Imports

Importing the required libraries and the data and displaying a small description of the data.

```
import pandas as pd
import numpy as np

import matplotlib.pyplot as plt
%matplotlib inline

data = pd.read_csv('Amazon - Movies and TV Ratings.csv')
data.head()
```

	user_id	Movie1	Movie2	Movie3	Movie4	Movie5	Movie6	Movie7	Movie8	Movie9	Movie10	Movie11	Movie12	Movie13	Movie14	Movie15	Movie16	Movie17	Movie18
0	A3R5OBKS7OM2IR	5.0	5.0	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
2	A3LKP6WPMP9UKX	NaN	NaN	NaN	5.0	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
4	A1CV1WROP5KTTW	NaN	NaN	NaN	NaN	5.0	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN

5 rows × 207 columns

```
display(data.describe())
```

	Movie1	Movie2	Movie3	Movie4	Movie5	Movie6	Movie7	Movie8	Movie9	Movie10	Movie11	Movie12	Movie13	Movie14	Movie15	Movie16	Movie17	Movie18	Movie19
count	1.0	1.0	1.0	2.0	29.000000	1.0	1.0	1.0	1.0	1.0	2.0	5.0	1.0	1.0	1.0	320.000000	1.0	1.0	2.00000
mean	5.0	5.0	2.0	5.0	4.103448	4.0	5.0	5.0	5.0	5.0	5.0	5.0	5.0	4.0	5.0	4.518750	3.0	5.0	3.50000
std	NaN	NaN	NaN	0.0	1.496301	NaN	NaN	NaN	NaN	NaN	0.0	0.0	NaN	NaN	NaN	0.795535	NaN	NaN	2.12132
min	5.0	5.0	2.0	5.0	1.000000	4.0	5.0	5.0	5.0	5.0	5.0	5.0	5.0	4.0	5.0	1.000000	3.0	5.0	2.00000
25%	5.0	5.0	2.0	5.0	4.000000	4.0	5.0	5.0	5.0	5.0	5.0	5.0	5.0	4.0	5.0	4.000000	3.0	5.0	2.75000
50%	5.0	5.0	2.0	5.0	5.000000	4.0	5.0	5.0	5.0	5.0	5.0	5.0	5.0	4.0	5.0	5.000000	3.0	5.0	3.50000
75%	5.0	5.0	2.0	5.0	5.000000	4.0	5.0	5.0	5.0	5.0	5.0	5.0	5.0	4.0	5.0	5.000000	3.0	5.0	4.25000
max	5.0	5.0	2.0	5.0	5.000000	4.0	5.0	5.0	5.0	5.0	5.0	5.0	5.0	4.0	5.0	5.000000	3.0	5.0	5.00000

8 rows × 206 columns

Exploratory Data Analysis Tasks

Task 1 - Which movies have maximum views/ratings?

For this task, I create a data frame of number of not null ratings for each movie, then find the row(movie) with max ratings.

```
movie_notNAN = pd.DataFrame()
movie_notNAN['Number of Ratings'] = data.notnull().sum(axis=0).drop('user_id')
display(movie_notNAN)
```

	Number of Ratings
Movie1	1
Movie2	1
Movie3	1
Movie4	2
Movie5	29
...	...
Movie202	6
Movie203	1
Movie204	8
Movie205	35
Movie206	13

206 rows × 1 columns

```
print("Movie with Maximum Views/Ratings:\n")
display(movie_notNAN[movie_notNAN['Number of Ratings'] == movie_notNAN['Number of Ratings'].max()])
```

```
Movie with Maximum Views/Ratings:
```

	Number of Ratings
Movie127	2313

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

For this task, I take the mean of all movie ratings along the row, and then find the max five values after sorting them in descending order.

```
movie_notNAN['Average Ratings'] = data.mean(axis=0,skipna=True)
```

```
2 display(movie_notNAN)
```

	Number of Ratings	Average Ratings
Movie1	1	5.000000
Movie2	1	5.000000
Movie3	1	2.000000
Movie4	2	5.000000
Movie5	29	4.103448
...	...	...
Movie202	6	4.333333
Movie203	1	3.000000
Movie204	8	4.375000
Movie205	35	4.628571
Movie206	13	4.923077

206 rows × 2 columns

```
1 display(movie_notNAN.sort_values(by = 'Number of Ratings', ascending=False).head(n=5))
```

	Number of Ratings	Average Ratings
Movie127	2313	4.111976
Movie140	578	4.833910
Movie16	320	4.518750
Movie103	272	4.562500
Movie29	243	4.806584

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

For this task, I sort by the `Number of Ratings` in ascending order and find the first 5.

```
1 display(movie_notNAN.sort_values(by = 'Number of Ratings').head(n=5))
```

	Number of Ratings	Average Ratings
Movie1	1	5.0
Movie71	1	4.0
Movie145	1	5.0
Movie69	1	1.0
Movie68	1	5.0

Tasks for building a 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.

Task 4 - Divide the data into training and test data

- **First, I fill null values with 0, invert the dataframe to get a `user_id` vs `Movie Number` dataframe, and then divide the data.**
- **I've used the `surprise` module from the `scikit-surprise` library.**

```
1 data = data.fillna(0)
2 display(data)
```

	user_id	Movie1	Movie2	Movie3	Movie4	Movie5	Movie6	Movie7	Movie8	Movie9	Movie10	Movie11	Movie12	Movie13	Movie14	Movie15	Movie16	Movie17	Movie
0	A3R5OBKS7OM2IR	5.0	5.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0
1	AH3QC2PC1VTGP	0.0	0.0	2.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0
2	A3LKP6WPMP9UKX	0.0	0.0	0.0	5.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0
3	AVIY68KEPQ5ZD	0.0	0.0	0.0	5.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0
4	A1CV1WROP5KTTW	0.0	0.0	0.0	0.0	5.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
4843	A1IMQ9WMFYKWH5	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0
4844	A1KLIKPUF5E88I	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0
4845	A5HG6WFZLO10D	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0
4846	A3UU690TWXCG1X	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0
4847	AI4J762YI6S06	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0

4848 rows × 207 columns

```
1 data_inverted = pd.melt(data,id_vars='user_id')   #pd.melt() method to combine Movies and user_id in a single row
2 data_inverted.rename(columns={'variable':'Movie Number', 'value':'Rating'}, inplace=True)
3 display(data_inverted)
```

	user_id	Movie Number	Rating
0	A3R5OBKS7OM2IR	Movie1	5.0
1	AH3QC2PC1VTGP	Movie1	0.0
2	A3LKP6WPMP9UKX	Movie1	0.0
3	AVIY68KEPQ5ZD	Movie1	0.0
4	A1CV1WROP5KTTW	Movie1	0.0
...	...	...	...
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

```
display(data_inverted[data_inverted['Movie Number'] == 'Movie69'])   #Random movie
```

	user_id	Movie Number	Rating
329664	A3R5OBKS7OM2IR	Movie69	0.0
329665	AH3QC2PC1VTGP	Movie69	0.0
329666	A3LKP6WPMP9UKX	Movie69	0.0
329667	AVIY68KEPQ5ZD	Movie69	0.0
329668	A1CV1WROP5KTTW	Movie69	0.0
...	...	...	...
334507	A1IMQ9WMFYKWH5	Movie69	0.0
334508	A1KLIKPUF5E88I	Movie69	0.0
334509	A5HG6WFZLO10D	Movie69	0.0
334510	A3UU690TWXCG1X	Movie69	0.0
334511	AI4J762YI6S06	Movie69	0.0

4848 rows × 3 columns

```
display(data_inverted[data_inverted['user_id'] == 'A3LKP6WPMP9UKX']) #random user_id
```

	user_id	Movie Number	Rating
2	A3LKP6WPMP9UKX	Movie1	0.0
4850	A3LKP6WPMP9UKX	Movie2	0.0
9698	A3LKP6WPMP9UKX	Movie3	0.0
14546	A3LKP6WPMP9UKX	Movie4	5.0
19394	A3LKP6WPMP9UKX	Movie5	0.0
...	...	...	...
974450	A3LKP6WPMP9UKX	Movie202	0.0
979298	A3LKP6WPMP9UKX	Movie203	0.0
984146	A3LKP6WPMP9UKX	Movie204	0.0
988994	A3LKP6WPMP9UKX	Movie205	0.0
993842	A3LKP6WPMP9UKX	Movie206	0.0

206 rows × 3 columns

```
!pip install scikit-surprise    #Install recommended library for such tasks
```

```
Collecting scikit-surprise
  Downloading https://files.pythonhosted.org/packages/97/37/5d334adaf5ddd65da99fc65f6507e0e4599d092ba048f4302fe8775619e8/scikit-surprise-1.1.1.tar.gz (11.8MB)
     |████████████████████████████████| 11.8MB 268kB/s
Requirement already satisfied: joblib>=0.11 in /usr/local/lib/python3.6/dist-packages (from scikit-surprise) (0.16.0)
Requirement already satisfied: numpy>=1.11.2 in /usr/local/lib/python3.6/dist-packages (from scikit-surprise) (1.18.5)
Requirement already satisfied: scipy>=1.0.0 in /usr/local/lib/python3.6/dist-packages (from scikit-surprise) (1.4.1)
Requirement already satisfied: six>=1.10.0 in /usr/local/lib/python3.6/dist-packages (from scikit-surprise) (1.15.0)
Building wheels for collected packages: scikit-surprise
  Building wheel for scikit-surprise (setup.py) ... done
  Created wheel for scikit-surprise: filename=scikit_surprise-1.1.1-cp36-cp36m-linux_x86_64.whl size=1670974 sha256=05f773c2ffadb93a62b83b4a65c8cbf4eaa4fa2fb70283c7544901b858
  Stored in directory: /root/.cache/pip/wheels/78/9c/3d/41b419c9d2aff5b6e2b4c0fc8d25c538202834058f9ed110d0
Successfully built scikit-surprise
Installing collected packages: scikit-surprise
Successfully installed scikit-surprise-1.1.1
```

```
from surprise import Reader, Dataset
from surprise.model_selection import train_test_split

reader = Reader()
model_data = Dataset.load_from_df(data_inverted[['user_id','Movie Number','Rating']], reader)

train, test = train_test_split(model_data, test_size = 0.2)
```

Task 5 - Build a recommendation model on training data

I have used a `SVD` model for this particular task.

```
from surprise import SVD, accuracy
model = SVD()
model.fit(train)
```

```
<surprise.prediction_algorithms.matrix_factorization.SVD at 0x7f60e267b978>
```

Task 6 - Make predictions on the test data

For this task, I use the `test()` method for prediction on test data, and show the root mean square error using `rmse()` method of the `accuracy` module.

```
predicted_values = model.test(test)   #Get list of predictions
predicted_values_df = pd.DataFrame(predicted_values)
predicted_values_df.rename(columns={'iid':'Predicted Movie Number', 'uid':'user_id'}, inplace=True)
display(predicted_values_df)
```

	user_id	Predicted Movie Number	r_ui	est	details
0	AQ7V1UO19U1N9	Movie37	0.0	1.0	{'was_impossible': False}
1	AAQB00CK6R301	Movie35	0.0	1.0	{'was_impossible': False}
2	A2T2HPQ16WP4XD	Movie196	0.0	1.0	{'was_impossible': False}
3	A1GLOXDN9N406Z	Movie116	0.0	1.0	{'was_impossible': False}
4	A1FQPOYRBTTK1	Movie146	0.0	1.0	{'was_impossible': False}
...	...	...	...	...	...
199733	A3PHVQMJR57DFY	Movie100	0.0	1.0	{'was_impossible': False}
199734	A2KJ92GG58CZNR	Movie204	0.0	1.0	{'was_impossible': False}
199735	A2NL42GF3KGW9	Movie58	0.0	1.0	{'was_impossible': False}
199736	AWUL6M6NGT6QH	Movie107	0.0	1.0	{'was_impossible': False}
199737	A3M672PZX4TG5O	Movie49	0.0	1.0	{'was_impossible': False}

199738 rows × 5 columns

```
accuracy.rmse(predicted_values)   #Calculate root mean square error
```

```
RMSE: 1.0253
1.0252521305939741
```