# 군집 분석

빅데이터 분석

# 군집 분석

## ■ 비지도 학습

- 훈련 데이터에 타깃값이 주어지지 않은 상태에서 학습을 수행하는 방식
- 훈련 데이터를 학습하여 모델을 생성하면서 유사한 특성(관계, 패턴 등)을 가지는 데이터를 클러스터로 구성
- 새로운 데이터의 특성을 분석하여 해당하는 클러스터를 예측

## ■ 군집화

- 데이터를 클러스터(군집)로 구성하는 작업

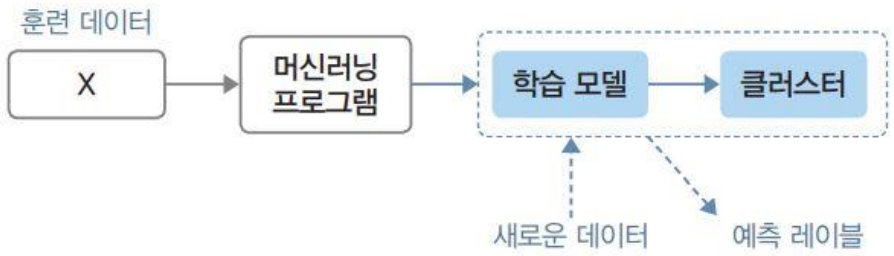



그림 12-1 머신러닝의 비지도 학습 구조

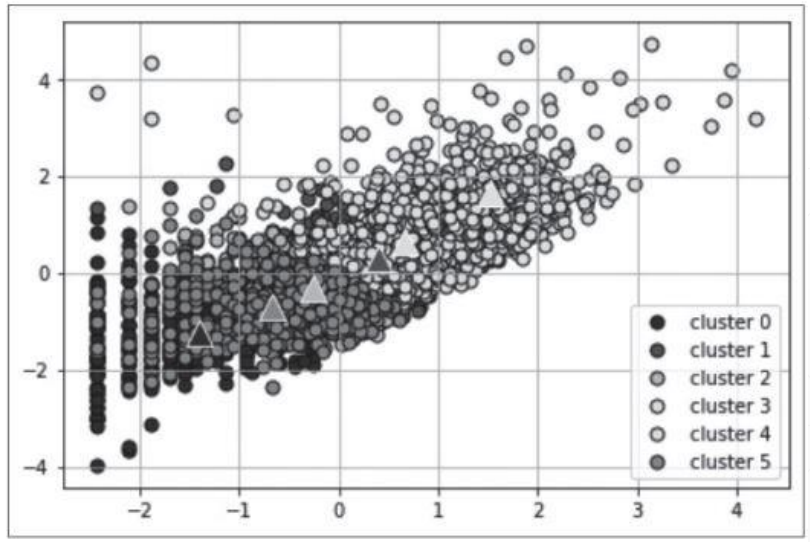
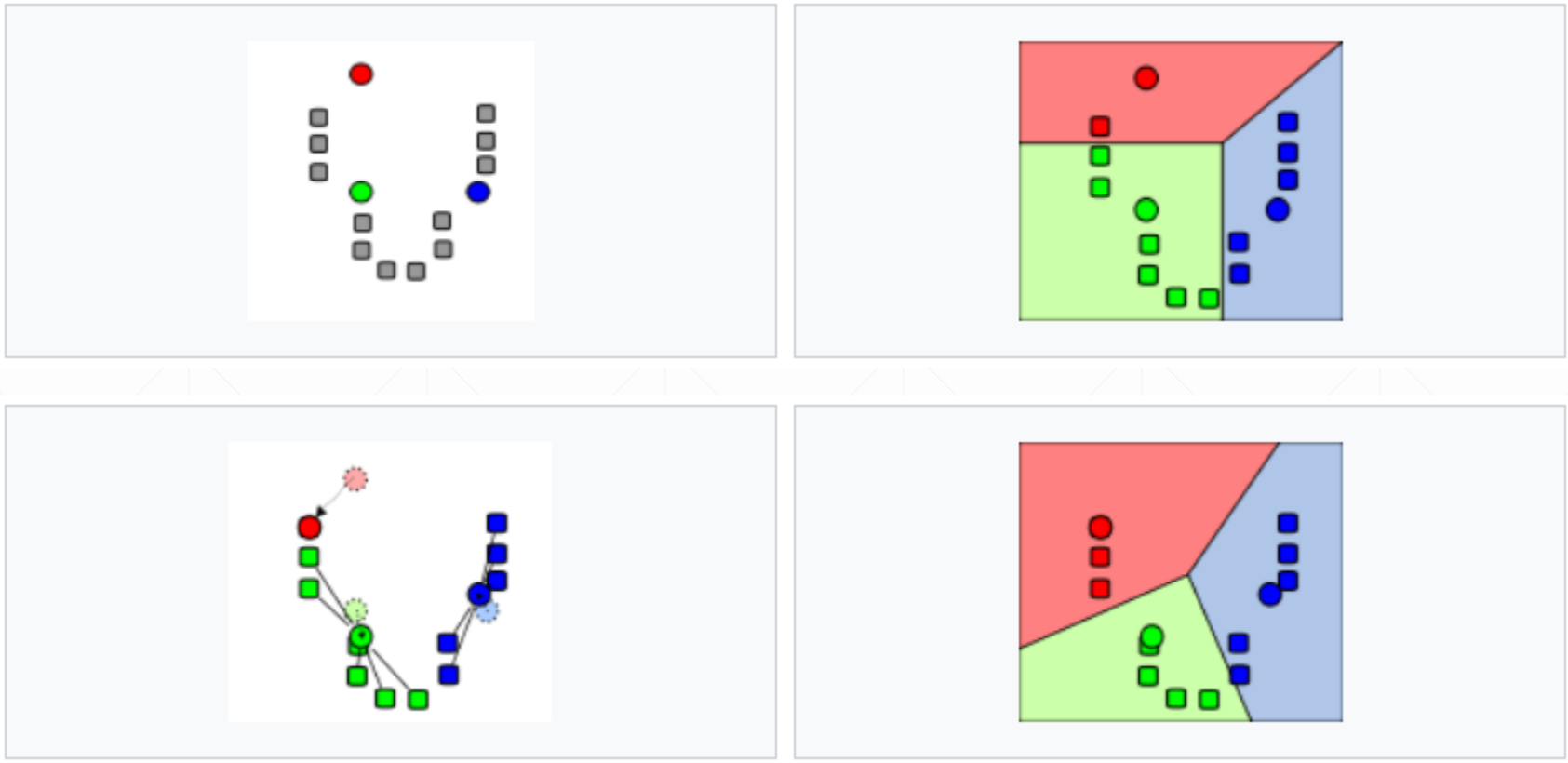


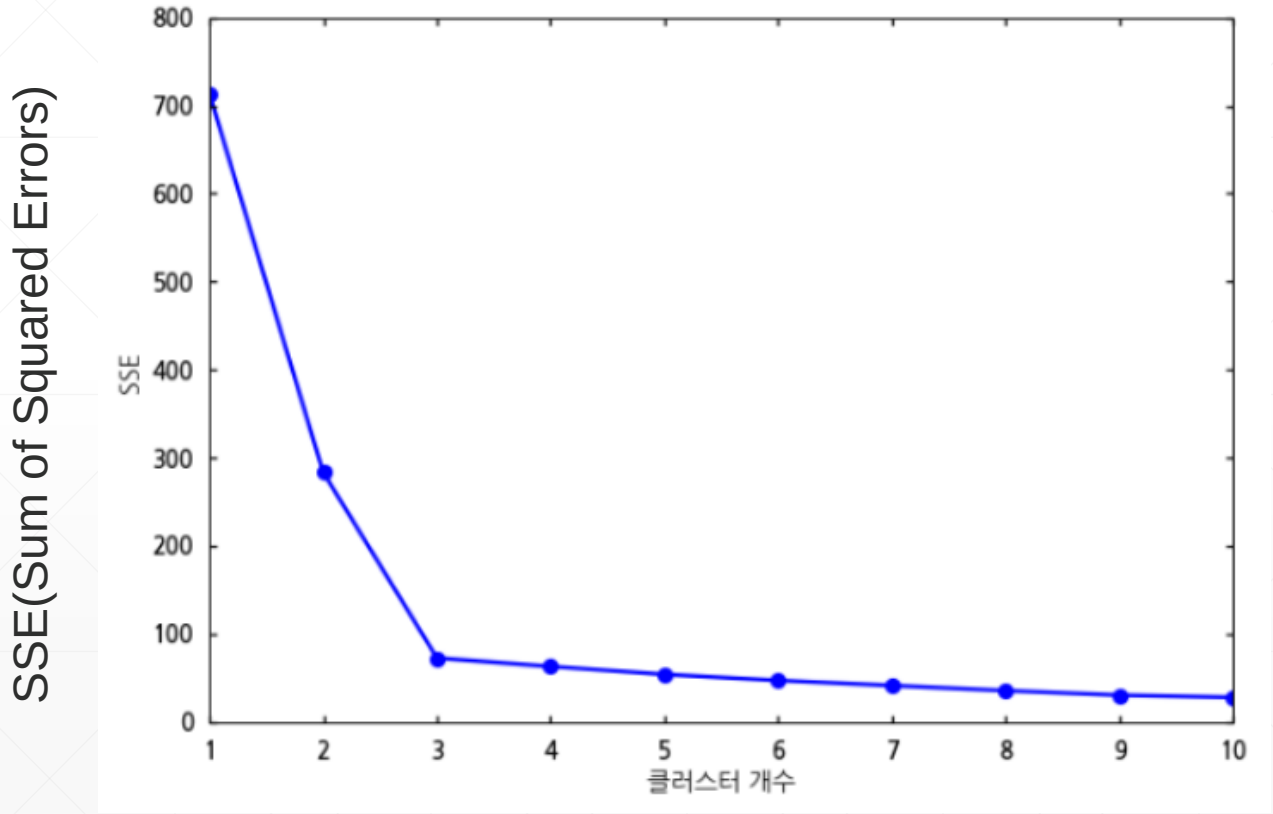

그림 12-2 데이터 군집화의 예

# K-평균 군집화 분석

- 알고리즘 이해

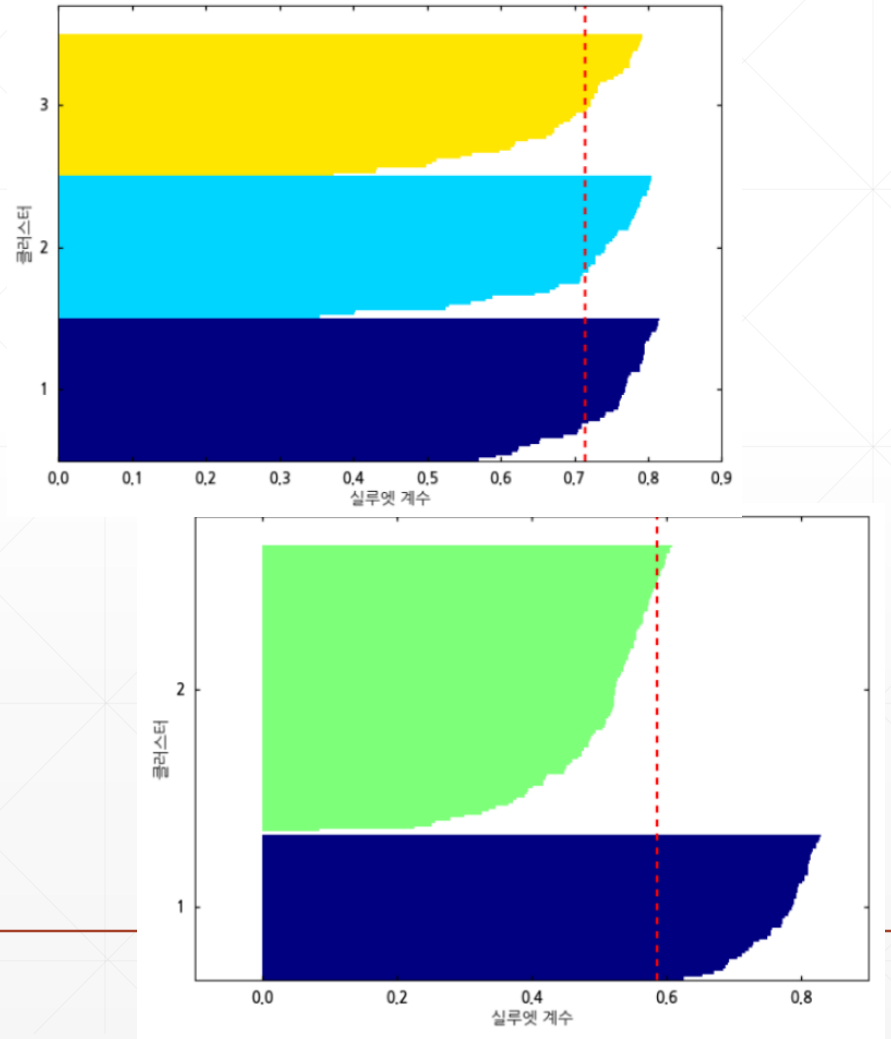


# 엘보 방법




- **왜곡 :** 클러스터의 중심점과 클러스터 내의 데이터 거리 차이의 제곱값의 합
- 클러스터의 개수 K의 변화에 따른 왜곡의 변화를 그래프로 그려보면 **그래프가 꺾이는 지점**인 엘보가 나타나는데, 그 지점의 K를 최적의 K로 선택

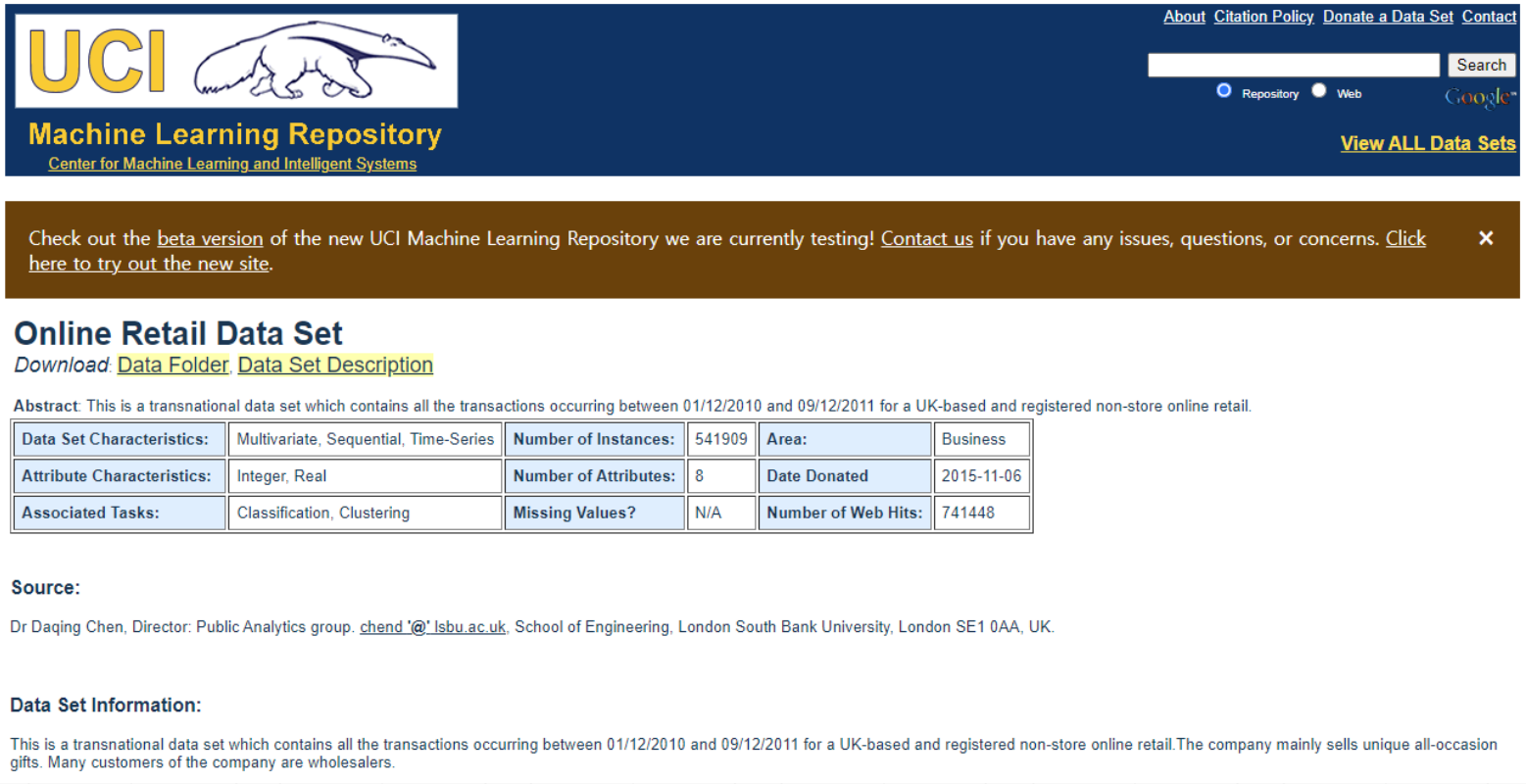
# 실루엣 분석




- 클러스터 내에 있는 데이터가 얼마나 조밀하게 모여있는지를 측정하는 그래프 도구
- $a(i)$ – 데이터 $i$가 해당 클러스터 내의 데이터와 얼마나 가까운가를 나타내는 클러스터 **응집력** ↓
- $b(i)$ – 가장 가까운 다른 클러스터 내의 데이터와 얼마나 떨어져 있는가를 나타내는 클러스터 **분리도** ↑
- $s(i) = \frac{b(i)-a(i)}{\max\{a(i),b(i)\}}$

0~1 사이의 값을 가지며 1에 가까울수록 좋은 군집화를 의미

# 타겟 마케팅을 위한 소비자 군집 분석하기



UCI Machine Learning Repository
Center for Machine Learning and Intelligent Systems

About Citation Policy Donate a Data Set Contact

Search
Repository Web
Google™

View ALL Data Sets

Check out the beta version of the new UCI Machine Learning Repository we are currently testing! Contact us if you have any issues, questions, or concerns. Click here to try out the new site.

## Online Retail Data Set

*Download*: Data Folder, Data Set Description

**Abstract**: This is a transnational data set which contains all the transactions occurring between 01/12/2010 and 09/12/2011 for a UK-based and registered non-store online retail.

| | | | | | |
|---|---|---|---|---|---|
| **Data Set Characteristics:** | Multivariate, Sequential, Time-Series | **Number of Instances:** | 541909 | **Area:** | Business |
| **Attribute Characteristics:** | Integer, Real | **Number of Attributes:** | 8 | **Date Donated** | 2015-11-06 |
| **Associated Tasks:** | Classification, Clustering | **Missing Values?** | N/A | **Number of Web Hits:** | 741448 |

**Source:**

Dr Daqing Chen, Director: Public Analytics group. chend '@' lsbu.ac.uk, School of Engineering, London South Bank University, London SE1 0AA, UK.

**Data Set Information:**

This is a transnational data set which contains all the transactions occurring between 01/12/2010 and 09/12/2011 for a UK-based and registered non-store online retail.The company mainly sells unique all-occasion gifts. Many customers of the company are wholesalers.

https://archive.ics.uci.edu/ml/datasets/online+retail

# 데이터 수집

| | |
|---|---|
| In [1]: | import pandas as pd<br>import math |
| In [2]: | retail_df = pd.read_excel('I:/My_Python/Online_Retail.xlsx')<br>retail_df.head() |

| | 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 |
| 3 | 536365 | 84029G | KNITTED UNION FLAG HOT WATER BOTTLE | 6 | 2010-12-01 08:26:00 | 3.39 | 17850.0 | United Kingdom |
| 4 | 536365 | 84029E | RED WOOLLY HOTTIE WHITE HEART. | 6 | 2010-12-01 08:26:00 | 3.39 | 17850.0 | United Kingdom |

# 데이터 준비 및 탐색

## 1. 데이터 정보 확인하기

In [3]:
```
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
```

- **InvoiceNo :** 6자리 정수로 이루어진 송장 번호. 'C'로 시작하는 것은 취소 주문
- **StockCode :** 제품 고유의 품목 코드
- **Description :** 제품 설명
- **Quantity :** 주문 수량
- **Country :** 주문 고객의 국적
- **InvoiceDate :** 주문 날짜와 시간
- **UnitPrice :** 제품 단가(£, 영국 파운드화)
- **CustomerID :** 주문 고객 번호

# 데이터 준비 및 탐색 (cont'd)

## 2. 데이터 정제하기

**In [4]:**

```
# 오류 데이터 정제
retail_df = retail_df[retail_df['Quantity']>0]
retail_df = retail_df[retail_df['UnitPrice']>0]
retail_df = retail_df[retail_df['CustomerID'].notnull()]

# 'CustomerID' 자료형을 정수형으로 변환
retail_df['CustomerID'] = retail_df['CustomerID'].astype(int)

retail_df.info()
print(retail_df.isnull().sum())
print(retail_df.shape)
```

**In [5]:**

```
# 중복 레코드 제거
retail_df.drop_duplicates(inplace = True)
print(retail_df.shape) # 작업 확인용 출력
```

# 데이터 준비 및 탐색 (cont'd)

**3.** 데이터 탐색을 위해 제품 수, 거래 건수, 고객 수를 알아보고 고객의 국적도 확인

| | |
|---|---|
| **In [6]:** | pd.DataFrame([{'Product':len(retail_df['StockCode'].value_counts()), 'Transaction':len(retail_df['InvoiceNo'].value_counts()), 'Customer':len(retail_df['CustomerID'].value_counts())}], columns = ['Product', 'Transaction', 'Customer'], index = ['counts']) |
| **In [7]:** | retail_df['Country'].value_counts() |

| | |
|---|---|
| United Kingdom | 349203 |
| Germany | 9025 |
| France | 8326 |
| EIRE | 7226 |
| Spain | 2479 |
| Netherlands | 2359 |
| Belgium | 2031 |
| Switzerland | 1841 |
| Portugal | 1453 |
| Australia | 1181 |
| Norway | 1071 |
| Italy | 758 |
| Channel Islands | 747 |
| Finland | 685 |
| Cyprus | 603 |
| Sweden | 450 |
| Austria | 398 |

# 데이터 준비 및 탐색 (cont'd)

| | CustomerID | Freq | SaleAmount | ElapsedDays |
|---|---|---|---|---|
| 0 | 12346 | 1 | 77183.60 | 2011-01-18 10:01:00 |
| 1 | 12347 | 182 | 4310.00 | 2011-12-07 15:52:00 |
| 2 | 12348 | 31 | 1797.24 | 2011-09-25 13:13:00 |
| 3 | 12349 | 73 | 1757.55 | 2011-11-21 09:51:00 |
| 4 | 12350 | 17 | 334.40 | 2011-02-02 16:01:00 |

4. 마케팅에 이용하기 위해 고객의 주문 횟수, 주문 총액, 그리고 마지막 주문 후 며칠이 지났는지에 대한 정보를 추출

**In [8]:**

```
# 주문 금액 컬럼 추가
retail_df['SaleAmount'] = retail_df['UnitPrice']*retail_df['Quantity']
retail_df.head() # 작업 확인용 출력
```

**In [9]:**

```
aggregations = {
 'InvoiceNo':'count',
 'SaleAmount':'sum',
 'InvoiceDate':'max'
}

customer_df = retail_df.groupby('CustomerID').agg(aggregations)
customer_df = customer_df.reset_index()
customer_df.head() # 작업 확인용 출력
```

**In [10]:**

```
customer_df = customer_df.rename(columns = {'InvoiceNo':'Freq', 'InvoiceDate':'ElapsedDays'})
customer_df.head() # 작업 확인용 출력
```

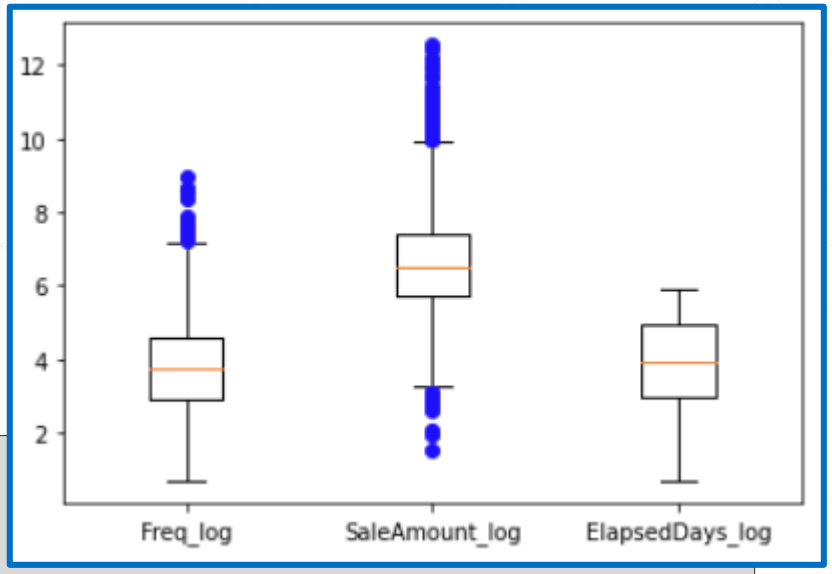
# 데이터 준비 및 탐색 (cont'd)

**5.** 마지막 주문일로부터 며칠이 지났는지에 대한 값을 **ElapsedDays** 컬럼에 저장
'기준 날짜 - 마지막 구매일'로 계산해 구함 (날짜기준: 2011년 12월 10일)

| | |
|---|---|
| **In [11]:** | import datetime<br><br>customer_df['ElapsedDays'] = datetime.datetime(2011,12,10) - customer_df['ElapsedDays']<br>customer_df.head() # 작업 확인용 출력 |
| **In [12]:** | customer_df['ElapsedDays'] = customer_df['ElapsedDays'].apply(lambda x: x.days+1)<br>customer_df.head() # 작업 확인용 출력 |

| | CustomerID | Freq | SaleAmount | ElapsedDays |
|---|---|---|---|---|
| 0 | 12346 | 1 | 77183.60 | 326 |
| 1 | 12347 | 182 | 4310.00 | 3 |
| 2 | 12348 | 31 | 1797.24 | 76 |
| 3 | 12349 | 73 | 1757.55 | 19 |
| 4 | 12350 | 17 | 334.40 | 311 |

# 데이터 준비 및 탐색 (cont'd)




## 6. 데이터 분포 조정하기

**In [13]:**

```
import matplotlib.pyplot as plt
import seaborn as sns

fig, ax = plt.subplots()
ax.boxplot([customer_df['Freq'], customer_df['SaleAmount'], customer_df['ElapsedDays']], sym = 'bo')
plt.xticks([1, 2, 3], ['Freq', 'SaleAmount','ElapsedDays'])
plt.show()
```

**In [14]:**

```
import numpy as np

customer_df['Freq_log'] = np.log1p(customer_df['Freq'])
customer_df['SaleAmount_log'] = np.log1p(customer_df['SaleAmount'])
customer_df['ElapsedDays_log'] = np.log1p(customer_df['ElapsedDays'])
customer_df.head() # 작업 확인용 출력
```

**In [15]:**

```
fig, ax = plt.subplots()
ax.boxplot([customer_df['Freq_log'], customer_df['SaleAmount_log'], customer_df['ElapsedDays_log']], sym = 'bo')
plt.xticks([1, 2, 3], ['Freq_log', 'SaleAmount_log', 'ElapsedDays_log'])
plt.show()
```

# 분석 모델 구축

## 1. X_features를 정규 분포로 스케일링하기

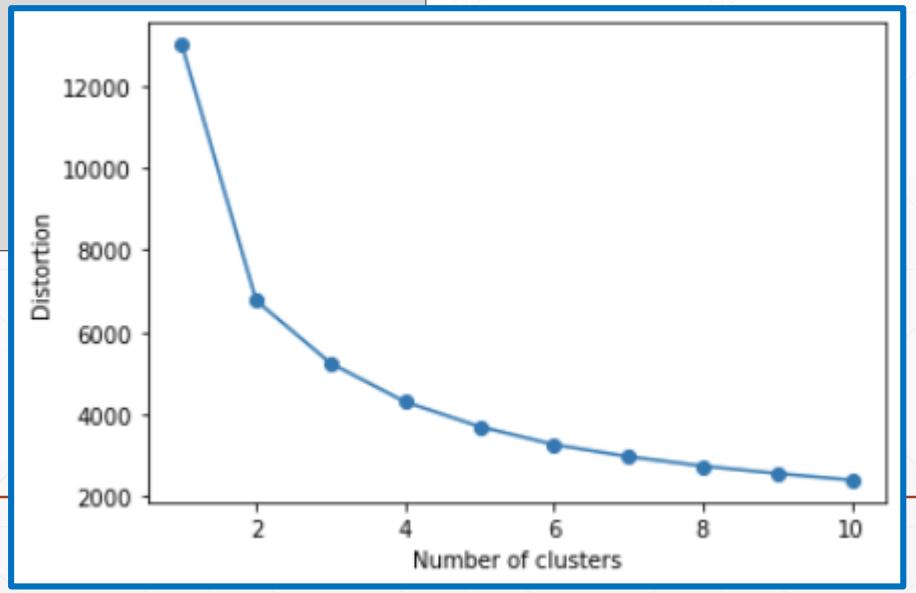
| | |
|---|---|
| In [16]: | from sklearn.cluster import KMeans<br>from sklearn.metrics import silhouette_score, silhouette_samples |
| In [17]: | X_features = customer_df[['Freq_log', 'SaleAmount_log', 'ElapsedDays_log']].values |
| In [18]: | from sklearn.preprocessing import StandardScaler<br>X_features_scaled = StandardScaler().fit_transform(X_features) |

# 분석 모델 구축 (cont'd)

## 2. 엘보 방법으로 클러스터 개수 선택하기

In [19]:

```
distortions = []

for i in range(1, 11):
    kmeans_i = KMeans(n_clusters = i, random_state = 0) # 모델 생성
    kmeans_i.fit(X_features_scaled)                     # 모델 훈련
    distortions.append(kmeans_i.inertia_)

plt.plot(range(1,11), distortions, marker = 'o')
plt.xlabel('Number of clusters')
plt.ylabel('Distortion')
plt.show()
```



# 분석 모델 구축 (cont'd)

3. 클러스터의 개수를 3으로 설정하여 K-평균 모델을 다시 구축한 뒤 모델에서 만든 클러스터 레이블을 확인

| | |
|---|---|
| In [20]: | kmeans = KMeans(n_clusters=3, random_state=0) # 모델 생성<br># 모델 학습과 결과 예측(클러스터 레이블 생성)<br>Y_labels = kmeans.fit_predict(X_features_scaled) |
| In [21]: | customer_df['ClusterLabel'] = Y_labels<br>customer_df.head() # 작업 확인용 출력 |

| | CustomerID | Freq | SaleAmount | ElapsedDays | Freq_log | SaleAmount_log | ElapsedDays_log | ClusterLabel |
|---|---|---|---|---|---|---|---|---|
| 0 | 12346 | 1 | 77183.60 | 326 | 0.693147 | 11.253955 | 5.789960 | 2 |
| 1 | 12347 | 182 | 4310.00 | 3 | 5.209486 | 8.368925 | 1.386294 | 1 |
| 2 | 12348 | 31 | 1797.24 | 76 | 3.465736 | 7.494564 | 4.343805 | 2 |
| 3 | 12349 | 73 | 1757.55 | 19 | 4.304065 | 7.472245 | 2.995732 | 2 |
| 4 | 12350 | 17 | 334.40 | 311 | 2.890372 | 5.815324 | 5.743003 | 0 |

# 분석 모델 구축 (cont'd)

## 4. 실루엣 계수를 구하고, 각 클러스터의 비중을 가로 바 차트로 시각화

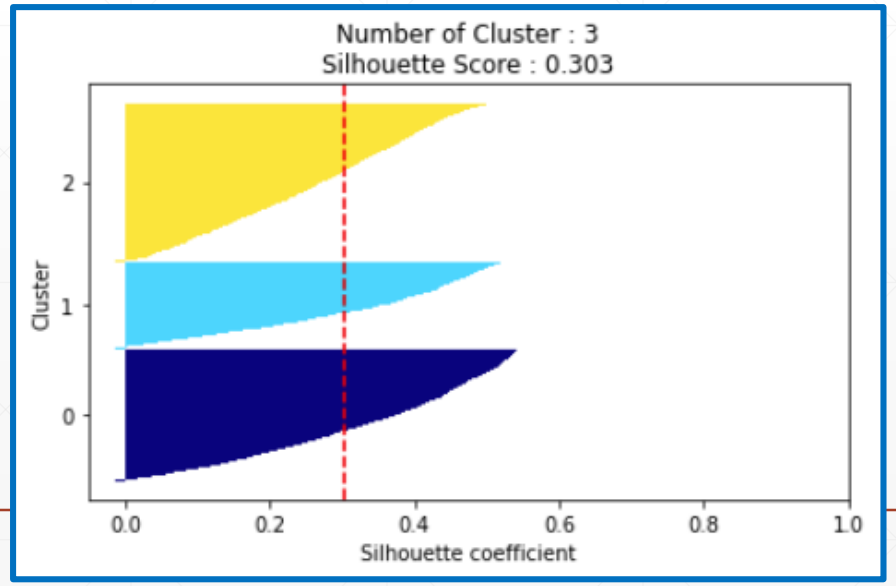
In [22]:

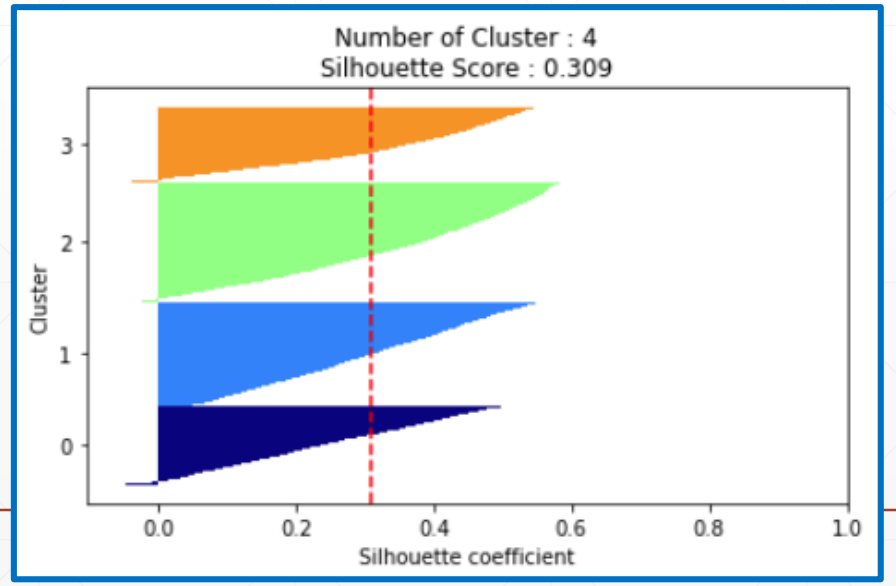
```
from matplotlib import cm

def silhouetteViz(n_cluster, X_features):
    kmeans = KMeans(n_clusters = n_cluster, random_state = 0)
    Y_labels = kmeans.fit_predict(X_features)

    silhouette_values = silhouette_samples(X_features, Y_labels, metric = 'euclidean')

    y_ax_lower, y_ax_upper = 0, 0
    y_ticks = []

    for c in range(n_cluster):
        c_silhouettes = silhouette_values[Y_labels == c]
        c_silhouettes.sort()
        y_ax_upper += len(c_silhouettes)
        color = cm.jet(float(c) / n_cluster)
        plt.barh(range(y_ax_lower, y_ax_upper), c_silhouettes, height = 1.0, edgecolor = 'none', color = color)
        y_ticks.append((y_ax_lower + y_ax_upper) / 2.)
        y_ax_lower += len(c_silhouettes)

    silhouette_avg = np.mean(silhouette_values)
    plt.axvline(silhouette_avg, color = 'red', linestyle = '--')
    plt.title('Number of Cluster : '+ str(n_cluster) + '₩n' + 'Silhouette Score : '+ str(round(silhouette_avg,3)))
    plt.yticks(y_ticks, range(n_cluster))
    plt.xticks([0, 0.2, 0.4, 0.6, 0.8, 1])
    plt.ylabel('Cluster')
    plt.xlabel('Silhouette coefficient')
    plt.tight_layout()
    plt.show()
```

# 분석 모델 구축 (cont'd)

4. 실루엣 계수를 구하고, 각 클러스터의 비중을 가로 바 차트로 시각화

| | |
|---|---|
| In [23]: | silhouetteViz(3, X_features_scaled) |
| In [24]: | silhouetteViz(4, X_features_scaled) |
| In [25]: | silhouetteViz(5, X_features_scaled) |
| In [26]: | silhouetteViz(6, X_features_scaled) |



# 결과 분석 및 시각화

## 1. 클러스터의 데이터 분포를 확인하기 위해 스캐터 차트로 시각화

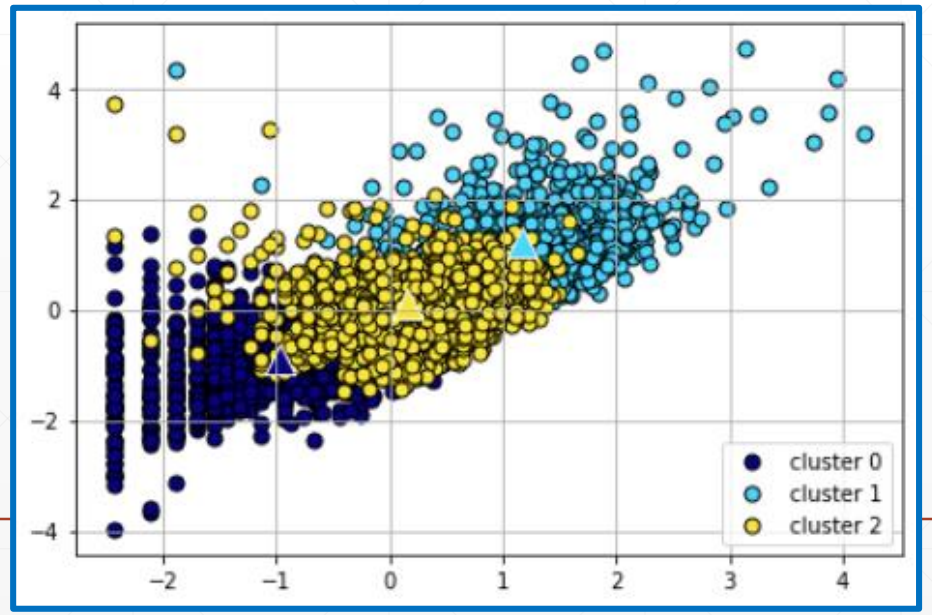
In [27]:

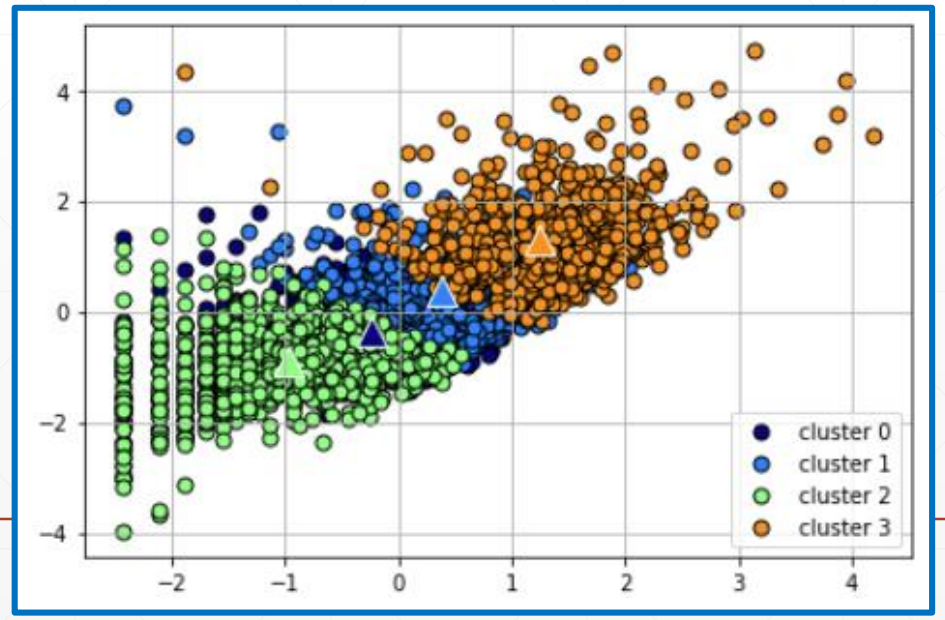
```
def clusterScatter(n_cluster, X_features):
    c_colors = []
    kmeans = KMeans(n_clusters = n_cluster, random_state = 0)
    Y_labels = kmeans.fit_predict(X_features)

    for i in range(n_cluster):
        c_color = cm.jet(float(i) / n_cluster) # 클러스터의 색상 설정
        c_colors.append(c_color)
        # 클러스터의 데이터 분포를 동그라미로 시각화
        plt.scatter(X_features[Y_labels == i,0], X_features[Y_labels == i,1], marker = 'o', color =  c_color, edgecolor = 'black', s = 50, label = 'cluster '+ str(i))

    # 각 클러스터의 중심점을 삼각형으로 표시
    for i in range(n_cluster):
        plt.scatter(kmeans.cluster_centers_[i,0], kmeans.cluster_centers_[i,1], marker = '^', color = c_colors[i], edgecolor = 'w', s = 200)

    plt.legend()
    plt.grid()
    plt.tight_layout()
    plt.show()
```

# 결과 분석 및 시각화 (cont'd)

1. 클러스터의 데이터 분포를 확인하기 위해 스캐터 차트로 시각화

| | |
|---|---|
| In [28]: | clusterScatter(3, X_features_scaled) |
| In [29]: | clusterScatter(4, X_features_scaled) |
| In [30]: | clusterScatter(5, X_features_scaled) |
| In [31]: | clusterScatter(6, X_features_scaled) |



# 결과 분석 및 시각화 (cont'd)

## 2. 최적의 클러스터 개수를 4로 결정

| | |
|---|---|
| In [32]: | best_cluster = 4<br>kmeans = KMeans(n_clusters = best_cluster, random_state = 0)<br>Y_labels = kmeans.fit_predict(X_features_scaled) |
| In [33]: | customer_df['ClusterLabel'] = Y_labels<br>customer_df.head() |
| In [34]: | customer_df.to_csv('I:/My_Python/Online_Retail_Customer_Cluster.csv') |

| | CustomerID | Freq | SaleAmount | ElapsedDays | Freq_log | SaleAmount_log | ElapsedDays_log | ClusterLabel |
|---|---|---|---|---|---|---|---|---|
| 0 | 12346 | 1 | 77183.60 | 326 | 0.693147 | 11.253955 | 5.789960 | 1 |
| 1 | 12347 | 182 | 4310.00 | 3 | 5.209486 | 8.368925 | 1.386294 | 3 |
| 2 | 12348 | 31 | 1797.24 | 76 | 3.465736 | 7.494564 | 4.343805 | 1 |
| 3 | 12349 | 73 | 1757.55 | 19 | 4.304065 | 7.472245 | 2.995732 | 1 |
| 4 | 12350 | 17 | 334.40 | 311 | 2.890372 | 5.815324 | 5.743003 | 2 |

# 결과 분석 및 시각화 (cont'd)

### 3. 클러스터의 특징을 살펴보기

In [35]:
```
customer_df.groupby('ClusterLabel')['CustomerID'].count()
```

```
ClusterLabel
0     891
1    1207
2    1368
3     872
Name: CustomerID, dtype: int64
```

# 결과 분석 및 시각화 (cont'd)

4. 고객 클러스터에서 총 구매 빈도와 총 구매 금액, 마지막 구매 이후 경과일 정보를 추출하고, 구매 1회당 평균 구매 금액도 계산

| | |
|---|---|
| In [36]: | customer_cluster_df = customer_df.drop(['Freq_log', 'SaleAmount_log', 'ElapsedDays_log'], axis = 1, inplace = False) |
| In [37]: | # 주문 1회당 평균 구매금액: SaleAmountAvg<br>customer_cluster_df['SaleAmountAvg'] = customer_cluster_df['SaleAmount']/customer_cluster_df['Freq']<br>customer_cluster_df.head() |
| In [38]: | customer_cluster_df.drop(['CustomerID'], axis = 1, inplace = False).groupby('ClusterLabel').mean() |

| | CustomerID | Freq | SaleAmount | ElapsedDays | ClusterLabel | SaleAmountAvg |
|---|---|---|---|---|---|---|
| 0 | 12346 | 1 | 77183.60 | 326 | 1 | 77183.600000 |
| 1 | 12347 | 182 | 4310.00 | 3 | 3 | 23.681319 |
| 2 | 12348 | 31 | 1797.24 | 76 | 1 | 57.975484 |
| 3 | 12349 | 73 | 1757.55 | 19 | 1 | 24.076027 |
| 4 | 12350 | 17 | 334.40 | 311 | 2 | 19.670588 |

| ClusterLabel | Freq | SaleAmount | ElapsedDays | SaleAmountAvg |
|---|---|---|---|---|
| 0 | 37.811448 | 603.494053 | 20.888889 | 32.256335 |
| 1 | 79.195526 | 1506.813034 | 96.000829 | 102.998219 |
| 2 | 15.052632 | 298.748151 | 188.111842 | 43.338802 |
| 3 | 278.464450 | 7020.739553 | 13.612385 | 96.665836 |