intro-to-data-science-rfm-analysis

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
```

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/sample-sales-data/sales_data_sample.csv',␣
↪encoding='unicode_escape')
display(df.head(5))
```

```
   ORDERNUMBER  QUANTITYORDERED  PRICEEACH  ORDERLINENUMBER    SALES  \
0        10107                30      95.70                  2  2871.00
1        10121                34      81.35                  5  2765.90
2        10134                41      94.74                  2  3884.34
3        10145                45      83.26                  6  3746.70
4        10159                49     100.00                 14  5205.27

        ORDERDATE   STATUS  QTR_ID  MONTH_ID  YEAR_ID  ...  \
0   2/24/2003 0:00  Shipped       1         2     2003  ...
1    5/7/2003 0:00  Shipped       2         5     2003  ...
2    7/1/2003 0:00  Shipped       3         7     2003  ...
3   8/25/2003 0:00  Shipped       3         8     2003  ...
4  10/10/2003 0:00  Shipped       4        10     2003  ...

                    ADDRESSLINE1  ADDRESSLINE2           CITY STATE  \
0        897 Long Airport Avenue           NaN            NYC    NY
1             59 rue de l'Abbaye           NaN          Reims   NaN
2  27 rue du Colonel Pierre Avia           NaN          Paris   NaN
3             78934 Hillside Dr.           NaN       Pasadena    CA
4                7734 Strong St.           NaN  San Francisco    CA

  POSTALCODE COUNTRY TERRITORY CONTACTLASTNAME CONTACTFIRSTNAME DEALSIZE
0      10022     USA       NaN              Yu             Kwai    Small
1      51100  France      EMEA         Henriot             Paul    Small
2      75508  France      EMEA        Da Cunha           Daniel   Medium
3      90003     USA       NaN           Young            Julie   Medium
4        NaN     USA       NaN           Brown            Julie   Medium

[5 rows x 25 columns]
```

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','ORDERNUMBER','SALES']
df = df[cols]
print(df.head(5))
```

```
              CUSTOMERNAME        ORDERDATE  ORDERNUMBER    SALES
0          Land of Toys Inc.   2/24/2003 0:00        10107  2871.00
1         Reims Collectables    5/7/2003 0:00        10121  2765.90
2            Lyon Souveniers    7/1/2003 0:00        10134  3884.34
3          Toys4GrownUps.com   8/25/2003 0:00        10145  3746.70
4  Corporate Gift Ideas Co.  10/10/2003 0:00        10159  5205.27
```

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]:
```python
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
AV Stores, Co.                 195
Alpha Cognac                    64
Amica Models & Co.             264
Anna's Decorations, Ltd         83
Atelier graphique              187
                              …
Vida Sport, Ltd                274
Vitachrome Inc.                207
Volvo Model Replicas, Co       193
West Coast Collectables Co.    488
giftsbymail.co.uk              211
Name: ORDERDATE, Length: 92, 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]:
```python
frequency = df.groupby(['CUSTOMERNAME','ORDERNUMBER']).size()
frequency = frequency.groupby('CUSTOMERNAME').size()
print(frequency.head())
```

```
CUSTOMERNAME
AV Stores, Co.             3
Alpha Cognac               3
Amica Models & Co.         2
Anna's Decorations, Ltd    4
Atelier graphique          3
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
AV Stores, Co.              157807.81
Alpha Cognac                 70488.44
Amica Models & Co.           94117.26
Anna's Decorations, Ltd     153996.13
Atelier graphique            24179.96
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
AV Stores, Co.               195          3  157807.81
Alpha Cognac                  64          3   70488.44
Amica Models & Co.           264          2   94117.26
Anna's Decorations, Ltd       83          4  153996.13
Atelier graphique            187          3   24179.96
```

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    80.25        2.0   70129.4325
0.50   185.00        3.0   86522.6100
0.75   229.25        3.0  120575.8750
```

[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  \
CUSTOMERNAME
AV Stores, Co.               195          3  157807.81             2
Alpha Cognac                  64          3   70488.44             4
Amica Models & Co.           264          2   94117.26             1
Anna's Decorations, Ltd       83          4  153996.13             3
Atelier graphique            187          3   24179.96             2

                        frequency_class monetary_class
CUSTOMERNAME
AV Stores, Co.                        2              4
Alpha Cognac                          2              2
```

```
Amica Models & Co.                       1              3
Anna's Decorations, Ltd                  4              4
Atelier graphique                        2              1
```

combine all of these individual class into a single column.

```
[10]: #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  \
CUSTOMERNAME
Salzburg Collectables              14          4  149798.63             4
Souveniers And Things Co.           2          4  151570.98             4
Mini Gifts Distributors Ltd.        2         17  654858.06             4
Danish Wholesale Imports           46          5  145041.60             4
Diecast Classics Inc.               1          4  122138.14             4
La Rochelle Gifts                   0          4  180124.90             4
The Sharp Gifts Warehouse          39          4  160010.27             4
Reims Collectables                 62          5  135042.94             4
Euro Shopping Channel               0         26  912294.11             4
Tokyo Collectables, Ltd            39          4  120562.74             4

                             frequency_class monetary_class  rfm_comb
CUSTOMERNAME
Salzburg Collectables                      4              4       444
Souveniers And Things Co.                  4              4       444
Mini Gifts Distributors Ltd.               4              4       444
Danish Wholesale Imports                   4              4       444
Diecast Classics Inc.                      4              4       444
La Rochelle Gifts                          4              4       444
The Sharp Gifts Warehouse                  4              4       444
Reims Collectables                         4              4       444
Euro Shopping Channel                      4              4       444
Tokyo Collectables, Ltd                    4              3       443
```