BORDEAUX INP Ensc

Rapport intermédiaire
Projet IA

LE CHENADEC Elouarn
ARNAUD Axel
3A

Année universitaire 2024-2025

Présentation du sujet

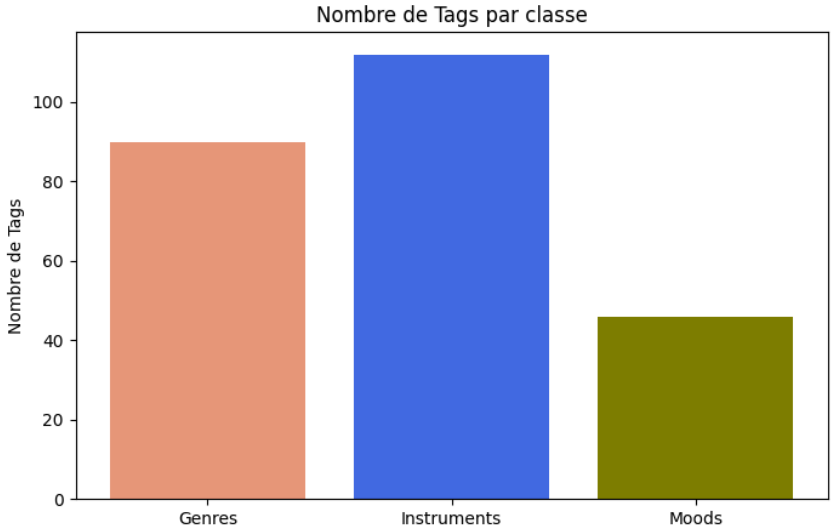
Mewo est une plateforme de gestion de catalogues pour les bibliothèques de musique de production utilisée par de grands acteurs français des médias, de la publicité et de la musique. Mewo ont déjà un système de labellisation automatique qui prédit des scores numériques indiquant la pertinence d'un morceau pour différents tags qui sont organisés en trois grandes catégories : genres, instruments, humeurs ("mood"). Cependant, pour exploiter ces prédictions, il est nécessaire de les convertir en décisions binaires : associer ou non un tag à une piste.

L'approche présentée dans le challenge utilise un seuil par tag pour maximiser un F1-score individuel. Bien que performante, cette méthode ne prend pas en compte les relations entre tags, catégories ou classes. Bien sûr le challenge date de 2021 et leur processus de labellisation a certainement un meilleur benchmark aujourd'hui. L'objectif donné par le benchmark est un weighted F1-Score de **54%** sur le **train set** et **46%** sur le **test set**.

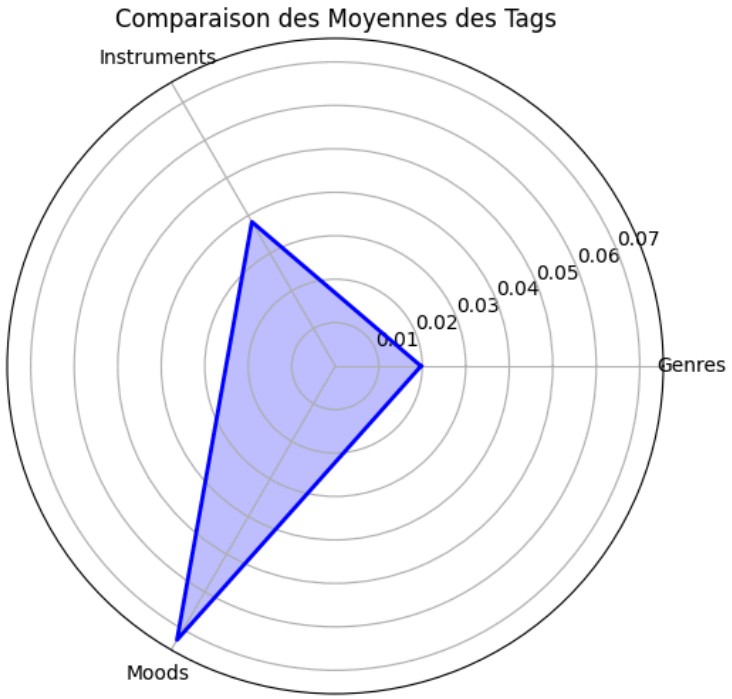
Le challenge consiste donc à concevoir une méthode plus avancée pour améliorer la qualité globale de la classification des labellisations.

Présentation du dataset

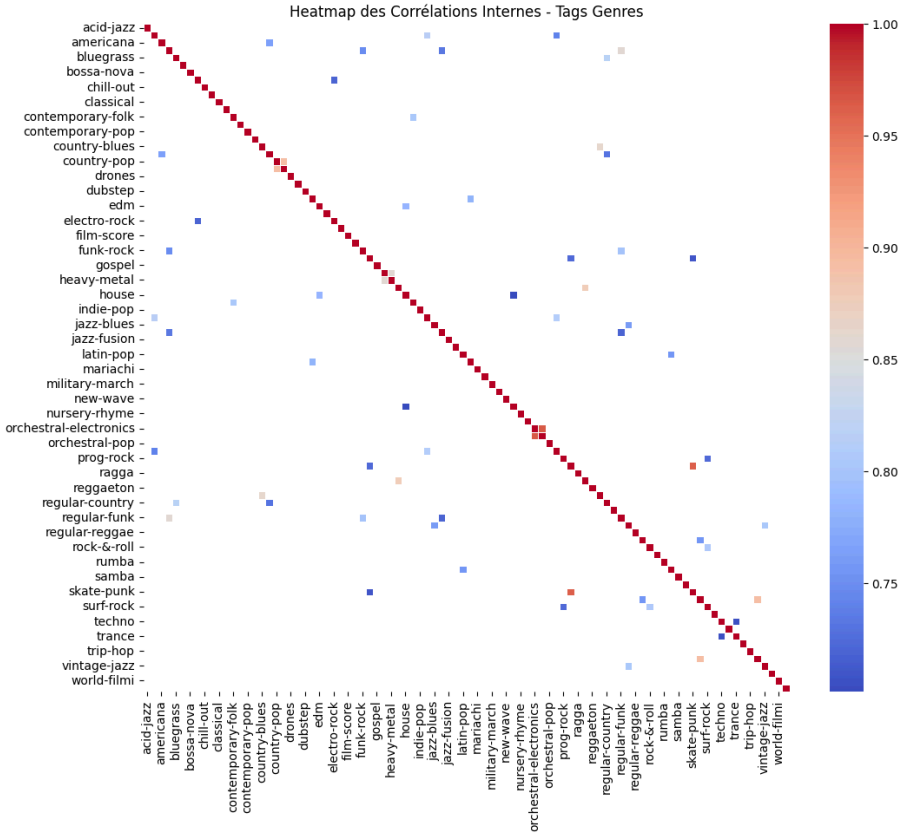
Nous avons à disposition un dataset important avec plus de **110 000** inputs, avec **248** features. Ces features décrivent la probabilité pour un tag de correspondre à une musique. Ces tags sont divisés en trois grandes classes : **instrument**, **genre** et **mood**.

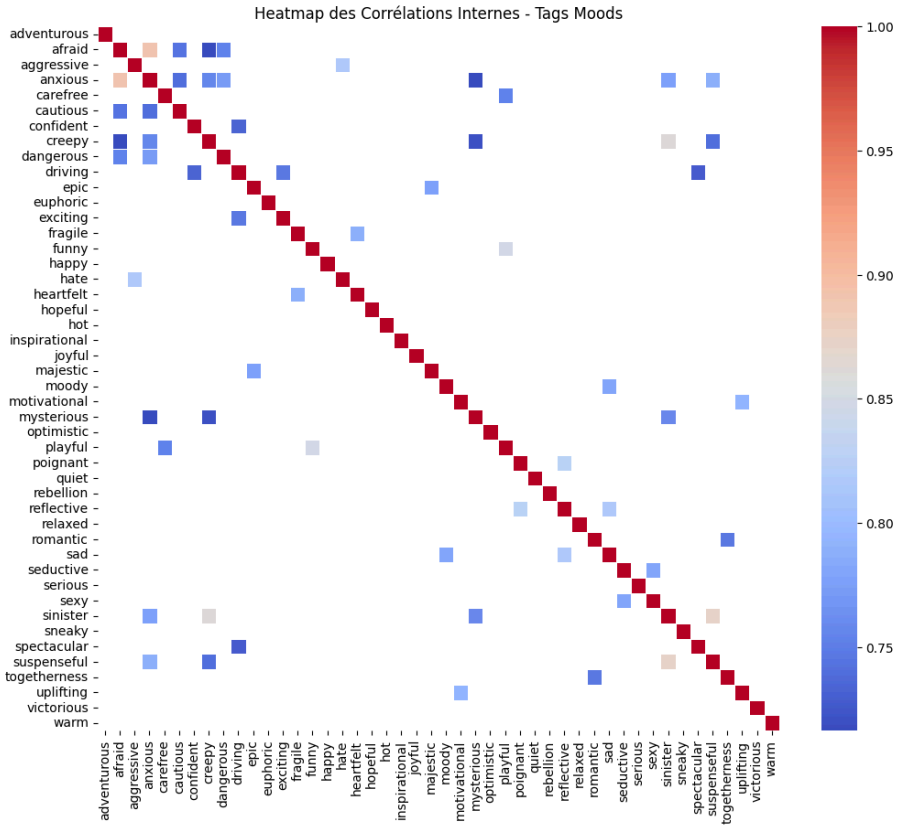


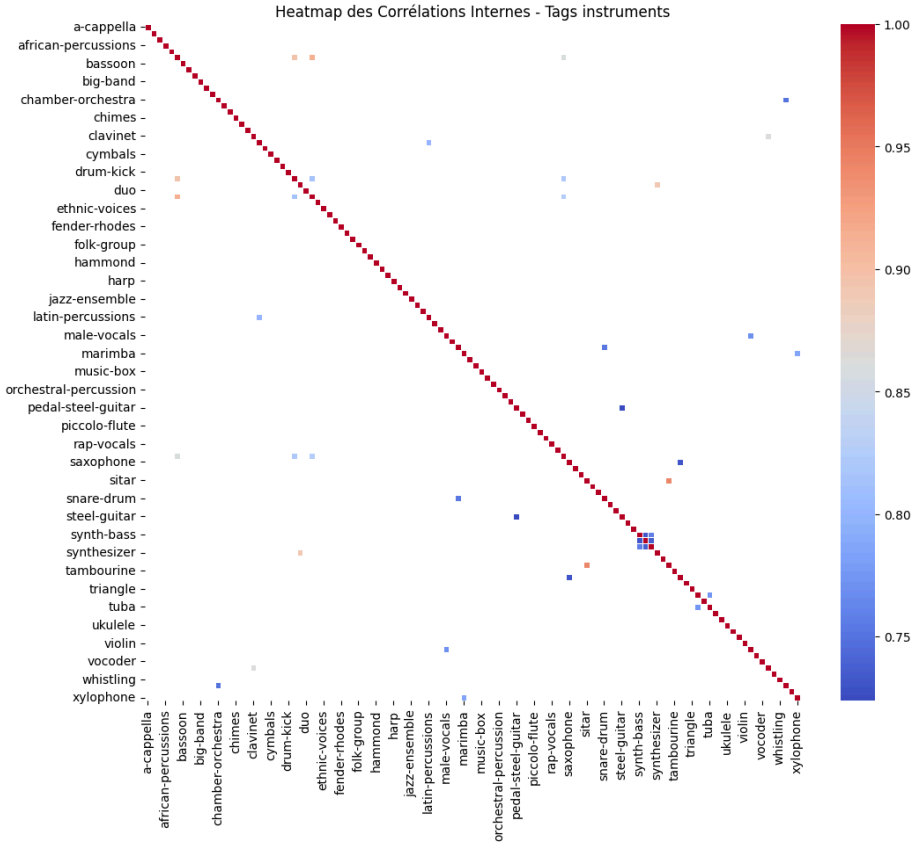
Les données ne semblent pas normalisées.

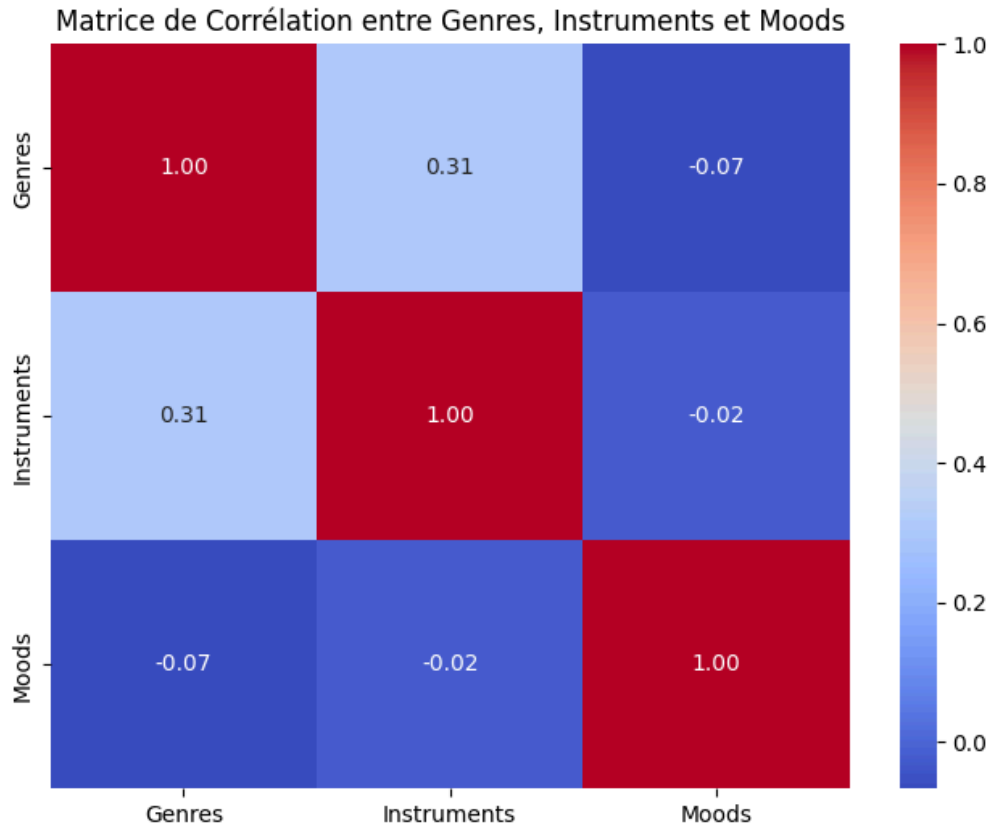


Au niveau de la corrélation dans le jeu de données, on a ressorti quatre graphiques montrant les corrélations inter-classes et intra-classes :




Heatmap des Corrélations Internes - Tags Moods
adventurous
afraid
aggressive
anxious
carefree
cautious
confident
creepy
dangerous
driving
epic
euphoric
exciting
fragile
funny
happy
hate
heartfelt
hopeful
hot
inspirational
joyful
majestic
moody
motivational
mysterious
optimistic
playful
poignant
quiet
rebellion
reflective
relaxed
romantic
sad
seductive
serious
sexy
sinister
sneaky
spectacular
suspenseful
togetherness
uplifting
victorious
warm
1.00
0.95
0.90
0.85
0.80
0.75


Heatmap des Corrélations Internes - Tags instruments
a-cappella
african-percussions
bassoon
big-band
chamber-orchestra
chimes
clavinet
cymbals
drum-kick
duo
ethnic-voices
fender-rhodes
folk-group
hammond
harp
jazz-ensemble
latin-percussions
male-vocals
marimba
music-box
orchestral-percussion
pedal-steel-guitar
piccolo-flute
rap-vocals
saxophone
sitar
snare-drum
steel-guitar
synth-bass
synthesizer
tambourine
triangle
tuba
ukulele
violin
vocoder
whistling
xylophone
1.00
0.95
0.90
0.85
0.80
0.75



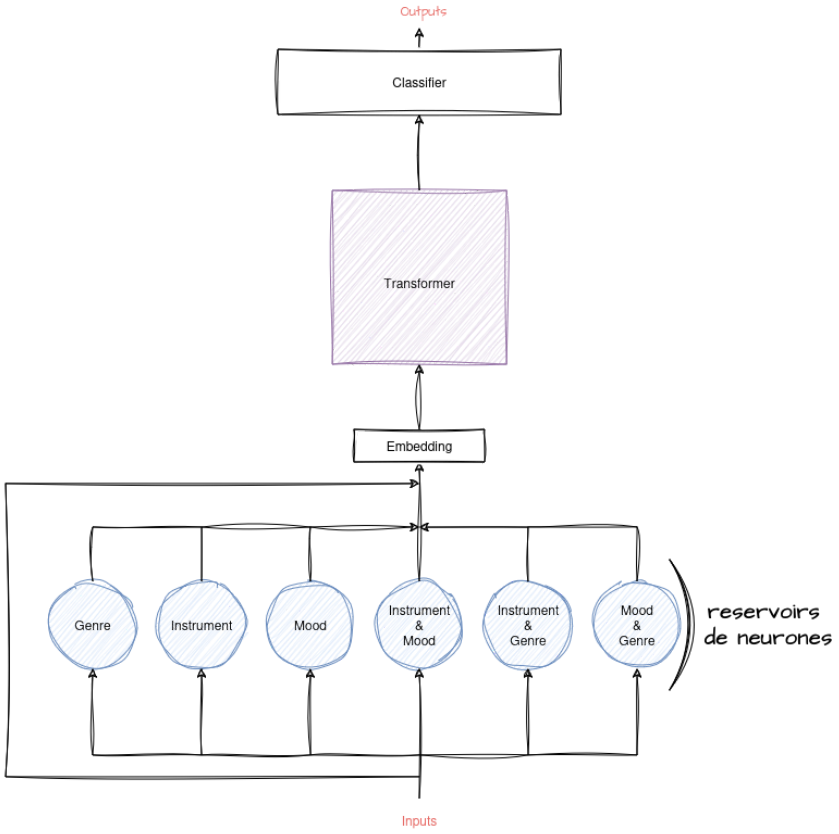
On remarque à première vue que les features sont très peu corrélées entre elles.

Environnement technique

Langage de programmation : python
Dépendances utilisées :

- pytorch
- reservoir.py
- pandas
- numpy
- scikit-learn
- matplotlib

Architecture du réseau neuronal

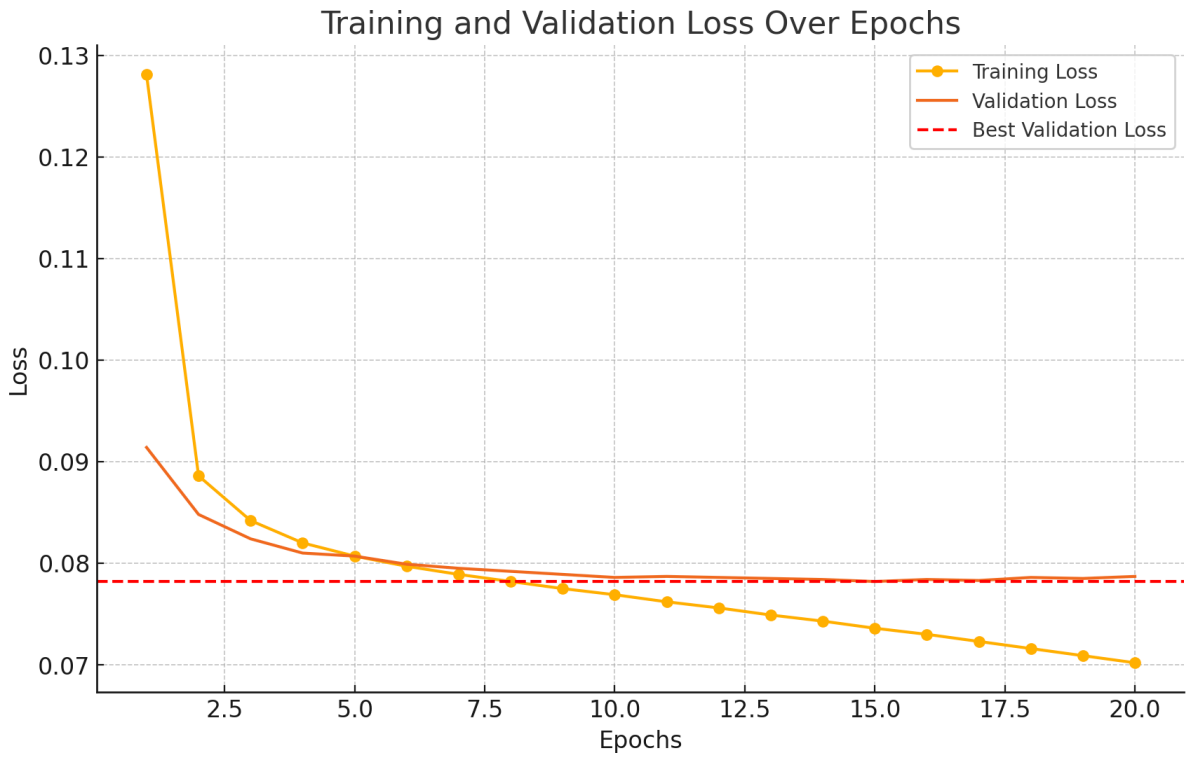


Pourquoi cette structure ?

On a un dataset très volumineux : plus de **110 000** inputs qui ont chacun **248** features.
En premier test on a fait passer les données dans un réseau de neurones par couches classiques, et malgré de nombreux essais en fine tuning, aucun résultat ne serait-ce qu'un peu probant ne ressortait, le modèle n'arrivait pas à traiter le problème et était moins efficace qu'un simple seuil à 0.5. On a pû en déduire qu'un réseau de neurones par couches "basique" ne serait pas efficace pour traiter ce problème de classification. On a donc cherché une autre solution, et on a particulièrement voulu expérimenter, d'où l'originalité de notre structure.
Pourquoi utiliser des réservoirs de neurones ? Les réservoirs de neurones sont communément utilisés dans des problèmes traitant des séries numériques (notamment prédiction), mais la particularité qui nous a intéressés chez ces réservoirs est que ceux-ci "enrichissent" la data et exacerbent les corrélations dans un jeu de données par la non linéarité de sa structure.
On a donc implémenté 6 réservoirs de neurones, chacun traitant différentes parties du dataset parallèlement. Ils sont d'ailleurs paramétrés un peu "spécialement" sur conseil de Xavier Hinaut. En effet les réservoirs sont utilisés sans récurrence (spectral radius à 0) et en enlevant le "leaking" des neurones (leak-rate à 1), on utilise ainsi une forme spécifique : les **extreme learning machine**.

Pourquoi un transformer ? Les transformers peuvent capturer des dépendances à long terme et des corrélations entre les différentes classes et catégories, notre problème est un problème à haute dimension (248 features) domaine dans lequel les transformers excellent. Le transformer est d'ailleurs très efficace pour de la classification.
On entraîne donc notre modèle en deux temps : d'abord les couches de sortie (read-out) des réservoirs, puis le transformer.

Nos résultats pour l'instant



Pour l'instant, l'utilisation de réservoirs n'améliore pas encore les résultats. On a des performances légèrement meilleures avec le transformer seul. Cependant, nous avons une erreur dans le code au moment de l'embedding : on concatène toutes les sorties des réservoirs ensemble pour les envoyer en entrée au transformer. Hors, pour tirer partie de la pleine puissance du transformer, il serait plus judicieux de faire les embeddings séparément et donc d'avoir plusieurs entrées (une par réservoir). C'est donc ce que nous voulons faire pour que l'utilisation des réservoirs puisse éventuellement améliorer la performance du modèle.
Résultat sur le dataset de test :

```
Accuracy: 0.9716728914987757
              precision    recall  f1-score   support

   micro avg     0.6807    0.4167    0.5170    200012
   macro avg     0.6061    0.2760    0.3557    200012
weighted avg     0.6529    0.4167    0.4885    200012
 samples avg     0.6686    0.4169    0.4918    200012
```

Ces résultats sont meilleurs que ceux du classement du challenge mais le score est fait sur une partie du x_train qu'on a séparé en amont de l'entraînement du reste et qui n'a donc pas servi à entraîner le modèle. Les résultats sont tout de même prometteurs.

Ce qu'il reste à faire

Il reste plusieurs points sur lesquels travailler :

- En entrée du transformer, on a pour l'instant concaténé ensemble les sorties des réservoirs, hors pour tirer partie de la puissance du transformer et de sa multi-head attention, il faudrait plutôt envoyer en entrées chaque sortie des réservoirs séparément.
- Il y a une petite partie des features qu'on a pas encore utilisé : quelques features donnant des probabilités de "catégorie" de musique. On pourrait utiliser un réservoir à part pour traiter ces features ou tout simplement les envoyer directement comme entrée du transformer.
- En parlant de réservoir, on voudrait essayer d'en ajouter un traitant toutes les features en parallèle des autres réservoirs, et étudier l'efficacité d'un tel ajout.
- Le y_test que nous avons à disposition sont les prédictions par Mewo, donc on doit comparer nos prédictions avec leurs prédictions (qui ne sont pas parfaites) et il nous faudrait un y_test représentatif des vraies caractéristiques de chaque musique . On ne sait pas si vous l'avez, si il faut plutôt demander à l'entreprise directement ou si on peut se contenter de comparer nos prédictions avec ce y_test.