Objective:

The objective of the hackathon is to allocate the most relevant set of products to each customer by maximizing total relevancy. You should use the column “relevancy_score” of Relevancy_table to get relevancy of products for customers.

Constraints:

1. Due to budget constraints, there is fixed volume of each product. For instance, product “5650512” cannot be allocated to more than 150 customers. Use the “Volume” column of the Products table.
2. A customer can get maximum 8 products and minimum 3 products. Drop all the customers who qualify for less than 3 products.
3. There are some set of products which cannot be assigned together (e.g. product “5649565” and “5649646” cannot be given together to any customer). You can get this list in the Exclusion table.
4. All the products allocated to a customer should be distinct (i.e. the same product cannot be allocated twice to the same customer)

In [1]:

```
import pandas as pd
exc = pd.read_csv('Exclusion.csv')
prod = pd.read_csv('Products.csv')
rel = pd.read_csv('Relevency_table.csv')
sample = pd.read_csv('Dunnhumby_Sample_Submission_-_Sheet1.csv')
```

In [2]:

```
exc.head(5)
```

Out[2]:

	product1	product2
0	5649565	5649646
1	5649585	5649910
2	5649585	5649921
3	5649607	5649931
4	5649607	5649929

In [3]:

```
prod.head(5)
```

Out[3]:

	product	volume
0	5650512	150
1	5650506	600
2	5649630	300
3	5650672	200
4	5650537	100

In [4]:

```
rel.head(5)
```

Out[4]:

	customers	product	relevancy_score
0	A10001	5649565	0.293978

	customers	product	relevancy_score
1	A10001	5649585	0.076184
2	A10001	5649607	0.312285
3	A10001	5649625	0.113652
4	A10001	5649630	0.108481

In [5]:

```
sample.head(3)
```

Out[5]:

	customers_org	product	relevancy_score
0	A10001	5650743	0.646916
1	A10001	5649965	0.608653
2	A10001	5649679	0.587336

In [6]:

```
#create a dictionary of products which shouldnot be recommended together
exclusive_prod={}
for i in exc.values:
    if i[0] not in exclusive_prod.keys():
            exclusive_prod[i[0]]=[i[1]]
    else:
            exclusive_prod[i[0]].append(i[1])
```

In [7]:

```
#create a dictionary of products with thier stocks in shop/warehouse
prod_vol={}
for i in prod.values:
    if i[0] not in prod_vol.keys():
            prod_vol[i[0]]=i[1]
    else:
            prod_vol[i[0]].append(i[1])
```

In [8]:

```
#create a dictionary with customer as a key and value as dictionary of product and its relevancy s
core

def customer_to_product_dictionary(rel):
    cus_to_prod={}
    for i in rel.values:
        if i[0] not in cus_to_prod.keys():
            cus_to_prod[i[0]]=[{'product':i[1],'rel_value':i[2]}]
        else:
            cus_to_prod[i[0]].append({'product':i[1],'rel_value':i[2]})
    return cus_to_prod


customer_to_product_rel=customer_to_product_dictionary(rel)
```

In [9]:

```
#for each customer sort the products based on its relevancy score in descending order such that hi
ghest relvancy-
#-scored product is first in list
def mysort_cus(e):
    return e['rel_value']

for i in customer_to_product_rel:
    customer_to_product_rel[i].sort(key=mysort_cus,reverse=True)
```

```
    customer_to_product_rel[i]=customer_to_product_rel[i][:24]
```

In [10]:

```
{k: customer_to_product_rel[k] for k in sorted(customer_to_product_rel.keys())[:2]}
```

Out[10]:


```
{'A10001': [{'product': 5650743, 'rel_value': 0.646915508},
  {'product': 5649965, 'rel_value': 0.608652774},
  {'product': 5649679, 'rel_value': 0.587335639},
  {'product': 5650455, 'rel_value': 0.581182403},
  {'product': 5650462, 'rel_value': 0.575268774},
  {'product': 5650048, 'rel_value': 0.558810395},
  {'product': 5650475, 'rel_value': 0.548357995},
  {'product': 5650704, 'rel_value': 0.5461014},
  {'product': 5650640, 'rel_value': 0.545054764},
  {'product': 5650785, 'rel_value': 0.535232519},
  {'product': 5650772, 'rel_value': 0.5288961},
  {'product': 5650777, 'rel_value': 0.5283564860000001},
  {'product': 5650677, 'rel_value': 0.52790005},
  {'product': 5650562, 'rel_value': 0.526961185},
  {'product': 5650760, 'rel_value': 0.525759346},
  {'product': 5650706, 'rel_value': 0.522248344},
  {'product': 5650578, 'rel_value': 0.49041221700000004},
  {'product': 5649701, 'rel_value': 0.47821906399999997},
  {'product': 5650489, 'rel_value': 0.462134},
  {'product': 5650570, 'rel_value': 0.448038029},
  {'product': 5650644, 'rel_value': 0.44380107399999996},
  {'product': 5650529, 'rel_value': 0.440344045},
  {'product': 5650486, 'rel_value': 0.419458461},
  {'product': 5649878, 'rel_value': 0.412093135}],
 'A10002': [{'product': 5650568, 'rel_value': 0.8893212309999999},
  {'product': 5650486, 'rel_value': 0.6429285},
  {'product': 5650743, 'rel_value': 0.637826398},
  {'product': 5650580, 'rel_value': 0.6210124029999999},
  {'product': 5650692, 'rel_value': 0.618344644},
  {'product': 5650475, 'rel_value': 0.612543984},
  {'product': 5650644, 'rel_value': 0.6050616870000001},
  {'product': 5650640, 'rel_value': 0.5883023079999999},
  {'product': 5649648, 'rel_value': 0.580287113},
  {'product': 5650785, 'rel_value': 0.57873772},
  {'product': 5649963, 'rel_value': 0.576988498},
  {'product': 5650772, 'rel_value': 0.572549261},
  {'product': 5650760, 'rel_value': 0.56948044},
  {'product': 5650570, 'rel_value': 0.567664862},
  {'product': 5650656, 'rel_value': 0.545341512},
  {'product': 5649836, 'rel_value': 0.529211061},
  {'product': 5649630, 'rel_value': 0.517174396},
  {'product': 5649845, 'rel_value': 0.5083355829999999},
  {'product': 5650535, 'rel_value': 0.5018051179999999},
  {'product': 5649859, 'rel_value': 0.48299767899999996},
  {'product': 5650496, 'rel_value': 0.47272831299999996},
  {'product': 5649849, 'rel_value': 0.472301435},
  {'product': 5650706, 'rel_value': 0.459755938},
  {'product': 5649607, 'rel_value': 0.449054261}]}
```


In [11]:

```
#create a dictionary with product as a key and value as dictionary of customer and its residual or
error value:

# NOTE: prod_to_cus only store information of customers of top <=8 products of all the customers s
ince we need to choose max8 elements and will be dynamically updated based on updation and changes
happens according to situation

#error value: 1)i) if data of cutomer 'a' has more than 8 products with relvancy score than we cho
ose only top 8 products because of the constraint
#               ii) since we need to maxmize the relevancy score we need to choose products with a
higher relevancy score as much as possible while choosing 8 products.
#               iii) hence i check in dictionary of cutomer 'a' if i remove high relevancy score r
of product p1 due to low volume of products as given in data, it will be substitued by lower relev
ancy product p9 of relevancy socre r9
```

```
#                    so how much relevancy score i will be loosing i.e r1-r8

#           Ex if need to need to assign p1 whose volume is just 1 piece hence i need to choose bet
ween two customer

#  before reomving p1  customer[a]=[{p1,0.69},{p5,0.65},{p6,0.60},{p2,0.55},{p5,0.45},{p8,0.35},{p
7,0.30},{p23,0.25},{p9,0.20},...]
#                           error is 0.69(p1)-0.20(p9)=0.49 here if i remove p1 then will be
replaced by p9 in top 8 products with the loss of 0.49
#  After removing p1   customer[a]=[{p5,0.65},{p6,0.60},{p2,0.55},{p5,0.45},{p8,0.35},{p7,0.30},{p
23,0.25},{p9,0.20},{p20...
#                       customer[b]=[{p5,0.79},{p1,0.75},{p3,0.74},{p45,0.72},{p5,0.70},{p85,0.69},
p73,0.67},{p43,0.65},{p46,0.60},...]
#

#     before          customer[b]=[{p5,0.79},{p1,0.75},{p3,0.74},{p45,0.72},{p5,0.70},{p85,0.69},{p
73,0.67},{p43,0.65},{p46,0.60},...]
#                           error 0.75(p1)-0.60(p46)=0.15 here if i remove p1 then will be replace
by p46 in top 8 products with the loss of 0.15
#                       customer[b]=[{p5,0.79},{p3,0.74},{p45,0.72},{p5,0.70},{p85,0.69},{p73,0.67},
p43,0.65},{p46,0.60},{p55,0.55}...]
#                       customer[a]=[{p1,0.69},{p5,0.65},{p6,0.60},{p2,0.55},{p5,0.45},{p8,0.35},{p7,
0.30},{p23,0.25},{p9,0.20},...]

#               since i am losing more relvacy score i.e 0.49 if i decide to assign p1 to cutomer
b' so to reduce the loss and maximize the relvancy score i will asssign to customer 'a'
#
#            2) if data of cutomer has <=8 products then it will have no products to substitute
hence error will be r1-0
#
#            Ex   customer[a]=[{p1,0.69},{p5,0.65},{p6,0.60},{p2,0.55},{p5,0.45},{p8,0.35},{p7,0.3
}]
#                             error 0.69(p1)-0=0.69
#                  customer[a]=[{p5,0.65},{p6,0.60},{p2,0.55},{p5,0.45},{p8,0.35},{p7,0.30}]
#
#                  customer[b]=[{p5,0.79},{p1,0.75},{p3,0.74},{p45,0.72},{p5,0.70}]
#                             error 0.75(p1)-0=0.75
#              since i am losing more relvacy score i.e 0.75 if i decide to assign p1 to cutomer 'a
' so to reduce the loss and maximize the relvancy score i will asssign to customer 'b'


prod_to_cus={}
for i in customer_to_product_rel:
    if len(customer_to_product_rel[i])<9:
        #if customer data has less than 8 products
        for j in range(0,len(customer_to_product_rel[i])):
            if customer_to_product_rel[i][j]['product'] not in prod_to_cus.keys():
                prod_to_cus[customer_to_product_rel[i][j]
['product']]=[{'customer':i,'error':customer_to_product_rel[i][j]['rel_value']}]
            else:
                prod_to_cus[customer_to_product_rel[i][j]['product']].append({'customer':i,'error':
customer_to_product_rel[i][j]['rel_value']})
    else:
        for j in range(0,8):
            #if customer data has more than or equals to 8 products
            if customer_to_product_rel[i][j]['product'] not in prod_to_cus.keys():
                prod_to_cus[customer_to_product_rel[i][j]
['product']]=[{'customer':i,'error':customer_to_product_rel[i][j]
['rel_value']-customer_to_product_rel[i][8]['rel_value']}]
            else:
                prod_to_cus[customer_to_product_rel[i][j]['product']].append({'customer':i,'error':
customer_to_product_rel[i][j]['rel_value']-customer_to_product_rel[i][8]['rel_value']})
```

In [12]:

```
#for each product sort the customers based on its error in descending order such that highest erro
r-
# customer w.r.t the product is first in list
def mysort(e):
```

```
    return e['error']

for i in prod_to_cus:
    prod_to_cus[i].sort(key=mysort,reverse=True)
```

In [13]:

```
{k: prod_to_cus[k] for k in sorted(prod_to_cus.keys())[:2]}
```

Out[13]:

```
{5649565: [{'customer': 'A10562', 'error': 0.06495776500000001},
  {'customer': 'A10662', 'error': 0.05817251600000001},
  {'customer': 'A10367', 'error': 0.025487542999999988},
  {'customer': 'A10661', 'error': 0.016762085},
  {'customer': 'A10722', 'error': 0.015338438999999995},
  {'customer': 'A10778', 'error': 0.014319370000000001},
  {'customer': 'A10707', 'error': 0.005257355999999998}],
 5649585: [{'customer': 'A10382', 'error': 0.021670202},
  {'customer': 'A10661', 'error': 0.016348998},
  {'customer': 'A10778', 'error': 3.8070999999999765e-05}]}
```

In [14]:

```
#Algorithm:

# 1) for each customer take top 8 prodcuts to check for mutually exclusive product constraint in p
roducts_to_check

# 2) for each product for a customer get mutually exclusive products from excluive table(i convert
ed to dict)

# 3) get the error of product 'p1' for which we are finding if there are any mutually exlusive pro
ducts in top 8

# 4) get mutually exclusive products of 'p1'
#    ex: customer[a]=[{p1,0.69},{p5,0.67},{p6,0.66},{p2,0.65},{p5,0.64},{p8,0.60},{p7,0.30},{p23,0
.25},{p9,0.20},...]
#     p1 =[{customer a,error : 0.44(0.69-0.25)},{customer c,error : 0.40},{customer b,error : 0.35
}...]
#     p5=[{customer c,error : 0.75},{customer a,0.39(0.64-0.25)},...]
#     p8=[{customer a,error : 0.35(0.65-0.25)},{customer g,0.04},...]

# 5) get mutually exc[p1]=[p5,p8,...]
#     and find the products in cutomer[a] which are in mutually exclusive products using
get_intersection_ele, in above example for p1, p5 and p8 cannot occur together with p1.

# 6)  to choose between p1 and p5/p8 we check error of p1 and total error of p5 and p8
#     ex: error of p1 of cutomer a=0.44 and and error of p5 aqnd p6 of customer a 0.35+0.39=0.74
#     so if i remove p5 and p6 just because they have relevance score lower than p1 i will lose re
levancy score of 0.74 which is way more than 0.44
#     hence remove the p1 and update the error since now at the 8th position a new element has com
e

# 7) this is done by checking is prod1_error > total_error if prod1_error is greater than remove a
ll mutually exlusive elements which are top 8

# 8) also remove mututally exclusive elements which are not in top8 since going ahead some element
s will replace current elements which are in top8 now and since those replaced elements may be
#    mutually exclusive product of p1 which we need to avoid, it is done by
remove_from_customer_to_product_after_lenght_8()

# 9) remove the product p5 and p8 from products_to_check as well if prod1_error>tota_error and als
o update new elements to products_to_check  which got into top8 recently after removing p5 and p8.

# 10) remove the customer and product entry of product p5 and p8 from prod_to_cus and customer_to_
prod
# 11) update the error since there are new elements are added after removal

# 12) if total_error > prod1_error the remove p1 from customer_to_prod and the customer a in prod_
to_cus of p1

# 13)update the error since there are new elements are added after removal.
```

```
def get_products_to_check(customer_to_product_rel,k,start,end):
    temp=[]
    for i in customer_to_product_rel[k][start:end]:
        temp.append(i['product'])
    return temp

def get_intersection_ele(lst1, lst2):
    #get common elements
    return list(set(lst1) & set(lst2))

def cal_error(prod_to_cus,k,mut_exc_ele):
    total_error=0
    for t in mut_exc_ele:
            for j in prod_to_cus[t]:
                if j["customer"]==k:
                    total_error=total_error+j['error']
    return total_error

def remove_from_customer_to_product(t,k,customer_to_product_rel):
    for i in customer_to_product_rel[k]:
        if i['product']==t:
            customer_to_product_rel[k].remove({'product':t,'rel_value':i['rel_value']})


def remove_from_product_to_customer(t,k,prod_to_cus):

     for i in prod_to_cus[t]:
        if i['customer']==k:
            prod_to_cus[t].remove({'customer':k,'error':i['error']})

def remove_from_customer_to_product_after_lenght_8(i,k,customer_to_product_rel):
    if len(customer_to_product_rel[k])>8:
        temp=get_products_to_check(customer_to_product_rel,k,8,len(customer_to_product_rel[k]))
        mut_exc_ele=get_intersection_ele(exclusive_prod[i],temp)

        for t in mut_exc_ele:
            remove_from_customer_to_product(t,k,customer_to_product_rel)



def update_error_in_product_to_customer(k,customer_to_product_rel,prod_to_cus):
    #check for length >8 or >=9
    if len(customer_to_product_rel[k])>=9:

        for j in range(0,8):

            if customer_to_product_rel[k][j]['product'] not in prod_to_cus.keys():

                #if the product which came in top 8 of customer k but was not in top8 of any
customer before,so need to add to prod_to_cus since it just hold top 8 info of cutomers of that pr
oduct
                prod_to_cus[customer_to_product_rel[k][j]
['product']]=[{'customer':k,'error':customer_to_product_rel[k][j]
['rel_value']-customer_to_product_rel[k][8]['rel_value']}]

            else:

                flag=0

                for i in range(0,len(prod_to_cus[customer_to_product_rel[k][j]['product']])):

                    #if the product which is already in top 8 of customer k and also in top 8 of c
thers m customers hence you need to just update the value
                    if k == prod_to_cus[customer_to_product_rel[k][j]['product']][i]['customer']:
                        prod_to_cus[customer_to_product_rel[k][j]['product']][i]["error"]= customer
to_product_rel[k][j]['rel_value']-customer_to_product_rel[k][8]['rel_value']
                        flag=1

                if flag==0:
                     #if the product which is already in top 8 of m customer but not in customer k
then need to add to the list
                    prod_to_cus[customer_to_product_rel[k][j]
['product']].append({'customer':k,'error':customer_to_product_rel[k][j]['rel_value']-customer_to_pr
oduct_rel[k][8]['rel_value']})
```

```
            prod_to_cus[customer_to_product_rel[k][j]['product']].sort(key=mysort,reverse=True)
    else:

        #if length <8
        for j in range(0,len(customer_to_product_rel[k])):
            if customer_to_product_rel[k][j]['product'] not in prod_to_cus.keys():

                #if the product which came in top <8 of customer k but was not in top<8 of any cust
omer before,so need to add to prod_to_cus since it just hold top <8 info of cutomers of that produ
ct
                prod_to_cus[customer_to_product_rel[k][j]
['product']]=[{'customer':k,'error':customer_to_product_rel[k][j]['rel_value']}]

            else:
                flag=0

                for i in range(0,len(prod_to_cus[customer_to_product_rel[k][j]['product']])):

                    #if the product which is already in top<8 of customer k and also in top 8 of ot
hers m customers hence you need to just update the value
                    if k == prod_to_cus[customer_to_product_rel[k][j]['product']][i]['customer']:
                        prod_to_cus[customer_to_product_rel[k][j]['product']][i]["error"]= customer
to_product_rel[k][j]['rel_value']
                        flag=1

                if flag==0:
                    #if the product which is already in top<8 of m customer but not in customer k t
hen need to add to the list
                    prod_to_cus[customer_to_product_rel[k][j]
['product']].append({'customer':k,'error':customer_to_product_rel[k][j]['rel_value']})
            prod_to_cus[customer_to_product_rel[k][j]['product']].sort(key=mysort,reverse=True)

#for each products
for k in customer_to_product_rel:
    products_to_check=[]
    length=min(len(customer_to_product_rel[k]),8)


    products_to_check=get_products_to_check(customer_to_product_rel,k,0,length)

    for i in products_to_check:
        mut_exc_ele=[]
        if i in exclusive_prod.keys():
            temp=get_products_to_check(customer_to_product_rel,k,0,length)

            mut_exc_ele=get_intersection_ele(exclusive_prod[i],temp)

            prod1_error=0

            #get product_error
            if len(mut_exc_ele)!=0:
                for j in prod_to_cus[i]:
                    if j['customer']==k:
                        prod1_error=j['error']

                total_error=cal_error(prod_to_cus,k,mut_exc_ele)

                if prod1_error>=total_error:
                    for t in mut_exc_ele:
                        remove_from_customer_to_product(t,k,customer_to_product_rel)
                        products_to_check.remove(t)
                        remove_from_product_to_customer(t,k,prod_to_cus)
                        if len(customer_to_product_rel[k])>=8:
                            products_to_check.append(customer_to_product_rel[k][7]['product'])
                    remove_from_customer_to_product_after_lenght_8(i,k,customer_to_product_rel)


                else:
                    remove_from_customer_to_product(i,k,customer_to_product_rel)
                    remove_from_product_to_customer(i,k,prod_to_cus)
                    if len(customer_to_product_rel[k])>=8:
                            products_to_check.append(customer_to_product_rel[k][7]['product'])

                update_error_in_product_to_customer(k,customer_to_product_rel,prod_to_cus)
            else:
                remove_from_customer_to_product_after_lenght_8(i,k,customer_to_product_rel)
```

In [15]:

```
#these are products whose stack is less
for i in prod_to_cus:
    if len(prod_to_cus[i])>prod_vol[i]:
        print(i,len(prod_to_cus[i]),prod_vol[i])
```

```
5650568 340 20
5650542 224 200
5650570 155 100
5649648 251 200
5650562 228 200
5650659 152 80
```

In [16]:

```
#these are products whose stock is less so store it in list for furthur process
products_shortage=[]
for i in prod_to_cus:
        if len(prod_to_cus[i])>prod_vol[i]:
            products_shortage.append(i)
```

In [17]:

```
#note: once i assign all products  to customer under capacity then they are in top8 since prod_to_
cus only hold top8 elements
#EX: p1 =[{customer a,error : 0.44},{customer c,error : 0.40},{customer b,error : 0.35},{customer
d,error : 0.25},...] and p1 has capacity 2
#    hence i need to remove customer from prod_to_cus after length 2
#    after removing p1 =[{customer a,error : 0.44},{customer c,error : 0.40}]
#    and from customer_to_product i need to remove p1 from customer b and customer d
#
count_of_loop=0

#run the loop until all products are assigned to cutomer under capacity
while(len(products_shortage)!=0):
    count_of_loop=count_of_loop+1

    for i in products_shortage:
        customer_to_remove_from_prod_to_cus=[]

        for j in prod_to_cus[i][prod_vol[i]:]:

            customer_to_remove_from_prod_to_cus.append(j['customer'])

            remove_from_customer_to_product(i,j['customer'],customer_to_product_rel)


        for k in customer_to_remove_from_prod_to_cus:
            update_error_in_product_to_customer(k,customer_to_product_rel,prod_to_cus)
            remove_from_product_to_customer(i,k,prod_to_cus)

    products_shortage=[]
    for i in prod_to_cus:
        if len(prod_to_cus[i])>prod_vol[i]:
            products_shortage.append(i)


print(count_of_loop)
```

```
3
```

In [18]:

```
#total relevancy score
sum1=0
for i in customer_to_product_rel:
    length=min(len(customer_to_product_rel[i]),8)
```

```
    for j in customer_to_product_rel[i][:length]:
        sum1=sum1+j['rel_value']
print("total relevancy score ",sum1)
```

```
total relevancy score  1107.396979670006
```

In [19]:

```
result=[]
for i in customer_to_product_rel:
    for j in customer_to_product_rel[i][:8]:
        result.append([i,j['product'],j['rel_value']])
```

In [20]:

```
result[:5]
```

Out[20]:

```
[['A10001', 5650743, 0.646915508],
 ['A10001', 5649965, 0.608652774],
 ['A10001', 5649679, 0.587335639],
 ['A10001', 5650455, 0.581182403],
 ['A10001', 5650640, 0.545054764]]
```

In [22]:

```
#create dataframe
df = pd.DataFrame(result, columns = ['customers_org', 'product','relevancy_score'])
```

In [23]:

```
df.head(5)
```

Out[23]:

	customers_org	product	relevancy_score
0	A10001	5650743	0.646916
1	A10001	5649965	0.608653
2	A10001	5649679	0.587336
3	A10001	5650455	0.581182
4	A10001	5650640	0.545055

In [249]:

```
#store in csv
df.to_csv("dunnhumby_hackathon_result.csv", encoding='utf-8', index=False)
```

In [ ]: