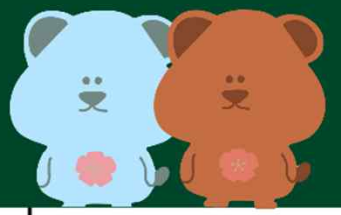


12주차 발표

여유진 여채윤 이가영

EWHA EURON

목차

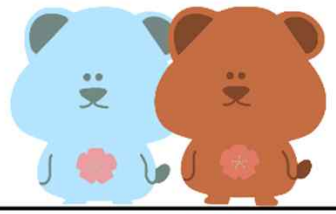
#01 Online Retail

#02 Customer Segmentation

#03 Mall Customer Segmentation Data

#04 Breast Cancer Wisconsin (Diagnostic) Data Set

#05 H&M recommendation



EWHA EURON

1. Online Retail

EWHA EURON

1.1 고객 세그먼테이션의 정의와 기법

✓ 고객 세그먼테이션 (Customer Segmentation) 이란? : 다양한 기준으로 고객을 분류하는 기법, CRM이나 마케팅의 중요 기반 요소

✓ 고객을 분류하는 요소:

1. 지역/결혼 여부/성별/소득 같은 개인의 신상 데이터
2. **어떤 상품을 얼마나 많은 비용을 써서 얼마나 자주 사용하는가**

-> 고객분류를 사용하는 비즈니스는 대부분 상품 판매에 중점을 둔다

✓ 고객 세그먼테이션의 목표: **타깃 마케팅**

- 고객을 여러 특성에 맞게 세분화해서 그 유형에 따라 맞춤형 마케팅이나 서비스를 제공하는 것

✓ **RFM 기법**

- RECENCY : 가장 최근 상품 구입 일에서 오늘까지의 기간
- FREQUENCY : 상품 구매 횟수
- MONETARY VALUE : 총 구매 금액

1.2 데이터 세트 로딩과 데이터 클렌징

데이터 칼럼

```
import pandas as pd
import datetime
import math
import numpy as np
import matplotlib.pyplot as plt
%matplotlib inline

os.chdir("/temp")

retail = pd.read_csv('OnlineRetail.csv', sep=",", encoding="ISO-8859-1", header=0)
retail.head(3)
```

✓ 0.8s Python

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

- InvoiceNo: 주문번호 (c로 시작하는 경우 취소 주문)
- StockCode: 제품 코드
- Description: 제품 설명
- Quantity: 주문 제품 건수
- InvoiceDate: 주문 일자
- UnitPrice: 제품 단가
- CustomerID: 고객 번호
- Country: 국가명(주문 고객의 국적)

EWHA EURON

1.2 데이터 세트 로딩과 데이터 클렌징

Null 값 제거

```
retail.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
```

1. 오류 데이터:
   Quantity와 UnitPrice가 0보다 작은 데이터 삭제
2. 고객 식별 번호가 없는 데이터 삭제
3. 영국이 대다수이므로 다른 국가의 데이터 삭제

```
retail = retail[retail['Quantity'] > 0]
retail = retail[retail['UnitPrice'] > 0]
retail = retail[retail['CustomerID'].notnull()]
print(retail.shape)
retail.isnull().sum()
```

```
(397884, 8)

InvoiceNo      0
StockCode      0
Description    0
Quantity       0
InvoiceDate    0
UnitPrice      0
CustomerID     0
Country        0
dtype: int64
```

```
retail['Country'].value_counts()[:5]
```

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

결과:
전체 데이터가 541,909개 -> 354,321개
Null 값 존재하지 않는다
영국의 데이터만 존재한다.

EWHA EURON

1.3 RFM 기반 데이터 가공

주문 금액 데이터 생성 & CustomerNo int화

```
retail['sale_amount'] = retail['Quantity'] * retail['UnitPrice']
retail['CustomerID'] = retail['CustomerID'].astype(int)
```

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

```
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
```

```
retail.groupby(['InvoiceNo','StockCode'])['InvoiceNo'].count().mean()
```

```
1.028702077315023
```

EWHA EURON

1.3 RFM 기반 데이터 가공

RFM 칼럼 생성

```
# DataFrame의 groupby() 의 multiple 연산을 위해 agg() 이용
# Recency는 InvoiceDate 컬럼의 max() 에서 데이터 가공
# Frequency는 InvoiceNo 컬럼의 count() , Monetary value는 sale_amount 컬럼의 sum()
aggregations = {
    'InvoiceDate': 'max',
    'InvoiceNo': 'count',
    'sale_amount':'sum'
}
cust = retail.groupby('CustomerID').agg(aggregations)
# groupby된 결과 컬럼값을 Recency, Frequency, Monetary로 변경
cust = cust.rename(columns = {'InvoiceDate':'Recency',
                              'InvoiceNo':'Frequency',
                              'sale_amount':'Monetary'
                             }
                  )
cust = cust.reset_index()
cust.head(3)
```

✓ 0.4s

	CustomerID	Recency	Frequency	Monetary
0	12346	18-01-2011 10:01	1	77183.60
1	12747	28-06-2011 10:06	103	4196.01
2	12748	31-07-2011 15:04	4595	33719.73

```
import datetime as dt
cust['Recency'] = pd.to_datetime(cust['Recency'],format='%d-%m-%Y %H:%M')
cust['Recency'] = (dt.datetime(2011,12,10) - cust['Recency'])
cust['Recency'] = cust['Recency'].apply(lambda x: x.days+1)
print('cust 로우와 컬럼 건수는 ',cust.shape)
cust.head(3)
```

✓ 0.2s

```
cust 로우와 컬럼 건수는  (3920, 4)
```

	CustomerID	Recency	Frequency	Monetary
0	12346	326	1	77183.60
1	12747	165	103	4196.01
2	12748	132	4595	33719.73

✓ RECENCY, FREQUENCY, MONETARY 칼럼이 생겼음을 확인할 수 있다.

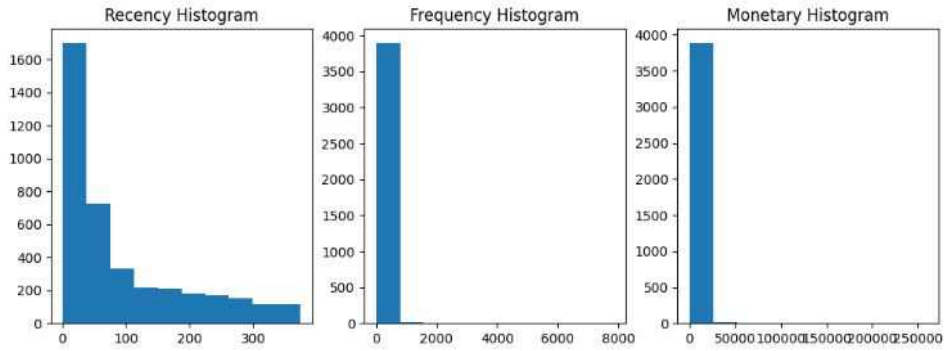
EWHA EURON

1.4 RFM 기반 고객 세그먼테이션

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

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

ax3.set_title('Monetary Histogram')
ax3.hist(cust['Monetary'])
```

Python

```
(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>)
```



✓ 데이터는 소매업체의 대규모 주문을 포함해서 주문 횟수와 주문 금액에서 개인 고객 주문과 큰 차이를 나타난다. 즉, 데이터가 왜곡되어 있다.

-> RECENCY FREQUENCY MONETARY VALUE
모두 왜곡된 데이터 값 분포도를 가지고 있다

```python
cust[['Recency','Frequency','Monetary']].describe()
```

	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

1.4 RFM 기반 고객 세그먼테이션

StandardScaling 후 K-평균 군집

```
from sklearn.preprocessing import StandardScaler
from sklearn.cluster import KMeans
from sklearn.metrics import silhouette_score, silhouette_samples

X_features = cust[['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['cluster_label'] = labels

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

✓ 2.1s

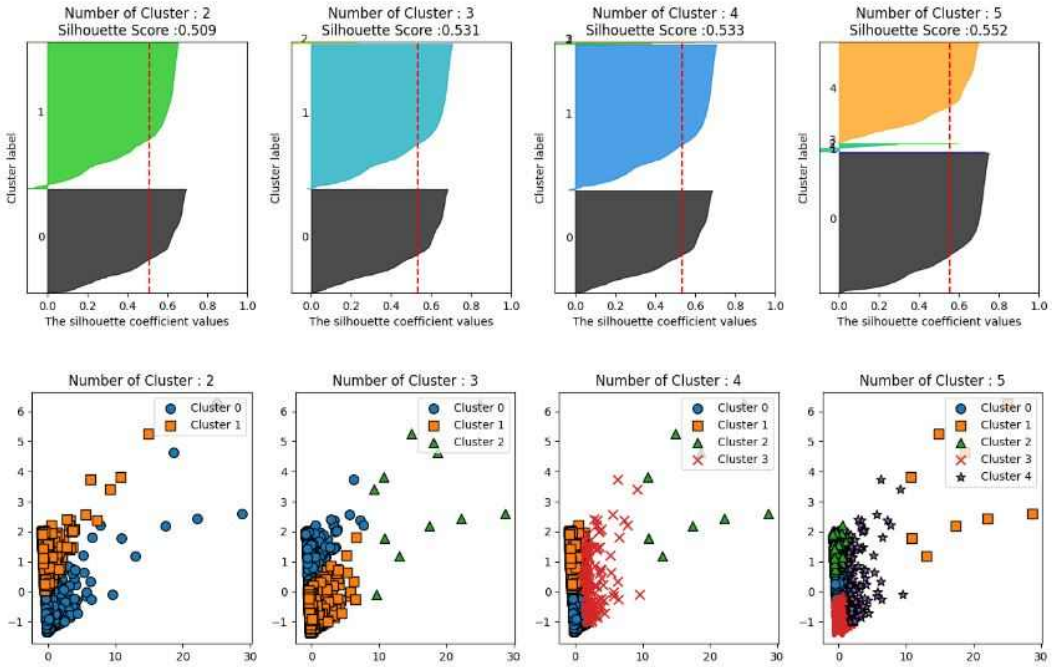
```
실루엣 스코어는 : 0.531
```

✓ 데이터 값이 거리 기반으로 광범위하게 퍼져 있어서 군집 수를 늘려도 개선이 되지 않고 해당 군집만 지속적으로 분리해 의미 없는 군집화 결과로 이어진다.

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

✓ 4.3s Python

```
C:\Users\yujin\AppData\Local\Temp\ipykernel_5676\3186299039.py:36: UserWarning: You passed a edgecolor/edgecolors ('k') for an unfilled marker ('x').  Matplotlib is ignoring the edgecolor in favor of the facecolor.  This behavior may change in the future.
  axs[ind].scatter(x=label_df['PCA1'], y=label_df['PCA2'], s=70,\
C:\Users\yujin\AppData\Local\Temp\ipykernel_5676\3186299039.py:36: UserWarning: You passed a edgecolor/edgecolors ('k') for an unfilled marker ('x').  Matplotlib is ignoring the edgecolor in favor of the facecolor.  This behavior may change in the future.
  axs[ind].scatter(x=label_df['PCA1'], y=label_df['PCA2'], s=70,\
```



1.4 RFM 기반 고객 세그먼테이션

로그 변환 후 K-평균 군집

```
### Log 변환을 통해 데이터 변환
from sklearn.preprocessing import StandardScaler
from sklearn.cluster import KMeans
from sklearn.metrics import silhouette_score, silhouette_samples

# Recency, Frequecny, Monetary 컬럼에 np.log1p() 로 Log Transformation
cust['Recency_log'] = np.log1p(cust['Recency'])
cust['Frequency_log'] = np.log1p(cust['Frequency'])
cust['Monetary_log'] = np.log1p(cust['Monetary'])

# Log Transformation 데이터에 StandardScaler 적용
X_features = cust[['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['cluster_label'] = labels

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

✓ 0.4s

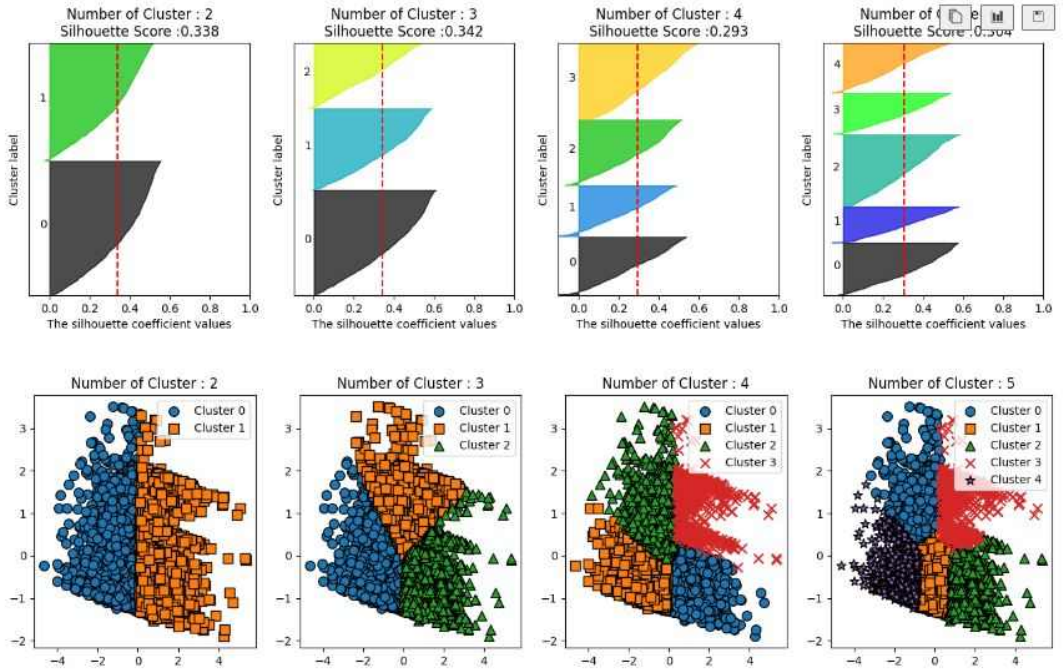
```
실루엣 스코어는 : 0.342
```

✓ 실루엣 스코어는 더 떨어지지만, 그럼에도 더 균일한 군집화가 구성될 수 있다.

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

✓ 3.8s Python

```
C:\Users\yujin\AppData\Local\Temp\ipykernel_5676\3186299039.py:36: UserWarning: You passed a edgecolor/edgecolors ('k') for an unfilled marker ('x').  Matplotlib is ignoring the edgecolor in favor of the facecolor.  This behavior may change in the future.
  axs[ind].scatter(x=label_df['PCA1'], y=label_df['PCA2'], s=70,\
C:\Users\yujin\AppData\Local\Temp\ipykernel_5676\3186299039.py:36: UserWarning: You passed a edgecolor/edgecolors ('k') for an unfilled marker ('x').  Matplotlib is ignoring the edgecolor in favor of the facecolor.  This behavior may change in the future.
  axs[ind].scatter(x=label_df['PCA1'], y=label_df['PCA2'], s=70,\
```

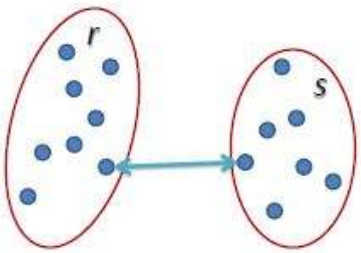


1.5 계층적 군집화

계층적 군집화 – 여러 개의 군집 중에서 가장 유사도가 높은 혹은 거리가 가까운 군집 두 개를 선택하여 하나로 합치면서 군집 개수를 줄여 가는 방법

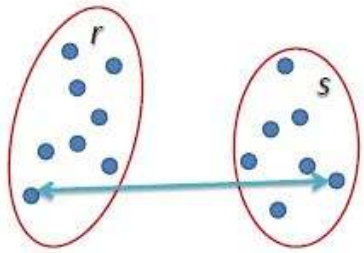
- 군집 구성법

1. 최단거리법(single) :
생성된 군집에서 중심과 거리가 가까운 데이터끼리 비교하여 가까운 데이터끼리 군집화



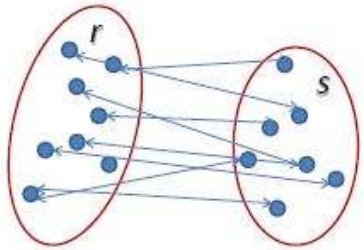
$$L(r,s) = \min(D(x_{ri}, x_{sj}))$$

2. 최장거리법(complete) :
생성된 군집에서 중심과 거리가 먼 데이터끼리 비교하여 가장 가까운 데이터끼리 군집화



$$L(r,s) = \max(D(x_{ri}, x_{sj}))$$

3. 평균기준법(average) :
한 군집 안에 속한 모든 데이터와 다른 군집에 속한 모든 데이터의 두 집단에 대한 거리 평균을 계산하여 가까운 데이터끼리 군집화



$$L(r,s) = \frac{1}{n_r n_s} \sum_{i=1}^{n_r} \sum_{j=1}^{n_s} D(x_{ri}, x_{sj})$$

1.5 계층적 군집화

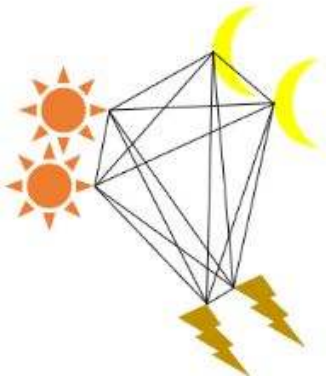
계층적 군집화 생성 과정

1) 기존 관측치들끼리의 거리를 측정한다

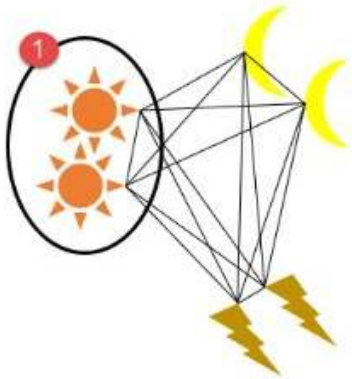
2) 가장 가까운 관측치들을 1번 클러스터로 묶는다 -> 클러스터 1 생성

3) 클러스터 1과 나머지 관측치들끼리도 거리를 측정 (군집 구성법에 따라 다르다)

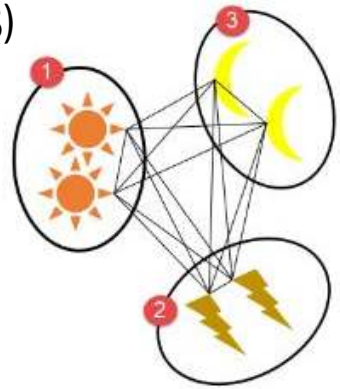
1)



2)



3)



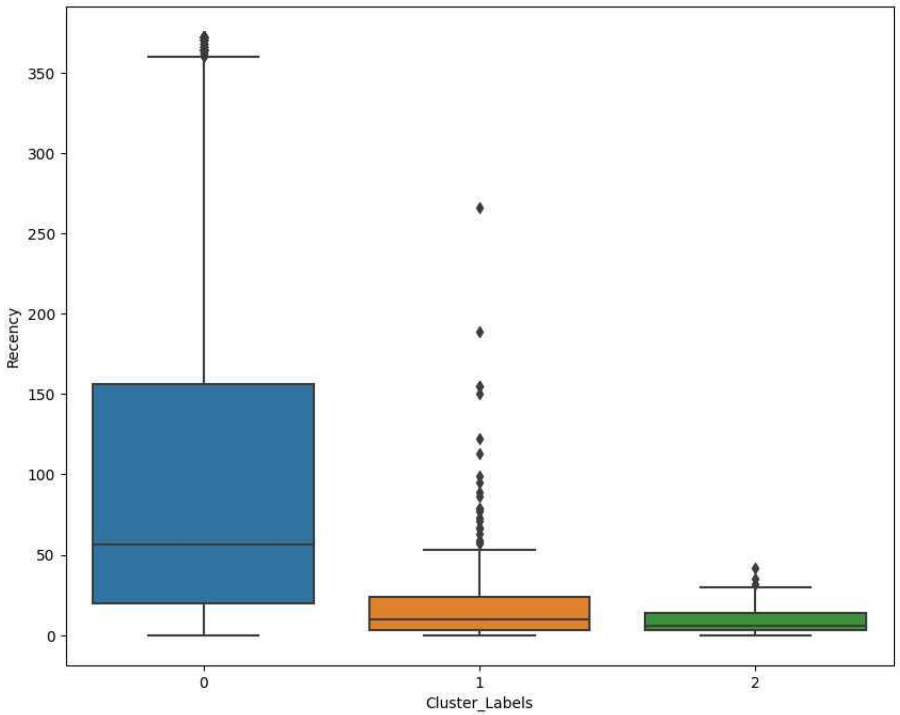
1.5 계층적 군집화

계층적 군집화 - RECENCY

```
# Plot Cluster Id vs Recency
sns.boxplot(x='Cluster_Labels', y='Recency', data=rfm)
```

0.2s Python

```
<AxesSubplot:xlabel='Cluster_Labels', ylabel='Recency'>
```

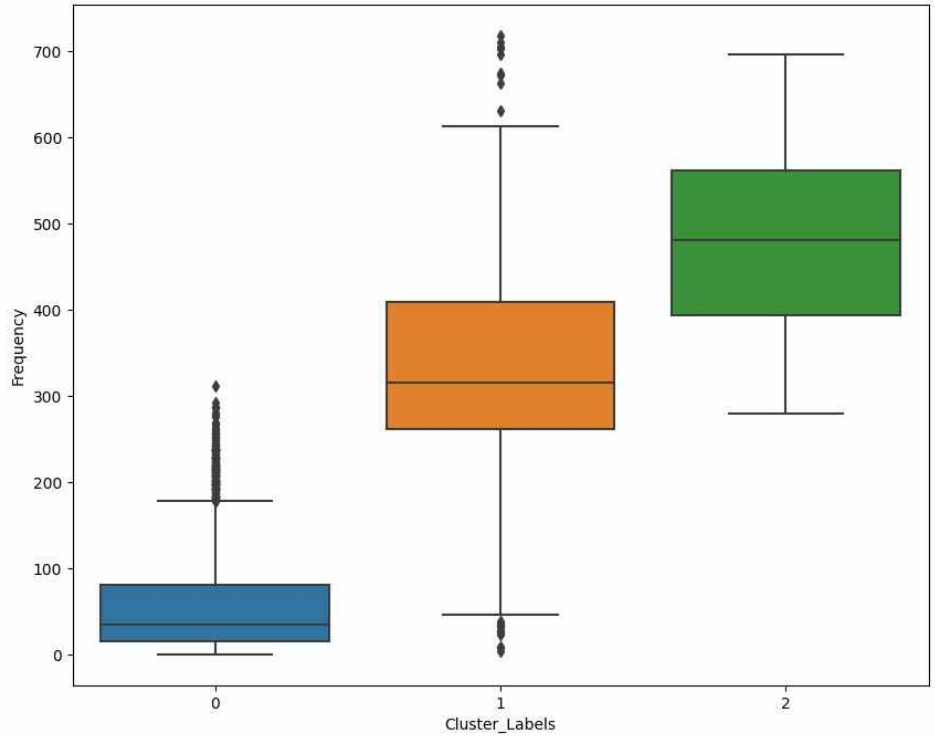


FREQUENCY

```
# Plot Cluster Id vs Frequency
sns.boxplot(x='Cluster_Labels', y='Frequency', data=rfm)
```

0.2s Python

```
<AxesSubplot:xlabel='Cluster_Labels', ylabel='Frequency'>
```



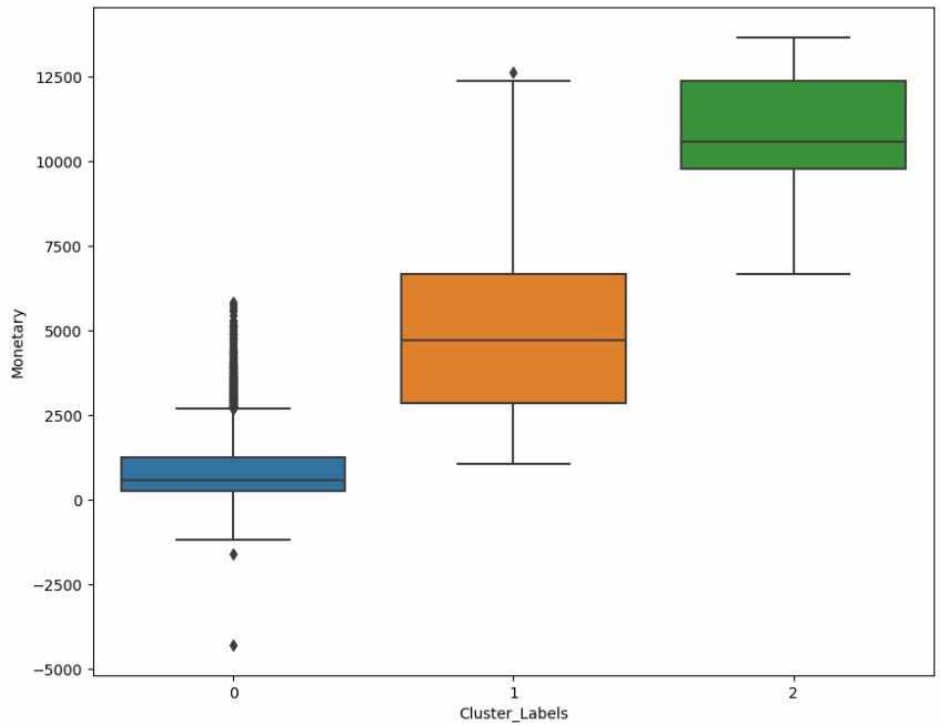
1.5 계층적 군집화

MONETARY VALUE

```
# Plot Cluster Id vs Monetary

sns.boxplot(x='Cluster_Labels', y='Monetary', data=rfm)
```

0.2s Python

```
<AxesSubplot:xlabel='Cluster_Labels', ylabel='Monetary'>
```



계층적 군집화 결론

- 군집 0의 고객들은 최근 구매자가 아니므로 비즈니스 관점으로 보았을 때 중요한 고객이 아니다
- 군집 2의 고객들이 제일 자주 구매한다
- 군집 2의 고객은 다른 고객에 비해 거래가 많다

EWHA EURON

2. Customer Segmentation

EWHA EURON

2.1 프로젝트 소개

소개:
마켓 고객을 군집화,
고객이 비즈니스에 미치는 중요성을 최적화할 수 있도록 군집화 한다.

EWHA EURON

2.2 Loading Data (데이터 세트 로딩)

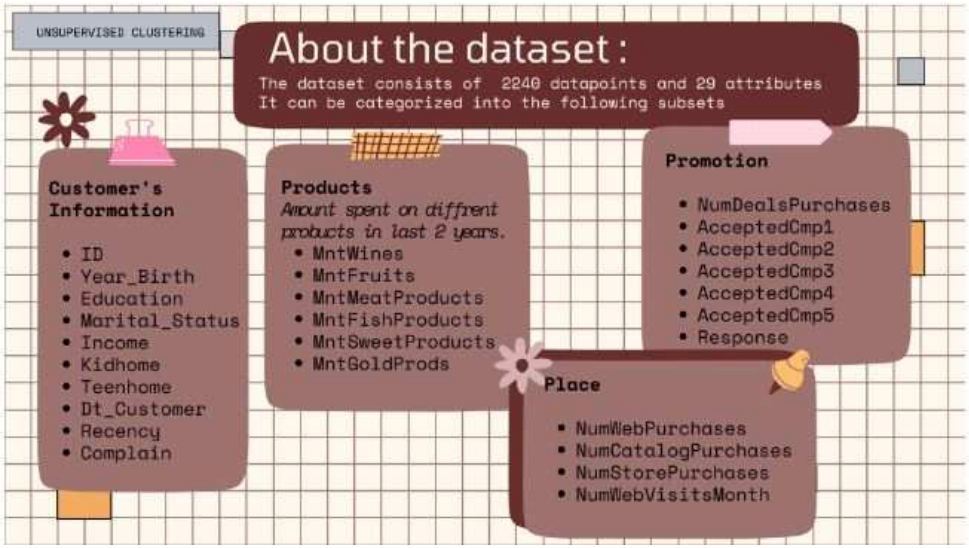
```python
#Importing the Libraries
import datetime
import sys
import warnings

import matplotlib
import matplotlib.pyplot as plt
import numpy as np
import pandas as pd
import seaborn as sns
from matplotlib import colors
from matplotlib.colors import ListedColormap
from mpl_toolkits.mplot3d import Axes3D
from sklearn import metrics
from sklearn.cluster import AgglomerativeClustering, KMeans
from sklearn.decomposition import PCA
from sklearn.preprocessing import LabelEncoder, StandardScaler
from yellowbrick.cluster import KElbowVisualizer

if not sys.warnoptions:
    warnings.simplefilter("ignore")
np.random.seed(42)
```

```python
#Loading the dataset

os.chdir("/temp")

data = pd.read_csv("marketing_campaign.csv", sep="\t")
print("Number of datapoints:", len(data))
data.head()
```

```
Number of datapoints: 2240
```

ID	Year_Birth	Education	Marital_Status	Income	Kidhome	Teenhome	Dt_Customer	Recency	MntWines	...
5524	1957	Graduation	Single	58138.0	0	0	04-09-2012	58	635	...
2174	1954	Graduation	Single	46344.0	1	1	08-03-2014	38	11	...
4141	1965	Graduation	Together	71613.0	0	0	21-08-2013	26	426	...
6182	1984	Graduation	Together	26646.0	1	0	10-02-2014	26	11	...
5324	1981	PhD	Married	58293.0	1	0	19-01-2014	94	173	...

칼럼:



2.3 Data Cleaning (데이터 클렌징)

```
#Information on features
data.info()
```

```
Output exceeds the size limit. Open the full output data in a text editor
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 2240 entries, 0 to 2239
Data columns (total 29 columns):
 #   Column               Non-Null Count  Dtype
---  ------               --------------  -----
 0   ID                   2240 non-null   int64
 1   Year_Birth           2240 non-null   int64
 2   Education            2240 non-null   object
 3   Marital_Status       2240 non-null   object
 4   Income               2216 non-null   float64
 5   Kidhome              2240 non-null   int64
 6   Teenhome             2240 non-null   int64
 7   Dt_Customer          2240 non-null   object
 8   Recency              2240 non-null   int64
 9   MntWines             2240 non-null   int64
 10  MntFruits            2240 non-null   int64
 11  MntMeatProducts      2240 non-null   int64
 12  MntFishProducts      2240 non-null   int64
 13  MntSweetProducts     2240 non-null   int64
 14  MntGoldProds         2240 non-null   int64
 15  NumDealsPurchases    2240 non-null   int64
 16  NumWebPurchases      2240 non-null   int64
 17  NumCatalogPurchases  2240 non-null   int64
 18  NumStorePurchases    2240 non-null   int64
 19  NumWebVisitsMonth    2240 non-null   int64
...
 27  Z_Revenue            2240 non-null   int64
 28  Response             2240 non-null   int64
dtypes: float64(1), int64(25), object(3)
memory usage: 507.6+ KB
```

```
#To remove the NA values
data = data.dropna()
```

```
# Created a featrue 'Dt_Customer'
#the number of days a customer is registered in the firm's database

data["Dt_Customer"] = pd.to_datetime(data["Dt_Customer"])
dates = []
for i in data["Dt_Customer"]:
    i = i.date()
    dates.append(i)
#Dates of the newest and oldest recorded customer
print("The newest customer's enrolment date in therecords:",max(dates))
print("The oldest customer's enrolment date in the records:",min(dates))
```

```
The newest customer's enrolment date in therecords: 2014-12-06
The oldest customer's enrolment date in the records: 2012-01-08
```

```
#Created a feature "Customer_For"
days = []
d1 = max(dates) #taking it to be the newest customer
for i in dates:
    delta = d1 - i
    days.append(delta)
data["Customer_For"] = days
data["Customer_For"] = pd.to_numeric(data["Customer_For"], errors="coerce")
```

EWHA EURON

2.3 Data Cleaning (데이터 클렌징)

```
#Feature Engineering
#Age of customer today
data["Age"] = 2021-data["Year_Birth"]

#Total spendings on various items
data["Spent"] = data["MntWines"]+ data["MntFruits"]+ data["MntMeatProducts"]+ data["MntFishProducts"]+ data["MntSweetProducts"]+ data["MntGoldProds"]

#Deriving living situation by marital status"Alone"
data["Living_With"]=data["Marital_Status"].replace({"Married":"Partner", "Together":"Partner", "Absurd":"Alone", "Widow":"Alone", "YOLO":"Alone", "Divorced":"Alone", "Single":"Alone",})

#Feature indicating total children living in the household
data["Children"]=data["Kidhome"]+data["Teenhome"]

#Feature for total members in the householde
data["Family_Size"] = data["Living_With"].replace({"Alone": 1, "Partner":2})+ data["Children"]

#Feature pertaining parenthood
data["Is_Parent"] = np.where(data.Children> 0, 1, 0)

#Segmenting education levels in three groups
data["Education"]=data["Education"].replace({"Basic":"Undergraduate","2n Cycle":"Undergraduate", "Graduation":"Graduate", "Master":"Postgraduate", "PhD":"Postgraduate"})

#For clarity
data=data.rename(columns={"MntWines": "Wines","MntFruits":"Fruits","MntMeatProducts":"Meat","MntFishProducts":"Fish","MntSweetProducts":"Sweets","MntGoldProds":"Gold"})

#Dropping some of the redundant features
to_drop = ["Marital_Status", "Dt_Customer", "Z_CostContact", "Z_Revenue", "Year_Birth", "ID"]
data = data.drop(to_drop, axis=1)
```

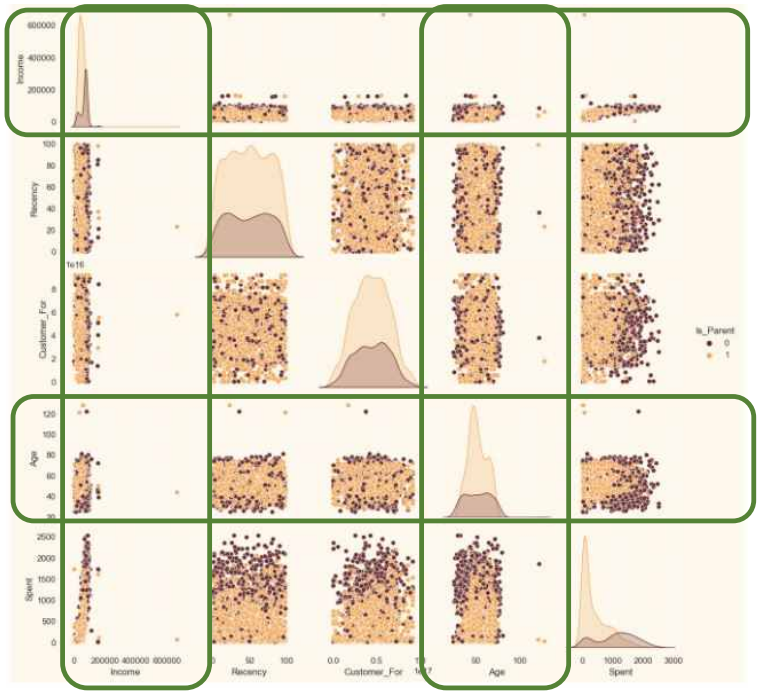
```
data.describe()
```

	Income	Kidhome	Teenhome	Recency	Wines	Fruits	Meat	Fish	Sweets	Gold	...	AcceptedCmp1	AcceptedCmp2	Complain	Response	Customer_For	Age	Spent	Children	Family_Size	Is_Parent
count	2216.000000	2216.000000	2216.000000	2216.000000	2216.000000	2216.000000	2216.000000	2216.000000	2216.000000	2216.000000	...	2216.000000	2216.000000	2216.000000	2216.000000	2.216000e+03	2216.000000	2216.000000	2216.000000	2216.000000	2216.000000
mean	52247.251354	0.441787	0.505415	49.012635	305.091606	26.356047	166.995939	37.637635	27.028881	43.965253	...	0.064079	0.013538	0.009477	0.150271	4.423735e+16	52.179603	607.075361	0.947202	2.592509	0.714350
std	25173.076661	0.536896	0.544181	28.948352	337.327920	39.793917	224.283273	54.752082	41.072046	51.815414	...	0.244950	0.115588	0.096907	0.357417	2.008532e+16	11.985554	602.900476	0.749062	0.905722	0.451825
min	1730.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	...	0.000000	0.000000	0.000000	0.000000	0.000000e+00	25.000000	5.000000	0.000000	1.000000	0.000000
25%	35303.000000	0.000000	0.000000	24.000000	24.000000	2.000000	16.000000	3.000000	1.000000	9.000000	...	0.000000	0.000000	0.000000	0.000000	2.937600e+16	44.000000	69.000000	0.000000	2.000000	0.000000
50%	51381.500000	0.000000	0.000000	49.000000	174.500000	8.000000	68.000000	12.000000	8.000000	24.500000	...	0.000000	0.000000	0.000000	0.000000	4.432320e+16	51.000000	396.500000	1.000000	3.000000	1.000000
75%	68522.000000	1.000000	1.000000	74.000000	505.000000	33.000000	232.250000	50.000000	33.000000	56.000000	...	0.000000	0.000000	0.000000	0.000000	5.927040e+16	62.000000	1048.000000	1.000000	3.000000	1.000000
max	666666.000000	2.000000	2.000000	99.000000	1493.000000	199.000000	1725.000000	259.000000	262.000000	321.000000	...	1.000000	1.000000	1.000000	1.000000	9.184320e+16	128.000000	2525.000000	3.000000	5.000000	1.000000

8 rows × 28 columns

EWHA EURON

2.3 Data Cleaning (데이터 클렌징)




```
#To plot some selected features
#Setting up colors prefrences
sns.set(rc={"axes.facecolor":"#FFF9ED","figure.facecolor":"#FFF9ED"})
pallet = ["#682F2F", "#9E726F", "#D6B2B1", "#B9C0C9", "#9F8A78", "#F3AB60"]
cmap = colors.ListedColormap(["#682F2F", "#9E726F", "#D6B2B1", "#B9C0C9", "#9F8A78", "#F3AB60"])
#Plotting following features
To_Plot = [ "Income", "Recency", "Customer_For", "Age", "Spent", "Is_Parent"]
print("Reletive Plot Of Some Selected Features: A Data Subset")
plt.figure()
sns.pairplot(data[To_Plot], hue= "Is_Parent",palette= (["#682F2F","#F3AB60"]))
#Taking hue
plt.show()
```

✓ Income과 Age 피처에 이상치가 있음을 확인할 수 있다.
-> 이 이상치들을 다 지워준다.

```
#Dropping the outliers by setting a cap on Age and income.
data = data[(data["Age"]<90)]
data = data[(data["Income"]<600000)]
print("The total number of data-points after removing the outliers are:", len(data))
```

```
The total number of data-points after removing the outliers are: 2212
```

2.4 Data Preprocessing (데이터 가공)

```
#Get list of categorical variables
s = (data.dtypes == 'object')
object_cols = list(s[s].index)

print("Categorical variables in the dataset:", object_cols)
```

```
Categorical variables in the dataset: ['Education', 'Living_With']
```

```
#Label Encoding the object dtypes.
LE=LabelEncoder()
for i in object_cols:
    data[i]=data[[i]].apply(LE.fit_transform)

print("All features are now numerical")
```

```
All features are now numerical
```

```
#Creating a copy of data
ds = data.copy()
# creating a subset of dataframe by dropping the features on deals accepted and promotions
cols_del = ['AcceptedCmp3', 'AcceptedCmp4', 'AcceptedCmp5',
            'AcceptedCmp1','AcceptedCmp2', 'Complain', 'Response']
ds = ds.drop(cols_del, axis=1)
#Scaling
scaler = StandardScaler()
scaler.fit(ds)
scaled_ds = pd.DataFrame(scaler.transform(ds),columns= ds.columns )
print("All features are now scaled")
```

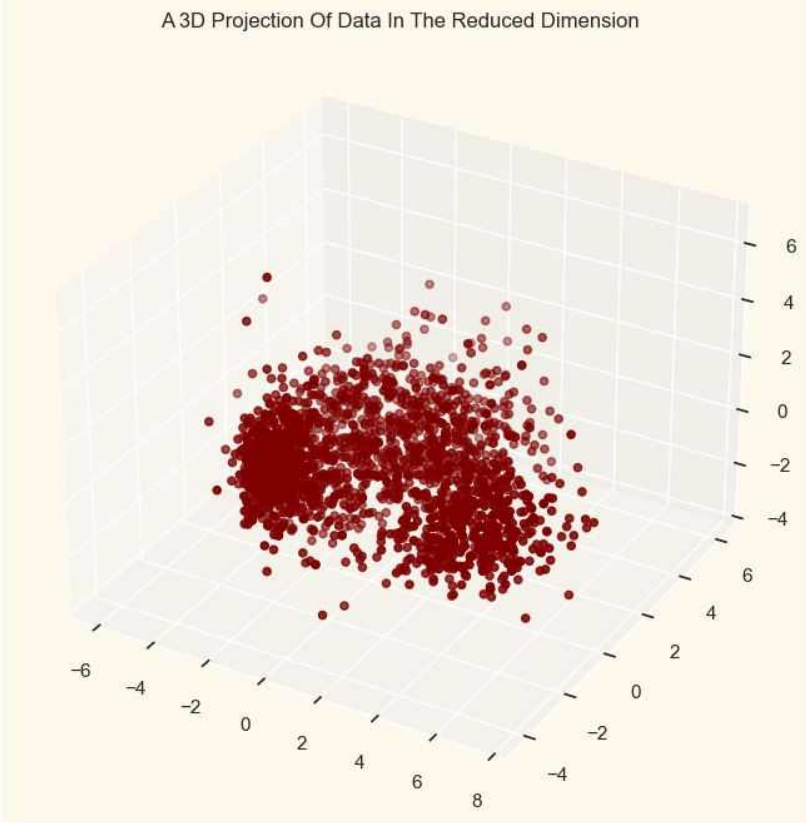
2.5 Dimensionality Reduction (차원 축소)

✓ 피처가 너무 많이 존재하므로 차원 축소를 진행한다

```
#Initiating PCA to reduce dimentions aka features to 3
pca = PCA(n_components=3)
pca.fit(scaled_ds)
PCA_ds = pd.DataFrame(pca.transform(scaled_ds), columns=(["col1","col2", "col3"]
PCA_ds.describe().T
```

	count	mean	std	min	25%	50%	75%	max
col1	2212.0	-1.927331e-17	2.878377	-5.969394	-2.538494	-0.780421	2.383290	7.444305
col2	2212.0	-4.497106e-17	1.706839	-4.312196	-1.328316	-0.158123	1.242289	6.142721
col3	2212.0	3.694051e-17	1.221956	-3.530416	-0.829067	-0.022692	0.799895	6.611222

```
#A 3D Projection Of Data In The Reduced Dimension
x =PCA_ds["col1"]
y =PCA_ds["col2"]
z =PCA_ds["col3"]
#To plot
fig = plt.figure(figsize=(10,8))
ax = fig.add_subplot(111, projection="3d")
ax.scatter(x,y,z, c="maroon", marker="o" )
ax.set_title("A 3D Projection Of Data In The Reduced Dimension")
plt.show()
```

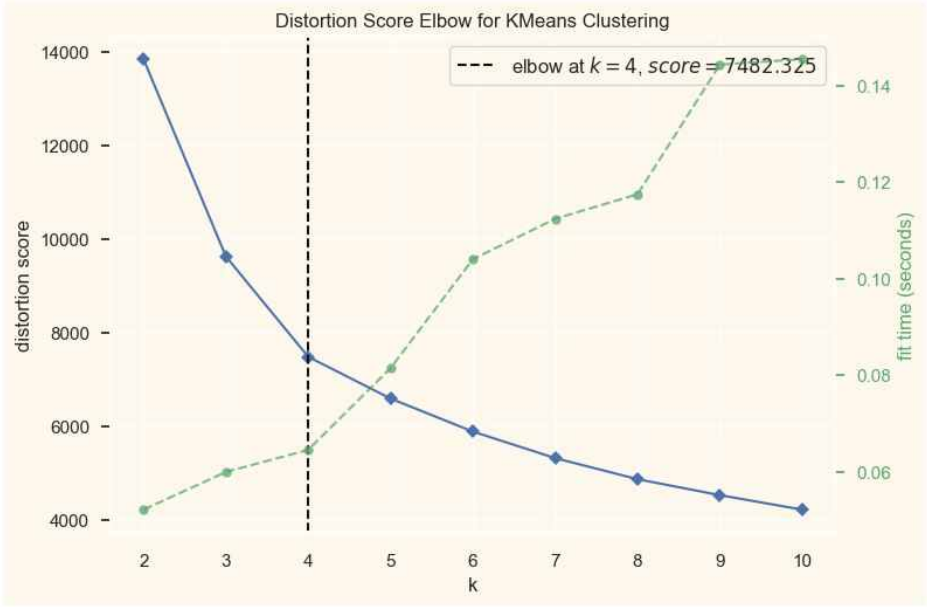


2.6 Clustering (군집화)

Elbow Method: (the number of clusters) = 4

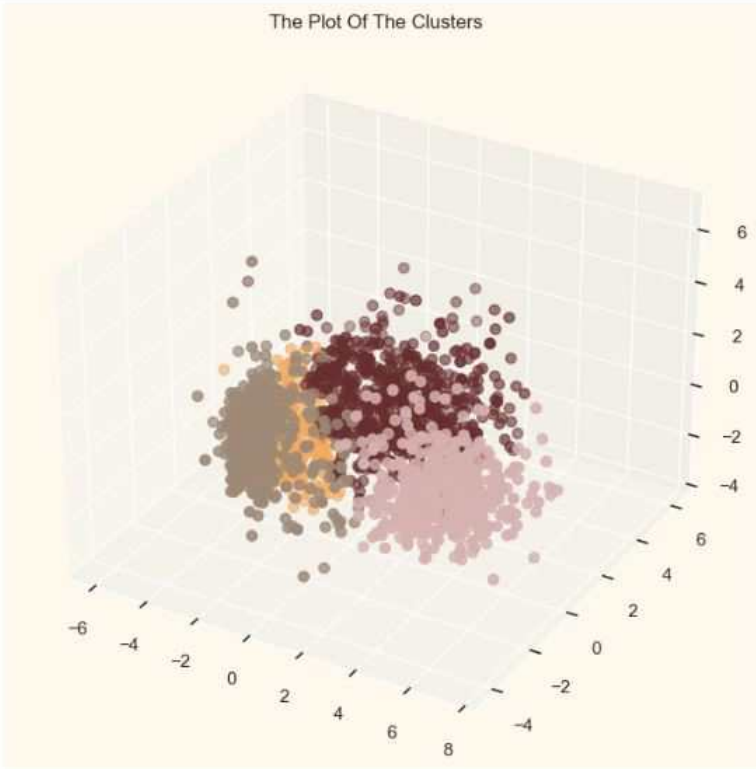
```
# Quick examination of elbow method to find numbers of clusters to make.
print('Elbow Method to determine the number of clusters to be formed:')
Elbow_M = KElbowVisualizer(KMeans(), k=10)
Elbow_M.fit(PCA_ds)
Elbow_M.show()
```

```
Elbow Method to determine the number of clusters to be formed:
```



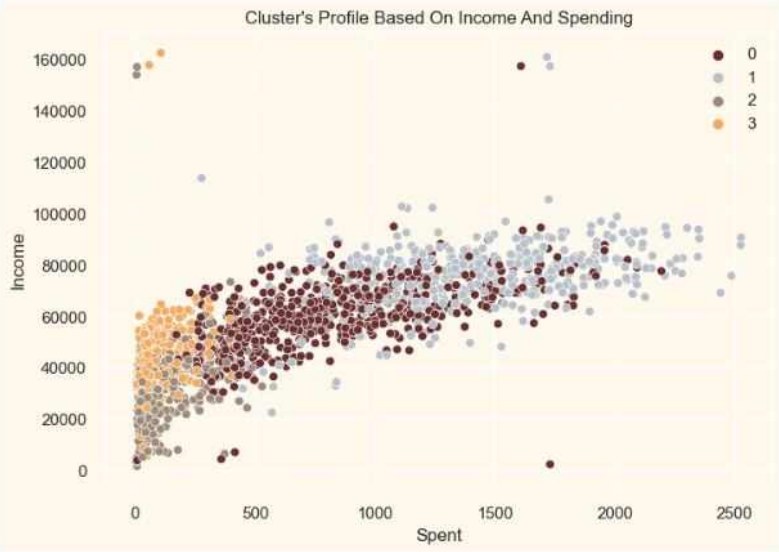

```
#Initiating the Agglomerative Clustering model
AC = AgglomerativeClustering(n_clusters=4)
# fit model and predict clusters
yhat_AC = AC.fit_predict(PCA_ds)
PCA_ds["Clusters"] = yhat_AC
#Adding the Clusters feature to the orignal dataframe.
data["Clusters"]= yhat_AC
```

```
#Plotting the clusters
fig = plt.figure(figsize=(10,8))
ax = plt.subplot(111, projection='3d', label="bla")
ax.scatter(x, y, z, s=40, c=PCA_ds["Clusters"], marker='o', cmap = cmap )
ax.set_title("The Plot Of The Clusters")
plt.show()
```



2.7 Evaluating Models

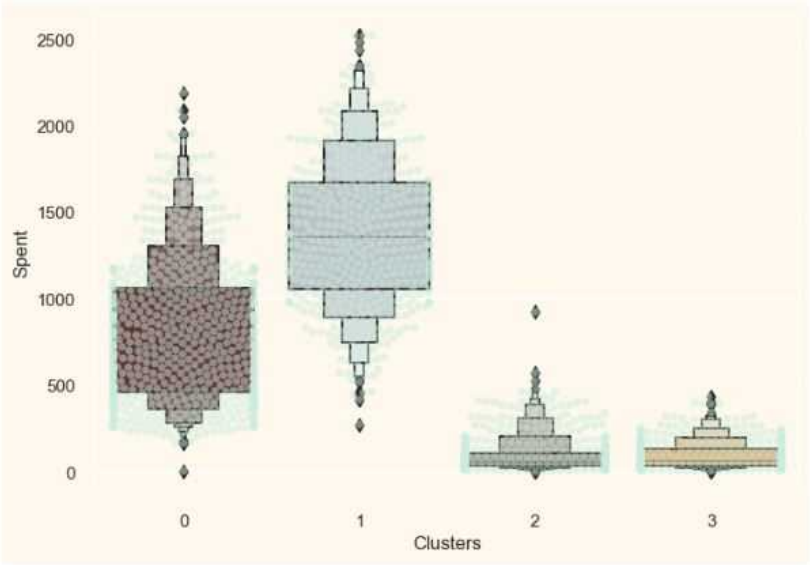
```
pl = sns.scatterplot(data = data,x=data["Spent"], y=data["Income"],hue=data["Clusters"], palette= pal)
pl.set_title("Cluster's Profile Based On Income And Spending")
plt.legend()
plt.show()
```



Spent vs. Income

Cluster 0: 높은 지출 & 평균 수입
Cluster 1: 높은 지출 & 높은 수입
Cluster 2 낮은 지출 & 낮은 수입
Cluster 3: 높은 지출 & 낮은 수입

```
plt.figure()
pl=sns.swarmplot(x=data["Clusters"], y=data["Spent"], color= "#CBEDDD", alpha=0.5 )
pl=sns.boxenplot(x=data["Clusters"], y=data["Spent"], palette=pal)
plt.show()
```



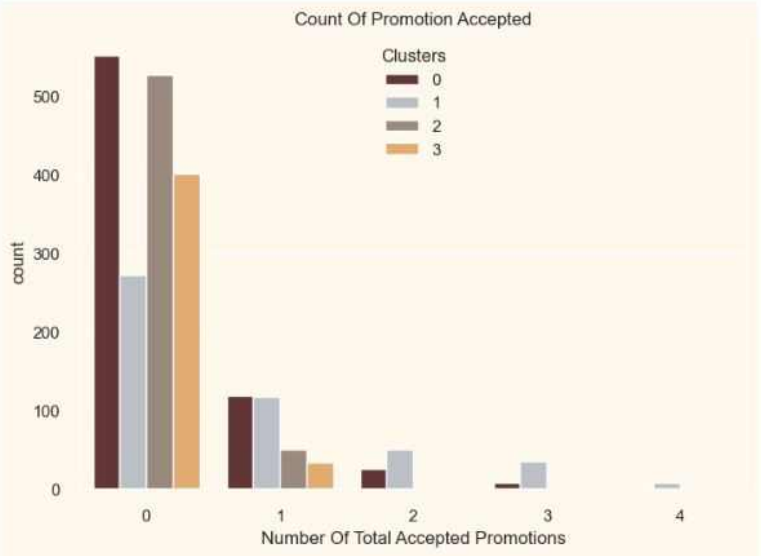
Spent

Cluster1, Cluster 0 순서로 Spent가 높다.

EWHA EURON

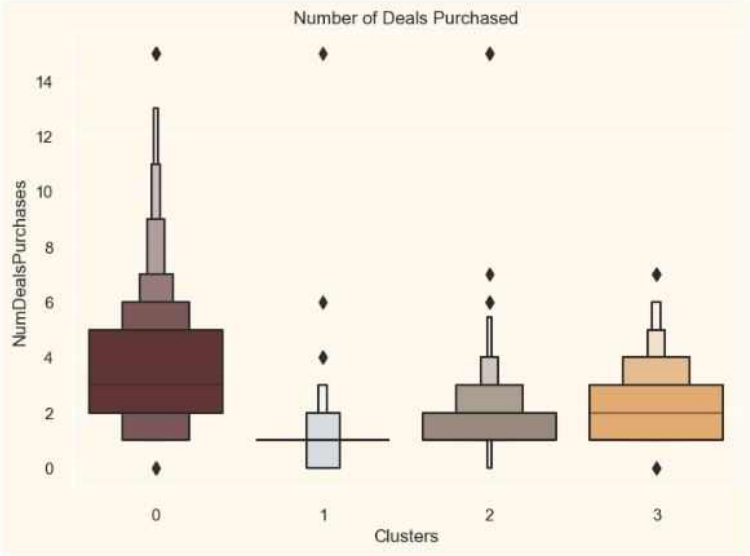
2.7 Evaluating Models

```
#Creating a feature to get a sum of accepted promotions
data["Total_Promos"] = data["AcceptedCmp1"]+ data["AcceptedCmp2"]
                     + data["AcceptedCmp3"]+ data["AcceptedCmp4"]+ data["AcceptedCmp5"]
#Plotting count of total campaign accepted.
plt.figure()
pl = sns.countplot(x=data["Total_Promos"],hue=data["Clusters"], palette= pal)
pl.set_title("Count Of Promotion Accepted")
pl.set_xlabel("Number Of Total Accepted Promotions")
plt.show()
```



Accepting Promotions

```
#Plotting the number of deals purchased
plt.figure()
pl=sns.boxenplot(y=data["NumDealsPurchases"],x=data["Clusters"], palette= pal)
pl.set_title("Number of Deals Purchased")
plt.show()
```



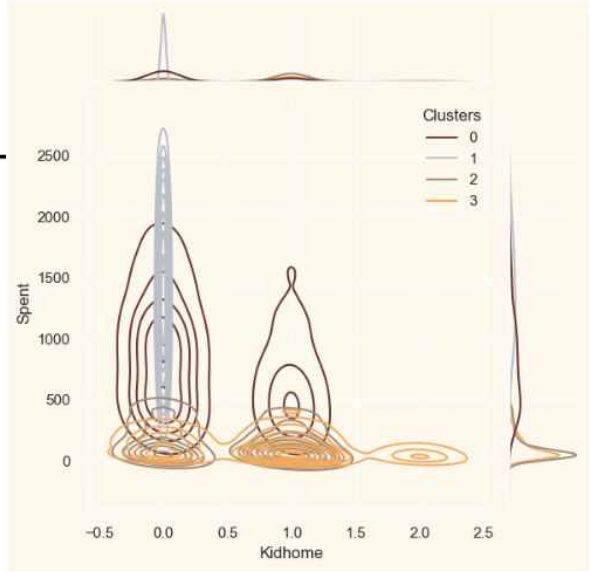
NumDealsPurchases

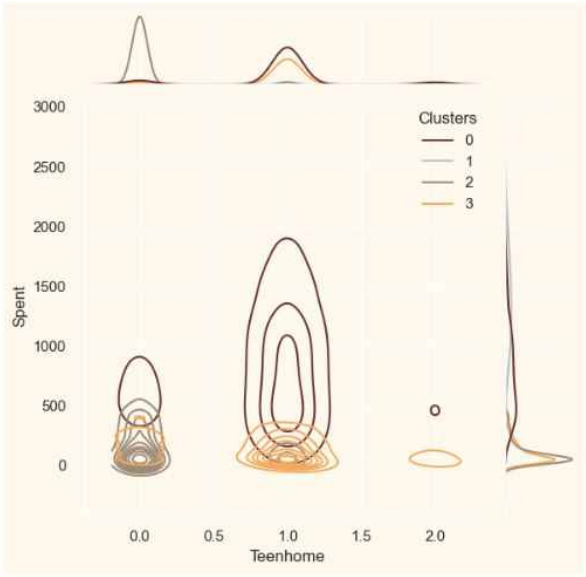
2.8 Profiling

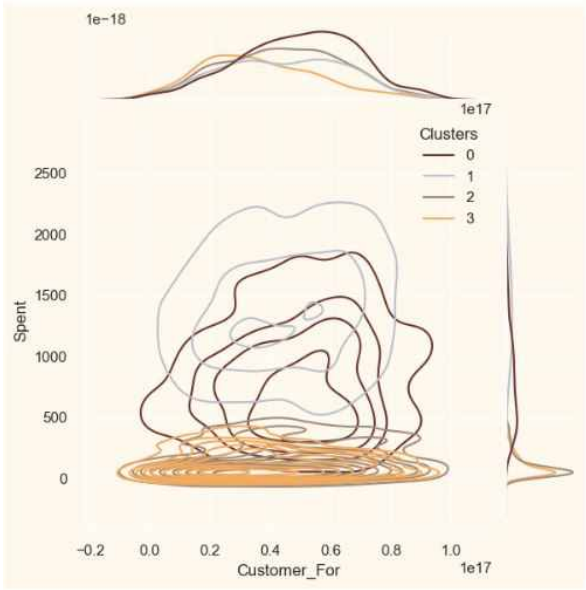
Profiling을 통해 군집에 속한 고객들의 특징을 파악할 수 있다

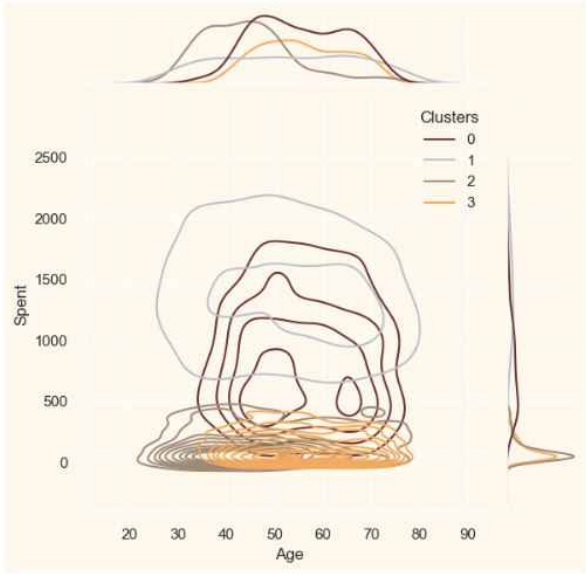
```
Personal = [ "Kidhome","Teenhome","Customer_For", "Age", "Children", "Family_Size", "Is_Parent", "Education","Living_With"]

for i in Personal:
    plt.figure()
    sns.jointplot(x=data[i], y=data["Spent"], hue =data["Clusters"], kind="kde", palette=pal)
    plt.show()
```

Python

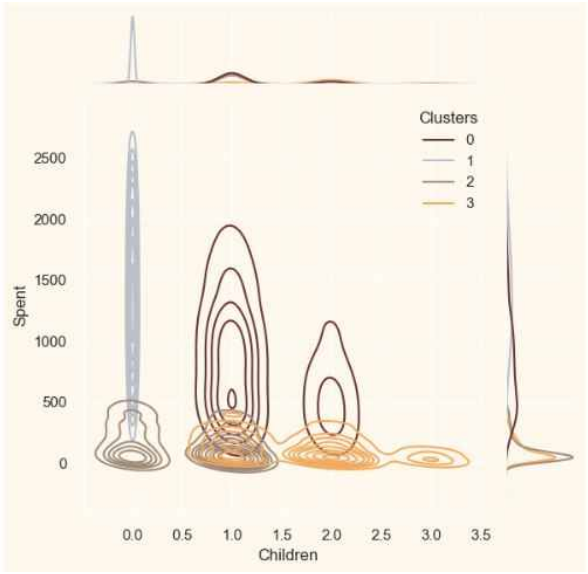


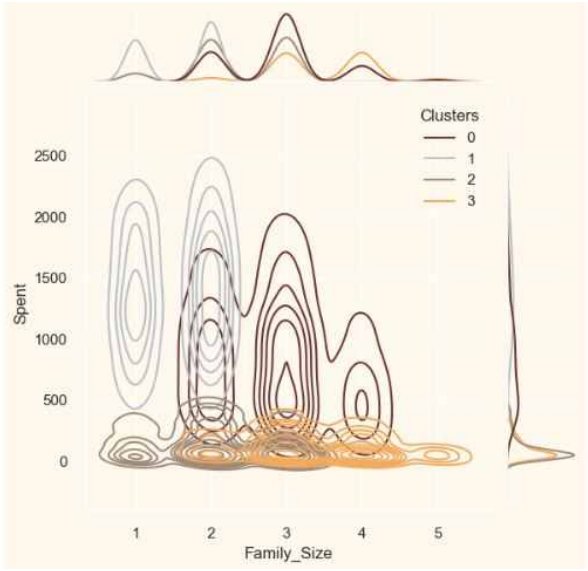


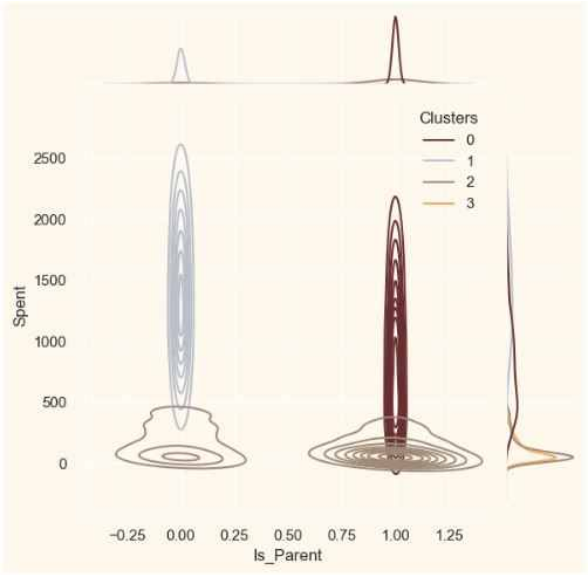


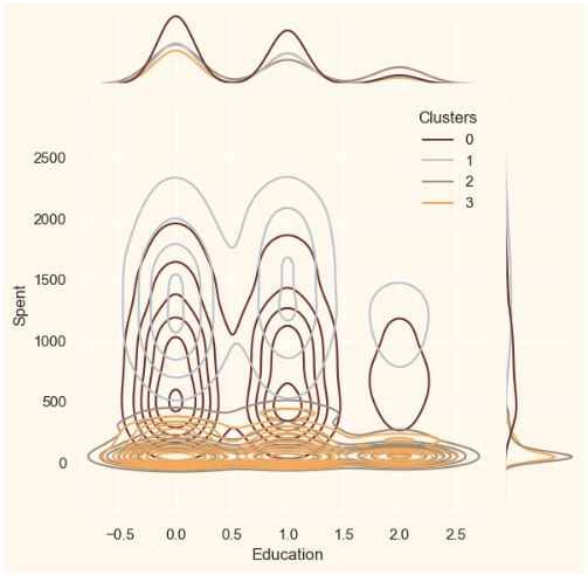


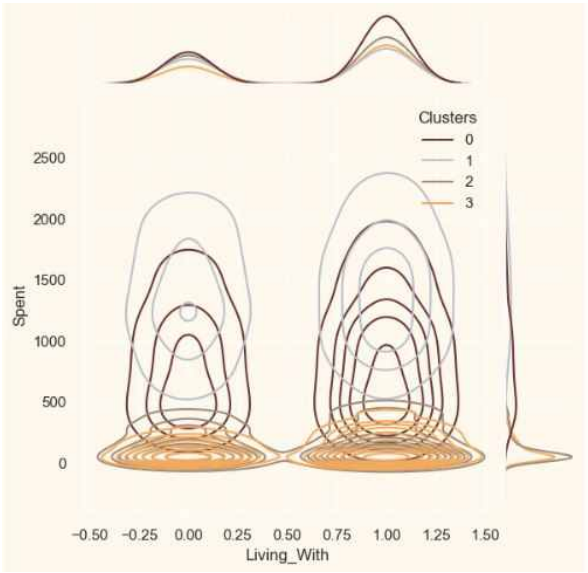
2.8 Profiling











EWHA EURON

2.9 Conclusion

	Cluster 0	Cluster 1	Cluster 2	Cluster 3
자식이 있는가?	있다	없다	대부분 있다	있다
가족 구성원의 수	2~4	~2	~3	~5
자녀 나이	10대		10대보다 어리다	대부분 10대
평균 나이	older		young	
Income		High		Low

3.Mall Customer Segmentation Data

EWHA EURON

3.1 고객 세그먼테이션이란?

- 고객 세그먼테이션 (Customer Segmentataion) : 다양한 기준으로 고객을 분류하는 기법
- CRM 이나 마케팅의 중요한기반 요소

고객 세그먼테이션의 중 목표는 바로 **“ 타깃 마케팅 ”**

- 타깃 마케팅이란 고객을 여러 특성에 맞게 세분화해서 그 유형에 따라 맞춤형 마케팅이나 서비스를 제공하는 것

- 고객 세그먼테이션은 고객의 어떤 요소를 기반으로 군집화할 것인가를 결정하는 것이 중요한데 여기서는 **RMF 기법을** 이용한다.
  (Recency, Frequency, Monetary Value)

EWHA EURON

3.1 대회 소개

- 4년 전, 쇼핑몰에서 타겟 고객으로 바꿀 수 있는 고객을 이해하기 위해서 마케팅 팀이 그에 따른 전략을 수립하고자 열린 대회

- 고객 세분화를 진행하여 다양한 전략으로 수요를 충족시키기 위함
  (특정 그룹을 대상으로 한 마케팅 활동, 고객의 요구에 맞춘 기능 출시, 제품 로드맵 개발)

- 비지도학습 중 하나인 군집화 대회이기 때문에 명확한 평가 기준이 없음

해당 발표에서는 3가지 알고리즘 기술을 사용해 데이터를 분할

- K Means Clustering
- Heirarchal Clustering 중 상향식 병합 군집 방식(Agglomerative Clustering)
- DBSCAN (Density Based Spatial Clustering of Applications with Noise)

3.2 Data Description

- 특정 몰에서 구매한 고객의 데이터 : https://www.kaggle.com/code/jaykumar1607/customer-segmentation-modelling-visuals
- 200개의 raw
- 4개의 column
    - CustomerID : 고객ID
    - Age : 나이
    - Annual Income(k$) : 연간수입
    - Spending Score(1~100) : 지출 지수, 고객 행동이나 구입 데이터와 같은 파라미터에 기반하여 할당된 점수

	CustomerID	Gender	Age	Annual Income (k$)	Spending Score (1-100)
0	1	Male	19	15	39
1	2	Male	21	15	81
2	3	Female	20	16	6
3	4	Female	23	16	77
4	5	Female	31	17	40

3.3 Data analysis processing

1. EDA

2. Data preprocessing

3. K-means

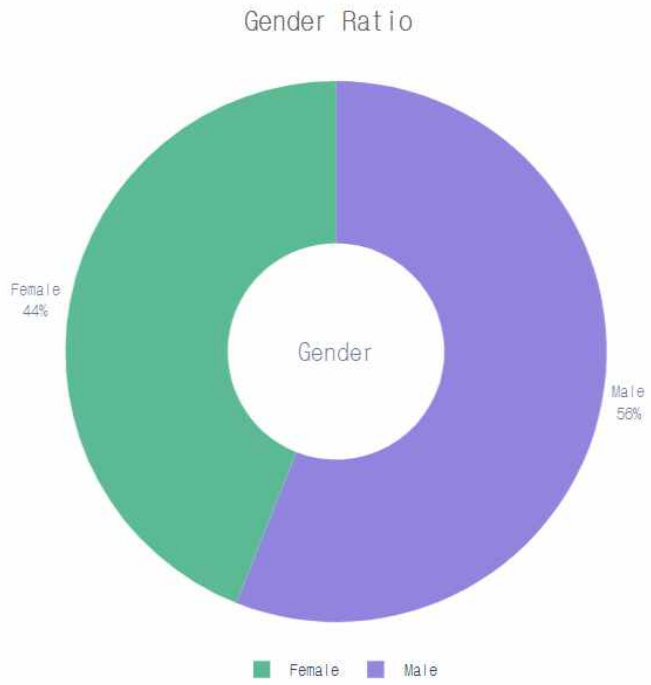
4. 상향식 병합 군집 방식(Agglomerative Clustering)

5. DBSCAN

3.3 Data analysis processing

1. EDA : gender

```
d= pd.DataFrame(df['Gender'].value_counts())
fig = px.pie(d,values='Gender',names=['Male','Female'],hole=0.4,opacity=0.7,
             color_discrete_sequence=[colors_mix[7],colors_mix[2]])

fig.add_annotation(text='Gender',
                   x=0.5,y=0.5,showarrow=False,font_size=18,opacity=0.7,font_family='monospace')

fig.update_layout(
    font_family='monospace',
    title=dict(text='Gender Ratio',x=0.5,y=0.98,
               font=dict(color=colors_dark[2],size=20)),
    legend=dict(x=0.37,y=-0.05,orientation='h',traceorder='reversed'),
    hoverlabel=dict(bgcolor='white'))

fig.update_traces(textposition='outside', textinfo='percent+label')

fig.show()
```

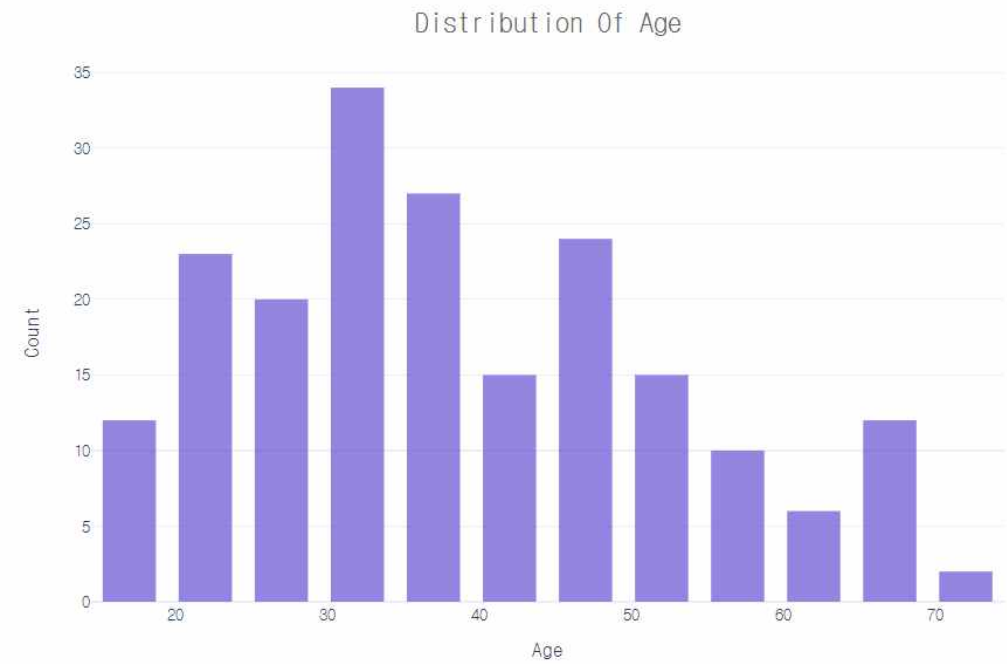


Female이 44%, Male이 56%로 남자가 더 많으나
거의 성비가 동일함을 볼 수 있음

3.3 Data analysis processing

1. EDA : Age

```
fig = px.histogram(df,x='Age',template='plotly_white',opacity=0.7,nbins=25,
                   color_discrete_sequence=[colors_mix[7]])

fig.update_layout(
    font_family='monospace',
    title=dict(text='Distribution Of Age',x=0.5,y=0.95,
               font=dict(color=colors_dark[2],size=20)),
    xaxis_title_text='Age',
    yaxis_title_text='Count',
    legend=dict(x=1,y=0.96,bordercolor=colors_dark[4],borderwidth=0,tracegroupgap=5),
    bargap=0.3,
)
fig.show()
```

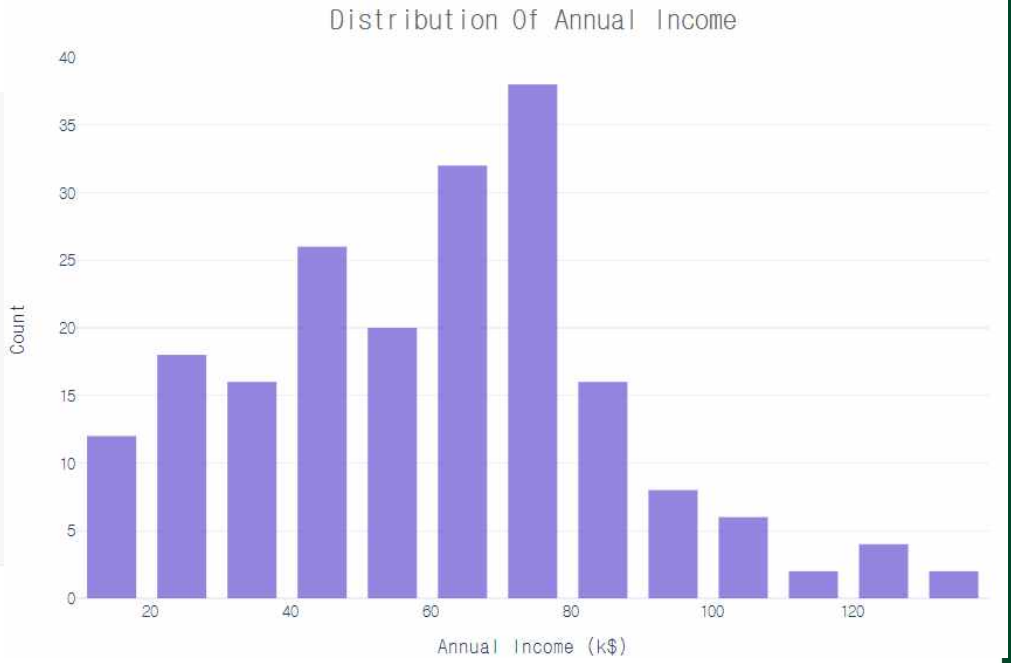


정규분포 모양은 아니지만 비교적 균등하게 분포함. 주로 젊은 층(20,30)의 고객들이 많다

3.3 Data analysis processing

1. EDA : annual income

```
fig = px.histogram(df,x='Annual Income (k$)',template='plotly_white',opacity=0.7,nbins=20,
                   color_discrete_sequence=[colors_mix[7]])

fig.update_layout(
    font_family='monospace',
    title=dict(text='Distribution Of Annual Income',x=0.5,y=0.95,
               font=dict(color=colors_dark[2],size=20)),
    xaxis_title_text='Annual Income (k$)',
    yaxis_title_text='Count',
    legend=dict(x=1,y=0.96,bordercolor=colors_dark[4],borderwidth=0,tracegroupgap=5),
    bargap=0.3,
)
fig.show()
```



정규분포 모양은 아니지만 비교적 균등하게 분포함. 100$ 이상은 거의 없음을 확인할 수 있다

3.3 Data analysis processing

1. EDA : spending score

```
fig = px.histogram(df,x='Spending Score (1-100)',template='plotly_white',opacity=0.7,nbins=20,
                   color_discrete_sequence=[colors_mix[7]])

fig.update_layout(
    font_family='monospace',
    title=dict(text='Distribution Of Spending Score',x=0.5,y=0.95,
               font=dict(color=colors_dark[2],size=20)),
    xaxis_title_text='Spending Score',
    yaxis_title_text='Count',
    legend=dict(x=1,y=0.96,bordercolor=colors_dark[4],borderwidth=0,tracegroupgap=5),
    bargap=0.3,
)
fig.show()
```

대칭적인 모습을 확인할 수 있다. 40~60점수가 가장 많음을 확인할 수 있다

EWHA EURON

3.3 Data analysis processing

2. Data preprocessing

```
df.drop('CustomerID',axis=1,inplace=True)
```

```
df.columns
```

```
Index(['Gender', 'Age', 'Annual Income (k$)', 'Spending Score (1-100)'], dtype='object')
```

Customer ID 칼럼은 분석에 불필요하므로 삭제

```
df['Gender'] = df['Gender'].apply(lambda x: 0 if x=='Male' else 1)
```

```
df['Gender']
```

```
0      0
1      0
2      1
3      1
4      1
      ..
195    1
196    1
197    0
198    0
199    0
Name: Gender, Length: 200, dtype: int64
```

성별 칼럼을 0과 1로 변경
Female ->1
Male -> 0

```
scaler = StandardScaler()
scaler.fit(df)
X = scaler.transform(df)
```

K-means가 유클리시안 거리와 같은 거리계산을 하므로 특이치, 이상치에 민감
-> StandardScale 진행

3.3 Data analysis processing

3. K-means

```
wcss= []    # within cluster sum of squares
ss = []     # silouette score
for i in range(2,11):
    model = KMeans(n_clusters=i)
    model.fit_transform(X)
    wcss.append(model.inertia_)
    ss.append(silhouette_score(X,labels=model.predict(df)))
```

먼저 n cluster를 2 부터 11까지 반복하여 모델을 생성한 후,

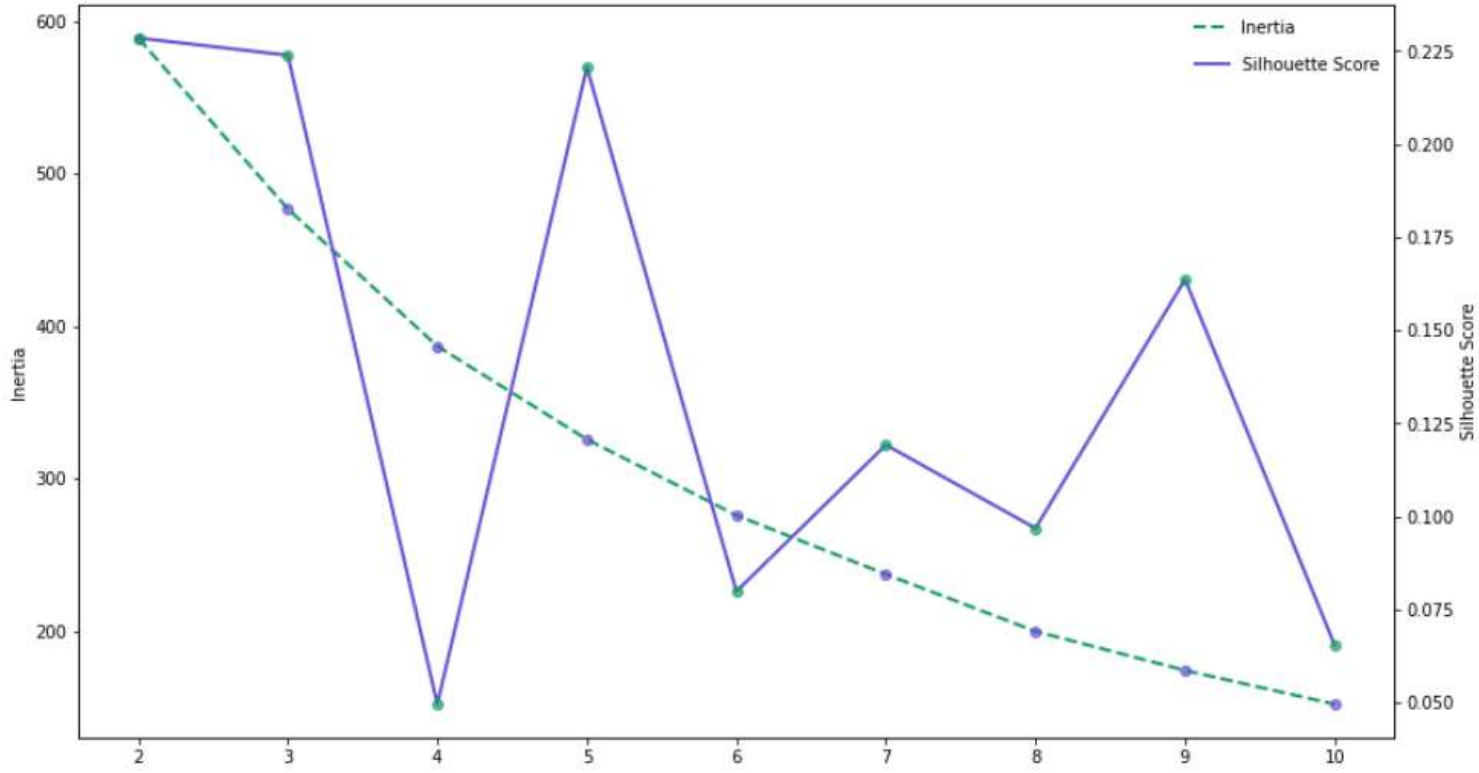
```
fig,ax1 = plt.subplots(figsize=(14,8))

ax1.plot(range(2, 11), wcss , '--', color=colors_mix[2], linewidth=2)
ax1.legend(['Inertia'],bbox_to_anchor=(0.9365,1),frameon=False)
ax1.plot(range(2, 11), wcss , 'o', color=colors_mix[7],alpha=0.7)
ax1.set_ylabel('Inertia')

ax2 = ax1.twinx()
ax2.plot(range(2, 11), ss, '-', color=colors_mix[7], linewidth=2)
ax2.legend(['Silhouette Score'],bbox_to_anchor=(1,0.95),frameon=False)
ax2.plot(range(2, 11), ss, 'o', color=colors_mix[2], alpha=0.7)
ax2.set_ylabel('Silhouette Score')

plt.xlabel('Number of clusters')
plt.show()
```

생성된 모델 각각에 대한
실루엣계수 그래프를 그린다

EWHA EURON

3.3 Data analysis processing

3. K-means

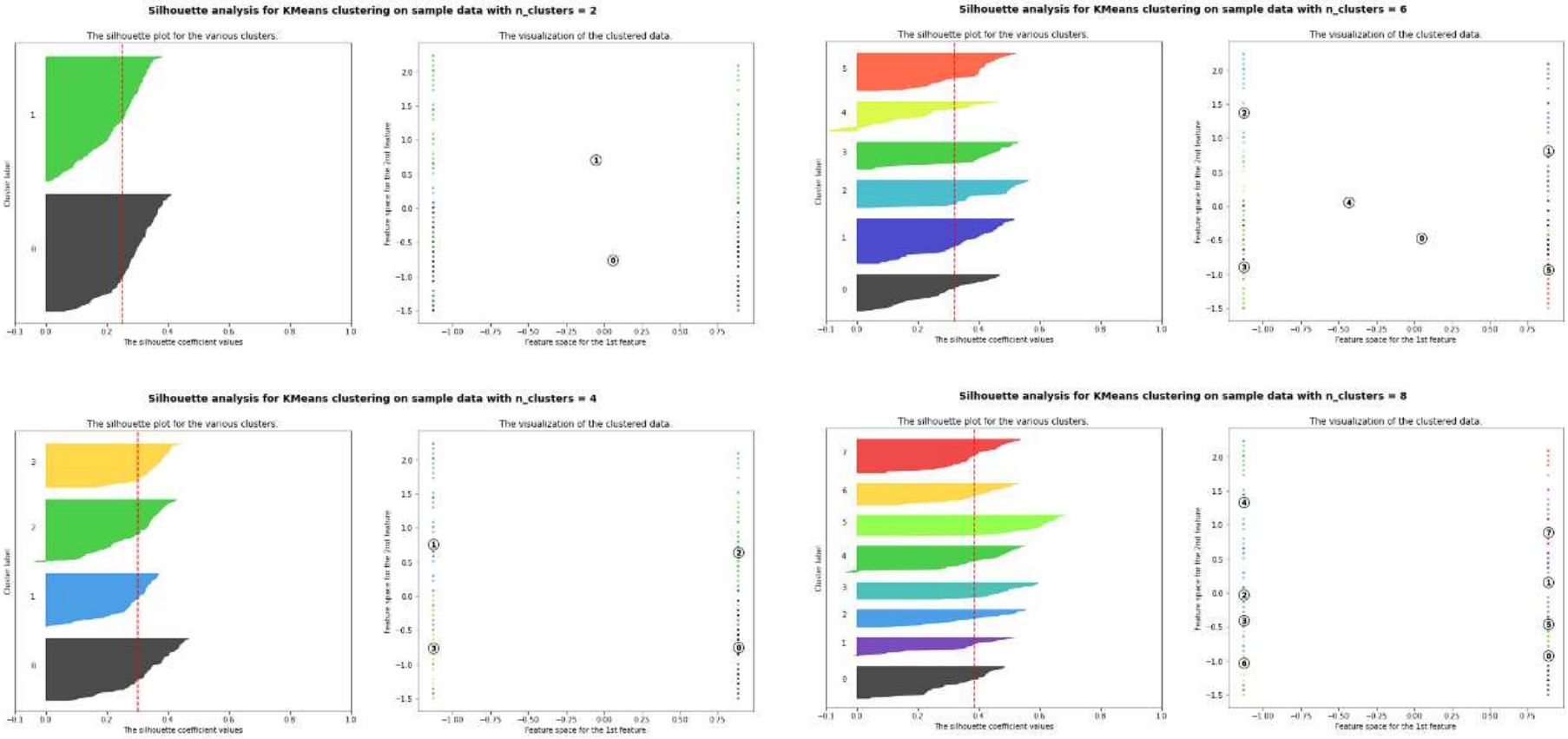


급격한 감소가 보이지 않음 -> n_cluster의 값을 찾을 수 없다

3.3 Data analysis processing

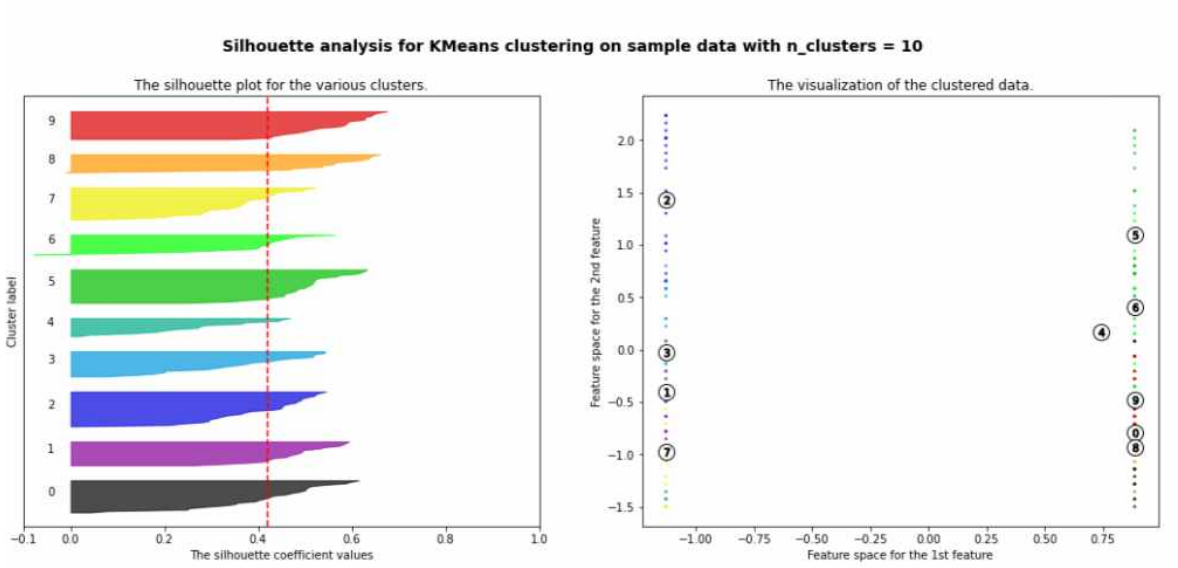
3. K-means

모델에서 만들 군집 수를 검증하기 위해 실루엣 분석을 사용한다.



3.3 Data analysis processing

3. K-means



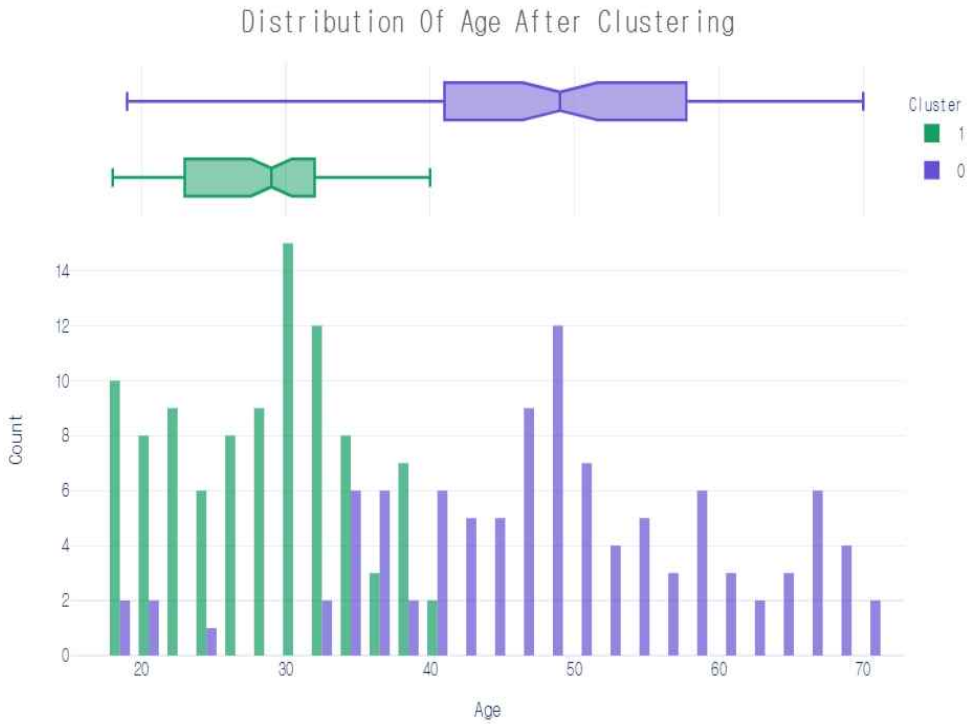

```
For n_clusters = 2 The average silhouette_score is : 0.25181529157884364
For n_clusters = 4 The average silhouette_score is : 0.30123231688013513
For n_clusters = 6 The average silhouette_score is : 0.3199872749106995
For n_clusters = 8 The average silhouette_score is : 0.38738083581583793
For n_clusters = 10 The average silhouette_score is : 0.42011198117622134
```

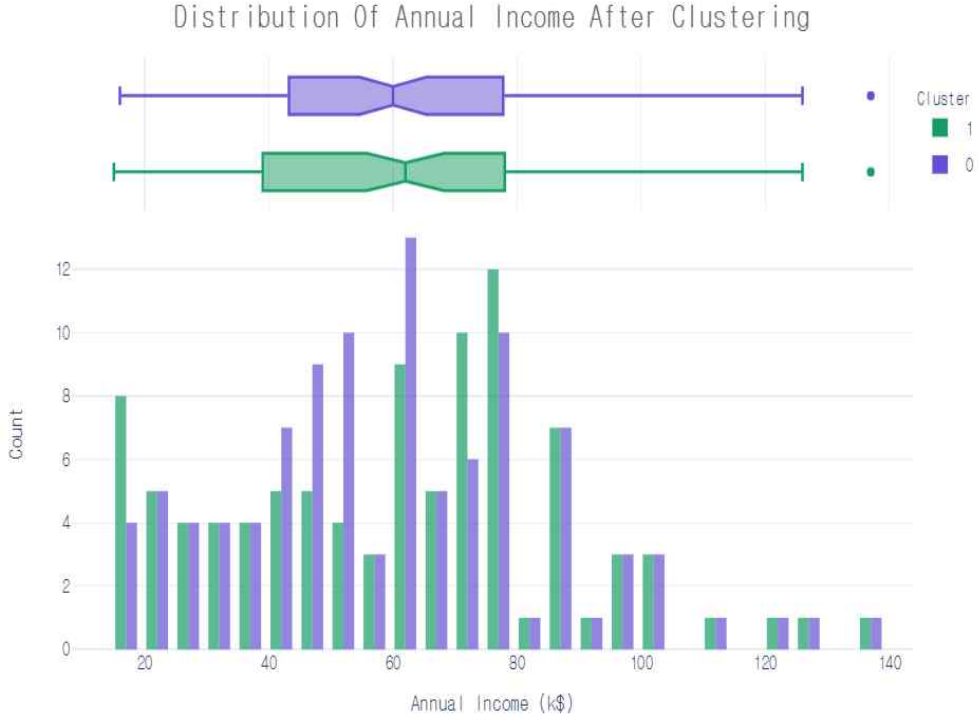
실루엣 분석 그래프를 살펴본 후 kMeans 모델로 n_clusters=2 를 선택

EWHA EURON

3.3 Data analysis processing

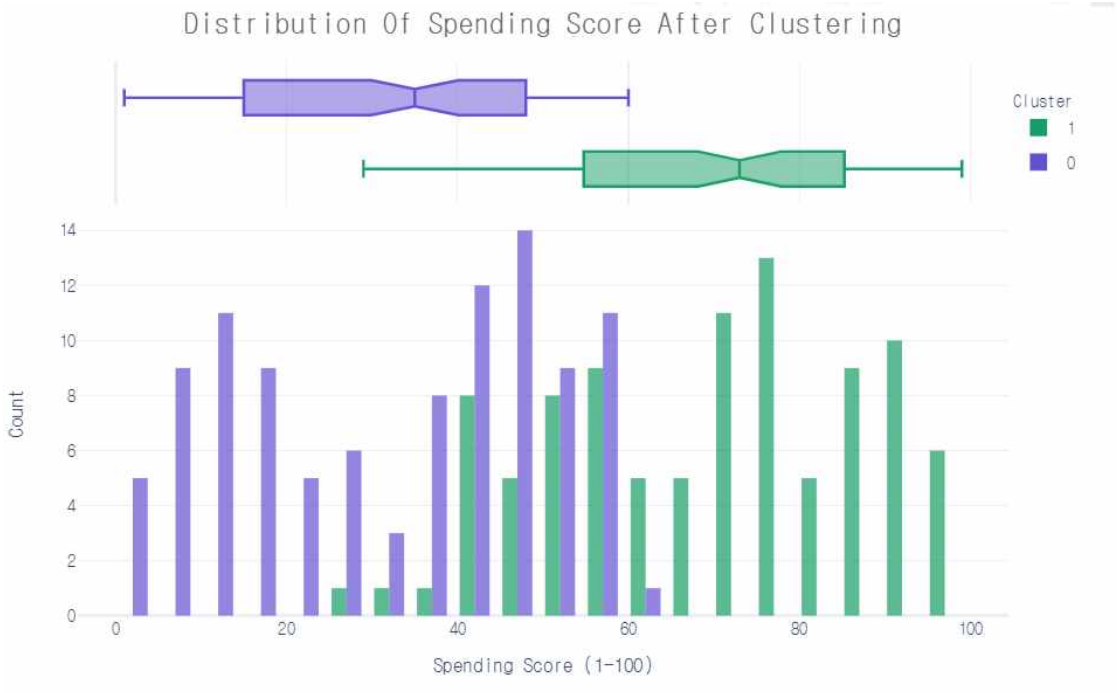
3. K-means : n_clusters =2 선택





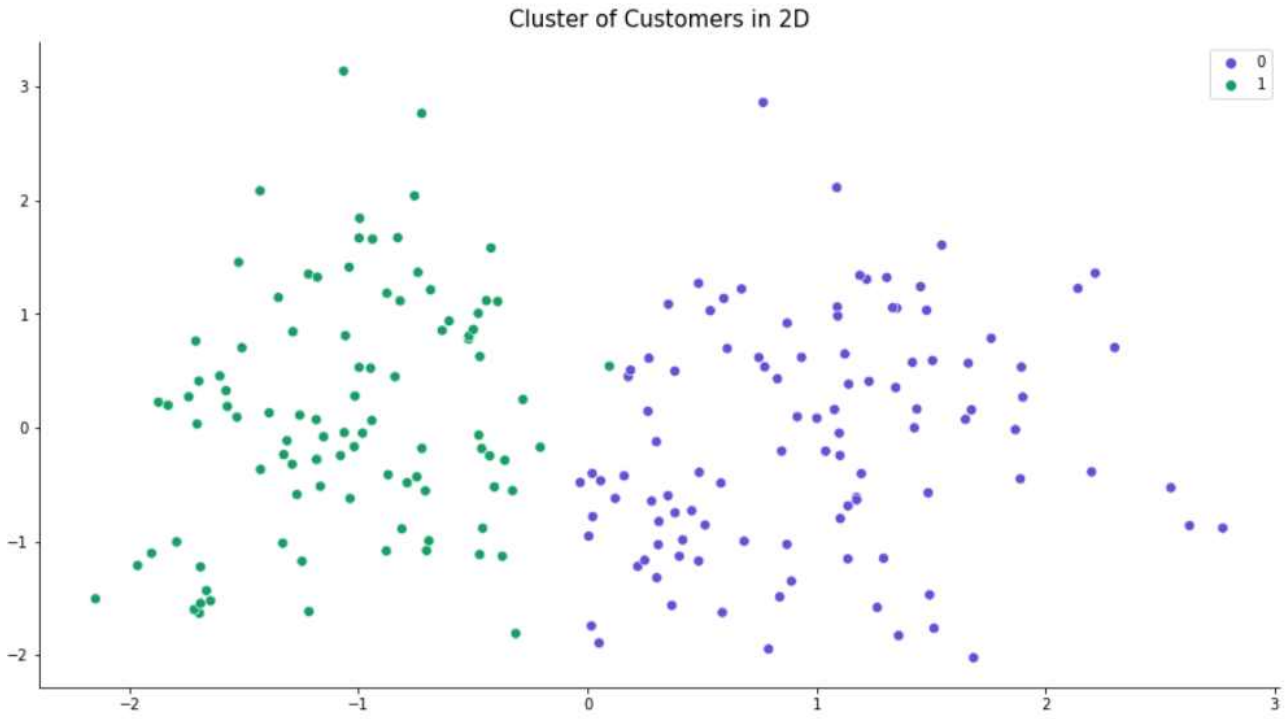
3.3 Data analysis processing

3. K-means : n_clusters =2 선택



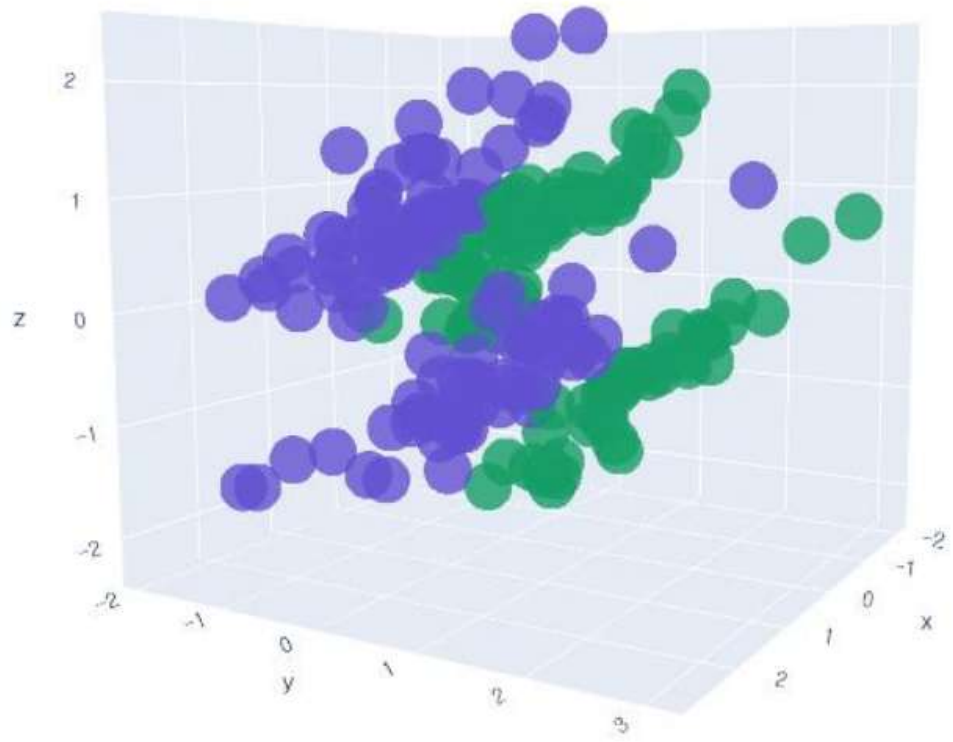
3.3 Data analysis processing

3. K-means : n_clusters =2 선택 -> pca(2) 사용



3.3 Data analysis processing

3. K-means : n_clusters =2 선택 -> pca(3) 사용



3.3 Data analysis processing

3. 상향식 병합 군집 방식(Agglomerative Clustering)
: 점들 간의 유사성에 기초하여 그룹 또는 클러스터를 형성하는 기술

```
model = AgglomerativeClustering(distance_threshold=0,n_clusters=None)
model.fit(X)
```

```
AgglomerativeClustering(distance_threshold=0, n_clusters=None)
```

EWHA EURON

3.3 Data analysis processing

3. 상향식 병합 군집 방식(Agglomerative Clustering)
: 점들 간의 유사성에 기초하여 그룹 또는 클러스터를 형성하는 기술

```
def plot_dendrogram(model, **kwargs):
    # Create linkage matrix and then plot the dendrogram

    # create the counts of samples under each node
    counts = np.zeros(model.children_.shape[0])
    n_samples = len(model.labels_)
    for i, merge in enumerate(model.children_):
        current_count = 0
        for child_idx in merge:
            if child_idx < n_samples:
                current_count += 1  # leaf node
            else:
                current_count += counts[child_idx - n_samples]
        counts[i] = current_count

    linkage_matrix = np.column_stack([model.children_, model.distances_,
                                      counts]).astype(float)

    # Plot the corresponding dendrogram
    dendrogram(linkage_matrix, **kwargs)
```

EWHA EURON

3.3 Data analysis processing

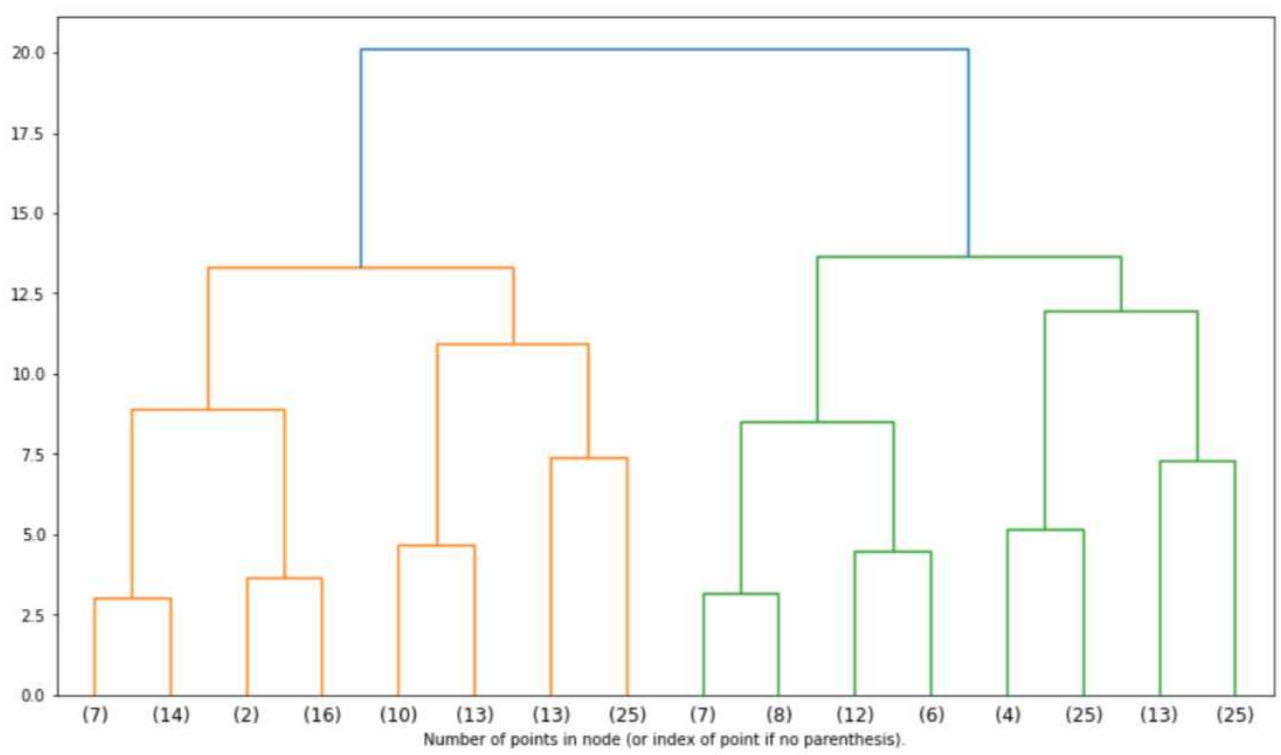
3. 상향식 병합 군집 방식(Agglomerative Clustering)
: 점들 간의 유사성에 기초하여 그룹 또는 클러스터를 형성하는 기술

```
def plot_dendrogram(model, **kwargs):
    # Create linkage matrix and then plot the dendrogram

    # create the counts of samples under each node
    counts = np.zeros(model.children_.shape[0])
    n_samples = len(model.labels_)
    for i, merge in enumerate(model.children_):
        current_count = 0
        for child_idx in merge:
            if child_idx < n_samples:
                current_count += 1  # leaf node
            else:
                current_count += counts[child_idx - n_samples]
        counts[i] = current_count

    linkage_matrix = np.column_stack([model.children_, model.distances_,
                                      counts]).astype(float)

    # Plot the corresponding dendrogram
    dendrogram(linkage_matrix, **kwargs)
```

```
plt.figure(figsize=(14,8))
plot_dendrogram(model,truncate_mode = 'level',p=3)
plt.xlabel("Number of points in node (or index of point if no parenthesis).")
plt.show()
```

3.3 Data analysis processing

3. 상향식 병합 군집 방식(Agglomerative Clustering)
: 점들 간의 유사성에 기초하여 그룹 또는 클러스터를 형성하는 기술



2개의 군집이 만들어진 것을 확인

EWHA EURON

3.3 Data analysis processing

5. DBSCAN
- 2개의 주요 파라미터
1) Eps : radius of each core 2) min_samples : minimum number of points

```
model = DBSCAN(eps=1,min_samples=5)
cluster_labels = model.fit_predict(X)
```

```
silhouette_score(X,cluster_labels)
```

```
0.2543871824295202
```

```
pca = PCA(2)
pca.fit(X)
X_PCA = pca.transform(X)
plt.figure(figsize=(15,8))
sns.scatterplot(x=X_PCA[:, 0], y=X_PCA[:, 1],
                hue=cluster_labels, palette=[colors_mix[3],colors_mix[2],colors_mix[7]], s=50)
plt.title('Cluster of Customers in 2D', size=15, pad=10)
sns.despine()
plt.legend(loc=0, bbox_to_anchor=[1,1])
plt.show()
```

EWHA EURON

3.3 Data analysis processing

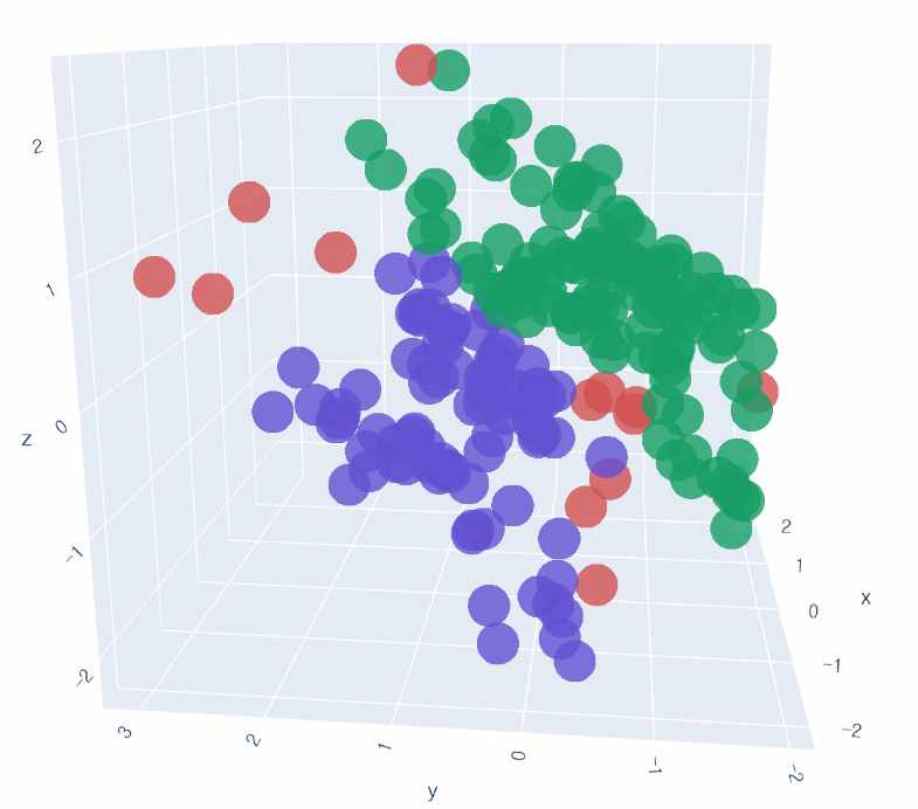
5. DBSCAN

2D 표현으로는 clusters 간의 차이가 명확해 보이지 않음 -> 3D

3.3 Data analysis processing

5. DBSCAN

```
pca = PCA(3)
pca.fit(X)
X_PCA = pca.transform(X)


fig = px.scatter_3d(x=X_PCA[:,0], y=X_PCA[:,1], z=X_PCA[:,2],
                    color=cluster_labels,opacity=0.8,
                    color_continuous_scale=[colors_mix[3],colors_mix[7],colors_mix[2]],
                    width=800,height=800)

fig.update_layout(font_family='monospace',
    title=dict(text='Customer Clusters in 3D',x=0.5,y=0.95,
    font=dict(color=colors_dark[2],size=20)),
    coloraxis_showscale=False)
fig.show()
```



명확한 차이가 보임

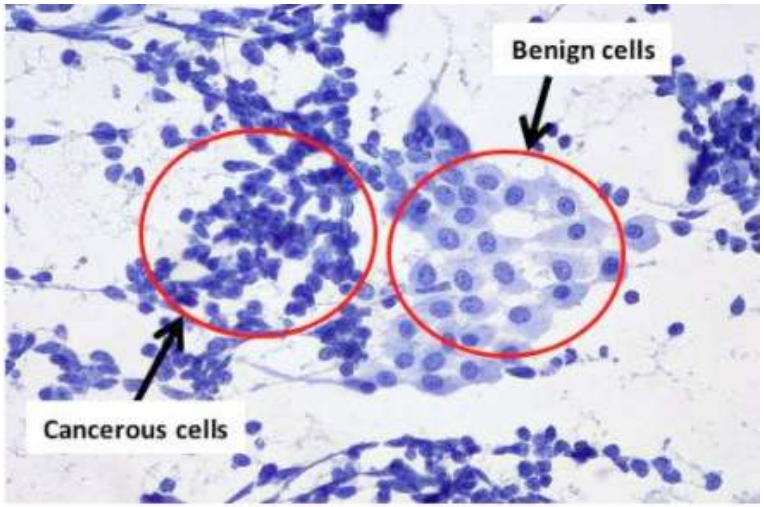
3.4 결론

- 실루엣 분석을 사용하여 n_clusters의 정확한 값을 찾음
- K-Means 의 경우 두 cluster가 연령과 점수에 따라 고객을 명확하게 구분
- 계층형 clustering을 사용하여 2개의 군집을 형성
- DBSCAN을 사용하여 형성된 클러스터는 K-Means와 다르고, 이상치가 발견됨

4. 유방암 예측

#4.1 대회 소개 및 Data Description

- 유방 조직의 이미지로 이루어진 위스콘신 유방암 데이터를 이용하여 양성인지 음성인지 분류해내는 것이 목표



EWHA EURON

#4.1 대회 소개 및 Data Description

- 유방 조직의 이미지로 이루어진 위스콘신 유방암 데이터를 이용하여 양성인지 음성인지 분류해내는 것이 목표

- Attribute information :
    - ID number
    - Diagnosis (M = malignant, B = benign)
    - radius : 반경
    - texture : gray-scale values 표준편차
    - Perimeter : 둘레
    - area : 영역
    - smoothness : 반경 길이의 국부적 변화
    - compactness : perimeter^2/area -1.0
    - concavity : 윤곽의 오목함
    - concave points : 오목한 부분의 개수
    - symmetry : 대칭성
    - fractal dimension : coastline approximation -1

4.2 Loading Libraries and Utilities

Data : https://www.kaggle.com/code/bhuvanchennoju/women-and-cancer-analysis-and-detection

Na 및 missing value 제거

```
print(color_class.BOLD_COLOR + 'Shape of The Dataset:'+ color_class.END + color_class.BOLD + '
{}'.format(df.shape) +2*'\n' + 'Note: There is {} missing values column in the data'.format(df.
isnull().any().sum())  +
     '\n')

print(color_class.BOLD_COLOR + 'Null Value Count Details as Follows'+ color_class.END + '\n')
print(color_class.BOLD)
print(df.isnull().sum())
```

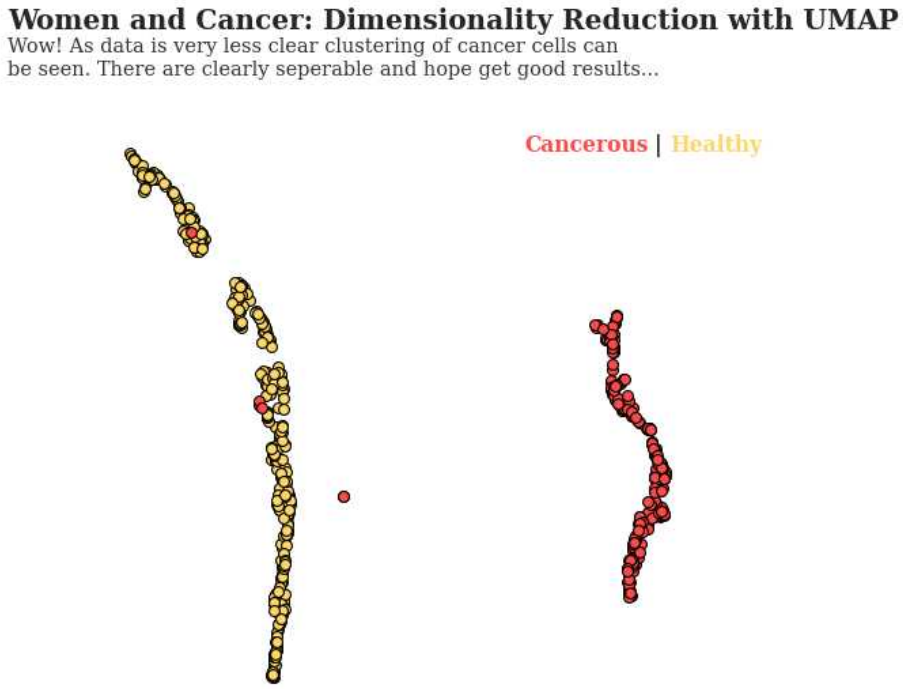
필요없는 열 제거 및 진단 데이터 0과 1로 변경 (M : 1, B : 0)

```
df.drop(columns = ['Unnamed: 32'], inplace = True)
df['diagnosis'] = df['diagnosis'].map({'M':1, 'B':0})

print(color_class.BOLD_COLOR + 'Sneak peak into the data...'+ color_class.END)
print(color_class.BOLD)
print(df.head(2).T)
```

EWHA EURON

4.3 Dimentionality Reduction with UMAP

```
temp = df.copy()

X_temp = temp.drop(columns = ['id','diagnosis'])
y_temp = temp['diagnosis']

# fitting on umap
umap = UMAP(random_state=2021)
model_umap = umap.fit_transform(X_temp, y_temp)

fig,ax = plt.subplots(figsize=(7,7),dpi =80)

# plots
ax.scatter(model_umap[temp['diagnosis'] == 0][:,0], model_umap[temp['diagnosis'] == 0][:,1], c=
colors[2], alpha=1,s=50, linewidth = 1, ec = 'black')
ax.scatter(model_umap[temp['diagnosis'] == 1][:,0], model_umap[temp['diagnosis'] == 1][:,1], c=
colors[0], alpha=1,s=50, linewidth = 1, ec = 'black')


## titles and text

ax.set_xticklabels('')
ax.set_yticklabels('')

fig.text(0,1.01,'Women and Cancer: Dimensionality Reduction with UMAP', {'font':'serif','size':
18, 'weight':'bold'}, alpha = 1)
fig.text(0,0.95,'''Wow! As data is very less clear clustering of cancer cells can
be seen. There are clearly seperable and hope get good results...''',{'font':'serif','size':13,
'weight':'normal'}, alpha = 0.95)

fig.text(0.68,0.85, "Cancerous",{'font':'serif','size':14, 'weight':'bold', 'color':colors[0]})
fig.text(0.85,0.85, '|',{'font':'serif','size':14, 'weight':'bold'})
fig.text(0.87,0.85, "Healthy",{'font':'serif','size':14, 'weight':'bold','color':colors[2]})

fig.text(0.65,0.05,'© Made by bhuvanchennoju/Kaggle',{'font':'serif', 'size':10,'weight':'bol
d'}, alpha = 0.85)


fig.show()
```



UMAP를 이용하면 암세포가 완벽하게 분리된다.

EWHA EURON

4.4 Outliers And Influential Points

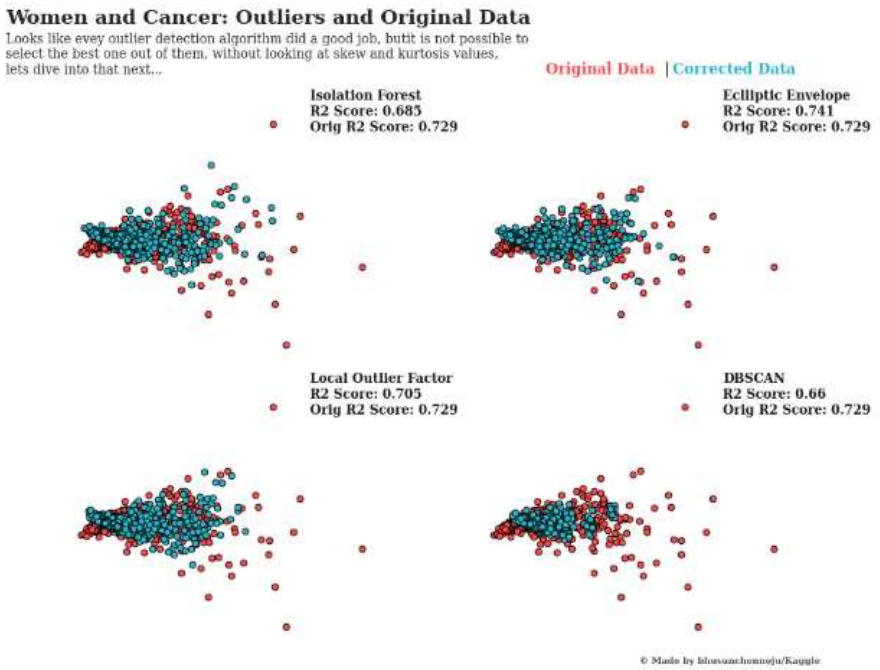
모든 분포가 완전히 정규 분포를 따르는 것은 아님. 따라서 특이치를 제거할 필요가 있음.
특이치 제거 방법
1) Elliptic Envelope 2) DBSCAN 3) Local Outlier Factor 4) Isolation Forest

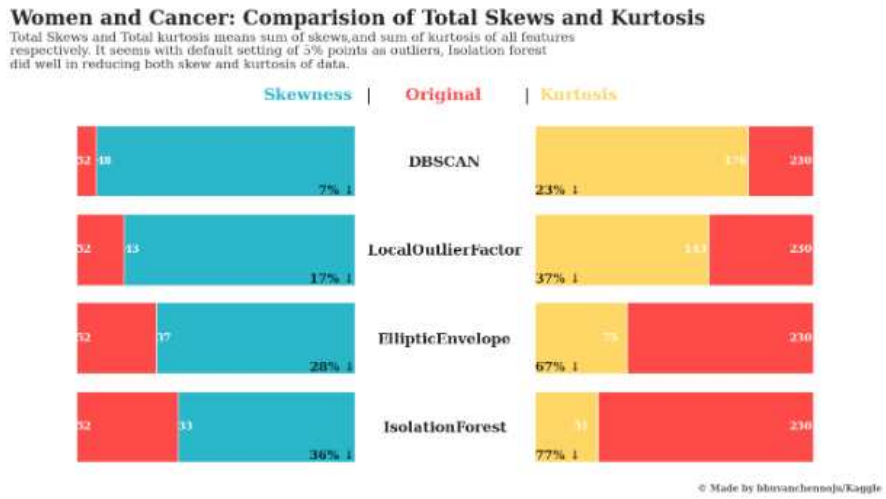
```
def outlier_detect(algo,data):
    cols = data.drop(columns = ['id']).columns
    # creating feature and target numpy arrays
    feat, tar = data[cols].drop(columns = 'diagnosis').values, data['diagnosis'].values
    # fitting the features to algo
    yhat = algo.fit_predict(feat)
    # masking the features that are not outliers
    mask = yhat != -1
    X,y = feat[mask,:], tar[mask]
    data_inarray = np.append(y.reshape(-1,1),X,axis = 1)
    return pd.DataFrame(data = data_inarray, columns = cols)

def skew_sum(data):
    return skew(data).sum()

def kurtosis_sum(data):
    return kurtosis(data).sum()

def shape(data):
    return data.shape
```

```
outlier_algos = [IsolationForest(contamination = 0.05),\
                 EllipticEnvelope(contamination = 0.05),\
                 LocalOutlierFactor(contamination = 0.05), \
                 DBSCAN(eps = 70, min_samples = 10)]
```

```
algorithms = ['Original','IsolationForest', 'EllipticEnvelope', 'LocalOutlierFactor', 'DBSCAN']
outliers_info = pd.DataFrame({'algorithms':algorithms,'df_list':df_list,'shapes':shapes, 'skew
s':skews, 'kurts': kurts})

outliers_info['skews_sum'] = outliers_info['skews'].apply(lambda x: round(x.sum(),2))
outliers_info['kurts_sum'] = outliers_info['kurts'].apply(lambda x: round(x.sum(),2))
outliers_info.sort_values(by = 'shapes').reset_index(drop = True, inplace = True)


for idx, df_ in enumerate(outliers_info['df_list']):
    from sklearn.metrics import f1_score

    lr = LinearRegression()
    X = df_.drop(columns = ['diagnosis'])
    y = df_['diagnosis']
    xtrain, xtest,ytrain,ytest = train_test_split(X,y,test_size = 0.2)

    # linear regression
    preds = LinearRegression().fit(xtrain.values,ytrain.values).predict(xtest.values)

    r2 = round(r2_score(ytest,preds),3)
    #ypred_class = preds > 0.85
    #acc = round(accuracy_score(ytest, ypred_class),3)
    #roc_auc = round(roc_auc_score(ytest,ypred_class),3)

    metric_list = r2

    outliers_info.loc[idx, 'r2_score'] = metric_list
```

4.5 Outliers And Influential Points






```
Shape of Original Data: (569, 31)
Shape of Corrected Data: (540, 31)
```

Isolation Forest를 통해 전체 왜도와 kurtosis가 크게 감소하고 R^2가 약간 향상되었음
-> IsolationForest 선택

EWHA EURON

4.6 Modeling

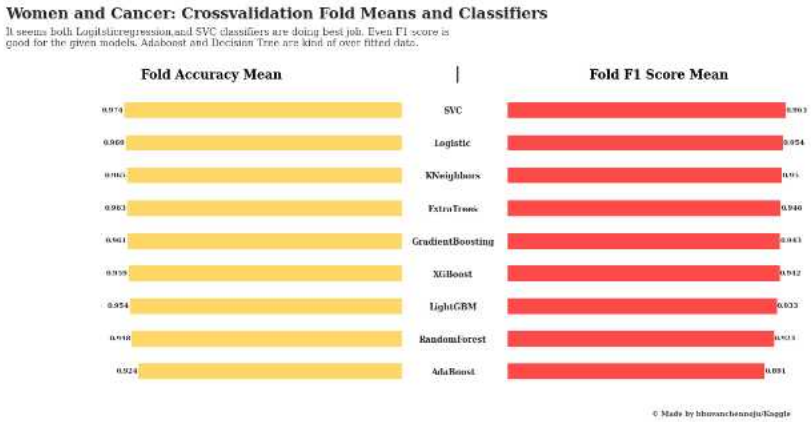
총 9개의 분류 모델에 대해 cross –validation score 측정 후 top 3개 bottom 2개 선택

```
stratified = StratifiedKFold(n_splits = 5, shuffle = True, random_state  = 20)
Algorithms = ["Logistic","SVC","KNeighbors","AdaBoost",
              "RandomForest","GradientBoosting",
              "ExtraTrees","XGBoost", "LightGBM"]

for classifier,algo in zip(classifiers,Algorithms):

    fold_accuracy = []
    fold_f1 = []
    fold_roc_auc = []
    fold_preds = []
    fold_class_states = []
    fold_valid_truths = []
    fold_valid_features = []
    fold_cm = []

    n = 0

    print(color_class.BOLD + '*'*17+  color_class.END + color_class.BOLD_COLOR + str(algo) + col
or_class.END + color_class.BOLD +  '*'*17 + color_class.END)
    for train_idx,valid_idx in stratified.split(xdata,ydata):
        xtrain,xvalid = xdata.iloc[train_idx],xdata.iloc[valid_idx]
        ytrain,yvalid = ydata.iloc[train_idx],ydata.iloc[valid_idx]

        ## scaling
        ss = StandardScaler()
        xtrain = ss.fit_transform(xtrain)
        xvalid = ss.transform(xvalid)
```

```
        # model
        model = classifier
        model.fit(xtrain,ytrain)
        preds = model.predict(xvalid)

        ## scores
        #### fold results, feaures,preds, states
        accuracy = accuracy_score(yvalid, preds)
        f1 = f1_score(yvalid,preds)
        roc_auc = roc_auc_score(yvalid,preds)
        cm = confusion_matrix(yvalid,preds)

        fold_accuracy.append(accuracy)
        fold_f1.append(f1)
        fold_roc_auc.append(roc_auc)
        fold_preds.append(preds)
        fold_class_states.append(model)
        fold_valid_truths.append(np.array(yvalid).astype(int))
        fold_valid_features.append(xvalid)
        fold_cm.append(cm)


        ## printing results
        print(color_class.BOLD)
        print("fold{}: Accuracy: {}, F1:{}, Roc_Auc: {} ".format(n, round(accuracy,2),round(f1,
2),round(roc_auc,2)))
        print(color_class.END)

        n+=1
```

EWHA EURON

4.6 Modeling



F1 score, AUC, 혼동행렬 등의 지표들을 통해 선택된 분류 모델

: Logistic, LightGBM, XGBoost, RandomForest, AdaBoost

EWHA EURON

05. H&M recommendation

EWHA EURON

#5.1 H&M recommendation 대회 소개

Overview Data Code Discussion Leaderboard Rules **Late Submission** ...

Overview

- Description
- Evaluation
- Timeline
- Prizes

H&M Group is a family of brands and businesses with 53 online markets and approximately 4,850 stores. Our online store offers shoppers an extensive selection of products to browse through. But with too many choices, customers might not quickly find what interests them or what they are looking for, and ultimately, they might not make a purchase. To enhance the shopping experience, product recommendations are key. More importantly, helping customers make the right choices also has a positive implications for sustainability, as it reduces returns, and thereby minimizes emissions from transportation.

In this competition, H&M Group invites you to develop product recommendations based on data from previous transactions, as well as from customer and product meta data. The available meta data spans from simple data, such as garment type and customer age, to text data from product descriptions, to image data from garment images.

There are no preconceptions on what information that may be useful – that is for you to find out. If you want to investigate a categorical data type algorithm, or dive into NLP and image processing deep learning, that is up to you.

Submissions are evaluated according to the Mean Average Precision @ 12 (MAP@12):

$$MAP@12 = \frac{1}{U}\sum_{u=1}^{U}\frac{1}{min(m,12)}\sum_{k=1}^{min(n,12)}P(k)\times rel(k)$$

where U is the number of customers, $P(k)$ is the precision at cutoff k, n is the number predictions per customer, m is the number of ground truth values per customer, and $rel(k)$ is an indicator function equaling 1 if the item at rank k is a relevant (correct) label, zero otherwise.

- ✓ H&M 패션 추천 대회
  - ✓ 이전 구매에 기반해 추천 제공
- ✓ 고객 및 제품 메타 데이터뿐만 아니라 이전 거래의 데이터를 기반으로 제품 권장사항을 개발하도록 함.
- ✓ 사용 가능한 메타 데이터는 의류 유형 및 고객 연령과 같은 간단한 데이터에서 제품 설명의 텍스트 데이터, 의류 이미지의 데이터에 이르기까지 다양함
- ✓ 대회에서는 Mean average precision을 바탕으로 평가됨.

EWHA EURON

#5.2 Data Description - Articles

`article_id` : **A unique identifier of every article.**

`product_code` , `prod_name` : **A unique identifier of every product and its name (not the same).**

`product_type` , `product_type_name` : **The group of product_code and its name**

`graphical_appearance_no` , `graphical_appearance_name` : **The group of graphics and its name**

`colour_group_code` , `colour_group_name` : **The group of color and its name**

`perceived_colour_value_id` , `perceived_colour_value_name` , `perceived_colour_master_id` , `perceived_colour_master_name` : **The added color info**

`department_no` , `department_name` : : **A unique identifier of every dep and its name**

`index_code` , `index_name` : : **A unique identifier of every index and its name**

`index_group_no` , `index_group_name` : : **A group of indeces and its name**

`section_no` , `section_name` : : **A unique identifier of every section and its name**

`garment_group_no` , `garment_group_name` : : **A unique identifier of every garment and its name**

`detail_desc` : : **Details**

	article_id	product_code	prod_name	product_type_no	product_type_name	product_group_name	graphical_appearance_no	graphical_appearance_name	colour_
0	108775015	108775	Strap top	253	Vest top	Garment Upper body	1010016	Solid	9
1	108775044	108775	Strap top	253	Vest top	Garment Upper body	1010016	Solid	10
2	108775051	108775	Strap top (1)	253	Vest top	Garment Upper body	1010017	Stripe	11
3	110065001	110065	OP T-shirt (Idro)	306	Bra	Underwear	1010016	Solid	9
4	110065002	110065	OP T-shirt (Idro)	306	Bra	Underwear	1010016	Solid	10

5 rows × 25 columns

✓ Article - H&M 대회에서 가장 중요한 데이터. 제품 유형, 색상, 의류 종류, 색상 등의 정보가 포함되어 있다.

#5.2 Data Description - Transactions

Transactions data description:

- `t_dat` : **A unique identifier of every customer**
- `customer_id` : **A unique identifier of every customer (in `customers` table)**
- `article_id` : **A unique identifier of every article (in `articles` table)**
- `price` : **Price of purchase**
- `sales_channel_id` : **1 or 2**

```
transactions.head()
```

	t_dat	customer_id	article_id	price	sales_channel_id
0	2018-09-20	000058a12d5b43e67d225668fa1f8d618c13dc232df0ca...	663713001	0.050831	2
1	2018-09-20	000058a12d5b43e67d225668fa1f8d618c13dc232df0ca...	541518023	0.030492	2
2	2018-09-20	00007d2de826758b65a93dd24ce629ed66842531df6699...	505221004	0.015237	2
3	2018-09-20	00007d2de826758b65a93dd24ce629ed66842531df6699...	685687003	0.016932	2
4	2018-09-20	00007d2de826758b65a93dd24ce629ed66842531df6699...	685687004	0.016932	2

✓ Transactions - 거래 데이터로 모든 고객의 식별자와 구매 가격의 정보를 가지고 있다.

EWHA EURON

#5.2 Data Description - customers

Customers data description:

- `customer_id` : **A unique identifier of every customer**
- `FN` : **1 or missed**
- `Active` : **1 or missed**
- `club_member_status` : **Status in club**
- `fashion_news_frequency` : **How often H&M may send news to customer**
- `age` : **The current age**
- `postal_code` : **Postal code of customer**

✓ Customers: 손님의 아이디, 활성 여부, 멤버 여부, 나이 등의 정보가 담겨있다.

```
pd.options.display.max_rows = 50
customers.head()
```

	customer_id	FN	Active	club_member_status	fashion_news_fre
0	00000dbacae5abe5e23885899a1fa44253a17956c6d1c3...	NaN	NaN	ACTIVE	NONE
1	0000423b00ade91418cceaf3b26c6af3dd342b51fd051e...	NaN	NaN	ACTIVE	NONE
2	000058a12d5b43e67d225668fa1f8d618c13dc232df0ca...	NaN	NaN	ACTIVE	NONE
3	00005ca1c9ed5f5146b52ac8639a40ca9d57aeff4d1bd2...	NaN	NaN	ACTIVE	NONE
4	00006413d8573cd20ed7128e53b7b13819fe5cfc2d801f...	1.0	1.0	ACTIVE	Regularly

#5.3 계절별 판매 경향을 기준으로 제품 군집화

```
import pandas as pd
import plotly.express as px
```

```
df_article = pd.read_csv('articles.csv',dtype={'article_id': str})

def set_gender_flg(x):
    x['is_for_male'] = 0
    x['is_for_female'] = 0
    x['is_for_mama'] = 0
    if x['index_group_name'] in ['Ladieswear','Divided']:
        x['is_for_female'] = 1
    if x['index_group_name'] == 'Menswear':
        x['is_for_male'] = 1
    if x['index_group_name'] in ['Baby/Children','Sport']:
        if 'boy' in x['department_name'].lower() or 'men' in x['department_name'].lower():
            x['is_for_male'] = 1
        if 'girl' in x['department_name'].lower() or 'ladies' in x['department_name'].lower():
            x['is_for_female'] = 1
    if x['section_name'] == 'Mama':
        x['is_for_mama'] = 1
    return x

df_article = df_article.apply(set_gender_flg, axis=1)
df_article.head()
```

e_name	colour_group_code	colour_group_name	...	index_group_no	index_group_name	section_no	section_name	garment_group_no	garment_group_name	detail_desc	is_for_male	is_for_female	is_for_mama
Solid	9	Black	...	1	Ladieswear	16	Womens Everyday Basics	1002	Jersey Basic	Jersey top with narrow shoulder straps.	0	1	0
Solid	10	White	...	1	Ladieswear	16	Womens Everyday Basics	1002	Jersey Basic	Jersey top with narrow shoulder straps.	0	1	0
Stripe	11	Off White	...	1	Ladieswear	16	Womens Everyday Basics	1002	Jersey Basic	Jersey top with narrow shoulder straps.	0	1	0

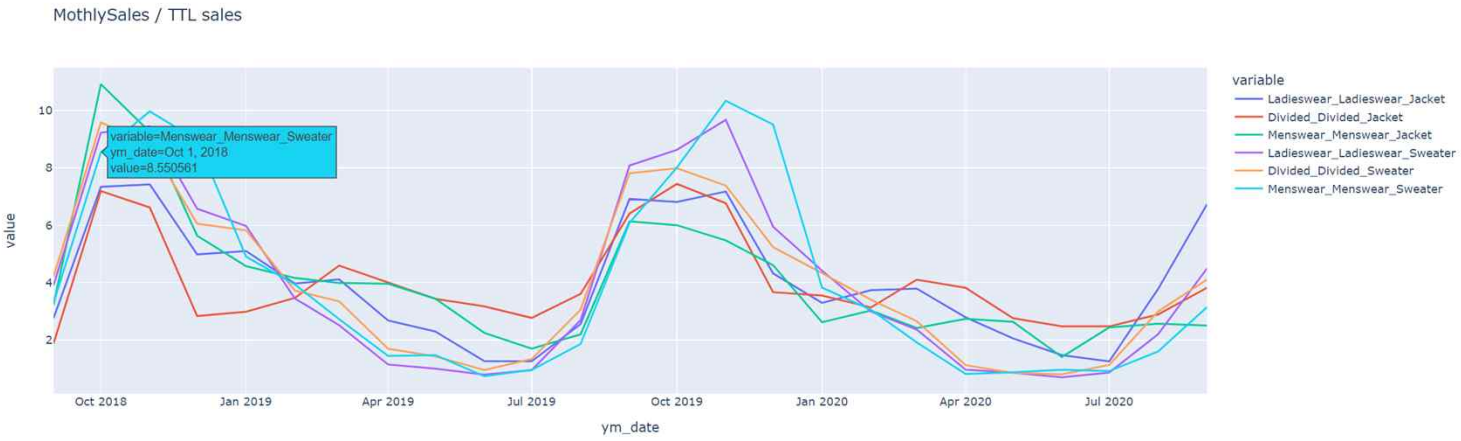
✓ Article 자료에서 is_for_male, is_for_female, is_for_mama 열을 디폴트가 0으로 새로 만들고 index_group_name에 따라 분류

EWHA EURON

#5.3 계절별 판매 경향을 기준으로 제품 군집화

```
df_article['idxgrp_idx_prdtyp'] = df_article['index_group_name'] + '_' + df_article['index_name'] + '_' + df_article['product_type_name']

df_trans = pd.read_csv('/content/drive/MyDrive/transactions_train.csv',dtype={'article_id': str})
df_trans = df_trans.sample(frac=0.1) # downsampling due to memory limits
df_trans['t_dat'] = pd.to_datetime(df_trans['t_dat'])
df_trans['YYYY_MM'] = df_trans['t_dat'].dt.year.astype(str) + '_' + df_trans['t_dat'].dt.month.astype(str)
df_trans['year'] = df_trans['t_dat'].dt.year
df_trans['month'] = df_trans['t_dat'].dt.month
df = pd.merge(df_trans, df_article, on='article_id', how='left')
del df_trans, df_article

dfgrp1 = df.groupby(['idxgrp_idx_prdtyp'])[['price']].sum().reset_index()
dfgrp2 = df.groupby(['idxgrp_idx_prdtyp', 'year', 'month'])[['price']].sum().reset_index()
dfgrp2 = pd.merge(dfgrp2, dfgrp1, on='idxgrp_idx_prdtyp', how='left')
dfgrp2['monthsales/ttl-sales'] = dfgrp2['price_x'] / dfgrp2['price_y'] * 100
dfgrp2['ym_date'] = dfgrp2['year'].astype(str) + '-' + dfgrp2['month'].astype(str) + '-1'
dfgrp2['ym_date'] = pd.to_datetime(dfgrp2['ym_date'])
dfgrp2 = pd.pivot_table(dfgrp2, index='ym_date', columns='idxgrp_idx_prdtyp', values='monthsales/ttl-sales').reset_index().fillna(0)
display(dfgrp2.head())

fig = px.line(dfgrp2, x='ym_date', y=['Ladieswear_Ladieswear_Jacket',
                                      'Divided_Divided_Jacket',
                                      'Menswear_Menswear_Jacket',
                                      'Ladieswear_Ladieswear_Sweater',
                                      'Divided_Divided_Sweater',
                                      'Menswear_Menswear_Sweater'], title="MothlySales / TTL sales")
fig.show()
```

idxgrp_idx_prdtyp	ym_date	Baby/Children_Baby Sizes 50-98_Accessories set	Baby/Children_Baby Sizes 50-98_Baby Bib	Baby/Children_Baby Sizes 50-98_Ballerinas	Baby/Children_Baby Sizes 50-98_Beanie	Baby/Children_Baby Sizes 50-98_Belt	Baby/Children_Bab Sizes 50 98_Blanke
0	2018-09-01	0.0	0.0	3.848800	0.0	11.935387	0.0000
1	2018-10-01	0.0	0.0	8.561861	0.0	14.926713	38.4437



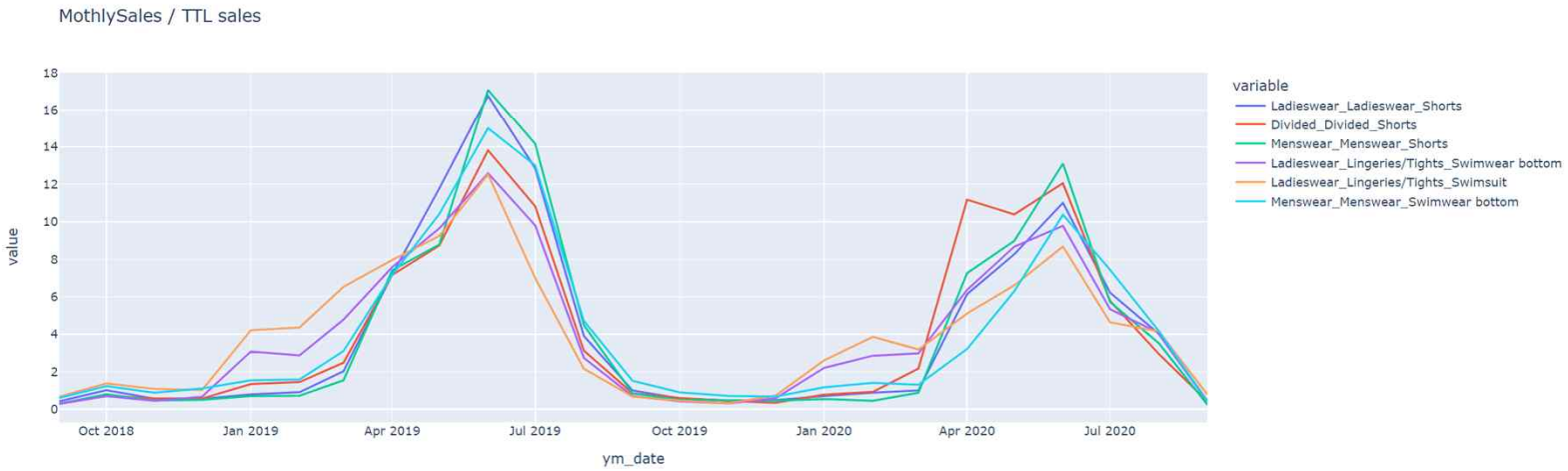
- ✓ 계절에 따라 잘 팔리는 article과 잘 팔리지 않는 article이 있다. 이러한 추세 중 일부이다. 9월 말에 잘 팔릴 것 같지 않은 article은 추천 후보가 될 가능성이 낮으므로 이러한 유형의 제품에 플래그를 지정한다.
- ✓ 월별 집계별 매출 추세를 지수 그룹별, 지수별, 상품유형별로 분류하여 확인할 수 있음.
- ✓ 그래프를 통해 재킷이 9월 말에 잘 팔린다는 사실을 확인 가능. Ladieswear_Ladieswear_Jacket 판매와 음의 상관 관계가 있는 article은 9월 말에 잘 팔리지 않는 제품이라고 예측 가능.

EWHA EURON

#5.3 계절별 판매 경향을 기준으로 제품 군집화

```
display(dfgrp2.corr()[['Ladieswear_Ladieswear_Jacket']].reset_index().sort_values(by='Ladieswear_Ladieswear_Jacket').head(10))

fig = px.line(dfgrp2, x='ym_date', y=['Ladieswear_Ladieswear_Shorts',
                                      'Divided_Divided_Shorts',
                                      'Menswear_Menswear_Shorts',
                                      'Ladieswear_Lingeries/Tights_Swimwear bottom',
                                      'Ladieswear_Lingeries/Tights_Swimsuit',
                                      'Menswear_Menswear_Swimwear bottom'], title="MothlySales / TTL sales")
fig.show()
```

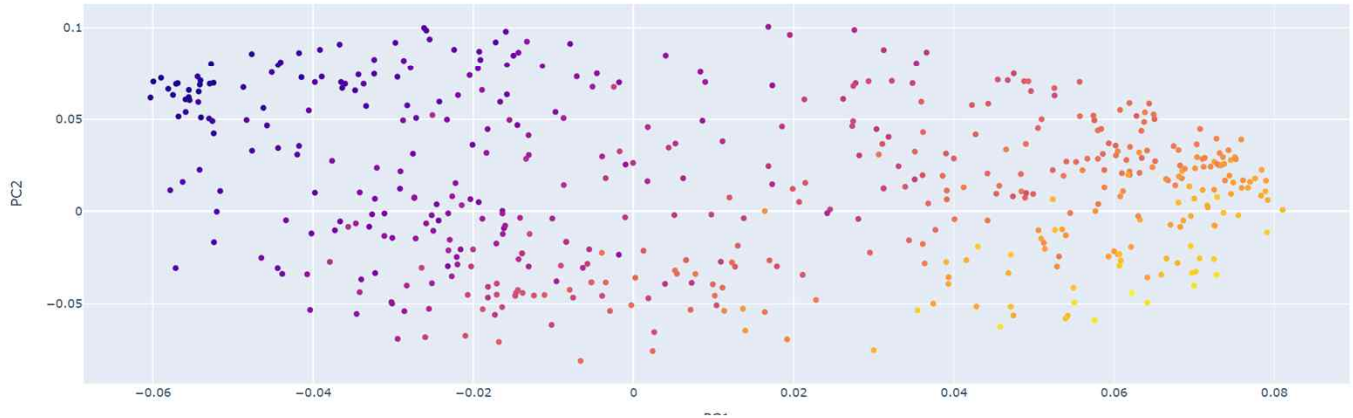
idxgrp_idx_prdtyp	idxgrp_idx_prdtyp	Ladieswear_Ladieswear_Jacket
381	Ladieswear_Lingeries/Tights_Bikini top	-0.789174
398	Ladieswear_Lingeries/Tights_Swimwear bottom	-0.786034
242	Divided_Divided_Shorts	-0.739839
460	Menswear_Menswear_Swimwear bottom	-0.736964
397	Ladieswear_Lingeries/Tights_Swimsuit	-0.734733
312	Ladieswear_Ladies Accessories_Wedge	-0.732673
453	Menswear_Menswear_Shorts	-0.730475
359	Ladieswear_Ladieswear_Shorts	-0.729501
376	Ladieswear_Ladieswear_Vest top	-0.727123
491	Sport_Sport_Shorts	-0.720590



- ✓ Ladieswear_Ladieswear_Jacket과의 상관 관계를 계산하고 가장 강한 음의 상관 관계를 가진 일부 article을 선택하여 가설 테스트.
- ✓ 상관 관계와의 상관성이 확실히 있음. 9월 말에 반바지와 수영복을 구매하는 고객은 거의 없음.
- ✓ 이 결과를 바탕으로 Ladieswear_Ladies_Jacket 상관 계수는 9월 말 제품의 에 대한 정량적 지표로 있다고 기대 가능.

#5.3 계절별 판매 경향을 기준으로 제품 군집화

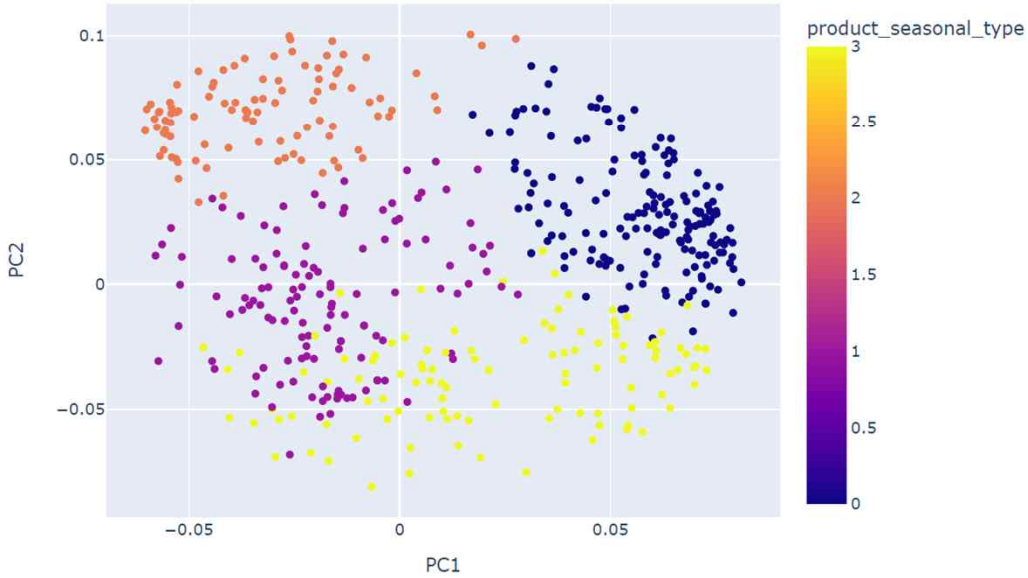
```
from sklearn.decomposition import PCA
from sklearn.preprocessing import StandardScaler

feat_cols = [col for col in dfgrp2.columns if col != 'ym_date']

df_pca = StandardScaler().fit_transform(dfgrp2[feat_cols])
model_pca = PCA(n_components=5)
model_pca.fit(df_pca)
feature = model_pca.transform(df_pca)

df_eigen = model_pca.components_.T
df_eigen = pd.DataFrame(df_eigen,
                        index=None,
                        columns=['PC1','PC2','PC3','PC4','PC5'])
df_eigen['idxgrp_idx_prdtyp'] = feat_cols
df_eigen = pd.merge(
    df_eigen,
    dfgrp2.corr()[['Ladieswear_Ladieswear_Jacket']].reset_index().rename(columns={'Ladieswear_Ladieswear_Jacket': 'autumn_sales_indicator'}),
    on='idxgrp_idx_prdtyp',
    how='left'
)
px.scatter(df_eigen, x='PC1', y='PC2', hover_name='idxgrp_idx_prdtyp', color='autumn_sales_indicator')
```



- ✓ GMM 클러스터링으로 제품 분류
- ✓ 각 제품 유형에 대한 판매에 강한 계절성이 있음을 발견함. 반면 악세사리 같이 수요가 크게 변하지 않는 일부 제품이 있음.
- ✓ 데이터가 제대로 군집화 되면 각 제품 유형을 어느 계절에 판매해야 하는지 입력하여 나중에 추천 모델을 만드는 데 유용함.
- ✓ 차원 축소는 주성분 분석에 의해 수행되며 각 제품 유형에 대한 주성분 점수는 산점도로 시각화됨.
- ✓ autumn_sales_indicateor는 대부분 PC1에서 설명하며 재킷, 스웨터 같은 제품은 오른쪽 하단에 있고 반바지, 수영복은 왼쪽 상단에 있음.

#5.3 계절별 판매 경향을 기준으로 제품 군집화

```
from sklearn.mixture import GaussianMixture
gmm = GaussianMixture(n_components=4, covariance_type='full')
gmm.fit(df_eigen[['PC1','PC2','PC3']])
df_eigen['product_seasonal_type'] = gmm.predict(df_eigen[['PC1','PC2','PC3']])
df_eigen['prob_cluster1'] = gmm.predict_proba(df_eigen[['PC1','PC2','PC3']])[:,0]
df_eigen['prob_cluster2'] = gmm.predict_proba(df_eigen[['PC1','PC2','PC3']])[:,1]
df_eigen['prob_cluster3'] = gmm.predict_proba(df_eigen[['PC1','PC2','PC3']])[:,2]
df_eigen['prob_cluster4'] = gmm.predict_proba(df_eigen[['PC1','PC2','PC3']])[:,3]
px.scatter(df_eigen, x='PC1', y='PC2', hover_name='idxgrp_idx_prdtyp', color='product_seasonal_type')
```

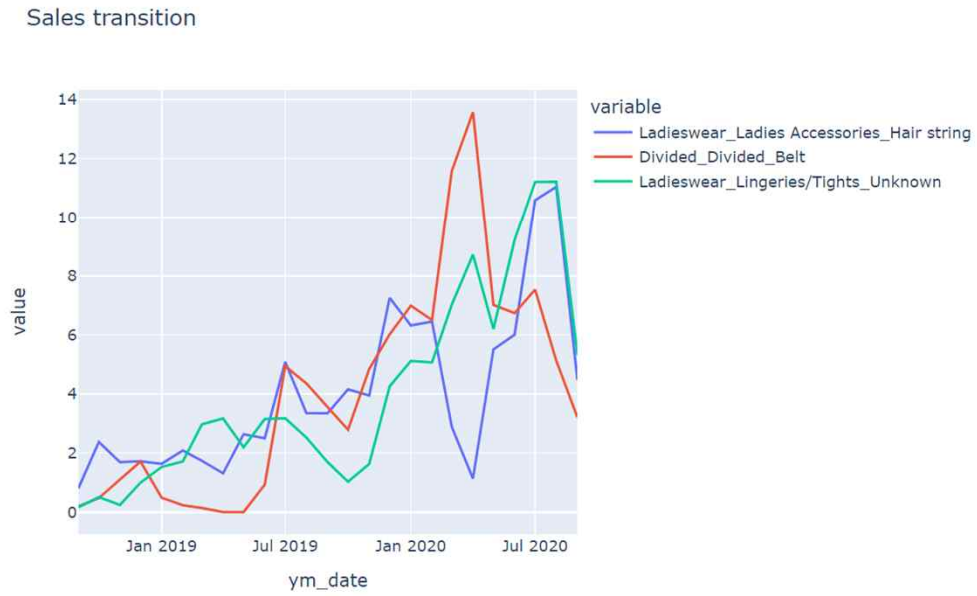


- ✓ 재킷과 스웨터는 클래스 1과 2로 반바지 등은 클래스 3으로 분류됨.
- ✓ 아직 추천 모델의 정확도를 보면 적절한 매개변수를 최적화할 여지가 있음.

EWHA EURON

#5.3 계절별 판매 경향을 기준으로 제품 군집화

```
fig = px.line(dfgrp2, x='ym_date', y=['Ladieswear_Ladies Accessories_Hair string',
                                      'Divided_Divided_Belt',
                                      'Ladieswear_Lingeries/Tights_Unknown'], title="Sales transition")
fig.show()
```




```
df = pd.merge(
    df,
    df_eigen[['idxgrp_idx_prdtyp','autumn_sales_indicator', 'product_seasonal_type', 'prob_cluster1', 'prob
    on='idxgrp_idx_prdtyp',
    how='left'
)
del dfgrp1, dfgrp2
df.head()
```

n_sales_indicator	product_seasonal_type	prob_cluster1	prob_cluster2	prob_cluster3	prob_cluster4
-0.574373	2	6.363056e-10	0.006056	0.993944	1.778720e-10
-0.661580	2	1.408225e-11	0.000226	0.999774	1.145774e-11
-0.505287	1	4.269176e-14	0.688068	0.305721	6.210442e-03
-0.517152	1	1.689499e-14	0.775790	0.182815	4.139420e-02
0.070845	2	2.215944e-03	0.124895	0.872889	1.825059e-07

- ✓ 클래스 0으로 식별된 일부 제품의 판매 계절성 확인
- ✓ 데이터 프레임에 제품 계절 타입 및 각 클러스터 확률 추가

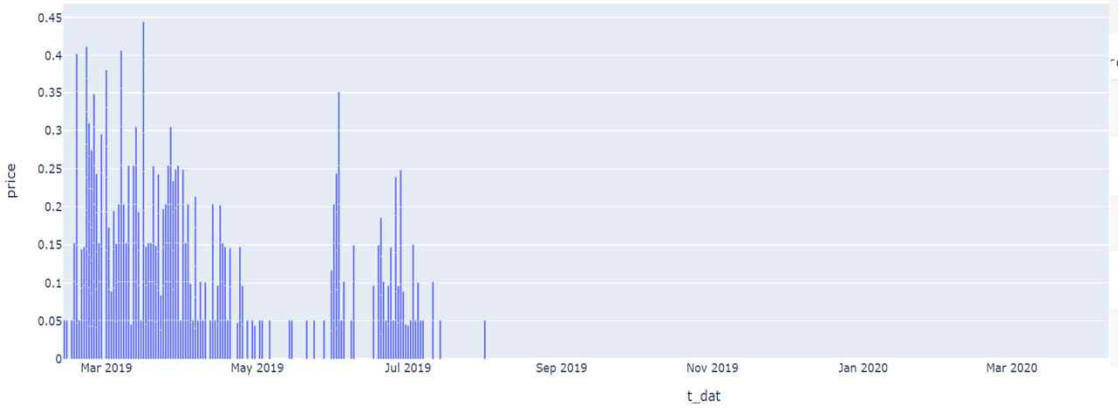
EWHA EURON

#5.3 계절별 판매 경향을 기준으로 제품 군집화

```
dfagg1 = df.groupby(['article_id'])[['price']].sum().reset_index().rename(columns={'price': 'article_ttl_sales'})
dfagg2 = df[df['t_dat'] < '2019-09-01'].groupby(['article_id'])[['price']].sum().reset_index().rename(columns={'price': 'before_201909_article_ttl_sales'})
dfagg3 = df[df['t_dat'] > '2019-09-01'].groupby(['article_id'])[['price']].sum().reset_index().rename(columns={'price': 'after_201909_article_ttl_sales'})

dfagg1 = pd.merge(dfagg1, dfagg2, on='article_id')
dfagg1 = pd.merge(dfagg1, dfagg3, on='article_id')
dfagg1['before_201909_article_ttl_sales_ratio'] = dfagg1['before_201909_article_ttl_sales'] / dfagg1['article_ttl_sales'] * 100
dfagg1['not_for_sales_flg'] = (dfagg1['before_201909_article_ttl_sales_ratio'] > 95).astype(int)

display(dfagg1.sort_values(by='before_201909_article_ttl_sales_ratio', ascending=False).head(5))
dftmp = df[df['article_id']=='0698276001'].sort_values(by='t_dat')
fig = px.bar(dftmp, x='t_dat', y='price', title="Not for sales article example")
fig.show()
```

	article_id	article_ttl_sales	before_201909_article_ttl_sales	after_201909_article_ttl_sales	before_201909_article_ttl_sales_ratio	not_for_sales_flg
10645	0698276001	16.301678	16.298305	0.003373	99.979310	1
78	0188183023	3.749797	3.748458	0.001339	99.964292	1
2256	0539723007	8.556000	8.549237	0.006763	99.920959	1
2197	0537688015	10.452034	10.443576	0.008458	9	
10305	0695545005	10.023339	10.015119	0.008220	9	

Not for sales article example



- ✓ **2020년 9월 말에 더 이상 판매되지 않을 제품 감지**
- ✓ 2020년 9월에서 더 이상 사용할 수 없다는 생각되는 article에 플래그 지정
- ✓ article_id를 살펴보며 2019년 9월 이전의 매출이 해당 article의 매출이 95% 이상 차지한다면 2020년 9월 말에 해당 article이 더 이상 판매되지 않거나 매우 드물게 판매된다고 가정하고 추천에서 제외.

```
df = pd.merge(
    df,
    dfagg1[['article_id','before_201909_article_ttl_sales_ratio','not_for_sales_flg']],
    on='article_id',
    how='left'
)
df.head()
```

rob_cluster2	prob_cluster3	prob_cluster4	before_201909_article_ttl_sales_ratio	not_for_sales_flg
0.006056	0.993944	1.778720e-10	NaN	NaN
0.000226	0.999774	1.145774e-11	NaN	NaN
0.688068	0.305721	6.210442e-03	0.705802	0.0
0.775790	0.182815	4.139420e-02	NaN	NaN
0.124895	0.872889	1.825059e-07	8.222472	0.0

- ✓ article_id를 기준으로 데이터 프레임에 2019년 9월 전 판매 비율, not_for_sales_flg 추가

#5.3 계절별 판매 경향을 기준으로 제품 군집화 - 결론

✓제품을 그룹, 인덱스, 타입 등으로 분류하여 남성용, 여성용, 유야용, 임산부용인지 등으로 제품을 특성을 파악할 수 있음. 이것들을 위해 플래그를 설정하는 프로그램을 도입했었고 결과적으로 고객을 이해할 때 유용한 정보가 될 수 있음.

✓특히 재킷 같이 일부 제품의 판매는 다른 제품들보다 계절에 따라 팔리는 양의 차이가 눈에 띄게 나타남. 재킷의 판매량을 연관시켜 autumn_sales_indicator를 도입해 가을에 잘 팔리는 제품과 그렇지 않은 제품을 정량적으로 식별함.

✓2020년 9월 말까지 더 이상 사용할 수 없는 제품에 플래그를 지정하는 프로그램 도입. 최종적으로 어떤 제품을 추천할 지 결정하는 데 효과적으로 사용 가능.

#5.4 월별 매출을 기준으로 제품 군집화

```
import matplotlib.pyplot as plt
import seaborn as sns; sns.set()

from collections import Counter, defaultdict
from PIL import Image
from pathlib import Path
path = Path("/kaggle/input/h-and-m-personalized-fashion-recommendations/")

def show_images(article_ids, cols=1, rows=-1):
    if isinstance(article_ids, int) or isinstance(article_ids, str):
        article_ids = [article_ids]
    article_count = len(article_ids)
    if rows < 0: rows = (article_count // cols) + 1
    plt.figure(figsize=(3 + 3.5 * cols, 3 + 5 * rows))
    for i in range(article_count):
        article_id = ("0" + str(article_ids[i]))[-10:]
        plt.subplot(rows, cols, i + 1)
        plt.axis('off')
        plt.title(article_id)
        try:
            image = Image.open(f"/kaggle/input/h-and-m-personalized-fashion-recommendations/images/{article_id[:3]}/{article_id}.jpg")
            plt.imshow(image)
        except:
            pass

articles = pd.read_csv(path / "articles.csv", dtype = {'article_id': str})

train = pd.read_csv(path / "transactions_train.csv", dtype = {'article_id': str})
train = train[["t_dat", "article_id", "sales_channel_id"]]
train["t_dat"] = pd.to_datetime(train["t_dat"])
train = train.query("sales_channel_id == 2")
train = train.sort_values(["article_id", "t_dat"], ascending=False)
```

- ✓ K-means를 사용하여 월별 매출을 기준으로 제품 군집화
- ✓ 품목의 판매 주기와 계절성에 대해 알아보는 노트북
- ✓ 가장 이른 구매 날짜와 가장 늦은 구매 날짜 사이의 일수로 계산된 판매 기간 사용
- ✓ 옷 종류에 대한 이미지 로드하고 transactions_train 자료 저장

EWHA EURON

#5.4 월별 매출을 기준으로 제품 군집화

```
sales_counts = Counter(train.article_id)
for i in articles.index:
    articles.at[i, "sales_count"] = sales_counts[articles.at[i, "article_id"]]

period_df = train.groupby(["article_id"])["t_dat"].agg(lambda x: (list(x)[0], list(x)[-1])).reset_index()
period_df = period_df.merge(articles["article_id"], how="right")

articles["latest"] = period_df["t_dat"].apply(lambda x: None if pd.isna(x) else x[0])
articles["earliest"] = period_df["t_dat"].apply(lambda x: None if pd.isna(x) else x[1])
articles["period"] = (articles.latest.values - articles.earliest).dt.total_seconds() // (60 * 60 * 24)

monthly_sales = {}
YM = [201809, 201810]
while YM[0] < 202010:
    start, end = "-".join(map(str, [YM[0] // 100, YM[0] % 100, 1])), "-".join(map(str, [YM[1] // 100, YM[1] % 100, 1]))
    monthly_sales = Counter(train.query(f"'{start}' <= t_dat < '{end}'").article_id)

    articles[YM[0]] = 0
    for i in articles.index:
        articles.at[i, YM[0]] = monthly_sales[articles.at[i, "article_id"]]

    print("\r Done :", YM[0], end="")
    YM[0] = YM[1]
    YM[1] = (YM[1] + 100 - 11) if YM[1] % 100 == 12 else (YM[1] + 1)

articles.iloc[:, 5:].to_csv("articles_sales_extension.csv")
```

✓ 가장 이른 구매 날짜와 가장 늦은 구매 날짜 사이의 일수로 계산된 판매 기간 사용

EWHA EURON

#5.4 월별 매출을 기준으로 제품 군집화

```
def plot_sales(article_id, imshow=False):
    plt.figure(figsize=(24, 1.5))
    plot_df = articles.query(f"article_id == '{article_id}'")
    sns.barplot(x=plot_df.columns[29:], y=list(*plot_df.values)[29:], palette=sns.husl_palette(12))
    plt.title(" ".join(["Monthly Sales of ID :", article_id, "    earliest :", str(plot_df.iloc[0, 27])[:10], "    latest :",
str(plot_df.iloc[0, 26])[:10]]))
    if imshow:
        show_images(articles.article_id[loc])
```

Hide code

```
temp_df = articles.sort_values([202009, "period"], ascending=False).head(100)
longsellers = temp_df.query("300 < period")
newitems = temp_df.query("period <= 30")
temp_df[["article_id", "product_type_name", "colour_group_name", "period"]].head(30)
```

	article_id	product_type_name	colour_group_name	period
3091	0448509014	Trousers	Blue	733.0
67522	0751471001	Trousers	Black	449.0
53892	0706016001	Trousers	Black	664.0
103796	0915529003	Sweater	Black	34.0
104045	0918292001	Leggings/Tights	Black	54.0

✓ Sales period(판매 시기)를 Longtime sellers, New items, Summer clothing, Autumn clothing으로 분류함.

EWHA EURON

#5.4 월별 매출을 기준으로 제품 군집화

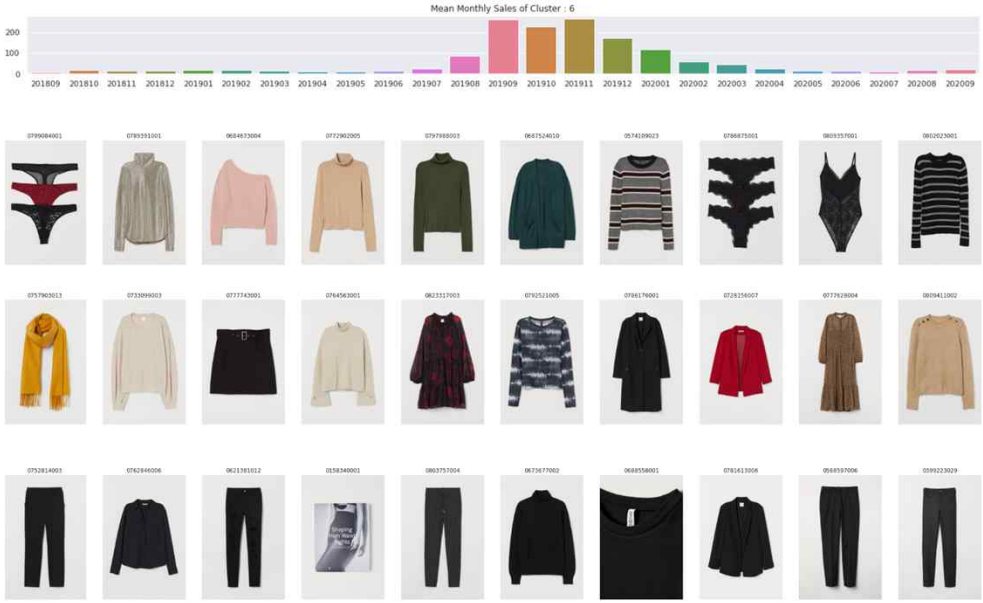
```
articles = articles.sort_values(by="sales_count")

from sklearn import cluster
n_clusters = 9
model = cluster.KMeans(n_clusters=n_clusters)
model.fit(articles.iloc[:,29:]) # consider the period

def plot_cluster(k, n=10):
    temp_df = articles[model.labels_==k]
    plt.figure(figsize=(24, 1.5))
    plot_df = temp_df.iloc[:,29:].describe().loc[["mean"]]
    sns.barplot(x=plot_df.columns, y=list(*plot_df.values), palette=sns.husl_palette(12))
    plt.title(" ".join(["Mean Monthly Sales of Cluster :", str(k)]))
    show_images(list(temp_df.article_id.values[:n]), 10)
    show_images(list(temp_df.article_id.values[-n:]), 10)
    return temp_df.iloc[:,[0] + [i for i in range(29, 54)]]
```

- ✓ K-means 클러스터링
- ✓ n_cluster를 9로 하여 k-means 클러스터링을 하고 시각화함.

EWHA EURON

#5.4 월별 매출을 기준으로 제품 군집화

```
temp_df = plot_cluster(6, 20)
for ID in list(temp_df.head(3).article_id): plot_sales(ID)
for ID in list(temp_df.tail(3).article_id): plot_sales(ID)
```

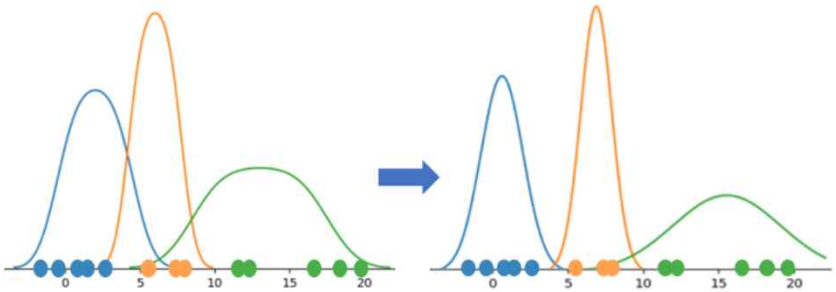


✓ 순서대로 0부터 9까지의 클러스터링 그룹을 시각화한다. 이 중 대표적으로 cluster 그룹 6을 시각화한 것을 가져왔다.

✓ 각각의 제품별로 시각화한 결과이다. 각각의 판매 기간동안 어떤 분포로 판매되었는지 나타나있다.

#5.5 H&M 대회 군집화 기법 복습 - GMM, K-means

1. 계절별 판매 경향을 기준으로 제품 군집화 - GMM clustering

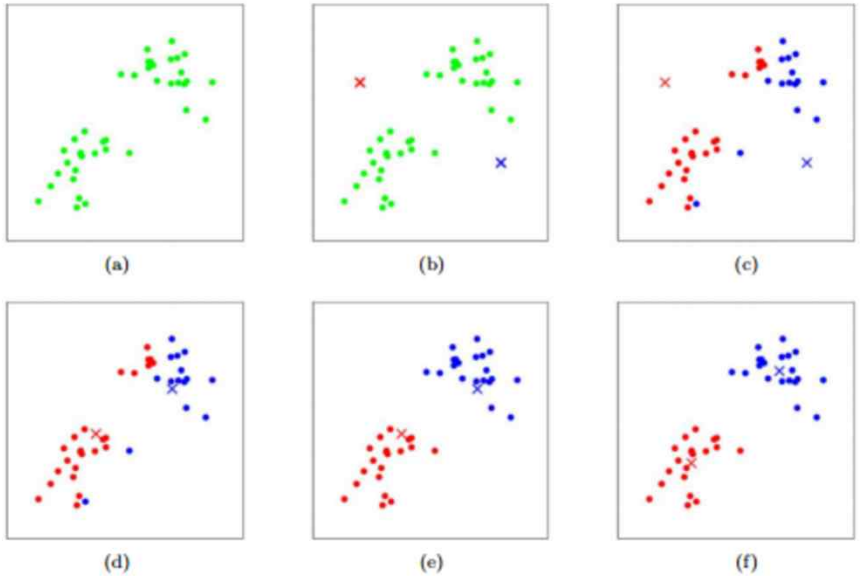


✓ 군집화를 적용하고자 하는 데이터가 여러 개의 가우시안(정규) 분포를 갖는 데이터 집합들이 섞여 생성된 것이라는 가정하에 군집화를 수행하는 방식

✓ GaussianMixture 객체의 가장 중요한 초기화 파라미터 : n_components

✓ 개별 데이터 포인트를 특정 그룹에 먼저 할당하고 그룹에 속한 데이터들로 기준 값을 다시 계산한다는 점에서 GMM 알고리즘과 K-means 알고리즘 수행 방식이 유사함.

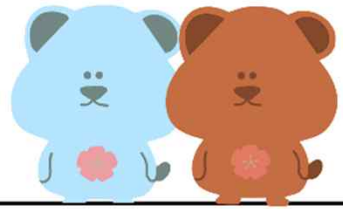
2. 월별 매출을 기준으로 제품 군집화 - K-means clustering



✓ K 개의 임의의 중심점에 데이터들을 할당하고 군집의 중심점을 업데이트하는 과정을 반복한다.

✓ 좋은 군집이란 각 군집의 샘플이 가까이 뭉쳐있고 다른 군집의 샘플과 멀리 떨어져 있는 군집이다.

THANK YOU



EWHA EURON