BUILDING USER-BASED RECOMMENDATION MODEL FOR AMAZON

Here we are developing a recommendation system for amazon based on the user input

DESCRIPTION ABOUT PROJECT:

The dataset provided contains movie reviews given by Amazon customers.

Reviews were given between May 1996 and July 2014.

*Data Dictionary UserID – 4848 customers who provided a rating for each movie.

Movie 1 to Movie 206 – 206 movies for which ratings are provided by 4848 distinct users

*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.

TASKS TO BE PERFORMED:

1.Exploratory Data Analysis:

2.Which movies have maximum views/ratings?

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

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

- 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.

5.Divide the data into training and test data

6.Build a recommendation model on training data

7.Make predictions on the test data

1.Exploratory data analysis(EDA):

```
#importing important libraries
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
```

```
#reading the amazon dataset
df=pd.read_csv('amazon.csv')
```

```
#displaying the dataset
df
```

	user_id	Movie1	Movie2	Movie3	Movie4	Movie5	Movie6	Movie7	Movie8	Movie9	Movie10	Movie11	Mov
0	A3R5OBKS7OM2IR	5.0	5.0	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	
1	AH3QC2PC1VTGP	NaN	NaN	2.0	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	
2	A3LKP6WPMP9UKX	NaN	NaN	NaN	5.0	NaN	NaN	NaN	NaN	NaN	NaN	NaN	
3	AVIY68KEPQ5ZD	NaN	NaN	NaN	5.0	NaN	NaN	NaN	NaN	NaN	NaN	NaN	
4	A1CV1WROP5KTTW	NaN	NaN	NaN	NaN	5.0	NaN	NaN	NaN	NaN	NaN	NaN	
...	...	...	...	...	...	...	...	...	...	...	...	...	...
4843	A1IMQ9WMFYKWH5	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	
4844	A1KLIKPUF5E88I	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	
4845	A5HG6WFZLO10D	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	
4846	A3UU690TWXCG1X	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	
4847	AI4J762YI6S06	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	

4848 rows × 207 columns

```
#displaying the top 5 rows using head function
df.head()
```

	user_id	Movie1	Movie2	Movie3	Movie4	Movie5	Movie6	Movie7	Movie8	Movie9	Movie10	Movie11	Movie1
0	A3R5OBKS7OM2IR	5.0	5.0	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	Na
1	AH3QC2PC1VTGP	NaN	NaN	2.0	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	Na

2	A3LKP6WPMP9UKX	NaN	NaN	NaN	5.0	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	Na
3	AVIY68KEPQ5ZD	NaN	NaN	NaN	5.0	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	Na
4	A1CV1WROP5KTTW	NaN	NaN	NaN	NaN	5.0	NaN	NaN	NaN	NaN	NaN	NaN	NaN	Na

5 rows × 207 columns

```
df.tail()
```

	user_id	Movie1	Movie2	Movie3	Movie4	Movie5	Movie6	Movie7	Movie8	Movie9	Movie10	Movie11	Mov
4843	A1IMQ9WMFYKWH5	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	
4844	A1KLIKPUF5E88I	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	
4845	A5HG6WFZLO10D	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	
4846	A3UU690TWXCG1X	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	
4847	AI4J762YI6S06	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	

5 rows × 207 columns

```
df.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
```

```
df.describe()
```

	Movie1	Movie2	Movie3	Movie4	Movie5	Movie6	Movie7	Movie8	Movie9	Movie10	Movie11	Movie12	Movie13	M

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
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
std	NaN	NaN	NaN	0.0	1.496301	NaN	NaN	NaN	NaN	NaN	0.0	0.0	NaN
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
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
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
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
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

8 rows × 206 columns

```
df.shape
```

```
(4848, 207)
```

2.Which movies have maximum views/ratings?

```
#get the top 10 movies with maximum number of ratings/views
df.describe().T["count"].sort_values(ascending = False)[0:10]
```

```
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
Name: count, dtype: float64
```

Q:Which movies have maximum views/ratings?

Answer: Movie127 has the maximum no.of ratings of 2313.

```
df_original = df


# Drop the user_id
df = df.drop('user_id',axis=1)


df
```

	Movie1	Movie2	Movie3	Movie4	Movie5	Movie6	Movie7	Movie8	Movie9	Movie10	Movie11	Movie12	Movie13	Movi
0	5.0	5.0	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	N
1	NaN	NaN	2.0	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	N

2	NaN	NaN	NaN	5.0	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	N
3	NaN	NaN	NaN	5.0	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	N
4	NaN	NaN	NaN	NaN	5.0	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	N

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

4845	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	N

```
# Get the movie with highest rating
movie_ratings = df.sum().sort_values(ascending=False).head(5)
movie_ratings
```

```
Movie127    9511.0
Movie140    2794.0
Movie16     1446.0
Movie103    1241.0
Movie29     1168.0
dtype: float64
```

Movie127 has the highest rating

```
#the average rating for each movie
df.mean().sort_values(ascending=False)
```

```
Movie1      5.0
Movie66     5.0
Movie76     5.0
Movie75     5.0
Movie74     5.0
            ...
Movie58     1.0
Movie60     1.0
Movie154    1.0
Movie45     1.0
Movie144    1.0
```

```
Length: 206, dtype: float64
```

Though movies 1, 66, 76 etc have avg. rating 5, they cannot be termed as movies with highest rating, since only few people have rated/viewed these movies.

```
# Fill the missing values with 0
df = df.fillna(0)
df
```

	Movie1	Movie2	Movie3	Movie4	Movie5	Movie6	Movie7	Movie8	Movie9	Movie10	Movie11	Movie12	Movie13	Movi
0	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	

4 Define the top 5 movies with the least audience

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

```
# Top 5 movies having least audience
df.describe().T['mean'].sort_values(ascending=True)[:5]
```

```
Movie67     0.000206
Movie154    0.000206
Movie58     0.000206
Movie60     0.000206
Movie45     0.000206
Name: mean, dtype: float64
```

4846 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

MODEL BUILDING

Surprise is a Python scikit for building and analyzing recommender systems.

We use matrix factorization and SVD (Singular Value Decomposition) to build this recommendation system.

We can think of all the ratings for movies by users as a matrix R. This matrix R can be factored into 2 smaller matrices.

Using SVD, we can get these smaller matrices.

Once we have these smaller matrices, we can predict the rating of any movie by any user by taking a dot product of these matrices.
!pip install surprise

```
pip install scikit-surprise
```

```
Requirement already satisfied: scikit-surprise in /usr/local/lib/python3.7/dist-packages (1.1.1)
Requirement already satisfied: six>=1.10.0 in /usr/local/lib/python3.7/dist-packages (from scikit-surprise) (1.15.0)
Requirement already satisfied: joblib>=0.11 in /usr/local/lib/python3.7/dist-packages (from scikit-surprise) (1.1.0)
Requirement already satisfied: numpy>=1.11.2 in /usr/local/lib/python3.7/dist-packages (from scikit-surprise) (1.19.5)
Requirement already satisfied: scipy>=1.0.0 in /usr/local/lib/python3.7/dist-packages (from scikit-surprise) (1.4.1)
```

```
import surprise
```

5.Divide the data into training and test data

```
from surprise import Reader
from surprise import Dataset
from surprise import SVD

from surprise.model_selection import train_test_split


df_original
```

	user_id	Movie1	Movie2	Movie3	Movie4	Movie5	Movie6	Movie7	Movie8	Movie9	Movie10	Movie11	Mov
0	A3R5OBKS7OM2IR	5.0	5.0	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	
1	AH3QC2PC1VTGP	NaN	NaN	2.0	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	

To use Surprise, the dataset object should contain the following fields in this order:user_id,movie, rating.

To achive this format, we use melt function.

The melt() function is used to unpivot the DataFrame from wide format to long format.

```
unpivot_df=df_original.melt(id_vars= df_original.columns[0],value_vars=df_original.columns[1:],var_name='Movie',value_name='
```

```
unpivot_df
```

```
unpivot_df = unpivot_df.fillna(0)
unpivot_df.shape
```

```
(998688, 3)
```

```
3          AVIY68KEPQ5ZD      Movie1      NaN
```

```
# step 1: create a reader. Reader tells SVD the lower and upper bounds of rating.
reader = Reader(rating_scale=(-1,10))
# sep 2: the dataset should contain the following fields in this order:user_id,movie, rating.
data = Dataset.load_from_df(unpivot_df, reader=reader)
```

```
#Divide the data into training and test data. keep 25% data for test.
train, test = train_test_split(data, test_size=0.25)
```

6.Build a recommendation model on training data

```
# train the svd with 100 latent features (number chosen arbitrarily)
svd = SVD(n_factors=100)
svd.fit(train)
```

```
<surprise.prediction_algorithms.matrix_factorization.SVD at 0x7f5bcd123990>
```

7.Make predictions on the test data

```
#Make predictions on the test data
predict= svd.test(test)
pd.DataFrame(predict).sort_values(ascending=False,by='est')
```

	uid	iid	r_ui	est	details
81667	A3LHVMF6OJXJNM	Movie127	0.0	3.329176	{'was_impossible': False}
178021	A27RJ30RN5K9MX	Movie127	5.0	3.278995	{'was_impossible': False}

243132	A1Z6CDRFVIHES5	Movie127	0.0	3.199882	{'was_impossible': False}
8142	AX9GYP3BMRQV3	Movie127	0.0	3.142771	{'was_impossible': False}
66033	A2DO13VLX1OL85	Movie127	0.0	3.141808	{'was_impossible': False}
...	...	...	...	...	...
226835	A1DKTO0RCVGDQK	Movie140	0.0	-0.734657	{'was_impossible': False}
40067	A34PAZQ73SL163	Movie16	0.0	-0.759890	{'was_impossible': False}
5989	A2E005KLWD9UH9	Movie16	0.0	-0.829433	{'was_impossible': False}
110512	A2T4IKODFTRVT	Movie140	0.0	-0.862628	{'was_impossible': False}
243936	AFFK11HUGFK75	Movie140	0.0	-0.979156	{'was_impossible': False}

Above dataframe shows the original rating(r_ui) and estimated rating (est) for each movie and user.

```
from surprise.model_selection import cross_validate
```

```
cross_validate(svd,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)    0.2801  0.2869  0.2785  0.2819  0.0036
MAE (testset)     0.0423  0.0429  0.0423  0.0425  0.0003
Fit time          44.36   46.63   43.00   44.66   1.49
Test time         6.02    5.00    4.50    5.18    0.63
{'fit_time': (44.36073327064514, 46.62818646430969, 43.00430130958557),
 'test_mae': array([0.04232092, 0.04286598, 0.04225997]),
 'test_rmse': array([0.28013813, 0.28693356, 0.27852625]),
 'test_time': (6.022913694381714, 5.003541469573975, 4.499566555023193)}
```

```
# Predict a rating for a movie by an user:
```

```
# Example User ID ATGE1GPHD7BAN and movie id Movie127
svd.predict('ATGE1GPHD7BAN','Movie127')
```

```
Prediction(uid='ATGE1GPHD7BAN', iid='Movie127', r_ui=None, est=4.553436455943398, details={'was_impossible': False})
```

Final Result: the prediction object has est - which shows the estimated rating for the movie by the user.

The estimated rating for the Movie127 by user id 'ATGE1GPHD7BAN' is 4.76.


THANK
YOU

✓ 0s completed at 3:45 PM