Université catholique de Louvain

Algorithms in Data Science

2020 - 2021

Music genre classification

Authors:
Archibald Duerinck (8217-17-00)
Arnaud Gueulette (1051-16-01)
Joachim Ndeze (9589-20-00)

Contents

1 Introduction

Spotify, Apple, Deezer, Amazon Prime Music and so on are platforms where classification music algorithms are used every day. Maybe one time in your life previously you wondered yourselves: « how are they able to recommend me good musics, custom personal playlists and so on? ». Based on your listened playlists, but also the songs that you listen, such algorithms are capable of analyzing these songs and then from these analyzes, recommending new ones similar to the ones listened previously.

This report will discuss music genre classification, but before going into the details about algorithms and fancy things, one question sill remains without answer: « how can we classify music? How are we able to do such things? ».

2 How to tackle such problem ?

At its basis, a song is just a special kind of audio signal which can be analyzed, but also decomposed. Indeed, using features like **length**, **tempo**, **harmony_mean**, **harmony_var** and so on, a song can be classified, but how to extract these features capable of classifying a song?

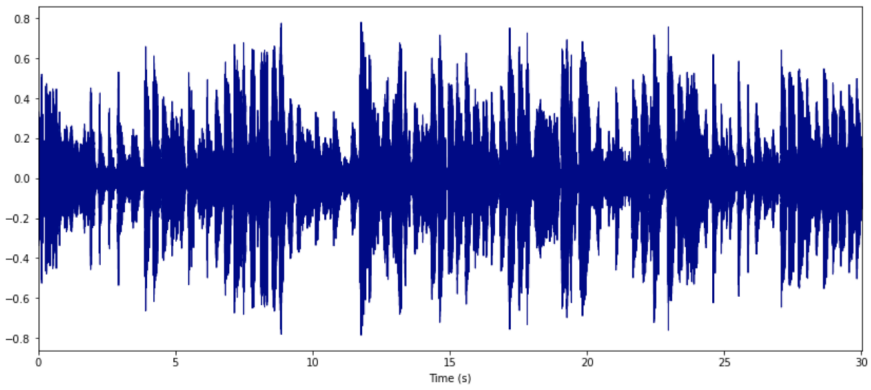

Figure 1: Waveform plot of one of our sample. We can observe some patterns into it. The y-axis unit is the amplitude.

Thanks to some python libraries like **librosa and pysoundfile**, it is easier now than before to decompose an audio signal into key elements. Beginning with its **waveform plot**, we can go deeper by analyzing its **spectrum**,

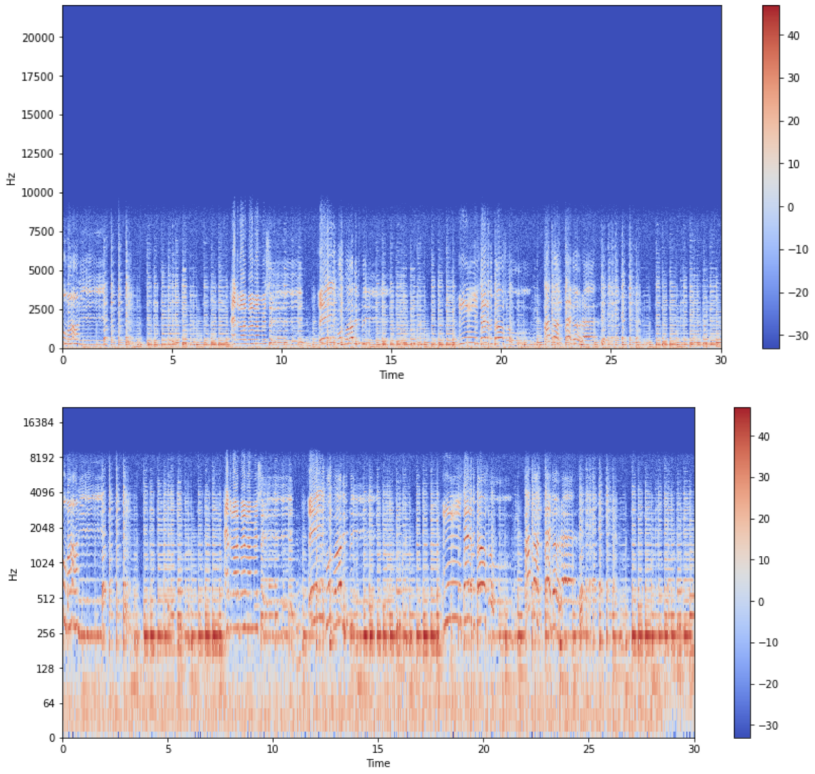

Figure 2: Spectrograms of our sample. These spectrograms show the loudness (dB) of the signal over the time and this, at different levels of frequency. Hz scale (top) and logarithm HZ scale (below).

its **spectral centroid** and **spectral rollof**,

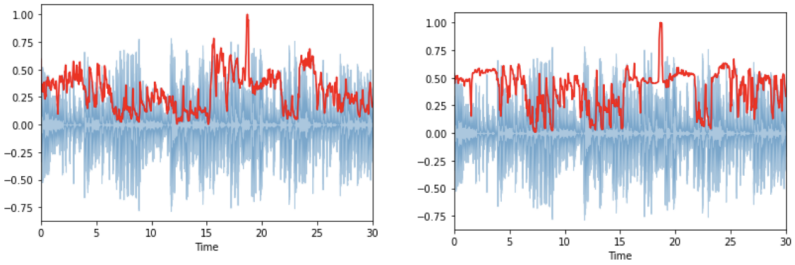

Figure 3: Spectral centroid (left) and spectrall rollof (right) of our sample. The y-unit axis is the amplitude.

its **chroma feature plot over the time**,

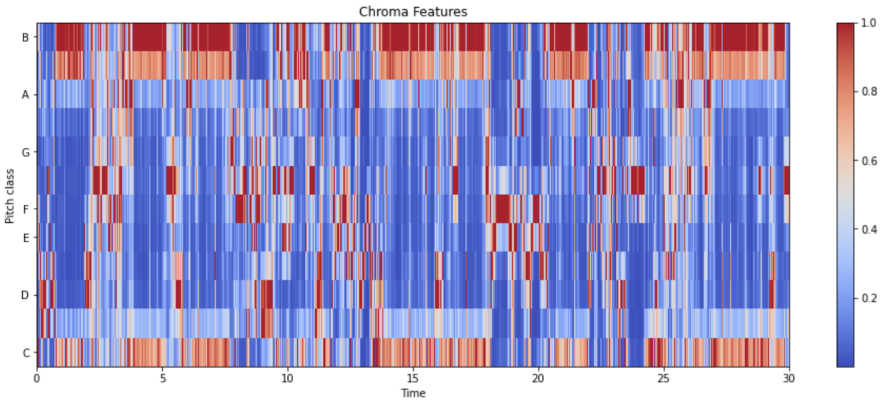

Figure 4: Chroma feature plot of our sample.

and finally, its **zero crossing rate plot**.

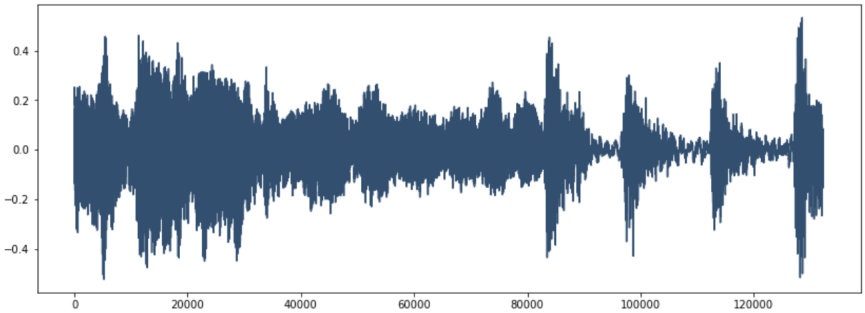

Figure 5: Zero crossing rate plot of our sample (30 sec). This plot displays the number of times that the sign of the audio signal changes over the time.

Using all these key elements, we can extract features representing a signal in another way. As one signal is different from another, these elements will differ allowing us to classify these audios according to some categories predefined. To extract features from these key elements, an embedding procedure is used, but this will not be discussed in this report since it is not a theoretical report, but a practical one. For more information, do not hesitate to look the Jupyter Notebook, but also the sources provided with it.

Doing this for all available signals, the results will be a dataset with only numerical features representing the signals at disposal.

Now that we know how to extract features from a signal, it this time to present the dataset used in this report.

3 Dataset presentation and exploratory analysis

Now that we understand better the feature extraction procedure, but also how to obtain our audio signals into another representation, we can now discuss the dataset used in this report. Based on 1000 different audio sounds of 30 seconds coming from different genres: blues, jazz, hip-hop, classical, rock, pop, ... but also the elements provided before, a dataset composed of 1000 rows and 59 columns was obtained. All the variables excepted the target one are numerical features. Moreover, as said before in the introduction, since it is a classification problem these rows are categorized into 10 categories according to the genre of the song (labelled data).

3.1 Dealing with missing values

No missing values were present into the dataset meaning that no specific procedure was done for dealing with this kind of problem.

3.2 Train, validation and test split

Since the target variable distribution was well balanced (for every genre, 100 audio sounds were provided), it was decided to use a Repeated Stratified k-Fold procedure in order to obtain robust validation values for our models. The value of K was fixed to 5 and the number of repetitions to 500 for some models and 100 for others due to computation constraints.

4 Explanatory analysis

It can be interesting before going into details about preprocessing and model building steps to analyze a little bit our data in order to catch some non-trivial relationships into them.

4.1 Numerical features

In fact since the features were just the products of different embedding methods with some adjustments from one to another, the features were well correlated.

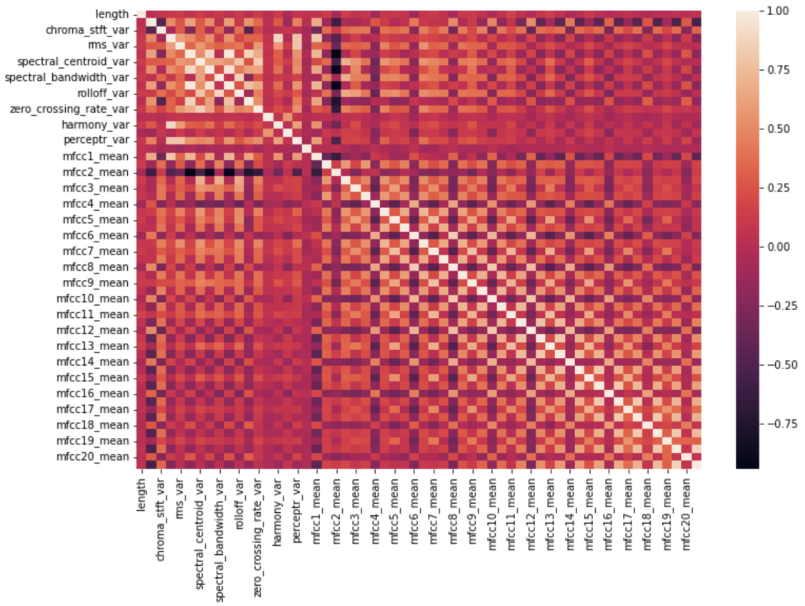

Figure 6: Correlation matrix of our features.

Indeed, a lot of pairs had a correlation value equal to 0.7 or higher that could be very problematic for our later models. **In order to deal with this problem, a principal component analysis was done onto our features with the goal to reduce the redundancy present within the data.**

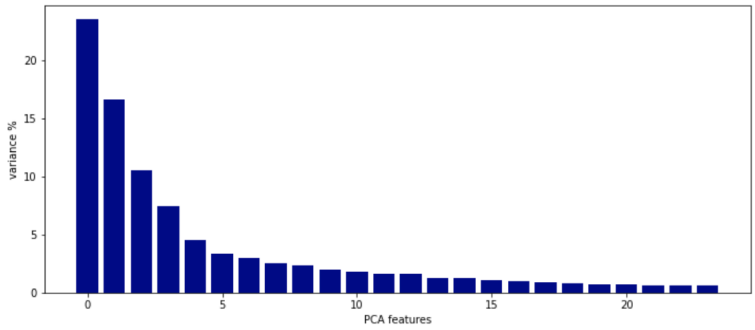

Figure 7: PCA feature importance.

Doing a principal component analysis onto our dataset, the 24 first principal components were able to explain 90% of the variance contained in the dataset, which is quite awesome[1]! It was then decided to keep those 24 components, but also the original dataset for later sections in order to compare the results obtained by the two approaches.

[1]Better graphs and plots are given in the Jupyter Notebook provided with this report.

We also noted that our data did not seem to be linearly separable with only the first three components since using a 3D plots[2], they were not enough for discriminate well the groups. Indeed, all the observations, except few ones, seem to be focused on the center of the PCA space. The algorithms later used will respond to the question whether a good discrimination is possible using the 24 components or not.

Finally, a mutual information analysis was done onto our original dataset in order to have a clear overview of the relationship between the features and the target variable. This element helped us to know which features discriminate the best the groups before doing any machine learning model.

```
chroma_stft_mean          0.534765
perceptr_var              0.510651
spectral_bandwidth_mean   0.457132
rolloff_mean              0.451191
spectral_centroid_var     0.402803
spectral_centroid_mean    0.398729
rms_var                   0.387327
mfcc1_mean                0.383646
zero_crossing_rate_var    0.365393
rms_mean                  0.349219
```

Figure 8: The ten most important features for discriminate the groups according to a mutual information analysis done.

5 Preprocessing

Some algorithms are very sensitive to the mean and variance of the data. Especially the ones that use distance in one way or another in their computation. Except for some cases, there is usually no gain to keep the data in its original form. We used standard scaling as preprocessing step. This transformation consist in subtracting the mean and keeping a unit standard deviation or variance.

In addition to this the labels were encoded with a label encoder that maps the labels to integers. These encoded vectors will be used as target for the model training.

6 Model selection

In order to tune the models and find the best hyper-parameters combination for each model, multiple runs were needed. To achieve this, most of the models have been cross-validated with a stratified train/validation set and then evaluated on an unseen test set. Sometimes, cross-validations with five runs were computationally too expensive (XGBoost and LSTM) leading to adjustments in the strategy (3 validation sets and one validation set). However, for all models the test set was the same and could be used to compare the models between them. Training and testing sets were stratified, but inside XGBoost the three validation sets were not meaning that some results have to be interpreted carefully.

[2]The interactive plots can be found in the Jupyter Notebook.

7 Model building

In this section, we will have a look at how we decided to build our models. As we said before, through our exploratory data analysis, we noticed that some variables were highly correlated and then decided to run a PCA in order to manage this problem for linear models. In this section we will therefore build our models on the orignal dataset and on the PCA dataset.

7.1 K Nearest Neighbors

Thanks to a grid search we found that the optimal combination of hyper parameters for the KNN model on the original dataset was to use 4 neighbors, to use the distance function to weights the importance of the neighbors and to set p equals to 1. For the PCA dataset we used the same method for tuning the parameter as we used for the original dataset. We found out that the optimal setting of parameters given by the grid search was to set the number of neighbors to 7, to use distance function to weight the importance of the neighbors and to set p equals to 2.

7.2 Multinomial Logisitic Regression

On the original dataset the optimal set of parameters found was to set an L1 penalty, to fit the intercept and to set C equals to 0.8. The C parameter corresponds to the inverse of regularization strength, so if C is small the regularization will be stronger. For the PCA dataset the optimal setting of parameters is the same as for the original dataset which is to set a l1 penalty, to fit the intercept and set C equals to 0.8.

7.3 Support Vector Machine

The Support Vector Machine (SVM) classifier is an algorithm used to classify data that are not linearly separable. Many times data are in a finite dimensional space, where it often happened that in this original space data are not linearly separable. SVM classifier will mapped the data into an higher dimensional space where the observation can be linearly separable. To do this mapping, the SVM classifier uses a kernel function. Based on the grid search we have done, the optimal set of parameters of the SVM for the original set is to used a Radial Basis Function (rbf) kernel, to set the regularization parameter C to 5 and to set the gamma parameter, which corresponds to kernel coefficient, to 0.01.

7.4 Extreme Gradient Boost

XGBoost is a gradient boosting method that has been recently created (2014). It has been a contest winner method in recent competitions. This is due to the fact that XGBoost has been designed to maximize accuracy performance as well as maximizing hardware constraints. It is a powerful algorithm and computationally efficient (lots of strategies have been included such as multi-threading the processing of blocks of data). XGBoost as other gradient boosting method creates a model, evaluates it and then increase the weight of the individuals that have been not well classified. In our case, we needed to fit around 1000 estimators. At each iteration (model built) the fitting is done on a (different) fraction of the data. This strategy prevents overfitting. In our case it uses 80% of the samples and 80% of the features. XGBoost has also the ability to impute automatically values that are missing. It provides, as other "tree" methods do a decision tree that is understandable, it explains on what basis a classification has been made. Today this property is often required in ML application

and is a real struggle to provide. As a result of this decision criterion awareness we can also rank features by importance (from root (most influent) decision to leaf (least impactfull of the most impactfull features)). As the model decides by itself what features are important, feature selection is not proven to be efficient for many cases.

7.5 Long Short-Term Memory Recurrent Neural Network

LSTM networks are a very important in everyday ml applications as speech recognition, computional linguistics, translation models as well as many others. This is due to some advantages that go along with this model. This class of neural networks have neurons with an internal state. It allows the model to remember parts of what it has already seen and acts like a memory. It can thus handle sequential data (video recognition, language processing...) with multiple lengths. But it has also good results for non-sequential data. In our project we selected the following network architecure: 2 LSTM layers and some dropout to prevent overfitting by shutting down randomly 10-20% of the weights.

8 Model comparison

In order to compare the different models, alike training and testing datasets were used. The only difference was the validation set because computationally expensive models were tested with less cross validation.

Time In the model comparison table, it appears that the best model according to the time it took to fit would be the K-Nearest Neighbors. It is interesting to see how it outperforms the linear model both in time to fit and predict but struggles in testing performance. The linear model has the second least accuracy and should only be employed as a baseline model since almost none of the data is linear. The model is thus not able to fit (maximum training accuracy is 87%) correctly the data. The Support-Vector Machine (with Gaussian kernel) has the best compromise between accuracy and both time to fit and time to predict. This makes it an interesting candidate if speed of recognition is a serious criterion.

Performance Often test performance is the main criterion to choose a model. For this classification problem, XGBoost has been the most performant model by far. It reaches a test accuracy of 81%.

Final model XGBoost is the solution we would have provided to a regular client (that needs performance). Besides of having the best accurracy, it also takes less time to fit than our best recurrent neural network. This shows that ml algorithms go way beyond deep learning. The ease of use, ease of tuning the hyperparameters were consistent advantages for this technique. But if it had to be a real client, we would for sure have investigated the dataset containing 3 seconds sampled musics. This sequential data may have fitted very well for a LSTM and is one lead to improve our best model.

criterion	KNN	KNN + PCA	Logistic	Logistic + PCA	SVM	XGBoost	LSTM
fit time (s)	0.002	0.0021	1.78	1.78	0.33	4.89	25.08
prediction time (s)	0.012	0.012	0.0016	0.0015	0.0085	0.1	0.52
mean validation score	0.71	0.68	0.73	0.68	0.76	0.82	0.81
mean standard deviation	0.031	0.032	0.031	0.031	0.032	0.009	NA
train score	1	1	0.87	0.77	0.96	1	1
test score	0.68	0.63	0.72	0.68	0.75	0.8	0.7

Table 1: Model comparison

9 Conclusion

With our different analysis we have seen that music classification is nowadays a lot more easier than ever. We have first seen that in order to classify music genres we had to work with a specific type of data that are signals. Since music has different harmony, tempo, rhythm and so on. If we decomposed these factor we can translate into signals.

Furthermore, thanks to our exploratory data analysis we noticed that the data were not linearly separable. This can explain the fact that linear model that we implemented (KNN and logistic multinomial) are the models with the lowest accuracy score. Even when we tried to apply PCA for these models we saw that the results were even worst than without PCA.

Finally, based on our model comparison, we can conclude that the best model is the XGBoost. But it is also important to highlight that this model is one of the model that takes the most time to fit and to predict. So if we want to improve the time to fit and predict the SVM model can be an alternative solution if we allowed our model to have at least 75% of accuracy.