Customer Segmentation and Profiling

December 23, 2021

[1]:

```python
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns

from scipy import stats
from sklearn.preprocessing import StandardScaler
from sklearn.manifold import TSNE
from sklearn.cluster import KMeans

from feature_engine.outlier_removers import Winsorizer
```

```
In C:\Users\olale\Anaconda3\lib\site-packages\matplotlib\mpl-
data\stylelib\_classic_test.mplstyle:
The text.latex.preview rcparam was deprecated in Matplotlib 3.3 and will be
removed two minor releases later.
In C:\Users\olale\Anaconda3\lib\site-packages\matplotlib\mpl-
data\stylelib\_classic_test.mplstyle:
The mathtext.fallback_to_cm rcparam was deprecated in Matplotlib 3.3 and will be
removed two minor releases later.
In C:\Users\olale\Anaconda3\lib\site-packages\matplotlib\mpl-
data\stylelib\_classic_test.mplstyle: Support for setting the
'mathtext.fallback_to_cm' rcParam is deprecated since 3.3 and will be removed
two minor releases later; use 'mathtext.fallback : 'cm' instead.
In C:\Users\olale\Anaconda3\lib\site-packages\matplotlib\mpl-
data\stylelib\_classic_test.mplstyle:
The validate_bool_maybe_none function was deprecated in Matplotlib 3.3 and will
be removed two minor releases later.
In C:\Users\olale\Anaconda3\lib\site-packages\matplotlib\mpl-
data\stylelib\_classic_test.mplstyle:
The savefig.jpeg_quality rcparam was deprecated in Matplotlib 3.3 and will be
removed two minor releases later.
In C:\Users\olale\Anaconda3\lib\site-packages\matplotlib\mpl-
data\stylelib\_classic_test.mplstyle:
The keymap.all_axes rcparam was deprecated in Matplotlib 3.3 and will be removed
two minor releases later.
In C:\Users\olale\Anaconda3\lib\site-packages\matplotlib\mpl-
data\stylelib\_classic_test.mplstyle:
```

```
The animation.avconv_path rcparam was deprecated in Matplotlib 3.3 and will be
removed two minor releases later.
In C:\Users\olale\Anaconda3\lib\site-packages\matplotlib\mpl-
data\stylelib\_classic_test.mplstyle:
The animation.avconv_args rcparam was deprecated in Matplotlib 3.3 and will be
removed two minor releases later.
```

[2]:

```
df = pd.read_excel('Trifactor final dataset.xlsx')
df
```

[2]:

```
           Index       Date event_type  product_id  category_id  \
0            0.0 2019-10-11       view      5016.0        183.0
1            1.0 2019-10-27       view      4240.0         43.0
2            2.0 2019-11-25       view      7721.0        208.0
3            3.0 2019-11-12       view       597.0         71.0
4            4.0 2019-11-13       view     13709.0         70.0
…          …        …        …         …          …
235996  235996.0 2019-11-17       cart       719.0         71.0
235997  235997.0 2019-11-17       cart       718.0         71.0
235998  235998.0 2019-11-16       cart       415.0         71.0
235999  235999.0 2019-11-15       cart      4649.0        175.0
236000       NaN        NaT        NaN         NaN          NaN

                        category_code    brand    price    user_id  \
0             appliances.kitchen.oven    artel    36.01  1095536.0
1                electronics.video.tv       lg  2445.08  1153084.0
2       appliances.environment.vacuum  philips   257.38   364298.0
3              electronics.smartphone   huawei   163.20  3536496.0
4               electronics.telephone    nokia    21.85  3542250.0
…                               …      …      …        …
235996         electronics.smartphone  samsung   298.33   450109.0
235997         electronics.smartphone  samsung   298.07  1516528.0
235998         electronics.smartphone  samsung    94.96  2816015.0
235999        appliances.kitchen.hood    bosch   144.15   247855.0
236000                            NaN      NaN      NaN        NaN

        user_session State  User_Score    Year  Quarter      Month  \
0          5342766.0    PA         3.0  2019.0      4.0    October
1          4600705.0    CT         4.0  2019.0      4.0    October
2          5991663.0    HI         3.0  2019.0      4.0   November
3          9397681.0    NH         3.0  2019.0      4.0   November
4          8711820.0    NV         1.0  2019.0      4.0   November
…              …   …         …     …      …       …
235996     9681143.0   NaN         NaN  2019.0      4.0   November
235997     3956291.0   NaN         NaN  2019.0      4.0   November
235998    12343925.0   NaN         NaN  2019.0      4.0   November
235999     9159758.0   NaN         NaN  2019.0      4.0   November
```

```
236000           NaN   NaN         NaN     NaN      NaN       NaN

        Week of Year Name of Day  Hour Event_Time
0               41.0      Friday  16.0   16:57:06
1               44.0      Sunday  14.0   14:23:53
2               48.0      Monday  12.0   12:57:47
3               46.0     Tuesday  19.0   19:50:55
4               46.0   Wednesday   4.0   04:59:13
…              …         …   …        …
235996          47.0      Sunday  16.0   16:07:29
235997          47.0      Sunday  12.0   12:39:00
235998          46.0    Saturday  14.0   14:02:53
235999          46.0      Friday   9.0   09:17:27
236000           NaN         NaN   NaN       None

[236001 rows x 19 columns]
```

[3]:
```
df.dtypes
```

[3]:
```
Index                    float64
Date              datetime64[ns]
event_type                object
product_id               float64
category_id              float64
category_code             object
brand                     object
price                    float64
user_id                  float64
user_session             float64
State                     object
User_Score               float64
Year                     float64
Quarter                  float64
Month                     object
Week of Year             float64
Name of Day               object
Hour                     float64
Event_Time                object
dtype: object
```

[4]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 236001 entries, 0 to 236000
Data columns (total 19 columns):
 #   Column          Non-Null Count   Dtype
---  ------          --------------   -----
 0   Index           236000 non-null  float64
 1   Date           236000 non-null  datetime64[ns]
 2   event_type     236000 non-null  object
 3   product_id     236000 non-null  float64
 4   category_id    236000 non-null  float64
 5   category_code  236000 non-null  object
 6   brand          236000 non-null  object
 7   price          236000 non-null  float64
 8   user_id        236000 non-null  float64
 9   user_session   236000 non-null  float64
 10  State          204630 non-null  object
 11  User_Score     204630 non-null  float64
 12  Year           236000 non-null  float64
 13  Quarter        236000 non-null  float64
 14  Month          236000 non-null  object
 15  Week of Year   236000 non-null  float64
 16  Name of Day    236000 non-null  object
 17  Hour           236000 non-null  float64
 18  Event_Time     236000 non-null  object
dtypes: datetime64[ns](1), float64(11), object(7)
memory usage: 34.2+ MB
```

```
[5]: df.isnull().sum()
```

```
[5]: Index                1
     Date                 1
     event_type           1
     product_id           1
     category_id          1
     category_code        1
     brand                1
     price                1
     user_id              1
     user_session         1
     State            31371
     User_Score       31371
     Year                 1
     Quarter              1
     Month                1
     Week of Year         1
     Name of Day          1
     Hour                 1
     Event_Time           1
     dtype: int64
```

```
[7]: df.skew()
```

```
[7]: Index            0.000000
     product_id       1.473684
     category_id      1.358517
     price            1.956892
     user_id          0.431609
     user_session    -0.000288
     User_Score       0.003175
     Year             0.000000
     Quarter          0.000000
     Week of Year    -0.421331
     Hour            -0.013025
     dtype: float64
```

```
[8]: df.kurtosis()
```

```
[8]: Index           -1.200000
     product_id       1.437978
     category_id      1.381260
     price            4.669445
     user_id         -0.981636
     user_session    -1.199220
     User_Score      -1.360660
     Year             0.000000
     Quarter          0.000000
     Week of Year    -1.017602
     Hour            -0.932056
     dtype: float64
```

```
[9]: column=['Index']
     df.drop(column, axis=1, inplace=True)
     df.head()
```

```
[9]:         Date event_type  product_id  category_id  \
     0 2019-10-11       view      5016.0        183.0
     1 2019-10-27       view      4240.0         43.0
     2 2019-11-25       view      7721.0        208.0
     3 2019-11-12       view       597.0         71.0
     4 2019-11-13       view     13709.0         70.0

                    category_code    brand    price    user_id  user_session  \
     0        appliances.kitchen.oven    artel    36.01  1095536.0     5342766.0
     1           electronics.video.tv       lg  2445.08  1153084.0     4600705.0
     2  appliances.environment.vacuum  philips   257.38   364298.0     5991663.0
     3         electronics.smartphone   huawei   163.20  3536496.0     9397681.0
     4          electronics.telephone    nokia    21.85  3542250.0     8711820.0

       State  User_Score    Year  Quarter     Month  Week of Year Name of Day  \
     0    PA          3.0  2019.0       4.0    October          41.0        Friday
     1    CT          4.0  2019.0       4.0    October          44.0        Sunday
     2    HI          3.0  2019.0       4.0   November          48.0        Monday
     3    NH          3.0  2019.0       4.0   November          46.0       Tuesday
     4    NV          1.0  2019.0       4.0   November          46.0     Wednesday

        Hour Event_Time
     0  16.0   16:57:06
     1  14.0   14:23:53
     2  12.0   12:57:47
     3  19.0   19:50:55
     4   4.0   04:59:13
```

[13]:

```
df.describe()
```

[13]:

```
          product_id    category_id          price       user_id  \
count  236000.000000  236000.000000  236000.000000  2.360000e+05
mean     5211.558271      94.114797     390.030293  1.587240e+06
std      6334.376969      55.245882     382.386132  1.136710e+06
min         1.000000       0.000000       1.260000  9.900000e+01
25%       707.000000      71.000000     135.010000  5.734125e+05
50%      1197.000000      71.000000     249.660000  1.409442e+06
75%      7813.000000     115.000000     501.920000  2.461314e+06
max     26295.000000     305.000000    2574.040000  4.062342e+06

       user_session     User_Score      Year   Quarter   Week of Year  \
count  2.360000e+05  204630.000000  236000.0  236000.0  236000.000000
mean   7.530784e+06       2.498045    2019.0       4.0      44.673119
std    4.347473e+06       1.118328       0.0       0.0       2.411971
min    1.500000e+01       1.000000    2019.0       4.0      40.000000
25%    3.758475e+06       1.000000    2019.0       4.0      43.000000
50%    7.543186e+06       2.000000    2019.0       4.0      45.000000
75%    1.129546e+07       3.000000    2019.0       4.0      47.000000
max    1.506575e+07       4.000000    2019.0       4.0      48.000000

                Hour
count  236000.000000
mean       11.091169
std         5.225956
min         0.000000
25%         7.000000
50%        11.000000
75%        15.000000
max        23.000000
```

[14]:

```
import datetime as dt
NOW = dt.date(2019,12,30)
```

[15]:
```
df['date'] = pd.DatetimeIndex(df.Date).date
```

[16]:
```
df.head()
```

[16]:
```
         Date event_type  product_id  category_id  \
0 2019-10-11       view      5016.0        183.0
1 2019-10-27       view      4240.0         43.0
2 2019-11-25       view      7721.0        208.0
3 2019-11-12       view       597.0         71.0
4 2019-11-13       view     13709.0         70.0

                 category_code    brand    price    user_id  user_session  \
0        appliances.kitchen.oven    artel    36.01  1095536.0     5342766.0
1           electronics.video.tv       lg  2445.08  1153084.0     4600705.0
2  appliances.environment.vacuum  philips   257.38   364298.0     5991663.0
3          electronics.smartphone   huawei   163.20  3536496.0     9397681.0
4           electronics.telephone    nokia    21.85  3542250.0     8711820.0

  State  User_Score    Year  Quarter     Month  Week of Year Name of Day  \
0    PA         3.0  2019.0      4.0   October          41.0      Friday
1    CT         4.0  2019.0      4.0   October          44.0      Sunday
2    HI         3.0  2019.0      4.0  November          48.0      Monday
3    NH         3.0  2019.0      4.0  November          46.0     Tuesday
4    NV         1.0  2019.0      4.0  November          46.0   Wednesday

   Hour Event_Time        date
0  16.0   16:57:06  2019-10-11
1  14.0   14:23:53  2019-10-27
2  12.0   12:57:47  2019-11-25
3  19.0   19:50:55  2019-11-12
4   4.0   04:59:13  2019-11-13
```

[32]:
```
df_recency = df.groupby(['user_id'],as_index=False)['date'].max()
df_recency.columns = ['user_id','Last_Purchase_Date']
```

[33]:
```
df_recency['Recency'] = df_recency.Last_Purchase_Date.apply(lambda x:(NOW - x).
 ↪days)
```

[34]:
```
df_recency.head()
```

[34]:
```
   user_id Last_Purchase_Date  Recency
0     99.0         2019-11-12       48
1    111.0         2019-11-30       30
2    226.0         2019-11-28       32
3    239.0         2019-11-13       47
4    244.0         2019-11-19       41
```

```
[35]: df_recency.drop(columns=['Last_Purchase_Date'],inplace=True)
```

```
[36]: FM_Table = df.groupby('user_id').agg({'product_id'   : lambda x:len(x),
                                            'price'   : lambda x:x.sum()})
```

```
[38]: FM_Table.rename(columns = {'product_id' :'Frequency',
                                  'price':'Monetary_Value'},inplace= True)
```

```
[39]: FM_Table.head()
```

```
[39]:          Frequency  Monetary_Value
      user_id
      99.0           1.0          257.15
      111.0          1.0          257.09
      226.0          1.0          743.62
      239.0          1.0          360.09
      244.0          1.0           97.56
```

```
[40]: RFM_Table = df_recency.merge(FM_Table,left_on='user_id',right_on='user_id')
      RFM_Table.head()
```

```
[40]:    user_id  Recency  Frequency  Monetary_Value
      0     99.0       48        1.0          257.15
      1    111.0       30        1.0          257.09
      2    226.0       32        1.0          743.62
      3    239.0       47        1.0          360.09
      4    244.0       41        1.0           97.56
```

```
[42]: df[df.user_id == 99]
```

```
[42]:             Date event_type  product_id  category_id          category_code  \
      66708 2019-11-12       view     16570.0         91.0  furniture.bedroom.bed

            brand   price  user_id  user_session State  User_Score    Year  Quarter  \
      66708    sv  257.15     99.0    11554527.0    CO         4.0  2019.0      4.0

                Month  Week of Year Name of Day  Hour Event_Time        date
      66708  November          46.0     Tuesday   3.0   03:30:32  2019-11-12
```

```
[44]: (NOW - dt.date(2019,11,12)).days == 48
```

```
[44]: True
```

```
[45]: quantiles = RFM_Table.quantile(q=[0.25,0.50,0.75])
      quantiles = quantiles.to_dict()
```

```
[46]: segmented_rfm = RFM_Table.copy()
```

[48]:

```
def RScore(x,p,d):
    if x <= d[p][0.25]:
        return 1
    elif x <= d[p][0.50]:
        return 2
    elif x <= d[p][0.75]:
        return 3
    else:
        return 4

def FMScore(x,p,d):
    if x <= d[p][0.25]:
        return 4
    elif x <= d[p][0.50]:
        return 3
    elif x <= d[p][0.75]:
        return 2
    else:
        return 1
```

[49]:

```
segmented_rfm['R_quartile'] = segmented_rfm['Recency'].apply(RScore,␣
 ↪args=('Recency',quantiles))
segmented_rfm['F_quartile'] = segmented_rfm['Frequency'].apply(FMScore,␣
 ↪args=('Frequency',quantiles))
segmented_rfm['M_quartile'] = segmented_rfm['Monetary_Value'].apply(FMScore,␣
 ↪args=('Monetary_Value',quantiles))
segmented_rfm.head()
```

[49]:

```
   user_id  Recency  Frequency  Monetary_Value  R_quartile  F_quartile  \
0     99.0       48        1.0          257.15           2           4
1    111.0       30        1.0          257.09           1           4
2    226.0       32        1.0          743.62           1           4
3    239.0       47        1.0          360.09           2           4
4    244.0       41        1.0           97.56           1           4

   M_quartile
0           3
1           3
2           1
3           2
4           4
```

[50]:

```
segmented_rfm['RFM_Segment'] = segmented_rfm.R_quartile.map(str)+segmented_rfm.
 ↪F_quartile.map(str)+segmented_rfm.M_quartile.map(str)
segmented_rfm.head()
```

```
[50]:    user_id  Recency  Frequency  Monetary_Value  R_quartile  F_quartile  \
      0     99.0       48        1.0          257.15           2           4
      1    111.0       30        1.0          257.09           1           4
      2    226.0       32        1.0          743.62           1           4
      3    239.0       47        1.0          360.09           2           4
      4    244.0       41        1.0           97.56           1           4

         M_quartile RFM_Segment
      0           3         243
      1           3         143
      2           1         141
      3           2         242
      4           4         144
```

```
[51]: segmented_rfm['RFM_Score'] =␣
       ↪segmented_rfm[['R_quartile','F_quartile','M_quartile']].sum(axis=1)
      segmented_rfm.head()
```

```
[51]:    user_id  Recency  Frequency  Monetary_Value  R_quartile  F_quartile  \
      0     99.0       48        1.0          257.15           2           4
      1    111.0       30        1.0          257.09           1           4
      2    226.0       32        1.0          743.62           1           4
      3    239.0       47        1.0          360.09           2           4
      4    244.0       41        1.0           97.56           1           4

         M_quartile RFM_Segment  RFM_Score
      0           3         243          9
      1           3         143          8
      2           1         141          6
      3           2         242          8
      4           4         144          9
```

```
[52]: print("Best Customers:␣
       ↪",len(segmented_rfm[segmented_rfm['RFM_Segment']=='111']))
      print('Loyal Customers: ',len(segmented_rfm[segmented_rfm['F_quartile']==1]))
      print("Big Spenders: ",len(segmented_rfm[segmented_rfm['M_quartile']==1]))
      print('Almost Lost: ', len(segmented_rfm[segmented_rfm['RFM_Segment']=='134']))
      print('Lost Customers:␣
       ↪',len(segmented_rfm[segmented_rfm['RFM_Segment']=='344']))
      print('Lost Cheap Customers:␣
       ↪',len(segmented_rfm[segmented_rfm['RFM_Segment']=='444']))
```

```
Best Customers:  5089
Loyal Customers:  24182
Big Spenders:  51153
Almost Lost:  0
Lost Customers:  12178
Lost Cheap Customers:  12928
```

```
[53]: segmented_rfm['RFM_Score'].unique()
```

```
[53]: array([ 9,  8,  6,  7, 12, 10,  3, 11,  5,  4], dtype=int64)
```

```
[54]: segmented_rfm.groupby('RFM_Score').agg({
          'Recency': 'mean',
          'Frequency': 'mean',
          'Monetary_Value': ['mean', 'count'] }).round(1)
```

```
[54]:           Recency Frequency Monetary_Value
                   mean      mean           mean  count
      RFM_Score
      3            37.5       2.6         1450.8   5089
      4            42.5       2.3          981.6   6651
      5            51.5       2.2          829.5   5650
      6            45.3       1.3          923.3  14110
      7            43.8       1.1          643.4  22048
      8            47.3       1.0          487.1  33055
      9            54.5       1.0          384.6  46111
      10           62.2       1.0          225.8  34179
      11           70.5       1.0          145.2  24809
      12           80.2       1.0           83.5  12928
```

```
[77]: RFM_Table.describe()
```

```
[77]:             user_id       Recency      Frequency  Monetary_Value
      count  2.046300e+05  204630.00000  204630.000000   204630.000000
      mean   1.618077e+06      55.50735       1.153301      449.822358
      std    1.142119e+06      16.82879       0.505220      492.311531
      min    9.900000e+01      30.00000       1.000000        1.260000
      25%    6.026678e+05      43.00000       1.000000      143.180000
      50%    1.444042e+06      51.00000       1.000000      271.460000
      75%    2.512144e+06      70.00000       1.000000      579.160000
      max    4.062342e+06      90.00000      30.000000    13589.550000
```