In [1]:
```
import pandas as pd
```

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

In [3]:
```
df.head(10)
```

Out[3]:

	user_id	Movie1	Movie2	Movie3	Movie4	Movie5	Movie6	Movie7	Movie8	Movie9	...	Movie197	Movie198	Movie199	M
0	A3R5OBKS7OM2IR	5.0	5.0	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	
2	A3LKP6WPMP9UKX	NaN	NaN	NaN	5.0	NaN	NaN	NaN	NaN	NaN	...	NaN	NaN	NaN	
3	AVIY68KEPQ5ZD	NaN	NaN	NaN	5.0	NaN	NaN	NaN	NaN	NaN	...	NaN	NaN	NaN	
4	A1CV1WROP5KTTW	NaN	NaN	NaN	NaN	5.0	NaN	NaN	NaN	NaN	...	NaN	NaN	NaN	
5	AP57WZ2X4G0AA	NaN	NaN	NaN	NaN	2.0	NaN	NaN	NaN	NaN	...	NaN	NaN	NaN	
6	A3NMBJ2LCRCATT	NaN	NaN	NaN	NaN	5.0	NaN	NaN	NaN	NaN	...	NaN	NaN	NaN	
7	A5Y15SAOMX6XA	NaN	NaN	NaN	NaN	2.0	NaN	NaN	NaN	NaN	...	NaN	NaN	NaN	
8	A3P671HJ32TCSF	NaN	NaN	NaN	NaN	5.0	NaN	NaN	NaN	NaN	...	NaN	NaN	NaN	
9	A3VCKTRD24BG7K	NaN	NaN	NaN	NaN	5.0	NaN	NaN	NaN	NaN	...	NaN	NaN	NaN	

10 rows × 207 columns

In [4]:
```
df.describe()
```

Out[4]:

	Movie1	Movie2	Movie3	Movie4	Movie5	Movie6	Movie7	Movie8	Movie9	Movie10	...	Movie197	Movie198	Movie199	Movie
count	1.0	1.0	1.0	2.0	29.000000	1.0	1.0	1.0	1.0	1.0	...	5.000000	2.0	1.0	8.000
mean	5.0	5.0	2.0	5.0	4.103448	4.0	5.0	5.0	5.0	5.0	...	3.800000	5.0	5.0	4.625
std	NaN	NaN	NaN	0.0	1.496301	NaN	NaN	NaN	NaN	NaN	...	1.643168	0.0	NaN	0.517
min	5.0	5.0	2.0	5.0	1.000000	4.0	5.0	5.0	5.0	5.0	...	1.000000	5.0	5.0	4.000
25%	5.0	5.0	2.0	5.0	4.000000	4.0	5.0	5.0	5.0	5.0	...	4.000000	5.0	5.0	4.000
50%	5.0	5.0	2.0	5.0	5.000000	4.0	5.0	5.0	5.0	5.0	...	4.000000	5.0	5.0	5.000
75%	5.0	5.0	2.0	5.0	5.000000	4.0	5.0	5.0	5.0	5.0	...	5.000000	5.0	5.0	5.000
max	5.0	5.0	2.0	5.0	5.000000	4.0	5.0	5.0	5.0	5.0	...	5.000000	5.0	5.0	5.000

8 rows × 206 columns

In [5]:
```
df_main = df.copy()
```

Which movies have maximum views/ratings?

In [6]:
```
df.describe().T["count"].sort_values(ascending = False)[:10].to_frame()
```

Out[6]:

	count
Movie127	2313.0
Movie140	578.0
Movie16	320.0
Movie103	272.0
Movie29	243.0
Movie91	128.0
Movie92	101.0
Movie89	83.0
Movie158	66.0
Movie108	54.0

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

In [7]:
```
df.drop('user_id', axis=1).sum().sort_values(ascending = False)[:10].to_frame()
```

Out[7]:

	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

Define the top 5 movies with the least audience.

In [8]:
```
df.describe().T.sort_values('count', ascending = True)
```

Out[8]:

	count	mean	std	min	25%	50%	75%	max
Movie1	1.0	5.000000	NaN	5.0	5.0	5.0	5.0	5.0
Movie71	1.0	4.000000	NaN	4.0	4.0	4.0	4.0	4.0
Movie145	1.0	5.000000	NaN	5.0	5.0	5.0	5.0	5.0
Movie69	1.0	1.000000	NaN	1.0	1.0	1.0	1.0	1.0
Movie68	1.0	5.000000	NaN	5.0	5.0	5.0	5.0	5.0
...	...	...	...	...	...	...	...	...
Movie29	243.0	4.806584	0.655269	1.0	5.0	5.0	5.0	5.0
Movie103	272.0	4.562500	1.039688	1.0	5.0	5.0	5.0	5.0
Movie16	320.0	4.518750	0.795535	1.0	4.0	5.0	5.0	5.0
Movie140	578.0	4.833910	0.609081	1.0	5.0	5.0	5.0	5.0
Movie127	2313.0	4.111976	1.420621	1.0	4.0	5.0	5.0	5.0

206 rows × 8 columns

Recommendation Model

In [9]:
```
!pip install scikit-surprise
```

```
Requirement already satisfied: scikit-surprise in c:\users\admin\anaconda3\lib\site-packages (1.1.0)
Requirement already satisfied: joblib>=0.11 in c:\users\admin\anaconda3\lib\site-packages (from scikit-surpris
e) (0.14.1)
Requirement already satisfied: numpy>=1.11.2 in c:\users\admin\anaconda3\lib\site-packages (from scikit-surpris
e) (1.17.4)
Requirement already satisfied: scipy>=1.0.0 in c:\users\admin\anaconda3\lib\site-packages (from scikit-surpris
e) (1.3.2)
Requirement already satisfied: six>=1.10.0 in c:\users\admin\anaconda3\lib\site-packages (from scikit-surprise)
(1.13.0)
```

In [10]:
```
from surprise import Reader
from surprise import accuracy
from surprise.model_selection import train_test_split
```

In [11]:
```
help(df.melt())
```

```
|  See Also
|  --------
|  DataFrame.from_records : Constructor from tuples, also record arrays.
|  DataFrame.from_dict : From dicts of Series, arrays, or dicts.
|  DataFrame.from_items : From sequence of (key, value) pairs
|      read_csv, pandas.read_table, pandas.read_clipboard.
|
|  Examples
|  --------
|  Constructing DataFrame from a dictionary.
|
|  >>> d = {'col1': [1, 2], 'col2': [3, 4]}
|  >>> df = pd.DataFrame(data=d)
|  >>> df
|     col1  col2
|  0     1     3
|  1     2     4
|
|  Notice that the inferred dtype is int64.
|
```

In [12]:
```
df.columns
```

Out[12]:
```
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 [13]:
```
melt_df = df.melt(id_vars = df.columns[0], value_vars = df.columns[1:], var_name="movie_name", value_name="ratin
melt_df
```

Out[13]:

	user_id	movie_name	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 [14]:
```
from surprise import Dataset
```

In [15]:
```
reader = Reader(rating_scale=(-1,10))

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

Dividing the data into training and test data

In [16]:
```
trainset, testset = train_test_split(data, test_size=0.25)
```

Building a recommendation model on training data

In [17]:
```
from surprise import SVD
```

In [27]:
```
algo = SVD()
```

In [28]:
```
algo.fit(trainset)
```

Out[28]:
```
<surprise.prediction_algorithms.matrix_factorization.SVD at 0x2aa0639c320>
```

Making predictions on the test data

In [29]:
```
predictions = algo.test(testset)
```

In [30]:
```
accuracy.rmse(predictions)
```

```
RMSE: 0.2841
```

Out[30]:
```
0.284063748999517
```

In [31]:
```
user_id = 'A3R5OBKS7OM2IR'
muvi_id = 'Movie1'
r_ui = 5.0
algo.predict(user_id, muvi_id, r_ui=r_ui, verbose=True)
```

```
user: A3R5OBKS7OM2IR item: Movie1     r_ui = 5.00   est = -0.01   {'was_impossible': False}
```

Out[31]:
```
Prediction(uid='A3R5OBKS7OM2IR', iid='Movie1', r_ui=5.0, est=-0.009525602596924613, details={'was_impossible':
False})
```

In [32]:
```
from surprise.model_selection import cross_validate
```

In [33]:
```
cross_validate(algo, 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)    0.2843  0.2829  0.2785  0.2819  0.0024
MAE (testset)     0.0424  0.0432  0.0416  0.0424  0.0006
Fit time          60.13   58.37   59.42   59.31   0.72
Test time         4.71    4.22    5.33    4.75    0.46
```

Out[33]:
```
{'test_rmse': array([0.28426991, 0.28289064, 0.27853396]),
 'test_mae': array([0.0424467 , 0.04318182, 0.04163725]),
 'fit_time': (60.127018451690674, 58.37384057044983, 59.417017221450806),
 'test_time': (4.7091333866119385, 4.216424942016602, 5.330347299575806)}
```

In [34]:
```
def repeat(algo_type, frame, min_, max_):
    reader = Reader(rating_scale=(min_, max_))

    data = Dataset.load_from_df(frame, reader=reader)

    algo = algo_type

    print(cross_validate(algo, data, measures=['RMSE', 'MAE'], cv=3, verbose=True))

    print("#"*10)

    user_id = 'A3R5OBKS7OM2IR'
    muvi_id = 'Movie1'
    r_ui = 5.0

    print(algo.predict(user_id, muvi_id, r_ui=r_ui, verbose=True))

    print("#"*10)
    print()
```

In [35]:
```
df = df.iloc[:1212, :50]

melt_df = df.melt(id_vars = df.columns[0], value_vars = df.columns[1:], var_name="movie_name", value_name="ratin
```

In [36]:
```
repeat(SVD(), melt_df.fillna(0), -1, 10)
repeat(SVD(), melt_df.fillna(melt_df.mean()), -1, 10)
repeat(SVD(), melt_df.fillna(melt_df.median()), -1, 10)
```

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

                  Fold 1  Fold 2  Fold 3  Mean    Std
RMSE (testset)    0.4724  0.4454  0.4350  0.4509  0.0158
MAE (testset)     0.1051  0.1023  0.1014  0.1029  0.0015
Fit time          3.36    3.32    3.34    3.34    0.02
Test time         0.17    0.19    0.23    0.20    0.02
{'test_rmse': array([0.472374  , 0.44538185, 0.43495296]), 'test_mae': array([0.10505185, 0.1023354 , 0.101417
]), 'fit_time': (3.362168550491333, 3.322235345840454, 3.338214635848999), 'test_time': (0.17496848106384277,
0.19293642044067383, 0.22793245315551758)}
##########
user: A3R5OBKS7OM2IR item: Movie1     r_ui = 5.00   est = 0.13   {'was_impossible': False}
user: A3R5OBKS7OM2IR item: Movie1     r_ui = 5.00   est = 0.13   {'was_impossible': False}
##########

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

                  Fold 1  Fold 2  Fold 3  Mean    Std
RMSE (testset)    0.0874  0.0901  0.0935  0.0903  0.0025
MAE (testset)     0.0201  0.0206  0.0207  0.0205  0.0003
Fit time          3.59    3.39    3.38    3.45    0.10
Test time         0.27    0.18    0.18    0.21    0.04
{'test_rmse': array([0.08739885, 0.09011852, 0.09352667]), 'test_mae': array([0.02009009, 0.02059292, 0.0207023
9]), 'fit_time': (3.5930655002593994, 3.3876466751098633, 3.382173538208008), 'test_time': (0.2682557106018066
4, 0.17503762245178223, 0.17795944213867188)}
##########
user: A3R5OBKS7OM2IR item: Movie1     r_ui = 5.00   est = 4.67   {'was_impossible': False}
user: A3R5OBKS7OM2IR item: Movie1     r_ui = 5.00   est = 4.67   {'was_impossible': False}
##########

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

                  Fold 1  Fold 2  Fold 3  Mean    Std
RMSE (testset)    0.1003  0.0879  0.1058  0.0980  0.0075
MAE (testset)     0.0199  0.0201  0.0195  0.0198  0.0002
Fit time          3.34    3.35    3.77    3.49    0.20
Test time         0.17    0.65    0.18    0.33    0.23
{'test_rmse': array([0.10031318, 0.08793845, 0.1057768 ]), 'test_mae': array([0.01994149, 0.02005148, 0.0195493
2]), 'fit_time': (3.3400988578796387, 3.3509843349456787, 3.767303943634033), 'test_time': (0.1719667911529541,
0.65281081199646, 0.17696881294250488)}
##########
user: A3R5OBKS7OM2IR item: Movie1     r_ui = 5.00   est = 5.02   {'was_impossible': False}
user: A3R5OBKS7OM2IR item: Movie1     r_ui = 5.00   est = 5.02   {'was_impossible': False}
##########
```

In [37]:
```
from surprise.model_selection import GridSearchCV
```

In [38]:
```
param_grid = { 'n_epochs': [20,30],
               'lr_all':[0.005, 0.01],
               'n_factors': [50,100]}
```

In [39]:
```
gs = GridSearchCV(SVD, param_grid, measures=['rmse','mae'],cv=3)
gs.fit(data)
```

In [40]:
```
gs.best_score
```

Out[40]:
```
{'rmse': 0.27799841511390994, 'mae': 0.04009161669622063}
```

In [ ]: