giuakippncln-3-1

October 22, 2023

0.1 Import

[1]:
```
from mpl_toolkits.mplot3d import Axes3D
from scipy import stats
from sklearn.metrics import  silhouette_score
from sklearn.preprocessing import StandardScaler
from sklearn.cluster import KMeans
from yellowbrick.cluster import KElbowVisualizer
import mplcursors

import pandas as pd
import matplotlib.pyplot as plt
import squarify
import seaborn as sns
import numpy as np

%matplotlib inline
```

0.2 Đọc file csv

[2]:
```
data_type = {
    '_CustomerID': str
}

sales_data = pd.read_csv('./sales_data.csv', dtype=data_type,␣
 ↪parse_dates=['OrderDate'])
```

[3]:
```
sales_data.describe()
```

[3]:
```
                           OrderDate  _SalesTeamID     _StoreID   _ProductID  \
count                           7991   7991.000000  7991.000000  7991.000000
mean   2019-09-15 11:01:09.828557312     14.384307   183.850081    23.771743
min              2018-05-31 00:00:00      1.000000     1.000000     1.000000
25%              2019-01-16 12:00:00      8.000000    91.000000    12.000000
50%              2019-09-15 00:00:00     14.000000   183.000000    24.000000
75%              2020-05-12 00:00:00     21.000000   276.000000    36.000000
max              2020-12-30 00:00:00     28.000000   367.000000    47.000000
```

```
std                              NaN     7.986086   105.903946    13.526545

       Order Quantity  Discount Applied   Unit Price    Unit Cost
count     7991.000000       7991.000000  7991.000000  7991.000000
mean         4.525341          0.114394  2284.536504  1431.911054
min          1.000000          0.050000   167.500000    68.675000
25%          3.000000          0.050000  1031.800000   606.115500
50%          5.000000          0.075000  1849.200000  1080.576000
75%          7.000000          0.150000  3611.300000  2040.250500
max          8.000000          0.400000  6566.000000  5498.556000
std          2.312631          0.085570  1673.096364  1112.413043
```

[4]:

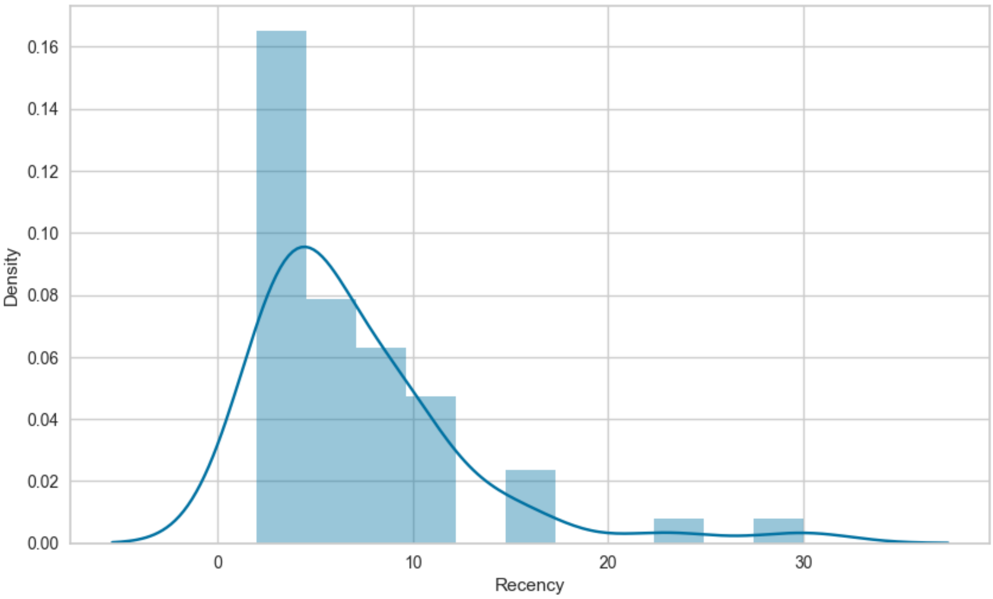
```
df = sales_data.copy()
df['Revenue'] = (df['Unit Price'] - (df['Unit Price'] * df['Discount Applied'])␣
 ↪- df['Unit Cost']) * df['Order Quantity']

columns = ['OrderNumber', '_CustomerID', 'OrderDate', 'Revenue']
df_dataset = df[columns]

today_date = pd.to_datetime('2021-01-01')

rfm_dataset = df_dataset.groupby('_CustomerID').agg ({
    'OrderDate' : lambda v : (today_date - v.max()).days,
    'OrderNumber' : 'count',
    'Revenue' : 'sum'
})

rfm_dataset.rename(
    columns= {
        'OrderDate' : 'Recency',
        'OrderNumber' : 'Frequency',
        'Revenue' : 'Monetary'
    },
    inplace= True
)

r = pd.qcut(rfm_dataset['Recency'], q = 5, labels=range(5,0,-1))
f = pd.qcut(rfm_dataset['Frequency'], q = 5, labels=range(1,6))
m = pd.qcut(rfm_dataset['Monetary'], q = 5, labels=range(1,6))

def segment(value):
        if value == '555' or value == '554'  or value == '544' or value ==␣
 ↪'545' or value == '454' or value == '455' or value == '445':
                return 'Champions'
        elif value == '543' or  value == '444' or value == '435' or value ==␣
 ↪'355' or value == '354' or value == '345' or value == '344' or value ==␣
 ↪'335':
```

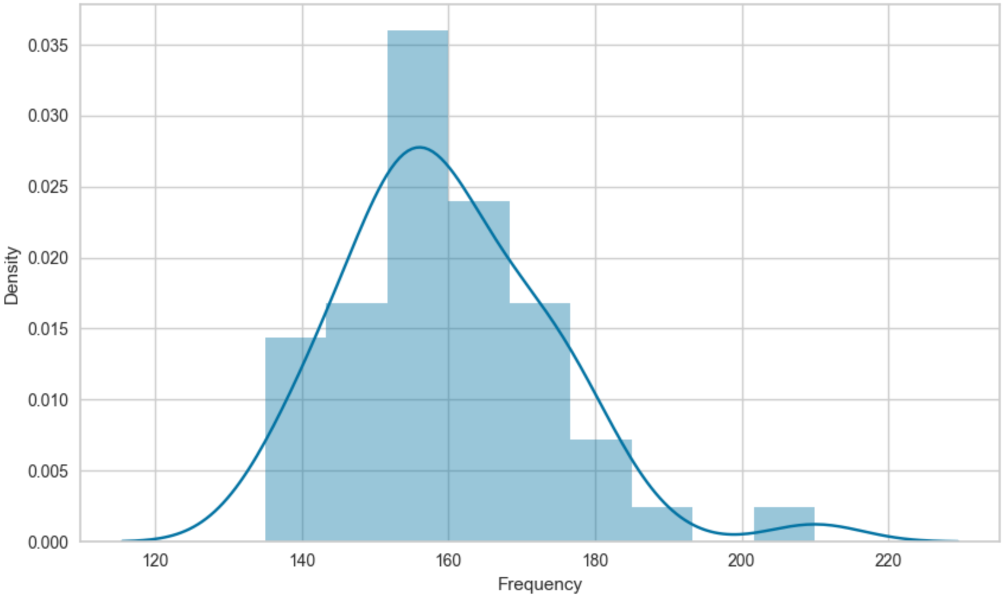
```
                return 'Loyal'
        elif value == '553' or  value == '551' or value == '552' or value ==␣
↪'541' or value == '542' or value == '533' or value == '532' or value ==␣
↪'531' or value == '452' or  value == '451' or value == '442' or value ==␣
↪'441' or value == '431' or value == '453' or value == '433' or value ==␣
↪'432' or value == '423' or  value == '353' or value == '352' or value ==␣
↪'351' or value == '342' or value == '341' or value == '333' or value ==␣
↪'323':
                return 'Potential Loyalist'
        elif value == '512' or  value == '511' or value == '422' or value ==␣
↪'421' or value == '412' or value == '411' or value == '311':
                return 'New Customers'
        elif value == '525' or  value == '524' or value == '523' or value ==␣
↪'522' or value == '521' or value == '515' or value == '514' or value ==␣
↪'513' or value == '425' or  value == '424' or value == '413' or value ==␣
↪'414' or value == '415' or value == '315' or value == '314' or value ==␣
↪'313':
                return 'Promising'
        elif value == '535' or  value == '534' or value == '443' or value ==␣
↪'434' or value == '343' or value == '334' or value == '325' or value ==␣
↪'324':
                return 'Need Attention'
        elif value == '331' or  value == '321' or value == '312' or value ==␣
↪'221' or value == '213' or value == '231' or value == '241' or value ==␣
↪'251':
                return 'About To Sleep'
        elif value == '255' or  value == '254' or value == '245' or value ==␣
↪'244' or value == '253' or value == '252' or value == '243' or value ==␣
↪'242' or value == '235' or  value == '234' or value == '225' or value ==␣
↪'224' or value == '153' or value == '152' or value == '145' or value ==␣
↪'143' or value == '142' or  value == '135' or value == '134' or value ==␣
↪'133' or value == '125' or value == '124':
                return "At Risk"
        elif value == '155' or  value == '154' or value == '144' or value ==␣
↪'214' or value == '215' or value == '115' or value == '114' or value ==␣
↪'113':
                return 'Cannot Lose Them'
        elif value == '332' or  value == '322' or value == '233' or value ==␣
↪'232' or value == '223' or value == '222' or value == '132' or value ==␣
↪'123' or value == '122' or value == '212' or value == '211':
                return 'Hibernating Customers'
        else:
                return 'Lost Customers'
```

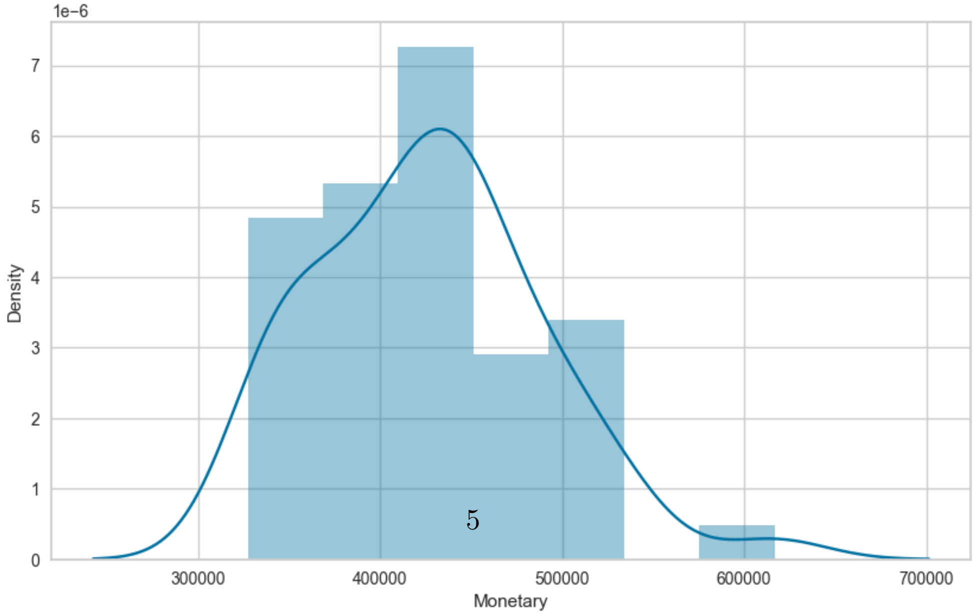
[5]:

```
fig, ax = plt.subplots(3,1, figsize=(10,20))
sns.distplot(rfm_dataset['Recency'], ax = ax[0])
sns.distplot(rfm_dataset['Frequency'], ax = ax[1])
sns.distplot(rfm_dataset['Monetary'], ax = ax[2])
plt.show()
```

```
C:\Users\ADMIN\AppData\Local\Temp\ipykernel_24684\3217253163.py:2: UserWarning:

`distplot` is a deprecated function and will be removed in seaborn v0.14.0.

Please adapt your code to use either `displot` (a figure-level function with
similar flexibility) or `histplot` (an axes-level function for histograms).

For a guide to updating your code to use the new functions, please see
https://gist.github.com/mwaskom/de44147ed2974457ad6372750bbe5751

  sns.distplot(rfm_dataset['Recency'], ax = ax[0])
C:\Users\ADMIN\AppData\Local\Temp\ipykernel_24684\3217253163.py:3: UserWarning:

`distplot` is a deprecated function and will be removed in seaborn v0.14.0.

Please adapt your code to use either `displot` (a figure-level function with
similar flexibility) or `histplot` (an axes-level function for histograms).

For a guide to updating your code to use the new functions, please see
https://gist.github.com/mwaskom/de44147ed2974457ad6372750bbe5751

  sns.distplot(rfm_dataset['Frequency'], ax = ax[1])
C:\Users\ADMIN\AppData\Local\Temp\ipykernel_24684\3217253163.py:4: UserWarning:

`distplot` is a deprecated function and will be removed in seaborn v0.14.0.

Please adapt your code to use either `displot` (a figure-level function with
similar flexibility) or `histplot` (an axes-level function for histograms).

For a guide to updating your code to use the new functions, please see
https://gist.github.com/mwaskom/de44147ed2974457ad6372750bbe5751

  sns.distplot(rfm_dataset['Monetary'], ax = ax[2])
```


0.16
0.14
0.12
0.10
0.08
0.06
0.04
0.02
0.00
Density
0
10
20
30
Recency


0.035
0.030
0.025
0.020
0.015
0.010
0.005
0.000
Density
120
140
160
180
200
220
Frequency


1e−6
7
6
5
4
3
2
1
0
Density
300000
400000
500000
600000
700000
Monetary

[6]:
```
# Tạo cột mới, sử dụng phương thức assign()
rfm = rfm_dataset.assign(R=r.values, F=f.values, M=m.values)
rfm['rfm_group'] = rfm['R'].astype(str) + rfm['F'].astype(str) + rfm['M'].
 ↪astype(str)
rfm['Class']=rfm['rfm_group'].apply(segment)
```

[7]:
```
rfm
```

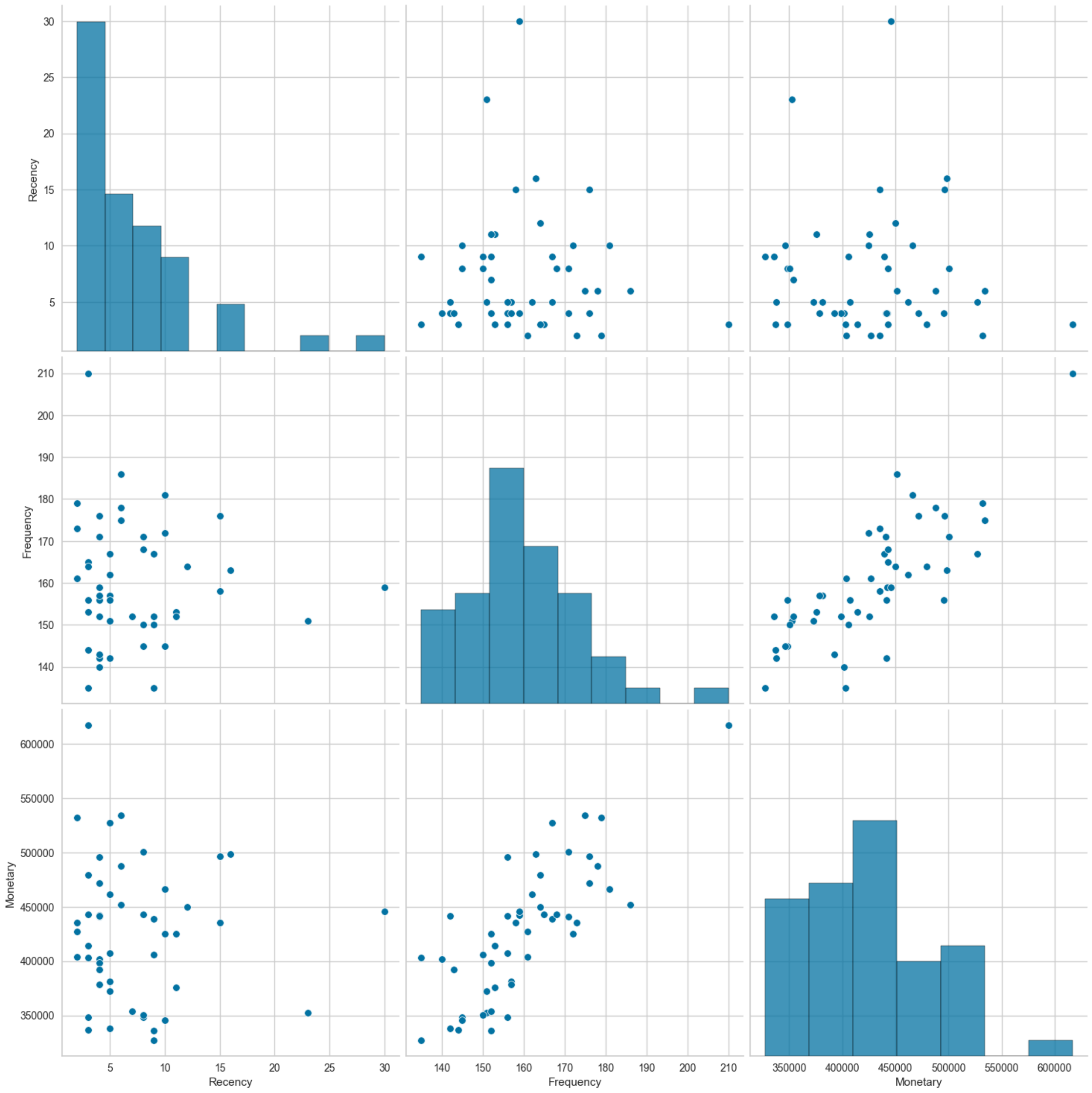
[7]:
```
             Recency  Frequency     Monetary  R  F  M rfm_group  \
_CustomerID
1                  9        152  335933.6115  2  2  1       221
10                15        158  435122.1870  1  3  3       133
11                 6        178  487614.2415  3  5  5       355
12                 3        210  616719.2550  5  5  5       555
13                 4        171  441003.2795  4  4  3       443
14                 5        157  381450.0280  3  3  2       332
15                 4        142  441668.3550  4  1  4       414
16                 3        135  402938.7705  5  1  2       512
17                 6        175  534027.3860  3  5  5       355
18                 6        186  451637.7540  3  5  4       354
19                 3        165  443231.8335  5  4  4       544
2                  9        135  327409.1345  2  1  1       211
20                 9        167  439147.9490  2  4  3       243
21                 3        164  479383.0905  5  4  5       545
22                 4        140  401721.7825  4  1  2       412
23                12        164  449782.2895  1  4  4       144
24                23        151  352505.5255  1  2  1       121
25                 5        162  461601.9940  3  4  4       344
26                11        153  375766.6860  1  2  2       122
27                 3        144  336959.7835  5  1  1       511
28                 8        145  348495.4750  2  1  1       211
29                 2        179  531770.6920  5  5  5       555
3                 10        181  466220.1365  2  5  4       254
30                 4        159  442372.3910  4  3  4       434
31                 4        152  398616.8350  4  2  2       422
32                 2        173  435206.0710  5  5  3       553
33                 4        156  495444.2635  4  2  5       425
34                15        176  496418.6780  1  5  5       155
35                10        145  345844.5865  2  1  1       211
36                 4        156  441225.9540  4  2  3       423
37                11        152  425241.0940  1  2  3       123
38                 8        150  350687.6480  2  1  1       211
39                 4        176  471608.1425  4  5  4       454
4                  5        167  526981.0630  3  4  5       345
40                 9        150  406016.9180  2  1  2       212
```

```
41                 2        161  403693.2910  5  3  2       532
42                 2        161  427159.6055  5  3  3       533
43                 5        151  372471.3245  3  2  2       322
44                 3        156  348209.0500  5  2  1       521
45                 5        156  407145.0305  3  2  3       323
46                 4        157  378414.7270  4  3  2       432
47                 8        168  442845.8465  2  4  4       244
48                10        172  424970.7490  2  5  3       253
49                 7        152  354161.3300  2  2  1       221
5                 30        159  445632.7450  1  3  4       134
50                16        163  498384.9940  1  4  5       145
6                  4        143  392141.0840  4  1  2       412
7                  3        153  414155.9775  5  2  3       523
8                  5        142  338000.0590  3  1  1       311
9                  8        171  500166.4570  2  4  5       245

                             Class
_CustomerID
1                   About To Sleep
10                         At Risk
11                           Loyal
12                       Champions
13                  Need Attention
14           Hibernating Customers
15                       Promising
16                   New Customers
17                           Loyal
18                           Loyal
19                       Champions
2            Hibernating Customers
20                         At Risk
21                       Champions
22                   New Customers
23                Cannot Lose Them
24                  Lost Customers
25                           Loyal
26           Hibernating Customers
27                   New Customers
28           Hibernating Customers
29                       Champions
3                          At Risk
30                  Need Attention
31                   New Customers
32              Potential Loyalist
33                       Promising
34                Cannot Lose Them
35           Hibernating Customers
```

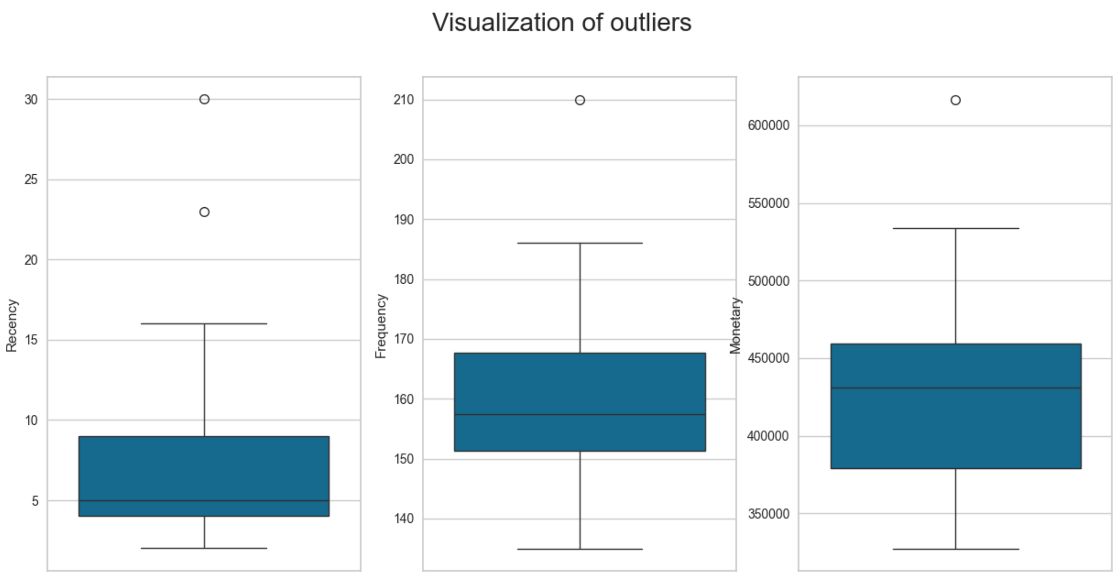
```
36             Potential Loyalist
37          Hibernating Customers
38          Hibernating Customers
39                      Champions
4                           Loyal
40          Hibernating Customers
41             Potential Loyalist
42             Potential Loyalist
43          Hibernating Customers
44                      Promising
45             Potential Loyalist
46             Potential Loyalist
47                        At Risk
48                        At Risk
49                 About To Sleep
5                         At Risk
50                        At Risk
6                   New Customers
7                       Promising
8                   New Customers
9                         At Risk
```

[8]:
```
sns.pairplot(rfm[['Recency','Frequency','Monetary']], height=5)
```

[8]:
```
<seaborn.axisgrid.PairGrid at 0x237f26f9f50>
```



[9]:
```
fig = plt.figure(figsize = (15,7))
fig.suptitle("Visualization of outliers",size=20)

axes = fig.add_subplot(1, 3, 1)
sns.boxplot(data=rfm,y="Recency")

axes = fig.add_subplot(1, 3, 2)
sns.boxplot(data=rfm,y="Frequency")

axes = fig.add_subplot(1, 3, 3)
sns.boxplot(data=rfm,y="Monetary")
plt.show()
```



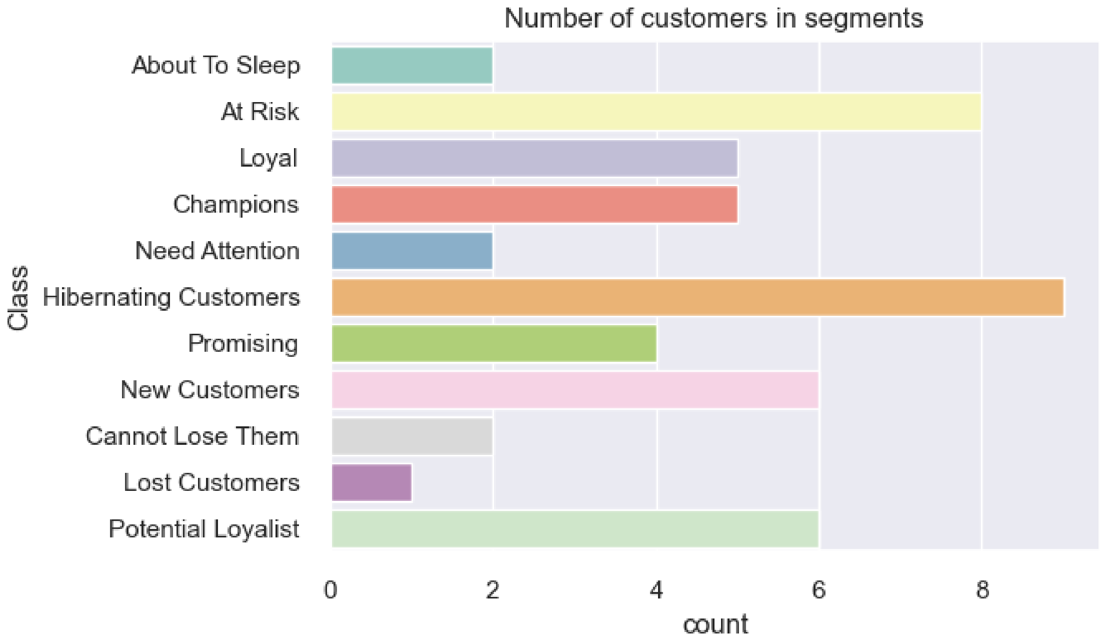
```
[10]: customer_segment = rfm.groupby('Class').agg({
          'Recency': 'mean',
          'Frequency': 'mean',
          'Monetary': ['mean', 'count'] }).round(1)
      customer_segment.columns = ['Mean_R','Mean_F','Mean_M','Count']
      customer_segment = customer_segment.sort_values(by= 'Count')
```

```
[11]: customer_segment
```

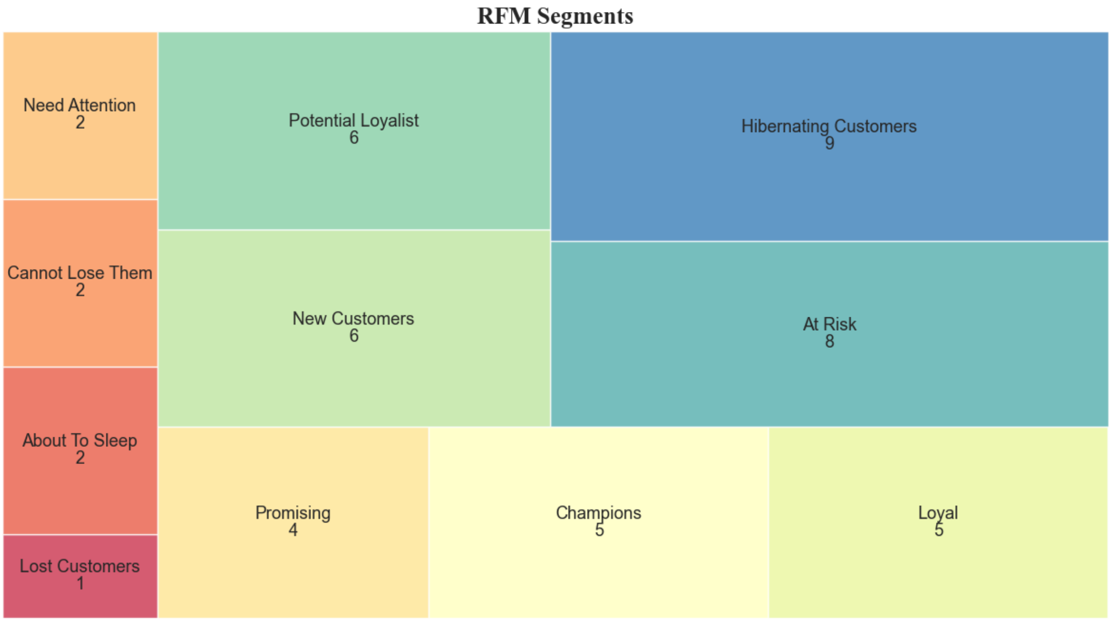
```
[11]:                        Mean_R  Mean_F    Mean_M  Count
      Class
      Lost Customers           23.0   151.0  352505.5      1
      About To Sleep            8.0   152.0  345047.5      2
      Cannot Lose Them         13.5   170.0  473100.5      2
      Need Attention            4.0   165.0  441687.8      2
      Promising                 3.5   151.8  424869.4      4
      Champions                 3.0   178.8  508542.6      5
      Loyal                     5.6   173.6  492372.5      5
      New Customers             3.8   142.7  378396.4      6
      Potential Loyalist        3.2   160.7  415474.1      6
      At Risk                  13.2   167.4  456561.4      8
      Hibernating Customers     8.4   148.7  370375.9      9
```

```
[12]: sns.set(style="darkgrid", rc={'figure.figsize': (6, 4)})
      ax = sns.countplot(y="Class", data=rfm, palette="Set3")
      ax.set_title('Number of customers in segments')
      plt.show()
```

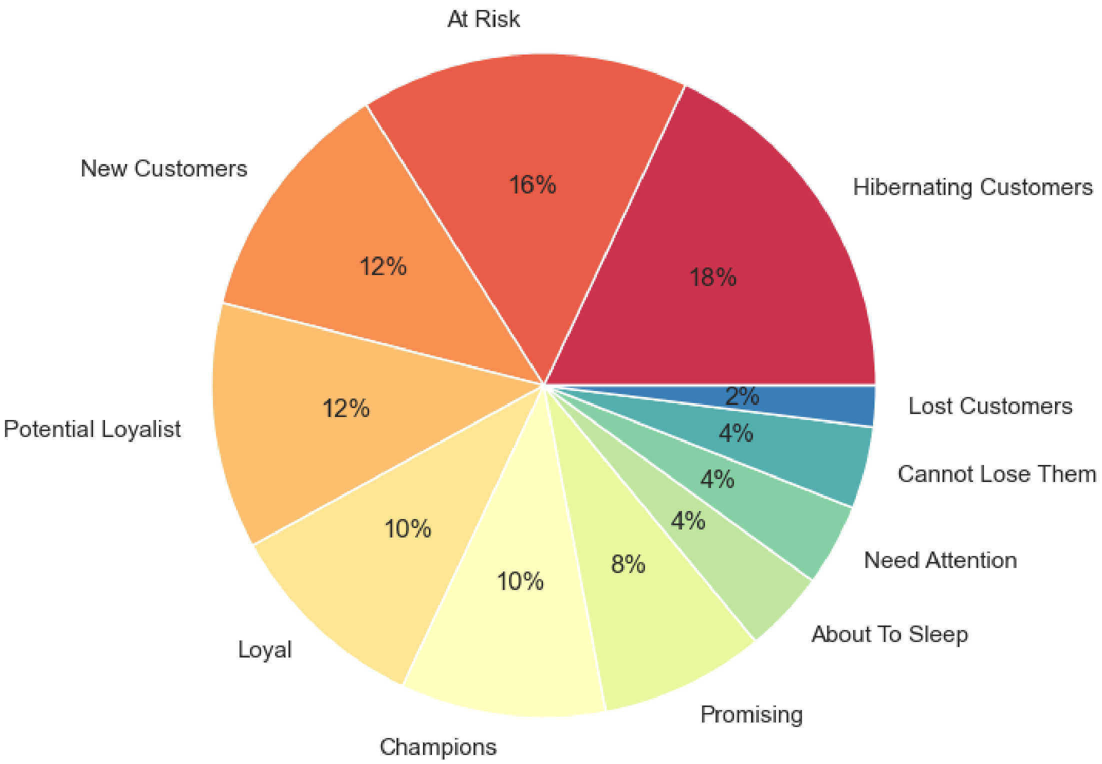
```
C:\Users\ADMIN\AppData\Local\Temp\ipykernel_24684\485670957.py:2: FutureWarning:

Passing `palette` without assigning `hue` is deprecated and will be removed in
v0.14.0. Assign the `y` variable to `hue` and set `legend=False` for the same
effect.

  ax = sns.countplot(y="Class", data=rfm, palette="Set3")
```



[13]:
```
fig = plt.gcf()
ax = fig.add_subplot()
fig.set_size_inches(15, 8)
squarify.plot(sizes=customer_segment.Count,
              label=customer_segment.index,
              value=customer_segment.Count,
              alpha=0.8,
              text_kwargs={'fontsize': 13},
              color=sns.color_palette("Spectral", len(customer_segment.index)))
plt.title("RFM Segments",fontsize=18,fontweight="bold",fontfamily='Times New␣
 ↪Roman')
plt.axis('off')
plt.show()
```



[14]:
```
plt.figure(figsize = (7,7))
plt.pie(rfm.Class.value_counts(),
        labels=rfm.Class.value_counts().index,
        autopct='%.0f%%',
        colors=sns.color_palette("Spectral", len(customer_segment.index)))
plt.show()
```



```
[15]: rfm1 = rfm[['Recency','Frequency','Monetary', 'Class']]
      rfm1
```

```
[15]:              Recency  Frequency     Monetary                  Class
      _CustomerID
      1                  9        152  335933.6115         About To Sleep
      10                15        158  435122.1870                At Risk
      11                 6        178  487614.2415                  Loyal
      12                 3        210  616719.2550              Champions
      13                 4        171  441003.2795         Need Attention
      14                 5        157  381450.0280  Hibernating Customers
      15                 4        142  441668.3550              Promising
      16                 3        135  402938.7705          New Customers
      17                 6        175  534027.3860                  Loyal
      18                 6        186  451637.7540                  Loyal
      19                 3        165  443231.8335              Champions
      2                  9        135  327409.1345  Hibernating Customers
      20                 9        167  439147.9490                At Risk
      21                 3        164  479383.0905              Champions
      22                 4        140  401721.7825          New Customers
```

```
23                12        164  449782.2895       Cannot Lose Them
24                23        151  352505.5255         Lost Customers
25                 5        162  461601.9940                  Loyal
26                11        153  375766.6860  Hibernating Customers
27                 3        144  336959.7835          New Customers
28                 8        145  348495.4750  Hibernating Customers
29                 2        179  531770.6920              Champions
3                 10        181  466220.1365                At Risk
30                 4        159  442372.3910         Need Attention
31                 4        152  398616.8350          New Customers
32                 2        173  435206.0710     Potential Loyalist
33                 4        156  495444.2635              Promising
34                15        176  496418.6780       Cannot Lose Them
35                10        145  345844.5865  Hibernating Customers
36                 4        156  441225.9540     Potential Loyalist
37                11        152  425241.0940  Hibernating Customers
38                 8        150  350687.6480  Hibernating Customers
39                 4        176  471608.1425              Champions
4                  5        167  526981.0630                  Loyal
40                 9        150  406016.9180  Hibernating Customers
41                 2        161  403693.2910     Potential Loyalist
42                 2        161  427159.6055     Potential Loyalist
43                 5        151  372471.3245  Hibernating Customers
44                 3        156  348209.0500              Promising
45                 5        156  407145.0305     Potential Loyalist
46                 4        157  378414.7270     Potential Loyalist
47                 8        168  442845.8465                At Risk
48                10        172  424970.7490                At Risk
49                 7        152  354161.3300         About To Sleep
5                 30        159  445632.7450                At Risk
50                16        163  498384.9940                At Risk
6                  4        143  392141.0840          New Customers
7                  3        153  414155.9775              Promising
8                  5        142  338000.0590          New Customers
9                  8        171  500166.4570                At Risk
```

[16]:

```
rfm_final = pd.DataFrame()
rfm_final['Class'] = rfm1['Class']
rfm_final['Recency'] = stats.boxcox(rfm1['Recency'])[0]
rfm_final['Frequency'] = stats.boxcox(rfm1['Frequency'])[0]
rfm_final['Monetary'] = pd.Series(np.cbrt(rfm1['Monetary'])).values
rfm_final
```

[16]:

```
                             Class   Recency  Frequency   Monetary
_CustomerID
1                   About To Sleep  1.671489   0.566399  69.515954
10                         At Risk  1.940735   0.566404  75.776942
```

```
11                    Loyal  1.430652   0.566418  78.709193
12                Champions  0.954935   0.566433  85.119521
13           Need Attention  1.162885   0.566414  76.116815
14     Hibernating Customers  1.313753   0.566403  72.523577
15                Promising  1.162885   0.566389  76.155059
16            New Customers  0.954935   0.566380  73.860632
17                    Loyal  1.430652   0.566416  81.131189
18                    Loyal  1.430652   0.566423  76.723796
19                Champions  0.954935   0.566410  76.244815
2      Hibernating Customers  1.671489   0.566380  68.922909
20                  At Risk  1.671489   0.566411  76.009922
21                Champions  0.954935   0.566409  78.263795
22            New Customers  1.162885   0.566386  73.786197
23          Cannot Lose Them  1.827529   0.566409  76.618583
24           Lost Customers  2.139986   0.566398  70.640751
25                    Loyal  1.313753   0.566407  77.283935
26     Hibernating Customers  1.781566   0.566400  72.161590
27            New Customers  0.954935   0.566391  69.586665
28     Hibernating Customers  1.604139   0.566392  70.371863
29                Champions  0.633977   0.566419  81.016747
3                   At Risk  1.730003   0.566420  77.540812
30           Need Attention  1.162885   0.566405  76.195503
31            New Customers  1.162885   0.566399  73.595605
32        Potential Loyalist  0.633977   0.566415  75.781811
33                Promising  1.162885   0.566402  79.128257
34          Cannot Lose Them  1.940735   0.566417  79.180099
35     Hibernating Customers  1.730003   0.566392  70.192977
36        Potential Loyalist  1.162885   0.566402  76.129624
37     Hibernating Customers  1.781566   0.566399  75.198944
38     Hibernating Customers  1.604139   0.566397  70.519110
39                Champions  1.162885   0.566417  77.838376
4                     Loyal  1.313753   0.566411  80.772775
40     Hibernating Customers  1.671489   0.566397  74.048235
41        Potential Loyalist  0.633977   0.566406  73.906706
42        Potential Loyalist  0.633977   0.566406  75.311863
43     Hibernating Customers  1.313753   0.566398  71.950025
44                Promising  0.954935   0.566402  70.352578
45        Potential Loyalist  1.313753   0.566402  74.116752
46        Potential Loyalist  1.162885   0.566403  72.330701
47                  At Risk  1.604139   0.566412  76.222676
48                  At Risk  1.730003   0.566414  75.183005
49           About To Sleep  1.525234   0.566399  70.751184
5                   At Risk  2.253078   0.566405  76.382236
50                  At Risk  1.972265   0.566408  79.284505
6             New Customers  1.162885   0.566390  73.194893
7                 Promising  0.954935   0.566400  74.539758
8             New Customers  1.313753   0.566389  69.658202
```

```
9                          At Risk  1.604139   0.566414  79.378859
```

[17]:

```
numerical_columns = rfm_final[['Recency', 'Frequency', 'Monetary']]

std_scaler = StandardScaler()


df_scaled = std_scaler.fit_transform(numerical_columns)


df_scaled = pd.DataFrame(df_scaled, columns=numerical_columns.columns)
```

[18]:

```
df_scaled
```

[18]:

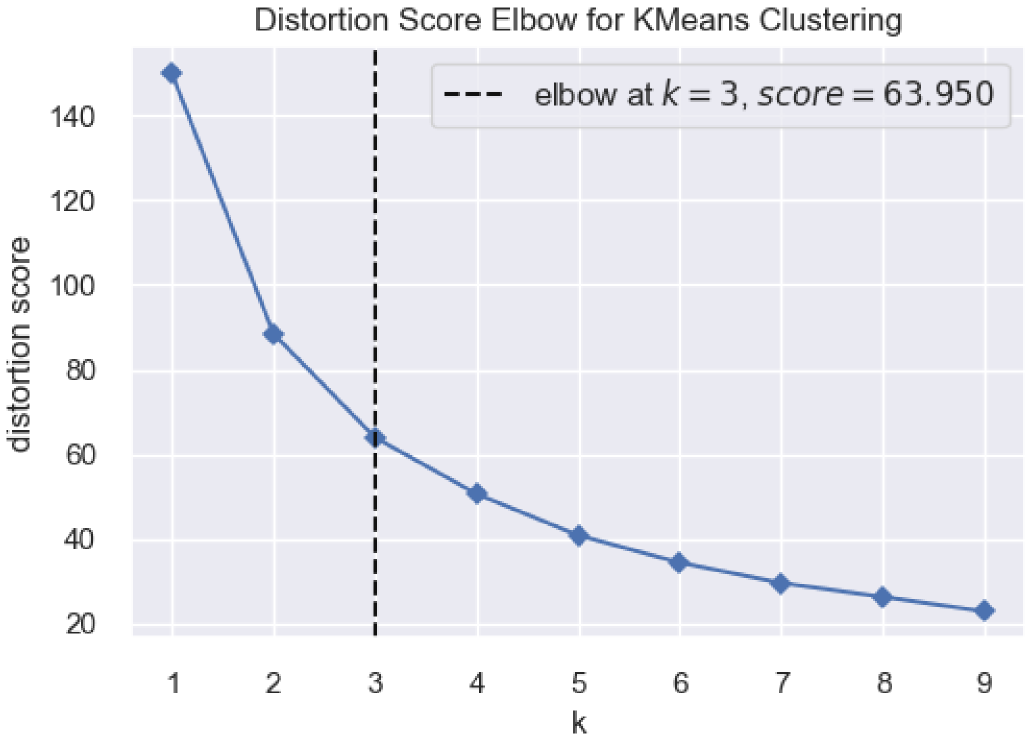
```
     Recency  Frequency  Monetary
0   0.766226  -0.496835 -1.570399
1   1.449556  -0.015878  0.188949
2   0.154997   1.274313  1.012916
3  -1.052342   2.668860  2.814229
4  -0.524577   0.869959  0.284454
5  -0.141686  -0.092520 -0.725252
6  -0.524577  -1.426199  0.295200
7  -1.052342  -2.192545 -0.349537
8   0.154997   1.106491  1.693501
9   0.154997   1.685724  0.455016
10 -1.052342   0.485066  0.320422
11  0.766226  -2.192545 -1.737046
12  0.766226   0.617619  0.254416
13 -1.052342   0.417108  0.887759
14 -0.524577  -1.634374 -0.370454
15  1.162246   0.417108  0.425451
16  1.955240  -0.582166 -1.254329
17 -0.141686   0.277698  0.612416
18  1.045593  -0.413042 -0.826971
19 -1.052342  -1.225977 -1.550529
20  0.595297  -1.128720 -1.329887
21 -1.866913   1.328531  1.661342
22  0.914732   1.434492  0.684599
23 -0.524577   0.059434  0.306565
24 -0.524577  -0.496835 -0.424011
25 -1.866913   0.990115  0.190317
26 -0.524577  -0.170523  1.130674
27  1.449556   1.163311  1.145241
28  0.914732  -1.128720 -1.380155
29 -0.524577  -0.170523  0.288053
30  1.045593  -0.496835  0.026530
31  0.595297  -0.669075 -1.288511
```

```
32 -0.524577   1.163311  0.768215
33 -0.141686   0.617619  1.592786
34  0.766226  -0.669075 -0.296821
35 -1.866913   0.206192 -0.336591
36 -1.866913   0.206192  0.058261
37 -0.141686  -0.582166 -0.886421
38 -1.052342  -0.170523 -1.335306
39 -0.141686  -0.170523 -0.277567
40 -0.524577  -0.092520 -0.779450
41  0.595297   0.682264  0.314201
42  0.914732   0.930520  0.022051
43  0.395039  -0.496835 -1.223297
44  2.242260   0.059434  0.359037
45  1.529576   0.347994  1.174580
46 -0.524577  -1.325120 -0.536611
47 -1.052342  -0.413042 -0.158702
48 -0.141686  -1.426199 -1.530427
49  0.595297   0.869959  1.201093
```

[19]:
```
model = KMeans()
visualizer = KElbowVisualizer(model, k=(1,10), timings= False)
visualizer.fit(df_scaled)
visualizer.show()
```

```
c:\Users\ADMIN\AppData\Local\Programs\Python\Python311\Lib\site-
packages\sklearn\cluster\_kmeans.py:1416: FutureWarning: The default value of
`n_init` will change from 10 to 'auto' in 1.4. Set the value of `n_init`
explicitly to suppress the warning
  super()._check_params_vs_input(X, default_n_init=10)
c:\Users\ADMIN\AppData\Local\Programs\Python\Python311\Lib\site-
packages\sklearn\cluster\_kmeans.py:1416: FutureWarning: The default value of
`n_init` will change from 10 to 'auto' in 1.4. Set the value of `n_init`
explicitly to suppress the warning
  super()._check_params_vs_input(X, default_n_init=10)
c:\Users\ADMIN\AppData\Local\Programs\Python\Python311\Lib\site-
packages\sklearn\cluster\_kmeans.py:1416: FutureWarning: The default value of
`n_init` will change from 10 to 'auto' in 1.4. Set the value of `n_init`
explicitly to suppress the warning
  super()._check_params_vs_input(X, default_n_init=10)
c:\Users\ADMIN\AppData\Local\Programs\Python\Python311\Lib\site-
packages\sklearn\cluster\_kmeans.py:1416: FutureWarning: The default value of
`n_init` will change from 10 to 'auto' in 1.4. Set the value of `n_init`
explicitly to suppress the warning
  super()._check_params_vs_input(X, default_n_init=10)
c:\Users\ADMIN\AppData\Local\Programs\Python\Python311\Lib\site-
packages\sklearn\cluster\_kmeans.py:1416: FutureWarning: The default value of
`n_init` will change from 10 to 'auto' in 1.4. Set the value of `n_init`
```

```
explicitly to suppress the warning
  super()._check_params_vs_input(X, default_n_init=10)
c:\Users\ADMIN\AppData\Local\Programs\Python\Python311\Lib\site-
packages\sklearn\cluster\_kmeans.py:1416: FutureWarning: The default value of
`n_init` will change from 10 to 'auto' in 1.4. Set the value of `n_init`
explicitly to suppress the warning
  super()._check_params_vs_input(X, default_n_init=10)
c:\Users\ADMIN\AppData\Local\Programs\Python\Python311\Lib\site-
packages\sklearn\cluster\_kmeans.py:1416: FutureWarning: The default value of
`n_init` will change from 10 to 'auto' in 1.4. Set the value of `n_init`
explicitly to suppress the warning
  super()._check_params_vs_input(X, default_n_init=10)
c:\Users\ADMIN\AppData\Local\Programs\Python\Python311\Lib\site-
packages\sklearn\cluster\_kmeans.py:1416: FutureWarning: The default value of
`n_init` will change from 10 to 'auto' in 1.4. Set the value of `n_init`
explicitly to suppress the warning
  super()._check_params_vs_input(X, default_n_init=10)
c:\Users\ADMIN\AppData\Local\Programs\Python\Python311\Lib\site-
packages\sklearn\cluster\_kmeans.py:1416: FutureWarning: The default value of
`n_init` will change from 10 to 'auto' in 1.4. Set the value of `n_init`
explicitly to suppress the warning
  super()._check_params_vs_input(X, default_n_init=10)
```



```
[19]: <Axes: title={'center': 'Distortion Score Elbow for KMeans Clustering'},
      xlabel='k', ylabel='distortion score'>
```

```
[20]: kmeans = KMeans(n_clusters=3, random_state=1)
      kmeans.fit(df_scaled)
      cluster_labels = kmeans.labels_
      centroids = kmeans.cluster_centers_
      centroid_df = pd.DataFrame(centroids, columns = list(df_scaled) )
      centroid_df
```

```
c:\Users\ADMIN\AppData\Local\Programs\Python\Python311\Lib\site-
packages\sklearn\cluster\_kmeans.py:1416: FutureWarning: The default value of
`n_init` will change from 10 to 'auto' in 1.4. Set the value of `n_init`
explicitly to suppress the warning
  super()._check_params_vs_input(X, default_n_init=10)
```

```
[20]:     Recency  Frequency  Monetary
      0  0.937861   0.719751  0.639827
      1  0.069201  -0.886310 -0.922752
      2 -0.972219   0.569066  0.694680
```

```
[21]: df_new = rfm1.assign(Cluster = cluster_labels)
```

```
[22]: df_new
```

```
[22]:              Recency  Frequency     Monetary                 Class  Cluster
      _CustomerID
      1                  9        152  335933.6115       About To Sleep        1
      10                15        158  435122.1870              At Risk        0
      11                 6        178  487614.2415                Loyal        0
      12                 3        210  616719.2550            Champions        2
      13                 4        171  441003.2795       Need Attention        2
      14                 5        157  381450.0280  Hibernating Customers        1
      15                 4        142  441668.3550            Promising        1
      16                 3        135  402938.7705        New Customers        1
      17                 6        175  534027.3860                Loyal        0
      18                 6        186  451637.7540                Loyal        0
      19                 3        165  443231.8335            Champions        2
      2                  9        135  327409.1345  Hibernating Customers        1
      20                 9        167  439147.9490              At Risk        0
      21                 3        164  479383.0905            Champions        2
      22                 4        140  401721.7825        New Customers        1
      23                12        164  449782.2895      Cannot Lose Them        0
      24                23        151  352505.5255       Lost Customers        1
      25                 5        162  461601.9940                Loyal        2
      26                11        153  375766.6860  Hibernating Customers        1
      27                 3        144  336959.7835        New Customers        1
```

```
28                 8        145  348495.4750  Hibernating Customers         1
29                 2        179  531770.6920              Champions         2
3                 10        181  466220.1365                At Risk         0
30                 4        159  442372.3910         Need Attention         2
31                 4        152  398616.8350          New Customers         1
32                 2        173  435206.0710     Potential Loyalist         2
33                 4        156  495444.2635              Promising         2
34                15        176  496418.6780       Cannot Lose Them         0
35                10        145  345844.5865  Hibernating Customers         1
36                 4        156  441225.9540     Potential Loyalist         2
37                11        152  425241.0940  Hibernating Customers         0
38                 8        150  350687.6480  Hibernating Customers         1
39                 4        176  471608.1425              Champions         2
4                  5        167  526981.0630                  Loyal         2
40                 9        150  406016.9180  Hibernating Customers         1
41                 2        161  403693.2910     Potential Loyalist         2
42                 2        161  427159.6055     Potential Loyalist         2
43                 5        151  372471.3245  Hibernating Customers         1
44                 3        156  348209.0500              Promising         1
45                 5        156  407145.0305     Potential Loyalist         1
46                 4        157  378414.7270     Potential Loyalist         1
47                 8        168  442845.8465                At Risk         0
48                10        172  424970.7490                At Risk         0
49                 7        152  354161.3300         About To Sleep         1
5                 30        159  445632.7450                At Risk         0
50                16        163  498384.9940                At Risk         0
6                  4        143  392141.0840          New Customers         1
7                  3        153  414155.9775              Promising         2
8                  5        142  338000.0590          New Customers         1
9                  8        171  500166.4570                At Risk         0
```

[23]:

```
df_result = df_new.groupby(['Cluster']).agg({
        'Recency': 'mean',
        'Frequency': 'mean',
        'Monetary': ['mean', 'count']
    }).round(2)
df_result
```

[23]:

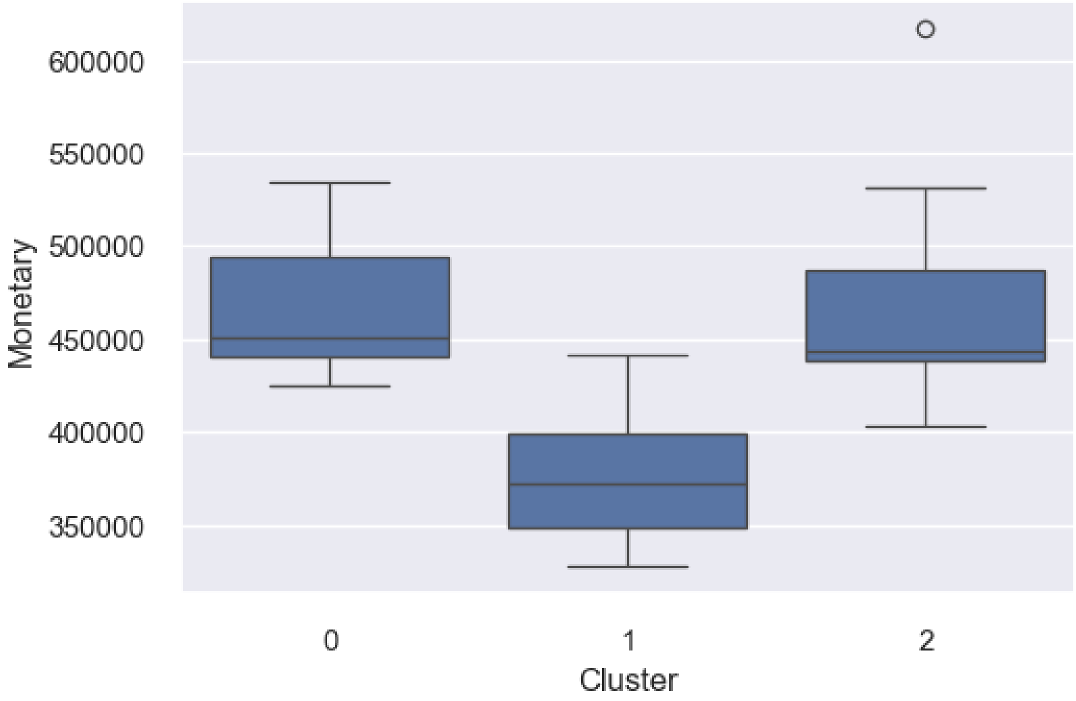
```
        Recency Frequency   Monetary
           mean      mean       mean count
Cluster
0         11.57    169.29  464086.61    14
1          6.81    148.00  371264.65    21
2          3.33    167.53  468770.46    15
```

[24]:

```
# Box plot to visualize Cluster Id vs Frequency
```

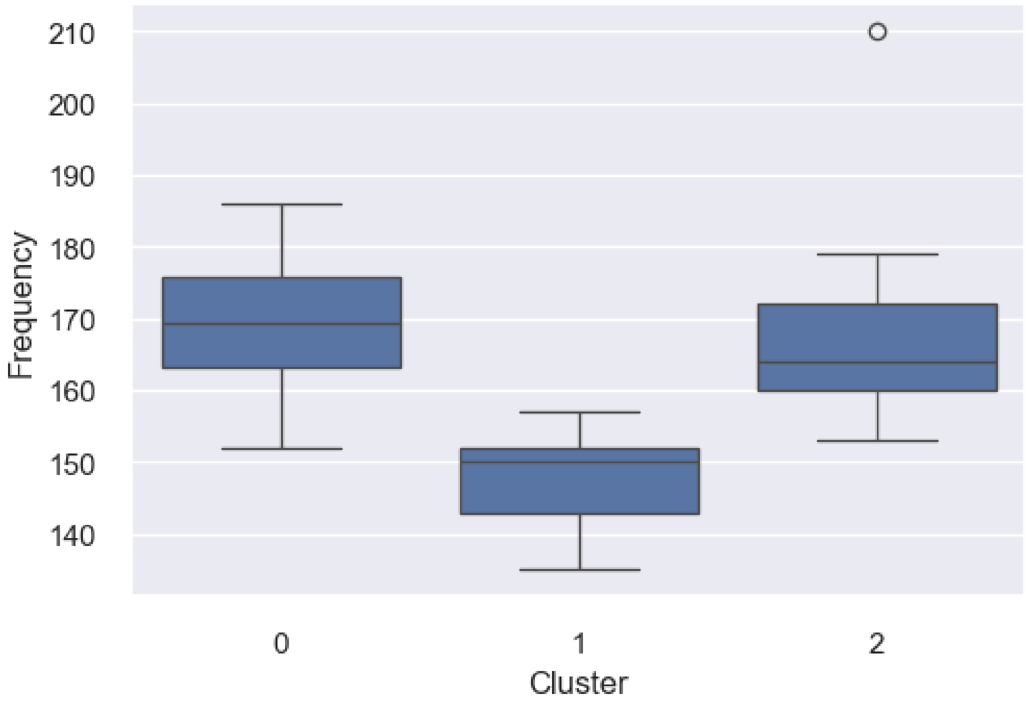
```
sns.boxplot(x='Cluster', y='Monetary', data=df_new)
```

[24]:
```
<Axes: xlabel='Cluster', ylabel='Monetary'>
```



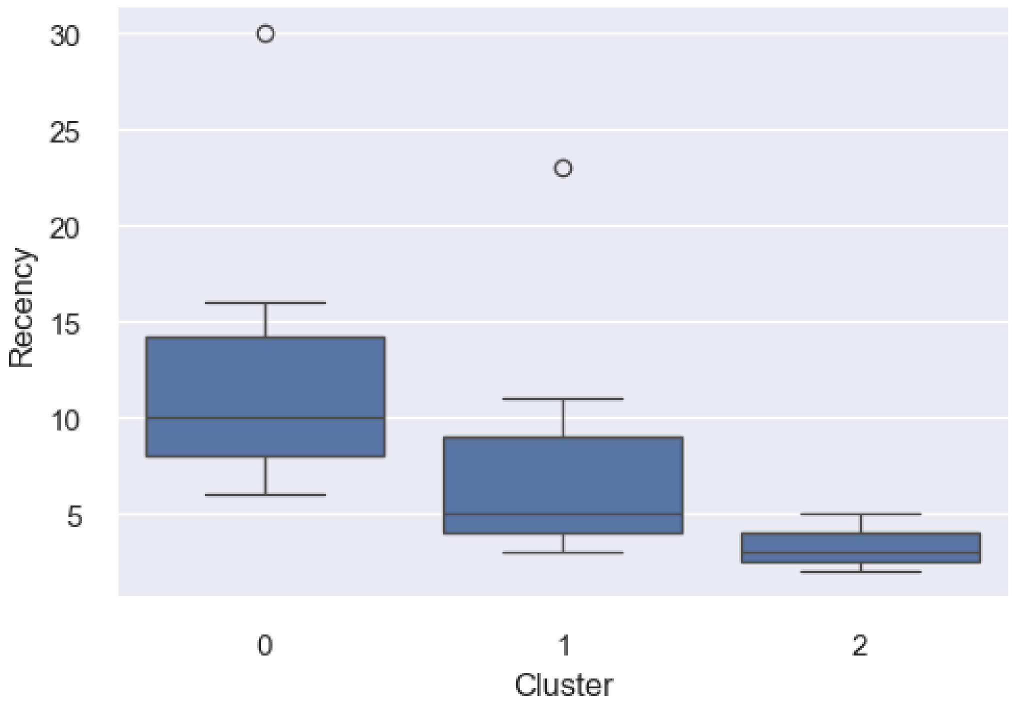
[25]:
```
# Box plot to visualize Cluster Id vs Frequency

sns.boxplot(x='Cluster', y='Frequency', data=df_new)
```

[25]:
```
<Axes: xlabel='Cluster', ylabel='Frequency'>
```



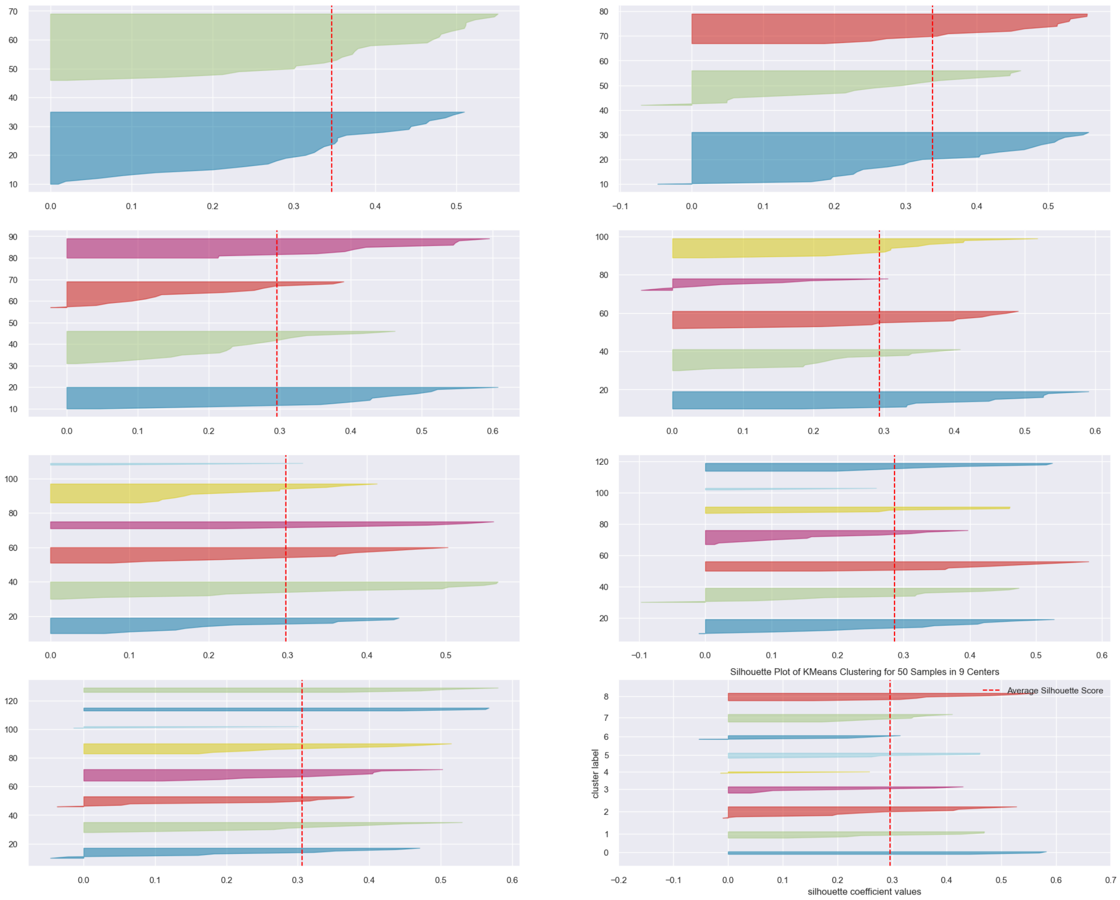
[26]:
```
# Box plot to visualize Cluster Id vs Recency

sns.boxplot(x='Cluster', y='Recency', data=df_new)
```

[26]:
```
<Axes: xlabel='Cluster', ylabel='Recency'>
```



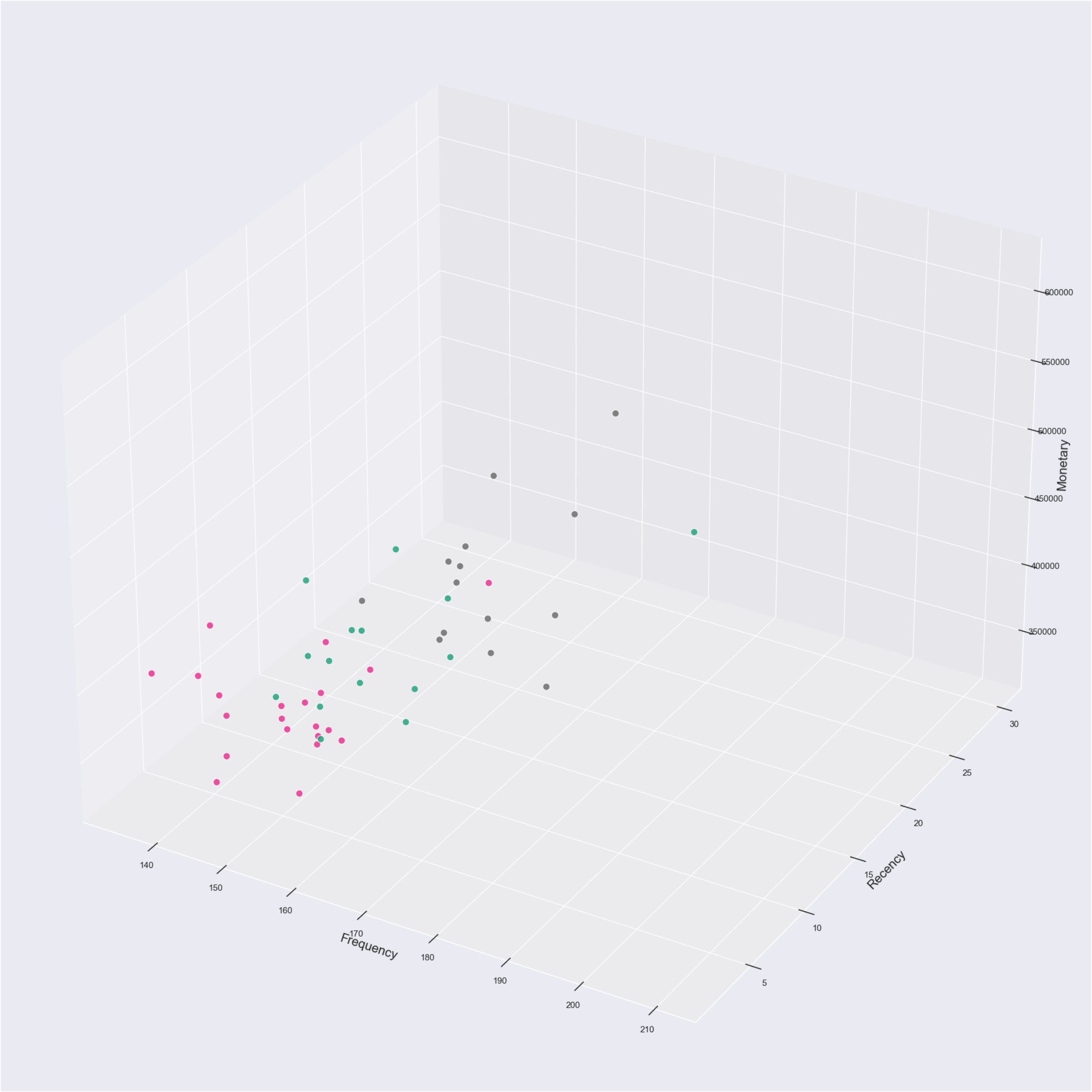
[27]:

```
#silhouette
from yellowbrick.cluster import SilhouetteVisualizer

fig, ax = plt.subplots(4, 2, figsize=(25,20))
for k in [2, 3, 4, 5, 6, 7, 8, 9]:

    km = KMeans(n_clusters=k, init='k-means++', n_init=10, max_iter=100,␣
 ↪random_state=42)
    q, mod = divmod(k, 2)

    visualizer = SilhouetteVisualizer(km, colors='yellowbrick', ax=ax[q-1][mod])
    visualizer.fit(df_scaled)

visualizer.show()
```



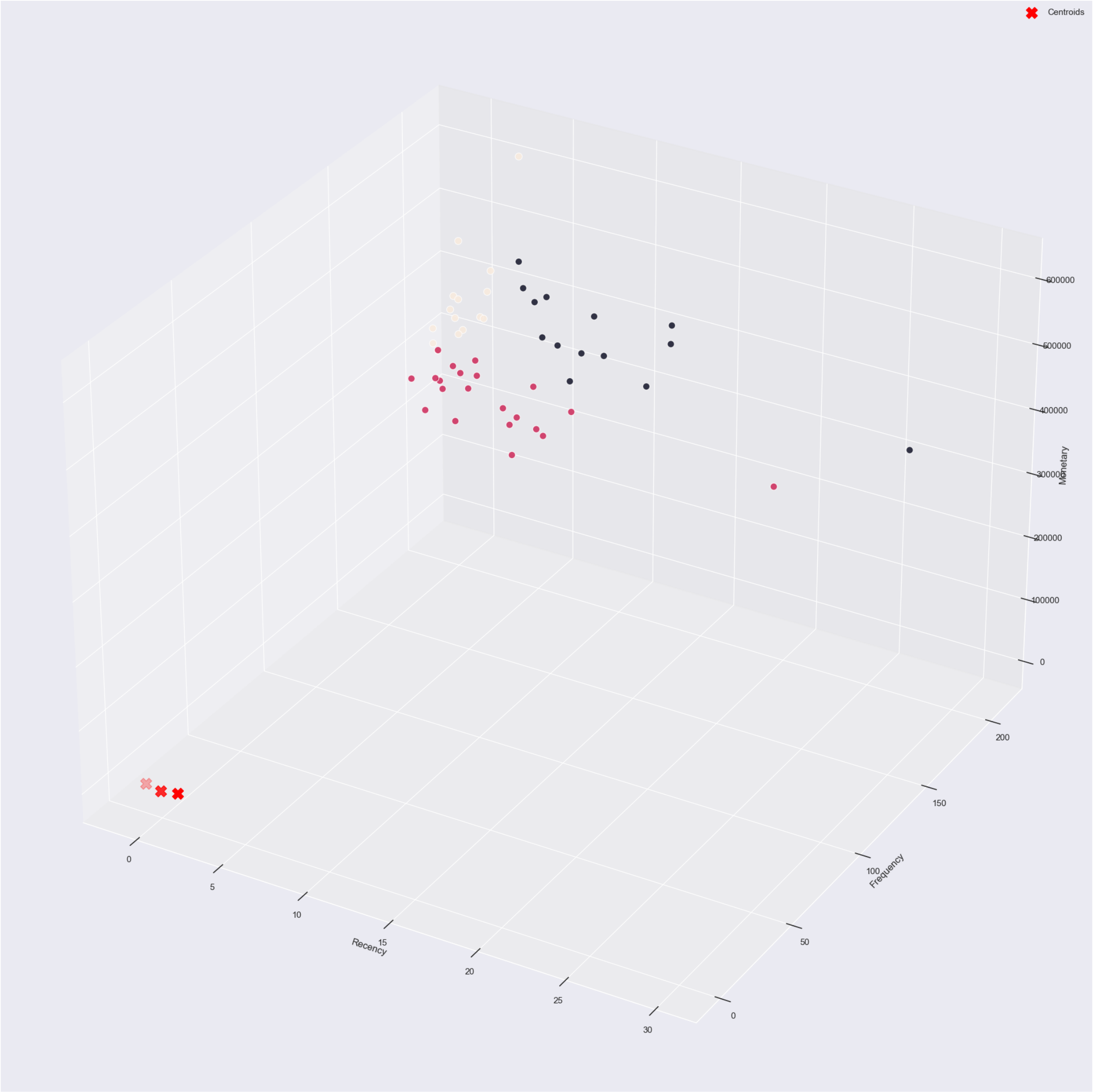
```
[27]: <Axes: title={'center': 'Silhouette Plot of KMeans Clustering for 50 Samples in
      9 Centers'}, xlabel='silhouette coefficient values', ylabel='cluster label'>
```

```
[28]: plt.rcParams['figure.figsize'] = (25, 25)
      fig = plt.figure(1)
      ax = fig.add_subplot(111, projection='3d')
      scatter = ax.scatter(df_new['Frequency'], df_new['Recency'], df_new['Monetary'],
                           c=df_new['Cluster'],
                           s=80,
                           cmap='Dark2_r',
                           alpha=0.8,
                           edgecolor='white')
      ax.set_xlabel('Frequency', fontsize=16)
      ax.set_ylabel('Recency', fontsize=16)
      ax.set_zlabel('Monetary', fontsize=16)

      plt.show()
```



[29]:
```
# Tạo biểu đồ 3D
fig = plt.figure()
ax = fig.add_subplot(111, projection='3d')

# Lấy dữ liệu từ df_new
cluster_labels = df_new['Cluster']
recency = df_new['Recency']
frequency = df_new['Frequency']
monetary = df_new['Monetary']

# Vẽ biểu đồ 3D cho các điểm dữ liệu
scatter = ax.scatter(recency, frequency, monetary, c=cluster_labels, s=80,␣
 ↪alpha=0.8, edgecolor='white')
```

```
ax.set_xlabel('Recency')
ax.set_ylabel('Frequency')
ax.set_zlabel('Monetary')

# Vẽ trung tâm cụm
centroid_recency = centroid_df['Recency']
centroid_frequency = centroid_df['Frequency']
centroid_monetary = centroid_df['Monetary']
ax.scatter(centroid_recency, centroid_frequency, centroid_monetary, c='red',␣
 ↪marker='X', s=200, label='Centroids')

plt.legend()
plt.show()
```


✖ Centroids
Recency
Frequency
Monetary