

# MOVIE RECOMMENDATION SYSTEM



REALISER PAR :

LAAMARI YOUSSEF

ENCADRÉ PAR:  
PR HDIOUD  
FERDOUSS

# SOMMAIRE

- 01**      Introduction
  - 1.1 Les systèmes de recommandation (SR)
  - 1.2 Les objectifs des systèmes de recommandation :
- 02**      Idée générale
- 03**      MOVIELENS Dataset
- 04**      Data Exploration
  - 4.1 Movie Table
  - 4.2 Rating Table
- 05**      Data Visualisation
- 06**      Definition de KNN algorithme
  - 6.1 Definiton de KNN
  - 6.2 Similitude cosinus
  - 6.3 Similitude Eucliiediene
- 07**      les Filtres de Recommandation
  - 7.1 Content Based Filtering
  - 7.2 Collaborative Filtering
    - 7.2.1 Item Based
    - 7.2.2 User Based
    - 7.2.3 Model Based (SVD)
  - 7.3 LES AVANTAGES ET LES INCONVÉNIENTS
    - 7.3.1 Collaborative Filtering using KNN
    - 7.3.2 Collaborative Filtering using SVD
- 08**      Exemple d'Execution
- 09**      Conclusion

# **1\_INTRODUCTION**

## **1.1 Les systèmes de recommandation (SR) :**

sont des systèmes de traitement d'information qui collectent activement divers types de données pour créer des suggestions. Ces données comprennent généralement des informations sur les articles à recommander et les utilisateurs qui recevront ces recommandations. Néanmoins, les données et les sources d'information disponibles pour les systèmes de recommandation peuvent être très variées, en fonction de la technique de recommandation utilisée.

## **1.2 Les objectifs des systèmes de recommandation :**

Les systèmes de recommandation (SR) visent à fournir aux utilisateurs des suggestions pertinentes et personnalisées en fonction de leurs intérêts et de leurs besoins. Ils s'efforcent d'atteindre plusieurs objectifs clés :

- 1. Réduire la surcharge d'information :** Dans un monde où l'information abonde, les SR aident les utilisateurs à naviguer dans la multitude d'options et à trouver rapidement ce qui leur correspond.
- 2. Améliorer la découverte de produits et services :** Les SR permettent aux utilisateurs de découvrir de nouveaux produits, services ou contenus qu'ils n'auraient peut-être pas trouvés autrement.
- 3. Personnaliser l'expérience utilisateur :** En tenant compte des préférences et des interactions passées des utilisateurs, les SR offrent une expérience plus personnalisée et engageante.
- 4. Augmenter la satisfaction des clients :** Des recommandations pertinentes peuvent conduire à une plus grande satisfaction des clients, une fidélité accrue et une augmentation des ventes.
- 5. Stimuler l'engagement des utilisateurs :** En proposant des contenus pertinents et intéressants, les SR peuvent inciter les utilisateurs à passer plus de temps sur une plateforme ou à utiliser un service plus fréquemment.
- 6. Optimiser les décisions des utilisateurs :** Les SR peuvent aider les utilisateurs à prendre des décisions plus éclairées en leur fournissant des informations et des suggestions contextuelles.

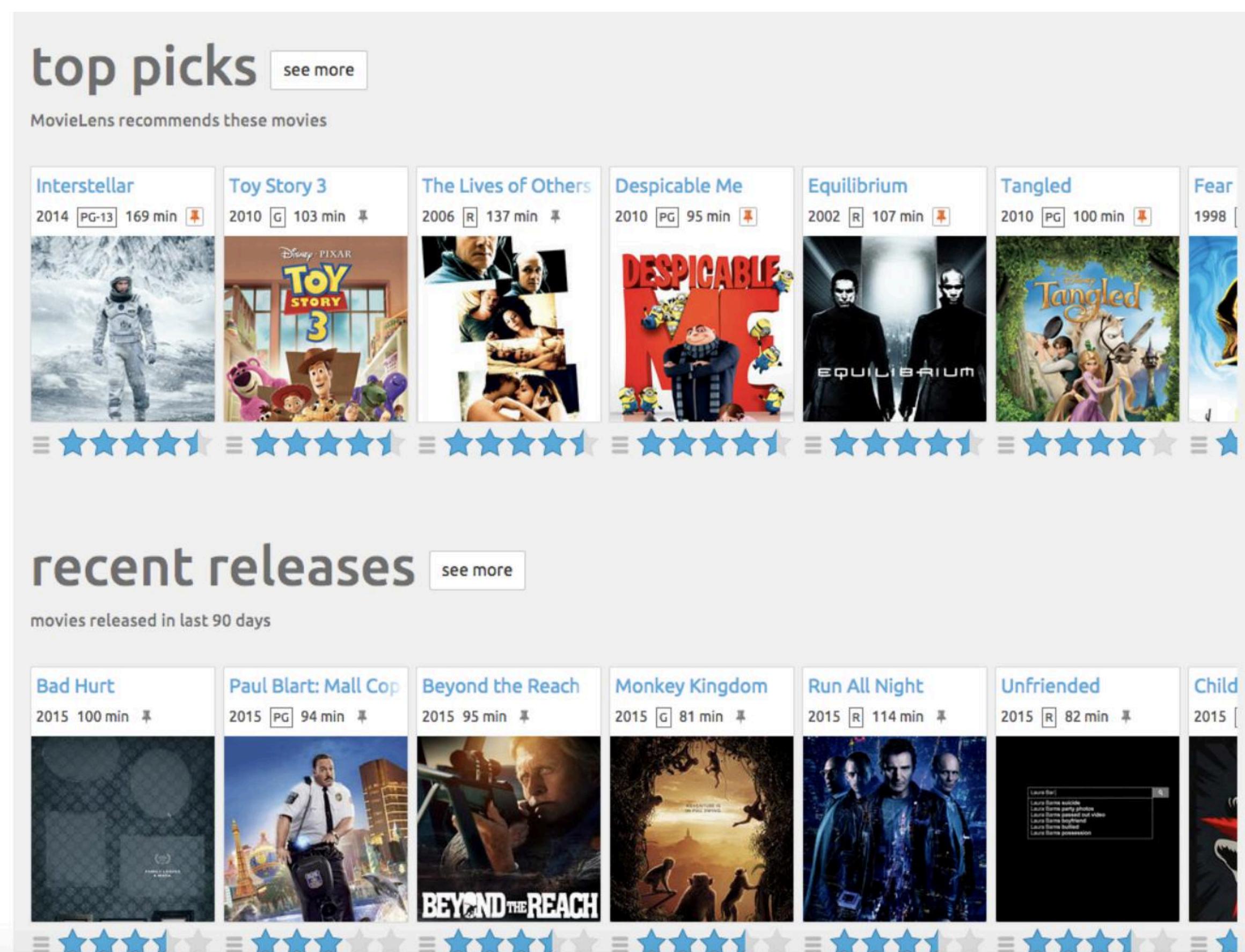
## 2\_IDÉE GÉNÉRALE

Récemment, le Web connaît un afflux massif d'informations, qui ne cesse de croître. Cet afflux fournit aux utilisateurs des renseignements sur les produits, les hôtels, les restaurants et divers services grâce à une multitude de sources. Malgré les avantages de cette information, la quantité massive de données rend difficile pour les utilisateurs son traitement et la prise de décision parmi les options disponibles. Cela conduit à une surcharge d'information et complique le processus décisionnel. Dans ce contexte, il est important de filtrer les informations à une quantité limitée en fonction des préférences actuelles de l'utilisateur/client afin de l'aider à prendre des décisions éclairées.



Ce filtrage est généralement effectué par des systèmes de recommandation (SR) et est développé pour répondre au problème de la surcharge d'information, en fournissant des recommandations personnalisées de services en fonction des préférences spécifiques des clients. Étape de recommandation Cette étape est une extension de l'étape de prédiction. Elle vise à filtrer les éléments les plus pertinents pour un utilisateur en utilisant diverses approches pour l'aider dans sa prise de décision. Elle propose à l'utilisateur de nouveaux articles (généralement les N premiers articles ayant les notes prédictives les plus élevées) qui sont susceptibles de l'intéresser le plus.

# 3\_MOVIELENS DATASET



Le dataset MovieLens 20M, disponible sur Kaggle, est une collection de données très utilisée dans le domaine de la recherche sur les systèmes de recommandation. Voici une explication détaillée de ce que contient ce dataset et de ses utilisations typiques.

Voici les principaux fichiers contenus dans le dataset :

- **ratings.csv**
  - **userId:** Identifiant de l'utilisateur qui a effectué la notation.
  - **movieId:** Identifiant du film noté.
  - **rating:** La note attribuée au film, sur une échelle de 0.5 à 5 étoiles.
  - **timestamp:** Le moment où la note a été donnée.
- **movies.csv**
  - **movieId:** Identifiant du film.
  - **title:** Titre du film.
  - **genres:** Genres du film, séparés par des barres verticales (par exemple, Comedy|Romance).

# 4. DATA EXPLORATION

## 4.1 Movie Table :

	movieId	title
0	1	Toy Story (1995)
1	2	Jumanji (1995)
2	3	Grumpier Old Men (1995)
3	4	Waiting to Exhale (1995)
4	5	Father of the Bride Part II (1995)

```
----  
0   movieId    27278 non-null  int64  
1   title      27278 non-null  object  
2   genres     27278 non-null  object  
dtypes: int64(1), object(2)  
memory usage: 639.5+ KB  
Info:
```

```
None  
Describe:
```

```
          movieId  
count    27278.000000  
mean     59855.480570  
std      44429.314697  
min      1.000000  
25%    6931.250000  
50%    68068.000000  
75%    100293.250000  
max    131262.000000  
Null values:
```

```
  movieId    0  
  title     0  
  genres    0  
  dtype: int64  
Duplicated values:
```

**Valeurs manquantes dans  
movie.csv : Aucune**

**Valeurs dupliquées dans  
movie.csv : Aucune**

```
0
```

# 4. DATA EXPLORATION

## 4.2 Rating Table :

	userId	movieId	rating	title
0	1	2	3.5	Jumanji (1995)
1	5	2	3.0	Jumanji (1995)
2	13	2	3.0	Jumanji (1995)
3	29	2	3.0	Jumanji (1995)
4	34	2	3.0	Jumanji (1995)

des Information sur la table Rating :

```
✓ 0s ⏎ # rating: Range from (0.5 - 5.0) with 0.5 increments
    print('Mean rating of a movie:', ratings['rating'].mean())

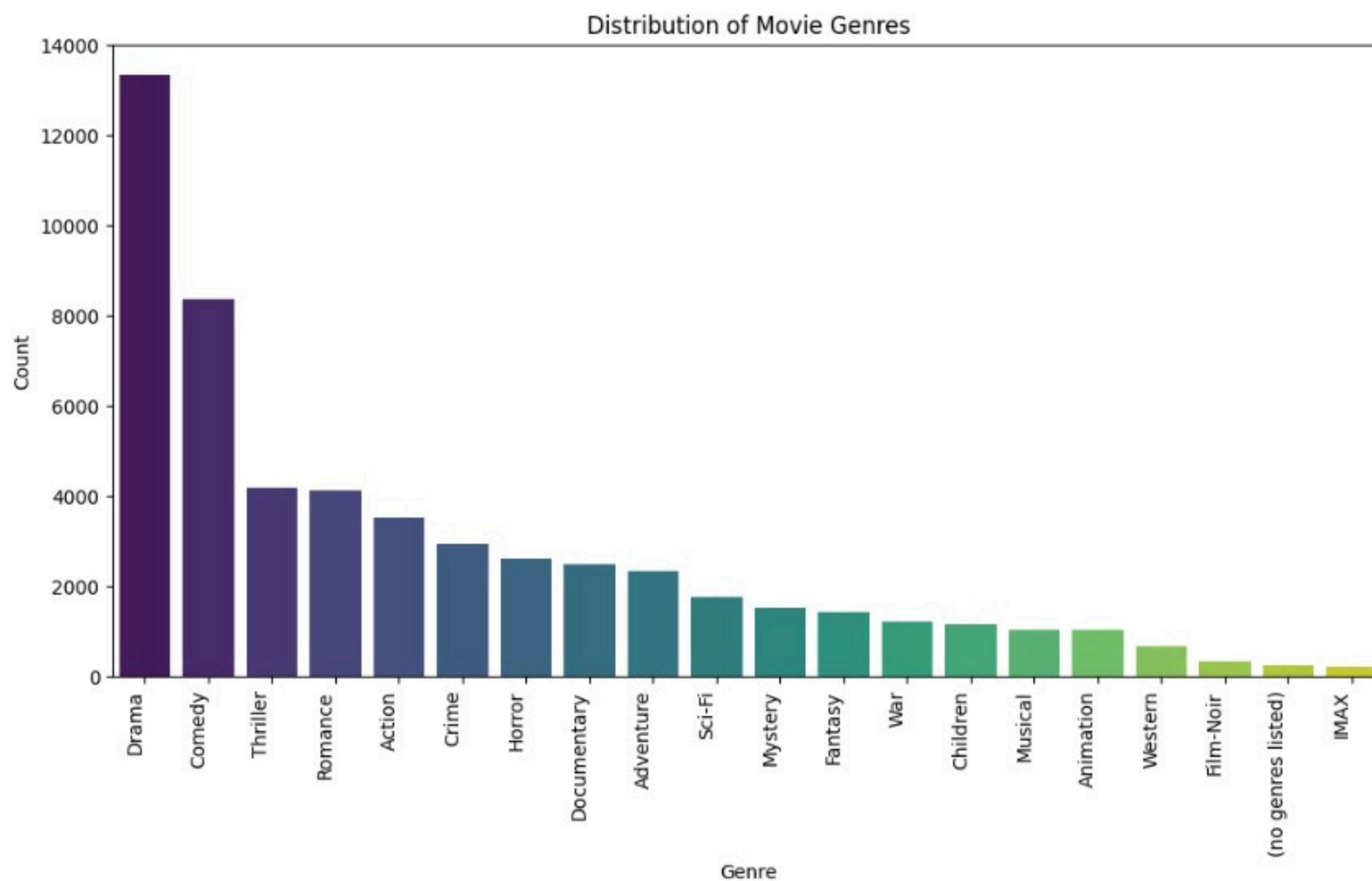
→ Mean rating of a movie: 3.5255285642993797

✓ 0s [8] # userId: Unique Id provided for each user.
    print('Number of users:', ratings['userId'].nunique())
    print('Mean number of ratings for each user:', ratings.groupby(by='userId').size().mean())

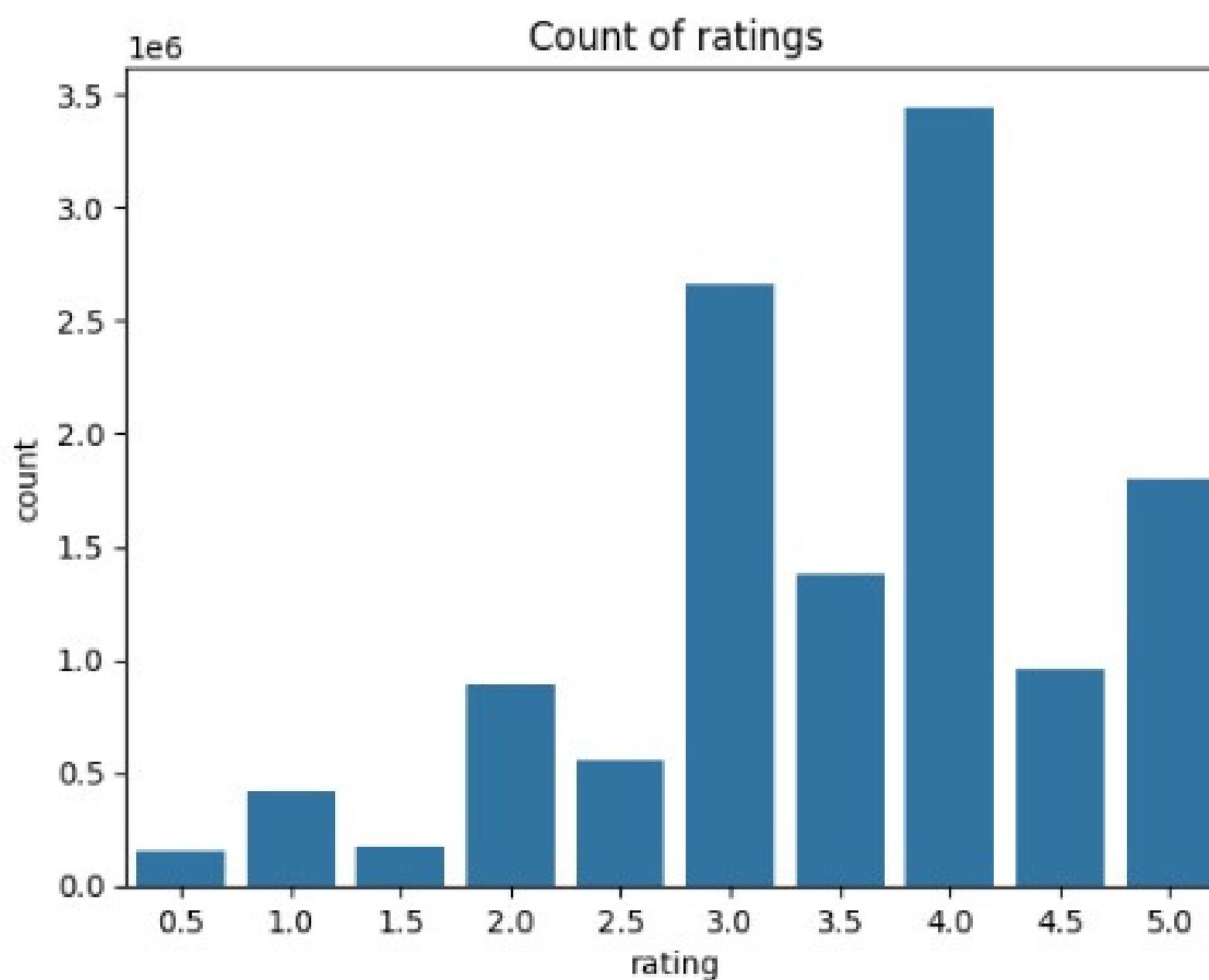
→ Number of users: 138493
    Mean number of ratings for each user: 144.4135299257002
```

```
Describe:              userId      movieId      rating
count  2.000026e+07  2.000026e+07  2.000026e+07
mean   6.904587e+04  9.041567e+03  3.525529e+00
std    4.003863e+04  1.978948e+04  1.051989e+00
min   1.000000e+00  1.000000e+00  5.000000e-01
25%   3.439500e+04  9.020000e+02  3.000000e+00
50%   6.914100e+04  2.167000e+03  3.500000e+00
75%   1.036370e+05  4.770000e+03  4.000000e+00
max   1.384930e+05  1.312620e+05  5.000000e+00
Null values: userId      0
movieId      0
rating       0
timestamp    0
dtype: int64
Duplicated values: 0
First few rows:    userId      movieId      rating      timestamp
0           1          2          3.5  2005-04-02 23:53:47
1           1         29          3.5  2005-04-02 23:31:16
2           1         32          3.5  2005-04-02 23:33:39
3           1         47          3.5  2005-04-02 23:32:07
4           1         50          3.5  2005-04-02 23:29:40
```

# 5 \_ DATA VISUALISATION



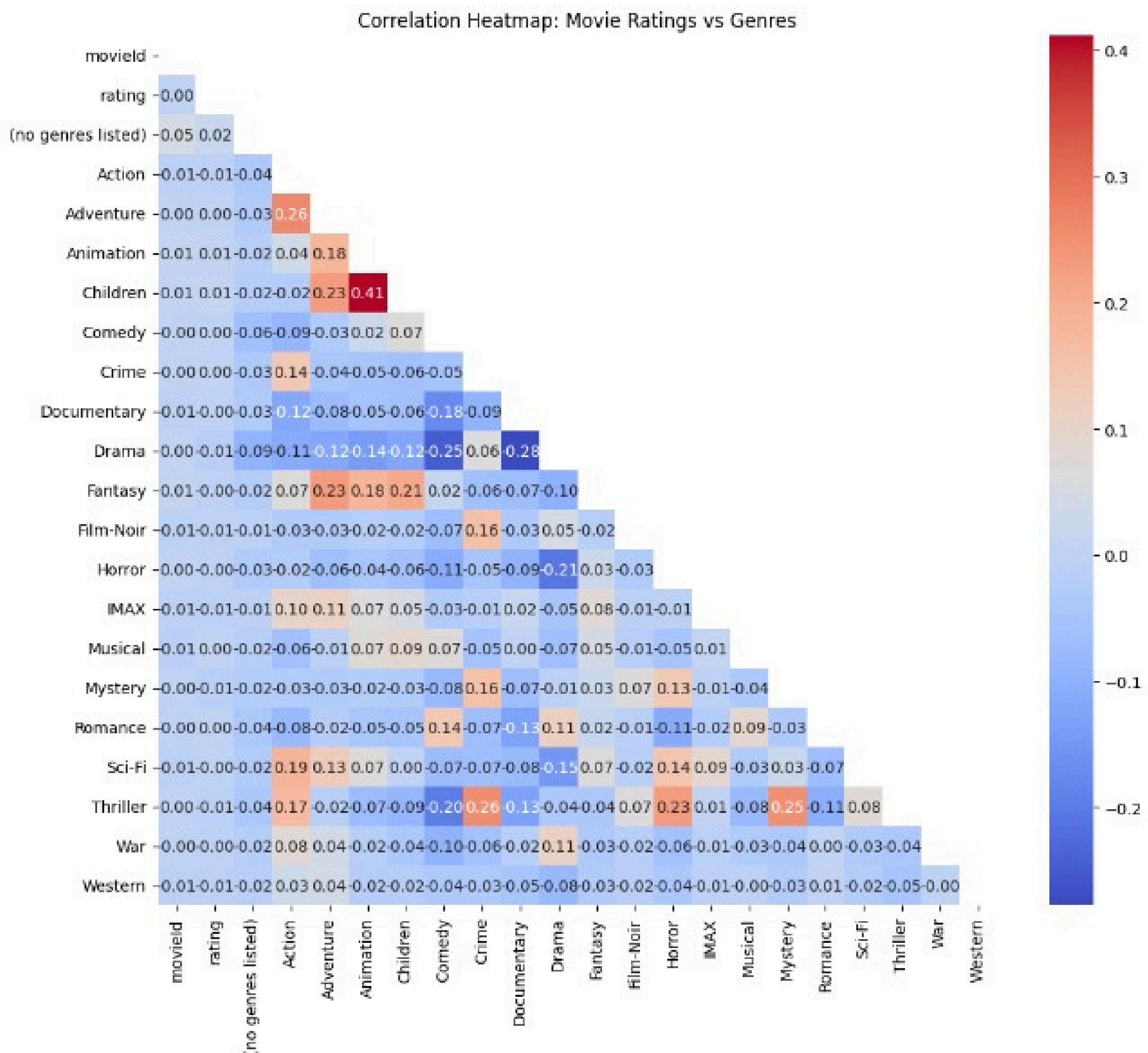
Le genre dramatique est le plus fréquent dans notre ensemble de données .



- La note la plus fréquente est de 4,0, suivie de 3,0 et 5,0.
- Les notes inférieures à 2,0 et supérieures à 4,0 sont relativement peu fréquentes.

# 5 \_ DATA VISUALISATION

La correlation entre les genres :



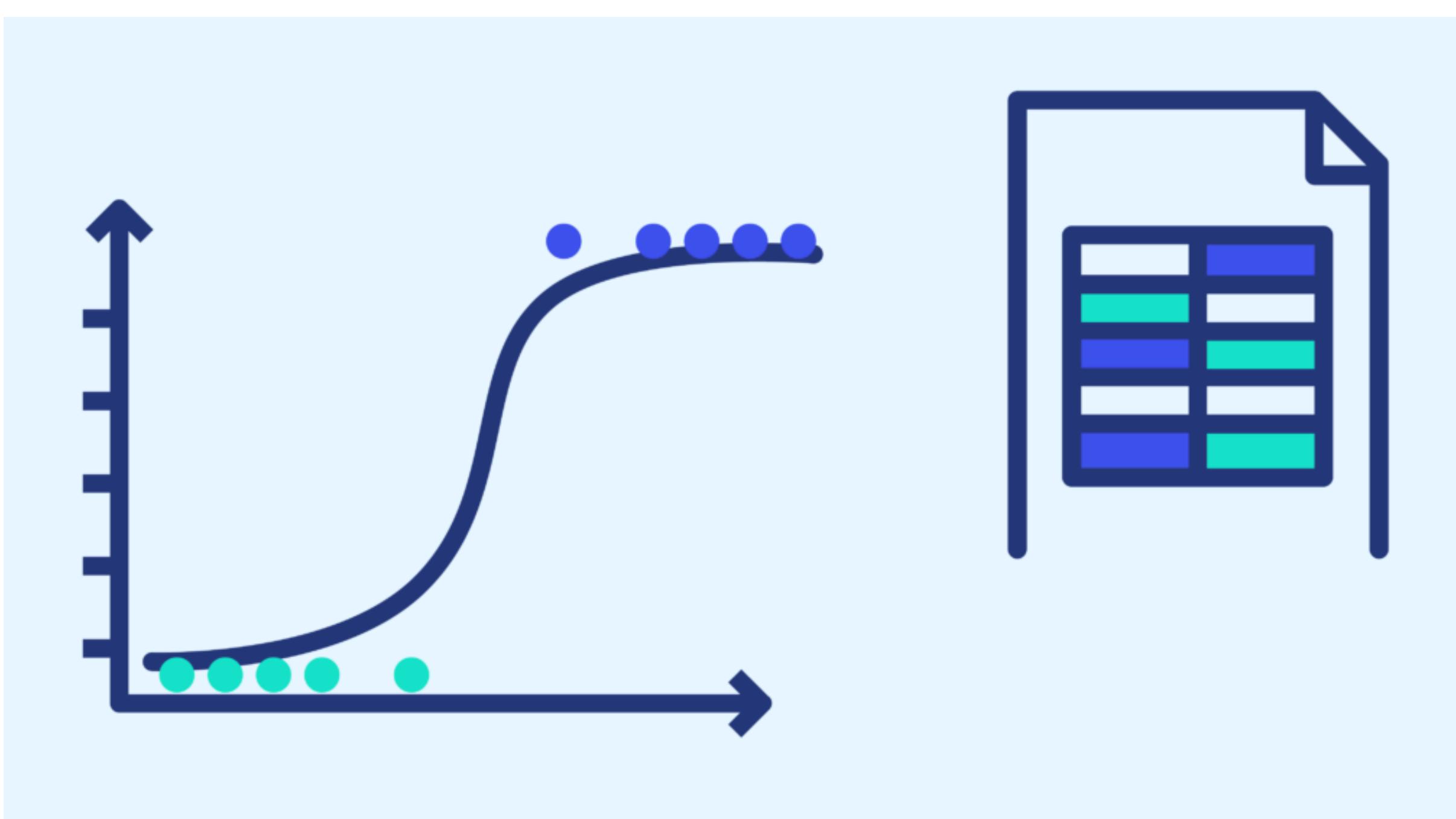
le genre Children et le genre Animation sont fortement Correeler .

# 6\_KNN

## 6.1 Definition de KNN :

L'algorithme des k plus proches voisins, également connu sous le nom de KNN ou k-NN, est un discriminant d'apprentissage supervisé non paramétrique, qui utilise la proximité pour effectuer des classifications ou des prédictions sur le regroupement d'un point de données individuel. Bien qu'il puisse être utilisé pour des problèmes de régression ou de classification, il est généralement utilisé comme algorithme de classification, en partant de l'hypothèse que des points similaires peuvent être trouvés les uns à côté des autres.

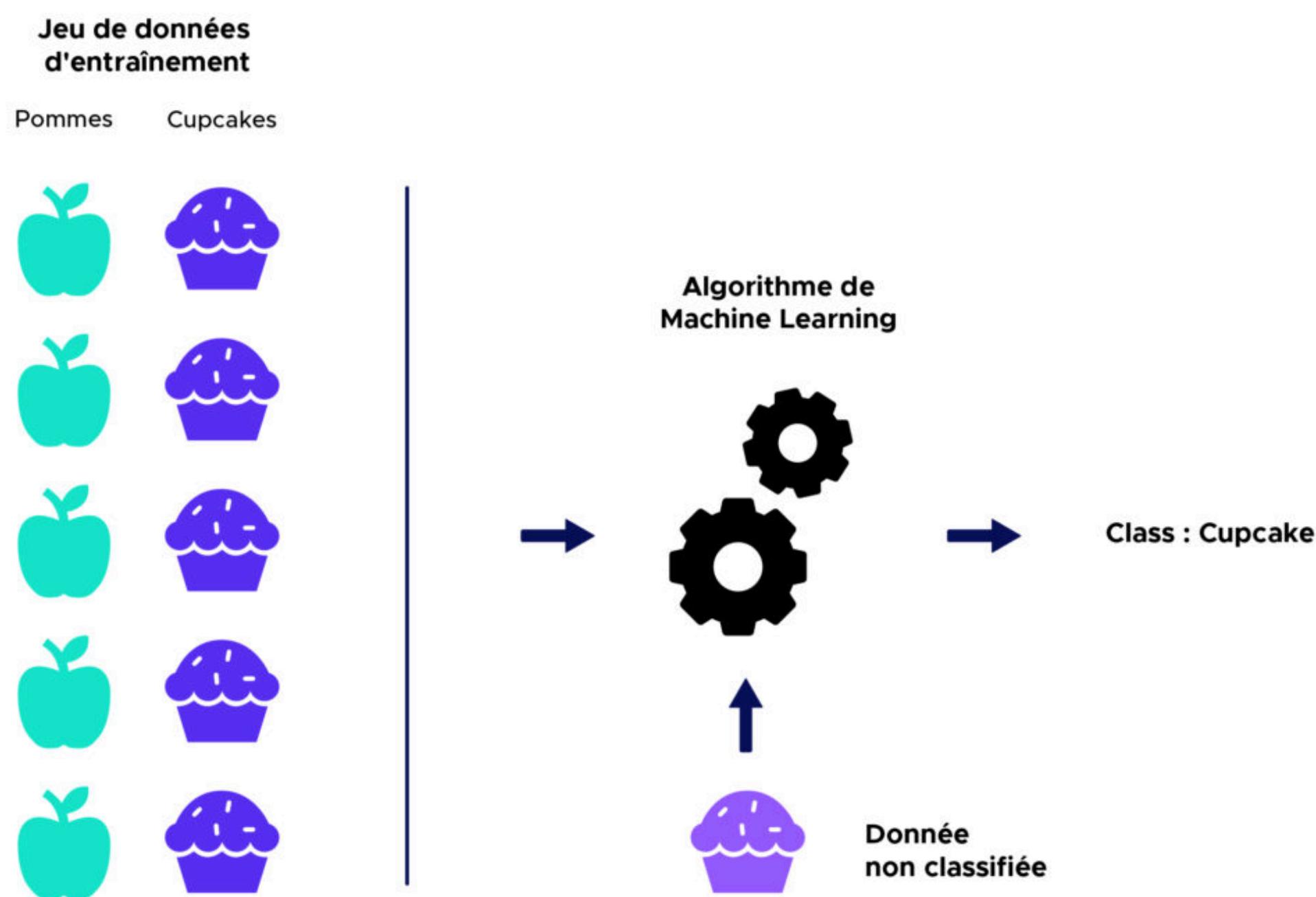
KNN visualisation :



Avant de nous concentrer sur l'algorithme KNN, il est nécessaire de reprendre les bases. Qu'est-ce qu'un algorithme d'apprentissage supervisé?

En apprentissage supervisé, un algorithme reçoit un ensemble de données qui est étiqueté avec des valeurs de sorties correspondantes sur lequel il va pouvoir s'entraîner et définir un modèle de prédiction. Cet algorithme pourra par la suite être utilisé sur de nouvelles données afin de prédire leurs valeurs de sorties correspondantes.

Voici une illustration simplifiée :



L'intuition derrière l'algorithme des K plus proches voisins est l'une des plus simples de tous les algorithmes de Machine Learning supervisé :

- Étape 1 : Sélectionnez le nombre K de voisins
- Étape 2 : Calculez la distance

$$\sum_{i=1}^n |x_i - y_i|$$

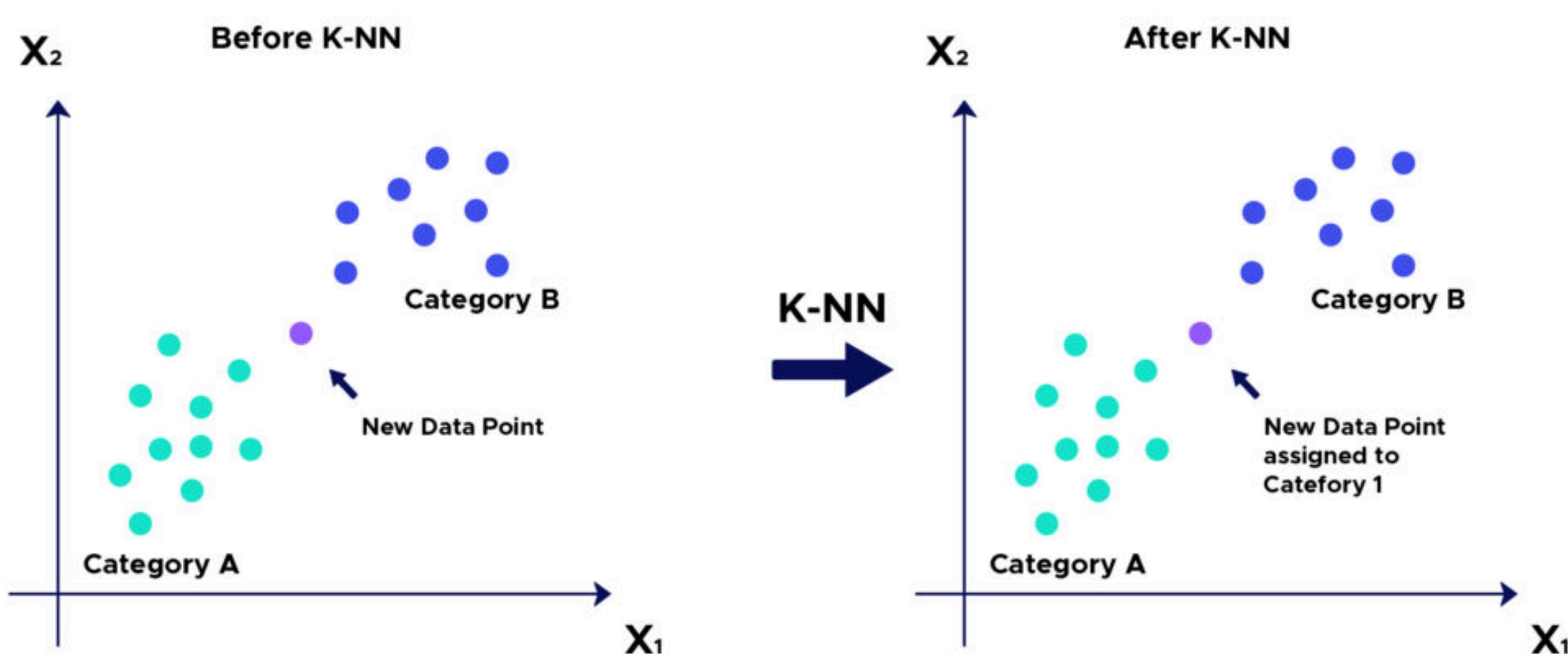
**Manhattan**

$$\sqrt{\sum_{i=1}^n (x_i - y_i)^2}$$

**Euclidienne**

Du point non classifié aux autres points.

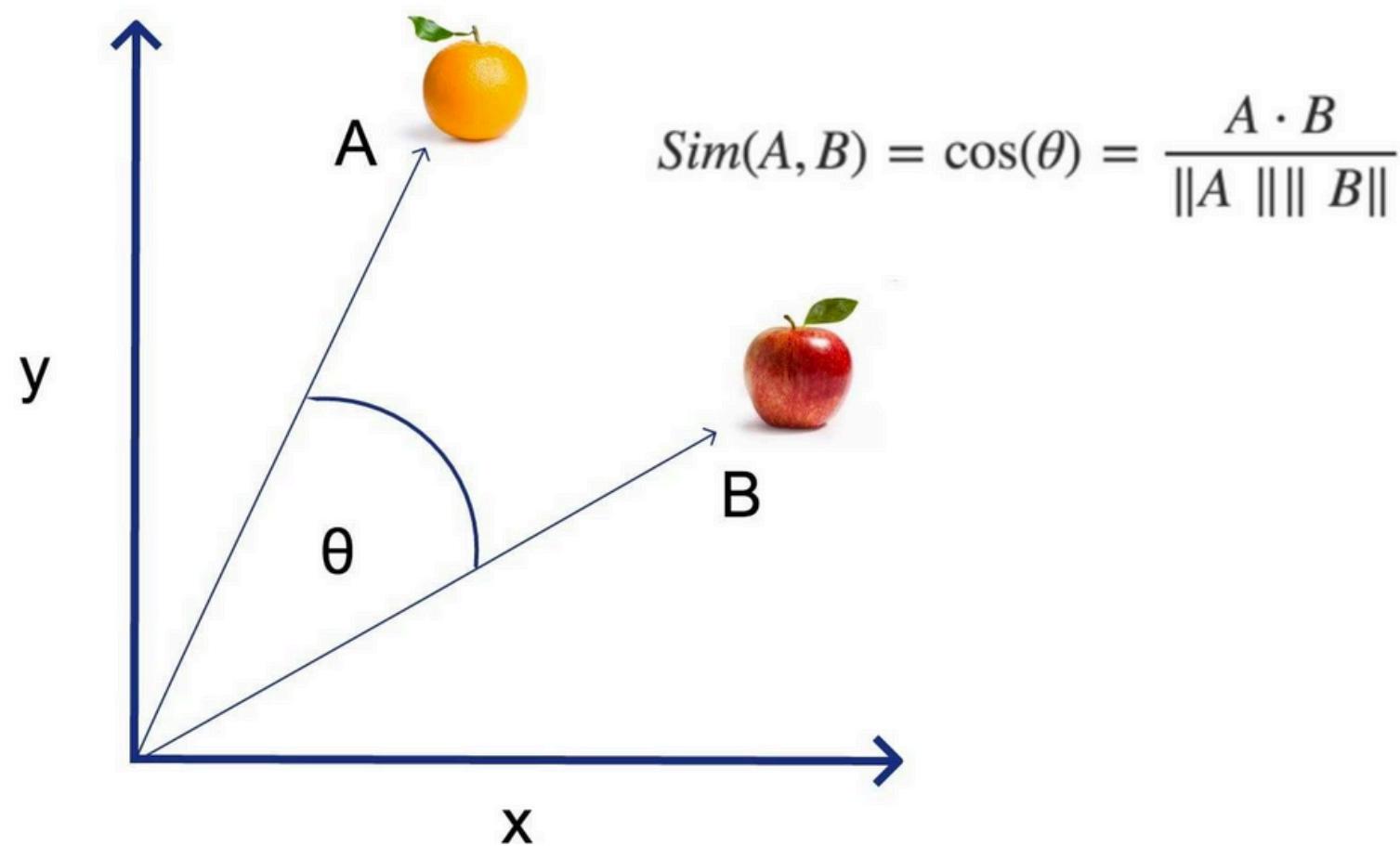
- Étape 3 : Prenez les K voisins les plus proches selon la distance calculée.
- Étape 4 : Parmi ces K voisins, comptez le nombre de points appartenant à chaque catégorie.
- Étape 5 : Attribuez le nouveau point à la catégorie la plus présente parmi ces K voisins.
- Étape 6 : Notre modèle est prêt :



## 6.2 Similitude cosinus :

La similitude cosinus est basée sur l'angle entre deux vecteurs qui représentent les documents. Plus l'angle est proche de zéro, plus les documents sont similaires. La similarité cosinus est facile à calculer, en particulier avec des matrices clairsemées, et elle peut capturer la similitude globale des documents quelle que soit leur longueur. Cependant, il présente également certains inconvénients. Il ne tient pas compte de la fréquence des mots, de sorte qu'il peut ne pas refléter l'importance des termes rares ou communs. Il ne tient pas compte non plus de l'ordre ou du contexte des mots, de sorte qu'il peut manquer certaines nuances ou différences sémantiques.

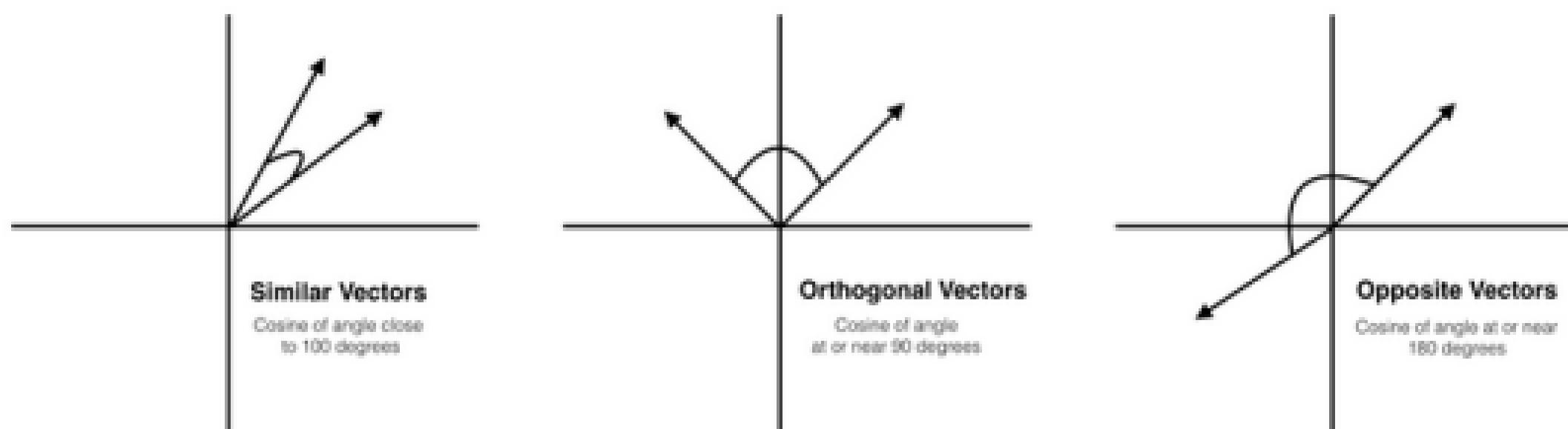
## Cosine Similarity



Dans le cas de deux vecteurs A et B, la similarité cosinus est donnée par :

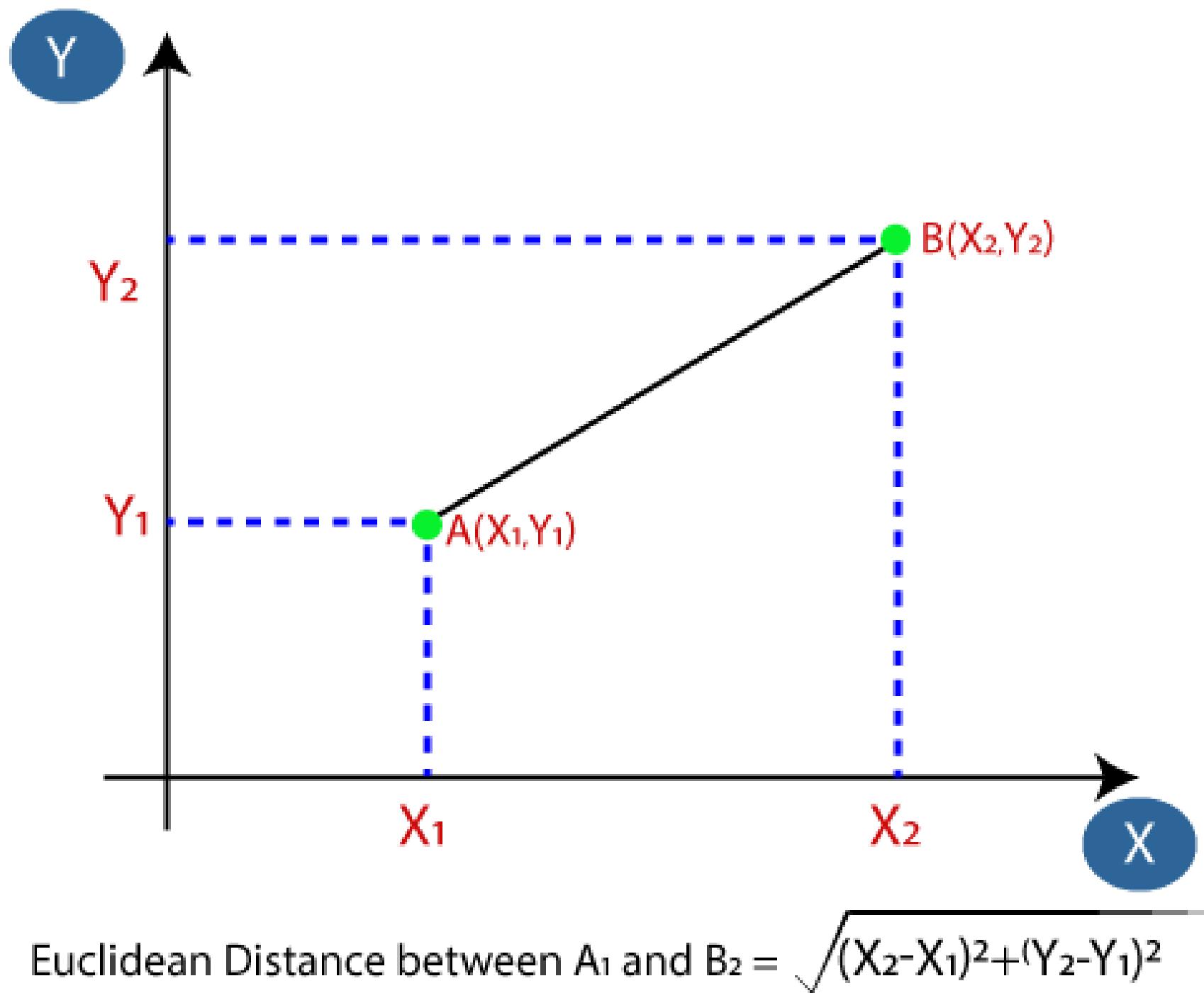
$$\cos(\theta) = \frac{\mathbf{A} \cdot \mathbf{B}}{\|\mathbf{A}\| \|\mathbf{B}\|} = \frac{\sum_{i=1}^n A_i B_i}{\sqrt{\sum_{i=1}^n A_i^2} \sqrt{\sum_{i=1}^n B_i^2}}$$

Les valeurs vont de -1 (complètement dissemblables) à 1 (complètement similaires)

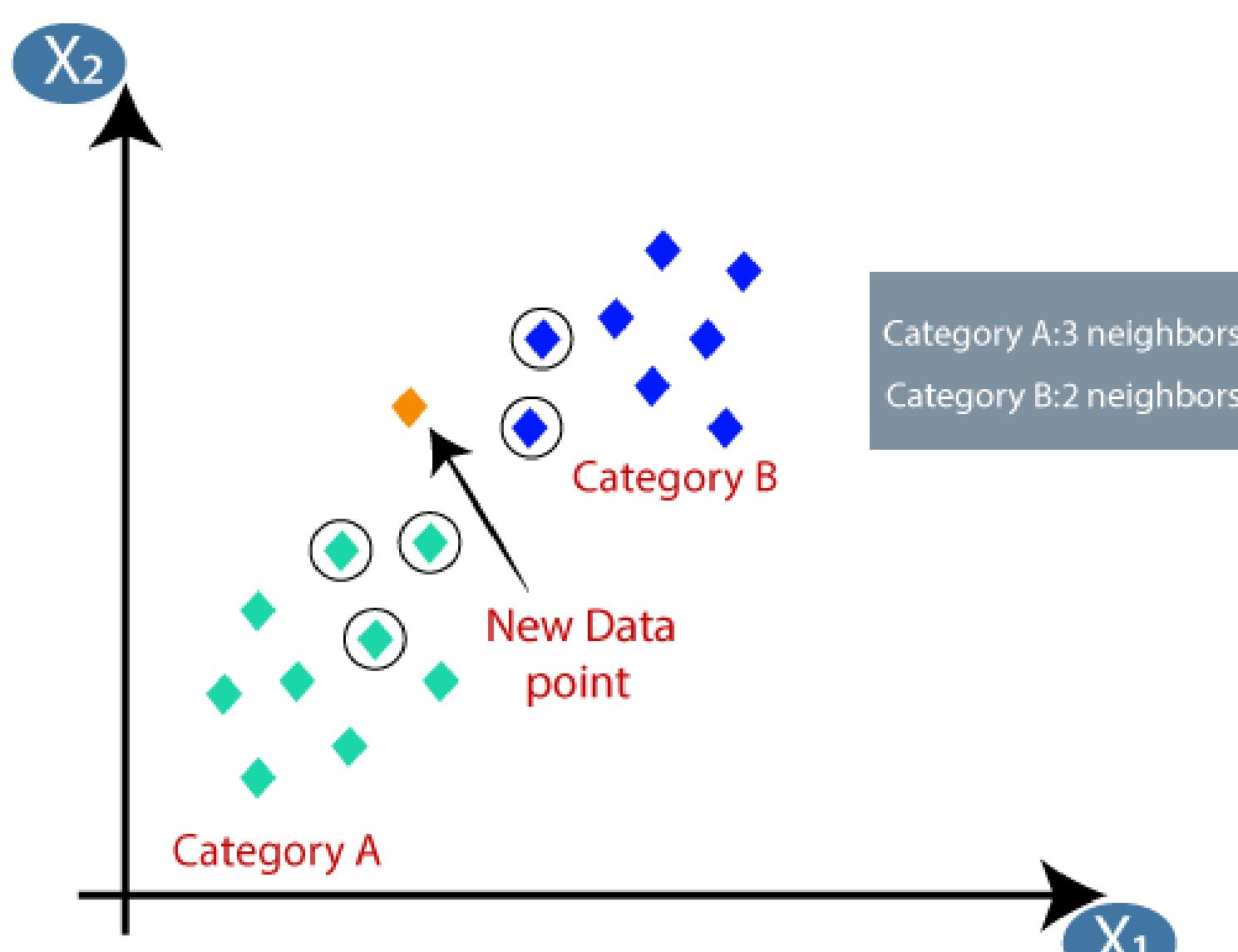


## 6.3 Similitude euclidienne :

La distance euclidienne est la distance entre deux points, que nous avons déjà étudiée en géométrie. Il peut être calculé comme suit:

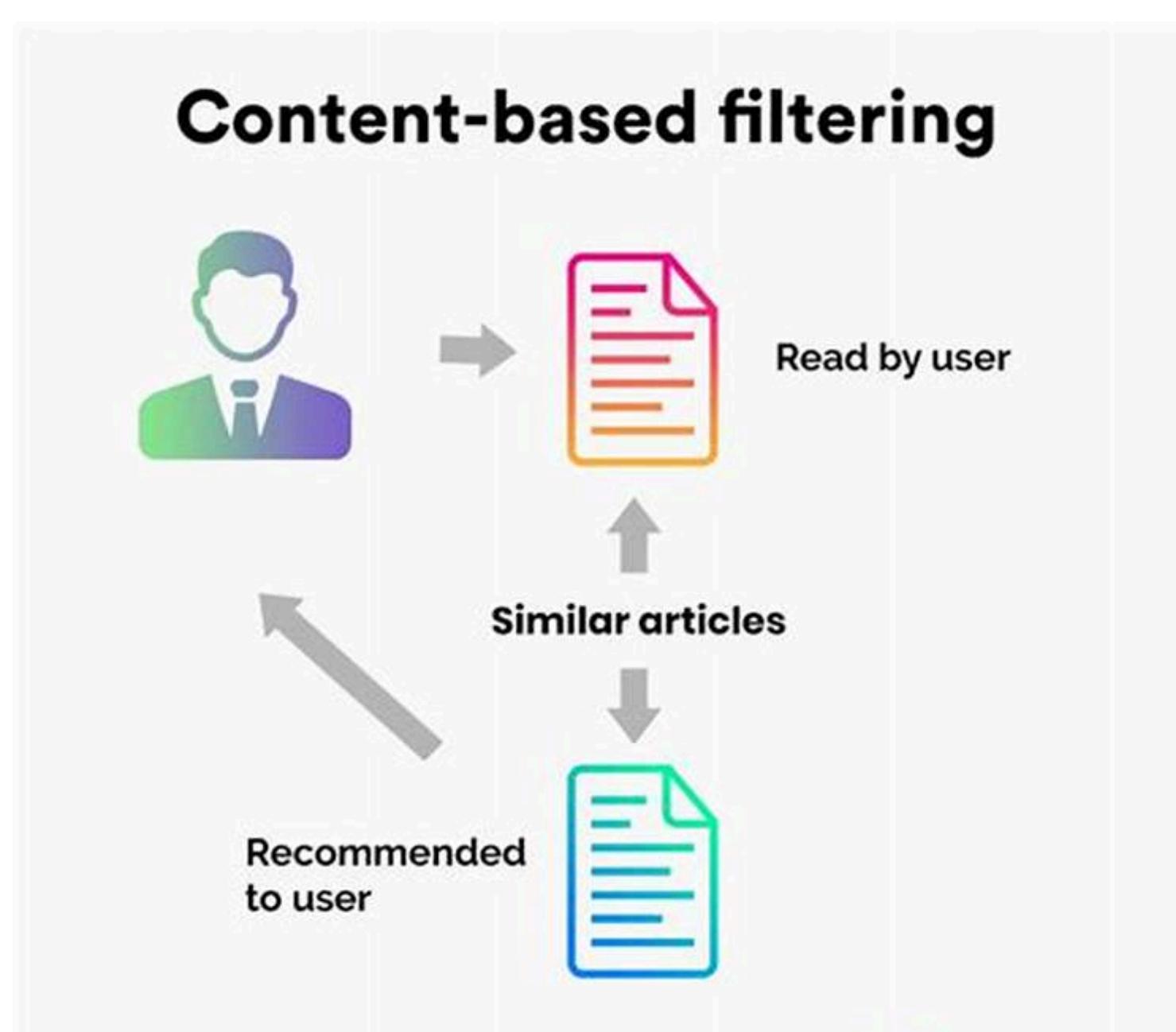


En calculant la distance euclidienne, nous avons obtenu les voisins les plus proches, comme trois voisins les plus proches dans la catégorie A et deux voisins les plus proches dans la catégorie B. Considérez l'image ci-dessous :



# 7. LES FILTRES DE RECOMMANDATION

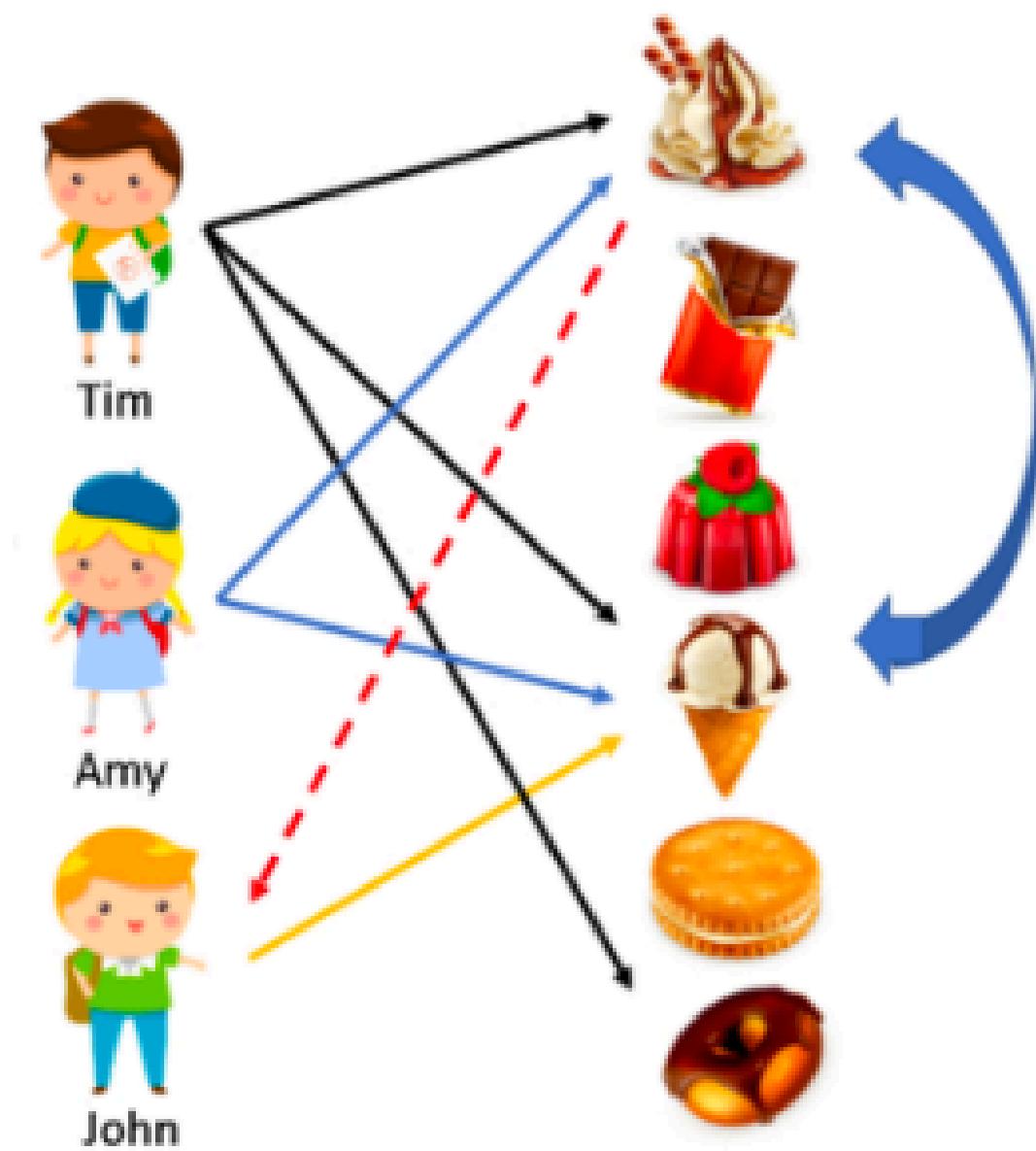
## 7.1 Content Based Filtering



Le filtrage basé sur le contenu est une méthode utilisée par les systèmes de recommandation pour suggérer des éléments personnalisés aux utilisateurs, en se basant sur leurs préférences et leur historique. Il analyse les caractéristiques des éléments et crée un profil utilisateur en fonction de leurs interactions passées. L'idée est que si un utilisateur apprécie un élément avec des caractéristiques spécifiques, il est probable qu'il aime d'autres éléments similaires. Par exemple, si un utilisateur aime les films d'action, le système recommandera d'autres films d'action en se basant sur des attributs comme le genre ou les acteurs. Bien que le filtrage basé sur le contenu offre des recommandations personnalisées et fonctionne même en l'absence de données sur d'autres utilisateurs, il a des limites, comme l'incapacité à recommander des éléments en dehors des préférences passées de l'utilisateur et le besoin d'attributs précis pour les éléments.

## 7.2 Collaborative Filtering

### 7.2.1 Item Based

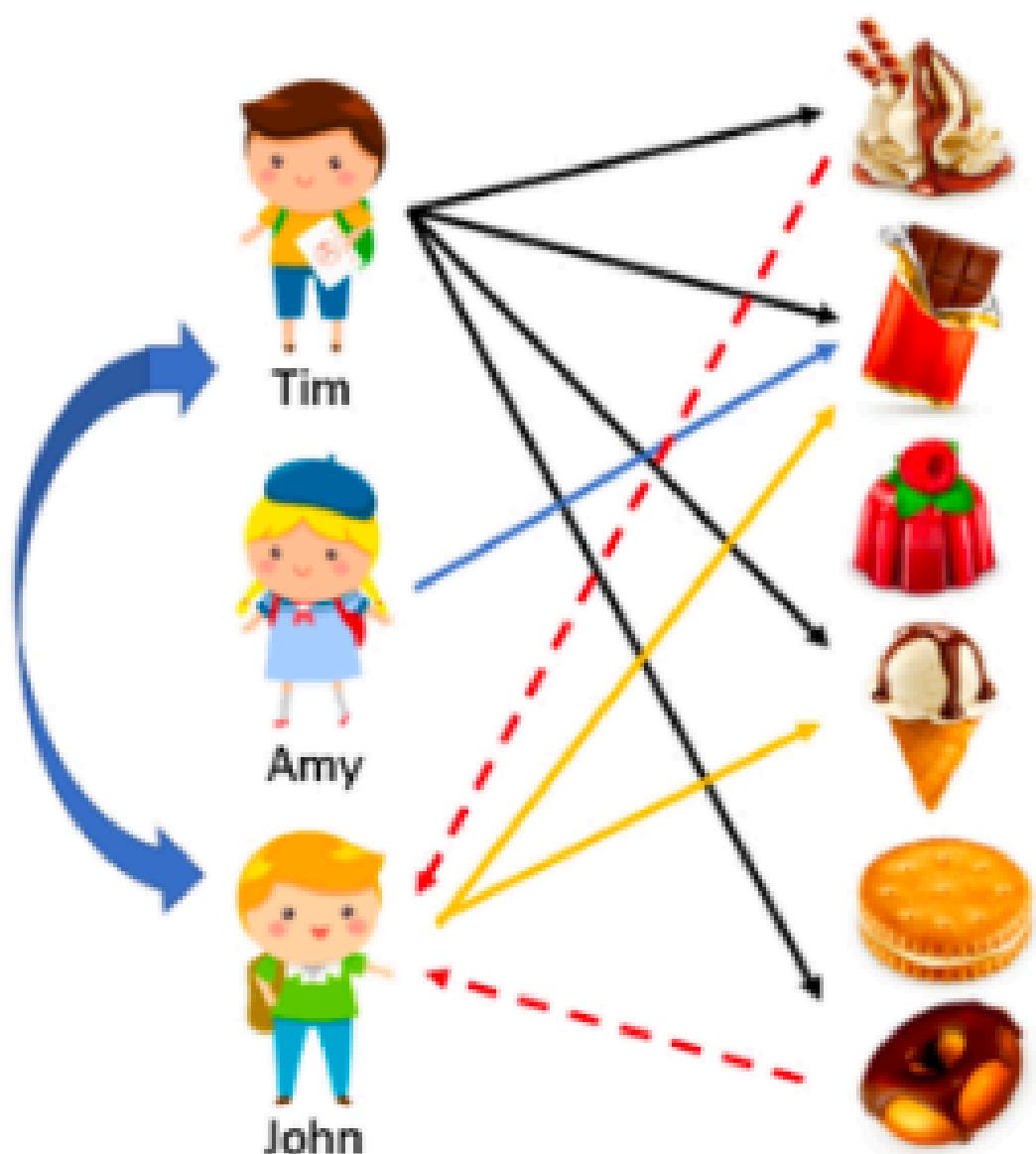


### Item Based

Le filtrage collaboratif basé sur les items est une technique utilisée dans les systèmes de recommandation pour fournir des suggestions personnalisées aux utilisateurs en fonction de leurs préférences et de celles d'utilisateurs similaires. C'est une forme de filtrage collaboratif qui se concentre sur la similarité entre les articles plutôt que sur les utilisateurs.

Dans le filtrage collaboratif basé sur les items, les recommandations sont générées en identifiant des articles similaires à ceux pour lesquels un utilisateur a déjà manifesté de l'intérêt. L'hypothèse sous-jacente est que si un utilisateur aime ou interagit avec un article particulier, il est probable qu'il ait des préférences similaires pour d'autres articles du même genre.

## 7.2.2 User Based



## User Based

Le filtrage collaboratif basé sur les utilisateurs est une autre technique utilisée dans les systèmes de recommandation pour fournir des suggestions personnalisées.

Contrairement à la méthode basée sur les items, celle-ci se concentre sur la similarité entre les utilisateurs plutôt que sur les articles.

Dans le filtrage collaboratif basé sur les utilisateurs, les recommandations sont générées en identifiant des utilisateurs ayant des goûts similaires à l'utilisateur cible. Le système suggère ensuite des articles que ces utilisateurs similaires ont appréciés ou avec lesquels ils ont interagi. L'hypothèse sous-jacente est que si deux utilisateurs partagent des intérêts et des préférences similaires, ils auront probablement des affinités pour d'autres articles du même type.

### 7.2.3 Model Based (SVD)

Items

		Items					
		The Godfather	Inception	Leon	The Departed	Pulp Fiction	Forrest Gump
Users	User 1	10	-1	8	10	9	4
	User 2	8	9	10	-1	-1	8
	User 3	10	5	4	9	-1	-1
	User 4	9	10	-1	-1	-1	3
	User 5	6	-1	-1	-1	8	10

User-item  
Interaction  
matrix

Dans les systèmes de recommandation, la Décomposition en Valeurs Singulières (SVD) est une technique fondamentale. Elle permet de comprendre les interactions complexes entre les utilisateurs et les éléments, tels que les films dans notre projet. En utilisant la SVD, nous pouvons décomposer la matrice des évaluations des utilisateurs pour les films en plusieurs composantes, nous permettant ainsi de générer des recommandations personnalisées.

[Le fonctionnement du modèle SVD repose sur la décomposition de la matrice des évaluations des utilisateurs pour les films en trois matrices : U,  $\Sigma$  et V. La matrice U représente les utilisateurs et leurs préférences, la matrice  $\Sigma$  contient les valeurs singulières qui indiquent la force des caractéristiques latentes, et la matrice V représente les films. En utilisant ces matrices, le modèle SVD est capable de prédire les évaluations des utilisateurs pour les films non évalués, fournissant ainsi des recommandations personnalisées basées sur les préférences des utilisateurs.

## 7.3.1 les Avantages et les Inconvénients Collaborative Filtering Item-Based using KNN :

### **Collaborative Filtering Item-Based using KNN :**

#### **Avantages :**

##### **Robustesse aux nouveaux éléments :**

Capable de recommander des éléments similaires même pour ceux qui n'ont pas encore été évalués par beaucoup d'utilisateurs.

##### **Facilité d'implémentation :**

L'algorithme de KNN est relativement simple à mettre en œuvre et peut être efficace dans des cas d'utilisation avec une quantité modérée de données.

**Capture des relations entre les éléments :** Peut identifier des relations complexes entre les éléments en fonction des préférences des utilisateurs.

#### **Inconvénients :**

##### **Problème de démarrage à froid :**

Les nouveaux éléments peuvent avoir du mal à être recommandés car ils n'ont pas encore de données d'interaction utilisateur.

**Échelle :** Le calcul des similarités entre les éléments peut devenir coûteux lorsque le nombre d'éléments augmente considérablement

##### **Sparsité des données :**

Si les utilisateurs ont noté très peu d'éléments, cela peut entraîner des recommandations moins précises.

## 7.3.2 les Avantages et les Inconvénients SVD

### (Décomposition en valeurs singulières) :

**SVD (Décomposition en valeurs singulières) :**

**Avantages :**

1. Traitement des données manquantes : Peut gérer efficacement les données manquantes ou la sparsité des données.
2. Recommandations de qualité : Peut fournir des recommandations de qualité même dans des cas où les données sont rares ou bruitées.
3. Réduction de la dimensionnalité : Peut aider à gérer efficacement les ensembles de données volumineux en réduisant leur dimensionnalité.

**Inconvénients :**

1. Calcul intensif : Le calcul de la décomposition en valeurs singulières peut être coûteux, surtout avec de grandes matrices de notation utilisateur-article.
2. Sensibilité aux données bruitées : Les recommandations peuvent être influencées par les données bruitées ou les évaluations aberrantes.
3. Difficulté d'interprétation : Les recommandations sont générées à partir de facteurs latents, ce qui peut rendre difficile leur interprétation et leur explication aux utilisateurs.

# Collaborative\_Filtering\_Item\_Based :

## 1. CRÉATION DE LA MATRICE UTILISATEUR-FILM

POUR GÉRER EFFICACEMENT LA MÉMOIRE, LA MATRICE EST CONVERTIE EN UNE MATRICE CREUSE À L'AIDE DE LA BIBLIOTHÈQUE SCIPY.

```
from scipy.sparse import csr_matrix  
  
movie_features_df_matrix = csr_matrix(matrix_pivot.values)
```

## 2. ENTRAÎNEMENT DU MODÈLE KNN :

UN MODÈLE K-NN EST INSTANCIÉ AVEC UNE MESURE DE SIMILARITÉ COSINUS ET L'ALGORITHME "BRUTE". IL EST ENSUITE AJUSTÉ AUX DONNÉES.

```
from sklearn.neighbors import NearestNeighbors  
model_knn = NearestNeighbors(metric = 'cosine', algorithm = 'brute')  
model_knn.fit(movie_features_df_matrix)
```

## 3. SAISIE DU FILM DE REQUÊTE :

L'UTILISATEUR ENTRE LE TITRE DU FILM POUR LEQUEL IL SOUHAITE OBTENIR DES RECOMMANDATIONS.

```
query_movie_title = input("Enter the title of the movie you want to use as the query: ")  
Enter the title of the movie you want to use as the query: (500) Days of Summer (2009)
```

## 4. RECHERCHE DES VOISINS LES PLUS PROCHES :

LE MODÈLE K-NN EST UTILISÉ POUR TROUVER LES K FILMS LES PLUS SIMILAIRES AU FILM DE REQUÊTE EN TERMES DE NOTATION PAR LES UTILISATEURS.

```
query_index = matrix_pivot.index.get_loc(query_movie_title)  
  
# Use the trained kNN model to find the nearest neighbors to the query movie  
distances, indices = model_knn.kneighbors(matrix_pivot.iloc[query_index, :].values.reshape(1, -1), n_neighbors=10)
```

## 5. AFFICHAGE DES RECOMMANDATIONS:

LES K FILMS RECOMMANDÉS SONT AFFICHÉS AVEC LEUR DISTANCE RESPECTIVE PAR RAPPORT AU FILM DE REQUÊTE:

```
# Print the recommended movies along with their distances from the query movie
print("Recommendations for {0}:\n".format(matrix_pivot.index[query_index]))
for i in range(0, len(distances.flatten())):
    if i == 0:
        print("Original movie: {0}".format(matrix_pivot.index[query_index]))
    else:
        print("{0}: {1}, with distance of {2}".format(i, matrix_pivot.index[indices.flatten()[i]], distances.flatten()[i]))
```

Recommendations for (500) Days of Summer (2009):

Original movie: (500) Days of Summer (2009)  
1: Up in the Air (2009), with distance of 0.5880272388458252  
2: Juno (2007), with distance of 0.5974820256233215  
3: Hangover, The (2009), with distance of 0.6031189560890198  
4: Slumdog Millionaire (2008), with distance of 0.6145920157432556  
5: Social Network, The (2010), with distance of 0.6158373355865479  
6: Inception (2010), with distance of 0.6165720224380493  
7: Up (2009), with distance of 0.6187136173248291  
8: Inglourious Basterds (2009), with distance of 0.6196826696395874  
9: Black Swan (2010), with distance of 0.6217919588088989

# Model\_Based\_Filtering basé sur SVD :

## APPLIQUER SVD MODEL SUR LA DATASET :

```
✓ 2s ⏎ #Modelling phase
trainset, testset = train_test_split(data, test_size=.25)
svd_model = SVD()
svd_model.fit(trainset)
predictions = svd_model.test(testset)

✓ 0s [15] accuracy.rmse(predictions)
→ RMSE: 0.9401
0.9400562871025276

✓ 0s [16] svd_model.predict(uid=1.0, iid=541, verbose=True)
→ user: 1.0      item: 541      r_ui = None    est = 4.24    {'was_impossible': False}
Prediction(uid=1.0, iid=541, r_ui=None, est=4.2426307976372994, details={'was_impossible': False})

✓ 0s ⏎ sample_df[sample_df["userId"] == 1]
→
movieId          title           genres  userId  rating   timestamp
3612352  Blade Runner (1982) Action|Sci-Fi|Thriller      1     4.0 2005-04-02 23:30:03

our model predicted 4.22
but originally it is 4.00 which is good accuracy
```

Accuracy est 0.94

Notre SVD model prédit 4.22 tant que le rating original est 4.0 donc le model est bien

## PREDICTER LE RATING POUR CHAQUE FILM

```
▶ def suggest(df,user_id,sug):
    didnt_watch = df["movieId"][~(df["userId"] == user_id)].drop_duplicates().values.tolist()
    temp_dict={}

    for i in didnt_watch:
        temp_dict[i] = svd_model.predict(uid=user_id, iid=i)[3]

    suggestions = pd.DataFrame(temp_dict.items(),columns=["movieId",'possible_rate']).sort_values(by="possible_rate", ascending=False)
    merged = pd.merge(suggestions,movie[["movieId","title"]], how="inner", on="movieId")

    return merged

[ ] suggest(df,15,50).sort_values(by="title", ascending=False).head(10)
→
movieId  possible_rate          title
20      89674      4.102815  Zombie Island Massacre (1984)
44      89800      4.102815  You're Telling Me! (1934)
29      89872      4.102815  Warrior of the Lost World (1983)
49      89774      4.102815      Warrior (2011)
13          1      4.102815      Toy Story (1995)
3       89753      4.102815  Tinker Tailor Soldier Spy (2011)
4       89747      4.102815      Taxi! (1932)
```

## Content-based filtering :

```
✓ 0s ⏪ # Test recs on "Toy Story (1995)" :  
index = find_idx_by_title("Toy Story (1995)")  
  
recommend_movies(index, pd.DataFrame(movies_features.iloc[index, 2:]).transpose(), model)
```

movieId	title	genres	year	distance
24092	Aladdin (1992)	Adventure Animation Children Comedy Fantasy	1992	0.0
3663	Adventures of Rocky and Bullwinkle, The (2000)	Adventure Animation Children Comedy Fantasy	2000	0.0
10987	Wild, The (2006)	Adventure Animation Children Comedy Fantasy	2006	0.0
24156	Boxtrolls, The (2014)	Adventure Animation Children Comedy Fantasy	2014	0.0
0	Toy Story (1995)	Adventure Animation Children Comedy Fantasy	1995	0.0
3027	Toy Story 2 (1999)	Adventure Animation Children Comedy Fantasy	1999	0.0
3922	Emperor's New Groove, The (2000)	Adventure Animation Children Comedy Fantasy	2000	0.0
24458	Toy Story Toons: Hawaiian Vacation (2011)	Adventure Animation Children Comedy Fantasy	2011	0.0
18274	Asterix and the Vikings (Astérix et les Vikings...)	Adventure Animation Children Comedy Fantasy	2006	0.0
24460	Toy Story Toons: Small Fry (2011)	Adventure Animation Children Comedy Fantasy	2011	0.0

As you can see, all the recommended movies have similar genres as the movie "Toy Story (1995)"

**COMME VOUS POUVEZ LE CONSTATER, TOUS LES FILMS RECOMMANDÉS ONT DES GENRES SIMILAIRES À CEUX DU FILM "TOY STORY (1995)".**

## CONTENT BASED FILTERING, WITH FEATURES = RELEASE YEAR:

```
✓ 0s ⏪ # Test recs on "Toy Story (1995)" :  
index = find_idx_by_title("Toy Story (1995)")  
  
recommend_movies(index, pd.DataFrame(movies_features.iloc[index, 1:2]).transpose(), model_1)
```

movieId	title	genres	year	distance
1	Jumanji (1995)	Adventure Children Fantasy	1995	0.0
7	Tom and Huck (1995)	Adventure Children	1995	0.0
4	Father of the Bride Part II (1995)	Comedy	1995	0.0
8	Sudden Death (1995)	Action	1995	0.0
0	Toy Story (1995)	Adventure Animation Children Comedy Fantasy	1995	0.0
3	Waiting to Exhale (1995)	Comedy Drama Romance	1995	0.0
5	Heat (1995)	Action Crime Thriller	1995	0.0
13	Nixon (1995)	Drama	1995	0.0
2	Grumpier Old Men (1995)	Comedy Romance	1995	0.0
9	GoldenEye (1995)	Action Adventure Thriller	1995	0.0

**COMME VOUS POUVEZ LE CONSTATER, TOUS LES FILMS RECOMMANDÉS LE MÊME ANNEE À CELLE DU FILM "TOY STORY (1995)".**

# CONTENT BASED FILTERING, WITH FEATURES = GENRES + RELEASE YEAR

```
# Test recs on "Toy Story (1995)" :  
index = find_idx_by_title("Toy Story (1995)")  
  
recommend_movies(index, pd.DataFrame(movies_features.iloc[index, 1:]).transpose(), nn)
```

	movieId	title	genres	year	distance
0	1	Toy Story (1995)	Adventure Animation Children Comedy Fantasy	1995	0.00000
2209	2294	Antz (1998)	Adventure Animation Children Comedy Fantasy	1998	0.05183
24092	114240	Aladdin (1992)	Adventure Animation Children Comedy Fantasy	1992	0.05183
3027	3114	Toy Story 2 (1999)	Adventure Animation Children Comedy Fantasy	1999	0.06593
10114	33463	DuckTales: The Movie - Treasure of the Lost La...	Adventure Animation Children Comedy Fantasy	1990	0.07869
3663	3754	Adventures of Rocky and Bullwinkle, The (2000)	Adventure Animation Children Comedy Fantasy	2000	0.07869
3922	4016	Emperor's New Groove, The (2000)	Adventure Animation Children Comedy Fantasy	2000	0.07869
9890	32352	Thief and the Cobbler, The (a.k.a. Arabian Kni...	Adventure Animation Comedy Fantasy	1995	0.08443
26053	124919	The Wind in the Willows (1995)	Adventure Animation Children Comedy	1995	0.08443
664	673	Space Jam (1996)	Adventure Animation Children Comedy Fantasy Sc...	1996	0.08873

COMME VOUS POUVEZ LE CONSTATER, TOUS LES FILMS RECOMMANDÉS ONT LE MEME ANNÉE ET LES GENRES SIMILAIRES À CEUX DU FILM "TOY STORY (1995)".

## Métriques utilisé dans le 'Content Based' :

### Cosine Similarity :

Métrique utilisée pour mesurer la similarité entre deux vecteurs de caractéristiques, tels que les genres des films.  
La similarité cosinus mesure l'angle entre les vecteurs dans un espace multidimensionnel, donnant une mesure de similarité indépendante de la magnitude des vecteurs.  
Utilisée dans le cas où les caractéristiques sont binaires (par exemple, la présence ou l'absence d'un genre de film).

### Euclidean Distance :

Mesure de la distance entre deux points dans un espace euclidien.  
Souvent utilisée pour mesurer la similarité entre des caractéristiques continues, telles que l'année de sortie des films.  
Plus la distance euclidienne entre deux points est petite, plus ils sont similaires.

## Collaborative\_User\_Based :

```
Calculate_Rating = all_in_one.dropna(axis = 0, subset = ['title'])
movie_ratingCount = (Calculate_Rating.
    groupby(by = ['title'])['rating'].
    count().
    reset_index().
    rename(columns = {'rating': 'totalRatingCount'}).
    [['title', 'totalRatingCount']])
)
movie_ratingCount.head()
```

	title	totalRatingCount
0	#chicagoGirl: The Social Network Takes on a Di...	2
1	\$ (Dollars) (1971)	11
2	\$5 a Day (2008)	27
3	\$9.99 (2008)	32
4	Sellebrity (Sellebrity) (2012)	2

**CALCUL DU NOMBRE D'ÉVALUATIONS PAR FILM :**  
POUR CHAQUE FILM, LE NOMBRE TOTAL D'ÉVALUATIONS REÇUES EST CALCULÉ. CETTE INFORMATION EST UTILISÉE POUR FILTRER LES FILMS, EN NE GARDANT QUE CEUX AYANT UN NOMBRE MINIMAL D'ÉVALUATIONS, POUR S'ASSURER QUE LES RECOMMANDATIONS SOIENT BASÉES SUR DES FILMS SUFFISAMMENT POPULAIRES.

```
combine_all = Calculate_Rating.merge(movie_ratingCount, left_on = 'title', right_on = 'title', how = 'left')
combine_all.head()
```

	userId	movieId	rating	title	totalRatingCount
0	1	2	3.5	Jumanji (1995)	13780
1	5	2	3.0	Jumanji (1995)	13780
2	13	2	3.0	Jumanji (1995)	13780
3	29	2	3.0	Jumanji (1995)	13780
4	34	2	3.0	Jumanji (1995)	13780

## FILTRAGE POUR GARDER SEULEMENT LES FILMS POPULAIRES

```
minimum_number_of_ratings = 1000
rating_popular_movie= combine_all.query('totalRatingCount >= @minimum_number_of_ratings')
rating_popular_movie.head()
```

	userId	movieId	rating	title	totalRatingCount
0	1	2	3.5	Jumanji (1995)	13780
1	5	2	3.0	Jumanji (1995)	13780
2	13	2	3.0	Jumanji (1995)	13780
3	29	2	3.0	Jumanji (1995)	13780
4	34	2	3.0	Jumanji (1995)	13780

## 3. CRÉATION DE LA MATRICE UTILISATEUR-FILM :

**PIVOTAGE DES DONNÉES :** LES DONNÉES SONT RESTRUCTURÉES SOUS FORME DE MATRICE OÙ LES LIGNES REPRÉSENTENT LES UTILISATEURS, LES COLONNES REPRÉSENTENT LES FILMS, ET LES VALEURS SONT LES ÉVALUATIONS DONNÉES PAR LES UTILISATEURS AUX FILMS.

**REMPLISSAGE DES VALEURS MANQUANTES :** LES ÉVALUATIONS MANQUANTES DANS LA MATRICE SONT REMPLIES AVEC ZÉRO, INDiquant UNE ABSENCE D'ÉVALUATION.

```
matrix_pivot = rating_popular_movie.pivot_table(index='userId',columns='title',values='rating').fillna(0)
```

```
from scipy.sparse import csr_matrix
movie_features_df_matrix = csr_matrix(matrix_pivot.values)
```

title	'Burbs, The (1989)	(500) Days of Summer (2009)	*batteries not included (1987)	Things I Hate About You (1999)	10,000 BC (2008)	101 Dalmatians (1996)	Dalmatians (One Hundred and One Dalmatians) (1961)	12 Angry Men (1957)	127 Hours (2010)	13 Going on 30 (2004)	...	Young Guns II (1990)	Young Sherlock Holmes (1985)	and Miri Make a Porno (2008)	Zero Effect (1998)	Zodiac (2007)	Zombieland (2009)	Zoo
userId																		
1	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	
2	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	
3	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	
4	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	
5	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	
85784	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	
85785	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	
85786	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	
85787	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	
85788	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	

85788 rows × 2341 columns

## 4. MODÈLE DE RECOMMANDATION :

**CONSTRUCTION DU MODÈLE K-NN : UN MODÈLE DES K PLUS PROCHES VOISINS EST ENTRAÎNÉ SUR LA MATRICE UTILISATEUR-FILM, UTILISANT LA MESURE DE DISTANCE COSINUS POUR ÉVALUER LA SIMILARITÉ ENTRE UTILISATEURS.**

**RECHERCHE DES VOISINS : POUR UN UTILISATEUR DONNÉ, LE SCRIPT RECHERCHE LES UTILISATEURS LES PLUS SIMILAIRES EN UTILISANT LE MODÈLE K-NN.**

```
from sklearn.neighbors import NearestNeighbors

model_knn = NearestNeighbors(metric = 'cosine', algorithm = 'brute')
model_knn.fit(movie_features_df_matrix)

NearestNeighbors(algorithm='brute', metric='cosine')
```

## 5. GÉNÉRATION DE RECOMMANDATIONS :

### Sélection des évaluations des utilisateurs similaires :

Les films évalués positivement par les utilisateurs similaires sont recueillis.

### Filtrage des films déjà évalués par l'utilisateur cible :

Les films déjà notés par l'utilisateur pour lequel les recommandations sont générées sont retirés de la liste des recommandations potentielles.

### Sélection finale :

Les films sont triés par évaluation et par nombre d'évaluations, et les meilleurs sont sélectionnés pour la recommandation.

## La fonction de recommendation des films en utilisant userId ,nombre\_users ,Top\_recommendation :

```
def recommend(userId=10, n_users=5, rec_top_n=10):

    distances, indices = model_knn.kneighbors(matrix_pivot.loc[matrix_pivot.index==userId].values.reshape(1, -1), n_neighbors = n_users)
    user_ids = []
    for index in range(0, len(distances.flatten())):
        user_ids.append(matrix_pivot.index[indices.flatten()[index]])
        if index == 0: # the movie chosen
            print(f"Users similar with user having user_id: {userId}")
            print("-----")
        else:
            print(f"{index}: {matrix_pivot.index[indices.flatten()[index]]} (dist: {distances.flatten()[index]})")

    # select movies that were highly ranked by the most similar users.

    # look only for movies highly rated by the similar users, not the current user
    candidate_user_ids = user_ids[1:]
    sel_ratings = rating_popular_movie.loc[rating_popular_movie.userId.isin(candidate_user_ids)]
    # sort by best ratings and total rating count
    sel_ratings = sel_ratings.sort_values(by=["rating", "totalRatingCount"], ascending=False)
    # eliminate from the selection movies that were ranked already by the current user
    movies_rated_by_targeted_user = list(rating_popular_movie.loc[rating_popular_movie.userId==user_ids[0]]["movieId"].values)
    sel_ratings = sel_ratings.loc[~sel_ratings.movieId.isin(movies_rated_by_targeted_user)]
    # aggregate and count total ratings and total totalRatingCount
    agg_sel_ratings = sel_ratings.groupby(["title", "rating"])["totalRatingCount"].max().reset_index()
    agg_sel_ratings.columns = ["title", "rating", "total_ratings"]
    agg_sel_ratings = agg_sel_ratings.sort_values(by=["rating", "total_ratings"], ascending=False)
    # only select top n (default top 10 here)
    rec_list = agg_sel_ratings["title"].head(rec_top_n).values
    print(f"\nMovies recommended to user_id: {user_ids[0]}\n-----")
    for i, rec in enumerate(rec_list):
        print(f"{i+1}: {rec}")


```

## Execution de Fonctions :

Type *Markdown* and *LaTeX*:  $\alpha^2$

```
recommend(10, 5, 10)

Users similar with user having user_id: 10
-----
1: 55 (dist: 0.5052522420883179)
2: 10420 (dist: 0.5308452844619751)
3: 41246 (dist: 0.5530043244361877)
4: 37415 (dist: 0.5623594522476196)
5: 54649 (dist: 0.5722603797912598)

Movies recommended to user_id: 10
-----
1: Terminator 2: Judgment Day (1991)
2: One Flew Over the Cuckoo's Nest (1975)
3: Aliens (1986)
4: Dr. Strangelove or: How I Learned to Stop Worrying and Love the Bomb (1964)
5: Platoon (1986)
6: Rocky (1976)
7: Boot, Das (Boat, The) (1981)
8: Goldfinger (1964)
9: Great Escape, The (1963)
10: Deer Hunter, The (1978)
```

# CONCLUSION

EN CONCLUSION, LA MISE EN ŒUVRE D'UN SYSTÈME DE RECOMMANDATION COMBINANT QUATRE APPROCHES DISTINCTES, À SAVOIR SVD, CONTENT-BASED, ITEM-BASED ET USER-BASED, PRÉSENTE UN POTENTIEL PROMETTEUR POUR OFFRIR DES SUGGESTIONS PERTINENTES ET PERSONNALISÉES AUX UTILISATEURS.

L'APPROCHE SVD, BASÉE SUR LA FACTORISATION MATRICIELLE, PERMET D'EXPLOITER LES RELATIONS LATENTES ENTRE LES UTILISATEURS ET LES ITEMS, CAPTURANT DES PATTERNS DE PRÉFÉRENCES ET D'INTERACTIONS QUI NE SONT PAS DIRECTEMENT OBSERVABLES.

LES APPROCHES CONTENT-BASED ET ITEM-BASED, QUANT À ELLES, S'APPUIENT SUR LES CARACTÉRISTIQUES DES ITEMS ET LES PROFILS DES UTILISATEURS POUR IDENTIFIER DES SIMILARITÉS ET PROPOSER DES ITEMS SUSCEPTIBLES DE CORRESPONDRE AUX GOÛTS DES UTILISATEURS.

ENFIN, L'APPROCHE USER-BASED SE BASE SUR LES COMPORTEMENTS D'UTILISATEURS SIMILAIRES POUR SUGGÉRER DES ITEMS APPRÉCIÉS PAR CES UTILISATEURS PROCHES.

LA MISE EN ŒUVRE RÉUSSIE DE CE SYSTÈME IMPLIQUE UNE ÉVALUATION RIGOUREUSE DES DIFFÉRENTES APPROCHES INDIVIDUELLES ET DE LEURS COMBINAISONS, EN S'APPUYANT SUR DES MÉTRIQUES D'ÉVALUATION PERTINENTES ET DES JEUX DE DONNÉES ADÉQUATS.

DE PLUS, L'OPTIMISATION DES HYPERPARAMÈTRES ET L'INTÉGRATION DE TECHNIQUES DE GESTION DE LA DIVERSITÉ ET DE L'EXPLICABILITÉ DES RECOMMANDATIONS CONTRIBUERONT À L'AMÉLIORATION CONTINUE DES PERFORMANCES ET À L'ADOPTION PAR LES UTILISATEURS.

EN SOMME, LE DÉVELOPPEMENT D'UN SYSTÈME DE RECOMMANDATION HYBRIDE INTÉGRANT LES APPROCHES SVD, CONTENT-BASED, ITEM-BASED ET USER-BASED OUVRE LA VOIE À UNE PERSONNALISATION ACCRUE ET À UNE EXPÉRIENCE UTILISATEUR PLUS SATISFAISANTE.