Build 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.

In [19]:

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
```

In [20]:

```
ratings_df = pd.read_csv('amazon-ratings.csv')
ratings_df.head()
```

Out[20]:

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

5 rows × 207 columns

In [21]:

```
ratings_df.shape
```

Out[21]:

```
(4848, 207)
```

In [22]:

```
ratings_df.describe().T
```

Out[22]:

	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

Top 10 rated movies

In [23]:

```
top_rated = ratings_df.drop('user_id', axis=1).sum().sort_values(ascending=False).
top_rated[:10]
```

Out[23]:

	0
Movie127	9511.0
Movie140	2794.0
Movie16	1446.0
Movie103	1241.0
Movie29	1168.0
Movie91	586.0
Movie92	482.0
Movie89	380.0
Movie158	318.0
Movie108	252.0

Average ratings for each movie

In [24]:

```
ratings_df.describe().T['mean'][:10]
```

Out[24]:

```
Movie1      5.000000
Movie2      5.000000
Movie3      2.000000
Movie4      5.000000
Movie5      4.103448
Movie6      4.000000
Movie7      5.000000
Movie8      5.000000
Movie9      5.000000
Movie10     5.000000
Name: mean, dtype: float64
```

Top 5 movies with least audience

In [25]:

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

Out[25]:

```
Movie1      1.0
Movie71     1.0
Movie145    1.0
Movie69     1.0
Movie68     1.0
Name: count, dtype: float64
```

In [26]:

```
!pip install surprise
```

```
Looking in indexes: https://pypi.org/simple, https://us-python.pkg.dev/colab-wheel
s/public/simple/
Requirement already satisfied: surprise in /usr/local/lib/python3.7/dist-packages
(0.1)
Requirement already satisfied: scikit-surprise in /usr/local/lib/python3.7/dist-pa
ckages (from surprise) (1.1.1)
Requirement already satisfied: numpy>=1.11.2 in /usr/local/lib/python3.7/dist-pack
ages (from scikit-surprise->surprise) (1.21.6)
Requirement already satisfied: six>=1.10.0 in /usr/local/lib/python3.7/dist-packag
es (from scikit-surprise->surprise) (1.15.0)
Requirement already satisfied: scipy>=1.0.0 in /usr/local/lib/python3.7/dist-packa
ges (from scikit-surprise->surprise) (1.4.1)
Requirement already satisfied: joblib>=0.11 in /usr/local/lib/python3.7/dist-packa
ges (from scikit-surprise->surprise) (1.1.0)
```

Recommendation model

In [27]:

```
# Using surpise for building our recommendation system
from surprise import Reader, Dataset, SVD, accuracy
from surprise.model_selection import train_test_split, cross_validate
```

In [28]:

```
ratings_df.columns
```

Out[28]:

```
Index(['user_id', 'Movie1', 'Movie2', 'Movie3', 'Movie4', 'Movie5', 'Movie6',
       'Movie7', 'Movie8', 'Movie9',
       ...
       'Movie197', 'Movie198', 'Movie199', 'Movie200', 'Movie201', 'Movie202',
       'Movie203', 'Movie204', 'Movie205', 'Movie206'],
      dtype='object', length=207)
```

In [29]:

```
melt_ratings = ratings_df.melt(id_vars=ratings_df.columns[0], value_vars=ratings_d
melt_ratings
```

Out[29]:

	user_id	Movie	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 [30]:

```
melt_ratings.isna().any().sum()
```

Out[30]:

```
1
```

In [31]:

```
reader = Reader(rating_scale=(-1,10))

data = Dataset.load_from_df(melt_ratings.fillna(0), reader=reader)
data
```

Out[31]:

```
<surprise.dataset.DatasetAutoFolds at 0x7fd8f81a11d0>
```

Divide the data into train and testing set and train the model

In [32]:

```
train, test = train_test_split(data, test_size=0.20, random_state=34)
model = SVD()
model.fit(train)
```

Out[32]:

```
<surprise.prediction_algorithms.matrix_factorization.SVD at 0x7fd8f82a7d10>
```

Make predictions on the test data

In [33]:

```
preds = model.test(test)
preds[:5]
```

Out[33]:

```
[Prediction(uid='A3KZUOZGIO6E4L', iid='Movie101', r_ui=0.0, est=-0.001623562891737
7646, details={'was_impossible': False}),
 Prediction(uid='A2CL7EKD6I7V1K', iid='Movie49', r_ui=0.0, est=0.00844616920899622
1, details={'was_impossible': False}),
 Prediction(uid='A2JJDM12OF39JK', iid='Movie132', r_ui=0.0, est=0.0218120895397022
6, details={'was_impossible': False}),
 Prediction(uid='A1E89BHC9S5IFW', iid='Movie45', r_ui=0.0, est=0.00506186612644253
2, details={'was_impossible': False}),
 Prediction(uid='A2Z9WAEEHR4PD1', iid='Movie49', r_ui=0.0, est=0.00229868972454490
98, details={'was_impossible': False})]
```

In [34]:

```
accuracy.rmse(preds)
accuracy.mae(preds)
```

```
RMSE: 0.2726
MAE:  0.0400
```

Out[34]:

```
0.04003937239790102
```

In [35]:

```
cross_val = cross_validate(model, data, measures=['rmse', 'mae'], cv=3, verbose=Tru
cross_val
```

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

                  Fold 1  Fold 2  Fold 3  Mean    Std     
RMSE (testset)    0.2866  0.2778  0.2844  0.2829  0.0037  
MAE (testset)     0.0432  0.0424  0.0432  0.0429  0.0004  
Fit time          47.80   37.36   36.42   40.53   5.16    
Test time         3.45    3.48    2.78    3.24    0.32    
```

Out[35]:

```
{'fit_time': (47.799657583236694, 37.36354207992554, 36.419992446899414),
 'test_mae': array([0.0431929 , 0.04239181, 0.04319095]),
 'test_rmse': array([0.28657397, 0.27784336, 0.28440121]),
 'test_time': (3.4529740810394287, 3.4816577434539795, 2.7821645736694336)}
```