

LOAN REPAYMENT PREDICTION

Machine Learning Project

노현진 박주희 박현태 육지현

이한나

목차

01 프로젝트 개요

분석 배경
흐름도
일정표
분석 방법론
개발 환경

02 EDA 및 데이터 전처리

EDA
데이터 병합
이상치 및 결측치 확인
파생변수 생성
피쳐 엔지니어링

03 모델링

모델 선정 및 이유
모델 최적화
모델 평가

04 웹 시연

Flask 기반 웹 구현

05 마무리 및 소감

팀원 소감

목차

CHAPTER 1. 프로젝트 개요

1. 분석 배경

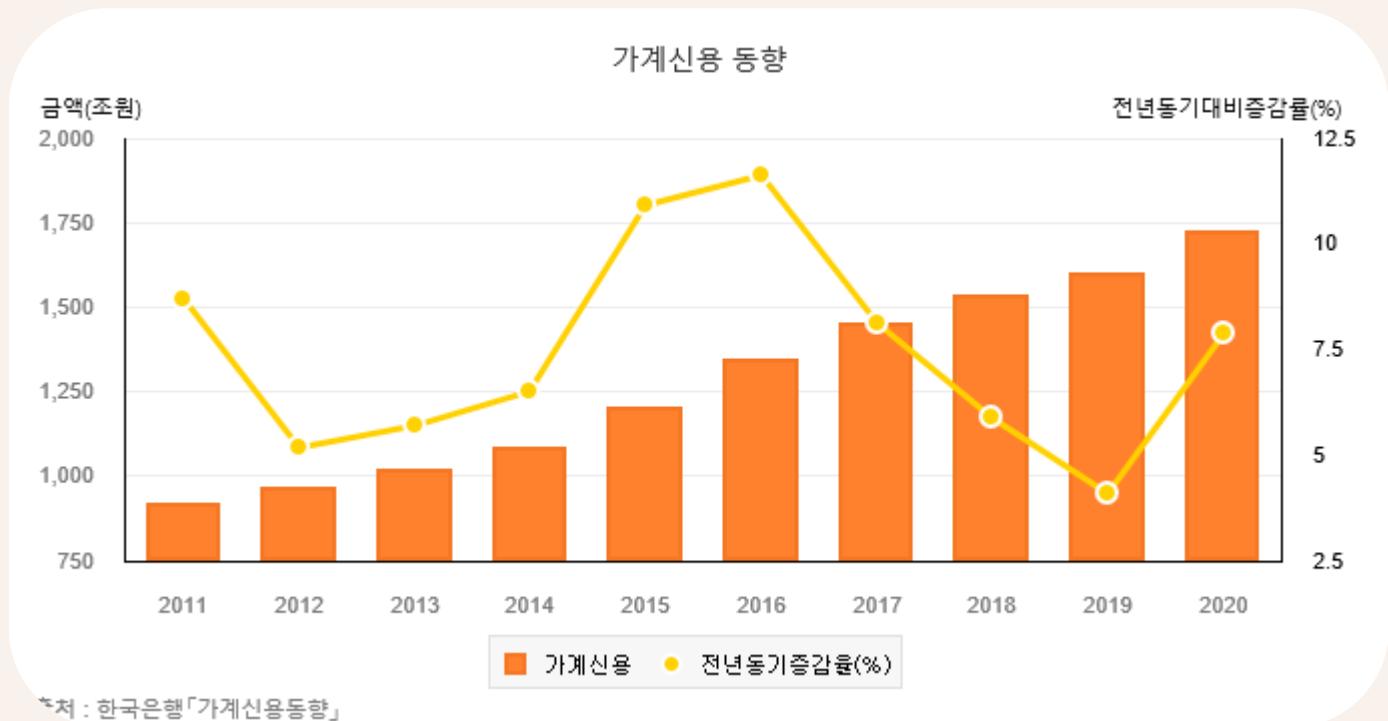
2. 흐름도

3. 일정표

4. 분석 방법론

5. 개발 환경

주제 선정 이유



- 2011년부터 2020년까지의 가계신용 동향 그래프를 통해 보면 다음과 같다.
- 또한 최근 기준금리가 작년 동기 기준금리에 비해 0.5% 상승한 1.0%로 결정되면서 가계대출 평균 금리가 2021년 11월 기준 5%를 넘어서면서 7년여만에 최고치를 기록하고 있다.

분석 배경

[Kaggle] Home Credit Defalut Risk

1997년 체코에서 설립된 Home Credit은 신용 기록이 없거나 적은 사람들을 대상으로 대출을 진행하는 비은행 금융 기관이며, 신용 기록이 없는 사람들에게도 대출이 가능하도록 '위 고객이 대출을 상환할 수 있는 능력이 되는가?'를 판단한다.

과거 신용 거래 이력 및
대출 기록, 대출 신청 시
작성 내용



통계 및 머신러닝 기법



해당 고객의 대출금액 상환 확률
예측

프로젝트 흐름도

주제 선정

머신러닝을 활용한
신용 대출 상환 능력 예측

[Kaggle] Home Credit Defalut Risk

데이터 선별

- Home Credit 대출 신청 시 작성 내용 (application_train / test)
- 개인 신용 평가기관에 기록된 신청자의 과거 신용 거래 내역 (bureau)
- Home Credit의 과거 대출 기록 (previous_application)

EDA

- 전 데이터 컬럼 분석
- 이상치 확인 및 선택적 제거
- 결측치 확인 후 부스팅 선택
- 상관관계 분석

PreProcessing

모델링

- lightGBM
- GridSearchCV
- F-fold

파생변수 추가

파라미터 최적화

Flask 기반 웹 구현

- 대출 신청자 ID 입력 시 대출 승인 여부 확인

일정표

Notion 활용

12월 23일 3	...	12월 24일 3		12월 27일 3		12월 28일 3	
깃 허브 연동 <input type="checkbox"/> Official		데이터셋 확인 및 각 컬럼 분석 <input type="checkbox"/> Official		결측치 처리 및 기타 피쳐 제거 <input type="checkbox"/> Official		상관관계 분석 <input type="checkbox"/> Official	
주제 선정 <input type="checkbox"/> Official		EDA 예제 필사 및 데이터 확인 <input type="checkbox"/> Official		상관분석 (unique값 확인, 데이터 분포 확인 등 시각화) <input type="checkbox"/> Official		이상치 처리 <input type="checkbox"/> Official	
가용 데이터 선별 <input type="checkbox"/> Official		EDA 및 시각화 <input type="checkbox"/> Official		EDA 및 시각화 <input type="checkbox"/> Official		ML 모델링, 인코딩, 스케일링 <input type="checkbox"/> Official	
+ 새로 만들기		+ 새로 만들기		+ 새로 만들기		+ 새로 만들기	

12월 29일 3	12월 30일 1	12월 31일 1
Flask 기반 웹 구현 <input type="checkbox"/> Official	발표 자료 제작 <input type="checkbox"/> Official	프로젝트 최종 발표 <input type="checkbox"/> Official
모델링 평가 <input type="checkbox"/> Official	+ 새로 만들기	+ 새로 만들기
발표 자료 제작 <input type="checkbox"/> Official		
+ 새로 만들기		

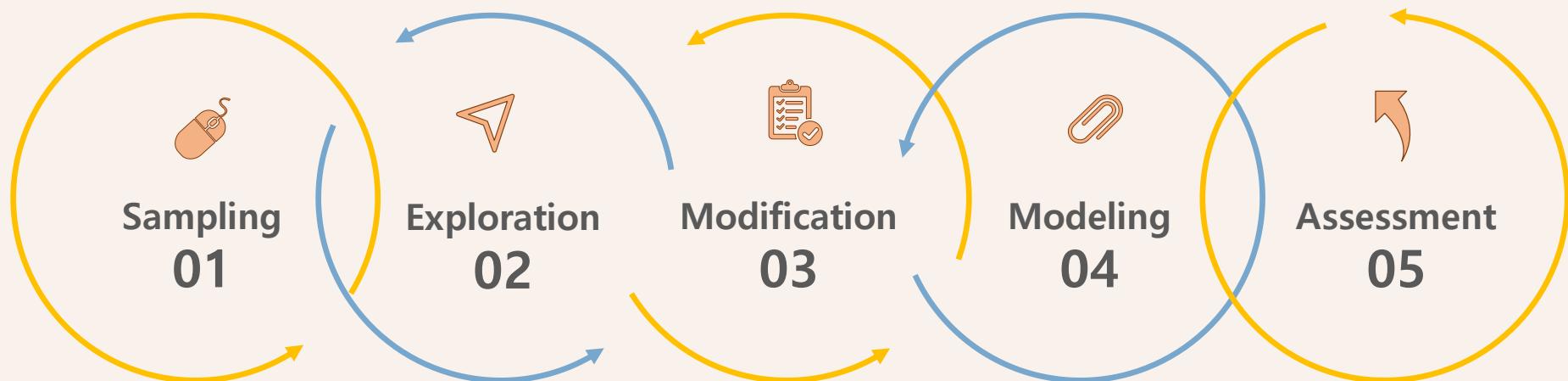
일정표

Notion 활용

	12/23		12/24		12/27		12/28		12/29		12/30		12/31	
	오전	오후												
주제 선정														
데이터 선별														
EDA														
데이터 전처리														
모델링														
파라미터 최적화														
웹 구현														
발표 준비														
발표														

분석 방법론

기술 중심, 통계 중심 방법론인 **SEMMA** 방법론 활용



분석 데이터 생

- 총 7개의 데이터셋 중 4개 데이터셋 선정

데이터 탐

- 세부 데이터 컬럼 분석
- 불균형한 target 분포 확인
- 통계 지표 분석

데이터 수

- 이상정 분석
 - 파생변수 생성
 - 데이터 병합
 - 결측치 확인 및 제거
- lightGBM, K-fold, ROS 기법 등 모델 선정
- 알고리즘 적용 모델 구축
- 파라미터 설정

모델

- 모델 가정 및 검증
- 최적 분석 결과
- SCORE 확인

평

개발 도구 및 환경

OS



Window 10

형상관리 Tools



Git Hub



Notion



Google Drive



Git-Fork

Language

Python
3.8.8

HTML5



CSS3

Java Script
1.7

IDE

Jupyter
6.4.5Colab
Pro +VS Code
1.63.2PyCharm
2021.2Atom
1.58.0

Kaggle code

Spyder
4.0.1

Library & Framework

NumPy
Numpy
1.21.4

pandas

Pandas
1.3.5

matplotlib

Matplotlib
3.5.0

seaborn

Seaborn
0.11.2Scikit learn
0.24.2Imbalanced learn
0.8.1Bootstrap
4.1.0Flask
1.1.2

LightGBM

목차

CHAPTER 2. 데이터 전처리

1. EDA

2. 데이터 병합

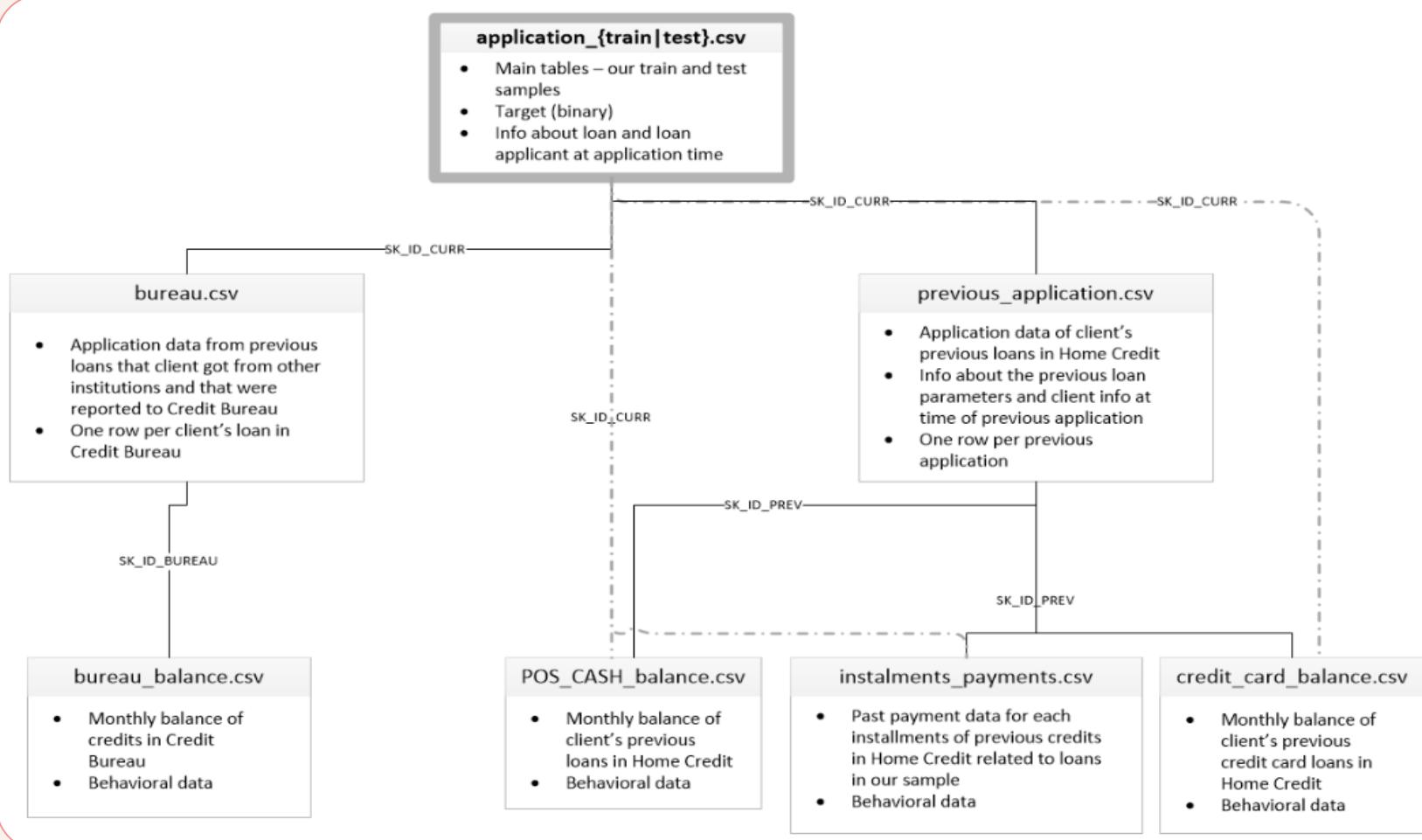
3. 이상치 및 결측치 확인

4. 파생변수 생성

5. 피쳐 엔지니어링

ERD

- csv 파일 구성은 아래와 같음
- 이 중에서 application_train.csv와 bureau.csv, previous_application.csv 파일을 이용함.

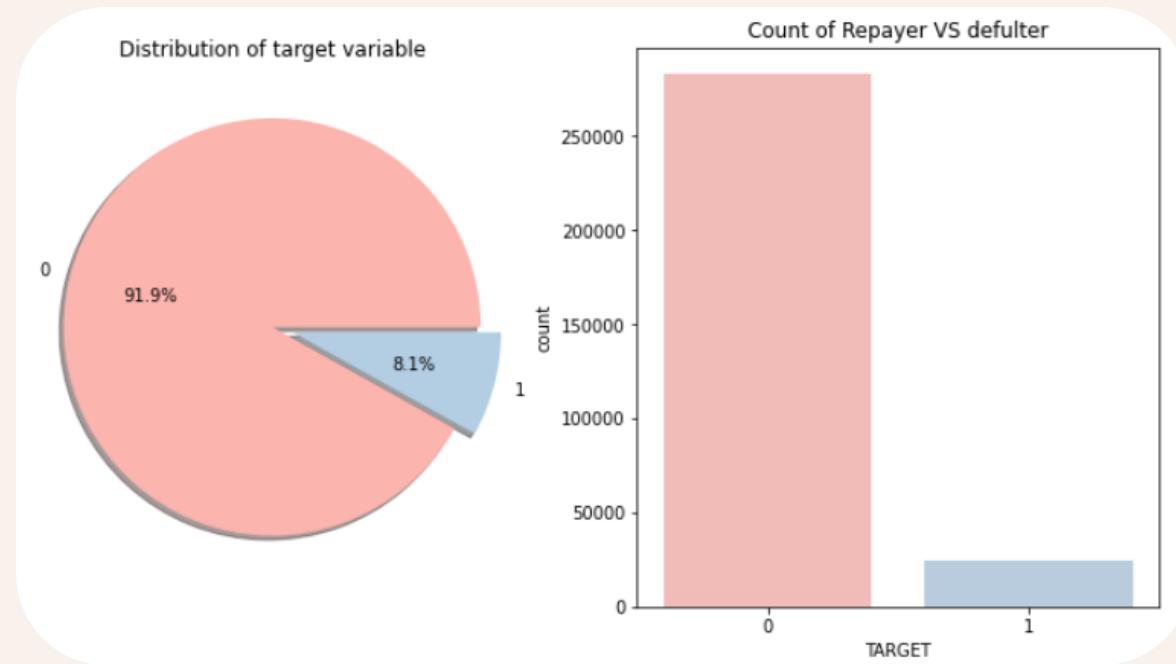


Data Dictionary

- 각 csv 파일의 column을 아래와 같이 정리해

테이블	열	설명
application_{train test}.csv	SK_ID_CURR	샘플의 대출 ID
application_{train test}.csv	TARGET	대상 변수(1 - 지불 문제 있는 고객: 샘플에서 대출의 첫 번째 Y 분할 중 하나 이상에 대해 X일 이상 연체, 0 - 기타 모든 경우)
application_{train test}.csv	NAME_CONTRACT_TYPE	대출이 현금인지 회전인지 확인
application_{train test}.csv	CODE_GENDER	클라이언트의 성별
application_{train test}.csv	FLAG_OWN_CAR	클라이언트가 자동차를 소유한 경우 플래그 지정
application_{train test}.csv	FLAG_OWN_REALTY	클라이언트가 집이나 아파트를 소유하고 있는 경우 플래그 지정
application_{train test}.csv	CNT_CHILDREN	클라이언트의 자녀 수
application_{train test}.csv	AMT_INCOME_TOTAL	클라이언트의 소득
application_{train test}.csv	AMT_CREDIT	대출의 신용 금액
application_{train test}.csv	AMT_ANNUITY	대출 연금
application_{train test}.csv	AMT_GOODS_PRICE	소비자 대출의 경우 대출이 제공되는 상품의 가격입니다.
application_{train test}.csv	NAME_TYPE_SUITE	대출을 신청할 때 고객과 동행한 사람
application_{train test}.csv	NAME_INCOME_TYPE	고객 소득 유형(사업가, 근로, 출산휴가 등)
application_{train test}.csv	NAME_EDUCATION_TYPE	클라이언트가 달성한 최고 교육 수준
application_{train test}.csv	NAME_FAMILY_STATUS	클라이언트의 가족 상태
application_{train test}.csv	NAME_HOUSING_TYPE	클라이언트의 주거 상황은 무엇입니까(임대, 부모와 동거, ...)
application_{train test}.csv	REGION_POPULATION_RELATIVE	고객이 거주하는 지역의 정규화된 인구(숫자가 높을수록 고객이 인구가 많은 지역에 거주한다는 의미)
application_{train test}.csv	DAYS_BIRTH	신청 당시 고객의 나이(일)
application_{train test}.csv	DAYS_EMPLOYED	신청자가 현재 고용을 시작하기 며칠 전
application_{train test}.csv	DAYS_REGISTRATION	클라이언트가 등록을 변경하기 며칠 전
application_{train test}.csv	DAYS_ID_PUBLISH	신청하기 며칠 전에 고객이 대출을 신청한 신분증을 변경했습니까?
application_{train test}.csv	OWN_CAR_AGE	고객 차량의 나이
application_{train test}.csv	FLAG_MOBIL	고객이 휴대전화를 제공했습니까(1=예, 0=아니요)
application_{train test}.csv	FLAG_EMP_PHONE	클라이언트가 직장 전화를 제공했습니까(1=예, 0=아니요)
application_{train test}.csv	FLAG_WORK_PHONE	고객이 집 전화를 제공했습니까(1=예, 0=아니요)
application_{train test}.csv	FLAG_CONT_MOBILE	휴대전화에 연결할 수 있었습니까(1=예, 0=아니요)
application_{train test}.csv	FLAG_PHONE	고객이 집 전화를 제공했습니까(1=예, 0=아니요)
application_{train test}.csv	FLAG_EMAIL	고객이 이메일을 제공했습니까(1=예, 0=아니요)
application_{train test}.csv	OCCUPATION_TYPE	클라이언트는 어떤 직업을 가지고 있습니까?
application_{train test}.csv	CNT_FAM_MEMBERS	고객의 가족 구성원은 몇 명입니까?

데이터 모형



- 위의 그래프는 application_train.csv 파일의 TARGET의 비율을 나타냄.
- 대출 상환 능력이 있음(0)이 91.9%이고, 대출 상환 능력이 없음(1)이 8.1%로 데이터의 비율이 불균형.
- 대출 상환 능력이 있음(0)의 경우 데이터의 개수는 282686개이고, 대출 상환 능력이 없음(1)의 경우 데이터의 개수는 24825개 임.

Target 분포

- `plot_stats()`는 입력한 `feature`에 대한 카테고리별 개수와 TARGET이 1인 경우일 때, 해당 `feature`의 카테고리별 비율을 barplot으로 나타냄

```
def plot_stats(feature, label_rotation=False, horizontal_layout=True):
    temp = df_train[feature].value_counts()
    df1 = pd.DataFrame({feature: temp.index, 'Number of contracts': temp.values})

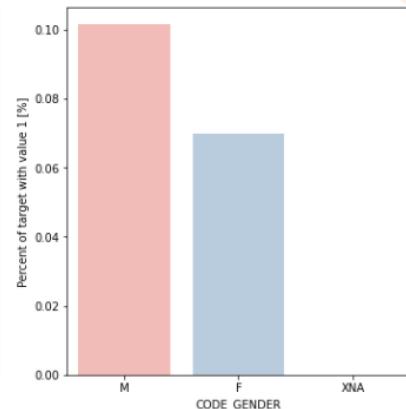
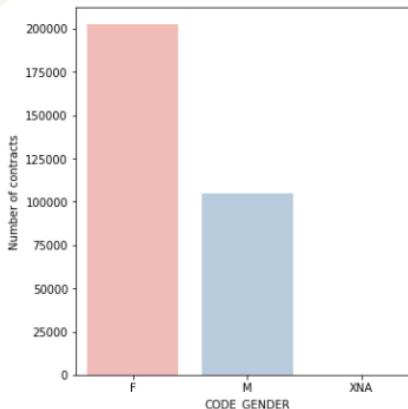
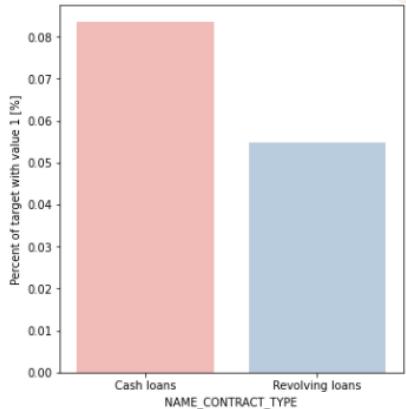
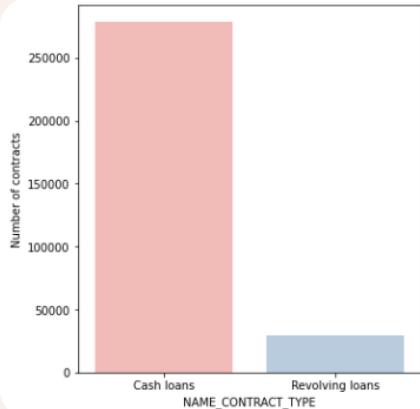
    # Calculate the percentage of target=1 per category value
    cat_perc = df_train[[feature, 'TARGET']].groupby([feature], as_index=False).mean()
    cat_perc.sort_values(by='TARGET', ascending=False, inplace=True)

    if(horizontal_layout):
        fig, (ax1, ax2) = plt.subplots(ncols=2, figsize=(12,6))
    else:
        fig, (ax1, ax2) = plt.subplots(nrows=2, figsize=(12,14))
    sns.set_color_codes("pastel")
    s = sns.barplot(ax=ax1, x = feature, y="Number of contracts", data=df1)
    if(label_rotation):
        s.set_xticklabels(s.get_xticklabels(), rotation=90)

    s = sns.barplot(ax=ax2, x = feature, y='TARGET', order=cat_perc[feature], data=cat_perc)
    if(label_rotation):
        s.set_xticklabels(s.get_xticklabels(), rotation=90)
    plt.ylabel('Percent of target with value 1 [%]', fontsize=10)
    plt.tick_params(axis='both', which='major', labelsize=10)

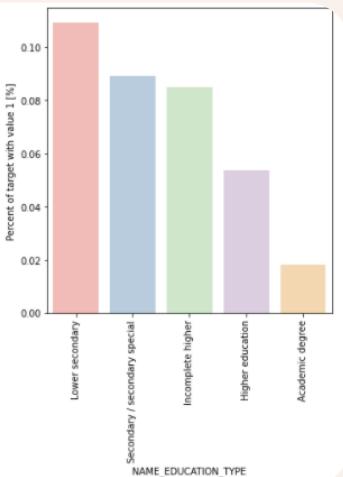
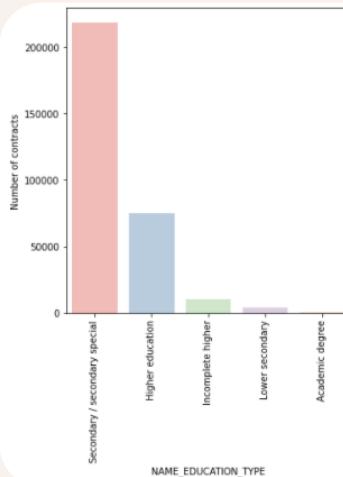
    plt.show();
```

Target 분포



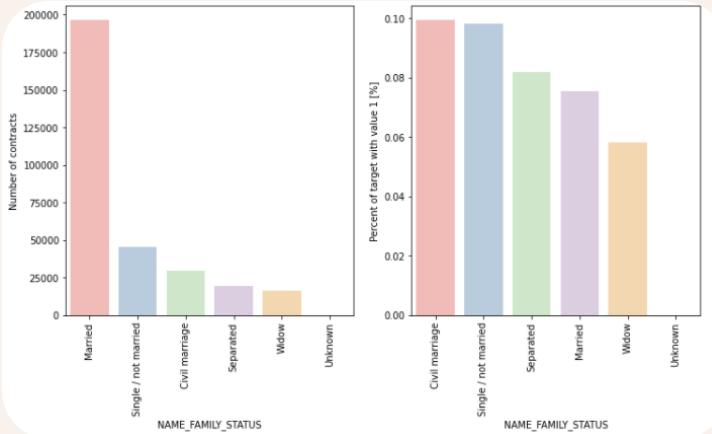
- **Cash loans** 대출 수가 전체 대출수의 약 95% **Revolving loans** 가 전체 대출수의 약

- 대출하는 고객은 여성이 남성보다 약 2배 이상
- 채무불이행은 남성이 더 높음 (남성10%/여성7%)

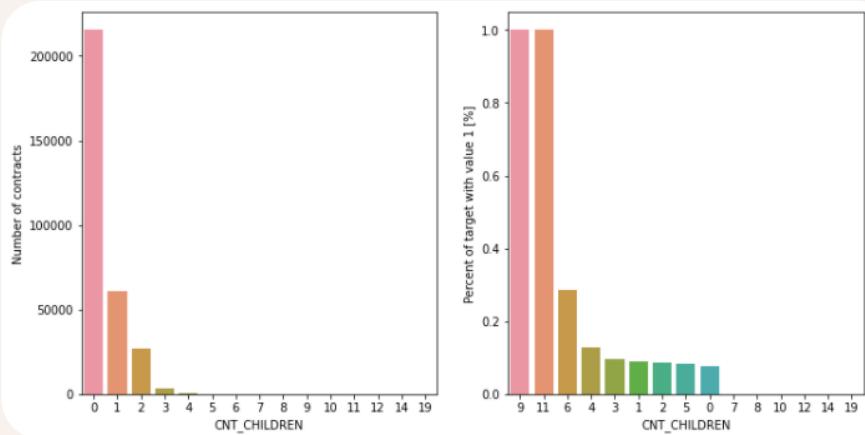


- 중등교육 받은 고객이 대다수, 학위 있는 사람은 적음

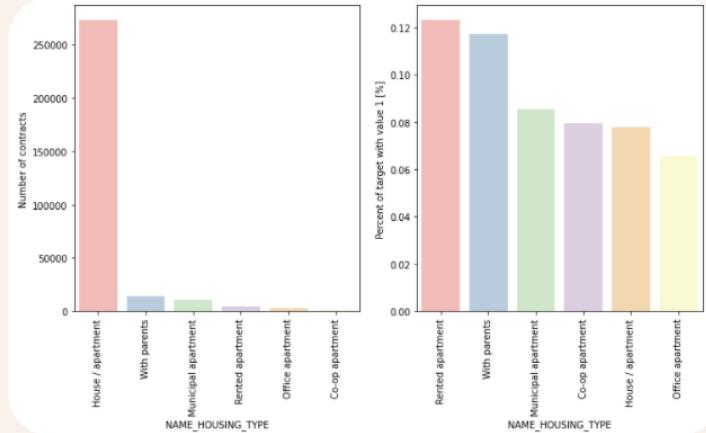
Target 분포



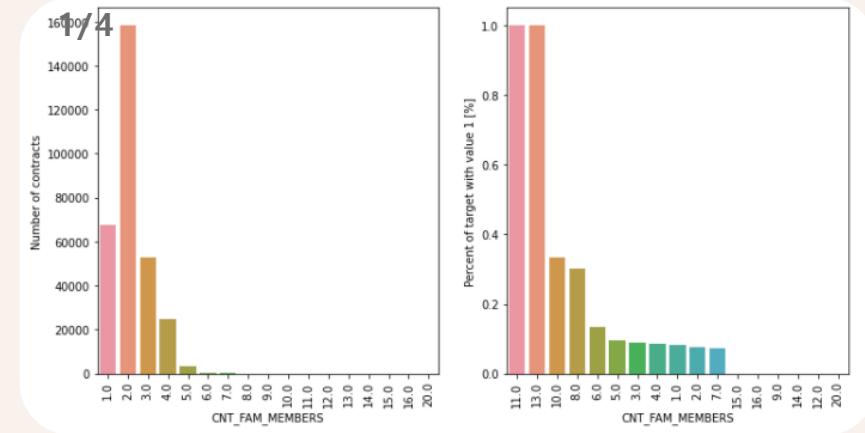
- 대부분의 고객이 결혼함



- 대출받는 고객의 대부분이 자녀가 없음.
- 자녀 1명을 둔 고객은 자녀 없는 고객의 약

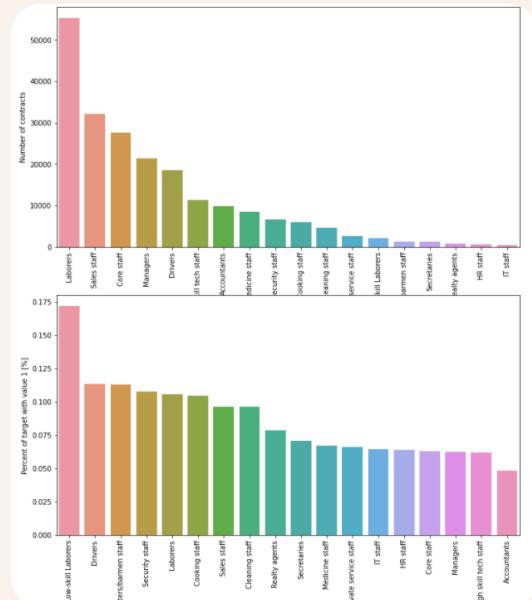
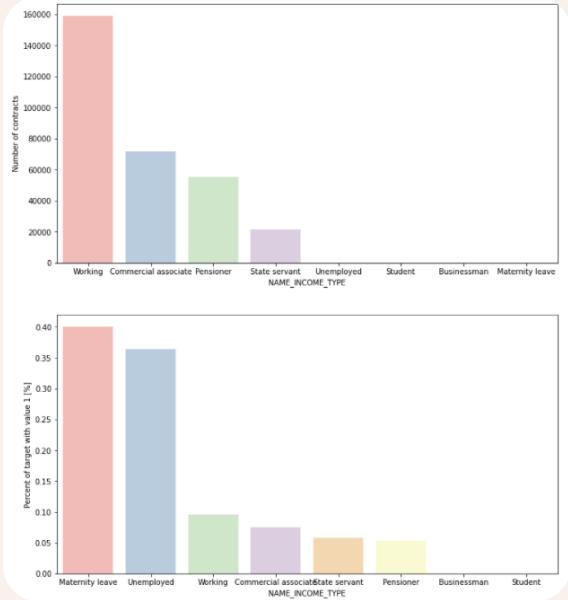
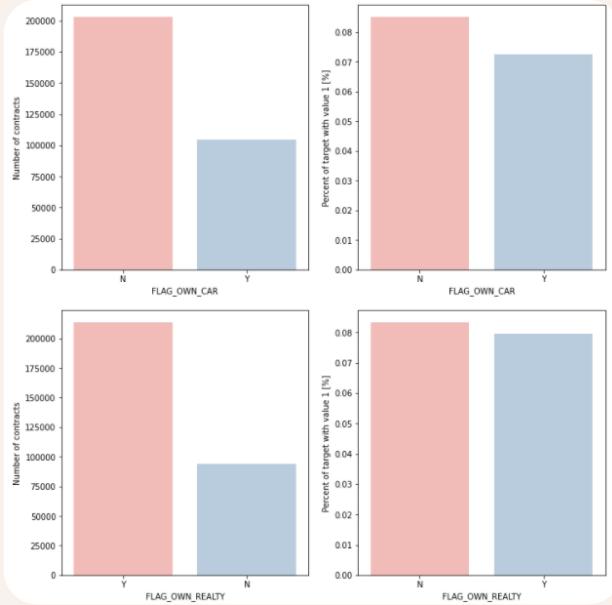


- 고객 대부분이 주택/아파트에 거주



- 가족인원이 2명인 경우 고객이 가장 많음

Target 분포

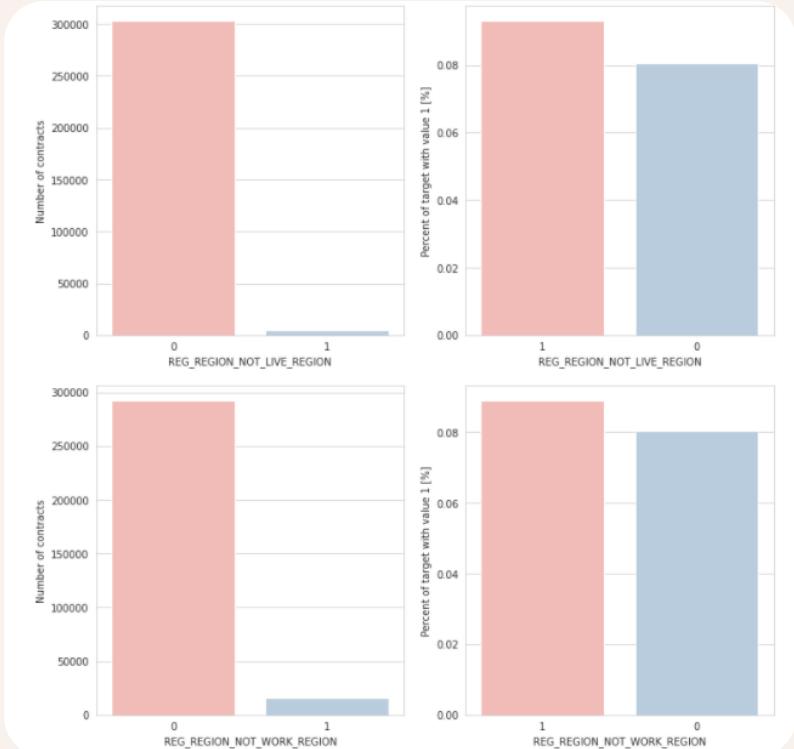


- 차를 소유하지 않은 사람이 소유한 사람보다 약 2배 많음
- 집을 소유한 사람이 소유하지 않은 사람보다 약 2배 많음
- 채무불이행 비율은 비슷함

- 대출 신청자는 근로소득이 대부분이고 상업관계자
- 채무불이행은 출산휴가자가 약 40%, 그리고 실업자가 약 37%

- 대출 신청자는 대부분 노동자, 영업직원 (IT직원이 가장 낮음)
- 채무 불이행은 비숙련 노동자가 많음 (그 다음, 운전, 이발사, 경비원, 노무자 등)

Target 분포



- 'REG_REGION_NOT_LIVE_REGION' 고객의 영구 주소가 연락처 주소와 일치하지 않는 경우 플래그 지정
- 'REG_REGION_NOT_WORK_REGION' 고객의 영구 주소와 직장 주소가 일치하는지
- 일반적으로 거주하거나 일하는 곳을 주소로 등록함
- 다른 곳에 등록할 경우 채무불이행 경우가 많음

Shape

	columns	rows
train	122	307,511
previous	37	1,670,214
bureau	17	1,716,428

Describe

application_train.csv

	SK_ID_CURR	TARGET	NAME_CONTRACT_TYPE	CODE_GENDER	FLAG_OWN_CAR	FLAG_OWN_REALTY	CNT_CHILDREN	AMT_INCOME_TOTAL	AMT_CREDIT	AMT_ANNUITY	AMT_GOODS_PRICE	NAME_TYPE_SUITE	...
count	307511.000000	307511.000000	307511	307511	307511	307511	307511.000000	3.075110e+05	3.075110e+05	307499.000000	3.072330e+05	306219	...
unique	NaN	NaN	2	3	2	2	NaN	NaN	NaN	NaN	NaN	NaN	7
top	NaN	NaN	Cash loans	F	N	Y	NaN	NaN	NaN	NaN	NaN	NaN	Unaccompanied
freq	NaN	NaN	278232	202448	202924	213312	NaN	NaN	NaN	NaN	NaN	NaN	248526
mean	278180.518577	0.080729	NaN	NaN	NaN	NaN	0.417052	1.687979e+05	5.990260e+05	27108.573909	5.383962e+05	NaN	...
std	102790.175348	0.272419	NaN	NaN	NaN	NaN	0.722121	2.371231e+05	4.024908e+05	14493.737315	3.694465e+05	NaN	...
min	100002.000000	0.000000	NaN	NaN	NaN	NaN	0.000000	2.565000e+04	4.500000e+04	1615.500000	4.050000e+04	NaN	...
25%	189145.500000	0.000000	NaN	NaN	NaN	NaN	0.000000	1.125000e+05	2.700000e+05	16524.000000	2.385000e+05	NaN	...
50%	278202.000000	0.000000	NaN	NaN	NaN	NaN	0.000000	1.471500e+05	5.135310e+05	24903.000000	4.500000e+05	NaN	...
75%	367142.500000	0.000000	NaN	NaN	NaN	NaN	1.000000	2.025000e+05	8.086500e+05	34596.000000	6.795000e+05	NaN	...
max	456255.000000	1.000000	NaN	NaN	NaN	NaN	19.000000	1.170000e+08	4.050000e+06	258025.500000	4.050000e+06	NaN	...

- 'AMT_INCOME_TOTAL'의 평균은 168,797 / 편차는 237,123 / 중위수는 147,150
- 'AMT_CREDIT'의 평균은 599,026 / 편차는 402,490 / 중위수는 513,531
- 'AMT_ANNUITY'의 평균은 27,108 / 편차는 14,493 / 중위수는 24,903

Describe

previous_application.csv

	SK_ID_PREV	SK_ID_CURR	NAME_CONTRACT_TYPE	AMT_ANNUITY	AMT_APPLICATION	AMT_CREDIT	AMT_DOWN_PAYMENT	AMT_GOODS_PRICE	WEEKDAY_APPR_PROCESS_START	HOUR_APPR_PROCESS_START	...
count	1.670214e+06	1.670214e+06		1670214	1.297979e+06	1.670214e+06	1.670213e+06	7.743700e+05	1.264699e+06		1.670214e+06
unique	NaN	NaN		4	NaN	NaN	NaN	NaN	NaN	7	NaN
top	NaN	NaN	Cash loans		NaN	NaN	NaN	NaN		TUESDAY	NaN
freq	NaN	NaN		747553	NaN	NaN	NaN	NaN		255118	NaN
mean	1.923089e+06	2.783572e+05		NaN	1.595512e+04	1.752339e+05	1.961140e+05	6.697402e+03	2.278473e+05	NaN	1.248418e+01
std	5.325980e+05	1.028148e+05		NaN	1.478214e+04	2.927798e+05	3.185746e+05	2.092150e+04	3.153966e+05	NaN	3.334028e+00
min	1.000001e+06	1.000010e+05		NaN	0.000000e+00	0.000000e+00	0.000000e+00	-9.000000e-01	0.000000e+00	NaN	0.000000e+00
25%	1.461857e+06	1.893290e+05		NaN	6.321780e+03	1.872000e+04	2.416050e+04	0.000000e+00	5.084100e+04	NaN	1.000000e+01
50%	1.923110e+06	2.787145e+05		NaN	1.125000e+04	7.104600e+04	8.054100e+04	1.638000e+03	1.123200e+05	NaN	1.200000e+01
75%	2.384280e+06	3.675140e+05		NaN	2.065842e+04	1.803600e+05	2.164185e+05	7.740000e+03	2.340000e+05	NaN	1.500000e+01
max	2.845382e+06	4.562550e+05		NaN	4.180581e+05	6.905160e+06	6.905160e+06	3.060045e+06	6.905160e+06	NaN	2.300000e+01

- 'AMT_ANNUITY'의 평균은 15,955 / 편차는 14,782 / 중위수는 11,250
- 'AMT_APPLICATION'의 평균은 175,233 / 편차는 292,779 / 중위수는 71,046
- 'AMT_CREDIT'의 평균은 196,114 / 편차는 318,574 / 중위수는 80,541

Describe

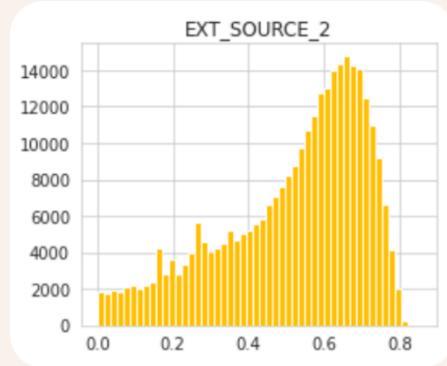
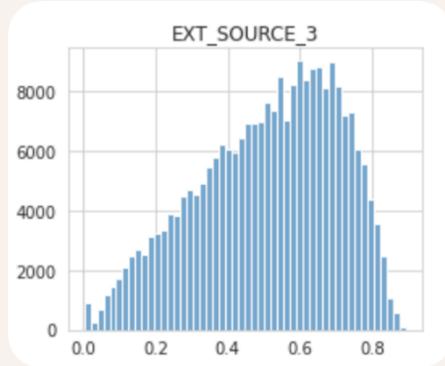
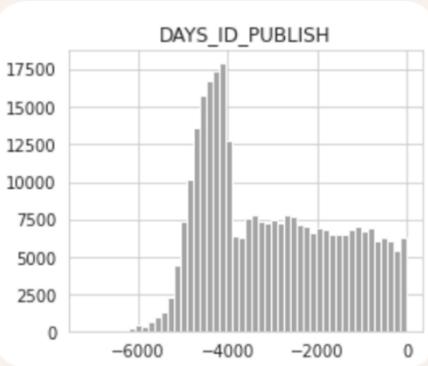
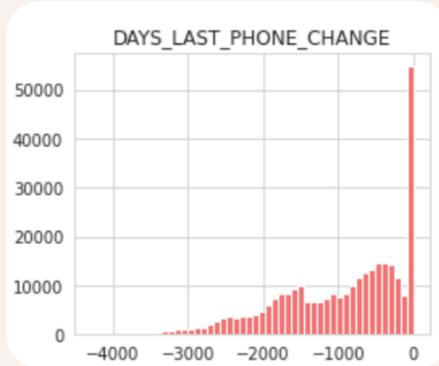
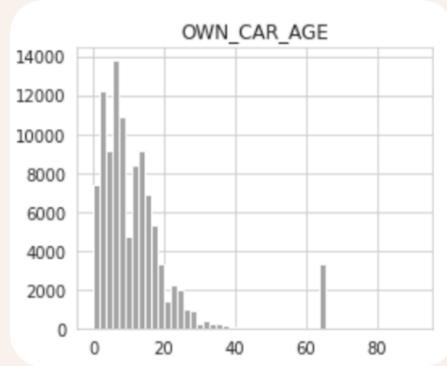
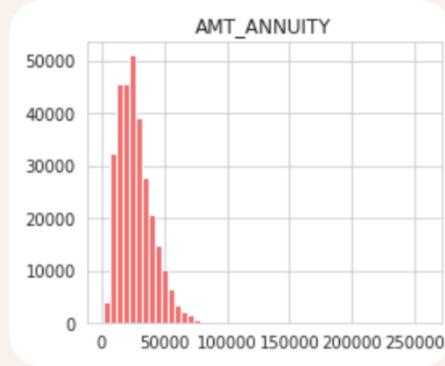
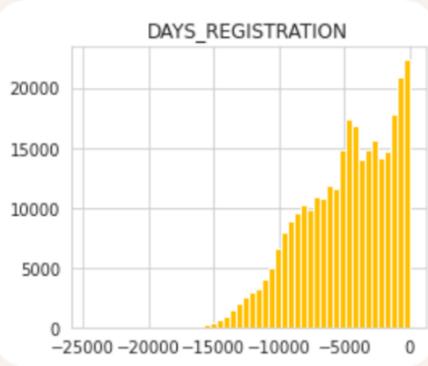
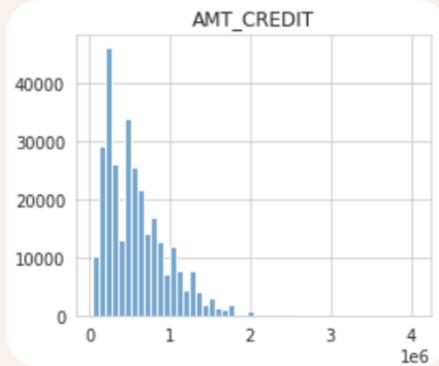
bureau.csv

	SK_ID_CURR	SK_ID_BUREAU	CREDIT_ACTIVE	CREDIT_CURRENCY	DAYES_CREDIT	...	AMT_CREDIT_SUM_LIMIT	AMT_CREDIT_SUM_OVERDUE	CREDIT_TYPE	DAYES_CREDIT_UPDATE	AMT_ANNUITY
count	1.716428e+06	1.716428e+06	1716428	1716428	1.716428e+06	...	1.124648e+06	1.716428e+06	1716428	1.716428e+06	4.896370e+05
unique	NaN	NaN	4	4	NaN	...	NaN	NaN	15	NaN	NaN
top	NaN	NaN	Closed	currency 1	NaN	...	NaN	NaN	Consumer credit	NaN	NaN
freq	NaN	NaN	1079273	1715020	NaN	...	NaN	NaN	1251615	NaN	NaN
mean	2.782149e+05	5.924434e+06	NaN	NaN	-1.142108e+03	...	6.229515e+03	3.791276e+01	NaN	-5.937483e+02	1.571276e+04
std	1.029386e+05	5.322657e+05	NaN	NaN	7.951649e+02	...	4.503203e+04	5.937650e+03	NaN	7.207473e+02	3.258269e+05
min	1.000010e+05	5.000000e+06	NaN	NaN	-2.922000e+03	...	-5.864061e+05	0.000000e+00	NaN	-4.194700e+04	0.000000e+00
25%	1.888668e+05	5.463954e+06	NaN	NaN	-1.666000e+03	...	0.000000e+00	0.000000e+00	NaN	-9.080000e+02	0.000000e+00
50%	2.780550e+05	5.926304e+06	NaN	NaN	-9.870000e+02	...	0.000000e+00	0.000000e+00	NaN	-3.950000e+02	0.000000e+00
75%	3.674260e+05	6.385681e+06	NaN	NaN	-4.740000e+02	...	0.000000e+00	0.000000e+00	NaN	-3.300000e+01	1.350000e+04
max	4.562550e+05	6.843457e+06	NaN	NaN	0.000000e+00	...	4.705600e+06	3.756681e+06	NaN	3.720000e+02	1.184534e+08

- ‘DAYES_CREDIT’ 의 평균은 -1142 / 편차는 +795 / 중위수는 -987
- ‘AMT_CREDIT_SUM_LIMIT’의 평균은 6,229 / 편차는 45,032 / 중위수는 0
- ‘AMT_ANNUITY’의 평균은 15,712 / 편차는 325,826 / 중위수는 0

Histogram

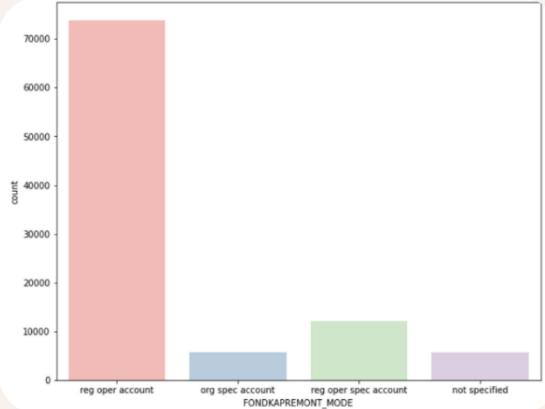
application_train.csv



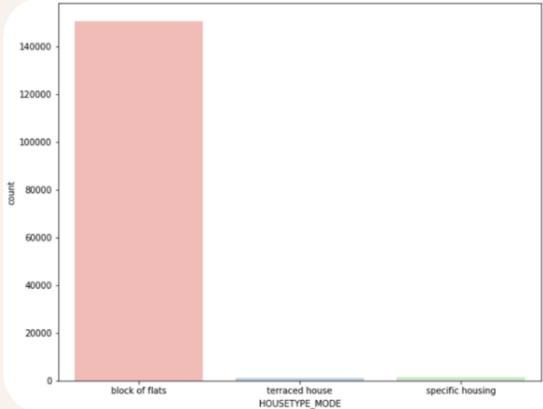
- 'AMT_CREDIT', 'AMT_ANNUITY', 'OWN_CAR_AGE' 는 오른쪽 꼬리 분포이고,
- 'DAY_REGISTRATION', 'DAY_LAST_PHONE_CHANGE', 'EXT_SOURCE_2'는 왼쪽 꼬리 분포이다.

Countplot

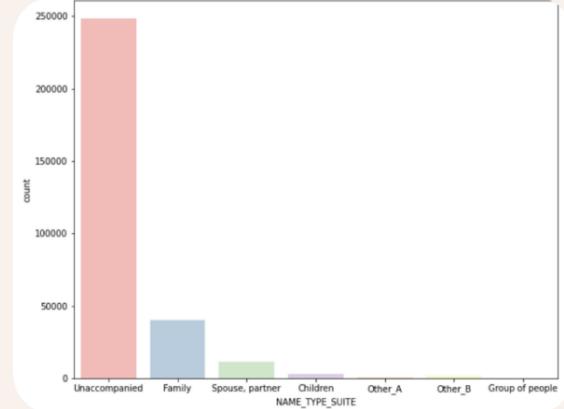
application_train.csv



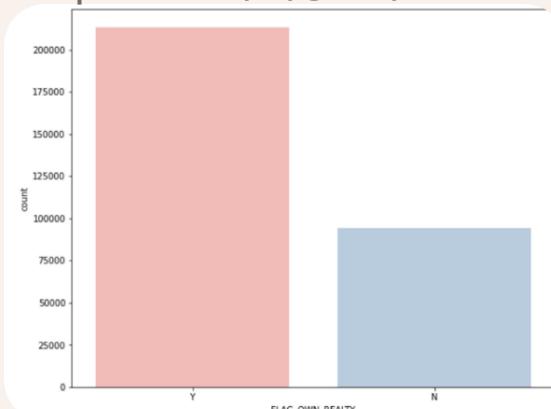
- 'reg oper account' 가 가장 많고, 'org spec account'와 'not specified' 가 가장 낮다.



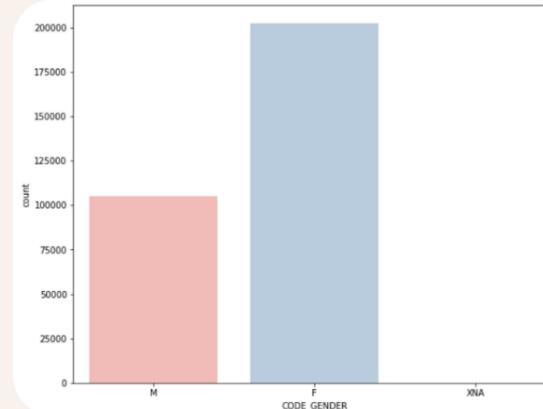
- 공동주택이 가장 많고,
- 연립주택이 가장 적다.



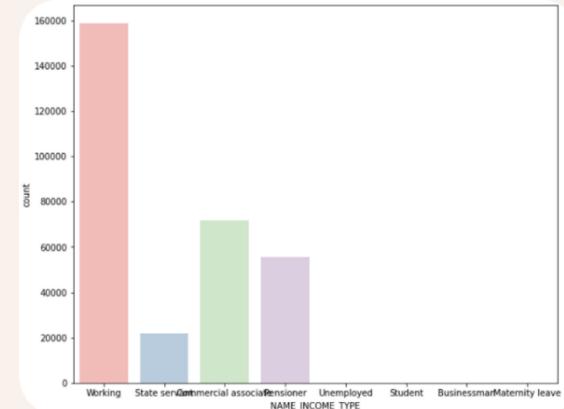
- 'Unaccompanied'가 가장 많고,
- 그 다음 'Family', 'Spouse, partner', 'Children' 순이다.



- 본인 소유의 부동산이 있는 경우가 더 많다.



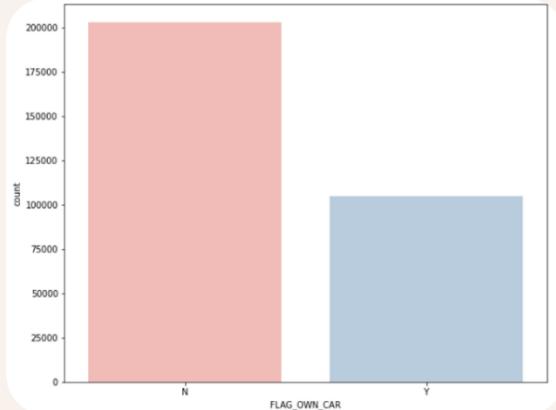
- 여성이 남성보다 더 많다.



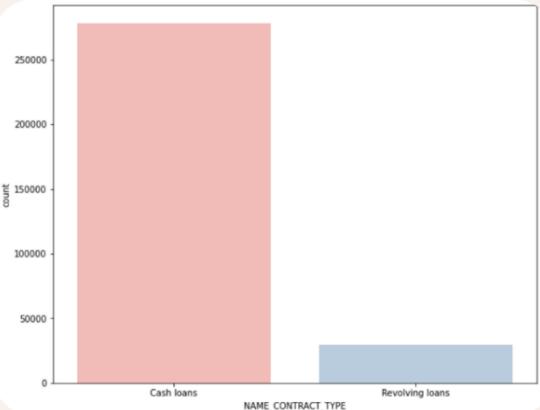
- 대출 신청자의 대부분은 소득 타입이 '근로'이며, '공무원 봉급'이 가장 적다.

Countplot

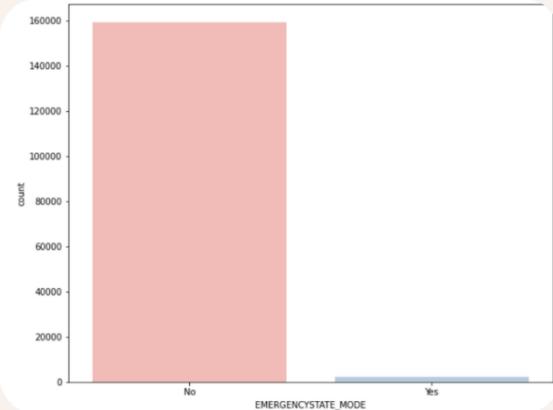
application_train.csv



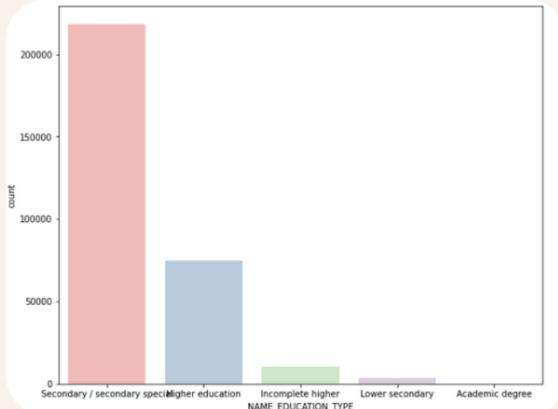
- 본인 소유의 차가 없는 경우가 더 많다.



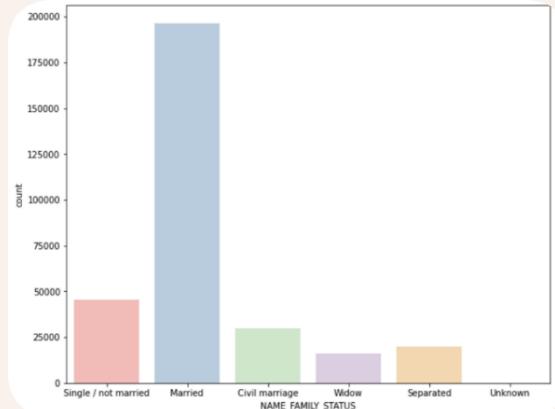
- 계약 타입이 'cash loans' 인 경우가 가장 많고, 'revolving loans' 이 적다.



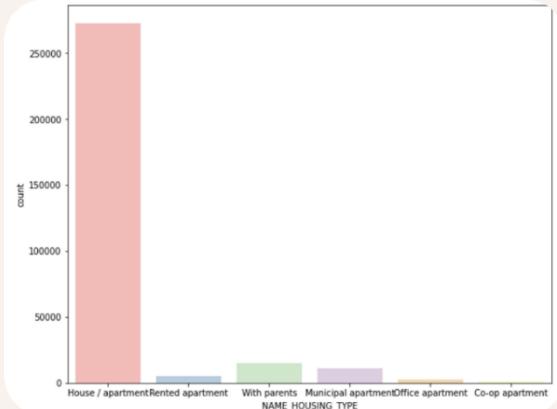
- 'EMERGENCYSTATE_MODE' 가 'No'인 경우가 더 많다.



- 대출 신청자의 대다수는 중등~고등 교육을 이수했다.



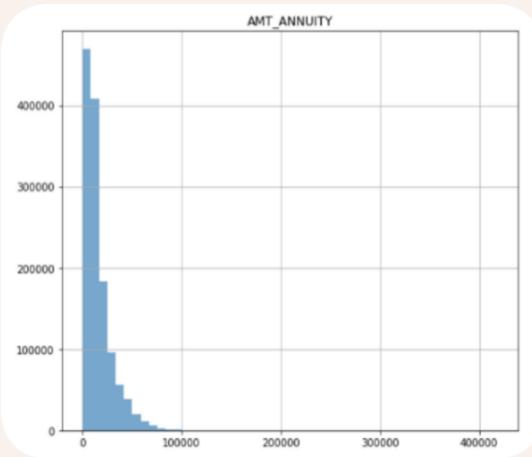
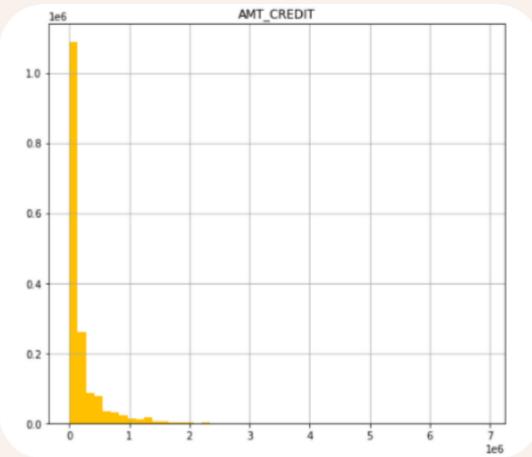
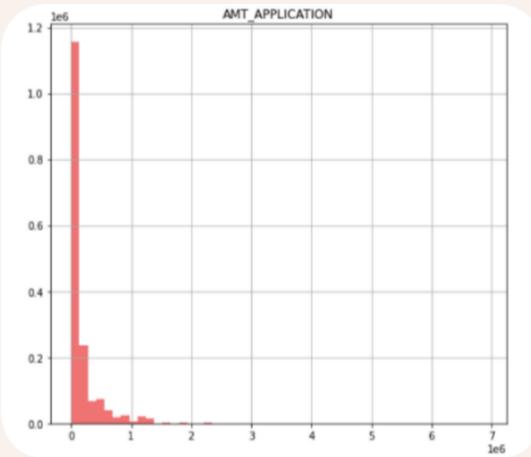
- 대부분은 결혼한 상태이고, 사별한 상태가 가장 적다.



- 집의 타입은 'House/apartment'가 가장 많다.

Histogram

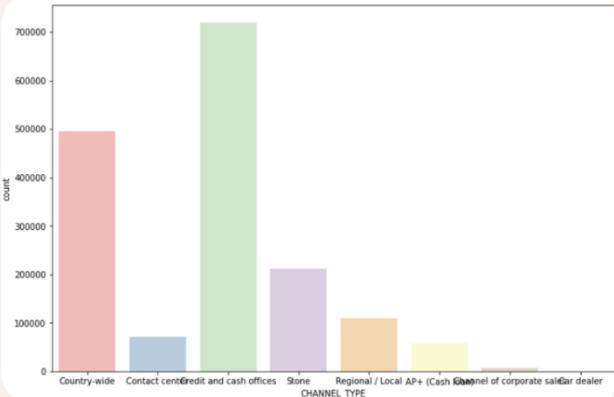
previous_application.csv



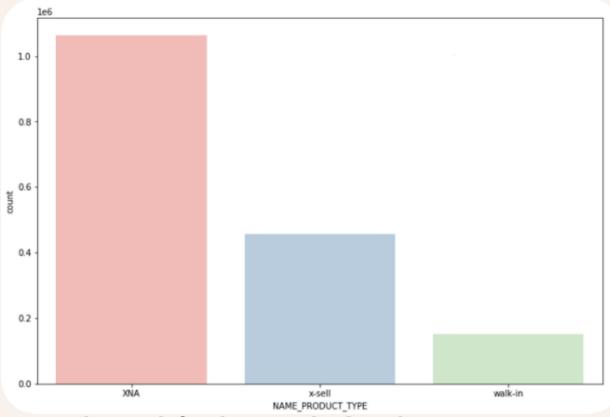
- 'AMT_APPLICATION', 'AMT_CREDIT', 'AMT_ANNUITY' 모두 오른쪽 꼬리 분포이다.

Countplot

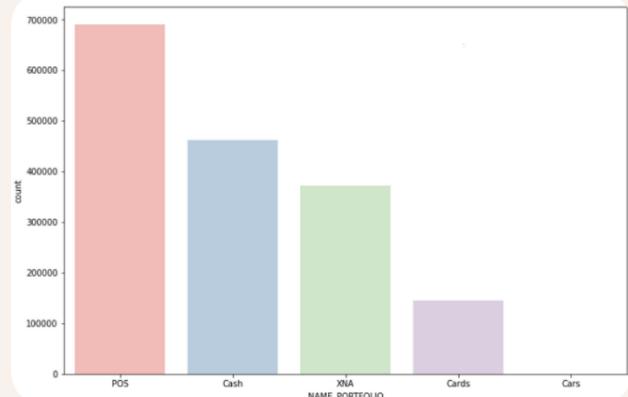
previous_application.csv



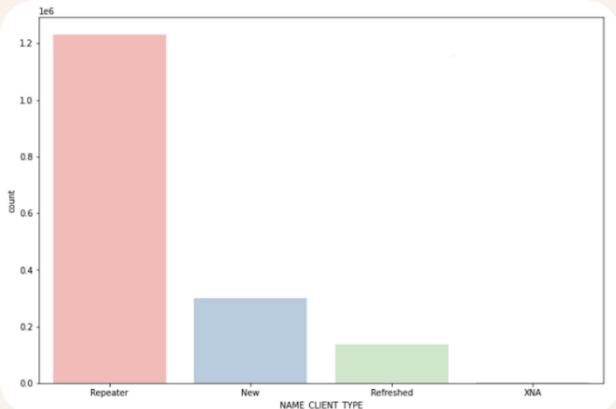
- 고객 유치 채널은 'credit and cash office'가 가장 많다.



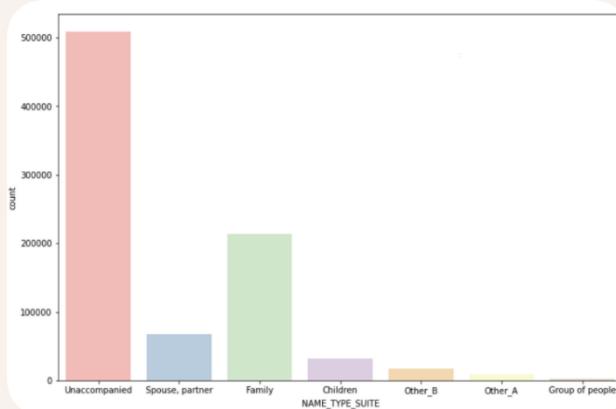
- 이전 대출에서 위험을 이미 평가받았던 고객이, 처음으로 대출을 받는 고객보다 더 많다.



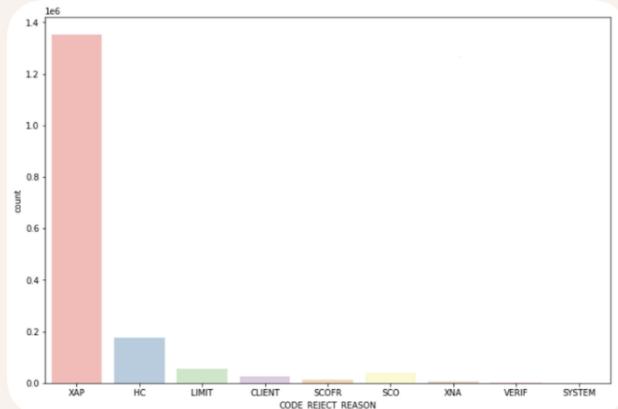
- 유가증권 보유 태입은 'POS'가 가장 많고, 'Card'가 가장 적다.



- 신규 고객보다 기존 고객이 더 많다.



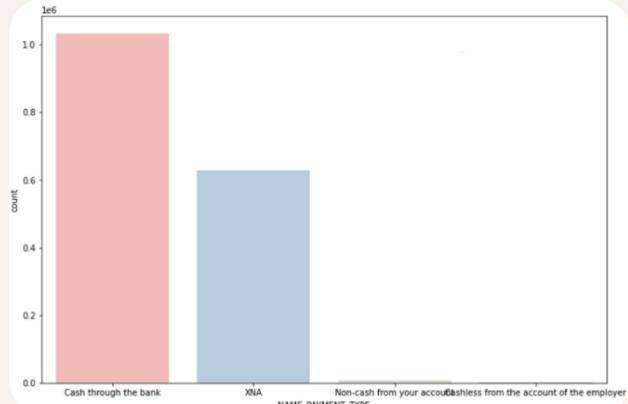
- 신청 시 동행인이 없는 경우가 가장 많고, 그 다음은 가족이 동행한 경우이다.



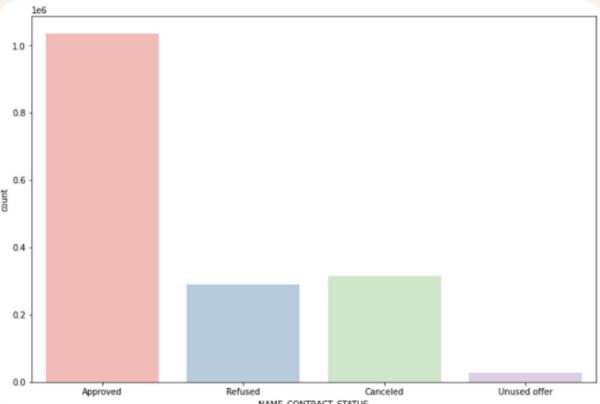
- 대출 거절 사유가 'HC'인 경우가 다른 경우보다 많다.

Countplot

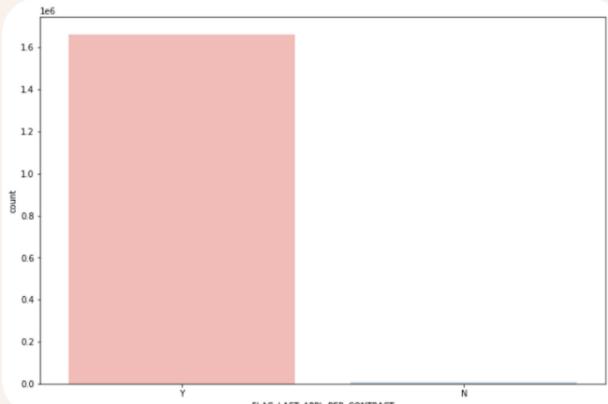
previous_application.csv



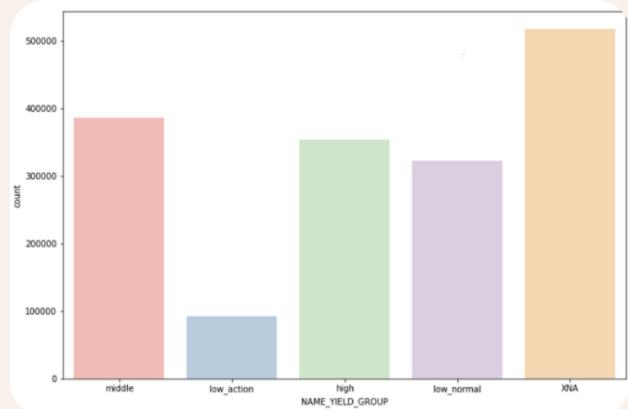
- 대출금 상환 방법은
무통장입금이 가장 많다.



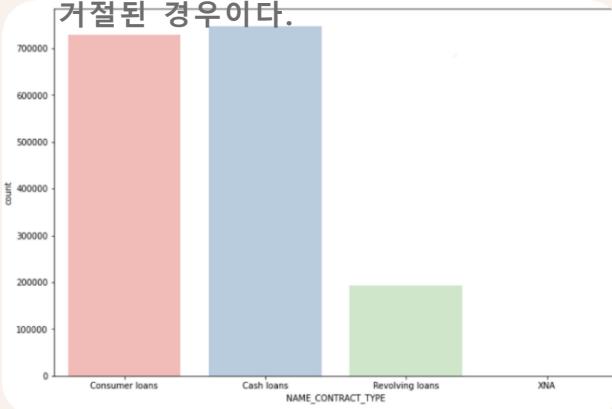
- 계약 상태는 승인된 경우가 가장
많고, 그 다음은 취소되거나
거절된 경우이다.



- 해당 계약의 마지막 지원서일 경우,
flag를 표시하는데, 'Y'가 더 많다.



- 이전 대출의 금리 그룹이 'middle',
'high', 'low_normal'의 순으로 많다.

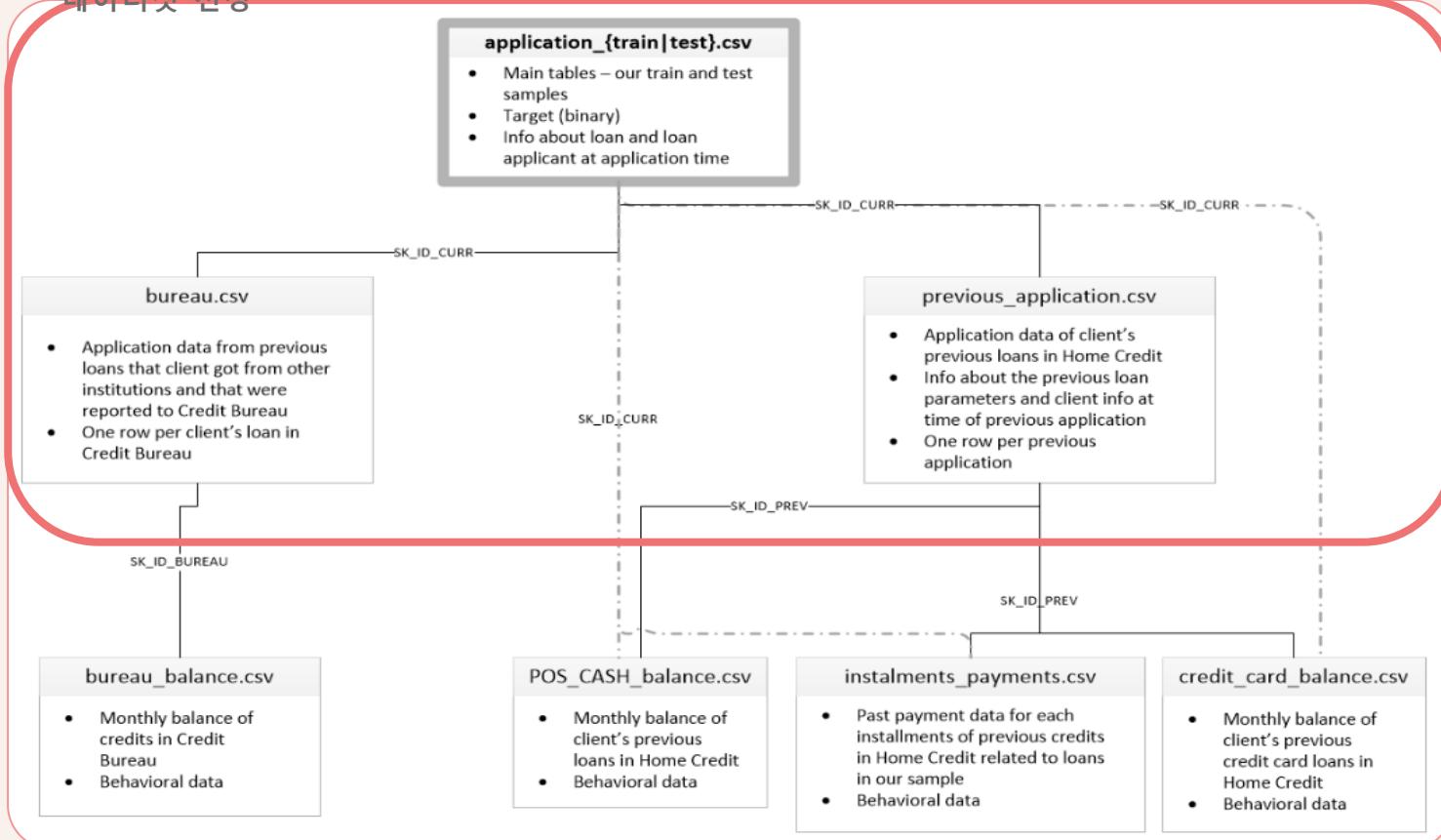


- 계약 타입이 'Cash loan'이 가장 많고,
그 다음은 'Consumer',
'Revolving' 순이다.

DataSet 선정 이유

- 기준이 되는 train 데이터와의 연관성과, 최종 데이터의 용량을 고려하여

데이터셋 선정



Outlier 확인 및 제거

모듈

- `get_outlier()` : column과 weight을 지정해 outlier index를 반환함

```
def get_outlier(df=None, column=None, weight=1.5):
    # target 값과 상관관계가 높은 열을 우선적으로 진행
    quantile_25 = np.percentile(df[column].values, 25)
    quantile_75 = np.percentile(df[column].values, 75)

    IQR = quantile_75 - quantile_25
    IQR_weight = IQR*weight

    lowest = quantile_25 - IQR_weight
    highest = quantile_75 + IQR_weight

    outlier_idx = df[column][(df[column] < lowest) | (df[column] > highest)].index
    return outlier_idx
```

- 범주형 데이터만 저장한 데이터프레임에 대한

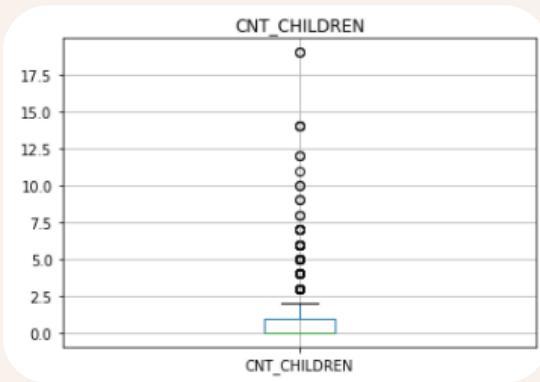
```
boxplot(df_train2.shape[1]):
    fig, axes = plt.subplots(1, 1, figsize=(6,4))
    bp = df_train2.boxplot(column=df_train2.columns[i])
    bp.set_title(df_train2.columns[i])
    plt.show()
```

- `application_train.csv` 파일의 이상치 제거 column 목록
 - CNT_CHILDREN
 - AMT_INCOME_TOTAL
 - AMT_CREDIT
 - DAYS_REGISTRATION
- `previous_application.csv` 파일은 column의 의미와 boxplot을 확인한 결과 이상치를 삭제하지 않음.
- `bureau.csv` 파일의 이상치 제거 column 목록
 - DAYS_CREDIT_UPDATE
 - DAYS_ENDDATE_FACT

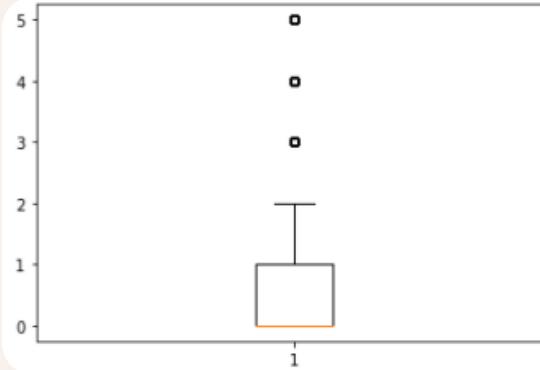
Outlier 확인 및 제거

application_train.csv

- application_train.csv : CNT_CHILDREN

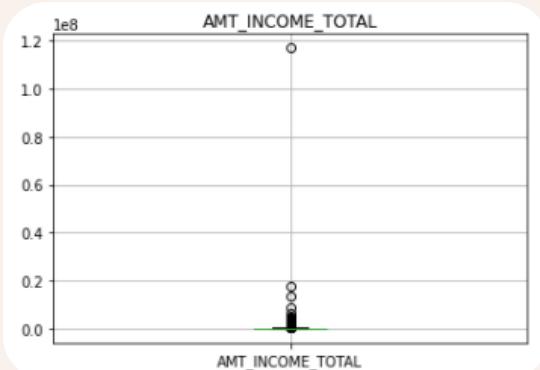


weight를 4.0으로
지정해 이상치 제거

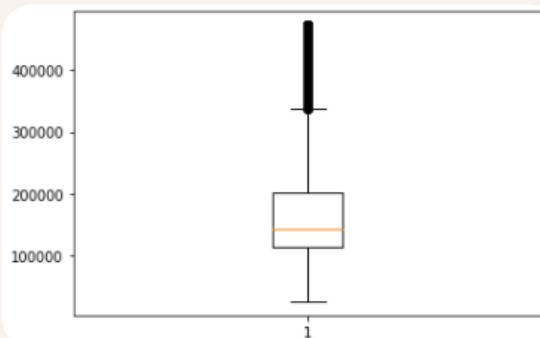


```
outlier_idx = get_outlier(df=df_train, column='CNT_CHILDREN', weight=4.0)
df_train.drop(outlier_idx, axis=0, inplace=True)
```

- application_train.csv : AMT_INCOME_TOTAL



weight를 3.0으로
지정해 이상치 제거

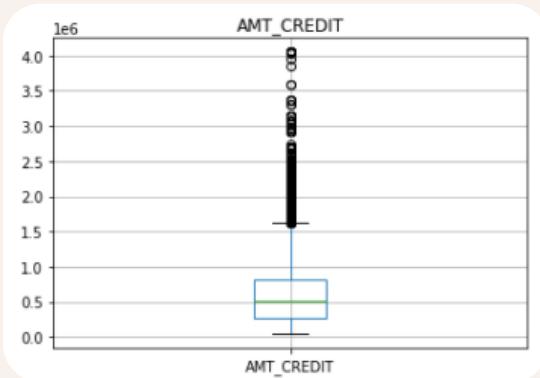


```
outlier_idx = get_outlier(df=df_train, column='AMT_INCOME_TOTAL', weight=3.0)
df_train.drop(outlier_idx, axis=0, inplace=True)
```

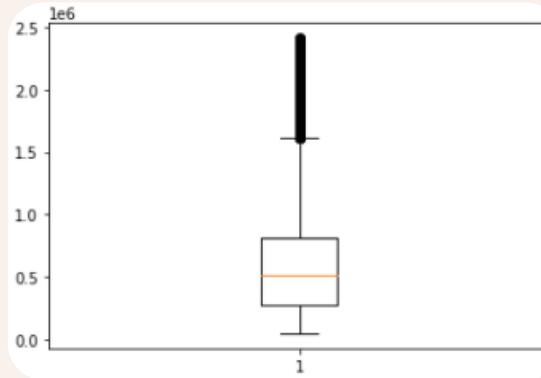
Outlier 확인 및 제거

application_train.csv

- application_train.csv : AMT_CREDIT

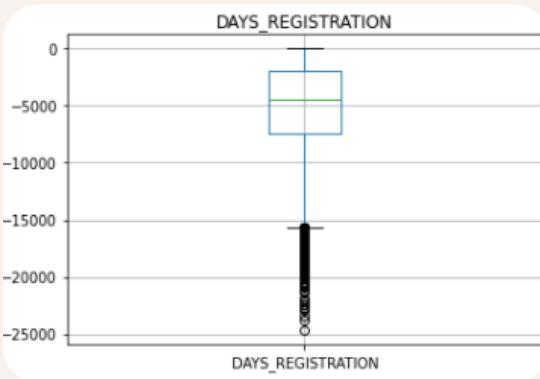


weight를 3.0으로
지정해 이상치 제거

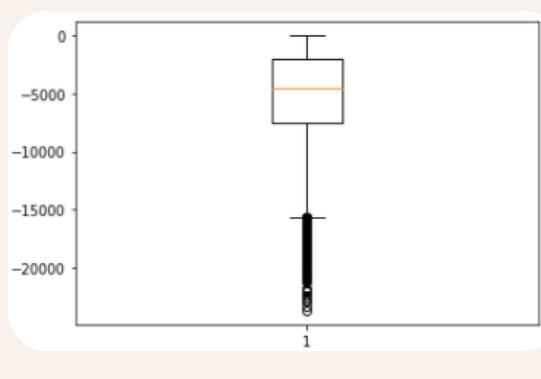


```
outlier_idx = get_outlier(df=df_train, column='AMT_CREDIT', weight=3.0)
df_train.drop(outlier_idx, axis=0, inplace=True)
```

- application_train.csv : DAYS_REGISTRATION



weight를 3.0으로
지정해 이상치 제거

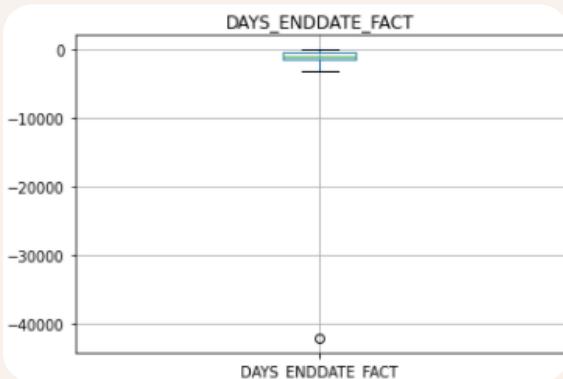


```
outlier_idx = get_outlier(df=df_train, column='DAYS_REGISTRATION', weight=3.0)
df_train.drop(outlier_idx, axis=0, inplace=True)
```

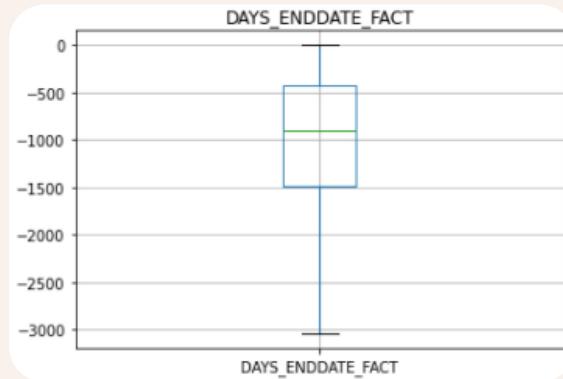
Outlier 확인 및 제거

bureau.csv

- bureau.csv : DAYS_ENDDATE_FACT

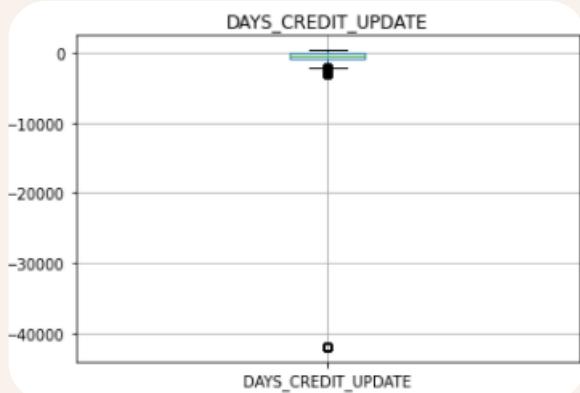


weight를 4.0으로
지정해 이상치 제거

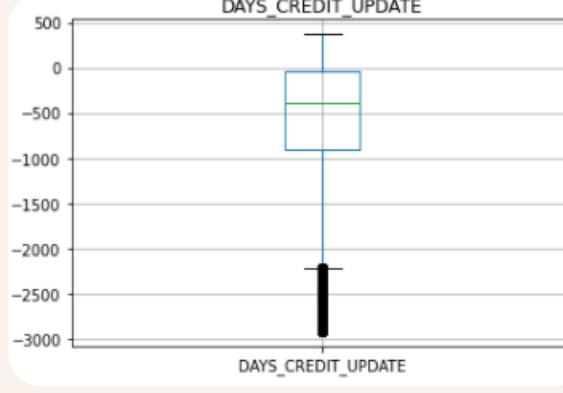


```
outlier_idx = get_outlier(df=df_bureau, column='DAYS_ENDDATE_FACT', weight=3.0)  
df_bureau.drop(outlier_idx, axis=0, inplace=True)
```

- bureau.csv : DAYS_CREDIT_UPDATE



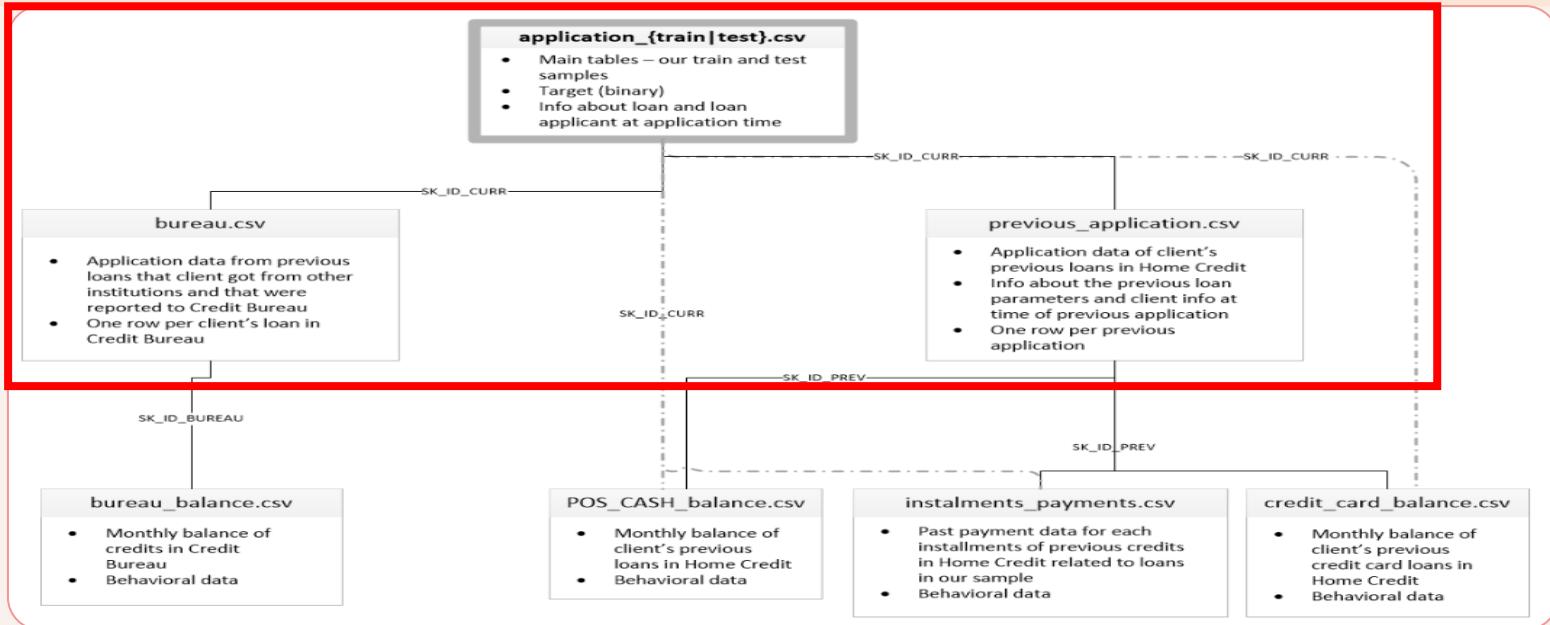
weight를 8.0으로
지정해 이상치 제거



```
outlier_idx = get_outlier(df=df_bureau, column='DAYS_CREDIT_UPDATE', weight=8.0)  
df_bureau.drop(outlier_idx, axis=0, inplace=True)
```

데이터 병합

application_train.csv, application_test.csv
bureau.csv, previous_application.csv



• 전체 데이터셋 양 확인

```

application_train - rows: 304088 columns: 122
application_test - rows: 48744 columns: 121
bureau - rows: 1716332 columns: 17
bureau_balance - rows: 27299925 columns: 3
credit_card_balance - rows: 3840312 columns: 23
installments_payments - rows: 13605401 columns: 8
previous_application - rows: 1670214 columns: 37
POS_CASH_balance - rows: 10001358 columns: 8

```

- 데이터 양 조절을 위해 임의적으로 상위 4가지 데이터셋 병합
- `application_train.csv` <- `bureau.csv`
`previous_application.csv`
- `application_test.csv` <- `bureau.csv`
`previous_application.csv`

데이터 병합

application_train.csv, application_test.csv
bureau.csv, previous_application.csv

- 도메인 지식을 활용하여 파생변수를 직접 생성

```
prev_id_cnt = previous_application.groupby(['SK_ID_CURR']).size()
new_col = pd.DataFrame(prev_id_cnt, columns=['CNT_PREV_CONTRACT'])

prev_approved = previous_application[previous_application['NAME_CONTRACT_STATUS'] == 'Approved']
prev_approved = prev_approved.groupby(['SK_ID_CURR']).size()
prev_approved = pd.DataFrame(prev_approved, columns=['CNT_PREV_APPROVED'])

new_col = pd.merge(new_col, prev_approved, left_on='SK_ID_CURR', right_on='SK_ID_CURR', how='left')

new_col['CNT_PREV_APPROVED'] = new_col['CNT_PREV_APPROVED'].fillna(0.0)

new_col['RATE_APPROVED'] = new_col['CNT_PREV_APPROVED'] / new_col['CNT_PREV_CONTRACT']

print(new_col)
```

SK_ID_CURR	CNT_PREV_CONTRACT	CNT_PREV_APPROVED	RATE_APPROVED
100001	1	1.0	1.00
100002	1	1.0	1.00
100003	3	3.0	1.00
100004	1	1.0	1.00
100005	2	1.0	0.50
...
456251	1	1.0	1.00
456252	1	1.0	1.00
456253	2	2.0	1.00
456254	2	2.0	1.00
456255	8	6.0	0.75

- bureau.csv와 previous_application.csv 파일을 병합해 application_train.csv 파일로 저장.
- bureau.csv와 previous_application.csv 파일을 병합해 application_test.csv 파일로 저장.
- 병합은 id 기준으로 하위 데이터셋에 어떤 feature가 높은 상관관계를 가질지 도메인 지식으로 추론해 봄.
- Ex) 승인률

파생변수

application_train.csv, application_test.csv
bureau.csv, previous_application.csv

```

numeric_df = df.select_dtypes('number')

# Group by the specified variable and calculate the statistics
agg = numeric_df.groupby(group_var).agg(['count', 'mean', 'max', 'min', 'sum']).reset_index()

# Select the categorical columns
categorical = pd.get_dummies(df.select_dtypes('object'))

# Groupby the group var and calculate the sum and mean
categorical = categorical.groupby(group_var).agg(['sum', 'mean'])

column_names = []

# Iterate through the columns in level 0
for var in categorical.columns.levels[0]:
    # Iterate through the stats in level 1
    for stat in ['count', 'count_norm']:
        # Make a new column name
        column_names.append('%s_%s_%s' % (df_name, var, stat))
    
```

	IL	IM
	prev_app_NAME_CONTRACT_STATUS_Approved_count	prev_app_NAME_CONTRACT_STATUS_Approved_count_norm
1		1
3		1
1		1
5		0.555555556
6		1
4		0.8
7		1
1		1
3		0.75
3		0.75
2		1
1		1
4		1
2		1
4		1
1		1
2		1
6		1
1		1
4		1

- 파생변수 생성을 함수화
- 연속형 변수는 'count', 'mean', 'max', 'min', 'sum' 으로 수치화
- 명목형 변수는 one-hot encoding 진행 후 'sum', 'mean'으로 1차 수치화 후 'count', 'count_norm'으로 2차 수치화

데이터 병합 결과

train_merged.csv, test_merged.csv

- 데이터 병합 함수

```
# Merging into one dataframe
train_merged, test_merged = module.merge_df(application_train, application_test, bureau, id, 'SK_ID_BUREAU', 'bureau')
train_merged, test_merged = module.merge_df(train_merged, test_merged, previous_application, id, 'SK_ID_PREV', 'prev_app')
```

- 병합 후 저장된 csv 파일

EU	EV	EW	EX	EY	EZ	FA	FB	FC	FD
prev_app_									
125	125	125	125	1	-25	-25	-25	-25	1
-1004.33	-386	-1980	-3013	3	-1054.33	-536	-1980	-3163	3
-694	-694	-694	-694	1	-724	-724	-724	-724	1
91584	365243	-215	366336	4	182477.5	365243	-425	729910	4
-837.2	346	-2056	-4186	5	72136.2	365243	-2056	360681	5
-1044.5	261	-2341	-4178	4	-1209.5	-69	-2341	-4838	4
-478.286	227	-1330	-3348	7	51666.86	365243	-1330	361668	7
-769	-769	-769	-769	1	-769	-769	-769	-769	1
120935.7	365243	-2147	362807	3	-1231	-289	-2147	-3693	3
-463.333	33	-1311	-1390	3	-563.333	-142	-1401	-1690	3
-198.5	86	-483	-397	2	182380	365243	-483	364760	2
-2090	-2090	-2090	-2090	1	-2090	-2090	-2090	-2090	1
-980.333	-327	-2009	-2941	3	-1050.33	-537	-2009	-3151	3
-1009.5	109	-2128	-2019	2	181557.5	365243	-2128	363115	2
-655.667	121	-1283	-1967	3	121057.7	365243	-1283	363173	3
-564	-564	-564	-564	1	-714	-714	-714	-714	1
-146	-1	-291	-292	2	182476	365243	-291	364952	2
72363.4	365243	-2622	361817	5	145338	365243	-2622	726690	5
301	301	301	301	1	365243	365243	365243	365243	1
90552.25	365243	-2123	362209	4	90372.25	365243	-2183	361489	4

- 데이터 병합 전/후 데이터 비교

```
=====BEFORE MERGE=====
Training Data Shape: (304088, 122)
Testing Data Shape: (48744, 121)
=====AFTER MERGE=====
Training Data Shape: (304088, 609)
Testing Data Shape: (48744, 608)
```

결측치 확인

train_ms.csv, test_ms.csv

- 결측치가 50%이상인 컬럼을 삭제하는 함수 호출(파라미터 : train set, test set, 50)

```
# Handling missing values on train, test dataset
train_ms, test_ms = module.handle_missing_values(train_merged, test_merged, 50)
```

• train

```
Your selected dataframe has 609 columns.
There are 554 columns that have missing values.

Missing Values % of Total Values
prev_app_RATE_INTEREST_PRIVILEGED_min 299533 98.5
prev_app_RATE_INTEREST_PRIVILEGED_max 299533 98.5
prev_app_RATE_INTEREST_PRIVILEGED_mean 299533 98.5
prev_app_RATE_INTEREST_PRIMARY_min 299533 98.5
prev_app_RATE_INTEREST_PRIMARY_mean 299533 98.5
prev_app_RATE_INTEREST_PRIMARY_max 299533 98.5
bureau_AMT_ANNUITY_max 225061 74.0
bureau_AMT_ANNUITY_min 225061 74.0
bureau_AMT_ANNUITY_mean 225061 74.0
COMMONAREA_AVG 213006 70.0
50
```

• test

```
Your selected dataframe has 608 columns.
There are 551 columns that have missing values.

Missing Values % of Total Values
prev_app_RATE_INTEREST_PRIMARY_min 47326 97.7
prev_app_RATE_INTEREST_PRIVILEGED_mean 47326 97.7
prev_app_RATE_INTEREST_PRIVILEGED_max 47326 97.7
prev_app_RATE_INTEREST_PRIVILEGED_min 47326 97.7
prev_app_RATE_INTEREST_PRIMARY_mean 47326 97.7
prev_app_RATE_INTEREST_PRIMARY_max 47326 97.7
COMMONAREA_AVG 33318 68.8
COMMONAREA_MEDI 33318 68.8
COMMONAREA_MODE 33318 68.8
NONLIVINGAPARTMENTS_MEDI 33169 68.5
35
```

- 결측치가 50%이상인 컬럼 수

```
There are 50 columns with more than 50% missing in either the training or testing data.
```

- ✓ train set의 결측치 : 50개
- ✓ test set의 결측치 : 35개
- ✓ test set의 결측치가 train set과 중복되어서 50%이상인 결측치가 50개가 됨

결측치 제거

train_ms.csv, test_ms.csv

- 발견된 결측 feature 완전 제거

```
# Drop the missing columns
train = train.drop(columns=missing_columns)
test = test.drop(columns=missing_columns)

return train, test
```

- 결측치 제거 전/후 데이터 비교

```
=====BEFORE HANDLING MISSING VALUE=====
Training Data Shape: (304088, 609)
Testing Data Shape: (48744, 608)
=====AFTER HANDLING MISSING VALUE=====
Training Data Shape: (304088, 559)
Testing Data Shape: (48744, 558)
```

상관계수

- 상위 10개 종속변수와 독립변수와의 양의 상관관계

	TARGET
TARGET	1.000000
bureau_DAYS_CREDIT_mean	0.089787
DAYS_BIRTH	0.078700
prev_app_NAME_CONTRACT_STATUS_Refused_count_norm	0.077664
bureau_CREDIT_ACTIVE_Active_count_norm	0.077410
bureau_DAYS_CREDIT_min	0.075434
bureau_DAYS_CREDIT_UPDATE_mean	0.073582
bureau_CREDIT_ACTIVE_Active_count	0.066772
prev_app_NAME_CONTRACT_STATUS_Refused_count	0.064401
prev_app_NAME_PRODUCT_TYPE_walk-in_count	0.062725

- 상위 10개 종속변수와 독립변수와의 음의 상관관계

	TARGET
FLOORSMAX_MEDI	-0.043012
FLOORSMAX_AVG	-0.043235
prev_app_DAYS_FIRST_DRAWING_min	-0.044915
DAYSPLOYED	-0.045362
prev_app_DAYS_FIRST_DRAWING_mean	-0.049054
prev_app_NAME_CONTRACT_STATUS_Approved_count_norm	-0.063426
prev_app_CODE_REJECT_REASON_XAP_count_norm	-0.073862
bureau_CREDIT_ACTIVE_Closed_count_norm	-0.079437
EXT_SOURCE_2	-0.160331
EXT_SOURCE_3	-0.179092

- 종속변수와 명목형 독립변수와의 상관관계

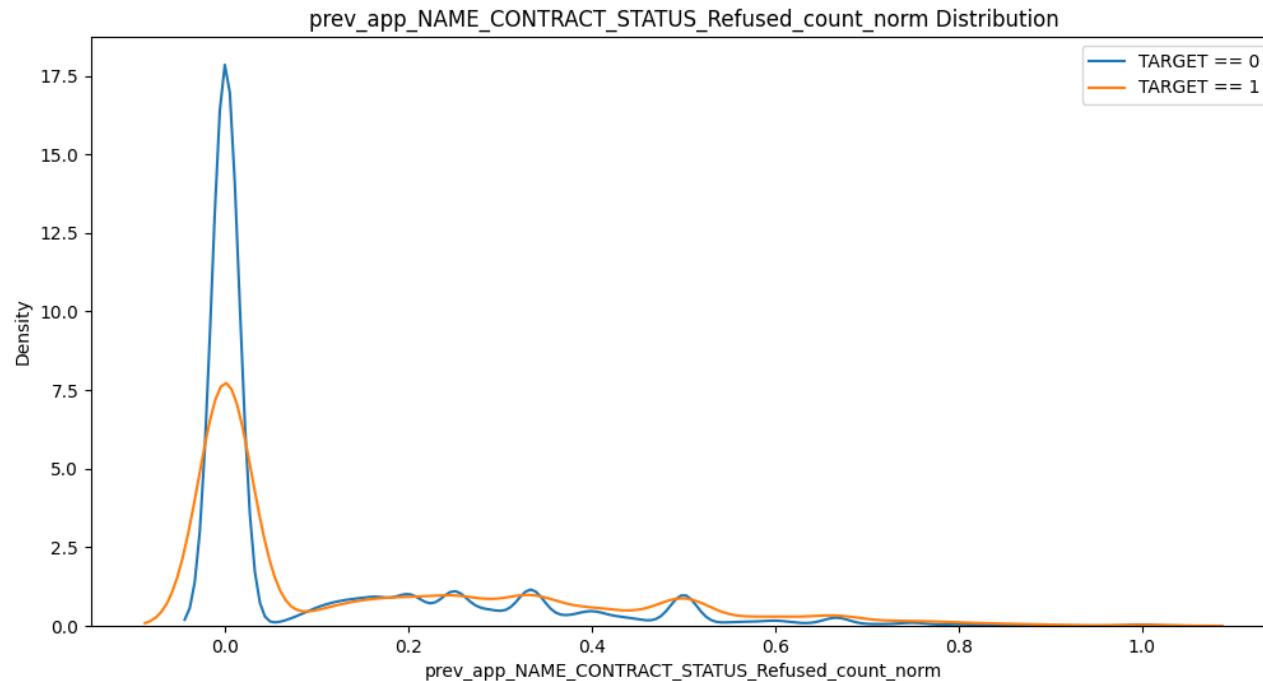
```

# 0 TARGET - NAME_CONTRACT_TYPE
카이 제곱 통계량 : 293.15054219964554
p-value : 1.0235150721172847e-65
# 1 TARGET - CODE_GENDER
카이 제곱 통계량 : 920.7913340361749
p-value : 1.1290217848908289e-200
# 2 TARGET - FLAG_OWN_CAR
카이 제곱 통계량 : 146.65601825576024
p-value : 9.330994431109667e-34
# 3 TARGET - FLAG_OWN_REALTY
카이 제곱 통계량 : 11.575827298084178
p-value : 0.0006681470317545887
# 4 TARGET - NAME_TYPE_SUITE
카이 제곱 통계량 : 32.825314655829914
p-value : 1.1329313903575907e-05
# 5 TARGET - NAME_INCOME_TYPE
카이 제곱 통계량 : 1253.470080924988
p-value : 1.9281456056858933e-266
# 6 TARGET - NAME_EDUCATION_TYPE
카이 제곱 통계량 : 1019.2131873088356
p-value : 2.4476812052198174e-219
# 7 TARGET - NAME_FAMILY_STATUS
카이 제곱 통계량 : 504.69408255632106
p-value : 7.744841561414258e-107
# 8 TARGET - NAME_HOUSING_TYPE
카이 제곱 통계량 : 420.55618983894664
p-value : 1.0990890032617707e-88
# 9 TARGET - OCCUPATION_TYPE
카이 제곱 통계량 : 1402.8467961927513
p-value : 3.784499856764699e-288
# 10 TARGET - WEEKDAY_APPR_PROCESS_START
카이 제곱 통계량 : 15.38755691304465
p-value : 0.017447369313895057
# 11 TARGET - ORGANIZATION_TYPE
카이 제곱 통계량 : 1609.2406359645206
p-value : 5.224541090300172e-299
# 12 TARGET - EMERGENCYSTATE_MODE
카이 제곱 통계량 : 23.678150227137568
p-value : 1.1386802431747463e-06

```

상관계수

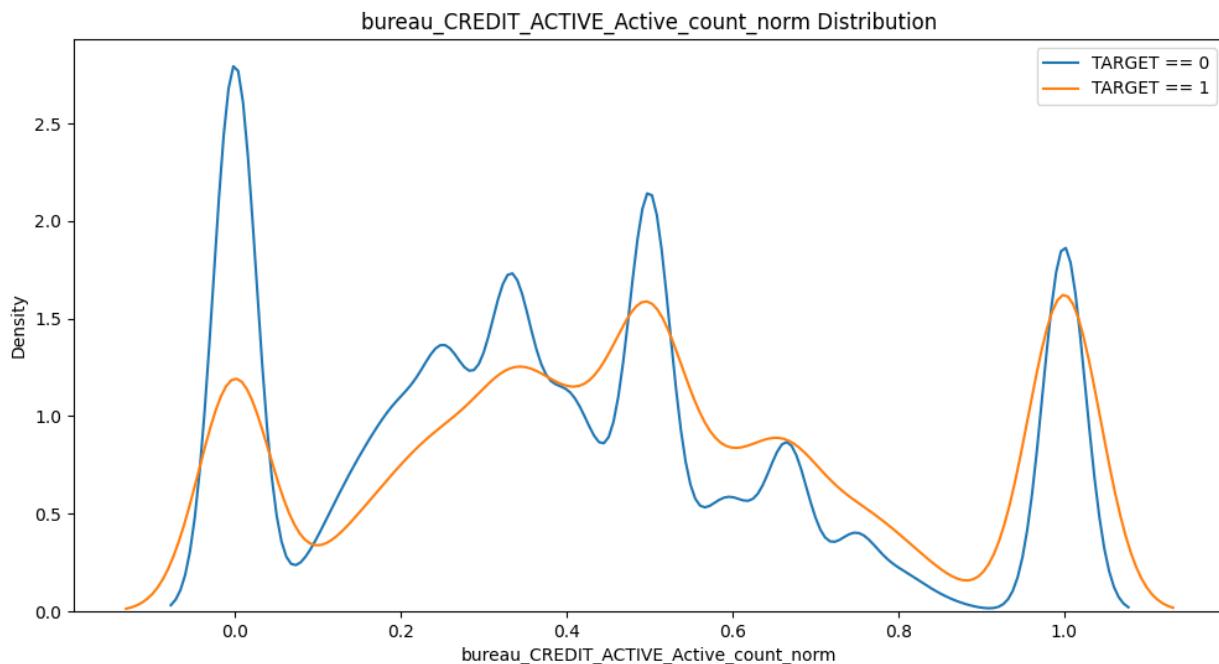
- EX1) '이전 대출 승인_거절횟수_정규화' - 밀도 Plot으로 시각화



```
The correlation between prev_app_NAME_CONTRACT_STATUS_Refused_count_norm and the TARGET is 0.0777
Median value for loan that was not repaid = 0.0000
Median value for loan that was repaid = 0.0000
```

상관계수

- EX2) '신용 활성_활성횟수_정규화' - 밀도 Plot으로 시각화



```
The correlation between bureau_CREDIT_ACTIVE_Active_count_norm and the TARGET is 0.0774
Median value for loan that was not repaid = 0.5000
Median value for loan that was repaid =      0.3636
```

다중공선성 임계치 설정

train_co.csv
test_co.csv

- 다중공선성 고려 임계치 0.7 이상인 Feature 삭제(중복 방지)

```
# Remove low correlated features in each dataset
train_co, test_co = module.corr_selection(train_ms, test_ms, 0.7)
```

```
train_corrs_removed = train.drop(columns=cols_to_remove)
test_corrs_removed = test.drop(columns=cols_to_remove)

return train_corrs_removed, test_corrs_removed
```

```
Number of columns to remove: 254
```

- 다중공선성 처리 전/후 데이터 비교

```
=====BEFORE HANDLING MULTICOLLINEARITY=====
Training Data Shape: (304088, 559)
Testing Data Shape: (48744, 558)
=====AFTER HANDLING MULTICOLLINEARITY=====
Training Data Shape: (304088, 305)
Testing Data Shape: (48744, 304)
```

Feature Engineering

train_fe.csv

test_fe.csv

- One-Hot Encoding/ Standard-Scaler 사용

```
train_fe, test_fe = module.feature_engineering(train_co, test_co, 'OneHotEncoder', 'StandardScaler')
```

- Id, label 탈락
- train/test 세트 concat
- 인코딩 후 다시 train/test 세트로 split

```
def feature_engineering(train, test, encoder, scaler):  
  
    label = train['TARGET']  
    id_train = train['SK_ID_CURR']  
    id_test = test['SK_ID_CURR']  
    # Remove the ids and target  
    train = train.drop(columns=['SK_ID_CURR', 'TARGET'])  
    test = test.drop(columns=['SK_ID_CURR'])  
  
    train, test = train.align(test, join='inner', axis=1)  
  
    train['training_set'] = True  
    test['training_set'] = False  
  
    df_full = pd.concat([train, test])
```

```
elif encoder == 'OneHotEncoder':  
  
    df_full = OneHotEnc(df_full)  
    train = df_full[df_full['training_set']==True]  
    train = train.drop('training_set', axis=1)  
    test = df_full[df_full['training_set'] == False]  
    test = test.drop('training_set', axis=1)
```

- Id, label 추가 후 반환

```
train = pd.concat([train, label], axis=1)  
train = pd.concat([id_train, train], axis=1)  
test = pd.concat([id_test, test], axis=1)  
return train, test
```

Feature Engineering

train_fe.csv

test_fe.csv

- Feature Engineering 전/후 데이터 비교

```
=====BEFORE FEATURE ENGINEERING=====  
Training Data Shape: (304088, 305)  
Testing Data Shape: (48744, 304)  
=====AFTER FEATURE ENGINEERING=====  
Training Data Shape: (304088, 418)  
Testing Data Shape: (48744, 417)
```

- 학습할 최종 데이터 크기

 test_final.csv	392,296KB
 train_final.csv	2,409,664KB

목차

CHAPTER 3. 모델링

1. 모델 선정 및 이유

2. 모델 최적화

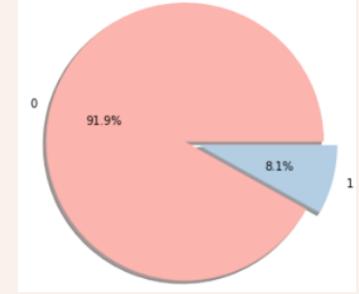
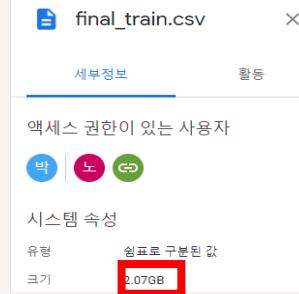
3. 모델 평가

모델 선정

데이터 특성

특성 고려

There are 448 columns that have missing values.		
	Missing Values	% of Total Values
prev_app_RATE_INTEREST_PRIVILEGED_mean	299533	98.5
prev_app_RATE_INTEREST_PRIMARY_mean	299533	98.5
prev_app_RATE_INTEREST_PRIVILEGED_min	299533	98.5
prev_app_RATE_INTEREST_PRIMARY_max	299533	98.5
prev_app_RATE_INTEREST_PRIMARY_min	299533	98.5
prev_app_RATE_INTEREST_PRIVILEGED_max	299533	98.5
COMMONAREA_MODE	213006	70.0
COMMONAREA_MEDI	213006	70.0
COMMONAREA_AVG	213006	70.0
NONLIVINGAPARTMENTS_AVG	211685	69.6
47		
There are 47 columns with more than 50% missing in either the training or testing data.		



- ✓ 결측치가 많아서 사용 가능한 모델이 제한적
- ✓ csv 파일의 크기가 2.07GB로 대용량
- ✓ 타겟의 분포가 91.9%와 8.1%로 불균형

고려 Model군

시간
고려

- Boosting
 - Adaboost
 - LightGBM
 - GBDT
 - Catboost
 - Xgboost

Model 선정

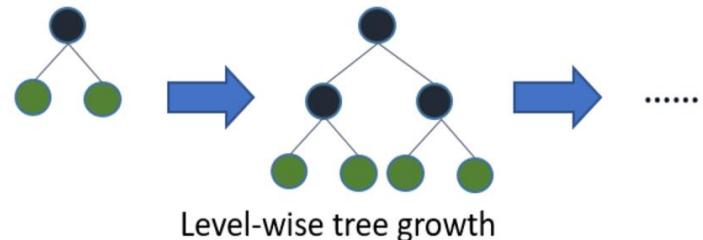
- LightGBM

LightGBM

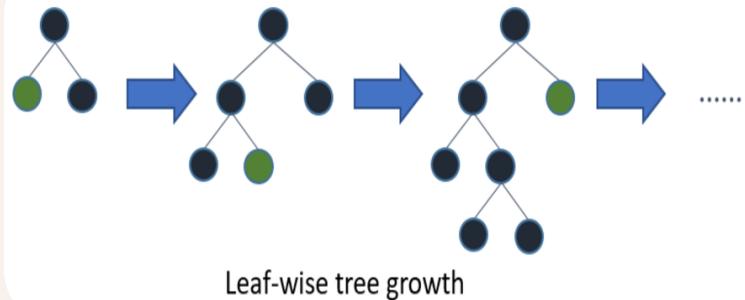
- LightGBM

- ✓ 가중치 업데이트를 하는데 경사하강법을 사용
- ✓ 오류값 = 실제값 - 예측값
- ✓ 분류의 실제값을 y , 피처(x_1, x_2, \dots, x_n)에 기반한 예측 함수를 $F(x)$ 로 설정
- ✓ 오류식 $\rightarrow h(x) = y - F(x)$
- ✓ $h(x)$ 를 최소화하는 방향성을 가지고 가중치를 업데이트
- ✓ 속도 개선(LightGBM의 장점)
- ✓ 리프 기반 훈련을 통한 성능 향상 기대

- 기존 Level 기반 boosting 기법

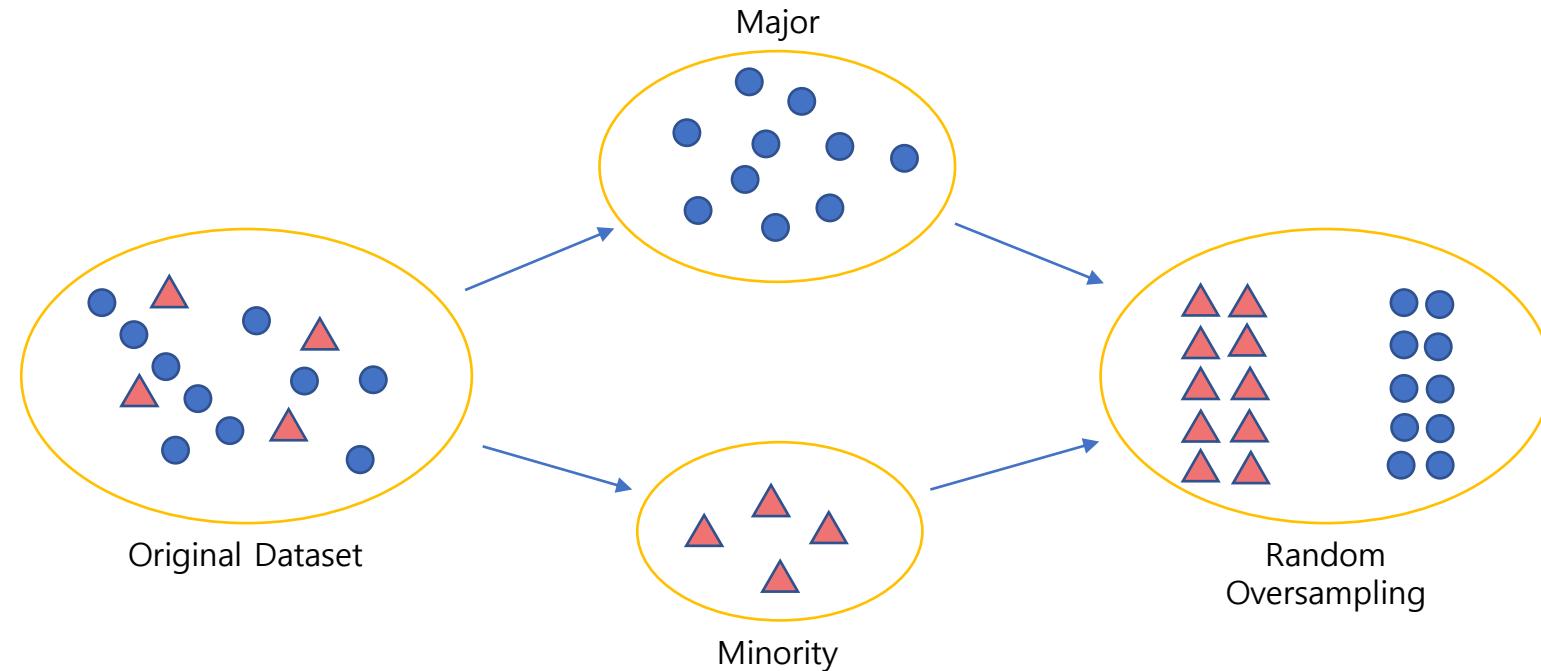


- LightGBM의 leaf 기반 boosting 기법



RandomOverSampling

- ROSE 기법* 사용
 - 불균형 완화



*2014년에 Menardi와 Torelli가 제안한 ROSE(random over sampling examples)(Menardi & Torelli, 2014)기법 참고
<https://cran.r-project.org/web/packages/ROSE/ROSE.pdf>

최종 모델

- 랜덤오버샘플링을 활용한 LightGBM

```

def model(train, test, id, label, n_folds=5):
    features = np.array(train)
    test_features = np.array(test)

    # Extract feature names
    feature_names = list(train.columns)

    # Empty array for feature importances
    feature_importance_values = np.zeros(len(feature_names))

    # Create the kfold object
    k_fold = StratifiedKFold(n_splits=n_folds, shuffle=True, random_state=50)
    X = train.drop([id], 1)
    X = train.drop([label], 1)
    y = train[label]

    ros = RandomOverSampler(random_state=0)
    X_resampled, y_resampled = ros.fit_resample(X, y)

    X_train, X_test, y_train, y_test = train_test_split(X_resampled, y_resampled, test_size=0.2, random_state=121)
    eval_set = [(X_test, y_test)]

    model = lgb.LGBMClassifier(num_leaves=36, bagging_fraction=0.5101, max_depth=8, lambda_l1=3.891, lambda_l2=2.61,
                               min_split_gain=0.05, min_child_weight=40.96, colsample_bytree=0.933)

    model.fit(X_train, y_train, early_stopping_rounds=100, eval_metric="auc", eval_set=eval_set)

```

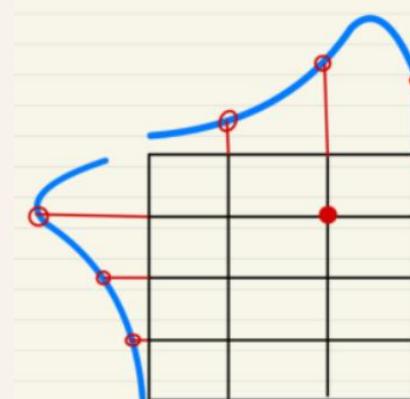
- RandomOverSampler를 활용해 Feature와 Target 1:1 비율로 오버샘플링
- LGMB 모델 생성 후 fit 실행

모델 최적화

- 모델 최적화에 사용된 기법 - 베이지안

최적화

	Grid Search	Bayesian Optimization
입력값	일정 간격	범위
파라미터 조합	모든 경우의 수	예상되는 최고의 성능
특징	파라미터를 추가할 수록 기하급수적으로 늘어나는 수행시간	학습을 통해 성능이 향상될 가능성이 있는 파라미터 값을 선택적으로 수행



<Grid Search>



<Bayesian optimization>

- Grid Search의 경우, 각 파라미터의 단일 값들의 조합으로 진행되기에 파라미터를 추가할수록 수행 시간이 기하급수적으로 증가하는 단점이 있음.
- 베이지안 최적화의 경우, 확률적으로 높은 성능을 보일 파라미터 값을 스스로 찾아가기에 수행시간 단축 가능

모델 최적화

- LGBM 파라미터 설명

구분	파라미터명	설명	기본값	범위	설정값
성능	num_leaves	Maximum leaves for each trained tree.	31	[1, ∞]	[24, 40]
	max_depth	Maximum depth of each trained tree.	-1	[1, ∞]	[5, 8]
과적합	lambda_l1	L1 Regularization for boosting.	0	[0, ∞]	[0, 5]
	lambda_l2	L1 Regularization for boosting.	0	[0, ∞]	[0, 3]
	min_split_gain	Prune by minimum loss requirement.	0	[0, ∞]	[0.01, 0.1]
	min_child_weight	Prune by minimum hessian requirement.	10	[0, ∞]	[5, 50]
속도	bagging_function	Percentage of rows used per iteration frequency.	1	[0, 1]	[0.5, 1]
	colsample_bytree	Percentage of columns used per iteration.	1	[0, 1]	[0.6, 1]

- Documents를 통해 파라미터 파악 후 필요한 파라미터 임의 선택.
- 일반적으로 learning_rate의 경우 범위가 아닌 값으로 fix하고 진행.

모델 최적화

- Bayesian optimization Code

```

1 def bayes_parameter_opt_lgb(X, y, init_round=15, opt_round=25, n_folds=5, random_seed=6, n_estimators=10000, learning_rate=0.05, output_process=False):
2     # prepare data
3     train_data = lgb.Dataset(data=X, label=y, free_raw_data=False)
4     # parameters
5     def lgb_eval(num_leaves, bagging_fraction, max_depth, lambda_l1, lambda_l2, min_split_gain, min_child_weight, colsample_bytree):
6         params = {'application': 'binary', 'num_iterations': n_estimators, 'learning_rate':learning_rate, 'early_stopping_round':100, 'metric':'auc'}
7         params["num_leaves"] = int(round(num_leaves))
8         params['bagging_fraction'] = max(min(bagging_fraction, 1), 0)
9         params['max_depth'] = int(round(max_depth))
10        params['lambda_l1'] = max(lambda_l1, 0)
11        params['lambda_l2'] = max(lambda_l2, 0)
12        params['min_split_gain'] = min_split_gain
13        params['min_child_weight'] = min_child_weight
14        params['colsample_bytree']= colsample_bytree
15        cv_result = lgb.cv(params, train_data, nfold=n_folds, seed=random_seed, stratified=True, verbose_eval =200, metrics=['auc'])
16        return max(cv_result['auc-mean'])
17
18    # range
19    lgbBO = BayesianOptimization(lgb_eval, {'num_leaves': (24, 40), # 성능 향상
20                                    'max_depth': (5, 8), # 성능 향상
21                                    'lambda_l1': (0, 5), # 과적합 방지
22                                    'lambda_l2': (0, 3), # 과적합 방지
23                                    'min_split_gain': (0.01, 0.1), # 과적합 방지
24                                    'min_child_weight': (5, 50), # 과적합 방지
25                                    'bagging_fraction': (0.5, 1), # 속도 향상
26                                    'colsample_bytree': (0.6, 1)}, # 속도 향상
27                                    random_state=0)
28
29    # optimize
30    lgbBO.maximize(init_points=init_round, n_iter=opt_round)

```

- Cross Validation을 Bayesian optimization에 적용.
- 최적의 파라미터 값을 찾을 범위를 입력.

모델 최적화

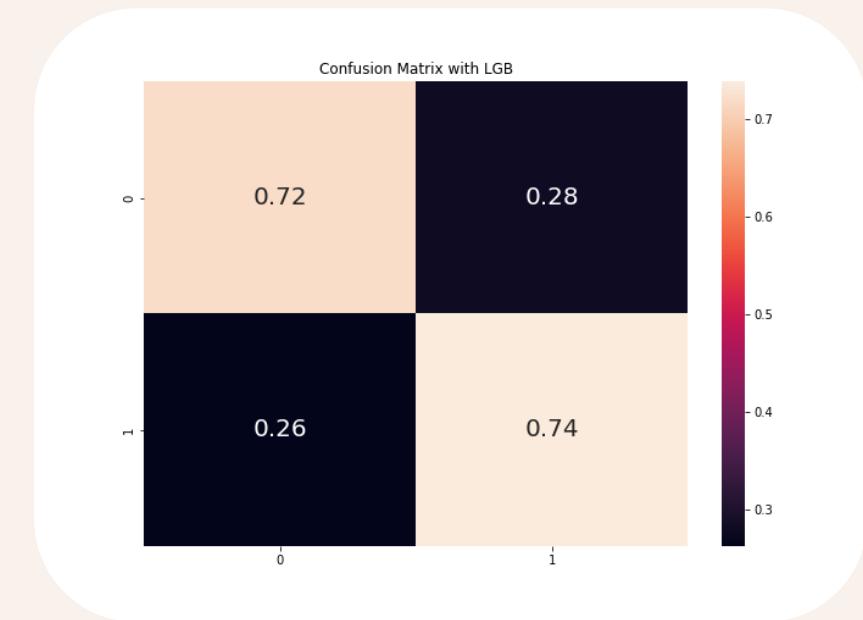
- Bayesian optimization Code 실행 output

iter	target	baggin...	colsam...	lambda_l1	lambda_l2	max_depth	min_ch...	min_sp...	num_le...
[200]	cv_agg's auc: 0.760764 + 0.0034519								
[400]	cv_agg's auc: 0.763263 + 0.00343982								
[600]	cv_agg's auc: 0.763185 + 0.0035157								
1	0.7634	0.7744	0.8861	3.014	1.635	6.271	34.07	0.04938	38.27
[200]	cv_agg's auc: 0.761303 + 0.00332476								
[400]	cv_agg's auc: 0.763582 + 0.0030559								
[600]	cv_agg's auc: 0.763813 + 0.00293282								
2	0.7639	0.9818	0.7534	3.959	1.587	6.704	46.65	0.01639	25.39
[200]	cv_agg's auc: 0.762050 + 0.00355955								
[400]	cv_agg's auc: 0.763428 + 0.00334608								
3	0.7635	0.5101	0.933	3.891	2.61	7.936	40.96	0.05153	36.49
[200]	cv_agg's auc: 0.7616 + 0.00329298								
[400]	cv_agg's auc: 0.763308 + 0.00299344								
4	0.7634	0.5591	0.856	0.7168	2.834	6.566	23.66	0.03381	36.39
[200]	cv_agg's auc: 0.761641 + 0.00334338								
[400]	cv_agg's auc: 0.763225 + 0.00316224								
5	0.7633	0.7281	0.8274	0.09395	1.853	6.836	32.76	0.09494	34.91
[200]	cv_agg's auc: 0.761619 + 0.00372874								
[400]	cv_agg's auc: 0.76387 + 0.00343012								
6	0.7642	0.6798	0.7748	3.488	0.1807	7.0	35.18	0.02893	26.06
[200]	cv_agg's auc: 0.761074 + 0.00310552								
[400]	cv_agg's auc: 0.7637 + 0.00272144								
7	0.7638	0.6577	0.7455	2.851	1.316	7.965	9.592	0.0288	26.58
[200]	cv_agg's auc: 0.760098 + 0.00311603								
[400]	cv_agg's auc: 0.763075 + 0.00292358								
[600]	cv_agg's auc: 0.763485 + 0.00298545								
8	0.7636	0.8266	0.7013	2.332	0.7333	5.477	9.967	0.06907	26.21

모델 평가

- Confusion matrix

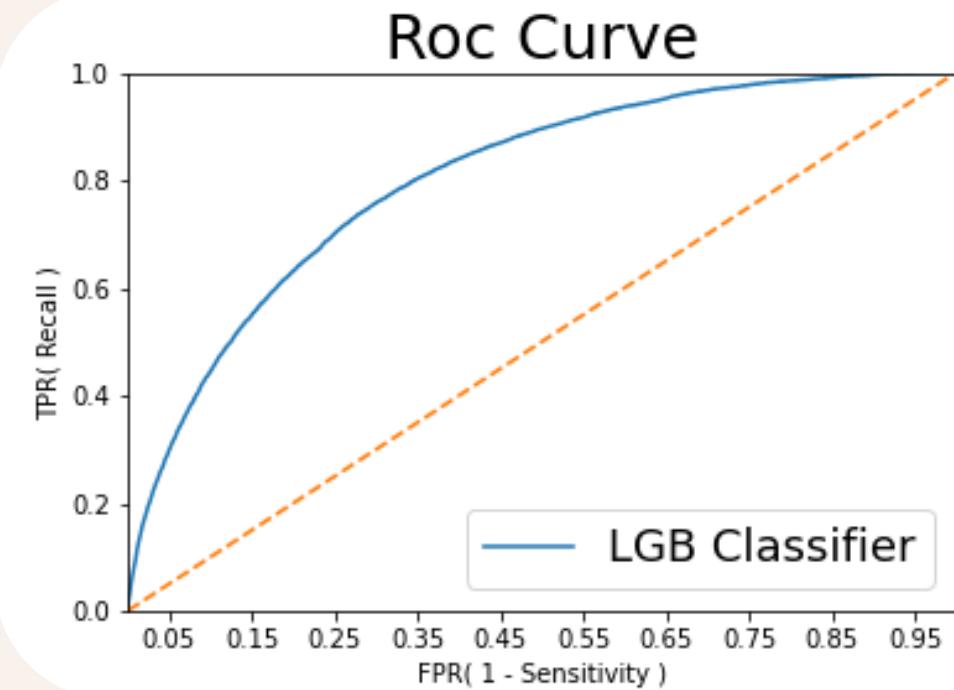
```
===== Result =====
Best score : 0.728885
Best precision score : 0.723813
Best recall score : 0.738004
Best F1 score : 0.73084
Best AUC : 0.8029235378567459
```



- Recall : 상환이 불가능한 고객을 상환이 불가능하다고 예측한 비율.
- Precision : 상환이 불가능하다고 예측했는데, 실제 상환이 불가능한 비율.

모델 평가

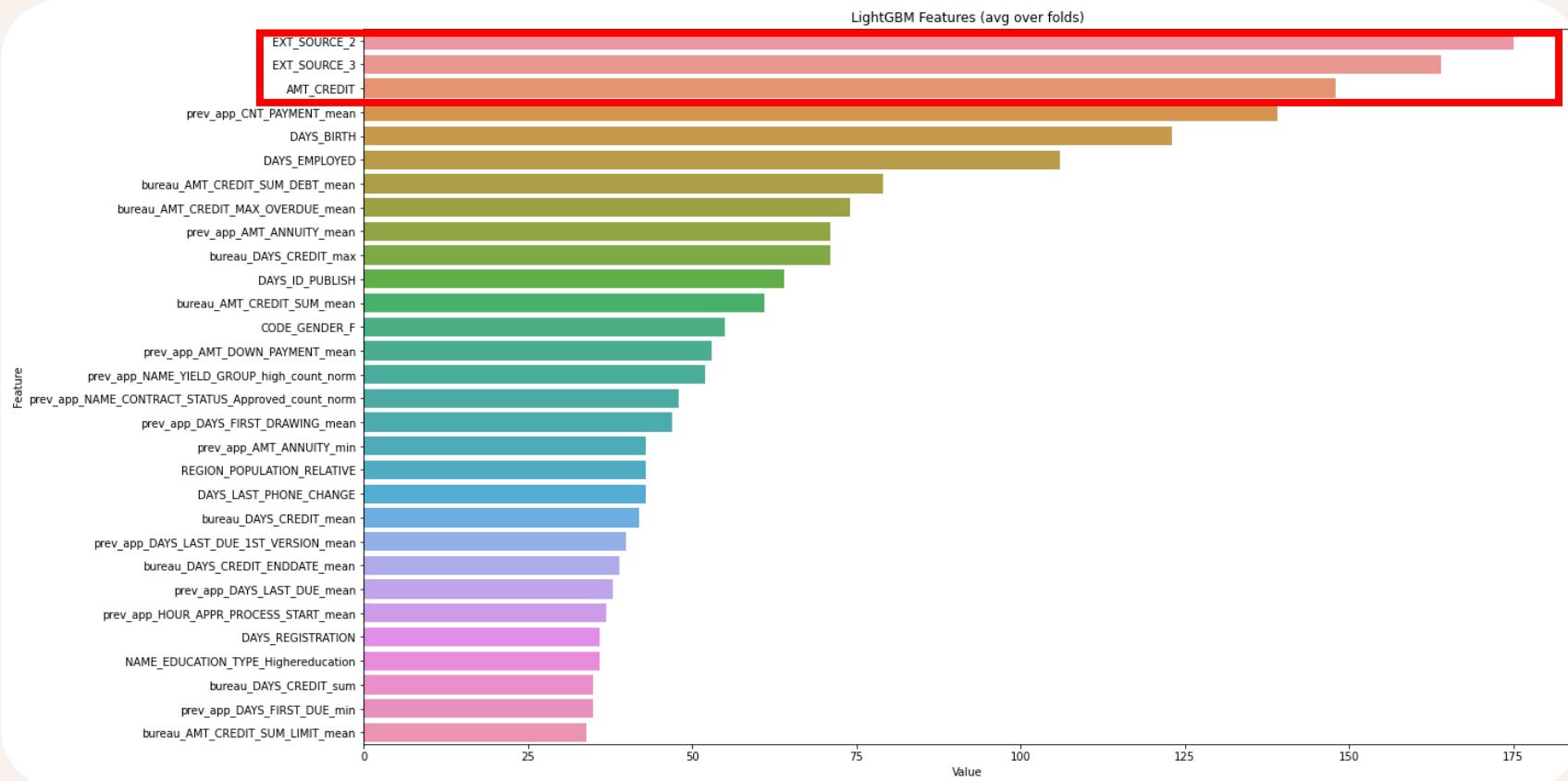
- ROC커브



- 임계치에 따라 변화하는 재현율의 점들을 이은 곡선
- 좌측 상단으로 곡선이 가까워질수록 좋은 모델
- 해당 데이터 셋으로 진행된 캐글 대회의 경우 ROC커브의 하단 면적을 의미하는 'AUC'를 평가 지표로 사용.

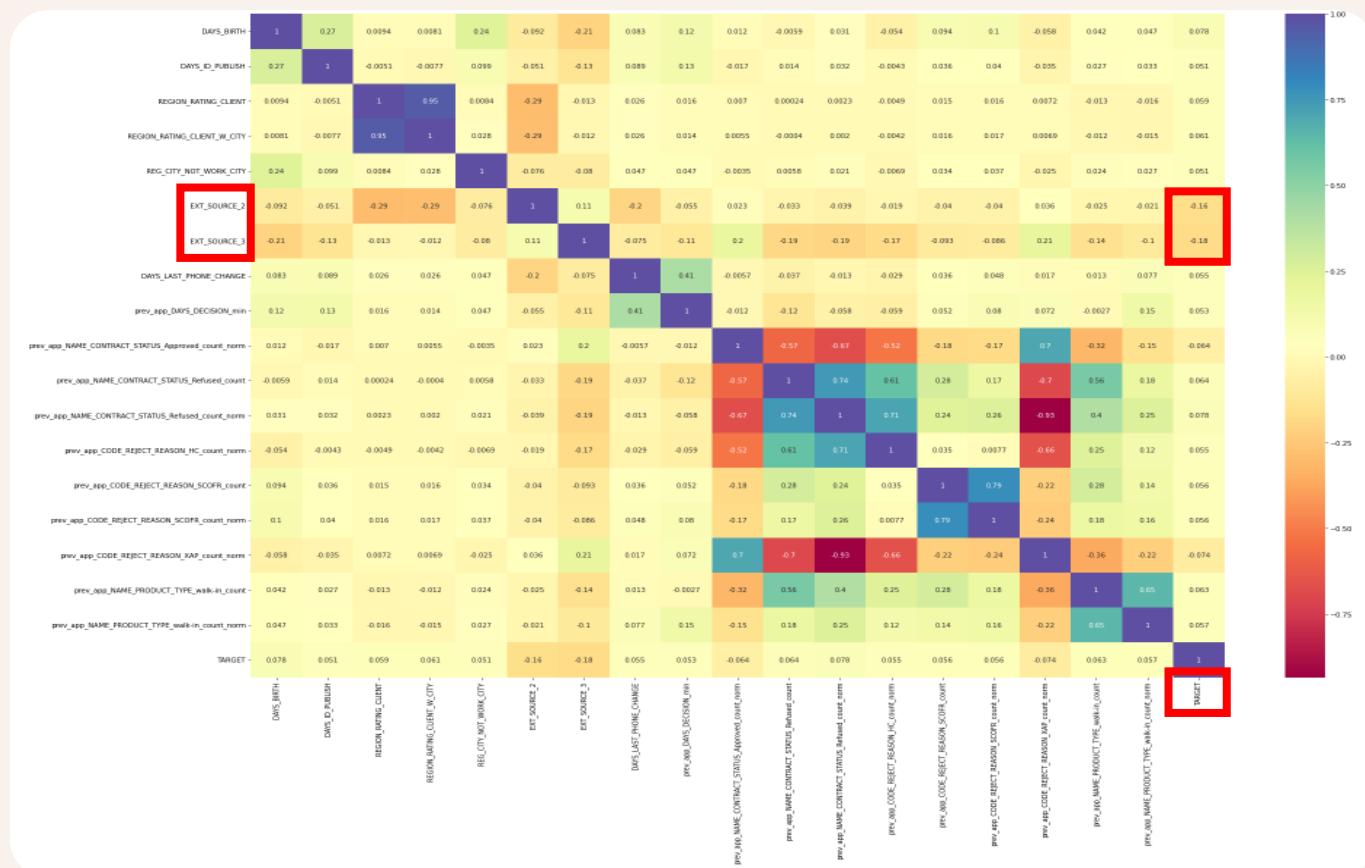
모델 평가

- Feature Importance



Modeling - 모델 평가

• 최종 상관관계 분석



캐글 점수

- 캐글

The screenshot shows a competition page for "Home Credit Default Risk" on Kaggle. The background features a close-up of a US dollar bill. At the top left, there's a trophy icon and the text "Featured Prediction Competition". On the right, it says "\$70,000 Prize Money". Below the title, the text "Can you predict how capable each applicant is of repaying a loan?" is displayed. Underneath the title, there's a logo for "Home Credit Group" and the text "7,176 teams · 3 years ago". The navigation bar includes links for Overview, Data, Code, Discussion, Leaderboard, Rules, Team (which is underlined), My Submissions, Late Submission (which is highlighted in a black box), and three dots. A section titled "Your most recent submission" shows a row of data: Name (prediction (1).csv), Submitted (a few seconds ago), Wait time (1 seconds), Execution time (1 seconds), and Score (0.75633). A green button labeled "Complete" is visible. At the bottom, there's a link "Jump to your position on the leaderboard ▾".

Name	Submitted	Wait time	Execution time	Score
prediction (1).csv	a few seconds ago	1 seconds	1 seconds	0.75633

Complete

[Jump to your position on the leaderboard ▾](#)

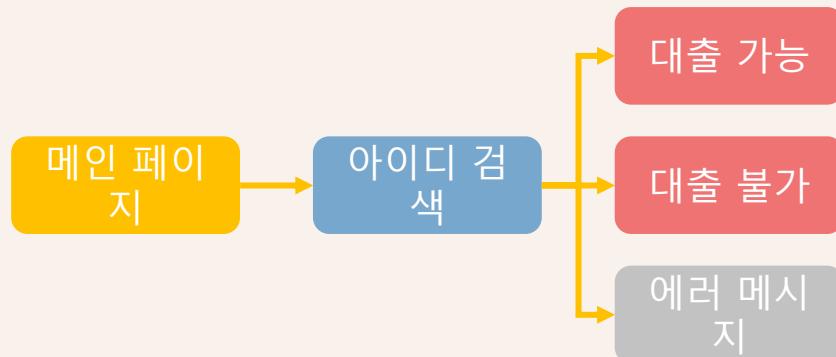
목차

CHAPTER 4. 웹 시연

1. Flask 기반 웹 구현

Flask 웹

- 페이지 흐름도



- search.html - 입력창

Check for Loan Availability

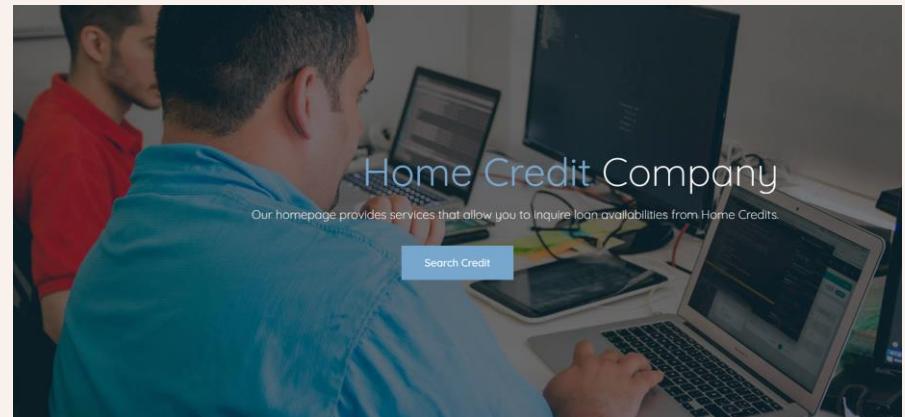
Please type your ID.

You can check if a loan will be possible.

Home Credit uses machine learning to analyze your information to find out your loan availability.

✓ index.html에서 Search Credit 버튼을 클릭하면 search.html 페이지로 이동

- index.html



- search.html – 잘못 입력한 경우

Check for Loan Availability

✗ Please Input Valid Value..!

Please type your ID.

You can check if a loan will be possible.

Home Credit uses machine learning to analyze your information to find out your loan availability.

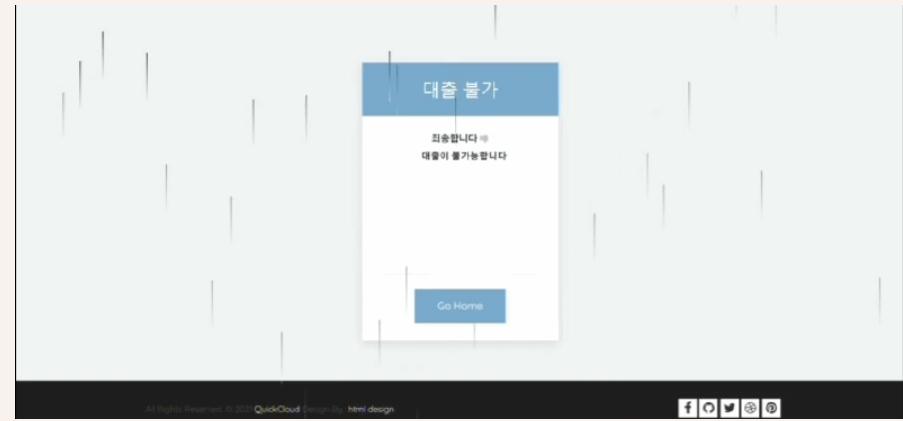
✓ 입력창에 잘못 입력한 경우는 flask의 flash로 올바른 값을 입력해달라는 문구가 생김

Flask 웹

- result0.html



- result1.html



✓ 입력한 아이디에 해당하는 사람이 대출 상환 능력이 있다고 판단된 경우, result0.html로 이동해, 대출 가능 문구를 보여줌.

✓ 입력한 아이디에 해당하는 사람이 대출 상환 능력이 없다고 판단된 경우, result1.html로 이동해, 대출 불가 문구를 보여줌.

Flask 웹

- app.py

```

1  from flask import Flask, render_template, url_for, request, redirect, flash
2  import find
3  app = Flask(__name__)
4
5  @app.route('/')
6  def index():
7      return render_template('index.html')
8
9  @app.route('/search')
10 def search():
11     return render_template('search.html')
12
13 @app.route('/result/', methods = ["POST"])
14 def result():
15     error = None
16     try:
17         customer_id = request.form['customer_id']
18         result = find.find_result(int(customer_id))
19
20         if result == 1 :
21             return render_template('result1.html')
22         elif result == 0 :
23             return render_template('result0.html')
24     except:
25         error = 'Please Input Valid Value..!'
26     return render_template('search.html', error=error)
27
28
29 if __name__ == '__main__':
30     app.run(debug=True)

```

- find.py

```

1  def find_result(customer_id) :
2      import pandas as pd
3      data = pd.read_csv("test.csv")
4      result = data[data['id'] == customer_id]['target'].values[0]
5      return result

```

- ✓ search.html의 form에서 입력한 아이디를 POST로 전달 받음.
- ✓ find.py는 해당 아이디의 결과 값 (대출 상환 능력이 있는지(0) 없는지(1) 결과)을 받아 옴.
- ✓ 대출 상환 능력이 없을 경우, result1.html로 이동.
- ✓ 대출 상환 능력이 있는 경우, result0.html로 이동.
- ✓ try~except를 이용해 잘못된 아이디를 입력하거나, 입력값이 없을 경우, flash를 이용해 error 메시지를 보여줌.

Flask 웹

- search.html – 검색 form과 error 메시지

```
<form action="{{ url_for('result') }}" method = "post" class="checkdomain form-inline">
  <div class="checkdomain-wrapper">
    <div style="color:red">
      {% if error %}
        <p><i class="fa fa-check"></i><strong> {{ error }}</strong></p>
      {% endif %}
    </div>
    <div class="form-group">
      <label class="sr-only" for="domainnamehere">Domain name</label>
      <input type="text" class="form-control" id="domainnamehere" name = 'customer_id' placeholder="Enter your customer id">
      <button type="submit" class="btn btn-primary grd1"><i class="fa fa-search"></i></button>
    </div>
  </div><!-- end checkdomain-wrapper -->
</form>
```

- ✓ form에 입력한 아이디를 POST로 전달.
- ✓ app.py에서 try ~ except에 지정한 것과 같이 값이 잘 전달 되었을 때는 error 메시지가 뜨지 않고, 잘못된 아이디를 입력하거나 입력 값이 없을 경우, flash를 이용해 error 메시지를 보여줌.

목차

CHAPTER 5. 마무리 및 소감

1. 팀원 소감