E-COMMERCE CUSTOMER SEGMENTATION

By: Malvin Hariyanto Kurniawan - FSDA RevoU
Madrid Section

OUTLINE

Topics Covered

- Business Problem
- Data Source
- Insights
- Data Preparation
- Clustering Analysis
- Recommendation

Business Problem

An e-commerce startup based in Portugal that recently opened an online website to sell their product struggling to handle the marketing target since the traffic growth too fast. To minimize the loss in marketing budget, the marketing team needs to increase the marketing conversion rate by doing targeted marketing using customer segmentation.

Data Source

Go Back to Home Page

Data Source

Brief explanation of Data Frame used for analysis

Cleaned Data Frame Preview

	order_id	customer_unique_id	order_purchase_timestamp	payment_value
0	e481f51cbdc54678b7cc49136f2d6af7	7c396fd4830fd04220f754e42b4e5bff	2017-10-02 10:56:33	18.12
1	53cdb2fc8bc7dce0b6741e2150273451	af07308b275d755c9edb36a90c618231	2018-07-24 20:41:37	141.46
2	47770eb9100c2d0c44946d9cf07ec65d	3a653a41f6f9fc3d2a113cf8398680e8	2018-08-08 08:38:49	179.12
3	949d5b44dbf5de918fe9c16f97b45f8a	7c142cf63193a1473d2e66489a9ae977	2017-11-18 19:28:06	72.20
4	ad21c59c0840e6cb83a9ceb5573f8159	72632f0f9dd73dfee390c9b22eb56dd6	2018-02-13 21:18:39	28.62

Cleaned Data Frame Properties

```
Data columns (total 4 columns):
 #   Column                    Non-Null Count  Dtype
---  ------                    --------------  -----
 0   order_id                  91659 non-null  object
 1   customer_unique_id        91659 non-null  object
 2   order_purchase_timestamp  91659 non-null  datetime64[ns]
 3   payment_value             91659 non-null  float64
```

- order_id : unique identification for each transaction (not duplicated)
- customer_unique_id : unique identification for each customer (duplicated)
- order_purchase_timestamp : Date time while transaction has been made
- payment_value : amount of paid for each transaction.

Insights

Go Back to Home Page

Insights

From the clustering analysis, customers can be classified to 3 clusters.

- Customer with small amount of spending but do made the purchase most recent have highest population among the other 2, with around 45% of total customers.
- Customer with big amount of spending and have not made the purchase for quite a long time since their last transaction have the least population or around 25% of total customers.
- The rest is customers with moderate amount of spending and have not made the purchase for very long time and can categorized as the lost customer, with population around 29% of total customers.
- Most of the customers are 1 timer buyers. Only few customer made the purchase more than 1 time.

Data Preparation 

Go Back to Home Page

Before Prepare the Data

By looking at the Data Frame, the closest analysis to be done is RFM analysis.

What is RFM Analysis

Recency, frequency, monetary value is a marketing analysis tool used to identify a company's or an organization's best customers by measuring and analyzing spending habits.

What is Recency, Frequency, Monetary

1. Recency: How recently a customer has made a purchase
2. Frequency: How often a customer makes a purchase
3. Monetary Value: How much money a customer spends on purchases

The concept of recency, frequency, monetary value (RFM) is thought to date from an article by Jan Roelf Bult and Tom Wansbeek, "Optimal Selection for Direct Mail," published in a 1995 issue of Marketing Science.RFM analysis often supports the marketing adage that "80% of business comes from 20% of the customers."

(source: https://www.investopedia.com/terms/r/rfm-recency-frequency-monetary-value.asp)

Data Preparation

Handling the Outliers

Any outliers from numerical columns should be removed.

Scaling the Values

Because the numeric column have different scale, we need to scale it first, so all numeric value have difference importance.

Ready for Modelling

The data frame is ready for modelling

Go Back to Home Page

Handling the Outliers

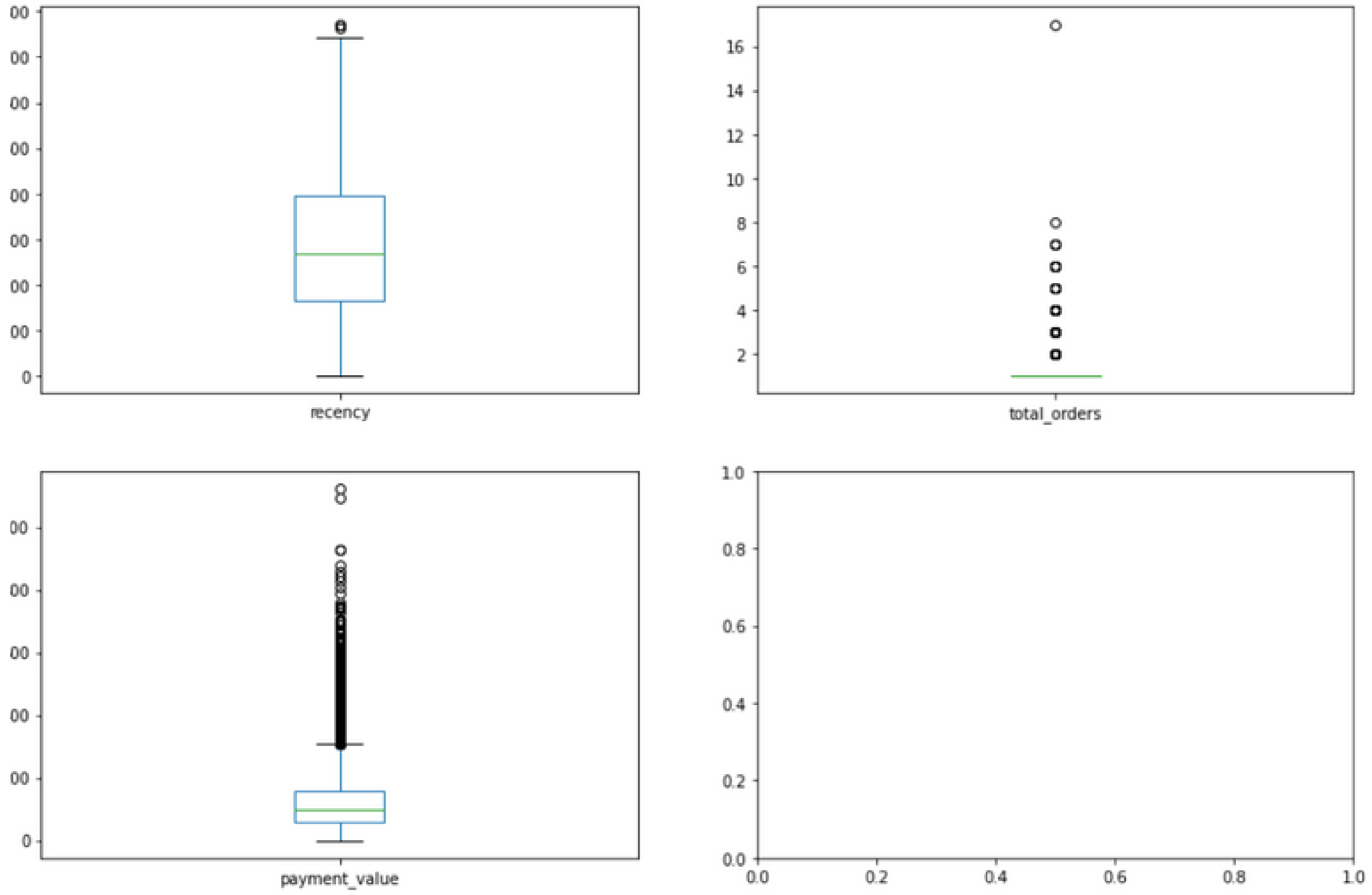
First we check for potential outliers, then delete the outliers from Data Frame.

Create Boxplot :

```
## Checking for outliers
numerical_col = ['recency','total_orders','payment_value']

fig,axes = plt.subplots(nrows=2, ncols = 2, figsize=(15,10))
for i, el in enumerate(numerical_col):
  a = cluster_join.boxplot(el,ax=axes.flatten()[i],grid=False)
plt.show()
```

Result :



Next, we will check the Outliers for each numerical columns

Handling the Outliers

First we check for potential outliers, then delete the outliers from Data Frame.

Define function:

```
cluster_outliers = cluster_join.copy()

# Define function for checking outliers
def check_outliers(data,col_name):
  q1 = data[col_name].quantile(0.25)
  q3 = data[col_name].quantile(0.75)
  iqr = q3 - q1
  c_min = q1 - 1.5*iqr
  c_max = q3 + 1.5*iqr
  print('Q1: ',q1)
  print('Q3: ',q3)
  print('IQR: ',iqr)
  print('Min: ',c_min)
  print('Max: ',c_max)
```

Apply function :

```
check_outliers(cluster_outliers,'recency')
```

Result :

```
Q1:  164.0
Q3:  397.0
IQR:  233.0
Min:  -185.5
Max:  746.5
```

Apply the function for each numerical column.

Handling the Outliers

First we check for potential outliers, then delete the outliers from Data Frame.

Delete outliers :

```
print('Total rows before remove outliers ',len(cluster_join))

# Remove the outliers
cluster_outliers = cluster_outliers[(cluster_outliers['recency'] >= -185.5) & (cluster_outliers['recency'] < 746.5)]
print('Total rows after remove outliers ',len(cluster_outliers))
```

Result :

```
Total rows before remove outliers  88662
Total rows after remove outliers  88659
```

Apply the step for each numerical column, then check the final result with boxplot.

Exception: For total_orders column, since nearly all values are 1, rows with total_orders > 6 will be removed.

Handling the Outliers

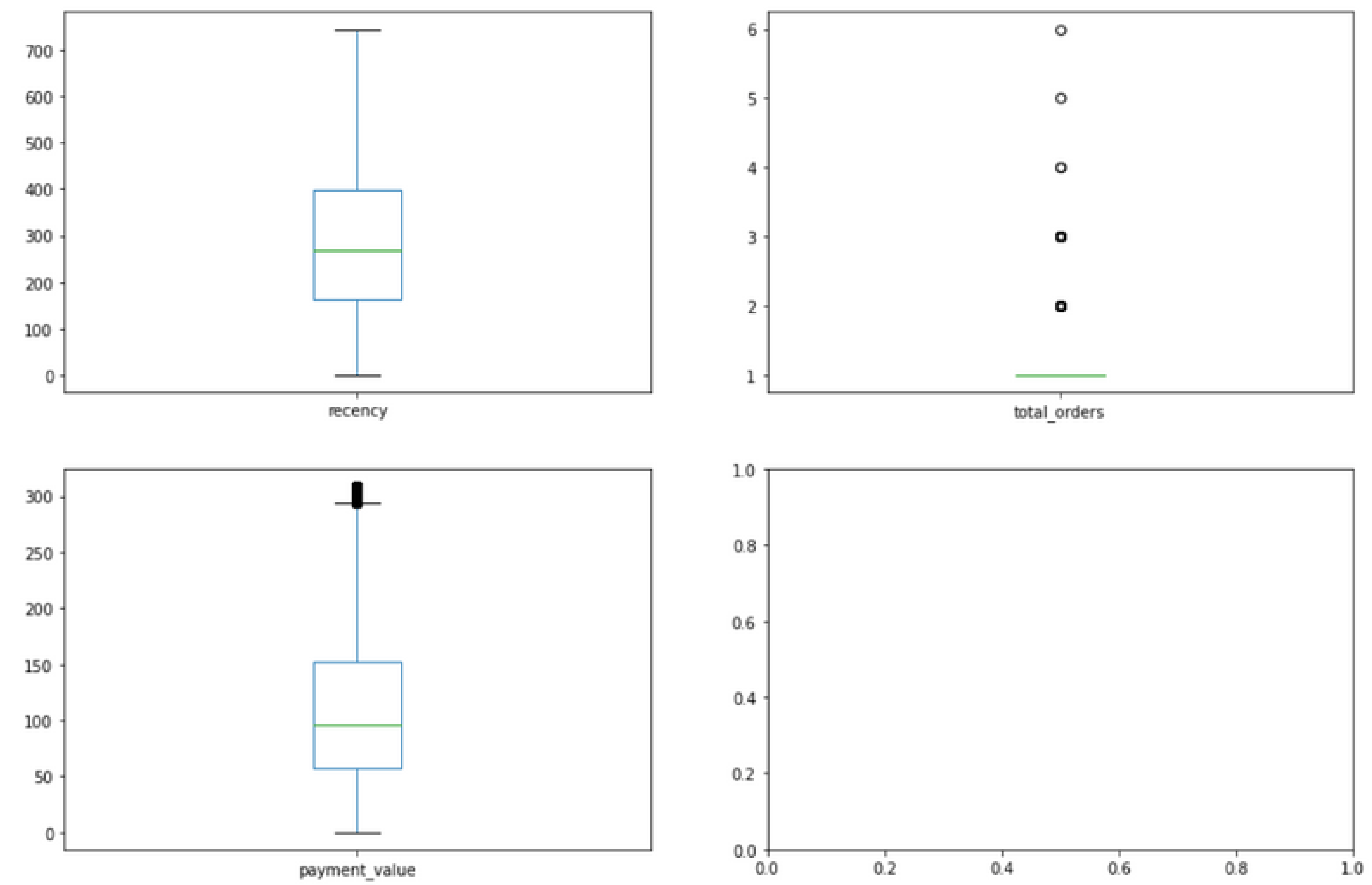
First we check for potential outliers, then delete the outliers from Data Frame.

Create Boxplot :

```
numerical_col = ['recency','total_orders','payment_value']

fig,axes = plt.subplots(nrows=2, ncols = 2, figsize=(15,10))
for i, el in enumerate(numerical_col):
  a = cluster_outliers.boxplot(el,ax=axes.flatten()[i],grid=False)
plt.show()
```

Result after remove the outliers:



Scaling the Values

By using scikit-learn library, all value in numeric column will be scaled.

Import libraries :

```
from sklearn import cluster
from sklearn.preprocessing import MinMaxScaler
```

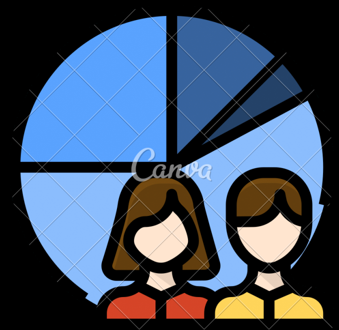
Create Data Frame with Scaled Values :

```
cluster_scale = cluster_clean.copy()
scaler = MinMaxScaler()
cluster_scale[numerical_col] = scaler.fit_transform(cluster_scale[numerical_col])
cluster_scale.head()
```

Result :

	customer_unique_id	recency	total_orders	payment_value
0	0000366f3b9a7992bf8c76cfdf3221e2	0.215054	0.0	0.459265
1	0000b849f77a49e4a4ce2b2a4ca5be3f	0.219086	0.0	0.087975
2	0000f46a3911fa3c0805444483337064	0.786290	0.0	0.279042
3	0000f6ccb0745a6a4b88665a16c9f078	0.495968	0.0	0.141156
4	0004aac84e0df4da2b147fca70cf8255	0.451613	0.0	0.637255

This Data Frame will be used for clustering analysis.

Clustering Analysis 

Go Back to Home Page

Clustering Analysis

Determine Cluster Number 1

by using Elbow method

Determine Cluster Number 2

by using Silhouette analysis

Clustering Data

After take conclusion from two methods to determine cluster number, all records will be clustered.

Cluster Characteristic

by evaluate the descriptive feature of each cluster to determine the name & character for each cluster.

Go Back to Home Page

Elbow Method

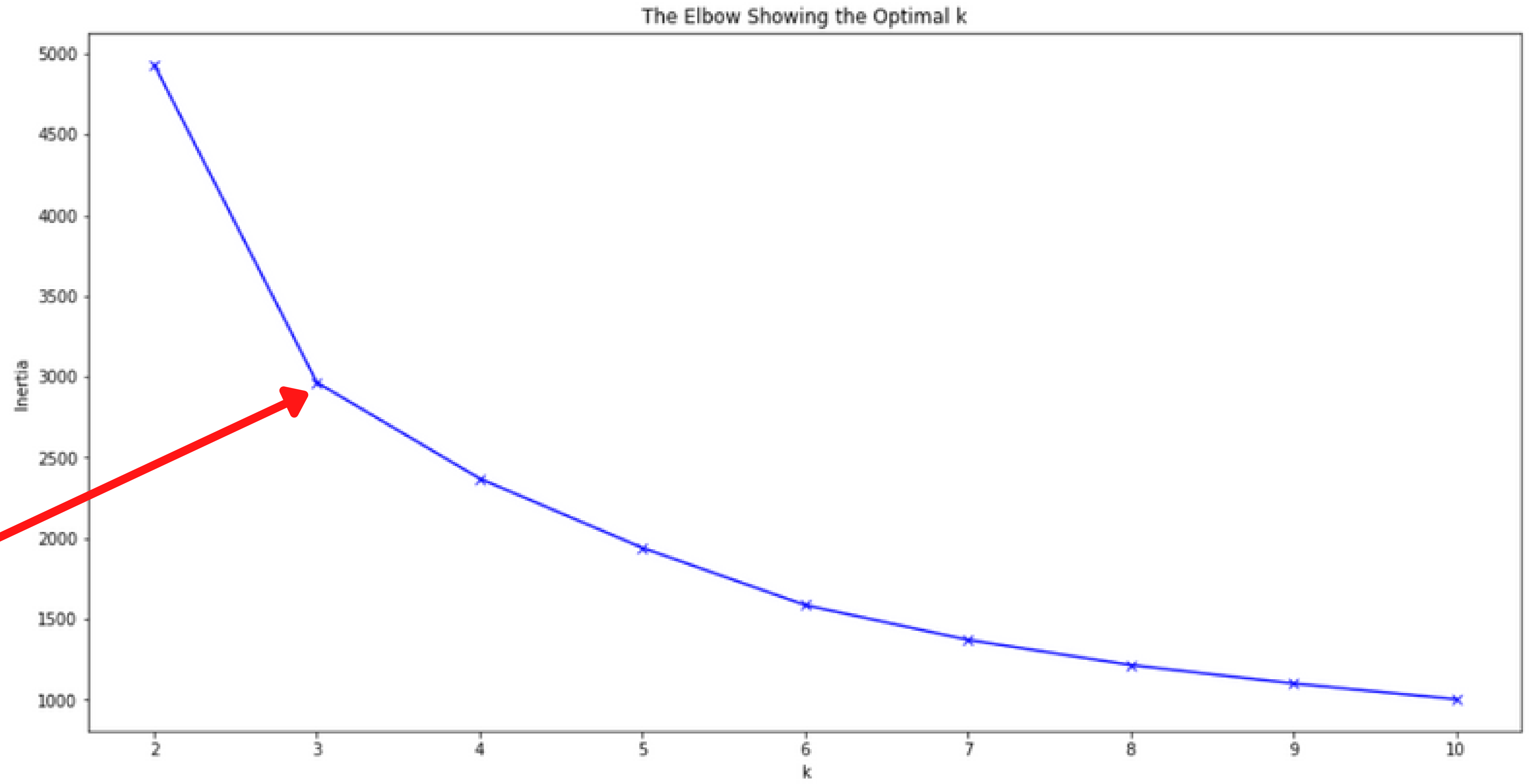
First, we determine cluster number by using Elbow method. The cluster number to check is from 2 to 10 which is range that make sense for business.

Create elbow plot :

```
distortions = []
K = range(2,11)
for k in K:
  kmeanModel = cluster.KMeans(n_clusters=k)
  kmeanModel.fit(cluster_check)
  distortions.append(kmeanModel.inertia_)

plt.figure(figsize=(16,8))
plt.plot(K, distortions,'bx-')
plt.xlabel('k')
plt.ylabel('Inertia')
plt.title('The Elbow Showing the Optimal k')
plt.show()
```

Result :



Optimal cluster number

Silhouette Analysis

Secondly, we determine cluster number by using Silhouette Analysis. The cluster number to check is from 2 to 10 which is range that make sense for business.

Import library :

```
from silhoutte import silhoutte_analysis
```

Run silhouette analysis :

```
silhoutte_analysis(cluster_check,list(range(2,11)))
```

Result :

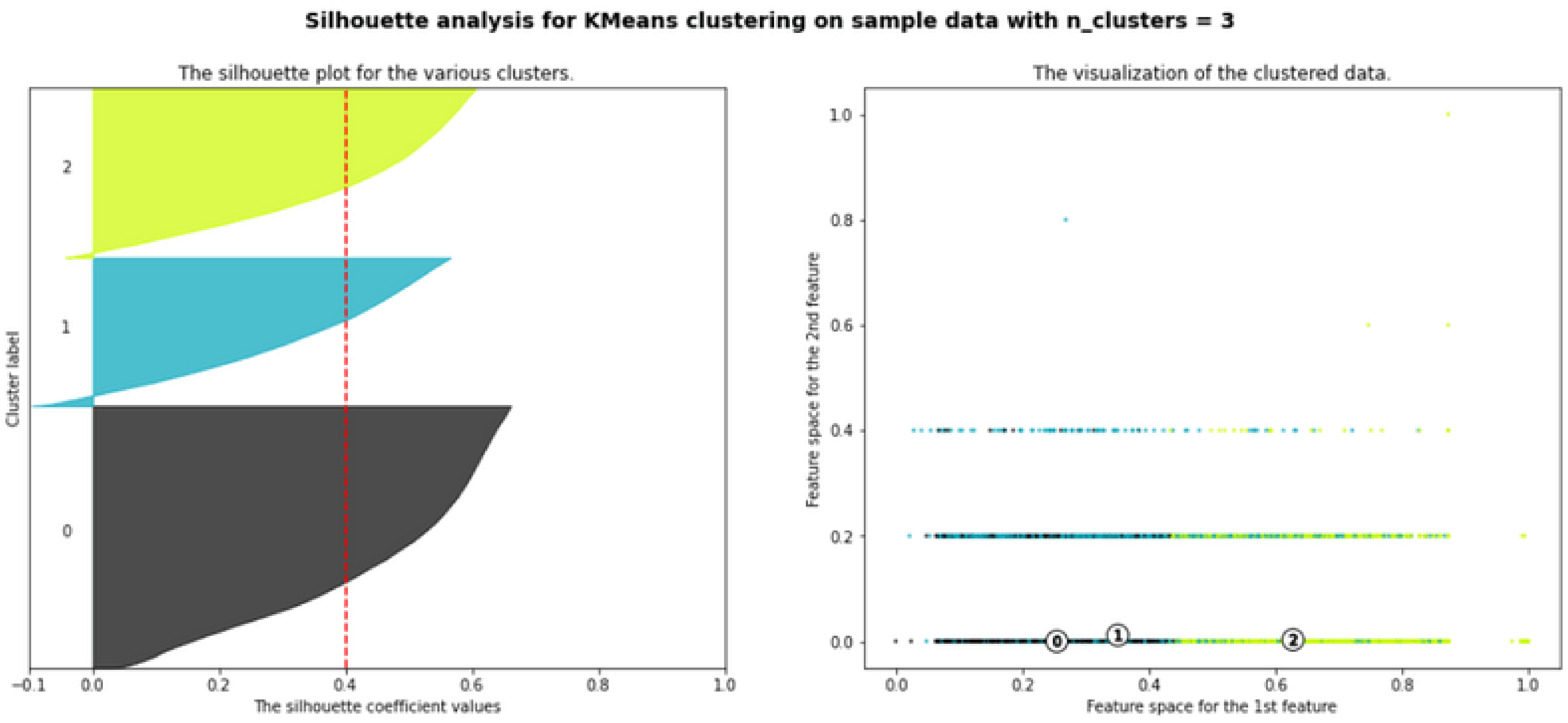
```
For n_clusters = 2 The average silhouette_score is : 0.37137875809963794
For n_clusters = 3 The average silhouette_score is : 0.39966313844034
For n_clusters = 4 The average silhouette_score is : 0.376883287591865З
For n_clusters = 5 The average silhouette_score is : 0.3431786118007398
For n_clusters = 6 The average silhouette_score is : 0.3496988837626782
For n_clusters = 7 The average silhouette_score is : 0.3547110983394691
For n_clusters = 8 The average silhouette_score is : 0.3500271403436149
For n_clusters = 9 The average silhouette_score is : 0.34968184138023956
For n_clusters = 10 The average silhouette_score is : 0.3357358413263723
```

The nearer silhouette _score to 1, the more optimal the cluster number. Cluster number = 3 is the most optimal.

Silhouette Analysis

Secondly, we determine cluster number by using Silhouette Analysis. The cluster number to check is from 2 to 10 which is range that make sense for business.

Silhouette Analysis Graph for n_cluster = 3 :



From both method, the optimal cluster number is 3.

Clustering Data

After determine cluster number, all data will be ready for cluster modeling.

Clustering Dataset :

```
cluster_model = cluster.KMeans(n_clusters=3, random_state = 2)
cluster_model.fit(cluster_check)
cluster_label = cluster_model.labels_
cluster_clean['cluster'] = cluster_label
cluster_clean
```

Data Frame Result :

	customer_unique_id	recency	total_orders	payment_value	cluster
0	0000366f3b9a7992bf8c76cfdf3221e2	160	1	141.90	0
1	0000b849f77a49e4a4ce2b2a4ca5be3f	163	1	27.19	0
2	0000f46a3911fa3c0805444483337064	585	1	86.22	2
3	0000f6ccb0745a6a4b88665a16c9f078	369	1	43.62	2
4	0004aac84e0df4da2b147fca70cf8255	336	1	196.89	1
...	...	...	...	...	...
88657	fffbf87b7a1a6fa8b03f081c5f51a201	293	1	167.32	1
88658	fffea47cd6d3cc0a88bd621562a9d061	310	1	84.58	0
88659	ffff371b4d645b6ecea244b27531430a	617	1	112.46	2
88660	ffff5962728ec6157033ef9805bacc48	168	1	133.69	0
88661	ffffd2657e2aad2907e67c3e9daecbeb	532	1	71.56	2

By now, all rows already assigned for their cluster.

Cluster Characteristic

Last but not least, the characteristic for each cluster need to be checked before naming each cluster. We can do this by evaluate descriptive feature (in this case median) for each metric.

Create descriptive analysis :

```
cluster_clean.groupby('cluster')['recency','total_orders','payment_value'].agg(['count','mean','median','max','min'])
```

Data Frame Result :

	recency					total_orders					payment_value				
	count	mean	median	max	min	count	mean	median	max	min	count	mean	median	max	min
cluster															
0	38942	188.566073	188.0	343	0	38942	1.011838	1.0	3	1	38942	74.395758	69.85	155.01	0.01
1	22119	259.861115	250.0	743	16	22119	1.058276	1.0	5	1	22119	202.815024	195.00	308.96	130.04
2	25143	466.039017	457.0	744	317	25143	1.016386	1.0	6	1	25143	85.476298	78.84	252.19	0.01

For this case, we will evaluate median value for each metric.

All clusters have similar mean and median of `total_orders` frequency score. Most of the customers are 1 time purchasers. Only few customers with repat purchase in this case.

Due to this condition, we will naming the clusters based on recency score and monetary score (payment_value).

Cluster Characteristic

Last but not least, the characteristic for each cluster need to be checked before naming each cluster. We can do this by evaluate descriptive feature (in this case median) for each metric.

Data Frame Result :

	recency					total_orders					payment_value				
	count	mean	median	max	min	count	mean	median	max	min	count	mean	median	max	min
cluster															
0	38942	188.566073	188.0	343	0	38942	1.011838	1.0	3	1	38942	74.395758	69.85	155.01	0.01
1	22119	259.861115	250.0	743	16	22119	1.058276	1.0	5	1	22119	202.815024	195.00	308.96	130.04
2	25143	466.039017	457.0	744	317	25143	1.016386	1.0	6	1	25143	85.476298	78.84	252.19	0.01

Naming Each Cluster:

By looking at the median value for both metrics, we can see the general characteristics for each cluster are:

- cluster 0: Low spend and recent purchase
- cluster 1: High spend and quite long time since last purchase
- cluster 2: Moderate spend and long time since last purchase

Referring to Internet slang for spenders we can naming for all clusters as follows:

- cluster 0: Let Minnow
- cluster 1: It's been a Whale
- cluster 2: Dolphin-itely miss you

Final Cluster for Customer

Let Minnow

Customer with small amount of spending and have made purchase most recent.

Dolphin-itely miss you

Customer with moderate amount of spending and have not made the purchase for very long time.

It's been a Whale

Customer with big amount of spending and have not made the purchase for quite a long time.

Recommendation

Go Back to Home Page

Recommendation

to improve the future customer engagement, there are several recommendation to be considered

To minimize the risks of customer retention, campaign for customers with "It's been a Whale" should be more focussed.

Beside the campaign, company should doing analysis for the customers's habit, this can be improving ads, product recommendation and promotion for customers.

To bring back the the customers who have left, company needs to consider to start the campaign by giving them promo such as promo campaign with certain payment method or specific payment gateway/ provider. This campaign can be focussed on "Dolphin-itely miss you" and "It's been a Whale" customers.

For "Let Minnow" customers, campaign to boost the spending amount and to keep them stay with the platform is strongly recommended. Such as cahsback promo or rewarding referall can be the option for the campaign.

Thank you!

Contrary to popular belief, Lorem Ipsum is not simply random text.

Appendix

Full Python Script:

https://tinyurl.com/2mjv5akw

Any question, comment or feedback?
Feel free to reach me at:

https://www.linkedin.com/in/malvinkurniawan/