Assignment

March 6, 2022

[3]:
```
import pandas as pd
import numpy as np
```

[4]:
```
dataset_amazon=pd.read_csv('Amazon - Movies and TV Ratings.csv')
```

[5]:
```
dataset_amazon.head()
```

[5]:
```
         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]
```

[6]:
```
dataset_amazon.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
```

[7]:
```
movie_data=pd.melt(dataset_amazon, id_vars=["user_id"], var_name="Movie_id",␣
↪value_name="Ratings")
movie_data
```

[7]:
```
              user_id Movie_id  Ratings
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]
```

1 1. Which movies have maximum views/ratings

[8]:
```
top5=movie_data.groupby('Movie_id').agg({"Ratings":'count'}).reset_index().
↪sort_values(by='Ratings',ascending=False)
top5.columns=['Movie_id', '# Ratings']
top5.head(10)
```

[8]:
```
     Movie_id  # Ratings
31   Movie127       2313
46   Movie140        578
67    Movie16        320
5    Movie103        272
128   Movie29        243
197   Movie91        128
198   Movie92        101
194   Movie89         83
65   Movie158         66
10   Movie108         54
```

Highest Ratings

[9]:
```
movie_data.groupby('Movie_id').agg({"Ratings":'sum'}).reset_index().
↪sort_values(by='Ratings',ascending=False).head(10)
```

[9]:
```
     Movie_id  Ratings
31   Movie127   9511.0
```

```
46   Movie140   2794.0
67    Movie16   1446.0
5    Movie103   1241.0
128   Movie29   1168.0
197   Movie91    586.0
198   Movie92    482.0
194   Movie89    380.0
65   Movie158    318.0
10   Movie108    252.0
```

[10]:
```
movie_data.groupby('Movie_id').agg({"Ratings":'sum'}).reset_index().
 ↪sort_values(by='Ratings',ascending=False)
```

[10]:
```
     Movie_id  Ratings
31   Movie127   9511.0
46   Movie140   2794.0
67    Movie16   1446.0
5    Movie103   1241.0
128   Movie29   1168.0
..        …       …
61   Movie154      1.0
160   Movie58      1.0
50   Movie144      1.0
170   Movie67      1.0
163   Movie60      1.0

[206 rows x 2 columns]
```

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

[11]:
```
top5_avg=movie_data.groupby('Movie_id').agg({"Ratings":'mean'}).reset_index()
top5_avg.columns=['Movie_id', 'Average Ratings']
top5_avg.head(10)
```

[11]:
```
   Movie_id  Average Ratings
0    Movie1           5.0000
1   Movie10           5.0000
2  Movie100           4.0000
3  Movie101           5.0000
4  Movie102           4.0000
5  Movie103           4.5625
6  Movie104           4.5000
7  Movie105           5.0000
8  Movie106           5.0000
```

```
9  Movie107          4.0000
```

[12]:
```python
ds_movie=pd.merge(top5, top5_avg, on='Movie_id')
ds_movie.head(10)
```

[12]:

	Movie_id	# Ratings	Average Ratings
0	Movie127	2313	4.111976
1	Movie140	578	4.833910
2	Movie16	320	4.518750
3	Movie103	272	4.562500
4	Movie29	243	4.806584
5	Movie91	128	4.578125
6	Movie92	101	4.772277
7	Movie89	83	4.578313
8	Movie158	66	4.818182
9	Movie108	54	4.666667

3 3. Define the top 5 movies with the least audience.

[13]:
```python
ds_movie.sort_values(['Average Ratings'], ascending=False).sort_values(['#␣
↪Ratings'], ascending=True)
```

[13]:

	Movie_id	# Ratings	Average Ratings
205	Movie1	1	5.000000
200	Movie153	1	5.000000
203	Movie36	1	5.000000
199	Movie152	1	5.000000
202	Movie37	1	5.000000
..	…	…	…
4	Movie29	243	4.806584
3	Movie103	272	4.562500
2	Movie16	320	4.518750
1	Movie140	578	4.833910
0	Movie127	2313	4.111976

[206 rows x 3 columns]

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. - 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.

[14]:
```
# executed below in Command prompt
# conda install -c conda-forge scikit-surprise
```

[15]:
```
!pip install scikit-surprise
```

```
Defaulting to user installation because normal site-packages is not writeable
Requirement already satisfied: scikit-surprise in /usr/local/lib/python3.7/site-
packages (1.1.0)
Requirement already satisfied: joblib>=0.11 in /usr/local/lib/python3.7/site-
packages (from scikit-surprise) (0.14.1)
Requirement already satisfied: numpy>=1.11.2 in /usr/local/lib/python3.7/site-
packages (from scikit-surprise) (1.18.2)
Requirement already satisfied: scipy>=1.0.0 in /usr/local/lib/python3.7/site-
packages (from scikit-surprise) (1.4.1)
Requirement already satisfied: six>=1.10.0 in /usr/local/lib/python3.7/site-
packages (from scikit-surprise) (1.14.0)
```

[16]:
```
from surprise import Reader
from surprise import Dataset
from surprise import KNNBasic, SVD
from surprise import accuracy
from surprise.model_selection import train_test_split
```

[17]:
```
movie_data.fillna(0, inplace=True)
```

[18]:
```
reader = Reader(line_format='user item rating', rating_scale=(-1, 10))
```

[19]:
```
Dataset=Dataset.load_from_df(movie_data, reader=reader)
```

[20]:
```
train_set, test_set=train_test_split(Dataset, test_size=.25, random_state=10)␣
↪#Split data into training and testing set
```

[21]:
```
#KNN MODEL
#Train the model
sim_options = {'name': 'pearson', 'user_based': True}
knn=KNNBasic(k=45, min_k=3, sim_options=sim_options)
```

[22]:
```
knn.fit(train_set)
```

```
Computing the pearson similarity matrix…
Done computing similarity matrix.
```

[22]:
```
<surprise.prediction_algorithms.knns.KNNBasic at 0x7f15c289ecd0>
```

[23]:
```
predictions=knn.test(test_set)
```

[24]:
```python
result = pd.DataFrame(predictions, columns=['user_id', 'movie_id', 'base_ratings', 'predict_ratings','additional'])
result.drop('additional',axis=1,inplace=True)
```

[25]:
```python
result
```

[25]:

	user_id	movie_id	base_ratings	predict_ratings
0	A3VURT1JDRQQRI	Movie61	0.0	0.000000
1	A1FP2LA6M2OJXO	Movie7	0.0	0.000000
2	A3BIIQC3A935LS	Movie46	0.0	0.022057
3	A39ZX6ML4X7E67	Movie34	0.0	0.000000
4	A2M5FI4CB6VUXF	Movie53	0.0	0.000000
…	…	…	…	…
249667	A1EBD2U23BP04Y	Movie128	0.0	0.000000
249668	A10A4PJBRIFGLE	Movie92	0.0	0.000000
249669	ACFK06NL9N7I8	Movie182	0.0	0.000000
249670	A1Y81TTIF0F5GX	Movie157	0.0	0.000000
249671	A1XYHEBZVXGKX1	Movie72	0.0	0.022057

[249672 rows x 4 columns]

[26]:
```python
accuracy.rmse(predictions)
```

RMSE: 0.3183

[26]: 0.31830539402296587

[27]:
```python
#SVD MODEL
#Train the model
svd=SVD()
```

[28]:
```python
svd.fit(train_set)
```

[28]: <surprise.prediction_algorithms.matrix_factorization.SVD at 0x7f156c82ead0>

[29]:
```python
predictions=svd.test(test_set)
```

[30]:
```python
result = pd.DataFrame(predictions, columns=['user_id', 'movie_id', 'base_ratings', 'predict_ratings','additional'])
result.drop('additional',axis=1,inplace=True)
```

[31]:
```python
result
```

[31]:

	user_id	movie_id	base_ratings	predict_ratings
0	A3VURT1JDRQQRI	Movie61	0.0	0.010079
1	A1FP2LA6M2OJXO	Movie7	0.0	0.004391
2	A3BIIQC3A935LS	Movie46	0.0	0.003338

```
3       A39ZX6ML4X7E67   Movie34          0.0         0.008307
4       A2M5FI4CB6VUXF   Movie53          0.0         0.013513
…                 …        …          …              …
249667  A1EBD2U23BP04Y  Movie128          0.0         0.004345
249668  A10A4PJBRIFGLE   Movie92          0.0        -0.086026
249669   ACFK06NL9N7I8  Movie182          0.0        -0.031928
249670  A1Y81TTIF0F5GX  Movie157          0.0         0.016468
249671  A1XYHEBZVXGKX1   Movie72          0.0        -0.012038

[249672 rows x 4 columns]
```

[32]:
```
accuracy.rmse(predictions)
```

```
RMSE: 0.2794
```

[32]: 0.2794406865119178

[33]:
```
#The best model is svd than knnbase due to lower RMSE value
```

[34]:
```
def recommendation_movies(user):
    #get the list of unique movies
    movie_data=pd.melt(dataset_amazon, id_vars=["user_id"],␣
 ↪var_name="Movie_id", value_name="Rating")
    movie_names=movie_data.Movie_id.unique()
    #movies_watched by the user
    movies_watched=movie_data[movie_data["user_id"]==user].dropna().Movie_id
    #Movies the user didn't watch
    movies_not_watched=np.setdiff1d(movie_names, movies_watched)
    #Build the model
    model=svd.fit(Dataset.build_full_trainset())
    #Predictions
    movies_predict=[]
    for i in movies_not_watched:
        movies_predict.append((i, model.predict(user, i).est))
    return pd.DataFrame(movies_predict, columns=['Movie_id', 'Predictions']).
 ↪sort_values(by='Predictions',ascending=False)['Movie_id'].head(10).to_list()
```

[35]:
```
#Top 10 movies recommended for user=A3VURT1JDRQQRI
recommendation_movies('A3VURT1JDRQQRI')
```

[35]:
```
['Movie140',
 'Movie205',
 'Movie206',
 'Movie182',
 'Movie185',
 'Movie158',
 'Movie184',
```

```
 'Movie196',
 'Movie204',
 'Movie173']
```

[36]:
```
#Top 10 movies recommended for user=A1FP2LA6M2OJXO
recommendation_movies('A1FP2LA6M2OJXO')
```

[36]:
```
['Movie140',
 'Movie205',
 'Movie206',
 'Movie182',
 'Movie185',
 'Movie158',
 'Movie184',
 'Movie127',
 'Movie196',
 'Movie204']
```

[ ]:
```

```