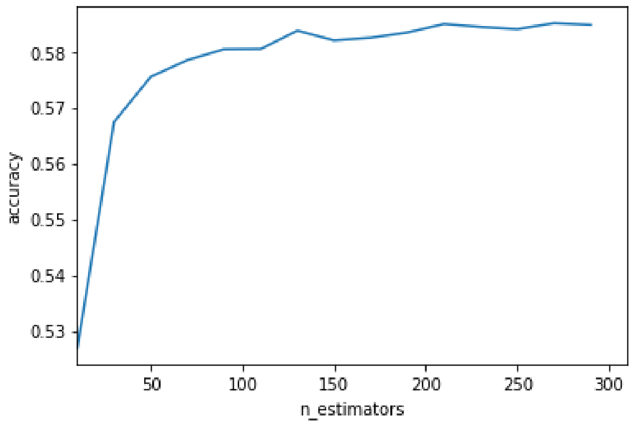

Figure 5. Extra Tree Classification with 5 max_features

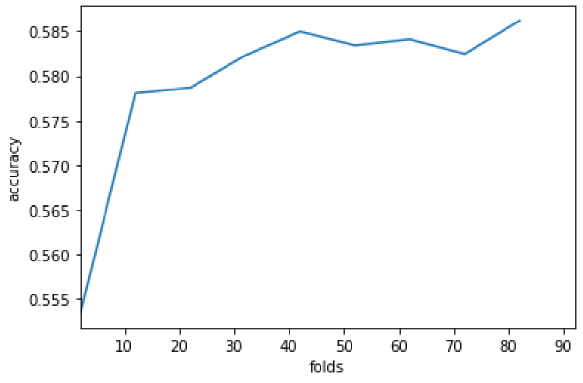

Figure 6. Extra Tree Classification with 140 max_features and 100 estimators

after which the precision almost doesn’t variate anymore. This can be seen in Fig. 6.

By testing the model with the entire MFCC features the best accuracy achieve was slightly above 0.585 with 10 folds and n_estimators = 150. This is a satisfying result compared to other types of parameters such as Chromas, that we tested later on, which only achieved a precision score of 0.43 at best using similar parameters.

B. *All features*

In order to compare the performance of the model using the MFCCs to the best achievable performance of our model with this dataset, we tried to feed the whole feature set as an input. This reduces considerably the speed of execution as there are a total of 518 features but gives a better accuracy overall.

As we can observe on Fig.7 the maximum accuracy attained by the model is 0.614 with 130 estimators. This is relatively higher than the previous results, but it isn’t

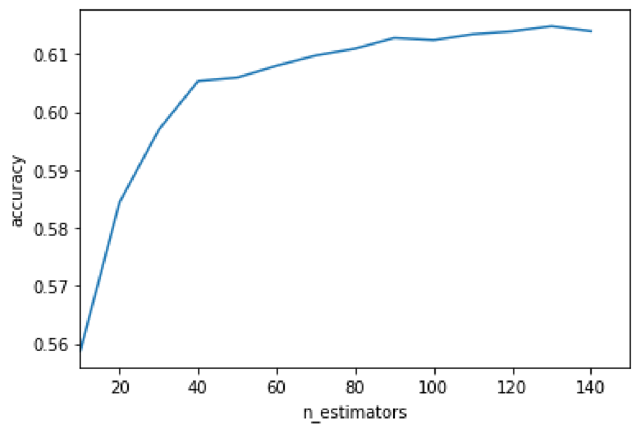

Figure 7. Extra Tree Classification with 518 max_features

such a significant bump when considering the number of additional features used.
It would hence be interesting to try to refine the model in order to find the optimal set of features to select, allowing to have a trade-off between accuracy and time efficiency. This could benefit some type of real-world application using genre recognition with the need of a fast response to such kind of queries.

C. Principal Component Analysis

As an attempt to keep only the most accurate features for genre classification and still have a decent precision score, we tried to use Principal Component Analysis.
First we used the method explained in [5] to automatically choose the number of dimensions to keep. It gave us the following results show in Fig. 8.

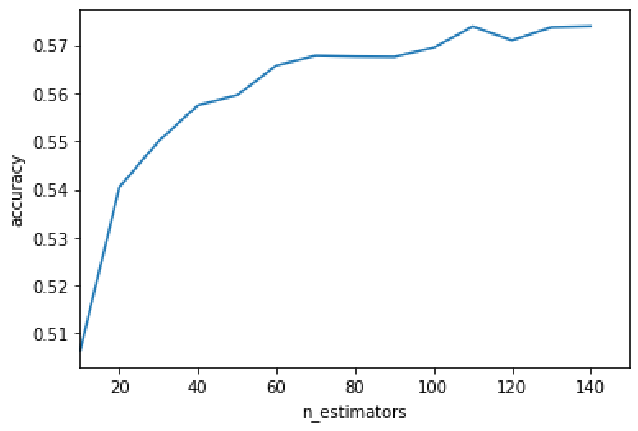

Figure 8. Extra Tree Classification with automatic PCA features

These results are similar to the one of the MFCCs but slightly lower for a notably higher number of features(about 500). We can therefore say that the method from [5] is not suitable for our task and that it would be better to use the MFCCs instead.
We then tested to set arbitrarily the dimension number to different values and check the results. The results for 100 PCA components were actually satisfying as shown in Fig. 9 as it is lower than the number of MFCC features and achieves a slightly better accuracy of 0.588 with 10 folds and n_estimators = 150.

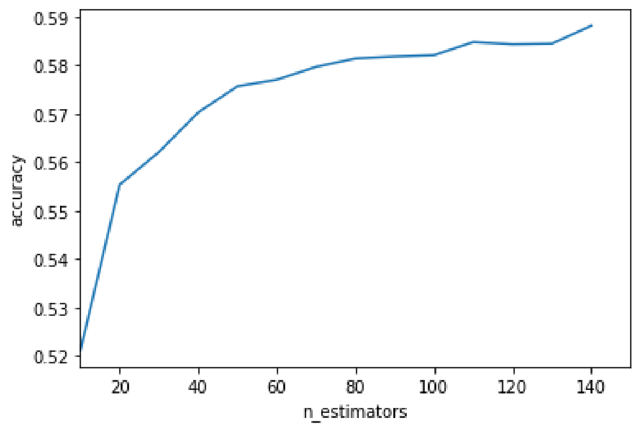

Figure 9. Extra Tree Classification with 100 PCA features

VI. Results

When we tried to solve the problem with 4 genres, during parameters tuning, the performance on the test set was quite poor when observing the gap with the training set for all models except K-Nearest Neighbor which surprisingly seemed to be quite stable in comparison to others. Most models seemed to over-fit too much, maybe due to the small number of training parameters on the validation set.

After hyperparameters tuning phase, we took the best estimators for each model to predict on ”unseen” data. The results are presented in the table below.

Model	Hyperparameters	Acc Train	Acc Test
Logistic regression	C=1, penalty=L1	77%	76%
K-nearest neighbors	metric=euclidean, neighbors=7, weights=uniform	77%	66%
OneVsRest with Linear SVC base	C = 0.1	79%	76%
SVC with infinite feature MAP (rbf)	C=100, γ=0.001	68%	68%
Multi-layer perceptron	activation=tanh, alpha=0.1, layers=2x200, learning rate=constant, solver=adam	77%	75%

Results obtained on training and test set

Against all expectations, K-nearest Neighbor which we assumed to be quite stable with small variance on validation data appeared to over-fit when trained on full data set. However, simple models such as Logistic Regression and OneVsRest Classifier with SVM using a Linear kernel seems to fit very well the underlying function when looking at the performance on the test set and their variance. Finally, as expected, neural networks which are known to work well the more data they are given performed well on the given task.

When comparing to a naive random model which would guess correctly the genre of a music 1 out 4 times, our best models guessed correctly the genre 3 times out of 4. This is quite good, however not effective enough to be used in a widely real-world distributed application.

Concerning the Extra Trees Classification model, it is interesting to note that we can considerably reduce the number of feature while keeping a precision close to what is achievable at best by the model. It would therefore be very important to weight the features and to select only the most relevant ones to reduce the time of computation.

VII. Conclusion

The results obtained revealed how genre recognition is known to be a difficult task as music is very varying regarding to acoustic measures.

As mentioned at the beginning, music genres can be heterogeneous and subjective, hence one might classify a music differently than another. Based on this reasoning, one could investigate a music recommendation system to take advantage of the duality of musical tastes.

Furthermore, our model still need to be updated with new data and some improvement need to be done.

Given more time at our disposal, we would like to investigate Convolutional Neural Network (CNN) or Long Short Term Memory Layers (LSTMs) which seem to be promising.

References

[1] P. V. X. B. Michael Defferrard, Kirell Benzi, “Fma: A dataset for music analysis,” 2017. [Online]. Available: https://arxiv.org/abs/1612.01840

[2] A. B. C. Tom LH. Li and A. H. Chun, “Automatic musical pattern feature extraction using convolutional neural network,” 2010. [Online]. Available: http://www.iaeng.org/publication/IMECS2010/IMECS2010_pp546-550.pdf

[3] P. Vandergheynst, “Laplacian eigenmaps,” 2018. [Online]. Available: EPFL

[4] D. Thanou, “Learning graphs from data: A general overview,” 2018. [Online]. Available: SwissDataScienceCenter

[5] T. P. Minka, “Automatic choice of dimensionality for pca,” 2000. [Online]. Available: http://vismod.media.mit.edu/tech-reports/TR-514.pdf