Building user-based recommendation model for Amazon

Analysis Task

- Exploratory Data Analysis:

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

Divide the data into training and test data

Build a recommendation model on training data

Make predictions on the test data

In [2]:

```
import numpy as np
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
%matplotlib inline
```

In [3]:

```
data= pd.read_csv('Amazon-Movies and TV Ratings.csv')
```

In [4]:

```
data.head()
```

Out[4]:

	user_id	Movie1	Movie2	Movie3	Movie4	Movie5	Movie6	Movie7	Movie8	Movi
0	A3R5OBKS7OM2IR	5.0	5.0	NaN	NaN	NaN	NaN	NaN	NaN	N
1	AH3QC2PC1VTGP	NaN	NaN	2.0	NaN	NaN	NaN	NaN	NaN	N
2	A3LKP6WPMP9UKX	NaN	NaN	NaN	5.0	NaN	NaN	NaN	NaN	N
3	AVIY68KEPQ5ZD	NaN	NaN	NaN	5.0	NaN	NaN	NaN	NaN	N
4	A1CV1WROP5KTTW	NaN	NaN	NaN	NaN	5.0	NaN	NaN	NaN	N

5 rows × 207 columns

In [5]:

```
data.shape
```

Out[5]:

```
(4848, 207)
```

In [6]:

```
data.describe()
```

Out[6]:

	Movie1	Movie2	Movie3	Movie4	Movie5	Movie6	Movie7	Movie8	Movie9	Movie10
count	1.0	1.0	1.0	2.0	29.000000	1.0	1.0	1.0	1.0	1.0
mean	5.0	5.0	2.0	5.0	4.103448	4.0	5.0	5.0	5.0	5.0
std	NaN	NaN	NaN	0.0	1.496301	NaN	NaN	NaN	NaN	NaN
min	5.0	5.0	2.0	5.0	1.000000	4.0	5.0	5.0	5.0	5.0
25%	5.0	5.0	2.0	5.0	4.000000	4.0	5.0	5.0	5.0	5.0
50%	5.0	5.0	2.0	5.0	5.000000	4.0	5.0	5.0	5.0	5.0
75%	5.0	5.0	2.0	5.0	5.000000	4.0	5.0	5.0	5.0	5.0
max	5.0	5.0	2.0	5.0	5.000000	4.0	5.0	5.0	5.0	5.0

8 rows × 206 columns

Task 1 - Which movies have maximum views/ratings?

In [7]:

```
#movie with highest views
data.describe().T['count'].sort_values(ascending=False)[:1].to_frame()
```

Out[7]:

	count
Movie127	2313.0

In [8]:

```
data.describe().T['count']
```

Out[8]:

```
Movie1       1.0
Movie2       1.0
Movie3       1.0
Movie4       2.0
Movie5      29.0
            ...
Movie202     6.0
Movie203     1.0
Movie204     8.0
Movie205    35.0
Movie206    13.0
Name: count, Length: 206, dtype: float64
```

In [9]:

```
#Movie with highest Ratings
data.drop('user_id',axis=1).sum().sort_values(ascending=False)[:1].to_frame()
```

Out[9]:

	0
Movie127	9511.0

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

In [10]:

```
data.drop('user_id',axis=1).mean()
```

Out[10]:

```
Movie1      5.000000
Movie2      5.000000
Movie3      2.000000
Movie4      5.000000
Movie5      4.103448
              ...
Movie202    4.333333
Movie203    3.000000
Movie204    4.375000
Movie205    4.628571
Movie206    4.923077
Length: 206, dtype: float64
```

In [11]:

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

Out[11]:

	0
Movie1	5.0
Movie55	5.0
Movie131	5.0
Movie132	5.0
Movie133	5.0

task3- Define the top 5 movies with the least audience.

In [12]:

```
data.describe().T['count']
```

Out[12]:

```
Movie1       1.0
Movie2       1.0
Movie3       1.0
Movie4       2.0
Movie5      29.0
            ...
Movie202     6.0
Movie203     1.0
Movie204     8.0
Movie205    35.0
Movie206    13.0
Name: count, Length: 206, dtype: float64
```

In [13]:

```
data.describe().T['count'].sort_values(ascending=True)[:5].to_frame()
```

Out[13]:

	count
Movie1	1.0
Movie71	1.0
Movie145	1.0
Movie69	1.0
Movie68	1.0

Recommendation Model

In [29]:

```
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 [30]:

```
data_melt = data.melt(id_vars = data.columns[0],value_vars=data.columns[1:],var_name="Movi
es",value_name="Rating")
```

In [31]:

```
data_melt
```

Out[31]:

	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 [32]:

```
rd = Reader()
df = Dataset.load_from_df(data_melt.fillna(0),reader=rd)
df
```

Out[32]:

```
<surprise.dataset.DatasetAutoFolds at 0x7f95c8be2c90>
```

In [33]:

```
trainset, testset = train_test_split(df,test_size=0.25)
```

In [34]:

```
svd = SVD()
svd.fit(trainset)
```

Out[34]:

```
<surprise.prediction_algorithms.matrix_factorization.SVD at 0x7f95c8be28d0>
```

In [35]:

```
pred = svd.test(testset)
```

In [36]:

```
accuracy.rmse(pred)
```

```
RMSE: 1.0262
```

Out[36]:

```
1.026193509669748
```

In [37]:

```
accuracy.mae(pred)
```

```
MAE:  1.0121
```

Out[37]:

```
1.0121444735294005
```

In [39]:

```
cross_validate(svd, df, 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)    1.0256  1.0273  1.0252  1.0260  0.0009
MAE (testset)     1.0119  1.0125  1.0115  1.0120  0.0004
Fit time          36.17   36.27   36.44   36.29   0.11
Test time         3.58    3.17    3.63    3.46    0.21
```

Out[39]:

```
{'test_rmse': array([1.02561997, 1.02733875, 1.02516155]),
 'test_mae': array([1.01189892, 1.012476  , 1.01154928]),
 'fit_time': (36.17048525810242, 36.26771950721741, 36.43838095664978),
 'test_time': (3.5817606449127197, 3.172581434249878, 3.6323931217193604)}
```

In [ ]:

```

```