nilimshop-rfm-analysis (1)

March 28, 2020

[1]:
```
# This Python 3 environment comes with many helpful analytics libraries␣
↪installed
# It is defined by the kaggle/python docker image: https://github.com/kaggle/
↪docker-python
# For example, here's several helpful packages to load in

import numpy as np # linear algebra
import pandas as pd # data processing, CSV file I/O (e.g. pd.read_csv)

# Input data files are available in the "../input/" directory.
# For example, running this (by clicking run or pressing Shift+Enter) will list␣
↪all files under the input directory

import os
for dirname, _, filenames in os.walk('/kaggle/input'):
    for filename in filenames:
        print(os.path.join(dirname, filename))

# Any results you write to the current directory are saved as output.
```

```
/kaggle/input/sample-sales-data/sales_data_sample.csv
/kaggle/input/nilimshopselldata/nilim-sell-data.csv
```

0.1 ### RFM Analysis

RFM (Recency, Frequency, Monetary) analysis is a marketing technique used to determine quantitatively which customers are the best ones by examining *how recently a customer has purchased* (recency), *how often they purchase* (frequency), and *how much the customer spends* (monetary). Using RFM analysis, customers are assigned a ranking number of 1,2,3,4(with 4 being highest) for each RFM parameter. The three scores together are referred to as an RFM "cell". The data is sorted to determine which customers were the best customers in the past, with a cell ranking of 444 being ideal.

import the dataset first

[2]:
```
df = pd.read_csv('/kaggle/input/nilimshopselldata/nilim-sell-data.csv')
print(df.describe())
display(df.head(5))
```

```
              SALES
count     21.000000
mean     351.809524
std      238.204034
min      140.000000
25%      209.000000
50%      290.000000
75%      400.000000
max     1200.000000
          CUSTOMERNAME   ORDERDATE ORDERID  SALES       AREA  GENDER  \
0       Hussein Mabruk   11/8/2019  3546-1  290.0     Mirpur    MALE
1  Tabassum Kabir Hena  11/13/2019  3546-2  290.0     Mirpur  FEMALE
2           Noor Islam  12/29/2019  3546-3  510.0     Mirpur    MALE
3                Julie    2/8/2020  3546-4  260.0  Moghbazar  FEMALE
4          Nusrat Hakim   2/8/2020  3546-5  189.0     Uttara  FEMALE

  Unnamed: 6 Unnamed: 7
0        NaN        NaN
1        NaN        NaN
2        NaN        NaN
3        NaN        NaN
4        NaN        NaN
```

Let's filter the columns we need. we will only need four columns. **CUSTOMERNAME** to group customers, **ORDERDATE** to calculate recency, **ORDERNUMBER** to calculate frequency, and **SALES** to calculate monetary.

[3]:
```
cols = ['CUSTOMERNAME','ORDERDATE','ORDERID','SALES']
df = df[cols]
print(df.head(5))
```

```
          CUSTOMERNAME   ORDERDATE ORDERID  SALES
0       Hussein Mabruk   11/8/2019  3546-1  290.0
1  Tabassum Kabir Hena  11/13/2019  3546-2  290.0
2           Noor Islam  12/29/2019  3546-3  510.0
3                Julie    2/8/2020  3546-4  260.0
4          Nusrat Hakim   2/8/2020  3546-5  189.0
```

0.2 ### Recency:

To find recency we have to find the date of the last order of each customer made and subtract the value from the most recent order date in the dataset. The difference give us the recency value, for recent customer the value will be smaller.

[4]:
```
df['ORDERDATE'] = pd.to_datetime(df['ORDERDATE'])

#group the data by CUSTOMERNAME and only retrive the ORDERDATE column
```

```
recent_order = df.groupby('CUSTOMERNAME')['ORDERDATE'].max()

most_recent = df['ORDERDATE'].max()

def subtract_date(date):
    days = (most_recent - date).days
    return days

recency = recent_order.apply(subtract_date)

print('Recency days : ', recency)
#print(recent_order.head(10))
#print(df.groupby(['CUSTOMERNAME','ORDERDATE']).count().head())
```

```
Recency days :  CUSTOMERNAME
Akhi                    28
Alefa                   31
Dilshad Sharmin          7
Fahima Nupur             0
Farhana Yasmin          22
Farzana Akter           33
Hussein Mabruk          10
Julie                   35
Nishat Afroz             0
Noor Islam              76
Nusrat Hakim            35
Sawon                    8
Shammi                  59
Shompa                  13
Sium Mahmud             11
Sorup                   11
Sumaiya Islam Mim        5
Tabassum Kabir Hena    122
Tanjin Ahsan            23
Utsob Roy                4
Name: ORDERDATE, dtype: int64
```

0.3 ### Frequency

Frequency is the measure how often a customer purchase a product. Now lets find out the number of times each customer has placed an order.

[5]:
```
frequency = df.groupby(['CUSTOMERNAME','ORDERID']).size()
frequency = frequency.groupby('CUSTOMERNAME').size()
print(frequency.head())
```

```
CUSTOMERNAME
Akhi                 1
```

```
Alefa               1
Dilshad Sharmin     1
Fahima Nupur        1
Farhana Yasmin      1
dtype: int64
```

size() and count() function both same but count() ignores missing/NaN/NULL values.

0.4 ### Monetary

Monetary is how each customer spent.

[6]:
```
#groupby the CUSTOMERNAME and only retrive the SALES and sum
monetary = df.groupby('CUSTOMERNAME')['SALES'].sum()
print(monetary.head())
```

```
CUSTOMERNAME
Akhi                 318.0
Alefa               1200.0
Dilshad Sharmin      159.0
Fahima Nupur         658.0
Farhana Yasmin       298.0
Name: SALES, dtype: float64
```

0.5 ### Putting All Togeather

Now we are going to put all these 3 parameter togeather.

[7]:
```
rfm = pd.DataFrame()
rfm['recency'] = recency
rfm['frequency'] = frequency
rfm['monetary'] = monetary

print(rfm.head())
```

```
                 recency  frequency  monetary
CUSTOMERNAME
Akhi                  28          1     318.0
Alefa                 31          1    1200.0
Dilshad Sharmin        7          1     159.0
Fahima Nupur           0          1     658.0
Farhana Yasmin        22          1     298.0
```

Now we will convert these raw value into class, based on which quantile it fall into.

[8]:
```
quantile_df = rfm.quantile([0.25,0.50,0.75])
display(quantile_df)
```

```
      recency  frequency  monetary
0.25     7.75        1.0    209.00
```

```
0.50     17.50         1.0    303.00
0.75     33.50         1.0    449.75
```

[9]:
```
def quantile_classes(x, quantile_value, attribute):
    if attribute == 'recency':
        if x <= quantile_value.loc[0.25,attribute]: # receny is less than 0.25%
            return '4'
        elif x >= quantile_value.loc[0.25,attribute] and x <= quantile_value.
 ↪loc[0.50,attribute]: # recency is larger than 25%
            return '3'
        elif x >= quantile_value.loc[0.50,attribute] and x <= quantile_value.
 ↪loc[0.75,attribute]:
            return '2'
        else:
            return '1'
    else:
        #frequncy and monetary
        if x <= quantile_value.loc[0.25,attribute]: # frequncy/monetary is less␣
 ↪than 0.25%
            return '1'
        elif x >= quantile_value.loc[0.25,attribute] and x <= quantile_value.
 ↪loc[0.50,attribute]: # frequncy/monetary is larger than 25%
            return '2'
        elif x >= quantile_value.loc[0.50,attribute] and x <= quantile_value.
 ↪loc[0.75,attribute]:
            return '3'
        else:
            return '4'


#convert rfm table raw value to class
rfm['recency_class'] = rfm['recency'].apply(quantile_classes, args =␣
 ↪(quantile_df,'recency'))
rfm['frequency_class'] = rfm['frequency'].apply(quantile_classes, args =␣
 ↪(quantile_df,'frequency'))
rfm['monetary_class'] = rfm['monetary'].apply(quantile_classes, args =␣
 ↪(quantile_df,'monetary'))

display(rfm.head())
```

```
                  recency  frequency  monetary recency_class frequency_class  \
CUSTOMERNAME
Akhi                   28          1     318.0             2               1
Alefa                  31          1    1200.0             2               1
Dilshad Sharmin         7          1     159.0             4               1
Fahima Nupur            0          1     658.0             4               1
```

```
Farhana Yasmin        22           1     298.0              2                1

                monetary_class
CUSTOMERNAME
Akhi                         3
Alefa                        4
Dilshad Sharmin              1
Fahima Nupur                 4
Farhana Yasmin               2
```

combine all of these individual class into a single column.

[10]:
```python
#join the string values
rfm['rfm_comb'] = rfm['recency_class'] + rfm['frequency_class'] +␣
 ↪rfm['monetary_class']

#convert to numeric value
rfm['rfm_comb'] = pd.to_numeric(rfm['rfm_comb'])

#sort values
rfm = rfm.sort_values(by=['rfm_comb'], ascending=False)

#display top 10 customer
display(rfm.head(10))
```

```
                   recency  frequency  monetary recency_class frequency_class  \
CUSTOMERNAME
Utsob Roy                4          1     509.0             4               1
Fahima Nupur             0          1     658.0             4               1
Sumaiya Islam Mim        5          1     308.0             4               1
Dilshad Sharmin          7          1     159.0             4               1
Nishat Afroz             0          1     209.0             4               1
Hussein Mabruk          10          2     430.0             3               4
Sorup                   11          1     258.0             3               1
Shompa                  13          1     189.0             3               1
Sawon                    8          1     209.0             3               1
Sium Mahmud             11          1     159.0             3               1

                  monetary_class  rfm_comb
CUSTOMERNAME
Utsob Roy                      4       414
Fahima Nupur                   4       414
Sumaiya Islam Mim              3       413
Dilshad Sharmin                1       411
Nishat Afroz                   1       411
Hussein Mabruk                 3       343
Sorup                          2       312
```

```
Shompa                         1       311
Sawon                          1       311
Sium Mahmud                    1       311
```