Recommendation model for Amazon

September 3, 2022

[1]:
```
import numpy as np
import pandas as pd
import re
import matplotlib.pyplot as plt
import seaborn as sns
%matplotlib inline
import surprise
```

[2]:
```
df = pd.read_csv('Amazon - Movies and TV Ratings.csv')
```

[3]:
```
df.head()
```

[3]:

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

	Movie8	Movie9	…	Movie197	Movie198	Movie199	Movie200	Movie201	\
0	NaN	NaN	…	NaN	NaN	NaN	NaN	NaN	
1	NaN	NaN	…	NaN	NaN	NaN	NaN	NaN	
2	NaN	NaN	…	NaN	NaN	NaN	NaN	NaN	
3	NaN	NaN	…	NaN	NaN	NaN	NaN	NaN	
4	NaN	NaN	…	NaN	NaN	NaN	NaN	NaN	

	Movie202	Movie203	Movie204	Movie205	Movie206
0	NaN	NaN	NaN	NaN	NaN
1	NaN	NaN	NaN	NaN	NaN
2	NaN	NaN	NaN	NaN	NaN
3	NaN	NaN	NaN	NaN	NaN
4	NaN	NaN	NaN	NaN	NaN

[5 rows x 207 columns]

[4]:
```
df.shape
```

[4]:
```
(4848, 207)
```

[5]:
```
df_org = df.copy()
```

[6]:
```
df.describe().T
```

[6]:

	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 x 8 columns]

[7]:
```
df.describe().T['count'].sort_values(ascending=False)[:1].to_frame()
```

[7]:

	count
Movie127	2313.0

[8]:
```
df.drop('user_id',axis=1).sum().sort_values(ascending=False)[:1].to_frame()
```

[8]:

	0
Movie127	9511.0

[9]:
```
df.drop('user_id',axis=1).mean().sort_values(ascending=False)[:5].to_frame()
```

[9]:

	0
Movie1	5.0
Movie55	5.0
Movie131	5.0
Movie132	5.0
Movie133	5.0

[10]:
```
df.describe().T['count'].sort_values(ascending=True)[:5].to_frame()
```

[10]:

	count
Movie1	1.0
Movie71	1.0
Movie145	1.0
Movie69	1.0

```
Movie68     1.0
```

[11]:
```
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
```

[12]:
```
df_melt = df.melt(id_vars = df.columns[0],value_vars=df.columns[1:
 ↪],var_name="Movies",value_name="Rating")
```

[13]:
```
df_melt
```

[13]:

	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 x 3 columns]

[14]:
```
rd = Reader()
data = Dataset.load_from_df(df_melt.fillna(0),reader=rd)
data
```

[14]:
```
<surprise.dataset.DatasetAutoFolds at 0x7f2d7433c3d0>
```

[15]:
```
trainset, testset = train_test_split(data,test_size=0.25)
```

[16]:
```
svd = SVD()
svd.fit(trainset)
```

[16]:
```
<surprise.prediction_algorithms.matrix_factorization.SVD at 0x7f2d74245ed0>
```

[17]:
```
pred = svd.test(testset)
```

[18]:
```
accuracy.rmse(pred)
```

```
RMSE: 1.0264
```

[18]: 1.026359768441249

[19]:
```
accuracy.mae(pred)
```

```
MAE:  1.0121
```

[19]: 1.0121462655228097

[20]:
```
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)    1.0259  1.0267  1.0258  1.0261  0.0004
MAE (testset)     1.0119  1.0123  1.0119  1.0120  0.0002
Fit time          45.39   42.84   38.77   42.33   2.73
Test time         3.79    4.11    3.12    3.67    0.41
```

[20]:
```
{'test_rmse': array([1.02586777, 1.02673753, 1.02575544]),
 'test_mae': array([1.01191813, 1.01229595, 1.01191836]),
 'fit_time': (45.39322519302368, 42.83663868904114, 38.765148639678955),
 'test_time': (3.786543369293213, 4.111851453781128, 3.1173317432403564)}
```

[21]:
```
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)
```

[22]:
```
repeat(SVD(),df_melt.fillna(df_melt['Rating'].mean()))
```

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

                  Fold 1  Fold 2  Fold 3  Mean    Std
RMSE (testset)    0.0873  0.0858  0.0850  0.0860  0.0010
MAE (testset)     0.0097  0.0097  0.0100  0.0098  0.0001
Fit time          37.24   38.48   41.61   39.11   1.84
Test time         3.54    3.43    4.16    3.71    0.32
{'test_rmse': array([0.08731964, 0.0857578 , 0.08498391]), 'test_mae':
array([0.00972595, 0.00970878, 0.0100081 ]), 'fit_time': (37.23628807067871,
38.47825622558594, 41.609673738479614), 'test_time': (3.5403342247009277,
3.4278509616851807, 4.1581056118011475)}
```

```
------------------------------
user: A3R5OBKS7OM2IR item: Movie1     r_ui = 5.00   est = 4.39
{'was_impossible': False}
user: A3R5OBKS7OM2IR item: Movie1     r_ui = 5.00   est = 4.39
{'was_impossible': False}
------------------------------
```

[23]:
```
from surprise.model_selection import GridSearchCV
```

[24]:
```
param_grid = {'n_epochs':[20,30],
              'lr_all':[0.005,0.001],
              'n_factors':[50,100]}
```

[ ]:
```
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)
```

[ ]:
```
gs.best_score
```

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

[ ]:
```

```