Market Segmentation

Learning Your Clients

Recency, Frequency and Monetary (RFM) matrix principle is a customer segmentation technique that uses past purchase behavior to divide customers into groups. RFM can be used to build a predictive model that effectively finds a company's best customers. By dividing target market into specific groups, companies can sell more products with less marketing expenses.

An objective statistical analysis like this can also be used to identify the type of customer that would respond to Politico Pro.

In addition to predicting behavior, segmentation can help provide insights about customer habits and preferences. These insights can further tailor marketing campaigns and improve current customer experience.

The RFM model answers the following questions:

- When was the last time they purchased? (Recency)
- How often and for how long have they purchased? (Frequency)
- How much have they purchased? (Monetary Value)

This project uses the Online Retail Data (http://archive.ics.uci.edu/ml/datasets/online+retail) from the UCI Repository, and will showcase the following skills:

- Building processes to help identify key marketing metrics, market measurement, performance and analysis
- Experimental design
- Prioritization on building innovative models

Depending on the historical data available, an algorithm using a logistic regression model, a random forest model, or clustering model can also be built.

-**Sarah Aristil**, Marketing Analyst Applicant

In [1]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
%matplotlib inline
```

In [2]:
```
from pandas import ExcelWriter
from pandas import ExcelFile
import warnings
warnings.filterwarnings('ignore')
```

In [3]:
```
df = pd.read_excel('Online_Retail.xlsx', sheetname='Online_Retail')
```

Exploring Dataset

In [4]:
```
df.head()
```

Out[4]:

	InvoiceNo	StockCode	Description	Quantity	InvoiceDate	UnitPrice	CustomerID	Country
0	536365	85123A	WHITE HANGING HEART T-LIGHT HOLDER	6	2010-12-01 08:26:00	2.55	17850.0	United Kingdom
1	536365	71053	WHITE METAL LANTERN	6	2010-12-01 08:26:00	3.39	17850.0	United Kingdom
2	536365	84406B	CREAM CUPID HEARTS COAT HANGER	8	2010-12-01 08:26:00	2.75	17850.0	United Kingdom
3	536365	84029G	KNITTED UNION FLAG HOT WATER BOTTLE	6	2010-12-01 08:26:00	3.39	17850.0	United Kingdom
4	536365	84029E	RED WOOLLY HOTTIE WHITE HEART.	6	2010-12-01 08:26:00	3.39	17850.0	United Kingdom

In [5]:
```
online_retail = df
```

In [6]:
```
online_retail.columns
```

Out[6]:
```
Index(['InvoiceNo', 'StockCode', 'Description', 'Quantity', 'InvoiceDate',
       'UnitPrice', 'CustomerID', 'Country'],
      dtype='object')
```

In [7]:
```
online_retail.Country.nunique()
```

Out[7]:
```
38
```

In [8]:
```
online_retail.Country.unique()
```

Out[8]:
```
array(['United Kingdom', 'France', 'Australia', 'Netherlands', 'Germany',
       'Norway', 'EIRE', 'Switzerland', 'Spain', 'Poland', 'Portugal',
       'Italy', 'Belgium', 'Lithuania', 'Japan', 'Iceland',
       'Channel Islands', 'Denmark', 'Cyprus', 'Sweden', 'Austria',
       'Israel', 'Finland', 'Bahrain', 'Greece', 'Hong Kong', 'Singapore',
       'Lebanon', 'United Arab Emirates', 'Saudi Arabia',
       'Czech Republic', 'Canada', 'Unspecified', 'Brazil', 'USA',
       'European Community', 'Malta', 'RSA'], dtype=object)
```

In [9]:
```
customer_country=online_retail[['Country','CustomerID']].drop_duplicates()
customer_country.groupby(['Country'])['CustomerID'].aggregate('count').reset_index().sort_values('Custom
```

Out[9]:

	Country	CustomerID
36	United Kingdom	3950
14	Germany	95
13	France	87
31	Spain	31
3	Belgium	25
33	Switzerland	21
27	Portugal	19
19	Italy	15
12	Finland	12
1	Austria	11
25	Norway	10
24	Netherlands	9
0	Australia	9
6	Channel Islands	9
9	Denmark	9
7	Cyprus	8
32	Sweden	8
20	Japan	8
26	Poland	6
34	USA	4
5	Canada	4
37	Unspecified	4
18	Israel	4
15	Greece	4
10	EIRE	3
23	Malta	2
35	United Arab Emirates	2
2	Bahrain	2
22	Lithuania	1
8	Czech Republic	1
21	Lebanon	1
28	RSA	1
29	Saudi Arabia	1
30	Singapore	1
17	Iceland	1
4	Brazil	1
11	European Community	1
16	Hong Kong	0

More than 90% of the customers in the data are from the United Kingdom, so the model will be restricted to the United Kingdom only.

In [10]:
```
online_retail = online_retail.loc[online_retail['Country'] == 'United Kingdom']
```

Cleaning data

In [11]:
```
#Check for missing values
online_retail.isnull().sum(axis=0)
```

Out[11]:
```
InvoiceNo           0
StockCode           0
Description      1454
Quantity            0
InvoiceDate         0
UnitPrice           0
CustomerID     133600
Country             0
dtype: int64
```

In [12]:
```
#There are 133,600 missing values in the CustomerID column
#The analysis is based on customers, so we'll remove these missing values
online_retail = online_retail[pd.notnull(online_retail['UnitPrice'])]
```

In [13]:
```
#Remove the negative values in Quantity column
online_retail.Quantity.min()
```

Out[13]:
```
-80995
```

In [14]:
```
online_retail = online_retail[(online_retail['Quantity']>0)]
online_retail.shape
online_retail.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 486286 entries, 0 to 541893
Data columns (total 8 columns):
InvoiceNo      486286 non-null object
StockCode      486286 non-null object
Description    485694 non-null object
Quantity       486286 non-null int64
InvoiceDate    486286 non-null datetime64[ns]
UnitPrice      486286 non-null float64
CustomerID     354345 non-null float64
Country        486286 non-null object
dtypes: datetime64[ns](1), float64(2), int64(1), object(4)
memory usage: 33.4+ MB
```

There is now 354,345 rows and 8 columns of data.

Building RFM Model

In [15]:
```
#Checking for the unique value for each column
def unique_counts(online_retail):
   for i in online_retail.columns:
       count = online_retail[i].nunique()
       print(i, ": ", count)
unique_counts(online_retail)
```

```
InvoiceNo :  18786
StockCode :  3936
Description :  4058
Quantity :  387
InvoiceDate :  17361
UnitPrice :  1248
CustomerID :  3921
Country :  1
```

In [16]:
```
#Add a total price column
online_retail['TotalPrice'] = online_retail['Quantity'] * online_retail['UnitPrice']
```

In [17]:
```
#Find the first and last order dates in the data.
online_retail['InvoiceDate'].min()
```

Out[17]:
```
Timestamp('2010-12-01 08:26:00')
```

In [18]:
```
online_retail['InvoiceDate'].max()
```

Out[18]:
```
Timestamp('2011-12-09 12:49:00')
```

In [19]:
```
#Recency will be calculated by the date following the last invoice
import datetime as dt
NOW = dt.datetime(2011,12,10)
online_retail['InvoiceDate'] = pd.to_datetime(online_retail['InvoiceDate'])
```

Create a RFM table

In [20]:
```
rfmTable = online_retail.groupby('CustomerID').agg({'InvoiceDate': lambda x: (NOW - x.max()).days, 'Invo
rfmTable['InvoiceDate'] = rfmTable['InvoiceDate'].astype(int)
rfmTable.rename(columns={'InvoiceDate': 'recency',
                         'InvoiceNo': 'frequency',
                         'TotalPrice': 'monetary_value'}, inplace=True)
```

In [21]:
```
rfmTable.head()
```

Out[21]:

CustomerID	recency	frequency	monetary_value
12346.0	325	1	77183.60
12747.0	2	103	4196.01
12748.0	0	4596	33719.73
12749.0	3	199	4090.88
12820.0	3	59	942.34

To intepret the rfm table, CustomerID 12346 has

- frequency: 1
- monetary value: $77,183.60
- recency: 325 days

Now calculate RFM metrics for each customer

In [22]:
```
first_customer = online_retail[online_retail['CustomerID'] == 12346.0]
first_customer
```

Out[22]:

	InvoiceNo	StockCode	Description	Quantity	InvoiceDate	UnitPrice	CustomerID	Country	TotalPrice
61619	541431	23166	MEDIUM CERAMIC TOP STORAGE JAR	74215	2011-01-18 10:01:00	1.04	12346.0	United Kingdom	77183.6

The first customer shopped only once, bought a large quantity (74,215) of one product with a very low unit price.

Next up, splitting the metrics into segments by using quartiles.

In [23]:
```
quantiles = rfmTable.quantile(q=[0.25,0.5,0.75])
quantiles = quantiles.to_dict()
```

In [24]:
```
#Create a segmented RFM table
segmented_rfm = rfmTable
```

In [25]:
```
segmented_rfm.head()
```

Out[25]:

CustomerID	recency	frequency	monetary_value
12346.0	325	1	77183.60
12747.0	2	103	4196.01
12748.0	0	4596	33719.73
12749.0	3	199	4090.88
12820.0	3	59	942.34

Best customers have the lowest recency, highest frequency and high monetary amounts

In [26]:
```
#The four segments
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
```

In [27]:
```
#Add segment numbers
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',quan
segmented_rfm.head()
```

Out[27]:

CustomerID	recency	frequency	monetary_value	r_quartile	f_quartile	m_quartile
12346.0	325	1	77183.60	4	4	1
12747.0	2	103	4196.01	1	1	1
12748.0	0	4596	33719.73	1	1	1
12749.0	3	199	4090.88	1	1	1
12820.0	3	59	942.34	1	2	2

In [28]:
```
segmented_rfm['RFMScore'] = segmented_rfm.r_quartile.map(str) + segmented_rfm.f_quartile.map(str) + segm
segmented_rfm.head()
```

Out[28]:

CustomerID	recency	frequency	monetary_value	r_quartile	f_quartile	m_quartile	RFMScore
12346.0	325	1	77183.60	4	4	1	441
12747.0	2	103	4196.01	1	1	1	111
12748.0	0	4596	33719.73	1	1	1	111
12749.0	3	199	4090.88	1	1	1	111
12820.0	3	59	942.34	1	2	2	122

Below are the top 10 of our best customers. The highest RFM score is 111 which describes customers that bought most recently, most often and spent the most.

In [29]:
```
segmented_rfm[segmented_rfm['RFMScore']=='111'].sort_values('monetary_value', ascending=False).head(10)
```

Out[29]:

CustomerID	recency	frequency	monetary_value	r_quartile	f_quartile	m_quartile	RFMScore
18102.0	0	431	259657.30	1	1	1	111
17450.0	8	337	194550.79	1	1	1	111
17511.0	2	963	91062.38	1	1	1	111
16684.0	4	277	66653.56	1	1	1	111
14096.0	4	5111	65164.79	1	1	1	111
13694.0	3	568	65039.62	1	1	1	111
15311.0	0	2379	60767.90	1	1	1	111
13089.0	2	1818	58825.83	1	1	1	111
15769.0	7	130	56252.72	1	1	1	111
15061.0	3	403	54534.14	1	1	1	111

This table identifies high value customers. Further segmentation would identify loyal customers, big spenders, and lost customers.