Name: olwethu matiwane

Email: jimmyolwethu7@gmail.com

BUILDING USER-BASED RECOMMENDATION MODEL FOR AMAZON - PROJECT 2

In [31]:
```
import numpy as np
import pandas as pd
import scipy as sp
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split
from sklearn import linear_model
from sklearn import metrics
from sklearn.linear_model import LogisticRegression
from sklearn import neighbors
from sklearn.tree import DecisionTreeClassifier
from sklearn.svm import SVC
```

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

In [5]:
```
amazon_data.head()
```

Out[5]:

	user_id	Movie1	Movie2	Movie3	Movie4	Movie5	Movie6	Movie7	Movie8	Movie9	...	Movie197	Movie198
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

In [34]:
```
# Use some Statitics analysis to get an idea of the Given data e.g mean,std, Min and Max etc
amazon_data.describe()
```

Out[34]:

	Movie1	Movie2	Movie3	Movie4	Movie5	Movie6	Movie7	Movie8	Movie9	Movie10	...	Movie197	Movie198
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
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
std	NaN	NaN	NaN	0.0	1.496301	NaN	NaN	NaN	NaN	NaN	...	1.643168	0.0
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
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
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
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
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

In [35]:
```
# Now to Check shape of the Dataset in Rows and Columns respectively
amazon_data.shape
```

Out[35]:
```
(4848, 207)
```

In [36]:
```
# Then to Make sure we can check for the Total Number of Observations or Size of the Data
amazon_data.size
```

Out[36]: 1003536

In [37]:
```
amazon_data.isna().any().head()
```

Out[37]:
```
user_id    False
Movie1      True
Movie2      True
Movie3      True
Movie4      True
dtype: bool
```

In [38]:
```
type(amazon_data)
```

Out[38]: pandas.core.frame.DataFrame

In [15]:
```
# displaying movies which have maximum views/ratings
amazon_data.max().head(10)
```

Out[15]:
```
user_id    AZZ1KF8RAO1BR
Movie1                 5
Movie2                 5
Movie3                 2
Movie4                 5
Movie5                 5
Movie6                 4
Movie7                 5
Movie8                 5
Movie9                 5
dtype: object
```

In [29]:
```
amazon_data1 = amazon_data.max(axis=0,skipna=True)
print("Movies with Max rating amongst all movies are :")
amazon_data1.head(10)
```

```
Movies with Max rating amongst all movies are :
```

Out[29]:
```
user_id    AZZ1KF8RAO1BR
Movie1                 5
Movie2                 5
Movie3                 2
Movie4                 5
Movie5                 5
Movie6                 4
Movie7                 5
Movie8                 5
Movie9                 5
dtype: object
```

In [23]:
```
# displaying the avarage rating for each movie
amazon_data.mean().head()
```

Out[23]:
```
Movie1    5.000000
Movie2    5.000000
Movie3    2.000000
Movie4    5.000000
Movie5    4.103448
dtype: float64
```

In [31]:
```
amazon_data3 = amazon_data.mean(axis=0,skipna=True)
print("Movie with mean rating amongst all movies are :")
amazon_data3.head(10)
```

```
Movie with mean rating amongst all movies are :
```

Out[31]:
```
Movie1     5.000000
Movie2     5.000000
Movie3     2.000000
Movie4     5.000000
Movie5     4.103448
Movie6     4.000000
Movie7     5.000000
Movie8     5.000000
Movie9     5.000000
Movie10    5.000000
dtype: float64
```

In [25]:
```
amazon_data.min().head(10)
```

Out[25]:
```
user_id    A0047322388NOTO4N8SKD
Movie1                         5
Movie2                         5
Movie3                         2
Movie4                         5
Movie5                         1
Movie6                         4
Movie7                         5
Movie8                         5
Movie9                         5
dtype: object
```

In [18]:
```
amazon_dataa0 = amazon_data.idxmax(axis=0,skipna=True)
amazon_dataa1 = amazon_data[1:].idxmax(axis=0,skipna=True)
amazon_dataa2 = amazon_data[2:].idxmax(axis=0,skipna=True)
amazon_dataa3 = amazon_data[4:].idxmax(axis=0,skipna=True)
amazon_dataa4 = amazon_data[5:].idxmax(axis=0,skipna=True)

print('My Top 5 Rated Movies are :',amazon_dataa0,',',amazon_dataa1,',',
      amazon_dataa2,',',amazon_dataa3,',',amazon_dataa4)
```

```
My Top 5 Rated Movies are : Movie1 , Movie2 , Movie4 , Movie5 , Movie7
```

In [30]:
```
amazon_data2 = amazon_data.min(axis=0,skipna=True)
print("Movie with min rating amongst all movies are :")
amazon_data2.head(10)
```

```
Movie with min rating amongst all movies are :
```

Out[30]:
```
user_id    A0047322388NOTO4N8SKD
Movie1                         5
Movie2                         5
Movie3                         2
Movie4                         5
Movie5                         1
Movie6                         4
Movie7                         5
Movie8                         5
Movie9                         5
dtype: object
```

In [53]:
```
# replacing NaN with 0s
amazon_data1 = amazon_data2.fillna(0)
```

In [57]:
```
amazon_data1.head()
```

Out[57]:

	user_id	Movie1	Movie2	Movie3	Movie4	Movie5	Movie6	Movie7	Movie8	Movie9	...	Movie197	Movie198
0	A3R5OBKS7OM2IR	5.0	5.0	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
2	A3LKP6WPMP9UKX	0.0	0.0	0.0	5.0	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
4	A1CV1WROP5KTTW	0.0	0.0	0.0	0.0	5.0	0.0	0.0	0.0	0.0	...	0.0	0.0

In [58]:
```
counts = amazon_data.count()
counts.head(10)
```

Out[58]:
```
user_id    4848
Movie1        1
Movie2        1
Movie3        1
Movie4        2
Movie5       29
Movie6        1
Movie7        1
Movie8        1
Movie9        1
dtype: int64
```

In [63]:
```
# checking the percentage of missing values in each variable
amazon_data.isnull().sum()/len(amazon_data)*100
```

Out[63]:
```
user_id      0.000000
Movie1      99.979373
Movie2      99.979373
Movie3      99.979373
Movie4      99.958746
Movie5      99.401815
Movie6      99.979373
Movie7      99.979373
Movie8      99.979373
Movie9      99.979373
Movie10     99.979373
```

In [64]:
```
clean=amazon_data.fillna('')
clean.head()
```

Out[64]:

	user_id	Movie1	Movie2	Movie3	Movie4	Movie5	Movie6	Movie7	Movie8	Movie9	...	Movie197	Movie198
0	A3R5OBKS7OM2IR	5	5								...		
1	AH3QC2PC1VTGP			2							...		
2	A3LKP6WPMP9UKX				5						...		
3	AVIY68KEPQ5ZD				5						...		
4	A1CV1WROP5KTTW					5					...		

5 rows × 207 columns

In [67]:
```
user = amazon_data['user_id'].fillna('')
user.head()
```

Out[67]:
```
0    A3R5OBKS7OM2IR
1     AH3QC2PC1VTGP
2    A3LKP6WPMP9UKX
3     AVIY68KEPQ5ZD
4    A1CV1WROP5KTTW
Name: user_id, dtype: object
```

In [68]:
```
del amazon_data1['user_id']
```

In [69]:
```
x = amazon_data1.iloc[:,1:]
print(x.head())
```

```
   Movie2  Movie3  Movie4  Movie5  Movie6  Movie7  Movie8  Movie9  Movie10  \
0     5.0     0.0     0.0     0.0     0.0     0.0     0.0     0.0      0.0
1     0.0     2.0     0.0     0.0     0.0     0.0     0.0     0.0      0.0
2     0.0     0.0     5.0     0.0     0.0     0.0     0.0     0.0      0.0
3     0.0     0.0     5.0     0.0     0.0     0.0     0.0     0.0      0.0
4     0.0     0.0     0.0     5.0     0.0     0.0     0.0     0.0      0.0

   Movie11  ...  Movie197  Movie198  Movie199  Movie200  Movie201  Movie202  \
0      0.0  ...       0.0       0.0       0.0       0.0       0.0       0.0
1      0.0  ...       0.0       0.0       0.0       0.0       0.0       0.0
2      0.0  ...       0.0       0.0       0.0       0.0       0.0       0.0
3      0.0  ...       0.0       0.0       0.0       0.0       0.0       0.0
4      0.0  ...       0.0       0.0       0.0       0.0       0.0       0.0

   Movie203  Movie204  Movie205  Movie206
0       0.0       0.0       0.0       0.0
1       0.0       0.0       0.0       0.0
2       0.0       0.0       0.0       0.0
3       0.0       0.0       0.0       0.0
4       0.0       0.0       0.0       0.0

[5 rows x 205 columns]
```

In [70]:
```
user_No= np.linspace(1,4848,4848)
print(user_No)
```

```
[1.000e+00 2.000e+00 3.000e+00 ... 4.846e+03 4.847e+03 4.848e+03]
```

In [74]:
```
movies = amazon_data1.iloc[:,:1]
print(movies.head())
```

```
   Movie1
0     5.0
1     0.0
2     0.0
3     0.0
4     0.0
```

In [75]:
```
# Then assign the X and Y respectively
x_feature = user_No
y_target = movies
```

In [77]:
```
from sklearn.model_selection import train_test_split
x_train, x_test, y_train, y_test = train_test_split(x_feature,y_target,test_size = 0.2)
```

In [78]:
```
x_train
```

Out[78]:
```
array([4044.,  344., 3498., ...,  731., 3852., 4315.])
```

```
In [80]: print(x_train.shape)
         print(y_train.shape)
         print(x_test.shape)
         print(y_test.shape)
```

```
(3878,)
(3878, 1)
(970,)
(970, 1)
```

```
In [127]: # testing
          predic = model.predict(x_test)
          predic
```

```
Out[127]: array([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., 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., 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., 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., 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., 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., 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., 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., 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., 0., 0., 0., 0., 0., 0., 0., 0.,
```

```
In [128]: # evaluation
          # confusion matrix
          print(metrics.confusion_matrix(y_test,predic))
```

```
[[970]]
```

```
In [129]: # classfification report
          print(metrics.classification_report(y_test, predic))
```

```
              precision    recall  f1-score   support

         0.0       1.00      1.00      1.00       970

    accuracy                           1.00       970
   macro avg       1.00      1.00      1.00       970
weighted avg       1.00      1.00      1.00       970
```