prject2_Raya

June 16, 2021

[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	Movie8	Movie9	…	Movie197	Movie198	Movie199	Movie200	Movie201	Movie202	Movie203	Movie204	Movie205	Movie206
0	A3R5OBKS7OM2IR	5.0	5.0	NaN	NaN	NaN	NaN	NaN	NaN	NaN	…	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	NaN	NaN	NaN	NaN	NaN	NaN	NaN
2	A3LKP6WPMP9UKX	NaN	NaN	NaN	5.0	NaN	NaN	NaN	NaN	NaN	…	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
3	AVIY68KEPQ5ZD	NaN	NaN	NaN	5.0	NaN	NaN	NaN	NaN	NaN	…	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
4	A1CV1WROP5KTTW	NaN	NaN	NaN	NaN	5.0	NaN	NaN	NaN	NaN	…	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN

[5 rows x 207 columns]

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

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

[5]:
```
dfOrg = 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]:
```
#Which movies have maximum views/ratings?
df.describe().T['count'].sort_values(ascending=False)[:1].to_frame()
#Movie127 has the highest views
```

[7]:

	count
Movie127	2313.0

[8]:
```
df.drop('user_id',axis=1).sum().sort_values(ascending=False)[:1].to_frame()
#Movie127 has the highest ratings
```

[8]:

	0
Movie127	9511.0

[9]:
```
#What is the average rating for each movie? Define the top 5 movies with the␣
 ↪maximum ratings.
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]:
```
#Define the top 5 movies with the least audience.
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]:
```
#Recommendation Model:
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 0x7fc078fbf250>
```

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

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

[16]:
```
<surprise.prediction_algorithms.matrix_factorization.SVD at 0x7fc078fbfe10>
```

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

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

```
RMSE: 1.0253
```

[18]:
```
1.0252786011770172
```

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

```
MAE:  1.0116
```

[19]:
```
1.0115972435024905
```

[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)    1.0260  1.0259  1.0267  1.0262  0.0003
MAE (testset)     1.0120  1.0120  1.0124  1.0121  0.0002
Fit time          39.49   36.76   36.24   37.50   1.42
Test time         3.39    3.00    3.00    3.13    0.18
```

[20]:
```
{'test_rmse': array([1.02604182, 1.02592851, 1.02670752]),
 'test_mae': array([1.01199013, 1.01201998, 1.01235094]),
 'fit_time': (39.49166703224182, 36.76410531997681, 36.24354648590088),
 'test_time': (3.3886313438415527, 2.997849464416504, 3.001559019088745)}
```

[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)    0.0863  0.0857  0.0872  0.0864  0.0006
MAE (testset)     0.0097  0.0096  0.0096  0.0096  0.0001
Fit time          35.86   36.79   36.56   36.40   0.40
Test time         3.18    3.62    3.18    3.32    0.21
{'test_rmse': array([0.08628026, 0.08568767, 0.08718247]), 'test_mae':
array([0.0097252 , 0.00956807, 0.00960138]), 'fit_time': (35.8600754737854,
36.7913875579834, 36.562650203704834), 'test_time': (3.175793170928955,
3.615278720855713, 3.1831581592559814)}
------------------------------
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]:
```
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)    0.0938  0.0935  0.0901  0.0925  0.0017
MAE (testset)     0.0073  0.0069  0.0070  0.0071  0.0002
Fit time          37.79   38.57   38.16   38.17   0.32
Test time         3.75    3.47    3.38    3.53    0.16
{'test_rmse': array([0.09379278, 0.09352264, 0.09013461]), 'test_mae':
array([0.00731853, 0.0069131 , 0.00702257]), 'fit_time': (37.787620067596436,
38.56571078300476, 38.16427397727966), 'test_time': (3.7450993061065674,
3.4689767360687256, 3.37705397605896)}
------------------------------
user: A3R5OBKS7OM2IR item: Movie1     r_ui = 5.00   est = 5.00
{'was_impossible': False}
user: A3R5OBKS7OM2IR item: Movie1     r_ui = 5.00   est = 5.00
{'was_impossible': False}
------------------------------
```

[24]:
```
#grid search and find optimum hyperparameter value for n_factors
from surprise.model_selection import GridSearchCV
```

[25]:
```
parameter_grid = {'n_epochs':[20,30],
             'lr_all':[0.005,0.001],
             'n_factors':[50,100]}
```

[26]:
```
gs = GridSearchCV(SVD,parameter_grid,measures=['rmse','mae'],cv=3)
data1 = Dataset.load_from_df(df_melt.fillna(df_melt['Rating'].mean()),reader=rd)
gs.fit(data1)
```

[27]:
```
gs.best_score
```

[27]:
```
{'rmse': 0.0846889858892285, 'mae': 0.008952823676053866}
```

[28]:
```
print(gs.best_score["rmse"])
print(gs.best_params["rmse"])
```

```
0.0846889858892285
{'n_epochs': 30, 'lr_all': 0.001, 'n_factors': 50}
```

[ ]: