Amazon_Project

April 14, 2021

[2]:
```
# imports
import pandas as pd
import numpy as np
from surprise import Reader
from surprise.model_selection import train_test_split
from surprise import accuracy
from surprise import Dataset
from surprise import SVD
from surprise.model_selection import cross_validate
```

[3]:
```
# import dataset
data = pd.read_csv("Amazon - Movies and TV Ratings.csv")
```

[4]:
```
#info on datase
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 4848 entries, 0 to 4847
Columns: 207 entries, user_id to Movie206
dtypes: float64(206), object(1)
memory usage: 7.7+ MB
```

[5]:
```
data.shape
```

[5]:
```
(4848, 207)
```

[6]:
```
data.dtypes
```

[6]:
```
user_id      object
Movie1      float64
Movie2      float64
Movie3      float64
Movie4      float64
             …
Movie202    float64
Movie203    float64
Movie204    float64
Movie205    float64
```

```
Movie206    float64
Length: 207, dtype: object
```

[7]:
```
data.columns
```

[7]:
```
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)
```

[8]:
```
data.head()
```

[8]:

	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]

[9]:
```
type(data)
```

[9]:
```
pandas.core.frame.DataFrame
```

[10]:
```
data.tail()
```

[10]:

	user_id	Movie1	Movie2	Movie3	Movie4	Movie5	Movie6	Movie7	\
4843	A1IMQ9WMFYKWH5	NaN	NaN	NaN	NaN	NaN	NaN	NaN	
4844	A1KLIKPUF5E88I	NaN	NaN	NaN	NaN	NaN	NaN	NaN	
4845	A5HG6WFZLO10D	NaN	NaN	NaN	NaN	NaN	NaN	NaN	

```
4846  A3UU690TWXCG1X     NaN     NaN     NaN     NaN     NaN     NaN     NaN
4847   AI4J762YI6S06     NaN     NaN     NaN     NaN     NaN     NaN     NaN

      Movie8  Movie9  …  Movie197  Movie198  Movie199  Movie200  Movie201  \
4843     NaN     NaN  …       NaN       NaN       NaN       NaN       NaN
4844     NaN     NaN  …       NaN       NaN       NaN       NaN       NaN
4845     NaN     NaN  …       NaN       NaN       NaN       NaN       NaN
4846     NaN     NaN  …       NaN       NaN       NaN       NaN       NaN
4847     NaN     NaN  …       NaN       NaN       NaN       NaN       NaN

      Movie202  Movie203  Movie204  Movie205  Movie206
4843       NaN       NaN       NaN       NaN       5.0
4844       NaN       NaN       NaN       NaN       5.0
4845       NaN       NaN       NaN       NaN       5.0
4846       NaN       NaN       NaN       NaN       5.0
4847       NaN       NaN       NaN       NaN       5.0

[5 rows x 207 columns]
```

[11]:
```
data.info
```

[11]:
```
<bound method DataFrame.info of             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
…                ….     …     …     …     …     …     …     …
4843  A1IMQ9WMFYKWH5     NaN     NaN     NaN     NaN     NaN     NaN     NaN
4844  A1KLIKPUF5E88I     NaN     NaN     NaN     NaN     NaN     NaN     NaN
4845   A5HG6WFZLO10D     NaN     NaN     NaN     NaN     NaN     NaN     NaN
4846  A3UU690TWXCG1X     NaN     NaN     NaN     NaN     NaN     NaN     NaN
4847   AI4J762YI6S06     NaN     NaN     NaN     NaN     NaN     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
…      …     …  …         …         …         …         …         …
4843     NaN     NaN  …       NaN       NaN       NaN       NaN       NaN
4844     NaN     NaN  …       NaN       NaN       NaN       NaN       NaN
4845     NaN     NaN  …       NaN       NaN       NaN       NaN       NaN
4846     NaN     NaN  …       NaN       NaN       NaN       NaN       NaN
4847     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
…          …         …         …         …         …
4843       NaN       NaN       NaN       NaN       5.0
4844       NaN       NaN       NaN       NaN       5.0
4845       NaN       NaN       NaN       NaN       5.0
4846       NaN       NaN       NaN       NaN       5.0
4847       NaN       NaN       NaN       NaN       5.0

[4848 rows x 207 columns]>
```

1 Analysis Task

Exploratory Data Analysis:

Which movies have maximum views/ratings?

[13]:
```
#dropping user Id
data_mod = data.drop(['user_id'], axis = 1)
```

[14]:
```
#movie with highest views
data_mod.describe().T.sort_values('count', ascending=False).head(n=1)
```

[14]:
```
           count      mean       std  min  25%  50%  75%  max
Movie127  2313.0  4.111976  1.420621  1.0  4.0  5.0  5.0  5.0
```

[15]:
```
#movie with highest ratings
data_mod.sum(axis=0).sort_values(ascending=False).head(n=1)
```

[15]:
```
Movie127    9511.0
dtype: float64
```

[16]:
```
data_mod.mean(axis=0).sort_values(ascending=False).head(n=5)
```

[16]:
```
Movie1      5.0
Movie55     5.0
Movie131    5.0
Movie132    5.0
Movie133    5.0
dtype: float64
```

Define the top 5 movies with the least audience.

[17]:
```
data_mod.describe().T.sort_values('count', ascending=True).head(n=5)
```

[17]:

	count	mean	std	min	25%	50%	75%	max
Movie1	1.0	5.0	NaN	5.0	5.0	5.0	5.0	5.0
Movie71	1.0	4.0	NaN	4.0	4.0	4.0	4.0	4.0
Movie145	1.0	5.0	NaN	5.0	5.0	5.0	5.0	5.0
Movie69	1.0	1.0	NaN	1.0	1.0	1.0	1.0	1.0
Movie68	1.0	5.0	NaN	5.0	5.0	5.0	5.0	5.0

2 Recommendation Model

[18]:
```
data = data.melt(id_vars = data.columns[0],value_vars=data.columns[1:
 ↪],var_name="Movie",value_name="Rating")
data
#split dataset to train and test sets
#x_train,x_test,y_train,y_test = train_test_split(x,y, test_size = 0.35,␣
 ↪random_state= 21)
```

[18]:

	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 x 3 columns]

[19]:
```
data.isnull().sum()
```

[19]:
```
user_id         0
Movie           0
Rating     993688
dtype: int64
```

[20]:
```
rd = Reader()
data = Dataset.load_from_df(data.fillna(0),reader=rd)
data
```

[20]:
```
<surprise.dataset.DatasetAutoFolds at 0x7fc8399ab7d0>
```

[21]:
```
#split dataset to train and test sets
train_data,test_data = train_test_split(data, test_size = 0.25)
```

[24]:
```
svd_instance = SVD()
svd_instance.fit(train_data)
prediction = svd_instance.test(test_data)
```

[23]:
```
accuracy.rmse(prediction)
```

```
RMSE: 1.0257
```

[23]:
```
1.0257033388745014
```

[25]:
```
accuracy.mae(prediction)
```

```
MAE:  1.0119
```

[25]:
```
1.0119379176139098
```

[26]:
```
cross_validate(svd_instance, data, measures = ['RMSE', 'MAE'], cv = 5, verbose␣
↪= True)
```

```
Evaluating RMSE, MAE of algorithm SVD on 5 split(s).

                  Fold 1  Fold 2  Fold 3  Fold 4  Fold 5  Mean    Std
RMSE (testset)    1.0259  1.0256  1.0268  1.0258  1.0237  1.0256  0.0010
MAE (testset)     1.0121  1.0118  1.0123  1.0119  1.0110  1.0118  0.0004
Fit time          43.80   44.26   44.05   43.93   43.76   43.96   0.18
Test time         2.19    1.79    1.76    2.17    1.75    1.93    0.20
```

[26]:
```
{'test_rmse': array([1.02591825, 1.02561209, 1.02675186, 1.02581706,
1.02371218]),
 'test_mae': array([1.01208584, 1.01183132, 1.01225757, 1.01185229,
1.01097737]),
 'fit_time': (43.803428649902344,
  44.256295680999756,
  44.04584884643555,
  43.93290400505066,
  43.76267957687378),
 'test_time': (2.1885664463043213,
  1.7878835201263428,
  1.7595062255859375,
  2.1735153198242188,
  1.753739595413208)}
```

[ ]:
```

```