B.A.C.S.

이커머스 고객 행동 분석과 로지스틱 회귀를 이용한 구매 예측 모델링

황지연 곽유민 김예린 최다연

Business Analytics & Consulting Society

미국 전자제품 이커머스의 10월 한 달간 고객 행동 로그 데이터를 사용함

데이터 셋 소개

데이터 출처: Kaggle의 미국 이커머스 고객 행동 로그 데이터 사용.

분석 기간: 10월 한 달 간의 데이터 사용.

주요 내용: 대형 가전 및 전자제품을 주로 판매하는 이커머스 데이터로, 80만 건의 고객 행동 로그를 포함.

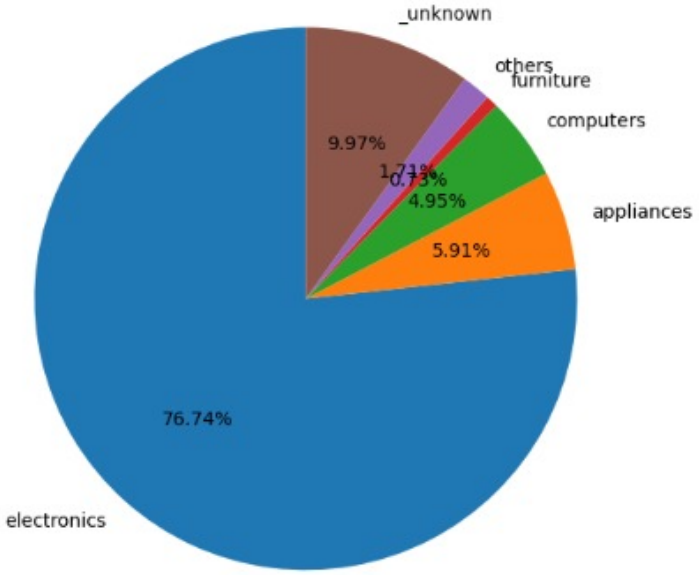
[주요 컬럼 설명]

- event_time: 이벤트 발생 시간 (UTC)
- event_type: 이벤트 유형 (view, cart, purchase 등)
- product_id: 상품 ID
- category_code: 카테고리 코드 (4단계 카테고리 체계)
- brand: 브랜드명 (예: Samsung, 기타 등)
- price: 상품 가격
- user_id: 사용자 고유 ID
- user_session: 사용자 세션 ID

event_time When event is was happened	event_type Event type: one of [view, cart, remove_from_cart, purchase]	product_id Product ID	category_id Product category ID	category_code Category meaningful name (if present)
845041 unique values	view 90% cart 6% Other (37346) 4%	102 4.18m	2144415921 2227847333 b b	[null] 27% computers.comp... 13% Other (532193) 60%
2020-09-24 11:57:06 UTC	view	1996170	2144415922528452715	electronics.telephone
2020-09-24 11:57:26 UTC	view	139905	2144415926932472027	computers.components.cooler
2020-09-24 11:57:27 UTC	view	215454	2144415927158964449	

전자제품 중에서도 스마트폰 판매에 집중하고 있음

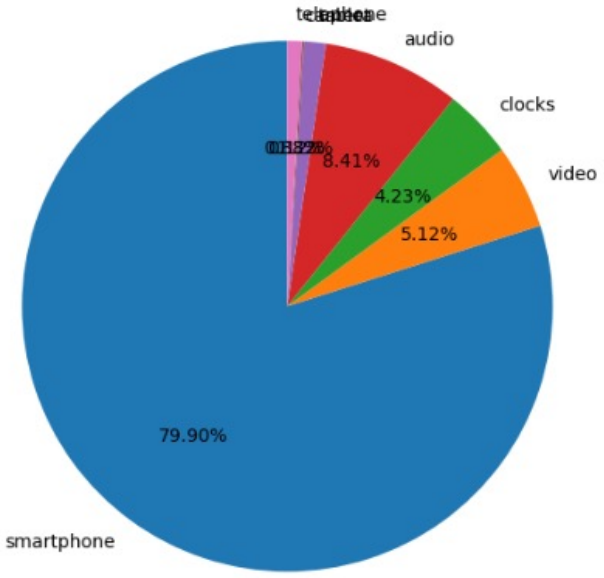
데이터셋 소개



1차 카테고리 중

(electronics, furniture, apparel 등)

electronic이 차지하는 매출 비중은 약 77%



2차 카테고리 중

(smartphone, audio, video 등)

smartphone의 매출 비중은 electronic 내 80%

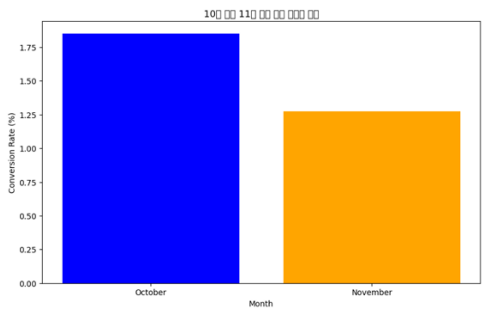
문제 가정) 전자기기 중심의 이커머스 플랫폼인 하이마트를 가상으로 설정하여, 고객 행동 문제를 분석할 시나리오로 활용

전체·상품별 구매 전환율과 매출 모두 감소를 보이고 있음

주제 선정 배경(문제 인식)

(1) 전체 구매 전환율 변화 시각화: 10월 대비 11월 전체 구매 전환율 감소

```python
import pandas as pd
import matplotlib.pyplot as plt

# 10월과 11월 데이터 불러오기
oct_data = pd.read_csv('./data/2019-Oct.csv')
nov_data = pd.read_csv('./data/2019-Nov.csv')

# 10월과 11월의 view와 purchase 이벤트 수 계산
oct_views = oct_data[oct_data['event_type'] == 'view'].shape[0]
oct_purchases = oct_data[oct_data['event_type'] == 'purchase'].shape[0]
nov_views = nov_data[nov_data['event_type'] == 'view'].shape[0]
nov_purchases = nov_data[nov_data['event_type'] == 'purchase'].shape[0]

# 전환율 계산
oct_conversion_rate = (oct_purchases / oct_views) * 100
nov_conversion_rate = (nov_purchases / nov_views) * 100

# 시각화
months = ['October', 'November']
conversion_rates = [oct_conversion_rate, nov_conversion_rate]

plt.figure(figsize=(10, 6))
plt.bar(months, conversion_rates, color=['blue', 'orange'])
plt.xlabel("Month")
plt.ylabel("Conversion Rate (%)")
plt.title("10월 대비 11월 전체 구매 전환율 변화")
plt.show()
```

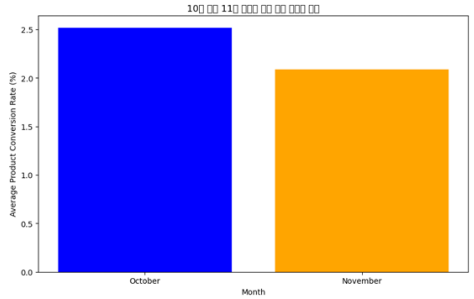


1.75% -> 1.2%로

약 **31.43% 감소**

(2) 상품별 구매 전환율: 10월 대비 11월 상품별 평균 구매 전환율 감소

```python
# 상품별 전환율 계산
oct_product_conversion = oct_data[oct_data['event_type'] == 'purchase'].groupby('product_id').size() / \
                         oct_data[oct_data['event_type'] == 'view'].groupby('product_id').size()
nov_product_conversion = nov_data[nov_data['event_type'] == 'purchase'].groupby('product_id').size() / \
                         nov_data[nov_data['event_type'] == 'view'].groupby('product_id').size()

# 평균 전환율 계산
oct_product_conversion_mean = oct_product_conversion.mean() * 100
nov_product_conversion_mean = nov_product_conversion.mean() * 100

# 시각화
plt.figure(figsize=(10, 6))
plt.bar(['October', 'November'], [oct_product_conversion_mean, nov_product_conversion_mean], color=['blue', 'orange'])
plt.xlabel("Month")
plt.ylabel("Average Product Conversion Rate (%)")
plt.title("10월 대비 11월 상품별 평균 구매 전환율 변화")
plt.show()
```

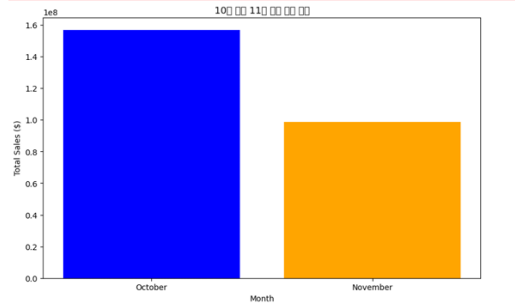


2.5% -> 2%

약 **20% 감소**

(3) 매출 변화 분석: 10월과 11월의 전체 매출 비교, 매출 하락세 확인

```python
# 10월과 11월 매출 계산
oct_total_sales = oct_data[oct_data['event_type'] == 'purchase']['price'].sum()
nov_total_sales = nov_data[nov_data['event_type'] == 'purchase']['price'].sum()

# 시각화
plt.figure(figsize=(10, 6))
plt.bar(['October', 'November'], [oct_total_sales, nov_total_sales], color=['blue', 'orange'])
plt.xlabel("Month")
plt.ylabel("Total Sales ($)")
plt.title("10월 대비 11월 전체 매출 변화")
plt.show()
```

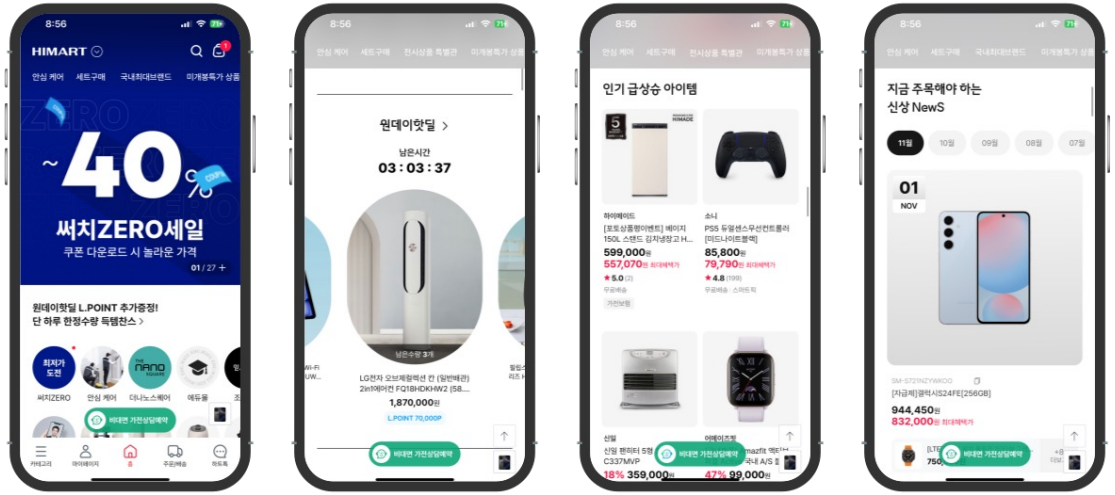


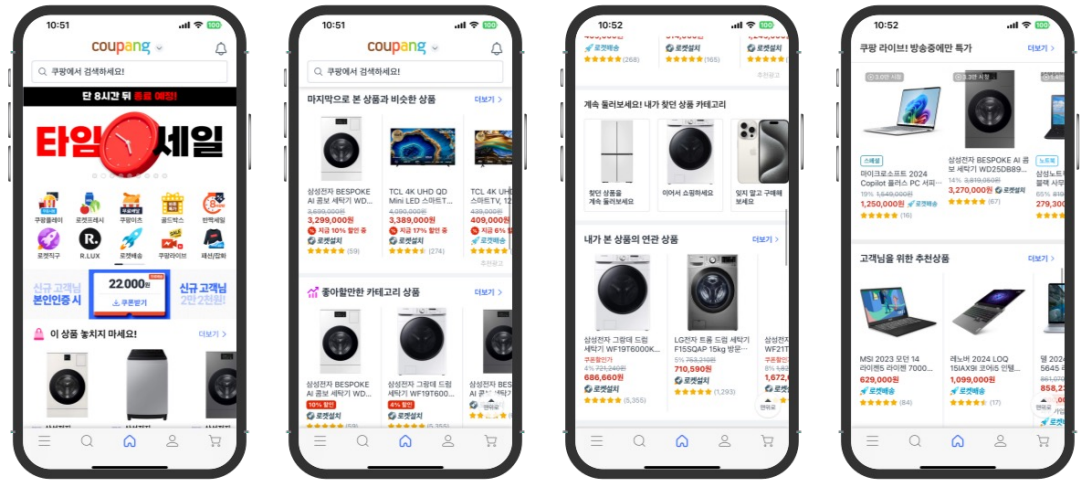
1.4억 달러 -> 1억 달러

약 **28.57% 감소**

하이마트는 쿠팡과 비교하였을 때, 개인화되지 않은 단순한 추천 시스템을 가지고 있음

하이마트와 쿠팡의 추천 시스템 비교





하이마트 홈화면 추천영역 = 단순 추천

- 신상품
- 원데이 핫딜
- 인기 급상승 상품

쿠팡 홈화면 추천영역 = 개인화된 추천

- 내가 본 상품 기반 추천
- 좋아할 만한 브랜드 상품
- 연관 상품 추천
- 관련 라이브 추천

하이마트의 추천 시스템은 고객의 의도를 반영하지 못하며, 클릭한 상품과 가격대 및 용도가 다른 상품을 추천하고 있음

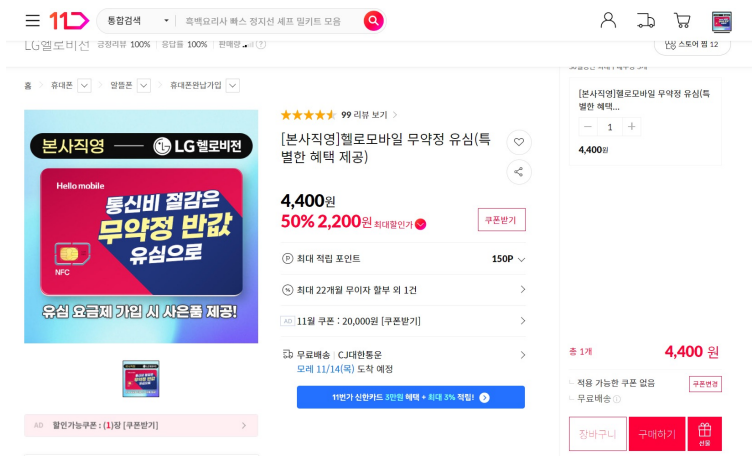
하이마트와 11번가의 추천 시스템 비교

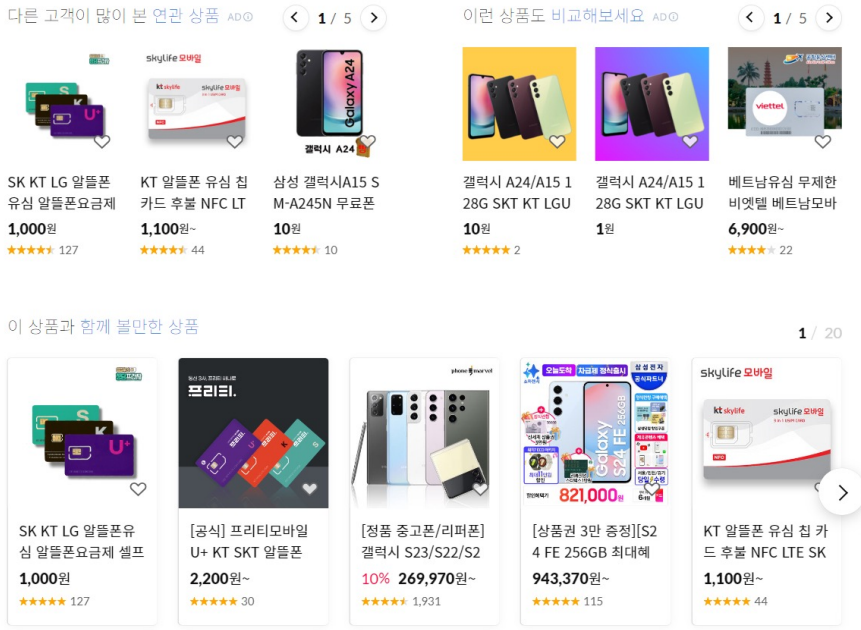
하이마트 추천 시스템의 문제점

1. **비관련 상품 추천**: USIM 칩 클릭 시 관련 없는 가전제품 추천 → 고객 의도를 반영하지 못해 신뢰도 저하.
2. **추천 상품의 비일관성**: 클릭한 상품과 가격대 및 용도가 다른 상품 추천 → 구매 결정 혼란 유발.
3. **가격대 미적합**: 비슷한 가격대가 아닌 고가 또는 저가 제품 추천 → 구매 욕구 감소 및 매출에 부정적 영향.

경쟁사인 11번가는 연관성 있는 상품만을 추천하여 고객의 관심을 효과적으로 반영하고 있음

하이마트와 11번가의 추천 시스템 비교





경쟁사(11번가)는 연관 상품과 비교 상품 등으로 추천을 세분화하고, 연관성 있는 상품만을 추천하여 고객의 관심을 효과적으로 반영하고 있음.

개인화된 추천 시스템 부족으로 고객이 원하는 상품 노출에 한계 발생

원인 분석 및 해결 방안 도출

문제:
구매전환율 **31.43% 감소**,
매출 **28.57% 감소**

원인: 개인화된 추천 시스템 미흡하여 고객 맞춤 상품 노출에 한계가 있음

해결방안: 추천 시스템 고도화

목표: 고객 맞춤형 추천을 통해 전환율 회복 및 매출 증대

이벤트/행동, 상품, 사용자 관련 3가지로 카테고리를 나누어 변수를 추가함

데이터 전처리

1. 이벤트/행동 관련 데이터
•**event_time**: 이벤트 발생 시간
•**event_type**: 이벤트 유형 (예: view, cart, purchase)
•**user_session**: 사용자 세션 ID

2. 상품 관련 데이터
•**product_id**: 상품 ID
•**category_code**: 카테고리 코드 (4단계 카테고리 체계)
•**brand**: 브랜드명
•**price**: 상품 가격

3. 사용자 관련 데이터
•**user_id**: 사용자 고유 ID

1. 이벤트/행동 관련 데이터
•**hour**: 이벤트 발생 시간의 시각 (시 단위)
•**is_weekend**: 주말 여부
•**view_count**: 조회 수
•**cart_count**: 장바구니에 추가한 횟수
•**purchase_count**: 구매 횟수
•**session_length**: 세션 길이
•**flow_classification**: 전환 흐름 분류
•**time_of_day**: 시간대 구분 (예: morning, afternoon 등)
•**first_event**: 세션에서의 첫 이벤트 유형

2. 상품 관련 데이터
•**large**: 대분류 카테고리
•**medium**: 중분류 카테고리
•**small**: 소분류 카테고리
•**category_avg_price**: 카테고리별 평균 가격
•**price_difference**: 평균 가격과의 차이
•**brand_popularity**: 브랜드 인기도

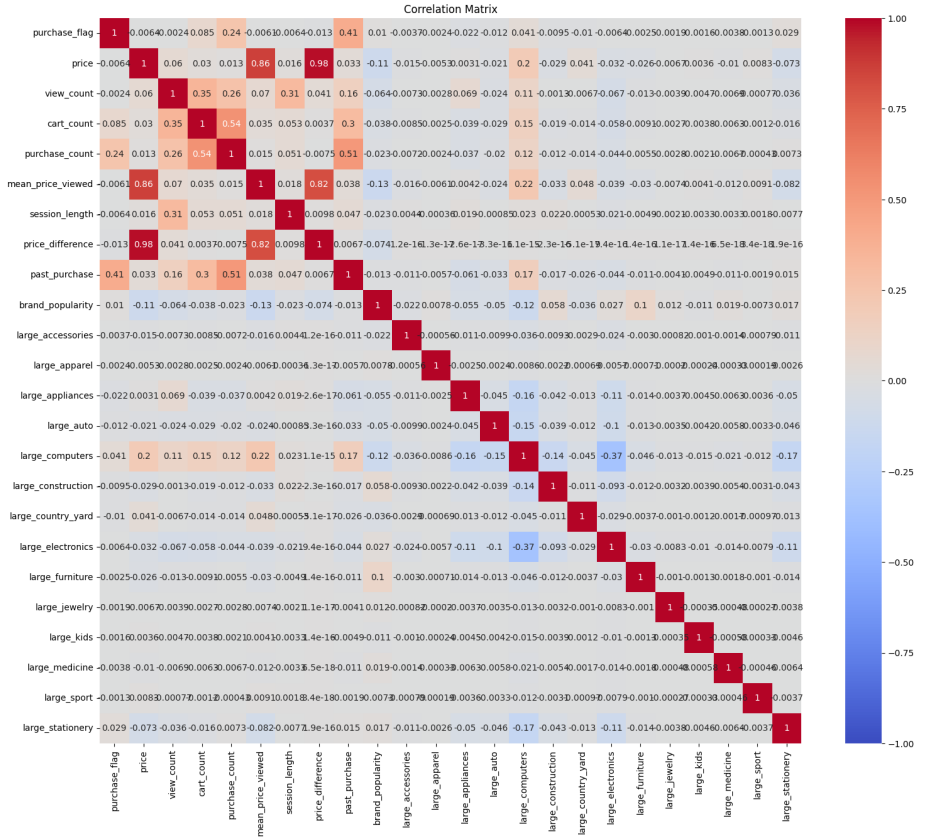
NaN 값을 처리하고, 원-핫 인코딩을 거쳐 상관관계를 분석함

데이터 전처리

```
# 연속형 변수에 대한 NaN 값 처리 (평균으로 대체)
oct['session_length'].fillna(oct['session_length'].mean(), inplace=True)
oct['session_event_count'].fillna(oct['session_event_count'].mean(), inplace=True)
oct['category_avg_price'].fillna(oct['category_avg_price'].mean(), inplace=True)
oct['price_difference'].fillna(oct['price_difference'].mean(), inplace=True)

# 범주형 변수에 대한 NaN 값 처리 ('unknown'으로 대체)
oct['large'].fillna('unknown', inplace=True)
oct['medium'].fillna('unknown', inplace=True)
oct['small'].fillna('unknown', inplace=True)
oct['first_event'].fillna('unknown', inplace=True)
```

```
# 범주형 열을 수치형으로 변환 (One-Hot Encoding)
categorical_features = ['large', 'medium', 'small', 'time_of_day', 'flow_classification', 'first_event']
oct_encoded = pd.get_dummies(oct, columns=categorical_features, drop_first=True)

# 변환된 데이터프레임 확인
oct_encoded.head()
```

Price와 상관관계가 높은 mean_price_viewed, price_difference, past_purchase 제거하기로 함



전체 예측 중 정답률은 높으나, '구매' 예측을 강화할 필요를 느낌

로지스틱 회귀 분석

```python
# 데이터 스케일링
scaler = StandardScaler()
X_scaled = scaler.fit_transform(X)

# 데이터 분할
X_train, X_test, y_train, y_test = train_test_split(X_scaled, y, test_size=0.2, random_state=42)

# 로지스틱 회귀 모델 학습
model = LogisticRegression(max_iter=1000, random_state=42)
model.fit(X_train, y_train)

# 예측
y_pred = model.predict(X_test)
y_pred_proba = model.predict_proba(X_test)[:, 1]

# 모델 평가
print("Accuracy:", accuracy_score(y_test, y_pred))
print("Precision:", precision_score(y_test, y_pred))
print("Recall:", recall_score(y_test, y_pred))
print("F1-Score:", f1_score(y_test, y_pred))
```

- **Accuracy (정확도): 96%**
  - 전체 예측 중 정답률은 높음

- **Precision (정밀도): 35%**
  - '구매'로 예측한 것 중 실제 구매 비율이 낮음

- **Recall (재현율): 4.4%**
  - 실제 '구매' 중에서 모델이 찾아낸 비율이 낮아 개선 필요

- **F1-Score: 7.8%**
  - 정밀도와 재현율의 조화 평균으로 낮은 점수.

정밀도와 재현율은 '구매' 할 것이라고 예측하는 고객들을 얼마나 모델이 예측했는지 평가하는 지표임

평가 지표 설명

•**정밀도**는 모델이 잘못된 '양성' 예측(거짓 긍정)을 얼마나 줄이는지 평가하는 데 중요합니다. 예를 들어, 마케팅 캠페인에서 '구매'할 가능성이 높은 고객을 대상으로 할 때, 잘못된 예측을 줄이는 것이 중요합니다.

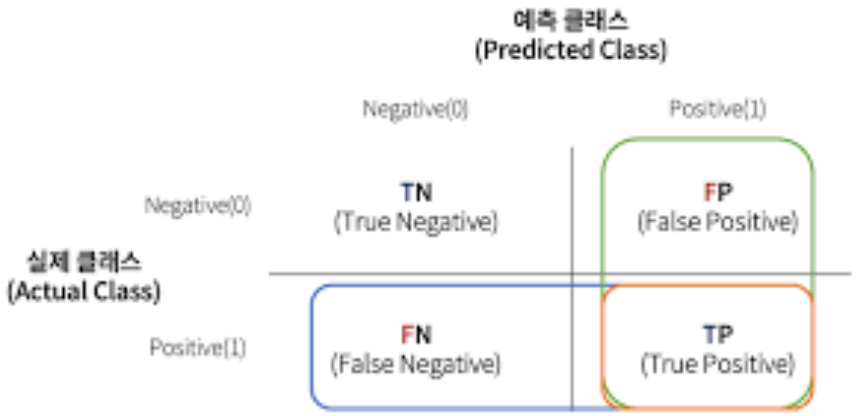
•TP / (TP + FP)
•TP: True Positive (실제 양성인 경우를 양성으로 정확히 예측)
•FP: False Positive (실제 음성인 경우를 양성으로 잘못 예측)

•**재현율**은 실제로 '양성'인 케이스를 얼마나 잘 찾아내는지 평가하는 데 중요합니다. 예를 들어, 질병 진단 시스템에서 모든 질병 환자를 찾아내는 것이 중요할 때, 재현율을 높이는 것이 필요합니다.

•TP / (TP + FN)
•FN: False Negative (실제 양성인 경우를 음성으로 잘못 예측)

F1-스코어 (F1-Score)는 정밀도와 재현율의 조화 평균으로, 두 지표 간의 균형을 평가합니다.

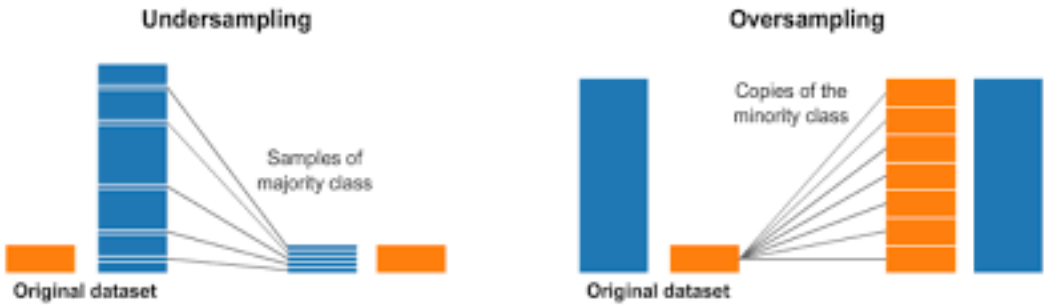
•2 × (정밀도 × 재현율) / (정밀도 + 재현율)



소수 클래스인 구매를 증강시켜 데이터 불균형 문제를 해결함

SMOTE 데이터 증강 기법 활용

```python
# 데이터 스케일링
scaler = StandardScaler()
X_scaled = scaler.fit_transform(X)

# 데이터 불균형 처리 (SMOTE)
sm = SMOTE(random_state=42)
X_resampled, y_resampled = sm.fit_resample(X_scaled, y)

# 데이터 분할
X_train, X_test, y_train, y_test = train_test_split(X_resampled, y_resampled, test_size=0.2, random_state=42)

# 로지스틱 회귀 모델 학습
model = LogisticRegression(max_iter=1000, random_state=42)
model.fit(X_train, y_train)

# 예측
y_pred = model.predict(X_test)
y_pred_proba = model.predict_proba(X_test)[:, 1]

# 모델 평가
accuracy = accuracy_score(y_test, y_pred)
precision = precision_score(y_test, y_pred)
recall = recall_score(y_test, y_pred)
f1 = f1_score(y_test, y_pred)
roc_auc = roc_auc_score(y_test, y_pred_proba)

print("Accuracy:", accuracy)
print("Precision:", precision)
print("Recall:", recall)
print("F1-Score:", f1)
```



event_type	count
view	793748
cart	54035
purchase	37346

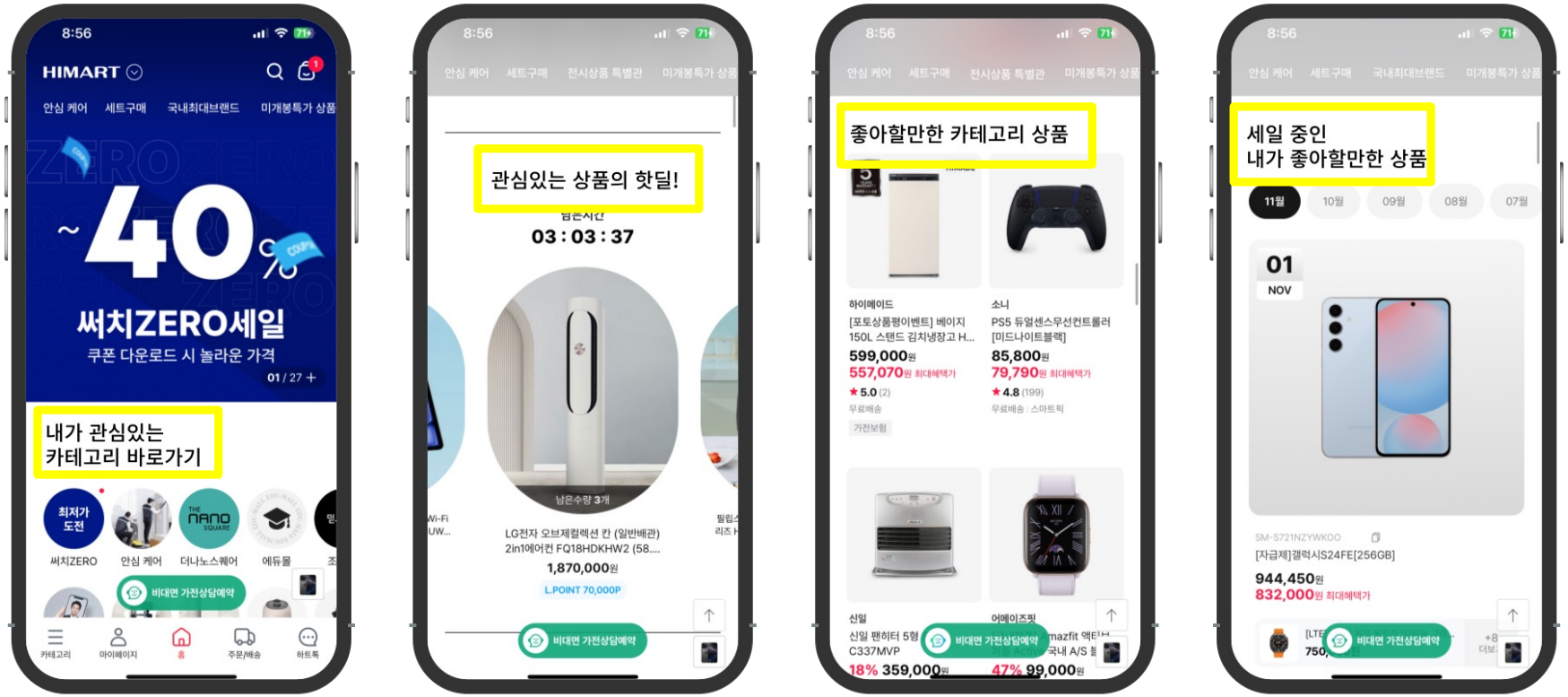
소수 클래스(purchase)를 증강하여 **데이터 불균형 문제**를 해결하는 데 매우 유용

SMOTE를 사용하면 **구매 이벤트 데이터**를 증강하여 모델이 **구매 예측**을 더 잘 학습할 수 있도록 도움

```
Accuracy: 0.8627364249190538
Precision: 0.8901028115290374
Recall: 0.8286456741057778
F1-Score: 0.8582754841066862
```

구매 예측을 통해 구매를 하겠다고 한 소비자들에게 맞춤형 추천을 제시할 것임

최종 프로토타입



추천 시스템 고도화

개인 맞춤형 홈 화면 UI/UX 개선

→ 전환율 ↑