대출 상환 여부 예측

YBIGTA & P-SAT
연합세미나

팀 헛개수

황원영
곽지훈
이명진
백상현
나지윤

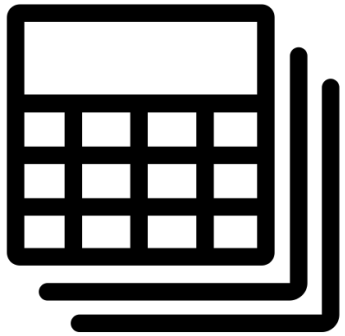
1주차 목차

- 데이터 셋 소개
- 데이터 전처리 및 EDA
  & Feature engineering
  - Application_Set
  - Bureau_Set
  - Previous_Application_Set
- 데이터 통합

데이터 셋 소개

주어진 data set은 각각의 loan 이 제때에 상환되었는지(target),
그리고 각각의 loan과 연관된 고객의 정보가 담겨있다

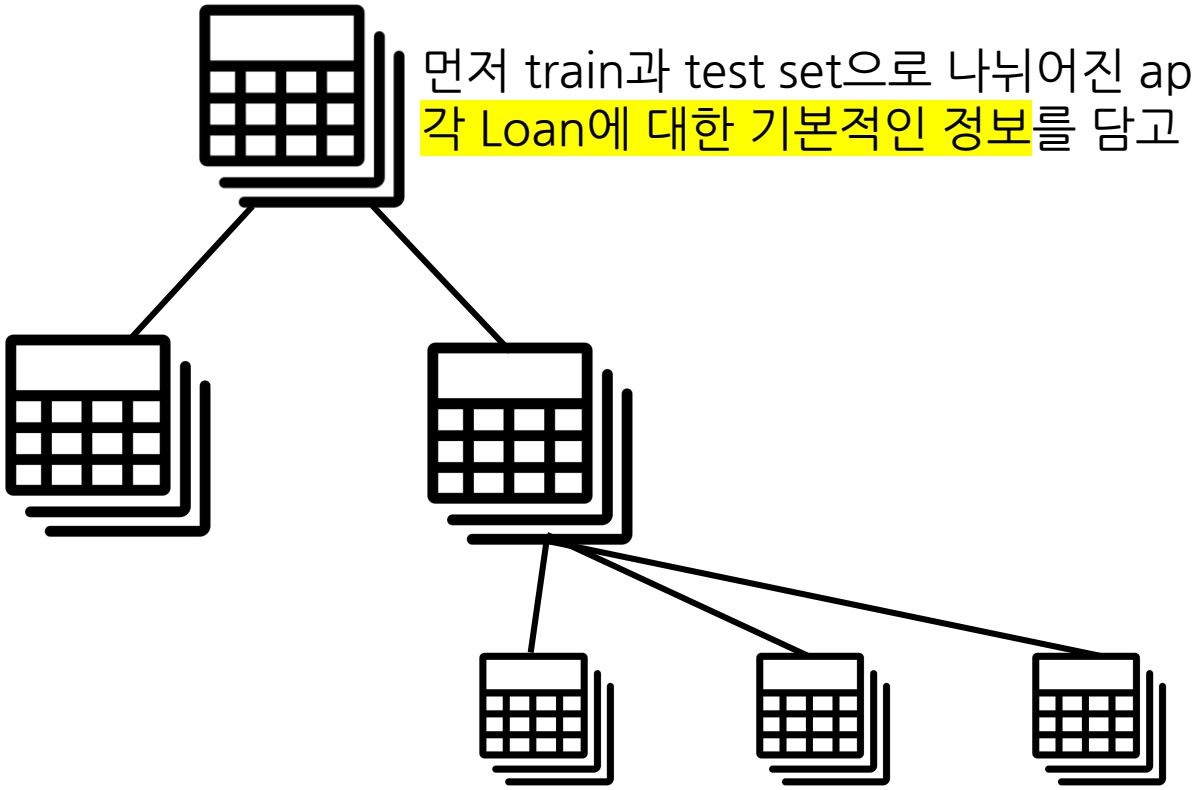
데이터 셋 소개



× 6

크게 나누면 6가지의 set으로 구성되어 있었는데,

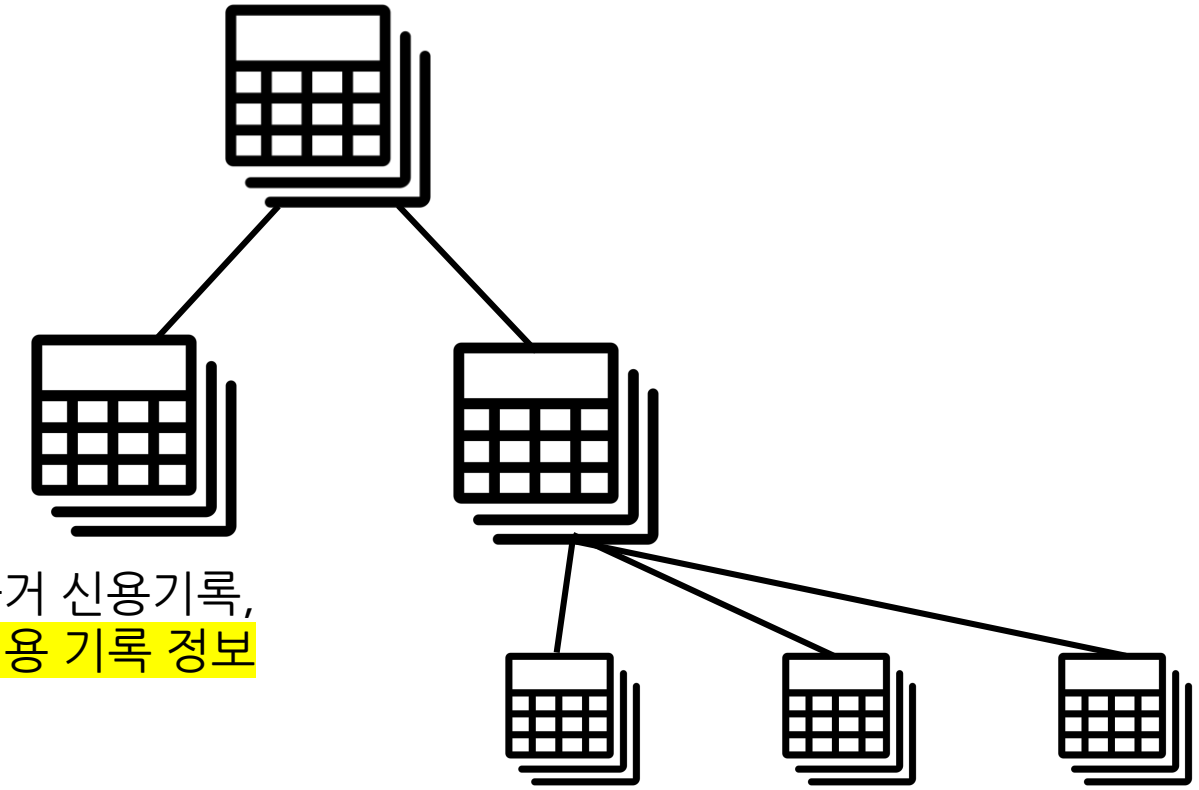
데이터 셋 소개



먼저 train과 test set으로 나뉘어진 application set,
각 Loan에 대한 기본적인 정보를 담고 있다

데이터 소개 | Application set | Bureau set | Previous application set | 데이터 통합

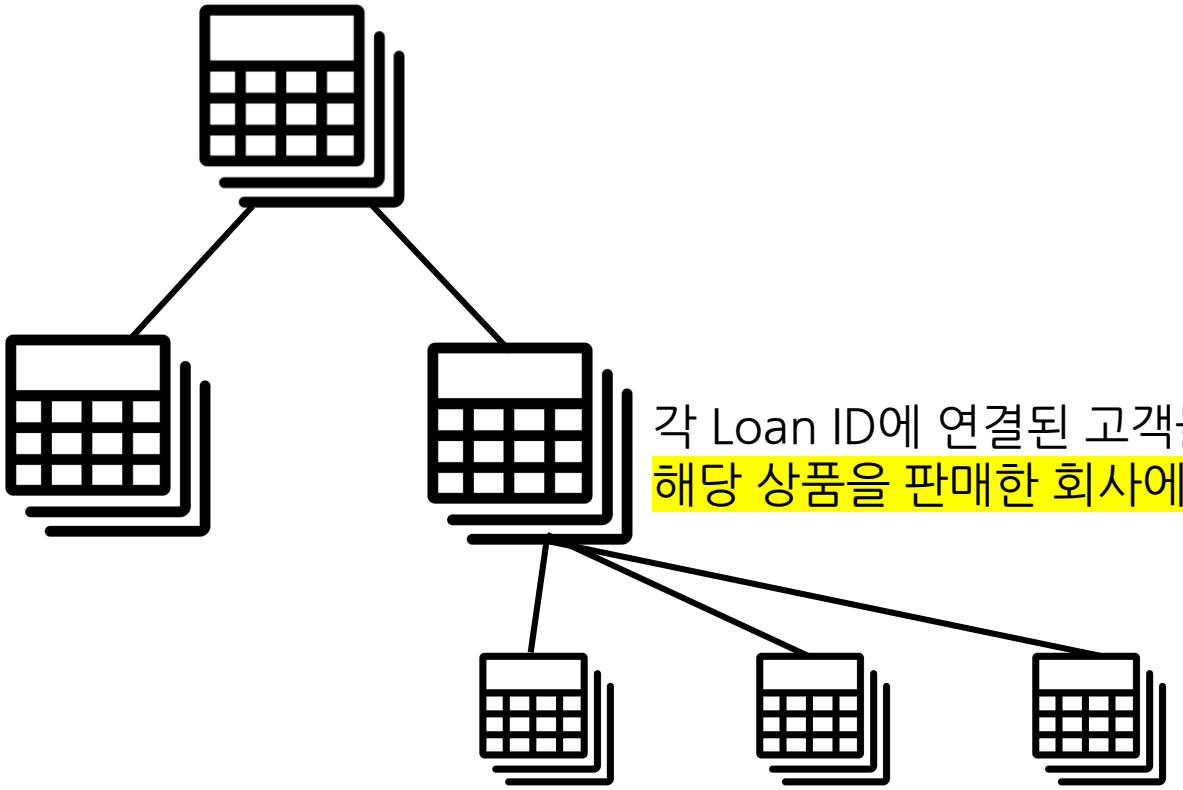
데이터 셋 소개



각 Loan ID에 연결된 고객들의 과거 신용기록,
Credit Bureau에 보고된 신용 기록 정보

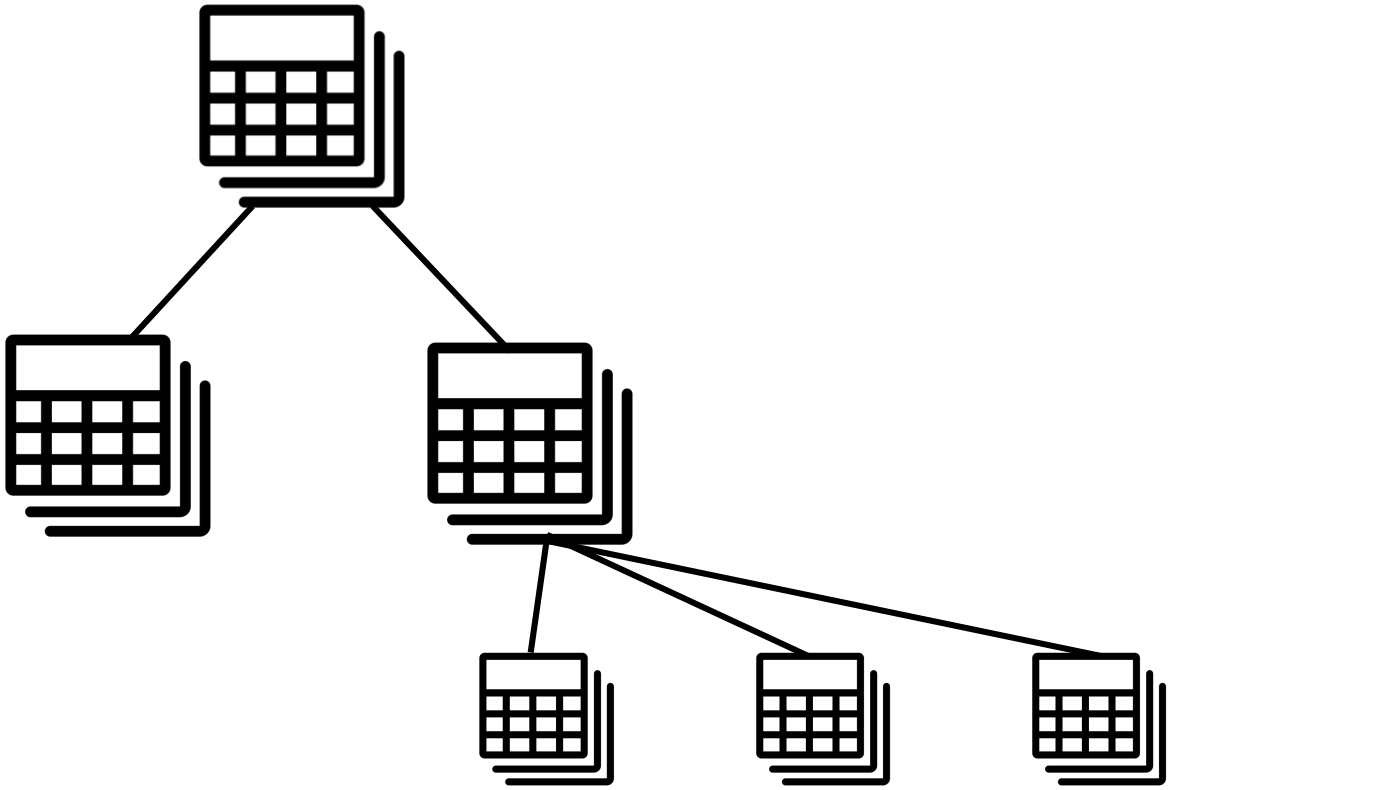
데이터 소개 | Application set | Bureau set | Previous application set | 데이터 통합

데이터 셋 소개

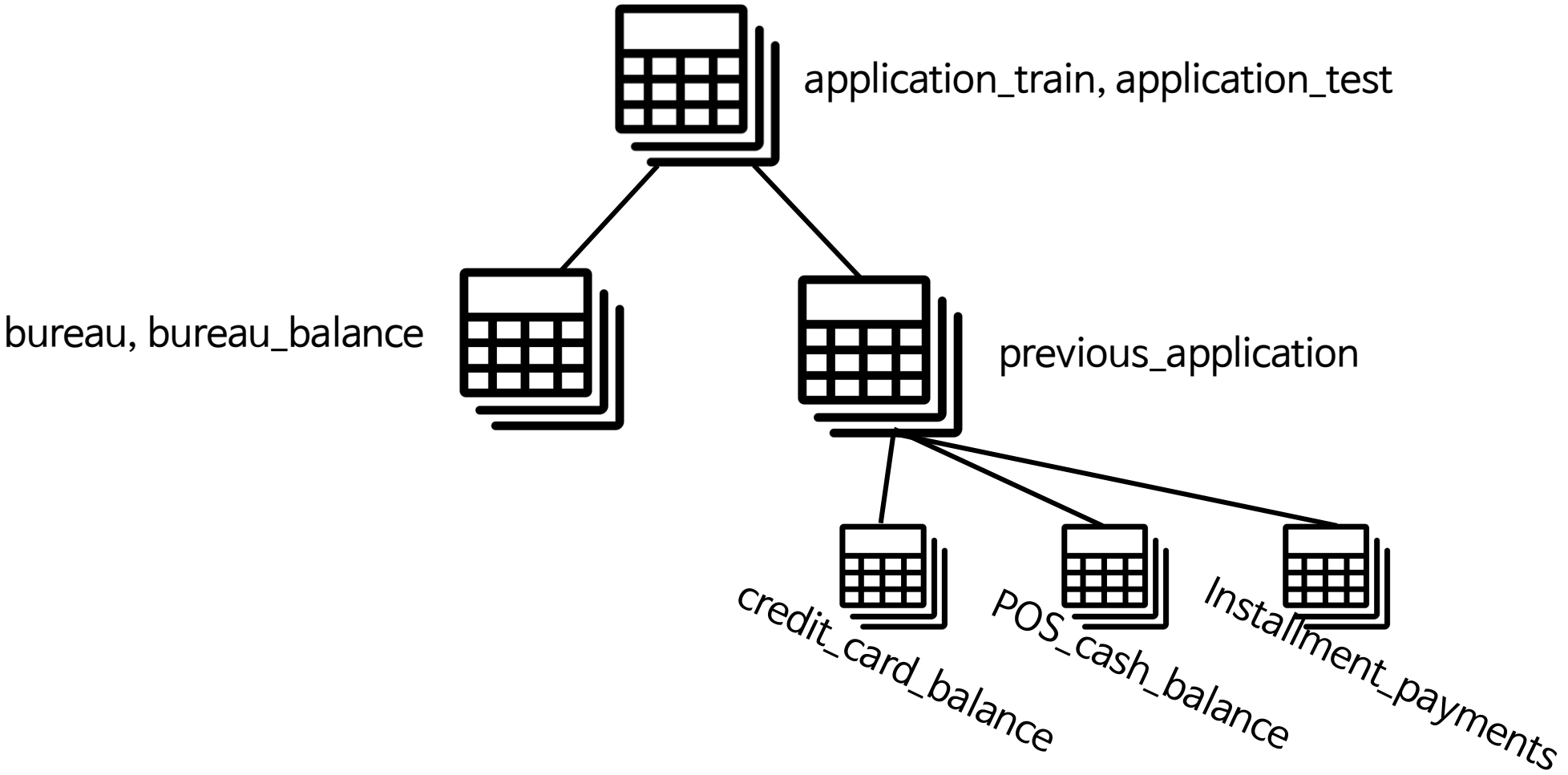


각 Loan ID에 연결된 고객들의 과거 신용기록,
해당 상품을 판매한 회사에 기록된 신용 기록 정보

데이터 셋 소개



데이터 셋 소개

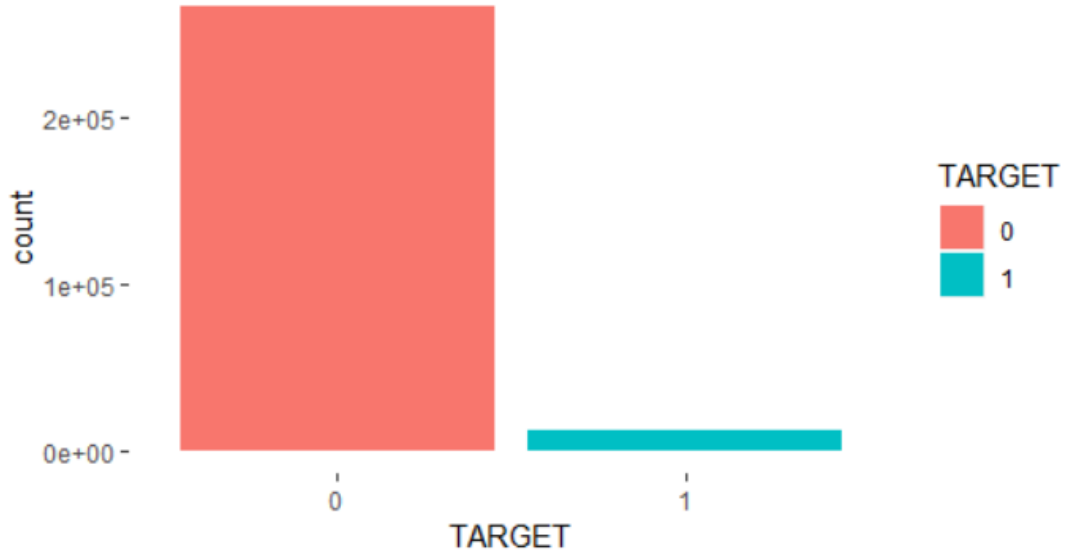

application_train, application_test
bureau, bureau_balance
previous_application
credit_card_balance
POS_cash_balance
Installment_payments

데이터 전처리 및 EDA

데이터 전처리 방향

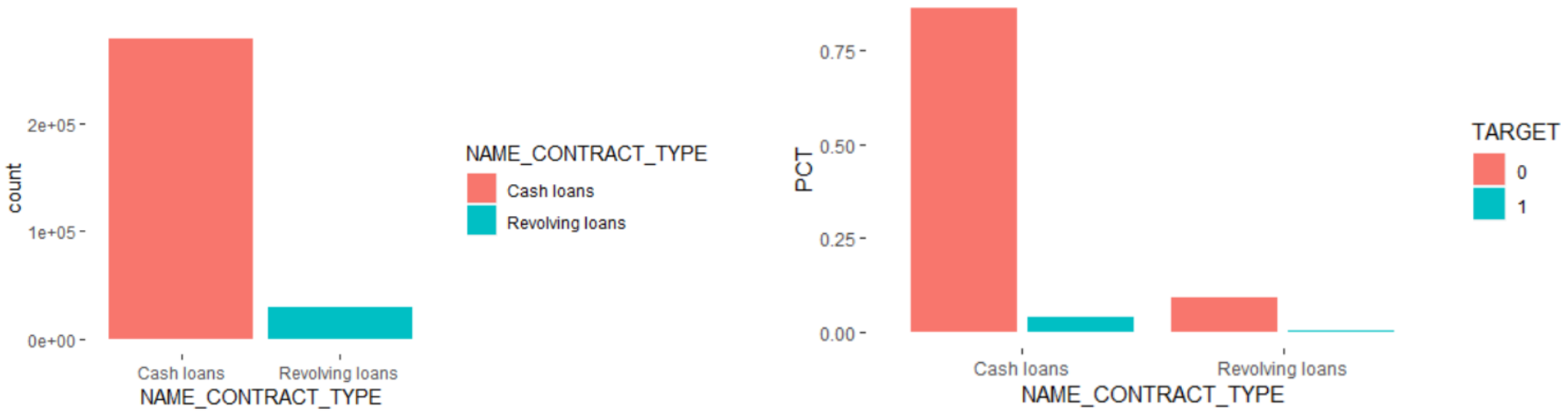
- ✓ 합리적인 결측치 처리
- ✓ 주어진 데이터 활용 최대화
- ✓ Redundant 한 변수 삭제

데이터 전처리 및 EDA
Application_Set



TARGET 이 굉장히 unbalanced 임을 확인 할 수 있다
(1 : 제때에 상환 못함, 0 : 제때에 상환 함)

데이터 전처리 및 EDA
Application_Set



Contract type은 Cash loan이 압도적으로 높았다
(Revolving loan의 예시로는 credit card가 있다, 고정된 지불액을 가지고 있지 않은 credit의 종류)

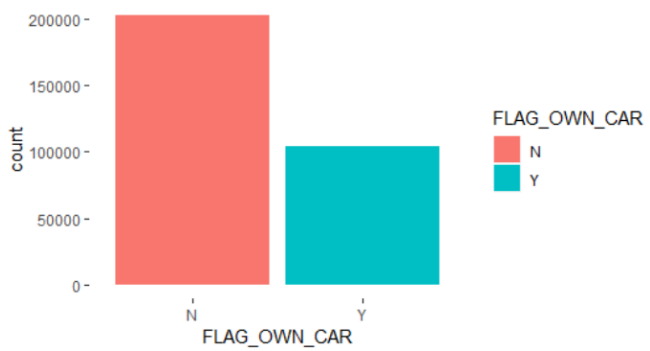
데이터 전처리 및 EDA

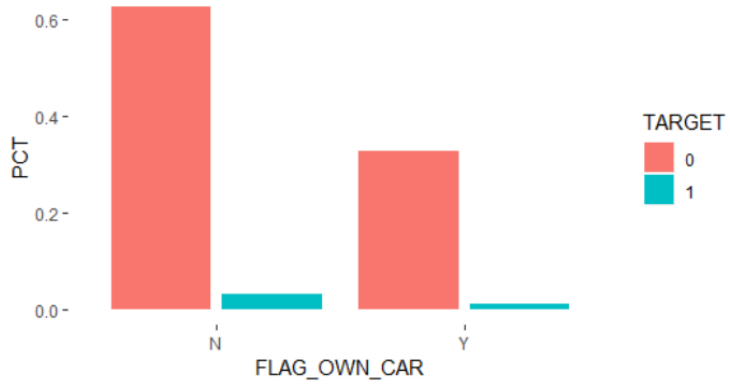
Application_Set

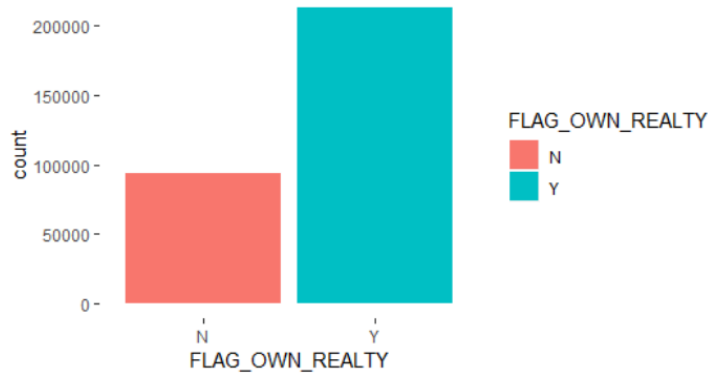
성별의 비율은 여성이 2배 정도로 높았으며,
성별이 기입되지 않은 obs가 2개 발견되었다

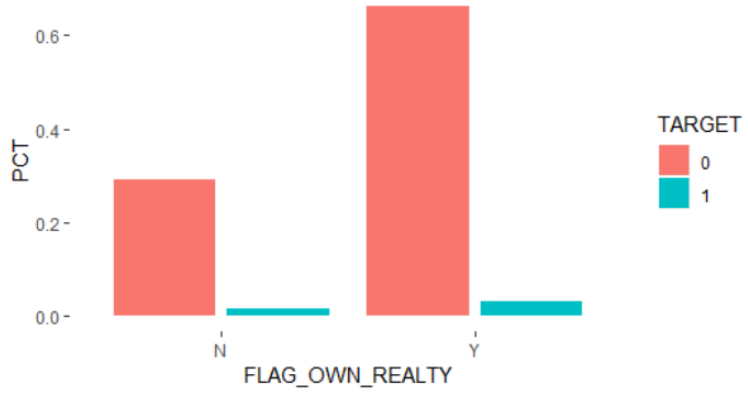
데이터 전처리 및 EDA

Application_Set



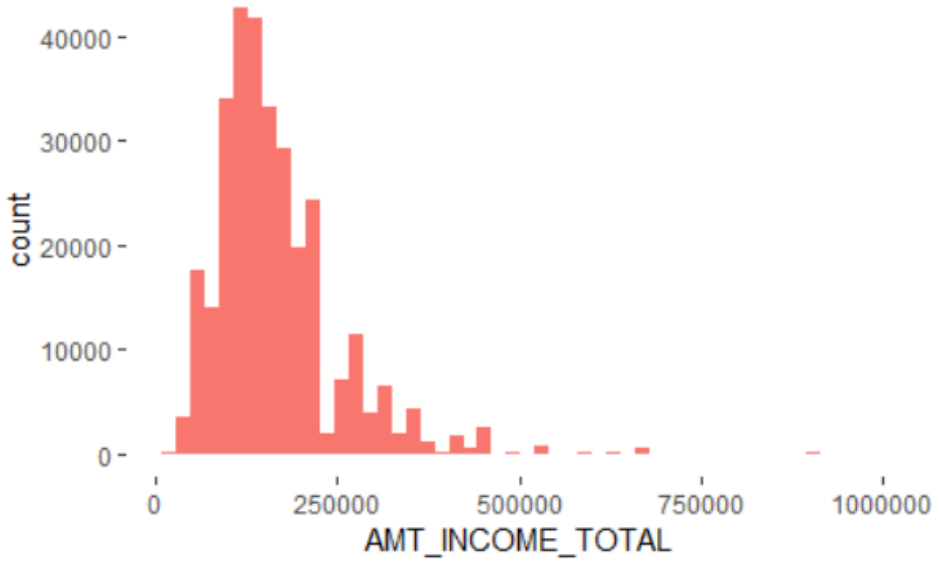






해당 고객이 차와 부동산을 소유했는지도 확인 할 수 있었다

데이터 전처리 및 EDA
Application_Set

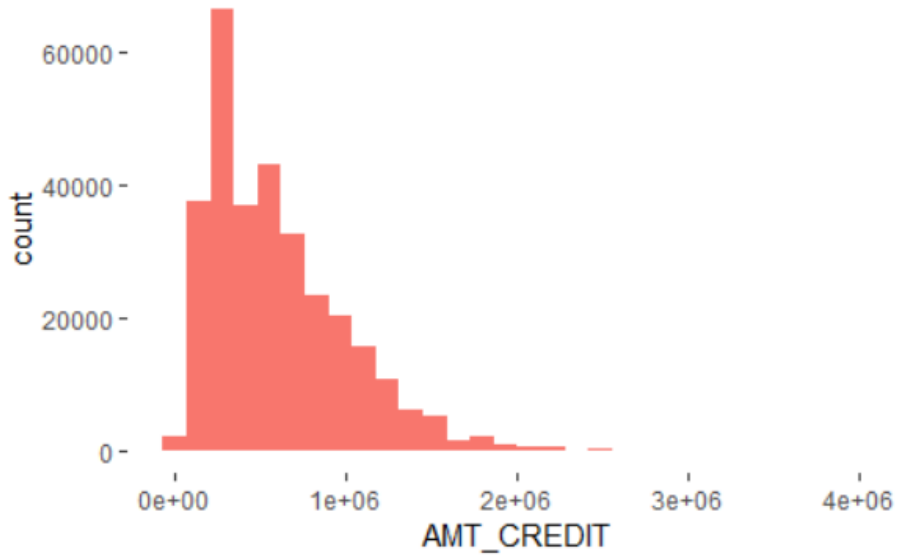


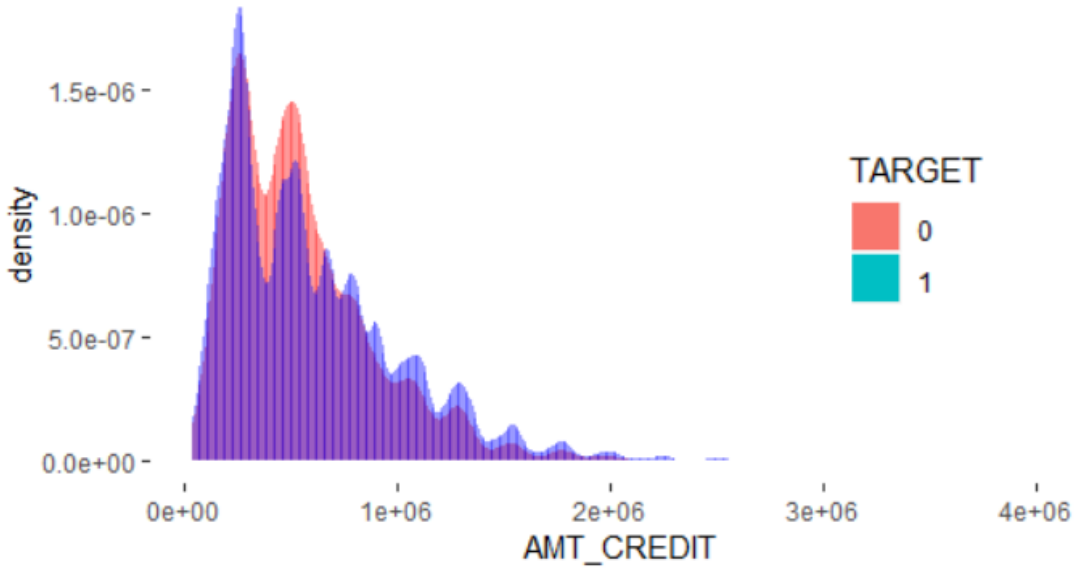
고객의 소득을 나타낸 plot인데 실제로는 더 right-skewed 하다,
(소득이 1000000$ 가 넘는 250개의 obs를 제외하고 plotting 한 결과)

소득 구간을 10분위로 나눈 뒤
각 구간에서 상환율을 비교해보았다

데이터 전처리 및 EDA

Application_Set

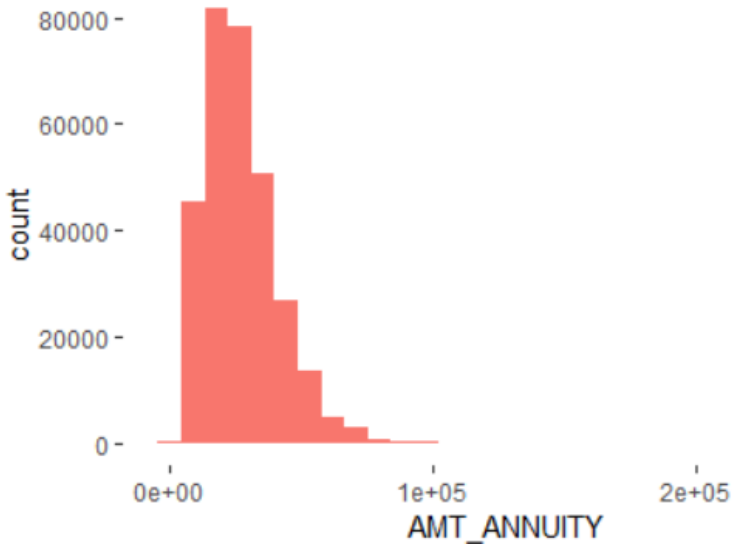


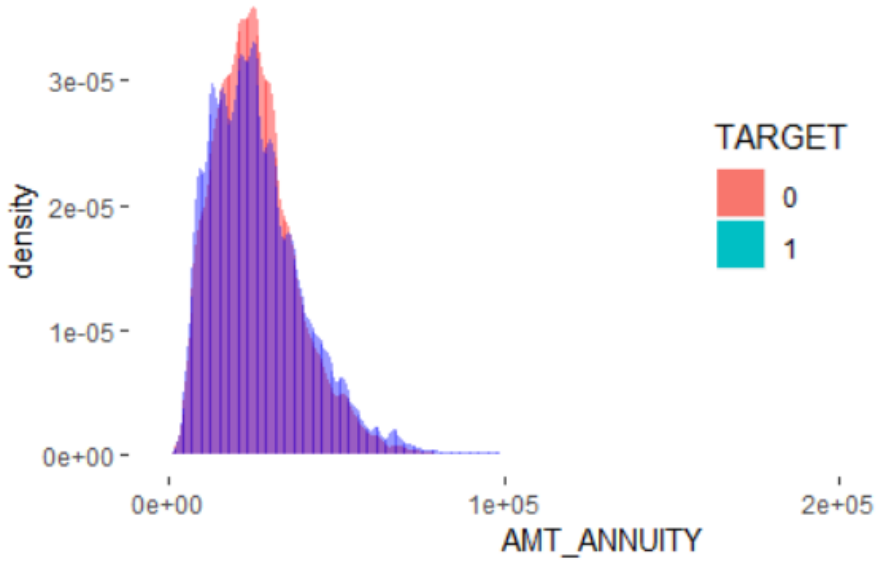


Credit amount도 right skewed함을 확인 가능하다,
오른쪽 plot은 target별 credit amount의 분포

데이터 전처리 및 EDA

Application_Set

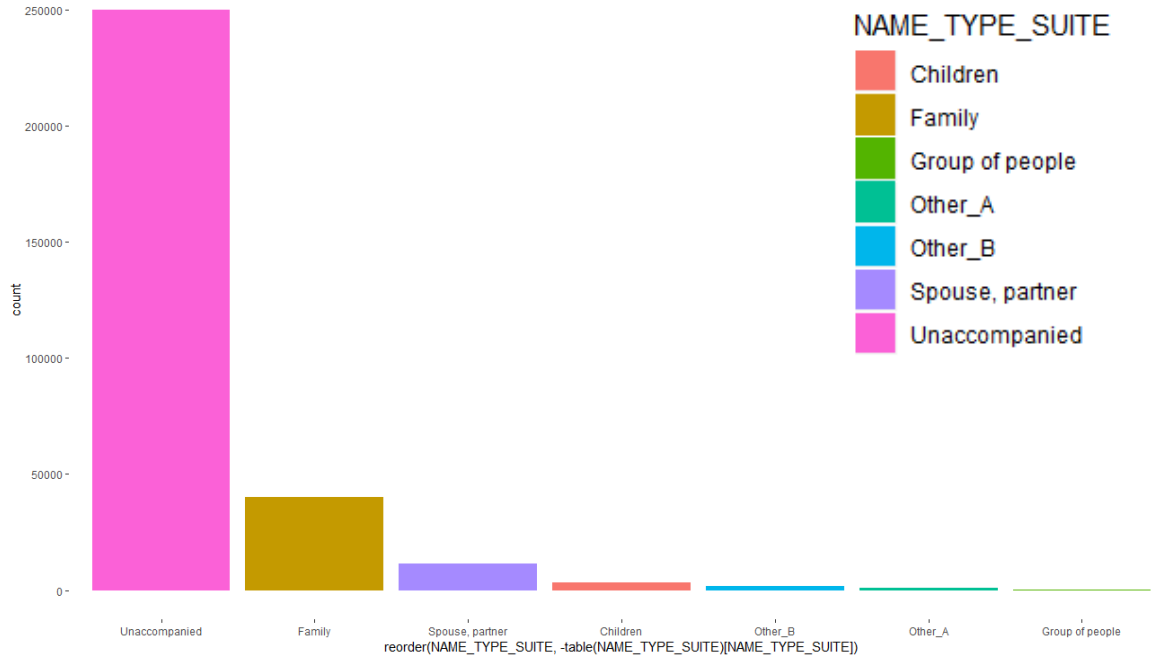




AMT_ANNUITY에서는 12개의 NA값이 발견 되었는데,
Loan annuity(정기적으로 갚아야 하는 돈)과 Amount credit의 correlation이 0.77임을 발견 regression Imputation 진행

데이터 전처리 및 EDA

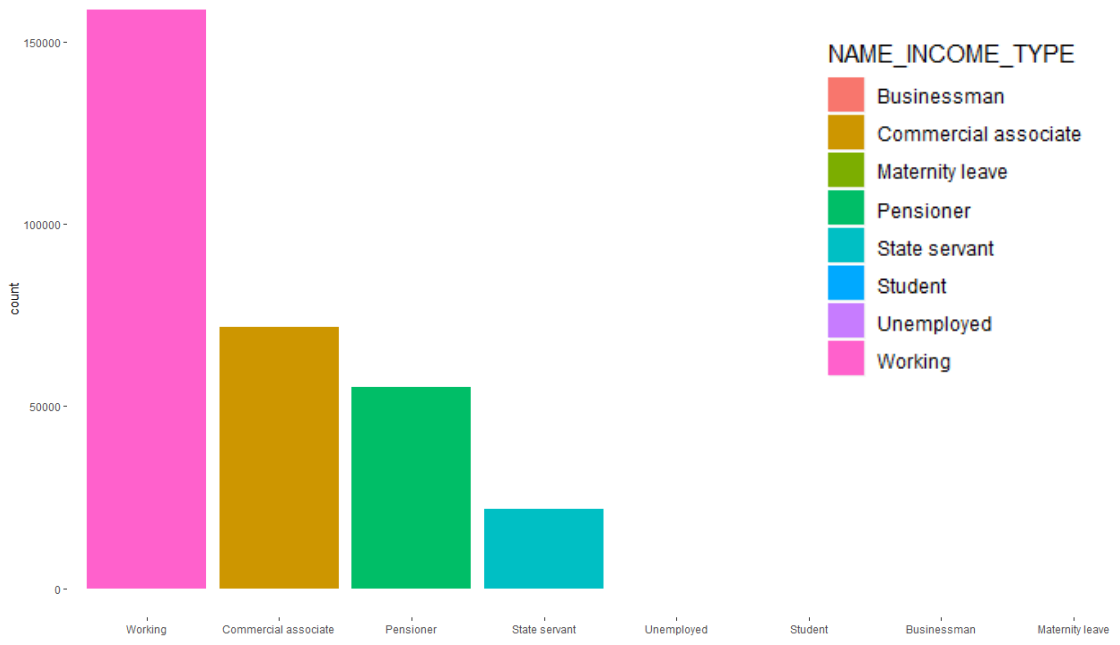
Application_Set

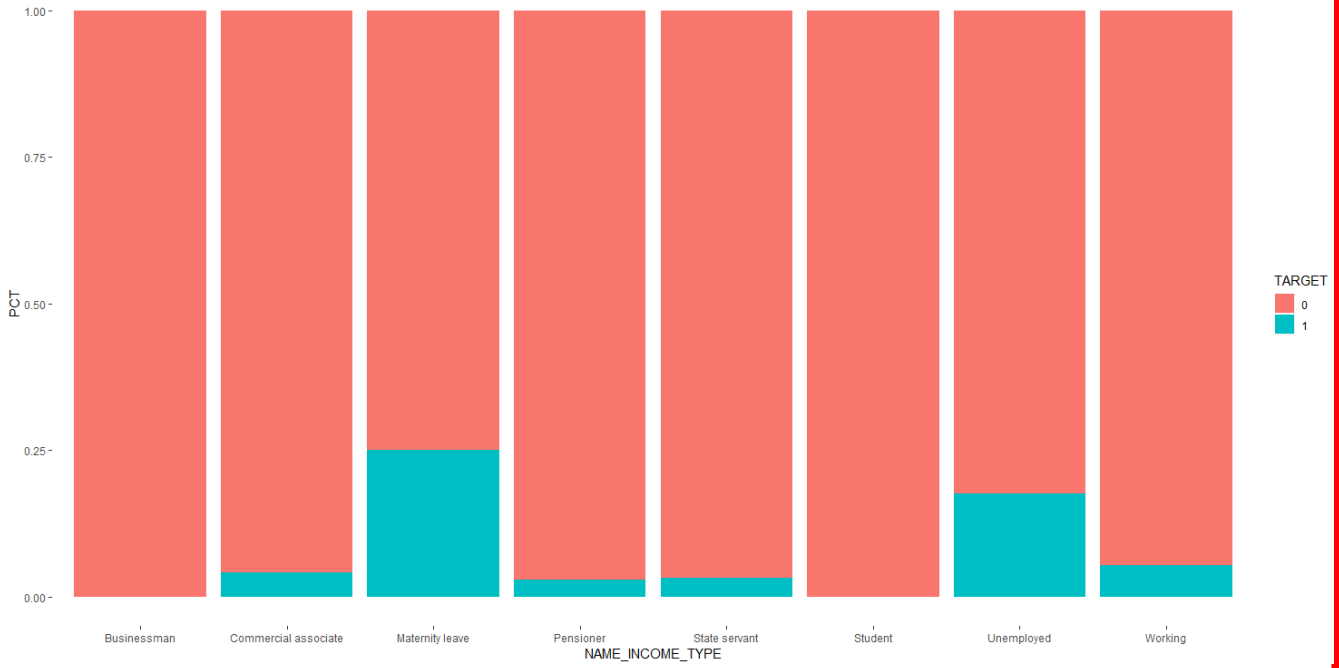


Loan 신청 당시 동반한 사람의 유형에서 발견된 1292개의 결측치는
동반자가 없었다는 것으로 판단 Unaccompanied로 교체했다

데이터 전처리 및 EDA

Application_Set

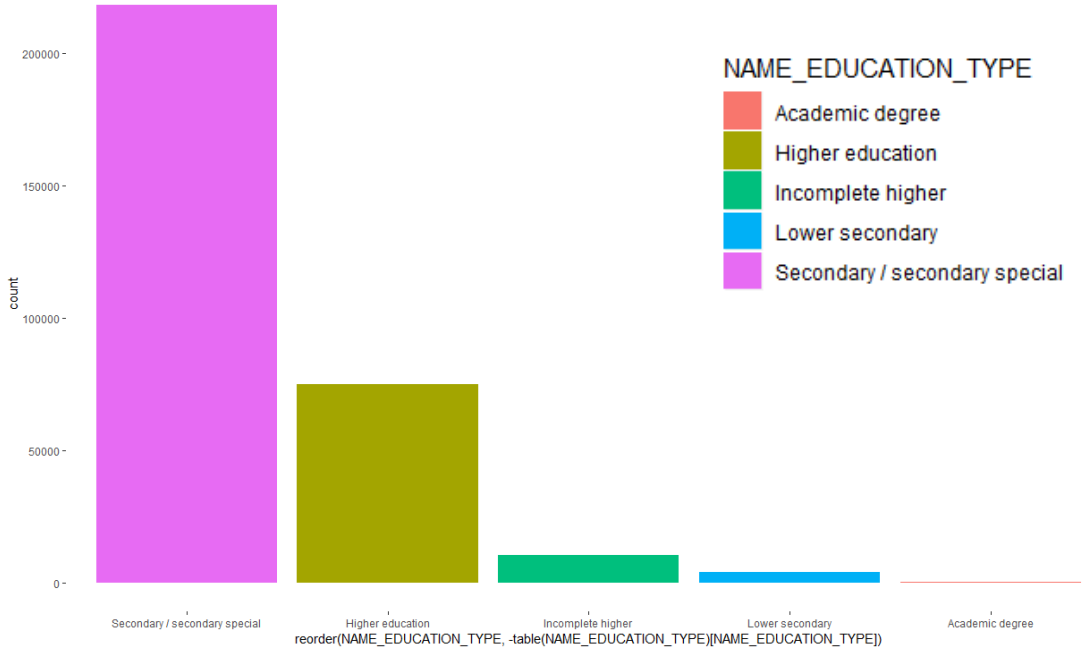


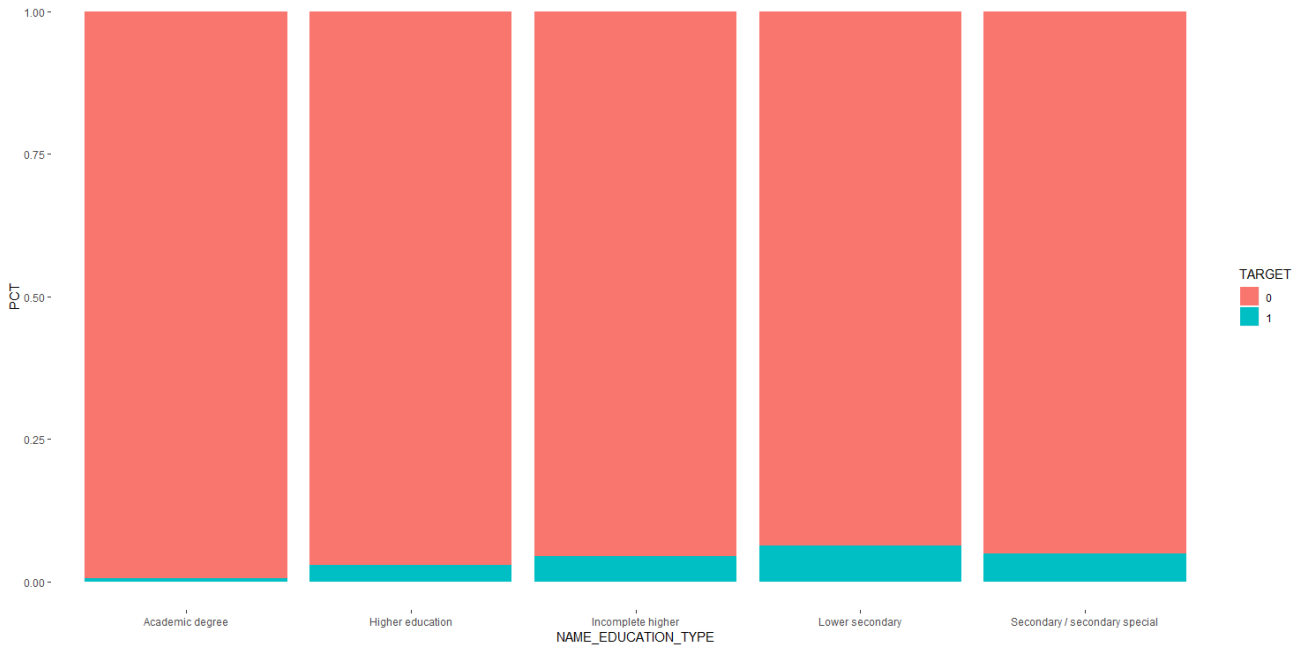


Income type에 대한 정보도 포함되어 있으며 각 Income type별 상환율을 비교한 결과
Maternity leave(출산휴가 수당)과 Unempolyed 의 경우 상환 실패 비율이 다소 높았다

데이터 전처리 및 EDA

Application_Set

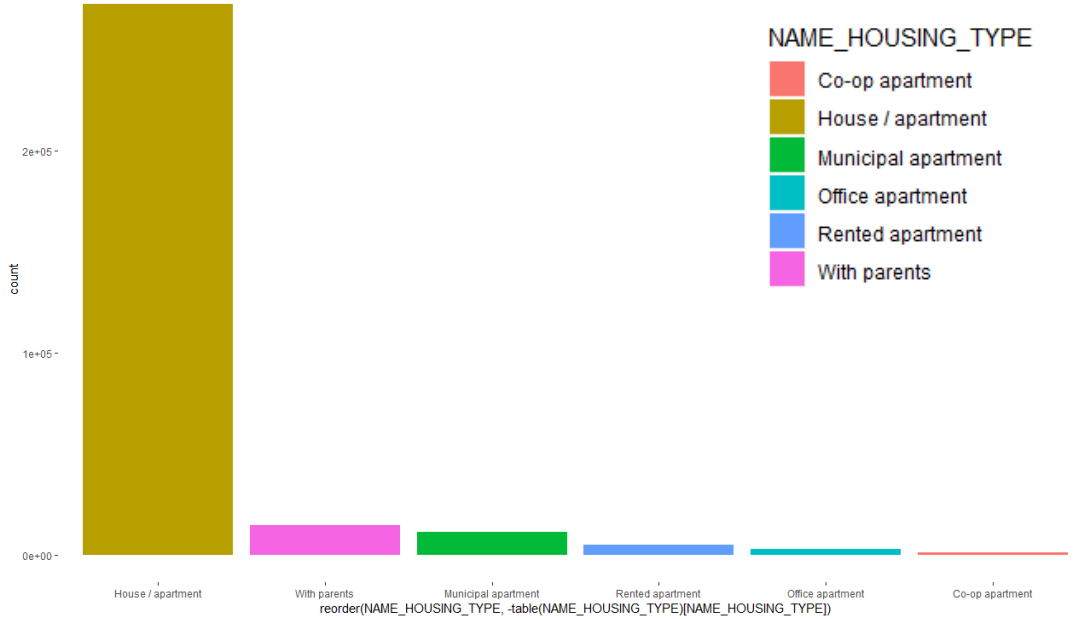


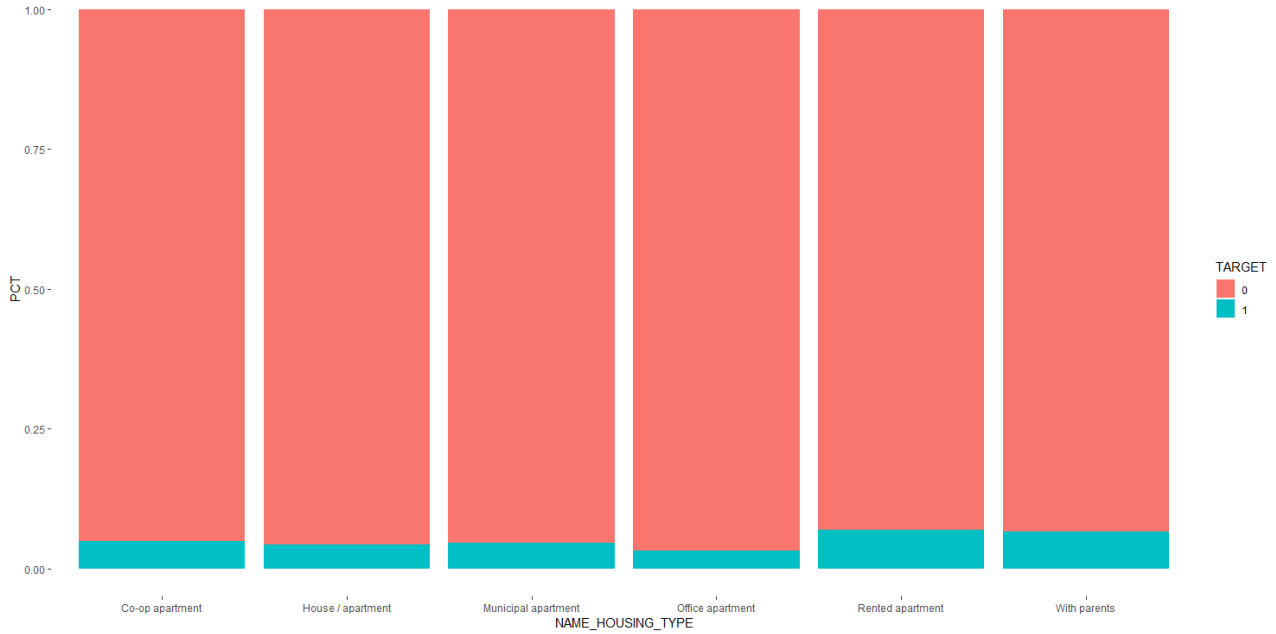


Education type에 대한 정보도 포함되어 있으며 각 Education type별 상환율을 비교한 결과
Lower secondary의 경우 상환 실패 비율이 다소 높았다

데이터 전처리 및 EDA

Application_Set

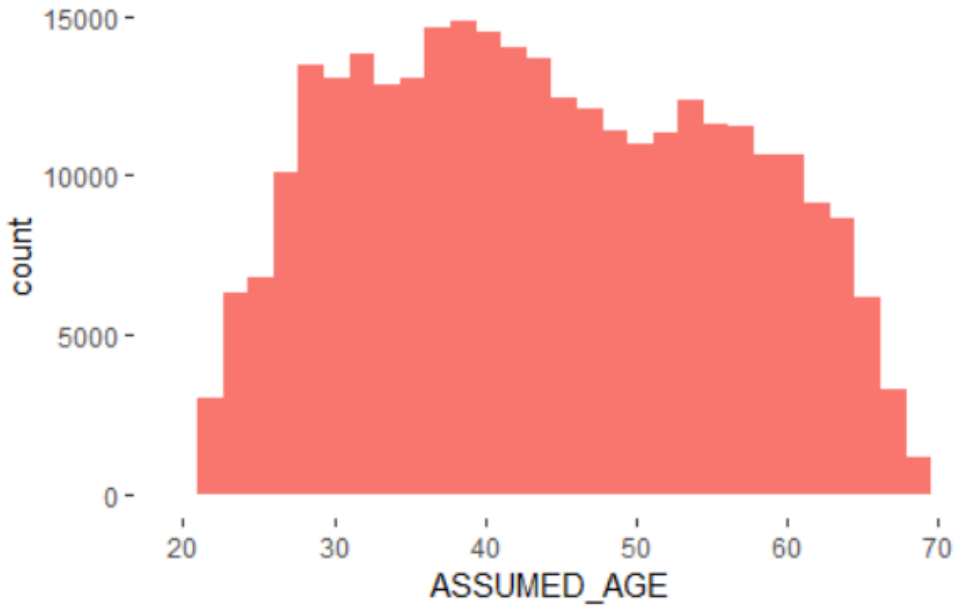


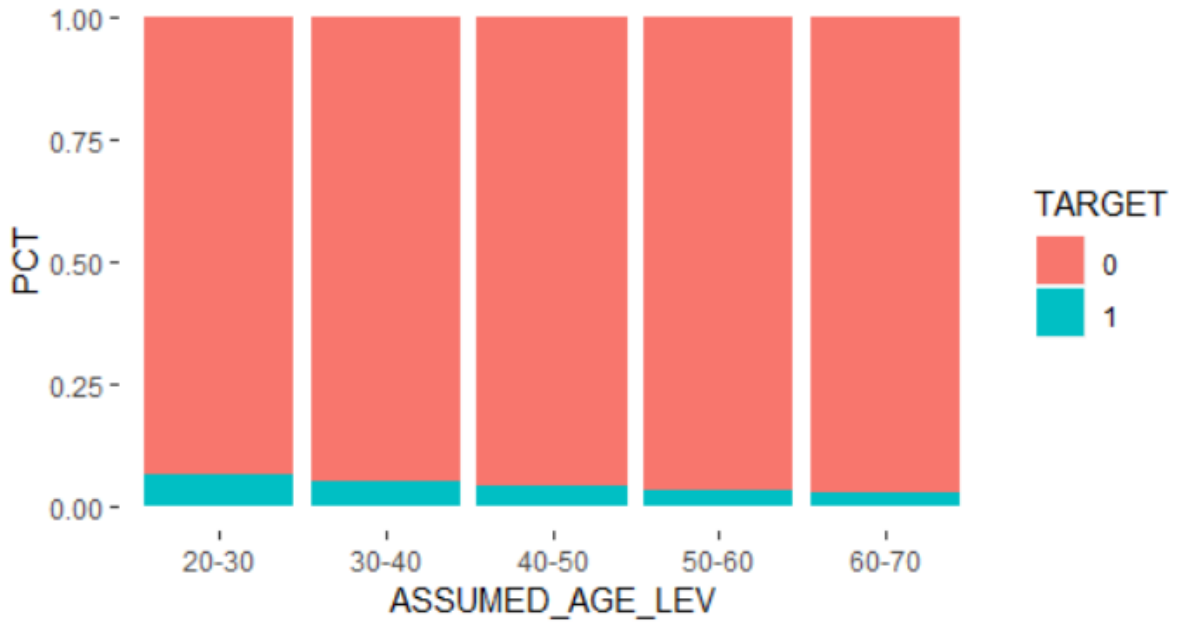


Housing type에 대한 정보도 포함되어 있으며 각 Housing type별 상환율을 비교한 결과
rented apartment의 경우 상환 실패 비율이 다소 높았다

데이터 전처리 및 EDA

Application_Set

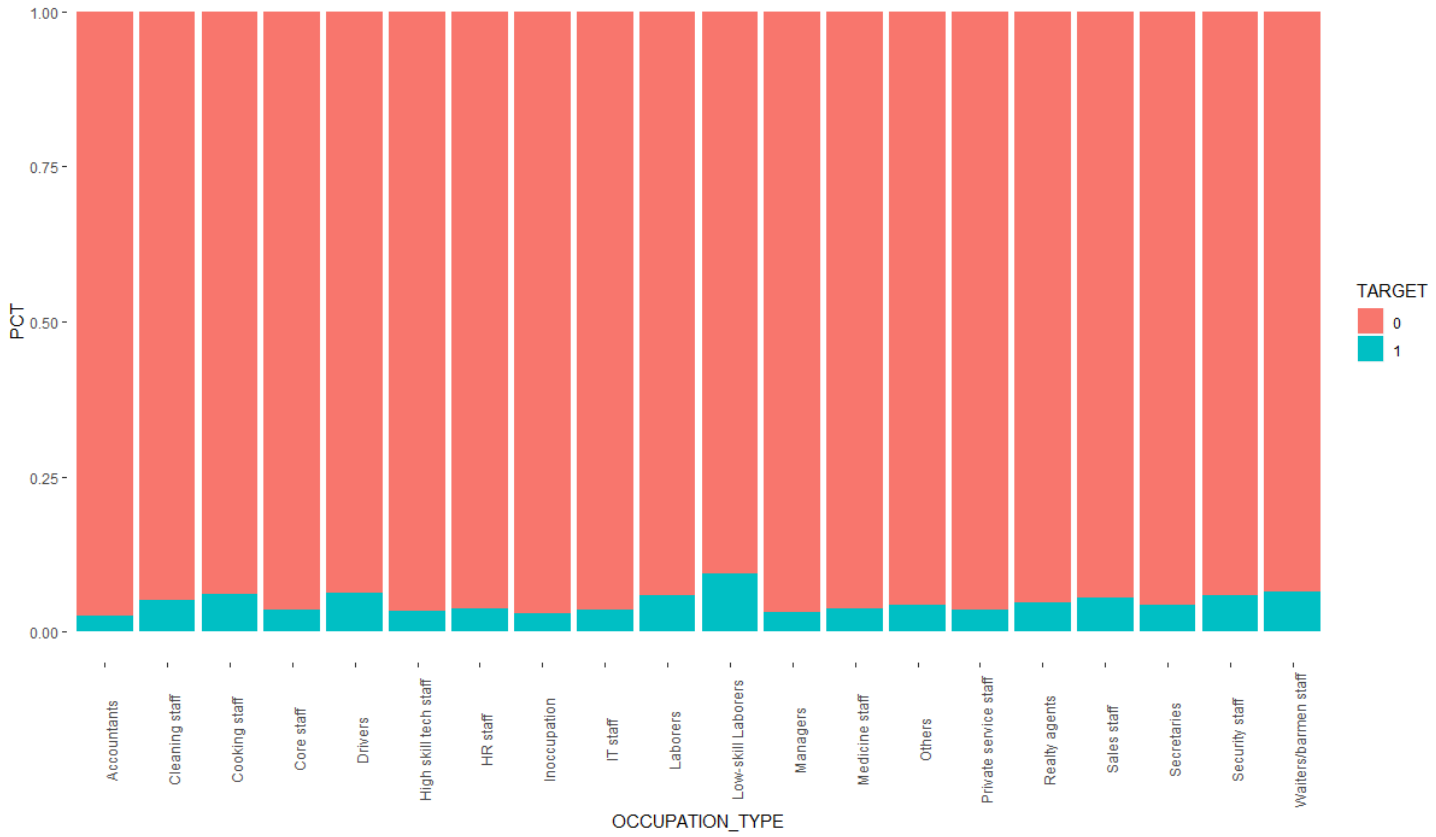




DAYS_BIRTH 변수는 해당 loan 상품 신청 당시 날과 고객이 태어난 날의 차이로 표현되어 있어
이를 사용해 고객의 나이(ASSUMED_AGE) 변수를 생성함,
오른쪽은 나이대별 상환율을 시각화한 plot, 20대의 상환 실패 비율이 다소 높다

데이터 전처리 및 EDA

Application_Set



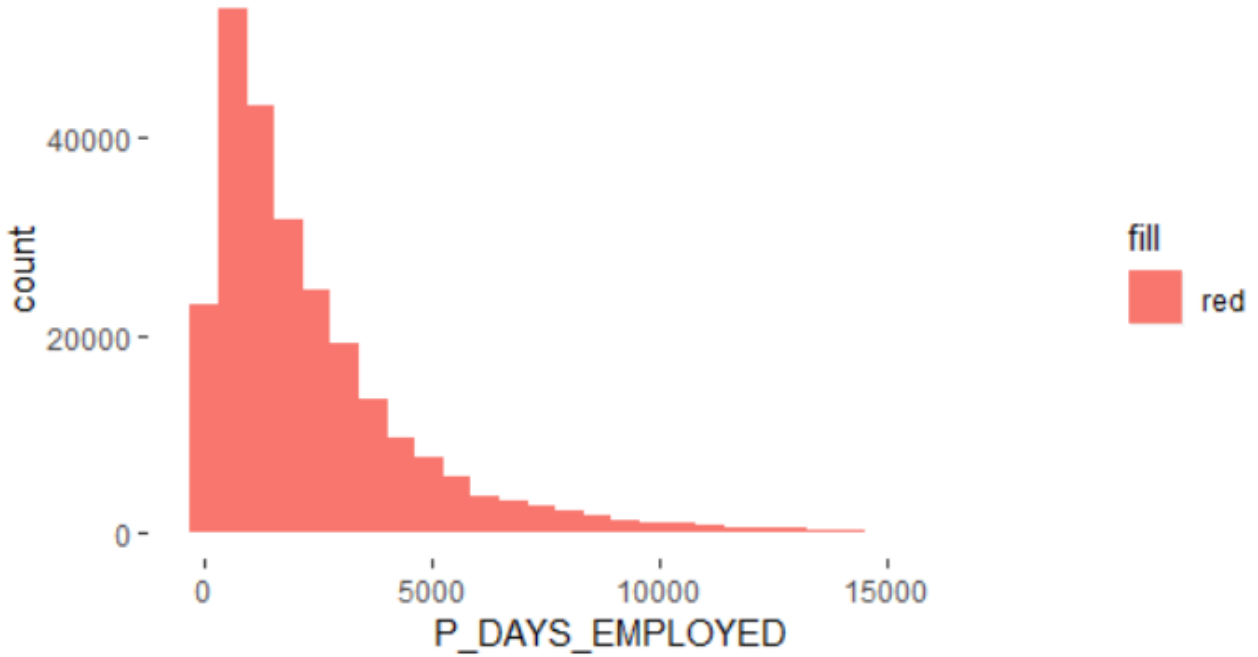
Occupation type 에서 NA로 기록 된 obs 중 Days employed도 NA인 obs는 무직이라고 판단 Inoccupation으로 교체

그렇지 않은 경우는 주어진 category에 해당하지 않는 직업이라고 판단 Others로 교체

왼쪽 plot은 직업별 상환율 비교, Low skilled Laborers 의 상환 실패 비율이 다소 높음

데이터 전처리 및 EDA

Application_Set



DAYS_EMPLOYED는 해당 loan 상품 신청 당시 날과 직업을 구한 날과의 차이(-)로 표현되어 있었으며
+365243라는 이상치를 지닌 55374개의 obs 발견,
해당 변수는 Occupation type에서 무직임이 발견되었고 Income type에서도 모두 Pensioner(연금 수급자)
로 나타났기에 직업이 없는 obs라 판단함

데이터 전처리 및 EDA

Application_Set

고객의 거주지 관련 정보(거실 크기, 건물 연식, 창고 크기 등등…)은 결측치의 비율이 매우 높아 사용할 수 없다고 판단함

대신 거주지 관련 정보를 하나라도 입력하면 1, 아니면 0이 들어가는 BUILDING_FILL변수 생성

데이터 전처리 및 EDA

Application_Set

AMT_REQ_CREDIT_BUREAU_HOUR

AMT_REQ_CREDIT_BUREAU_DAY

AMT_REQ_CREDIT_BUREAU_WEEK

AMT_REQ_CREDIT_BUREAU_MON

AMT_REQ_CREDIT_BUREAU_QRT

AMT_REQ_CREDIT_BUREAU_YEAR

Loan 신청 전 credit bureau 에 문의한 횟수,
모든 변수에 걸쳐 NA값을 갖는 obs가 발견되었는데
문의사항이 없었다고 판단 0으로 교체

OBS_30_CNT_SOCIAL_CIRCLE

DEF_30_CNT_SOCIAL_CIRCLE

OBS_60_CNT_SOCIAL_CIRCLE

DEF_60_CNT_SOCIAL_CIRCLE

Number of observations of client's social surroundings with observable DPD (Days Past Due)
모든 변수에 걸쳐 NA값을 갖는 obs가 발견되었는데
DPD observation이 없었다고 판단 0으로 교체

데이터 전처리 및 EDA
Application_Set

그 외...

DAYS_LAST_PHONE_CHANGE : 결측치 1개 존재, mean Imputation

AMT_GOODS_PRICE : AMT_CREDIT과의 correlation 이 0.98로 높아 열 삭제

OWN_CAR_AGE : missing value의 비율이 매우 높아 열 삭제

DOCUMENT_NUM : 총 제출한 문서의 수 변수 생성(FLAG_DOCUMENT 변수 활용)

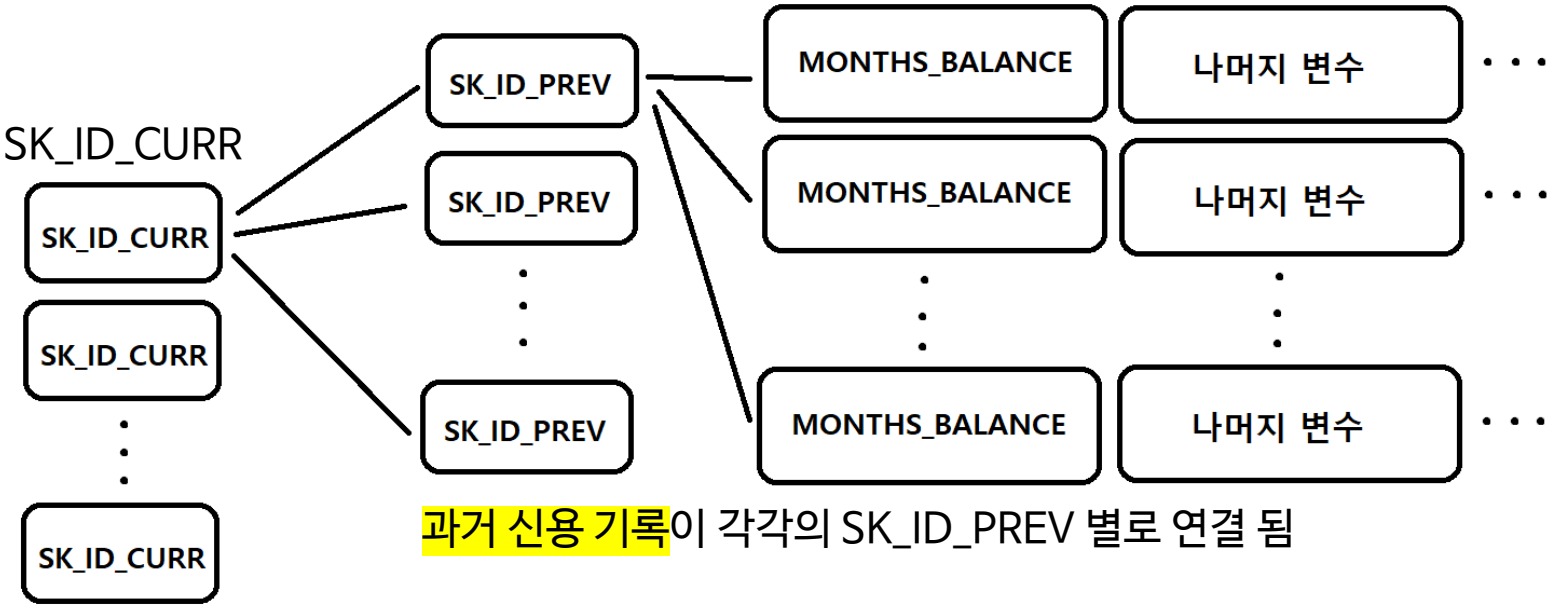
Feature Engineering

Application set 외의 Data set은 과거의 신용기록을 담은 Data set 이었기에,
각각의 Loan(SK_ID_CURR)별로 여러가지 정보를 담고 있었다

Feature Engineering

각 SK_ID_PREV 별로 여러가지 정보가 담겨 있음

Data 통합 시에 key가 되는 SK_ID_CURR

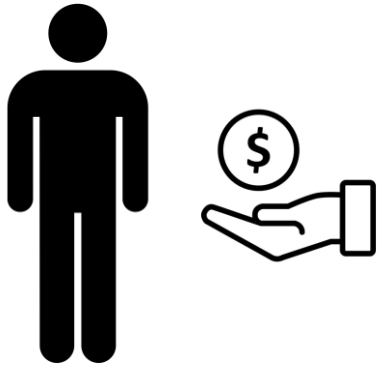


과거 신용 기록이 각각의 SK_ID_PREV 별로 연결 됨

Feature Engineering

SK_ID_CURR 당 row가 두개 이상 생기는 것을 방지하기 위해,
과거 신용기록의 통계량 또는 가장 의미 있어 보이는 값을 활용해 새로운 feature 생성

Feature Engineering
Bureau_Set



Bureau data set은 SK_ID_CURR별로 Credit Bureau에 보고된
이전 신용상품에 대한 정보가 담겨있다

Feature Engineering
Bureau_Set

주요 변수 설명

"ACT_NUM" : Credit bureau 에 보고된 대출 상품의 수 중 Status 가 Active 한 상품의 수

"ACT_PCT" : Credit bureau 에 보고된 대출 상품의 수 중 Status 가 Active 한 상품의 비율

"MEAN_DAYS_CREDIT_ACT" : application set에 있는 대출 신청한 시점과 CB 에 보고된 대출 상품 신청 시점 기간의 평균

Loan 신청 당시 이미 다른 신용 상품을 사용하고 있다면 그 신용상품을 얼마큼 전에 신청했는지 알 수 있다.

(Active하지 않은 상품은 의미 없다고 판단해서 Active한 신용 상품만 고려함)

"MEAN_AMT_CREDIT_SUM" : application set 에 대출 신청한 시점에 CB에 보고된 대출 상품 금액의 평균

Loan 신청 당시 이미 다른 신용 상품을 사용하고 있다면 그 신용상품의 크기가 얼마인지 알 수 있다.

(Active하지 않은 상품은 의미 없다고 판단해서 Active한 신용 상품만 고려함)

Feature Engineering
Bureau_Set

주요 변수 설명

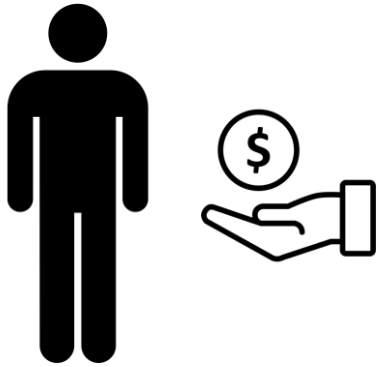
"SUM_CNT_CREDIT_PROLONG" : Credit bureau에 보고된 대출 상품 중 기한 연장 된 상품 수의 총합

"MEAN_AMT_CREDIT_SUM_OVERDUE" : Credit bureau 에 보고된 대출 상품 중 기한을 넘긴 상품 액수의 평균

"MOST_FREQ_CREDIT_TYPE" : 가장 빈번하게 쓰이는 CREDIT_TYPE

"HIGHEST_DPD" : 가장 높았던 days past due level (1인 경우는 기한을 넘긴 적이 없는 경우,
5인 경우는 기한을 120일 넘었거나 부채를 탕감한 경우)

Feature Engineering
Previous_Application_Set



Previous application data set은 SK_ID_CURR별로
과거에 같은 회사에서 신청한 신용상품에 대한 정보가 담겨있다

Feature Engineering
Previous_Application_Set

주요 변수 설명

"CNT_CASH" : cash 대출을 신청한 수의 총합

"CNT_POS" : POS 대출을 신청한 수의 총합

"CNT_CARD" : credit card 대출을 신청한 수의 총합

"MEAN_MONTH" : 평균적으로 할부를 건 개월의 수

"DIFF_APPL_FINAL" : 대출 신청 금액과 실제 수령 금액의 차이,

대출 신청 금액과 실제 수령 금액의 차이가 있을 수 있는 이유는 회사 승인 과정에서 실제 대출 가능한 금액이 줄어들 수 있기 때문

Feature Engineering
Previous_Application_Set

주요 변수 설명

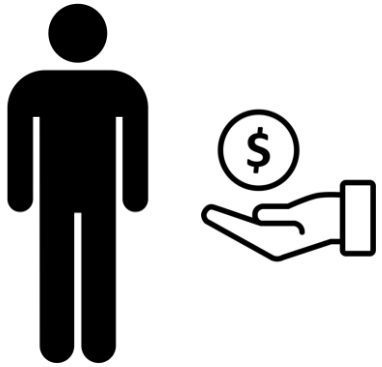
"RATE_APPROVED" : 전체 신청 건수 대비 승인 된 건수의 비율

"RATE_REFUSED" : 전체 신청 건수 대비 승인 되지 않은 건수의 비율

"RATE_NO_PAYMENT" : 전체 신청 건수 대비 상환 방법을 명시하지 않은 건수의 비율

"RATE_WALK_IN" : 전체 신청 건 중 walk-in (신용 평가를 받지 않은 상태)의 비율

Feature Engineering
Previous_Application_Set



Previous application set은 이전 대출 상품의 종류(Credit card, POS cash, Installment)에 따라 더 세분화된 data를 갖고 있었다.

Feature Engineering
Credit_Card_Set

주요 변수 설명

"CONTRACT_STATUS" : 해당 고객의 신용카드 상태(더미화 된 7개의 column 존재)

"credit_card_balance_credit_limit_ratio" : 고객의 신용카드 당월 채무 금액 / 고객의 당월 신용카드 한도

"credit_card_ATM/ALL_Drawings_AMT_Ratio" : ATM 인출 금액 / 전체 신용카드 한도 사용 금액

"credit_card_POS/ALL_Drawings_AMT_Ratio" : 상품 구매 금액 / 전체 신용카드 한도 사용 금액

"credit_card_Other/ALL_Drawings_AMT_Ratio" : 그 외의 금액 / 전체 신용카드 한도 사용 금액

금액 뿐만 아니라 횟수로도 위와 같이 3개의 변수를 생성

Feature Engineering
Credit_Card_Set

주요 변수 설명

“First_DPD_After_n_months” : 첫 신용카드 연체까지 걸린 개월 수(여러 개 카드라면 평균 사용)

“AVG_DPD_TOTAL” : 평균 연체 일

Feature Engineering
Credit_Card_Set

CNT_DRAWINGS_ATM_CURRENT	CNT_DRAWINGS_CURRENT	CNT_DRAWINGS_OTHER_CURRENT	CNT_DRAWINGS_POS_CURRENT
NaN	0	NaN	NaN
NaN	0	NaN	NaN
NaN	0	NaN	NaN
NaN	0	NaN	NaN
NaN	0	NaN	NaN
NaN	0	NaN	NaN
AMT_DRAWINGS_ATM_CURRENT	**AMT_DRAWINGS_CURRENT**	**AMT_DRAWINGS_OTHER_CURRENT**	**AMT_DRAWINGS_POS_CURRENT**
NaN	0.00	NaN	NaN
NaN	0.00	NaN	NaN
NaN	0.00	NaN	NaN
NaN	0.00	NaN	NaN
NaN	0.00	NaN	NaN
NaN	0.00	NaN	NaN

⋮

신용카드를 사용하지 않은 시점의
기록에서는 결측치 발생

↓

소비가 없었다는 의미로 보고
0으로 교체

Feature Engineering
Installment_Set

주요 변수 설명

“GAP_DAYS” : 상환 예정일과 실제 상환일의 차이

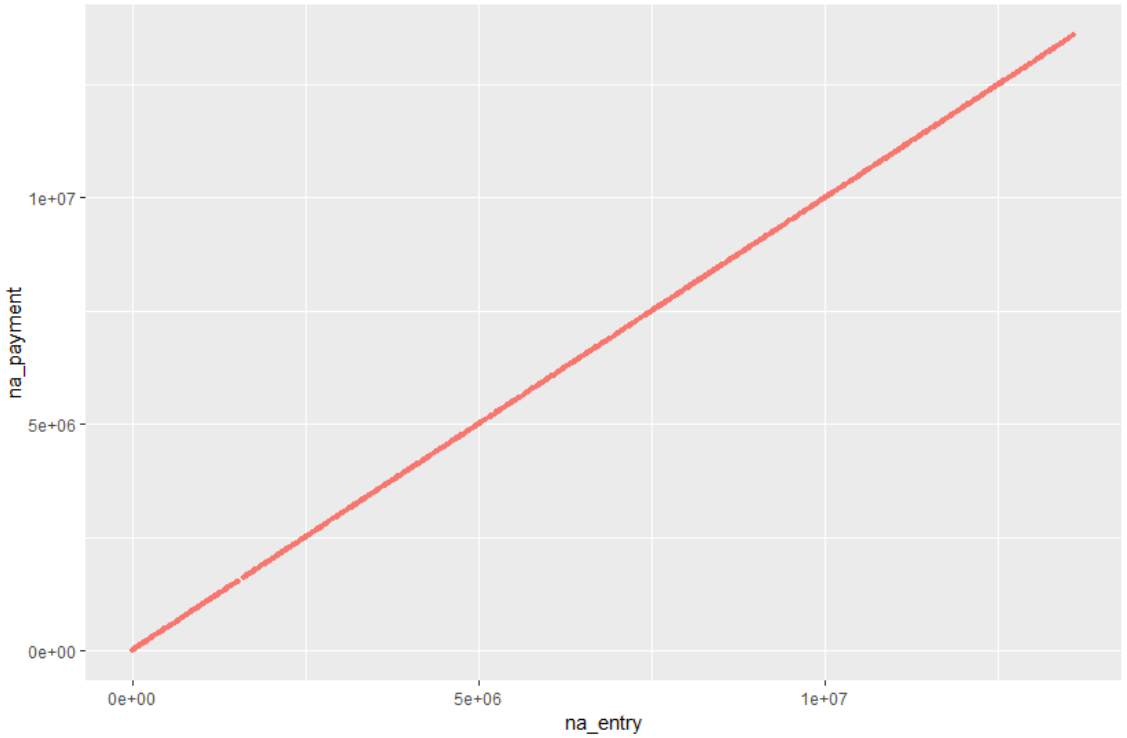
“GAP_AMOUNT” : 상환 예정 금액과 실제 상환 금액의 차이

“AVG_INTALLMENT” : 상환 예정 금액 평균

“AVG_PAYMENT” : 실제 상환 금액 평균

위의 4가지 값들과 함께 sum, average 등의 방법을 사용해 새로운 feature 생성해 냄
새로 생성된 변수 명 : SUM_GAP_DAYS, MEAN_GAP_DAYS, SUM_GAP_AMOUNT, MEAN_GAP_AMOUNT

Feature Engineering
Installment_Set



DAYS_ENTRY_PAYMENT(상환일)가 NA값을 가지는 index와
AMT_PAYMENT(상환액)가 NA값을 가지는 index가 정확히 일치

↓

해당 obs들은 해당 날짜(상환해야 하는 날짜)에 상환하지 못한 것으로 판단하여 0으로 교체

Feature Engineering
POS_Cash_Set

주요 변수 설명

"SK_DPD_SUM" : 상환 예정일과 실제 상환일의 차이

"SK_DPD_MEAN" : 상환 예정 금액과 실제 상환 금액의 차이

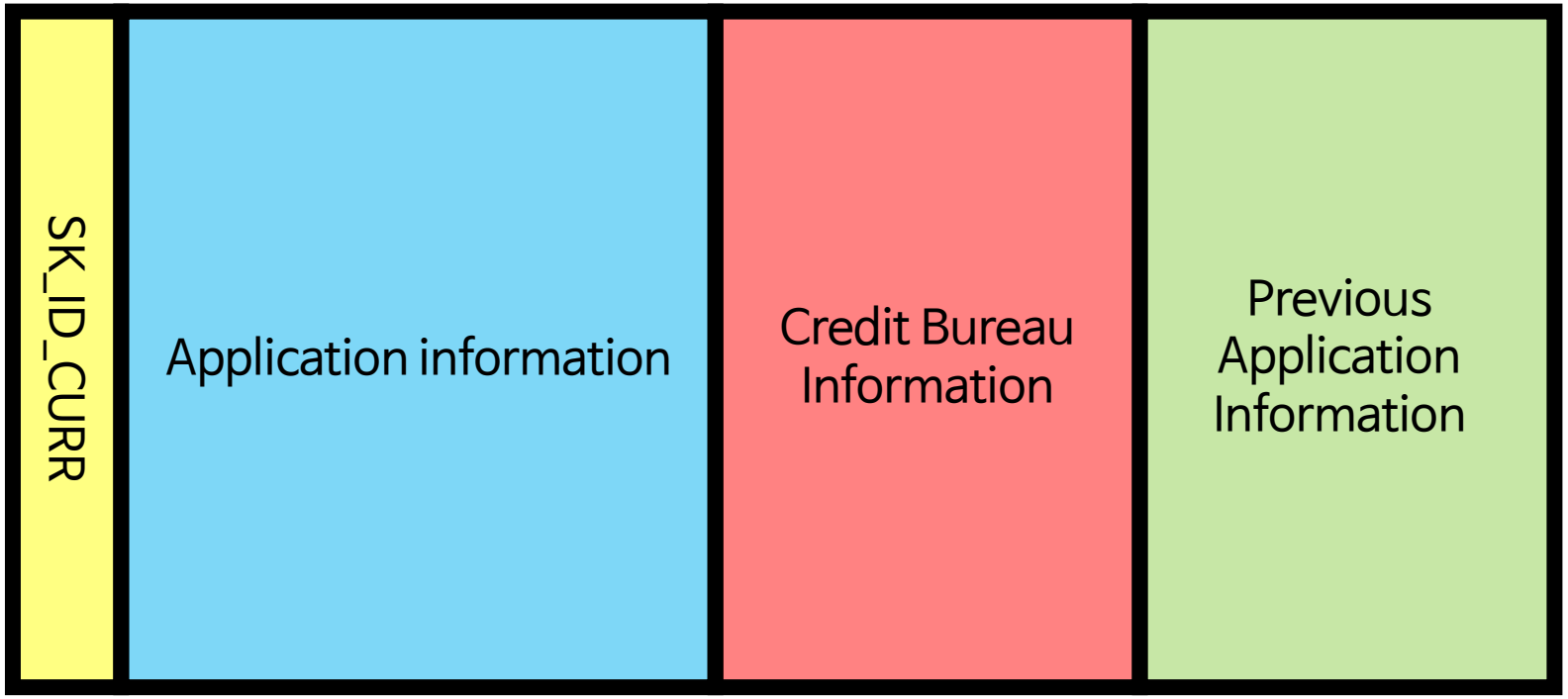
"SK_DPD_DEF_SUM" : 상환 예정일과 실제 상환일의 차이(낮은 부채는 무시)

"SK_DPD_DEF_MEAN" : 상환 예정 금액과 실제 상환 금액의 차이(낮은 부채는 무시)

데이터 통합

307511개의 obs와 237개의 variable을 가진
통합 데이터 셋 완성

데이터 통합



각각의 ID(Loan)가 해당 Loan과 연관된 기본적인 정보와 과거의 신용상품 기록 정보를 가지게 되었다

감사합니다

대출 상환 여부 예측

YBIGTA & P-SAT
연합세미나

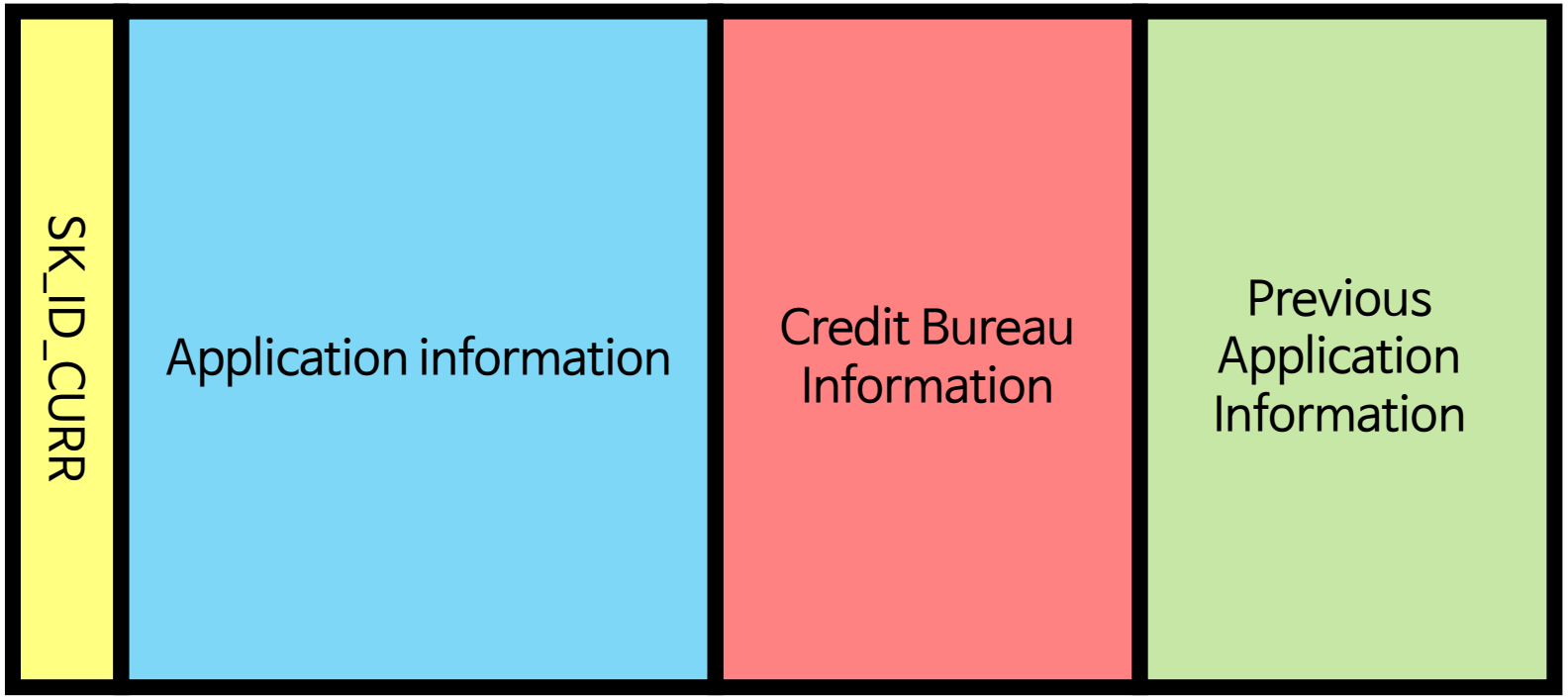
팀 헛개수

황원영
곽지훈
이명진
백상현
나지윤

2주차 목차

- 결측치 처리 : MICE
- 샘플링
- 모델링
  - Boosting
  - NN
- 결론

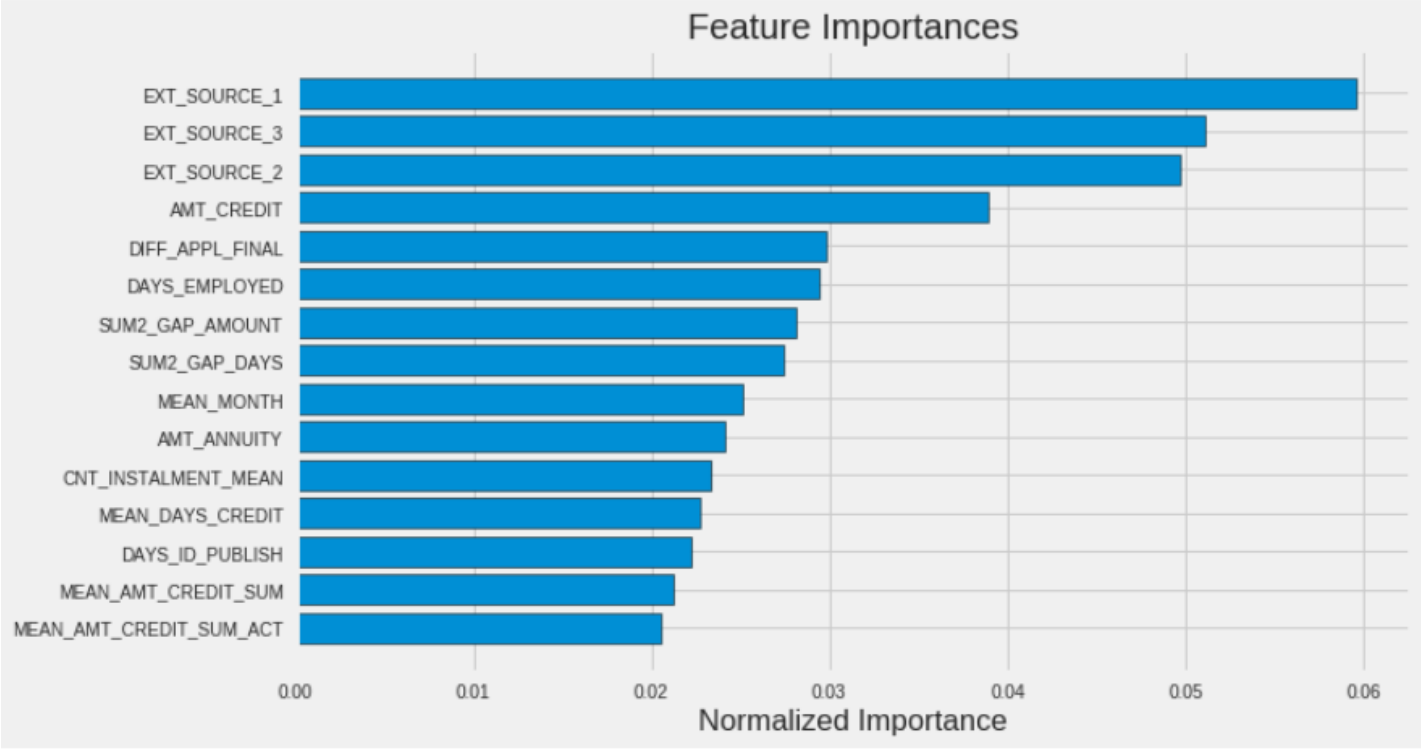
전처리 결과



각각의 ID(Loan)가 해당 Loan과 연관된 기본적인 정보와 과거의 신용상품 기록 정보를 가지게 되었다

전처리 결과

LGBM's importance plot



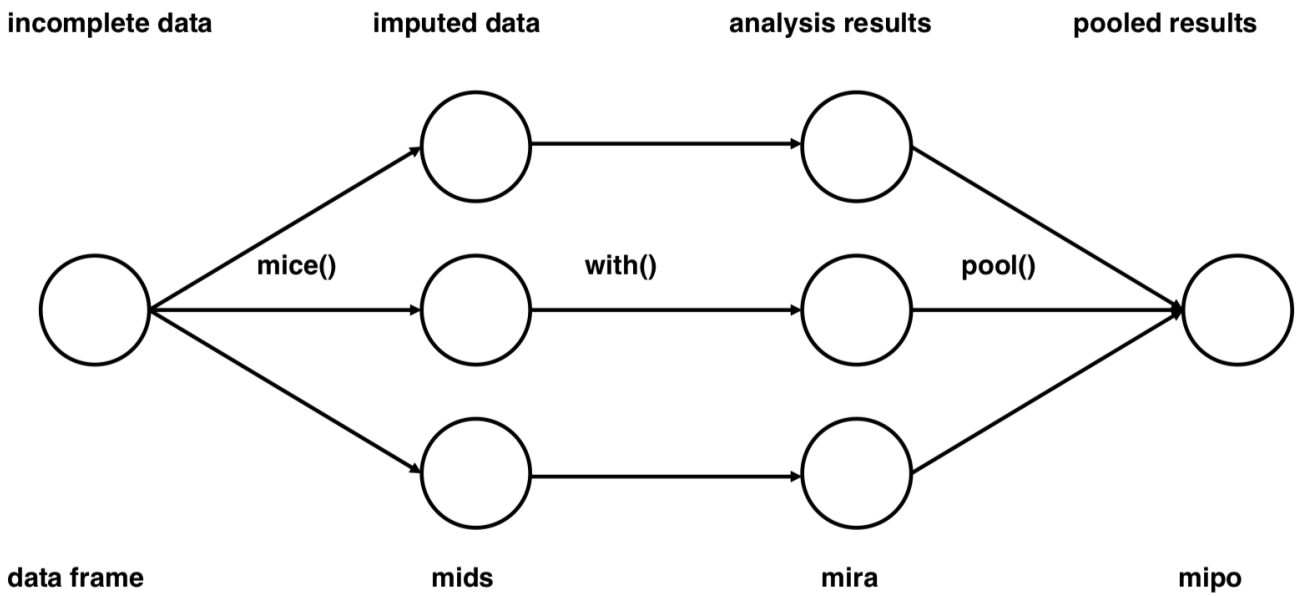
이후 여러가지 모델의 importance plot에서 EXT_SOURCE 변수들이 중요한 변수로 사용되고 있음을 확인했는데,

전처리 결과

EXT_SOURCE 1,2는 비교적 높은 결측치 비율을 가지고 있는 변수였고,
이후 결측치를 자동으로 처리하는 모델에서 해당 변수들 내의 결측값 들이 모두 똑같은 의미로 해석되면 위험하다고 판단
imputation을 진행하기로 함

결측치 처리 | 샘플링 | 모델링 : Boosting | 모델링 : NN | 결론

결측치 처리 : MICE (Multiple Imputation by Chained Equation)



다중대체방법: 여러변수에 걸쳐 존재하는 결측값을 관찰값을 이용하여 예측한다.

결측치 처리 : MICE (Multiple Imputation by Changed Equation)

1. 몇 가지 통계적 기법을 이용하여 결측치에 plausible한 값을 채워 넣은 imputed set을 여러 개 만든다.

2. 각 imputed dataset에서 별도의 통계분석을 시행한다

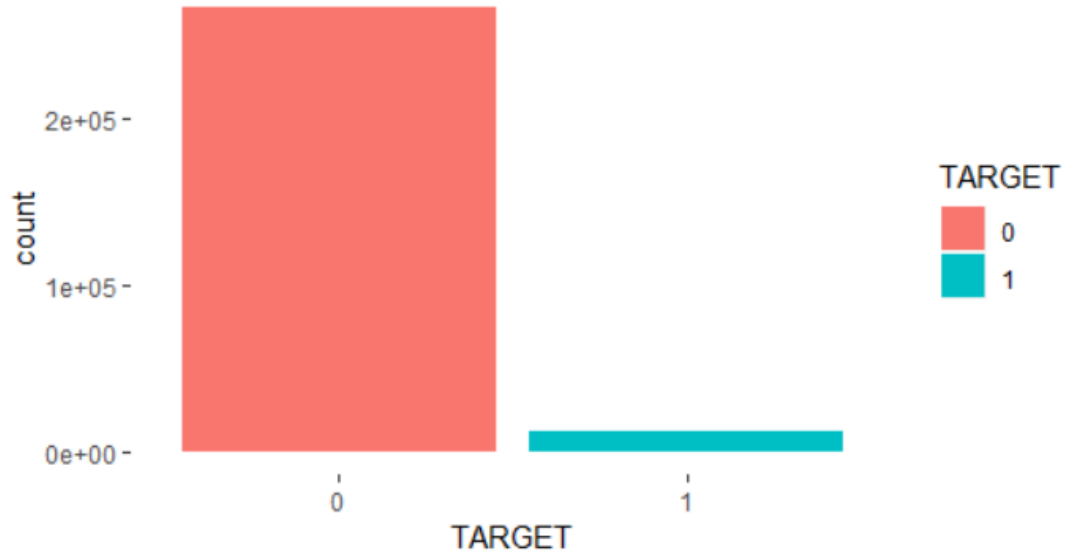
3. 각 imputed dataset의 결과를 pooling한다

결측치 처리 : MICE (Multiple Imputation by Changed Equation)

SK_ID_CURR	EXT_SOURCE_1	EXT_SOURCE_2	EXT_SOURCE_3	...
1000002				
1000003				
1000004				

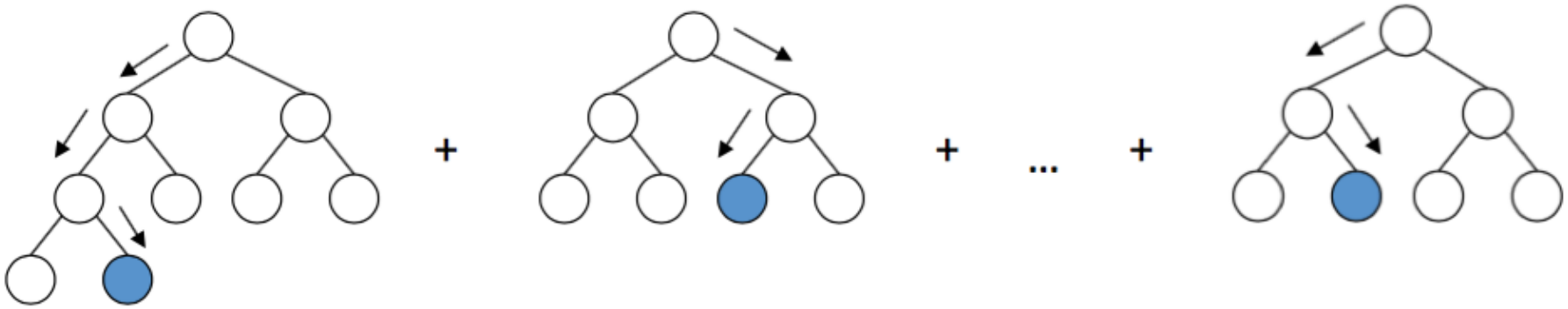
Application set을 사용하여 EXT_SOURCE_1,2,3의 결측치를 모두 채웠다

샘플링



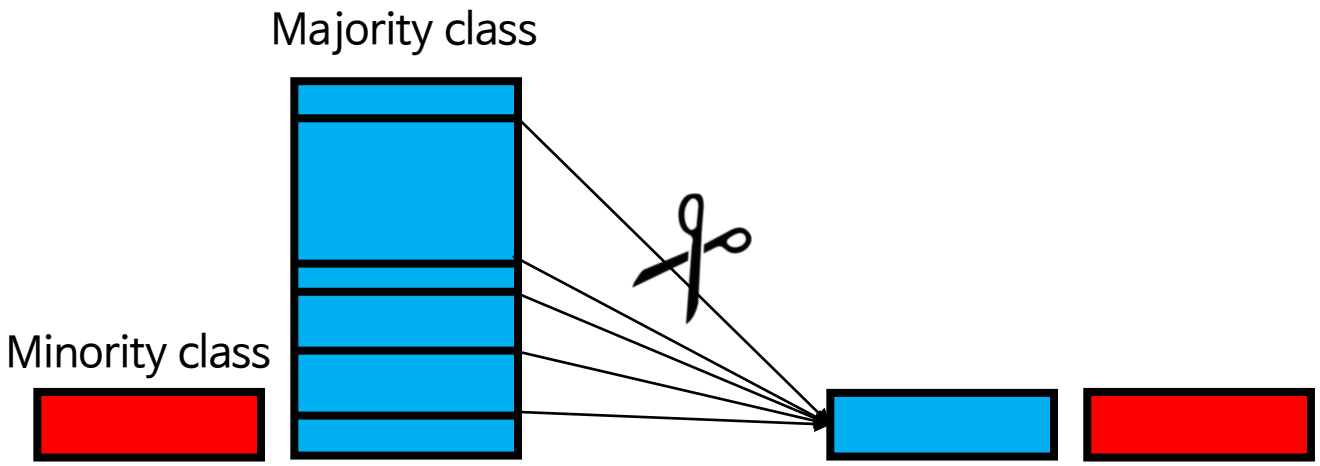
TARGET의 분포는 저번주에 설명했듯 매우 imbalance 했는데 ,

샘플링



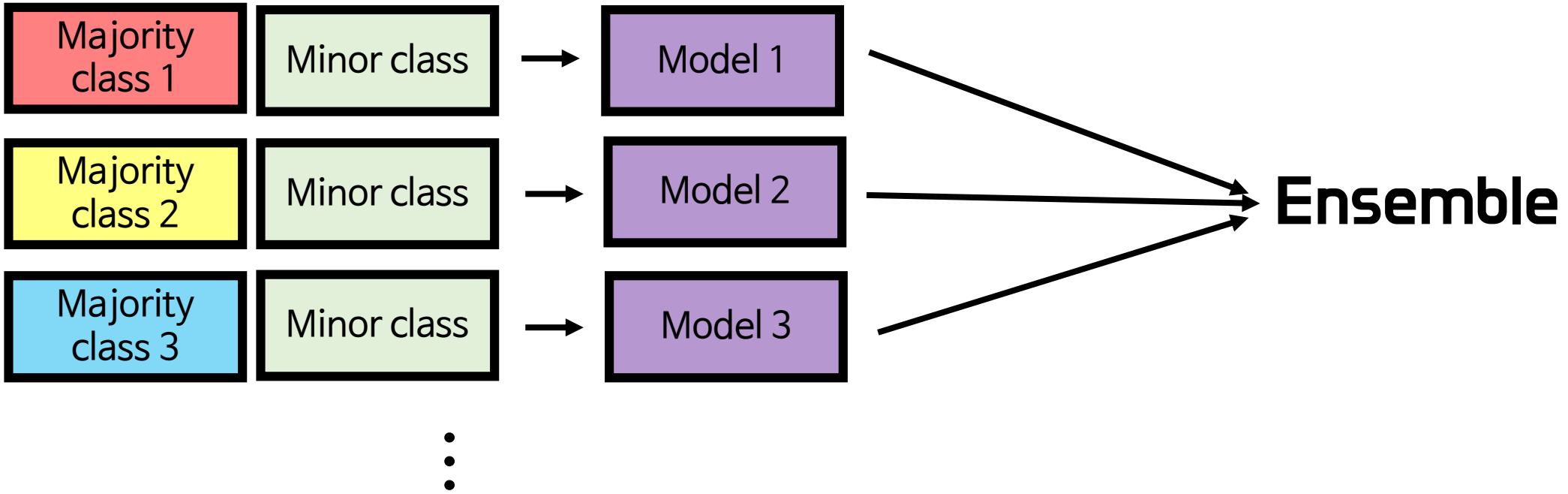
기존 머신러닝 알고리즘은 class 의 proportion, distribution을 고려하지 않고 accuracy를 향상시키려 하기 때문에, minority class의 예측에 있어서는 accuracy가 낮을 확률이 높다

샘플링



이를 해결하기 위한 여러가지 sampling 방법 중
majority target의 balance를 임의로 줄이는 under sampling 방법을 선택함

샘플링



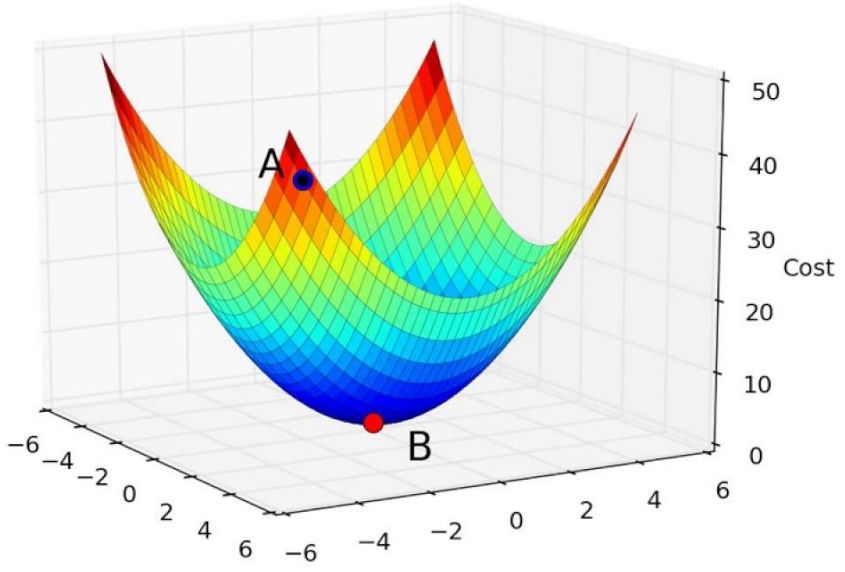
Information loss를 최소화 하기 위해
여러 개의 exclusive한 majority data set을 활용해 다수의 모델을 생성, 앙상블 하기로 했다

모델링 : XGBoost

주요 모델로 **XGBoost** 선정,

- ✓ Prediction에 있어 높은 성능
- ✓ NA value 자동 처리 가능
- ✓ 속도와 정교함 모두 우수

모델링 : XGBoost



✓ *XGBoost* 모델 설명

1. 모델의 residual에 새로운 트리를 fitting

2. gradient descent 를 이용해 loss function 최적화

3. Loss function을 customize 가능

4. F1 score는 미분이 불가능해 손실함수로 설정할 수 없으나 평가지표(eval_metric)로 활용 가능

모델링 : XGBoost

XGBoost *parameter tuning*

다양한 parameter 중 다음 파라미터를 튜닝하기로 함

max_depth

트리의 최대 깊이,
값이 높을수록
overfitting 위험

min_child_weight

split 할 때 child node에
필요한 최소 instance,
값이 높을수록
conservative 한 모델

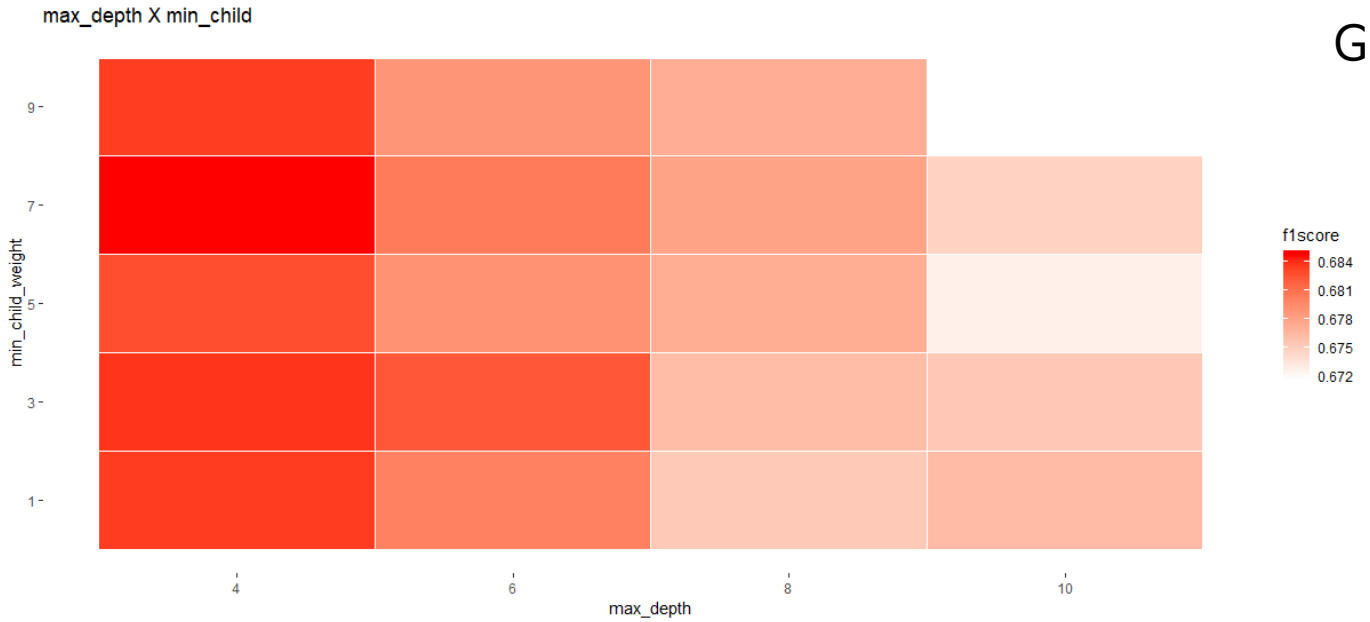
subsample

boosting iteration 마다
추출되는 sample의 비율

colsample_bytree

boosting iteration 마다
추출되는 column의 비율

모델링 : XGBoost



Grid search를 통해 최적 hyperparameter 탐색

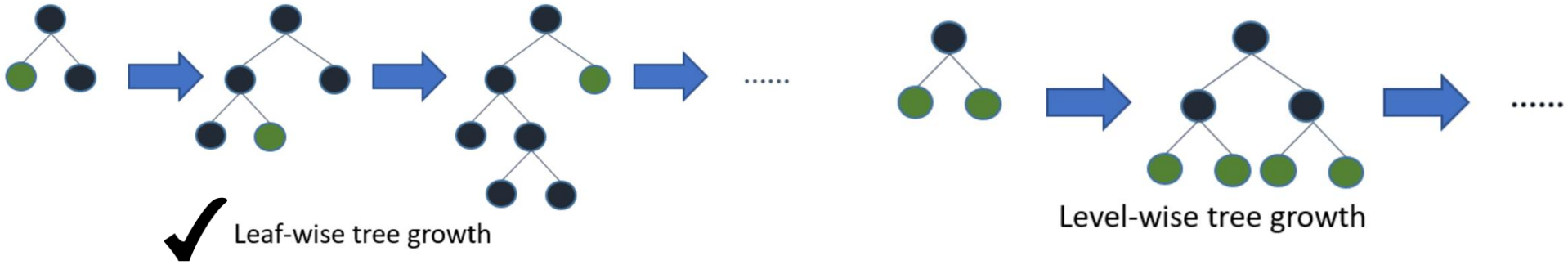
"max_depth" : 4

"min_child_weight" : 7

"subsample" : 0.5

"colsample_bytree" : 0.5

모델링 : LightGBM



Level-wise하게 진행되는 algorithm과 달리,
Leaf-wise하게 모델 학습을 진행 함으로서 손실 감축을 극대화한다

모델링 : LightGBM

✓ *LightGBM* 모델 설명

기존의 Boosting 모델은 각 변수마다 가능한 모든 분할점의 IG(Information Gain)을 평가하기 위해
데이터 개체를 모두 훑기 때문에 고차원 변수를 가진 데이터의 경우 효율성 ↓

LGBM은 변수 개수를 효율적으로 줄이는 방법(GOSS, EFB)을 고안,
정확도의 훼손을 낮추면서 속도를 높인다

하지만 overfitting에 민감하므로 비교적 적은 dataset에는 사용을 지양하는 것을 권장한다고 한다

모델링 : LightGBM

Under-sampling → Model Tuning → Modeling → Prediction

모델링 : LightGBM

100개 이상의 hyperparameter 가 존재

random search로 탐색

0.293494	{'boosting_type': 'gbdt', 'colsample_bytree': 0.8373616754422326, 'is_unbalanc	1	14.56827	0.706506
0.306489	{'boosting_type': 'goss', 'colsample_bytree': 0.6760509284169072, 'is_unbalanc	2	13.27873	0.693511
0.29514	{'boosting_type': 'gbdt', 'colsample_bytree': 0.6556324229784403, 'is_unbalanc	3	14.30105	0.70486
0.305168	{'boosting_type': 'goss', 'colsample_bytree': 0.7611077691986848, 'is_unbalanc	4	17.42898	0.694832
0.321118	{'boosting_type': 'goss', 'colsample_bytree': 0.6833773812382266, 'is_unbalanc	5	5.405916	0.678882
0.309437	{'boosting_type': 'goss', 'colsample_bytree': 0.961487786096467, 'is_unbalance	6	14.22988	0.690563
0.306227	{'boosting_type': 'goss', 'colsample_bytree': 0.880149827462579, 'is_unbalance	7	12.25013	0.693773
0.309635	{'boosting_type': 'goss', 'colsample_bytree': 0.7372708695819636, 'is_unbalanc	8	6.968797	0.690365
0.307449	{'boosting_type': 'goss', 'colsample_bytree': 0.9503391049322831, 'is_unbalanc	9	9.114075	0.692551
0.321263	{'boosting_type': 'goss', 'colsample_bytree': 0.6540065193167791, 'is_unbalanc	10	5.435293	0.678737
0.308061	{'boosting_type': 'goss', 'colsample_bytree': 0.7528785675443, 'is_unbalance': 1	11	8.02683	0.691939
0.320044	{'boosting_type': 'goss', 'colsample_bytree': 0.7640147038777318, 'is_unbalanc	12	4.973127	0.679956
0.307376	{'boosting_type': 'goss', 'colsample_bytree': 0.6452609337437855, 'is_unbalanc	13	10.12023	0.692624

〈주요 파라미터〉

- application: regression / binary / multiclass
- boosting type: gbdt / dart / goss
- num_leaves
- learning_rate
- subsample_for_bin
- min_child_samples
- reg_alpha
- reg_lambda
- colsample_bytree
- subsample
- Is_unbalance

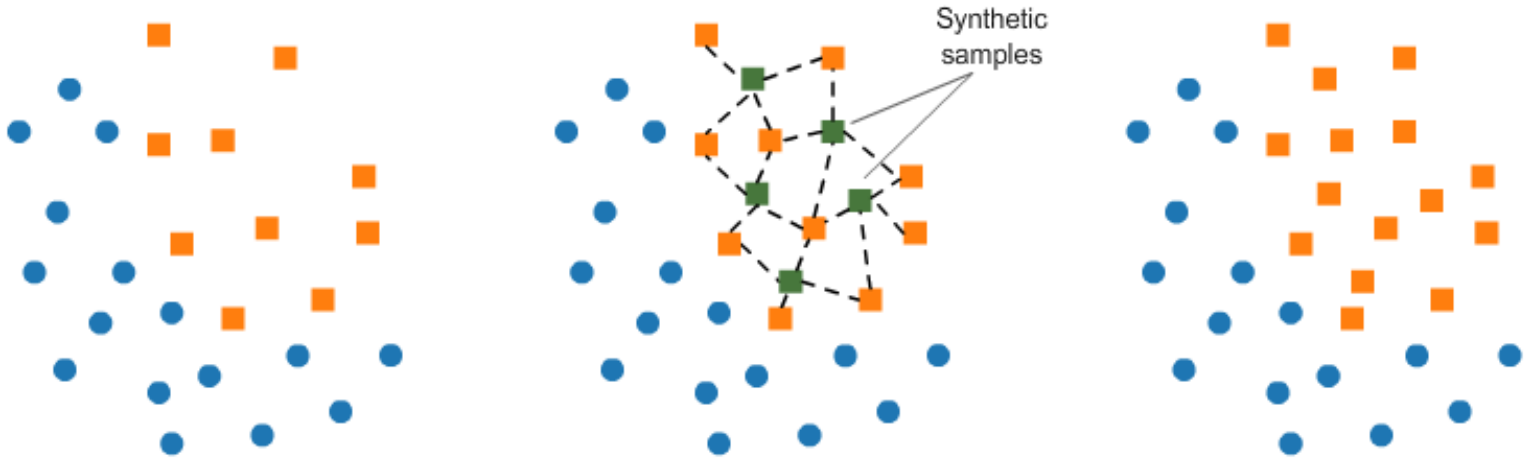
모델링 : Deep Neural Network

SMOTE → Drop Columns (With >70% Null) → Fill NA → Scale Data (Standard Scaler) → Use Data (Train/Test)

모델링 : Deep Neural Network

SMOTE 란?

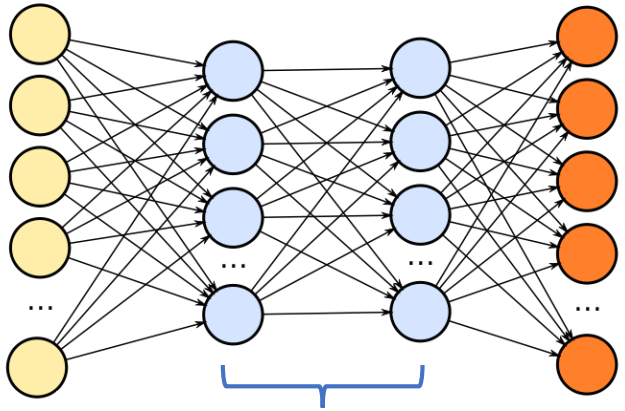
Under-sampling 의 결과가 좋지 않게 나와
SMOTE 활용하기로 함



비율이 낮은 class의 데이터 샘플을 취한 뒤, 각 샘플의 최근접 이웃을 찾고
두 샘플 사이에 임의의 샘플을 생성해낸다

모델링 : Deep Neural Network

Drop column & fill NA



3 Hidden Layer

Neural network 모델은 NA값 자동 처리가 되지 않기 때문에,

결측치 비율이 높은 column(credit card관련 변수들이 대부분)을 삭제하고

나머지 column은 mean imputation 진행함

Deep Neural Network

Input: 241 X 1
Hidden Layer: 3 x 128 nodes
Output: 2 X 1

Optimization Algorithm: Adam

Loss function:
sparse categorical cross-entropy

Metrics: accuracy

In [19]:
```
model = tf.keras.models.Sequential()
model.add(tf.keras.layers.Dense(128, activation=tf.nn.relu,input_dim=X_train.shape[1]))
model.add(tf.keras.layers.Dense(128, activation=tf.nn.relu))
model.add(tf.keras.layers.Dense(128, activation=tf.nn.relu))
model.add(tf.keras.layers.Dense(2, activation=tf.nn.softmax))
```

```
WARNING:tensorflow:From C:₩Users₩naval₩Anaconda3₩lib₩site-packages₩tensorflow₩python₩ops₩resource_variable_ops.py:435: colocate_with (from tensorflow.python.framework.ops) is deprecated and will be removed in a future version.
Instructions for updating:
Colocations handled automatically by placer.
```

In [46]:
```
model.compile(optimizer='adam',
              loss='sparse_categorical_crossentropy',
              metrics=['accuracy'])
model.fit(X_train, y_train, epochs=5)
```

```
Epoch 1/5
171175/171175 [==============================] - 9s 50us/sample - loss: 0.1128 - acc: 0.9553
Epoch 2/5
171175/171175 [==============================] - 8s 48us/sample - loss: 0.1060 - acc: 0.9581
Epoch 3/5
171175/171175 [==============================] - 8s 49us/sample - loss: 0.1005 - acc: 0.9604
Epoch 4/5
171175/171175 [==============================] - 9s 50us/sample - loss: 0.0955 - acc: 0.9627
Epoch 5/5
171175/171175 [==============================] - 8s 48us/sample - loss: 0.0905 - acc: 0.9648
```

Out[46]:
```
<tensorflow.python.keras.callbacks.History at 0x252333b3ac8>
```

결론

제출한 test prediction F-1 score

✓XGBoost : 0.734

CatBoost : 0.733

LGBM : 0.691

DNN : 0.662

결론

HOME CREDIT

활용방안

1. 생성한 모델링을 기반하여 고객의 상환여부를 기업에 알려줌으로써 기업에게 확신을, 고객에겐 양심을 고양시킬 수 있음.
2. 이 기술을 동남아 등 지역에 수출하여 사용함으로써 기술에 대한 신뢰를 확산시켜 다른 분야로 확대시킬 수 있음.

감사합니다