MOVIE RECOMMENDATION SYSTEM

[49]:
```
import numpy as np
import pandas as pd
```

[50]:
```
movies_df=pd.read_csv('/content/movies.csv')
ratings_df=pd.read_csv('/content/ratings.csv')
movies_df.head()
```

[50]:
```
   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
```

[51]:
```
movies_df.shape
```

[51]:
```
(62423, 3)
```

[52]:
```
movies_df.isna().sum()
```

[52]:
```
movieId    0
title      0
genres     0
dtype: int64
```

[53]:
```
movies_df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 62423 entries, 0 to 62422
Data columns (total 3 columns):
 #   Column   Non-Null Count  Dtype
```

```
---  ------   --------------  -----
 0   movieId  62423 non-null  int64
 1   title    62423 non-null  object
 2   genres   62423 non-null  object
dtypes: int64(1), object(2)
memory usage: 1.4+ MB
```

[54]:
```
ratings_df.head()
```

[54]:
```
   userId  movieId  rating   timestamp
0       1      296     5.0  1147880044
1       1      306     3.5  1147868817
2       1      307     5.0  1147868828
3       1      665     5.0  1147878820
4       1      899     3.5  1147868510
```

[55]:
```
ratings_df.isna().sum()
```

[55]:
```
userId       0
movieId      0
rating       0
timestamp    0
dtype: int64
```

[56]:
```
ratings_df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 9992900 entries, 0 to 9992899
Data columns (total 4 columns):
 #   Column     Dtype
---  ------     -----
 0   userId     int64
 1   movieId    int64
 2   rating     float64
 3   timestamp  int64
dtypes: float64(1), int64(3)
memory usage: 305.0 MB
```

[57]:
```
ratings_df.shape
```

[57]:
```
(9992900, 4)
```

[58]:
```
df=movies_df.merge(ratings_df)
df1=df.drop(['genres', 'timestamp'], axis=1)
df1.head()
```

```
[58]:    movieId             title  userId  rating
      0        1  Toy Story (1995)       2     3.5
      1        1  Toy Story (1995)       3     4.0
      2        1  Toy Story (1995)       4     3.0
      3        1  Toy Story (1995)       5     4.0
      4        1  Toy Story (1995)       8     4.0
```

```
[59]: df2=df1.groupby('title')['rating'].count().reset_index().
       ↪rename(columns={'rating': 'ratingcount'})
      df2.head()
```

```
[59]:                                 title  ratingcount
      0  "Great Performances" Cats (1998)           70
      1        #1 Cheerleader Camp (2010)            4
      2           #Female Pleasure (2018)            1
      3                  #FollowMe (2019)            2
      4                    #Horror (2015)            8
```

```
[60]: df3=df1.merge(df2,on='title',how='left')
      df3.head()
```

```
[60]:    movieId             title  userId  rating  ratingcount
      0        1  Toy Story (1995)       2     3.5        22925
      1        1  Toy Story (1995)       3     4.0        22925
      2        1  Toy Story (1995)       4     3.0        22925
      3        1  Toy Story (1995)       5     4.0        22925
      4        1  Toy Story (1995)       8     4.0        22925
```

```
[61]: print(f'10% of count= {df3.ratingcount.quantile(.1)}')
      print(f'25% of count= {df3.ratingcount.quantile(.25)}')
      print(f'50% of count= {df3.ratingcount.quantile(.5)}')
      print(f'75% of count= {df3.ratingcount.quantile(.75)}')
      print(f'85% of count= {df3.ratingcount.quantile(.85)}')
      print(f'100% of count= {df3.ratingcount.quantile(1)}')
```

```
10% of count= 332.0
25% of count= 1184.0
50% of count= 3642.0
75% of count= 8321.0
85% of count= 12267.0
100% of count= 32581.0
```

```
[62]: df4=df3[df3.ratingcount>=10000]
      df4.shape
```

```
[62]: (1985080, 5)
```

[63]:
```
movie_rating_pivote=df4.pivot_table(index='title',columns='userId',␣
    ↪values='rating').fillna(0)
movie_rating_pivote.head()
```

[63]:
```
userId                             1      2      3      4      5      6      \
title
2001: A Space Odyssey (1968)         0.0    0.0    5.0    4.0    0.0    4.0
Ace Ventura: Pet Detective (1994)    0.0    0.0    0.0    0.0    4.0    0.0
Aladdin (1992)                       0.0    2.0    0.0    0.0    4.0    0.0
Alien (1979)                         0.0    0.0    4.0    2.5    0.0    0.0
Aliens (1986)                        0.0    0.0    4.0    3.5    0.0    0.0

userId                             7      8      9      10     …   64836  \
title                                                            …
2001: A Space Odyssey (1968)         0.0    5.0    3.0    4.5  …     0.0
Ace Ventura: Pet Detective (1994)    2.0    5.0    0.0    0.0  …     0.0
Aladdin (1992)                       4.0    5.0    5.0    3.0  …     0.0
Alien (1979)                         0.0    4.0    4.0    0.0  …     5.0
Aliens (1986)                        0.0    3.0    3.0    0.0  …     4.0

userId                             64837  64838  64839  64840  64841  64842  \
title
2001: A Space Odyssey (1968)         0.0    0.0    0.0    0.0    0.0    0.0
Ace Ventura: Pet Detective (1994)    0.0    0.0    0.0    0.0    0.0    0.0
Aladdin (1992)                       0.0    0.0    0.0    2.0    0.0    2.0
Alien (1979)                         0.0    0.0    0.0    0.0    0.0    0.0
Aliens (1986)                        0.0    4.0    0.0    0.0    0.0    0.0

userId                             64843  64844  64845
title
2001: A Space Odyssey (1968)         4.0    5.0    0.0
Ace Ventura: Pet Detective (1994)    0.0    0.0    0.0
Aladdin (1992)                       0.0    1.5    0.0
Alien (1979)                         4.0    5.0    0.0
Aliens (1986)                        4.0    4.0    0.0

[5 rows x 63853 columns]
```

[65]:
```
from scipy.sparse import csr_matrix
from sklearn.neighbors import NearestNeighbors

movie_rating_df=csr_matrix(movie_rating_pivote.values)

model_knn=NearestNeighbors(metric='cosine', algorithm='brute',n_neighbors=11)
model_knn.fit(movie_rating_df)
```

[65]:
```
NearestNeighbors(algorithm='brute', metric='cosine', n_neighbors=11)
```

[66]:
```
movie_rating_df.shape
```

[66]:
```
(130, 63853)
```

[67]:
```
distance,indices=model_knn.kneighbors(movie_rating_pivote.iloc[np.random.
  ↪choice(movie_rating_df.shape[0]),:].values.reshape(1,-1),n_neighbors=11)

for i in range(0, len(distance.flatten())):
    if i == 0:
        print(f'Recommendation for a movie: {movie_rating_pivote.index[indices.
  ↪flatten()[i]]}')
    else:
        print(f'{i}: {movie_rating_pivote.index[indices.flatten()[i]]} with a␣
  ↪distance {distance.flatten()[i]}')
```

```
Recommendation for a movie: Lord of the Rings: The Return of the King, The
(2003)
1: Lord of the Rings: The Two Towers, The (2002) with a distance
0.12316745403419926
2: Lord of the Rings: The Fellowship of the Ring, The (2001) with a distance
0.13908764954585817
3: Matrix, The (1999) with a distance 0.3408159137531891
4: Pirates of the Caribbean: The Curse of the Black Pearl (2003) with a distance
0.3488238931569979
5: Dark Knight, The (2008) with a distance 0.39843788006466696
6: Batman Begins (2005) with a distance 0.39931784431053596
7: Fight Club (1999) with a distance 0.41173312911828774
8: Shrek (2001) with a distance 0.4151812657303918
9: Gladiator (2000) with a distance 0.415449049801484
10: Finding Nemo (2003) with a distance 0.41633267703600285
```

[ ]:

[ ]:

[ ]:

[ ]: