Projet P3

Organisation des données pour un système de recommandation

Université du Québec en Outaouais

Titre du cours : INF1473 - Entreposage et prospection des données

Nom et Prénom : Sankara Kabem Abdoul Charif

Kouyate Yasmine Jawad

Date : 20 décembre 2024

Table des matières

I Conception du modele en étoile 3
I.1 Analyse de données 3
I.2 Définition des Dimensions 3
II Développement du Processus ETL, Construction de la base de données et Operations OLAP 4
II.1 Développement du Processus ETL 4
II.2 Construction de la base de données et Operations OLAP 7
III Clustering, classification et profil des utilisateurs 13
III.1 Création de profil utilisateur et vecteur Film 13
III.2 Clustering 14
III.3 Classification 16

Introduction

Avec l'explosion des données numériques, les systèmes de recommandation sont devenus indispensables pour aider les utilisateurs à naviguer dans une vaste quantité d'informations. Utilisés sur des plateformes de streaming et de commerce en ligne, ces systèmes prédisent les préférences des utilisateurs et leur fournissent des suggestions personnalisées. Le projet vise à créer un entrepôt de données structurant les évaluations des utilisateurs pour développer un système de recommandation.

Le projet utilise le jeu de données MovieLens, fourni par le laboratoire GroupLens, un ensemble de données réputé dans la recherche sur les systèmes de recommandation. Il contient des évaluations anonymes de films par des utilisateurs, ainsi que des informations détaillées sur les films. Les préférences des utilisateurs sont représentées par des notes attribuées aux films, permettant d'analyser leurs goûts et de créer des profils comportementaux.

L'objectif principal est de créer un entrepôt de données relationnel pour analyser les informations sur les utilisateurs et les films, facilitant ainsi la mise en place d'un système de recommandation personnalisé. Le projet se divise en plusieurs étapes clés :

1. **Conception de l'entrepôt de données** : Modélisation d'un schéma relationnel pour structurer les données des utilisateurs et des films.
2. **Création du processus ETL** : Développement d'un pipeline automatisé pour extraire, transformer et charger les données dans l'entrepôt.
3. **Clustering des utilisateurs** : Identification des groupes d'utilisateurs basés sur leurs préférences cinématographiques.
4. **Classification des nouveaux utilisateurs** : Application d'un modèle supervisé pour prédire les profils des nouveaux utilisateurs.

I Conception du modele en étoile

I.1 Analyse de données

L'analyse du jeu de données *MovieLens* a pour objectif d'extraire des informations clés sur les préférences cinématographiques des utilisateurs, les tendances des évaluations de films, ainsi que les caractéristiques principales des œuvres, en fonction des variables disponibles. La base de données comprend quatre fichiers CSV : **ratings**, **movies**, **tags** et **links**. Cependant, nous avons décidé d'utiliser uniquement les fichiers **movies** et **ratings**, car ils contiennent les données les plus pertinentes pour un système de recommandation.

Description et Exploration des Données

- **User_ID** : L'identifiant unique pour chaque utilisateur permet de suivre ses évaluations et de construire un profil personnalisé basé sur ses interactions.
- **Movie_ID** : L'identifiant unique pour chaque film facilite l'association des évaluations avec les œuvres correspondantes.
- **Title et Genres** : Ces informations décrivent les films en fonction de leurs titres et catégories, permettant une segmentation par type de contenu (comédie, drame, action, etc.).
- **Rating** : La note attribuée par un utilisateur à un film indique son niveau d'appréciation, servant à modéliser ses préférences.
- **Timestamp** : L'horodatage des évaluations aide à identifier les tendances temporelles, comme les périodes de forte activité ou les films les plus populaires à un moment donné.

I.2 Définition des Dimensions

Pour la création de notre entrepôt de données, nous avons transformé le jeu de données MovieLens afin d'en extraire deux tables de dimensions et une table de faits. Ce modèle en étoile permet une analyse détaillée et structurée des préférences et des comportements des utilisateurs.

Tables de Dimensions

- **Dimension Utilisateur** : Cette dimension contient des informations détaillées sur les utilisateurs, telles que leur identifiant unique, leur liste de films évalués et des caractéristiques de leur comportement, facilitant les analyses de segmentation et de personnalisation.
- **Dimension Movies** : Inclut des informations spécifiques sur les films, comme le titre, les genres et l'identifiant unique. Cela permet d'effectuer des analyses de tendance par type de contenu et de popularité des films.

Table de Faits

- **Faits Ratings_Fact** : Cette table contient toutes les mesures nécessaires pour analyser les interactions entre les utilisateurs et les films. Elle inclut les identifiants des différentes dimensions ainsi que les informations sur les évaluations notes et horodatage que nous avons convertis en Date time.

Le schéma en étoile ci-dessous représente les relations entre les tables dans le modèle de données, permettant une navigation simple et des requêtes analytiques performantes.

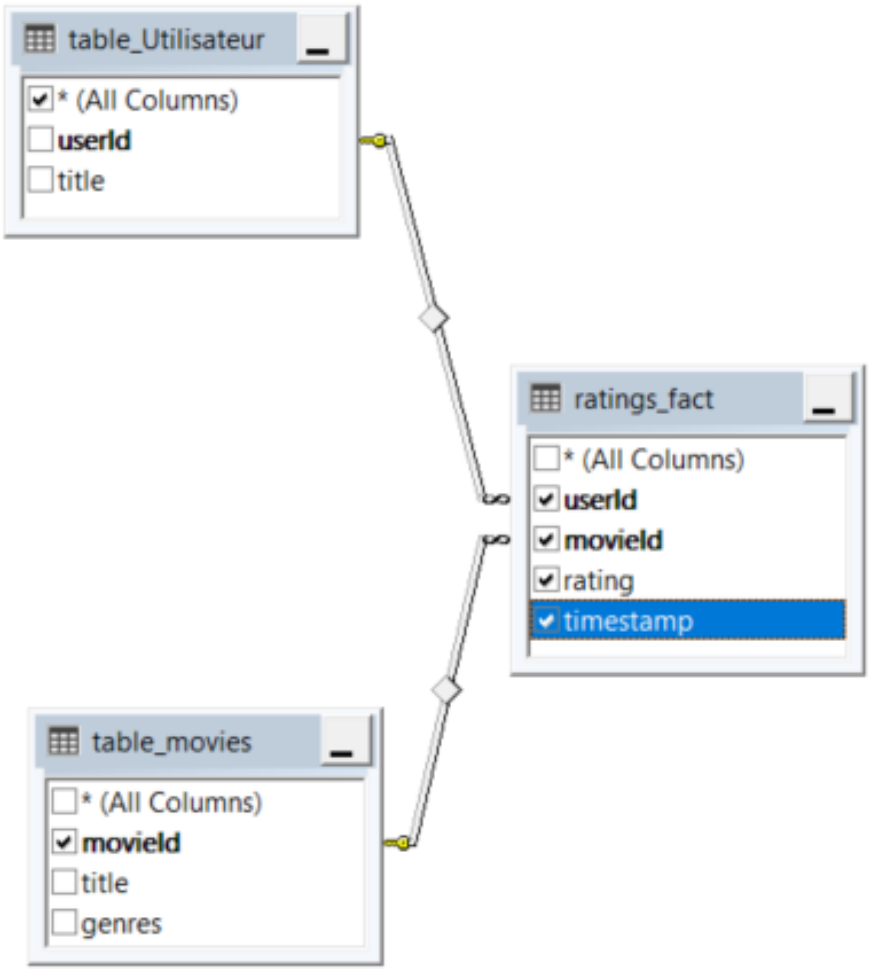


FIGURE 1 – Modèle en étoile

II Développement du Processus ETL, Construction de la base de données et Operations OLAP

II.1 Développement du Processus ETL

Pour le processus ETL de notre projet, nous avons utilisé **Google Colab** afin d'importer les données, de les extraire, puis de les transformer avant de les charger dans **SQL Server**. Le processus ETL sera expliqué en plusieurs étapes et illustré par des captures d'écran montrant les résultats. En pièce jointe de ce document, vous trouverez le fichier **ETL.ipynb** qui vous permettra de vérifier le processus. Cette section résume, en quelques points, les étapes clés du processus ETL.

Importation des données dans google colab

```
[7] import pandas as pd
    import numpy as np
    import re
    from sklearn.feature_extraction.text import TfidfVectorizer
    from sklearn.metrics.pairwise import cosine_similarity
    from sklearn.model_selection import train_test_split
```

Chargement des données

```
[4]
    url_movies="/content/movies.csv"
    url_ratings="/content/ratings.csv"
```

```
[5] # Lire les fichiers CSV
    df_movies = pd.read_csv(url_movies)
    df_ratings = pd.read_csv(url_ratings)
```

FIGURE 2 – Importation des données

Échantillonnage

Après avoir importé nos données, nous avons procédé à un échantillonnage, car le volume des données initiales était trop important. Pour garantir que l'échantillon reste représentatif de l'ensemble des données, nous avons utilisé un *échantillonnage stratifié* à hauteur de 10 %. Cette méthode consiste à diviser les données en groupes homogènes appelés « strates » selon une ou plusieurs variables d'intérêt, puis à prélever un échantillon proportionnel dans chaque strate. Cela permet de conserver la distribution des caractéristiques principales dans l'échantillon.

Echantillonage des données

Nous allons prendre un echantillons stratifés des données afin de garder la distribution et prendre que 10% des données

```
[8] df_movies = pd.DataFrame(df_movies)

    # Échantillonnage stratifié
    movies, _ = train_test_split(
        df_movies,
        train_size=0.1,  # Garde 10% des données
        stratify=None,   # Pas de colonne catégorielle spécifique ici, peut être ajusté si nécessaire
        random_state=42  # Assure la reproductibilité
    )

    # Résultat
    print("Données originales :")
    print(df_movies)

    print("\nÉchantillon stratifié (10%) :")
    print(movies)
```

FIGURE 3 – Échantillonnage

Nous avons réduire le volume des données en passant de 87 585 lignes à 8 758 lignes pour le fichier movies :

```
Données originales :
       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)
...        ...                                 ...
87580   292731           The Monroy Affaire (2022)
87581   292737          Shelter in Solitude (2023)
87582   292753                        Orca (2023)
87583   292755              The Angry Breed (1968)
87584   292757           Race to the Summit (2023)

                                        genres
0      Adventure|Animation|Children|Comedy|Fantasy
1                       Adventure|Children|Fantasy
2                                  Comedy|Romance
3                           Comedy|Drama|Romance
4                                         Comedy
...                                          ...
87580                                     Drama
87581                              Comedy|Drama
87582                                     Drama
87583                                     Drama
87584                 Action|Adventure|Documentary

[87585 rows x 3 columns]
```

```
Échantillon stratifié (10%) :
       movieId                                          title  \
4947      5052                   Time of Favor (Ha-Hesder) (2000)
74163   242112                           The New Frontier (1935)
43220   164566                             Cemento Armato (2007)
9967     33585                                    9 Songs (2004)
65629   213081                              Cotton Candy (1978)
...        ...                                              ...
6265      6383  2 Fast 2 Furious (Fast and the Furious 2, The)...
54886   189071                                     The Tale (2018)
76820   254506             Hiroshima Nagasaki August, 1945 (1970)
860        879                                     Relic, The (1997)
15795    83264                          Band Baaja Baaraat (2010)

                     genres
4947              Drama|War
74163               Western
43220              Thriller
9967           Drama|Romance
65629                 Comedy
...                      ...
6265   Action|Crime|Thriller
54886                 Drama
76820        Documentary|War
860          Horror|Thriller
15795   Comedy|Drama|Musical

[8758 rows x 3 columns]
```

FIGURE 4 – Résultats de échantillonnage

Transformations et création des tables

1. Avant de commencer à transformer nos données et à créer nos différentes tables, nous avons d'abord réalisé une analyse exploratoire afin de vérifier la présence de doublons ou de valeurs nulles dans nos données. Cette vérification a donné des résultats impeccables : aucune valeur dupliquée ou nulle n'a été détectée.

```
[14] print('Combien de données dupliquer dans la table rating: ',ratings.duplicated().sum())
     print('Combien de données dupliquer dans la table movies: ',movies.duplicated().sum())
     print('Combien de valeur nulle dans la table rating: ', ratings.isnull().sum())
     print('Combien de valeur nulle dans la table movies: ', movies.isnull().sum())
```

```
Combien de données dupliquer dans la table rating:  0
Combien de données dupliquer dans la table movies:  0
Combien de valeur nulle dans la table rating:  userId       0
movieId      0
rating       0
timestamp    0
dtype: int64
Combien de valeur nulle dans la table movies:  movieId    0
title      0
genres     0
dtype: int64
```

FIGURE 5 – EDA

2. Après cette étape, nous avons transformé les données des fichiers **movies** et **ratings**. Nous avons notamment forcé le formatage des données du fichier **ratings** en types entiers (`int`) et en format date-heure (`datetime`) afin de faciliter leur traitement et leur chargement.

```python
# Convertir les autres colonnes en types appropriés
ratings['userId'] = ratings['userId'].astype(int)
ratings['movieId'] = ratings['movieId'].astype(int)
ratings['rating'] = ratings['rating'].astype(int)
# Supprimer les lignes avec des valeurs NaN (causées par des valeurs non convertibles)
ratings = ratings.dropna(subset=['rating'])

# Vérification
print(ratings.dtypes)
print(ratings.head())
```

Afficher la sortie masquée

```python
[19] # Convertir la colonne 'timestamp' en date
ratings['timestamp'] = pd.to_datetime(ratings['timestamp'], unit='s')

# Afficher le résultat pour vérifier
print(ratings.head())
```

```
        userId  movieId  rating           timestamp
431370    2818   168760       4 2021-07-30 22:11:53
64739      391     4034       5 2001-04-01 08:16:35
267141    1731     1073       3 2007-05-25 09:41:22
304579    1950    39183       3 2021-03-08 16:27:47
138359     891     5378       4 2010-05-12 00:54:29
```

FIGURE 6 – Formatage des données

3. Pour terminer, nous avons créé et chargé les différentes tables avant de les exporter vers SQL Server. Parmi ces tables, nous avons notamment créé la table **Utilisateur**, qui contient l'identifiant de chaque utilisateur ainsi que la liste des films auxquels ils ont attribué une évaluation.

```python
[22] merged = pd.merge(ratings, movies, on='movieId', how='inner')

# Regroupement par userId et création d'une colonne avec la liste des films
Utilisateur = merged.groupby('userId')['title'].agg(lambda x: list(x)).reset_index()

# Affichage de la table avec la liste des films par utilisateur
print(Utilisateur)
```

```
      userId                                              title
0          1      [Graduate, The (1967), Apocalypse Now (1979)]
1          2                      [Beauty and the Beast (1991)]
2          3                                [Waterworld (1995)]
3          5                              [Crimson Tide (1995)]
4          7  [Dumb & Dumber (Dumb and Dumber) (1994), Water...
...      ...                                                ...
2138    3717  [Bullet Train (2022), Great Gatsby, The (2013)...
2139    3718  [Halloween (1978), Die Hard: With a Vengeance ...
2140    3720                       [Edward Scissorhands (1990)]
2141    3721  [Highlander (1986), Beavis and Butt-Head Do Am...
2142    3722  [Cowboy Bebop: The Movie (Cowboy Bebop: Tengok...

[2143 rows x 2 columns]
```

FIGURE 7 – Creation de la table user

II.2 Construction de la base de données et Operations OLAP

Importation des données dans SQL SERVER

Nous avons ensuite créé notre base de données en important directement nos tables issues des fichiers CSV à l'aide de l'outil **Import Flat File** de SQL Server. Cet outil permet de charger rapidement des fichiers CSV dans une base de données en détectant automatiquement la structure des colonnes et en générant un schéma adapté. Une fois les fichiers importés, nous avons défini les différentes clés étrangères afin de lier les dimensions à la table de faits.

Pour illustrer, prenons l'exemple de la table de faits **ratings**.

— **Etape 1 :**Accéder a Import Flat file via Taks dans Sql Server

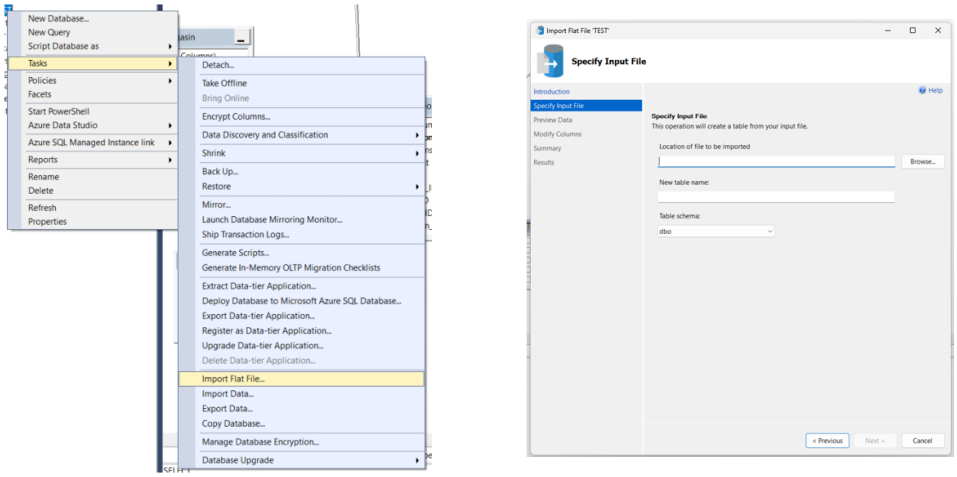


FIGURE 8 – Import Flat file

— **Etape 2 :**Après avoir sélectionné le fichier, l'outil nous présente la structure de la table **ratings**. À ce stade, nous pouvons choisir les types de données appropriés pour chaque colonne et définir notre clé primaire. Dans notre cas, nous avons choisi **movieId** et **userId** comme clés primaires. Dans le contexte de la table **ratings**, la combinaison de **movieId** (l'identifiant du film) et **userId** (l'identifiant de l'utilisateur) permet d'assurer l'unicité de chaque évaluation de film, car chaque utilisateur peut évaluer un film une seule fois. Cette combinaison sert à éviter les doublons et à garantir l'intégrité des données.

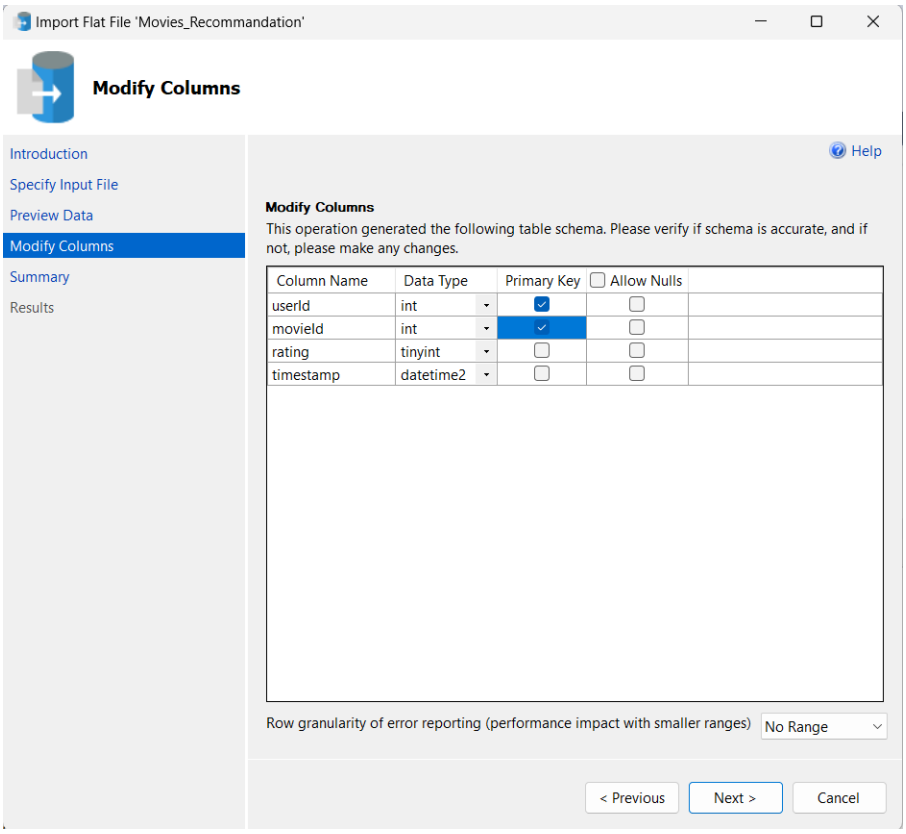


FIGURE 9 – Visualisation

— **Etape 3 :**Après avoir inséré les données et créé la table, nous avons ajouté les clés étrangères afin de lier les tables entre elles, à l'aide du code suivant :

Listing 1 – Ajout des clés étrangères dans la table ratings

```sql
-- Ajouter la cle etrangere pour la colonne userId
ALTER TABLE ratings_fact
ADD CONSTRAINT FK_ratings_fact_userId
FOREIGN KEY (userId) REFERENCES table_Utilisateur(userId)
ON DELETE CASCADE
ON UPDATE CASCADE;

-- Ajouter la cle etrangere pour la colonne movieId
ALTER TABLE ratings_fact
ADD CONSTRAINT FK_ratings_fact_movieId
FOREIGN KEY (movieId) REFERENCES table_movies(movieId)
ON DELETE CASCADE
ON UPDATE CASCADE;
```

— **Etape 4 :**Enfin, nous avons pu créer notre base de données avec les données déjà insérées et nos différentes clés. Ce processus nous a permis de structurer efficacement les informations et de garantir l'intégrité des données au sein du modèle relationnel.

La table `ratings_fact`, qui constitue l'élément central de notre modèle, est correctement reliée aux autres tables grâce à ses clés étrangères. Ces relations assurent une navigation fluide entre les dimensions et facilitent les requêtes analytiques.

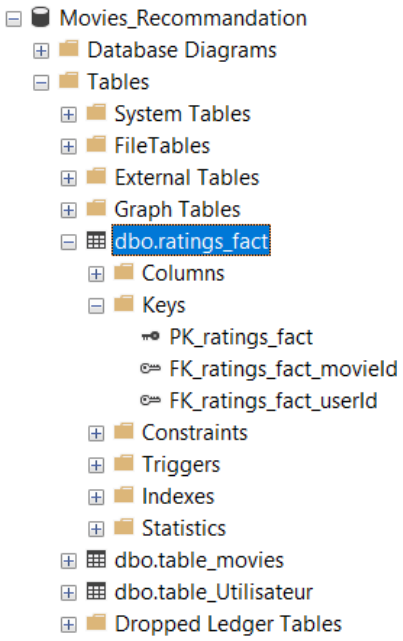


FIGURE 10 – Data Base

Operations OLAP

Utiliser des opérations OLAP (Slice, Dice, Drill-down, Roll-up) permet d'extraire des insights sur les tendances d'évaluation et les préférences par groupes d'utilisateurs. Ces opérations facilitent l'exploration des données sous différents angles pour obtenir des informations plus détaillées et pertinentes.

En annexe le fichier SQL contenant les différentes opérations.

SLICE

```sql
-- Operation SLICE

-- 1- Obtenir les evaluations des utilisateurs pour un film specifique "Grumpier Old Men (1995)".
SELECT u.userId, r.rating
FROM ratings_fact r
JOIN table_Utilisateur u ON r.userId = u.userId
JOIN table_movies m ON r.movieId = m.movieId
WHERE m.title = 'Grumpier Old Men (1995)';

\
-- Operation SLICE

-- 2- Obtenir les evaluations des films de genre "Drama" uniquement.
SELECT rf.userId, rf.movieId, rf.rating, rf.timestamp
FROM ratings_fact rf
JOIN table_movies tm ON rf.movieId = tm.movieId
WHERE tm.genres LIKE '%Drama%';
```

Results | Messages

	userId	rating
1	3847	4
2	3741	5
3	42594	5
4	42318	5
5	32119	3
6	32100	2
7	32080	3
8	31952	2
9	31703	3
10	31695	3
11	22169	3
12	22135	3
13	22088	2
14	21932	3
15	21908	3
16	21879	3
17	21853	2
18	13167	5
19	12819	3
20	12657	3
21	3639	4
22	3597	4
23	3575	5
24	3548	5
25	3533	3
26	3380	4
27	41950	1
28	41836	3
29	193430	3
30	193198	2
31	184716	3
32	184567	5
33	184352	3
34	184323	0

Query executed successfully.

(a) Resultat SLICE 1

Results | Messages

	userId	movieId	rating	timestamp
1	1	1094	4	1999-11-22 00:40:36.0000000
2	1	1272	5	1999-11-22 00:40:36.0000000
3	9	1653	5	2006-03-08 18:29:57.0000000
4	15	79132	4	2010-11-15 16:53:26.0000000
5	16	72641	3	2018-01-27 02:41:33.0000000
6	16	102407	4	2019-11-03 00:16:49.0000000
7	18	3481	3	2009-09-03 17:23:57.0000000
8	20	1682	1	2019-03-21 16:04:51.0000000
9	20	6711	3	2019-03-21 15:06:46.0000000
10	22	79132	4	2023-01-21 20:47:07.0000000
11	28	205	3	2000-06-19 18:13:33.0000000
12	28	345	4	2000-06-19 18:00:14.0000000
13	28	1178	4	2002-06-26 16:21:22.0000000
14	28	1190	3	2000-06-20 16:16:50.0000000
15	28	2125	3	2000-06-20 15:12:03.0000000
16	28	2260	2	2000-06-20 15:51:58.0000000
17	28	2447	4	2000-06-19 18:17:42.0000000
18	28	2942	2	2000-06-20 15:21:01.0000000
19	28	3504	4	2000-06-21 13:49:52.0000000
20	28	4903	4	2002-10-29 19:58:30.0000000
21	28	5956	4	2004-01-13 14:38:42.0000000
22	28	6493	2	2004-07-12 14:08:35.0000000
23	28	6947	4	2005-01-13 17:32:30.0000000
24	28	41527	4	2006-04-24 17:48:45.0000000
25	28	52712	3	2008-07-15 19:24:49.0000000
26	28	95309	4	2017-11-24 16:34:51.0000000
27	28	214242	4	2020-07-06 15:35:58.0000000
28	29	345	3	1996-10-11 17:53:21.0000000
29	32	500	3	1996-07-17 16:51:26.0000000
30	33	1682	5	2009-10-27 03:26:59.0000000
31	34	500	4	2007-11-11 00:43:25.0000000
32	34	539	3	2008-08-23 07:41:37.0000000
33	34	30707	2	2008-08-23 07:00:32.0000000
34	35	2291	4	2015-07-25 15:40:18.0000000

Query executed successfully.

(b) Resultat SLICE 2

DICE

```sql
-- Operation DICE

-- 1- Obtenir les evaluations des films du genre "Comedy" pour les utilisateurs avec une evaluation superieure 3.

SELECT u.userId, m.title, r.rating
FROM ratings_fact r
JOIN table_Utilisateur u ON r.userId = u.userId
JOIN table_movies m ON r.movieId = m.movieId
WHERE m.genres LIKE '%Comedy%'
AND r.rating > 3;

-- 2- Obtenir les evaluations des films de genre "Drama" ou "Comedy" par les utilisateurs ayant evalue plus de 3 films.

SELECT rf.userId, rf.movieId, rf.rating, rf.timestamp
FROM ratings_fact rf
JOIN table_movies tm ON rf.movieId = tm.movieId
WHERE (tm.genres LIKE '%Drama%' OR tm.genres LIKE '%Comedy%')
AND rf.userId IN (
    SELECT userId
    FROM ratings_fact
    GROUP BY userId
    HAVING COUNT(movieId) > 3
);
```

Results | Messages

	userId	title	rating
1	10	Shoot 'Em Up (2007)	4
2	14	Crackerjack (2002)	5
3	19	Beavis and Butt-Head Do America (1996)	5
4	28	Longest Yard, The (1974)	4
5	33	Drag Me to Hell (2009)	4
6	34	Mrs. Doubtfire (1993)	4
7	40	Truman Show, The (1998)	4
8	46	50/50 (2011)	5
9	56	Lost in Translation (2003)	5
10	59	Manhattan (1979)	4
11	74	Kung Fu Panda (2008)	4
12	76	Superbad (2007)	5
13	78	My Fair Lady (1964)	5
14	86	Graduate, The (1967)	5
15	16604	Kung Fu Panda (2008)	4
16	8035	The Passport (1990)	4
17	87	Austin Powers: The Spy Who Shagged Me (1999)	4
18	88	My Fair Lady (1964)	4
19	108	Animal House (1978)	5
20	114	Inside Out (2015)	5
21	120	Chasing Amy (1997)	4
22	134	My Fair Lady (1964)	5
23	138	Monty Python Live at the Hollywood Bowl (1982)	4
24	16605	Who Framed Roger Rabbit? (1988)	4
25	146	Army of Darkness (1993)	4
26	157	Role Models (2008)	4
27	158	Player, The (1992)	4
28	16606	Truman Show, The (1998)	4
29	159	Maverick (1994)	5
30	163	Meet the Parents (2000)	4
31	16610	Kingpin (1996)	4
32	174	Meet the Parents (2000)	4
33	16612	Big Chill, The (1983)	4
34	177	Truman Show, The (1998)	4

Query executed successfully.

(a) Resultat DICE 1

Results | Messages

	userId	movieId	rating	timestamp
1	1551	194	2	2007-05-23 08:29:15.0000000
2	1551	903	5	2015-08-08 06:23:32.0000000
3	1551	1190	3	2021-09-05 08:28:33.0000000
4	1551	1964	3	2005-08-04 10:01:17.0000000
5	1551	2009	3	2007-04-17 14:38:00.0000000
6	1551	2183	4	2005-08-03 21:22:33.0000000
7	1551	2291	2	2005-08-04 12:36:31.0000000
8	1551	4914	3	2005-08-06 07:27:03.0000000
9	1551	51111	3	2023-05-14 20:06:06.0000000
10	1551	57209	3	2012-11-07 19:41:14.0000000
11	1551	78041	2	2018-04-22 11:35:31.0000000
12	1551	83361	3	2011-01-23 23:43:12.0000000
13	1551	143859	3	2018-01-02 09:55:52.0000000
14	1555	1682	3	2023-07-08 21:36:30.0000000
15	1555	6711	4	2022-07-31 17:53:26.0000000
16	1576	231	4	2004-02-15 16:35:06.0000000
17	1576	852	3	2004-10-06 20:08:28.0000000
18	1590	180	4	2000-11-21 06:35:48.0000000
19	1590	279	4	2000-11-22 00:49:23.0000000
20	1590	1513	4	2000-11-21 07:02:18.0000000
21	1590	3504	4	2000-11-21 05:31:33.0000000
22	1617	500	5	2006-05-13 00:46:14.0000000
23	1617	1517	1	2016-11-13 02:32:52.0000000
24	1617	2058	3	2016-11-13 02:32:51.0000000
25	1617	2291	4	2016-11-13 02:32:57.0000000
26	1617	2987	3	2016-11-13 02:32:52.0000000
27	1617	3257	5	2016-11-13 02:33:01.0000000
28	1617	5956	4	2016-11-13 02:32:52.0000000
29	1617	6395	1	2020-08-23 01:22:22.0000000
30	1617	8360	3	2016-11-13 02:32:55.0000000
31	1617	27317	0	2020-08-23 01:22:17.0000000
32	1617	55094	4	2016-11-13 02:32:45.0000000
33	1617	89753	3	2020-08-23 01:22:14.0000000
34	1617	133782	3	2020-08-23 01:22:16.0000000

Query executed successfully.

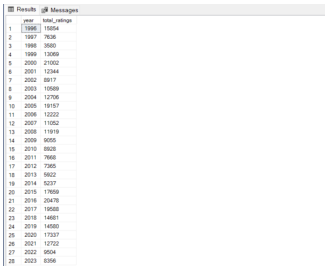
(b) Resultat DICE 2

DRILL-DOWN

```sql
-- Operation DRILL-DOWN

-- 1- Obtenir les evaluations detaillees pour un film specifique dans le genre "Drama".
SELECT u.userId, r.rating, r.timestamp
FROM ratings_fact r
JOIN table_Utilisateur u ON r.userId = u.userId
JOIN table_movies m ON r.movieId = m.movieId
WHERE m.genres LIKE '%Drama%'
AND m.title = 'Grumpier Old Men (1995)';

-- 2- Obtenir les evaluations des films par annee.
SELECT YEAR(rf.timestamp) AS year, COUNT(rf.rating) AS total_ratings
FROM ratings_fact rf
GROUP BY YEAR(rf.timestamp)
ORDER BY year;
```



Results | Messages

	year	total_ratings
1	1996	15854
2	1997	7636
3	1998	3580
4	1999	13069
5	2000	21002
6	2001	12344
7	2002	8917
8	2003	10569
9	2004	12706
10	2005	19157
11	2006	12222
12	2007	11052
13	2008	11919
14	2009	9055
15	2010	8828
16	2011	7668
17	2012	7365
18	2013	5622
19	2014	5237
20	2015	17669
21	2016	20476
22	2017	19588
23	2018	14681
24	2019	14580
25	2020	17337
26	2021	12732
27	2022	9504
28	2023	6356

FIGURE 13 – Drill_down

ROLL-UP

```sql
-- Operation ROLL-UP

-- 1- Obtenir le nombre total d evaluations par genre.

SELECT tm.genres, COUNT(rf.rating) AS total_ratings
FROM ratings_fact rf
JOIN table_movies tm ON rf.movieId = tm.movieId
GROUP BY tm.genres
WITH ROLLUP;

-- 2- Obtenir la moyenne des evaluations pour chaque film en regroupant par le titre du film.

SELECT
    m.title,
    AVG(r.rating) AS avg_rating
FROM ratings_fact r
JOIN table_movies m ON r.movieId = m.movieId
GROUP BY m.title
ORDER BY avg_rating DESC;
```

Results | Messages

	genres	total_ratings
1	(no genres listed)	450
2	Action	382
3	Action|Adventure	8017
4	Action|Adventure|Animation	59
5	Action|Adventure|Animation|Children	17
6	Action|Adventure|Animation|Children|Comedy	21
7	Action|Adventure|Animation|Children|Sci-Fi	11
8	Action|Adventure|Animation|Crime|Fantasy	5
9	Action|Adventure|Animation|Drama|Fantasy	21
10	Action|Adventure|Animation|Fantasy	21
11	Action|Adventure|Animation|Fantasy|Horror	28
12	Action|Adventure|Animation|Fantasy|IMAX	346
13	Action|Adventure|Animation|Horror	29
14	Action|Adventure|Animation|Sci-Fi	989
15	Action|Adventure|Children|Comedy	26
16	Action|Adventure|Children|Comedy|Fantasy	1
17	Action|Adventure|Children|Comedy|Sci-Fi	34
18	Action|Adventure|Children|Crime|Mystery|Thriller	16
19	Action|Adventure|Children|Drama	1
20	Action|Adventure|Children|Drama|Western	1
21	Action|Adventure|Children|IMAX	541
22	Action|Adventure|Comedy	5701
23	Action|Adventure|Comedy|Crime	2
24	Action|Adventure|Comedy|Crime|Sci-Fi	6
25	Action|Adventure|Comedy|Crime|Thriller	36
26	Action|Adventure|Comedy|Drama|Thriller	27
27	Action|Adventure|Comedy|Fantasy|Horror	1450
28	Action|Adventure|Comedy|Fantasy|Mystery	448
29	Action|Adventure|Comedy|Fantasy|Sci-Fi	1
30	Action|Adventure|Comedy|Mystery|Romance	1
31	Action|Adventure|Comedy|Romance	4
32	Action|Adventure|Comedy|Sci-Fi	1059
33	Action|Adventure|Crime	9
34	Action|Adventure|Crime|Drama|Fantasy|Horror|...	1

Query executed successfully.

(a) Resultat Roll up 1

Results | Messages

	title	avg_rating
1	Bereave (2015)	5
2	Plastic Paradise: The Great Pacific Garbage Patch ...	5
3	The Other Side (2015)	5
4	Five Dedicated to Ozu (2003)	5
5	Purple Plain, The (1954)	5
6	The House by the Sea (2017)	5
7	The Nun (2013)	5
8	Thug Rose: Mixed Martial Artist (2022)	5
9	Harmony (2015)	5
10	Uno: The Movie	5
11	Under the Bombs (2007)	5
12	History of the Eagles (2013)	5
13	Marsh, The (2006)	5
14	The Garden of Sinners - Chapter 5: Paradox Paradi...	5
15	Moana (1926)	5
16	588 Rue Paradis (Mother) (1992)	5
17	Pizza My Heart (2005)	5
18	Alvin Purple (1973)	5
19	The West (1996)	5
20	What She Said: The Art of Pauline Kael (2019)	5
21	SEE HEAR LOVE (2023)	5
22	Terror of Frankenstein (1977)	5
23	Illustrious Corpses (Cadaveri eccellenti) (1976)	5
24	A Fortunate Man (2018)	5
25	Parched (2015)	5
26	Paradigm (1969)	5
27	Ciao Ni! (1979)	5
28	Inventing David Geffen (2012)	5
29	End of the century (2019)	5
30	Diaries Notes and Sketches (Walden) (1969)	5
31	Wind (1996)	5
32	Blunder Below (1942)	5
33	Ghatak: Lethal (1996)	5
34	The Last Light (2014)	5

Query executed successfully.

(b) Resultat Roll up 2

III Clustering, classification et profil des utilisateurs

Dans cette section, nous présentons brièvement les étapes de clustering et de classification appliquées aux données des utilisateurs. Le clustering permet d'identifier des groupes distincts d'utilisateurs selon leurs préférences cinématographiques, chaque groupe étant caractérisé par une étiquette comportementale. Par la suite, la classification des nouveaux utilisateurs repose sur un modèle supervisé, capable de prédire leur étiquette en fonction de leurs premières interactions avec la base. Les résultats obtenus sont également discutés ici. Pour plus de détails sur le processus complet, veuillez consulter le fichier **Cluster_et_Classification.ipynb** fourni en pièce jointe.

III.1 Création de profil utilisateur et vecteur Film

Nous avons commencé par créer une base de données en effectuant la jointure des tables `movies` et `ratings`. Cette opération a permis de générer les différents profils utilisateurs ainsi que les vecteurs représentant les films.

1. **Etape 1 :**Création d'une base de données basé la jointure de `movies` et `ratings`.

```
[ ] # Fusion de `ratings` avec `movies`
    data = pd.merge(ratings, movies, on="movieId", how="inner")
```

```
[ ] data
```

	userId	movieId	rating	timestamp	title	genres
0	133236	1240	4	2017-02-12 15:51:30	Terminator, The (1984)	[Action, Sci-Fi, Thriller]
1	183219	140174	5	2021-02-14 18:43:55	Room (2015)	[Drama]
2	149399	165	3	1996-06-15 10:39:25	Die Hard: With a Vengeance (1995)	[Action, Crime, Thriller]
3	151719	1089	3	2005-01-26 13:20:32	Reservoir Dogs (1992)	[Crime, Mystery, Thriller]
4	196009	4914	3	2008-05-22 12:52:43	Breathless (À bout de souffle) (1960)	[Crime, Drama, Romance]
...	...	...	...	...	...	...
339122	134437	4239	5	2008-11-10 02:36:36	Blow (2001)	[Crime, Drama]
339123	58093	6703	2	2003-09-12 02:25:54	Order, The (2003)	[Horror, Mystery, Thriller]
339124	65991	345	3	1996-09-11 14:46:45	Adventures of Priscilla, Queen of the Desert, ...	[Comedy, Drama]
339125	60153	1333	2	2003-06-03 03:38:21	Birds, The (1963)	[Horror, Thriller]
339126	100887	224264	3	2023-05-06 22:39:52	Connected (2020)	[Adventure, Animation, Children, Comedy, Sci-Fi]

339127 rows × 6 columns

FIGURE 15 – jointure

2. **Etape 2 :** Nous avons utilisé la classe `MultiLabelBinarizer` de Scikit-learn pour transformer les genres de films en colonnes binaires. Chaque genre est représenté par une colonne indiquant sa présence (`1`) ou son absence (`0`) pour chaque film. Cette transformation a permis d'obtenir une structure claire et exploitable, comme illustré par le résultat suivant.

```
from sklearn.preprocessing import MultiLabelBinarizer

mlb = MultiLabelBinarizer()
genres_encoded = pd.DataFrame(mlb.fit_transform(movies['genres']), columns=mlb.classes_, index=movies.index)
movies = pd.concat([movies, genres_encoded], axis=1)
```

```
movies
```

	movieId	title	genres	(no genres listed)	Action	Adventure	Animation	Children	Comedy	Crime	...	Film-Noir	Horror	IMAX	Musical	Mystery	Romance	Sci-Fi	Thriller	War	Western
0	5052	Time of Favor (Ha-Hesder) (2000)	[Drama, War]	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0	1	0
1	242112	The New Frontier (1935)	[Western]	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0	0	1
2	164566	Cemento Armato (2007)	[Thriller]	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	1	0	0
3	33585	9 Songs (2004)	[Drama, Romance]	0	0	0	0	0	0	0	...	0	0	0	0	0	1	0	0	0	0
4	213081	Cotton Candy (1978)	[Comedy]	0	0	0	0	0	1	0	...	0	0	0	0	0	0	0	0	0	0
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
8753	6383	2 Fast 2 Furious (Fast and the Furious 2, The)...	[Action, Crime, Thriller]	0	1	0	0	0	0	1	...	0	0	0	0	0	0	0	1	0	0
8754	189071	The Tale (2018)	[Drama]	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0	0	0
8755	254506	Hiroshima Nagasaki August, 1945 (1970)	[Documentary, War]	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0	1	0
8756	879	Relic, The (1997)	[Horror, Thriller]	0	0	0	0	0	0	0	...	0	1	0	0	0	0	0	1	0	0
8757	83264	Band Baaja Baaraat (2010)	[Comedy, Drama, Musical]	0	0	0	0	0	1	0	...	0	0	0	1	0	0	0	0	0	0

8758 rows × 23 columns

FIGURE 16 – Vecteur film

3. **Etape 3 :** Nous avons utilisé une approche basée sur le regroupement des données par `userId` pour calculer les profils utilisateurs. En multipliant les colonnes binaires des genres par les notes des utilisateurs, puis en calculant la moyenne pour chaque utilisateur, nous avons généré un vecteur de profil représentatif de leurs préférences cinématographiques. Le résultat final est présenté ci-dessous.

```
user_profiles = data.groupby('userId').apply(lambda x: (x.iloc[:, -len(mlb.classes_):].T * x['rating']).mean(axis=1))
user_profiles = user_profiles.reset_index().rename(columns={0: "profile_vector"})
```

```
<ipython-input-62-f5589ede27f4>:1: DeprecationWarning: DataFrameGroupBy.apply operated on the grouping columns. This behavior is deprecated, and in a future version of pandas the grouping columns will be
  user_profiles = data.groupby('userId').apply(lambda x: (x.iloc[:, -len(mlb.classes_):].T * x['rating']).mean(axis=1))
```

```
user_profiles
```

	userId	(no genres listed)	Action	Adventure	Animation	Children	Comedy	Crime	Documentary	Drama	...	Film-Noir	Horror	IMAX	Musical	Mystery	Romance	Sci-Fi	Thriller	War	Western
0	1	0.0	0.000000	0.0	0.0	0.0	0.0	0.000000	0.0	4.50	...	0.0	0.0	0.0	0.0	0.0	2.0	0.0	2.00	2.5	0.0
1	3	0.0	2.666667	0.0	0.0	1.0	1.0	0.666667	0.0	0.00	...	0.0	0.0	0.0	0.0	0.0	1.0	2.0	1.00	0.0	0.0
2	5	0.0	3.000000	3.0	0.0	0.0	0.0	0.000000	0.0	0.00	...	0.0	0.0	0.0	0.0	0.0	0.0	3.0	0.00	0.0	0.0
3	6	0.0	5.000000	5.0	0.0	0.0	0.0	0.000000	0.0	0.00	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.00	0.0	0.0
4	7	0.0	3.000000	0.0	0.0	0.0	0.0	3.000000	0.0	0.00	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	3.00	0.0	0.0
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
114775	200943	0.0	0.500000	0.5	0.0	0.0	0.5	0.750000	0.0	1.75	...	0.0	0.0	0.0	0.0	0.0	0.0	2.0	1.75	0.0	0.0
114776	200944	0.0	1.500000	1.5	1.5	0.0	0.0	0.000000	0.0	1.50	...	0.0	0.0	1.5	0.0	0.0	0.0	0.0	0.00	0.0	0.0
114777	200945	0.0	0.000000	0.0	0.0	0.0	0.0	0.000000	0.0	5.00	...	0.0	0.0	0.0	0.0	0.0	2.5	0.0	2.50	0.0	0.0
114778	200947	0.0	4.000000	0.0	0.0	0.0	0.0	4.000000	0.0	4.00	...	0.0	0.0	4.0	0.0	4.0	0.0	4.0	4.00	0.0	0.0
114779	200948	0.0	1.400000	0.0	0.0	0.0	1.2	0.400000	0.0	2.60	...	0.0	0.0	0.0	0.0	0.0	1.8	1.8	1.00	0.0	0.0

114780 rows × 21 columns

FIGURE 17 – Profil

III.2 Clustering

Nous avons utilisé la méthode `StandardScaler` pour normaliser les vecteurs de profils utilisateurs, garantissant que chaque genre contribue équitablement à la distance dans l'espace des caractéristiques. Ensuite, l'algorithme de `K-means` a été appliqué pour regrouper les utilisateurs en trois clusters distincts, représentant différents groupes de préférences cinématographiques. Les résultats des clusters obtenus sont présentés ci-dessous.

```
[ ] from sklearn.preprocessing import StandardScaler

    # Exclure la colonne userId
    user_profiles_data = user_profiles.drop(columns=['userId'])

    # Normaliser les scores des genres
    scaler = StandardScaler()
    user_profiles_scaled = scaler.fit_transform(user_profiles_data)

    # Vérifier les données après normalisation
    print(user_profiles_scaled[:5])
```

Afficher la sortie masquée

```
from sklearn.cluster import KMeans

# Appliquer K-means pour créer des groupes d'utilisateurs
kmeans = KMeans(n_clusters=3, random_state=42)  # Ajustez le nombre de clusters si nécessaire
user_profiles['cluster'] = kmeans.fit_predict(user_profiles_scaled)

# Afficher les résultats des clusters
print(user_profiles[['userId', 'cluster']].head())
```

```
   userId  cluster
0       1        1
1       3        0
2       5        0
3       6        0
4       7        0
```

FIGURE 18 – Cluster

Après avoir identifié les différents clusters, nous avons souhaité visualiser la répartition des utilisateurs selon ces clusters. Cette visualisation nous a permis d'interpréter les résultats et de mieux comprendre la séparation des utilisateurs en fonction de leurs comportements ou préférences.

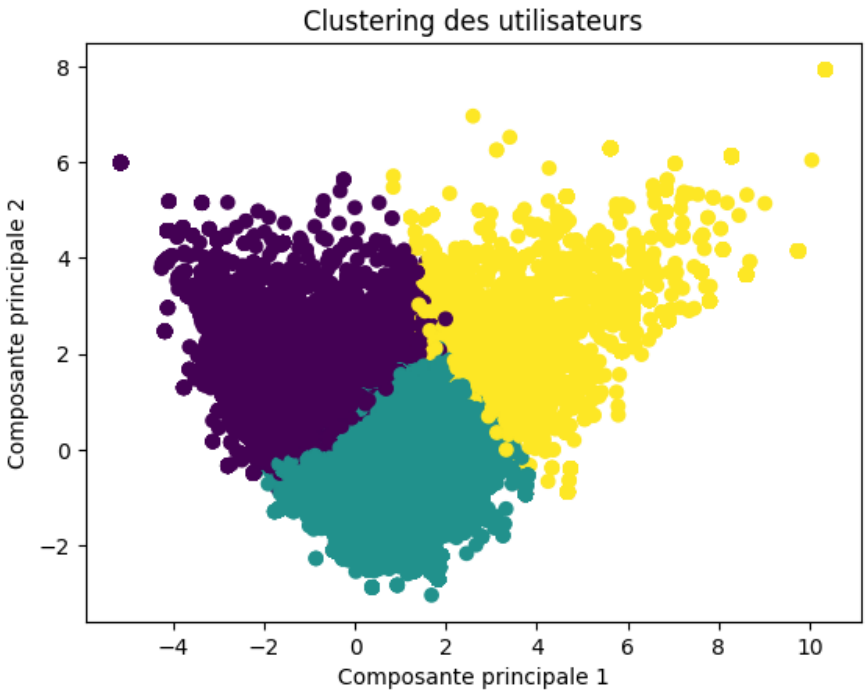


FIGURE 19 – Representation Cluster

Visualisation et Analyse des Clusters d'Utilisateurs :

Le clustering des utilisateurs a permis de regrouper ces derniers en trois groupes distincts (violet, jaune, vert) selon leurs comportements d'évaluation et leurs préférences cinématographiques. La séparation nette entre les clusters montre que l'algorithme a bien segmenté les utilisateurs, facilitant ainsi une analyse plus ciblée.

— **Cluster Jaune :** Un groupe d'utilisateurs avec des préférences variées, probablement intéressé par différents genres de films.

— **Clusters Violet et Vert :** Des groupes plus homogènes, représentant des utilisateurs ayant des goûts plus marqués pour certains genres ou types de films spécifiques.

L'utilisation des composantes principales a permis de projeter les données dans un espace de dimension réduite, mettant en évidence ces différences de comportement de manière plus claire.

Applications :

— **Segmentation des Utilisateurs :** Chaque cluster représente un groupe avec des préférences similaires, utile pour personnaliser les recommandations de films.

— **Détection des Comportements Atypiques :** Les points isolés dans le graphique peuvent correspondre à des utilisateurs avec des comportements rares ou uniques, comme des évaluations extrêmes.

III.3 Classification

Dans cette partie du projet, nous avons utilisé un classificateur Random Forest pour prédire à quel cluster un utilisateur nouvellement ajouté appartient, en utilisant ses caractéristiques comportementales.

— **Préparation des Données :** Les caractéristiques (*features*) sont extraites des profils d'utilisateurs normalisés, et la cible (*target*) correspond aux clusters identifiés précédemment.

— **Division des Données :** Les données sont séparées en deux ensembles : un pour l'entraînement du modèle et un autre pour tester sa performance (70% pour l'entraînement, 30% pour le test).

— **Entraînement du Modèle :** Un modèle de forêt aléatoire (Random Forest) est entraîné sur les données d'entraînement en utilisant la méthode `fit`.

— **Prédictions et Évaluation :** Le modèle fait des prédictions sur l'ensemble de test, et nous évaluons sa performance en affichant le rapport de classification, qui fournit des métriques comme la précision, le rappel et la F1-score, ainsi que la précision globale du modèle.

```python
from sklearn.ensemble import RandomForestClassifier
from sklearn.model_selection import train_test_split
from sklearn.metrics import classification_report, accuracy_score

# Préparation des données
X = user_profiles_scaled  # Caractéristiques (features)
y = user_profiles['cluster']  # Cible (target)

# Division des données en ensembles d'entraînement et de test
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=42)

# Entraîner le modèle Random Forest
clf = RandomForestClassifier(random_state=42)
clf.fit(X_train, y_train)

# Prédictions sur l'ensemble de test
y_pred = clf.predict(X_test)

# Évaluation du modèle
print("Classification Report:")
print(classification_report(y_test, y_pred))

# Précision globale
accuracy = accuracy_score(y_test, y_pred)
print(f"Précision du modèle : {accuracy:.2f}")
```

```
Classification Report:
              precision    recall  f1-score   support

           0       0.98      0.97      0.98      9965
           1       0.99      0.99      0.99     22432
           2       1.00      0.97      0.98      2037

    accuracy                           0.99     34434
   macro avg       0.99      0.98      0.98     34434
weighted avg       0.99      0.99      0.99     34434

Précision du modèle : 0.99
```

FIGURE 20 – Résultats classification

Interprétation des résultats :

- **Précision** : La précision est élevée pour tous les clusters, avec une précision globale de **0.99**. Cela signifie que le modèle fait une prédiction correcte pour presque toutes les observations.
- **Rappel** : Le rappel est également élevé, en particulier pour le cluster 1, avec un rappel de **0.99**. Cela signifie que le modèle identifie presque tous les utilisateurs appartenant à chaque cluster.
- **F1-Score** : Le score moyen global est de **0.98**, ce qui reflète un bon équilibre entre précision et rappel pour chaque cluster.
- **Support** : Le support représente le nombre d'éléments dans chaque classe. Le cluster 1 est le plus grand (22432 utilisateurs), suivi des clusters 0 (9965 utilisateurs) et 2 (2037 utilisateurs).
- **Précision globale du modèle** : La précision globale du modèle est de **0.99**, ce qui indique que 99% des prédictions sont correctes, ce qui est un excellent résultat.

Conclusion Générale

Ce projet a permis d'organiser des données pour un système de recommandation de films basé sur l'analyse des données de préférences cinématographiques des utilisateurs. À partir du jeu de données MovieLens, nous avons conçu un entrepôt de données et traité les informations sur les utilisateurs et les films pour créer des profils utilisateurs et des vecteurs films. En utilisant des techniques de clustering et de classification, nous avons segmenté les utilisateurs en groupes homogènes selon leurs préférences, puis nous avons prédit les groupes des utilisateurs nouveaux à l'aide d'un modèle supervisé basé sur les Random Forests. Les résultats obtenus montrent une excellente précision de prédiction avec un modèle performant à 99%.

La réduction de la dimensionnalité avec PCA a permis de visualiser clairement la séparation entre les clusters d'utilisateurs, offrant ainsi une meilleure compréhension des profils comportementaux. Enfin, des requêtes OLAP ont été utilisées pour analyser les tendances temporelles des votes et les classements des films les plus populaires.

Ce travail a non seulement démontré l'efficacité des algorithmes de clustering et de classification dans le domaine des recommandations personnalisées, mais aussi l'importance d'une gestion et d'une analyse de données bien structurées pour améliorer l'expérience utilisateur.