PROJECT : Building user-based recommendation model for Amazon.

Description :

The dataset provided contains movie reviews given by Amazon customers. Reviews were given between May 1996 and July 2014.

Data Dictionary :

UserID – 4848 customers who provided a rating for each movie Movie 1 to Movie 206 – 206 movies for which ratings are provided by 4848 distinct users

Data Considerations

- All the users have not watched all the movies and therefore, all movies are not rated. These missing values are represented by NA.
- Ratings are on a scale of -1 to 10 where -1 is the least rating and 10 is the best.

Analysis Task

- Exploratory Data Analysis:

  1. Which movies have maximum views/ratings?
  2. What is the average rating for each movie? Define the top 5 movies with the maximum ratings.
  3. Define the top 5 movies with the least audience.

  - Recommendation Model: Some of the movies hadn't been watched and therefore, are not rated by the users.
  - Netflix would like to take this as an opportunity and build a machine learning recommendation algorithm which provides the ratings for each of the users.

  4. Divide the data into training and test data
  5. Build a recommendation model on training data
  6. Make predictions on the test data

1.Import Necessary Libraries

In [11]:
```
import numpy as np
import pandas as pd
import re
import matplotlib.pyplot as plt
import seaborn as sns
%matplotlib inline
#import surprise
```

2.Load Dataset

In [12]:
```
df=pd.read_csv("Amazon - Movies and TV Ratings.csv")
```

3.Explore Dataset

In [13]:
```
df.head() # Check top 5 record
```

Out[13]:

	user_id	Movie1	Movie2	Movie3	Movie4	Movie5	Movie6	Movie7	Movie8	Movie9	...	Movie197	Movie198	Movie199	M
0	A3R5OBKS7OM2IR	5.0	5.0	NaN	NaN	NaN	NaN	NaN	NaN	NaN	...	NaN	NaN	NaN	
1	AH3QC2PC1VTGP	NaN	NaN	2.0	NaN	NaN	NaN	NaN	NaN	NaN	...	NaN	NaN	NaN	
2	A3LKP6WPMP9UKX	NaN	NaN	NaN	5.0	NaN	NaN	NaN	NaN	NaN	...	NaN	NaN	NaN	
3	AVIY68KEPQ5ZD	NaN	NaN	NaN	5.0	NaN	NaN	NaN	NaN	NaN	...	NaN	NaN	NaN	
4	A1CV1WROP5KTTW	NaN	NaN	NaN	NaN	5.0	NaN	NaN	NaN	NaN	...	NaN	NaN	NaN	

5 rows × 207 columns

In [14]:
```
df.shape  # printing No of rows and columns
```

Out[14]: (4848, 207)

In [15]:
```
df_orginal=df.copy() # Make a copy of original dataset
```

In [16]:
```
df.describe().T # Calculating some statistical data
```

Out[16]:

	count	mean	std	min	25%	50%	75%	max
Movie1	1.0	5.000000	NaN	5.0	5.00	5.0	5.0	5.0
Movie2	1.0	5.000000	NaN	5.0	5.00	5.0	5.0	5.0
Movie3	1.0	2.000000	NaN	2.0	2.00	2.0	2.0	2.0
Movie4	2.0	5.000000	0.000000	5.0	5.00	5.0	5.0	5.0
Movie5	29.0	4.103448	1.496301	1.0	4.00	5.0	5.0	5.0
...	...	...	...	...	...	...	...	...
Movie202	6.0	4.333333	1.632993	1.0	5.00	5.0	5.0	5.0
Movie203	1.0	3.000000	NaN	3.0	3.00	3.0	3.0	3.0
Movie204	8.0	4.375000	1.407886	1.0	4.75	5.0	5.0	5.0
Movie205	35.0	4.628571	0.910259	1.0	5.00	5.0	5.0	5.0
Movie206	13.0	4.923077	0.277350	4.0	5.00	5.0	5.0	5.0

206 rows × 8 columns

Task 1: Which movies have maximum views/ratings?

In [17]:
```
# Movie with highest views

df.describe().T['count'].sort_values(ascending=False)[:1].to_frame()
```

Out[17]:

	count
Movie127	2313.0

In [18]:
```
# Movie with highest rating
df.drop('user_id',axis=1).sum().sort_values(ascending=False)[:1].to_frame()
```

Out[18]:

	0
Movie127	9511.0

Task 2: What is the average rating for each movie? Define the top 5 movies with the maximum ratings.

In [19]:
```
df.drop('user_id',axis=1).mean().sort_values(ascending=False)[:5].to_frame()
```

Out[19]:

	0
Movie1	5.0
Movie66	5.0
Movie76	5.0
Movie75	5.0
Movie74	5.0

Task 3: Define the top 5 movies with the least audience

In [20]:
```
df.drop('user_id',axis=1).mean().sort_values(ascending=True)[:5].to_frame()
```

Out[20]:

	0
Movie144	1.0
Movie67	1.0
Movie45	1.0
Movie58	1.0
Movie60	1.0

Task 4 : Recommendation Model

In [21]:
```
from surprise import Reader
from surprise import accuracy
from surprise import Dataset
from surprise.model_selection import train_test_split
from surprise import SVD
from surprise.model_selection import cross_validate
```

In [23]:
```
df_melt=df.melt(id_vars=df.columns[0],value_vars=df.columns[1:],var_name="Movies",value_name="Rating")
```

In [24]:
```
df_melt
```

Out[24]:

	user_id	Movies	Rating
0	A3R5OBKS7OM2IR	Movie1	5.0
1	AH3QC2PC1VTGP	Movie1	NaN
2	A3LKP6WPMP9UKX	Movie1	NaN
3	AVIY68KEPQ5ZD	Movie1	NaN
4	A1CV1WROP5KTTW	Movie1	NaN
...	...	...	...
998683	A1IMQ9WMFYKWH5	Movie206	5.0
998684	A1KLIKPUF5E88I	Movie206	5.0
998685	A5HG6WFZLO10D	Movie206	5.0
998686	A3UU690TWXCG1X	Movie206	5.0
998687	AI4J762YI6S06	Movie206	5.0

998688 rows × 3 columns

In [25]:
```
rd=Reader()
data=Dataset.load_from_df(df_melt.fillna(0),reader=rd)
data
```

Out[25]: <surprise.dataset.DatasetAutoFolds at 0x2347b193220>

In [26]:
```
trainset,testset=train_test_split(data,test_size=0.25)
```

In [27]:
```
#Using SVD (Singular Value Descomposition)
svd=SVD()
svd.fit(trainset)
```

Out[27]: <surprise.prediction_algorithms.matrix_factorization.SVD at 0x23400fdffa0>

In [29]:
```
pred=svd.test(testset)
```

In [30]:
```
accuracy.rmse(pred)
```

```
RMSE: 1.0257
```

Out[30]:
```
1.025728887899078
```

In [31]:
```
accuracy.mae(pred)
```

```
MAE:  1.0119
```

Out[31]:
```
1.0118749492558399
```

In [33]:
```
cross_validate(svd,data,measures=['RMSE','MAE'],cv=3, verbose= True)
```

```
Evaluating RMSE, MAE of algorithm SVD on 3 split(s).

                  Fold 1  Fold 2  Fold 3  Mean    Std
RMSE (testset)    1.0250  1.0268  1.0266  1.0261  0.0008
MAE (testset)     1.0116  1.0124  1.0122  1.0120  0.0003
Fit time          44.05   50.15   48.29   47.49   2.55
Test time         4.22    4.63    3.50    4.12    0.47
```

Out[33]:
```
{'test_rmse': array([1.02501788, 1.02680059, 1.02659326]),
 'test_mae': array([1.01156378, 1.01239774, 1.01215008]),
 'fit_time': (44.04900121688843, 50.14578437805176, 48.28626322746277),
 'test_time': (4.220624208450317, 4.626606225967407, 3.501664876937866)}
```

In [34]:
```
def repeat(ml_type,dframe):
    rd=Reader()
    data=Dataset.load_from_df(dframe,reader=rd)
    print(cross_validate(ml_type,data,measures=['RMSE','MAE'],cv=3,verbose=True))
    print("--"*15)
    usr_id = 'A3R5OBKS7OM2IR'
    mv = 'Movie1'
    r_u = 5.0
    print(ml_type.predict(usr_id,mv,r_ui = r_u,verbose=True))
    print("--"*15)
```

In [35]:
```
repeat(SVD(),df_melt.fillna(df_melt['Rating'].mean()))
#repeat(SVD(),df_melt.fillna(df_melt['Rating'].median()))
```

```
Evaluating RMSE, MAE of algorithm SVD on 3 split(s).

                  Fold 1  Fold 2  Fold 3  Mean    Std
RMSE (testset)    0.0856  0.0845  0.0881  0.0860  0.0015
MAE (testset)     0.0098  0.0097  0.0098  0.0098  0.0000
Fit time          47.95   47.77   49.64   48.46   0.84
Test time         3.75    5.13    4.14    4.34    0.58
{'test_rmse': array([0.08556113, 0.08450821, 0.08807983]), 'test_mae': array([0.00982059, 0.00974776, 0.009760
51]), 'fit_time': (47.950905323028564, 47.77400255203247, 49.64251399040222), 'test_time': (3.752955198287964,
5.131466865539551, 4.141684293746948)}
------------------------------
user: A3R5OBKS7OM2IR item: Movie1     r_ui = 5.00   est = 4.40   {'was_impossible': False}
user: A3R5OBKS7OM2IR item: Movie1     r_ui = 5.00   est = 4.40   {'was_impossible': False}
------------------------------
```

In [36]:
```
#trying grid search and find optimum hyperparameter value for n_factors
from surprise.model_selection import GridSearchCV
```

In [37]:
```
param_grid = {'n_epochs':[20,30],
             'lr_all':[0.005,0.001],
             'n_factors':[50,100]}
```

In [38]:
```
gs = GridSearchCV(SVD,param_grid,measures=['rmse','mae'],cv=3)
data1 = Dataset.load_from_df(df_melt.fillna(df_melt['Rating'].mean()),reader=rd)
gs.fit(data1)
```

In [39]:
```
gs.best_score
```

Out[39]:
```
{'rmse': 0.08478880296660705, 'mae': 0.009003663059166988}
```

In [40]:
```
print(gs.best_score["rmse"])
print(gs.best_params["rmse"])
```

```
0.084788802966660705
{'n_epochs': 30, 'lr_all': 0.001, 'n_factors': 50}
```

In [ ]: