RFM Using K means Clustering and Dendograms

December 23, 2021

[1]:
```
import pandas as pd
import numpy as np
import matplotlib as plt
import seaborn as sn
%matplotlib inline
```

```
C:\ProgramData\Anaconda3\lib\site-packages\statsmodels\tools\_testing.py:19:
FutureWarning: pandas.util.testing is deprecated. Use the functions in the
public API at pandas.testing instead.
  import pandas.util.testing as tm
```

[23]:
```
df = pd.read_excel('P6-SuperStoreUS-2015.xlsx')
df
```

[23]:
```
      Row ID Order Priority  Discount  Unit Price  Shipping Cost  Customer ID  \
0      20847           High      0.01        2.84           0.93            3
1      20228  Not Specified      0.02      500.98          26.00            5
2      21776       Critical      0.06        9.48           7.29           11
3      24844         Medium      0.09       78.69          19.99           14
4      24846         Medium      0.08        3.28           2.31           14
…        …              …       …          …             …            …
1947   19842           High      0.01       10.90           7.46         3397
1948   19843           High      0.10        7.99           5.03         3397
1949   26208  Not Specified      0.08       11.97           5.81         3399
1950   24911         Medium      0.10        9.38           4.93         3400
1951   25914           High      0.10      105.98          13.99         3403

          Customer Name       Ship Mode Customer Segment Product Category  \
0         Bonnie Potter     Express Air        Corporate  Office Supplies
1        Ronnie Proctor  Delivery Truck      Home Office        Furniture
2         Marcus Dunlap     Regular Air      Home Office        Furniture
3     Gwendolyn F Tyson     Regular Air   Small Business        Furniture
4     Gwendolyn F Tyson     Regular Air   Small Business  Office Supplies
…                   …             …              …              …
1947        Andrea Shaw     Regular Air   Small Business  Office Supplies
1948        Andrea Shaw     Regular Air   Small Business       Technology
1949        Marvin Reid     Regular Air   Small Business  Office Supplies
1950      Florence Gold     Express Air   Small Business        Furniture
```

```
1951      Tammy Buckley     Express Air        Consumer        Furniture

      …   Region State or Province         City  Postal Code Order Date  \
0     …     West        Washington    Anacortes        98221 2015-01-07
1     …     West        California  San Gabriel        91776 2015-06-13
2     …     East        New Jersey      Roselle         7203 2015-02-15
3     …  Central         Minnesota   Prior Lake        55372 2015-05-12
4     …  Central         Minnesota   Prior Lake        55372 2015-05-12
…   …      …                 …           …          …          …
1947  …  Central          Illinois     Danville        61832 2015-03-11
1948  …  Central          Illinois     Danville        61832 2015-03-11
1949  …  Central          Illinois  Des Plaines        60016 2015-03-29
1950  …     East     West Virginia     Fairmont        26554 2015-04-04
1951  …     West           Wyoming     Cheyenne        82001 2015-02-08

      Ship Date     Profit Quantity ordered new    Sales Order ID
0     2015-01-08     4.5600                    4    13.01    88522
1     2015-06-15  4390.3665                   12  6362.85    90193
2     2015-02-17   -53.8096                   22   211.15    90192
3     2015-05-14   803.4705                   16  1164.45    86838
4     2015-05-13   -24.0300                    7    22.23    86838
…           …          …                     …      …        …
1947 2015-03-12  -116.7600                   18   207.31    87536
1948 2015-03-12  -160.9520                   22   143.12    87536
1949 2015-03-31   -41.8700                    5    59.98    87534
1950 2015-04-04   -24.7104                   15   135.78    87537
1951 2015-02-11   349.4850                    5   506.50    87530

[1952 rows x 25 columns]
```

[24]:
```python
#Convert the date in YYYY-mm-dd HH:MM format and store that date in 'Date'␣
↪column
df['Date']=pd.to_datetime(df['Order Date'], format = '%Y-%m-%d %H:%M:%S')
#Retail_df['Date']=Retail_df['Date'].apply(lambda x: x.strftime('%Y-%d-%m %H:
↪%M'))

# Count the unique no of attributes in Retail data
def unique_counts(df):
   for i in df.columns:
       count = df[i].nunique()
       print(i, ": ", count)
unique_counts(df)
```

```
Row ID :  1951
Order Priority :  6
Discount :  13
Unit Price :  597
Shipping Cost :  497
```

```
Customer ID :  1130
Customer Name :  1130
Ship Mode :  3
Customer Segment :  4
Product Category :  3
Product Sub-Category :  17
Product Container :  7
Product Name :  913
Product Base Margin :  51
Country :  1
Region :  4
State or Province :  49
City :  869
Postal Code :  981
Order Date :  179
Ship Date :  187
Profit :  1898
Quantity ordered new :  76
Sales :  1922
Order ID :  1365
Date :  179
```

[25]:
```
df['Total_Price']=df['Quantity ordered new']*df['Unit Price']
df.head(10)
```

[25]:
```
   Row ID Order Priority  Discount  Unit Price  Shipping Cost  Customer ID  \
0   20847           High      0.01        2.84           0.93            3
1   20228  Not Specified      0.02      500.98          26.00            5
2   21776       Critical      0.06        9.48           7.29           11
3   24844         Medium      0.09       78.69          19.99           14
4   24846         Medium      0.08        3.28           2.31           14
5   24847         Medium      0.05        3.28           4.20           14
6   24848         Medium      0.05        3.58           1.63           14
7   18181       Critical      0.00        4.42           4.99           15
8   20925         Medium      0.01       35.94           6.66           15
9   26267           High      0.04        2.98           1.58           16

       Customer Name       Ship Mode Customer Segment Product Category  …  \
0      Bonnie Potter     Express Air        Corporate  Office Supplies  …
1     Ronnie Proctor  Delivery Truck      Home Office        Furniture  …
2      Marcus Dunlap     Regular Air      Home Office        Furniture  …
3  Gwendolyn F Tyson     Regular Air   Small Business        Furniture  …
4  Gwendolyn F Tyson     Regular Air   Small Business  Office Supplies  …
5  Gwendolyn F Tyson     Regular Air   Small Business  Office Supplies  …
6  Gwendolyn F Tyson     Regular Air   Small Business  Office Supplies  …
7      Timothy Reese     Regular Air   Small Business  Office Supplies  …
8      Timothy Reese     Regular Air   Small Business  Office Supplies  …
```

```
9       Sarah Ramsey     Regular Air   Small Business  Office Supplies  …

           City Postal Code Order Date  Ship Date     Profit  \
0    Anacortes       98221 2015-01-07 2015-01-08     4.5600
1  San Gabriel       91776 2015-06-13 2015-06-15  4390.3665
2      Roselle        7203 2015-02-15 2015-02-17   -53.8096
3   Prior Lake       55372 2015-05-12 2015-05-14   803.4705
4   Prior Lake       55372 2015-05-12 2015-05-13   -24.0300
5   Prior Lake       55372 2015-05-12 2015-05-13   -37.0300
6   Prior Lake       55372 2015-05-12 2015-05-13    -0.7100
7    Smithtown       11787 2015-04-08 2015-04-09   -59.8200
8    Smithtown       11787 2015-05-28 2015-05-28   261.8757
9     Syracuse       13210 2015-02-12 2015-02-15     2.6300

  Quantity ordered new    Sales Order ID       Date Total_Price
0                    4    13.01     88522 2015-01-07       11.36
1                   12  6362.85     90193 2015-06-13     6011.76
2                   22   211.15     90192 2015-02-15      208.56
3                   16  1164.45     86838 2015-05-12     1259.04
4                    7    22.23     86838 2015-05-12       22.96
5                    4    13.99     86838 2015-05-12       13.12
6                    4    14.26     86838 2015-05-12       14.32
7                    7    33.47     86837 2015-04-08       30.94
8                   10   379.53     86839 2015-05-28      359.40
9                    6    18.80     86836 2015-02-12       17.88

[10 rows x 27 columns]
```

[26]:
```
Online_retail_df = df[np.isfinite(df['Customer ID'])]
```

[27]:
```
unique_counts(Online_retail_df)
```

```
Row ID :  1951
Order Priority :  6
Discount :  13
Unit Price :  597
Shipping Cost :  497
Customer ID :  1130
Customer Name :  1130
Ship Mode :  3
Customer Segment :  4
Product Category :  3
Product Sub-Category :  17
Product Container :  7
Product Name :  913
Product Base Margin :  51
Country :  1
Region :  4
```

```
State or Province :  49
City :  869
Postal Code :  981
Order Date :  179
Ship Date :  187
Profit :  1898
Quantity ordered new :  76
Sales :  1922
Order ID :  1365
Date :  179
Total_Price :  1683
```

[28]:
```python
#For the sake of calculating recency and frequency, drop the rows with negative␣
↪values of Quantity and store the data in final_df
final_retail = Online_retail_df[Online_retail_df['Quantity ordered new'] > 0]
```

[29]:
```python
final_retail.shape
```

[29]: (1952, 27)

[30]:
```python
unique_counts(final_retail)
```

```
Row ID :  1951
Order Priority :  6
Discount :  13
Unit Price :  597
Shipping Cost :  497
Customer ID :  1130
Customer Name :  1130
Ship Mode :  3
Customer Segment :  4
Product Category :  3
Product Sub-Category :  17
Product Container :  7
Product Name :  913
Product Base Margin :  51
Country :  1
Region :  4
State or Province :  49
City :  869
Postal Code :  981
Order Date :  179
Ship Date :  187
Profit :  1898
Quantity ordered new :  76
Sales :  1922
Order ID :  1365
Date :  179
```

```
Total_Price :  1683
```

[31]:
```python
type(final_retail['Date'].max())
```

[31]:
```
pandas._libs.tslibs.timestamps.Timestamp
```

[32]:
```python
final_retail['Date'].min()
```

[32]:
```
Timestamp('2015-01-01 00:00:00')
```

[33]:
```python
final_retail['Date'].max()
```

[33]:
```
Timestamp('2015-06-30 00:00:00')
```

[34]:
```python
import datetime as dt
NOW = dt.datetime(2015,6,30)
```

[35]:
```python
rfmTable = final_retail.groupby('Customer ID').agg({'Date': lambda x: (NOW - x.
 ↪max()).days, 'Order ID': lambda x: len(x),'Total_Price': lambda x: x.sum()})
rfmTable['Date'] = rfmTable['Date'].astype(int)
rfmTable.rename(columns={'Date': 'recency',
                         'Order ID': 'frequency',
                       'Total_Price': 'monetary_value'}, inplace=True)
```

[36]:
```python
#rfmTable = pd.merge(mTable, rfTable, on="CustomerID",how = 'inner')
rfmTable.shape
```

[36]:
```
(1130, 3)
```

[ ]:
```
Below are top ten customers after sorting
```

[ ]:
```
```

[37]:
```python
rfmTable.head(10)
```

[37]:

Customer ID	recency	frequency	monetary_value
3	174	1	11.36
5	17	1	6011.76
11	135	1	208.56
14	49	4	1309.44
15	33	2	390.34
16	138	2	1177.78
18	46	1	450.16
19	40	1	233.82
21	40	3	3065.89
24	153	2	57.16

[38]:
```
rfmTable.sort_values(['frequency', 'monetary_value'], ascending=[False, False],␣
 ↪inplace=True)
```

Ascertain your top ten customers based on Frequency and Monetary Value

[39]:
```
rfmTable.head(10)
```

[39]:

Customer ID	recency	frequency	monetary_value
699	0	9	8039.29
2882	0	8	12748.90
3079	19	7	16736.71
2491	65	7	15513.56
1193	2	7	11514.94
3151	27	7	5626.41
1129	17	6	12840.69
3133	118	6	3733.30
2618	99	6	2394.79
693	56	5	14883.71

[40]:
```
from sklearn.cluster import KMeans
from sklearn.preprocessing import StandardScaler
scaler = StandardScaler()
X_scaled = scaler.fit_transform( rfmTable )
```

[41]:
```
clusters = KMeans(3)  # 3 clusters
clusters.fit( X_scaled )
```

[41]:
```
KMeans(algorithm='auto', copy_x=True, init='k-means++', max_iter=300,
       n_clusters=3, n_init=10, n_jobs=None, precompute_distances='auto',
       random_state=None, tol=0.0001, verbose=0)
```

[43]:
```
import random
random.seed(9008)
X_sample = np.array(random.sample(X_scaled.tolist(),15))
#type(X_scaled)
```

[44]:
```
rfmTable["cluster_new"] = clusters.labels_
```

[45]:
```
rfmTable
type(X_scaled)
```

[45]:
```
numpy.ndarray
```

[46]:
```
rfmTable.groupby('cluster_new' ).mean()
```

[46]:

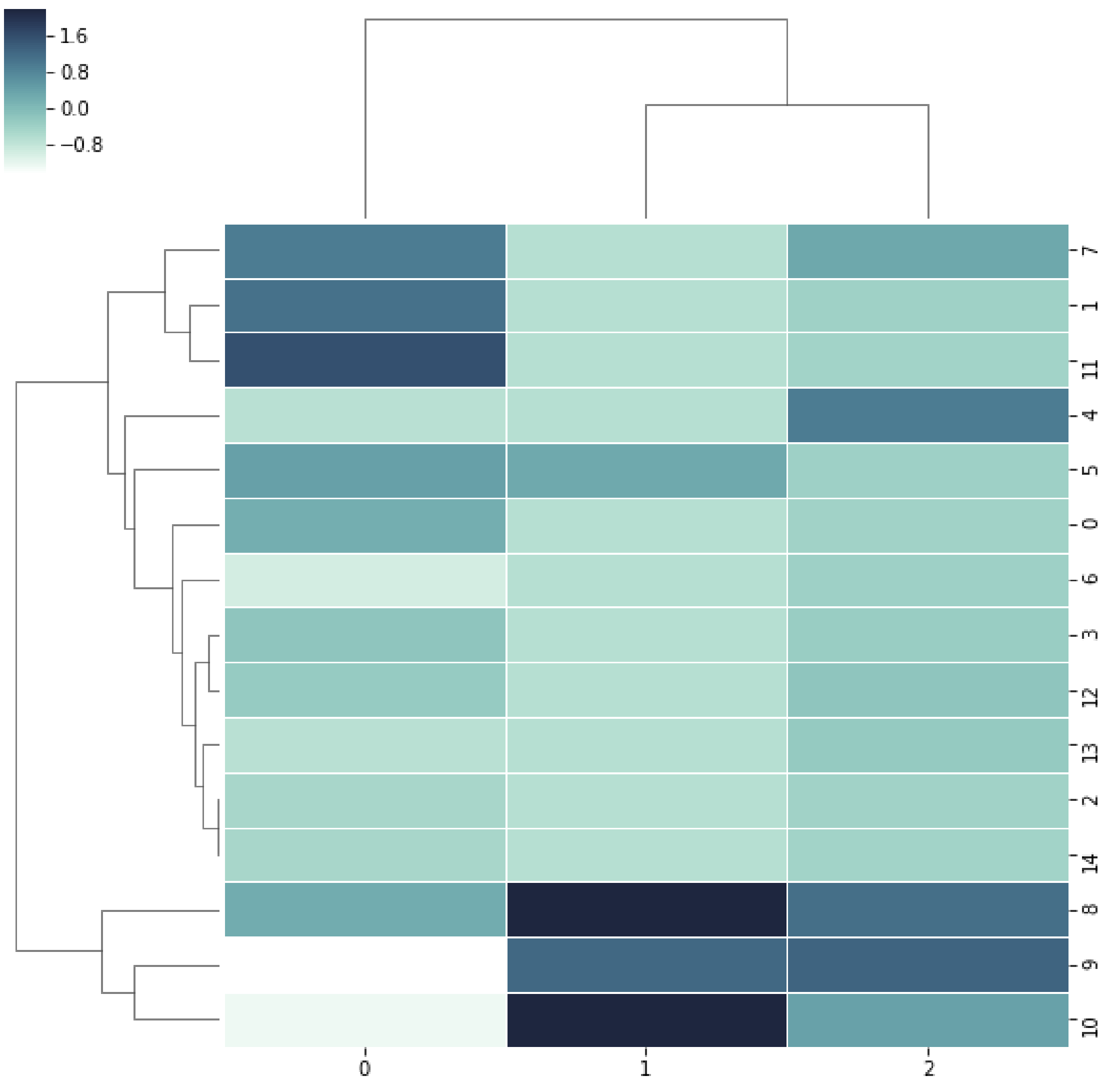
```
             recency  frequency  monetary_value
cluster_new
```

0	43.859127	1.474206	914.843036
1	62.393548	3.619355	7221.479806
2	134.382166	1.375796	857.864480

[47]:
```
rfmTable.drop( 'cluster_new', axis = 1, inplace = True )
```

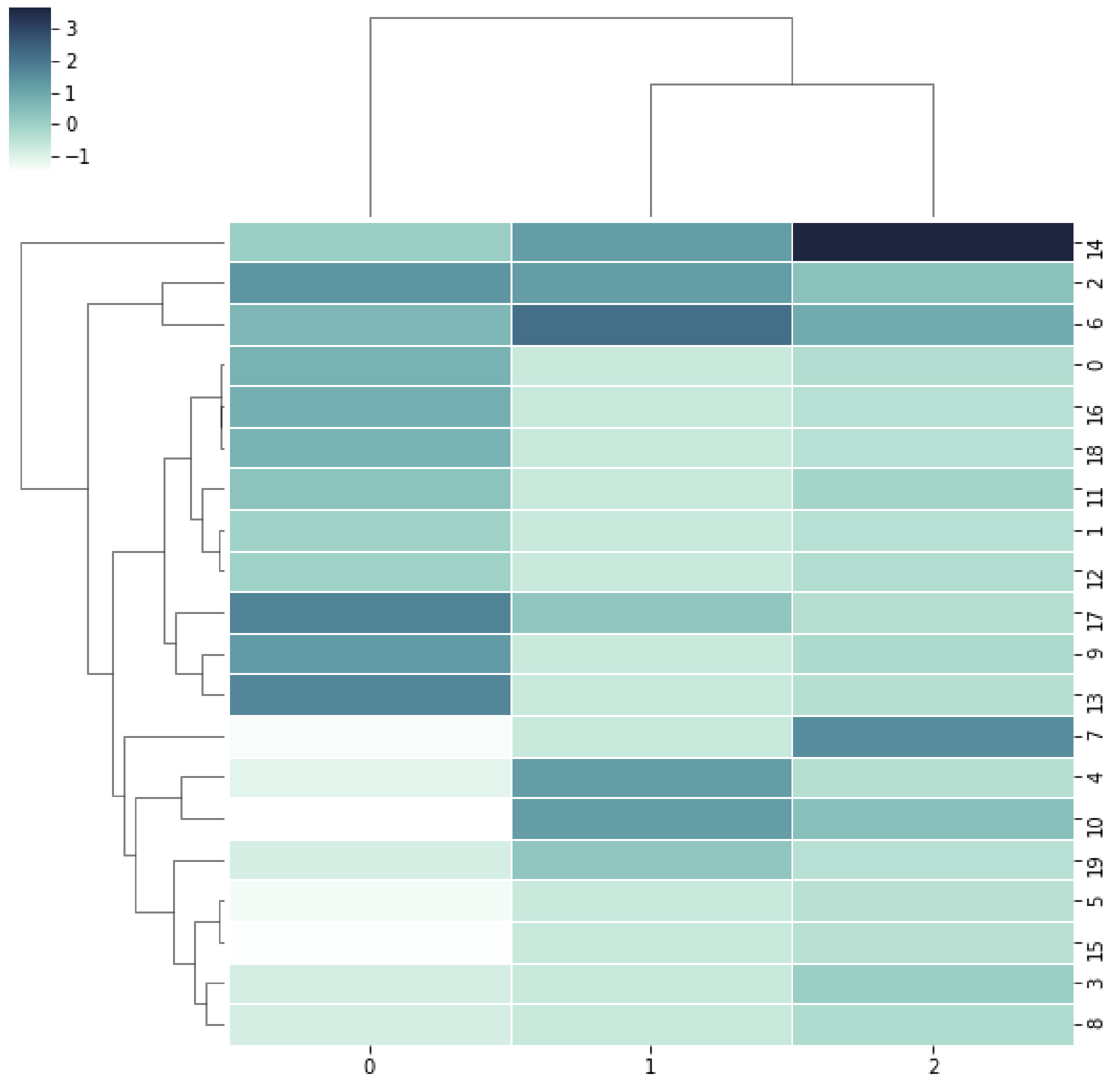
[ ]:
```
#Dendogram built with random samples from X_scaled
```

[48]:
```
cmap = sn.cubehelix_palette(as_cmap=True, rot=-.3, light=1)
g = sn.clustermap(X_sample, cmap=cmap, linewidths=.5)
```



[49]:
```
#Lets take one more sample to validate dendogram
random.seed(9005)
```

```
X_sample = np.array(random.sample(X_scaled.tolist(),20))
cmap = sn.cubehelix_palette(as_cmap=True, rot=-.3, light=1)
g = sn.clustermap(X_sample, cmap=cmap, linewidths=.5)
```



The Dendogram shows 1-3 distinct clusters

A random sample of 15-30 data points is used to build our dendogram

Elbow Method is used to determine the cluster segmentation

[50]:
```
cluster_range = range( 1, 10 )
cluster_errors = []

for num_clusters in cluster_range:
  clusters = KMeans( num_clusters )
```

```
    clusters.fit( X_scaled )
    cluster_errors.append( clusters.inertia_ )
```

[51]:
```
clusters_df = pd.DataFrame( { "num_clusters":cluster_range, "cluster_errors":
 ↪cluster_errors } )
```

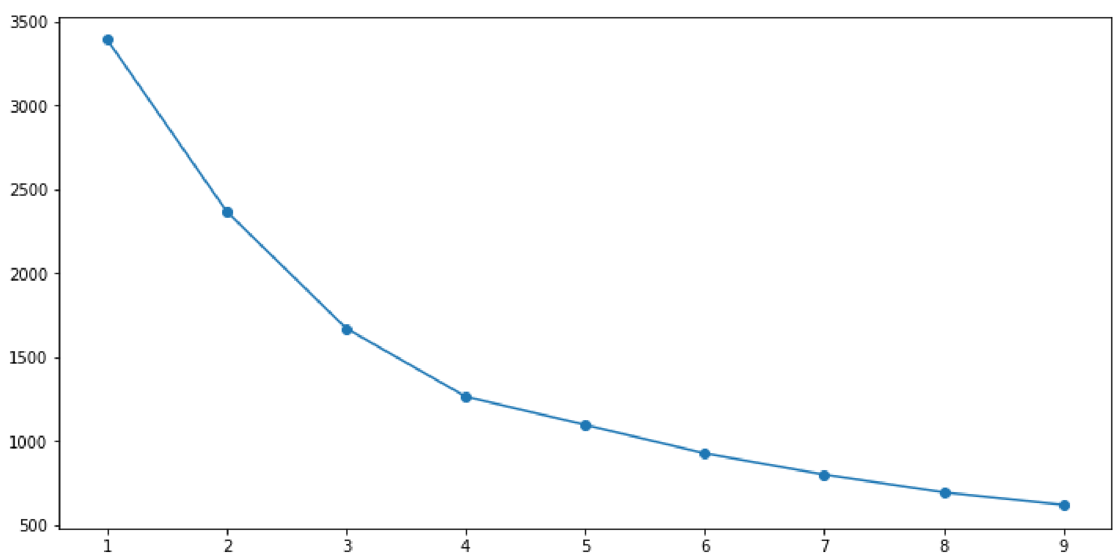
[53]:
```
clusters_df.head(10)
```

[53]:

	num_clusters	cluster_errors
0	1	3390.000000
1	2	2366.214582
2	3	1669.574012
3	4	1263.247749
4	5	1095.070323
5	6	925.602554
6	7	798.419375
7	8	692.878998
8	9	618.731232

[54]:
```
import matplotlib.pyplot as plt
plt.figure(figsize=(12,6))
plt.plot( clusters_df.num_clusters, clusters_df.cluster_errors, marker = "o" )
```

[54]: [<matplotlib.lines.Line2D at 0x24175d7e588>]



[55]:
```
clusters = KMeans(3)  # 3 clusters
clusters.fit( X_scaled )
rfmTable["cluster_label"] = clusters.labels_
```

[56]:
```
rfmTable.groupby('cluster_label').mean()
```

[56]:

	recency	frequency	monetary_value
cluster_label			
0	61.560000	3.633333	7399.300400
1	43.830040	1.480237	920.455553
2	134.299578	1.388186	862.487046

[66]:
```
rfmTable_0 = rfmTable[rfmTable.cluster_label == 0]
```

All the customers with high recency and low frequency and low monetary value are segmented in this cluster

These are the least profitable customers for the company computed below

[67]:
```
rfmTable_0.head(10)
```

[67]:

	recency	frequency	monetary_value	cluster_label
Customer ID				
899	154	4	409.62	0
2270	144	3	3892.79	0
2795	155	3	3234.40	0
2302	139	3	2831.74	0
1020	115	3	2732.72	0
2486	144	3	2482.25	0
1636	167	3	2253.67	0
2290	146	3	2013.60	0
2202	150	3	1873.62	0
247	101	3	1682.91	0

[70]:
```
rfmTable_1 = rfmTable[rfmTable.cluster_label == 1]
rfmTable_1.head(10)
```

[70]:

	recency	frequency	monetary_value	cluster_label
Customer ID				
1502	1	3	2882.33	1
3069	29	3	2042.90	1
266	43	3	1993.68	1
445	7	3	1935.48	1
3154	33	3	1851.52	1
3176	5	3	1830.05	1
1940	72	3	1787.29	1
1026	32	3	1664.68	1
269	25	3	1457.32	1
1062	31	3	1288.68	1

Each customer is assigned with the cluster label

This cluster has customers that are potential customers with decent frequency and monetary value

Company should work towards them to convert them to most profitable customers

[71]:
```
rfmTable_2 = rfmTable[rfmTable.cluster_label == 2]
rfmTable_2.head(10)
```

[71]:

Customer ID	recency	frequency	monetary_value	cluster_label
699	0	9	8039.29	2
2882	0	8	12748.90	2
3079	19	7	16736.71	2
2491	65	7	15513.56	2
1193	2	7	11514.94	2
3151	27	7	5626.41	2
1129	17	6	12840.69	2
3133	118	6	3733.30	2
2618	99	6	2394.79	2
693	56	5	14883.71	2

[72]:
```
rfmTable_0.mean()
```

[72]:
```
recency           133.232990
frequency           1.381443
monetary_value    847.686247
cluster_label       0.000000
dtype: float64
```

[73]:
```
rfmTable_1.mean()
```

[73]:
```
recency            42.864646
frequency           1.488889
monetary_value    936.245535
cluster_label       1.000000
dtype: float64
```

[74]:
```
rfmTable_2.mean()
```

[74]:
```
recency             61.560000
frequency            3.633333
monetary_value    7399.300400
cluster_label        2.000000
dtype: float64
```

[75]:
```
clusters = KMeans(3)  # 5 clusters
clusters.fit( X_scaled )
```

[75]:
```
KMeans(algorithm='auto', copy_x=True, init='k-means++', max_iter=300,
       n_clusters=3, n_init=10, n_jobs=None, precompute_distances='auto',
       random_state=None, tol=0.0001, verbose=0)
```

[76]:
```
rfmTable.head(10)
```

[76]:

	recency	frequency	monetary_value	cluster_label
Customer ID				
699	0	9	8039.29	2
2882	0	8	12748.90	2
3079	19	7	16736.71	2
2491	65	7	15513.56	2
1193	2	7	11514.94	2
3151	27	7	5626.41	2
1129	17	6	12840.69	2
3133	118	6	3733.30	2
2618	99	6	2394.79	2
693	56	5	14883.71	2

Each customer is assigned with the cluster label

All the customers with low recency and high frequency and and monetary value are segmented in this Cluster

These are the most profitable and highly valued customers company should look at.

[77]:
```
rfmTable.groupby('cluster_label').mean()
```

[77]:

	recency	frequency	monetary_value
cluster_label			
0	133.232990	1.381443	847.686247
1	42.864646	1.488889	936.245535
2	61.560000	3.633333	7399.300400

[ ]:
```
```