Customer Segmentation Using RFM Analysis

RFM Analysis is a method mainly used in marketing, which uses three factors to segment customers in groups with similar purchasing behaviors.
RFM stands for **Recency**, **Frequency** and **Monetary value**, where:

- **Recency (R):** how recently a customer made a purchase or visited our website?.
- **Frequency (F):** how many often do they make purchase?.
- **Monetary value (M):** how much income we receive from the purchases they make?.

In [1]:

```
# Importing libraries
import numpy as np
import pandas as pd
import datetime as dt
import matplotlib.pyplot as plt
import seaborn  as sns
%matplotlib inline
```

In [2]:

```
# Visualization style
sns.set_style("darkgrid", {"grid.color": ".6", "grid.linestyle": ":"})
```

In [3]:

```
# Importing dataset
data = pd.read_excel('Online Retail.xlsx')
```

In [4]:

```
# Data sample
data.sample(5)
```

Out[4]:

	InvoiceNo	StockCode	Description	Quantity	InvoiceDate	UnitPrice	CustomerID	Cou
212432	555479	22427	ENAMEL FLOWER JUG CREAM	24	2011-06-03 12:35:00	5.45	15189.0	U King
24671	538349	22714	CARD BIRTHDAY COWBOY	1	2010-12-10 14:59:00	0.85	NaN	U King
421005	572913	22562	MONSTERS STENCIL CRAFT	2	2011-10-26 16:21:00	1.25	15993.0	U King
324187	565396	21936	RED RETROSPOT PICNIC BAG	1	2011-09-02 16:39:00	5.79	NaN	U King
332017	566053	22412	METAL SIGN NEIGHBOURHOOD WITCH	4	2011-09-08 14:57:00	2.10	14410.0	U King

In [5]:

```
data.shape
```

Out[5]:

```
(541909, 8)
```

In [6]:

```
data.columns
```

Out[6]:

```
Index(['InvoiceNo', 'StockCode', 'Description', 'Quantity', 'InvoiceDate',
       'UnitPrice', 'CustomerID', 'Country'],
      dtype='object')
```

In [7]:

```
# Data information
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 541909 entries, 0 to 541908
Data columns (total 8 columns):
 #   Column       Non-Null Count   Dtype         
---  ------       --------------   -----         
 0   InvoiceNo    541909 non-null  object        
 1   StockCode    541909 non-null  object        
 2   Description  540455 non-null  object        
 3   Quantity     541909 non-null  int64         
 4   InvoiceDate  541909 non-null  datetime64[ns]
 5   UnitPrice    541909 non-null  float64       
 6   CustomerID   406829 non-null  float64       
 7   Country      541909 non-null  object        
dtypes: datetime64[ns](1), float64(2), int64(1), object(4)
memory usage: 33.1+ MB
```

In [8]:

```
# Data description
data.describe()
```

Out[8]:

	Quantity	UnitPrice	CustomerID
count	541909.000000	541909.000000	406829.000000
mean	9.552250	4.611114	15287.690570
std	218.081158	96.759853	1713.600303
min	-80995.000000	-11062.060000	12346.000000
25%	1.000000	1.250000	13953.000000
50%	3.000000	2.080000	15152.000000
75%	10.000000	4.130000	16791.000000
max	80995.000000	38970.000000	18287.000000

In [9]:

```
print('='*64)
print('The data corresponds from {} to {}'.format(data.InvoiceDate.min(),
                                                   data.InvoiceDate.max()))
print('='*64)
```

```
================================================================
The data corresponds from 2010-12-01 08:26:00 to 2011-12-09 12:50:00
================================================================
```

In [10]:

```
# Drop instances where 'CustomerID' is null value
data = data.dropna(subset=['CustomerID'], axis=0)

# Transform 'CustomerID' to int type
```

```
data['CustomerID'] =  data['CustomerID'].astype('int')

# Calculate total sale
data['Sales'] = data.UnitPrice * data.Quantity

# Extract date - no time
data['InvoiceDate'] = data['InvoiceDate'].dt.date
```

RFM Features

In [11]:

```
# CREATING RFM FEATURES
# ==============================================================================

snapshot_date = data.InvoiceDate.max() + dt.timedelta(days=1)

# Aggregate data on a customer level
datamart = data.groupby('CustomerID').agg({'InvoiceDate': lambda x: (snapshot_da
                                          'InvoiceNo': 'count',
                                          'Sales': 'sum'})

# Rename columns
datamart.rename(columns={'InvoiceDate': 'Recency',
                         'InvoiceNo': 'Frequency',
                         'Sales': 'MonetaryValue'}, inplace=True)

datamart.sample(5)
```

Out[11]:

	Recency	Frequency	MonetaryValue
CustomerID			
14960	9	27	221.27
15235	218	143	2247.51
14382	27	131	626.07
17354	51	16	1393.06
18141	361	1	-35.40

RFM segments and scores

In [12]:

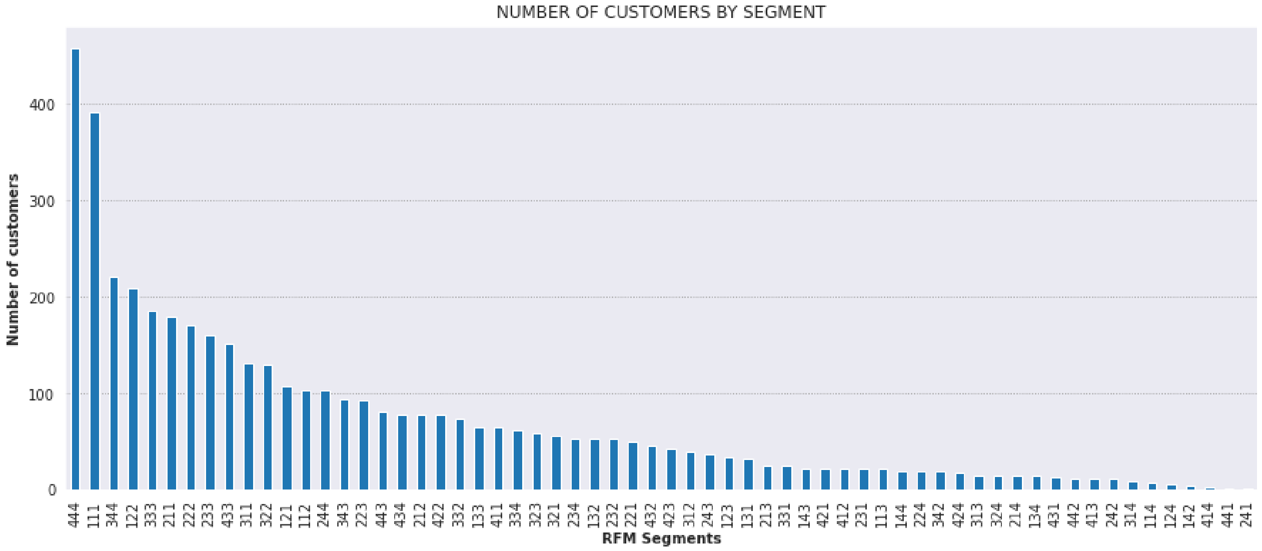
```
# Recency quartiles
r_quartiles = pd.qcut(datamart.Recency, 4, labels=range(4, 0, -1))
datamart = datamart.assign(R = r_quartiles.values)

# Frequency quartiles
f_quartiles = pd.qcut(datamart.Frequency, 4, labels=range(1, 5))
datamart = datamart.assign(F = f_quartiles.values)

# Monetary value quartiles
m_quartiles = pd.qcut(datamart.MonetaryValue, 4, labels=range(1, 5))
datamart = datamart.assign(M = m_quartiles.values)

# Building RFM segments
def rfm_seg(x):
    return str(int(x['R'])) + str(int(x['F'])) + str(int(x['M']))

# Create segment label
datamart['RFM_Segment'] = datamart.apply(rfm_seg, axis=1)

# RFM Score - Sum of scores
datamart['RFM_Score'] = datamart[['R','F','M']].sum(axis=1)

datamart.sample(10)
```

Out[12]:

	Recency	Frequency	MonetaryValue	R	F	M	RFM_Segment	RFM_Score
CustomerID								
13848	93	5	1255.00	2	1	3	213	6
15454	363	25	179.10	1	2	1	121	4
14713	10	341	2664.26	4	4	4	444	12
15810	79	112	1145.43	2	4	3	243	9
16940	53	305	3049.88	2	4	4	244	10
16986	30	3	1873.20	3	1	4	314	8
16655	18	261	3794.52	3	4	4	344	11
15416	65	193	3974.37	2	4	4	244	10
16027	92	17	852.12	2	1	3	213	6
12908	59	4	246.00	2	1	1	211	4

In [13]:

```
# Number of customers by RFM Segment
segments = datamart['RFM_Segment'].value_counts().sort_values(ascending=False)
plt.figure(figsize=(15,6))
plt.title('NUMBER OF CUSTOMERS BY SEGMENT')
segments.plot(kind='bar')
plt.xlabel('RFM Segments', fontweight='bold')
plt.ylabel('Number of customers', fontweight='bold')
plt.grid(axis='x')
plt.show()
```



In [14]:

```python
# Data by RFM score
datamart.groupby('RFM_Score').agg({'Recency': 'mean',
                                   'Frequency': 'mean',
                                   'MonetaryValue': 'mean'}).round(1)
```

Out[14]:

	Recency	Frequency	MonetaryValue
RFM_Score			
3	265.6	7.8	109.1
4	175.6	13.9	227.0
5	152.7	21.1	343.8
6	95.1	28.6	491.7
7	79.5	39.5	725.4
8	63.0	57.1	972.3
9	44.7	78.8	1361.9
10	32.0	115.3	1897.6
11	21.1	199.9	3993.5
12	6.9	372.7	8889.8

Customers Segment Labeling

In [15]:

```python
# Labeling segments
def segment_label(df):
    if df['RFM_Score'] >= 9:
        return 'Gold'
    elif (df['RFM_Score'] >= 6) and (df['RFM_Score'] < 9):
        return 'Silver'
    else:
        return 'Bronze'

# Assigning segment labels
datamart['SegmentLabel'] = datamart.apply(segment_label, axis=1)

datamart.sample(10)
```

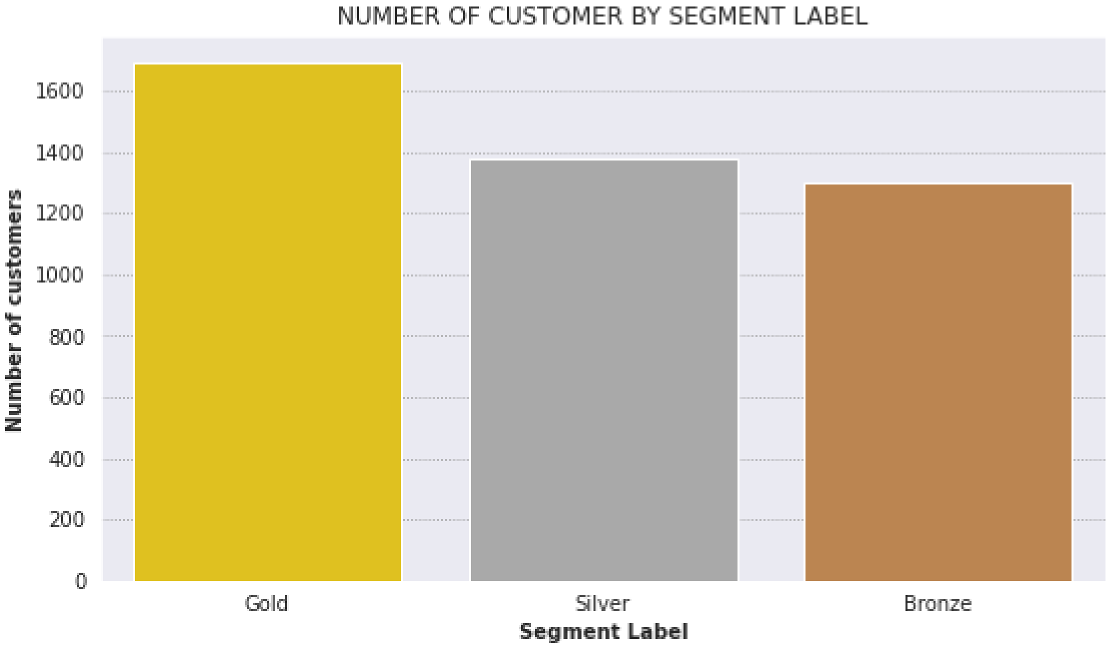
Out[15]:

	Recency	Frequency	MonetaryValue	R	F	M	RFM_Segment	RFM_Score	SegmentLabe
CustomerID									
14514	62	73	1055.35	2	3	3	233	8	Silve
13323	4	21	787.85	4	2	3	423	9	Gol
18079	46	127	3651.97	3	4	4	344	11	Gol
18269	359	8	138.90	1	1	1	111	3	Bronz
14379	45	46	348.10	3	3	2	332	8	Silve
15621	5	18	1158.77	4	2	3	423	9	Gol
15904	9	30	164.68	4	2	1	421	7	Silve
14901	12	98	1414.99	4	3	3	433	10	Gol

	Recency	Frequency	MonetaryValue	R	F	M	RFM_Segment	RFM_Score	SegmentLabe
CustomerID									
13636	37	64	941.62	3	3	3	333	9	Gol
15453	2	286	1388.37	4	4	3	443	11	Gol

In [16]:

```
# Visualization of number of customers by segment label
plt.figure(figsize=(9,5))
plt.title('NUMBER OF CUSTOMER BY SEGMENT LABEL')
sns.countplot(x=datamart.SegmentLabel, order=['Gold','Silver','Bronze'], palette
plt.xlabel('Segment Label', fontweight='bold')
plt.ylabel('Number of customers',fontweight='bold')
plt.show()
```



In [17]:

```
datamart.groupby('SegmentLabel').mean()
```

Out[17]:

	Recency	Frequency	MonetaryValue	RFM_Score
SegmentLabel				
Bronze	193.511914	14.951576	238.276765	4.099154
Gold	25.889480	194.922577	4127.427164	10.518322
Silver	79.188542	41.746193	729.848427	7.000000