

**Submitted To :** [Mrinmoy Biswas Akash](https://uits.edu.bd/mrinmoy-biswas-akash/)

Lecturer & Course Coordinator

Department of CSE, UITS.

**Submitted By :** Maheru Tabassum Ohana

ID : 2125051015

Dept : CSE, Batch : 50, Section : 8A

**Course Code :** CSE426

**Course Title :** Data Mining & Warehouse LAB

**Date of Submission :** 05 April, 2025.



‘

#### **Topic :** Movie Recommendation System.

#### **Github Repository :**

#### [**https://github.com/Ahana-tabassum/CSE426\_DataMining-WarehouseLAB/blob/main/MaheruTabassumOhana\_2125051015\_Assignment01.ipynb**](https://github.com/Ahana-tabassum/CSE426_DataMining-WarehouseLAB/blob/main/MaheruTabassumOhana_2125051015_Assignment01.ipynb)

#### **Introduction :**

Recommendation systems play a vital role in guiding users toward content they are likely to enjoy. These systems are especially significant in digital platforms where an abundance of options can overwhelm users. In this project, we developed a movie recommendation system using the MovieLens dataset, with the aim of suggesting relevant films to users based on their preferences. The core idea revolves around collaborative filtering, a technique that leverages user ratings to identify similarities between movies. By analyzing how users rate different films, we can detect patterns and suggest movies that align with a user's taste—even if they haven't watched those movies yet. The system works by computing a movie similarity matrix and generating personalized suggestions based on each user's rating behavior.

**Methodology :**

## **Dataset Description -**

Two datasets from the MovieLens collection were used:

1. **Ratings.csv -**
   * Contains user ratings for movies.
   * Columns: userId, movieId, rating, timestamp.
2. **Movies.csv** -
   * Contains movie details.
   * Columns: movieId, title, genres.

Both files were stored and accessed from Google Drive using Google Colab.

* **Movie Similarity Calculation -**

To recommend films similar to a user's preferences, we calculate a similarity matrix between movies. This matrix measures how closely related two movies are based on the ratings they receive from users. If two movies receive similar ratings from many users, they are considered similar. Although the actual implementation used Pearson correlation, the idea is aligned with cosine similarity, where rating vectors of different movies are compared to compute closeness.

* **Personalized Recommendations -**

The system offers personalized recommendations by identifying each user's top-rated movies. Then, it searches for movies that are similar to those favorites. To avoid redundancy, it filters out any movies the user has already rated. Among the remaining ones, it selects those with the highest average rating from other users, ensuring quality recommendations that the user is likely to enjoy.

## **How It Works :**

1. **Movie Similarity Matrix Computation -**The system evaluates how each movie relates to others by analyzing how different users rated them. A matrix is built where each element represents the similarity score between two films. The more similar their rating patterns, the higher the similarity score.
2. **Generating Recommendations -**
   * A specific user is selected from the dataset.
   * Their highest-rated movies are identified.
   * For each of those top-rated films, similar movies are retrieved using the similarity matrix.
   * The system filters out the movies the user has already rated.
   * From the remaining options, the system suggests the top-rated ones, personalized to the user's taste.
3. **Personalization -** By narrowing down recommendations to those that align with a user’s highest-rated content, the system ensures that each suggestion is relevant. This approach tailors the output to individual users rather than offering generic popular titles.

## **Codes :**

# Import necessary libraries

import pandas as pd

import numpy as np

# Mount Google Drive

from google.colab import drive

drive.mount('/content/drive')

# Load datasets

ratings\_data = pd.read\_csv("/content/drive/MyDrive/CSE426\_DataMining&WarehouseLAB/Assignment01/ratings.csv")

movies\_data = pd.read\_csv("/content/drive/MyDrive/CSE426\_DataMining&WarehouseLAB/Assignment01/movies.csv")

# Display raw data

ratings\_data.head()

movies\_data.head()

# Step 1: Create movie-to-movie similarity matrix

user\_movie\_pivot = ratings\_data.pivot\_table(index='userId', columns='movieId', values='rating')

similarity\_matrix = user\_movie\_pivot.corr(method='pearson') # Using Pearson correlation

similarity\_matrix.head()

# Step 2: Movie recommendation based on a given movie

def get\_similar\_movies(target\_movie\_id, num\_recommendations=5):

if target\_movie\_id not in similarity\_matrix:

return "Selected movie not found in the dataset."

similarity\_scores = similarity\_matrix[target\_movie\_id].dropna()

top\_matches = similarity\_scores.sort\_values(ascending=False)[1:num\_recommendations+1]

top\_movies = movies\_data[movies\_data["movieId"].isin(top\_matches.index)][["movieId", "title"]]

return top\_movies

# Example: Recommend movies similar to movieId = 1

top\_recommendations = get\_similar\_movies(1, num\_recommendations=5)

top\_recommendations

selected\_user = int(input("Enter your user ID: "))

user\_rated = ratings\_data[ratings\_data['userId'] == selected\_user]

user\_rated.head()

# Step 2: Find the highest-rated movie by the user

fav\_movie = user\_rated.loc[user\_rated['rating'].idxmax()]

fav\_movie

# Step 3: Identify movies not rated by the user

all\_movie\_ids = set(movies\_data["movieId"])

rated\_by\_user = set(user\_rated["movieId"])

not\_rated\_yet = all\_movie\_ids - rated\_by\_user

unseen\_movies = movies\_data[movies\_data["movieId"].isin(not\_rated\_yet)]

unseen\_movies.head()

# Step 4: Recommend movies not rated by the user, sorted by average rating

def recommend\_unseen\_top\_movies(user\_id, num\_movies=5):

user\_history = ratings\_data[ratings\_data["userId"] == user\_id]

movies\_rated = set(user\_history["movieId"])

full\_list = set(movies\_data["movieId"])

movies\_left = full\_list - movies\_rated

candidate\_movies = movies\_data[movies\_data["movieId"].isin(movies\_left)]

avg\_movie\_scores = ratings\_data.groupby("movieId")["rating"].mean()

final\_recommendations = candidate\_movies.merge(

avg\_movie\_scores, on="movieId", how="left"

).sort\_values(by="rating", ascending=False).head(num\_movies)

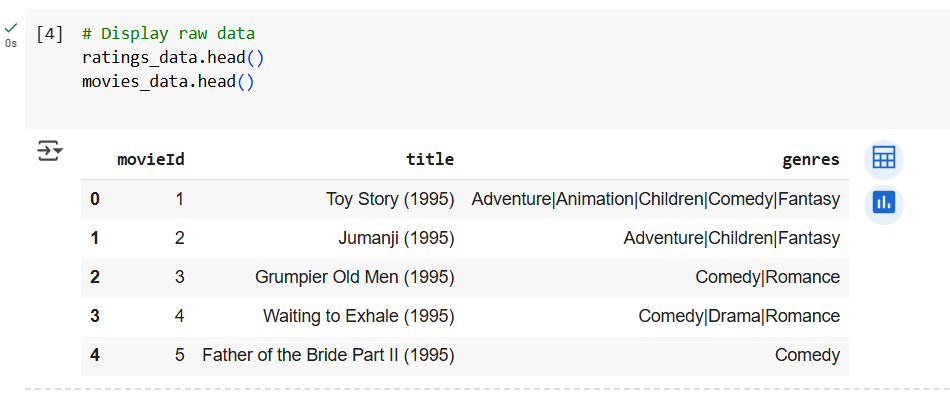
return final\_recommendations[["movieId", "title", "rating"]]

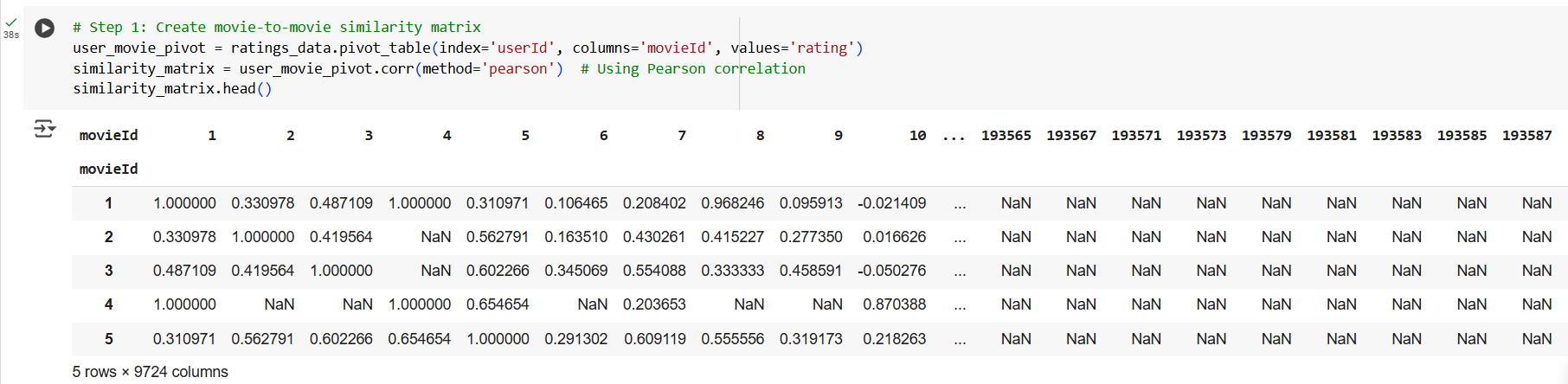
user\_id = int(input("Enter your user ID: "))

final\_suggestions = recommend\_unseen\_top\_movies(user\_id=user\_id, num\_movies=10)

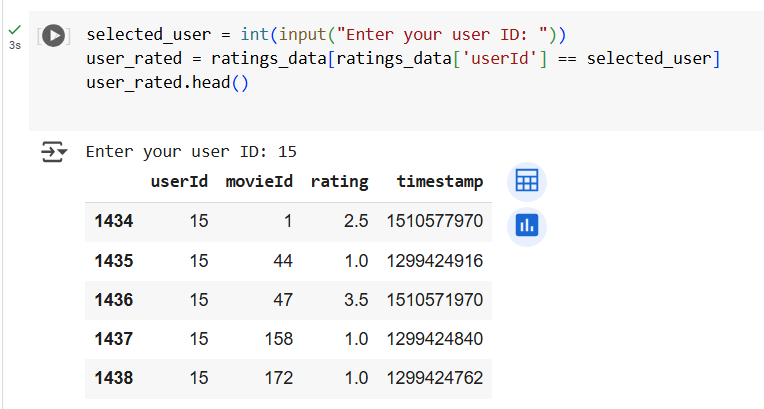
Final\_suggestions

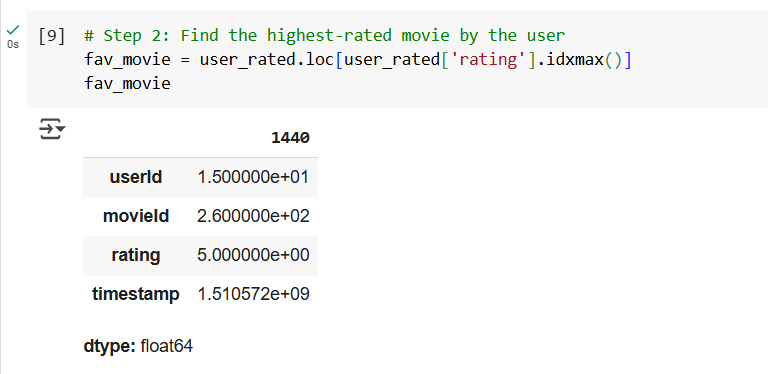
**Input’s & Output’s :**

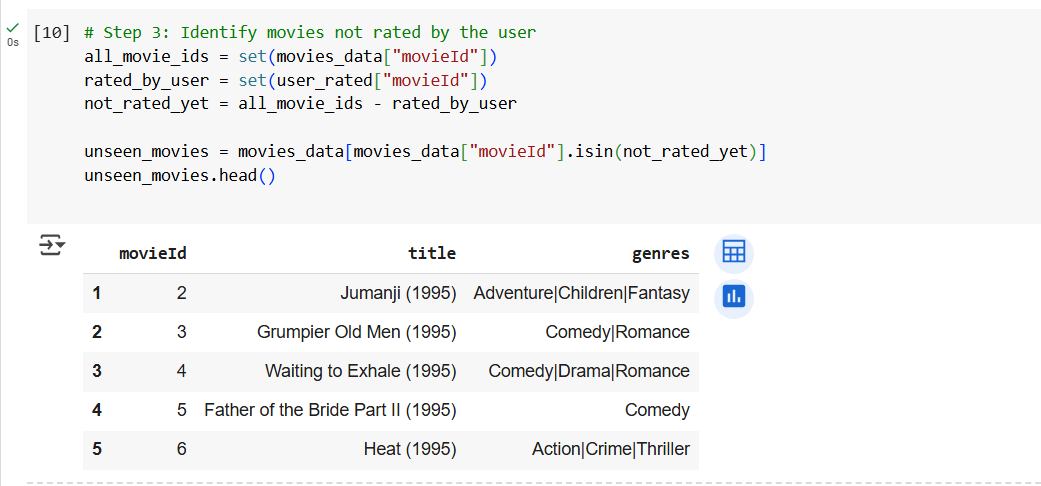


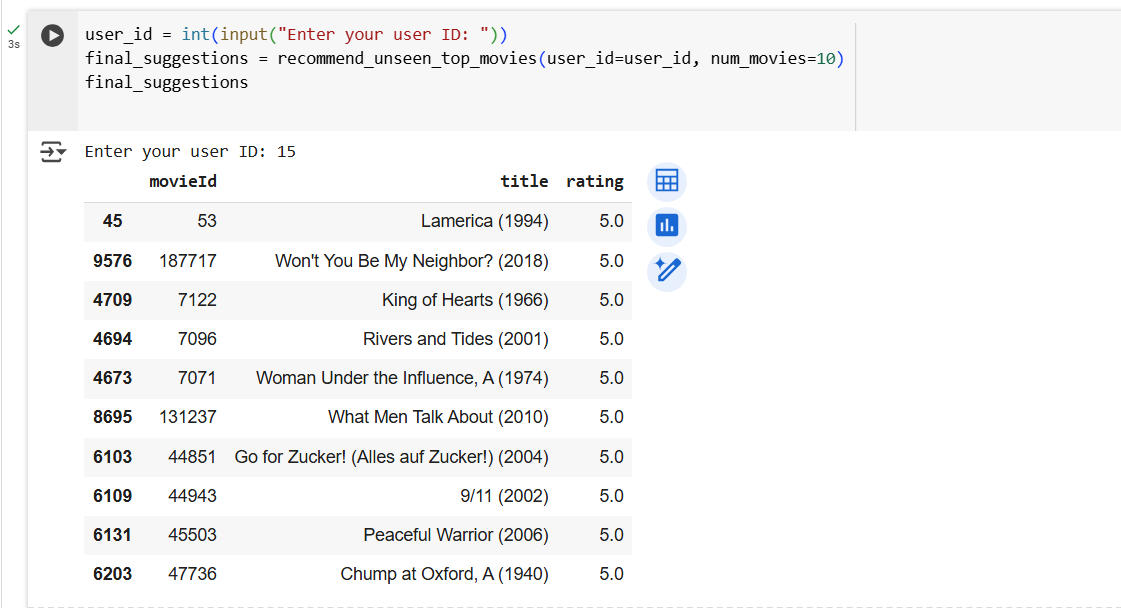












## **Results :**

* The system effectively computed a movie similarity matrix using collaborative filtering based on user ratings. This allowed it to identify films with strong correlations in viewer preferences.
* Personalized recommendations were generated by identifying each user’s highest-rated movies and suggesting similar ones they haven’t yet rated. The system ensures that suggestions are relevant and align with each user's unique taste.
* **Example**:  
  For **User 15**, the system generated the following top 10 movie recommendations that the user had not previously rated. These suggestions are based on movies that are highly rated by users with similar preferences:

| **Sl. No.** | **Movie Title** | **Rating** |
| --- | --- | --- |
| 1 | *Lamerica (1994)* | 5.0 |
| 2 | *Won't You Be My Neighbor? (2018)* | 5.0 |
| 3 | *King of Hearts (1966)* | 5.0 |
| 4 | *Rivers and Tides (2001)* | 5.0 |
| 5 | *A Woman Under the Influence (1974)* | 5.0 |
| 6 | *What Men Talk About (2010)* | 5.0 |
| 7 | *Go for Zucker! (Alles auf Zucker!) (2004)* | 5.0 |
| 8 | *9/11 (2002)* | 5.0 |
| 9 | *Peaceful Warrior (2006)* | 5.0 |
| 10 | *Chump at Oxford, A (1940)* | 5.0 |

These recommendations demonstrate the system’s ability to align movie suggestions with a user’s likely interests, even for less mainstream or internationally produced films.  
  
**Tools & Technologies :**

* Python 3
* Google Colab
* Pandas, NumPy (for data handling)
* Pearson Correlation (for similarity)

## **Conclusion :**

This project successfully developed a functional and personalized movie recommendation system using the MovieLens dataset. By applying collaborative filtering techniques and computing a similarity matrix between films, the system provides movie suggestions tailored to each user’s preferences.

The recommendations are generated by focusing on the movies a user rated most positively and suggesting similar but unseen titles with high average ratings. This approach enhances user experience by reducing the time spent searching for content and increasing the relevance of what is suggested.

The methodology proved efficient and effective, and the system lays the foundation for future improvements, such as integrating genre-based filtering, user-user similarity comparison, or advanced deep learning models for recommendation.

## **Future Improvements :**

* Integrate genre filtering to give content-aware suggestions.
* Implement user-user collaborative filtering.
* Explore model-based methods like Matrix Factorization (SVD) or Deep Learning-based recommenders.
* Add a frontend or deploy as a mini web app.