Recommendation Systems

1. Give a formal description of this application in terms of Task, Experience, and Performance. (Make it a well posed problem)

Task(T): : Recommend items to users based on similarity of ratings provided by other users who bought items same as to that of a particular customer.

Experience (E): Dataset of different items with different user rating.

Performance(P): RMSE, Root mean square error value between actual rating and predicted rating.

In [128…
```
import pandas as pd
```

In [5]:
```
amazon_rating = pd.read_excel("rating-users.xlsx")
amazon_rating
```

Out[5]:

	Item	User	Rating
0	098949232X	A1GG51FWU0XQYH	5
1	098949232X	AVFIDS9RK38E0	5
2	098949232X	A2S4AVR5SJ7KMI	5
3	098949232X	AEMMMVOR9BFLI	5
4	098949232X	A2DZXMBTY7KLYP	5
...	...	...	...
2495	9707716436	A38M17GVUSEF2J	2
2496	9707716436	A3I7I0MKM53JUC	5
2497	9707716436	AO7G5C1J62LDD	4
2498	9707716436	A3QZFC96BQL09I	2
2499	9707716436	A3KZU79DHJGB6D	1

2500 rows × 3 columns

A dataset is provided consisting of the ratings of various products from Amazonusers. Write a function Utility_matrix(data) that converts the uploaded dataset'amazon_rating' into an Utility matrix where columns represent items and rowsrepresent users and the values represent rating.

In [6]:
```
def utility(name):
    #data= pd.read_excel(name)
    new_data=name.pivot(index="User",columns="Item")['Rating']
    my_data=new_data
    #new_data = new_data.fillna("")
    return my_data
```

In [24]:
```
original=utility(amazon_rating)
```

```
original
```

Out[24]:

Item	1060297744	1060697254	1610121147	3993854748	5891061139	5891090
User						
A0617213KGAVUMXH6NK4	NaN	NaN	NaN	NaN	NaN	N
A0755549VZ3OU6OE9EHO	NaN	NaN	NaN	NaN	NaN	N
A100C9FK1V6VVT	NaN	NaN	NaN	NaN	NaN	N
A103RLAWEHYFHB	NaN	NaN	NaN	NaN	NaN	N
A103XTS7PCURDJ	NaN	NaN	NaN	NaN	NaN	N
...	...	...	...	...	...	
AZRLKXHT3AV2U	NaN	NaN	NaN	NaN	NaN	N
AZSP9XAX38DG0	NaN	NaN	NaN	NaN	NaN	N
AZVWF96X0IXHJ	NaN	NaN	NaN	NaN	NaN	N
AZW6WE7UXAMU0	NaN	NaN	NaN	NaN	NaN	N
AZYXGC2G6GM71	NaN	NaN	NaN	NaN	NaN	N

2472 rows × 37 columns

In [86]:

```
original.loc["A0617213KGAVUMXH6NK4"][1060297744] = 2
original.loc["A0617213KGAVUMXH6NK4"][7508492919] = 1
original.loc["A0755549VZ3OU6OE9EHO"][1610121147] = 4
original.loc["A0755549VZ3OU6OE9EHO"][7508492919] = 1
original.loc["A0617213KGAVUMXH6NK4"][5891090295] = 5
original.loc["A0755549VZ3OU6OE9EHO"][5891090295] = 5
original.loc["A0755549VZ3OU6OE9EHO"][1060297744] = 2
original.loc["A0755549VZ3OU6OE9EHO"][7508492919] = 2
original.loc["AZRLKXHT3AV2U"][7508492919] = 3
original.loc["AZRLKXHT3AV2U"][1610121147] = 4
original.loc["A100C9FK1V6VVT"][7508492919] = 2
original.loc["A100C9FK1V6VVT"][1610121147] = 1
```

Write a function Normalize(U) to normalize the ratings of the users for the items.

In [87]:

```
def Normalize (name):
        normalize1=name.copy()
        normalize1=normalize1.fillna(0)
        #print(normalize1)
        normalizen=name.copy()
        #print(normalize)
        normalize1['count']=normalize1.astype(bool).sum(axis=1)

        normalize1['sum']=normalizen. sum(axis=1)
        normalize1['average']=normalize1['sum']/normalize1['count']

        normalizednew=name.sub(normalize1['average'], axis=0)
        print(normalizednew)
        return normalizednew,normalize1
```

In [129…
```
normalize=Normalize(original)
q
```

Item	1060297744	1060697254	1610121147	3993854748	\
User					
A0617213KGAVUMXH6NK4	-1.25	NaN	NaN	NaN	
A0755549VZ3OU6OE9EHO	-1.25	NaN	0.750000	NaN	
A100C9FK1V6VVT	NaN	NaN	-0.333333	NaN	
A103RLAWEHYFHB	NaN	NaN	NaN	NaN	
A103XTS7PCURDJ	NaN	NaN	NaN	NaN	
...	...	...	...	...	
AZRLKXHT3AV2U	NaN	NaN	0.000000	NaN	
AZSP9XAX38DG0	NaN	NaN	NaN	NaN	
AZVWF96X0IXHJ	NaN	NaN	NaN	NaN	
AZW6WE7UXAMU0	NaN	NaN	NaN	NaN	
AZYXGC2G6GM71	NaN	NaN	NaN	NaN	

Item	5891061139	5891090295	7391002801	7508492919	\
User					
A0617213KGAVUMXH6NK4	NaN	1.75	NaN	-2.250000	
A0755549VZ3OU6OE9EHO	NaN	1.75	NaN	-1.250000	
A100C9FK1V6VVT	NaN	NaN	NaN	0.666667	
A103RLAWEHYFHB	NaN	NaN	NaN	NaN	
A103XTS7PCURDJ	NaN	NaN	NaN	NaN	
...	...	...	...	...	
AZRLKXHT3AV2U	NaN	NaN	NaN	-1.000000	
AZSP9XAX38DG0	NaN	NaN	NaN	NaN	
AZVWF96X0IXHJ	NaN	NaN	NaN	NaN	
AZW6WE7UXAMU0	NaN	NaN	NaN	NaN	
AZYXGC2G6GM71	NaN	NaN	NaN	NaN	

Item	7532385086	7887421268	...	9678315173	9707716371	\
User			...			
A0617213KGAVUMXH6NK4	NaN	NaN	...	NaN	NaN	
A0755549VZ3OU6OE9EHO	NaN	NaN	...	NaN	NaN	
A100C9FK1V6VVT	NaN	NaN	...	NaN	NaN	
A103RLAWEHYFHB	NaN	NaN	...	NaN	NaN	
A103XTS7PCURDJ	NaN	NaN	...	0.0	NaN	
...	...	...	...	...	...	
AZRLKXHT3AV2U	NaN	NaN	...	NaN	NaN	
AZSP9XAX38DG0	NaN	NaN	...	NaN	NaN	
AZVWF96X0IXHJ	NaN	NaN	...	NaN	NaN	
AZW6WE7UXAMU0	NaN	NaN	...	NaN	NaN	
AZYXGC2G6GM71	NaN	0.0	...	NaN	NaN	

Item	9707716436	098949232X	399889988X	819960459X	\
User					
A0617213KGAVUMXH6NK4	NaN	NaN	NaN	NaN	
A0755549VZ3OU6OE9EHO	NaN	NaN	NaN	NaN	
A100C9FK1V6VVT	NaN	NaN	NaN	NaN	
A103RLAWEHYFHB	NaN	NaN	NaN	NaN	
A103XTS7PCURDJ	NaN	NaN	NaN	NaN	
...	...	...	...	...	
AZRLKXHT3AV2U	NaN	NaN	NaN	NaN	
AZSP9XAX38DG0	NaN	0.0	NaN	NaN	
AZVWF96X0IXHJ	NaN	NaN	NaN	NaN	
AZW6WE7UXAMU0	NaN	NaN	NaN	NaN	
AZYXGC2G6GM71	NaN	NaN	NaN	NaN	

Item	828885382X	828886922X	961301375X	962886436X
User				
A0617213KGAVUMXH6NK4	NaN	NaN	NaN	NaN
A0755549VZ3OU6OE9EHO	NaN	NaN	NaN	NaN
A100C9FK1V6VVT	NaN	NaN	NaN	NaN
A103RLAWEHYFHB	NaN	NaN	NaN	NaN

```
A103XTS7PCURDJ              NaN         NaN         NaN         NaN
...                         ...         ...         ...         ...
AZRLKXHT3AV2U               NaN         NaN         NaN         NaN
AZSP9XAX38DG0               NaN         NaN         NaN         NaN
AZVWF96X0IXHJ               NaN         NaN         NaN         NaN
AZW6WE7UXAMU0               NaN         NaN         NaN         NaN
AZYXGC2G6GM71               NaN         NaN         NaN         NaN

[2472 rows x 37 columns]
```

Out[129…

Item	1060297744	1060697254	1610121147	3993854748	5891061139	5891090
User						
A0617213KGAVUMXH6NK4	-1.25	NaN	NaN	NaN	NaN	
A0755549VZ3OU6OE9EHO	-1.25	NaN	0.750000	NaN	NaN	
A100C9FK1V6VVT	NaN	NaN	-0.333333	NaN	NaN	N
A103RLAWEHYFHB	NaN	NaN	NaN	NaN	NaN	N
A103XTS7PCURDJ	NaN	NaN	NaN	NaN	NaN	N
...	...	...	...	...	...	
AZRLKXHT3AV2U	NaN	NaN	0.000000	NaN	NaN	N
AZSP9XAX38DG0	NaN	NaN	NaN	NaN	NaN	N
AZVWF96X0IXHJ	NaN	NaN	NaN	NaN	NaN	N
AZW6WE7UXAMU0	NaN	NaN	NaN	NaN	NaN	N
AZYXGC2G6GM71	NaN	NaN	NaN	NaN	NaN	N

2472 rows × 37 columns

In [74]:
```
normalize
```

Out[74]:

Item	1060297744	1060697254	1610121147	3993854748	5891061139	5891090
User						
A0617213KGAVUMXH6NK4	2.0	0.0	0.0	0.0	0.0	
A0755549VZ3OU6OE9EHO	0.0	0.0	4.0	0.0	0.0	
A100C9FK1V6VVT	0.0	0.0	1.0	0.0	0.0	
A103RLAWEHYFHB	0.0	0.0	0.0	0.0	0.0	
A103XTS7PCURDJ	0.0	0.0	0.0	0.0	0.0	
...	...	...	...	...	...	
AZRLKXHT3AV2U	0.0	0.0	4.0	0.0	0.0	
AZSP9XAX38DG0	0.0	0.0	0.0	0.0	0.0	
AZVWF96X0IXHJ	0.0	0.0	0.0	0.0	0.0	
AZW6WE7UXAMU0	0.0	0.0	0.0	0.0	0.0	
AZYXGC2G6GM71	0.0	0.0	0.0	0.0	0.0	

2472 rows × 40 columns

In [89]:
```
import math
import numpy as np
```

Write a function PearsonCorr(x,y) to find the similarity between the ratings of the items rated by both users x and y

In [110…
```
def pearson_correlation(x,y):
    normalizen = q.copy()
    normalizen = normalizen.fillna(0)
    #print(normalizen)
    rxs=normalizen.loc[x]
    #print(rxs)
    rys=normalizen.loc[y]
    #print(rys)
    rx_mean=normalize.at[x, 'average']
    ry_mean=normalize.at[y, 'average']
    #print(ry_mean)
    x1= np.array(rxs)
    y1 =np.array(rys)

    numerator_product = 0
    denominator_product = 0
    for i in range(len(x1)):
        #print(y1[i])
        if (x1[i] != 0 and y1[i] != 0):
            rx_diff = (x1[i] -rx_mean)
            ry_diff = (y1[i] -ry_mean)
            #print( rx_sub_avg)
            #print(ry_sub_avg)
            #print(x1[i])
            #print(y1[i])
            numerator_product = numerator_product + rx_diff * ry_diff
            #print(num_product)
            #print(num_product)
            #print(int(math.pow (rx_sub_avg, 2)))
            #print(int(math.pow (ry_sub_avg, 2)))
            denominator_product = denominator_product + (int(math.pow (rx_diff, 2))) *
            #print(deno_product)
    numerator = numerator_product
    denominator = np.sqrt( denominator_product)

    if (denominator == 0):
        similarity = 0
    else:
        similarity=numerator/denominator
        similarity=similarity.astype(np.int)
        similarity= similarity.tolist()
    return similarity
```

In [111…
```
k=pearson_correlation('A0617213KGAVUMXH6NK4','A0755549VZ3OU6OE9EHO')
print(k)
```

1

In [14]:
```
import ast
```

In [17]:
```
import operator
```

Write a function NearestNbrs(U,q,k) that takes the normalized utility matrix 'U',the query 'q' as the rating vector of a user, and finds the best 'k' neighbours from'U' based on the similarity metric Pearson Correlation coefficient

In [114…
```
def NearestNbrs(U,q,k):
    similar=list()
    matrix=utility(U)
    similar_dict={}
    rating_vector=q
    num_neighbors=k
    #print(len(matrix)
    for row in matrix.index:
         similar_dict[row]=pearson_correlation(rating_vector,row )
    sorted_similar_dict = dict(sorted(sim_dict.items(),key=operator.itemgetter(1),rever

    count=k
    neighbours=[]
    for k,v in sorted_similar_dict.items():
        if count!=0:
            neighbours.append((k,v))
            count-=1
    return neighbours
```

In [115…
```
NearestNbrs(amazon_rating,'A0617213KGAVUMXH6NK4',2)
```

Out[115… `[('A0617213KGAVUMXH6NK4', 1), ('A0755549VZ3OU6OE9EHO', 1)]`

Write a function PredictRating(x,s) that predicts the rating of user x for item i based on the following formula

In [123…
```
def predict_ratings(data,x,item,n):#4th
    similar_users=NearestNbrs(data,x,n)
    numerator=0
    denominator=0
    predicted_rating=0
    #print(similar_users)
    for neigbour,sim_score in similar_users:
        #print(score)
        #print(normalize.loc[u][item])
        numerator+= sim_score*normalize.loc[neigbour][item]
        #print(numerator)
        denominator+=sim_score
        predicted_rating = numerator/denominator
    return predicted_rating
```

In [124…
```
predict_ratings(amazon_rating,'A0617213KGAVUMXH6NK4',5891090295,3)
```

Out[124… `3.3333333333333335`

In [127…
```
orginal_value=normalize.loc['A0617213KGAVUMXH6NK4'][5891090295]
orginal_value
```

Out[127… `5.0`

```
#using 2 arguments
```

In [ ]:

```
def predict_ratings(x,item):#4th
    similar_users=NearestNbrs(amazon_rating,x,2)
    numerator=0
    denominator=0
    predicted_rating=0
    #print(similar_users)
    for neigbour,sim_score in similar_users:
        #print(score)
        #print(normalize.loc[u][item])
        numerator+= sim_score*normalize.loc[neigbour][item]
        #print(numerator)
        denominator+=sim_score
        predicted_rating = numerator/denominator
    return predicted_rating
```