M2 IASD

data science Projet :

RECOMMENDATION SYSTEM (COLLABORATIVE FILTERING)

FOR THE RECOMMENDATION OF THE FILMS

Directed by:

Ben ammar aziz

Gaidi Mohamed amine

Salem amenallah

Aouedni amal

framed by:

Ben amor Nahla

Elouedi Zied

2019/2020

Table des matières

1 Introduction :

We all wonder how Amazon or Netflix got to this "power" and success? How can Netflix know our cinematic preferences? How did Amazon know I'm an avid fan of Games of Thrones, The North Face and Geography?

Without revealing a secret, these are the systems of recommendations! Today, Netflix all knows about its users: what they watch, their favorite video and how often it watches movies and series. This huge database was able to boost their algorithm.

What's a recommendation system?

First of all, a recommendation system is a business tool, it strengthens up to 30% of a company's revenues. Today, a user does not want to be offered on the internet products that he has already bought or that he is not interested in. That's why recommendation systems aim to understand user behavior. This will make his life easier and the site or app will gain his trust.

Today, many sectors use recommendation systems. As proof, they are present in online stores such as Amazon, streaming services like Netflix or Spotify or specific recommendation systems for content-based advertising. These recommendation systems share the same principle of filtering in advance among a large mass of objects that may be of interest to the buyer (whether they are products, books, films, etc.).

There are several types of filtering for the recommendation. The information filtering system delivers information to users based on their profiles. These are established through techniques to learn the tastes of users.

Traditionally, information filtering systems have been categorized into three categories: content-based systems, collaborative filtering systems and hybrid filtering systems. This classification depends on how the potential utility or relevance is calculated or estimated.

Our project is to implement a film recommendation algorithm based on collaborative filtering.

2 COLLABORATIVE FILTERING :

This is the recommendation of products based on the interests of a large group of users. This method tries to find a group of users who have the same tastes and preferences of the target user. Thus, he uses this group to recommend their products. Based on the assumption that users with similar tastes have the same preferences.

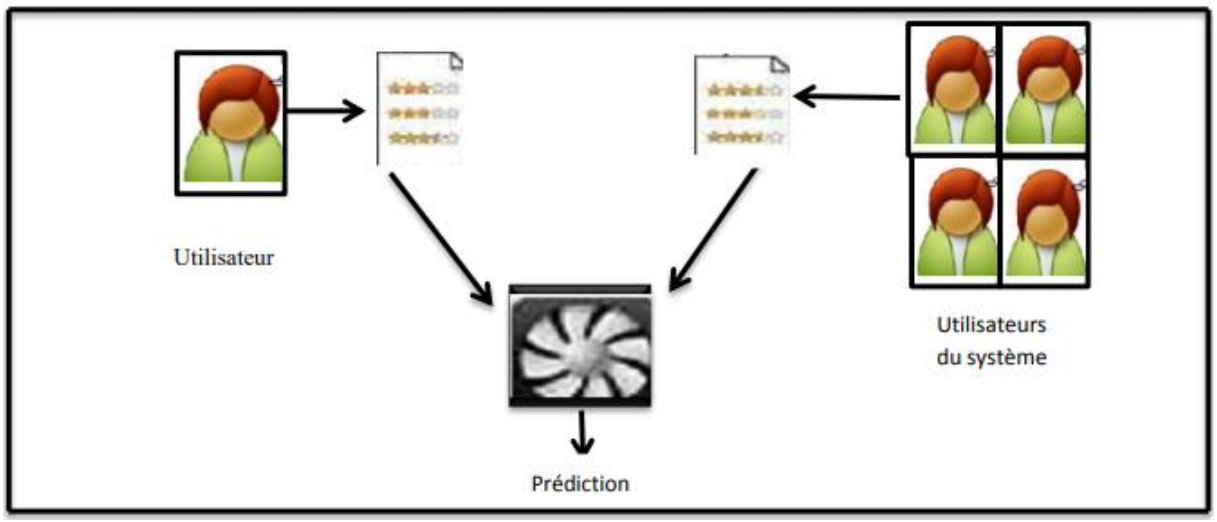


Figure 1 : collaboratif filtering

Collaborative filtering uses statistical methods to make predictions based on the assessment of users' interests. These forecasts are used to make proposals based on the correlation between one's personal profile and the profiles of other users (who have similar interests).

The user can then ask the system to:

- Suggest a resource that might please him.
- Prevent him from resources he should not appreciate.
- Give an estimate of an assessment of a certain resource.

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

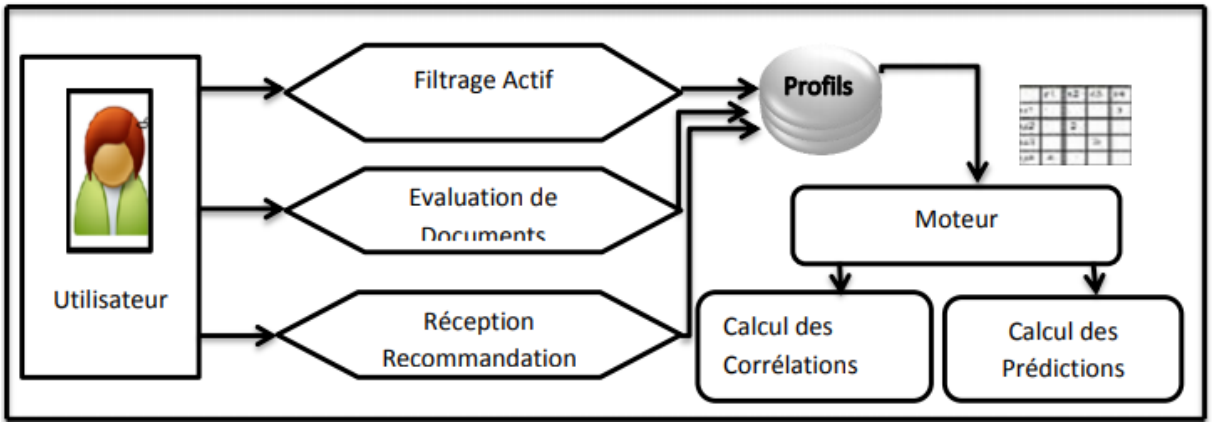


Figure 2 : Processus of collaborative filtrage

So we use two approaches in collaborative filtering:

- User-based recommendation
- Based on the product (item-based recommendation)

3 Pretreating:

We start by downloading the two dataset: :

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

After the first part of pre-processing with data wrangling. So we're going to delete movies that are rated by less than 6 base users 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

After the second part of the pre-processing that gets added to remove users who give the same rating to the movies they have evaluated in the 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

4 Data exploration/data visualization :

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

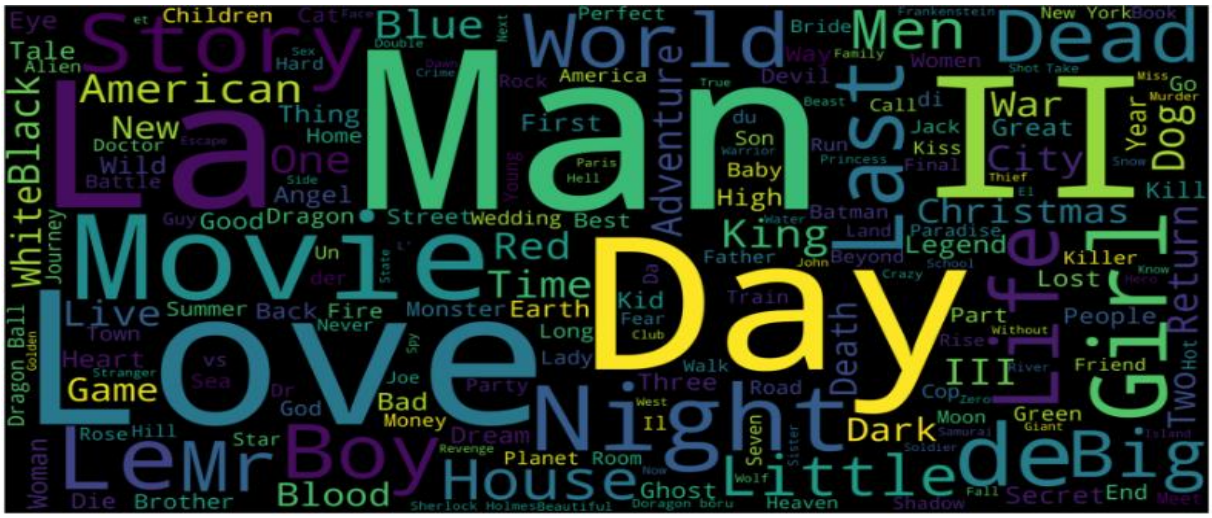
	movieId	title	genres
0	1	Toy Story (1995)	Adventure|Animation|Children|Comedy|Fantasy
1	2	Jumanji (1995)	Adventure|Children|Fantasy
2	3	Grumpier Old Men (1995)	Comedy|Romance
3	4	Waiting to Exhale (1995)	Comedy|Drama|Romance
4	5	Father of the Bride Part II (1995)	Comedy
...	...	...	...
9737	193581	Black 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()
```

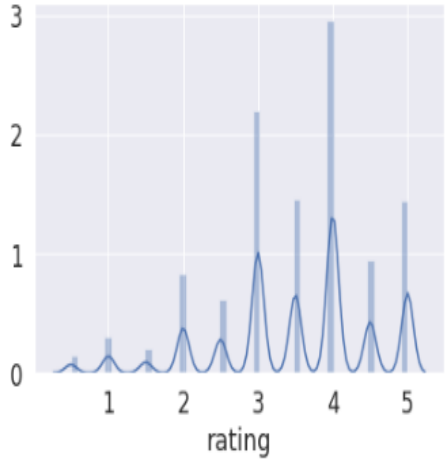


Statistics and distribution.

```
[ ] # 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>
```



©

It seems that users are quite generous in their ratings. The average score is 3.58 on a scale of 5. Half of the films have a score of 4 and 5. W personally thinks that a 5-level scoring skill was not a good indicator that people might have different scoring styles (i.e. person A could always use 4 for an average movie, while Person B gives only 4 for their favorites). Each user has evaluated at least 20 films, so we doubt that the distribution could be caused just by the random variance in the quality of the films. Let's also take a look at a subset of 20 movies with the highest rating.

– – – – – – – – – – – – – – – –

Join us all 3 files in a data frame and join all 3 files in a View 20 movies with the highest ratings

```
# 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

The gender variable will surely be important when constructing the recommendation engines since it describes the content of the film (i.e. Animation, Horror, Sci-Fi). A basic assumption is that films in the same genre should have similar content. We will try to see exactly which genres are the most popular.

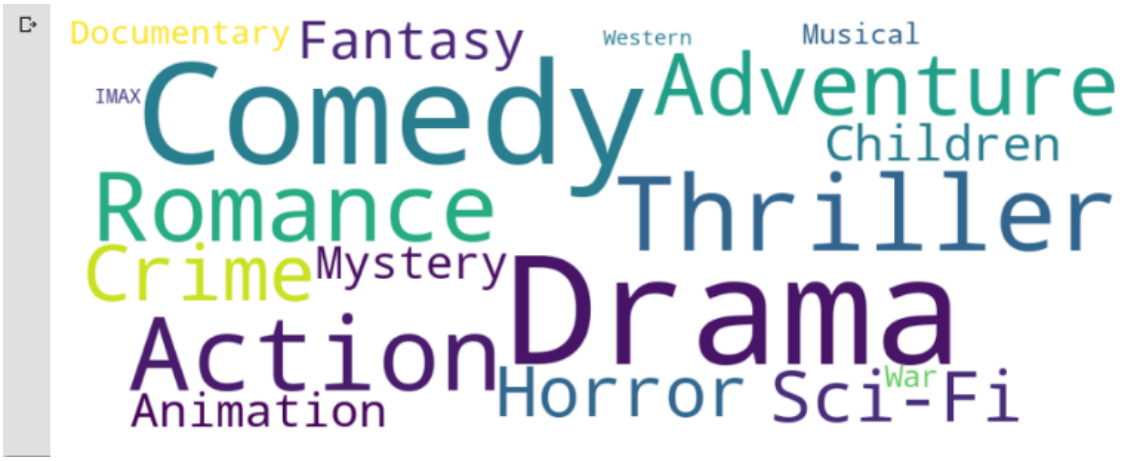
```
[['Drama', 4361],
 ['Comedy', 3756],
 ['Thriller', 1894],
 ['Action', 1828],
 ['Romance', 1596]]
```

The top 5 genres are, in this respect order: Drama, Comedy, Action, Thriller, and Romance. I'll show this on a wordcloud too in order to make it more visually appealing.

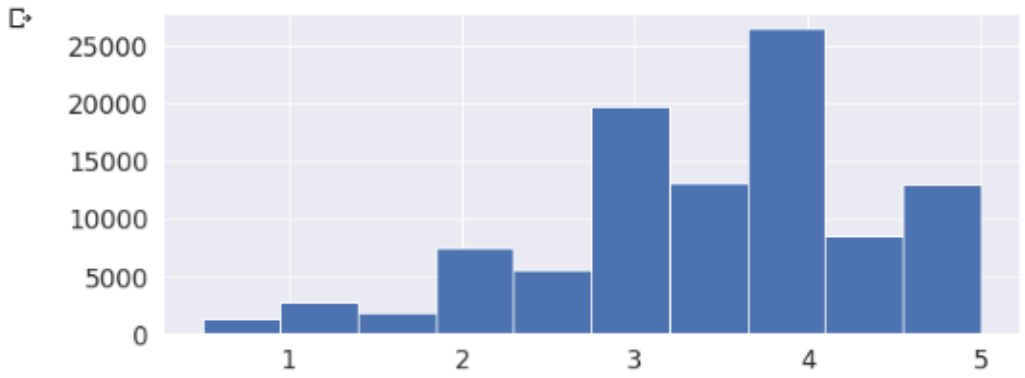
```
# Define the dictionary used to produce the genre wordcloud
genres = dict()
trunc_occurences = keyword_occurences[0:18]
for s in trunc_occurences:
    genres[s[0]] = s[1]

# 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);
```



We can see from the histogram plot that the most frequent movie rating given is 4. The result is also an unbalanced target variable, so we will have to address this issue in machine learning.

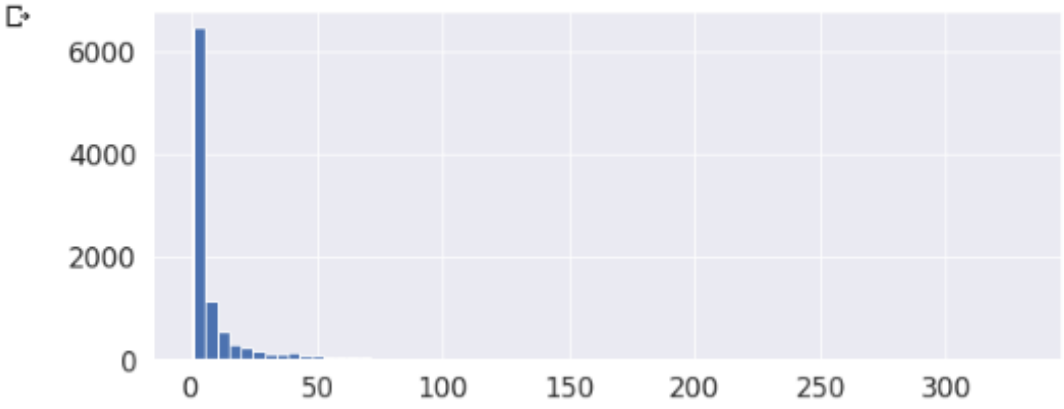
— — — — — — — — — — — — — —

```
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
```

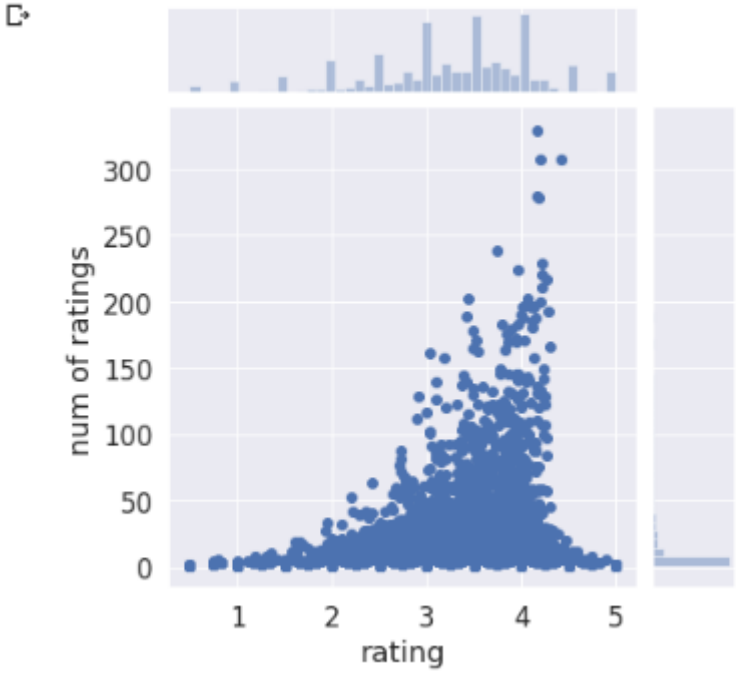
title	rating	num of ratings
'71 (2014)	4.000000	1
'Hellboy': The Seeds of Creation (2004)	4.000000	1
'Round Midnight (1986)	3.500000	2
'Salem's Lot (2004)	5.000000	1
'Til There Was You (1997)	4.000000	2
...	...	...
eXistenZ (1999)	3.863636	22
xXx (2002)	2.770833	24
xXx: State of the Union (2005)	2.000000	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);
```



5 user-based recommendation :

This method identifies users who are similar to the user surveyed and estimates the desired rating as the weighted average of the ratings of these similar users.

Well, UB-CF uses this logic and recommends elements by finding users similar to the active user (to whom we try to recommend a movie). A specific application of this is the nearby user-based algorithm.

In this approach, a matrix A is constructed: [User x Product]

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

In this matrix, user 1 and User 2 are correlated (3 out of 5 products have the same score). This is how one user is recommended the preferred products of the second. But how do you calculate the correlation or similarity between two users?

Similarity methods:

- **Cosine similarity :**

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

The correlation of two users is important for such important u values

- **Jaccard similarity :**

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

This method is to ignore the scores :

Fonction score :

It's easy to find a feature for non-custom collaborative filtering (i.e. we don't consider the likes, dislikes and notation of the active user of the past) that returns a user-taking score you and element i as input settings.

The feature produces a score that quantifies the strength of a user you like/prefers the i element.

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

Surprise is a scikit Python for analyzing recommendation systems that process explicit evaluation data.

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

6 item-based recommendation

In this approach, the same A-matrix is also built. But the difference is by correlation between the products. We're looking for products that are potentially correlated. And we offer a product that is related to other products (having the most degree of correlation).

Element-based collaborative filtering is a model-based algorithm for making recommendations. In the algorithm, the similarities between the different elements of the data set are calculated using one of the many similarity measures, and then these similarity values are used to predict the assessments of user object pairs that are not present in the dataset.

The similarity values between the elements are measured by observing all users who evaluated both elements. The similarity between the two elements depends on the evaluations assigned to the elements by the users who evaluated both.

.

7 KNN :

To implement an element-based collaborative filter, KNN is a perfect reference model and also a very good baseline for the development of the recommendation system. But what is the KNN?

KNN is a non-parametric and lazy learning method. It uses a database in which data points are separated into several clusters to make inference for new samples.

KNN makes no assumptions about the distribution of underlying data, but it relies on the similarity of the element. When KNN makes an inference on a movie, KNN will calculate the "distance" between the target film and all the other films in its database, then it ranks its distances and returns the best NK neighboring films closer as the most similar movie recommendations.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

Cry, the Beloved Country (1995) 5

Indian in the Cupboard, The (1995) 5

Screamers (1995) 5

In the Bleak Midwinter (1995) 5

8 Conclusion :

In recent years, recommendation systems have played a particularly important role in online marketing. Thanks to them, e-commerce companies have been able to differentiate themselves from their competitors, make life easier for existing customers and reach their potential customers.

Depending on the company's strategies, several recommendation techniques are integrated to adapt needs. As we have seen in our work, these methods have different advantages and disadvantages, so no solution can meet all the problems. In reality, companies use multiple approaches and combine them to have a better recommendation, assessed by certain pre-defined criteria in their context as well as their objectives.

While the operation of a recommendation system is fairly simple, its implementation is complicated. These difficulties lie in aspects such as the collection and selection of relevant data, the size and quality of data, the scarcity of data, the construction of user profiles, the prediction for new user profiles or new products.