

마케팅 데이터 활용 군집분석

갈라놓을거야...
고객들 사이 ♥



목차

1. 데이터 분석 개요

- a. 분석 목적
- b. 데이터 설명

2. 데이터 탐색 및 전처리

- a. 결측치 제거
- b. 데이터 타입 변경
- c. 파생변수 생성
- d. product_category 정리
- e. 구매 1년 이하 고객들로 축소 및 이상치 제거

3. 클러스터링

4. 클러스터 EDA

5. 인사이트 및 마케팅 전략 제안

1. 데이터 분석 개요 - 분석 목적

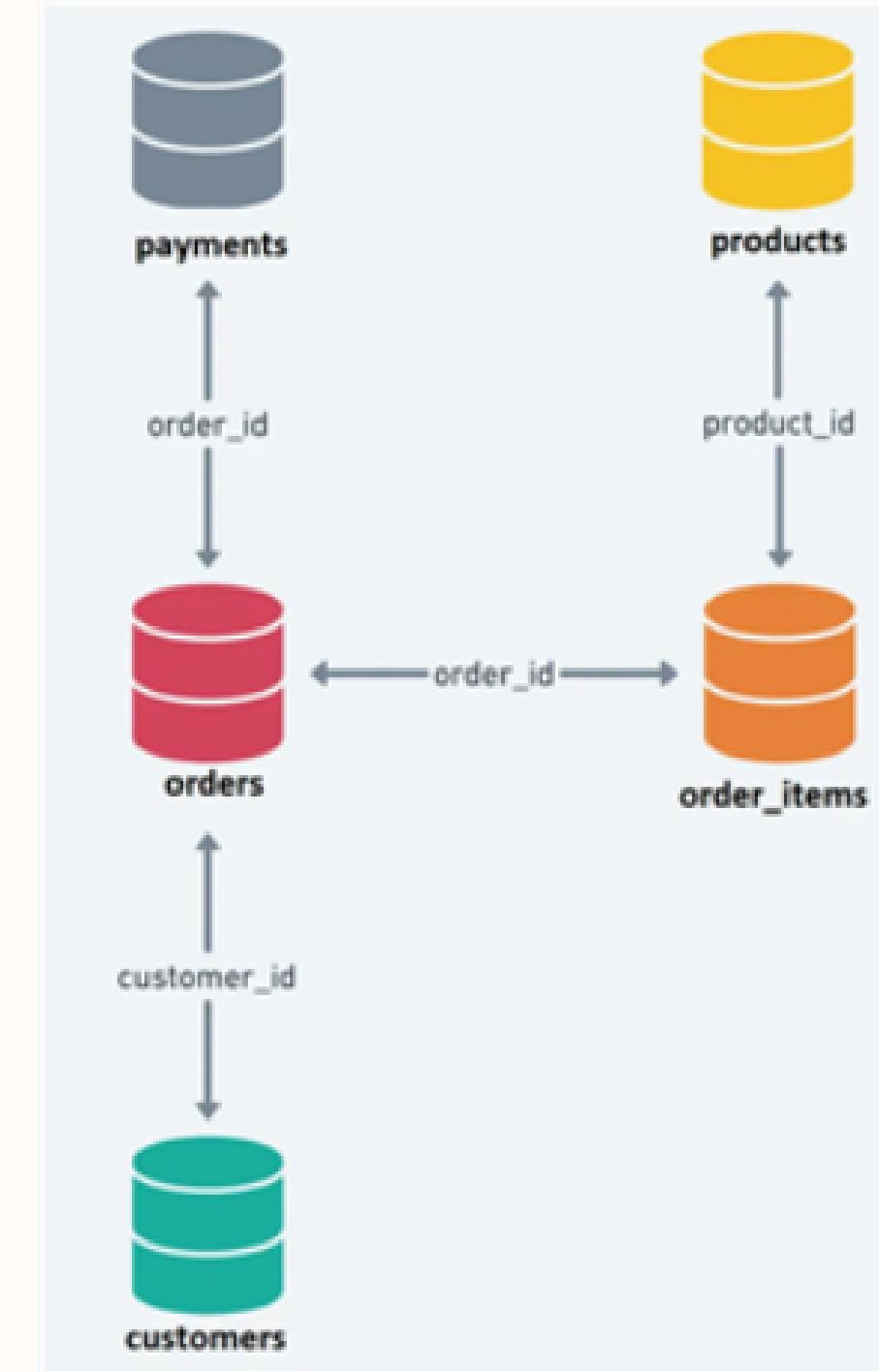
주제

군집분석을 통한 고객 맞춤 서비스 제공

내용

- 상품 특성을 파악해 고객 분류
- 분류 고객에 맞춤 서비스 제공

1. 데이터 분석 개요 - 데이터 설명



orders : 주문 테이블

order_items : 주문 항목 테이블

products : 제품 테이블

payments : 결제 테이블

customers : 고객 테이블

1. 데이터 분석 개요 - 데이터 설명

데이터 기본 정보

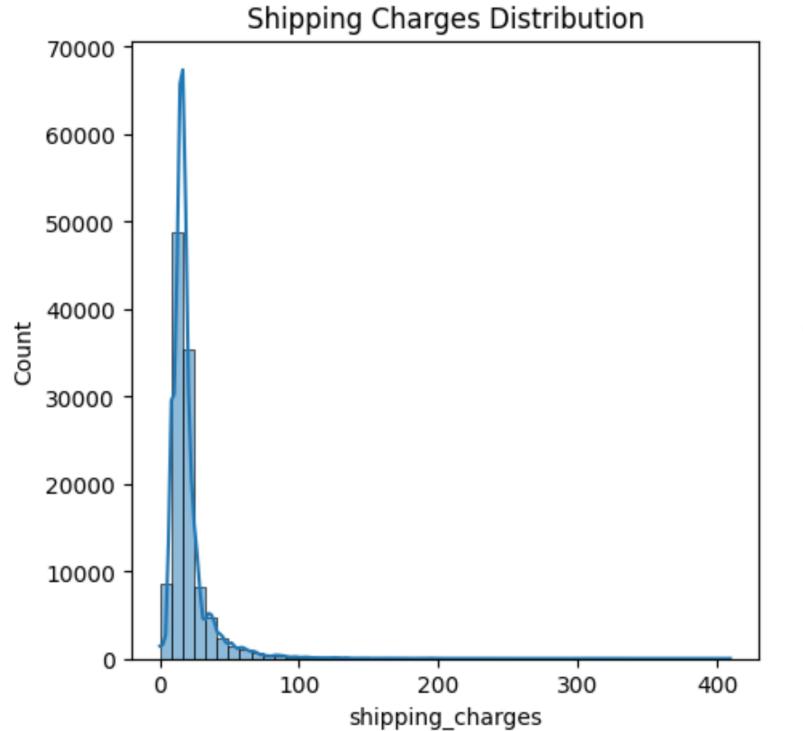
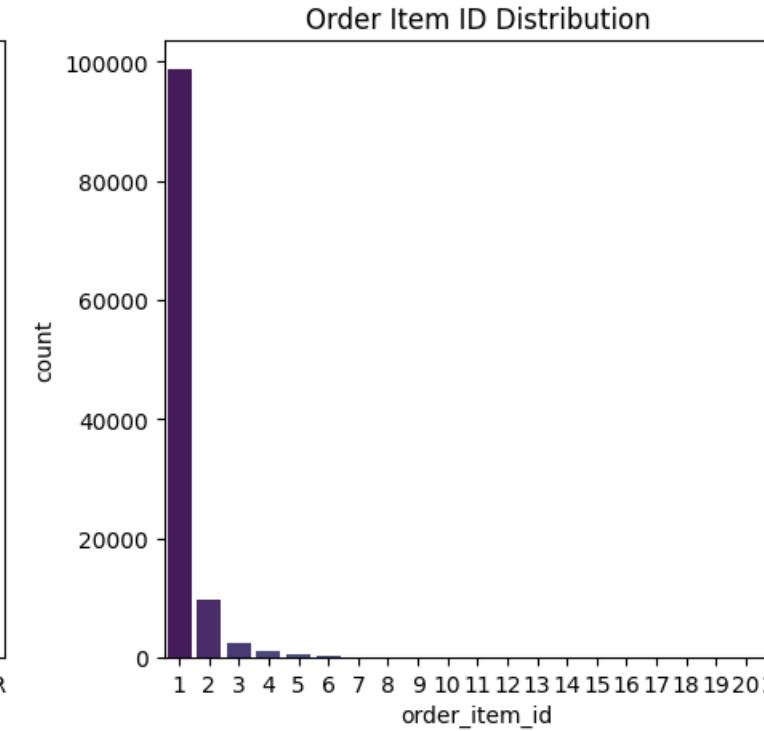
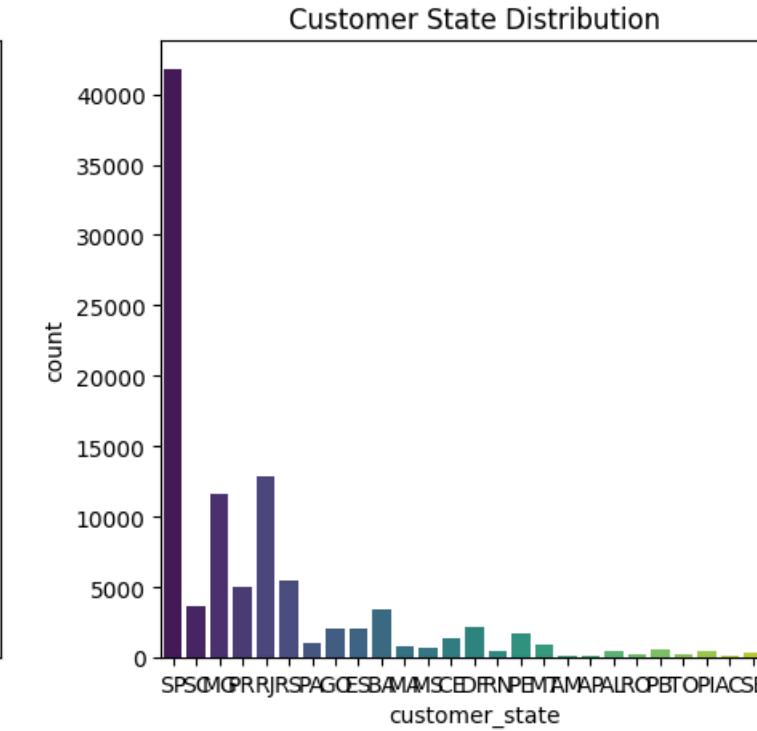
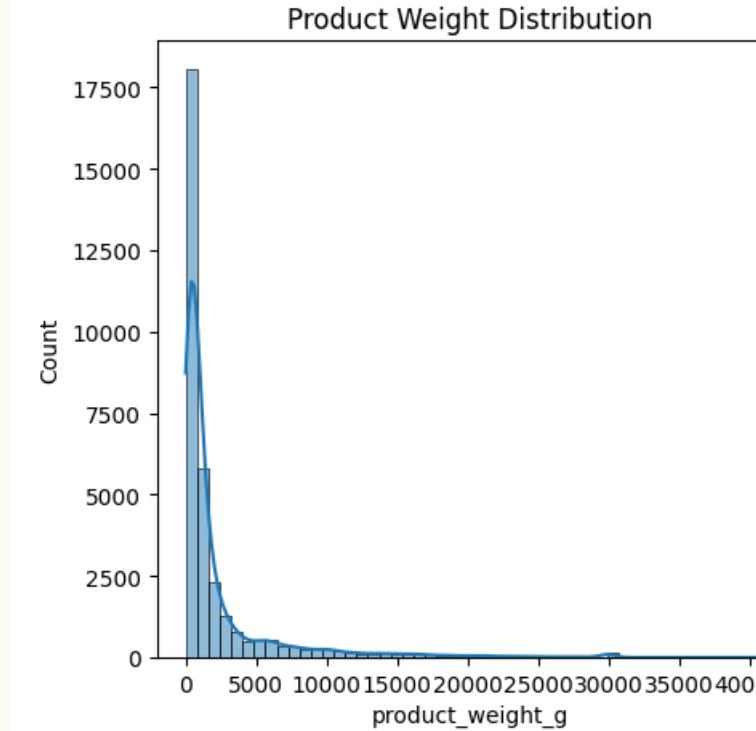
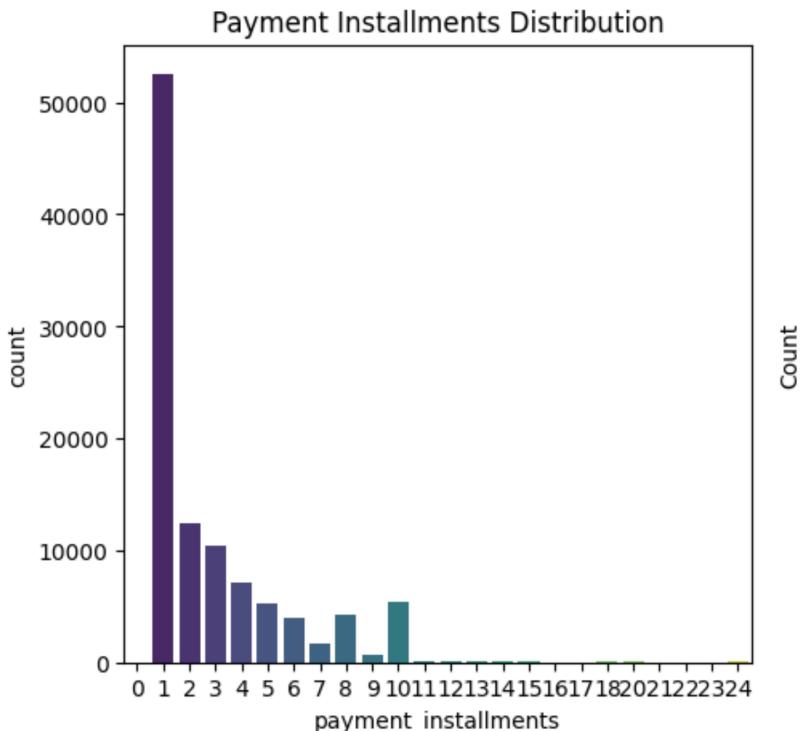
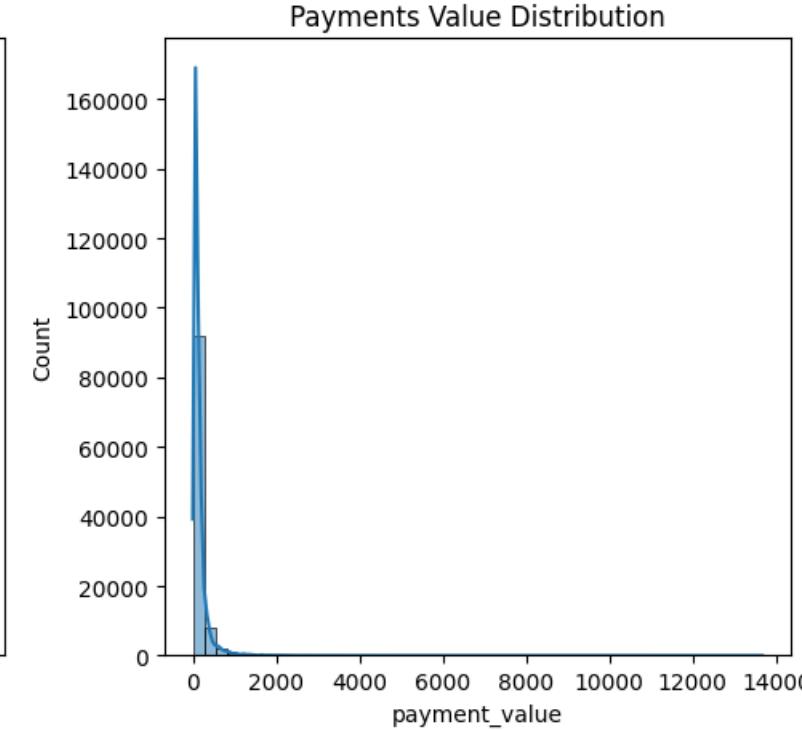
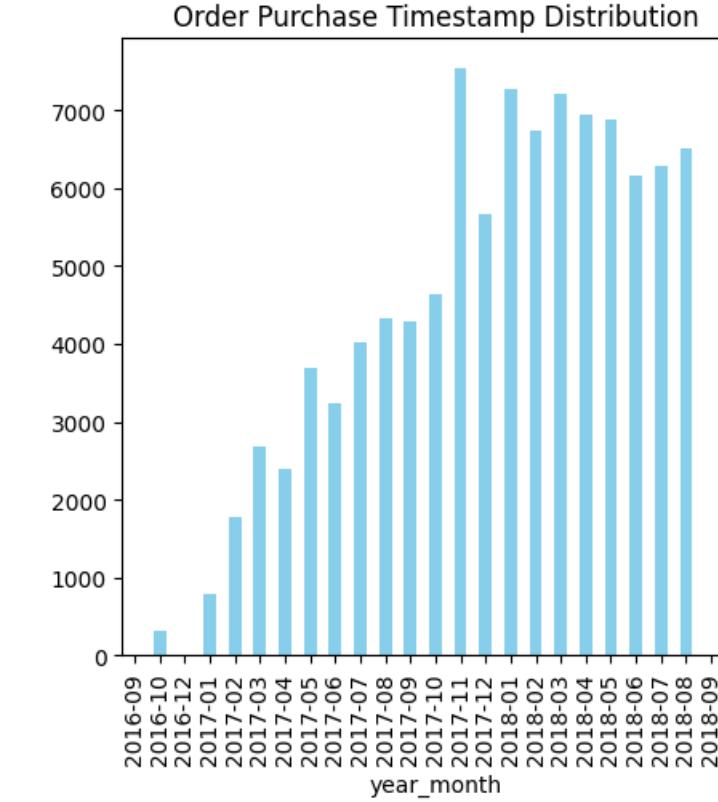
```
1 data_cleaned_df.describe().round(2)
```

✓ 0.0s

Python

	order_item_id	price	shipping_charges	payment_installments	payment_value	product_weight_g	product_length_cm	product_height_cm	product_width_cm	customer_zip_code_prefix
count	119614.00	119614.00	119614.00	119614.00	119614.00	119614.00	119614.00	119614.00	119614.00	119614.00
mean	1.19	101.99	18.45	2.85	145.10	1653.63	29.59	15.70	22.53	34806.05
std	0.63	107.05	10.69	2.63	137.73	2515.89	15.49	12.15	11.15	29834.72
min	1.00	0.85	0.00	0.00	0.00	0.00	7.00	2.00	6.00	1003.00
25%	1.00	39.00	12.84	1.00	59.06	283.00	18.00	8.00	15.00	11045.00
50%	1.00	69.90	16.11	1.00	104.52	650.00	25.00	12.00	20.00	24210.00
75%	1.00	125.87	20.16	4.00	177.91	1650.00	37.00	20.00	30.00	58039.00
max	21.00	999.90	99.65	10.00	999.68	14950.00	105.00	105.00	118.00	99980.00

1. 데이터 분석 개요 - 컬럼별 raw data 분포



데이터 전처리



2. 데이터 전처리 - 데이터 정제

(1) 결측치 확인

```
# 데이터 결측치 확인  
data_cleaned_df.info()  
  
→ <class 'pandas.core.frame.DataFrame'>  
RangeIndex: 119614 entries, 0 to 119613  
Data columns (total 22 columns):  
 #   Column           Non-Null Count  Dtype     
---  --  
 0   order_id         119614 non-null  object    
 1   customer_id      119614 non-null  object    
 2   order_purchase_timestamp  119614 non-null  object    
 3   order_approved_at  119614 non-null  object    
 4   order_delivered_timestamp  119614 non-null  object    
 5   order_estimated_delivery_date  119614 non-null  object    
 6   order_item_id     119614 non-null  int64     
 7   product_id        119614 non-null  object    
 8   seller_id         119614 non-null  object    
 9   price             119614 non-null  float64   
 10  shipping_charges 119614 non-null  float64   
 11  payment_type      119614 non-null  object    
 12  payment_installments 119614 non-null  int64     
 13  payment_value      119614 non-null  float64   
 14  product_category_name 119614 non-null  object    
 15  product_weight_g    119614 non-null  float64   
 16  product_length_cm   119614 non-null  float64   
 17  product_height_cm   119614 non-null  float64   
 18  product_width_cm    119614 non-null  float64   
 19  customer_zip_code_prefix 119614 non-null  int64     
 20  customer_city        119614 non-null  object    
 21  customer_state       119614 non-null  object    
dtypes: float64(7), int64(3), object(12)  
memory usage: 20.1+ MB
```

(2) 데이터 Type 변경

```
# item_id, customer_zip_code_prefix 끝자리에서 문자열로 변경  
data_cleaned_df['order_item_id'] = data_cleaned_df['order_item_id'].astype(str)  
data_cleaned_df['customer_zip_code_prefix'] = data_cleaned_df['customer_zip_code_prefix'].astype(str)  
  
# to_datetime  
data_cleaned_df['order_purchase_timestamp'] = pd.to_datetime(data_cleaned_df['order_purchase_timestamp'])  
data_cleaned_df['order_delivered_timestamp'] = pd.to_datetime(data_cleaned_df['order_delivered_timestamp'])  
data_cleaned_df['order_approved_at'] = pd.to_datetime(data_cleaned_df['order_approved_at'])  
data_cleaned_df['order_estimated_delivery_date'] = pd.to_datetime(data_cleaned_df['order_estimated_delivery_date'])  
  
# to_numeric  
data_cleaned_df['order_item_id'] = pd.to_numeric(data_cleaned_df['order_item_id'], errors='coerce').astype(int)
```

- 숫자 → 문자열 변경
- 문자 → 숫자형 변경
- 문자 → datetime 변경

2. 데이터 전처리 - 파생변수 생성

(3) 파생변수 생성

- **total_payment** (총 구매 금액)

price(가격) + shipping charges(배송비)

- **price per unit** (개당 가격)

(price가격 - shipping charge배송비) / orders_item_id 주문 항목 아이디

- **volume** (상품 부피)

product_height_cm 높이 x product_length_cm 길이 x product_width_cm 너비

- **delivery_hours** (배달 시간)

order_delivered_timestamp 제품 도착시간 - order_purchase_timestamp 제품 구입 시간//
3600

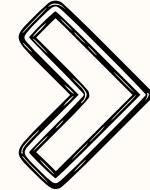
- **Diff_days** (구매 경과일)

max_date(가장 최근 구매일) - order_purchase_timestamp(구매일).일수(days) + 1

2. 데이터 전처리 - 컬럼 정리

(4) Payment Type 범주화

```
['credit_card', 'debit_card', 'voucher', 'wallet']
```



```
array(['credit_card', 'wallet', 'credit_card/voucher', 'debit_card',
       'voucher', 'credit_card/debit_card'], dtype=object)
```

(5) Category 컬럼 정리

유사 카테고리 제품들끼리 묶어 새로운 카테고리로 정리

ex) [food, drinks, food_drink, la_cuisine] → [food]

(6) 최근 1년 이하로 제품 구매한 고객들로 data 축소

```
merged_df = merged_df[merged_df['Diff_days'] <= 365]
merged_df.T
```

2. 데이터 전처리 - 이상치 제거

(7) 이상치 처리

orders 테이블 → '물건구매'가 되기 전 '주문 승인'이 된 건을 이상치로 판단 후 제거

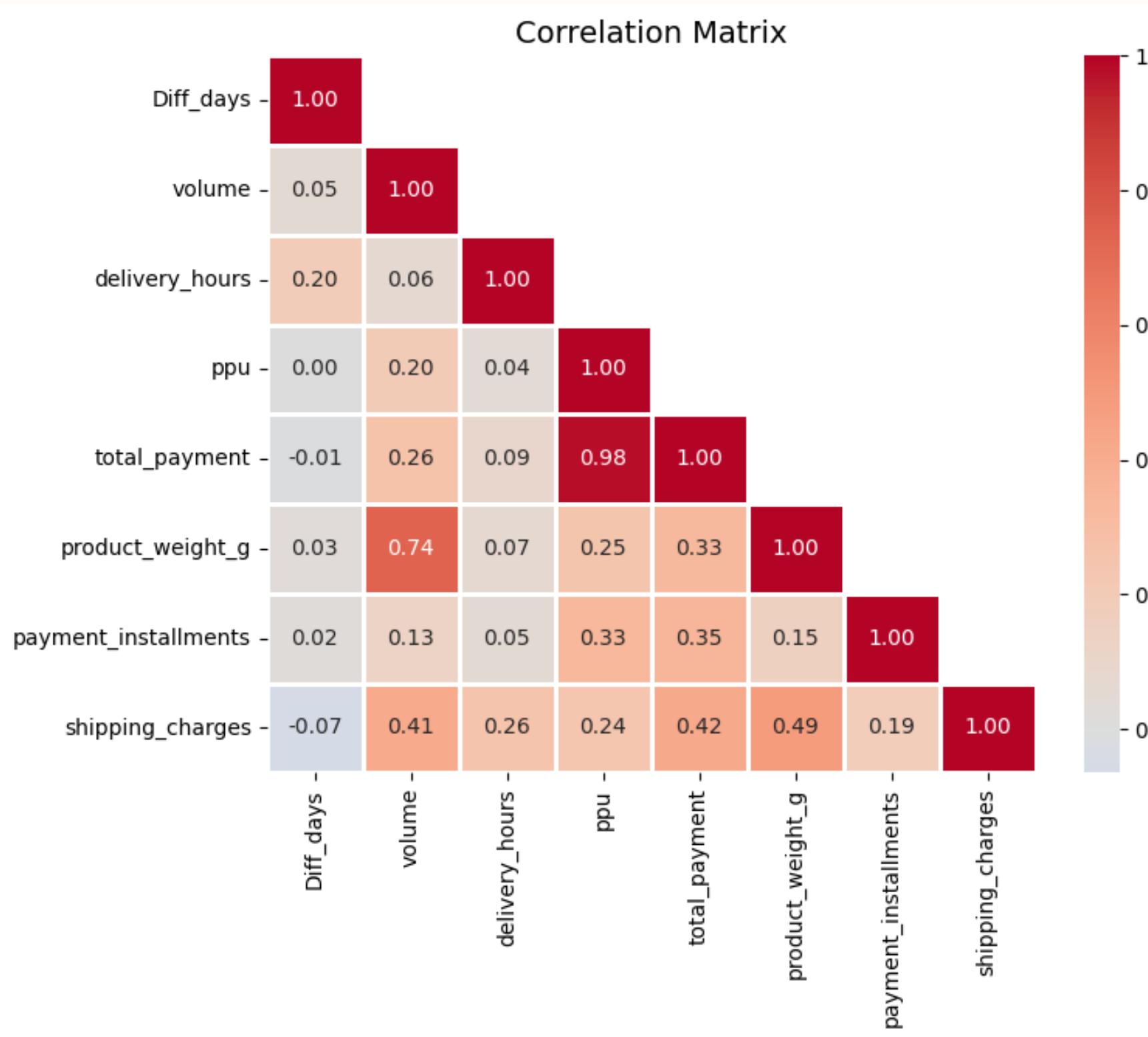
- a. 물건구매가 일어나기 전 주문 승인된 건의 주문ID 구하기
- b. 해당 아이디 merged_df table에서 제거

```
# `물건구매`가 되기 전 `주문 승인`이 된 건
Check_date_outliers = orders[
    (orders['order_purchase_timestamp'] > orders['order_approved_at']) |
    (orders['order_approved_at'] > orders['order_delivered_timestamp']) ]

# 이상치에 해당하는 값을 가진 주문 id 확인
out_ids = Check_date_outliers['order_id'].unique().tolist()

# 'order_id' 열의 값이 out_ids에 포함된 행 삭제
merged_cleaned_df = merged_df[~merged_df['order_id'].isin(out_ids)].reset_index(drop=True)
merged_cleaned_df.T
```

2. 데이터 전처리 - 변수별 상관관계 확인



상관계수 히트맵

- total_payment ↔ ppu 0.98

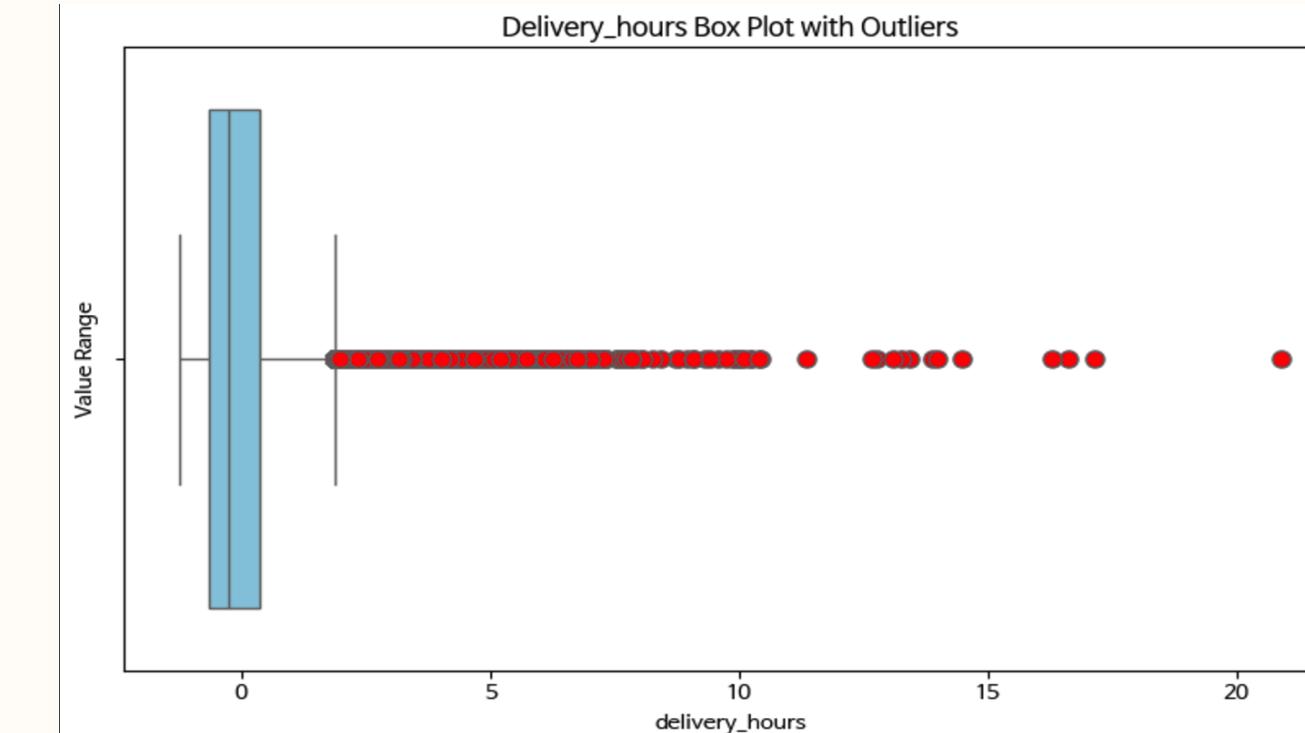
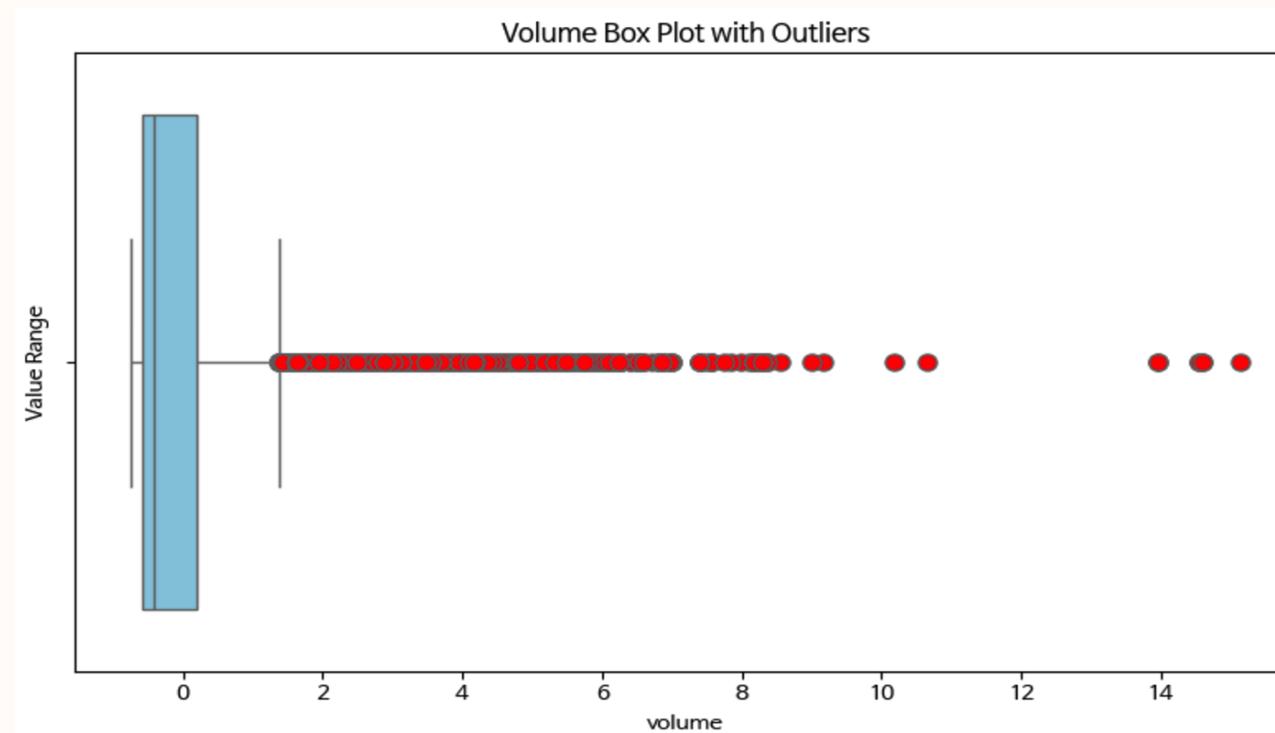
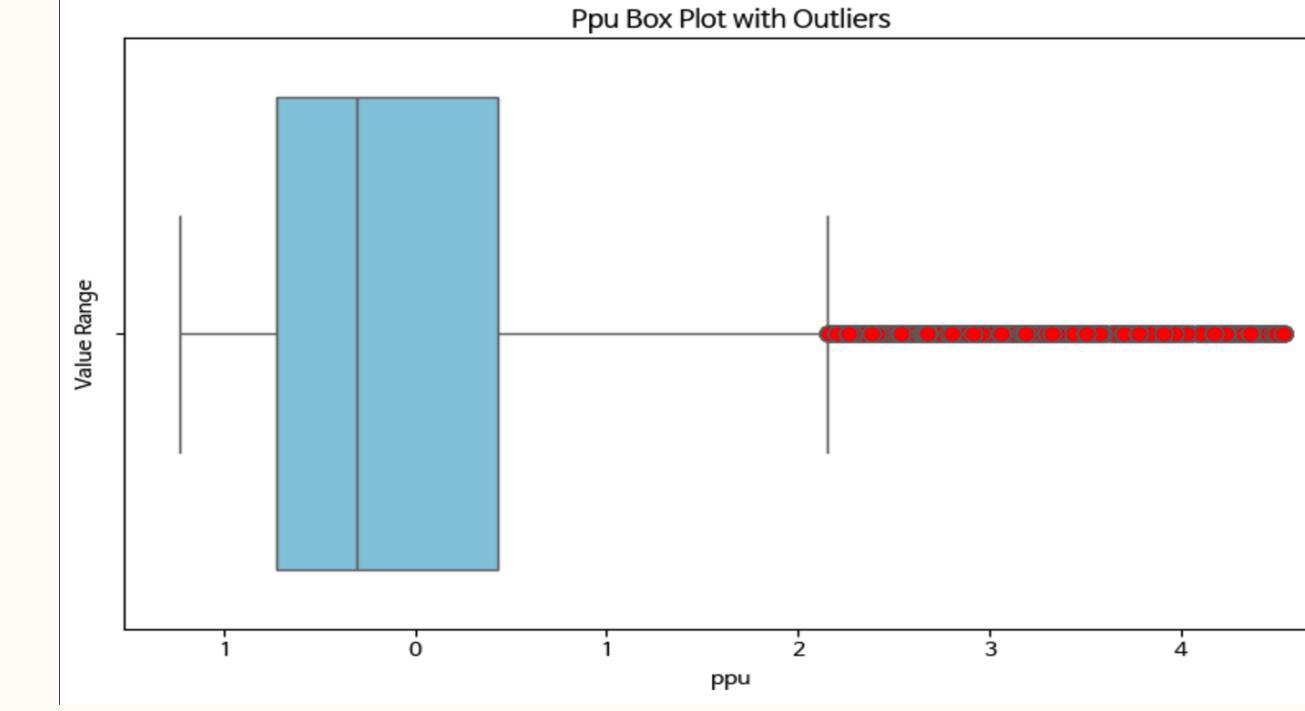
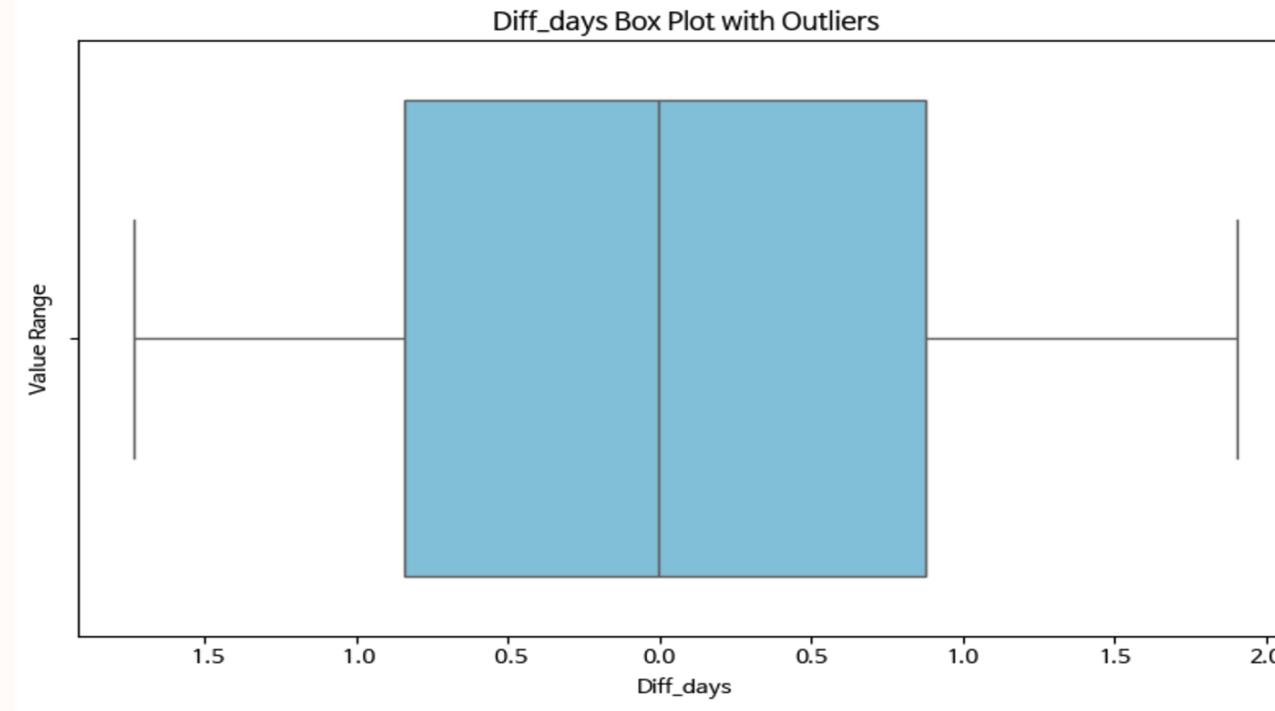
- product_weight_g ↔ volume 0.74

군집분석 진행 시 다중 공선성 고려

클러스터링



3. 클러스터링 - 이상치 확인



3. 클러스터링 - 스케일링 및 Z-Score

Standard scale

```
# StandardScaler 초기화 및 스케일링
scaler = StandardScaler()
scaled_data = scaler.fit_transform(merged_result)

# 스케일링 결과를 DataFrame으로 변환
scaled_df = pd.DataFrame(scaled_data, columns=merged_result.columns, index=merged_result.index)

print('feature들의 평균 값')
print(scaled_df.mean())
print('feature들의 분산 값')
print(scaled_df.var())
```

Z - Score

```
# check Z score
df_Zscore = pd.DataFrame()
outlier_dict = {}
outlier_idx_list = []

for one_col in scaled_df.columns:
    print("Check", one_col)
    df_Zscore[f'{one_col}_Zscore'] = scipy.stats.zscore(scaled_df[one_col])
    outlier_dict[one_col] = df_Zscore[f'{one_col}_Zscore'][(df_Zscore[f'{one_col}_Zscore']>2) | (df_Zscore[f'{one_col}_Zscore']<-2)]
    outlier_idx_list.append(list(outlier_dict[one_col].index))
    if len(outlier_dict[one_col]):
        print(one_col, 'Has outliers\n', outlier_dict[one_col])
    else:
        print(one_col, "Has Not outlier")
    print()
```

3. 클러스터링 - PCA

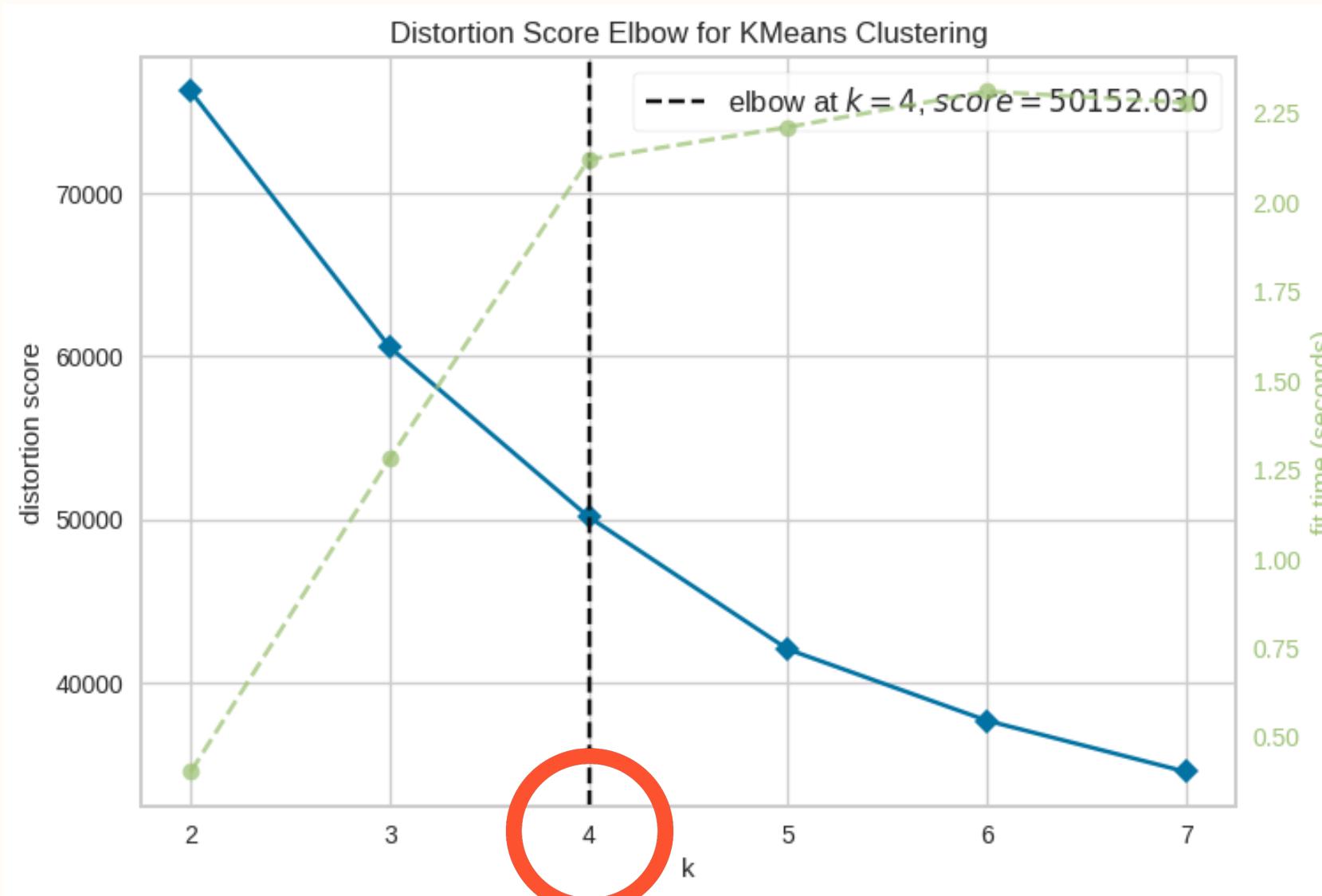
PCA

```
[121] from sklearn.decomposition import PCA  
  
# 주성분 개수를 판단하기 위한 pca임의 시행  
pca = PCA(n_components=2)  
pca.fit(preprocessed_df)  
  
# 설정한 주성분의 갯수로 전체 데이터 분산을 얼만큼 설명 가능한지  
pca.explained_variance_ratio_.sum()  
  
→ 0.5577934055772529
```

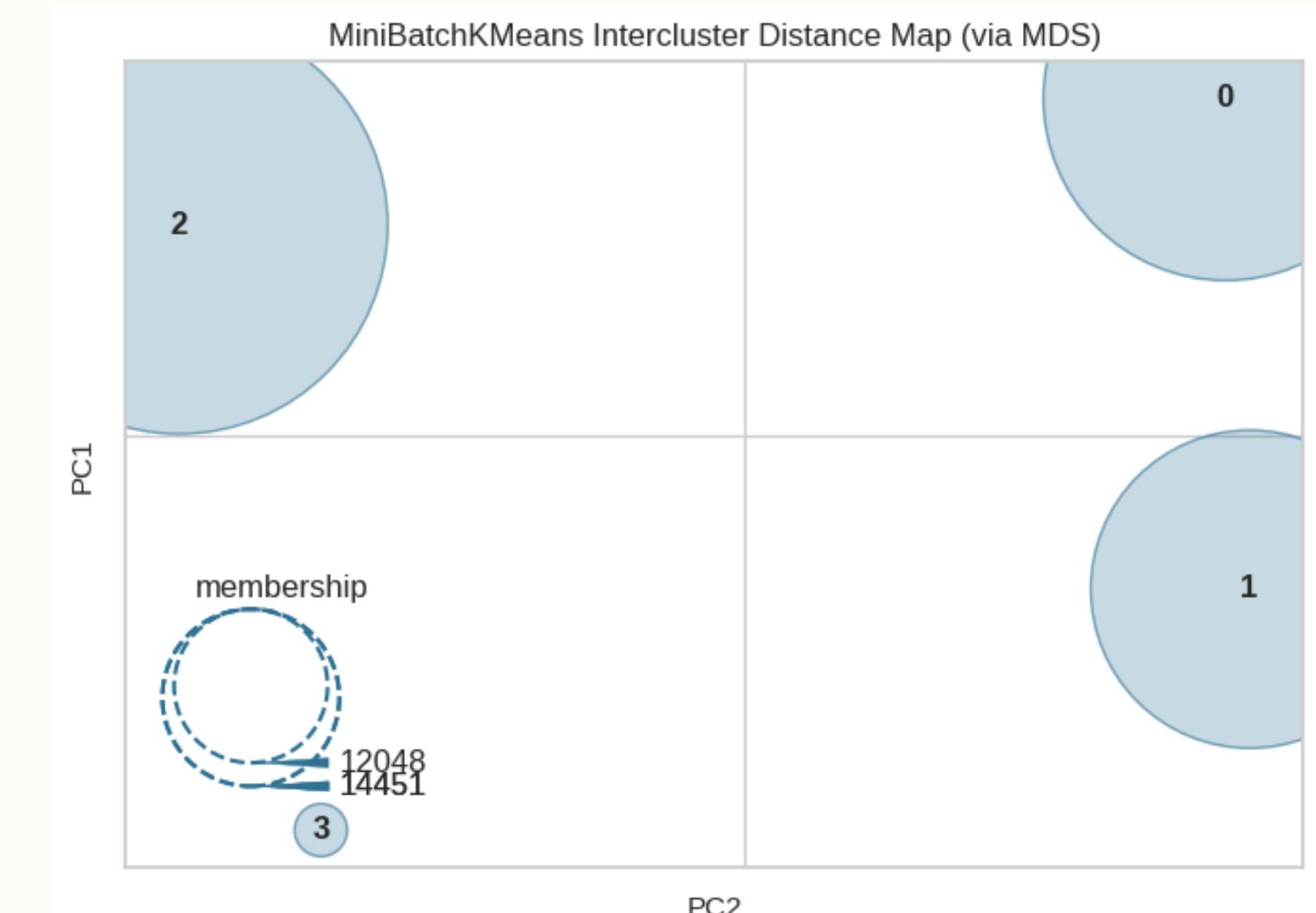
(1) PCA component = 2
→ 설명력 약 55%

3. 클러스터링 - 최적의 K값 탐색

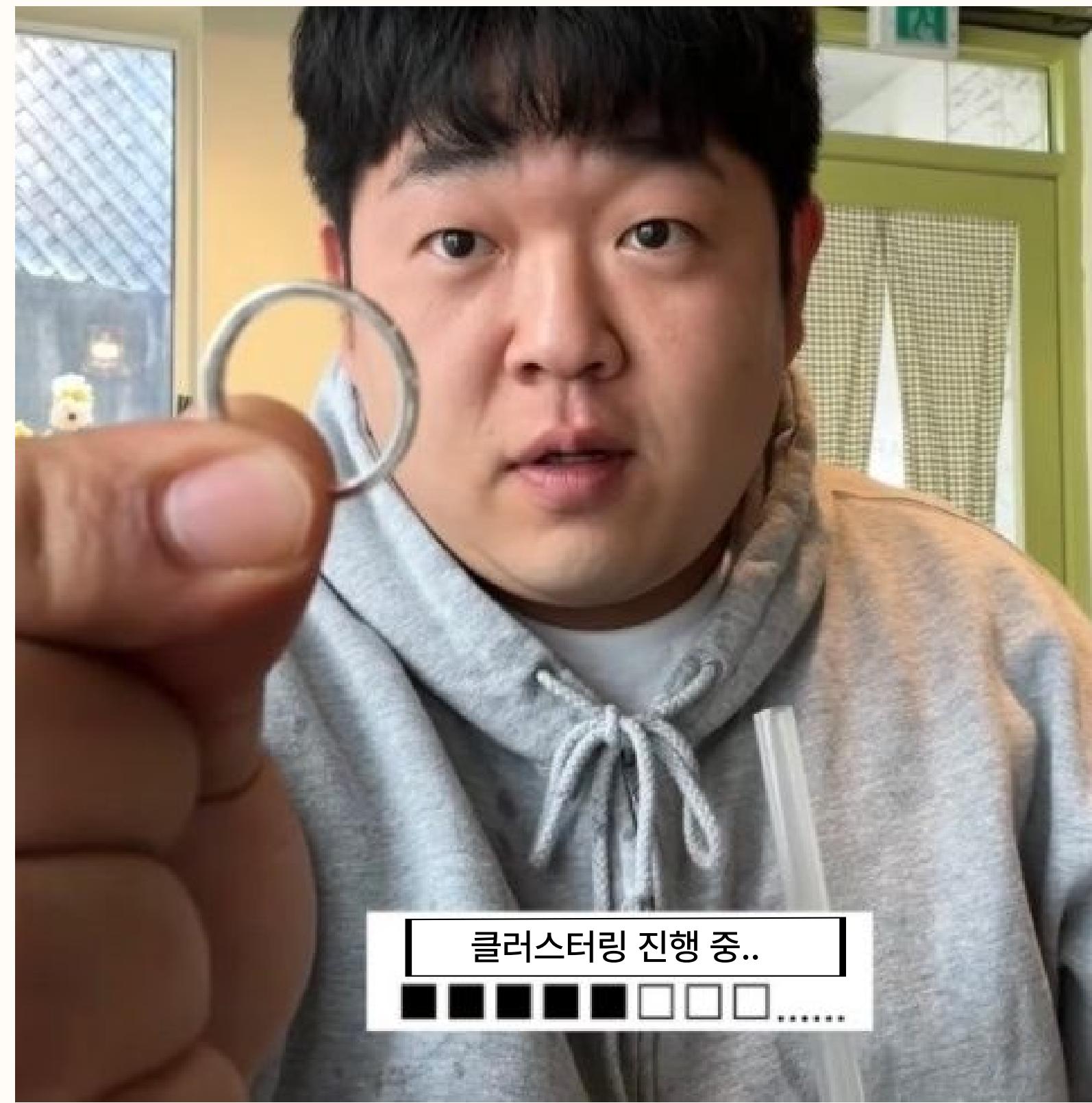
elbow 기법



Distance Map

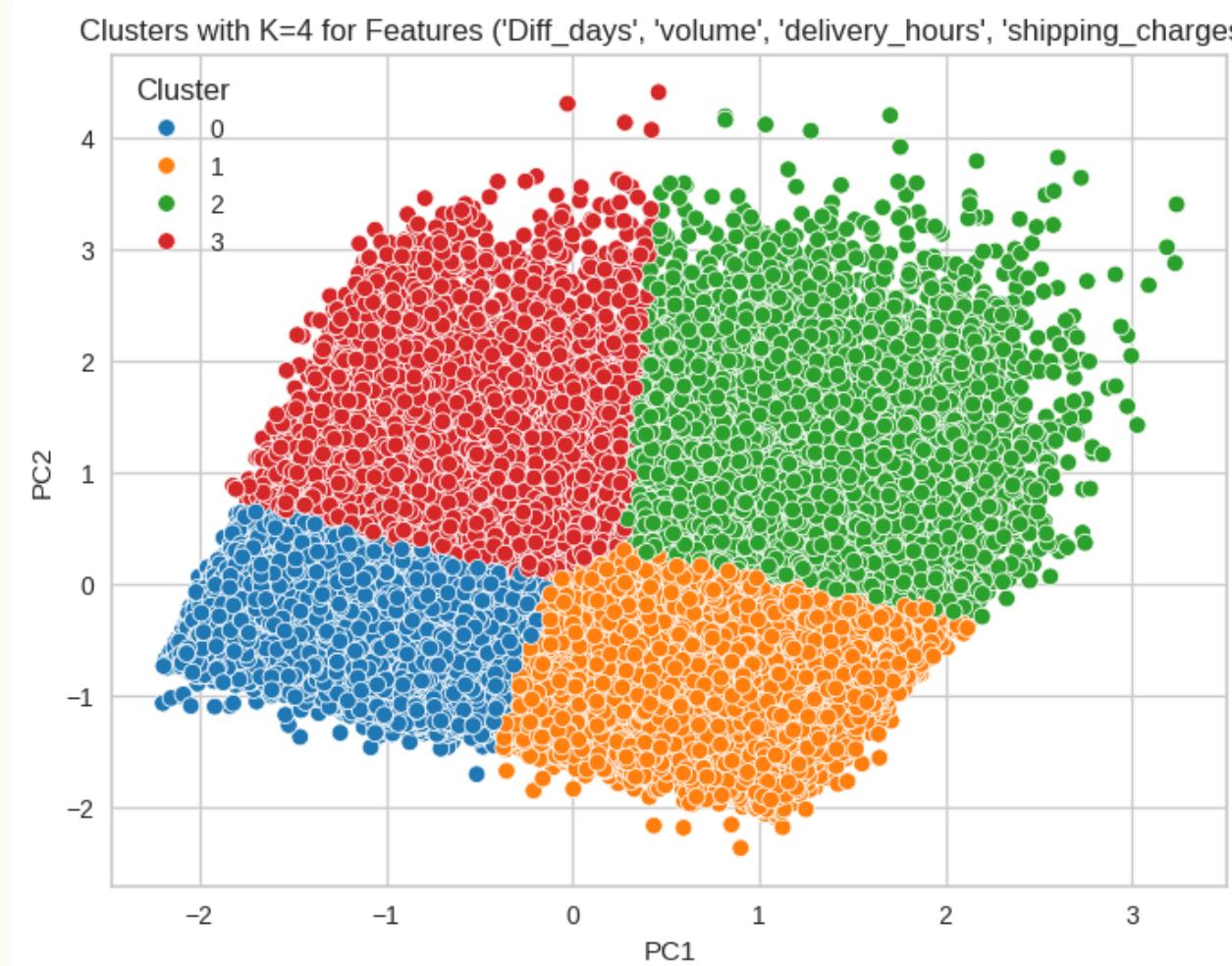


클러스터링 시각화 결과와 실루엣 계수 및 다른 측정 지표를 비교해 더 나은 클러스터링 선택

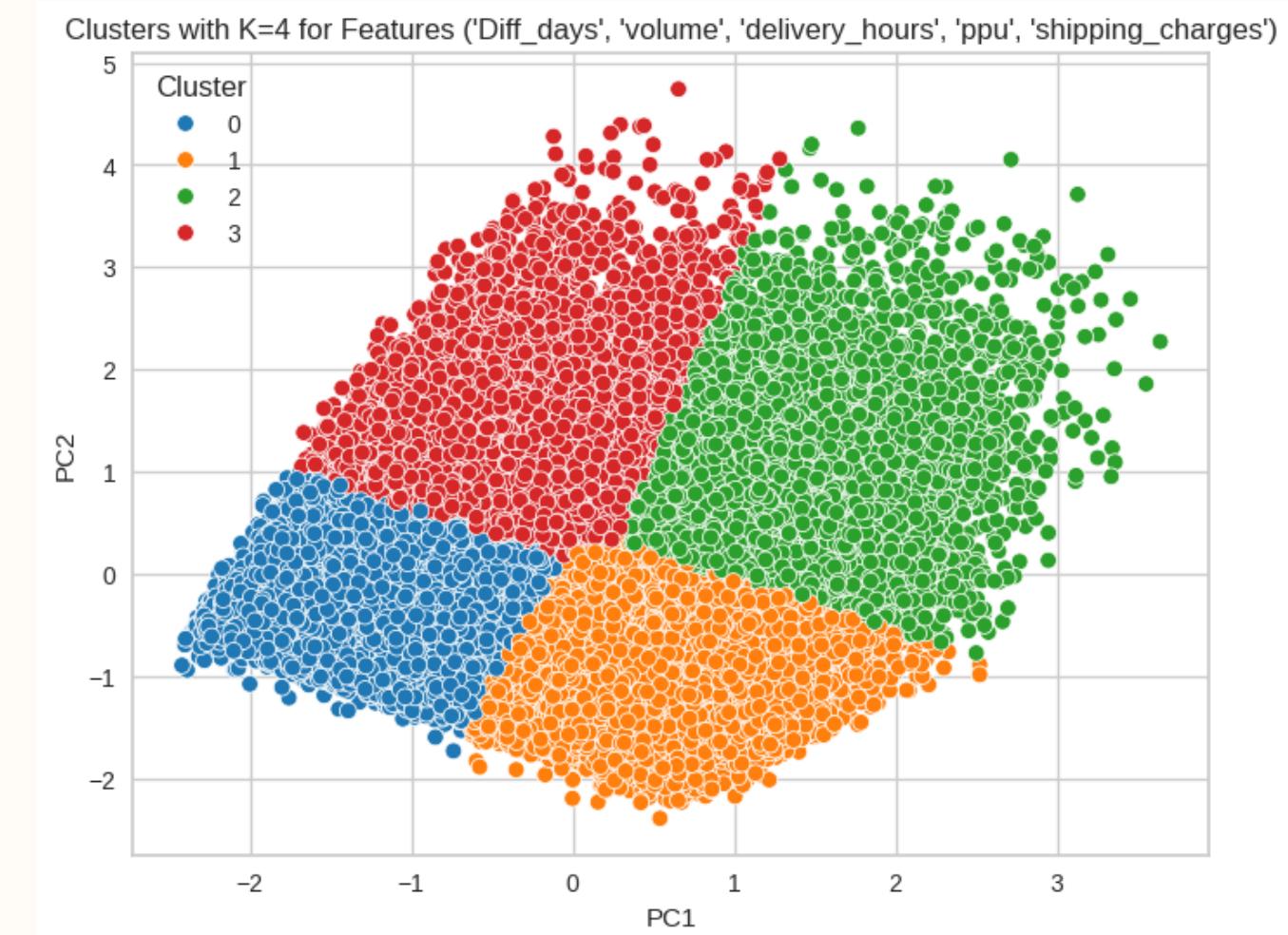


3. 클러스터링 - 클러스터링 결과 비교

Version 1



Version 2



큰 차이 없음

군집 비중

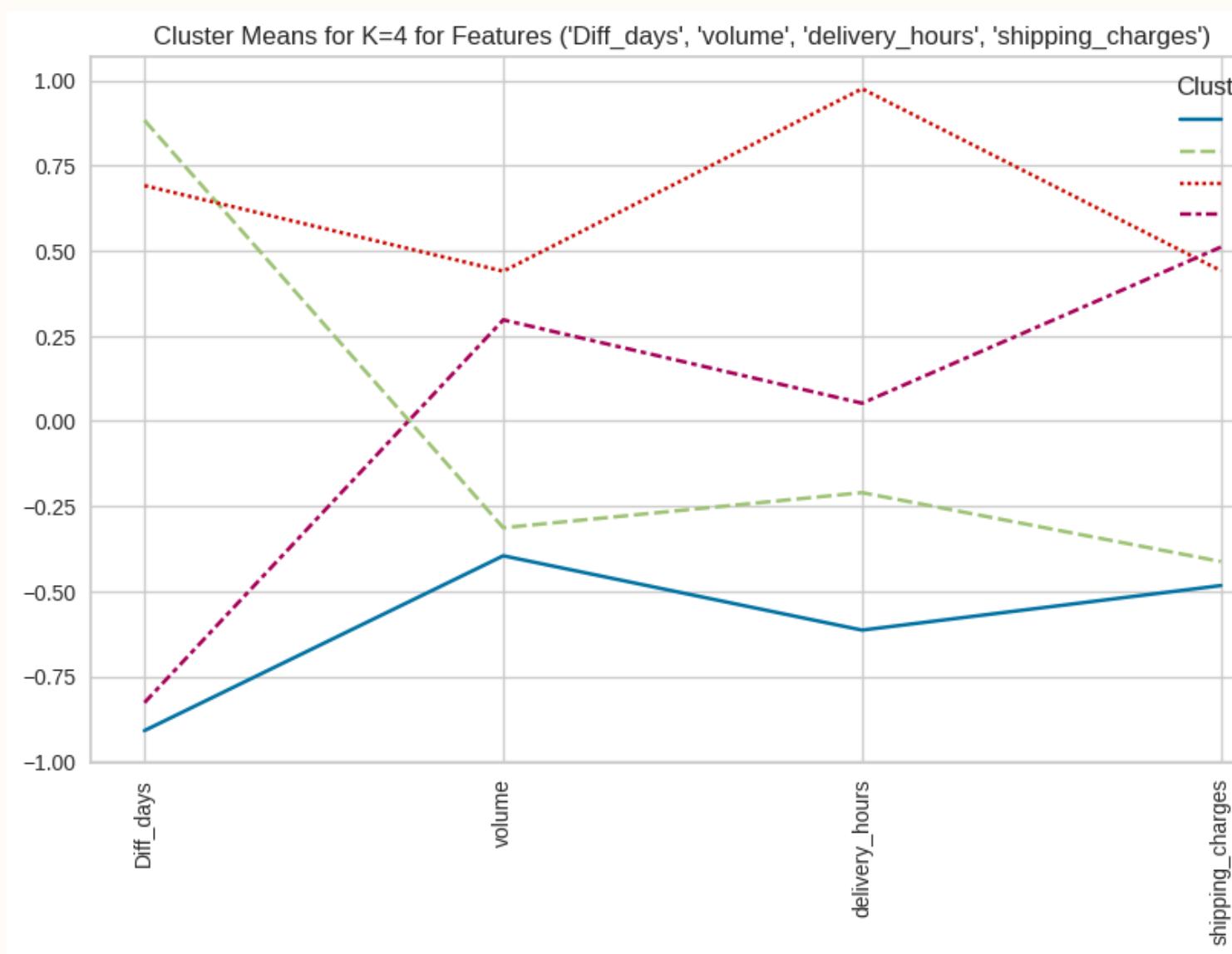
Cluster 0 : 30.95% Cluster 1 : 35.00%
Cluster 2 : 16.83% Cluster 3 : 17.22%

군집별 비중

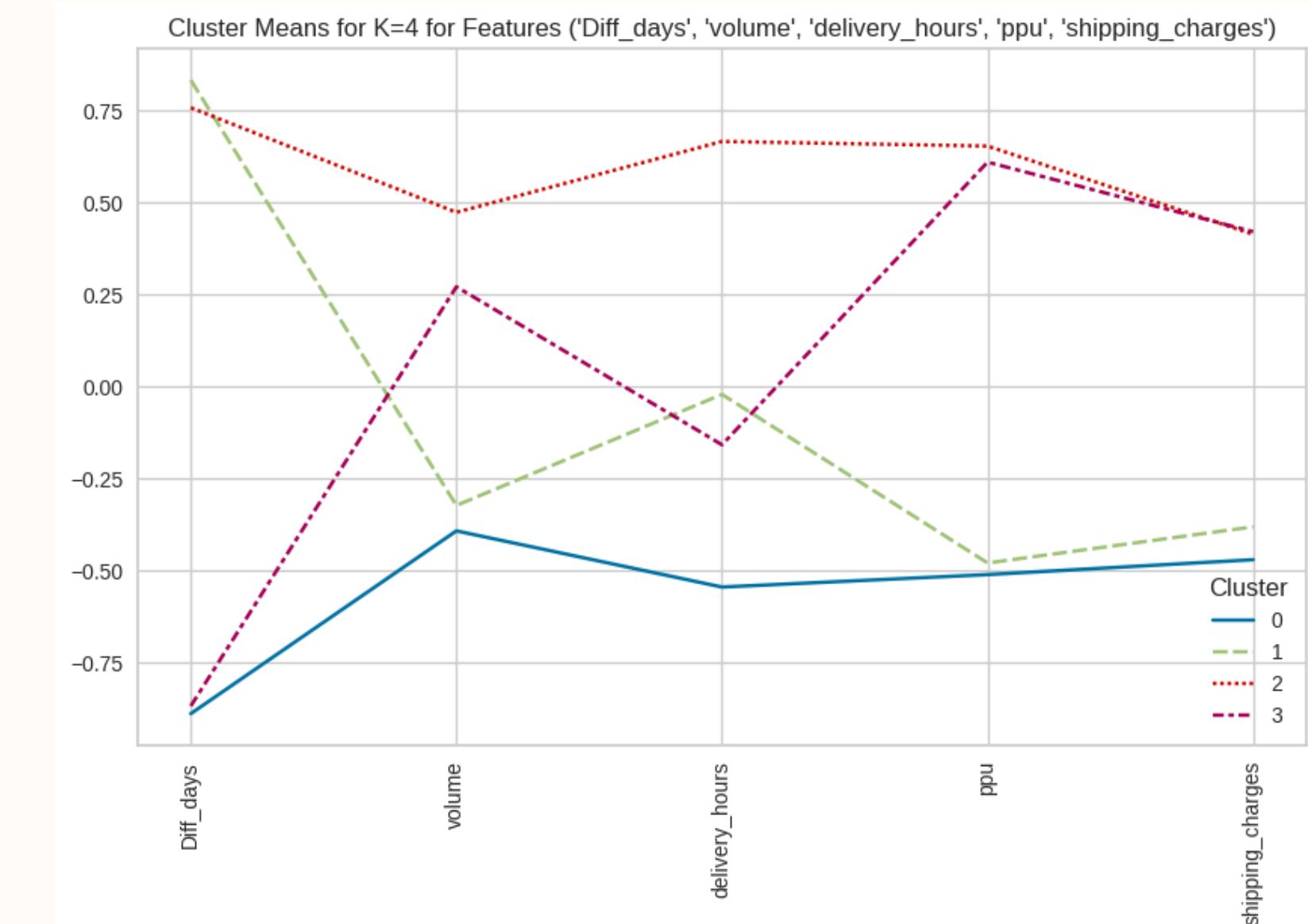
Cluster 0 : 30.66% Cluster 1 : 35.69%
Cluster 2 : 16.63% Cluster 3 : 17.01%

3. 클러스터링 - 클러스터링 결과 비교

Version 1

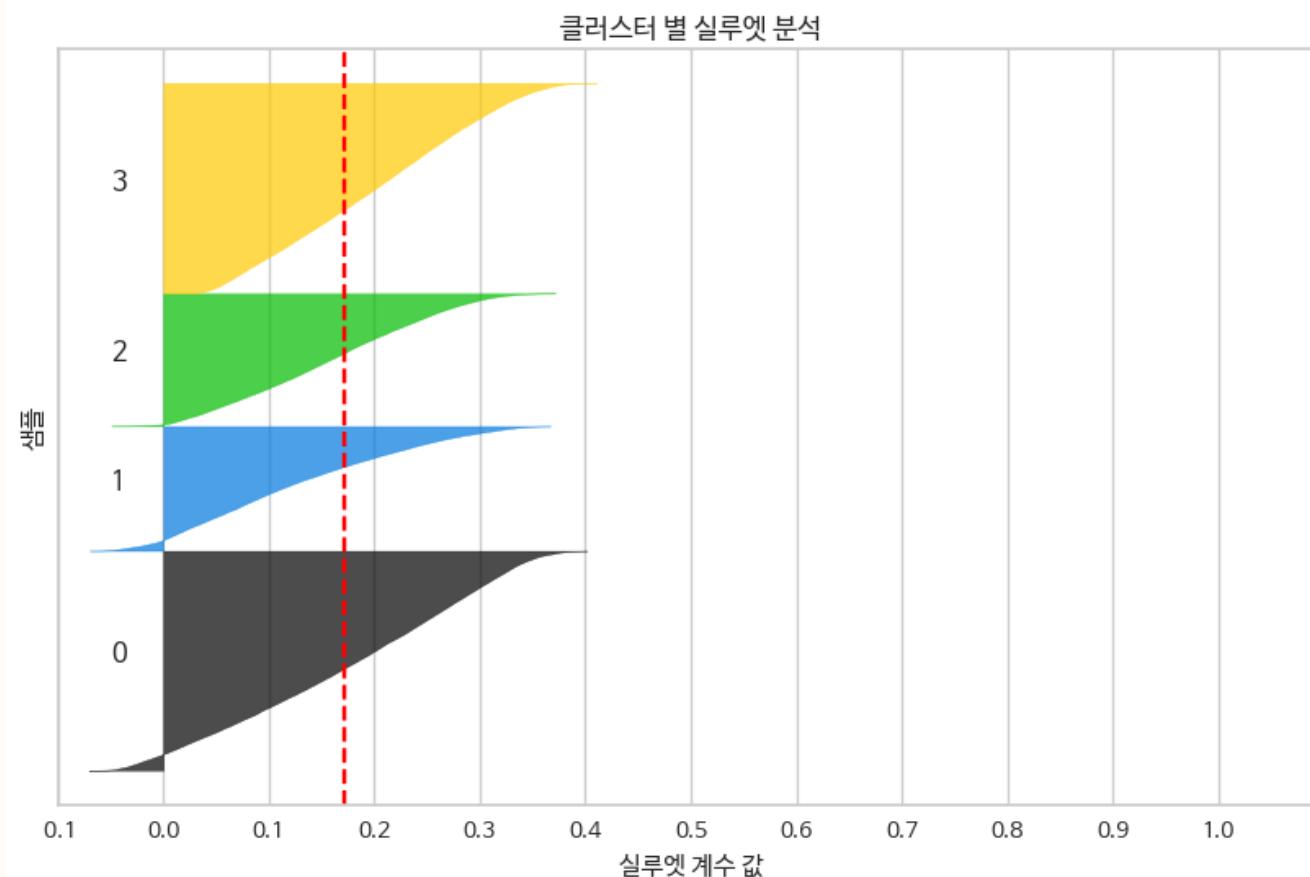


Version 2

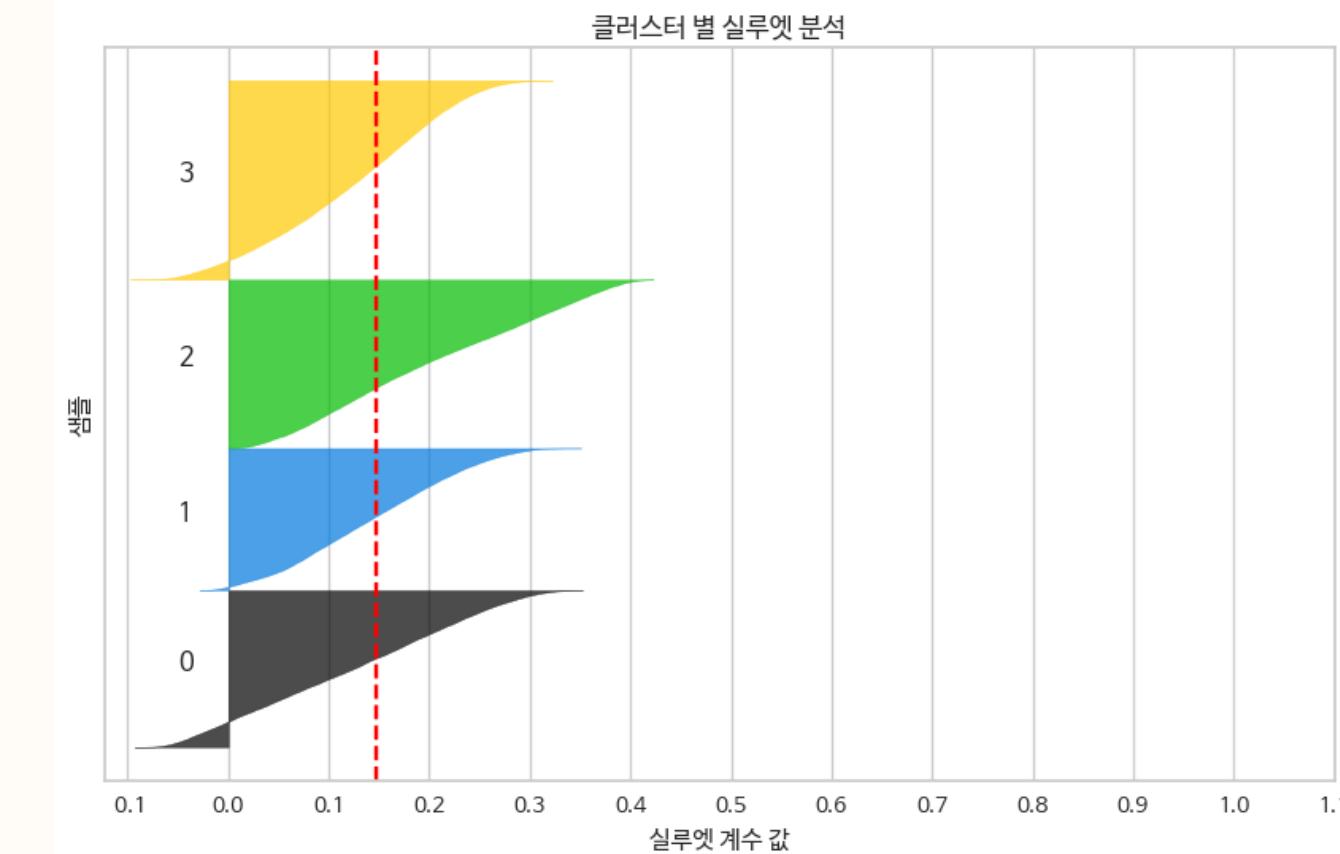


3. 클러스터링 - 클러스터링 결과 비교

Version 1



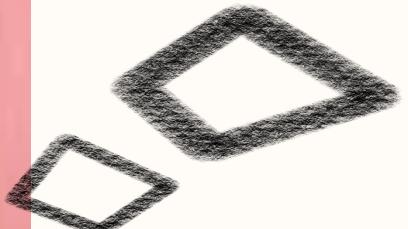
Version 2



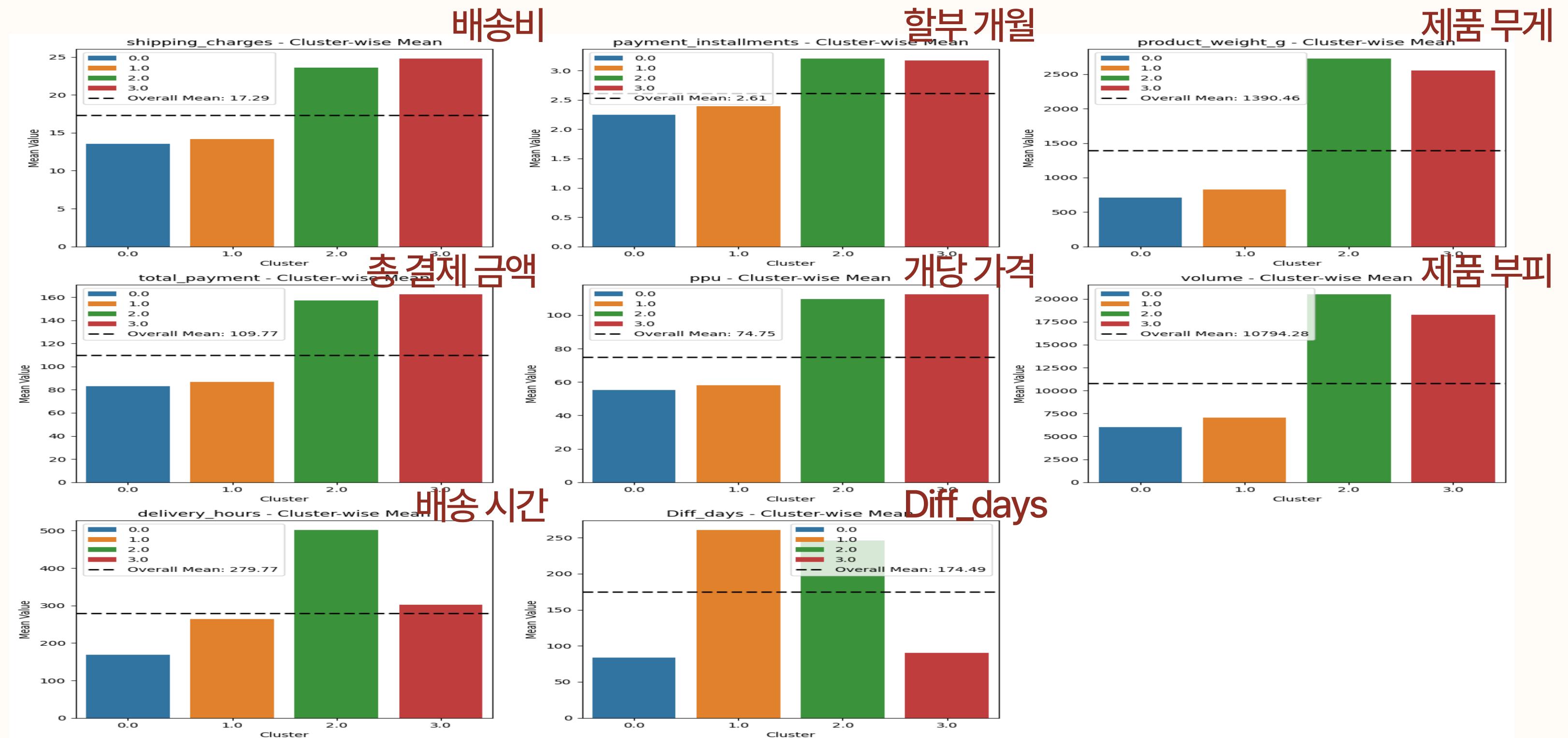
Metric	Value
Number of Observations	65127
Silhouette Score	0.3853691774562848
Calinski Harabasz Score	52430.3892351915
Davies Bouldin Score	0.8644005868311542

Metric	Value
Number of Observations	65127
Silhouette Score	0.3853691774562848
Calinski Harabasz Score	52430.3892351915
Davies Bouldin Score	0.8644005868311542

클러스터 EDA



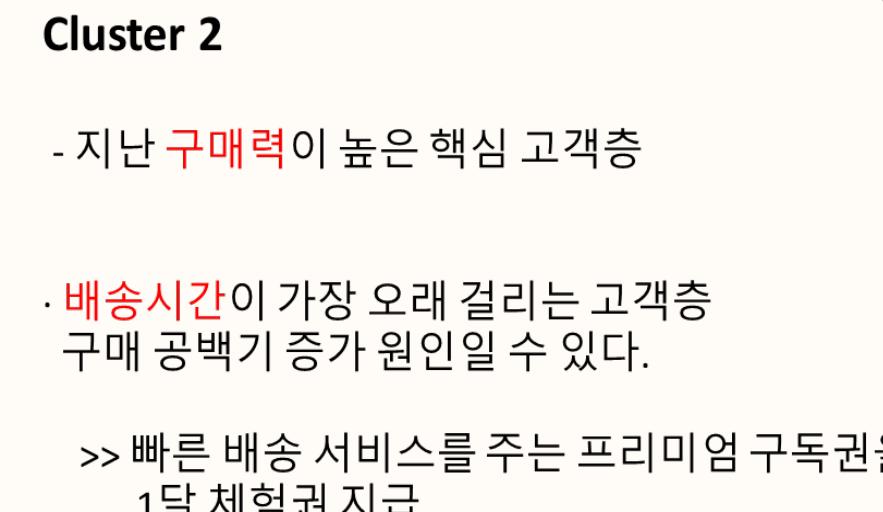
4. 클러스터 EDA - 클러스터별 컬럼별 Mean 비교



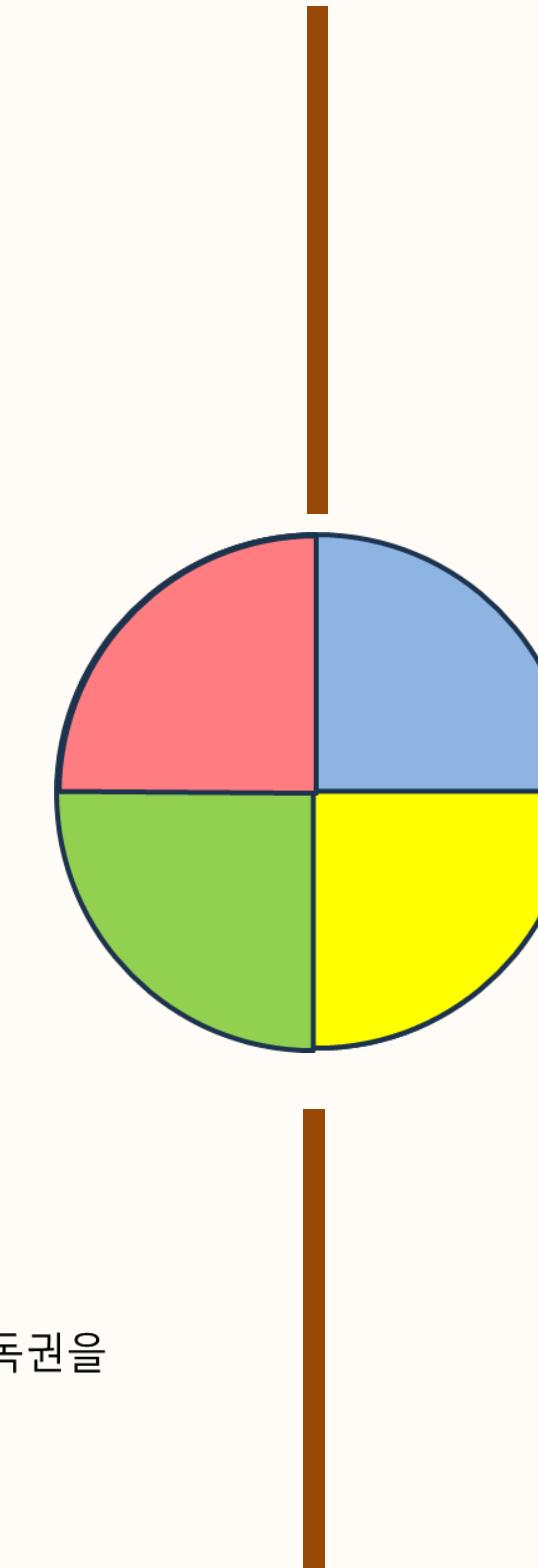
4. 클러스터 EDA - 클러스터별 분석 결과



- 최근 **구매력**이 높은 핵심 고객층
 - . 가장 **비싼 배송비**를 지불하는 고객층
- 배송비 할인 쿠폰 지급을 통한 부담감 완화

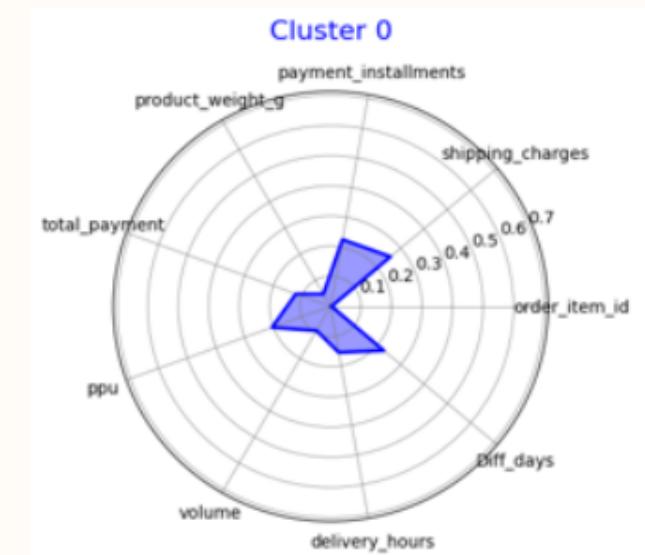


- 지난 **구매력**이 높은 핵심 고객층
 - . **배송시간**이 가장 오래 걸리는 고객층
구매 공백기 증가 원인일 수 있다.
- >> 빠른 배송 서비스를 주는 프리미엄 구독권을
1달 체험권 지급



Cluster 0

- 최근 유입된 잠재 고객층
 - . 모든 지표에서 가장 **소극적**인 고객층으로
다양한 이벤트로 구매 적극성 향상
1. 번들 판매 전략
 2. 무료 배송 서비스



Cluster 1

- 낮은 구매력을 가진 이탈 위험 고객층
 - . 구매 **공백 기간**이 늘어나고 있다.
컴백 프로모션에서 다양한 시도가 필요해 보임
1. 컴백 할인 쿠폰
 2. 무료 배송 서비스



인사이트 및 마케팅 전략 제안



5. 인사이트 및 마케팅 전략

Cluster 0

번들 판매 전략



Cluster 1

재방문율 향상
신뢰구축 및 고객 경험 개선



Cluster 2

배송 추적 서비스 강화
우선 배송 서비스
배송비 절감

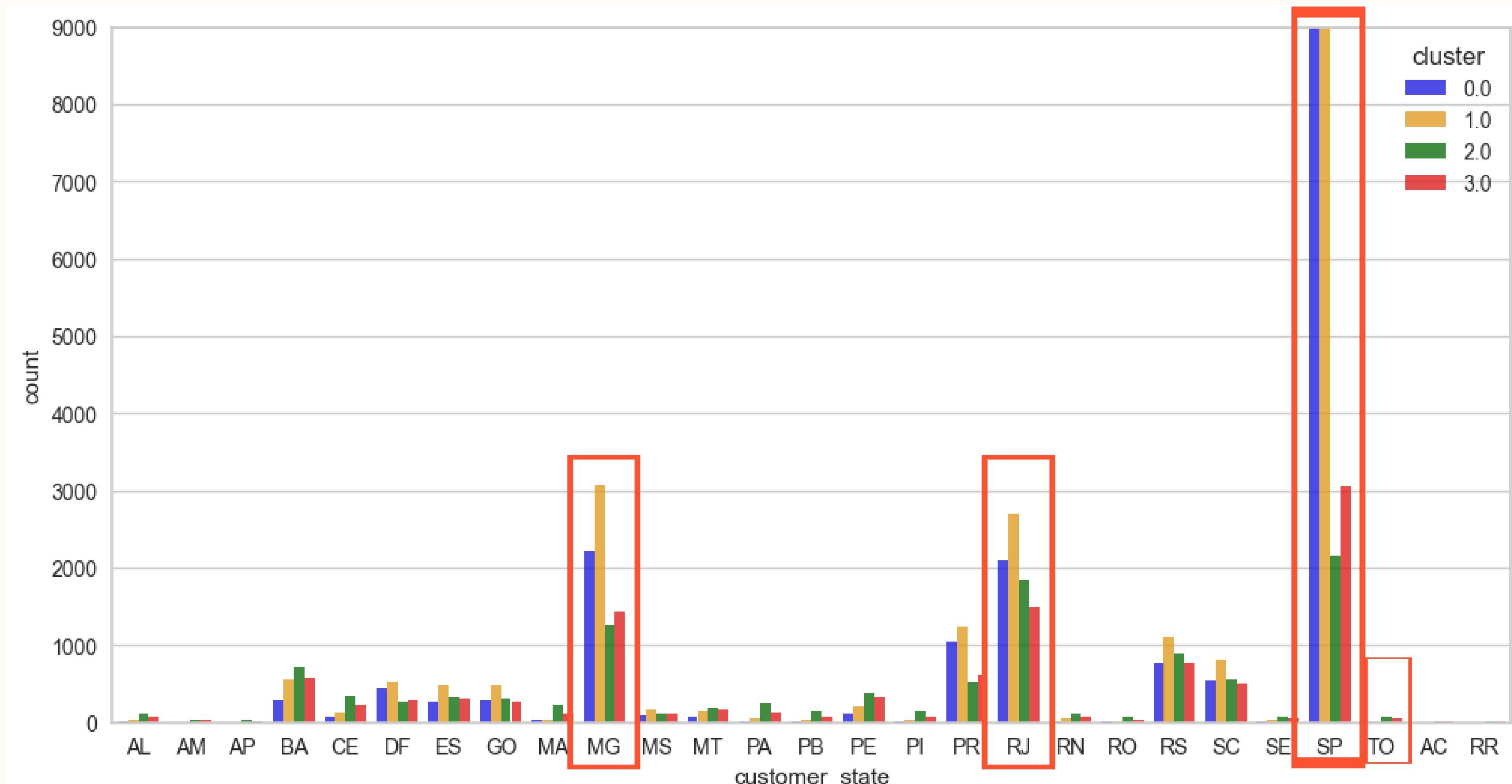


Cluster 3

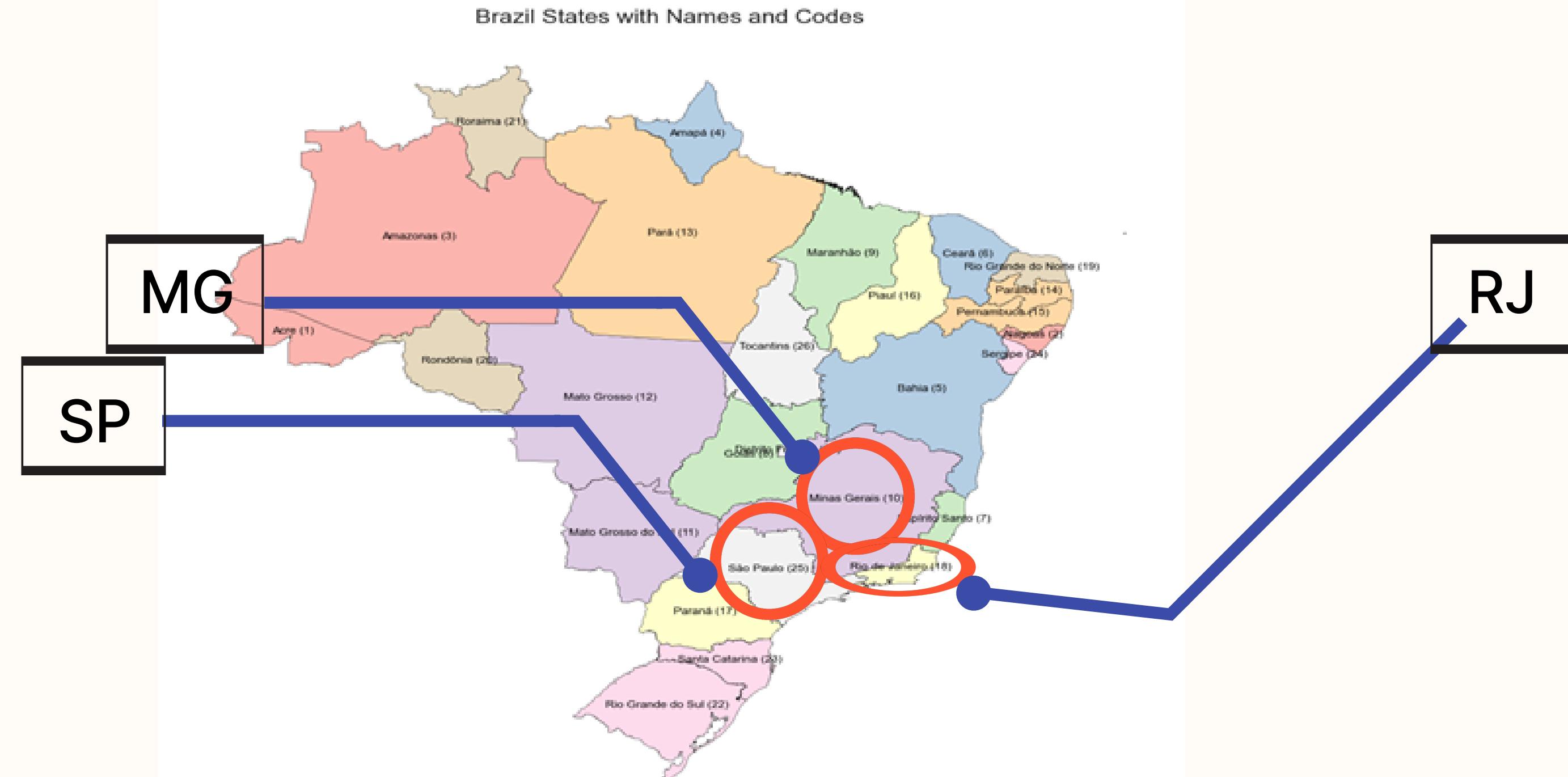
VIP 무료 배송
월간 구독 서비스

주요 이커머스 멤버십 비교			
위메프	NAVER	coupon	eBay
무료 VIP 멤버십	클라우드 멤버십	관련상품 결합	무료 결합
기타 혜택	점수 적립 전 세대폰 10% 할인 혜택	할인 혜택	할인 혜택
주요 혜택	- 제품 행사 참여 혜택 - 카드 결제 혜택 - 쿠폰 혜택	- 새마리아 충전 시 최대 1% 적립 - 카카오 카드 혜택 - 쿠폰 혜택	- 회원 혜택 - 카카오 카드 혜택 - 쿠폰 혜택 - 할인 혜택 - 새마리아 충전 혜택
관련 혜택	기타 혜택 무료 혜택	기타 혜택 혜택 혜택	기타 혜택 혜택 혜택

5. 인사이트 및 마케팅 전략



5. 인사이트 및 마케팅 전략





끝나고 하늘로 승천

발표 들어주셔서 감사합니다 ><