Groupe de statistique appliquée

Mémoire

Multi-label music genre classification of tracks

Zineb Bentires, Sirine Louati, Louise Sirven, Mélissa Tamine

zineb.bentires@ensae.fr, sirine.louati@ensae.fr,
louise.sirven@ensae.fr, melissa.tamine@ensae.fr

ENSAE 2ème année - Année scolaire 2021-2022

19 mai 2022

Table des matières

1 Introduction

Les systèmes de recommandation de chansons sont au cœur du succès grandissant des plateformes permettant d'écouter de la musique en ligne. La construction des systèmes de recommandation pour un utilisateur donné repose sur deux stratégies différentes : la recommandation fondée sur les composantes intrinsèques des chansons écoutées ou bien sur les choix musicaux d'autres utilisateurs similaires à l'utilisateur cible [Koren *et al.*, 2009].

Parmi les composantes musicales utilisées pour la recommandation, le genre musical apparaît comme une donnée cruciale, tant pour la construction des systèmes de recommandation, que pour la classification des musiques présentes sur les plateformes musicales. Sur un plan plus théorique, alors que la stagnation des performances des systèmes de classification automatique des genres ont pu conduire certains à appeler à l'abandon de ce champ de recherche au profit de travaux à un niveau plus général sur les systèmes de recommandation ou les similitudes, de récents travaux ont montré l'importance qu'accordent les utilisateurs finaux à la recommandation par genre [McKay et Fujinaga, 2006], prouvant ainsi que ce type de classification reste, encore aujourd'hui, un sujet de recherche pertinent.

Une première difficulté liée à la classification par genre réside dans le caractère très subjectif de la définition d'un genre musical : à la différence du style musical, le genre musical d'une chanson est fortement influencé par des caractéristiques culturelles et les caractéristiques d'un genre sont souvent ambiguës, voire contradictoires, suivant les sources considérées [McKay et Fujinaga, 2006].

La classification par genre se fonde, la plupart du temps, sur des données audio (timbre, ton du morceau, etc), sur le scraping web ou encore, moins couramment, sur d'autres types de données comme le texte (paroles des chansons, titre) ou l'image (couvertures d'albums). Bien souvent, les approches multimodales (i.e. s'appuyant sur des données de types différents) surpassent en performances les approches unimodales [Oramas *et al.*, 2017]. Aujourd'hui, de nombreuses approches s'appuient sur des techniques de *deep learning* (réseaux de neurones convolutifs par exemple [Choi *et al.*, 2016]) ou de factorisation matricielle [Koren *et al.*, 2009].

C'est dans ce contexte que s'inscrit notre projet de statistique appliquée, issu d'un sujet proposé par la plateforme musicale Deezer. Sur cette plateforme, les labels associés aux titres (genre, époque, instruments, etc) sont issus d'un travail d'étiquetage manuel ou bien de web-scraping. Les labels utilisés dans la base sur laquelle nous travaillons sont issus de l'étiquetage manuel des chansons par les "editos" (les spécialistes de la musiques qui créent les playlists sur Deezer). Cela permet d'associer un genre à une partie des chansons, mais cet étiquetage n'est pas exhaustif. A partir de données issues du signal audio des titres musicaux et de données liées aux auditeurs, le projet présenté dans ce mémoire vise à construire un modèle permettant d'associer à chaque titre de la base un label de genre. Un même titre pouvant être associé à différents genres, nous nous plaçons ici dans le cadre de la classification multilabel.

Notre mémoire s'articule en quatre parties : après avoir présenté la base de données sur laquelle nous travaillons, puis les métriques qui nous permettront d'évaluer les performances des méthodes de classification que nous utilisons, nous présentons successivement les modèles que nous avons implémentés ainsi que leurs performances avant de conclure sur les modèles les plus performants.

2 Analyse descriptive des données

La base reçue est sous format *.parquet*. Pour la lire, nous utilisons la méthode read.parquet du package pandas en spécifiant engine="pyarrow".
La base de données contient 56763 lignes et 26 variables :

- 21 types de genres musicaux de type binaire : le type le plus représenté est le Rock avec 16% suivi du Pop avec 13% et du Metal avec 12% . La figure 1 donne la répartition des genres dans la base. On constate une inégale répartition : seuls trois genres sont au dessus de la barre des 10% et 9 genres représentent moins de 9% des genres étiquetés. Le Blues et l'Asian sont les genres les moins représentés. Ici les labels ont été déterminés par un panel de personnes au sein de l'entreprise (les editos de Deezer) et leur exactitude peut être discutée.
- un index
- 2 groupes de variables : des embeddings pour les audio features de dimension 256 et des embeddings pour les usage features de dimension 128. Les audio features sont basés sur les caractéristiques sonores et musicales du titre (fréquence, type d'instruments utilisés, variations de volume...) Les usage features reposent sur l'utilisation de la plateforme par les utilisateurs, il s'agira donc d'un agrégat des caractéristiques et données correspondant aux personnes écoutant ce genre de musique en particulier. Ainsi, lorsque les audio features ont une importance élevée sur un modèle et sur un genre cela signifie que ce genre est caractérisable par ses attributs techniques. En revanche, lorsque l'importance des usage features est plus élevée, cela signifie que le genre est davantage expliqué à travers l'ensemble des personnes qui l'écoutent, c'est-à-dire qu'il est co-écouté par des personnes présentant des caractéristiques reconnaissables.
- 2 variables de type chaînes de caractères : le titre du morceau et le nom du chanteur.

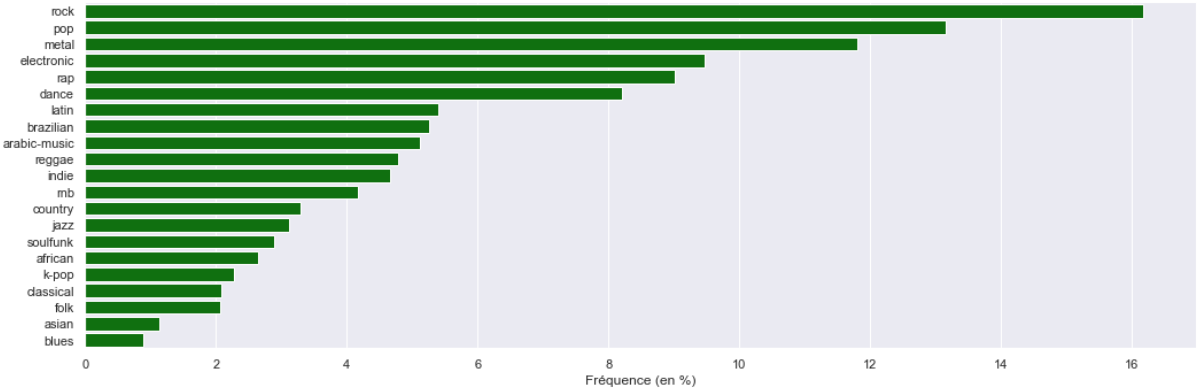


FIGURE 1 – La fréquence d'apparition des genres dans le jeu de données

2.1 La répartition des labels dans la base de données

Il est intéressant d'étudier la distribution des labels dans le data-set : quel est le nombre moyen de labels par titre ? Ce nombre est-il faible comparé au nombre possible de labels ? Ces valeurs sont différentes entre jeux de données et peuvent être à l'origine des performances différentes des algorithmes de classification.

Dans notre base de données, le nombre de labels par titre varie entre 1 et 6, mais la répartition est inégale (*Tableau 1*) : 84% des titres sont labellisés seulement par un seul genre et seules 3 chansons se voient associer 6 labels chacune.
On note N le nombre de titres dans la base de données, Y_i le jeu de labels associé au titre X_i et L le jeu de labels possibles. Deux grandeurs sont particulièrement pertinentes pour

Nombre de labels par chanson	Proportion de titres
1	84%
2	14%
3	1,6%
4	0,2%
5	0,02%
6	0,006%

TABLE 1 – Proportion de chansons en fonction du nombre de labels associés à la chanson

appréhender la distribution des labels dans le data-set [Tsoumakas et Katakis, 2009] :

— Le nombre moyen de labels par titre (*label cardinality*) :

$$LC(N) = \frac{1}{N}\sum_{i=1}^{N}|Y_i| \tag{1}$$

— La densité (*label density*) qui permet de comparer le nombre moyen de labels par titre au nombre total de labels possibles :

$$LD(N) = \frac{LC(N)}{|L|} = \frac{1}{N}\sum_{i=1}^{N}\frac{|Y_i|}{|L|} \tag{2}$$

Pour notre jeu de données, $LC = 1,18$ et $LD = 0,056$.

2.2 Liens entre les différents genres

Une même chanson peut être labellisée par plusieurs genres musicaux. Certains genres sont plus proches que d'autres ; on peut donc s'attendre à retrouver certaines associations de genres musicaux lorsqu'une chanson est labellisée par plusieurs genres.

La figure 2 permet de visualiser sous forme de graphe les liens entre les différents genres. Sur ce graphe, les noeuds correspondent aux genres de la base de données et la taille de ces noeuds est proportionnelle aux nombres de chansons étiquetées par ce genre. Les arrêtes du graphe permettent de visualiser les liens entre les différents genres : deux genres sont reliés par une arrête lorsqu'ils labellisent tous les deux une même chanson. La taille des arrêtes est proportionnelle au nombre de chansons labellisées par les deux genres à la fois : *pop-métal, dance-electronic et pop-rnb* semblent donc les associations les plus représentées dans la base de données.

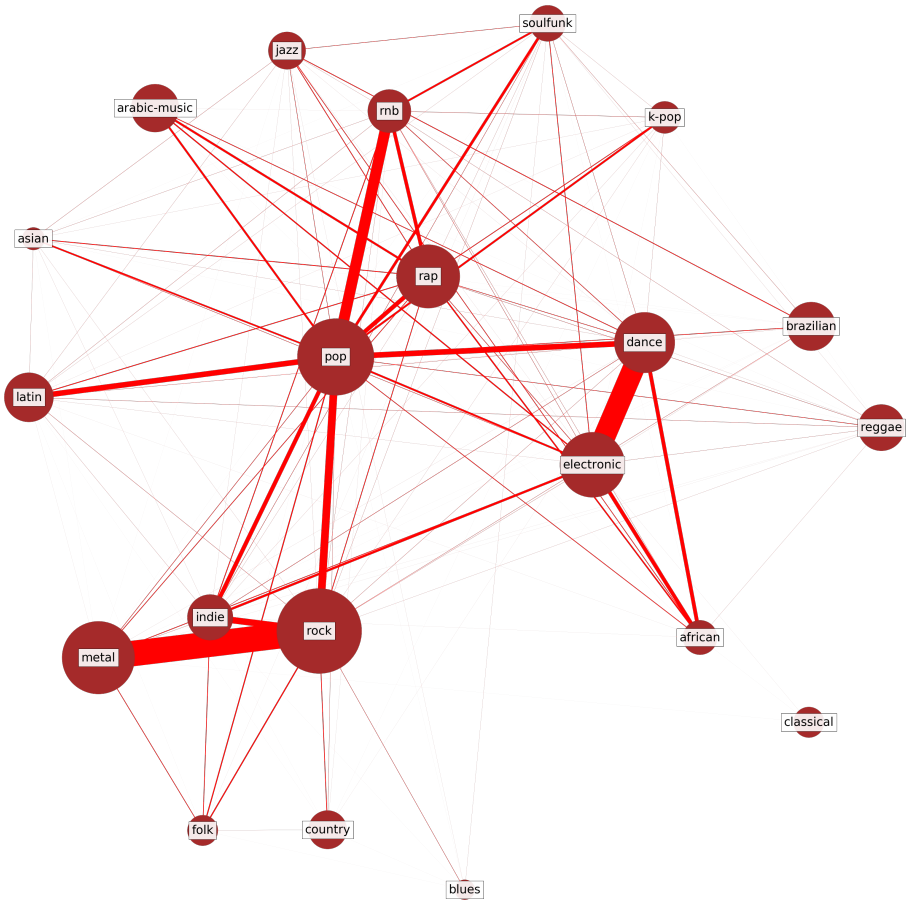


FIGURE 2 – Représentation sous forme de graphe de la distribution des labels

	asian	rnb	reggae	blues	pop	dance	folk	arabic-music	indie	rock	soulfunk	latin	classical	k-pop	brazilian	metal	rap	jazz	electronic	african	country
asian	1.00	-0.01	-0.02	-0.01	0.02	-0.03	-0.02	-0.03	-0.02	-0.04	-0.02	-0.02	-0.02	-0.01	-0.02	-0.04	-0.00	-0.00	-0.02	-0.02	-0.02
rnb	-0.01	1.00	-0.04	-0.02	0.13	-0.05	-0.03	-0.05	-0.02	-0.09	0.04	-0.05	-0.03	-0.02	-0.05	-0.07	0.01	-0.03	-0.06	-0.02	-0.04
reggae	-0.02	-0.04	1.00	-0.02	-0.08	-0.06	-0.03	-0.05	-0.05	-0.09	-0.04	-0.05	-0.03	-0.03	-0.05	-0.08	-0.07	-0.04	-0.07	-0.03	-0.04
blues	-0.01	-0.02	-0.02	1.00	-0.04	-0.03	-0.01	-0.02	-0.02	-0.03	-0.01	-0.02	-0.01	-0.01	-0.02	-0.03	-0.03	-0.01	-0.03	-0.02	-0.02
pop	0.02	0.13	-0.08	-0.04	1.00	-0.04	-0.03	-0.06	-0.00	-0.09	-0.00	0.00	-0.06	-0.00	-0.08	-0.14	-0.07	-0.06	-0.10	-0.05	-0.07
dance	-0.03	-0.05	-0.06	-0.03	-0.04	1.00	-0.04	-0.06	-0.06	-0.13	-0.04	-0.07	-0.04	-0.04	-0.07	-0.11	-0.08	-0.05	0.28	0.06	-0.05
folk	-0.02	-0.03	-0.03	-0.01	-0.03	-0.04	1.00	-0.03	-0.00	-0.03	-0.02	-0.03	-0.02	-0.02	-0.03	-0.03	-0.05	-0.03	-0.05	-0.02	-0.02
arabic-music	-0.03	-0.05	-0.05	-0.02	-0.06	-0.06	-0.03	1.00	-0.05	-0.10	-0.04	-0.06	-0.04	-0.04	-0.06	-0.09	-0.02	-0.04	-0.05	-0.04	-0.04
indie	-0.02	-0.02	-0.05	-0.02	-0.00	-0.06	-0.00	-0.05	1.00	0.02	-0.03	-0.05	-0.03	-0.03	-0.05	-0.08	-0.07	-0.04	-0.03	-0.04	-0.04
rock	-0.04	-0.09	-0.09	-0.03	-0.09	-0.13	-0.03	-0.10	0.02	1.00	-0.07	-0.10	-0.06	-0.06	-0.10	0.13	-0.13	-0.08	-0.14	-0.07	-0.06
soulfunk	-0.02	0.04	-0.04	-0.01	-0.00	-0.04	-0.02	-0.04	-0.03	-0.07	1.00	-0.04	-0.03	-0.02	-0.03	-0.06	-0.05	-0.01	-0.04	-0.03	-0.03
latin	-0.02	-0.05	-0.05	-0.02	0.00	-0.07	-0.03	-0.06	-0.05	-0.10	-0.04	1.00	-0.04	-0.04	-0.06	-0.09	-0.06	-0.04	-0.08	-0.04	-0.04
classical	-0.02	-0.03	-0.03	-0.01	-0.06	-0.04	-0.02	-0.04	-0.03	-0.06	-0.03	-0.04	1.00	-0.02	-0.03	-0.05	-0.05	-0.03	-0.05	-0.03	-0.03
k-pop	-0.01	-0.02	-0.03	-0.01	-0.00	-0.04	-0.02	-0.04	-0.03	-0.06	-0.02	-0.04	-0.02	1.00	-0.03	-0.05	-0.03	-0.03	-0.05	-0.03	-0.03
brazilian	-0.02	-0.05	-0.05	-0.02	-0.08	-0.07	-0.03	-0.06	-0.05	-0.10	-0.03	-0.06	-0.03	-0.03	1.00	-0.08	-0.07	-0.01	-0.07	-0.04	-0.04
metal	-0.04	-0.07	-0.08	-0.03	-0.14	-0.11	-0.03	-0.09	-0.08	0.13	-0.06	-0.09	-0.05	-0.05	-0.08	1.00	-0.11	-0.06	-0.11	-0.06	-0.07
rap	-0.00	0.01	-0.07	-0.03	-0.07	-0.09	-0.05	-0.02	-0.07	-0.13	-0.05	-0.06	-0.05	-0.03	-0.07	-0.11	1.00	-0.03	-0.10	-0.01	-0.06
jazz	-0.00	-0.03	-0.04	-0.01	-0.06	-0.05	-0.03	-0.04	-0.04	-0.08	-0.01	-0.04	-0.03	-0.03	-0.01	-0.06	-0.03	1.00	-0.06	0.00	-0.03
electronic	-0.02	-0.06	-0.07	-0.03	0.10	0.28	-0.05	-0.05	-0.03	-0.14	-0.04	-0.08	-0.05	-0.05	-0.07	-0.11	-0.10	-0.06	1.00	0.05	-0.06
african	-0.02	-0.02	-0.03	-0.02	-0.05	0.06	-0.02	-0.04	-0.04	-0.07	-0.03	-0.04	-0.03	-0.03	-0.04	-0.06	-0.01	0.00	0.05	1.00	-0.03
country	-0.02	-0.04	-0.04	-0.02	-0.07	-0.05	-0.02	-0.04	-0.04	-0.06	-0.03	-0.04	-0.03	-0.03	-0.04	-0.07	-0.06	-0.03	-0.06	-0.03	1.00

FIGURE 3 – Matrice de corrélation (méthode Pearson) entre les genres musicaux
A titre d'exemple, la corrélation entre le Dance et l'Electronic est de 0.28, celle entre le Metal et le Rock est de 0.13 et celle entre le Pop et le Rnb est de 0.13 également. On observe également des anticorrélations : par exemple, la corrélation entre Metal et Pop est de -0.14, celle entre Electronic et Rock est de -0.14 et celle entre Dance et Rock est de -0.13.

La matrice de corrélation entre les genres selon la méthode 'Pearson' (*Figure 3*) montre, quant à elle, que la corrélation entre les genres musicaux est relativement faible. Cela peut potentiellement être dû au fait que 84% des titres sont labellisés par un seul genre. Ces corrélations faibles n'empêchent pas de remarquer des corrélations positives entre

certains genres (*Dance-Electronic*, *Metal-Rock*, ou encore *Pop-Rnb*). Ces corrélations sont cohérentes avec la représentation précédente du data-set sous forme de graphe figure 2. A contrario, on remarque des corrélations négatives entre certains genres (*Rock-Electronic*, *Metal-Pop*, ou encore *Dance-Rock* par exemple).

Nous avons également représenté les variables correspondant aux caractéristiques du signal audio (*audio features*) ainsi que celles correspondant aux caractéristiques d'usage des utilisateurs (*usage features*) de quelques titres musicaux de l'ensemble de données. Pour cela, nous avons utilisé l'algorithme t-SNE (*t-distributed stochastic neighbor embedding*) qui est une technique de réduction de dimension pour la visualisation de données. Ainsi, nous avons pu représenter des ensembles de points d'un espace à grande dimension (de dimension 256 dans le cas des *audio features* et 128 dans le cas des *usage features*) dans un espace de deux dimensions. Les résultats obtenus sont présentés figure 4.

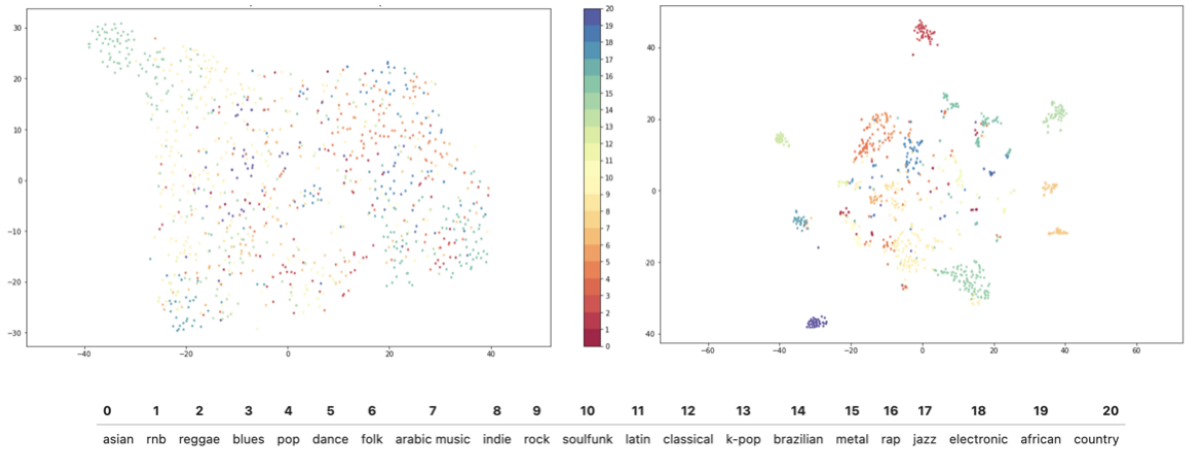


FIGURE 4 – Visualisation après réduction de dimension par l'algorithme t-SNE des *audio features* (à gauche) et des *usage features* (à droite) pour 1000 titres musicaux de l'ensemble de données

Les graphes de la figure 4 indiquent la présence de *clusters* associés à un même label pour les *usage features*, ce qui n'est pas le cas pour les *audio features*. Cela semblerait signifier que les *usage features* permettent davantage de caractériser le genre musical que les *audio features*.

3 Choix des métriques d'évaluation

Les problèmes de classification multilabel requièrent des métriques différentes de celles habituellement utilisées dans les problèmes de classification avec un seul type de label. En effet, dans le cas spécifique de la classification multilabel, il apparaît important que la métrique puisse évaluer le caractère partiellement correct des prédictions (pour une chanson, certains labels sont justes et pas d'autres) et pas seulement le caractère complètement juste ou non d'une prédiction.

Les métriques utilisées pour la classification multilabel sont bien souvent inspirées et adaptées des métriques utilisés pour le cas d'un seul label [Tsoumakas et Katakis, 2009] : on calcule les scores des métriques habituelles en unilabel, puis on moyenne ces résultats afin d'obtenir un résultat prenant en compte le caractère multilabel du problème. Pour ce qui est de la moyenne, deux options sont possibles : la moyenne macro ou la moyenne micro. La moyenne macro est une simple moyenne des scores obtenus pour chacun des la-

bels. La moyenne micro est quant à elle calculée en considérant tous les faux positifs, faux négatifs, vrais positifs et vrais négatifs de tous les genres, puis en calculant les métriques habituelles. La moyenne micro est recommandée pour des data-sets non équilibrés. C'est cette moyenne que nous considèrerons. Nous présentons ci-dessous plus en détails les métriques utilisées pour évaluer nos différents modèles.

De même que précédemment, nous notons N le nombre de titres dans la base de données et L le nombre de labels possibles. Nous définissons alors quatre grandeurs qui nous serons utiles pour calculer les métriques en moyenne micro :

— *Faux positifs* : La somme sur l'ensemble des genres des faux positifs (*ie* les titres prédits comme appartenant à un certain genre alors qu'ils ne sont pas labellisés par ce genre dans la base de données) de chaque genre. En notant FP_i les faux positifs du label i :

$$FP = \sum_{i=1}^{L} FP_i \tag{3}$$

— *Faux négatifs* : La somme sur l'ensemble des genres des faux négatifs (*ie* les titres prédits comme n'appartenant pas à un certain genre alors qu'ils sont en réalité labellisés par ce genre dans la base de données) de chaque genre.En notant FN_i les faux négatifs du label i :

$$FN = \sum_{i=1}^{L} FN_i \tag{4}$$

— *Vrais positifs* : La somme sur l'ensemble des genres des vrais positifs (*ie* les titres prédits comme appartenant à un certain genre et réellement étiquetés par ce genre dans la base de données) de chaque genre. En notant VP_i les vrais positifs du label i :

$$VP = \sum_{i=1}^{L} VP_i \tag{5}$$

— *Vrais négatifs* : La somme sur l'ensemble des genres des vrais négatifs (*ie* les titres prédits comme n'appartenant pas à un certain genre et réellement non étiquetés par ce genre dans la base de données) de chaque genre. En notant VN_i les vrais négatifs du label i :

$$VN = \sum_{i=1}^{L} VN_i \tag{6}$$

Ces grandeurs ainsi définies nous permettent de calculer les métriques de performance pour le multilabel :

— *Hamming Loss* : Il s'agit de la fraction de labels erronés sur le nombre total de combinaisons de labels possibles.

$$HL = \frac{FN + FP}{N \cdot L} \tag{7}$$

— *Accuracy* : Il s'agit de la fraction de labels du dataset correctement prédits.

$$A = \frac{VP + VN}{N \cdot L} \tag{8}$$

— *Precision* : il s'agit de la moyenne micro de la precision unilabel. Cette métrique mesure la capacité du modèle à ne pas prédire trop de faux positifs parmi les labels positifs qu'il prédit.

$$P = \frac{VP}{VP + FP} \tag{9}$$

— *Recall* : il s'agit de la moyenne micro du *recall* unilabel. Cette métrique mesure la fraction de labels positifs que le modèle arrive à prédire parmi tous les positifs de la base.

$$R = \frac{VP}{VP + FN} \tag{10}$$

— *F1 score* : il s'agit de la moyenne harmonique entre la *precision* et le *recall*. Le *F1 score* permet donc de combiner les résultats de ces deux mesures.

$$F1 = 2 \cdot \frac{R \cdot P}{R + P} \tag{11}$$

Une *Hamming Loss* proche de 0 et des *precision, recall, accuracy* et *F1-score* proches de 1 sont signes d'un bon modèle.

4 Modélisation

4.1 Modèles naïfs

4.1.1 Classifieur randomisé

Le premier modèle que nous avons implémenté est un *random* ou *dummy classifier* (classifieur randomisé). Il s'agit d'un classifieur réalisant des prédictions qui ne tiennent pas compte des données en entrée. Il attribue en effet de manière aléatoire un genre/un label à chaque titre musicaux de l'ensemble de données *test*.

Ce classifieur est utile dans la mesure où il sert de base de comparaison simple à d'autres classifieurs plus complexes.

4.1.2 Modèle « Top genre »

La deuxième méthode implementée est un *top-genre classifier*. Le principe est d'attribuer à l'ensemble des titres musicaux de l'échantillon *test*, un unique label : le label le plus représenté sur l'échantillon *train*. Ce modèle est très naïf mais tout comme le *random classifier*, il nous servira de base. Autrement dit, les performances des modèles étudiés (pour toutes les métriques) seront comparées à la performance du modèle *top genre*.

4.1.3 Modèle « Average-by-genre »

La troisième et dernière méthode naïve implementée est un *average-by-genre classifier*, modèle un peu plus élaboré que les deux premiers. Comme pour les deux premiers modèles, cette méthode n'attribue qu'un seul genre à chaque titre. Ce modèle se décompose en deux étapes principales :

— On modélise chaque genre par la moyenne des features (audio et/ou usage) des tracks de l'échantillon *train* correspondant à chacun des genres.
— On attribue à chaque titre de l'échantillon *test* le genre dont les *features moyennes* sont les plus proches des *features* du titre. Pour cela, nous comparons le vecteur des moyennes des *features* de chaque label au vecteur de *features* de chaque titre en utilisant la distance euclidienne ; le cosine normalisé ne donnant pas des résultats satisfaisants.

Les résultats du modèle *average-by-genre* dépendent des features prises en compte (audio ou usage). Nous avons testé ce modèle en considérant les *audio features* et les *usage features* seules, puis en considérant ces deux types de features à la fois. Les *usage features*

apparaissent comme les plus appropriées pour prédire le genre musical dans ce modèle (résultats table 2). La meilleure performance du modèle avec les usages features seules est confirmée par la données des AUC par label (figure 13 en annexe) ; on y note également des différences entre les genres : le Classique par exemple obtient de bons résultats avec les *audio features* seules, alors que les résultats pour le Blues sont bien supérieurs en prenant en compte les usage features seules.

4.1.4 Conclusion sur les modèles naïfs

La table 2 donne les résultats obtenus sur la base de données avec les différentes métriques décrites précédemment. Le modèle *average-by-genre* sur les usage features seules est le modèle qui obtient les meilleurs résultats et ce pour toutes les métriques.

Méthode	*Accuracy*	*Précision micro*	*Recall micro*	*F1 score*	*Hamming-Loss*
Classifier randomisé	0.047	0.047	0.047	0.047	0.950
Top genre	0.108	0.162	0.138	0.149	0.088
Average-by-genre (usage features)	**0.795**	**0.795**	**0.795**	**0.795**	**0.019**
Average-by-genre (audio features)	0.444	0.444	0.444	0.444	0.053
Average-by-genre (audio et usage features)	0.487	0.487	0.487	0.487	0.049

TABLE 2 – Résultats obtenus pour les différentes métriques avec les méthodes naïves
Les résultats en gras sont ceux du meilleur modèle. Le modèle d'Average-by-genre utilisant seulement les usage features apparaît comme le modèle naïf le plus performant pour toutes les métriques considérées. Les scores similaires obtenus pour l'accuracy, la précision, le recall et le F1-score pour tous les modèles du tableau sauf le Top genre ne viennent pas du hasard. En effet, en prenant la moyenne micro sur des données unilabels, on peut montrer mathématiquement que cette moyenne est identique entre ces métriques.

4.2 Régression logistique

4.2.1 Cadre de travail

Le cas de la classification Unilabel Soient $(x_i, y_i)_{1\leq i\leq n}$ et $(x_i, y_i)_{1\leq i\leq m}$ où $n, m \in \mathbb{N}$ des échantillons. Ici nous sommes dans le cas de la classification binaire usuelle i.e $\forall i, y_i \in \{0, 1\}$ et $x_i \in \mathbb{R}^p$. On considère que le premier échantillon est la partie *train*, le second la partie *test*. L'approche consiste donc à modéliser le problème comme suit :

$$\forall i, \hat{y}_i = \mathbb{1}_{\sigma(x_i^T\beta)\geq 0.5} \text{ où } \sigma(z) = \frac{1}{1+e^{-z}}$$

Cette représentation a l'avantage de présenter une fonction de lien σ dont l'image est comprise entre 0 et 1. Nous réalisons la prédiction de probabilité, puis la classification se fait par le biais d'un seuil.
Nous cherchons donc $\beta^* \in \mathbb{R}^p$ qui minimisera la fonction *loss* ou de *coût*. Puisque nous sommes dans le cas d'une classification binaire dans un modèle de machine learning, il s'agira de minimiser :

$$\frac{1}{n}\sum_{i=1}^{n} \mathbb{1}_{y_i\neq\hat{y}_i}$$

En utilisant le fait que cela équivaut à maximiser la vraisemblance, et donc la log-vraisemblance par rapport à β, nous obtenons une forme non close et nous procédons

à une descente de gradient afin d'obtenir une approximation de β^*. En effet la fonction à minimiser est :

$$\frac{1}{n}\sum_{i=1}^{n}\log(1+\exp(x_i^T\beta)) - y_i x_i^T\beta$$

Nous nous servons uniquement de l'échantillon *train*, comme nous pouvons le remarquer à la forme de l'expression à minimiser. Nous utiliserons ensuite l'échantillon *test*, indépendant du premier pour évaluer les prédictions du modèle.

Le cas de la classification multilabel Dans notre exemple, nous avons 21 labels musicaux. Ici notre variable cible est définie comme suit :

$$y_i \in \mathbb{R}^{21}, \forall 1 \leq l \leq 21, y_i^l = \begin{cases} 1 & \text{si la musique i est classifiée comme label l} \\ 0 & \text{sinon.} \end{cases}$$

Pour la classification multilabels, on effectue 21 classifications unilabel en prenant pour échantillon $(x_i, y_i^l), \forall 1 \leq l \leq 21l$. Le résultat final sera donc :

$$\hat{y}_i \in \mathbb{R}^{21}, \forall 1 \leq l \leq 21, \hat{y}_i^l = \begin{cases} 1 & \text{si la musique i est } \textbf{prédite} \text{ dans le label l} \\ 0 & \text{sinon.} \end{cases}$$

4.2.2 Les sous cas étudiés

Notre base de données nous fournit des variables explicatives de deux catégories distinctes : **audio** et **usage**. Afin d'évaluer l'importance des ces variables pour la classification, nous réalisons 3 types de régressions :

- audio features seulement
- usage features seulement
- audio et usage

Il y a 256 variables audios et 128 variable usages. Nous sommes donc confrontés au problème du fléau de la dimension. A cause du grand nombre de variables et de leur corrélation probable, un problème **sur-apprentissage** peut survenir, aussi les coefficients associés aux différentes variables sont plus difficile à interpréter au vu de leur quantité. Les métriques évaluées sont très proches de l'optimum (1 sauf pour la Hamming Loss : 0) pour l'échantillon d'apprentissage, mais ne sont pas aussi bonnes pour l'échantillon test ; le modèle "colle" trop aux données. Pour résoudre ce problème, on effectue une sélection de variables en utilisant une pénalisation L^1 c'est-à-dire un Lasso. Ceci se modélise comme suit ; pour chacune des 21 régressions unilabel, la fonction à minimiser devient :

$$\frac{1}{n}\sum_{i=1}^{n}\log(1+\exp(x_i^T\beta)) - y_i x_i^T\beta + \lambda\|\beta\|_1$$

Le paramètre λ est appelé paramètre de pénalisation et est positif. S'il est égal à 0, il n'y a aucune pénalisation, c'est la régression logistique classique qui est effectuée. En revanche s'il est très grand, très peu de variables seront sélectionnées. Cette fonction, grâce aux caractéristiques de la norme 1 a pour avantage de mettre des coefficients à 0 et donc de sélectionner des variables. Ceci permettra de réduire la variance, c'est-à-dire le sur-apprentissage, et donc d'améliorer en principe les résultats sur l'échantillon test. Ainsi, pour le dernier type de régression cité ci-dessus, on pourra notamment observer quel type de variable sera sélectionné.

4.2.3 Les résultats globaux

Les résultats obtenus pour différentes combinaisons de paramètres sont présentés tableau 3. Les métriques utilisées sont celles du multilabel, c'est-à-dire qu'on évalue la performance en considérant la labellisation originale et l'union des labels pour la prédiction.

Méthode	*Accuracy*	*Précision micro*	*Recall micro*	*F1 score*	*Hamming-Loss*
RL audio	0.360	0.725	0.432	0.541	0.041
RL audio pénalisée (optimisée)	0.354	0.738	0.421	0.536	0.041
RL usage	0.715	0.871	0.788	0.827	**0.018**
RL usage pénalisée (optimisée)	0.715	0.871	0.787	0.827	**0.018**
RL audio usage	**0.720**	0.870	**0.797**	**0.832**	**0.018**
RL audio usage pénalisée (optimiséee)	0.718	**0.875**	0.788	0.829	**0.018**

TABLE 3 – Résultats obtenus pour les différentes métriques avec la méthode de régression logistique
Plus l'accuracy, le recall, la precision et le F1 score sont proches de 1 et plus le modèle est performant. Plus la Hamming-Loss est proche de 0 et plus le modèle est performant. Pour chaque métrique, le résultat en gras correspond au modèle le plus performant pour cette métrique. Le modèle non pénalisé avec les audio et les usage features semble le plus performant globalement.

Dans un premier temps, on note que la régression logistique performe aussi bien ou mieux sans pénalisation pour les deux cas d'*audio features* ou *usage features* seulement. Cela est dû au fait que le modèle ne fait pas d'*overfitting* au départ ; les variables au sein d'une même catégorie peuvent donc être supposées non corrélées entre elles.

D'autre part, les caractéristiques *usage* semblent 1,5 à 2 fois plus efficaces en moyenne pour prédire les labels du point de vue de l'*accuracy,* du *recall*, du *F1 score*, et de la *Hamming-Loss* : toutes les métriques sauf la précison. Le meilleur résultat pour toutes les métriques est atteint pour la régression logistique prenant en entrée les *audio* et *usage features*. Lorsque ce modèle est pénalisé avec une pénalisation L^1 dont l'hyperparamètre λ a été optimisé par cross-validation, le gain est minimum (0,2% par métrique en moyenne). Malgré cette augmentation de performance qui n'est pas importante, les embeddings sélectionné peuvent donner une information supplémentaire pour la création/modification de prochains embeddings.

Les histogrammes des AUC par label en fonction des méthodes disponibles en figure 5 permettent d'avoir un ordre d'idée de quels labels sont bien classifiés ou au contraire plutôt mal expliqués.

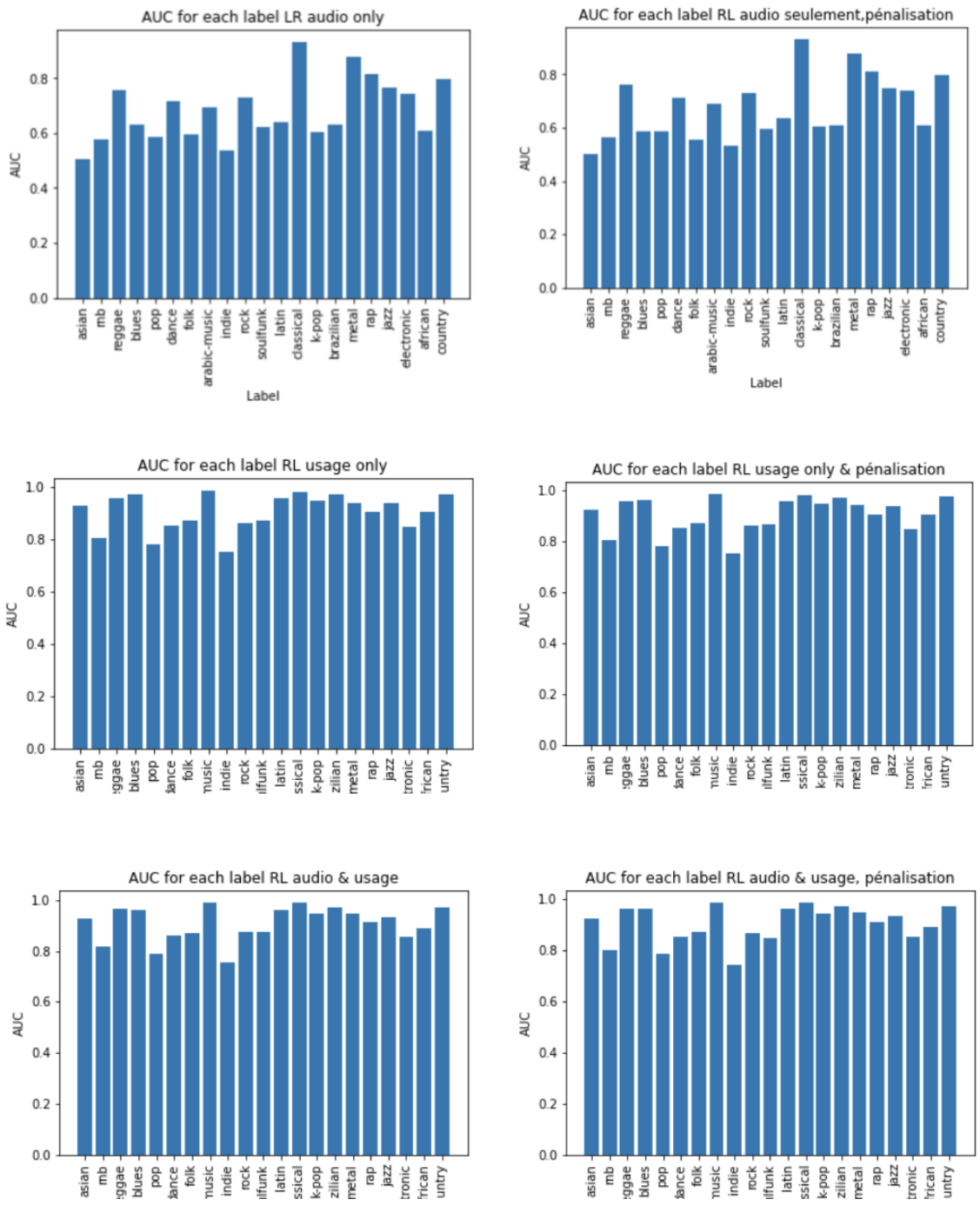


FIGURE 5 – Histogramme des AUC par genre, sans pénalisation (à gauche), avec pénalisation (à droite), respectivement de haut en bas : utilisation des *audio features* seulement, utilisation des *usage features* seulement, utilisation des *audio* et *usage features*

4.2.4 Étude de la sélection de variables

Nous considérons d'abord le nombre de variables sélectionnées lorsque nous entraînons le modèle uniquement avec les variables audio, puis uniquement les variables usage (tableau 4).

Label	% features sélectionnés (RL audio)	% features sélectionnés (RL usage)
asian	0.0	25.0
rnb	21.09	49.22
reggae	42.19	39.84
Blues	12.11	29.69
pop	44.53	43.75
dance	40.23	50.78
folk	14.84	40.62
arabic-music	42.19	43.75
indie	44.53	48.44
rock	25.78	57.03
soulfunk	19.14	46.09
latin	42.58	46.09
classical	29.3	32.81
k-pop	34.77	34.38
brazilian	19.92	46.88
metal	24.22	46.09
rap	19.53	44.53
jazz	16.8	39.84
electronic	17.58	52.34
african	39.84	45.31
country	35.94	39.84

TABLE 4 – Nombres de variables sélectionnées par genre

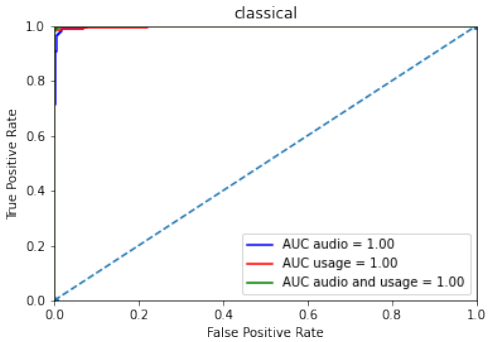
Nous regardons ensuite la répartition des variables sélectionnées lors de la régression logistique avec les deux types de features (tableau 5).

Label	Nombre de variables sélectionnées / 384	% audio	% usage
asian	42	21.43	78.57
rnb	63	33.33	66.67
reggae	116	54.31	45.69
Blues	68	51.47	48.53
pop	110	50.91	49.09
dance	88	35.23	64.77
folk	121	59.5	40.5
arabic-music	30	16.67	83.33
indie	72	40.28	59.72
rock	101	41.58	58.42
soulfunk	61	44.26	55.74
latin	127	57.48	42.52
classical	52	36.54	63.46
k-pop	36	52.78	47.22
brazilian	38	15.79	84.21
metal	86	40.7	59.3
rap	90	42.22	57.78
jazz	128	61.72	38.28
electronic	90	32.22	67.78
african	62	27.42	72.58
country	107	54.21	45.79
Moyenne sur tous les labels	80.0	41.43	58.57

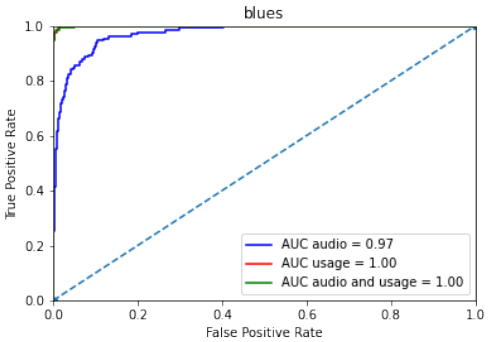
TABLE 5 – Répartition des variables sélectionnées avec utilisation des *audio* et *usage features*

4.2.5 Quelques résultats particuliers

Afin d'illustrer nos propos, nous nous concentrons sur deux genres : le Classique et le Blues. A priori, en regardant les AUC générales, le Classique semble bien performer en utilisant les features audios ; ajouter les features usages ne semble pas intéressant. C'est un résultat que l'on observera également sur d'autres modèles. En revanche, le Blues est un bon exemple de label qui se caractérise par les usages features, c'est-à-dire que les personnes écoutant ce genre l'écoutent presque exclusivement, ce qui permet la caractérisation. Cela est également observable sur les courbes ROC des deux genres (figure 6).



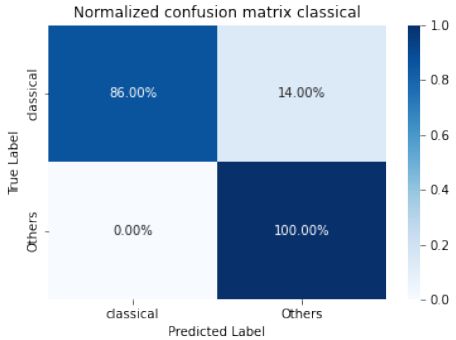
(a) Courbes ROC pour les RL pénalisées : genre Classique



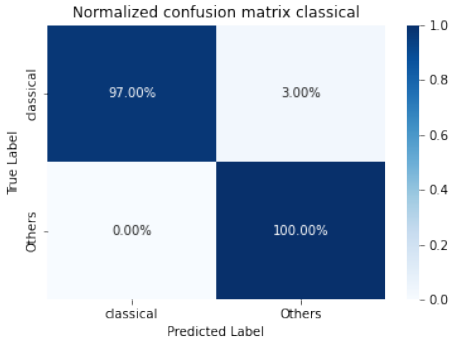
(b) Courbes ROC pour les RL pénalisées : genre Blues

FIGURE 6 – Courbes ROC Classique et Blues

De même, en comparant les matrices de confusion (figures 7 et 8) lorsqu'il y a les audio features seulement ou les usages features seulement (pénalisés), on note que le taux de TP passe de 18% à 92% pour le Blues, tandis que pour le Classique, il passe de 86% à 97%. Il semblerait, au vu des résultats globaux, que les usages features soient plus utiles pour la prédiction. Ceci est d'autant plus accentué pour des genres comme le Blues, l'Asian music, l'Arabic music, contrairement à notre intuition première. En effet, au premier abord nous avions émis l'hypothèse que les musiques éthniques ou du monde se caractérisaient par des instruments, une langue et un rythme différents, ce qui serait notable sur les audio features.

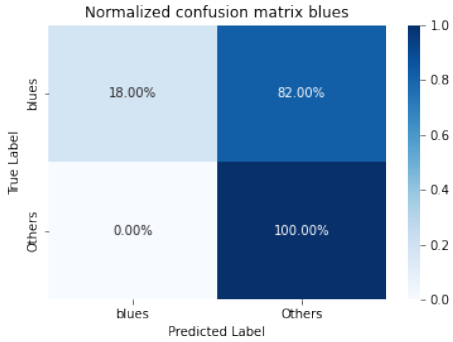


(a) Matrice de Confusion - audio features : genre Classique

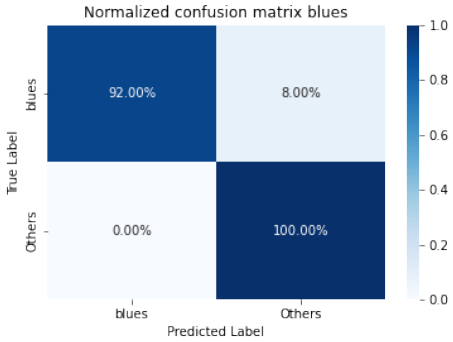


(b) Matrice de Confusion - usage features : genre Classique

FIGURE 7 – Matrices de Confusion : genre Classique



(a) Matrice de Confusion - audio features : genre Blues



(b) Matrice de Confusion - usage features : genre Blues

FIGURE 8 – Matrices de Confusion : genre Blues

4.3 Classifieur XGBoost

Comme pour le modèle de Régression Logistique, le XGBoost **ne prend pas** en compte le caractère multilabel de nos données. Il se contentera de réaliser 21 classifications binaires (unilabel) et de déterminer le résultat final en utilisant l'union de ces classifications.

4.3.1 Cadre de travail

Nous nous plaçons dans le cadre unilabel (ie $1 \leq l \leq 21$ est fixé). Soient $(x_i, y_i)_{1 \leq i \leq n}$ et $(x_i, y_i)_{1 \leq i \leq m}$ où $n, m \in \mathbb{N}$ des échantillons. Ici nous sommes dans le cas de la classification Classique i.e $\forall i, y_i \in \{0, 1\}$ et $x_i \in \mathbb{R}^p$. On considère que le premier échantillon est la partie *train*, le second la partie *test*. Comme pour la Régression Logistique, on cherche à déterminer $\forall 1 \leq i \leq n, \forall 1 \leq l \leq 21, \mathbb{P}(\hat{y}_i^l = 1)$, la probabilité que la musique i soit classifiée comme appartenant au label l.

Comme son nom l'indique, l'algorithme repose sur la technique de boosting, c'est-à-dire que plusieurs estimateurs (arbres de décisions) **non indépendants** sont combinés, chacun ayant pour but d'améliorer la marge d'erreur faite en se servant de l'estimateur précédent.

Afin de comprendre le principe général de l'algorithme, il convient de définir les principales variables et métriques utilisées. Supposons l, t fixés sans perte de généralité, où t désigne l'étape où l'on se trouve.

— **Résidu i** : Ici, une fois la probabilité $\mathbb{P}(\hat{y}_i^l = 1)$ à l'étape $t-1$ calculée, le résidu à l'étape t vaut :

$$res_{i,t} = y_i - \mathbb{P}_{t-1}(\hat{y}_i = 1) \in [-1, 1]$$

Il change à chaque étape puisque la probabilité prédite change à chaque étape.

— **Cover (ou min_child_weight)** : défini pour chaque feuille de chaque estimateur. C'est le nombre minimal de résidus qu'on doit trouver dans une feuille. Se calcule à partir de l'étape 2, puisqu'à l'étape 1 nous n'avons qu'une feuille (contenant tous les résidus). Soit f une feuille à l'étape t :

$$Cover_t^f = \sum_{i=1}^{m} \mathbb{P}_{t-1}(\hat{y}_i = 1)(1 - \mathbb{P}_{t-1}(\hat{y}_i = 1))\mathbb{1}_{res_{i,t} \in f}$$

— **Lambda** λ : paramètre de régularisation. Son utilisation réduit l'overfitting et l'impact des observations isolées.

— **Similarity Score** défini pour chaque feuille f

$$similarity\_score_{f,t} = \frac{(\sum_{i=1}^{m} res_{i,t} \mathbb{1}_{res_{i,t} \in f})^2}{cover_t^f + \lambda}$$

— **Gain** : défini pour chaque noeud, c'est-à-dire dont deux branches sont issues. Soit N le noeud, fg la feuille gauche et fd la feuille droite.

$$gain_N = similarity\_score_{fg,t} + similarity\_score_{fd,t} - similarity\_score_{N,t-1}$$

— **Level** : Profondeur maximum des arbres de décision.

— **Gamma** γ : paramètre de contrôle pour le gain. Soit N un noeud, si $_N - \gamma \leq 0$, on va élaguer, c'est-à-dire qu'on va couper des branches jusqu'à ce que l'arbre soit suffisamment performant.

— **Learning Rate** η : coefficient mis devant chaque arbre de décision pour les prédictions finales, lorsqu'il est grand cela signifie qu'on apprend beaucoup de chaque arbre.

— **Output Value** : définie pour une feuille, c'est la valeur prédite lorsque le résidu appartient à cette feuille. Ce n'est pas encore une probabilité ; une transformation préalable est nécessaire.

$$output\_value_{f,t} = \frac{\sum_{i=1}^{m} res_{i,t} \mathbb{1}_{res_{i,t} \in f}}{cover_t^f + \lambda}$$

— **Log Odds** : ou transformation logistique de la probabilité, passage intermédiaire, se calcule à partir de l'output value en tout instant t. Il est différent pour chaque valeur de l'échantillon. On définit

$$\mathbb{P}_t(\hat{y}_i = 1) = \frac{\exp lo_{t,i}}{1 + \exp lo_{t,i}}$$

L'algorithme reposant sur le principe boosting, il se compose de quatre étapes distinctes :

— Étape 1 : on construit une feuille initiale. La *value* associée au premir *log of the odds* est donc

$$lo_1 = \log(\frac{Card(i, Y_i = 1)}{n})$$

— Étape 2 : on calcule les résidus associés aux prédictions de l'échantillon d'entraînement
— Étape 3 : à l'aide des arbres de décision, on construit l'arbre le plus optimal (en maximisant le gain), en respectant les contraintes imposées pour calculer les résidus précedents
— Étape 4 : on répète récursivement les deux opérations précédentes en respectant bien la contrainte du nombre maximal d'estimateurs imposé.

Comment calcule-t-on la probabilité/prédiction associée à l'étape t ? S'il s'agit d'une valeur de l'échantillon train, on parle de probabilité estimée à l'étape t car elle est utile à l'obtention de l'estimateur en $t+1$, sinon on calcule la prédiction finale sur l'échantillon test et on prend $t = T = n\_estimateurs\_max\_train$ Soit i (dans le *train* ou le *test*) fixé, en supposant la feuille initiale et les $t-1$ arbres de décisions construits, supposons que nous avons calculé aussi $lo_{i,t-1}$ à l'étape précédente. Alors on a :

$$lo_{i,t} = lo_{i,t-1} + \eta output\_value_{f,t} * \mathbb{1}_{res_{i,t}}$$

On calcule la probabilité associée en passant par la fonction $\frac{\exp(x)}{1+\exp(x)}$

4.3.2 Les résultats globaux

Nous avons choisi d'implémenter le XGBoost avec les hyperparamètres par défaut de la librairie. En effet c'est un algorithme très efficace pour la classification unilabel sur les larges datasets, sans tunning d'hyperparamètres nécessairement. En effet, d'une part nous avons obtenu de bons résultats qui nous semblaient difficile d'améliorer au vu du caractère unilabel de la méthode. D'autre part, l'algorithme du XGBoost étant défini par une multitude d'hyperparamètres co-dépendants, la *gridsearch*, c'est à dire la comparaison des résultats d'*accuracy* pour différentes combinaisons d'hyperparamètres, est longue d'exécution. Nous obtenons les résultats suivant : (tableau 6).

Méthode	Accuracy	Precision micro	Recall micro	F1 score micro	Hamming-Loss
XGBoost	0.749	0.904	0.799	0.849	0.016

TABLE 6 – Résultats obtenus avec l'utilisation du classifieur XGBoost

L'histogramme des AUC (figure 9) montre que toutes les AUC sont très proches de 1. Le XGBoost performe très bien malgré son caractère unilabel.

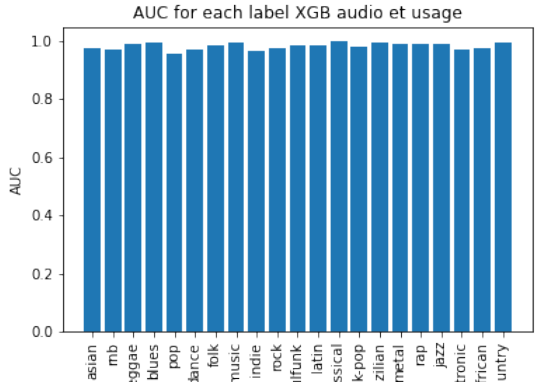
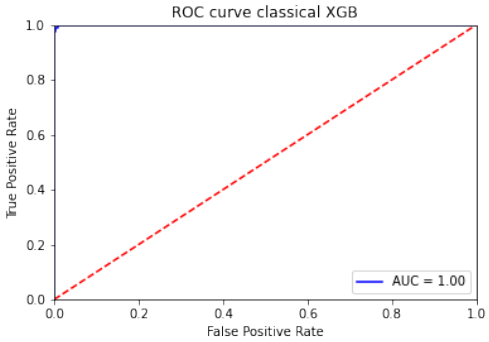


FIGURE 9 – Histogramme AUC méthode XGB standard

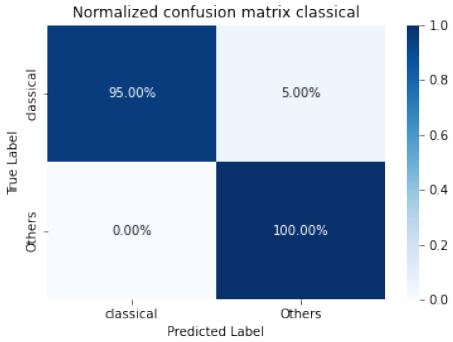
4.3.3 Quelques résultats particuliers

Comme nous pouvons le voir sur l'histogramme des AUC, le XGB performe très bien pour le genre Classique : l'AUC est à 1, nous sommmes très proche du classifieur parfait.

Genre Classique



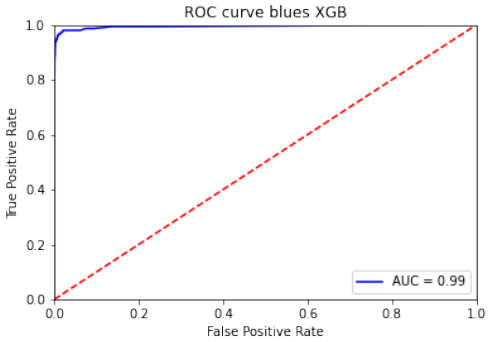
(a) ROC



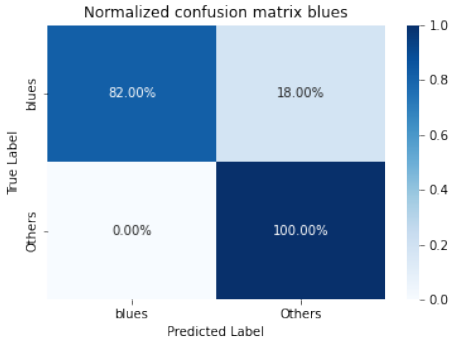
(b) Matrice de Confusion

FIGURE 10 – XGB : genre Classique

Genre Blues



(a) ROC



(b) Matrice de Confusion

FIGURE 11 – XGB : genre Blues

Nous avons aussi choisi d'optimiser les hyperparamètres par GridSearch, pour l'un des genres les moins bien prédits par le XGBoost, l'Indie. Voici les résultats obtenus pour les hyperparamètres :

- colsample_bytree : 0.7 parmi $\{0.3, 0.5, 0.7\}$
- gamma : 0.0 parmi $\{0.0, 0.1, 0.25\}$ même paramètre par défaut, on ne choisit pas d'élaguer sauf si le gain est négatif, c'est-à-dire que la nouvelle classification performe moins bien que la précédente.
- learning_rate : 0.05, par défaut il est égal à 0.3, ici on fait donc plus d'étapes. parmi $\{0.05, 0.1, 0.25\}$
- max_depth : 10, parmi $\{3, 6, 10\}$, par défaut il est égal à 6, on a donc des arbres plus profonds, plus complexes.
- min_child_weight : 1 parmi $\{1, 3, 5\}$. Même paramètre que celui par défaut, on partitionne jusqu'au bout.
- n_estimators : 100, parmi $\{100, 500, 1000\}$. Même paramètre par default, permet de contrôler le nombre d'arbres pour prendre la décision finale.

Nous avons choisi de ne pas appliquer de régularisation car il ne semble pas y avoir de sur-apprentissage. Les résultats avant et après optimisation sont présentés tableau 7

Méthode	*Accuracy*	*Précision*	*Recall*	*F1 score*	*Hamming-Loss*
XGB Indie sans optimisation	0.970	0.783	0.563	0.655	0.029
XGB Indie avec optimisation	0.975	0.846	0.597	0.70	0.025

TABLE 7 – Résultats obtenus pour la classification XGBoost du label Indie

4.4 Forêts aléatoires

4.4.1 Cadre de travail

Le *Random Forest* (ou Forêts aléatoires) est un algorithme de Machine Learning basé sur plusieurs arbres de décision créés aléatoirement par un processus de Bootstrap-Aggregating, appelé aussi Bagging.

Les arbres de décision (ou arbres décisionnels) sont construits à partir d'une série de questions appelées tests qui permet d'aboutir à une décision finale. En effet, on considère un ensemble de données constitué d'un ensemble de points $(x_1, ..., x_p, y)$ où les $(x_i)_{1\leq i\leq p}$ sont des variables explicatives (de type scalaire, catégorielle ou chaine de caractères) et y une réponse binaire .

Le but de l'algorithme est de construire une suite de règles de divisions afin de prédire y à partir des variables explicatives. A chaque itération, on fixe une variable et une règle de division définie par une valeur seuil a si la variable est quantitative ou par un partage en deux groupes des modalités disjoints si la variable est qualitative. L'objectif est d'obtenir, suite à cette division, deux sous-ensembles les plus différents possible. La règle de division orientera vers une des deux branches issues de ce noeud jusqu'à atteindre une feuille, c'est-à-dire un noeud terminal, qui contiendra la réponse finale.

L'algorithme peut s'achever de deux manières possibles :

- on arrête le split (la division d'un noeud en deux sous-branches) dès que le nombre de données par noeud est faible ou dès qu'on estime qu'un nouveau split n'aboutira pas à de meilleurs résultats.
- on split jusqu'à avoir un point de données par noeud, puis on élague l'arbre. Le pruning (élagage en français), consiste à couper des branches de l'arbre afin de réduire le sur-apprentissage et donc d'augmenter la performance de l'arbre.

Le choix du couple (variable , règle de division), et donc la procédure de split, est basé sur l'algorithme suivant :

1. Initialisation : on se place sur la première cellule C, c'est-à-dire la racine de notre arbre. On dispose alors de toutes les données au niveau de la racine.
2. Pour chaque variable i :
   - On définit V_C l'ensemble des valeurs que peut prendre la variable explicative i
   - Pour toute partition à deux ensembles de V_C disjoints et non vides $\{A, B\}$ on calcule :

$$p_- = \frac{|\{x, y\} \in A | y = 1|}{|A|}$$

$$p_+ = \frac{|\{x, y\} \in B | y = 1|}{|B|}$$

En choisissant une fonction d'indice I, on définit l'incertitude :

$$\frac{|A|}{|A|+|B|}I(p_-;1-p_-)+\frac{|B|}{|A|+|B|}I(p_+;1-p_+)$$

— On choisit parmi toutes les partitions $\{A,B\}$ possibles celle dont l'incertitude est minimale.

3. On réapplique l'algorithme sur chacune des nouvelles cellules A et B jusqu'à terminaison, c'est-à-dire jusqu'à ce que la cellule contienne uniquement des points avec le même y ou plusieurs points avec les mêmes variables explicatives.
4. Pour chaque cellule terminale (feuille), on affecte le résultat majoritaire comme prédiction.

L'indice I utilisé dans l'algorithme est choisi parmi les indices ci-dessous :
— **Indice de Gini** : $I(p;1-p)=2p(1-p)$
— **Entropie** : $I(p;1-p)=plog_2(p)$
— **Greedy Misclassification error** : $I(p;1-p)=1-max(p,1-p)$

Le principe de construction d'un Random Forest est le suivant :
— on tire de l'échantillon initial (de taille n) B nouveaux échantillons avec replacement (**Bootstrap**). On dispose alors de B arbres de décision.
— Chaque arbre renvoie une réponse et l'ensemble des réponses est aggrégé par vote majoritaire (**Aggregating**).

Cette méthode permet d'augmenter la performance et de réduire considérablement le sur-apprentissage.

Le *Random Forest* unilabel peut facilement être étendu à la classification multilabel : de même que pour les modèles de régression logistique et XGBoost, on ne crée pas un seul modèle pour le problème de classification multilabel ; on crée un modèle pour chaque label en utilisant la stratégie One-vs-Rest (One-vs-All). La prédiction finale provient de l'union des prédictions pour chacun des labels. Supposons par exemple que nous souhaitions résoudre un problème de classification multilabel avec 10 labels. Avec cette approche, nous allons créer un modèle différent pour chaque label (donc 10 modèles au total). Pour prédire le genre d'un nouveau titre musical, nous donnerons ce titre comme entrée à tous les modèles et la prédiction proviendra de l'union des prédictions pour chaque label.

Nous présentons ci-après les différents modèles de *Random Forest* que nous avons implémentés et présentons les scores obtenus par les différentes méthodes dans le tableau 9.

4.4.2 Random Forest unilabel

Afin de nous familiariser avec le modèle, nous avons commencé par considérer chaque label de musique séparément en créant un *Random Forest* sur chacun des labels. Une fois les résultats obtenus pour chaque label, nous avons aggrégé, pour chaque titre, les résultats obtenus pour chacun des labels. De part sa construction, cette méthode ne prend pas en compte les corrélations entre les genres puisque chaque label est prédit indépendamment des autres. Cette méthode revient à utiliser la méthode *MultiOutputClassifier* fournit par *Sklearn*. Les résultats obtenus avec cette méthode "à la main" sont très proches de ceux obtenus avec la méthode *MultiOutputClassifier*.

4.4.3 Random Forest multilabels

Nous implémentons le modèle de *Random Forest Multilabel* avec la méthode *MultiOutputClassifier* décrite plus haut. De même que pour les modèles précédents, il est possible de choisir comme données d'entrée les *audio features* seules, les *usage features* seules ou bien de combiner les deux. Par ailleurs, l'algorithme *Random Forest* prend en entrée de nombreux paramètres qu'il est possible de choisir par défaut ou bien d'optimiser afin de rendre l'algorithme plus performant. Dans notre cas, il conviendrait d'optimiser les hyperparamètres de chacun des 21 *Random Forests* correspondant à chacun des labels. Parmi les 16 hyperparamètres optimisables, nous optimisons les hyperparamètres suivants :

— **"n-estimators"** : le nombre d'arbres du *Random Forest*.
— **"max-depth"** : la profondeur maximale des arbres aléatoires. La valeur par défaut pour *max-depth* est "None", ce qui signifie que chaque arbre se développera jusqu'à ce que chaque feuille soit pure. Une feuille pure est une feuille pour laquelle toutes les données de la feuille proviennent de la même classe.
— **"min-samples-split"** : le nombre minimum de points de données requis pour diviser un nœud interne. La valeur par défaut de ce paramètre est 2, ce qui signifie qu'un nœud interne doit avoir au moins deux points de données avant de pouvoir être divisé pour avoir une classification plus spécifique.
— **"min-samples-leaf"** : le nombre minimum de points de données requis pour constituer une feuille (les feuilles correspondant aux noeuds terminaux). La valeur par défaut de ce paramètre est 1, ce qui signifie que chaque feuille doit avoir au moins 1 point de données qu'elle classe.

Pour cela nous créons une grille qui contient plusieurs valeurs possibles pour chacun des hyperparamètres à optimiser et nous sélectionnons la combinaison qui maximise les métriques. La combinaison retenue pour le label pop est la suivante :

— nombre d'estimateurs : 4000
— profondeur maximale des arbres aléatoires : None
— nombre minimal de split : 2
— nombre minimal de feuilles : 1

Les paramètres obtenus après optimisation sont identiques aux paramètres par défaut, mis à part pour le nombre d'estimateurs qui était de 100 par défaut. Cette première optimisation sur les hyperparamètres du label Pop n'a pas permis d'améliorer sensiblement les résultats (tableau 8) et a demandé un temps de calcul important (plus de 6 heures). Nous avons donc choisi de conserver les paramètres par défaut du *Random Forest*.

Méthode	*Accuracy*	*Précision*	*Recall*	*F1 score*
RF **non optimisé** audio and usage features	0.93	0.87	0.56	0.69
RF **optimisé** audio and usage features	0.93	0.88	0.57	0.70

TABLE 8 – Performances obtenues avec ou sans optimisation des hyperparamètres pour le label Pop
Plus l'accuracy, le recall, la precision et le F1 score sont proches de 1 et plus le modèle est performant.

Les résultats du *Random Forest Multioutput* sont donnés tableau 9. Les meilleurs résultats sont obtenus en ne prenant en entrée que les *usage features*. Les résultats sont meilleurs

que pour le meilleur des modèles naïfs. En ne considérant que les *audio features*, les résultats sont sensiblement meilleurs que pour le modèle *Average-by-genre* avec les audio features seules pour la précision et la *Hamming-Loss*.

Dans un deuxième temps, nous implémentons un modèle de Random Forest Multilabel en chainant les régressions des différents labels : il s'agit toujours de créer un modèle différent pour chaque label, mais ces régressions sont effectuées dans un certain ordre (défini aléatoirement) et prennent en compte, en les intégrant aux features, les résultats des régressions précédentes. Cette méthode permet de prendre en compte les éventuelles corrélations entre les différents labels. Cependant, cettte méthode n'améliore pas les résultats obtenus avec le *Multioutput* simple.

Nous présentons en annexe des résultats de performance détaillés à l'échelle des labels. Les histogrammes des AUC (figure 14) montrent des différences de performances entre les genres. Globalement, les AUC sont meilleures en prenant en entrée les *usage features* seules et les *audio et les usage features* ensembles. On remarque cependant que certains genres (Classique ou Métal par exemple) performent déjà très bien avec les *audio features* seules. Les ROC curves des genres Classique et Blues (figure 15 toujours en annexe) confirment ces différences entre genres.

4.4.4 Résultats Random Forest

Les résultats (tableau 9) pour les méthodes de *Random Forest* implémentées dans cette partie montrent encore une fois que l'utilisation des usage features produit de meilleurs résultats que les audio features : le modèle le plus performant semble être le *Random Forest* avec la méthode *Multioutput* en prenant en entrée les usage features seules. L'utilisation de l'option *chained*, bien que permettant de prendre en compte d'éventuelles corrélations entre genres, n'améliore pas sensiblement les résultats.

Méthode	*Accuracy*	*Précision micro*	*Recall micro*	*F1 score*	*Hamming-Loss*
RF Multioutput audio and usage features	0.740	**0.923**	0.772	0.841	0.016
RF Chained audio and usage features	0.742	**0.923**	0.772	0.841	0.016
RF Multioutput usage features	0.774	0.921	**0.811**	**0.862**	**0.014**
RF Chained usage features	**0.777**	0.919	0.810	0.861	0.015
RF Multioutput audio features	0.317	0.804	0.350	0.488	0.041
RF Chained audio features	0.328	0.800	0.357	0.494	0.041

TABLE 9 – Résultats obtenus pour les différentes métriques par les méthodes de Random Forest implémentées
Plus l'accuracy, le recall, la precision et le F1 score sont proches de 1 et plus le modèle est performant. Plus la Hamming-Loss est proche de 0 et plus le modèle est performant. Pour chaque métrique, le résultat en gras correspond au modèle le plus performant pour cette métrique.

4.5 Réseaux de neurones

4.5.1 Principe de l'apprentissage profond (*deep learning*)

La dernière approche que nous avons explorée au sein de notre étude pour accomplir la tâche de classification multilabel des genres associés à un titre musical est celle de **l'apprentissage profond** (*deep learning*). Pour définir l'apprentissage profond et comprendre la différence entre l'apprentissage profond et les autres approches d'apprentissage automatique précédemment évoquées, nous devons d'abord décrire la manière dont fonctionnent les algorithmes d'apprentissage automatique. Comme le suggère les différentes

méthodes présentées au sein de ce mémoire, le principe de l'apprentissage automatique est de découvrir des règles permettant d'exécuter une tâche de traitement de données lorsque lui sont fournis des exemples de résultats attendus. Pour l'apprentissage automatique, nous avons donc besoin de trois choses :

1. Des données d'entrées — dans notre cas, pour la classification de genres musicaux, les entrées utilisées sont des vecteurs de type *audio* et *usage*.
2. Des exemples de sortie attendue — dans le cas de notre étude qui est une tâche d'étiquetage de musiques, les sorties attendues sont des étiquettes de genre telles que « Asian », « Rock », etc.
3. Un moyen de mesurer la performance de l'algorithme — c'est un élément nécessaire pour déterminer la distance, au sens mathématique, entre la sortie effective de l'algorithme et la sortie attendue ; la mesure est utilisée pour ajuster le fonctionnement de l'algorithme ; cette étape d'ajustement est appelée l'apprentissage.

Un modèle d'apprentissage automatique transforme ses données d'entrée en sorties qui ont un sens, c'est un processus qui est « appris » à partir de l'exposition à des exemples connus d'entrées et de sorties. Par conséquent, le problème central de l'apprentissage automatique et donc de l'apprentissage profond est de transformer de manière utile les données : en d'autres termes, d'apprendre des représentations utiles des données d'entrée disponibles qui approchent le résultat attendu.

L'apprentissage profond est par définition un sous-domaine spécifique de l'apprentissage automatique : c'est une nouvelle approche de l'apprentissage des représentations à partir des données qui met l'accent sur l'apprentissage de couches successives de représentations qui sont de plus en plus significatives. En apprentissage profond, ces représentations en couches sont apprises au moyen de modèles appelés réseaux de neurones, structurés en couches littéralement superposées les unes sur les autres.

La spécification de ce que fait une couche sur ses données d'entrée est stockée dans les poids de la couche qui sont en substance un ensemble de nombres. Dans ce contexte, apprendre signifie rechercher un ensemble de valeurs pour les poids de toutes les couches dans un réseau, de sorte que le réseau met correctement en correspondance les exemples d'entrées avec les étiquettes cibles qui leur sont associées.

Pour contrôler la sortie d'un réseau de neurones, il faut pouvoir mesurer dans quelle mesure cette sortie est différente de celle attendue. C'est le travail de la fonction de perte du réseau. La fonction de perte prend les prédictions du réseau et la cible réellement attendue et calcule un score de distance. L'astuce fondamentale de l'apprentissage profond est alors d'utiliser ce score comme un signal de retour pour ajuster légèrement la valeur des poids dans un sens qui réduira le score de perte. Cet ajustement est le travail de l'optimiseur qui met en œuvre ce que l'on appelle l'algorithme de rétropropagation.

4.5.2 Les différents modèles implémentés

Dans le cas de notre étude, nous avons implémenté 9 modèles de réseaux de neurones assez simples, qui diffèrent par leurs structures, mais aussi par le type de données traitées en entrée. La figure 12 présente les différentes architectures utilisées tandis que le tableau 10 présente les spécificités de chacun des modèles implémentés :

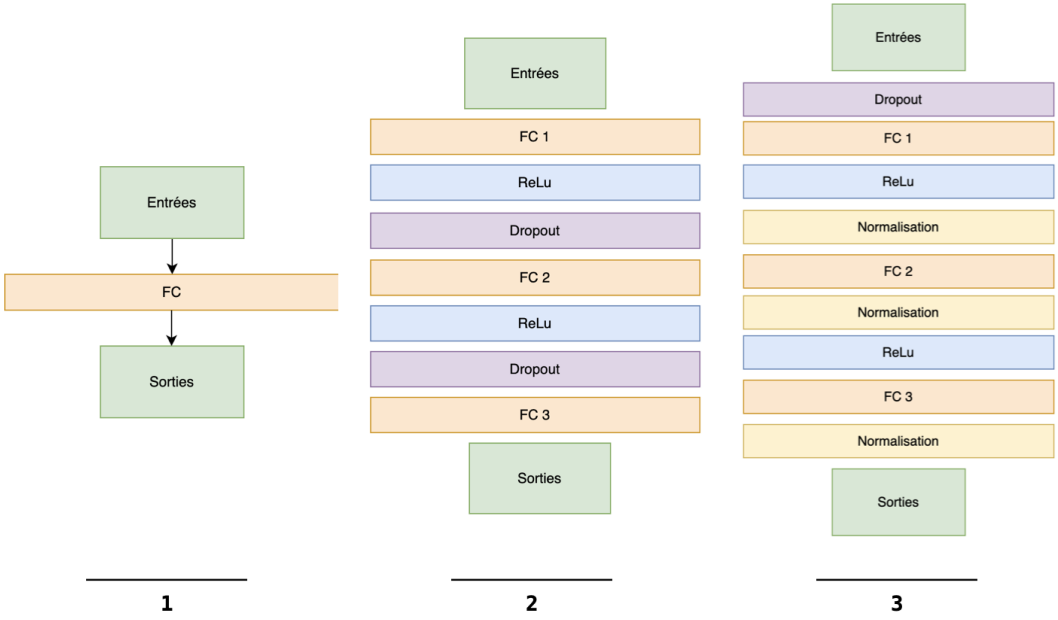


FIGURE 12 – Les différentes architectures utilisées

Modèle	Données en entrée	Architecture	Learning rate	Optimiseur	Fonction de perte
AudioNet1	audio	1	0.0001	Adam	Binary Cross Entropy
AudioNet2	audio	2	0.0001	Adam	Binary Cross Entropy
AudioNet3	audio	3	0.0001	Adam	Binary Cross Entropy
UsageNet1	usage	1	0.0001	Adam	Binary Cross Entropy
UsageNet2	usage	2	0.0001	Adam	Binary Cross Entropy
UsageNet3	usage	3	0.0001	Adam	Binary Cross Entropy
MixNet1	audio + usage	1	0.0001	Adam	Binary Cross Entropy
MixNet2	audio + usage	2	0.0001	Adam	Binary Cross Entropy
MixNet3	audio + usage	3	0.0001	Adam	Binary Cross Entropy

TABLE 10 – Les différents modèles implémentés, leurs structures et paramètres importants
Nous avons fait le choix de tester le modèle du plus simple au plus sophistiqué. Le learning rate ainsi que le niveau de dropout pour chaque modèle ont été déterminés en réalisant une gridsearch. Ces modèles ont été entraînés sur 100 époques. La figure 16 en annexe montre l'évolution de la Binary Cross Entropy loss sur les 100 époques pour le modèle MixNet2 par exemple.

Les architecture présentées en 12 présentent différents type de couches :

- **FC** (*fully connected*) : dans les couches entièrement connectées, le neurone applique une transformation linéaire au vecteur d'entrée à travers une matrice de poids.
- **ReLU** (*Rectified Linear Units*) désigne la fonction réelle non-linéaire définie par $ReLU(x) = \max(0, x)$. La couche de correction ReLU remplace donc toutes les valeurs négatives reçues en entrées par des zéros. Elle joue le rôle de fonction d'activation.
- **Dropout** : désactive temporairement certains neurones dans le réseau, ainsi que toutes ses connexions entrantes et sortantes. Le choix des neurones à désactiver est aléatoire. Nous attribuons donc une probabilité p à tous les neurones (choisies optimalement grâce à une méthode de validation croisée) qui détermine leur activation et à chaque époque, cette désactivation aléatoire est appliquée.

— **Normalisation** : normalise les valeurs du vecteur d'entrée.

L'optimiseur **Adam** est quant à lui une méthode de descente en gradient stochastique basée sur l'estimation adaptative des moments de premier et de second ordre, tandis que la fonction de perte **Binary Cross Entropy** est définie de la manière suivante :

$$BCE = -\frac{1}{\text{output size}} \sum_{i=1}^{\text{output size}} y_i \log \hat{y}_i + (1 - y_i) \log(1 - \hat{y}_i)$$

avec $\hat{y}_i$ la i-ème valeur du vecteur de sortie du modèle, y_i la valeur cible correspondante, et *output size* le nombre de valeurs dans le vecteur de sortie du modèle (soit 21 dans notre cas).

4.5.3 Les résultats obtenus

Nous obtenons les résultats suivants après entraînement des différents modèles (tableau 11) :

Modèle	*Accuracy*	*Precision micro*	*Recall micro*	*F1-score micro*	*Haming loss*
AudioNet1	0.49	0.644	0.504	0.565	0.048
AudioNet2	0.52	0.693	0.543	0.609	0.043
AudioNet3	0.61	0.773	0.612	0.665	0.038
UsageNet1	0.64	0.784	0.620	0.693	0.034
UsageNet2	0.69	0.874	0.698	0.776	0.025
UsageNet3	0.75	0.889	0.728	0.782	0.021
MixNet1	0.68	0.880	0.682	0.769	0.025
MixNet2	0.71	0.882	0.698	0.779	0.024
MixNet3	**0.78**	**0.921**	**0.803**	**0.857**	**0.017**

TABLE 11 – Résultats obtenus pour les différentes métriques selon les modèles de réseaux de neurones utilisés
Plus l'accuracy, le recall, la precision et le F1 score sont proches de 1 et plus le modèle est performant. Plus la Hamming-Loss est proche de 0 et plus le modèle est performant. Pour chaque métrique, le résultat en gras correspond au modèle le plus performant pour cette métrique. Le modèle le plus performant est le MixNet3 qui prend en compte à la fois les audio et les usage features.

Le meilleur modèle obtenu est donc le modèle **MixNet3**. Les résultats (tableau 11) nous montrent également que l'utilisation des *usage features* augmente la performance des modèles.

5 Conclusion

Partant de modèles naïfs pour aller vers des modèles plus évolués, nous avons construit plusieurs modèles permettant d'associer à un titre musical ses labels de genre. Il est maintenant intéressant de déterminer le modèle qui semble le plus adapté pour la prédiction des genres de la base.

Nous récapitulons ici les performances obtenus pour les modèles naïfs testés en début d'analyse (12) ainsi que les modèles plus élaborés testés ensuite (13) en ne gardant que la version de chaque modèle avec la meilleure performance. Nous indiquons en gras, pour chaque métrique, la meilleure performance obtenue.

Méthode	*Accuracy*	*Précision micro*	*Recall micro*	*F1 score*	*Hamming-Loss*
Classifier randomisé	0.047	0.047	0.047	0.047	0.950
Top genre	0.108	0.162	0.138	0.149	0.088
Average-by-genre	**0.795**	**0.795**	**0.795**	**0.795**	**0.019**

TABLE 12 – Résultats obtenus pour les différentes métriques avec les méthodes naïves
Plus l'accuracy, le recall, la precision et le F1 score sont proches de 1 et plus le modèle est performant. Plus la Hamming-Loss est proche de 0 et plus le modèle est performant. Pour chaque métrique, le résultat en gras correspond au modèle le plus performant pour cette métrique.

Méthode	*Accuracy*	*Précision micro*	*Recall micro*	*F1 score*	*Hamming-Loss*
Régression Logistique	0.720	0.870	0.797	0.832	0.018
XGBoost	0.749	0.904	0.799	0.849	0.016
Random Forest	**0.774**	**0.921**	**0.811**	**0.862**	**0.014**
Réseau de neurones	0.772	**0.921**	0.803	0.857	0.017

TABLE 13 – Résultats obtenus pour les différentes métriques avec les modèles les plus performants de chaque méthode testée
Plus l'accuracy, le recall, la precision et le F1 score sont proches de 1 et plus le modèle est performant. Plus la Hamming-Loss est proche de 0 et plus le modèle est performant. Pour chaque métrique, le résultat en gras correspond au modèle le plus performant pour cette métrique.

Le modèle de *Random Forest* semble être le modèle le plus performant globalement : il obtient les meilleures performances pour toutes les métriques. Ce modèle de *Random Forest* prend en entrée les *usage features* seules. On arriverait à des conclusions différentes en choisissant le meilleur modèle ne prenant en entrée que les *audio features* : en effet, c'est alors le réseau de neurones qui surpasse de loin les autres modèles en performances, notamment pour l'*accuracy*, le *recall* et le *F1-score*. Comme nous avons pu l'observer en se penchant sur les performances de chaque genre, les différents genres musicaux ne se comportent pas de la même manière suivant les *features* prises en entrée : certains genres performent très bien avec les *audio features* seules, quand d'autres (une majorité) performent mieux en ajoutant les *usage features*. Essayer de comprendre d'où viennent ces différences pourrait constituer une suite à notre projet.

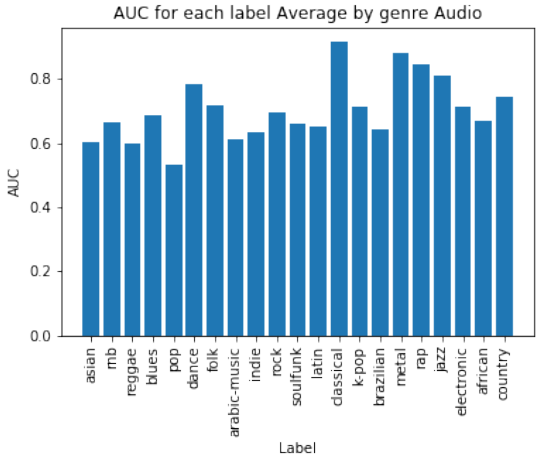
Aussi, nous pouvons noter que l'algorithme qui performe le mieux (*Random Forest*) est un algorithme qui ne prend pas en compte le caractère multilabel de nos données. C'est aussi le cas pour le XGBoost dont les perfomances sont relativement proches. Ceci est probablement dû au fait que nous ayons plus de 80% de données unilabel. Une piste de poursuite envisagée serait donc de construire un dataset plus équilibré afin de pouvoir vérifier que les réseaux de neurones par exemple ne performent pas mieux.

6 Annexe

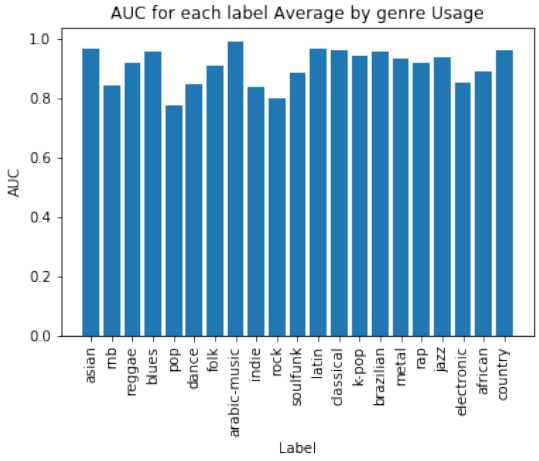
6.1 Lien du repository GitHub

L'ensemble du code nous ayant permis de produire les résultats de notre étude est disponible dans le *repository* GitHub suivant : https://github.com/taminemelissa/multilabel-classification

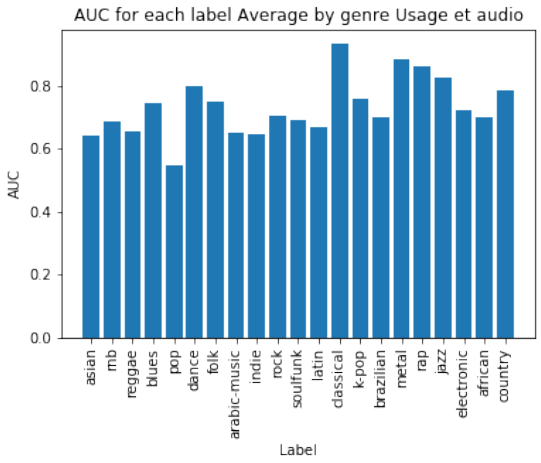
6.2 Average-by-genre, détail des performances par label



(a) Audio features



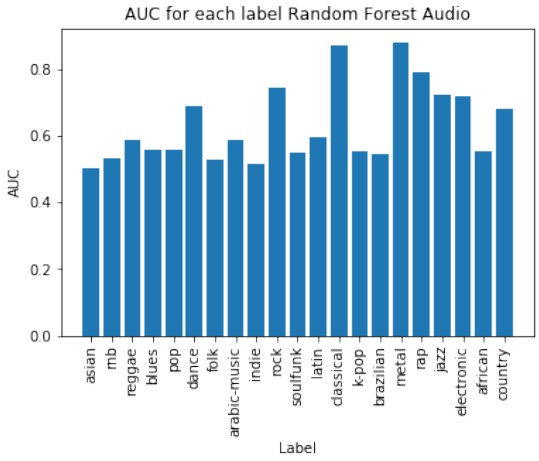
(b) Usage features



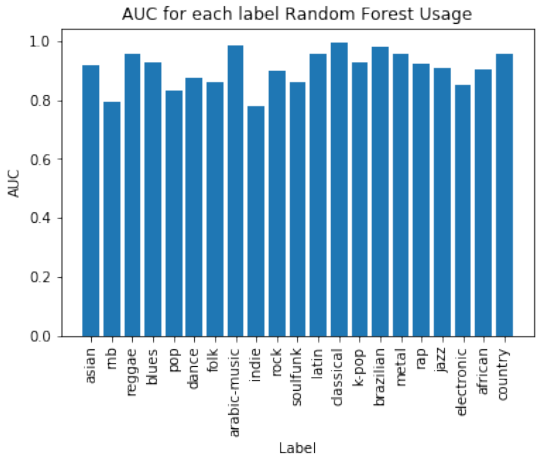
(c) Usage and audio features

FIGURE 13 – Histogrammes des AUC par genre pour le modèle Average-by-genre suivant les features prises en entrée.
On observe de bien meilleures AUC avec les usage features seules. Les résultats diffèrent suivant les genres : le Classique obtient ainsi de bonnes performances pour les trois modèles.

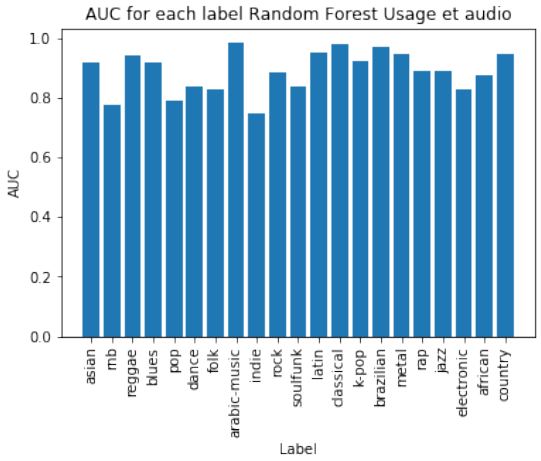
6.3 Random Forest, détail des performances par label



(a) Audio features

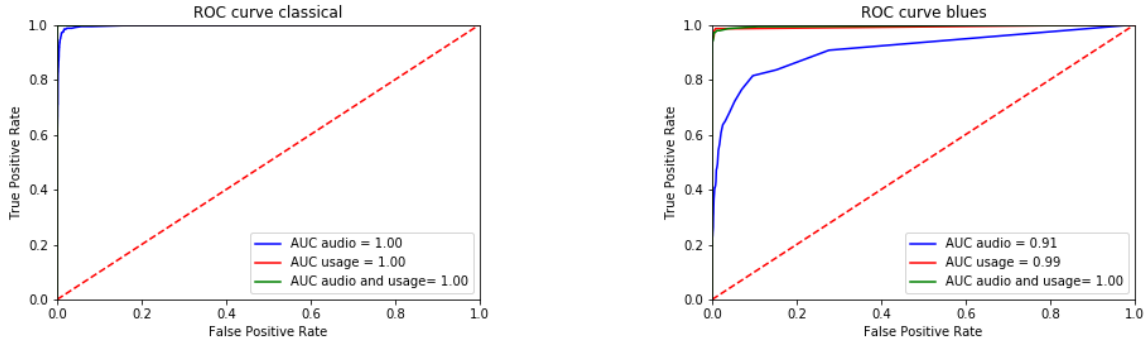


(b) Usage features



(c) Usage and audio features

FIGURE 14 – Histogrammes des AUC par genre pour le modèle Random Forest Multioutput suivant les features prises en entrée.
Les AUC avec les audio features seules sont bin moins bonnes que pour les AUC avec usage featurs seules ou audio et usage features. Les résultats diffèrent suivant les genres : le Classique performe ainsi très bien avec les audio features seules. Contrairement à l'Average-by-genre, le modèle avec audio et usage features performe très bien aussi pour la plupart des genres.



(a) Genre Classique (b) Genre Blues

FIGURE 15 – ROC curve pour le Classique et le Blues avec le modèle Random Forest Multioutput.
Sont présentées les performances suivant les features prises en entrée (courbes de différentes couleurs). De même que pour le modèle Average-by-genre, on observe que le genre Classique est très bien prédit pour toutes les features d'entrée. A contrario, les performances pour le Blues avec les audio features seules sont plus faibles.

6.4 Exemple de tracé de l'évolution de la Binary Cross Entropy loss

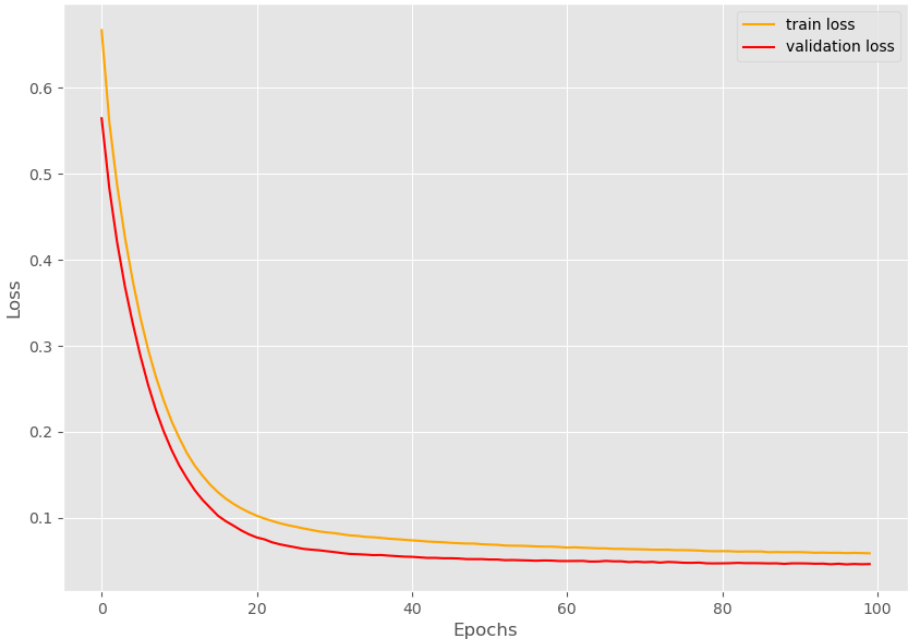


FIGURE 16 – Évolution de la Binary Cross Entropy loss sur 100 époques pour le modèle de réseau de neurones MixNet2

Références

[Choi *et al.*, 2016] Choi, K., Fazekas, G. et Sandler, M. B. (2016). Automatic tagging using deep convolutional neural networks. *ArXiv*, abs/1606.00298.

[Koren *et al.*, 2009] Koren, Y., Bell, R. et Volinsky, C. (2009). Matrix factorization techniques for recommender systems. *Computer*, 42(8):30–37.

[McKay et Fujinaga, 2006] McKay, C. et Fujinaga, I. (2006). Musical genre classification : Is it worth pursuing and how can it be improved ?

[Oramas *et al.*, 2017] Oramas, S., Nieto, O., Barbieri, F. et Serra, X. (2017). Multi-label music genre classification from audio, text, and images using deep features.

[Tsoumakas et Katakis, 2009] Tsoumakas, G. et Katakis, I. (2009). Multi-label classification : An overview. *International Journal of Data Warehousing and Mining*, 3:1–13.