Building user-based recommendation model for Amazon.

DESCRIPTION

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.

Analysis Task

- Exploratory Data Analysis:

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

Divide the data into training and test data Build a recommendation model on training data Make predictions on the test data

Load Dataset

In [1]:
```python
import numpy as np
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
```

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

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

Out[3]:

	user_id	Movie1	Movie2	Movie3	Movie4	Movie5	Movie6	Movie7	Movie8	Movie9	...	Movie197	Movie198	Movie199	Mo
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 rows × 207 columns

Explotaray Data Analysis

In [4]:
```python
df.shape
```

Out[4]:
```
(4848, 207)
```

In [5]:
```python
df['user_id'].value_counts()
```

Out[5]:
```
AU9XHH0D4PG4J     1
A2ASHBJCV8PVNW    1
A29UDNSUXVM28X    1
A3CHMPBVLG1H7W    1
AWQ6YZC7Z8J87     1
                 ..
A2KH3EOZO3DRCO    1
A1FZ4YSYCTX3Z6    1
A6G6R1KABLJ49     1
A289FKI0KVB097    1
A1Y28SBS1NUC0Z    1
Name: user_id, Length: 4848, dtype: int64
```

In [6]:
```python
df[df.columns].isna().sum()/df.shape[0]
```

Out[6]:
```
user_id     0.000000
Movie1      0.999794
Movie2      0.999794
Movie3      0.999794
Movie4      0.999587
              ...
Movie202    0.998762
Movie203    0.999794
Movie204    0.998350
Movie205    0.992781
Movie206    0.997318
Length: 207, dtype: float64
```

In [7]:
```python
#filling all nan with zero
df_ordered=df.fillna(0)
```

In [8]:
```python
df_ordered.head()
```

Out[8]:

	user_id	Movie1	Movie2	Movie3	Movie4	Movie5	Movie6	Movie7	Movie8	Movie9	...	Movie197	Movie198	Movie199	Mo
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	
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	
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	
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	
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	

5 rows × 207 columns

In [9]:
```python
#making the numbered ordered user_id
df_ordered['user_id']=np.arange(df_ordered.shape[0])
df_ordered.head()
```

Out[9]:

	user_id	Movie1	Movie2	Movie3	Movie4	Movie5	Movie6	Movie7	Movie8	Movie9	...	Movie197	Movie198	Movie199	Movie200	Movi
0	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	
1	1	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	
2	2	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	
3	3	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	
4	4	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	

5 rows × 207 columns

Which movies have maximum views/ratings? What is the average rating for each movie? Define the top 5 movies with the maximum ratings. Define the top 5 movies with the least audience.

Average Rating of Each Movie is Shown in Rating columns

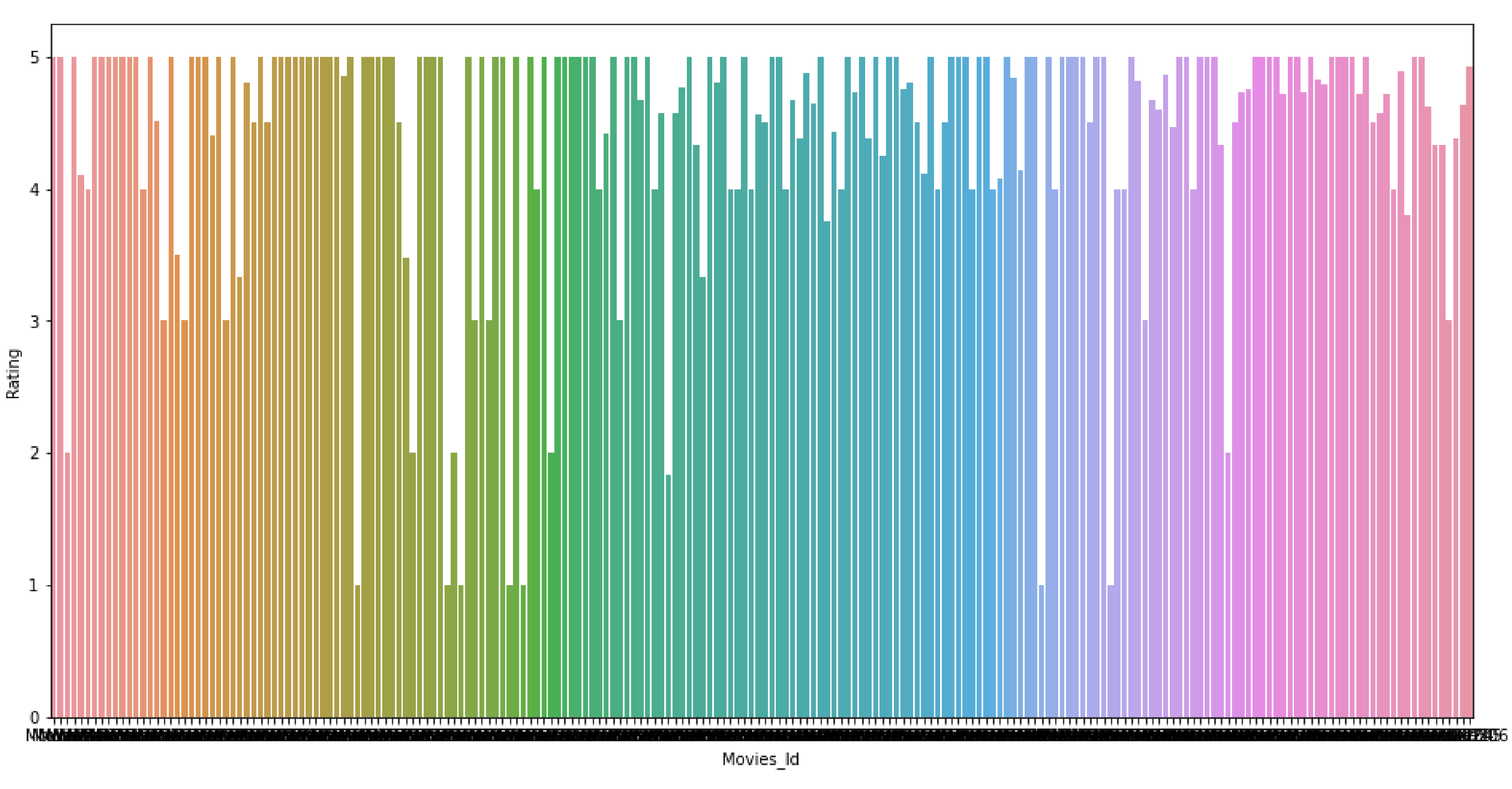
In [10]:
```python
df_max=pd.DataFrame()
df_max['Rating']= df[df.columns].drop(columns=['user_id']).mean()
df_max['views']=df[df.columns].drop(columns=['user_id']).count()
df_max['Movies_Id']=df_max.index
df_max['view_upon_rating']=df_max['views']/df_max['Rating']
df_max.head(10)
```

Out[10]:

	Rating	views	Movies_Id	view_upon_rating
Movie1	5.000000	1	Movie1	0.200000
Movie2	5.000000	1	Movie2	0.200000
Movie3	2.000000	1	Movie3	0.500000
Movie4	5.000000	2	Movie4	0.400000
Movie5	4.103448	29	Movie5	7.067227
Movie6	4.000000	1	Movie6	0.250000
Movie7	5.000000	1	Movie7	0.200000
Movie8	5.000000	1	Movie8	0.200000
Movie9	5.000000	1	Movie9	0.200000
Movie10	5.000000	1	Movie10	0.200000

In [11]:
```python
plt.figure(figsize=(16,8))
sns.barplot(data=df_max,y='Rating',x='Movies_Id')
```

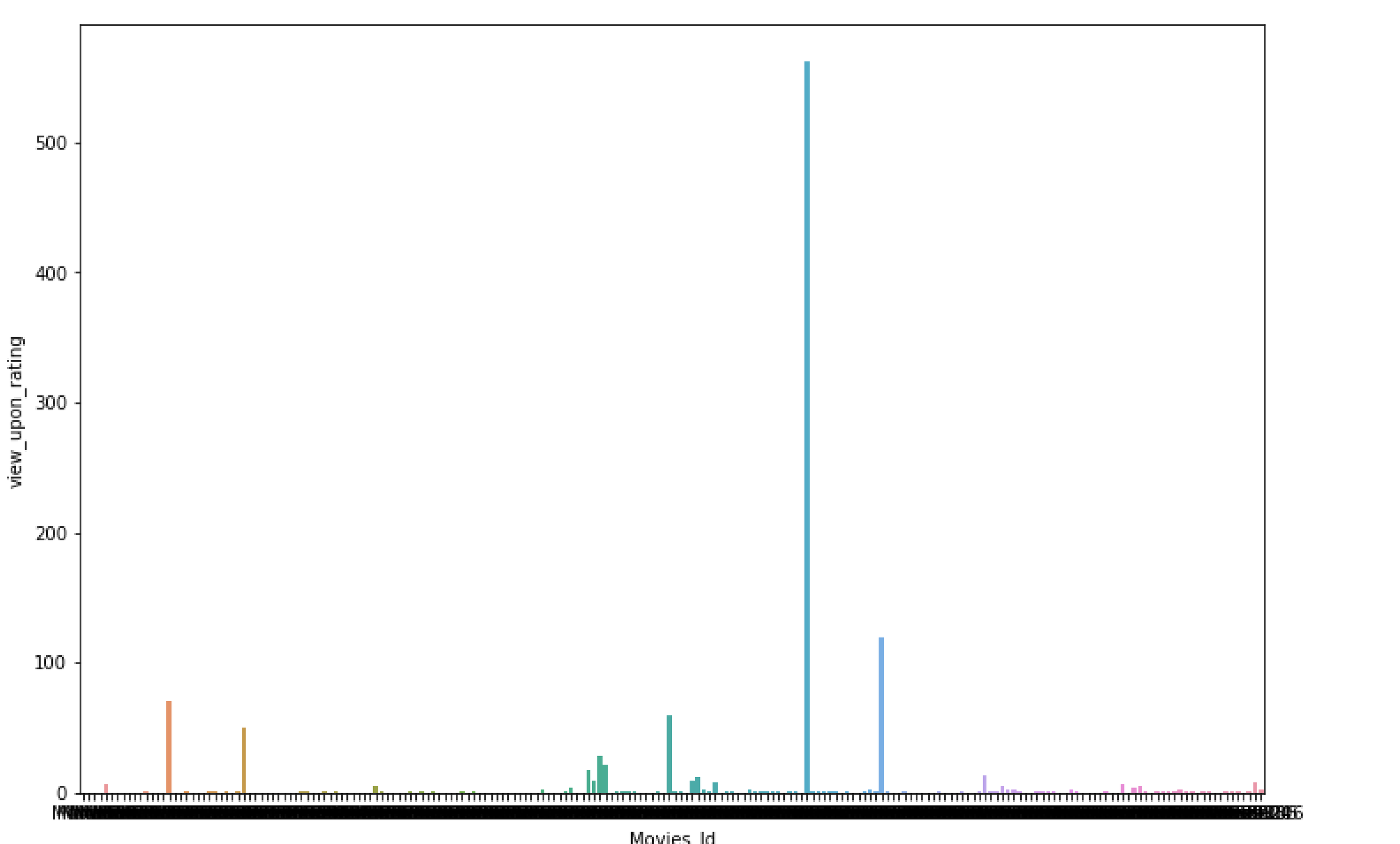
Out[11]:
```
<AxesSubplot:xlabel='Movies_Id', ylabel='Rating'>
```



Top 5 with Max Views/Ratings

In [12]:
```python
plt.figure(figsize=(12,8))
sns.barplot(data=df_max,y='view_upon_rating',x='Movies_Id')
```

Out[12]:
```
<AxesSubplot:xlabel='Movies_Id', ylabel='view_upon_rating'>
```



In [13]:
```python
df_max.sort_values(['view_upon_rating'],ascending=False).head(5)
```

Out[13]:

	Rating	views	Movies_Id	view_upon_rating
Movie127	4.111976	2313	Movie127	562.503312
Movie140	4.833910	578	Movie140	119.571940
Movie16	4.518750	320	Movie16	70.816044
Movie103	4.562500	272	Movie103	59.616438
Movie29	4.806584	243	Movie29	50.555651

Top 5 With least Viewers

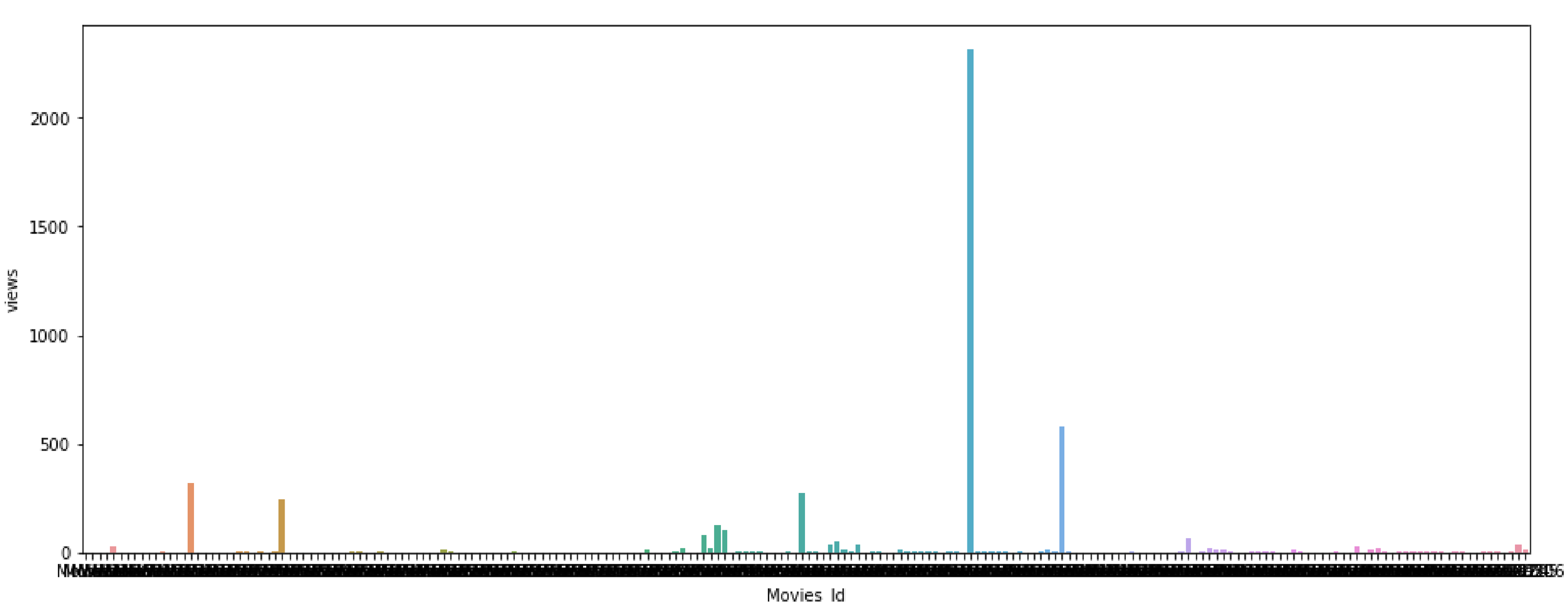
In [14]:
```python
df_max.sort_values(['views','view_upon_rating'],ascending=True).head(5)
```

Out[14]:

	Rating	views	Movies_Id	view_upon_rating
Movie1	5.0	1	Movie1	0.2
Movie2	5.0	1	Movie2	0.2
Movie7	5.0	1	Movie7	0.2
Movie8	5.0	1	Movie8	0.2
Movie9	5.0	1	Movie9	0.2

In [15]:
```python
plt.figure(figsize=(16,6))
sns.barplot(data=df_max,y='views',x='Movies_Id')
```

Out[15]:
```
<AxesSubplot:xlabel='Movies_Id', ylabel='views'>
```



Top 5 Maximum Rating

In [16]:
```python
df_max.sort_values(['Rating','views'],ascending=False).head(5)
```

Out[16]:

	Rating	views	Movies_Id	view_upon_rating
Movie186	5.0	9	Movie186	1.8
Movie188	5.0	6	Movie188	1.2
Movie191	5.0	6	Movie191	1.2
Movie12	5.0	5	Movie12	1.0
Movie101	5.0	5	Movie101	1.0

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.

In [17]:
```python
from sklearn.model_selection import train_test_split
train_df,test_df=train_test_split(df_ordered,test_size=0.25,random_state=42)
```

In [18]:
```python
no_user= df.shape[0]
no_movies=df.shape[1]-1
```

In [19]:
```python
no_user,no_movies
```

Out[19]:
```
(4848, 206)
```

In [20]:
```python
test_df[test_df['user_id'].isin([0,1,4846,4847])]
```

Out[20]:

	user_id	Movie1	Movie2	Movie3	Movie4	Movie5	Movie6	Movie7	Movie8	Movie9	...	Movie197	Movie198	Movie199	Movie200	M
4847	4847	0.0	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 rows × 207 columns

In [21]:
```python
train_mat=np.zeros((no_user,no_movies))
for line in train_df.itertuples():
        for i in range(1,no_movies+1):
            train_mat[line[0],i-1]=line[i+1]

train_mat
```

Out[21]:
```
array([[5., 5., 0., ..., 0., 0., 0.],
       [0., 0., 2., ..., 0., 0., 0.],
       [0., 0., 0., ..., 0., 0., 0.],
       ...,
       [0., 0., 0., ..., 0., 0., 5.],
       [0., 0., 0., ..., 0., 0., 5.],
       [0., 0., 0., ..., 0., 0., 0.]])
```

In [22]:
```python
test_mat=np.zeros((no_user,no_movies))
for line in test_df.itertuples():
        for i in range(1,no_movies+1):
            test_mat[line[0],i-1]=line[i+1]

test_mat
```

Out[22]:
```
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., 5.]])
```

In [23]:
```python
from sklearn.metrics import pairwise_distances
def RatingPredictionModel(mat):
    user_similarity = pairwise_distances(mat,metric='cosine')
    movie_similarity = pairwise_distances(mat.T,metric='cosine')
    mean_user_rating= mat.mean(axis=1)[:,np.newaxis]
    ratings_diff = (mat-mean_user_rating)
    user_pred = mean_user_rating + user_similarity.dot(ratings_diff)/np.array([(np.abs(user_similarity).sum(axi
    return user_pred
```

In [24]:
```python
pred_train=RatingPredictionModel(train_mat)
pred_train
```

Out[24]:
```
array([[ 0.03211675,  0.03211675,  0.03252938, ...,  0.0391314 ,
         0.05522383,  0.04119453],
       [-0.00569464, -0.00569464, -0.00672621, ...,  0.00028844,
         0.01638087,  0.00235157],
       [ 0.0088733 ,  0.0088733 ,  0.00825423, ...,  0.01485761,
         0.03095336,  0.01692117],
       ...,
       [ 0.00888513,  0.00888513,  0.00826517, ...,  0.0148781 ,
         0.03099714,  0.00785186],
       [ 0.00888513,  0.00888513,  0.00826517, ...,  0.0148781 ,
         0.03099714,  0.00785186],
       [-0.01540539, -0.01540539, -0.01602432, ..., -0.0094223 ,
         0.00667013, -0.00735917]])
```

In [25]:
```python
pred_test=RatingPredictionModel(test_mat)
pred_test
```

Out[25]:
```
array([[-0.00552439, -0.00552439, -0.00552439, ..., -0.00531807,
         0.00479127, -0.00139812],
       [-0.00552439, -0.00552439, -0.00552439, ..., -0.00531807,
         0.00479127, -0.00139812],
       [-0.00552439, -0.00552439, -0.00552439, ..., -0.00531807,
         0.00479127, -0.00139812],
       ...,
       [-0.00552439, -0.00552439, -0.00552439, ..., -0.00531807,
         0.00479127, -0.00139812],
       [-0.00552439, -0.00552439, -0.00552439, ..., -0.00531807,
         0.00479127, -0.00139812],
       [ 0.01876408,  0.01876408,  0.01876408, ...,  0.01897052,
         0.02908613,  0.01876408]])
```

Hence the required model is RatingPredictionModel and required matrix is pred_test

In [ ]: