PyTune
Rapport de « projet file rouge »

DataScientest promo DS septembre 2021

Jiayang GAO - Yann HUET 2022

Table des matières

1. Introduction 3
2. Description du jeu de données 3
3. Audit des données 4
4. Exploration des données 4
A. Analyse temporelle 4
B. Analyse des caractéristiques musicales 7
C. Analyse de l'influence culturelle 11
5. Changement de jeu de données 14
6. Train test split 15
7. Choix des métriques 16
8. Les différents types de système de recommandations 18
9. Explorations des modèles 18
a. Modèles content-based 19
i. User based 20
ii. Item based 21
iii. Avantages et inconvénients du Content-based 22
b. Modèles collaborative filtering 23
i. Surprise KNN 24
ii. Surprise SVD 25
iii. Implicite ALS 27
iv. Avantages et inconvénients du collaborative filtering 28
c. Les modèles hybrides 28
10. Comparaison des modèles 29
11. Conclusion 30

Figure 1 : Distribution des événements d'écoute groupés par jour suivant le jour de la semaine 4
Figure 2 : Distribution des événements d'écoute groupés par jour suivant l'heure de la journée 5
Figure 3 : Influence du jour de la semaine sur la danceability de la musique 6
Figure 4 : Corrélation des caractéristiques continues 7
Figure 5 : Vue d'ensemble des distributions des variables quantitatives et des relations entre elles 8
Figure 6 : Boxplot et distribution de la variable intrumentalness 9
Figure 7 : Relation entre « energy » et « acousticness » (gauche) et relation entre « danceability » et « valence » (droite) 9
Figure 8 : Relation entre « loudness » et « energy » 10
Figure 9 : Boxplot de la variable tempo en fonction de la Key 11

Figure 10 : Niveau d'acousticness répartis par mode 11
Figure 11 : Influence de la langue de tweet sur la tonalité de la musique 12
Figure 12 : Influence de la culture sur le tempo de la musique 13
Figure 13 : Train test split usuel 15
Figure 14 : Train test split par masque 15
Figure 15 : Les types de systèmes de recommandation 18
Figure 16: Schéma explicatif du content based filtering 19
Figure 17 : User based (modèle 1) 20
Figure 18 : Item based (modèle 2) 21
Figure 19: Collaborative filtering User based 23
Figure 20 : Item based collaborative filtering 24
Figure 21 : Modèle à factorisation de matrices 26

Tableau 1 : Distribution de la variable instrumentalness 9
Tableau 2 : Évaluation du modèle Content based user 21
Tableau 3 : Évaluation du modèle Content based item 22
Tableau 4 : Évaluation du modèle KNN 25
Tableau 5 : Évaluation du modèle SVD 26
Tableau 6 : Évaluation du modèle Implicit ALS 27
Tableau 7 : Évaluation du modèle LightFM 29
Tableau 8 : Récapitulatif des scores obtenus pour les différents modèles explorés 29

1. Introduction

Dans le cadre de notre formation de « Data Scientist » nous réalisons un projet fil rouge qui matérialise l'ensemble des méthodes apprises au cours des neuf mois de la formation. Notre projet s'intitule « *Recommandations musicales — prédire les musiques les plus appréciées sur Twitter* ».

Le but ici sera de prédire le degré d'appréciation d'une musique à partir d'événements d'écoute. Pour cela, nous disposons d'un triplet de jeu de données provenant du concours Kaagle Nowplayingrs Datasets :

https://www.kaggle.com/chelseapower/nowplayingrs?select=user_track_hashtag_timestamp.csv

Ce rapport reprend les différentes étapes exécutées, les axes de recherches suivis et les méthodologies utilisées pour tenter de répondre à la problématique. Il se compose de trois grandes parties. La première partie présente une exploration des données, dans un second temps nous procédons au nettoyage et à la préparation des données. Enfin, la troisième partie est consacrée à la création et à l'entrainement des modèles.

2. Description du jeu de données

Le dataset nowplayign-RS contient à la fois des caractéristiques contextuelles et de contenu pour chaque événement d'écoute. Il est composé de 11,6 M d'événements d'écoute par 139 K utilisateurs sur 346 k morceaux de musique. Ce dataset regroupe un large panel « d'item content features » et de « user context features ». De plus, certaines caractéristiques contextuelles nous renseignent sur les origines culturelles des utilisateurs et d'autres variables, comme les hashtags, fournissent des indices sur l'état émotionnel des utilisateurs lors de l'écoute.

Les données disponibles sont réparties dans les trois jeux de données suivants.

usertrackhashtagtimestamp.csv
Ce fichier contient les informations de chaque événement d'écoute. Pour chaque événement nous disposons des informations suivantes : id, userid, trackid, hashtag, created at.

contextcontentfeatures.csv
Ce fichier contient toutes les caractéristiques contextuelles et de contenu de chaque événement d'écoute. Pour chaque événement nous disposons des variables suivantes : user*id, track*id, artist*id, content features regarding the track mentioned in the event (instrumentalness, liveness, speechiness, danceability, valence, loudness, tempo, acousticness, energy, mode, key) and context features regarding the listening event (coordinates, place, geo, tweet*language, created *at, user*lang, time_zone, entities contained in the tweet).

sentiment_values.csv
Ce fichier contient les informations pour chaque hashtag ainsi que les valeurs de sentiment obtenues par différents dictionnaires de notation à savoir AFINN, Opinion Lexicon, Sentistrength Lexicon et vader.
Pour chaque dictionnaire, les éléments suivants sont fournis : le minimum, maximum, la somme et la moyenne chacun des token du hashtag. Cependant, comme beaucoup de hashtags ne sont composés

que d'un seul mot, l'ensemble de ces valeurs sont identiques. On notera que ces dictionnaires sont très différents et donnent des valeurs de sentiment très variées les unes des autres.

3. Audit des données

Nous avons réalisé un audit sur les trois jeux de données. Le résultat est présenté dans le fichier Excel fourni en annexe.

4. Exploration des données

Nous avons organisé cette première exploration des données autour de quelques questions auxquelles nous cherchons à répondre.

A. Analyse temporelle

Dans un premier temps, nous allons réaliser une analyse temporelle des données. En particulier, nous chercherons à déterminer si, durant l'année couverte par le Dataset, la fréquence des événements d'écoute varie suivant le jour de la semaine ou l'heure de la journée.

Question : Écoute-t-on plus de musique le weekend ?

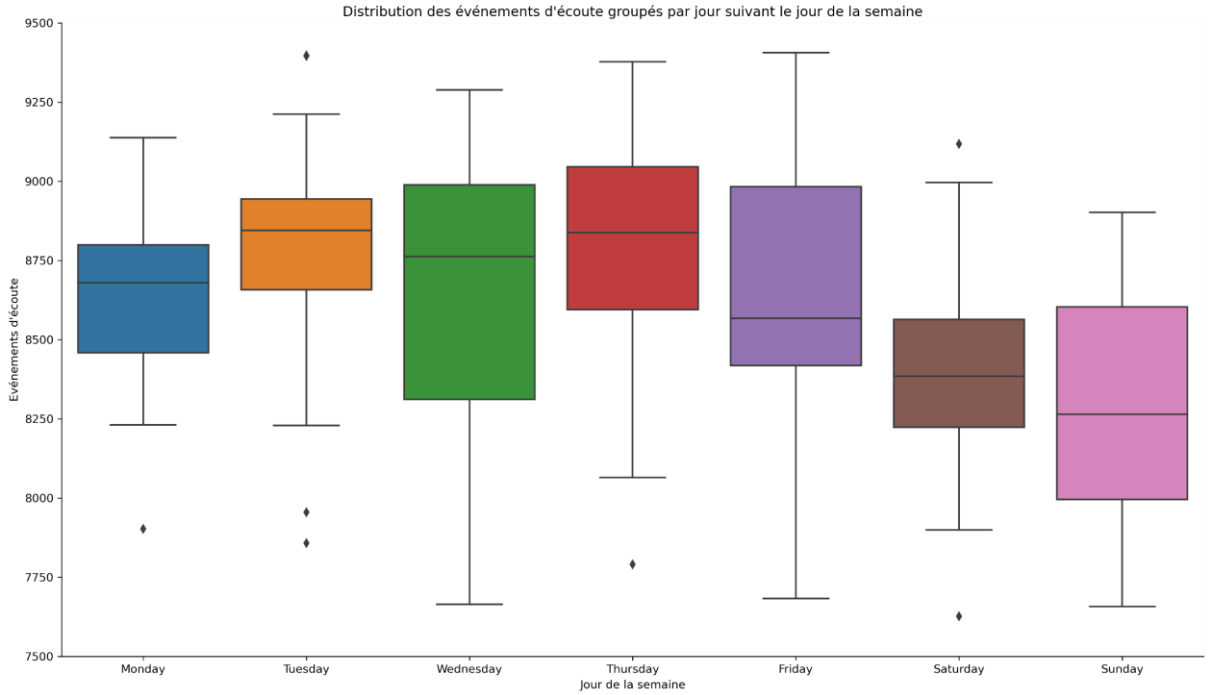


Figure 1 : Distribution des événements d'écoute groupés par jour suivant le jour de la semaine

La Figure 1 nous montre une réelle variabilité de la fréquence quotidienne d'écoute, avec une fréquence plus faible le weekend. Ceci va à l'encontre de notre hypothèse de travail. On note également une plus forte variance de la fréquence d'écoute les mercredis et le vendredi. Bien que visuellement le nombre d'évènements d'écoute et le jour de la semaine semblent corrélés, le test statistique semble confirmer l'hypothèse H0 et donc statuer sur l'indépendance de ces deux variables. Ce résultat est surprenant. On notera quand même un mean_sq très élevé (cf. test ci-dessous).

	df	sum_sq	mean_sq	F	PR(>F)
Week_day	6.0	1.881174e+07	3.135290e+06	1.41522	0.208386
Residual	296.0	6.557608e+08	2.215408e+06	NaN	NaN

Question : Quelle est l'évolution de la fréquence des événements d'écoute au cours de la journée ?

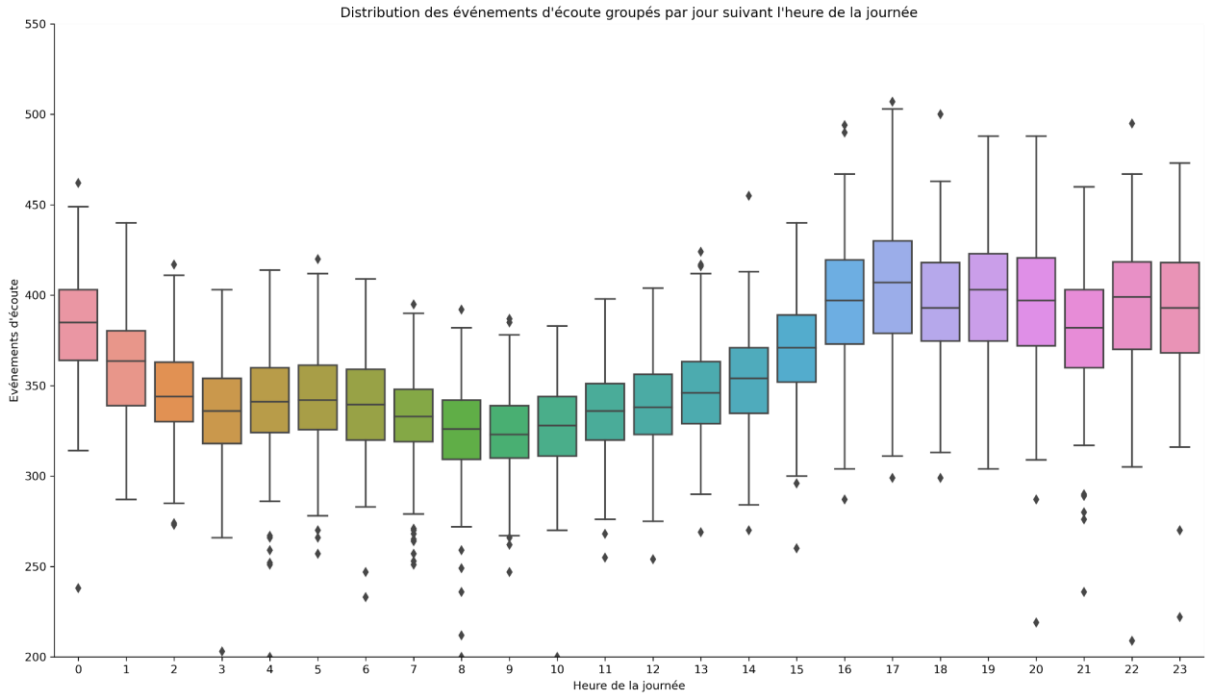


Figure 2 : Distribution des événements d'écoute groupés par jour suivant l'heure de la journée

La Figure 2 nous montre que le nombre d'événements d'écoute varie suivant l'heure de la journée. On note en moyenne plus d'événements d'écoute en fin de journée de 15 h à 0 h. La variance reste en revanche plutôt stable au cours de la journée. En accord avec notre observation concernant l'évolution de la fréquence moyenne d'écoutes, le test statistique d'indépendance entre l'heure de la journée et le nombre d'écoutes confirme bien le rejet de l'hypothèse H0 : les deux variables sont donc dépendantes (cf. test ci-dessous).

	df	sum_sq	mean_sq	F	PR(>F)
listening_h	1.0	2.184172e+06	2.184172e+06	1308.243786	1.437158e-262
Residual	6929.0	1.156828e+07	1.669545e+03	NaN	NaN

Question : Ecoute-t-on davantage de musique dansante (forte danceability) le weekend ?

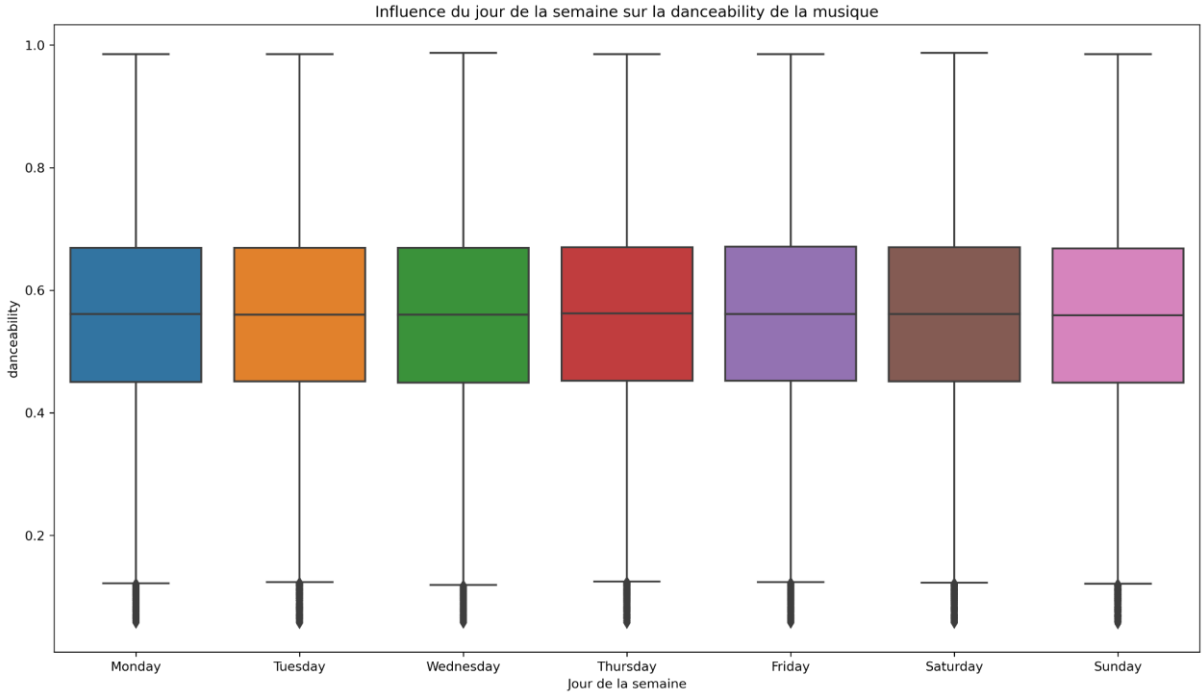


Figure 3 : Influence du jour de la semaine sur la danceability de la musique

La Figure 3 suggère qu'on n'écoute pas plus de musique dansante le weekend qu'en semaine. Pour autant, le test statistique d'indépendance entre la variable « danceability » et le jour de la semaine nous conduit à rejeter l'hypothèse H0. Ces deux variables sont donc en fait dépendantes. On remarque un très faible mean_sq (cf. le test ci-dessous).

	df	sum_sq	mean_sq	F	PR(>F)
Week_day	6.0	1.938932	0.323155	12.670875	2.361665e-14
Residual	2492205.0	63560.666897	0.025504	NaN	NaN

De la même manière que pour la « danceability », nous avons testé la relation de chaque caractéristique liée à la musique sans remarquer de corrélation avec les jours de la semaine.

B. Analyse des caractéristiques musicales

Dans un second temps, nous réalisons une analyse plus ciblée des relations entre les différentes caractéristiques des morceaux de musique. On cherche à mettre en évidence les corrélations existantes entre chacune d'entre elles.

Comme le Dataset est composé d'environ 11,6 millions de lignes, dans un souci de performance, les différentes analyses de cette section ont étés réalisées sur un échantillon de 1000 observations.

Question : Quelles sont les corrélations entre les caractéristiques musicales ?

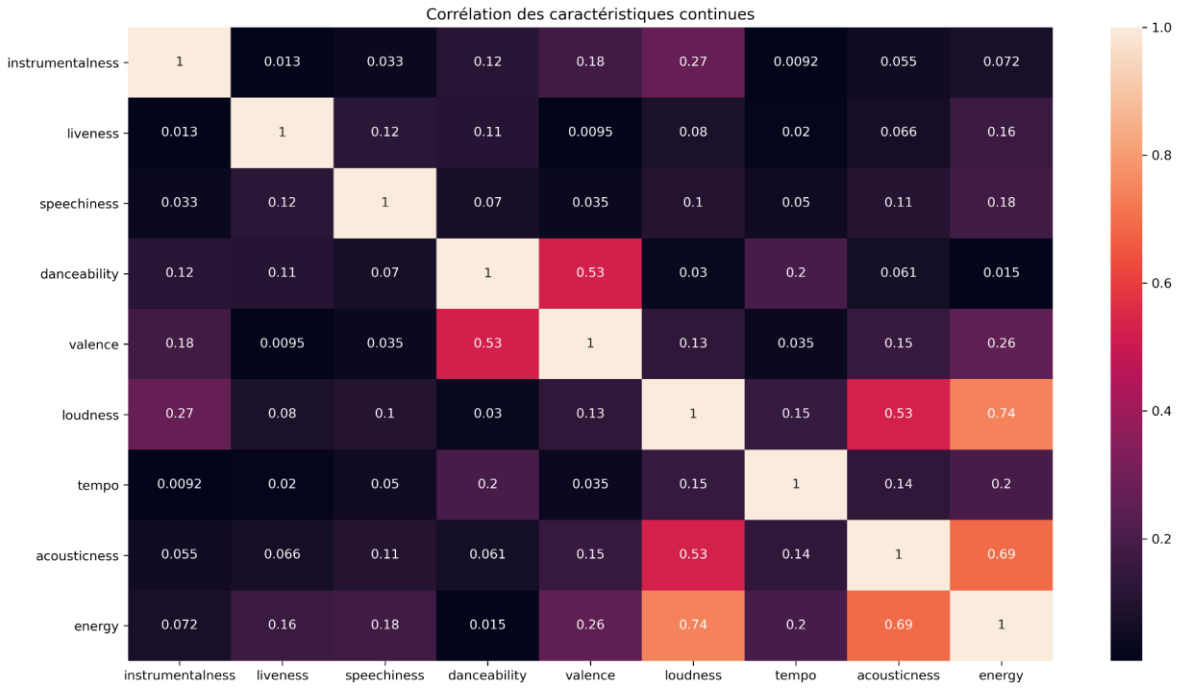


Figure 4 : Corrélation des caractéristiques continues

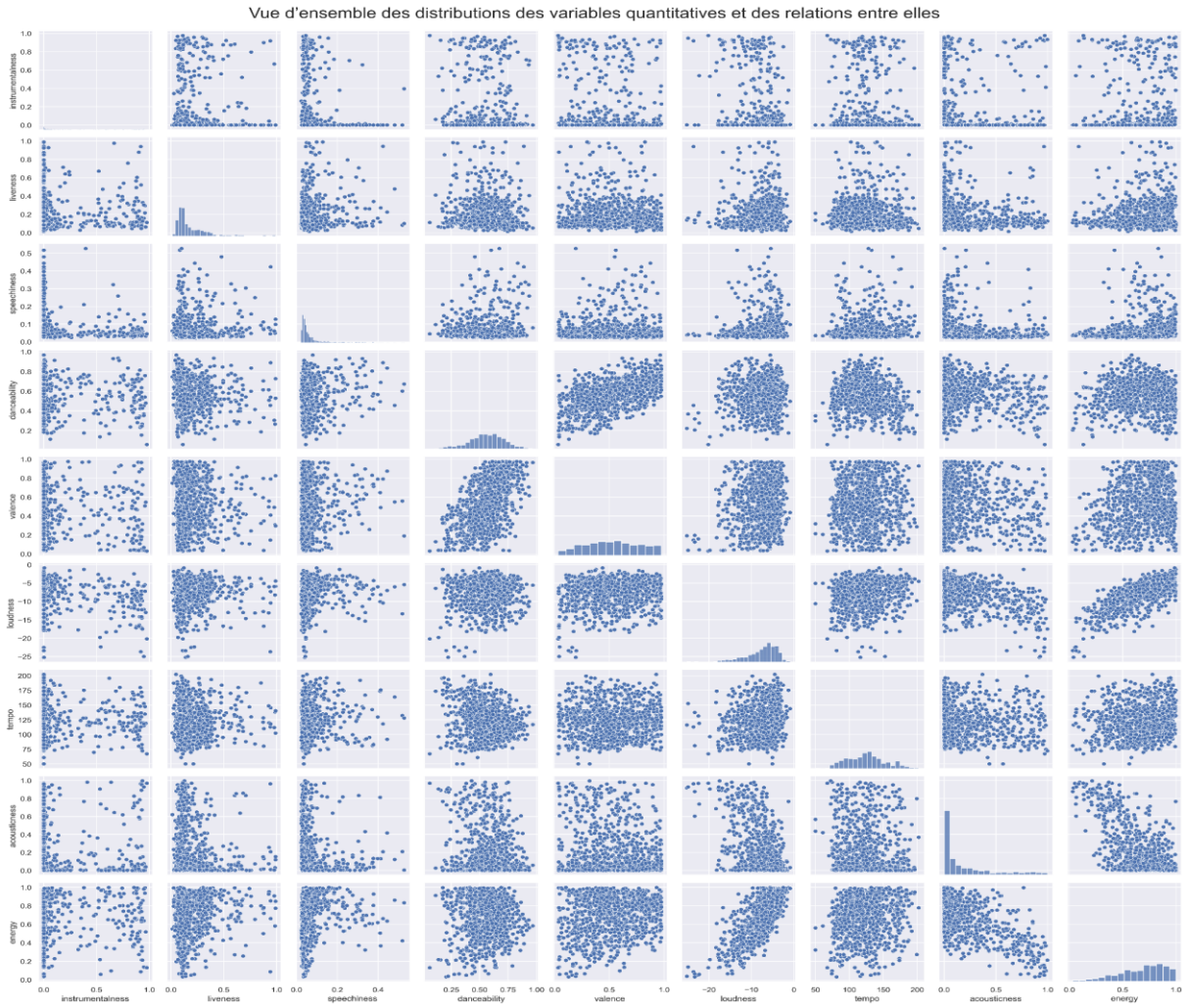


Figure 5 : Vue d'ensemble des distributions des variables quantitatives et des relations entre elles

Les Figure 4 et Figure 5 révèlent les points suivants :

- La distribution de la variable « instrumentalness » n'est pas visible, on en réalise donc un graphique spécifique (Figure 6)

- Les variables « danceability » et « valence » semblent corrélées, de même pour « loudness » et « acousticness », « loudness » et « energy ». On en réalise également des graphiques spécifiques (Figure 7 et Figure 8)

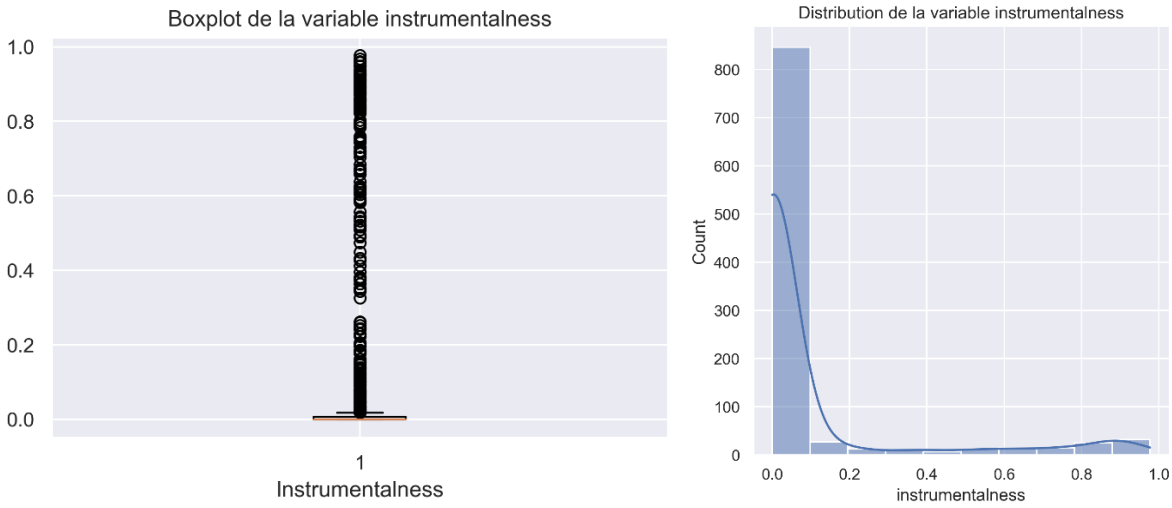


Figure 6 : Boxplot et distribution de la variable intrumentalness

On constate sur la Figure 6 que les valeurs de la variable « instrumentalness » sont majoritairement concentrées dans le voisinage de 0, c'est-à-dire que majorité des musiques contiennent des voix. En effet, le tableau 1 révèle que 75% des valeurs de « instrumentalness » sont comprises entre 0 et 0.105.

count	mean	std	min	25 %	50 %	75 %	max
11 603 890	0,096 467 58	0,238 675 6	0	0	0,000 055 6	0,010 5	0,999

Tableau 1 : Distribution de la variable instrumentalness

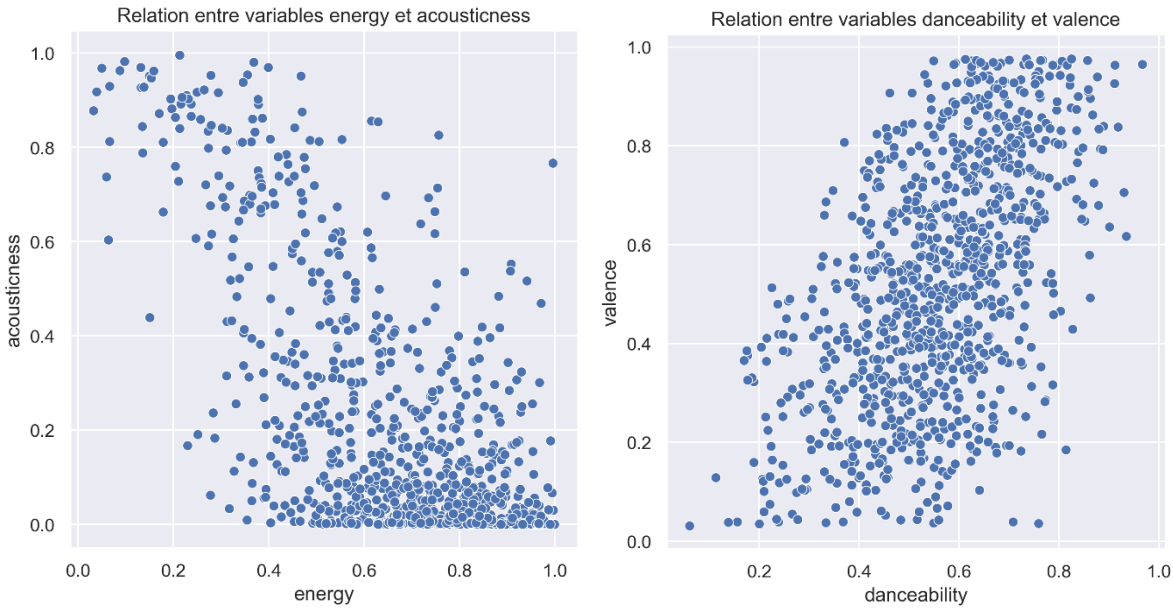


Figure 7 : Relation entre « energy » et « acousticness » (gauche) et relation entre « danceability » et « valence » (droite)

La figure 7 (gauche) suggère l'existence d'une corrélation négative entre les variables « acousticness » et « energy », tandis que sa partie droite révèle une corrélation positive entre « danceability » et

« valence ». Nous testons ci-après ces corrélations à l'aide des tests de Person et de Spearman. L'hypothèse H0 est la suivante : les deux variables testées sont indépendantes.

Test de corrélation entre danceability et valence :

Résultat Person : p-value=0, les deux variables ne sont pas indépendantes, coefficient de corrélation Person =0,524.

Résultat Spearman : p-value=0, les deux variables ne sont pas indépendantes, coefficient de corrélation Spearman=0.521.

Test de corrélation entre energy et acousticness :

Résultat Person : p-value=0, les deux variables ne sont pas indépendantes, coefficient de corrélation Person =-0,518

Résultat Spearman : p-value=0, les deux variables ne sont pas indépendantes, coefficient de corrélation Spearman=-0.473.

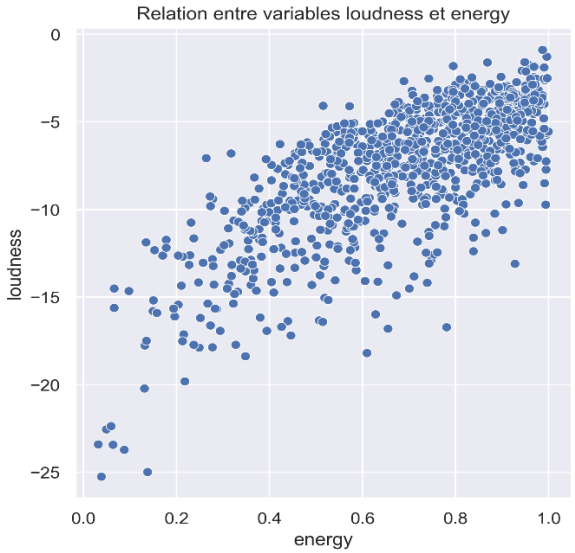


Figure 8 : Relation entre « loudness » et « energy »

Pour confirmer la corrélation entre les variables « energy » et « loudness », illustrée par la figure 8, nous leur appliquons également les tests de Person et de Spearman. L'hypothèse H0 est la suivante : les deux variables testées sont indépendantes.

Résultat Person : p-value = 0, les deux variables ne sont pas indépendantes, coefficient de corrélation Person = 0,736

Résultat Spearman : p-value = 0, les deux variables ne sont pas indépendantes, coefficient de corrélation Spearman = 0,707.

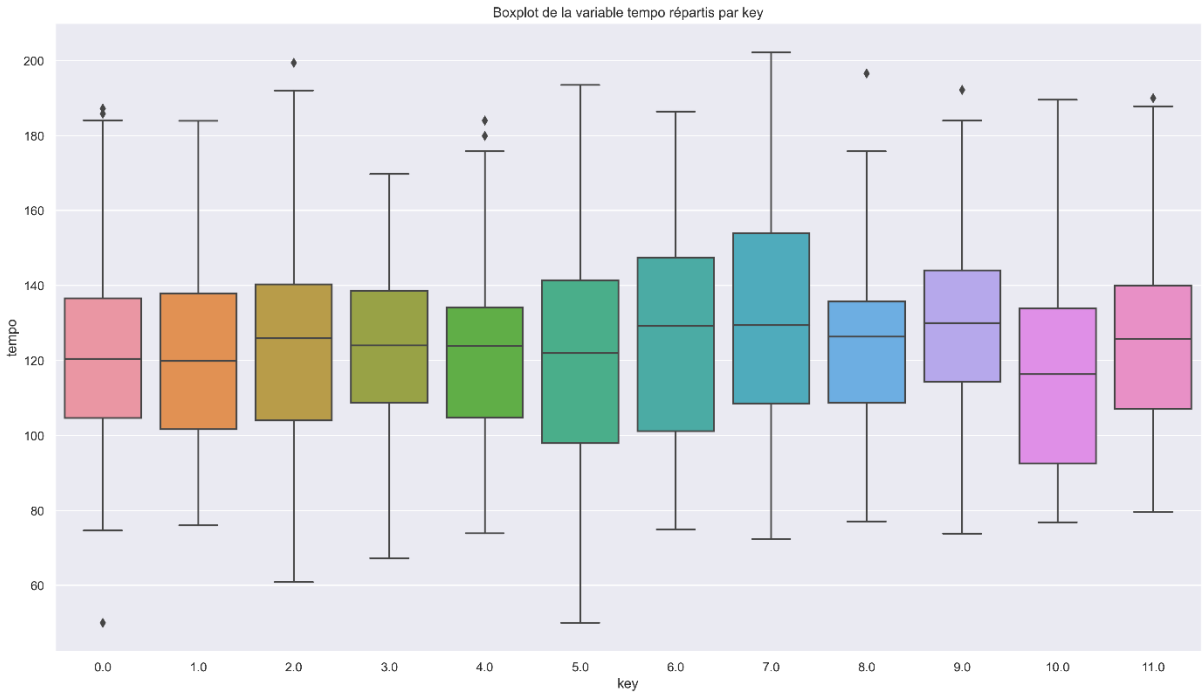


Figure 9 : Boxplot de la variable tempo en fonction de la Key

La Figure 9 révèle une légère variabilité du tempo suivant la tonalité. On notera un tempo moyen plus faible pour la Key=10 et un tempo moyen plus élevé pour la key=3.

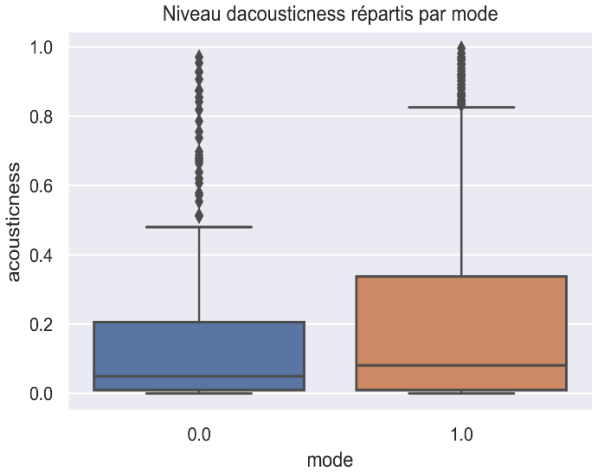


Figure 10 : Niveau d'acousticness répartis par mode

Nous avons testé la corrélation du « mode » avec les différentes caractéristiques. De ces tests, seule la variable « acousticness » montre une légère variabilité suivant le mode (Figure 10).

C. Analyse de l'influence culturelle

Le jeu de donnée incorpore des variables permettant de faire un lien entre la « culture » de l'utilisateur et le type de musique qu'il écoute. Nous allons ici explorer les relations existantes entre la culture et les différentes caractéristiques des morceaux de musique.

Question : La culture a-t-elle une influence sur les caractéristiques du morceau écouté ?

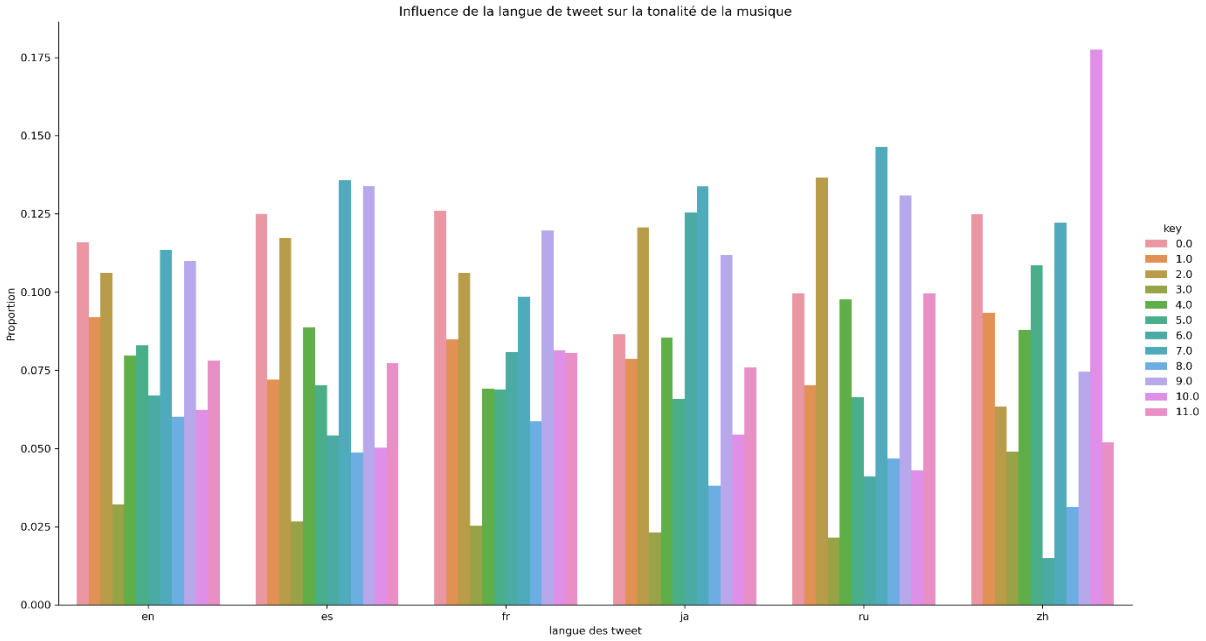


Figure 11 : Influence de la langue de tweet sur la tonalité de la musique

La Figure 11 montre une variabilité du ratio des différentes tonalités (key) présent dans les morceaux écoutés suivant la langue des tweets de l'utilisateur (ici, une sous sélection des langues en, es, fr, ja, ru et zh). On remarque entre autres que les utilisateurs chinois (= zh) écoutent plus de morceaux dans la tonalité key=10 que les autres langues sélectionnées. On notera aussi une forte similarité entre les cultures occidentales (en, es, fr).

La variable « Tweet_lang » comporte beaucoup de modalités. Pour plus de facilité, nous avons fait le choix de regrouper les différentes langues par culture pour tester plus facilement les relations entre caractéristiques des morceaux de musique et la culture.

Test du chi2 entre « Tweet_lang » et « key » :

- Statistique : 24 014,6
- P_value : 0,0
- Degré de liberté : 462

Les différentes langues ont étés regroupées suivant les cultures suivantes : 'US', 'EU_west', 'EU_est', ' Asia', 'Oceania', 'Arabic', 'Pacific', 'Persian', 'Hebrew'

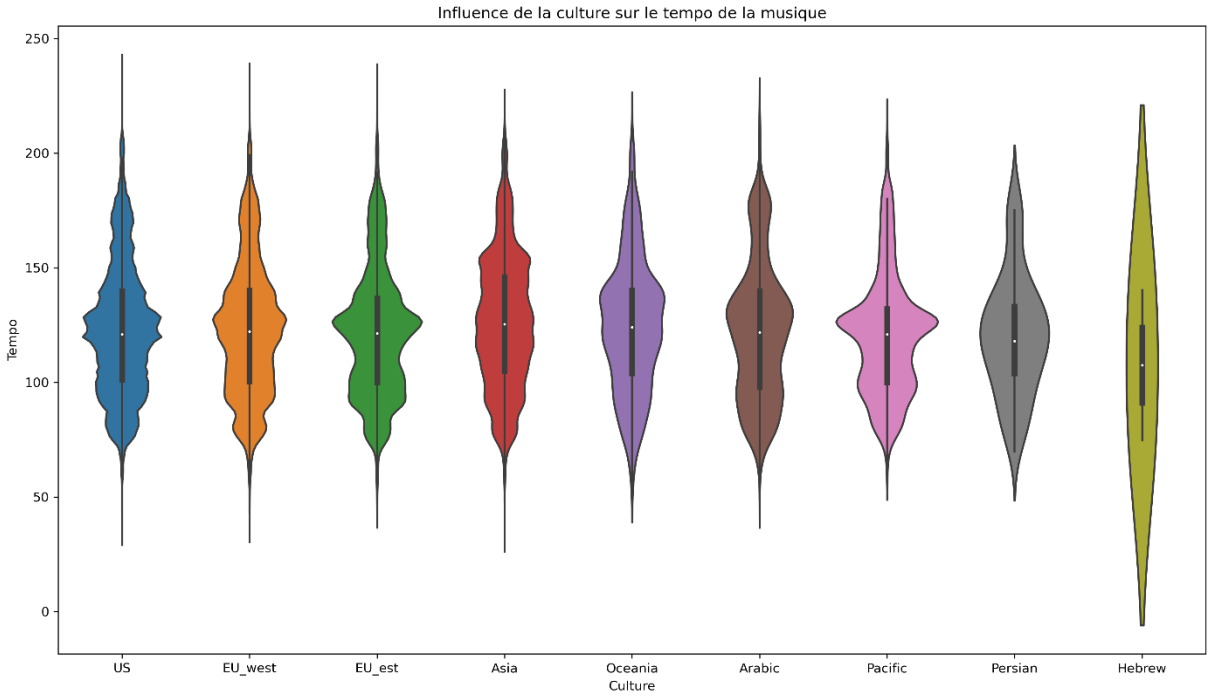


Figure 12 : Influence de la culture sur le tempo de la musique

La Figure 12 suggère que la distribution du tempo varie suivant la culture de l'utilisateur. On notera que les distributions du tempo sont similaires pour les cultures occidentales, sans pour autant être identiques. On peut noter des similitudes entre l'Asie et l'Océanie.

	df	sum_sq	mean_sq	F	PR(>F)
region	9.0	6.016005e+05	66844.503751	83.759613	1.932136e-156
Residual	2492202.0	1.988906e+09	798.051731	NaN	NaN

Le test statistique d'indépendance entre le tempo et la culture de l'utilisateur confirme bien le rejet de l'hypothèse H_0 et donc le rejet de l'indépendance de ces deux variables. On remarque un mean_sq très fort. De la même manière, l'ensemble des variables ('instrumentalness', 'liveness', 'speechiness', 'danceability', 'valence', 'loudness', 'tempo', 'acousticness', 'energy', 'mode', 'key') ont été testées. Elles ont toutes montré, à différents degrés, une variabilité suivant la culture de l'utilisateur. On en conclut que l'aspect culturel sera important dans l'élaboration de nos modèles.

5. Changement de jeu de données

Avec l'accord de notre superviseur, nous avons fait le choix de chercher un nouveau jeu de données pour mener à bien notre projet. En effet, le dataset de base n'ayant pas de méta data nous indiquant le titre et l'artiste de chaque chanson, l'interprétation de notre système de recommandation était impossible.

Après quelques difficultés, nous avons trouvé le jeu de données last.fm 1k, proposé par le MTG (Musique technologie groupe) de l'université Pompeu Fabra de Barcelone. Ce jeu regroupe les événements d'écoute pour 1000 utilisateurs sur une période d'un an (2008 à 2009), collectés à travers L'API de Last.fm.

Description du dataset :

Le jeu de données est composé de deux fichiers.

- **Le premier** « *userid-timestamp-artid-artname-traid-traname.tsv* » regroupe 190 millions d'intera ctions et renseigne les champs suivants : user_id, time_stamp, artist_id, artist_name, track_id, tra ck_name.

- **Le second** « *userid-profile.tsv* » donne des informations concernant les utilisateurs et renseigne le s champs suivants : user_id, gender, age, country, registered.

Filtrage du dataset :

Plusieurs traitements ont été appliqués sur ces jeux de données, à savoir :

- Une réinitialisation des id_user, id_artist, et id_track pour les homogénéiser vers des id de type int.
- Un filtrage des utilisateurs pour ne conserver que ceux ayant écouté au minimum 50 chansons différentes.
- Filtrage des titres musicaux pour ne garder que ceux écoutées par au moins 10 utilisateurs différents.

Le but est de réduire la taille du dataset à cause de la quantité limitée de RAM de nos ordinateurs personnels, tout en ne gardant que les interactions les plus pertinentes. Le dataset final comporte ainsi 104 millions d'interactions, 959 utilisateurs et 80 000 titres différents.

Ajout de données :

Ce nouveau dataset contient des métas datas (titre des morceaux musicaux et noms des artistes), mais il ne comporte pas de features descriptives pour chaque titre[1], ce qui est handicapant pour les modèles de type content based. De telles features sont présentes sur Spotify. Nous avons donc créé, via le programme Spotify for Developers, un robot capable de les récupérer. A noter que nous n'avons conservé que les titres pour lesquels le robot obtenait une correspondance parfaite entre le titre et l'artisite de Spotify et ceux de notre jeu de données.

[1] Il s'agit des caractéristiques déjà évoquées auparavant, à savoir : danceability, energy, key, loudness, mode, speechiness, acousticness, instrumentalness, liveness, valence et tempo

Après 800 000 requêtes, 10 heures et 3 bannissements, les features pour 45 000 titres ont été récupérées (sur les 80 000 morceaux de notre base de données).

6. Train test split

Comme tout modèle de machine Learning, nous avons besoin de savoir si le modèle que nous avons entraîné est pertinent sur de nouvelles données inconnues. Typiquement, ceci est fait à l'aide d'un jeu de test créé à parti d'une sélection aléatoire d'une sous partie du jeu de donnée original, comme illustré dans la Figure 13 ci-dessous.

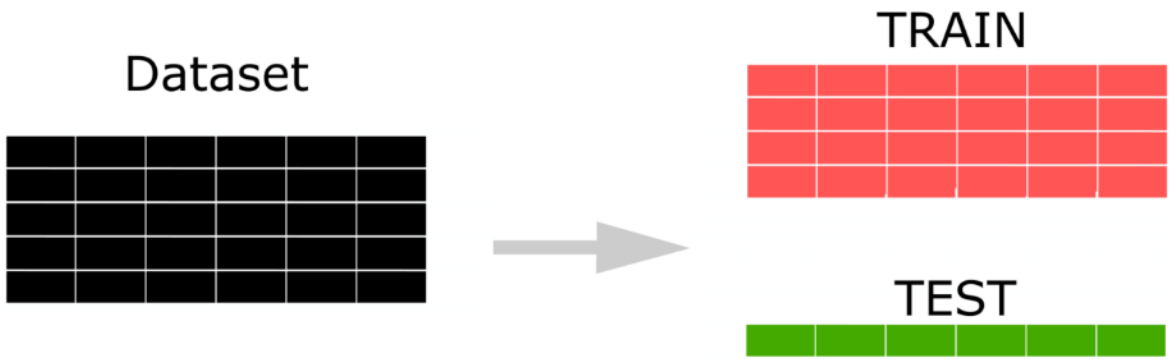


Figure 13 : Train test split usuel

Cependant, avec les modèles de type collaborative filtering, cette procédure usuelle ne marchera pas, car ces modèles nécessitent la présence de tous les users et items dans chacun des sous-ensembles de test et d'entrainement. L'idéal serait de faire un test dit A/B sur deux populations, ce qui n'est pas possible dans notre cas. Afin de séparer notre jeu de données en respectant cette contrainte, nous créons un masque cachant un pourcentage des interactions lors de l'entraînement (cf. Figure 14). Cette procédure garantie la présence de tous les items et de tous les utilisateurs dans les deux sous jeux de données. Ainsi, nous pouvons nous affranchir des problématiques de cold start.

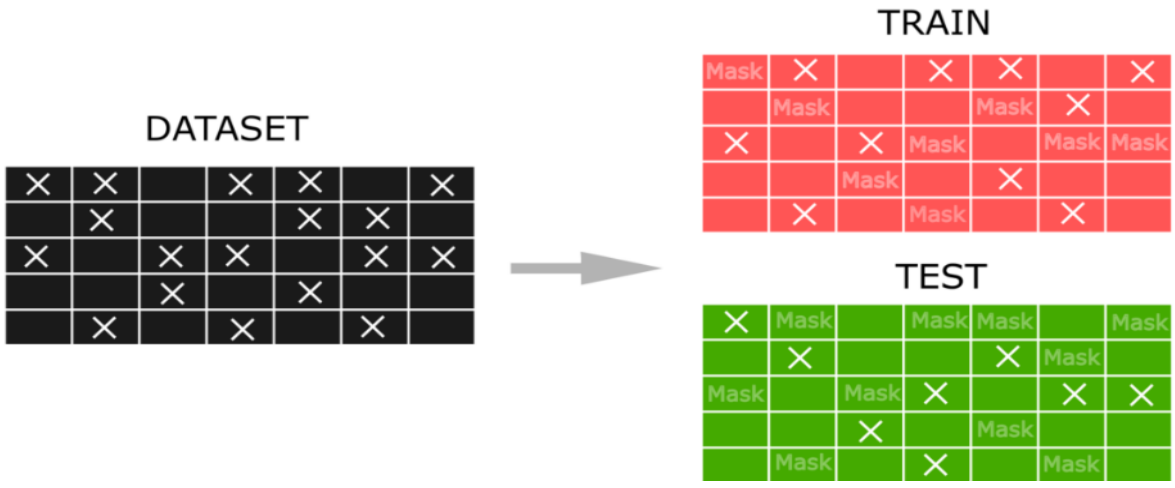


Figure 14 : Train test split par masque

Nos jeux de test et d'entraînement sont complémentaires, ils garantissent la présence de tous les utilisateurs et de tous les items, en évitant la présence d'interactions en double. Les masques sur le jeu d'entraînement nous permettent de simuler la non-écoute d'un titre.

Notre modèle devra à la fois être capable de recommander des titres déjà écoutés et d'autres non écoutés, mais présents dans le jeu de test.

7. Choix des métriques

Le choix de la métrique d'évaluation pour les systèmes de recommandation n'est pas trivial. En effet, les items recommandés doivent être proches des goûts de l'utilisateur, sans pour autant l'enfermer dans une bulle trop étroite, imperméable à la nouveauté. Une bonne métrique ou groupe de métriques devra donc être capable d'évaluer à quel point nous avons capturé les goûts musicaux d'un utilisateur tout en permettant l'introduction d'une certaine sérendipité dans nos recommandations.

La meilleure des options serait de réaliser un test A/B sur deux sous-ensembles de population parmi les utilisateurs d'un service. Malheureusement, nous n'avons pas de véritable service et encore moins une population d'utilisateurs réels. Nous allons donc séparer notre jeu de données en deux. Un jeu d'entraînement et un jeu de test composé d'interactions uniques au sous-jeu (cf train_test_split).

Les métriques MSE et MAE ne nous permettent pas d'atteindre nos objectifs pour plusieurs raisons. Premièrement, nos données ne comportent pas d'interaction explicite – c'est-à-dire qu'elles ne font pas intervenir de système de notation, mais des interactions implicites (nombre d'écoutes). Deuxièmement, nos données comportent beaucoup d'interactions manquantes, ce qui a pour effet de biaiser ces métriques avec une surreprésentation des interactions négatives.

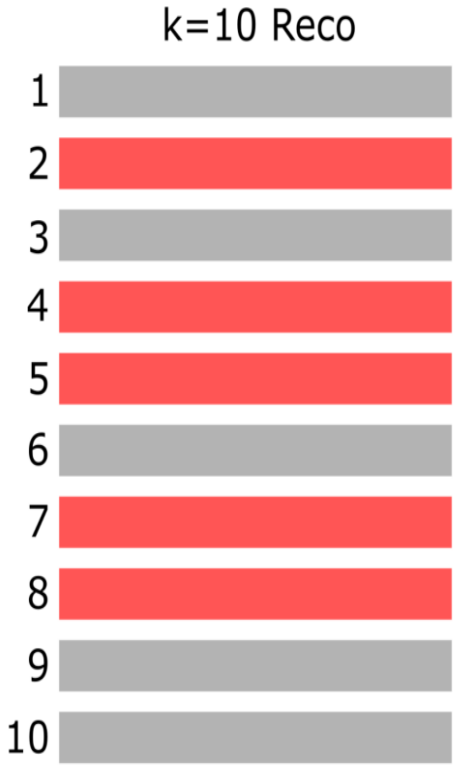
Une bonne métrique pour évaluer nos modèles sur l'ensemble des interactions est le score AUC. Ce score évalue le ratio entre les Vrais Positifs (recommandation écoutée par l'utilisateur) et les Faux Positifs (recommandation non écoutée). Ce ratio, compris entre 0 et 1, est ensuite transformé en nombre binaire selon la règle suivante : 0 si ratio <seuil ou 1 si ratio >=seuil.

Bien que l'AUC permette d'évaluer notre modèle sur l'ensemble des données, dans la pratique les utilisateurs n'interagissent bien souvent qu'avec les K premières recommandations. Nous chercherons donc à maximiser le nombre de recommandations pertinentes. Pour ce faire, nous utilisons le Cumulative Gain (CG), qui est la somme des recommandations pertinentes R (R=1 si écouté, 0 sinon) parmi les K recommandations.

Nous désirons en outre que ces recommandations pertinentes figurent le plus possible en tête des K recommandations (donc un rang i le plus faible possible). Dans cette perspective, nous utilisons le Discounted Cumulative Gain (DCG) en divisant chaque Ri par le Log base 2 de (1+i). Plus une recommandation pertinente est en queue de liste, plus son apport au gain est faible. Enfin, pour avoir un score compris entre 0 et 1, nous utilisons le Normalized Disconted Cumulative Gain (NDGC) - qui est le ratio entre le DGC et le DGC idéal (IDGC) représentant le DGC parfait où l'ensemble des recommandations pertinentes se trouveraient en tête de liste.

Le calcul de ces différentes métriques est résumé ci-dessous :

$$CG = \sum_{i}^{k} Ri \qquad DCG = \sum_{i}^{k} \frac{Ri}{\log2(1+i)} \qquad NDCG = \frac{DCG}{IDCG}$$



Exemple sur une recommandation avec K=10 (rouge = pertinentes, gris = non pertinentes) :

$$CG = 0 + 1 + 0 + 1 + 1 + 0 + 1 + 1 + 0 + 0$$

$$\boxed{CG = 5}$$

$$DCG = \frac{0}{\log2\,(1+1)} + \frac{1}{\log2\,(1+2)} + \cdots + \frac{0}{\log2\,(1+10)}$$

$$\boxed{DCG = 2.1}$$

$$IDCG = \frac{1}{\log2\,(1+1)} + \frac{1}{\log2\,(1+2)} + \cdots + \frac{1}{\log2\,(1+5)}$$

$$\boxed{IDCG = 2.95}$$

$$\boxed{NDGC = \frac{2.1}{2.95} = 0.71}$$

8. Les différents types de système de recommandations

« Les systèmes de recommandation sont une forme spécifique de filtrage de l'information visant à présenter les éléments d'information (films, musique, livres, news, images, pages Web, etc.) qui sont susceptibles d'intéresser l'utilisateur » (Wikipédia[2]). Autrement dit, il s'agit d'un algorithme qui suggère des éléments pertinents aux utilisateurs.

Il existe de nombreux exemples de systèmes de recommandation largement utilisés aujourd'hui. Ainsi, Netflix propose des films aux goûts des utilisateurs, Amazon suggère des produits aux acheteurs, les applications musicales comme Spotify ou Deezer recommandent des musiques. Ces systèmes peuvent être de plusieurs types – résumés dans la Figure 15. Chacun d'eux est présenté en détail et appliqué à notre jeu de données dans les sections suivantes du rapport.

Type de système de recommandation.

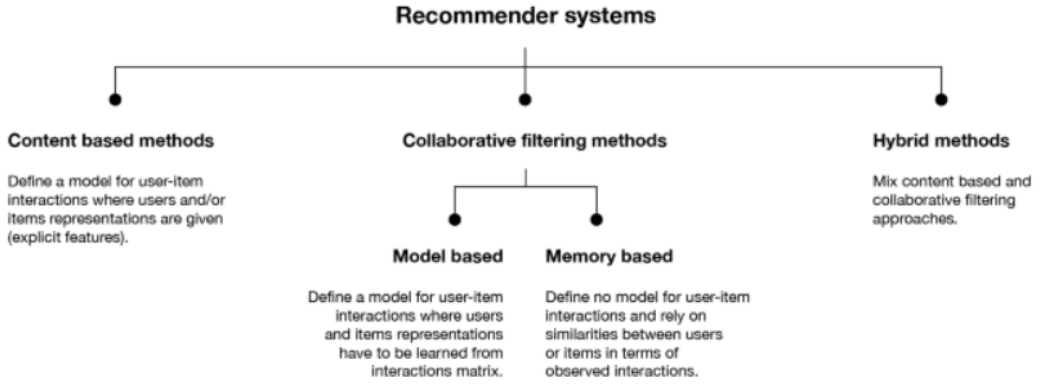


Figure 15 : Les types de systèmes de recommandation

9. Explorations des modèles

Pour chaque modèle exploré, nous présenterons succinctement la théorie sur laquelle il repose puis nous commenterons les différentes étapes d'entrainement ainsi que les résultats obtenus. L'évaluation se fera à travers les deux métriques retenues, l'AUC calculé sur l'ensemble du dataset et le NDGC calculé sur un ensemble de 50 recommandations. A chaque fois, le score final retenu est la moyenne des scores obtenue sur l'ensemble des utilisateurs.

La validité du modèle sera évaluée à partir de cinq utilisateurs factices dont les goûts sont très tranchés (Jazz, classique, pop, rock et rap). On notera la qualité des recommandations complètes (ensemble des items) et des recommandations filtrées (sous ensemble des items ne comprenant pas les items déjà rencontré par l'utilisateur).

[2] https://fr.wikipedia.org/wiki/Syst%C3%A8me_de_recommandation

a. Modèles content-based

Le content-based filtering est un type de système de recommandation qui tente de deviner ce qu'un utilisateur peut aimer en fonction de son activité historique, ceci en utilisant des mots-clés et les attributs des items (par exemple, des tags pour un film, les caractéristiques pour une musique, etc.) et en les faisant correspondre à un profil d'utilisateur. Le profil d'un utilisateur est créé par ses actions historiques, telles que des achats, des évaluations (j'aime et n'aime pas ou des notes de 1 à 5), des téléchargements, des écoutes de musiques, etc. Ce processus est représenté Figure 16 : un profil d'utilisateur est créé sur la base de lectures passées. Ces informations permettent au système de proposer en retours des ouvrages jugés similaires et donc susceptibles d'intéresser l'utilisateur.

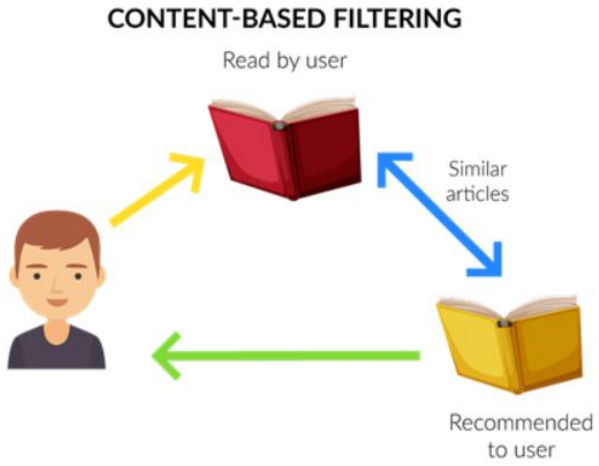


Figure 16: Schéma explicatif du content based filtering

Selon le type de données, il existe deux façons de réaliser le content-based filtering :

Modèle 1 : On commence par calculer la similarité entre le vecteur des préférences de l'utilisateur grâce à son historique d'actions. Chaque item étant défini par un vecteur de caractéristiques, le système propose les N items qui correspondent le mieux au vecteur des préférences (historiques) de l'utilisateur.

Modèle 2 : On commence par calculer les similarités entre les items. Lorsque l'utilisateur en sélectionne un, le système lui suggère ensuite les N items qui s'en rapprochent le plus. En quelque sorte, il s'agit ici d'un système de recommandation basé sur l'action présente, tandis que le modèle 1 recommande sur la base d'un historique d'actions. Le modèle 2 semble donc davantage pertinent face à des utilisateurs versatiles.

Ces deux modèles de content-based filtering sont présentés plus en détail et appliqués à notre jeu de données dans les sous-sections suivantes.

i. User based

Les étapes de ce modèle comme représenté dans la Figure 17 sont les suivantes

- Etape 1 : Taguer chaque item, construction d'un *item's features vector* (V_i)
- Etape 2 : Calculer un *user's preferred feature vector* (V_u) selon l'historique d'actions de l'utilisateur.
- Etape 3 : Calculer la similarité entres les vecteurs V_i et le vecteur V_u, puis proposer à l'utilisateur les N items les plus proches de ses goûts.

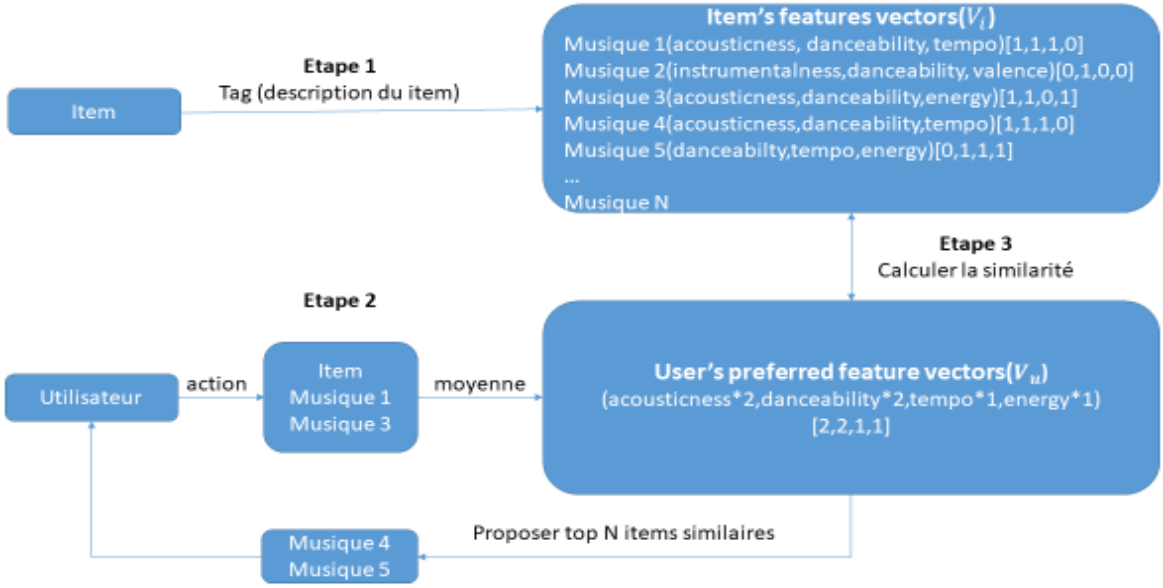


Figure 17 : User based (modèle 1)

Entraînement et résultat du modèle user-based

Les différentes étapes de traitement et les résultats du modèle sont disponibles dans les notebook suivants :

- **4.1 CONTENT BASED (model) :** entraînement et évaluation des modèles
- **4.2 content based (Fake_users):** test du modèle sur de faux utilisateurs

Pour ce modèle, nous avons fait le choix de définir le vecteur caractéristique de l'utilisateur comme la moyenne des features des items (titres) avec lesquels il a eu une interaction. La recommandation est faite en calculant, pour chaque utilisateur, la cosine similarity entre son vecteur caractéristique et les vecteurs item. Ne sont gardés que les 50 item ayant la cosine similarity la plus élevée. Les résultats de ce modèle sont présentés dans le Tableau 2 : Évaluation du modèle Content based user ci-dessous.

DATASET	Mean NDCG		Mean AUC	
	Train	Test	Train	Test
Last-fm 45k	0.240	0.138	0.537	0.533

Tableau 2 : Évaluation du modèle Content based user

Les résultats de ce modèle ne sont pas très bons. On s'y attendait. Même si L'AUC montre que le modèle est meilleur que le hasard (qui produirait un score de 0.5), les scores NDGC indiquent qu'il n'arrive pas à positionner beaucoup de recommandations pertinentes parmi les 50 meilleures prédictions.

Ces résultats se confirment lors de l'évaluation du modèle sur les cinq utilisateurs factices. En effet, sur l'ensemble des 10 recommandations réalisées, très peu - voire aucunes - ne semblent cohérentes avec leurs penchants musicaux.

ii. Item based

Les étapes de ce modèle comme représenté dans la Figure 18 sont les suivantes

- Etape 1 : Taguer chaque item, construction d'un *item's features vector* (V_i).
- Etape 2 : Calculer la similarité entre chaque V_i.
- Etape 3 : Lorsqu'un item est sélectionné par l'utilisateur, proposer les N autres items les plus similaires.

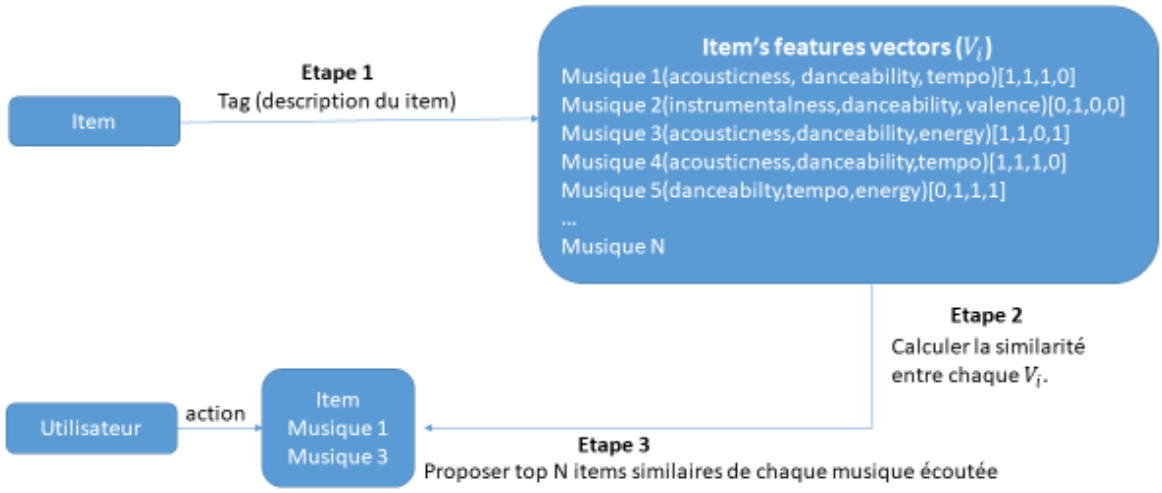


Figure 18 : Item based (modèle 2)

Entraînement et résultat du mode item-based

Les différentes étapes de traitement et les résultats du modèle sont disponibles dans les notebook suivants :

- **4.1 CONTENT BASED (model) :** entraînement et évaluation des modèles
- **4.2 content based (Fake_users):** test du modèle sur de faux utilisateurs

Pour des raisons de performances, la recommandation est faite sur la base des 10 items les plus écoutés pour chaque utilisateur. Pour chaque item dans ce top-10, on récupère les 50 items les plus

proches à l'aide de la cosine similarity, pour ne garder comme recommandation finale que les 50 meilleurs des 500 items obtenus. Les résultats de ce modèle sont présentés dans le Tableau 3 ci-dessous.

DATASET	Mean NDCG		Mean AUC	
	Train	Test	Train	Test
Last-fm 45k	0.227	0.129	0.5	0.50

Tableau 3 : Évaluation du modèle Content based item

Ce modèle produit des résultats encore plus mauvais que le modèle précédent. Ici, l'AUC montre clairement que le modèle n'est pratiquement pas meilleur que le hasard. De plus, les scores NDGC indiquent qu'il n'arrive pas à positionner beaucoup de recommandations pertinentes parmi les 50 meilleures prédictions.

Encore une fois, ces résultats se confirment lors de l'évaluation du modèle sur les cinq utilisateurs factices. Sur l'ensemble des 10 recommandations réalisées, très peu - voire aucune - ne semblent cohérentes avec les penchants musicaux des utilisateurs.

iii. Avantages et inconvénients du Content-based

Le content-based filtering présente un certain nombre d'avantages, mais aussi des limites. Ces dernières expliquent pourquoi il est utile de considérer d'autres types de systèmes de recommandation.

1. **Avantages :**
   - Le modèle n'a pas besoin des données des autres utilisateurs.
   - Le modèle peut être appliqué aisément à une base de données contenant un grand nombre d'utilisateurs.
   - Le modèle peut capter les intérêts spécifiques de l'utilisateur et peut lui recommander des éléments qui intéressent très peu d'autres utilisateurs.

2. **Inconvénients :**
   - La représentation des caractéristiques des items est conçue à la main dans une certaine mesure, ce qui nécessite beaucoup de connaissances dans le domaine considéré.
   - Le modèle ne peut faire que des recommandations basées sur l'intérêt existant d'un utilisateur. En d'autres termes, le modèle ne permet pas de proposer de la nouveauté.

Dans le cas présent, les deux modèles expérimentés présentent des performances médiocres. Nous allons rechercher une meilleure alternative.

b. Modèles collaborative filtering

Le collaborative filtering (CF) est basé sur l'hypothèse selon laquelle les individus aiment des choses similaires aux autres personnes qui leur ressemblent, on parle alors de User-Based Collaborative Filtering (UB-CF). Alternativement, le CF suppose que l'individu peut être intéressé par des produits similaires à ses achats ou centres d'intérêt, on parle alors de Item-Based Collaborative Filtering (IB-CF). Le CF utilise l'intelligence collective pour calculer ces similarités. Il permet de recommander des nouveaux items aux utilisateurs sur la base des interactions entre utilisateurs et items, sans avoir besoin d'indications relatives aux caractéristiques de ces items.

User-Based Collaborative Filtering

L'idée principale derrière UB-CF est que les personnes ayant des caractéristiques similaires partagent des goûts similaires. Par exemple, dans la figure ci-dessous, le tigre et le lion aiment tous les deux le même aliment (la cuisse de poulet), ce qui permet de supposer que le tigre aimerait aussi le morceau de viande, comme le lion. Cet aliment lui serait donc proposé via le UB-CF.

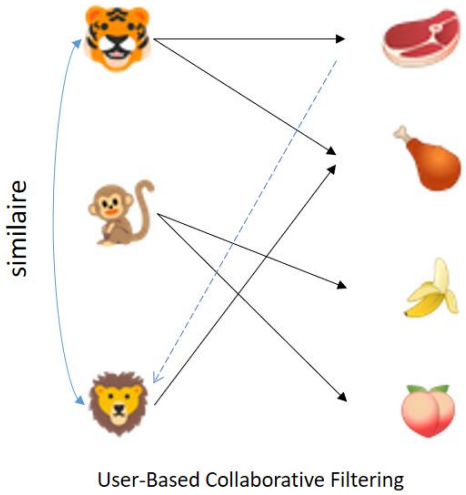


Figure 19: Collaborative filtering User based

Item-Based Collaborative Filtering

Le IB-CF, développé par Amazon en 1998, joue un rôle important dans son succès. L'idée de base derrière cette technique est de mesurer la similarité entre produits sur la base des articles que les utilisateurs ont déjà évalués/consommés. Ensuite, il s'agit de recommander des articles similaires aux consommateurs. La zone 'Customers who bought this item also bought…' sur le site d'Amazon est une application du item-based collaborative filtering (cf. Figure ci-dessous).

Figure 20 : Item based collaborative filtering

Une règle de décision simple permet de choisir quelle méthode utiliser entre le UB-CF et le IB-CF : si, dans la base de données, le nombre d'utilisateurs est inférieur au nombre d'items, alors il est préférable d'utiliser le modèle user-based pour réduire le temps de calcul. En revanche, si le nombre d'utilisateurs est supérieur au nombre d'items, alors on préfèrera le modèle item-based. Par exemple, la recommandation d'Amazon est basée sur les produits, parce qu'Amazon a beaucoup plus de clients que de produits.

Dans les entreprises, le IB-CF est généralement préféré, car l'approche alternative (user-based) est souvent plus difficile à industrialiser en raison de la nature dynamique des utilisateurs, alors que items (produits) ne changent généralement pas beaucoup. De plus, les modèles item-based présentent l'avantage de pouvoir être calculés hors ligne.

Dans notre cas, nous avons largement moins d'utilisateurs (1000) que d'items (80 000). Nous privilégierons donc les modèles UB_CF.

i. Surprise KNN

Le package SURPRISE impose de présenter les données sous forme de trois colonnes : *user_id, item_id* et *rating*. Notre jeu de données ne présente pas d'évaluation des musiques par les utilisateurs. Nous contournons ce problème en utilisant le nombre d'écoutes d'un morceau de musique par un utilisateur comme indicateur de 'rating'. L'hypothèse est qu'au plus un utilisateur écoute une musique, plus il l'apprécie.

Entraînement et résultat du modèle KNN

Les différentes étapes de traitement et les résultats du modèle sont disponibles dans les notebook suivants :

- **5.1 COLLABORATIVE FILTERING (Surprise KNN) :** entraînement et évaluation des modèles
- **5.2 COLLABORATIVE FILTERING (Surprise KNN Fake_users) :** test du modèle sur de faux utilisateurs

Pour ce modèle, nous avons créé une matrice d'interaction entre les utilisateurs et les items, prenant les valeurs de Rating, qui est fonction du nombre d'écoutes.

Afin d'entrainer un modèle KNNBasic (Module Surprise), nous avons effectué un GridSearch permettant de déterminer les paramètres optimaux. A noter que ce modèle avait tendance au sur-apprentissage. Les 50 meilleures recommandations ont alors été obtenues à l'aide d'une fonction proposée par le modèle. Les résultats de ce modèle sont présentés dans le Tableau 4 ci-dessous.

DATASET	Mean NDCG		Mean AUC	
	Train	Test	Train	Test
Last-fm 80k	0.583	0.492	0.545	0.498

Tableau 4 : Évaluation du modèle KNN

Les résultats sont mitigés. L'AUC indique un modèle plus mauvais que le hasard dans le cas des données de test. Il semblerait que le modèle n'ait pas réussi à généraliser. À l'inverse, les scores NDGC sont les meilleurs obtenus pour le moment.

Ces bons résultats – relatifs - sur le NDGC sont confirmés par les recommandations faites aux cinq utilisateurs factices. Pour chacun d'eux, le modèle a été capable de recommander des titres en accord avec leurs goûts spécifiques. Pour autant, un examen plus attentif révèle que les titres recommandés sont en fait des titres déjà écoutés par l'individu.

Dans le cas des recommandations filtrées (sans les items déjà écoutés), les résultats sont plus mitigés : les recommandations sont cohérentes pour l'utilisateur aimant le rap, mais elles manquent de pertinence dans le cas de celui aimant la musique classique.

Au total, les résultats obtenus suggèrent quelques limitations du surprise KNN :

- Problème de personnalisation : les musiques recommandées sont généralement des musiques très populaires.
- Problème de 'cold-start' : Des nouveaux morceaux de musiques avec peu ou sans interaction avec les utilisateurs ne rentrent pas dans les listes de recommandations.
- Problème d'évolutivité : Notre matrice ayant une grande dimension et une faible densité (matrice creuse, qui contient beaucoup de zéros), nous devons réduire fortement la taille de la base de données par soucis d'économie de mémoire vive. De fait, dans les entreprises, avec beaucoup plus de données, il n'est pas possible d'évaluer les modèles KNN pour cette raison.

ii. Surprise SVD

Les limitations des algorithmes KNNs imposent de rechercher une alternative. Nous explorons donc les algorithmes à base de factorisation matricielle. Lorsqu'on construit une matrice d'interaction user-item avec les vraies données d'entreprise, on obtient souvent une matrice extrêmement clairsemée avec près de 99% de valeurs manquantes.

La factorisation est une solution pour résoudre ce problème de matrice creuse. Les algorithmes de factorisation matricielle fonctionnent en décomposant l'énorme matrice creuse en un produit de deux matrices rectangulaires de dimensionnalité inférieure avec des facteurs latents tels que dans la Figure 21 ci-dessous.

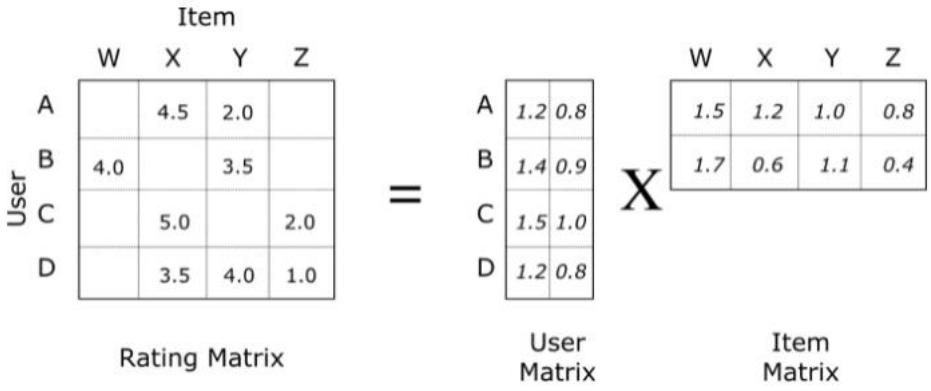


Figure 21 : Modèle à factorisation de matrices

En faisant une réduction de dimension de ce type, on induit une compression de notre matrice principale. Cette compression force la généralisation du modèle sur les données d'entrées.

Plus le nombre de facteurs utilisé est faible plus on généralise. Il faut donc trouver l'équilibre entre recommandations personnalisées et généralisation. Un modèle trop peu généralisé ne recommandera rien de nouveau alors qu'un modèle trop généralisé aura tendance à faire des recommandations faiblement personnalisées.

Ces modèles ont montré leur robustesse pour la recommandation notamment avec le modèle SVD proposé par le module-Surprise, aussi nommé Funk SVD, créé par Simon Funk lors du Netflix Prize en 2006.

Entraînement et résultat du modèle SVD

Les différentes étapes de traitement et les résultats du modèle sont disponibles dans les notebook suivants :

- **6.1 COLLABORATIVE FILTERING (Surprise SVD) :** entraînement et évaluation des modèles
- **6.2 COLLABORATIVE FILTERING (Surprise SVD Fake_users) :** test du modèle sur de faux utilisateurs

Nous avons entrainé un Model SVD (Module Surprise) et réalisé un GridSearch afin d'identifier les paramètres optimaux. Comme pour les autres modèles, nous identifions les 50 meilleures recommandations. Les résultats sont présentés dans le Tableau 5 ci-dessous.

DATASET	Mean NDCG		Mean AUC	
	Train	Test	Train	Test
Last-fm 80k	0.946	0.135	0.542	0.532

Tableau 5 : Évaluation du modèle SVD

Ce modèle donne des résultats décevants au vu de sa popularité. L'AUC indique des performances à peine meilleures que le hasard aussi bien pour le jeu d'entraînement que pour le jeu de test. À l'inverse,

les scores NDGC sont très contrastés : le modèle est très performant sur le jeu d'entraînement, mais très mauvais pour le jeu de test.

Ces résultats sont confirmés lors des recommandations faites pour nos cinq utilisateurs factices. À l'instar du modèle KNN, le modèle semble avoir simplement appris par cœur le jeu d'entraînement à l'exception de quelques recommandations fantaisistes. La difficulté de généralisation est confirmée avec les recommandations filtrées (sans les items déjà écoutés) qui sont peu pertinentes.

iii. Implicite ALS

Une des limitations principales du module-surprise est qu'il ne supporte que les données de type explicite (ex. une notation sur une échelle de 1 à 5) traitant ainsi les données manquantes comme des inconnues. Or, nous faisons face à un système de notation implicite (nombre d'écoutes), qui nous permet de savoir lorsqu'un morceau a été aimé/ écouté, mais nous ne disposons d'aucun signal contraire indiquant qu'un morceau n'a pas été aimé. Pour l'heure, une non-interaction est traitée comme un désamour.

Cela introduit un certain nombre de défis. Premièrement, traiter tous les cas manquants implique une matrice de dimension 80 000*1000 pour 21 000 entrées non nulles (97% d'interactions nulles). Deuxièmement, nous ne pouvons pas être sûrs que l'absence d'interaction d'un utilisateur avec un item correspond à un « dislike ».

La librairie Implicit va nous aider à répondre à ces défis. Le modèle implicie_ALS (Alternating least square) qu'elle contient permet en effet de créer une matrice de factorisation utilisant plusieurs niveaux de confiance sur des préférences binaires. Dans ce cas, les items non écoutés (interaction nulle) sont traités négativement avec une confiance faible alors que les items écoutés (interaction positive) sont traités comme positifs avec une bien meilleure confiance. On évite donc de donner trop d'importance aux items avec lesquels l'utilisateur n'a pas interagi.

Entraînement et résultat du modèle Implicit ALS

Les différentes étapes de traitement et les résultats du modèle sont disponibles dans les notebook suivants :

- **7.1 COLLABORATIVE FILTERING (Implicit) :** entraînement et évaluation des modèles
- **7.2 COLLABORATIVE FILTERING (Implicit SVD Fake_users) :** test du modèle sur de faux utilisateurs

Ici, nous avons choisi un modèle implicite ALS (Alternating least square) pour lequel nous avons effectué un GridSearch afin d'identifier les paramètres optimisant les métriques sur le jeu de test. Les 50 meilleures recommandations sont obtenues à l'aide d'une fonction proposée par le modèle. Les résultats de ce modèle sont présentés dans le Tableau 6 ci-dessous.

DATASET	Mean NDCG		Mean AUC	
	Train	Test	Train	Test
Last-fm 80k	0.815	0.470	0.80	0.74

Tableau 6 : Évaluation du modèle Implicit ALS

Ce modèle donne les résultats les plus encourageants. L'AUC obtenu démontre de bonnes performances sur l'ensemble des données - aussi bien sur le jeu d'entraînement que sur le jeu de test. Ces résultats sont confirmés par le NDGC qui donne les meilleurs résultats obtenus pour la recommandation de nouveaux items.

Comme pour les modèles précédents, les items recommandés aux utilisateurs factices sont principalement ceux présents dans le jeu d'entraînement. Pour autant, le NDGC sur le jeu de test grimpe à 0.636 dans le cas des recommandations filtrées (en retirant les items déjà rencontrés). Les recommandations filtrées semblent donc plutôt pertinentes.

En résumé, un modèle optimisé pour les données implicites semble être le bon choix pour notre jeu de données.

iv. Avantages et inconvénients du collaborative filtering

1. Avantages :

- Nous n'avons pas besoin des caractéristiques des items : les actions historiques des utilisateurs suffisent.
- Ces modèles peuvent proposer de la nouveauté aux utilisateurs.
- Les modèles marchent bien, même si le jeu de données est petit.

2. Inconvénients :

- Pour certains modèles (KNN), nous avons besoin d'une importante quantité de mémoire vive, il est donc difficile de les faire fonctionner en local.
- Ces modèles ne peuvent pas gérer les nouveaux items (problème de Cold Start).
- Certains modèles demandent un système de notation explicite (par exemple comprise entre 1 et 5)

c. Les modèles hybrides

Un modèle hybride est un type particulier de système de recommandation qui mélange le modèle collabarative filtering et le modèle contend based. Le but est de profiter des avantages de ces deux types d'approches afin de dépasser leurs inconvénients respectifs. Parmi ces inconvénients, on peut citer : le démarrage à froid (Cold Start), le problème de l'éparpillement (Data Sparsity) et le passage à l'échelle (Scalability).

Il existe plusieurs types des modèles hybrides. Ici, nous avons utilisé le modèle LightFM de la librairie du même nom. Il s'agit d'un modèle hybride de factorisation matricielle, qui nous permet d'insérer les métadonnées d'un item et celles d'un utilisateur dans les algorithmes traditionnels de factorisation matricielle. Chaque utilisateur et chaque item sont représentés comme la somme des représentations latentes de leurs métadonnées. De cette manière, le modèle favorise la recommandation de nouveaux items ainsi que la prise en charge de nouveaux utilisateurs.

Entraînement et résultat du modèle Hybrid LightFM

Les différentes étapes de traitement et les résultats du modèle sont disponibles dans les notebook suivants :

- **8.1 HYBRIDE (lightFM) :** entraînement et évaluation des modèles
- **8.2 HYBRIDE (lightFM-fake user) :** test du modèle sur de faux utilisateurs

DATASET	Mean NDCG		Mean AUC	
	Train	Test	Train	Test
Last-fm 45k	0.85	0.307	0.97*	0.83*

Tableau 7 : Évaluation du modèle LightFM

Le score mean NDCG dans le Tableau 8 révèle un certain sur-ajustement, puisque les résultats obtenus sur l'ensemble de test sont très inférieurs à ceux obtenus sur l'ensemble d'entraînement. A noter qu'il est difficile de commenter les résultats du mean AUC. En effet, jusqu'à présent, cette statistique était calculée par nous-même, alors qu'ici elle est produite automatiquement par le modèle.

Les résultats des recommandations aux utilisateurs factices sont, quant à eux, mitigés. D'un côté, le modèle propose des items pertinents aux utilisateurs aimant les genres musicaux populaires (pop, rock, jazz, rap), mais d'un autre côté, les recommandations faites aux utilisateurs qui aiment la musique classique ne semblent pas appropriées. Nous faisons l'hypothèse que cette inadaptation à la musique classique provient de la composition de notre jeu de données, qui ne contient – il est vrai - que peu d'items relevant de ce genre musical.

Il nous reste à tester les recommandations faites aux nouveaux utilisateurs, ainsi qu'à ajouter de nouveaux items musicaux dans la base de données.

10. Comparaison des modèles

Model	DATASET	Mean NDCG		Mean AUC		Utilisateurs factices	
		Train	Test	Train	Test	Complet	Filtré
Content based User	Last-fm 45k	0.240	0.138	0.537	0.533	Médiocre	X
Content based Item	Last-fm 45k	0.227	0.129	0.500	0.500	Médiocre	X
KNN (Surprise)	Last-fm 80k	0.583	0.492	0.545	0.498	Moyen	Faible
SVD (Surprise)	Last-fm 80k	0.946	0.135	0.542	0.532	Moyen	Médiocre
ALS (Implicit)	Last-fm 80k	0.815	0.470	0.800	0.740	Bon	Bon
Hybride (LightFM)	Last-fm 45k	0.850	0.307	0.970	0.830	Bon	Moyen

Tableau 8 : Récapitulatif des scores obtenus pour les différents modèles explorés

La première constatation est que les modèles les plus simples, basés sur le contenu (content based), ne sont pas très performants. Avec des AUC inférieurs à 0.55 et des NDGC ne dépassant pas les 0.3, on peut dire que leurs recommandations manquent de pertinence.

En seconds lieux, la spécificité de notre jeu de donnée composé d'interactions implicites a avantagé certains modèles, en particulier ceux de type le collaboratif Filtering (surprise KNN, surprise SVD et implicit ALS). Parmi ces modèles, ceux de type surprise ont eu des difficultés à appréhender le jeu de

données composé de 97% d'interactions nulles. Ainsi, les modèles surprise KNN et surprise SVD peinent à généraliser et ne sont pas capables d'assimiler les goûts spécifiques des utilisateurs.

Les modèles les plus performants sont ceux optimisés pour les données implicites, à savoir ceux du module implicit de ligthFM. Ces derniers ont montré un bon apprentissage sur l'ensemble des données avec des AUC supérieur à 0.75 et une capacité à généraliser avec de bons NDGC ainsi que de bonnes recommandations filtrées.

Au Final, le modèle le plus performent et l'ALS du module implicite, mais il ne devance le modèle lightFM que de très peu. A noter que lightFM n'a pas pu être entrainé sur le jeu de données complet – de 80 000 interactions, mais sur un jeu de 45 000 interactions. Ce modèle a donc été désavantagé.

11. Conclusion

Dans le cadre de notre projet fil rouge, plusieurs modèles de systèmes de recommandation ont été testés et évalués suivant deux métriques (l'AUC et le NDGC) ainsi que sur un panel de cinq utilisateurs factices. Il ressort une domination des modèles de type collaborative filtering. Ils se révèlent en effet plus performants pour capter les goûts spécifiques des utilisateurs. En particulier, la notation implicite présente dans notre jeu de données a donné l'avantage au modèle implicit ALS.

Dans le temps imparti pour ce projet, il n'a malheureusement pas été possible d'explorer tout le potentiel des modèles hybrides. Il serait ainsi intéressant de chercher à optimiser le modèle LightFM. Par ailleurs, il n'a pas été possible d'explorer des systèmes de recommandation basés sur les réseaux de neurones. Ces modèles seraient ainsi une suite logique à notre projet.

Une nouvelle tendance des systèmes de recommandation (de films ou de musiques) est de laisser l'utilisateur choisir le style d'items qu'il désire se voir recommander. Ainsi, par exemple, le « flow » proposé par Deezzer permet à l'utilisateur de choisir entre plusieurs ambiances (Motivation, chill, focus, party, …). À l'aide de certaines track_features comme la danceability ou le tempo, il serait possible d'adapter un modèle pour proposer le même type de service.

Enfin, nous pourrions explorer la possibilité d'avoir un curseur contrôlant le niveau d'éclectisme des recommandations, ceci afin d'éviter le phénomène de bulles hermétiques dans lesquelles les modèles testés ont tendance à enfermer les utilisateurs.