part2-customer-segmentation

December 13, 2019

1 Part 2: Customer Segmentation

Now that we what are major metrics and how to track by using Python we can focus on customers and segment them with **RFM Clustering.**

In this notebook we will implement one popular segmentation to our business: **RFM**. We need to calculate Recency, Frequency and Monetary Value (we will call it Revenue from now on) and apply unsupervised machine learning to identify different groups (clusters) for each.

RFM stands for *Recency - Frequency - Monetary Value*. Theoretically we will have segments like below: * Low Value: Customers who are less active than others, not very frequent buyer/visitor and generates very low - zero - maybe negative revenue. * Mid Value: In the middle of everything. Often using our platform (but not as much as our High Values), fairly frequent and generates moderate revenue. * High Value: The group we don't want to lose. High Revenue, Frequency and low Inactivity.

Segmentation is important be because we can't treat every customer the same way with the same content, same channel, same importance. They will find another option which understands them better.

Customers who use your platform have different needs and they have their own different profile, so you should adapt your actions depending on that.

You can do many different segmentations according to what you are trying to achieve. If you want to increase retention rate, you can do a segmentation based on churn probability and take actions. But there are other very common and useful segmentation methods as well.

1.1 Load the dataset

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 541909 entries, 0 to 541908
Data columns (total 8 columns):
InvoiceNo      541909 non-null object
StockCode      541909 non-null object
Description    540455 non-null object
Quantity       541909 non-null int64
InvoiceDate    541909 non-null datetime64[ns]
UnitPrice      541909 non-null float64
CustomerID     406829 non-null float64
Country        541909 non-null object
```

```
dtypes: datetime64[ns](1), float64(2), int64(1), object(4)
memory usage: 33.1+ MB
```

1.2 Recency

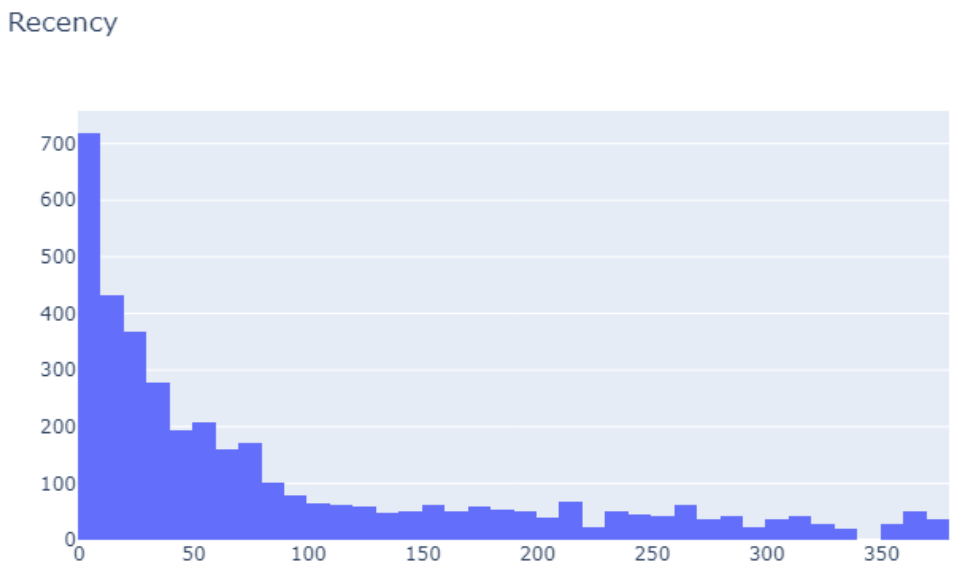
To calculate recency, we need to find out most recent purchase date of each customer and see how many days they are inactive for. After having no. of inactive days for each customer, we will apply K-means* clustering to assign customers a recency score.

```
   CustomerID  Recency
0     17850.0      301
1     13047.0       31
2     13748.0       95
3     15100.0      329
4     15291.0       25
```

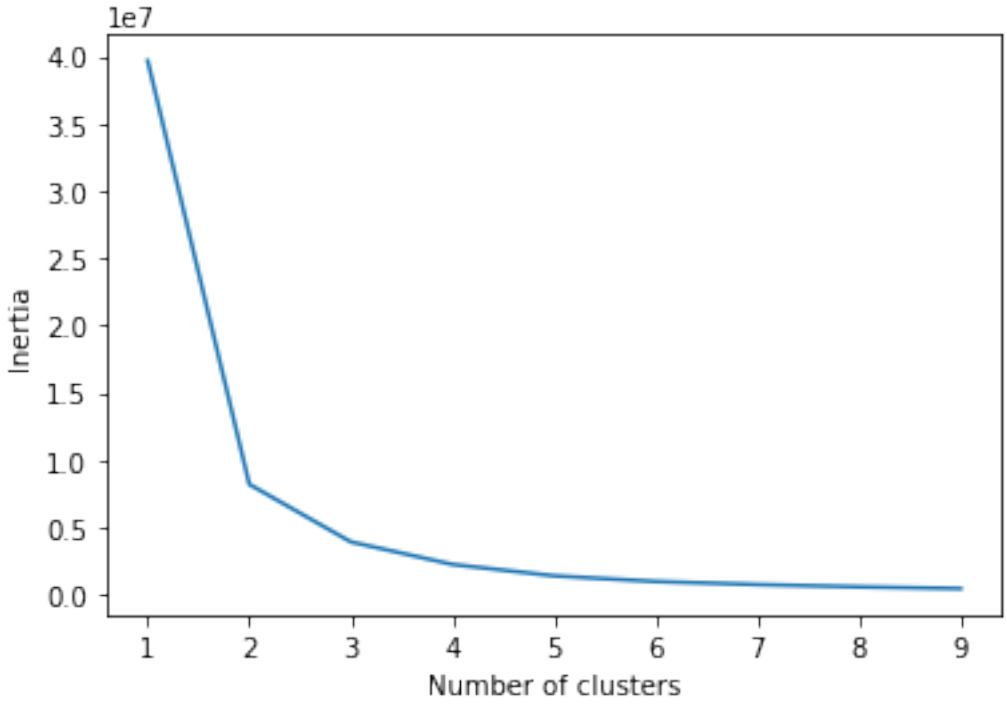
To get a snapshot about how recency looks like, let's look at mean, min, max, count and percentiles of our data.

```
count    3950.000000
mean       90.778481
std       100.230349
min         0.000000
25%        16.000000
50%        49.000000
75%       142.000000
max       373.000000
Name: Recency, dtype: float64
```

We see that even though the average is 90 day recency, median is 49. Histogram shows us the distribution of recency across our customers.



Next, we are going to apply K-means clustering to assign a recency score. To find out how many cluster is optimal, we will apply Elbow Method. Elbow Method simply tells the optimal cluster number for optimal inertia.



Here it looks like 3 is the optimal one. Based on business requirements, we can go ahead with less or more clusters. We will be selecting 4 for this example:

```
                 count        mean        std    min     25%    50%     75%  \
RecencyCluster
0                1950.0   17.488205  13.237058    0.0    6.00   16.0   28.00
1                 478.0  304.393305  41.183489  245.0  266.25  300.0  336.00
2                 568.0  184.625000  31.753602  132.0  156.75  184.0  211.25
3                 954.0   77.679245  22.850898   48.0   59.00   72.5   93.00

                  max
RecencyCluster
0                47.0
1               373.0
2               244.0
3               131.0
```

We have calculated clusters and assigned them to each Customer in our dataframe tx_user. We can see how our recency clusters have different characteristics. The customers in Cluster 1 are very recent compared to Cluster 2 which are very inactive.

K-means assigns clusters as numbers but not in an ordered way. We can't say cluster 0 is the worst and cluster 4 is the best. `order_cluster()` method does this and our new dataframe looks much neater:

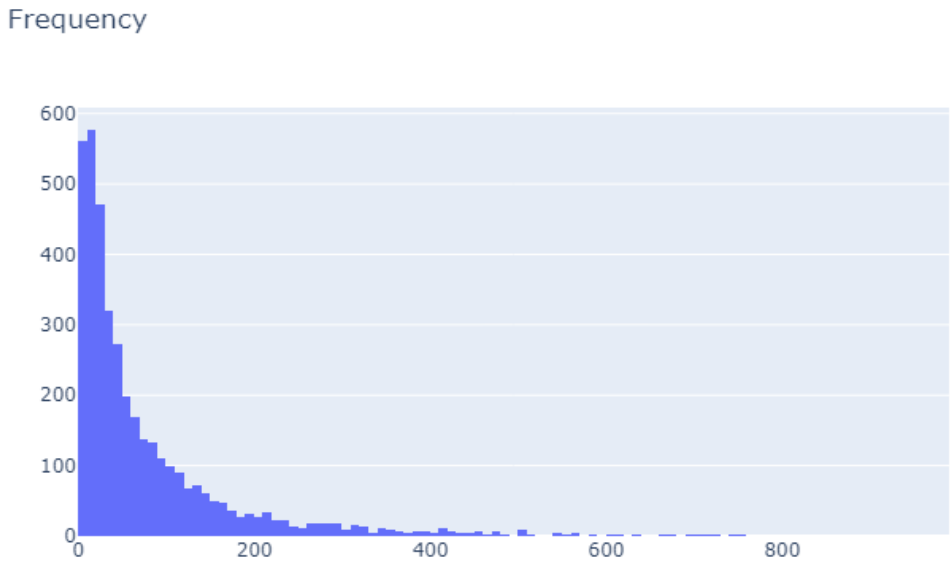
```
                 count        mean        std    min     25%    50%     75%  \
RecencyCluster
0                 478.0  304.393305  41.183489  245.0  266.25  300.0  336.00
1                 568.0  184.625000  31.753602  132.0  156.75  184.0  211.25
2                 954.0   77.679245  22.850898   48.0   59.00   72.5   93.00
3                1950.0   17.488205  13.237058    0.0    6.00   16.0   28.00

                  max
RecencyCluster
0               373.0
1               244.0
2               131.0
3                47.0
```

Great! Cluser 3 covers most recent customers whereas cluster 0 has the most inactive ones.

1.3 Frequency

To create frequency clusters, we need to find total number orders for each customer. First calculate this and see how frequency look like in our customer database:

	CustomerID	Recency	RecencyCluster	Frequency
0	17850.0	301	0	312
1	15100.0	329	0	6
2	18074.0	373	0	13
3	16250.0	260	0	24
4	13747.0	373	0	1



Apply the same logic to have frequency clusters and assign cluster number to each customer:

FrequencyCluster	count	mean	std	min	25%	50%	\
0	3496.0	49.525744	44.954212	1.0	15.0	33.0	
1	429.0	331.221445	133.856510	191.0	228.0	287.0	
2	22.0	1313.136364	505.934524	872.0	988.5	1140.0	
3	3.0	5917.666667	1805.062418	4642.0	4885.0	5128.0	

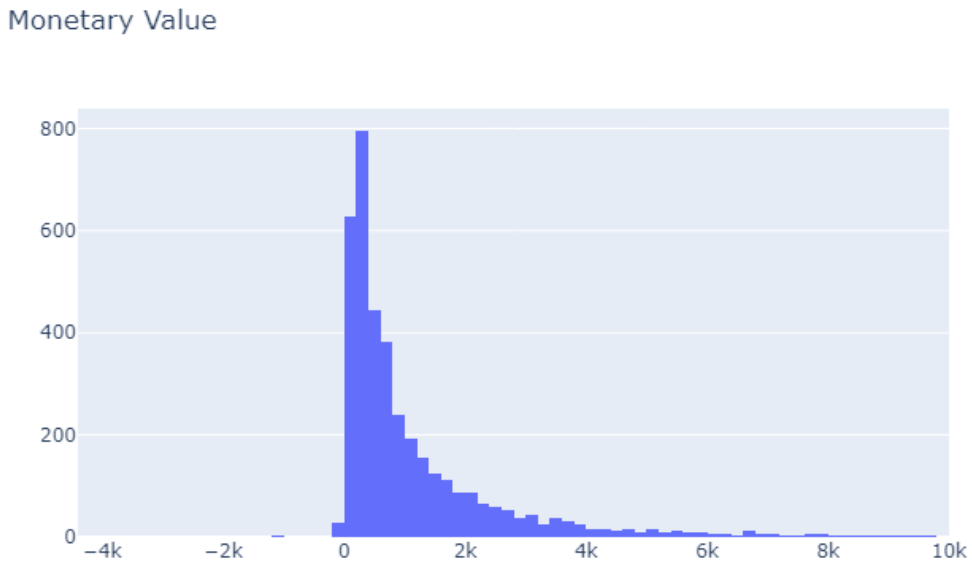
FrequencyCluster	75%	max
0	73.0	190.0
1	399.0	803.0
2	1452.0	2782.0
3	6555.5	7983.0

We follow same notation as for recency clusters - high frequency number indicates better customers.

1.4 Revenue

Let's see how our customer database looks like when we cluster them based on revenue. We will calculate revenue for each customer, plot a histogram and apply the same clustering method.

	CustomerID	Recency	RecencyCluster	Frequency	FrequencyCluster	Revenue
0	17850.0	301	0	312	1	5288.63
1	15808.0	305	0	210	1	3724.77
2	13047.0	31	3	196	1	3079.10
3	14688.0	7	3	359	1	5107.38
4	16029.0	38	3	274	1	50992.61



Note that we have some customers with negative revenue as well. Let's continue and apply k-means clustering:

	count	mean	std	min	25%	\
RevenueCluster						
0	3687.0	907.254414	921.910820	-4287.63	263.115	
1	234.0	7760.699530	3637.173671	4330.67	5161.485	
2	27.0	43070.445185	15939.249588	25748.35	28865.490	
3	2.0	221960.330000	48759.481478	187482.17	204721.250	

	50%	75%	max
RevenueCluster			
0	572.56	1258.220	4314.72

```
1                6549.38    9142.305   21535.90
2               36351.42   53489.790   88125.38
3              221960.33  239199.410  256438.49
```

We follow same notation as for last two clusters - high revenue indicates better customers. We also note cluster 3 identifies ultra high value customers.

1.5 Overall Score

Up until now we have calculated scores (cluster numbers) for recency, frequency & revenue. Let's create an overall score out of them:

```
                 Recency    Frequency        Revenue
OverallScore
0             304.584388    21.995781     303.339705
1             185.362989    32.596085     498.087546
2              78.991304    46.963043     868.082991
3              20.689610    68.419590    1091.416414
4              14.892617   271.755034    3607.097114
5               9.662162   373.290541    9136.946014
6               7.740741   876.037037   22777.914815
7               1.857143  1272.714286  103954.025714
8               1.333333  5917.666667   42177.930000

OverallScore
0     474
1     562
2     920
3    1511
4     298
5     148
6      27
7       7
8       3
Name: Recency, dtype: int64
```

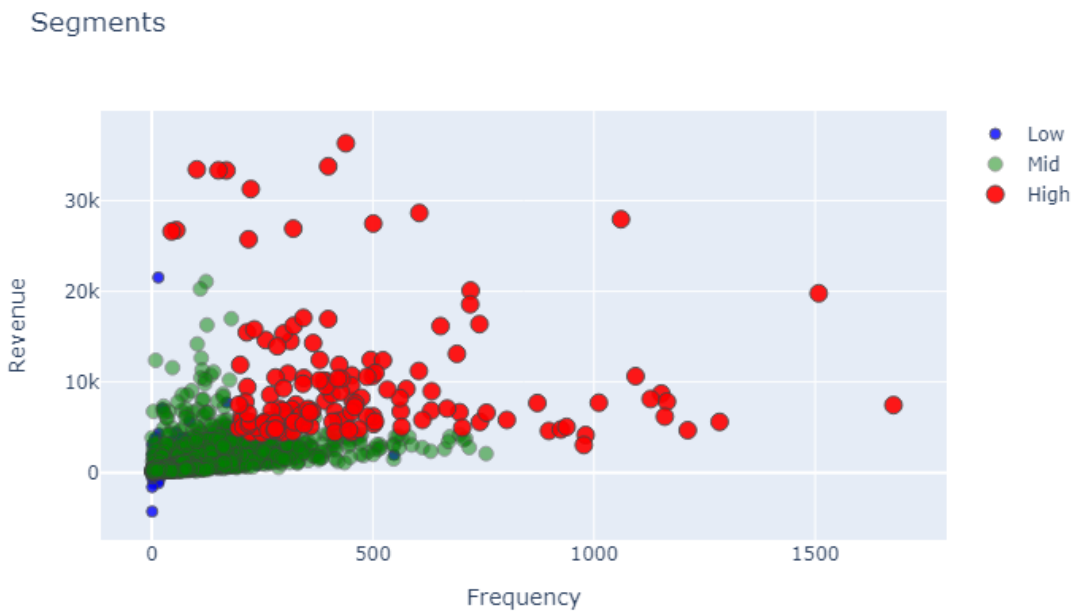
The scoring above clearly shows us that customers with score 8 is our best customers whereas 0 is the worst.

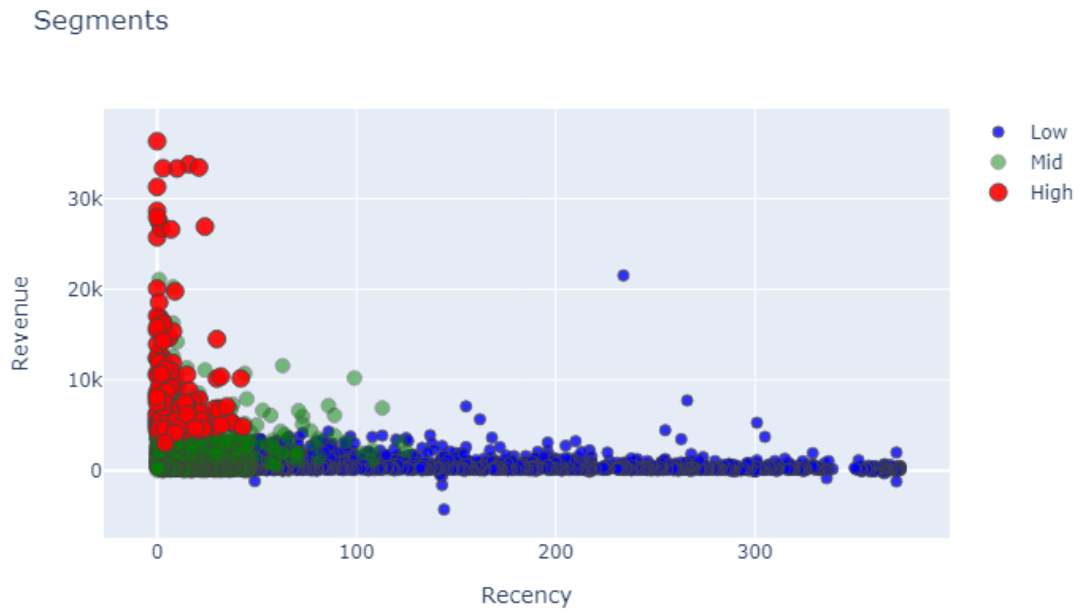
To keep things simple, we will name these scores: * 0 to 2: Low Value * 3 to 4: Mid Value * 5+: High Value

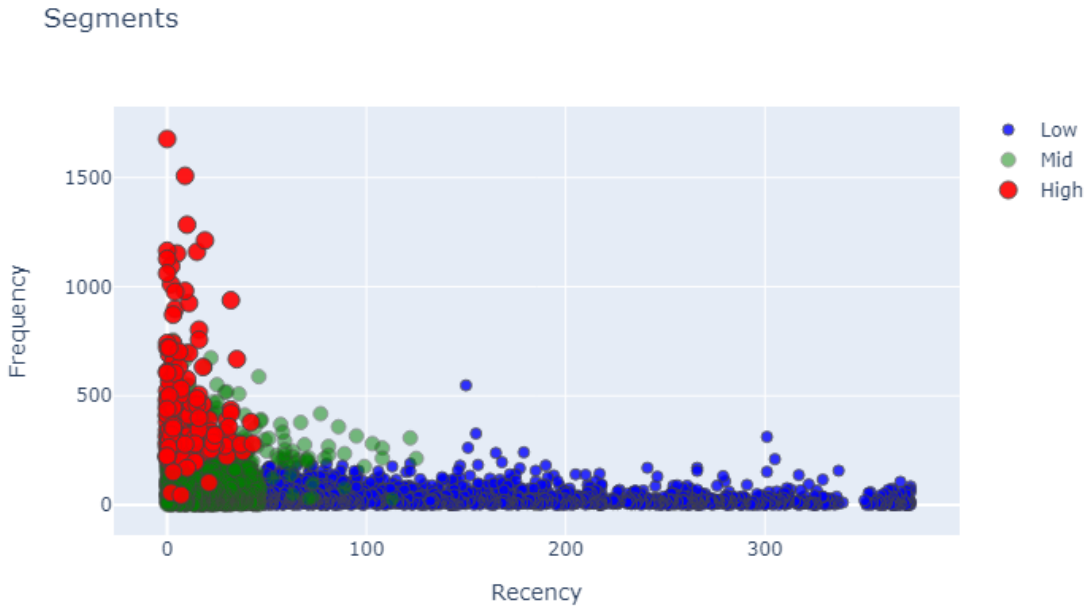
```
   CustomerID  Recency  RecencyCluster  Frequency  FrequencyCluster   Revenue  \
0     17850.0      301               0        312                 1   5288.63
1     14688.0        7               3        359                 1   5107.38
2     13767.0        1               3        399                 1  16945.71
3     15513.0       30               3        314                 1  14520.08
4     14849.0       21               3        392                 1   7904.28
```

	RevenueCluster	OverallScore	Segment
0	1	2	Low-Value
1	1	5	High-Value
2	1	5	High-Value
3	1	5	High-Value
4	1	5	High-Value

Let's see how our segments are distributed on a scatter plot:







We can see how the segments are clearly differentiated from each other in terms of RFM.

We can start taking actions with this segmentation. The main strategies are quite clear: * High Value: Improve Retention * Mid Value: Improve Retention + Increase Frequency * Low Value: Increase Frequency

Note: What we do here can be easily achieved by using quantiles or simple binning (or Jenks natural breaks optimization to make groups more accurate).

2 Summary

In this notebook we have segmented our customers into RFM clusters and identified main strategies for treating them.

In next part we will caclculate and predict lifetime value of our customers.