Rapport de Projet AARES

Classification et Recommandation Musicale

[Faycal Raghibi, Guerouaoui Ilyas]

February 15, 2026

Abstract

Ce rapport présente notre travail sur le projet AARES qui explore la classification et la recommandation musicale à partir du jeu de données Spotify. Nous avons implémenté un classificateur Random Forest pour prédire les genres musicaux parmi 23 catégories, construit un modèle de régression pour prédire la popularité des chansons, analysé la distribution des genres multiples, et développé un système de recommandation basé sur le contenu utilisant la similarité cosinus. Notre approche démontre comment l'apprentissage automatique peut extraire des motifs significatifs à partir de caractéristiques audio pour résoudre des problèmes pratiques de recherche d'information musicale.

Contents

1 Introduction

L'objectif de ce projet est d'appliquer des techniques d'apprentissage automatique aux données musicales extraites de l'API Spotify. Le jeu de données contient diverses caractéristiques audio telles que l'acousticité, la dansabilité, l'énergie et le tempo, ainsi que des métadonnées comme l'année de sortie et les étiquettes de genre. Ces caractéristiques constituent la base de trois tâches principales : classer les chansons par genre, prédire les scores de popularité et recommander des morceaux similaires.

Le projet est divisé en plusieurs exercices. L'exercice 1 se concentre sur l'entraînement d'un classificateur pour prédire le genre des chansons d'un ensemble de test. L'exercice 2 passe à la régression pour la prédiction de popularité et inclut une analyse exploratoire des données multi-genres. L'exercice 3 implémente un système de recommandation basé sur le contenu qui trouve des chansons similaires en fonction de leurs caractéristiques audio.

2 Exercice 1 : Challenge de Classification

Le premier exercice nous demande de classifier des chansons parmi 23 genres en utilisant un ensemble d'entraînement de 25 492 chansons. L'ensemble de test contient 2 833 chansons sans étiquettes de genre, et nos prédictions sont évaluées à l'aide du score F1 avec moyenne micro.

2.1 Analyse des Données

L'ensemble d'entraînement est massif, contenant 25 492 chansons réparties sur 23 genres musicaux. Lorsque nous analysons ces données, nous observons un mélange complexe de caractéristiques qui définissent l'identité d'une chanson.

Nous rencontrons une disparité numérique importante : des caractéristiques comme `energy` et `danceability` (plage 0–1) côtoient `duration_ms` (plage 200 000–300 000) et `tempo` (plage 50–150). Cette différence d'échelle pose un problème car sans normalisation, les caractéristiques à grande magnitude comme la durée en millisecondes pourraient dominer le processus d'apprentissage en « intimidant » les petites valeurs comme l'acousticité.

Le contexte musical est également capturé par l'inclusion de `key` (0–11) et `mode` (Majeur/Mineur) qui fournissent un profil harmonique, tandis que `speechiness` et `instrumentalness` aident à distinguer un couplet de rap d'une symphonie classique.

Le défi principal réside dans l'ambiguïté des frontières entre genres. Avec 23 genres, les limites sont floues (par exemple, « hip hop » vs « rap » ou « electro » vs « techno »). Cela nécessite un modèle capable de trouver des motifs non linéaires dans les données.

2.2 Entraînement du Classificateur

Dans `src/classification.py`, nous avons implémenté un classificateur **Random Forest**: il s'agit d'une méthode d'ensemble qui gère les données « bruitées » et les caractéristiques haute dimension mieux qu'un simple arbre de décision. Les Random Forests combinent plusieurs arbres de décision pour réduire le surapprentissage et améliorer la généralisation. Chaque arbre apprend des motifs différents, et leur agrégation produit des prédictions plus robustes.

2.2.1 Le Pipeline de Prétraitement

Nous avons utilisé un `ColumnTransformer` pour gérer simultanément différents types de données avec des transformations appropriées.

Pour les caractéristiques numériques, nous avons appliqué une imputation avec la médiane. Les valeurs numériques manquantes ont été remplies en utilisant la médiane, qui est plus robuste aux valeurs aberrantes que la moyenne. C'est crucial pour des caractéristiques comme `loudness`

et `tempo` qui peuvent contenir des valeurs extrêmes. Si une chanson a un tempo anormalement élevé ou faible, la médiane n'en sera pas affectée, contrairement à la moyenne qui serait faussée.

Ensuite, nous avons utilisé `StandardScaler` pour transformer les caractéristiques afin qu'elles aient une moyenne de 0 et une variance de 1. Cette standardisation empêche un grand nombre comme `duration_ms` de « dominer » un petit nombre comme `acousticness` pendant l'entraînement. La formule appliquée à chaque caractéristique x est :

$$z = \frac{x - \mu}{\sigma} \tag{1}$$

où μ est la moyenne et σ l'écart-type.

Pour les caractéristiques catégorielles, nous avons utilisé l'imputation par valeur la plus fréquente suivie d'un encodage one-hot. L'utilisation de `OneHotEncoder` pour la caractéristique `key` est essentielle. Les clés musicales sont nominales, pas ordinales. La clé 0 (Do) n'est pas « inférieure » à la clé 1 (Do#) ; ce sont simplement des centres tonaux différents. L'encodage one-hot crée 12 caractéristiques binaires, permettant au modèle d'apprendre des relations indépendantes pour chaque clé.

2.2.2 Équilibrage des Classes

Pour garantir que le modèle ne prédit pas simplement le genre le plus fréquent (comme « pop »), nous avons utilisé `class_weight='balanced'`. Ce paramètre ajuste automatiquement les poids de manière inversement proportionnelle aux fréquences des classes :

$$w_i = \frac{n\_samples}{n\_classes \times n\_samples_i} \tag{2}$$

où w_i est le poids pour la classe i, et $n\_samples_i$ est le nombre d'échantillons dans la classe i. Cela force le modèle à accorder autant d'attention aux genres rares qu'aux genres courants.

2.3 Validation et Évaluation

Nous avons effectué une validation croisée à 5 plis pendant l'entraînement. Cette technique divise les données d'entraînement en 5 parties, entraîne le modèle sur 4 parties et le teste sur la cinquième, en répétant ce processus 5 fois. Cela nous donne une estimation fiable des performances du modèle sur des données non vues.

La métrique d'évaluation est le score F1 avec moyenne micro. Cette métrique calcule le nombre total de vrais positifs, faux négatifs et faux positifs sur les 23 genres. Elle est particulièrement utile car elle donne un poids égal à chaque chanson individuelle, quel que soit le genre auquel elle appartient. Le F1 micro est calculé comme :

$$F1_{micro} = \frac{2 \times TP}{2 \times TP + FP + FN} \tag{3}$$

où TP, FP et FN sont les totaux agrégés sur tous les genres.

Après l'entraînement, nous avons généré les prédictions pour les 2 833 chansons de l'ensemble de test et les avons sauvegardées dans un fichier CSV pour la soumission au challenge.

3 Exercice 2 : Analyse des Données

Le deuxième exercice utilise un jeu de données différent où les chansons peuvent appartenir à plusieurs genres. Cela reflète la réalité des plateformes de streaming musical, où les étiquettes de genre sont souvent ambiguës et se chevauchent.

3.1 Prédiction de la « Popularité »

Dans `src/analysis.py`, nous sommes passés de la classification (catégories) à la régression (prédiction d'un score de 0 à 100). Nous avons implémenté un Random Forest Regressor pour cette tâche.

Le Random Forest Regressor fonctionne de manière similaire au classificateur, mais au lieu de voter pour une classe, les arbres prédisent des valeurs continues et la prédiction finale est la moyenne de tous les arbres. Nous avons utilisé 100 arbres avec toutes les caractéristiques numériques disponibles.

Nous avons évalué le modèle en utilisant l'erreur quadratique moyenne (MSE) et le score R^2. Le MSE nous indique l'erreur moyenne au carré entre les valeurs prédites et réelles de popularité. Le score R^2 indique la proportion de variance dans la popularité que le modèle explique. Un R^2 de 1,0 signifierait des prédictions parfaites, tandis qu'un R^2 de 0 signifierait que le modèle ne fait pas mieux qu'une simple moyenne.

3.1.1 Importance des Caractéristiques

L'une des sorties les plus précieuses des modèles Random Forest est le classement d'importance des caractéristiques. Le modèle a identifié quels traits déterminent la popularité. Par exemple, `year` est souvent un facteur majeur car les chansons plus récentes ont tendance à avoir un engagement plus élevé sur la plateforme. Cela s'explique par plusieurs facteurs : les plateformes de streaming ont tendance à promouvoir les nouvelles sorties, l'engagement des utilisateurs se concentre naturellement sur le contenu récent, et la croissance de la plateforme au fil du temps signifie que les chansons récentes ont un public potentiel plus large.

D'autres caractéristiques importantes incluent probablement `danceability`, `energy` et `loudness`. Les chansons optimisées pour les playlists et les réseaux sociaux (avec une dansabilité élevée et un son moderne) ont tendance à mieux performer. Cependant, il est important de noter que l'importance des caractéristiques doit être interprétée avec prudence. Une importance élevée n'implique pas nécessairement une relation causale directe. Par exemple, l'année pourrait être corrélée avec la qualité de production ou la taille de la base de fans d'un artiste, plutôt que de causer directement la popularité.

3.2 Gestion de la Classe « genres »

Dans le jeu de données de sous-ensemble, `genres` est stocké sous forme de chaîne de caractères représentant des listes (par exemple, `"['rock', 'pop']"`). Pour travailler avec ces données, nous devions d'abord les parser.

Nous avons utilisé `ast.literal_eval` pour convertir ces chaînes en véritables listes Python. Cette fonction évalue en toute sécurité les structures littérales Python, contrairement à `eval()` qui pourrait exécuter du code arbitraire et présenter un risque de sécurité. Après la conversion, nous avons pu analyser les genres individuels.

Nous avons « aplati » ces listes pour compter la fréquence de chaque genre. L'analyse a révélé que de nombreuses chansons appartiennent à plusieurs sous-genres, ce qui explique pourquoi la classification à genre unique dans l'exercice 1 est si difficile. Certaines chansons appartiennent véritablement à plusieurs styles, et les forcer dans une seule étiquette perd de l'information.

Nous avons visualisé les 20 genres les plus fréquents dans un graphique à barres (Figure 1), révélant une distribution en longue traîne où quelques genres sont très courants, mais de nombreux sous-genres spécifiques apparaissent rarement. Le rock domine largement avec plus de 800 occurrences, suivi de la pop et de l'EDM. La présence d'étiquettes hiérarchiques comme « rock », « alternative rock », « modern rock », et « classic rock » montre que la taxonomie des genres de Spotify est assez détaillée et capture des nuances stylistiques au sein de grandes catégories.

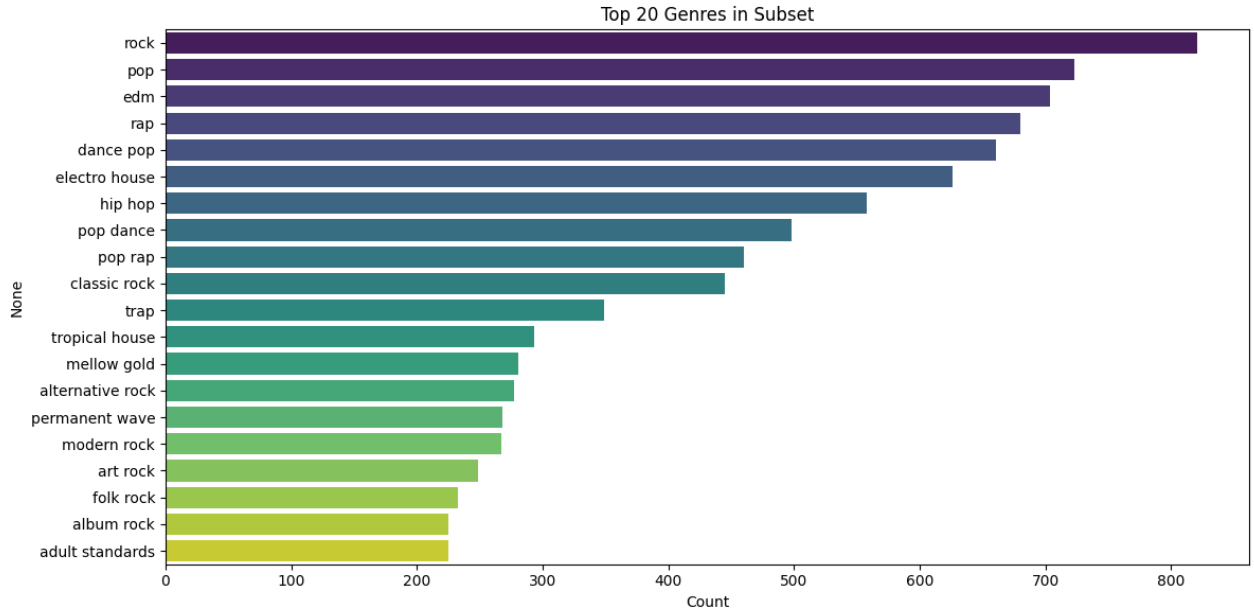

Figure 1: Distribution des 20 genres les plus fréquents dans le jeu de données de sous-ensemble. Le rock domine avec plus de 800 occurrences, révélant une distribution en longue traîne typique des taxonomies musicales.

3.3 Motifs et Visualisation

Pour visualiser l'espace des caractéristiques à haute dimension, nous avons utilisé l'Analyse en Composantes Principales (PCA) pour réduire nos 10 caractéristiques numériques à un espace 2D.

L'ACP trouve les directions de variance maximale dans les données. Les deux premières composantes principales capturent le plus d'information tout en permettant une visualisation en 2D. La formule mathématique implique le calcul des vecteurs propres de la matrice de covariance, mais conceptuellement, l'ACP projette les données sur de nouveaux axes qui maximisent la variance expliquée.

Le graphique de dispersion résultant (Figure 2) montre des clusters naturels. Les chansons avec une acousticité élevée et une énergie faible (comme le classique) dérivent naturellement vers un côté du graphique, tandis que les chansons à haute énergie et forte intensité (comme le metal ou l'EDM) se regroupent du côté opposé. Cette visualisation confirme que les caractéristiques audio contiennent effectivement des informations pertinentes pour le genre, même si les frontières ne sont pas parfaitement séparables.

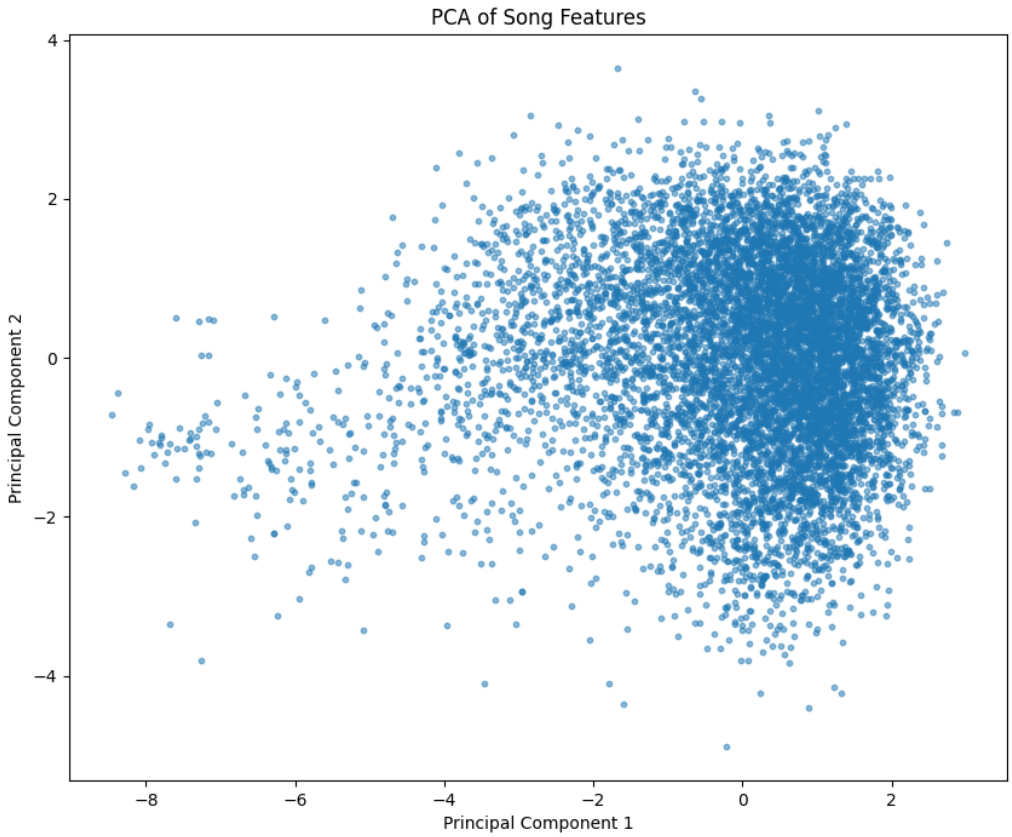

Figure 2: Visualisation PCA des caractéristiques audio des chansons. Les deux premières composantes principales capturent les directions de variance maximale, révélant une structure naturelle dans l'espace des caractéristiques.

L'interprétation de l'espace ACP suggère un axe acoustique vs électronique. D'un côté, nous avons des chansons avec forte acousticité et faible énergie (classique, folk, jazz, blues) caractérisées par des instruments organiques, une plage dynamique et un son plus doux. De l'autre côté, nous trouvons des chansons avec faible acousticité et haute énergie (metal, EDM, techno, trap) caractérisées par une production électronique, de la compression et un son agressif.

4 Exercice 3 : Recommandation de Chansons

Le troisième exercice nous demande de construire un système de recommandation qui suggère des chansons similaires basées sur une entrée donnée. Nous avons implémenté une approche de filtrage basé sur le contenu utilisant la similarité cosinus.

4.1 La Méthode de Proximité

Le système de recommandation dans `src/recommendation.py` transforme chaque chanson en une coordonnée mathématique (un vecteur). Notre classe `Recommender` construit une matrice de caractéristiques en utilisant neuf caractéristiques audio : acousticité, dansabilité, énergie, instrumentalité, vivacité, intensité, discours, tempo et valence.

Nous avons supprimé toutes les chansons avec des valeurs manquantes dans ces caractéristiques pour garantir l'intégrité des calculs. Ensuite, nous avons normalisé les caractéristiques en utilisant `StandardScaler`. Cette normalisation est cruciale car elle garantit que toutes les caractéristiques contribuent également au calcul de similarité. Sans cela, le tempo (qui varie de 50 à 150) dominerait des caractéristiques comme la valence (qui varie de 0 à 1).

Pour permettre des recherches rapides, nous avons créé des mappages entre les identifiants de chansons et leurs indices dans la matrice de caractéristiques. Cela nous permet de récupérer rapidement le vecteur de caractéristiques pour n'importe quelle chanson donnée.

4.2 Similarité Cosinus

Nous avons utilisé la similarité cosinus pour mesurer la distance entre les vecteurs. La formule pour la similarité entre deux chansons $\mathbf{s}_1$ et $\mathbf{s}_2$ est :

$$\text{similarité}(\mathbf{s}_1, \mathbf{s}_2) = \frac{\mathbf{s}_1 \cdot \mathbf{s}_2}{\|\mathbf{s}_1\|\|\mathbf{s}_2\|} = \frac{\sum_{i=1}^{9} s_{1i} \cdot s_{2i}}{\sqrt{\sum_{i=1}^{9} s_{1i}^2} \cdot \sqrt{\sum_{i=1}^{9} s_{2i}^2}} \tag{4}$$

Cette métrique mesure le cosinus de l'angle entre deux vecteurs, donnant des valeurs dans $[-1, 1]$ où 1 indique des profils de caractéristiques identiques et 0 indique une absence de corrélation.

Nous avons choisi la similarité cosinus plutôt que la distance euclidienne pour plusieurs raisons. Premièrement, elle mesure l'orientation des vecteurs plutôt que leur magnitude absolue. Cela signifie que deux chansons avec des caractéristiques proportionnelles sont considérées comme similaires même si l'une a des valeurs absolues plus grandes. Deuxièmement, elle gère naturellement les différentes échelles de caractéristiques après normalisation. Troisièmement, elle peut être calculée efficacement en utilisant des opérations matricielles. Enfin, le score de similarité correspond directement à l'alignement des profils des deux chansons, offrant une interprétation intuitive.

4.3 Le Résultat

Étant donné une chanson « seed », le système balaye la base de données et retourne les 10 chansons les plus similaires en fonction de leur « vibe » technique. L'algorithme calcule la similarité cosinus entre le vecteur de la chanson seed et tous les autres vecteurs, trie les résultats par ordre décroissant, exclut la chanson seed elle-même, et retourne les 10 meilleurs résultats avec leurs scores de similarité.

Les chansons recommandées partagent des caractéristiques similaires : une énergie et une dansabilité comparables pour une ambiance cohérente, une acousticité similaire pour un style de production cohérent, et une valence (positivité/négativité) apparentée pour un ton émotionnel similaire.

4.4 Extension à l'Expérience Utilisateur

Pour aller au-delà de la simple similarité de contenu, nous devons examiner le filtrage collaboratif. La logique est la suivante : si l'utilisateur A aime les chansons 1 et 2, et que l'utilisateur B aime la chanson 1, nous pouvons recommander la chanson 2 à l'utilisateur B même si les chansons 1 et 2 ne sont pas techniquement similaires.

Un système hybride combinerait notre similarité de caractéristiques techniques avec l'historique d'écoute des utilisateurs pour créer des recommandations à la fois musicalement précises et socialement pertinentes. Le filtrage collaboratif basé sur les utilisateurs suit le principe : « Les utilisateurs qui étaient d'accord dans le passé seront d'accord à l'avenir. »

Pour implémenter cela, nous construirions une matrice d'interaction utilisateur-chanson (par exemple, le nombre d'écoutes, les likes). Pour un utilisateur cible, nous trouverions des utilisateurs similaires basés sur leur historique d'écoute et recommanderions les chansons que les utilisateurs similaires ont aimées mais que l'utilisateur cible n'a pas encore entendues.

L'insight clé est que les chansons 3 et 5 dans notre exemple pourraient ne pas être soniquement similaires (elles pourraient être de genres complètement différents), mais elles plaisent aux utilisateurs ayant des préférences qui se chevauchent. Un système hybride offrirait le meilleur des deux approches. La composante basée sur le contenu garantit que les recommandations sont musicalement cohérentes, tandis que la composante collaborative introduit la sérendipité et exploite la sagesse collective.