St. Francis Institute of Technology, Mumbai-400 103
Department of Information Technology

A.Y. 2024-2025
Class: TE-ITA/B, Semester: VI

Subject: **Data Science Lab**

Experiment – 8: To develop a recommendation system.

1. **Aim:** To implement Recommendation system
2. **Objectives:** After study of this experiment, the student will be able to
   - Understand how Recommendation systems are designed
   - To learn various machine learning techniques to solve complex real-world problems
3. **Outcomes:** After study of this experiment, the student will be able to
   - Analyze and apply the supervised machine learning techniques on data for building the models of data and solve the problems.
4. **Prerequisite:** Fundamentals of Python Programming and Database Management System.
5. **Requirements:** Python Installation, Personal Computer, Windows operating system, Internet Connection, Microsoft Word.
6. **Pre-Experiment Exercise:**

Brief Theory:

- What is a Recommendation Engine?**.**
- Different types of Recommendation in machine learning : Collaborative based and content based filtering

Laboratory Exercise

A. **Procedure:** Commands related to the experiments

B. Paste Screenshots of above commands.

8. Post-Experiments Exercise

A. Extended Theory: (Soft Copy)

Explain in detail the Problem statement and methodology used to perform recommendation in the experiment

B. Questions:

1. Explain in short cosine similarity used for recommendation
2. Explain TFIDF (Term Frequency Inverse document frequency) similarity matrix

C. Conclusion:

Write the significance of the topic studied in the experiment.

D. References:

1. https://www.analyticsvidhya.com/blog/2018/06/comprehensive-guide-recommendation-engine-python/
2. https://www.analyticsvidhya.com/blog/2021/08/developing-a-course-recommender-system-using-python/

Name: Vishal Rajesh Mahajan AIDS EXP 07
Class: TE IT A Roll No: 56

A. Extended Theory:

Explain in detail the Problem statement and methodology used to perform recommendation in the experiment:

Problem Statement

The goal of this experiment is to develop a recommendation system that can intelligently suggest items to users based on their preferences and past behaviors. In a world with overwhelming choices—whether it's products on Amazon, shows on Netflix, or songs on Spotify—users need assistance discovering relevant content. This problem revolves around using machine learning techniques to automate and personalize this process.

The core challenge lies in analyzing the available data to identify patterns in user behavior. Once these patterns are understood, the system must generate personalized suggestions for each user. The problem may involve handling large datasets, dealing with missing values, and choosing the appropriate recommendation strategy—collaborative filtering or content-based filtering—based on the available data and context.

Methodology

To solve the above problem, the methodology begins with data preprocessing. This includes steps such as removing duplicates, handling null values, converting categorical variables into numerical format, and normalizing the dataset. If the data includes textual content (like product descriptions), TF-IDF (Term Frequency–Inverse Document Frequency) is applied to convert it into numerical vectors for analysis.

The next step is similarity computation. In content-based filtering, the system calculates the similarity between items using measures like cosine similarity—which evaluates how close two items are in terms of their features. In collaborative filtering, the focus shifts to understanding the relationships between users or between items based on interaction history, using a user-item matrix.

Once similarities are computed, the system identifies and recommends items that are most relevant. A content-based system will suggest items that are similar to those the user already likes, while a collaborative system will suggest items liked by other users with similar tastes.

After generating the recommendations, the system is evaluated. If actual user ratings or preferences are available, metrics like Root Mean Squared Error (RMSE), precision, and recall are used to assess how accurate and useful the recommendations are. These metrics help in validating and improving the performance of the model.

This step-by-step approach ensures that students understand the complete pipeline of building and evaluating a recommendation system using supervised and unsupervised machine learning techniques.

In [6]:

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.metrics.pairwise import cosine_similarity
from sklearn.feature_extraction.text import TfidfVectorizer
from scipy.sparse.linalg import svds

# Load datasets
movies = pd.read_csv("movies.csv")
ratings = pd.read_csv("ratings.csv")

# Preview datasets
print(movies.head())
print(ratings.head())
```

```
   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)

                                        genres
0  Adventure|Animation|Children|Comedy|Fantasy
1                   Adventure|Children|Fantasy
2                               Comedy|Romance
3                         Comedy|Drama|Romance
4                                       Comedy
   userId  movieId  rating   timestamp
0       1      110     1.0  1425941529
1       1      147     4.5  1425942435
2       1      858     5.0  1425941523
3       1     1221     5.0  1425941546
4       1     1246     5.0  1425941556
```

In [7]:

```
# Merge datasets on movieId
df = pd.merge(ratings, movies, on="movieId")

# Check for missing values
print(df.isnull().sum())

# Drop unnecessary columns
df.drop(["timestamp"], axis=1, inplace=True)

# Convert ratings into user-item matrix
user_movie_ratings = df.pivot_table(index="userId", columns="title", values="rating")

# Fill missing values with 0
user_movie_ratings = user_movie_ratings.fillna(0)

print(user_movie_ratings.head())
```

```
userId       0
movieId      0
rating       0
timestamp    0
title        0
genres       0
dtype: int64
title   "Great Performances" Cats (1998)  \
userId
1                                    0.0
2                                    0.0
3                                    0.0
4                                    0.0
5                                    0.0

title   #chicagoGirl: The Social Network Takes on a Dictator (2013)  \
userId
1                                                     0.0
2                                                     0.0
3                                                     0.0
4                                                     0.0
5                                                     0.0

title   $ (Dollars) (1971)  $5 a Day (2008)  $9.99 (2008)  \
userId
1                      0.0              0.0           0.0
2                      0.0              0.0           0.0
3                      0.0              0.0           0.0
4                      0.0              0.0           0.0
5                      0.0              0.0           0.0

title   (500) Days of Summer (2009)  (Untitled) (2009)  \
userId
1                               0.0                0.0
2                               0.0                0.0
3                               0.0                0.0
4                               0.0                0.0
5                               0.0                0.0

title   *batteries not included (1987)  \
userId
1                                  0.0
2                                  0.0
3                                  0.0
4                                  0.0
5                                  0.0

title   ...All the Marbles (California Dolls, The) (1981)  \
userId
1                                                     0.0
2                                                     0.0
3                                                     0.0
4                                                     0.0
5                                                     0.0

title   ...And God Spoke (1993)  ...  \
userId                           ...
1                           0.0  ...
2                           0.0  ...
3                           0.0  ...
4                           0.0  ...
5                           0.0  ...

title   loudQUIETloud: A Film About the Pixies (2006)  night Mother (1986)  \
userId
1                                                 0.0                  0.0
2                                                 0.0                  0.0
3                                                 0.0                  0.0
```

```
4                                                 0.0                 0.0
5                                                 0.0                 0.0

title   xXx (2002)  xXx: State of the Union (2005)  ¡Three Amigos! (1986)  \
userId
1              0.0                             0.0                    0.0
2              0.0                             0.0                    0.0
3              0.0                             0.0                    0.0
4              0.0                             0.0                    0.0
5              0.0                             0.0                    0.0

title   À nos amours (1983)  À nous la liberté (Freedom for Us) (1931)  \
userId
1                       0.0                                        0.0
2                       0.0                                        0.0
3                       0.0                                        0.0
4                       0.0                                        0.0
5                       0.0                                        0.0

title   À propos de Nice (1930)  Åsa-Nisse - Wälkom to Knohult (2011)  \
userId
1                           0.0                                   0.0
2                           0.0                                   0.0
3                           0.0                                   0.0
4                           0.0                                   0.0
5                           0.0                                   0.0

title   貞子3D (2012)
userId
1               0.0
2               0.0
3               0.0
4               0.0
5               0.0

[5 rows x 17727 columns]
```

Approach 1: Content-Based Filtering Using TF-IDF

In [3]:

```python
tfidf = TfidfVectorizer(stop_words="english")

# Replace NaN with empty string
movies["genres"] = movies["genres"].fillna("")

# Compute TF-IDF matrix
tfidf_matrix = tfidf.fit_transform(movies["genres"])

# Compute similarity scores
cosine_sim = cosine_similarity(tfidf_matrix, tfidf_matrix)

# Create a mapping of movie titles to index
movie_indices = pd.Series(movies.index, index=movies["title"]).drop_duplicates()

# Function to get recommendations based on content similarity
def recommend_movies_content_based(movie_title, cosine_sim=cosine_sim):
    idx = movie_indices[movie_title]
    sim_scores = list(enumerate(cosine_sim[idx]))
    sim_scores = sorted(sim_scores, key=lambda x: x[1], reverse=True)
    sim_scores = sim_scores[1:11]  # Get top 10 similar movies
    movie_indices_top = [i[0] for i in sim_scores]
    return movies["title"].iloc[movie_indices_top]

# Example usage
print(recommend_movies_content_based("Toy Story (1995)"))
```

```
2209                                              Antz (1998)
3027                                       Toy Story 2 (1999)
3663        Adventures of Rocky and Bullwinkle, The (2000)
3922                         Emperor's New Groove, The (2000)
4790                                  Monsters, Inc. (2001)
10114    DuckTales: The Movie - Treasure of the Lost La...
10987                                        Wild, The (2006)
11871                                  Shrek the Third (2007)
13337                        Tale of Despereaux, The (2008)
18274    Asterix and the Vikings (Astérix et les Viking...
Name: title, dtype: object
```

Approach 2: Collaborative Filtering Using Matrix Factorization (SVD)

In [8]:

```python
# Convert ratings to numpy array
ratings_matrix = user_movie_ratings.values

# Normalize by subtracting mean rating of each user
user_ratings_mean = np.mean(ratings_matrix, axis=1)
ratings_demeaned = ratings_matrix - user_ratings_mean.reshape(-1, 1)

# Apply Singular Value Decomposition (SVD)
U, sigma, Vt = svds(ratings_demeaned, k=50)
sigma = np.diag(sigma)

# Reconstruct predicted ratings
predicted_ratings = np.dot(np.dot(U, sigma), Vt) + user_ratings_mean.reshape(-1, 1)

# Convert back to DataFrame
predicted_ratings_df = pd.DataFrame(predicted_ratings, index=user_movie_ratings.index, columns=

# Function to get top movie recommendations
def recommend_movies_collaborative(user_id, num_recommendations=5):
    sorted_user_ratings = predicted_ratings_df.loc[user_id].sort_values(ascending=False)
    return sorted_user_ratings.head(num_recommendations)

# Example usage
print(recommend_movies_collaborative(user_id=1))
```

```
title
Godfather, The (1972)             4.438614
Godfather: Part II, The (1974)    3.310644
Fight Club (1999)                 3.118563
Dark Knight, The (2008)           2.360485
Inception (2010)                  1.940191
Name: 1, dtype: float64
```