A Recommendation System aims at suggesting optimal recommendation of items to various users. Collaborative Filtering approach is based on the ratings of the users for various items in a domain.

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

TASK: Predict the rating for products from a user.

EXPERIENCE: The past of data about different users and their rating for various products.

PERFORMANCE: The difference between the rating predicted and the actual rating is low.

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

```
from google.colab import files
uploaded = files.upload()
```

Choose Files rating-users.xlsx

- **rating-users.xlsx**(application/vnd.openxmlformats-officedocument.spreadsheetml.sheet) - 81752 bytes, last modified: 2/21/2021 - 100% done

```
Saving rating-users.xlsx to rating-users.xlsx
```

```
import pandas as pd
import math
import numpy as np
import time
import heapq
```

```
# loading the excel as dataframe
data = pd.read_excel("rating-users.xlsx")
data.head()
```

	Item	User	Rating
0	098949232X	A1GG51FWU0XQYH	5
1	098949232X	AVFIDS9RK38E0	5
2	098949232X	A2S4AVR5SJ7KMI	5
3	098949232X	AEMMMVOR9BFLI	5
4	098949232X	A2DZXMBTY7KLYP	5

```
def utility(df_name):
  """
  This function changes the dataframe such that the columns represent items and rows represe
  """
  new_df = df_name.pivot(index="User",columns="Item")
  return new_df

# Utility function call
new_data1 =  utility(data)
new_data1.shape
```

```
(2472, 37)
```

```
new_data1.head(5)
```

	Rating					
Item	**1060297744**	**1060697254**	**1610121147**	**3993854748**	**5891061139**	**58**
User						
A0617213KGAVUMXH6NK4	NaN	NaN	NaN	NaN	NaN	
A0755549VZ3OU6OE9EHO	NaN	NaN	NaN	NaN	NaN	
A100C9FK1V6VVT	NaN	NaN	NaN	NaN	NaN	
A103RLAWEHYFHB	NaN	NaN	NaN	NaN	NaN	
A103XTS7PCURDJ	NaN	NaN	NaN	NaN	NaN	

```
new_data1.columns = new_data1.columns.droplevel()
# Adding data for testing purpose
new_data1.loc["A0617213KGAVUMXH6NK4"][1060297744] = 2
new_data1.loc["A0617213KGAVUMXH6NK4"][7508492919] = 1
new_data1.loc["A0755549VZ3OU6OE9EHO"][1610121147] = 4
new_data1.loc["A0755549VZ3OU6OE9EHO"][7508492919] = 1
```

```
new_data1.head()
```

Item	1060297744	1060697254	1610121147	3993854748	5891061139	58
User						
A0617213KGAVUMXH6NK4	2.0	NaN	NaN	NaN	NaN	
A0755549VZ3OU6OE9EHO	NaN	NaN	4.0	NaN	NaN	
A100C9FK1V6VVT	NaN	NaN	NaN	NaN	NaN	
A103RLAWEHYFHB	NaN	NaN	NaN	NaN	NaN	
A103XTS7PCURDJ	NaN	NaN	NaN	NaN	NaN	

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

```
def Normalize (name):
    """
    This function normalizes the user rating.
    """
        normalized_summary = new_data1.copy()
        normalized_summary = normalized_summary.fillna(0)
        normalizen=normalized_summary.copy()
        normalized_summary['count']=normalized_summary.astype(bool).sum(axis=1)
        normalized_summary['sum']=normalizen.sum(axis=1)
        normalized_summary['average']=normalized_summary['sum']/normalized_summary['count']
        normalized=name.sub(normalized_summary['average'], axis=0)
        return normalized , normalized_summary


# normalize function call
normalised_data , normalized_summary = Normalize(new_data1)


# The input dataset after normalizing
normalised_data
```

Item	1060297744	1060697254	1610121147	3993854748	5891061139	58
User						
A0617213KGAVUMXH6NK4	-0.666667	NaN	NaN	NaN	NaN	
A0755549VZ3OU6OE9EHO	NaN	NaN	1.5	NaN	NaN	
A100C9FK1V6VVT	NaN	NaN	NaN	NaN	NaN	
A103RLAWEHYFHB	NaN	NaN	NaN	NaN	NaN	

```
#Input dataset along with Count of rating, Average rating and sum of rating details added.
normalized_summary
```

Item	1060297744	1060697254	1610121147	3993854748	5891061139	58
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	0.0	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	0.0	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

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

Sxy = items rated by both users x and y $sim(x,y)=\frac{\sum_{s\in S_{xy}}(r_{xs}-\bar{r}_x)(r_{ys}-\bar{r}_y)}{\sqrt{\sum_{s\in S_{xy}}(r_{xs}-\bar{r}_x)^2}\sqrt{\sum_{s\in S_{xy}}(r_{ys}-\bar{r}_y)^2}}$

```
def pearson_correlation(x,y):
  """
  This function finds the similarity between the two passed used using pearson correlation fo
  """
  normalizen = new_data1.copy()
```

```
  normalizen = normalizen.fillna(0)

  rxs=normalizen.loc[x]
  rys=normalizen.loc[y]
  Normalize(new_data1)
  rx_mean=normalized_summary.at[x, 'average']
  ry_mean=normalized_summary.at[y, 'average']

  x1= np.array(rxs)
  y1 =np.array(rys)

  num_product = 0
  deno_product = 0
  for i in range(len(x1)):

    if (x1[i] != 0 and y1[i] != 0):
       rx_sub_avg = (x1[i] -rx_mean)
       ry_sub_avg = (y1[i] -ry_mean)

       num_product = num_product + rx_sub_avg * ry_sub_avg
       deno_product = deno_product + (int(math.pow (rx_sub_avg, 2))) * (int(math.pow (ry_sub_

  numerator = num_product
  denominator = np.sqrt(deno_product)

  if (denominator == 0):
    similarity = 0
  else:
    similarity=numerator/denominator
  return similarity


# Function call
similarity = pearson_correlation("A0617213KGAVUMXH6NK4","A0755549VZ3OU6OE9EHO")
similarity
```

```
1.25
```

5. 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

```
def NearestNbrs(u,q,k):
    """
    This function returns the K nearest neighbours and thier similarity to the passed vector
    """
    similar=[]
    users = []
```

```
    for i in u.index:
      x = np.array(u.loc[i])
      y = np.array(q)
      similarity_data= distance(x.astype(float),y.astype(float))
      similar.append(similarity_data)
      users.append(i)

    data = {'user': users,'Similarity': similar}
    df = pd.DataFrame (data, columns = ['user','Similarity'])

    df = df.sort_values(by=['Similarity'],ascending=False)
    return (df.iloc[0:k])


def distance(vector_x,vector_y):
  """
  This function calculates the pearson correlation between two vectors of rating.
  """
  count_x = np.count_nonzero(vector_x)
  count_y = np.count_nonzero(vector_y)

  sum_x = sum(vector_x)
  sum_y = sum(vector_y)

  if count_x == 0 :
    avg_x = 0
  else:
    avg_x = sum_x/count_x

  if count_y == 0 :
    avg_y = 0
  else:
    avg_y = sum_y/count_y

  num1 = vector_x - avg_x
  num2 = vector_y - avg_y

  numerator = sum(num1 * num2)
  denominator = np.sqrt( (sum(num1 * num1)) * (sum(num2 * num2)) )
  if (denominator == 0):
    similarity = 0
  else:
    similarity = numerator / denominator
  return similarity



# Function call
NearestNbrs(df1,q,2)
```

	user	Similarity
4	A207HOQVQ3F552	0.994558
2	A39XKVLWEHYCI1	0.852803

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

rxi=$\Sigma sxy \cdot ryiy \in N \Sigma sxyy \in N$ where sxy=$sim(x,y)$ as given in Q.4

```
def predictRating(x,i):
  """
  This function predicts the rating of a user.
  """
    similar_users = NearestNbrs(df1,x,3)
    numerator=0
    denominator=0
    result=0
    for u,score in similar_users:
        numerator+= score*ru.loc[u][i]
        denominator+=score
    result = numerator/denominator
    return result



# Sample Test Data
data1 = {'user':  ['A3KP1BUNRQY69J', 'A3P7REOQEXHATA','A39XKVLWEHYCI1','A340KNBKZ86MKN','A207
        'item1': ['1', '0','0','0','2','3'],
        'item2': ['0', '0','1','0','5','1'],
        "item3": ['4', '1','0','0','0','0']
        }

df1 = pd.DataFrame (data1, columns = ['user','item1','item2','item3'])
q=(1,2,0)
df1 = df1.set_index('user')
print (df1)
```

```
                 user item1 item2 item3
     0  A3KP1BUNRQY69J     1     0     4
     1  A3P7REOQEXHATA     0     0     1
     2  A39XKVLWEHYCI1     0     1     0
     3  A340KNBKZ86MKN     0     0     0
     4  A207HOQVQ3F552     2     5     0
     5   AT9HSLVQB7OFT     3     1     0
```