UNIVERSITÉ DE SHERBROOKE

Faculté des Sciences

Département d'Informatique

IFT799 - Science des Données

RAPPORT - TP 3 :

Systèmes de recommandation

Enseignants :

Shengrui Wang

Etienne G. Tajeuna

Auteurs :

Abdoul Madjid SANOUSSI LABO **(**CIP : SANA2909**)**

Abdou Rahime DAOUDA (CIP: DAOA2504)

Mohamed BOUBACAR BOUREIMA (CIP: BOUM3688)

Table des Matières

Présentation

Le présent document a été rédigé dans le cadre de l'exploration du thème « Sciences de Données ». Ce document présente la mise en pratique de l'essentiel sur la compréhension, des systèmes de recommandations et les autres modules du thème. Notre rapport se base sur les matériaux présentés en cours et les documents qui ont été mis à notre disposition.

Exploration et statistiques descriptives de données

L'exploration et les statistiques descriptives sont des étapes cruciales pour mieux comprendre nos données avant de prendre des décisions plus avancées concernant notre analyse. Elles nous permettent également de repérer d'éventuelles anomalies ou problèmes dans les données.

Nous disposons de deux jeux de données dont **ratings.csv,** contenant 24973468 de votes de 162541 utilisateurs sur 62423 films avec chacun d'utilisateur, ayant évaluée au moins 20 films et **movies.csv** contenant les informations sur les 62423 films et dont la principale variable nous intéressant, c'est l'ensemble des genres de chaque film. Quelques statistiques descriptive et informations sur les jeux de données, se présentent comme suit : (Voir notre fichier *Code – TP3 IFT799-Equipe 2.ipynb*)

- Pour **Movie.csv** :

Jeu de données 1 : Movies

```
In [3]: data_1
```

Out[3]:

	movieId	title	genres
0	1	Toy Story (1995)	Adventure|Animation|Children|Comedy|Fantasy
1	2	Jumanji (1995)	Adventure|Children|Fantasy
2	3	Grumpier Old Men (1995)	Comedy|Romance
3	4	Waiting to Exhale (1995)	Comedy|Drama|Romance
4	5	Father of the Bride Part II (1995)	Comedy
...	...	...	...
62418	209157	We (2018)	Drama
62419	209159	Window of the Soul (2001)	Documentary
62420	209163	Bad Poems (2018)	Comedy|Drama
62421	209169	A Girl Thing (2001)	(no genres listed)
62422	209171	Women of Devil's Island (1962)	Action|Adventure|Drama

62423 rows × 3 columns

D'autres part, on s'est intéressé à la présence des valeurs manquantes, et on n'en trouve aucune présente sur le jeu de donnée. Ce qui est plutôt intéressant pour l'analyse qui suit.

```
Jeu de données :  Movies
```

Out[6]:

	Types	Nb_Uniques	Nb manquants	Ratio manquants%
movieId	int64	62423	0	0.0
title	object	62325	0	0.0
genres	object	1639	0	0.0

```
Total ratio =  0.0
Total valeurs manquantes =  0
```

➢ Pour **ratings.csv :**

	userId	movieId	rating	timestamp
0	1	296	5.0	1147880044
1	1	306	3.5	1147868817
2	1	307	5.0	1147868828
3	1	665	5.0	1147878820
4	1	899	3.5	1147868510
...	...	...	...	...
25000090	162541	50872	4.5	1240953372
25000091	162541	55768	2.5	1240951998
25000092	162541	56176	2.0	1240950697
25000093	162541	58559	4.0	1240953434
25000094	162541	63876	5.0	1240952515

25000095 rows × 4 columns

	userId	movieId	rating	timestamp
count	2.500010e+07	2.500010e+07	2.500010e+07	2.500010e+07
mean	8.118928e+04	2.138798e+04	3.533854e+00	1.215601e+09
std	4.679172e+04	3.919886e+04	1.060744e+00	2.268758e+08
min	1.000000e+00	1.000000e+00	5.000000e-01	7.896520e+08
25%	4.051000e+04	1.196000e+03	3.000000e+00	1.011747e+09
50%	8.091400e+04	2.947000e+03	3.500000e+00	1.198868e+09
75%	1.215570e+05	8.623000e+03	4.000000e+00	1.447205e+09
max	1.625410e+05	2.091710e+05	5.000000e+00	1.574328e+09

Ratings a 4 variables, dont deux identificatrices (Utilisateur et Film), une catégorielle qui représente les votes donnés par chaque utilisateur sur chaque film et une date correspondante à celle ou le vote a été fait, en format timestamp. Quelques mesures descriptives sur les variables présentes, l'image en haut a droite, mais ne présentant aucun sens pour nous, car aucune n'est quantitative numérique.

On s'est également intéressé à la présence des valeurs manquantes, et on n'en trouve aucune présente sur le jeu de donnée Ratings.

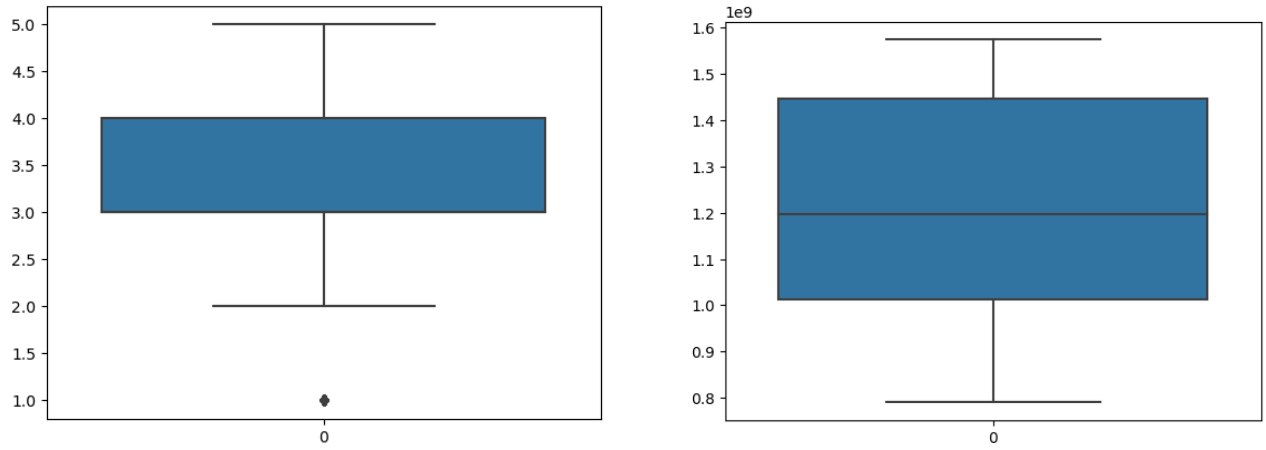
```
Jeu de données :  Ratings
```

Out[16]:

	Types	Nb_Uniques	Nb manquants	Ratio manquants%
userId	int64	162541	0	0.0
movieId	int64	59047	0	0.0
rating	float64	10	0	0.0
timestamp	int64	20115267	0	0.0

```
Total ratio =  0.0
Total valeurs manquantes =  0
```

Pour les variables timestamp et rating, nous cherchons à voir la distribution de celles-ci ainsi que la présence potentielle de valeur aberrantes.



Distribution symétrique pour la variable, le timestamp (les dates des votes) et non symétrique pour les ratings, ce qui conclut la présence des données non balancées. Pour les valeurs aberrantes, nous émettons l'hypothèse, qu'aucune valeur aberrante n'est présente.

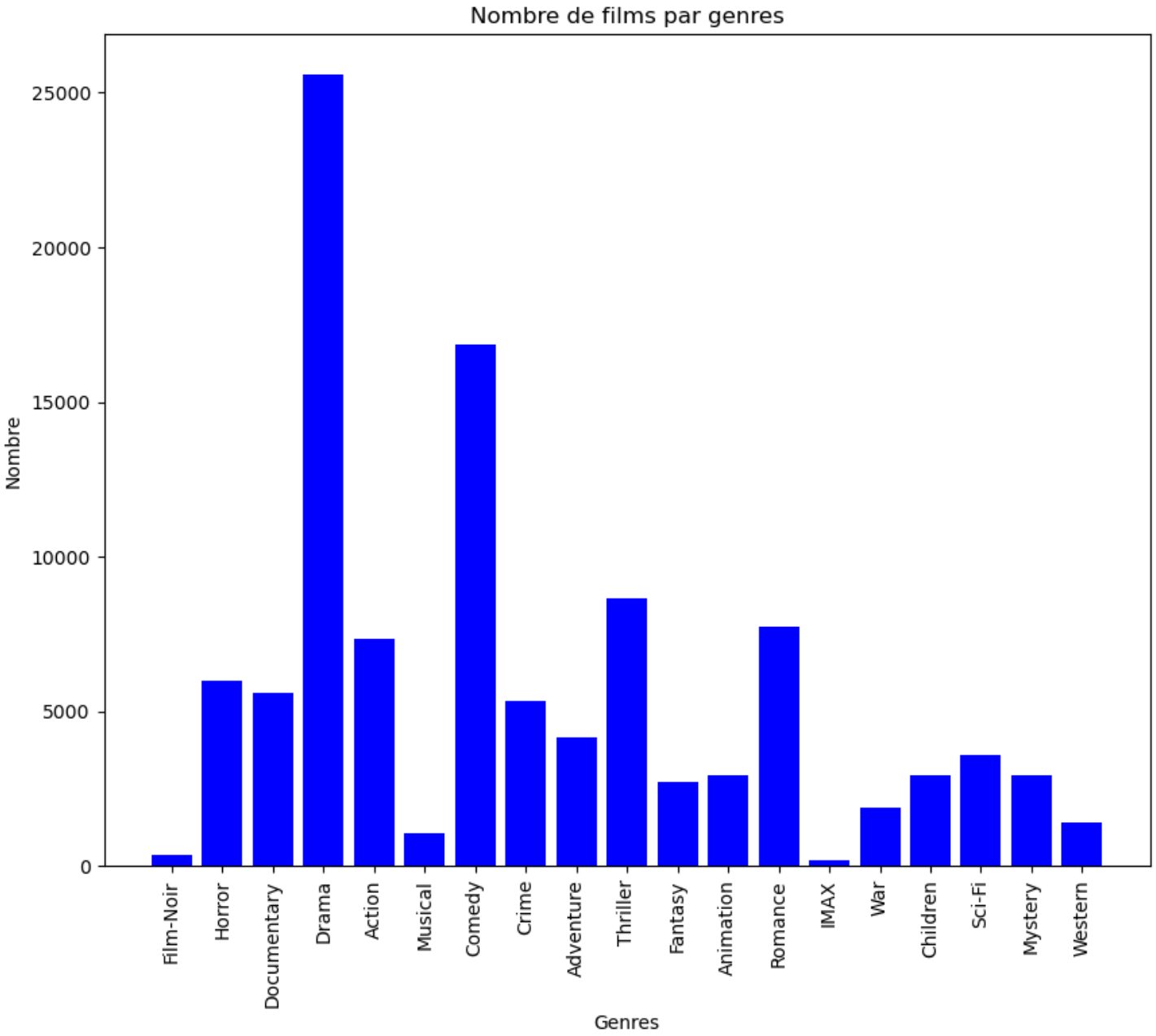
Il serait également important de connaitre la distribution des classes présente, tout autant que leur nombre. Le camembert ci-dessous donne un aperçu sur cette connaissance des étiquettes.

Nous verrons dans les prochaines étapes, les prétraitements requis pour la suite de notre modélisation du système de recommandation et aussi les statistiques descriptives qu'on aurait manqué dans cette section.

Question 1 : Construction du diagramme en bâton illustrant le nombre de films que l'on a par genre

Pour répondre à cette question, il suffisait de créer un dictionnaire dont la clé c'est le genre et la valeur, c'est le nombre de fois qu'on rencontre le genre, lors du parcours de l'ensemble du jeu de donnée film.

Nous soulignons ici que les films au genre non listés: ' (no genres listed) ' seront ignoré lors du comptage. Nous obtenons ainsi le diagramme demandé comme suit :



Sur cette image, nous constatons la présence de 19 genres distincts globalement et que les genres majoritaires sont le **Drama** et la **Comédie** et ceux minoritaire sont **IMAX** et **Film-Noir**.

Autre remarque, il serait possible de séparer le goût des utilisateurs en deux grandes catégories. Ceux positives (Ramance, Comedy,…) et ceux négatives (Horror, Crime, War).

Question 2 : Extraction des nouveaux jeux de données movies1.csv et ratings1.csv

Il est demandé ici de créer deux jeux de données, issus d'un prétraitement sur les jeux de données initiaux.

- Movies1.csv : En supprimant, de movies.csv, toutes lignes dont l'identifiant du film est non listé. En sortie, on a un nouveau jeu avec 57361 films, décrit par le tableau suivant :

```
movies1
```

	movieId	title	genres
0	1	Toy Story (1995)	Adventure|Animation|Children|Comedy|Fantasy
1	2	Jumanji (1995)	Adventure|Children|Fantasy
2	3	Grumpier Old Men (1995)	Comedy|Romance
3	4	Waiting to Exhale (1995)	Comedy|Drama|Romance
4	5	Father of the Bride Part II (1995)	Comedy
...	...	...	...
57356	209155	Santosh Subramaniam (2008)	Action|Comedy|Romance
57357	209157	We (2018)	Drama
57358	209159	Window of the Soul (2001)	Documentary
57359	209163	Bad Poems (2018)	Comedy|Drama
57360	209171	Women of Devil's Island (1962)	Action|Adventure|Drama

57361 rows × 3 columns

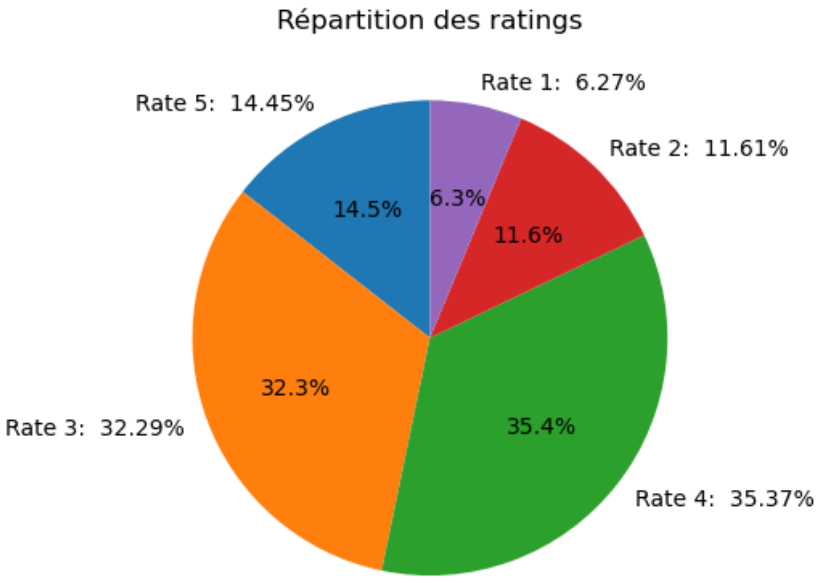
- Ratings1.csv : En changeant les ratings de valeurs 5.5, 4.5, 3.5, 2.5, 1.5, 0.5 par les valeurs respectives de 5, 4, 3, 2, 1, 1 et également, tous les votes des utilisateurs dont le film est sans genre, seront supprimés. On a en sortie le nouveau jeu avec 24973468 votes au total.

```
ratings1
```

	userId	movieId	rating	timestamp
0	1	296	5.0	1147880044
1	1	306	3.0	1147868817
2	1	307	5.0	1147868828
3	1	665	5.0	1147878820
4	1	899	3.0	1147868510
...	...	...	...	...
24973463	162541	50872	4.0	1240953372
24973464	162541	55768	2.0	1240951998
24973465	162541	56176	2.0	1240950697
24973466	162541	58559	4.0	1240953434
24973467	162541	63876	5.0	1240952515

24973468 rows × 4 columns

Pour mieux voir la nouvelle distribution de la variable rating, nous décidons de tracer la proportion des votes, via ce diagramme de camembert :



Les votes sont asymétriquement distribués toujours. Un jeu de données note balancé avec le rating 4 est majoritairement présent dans notre jeu de

donnée (35,4%), et inversement Rate 1, qui est minoritaire (6.3%). Les données ne sont pas balancées, une classe pourrait être discriminée, lors de sa prédiction, basé sur un modèle conçu sur cette base de données.

Le code ayant servi à ce prétraitement, donnera plus de détails sur le procède. Les fichiers résultants sont enregistrés et seront envoyés, avec ce rapport, tel que demandé par la consigne.

Question 3 : Construction de la Matrice binaire de contenu

A base des 19 genres associés à chaque film, nous coconstruisons la matrice binaire de contenu C caractérisant chaque film. Une matrice a 57361 lignes de film et 19 colonnes, une pour chaque caractéristique, dont les valeurs sont 1 si le film présente ce genre et 0 sinon. Une colonne additionnelle pour représenter l'identifiant du film en question (movieId). Ci-dessous un aperçu de la matrice résultante :

	movieId	Musical	Romance	Crime	Thriller	Children	Sci-Fi	Drama	Mystery	Western	IMAX	War	Film-Noir	Animation	Documentary	Fantasy	Adventure	Action	Comedy	Horror
0	1	0	0	0	0	1	0	0	0	0	0	0	0	1	0	1	1	0	1	0
1	2	0	0	0	0	1	0	0	0	0	0	0	0	0	0	1	1	0	0	0
2	3	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0
3	4	0	1	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	1	0
4	5	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
57356	209155	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	0
57357	209157	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0
57358	209159	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0
57359	209163	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	1	0
57360	209171	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	1	1	0	0

57361 rows × 20 columns

Pour voir le détail de la façon d'extraire ce contenu, veiller consulter le fichier code de notre TP.

Question 4 : Construction de la matrice de profil des utilisateurs (P)

Notre système de recommandation est un système mixte, basé le contenu et le collaboratif. Pour pouvoir exploité les deux procédures, on devrait avoir en entrer du modèle de recommandation, des données qui prennent en compte le contenu d'un côté, ici la matrice binaire des genres, et les votes des utilisateurs de l'autre côté. La matrice de profil, fait effectivement le compromis entre ces deux utilisateurs, en procurant un certain score d'intérêt à un utilisateur donné, pour les groupes de caractéristique donnés.

Cette matrice est donnée par la formule suivante :

$$Profil(U_u \mid \boldsymbol{I}_u) = \sum_{I_i \in \boldsymbol{I}_u} r_{u,i} C_i$$

- U_u est l'utilisateur considéré et I_u l'ensemble des films qu'il a regardé.
- $r_{u,i}$, Le rating donné par U_u sur I_u
- $C_{i,}$ le vecteur binaire caractérisant le film $I_{i.}$ (La ligne i de votre matrice C).

Est en détails, l'implémentation de ladite matrice, dans le fichier code du TP.

Un aperçu de la matrice de profil :

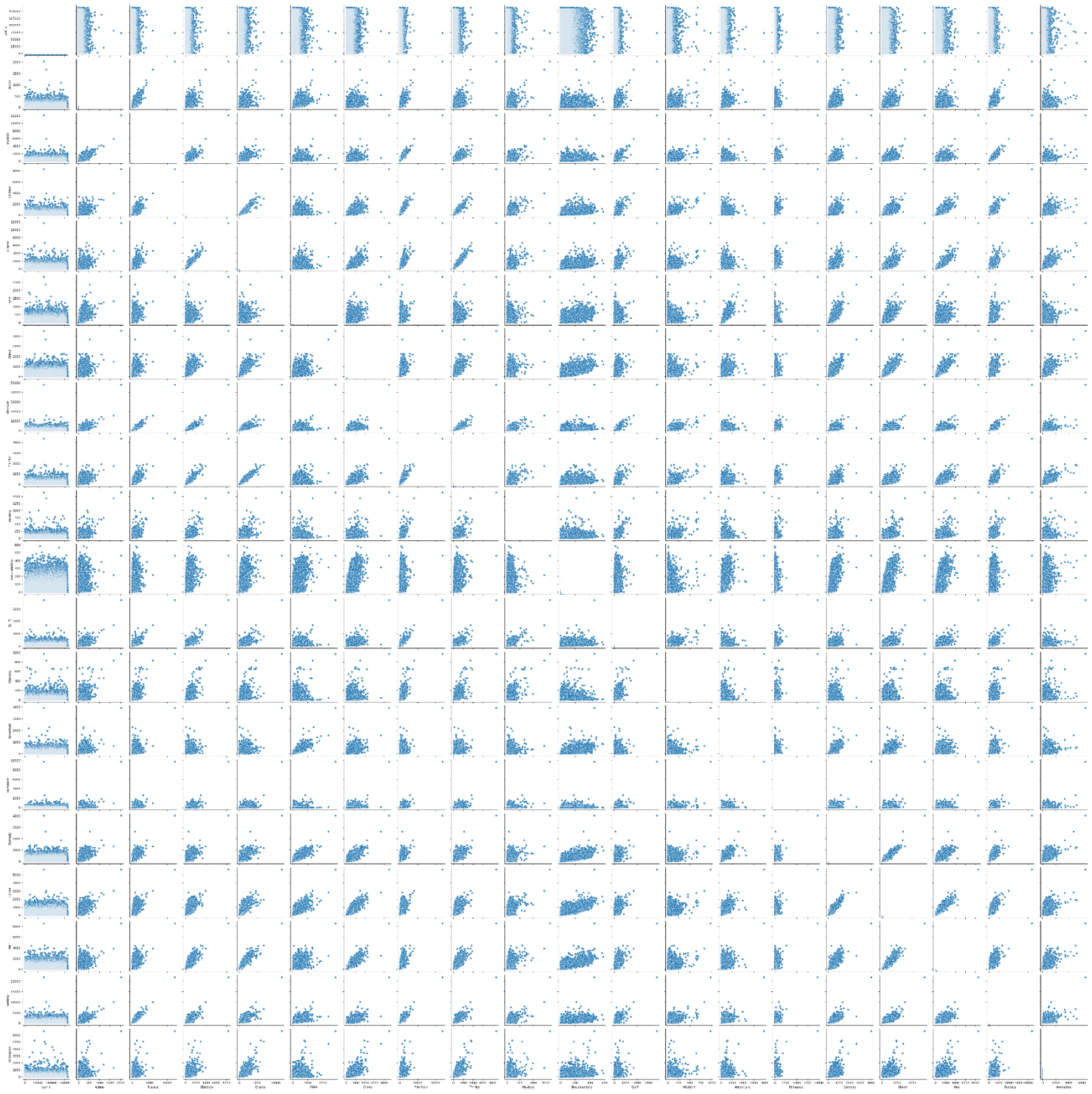
	userId	Children	Romance	Action	Animation	Thriller	Crime	Mystery	Sci-Fi	Western	Film-Noir	Horror	IMAX	Drama	Comedy	Adventure	Documentary	Musical	Fantasy	War
0	1	18.0	71.0	31.0	21.0	11.0	17.0	196.0	13.0	2.0	0.0	19.0	3.0	8.0	2.0	16.0	39.0	16.0	84.0	3.0
1	2	34.0	105.0	55.0	112.0	87.0	109.0	316.0	29.0	12.0	28.0	50.0	0.0	58.0	0.0	111.0	285.0	237.0	202.0	12.0
2	3	21.0	204.0	496.0	839.0	171.0	786.0	876.0	224.0	28.0	290.0	93.0	21.0	193.0	9.0	276.0	696.0	1160.0	583.0	152.0
3	4	24.0	32.0	136.0	183.0	82.0	251.0	174.0	64.0	24.0	90.0	28.0	0.0	100.0	20.0	103.0	325.0	424.0	271.0	30.0
4	5	26.0	71.0	58.0	96.0	30.0	45.0	172.0	33.0	15.0	12.0	9.0	0.0	15.0	0.0	28.0	81.0	67.0	175.0	14.0
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
162536	162537	50.0	136.0	38.0	63.0	46.0	18.0	139.0	13.0	5.0	0.0	10.0	0.0	26.0	3.0	36.0	67.0	75.0	239.0	22.0
162537	162538	16.0	221.0	52.0	71.0	30.0	42.0	286.0	26.0	8.0	24.0	37.0	0.0	26.0	0.0	42.0	79.0	73.0	222.0	38.0
162538	162539	5.0	56.0	20.0	40.0	10.0	43.0	125.0	15.0	0.0	0.0	97.0	13.0	13.0	0.0	13.0	61.0	78.0	62.0	5.0
162539	162540	13.0	96.0	32.0	93.0	76.0	41.0	135.0	31.0	0.0	33.0	22.0	0.0	80.0	0.0	56.0	111.0	82.0	111.0	37.0
162540	162541	16.0	121.0	81.0	102.0	54.0	117.0	257.0	34.0	15.0	19.0	15.0	4.0	50.0	0.0	101.0	183.0	162.0	232.0	24.0

162541 rows × 20 columns

Nous avons également accompagné la matrice par userId, pour pouvoir identifié l'utilisateur cherché, pour des raisons futures.

Une interprétation plausible, pour le **profil de l'utilisateur 1**, est qu'il s'intéresse plus aux films de genre **Mystery** (avec un score de 196) et absolument pas aux films de genre **Film-Noir** (0 comme score). Mieux, on pourrait avouer qu'il n'en a jamais regardé, car aucun vote sur celui-ci).

Un **pairplot** des caractéristiques, révèle l'existence de certains genres, corrélé entres eux. C'est le cas par exemple de **Horror** et **War** ou encore **Musical et Fantasy**.



Question 5 : Clustering spectral pour limiter la recherche d'items et d'utilisateurs

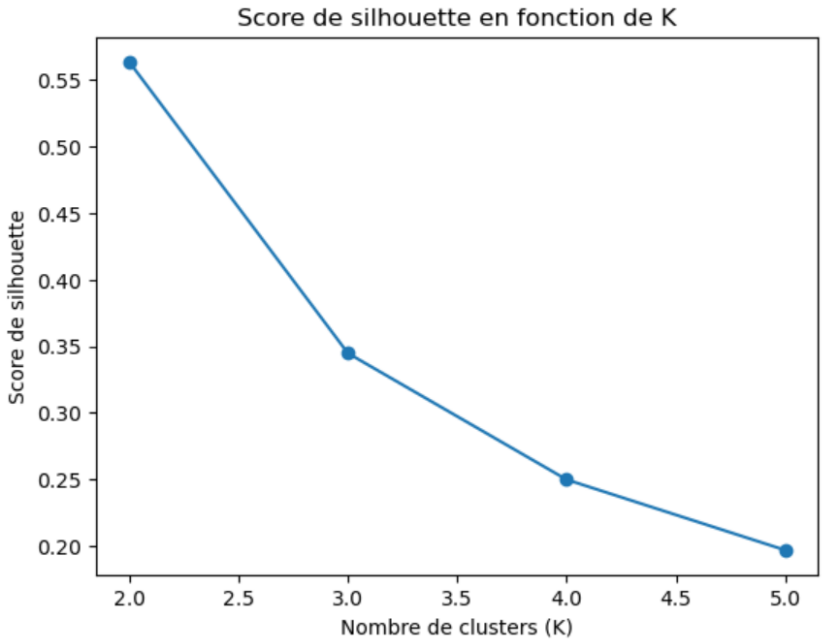
Ici on voudrait trouver les groupes d'utilisateurs et d'items présentant les mêmes caractéristiques. En utilisant la matrice de profil P. Testons plusieurs valeurs de K = {2, 3, 4, 5} pour performer un clustering spectral, à k clusters et retenons celle dont le score de silhouette est le plus élevé.

Pour ce faire, nous avons effectué deux méthodes différentes, la seconde pour remédier à la problématique des ressources qu'on a rencontré lors de la première.

- **En utilisant la fonction SpectralClustering de sk-learn :**

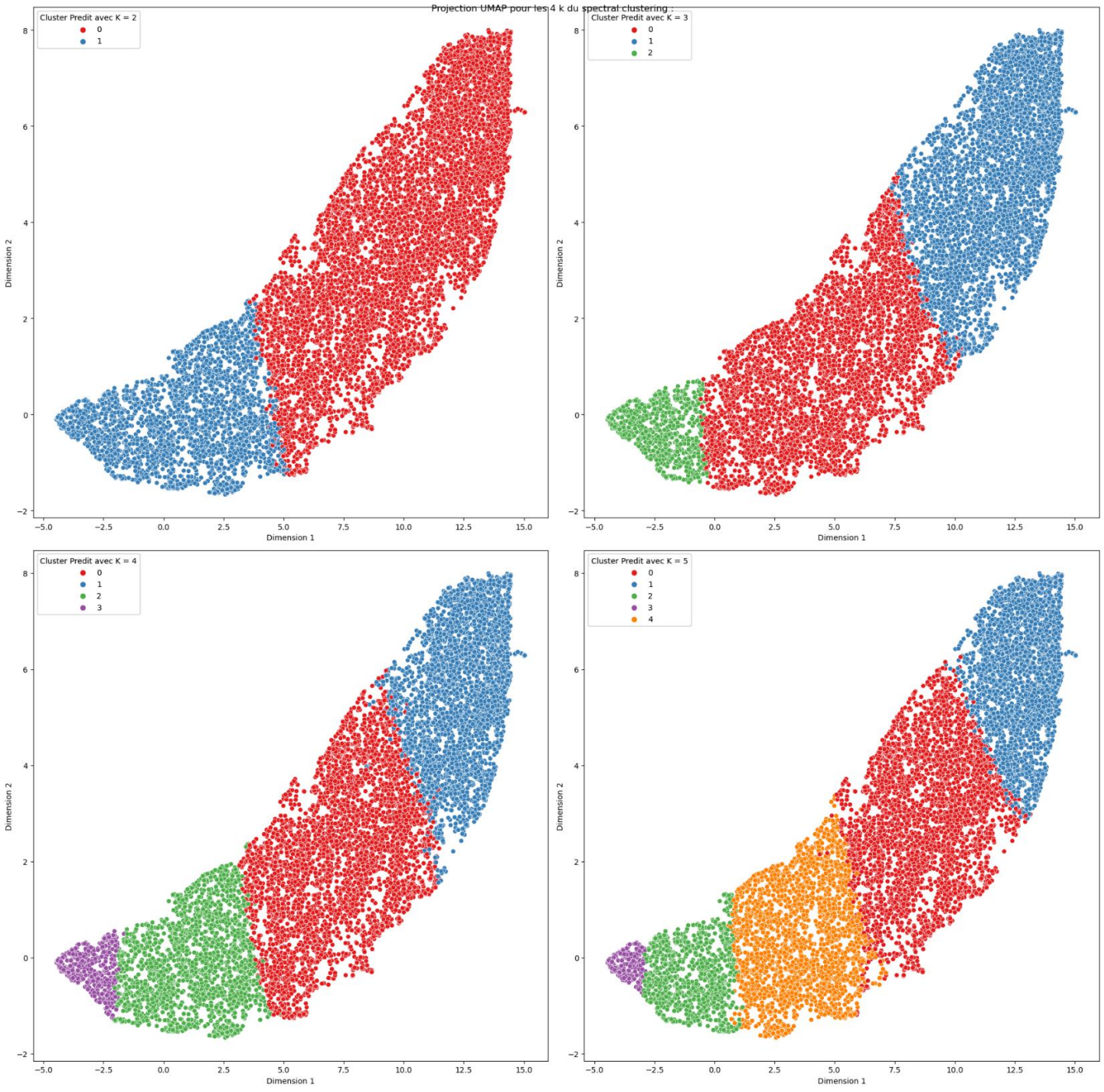
Nous testons le modèle sur les 4 valeurs de k (2,3,4,5). Pour chacune, nous évaluons le score silhouette, pour comparaison. Toutefois, nos ressource étant limitées, il nous a été impossible d'avoir un résultat en utilisant tout d'un coup la matrice de profil. L'algorithme mettant du temps à exécuter. Nous décidons alors d'utiliser une matrice réduite des 10000 premiers utilisateurs pour espérer obtenir les clusters optimaux.

Ci-dessous les scatterplots obtenues sur une projection UMAP, pour les 4 clusters coloriés, ainsi que la courbe associée au score silhouette pour les 4 valeurs de k.




```
La meilleure valeur de K est : 2
```

Pour k = 2 clusters, on obtient le meilleur score, quoi que la division ne soit pas équitable. Toutefois, on pourrait être en possession des données non balancées. Et rappelons-nous l'hypothèse selon laquelle on a deux catégories de caractéristique (positive et négative). Drama et Comedy sont les plus majoritaires et peuvent être vu comme étant des genres positives. S'ajoute le fait que si les utilisateurs préfèrent ces genres majoritaires.

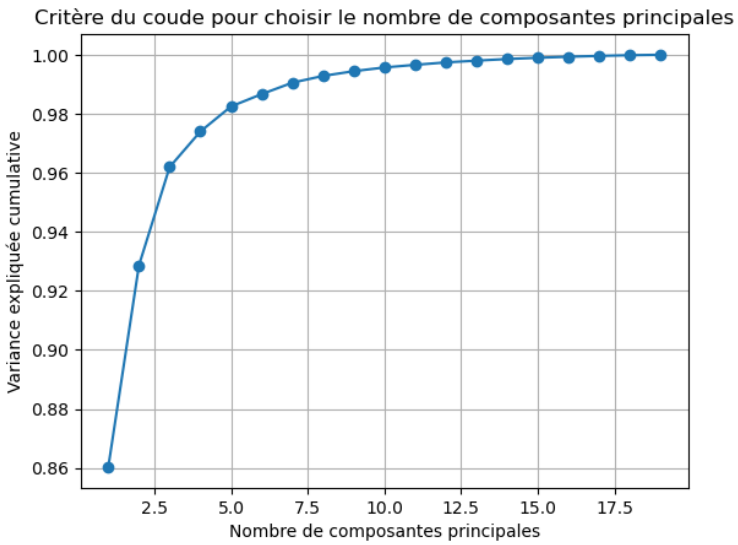


Ainsi notre clustering semble logique, toutefois serait-il possible de généraliser sur les 150000 autres utilisateurs ? Sans aucune idée sur leur distribution? D'où l'utilité de la prochaine méthode que nous implémentons.

- **Procédure 'Traditionnelle' ACP + K-means :**

Ici, nous performons une **ACP** sur l'ensembles des données initiales, pour choisir un nombre de composante principale limité, qui représenterons le maximum possible de la variance de nos données. Puis faire un K-means sur les 4 composantes principales.

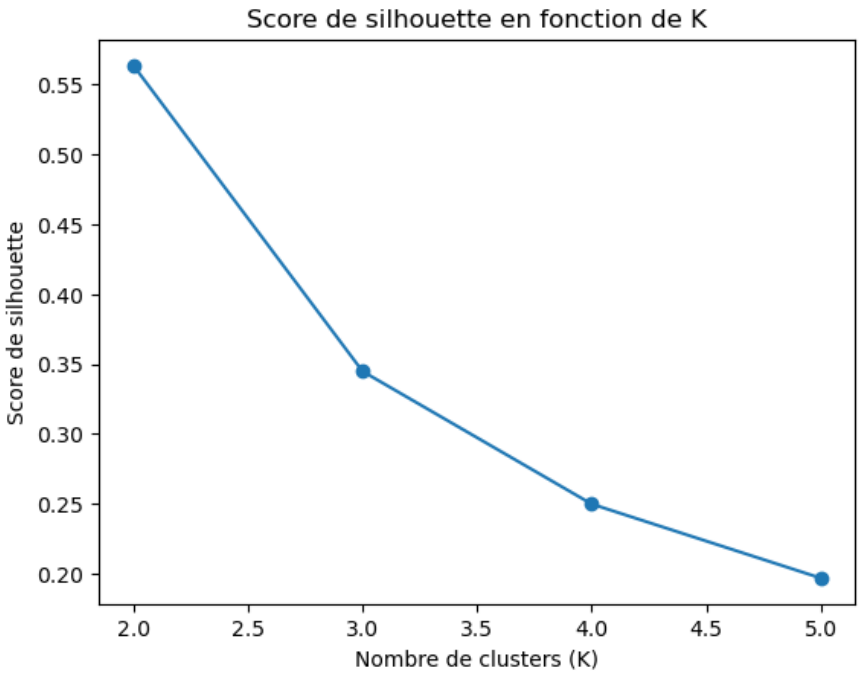
Le choix du nombre de composante est fait selon le critère de coude sur la variance expliquée.

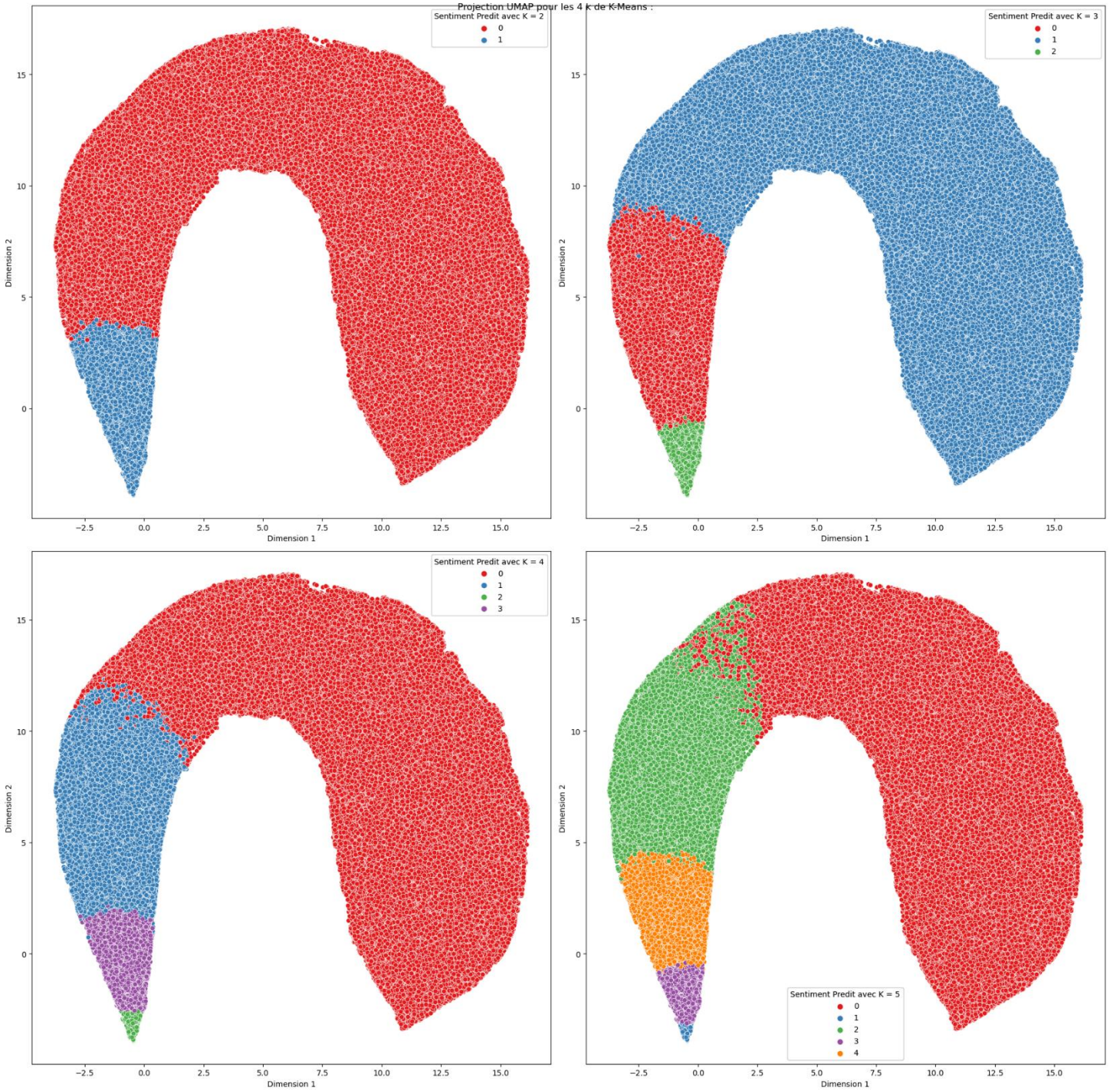


	PC1	PC2	PC3	PC4	userId
0	-217.730858	134.014759	-2.552757	25.368056	1
1	119.060338	-65.775944	-54.215921	137.250310	2
2	1561.470399	-813.943580	301.234893	169.229252	3
3	187.343159	-346.031404	-45.521748	69.900633	4
4	-124.958841	50.535057	-37.640359	-31.566212	5
...	...	...	...	...	...
162536	-120.614658	64.030587	-132.762376	-59.588411	162537
162537	-4.589969	147.893823	-81.341751	-2.257366	162538
162538	-236.611007	37.800545	13.435102	26.805302	162539
162539	-150.680671	-3.107588	-46.797100	33.641261	162540
162540	50.172159	-16.145734	-73.623011	42.141303	162541

162541 rows × 5 columns

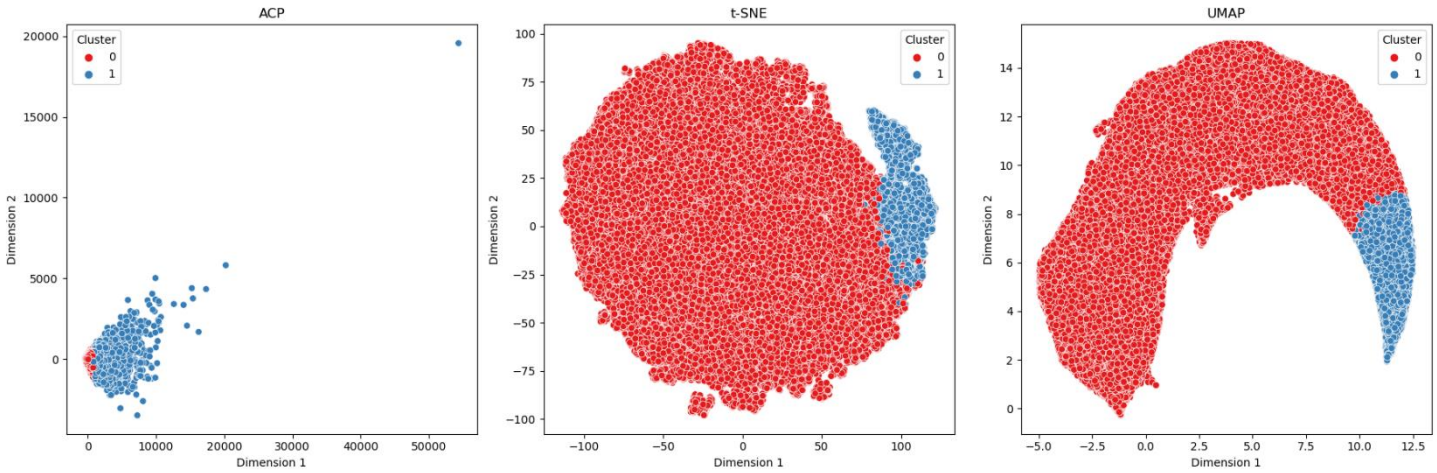
Les 4 premières composantes, représentent à elles seules 97,4% d'information. Nous n'en garderons qu'elle seule pour effectuer notre clustering K-Means. Les 4 k résultants accompagnés par le score silhouette sont comme suit :





L'une ou l'autre des méthodes, k = 2 clusters, présente le plus grand score silhouette quoi que les données ne soient pas bien balancées. Nous allons nous en servir pour effectuer la classification des utilisateurs. La seconde méthode serait utilisée pour cette classification vu qu'elle prend en compte les utilisateurs dans leur entièreté.

Ensuite, nous effectuerons une assignation de la matrice de profil initiale, sur tous les utilisateurs, selon les clusters résultants du K-means sur la matrice de profil.



Nous obtenons, le scaterplot résultant des clusters selon les 3 projections, ACP, TSNE et UMAP. La forme de la projection t-SNE, nous révèle une forme, suffisamment séparée (graine qui pousse), pour valider le clustering avec k = 2.

```
0    151629
1     10912
```

Question 6 : Conception du model de système de recommandation

On voudrait à présent modéliser notre système de recommandation. Pour cela nous préparons les données qui permettront de valider notre futur système de recommandation. Sachant que les votes varient de 1 à 5, on voudrait s'assurer que tous les cas de figures soient pris en compte lors de l'apprentissage. Pour ce faire, suivant le fichier ratings1.csv, on va subdiviser en trois fichiers distincts, ratings train.csv, ratings evaluation.csv et ratings test.csv. Le fichier ratings train.csv contient 60% des données de ratings1.csv tandis que ratings evaluation.csv et ratings test.csv doivent contenir 20% des données de ratings1.csv chacun. La subdivision se fait de manière aléatoire. Un petit aperçu des données issues de cette étape, qui est une forme d'évaluation hors ligne du modèle.

```
Taille de l'ensemble d'entraînement : 14984080
Taille de l'ensemble de test : 4994694
Taille de l'ensemble d'évaluation : 4994694
Compte des votes unique de l'ensemble d'entraînement :
 4.0    5299735
3.0    4838879
5.0    2165651
2.0    1739868
1.0     939947
Name: rating, dtype: int64
```

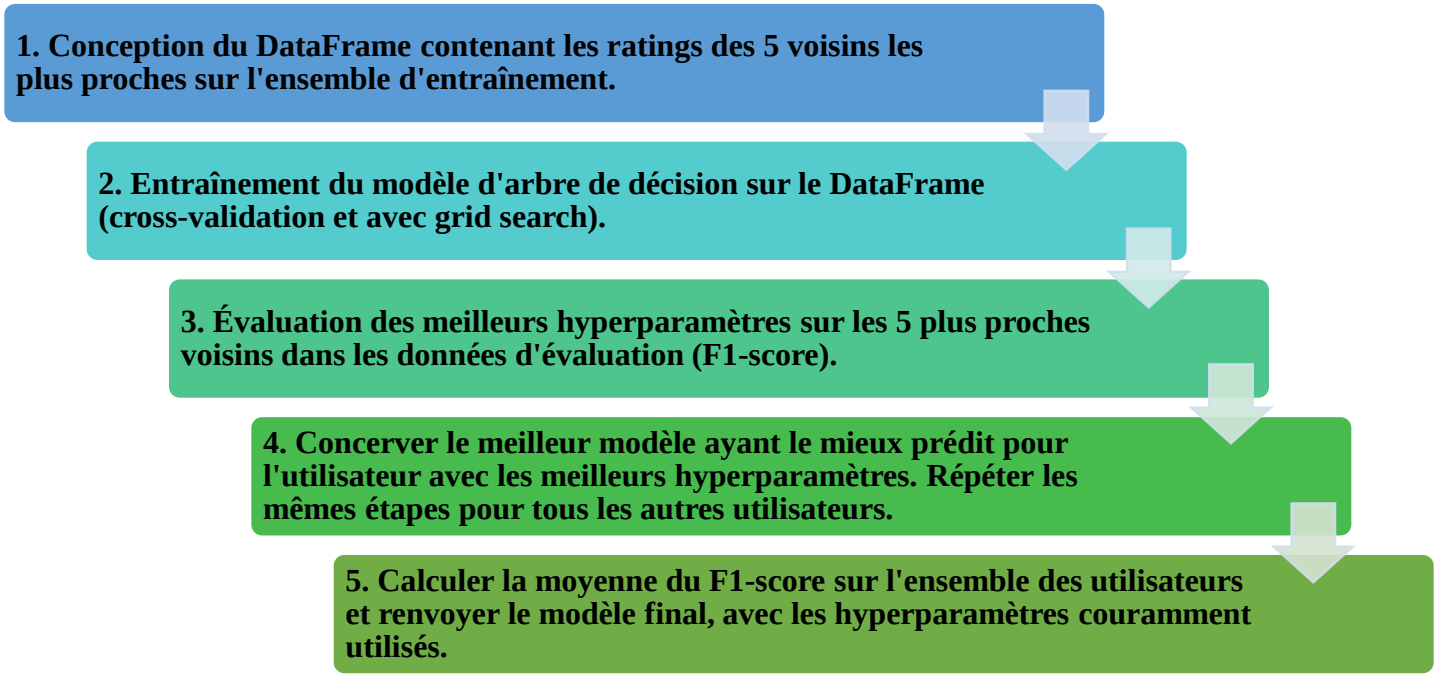
```
Compte des votes unique de l'ensemble de test :
 4.0    1766579
3.0    1612959
5.0     721884
2.0     579957
1.0     313315
Name: rating, dtype: int64
Compte des votes unique de l'ensemble d'évaluation :
 4.0    1766579
3.0    1612960
5.0     721883
2.0     579956
1.0     313316
Name: rating, dtype: int64
```

a. Entrainement et évaluation du modèle de prédiction des votes des utilisateurs :

Entrainement d'un modèle d'arbre de décision, pour la prédiction du rating d'un utilisateur u selon le vote de ses 5 plus proches voisins en exploitant les clusters pour retrouver ces plus proches voisins.

NB : On testera plusieurs hyperparamètres pour notre arbre de décision et retenir celui qui nous donnerait les meilleurs résultats suivant le critère F1 score sur les données d'évaluation.

Les différentes étapes pour la conception du prédicteur, sont comme suit :



✓ **1. Conception du DataFrame contenant les ratings des 5 voisins :**

Premièrement, nous établissons une fonction qui identifie les 5 utilisateurs les plus proches d'un utilisateur donné (identifié par user_id). Ces 5 utilisateurs sont cherchés dans la matrice de profil, en calculant la distance de l'utilisateur, avec le reste des utilisateurs. Cette fonction construit ensuite un DataFrame contenant les données de ces 5 utilisateurs les plus proches à partir de la matrice des évaluations. Cette matrice peut être extraite des données d'entraînement si l'objectif est d'entraîner le modèle, ou des données d'évaluation si l'on souhaite évaluer le modèle.

Lors du calcul des distances, nous prenons en compte le fait qu'un utilisateur n'aurait pas regardé un film. Cette information est déjà intégrée dans la matrice de profil, provenant de la matrice des contenus. Ainsi, il n'est pas nécessaire de filtrer les utilisateurs en fonction de leurs interactions avec des films spécifiques. Pour plus de détails sur la mise en œuvre de cette fonction, veuillez consulter le fichier de code associé. Un exemple de la fonction, appliqué sur l'utilisateur 1 et 3, appartenant au cluster 0 et 1 respectivement.

```
X_k_pp_1,y_k_pp_1 = data_k_pp(1,X_train,y_train)
```

```
Les 5 utilisateurs les plus proches de l'utilisateur 1 appartenant au cluster [0] sont : [92046, 80974, 20055, 33844, 1
37293]
```

```
X_k_pp_3,y_k_pp_3 = data_k_pp(3,X_train,y_train)
```

```
Les 5 utilisateurs les plus proches de l'utilisateur 3 appartenant au cluster [1] sont : [72315, 80974, 137293, 75309,
110971]
```

	userId	movieId	timestamp
0	33844	7624	1436135320
1	92046	1683	1000146131
2	33844	445	1436135423
3	80974	4085	997037505
4	92046	3081	1000834795
...	...	...	...
23915	33844	109372	1460188873
23916	80974	56607	1210272627
23917	20055	7936	1181379116
23918	80974	7390	1123530092
23919	137293	85389	1301061259

23920 rows × 3 columns

	rating
0	3.0
1	5.0
2	1.0
3	4.0
4	4.0
...	...
23915	1.0
23916	4.0
23917	4.0
23918	3.0
23919	3.0

23920 rows × 1 columns

Pour l'utilisateur 1

✓ **2. Entraînement du modèle d'arbre de décision :**

Le modèle est ensuite entraîné en utilisant la méthode de validation croisée (cross-validation avec 5 groupes) pour évaluer ses performances sur différentes subdivisions des données. Une recherche de grille (grid search) est effectuée pour trouver les meilleurs hyperparamètres du modèle, optimisant ainsi sa capacité à prédire les votes. La grille est composée de :

```
param_grid
```

```
{'criterion': ['gini', 'entropy'],
 'splitter': ['best', 'random'],
 'max_depth': [None, 10, 20, 30],
 'min_samples_split': [2, 5, 10],
 'min_samples_leaf': [1, 2, 4]}
```

On obtient pour U1 :

Les meilleurs hyperparamètres suivant :

```
({'criterion': 'gini',
  'max_depth': None,
  'min_samples_leaf': 1,
  'min_samples_split': 2,
  'splitter': 'best'},
```

✓ **3. Évaluation des meilleurs hyperparamètres sur les données d'évaluation :**

Les meilleurs hyperparamètres identifiés précédemment sont utilisés pour prédire les évaluations des 5 utilisateurs les plus proches dans le jeu de données d'évaluation. Le score F1 est calculé pour évaluer la précision des prédictions par rapport aux évaluations réelles. Dans cette étape, on ne s'intéresse qu'à la prédiction, des votes des films déjà regardés, pour l'utilisateur en entré. Le vote des nouveaux films se fera ultérieurement. Obtient comme sortie :

- Un F1-score **de 0.281.**

- Et une prédiction des films regards comme suit :

```
{8786.0: 3.0,
 1250.0: 3.0,
 2843.0: 3.0,
 5269.0: 3.0,
 899.0: 4.0,
 7327.0: 3.0,
 7940.0: 4.0,
 1260.0: 3.0,
 5878.0: 3.0,
 7365.0: 2.0,
 1237.0: 3.0,
 5684.0: 3.0,
 8685.0: 3.0,
 7938.0: 4.0,
 5767.0: 3.0,
 8973.0: 3.0,
 2011.0: 3.0,
 1217.0: 3.0,
 7323.0: 3.0,
 306.0: 2.0,
 2012.0: 3.0,
 2161.0: 3.0,
 8154.0: 4.0,
 2632.0: 2.0,
 8873.0: 3.0,
 6711.0: 3.0,
 6539.0: 3.0,
 8405.0: 3.0,
 665.0: 4.0,
 3569.0: 1.0,
 4973.0: 3.0,
 32591.0: 3.0,
 5952.0: 2.0,
 296.0: 2.0,
 2573.0: 2.0,
 27721.0: 3.0,
 2068.0: 3.0,
 4703.0: 3.0,
 7937.0: 4.0,
 8327.0: 3.0,
 3448.0: 1.0,
 1653.0: 3.0,
 8729.0: 3.0,
 7361.0: 2.0,
 1175.0: 3.0,
 131072.0: 3.0,
```

✓ **4. Retour du meilleur modèle et Répétition des étapes pour tous les autres utilisateurs :**

Le modèle d'arbre de décision optimal, ayant démontré les meilleures performances pour l'utilisateur 1, sur les données d'entraînement et d'évaluation, est retourné. Les étapes 1,2,3 et 4 sont répétées pour chaque

utilisateur, en identifiant les 5 utilisateurs les plus proches spécifiques à chaque utilisateur et optimisant les hyperparamètres

```
Meilleur hyperparametres pour U1 :  {'criterion': 'gini', 'max_depth': None, 'min_samples_leaf': 1, 'min_samples_spli
t': 2, 'splitter': 'best'}
F1-score pour U1 :  0.30346346780596434
La prediction des films regardés  :
```

	7325.0	8781.0	1882.0	115713.0	6373.0	66097.0	8969.0	112623.0	106002.0	55765.0	...	131052.0	131054.0	131056.0	131058.0	131060.0	131062.0	1
0	3.0	3.0	4.0	3.0	3.0	3.0	3.0	4.0	3.0	3.0	...	3.0	3.0	3.0	3.0	3.0	3.0	

1 rows × 57088 columns

.

.

.

.

✓ **5. Calcul de la moyenne du F1-score sur l'ensemble des utilisateurs :**

Les scores F1 obtenus pour chaque utilisateur sont moyennés pour obtenir une mesure agrégée de la performance du modèle sur l'ensemble des utilisateurs.

Le modèle final, avec les hyperparamètres les plus couramment utilisés, est retourné comme le modèle optimal pour cette tâche particulière. Et celui-ci pourrait être utilisé pour prédire les votes d'un nouvel utilisateur. Pour ce faire, il conviendrait de passer l'utilisateur en question, avec l'ensemble de ses votes donnés aux films qu'il a regardé. Et la procédure de la prédiction, est identique que pour la conception du modèle.

Problématique :

Cependant, la même problématique de ressource que précédemment, survient à cette étape. Car reproduire pour chaque utilisateur les 5 étapes précédentes, ne serait-ce que le calcul de distance, prend du temps à plus forte raison l'entrainement de l'arbre de décision, sur le data des 5 utilisateurs.

Remarque : De plus, la fonction de classification utilisée, semble inappropriée dans notre cas, surtout vu l'inexistence des features intéressants, dans la matrice en entrée de l'arbre de décision. Juste l'identifiant du film et le timestamp qui n'est que la date ou le vote est fait. L'arbre de décision aurait du mal à classifier les votes, et même s'il arrive, ces résultats n'auront pas une très grande signification.

La consigne du TP, suggérer que les données en entrée seront les ratings des 5 plus proche voisins, mais l'on ne peut pas quand même utiliser la variable rating uniquement pour faire la prédiction du rating. Non plus l'inclure

dans les feature du modèle d'entrainement, car celle-ci discriminera le modèle, et fera ressortir un F1-score de 1. Notre objectif étant de prédire les votes de l'utilisateur pour un film nouveau, on ne pourrait disposer de son vote, sur le film non regardé.

Solution proposée :

Pour réduire la complexité de l'algorithme, on s'en servira du clustering pour prendre uniquement un nombre restreint N dans chaque cluster, puis calculer la moyenne de leurs prédictions (la meilleure technique qu'on aurait due utilisée pour la fonction de classifieur, sur les 5 plus proches voisins), ensuite généraliser sur le cluster. Car plus les utilisateurs auront le même profil (même gout), plus ils auront tendance à faire le même vote sur le même film. Ceci n'est que dans l'objectif d'obtenir le meilleur modèle déjà entraîné et évalué, pour s'en servir de celui-ci plus tard. Notre solution, implique également la prédiction des film non regardés en plus de ceux regardés.

La concaténation pourrait se faire en par colonne également, mais l'algo sera plus complexe. Nous essayons d'éviter celle-ci, mais potentiellement, elle donnera un meilleur résultat sur l'arbre de décision.

	userId	movieId	timestamp	Proche1	Proche2	Proche3	Proche4	Proche5
62	1	8786	1147877853	None	None	None	None	None
9	1	1250	1147868414	None	None	None	None	None
20	1	2843	1147868891	None	None	None	None	None
31	1	5269	1147879571	None	None	None	None	None
4	1	899	1147868510	None	None	None	None	None
47	1	7327	1147868855	None	None	None	None	None
54	1	7940	1147877967	None	None	None	None	None
10	1	1260	1147877857	None	None	None	None	None
34	1	5878	1147868807	None	None	None	None	None
49	1	7365	1147869033	None	None	None	None	None
8	1	1237	1147868839	None	None	None	None	None
32	1	5684	1147879797	None	None	None	None	None
60	1	8685	1147878023	None	None	None	None	None
52	1	7938	1147878063	None	None	None	None	None
33	1	5767	1147878729	None	None	None	None	None
64	1	8973	1147869211	None	None	None	None	None
12	1	2011	1147868079	None	None	None	None	None

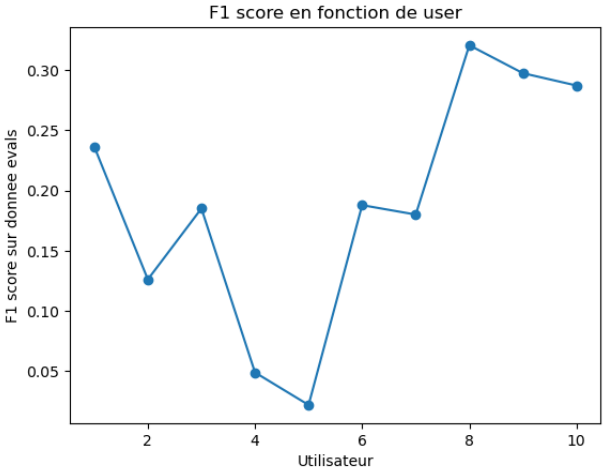
Ci-dessous la procédure suivie, pour implémenter cette solution et quelques sorties de la fonction de prédiction.

- Recherche des N utilisateur dans les deux clusters distincts et prédiction de leur vote individuellement, sur les films regardés et non regardés.

  On a en sortie ici, les meilleurs hyperparamètres pour chaque individu et les F1-scores :

  Cluster 0 :

	criterion	max_depth	min_samples_leaf	min_samples_split	splitter
0	gini	None	1	2	best
1	gini	None	1	2	best
2	gini	None	1	2	best
3	gini	None	1	2	best
4	gini	None	1	2	best
5	gini	None	1	2	best
6	gini	None	1	2	best
7	gini	None	1	2	best
8	gini	None	1	2	best
9	gini	None	1	2	best



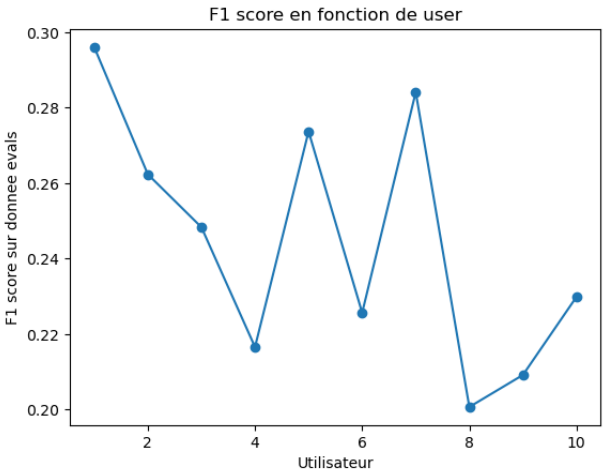
	7325.0	8781.0	1882.0	115713.0	6373.0	66097.0	8969.0	112623.0	106002.0	55765.0	...	30707.0	48516.0	50068.0	51255.0	69481.0	72011.0	50.0
0	3.0	3.0	4.0	3.0	1.0	3.0	3.0	4.0	3.0	3.0	...	NaN	NaN	NaN	NaN	NaN	NaN	NaN
1	3.0	3.0	3.0	3.0	3.0	2.0	4.0	3.0	3.0	3.0	...	NaN	NaN	NaN	NaN	NaN	NaN	NaN
2	3.0	3.0	2.0	3.0	3.0	2.0	4.0	2.0	3.0	3.0	...	3.0	3.0	3.0	3.0	2.0	4.0	NaN
3	3.0	3.0	2.0	3.0	3.0	2.0	4.0	3.0	3.0	3.0	...	3.0	3.0	3.0	3.0	2.0	4.0	4.0
4	2.0	3.0	NaN	3.0	4.0	2.0	3.0	3.0	3.0	3.0	...	3.0	3.0	3.0	3.0	3.0	3.0	4.0
5	NaN	4.0	2.0	3.0	NaN	2.0	4.0	3.0	3.0	3.0	...	3.0	3.0	3.0	3.0	2.0	4.0	NaN
6	3.0	3.0	2.0	3.0	3.0	2.0	4.0	2.0	3.0	3.0	...	3.0	3.0	3.0	3.0	2.0	4.0	4.0
7	3.0	3.0	2.0	3.0	NaN	NaN	4.0	2.0	3.0	NaN	...	3.0	NaN	3.0	3.0	2.0	4.0	3.0
8	NaN	3.0	2.0	3.0	3.0	2.0	4.0	3.0	3.0	3.0	...	3.0	3.0	3.0	4.0	2.0	NaN	2.0
9	3.0	3.0	NaN	3.0	3.0	2.0	4.0	3.0	3.0	3.0	...	3.0	3.0	3.0	3.0	2.0	4.0	NaN

10 rows × 57361 columns

Une valeur NaN, implique l'impossibilité de prédire le vote, car même les 5 plus proches utilisateur n'ont pas regardé le film en question.

Cluster 1 :

	criterion	max_depth	min_samples_leaf	min_samples_split	splitter
0	gini	None	1	2	best
1	gini	None	1	2	best
2	gini	None	1	2	best
3	gini	None	1	2	best
4	gini	None	1	2	best
5	gini	None	1	2	best
6	gini	None	1	2	best
7	gini	None	1	2	best
8	gini	None	1	2	best
9	gini	None	1	2	best



- Ensuite on moyenne les F1-scores et on cherche les hyperparamètres populaires :

Cluster 0 : **Cluster 1 :**

```
Meilleurs hyperparametre apres validation :
 gini
 NaN
   1
   2
best
```

```
Meilleurs hyperparametre apres validation :
 gini
 NaN
   1
   2
best
```

```
F1-score pour les utilisateurs appartenant a la classe 0, obtenu sur les donnees evals :  0.18928140855377013
```

```
F1-score pour les utilisateurs appartenant a la classe 1, obtenu sur les donnees evals :  0.2490289770522513
```

```
F1-score pour l'ensemble des utilisateurs, obtenu sur les donnees evals :  0.2195212386360011
```

- Puis on moyenne également les votes résultants, pour ces N utilisateurs. Il faudrait ici souligner que le moyennage, permet d'obtenir les votes d'un utilisateur, même s'il n'a pas regardé le même film, et aussi ses plus proches utilisateurs. Il acquiert ce vote, de la part des autres membres de son cluster.

Cluster 0 : **Cluster 1 :**

Moyenne sur toutes les lignes :

	vote
1.0	2.0
2.0	3.0
3.0	3.0
4.0	3.0
5.0	3.0
...	...
209155.0	3.0
209157.0	3.0
209159.0	3.0
209163.0	3.0
209171.0	3.0

57361 rows × 1 columns

Moyenne sur toutes les lignes :

	vote
1.0	3.0
2.0	3.0
3.0	3.0
4.0	3.0
5.0	3.0
...	...
209155.0	3.0
209157.0	3.0
209159.0	3.0
209163.0	3.0
209171.0	3.0

57361 rows × 1 columns

- Enfin on généralise le vote sur l'ensemble des utilisateurs des 2 clusters dans les données initiales.

	0	1	2	3	4	5	6	7	8	9	...	2575	2576	2577	2578	2579	2580	2581	2582	2583	Cluster
userId																					
1	4.0	2.0	3.0	3.0	4.0	4.0	4.0	4.0	5.0	4.0	...	4.0	4.0	4.0	3.0	3.0	3.0	3.0	4.0	3.0	0.0
2	4.0	2.0	3.0	3.0	4.0	4.0	4.0	4.0	5.0	4.0	...	4.0	4.0	4.0	3.0	3.0	3.0	3.0	4.0	3.0	1.0
3	3.0	3.0	3.0	4.0	3.0	4.0	5.0	2.0	5.0	4.0	...	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	0.0
4	4.0	2.0	3.0	3.0	4.0	4.0	4.0	4.0	5.0	4.0	...	4.0	4.0	4.0	3.0	3.0	3.0	3.0	4.0	3.0	0.0
5	4.0	2.0	3.0	3.0	4.0	4.0	4.0	4.0	5.0	4.0	...	4.0	4.0	4.0	3.0	3.0	3.0	3.0	4.0	3.0	0.0
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
162537	4.0	2.0	3.0	3.0	4.0	4.0	4.0	4.0	5.0	4.0	...	4.0	4.0	4.0	3.0	3.0	3.0	3.0	4.0	3.0	0.0
162538	4.0	2.0	3.0	3.0	4.0	4.0	4.0	4.0	5.0	4.0	...	4.0	4.0	4.0	3.0	3.0	3.0	3.0	4.0	3.0	0.0
162539	4.0	2.0	3.0	3.0	4.0	4.0	4.0	4.0	5.0	4.0	...	4.0	4.0	4.0	3.0	3.0	3.0	3.0	4.0	3.0	0.0
162540	4.0	2.0	3.0	3.0	4.0	4.0	4.0	4.0	5.0	4.0	...	4.0	4.0	4.0	3.0	3.0	3.0	3.0	4.0	3.0	0.0
162541	4.0	2.0	3.0	3.0	4.0	4.0	4.0	4.0	5.0	4.0	...	4.0	4.0	4.0	3.0	3.0	3.0	3.0	4.0	3.0	NaN

162541 rows × 2585 columns

La sortie finale de notre entrainement-validation, c'est le meilleur modèle accompagné de ses meilleurs hyperparamètres et du meilleur F1-score résultant. F1-score de 0.22, laisse croire que le modèle n'est pas suffisamment performant, du certainement à 2 raisons :

- Arbre de décision, inapproprié pour la prédiction.
- La généralisation par moyennage appliquée, cause de manque de ressource.

On va toutefois s'en servir pour le tester sur les données tests et calculer le F1-score.

b. Prédiction et évaluation du fichier ratings test.csv :

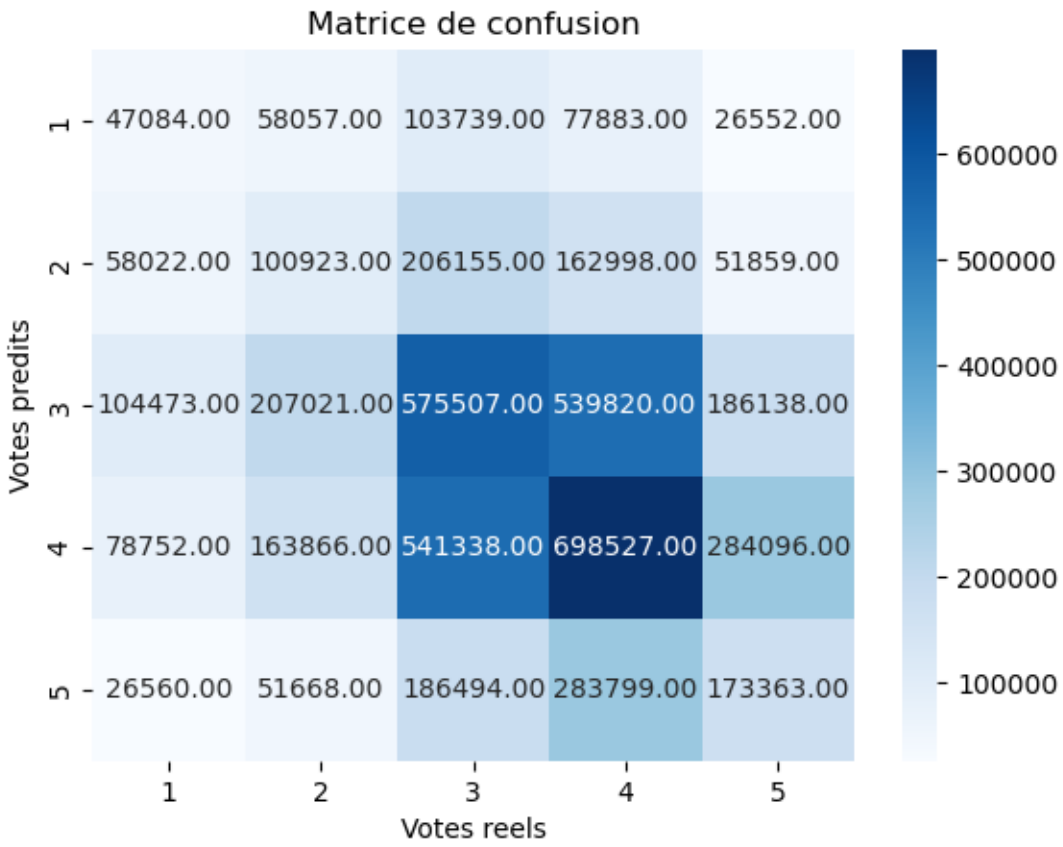
A partir du prédicteur conçu précédemment, D() nous prédisons les votes des utilisateurs qui se trouvent dans le fichier ratings test.csv, tout en évaluant la performance du modèle suivant le critère de F1-score. Ci-dessous un aperçu de la prédiction :

	UserId	movieId	Rate_predict
3360370	1	7318	4.0
2270698	1	7209	4.0
3218004	1	4144	3.0
4408888	1	307	5.0
3253731	1	1088	3.0
...	...	...	...
1921049	162541	4080	3.0
3560178	162541	6440	3.0
2443876	162541	2105	4.0
4946165	162541	616	2.0
3884873	162541	6548	1.0

4994694 rows × 3 columns

Il faut noter ici, qu'on ne s'intéressera qu'au vote des films regardes, dans l'intention de pouvoir évaluer notre modèle. Ci-dessous la matrice de confusion associée ainsi que d'autre critères d'évaluation.

Le dataframe serait en accompagnement à ce rapport, ainsi que le fichier code.



	precision	recall	f1-score	support
1.0	0.15	0.15	0.15	313315
2.0	0.17	0.17	0.17	579957
3.0	0.36	0.36	0.36	1612959
4.0	0.40	0.40	0.40	1766579
5.0	0.24	0.24	0.24	721884
accuracy			0.32	4994694
macro avg	0.26	0.26	0.26	4994694
weighted avg	0.32	0.32	0.32	4994694

```
F1-score pour l'ensemble des utilisateurs, obtenu sur les donnees de test :  0.31949664145554185
```

Nous terminons avec un f1-score faible de **0.32** sur cet ensemble test. Le modèle semble ne pas bien performer sur nos données test, tout comme sur les données d'entrainement, vu l'abstention de l'utilisation de certaines données qu'on a faite, en les remplaçant par une sorte de généralisation.

Il pourrait à présent être déployer en ligne, pour mieux évaluer ce dernier ci, en utilisant de nouvelles données des utilisateurs, et donc on pourrait actualiser le modèle de façon à améliorer son f1-score, en permanence.

Conclusion

Ce rapport de TP sur les systèmes de recommandation a permis une exploration approfondie des données, ainsi que la mise en œuvre de diverses techniques et modèles pour la construction d'un système de recommandation. Nous avons réalisé des statistiques descriptives, extrait de nouveaux jeux de données, construit des matrices de contenu et de profil d'utilisateurs, et utilisé le clustering spectral pour limiter la recherche d'items et d'utilisateurs. L'entraînement et l'évaluation du modèle de prédiction des votes des utilisateurs ont également été abordés, avec une attention particulière portée à l'évaluation des performances à l'aide du F1-score.

Malgré les défis rencontrés, tels que la présence de données non balancées et l'identification de modèles sous-performant, notre analyse a permis de mettre en lumière des pistes d'amélioration pour le déploiement futur du système de recommandation. En effet, la prédiction et l'évaluation sur le fichier ratings test.csv ont révélé un F1-score de 0.32, indiquant une performance faible du modèle sur cet ensemble de test. Cela souligne la nécessité d'actualiser le modèle en permanence en utilisant de nouvelles données d'utilisateurs pour améliorer sa précision et son efficacité (évaluation en ligne).

En conclusion, ce rapport a été une opportunité d'appliquer les concepts et les techniques appris dans le cadre du cours de Sciences des Données, et a mis en lumière l'importance de l'exploration approfondie des données, de la modélisation précise et de l'évaluation rigoureuse des performances pour la construction de systèmes de recommandation efficaces.