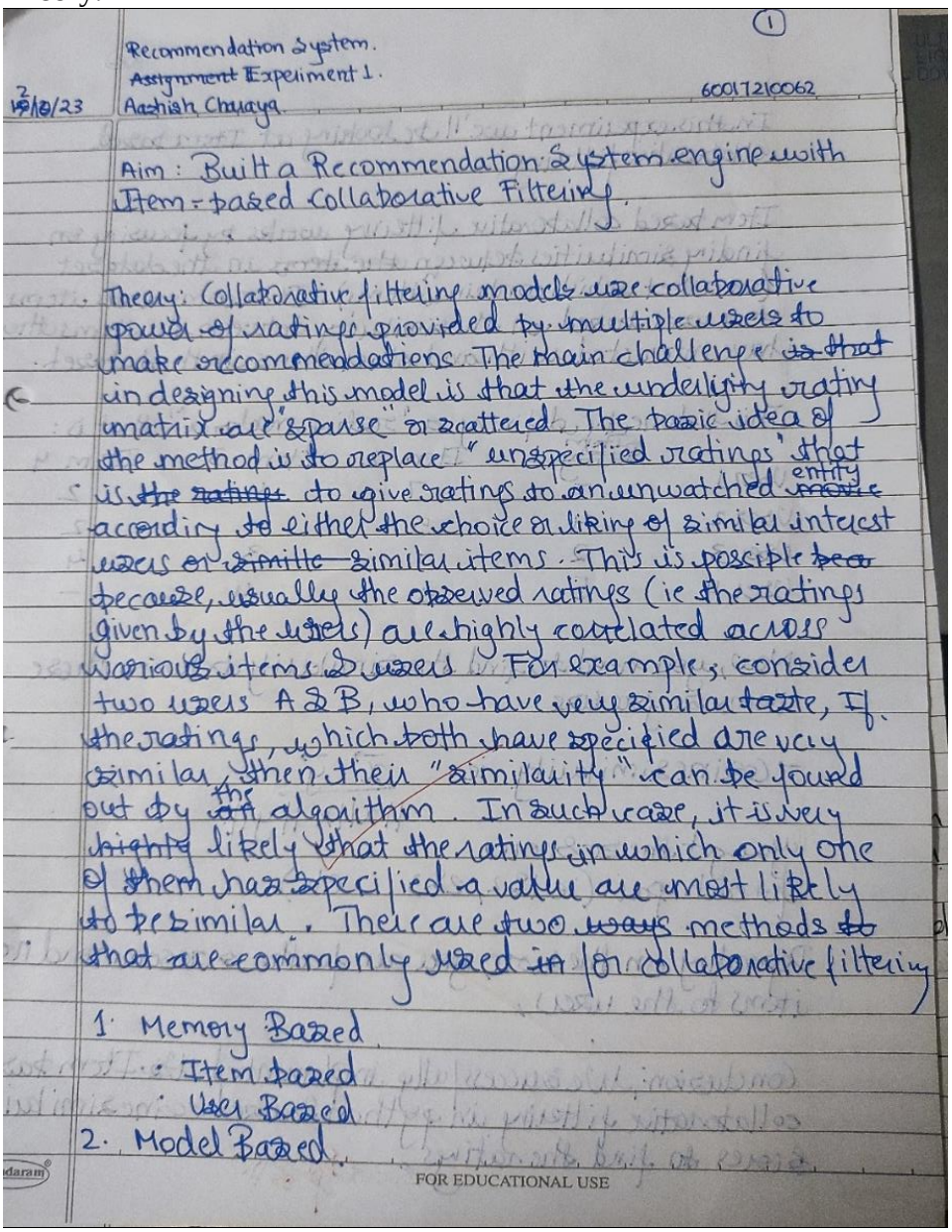
Recommendation System

Experiment 1

Aashish Charaya 60017210062 AIML

Aim: Build an item based collaborative filtering recommendation system for different datasets.

Theory:



Recommendation System. (1)

~~Assignment~~ Experiment 1.

2/10/23 Aashish Charaya 60017210062

Aim: Built a Recommendation System engine with Item-based Collaborative Filtering.

Theory: Collaborative filtering models use collaborative power of ratings provided by multiple users to make recommendations. The main challenge ~~is that~~ in designing this model is that the underlying rating matrix are "sparse" or scattered. The basic idea of the method is to replace "unspecified ratings" that is, ~~the ratings~~ to give ratings to an unwatched entry according to either the choice or liking of similar interest users or ~~similar~~ similar items. This is possible ~~because~~ because, usually the observed ratings (ie the ratings given by the users) are highly correlated across various items & users. For example, consider two users A & B, who have very similar taste, If the ratings, which both have specified are very similar, then their "similarity" can be found out by the algorithm. In such case, it is very highly likely that the ratings in which only one of them has specified a value are most likely to be similar. There are two ways methods ~~to~~ that are commonly used in for collaborative filtering

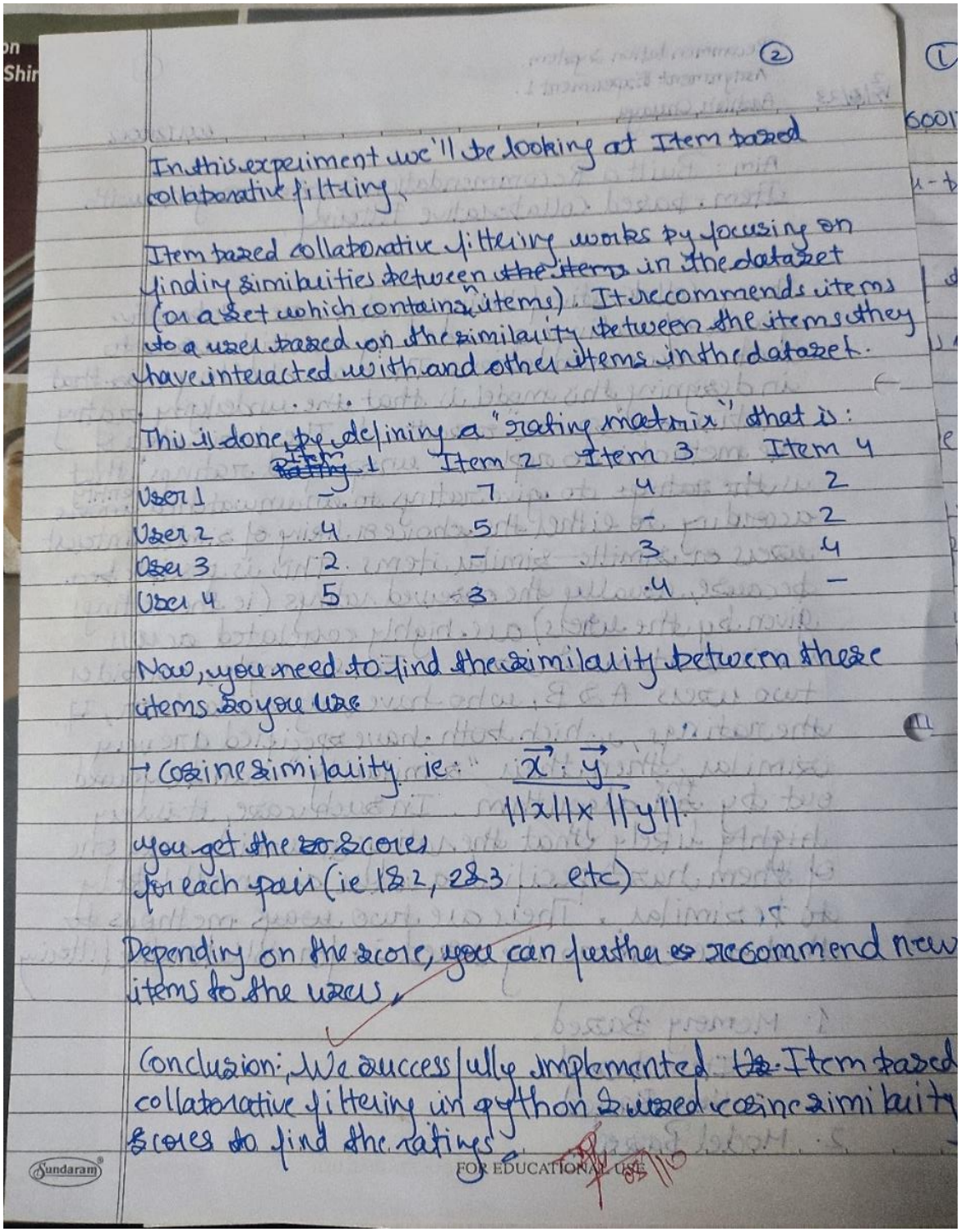
1. Memory Based
   - Item based
   - User Based
2. Model Based

FOR EDUCATIONAL USE

Recommendation System

Experiment 1

Aashish Charaya 60017210062 AIML



In this experiment we'll be looking at Item based collaborative filtering.

Item based collaborative filtering works by focusing on finding similarities between the items in the dataset (or a set which contains items). It recommends items to a user based on the similarity between the items they have interacted with and other items in the dataset.

This is done by defining a "rating matrix" that is:

Item / User	Item 1	Item 2	Item 3	Item 4
User 1		7	4	2
User 2	4	5	-	2
User 3	2	-	3	4
User 4	5	3	4	-

Now, you need to find the similarity between these items so you use

→ Cosine similarity ie: $\frac{\vec{x} \cdot \vec{y}}{\|x\| \times \|y\|}$

you get the scores for each pair (ie 1&2, 2&3 ... etc)

Depending on the score, you can further recommend new items to the users.

Conclusion: We successfully implemented Item based collaborative filtering in python & used cosine similarity scores to find the ratings.

Recommendation System

Experiment 1

Aashish Charaya 60017210062 AIML

Expt no.1: Item-Based Collaborative Filtering

```
# Data processing
import pandas as pd
import numpy as np
import scipy.stats
# Visualization
import seaborn as sns
# Similarity
from sklearn.metrics.pairwise import cosine_similarity

# Mount Google Drive
from google.colab import drive
drive.mount('/content/drive')
# Change directory
import os
os.chdir("/content/drive/MyDrive/recommendation_system")
# Print out the current directory
!pwd
```

```
Drive already mounted at /content/drive; to attempt to forcibly remount, call
drive.mount("/content/drive", force_remount=True).
/content/drive/MyDrive/recommendation_system
```

```
# Read in data
ratings=pd.read_csv('ml-latest-small/ratings.csv')
# Take a look at the data
ratings.head()
```

```
   userId  movieId  rating  timestamp
0       1        1     4.0  964982703
1       1        3     4.0  964981247
2       1        6     4.0  964982224
3       1       47     5.0  964983815
4       1       50     5.0  964982931
```

```
# Get the dataset information
ratings.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 100836 entries, 0 to 100835
Data columns (total 4 columns):
 #   Column     Non-Null Count   Dtype
---  ------     --------------   -----
 0   userId     100836 non-null  int64
 1   movieId    100836 non-null  int64
 2   rating     100836 non-null  float64
 3   timestamp  100836 non-null  int64
```

```
dtypes: float64(1), int64(3)
memory usage: 3.1 MB
```

```
# Number of users
print('The ratings dataset has', ratings['userId'].nunique(), 'unique users')
# Number of movies
print('The ratings dataset has', ratings['movieId'].nunique(), 'unique
movies')
# Number of ratings
print('The ratings dataset has', ratings['rating'].nunique(), 'unique
ratings')
# List of unique ratings
print('The unique ratings are', sorted(ratings['rating'].unique()))
```

```
The ratings dataset has 610 unique users
The ratings dataset has 9724 unique movies
The ratings dataset has 10 unique ratings
The unique ratings are [0.5, 1.0, 1.5, 2.0, 2.5, 3.0, 3.5, 4.0, 4.5, 5.0]
```

```
# Read in data
movies = pd.read_csv('ml-latest-small/movies.csv')
# Take a look at the data
movies.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
```

```
# Merge ratings and movies datasets
df = pd.merge(ratings, movies, on='movieId', how='inner')
# Take a look at the data
df.head()
```

```
   userId  movieId  rating   timestamp            title  \
0       1        1     4.0   964982703  Toy Story (1995)
```

Recommendation System

Experiment 1

Aashish Charaya 60017210062 AIML

```
1       5        1     4.0   847434962  Toy Story (1995)
2       7        1     4.5  1106635946  Toy Story (1995)
3      15        1     2.5  1510577970  Toy Story (1995)
4      17        1     4.5  1305696483  Toy Story (1995)
```

```
                                        genres
0  Adventure|Animation|Children|Comedy|Fantasy
1  Adventure|Animation|Children|Comedy|Fantasy
2  Adventure|Animation|Children|Comedy|Fantasy
3  Adventure|Animation|Children|Comedy|Fantasy
4  Adventure|Animation|Children|Comedy|Fantasy
```

```
# Aggregate by movie
agg_ratings = df.groupby('title').agg(mean_rating = ('rating', 'mean'),
                                      number_of_ratings =
('rating', 'count')).reset_index()
# Keep the movies with over 100 ratings
agg_ratings_GT100 = agg_ratings[agg_ratings['number_of_ratings']>100]
# Check the information of the dataframe
agg_ratings_GT100.info()
```

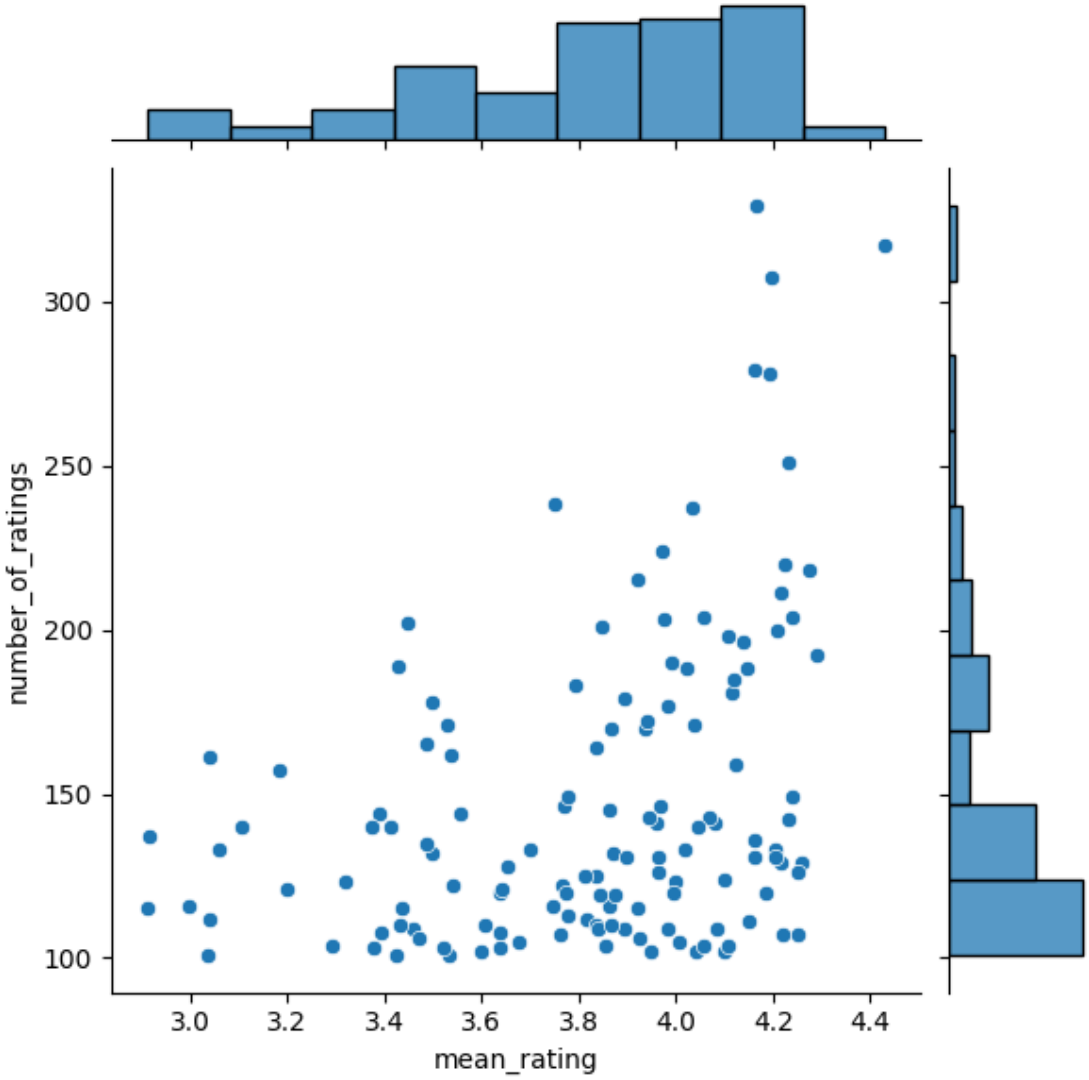
```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 134 entries, 74 to 9615
Data columns (total 3 columns):
 #   Column             Non-Null Count  Dtype
---  ------             --------------  -----
 0   title              134 non-null    object
 1   mean_rating        134 non-null    float64
 2   number_of_ratings  134 non-null    int64
dtypes: float64(1), int64(1), object(1)
memory usage: 4.2+ KB
```

```
# Check popular movies
agg_ratings_GT100.sort_values(by='number_of_ratings', ascending=False).head()
```

```
                                title  mean_rating  number_of_ratings
3158               Forrest Gump (1994)     4.164134                329
7593  Shawshank Redemption, The (1994)     4.429022                317
6865               Pulp Fiction (1994)     4.197068                307
7680  Silence of the Lambs, The (1991)     4.161290                279
5512                Matrix, The (1999)     4.192446                278
```

```
# Visulization
sns.jointplot(x='mean_rating', y='number_of_ratings', data=agg_ratings_GT100)
```

```
<seaborn.axisgrid.JointGrid at 0x7c47c7a36770>
```




```
# Merge data
df_GT100 = pd.merge(df, agg_ratings_GT100[['title']], on='title',
how='inner')
df_GT100.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 19788 entries, 0 to 19787
Data columns (total 6 columns):
 #   Column     Non-Null Count  Dtype
---  ------     --------------  -----
```

```
 0   userId     19788 non-null  int64
 1   movieId    19788 non-null  int64
 2   rating     19788 non-null  float64
 3   timestamp  19788 non-null  int64
 4   title      19788 non-null  object
 5   genres     19788 non-null  object
dtypes: float64(1), int64(3), object(2)
memory usage: 1.1+ MB
```

```
# Number of users
print('The ratings dataset has', df_GT100['userId'].nunique(), 'unique
users')
# Number of movies
print('The ratings dataset has', df_GT100['movieId'].nunique(), 'unique
movies')
# Number of ratings
print('The ratings dataset has', df_GT100['rating'].nunique(), 'unique
ratings')
# List of unique ratings
print('The unique ratings are', sorted(df_GT100['rating'].unique()))
```

```
The ratings dataset has 597 unique users
The ratings dataset has 134 unique movies
The ratings dataset has 10 unique ratings
The unique ratings are [0.5, 1.0, 1.5, 2.0, 2.5, 3.0, 3.5, 4.0, 4.5, 5.0]
```

```
# Create user-item matrix
matrix = df_GT100.pivot_table(index='title', columns='userId',
values='rating')
matrix.head()
```

```
userId                             1    2    3    4    5    6    7    8    \
title
2001: A Space Odyssey (1968)       NaN  NaN  NaN  NaN  NaN  NaN  4.0  NaN
Ace Ventura: Pet Detective (1994)  NaN  NaN  NaN  NaN  3.0  3.0  NaN  NaN
Aladdin (1992)                     NaN  NaN  NaN  4.0  4.0  5.0  3.0  NaN
Alien (1979)                       4.0  NaN  NaN  NaN  NaN  NaN  NaN  NaN
Aliens (1986)                      NaN  NaN  NaN  NaN  NaN  NaN  NaN  NaN

userId                             9    10   ...  601  602  603  604  605  \
title                                        ...
2001: A Space Odyssey (1968)       NaN  NaN  ...  NaN  NaN  5.0  NaN  NaN
Ace Ventura: Pet Detective (1994)  NaN  NaN  ...  NaN  2.0  NaN  2.0  NaN
Aladdin (1992)                     NaN  4.0  ...  NaN  NaN  NaN  3.0  3.5
```

```
Alien (1979)                    NaN  NaN  ...  NaN  NaN  5.0  NaN  NaN
Aliens (1986)                   NaN  NaN  ...  NaN  NaN  4.0  NaN  NaN

userId                          606  607  608  609  610
title
2001: A Space Odyssey (1968)    5.0  NaN  3.0  NaN  4.5
Ace Ventura: Pet Detective (1994)  NaN  NaN  3.5  NaN  3.0
Aladdin (1992)                  NaN  NaN  3.0  NaN  NaN
Alien (1979)                    4.0  3.0  4.0  NaN  4.5
Aliens (1986)                   3.5  NaN  4.5  NaN  5.0

[5 rows x 597 columns]
```

```
# Normalize user-item matrix
matrix_norm = matrix.subtract(matrix.mean(axis=1), axis = 0)
matrix_norm.head()
```

```
userId                                1    2    3        4         5    \
title
2001: A Space Odyssey (1968)          NaN  NaN  NaN      NaN       NaN
Ace Ventura: Pet Detective (1994)     NaN  NaN  NaN      NaN -0.040373
Aladdin (1992)                        NaN  NaN  NaN  0.20765  0.207650
Alien (1979)                     0.030822  NaN  NaN      NaN       NaN
Aliens (1986)                         NaN  NaN  NaN      NaN       NaN

userId                                6         7    8    9        10   ...
\
title                                                                   ...
2001: A Space Odyssey (1968)          NaN  0.105505  NaN  NaN      NaN  ...
Ace Ventura: Pet Detective (1994) -0.040373       NaN  NaN  NaN      NaN  ...
Aladdin (1992)                    1.207650 -0.792350  NaN  NaN  0.20765  ...
Alien (1979)                          NaN       NaN  NaN  NaN      NaN  ...
Aliens (1986)                         NaN       NaN  NaN  NaN      NaN  ...

userId                           601       602       603       604      605
\
title
2001: A Space Odyssey (1968)     NaN       NaN  1.105505       NaN      NaN
Ace Ventura: Pet Detective (1994)  NaN -1.040373       NaN -1.040373      NaN
Aladdin (1992)                   NaN       NaN       NaN -0.792350 -0.29235
Alien (1979)                     NaN       NaN  1.030822       NaN      NaN
Aliens (1986)                    NaN       NaN  0.035714       NaN      NaN

userId                                606       607       608  609
610
```

Recommendation System

Experiment 1

Aashish Charaya 60017210062 AIML

```
title
2001: A Space Odyssey (1968)       1.105505       NaN -0.894495  NaN
0.605505
Ace Ventura: Pet Detective (1994)       NaN       NaN  0.459627  NaN -
0.040373
Aladdin (1992)                          NaN       NaN -0.792350  NaN
NaN
Alien (1979)                       0.030822 -0.969178  0.030822  NaN
0.530822
Aliens (1986)                     -0.464286       NaN  0.535714  NaN
1.035714

[5 rows x 597 columns]
```

```
# Item similarity matrix using Pearson correlation
item_similarity = matrix_norm.T.corr()
item_similarity.head()
```

```
title                              2001: A Space Odyssey (1968)  \
title
2001: A Space Odyssey (1968)                           1.000000
Ace Ventura: Pet Detective (1994)                     -0.036319
Aladdin (1992)                                         0.017446
Alien (1979)                                           0.318523
Aliens (1986)                                          0.317386

title                              Ace Ventura: Pet Detective (1994)  \
title
2001: A Space Odyssey (1968)                               -0.036319
Ace Ventura: Pet Detective (1994)                           1.000000
Aladdin (1992)                                              0.302193
Alien (1979)                                               -0.208017
Aliens (1986)                                              -0.107524

title                              Aladdin (1992)  Alien (1979)  \
title
2001: A Space Odyssey (1968)             0.017446      0.318523
Ace Ventura: Pet Detective (1994)        0.302193     -0.208017
Aladdin (1992)                           1.000000      0.026514
Alien (1979)                             0.026514      1.000000
Aliens (1986)                            0.151152      0.705925

title                              Aliens (1986)  \
title
2001: A Space Odyssey (1968)            0.317386
```

```
Ace Ventura: Pet Detective (1994)      -0.107524
Aladdin (1992)                          0.151152
Alien (1979)                            0.705925
Aliens (1986)                           1.000000

title                                Amelie (Fabuleux destin d'Amélie Poulain,
Le) (2001)  \
title
2001: A Space Odyssey (1968)
0.324150
Ace Ventura: Pet Detective (1994)                                          -
0.030425
Aladdin (1992)
0.445204
Alien (1979)
0.387215
Aliens (1986)
0.540458

title                                American Beauty (1999)  \
title
2001: A Space Odyssey (1968)                    0.193592
Ace Ventura: Pet Detective (1994)               0.040435
Aladdin (1992)                                  0.127764
Alien (1979)                                    0.215751
Aliens (1986)                                   0.111452

title                                American History X (1998)  \
title
2001: A Space Odyssey (1968)                       0.152405
Ace Ventura: Pet Detective (1994)                  0.065549
Aladdin (1992)                                     0.262014
Alien (1979)                                       0.035373
Aliens (1986)                                      0.139326

title                                American Pie (1999)  Apocalypse Now (1979)
\
title
2001: A Space Odyssey (1968)                  0.011490              0.478877
Ace Ventura: Pet Detective (1994)             0.173855              0.245829
Aladdin (1992)                                0.367076              0.015038
Alien (1979)                                 -0.006804              0.378709
Aliens (1986)                                 0.076674              0.221920

title                                ...  True Lies (1994)  \
title                                ...
2001: A Space Odyssey (1968)         ...         -0.108291
Ace Ventura: Pet Detective (1994)    ...          0.139896
```

Recommendation System

Experiment 1

Aashish Charaya 60017210062 AIML

```
Aladdin (1992)                    ...          0.333687
Alien (1979)                      ...          0.199538
Aliens (1986)                     ...          0.369971

title                             Truman Show, The (1998)  \
title
2001: A Space Odyssey (1968)                    -0.012451
Ace Ventura: Pet Detective (1994)                0.188089
Aladdin (1992)                                   0.562311
Alien (1979)                                     0.178620
Aliens (1986)                                    0.287243

title                             Twelve Monkeys (a.k.a. 12 Monkeys) (1995)
\
title
2001: A Space Odyssey (1968)                                      -0.041791
Ace Ventura: Pet Detective (1994)                                  0.054408
Aladdin (1992)                                                    -0.069176
Alien (1979)                                                       0.108327
Aliens (1986)                                                      0.084792

title                             Twister (1996)  Up (2009)  \
title
2001: A Space Odyssey (1968)           -0.458642   0.152271
Ace Ventura: Pet Detective (1994)       0.176930  -0.007853
Aladdin (1992)                          0.137215   0.171330
Alien (1979)                            0.022007  -0.098813
Aliens (1986)                           0.092412   0.195581

title                             Usual Suspects, The (1995)  WALL·E (2008)
\
title
2001: A Space Odyssey (1968)                        0.245279       0.100172
Ace Ventura: Pet Detective (1994)                  -0.061520       0.170717
Aladdin (1992)                                      0.153934       0.272375
Alien (1979)                                        0.350428       0.270697
Aliens (1986)                                       0.296933       0.294852

title                             Waterworld (1995)  \
title
2001: A Space Odyssey (1968)              -0.447306
Ace Ventura: Pet Detective (1994)          0.176155
Aladdin (1992)                             0.065342
Alien (1979)                               0.119849
Aliens (1986)                             -0.014274

title                             Willy Wonka & the Chocolate Factory (1971)
\
```

Recommendation System

Experiment 1

Aashish Charaya 60017210062 AIML

```
title
2001: A Space Odyssey (1968)                                    0.087803
Ace Ventura: Pet Detective (1994)                               0.051239
Aladdin (1992)                                                  0.164459
Alien (1979)                                                    0.117749
Aliens (1986)                                                   0.111864

title                                 X-Men (2000)
title
2001: A Space Odyssey (1968)          -0.123862
Ace Ventura: Pet Detective (1994)      0.045676
Aladdin (1992)                         0.285480
Alien (1979)                           0.030257
Aliens (1986)                          0.225923

[5 rows x 134 columns]
```

```
# Pick a user ID
picked_userid = 1
# Pick a movie
picked_movie = 'American Pie (1999)'
# Movies that the target user has watched
picked_userid_watched =
pd.DataFrame(matrix_norm[picked_userid].dropna(axis=0, how='all')\
                          .sort_values(ascending=False))\
                          .reset_index()\
                          .rename(columns={1:'rating'})
picked_userid_watched.head()
```

```
                                        title    rating
0       Dumb & Dumber (Dumb and Dumber) (1994)  1.939850
1  Indiana Jones and the Temple of Doom (1984)  1.361111
2                                X-Men (2000)  1.300752
3           E.T. the Extra-Terrestrial (1982)  1.233607
4   Ghostbusters (a.k.a. Ghost Busters) (1984)  1.225000
```

```
# Similarity score of the movie American Pie with all the other movies
picked_movie_similarity_score =
item_similarity[[picked_movie]].reset_index().rename(columns={'American Pie
(1999)':'similarity_score'})
# Rank the similarities between the movies user 1 rated and American Pie.
n = 5
picked_userid_watched_similarity = pd.merge(left=picked_userid_watched,

right=picked_movie_similarity_score,
                                            on='title',
```

```
                                                  how='inner')\
                                        .sort_values('similarity_score',
ascending=False)[:5]
# Take a look at the User 1 watched movies with highest similarity
picked_userid_watched_similarity
```

```
                                              title    rating  \
52                       Mission: Impossible (1996) -0.537037
47                                 Twister (1996) -0.321138
16  Star Wars: Episode I - The Phantom Menace (1999)  0.892857
10                              Fugitive, The (1993)  1.007895
19                            Green Mile, The (1999)  0.851351

    similarity_score
52          0.510888
47          0.476518
16          0.443614
10          0.442128
19          0.429560
```

```
# Calculate the predicted rating using weighted average of similarity scores
and the ratings from user 1
predicted_rating =
round(np.average(picked_userid_watched_similarity['rating'],

weights=picked_userid_watched_similarity['similarity_score']), 6)
print(f'The predicted rating for {picked_movie} by user {picked_userid} is
{predicted_rating}' )
```

```
The predicted rating for American Pie (1999) by user 1 is 0.338739
```

```
# Item-based recommendation function
def item_based_rec(picked_userid=1, number_of_similar_items=5,
number_of_recommendations =3):
  import operator
  # Movies that the target user has not watched
  picked_userid_unwatched =
pd.DataFrame(matrix_norm[picked_userid].isna()).reset_index()
  picked_userid_unwatched =
picked_userid_unwatched[picked_userid_unwatched[1]==True]['title'].values.tol
ist()
  # Movies that the target user has watched
  picked_userid_watched =
pd.DataFrame(matrix_norm[picked_userid].dropna(axis=0, how='all')\
                            .sort_values(ascending=False))\
                            .reset_index()\
```

```
                .rename(columns={1:'rating'})

  # Dictionary to save the unwatched movie and predicted rating pair
  rating_prediction ={}
  # Loop through unwatched movies
  for picked_movie in picked_userid_unwatched:
    # Calculate the similarity score of the picked movie iwth other movies
    picked_movie_similarity_score =
item_similarity[[picked_movie]].reset_index().rename(columns={picked_movie:'s
imilarity_score'})
    # Rank the similarities between the picked user watched movie and the
picked unwatched movie.
    picked_userid_watched_similarity = pd.merge(left=picked_userid_watched,

right=picked_movie_similarity_score,
                                                on='title',
                                                how='inner')\
                                        .sort_values('similarity_score',
ascending=False)[:number_of_similar_items]
    # Calculate the predicted rating using weighted average of similarity
scores and the ratings from user 1
    predicted_rating =
round(np.average(picked_userid_watched_similarity['rating'],

weights=picked_userid_watched_similarity['similarity_score']), 6)
    # Save the predicted rating in the dictionary
    rating_prediction[picked_movie] = predicted_rating
    # Return the top recommended movies
  return sorted(rating_prediction.items(), key=operator.itemgetter(1),
reverse=True)[:number_of_recommendations]
# Get recommendations
recommended_movie = item_based_rec(picked_userid=1,
number_of_similar_items=5, number_of_recommendations =3)
recommended_movie
```

```
[('Austin Powers: The Spy Who Shagged Me (1999)', 1.096288),
 ('Crouching Tiger, Hidden Dragon (Wo hu cang long) (2000)', 0.92924),
 ('Lord of the Rings: The Return of the King, The (2003)', 0.926824)]
```

GitHub Repo : https://github.com/Aashish-Charaya/RS_practicals/tree/main

Conclusion: Implemented an Item-based Collaborative filtering recommendation engine on different datasets.