

```
In [3]: import pandas as pd
import datetime
import math
import numpy as np
import matplotlib.pyplot as plt
%matplotlib inline

retail_df = pd.read_excel(io='Online Retail.xlsx')
retail_df.head(3)
```

```
Out[3]:
```

	InvoiceNo	StockCode	Description	Quantity	InvoiceDate	UnitPrice	CustomerID	Country
0	536365	85123A	WHITE HANGING HEART T- LIGHT HOLDER	6	2010-12-01 08:26:00	2.55	17850.0	United Kingdom
1	536365	71053	WHITE METAL LANTERN	6	2010-12-01 08:26:00	3.39	17850.0	United Kingdom
2	536365	84406B	CREAM CUPID HEARTS COAT HANGER	8	2010-12-01 08:26:00	2.75	17850.0	United Kingdom

```
In [4]: retail_df.info()

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 541909 entries, 0 to 541908
Data columns (total 8 columns):
#   Column          Non-Null Count  Dtype
---  -
0   InvoiceNo        541909 non-null object
1   StockCode        541909 non-null object
2   Description      540455 non-null object
3   Quantity         541909 non-null int64
4   InvoiceDate      541909 non-null datetime64[ns]
5   UnitPrice        541909 non-null float64
6   CustomerID       406829 non-null float64
7   Country          541909 non-null object
dtypes: datetime64[ns](1), float64(2), int64(1), object(4)
memory usage: 33.1+ MB
```

```
In [5]: retail_df = retail_df[retail_df['Quantity'] > 0]
retail_df = retail_df[retail_df['UnitPrice'] > 0]
retail_df = retail_df[retail_df['CustomerID'].notnull()]
print(retail_df.shape)
retail_df.isnull().sum()
```

```
(397884, 8)
```

```
Out[5]: InvoiceNo      0
StockCode    0
Description  0
Quantity     0
InvoiceDate  0
UnitPrice    0
CustomerID   0
Country      0
dtype: int64
```

```
In [6]: retail_df['Country'].value_counts()[:5]
```

```
Out[6]: United Kingdom    354321
        Germany          9040
        France           8341
        EIRE             7236
        Spain            2484
        Name: Country, dtype: int64
```

```
In [7]: retail_df = retail_df[retail_df['Country']=='United Kingdom']
        print(retail_df.shape)
```

```
(354321, 8)
```

```
In [8]: retail_df['sale_amount'] = retail_df['Quantity'] * retail_df['UnitPrice']
        retail_df['CustomerID'] = retail_df['CustomerID'].astype(int)
```

```
In [9]: print(retail_df['CustomerID'].value_counts().head(5))
        print(retail_df.groupby('CustomerID')['sale_amount'].sum().sort_values(ascending=False))
```

```
17841    7847
14096    5111
12748    4595
14606    2700
15311    2379
Name: CustomerID, dtype: int64
CustomerID
18102    259657.30
17450    194550.79
16446    168472.50
17511     91062.38
16029     81024.84
Name: sale_amount, dtype: float64
```

```
In [10]: retail_df.groupby(['InvoiceNo', 'StockCode'])['InvoiceNo'].count().mean() ##??
```

```
Out[10]: 1.028702077315023
```

```
In [11]: #DataFrame의 groupby()의 multiple 연산을 위해 agg()이용
        #Recency는 InvoiceDate 칼럼의 max()에서 데이터 가공
        #Frequency는 InvoiceNo 칼럼의 count(), Monetary value는 sale_amount 칼럼의 sum()
        aggregations = {
            'InvoiceDate': 'max',
            'InvoiceNo' : 'count',
            'sale_amount': 'sum'
        }
        cust_df = retail_df.groupby('CustomerID').agg(aggregations)
        #groupby된 결과 칼럼 값을 Recency, Frequency, Monetary로 변경
        cust_df = cust_df.rename(columns = {'InvoiceDate':'Recency',
                                            'InvoiceNo':'Frequency',
                                            'sale_amount':'Monetary'
                                           })
        cust_df = cust_df.reset_index()
        cust_df.head(3)
```

```
Out[11]:
```

	CustomerID	Recency	Frequency	Monetary
0	12346	2011-01-18 10:01:00	1	77183.60
1	12747	2011-12-07 14:34:00	103	4196.01
2	12748	2011-12-09 12:20:00	4595	33719.73

```
In [12]: import datetime as dt
```

```

cust_df['Recency'] = dt.datetime(2011, 12, 10) - cust_df['Recency']
cust_df['Recency'] = cust_df['Recency'].apply(lambda x: x.days+1)
print('cust_df 로우와 칼럼 건수는 ', cust_df.shape)
cust_df.head(3)

```

cust_df 로우와 칼럼 건수는 (3920, 4)

Out[12]:

	CustomerID	Recency	Frequency	Monetary
0	12346	326	1	77183.60
1	12747	3	103	4196.01
2	12748	1	4595	33719.73

In [13]:

```

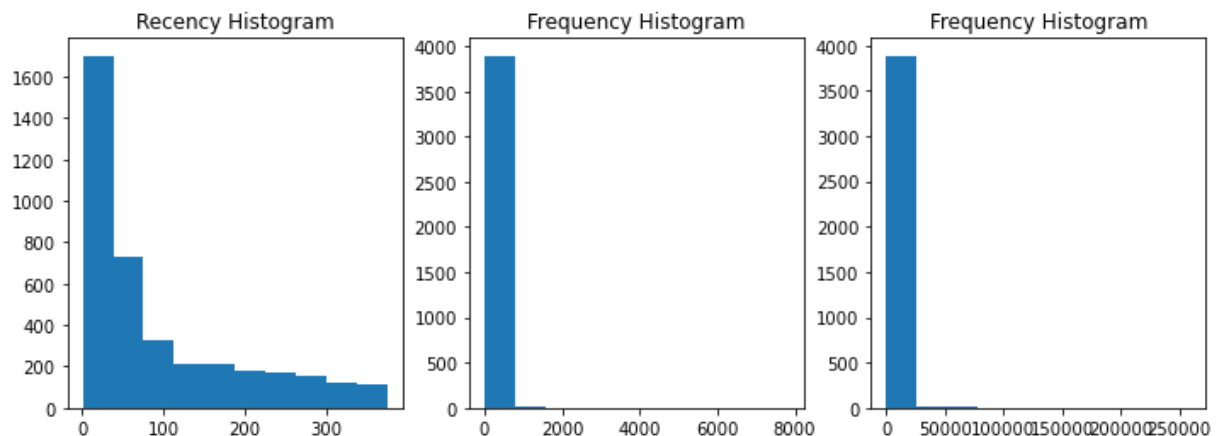
fig, (ax1, ax2, ax3) = plt.subplots(figsize=(12, 4), nrows = 1, ncols=3)
ax1.set_title('Recency Histogram')
ax1.hist(cust_df['Recency'])

ax2.set_title('Frequency Histogram')
ax2.hist(cust_df['Frequency'])

ax3.set_title('Frequency Histogram')
ax3.hist(cust_df['Monetary'])

```

Out[13]: (array([3.887e+03, 1.900e+01, 9.000e+00, 2.000e+00, 0.000e+00, 0.000e+00,
1.000e+00, 1.000e+00, 0.000e+00, 1.000e+00]),
array([3.75000000e+00, 2.59691050e+04, 5.19344600e+04, 7.78998150e+04,
1.03865170e+05, 1.29830525e+05, 1.55795880e+05, 1.81761235e+05,
2.07726590e+05, 2.33691945e+05, 2.59657300e+05]),
<BarContainer object of 10 artists>)



In [14]: cust_df[['Recency', 'Frequency', 'Monetary']].describe()

Out[14]:

	Recency	Frequency	Monetary
count	3920.000000	3920.000000	3920.000000
mean	92.742092	90.388010	1864.385601
std	99.533485	217.808385	7482.817477
min	1.000000	1.000000	3.750000
25%	18.000000	17.000000	300.280000
50%	51.000000	41.000000	652.280000
75%	143.000000	99.250000	1576.585000
max	374.000000	7847.000000	259657.300000

```
In [15]: from sklearn.preprocessing import StandardScaler
from sklearn.cluster import KMeans
from sklearn.metrics import silhouette_score, silhouette_samples

X_features = cust_df[['Recency', 'Frequency', 'Monetary']].values
X_features_scaled = StandardScaler().fit_transform(X_features)

kmeans = KMeans(n_clusters=3, random_state=0)
labels = kmeans.fit_predict(X_features_scaled)
cust_df['cluster_label'] = labels

print('실루엣 스코어는 : {0:3f}'.format(silhouette_score(X_features_scaled, labels)))

실루엣 스코어는 : 0.592454
```

```
In [20]: def visualize_silhouette(cluster_lists, X_features):
    from sklearn.datasets import make_blobs
    from sklearn.cluster import KMeans
    from sklearn.metrics import silhouette_samples, silhouette_score

    import matplotlib.pyplot as plt
    import matplotlib.cm as cm
    import math
    #입력값으로 클러스터링 갯수들을 리스트로 받아서, 각 갯수별로 클러스터링을 적용하고
    n_cols = len(cluster_lists)
    #plt.subplots()으로 리스트에 기재된 클러스터링 만큼의 sub figures를 가지는 axs 생성
    fig, axs = plt.subplots(figsize=(4*n_cols, 4), nrows = 1, ncols = n_cols)
    #리스트에 기재된 클러스터링 갯수들을 차례로 iteration 수행하면서 실루엣 개수 시각화
    for ind, n_cluster in enumerate(cluster_lists):
        #KMeans 클러스터링 수행하고, 실루엣 스코어와 개별 데이터의 실루엣 값 계산
        clusterer = KMeans(n_clusters = n_cluster, max_iter=500, random_state = 0)
        cluster_labels = clusterer.fit_predict(X_features)
        sil_avg = silhouette_score(X_features, cluster_labels)
        sil_values = silhouette_samples(X_features, cluster_labels)
        y_lower = 10
        axs[ind].set_title('Number of Cluster : ' + str(n_cluster) +
                           'Silhouette Score : ' + str(round(sil_avg, 3)) )
        axs[ind].set_xlabel("The silhouette coefficient values"),
        axs[ind].set_ylabel("Cluster label"),
        axs[ind].set_xlim([-0.1, 1]),
        axs[ind].set_ylim([0, len(X_features) + (n_cluster + 1) * 10])
        axs[ind].set_yticks([]) #Clear the yaxis labels / ticks
        axs[ind].set_xticks([0, 0.2, 0.4, 0.6, 0.8, 1])

        #클러스터링 갯수별로 fill_betweenx()형태의 막대 그래프 표현.
        for i in range(n_cluster):
            ith_cluster_sil_values = sil_values[cluster_labels==i]
            ith_cluster_sil_values.sort()

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

            color = cm.nipy_spectral(float(i) / n_cluster)
            axs[ind].fill_betweenx(np.arange(y_lower, y_upper), 0, ith_cluster_sil_va
                                  facecolor=color, edgecolor = color, alpha = 0.7)
            axs[ind].text(-0.05, y_lower + 0.5 * size_cluster_i, str(i))
            y_lower = y_upper + 10

        axs[ind].axvline(x=sil_avg, color = "red", linestyle = "--")
```

```
In [37]: #여러객의 클러스터링 갯수를 list로 입력 받아 각각의 클러스터링 결과를 시각화
def visualize_kmeans_plot_multi(cluster_lists, X_features):
    from sklearn.cluster import KMeans
```

```

from sklearn.decomposition import PCA
import pandas as pd
import numpy as np

#plt.subplot()으로 리스트에 기재된 클러스터링 만큼의 sub figures를 가지는 axs 생성
n_cols = len(cluster_lists)
fig, axs = plt.subplots(figsize = (4*n_cols, 4), nrows = 1, ncols=n_cols)
#입력 데이터의 FEATURE가 여러개일 경우 2차원 데이터 시각화가 어려우므로 PCA 변환하
pca = PCA(n_components=2)
pca_transformed = pca.fit_transform(X_features)
dataframe = pd.DataFrame(pca_transformed, columns=['PCA1', 'PCA2'])
#리스트에 기재된 클러스터링 갯수들을 차례로 iteration 수행하면서 KMeans 클러스터링
for ind, n_cluster in enumerate(cluster_lists):
    #KMeans 클러스터링으로 클러스터링 결과를 dataframe에 저장.
    clusterer = KMeans(n_clusters = n_cluster, max_iter = 500, random_state=0)
    cluster_labels = clusterer.fit_predict(pca_transformed)
    dataframe['cluster'] = cluster_labels
    unique_labels = np.unique(clusterer.labels_)
    markers=['o', 's', '^', 'x', '*']
    #클러스터링 결과값 별로 scatter plot으로 시각화
    for label in unique_labels:
        label_df = dataframe[dataframe['cluster']==label]
        if label == -1:
            cluster_legend = 'Noise'
        else:
            cluster_legend = 'Cluster '+str(label)
        axs[ind].scatter(x=label_df['PCA1'], y=label_df['PCA2'], s=70,
                        edgecolor = 'k', marker=markers[label], label=cluster_leg
        axs[ind].set_title('Number of Cluster : '+str(n_cluster))
        axs[ind].legend(loc='upper right')

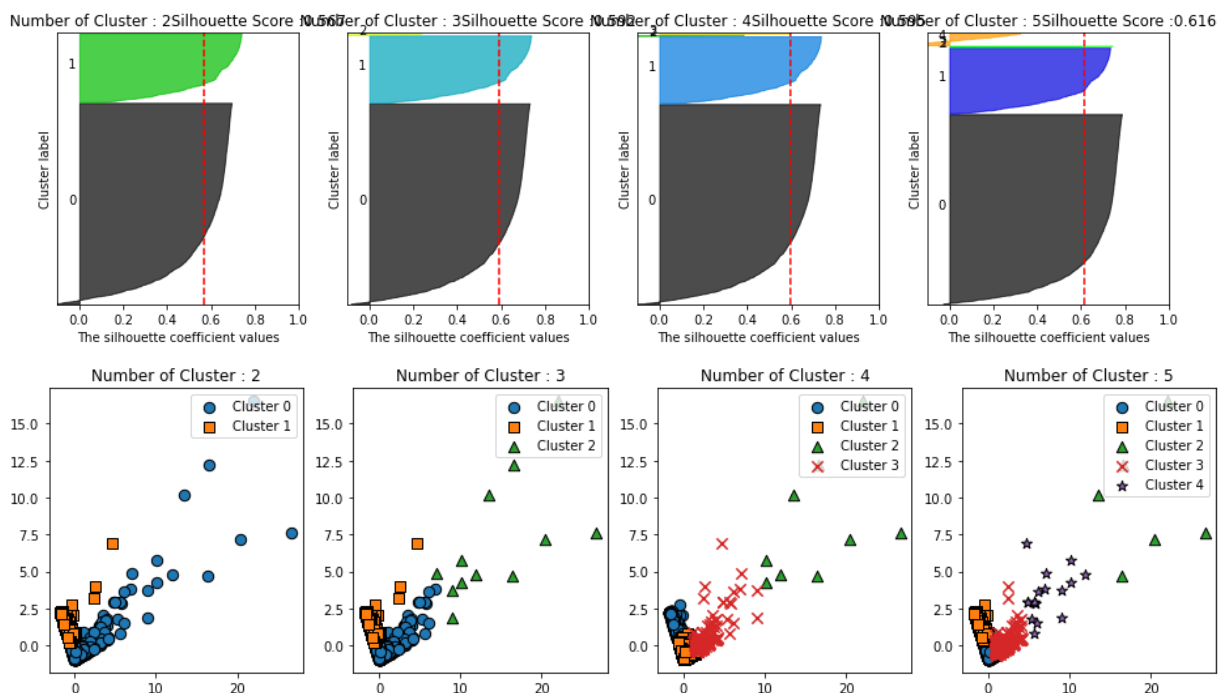
plt.show()

```

```

In [38]: visualize_silhouette([2, 3, 4, 5], X_features_scaled)
visualize_kmeans_plot_multi([2, 3, 4, 5], X_features_scaled)

```



```

In [39]: from sklearn.preprocessing import StandardScaler
from sklearn.cluster import KMeans
from sklearn.metrics import silhouette_score, silhouette_samples

```

```
#Recency, Frequency, Monetary 칼럼에 np.log1p()로 Log Transformation
cust_df['Recency_log'] = np.log1p(cust_df['Recency'])
cust_df['Frequency_log'] = np.log1p(cust_df['Frequency'])
cust_df['Monetary_log'] = np.log1p(cust_df['Monetary'])

#Log Transformation 데이터에 StandardScaler 적용
X_features = cust_df[['Recency_log', 'Frequency_log', 'Monetary_log']].values
X_features_scaled = StandardScaler().fit_transform(X_features)

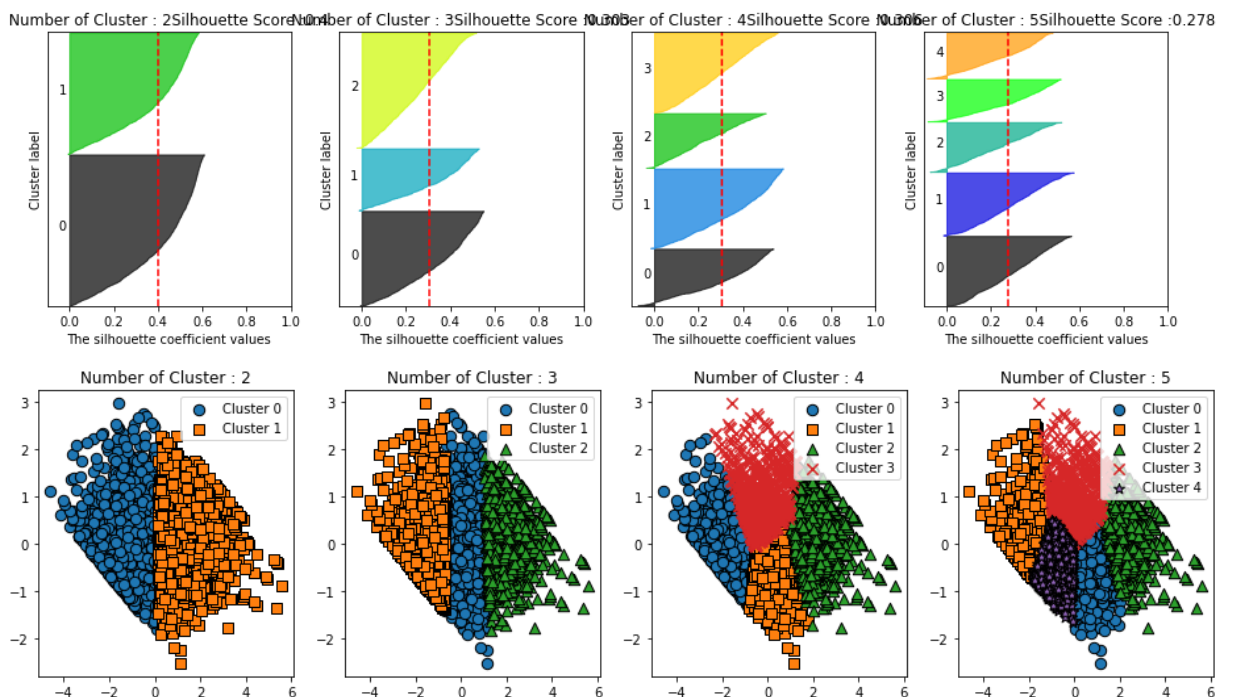
kmeans = KMeans(n_clusters = 3, random_state=0)
labels = kmeans.fit_predict(X_features_scaled)

cust_df['cluster_label'] = labels

print('실루엣 스코어는 : {0:.3f}'.format(silhouette_score(X_features_scaled, labels)))
```

실루엣 스코어는 : 0.303

```
In [40]: visualize_silhouette([2, 3, 4, 5], X_features_scaled)
visualize_kmeans_plot_multi([2, 3, 4, 5], X_features_scaled)
```



In []:

In []:

In []:

In []:

In []:

In []:

In []:

In []:

In []:

