Hit ou Flop ? Analyse Prédictive par Machine Learning et Deep Learning

De-yed Abraham

Juillet 2024

Abstract

Dans cette étude, je vérifie la possibilité de prédire les succès musicaux en combinant des techniques de machine learning et de deep learning. J'ai créé un ensemble de données original incluant des morceaux populaires et non populaires, permettant d'extraire des caractéristiques musicales via des analyses audio classiques et des réseaux de neurones convolutifs (CNN) appliqués à des spectrogrammes de Mel et CQT. Après un prétraitement minutieux, plusieurs modèles, dont la régression logistique, Random Forest, Gradient Boosting, et CNN, ont été évalués. Les résultats montrent qu'intégrer les caractéristiques des CNN avec celles des analyses audio de base dans un modèle de Gradient Boosting améliore la précision des prédictions à 60 %. Ce modèle, fondé sur un ensemble de données unique, pourrait aider les artistes à optimiser leurs créations et améliorer les systèmes de recommandation. eMalgré ces résultats prometteurs, des défis subsistent, tels que la labellisation des données et la recherche de caractéristiques discriminantes. Ce travail constitue une avancée dans la prédiction du succès musical et suggère des pistes pour de futures recherches.

1 Introduction

"Sir Coxone was a smart guy and he had the ears for music, good music", "He might not be an instrument player, he might not even know if the guitar is tuned, but he knows when the song is RIGHT!" affirmaient Lloyd Mc Donald et Neville Livingstone en décrivant Clement "Coxsone" Dodd, producteur et fondateur de Studio One, dans un documentaire de Kevin MacDonald sur Bob Marley [9]. Ces remarques ont profondément inspiré l'étude que je vous présente.

Depuis mon enfance, la musique a occupé une place centrale dans ma vie, influencée par mon héritage familial de musiciens ghanéens du côté de ma mère. Mes parents, soucieux de perpétuer cette tradition, m'ont encouragé à apprendre le piano. Cette immersion dans le monde de la musique m'a amené à m'interroger sur un phénomène : le succès musical. J'ai souvent entendu des artistes évoquer la répétition de certains schémas dans leurs morceaux à succès, des motifs qui semblent revenir de manière presque instinctive ou voulue. Certains pourraient trouver cette idée absurde, mais il est indéniable que d'autres comme Coxsone Dodd, ont un talent inné pour reconnaître les chansons qui cartonneraient. Peut-être, comme Coxsone, avons-nous tous cette capacité, consciente ou pas, à identifier ces schémas récurrents dans les hits.

Des morceaux intemporels tels que "**Beat It**" de Michael Jackson ou "**Gangnam Style**" de Psy semblent transcender les barrières culturelles et temporelles, suggérant qu'il pourrait y avoir des éléments communs à tous ces succès. Cela m'amène à une question essentielle : existe-t-il réellement des schémas musicaux récurrents dans les hits, indépendamment du style, de l'époque, des cultures ?

Pour répondre à cette hypothèse, j'ai choisi d'explorer l'univers fascinant de l'apprentissage automatique. En utilisant des modèles de machine learning, je tenterai de détecter ces schémas dans les

caractéristiques musicales des morceaux. Si de tels schémas existent, nos modèles devraient afficher une précision prédictive élevée. Bien que d'autres facteurs, comme les tendances, la publicité, ou la démographie de l'audience, (ce que l'on appelle "external perspective" [17]) jouent un rôle indéniable dans le succès musical, cette étude se concentrera exclusivement sur l'aspect musical. Après tout, une chanson doit avant tout plaire au public pour devenir un hit.

Prédire le succès musical relève d'un domaine appelé Hits Songs Science (HSS), une sous-branche de la Music Information Retrieval (MIR), qui se consacre à l'analyse et à la récupération d'informations à partir de la musique. Bien que peu nombreuses, des études de prédiction ont déjà été menées sur ce sujet, souvent en se basant uniquement sur des données de plateformes comme YouTube, Spotify, ou du Billboard [7, 17]. Certaines études incluent des paramètres tels que la "superstar variable", qui prend en compte la notoriété d'un artiste, ou le score de popularité des morceaux [10]. Cependant, ces mesures peuvent fortement influencer la capacité prédictive des modèles. Dans cette étude, je propose d'introduire de nouvelles variables basées sur des caractéristiques musicales "abstraites" extraites des spectrogrammes.

Cet article sera organisé comme suit : Dans la deuxième section on parlera de la création du jeu de données et des différentes caractéristiques musicales extraites. Dans la troisième section, on fera une analyse de la base de données. La quatrième section abordera les techniques d'apprentissage automatique qui ont permis d'explorer l'hypothèse initiale. Enfin, on concluera sur les résultats, les limites et les directions futures.

2 Création de la base de données

L'une des tâches les plus chronophages de ce projet a été la collecte de données. Aucun jeu de données avec une classification binaire (Hit / non-Hit) n'était disponible, ce qui a rendu cette tâche fastidieuse, représentant environ 60% du temps investi (figure 1).

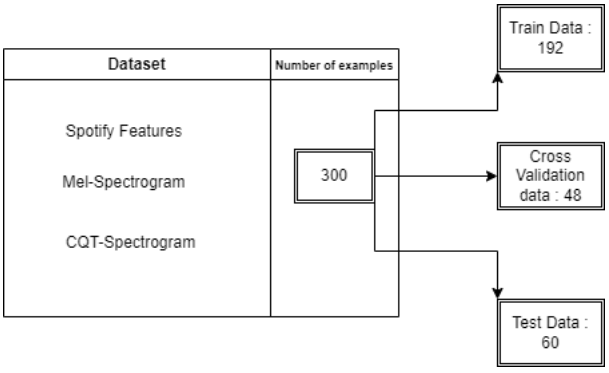

Figure 1: La composition du jeu de données

La création de la base de données s'est déroulée en trois étapes. La première consistait à dresser une liste de sons hits/non-hits. Ensuite, deux playlists distinctes sont créées sur Spotify pour extraire les caractéristiques musicales des morceaux grâce à l'API de Spotify. Enfin, l'API de YouTube a été utilisée pour obtenir les vidéos d'artistes au format MP4 et les convertir en MP3, afin de générer des Mel-spectrogrammes et CQT-spectrogrammes.

2.1 Notion de hits/non-hits et Notion de streamed/non-streamed

Avant de poursuivre cette étude, une précision s'impose. La désignation d'un son comme non-hit est problématique, surtout pour les artistes connus. Comment expliquerez-vous à des fans de Michael Jackson

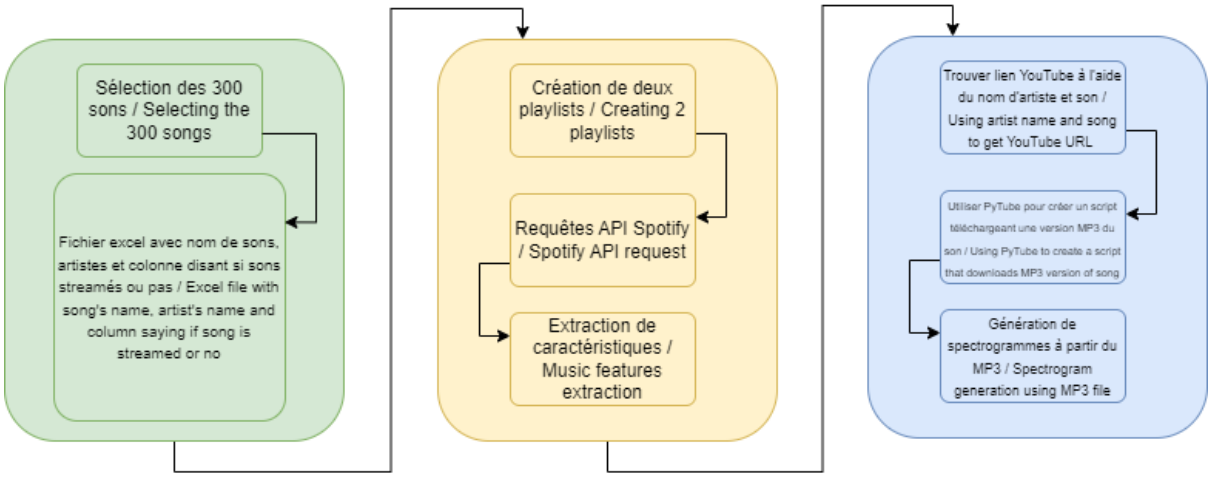

Figure 2: Etapes de la création de la Base de données

que la reprise de "We are the World", "We are the world 25 for Haïti" est considérée comme un des "pires morceaux de l'histoire" malgré ses 375 millions de vues sur YouTube [16]. De plus, contrairement aux morceaux qui sont des hits, il n'existe pas de listes fiables. Les listes faites sont en général assez subjectives étant donné qu'elles sont faites selon l'appréciation de la personne qui les a faites.

Ainsi, il est plus judicieux de parler de sons streamés ou moins streamés. Un son sera considéré comme moins streamé si le rapport entre le nombre de vues sur YouTube et le nombre d'abonnés de l'artiste est inférieur ou égal à 1.5. Cette mesure sera appelée "streamabilité". Par exemple, "Gasoline" de The Weeknd considéré dans cette étude comme moins streamé a une streamabilité de 0.51, soit un rapport de 18 millions de vues sur 35.4 millions d'abonnés.

Inversement, la liste de sons streamés est plus facile à établir en se référant au Billboard TOP, aux morceaux ayant une streamabilité de 10 ou plus, ou aux hits intemporels de divers pays. J'ai sélectionné les meilleurs morceaux de pays musicalement influents.

2.2 Etape 1 - Création de la liste de sons streamés/non-streamés

Comme mentionné précédemment, choisir les morceaux les plus streamés n'a pas été difficile. Il existe de nombreux classements, comme les TOP 10, 20, 50, et 100 des sons les plus streamés de tous les temps. Je me suis principalement appuyé sur les classements du Billboard et les morceaux les plus écoutés dans chaque pays.

Pour les morceaux moins streamés, j'ai utilisé le critère de streamabilité, en sélectionnant ceux avec une valeur inférieure à 1.5. Pour simplifier la création de cette liste, j'ai principalement choisi d'autres morceaux des mêmes artistes déjà présents dans la liste des streamés.

2.3 Etape 2 - Utilisation de l'API de Spotify et Extraction de caractéristiques musicales basiques

Dans cette étape, l'objectif est de créer deux playlists sur Spotify. Pourquoi créer des playlists ? Et surtout pourquoi en créer deux ?

L'utilisation de l'API Spotify présente des limitations. Bien qu'il soit possible de vérifier la popularité d'un morceau(une mesure variant entre 0 et 100, 100 indiquant que le morceau très populaire), il n'est pas possible de filtrer directement par popularité et par pays. Par conséquent, la création manuelle des playlists est nécessaire. Cette approche manuelle, bien que chronophage, permet d'éviter les erreurs dans

la sélection des morceaux, en particulier pour les titres en langues moins courantes (arabe, mandarin, alphasyllabaire Thaï...). De plus, utiliser l'API de Spotify pour extraire les caractéristiques musicales des playlists présente des avantages. En récupérant l'identifiant de la playlist (ID à récupérer sur Spotify une fois la playlist créée), on passe d'une extraction singulière par ID de morceau (morceau par morceau) à une extraction plurielle par ID de playlist (tous les morceaux de la playlist) en une seule requête, simplifiant le processus d'extraction et garantissant l'intégrité des données.

Pour la deuxième question, la raison est toute simple, Avoir deux playlists facilitent la séparation entre les sons streamés et moins streamés, réduisant le risque d'erreurs de labellisation.

Un script a été mis en place pour gérer ce processus. A la fin de l'étape 2,on a deux fichiers CSV contenant les caractéristiques musicales essentielles pour le reste de l'étude.

Voici la liste des caractéristiques musicales qu'on peut obtenir grâce à Spotify [14] :

- L'acousticness : "Une mesure de confiance entre 0,0 à 1,0 pour déterminer si la piste est acoustique. 1,0 représente une confiance élevée dans le fait que la piste soit acoustique. " Un son acoustique est un son dans lequel il n'y a que des instruments de musique fonctionnant sans électricité.

- La dansabilité : La dansabilité décrit dans quelle mesure un morceau est dansant sur la base d'une combinaison d'éléments musicaux tels que le tempo, la stabilité du rythme, la force de la pulsation et la régularité générale. Une valeur de 0.0 est la moins dansante et une valeur de 1.0 est la plus dansante.

- La durée (en millisecondes)

- L'instrumentalité : Détermine si une piste ne contient pas de voix. Les sons "Ooh" et "aah" sont considérés comme instrumentaux dans ce contexte. Les morceaux de rap ou de musique parlée sont clairement "vocaux". Plus la valeur de l'instrumentalité est proche de 1.0, plus il est probable que la piste ne contienne pas de contenu vocal. Les valeurs supérieures à 0.5 sont censées représenter des pistes instrumentales, mais la confiance est d'autant plus grande que la valeur s'approche de 1.0.

- La Clé (ou la gamme) : Exemple : 0 pour Do/C , 1 pour Do#/Reb ou C#/Db ainsi de suite. Les valeurs sont entre -1 et 11. Une clé de -1 correspond à une piste pour laquelle on n'a pas détecté la clé.

- L'intensité sonore : L'intensité sonore globale d'une piste en décibels (dB). Les valeurs d'intensité sonore sont calculées en moyenne sur l'ensemble de la piste et sont utiles pour comparer l'intensité sonore relative des pistes. Les valeurs se situent généralement entre -60 et 0 db.

- Le mode : (mineur ou majeure) : 1 pour majeur et 0 pour mineur

- La speechiness : L'attribut Speechiness détecte la présence de mots parlés dans une piste. Plus l'enregistrement est exclusivement vocal (par exemple, talk-show, livre audio, poésie), plus la valeur de l'attribut est proche de 1.0. Les valeurs supérieures à 0.66 décrivent des pistes qui sont probablement composées exclusivement de paroles. Les valeurs comprises entre 0.33 et 0.66 décrivent des pistes qui peuvent contenir à la fois de la musique et de la parole, soit en sections, soit en couches, y compris des cas tels que la musique rap. Les valeurs inférieures à 0.33 représentent très probablement de la musique et d'autres pistes ne ressemblant pas à de la parole.

- Le tempo : estimation des BPM (beats par minute).

- La signature temporelle (ou métrique): Une estimation de la signature temporelle. La métrique est une convention de notation qui permet de spécifier le nombre de temps dans chaque mesure.

La signature temporelle va de 3 à 7, indiquant des signatures temporelles de "3/4" à "7/4". " A noter qu'il existe d'autres signatures. Par exemple, le 3/8, 6/8... Pour cette caractéristique, on restera tout de même sur les informations de Spotify.

- La valence : Une mesure de 0.0 à 1.0 décrivant la positivité musicale transmise par une piste. Les morceaux ayant une valence élevée sonnent plus positivement (par exemple, heureux, gai, euphorique), tandis que les morceaux ayant une valence faible sonnent plus négativement (par exemple, triste, déprimé, en colère).
- L'énergie : L'énergie est une mesure comprise entre 0.0 et 1.0 et représente une mesure perceptuelle de l'intensité et de l'activité. En règle générale, les morceaux énergiques sont rapides, forts et bruyants. Par exemple, le death metal a une énergie élevée, tandis qu'un prélude de Bach obtient un score faible sur l'échelle. Les caractéristiques perceptives contribuant à cet attribut comprennent la gamme dynamique, l'intensité sonore perçue, le timbre, la vitesse d'apparition et l'entropie générale.
- La liveness : Détecte la présence d'un public dans l'enregistrement. Des valeurs de vivacité plus élevées représentent une probabilité accrue que la piste ait été jouée en direct. Une valeur supérieure à 0.8 indique une forte probabilité que la piste soit en live.

  D'autres caractéristiques sont récupérables mais elles ne seront pas considérées comme musicales (l'ID du son, le lien spotify, uri, ...)

Figure 3: Caractéristiques obtenues sur Spotify

2.4 Etape 3 - Utilisation de l'API de YouTube et Génération de spectrogrammes

2.4.1 Rappel sur le traitement numérique du signal

Il est très courant de réprésenter l'évolution d'un signal dans le domaine fréquentiel ou dans le fréquenciel [5]. Cette dernière réprésentation est obtenue en appliquant la Transformée de Fourier au signal dans

son domaine temporel. En traitement du signal, tout signal est échantillonné, c'est-à-dire qu'une discrétisation se fait, avec un pas de discrétisation qu'on appelle la fréquence d'échantillonnage, ou sample rate (sr). Par exemple, pour un signal de 10 secondes, en fixant le sr a 44100, le signal comptera 44100 échantillons par seconde. La taille totale du signal sera de 441 000 échantillons.

$$Ns = \text{sr} \times \text{durée du signal (en secondes)}$$

où Ns réprésente le nombre total d'échantillons du signal.

Seulement, utiliser des valeurs discrètes demande à adapter la transformée de Fourier classique. De ce fait, on utilise ce qu'on appelle la Fast Fourier Transform *(FFT)* qui est la transformée de Fourier adaptée au domaine du traitement numérique du signal [11].

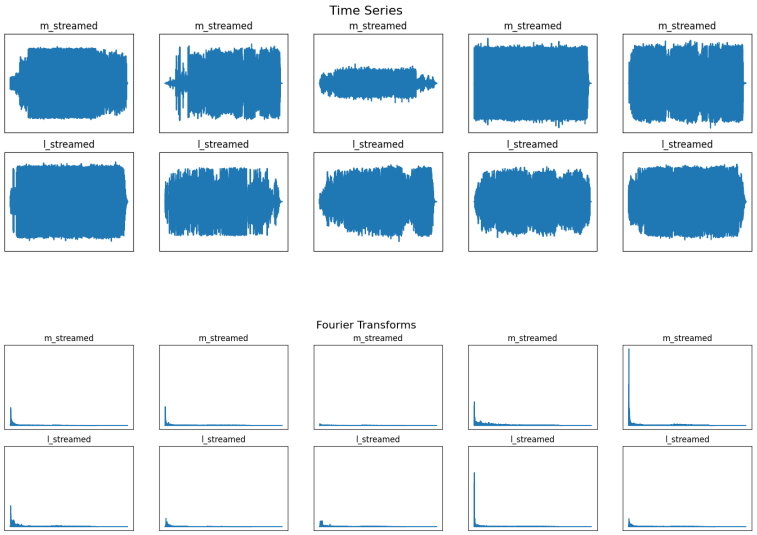

Figure 4: Evolutions temporelles et fréquentielles de morceaux streamés et non streamés : m_streamed pour morceaux streamés et l_streamed pour morceaux moins streamés

Une grande partie de cette étude se porte sur l'extraction de caractéristiques musicales à partir d'images de spectrogrammes. Un spectrogramme est un diagramme sur lequel on peut observer le spectre d'un son dans le temps. Pour ceux qui sont familiers aux traitements du son, le spectrogramme se calcule grâce à la *TFCT* (Transformée de Fourier à court Terme) ou *STFT* (Short-Time Fourier Transform) [3]. Il y a une petite différence entre la FFT et la STFT. Avec la STFT, on peut avoir cette dimension temporelle qu'on n'a pas avec la FFT parce que la STFT est une FFT calculée sur plusieurs segments du signal. La taille de la fenêtre est définie en amont et varie en fonction de l'utilisation qu'on a à faire. Lorsque la FFT est appliquée à un segment du signal, on obtient une trame. La STFT permet donc d'avoir plusieurs trames. Deux paramètres sont à connaître également : *N_FFT*, la taille de la FFT, c'est-à-dire la taille de la fenêtre de chaque trame et "*HOP_LENGTH* ou pas de trame en français" qui détermine le nombre d'échantillons se chevauchant entre deux trames. Prenons un tout petit signal audio de 12 échantillons en tout. On décide de calculer le nombre de points qu'il y aura pour son spectrogramme. On fixe pour cet exemple *N_FFT* à 6 et *HOP_LENGTH* à 2. Cela veut dire que pour chaque fenêtre de 6 échantillons, on aura :

$$6 - 2 \Rightarrow (N\_FFT - HOP\_LENGTH)$$

soit 4 échantillons qui se chevaucheront entre deux trames consécutives. Il en résultera au total 4 trames

: 1 à 6, 3 à 8, 5 à 10 et 7 à 12. Vous constaterez que le numéro de début de chaque trame est : 1, 3, 5, 7. Il y a une différence de 2 à chaque fois, réprésentant la taille de *HOP_LENGTH* choisie.

Il faut noter qu'une fenêtre trop grande (*N_FFT*) permet d'avoir une bonne résolution fréquentielle, mais engendre d'un autre côté baisse de la résolution temporelle et vice versa [8]. Un HOP LENGTH faible entrainera un chevauchement plus important, et permettra de voir une continuité de la trame n sur la trame n+1, du fait qu'ils aient des échantillons communs. Pour les morceaux de cette étude, la fréquence d'échantillonnage choisie est de 44100 Hz. Les fenêtres seront de 1024 ou 2048 échantillons en fonction de la tâche à exécuter : 2048 échantillons correspondent à 2048/44100 secondes, soit 0.046 secondes ou 46 millisecondes)

Deux types de spectrogrammes ont été utilisés pour cette étape:

- Mel-Spectrogramme : un spectrogramme avec une lisibilité bien meilleure au niveau des fréquences (les fréquences sont sur les ordonnées du spectrogramme). L'oreille humaine identifie plus facilement les fréquences basses que hautes. A titre d'exemple, la différence entre le Si1 et le Do2 (123.4 Hz à 130.8 Hz) est de 7.4Hz alors que celle entre le Si7 et le Do8 (3951 Hz et 4186 Hz) est de 235Hz. La différence de fréquences dans les basses fréquences est assez faible. Cette différence augmente dans les hautes fréquences. Les fréquences que perçoivent les humains sont sur une échelle logarithmique [15]. D'où l'importance du Mel-Spectrogramme. C'est un spectrogramme qui prend en compte cette particularité de la perception des fréquences de l'oreille humaine.

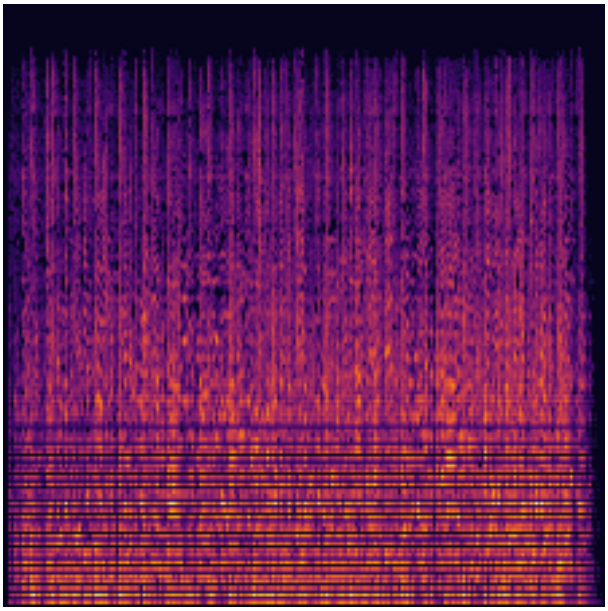

Figure 5: Exemple de Mel-Spectrogramme de *Is This Love de Bob Marley*

- CQT-Spectrogramme : Le CQT spectrogramme est calculé à partir de la CQT (Constant-Q Transform). Elle donne une représentation de la piste musicale en fonction de bins de fréquence. Les bins sont les intervalles dans l'espace des fréquences [4]. De manière générale, on choisit 12 bins, qui correspondent aux 12 notes musicales (du Do/C au Si/B). Et si on a 7 octaves, ça donne 84 bins (12 * 7 bins au total). Ainsi avec le CQT spectrogramme on a une représentation du son en fonction des bins de fréquence. La raison du choix du CQT spectrogramme est qu'il est souvent utilisé en estimation de fréquences fondamentales [12] ou en classification d'audio et donne de bons résultats.

Ces informations sont assez simplistes. Le but n'étant pas de faire un cours de traitement numérique du signal mais surtout de montrer la démarche dans l'exécution de ce projet.

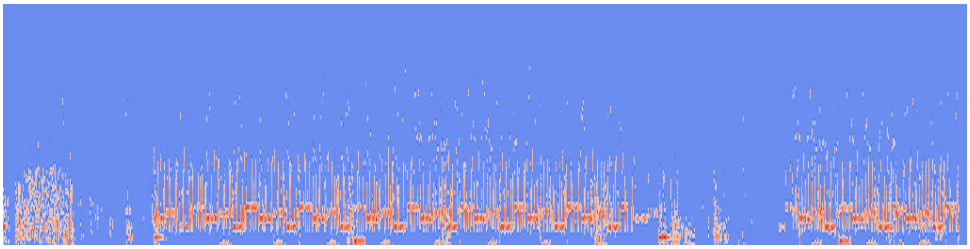

Figure 6: Exemple de CQT-Spectrogramme de *Gasoline de The Weeknd*

2.4.2 Le rôle de l'API de YouTube

L'objectif principal de cette étape est de générer des spectrogrammes. Lors de l'étape de création de la liste (étape 1), j'ai utilisé un fichier Excel contenant tous les sons et les noms d'artiste. Une colonne pour la labellisation a été ajoutée également afin de distinguer les "streamés" et "moins streamés". Ce fichier facilitera les requêtes de recherche sur l'API de YouTube via un script qui utilisera le nom de l'artiste et du morceau.

L'avantage de YouTube réside dans la précision de ses recherches. Le script entre le nom de l'artiste et du morceau dans la requête pour trouver le lien de la vidéo YouTube. Ce lien est ensuite utilisé avec le module "PyTube" pour télécharger le son en format MP3.

Il est important de noter que YouTube impose une limite quotidienne sur le nombre de requêtes. Chaque recherche consomme 100 points, avec une limite de 10,000 points par jour. Cela permet de télécharger jusqu'à 100 vidéos par jour. Ainsi, pour traiter 300 sons, il faudra trois jours. Pour une base de données de 1,000 à 10,000 sons, le temps requis est bien plus considérable. C'était le premier frein et le plus important à la création d'une base de données conséquentes

3 Analyse de la base de données

Dans cette partie, on va analyser le jeu de données afin de mieux comprendre les données, et voir si des relations existent entre les variables explicatives (dansabilité, speechness, tempo, ...) et la variable cible (Streamé ou Non-streamé).

3.1 Analyse des variables explicatives continues

Dans les caractéristiques obtenues lors de la 2ème étape de la création de la base de données, on distingue neuf variables continues. Analysons leurs distributions :

- Dansabilité : On remarque que la densité de la dansabilité pour les sons streamés est légèrement en dessous de celle des sons non streamés lorsque la dansabilité est inférieure à 0.6. On remarque néanmoins un pic de densité autour de 0.8 pour les sons streamés.
- Energie : Les sons streamés ont un pic de densité d'énergie autour de 0.8, contre 0.6 pour les non streamés. Les sons streamés semblent être beaucoup plus énergiques que les non streamés.
- Intensité Sonore : La courbe de densité des non streamés est au-dessus de celle des streamés principalement entre [-10dB et -5dB].
- Speechiness : La densité de la speechiness des streamés est très au-dessus de celle des non streamés entre 0.0 et 0.1. La majorité des sons streamés semblent avoir moins de speechness en général.

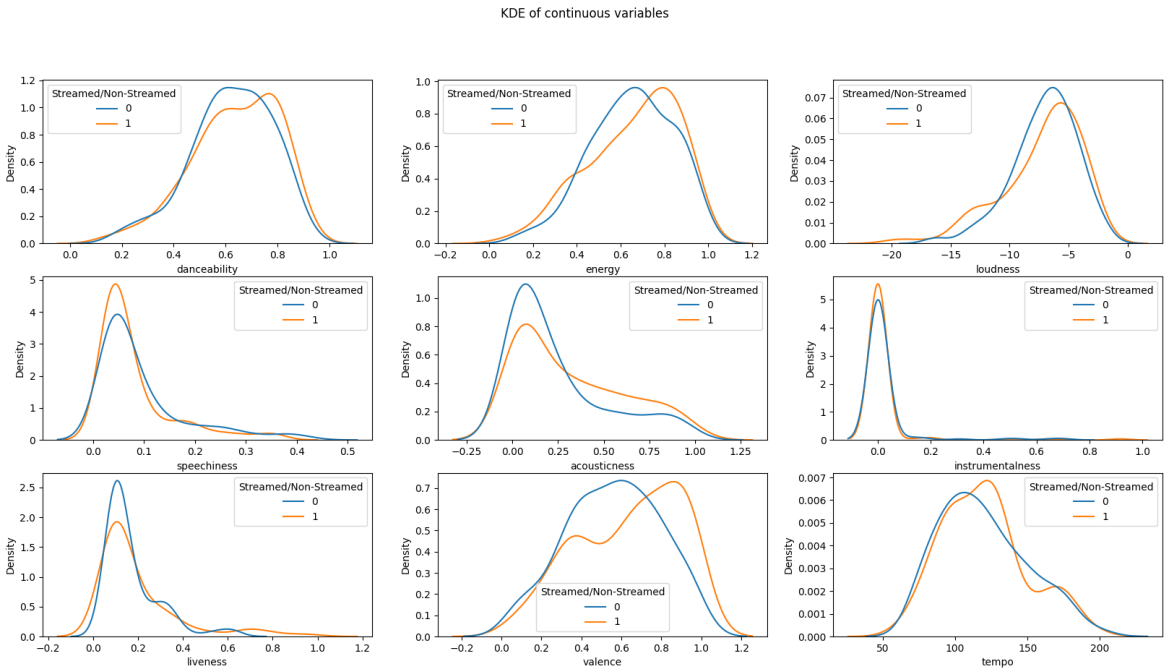

Figure 7: Analyse des variables continues : la dansabilité, l'énergie, l'intensité sonore, la speechiness, l'acousticness, l'instrumentalness, la liveness, la valence, le tempo

- Acousticness : Les sons streamés ont un pic de densité inférieur à celui des non streamés lorsque les sons ne pas très acoustiques. Mais on retrouve plus d'acoustique dans les streamés lorsque celle ci augmente.
- Instrumentalité : Les streamés et non streamés ont tous les deux un pic de densité aux mêmes valeurs instrumentalité. Celle des streamés est légèrement plus élevée. Cette caractéristique ne sera pas forcément discrimative lors de la modélisation.
- Liveness : On peut affirmer que sur la quasi-totalité du graphe, la densité de liveness est beaucoup plus faible sur les streamés. On pouvait s'attendre à ça, les morceaux qu'on écoute en général ont été mixés dans un studio.
- Valence : On remarque que la densité de valence des sons streamés est plus élevée lorsque la valence augmente. Les morceaux streamés semblent être des morceaux plus positifs, joyeux que les non streamés.
- Tempo : On observe un pic de densité pour les sons streamés autour de 130-140 bpm, qu'on peut considérer (si on faisait une discrétisation selon les quantiles) comme tempo rapide. Pour les non streamés, le pic est autour de 100 bpm, donc tempo modéré.

3.2 Analyse des variables explicatives discrètes

Dans les caractéristiques obtenues lors de la 2ème étape de la création de la base de données, on distingue également trois variables discrètes. Analysons leur distribution

- Clé / Gamme : On remarque que le nombre de sons streamés est très important dans la gamme de C/DO, C#/DO#, Eb/Mibémol (0,1,3) comparativement aux sons non streamés

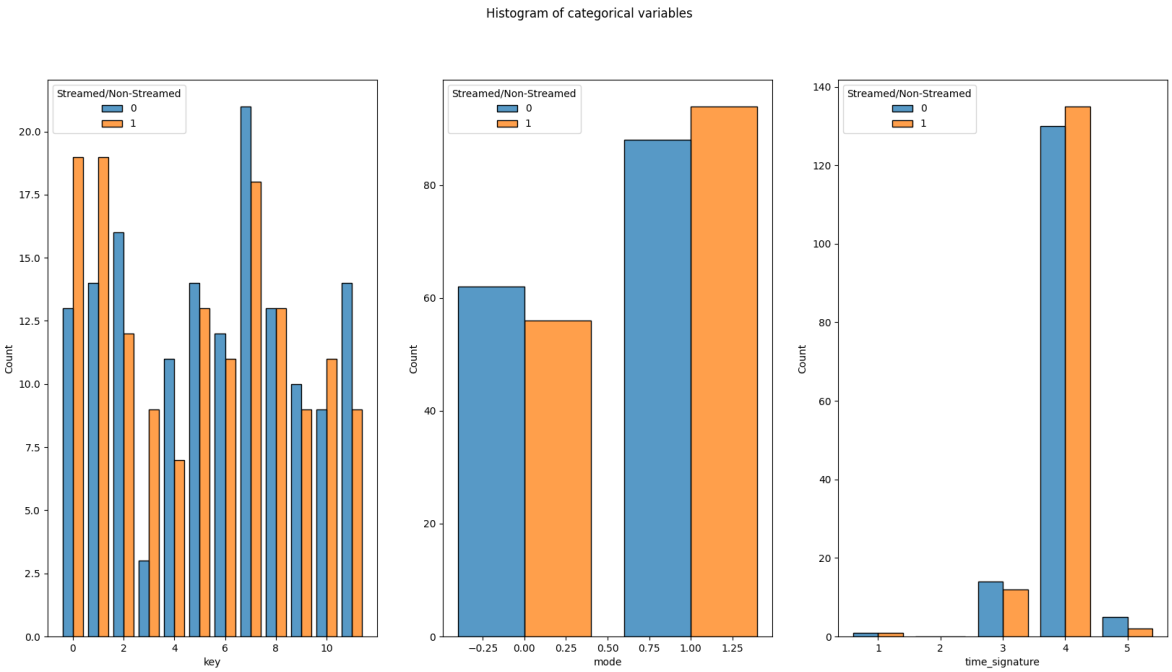

Figure 8: Analyse des variables discrètes : la clé, le mode, la métrique

- Mode : On peut déduire de ce graphe qu'on retrouve plus de sons dans les gammes mineures chez les non streamés. C'est l'inverse dans les gammes majeures. Ce qui rejoint quelque part l'analyse faite sur la valence.
- Métrique / signature temporelle : Il y a un pic partagé autour de la métrique 4/4. La proportion est juste un peu plus élevée pour les hits. Mais on peut supposer que cette variable ne sera pas assez discriminative.

4 Modélisation

Dans cette section, je vais utiliser des techniques d'apprentissage automatique pour construire un modèle prédictif solide. Les spectrogrammes générés lors de la troisième étape de création de la base de données seront cruciaux. L'objectif est d'extraire des caractéristiques abstraites des spectrogrammes et de les combiner aux caractéristiques de base de notre jeu de données. Cela pourrait améliorer la précision du modèle de prédiction.

4.1 Modèles de Machine Learning sur le jeu de données de base

Afin de mieux évaluer l'impact des features abstraites des spectrogrammes sur la prédiction, il est impératif de vérifier les performances des modèles de prédiction en utilisant uniquement les 12 features de base. A cette fin, deux modèles de machine learning ont été comparés : La régression logistique et les forêts aléatoires.

4.1.1 Régression Logistique et résultats de prédiction

La régression logistique est une méthode permettant de prédire la probabilité d'un évenement binaire. D'une façon, elle découle d'un modèle de Bernoulli [1]. Avec x_n un ensemble de prédicteurs et y_n la

sortie binaire dont on veut connaitre la probabilité, on a :

$$P(Y_n = 1 \mid x_n) = \sigma(w \cdot x_n)$$

où $\sigma(z) = \frac{1}{1+e^{-z}}$, avec $z = w \cdot x_n$, z étant dans l'intervalle (0,1) et w de la même longeur que x_n et réprésentant les poids de chaque prédicteur. Dans la régression logistique, un autre principe de probabilité entre en jeu, le principe de vraisemblance. A travers la fonction de Vraisemblance, on évalue la probabilité des observations données au modèle, et on trouve les coefficients qui maximisent la vraisemblance de ces observations.

Dans notre cas, la régression logistique donnait une précision de 57% sur les données de test.

Classification Report:

	precision	recall	f1-score	support
0	0.58	0.58	0.58	31
1	0.55	0.55	0.55	29
accuracy			0.57	60
macro avg	0.57	0.57	0.57	60
weighted avg	0.57	0.57	0.57	60

Figure 9: Précision du modèle avec régression logistique

On a également obtenu les résultats ci-dessous :

Logit Regression Results

Dep. Variable:	Streamed/Non-Streamed	No. Observations:	240
Model:	Logit	Df Residuals:	227
Method:	MLE	Df Model:	12
Date:	Wed, 24 Jul 2024	Pseudo R-squ.:	0.05608
Time:	15:20:07	Log-Likelihood:	-157.02
converged:	True	LL-Null:	-166.35
Covariance Type:	nonrobust	LLR p-value:	0.09710

	coef	std err	z	P>|z|	[0.025	0.975]
const	-1.8556	1.931	-0.961	0.337	-5.641	1.930
danceability	-0.1443	1.130	-0.128	0.898	-2.358	2.070
energy	1.1134	1.274	0.874	0.382	-1.384	3.611
loudness	-0.0523	0.065	-0.809	0.419	-0.179	0.074
speechiness	-1.9642	1.925	-1.020	0.308	-5.738	1.809
acousticness	1.6068	0.639	2.515	0.012	0.355	2.859
instrumentalness	-0.9100	1.308	-0.696	0.487	-3.474	1.654
liveness	1.0905	0.926	1.178	0.239	-0.724	2.905
valence	1.2388	0.734	1.687	0.092	-0.201	2.678
tempo	0.0016	0.005	0.333	0.739	-0.008	0.011
key	-0.0345	0.039	-0.875	0.381	-0.112	0.043
mode	-0.1928	0.286	-0.675	0.500	-0.753	0.367
time_signature	-0.0598	0.326	-0.183	0.855	-0.699	0.579

Figure 10: Résultats de classification avec la régression logistique

On peut remarquer qu'il y a que l'acousticness qui a un effet statistiquement significatif sur les prédictions. La plupart des variables explicatives ne montrent pas d'effet significatif sur le résultat. Le modèle est à améliorer.

4.1.2 Random Forest et résultats de prédiction

Les forêts aléatoires (random forest) sont des modèles d'ensemble, introduit en 2001 par Breiman. Le but est de corriger la propension de chaque arbre individuel en utilisant ce qu'on appelle "le bagging", technique introduite en 1996 par Brieman, afin de construire une large collection d'arbres de décision "décorrélés" [2]. C'est une méthode utilisée couramment pour sa robustesse et qui permet d'éviter le surapprentissage.

Dans notre cas, cette méthode a permis d'avoir un modèle de prédiction avec une précision de 53% sur les données de test.

Classification Report:

	precision	recall	f1-score	support
0	0.55	0.52	0.53	31
1	0.52	0.55	0.53	29
accuracy			0.53	60
macro avg	0.53	0.53	0.53	60
weighted avg	0.53	0.53	0.53	60

Figure 11: Résultats de classification avec la random forest

4.1.3 Choix du modèle pour les caractéristiques de base

On peut clairement dire que les deux modèles ont une précision très faible. La précision étant inférieur à 60%, on peut dire qu'un modèle avec des prédictions aléatoires auraient les mêmes résultats (après tout, les données sont reparties de façon égale, 50% pour les streamés, et 50% pour les non-streamés). Bien que la régression logistique ait donné de meilleurs résultats, nous utiliserons une autre technique d'ensemble utilisant des arbres de décision (un autre technique d'ensemble comme les random forest) : Gradient Boosting. Les résultats de régression ne sont pas statistiquement convaincants pour les combiner aux résultats qu'on obtiendra des spectrogrammes. Vous retrouverez les sources codes de cette partie dans le fichier suivant : `modelcomparisondatasetnospectrogram.ipynb`

4.2 Méthodes proposées

Pour l'extraction de features abstraites, j'ai exploreré deux approches. Des approches assez similaires avaient été utilisées dans des modèles d'estimation de fréquences fondamentales , avec des résultats prometteurs [13]. Une fois les spectrogrammes générés, ils seront utilisés en entrées pour des CNN afin d'extraire des caractéristiques musicales . Ces caractéristiques seront ensuite combinées à celles de l'API de Spotify afin de voir si cette combinaison améliore les prédictions.

4.2.1 Première approche

La première approche est relativement simple et directe. Deux types de spectrogramme sont générés à partir des fichiers audios : un Mel-spectrogramme et un CQT spectrogramme. La durée de fichier audio étant différente, il en résulte des spectrogrammes de tailles différentes [1]. La taille de *N_FFT* sera de 2048 et *HOP_LENGTH* 512. Pour faciliter le traitement, chaque spectrogramme est converti

[1]Il faut distinguer la matrice du spectrogramme et les images. Les images sont créées grâce à la matrice et ont leur dimension de largeur réduite par rapport aux spectrogrammes.

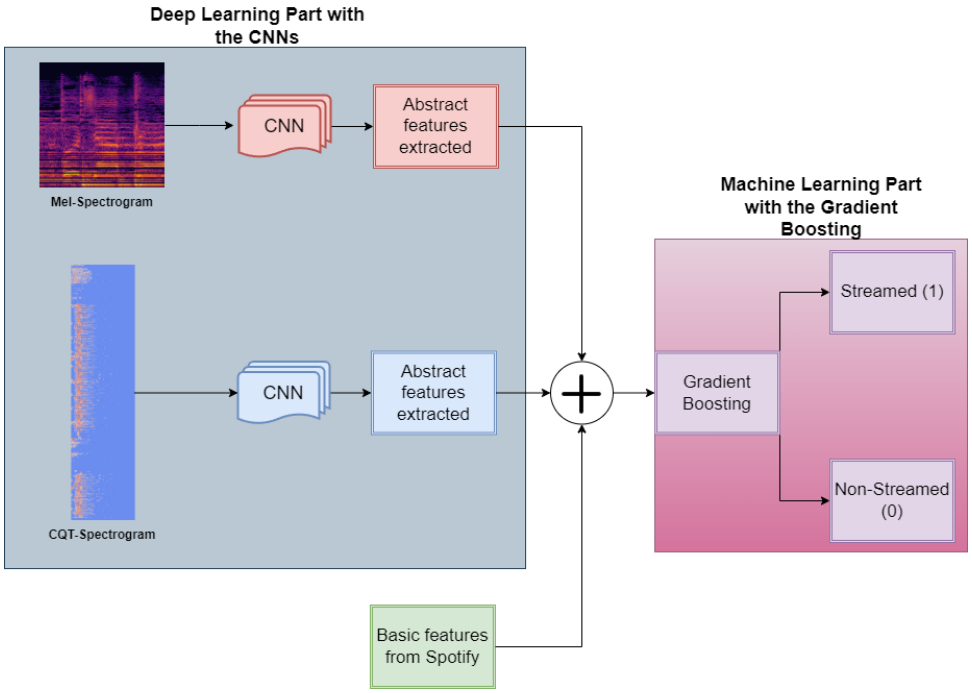

Figure 12: Illustration du modèle proposé

en une image avec des dimensions fixes : 224x224 pixels pour les Mel-spectrogrammes et 256x512 pixels pour les CQT-spectrogrammes, en utilisant trois canaux (RGB). Les dimensions de 224x224x3 pour les Mel-spectrogrammes et 256x512x3 pour les CQT-spectrogrammes ont été choisies après plusieurs expérimentations. Cette standardisation facilite le traitement tout en équilibrant la perte d'information due à la réduction de la taille des images. Cette approche a différents avantages. Elle permet d'avoir tout le spectre de chaque son et gérer la variabilité de la durée des signaux audio sans nécessiter de traitement supplémentaire. De plus, elle a un coût computationnel relativement faible. La principale limite est la perte d'information due à la conversion des matrices de spectrogrammes en images. Les dimensions réduites peuvent entraîner une perte de détails importants des spectrogrammes. En effet, chaque spectrogramme est fondamentalement une matrice que je convertis en image. Pour chaque spectrogramme, les dimensions sont généralement (N_FFT/2, H) ou (nombre de bins * nombre d'octaves, H), où H est un nombre supérieur à 10000 dans notre étude. Lorsqu'on sauvegarde ces spectrogrammes sous forme d'images avec des dimensions de 224x224x3 ou 256x512x3, il y a nécessairement une perte d'information due à la réduction de taille.

Cela nous pousse à considérer la deuxième approche, qui vise à préserver davantage les détails des spectrogrammes en générant plusieurs images à partir d'un seul spectrogramme.

4.2.2 Deuxième approche

La deuxième approche vise à maximiser la conservation de l'information en créant plusieurs images de spectrogrammes pour chaque piste sonore. Pour les Mel-spectrogrammes, 30 images sont générées en découpant le spectrogramme en petites matrices de 200 échantillons chacune. Les Mel-spectrogrammes ont été générés avec une *N_FFT* de 1024 et un *HOP_LENGTH* de 256. Il est important de noter que dans ce contexte, 200 échantillons correspondent à 200 trames. Ainsi, chaque petite matrice représente 200 trames extraites du spectrogramme original. En créant ces 30 petites matrices, on obtient une vue détaillée de différentes portions du spectrogramme, permettant ainsi de conserver plus d'information tout en travaillant avec des segments plus gérables.

Pour les CQT-spectrogrammes, chaque matrice est divisée en deux spectrogrammes de tailles égales, ou en un spectrogramme avec une trame supplémentaire si le nombre total d'échantillons est impair.

Cette méthode conserve une grande quantité d'information en offrant plusieurs vues détaillées des spectrogrammes. Cela pourrait potentiellement améliorer la performance du modèle en capturant des nuances supplémentaires. L'inconvénient principal est le coût computationnel très élevé.

4.3 Réprésentation de l'entrée

Le model utilisé aura deux différentes parties (voir figure 12) : Une partie Deep Learning avec les CNN et une partie Machine Learning avec la concaténation de toutes les caractéristiques (basiques et abstraites) qu'on utilisera pour le Gradient Boosting.

4.3.1 Partie Deep Learning

Pour chacune des deux approches on aura :

- Pour la première approche : Pour les Mel-spectrogrammes, chaque image est de taille 224x224x3, ce qui donne une matrice d'entrée de dimensions (300, 224, 224, 3) pour les 300 fichiers audio. Pour les CQT-spectrogrammes, chaque image est de taille 256x512x3, produisant une matrice d'entrée de dimensions (300, 256, 512, 3).
- Pour la deuxième approche : Pour les Mel-spectrogrammes, chaque image est de taille 224x224x3 (ou 224x224x4, certains tests ont été faits en considérant le canal alpha, l'opacité de l'image). Sachant qu'on a 30 spectrogrammes par son, on aura à la fin une matrice de 9000x224x224x3 (ou 9000x224x224x4). C'est le même principe pour les CQT-spectrogrammes. On aura en trée des images de 256x512x3 (ou 256,512,4). Du fait qu'il y ait deux images de CQT-spectrogrammes pour chaque son dans cette approche, il en résulte une matrice de 600x256x512x3 ou (600x256x512x4) [2].

4.3.2 Partie Machine Learning

La partie Machine Learning aura en entrée la concaténation des informations de la partie Deep Learning et les 12 caractéristiques extraites de Spotify. Le nombre de caractéristiques extraites de la partie Deep Learning sera de 128 pour chaque type de spectrogramme. Pour la partie Machine Learning, les tests ont seulement été réalisées selon la première approche à cause du coût computationnel important de la deuxième approche. On aura donc en entrée de cette partie une matrice de dimension (300,128,128,12) où 300 est le nombre de sons [3]

4.4 Réprésentation de la sortie

4.4.1 Partie Deep Learning

En sortie des CNN, une classification binaire a été faite afin de voir si les résultats de classification avec les spectrogrammes étaient concluants. Ceci est juste un check afin de voir si le modèle de CNN (qu'on décrira plus bas) performait. Seuelement, il ne faut pas oublier que le CNN a seulement le but d'extraire des caractéristiques abstraites et pas de faire de classification. De ce fait, Les CNN fournissent, en termes de caractéristiques abstraites,128 caractéristiques pour les Mel-spectrogrammes et 128 pour les

[2] Une chose cruciale est à savoir. Chaque spectrogramme doit être associé au son auquel il correspond afin d'avoir un modèle cohérent. Il faut **impérativement** qu'un son donné soit correctement associé à ces images de spectrogrammes respectifs et combiné aux caractéristiques basiques qui lui sont propres. Cette précaution est assurée par un script.

[3] A noter que de manière indépendante, on a en fait (300,128) pour les Mel-spectrogrammes, (300,128) pour les CQT-spectrogrammes et (300,12) pour les caractéristiques de base. C'est la concaténation de tout ça qui donne : (300,128,128,12)

CQT-spectrogrammes.Ces caractéristiques abstraites sont ensuite utilisées comme entrée pour la partie Machine Learning.

4.4.2 Partie Machine Learning

La sortie de la partie Machine Learning est une classification binaire qui indique si un son est susceptible d'être streamé ou non. Cette classification est basée sur les caractéristiques combinées des spectrogrammes et de Spotify.

4.5 Architecture du réseau de neurones convolutifs utilisé

Pour l'architecture du réseau de neurones (voir figure 13), il fallait éviter les modèles complexes. Vu la très faible quantité de données, un réseau complexe engendrerait forcément un surapprentissage. Pour cette étude, le modèle utilisé s'inspire de celui d'un modèle d'estimation de fréquences fondamentales [6]. Certaines couches ont été revues et un DropOut a été ajouté en sortie de la majorité des couches de convolution et des couches denses afin de palier au maximum au problème de surapprentissage.

Le modèle sera entrainé sur 15 époques avec un batch size de 32. Comme Optimizer, on utilisera Adam. 20% des données seront utilisées pour la validation croisée. (Rappelons encore une fois que le but du CNN n'est pas de classer dans notre cas, mais d'obtenir des caractéristiques).

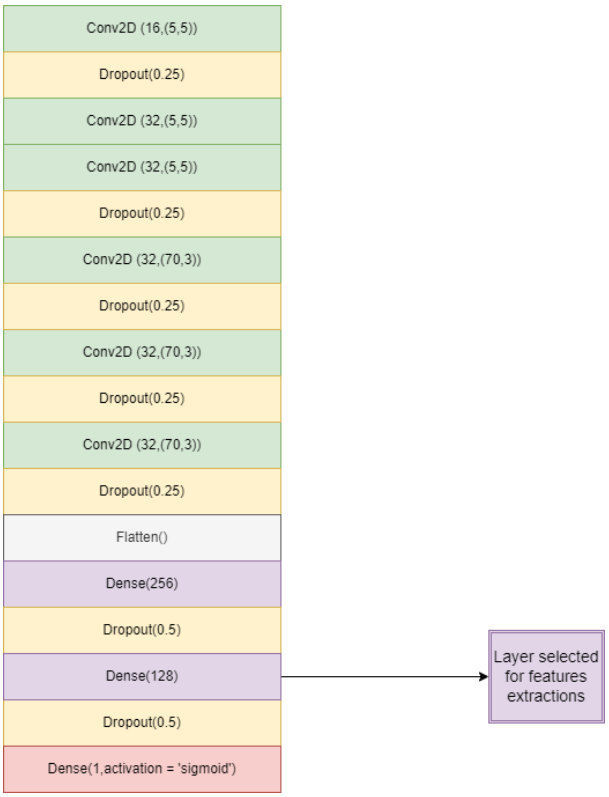

Figure 13: Architecture du CNN utilisé pour l'extraction de features des Mel-spectrogrammes et CQT-spectrogrammes

On pourra remarquer, sur la figure 13, que la couche dense avec les 128 neurones, est celle utilisée pour l'extraction des 128 caractéristiques abstraites.

5 Résultats

Cette section sera divisée en deux parties : premièrement, nous évaluerons les résultats de classification des spectrogramme avec le CNN et leur impact sur la capacité de prédiction du modèle de Gradient Boosting. Premièrement, nous évaluerons uniquement la première approche. Ce choix sera expliquée dans la deuxième partie, lorsque nous concluerons sur les limites rencontrées.

5.1 Evaluation de l'ensemble du modèle

5.1.1 Evaluation de la partie Deep Learning

Les résultats obtenus dans la partie Deep Learning sont les suivants :

- Mel-Spectrogramme : Le modèle atteint une précision de 47.5% sur les données d'apprentissage et 46.6% sur les données de validation croisée. Ces résultats indiquent une performance moyenne et suggèrent que le modèle n'est ni surappris ni particulièrement précis.
- CQT-spectrogramme : Le modèle atteint une précision de 49.17% sur les données d'apprentissage et 53.3% sur les données de validation croisée. Bien que ce modèle soit légèrement meilleur, il reste insuffisant pour une généralisation solide.

Les résultats de ces CNN ne permettent pas de faire une généralisation. Comme on a pu le constater, la précision des deux modèles convolutifs est très basse. Ceci peut être dû à plusieurs facteurs : la très faible quantité de données, ou la difficulté d'interpétation des caractéristiques pour le modèle, ou même l'absence de caractéristiques discriminatives dans les spectrogrammes. Il faudrait dans un premier temps augmenter la quantité de données pour être vraiment fixé. Malheureusement, ceci représente un temps colossal. C'est une piste à explorer dans le futur.

Néanmoins, les deux modèles ont permis d'extraire des caractéristiques abstraites : 128 caractéristiques pour chacun des spectrogrammes (MEL et CQT) qu'on a pu combiner aux 12 autres caractéristiques de l'API de Spotify. Vous retrouverez les détails de cette partie dans le fichier **modelisationspectrogramanddataset.ipynb**

5.1.2 Evaluation du modèle de Gradient Boosting avec les nouvelles caractéristiques

Avant la concaténation des 3 données (Mel-spectrogrammes, CQT-spectrogrammes et Caractéristiques Spotify), on avait une accuracy de 53% avec une random forest. (Les résultats de la régression logistique n'étant pas pris en considération vu qu'ils ne sont statistiquement pas convaincants). Après la fusion des données, on a un modèle avec une précision de 60% (Voir figure 14) sur les données de Test avec le Gradient Boosting [4].

On passe d'un modèle d'une précisionde 53% à un modèle d'une précision de 60%.

Avec la matrice de confusion (figure 15), on peut noter que sur les données qui ont servi de test (60 en tout) :

- True Positive : Le modèle a correctement prédit 19 sons streamés.
- True Negative : Le modèle a correctement prédit 17 sons non-streamés .
- False Negative : 13 sons non streamés ont été classés comme streamés.
- False Positive : 11 sons streamés ont été classés comme non-streamés.

Classification Report:

	precision	recall	f1-score	support
0	0.61	0.57	0.59	30
1	0.59	0.63	0.61	30
accuracy			0.60	60
macro avg	0.60	0.60	0.60	60
weighted avg	0.60	0.60	0.60	60

Figure 14: Résultat du modèle final

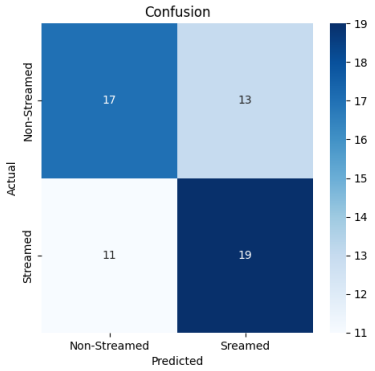

Figure 15: Matrice de confusion du modèle

La courbe ROC (figure 16) obtenu permet de dire que le modèle a quand même un faible pouvoir de discrimination. Cependant, le modèle prédit correctement plus d'échantillons qu'un modèle de prédiction aléatoire ferait. Ça reste améliorable (en ajoutant des données et optimisant encore plus le modèle).

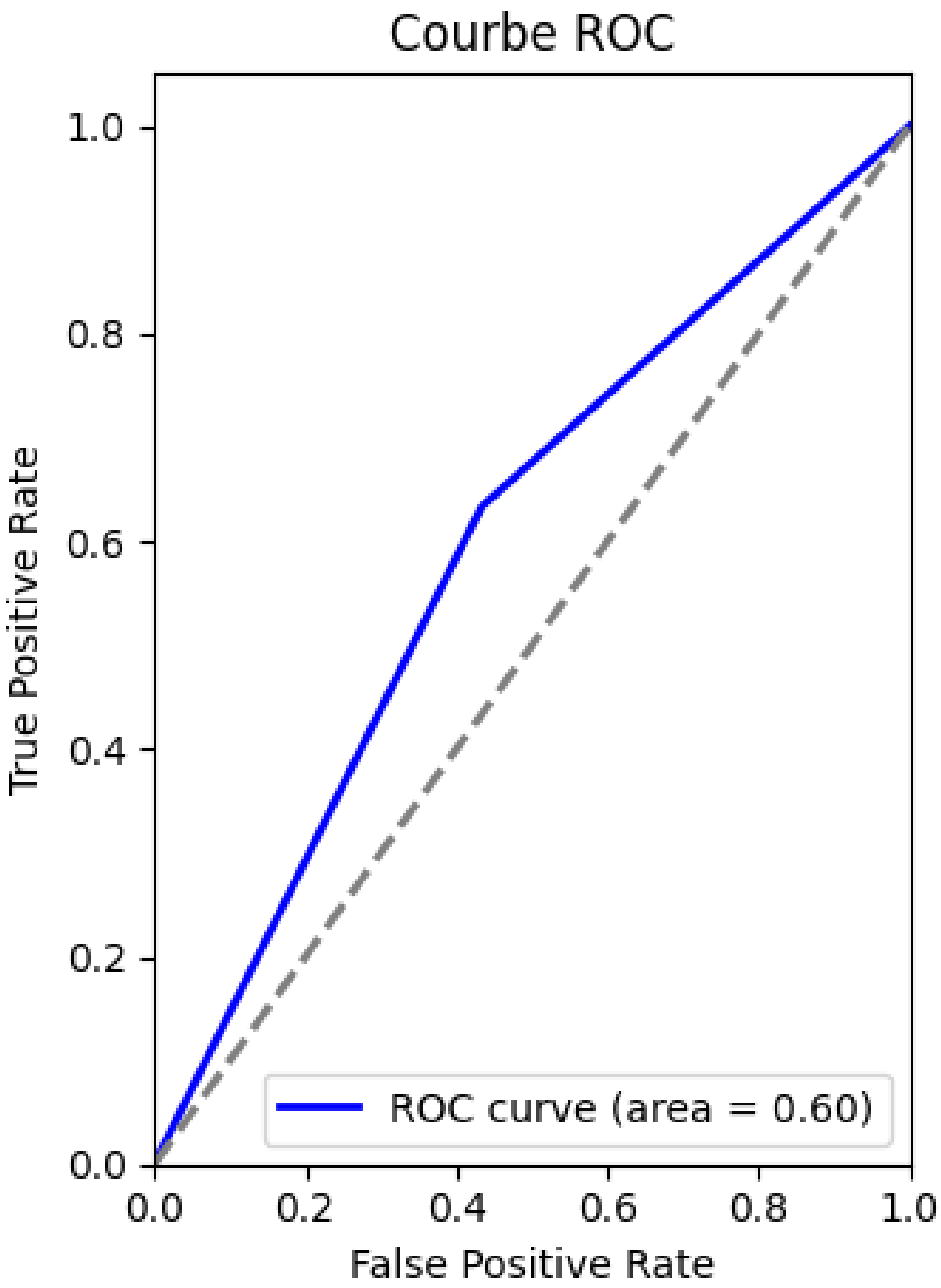

Figure 16: Courbe ROC du Gradient Boosting

Sur toutes les caracctéristiques nouvelles (figure 17) qu'on a pu créer, on peut remarquer que le modèle a su créer 7 features abstraites qui sont plus impactantes que les features obtenues avec l'API de Spotify. Dans les 40 premières caractéristiques importantes, on remarque qu'il n'y a pas le "mode", ce qui peut être surprenant. Cependant, l'acousticness, la valence font toujours partie des caractéristiques importantes.

[4]A noter qu'avec la random Forest nous avons eu une précision de 58% après l'ajout des nouvelles caractéristiques, un gain de 5%

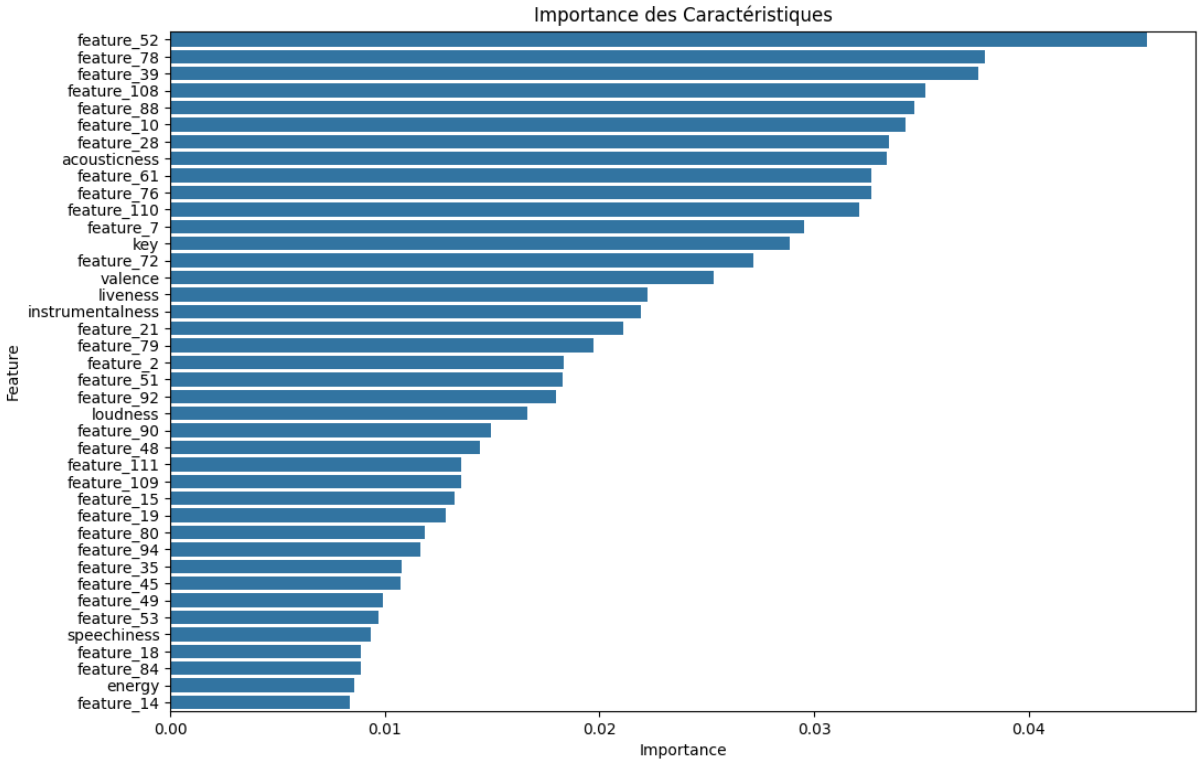

Figure 17: Importance des caractéristiques (Gradient Boosting)

5.2 Limite Computationnelle et discussion sur la deuxième approche

La deuxième approche a un coût computationnel élevé. Cependant, dans le nombre restreints de tests faits, elle semblait avoir des résultats prometteurs. L'idée n'était pas de l'appliquer uniquement à un CNN mais également à un réseau de neurones récurrents (RNN), vu que chaque son a 2 ou 30 images (2 trames pour les CQT et 30 pour les Mel). Même en exécutant avec des ressources assez importantes (300Gb RAM du système ou 51Gb RAM du système couplé à 15Gb RAM du GPU). La possibilité de réduire la complexité de la partie du CNN et du RNN a été envisagée mais le coût computationnel restait élevé. Vous trouverez les notebooks de la seconde approche si l'idée d'explorer cette piste vous intéresse. Je la trouve promettrice.

6 Conclusion

Pour conclure, ce projet visant à voir s'il existait des patterns dans les musiques qui marchaient le mieux a permis d'avoir une robustesse qui peut laisser supposer qu'il pourrait bien avoir des patterns dans les musiques qui deviennent des hits (les plus streamés). Grâce au Gradient Boosting on a pu avoir un modèle précis à 60% (au-dessus du hasard). Ces résultats ne permettent pas forcément de valider complètement l'hypothèse de départ mais montrent bien qu'il faudrait creuser sur le sujet. Une telle précision peut nous montrer également que la musique est beaucoup plus complexe et que chaque individu n'a pas la même appréciation. Cependant, on peut pousser nos recherches avec une base de données plus "sûres" et un nombre d'échantillons beaucoup plus importants. De plus, il faudrait creuser dans la litterature sur les caractétisques musicales qu'on peut obtenir. En mettant en place des méthodes d'extraction de features intéressantes, on pourrait peut-être améliorer le modèle. Néanmoins, je suis globalement satisfait du résultat, vu la faible quantité de données. Un autre point est à creuser. Il est vrai que dire

à un artiste que son morceau a des chances d'être très streamé est bien, mais il serait mieux de lui dire ce qu'il faut améliorer pour que le son le devienne (dans le cas où le modèle prédit que le son ne sera pas streamé). Seulement, c'est là qu'on peut être limité, vu que les features impactantes sont pour la plupart abstraites. Mettre un nom sur quelque chose d'abstrait et comprendre comment notre cerveau interprète ces informations abstraites est assez fascinant.

References

[1] Lyle H. Ungar Andrew I. Schein. Active learning for logistic regression: an evaluation. 2007.

[2] Timothy C. Au. Random forests, decision trees, and categorical predictors: The "absent levels" problem. 2018.

[3] B. Boashash. The short-time fourier transform. *IEEE Signal Processing Magazine*, 11(3):267–270, 1992.

[4] Nicki Holighaus Christian Schörkhuber, Anssi Klapuri and Monika Dôrfler. A matlab toolbox for efficient perfect reconstruction time-frequency transforms with log-frequency resolution. 2014.

[5] Leon Cohen. *Time-Frequency Analysis*. Prentice Hall, 1995. Voir pages 45–52.

[6] Emilia Gómez Helena Cuesta1, Brian McFee2. Multiple f0 estimation in vocal ensemble using convolutional neural networks. 2020.

[7] Kian Sheik Kai Middlebrook. Song hit prediction : Predicting billboard hits using spotify data. 2019.

[8] Richard G. Lyons. *Understanding Digital Signal Processing*. Prentice Hall, 2010. Voir pages 234–237.

[9] Kevin McDonald. Marley, le film. `https://www.youtube.com/watch?v=GKN6oATJXiw&t=727s`, 2012.

[10] Lijia Wang Jienian Yang Zhaoxia Yu Myra Interiano, Kamyar Kazemi and Natalia L. Komarova. Musical trends and predictability of success in contemporary songs in and out of top charts. 2018.

[11] Sophocles J. Orfanidis. *Introduction to Signal Processing*. Prentice Hall, 1996. Voir pages 320–325.

[12] Brian McFee1 Rachel M. Bittner and Juan P. Bello. Multitask learning for fundamental frequency estimation in music. 2018.

[13] Humar Kahramanlı Örnek Semiye Demircan. Comparison of the effects of mel coefficients and spectrogram images via deep learning in emotion classification. 2020.

[14] Spotify. Liste des caractéristiques musicales de spotify. `https://developer.spotify.com/documentation/web-api/reference/get-audio-features`.

[15] Richard M. Warren. *Auditory Perception: A New Synthesis*. Cambridge University Press, 2008. Voir pages 34–36.

[16] Wikipedia. Liste des musiques les moins appréciées par la critique. `https://fr.wikipedia.org/wiki/Liste_des_musiques_les_moins_appr%C3%A9ci%C3%A9es_par_la_critique`.

[17] Raheem M. Yee YK. Predicting music popularity using spotify and youtube features. *Indian Journal of Science and Technology*, page 2, 2022.