finda

앱사용성 데이터를 통한 대출신청 예측 분석

통계금쪽이들

마주연(jooyeon0203@gmail.com), 신보람(qhfka6628@daum.net),
신해빈(hhbb0030@gmail.com), 정유정(dbwjd516@naver.com)

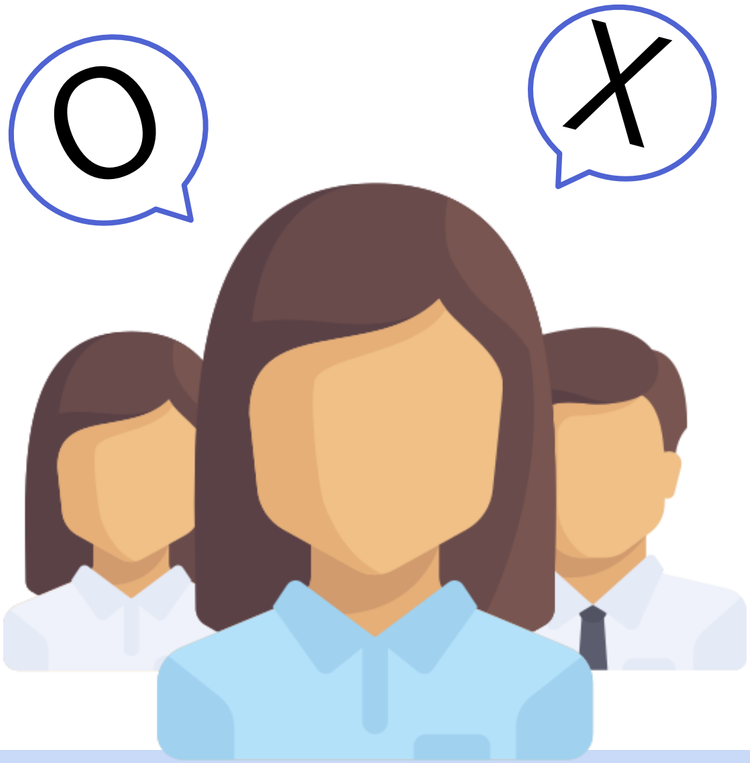
Contents

문제1

01 데이터 소개 및 분석 목적

- 데이터 소개 2022년 3~5월의 가명화된 핀다 앱 사용성 데이터
- 분석 목적 예측 모델을 통한 정확한 대출 신청 예측

application_id	bank_id	product_id	user_id	birth_year	…	is_applied
2157865	54	235	346970	1970.0	…	1
576643	11	118	545882	1977.0	…	0
2136706	22	221	558819	1983.0	…	0
1969227	19	231	109899	1977.0	…	0



01 데이터 소개 및 분석 목적

테이블 소개 USER SPEC (사용자 신용 정보)

날짜형 변수

Columns	변수 설명	Columns	변수 설명
application_id	신청서 번호	insert_time	생성일시
user_id	유저 번호	company_enter_month	입사연월
birth_year	유저 생년월일	gender	유저 성별
desired_amount	기대 한도	income_type	근로형태
existing_loan_cnt	기대출수	employment_type	고용형태
existing_loan_amt	기대출금액	houseown_type	주거소유형태
credit_score	신용점수	purpose	대출 목적
yearly_income	연소득	personal_rehabilitation_yn	개인회생자 여부
		personal_rehabilitation_complete_yn	개인회생자 납입 완료 여부

수치형 변수

범주형 변수

01 데이터 소개 및 분석 목적

테이블 소개 LOAN RESULT (대출 결과)

Columns	변수 설명
application_id	신청서 번호
bank_id	금융사 번호
loanapply_insert_time	한도조회 일시
product_id	상품 번호
loan_limit	승인 한도
loan_rate	승인 금리
is_applied	신청 여부

수치형 변수

날짜형 변수

수치형 변수

Target 변수 (0:신청안함 / 1:신청함)

01 데이터 소개 및 분석 목적

테이블 결합 USER SPEC (사용자 신용 정보) + LOAN RESULT (대출 결과)

은행 상품
대출 금리 및 한도
대출 신청 여부

LOAN RESULT
TABLE

USER SPEC
TABLE

신용 점수
기대출 정보
생년월일 및 수입 정보

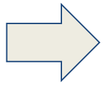
LEFT JOIN

KEY = ‘application_id’

변수 탐색 purpose

- 영어와 한글로 나눠진 클래스 값을 **한글**로 통일

Class1	Class2
LIVING	생활비
SWITCHLOAN	대환대출
BUSINESS	사업자금
ETC	기타
BUYHOUSE	주택구입
HOUSEDEPOSIT	전월세보증금
BUYCAR	자동차구입
INVEST	투자



생활비
대환대출
사업자금
기타
주택구입
전월세보증금
자동차구입
투자

변수 탐색 insert_time & loan_apply_time

Insert_time

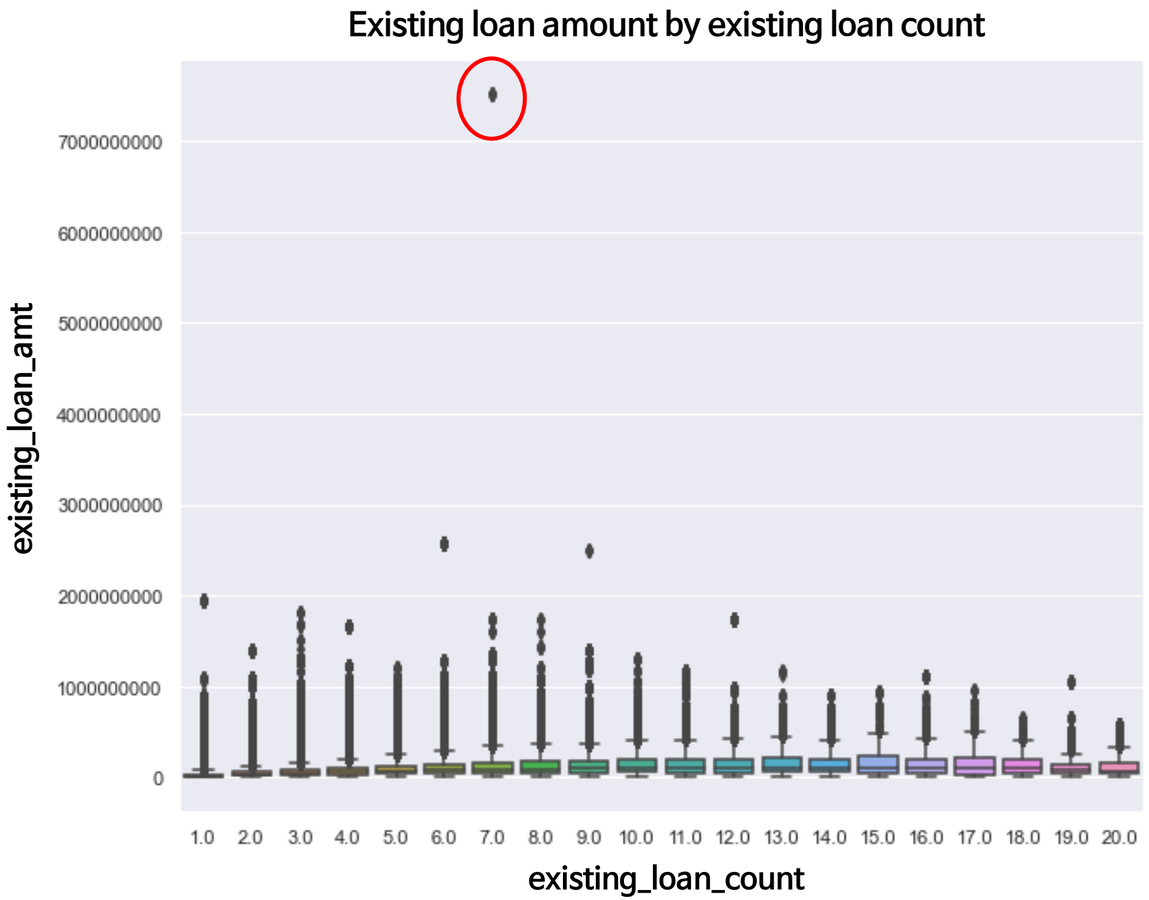
(생성 일시)
application_id별 고유한 값

loan_apply_time

(한도 조회 일시)
application_id 에 있는 여러 개의 은행 상품으로 인해 여러 값 존재

loan_apply_time 변수 **제거**

변수 탐색 existing_loan_cnt & existing_loan_amt



변수 탐색 existing_loan_cnt & existing_loan_amt

existing_loan_cnt (기대출건수)

기대출건수	유저수
51	3
52	1
53	2
⋮	⋮
157	1
178	1

기대출건수가 50 이상일 때,
기대출건수별 유저는 1~3명씩 존재

따라서 기대출건수가 50 이상인 경우,
기대출건수를 50으로 대체

existing_loan_amt (기대출금액)

기대출건수 별 기대출금액 ≧ 2 × 99 percentile

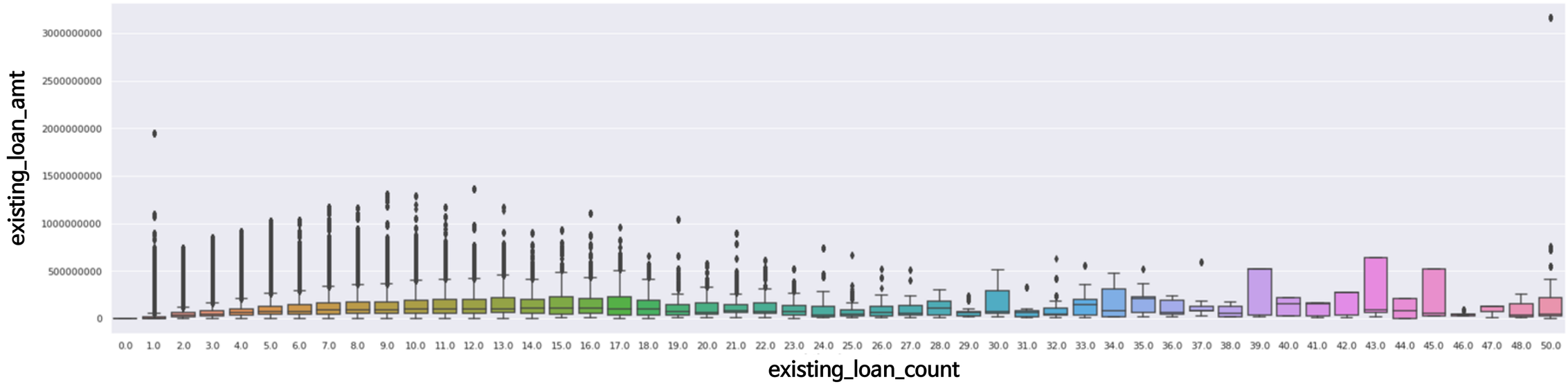
⇨ 2 × 99 percentile 대체

변수 탐색 existing_loan_cnt & existing_loan_amt

existing_loan_cnt
existing_loan_amt

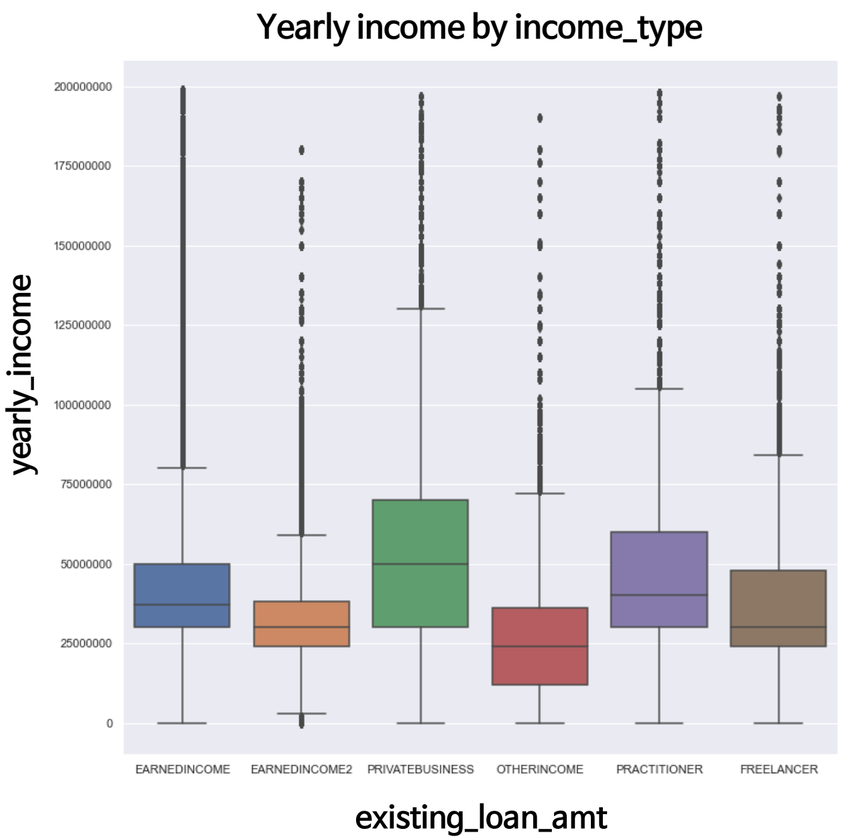
결측치: 0 으로 대체

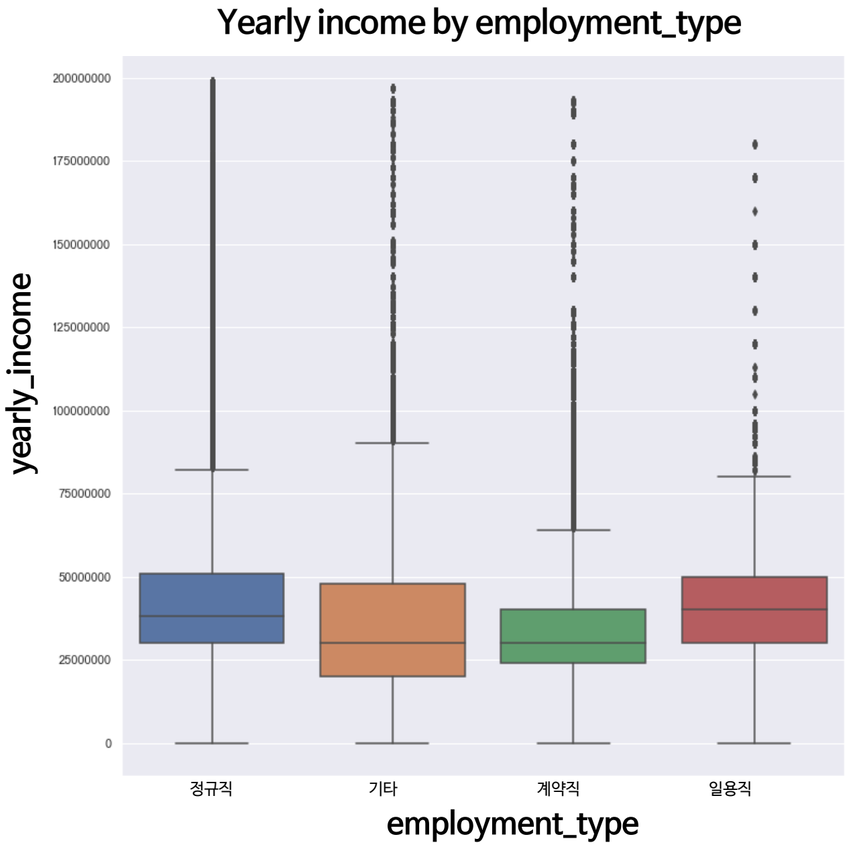
이상치 및 결측치 처리 후 그래프



02 데이터 탐색과 전처리 | 이상치 처리

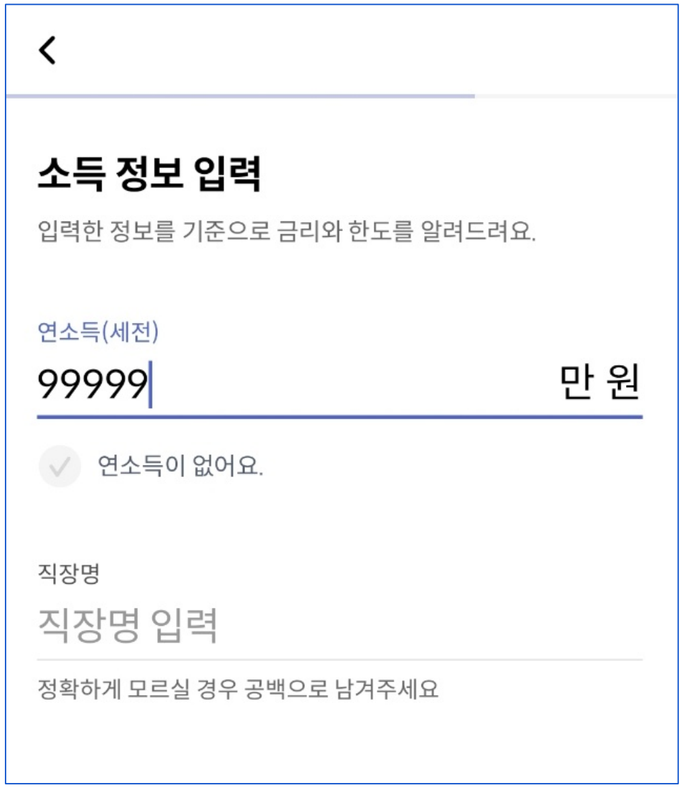
변수 탐색 yearly income





02 데이터 탐색과 전처리 | 이상치 처리

변수 탐색 yearly income



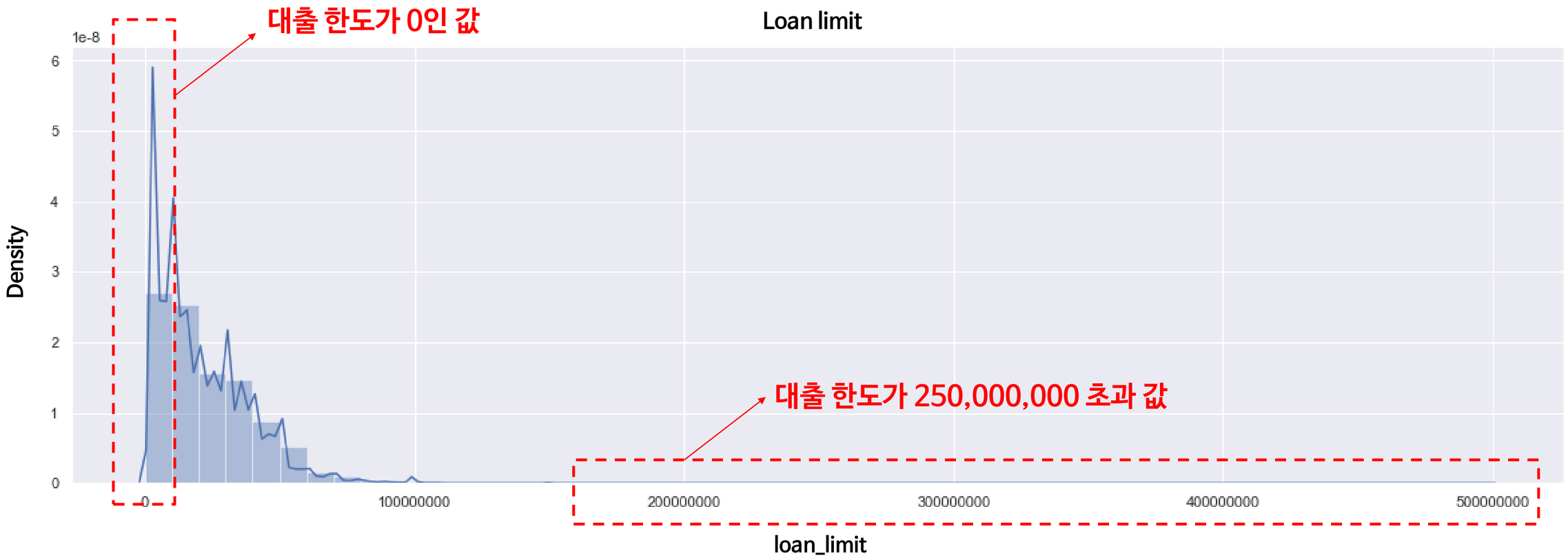
어플 UI 상 **10억 이상** 입력 불가

yearly income과 상관성 있는 변수들의 그룹화된 **중앙값**으로 대체

income_type (근로 형태)	employment_type (고용 형태)	yearly_income_median (연소득 중앙값)
EARNEDINCOME	계약직	30,000,000
EARNEDINCOME	정규직	38,000,000

02 데이터 탐색과 전처리 | 이상치 처리

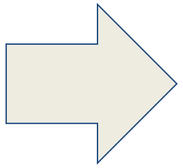
변수 탐색 loan_limit



변수 탐색 loan_limit

loan_limit=0

loan_limit〉 250000000



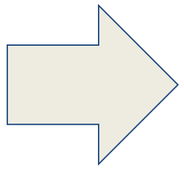
같은 상품 , 금리별 그룹화된 **중앙값**으로 대체

product_id	loan_rate	loan_limit
1	4.8	50,000,000
1	8.9	36,000,000

변수 탐색 loan_limit

loan_limit=0

loan_limit〉 250,000,000

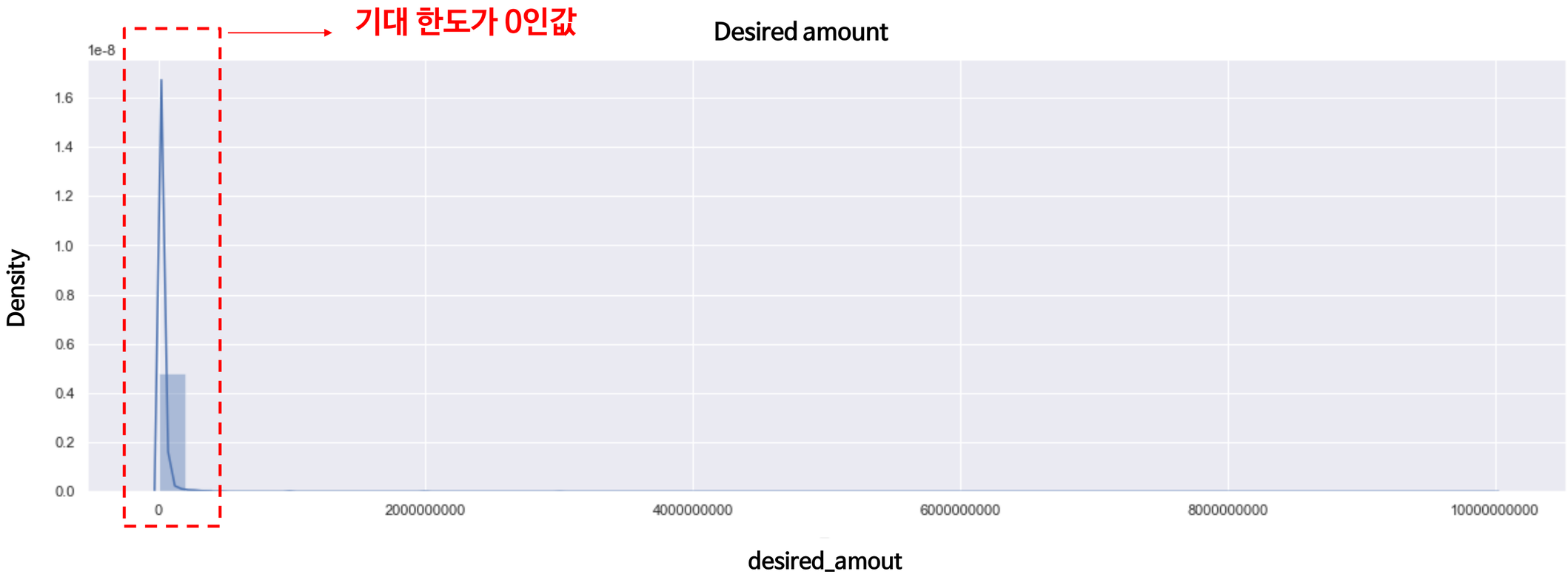


같은 상품의 같은 금리가 존재하지 않는 경우,

바로 **앞, 뒤 금리값**에 해당하는 대출 한도 중앙값으로 대체

product_id	loan_rate	loan_limit
244	7.4	369,000,000
244	7.5	median 392,000,000
244	7.6	415,000,000

변수 탐색 desired_amount



변수 탐색 desired_amount

desired_amount = 0
(기대한도)

100만 단위 반올림 이므로,
0과 500,000의 중간값인 250,000으로 대체

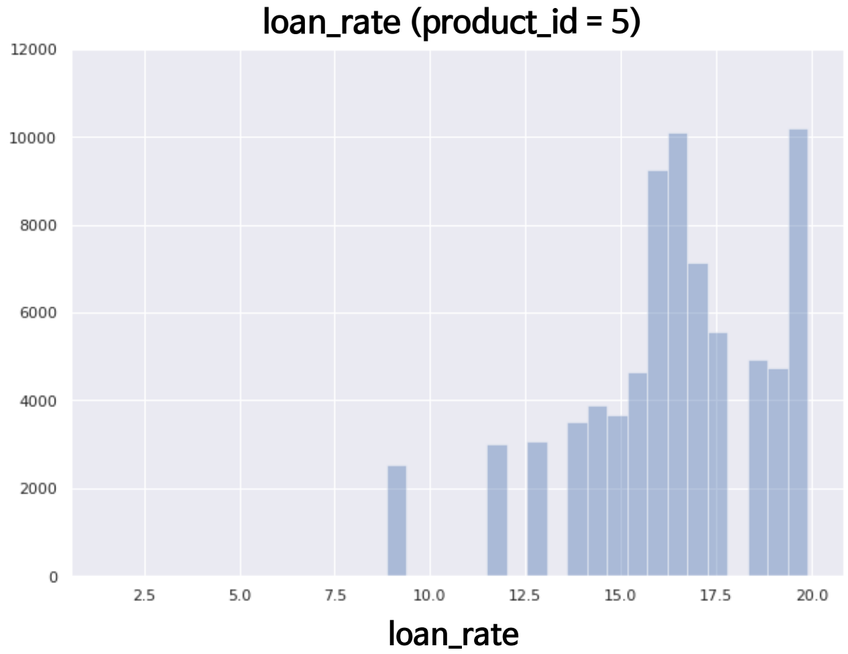
100만 단위 반올림한 desired_amount	실제 desired_amount
0	0 ~ 500,000
1,000,000	500,000 ~ 1,000,000

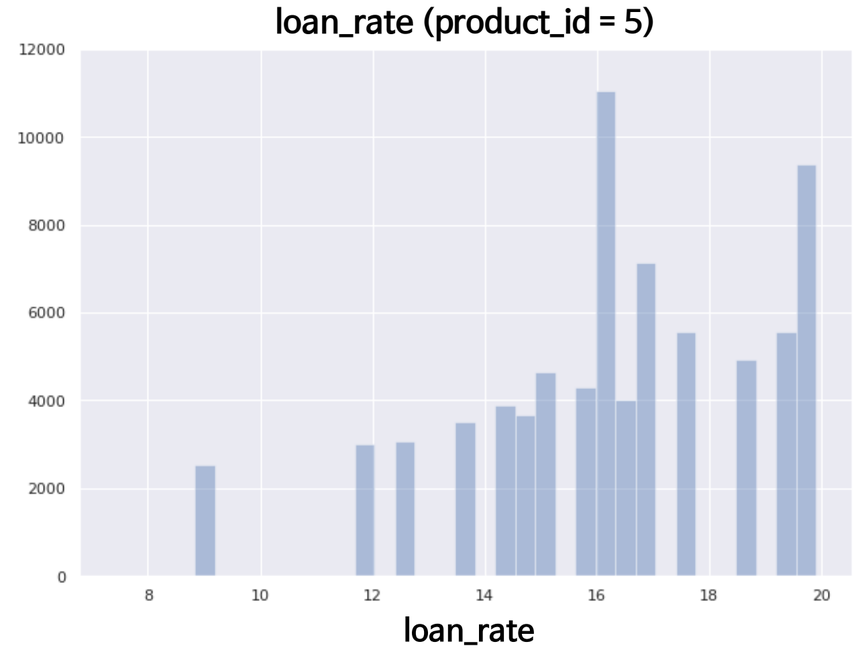
변수 탐색 loan_rate

product_id = 5의 loan_rate : **1.5%**, 7.4%, 8.9%, … , 19.5%, 19.9%

따라서, loan_rate가 1.5%인 경우, **데이터 오류**로 판단

→ **15%**로 대체





변수 탐색 company_enter_month

1) 입사연월이 결측이고, 근로형태가 '기타소득'인 경우

- 입력이 불필요한 값이므로 모두 2023년으로 대체

income_type (근로 형태)	company_enter_month (입사연월)
OTHERINCOME	NA

company_enter_month (입사연월)
2023-01

변수 탐색 company_enter_month

2) 입사연월이 결측이고, 근로형태가 '기타소득'이 아닌 경우 (유저 4명 존재)

- **같은 나이** 고객들의 '**직업별 입사 만나이 중앙값 + 생년월일**'을 이용하여 대체

user_id	income_type	birth_year	company_enter_month
174784	EARNEDINCOME	1998	2021-01
418233	PRIVATEBUSINESS	1985	2019-01
776609	EARNEDINCOME	1991	2020-01
861913	EARNEDINCOME	1980	2019-01

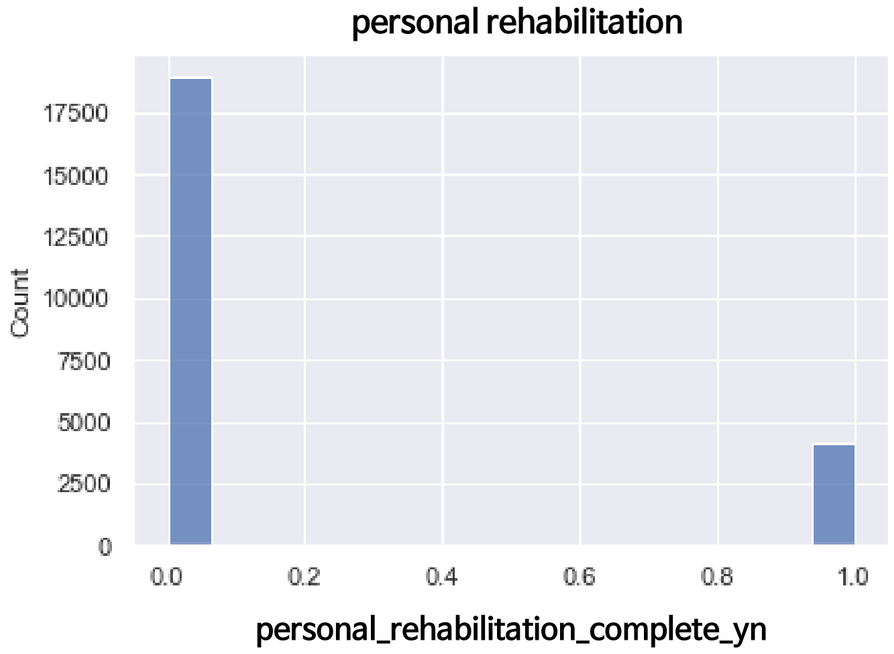
변수 탐색 company_enter_month

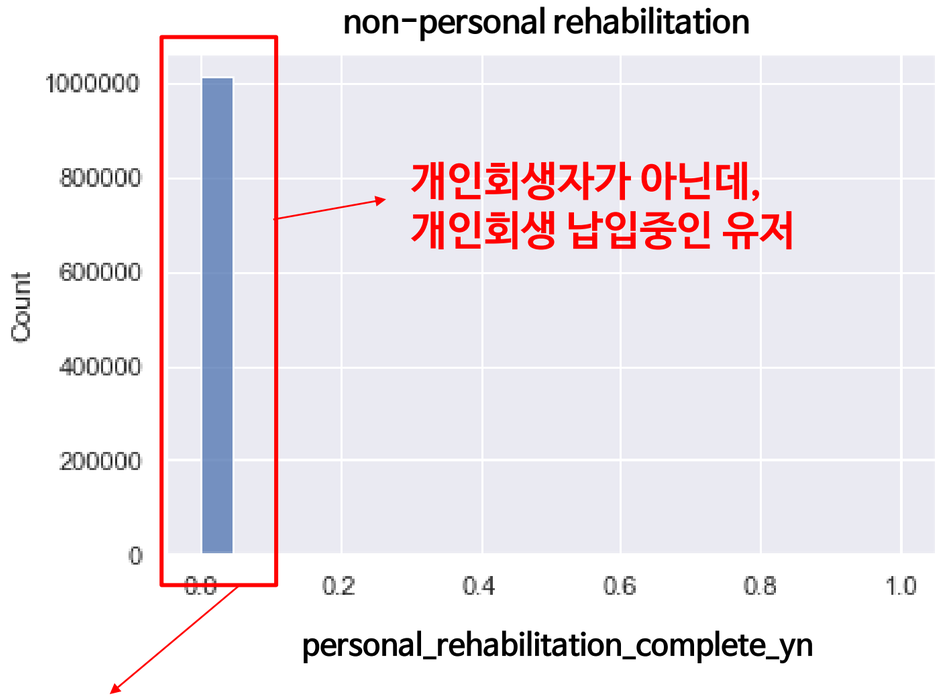
- 생년월일보다 입사년도가 오래되거나 같은 행 709개 존재
- **같은 나이** 고객들의 '**직업별 입사 만나이 중앙값 + 생년월일**'을 이용하여 대체

user_id	income_type	birth_year	company_enter_month
346970	EARNEDINCOME	1970	2013-12
545882	EARNEDINCOME	1977	2019-08
558819	EARNEDINCOME	1983	2012-08
341662	EARNEDINCOME	1991	2021-08

변수 탐색 personal_rehabilitation_yn & personal_rehabilitation_complete_yn

0 : 개인회생 납입중 / 1: 개인회생 납입완료

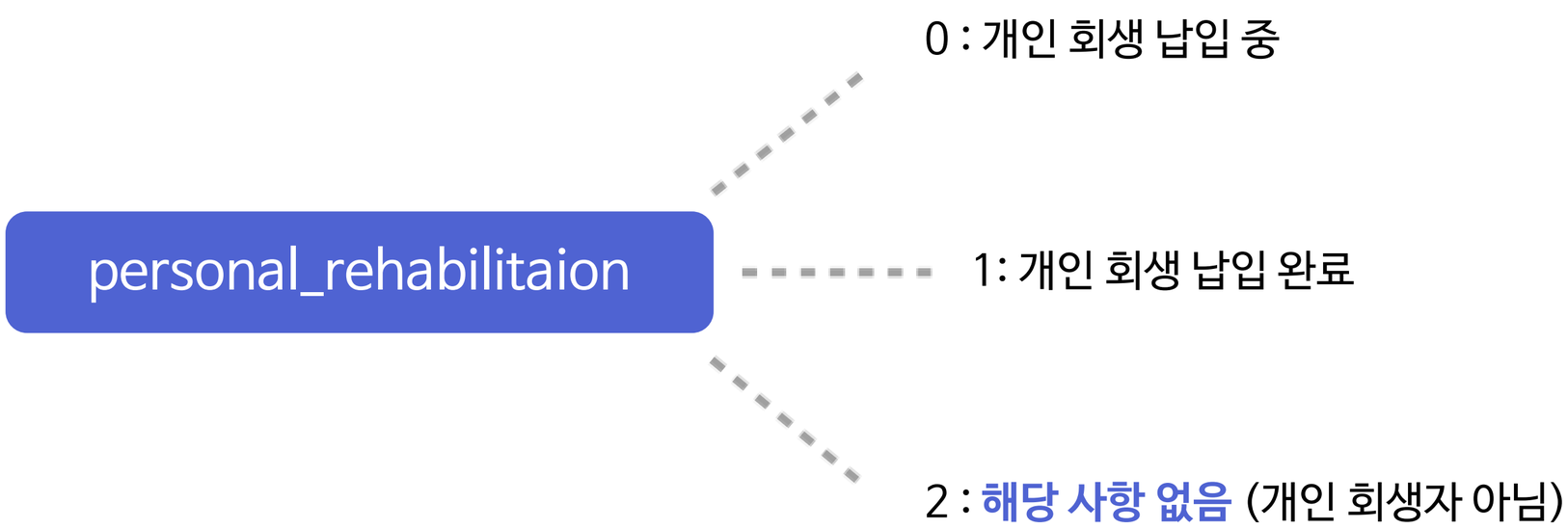




personal_rehabilitation_yn 값을 1로 대체

변수 탐색 personal_rehabilitation_yn & personal_rehabilitation_complete_yn

- ‘개인회생자 여부’ 변수와 ‘개인회생자 납입 완료 여부’ 변수를 합쳐 하나의 변수로 표현



변수 탐색 credit_score

신용점수는 다 섯가지 항목으로 평가됩니다. **첫 번째는 상환 이력**. 과거의 채무 상황을 봤을 때 얼마나

⋮

간단히 정리하면, 기존 대출과 카드값, 각종 세금, 통신비 등을 연체하지 않고 제때 내고 있는지, 직장이나 직업, 연봉 수준, 기존 자산, 금융거래 내역 등을 분석해서 돈을 갚을 수 있는 능력과 의지가 얼마나 있는지를 점수로 낸다는 거죠.

※ 출처: finda 포스트, '신용점수 전격 해부, 전략만 잘 세워도 신용점수가 올라갑니다.' (2022.05.13)

신용점수 평가 방법 : **상환이력, 부채 수준, 연봉 수준, 금융거래 내역** 등을 분석

변수 탐색 credit_score

상환이력, 부채 수준, 연봉 수준, 금융거래 내역과 관련있는 변수 선택

personal_rehabilitation(개인회생납입여부), income_type(근로 형태),
existing_loan_cnt(기대출건수), existing_loan_amt(기대출금액),
yearly_income(연소득)

과적합 가능성이 낮으며 빠르고 정확도가 높은
Random Forest를 이용하여 결측치 대체

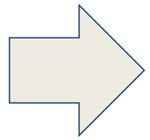
파생 변수 company_enter

- 한도조회일시 기준으로 입사한지 몇 개월이 지났는지를 나타내는 '재직 기간' 변수

insert_time	insert_date	company_enter_month	company_enter
2022-05-09 08:44:57	2022-05-09	2013-12-01	101
2022-05-09 10:54:51	2022-05-09	2019-08-01	33
2022-05-09 10:41:05	2022-05-09	2012-03-01	122
2022-05-09 09:15:31	2022-05-09	2021-08-01	9

- 입사연월이 음수인 경우, 모두 무직이므로 company_enter(재직 기간) 변수를 0으로 채워줌

income_type	insert_date	company_enter_month	company_enter
OTHERINCOME	2022-05-09	2023-01	-8



company_enter
0

파생 변수 insert_hour

- ‘한도조회 당시 시간대’를 새벽(0), 오전(1), 오후(2), 밤(3)으로 분류

insert_time	insert_hour
2022-05-09 00:25:07	0
2022-05-09 05:15:02	0
2022-05-09 08:44:57	1
2022-05-09 10:41:05	1
2022-05-09 10:54:51	1
2022-05-09 13:39:36	2
2022-05-09 17:33:24	2
2022-05-09 19:56:54	3
2022-05-09 18:37:17	3

파생 변수 product_freq

- ‘대출 상품별 출현 비율’을 나타내는 변수

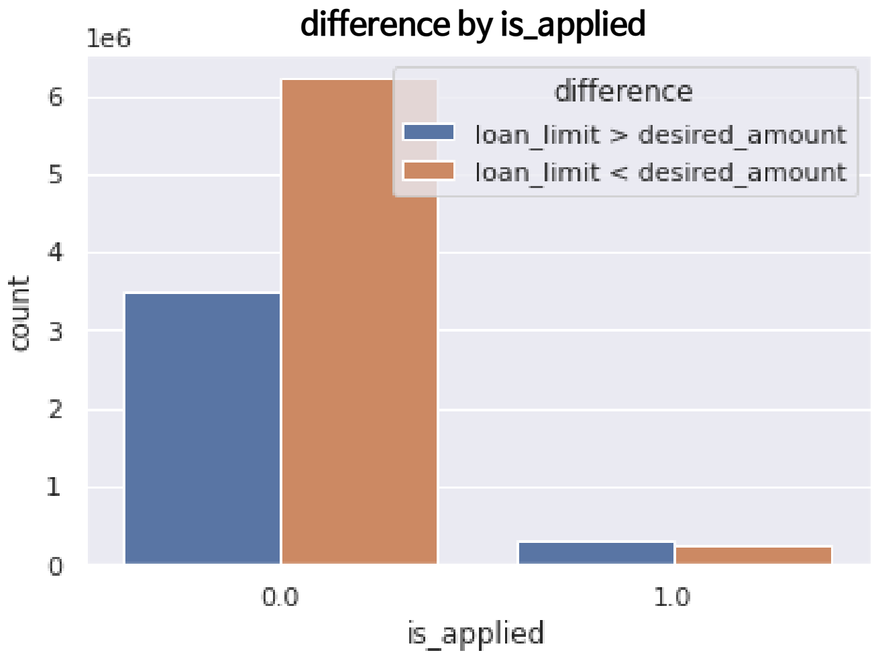
product_id	product_freq
1	0.088779
5	0.104571
7	0.039903
8	0.155382
12	0.035193
15	0.003282
16	0.186274
19	0.222622
20	0.009502

$$\text{product\_freq} = \frac{\text{해당 상품의 출현 빈도}}{\text{전체 application\_id 개수}}$$

파생 변수 diff

- 대출승인한도(loan_limit)와 대출희망금액(desired_amount) 차이

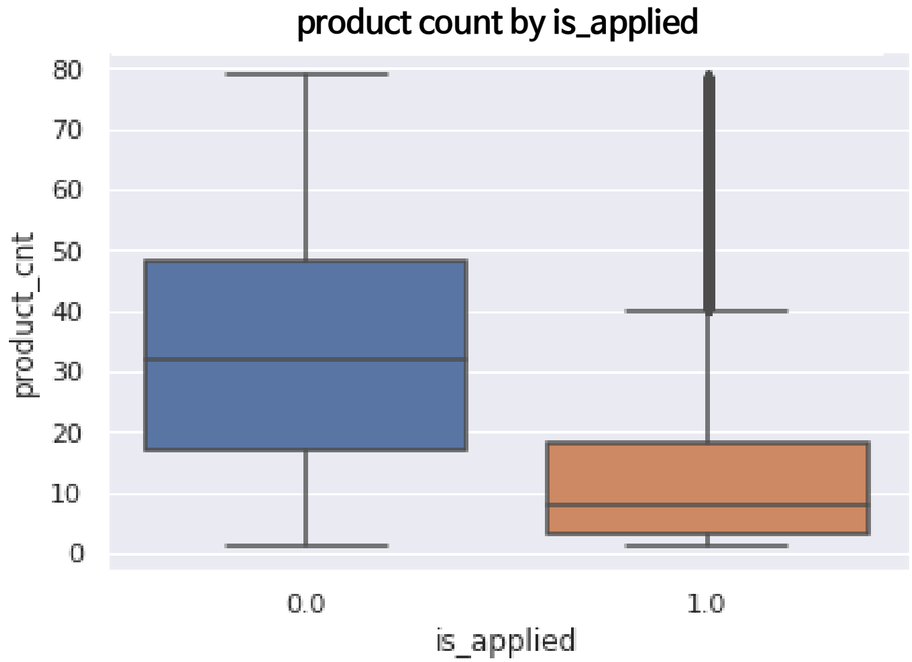
loan_limit	desired_amount	diff
20,000,000	10,000,000	10,000,000
11,000,000	20,000,000	-9,000,000
3,000,000	20,000,000	-17,000,000
10,000,000	80,000,000	-70,000,000
6,000,000	1,000,000	5,000,000
5,000,000	2,000,000	3,000,000
46,000,000	3,000,000	43,000,000
26,000,000	25,000,000	1,000,000
2,000,000	25,000,000	-23,000,000



파생 변수 product_cnt

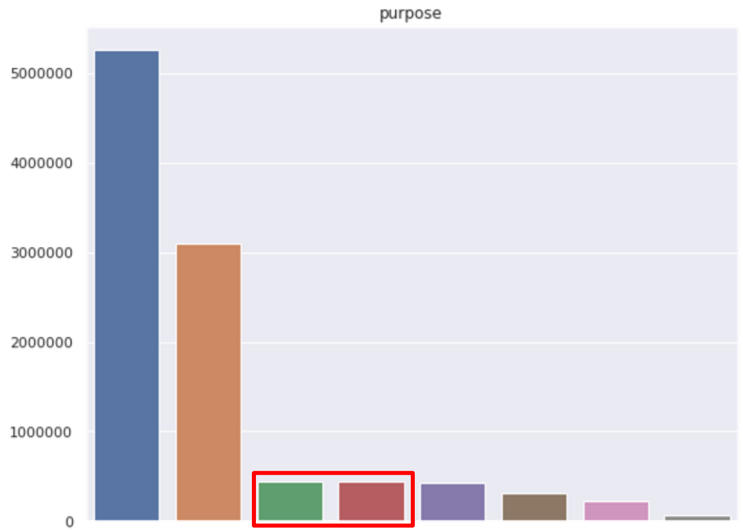
- application_id별 추천된 대출 상품 개수

application_id	product_cnt
2157865	1
576643	2
2136706	40
380021	52
949973	4
679142	24
1831090	10
1123617	9
178618	1

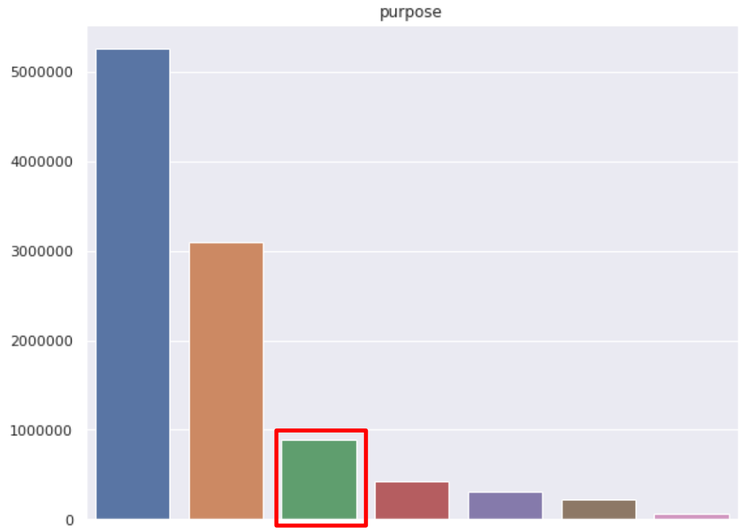


재범주화 purpose

- 범주가 너무 많으면 모델 성능에 좋지 않은 영향을 주므로 범주의 개수를 줄여줌
- 비슷한 범주끼리 묶어줌



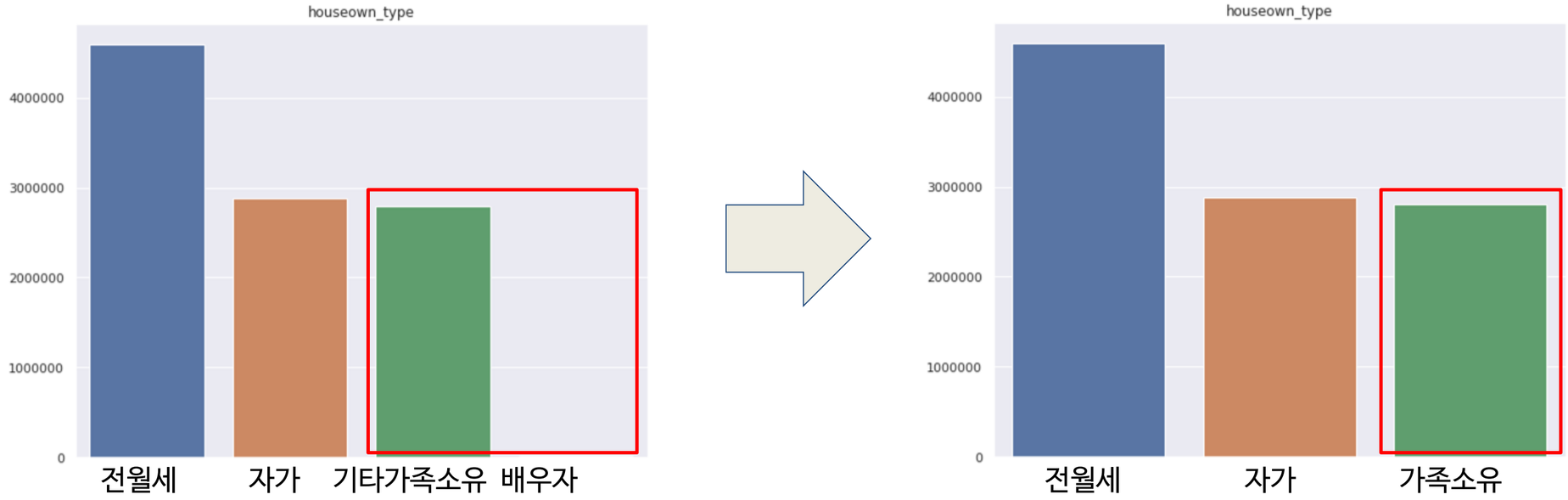
전월세보증금, 주택구입



집

재범주화 houseown_type

- 범주가 너무 많으면 모델 성능에 좋지 않은 영향을 주므로 범주의 개수를 줄여줌
- 비슷한 범주끼리 묶어줌



재범주화 income_type & employment_type

- 근로형태와 고용형태를 하나로 묶어 'income'변수 생성

- 직장가입자

income_type	employment_type	income
EARNEDINCOME	정규직	정규직1
EARNEDINCOME	계약직	기타1
EARNEDINCOME	일용직	기타1
EARNEDINCOME	기타	기타1
EARNEDINCOME2	정규직	정규직2
EARNEDINCOME2	계약직	기타2
EARNEDINCOME2	일용직	기타2
EARNEDINCOME2	기타	기타2

EARNEDINCOME : 직장가입자(4대보험 O)

EARNEDINCOME2 : 직장가입자(4대보험 X)

재범주화 income_type & employment_type

- 프리랜서

income_type	employment_type	income
FREELANCER	정규직	프리랜서
FREELANCER	계약직	프리랜서
FREELANCER	일용직	프리랜서
FREELANCER	기타	프리랜서

- 전문직

income_type	employment_type	income
PRACTITIONER	정규직	전문직
PRACTITIONER	계약직	전문프리랜서
PRACTITIONER	일용직	전문프리랜서
PRACTITIONER	기타	전문프리랜서

재범주화 income_type & employment_type

- 개인사업자

income_type	employment_type	income
PRIVATEBUSINESS	정규직	개인사업자
PRIVATEBUSINESS	계약직	개인사업자
PRIVATEBUSINESS	일용직	개인사업자
PRIVATEBUSINESS	기타	개인사업자

- 기타소득

income_type	employment_type	income
OTHERINCOME	정규직	무직
OTHERINCOME	계약직	무직
OTHERINCOME	일용직	무직
OTHERINCOME	기타	무직

Label Encoding **income, purpose, houseown_type**

- 문자열 값을 수치형으로 인코딩

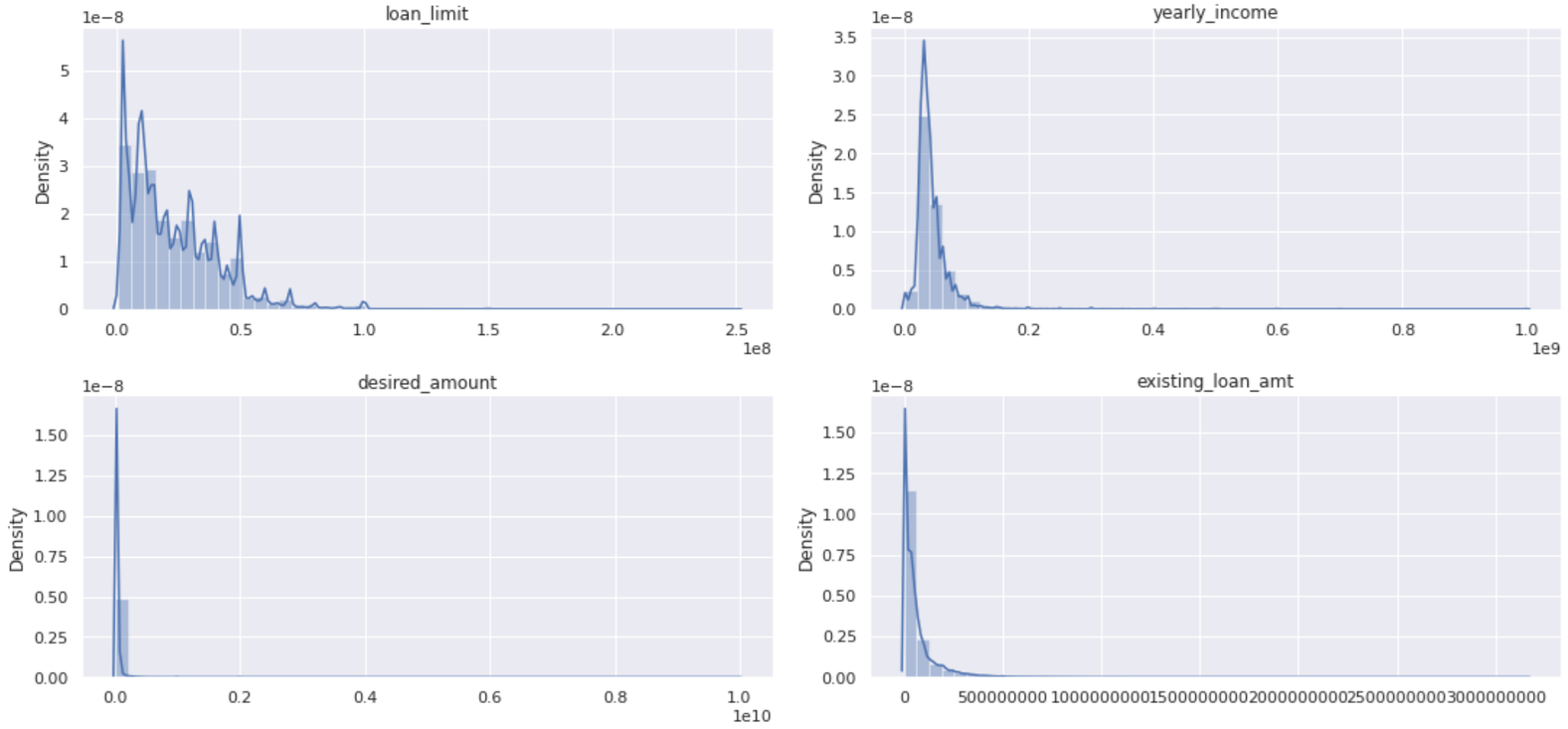
income	label encoding
개인사업자	0
기타1	1
기타2	2
무직	3
전문직	4
전문프리랜서	5
정규직1	6
정규직2	7
프리랜서	8

purpose	label encoding
기타	0
대환대출	1
사업자금	2
생활비	3
자동차구입	4
집	5
투자	6

houseown_type	label encoding
가족소유	0
자가	1
전월세	2

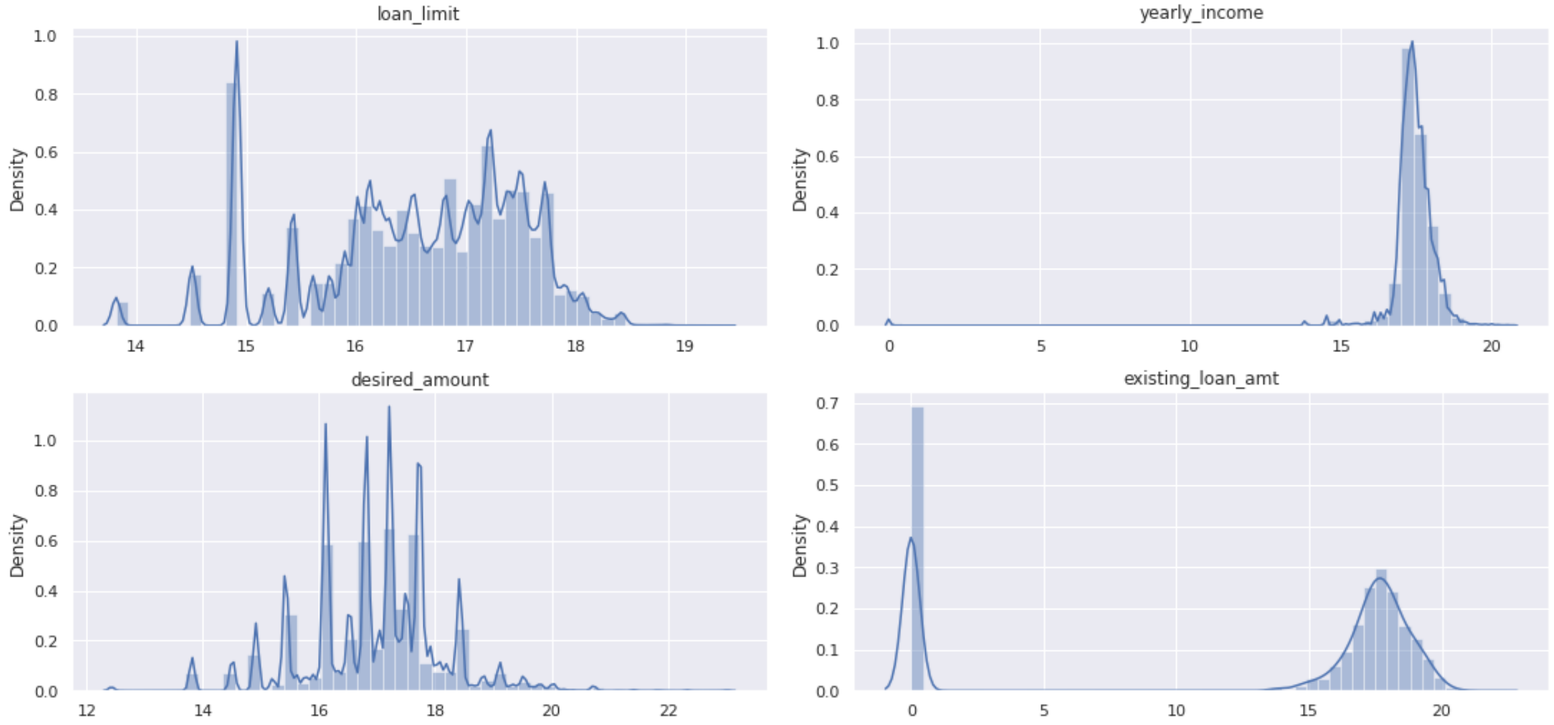
수치형 변수 분포 확인

- 매우 심한 right-skewed distribution



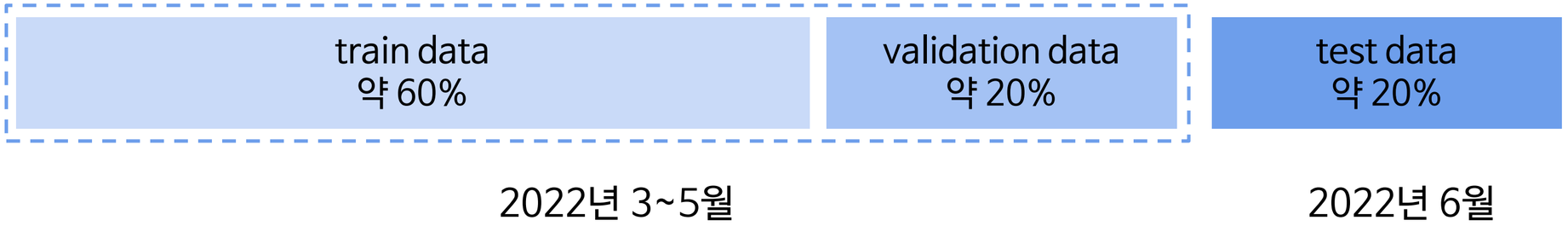
수치형 변수 분포 확인

- 로그 변환 후, 수치형 변수 히스토그램



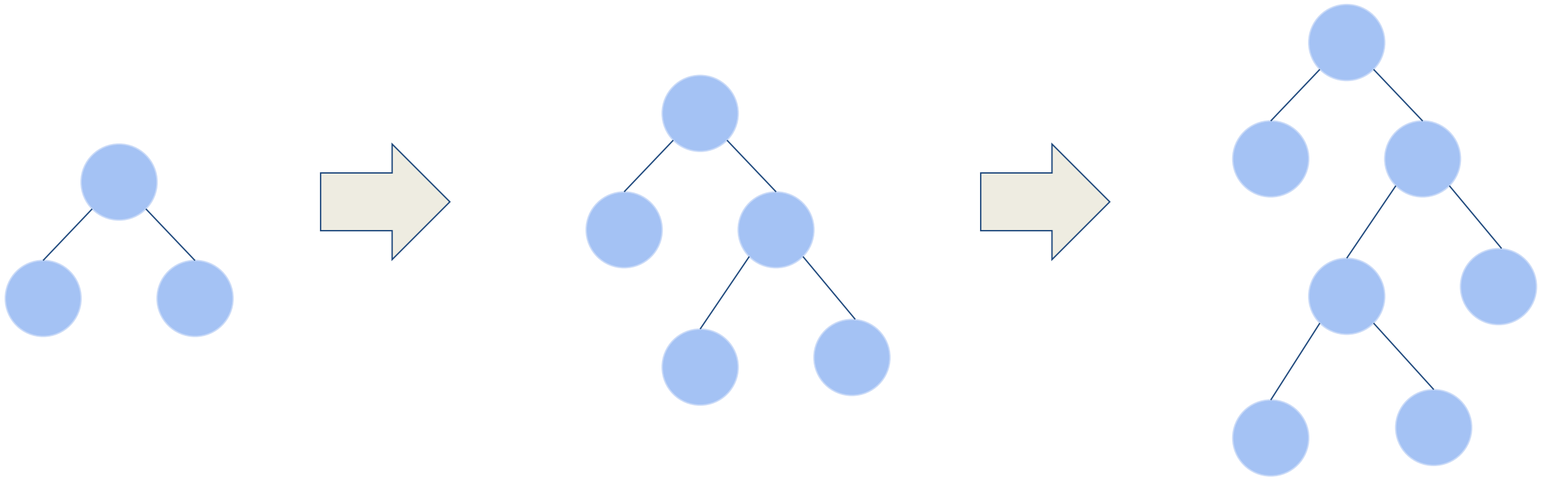
03 모형 구축 | 학습 및 평가 데이터 분할

train data, validation data, test data 분할



LGBM(Light Gradient Boosting Model)

- 기존의 다른 Tree 기반 알고리즘과 다르게 Tree를 수직적으로 확장하는 리프 기준 분할 방식
- Tree의 균형을 맞추지 않고 최대 손실 값을 갖는 리프 노드만 분할
- 적은 메모리 사용량 및 빠른 학습 시간이 장점
- 범주형 변수를 원-핫 인코딩하지 않고, 그대로 사용 가능



모델링에 사용할 변수 선택

1) id변수 제거

- application_id, user_id, product_id, bank_id

2) gender, birth_year 제거

- 해당 변수와 target과의 상관성이 거의 없으며, 결측치를 다른 변수를 이용해 대체하기 어려움

최종 모델에 사용된 변수

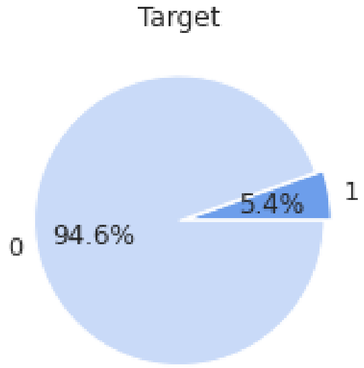
변수명	변수명
loan_limit	personal_rehabilitation
loan_rate	houseown_type
income	insert_hour
yearly_income	purpose
desired_amount	diff
credit_score	product_freq
existing_loan_cnt	company_enter
existing_loan_amt	product_cnt

03 모형 구축 | 모형 튜닝

LGBM(Light Gradient Boosting Model)

- 하이퍼 파라미터 튜닝

하이퍼 파라미터	설명
scale_pos_weight	이진분류 문제에서 positive label(1)에 가중치 부여
learning_rate	부스팅 스탭 반복할 때 학습률, 0~1사이의 값
num_iterations	반복 수행하는 트리 개수
max_depth	트리의 깊이
n_jobs	사용할 CPU 코어의 개수(-1: 모두 사용)
num_leaves	하나의 트리가 가질 수 있는 최대 리프 개수
boost_from_average	타겟 레이블 값이 불균형한 경우 성능이 떨어지는 것을 방지하기 위해 “False”로 설정
objective	이진분류이면 ‘binary’



- Target변수의 분포가 매우 **불균형**

```
LGBMClassifier(scale_pos_weight=3,
               learning_rate=0.05,
               num_iterations=1000,
               max_depth=24,
               n_jobs=-1,
               num_leaves=128,
               boost_from_average=False,
               objective='binary')
```

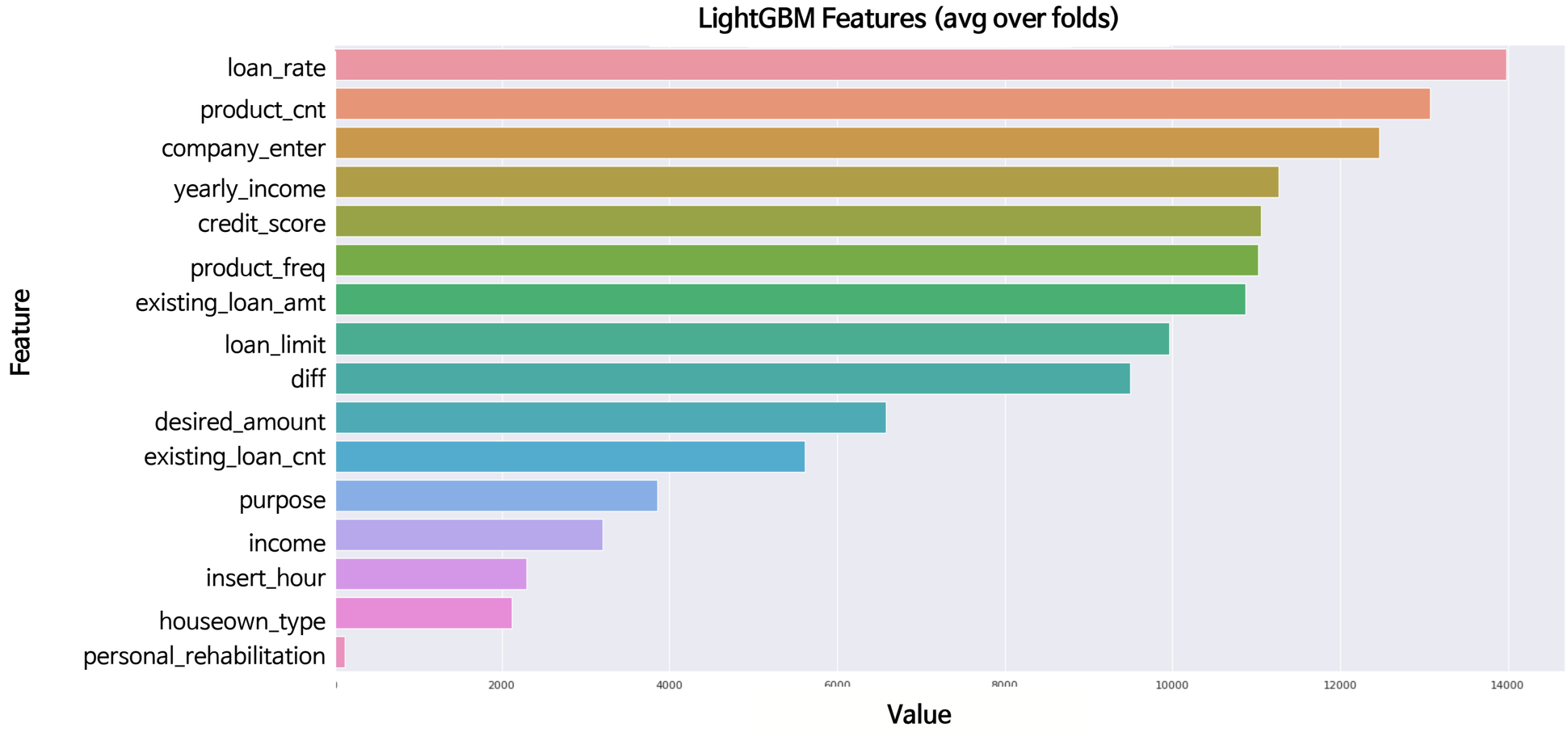
모델 성능 및 결과

Confusion matrix

	precision	recall	f1-score
0	0.97	0.96	0.96
1	0.41	0.50	0.45

Train f1 score : 0.471
valid f1 score : 0.451

변수 중요도


LightGBM Features (avg over folds)
Feature
loan_rate
product_cnt
company_enter
yearly_income
credit_score
product_freq
existing_loan_amt
loan_limit
diff
desired_amount
existing_loan_cnt
purpose
income
insert_hour
houseown_type
personal_rehabilitation
2000
4000
6000
8000
10000
12000
14000
Value

변수 중요도

score	feature
13983	loan_rate
13069	product_cnt
12463	company_enter
11265	yearly_income
11052	credit_score
11017	product_freq
10874	existing_loan_amt
9958	loan_limit
9491	diff

대출 상품 변수

대출 금리, 대출 한도,
상품 개수, 상품 인기도

유저 신용 변수

신용점수, 입사연월, 연소득,
기대출, 기대한도

결론

1. 대출 신청 여부를 예측하는데, “ **대출 금리, 대출 한도, 연소득, 입사 연월**” 등 유저의 신용 정보에 따라 보여지는 상품이 다르기 때문에 높은 변수 중요도를 보였다.

2. 유저별 보여지는 **대출 상품의 개수**를 나타내는 product_cnt 변수와 **상품별 인기도**를 나타내는 product_freq 변수 또한 유의미한 중요도를 보였다.

이를 기반으로 측정된 **상품별 인기도**를 통해 유저의 **신용 정보**와 종합하고

진입장벽이 낮은 상품을 적극적으로 추천하여 대출 신청 확률을 높이는 방안을 제안한다.

Contents

문제2

01 데이터 소개 및 분석 목적

데이터 소개 2022년 3~5월 의 가명화된 핀다 앱 사용 **로그 데이터**

분석 목적 핀다 홈화면 진입 고객 군집화 → 군집별 **서비스 메시지 제안**

user_id	event	timestamp	date_cd
1	GetCreditInfo	2022-05-03 14:52:28	2022-05-03
1	GetCreditInfo	2022-05-03 14:52:35	2022-05-03
1	UseLoanManage	2022-06-16 23:58:41	2022-06-16
1	Login	2022-06-16 23:58:41	2022-06-16
1	GetCreditInfo	2022-06-16 23:58:42	2022-06-16
7	GetCreditInfo	2022-05-22 16:39:49	2022-05-22
9	GetCreditInfo	2022-05-21 23:37:58	2022-05-21
9	GetCreditInfo	2022-05-21 23:43:33	2022-05-21

(17,843,993행 4열)

01 데이터 소개 및 분석 목적

테이블 소개 LOG DATA (핀다 앱 사용 로그)

Columns	변수 설명
user_id	유저 번호
event	행동명
timestamp	행동일시
date_cd	일 코드

event	event 설명
SignUp	회원가입
OpenApp	핀다 앱 실행
Login	핀다 앱 로그인
ViewLoanApplyIntro	한도조회 인트로 페이지 조회
StartLoanApply	한도조회 시작하기 버튼 클릭
CompleteIDCertification	본인 인증 완료
EndLoanApply	한도조회 결과 확인
UseLoanMange	대출 관리 서비스 이용
UsePrepayCalc	여윳돈 계산기 서비스 이용
UseDSRCalc	DSR 계산기 서비스 이용
GetCreditInfo	KCB 신용정보 조회

02 데이터 탐색과 전처리

테이블 탐색

user_id	event	timestamp	date_cd
1	GetCreditInfo	2022-05-03 14:52:28	2022-05-03
1	GetCreditInfo	2022-05-03 14:52:35	2022-05-03
1	UseLoanManage	2022-06-16 23:58:41	2022-06-16
1	Login	2022-06-16 23:58:41	2022-06-16
1	GetCreditInfo	2022-06-16 23:58:42	2022-06-16
7	GetCreditInfo	2022-05-22 16:39:49	2022-05-22
9	GetCreditInfo	2022-05-21 23:37:58	2022-05-21
9	GetCreditInfo	2022-05-21 23:43:33	2022-05-21

▲ user_id 별 핀다 앱 사용 log 날짜 순 정렬 예시

- **분석 목적** : USER SPEC + LOG DATA

  → user_id 별 특성 군집화

- USER SPEC + LOG DATA TYPE

  LOG DATA O, USER SPEC O

  LOG DATA O, USER SPEC X

  LOG DATA X, USER SPEC O : 로그 유실 데이터

  → **LOG 존재하는 USER 대상**으로 분석 진행

02 데이터 탐색과 전처리

LOG DATA 전처리 | app_time 변수 생성

- user_id 별 특성 군집화를 위해 여러 행으로 표현된 LOG DATA 정리 필요함
- timestamp 변수 간 차이 이용 → user_id 별 한도 조회(insert_time)까지 소요된 '**앱 사용 시간**' 변수 생성

user_id	timestamp	date_cd	application_id	insert_time	diff (초)
9	2022-05-21 23:37:58	2022-05-21	248576.0	2022-05-21 23:41:09	-5057039.0
11	2022-03-24 10:53:59	2022-03-24	1180616.0	2022-03-24 10:58:22	8.0
11	2022-03-24 10:54:07	2022-03-24	1180616.0	2022-03-24 10:58:22	1.0
11	2022-03-24 10:54:08	2022-03-24	1180616.0	2022-03-24 10:58:22	11.0
11	2022-03-24 10:54:19	2022-03-24	1180616.0	2022-03-24 10:58:22	17.0
11	2022-03-24 10:54:36	2022-03-24	1180616.0	2022-03-24 10:58:22	67.0
11	2022-03-24 10:55:43	2022-03-24	1180616.0	2022-03-24 10:58:22	43.0
11	2022-03-24 10:56:26	2022-03-24	1180616.0	2022-03-24 10:58:22	2329664.0
11	2022-04-20 10:04:10	2022-04-20	1751522.0	2022-03-24 10:58:22	6.0

02 데이터 탐색과 전처리

LOG DATA 전처리 | app_time 변수 생성

- timestamp 차이 이용 시 문제점 [문제1]
  - application_id가 바뀌는 경우, 윗 행에서 비정상적으로 큰 양수 or 음수 나타남

user_id	timestamp	date_cd	application_id	insert_time	diff (초)
9	2022-05-21 23:37:58	2022-05-21	248576.0	2022-05-21 23:41:09	-5057039.0
11	2022-03-24 10:53:59	2022-03-24	1180616.0	2022-03-24 10:58:22	8.0
11	2022-03-24 10:54:07	2022-03-24	1180616.0	2022-03-24 10:58:22	1.0
11	2022-03-24 10:54:08	2022-03-24	1180616.0	2022-03-24 10:58:22	11.0
11	2022-03-24 10:54:19	2022-03-24	1180616.0	2022-03-24 10:58:22	17.0
11	2022-03-24 10:54:36	2022-03-24	1180616.0	2022-03-24 10:58:22	67.0
11	2022-03-24 10:55:43	2022-03-24	1180616.0	2022-03-24 10:58:22	43.0
11	2022-03-24 10:56:26	2022-03-24	1180616.0	2022-03-24 10:58:22	2329664.0
11	2022-04-20 10:04:10	2022-04-20	1751522.0	2022-03-24 10:58:22	6.0

application_id 경계값에서 발생하는 **이상치**, 어떻게 처리할 것인가?

02 데이터 탐색과 전처리

LOG DATA 전처리 | app_time 변수 생성

1) **음수**인 경우 : [insert_time - timestamp] 값으로 대체

user_id	timestamp	date_cd	application_id	insert_time	diff (초)	diff_대체
9	2022-05-21 23:37:58	2022-05-21	248576.0	2022-05-21 23:41:09	-5057039.0	191.0

2) **양수**인 경우

- 1차 시도 : diff ≥ 86,400 (1일) → [insert_time - timestamp] 값으로 대체
- but, application_id 바뀌었음에도 처리되지 않는 행 발생 (ex. index 76)
- 2차 시도 : diff ≥ 74,107 (index 76) → [insert_time - timestamp] 값으로 대체

→ application_id 경계값 app_time 이상치 해결 완료!

02 데이터 탐색과 전처리

LOG DATA 전처리 | app_time 변수 생성

- [문제2] 앱 종료 X → 백그라운드에서 앱 사용이 지속되는 경우, 앱 사용 시간 **이상치**로 나타남

diff 몇 초 이상일 때 user가 활동을 하지 않는다고 판단할 것인가?

user_id	timestamp	date_cd	application_id	insert_time	diff (초)	diff_대체
17	2022-03-07 07:20:49	2022-03-07	2125512.0	2022-03-07 13:31:09	21.0	21.0
17	2022-03-07 07:21:10	2022-03-07	2125512.0	2022-03-07 13:31:09	22093.0	0
17	2022-03-07 13:29:23	2022-03-07	2125512.0	2022-03-07 13:31:09	5.0	5.0

- CompleteIDCertification → EndLoanApply 즉, 대출 상품 조회 시간 40분인 user 존재
- diff ≥ 3,600 (1시간) → 활동 안했다고 가정

→ 백그라운드 앱 실행으로 인한 app_time 이상치 해결 완료!

02 데이터 탐색과 전처리

LOG DATA 전처리 | app_time 변수 생성

- diff 이상치 처리 완료 후, application_id 별 diff 변수 sum → **app_time 변수** 생성
- 17,843,993 → 645,748 행 개수 축소 (application_id 별 1개 행 존재)

- app_time 이상치 처리
  - app_time ≥ 99 percentile 인 6,461행 99 percentile 값으로 대체

- 이후 USER SPEC + LOG DATA
  - KEY : ‘ user_id + application_id’

count	645748
mean	439.0380
std	1067.6093
min	0
25%	91
50%	145
75%	287
99%	5289
max	32831

03 파생 변수 생성

USER SPEC 관련 파생 변수 생성

application_count

user 별 application_id 총 개수
(총 대출 신청 조회 수)

app_time

앱 사용 총 시간
(application_id 별 앱 사용 시간 합)

apply_count

대출 결정 상품 개수
(is_applied = 1)

user_id	application_count	app_time	apply_count
1	0	0	0
7	0	0	0
9	26	4966.0	0
11	50	16900.0	3
12	0	0	0

(560,959 행)

- 3개 변수 모두 0으로 채워진 경우
  : LOG DATA O, USER SPEC X

03 파생 변수 생성

LOG DATA event 관련 파생 변수 생성

idc_loan_count

CompleteIDCertification
→ EndLoanApply

본인 인증 후, 대출 상품 조회
이뤄진 횟수

creditinfo_cnt

GetCreditInfo 횟수

KCB 신용 정보 조회 횟수
높을수록 적극적인 회원

useloanmanage_cnt

UseLoanMange 횟수

대출 관리 서비스 이용 횟수
높을수록 적극적인 회원

user_id	application_count	app_time	apply_count	idc_loan_count	creditinfo_cnt	useloanmanage_cnt
1	0	0	0	0	2	1
7	0	0	0	0	1	0
9	26	4966.0	0	0	1	0
11	50	16900.0	3	4	2	2
12	0	0	0	14	4	1

03 파생 변수 생성

이상치 및 결측치 처리

user_id	application_count	app_time	apply_count	idc_loan_count	creditinfo_cnt	useloanmanage_cnt
1	0	0	0	0	2	1
7	0	0	0	0	1	0
9	26	4966.0	0	0	1	0
11	50	16900.0	3	4	2	2
12	0	0	0	14	4	1

- 각 변수에 대해 2 x 99 percentile 보다 큰 값 → 2 x 99 percentile 값 대체
- application_count / app_time / apply_count **모두 0**인 경우?
  - LOG DATA O, USER SPEC X
  - 대출에 관심은 있지만, 대출 조회는 하지 않은 경우 → application_id 생성 X
  - 대출 조회를 한 고객 vs. 대출 조회 하지 않은 고객 분리 필요

03 파생 변수 생성

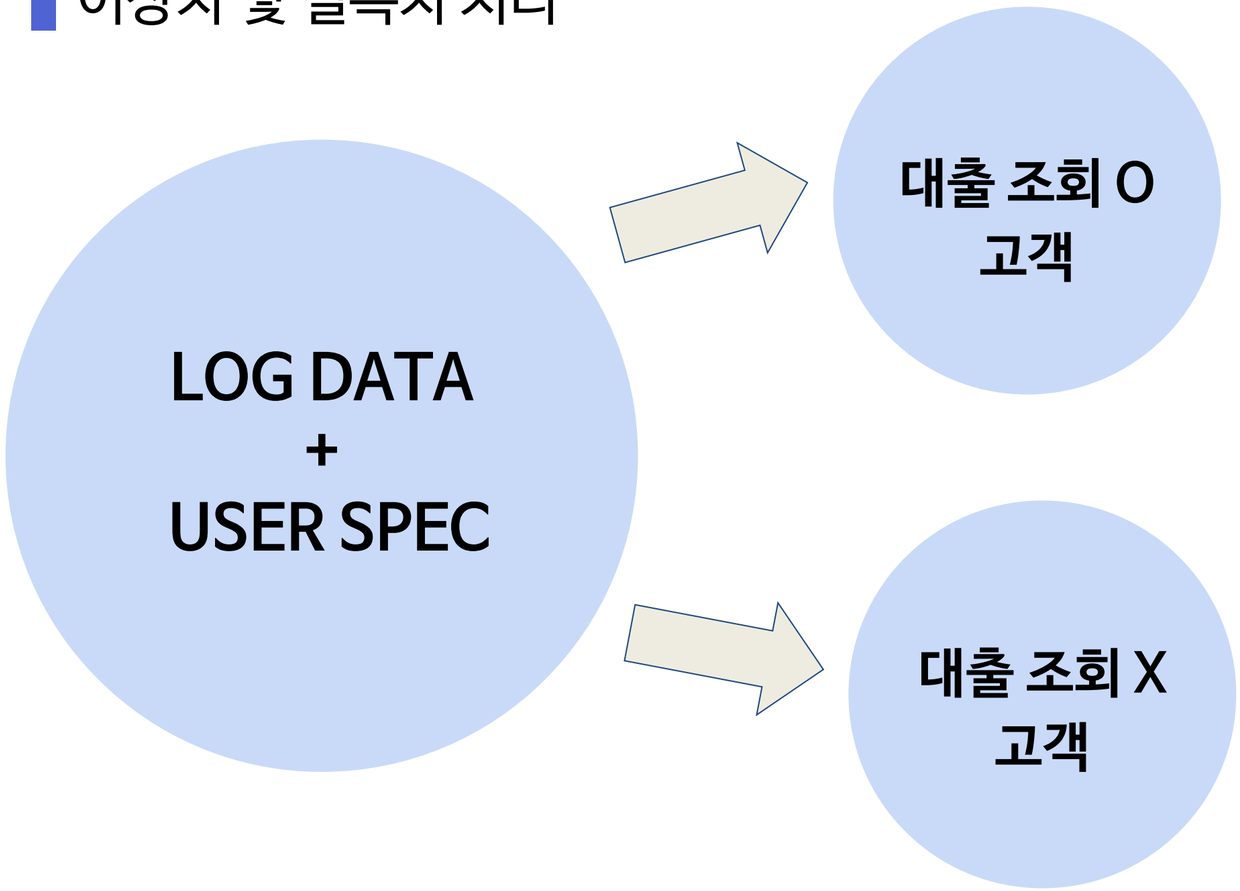
이상치 및 결측치 처리

user_id	application_count	app_time	apply_count	idc_loan_count	creditinfo_cnt	useloanmanage_cnt
1	-70	0	-22	0	2	1
7	-97	0	-22	0	1	0
9	26	4966.0	0	0	1	0
11	50	16900.0	3	4	2	2
12	-66	0	-22	14	4	1

- application_count / app_time / apply_count 모두 0인 경우?
- 대출 조회를 한 고객 vs. 대출 조회 하지 않은 고객 분리 위해
  - application_count : 이상값(-1~-100 랜덤) 대체
  - app_time : 0으로 채우기
  - apply_count : 이상값(-22; apply_count 최댓값 = 22) 대체

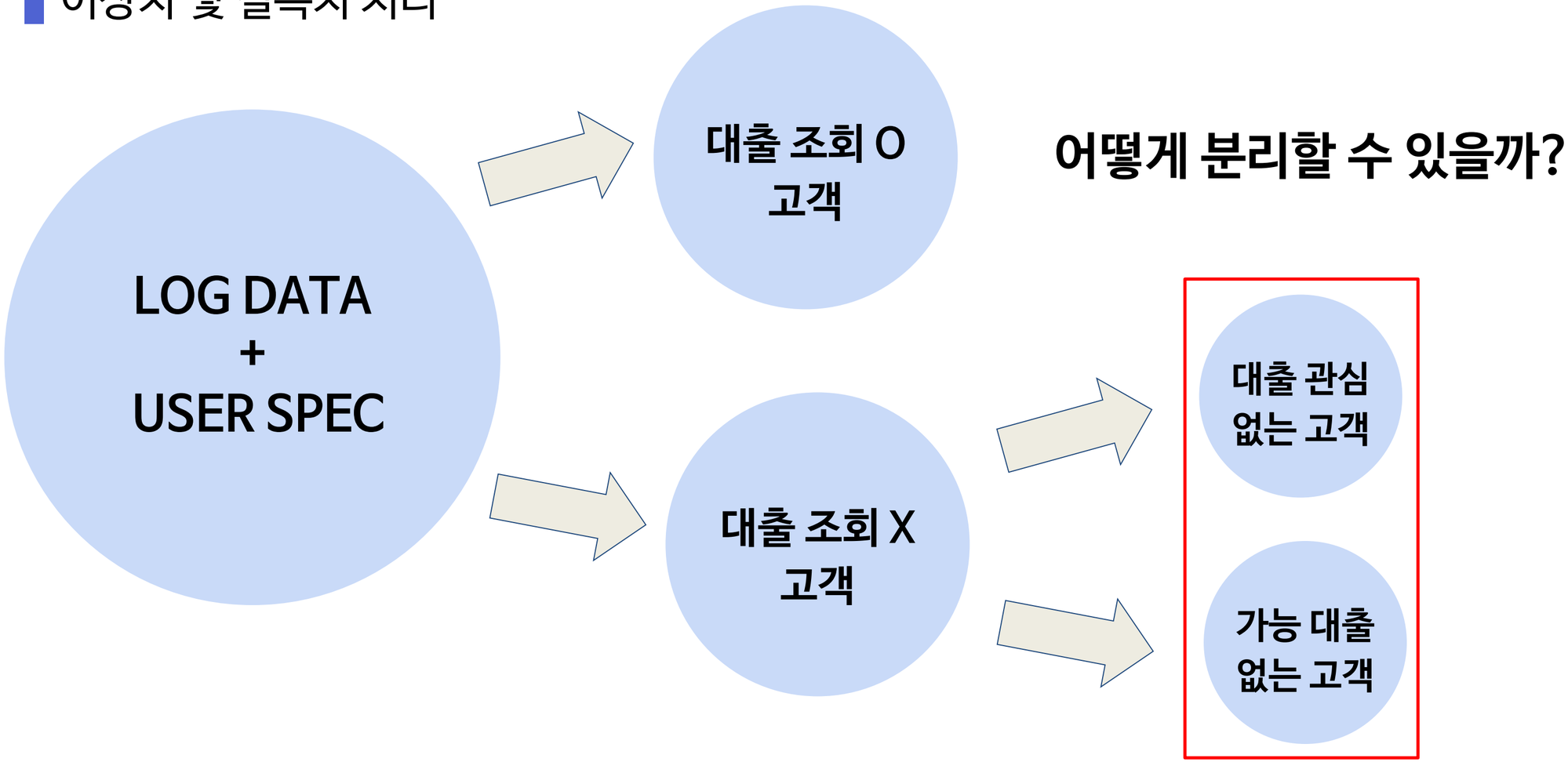
03 파생 변수 생성

이상치 및 결측치 처리



03 파생 변수 생성

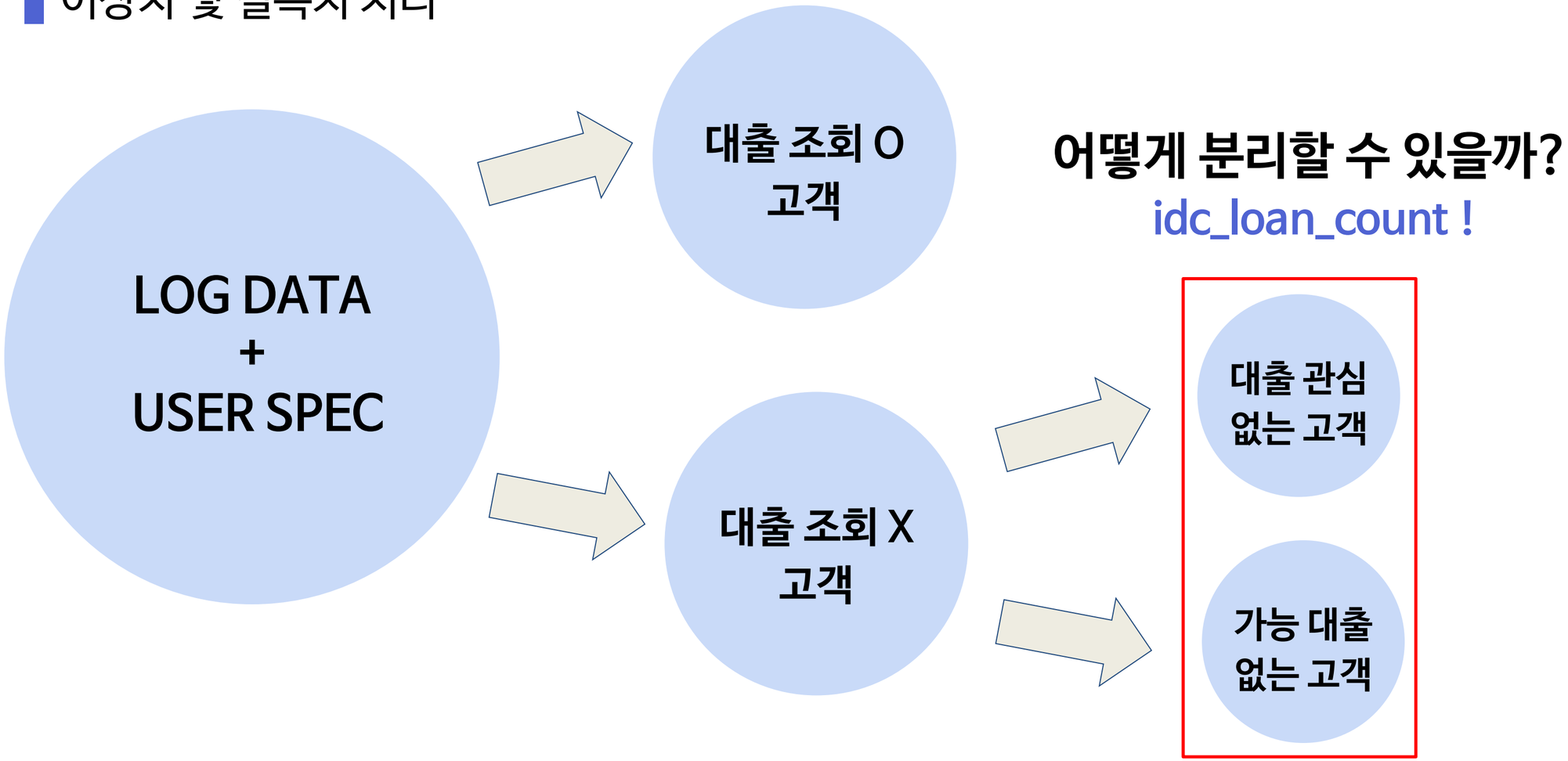
이상치 및 결측치 처리



어떻게 분리할 수 있을까?

03 파생 변수 생성

이상치 및 결측치 처리

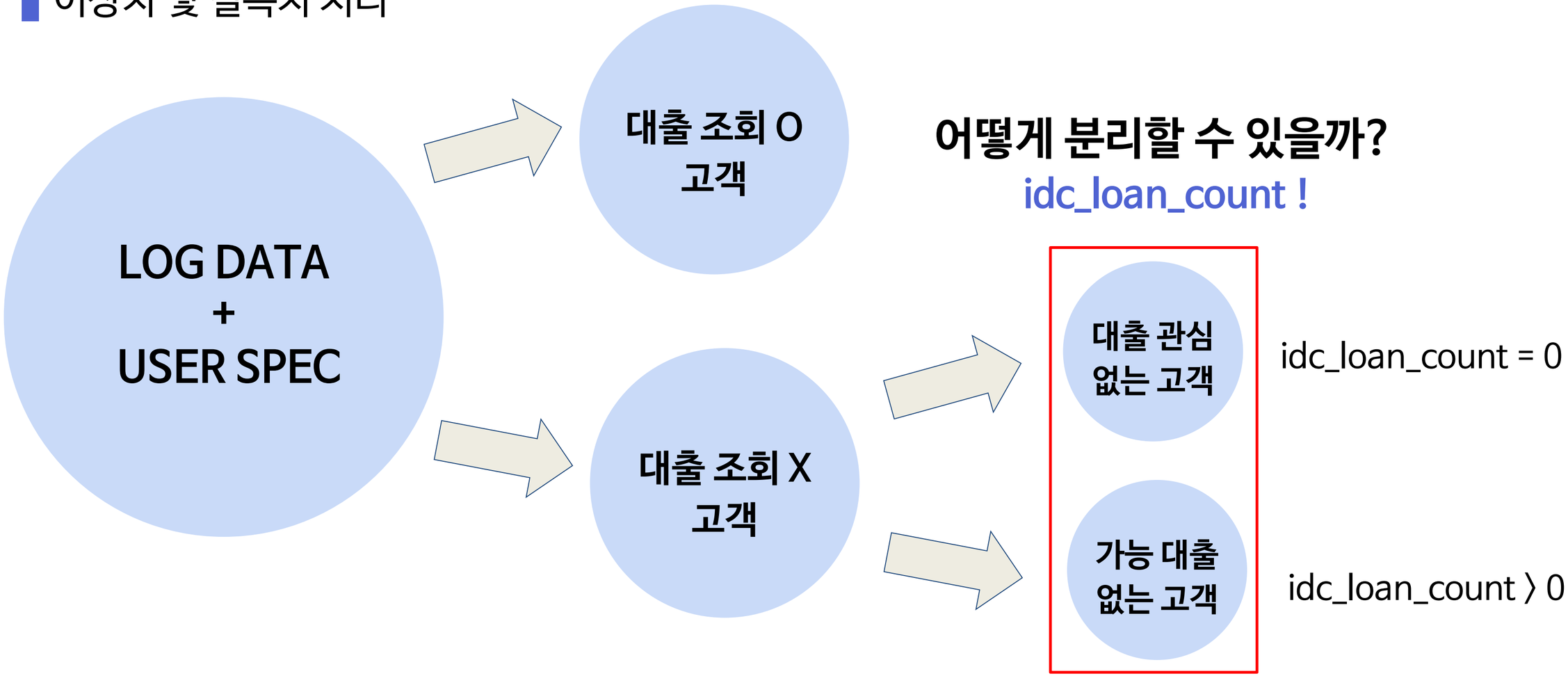


어떻게 분리할 수 있을까?

idc_loan_count !

03 파생 변수 생성

이상치 및 결측치 처리



03 파생 변수 생성

이상치 및 결측치 처리

idc_loan_count

- LOG DATA event 파생 변수로, 중요하다고 판단 → **가중치** 부여
- 대출 조회 한 고객 vs. 대출 조회하지 않은 고객 분리 위해
  - idc_loan_count = 0 인 경우
    - 대출 조회 고객 idc_loan_count = 0
    - 대출 조회 하지 않은 고객 idc_loan_count = -1

user_id	application_count	app_time	apply_count	idc_loan_count	creditinfo_cnt	useloanmanage_cnt
1	-70	0	-22	-1	2	1
7	-97	0	-22	-1	1	0
9	26	4966.0	0	0	1	0
11	50	16900.0	3	40	2	2
12	-66	0	-22	140	4	1

04 K-means Clustering

Min-Max Scaling

$$x_{scaled} = \frac{x - x_{min}}{x_{max} - x_{min}}$$

user_id	application_count	app_time	apply_count	idc_loan_count	creditinfo_cnt	useloanmanage_cnt
1	0.0582	0.0000	0.0000	0.0000	0.0455	0.0294
7	0.0075	0.0000	0.0000	0.0000	0.0227	0.0000
9	0.2383	0.0244	0.5000	0.0028	0.0227	0.0000
11	0.2833	0.0830	0.5682	0.1136	0.0455	0.0588
12	0.0657	0.0000	0.0000	0.3906	0.0909	0.0294

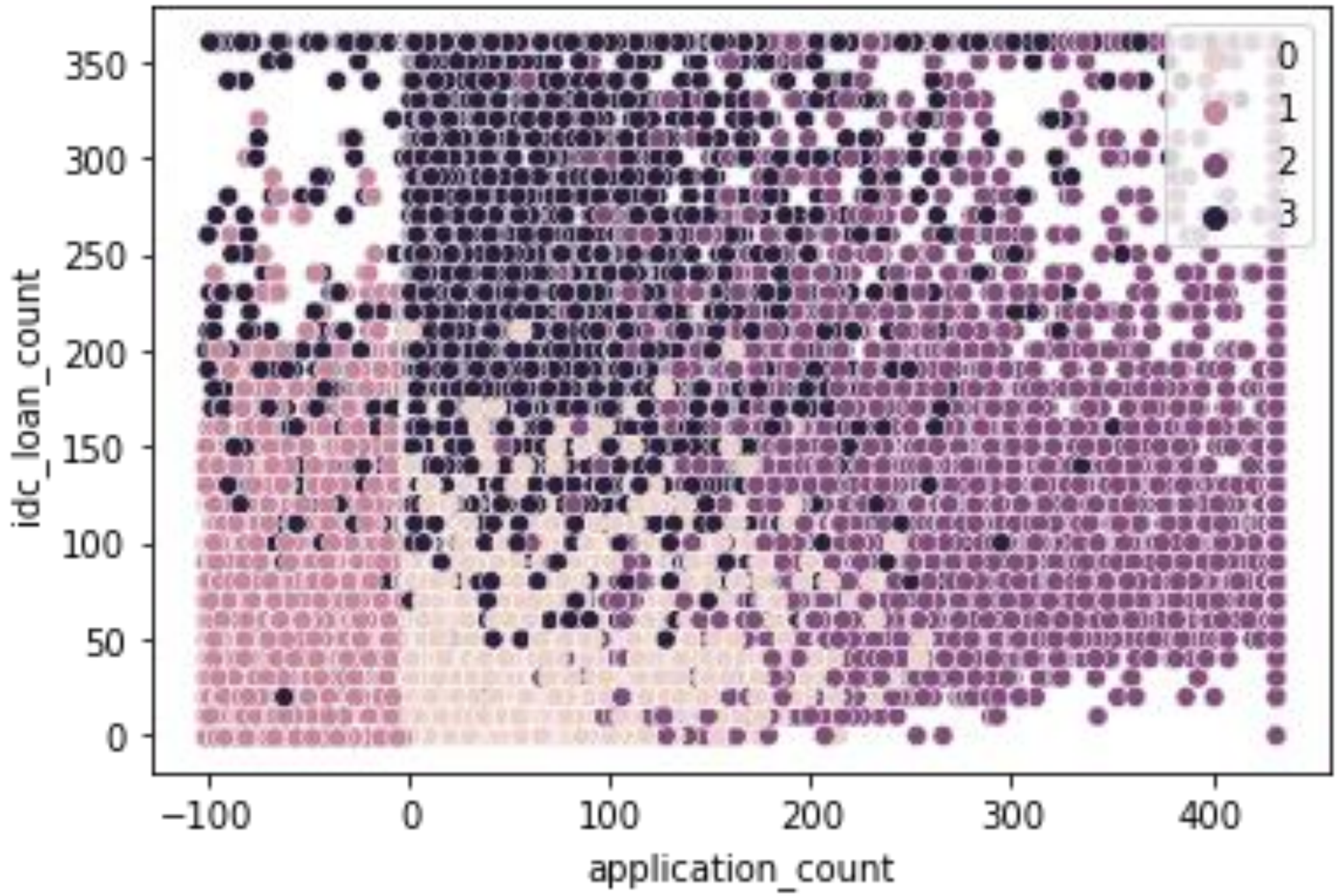
04 K-means Clustering

Cluster 확인

user_id	application_count	app_time	apply_count	idc_loan_count	creditinfo_cnt	useloanmanage_cnt	cluster
1	-70	0	-22	-1	2	1	1
7	-97	0	-22	-1	1	0	1
9	26	4966.0	0	0	1	0	0
11	50	16900.0	3	40	2	2	0
12	-66	0	-22	140	4	1	1

05 분석 결과 해석

Cluster 시각화



05 분석 결과 해석

Cluster 별 변수 mean값

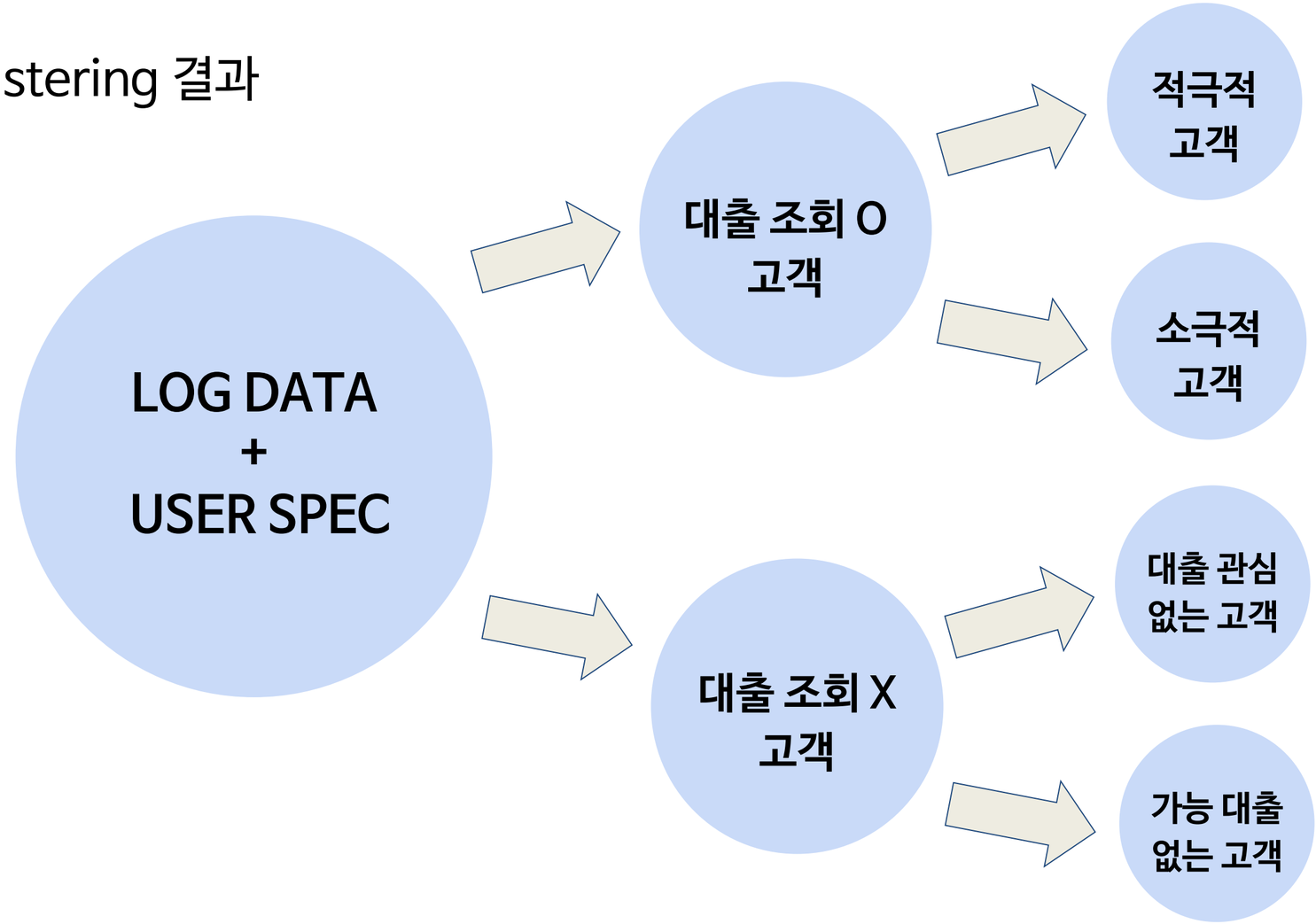
cluster	application_count	app_time	apply_count	idc_loan_count	creditinfo_cnt	useloanmanage_cnt
0	29.0841	8082.5740	1.7277	24.3455	2.5247	1.7117
1	-51.4646	0.0000	-22.0000	3.6982	1.5412	0.8671
2	215.2526	111135.1302	6.0357	90.6085	7.4004	5.4607
3	39.0984	13984.6070	3.1467	118.4409	17.4944	13.6670

- 대출 조회 **가능** 고객 Cluster 0, Cluster 2 ⇒ 모든 변수에 대해 Cluster 0 < Cluster 2 (활동성 ↑)
  - Cluster 0 : 소극적 고객
  - Cluster 2 : 적극적 고객

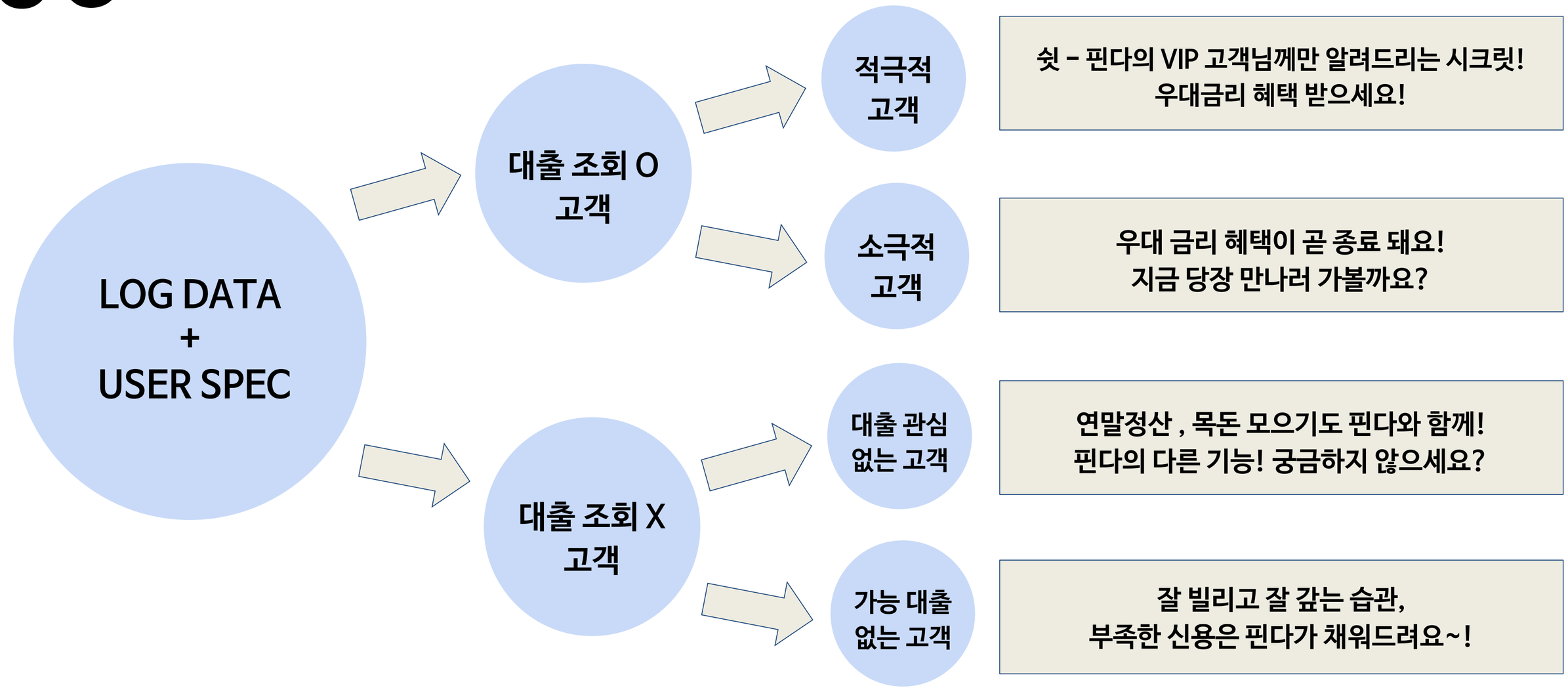
- 대출 조회 **불가능** 고객 Cluster 1, Cluster 3 ⇒ 모든 변수에 대해 Cluster 1 < Cluster 3 (활동성 ↑)
  - Cluster 1 : 대출 관심 없는 고객
  - Cluster 3 : 대출 관심 있는 고객

05 분석 결과 해석

고객 Clustering 결과



06 서비스 메시지 제안


LOG DATA
+
USER SPEC
대출 조회 O
고객
대출 조회 X
고객
적극적
고객
소극적
고객
대출 관심
없는 고객
가능 대출
없는 고객
쉿 - 핀다의 VIP 고객님께만 알려드리는 시크릿!
우대금리 혜택 받으세요!
우대 금리 혜택이 곧 종료 돼요!
지금 당장 만나러 가볼까요?
연말정산 , 목돈 모으기도 핀다와 함께!
핀다의 다른 기능! 궁금하지 않으세요?
잘 빌리고 잘 갚는 습관,
부족한 신용은 핀다가 채워드려요~!

감사합니다 :)