신용카드 사용자 연체 예측

Dacon - 신용카드 사용자 연체 예측 AI 경진대회

경영학과 이주희

CONTENTS

01

About Project

프로젝트 설명

신용카드 사용자 연체 예측

신용카드 사용자들의 개인 신상정보

신용카드 사용자의 **신용등급**을 예측

신용 등급

- 신용등급은 신청자의 향후 **채무불이행**과 신용카드 대급 **연체 가능성**을 내포함
- 신용카드 신청자가 제출한 개인정보와 데이터를 활용해 신용등급을 산정

데이터 설명

Feature data

신용카드 사용자들의 개인 신상정보

성별, 차량 부동산 소유여부, 연간 소득, 교육 수준 등.. 19개의 컬럼

Binary Category: 성별, 차량, 부동산, 핸드폰, 전화, 이메일 소유 여부

Multi Category: 소득 분류, 교육 수준, 결혼 여부, 생활 방식, 직업 유형

Numerical value: 연간 소득, 가족 규모, 자녀 수, 신용카드 발급 월 수, 출생일, 고용일수

Tatarget data

credit

사용자의 신용카드 대금 연체를 기준으로 한 신용도

0, 1, 2 세 개의 라벨로 구성

신용등급이 낮을 수록 신용이 높음을 의미

Feature Data 상세

Binary Category

gender	성별
car	차량 소유 여부
reality	부동산 소유 여부
FLAG_MOBIL	핸드폰 소유 여부
work_phone	업무용 전화 소유 여부
phone	전화 소유 여부
email	이메일 소유 여부

Multi Category

income_type	소득 분류
edu_type	교육 수준
family_type	결혼 여부
house_type	생활 방식
occyp_type	직업 유형

Numerical value

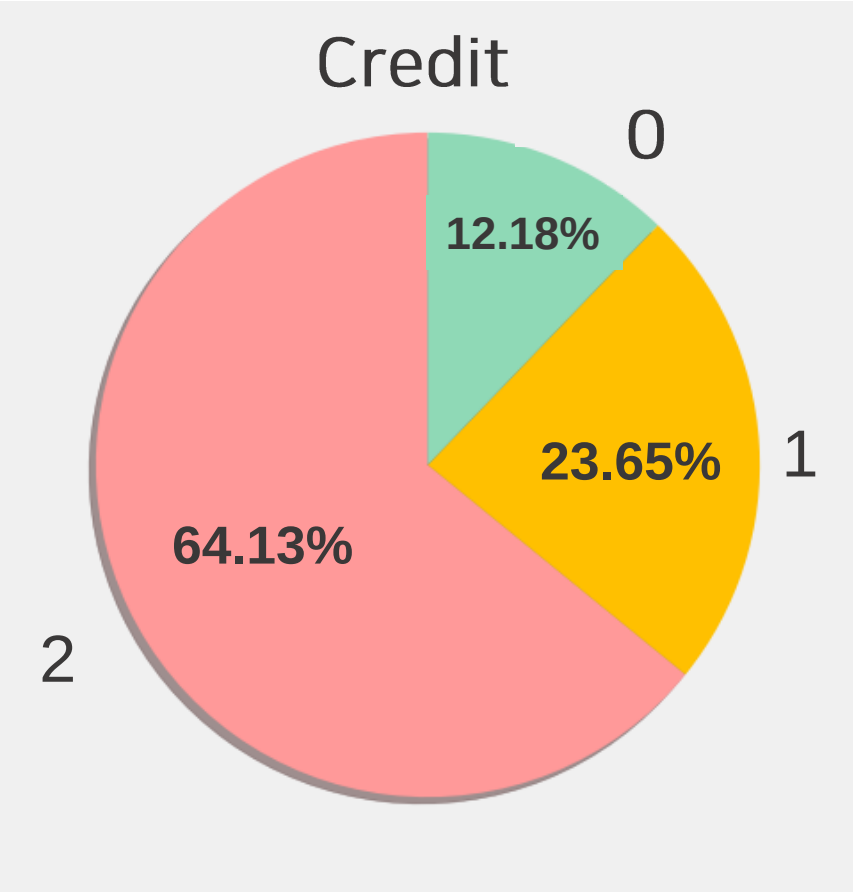
income_total	연간 소득
family_size	가족 규모
child_num	자녀 수
begin_month	신용카드 발급 월 수
DAYS_BIRTH	출생일
DAYS_EMPLOYED	고용일수

02

EDA

탐색적 데이터 분석

Target data 확인



Credit - target data

credit이 낮을수록 높은 신용을 의미

(credit=0 은 가장 높은 신용 등급)

신용 등급 비율

신용이 낮은 사용자가 매우 많음

신용등급이 2인 사용자는 0인 사용자의 약 5배에 달함

→ Label의 불균형이 존재

Categorical Variable EDA

occyp_type : 직업유형

train, test 데이터에 결측치가 존재하는 것을 확인

train 데이터의 결측치 비율은 전체 데이터의 37%

→ 해당 row나 feature를 drop하는 것은 좋지 않다고 판단

▶ 특정 값으로 결측치를 대체

FLAG_MOBIL: 핸드폰 소유 여부

train 데이터의 모든 사용자가 핸드폰을 소지하고 있음

의미 없는 feature라고 판단

▶ 해당 feature 제거 필요

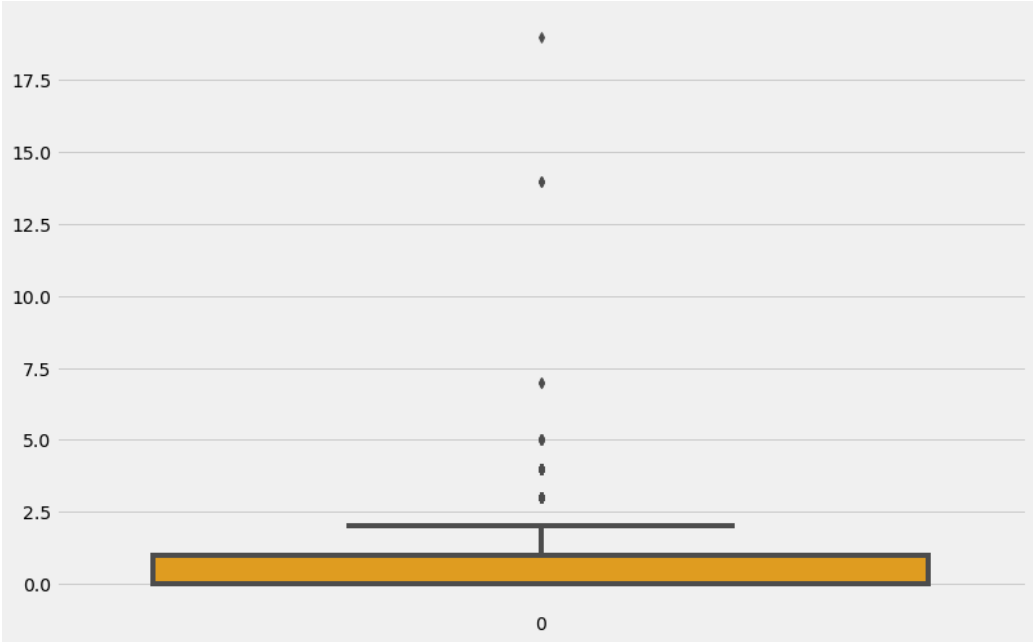
Numerical Variable EDA

child_num과 family_size가 지나치게 큰 값을 갖는 경우 존재

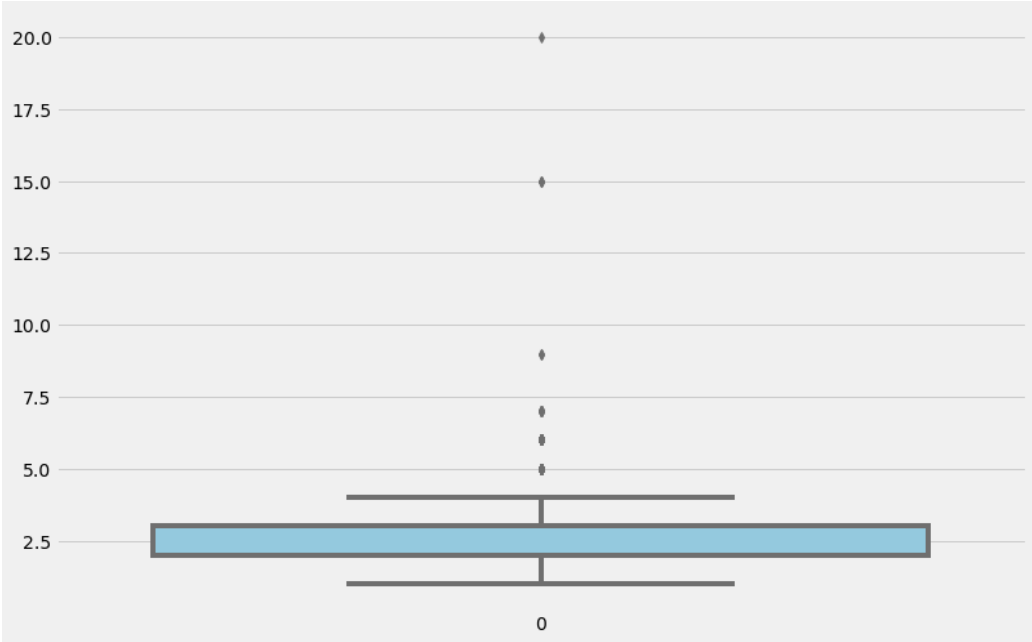
child_num이 6 이상인 데이터 = family_size가 7보다 큰 데이터

→ 이상치 확인

child_num (자녀 수)



family_size (가족 규모)



Numerical Variable EDA

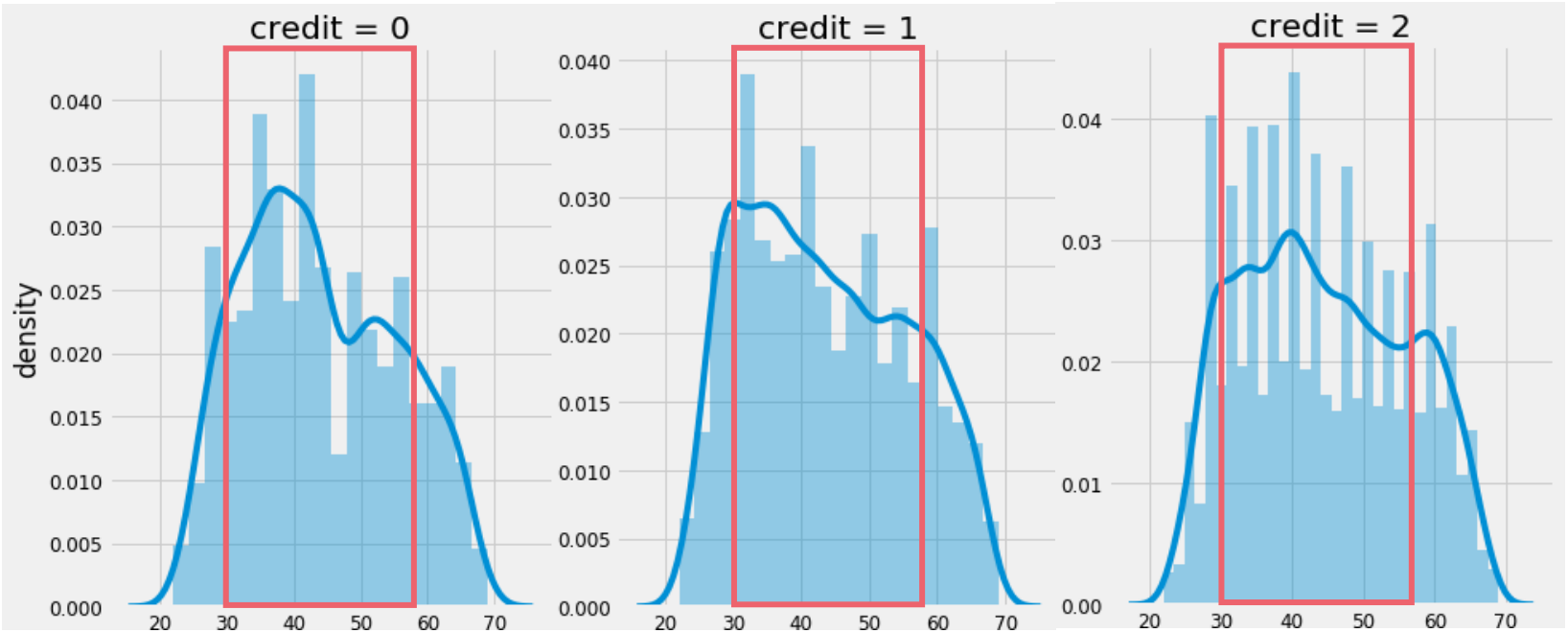
DAYS_BIRTH : 출생일을 의미

태어난 날을 기준으로 살아온 날을 음수로 표기 ex) 5000일 전에 태어난 경우 -5000으로 표기

▶ DAYS_BIRTH를 나이로 변환하여 age 파생 변수 생성

신용등급별 연령대 확인: 모든 등급에서 30~50대가 주를 이루고 있고, 20대의 비율이 매우 낮음

< age – 신용등급별 연령대 >



Numerical Variable EDA

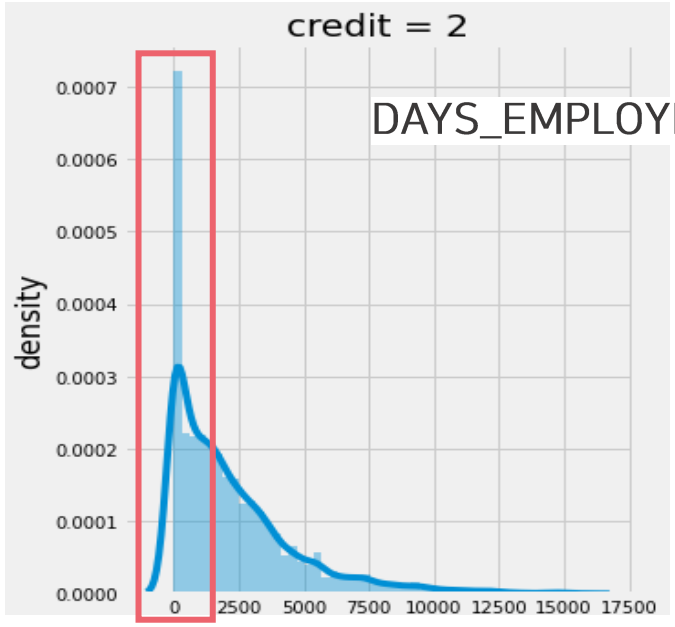
DAYS_EMPLOYED: 고용되어 일한 날을 의미 (음수 표기)

데이터 수집 당시 (0)부터 역으로 계산

ex) -100은 데이터 수집일 100일 전부터 일을 시작함을 의미

양수 값은 무직 상태

▶ 모든 신용등급에서 무직인 경우가 매우 많은 것을 확인

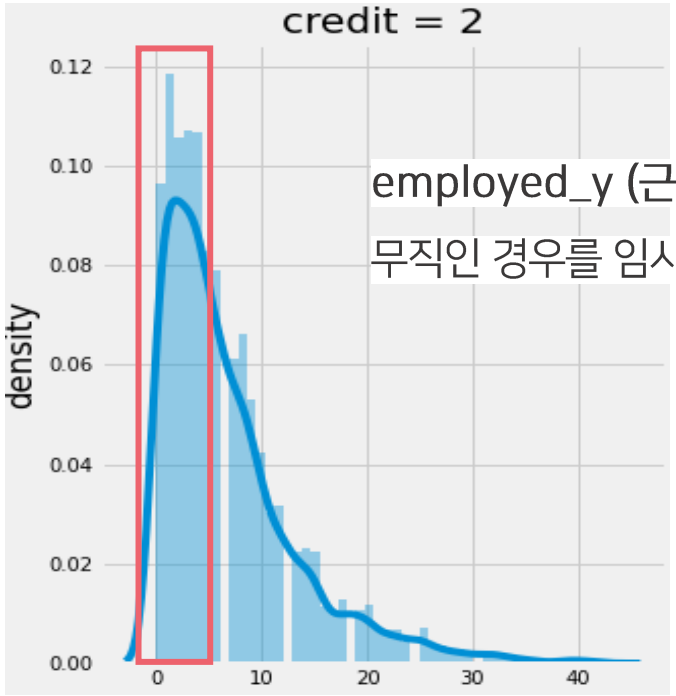


DAYS_EMPLOYED(고용일)

▶ employed_y (근속 연수) 파생변수 생성

모든 신용등급에서 근속연수의 차이가 없음

근속연수가 0인 경우가 매우 많고, 대부분 근속연수가 5년 이하



employed_y (근속연수)

무직인 경우를 임시로 제외하고 확인

Numerical Variable EDA

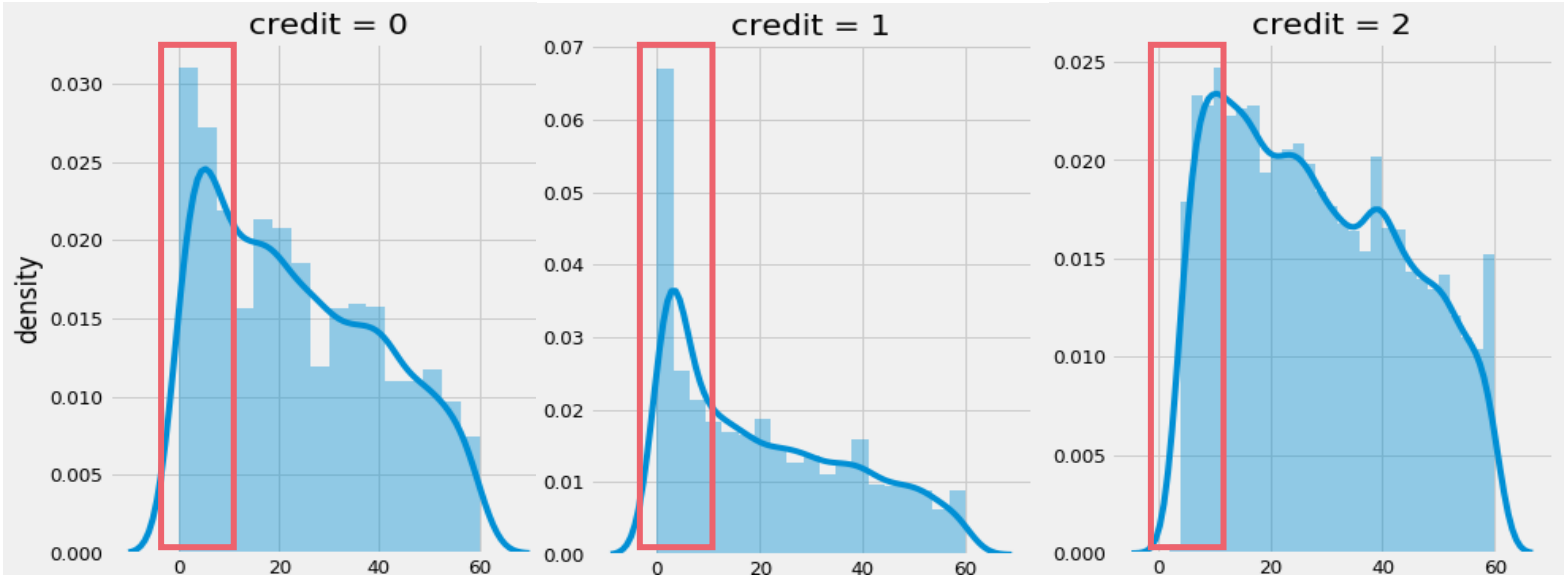
begin_month: 카드 발급 경과 월 수

데이터 수집 당시 (0)부터 역으로 계산 ex) 3개월 전에 카드 발급을 신청했다면 -3으로 표기

- 모든 신용등급에서 카드를 발급 받은 지 10달 이하인 사람들이 많음
- begin_month와 credit을 제외하고 모든 컬럼의 값이 같은 경우 존재

이 경우 **동일한 사용자가 여러 개의 신용카드를 만든 경우**를 의미 ⇨ **중복 사용자 처리 필요!**

< 카드 사용 개월 수>



03

Data Preprocessing

데이터 전처리

Data Preprocessing

탐색적 데이터 분석(EDA)를 기반으로 전처리 수행

Step1 결측치 대체

occyp_type(직업유형)의 결측치 대체

train, test 데이터에 결측치가 존재하고, train 데이터의 결측치 비율이 전체 데이터의 37%이므로 이므로 해당 row나 feature를 drop하는 것을 불가능

- DAYS_EMPLOYED(고용일)이 양수인 경우, 무직을 의미하므로 'No_job'으로 대체
- 이외 값은 'NaN' 으로 대체 (결측치 중 DAYS_EMPLOYED가 음수인 데이터)

Step2 중복 값 처리

중복 사용자 처리

동일한 사용자가 begin_month에 따라 다른 credit값을 가지고 있는 경우 확인

한 명의 사용자가 여러 개의 카드를 만든 경우를 의미함

- ID 변수를 추가해서 중복 사용자 처리

Step3 이상치 제거

child_num이 6 이상 이고, family_size가 7보다 큰 데이터

- family_size > 7 인 데이터 제거

04 Feature engineering

Feature engineering

1) 특정 feature의 음수 값을 양수로 바꾸기

DAYS_BIRTH, begin_month, DAYS_EMPLOYED는 데이터 수집일로부터 음수로 표기 → 해당 값을 양수로 변경

2) 의미없는 변수 제거

Index, FLAG_MOBIL 변수 삭제

3) 파생변수 생성

중요도가 높은 DAYS_EMPLOYED와 income_total 관련 파생변수 생성

age(나이) employed_y(근속연수), employed_m(고용된 달), employed_w(고용된 주.고용연도의 n주차)

income_ability: 소득/(살아온 일수+ 근무일수), income_mean(소득/가족 수)

4) Scailing

Income_total 변수를 log scale

StandardScale을 통해 income_total을 제외한 나머지 numeric 변수를 정규화

5) Encoding

카테고리 변수 OrdinalEncoder 변환

Feature engineering

6) Feature Selection

다중공선성 확인 후 컬럼 삭제하고, 필요한 컬럼만 사용

(1) 상관계수를 통한 다중공선성 확인

변수들간의 상관관계가 0.5가 넘어가면 다중공선성 발생을 의심할 수 있음

But 다중공선성이 있다면 상관관계는 높지만, 상관관계가 높다고 다중공선성이 반드시 있는 것은 아님

child_num과 family_size : 0.89

income_total과 income_mean : 0.67

income_total과 income_ability : 0.81

income_ ability와 income_mean : 0.59

(2) VIF를 통한 다중공선성 확인

VIF가 10이 넘으면 다중공선성이 있다고 판단

파생변수와 관련된 feature의 VIF가 높게 나타남

VIF가 높은 변수들을 하나씩 제거하면서 다중공선성을 확인한 결과, 4개의 컬럼 삭제 후 다중공선성 문제가 해결

DAYS_BIRTH, DAYS_EMPLOYED, income_total, child_num

05

Modeling

모델 선택 및 학습 과정

Modeling with Pycaret

적합한 모델을 빠르게 판단하고 비교하기 위해 Pycaret 사용

Pycaret

AutoML 파이썬 라이브러리

scikit-learn 패키지를 기반으로 하고 있으며 다양한 모델을 지원

전처리, 모델 학습, 모델 선택, 파라미터 튜닝 작업을 자동화

모델 평가 기준: LogLoss

분류모델에서 사용하는 평가지표

모델이 예측한 확률 값을 직접적으로 반영하여 평가

확률 값을 음의 log함수에 넣어 변환을 시킨 값으로 평가함으로써 잘못 예측한 경우 패널티를 부여

모델 별 성능 비교 - Best 7

모델	LogLoss	Accuracy
CatBoost	0.7594	70.35%
LightGBM	0.7606	69.93%
Gradient Boosting	0.7946	69.32%
LDA	0.8625	64.35%
Logistic Regression	0.8655	64.22%
Naïve Bayes	0.8670	64.90% ▲
Random Forest	0.9872	70.13% ▲

Modeling with Pycaret

다양한 모델의 시도

모델	LogLoss	Accuracy
CatBoost 튜닝 모델	0.7499	70.55%
LightGBM 튜닝 모델	0.7557	70.42%
CatBoost	0.7594	70.35%
LightGBM	0.7606	69.93%
Best2 모델 Blending	0.7746	69.62%
Custom 모델 Blending	0.7871	69.53%
Best5 모델 Blending	0.7872	69.47%

모델 튜닝을 통한 성능 향상

Best5 모델에서 LR을 제외하고 Accuracy가 높은 Naïve Bayes와 Random Forest 추가

Blending
CatBoost와 LGBM 단일 모델보다 좋지 않은 성능

➔ 가장 높은 성능을 보이는 CatBoost 단일 모델을 최종 모델로 결정

최종 모델링 과정

StratifiedKFold를 통한 교차 검증

label 분포의 불균형을 해결

각 fold가 전체 데이터셋을 대표할 수 있도록 함

학습과 평가에 사용되는 데이터 편중을 방지

편향되지 않게 학습되지 않고 좀 더 일반화된 모델을 만들기 위해 사용

Modeling with CatBoost

기존 GBM 알고리즘의 과적합 문제를 해결하고, 학습속도를 개선한 알고리즘

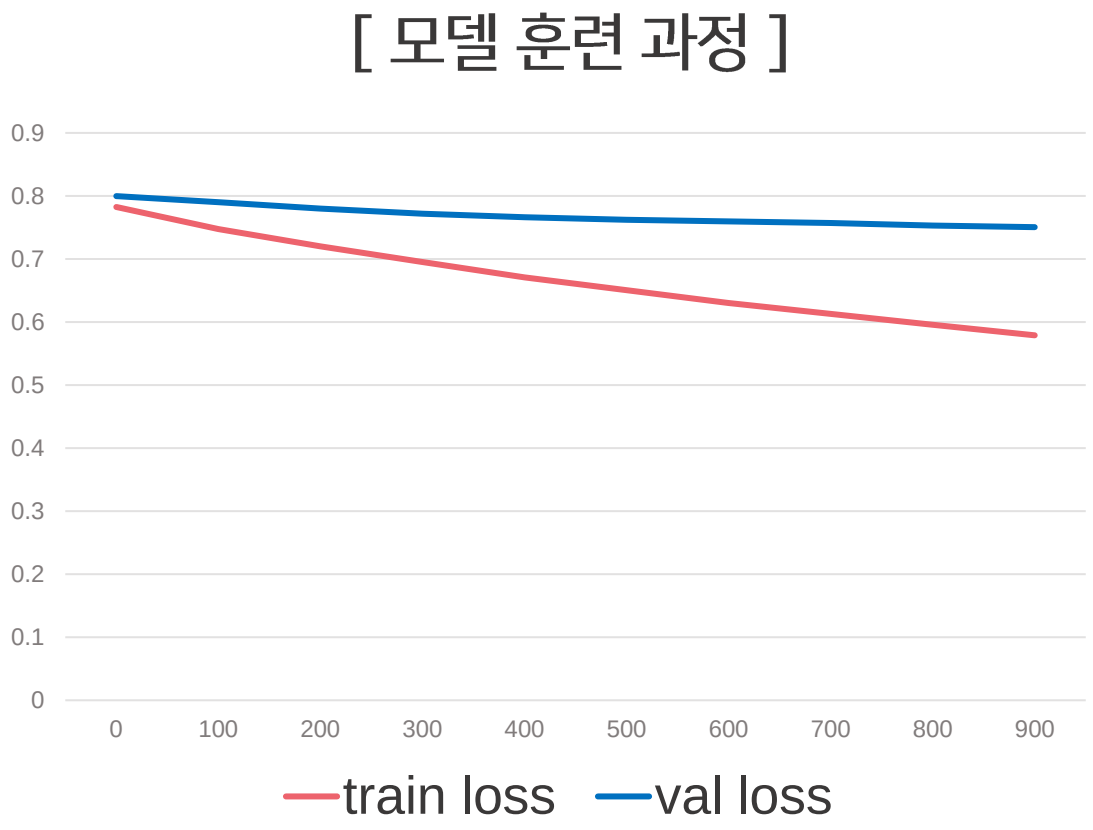
XGBoost, LightGBM이 Hyper-parameter에 따라 성능이 달라지는 민감한 문제를 해결

- Feature를 모두 동일하게 대칭적인 트리 구조를 형성 → 예측시간 감소
- Ordered Boosting: 일부만으로 잔차 계산을 하여 모델을 만들고, 이후의 데이터 잔차는 이 모델로 예측한 값을 사용
- 데이터를 N개의 Fold로 나누어서 각 Fold에 속한 데이터셋들에 Ordered Boosting을 적용 → 과적합 해결
- Ordered Target Encoding: 현재 데이터의 인코딩을 위해 이전 데이터들의 인코딩된 값을 사용
  → 과적합 해결, 수치 값의 다양성
- 기본적으로 파라미터 최적화되어있어, 파라미터 튜닝에 신경쓰지 않아도 된다

06

Conclusion

모델 학습 및 예측 결과



CatBoost 모델 학습 best score

Learn: 0.5793175154580043

Validation: 0.7505804760049398

Dacon 제출 결과

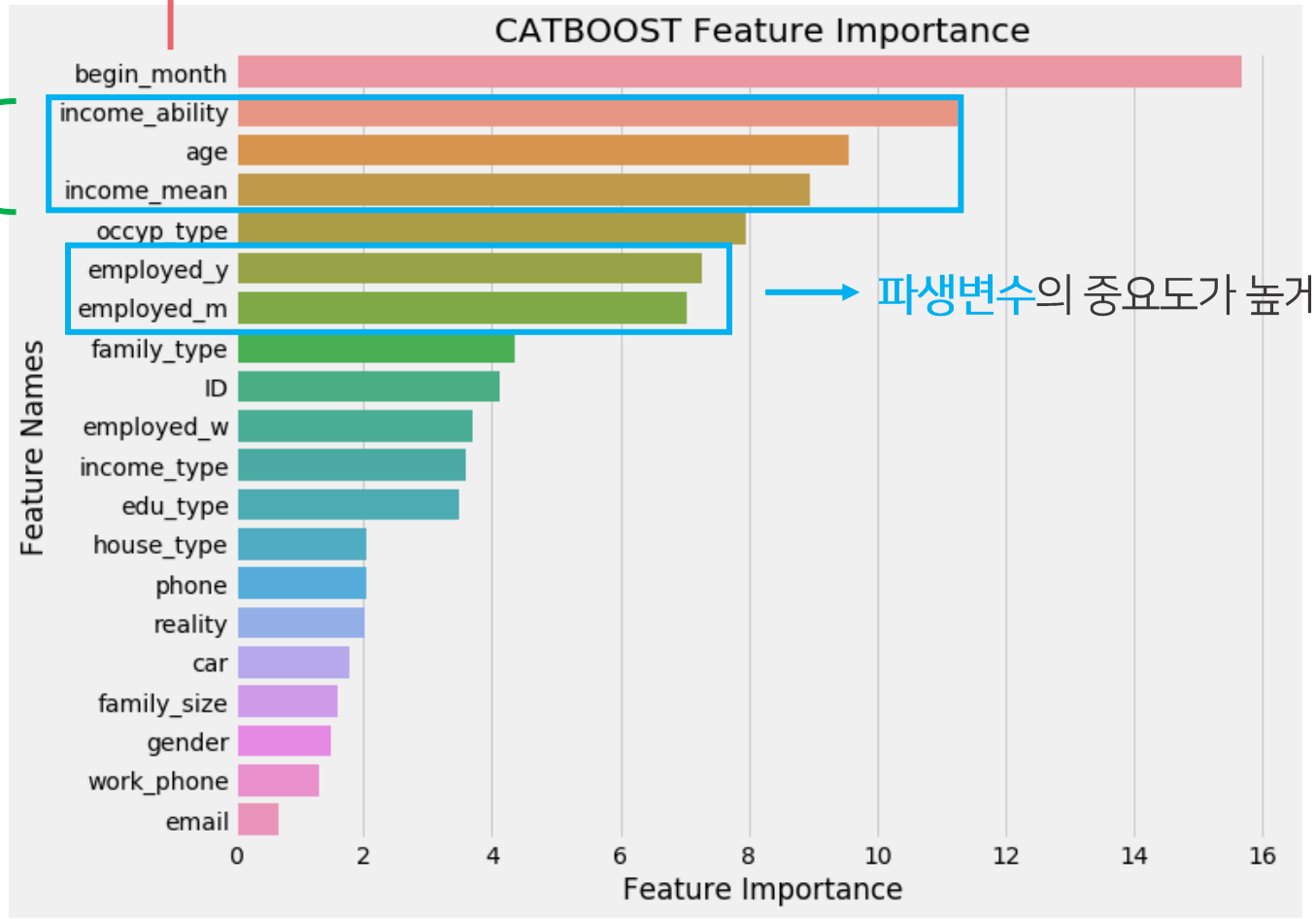
Public score: 0.7318231351

Private score: 0.7147909377

Feature Importance

begin_month(카드 발급 경과 월 수)의 중요도가 매우 높음

소득 관련 변수의 중요도가 높음



파생변수의 중요도가 높게 나타남

프로젝트를 마무리하며

프로젝트를 통해 배운 점과 아쉬운 점

EDA(탐색적 데이터 분석)의 중요성

탐색적 데이터 분석을 통한 다양한 인사이트

이를 기반으로 전처리와 Feature engineering을 수행하여 성능을 향상

Pycaret의 활용

쉽고 빠르게 모델링을 수행, 다양한 모델의 성능을 비교

성능 향상을 위한 여러가지 방법 시도

최종 학습 시, Catboost 모델의 세밀한 튜닝

스태킹, weighted Averaging ensemble 등 앙상블 방법의 시도

Thank you for listening