CS422 - Data Mining

Rutul Mehta - A20476293

Problem: 1

In [1]:
```python
import numpy as np
import pandas as pd
import math

from sklearn import metrics
```

In [2]:
```python
# Import rating and movie data
user_cols = ['user_id',
             'age',
             'gender',
             'occupation',
             'zip_code']

users = pd.read_csv('ml-100k/u.user',
                    sep='|',
                    names=user_cols)
```

In [3]:
```python
rating_cols = ['user_id',
               'movie_id',
               'rating',
               'timestamp']

ratings = pd.read_csv('ml-100k/u.data',
                      sep='\t',
                      names=rating_cols)
```

In [4]:
```python
item_cols = ['movie id',
             'movie title',
             'release date',
             'video release date',
             'IMDb URL',
             'Unknown',
             'Action',
             'Adventure',
             'Animation',
             'Childrens',
             'Comedy',
             'Crime',
             'Documentary',
             'Drama',
             'Fantasy',
             'FilmNoir',
             'Horror',
             'Musical',
             'Mystery',
             'Romance',
             'SciFi',
             'Thriller',
             'War',
             'Western']

items = pd.read_csv('ml-100k/u.item',
                    sep='|',
                    names=item_cols,
                    encoding='latin-1')
```

In [5]:
```python
# contain all the information about movie
items
```

Out[5]:

	movie id	movie title	release date	video release date	IMDb URL	Unknown	Action	Adventure	Animation	Childrens	...	Fantasy	Filml
0	1	Toy Story (1995)	01-Jan-1995	NaN	http://us.imdb.com/M/title-exact?Toy%20Story%2...	0	0	0	1	1	...	0	
1	2	GoldenEye (1995)	01-Jan-1995	NaN	http://us.imdb.com/M/title-exact?GoldenEye%20(...	0	1	1	0	0	...	0	
2	3	Four Rooms (1995)	01-Jan-1995	NaN	http://us.imdb.com/M/title-exact?Four%20Rooms%...	0	0	0	0	0	...	0	
3	4	Get Shorty (1995)	01-Jan-1995	NaN	http://us.imdb.com/M/title-exact?Get%20Shorty%...	0	1	0	0	0	...	0	
4	5	Copycat (1995)	01-Jan-1995	NaN	http://us.imdb.com/M/title-exact?Copycat%20(1995)	0	0	0	0	0	...	0	
...	...	...	...	...	...	...	...	...	...	...	...	...	
1677	1678	Mat' i syn (1997)	06-Feb-1998	NaN	http://us.imdb.com/M/title-exact?Mat%27+i+syn+...	0	0	0	0	0	...	0	
1678	1679	B. Monkey (1998)	06-Feb-1998	NaN	http://us.imdb.com/M/title-exact?B%2E+Monkey+(...	0	0	0	0	0	...	0	
1679	1680	Sliding Doors (1998)	01-Jan-1998	NaN	http://us.imdb.com/Title?Sliding+Doors+(1998)	0	0	0	0	0	...	0	
1680	1681	You So Crazy (1994)	01-Jan-1994	NaN	http://us.imdb.com/M/title-exact?You%20So%20Cr...	0	0	0	0	0	...	0	
1681	1682	Scream of Stone (Schrei aus Stein) (1991)	08-Mar-1996	NaN	http://us.imdb.com/M/title-exact?Schrei%20aus%...	0	0	0	0	0	...	0	

1682 rows × 24 columns

In [6]:
```python
# Here we find the discription of the movie and their ratings
ratings
```

Out[6]:

	user_id	movie_id	rating	timestamp
0	196	242	3	881250949
1	186	302	3	891717742
2	22	377	1	878887116
3	244	51	2	880606923
4	166	346	1	886397596
...	...	...	...	...
99995	880	476	3	880175444
99996	716	204	5	879795543
99997	276	1090	1	874795795
99998	13	225	2	882399156
99999	12	203	3	879959583

100000 rows × 4 columns

In [7]:
```python
# This is the pivot table which represent the rating of each and every users range from 0-5.
description = ratings.pivot(index='user_id',columns='movie_id',values='rating')
description
```

Out[7]:

movie_id	1	2	3	4	5	6	7	8	9	10	...	1673	1674	1675	1676	1677	1678	1679	1680	1681	1682
user_id																					
1	5.0	3.0	4.0	3.0	3.0	5.0	4.0	1.0	5.0	3.0	...	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
2	4.0	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	2.0	...	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
3	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	...	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
4	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	...	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
5	4.0	3.0	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	...	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
939	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	5.0	NaN	...	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
940	NaN	NaN	NaN	2.0	NaN	NaN	4.0	5.0	3.0	NaN	...	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
941	5.0	NaN	NaN	NaN	NaN	NaN	4.0	NaN	NaN	NaN	...	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
942	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	...	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
943	NaN	5.0	NaN	NaN	NaN	NaN	NaN	NaN	3.0	NaN	...	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN

943 rows × 1682 columns

In [8]:
```python
# Take mean of movie to subtract from all values to remove bias
mean = description.mean(axis=1)
mean.head()
```

Out[8]:
```
user_id
1    3.610294
2    3.709677
3    2.796296
4    4.333333
5    2.874286
dtype: float64
```

In [9]:
```python
# remove the null values with 0
Centered = description - mean
Centered = Centered.where((pd.notnull(Centered)),0)
Centered.head()
```

Out[9]:

	1	2	3	4	5	6	7	8	9	10	...	1673	1674	1675	167
user_id															
1	1.389706	-0.709677	1.203704	-1.333333	0.125714	1.364929	0.034739	-2.79661	0.727273	-1.206522	...	0.0	0.0	0.0	0
2	0.389706	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.00000	0.000000	-2.206522	...	0.0	0.0	0.0	0
3	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.00000	0.000000	0.000000	...	0.0	0.0	0.0	0
4	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.00000	0.000000	0.000000	...	0.0	0.0	0.0	0
5	0.389706	-0.709677	0.000000	0.000000	0.000000	0.000000	0.000000	0.00000	0.000000	0.000000	...	0.0	0.0	0.0	0

5 rows × 1682 columns

In [10]:
```python
New = description.where((pd.notnull(description)),0)
```

In [11]:
```python
item1 = items[94:95]
item1.head()
```

Out[11]:

	movie id	movie title	release date	video release date	IMDb URL	Unknown	Action	Adventure	Animation	Childrens	...	Fantasy	FilmNoir
94	95	Aladdin (1992)	01-Jan-1992	NaN	http://us.imdb.com/M/title-exact?Aladdin%20(1992)	0	0	0	1	1	...	0	0

1 rows × 24 columns

In [12]:
```python
feat1 = item1.iloc[:,5:24]
feat1.head()
```

Out[12]:

	Unknown	Action	Adventure	Animation	Childrens	Comedy	Crime	Documentary	Drama	Fantasy	FilmNoir	Horror	Musical	Myst
94	0	0	0	1	1	1	0	0	0	0	0	0	1	

In [13]:
```python
itemProfile = items.iloc[:,5:24]
itemProfile = itemProfile.apply(np.int64)
itemProfile.head()
```

Out[13]:

	Unknown	Action	Adventure	Animation	Childrens	Comedy	Crime	Documentary	Drama	Fantasy	FilmNoir	Horror	Musical	Myste
0	0	0	0	1	1	1	0	0	0	0	0	0	0	
1	0	1	1	0	0	0	0	0	0	0	0	0	0	
2	0	0	0	0	0	0	0	0	0	0	0	0	0	
3	0	1	0	0	0	1	0	0	1	0	0	0	0	
4	0	0	0	0	0	0	1	0	1	0	0	0	0	

In [14]:
```python
New = New.apply(np.int64)
```

In [15]:
```python
# user profile generated
userProfile = np.dot(New,itemProfile)
print("\nUser Profile:\n", userProfile)
```

```
User Profile:
 [[  4 250 123 ... 188  92  22]
 [  0  38  13 ...  43  11   0]
 [  0  39  14 ...  53  14   0]
 ...
 [  0  38  27 ...  28   5   0]
 [  0  74  52 ...  80  47  14]
 [  0 227 114 ... 134  53  23]]
```

In [16]:
```python
userProfile200 = userProfile[199]
print("\nUser Profile (200):\n", userProfile200)
```

```
User Profile (200):
 [  0 332 235  66 166 193  37   2 251  41  10  44  72  15 148 188 201  73
  16]
```

In [17]:
```python
userProfile15 = userProfile[14]
print("\nUser Profile (15):\n", userProfile15)
```

```
User Profile (15):
 [  0  59  36   2  13  75  15   0 153  10   6   2   4  17  86  32  59  34
   0]
```

In [18]:
```python
cosine = metrics.pairwise.cosine_similarity(userProfile,feat1)
```

In [19]:
```python
# Cosine similarity of both the users
print("Cosine Similarity (User 200):", cosine[199])
print("Cosine Similarity (User 15): ", cosine[14])
```

```
Cosine Similarity (User 200): [0.38745727]
Cosine Similarity (User 15):  [0.21517341]
```

In [20]:
```python
# Cosine distance of both the users
print("Cosine Distance (User 200):  ", 1-cosine[199])
print("Cosine Distance (User 15):   ", 1-cosine[14])
```

```
Cosine Distance (User 200):   [0.61254273]
Cosine Distance (User 15):    [0.78482659]
```

Conclusion :

Cosine similarity is a used to test weather two vectors are similar or not. Therefore, recommender system would recommend movie 95 to user 200.

Problem 2

In [21]:
```python
New
```

Out[21]:

movie_id	1	2	3	4	5	6	7	8	9	10	...	1673	1674	1675	1676	1677	1678	1679	1680	1681	1682
user_id																					
1	5	3	4	3	3	5	4	1	5	3	...	0	0	0	0	0	0	0	0	0	0
2	4	0	0	0	0	0	0	0	0	2	...	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	0	0	0	0	0	0
4	0	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0	0	0
5	4	3	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0	0	0
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
939	0	0	0	0	0	0	0	0	5	0	...	0	0	0	0	0	0	0	0	0	0
940	0	0	0	2	0	0	4	5	3	0	...	0	0	0	0	0	0	0	0	0	0
941	5	0	0	0	0	0	4	0	0	0	...	0	0	0	0	0	0	0	0	0	0
942	0	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0	0	0
943	0	5	0	0	0	0	0	0	3	0	...	0	0	0	0	0	0	0	0	0	0

943 rows × 1682 columns

In [22]:
```python
Rest = New.iloc[1:,]
Rest.head()
```

Out[22]:

movie_id	1	2	3	4	5	6	7	8	9	10	...	1673	1674	1675	1676	1677	1678	1679	1680	1681	1682
user_id																					
2	4	0	0	0	0	0	0	0	0	2	...	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	0	0	0	0	0	0
4	0	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0	0	0
5	4	3	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0	0	0
6	4	0	0	0	0	0	2	4	4	0	...	0	0	0	0	0	0	0	0	0	0

5 rows × 1682 columns

In [23]:
```python
user1 = New.iloc[:1,]
user1.head()
```

Out[23]:

movie_id	1	2	3	4	5	6	7	8	9	10	...	1673	1674	1675	1676	1677	1678	1679	1680	1681	1682
user_id																					
1	5	3	4	3	3	5	4	1	5	3	...	0	0	0	0	0	0	0	0	0	0

1 rows × 1682 columns

In [24]:
```python
#To find the similar users, cosine of all users with respect to user 1.
cosine = metrics.pairwise.cosine_similarity(user1,Rest)
```

In [25]:
```python
top10 = np.argpartition(cosine, -10, axis=1)[:, -10:]
#top 10 values
count = 0
ratings = [0,0,0,0,0,0,0,0,0,0]
print(top10)
```

```
[[274 736 427 301 455 266  90 862 433 914]]
```

In [26]:
```python
#take item 508 from all similar users. Remove those users who didn't rate on item 508
for i in range(len(top10[0])):
    ratings[i] = Rest[508][top10[0][i]]
    if ratings[i] != 0.0:
        count+=1
```

In [27]:
```python
total = math.fsum(ratings)
mean = total/count
```

In [28]:
```python
print("User 1 will rate item 508 as : ",mean)
```

```
User 1 will rate item 508 as :   4.0
```