Import Libraries

```
import pandas as pd
import numpy as np
import matplotlib as mpl
import matplotlib.pyplot as plt
import seaborn as sns
from collections import defaultdict
from scipy import sparse
from scipy.stats import pearsonr
from sklearn.metrics.pairwise import cosine_similarity
from sklearn.neighbors import NearestNeighbors
import warnings
from cmfrec import CMF
from sklearn.metrics import mean_absolute_percentage_error
from sklearn.metrics import mean_squared_error
warnings.simplefilter('ignore')
pd.set_option("display.max_columns", None)
pd.options.display.float_format = '{:.2f}'.format
sns.set_style('white')
```

Data Formatting

```
movies = pd.read_fwf('zee-movies.dat', encoding='ISO-8859-1')
print(movies.shape)
movies.head()
```

```
(3883, 3)
```


```
{"summary":"{\n  \"name\": \"movies\",\n  \"rows\": 3883,\n
\"fields\": [\n    {\n      \"column\": \"Movie ID::Title::Genres\",\n
\"properties\": {\n        \"dtype\": \"string\",\n
\"num_unique_values\": 3883,\n        \"samples\": [\n
\"1365::Ridicule (1996)::Drama\",\n          \"2706::American Pie
(1999)::Comedy\",\n          \"3667::Rent-A-Cop (1988)::Action|
Comedy\"\n        ],\n        \"semantic_type\": \"\",\n
\"description\": \"\"\n      }\n    },\n    {\n      \"column\":
\"Unnamed: 1\",\n      \"properties\": {\n        \"dtype\":
\"category\",\n        \"num_unique_values\": 73,\n
\"samples\": [\n          \"|Sci\",\n          \"71):\",\n
\"er,\"\n        ],\n        \"semantic_type\": \"\",\n
\"description\": \"\"\n      }\n    },\n    {\n      \"column\":
\"Unnamed: 2\",\n      \"properties\": {\n        \"dtype\":
\"category\",\n        \"num_unique_values\": 45,\n
\"samples\": [\n          \"Children's|Fan\",\n
\"(1975)::Comed\",\n          \"ler\"\n        ],\n
```

\"semantic_type\": \"\",\n   \"description\": \"\"\n   }\n   }\n]\n}","type":"dataframe","variable_name":"movies"}

```
movies.drop(columns=['Unnamed: 1', 'Unnamed: 2'], axis=1,
inplace=True)

delimiter = '::'
movies = movies['Movie ID::Title::Genres'].str.split(delimiter,
expand=True)
movies.columns = ['Movie ID', 'Title', 'Genres']

movies.rename(columns={'Movie ID':'MovieID'}, inplace=True)
movies1=movies.copy()
movies.head()
```

{"summary":"{\n \"name\": \"movies\",\n \"rows\": 3883,\n \"fields\": [\n   {\n   \"column\": \"MovieID\",\n   \"properties\": {\n   \"dtype\": \"string\",\n   \"num_unique_values\": 3883,\n   \"samples\": [\n   \"1365\",\n   \"2706\",\n   \"3667\"\n   ],\n   \"semantic_type\": \"\",\n   \"description\": \"\"\n   }\n   },\n   {\n   \"column\": \"Title\",\n   \"properties\": {\n   \"dtype\": \"string\",\n   \"num_unique_values\": 3883,\n   \"samples\": [\n   \"Ridicule (1996)\",\n   \"American Pie (1999)\",\n   \"Rent-A-Cop (1988)\"\n   ],\n   \"semantic_type\": \"\",\n   \"description\": \"\"\n   }\n   },\n   {\n   \"column\": \"Genres\",\n   \"properties\": {\n   \"dtype\": \"category\",\n   \"num_unique_values\": 360,\n   \"samples\": [\n   \"Action|Thriller|War\",\n   \"Crime\",\n   \"Action|Adventure|Sci-Fi|Thriller|War\"\n   ],\n   \"semantic_type\": \"\",\n   \"description\": \"\"\n   }\n   }\n]\n}","type":"dataframe","variable_name":"movies"}

```
ratings = pd.read_fwf('zee-ratings.dat', encoding='ISO-8859-1')

ratings =
ratings['UserID::MovieID::Rating::Timestamp'].str.split(delimiter,
expand=True)
ratings.columns = ['UserID', 'MovieID', 'Rating', 'Timestamp']

print(ratings.shape)
ratings1=ratings.copy()
ratings.head()
```

```
(1000209, 4)
```

{"type":"dataframe","variable_name":"ratings"}

```
users = pd.read_fwf('zee-users.dat', encoding='ISO-8859-1')
```

```
users = users['UserID::Gender::Age::Occupation::Zip-
code'].str.split(delimiter, expand=True)
users.columns = ['UserID', 'Gender', 'Age', 'Occupation', 'Zip-code']
users1=users.copy()

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

users.replace({'Occupation':{'0': "other",
                             '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)

print(users.shape)
users.head()
```

```
(6040, 5)
```

{"summary":"{\n \"name\": \"users\",\n \"rows\": 6040,\n \"fields\": [\n   {\n   \"column\": \"UserID\",\n   \"properties\": {\n   \"dtype\": \"string\",\n   \"num_unique_values\": 6040,\n   \"samples\": [\n   \"5530\",\n   \"711\",\n   \"4924\"\n   ],\n   \"semantic_type\": \"\",\n   \"description\": \"\"\n   }\n   },\n   {\n   \"column\": \"Gender\",\n   \"properties\": {\n   \"dtype\": \"category\",\n   \"num_unique_values\": 2,\n   \"samples\": [\n   \"M\",\n   \"F\"\n   ],\n   \"semantic_type\": \"\",\n   \"description\": \"\"\n

```
}\n    },\n    {\n      \"column\": \"Age\",\n      \"properties\": {\n        \"dtype\": \"category\",\n        \"num_unique_values\": 7,\n        \"samples\": [\n          \"Under 18\",\n          \"56 Above\"\n        ],\n        \"semantic_type\": \"\",\n        \"description\": \"\"\n      }\n    },\n    {\n      \"column\": \"Occupation\",\n      \"properties\": {\n        \"dtype\": \"category\",\n        \"num_unique_values\": 21,\n        \"samples\": [\n          \"k-12 student\",\n          \"tradesman/craftsman\"\n        ],\n        \"semantic_type\": \"\",\n        \"description\": \"\"\n      }\n    },\n    {\n      \"column\": \"Zip-code\",\n      \"properties\": {\n        \"dtype\": \"string\",\n        \"num_unique_values\": 3439,\n        \"samples\": [\n          \"02865\",\n          \"43213\"\n        ],\n        \"semantic_type\": \"\",\n        \"description\": \"\"\n      }\n    }\n  ]\n}","type":"dataframe","variable_name":"users"}
```


Exploratory Data Analysis

```
movies['Year'] = movies.Title.str.extract('(\(\d\d\d\d\))',expand=False)

movies['Year'] = movies.Year.str.extract('(\d\d\d\d)',expand=False)

movies['Title'] = movies.Title.str.replace('(\(\d\d\d\d\))', '')

movies['Title'] = movies['Title'].apply(lambda x: x.strip())
movies.head()
```


```
{"summary":"{\n  \"name\": \"movies\",\n  \"rows\": 3883,\n  \"fields\": [\n    {\n      \"column\": \"MovieID\",\n      \"properties\": {\n        \"dtype\": \"string\",\n        \"num_unique_values\": 3883,\n        \"samples\": [\n          \"1365\",\n          \"2706\",\n          \"3667\"\n        ],\n        \"semantic_type\": \"\",\n        \"description\": \"\"\n      }\n    },\n    {\n      \"column\": \"Title\",\n      \"properties\": {\n        \"dtype\": \"string\",\n        \"num_unique_values\": 3883,\n        \"samples\": [\n          \"Ridicule (1996)\",\n          \"American Pie (1999)\",\n          \"Rent-A-Cop (1988)\"\n        ],\n        \"semantic_type\": \"\",\n        \"description\": \"\"\n      }\n    },\n    {\n      \"column\": \"Genres\",\n      \"properties\": {\n        \"dtype\": \"category\",\n        \"num_unique_values\": 360,\n        \"samples\": [\n          \"Action|Thriller|War\",\n          \"Crime\",\n          \"Action|Adventure|Sci-Fi|Thriller|War\"\n        ],\n        \"semantic_type\": \"\",\n        \"description\": \"\"\n      }\n    },\n    {\n      \"column\": \"Year\",\n      \"properties\": {\n        \"dtype\": \"object\",\n        \"num_unique_values\": 81,\n        \"samples\": [\n          \"1948\",\n          \"1995\",\n          \"1960\"\n        ],\n        \"semantic_type\": \"\",\n
```

\"description\": \"\"\n   }\n   }\n]\
n}","type":"dataframe","variable_name":"movies"}

```
dfmov = movies.copy()
dfmov.dropna(inplace=True)
dfmov.Genres = dfmov.Genres.str.split('|')
dfmov['Genres'] = dfmov['Genres'].apply(lambda x: [i for i in x if i!
='A' and i!='D' and i!= 'F' and i!='C' and i!='M' and i!= 'W' and i!=
' '])
for i in dfmov['Genres']:
    for j in range(len(i)):
        if i[j] == 'Ro' or i[j] == 'Rom' or i[j] == 'Roman' or i[j] ==
'R' or i[j] == 'Roma':
            i[j] = 'Romance'
        elif i[j] == 'Chil' or i[j] == 'Childre' or i[j] == 'Childr'
or i[j] == "Children'" or i[j] =='Children' or i[j] =='Chi':
            i[j] = "Children's"
        elif i[j] == 'Fantas' or i[j] == 'Fant':
            i[j] = 'Fantasy'
        elif i[j] == 'Dr' or i[j] == 'Dram':
            i[j] = 'Drama'
        elif i[j] == 'Documenta'or i[j] == 'Docu' or i[j] ==
'Document' or i[j] == 'Documen':
            i[j] = 'Documentary'
        elif i[j] == 'Wester'or i[j] == 'We':
            i[j] = 'Western'
        elif i[j] == 'Animati':
            i[j] = 'Animation'
        elif i[j] == 'Come'or i[j] == 'Comed' or i[j] == 'Com':
            i[j] = 'Comedy'
        elif i[j] == 'Sci-F'or i[j] == 'S' or i[j] == 'Sci-' or i[j]
== 'Sci':
            i[j] = 'Sci-Fi'
        elif i[j] == 'Adv'or i[j] == 'Adventu' or i[j] == 'Adventur'
or i[j] == 'Advent':
            i[j] = 'Adventure'
        elif i[j] == 'Horro'or i[j] == 'Horr':
            i[j] = 'Horror'
        elif i[j] == 'Th'or i[j] == 'Thri' or i[j] == 'Thrille':
            i[j] = 'Thriller'
        elif i[j] == 'Acti':
            i[j] = 'Action'
        elif i[j] == 'Wa':
            i[j] = 'War'
        elif i[j] == 'Music':
            i[j] = 'Musical'
dfmov.head()
```

{"summary":"{\n \"name\": \"dfmov\",\n \"rows\": 3858,\n
\"fields\": [\n   {\n   \"column\": \"MovieID\",\n

\"properties\": {\n   \"dtype\": \"string\",\n
\"num_unique_values\": 3858,\n   \"samples\": [\n
\"1927\",\n   \"1285\",\n   \"2746\"\n   ],\n
\"semantic_type\": \"\",\n   \"description\": \"\"\n   }\
n   },\n   {\n   \"column\": \"Title\",\n   \"properties\": {\
n   \"dtype\": \"string\",\n   \"num_unique_values\": 3858,\
n   \"samples\": [\n   \"All Quiet on the Western Front
(1930)\",\n   \"Heathers (1989)\",\n   \"Little Shop of
Horrors (1986)\"\n   ],\n   \"semantic_type\": \"\",\n
\"description\": \"\"\n   }\n   },\n   {\n   \"column\":
\"Genres\",\n   \"properties\": {\n   \"dtype\": \"object\",\n
\"semantic_type\": \"\",\n   \"description\": \"\"\n   }\
n   },\n   {\n   \"column\": \"Year\",\n   \"properties\": {\n
\"dtype\": \"object\",\n   \"num_unique_values\": 81,\n
\"samples\": [\n   \"1943\",\n   \"1995\",\n
\"1955\"\n   ],\n   \"semantic_type\": \"\",\n
\"description\": \"\"\n   }\n   }\n]\
n}","type":"dataframe","variable_name":"dfmov"}

Merge all above dataframes

```
df_1 = pd.merge(dfmov, ratings, how='inner', on='MovieID')
df_1.head()
```


{"type":"dataframe","variable_name":"df_1"}


```
data = pd.merge(df_1, users, how='inner', on='UserID')
data.head()
```


{"type":"dataframe","variable_name":"data"}


```
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 996144 entries, 0 to 996143
Data columns (total 11 columns):
 #   Column      Non-Null Count   Dtype 
---  ------      --------------   ----- 
 0   MovieID     996144 non-null  object
 1   Title       996144 non-null  object
 2   Genres      996144 non-null  object
 3   Year        996144 non-null  object
 4   UserID      996144 non-null  object
 5   Rating      996144 non-null  object
 6   Timestamp   996144 non-null  object
 7   Gender      996144 non-null  object
 8   Age         996144 non-null  object
 9   Occupation  996144 non-null  object
 10  Zip-code    996144 non-null  object
```

```
dtypes: object(11)
memory usage: 83.6+ MB
```

Missing Values

```python
missing_value = pd.DataFrame({
    'Missing Value': data.isnull().sum(),
    'Percentage': (data.isnull().sum() / len(data))*100
})

missing_value.sort_values(by='Percentage', ascending=False)
```


```
{"summary":"{\n  \"name\": \"missing_value\",\n  \"rows\": 11,\n
\"fields\": [\n    {\n      \"column\": \"Missing Value\",\n
\"properties\": {\n        \"dtype\": \"number\",\n        \"std\":
0,\n        \"min\": 0,\n        \"max\": 0,\n
\"num_unique_values\": 1,\n        \"samples\": [\n          0\n
],\n        \"semantic_type\": \"\",\n        \"description\": \"\"\n
}\n    },\n    {\n      \"column\": \"Percentage\",\n
\"properties\": {\n        \"dtype\": \"number\",\n        \"std\":
0.0,\n        \"min\": 0.0,\n        \"max\": 0.0,\n
\"num_unique_values\": 1,\n        \"samples\": [\n          0.0\n
],\n        \"semantic_type\": \"\",\n        \"description\": \"\"\n
}\n    }\n  ]\n}","type":"dataframe"}
```


Feature Engineering

```python
data['Datetime'] = pd.to_datetime(data['Timestamp'], unit='s')
data['Year']=data['Year'].astype('int32')
data['Rating']=data['Rating'].astype('int32')

data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 996144 entries, 0 to 996143
Data columns (total 12 columns):
 #   Column      Non-Null Count   Dtype
---  ------      --------------   -----
 0   MovieID     996144 non-null  object
 1   Title       996144 non-null  object
 2   Genres      996144 non-null  object
 3   Year        996144 non-null  int32
 4   UserID      996144 non-null  object
 5   Rating      996144 non-null  int32
 6   Timestamp   996144 non-null  object
 7   Gender      996144 non-null  object
 8   Age         996144 non-null  object
 9   Occupation  996144 non-null  object
 10  Zip-code    996144 non-null  object
 11  Datetime    996144 non-null  datetime64[ns]
```

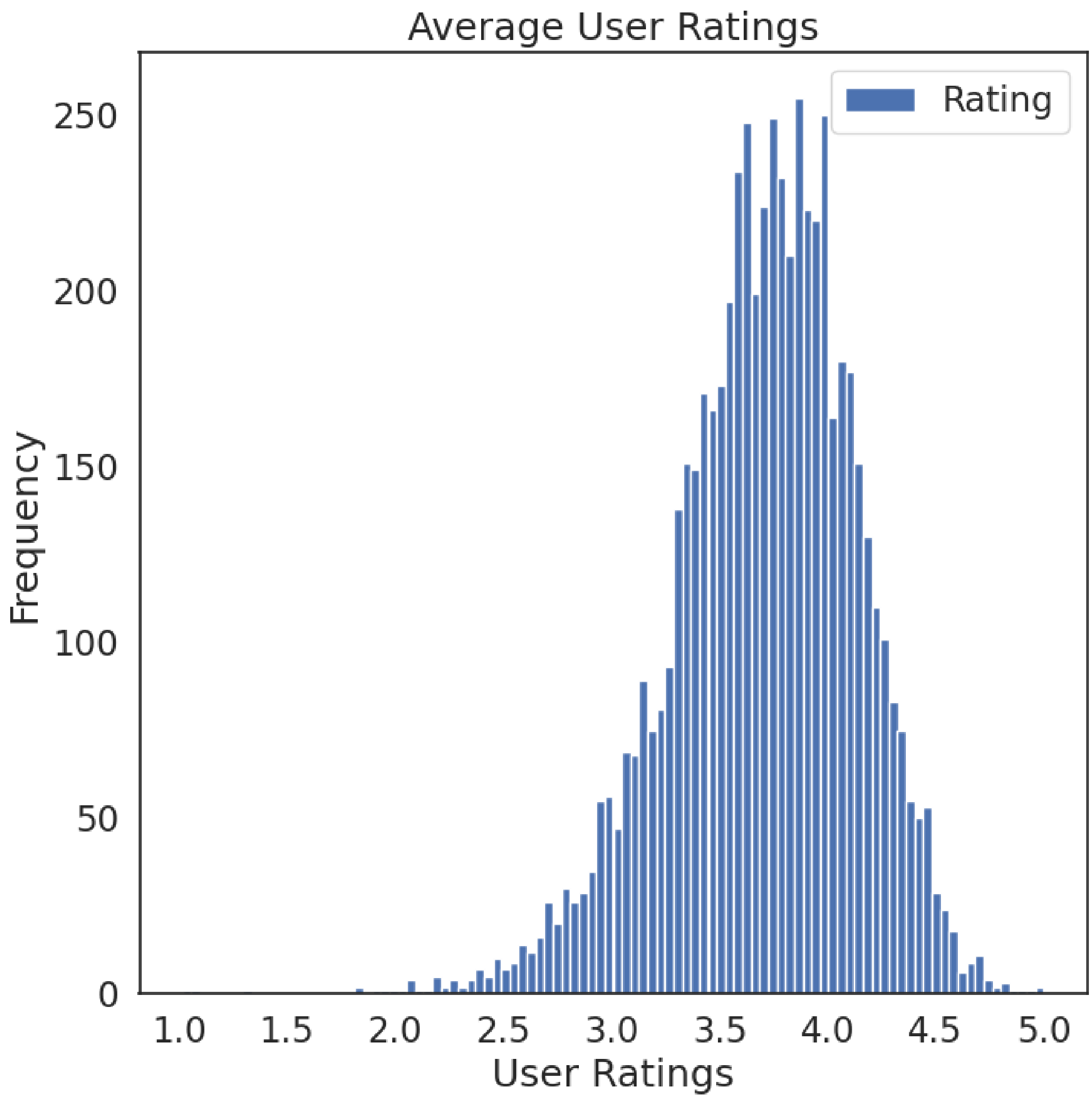
```
dtypes: datetime64[ns](1), int32(2), object(9)
memory usage: 83.6+ MB

bins = [1919, 1929, 1939, 1949, 1959, 1969, 1979, 1989, 2000]
labels = ['20s', '30s', '40s', '50s', '60s', '70s', '80s', '90s']
data['ReleaseDec'] = pd.cut(data['Year'], bins=bins, labels=labels)

data.head()
```

{"type":"dataframe","variable_name":"data"}

Understanding the Dataset

Average User Ratings

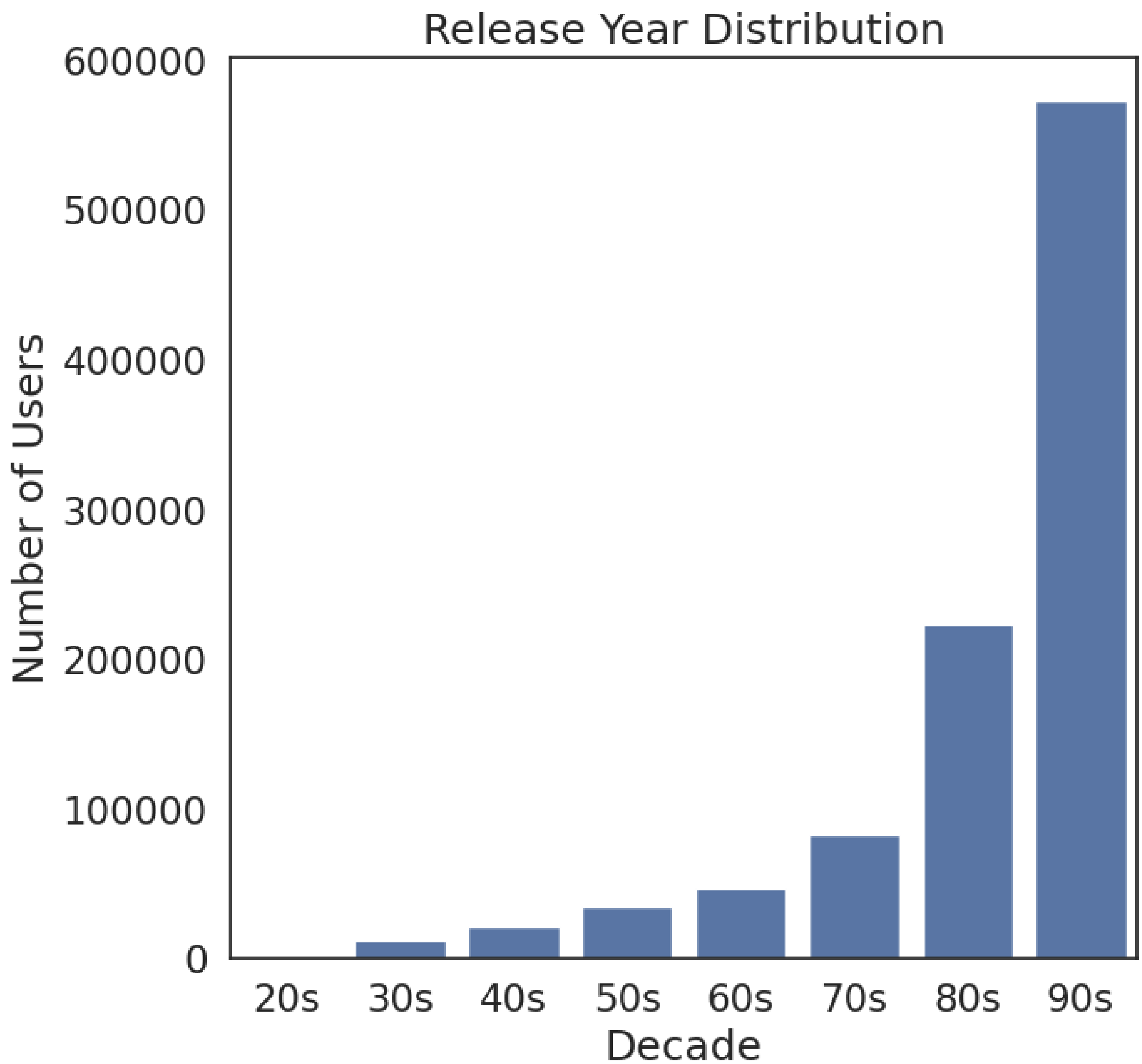
```
user_ratings =data[['UserID','Rating']].groupby('UserID').mean()

fig = plt.figure(figsize = (8,8))
user_ratings.plot(kind = 'hist', bins = 100, figsize = (8,8))
plt.plot()
plt.xlabel('User Ratings')
plt.title('Average User Ratings')
plt.ylabel('Frequency')

Text(0, 0.5, 'Frequency')

<Figure size 800x800 with 0 Axes>
```



Movies are rated 3.5–4 by average, users are less likely to give a film a rating lower than 3 if they didn't enjoy it.

No.of movies by Release year.

```
plt.figure(figsize=(7, 7))
sns.countplot(x='ReleaseDec', data=data)
plt.title('Release Year Distribution')
plt.xlabel('Decade')
plt.ylabel('Number of Users')
plt.show()
```



Most of the movies present in the dataset were released in the year 90s.

```
l = dfmov.Genres.iloc[:5]

pd.get_dummies(l.apply(pd.Series).stack()).groupby(level=0).sum()
```

{"summary":"{\n \"name\": \"pd\",\n \"rows\": 5,\n \"fields\": [\n
{\n   \"column\": \"Adventure\",\n   \"properties\": {\n
\"dtype\": \"number\",\n   \"std\": 0,\n   \"min\": 0,\n
\"max\": 1,\n   \"num_unique_values\": 2,\n   \"samples\":
[\n   1,\n   0\n   ],\n   \"semantic_type\":
\"\",\n   \"description\": \"\"\n   }\n   },\n   {\n
\"column\": \"Animation\",\n   \"properties\": {\n
\"dtype\": \"number\",\n   \"std\": 0,\n   \"min\": 0,\n
\"max\": 1,\n   \"num_unique_values\": 2,\n   \"samples\":

[\n   0,\n   1\n   ],\n   \"semantic_type\": \"\",\n   \"description\": \"\"\n   }\n   },\n   {\n   \"column\": \"Children's\",\n   \"properties\": {\n   \"dtype\": \"number\",\n   \"std\": 0,\n   \"min\": 0,\n   \"max\": 1,\n   \"num_unique_values\": 2,\n   \"samples\": [\n   0,\n   1\n   ],\n   \"semantic_type\": \"\",\n   \"description\": \"\"\n   }\n   },\n   {\n   \"column\": \"Comedy\",\n   \"properties\": {\n   \"dtype\": \"number\",\n   \"std\": 0,\n   \"min\": 0,\n   \"max\": 1,\n   \"num_unique_values\": 2,\n   \"samples\": [\n   0,\n   1\n   ],\n   \"semantic_type\": \"\",\n   \"description\": \"\"\n   }\n   },\n   {\n   \"column\": \"Drama\",\n   \"properties\": {\n   \"dtype\": \"number\",\n   \"std\": 0,\n   \"min\": 0,\n   \"max\": 1,\n   \"num_unique_values\": 2,\n   \"samples\": [\n   1,\n   0\n   ],\n   \"semantic_type\": \"\",\n   \"description\": \"\"\n   }\n   },\n   {\n   \"column\": \"Fantasy\",\n   \"properties\": {\n   \"dtype\": \"number\",\n   \"std\": 0,\n   \"min\": 0,\n   \"max\": 1,\n   \"num_unique_values\": 2,\n   \"samples\": [\n   1,\n   0\n   ],\n   \"semantic_type\": \"\",\n   \"description\": \"\"\n   }\n   },\n   {\n   \"column\": \"Romance\",\n   \"properties\": {\n   \"dtype\": \"number\",\n   \"std\": 0,\n   \"min\": 0,\n   \"max\": 1,\n   \"num_unique_values\": 2,\n   \"samples\": [\n   1,\n   0\n   ],\n   \"semantic_type\": \"\",\n   \"description\": \"\"\n   }\n   }\n]\n}","type":"dataframe"}

```
pd.Series(l.iloc[0])
```

```
0     Animation
1    Children's
2        Comedy
dtype: object
```

```
dfmov.head(2)
```

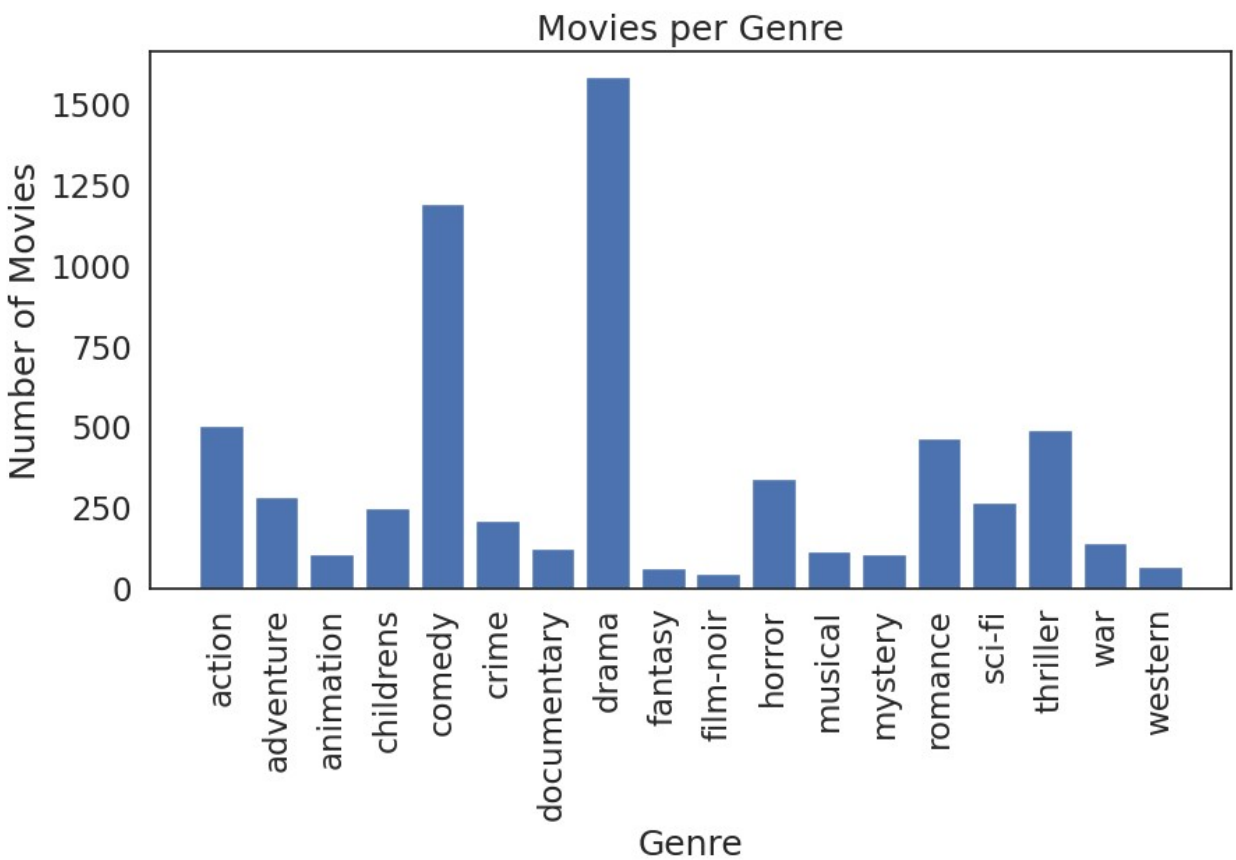
{"summary":"{\n \"name\": \"dfmov\",\n \"rows\": 3858,\n \"fields\": [\n   {\n   \"column\": \"MovieID\",\n   \"properties\": {\n   \"dtype\": \"string\",\n   \"num_unique_values\": 3858,\n   \"samples\": [\n   \"1927\",\n   \"1285\",\n   \"2746\"\n   ],\n   \"semantic_type\": \"\",\n   \"description\": \"\"\n   }\n   },\n   {\n   \"column\": \"Title\",\n   \"properties\": {\n   \"dtype\": \"string\",\n   \"num_unique_values\": 3858,\n   \"samples\": [\n   \"All Quiet on the Western Front (1930)\",\n   \"Heathers (1989)\",\n   \"Little Shop of Horrors (1986)\"\n   ],\n   \"semantic_type\": \"\",\n   \"description\": \"\"\n   }\n   },\n   {\n   \"column\":

```
\"Genres\",\n      \"properties\": {\n        \"dtype\": \"object\",\n
\"semantic_type\": \"\",\n        \"description\": \"\"\n      }\
n    },\n    {\n      \"column\": \"Year\",\n      \"properties\": {\n
\"dtype\": \"object\",\n        \"num_unique_values\": 81,\n
\"samples\": [\n          \"1943\",\n          \"1995\",\n
\"1955\"\n        ],\n        \"semantic_type\": \"\",\n
\"description\": \"\"\n      }\n    }\n  ]\
n}","type":"dataframe","variable_name":"dfmov"}
```


Top 10 Genres based on movies count

```
genres_df = (
    pd.get_dummies(dfmov['Genres'].apply(pd.Series).stack())
    .groupby(level=0)
    .sum()
)

test = genres_df.iloc[:,0:].sum()
test=test.iloc[1:]
print(test)

Action          501
Adventure       282
Animation       104
Children's      249
Comedy         1189
Crime           210
Documentary     124
Drama          1582
Fantasy          62
Film-Noir        44
Horror          340
Musical         113
Mystery         105
Romance         462
Sci-Fi          265
Thriller        488
War             139
Western          68
dtype: int64

print(type(pd.to_numeric(test)))
print(type(test.to_numpy().reshape(18,)[0]))

test2 = test.to_numpy().reshape(18,)

<class 'pandas.core.series.Series'>
<class 'numpy.int64'>
```

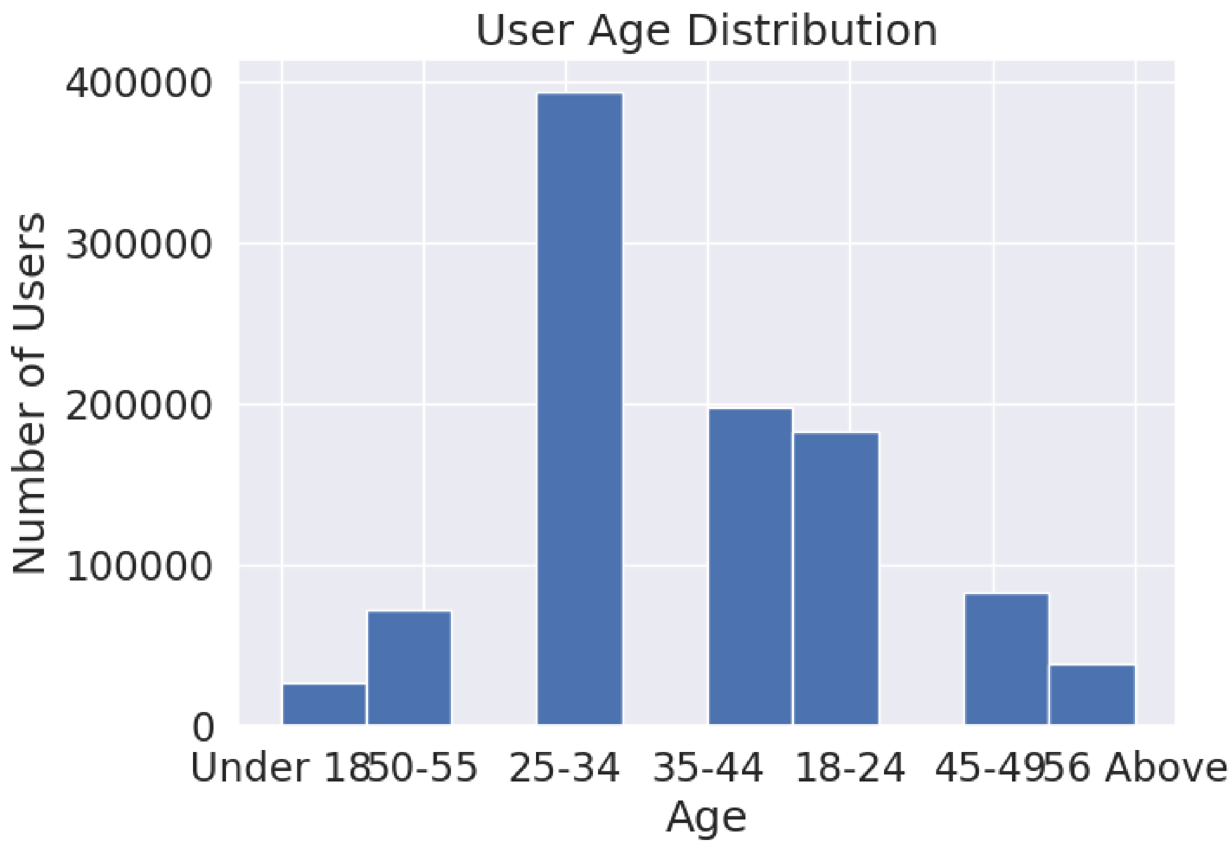
```
genre_list=['action', 'adventure','animation', 'childrens', 'comedy',
'crime', 'documentary', 'drama','fantasy', 'film-noir', 'horror',
'musical', 'mystery', 'romance', 'sci-fi','thriller', 'war',
'western']
x = np.arange(18)
plt.figure(figsize = (10,5))
plt.bar(x, test2)
plt.xticks(x, genre_list, rotation = 'vertical')
plt.xlabel('Genre')
plt.ylabel('Number of Movies')
plt.title('Movies per Genre')
sns.set(font_scale=1.5)
plt.show()
```



most the movies in the dataset belongs to Comedy and Drama genres.

Distribution by Age

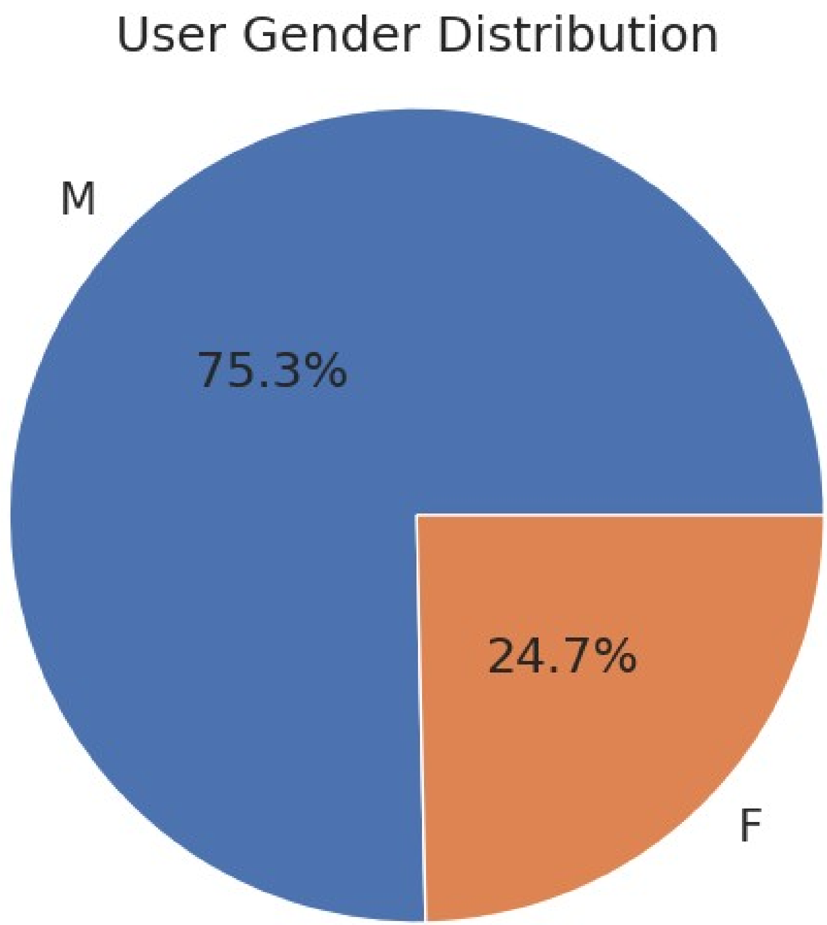
```
data['Age'].hist(figsize=(7, 5))
plt.title('User Age Distribution')
plt.xlabel('Age')
plt.ylabel('Number of Users')
plt.show()
```



25-34 age group have watched and rated the most number of movies

Distribution by Gender

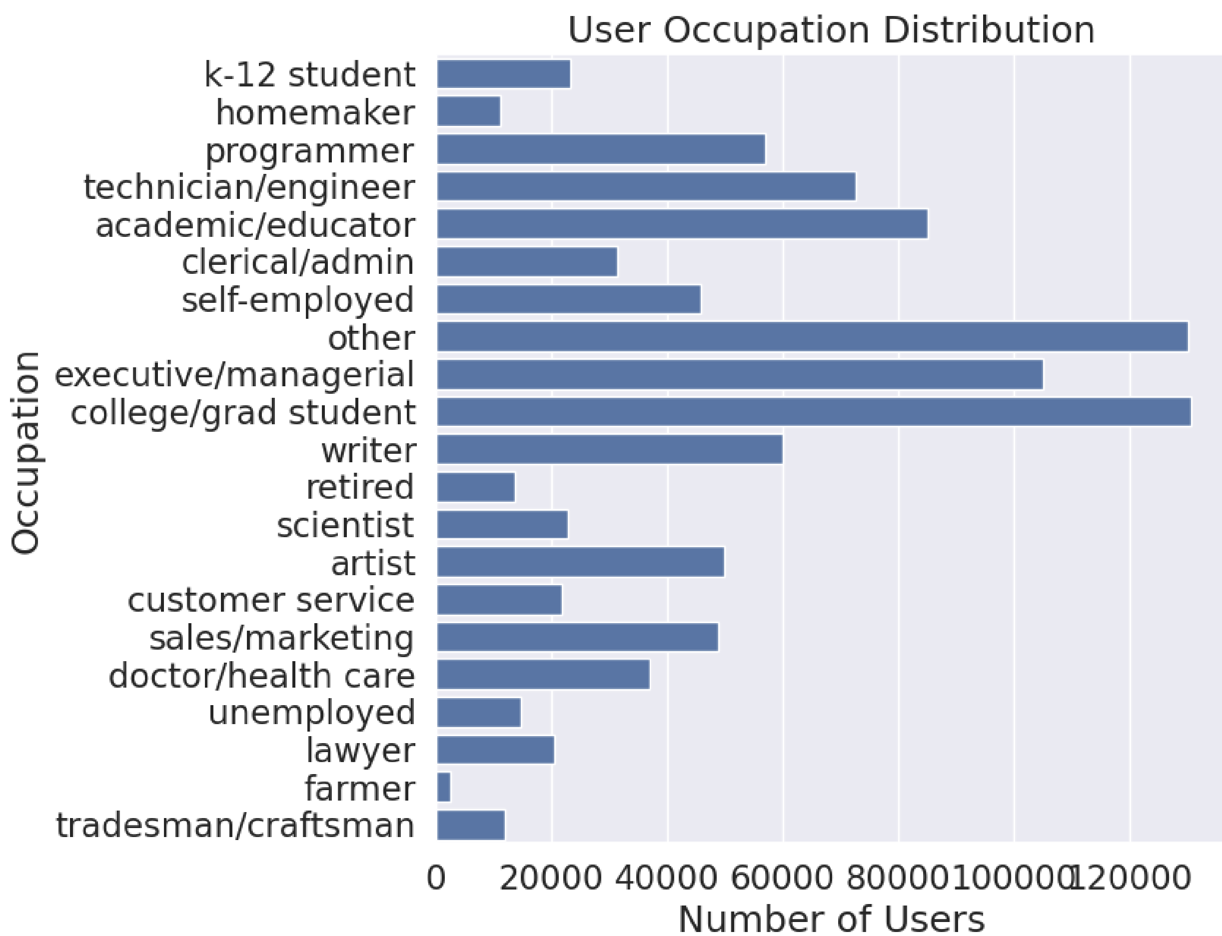
```
x = data['Gender'].value_counts().values
plt.figure(figsize=(7, 6))
plt.pie(x, center=(0, 0), radius=1.5, labels=['M','F'], autopct='%1.1f
%%', pctdistance=0.5)
plt.title('User Gender Distribution')
plt.axis('equal')
plt.show()
```



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

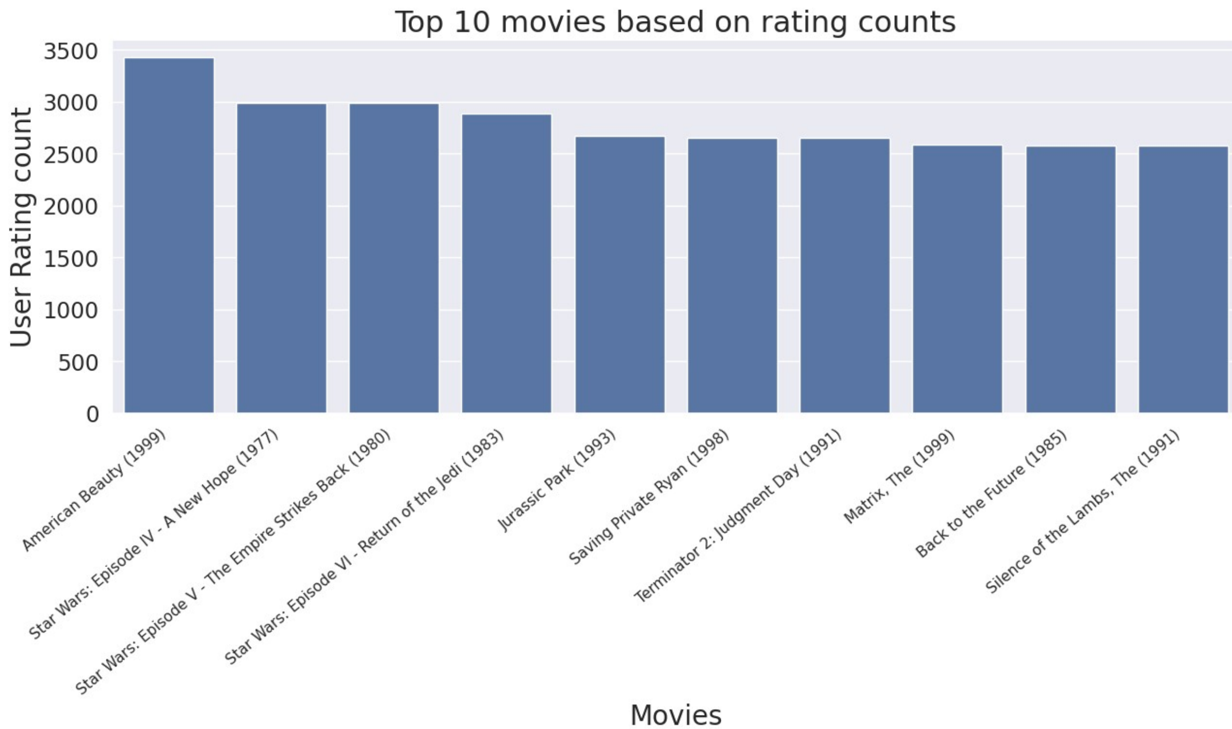
Distribution by Occupation

```
plt.figure(figsize=(7, 7))
sns.countplot(y='Occupation', data=data)
plt.title('User Occupation Distribution')
plt.xlabel('Number of Users')
plt.ylabel('Occupation')
plt.show()
```



users belonging to college/grad student profession have watched and rated the most movies.

```
movies_rating_count = data.groupby(by = ['Title'])
['Rating'].count().reset_index()[['Title', 'Rating']] ## Counting the
ratings based on movies
movies_rating_count.rename(columns = {'Rating':
'totalRatingCount'},inplace=True)

top10_movies=movies_rating_count[['Title',
'totalRatingCount']].sort_values(by = 'totalRatingCount',ascending =
False).head(10)

plt.figure(figsize=(15,5))
ax=sns.barplot(x="Title", y="totalRatingCount", data=top10_movies)
ax.set_xticklabels(ax.get_xticklabels(), fontsize=11, rotation=40,
ha="right")
ax.set_title('Top 10 movies based on rating counts',fontsize = 22)
ax.set_xlabel('Movies',fontsize = 20)
ax.set_ylabel('User Rating count', fontsize = 20)

Text(0, 0.5, 'User Rating count')
```



movie with maximum number of ratings is American Beauty.

Recommendations systems

User-Interaction Matrix

```
matrix = pd.pivot_table(data, index='UserID', columns='Title',
values='Rating', aggfunc='mean')
matrix.fillna(0, inplace=True)

print(matrix.shape)

matrix.head(10)
```

```
(6040, 3682)
```

{"type":"dataframe","variable_name":"matrix"}

```
n_users = data['UserID'].nunique()
n_movies = data['MovieID'].nunique()
sparsity = round(1.0 - data.shape[0] / float( n_users * n_movies), 3)
print('The sparsity level of dataset is ' +  str(sparsity * 100) +
'%')
```

```
The sparsity level of dataset is 95.5%
```

Pearson Correlation

Pearson's Correlation measures the degree of linear relationship between two numeric variables and lies between -1 to +1. It is represented by 'r'.

r=1 means perfect positive correlation

r=-1 means perfect negative correlation

r=0 means no linear correlation (note, it does not mean no correlation)

Item - Based approach

```
data[data['Title']=='Your Friends and Neighbors (1998)']
```

{"repr_error":"0","type":"dataframe"}

```
movie_name='Your Friends and Neighbors (1998)'
movie_rating = matrix[movie_name] # Taking the ratings of that movie
print(movie_rating)
```

```
UserID
1       0.00
10      0.00
100     0.00
1000    0.00
1001    4.00
        ...
995     0.00
996     0.00
997     0.00
998     0.00
999     1.00
Name: Your Friends and Neighbors (1998), Length: 6040, dtype: float64
```

```
similar_movies = matrix.corrwith(movie_rating)

sim_df = pd.DataFrame(similar_movies, columns=['Correlation'])
sim_df.sort_values('Correlation', ascending=False, inplace=True)

sim_df.iloc[1: , :].head()
```

{"summary":"{\n \"name\": \"sim_df\",\n \"rows\": 5,\n \"fields\": [\n   {\n   \"column\": \"Title\",\n   \"properties\": {\n   \"dtype\": \"string\",\n   \"num_unique_values\": 5,\n   \"samples\": [\n   \"Trees Lounge (1996)\",\n   \"Ice Storm, The (1997)\",\n   \"Deconstructing Harry (1997)\"\n   ],\n   \"semantic_type\": \"\",\n   \"description\": \"\"\n   }\n   },\n   {\n   \"column\": \"Correlation\",\n   \"properties\": {\n   \"dtype\": \"number\",\n   \"std\": 0.01866719664621118,\n   \"min\": 0.26680645198760017,\n   \"max\": 0.3137826946788953,\n   \"num_unique_values\": 5,\n   \"samples\": [\n   0.2852697095935018,\n   0.26680645198760017,\n   0.28039653648243523\n   ],\n

\"semantic_type\": \"\",\n   \"description\": \"\"\n   }\n   }\n]\n}","type":"dataframe"}

Cosine Similarty

Cosine similarity is a measure of similarity between two sequences of numbers. Those sequences are viewed as vectors in a higher dimensional space, and the cosine similarity is defined as the cosine of the angle between them

The cosine similarity always belongs to the interval [-1,1].

```
item_sim = cosine_similarity(matrix.T)
item_sim
```

```
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.        ]])
```

```
item_sim.shape
```

```
(3682, 3682)
```

Item-Based Similarity

```
item_sim_matrix = pd.DataFrame(item_sim, index=matrix.columns,
columns=matrix.columns)
item_sim_matrix.head()
```

{"type":"dataframe","variable_name":"item_sim_matrix"}

User-Based Similarity

```
user_sim = cosine_similarity(matrix)
user_sim
```

```
array([[1.        , 0.25586725, 0.12396703, ..., 0.15926709,
0.11935626,
        0.12205855],
       [0.25586725, 1.        , 0.25863269, ..., 0.16071024,
0.13280705,
        0.24681021],
       [0.12396703, 0.25863269, 1.        , ..., 0.20430203,
0.11352239,
        0.30610356],
       ...,
       [0.15926709, 0.16071024, 0.20430203, ..., 1.        ,
0.18657496,
        0.18245166],
       [0.11935626, 0.13280705, 0.11352239, ..., 0.18657496, 1.
,
        0.10797727],
       [0.12205855, 0.24681021, 0.30610356, ..., 0.18245166,
0.10797727,
        1.        ]])
```

```
user_sim_matrix = pd.DataFrame(user_sim, index=matrix.index,
columns=matrix.index)
user_sim_matrix.head()
```

{"type":"dataframe","variable_name":"user_sim_matrix"}

Nearest Neighbors

```
model_knn = NearestNeighbors(metric='cosine')
model_knn.fit(matrix.T)
```

```
NearestNeighbors(metric='cosine')
```

```
distances, indices = model_knn.kneighbors(matrix.T, n_neighbors= 6)
```

```
result = pd.DataFrame(indices, columns=['Title1', 'Title2', 'Title3',
'Title4', 'Title5','Title6'])
result.head()
```

{"summary":"{\n \"name\": \"result\",\n \"rows\": 3682,\n
\"fields\": [\n   {\n   \"column\": \"Title1\",\n
\"properties\": {\n   \"dtype\": \"number\",\n   \"std\":
1067,\n   \"min\": 0,\n   \"max\": 3681,\n
\"num_unique_values\": 3649,\n   \"samples\": [\n   1408,\
n   3630,\n   3679\n   ],\n
\"semantic_type\": \"\",\n   \"description\": \"\"\n   }\
n   },\n   {\n   \"column\": \"Title2\",\n   \"properties\":
{\n   \"dtype\": \"number\",\n   \"std\": 1062,\n
\"min\": 1,\n   \"max\": 3680,\n   \"num_unique_values\":
1840,\n   \"samples\": [\n   3106,\n   2343,\n

3540\n   ],\n   \"semantic_type\": \"\",\n
\"description\": \"\"\n   }\n   },\n   {\n   \"column\":
\"Title3\",\n   \"properties\": {\n   \"dtype\": \"number\",\n
\"std\": 1052,\n   \"min\": 0,\n   \"max\": 3679,\n
\"num_unique_values\": 1793,\n   \"samples\": [\n   266,\n
2138,\n   789\n   ],\n   \"semantic_type\": \"\",\n
\"description\": \"\"\n   }\n   },\n   {\n   \"column\":
\"Title4\",\n   \"properties\": {\n   \"dtype\": \"number\",\n
\"std\": 1068,\n   \"min\": 0,\n   \"max\": 3679,\n
\"num_unique_values\": 1796,\n   \"samples\": [\n   3519,\
n   2697,\n   3233\n   ],\n
\"semantic_type\": \"\",\n   \"description\": \"\"\n   }\
n   },\n   {\n   \"column\": \"Title5\",\n   \"properties\":
{\n   \"dtype\": \"number\",\n   \"std\": 1052,\n
\"min\": 4,\n   \"max\": 3679,\n   \"num_unique_values\":
1803,\n   \"samples\": [\n   2320,\n   1713,\n
3595\n   ],\n   \"semantic_type\": \"\",\n
\"description\": \"\"\n   }\n   },\n   {\n   \"column\":
\"Title6\",\n   \"properties\": {\n   \"dtype\": \"number\",\n
\"std\": 1046,\n   \"min\": 0,\n   \"max\": 3679,\n
\"num_unique_values\": 1803,\n   \"samples\": [\n   3111,\
n   1958,\n   389\n   ],\n
\"semantic_type\": \"\",\n   \"description\": \"\"\n   }\
n   }\n]\n}","type":"dataframe","variable_name":"result"}

```
result2 = result.copy()
for i in range(1, 7):
    mov = pd.DataFrame(matrix.T.index).reset_index()
    mov = mov.rename(columns={'index':f'Title{i}'})
    result2 = pd.merge(result2, mov, on=[f'Title{i}'], how='left')
    result2 = result2.drop(f'Title{i}', axis=1)
    result2 = result2.rename(columns={'Title':f'Title{i}'})
result2.head()
```

{"summary":"{\n \"name\": \"result2\",\n \"rows\": 3682,\n
\"fields\": [\n   {\n   \"column\": \"Title1\",\n
\"properties\": {\n   \"dtype\": \"string\",\n
\"num_unique_values\": 3649,\n   \"samples\": [\n
\"Greatest Show on Earth, The (1952)\",\n   \"Withnail and I
(1987)\",\n   \"Zero Kelvin (Kj\\u00e6rlighetens kj\\u00f8tere)
(1995)\"\n   ],\n   \"semantic_type\": \"\",\n
\"description\": \"\"\n   }\n   },\n   {\n   \"column\":
\"Title2\",\n   \"properties\": {\n   \"dtype\":
\"category\",\n   \"num_unique_values\": 1840,\n
\"samples\": [\n   \"Splendor in the Grass (1961)\",\n
\"Niagara, Niagara (1997)\",\n   \"Waterboy, The (1998)\"\n
],\n   \"semantic_type\": \"\",\n   \"description\": \"\"\n
}\n   },\n   {\n   \"column\": \"Title3\",\n   \"properties\":
{\n   \"dtype\": \"category\",\n   \"num_unique_values\":
1793,\n   \"samples\": [\n   \"Bad Moon (1996)\",\n

\"Metroland (1997)\",\n   \"Criminal Lovers (Les Amants Criminels) (1999)\"\n   ],\n   \"semantic_type\": \"\",\n   \"description\": \"\"\n   }\n   },\n   {\n   \"column\": \"Title4\",\n   \"properties\": {\n   \"dtype\": \"category\",\n   \"num_unique_values\": 1796,\n   \"samples\": [\n   \"Waiting for Guffman (1996)\",\n   \"Railroaded! (1947)\",\n   \"T-Men (1947)\"\n   ],\n   \"semantic_type\": \"\",\n   \"description\": \"\"\n   }\n   },\n   {\n   \"column\": \"Title5\",\n   \"properties\": {\n   \"dtype\": \"category\",\n   \"num_unique_values\": 1803,\n   \"samples\": [\n   \"Nell (1994)\",\n   \"JFK (1991)\",\n   \"Who's That Girl? (1987)\"\n   ],\n   \"semantic_type\": \"\",\n   \"description\": \"\"\n   }\n   },\n   {\n   \"column\": \"Title6\",\n   \"properties\": {\n   \"dtype\": \"category\",\n   \"num_unique_values\": 1803,\n   \"samples\": [\n   \"Squanto: A Warrior's Tale (1994)\",\n   \"Logan's Run (1976)\",\n   \"Big One, The (1997)\"\n   ],\n   \"semantic_type\": \"\",\n   \"description\": \"\"\n   }\n   }\n]\n}","type":"dataframe","variable_name":"result2"}

```
movie_name = 'Mad Love (1995)'
result2.loc[result2['Title1']==movie_name]
```

{"summary":"{\n \"name\": \"result2\",\n \"rows\": 1,\n \"fields\": [\n   {\n   \"column\": \"Title1\",\n   \"properties\": {\n   \"dtype\": \"string\",\n   \"num_unique_values\": 1,\n   \"samples\": [\n   \"Mad Love (1995)\"\n   ],\n   \"semantic_type\": \"\",\n   \"description\": \"\"\n   }\n   },\n   {\n   \"column\": \"Title2\",\n   \"properties\": {\n   \"dtype\": \"string\",\n   \"num_unique_values\": 1,\n   \"samples\": [\n   \"To Gillian on Her 37th Birthday (1996)\"\n   ],\n   \"semantic_type\": \"\",\n   \"description\": \"\"\n   }\n   },\n   {\n   \"column\": \"Title3\",\n   \"properties\": {\n   \"dtype\": \"string\",\n   \"num_unique_values\": 1,\n   \"samples\": [\n   \"Music From Another Room (1998)\"\n   ],\n   \"semantic_type\": \"\",\n   \"description\": \"\"\n   }\n   },\n   {\n   \"column\": \"Title4\",\n   \"properties\": {\n   \"dtype\": \"string\",\n   \"num_unique_values\": 1,\n   \"samples\": [\n   \"Now and Then (1995)\"\n   ],\n   \"semantic_type\": \"\",\n   \"description\": \"\"\n   }\n   },\n   {\n   \"column\": \"Title5\",\n   \"properties\": {\n   \"dtype\": \"string\",\n   \"num_unique_values\": 1,\n   \"samples\": [\n   \"Something to Talk About (1995)\"\n   ],\n   \"semantic_type\": \"\",\n   \"description\": \"\"\n   }\n   },\n   {\n   \"column\": \"Title6\",\n   \"properties\": {\n   \"dtype\": \"string\",\n   \"num_unique_values\": 1,\n   \"samples\": [\n   \"Bye Bye, Love (1995)\"\n   ],\n

\"semantic_type\": \"\",\n   \"description\": \"\"\n   }\n   }\n]\n}","type":"dataframe"}

Matrix Factorization

```
rm = data.pivot(index = 'UserID', columns ='MovieID', values =
'Rating').fillna(0)
rm.head()
```

{"type":"dataframe","variable_name":"rm"}

```
user_itm = data[['UserID', 'MovieID', 'Rating']].copy()
user_itm.columns = ['UserId', 'ItemId', 'Rating']
user_itm.head(2)
```

{"type":"dataframe","variable_name":"user_itm"}

```
print(user_itm.shape)
print("No.of Users:",len(user_itm['UserId'].unique()))
print("No.of Items:",len(user_itm['ItemId'].unique()))
```

```
(996144, 3)
No.of Users: 6040
No.of Items: 3682
```

```
model = CMF(method="als", k=4, lambda_=0.1, user_bias=False,
item_bias=False, verbose=False)
model.fit(user_itm)
```

```
Collective matrix factorization model
(explicit-feedback variant)
```

```
model.A_.shape, model.B_.shape
```

```
((6040, 4), (3682, 4))
```

```
user_itm.Rating.mean(), model.glob_mean_
```

```
(np.float64(3.57998542379415), 3.5799853801727295)
```

```
user=cosine_similarity(model.A_)
```

```
user_sim_matrix = pd.DataFrame(user, index=matrix.index,
columns=matrix.index)
user_sim_matrix.head()
```

{"type":"dataframe","variable_name":"user_sim_matrix"}

```
itm=cosine_similarity(model.B_)
```

```
itm_sim_matrix = pd.DataFrame(itm, index=user_itm['ItemId'].unique(),
```

```
columns=user_itm['ItemId'].unique())
itm_sim_matrix.head()
```

{"type":"dataframe","variable_name":"itm_sim_matrix"}

```
movie_name='586'
movie_rating = itm_sim_matrix[movie_name]
print(movie_rating)
```

```
1       0.25
2       0.96
3       0.94
4       0.70
5       0.97
        ...
3948    0.53
3949   -0.38
3950    0.44
3951    0.11
3952    0.45
Name: 586, Length: 3682, dtype: float32
```

```
similar_movies = itm_sim_matrix.corrwith(movie_rating)

sim_df = pd.DataFrame(similar_movies, columns=['Correlation'])
sim_df.sort_values('Correlation', ascending=False, inplace=True)

sim_df.iloc[1: , :].head()
```

{"summary":"{\n \"name\": \"sim_df\",\n \"rows\": 5,\n \"fields\": [\n   {\n   \"column\": \"Correlation\",\n   \"properties\": {\n   \"dtype\": \"number\",\n   \"std\": 0.0010979005942706333,\n   \"min\": 0.9963342542164175,\n   \"max\": 0.9992759377526796,\n   \"num_unique_values\": 5,\n   \"samples\": [\n   0.9978293972277665,\n   0.9963342542164175,\n   0.9971800200987978\n   ],\n   \"semantic_type\": \"\",\n   \"description\": \"\"\n   }\n   }\n]\n}","type":"dataframe"}

```
item_mov = data[['MovieID', 'Title']].copy()
item_mov.drop_duplicates(inplace=True)
item_mov.reset_index(drop=True,inplace=True)

sim_df1= sim_df.copy()
sim_df1.reset_index(inplace=True)
sim_df1.rename(columns = {'index':'MovieID'}, inplace = True)
sim_mov = pd.merge(sim_df1,item_mov,on='MovieID',how='inner')
sim_mov.head(6)
```

{"summary":"{\n \"name\": \"sim_mov\",\n \"rows\": 3682,\n \"fields\": [\n   {\n   \"column\": \"MovieID\",\n

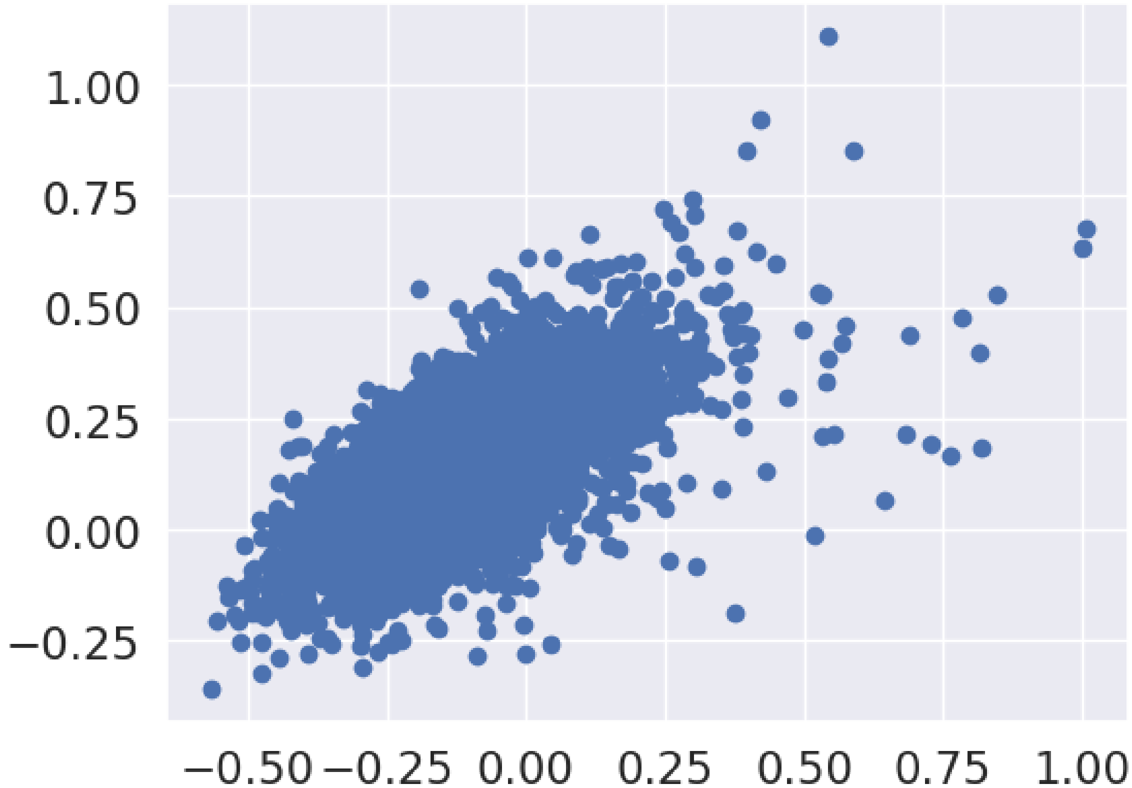
```
\"properties\": {\n        \"dtype\": \"string\",\n        \"num_unique_values\": 3682,\n        \"samples\": [\n          \"2897\",\n          \"3902\",\n          \"2363\"\n        ],\n        \"semantic_type\": \"\",\n        \"description\": \"\"\n      }\n    },\n    {\n      \"column\": \"Correlation\",\n      \"properties\": {\n        \"dtype\": \"number\",\n        \"std\": 0.5356562435625112,\n        \"min\": -0.9630742003292803,\n        \"max\": 1.0,\n        \"num_unique_values\": 3658,\n        \"samples\": [\n          0.8669065868676699,\n          0.5156658550316853,\n          0.8911711923375352\n        ],\n        \"semantic_type\": \"\",\n        \"description\": \"\"\n      }\n    },\n    {\n      \"column\": \"Title\",\n      \"properties\": {\n        \"dtype\": \"string\",\n        \"num_unique_values\": 3682,\n        \"samples\": [\n          \"And the Ship Sails On (E la nave va) (1984)\",\n          \"Goya in Bordeaux (Goya en Bodeos) (1999)\",\n          \"Godzilla (Gojira) (1954)\"\n        ],\n        \"semantic_type\": \"\",\n        \"description\": \"\"\n      }\n    }\n  ]\n}","type":"dataframe","variable_name":"sim_mov"}
```


```
model1 = CMF(method="als", k=2, lambda_=0.1, user_bias=False,
item_bias=False, verbose=False)
model1.fit(user_itm)
```

```
Collective matrix factorization model
(explicit-feedback variant)
```

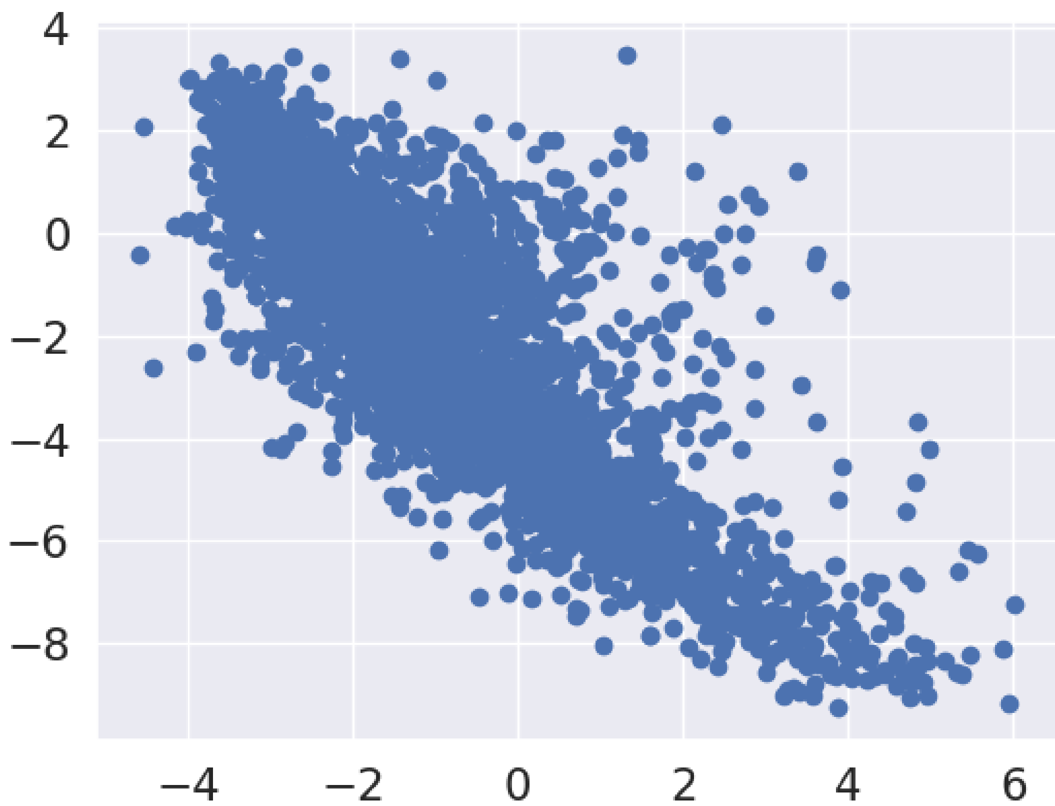
```
plt.scatter(model1.A_[:, 0], model1.A_[:, 1], cmap = 'hot')
```

```
<matplotlib.collections.PathCollection at 0x787709a29d50>
```




```
plt.scatter(model1.B_[:, 0], model1.B_[:, 1], cmap='hot')
```

```
<matplotlib.collections.PathCollection at 0x7876fed59b10>
```



Questionnaire

Users of which age group have watched and rated the most number of movies? :- 25-34 age group

Users belonging to which profession have watched and rated the most movies? :- college/grad student

Most of the users in our dataset who've rated the movies are Male. (T/F):- True

Most of the movies present on our dataset were released in which decade? :- b.90s a.70s b. 90s c. 50s d.80s

The movie with maximum no. of ratings is ___ :- American Beauty

Name the top 3 movies similar to 'Liar Liar' on the item-based approach. :- Mrs. Doubtfire, Ace Ventura: Pet, Detective Dumb & Dumber

On the basis of approach, Collaborative Filtering methods can be classified into Memory-based and Model-based.

Pearson Correlation ranges between -1 to 1 whereas, Cosine Similarity belongs to the interval between -1 to 1

Mention the RMSE and MAPE that you got while evaluating the Matrix Factorization model.:- RMSE:0.701 and MAPE: 0.54

Give the sparse ‘row’ matrix representation for the following dense matrix - [[1 0],[3 7]]

```
from scipy.sparse import csr_matrix

A = np.array([[1,0],[3,7]])

S = csr_matrix(A)
print(S)

<Compressed Sparse Row sparse matrix of dtype 'int64'
	with 3 stored elements and shape (2, 2)>
  Coords	Values
  (0, 0)	1
  (1, 0)	3
  (1, 1)	7
```