신용카드 연체 여부 예측

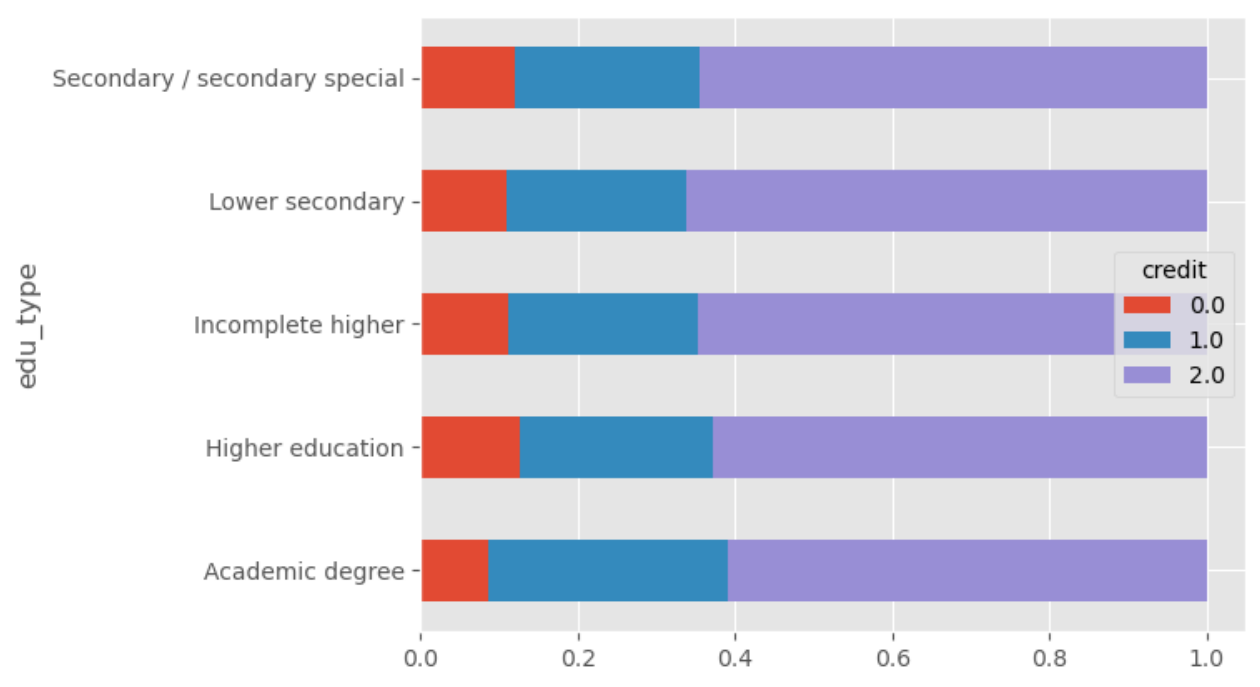
Kuggle 9기 2조

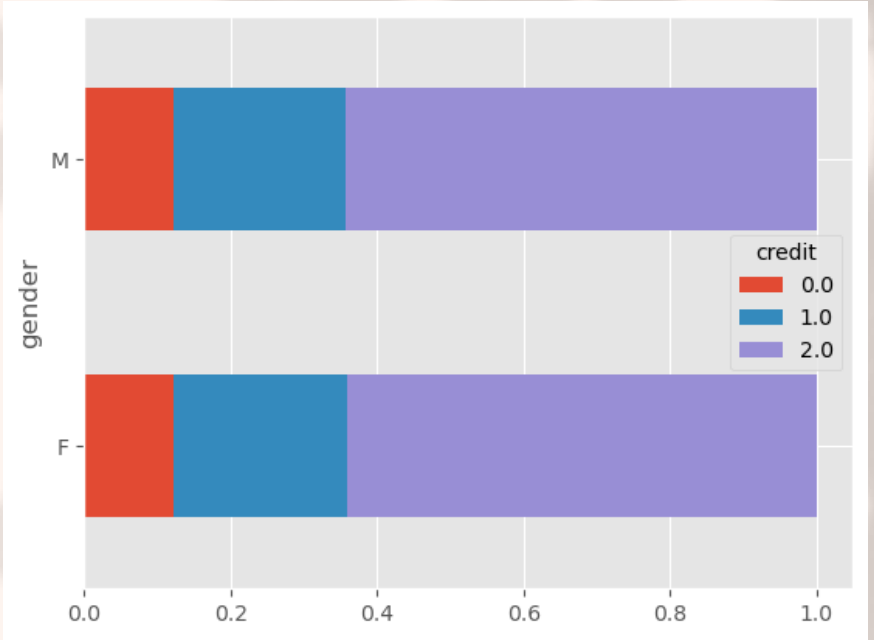
주제 선정

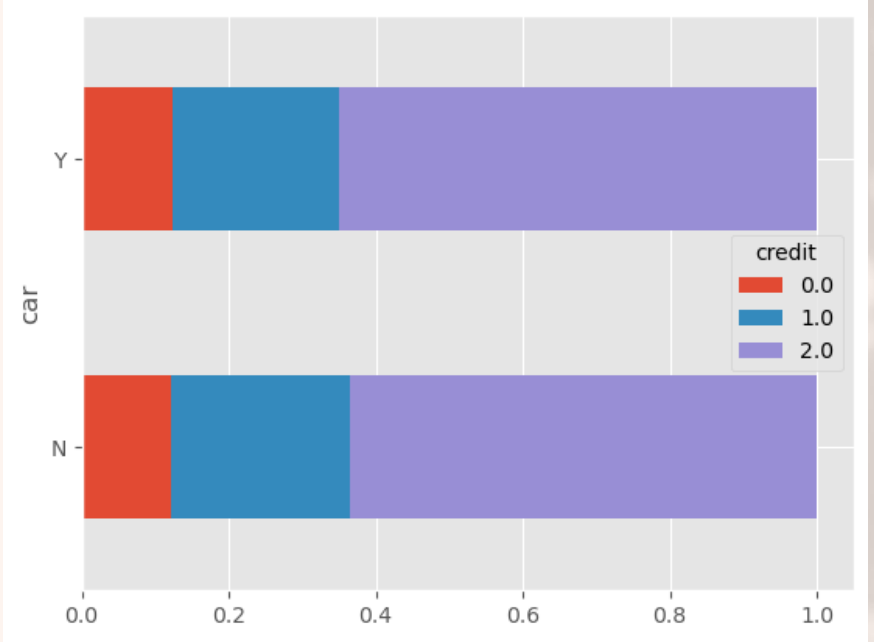
신용카드 이용자의 여러 정보를 토대로 신용카드 연체 여부를 예측하는 모델을 만들고자 합니다.

이 데이터셋에서는 기존 변수들을 토대로 파생 변수를 생성하는 작업이 매우 중요하기 때문에, 피처 엔지니어링의 기초를 배울 수 있는 좋은 기회가 될 것이라 생각했습니다.

데이터 시각화


Secondary / secondary special
Lower secondary
Incomplete higher
Higher education
Academic degree
edu_type
credit
0.0
1.0
2.0
0.0
0.2
0.4
0.6
0.8
1.0


M
F
gender
credit
0.0
1.0
2.0
0.0
0.2
0.4
0.6
0.8
1.0


Y
N
car
credit
0.0
1.0
2.0
0.0
0.2
0.4
0.6
0.8
1.0

범주형 피처 별 경향

대다수의 피처에서 특정 집단이 유의할 정도로 높거나 낮은 연체도를 보이지는 않음

다만, 자녀 수, 교육 정도, 주거 형태 등 일부 소수 피처에서 유의미한 차이가 존재함

특히, 직업 분류 피처에서는 각 집단별로 유의미한 연체도의 차이가 발견됨

가령 기업 인사팀 직원의 경우, 연체도가 낮은 고객의 비율이 매우 높게 나타남

또한, 이렇게 집단 별 유의한 차이점이 존재했던 피처에서,

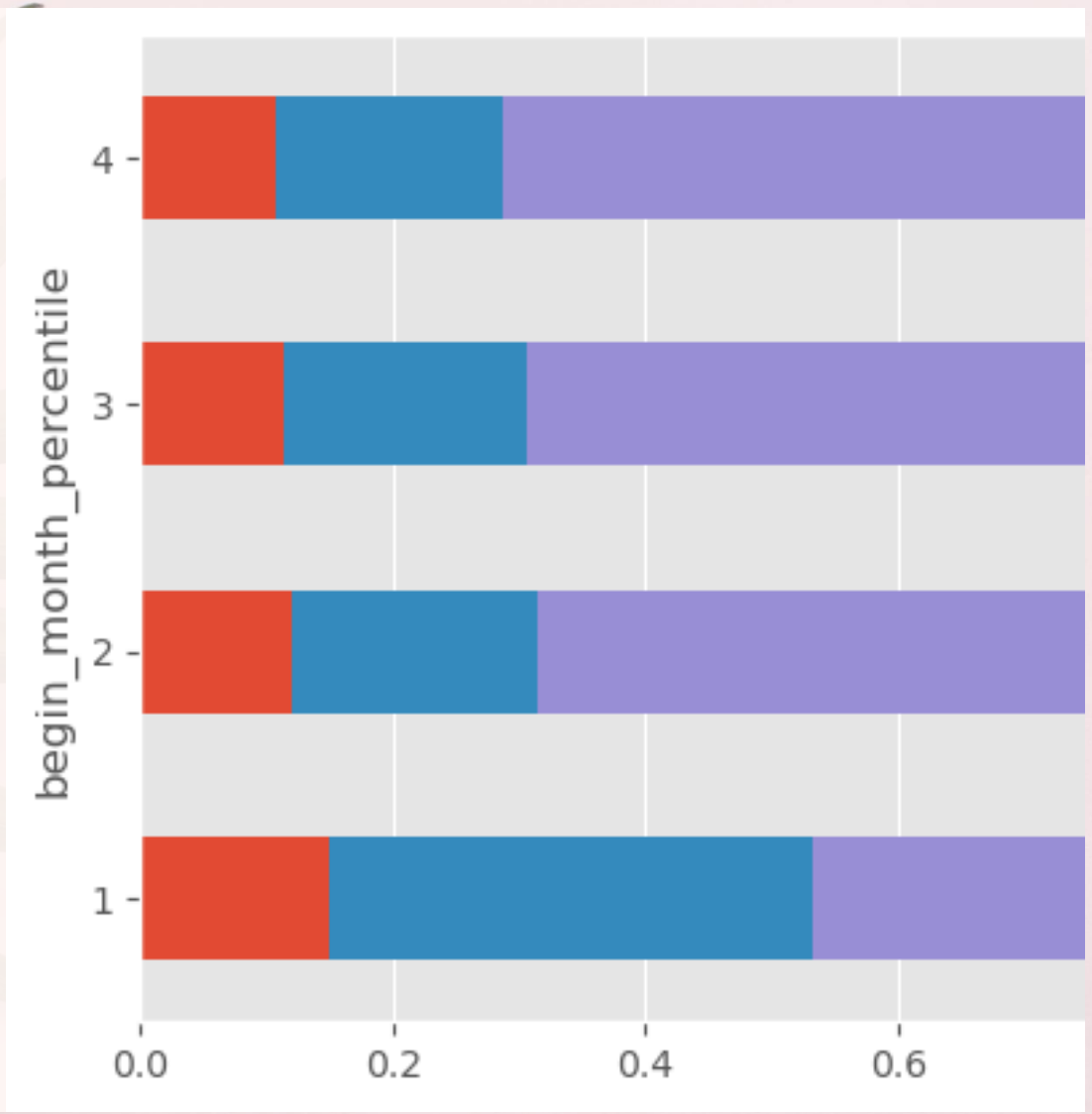
특이하게도 연체도가 낮은 고객 비율이 높은 집단이라 할 지라도, 연체도가 낮은 고객의 비율이 타 집단에 비해 낮지는 않았다는 점

연속형 피처

연속형 피처의 경우 사분위수를 기준으로 총 4개의 집단으로 구별한 후 시각화 하였습니다.

그 결과, 소득 수준, 근로 일수, 생후 일수에서는 유의미한 차이가 없었으나,

신용카드 가입 후 경과일수가 하위 25%인 집단에서 유의미할 정도로 연체도가 높은 고객의 비율이 낮게 나타났습니다.



기본 데이터 전처리

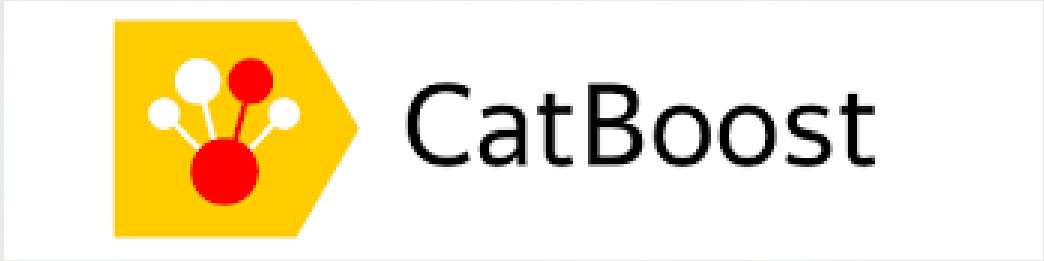
① 이상치 / 결측치 처리

② 음수 값을 가지는 피처 양수화

③ 무의미한 피처 제거

④ 범주형 피처 – 정수 인코딩

사용 모델



개요

Catboost는 lightGBM과 XGBoost와 마찬가지로, 트리 기반의 부스팅 모델입니다.

범주형 변수가 많은 데이터셋에 강하다는 장점이 있습니다.

선택 이유

- 범주형 변수가 많이 포함된 이 데이터셋에 적합하다고 판단

- 사전 테스트에서도 가장 성능이 좋게 나타남

파생 변수 생성

파생 변수 생성은 이 데이터셋을 다루는데 있어 가장 중요한 부분입니다.

기존의 변수만으로는 나타낼 수 없는 새로운 정보에 대한 인사이트를 제공합니다.

파생 변수

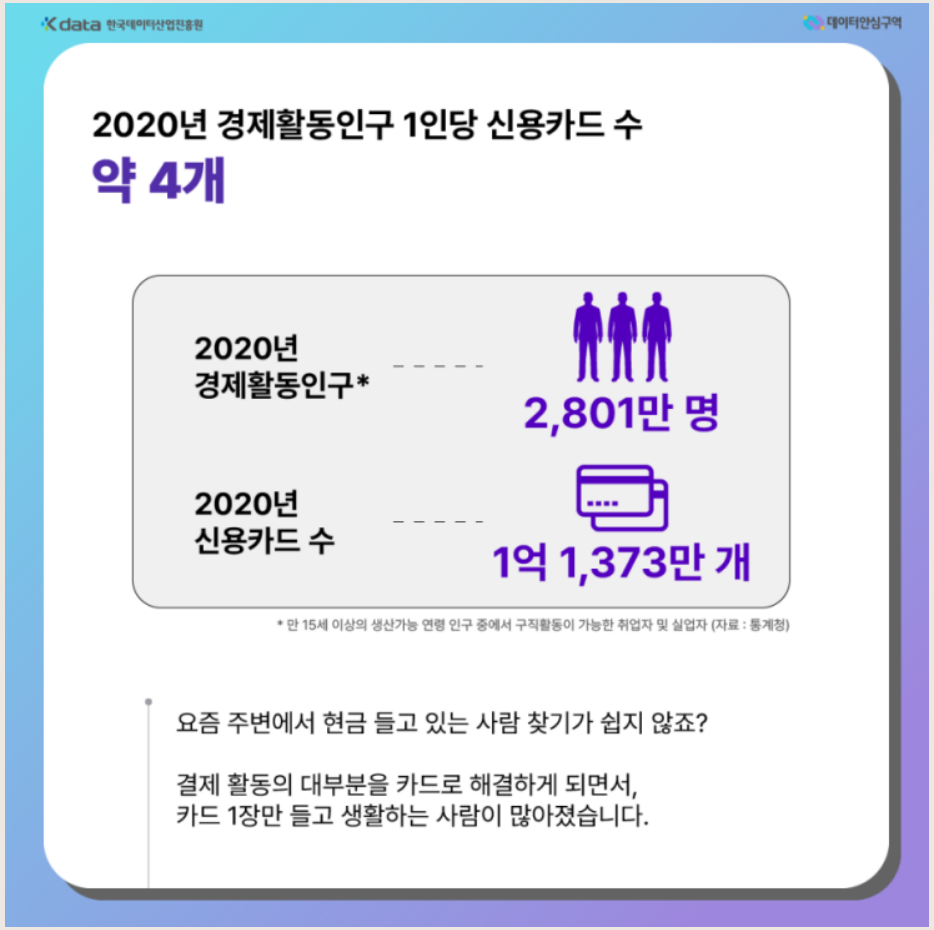
식별 ID

옆의 자료에서 알 수 있듯,
경제활동인구 1인당 신용카드 수는 무려 4장에 달합니다.

이는 데이터셋 상에서 서로 다른 신용카드로 존재하는 두 행이, 실제로는 동일인물이 소지한 카드일 수 있다는 이야기입니다.

따라서 각 데이터 행에 식별 ID를 부여합니다.

식별 ID는 수입, 고용일, 생후일수, 직업 유형을 문자 형태로 나열한 후, 정수로 인코딩합니다.



파생 변수

취업 전 생후 개월 수

취업 전 생후 개월 수는 생후 개월 수에서 취업 월 수를
뺀 피처입니다.
이 피처를 토대로 고객의 취업 시기를 파악할 수 있습니다.

취업 전 생후 개월 수가 적은 사람은 보다 적은 교육을 받고 곧바로
취업 시장에 뛰어들었을 가능성이 높습니다.

반대로 취업 전 생후 개월 수가 많은 사람은 교육 수준이 높거나,
혹은 돈을 벌어야 한다는 금전적 압박에 적게 시달렸을 가능성이
높습니다.

파생 변수

근로기간 대비 수입

근로기간 대비 수입 피처는 수입 피처를 근로 기간으로 나누어 얻었습니다.

일반적으로 한 직장에서 오래 일하고, 경력을 쌓을 수록,
더 높은 임금을 받게 되는 경향이 있습니다.

이 피처는 이러한 임금 상승률에 대한 정보를 제공해줄 것입니다.

파생 변수

가족구성원 수 대비 수입

가족구성원 수 대비 수입 피처는 수입 피처를 가족 구성원 수로 나누어 얻었습니다.

오른쪽 그림의 최저생계비를 보면 알 수 있듯,
부양해야 하는 가구원 수가 늘어날수록 더 많은 생계비가 필요합니다.

가령 절대적인 수입 자체는 더 많더라도, 그만큼 부양해야 하는 가구원 수가 많다면 결코 풍족하다고 할 수 없을 것입니다.

부양가족	최저생계비
1인 가구	1,246,735원
2인 가구	2,073,693원
3인 가구	2,660,890원
4인 가구	3,240,578원
5인 가구	3,798,413원

하이퍼 파라미터 튜닝

OPTUNA

하이퍼 파라미터 튜닝을 하는 경우 보통 그리드 서치 혹은 베이지안 최적화를 먼저 떠올리기 마련입니다. 하지만 OPTUNA는 더 간편하고 효과적으로 하이퍼 파라미터 튜닝을 할 수 있도록 강력한 도구를 제공하는 프레임워크입니다.

가령 하이퍼 파라미터에 대한 이해를 토대로 미리 변수 목록을 지정해주면 모든 경우의 수에 대해 테스트를 하여 가장 성능이 좋은 조합을 찾는 그리드 서치와 달리, 다소 포괄적인 변수의 범위를 지정할 수 있으며, 튜닝 결과의 시각화나 변수 중요도 정보도 제공합니다.

Optuna 사용

```python
def objective(trial: Trial) -> float:
    params_cat = {"iterations": trial.suggest_int("iterations", 400, 1600),
                  "early_stopping_rounds": trial.suggest_int("early_stopping_rounds", 3, 9),
                  "learning_rate": trial.suggest_float("learning_rate", 0.01, 0.1),
                  "eval_metric": "MultiClass",
                  "l2_leaf_reg": trial.suggest_loguniform("l2_leaf_reg", 1e-6, 3e-5),
                  "depth": trial.suggest_int("depth", 2, 16),
                  "rsm": trial.suggest_float("rsm", 0.5, 1.0),
                  "grow_policy": trial.suggest_categorical("grow_policy",["SymmetricTree","Depthwise","Lossguide"])
                  }

    x = data.drop('credit',axis=1)
    y = data['credit']

    model = CatBoostClassifier(**params_cat)
    log_score = cross_val_score(estimator=model, X=x, y=y, cv=5, scoring='neg_log_loss', n_jobs=-1).mean()

    return log_score
```

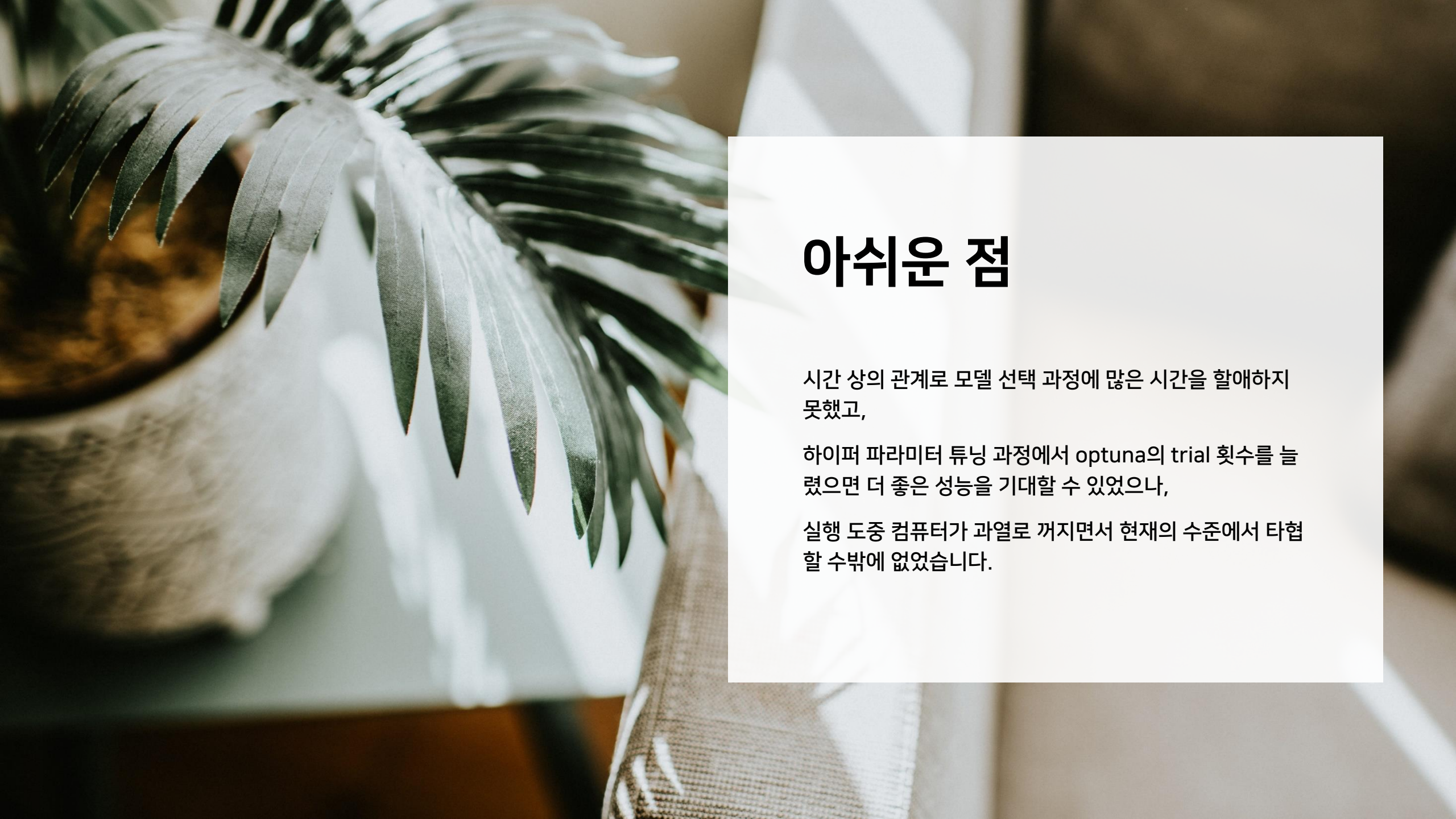
최종 결과

기본값 모델 & 정수 인코딩만 적용한 데이터셋

log_loss : 0.8066

파생 변수 생성 & 튜닝된 모델

log_loss : 0.7309



아쉬운 점

시간 상의 관계로 모델 선택 과정에 많은 시간을 할애하지 못했고,

하이퍼 파라미터 튜닝 과정에서 optuna의 trial 횟수를 늘렸으면 더 좋은 성능을 기대할 수 있었으나,

실행 도중 컴퓨터가 과열로 꺼지면서 현재의 수준에서 타협할 수밖에 없었습니다.

감사합니다.