# 추천시스템과 OpenCV 영상 합성 기반 고객 맞춤 ppl

강혜연 윤지의 이혜원

# CONTENTS



PART 1

# 주제 및 선정 이유

PART 1

# 주제 및 선정 이유

## “ 추천시스템과 OpenCV 영상 합성 기반 고객 맞춤 ppl ”

영화나 드라마 등과 같은 영상물은 오랜 시간, 다수의 사람들로부터
반복 시청이 이루어지지만, **한번 연출된 PPL은 바꾸지 못해 시의적절한 광고를 하지 못한다**는 문제점을 바탕으로 주제를 선정하게 됨

# 주제 및 선정 이유

## 해결 방법

- 추천시스템으로 시청자 취향에 맞는 아이템 선택
- 영상 합성 기술로 영상에 광고 콘텐츠 합성

▲ 원본 영상


▲ 추천시스템

▲ 합성 영상

# 주제 및 선정 이유

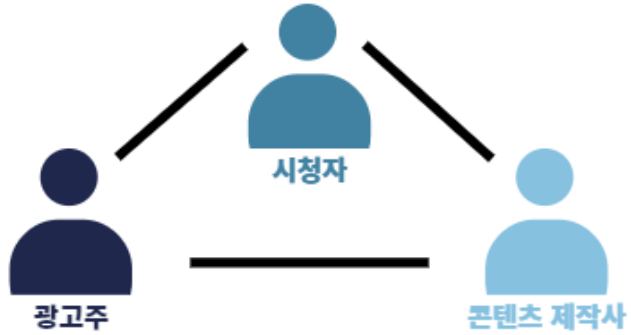
## 기대 효과

같은 장면이라도 시청자 맞춤으로 간접광고 가능
ex) 성인이면 맥주 광고, 청소년이면 탄산수 광고

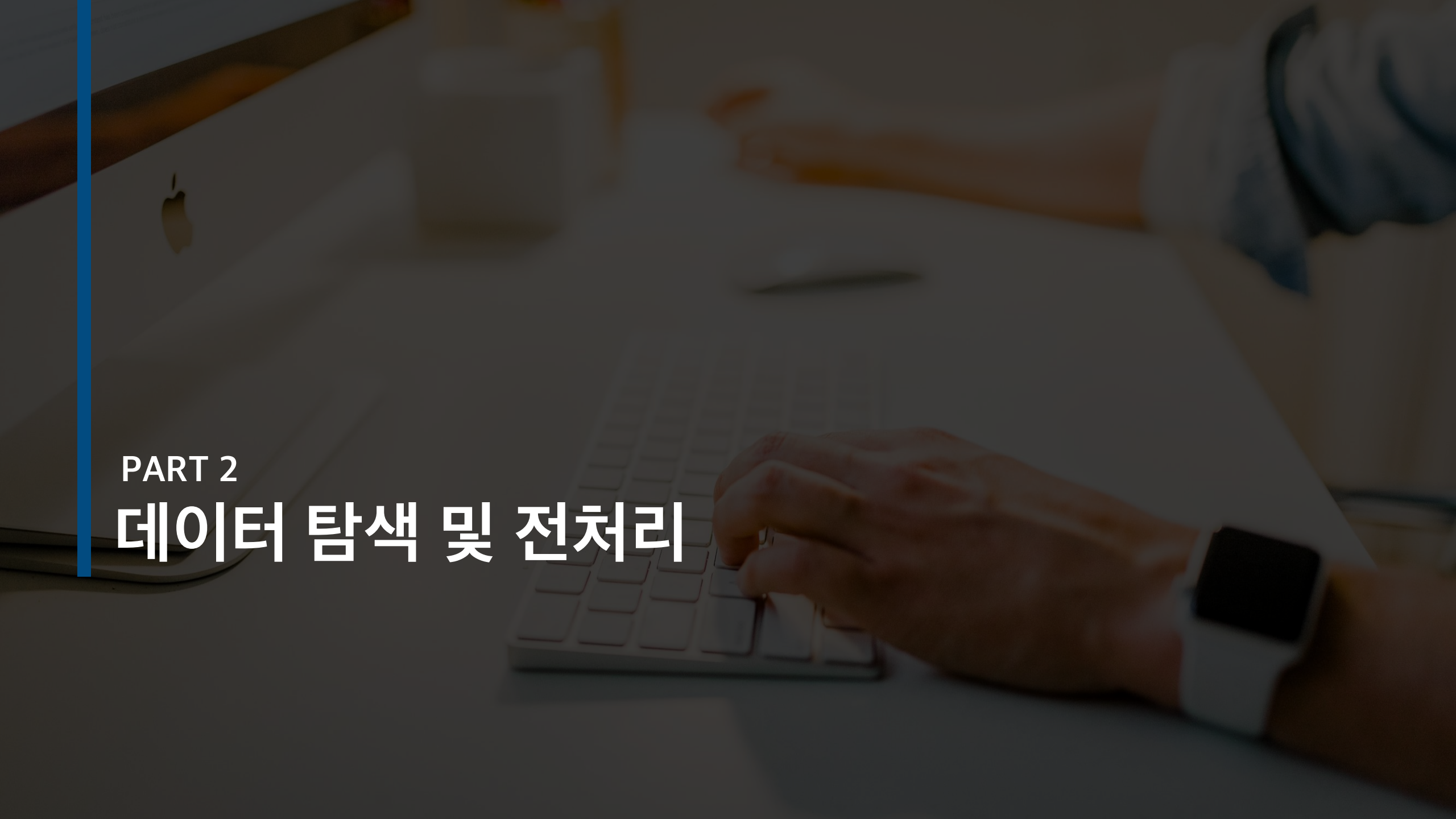
Version 1.

Version 2.


광고주 , 시청자 ,콘텐츠 제작사 모두 만족




- 광고주: 개인 맞춤 광고로 광고 효과 극대화
- 시청자: 원하는 상품 정보를 얻을 수 있음
- 콘텐츠 제작사: PPL상품을 다양하게 바꿀 수 있어 지속적인 수익 창출 가능



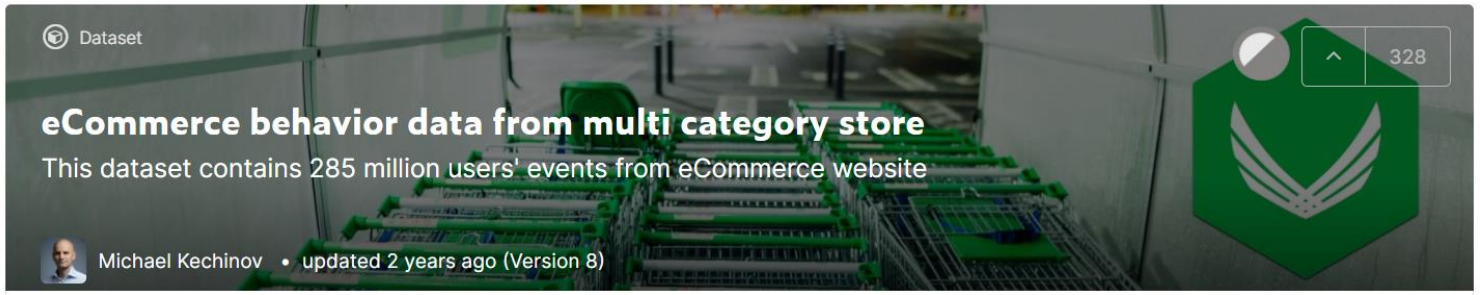
PART 2

# 데이터 탐색 및 전처리

# 데이터 탐색 및 전처리

## 사용 데이터

kaggle ‘eCommerce behavior data from multi category store’ 데이터
(https://www.kaggle.com/mkechinov/ecommerce-behavior-data-from-multi-category-store)




➔ 대형 온라인 상점의 7개월(2019년 10월부터 2020년 4월) 동안의 행동 데이터로 구성되어 있음

PART 2

# 데이터 탐색 및 전처리

```
df=pd.read_csv('./2019-Oct.csv')
df
```

| | event_time | event_type | product_id | category_id | category_code | brand | price | user_id | user_session |
|---|---|---|---|---|---|---|---|---|---|
| 0 | 2019-10-01 00:00:00 UTC | view | 44600062 | 2103807459595387724 | NaN | shiseido | 35.79 | 541312140 | 72d76fde-8bb3-4e00-8c23-a032dfed738c |
| 1 | 2019-10-01 00:00:00 UTC | view | 3900821 | 2053013552326770905 | appliances.environment.water_heater | aqua | 33.20 | 554748717 | 9333dfbd-b87a-4708-9857-6336556b0fcc |
| 2 | 2019-10-01 00:00:01 UTC | view | 17200506 | 2053013559792632471 | furniture.living_room.sofa | NaN | 543.10 | 519107250 | 566511c2-e2e3-422b-b695-cf8e6e792ca8 |
| 3 | 2019-10-01 00:00:01 UTC | view | 1307067 | 2053013558920217191 | computers.notebook | lenovo | 251.74 | 550050854 | 7c90fc70-0e80-4590-96f3-13c02c18c713 |
| 4 | 2019-10-01 00:00:04 UTC | view | 1004237 | 2053013555631882655 | electronics.smartphone | apple | 1081.98 | 535871217 | c6bd7419-2748-4c56-95b4-8cec9ff8b80d |
| ... | ... | ... | ... | ... | ... | ... | ... | ... | ... |
| 42448759 | 2019-10-31 23:59:58 UTC | view | 2300275 | 2053013560530830019 | electronics.camera.video | gopro | 527.40 | 537931532 | 22c57267-da98-4f28-9a9c-18bb5b385193 |
| 42448760 | 2019-10-31 23:59:58 UTC | view | 10800172 | 2053013554994348409 | NaN | redmond | 61.75 | 527322328 | 5054190a-46cb-4211-a8f1-16fc1a060ed8 |
| 42448761 | 2019-10-31 23:59:58 UTC | view | 5701038 | 2053013553970938175 | auto.accessories.player | kenwood | 128.70 | 566280422 | 05b6c62b-992f-4e8e-91f7-961bcb4719cd |
| 42448762 | 2019-10-31 23:59:59 UTC | view | 21407424 | 2053013561579406073 | electronics.clocks | tissot | 689.85 | 513118352 | 4c14bf2a-2820-4504-929d-046356a5a204 |
| 42448763 | 2019-10-31 23:59:59 UTC | view | 13300120 | 2053013557166998015 | NaN | swisshome | 155.73 | 525266378 | 6e57d2d7-6022-46e6-81d6-fa77f14cefd8 |

42448764 rows × 9 columns

| | |
|---|---|
| event_time | 이벤트 발생 시간 |
| event_type | 사용자 행동 패턴<br>(view / cart / purchase) |
| product_id | 제품 id |
| category_id | 제품 카테고리 id |
| category_code | 제품 분류<br>‘.’을 기준으로 대분류, 중분류, 소분류 |
| brand | 제품 브랜드 이름 |
| price | 제품 가격 |
| user_id | 사용자 id |
| user_session | 사용자 세션 id |

PART 2

# 데이터 탐색 및 전처리

원본 데이터
(42448764 × 9 )

Null값 제거
(26560620 × 9 )

4일치 데이터 추출
(3183153 × 9 )

user_id 10,000개 추출
(70731 × 9 )

| | event_time | event_type | product_id | category_id | category_code | brand | price | user_id | user_session |
|---|---|---|---|---|---|---|---|---|---|
| 0 | 2019-10-01 00:00:23 UTC | view | 6200260 | 2053013552293216471 | appliances.environment.air_heater | midea | 47.62 | 538645907 | 7d9a8784-7b6c-426e-9924-9f688812fd71 |
| 1 | 2019-10-01 00:08:35 UTC | view | 1004767 | 2053013555631882655 | electronics.smartphone | samsung | 254.82 | 555448677 | 3886fcc8-3618-4bb0-a9f4-614bfb2d5263 |
| 2 | 2019-10-01 00:11:37 UTC | view | 1004767 | 2053013555631882655 | electronics.smartphone | samsung | 254.82 | 555448864 | 32f9d470-c71e-46cf-bd3d-0afd1cd961ed |
| 3 | 2019-10-01 00:14:25 UTC | view | 1004870 | 2053013555631882655 | electronics.smartphone | samsung | 286.86 | 555448864 | 32f9d470-c71e-46cf-bd3d-0afd1cd961ed |
| 4 | 2019-10-01 00:14:37 UTC | view | 1004870 | 2053013555631882655 | electronics.smartphone | samsung | 286.86 | 555448864 | 32f9d470-c71e-46cf-bd3d-0afd1cd961ed |
| ... | ... | ... | ... | ... | ... | ... | ... | ... | ... |
| 70726 | 2019-10-04 23:57:09 UTC | view | 45600336 | 2116907524572577889 | apparel.shoes | garvalin | 69.50 | 556879445 | f33a90d8-ffd1-4174-8fa1-dff866e00657 |
| 70727 | 2019-10-04 23:57:12 UTC | view | 1201465 | 2172371436436455782 | electronics.tablet | samsung | 238.56 | 534887795 | 38022291-a33e-4b8b-a532-70e4e90f0b4b |
| 70728 | 2019-10-04 23:58:27 UTC | view | 1200617 | 2172371436436455782 | electronics.tablet | samsung | 128.42 | 534887795 | 38022291-a33e-4b8b-a532-70e4e90f0b4b |
| 70729 | 2019-10-04 23:59:22 UTC | view | 1200617 | 2172371436436455782 | electronics.tablet | samsung | 128.42 | 534887795 | 38022291-a33e-4b8b-a532-70e4e90f0b4b |
| 70730 | 2019-10-04 23:59:45 UTC | view | 1200620 | 2172371436436455782 | electronics.tablet | samsung | 128.42 | 534887795 | 38022291-a33e-4b8b-a532-70e4e90f0b4b |

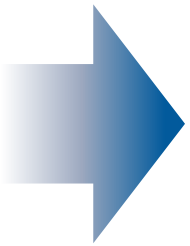
70731 rows × 9 columns

# 데이터 탐색 및 전처리

```
df['user_score'] = df['event_type'].map({'view':1,'cart':10,'purchase':50})
df['user_purchase'] = df['event_type'].apply(lambda x: 1 if x=='purchase' else 0)
```

```
df['price_category'] = 0
df=df.astype({'price_category':'object'})
for i in df['category_code'].unique():
    df.loc[df['category_code']==i,'price_category'] = pd.qcut(x=df['price'][df['category_code']==i],q=5,labels=False,duplicates='drop')
df.dropna(subset=['price_category'],inplace=True)
df
```

| | event_type | price |
|---|---|---|
| **0** | view | 47.62 |
| **1** | view | 254.82 |
| **2** | view | 254.82 |
| **3** | view | 286.86 |
| **4** | view | 286.86 |



| user_score | user_purchase | price_category |
|---|---|---|
| 1 | 0 | 3 |
| 1 | 0 | 2 |
| 1 | 0 | 2 |
| 1 | 0 | 2 |
| 1 | 0 | 2 |

# 데이터 탐색 및 전처리

```
group = df.groupby(['user_id','product_id'])['user_score','user_purchase'].sum().reset_index()
group['user_purchase'] = group['user_purchase'].apply(lambda x: 1 if x>1 else x)
group['user_score'] = group['user_score'].apply(lambda x: 100 if x>100 else x)

std = MinMaxScaler(feature_range=(0.025, 1))
std.fit(group['user_score'].values.reshape(-1,1))
group['interaction_score'] = std.transform(group['user_score'].values.reshape(-1,1))
group
```

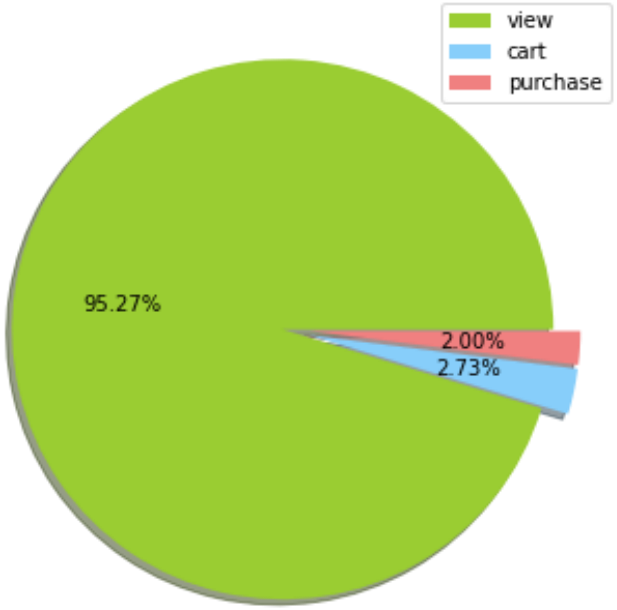
| | user_id | product_id | user_score | user_purchase | interaction_score |
|---|---|---|---|---|---|
| **0** | 241784978 | 1002544 | 1 | 0 | 0.025000 |
| **1** | 336420903 | 21406140 | 2 | 0 | 0.034848 |
| **2** | 336420903 | 21406169 | 2 | 0 | 0.034848 |
| **3** | 336420903 | 21407286 | 3 | 0 | 0.044697 |
| **4** | 373445245 | 3701095 | 2 | 0 | 0.034848 |
| **...** | ... | ... | ... | ... | ... |
| **39818** | 556878415 | 6200514 | 1 | 0 | 0.025000 |
| **39819** | 556878415 | 6200515 | 2 | 0 | 0.034848 |
| **39820** | 556878415 | 6200718 | 2 | 0 | 0.034848 |
| **39821** | 556878415 | 6200724 | 2 | 0 | 0.034848 |
| **39822** | 556879445 | 45600336 | 1 | 0 | 0.025000 |

39823 rows × 5 columns

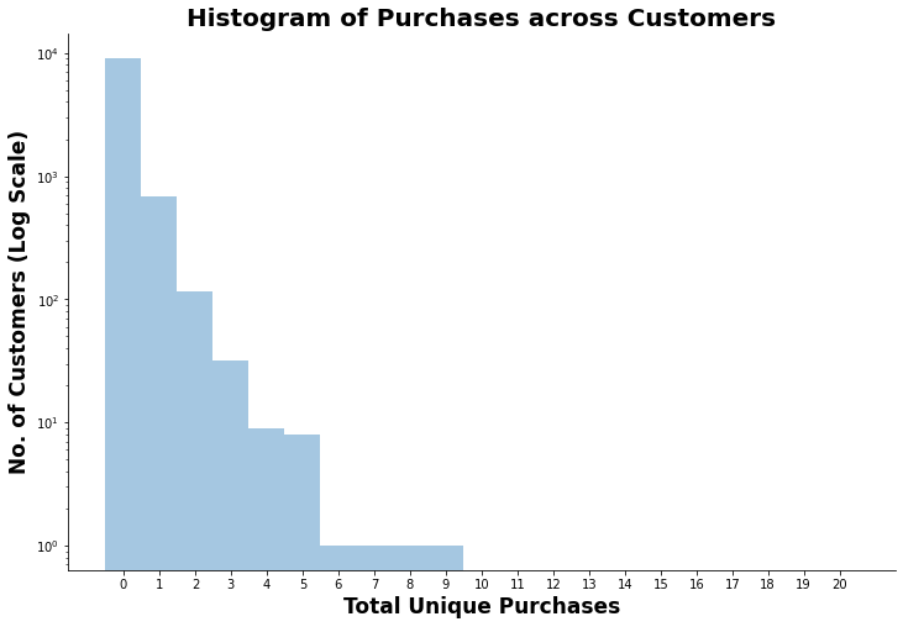
- **groupby()** 함수를 사용해 user_id별 중복되는 product_id에 대해 user_score의 합을 계산해 줌
- user_score에 **MinMaxSacler**를 적용하여 **interaction_score** 변수 생성
- **interaction_score** 가 0.5 이상이면 1번 이상 구매를 했다고 볼 수 있음

# 데이터 탐색 및 전처리

## event_type의 약 95%가 view




## 구매가 0인 고객이 대다수를 차지

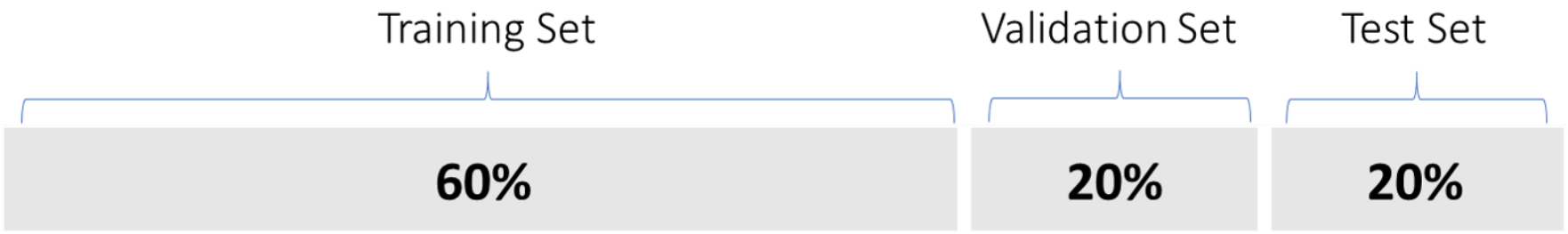



➔ View, Cart, Purchase의 비율이 매우 불균형, 0번 구매 고객이 다수

PART 3

# 추천시스템 구현

# 추천시스템 구현



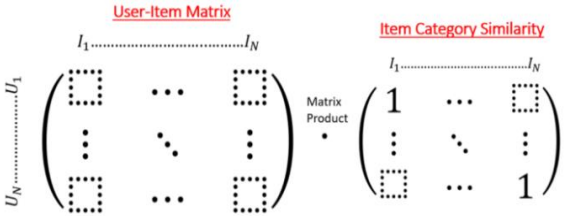

데이터 분리 train : validation : test = 6:2:2

# 추천시스템 구현

## 3가지 종류의 콘텐츠 기반 필터링 사용

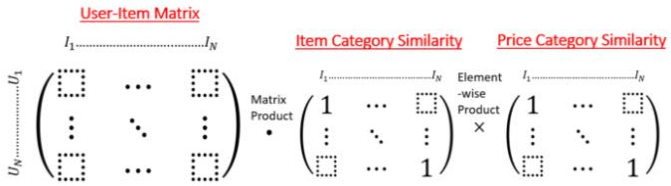
1) filtering by item category

(user-item matrix) X (item category similarity)




2) filtering by item category and price category

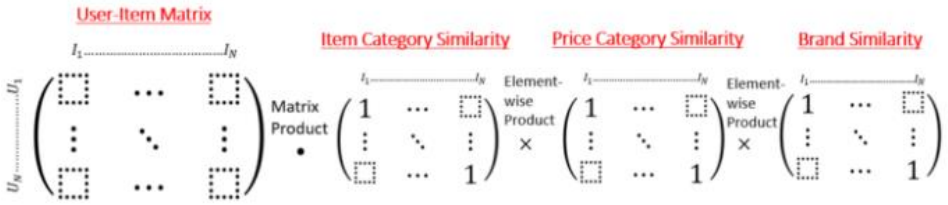
(user-item matrix) X (item category similarity) X (price category similarity)




3) filtering by item category, price category and brand

(user-item matrix) X (item category similarity)

X (price category similarity) X (brand similarity)



# 추천시스템 구현

## Item category similarity

```
tfidf_vectorizer = TfidfVectorizer()
doc_term = tfidf_vectorizer.fit_transform(list(product_cat['category_code']))
dt_matrix = pd.DataFrame(doc_term.toarray().round(3), index=[i for i in product_cat['product_id']],
                         columns=tfidf_vectorizer.get_feature_names())
cos_similar_matrix = pd.DataFrame(cosine_similarity(dt_matrix.values),columns=product_cat['product_id'],
                                  index=product_cat['product_id'])
cos_similar_matrix
```

| product_id | 1001588 | 1002042 | 1002098 | 1002099 | 1002100 | 1002101 | 1002102 | 1002225 | 1002367 | 1002396 | ... | 52900072 | 52900075 | 52900077 | 52900084 | 5290101 | 52900102 | 52900103 | 53900007 | 53900008 | 53900009 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| product_id | | | | | | | | | | | | | | | | | | | | | |
| 1001588 | 1.0 | 1.0 | 1.0 | 1.0 | 1.0 | 1.0 | 1.0 | 1.0 | 1.0 | 1.0 | ... | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 |
| 1002042 | 1.0 | 1.0 | 1.0 | 1.0 | 1.0 | 1.0 | 1.0 | 1.0 | 1.0 | 1.0 | ... | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 |
| 1002098 | 1.0 | 1.0 | 1.0 | 1.0 | 1.0 | 1.0 | 1.0 | 1.0 | 1.0 | 1.0 | ... | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 |
| 1002099 | 1.0 | 1.0 | 1.0 | 1.0 | 1.0 | 1.0 | 1.0 | 1.0 | 1.0 | 1.0 | ... | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 |
| 1002100 | 1.0 | 1.0 | 1.0 | 1.0 | 1.0 | 1.0 | 1.0 | 1.0 | 1.0 | 1.0 | ... | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 |
| ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... |
| 52900102 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | ... | 1.0 | 1.0 | 1.0 | 1.0 | 1.0 | 1.0 | 1.0 | 0.0 | 0.0 | 0.0 |
| 52900103 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | ... | 1.0 | 1.0 | 1.0 | 1.0 | 1.0 | 1.0 | 1.0 | 0.0 | 0.0 | 0.0 |
| 53900007 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | ... | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 1.0 | 1.0 | 1.0 |
| 53900008 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | ... | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 1.0 | 1.0 | 1.0 |
| 53900009 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | ... | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 1.0 | 1.0 | 1.0 |

7228 rows × 7228 columns

# 추천시스템 구현

## Price category similarity

```
product_cat_matrix = np.reciprocal(euclidean_distances(np.array(product_cat['price_category']).reshape(-1,1))+1)
euclidean_matrix = pd.DataFrame(product_cat_matrix,columns=product_cat['product_id'],index=product_cat['product_id'])
euclidean_matrix
```

| product_id | 1001588 | 1002042 | 1002098 | 1002099 | 1002100 | 1002101 | 1002102 | 1002225 | 1002367 | 1002396 | ... | 52900072 | 52900075 | 52900077 | 52900084 | 52900101 | 52900102 | 52900103 | 53900007 | 53900008 | 53900009 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| product_id | | | | | | | | | | | | | | | | | | | | | |
| 1001588 | 1.000000 | 1.000000 | 0.333333 | 0.333333 | 0.333333 | 0.333333 | 0.333333 | 1.000000 | 0.5 | 0.333333 | ... | 0.333333 | 0.5 | 0.200000 | 0.200000 | 0.200000 | 0.333333 | 0.333333 | 0.5 | 0.333333 | 0.333333 |
| 1002042 | 1.000000 | 1.000000 | 0.333333 | 0.333333 | 0.333333 | 0.333333 | 0.333333 | 1.000000 | 0.5 | 0.333333 | ... | 0.333333 | 0.5 | 0.200000 | 0.200000 | 0.200000 | 0.333333 | 0.333333 | 0.5 | 0.333333 | 0.333333 |
| 1002098 | 0.333333 | 0.333333 | 1.000000 | 1.000000 | 1.000000 | 1.000000 | 1.000000 | 0.333333 | 0.5 | 1.000000 | ... | 1.000000 | 0.5 | 0.333333 | 0.333333 | 0.333333 | 1.000000 | 1.000000 | 0.5 | 1.000000 | 1.000000 |
| 1002099 | 0.333333 | 0.333333 | 1.000000 | 1.000000 | 1.000000 | 1.000000 | 1.000000 | 0.333333 | 0.5 | 1.000000 | ... | 1.000000 | 0.5 | 0.333333 | 0.333333 | 0.333333 | 1.000000 | 1.000000 | 0.5 | 1.000000 | 1.000000 |
| 1002100 | 0.333333 | 0.333333 | 1.000000 | 1.000000 | 1.000000 | 1.000000 | 1.000000 | 0.333333 | 0.5 | 1.000000 | ... | 1.000000 | 0.5 | 0.333333 | 0.333333 | 0.333333 | 1.000000 | 1.000000 | 0.5 | 1.000000 | 1.000000 |
| ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... |
| 52900102 | 0.333333 | 0.333333 | 1.000000 | 1.000000 | 1.000000 | 1.000000 | 1.000000 | 0.333333 | 0.5 | 1.000000 | ... | 1.000000 | 0.5 | 0.333333 | 0.333333 | 0.333333 | 1.000000 | 1.000000 | 0.5 | 1.000000 | 1.000000 |
| 52900103 | 0.333333 | 0.333333 | 1.000000 | 1.000000 | 1.000000 | 1.000000 | 1.000000 | 0.333333 | 0.5 | 1.000000 | ... | 1.000000 | 0.5 | 0.333333 | 0.333333 | 0.333333 | 1.000000 | 1.000000 | 0.5 | 1.000000 | 1.000000 |
| 53900007 | 0.500000 | 0.500000 | 0.500000 | 0.500000 | 0.500000 | 0.500000 | 0.500000 | 0.500000 | 1.0 | 0.500000 | ... | 0.500000 | 1.0 | 0.250000 | 0.250000 | 0.250000 | 0.500000 | 0.500000 | 1.0 | 0.500000 | 0.500000 |
| 53900008 | 0.333333 | 0.333333 | 1.000000 | 1.000000 | 1.000000 | 1.000000 | 1.000000 | 0.333333 | 0.5 | 1.000000 | ... | 1.000000 | 0.5 | 0.333333 | 0.333333 | 0.333333 | 1.000000 | 1.000000 | 0.5 | 1.000000 | 1.000000 |
| 53900009 | 0.333333 | 0.333333 | 1.000000 | 1.000000 | 1.000000 | 1.000000 | 1.000000 | 0.333333 | 0.5 | 1.000000 | ... | 1.000000 | 0.5 | 0.333333 | 0.333333 | 0.333333 | 1.000000 | 1.000000 | 0.5 | 1.000000 | 1.000000 |

7228 rows × 7228 columns

# 추천시스템 구현

## Brand similarity

```
tfidf_vectorizer = TfidfVectorizer()
doc_term = tfidf_vectorizer.fit_transform(list(product_cat['brand']))
dt_matrix1 = pd.DataFrame(doc_term.toarray().round(3), index=[i for i in product_cat['product_id']],
                          columns=tfidf_vectorizer.get_feature_names())
dt_matrix1 = dt_matrix1 + 0.01
cos_similar_matrix1 = pd.DataFrame(cosine_similarity(dt_matrix1.values),columns=product_cat['product_id'],
                                   index=product_cat['product_id'])
cos_similar_matrix1
```

| product_id | 1001588 | 1002042 | 1002098 | 1002099 | 1002100 | 1002101 | 1002102 | 1002225 | 1002367 | 1002396 | ... | 52900072 | 52900075 | 52900077 | 52900084 | 52900101 | 52900102 | 52900103 | 53900007 | 53900008 | 53900009 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| product_id | | | | | | | | | | | | | | | | | | | | | |
| 1001588 | 1.000000 | 0.087175 | 0.087175 | 0.087175 | 0.087175 | 0.087175 | 0.087175 | 0.087175 | 1.000000 | 0.087175 | ... | 0.087175 | 0.087175 | 0.087175 | 0.087175 | 0.087175 | 0.087175 | 0.087175 | 0.087175 | 0.087175 | 0.087175 |
| 1002042 | 0.087175 | 1.000000 | 1.000000 | 1.000000 | 1.000000 | 1.000000 | 1.000000 | 0.087175 | 0.087175 | 0.087175 | ... | 0.087175 | 0.087175 | 0.087175 | 0.087175 | 0.087175 | 0.087175 | 0.087175 | 0.087175 | 0.087175 | 0.087175 |
| 1002098 | 0.087175 | 1.000000 | 1.000000 | 1.000000 | 1.000000 | 1.000000 | 1.000000 | 0.087175 | 0.087175 | 0.087175 | ... | 0.087175 | 0.087175 | 0.087175 | 0.087175 | 0.087175 | 0.087175 | 0.087175 | 0.087175 | 0.087175 | 0.087175 |
| 1002099 | 0.087175 | 1.000000 | 1.000000 | 1.000000 | 1.000000 | 1.000000 | 1.000000 | 0.087175 | 0.087175 | 0.087175 | ... | 0.087175 | 0.087175 | 0.087175 | 0.087175 | 0.087175 | 0.087175 | 0.087175 | 0.087175 | 0.087175 | 0.087175 |
| 1002100 | 0.087175 | 1.000000 | 1.000000 | 1.000000 | 1.000000 | 1.000000 | 1.000000 | 0.087175 | 0.087175 | 0.087175 | ... | 0.087175 | 0.087175 | 0.087175 | 0.087175 | 0.087175 | 0.087175 | 0.087175 | 0.087175 | 0.087175 | 0.087175 |
| ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... |
| 52900102 | 0.087175 | 0.087175 | 0.087175 | 0.087175 | 0.087175 | 0.087175 | 0.087175 | 0.087175 | 0.087175 | 0.087175 | ... | 1.000000 | 1.000000 | 1.000000 | 1.000000 | 1.000000 | 1.000000 | 1.000000 | 0.087175 | 0.087175 | 1.000000 |
| 52900103 | 0.087175 | 0.087175 | 0.087175 | 0.087175 | 0.087175 | 0.087175 | 0.087175 | 0.087175 | 0.087175 | 0.087175 | ... | 1.000000 | 1.000000 | 1.000000 | 1.000000 | 1.000000 | 1.000000 | 1.000000 | 0.087175 | 0.087175 | 1.000000 |
| 53900007 | 0.087175 | 0.087175 | 0.087175 | 0.087175 | 0.087175 | 0.087175 | 0.087175 | 0.087175 | 0.087175 | 0.087175 | ... | 0.087175 | 0.087175 | 0.087175 | 0.087175 | 0.087175 | 0.087175 | 0.087175 | 1.000000 | 0.087175 | 0.087175 |
| 53900008 | 0.087175 | 0.087175 | 0.087175 | 0.087175 | 0.087175 | 0.087175 | 0.087175 | 0.087175 | 0.087175 | 0.087175 | ... | 0.087175 | 0.087175 | 0.087175 | 0.087175 | 0.087175 | 0.087175 | 0.087175 | 0.087175 | 1.000000 | 0.087175 |
| 53900009 | 0.087175 | 0.087175 | 0.087175 | 0.087175 | 0.087175 | 0.087175 | 0.087175 | 0.087175 | 0.087175 | 0.087175 | ... | 1.000000 | 1.000000 | 1.000000 | 1.000000 | 1.000000 | 1.000000 | 1.000000 | 0.087175 | 0.087175 | 1.000000 |

7228 rows × 7228 columns

# 추천시스템 구현

## User-Item Matrix

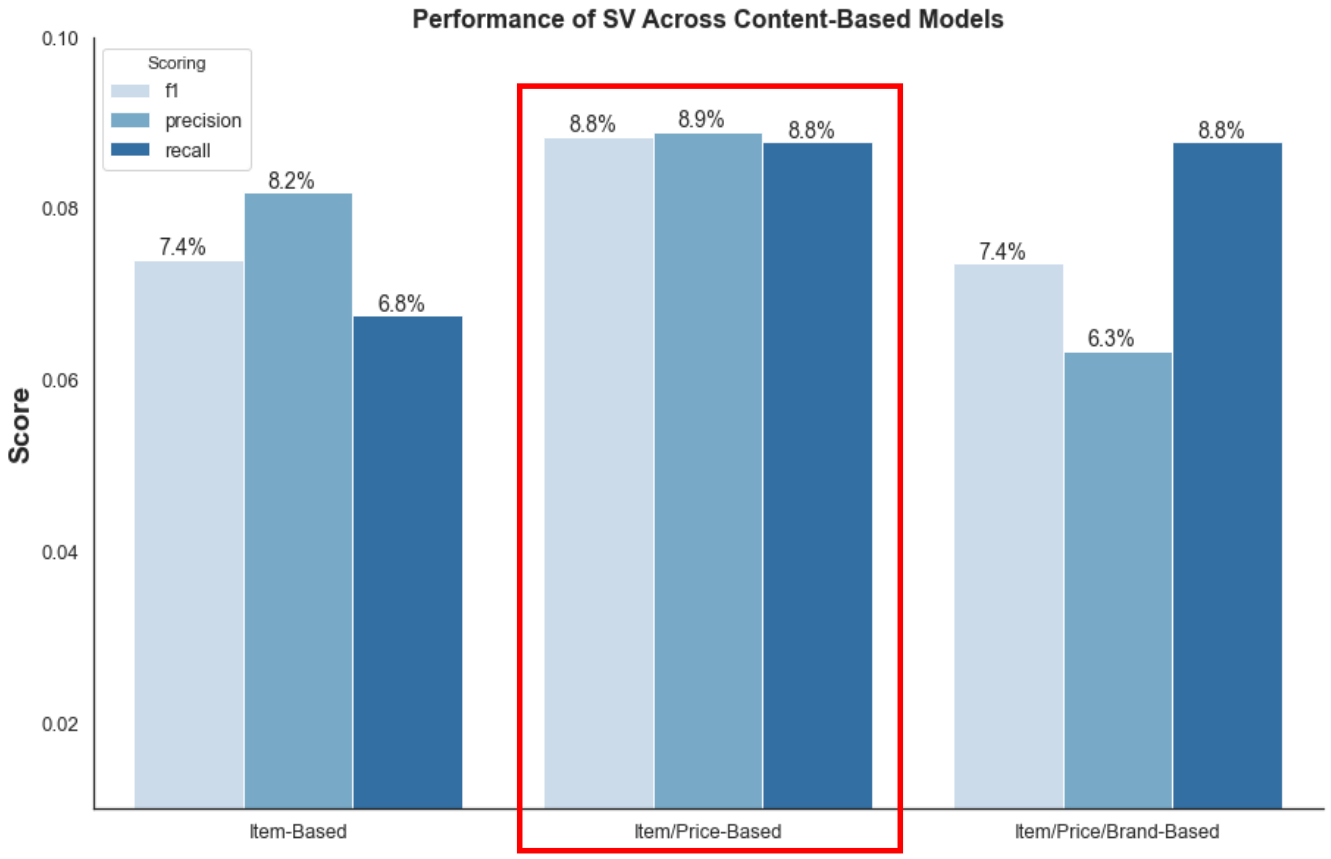
```
X_train_matrix = pd.pivot_table(X_train,values='user_score',index='user_id',columns='product_id')
X_train_matrix=X_train_matrix.fillna(0)
X_train_matrix
```

| product_id | 1001588 | 1002042 | 1002098 | 1002099 | 1002100 | 1002101 | 1002102 | 1002225 | 1002367 | 1002396 | ... | 52900072 | 52900075 | 52900077 | 52900084 | 52900101 | 52900102 | 52900103 | 53900007 | 53900008 | 53900009 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| user_id | | | | | | | | | | | | | | | | | | | | | |
| 241784978 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | ... | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 |
| 390634364 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | ... | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 |
| 421816670 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | ... | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 |
| 429341347 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | ... | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 |
| 430276841 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | ... | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 |
| ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... |
| 556862843 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | ... | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 |
| 556866450 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | ... | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 |
| 556875083 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | ... | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 |
| 556878415 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | ... | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 |
| 556879445 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | ... | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 |

8006 rows × 7228 columns

# 추천시스템 구현

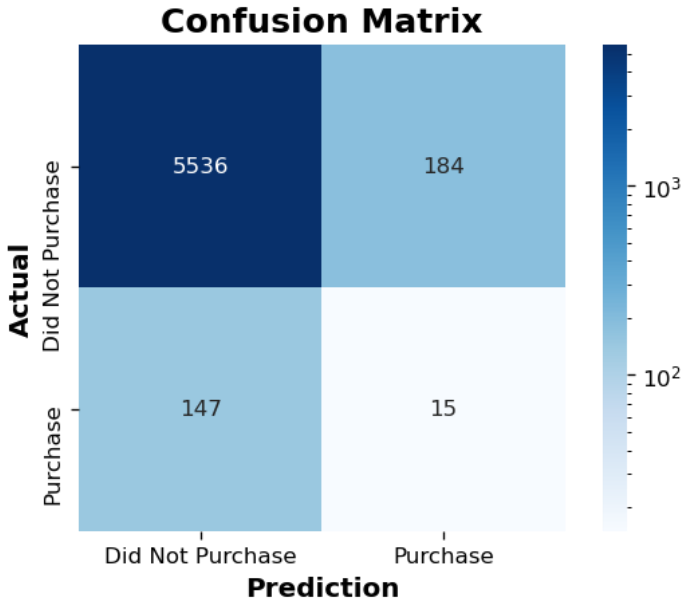
## 성능 비교 및 모델 선택

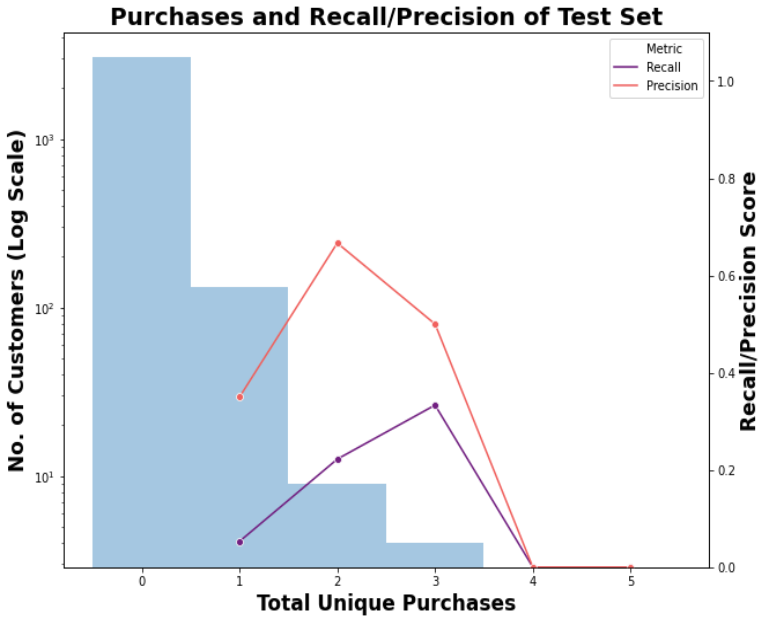



성능이 가장 좋은
Item/Price-Based 모델 채택

# 추천시스템 구현

## 최종 모델 평가




- 모델의 정밀도가 한번 이상 구매한 고객의 경우 훨씬 더 높음
- 전체 정밀도는 구매하지 않은 고객의 예측된 구매에 대한 오탐으로 인해 저하됨
  ➔ 모델의 성능을 측정하는 방법이 구매 고객에 더 편향되어 있기 때문

# 추천시스템 구현

## 모델 시연

- train_set
  - user_id : 514233738

| | user_id | product_id | user_score | user_purchase | interaction_score | category_code | brand | price | price_category |
|---|---|---|---|---|---|---|---|---|---|
| **27637** | 514233738 | 1004617 | 1 | 0 | 0.025000 | electronics.smartphone | xiaomi | 499.37 | 3 |
| **28652** | 514233738 | 1004388 | 1 | 0 | 0.025000 | electronics.smartphone | sony | 694.74 | 3 |
| **28367** | 514233738 | 1005062 | 2 | 0 | 0.034848 | electronics.smartphone | xiaomi | 208.13 | 1 |
| **5420** | 514233738 | 1004836 | 9 | 0 | 0.103788 | electronics.smartphone | samsung | 241.18 | 1 |
| **15779** | 514233738 | 1003306 | 72 | 1 | 0.724242 | electronics.smartphone | apple | 587.56 | 3 |

# 추천시스템 구현

## 모델 시연

- Recommend_set

| | user_id | product_id | predicted_interaction | category_code | brand | price | price_category |
|---|---|---|---|---|---|---|---|
| 0 | 514233738 | 1003801 | 0.339426 | electronics.smartphone | apple | 643.49 | 3 |
| 1 | 514233738 | 1004016 | 0.339426 | electronics.smartphone | samsung | 720.46 | 3 |
| 2 | 514233738 | 1004477 | 0.339426 | electronics.smartphone | google | 900.67 | 3 |
| 3 | 514233738 | 1003815 | 0.339426 | electronics.smartphone | huawei | 772.19 | 3 |
| 4 | 514233738 | 1003879 | 0.339426 | electronics.smartphone | huawei | 694.48 | 3 |
| 5 | 514233738 | 1004901 | 0.339426 | electronics.smartphone | oppo | 592.01 | 3 |
| 6 | 514233738 | 1004902 | 0.339426 | electronics.smartphone | oppo | 592.01 | 3 |
| 7 | 514233738 | 1003896 | 0.339426 | electronics.smartphone | xiaomi | 478.43 | 3 |
| 8 | 514233738 | 1003898 | 0.339426 | electronics.smartphone | oneplus | 540.30 | 3 |
| 9 | 514233738 | 1003899 | 0.339426 | electronics.smartphone | oneplus | 642.46 | 3 |

- test_set
  - user_id : 514233738

| | user_id | product_id | user_score | user_purchase | interaction_score | category_code | brand | price | price_category | predicted_interaction | predicted_purchase |
|---|---|---|---|---|---|---|---|---|---|---|---|
| 772 | 514233738 | 1003499 | 1 | 0 | 0.025000 | electronics.smartphone | oneplus | 462.05 | 2 | 0.136917 | 0 |
| 2389 | 514233738 | 1004739 | 1 | 0 | 0.025000 | electronics.smartphone | xiaomi | 197.55 | 1 | 0.211904 | 0 |
| 4581 | 514233738 | 1005065 | 1 | 0 | 0.025000 | electronics.smartphone | xiaomi | 233.91 | 1 | 0.211904 | 0 |
| 4875 | 514233738 | 1005038 | 53 | 1 | 0.537121 | electronics.smartphone | google | 579.17 | 3 | 0.339426 | 0 |
| 5287 | 514233738 | 1005160 | 1 | 0 | 0.025000 | electronics.smartphone | xiaomi | 231.41 | 1 | 0.211904 | 0 |
| 5544 | 514233738 | 1005169 | 1 | 0 | 0.025000 | electronics.smartphone | samsung | 251.97 | 1 | 0.211904 | 0 |

# 추천시스템 구현

## 모델 시연

- train_set
  - user_id : 512714093

| | user_id | product_id | user_score | user_purchase | interaction_score | category_code | brand | price | price_category |
|---|---|---|---|---|---|---|---|---|---|
| **20411** | 512714093 | 28705258 | 2 | 0 | 0.034848 | apparel.shoes | respect | 23.94 | 0 |
| **10985** | 512714093 | 2701639 | 5 | 0 | 0.064394 | appliances.kitchen.refrigerators | indesit | 257.15 | 1 |
| **20407** | 512714093 | 2800656 | 1 | 0 | 0.025000 | appliances.kitchen.refrigerators | arg | 180.16 | 0 |
| **20402** | 512714093 | 2800428 | 1 | 0 | 0.025000 | appliances.kitchen.refrigerators | haier | 231.33 | 1 |
| **20432** | 512714093 | 45600846 | 1 | 0 | 0.025000 | apparel.shoes | keddo | 54.80 | 0 |
| **20331** | 512714093 | 2700274 | 1 | 0 | 0.025000 | appliances.kitchen.refrigerators | pozis | 272.59 | 1 |
| **20367** | 512714093 | 2701683 | 2 | 0 | 0.034848 | appliances.kitchen.refrigerators | beko | 237.94 | 1 |
| **20384** | 512714093 | 2702529 | 2 | 0 | 0.034848 | appliances.kitchen.refrigerators | dauscher | 771.45 | 4 |
| **20360** | 512714093 | 2701650 | 1 | 0 | 0.025000 | appliances.kitchen.refrigerators | indesit | 257.38 | 1 |
| **20380** | 512714093 | 2702086 | 1 | 0 | 0.025000 | appliances.kitchen.refrigerators | dauscher | 213.62 | 0 |
| **4174** | 512714093 | 2702277 | 2 | 0 | 0.034848 | appliances.kitchen.refrigerators | lg | 465.70 | 3 |

# 추천시스템 구현

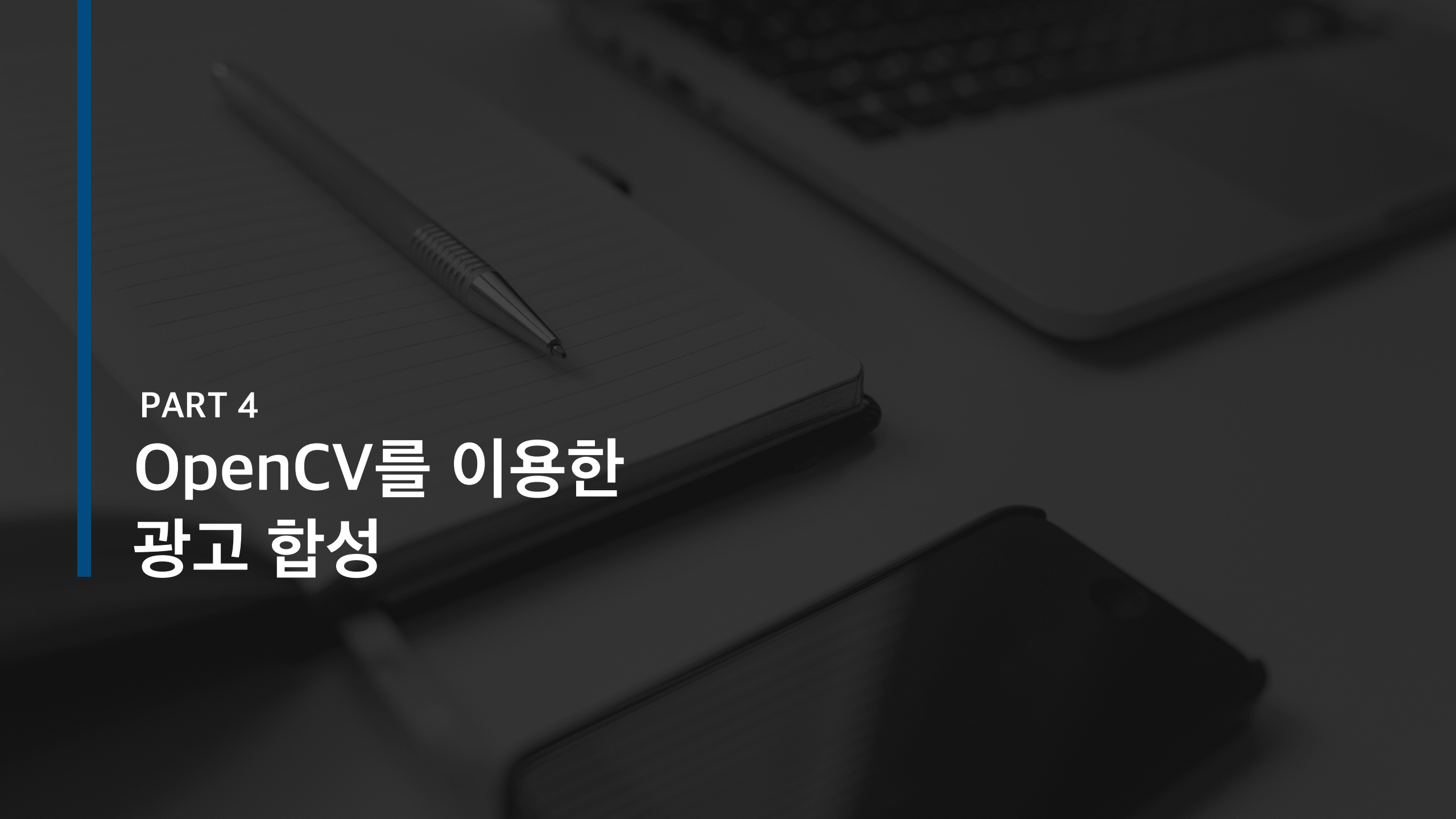
## 모델 시연

- Recommend_set

| | user_id | product_id | predicted_interaction | category_code | brand | price | price_category |
|---|---|---|---|---|---|---|---|
| 0 | 512714093 | 2800229 | 0.284685 | appliances.kitchen.refrigerators | elenberg | 231.64 | 1 |
| 1 | 512714093 | 2800454 | 0.284685 | appliances.kitchen.refrigerators | atlantic | 242.45 | 1 |
| 2 | 512714093 | 2800007 | 0.284685 | appliances.kitchen.refrigerators | atlant | 257.15 | 1 |
| 3 | 512714093 | 2800020 | 0.284685 | appliances.kitchen.refrigerators | midea | 244.54 | 1 |
| 4 | 512714093 | 2800037 | 0.284685 | appliances.kitchen.refrigerators | midea | 223.66 | 1 |
| 5 | 512714093 | 2800057 | 0.284685 | appliances.kitchen.refrigerators | pozis | 249.17 | 1 |
| 6 | 512714093 | 2800157 | 0.284685 | appliances.kitchen.refrigerators | midea | 257.38 | 1 |
| 7 | 512714093 | 2800174 | 0.284685 | appliances.kitchen.refrigerators | atlantic | 275.04 | 1 |
| 8 | 512714093 | 2800179 | 0.284685 | appliances.kitchen.refrigerators | atlant | 278.59 | 1 |
| 9 | 512714093 | 2700071 | 0.284685 | appliances.kitchen.refrigerators | atlant | 226.49 | 1 |

- test_set
  - user_id : 512714093

| | user_id | product_id | user_score | user_purchase | interaction_score | category_code | brand | price | price_category | predicted_interaction | predicted_purchase |
|---|---|---|---|---|---|---|---|---|---|---|---|
| 2147 | 512714093 | 45600708 | 1 | 0 | 0.025000 | apparel.shoes | crosby | 49.91 | 0 | 0.094340 | 0 |
| 4289 | 512714093 | 2700588 | 57 | 1 | 0.576515 | appliances.kitchen.refrigerators | indesit | 295.99 | 2 | 0.169431 | 0 |
| 4295 | 512714093 | 2702476 | 1 | 0 | 0.025000 | appliances.kitchen.refrigerators | midea | 218.54 | 0 | 0.149679 | 0 |
| 5734 | 512714093 | 2700609 | 1 | 0 | 0.025000 | appliances.kitchen.refrigerators | indesit | 310.37 | 2 | 0.169431 | 0 |



PART 4

# OpenCV를 이용한 광고 합성

# OpenCV를 이용한 광고 합성


OpenCV (Open Source Computer Vision)

## OpenCV란?

- 실시간 **컴퓨터 비전**을 목적으로 한 프로그래밍 라이브러리
- 라이센스 비용 지불을 하지 않아 많은 기업이나 개인 개발자가 사용 중
- C++, C, Python, JAVA와 같은 다양한 인터페이스와 Windows, Linux, Mac OS, iOS 및 Android같은 다양한 OS를 지원
- 알고리즘 상으로 계산 효율성과 실시간 응용 프로그램에 중점을 두고 설계되었기 때문에 최적화 알고리즘을 생각하지 않고도 품질 좋은 상용 프로그램을 만들 수 있음
- 멀티코어 프로세싱을 지원하기 때문에 다양한 상황에 응용이 가능

PART 4

# OpenCV를 이용한 광고 합성

## 1) 사용한 라이브러리 및 OpenCV버전

```
import cv2
import numpy as np
import sys
import pandas as pd
import random
import os
```

```
print(cv2.__version__)
```

```
4.5.3
```

## 2) 영상에 합성할 product_id 뽑기

```
recommend=pd.read_csv('./recommend.csv')
recommend
```

| | product_id | category_code | brand | price |
|---|---|---|---|---|
| 0 | 4400519 | appliances.kitchen.coffee_machine | polaris | 24.88 |
| 1 | 4400406 | appliances.kitchen.coffee_machine | galaxy | 17.25 |
| 2 | 4400484 | appliances.kitchen.coffee_machine | galaxy | 16.96 |
| 3 | 4400346 | appliances.kitchen.coffee_machine | galaxy | 20.57 |
| 4 | 4400322 | appliances.kitchen.coffee_machine | maxwell | 19.02 |
| 5 | 4400255 | appliances.kitchen.coffee_machine | bosch | 32.43 |
| 6 | 4400490 | appliances.kitchen.coffee_machine | vitek | 23.14 |
| 7 | 4400405 | appliances.kitchen.coffee_machine | galaxy | 17.25 |
| 8 | 4400260 | appliances.kitchen.coffee_machine | bene | 8.99 |
| 9 | 4400168 | appliances.kitchen.coffee_machine | vitek | 28.24 |

```
#random.seed(14)
ad_product_id=str(random.sample(recommend['product_id'].unique().tolist(),1)[0])
ad_product_id
```

```
'4400406'
```

# OpenCV를 이용한 광고 합성

### 3) 합성할 영상과 광고 사진 선택


▲ 원본 영상


▲ 크로마키 처리된 영상

➔ 합성할 부분이 크로마키 처리된 영상을 불러옴

# OpenCV를 이용한 광고 합성

## 3) 합성할 영상과 광고 사진 선택

```
path_dir='./ad'
file_list=os.listdir(path_dir)
file_list
```

```
['1002544.png',
 '1004767.png',
 '1004870.png',
 '1304849.jpg',
 '1306449.jpg',
 '1307449.jpg',
 '1480707.jpg',
 '1801858.jpg',
 '21401593.jpg',
 '21403655.jpg',
 '21404354.jpg',
 '21405806.jpg',
 '21406140.png',
 '21408704.jpg',
 '2900536.jpg',
 '321655812.jpg',
 '3600182.jpg',
 '3600231.jpg',
 '3600987.jpg',
 '3601244.jpg',
```

```
product_id=ad_product_id+'.jpg'
imgfile=path_dir+'/'+product_id
img=cv2.imread(imgfile,cv2.IMREAD_COLOR)
```


▲ 광고 이미지

➔ 2)에서 추출한 product_id에 해당하는 광고 이미지를 가져옴

# OpenCV를 이용한 광고 합성

## 4) 마스크 생성 및 크로마키 모양 파악

```
frame_hsv=cv2.cvtColor(frame,cv2.COLOR_BGR2HSV)
frame_threshold=cv2.inRange(frame_hsv,(50,150,0),(80,255,255))
kernel=np.ones((5,5),np.uint8)
mask=cv2.erode(frame_threshold,kernel)
```

- OpenCV는 RBG 순서가 아닌 BGR순서이며 HSV 타입으로 이용 가능
- inRange()함수를 사용하여 크로마키에 해당하는 부분을 이진화하여 mask를 생성
- cv2.erode()함수를 사용하여 작은 noise들을 제거

▲ mask

```
contours,_ =cv2.findContours(mask,cv2.RETR_LIST,cv2.CHAIN_APPROX_SIMPLE)
for contour in contours:
    point=cv2.approxPolyDP(contour,0.01*cv2.arcLength(contour,True),True)
```

- cv2.findContours()를 사용하여 mask의 도형이 어떤 모형인지 파악
- cv2.approxPolyDP()를 사용하여 도형의 꼭짓점을 알아내고, point에 그 값들 저장

PART 4
# OpenCV를 이용한 광고 합성

## 5) 이미지 조정 및 합성

```
if len(point)==4:
    pts=point.copy()
    pts=np.reshape(point,(4,-1))

    ## 합성할 이미지 사이즈 조정
    width=max(pts[2][0],pts[3][0])-min(pts[0][0],pts[1][0])
    height=max(pts[1][1],pts[2][1])-min(pts[0][1],pts[3][1])
    if width!=0 and height!=0:

        new=cv2.resize(img, dsize=(width,height), interpolation=cv2.INTER_AREA)

        ## 이미지합성
        origin=frame2[min(pts[0][1],pts[3][1]):max(pts[1][1],pts[2][1]),min(pts[0][0],pts[1][0]):max(pts[2][0],pts[3][0])]
        adver=cv2.addWeighted(origin,0.3,new,0.7,0)
        frame2[min(pts[0][1],pts[3][1]):max(pts[1][1],pts[2][1]),min(pts[0][0],pts[1][0]):max(pts[2][0],pts[3][0])]=adver
```

- 4)에서 파악한 모형이 사각형일 경우, 크로마키의 width와 height를 구한 후 **cv2.resize()**를 이용하여 크로마키 사이즈에 맞게 광고 이미지의 사이즈를 조절함
- 원본 영상과의 이질감을 덜기 위해 **cv2.addWeighted()**를 사용하여 원본의 배경과 광고 이미지를 blending 시켜줌
- 합성된 이미지를 영상에 합성시킴

PART 4

# OpenCV를 이용한 광고 합성

## 6) 합성 결과

1

2

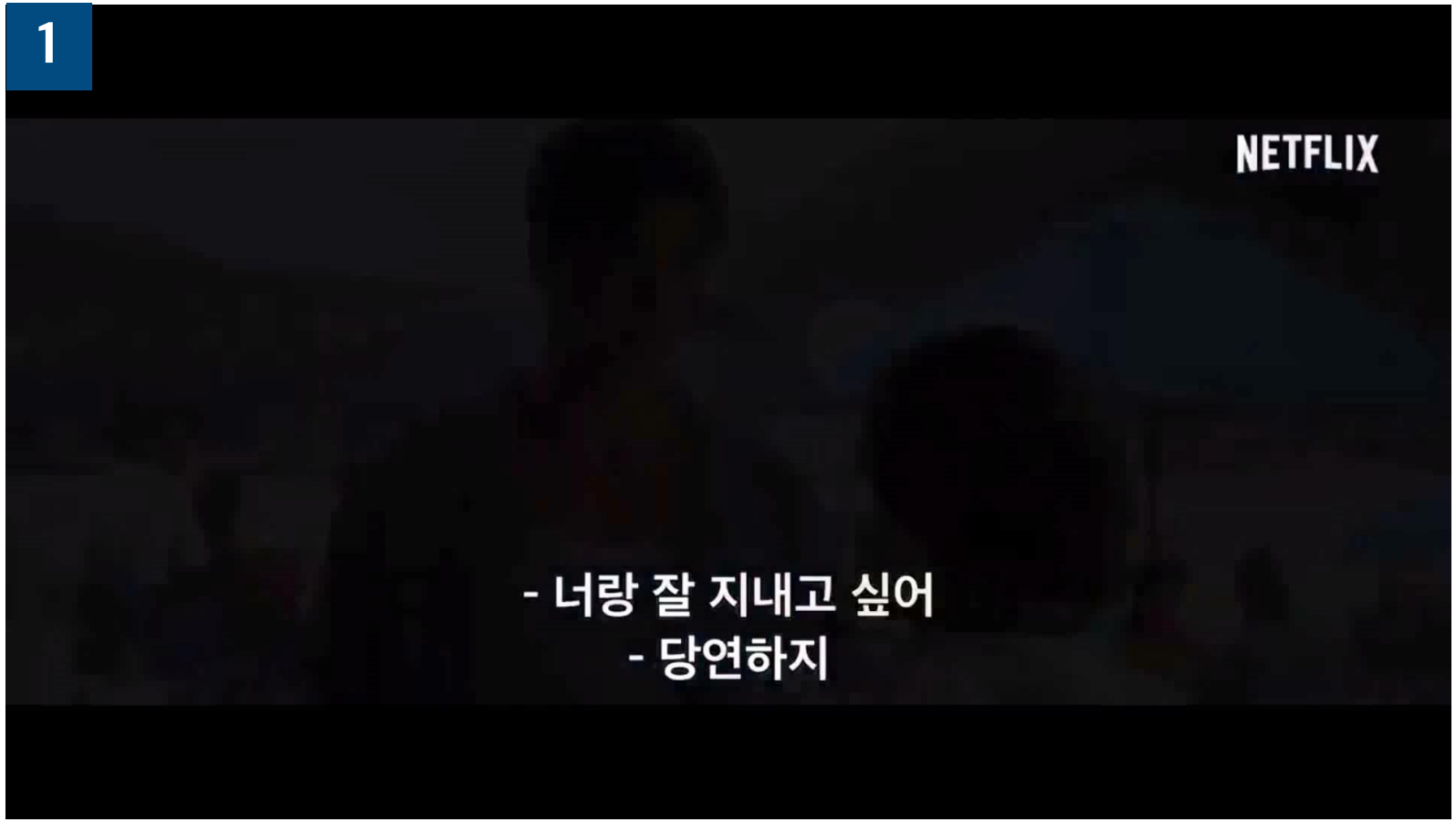

3

PART 4

# OpenCV를 이용한 광고 합성

## 7) 광고 합성 응용 - 움직이는 크로마키

1




2


3

# OpenCV를 이용한 광고 합성

## 8) 광고 합성 응용 - 영상 광고 합성

```
frame2 = cv2.resize(frame2, (mov_width,mov_height))
```

- 광고 영상을 배경이 되는 영화 영상과 같은 사이즈로 수정

```
if len(point)==4:
    pts=point.copy()
    pts=np.reshape(point,(4,-1))

    ## 합성할 이미지 사이즈 조정
    width=abs(max(pts[2][0],pts[3][0])-min(pts[0][0],pts[1][0]))
    height=abs(max(pts[1][1],pts[2][1])-min(pts[0][1],pts[3][1]))
    if width!=0 and height!=0:
        new=cv2.resize(frame3, dsize=(width,height), interpolation=cv2.INTER_AREA)

        ## 이미지합성
        frame2[min(pts[0][1],pts[3][1]):max(pts[1][1],pts[2][1]),min(pts[0][0],pts[1][0]):max(pts[2][0],pts[3][0])]=new
```

- 4)에서 파악한 모형이 사각형일 경우, 크로마키의 width와 height를 구한 후 **cv2.resize()**를 이용해 크로마키 사이즈에 맞게 광고 영상의 사이즈를 조절함
- 영화 영상 사이즈의 광고 영상 위에 크로마키 사이즈의 광고 영상을 합성함

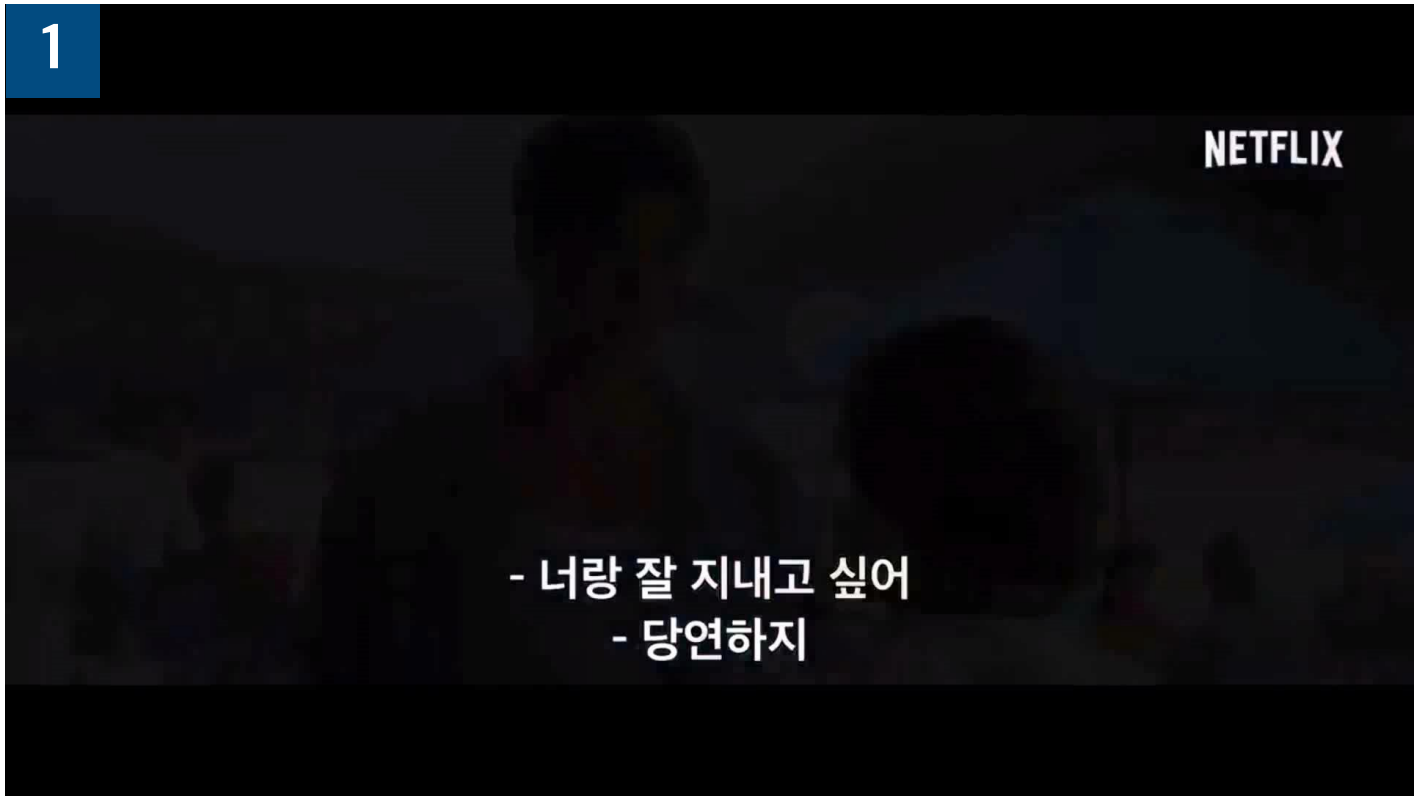
```
## 영상에 광고 영상 합성
cv2.copyTo(frame2,mask,frame1)
```

- **cv2.copyTo()**를 이용하여 광고 영상, mask, 영화 영상을 합성함

PART 4

# OpenCV를 이용한 광고 합성

## 8) 광고 합성 응용 - 영상 광고 합성





PART 5

# 의의 및 한계

PART 5

# 의의 및 한계

## 의의

- 비타민에서 공부했던 추천시스템을 구현하는 데에 그치지 않고, 심화적으로 새로운 분야를 공부하여 우리만의 알고리즘을 개발한 것에 의의가 있음
- 배우, 작가, 감독 등이 따로 ppl을 위한 대본을 고민하지 않아도 됨
- 같은 영상이라도 사람마다, 상황마다 다른 광고를 보여줄 수 있음
- 미디어 내 ppl의 증가 추세를 보아, 경영·마케팅 산업에 실제로 적용해 볼 가치가 있는 프로젝트임

## 한계

- 데이터 개수가 너무 많아 컴퓨터 내 로드할 수 있는 RAM을 초과하여 일부 데이터를 걸러내야 했음
- 구매를 0번 한 고객들이 상대적으로 많아, 데이터의 분포가 불균형적이었음
- 움직이는 크로마키에 합성했을 때 약간의 어색함이 남아 있음
- 영상 속 크로마키의 모양에 따라 합성할 이미지의 모양이 유연하게 변하는 데에 한계가 있음

PART 6

# 참고 자료

PART 6

# 참고자료


https://towardsdatascience.com/a-content-based-recommender-for-e-commerce-web-store-7554b5b73eac

https://eochodevlog.tistory.com/44

https://pysource.com/blog/

https://blog.naver.com/samsjang/220503082434

# THANK YOU