Customer Segmentation (K-Means, RFM)

Importing the libraries

```
!pip -q install pandas-profiling
!pip -q install ipywidgets
```

```
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 324.4/324.4 kB 4.1 MB/s eta 0:00:00
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 357.9/357.9 kB 12.4 MB/s eta 0:00:00
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 102.7/102.7 kB 7.9 MB/s eta 0:00:00
  Preparing metadata (setup.py) ... done
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 686.1/686.1 kB 12.6 MB/s eta 0:00:00
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 293.3/293.3 kB 11.4 MB/s eta 0:00:00
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 296.5/296.5 kB 11.7 MB/s eta 0:00:00
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 4.7/4.7 MB 19.5 MB/s eta 0:00:00
  Building wheel for htmlmin (setup.py) ... done
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 1.6/1.6 MB 8.2 MB/s eta 0:00:00
```

```
%matplotlib inline
import warnings
warnings.filterwarnings("ignore")
from sklearn.cluster import KMeans
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
sns.set(color_codes=True)
```

Importing the dataset

```
from google.colab import drive
drive.mount('/content/drive')
```

```
Mounted at /content/drive
```

```
# df = pd.read_csv('./data/Mall_Customers.csv')
#df = pd.read_csv('https://docs.google.com/spreadsheets/d/1_VLG-QVuEQFaABcsXxWmG1ASV-SVcEoEvoFDH8pUN4A/gviz/tq?tqx=out:csv&sheet=Mall_Cu
df = pd.read_csv('/content/drive/MyDrive/me/TUXSA/Independent Study/Dataset/dataset_2021-2023_CSV_colab.csv')
```

```
df.head()
```

	DocumentNo	Account	BusA	branch_no	Sale office	Assignment	Typ	Doc/ Date	Pstng Date	Net due dt	Amt in loc.cur.	LCurr	Clrng doc.	Clearing	Inv. r
0	8080370315	10000010	50	170	2005	5.011779e+09	Y6	44628	44628	44676	455905.6	THB	4.000576e+09	44676.0	808037(
1	8080648047	10000010	50	D001	2005	5.011818e+09	Y6	44652	44652	44742	64200.0	THB	4.000725e+09	44706.0	8080648
2	8080649893	10000010	50	D001	2005	5.011818e+09	Y6	44652	44652	44742	64200.0	THB	4.000725e+09	44706.0	8080649
3	8080939266	10000010	50	170	2005	5.011864e+09	Y6	44681	44681	44706	427411.5	THB	4.000725e+09	44706.0	8080939

```
df.describe()
```

	DocumentNo	Account	BusA	Assignment	Doc/ Date	Pstng Date	Net due dt	Amt in loc.cur.	Clrng doc.	C
count	1.034628e+06	1.034628e+06	1.034628e+06	1.034616e+06	1.034628e+06	1.034628e+06	1.034628e+06	1.034628e+06	9.873260e+05	98732
mean	8.108945e+09	1.000642e+07	5.031089e+01	5.760641e+10	4.474614e+04	4.474616e+04	4.477573e+04	3.279343e+05	4.269335e+09	4475
std	6.168763e+07	8.754358e+03	5.244167e-01	2.054090e+12	3.107723e+02	3.107722e+02	3.115730e+02	5.862482e+06	4.902036e+08	29
min	8.021093e+09	1.000001e+07	1.000000e+01	5.011051e+09	4.412600e+04	4.419700e+04	4.419700e+04	-1.122214e+09	2.100007e+09	4419
25%	8.078698e+09	1.000106e+07	5.000000e+01	5.011718e+09	4.448700e+04	4.448700e+04	4.451500e+04	2.792700e+03	4.000659e+09	4450
50%	8.082962e+09	1.000135e+07	5.000000e+01	5.012297e+09	4.474600e+04	4.474600e+04	4.477300e+04	1.350601e+05	4.001306e+09	4475
75%	8.205431e+09	1.000832e+07	5.100000e+01	5.513121e+09	4.500900e+04	4.500900e+04	4.504500e+04	3.967389e+05	4.800232e+09	4500

```
1 df.dtypes
```

```
DocumentNo              int64
Account                 int64
BusA                    int64
branch_no              object
Sale office            object
Assignment            float64
Typ                    object
Doc/ Date               int64
Pstng Date              int64
Net due dt              int64
   Amt in loc.cur.    float64
LCurr                  object
Clrng doc.            float64
Clearing              float64
Inv. ref.               int64
dtype: object
```

```
1 df.isnull().sum()
```

```
DocumentNo                0
Account                   0
BusA                      0
branch_no             18346
Sale office               0
Assignment               12
Typ                       0
Doc/ Date                 0
Pstng Date                0
Net due dt                0
   Amt in loc.cur.        0
LCurr                     0
Clrng doc.            47302
Clearing              47302
Inv. ref.                 0
dtype: int64
```

RFM Analysis

```
1 #pivot_table
2 #pivot_df = df.pivot(index='Date', columns='Category', values='Value', aggfunc= {'Fee': 'mean', 'Discount': 'sum'})
3 pivot_df = df.pivot_table(index=['Account'],aggfunc= {'Doc/ Date': 'max', 'DocumentNo': 'count', '   Amt in loc.cur.': 'sum'})
4
5
6 print(pivot_df)
7
8
```

```
             Amt in loc.cur.  Doc/ Date  DocumentNo
Account
10000009         47758380.00      45045         164
10000010          7745730.48      45231          45
10000012           609823.78      45283          46
10000016          4575790.80      45267          28
10000019         10255736.00      45279          67
...                      ...        ...         ...
10036433           646091.68      45225           1
10036458          3719833.60      45286           8
10036535          4943833.01      45287          99
```

```
10036585          1225641.56      45247           2
10036626           971153.40      45274           3

[1481 rows x 3 columns]
```

```
pivot_df.iloc[:,1]=45291-pivot_df.iloc[:,1]
print(pivot_df.head())
```

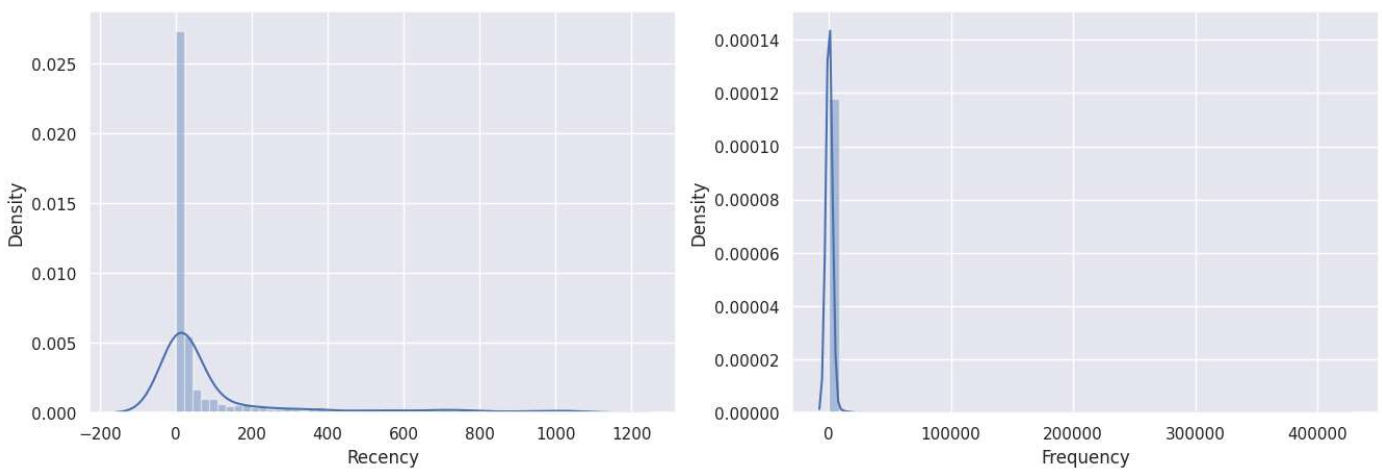
```
             Amt in loc.cur.  Doc/ Date  DocumentNo
Account
10000009         47758380.00        246         164
10000010          7745730.48         60          45
10000012           609823.78          8          46
10000016          4575790.80         24          28
10000019         10255736.00         12          67
```

```
pivot_df = pivot_df.rename(columns={'Doc/ Date': 'Recency', 'DocumentNo': 'Frequency', '   Amt in loc.cur.': 'Monetary'})
print(pivot_df.head())
```

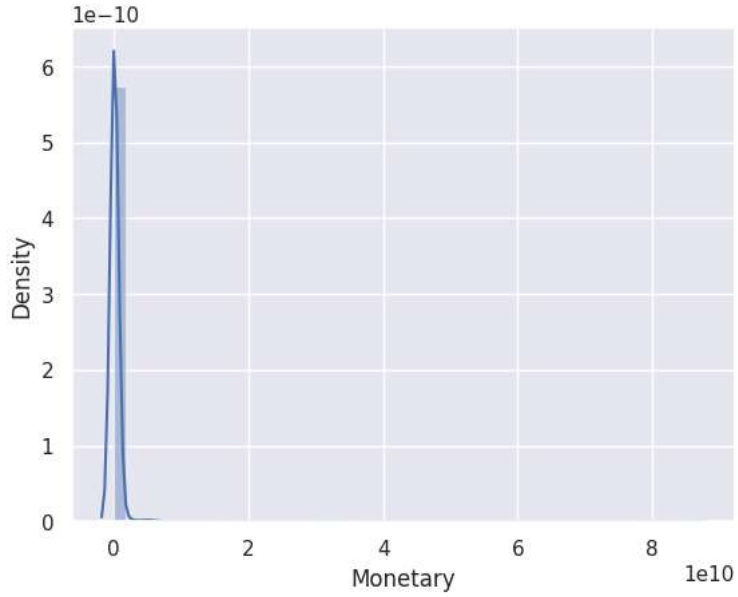
```
             Monetary  Recency  Frequency
Account
10000009  47758380.00      246        164
10000010   7745730.48       60         45
10000012    609823.78        8         46
10000016   4575790.80       24         28
10000019  10255736.00       12         67
```

Visualization the dataset

```
plt.figure(figsize=(16,5))
plt.subplot(1,2,1)
sns.distplot(pivot_df['Recency'])
plt.subplot(1,2,2)
sns.distplot(pivot_df['Frequency'])
plt.show()
```




```
sns.distplot(pivot_df['Monetary'])
plt.show()
```

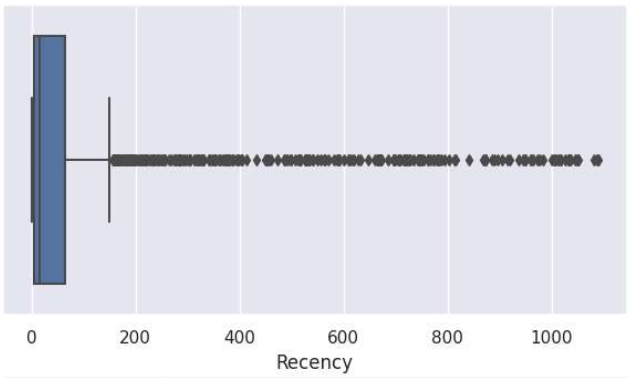


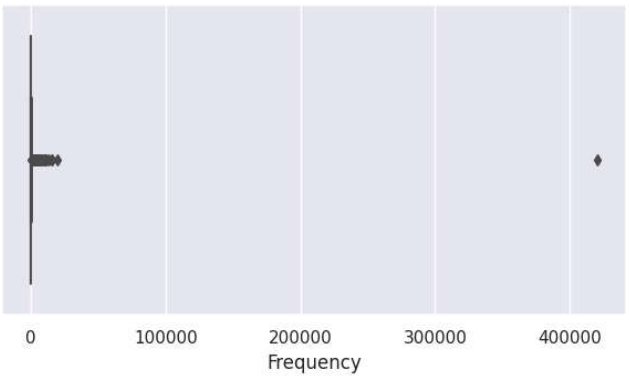

```
print(pivot_df.index)
pivot_df.shape
```

```
Int64Index([10000009, 10000010, 10000012, 10000016, 10000019, 10000022,
            10000023, 10000024, 10000033, 10000038,
            ...
            10036106, 10036233, 10036235, 10036307, 10036404, 10036433,
            10036458, 10036535, 10036585, 10036626],
           dtype='int64', name='Account', length=1481)
(1481, 3)
```

```
#outlierIQR Based Filtering
#Used when our data distribution is skewed.
```

```
plt.figure(figsize=(16,8))
plt.subplot(2,2,1)
sns.boxplot(x=pivot_df['Recency'])  #'x= '
plt.subplot(2,2,2)
sns.boxplot(x=pivot_df['Frequency'])  #'x= '
plt.subplot(2,2,3)
sns.boxplot(x=pivot_df['Monetary'])  #'x= '
plt.show()
```






```
1 #find iqr1
2 percentile25_R = pivot_df['Recency'].quantile(0.25)
3 percentile75_R = pivot_df['Recency'].quantile(0.75)
4 percentile25_F = pivot_df['Frequency'].quantile(0.25)
5 percentile75_F = pivot_df['Frequency'].quantile(0.75)
6 percentile25_M = pivot_df['Monetary'].quantile(0.25)
7 percentile75_M = pivot_df['Monetary'].quantile(0.75)
```

```
1 #finding iqr2
2 iqr_R = percentile75_R-percentile25_R
3 iqr_F = percentile75_R-percentile25_F
4 iqr_M = percentile75_R-percentile25_M
```

```
1 upper_limit_R = percentile75_R + 1.5 * iqr_R
2 lower_limit_R = percentile25_R - 1.5 * iqr_R
3 upper_limit_F = percentile75_F + 1.5 * iqr_F
4 lower_limit_F = percentile25_F - 1.5 * iqr_F
5 upper_limit_M = percentile75_M + 1.5 * iqr_M
6 lower_limit_M = percentile25_M - 1.5 * iqr_M
```

```
1 #finding outlier
2 print(pivot_df[pivot_df['Recency'] > upper_limit_R])
3 print(pivot_df[pivot_df['Recency'] < lower_limit_R])
4 print(pivot_df[pivot_df['Frequency'] > upper_limit_F])
5 print(pivot_df[pivot_df['Frequency'] < lower_limit_F])
6 print(pivot_df[pivot_df['Monetary'] > upper_limit_M])
7 print(pivot_df[pivot_df['Monetary'] < lower_limit_M])
```

```
              Monetary  Recency  Frequency
Account
10000009  4.775838e+07      246        164
10000144  2.852192e+06      188         19
10000179  6.659821e+09      220        581
10000182  5.902830e+09      318       9574
10000183  2.471346e+09      313       4661
...                ...      ...        ...
10033988  8.025899e+04      456          1
10034034  1.637100e+04      542          1
```

```
10034576  2.013290e+07      241          77
10034932  8.849786e+06      281          15
10034933  2.201279e+06      377           3

[268 rows x 3 columns]
Empty DataFrame
Columns: [Monetary, Recency, Frequency]
Index: []
              Monetary  Recency  Frequency
Account
10000022  2.077755e+08        6        546
10000044  8.730452e+07        6        994
10000063  3.576718e+08        5        462
10000085  2.978213e+08        3        584
10000164  4.365234e+07        6        999
...                ...      ...        ...
10032962  1.814979e+08        5        433
10032987  8.736211e+07      180        390
10033506  2.457659e+08        3       3453
10035056  2.795564e+08        2       1070
10035438  1.143263e+08        1        736

[332 rows x 3 columns]
Empty DataFrame
Columns: [Monetary, Recency, Frequency]
Index: []
              Monetary  Recency  Frequency
Account
10000022  2.077755e+08        6        546
10000044  8.730452e+07        6        994
10000063  3.576718e+08        5        462
10000085  2.978213e+08        3        584
10000147  1.505361e+08       13         40
...                ...      ...        ...
10033548  1.413332e+08       17        165
10033926  9.062658e+07        1        195
10035056  2.795564e+08        2       1070
10035438  1.143263e+08        1        736
10035915  6.251411e+07        5        176

[401 rows x 3 columns]
              Monetary  Recency  Frequency
Account
10000010   7745730.48       60         45
10000012    609823.78        8         46
10000016   4575790.80       24         28
10000019  10255736.00       12         67
10000023    146135.25       38          9
```

```
#TRIM OUTLIER
new_df1 = pivot_df[pivot_df['Recency'] < upper_limit_R]
new_df2 = new_df1[new_df1['Recency'] > lower_limit_R]
new_df3 = new_df2[new_df2['Frequency'] < upper_limit_F]
new_df4 = new_df3[new_df3['Frequency'] > lower_limit_F]
new_df5 = new_df4[new_df4['Monetary'] < upper_limit_M]
new_df6 = new_df5[new_df5['Monetary'] > lower_limit_M]
new_df6.shape
```

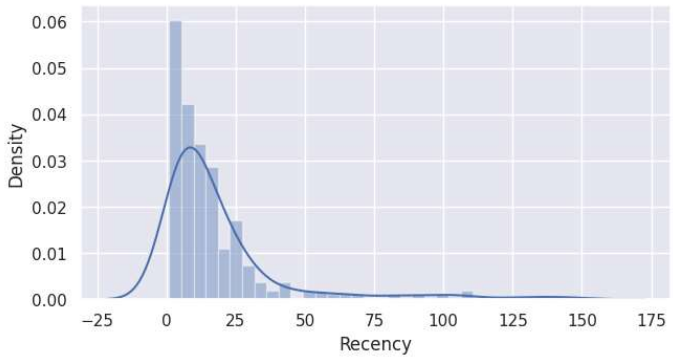
```
(374, 3)
```

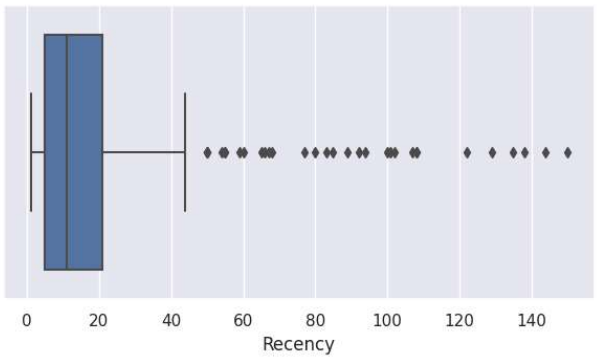
```
new_df6.head()
```

Account	Monetary	Recency	Frequency
10000033	14950789.00	21	63
10000079	20501257.78	101	65
10000083	31759151.50	9	84
10000099	21948139.60	3	66
10000141	50072479.70	4	127

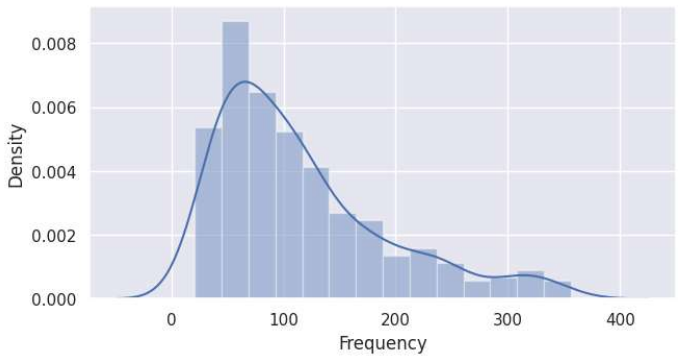
```
plt.figure(figsize=(16,8))
plt.subplot(2,2,1)
sns.distplot(new_df6['Recency'])
plt.subplot(2,2,2)
```

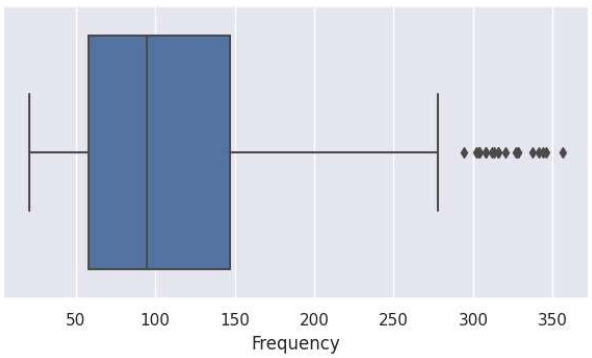
```
sns.boxplot(x=new_df6['Recency'])  #'x= '
plt.show()
```



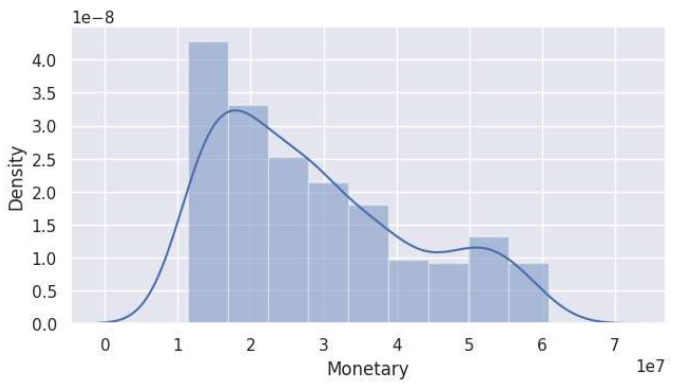


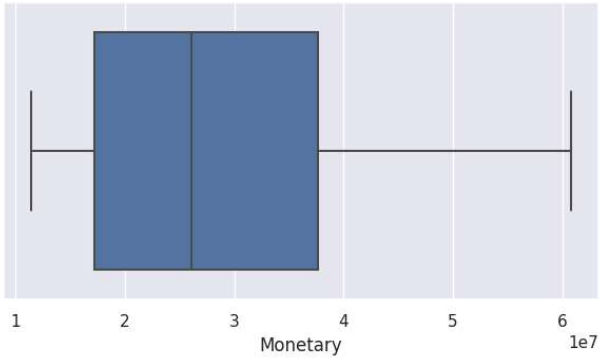

```
plt.figure(figsize=(16,8))
plt.subplot(2,2,1)
sns.distplot(new_df6['Frequency'])
plt.subplot(2,2,2)
sns.boxplot(x=new_df6['Frequency'])
plt.show()
```






```
plt.figure(figsize=(16,8))
plt.subplot(2,2,1)
sns.distplot(new_df6['Monetary'])
plt.subplot(2,2,2)
sns.boxplot(x=new_df6['Monetary'])
plt.show()
```





```
#merged_data = pd.merge(pivot_df, df, on='Account',how ='left')
'''
validatestr, optional
If specified, checks if merge is of specified type.

“one_to_one” or “1:1”: check if merge keys are unique in both left and right datasets.

“one_to_many” or “1:m”: check if merge keys are unique in left dataset.

“many_to_one” or “m:1”: check if merge keys are unique in right dataset.

“many_to_many” or “m:m”: allowed, but does not result in checks.
'''
```

```
'\nvalidatestr, optional\nIf specified, checks if merge is of specified type.\n\n“one_to_one” or “1:1”: check if merge keys are unique in both left and right datasets.\n\n“one_to_many” or “1:m”: check if merge keys are unique in left dataset.\n\n“many_to_one” or “m:1”: check if merge keys are unique in right dataset.\n\n“many_to_many” or “m:m”: allowed, but does not result in checks.\n'
```

```
#'merged_data = merged_data.drop_duplicates(subset=['Account'])'
```

```
#'merged_data.shape'
```

```
#merged_data.head()
```

```
new_df6_zscore = (new_df6- new_df6.mean())/new_df6.std()
```

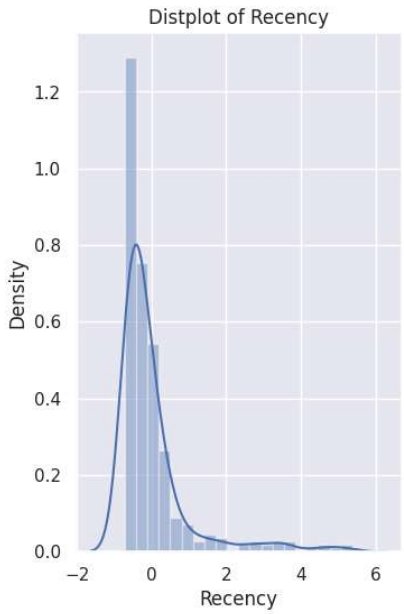
```
new_df6_zscore.head()
```

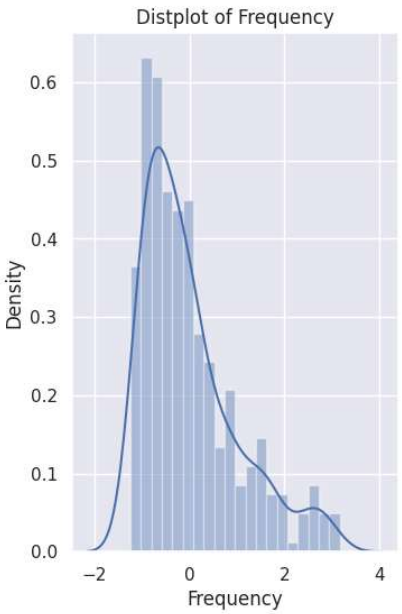
Account	Monetary	Recency	Frequency
10000033	-1.031700	0.094370	-0.692353
10000079	-0.620426	3.369948	-0.666057
10000083	0.213751	-0.396967	-0.416242
10000099	-0.513217	-0.642635	-0.652909
10000141	1.570717	-0.601690	0.149130

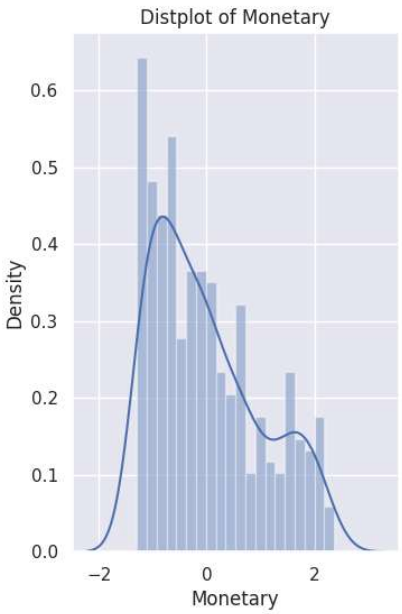
```
'Detect out lier'
'Normalization HERE!!!!'
```

```
'Normalization HERE!!!!'
```

```
plt.figure(1 , figsize = (15 , 6))
n = 0
for x in ['Recency' , 'Frequency' , 'Monetary']:
    n += 1
    plt.subplot(1 , 3 , n)
    plt.subplots_adjust(hspace =0.5 , wspace = 0.5)
    sns.distplot(new_df6_zscore[x] , bins = 20) # change 'df' here
    plt.title('Distplot of {}'.format(x))
plt.show()
```

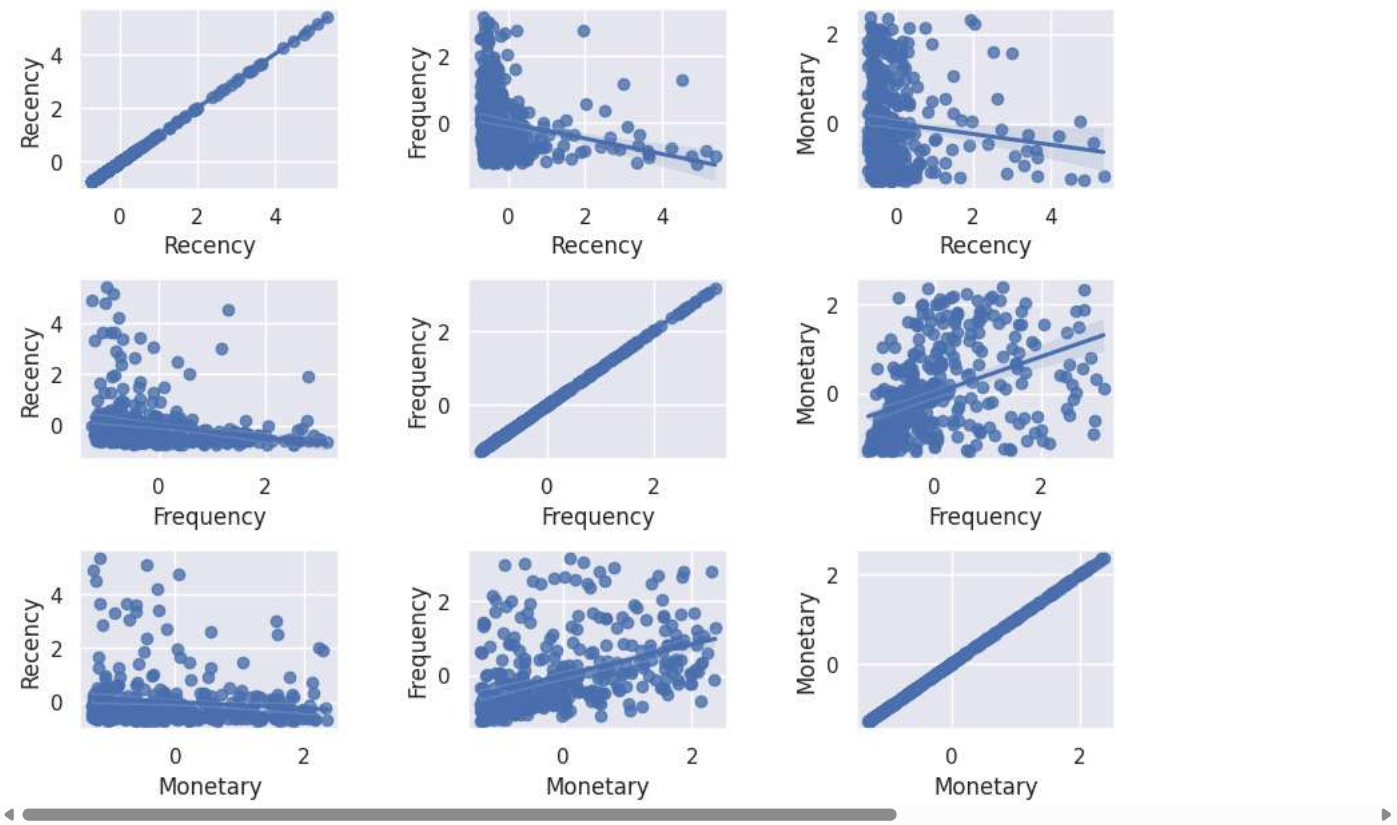






Relationship between Age, Annual Income, Spending Score

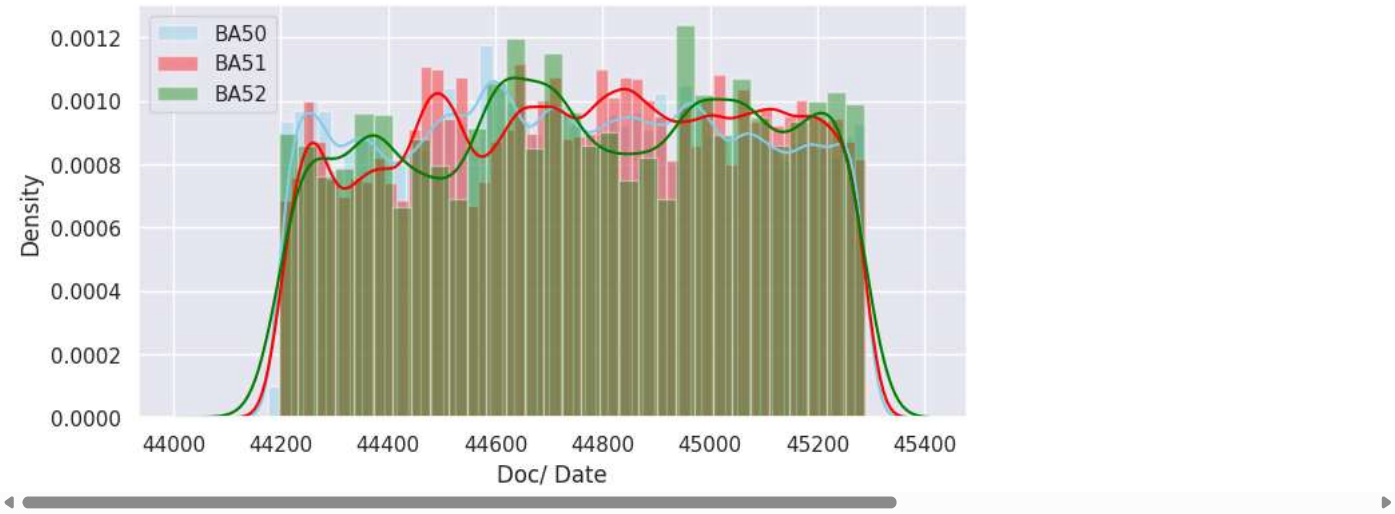
```
plt.figure(1 , figsize = (10 , 7))
n = 0
for x in ['Recency' , 'Frequency' , 'Monetary']:
    for y in ['Recency' , 'Frequency' , 'Monetary']:
        n += 1
        plt.subplot(3 , 3 , n)
        plt.subplots_adjust(hspace = 0.5 , wspace = 0.5)
        sns.regplot(x = x , y = y , data = new_df6_zscore) # change 'df' here or new_df6
        plt.ylabel(y.split()[0]+' '+y.split()[1] if len(y.split()) > 1 else y )
plt.show()
```



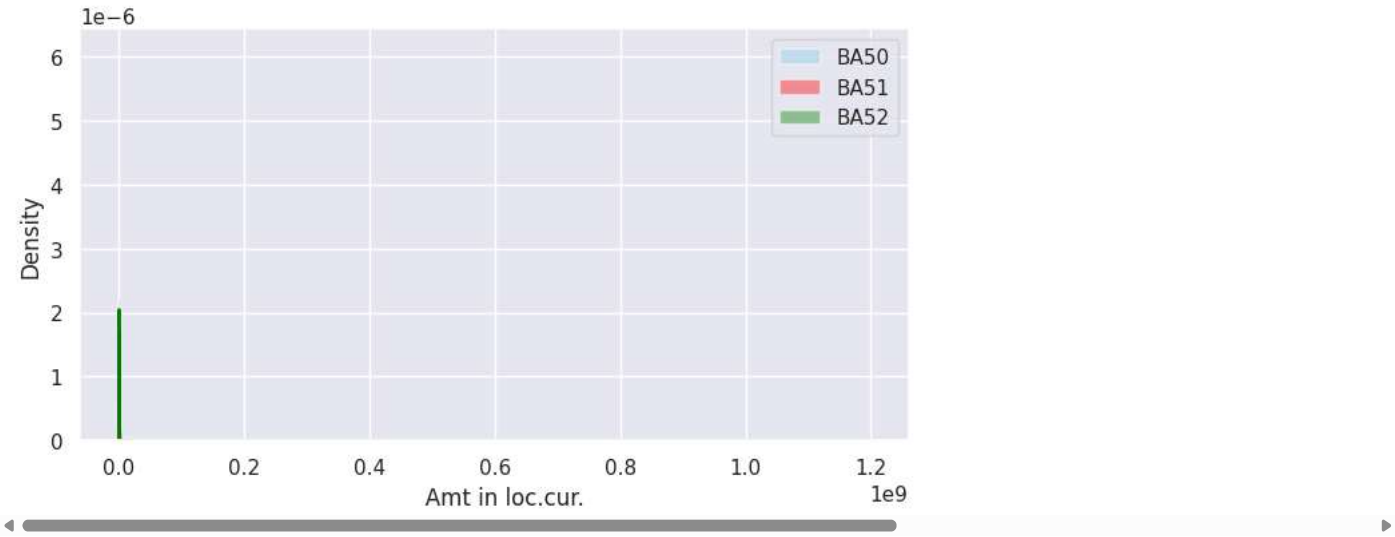

```
1 new_df6.head()
```

Account	Monetary	Recency	Frequency
10000033	14950789.00	21	63
10000079	20501257.78	101	65
10000083	31759151.50	9	84
10000099	21948139.60	3	66
10000141	50072479.70	4	127

```
1 #date by BA
2 plt.figure(figsize=(8,4))
3 sns.distplot( df[df["BusA"]==50]['Doc/ Date'], color="skyblue",label="BA50")
4 sns.distplot( df[df["BusA"]==51]['Doc/ Date'], color="Red",label="BA51")
5 sns.distplot( df[df["BusA"]==52]['Doc/ Date'], color="green",label="BA52")
6 #sns.distplot( df[df["BusA"]==10]['Doc/ Date'], color="yellow",label="BA10")
7 plt.legend();
```




```
#amount by BA
plt.figure(figsize=(8,4))
sns.distplot( df[df["BusA"]==50][df['   Amt in loc.cur.']>50000]['   Amt in loc.cur.'], color="skyblue",label="BA50")
sns.distplot( df[df["BusA"]==51][df['   Amt in loc.cur.']>50000]['   Amt in loc.cur.'], color="Red",label="BA51")
sns.distplot( df[df["BusA"]==52][df['   Amt in loc.cur.']>50000]['   Amt in loc.cur.'], color="green",label="BA52")
#sns.distplot( df[df["BusA"]==10]['Doc/ Date'], color="yellow",label="BA10")
plt.legend();
```

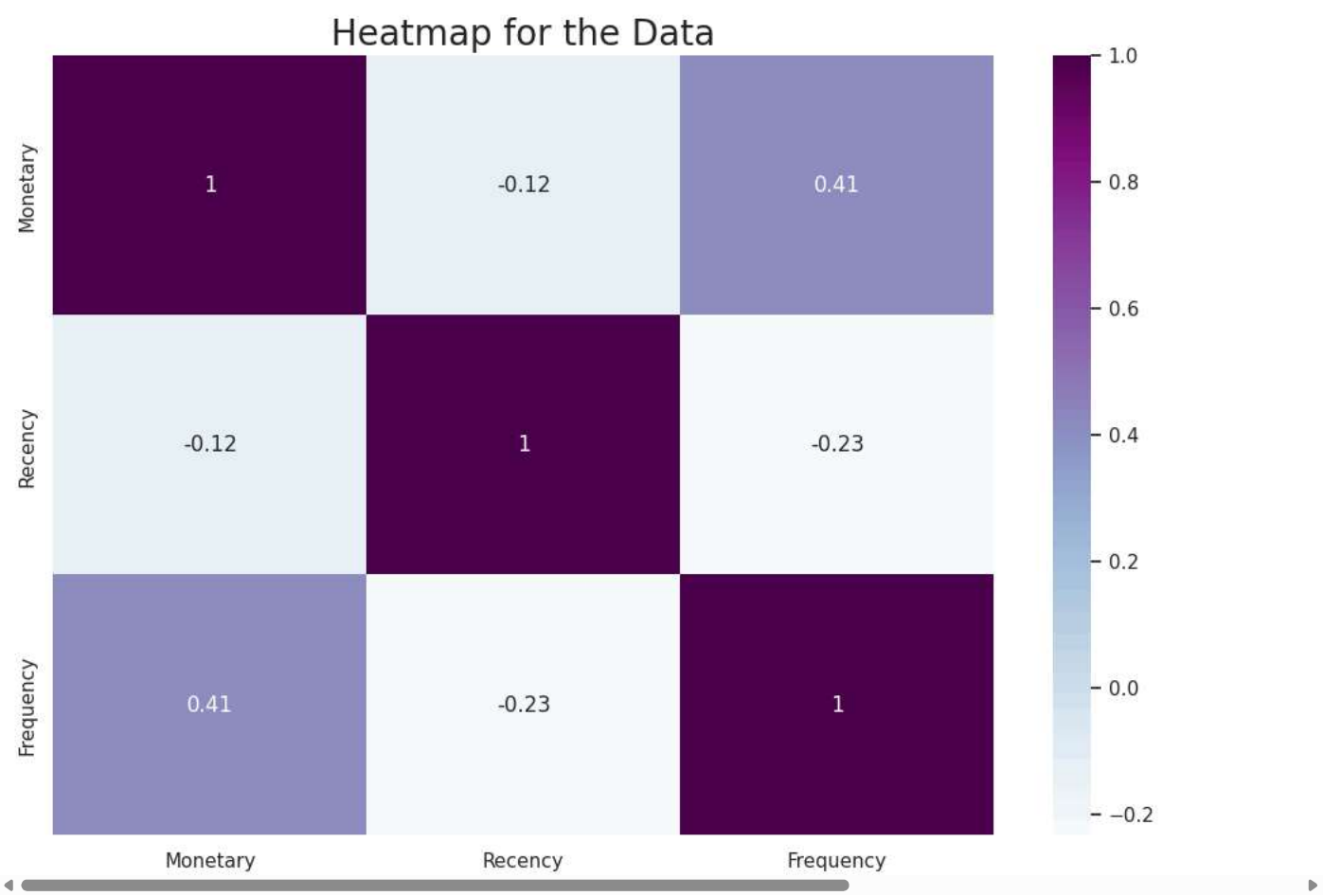



```
df.head()
```

	DocumentNo	Account	BusA	branch_no	Sale office	Assignment	Typ	Doc/ Date	Pstng Date	Net due dt	Amt in loc.cur.	LCurr	Clrng doc.	Clearing	Inv. r
0	8080370315	10000010	50	170	2005	5.011779e+09	Y6	44628	44628	44676	455905.6	THB	4.000576e+09	44676.0	808037(
1	8080648047	10000010	50	D001	2005	5.011818e+09	Y6	44652	44652	44742	64200.0	THB	4.000725e+09	44706.0	8080648
2	8080649893	10000010	50	D001	2005	5.011818e+09	Y6	44652	44652	44742	64200.0	THB	4.000725e+09	44706.0	8080649
3	8080939266	10000010	50	170	2005	5.011864e+09	Y6	44681	44681	44706	427411.5	THB	4.000725e+09	44706.0	8080939

Correlation Heatmap

```
plt.rcParams['figure.figsize'] = (12, 8)
sns.heatmap(new_df6_zscore.corr(), cmap = 'BuPu', annot = True)
plt.title('Heatmap for the Data', fontsize = 20)
plt.show()
```



สำหรับ kMean ข้อมูลที่ใช้จำเป็นต้องเป็น numerical

```
1 X = new_df6
2 X.head()
```

Account	Monetary	Recency	Frequency
10000033	14950789.00	21	63
10000079	20501257.78	101	65
10000083	31759151.50	9	84
10000099	21948139.60	3	66
10000141	50072479.70	4	127

```
1 from ydata_profiling import ProfileReport
2 ProfileReport(X)
```

Summarize dataset: 100% 21/21 [00:04<00:00, 2.93it/s, Completed]

Generate report structure: 100% 1/1 [00:02<00:00, 2.28s/it]

Render HTML: 100% 1/1 [00:00<00:00, 1.81it/s]

Overview

Dataset statistics

Number of variables	3
Number of observations	374
Missing cells	0
Missing cells (%)	0.0%
Duplicate rows	0
Duplicate rows (%)	0.0%
Total size in memory	11.7 KiB
Average record size in memory	32.0 B

Variable types

Numeric	3

Alerts

`Frequency` is highly overall correlated with `Monetary`	High correlation
`Monetary` is highly overall correlated with `Frequency`	High correlation
`Monetary` has unique values	Unique

Reproduction

Analysis started	2024-03-06 15:18:36.715653

Feature Scaling

```
1 from sklearn.preprocessing import StandardScaler
2 scaler = StandardScaler()
3 X_standard=scaler.fit_transform(X)
```

```
1 print(X_standard)
```

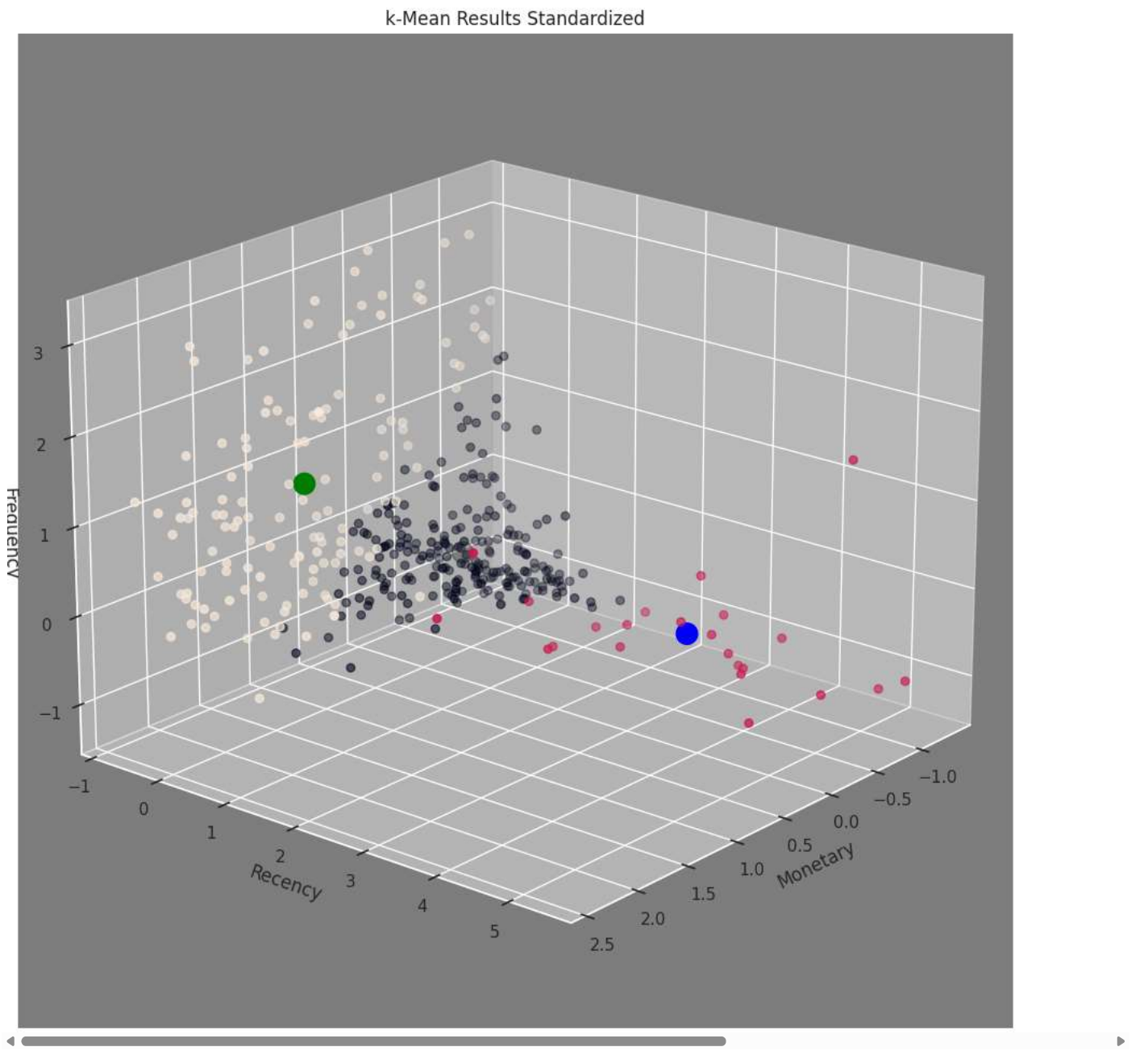
```
[[-1.03308236  0.09449634 -0.69328088]
 [-0.62125749  3.37446217 -0.6669493 ]
 [ 0.21403784 -0.39749853 -0.41679931]
 ...
 [ 0.09677253 -0.06950195 -0.95659666]
 [ 0.02161701 -0.56149682 -0.73277825]
 [-1.18003601 -0.39749853 -1.20674666]]
```

```
1 from sklearn.cluster import KMeans
2 km = KMeans(n_clusters=3)
3 km.fit(X_standard)
4 label = km.predict(X_standard)
5 centers_standard = km.cluster_centers_
```

```
1 from mpl_toolkits.mplot3d import Axes3D
2 import ipywidgets as widgets
```

```
from IPython.display import display
```

```
fig = plt.figure(1, figsize=(12,12))
ax = fig.add_subplot(projection='3d')
ax.set_facecolor("gray")
ax.scatter(X_standard[:, 0], X_standard[:, 1], X_standard[:, 2], c=label, s=30)
ax.scatter (centers_standard[:,0],centers_standard[:,1],centers_standard[:,2], c=['black','blue','green'],s=200)
ax.set_title("k-Mean Results Standardized")
ax.set_xlabel("Monetary")
ax.set_ylabel("Recency")
ax.set_zlabel("Frequency")
angle1 = 20
angle2 = 40
ax.view_init(angle1, angle2)
plt.show()
```



เปลี่ยนข้อมูลกลับไป scale เดิม

```
centers_original = scaler.inverse_transform(centers_standard)
X_original = scaler.inverse_transform(X_standard) # จริงๆใช้ค่า X ตั้งต้นก็ได้
```

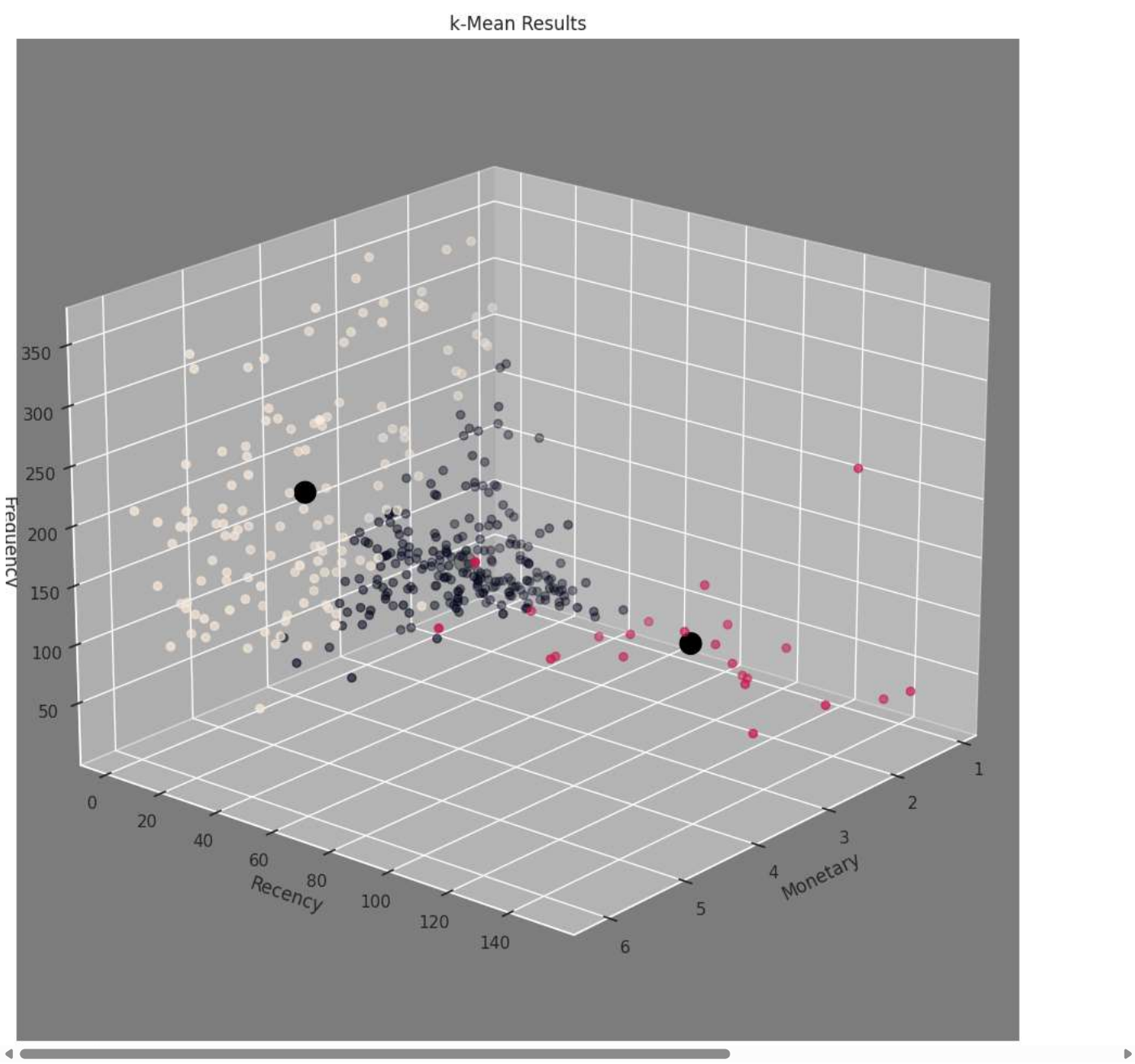
```
pd.DataFrame(centers_original, columns=X.columns) #ใส่ชื่อ column เดิมดู
```

	Monetary	Recency	Frequency
0	2.165939e+07	14.536036	77.936937
1	2.359678e+07	99.782609	73.217391
2	4.223191e+07	11.395349	188.139535

```
print(X_original)
```

```
[[1.49507890e+07 2.10000000e+01 6.30000000e+01]
 [2.05012578e+07 1.01000000e+02 6.50000000e+01]
 [3.17591515e+07 9.00000000e+00 8.40000000e+01]
 ...
 [3.01786802e+07 1.70000000e+01 4.30000000e+01]
 [2.91657536e+07 5.00000000e+00 6.00000000e+01]
 [1.29701858e+07 9.00000000e+00 2.40000000e+01]]
```

```
fig = plt.figure(1, figsize=(12,12))
ax = fig.add_subplot(projection='3d')
ax.scatter(X_original[:, 0], X_original[:, 1], X_original[:, 2], c=label, s=30)
ax.scatter (centers_original[:,0],centers_original[:,1],centers_original[:,2], c='black',s=200)
ax.set_title("k-Mean Results")
ax.set_xlabel("Monetary")
ax.set_ylabel("Recency")
ax.set_zlabel("Frequency")
ax.set_facecolor("gray")
angle1 = 20
angle2 = 40
ax.view_init(angle1, angle2)
plt.show()
```



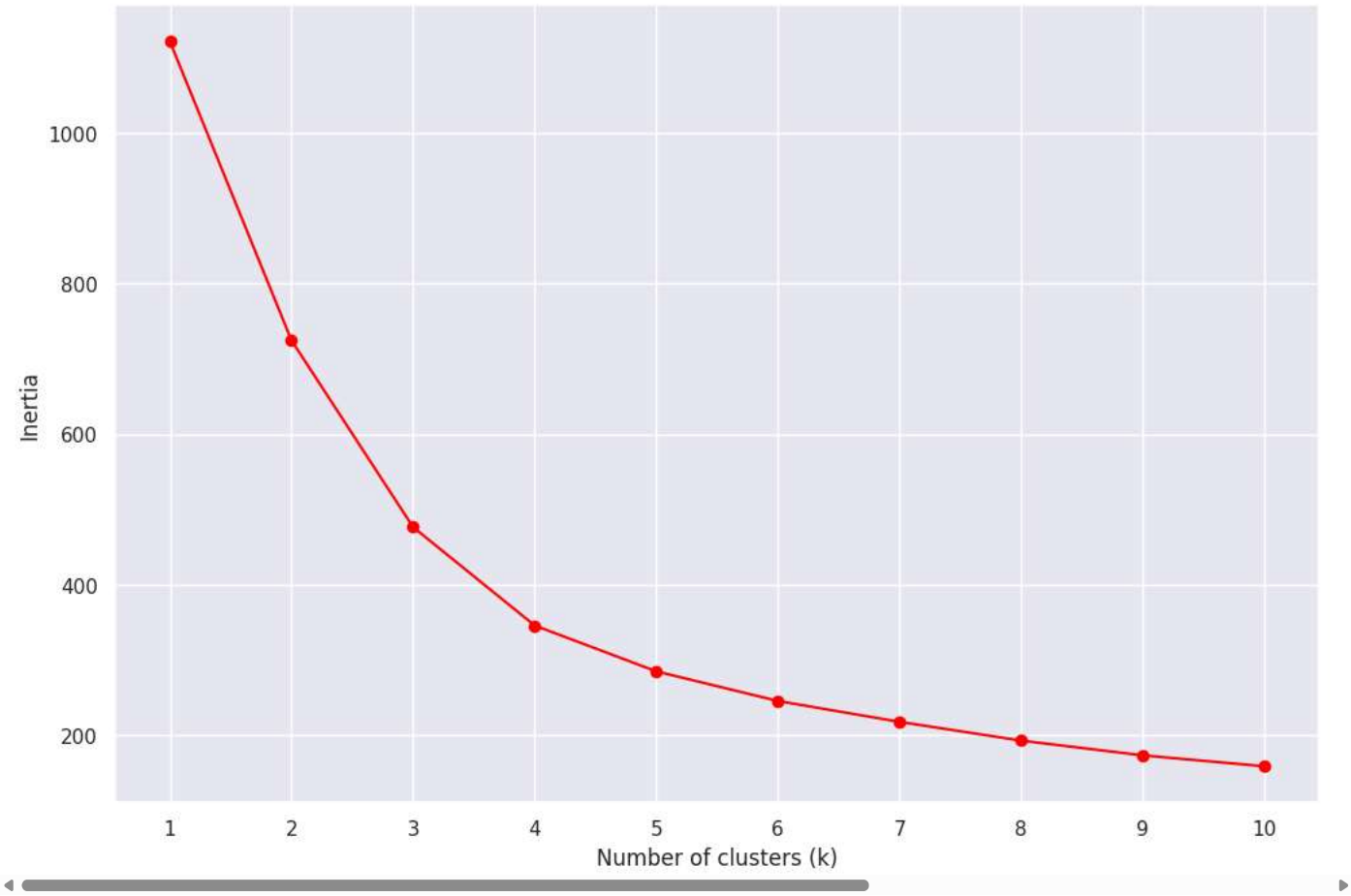
Choosing proper *k*

ทดลอง cluster ด้วยค่า k หลายๆค่าและเลือกอันที่ inertia เหมาะสมที่สุด (ที่ "**elbow**")

```
inertia_list = []
for i in range(1, 11):
    kmeans = KMeans(n_clusters = i, random_state = 42)
    kmeans.fit(X_standard)
    # inertia method returns wcss for that model
    inertia_list.append(kmeans.inertia_)
```

```
plt.plot(inertia_list,marker='o',color='red')
plt.xticks(range(0,10), [str(i) for i in range (1,11)])
plt.xlabel("Number of clusters (k)")
plt.ylabel("Inertia")
```

```
Text(0, 0.5, 'Inertia')
```



จะเห็นได้ว่าค่าของ k=2,4 เป็นอีกตัวเลือกหนึ่งที่น่าสนใจ

```
km2 = KMeans(n_clusters=4)
km2.fit(X_standard)
label2 = km2.predict(X_standard)
centers_standard2 = km2.cluster_centers_
```

```
centers_original2 = scaler.inverse_transform(centers_standard2)
```

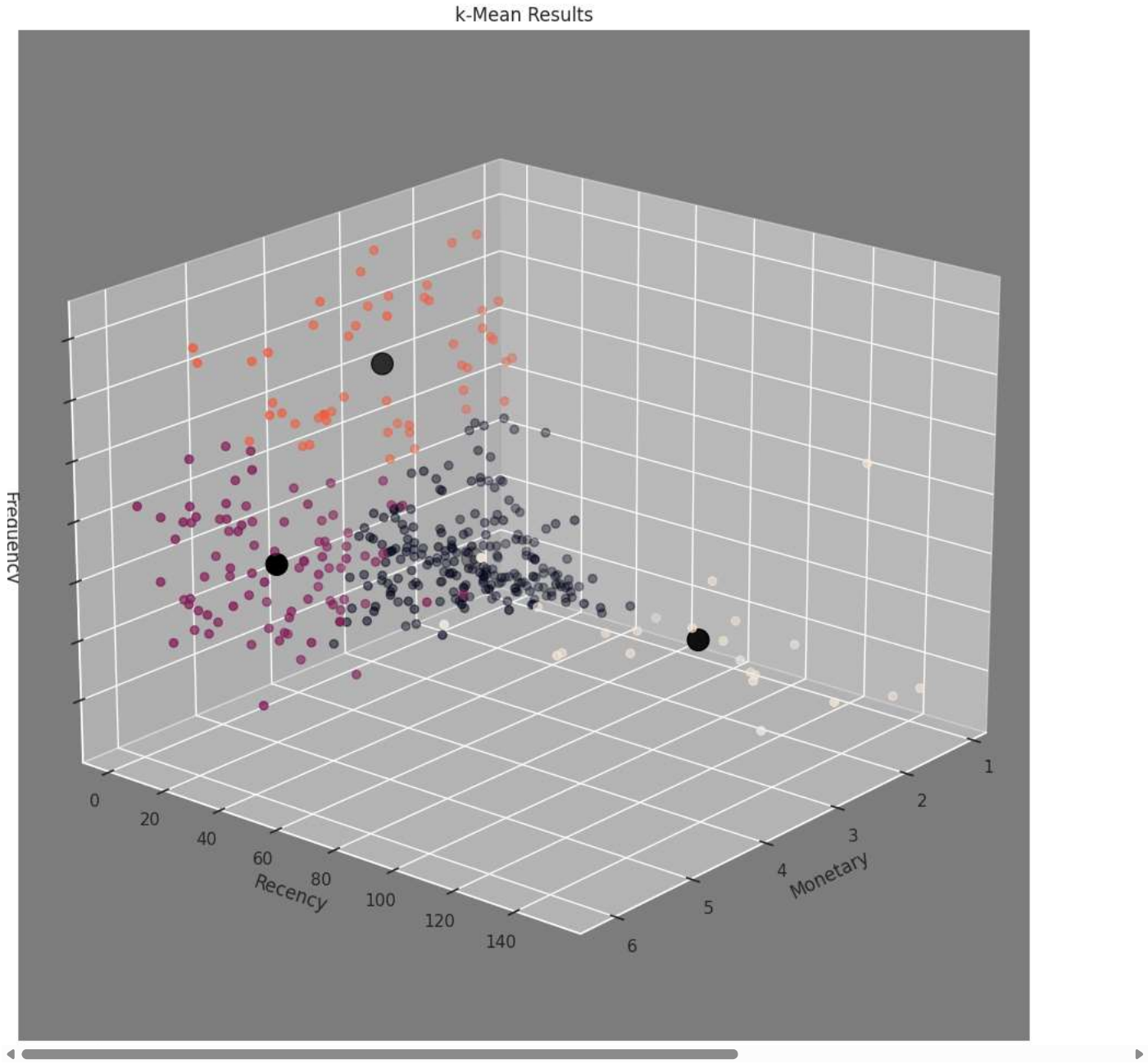
```
pd.DataFrame(centers_original2, columns=X.columns) #ใส่ชื่อ column เดิมดู
```

	Monetary	Recency	Frequency
0	2.134008e+07	14.448598	76.378505
1	4.746112e+07	13.678161	137.551724
2	3.120815e+07	8.300000	265.200000
3	2.359678e+07	99.782609	73.217391

```
from mpl_toolkits.mplot3d import Axes3D
import ipywidgets as widgets
from IPython.display import display
```

```
fig = plt.figure(2, figsize=(12,12))
ax = fig.add_subplot(projection='3d')
ax.scatter(X_original[:, 0], X_original[:, 1], X_original[:, 2], c=label2, s=30)
ax.scatter (centers_original2[:,0],centers_original2[:,1],centers_original2[:,2], c='black',s=200)
ax.set_title("k-Mean Results")
ax.set_xlabel("Monetary")
ax.set_ylabel("Recency")
ax.set_zlabel("Frequency")
ax.w_zaxis.set_ticklabels([])
ax.set_facecolor("gray")
```

```
11 angle1 = 20
12 angle2 = 40
13 ax.view_init(angle1, angle2)
14
15 plt.show()
```



Code ของ widget นี้ปรับมาจาก (https://stackoverflow.com/questions/42998009/clear-matplotlib-figure-in-jupyter-python-notebook)

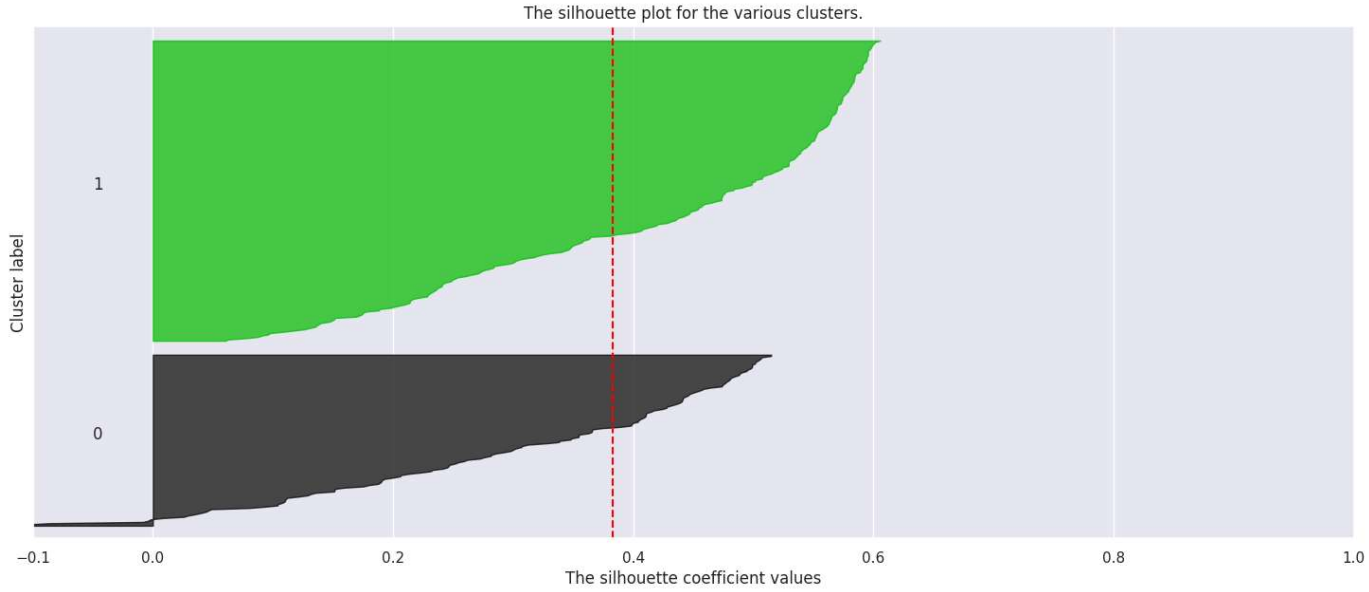
Silouette Score

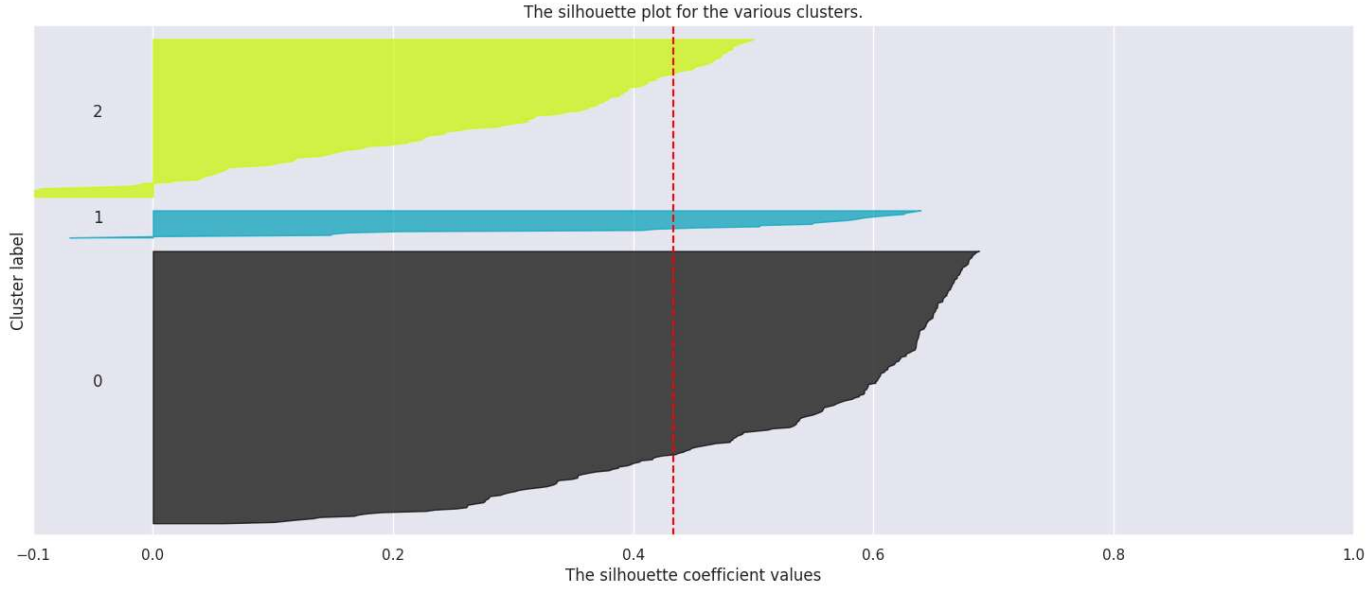
เพื่อช่วยในการตัดสินใจ เราอาจนำ silouette score เข้ามาช่วยพิจารณาด้วย *(code ของ plot ปรับจาก https://scikit-learn.org/stable/modules/generated/sklearn.metrics.silhouette_score.html)*

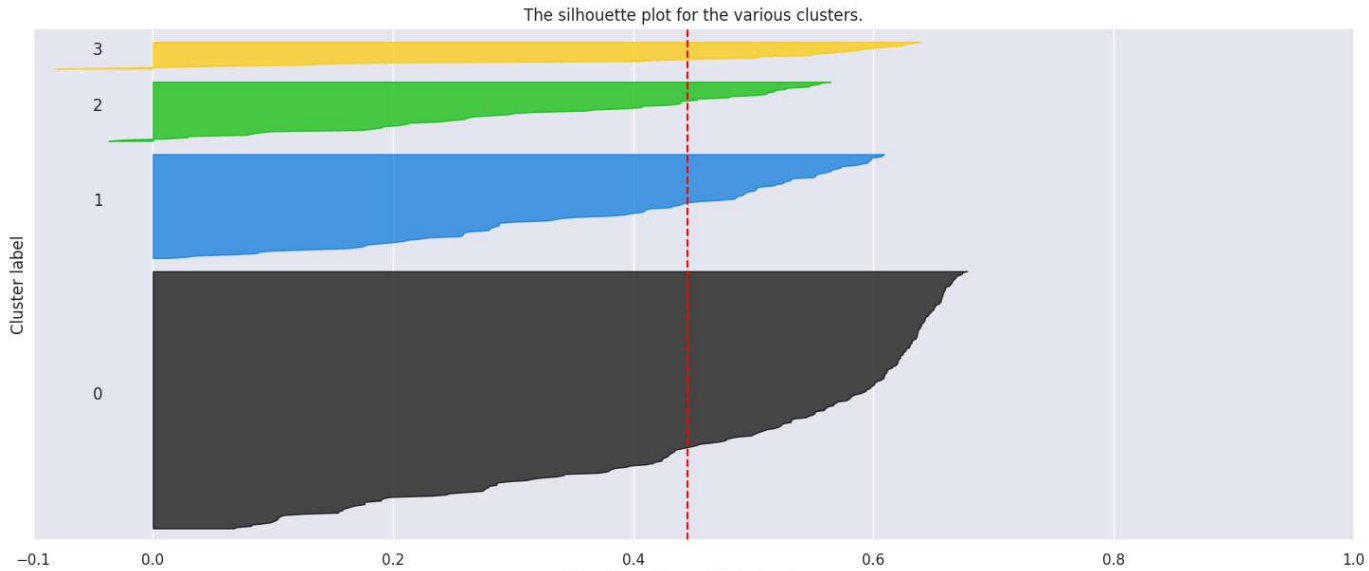
```
1 %matplotlib inline
2 from sklearn.metrics import silhouette_samples, silhouette_score
3 import matplotlib.cm as cm
4
5 range_n_clusters = range(2,10)
6
7 for n_clusters in range_n_clusters:
```

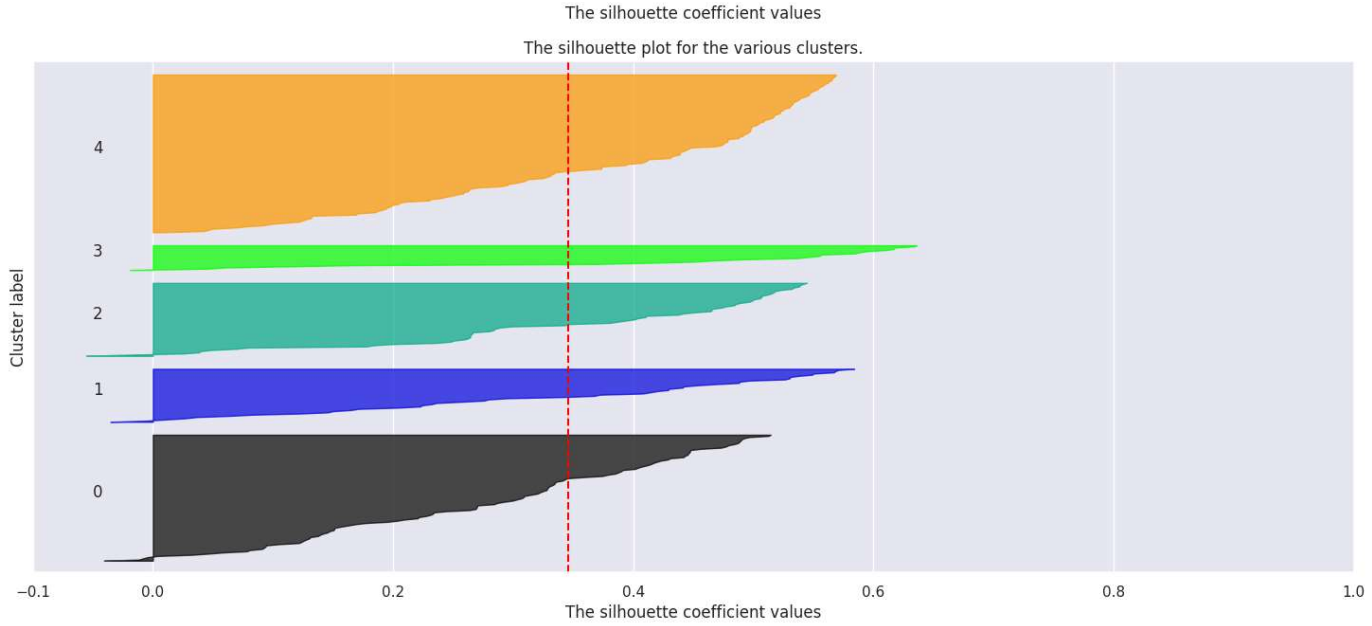
```
    # Create a subplot with 1 row and 2 columns
    fig, ax1 = plt.subplots(1, 1)
    fig.set_size_inches(18, 7)

    # The 1st subplot is the silhouette plot
    # The silhouette coefficient can range from -1, 1 but in this example all
    # lie within [-0.1, 1]
    ax1.set_xlim([-0.1, 1])
    # The (n_clusters+1)*10 is for inserting blank space between silhouette
    # plots of individual clusters, to demarcate them clearly.
    ax1.set_ylim([0, len(X_standard) + (n_clusters + 1) * 10])

    # Initialize the clusterer with n_clusters value and a random generator
    # seed of 10 for reproducibility.
    clusterer = KMeans(n_clusters=n_clusters, random_state=10)
    cluster_labels = clusterer.fit_predict(X_standard)

    # The silhouette_score gives the average value for all the samples.
    # This gives a perspective into the density and separation of the formed
    # clusters
    silhouette_avg = silhouette_score(X_standard, cluster_labels)
    print("For n_clusters =", n_clusters,
          "The average silhouette_score is :", silhouette_avg)

    # Compute the silhouette scores for each sample
    sample_silhouette_values = silhouette_samples(X_standard, cluster_labels)

    y_lower = 10
    for i in range(n_clusters):
        # Aggregate the silhouette scores for samples belonging to
        # cluster i, and sort them
        ith_cluster_silhouette_values = \
            sample_silhouette_values[cluster_labels == i]

        ith_cluster_silhouette_values.sort()

        size_cluster_i = ith_cluster_silhouette_values.shape[0]
        y_upper = y_lower + size_cluster_i

        color = cm.nipy_spectral(float(i) / n_clusters)
        ax1.fill_betweenx(np.arange(y_lower, y_upper),
                          0, ith_cluster_silhouette_values,
                          facecolor=color, edgecolor=color, alpha=0.7)

        # Label the silhouette plots with their cluster numbers at the middle
        ax1.text(-0.05, y_lower + 0.5 * size_cluster_i, str(i))

        # Compute the new y_lower for next plot
        y_lower = y_upper + 10  # 10 for the 0 samples

    ax1.set_title("The silhouette plot for the various clusters.")
    ax1.set_xlabel("The silhouette coefficient values")
    ax1.set_ylabel("Cluster label")

    # The vertical line for average silhouette score of all the values
    ax1.axvline(x=silhouette_avg, color="red", linestyle="--")

    ax1.set_yticks([])  # Clear the yaxis labels / ticks
    ax1.set_xticks([-0.1, 0, 0.2, 0.4, 0.6, 0.8, 1])


plt.show()
```

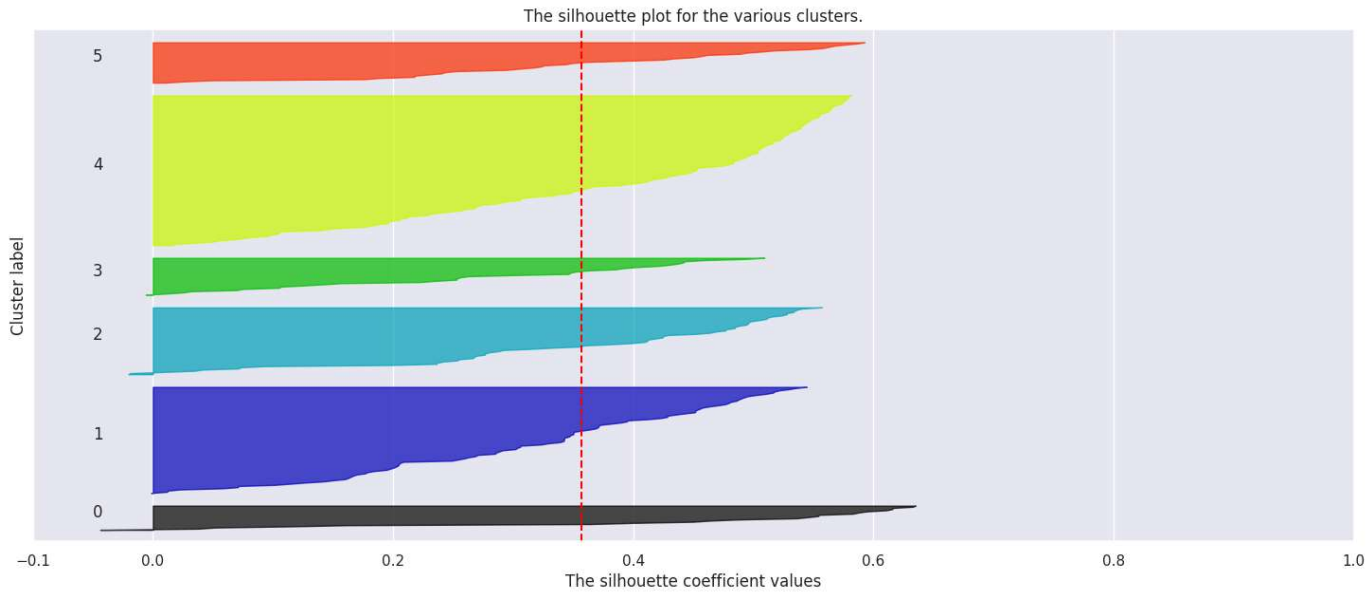
```
For n_clusters = 2 The average silhouette_score is : 0.383052847234259
For n_clusters = 3 The average silhouette_score is : 0.43302401618030767
For n_clusters = 4 The average silhouette_score is : 0.4449268248469187
For n_clusters = 5 The average silhouette_score is : 0.34601012131136283
For n_clusters = 6 The average silhouette_score is : 0.3571254056887799
For n_clusters = 7 The average silhouette_score is : 0.36297908775183924
For n_clusters = 8 The average silhouette_score is : 0.33885076059901126
For n_clusters = 9 The average silhouette_score is : 0.3436551120476924
```

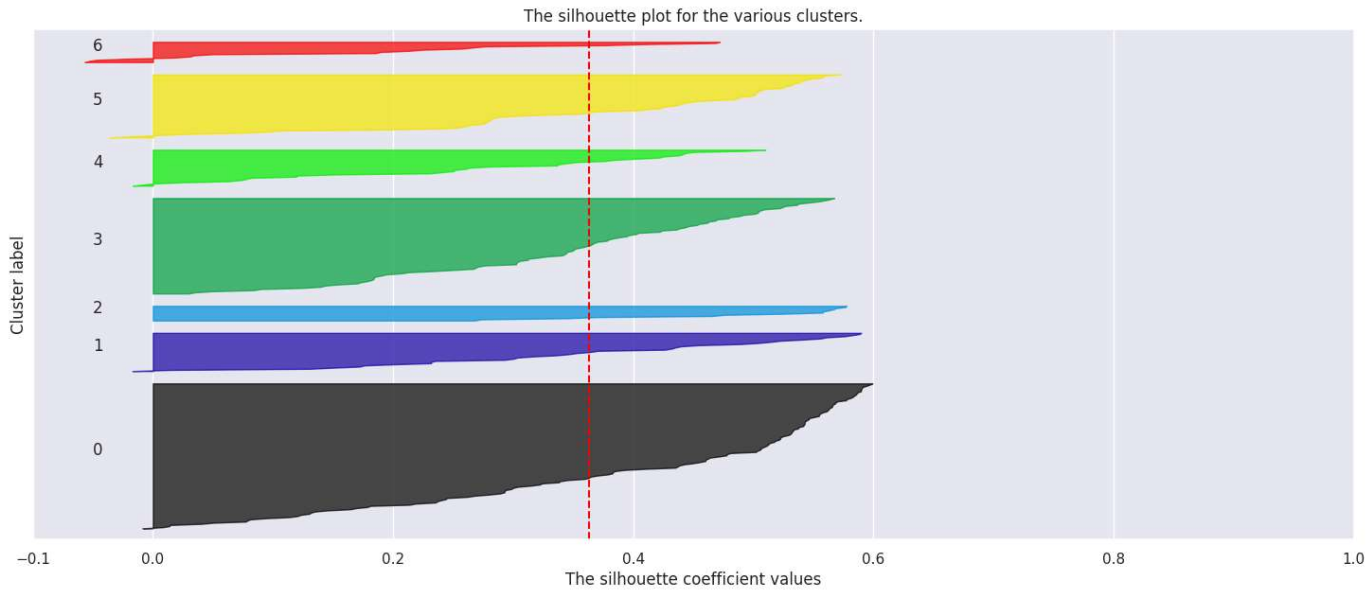


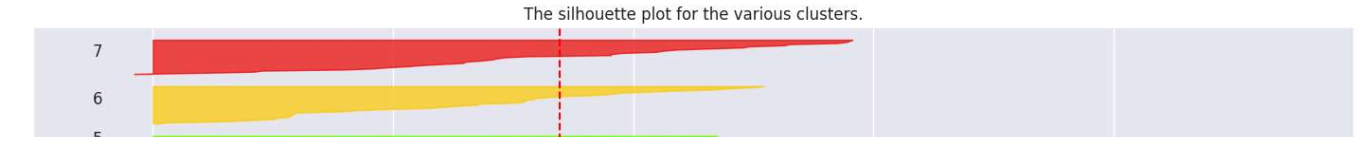


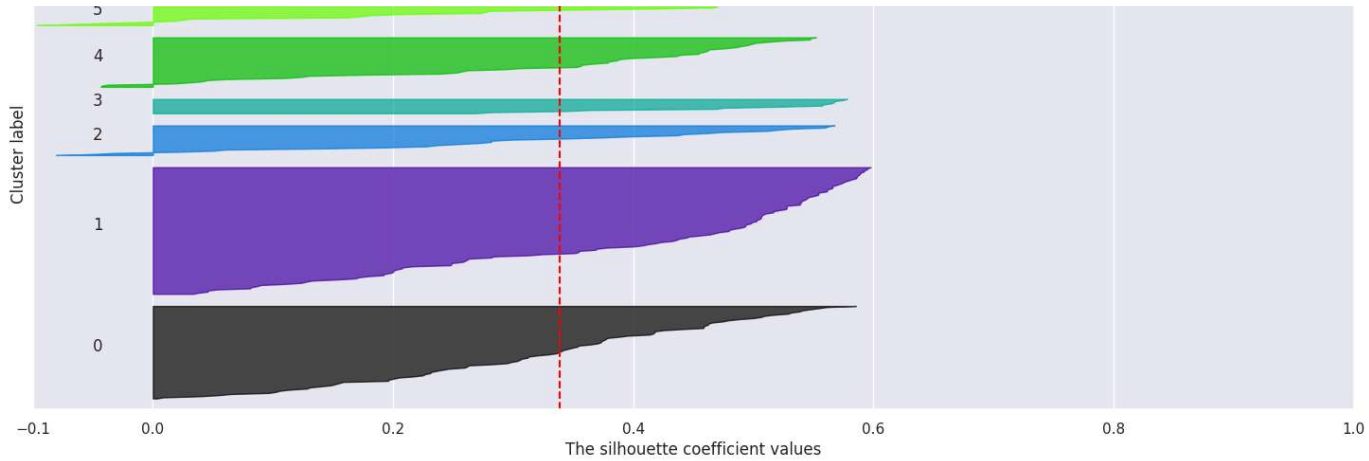



The silhouette coefficient values
The silhouette plot for the various clusters.
Cluster label
4
3
2
1
0
−0.1
0.0
0.2
0.4
0.6
0.8
1.0
The silhouette coefficient values


The silhouette plot for the various clusters.
Cluster label
5
4
3
2
1
0
−0.1
0.0
0.2
0.4
0.6
0.8
1.0
The silhouette coefficient values


The silhouette plot for the various clusters.
Cluster label
6
5
4
3
2
1
0
−0.1
0.0
0.2
0.4
0.6
0.8
1.0
The silhouette coefficient values


The silhouette plot for the various clusters.
7
6



The silhouette plot for the various clusters.