rfm_segmentation

September 23, 2019

1 *Introduction to RFM segmentation*

[46]:
```
import pandas as pd
from datetime import timedelta
import os
import openpyxl
import numpy as np

print(os.getcwd())
```

```
/home/hans/python_codes/DataCamp/frm_segmentation
```

1.1 Recency, Frequency, Monetary Value calculation

1.1.1 Definition:

- Recency - days since last custmer transaction
- Frequency - number of transactions in the last 12 months
- Monetary Value - total spend in the last 12 months

1.2 Dataset an preparation

[47]:
```
#online = pd.read_excel('../data/Online Retail.xlsx')
```

- Need to do some data preparation

[48]:
```
online['TotalSum'] = online['Quantity'] * online['UnitPrice']
```

[49]:
```
online.head()
```

[49]:

	InvoiceNo	StockCode	Description	Quantity	\
0	536365	85123A	WHITE HANGING HEART T-LIGHT HOLDER	6	
1	536365	71053	WHITE METAL LANTERN	6	
2	536365	84406B	CREAM CUPID HEARTS COAT HANGER	8	
3	536365	84029G	KNITTED UNION FLAG HOT WATER BOTTLE	6	
4	536365	84029E	RED WOOLLY HOTTIE WHITE HEART.	6	

	InvoiceDate	UnitPrice	CustomerID	Country	TotalSum
0	2010-12-01 08:26:00	2.55	17850.0	United Kingdom	15.30
1	2010-12-01 08:26:00	3.39	17850.0	United Kingdom	20.34
2	2010-12-01 08:26:00	2.75	17850.0	United Kingdom	22.00
3	2010-12-01 08:26:00	3.39	17850.0	United Kingdom	20.34
4	2010-12-01 08:26:00	3.39	17850.0	United Kingdom	20.34

1.2.1 Data preparation steps

We're starting with pre-processed DataFrame with only the latest 12 months of data:

[50]:
```
print('Min:{}; Max:{}'.format(min(online.InvoiceDate),
                              max(online.InvoiceDate)))
```

```
Min:2010-12-01 08:26:00; Max:2011-12-09 12:50:00
```

Let's create a hypothetical **snapshot_day** data as if we're doing analysis recently

[51]:
```
snapshot_date = max(online.InvoiceDate) + timedelta(days=1)
```

1.3 Caluclate RFM metrics

[52]:
```
# Aggregate data on a customer level
datamart = online.groupby(['CustomerID']).agg({
    'InvoiceDate': lambda x: (snapshot_date - x.max()).days,
    'InvoiceNo': 'count',
    'TotalSum': 'sum'
})
# rename columns for easier interpretation
datamart.rename(columns ={'InvoiceDate':'Recency',
                          'InvoiceNo': 'Frequency',
                          'TotalSum': 'MonetaryValue'}, inplace = True)
```

1.3.1 Final RFM values

Our table for RFM segmentation is completed!

[53]:
```
# check the first rows
datamart.head()
```

[53]:

CustomerID	Recency	MonetaryValue	Frequency
12346.0	326	0.00	2
12347.0	2	4310.00	182
12348.0	75	1797.24	31

```
12349.0          19        1757.55          73
12350.0         310         334.40          17
```

1.3.2 Will calculate quartile value for each column and name then R, F, M

- Recency quartile

[54]:
```
r_labels = (range(4, 0, -1))
r_quartiles = pd.qcut(datamart['Recency'], 4, labels = r_labels)
datamart = datamart.assign(R=r_quartiles.values)
datamart.head(2)
```

[54]:
```
            Recency  MonetaryValue  Frequency  R
CustomerID
12346.0         326            0.0          2  1
12347.0           2         4310.0        182  4
```

- Frequency and Monetary quartile

[55]:
```
f_labels = range(1, 5)
m_labels = range(1, 5)
f_quartiles = pd.qcut(datamart['Frequency'], 4, labels = f_labels)
m_quartiles = pd.qcut(datamart['MonetaryValue'], 4, labels = m_labels)

datamart = datamart.assign(F = f_quartiles.values)
datamart = datamart.assign(M = m_quartiles.values)
datamart.head()
```

[55]:
```
            Recency  MonetaryValue  Frequency  R  F  M
CustomerID
12346.0         326           0.00          2  1  1  1
12347.0           2        4310.00        182  4  4  4
12348.0          75        1797.24         31  2  2  4
12349.0          19        1757.55         73  3  3  4
12350.0         310         334.40         17  1  1  2
```

1.4 Build RFM Segment and RFM Score

- Concatenate RFM quartile values to *RFM_Segment*
- Sum RFM quartiles values to *RFM_Score*

[56]:
```
def join_rfm(x): return str(x['R']) + str(x['F']) + str(x['M'])

datamart['RFM_Segment'] = datamart.apply(join_rfm, axis = 1)
datamart['RFM_Score'] = datamart[['R','F','M']].sum(axis = 1)
```

1.5 Final result:

[57]:
```
datamart.to_excel('../data/datamart.xlsx') # save for later use
datamart.head()
```

[57]:

CustomerID	Recency	MonetaryValue	Frequency	R	F	M	RFM_Segment	RFM_Score
12346.0	326	0.00	2	1	1	1	111	3.0
12347.0	2	4310.00	182	4	4	4	444	12.0
12348.0	75	1797.24	31	2	2	4	224	8.0
12349.0	19	1757.55	73	3	3	4	334	10.0
12350.0	310	334.40	17	1	1	2	112	4.0

1.6 Analyzing RFM Segments

1.6.1 Largest RFM Segments

[58]:
```
datamart.groupby('RFM_Segment').size().sort_values(ascending=False)[:10]
```

[58]:
```
RFM_Segment
444    471
111    392
122    209
344    206
211    181
333    176
222    173
233    164
433    156
322    126
dtype: int64
```

1.6.2 Filtering on RFM segments

- Select bottom RFM segment ”111” and view top 5 rows

[59]:
```
datamart[datamart['RFM_Segment']=='111'].head(5)
```

[59]:

CustomerID	Recency	MonetaryValue	Frequency	R	F	M	RFM_Segment	RFM_Score
12346.0	326	0.0	2	1	1	1	111	3.0
12353.0	204	89.0	4	1	1	1	111	3.0
12361.0	287	189.9	10	1	1	1	111	3.0
12401.0	303	84.3	5	1	1	1	111	3.0
12402.0	323	225.6	11	1	1	1	111	3.0

1.6.3 Summary metrics per RFM Score

[60]:
```
datamart.groupby('RFM_Score').agg({
    'Recency': 'mean',
    'Frequency': 'mean',
    'MonetaryValue': ['mean', 'count']
}).round(1)
```

[60]:

	MonetaryValue		Recency	Frequency
	mean	count	mean	mean
RFM_Score				
3.0	109.1	392	264.8	7.8
4.0	227.1	391	174.5	13.9
5.0	346.8	517	153.0	21.2
6.0	491.8	468	94.3	28.5
7.0	724.2	447	78.8	39.7
8.0	974.7	467	62.7	57.0
9.0	1369.6	411	44.2	79.0
10.0	1894.0	440	31.3	115.3
11.0	3845.7	368	20.5	193.9
12.0	8850.7	471	6.7	371.8

1.7 Grouping into named segments

- Use RFM score to group customers in **Gold**, **Silver** and **Bronze** segments

[61]:
```
def segment_me(df):
    if df['RFM_Score'] >= 9:
        return 'Gold'
    elif (df['RFM_Score'] >= 5) and (df['RFM_Score'] < 9):
        return 'Silver'
    else:
        return 'Bronze'


datamart['General_Segment'] = datamart.apply(segment_me, axis=1)
datamart.head()
```

[61]:

CustomerID	Recency	MonetaryValue	Frequency	R	F	M	RFM_Segment	RFM_Score	\
12346.0	326	0.00	2	1	1	1	111	3.0	
12347.0	2	4310.00	182	4	4	4	444	12.0	
12348.0	75	1797.24	31	2	2	4	224	8.0	
12349.0	19	1757.55	73	3	3	4	334	10.0	
12350.0	310	334.40	17	1	1	2	112	4.0	

```
            General_Segment
CustomerID
12346.0              Bronze
12347.0                Gold
12348.0              Silver
12349.0                Gold
12350.0              Bronze
```

[62]:
```
datamart.groupby('General_Segment').agg({
    'Recency' : 'mean',
    'Frequency': 'mean',
    'MonetaryValue': ['mean', 'count']
}).round(1)
```

[62]:
```
                MonetaryValue       Recency Frequency
                         mean count    mean      mean
General_Segment
Bronze                  168.0   783   219.7      10.9
Gold                   4130.3  1690    25.2     195.1
Silver                  625.8  1899    98.9      36.1
```

[63]:
```
datamart
```

[63]:
```
             Recency  MonetaryValue  Frequency  R  F  M RFM_Segment  RFM_Score  \
CustomerID
12346.0          326           0.00          2  1  1  1         111        3.0
12347.0            2        4310.00        182  4  4  4         444       12.0
12348.0           75        1797.24         31  2  2  4         224        8.0
12349.0           19        1757.55         73  3  3  4         334       10.0
12350.0          310         334.40         17  1  1  2         112        4.0
…                …              …          … .. .. ..         …          …
18280.0          278         180.60         10  1  1  1         111        3.0
18281.0          181          80.82          7  1  1  1         111        3.0
18282.0            8         176.60         13  4  1  1         411        6.0
18283.0            4        2094.88        756  4  4  4         444       12.0
18287.0           43        1837.28         70  3  3  4         334       10.0

            General_Segment
CustomerID
12346.0              Bronze
12347.0                Gold
12348.0              Silver
12349.0                Gold
12350.0              Bronze
…                     …
18280.0              Bronze
18281.0              Bronze
```

```
18282.0             Silver
18283.0               Gold
18287.0               Gold

[4372 rows x 9 columns]
```