Zee_Recommender_Systems_Case_Study

August 15, 2024

[2]:
```
!gdown 15QeQgmjoeBxRDEOFPSrMr8eIvwk6QgUQ
```

```
Downloading…
From: https://drive.google.com/uc?id=15QeQgmjoeBxRDEOFPSrMr8eIvwk6QgUQ
To: /content/zee-movies.dat
100% 171k/171k [00:00<00:00, 38.3MB/s]
```

[3]:
```
!gdown 1XJpSzv-UMeSmCCOdyZviRF4XCxiKJUrk
```

```
Downloading…
From: https://drive.google.com/uc?id=1XJpSzv-UMeSmCCOdyZviRF4XCxiKJUrk
To: /content/zee-ratings.dat
100% 24.6M/24.6M [00:00<00:00, 34.1MB/s]
```

[4]:
```
!gdown 1-xsSNizetOruiMgKLWRcmme8L_iDejhc
```

```
Downloading…
From: https://drive.google.com/uc?id=1-xsSNizetOruiMgKLWRcmme8L_iDejhc
To: /content/zee-users.dat
100% 134k/134k [00:00<00:00, 62.6MB/s]
```

1 Business Problem:

To create a **Recommender System** to show **personalized movie recommendations** based on ratings given **by a user and other users similar** to them in order **to improve user experience.**

1.0.1 Importing required packages:

[5]:
```
import pandas as pd
import numpy as np
import math as math
import keras
import seaborn as sns
sns.set(style='whitegrid')
from scipy import stats
import matplotlib.pyplot as plt
import warnings
warnings.filterwarnings("ignore")
```

```
plt.rcParams['figure.figsize'] = (15, 6)
```

2 Exploratory Data Analysis:

[7]:
```
#fixed-width formatted (fwf)
movies = pd.read_fwf('zee-movies.dat',encoding='latin-1')
ratings = pd.read_fwf('zee-ratings.dat',encoding = 'ISO-8859-1')
users = pd.read_fwf('zee-users.dat',encoding = 'ISO-8859-1')
```

2.1 Movies (EDA):

[8]:
```
movies.head()
```

[8]:
```
                            Movie ID::Title::Genres Unnamed: 1 Unnamed: 2
0  1::Toy Story (1995)::Animation|Children's|Comedy        NaN        NaN
1   2::Jumanji (1995)::Adventure|Children's|Fantasy        NaN        NaN
2        3::Grumpier Old Men (1995)::Comedy|Romance        NaN        NaN
3         4::Waiting to Exhale (1995)::Comedy|Drama        NaN        NaN
4      5::Father of the Bride Part II (1995)::Comedy        NaN        NaN
```

[9]:
```
# Feature Engineering

movies.drop(columns=['Unnamed: 1','Unnamed: 2'],inplace=True)
delimeter = "::"
movies = movies['Movie ID::Title::Genres'].str.split(delimeter,expand=True)
movies.columns = ['MovieID','Title','Genres']
```

[10]:
```
movies.head()
```

[10]:
```
  MovieID                               Title                        Genres
0       1                    Toy Story (1995)   Animation|Children's|Comedy
1       2                      Jumanji (1995)  Adventure|Children's|Fantasy
2       3             Grumpier Old Men (1995)                Comedy|Romance
3       4            Waiting to Exhale (1995)                  Comedy|Drama
4       5  Father of the Bride Part II (1995)                        Comedy
```

[11]:
```
movies.describe()
```

[11]:
```
       MovieID              Title Genres
count     3883               3883   3858
unique    3883               3883    360
top          1  Toy Story (1995)  Drama
freq         1                  1    830
```

[12]:
```
movies.shape
```

[12]: (3883, 3)

[13]:
```
movies.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 3883 entries, 0 to 3882
Data columns (total 3 columns):
 #   Column   Non-Null Count  Dtype
---  ------   --------------  -----
 0   MovieID  3883 non-null   object
 1   Title    3883 non-null   object
 2   Genres   3858 non-null   object
dtypes: object(3)
memory usage: 91.1+ KB
```

2.2 Ratings (EDA)

[14]:
```
ratings.head()
```

[14]:
```
  UserID::MovieID::Rating::Timestamp
0              1::1193::5::978300760
1               1::661::3::978302109
2               1::914::3::978301968
3              1::3408::4::978300275
4              1::2355::5::978824291
```

[15]:
```
delimeter = "::"
ratings = ratings['UserID::MovieID::Rating::Timestamp'].str.
 ↪split(delimeter,expand=True)
ratings.columns = ['UserID','MovieID','Rating','Timestamp']
```

[16]:
```
ratings.head()
```

[16]:

	UserID	MovieID	Rating	Timestamp
0	1	1193	5	978300760
1	1	661	3	978302109
2	1	914	3	978301968
3	1	3408	4	978300275
4	1	2355	5	978824291

[17]:
```
ratings.shape
```

[17]: (1000209, 4)

[18]:
```
ratings.describe()
```

[18]:

	UserID	MovieID	Rating	Timestamp
count	1000209	1000209	1000209	1000209
unique	6040	3706	5	458455
top	4169	2858	4	975528402
freq	2314	3428	348971	30

2.3 Users (EDA):

[19]:
```
users.head()
```

[19]:

	UserID::Gender::Age::Occupation::Zip-code
0	1::F::1::10::48067
1	2::M::56::16::70072
2	3::M::25::15::55117
3	4::M::45::7::02460
4	5::M::25::20::55455

[20]:
```
delimeter = "::"
users = users['UserID::Gender::Age::Occupation::Zip-code'].str.
 ↪split(delimeter,expand=True)
users.columns = ['UserID','Gender','Age','Occupation','Zip-code']
```

[21]:
```
users.head()
```

[21]:

	UserID	Gender	Age	Occupation	Zip-code
0	1	F	1	10	48067
1	2	M	56	16	70072
2	3	M	25	15	55117
3	4	M	45	7	02460
4	5	M	25	20	55455

[22]:
```
users.shape
```

[22]: (6040, 5)

[23]:
```
users.describe()
```

[23]:

	UserID	Gender	Age	Occupation	Zip-code
count	6040	6040	6040	6040	6040
unique	6040	2	7	21	3439
top	1	M	25	4	48104
freq	1	4331	2096	759	19

[23]:
```

```

2.4 Merging Datasets(movies, ratings and users)

[24]:
```
df = pd.merge(pd.merge(movies,ratings,left_on =␣
↪'MovieID',right_on='MovieID',how='inner'),users,on='UserID',how='inner')
```

[25]:
```
df.head()
```

[25]:
```
  MovieID                                    Title  \
0       1                         Toy Story (1995)
1      48                        Pocahontas (1995)
2     150                         Apollo 13 (1995)
3     260  Star Wars: Episode IV - A New Hope (1977)
4     527                  Schindler's List (1993)

                          Genres UserID Rating  Timestamp Gender Age  \
0            Animation|Children's|Comedy      1      5  978824268      F   1
1  Animation|Children's|Musical|Romance      1      5  978824351      F   1
2                                  Drama      1      5  978301777      F   1
3                Action|Adventure|Fantas      1      4  978300760      F   1
4                              Drama|War      1      5  978824195      F   1

  Occupation Zip-code
0         10    48067
1         10    48067
2         10    48067
3         10    48067
4         10    48067
```

[26]:
```
df.shape
```

[26]:
```
(1000209, 10)
```

[27]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1000209 entries, 0 to 1000208
Data columns (total 10 columns):
 #   Column      Non-Null Count    Dtype
---  ------      --------------    -----
 0   MovieID     1000209 non-null  object
 1   Title       1000209 non-null  object
 2   Genres      996144 non-null   object
 3   UserID      1000209 non-null  object
 4   Rating      1000209 non-null  object
 5   Timestamp   1000209 non-null  object
 6   Gender      1000209 non-null  object
 7   Age         1000209 non-null  object
 8   Occupation  1000209 non-null  object
```

```
 9   Zip-code    1000209 non-null  object
dtypes: object(10)
memory usage: 76.3+ MB
```

[27]:

```

```

2.5 Feature Engineering:

[28]:

```
df['Age'] = df['Age'].astype('int32')
df['Rating'] = df['Rating'].astype('int32')
df['Timestamp'] = pd.to_datetime(df['Timestamp'],unit='s')
```

[36]:

```
df['ReleaseYear'] = df['Title'].str.rsplit(' ', n=1).str[1].str.lstrip("(").str.
↪rstrip(")")
```

[32]:

```
df.replace({'Age':{1: "Under 18",
                   18: "18-24",
                   25: "25-34",
                   35: "35-44",
                   45: "45-49",
                   50: "50-55",
                   56: "56+"}},inplace=True)
```

[33]:

```
df.replace({'Occupation':{'0': "other" or not specified,
                          '1': "academic/educator",
                          '2': "artist",
                          '3': "clerical/admin",
                          '4': "college/grad student",
                          '5': "customer service",
                          '6': "doctor/health care",
                          '7': "executive/managerial",
                          '8': "farmer",
                          '9': "homemaker",
                          '10': "K-12 student",
                          '11': "lawyer",
                          '12': "programmer",
                          '13': "retired",
                          '14': "sales/marketing",
                          '15': "scientist",
                          '16': "self-employed",
                          '17': "technician/engineer",
                          '18': "tradesman/craftsman",
                          '19': "unemployed",
                          '20': "writer"

}},inplace=True)
```

[34]:
```python
df.head()
```

[34]:
```
   MovieID                                     Title  \
0        1                          Toy Story (1995)
1       48                        Pocahontas (1995)
2      150                         Apollo 13 (1995)
3      260  Star Wars: Episode IV - A New Hope (1977)
4      527                  Schindler's List (1993)

                                 Genres UserID  Rating           Timestamp  \
0           Animation|Children's|Comedy      1       5 2001-01-06 23:37:36
1  Animation|Children's|Musical|Romance      1       5 2001-01-06 23:38:40
2                                 Drama      1       5 2000-12-31 22:29:20
3               Action|Adventure|Fantas      1       4 2000-12-31 22:12:16
4                             Drama|War      1       5 2001-01-06 23:36:32

  Gender       Age    Occupation Zip-code
0      F  Under 18  K-12 student    48067
1      F  Under 18  K-12 student    48067
2      F  Under 18  K-12 student    48067
3      F  Under 18  K-12 student    48067
4      F  Under 18  K-12 student    48067
```

[37]:
```python
df['ReleaseYear'].unique()
```

[37]:
```
array(['1995', '1977', '1993', '1992', '1937', '1991', '1996', '1964',
       '1939', '1958', '1950', '1941', '1965', '1982', '1975', '1987',
       '1962', '1989', '1985', '1959', '1997', '1998', '1988', '1942',
       '1947', '1999', '1980', '1983', '1986', '1990', '2000', '1964):',
       '1994', '1978', '1961', '1984', '1972', '1976', '1981', '1973',
       '1974', '1940', 'Bo', '1952', '1954', '1953', '1944', '1968',
       '1957', '1946', '1949', '1951', '1963', '1971', '1979', '1967',
       '1966', '1948', '1933', '1970', '1969', '1930', '1955', '1956', '',
       '1920', '1925', '1938', '195', '1960', '1935', '1932', '1931',
       '1945', '1943', '1981):', '1934', '1936', '1929', 'the', '1926',
       'Arta', 'B', '1927', '19', '1922', 'Polar', '1919', '1921', "d'A",
       '1923', '1989):', '1928', '1995):', 'prendront', '1'], dtype=object)
```

[38]:
```python
df['ReleaseYear'].nunique()
```

[38]: 96

[39]:
```python
df['ReleaseYear'].replace(['1981):','1964):','1989):','1995):
↪'],['1981','1964','1989','1995'],inplace=True)
```

[40]:
```python
idx_val = df[(df['ReleaseYear'] == 'prendront') |
   (df['ReleaseYear'] == 'Polar')|
```

```
    (df['ReleaseYear'] == 'Bo')|
    (df['ReleaseYear'] == 'Arta')|
    (df['ReleaseYear'] == 'B')|
    (df['ReleaseYear'] == "d'A")|
    (df['ReleaseYear'] == '19')|
    (df['ReleaseYear'] == '')|
    (df['ReleaseYear'] == 'the')|
    (df['ReleaseYear'] == '195')].index
df.drop(index=idx_val,inplace=True)
```

[42]:
```
df['Title'] = df['Title'].str.rsplit(' ',n=1).str[0]
```

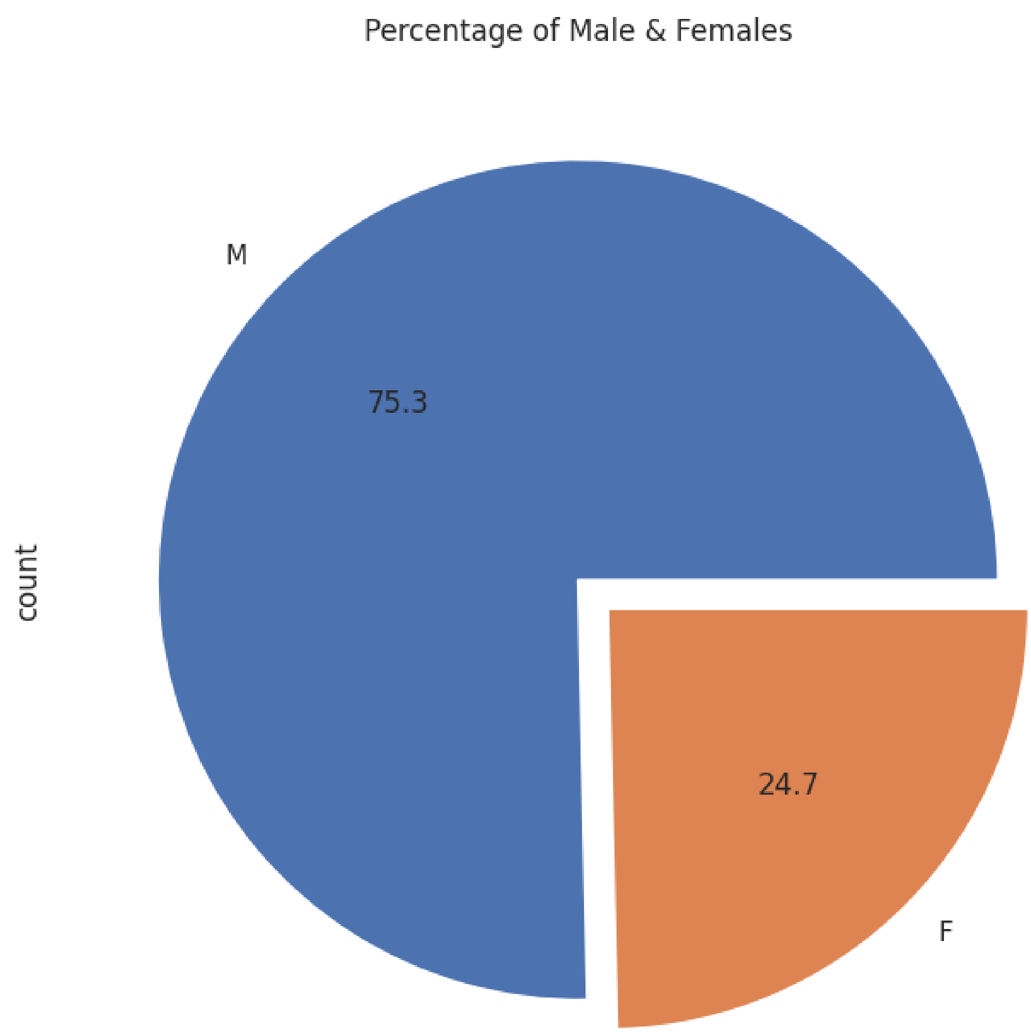
[43]:
```
df.head()
```

[43]:
```
   MovieID                               Title  \
0        1                           Toy Story
1       48                          Pocahontas
2      150                           Apollo 13
3      260  Star Wars: Episode IV - A New Hope
4      527                    Schindler's List

                                Genres UserID  Rating           Timestamp  \
0          Animation|Children's|Comedy      1       5 2001-01-06 23:37:36
1  Animation|Children's|Musical|Romance      1       5 2001-01-06 23:38:40
2                                Drama      1       5 2000-12-31 22:29:20
3              Action|Adventure|Fantas      1       4 2000-12-31 22:12:16
4                            Drama|War      1       5 2001-01-06 23:36:32

  Gender       Age    Occupation Zip-code ReleaseYear
0      F  Under 18  K-12 student    48067        1995
1      F  Under 18  K-12 student    48067        1995
2      F  Under 18  K-12 student    48067        1995
3      F  Under 18  K-12 student    48067        1977
4      F  Under 18  K-12 student    48067        1993
```

[43]:

2.6 EDA w.r.t Gender

[44]:
```
df['Gender'].value_counts().plot(kind='pie',figsize=(8,8),autopct='%1.
 ↪1f',explode=[0,0.1])
plt.title('Percentage of Male & Females')
plt.show()
```



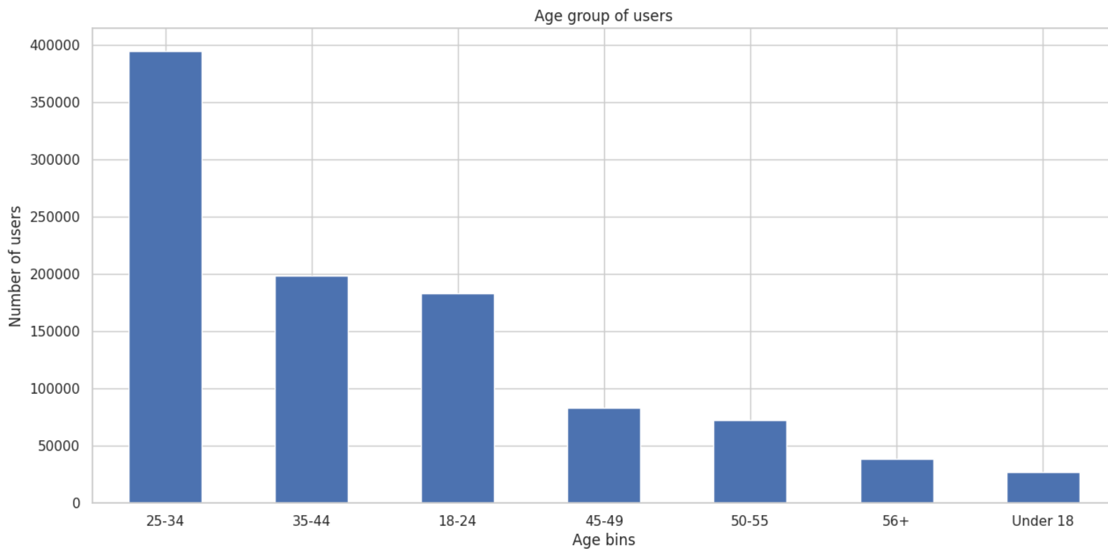
[44]:

```

```

2.7 EDA w.r.t Age:

[45]:

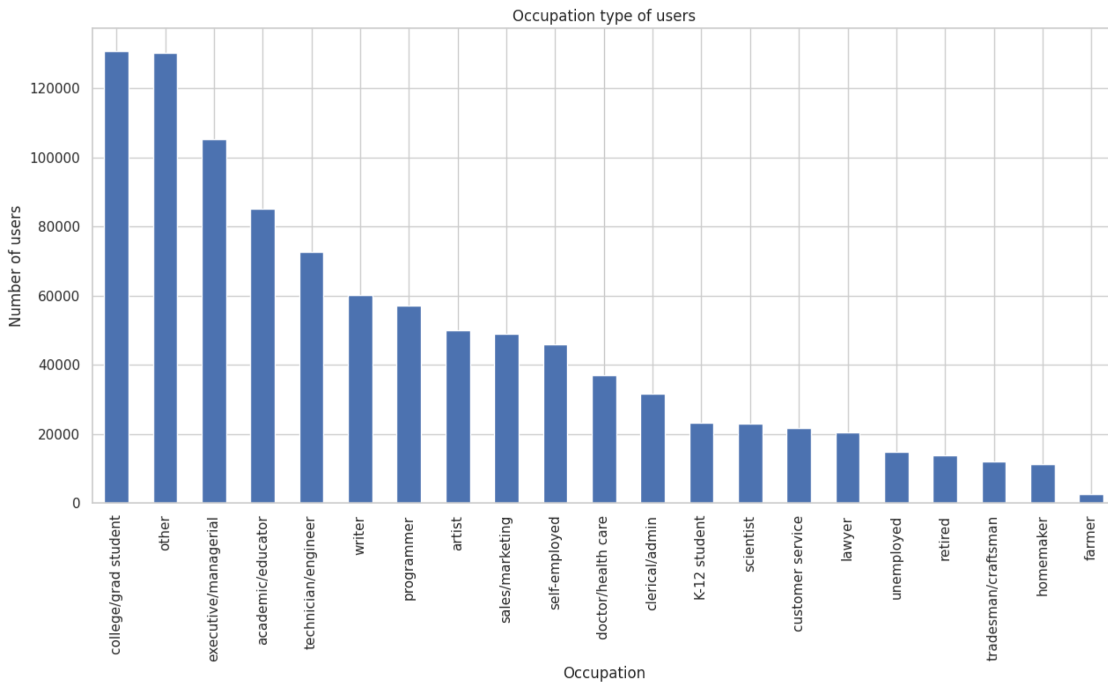
```
df['Age'].value_counts().plot(kind='bar',figsize=(15,7))
plt.title('Age group of users')
plt.xlabel('Age bins')
plt.xticks(rotation = 360)
plt.ylabel('Number of users')
plt.show()
```



[45]:

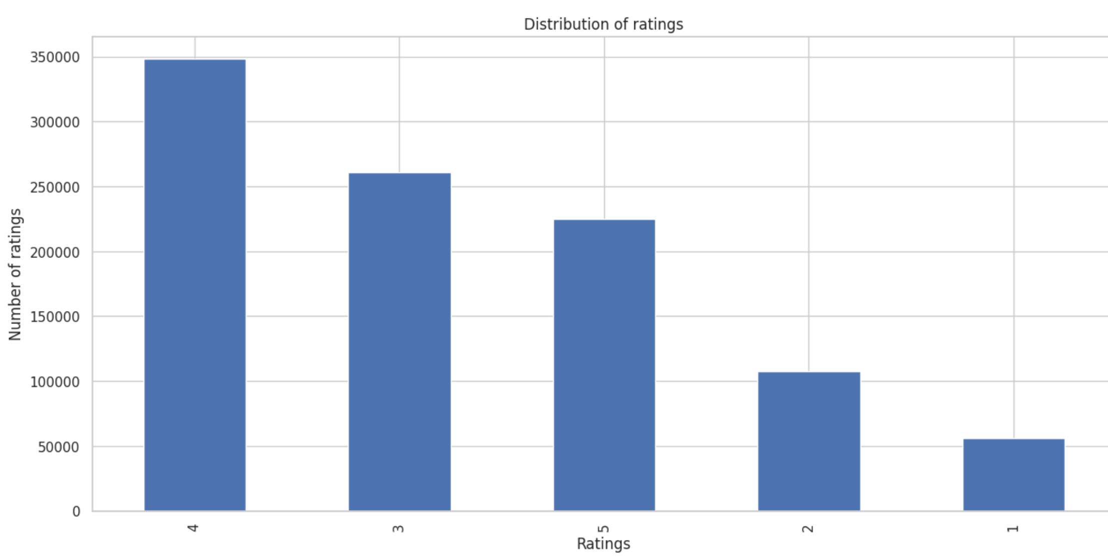
2.8 EDA w.r.t Occupation:

[46]:
```
df['Occupation'].value_counts().plot(kind='bar',figsize=(15,7))
plt.title('Occupation type of users')
plt.xlabel('Occupation')
# plt.xticks(rotation = 45)
plt.ylabel('Number of users')
plt.show()
```



2.9 EDA w.r.t Ratings:

[47]:
```
df['Rating'].value_counts().plot(kind='bar',figsize=(15,7))
plt.title("Distribution of ratings")
plt.xlabel('Ratings')
plt.ylabel('Number of ratings')
plt.show()
```



[47]:

```

```

2.10 Missing and duplicate values:

[48]:

```
def missing_values(df):
    total_null = df.isnull().sum().sort_values(ascending  = False)
    percent_null = ((df.isnull().sum()/df.shape[0])*100).sort_values(ascending␣
 ↪= False)
    print(f"Total records in our data =  {df.shape[0]} where missing values are␣
 ↪as follows:")
    missing_data = pd.concat([total_null, percent_null.round(2)], axis =␣
 ↪1,keys=['Total Missing','In Percent'])
    return missing_data
```

[49]:

```
missing_values(df)
```

```
Total records in our data =  997640 where missing values are as follows:
```

[49]:

	Total Missing	In Percent
Genres	1496	0.15
MovieID	0	0.00
Title	0	0.00
UserID	0	0.00
Rating	0	0.00
Timestamp	0	0.00
Gender	0	0.00
Age	0	0.00
Occupation	0	0.00
Zip-code	0	0.00
ReleaseYear	0	0.00

[50]:

```
duplicates = df[df.duplicated()]
print('The number of duplicates rows:', duplicates.shape[0])
```

```
The number of duplicates rows: 0
```

[50]:

```

```

2.11 Data Aggregation:

[51]:

```
# Grouping the data in terms of Average Rating and No. of Ratings given

df.groupby('Title')['Rating'].mean().sort_values(ascending=False).to_frame().
 ↪reset_index().rename(columns={'Rating':'Average Rating'})
```

[51]:

```
                                                  Title  Average Rating
0                                      Follow the Bitch             5.0
1                                     Bittersweet Motel             5.0
2                  Schlafes Bruder (Brother of Sleep)             5.0
3                                         Smashing Time             5.0
4                                                 Lured             5.0
…                                                     …              …
3646                                  Santa with Muscles             1.0
3647                                     McCullochs, The             1.0
3648  Torso (Corpi Presentano Tracce di Violenza Car…            1.0
3649        Fantastic Night, The (La Nuit Fantastique)             1.0
3650  Blood Spattered Bride, The (La Novia Ensangren…            1.0

[3651 rows x 2 columns]
```

[52]:

```python
df.groupby('Title')['Rating'].count().sort_values(ascending=False).to_frame().
 ↪reset_index().rename(columns={'Rating':'Count'})
```

[52]:

```
                                               Title  Count
0                                    American Beauty   3428
1                 Star Wars: Episode IV - A New Hope   2991
2     Star Wars: Episode V - The Empire Strikes Back   2990
3         Star Wars: Episode VI - Return of the Jedi   2883
4                                      Jurassic Park   2672
…                                                …    …
3646                          Slappy and the Stinkers      1
3647                                          Bye-Bye      1
3648       Silence of the Palace, The (Saimt el Qusur)      1
3649                                  Broken Vessels      1
3650                                  For Ever Mozart      1

[3651 rows x 2 columns]
```

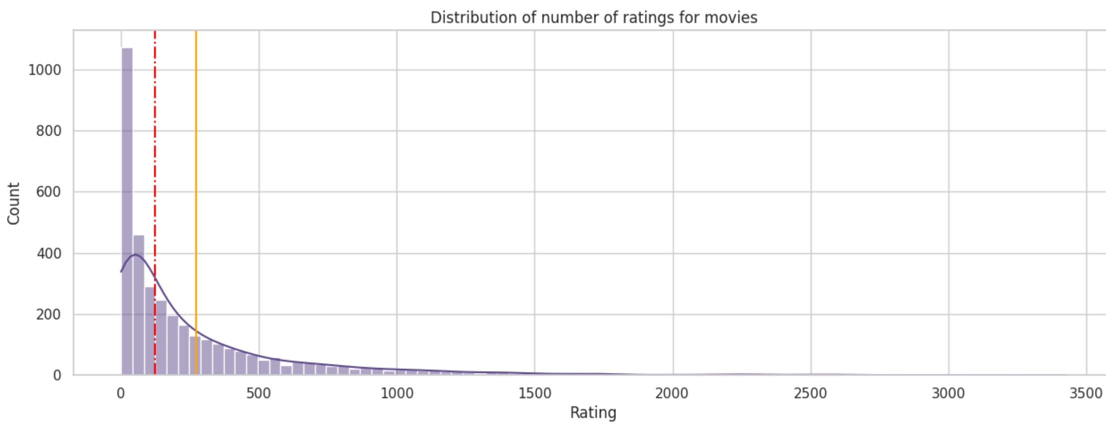
[53]:

```python
df_2 = pd.merge(df.groupby('Title')['Rating'].mean().
 ↪sort_values(ascending=False).to_frame().reset_index().
 ↪rename(columns={'Rating':'Average Rating'}),df.groupby('Title')['Rating'].
 ↪count().sort_values(ascending=False).to_frame().reset_index().
 ↪rename(columns={'Rating':'Count'}),on='Title')
df_2.head()
```

[53]:

```
                                 Title  Average Rating  Count
0                     Follow the Bitch             5.0      1
1                    Bittersweet Motel             5.0      1
2  Schlafes Bruder (Brother of Sleep)             5.0      1
3                        Smashing Time             5.0      2
4                                Lured             5.0      1
```

[54]:
```
import seaborn as sns
fig = plt.figure(figsize=(15,5))
ax = fig.add_subplot(111)
sns.histplot(df.groupby('Title')['Rating'].
 ↪count(),kde=True,ax=ax,color='#5F4B8B')
ax.axvline(df.groupby('Title')['Rating'].count().mean(), color='orange',␣
 ↪linestyle='-',linewidth=1.5)
ax.axvline(df.groupby('Title')['Rating'].count().median(), color='red',␣
 ↪linestyle='-.',linewidth=1.5)
ax.set_title("Distribution of number of ratings for movies")
plt.show()
```



[55]:
```
df.groupby('Title')['Rating'].count().median(), df.groupby('Title')['Rating'].
 ↪count().mean().round(2)
```

[55]: (125.0, 273.25)

[55]:
```

```

On an average, every movie receives 273 ratings and the median count for ratings is 125. Hence we shall filter all movies which have received less than 125 ratings to find the top rated movies

2.12 Top 5 Rating-wise movies:

[56]:
```
df_2[df_2['Count'] > 125].sort_values(by=['Average Rating'],ascending=False).
 ↪head(5)
```

[56]:

	Title	Average Rating	Count
14	Shawshank Redemption, The	4.554558	2227
15	Godfather, The	4.524966	2223
16	Close Shave, A	4.520548	657
17	Usual Suspects, The	4.517106	1783

```
18          Schindler's List        4.510417   2304
```

[56]:

```

```

3 Building a Recommender System based on Pearson Correlation

3.0.1 Creating a pivot table of movie titles & user id and imputing the NaN values

[57]:

```python
# pivot table

tab = pd.
 ↪pivot_table(df,index='UserID',columns='Title',values='Rating',aggfunc='mean')
tab.head()
```

[57]:

```
Title   $1,000,000 Duck  'Night Mother  'Til There Was You  'burbs, The  \
UserID
1                   NaN            NaN                 NaN          NaN
10                  NaN            NaN                 NaN          4.0
100                 NaN            NaN                 NaN          NaN
1000                NaN            NaN                 NaN          NaN
1001                NaN            NaN                 NaN          NaN

Title   …And Justice for All  1-900  10 Things I Hate About You  \
UserID
1                        NaN    NaN                         NaN
10                       NaN    NaN                         NaN
100                      NaN    NaN                         NaN
1000                     NaN    NaN                         NaN
1001                     NaN    NaN                         NaN

Title   101 Dalmatians  12 Angry Men  13th Warrior, The  …  \
UserID                                                   …
1                  NaN           NaN                NaN  …
10                 NaN           3.0                4.0  …
100                NaN           NaN                NaN  …
1000               4.0           NaN                NaN  …
1001               3.0           NaN                NaN  …

Title   Young Poisoner's Handbook, The  Young Sherlock Holmes  \
UserID
1                                  NaN                    NaN
10                                 NaN                    NaN
100                                NaN                    NaN
1000                               NaN                    NaN
1001                               NaN                    NaN
```

```
Title   Young and Innocent  Your Friends and Neighbors  Zachariah  \
UserID
1                      NaN                         NaN        NaN
10                     NaN                         NaN        NaN
100                    NaN                         NaN        NaN
1000                   NaN                         NaN        NaN
1001                   NaN                         4.0        NaN

Title   Zed & Two Noughts, A  Zero Effect  \
UserID
1                        NaN          NaN
10                       NaN          NaN
100                      NaN          NaN
1000                     NaN          NaN
1001                     NaN          NaN

Title   Zero Kelvin (Kjærlighetens kjøtere)  Zeus and Roxanne  eXistenZ
UserID
1                                       NaN               NaN       NaN
10                                      NaN               NaN       NaN
100                                     NaN               NaN       NaN
1000                                    NaN               NaN       NaN
1001                                    NaN               NaN       5.0

[5 rows x 3651 columns]
```

We can clearly see that there are a lot of NaN values, we should fill it with 0. We have a sparse matrix

[58]:
```
tab.fillna(0,inplace=True)
```

[59]:
```
tab.shape
```

[59]:
```
(6040, 3651)
```

3.1 Item based approach

Here we shall take a movie name as input from the user and return 5 other movies which are similar to the user's choice. To calculate the similarity, I am using pearson correlation.

[67]:
```
mov = input("Enter a movie name : ")
mov_rating = tab[mov]
```

```
Enter a movie name : 101 Dalmatians
```

[68]:
```
similar_movies = tab.corrwith(mov_rating)
```

[69]:
```
#Pearson Correlation

similar_movies.sort_values(ascending=False).to_frame().rename(columns={0:
    "Correlation"}).head()
```

[69]:
```
                  Correlation
Title
101 Dalmatians       1.000000
Bambi                0.463776
Pinocchio            0.459928
Cinderella           0.459570
Dumbo                0.457344
```

[69]:
```

```

3.2 User-based approach

Using the User-based approach to create a recommender system that uses Pearson Correlation

[70]:
```
tab_transpose = tab.T
```

[71]:
```
tab_transpose
```

[71]:
```
UserID                                  1   10  100  1000  1001  1002  1003  \
Title
$1,000,000 Duck                       0.0  0.0  0.0   0.0   0.0   0.0   0.0
'Night Mother                         0.0  0.0  0.0   0.0   0.0   0.0   0.0
'Til There Was You                    0.0  0.0  0.0   0.0   0.0   0.0   0.0
'burbs, The                           0.0  4.0  0.0   0.0   0.0   0.0   0.0
…And Justice for All                0.0  0.0  0.0   0.0   0.0   0.0   0.0
…                                   …   …   …    …    …    …    …
Zed & Two Noughts, A                  0.0  0.0  0.0   0.0   0.0   0.0   0.0
Zero Effect                           0.0  0.0  0.0   0.0   0.0   0.0   0.0
Zero Kelvin (Kjærlighetens kjøtere)  0.0  0.0  0.0   0.0   0.0   0.0   0.0
Zeus and Roxanne                      0.0  0.0  0.0   0.0   0.0   0.0   0.0
eXistenZ                              0.0  0.0  0.0   0.0   5.0   0.0   0.0

UserID                                1004  1005  1006  …  990  991  992  \
Title                                                   …
$1,000,000 Duck                        0.0   0.0   0.0  …  0.0  0.0  0.0
'Night Mother                          0.0   0.0   0.0  …  0.0  0.0  0.0
'Til There Was You                     0.0   0.0   0.0  …  0.0  0.0  0.0
'burbs, The                            0.0   0.0   0.0  …  0.0  0.0  0.0
…And Justice for All                 0.0   0.0   0.0  …  0.0  0.0  0.0
…                                    …    …    …   …   …   …   …
Zed & Two Noughts, A                   0.0   0.0   0.0  …  0.0  0.0  0.0
Zero Effect                            0.0   0.0   0.0  …  0.0  0.0  0.0
```

```
Zero Kelvin (Kjærlighetens kjøtere)   0.0   0.0   0.0  …  0.0  0.0  0.0
Zeus and Roxanne                      0.0   0.0   0.0  …  0.0  0.0  0.0
eXistenZ                              0.0   0.0   3.0  …  0.0  0.0  0.0

UserID                               993  994  995  996  997  998  999
Title
$1,000,000 Duck                      0.0  0.0  0.0  0.0  0.0  0.0  0.0
'Night Mother                        0.0  0.0  0.0  0.0  0.0  0.0  0.0
'Til There Was You                   0.0  0.0  0.0  0.0  0.0  0.0  0.0
'burbs, The                          0.0  0.0  0.0  0.0  0.0  0.0  0.0
…And Justice for All               0.0  0.0  0.0  0.0  0.0  0.0  3.0
…                                  …   …   …   …   …   …   …
Zed & Two Noughts, A                 0.0  0.0  0.0  0.0  0.0  0.0  0.0
Zero Effect                          0.0  0.0  0.0  0.0  0.0  0.0  0.0
Zero Kelvin (Kjærlighetens kjøtere)  0.0  0.0  0.0  0.0  0.0  0.0  0.0
Zeus and Roxanne                     0.0  0.0  0.0  0.0  0.0  0.0  0.0
eXistenZ                             0.0  0.0  0.0  4.0  0.0  0.0  3.0

[3651 rows x 6040 columns]
```

[72]:
```
user_id =input("Enter a user_id : ")
user_id_recomm = tab_transpose[user_id]
```

```
Enter a user_id : 1000
```

[73]:
```
similar_movie_user_based = tab_transpose.corrwith(user_id_recomm)
```

[74]:
```
#Pearson Correlation

similar_movie_user_based.sort_values(ascending=False).to_frame().
↪rename(columns={0:"Correlation"}).head()
```

[74]:
```
        Correlation
UserID
1000       1.000000
5842       0.522725
2539       0.522259
4966       0.499219
3906       0.464419
```

Obs ->

- As we can clearly see, movies interests similar to user with user id == 5 are the users(1484, 5452, 281, 3538) and thus we can recommend these users similar movies that user_id == 5 is watching.
- Although, this is a weak correlation (0.33)

4 Building a Recommender System based on Cosine Similarity and KNN

[75]:
```
from sklearn.metrics.pairwise import cosine_similarity
item_sim = cosine_similarity(tab.T) # row based and hence for getting moview␣
↪(items) entries in rows, we are transposing
item_sim
```

[75]:
```
array([[1.        , 0.07235746, 0.03701053, …, 0.        , 0.12024178,
        0.02700277],
       [0.07235746, 1.        , 0.11528952, …, 0.        , 0.        ,
        0.07780705],
       [0.03701053, 0.11528952, 1.        , …, 0.        , 0.04752635,
        0.0632837 ],
       …,
       [0.        , 0.        , 0.        , …, 1.        , 0.        ,
        0.04564448],
       [0.12024178, 0.        , 0.04752635, …, 0.        , 1.        ,
        0.04433508],
       [0.02700277, 0.07780705, 0.0632837 , …, 0.04564448, 0.04433508,
        1.        ]])
```

Obs -> Similarity of one movie each (each list) accross all the users

4.1 Item-Item Similarity:

[76]:
```
# Item-Item Similarity Matrix

item_sim_mat = pd.DataFrame(item_sim,index=tab.columns,columns = tab.columns)
item_sim_mat
```

[76]:
```
Title                                 $1,000,000 Duck  'Night Mother  \
Title
$1,000,000 Duck                              1.000000       0.072357
'Night Mother                                0.072357       1.000000
'Til There Was You                           0.037011       0.115290
'burbs, The                                  0.079291       0.115545
…And Justice for All                         0.060838       0.159526
…                                                 …              …
Zed & Two Noughts, A                         0.045280       0.091150
Zero Effect                                  0.039395       0.074787
Zero Kelvin (Kjærlighetens kjøtere)          0.000000       0.000000
Zeus and Roxanne                             0.120242       0.000000
eXistenZ                                     0.027003       0.077807

Title                                 'Til There Was You  'burbs, The  \
Title
```

```
$1,000,000 Duck                                0.037011     0.079291
'Night Mother                                  0.115290     0.115545
'Til There Was You                             1.000000     0.098756
'burbs, The                                    0.098756     1.000000
…And Justice for All                         0.066301     0.143620
…                                                …            …
Zed & Two Noughts, A                           0.022594     0.055704
Zero Effect                                    0.079261     0.161174
Zero Kelvin (Kjærlighetens kjøtere)            0.000000     0.000000
Zeus and Roxanne                               0.047526     0.033567
eXistenZ                                       0.063284     0.110525

Title                                 …And Justice for All     1-900  \
Title
$1,000,000 Duck                                    0.060838  0.000000
'Night Mother                                      0.159526  0.000000
'Til There Was You                                 0.066301  0.080250
'burbs, The                                        0.143620  0.000000
…And Justice for All                             1.000000  0.000000
…                                                    …         …
Zed & Two Noughts, A                               0.086080  0.000000
Zero Effect                                        0.110867  0.000000
Zero Kelvin (Kjærlighetens kjøtere)                0.074317  0.000000
Zeus and Roxanne                                   0.000000  0.000000
eXistenZ                                           0.111040  0.039561

Title                                 10 Things I Hate About You  \
Title
$1,000,000 Duck                                         0.058619
'Night Mother                                           0.076798
'Til There Was You                                      0.127895
'burbs, The                                             0.192191
…And Justice for All                                  0.075093
…                                                         …
Zed & Two Noughts, A                                    0.012702
Zero Effect                                             0.175771
Zero Kelvin (Kjærlighetens kjøtere)                     0.000000
Zeus and Roxanne                                        0.058708
eXistenZ                                                0.162060

Title                                 101 Dalmatians  12 Angry Men  \
Title
$1,000,000 Duck                             0.189843      0.094785
'Night Mother                               0.137135      0.111413
'Til There Was You                          0.128523      0.079115
'burbs, The                                 0.250140      0.170719
…And Justice for All                      0.178928      0.205486
```

```
…                                          …            …
Zed & Two Noughts, A                       0.042295     0.039344
Zero Effect                                0.157313     0.133061
Zero Kelvin (Kjærlighetens kjøtere)        0.033120     0.036867
Zeus and Roxanne                           0.089840     0.058692
eXistenZ                                   0.120762     0.098731

Title                                13th Warrior, The  …  \
Title                                                   …
$1,000,000 Duck                               0.058418  …
'Night Mother                                 0.046135  …
'Til There Was You                            0.066598  …
'burbs, The                                   0.197808  …
…And Justice for All                        0.122431  …
…                                                  …  …
Zed & Two Noughts, A                          0.041324  …
Zero Effect                                   0.156505  …
Zero Kelvin (Kjærlighetens kjøtere)           0.034797  …
Zeus and Roxanne                              0.034623  …
eXistenZ                                      0.230799  …

Title                                Young Poisoner's Handbook, The  \
Title
$1,000,000 Duck                                            0.038725
'Night Mother                                              0.053010
'Til There Was You                                         0.029200
'burbs, The                                                0.113386
…And Justice for All                                     0.089998
…                                                                …
Zed & Two Noughts, A                                       0.047282
Zero Effect                                                0.179315
Zero Kelvin (Kjærlighetens kjøtere)                        0.048440
Zeus and Roxanne                                           0.000000
eXistenZ                                                   0.115734

Title                                Young Sherlock Holmes  \
Title
$1,000,000 Duck                                   0.076474
'Night Mother                                     0.087828
'Til There Was You                                0.062893
'burbs, The                                       0.207897
…And Justice for All                            0.153006
…                                                      …
Zed & Two Noughts, A                              0.073996
Zero Effect                                       0.169677
Zero Kelvin (Kjærlighetens kjøtere)               0.046892
Zeus and Roxanne                                  0.046658
```

```
eXistenZ                                        0.180174

Title                                     Young and Innocent  \
Title
$1,000,000 Duck                                     0.000000
'Night Mother                                       0.063758
'Til There Was You                                  0.000000
'burbs, The                                         0.019962
…And Justice for All                                0.067009
…                                                   …
Zed & Two Noughts, A                                0.070409
Zero Effect                                         0.021362
Zero Kelvin (Kjærlighetens kjøtere)                 0.000000
Zeus and Roxanne                                    0.000000
eXistenZ                                            0.024437

Title                                     Your Friends and Neighbors  Zachariah  \
Title
$1,000,000 Duck                                             0.044074        0.0
'Night Mother                                               0.135962        0.0
'Til There Was You                                          0.079187        0.0
'burbs, The                                                 0.138064        0.0
…And Justice for All                                        0.109029        0.0
…                                                           …           …
Zed & Two Noughts, A                                        0.141537        0.0
Zero Effect                                                 0.206870        0.0
Zero Kelvin (Kjærlighetens kjøtere)                         0.000000        0.0
Zeus and Roxanne                                            0.024489        0.0
eXistenZ                                                    0.149749        0.0

Title                                     Zed & Two Noughts, A  Zero Effect  \
Title
$1,000,000 Duck                                       0.045280     0.039395
'Night Mother                                         0.091150     0.074787
'Til There Was You                                    0.022594     0.079261
'burbs, The                                           0.055704     0.161174
…And Justice for All                                  0.086080     0.110867
…                                                     …            …
Zed & Two Noughts, A                                  1.000000     0.084020
Zero Effect                                           0.084020     1.000000
Zero Kelvin (Kjærlighetens kjøtere)                   0.124939     0.059228
Zeus and Roxanne                                      0.000000     0.016838
eXistenZ                                              0.137372     0.242043

Title                                     Zero Kelvin (Kjærlighetens kjøtere)  \
Title
$1,000,000 Duck                                                      0.000000
```

```
'Night Mother                                               0.000000
'Til There Was You                                          0.000000
'burbs, The                                                 0.000000
…And Justice for All                                      0.074317
…                                                              …
Zed & Two Noughts, A                                        0.124939
Zero Effect                                                 0.059228
Zero Kelvin (Kjærlighetens kjøtere)                         1.000000
Zeus and Roxanne                                            0.000000
eXistenZ                                                    0.045644

Title                                 Zeus and Roxanne  eXistenZ
Title
$1,000,000 Duck                               0.120242  0.027003
'Night Mother                                 0.000000  0.077807
'Til There Was You                            0.047526  0.063284
'burbs, The                                   0.033567  0.110525
…And Justice for All                        0.000000  0.111040
…                                                  …         …
Zed & Two Noughts, A                          0.000000  0.137372
Zero Effect                                   0.016838  0.242043
Zero Kelvin (Kjærlighetens kjøtere)           0.000000  0.045644
Zeus and Roxanne                              1.000000  0.044335
eXistenZ                                      0.044335  1.000000

[3651 rows x 3651 columns]
```

[77]:

```
user_sim = cosine_similarity(tab)
user_sim
```

[77]:

```
array([[1.        , 0.25531859, 0.12396703, …, 0.15926709, 0.11935626,
        0.12239079],
       [0.25531859, 1.        , 0.25964457, …, 0.16569953, 0.13332665,
        0.24845029],
       [0.12396703, 0.25964457, 1.        , …, 0.20430203, 0.11352239,
        0.30693676],
       …,
       [0.15926709, 0.16569953, 0.20430203, …, 1.        , 0.18657496,
        0.18563871],
       [0.11935626, 0.13332665, 0.11352239, …, 0.18657496, 1.        ,
        0.10827118],
       [0.12239079, 0.24845029, 0.30693676, …, 0.18563871, 0.10827118,
        1.        ]])
```

4.2 User-User Similarity:

[78]:
```
# User Similarity Matrix:

user_sim_mat = pd.DataFrame(user_sim,index=tab.index,columns = tab.index)
user_sim_mat
```

[78]:
```
UserID         1        10       100      1000      1001      1002      1003  \
UserID
1       1.000000  0.255319  0.123967  0.207800  0.139317  0.110320  0.121384
10      0.255319  1.000000  0.259645  0.280479  0.158703  0.112917  0.141985
100     0.123967  0.259645  1.000000  0.306067  0.075736  0.110450  0.358686
1000    0.207800  0.280479  0.306067  1.000000  0.099117  0.047677  0.201722
1001    0.139317  0.158703  0.075736  0.099117  1.000000  0.164854  0.053887
…            …         …         …         …         …         …         …
995     0.035731  0.146552  0.033754  0.044404  0.109700  0.072578  0.031406
996     0.170184  0.304806  0.344290  0.330748  0.222119  0.224779  0.185226
997     0.159267  0.165700  0.204302  0.172803  0.103255  0.068980  0.170771
998     0.119356  0.133327  0.113522  0.098456  0.269952  0.218905  0.141829
999     0.122391  0.248450  0.306937  0.250564  0.178399  0.178474  0.198656

UserID      1004      1005      1006  …       990       991       992  \
UserID                                …
1       0.180073  0.103896  0.052816  …  0.079367  0.038048  0.032136
10      0.432171  0.194915  0.102487  …  0.154412  0.186234  0.083739
100     0.237292  0.172872  0.099147  …  0.098235  0.097953  0.065152
1000    0.355619  0.325966  0.130702  …  0.170100  0.076779  0.000000
1001    0.150069  0.138602  0.134710  …  0.146270  0.026891  0.097011
…            …         …         … …  …         …         …
995     0.088763  0.061450  0.032265  …  0.080559  0.252222  0.074207
996     0.351716  0.287965  0.164045  …  0.205186  0.086546  0.062523
997     0.175340  0.106303  0.049536  …  0.192642  0.030588  0.081380
998     0.075474  0.112029  0.052900  …  0.061241  0.074269  0.086398
999     0.334188  0.164777  0.143866  …  0.214809  0.085280  0.040307

UserID       993       994       995       996       997       998       999
UserID
1       0.067631  0.070052  0.035731  0.170184  0.159267  0.119356  0.122391
10      0.125894  0.118558  0.146552  0.304806  0.165700  0.133327  0.248450
100     0.178664  0.271311  0.033754  0.344290  0.204302  0.113522  0.306937
1000    0.200343  0.380741  0.044404  0.330748  0.172803  0.098456  0.250564
1001    0.119609  0.092234  0.109700  0.222119  0.103255  0.269952  0.178399
…            …         …         …         …         …         …         …
995     0.098705  0.048650  1.000000  0.063925  0.019459  0.075830  0.052571
996     0.186441  0.217672  0.063925  1.000000  0.179404  0.178834  0.418466
997     0.162615  0.110656  0.019459  0.179404  1.000000  0.186575  0.185639
998     0.166462  0.018659  0.075830  0.178834  0.186575  1.000000  0.108271
```

```
999     0.168252  0.161995  0.052571  0.418466  0.185639  0.108271  1.000000

[6040 rows x 6040 columns]
```

[79]:
```
# Creating a csr based decomposition/conversion of sparse matrix

from scipy.sparse import csr_matrix
csr_mat = csr_matrix(tab.T.values)
```

5 KNN based Recommender System:

[80]:
```
from sklearn.neighbors import NearestNeighbors
knn = NearestNeighbors(n_neighbors= 5,metric = 'cosine', n_jobs=-1)
knn.fit(csr_mat)
```

[80]:
```
NearestNeighbors(metric='cosine', n_jobs=-1)
```

[81]:
```
movie_name = 'Shawshank Redemption, The'
distances, indices = knn.kneighbors(tab[movie_name].values.
 ↪reshape(1,-1),n_neighbors=6)
for i in range (0, len(distances.flatten())):
  if i == 0 :
    print("The movies close similar to",movie_name,"are \n")
  else:
    print(tab.columns[indices.flatten()[i]],"with a distance␣
 ↪of",round(distances.flatten()[i],3))
```

```
The movies close similar to Shawshank Redemption, The are

Silence of the Lambs, The with a distance of 0.319
Pulp Fiction with a distance of 0.341
Fargo with a distance of 0.344
Schindler's List with a distance of 0.345
Good Will Hunting with a distance of 0.368
```

- Note, we are **getting the same two movies as we got in case of Pearson corelation** approach.
- **No significant difference with KNN** based approach

6. Name the top 3 movies similar to 'Liar Liar' on the item-based approach.

[82]:
```
# 6. Name the top 3 movies similar to 'Liar Liar' on the item-based approach.
movie_name = 'Liar Liar'
distances, indices = knn.kneighbors(tab[movie_name].values.
 ↪reshape(1,-1),n_neighbors=4)
for i in range (0, len(distances.flatten())):
  if i == 0 :
```

```
    print("The movies close similar to",movie_name,"are \n")
  else:
    print(tab.columns[indices.flatten()[i]],"with a distance␣
↪of",round(distances.flatten()[i],3))
```

```
The movies close similar to Liar Liar are

Mrs. Doubtfire with a distance of 0.443
Ace Ventura: Pet Detective with a distance of 0.483
Dumb & Dumber with a distance of 0.487
```

6 Building a Recommender System based on Matrix Factorization

[83]:
```
users = df.UserID.unique()
movies = df.MovieID.unique()
```

[84]:
```
userid2idx = {o:i for i,o in enumerate(users)}
movieid2idx = {o:i for i,o in enumerate(movies)}
```

[85]:
```
df['UserID'] = df['UserID'].apply(lambda x : userid2idx[x])
df['MovieID'] = df['MovieID'].apply(lambda x : movieid2idx[x])
```

[86]:
```
train = df[np.random.rand(len(df)) < 0.8]
valid = df[~(np.random.rand(len(df)) < 0.8)]
print(train.shape,valid.shape)
```

```
(797804, 11) (199288, 11)
```

[87]:
```
n_movies = len(df['MovieID'].unique())
n_users = len(df['UserID'].unique())
n_latent_factors = 64 # Hyperparameter or hidden model factors
```

[88]:
```
# Creating embedding vector for user using embedding layer from Keras

import tensorflow as tf
user_input = tf.keras.Input(shape=(1,) ,name='user_input', dtype='int64')
user_embedding = tf.keras.layers.Embedding(n_users,n_latent_factors,␣
↪name='user_embedding')(user_input)
user_vec = tf.keras.layers.Flatten(name='FlattenUsers')(user_embedding)
```

[89]:
```
# Creating embedding vector for movie using embedding layer from Keras

movie_input = tf.keras.Input(shape=(1,) ,name='movie_input', dtype='int64')
movie_embedding = tf.keras.layers.Embedding(n_users,n_latent_factors,␣
↪name='movie_embedding')(movie_input)
movie_vec = tf.keras.layers.Flatten(name='FlattenMovies')(movie_embedding)
```

[91]:
```
!pip install keras
```

```
Requirement already satisfied: keras in /usr/local/lib/python3.10/dist-packages
(3.4.1)
Requirement already satisfied: absl-py in /usr/local/lib/python3.10/dist-
packages (from keras) (1.4.0)
Requirement already satisfied: numpy in /usr/local/lib/python3.10/dist-packages
(from keras) (1.26.4)
Requirement already satisfied: rich in /usr/local/lib/python3.10/dist-packages
(from keras) (13.7.1)
Requirement already satisfied: namex in /usr/local/lib/python3.10/dist-packages
(from keras) (0.0.8)
Requirement already satisfied: h5py in /usr/local/lib/python3.10/dist-packages
(from keras) (3.11.0)
Requirement already satisfied: optree in /usr/local/lib/python3.10/dist-packages
(from keras) (0.12.1)
Requirement already satisfied: ml-dtypes in /usr/local/lib/python3.10/dist-
packages (from keras) (0.4.0)
Requirement already satisfied: packaging in /usr/local/lib/python3.10/dist-
packages (from keras) (24.1)
Requirement already satisfied: typing-extensions>=4.5.0 in
/usr/local/lib/python3.10/dist-packages (from optree->keras) (4.12.2)
Requirement already satisfied: markdown-it-py>=2.2.0 in
/usr/local/lib/python3.10/dist-packages (from rich->keras) (3.0.0)
Requirement already satisfied: pygments<3.0.0,>=2.13.0 in
/usr/local/lib/python3.10/dist-packages (from rich->keras) (2.16.1)
Requirement already satisfied: mdurl~=0.1 in /usr/local/lib/python3.10/dist-
packages (from markdown-it-py>=2.2.0->rich->keras) (0.1.2)
```

[94]:
```
import tensorflow as tf
from tensorflow.keras.layers import dot  # Import dot directly from tensorflow.
 ↪keras.layers

sim = dot([user_vec, movie_vec], name='Similarity-Dot-Product', axes=1)
model = tf.keras.models.Model([user_input, movie_input], sim)  # Use tf.keras.
 ↪models for consistency
```

[96]:
```
from tensorflow.keras.optimizers import Adam
model.compile(optimizer = Adam(learning_rate = 1e-4),loss='mse') # Use␣
 ↪learning_rate instead of lr
```

[97]:
```
model.summary()
```

Model: "functional"

```
Layout (type)                  Output Shape                   Param #   Connected␣
↪to

user_input (InputLayer)        (None, 1)                            0   -          ␣
↪

movie_input (InputLayer)       (None, 1)                            0   -          ␣
↪

user_embedding                 (None, 1, 64)                  386,560  ␣
↪user_input[0][0]
(Embedding)                                                                        ␣
↪

movie_embedding                (None, 1, 64)                  386,560  ␣
↪movie_input[0][0]
(Embedding)                                                                        ␣
↪

FlattenUsers (Flatten)         (None, 64)                           0  ␣
↪user_embedding[0][0]

FlattenMovies (Flatten)        (None, 64)                           0  ␣
↪movie_embedding[0][0]

Similarity-Dot-Product         (None, 1)                            0  ␣
↪FlattenUsers[0][0],
(Dot)                                                                  ␣
↪FlattenMovies[0][0]
```

Total params: 773,120 (2.95 MB)

Trainable params: 773,120 (2.95 MB)

Non-trainable params: 0 (0.00 B)

- The metrics used here is **Mean Squared Error (MSE)**
- Our objective is to **minimize this MSE** on traing data set i.e over the values which user has rated

[98]:
```
# Model Training
```

```python
model_hist = model.fit([train.UserID, train.MovieID], train.Rating, batch_size
= 120, epochs = 20, validation_data = ([valid.UserID, valid.MovieID], valid.
Rating), verbose = 1)
```

```
Epoch 1/20
6649/6649             72s 10ms/step -
loss: 14.0371 - val_loss: 13.3049
Epoch 2/20
6649/6649             80s 10ms/step -
loss: 11.4688 - val_loss: 4.8607
Epoch 3/20
6649/6649             66s 10ms/step -
loss: 3.7482 - val_loss: 1.9177
Epoch 4/20
6649/6649             65s 10ms/step -
loss: 1.6815 - val_loss: 1.2216
Epoch 5/20
6649/6649             61s 9ms/step -
loss: 1.1532 - val_loss: 0.9913
Epoch 6/20
6649/6649             83s 9ms/step -
loss: 0.9680 - val_loss: 0.8994
Epoch 7/20
6649/6649             63s 9ms/step -
loss: 0.8911 - val_loss: 0.8578
Epoch 8/20
6649/6649             59s 9ms/step -
loss: 0.8566 - val_loss: 0.8365
Epoch 9/20
6649/6649             59s 9ms/step -
loss: 0.8355 - val_loss: 0.8240
Epoch 10/20
6649/6649             84s 9ms/step -
loss: 0.8233 - val_loss: 0.8150
Epoch 11/20
6649/6649             80s 9ms/step -
loss: 0.8131 - val_loss: 0.8070
Epoch 12/20
6649/6649             86s 10ms/step -
loss: 0.8065 - val_loss: 0.7997
Epoch 13/20
6649/6649             76s 9ms/step -
loss: 0.7974 - val_loss: 0.7919
Epoch 14/20
6649/6649             58s 9ms/step -
loss: 0.7937 - val_loss: 0.7849
Epoch 15/20
```

```
6649/6649             62s 9ms/step -
loss: 0.7850 - val_loss: 0.7780
Epoch 16/20
6649/6649             59s 9ms/step -
loss: 0.7761 - val_loss: 0.7710
Epoch 17/20
6649/6649             59s 9ms/step -
loss: 0.7688 - val_loss: 0.7643
Epoch 18/20
6649/6649             59s 9ms/step -
loss: 0.7593 - val_loss: 0.7578
Epoch 19/20
6649/6649             59s 9ms/step -
loss: 0.7553 - val_loss: 0.7518
Epoch 20/20
6649/6649             58s 9ms/step -
loss: 0.7450 - val_loss: 0.7454
```

[99]:
```
# Model Evaluation:

y_pred = model.predict([valid.UserID, valid.MovieID], verbose  =0)
y_pred_class  =np.argmax(y_pred, axis  =1)
```

[109]:
```
# RMSE

from sklearn.metrics import mean_squared_error
rmse = mean_squared_error(valid.Rating, y_pred, squared = False)
print(f"RMSE : {rmse}")
```

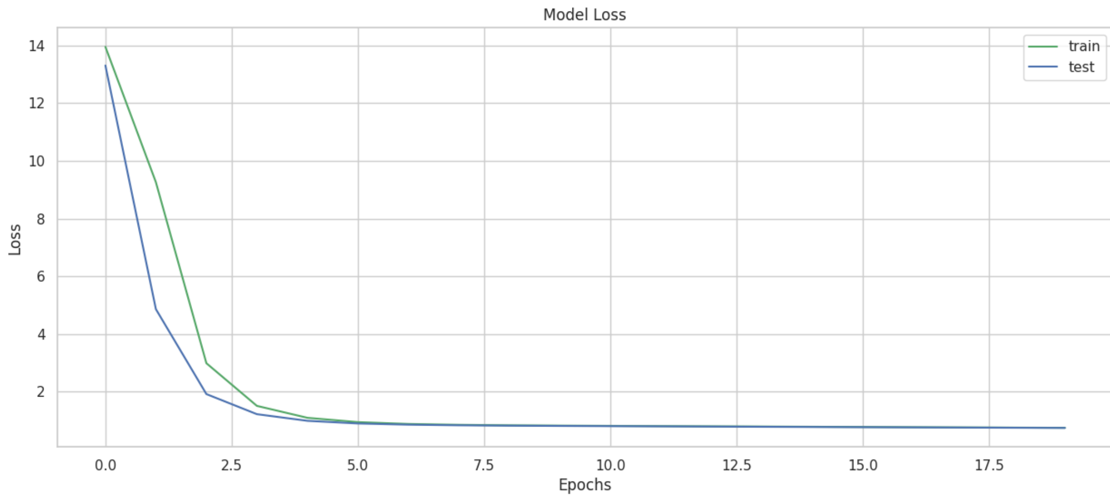
```
RMSE : 0.8633609489643959
```

[101]:
```
# MAPE

from sklearn.metrics import mean_absolute_percentage_error
mape = mean_absolute_percentage_error(valid.Rating, y_pred)
print(f"MAPE : {mape}")
```

```
MAPE : 0.26376498911146157
```

[102]:
```
# Plotting the model loss

# rcParams['figure.figsize'] = 10, 5
plt.plot(model_hist.history['loss'], 'g')
plt.plot(model_hist.history['val_loss'], 'b')
plt.title('Model Loss')
plt.ylabel('Loss')
plt.xlabel('Epochs')
plt.legend(['train', 'test'], loc = 'upper right')
```

```
plt.grid(True)
plt.show()
```



[102]:

```

```

- As we can see, after 4 epochs, the loss is becoming almost constant.
- Train and test losses are coinsiding.
- The model is good.

[103]:

```
df
```

[103]:

```
         MovieID                                  Title  \
0              0                              Toy Story
1              1                             Pocahontas
2              2                              Apollo 13
3              3     Star Wars: Episode IV - A New Hope
4              4                       Schindler's List
…            …                                      …
1000204      288                    Rules of Engagement
1000205     1696                        American Psycho
1000206      102                      Keeping the Faith
1000207     1477                                  U-571
1000208      104                              Gladiator

                                Genres  UserID  Rating  \
0                 Animation|Children's|Comedy       0       5
1         Animation|Children's|Musical|Romance       0       5
2                                       Drama       0       5
3                     Action|Adventure|Fantas       0       4
4                                   Drama|War       0       5
```

```
…                                       …     …     …
1000204                   Drama|Thriller   6039      4
1000205          Comedy|Horror|Thriller   6039      2
1000206                   Comedy|Romance   6039      5
1000207                  Action|Thriller   6039      3
1000208                     Action|Drama   6039      5

                   Timestamp Gender      Age          Occupation Zip-code  \
0        2001-01-06 23:37:36      F  Under 18         K-12 student    48067
1        2001-01-06 23:38:40      F  Under 18         K-12 student    48067
2        2000-12-31 22:29:20      F  Under 18         K-12 student    48067
3        2000-12-31 22:12:16      F  Under 18         K-12 student    48067
4        2001-01-06 23:36:32      F  Under 18         K-12 student    48067
…                        …      …        …                  …        …
1000204  2000-05-16 15:13:04      M     25-34  college/grad student    92843
1000205  2000-05-16 15:13:04      M     25-34  college/grad student    92843
1000206  2000-05-16 15:12:00      M     25-34  college/grad student    92843
1000207  2000-05-16 15:24:48      M     25-34  college/grad student    92843
1000208  2000-05-16 15:16:16      M     25-34  college/grad student    92843

         ReleaseYear
0               1995
1               1995
2               1995
3               1977
4               1993
…                  …
1000204         2000
1000205         2000
1000206         2000
1000207         2000
1000208         2000

[997640 rows x 11 columns]
```

1. Users of which age group have watched and rated the most number of movies?

[104]:
```python
df.groupby('Age')['Rating'].count().sort_values(ascending=False).to_frame().
↪reset_index().rename(columns={'Rating':'Count'})
```

[104]:

	Age	Count
0	25-34	394707
1	35-44	198446
2	18-24	183172
3	45-49	83329
4	50-55	72223
5	56+	38616

```
6  Under 18   27147
```

[104]:

```

```

2. Users belonging to which profession have watched and rated the most movies?

[105]:

```
df.groupby('Occupation')['Rating'].count().sort_values(ascending=False).
↪to_frame().reset_index().rename(columns={'Rating':'Count'})
```

[105]:

	Occupation	Count
0	college/grad student	130743
1	other	130208
2	executive/managerial	105168
3	academic/educator	85078
4	technician/engineer	72646
5	writer	60197
6	programmer	57040
7	artist	49926
8	sales/marketing	49002
9	self-employed	45898
10	doctor/health care	37110
11	clerical/admin	31559
12	K-12 student	23245
13	scientist	22866
14	customer service	21804
15	lawyer	20495
16	unemployed	14871
17	retired	13690
18	tradesman/craftsman	12069
19	homemaker	11327
20	farmer	2698

3. Most of the users in our dataset who've rated the movies are Male.

[106]:

```
df.groupby('Gender')['Rating'].count().sort_values(ascending=False).to_frame().
↪reset_index().rename(columns={'Rating':'Count'})
```

[106]:

	Gender	Count
0	M	751679
1	F	245961

4. Most of the movies present on our dataset were released in which decade?

[108]:

```
df.groupby('ReleaseYear')['MovieID'].count().sort_values(ascending=False).
↪to_frame().reset_index().head(20)
```

[108]:

	ReleaseYear	MovieID
0	1999	86833

1	1998	68165
2	1997	65402
3	1995	60754
4	1996	59271
5	1994	52963
6	1993	46245
7	2000	41000
8	1992	37866
9	1986	30865
10	1990	30317
11	1989	28054
12	1984	27012
13	1985	24794
14	1991	24783
15	1987	24770
16	1988	24020
17	1982	18712
18	1980	16234
19	1981	15346

5. The movie with maximum no. of ratings

[ ]:
```
df.groupby('Title')['Rating'].sum().sort_values(ascending=False).to_frame().
↪reset_index().rename(columns={'Rating':'Ratings_Sum'}).head(5)
```

6. Name the top 3 movies similar to 'Liar Liar' on the item-based approach.

[ ]:
```
# 6. Name the top 3 movies similar to 'Liar Liar' on the item-based approach.
movie_name = 'Liar Liar'
distances, indices = knn.kneighbors(tab[movie_name].values.
↪reshape(1,-1),n_neighbors=4)
for i in range (0, len(distances.flatten())):
  if i == 0 :
    print("The movies close similar to",movie_name,"are \n")
  else:
    print(tab.columns[indices.flatten()[i]],"with a distance␣
↪of",round(distances.flatten()[i],3))
```

Questionnaire:

1. Users of which age group have watched and rated the most number of movies? Ans -> **25-34 age group**
2. Users belonging to which profession have watched and rated the most movies? Ans -> **college/grad student**
3. Most of the users in our dataset who've rated the movies are Male. (T/F) Ans -> **True (Male are most- 751,679)**
4. Most of the movies present on our dataset were released in which decade?

- 70s **b. 90s** c. 50s d.80s Ans -> **90s**

5. The movie with maximum no. of ratings is _____. Ans ->**American Beauty**
6. Name the top 3 movies similar to 'Liar Liar' on the item-based approach. Ans ->

- **Mrs. Doubtfire** with a distance of 0.443
- **Ace Ventura: Pet Detective** with a distance of 0.483
- **Dumb & Dumber** with a distance of 0.487

7. On the basis of approach, Collaborative Filtering methods can be classified into **memory-based** and **model-based.** -> Memory based colab filtering has user-user and item-item whereas, model-based has Matrix Factorization
8. Pearson Correlation ranges between **-1 to +1** whereas, Cosine Similarity belongs to the interval between **0 to 1**
9. Mention the RMSE and MAPE that you got while evaluating the Matrix Factorization model. Ans ->

- **RMSE -86 %**
- **MAPE -> 26 %**

10. Give the sparse 'row' matrix representation for the following dense matrix -

[[1 0] [3 7]]

Ans - CSR can be represented as - (0,0)1 - (0,1)0 - (1,0)3 - (1,1)7

[ ]:

```

```