Task 1: Introduction

For this project we are going to create a recommendation engine for movies for users based on there past behaviour.

We will focus on the **collaborative filtering** approach, that is:

The user is recommended items that people with similar tastes and preferences liked in the past. In another word, this method predicts unknown ratings by using the similarities between users.

Note: This notebook uses `python 3` and these packages: `pandas`, `numpy`, `matplotlib` and `scikit-surprise`

We can install them using:

```
pip3 install pandas matplotlib numpy scikit-surprise
```

1.1: Installing Libraries

In [55]:
```
print('>> Installing Libraries')

!pip3 install pandas matplotlib numpy scikit-surprise

print('>> Libraries Installed')
```

```
>> Installing Libraries
Requirement already satisfied: pandas in /usr/local/lib/python3.6/dist-packag
es (1.0.5)
Requirement already satisfied: matplotlib in /usr/local/lib/python3.6/dist-pa
ckages (3.2.2)
Requirement already satisfied: numpy in /usr/local/lib/python3.6/dist-package
s (1.18.5)
Requirement already satisfied: scikit-surprise in /usr/local/lib/python3.6/di
st-packages (1.1.1)
Requirement already satisfied: pytz>=2017.2 in /usr/local/lib/python3.6/dist-
packages (from pandas) (2018.9)
Requirement already satisfied: python-dateutil>=2.6.1 in /usr/local/lib/pytho
n3.6/dist-packages (from pandas) (2.8.1)
Requirement already satisfied: pyparsing!=2.0.4,!=2.1.2,!=2.1.6,>=2.0.1 in /u
sr/local/lib/python3.6/dist-packages (from matplotlib) (2.4.7)
Requirement already satisfied: kiwisolver>=1.0.1 in /usr/local/lib/python3.6/
dist-packages (from matplotlib) (1.2.0)
Requirement already satisfied: cycler>=0.10 in /usr/local/lib/python3.6/dist-
packages (from matplotlib) (0.10.0)
Requirement already satisfied: scipy>=1.0.0 in /usr/local/lib/python3.6/dist-
packages (from scikit-surprise) (1.4.1)
Requirement already satisfied: joblib>=0.11 in /usr/local/lib/python3.6/dist-
packages (from scikit-surprise) (0.16.0)
Requirement already satisfied: six>=1.10.0 in /usr/local/lib/python3.6/dist-p
ackages (from scikit-surprise) (1.15.0)
>> Libraries Installed
```

1.2: Importing Libraries

First of all, we will need to import some libraries. This includes surprise which we will use to create the recommendation system.

In [56]:
```
print('>> Importing Libraries')

import pandas as pd

from surprise import Reader, Dataset, SVD

from surprise.accuracy import rmse, mae
from surprise.model_selection import cross_validate

print('>> Libraries imported.')
```

```
>> Importing Libraries
>> Libraries imported.
```

Task 2: Importing Data

We will use open-source dataset from GroupLens Research (movielens.org (http://movielens.org))

2.1: Importing the Data

The dataset is saved in a `ratings.csv` file. We will use pandas to take a look at some of the rows.

In [57]:
```
df = pd.read_csv ("ratings.csv")
df.head()
```

Out[57]:

	userId	movieId	rating	timestamp
0	1	1	4.0	964982703
1	1	3	4.0	964981247
2	1	6	4.0	964982224
3	1	47	5.0	964983815
4	1	50	5.0	964982931

2.2 Dropping timestamp

We won't be using the timestamp when user gave the particular rating. So we will drop that column.

In [58]:
```
df.drop('timestamp', axis=1, inplace=True)
df.head()
```

Out[58]:

	userId	movieId	rating
0	1	1	4.0
1	1	3	4.0
2	1	6	4.0
3	1	47	5.0
4	1	50	5.0

2.3 Check for Missing Data

It's a good practice to check if the data has any missing values. In real world data, this is quite common and must be taken care of before any data pre-processing or model training.

In [59]:
```
df.isna().sum()
```

Out[59]:
```
userId     0
movieId    0
rating     0
dtype: int64
```

Task 3: EDA (Exploratory data analysis)

In statistics, exploratory data analysis is an approach to analyzing data sets to summarize their main characteristics.

3.1 Number of movies/users

In [60]:
```
n_movies = df["movieId"].nunique()
n_users = df["userId"].nunique()

print(f'Number of movies: {n_movies}')
print(f'Number of users: {n_users}')
```

```
Number of movies: 9724
Number of users: 610
```

3.2 Sparsity of our data

Sparsity (%) = `(No of missing values/ (Total Values))X100`

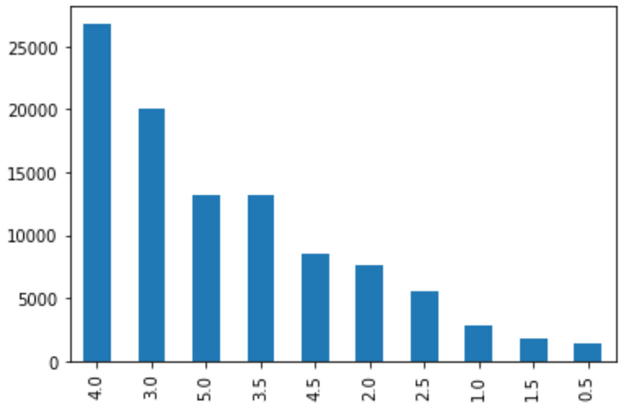
In [61]:
```
available_ratings = df['rating'].count()
total_ratings = n_movies * n_users
missing_ratings = total_ratings - available_ratings
sparsity = (missing_ratings/total_ratings)*100
print(available_ratings, total_ratings, missing_ratings)
print(f'{sparsity}%')
```

```
100836 5931640 5830804
98.30003169443864%
```

3.3 Ratings Distribution

In [62]:
```
df['rating'].value_counts().plot(kind='bar')
```

Out[62]: `<matplotlib.axes._subplots.AxesSubplot at 0x7f00cb3716d8>`



Task 4: Dimensionality Reduction

To reduce the dimensionality of the dataset, we will filter out rarely rated movies and rarely rating users

4.1 Filter movies with less than 3 ratings

In [63]:
```
min_ratings = 3
filter_movies = df['movieId'].value_counts() > min_ratings
filter_movies = filter_movies[filter_movies].index.tolist()
```

4.2 Filter users with less than 3 movies rated

In [64]:
```
min_user_ratings = 3
filter_users = df['userId'].value_counts() > min_user_ratings
filter_users = filter_users[filter_users].index.tolist()
```

4.3 Remove rarely rated movies and rarely rating users

In [65]:
```
print('The original data frame shape:\t{}'.format(df.shape))
df = df[(df['movieId'].isin(filter_movies)) & (df['userId'].isin(filter_user
s))]
print('The new data frame shape:\t{}'.format(df.shape))
```

```
The original data frame shape:  (100836, 3)
The new data frame shape:       (92394, 3)
```

Task 5: Create Training and Test Sets

5.1 Columns used for training

In [66]:
```
cols = ['userId', 'movieId', 'rating']
```

5.2 Create surprise dataset

In [67]:
```
reader = Reader(rating_scale=(0.5, 5))
data = Dataset.load_from_df(df[cols], reader)
```

5.3 Create Train-set and Prediction-set

In [68]:
```
trainset = data.build_full_trainset()
antiset = trainset.build_anti_testset()
```

Task 6: Creating and training the model

6.1 Creating the model

SVD (Singular Value Decomposition)

Interaction Matrix = A X B X C

In [69]:
```
algo = SVD(n_epochs = 25, verbose = True)
```

6.2 Training the model

Mean Absolute Error (MAE): MAE measures the average magnitude of the errors in a set of predictions, without considering their direction.

Root mean squared error (RMSE): RMSE is the square root of the average of squared differences between prediction and actual observation.

In [70]:
```
cross_validate(algo, data, measures=['RMSE', 'MAE'], cv=5, verbose = True)
print('>> Training Done')
```

```
Processing epoch 0
Processing epoch 1
Processing epoch 2
Processing epoch 3
Processing epoch 4
Processing epoch 5
Processing epoch 6
Processing epoch 7
Processing epoch 8
Processing epoch 9
Processing epoch 10
Processing epoch 11
Processing epoch 12
Processing epoch 13
Processing epoch 14
Processing epoch 15
Processing epoch 16
Processing epoch 17
Processing epoch 18
Processing epoch 19
Processing epoch 20
Processing epoch 21
Processing epoch 22
Processing epoch 23
Processing epoch 24
Processing epoch 0
Processing epoch 1
Processing epoch 2
Processing epoch 3
Processing epoch 4
Processing epoch 5
Processing epoch 6
Processing epoch 7
Processing epoch 8
Processing epoch 9
Processing epoch 10
Processing epoch 11
Processing epoch 12
Processing epoch 13
Processing epoch 14
Processing epoch 15
Processing epoch 16
Processing epoch 17
Processing epoch 18
Processing epoch 19
Processing epoch 20
Processing epoch 21
Processing epoch 22
Processing epoch 23
Processing epoch 24
Processing epoch 0
Processing epoch 1
Processing epoch 2
Processing epoch 3
Processing epoch 4
Processing epoch 5
Processing epoch 6
Processing epoch 7
Processing epoch 8
Processing epoch 9
Processing epoch 10
Processing epoch 11
Processing epoch 12
Processing epoch 13
Processing epoch 14
Processing epoch 15
Processing epoch 16
Processing epoch 17
```

Task 7: Predictions

7.1 Predict ratings for all pairs (user, items) that are NOT in the training set.

In [71]:
```
predictions = algo.test(antiset)
```

In [72]:
```
predictions[0]
```

Out[72]:
```
Prediction(uid=1, iid=318, r_ui=3.529119856267723, est=5, details={'was_impos
sible': False})
```

7.2 Recommending top 3 movies movies based on predictions

In [73]:
```
from collections import defaultdict
def get_top_n(predictions, n):
    # First map the predictions to each user.
    top_n = defaultdict(list)
    for uid, iid, true_r, est, _ in predictions:
        top_n[uid].append((iid, est))

    # Then sort the predictions for each user and retrieve the n highest one
s.
    for uid, user_ratings in top_n.items():
        user_ratings.sort(key=lambda x: x[1], reverse=True)
        top_n[uid] = user_ratings[:n]

    return top_n
top_n = get_top_n(predictions, n=3)
for uid, user_ratings in top_n.items():
    print(uid, [iid for (iid, rating) in user_ratings])
```

```
1 [318, 1704, 48516]
2 [912, 260, 750]
3 [1204, 91529, 3578]
4 [1204, 3030, 955]
5 [1201, 58559, 912]
6 [1270, 58, 916]
7 [1204, 2571, 1203]
8 [2324, 1262, 1261]
9 [318, 898, 4973]
10 [4878, 1262, 1270]
11 [1148, 1283, 1235]
12 [47, 50, 110]
13 [57669, 750, 1204]
14 [1104, 1283, 2324]
15 [50, 1208, 1223]
16 [1204, 2324, 1283]
17 [1283, 1204, 1276]
18 [750, 3030, 1204]
19 [858, 296, 8874]
20 [1262, 50, 1204]
21 [898, 265, 1234]
22 [110, 2571, 1136]
23 [1204, 898, 3451]
24 [1203, 1283, 912]
25 [50, 899, 2324]
26 [1104, 898, 1283]
27 [318, 912, 2324]
28 [1193, 1104, 2324]
29 [912, 1223, 260]
30 [904, 750, 4011]
31 [1204, 1201, 58559]
32 [1204, 6350, 4226]
33 [1136, 912, 1204]
34 [912, 2692, 105504]
35 [2019, 5618, 1136]
36 [2959, 1104, 1248]
37 [1248, 1204, 912]
38 [1204, 3681, 3266]
39 [912, 541, 1266]
40 [5690, 1208, 953]
41 [122926, 3275, 4993]
42 [1208, 27773, 898]
43 [50, 223, 260]
44 [1283, 750, 1204]
45 [1223, 318, 1248]
46 [904, 527, 356]
47 [1204, 260, 44195]
48 [922, 912, 1283]
49 [898, 912, 1136]
50 [4993, 16, 4973]
51 [1136, 1258, 2858]
52 [50, 2329, 3147]
53 [47, 50, 110]
54 [4993, 4973, 3030]
55 [318, 58559, 3468]
56 [57669, 904, 527]
57 [1262, 79132, 5747]
58 [260, 1204, 750]
59 [4993, 1248, 1199]
60 [904, 912, 260]
61 [904, 168252, 110]
62 [1204, 750, 50]
63 [750, 1954, 1212]
64 [912, 5618, 1041]
65 [898, 904, 1248]
66 [1276, 750, 904]
67 [1204, 1276, 3468]
68 [115569, 1283, 1204]
```