M2 IASD

Projet data science :

SYSTEME DE RECOMMANDATION (COLLABORATIVE FILTERING)

POUR LA RECOMMANDATION DES FILMS

Réalisé par :

Ben Ammar Aziz

Gaidi Mohamed Amine

Salem Amenallah

Aouadni Amel

encadré par :

Ben Amor Nahla

Elouedi Zied

2019/2020

Table des matières

1 Introduction :

On se demande tous comment Amazon ou Netflix sont arrivés à ce « power » et ce succès ? Comment Netflix peut connaître nos préférences cinématographiques ? Comment Amazon a su que je suis un fan inconditionnel de Games of Thrones, The North Face et de géographie ?

Sans dévoiler un secret, ce sont les systèmes de recommandations ! Aujourd'hui, Netflix sait tous sur ses utilisateurs : ce qu'ils regardent, leur vidéo préférée ainsi que la fréquence à laquelle il regarde films et séries. Cette gigantesque base de données a pu booster leur algorithme.

2 C'est quoi un système de recommandation ?

Tout d'abord, un système de recommandation est un outil de business, cela renforce jusqu'à 30% les revenus d'une entreprise. Aujourd'hui, un utilisateur ne veut pas qu'on lui propose sur internet des produits qu'il a déjà achetés ou bien qui ne lui intéressent pas. C'est pourquoi les systèmes de recommandation ont pour objectif de comprendre le comportement de l'utilisateur. Cela facilitera sa vie et le site ou l'application gagnera sa confiance.

Aujourd'hui de nombreux secteurs utilisent les systèmes de recommandation. Pour preuve, ils sont présents dans les boutiques en ligne comme Amazon, les services de streaming comme Netflix ou Spotify ou encore les systèmes de recommandation spécifiques pour la publicité basée sur du contenu. Ces systèmes de recommandation partagent le même principe qui est de filtrer à l'avance parmi une grande masse d'objets susceptibles d'intéresser l'acheteur (qu'il s'agisse de produits, livres, films etc.).

Il existe plusieurs types de filtrage pour la recommandation. Le système de filtrage d'informations achemine l'information aux utilisateurs en se basant sur leurs profils. Ces derniers sont établis grâce à des techniques d'apprentissage des goûts des utilisateurs.

Traditionnellement, les systèmes de filtrage d'informations ont été classés en trois catégories : les systèmes à base de contenu, les systèmes de filtrage collaboratif et les systèmes de filtrage hybrides. Cette classification dépend de la manière avec laquelle l'utilité ou la pertinence éventuelle est calculée ou estimée.

Notre projet consiste à implémenter un algorithme de recommandation de films basé sur le filtrage collaboratif.

3 Filtrage collaboratif (COLLABORATIVE FILTERING) :

C'est la recommandation des produits basés sur les intérêts d'un grand groupe d'utilisateurs. Cette méthode essaie de trouver un groupe d'utilisateurs qui possède les mêmes goûts et préférences de l'utilisateur cible. Ainsi, il utilise ce groupe pour recommander leurs produits. En partant de l'hypothèse que les utilisateurs avec des goûts similaires ont les mêmes préférences.

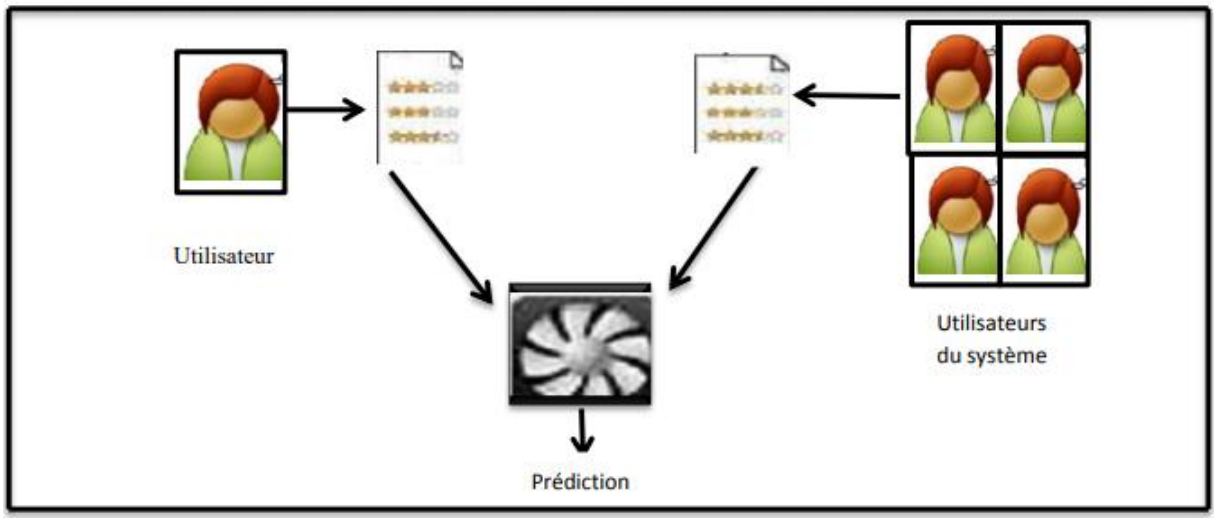


Figure 1 : Filtrage collaboratif

Le filtrage collaboratif utilise des méthodes statistiques pour faire des prévisions basées sur l'évaluation des intérêts des utilisateurs. Ces prévisions sont utilisées pour faire des propositions en se basant sur la corrélation entre son profil personnel et les profils des autres utilisateurs (qui présentent des intérêts semblables).

L'utilisateur peut alors demander au système de :

•Lui suggérer une ressource susceptible de lui plaire.

• Le prévenir des ressources qu'il ne devrait pas apprécier.

•Donner une estimation d'évaluation d'une certaine ressource.

Le schéma suivant montre l'architecture globale d'un système de filtrage collaboratif.

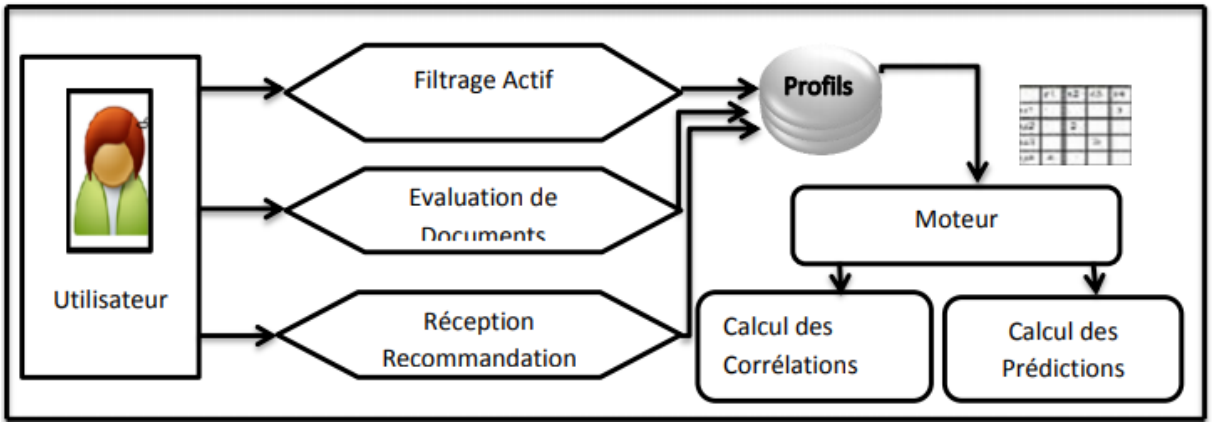


Figure 2 : Processus de filtrage collaboratif.

Donc on utilise deux approches dans le filtrage collaboratif :

- Se baser sur l'utilisateur (user-based recommendation)
- Se baser sur le produit (item-based recommendation)

4 Pré-traitement :

On commence par le téléchargement des deux dataset :

```
ratings = pd.read_csv('/content/drive/My Drive/ml-latest-small/ratings.csv')
ratings = ratings.sort_values(by='movieId', ascending=True)
```

Après la première partie du pré-traitement avec la data wrangling. Donc on va Supprimer les films qui sont évalués par moins de 6 utilisateurs de la base small_ratings.

```
r = ratings['movieId'].value_counts().sort_values(ascending=True)
r=r.sort_index()
L=[]
for i in range(1,max(ratings['movieId'])):
  try:
    if r[i] <6:
      L.append(i)
  except:
    pass

ratings=ratings.reset_index()
for i in range(len(L)):
  try:
    ratings.drop(ratings['movieId'][L[i]], axis=0, inplace=True)
  except:
    pass
```

	index	userId	movieId	rating	timestamp
0	0	1	1	4.0	964982703
4	30601	214	1	3.0	853937855
8	80373	509	1	4.0	1435992343
14	29936	206	1	5.0	850763267
23	77950	484	1	4.5	1342295949
...	...	...	...	...	...
100831	27256	184	193581	4.0	1537109082
100832	27257	184	193583	3.5	1537109545
100833	27258	184	193585	3.5	1537109805
100834	27259	184	193587	3.5	1537110021
100835	51362	331	193609	4.0	1537157606

99564 rows × 5 columns

Après la deuxième partie du pré-traitement qui s'agit de supprimer les utilisateurs qui donnent la même note aux films qu'ils ont évalués dans la base small_ratings.

```
[ ]  ratings.drop(columns='index')
     ratings.sort_values('userId', ascending=True)
```

```
r2 = ratings['userId'].value_counts().sort_values(ascending=True)
r2=r2.sort_index()
L=[]
for i in range(1,max(ratings['userId'])):
  mask1 = ratings['userId'] == i
  mask2 = max(ratings['rating']) == min(ratings['rating'])
  if max(ratings[mask1]['rating']) == min(ratings[mask1]['rating']):
    removeUserId=i
  else:
    pass

for i in range(1,max(ratings['userId'])):
  try:
    if r2[i] == removeUserId:
      L.append(i)
  except:
    pass
ratings=ratings.reset_index()
for i in range(len(L)):
  try:
    ratings.drop(ratings['userId'][L[i]], axis=0, inplace=True)
  except:
    pass
ratings.drop(['index','level_0'], axis=1, inplace=True)
```

	userId	movieId	rating	timestamp
0	1	1	4.0	964982703
1	214	1	3.0	853937855
2	509	1	4.0	1435992343
3	206	1	5.0	850763267
4	484	1	4.5	1342295949
...	...	...	...	...
99559	184	193581	4.0	1537109082
99560	184	193583	3.5	1537109545
99561	184	193585	3.5	1537109805
99562	184	193587	3.5	1537110021
99563	331	193609	4.0	1537157606

99559 rows × 4 columns

5 Data exploration/data visualization :

```
movies = pd.read_csv('/content/drive/My Drive/ml-latest-small/movies.csv')
movies
```

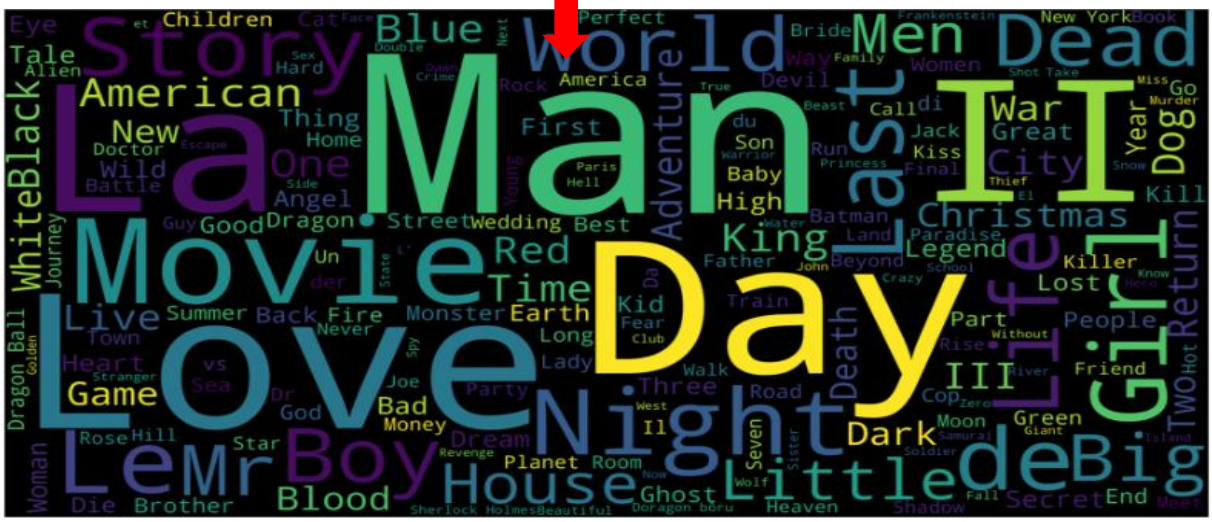
	movieId	title	genres
0	1	Toy Story (1995)	Adventure|Animation|Children|Comedy|Fantasy
1	2	Jumanji (1995)	Adventure|Children|Fantasy
2	3	Grumpier Old Men (1995)	Comedy|Romance
3	4	Waiting to Exhale (1995)	Comedy|Drama|Romance
4	5	Father of the Bride Part II (1995)	Comedy
...	...	...	...
9737	193581	Black Butler: Book of the Atlantic (2017)	Action|Animation|Comedy|Fantasy
9738	193583	No Game No Life: Zero (2017)	Animation|Comedy|Fantasy
9739	193585	Flint (2017)	Drama
9740	193587	Bungo Stray Dogs: Dead Apple (2018)	Action|Animation
9741	193609	Andrew Dice Clay: Dice Rules (1991)	Comedy

9742 rows × 3 columns

```
[ ] # Import new libraries
    %matplotlib inline
    import wordcloud
    from wordcloud import WordCloud, STOPWORDS
    import matplotlib.pyplot as plt

    # Create a wordcloud of the movie titles
    movies['title'] = movies['title'].fillna("").astype('str')
    title_corpus = ' '.join(movies['title'])
    title_wordcloud = WordCloud(stopwords=STOPWORDS, background_color='black', height=2000, width=4000).generate(title_corpus)

    # Plot the wordcloud
    plt.figure(figsize=(16,8))
    plt.imshow(title_wordcloud)
    plt.axis('off')
    plt.show()
```

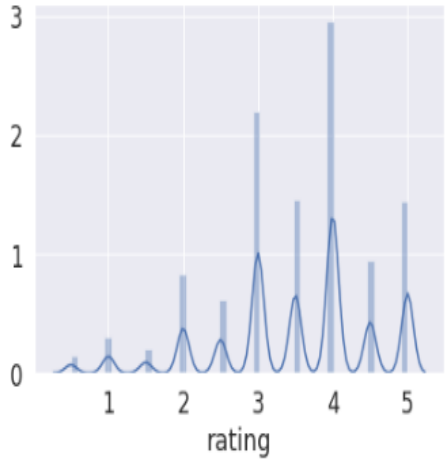
— — — — — — — — — — — — — —

Statistiques et distribution.

```
[ ] # Import seaborn library
    import seaborn as sns
    sns.set_style('whitegrid')
    sns.set(font_scale=1.5)
    %matplotlib inline

    # Display distribution of rating
    sns.distplot(ratings['rating'].fillna(ratings['rating'].median()))
```

```
/usr/local/lib/python3.6/dist-packages/statsmodels/tools/_testing.py:19: FutureWarning: pandas.util.testing is deprecated. Use the
  import pandas.util.testing as tm
<matplotlib.axes._subplots.AxesSubplot at 0x7f0b7bcde8d0>
```



©

Il semble que les utilisateurs sont assez généreux dans leurs évaluations. La note moyenne est de 3,58 sur une échelle de 5. La moitié des films ont une note de 4 et 5. W personnellement pense qu'une compétence de notation de 5 niveaux n'était pas un bon indicateur que les gens pourraient avoir des styles de notation différents (c'est-à-dire la personne A pourrait toujours utiliser 4 pour un film moyen, tandis que la personne B donne seulement 4 pour leurs favoris). Chaque utilisateur a évalué au moins 20 films, donc nous doutons que la distribution pourrait être causée juste par la variance par hasard dans la qualité des films. Jetons également un coup d'oeil à un sous-ensemble de 20 films avec la cote la plus élevée.

— — — — — — — — — — — — — —

Joignez-nous à l'ensemble des 3 fichiers dans un cadre de données et Joignez-nous à tous les 3 fichiers dans un Afficher 20 films avec les cotes les plus élevées

```
# Join all 3 files into one dataframe
dataset = pd.merge(movies, ratings)
# Display 20 movies with highest ratings
dataset[['title','genres','rating']].sort_values('rating', ascending=False).head(20)
```

	title	genres	rating
27283	Deer Hunter, The (1978)	Drama|War	5.0
12850	Philadelphia (1993)	Drama	5.0
77561	Scanner Darkly, A (2006)	Animation|Drama|Mystery|Sci-Fi|Thriller	5.0
54476	Gladiator (2000)	Action|Adventure|Drama	5.0
32071	Men in Black (a.k.a. MIB) (1997)	Action|Comedy|Sci-Fi	5.0
12834	Philadelphia (1993)	Drama	5.0
47404	Perfect Blue (1997)	Animation|Horror|Mystery|Thriller	5.0
12836	Philadelphia (1993)	Drama	5.0
32068	Men in Black (a.k.a. MIB) (1997)	Action|Comedy|Sci-Fi	5.0
23969	Princess Bride, The (1987)	Action|Adventure|Comedy|Fantasy|Romance	5.0
47425	Three Days of the Condor (3 Days of the Condor...	Drama|Mystery|Romance|Thriller	5.0
47427	Three Days of the Condor (3 Days of the Condor...	Drama|Mystery|Romance|Thriller	5.0
12845	Philadelphia (1993)	Drama	5.0
23966	Princess Bride, The (1987)	Action|Adventure|Comedy|Fantasy|Romance	5.0
47430	Three Days of the Condor (3 Days of the Condor...	Drama|Mystery|Romance|Thriller	5.0
23965	Princess Bride, The (1987)	Action|Adventure|Comedy|Fantasy|Romance	5.0
12827	Philadelphia (1993)	Drama	5.0
59333	3000 Miles to Graceland (2001)	Action|Thriller	5.0

La variable des genres sera sûrement importante lors de la construction des moteurs de recommandation puisqu'elle décrit le contenu du film (c.-à-d. Animation, Horreur, Sci-Fi). Une hypothèse de base est que les films dans le même genre devraient avoir des contenus similaires. Nous allons essayer de voir exactement quels genres sont les plus populaires.

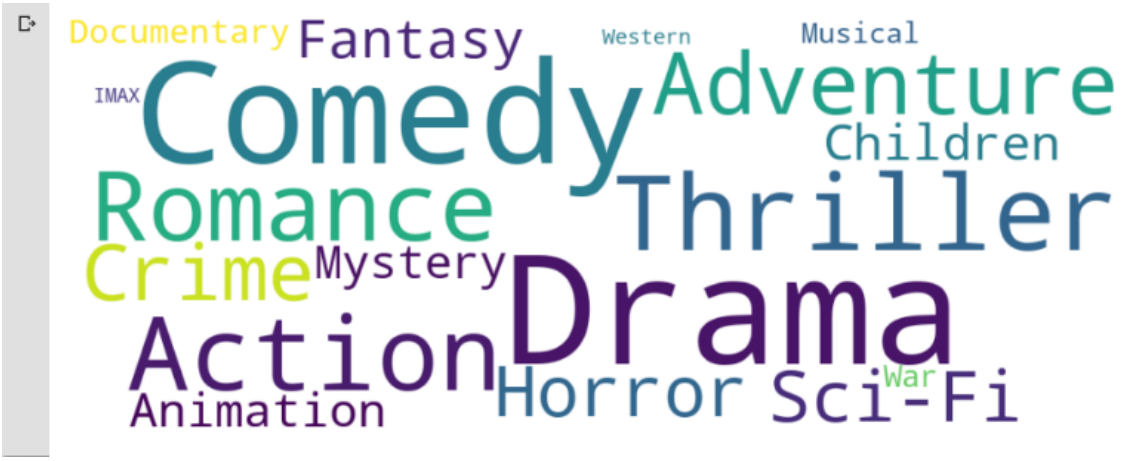
```
[['Drama', 4361],
 ['Comedy', 3756],
 ['Thriller', 1894],
 ['Action', 1828],
 ['Romance', 1596]]
```

Les 5 principaux genres sont, à cet égard ordre: Drame, Comédie, Action, Thriller, et Romance. Je vais montrer cela sur un wordcloud trop afin de le rendre plus visuellement attrayant.

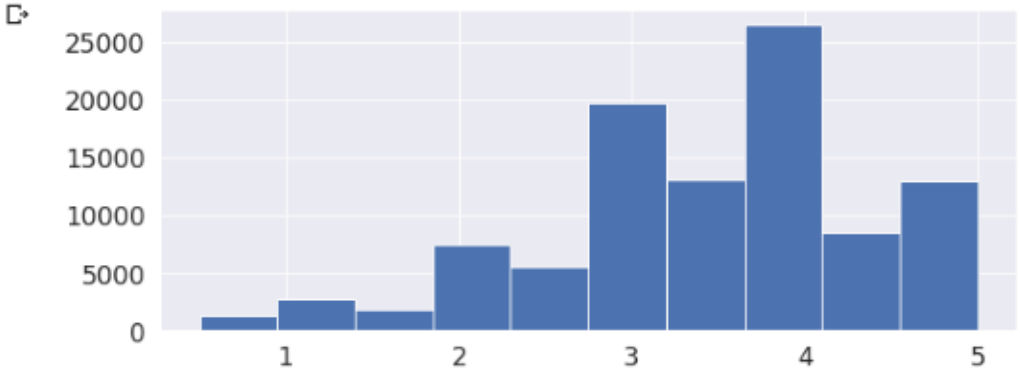
```
# Define the dictionary used to produce the genre wordcloud
genres = dict()
trunc_occurences = keyword_occurences[0:18]
for s in trunc_occurences:
    genres[s[0]] = s[1]

# Create the wordcloud
genre_wordcloud = WordCloud(width=1000,height=400, background_color='white')
genre_wordcloud.generate_from_frequencies(genres)

# Plot the wordcloud
f, ax = plt.subplots(figsize=(16, 8))
plt.imshow(genre_wordcloud, interpolation="bilinear")
plt.axis('off')
plt.show()
```




```
plt.figure(figsize=(10,4))
ratings['rating'].hist(bins=10);
```



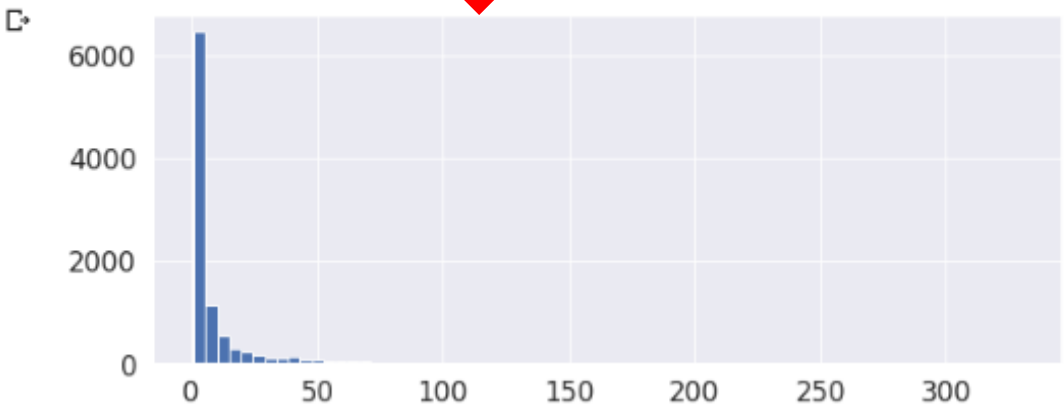
Nous pouvons voir de l'intrigue histogramme que la cote de film la plus fréquente donnée est de 4. Il en résulte également une variable cible déséquilibrée, donc nous devrons nous attaquer à cette question lors de l'apprentissage automatique.

```
df = pd.merge(ratings,movies,on='movieId')
df.groupby('title')['rating'].mean().sort_values(ascending=False).head()
df.groupby('title')['rating'].count().sort_values(ascending=False).head()
grading = pd.DataFrame(df.groupby('title')['rating'].mean())
grading['num of ratings'] = pd.DataFrame(df.groupby('title')['rating'].count())
grading
```

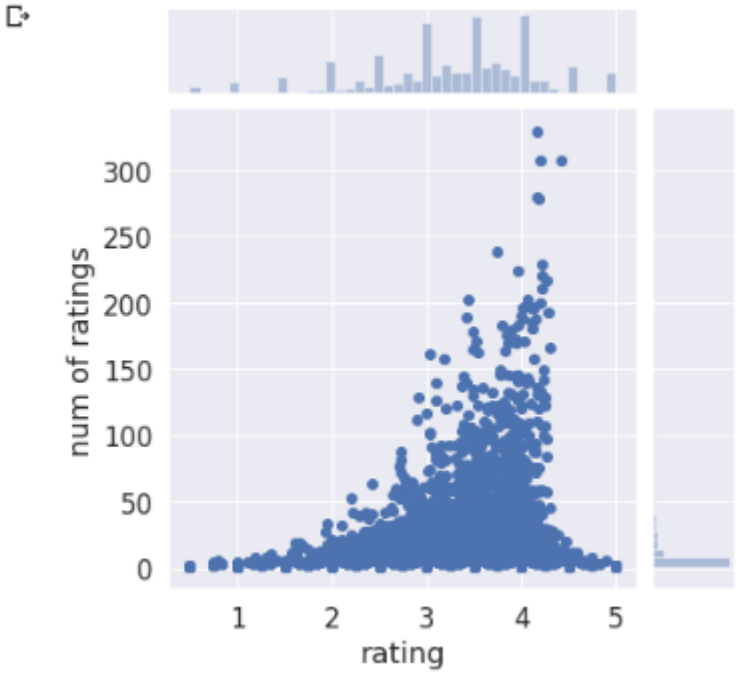
title	rating	num of ratings
'71 (2014)	4.000000	1
'Hellboy': The Seeds of Creation (2004)	4.000000	1
'Round Midnight (1986)	3.500000	2
'Salem's Lot (2004)	5.000000	1
'Til There Was You (1997)	4.000000	2
...	...	...
eXistenZ (1999)	3.863636	22
xXx (2002)	2.770833	24
xXx: State of the Union (2005)	2.000000	5
¡Three Amigos! (1986)	3.120000	25
À nous la liberté (Freedom for Us) (1931)	1.000000	1

9705 rows × 2 columns

```
[ ] plt.figure(figsize=(10,4))
    grading['num of ratings'].hist(bins=70);
```



```
[ ]  sns.jointplot(x='rating',y='num of ratings',data=grading);
```



6 RECOMMANDATION BASÉE SUR L'UTILISATEUR ((user-based recommendation)) :

Cette méthode identifie les utilisateurs qui sont similaires à l'utilisateur interrogé et estime la note souhaitée comme la moyenne pondérée des évaluations de ces utilisateurs similaires.

Eh bien, UB-CF utilise cette logique et recommande des éléments en trouvant des utilisateurs similaires à l'utilisateur actif (à qui nous essayons de recommander un film). Une application spécifique de ceci est l'algorithme voisin proche basé sur l'utilisateur.

Dans cette approche, on construit une matrice A : [Utilisateur x Produit]

A	Item 1	Item 2	Item 3	Item 4	Item 5
User 1	9	3	?	7	10
User 2		2	6	7	9
User 3	9	3	9	7	
User 4	6	6		2	1
User 5		7	9	3	4

Dans cette matrice, on trouve qu'User 1 et User 2 sont corrélés (3 sur 5 produits ont le même score). C'est ainsi qu'on recommande à l'un des utilisateurs les produits préférés du deuxième. Mais comment calculer la corrélation ou la similarité entre deux utilisateurs ?

Les méthodes de similarité :

- **Cosine similarity :**

$$\mu = \cos(\text{User 1}, \text{User 2}) = \cos(\alpha) = \frac{User1 \,.User\ 2}{||User\ 1||\ ||User\ 2||}$$

La corrélation de deux utilisateurs est importante pour des valeurs de u aussi importantes

- **Jaccard similarity :**

$$J(\text{User 1}, \text{User 2}) = \frac{|User\ 1 \cap User\ 2|}{|User\ 1 \cup User\ 2|}$$

Cette méthode consiste à ignorer les scores

- **Fonction score :**

Il est facile de trouver une fonction pour le filtrage collaboratif non personnalisé (c.-à-d. que nous ne considérons pas les goûts, les aversions et la notation de l'utilisateur actif du passé) qui renvoie un score prenant utilisateur vous et l'élément i comme paramètres d'entrée.

La fonction produit un score qui quantifie la force d'un utilisateur que vous aimez / préfère l'élément i.

Nous avons utilisé dans notre modèle les bibliothèque surprise ,collections, operator

Surprise est une librairie Python pour l'analyse des systèmes de recommandation qui traitent des données d'évaluation explicites.

```
[ ]  !pip install surprise
     from surprise import KNNBasic, Reader, Dataset
     from collections import defaultdict
     from operator import itemgetter
     import csv, sys, os
     import heapq
```

7 RECOMMANDATION BASÉE SUR LE PRODUIt (item-based recommendation)

Dans cette approche, on construit aussi la même matrice A. Mais la différence se présente par corrélation entre les produits. On cherche les produits qui sont potentiellement corrélés. Et on offre un produit qui est rattaché aux autres produits (ayant le plus de degré de corrélation).

Le filtrage collaboratif basé sur les éléments est un algorithme basé sur un modèle pour formuler des recommandations. Dans l'algorithme, les similitudes entre les différents éléments de l'ensemble de données sont calculées à l'aide d'une des nombreuses mesures de similitude, puis ces valeurs de similitude sont utilisées pour prédire les évaluations des paires d'objets utilisateur qui ne sont pas présentes dans le jeu de données.

Les valeurs de similitude entre les éléments sont mesurées en observant tous les utilisateurs qui ont évalué les deux éléments. La similitude entre deux éléments dépend des évaluations attribuées aux éléments par les utilisateurs qui ont évalué les deux.

8 KNN :

Pour implémenter un filtre collaboratif basé sur un élément, KNN est un modèle parfait de référence et aussi une très bonne base de base pour le développement du système de recommandation. Mais qu'est-ce que le KNN?

KNN est une méthode d'apprentissage non paramétrique et paresseuse. Elle utilise une base de données dans laquelle les points de données sont séparés en plusieurs clusters pour faire l'inférence pour de nouveaux échantillons.

KNN ne fait aucune hypothèse sur la distribution de données sous-jacente, mais il s'appuie sur la similitude de l'élément. Lorsque KNN fait une inférence sur un film, KNN calculera la «distance» entre le film cible et tous les autres films dans sa base de données, puis il classe ses

distances et retourne les meilleurs films voisins K plus proche comme les recommandations de films les plus similaires.

9 Evaluation user KNN/ itemKNN/Random:

Loading movie ratings...

Computing movie popularity ranks so we can measure novelty later...

Estimating biases using als...

Computing the cosine similarity matrix...

Done computing similarity matrix.

Evaluating User KNN ...

Evaluating accuracy...

Computing the cosine similarity matrix...

Done computing similarity matrix.

Analysis complete.

Evaluating Item KNN ...

Evaluating accuracy...

Computing the cosine similarity matrix...

Done computing similarity matrix.

Analysis complete.

Evaluating Random ...

Evaluating accuracy...

Analysis complete.

Algorithm	RMSE	MAE
User KNN	0.9802	0.7560
Item KNN	0.9749	0.7582
Random	1.4115	1.1256

Legend:

RMSE: Root Mean Squared Error. Lower values mean better accuracy.

MAE: Mean Absolute Error. Lower values mean better accuracy.

- **Using recommender User KNN :**

Building recommendation model...

Computing the cosine similarity matrix...

Done computing similarity matrix.

Computing recommendations...

- We recommend:

Heidi Fleiss: Hollywood Madam (1995) 5

Awfully Big Adventure, An (1995) 5

In the Realm of the Senses (Ai no corrida) (1976) 5

What Happened Was... (1994) 5

Denise Calls Up (1995) 5

- **Using recommender Item KNN**

Building recommendation model...

Computing the cosine similarity matrix...

Done computing similarity matrix.

Computing recommendations...

- We recommend:

Two if by Sea (1996) 5

Awfully Big Adventure, An (1995) 5

Love & Human Remains (1993) 5

Fluke (1995) 5

Inkwell, The (1994) 5

- **Using recommender Random**

Building recommendation model...

Computing recommendations...

- We recommend:

Shanghai Triad (Yao a yao yao dao waipo qiao) (1995) 5

10 Conclusion :

Depuis quelques années, les systèmes de recommandation ont une place particulièrement importante dans le marketing en ligne. Grâce à eux, des entreprises du e-commerce ont pu se différencier de leurs concurrents, faciliter la vie des clients actuels et atteindre leurs clients potentiels.

Selon les stratégies des entreprises, plusieurs techniques de recommandation sont intégrées pour adapter les besoins. Comme nous avons pu le voir dans notre travail, ces méthodes ont différents avantages et inconvénients, aucune solution ente eux ne peut donc répondre à toutes les problématiques. En réalité, les entreprises utilisent plusieurs approches et les combinent pour avoir une meilleure recommandation, évaluée par certains critères prédéfinis dans leur contexte ainsi que leurs objectifs.

Si le fonctionnement d'un système de recommandation est assez simple, sa mise en place est toutefois compliquée. Ces difficultés résident dans certains aspects comme la collection et la sélection de données pertinentes, la taille et la qualité de données, la rareté des données, la construction de profils utilisateurs, la prédiction pour des nouveaux profils d'utilisateurs ou de nouveaux produits.