Deep Learning - Cas d'étude

Thibaut JUILLARD† and Moqim GHIZLAN†

*Corresponding author(s). E-mail(s): ThibautJuillard21@gmail.com;
mooqeemgh@gmail.com;
†These authors contributed equally to this work.

Abstract

This study explores the development of a classification tool based on artificial neural networks to predict the genre of music tracks using their features. Utilizing an excerpt of the Spotify dataset, we implemented descriptive statistics to gain insight into the data, followed by benchmarking with standard classification algorithms such as SVM and decision trees. Subsequently, we designed and optimized a neural network architecture, experimenting with various hyperparameters and heuristics to enhance performance. Additionally, we incorporated semantic features derived from track titles to augment the classification process.

Keywords: Deep Learning, PyTorch, Machine Learning, Classification

1 Introduction

Music genre classification is a critical task in music information retrieval, widely used in recommendation systems, content organization, and user personalization on streaming platforms. This classification challenge involves predicting the genre of a music track based on its features, which can include both numerical descriptors of the audio signal and textual metadata. While traditional machine learning methods such as support vector machines (SVMs) and decision trees have been effective, the rise of artificial neural networks (ANNs) offers new possibilities for handling complex data structures and improving predictive accuracy.

For this study, we selected a subset of the Spotify dataset, which contains detailed numerical features describing audio characteristics such as tempo, energy, and danceability, as well as textual metadata like track titles. This dataset was chosen because it offers a realistic and diverse representation of musical genres while being rich in information for both numerical and textual analysis. Additionally, its preexisting use

in similar studies provides a reference point for evaluating the performance of our approach.

In the first part, we will provide a detailed analysis of the dataset 2 and present the results achieved using standard machine learning models. Next, we will describe our deep learning architecture 3, including the variants explored and the training and evaluation protocol employed. Finally, we will present the results obtained 4 and conclude 5 with the key insights gained from this study.

2 Dataset analysis

The dataset contains 21,428 rows and 20 variables. Among these, 6 variables are categorical, while 14 are quantitative. The dataset is balanced across 5 labels, as shown in the distribution 1.

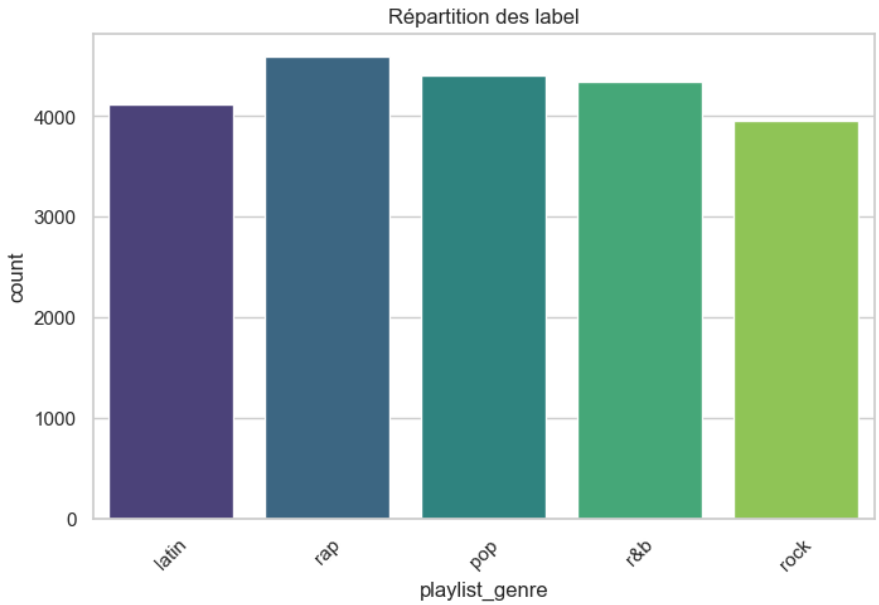

Fig. 1 This is the label distribution in the dataset.

The quantitative variables' distribution 2.

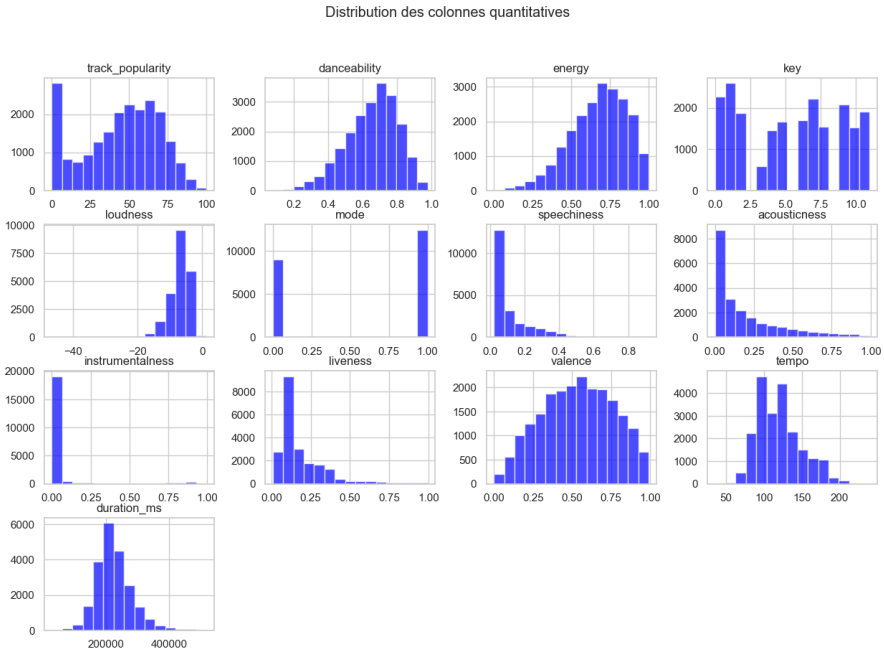

Fig. 2 This is the distribution of quantitative data.

To establish a baseline for comparison, we performed an initial classification using various models. For each model, a preprocessing step with 'StandardScaler' was applied to standardize the data. We have chosen as model :

- SVC
- Random Forest
- Logistic Regression
- KNeighbors Classifier

The results obtained are summarized in Machine Learning Results 1, indicating that the best-performing model is the **Random Forest classifier**.

Table 1 Machine Learning Results

Metrics	SVC	RF	LR	KNN
Accuracy	0.535	**0.562**	0.488	0.459
F1 score macro	0.532	**0.559**	0.485	0.462
Recall	0.536	**0.563**	0.490	0.460
Precision	0.531	**0.558**	0.483	0.472

3 Deep Learning architecture

For the first model, we selected a standard Multi-Layer Perceptron (MLP). The architecture is illustrated in Figure 3.

```
MLP_BASE(
  (fc1): Linear(in_features=13, out_features=32, bias=True)
  (relu): ReLU()
  (fc2): Linear(in_features=32, out_features=5, bias=True)
  (softmax): Softmax(dim=1)
)
```

Fig. 3 Basic model MLP (Quantitative value only).

The second model introduces various modifications to improve performance, as shown in Figure 4.

```
MLP_MODIF(
  (fc1): Linear(in_features=13, out_features=32, bias=True)
  (dropout1): Dropout(p=0.2, inplace=False)
  (act1): GELU(approximate='none')
  (fc2): Linear(in_features=32, out_features=10, bias=True)
  (norm): BatchNorm1d(10, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
  (act2): GELU(approximate='none')
  (fc3): Linear(in_features=10, out_features=32, bias=True)
  (dropout2): Dropout(p=0.2, inplace=False)
  (act3): GELU(approximate='none')
  (fc4): Linear(in_features=32, out_features=5, bias=True)
)
```

Fig. 4 Modified model (Quantitative value only).

The third model explicitly incorporates categorical variables into the architecture, as depicted in Figure 5.

```
MLP_MODIF_TEXT(
  (lstm): LSTM(397, 64, batch_first=True)
  (fc1): Linear(in_features=64, out_features=32, bias=True)
  (act1): GELU(approximate='none')
  (fc2): Linear(in_features=32, out_features=10, bias=True)
  (norm): BatchNorm1d(10, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
  (act2): GELU(approximate='none')
  (fc3): Linear(in_features=10, out_features=32, bias=True)
  (act3): GELU(approximate='none')
  (fc4): Linear(in_features=32, out_features=5, bias=True)
)
```

Fig. 5 Modified model (Quantitative and categorical value).

For training the models, we use the Cross Entropy loss function, defined as:

$$\text{CrossEntropy}(y, \hat{y}) = -\sum_{i=1}^{N} y_i \log(\hat{y}_i), \tag{1}$$

where y is the true label and $\hat{y}$ is the predicted probability distribution.

We employ the Adam optimizer with a learning rate of 0.001. During the training loop, we use mini-batches of size 64 and apply techniques such as gradient clipping to prevent exploding gradients.

For model evaluation, we rely on multiple metrics, including the confusion matrix, accuracy, precision, recall, and the F2-score. These metrics provide a comprehensive assessment of the models' performance across various aspects of classification.

4 Results

For the first model, the results are suboptimal, as shown in the performance curves in Figure 6. The model fails to achieve satisfactory accuracy, indicating that further adjustments are needed.

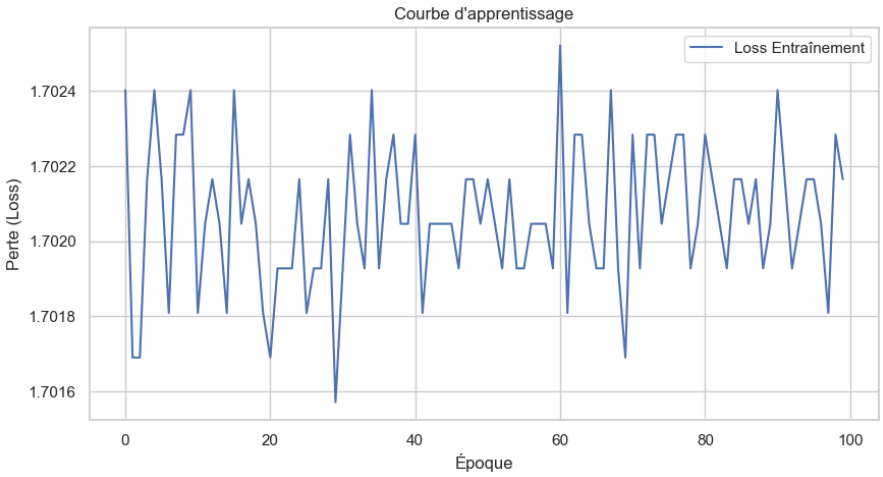

Fig. 6 Training curve of the basic model.

The second model, shown in Figure 7, yields better performance, but still does not reach the desired level of accuracy. The curves highlight improvements, yet some aspects of the model could be further optimized.

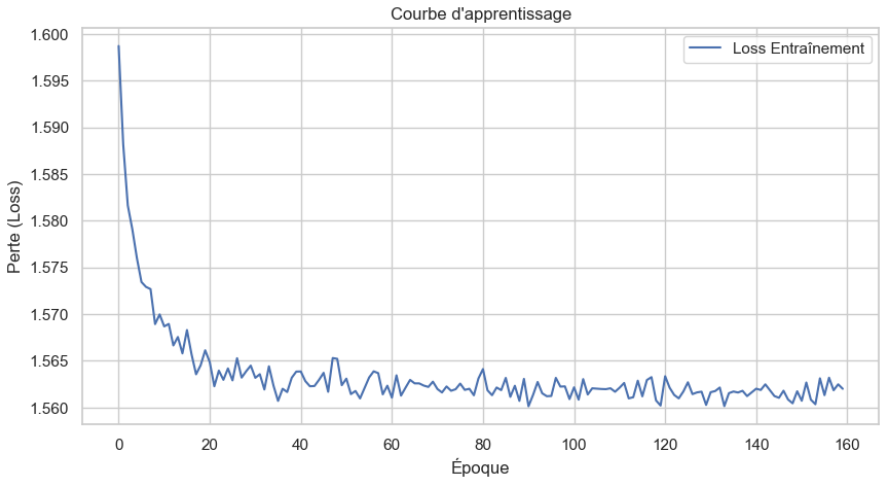

Fig. 7 Training curve of the modified model.

The third model, illustrated in Figure 8, has the same training problems as the first. The model still fails to achieve satisfactory accuracy, indicating that further adjustments are needed.

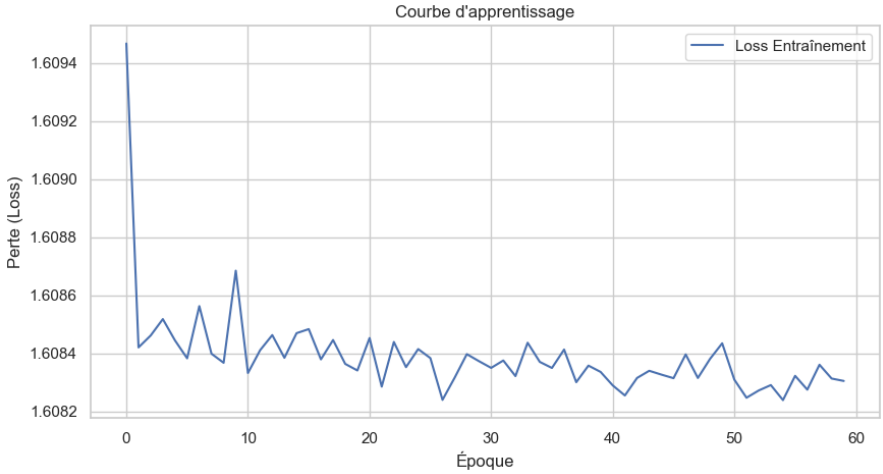

Fig. 8 Training curve of the text model.

The detailed results for all models are summarized in Table 2. This table provides a comprehensive comparison of the performance metrics across the three models.

Table 2 Results

Metrics	Basic model (First)	Modified model (Second)	Text model (Third)
Accuracy	0.202	**0.221**	0.214
F1 score macro	0.067	**0.159**	0.070
Recall	0.2	**0.211**	0.2
Precision	0.040	**0.266**	0.043

5 Conclusion

The best result was achieved using a Random Forest classifier. To further enhance the study and obtain even better results, it would be beneficial to explore the use of pre-trained models such as BERT or RoBERTa. These models, which have been trained on large text corpora, could improve the performance of the classification task, particularly by leveraging their ability to capture more complex patterns and relationships within the data.