CJ더마켓 프라임 회원 예측 모델

대상이조

박시현

손민규

임성연

정예린

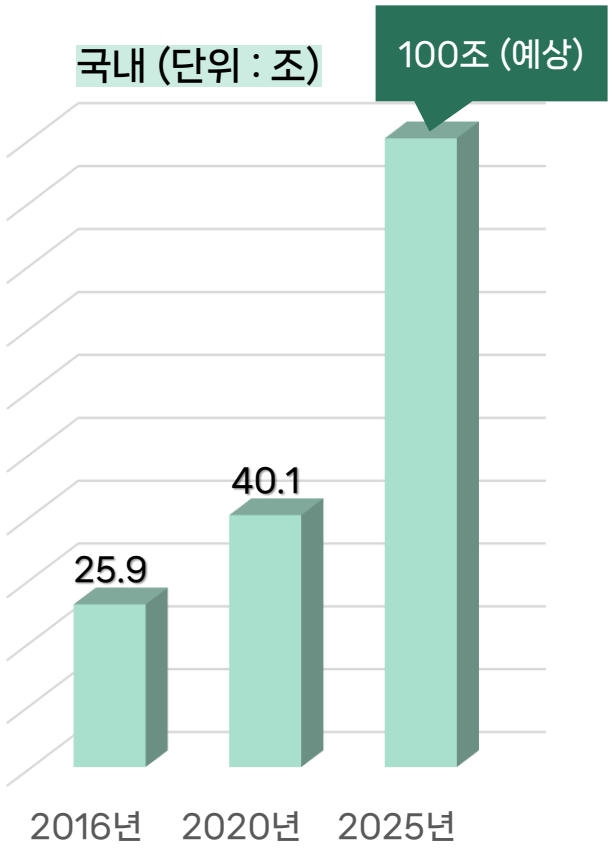
Contents

01

분석 목적 및 필요성

- 구독경제 시장의 성장 및 분석 목적

분석 목적 및 필요성 | 구독경제 시장의 성장 및 분석 목적



자료 : KT경제경영연구소

구독 경제 시장의 성장

✓ 구독 경제 시장

- 전 세계 구독 경제 시장 연평균 68%씩 성장
- 국내 구독 경제 시장 규모 2020년 약 55% 증가
- 2025년 100조원까지 확대 예상

✓ 상위 플랫폼 주요 혜택

- 쿠팡 : 배송 시스템
- 네이버 : 포인트 적립
- 스마일클럽 : 즉시 적립 및 할인 쿠폰

분석 목적

✓ CJ더마켓 문제 진단

- 인지도가 낮음
- 더프라임 제도의 빈번한 혜택 변경
- 주요한 혜택의 부진 → 불안정

✓ 분석 목적

- 프라임 회원 모델을 이용한 회원 그룹화
- 그룹별 차별화된 마케팅 전략 수립
- 충성 고객의 구매 형태 및 니즈 파악

02

데이터 전처리

- 인코딩
- 변수 추가
- 데이터 축약
- 데이터 분리

데이터 전처리 | 인코딩

성별

1 - 남성

0 - 여성

나이대

- 순서가 유의미
  → 레이블 인코딩
- 추가 전처리 없이 (1~6)

구매일자

1. 평일, 주말 및 공휴일 (0,1)
   - 2번보다 현저히 적은 변수
   - 정보 소실 우려

2. 월,화,수,목,금,토,일 (0~6)
   - 많은 변수 추가
   - 모델 학습 저해 우려

카테고리

1. 카테고리별 제품 크롤링 (총 12개의 카테고리)

2. 크롤링을 이용한 인코딩 (0~11)

3. 크롤링에 포함되지 않은 제품 인코딩
   - 21904개의 데이터 새로운 인코딩 필요
   - 각 카테고리별 키워드 선정 후 인코딩

4. 12개의 카테고리 4개로 축약
   - 과다한 변수 추가 우려
   - 요리, 간식, 식사, 기타 : 4개의 카테고리로 축약

데이터 전처리 | 변수 추가

이벤트 제품

- 이벤트 참여 횟수가 높을수록 충성 고객일 확률이 높다는 연구 결과
- 이벤트 제품 구매 횟수 추가
- 제품명에 '[]'가 포함되면 이벤트 제품임을 확인
- 식물성, 냉동, 배송, 유산균, 눈건강, 피부건강, 1BOX, 쿡킷은 이벤트명에서 제외

카테고리 종류 수

- 구매한 카테고리가 다양할수록 충성 고객일 확률이 높다는 연구 결과
- 4개의 카테고리로 축약하기 전의 기존 12개의 카테고리로 반영
- 구매 제품의 카테고리 개수 변수 추가

더세페

- 더세페
  : 매달 1일 ~ 10일 할인행사 기간
  → 해당 기간에 주문수량 대비 주문금액 낮을 것으로 예상
- (총 주문금액)/(총 주문수량) 변수 추가

분석 목적 및 필요성 | 데이터 축약

```
# 주문번호 별로 합치기
# gender, age_grp, weekday, prim_yn, employee_yn -> 동일한 값
# total_qty, total_amt, event_product -> sum
# category -> 카테고리 리스트
# num_category -> nunique
# amt/qty -> mean
cj_grp = cj_new.groupby('scd').agg({'gender': pd.Series.mode,
                                    'age_grp': pd.Series.mode,
                                    'weekday': pd.Series.mode,
                                    'total_qty': 'sum',
                                    'total_amt': 'sum',
                                    'category': lambda x: ",".join(set(x)),
                                    'event_product': 'sum',
                                    'num_category': pd.Series.nunique,
                                    'amt/qty': 'mean',
                                    'prime_yn': pd.Series.mode,
                                    'employee_yn': pd.Series.mode})
cj_grp
```

✓ 데이터 축약 필요성

- 한 고객이 여러 제품 구매 시 데이터 중복 문제 발생
- 주문번호별 축약 필요

✓ 축약 방식

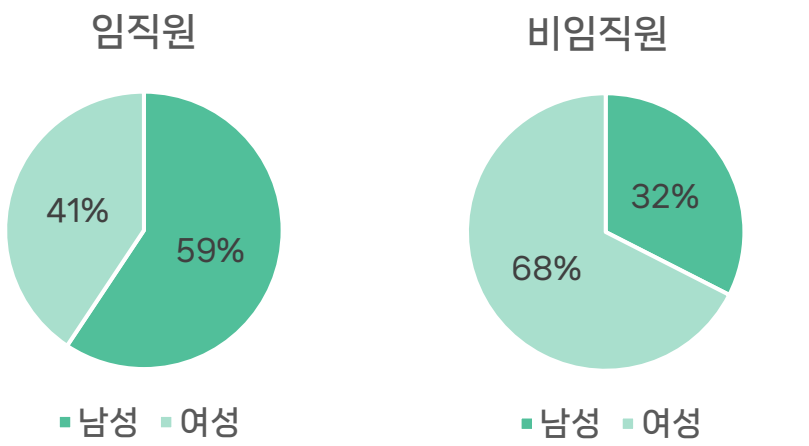
- 성별, 나이대, 주문일자 → mode
- 구매금액, 구매수량, 이벤트 제품 → sum
- 주문금액 대비 주문수량 → mean
- 구매 카테고리 수 → nunique

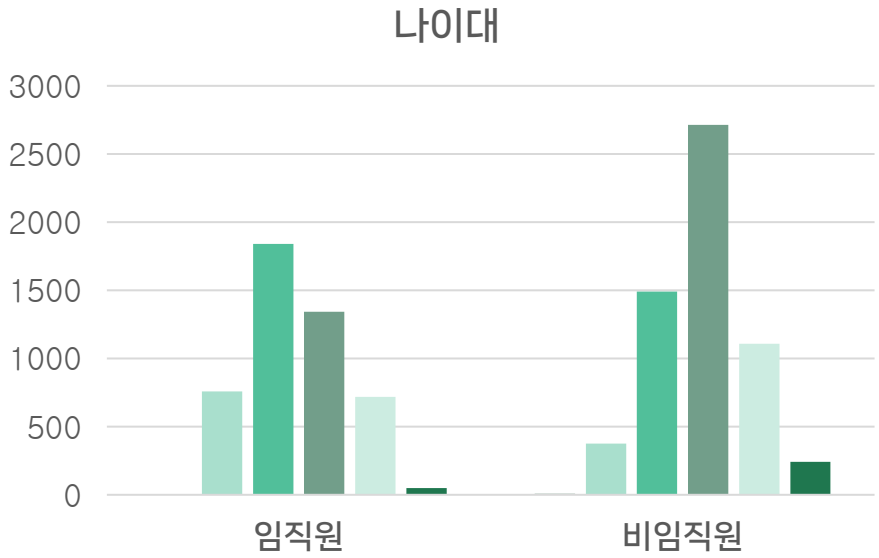
✓ 카테고리 축약 방식

1. 원-핫 인코딩: 개수 상관없이 0,1로 구매여부만 표현
2. 원-핫 인코딩: 가장 많이 구매한 카테고리만 표현
3. 카테고리별 구매수량 표현

데이터 전처리 | 데이터 분리

[실제 데이터 분석 결과]





✓ 임직원, 비임직원 데이터 분리의 필요성

- 각 데이터의 성격이 다름
  - 성비 차이 : 임직원은 남성, 비임직원은 여성이 우세
  - 나이대 차이
  - 구매형태 차이 : 임직원 → 적극적 소비
    비임직원 → 상대적으로 소극적 소비
- 주문수량, 주문금액, 성별, 나이대에서 다른 성격을 보임
- 모델의 성능 저하 우려

03 모델링

- 모델링 시도
- 가설 검정
- 파라미터 조정

03. 모델링 | 모델링 시도

CJ 더마켓의 프라임 회원 여부 예측 모델링 (prime_yn = 0/1) → 이진 분류 모델 적용

- Decision Tree : 트리 기반 학습
- Random Forest : 트리 기반 학습
- Support Vector Machine : 커널 기반 학습
- Gradient Boosting : 부스팅 알고리즘
- XGBoost : 부스팅 알고리즘

03. 모델링 | 모델링 시도

Decision Tree

	Accuracy	F1-score
임직원	0.5267	0.5915
비임직원	0.5866	0.5081

Random Forest

	Accuracy	F1-score
임직원	0.6008	0.6954
비임직원	0.6123	0.4859

SVM

	Accuracy	F1-score
임직원	0.5605	0.6004
비임직원	0.4839	0.3548

✓ Gradient Boosting

	Accuracy	F1-score
임직원	0.6156	0.7221
비임직원	0.6239	0.4408

✓ XGBoost

	Accuracy	F1-score
임직원	0.6068	0.6888
비임직원	0.6316	0.5280

03. 모델링 | 가설 검정

데이터 축약

	precision	recall	f1-score	support
0	0.63	0.23	0.33	564
1	0.64	0.91	0.75	850
accuracy			0.64	1414
macro avg	0.63	0.57	0.54	1414
weighted avg	0.63	0.64	0.58	1414

정확도: 0.6379, 정밀도: 0.6397, 재현율: 0.9106, F1:0.7515

	precision	recall	f1-score	support
0	0.67	0.76	0.71	1013
1	0.62	0.52	0.56	769
accuracy			0.65	1782
macro avg	0.65	0.64	0.64	1782
weighted avg	0.65	0.65	0.65	1782

정확도: 0.6532, 정밀도: 0.6171, 재현율: 0.5176, F1:0.5629

✓ 주문번호별 축약 시 카테고리 데이터

✓ F1-score 가장 높은 방식 선택

앞서 축약했던 4개의 카테고리에 대해

주문번호별 제품 카테고리 여부를 0과 1로 표현

↓

카테고리 변수 (ctg_0 ~ctg_3) 추가

03. 모델링 | 가설 검정

설 연휴 데이터

[설 연휴 데이터 제거한 데이터 성능]

분류예측 레포트:

	precision	recall	f1-score	support
0	0.60	0.33	0.42	340
1	0.64	0.85	0.73	489
accuracy			0.63	829
macro avg	0.62	0.59	0.58	829
weighted avg	0.62	0.63	0.60	829

f1 score:
0.7314487632508834

분류예측 레포트:

	precision	recall	f1-score	support
0	0.67	0.76	0.71	802
1	0.60	0.48	0.53	580
accuracy			0.64	1382
macro avg	0.63	0.62	0.62	1382
weighted avg	0.64	0.64	0.64	1382

f1 score:
0.5310410697230182

✓ 설 연휴 데이터의 포함 여부 파악

설 연휴 데이터 제거 후 f1-score 확인

↓

임직원, 비임직원 데이터 모두

성능이 오히려 감소하여

설 연휴 데이터 포함

03. 모델링 | 파라미터 조정

1. 임직원

파라미터 조정 이전

```
# Gradient Boosting
from sklearn.ensemble import GradientBoostingClassifier
from sklearn.metrics import accuracy_score
from sklearn.metrics import classification_report
from sklearn.metrics import f1_score

# 모델 학습, 파라미터 조정
model_1 = GradientBoostingClassifier()
model_1.fit(X_train, y_train)

# 예측
pred = model_1.predict(X_valid)

# 정확도
accuracy = accuracy_score(y_valid, pred)
print("Accuracy:", accuracy)

# 평가
cfreport = classification_report(y_valid, pred)
print("분류예측 레포트:₩n", cfreport)

f1 = f1_score(y_valid, pred)
print("f1 score:₩n", f1)
```

```
Accuracy: 0.6522476675148431
분류예측 레포트:
              precision    recall  f1-score   support

           0       0.67      0.25      0.37       471
           1       0.65      0.92      0.76       708

    accuracy                           0.65      1179
   macro avg       0.66      0.59      0.56      1179
weighted avg       0.66      0.65      0.60      1179

f1 score:
 0.7599531615925059
```

파라미터 조정 이후

```
from sklearn.model_selection import GridSearchCV

gradient_boosting = GradientBoostingClassifier()

param_grid = {
    'max_depth': range(3,10),
    'learning_rate': [0.1, 0.01, 0.001],
    'n_estimators': [100, 500, 1000]
}

# 그리드 서치
grid_search = GridSearchCV(estimator=gradient_boosting, param_grid=param_grid, cv=3, scoring='accuracy')
grid_search.fit(X_train, y_train)

# 최적의 파라미터
print("Best Parameters:", grid_search.best_params_)
print("Best Score:", grid_search.best_score_)
```

```
Accuracy: 0.6675148430873622
분류예측 레포트:
              precision    recall  f1-score   support

           0       0.60      0.49      0.54       471
           1       0.70      0.79      0.74       708

    accuracy                           0.67      1179
   macro avg       0.65      0.64      0.64      1179
weighted avg       0.66      0.67      0.66      1179

f1 score:
 0.740053050397878
```

→ Grid Search로 파라미터를 조정하면
f1-score는 오히려 감소하는 것을 확인함

03. 모델링 | 파라미터 조정

2. 비임직원

파라미터 조정 이전

```
import xgboost as xgb

# XGBoost
xgb = xgb.XGBClassifier()
xgb.fit(X_train, y_train)

# 테스트 데이터로 예측 수행
pred = xgb.predict(X_valid)

# 정확도 평가
accuracy = accuracy_score(y_valid, pred)
print("Accuracy:", accuracy)
# 모델 예측 및 평가
cfreport = classification_report(y_valid, pred)
print("분류예측 레포트:₩n", cfreport)

f1 = f1_score(y_valid, pred)
print("f1 score:₩n", f1)
```

```
분류예측 레포트:
              precision    recall  f1-score   support

           0       0.67      0.75      0.71       861
           1       0.59      0.49      0.54       624

    accuracy                           0.64      1485
   macro avg       0.63      0.62      0.62      1485
weighted avg       0.63      0.64      0.63      1485

f1 score:
 0.5356521739130435
```

파라미터 조정 이후

```
from sklearn.model_selection import GridSearchCV
import xgboost as xgb

# XGBoost 모델 초기화
xgb_model = xgb.XGBClassifier()

param_grid = {
    'max_depth': range(3,10),
    'learning_rate': [0.1, 0.01, 0.001],
    'n_estimators': [100, 500, 1000]
}

# 그리드 서치
grid_search = GridSearchCV(estimator=xgb_model, param_grid=param_grid, cv=3, scoring='accuracy')
grid_search.fit(X_train, y_train)

# 최적의 파라미터
print("Best Parameters:", grid_search.best_params_)
print("Best Score:", grid_search.best_score_)
```

```
Best Parameters: {'learning_rate': 0.01, 'max_depth': 7, 'n_estimators': 500}
```

```
# XGBoost
import xgboost as xgb

# 모델 학습, 파라미터 조정
model_3 = xgb.XGBClassifier(learning_rate=0.1, max_depth=5, n_estimators=500)
model_3.fit(X_train, y_train)

# 예측
pred = model_3.predict(X_valid)

# 정확도
accuracy = accuracy_score(y_valid, pred)
print("Accuracy:", accuracy)

# 평가
cfreport = classification_report(y_valid, pred)
print("분류예측 레포트:₩n", cfreport)

f1 = f1_score(y_valid, pred)
print("f1 score:₩n", f1)
```

```
Accuracy: 0.6397306397306397
분류예측 레포트:
              precision    recall  f1-score   support

           0       0.67      0.75      0.71       861
           1       0.59      0.49      0.53       624

    accuracy                           0.64      1485
   macro avg       0.63      0.62      0.62      1485
weighted avg       0.63      0.64      0.63      1485

f1 score:
 0.5335658238884046
```

→ Grid Search로 파라미터를 조정하면
성능이 크게 차이가 없어 하이퍼 파라미터 적용함

03. 모델링 | 최종 결과

Gradient Boosting

	Accuracy	F1-score
임직원	0.6480	0.7519

XGBoost

	Accuracy	F1-score
비임직원	0.6397	0.5336

Team score

F1-score
0.6262

04 한계점 및 개선점

04. 한계점 및 개선점

[그림 1]

	precision	recall	f1-score	support
0	0.69	0.26	0.38	471
1	0.65	0.92	0.76	708

	precision	recall	f1-score	support
0	0.66	0.83	0.73	861
1	0.63	0.41	0.49	624

[그림 2]

	Accuracy	F1-score
임직원	0.6735	0.7679
비임직원	0.6431	0.5383

1. 임직원, 비임직원의 클래스별 f1-score 성능 차이
   - [그림 1] 클래스 0/1에 대한 예측 성능 차이가 큼

   → 각 데이터의 특성을 파악하여 모델링을 진행한다면 성능이 향상될 것으로 예상

2. 카테고리 분류 과정 오류
   - 카테고리를 분류하는 과정에서 전처리가 일부 잘못됨
   - [그림 2] 수정 결과, 성능 미미하게 향상 → 변수 처리 방식 고려

3. 주문번호별 축약 시 데이터 손실 문제
   - 모델이 학습 가능한 데이터의 절대적인 양이 크게 감소함

   → 주문번호를 축약하지 않고 전처리를 진행한다면 성능이 향상될 것으로 예상

05

모델 활용방안 및 기대효과

05. 모델 활용방안 및 기대효과

CJ더마켓의 충성 고객 확보 & 인지도 및 경쟁력 증진

예측 결과

일반 회원 → 프라임 회원

- ✓ 집중적인 프로모션
- ✓ 적극적인 타겟 마케팅 진행

일반 회원 → 일반 회원

- ✓ 더프라임 회원 제도 홍보

프라임 회원 → 프라임 회원

- ✓ 이미 충성고객 → 유지 전략
- ✓ 만족도 조사 실시

프라임 회원 → 일반 회원

- ✓ 이탈 가능성이 높은 고객 집단
- ✓ 이탈 원인 파악 및 보완

감사합니다

대상이조

박시현

손민규

임성연

정예린