



# 데이터 과학 기반의 파이썬 빅데이터 분석

## Chapter 12 군집 분석

# 목차

01 [K-평균 군집화 분석 + 그래프] 타겟 마케팅을 위한 소비자 군집 분석하기

# 학습목표

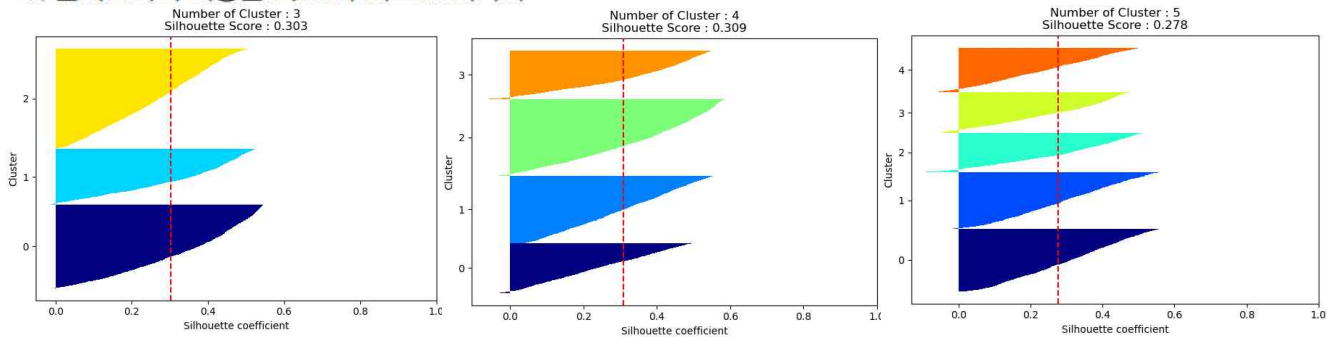
- 머신러닝의 비지도 학습 방식을 이해한다.
- 군집화와 K-평균 알고리즘을 이해한다.
- 군집 분석을 이용하여 소비자 군집을 생성할 수 있다.

# 01. [K-평균 군집화 분석 + 그래프] 타겟 마케팅을 위한 소비자 군집 분석하기

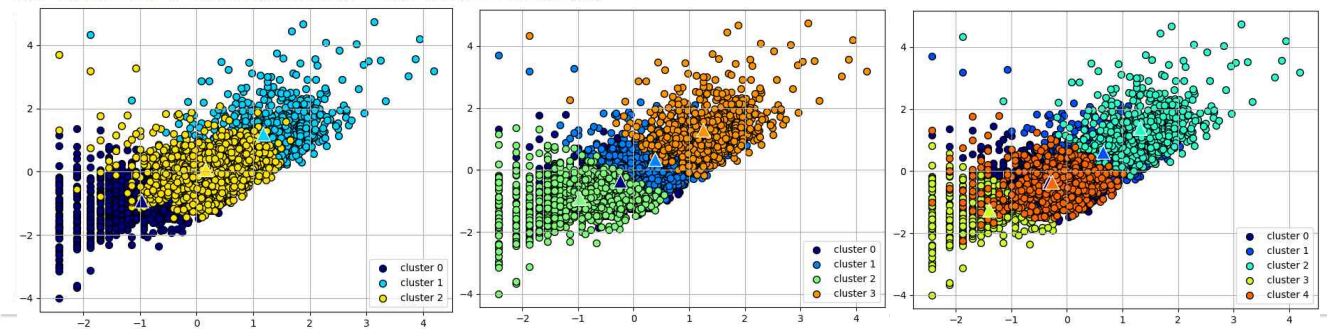
## ■ 분석 미리보기

타겟 마케팅을 위한 소비자 군집 분석하기	
목표	온라인 판매 데이터를 분석하여 타겟 마케팅에 필요한 소비자 군집을 구성한다.
핵심 개념	타겟 마케팅, 비지도 학습, 군집화, K-평균, 엘보우 방법, 실루엣 분석
데이터 수집	온라인 판매 데이터: UCI Machine Learning Repository에서 다운로드
데이터 준비 및 탐색	1. 데이터 정제: 자료형 변환, 오류 및 중복 데이터 제거 2. 데이터프레임의 컬럼 추출 및 분석용 데이터 생성 3. 로그 함수를 이용한 데이터 분포 조정: 데이터 치우침 조정
분석 모델 구축	사이킷런의 K-평균 군집화 모델 구축
결과 시각화	

### 1. 클러스터의 비중을 가로 바 차트로 시각화



### 2. 클러스터의 데이터 분포를 스캐터 차트로 시각화



# 01. [K-평균 군집화 분석 + 그래프] 타겟 마케팅을 위한 소비자 군집 분석하기

## 1 목표 설정

- K-평균으로 온라인 판매 데이터를 분석한 후 타겟 마케팅을 위한 소비자 군집을 만듦

## 2 핵심 개념 이해

### ■ 비지도 학습

- 훈련 데이터에 타겟값이 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

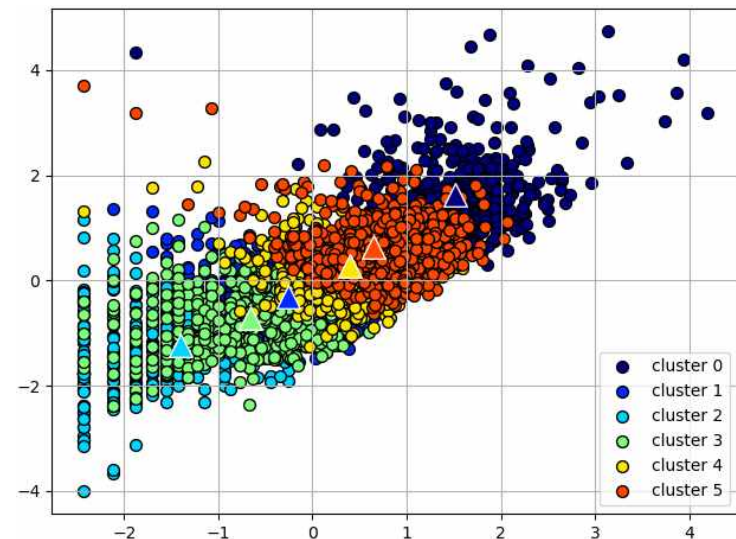


그림 12-2 데이터 군집화의 예

# 01. [K-평균 군집화 분석 + 그래프] 타겟 마케팅을 위한 소비자 군집 분석하기

## 2 핵심 개념 이해

### ■ K-평균 알고리즘

K-평균(K-means) 알고리즘은 k개의 클러스터를 구성하기 위한 것이다. k개의 중심점을 임의의 위치로 잡고 중심점을 기준으로 가까이 있는 데이터를 확인한 뒤 그들과의 거리(유클리디안 거리의 제곱을 사용하여 계산)의 평균 지점으로 중심점을 이동하는 방식이다. 이동한 위치에서 가까이 있는 데이터를 다시 확인하고 그들의 평균 지점으로 중심점을 이동하는 과정을 반복한다. 더 이상 이동이 발생하지 않는 위치를 찾으면 각 중심점을 기준으로 k개의 클러스터가 구성된다. 가장 많이 활용하는 군집화 알고리즘이지만, 클러스터의 수를 나타내는 k를 직접 지정해야 하는 문제가 있다. 적합한 클러스터의 수, 즉 가장 좋은 k를 찾는 데는 엘보 방법이나 실루엣 방법을 사용할 수 있다.

### ■ 엘보 방법

클러스터의 중심점과 클러스터 내의 데이터 거리 차이의 제곱값 합을 왜곡(distortion)이라고 한다. 엘보 방법(elbow method)은 이러한 왜곡을 이용하여 최적의 k를 찾는 것이다. 클러스터의 개수 k의 변화에 따른 왜곡의 변화를 그래프로 그려보면 그래프가 꺾이는 지점인 엘보가 나타나는데, 그 지점의 k를 최적의 k로 선택한다.

# 01. [K-평균 군집화 분석 + 그래프] 타겟 마케팅을 위한 소비자 군집 분석하기

## 2 핵심 개념 이해

### ■ 실루엣 분석

실루엣 분석(silhouette analysis)은 클러스터 내에 있는 데이터가 얼마나 조밀하게 모여있는지를 측정하는 그래프 도구다. 데이터  $i$ 가 해당 클러스터 내의 데이터와 얼마나 가까운가를 나타내는 클러스터 응집력(cluster coherence)  $a(i)$ 와 가장 가까운 다른 클러스터 내의 데이터와 얼마나 떨어져있는가를 나타내는 클러스터 분리도(cluster separation)  $b(i)$ 를 이용하여 실루엣 계수  $s(i)$ 를 계산한다. 실루엣 계수는 -1에서 1 사이의 값을 가지며 1에 가까울수록 좋은 군집화를 의미한다.

$$s(i) = \frac{b(i) - a(i)}{\max(a(i), b(i))}$$

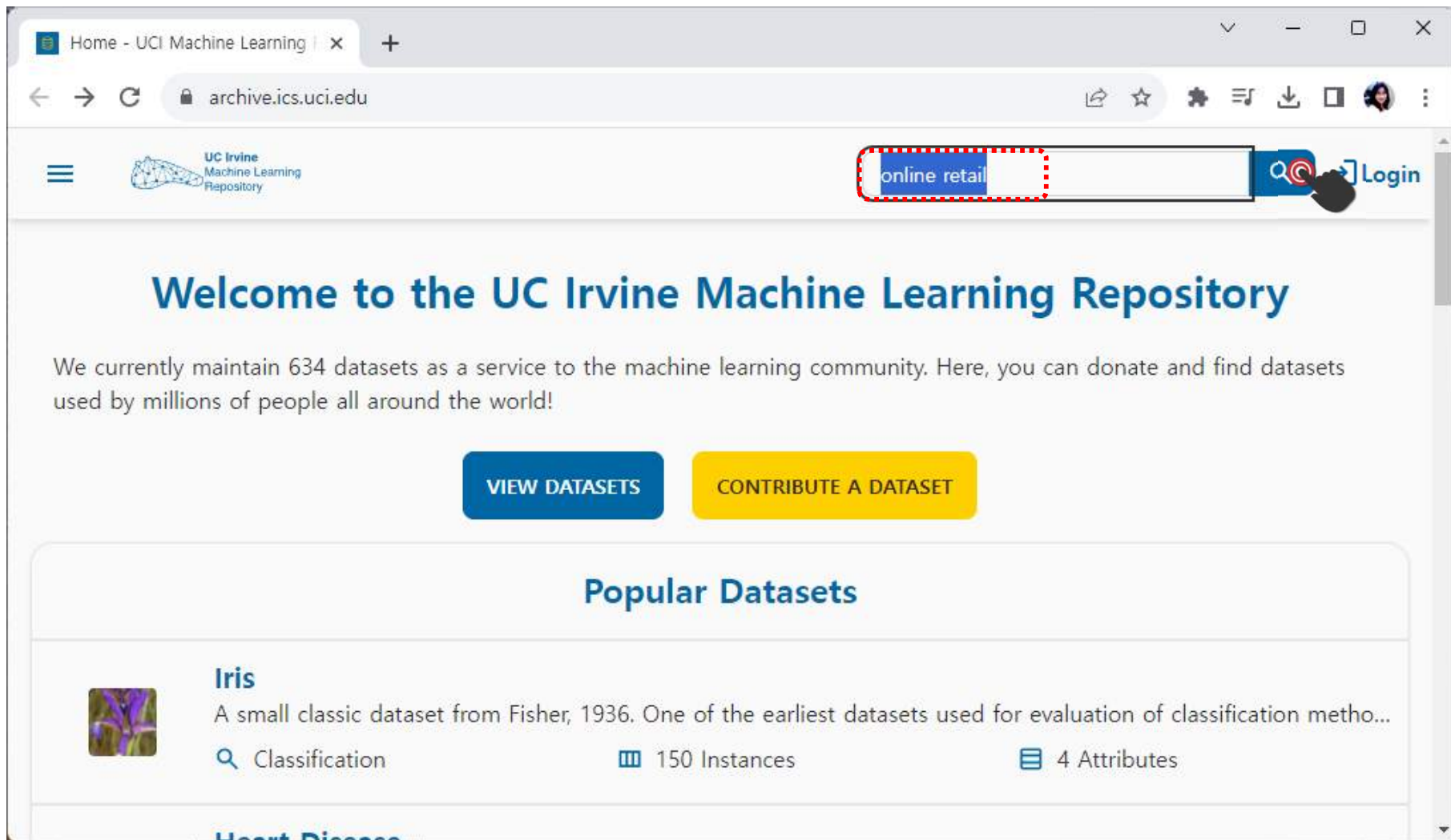


# 01. [K-평균 군집화 분석 + 그래프] 타겟 마케팅을 위한 소비자 군집 분석하기

## 3 데이터 수집

### 1. 온라인 거래 데이터 수집하기

UCI Machine Learning Repository( <https://archive.ics.uci.edu> )에 접속하여 'online retail'을 검색

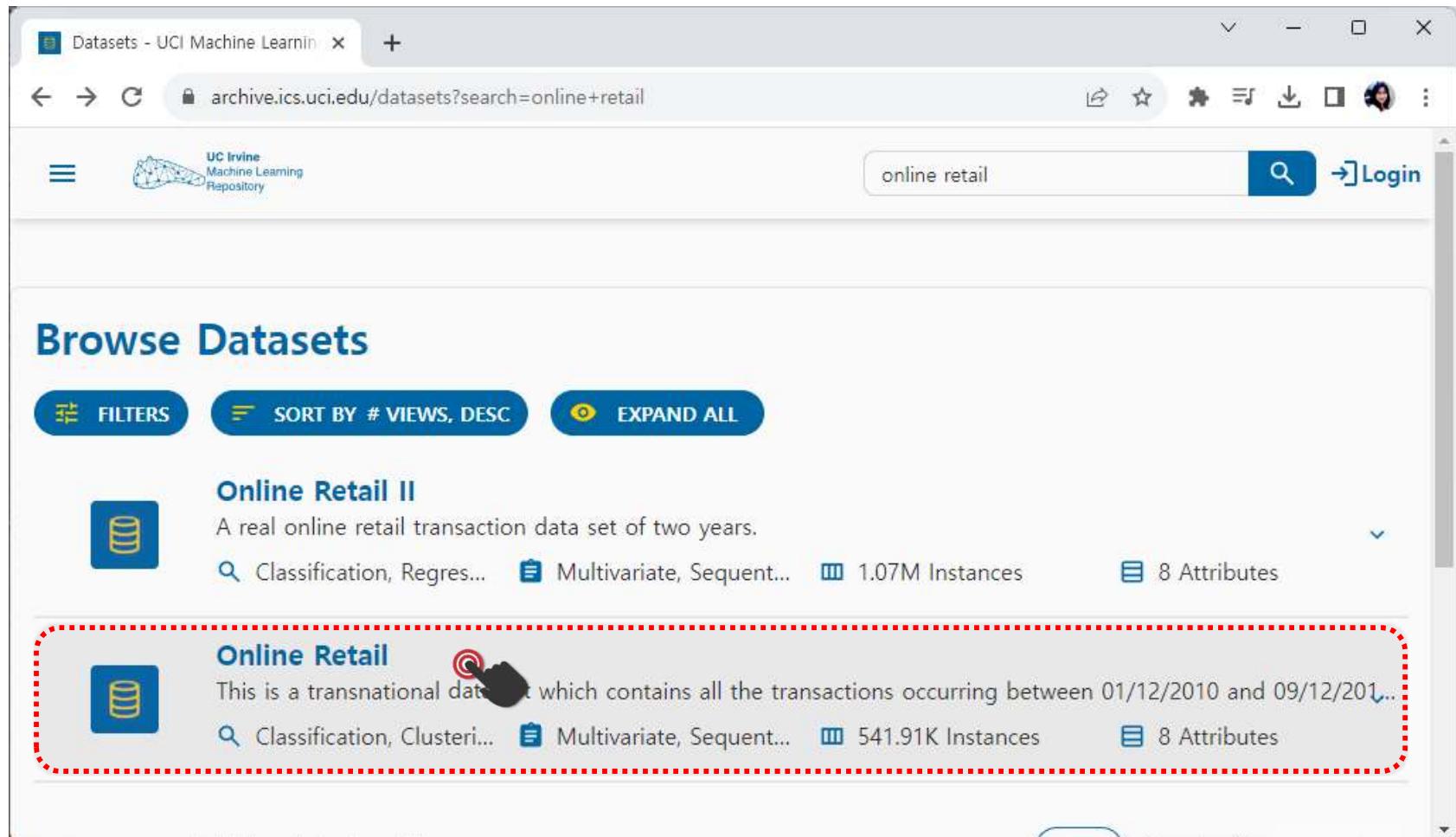




# 01. [K-평균 군집화 분석 + 그래프] 타겟 마케팅을 위한 소비자 군집 분석하기

## 3 데이터 수집

2. 검색 결과 목록에서 'Online Retail Data Set - UCI Machine Learning Repository'를 클릭



# 01. [K-평균 군집화 분석 + 그래프] 타겟 마케팅을 위한 소비자 군집 분석하기

## 3 데이터 수집

3. DOWNLOAD를 클릭하여 online+retail.zip 을 다운로드 받아서 압축풀기.  
→ Online Retail.xlsx

Online Retail - UCI Machine Learning Repository

archive.ics.uci.edu/dataset/352/online+retail

Search datasets... Login

### Online Retail

Donated on 11/5/2015

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.

Dataset Characteristics	Subject Area	Associated Tasks
Multivariate, Sequential, Time-Series	Business	Classification, Clustering

Attribute Type	# Instances	# Attributes
Integer, Real	541909	8

**Information**

**DOI**  
10.24432/C5BW33

**License**  
This dataset is licensed under a Creative Commons Attribution

**DOWNLOAD**

**CITE**

7 citations  
30916 views

# 01. [K-평균 군집화 분석 + 그래프] 타겟 마케팅을 위한 소비자 군집 분석하기

## 3 데이터 수집

4. My\_Python 폴더에 12장\_data 폴더를 만든 뒤 다운로드한 'Online Retail.xlsx' 파일을 옮기고 파일 이름을 'Online\_Retail.xlsx'로 수정

In [1]: 필요한 모듈을 임포트

In [2]: 'Online\_Retail.xlsx' 파일을 로드, 내용을 확인하기 위해 상위 5개 레코드를 표시

```
In [1]: import pandas as pd
import math
```

```
In [2]: retail_df = pd.read_excel('./12장_data/Online_Retail.xlsx')
retail_df.head() #작업 확인용 출력
```

Out [2]:

	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

# 01. [K-평균 군집화 분석 + 그래프] 타겟 마케팅을 위한 소비자 군집 분석하기

## 4 데이터 준비 및 탐색

### 1. 데이터 정제하기

1. 데이터 정보 확인하기 - 데이터에 대한 정보를 확인하기 위해 다음을 입력

In [3]: 데이터셋의 정보를 확인

```
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
```

- 데이터를 구성하는 항목 8개

- 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

# 01. [K-평균 군집화 분석 + 그래프] 타겟 마케팅을 위한 소비자 군집 분석하기

## 4 데이터 준비 및 탐색

### 1. 데이터 정제하기

2. 데이터 정제하기 - Quantity와 UnitPrice가 음수인 것 제거, CustomerID가 없는 데이터는 분석에 사용할 수 없으므로 제거, CustomerID는 정수 형태이므로 자료형을 정수형으로 변경

In [4]: 오류 데이터를 필터링하고 CustomerID의 자료형을 정수형으로 변환`astype(int)`

In [5]: 정리한 결과를 확인하면`retail_df.info()` 데이터는 397,884개

In [6]: 중복 레코드를 제거하면`drop_duplicates()` 데이터는 392,692개

```
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)
```

```
In [5]: retail_df.info()
print(retail_df.isnull().sum())
print(retail_df.shape)

<class 'pandas.core.frame.DataFrame'>
Index: 397884 entries, 0 to 541908
Data columns (total 8 columns):
#   Column          Non-Null Count  Dtype
---  -
0   InvoiceNo        397884 non-null object
1   StockCode       397884 non-null object
2   Description     397884 non-null object
3   Quantity        397884 non-null int64
4   InvoiceDate     397884 non-null datetime64[ns]
5   UnitPrice       397884 non-null float64
6   CustomerID      397884 non-null int32
7   Country         397884 non-null object
dtypes: datetime64[ns](1), float64(1), int32(1), int64(1), object(4)
memory usage: 25.8+ MB
InvoiceNo      0
StockCode      0
Description    0
Quantity       0
InvoiceDate    0
UnitPrice      0
CustomerID     0
Country        0
dtype: int64
(397884, 8)
```

```
In [6]: # 중복 레코드 제거
retail_df.drop_duplicates(inplace=True)

print(retail_df.shape) #작업 확인용 출력

(392692, 8)
```

# 01. [K-평균 군집화 분석 + 그래프] 타겟 마케팅을 위한 소비자 군집 분석하기

## 4 데이터 준비 및 탐색

### 2. 데이터프레임의 컬럼 추출 및 분석용 데이터 생성하기

1. 데이터 탐색을 위해 제품 수, 거래 건수, 고객 수를 알아보고 고객의 국적 확인

In [7]: 개별 제품을 알 수 있는 StockCode의 개수value\_counts()로 제품 수, InvoiceNo의 개수로 거래 건수, CustomerID의 개수로 고객 수를 구함 고객의 수는 4,338명

In [8]: 중복 레코드를 제거하면drop\_duplicates() 데이터는 392,692개

```
In [7]: 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'])
```

Out [7]:

	Product	Transaction	Customer
counts	3665	18532	4338

```
In [8]: retail_df['Country'].value_counts()
```

Out [8]:

```
Country
United Kingdom    349203
Germany           9025
France            8326
EIRE              7226
Spain            2479
Netherlands      2359
Belgium          2031
Switzerland      1841
Portugal         1453
```



# 01. [K-평균 군집화 분석 + 그래프] 타겟 마케팅을 위한 소비자 군집 분석하기

## 4 데이터 준비 및 탐색

### 2. 데이터프레임의 컬럼 추출 및 분석용 데이터 생성하기

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

In [9]: 제품 단가UnitPrice와 주문 개수Quantity를 곱하여 주문 금액SaleAmount을 계산하고 컬럼으로 추가

```
In [9]: # 주문금액 컬럼 추가
retail_df['SaleAmount'] = retail_df['UnitPrice'] * retail_df['Quantity']

retail_df.head() #작업 확인용 출력
```

Out [9]:

	InvoiceNo	StockCode	Description	Quantity	InvoiceDate	UnitPrice	CustomerID	Country	SaleAmount
0	536365	85123A	WHITE HANGING HEART T-LIGHT HOLDER	6	2010-12-01 08:26:00	2.55	17850	United Kingdom	15.30
1	536365	71053	WHITE METAL LANTERN	6	2010-12-01 08:26:00	3.39	17850	United Kingdom	20.34
2	536365	84406B	CREAM CUPID HEARTS COAT HANGER	8	2010-12-01 08:26:00	2.75	17850	United Kingdom	22.00
3	536365	84029G	KNITTED UNION FLAG HOT WATER BOTTLE	6	2010-12-01 08:26:00	3.39	17850	United Kingdom	20.34
4	536365	84029E	RED WOOLLY HOTTIE WHITE HEART.	6	2010-12-01 08:26:00	3.39	17850	United Kingdom	20.34



# 01. [K-평균 군집화 분석 + 그래프] 타겟 마케팅을 위한 소비자 군집 분석하기

## 4 데이터 준비 및 탐색

### 2. 데이터프레임의 컬럼 추출 및 분석용 데이터 생성하기

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

In [10]: 각 고객의 정보를 추출하기 위해 CustomerID를 기준으로 그룹을 만들고 `groupby()`, 주문 횟수를 계산하기 위해 InvoiceNo의 개수 `count`를 구함 주문 금액 `SaleAmount`의 총액 `sum`을 구하고, 주문일 `InvoiceDate` 중에서 가장 최근 날짜 `max`를 찾아 새로운 데이터프레임 객체인 `customer_df`를 생성

```
In [10]: 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
```

Out [10]:

	CustomerID	InvoiceNo	SaleAmount	InvoiceDate
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

# 01. [K-평균 군집화 분석 + 그래프] 타겟 마케팅을 위한 소비자 군집 분석하기

## 4 데이터 준비 및 탐색

### 2. 데이터프레임의 컬럼 추출 및 분석용 데이터 생성하기

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

In [11]: customer\_df의 컬럼 이름을 변경rename()

```
In [11]: # 컬럼이름 바꾸기
customer_df = customer_df.rename(columns = {'InvoiceNo':'Freq', 'InvoiceDate':'ElapsedDays'})

customer_df.head() #작업 확인용 출력
```

Out [11]:

	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

# 01. [K-평균 군집화 분석 + 그래프] 타겟 마케팅을 위한 소비자 군집 분석하기

## 4 데이터 준비 및 탐색

### 2. 데이터프레임의 컬럼 추출 및 분석용 데이터 생성하기

3. 마지막 주문일로부터 며칠이 지났는지에 대한 값을 ElapsedDays 컬럼에 저장

'기준 날짜 - 마지막 구매일'로 계산해 구함 (기준날짜: 2011년 12월 10일)

In [12]: '기준 날짜 - 마지막 구매일'을 계산

In [13]: 마지막 구매 후 몇 일이 지났는지를 날짜수로 환산하여 ElapsedDays 값을 구함

```
In [12]: import datetime

customer_df['ElapsedDays'] = datetime.datetime(2011,12,10) - customer_df['ElapsedDays']

customer_df.head() #작업 확인용 출력
```

Out [12]:

	CustomerID	Freq	SaleAmount	ElapsedDays
0	12346	1	77183.60	325 days 13:59:00
1	12347	182	4310.00	2 days 08:08:00
2	12348	31	1797.24	75 days 10:47:00
3	12349	73	1757.55	18 days 14:09:00
4	12350	17	334.40	310 days 07:59:00

```
In [13]: customer_df['ElapsedDays'] = customer_df['ElapsedDays'].apply(lambda x: x.days+1)

customer_df.head() #작업 확인용 출력
```

Out [13]:

	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

# 01. [K-평균 군집화 분석 + 그래프] 타겟 마케팅을 위한 소비자 군집 분석하기

## 4 데이터 준비 및 탐색

### 3. 데이터 분포 조정하기

1. 마지막 주문일로부터 며칠이 지났는지에 대한 값을 ElapsedDays 컬럼에 저장

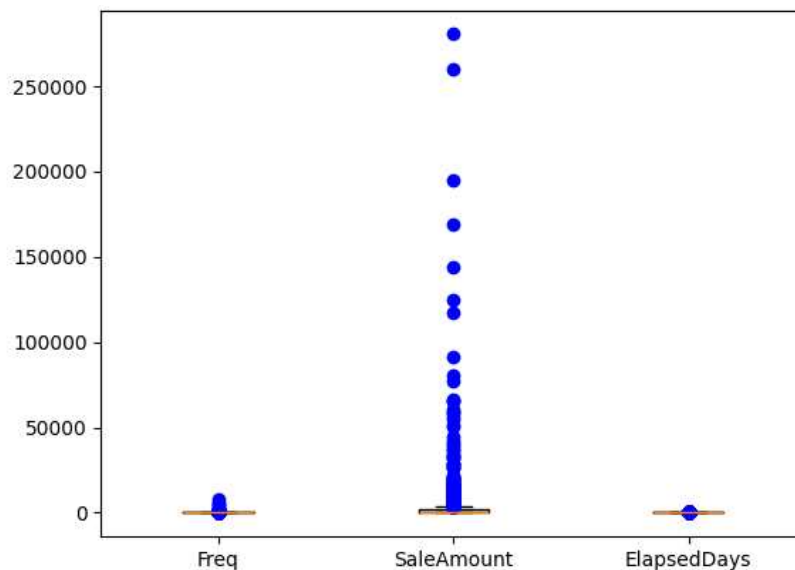
In [14]: customer\_df의 3개 컬럼으로 박스플롯 3개를 그림

- 파란색 점으로 표시된 `sym='bo'` 아웃라이어 값이 많은 것은 데이터 값이 치우침을 나타냄

```
In [14]: 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()
```



# 01. [K-평균 군집화 분석 + 그래프] 타겟 마케팅을 위한 소비자 군집 분석하기

## 4 데이터 준비 및 탐색

### 3. 데이터 분포 조정하기

2. 로그 함수를 적용하여 값의 분포를 고르게 조정

In [15]: 컬럼 값에 로그 함수 `log1p()`를 취한 값을 새 컬럼으로 추가하여 저장

```
In [15]: 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
```

Out [15]:

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

# 01. [K-평균 군집화 분석 + 그래프] 타겟 마케팅을 위한 소비자 군집 분석하기

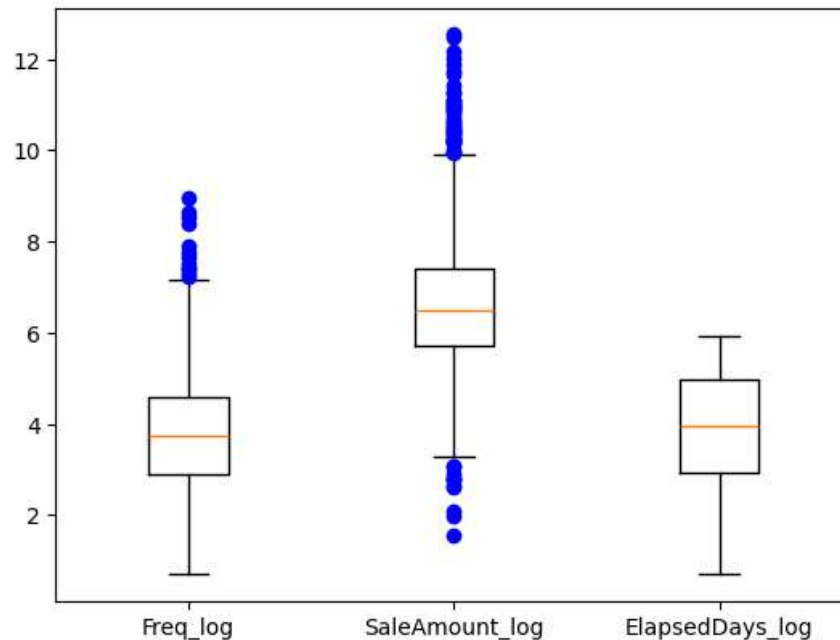
## 4 데이터 준비 및 탐색

### 3. 데이터 분포 조정하기

2. 로그 함수를 적용하여 값의 분포를 고르게 조정

In [16]: 박스플롯을 다시 그림

```
In [16]: # 조정된 데이터 분포를 다시 박스플롯으로 확인하기
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()
```



# 01. [K-평균 군집화 분석 + 그래프] 타겟 마케팅을 위한 소비자 군집 분석하기

## 5 분석 모델 구축

### 1. K-평균 군집화 모델을 이용하여 분석 모델 구축하기

1. X\_features를 정규 분포로 스케일링하기

In [17]: K-평균 군집화 모델링을 위한 KMeans와 실루엣 계수 계산에 사용할 silhouette\_score, silhouette\_samples를 임포트

In [18]: K-평균 모델에 사용할 값을 위해 Freq\_log, SaleAmount\_log, ElapsedDays\_log 컬럼을 X\_features에 저장

In [19]: X\_features를 정규 분포로 스케일링 StandardScaler().fit\_transform하여 X\_features\_scaled에 저장

```
In [17]: from sklearn.cluster import KMeans  
         from sklearn.metrics import silhouette_score, silhouette_samples
```

```
In [18]: X_features = customer_df[['Freq_log', 'SaleAmount_log', 'ElapsedDays_log']].values
```

```
In [19]: # 정규 분포로 다시 스케일링하기  
         from sklearn.preprocessing import StandardScaler  
  
         X_features_scaled = StandardScaler().fit_transform(X_features)
```



# 01. [K-평균 군집화 분석 + 그래프] 타겟 마케팅을 위한 소비자 군집 분석하기

## 5 분석 모델 구축

### 1. K-평균 군집화 모델을 이용하여 분석 모델 구축하기

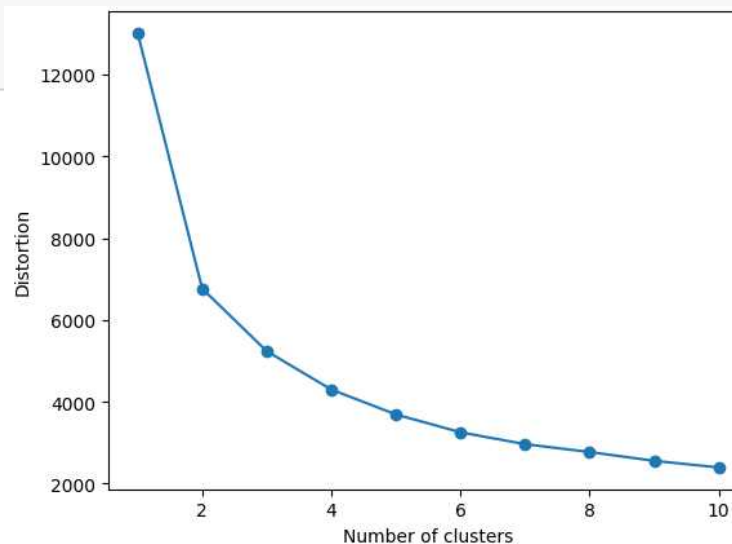
#### 2. 엘보 방법으로 클러스터 개수 k 선택하기

In [20]: K-평균 모델을 생성하고 `KMeans()` 훈련하는 `fit()` 작업을 클러스터의 개수인 1부터 10 까지 반복하면서 왜곡 값 `inertia_`을 리스트 `distortions`에 저장 `append()`  
클러스터 개수에 따른 왜곡 값의 변화를 그래프로 그려서 `plot()` 시각화

```
In [20]: distortions = []

for i in range(1, 11):
    kmeans_i = KMeans(n_clusters=i, random_state=0, n_init='auto') # 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()
```



# 01. [K-평균 군집화 분석 + 그래프] 타겟 마케팅을 위한 소비자 군집 분석하기

## 5 분석 모델 구축

### 1. K-평균 군집화 모델을 이용하여 분석 모델 구축하기

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

In [21]: 클러스터의 개수를 3으로 설정하고 `n_clusters=3` 다시 K-평균 군집화 모델을 생성

생성된 모델에서 `X_features_scaled`를 적용하여 학습하고 클러스터에 대한 레이블 예측 값 `Y_labels`을 구함 `fit_predict()`

In [22]: 레이블 예측값 `Y_labels`을 `customer_df`에 컬럼으로 추가하고 확인

```
In [21]: kmeans = KMeans(n_clusters=3, random_state=0, n_init='auto') # 모델 생성

# 모델 학습과 결과 예측(클러스터 레이블 생성)
Y_labels = kmeans.fit_predict(X_features_scaled)
```

```
In [22]: customer_df['ClusterLabel'] = Y_labels

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	0
1	12347	182	4310.00	3	5.209486	8.368925	1.386294	2
2	12348	31	1797.24	76	3.465736	7.494564	4.343805	0
3	12349	73	1757.55	19	4.304065	7.472245	2.995732	0
	90372					5.815324	5.743003	1

여기서 잠깐 클러스터 번호 정하기

클러스터를 3개로 설정하였으므로 ClusterLabel에 적힌 클러스터 번호가 0~2로 나타난다. 0, 1, 2 중에서 클러스터에 사용하는 번호는 어떻게 정할까? 클러스터 크기 순서대로? 클러스터 위치 순서대로? 클러스터 번호를 정하는 원칙은 없다. Kmeans 모델뿐만 아니라 클러스터링을 하는 비지도 학습 모델은 모두 번호를 할당하는 원칙 없이 임의의 번호를 할당한다. 그러므로 실습하는 시스템에 따라서 ClusterLabel의 값이 책의 결과와 다를 수 있다. 예를 들어, A 컴퓨터에서 2번을 할당한 클러스터에 대해 B 컴퓨터는 0번을 할당할 수 있다. 비지도 학습에 의한 클러스터링은 어떤 것이 같은 클러스터로 분석되었는지, 클러스터에 잠재된 패턴이 무엇인지가 중요한 것이 아니라 클러스터 번호가 0번인지 1번인지는 의미가 없다.

👉 실행 할 때마다 클러스터 번호는 달라질 수 있습니다.

# 01. [K-평균 군집화 분석 + 그래프] 타겟 마케팅을 위한 소비자 군집 분석하기

## 6 결과 분석 및 시각화

### 1. 클러스터의 비중과 데이터 분포를 차트로 시각화하기

1. 각 클러스터의 비중을 가로 바 차트로 시각화

In [23]: 실루엣 계수를 구하고, 각 클러스터의 비중을 가로 바 차트 `barh()`로 시각화하기 위해 `silhouetteViz` 함수를 정의

```
In [23]: from matplotlib import cm

def silhouetteViz(n_cluster, X_features):

    kmeans = KMeans(n_clusters=n_cluster, random_state=0, n_init='auto')
    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()
```

# 01. [K-평균 군집화 분석 + 그래프] 타겟 마케팅을 위한 소비자 군집 분석하기

## 6 결과 분석 및 시각화

### 1. 클러스터의 비중과 데이터 분포를 차트로 시각화하기

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

In [24]: 클러스터에 대한 데이터의 분포를 스캐터 차트 `scatter()`로 시각화하기 위해 cluster Scatter 함수를 정의

```
In [24]: def clusterScatter(n_cluster, X_features):
    c_colors = []
    kmeans = KMeans(n_clusters=n_cluster, random_state=0, n_init='auto')
    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()
```

# 01. [K-평균 군집화 분석 + 그래프] 타겟 마케팅을 위한 소비자 군집 분석하기

## 6 결과 분석 및 시각화

### 1. 클러스터의 비중과 데이터 분포를 차트로 시각화하기

3. In [20]에서 생성한 그래프를 고려하여 클러스터 개수가 3, 4, 5, 6인 경우의 실루엣 계수 와 각 클러스터의 비중, 그리고 데이터 분포를 시각화하여 비교

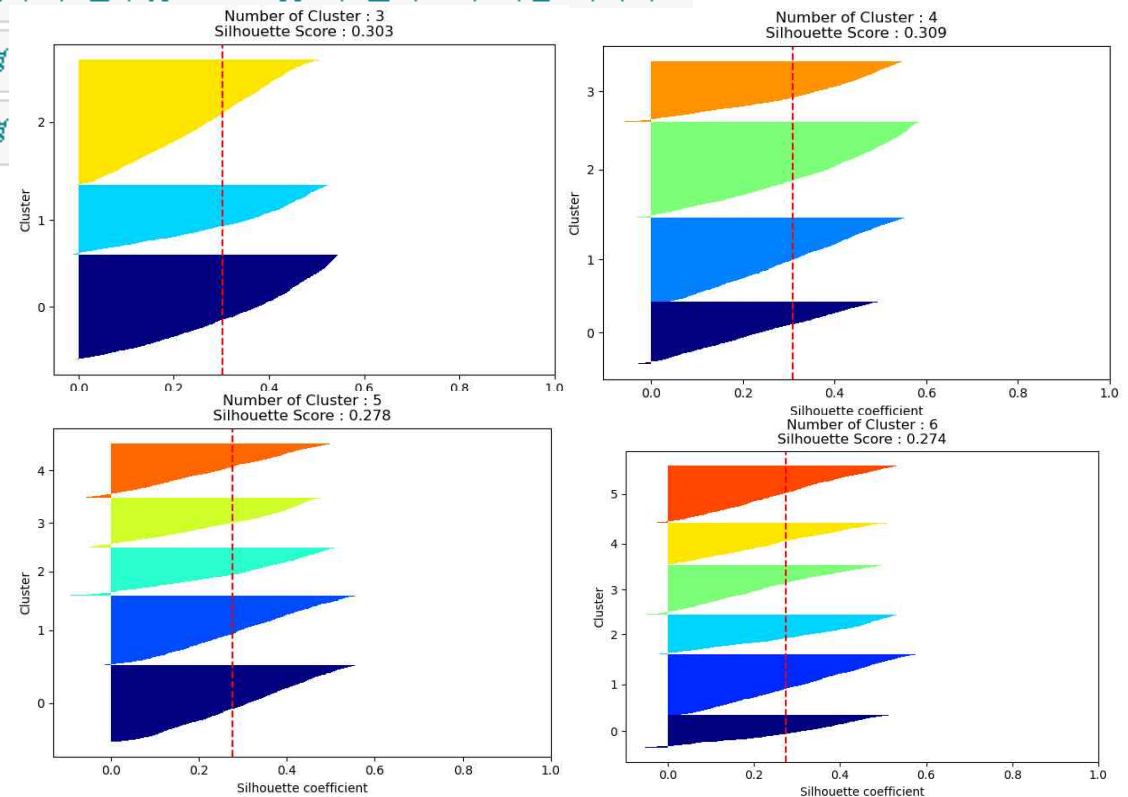
In [25]~[28]: silhouetteViz 함수를 호출하여 실루엣 계수와 클러스터의 비중을 시각화

In [25]: `silhouetteViz(3, X_features_scaled)` #클러스터 3개인 경우의 실루엣 score 및 각 클러스터 비중 시각화

In [26]: `silhouetteViz(4, X_features_scaled)` #클러스터 4개인 경우의 실루엣 score 및 각 클러스터 비중 시각화

In [27]: `silhouetteViz(5, X_features_scaled)` #클러스터 5개인 경우

In [28]: `silhouetteViz(6, X_features_scaled)` #클러스터 6개인 경우



# 01. [K-평균 군집화 분석 + 그래프] 타겟 마케팅을 위한 소비자 군집 분석하기

## 6 결과 분석 및 시각화

### 1. 클러스터의 비중과 데이터 분포를 차트로 시각화하기

4. 클러스터 분포를 이용하여 최적의 클러스터 수를 확인

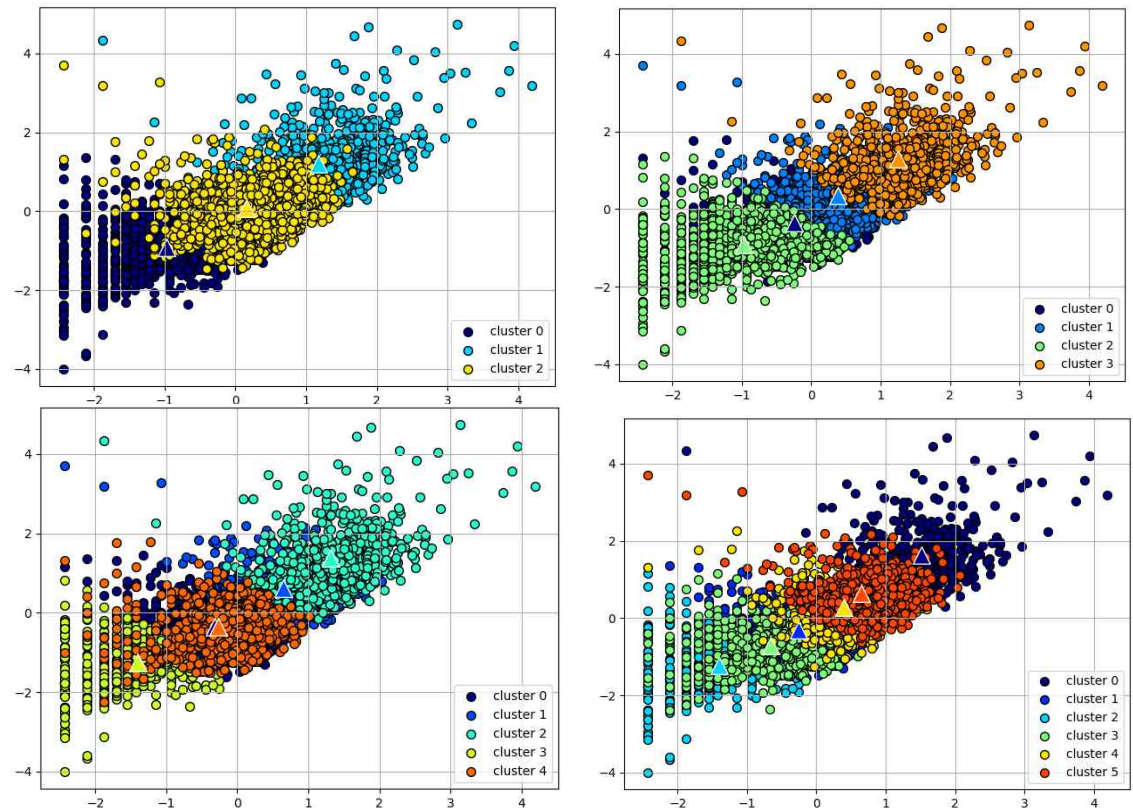
In [29]~[32]: clusterScatter 함수를 호출하여 클러스터의 데이터 분포(원으로 표시)와 클러스터의 중심점 위치(삼각형으로 표시)를 시각화

```
In [29]: clusterScatter(3, X_features_scaled)
```

```
In [30]: clusterScatter(4, X_features_scaled)
```

```
In [31]: clusterScatter(5, X_features_scaled)
```

```
In [32]: clusterScatter(6, X_features_scaled)
```





# 01. [K-평균 군집화 분석 + 그래프] 타겟 마케팅을 위한 소비자 군집 분석하기

## 6 결과 분석 및 시각화

### 1. 클러스터의 비중과 데이터 분포를 차트로 시각화하기

5. silhouetteViz 함수를 호출한 결과에서 클러스터가 4개인 경우가 더 좋은 것으로 나타났으므로 최종적으로 최적의 클러스터 개수 k를 4로 결정

In [33]: 최적의 K-평균 군집화 모델의 레이블 예측값 Y\_labels을 구함

In [34]: 레이블 예측값 Y\_labels을 customer\_df에 저장

In [35]: customer\_df를 CSV 파일로 저장

```
In [33]: best_cluster = 4

kmeans = KMeans(n_clusters=best_cluster, random_state=0, n_init='auto')
Y_labels = kmeans.fit_predict(X_features_scaled)
```

```
In [34]: customer_df['ClusterLabel'] = Y_labels

customer_df.head()    #작업 확인용 출력
```

Out [34]:

	CustomerID	Freq	SaleAmount	ElapsedDays	Freq_log	SaleAmount_log	ElapsedDays_log	ClusterLabel
0	12346	1	77183.60	326	0.693147	11.253955	5.789960	3
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	3
3	12349	73	1757.55	19	4.304065	7.472245	2.995732	3
4	12350	17	334.40	311	2.890372	5.815324	5.743003	0

- ClusterLabel이 추가된 데이터를 파일로 저장

```
In [35]: customer_df.to_csv('./12장_data/Online_Retail_Customer_Cluster.csv')
```

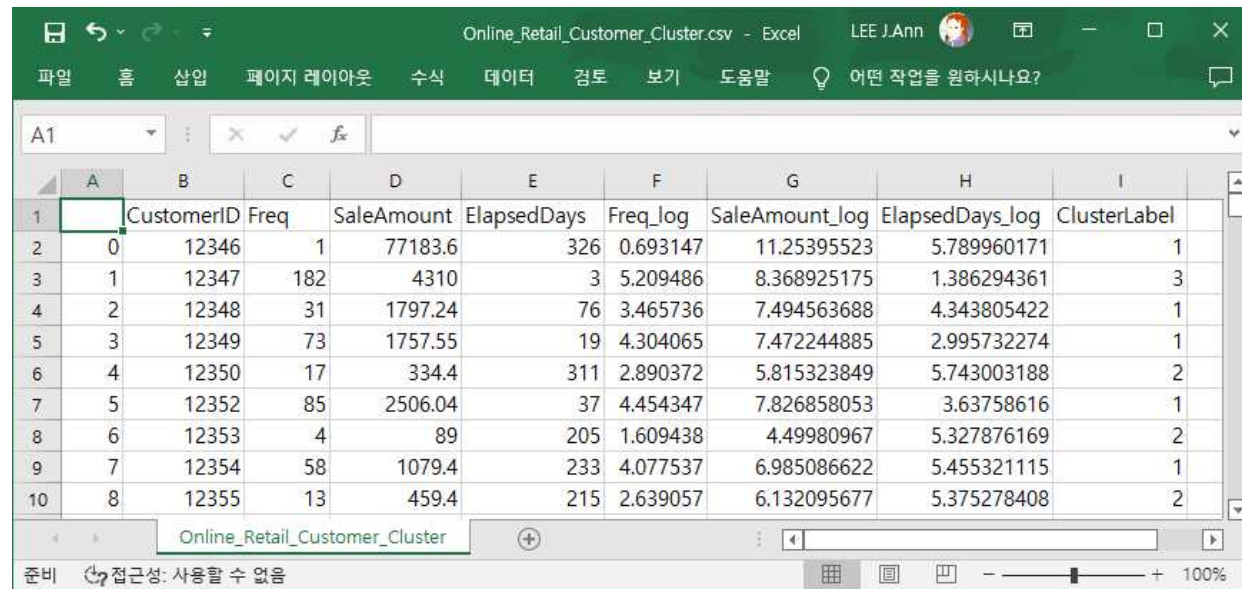


# 01. [K-평균 군집화 분석 + 그래프] 타겟 마케팅을 위한 소비자 군집 분석하기

## 6 결과 분석 및 시각화

### 1. 클러스터의 비중과 데이터 분포를 차트로 시각화하기

- '타겟 마케팅에 필요한 소비자 군집'



	A	B	C	D	E	F	G	H	I
1		CustomerID	Freq	SaleAmount	ElapsedDays	Freq_log	SaleAmount_log	ElapsedDays_log	ClusterLabel
2	0	12346	1	77183.6	326	0.693147	11.25395523	5.789960171	1
3	1	12347	182	4310	3	5.209486	8.368925175	1.386294361	3
4	2	12348	31	1797.24	76	3.465736	7.494563688	4.343805422	1
5	3	12349	73	1757.55	19	4.304065	7.472244885	2.995732274	1
6	4	12350	17	334.4	311	2.890372	5.815323849	5.743003188	2
7	5	12352	85	2506.04	37	4.454347	7.826858053	3.63758616	1
8	6	12353	4	89	205	1.609438	4.49980967	5.327876169	2
9	7	12354	58	1079.4	233	4.077537	6.985086622	5.455321115	1
10	8	12355	13	459.4	215	2.639057	6.132095677	5.375278408	2

그림 12-8 완성된 소비자 군집 파일

# 01. [K-평균 군집화 분석 + 그래프] 타겟 마케팅을 위한 소비자 군집 분석하기

## 6 결과 분석 및 시각화

### 2. 추가 분석하기

1. 클러스터의 특징을 살펴보기 위해 먼저 ClusterLabel을 기준으로 그룹을 만듦

```
In [36]: customer_df.groupby('ClusterLabel')['CustomerID'].count()
```

```
Out[36]: ClusterLabel
0      1370
1       868
2       893
3      1207
Name: CustomerID, dtype: int64
```

전체 고객 4,338명 중에서 클러스터 0은 1,370명이고 클러스터 1은 868명, 클러스터 2는 893명, 클러스터 3은 1,207명으로 구성

# 01. [K-평균 군집화 분석 + 그래프] 타겟 마케팅을 위한 소비자 군집 분석하기

## 6 결과 분석 및 시각화

### 2. 추가 분석하기

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

In [37]:	customer_cluster_df = customer_df.drop(['Freq_log', 'SaleAmount_log', 'ElapsedDays_log'],axis = 1, inplace = False)																																										
In [38]:	<div>#주문 1회당 평균 구매금액: SaleAmountAvg</div> <div>customer_cluster_df['SaleAmountAvg'] = customer_cluster_df['SaleAmount']/customer_cluster_df['Freq']</div> <div>customer_cluster_df.head()</div>																																										
Out[38]:	<table><thead><tr><th></th><th>CustomerID</th><th>Freq</th><th>SaleAmount</th><th>ElapsedDays</th><th>ClusterLabel</th><th>SaleAmountAvg</th></tr></thead><tbody><tr><td>0</td><td>12346</td><td>1</td><td>77183.60</td><td>326</td><td>3</td><td>77183.600000</td></tr><tr><td>1</td><td>12347</td><td>182</td><td>4310.00</td><td>3</td><td>1</td><td>23.681319</td></tr><tr><td>2</td><td>12348</td><td>31</td><td>1797.24</td><td>76</td><td>3</td><td>57.975484</td></tr><tr><td>3</td><td>12349</td><td>73</td><td>1757.55</td><td>19</td><td>3</td><td>24.076027</td></tr><tr><td>4</td><td>12350</td><td>17</td><td>334.40</td><td>311</td><td>0</td><td>19.670588</td></tr></tbody></table>		CustomerID	Freq	SaleAmount	ElapsedDays	ClusterLabel	SaleAmountAvg	0	12346	1	77183.60	326	3	77183.600000	1	12347	182	4310.00	3	1	23.681319	2	12348	31	1797.24	76	3	57.975484	3	12349	73	1757.55	19	3	24.076027	4	12350	17	334.40	311	0	19.670588
	CustomerID	Freq	SaleAmount	ElapsedDays	ClusterLabel	SaleAmountAvg																																					
0	12346	1	77183.60	326	3	77183.600000																																					
1	12347	182	4310.00	3	1	23.681319																																					
2	12348	31	1797.24	76	3	57.975484																																					
3	12349	73	1757.55	19	3	24.076027																																					
4	12350	17	334.40	311	0	19.670588																																					
In [39]:	customer_cluster_df.drop(['CustomerID'],axis = 1, inplace = False).groupby('ClusterLabel').mean()																																										
Out[39]:	<table><thead><tr><th></th><th>Freq</th><th>SaleAmount</th><th>ElapsedDays</th><th>SaleAmountAvg</th></tr></thead><tbody><tr><td>ClusterLabel</td><td></td><td></td><td></td><td></td></tr><tr><td>0</td><td>15.100000</td><td>298.966147</td><td>188.241606</td><td>43.290636</td></tr><tr><td>1</td><td>279.207373</td><td>7031.952834</td><td>13.479263</td><td>96.921011</td></tr><tr><td>2</td><td>37.793953</td><td>603.425354</td><td>20.959686</td><td>32.226856</td></tr><tr><td>3</td><td>79.455675</td><td>1520.324252</td><td>95.595692</td><td>103.086284</td></tr></tbody></table>		Freq	SaleAmount	ElapsedDays	SaleAmountAvg	ClusterLabel					0	15.100000	298.966147	188.241606	43.290636	1	279.207373	7031.952834	13.479263	96.921011	2	37.793953	603.425354	20.959686	32.226856	3	79.455675	1520.324252	95.595692	103.086284												
	Freq	SaleAmount	ElapsedDays	SaleAmountAvg																																							
ClusterLabel																																											
0	15.100000	298.966147	188.241606	43.290636																																							
1	279.207373	7031.952834	13.479263	96.921011																																							
2	37.793953	603.425354	20.959686	32.226856																																							
3	79.455675	1520.324252	95.595692	103.086284																																							

고객 클러스터 1은 다른 클러스터보다 구매 횟수가 월등히 높지만 구매당 평균 금액은 두 번째로 높음.  
구매당 평균 금액은 고객 클러스터 3이 가장 높음



**감사합니다.**