

제10회 2022 빅콘테스트 데이터 분석 퓨처스리그

앱 사용성 데이터를 통한 대출 신청 예측 모델링 및 군집분석

웃음꽃핀다

팀장 임승섭 (y7511204@naver.com)

팀원 이정우 (gksekdrn@naver.com)

조은아 (eunalunacho@gmail.com)

신은빈 (dnflrhra@naver.com)

황찬웅 (hcw1027@gmail.com)

CONTENTS

01. Introduction

- 분석 배경
- 분석 목표

02. Data exploration

- 데이터 병합
- train 및 test 데이터 분리
- 데이터 분포 확인
- 상관관계

03. Preprocessing

- 불필요한 변수 제거
- 파생변수 생성
- 결측치 처리
- 이상치 처리
- 인코딩
- 스케일링

04. Modeling

- 모델 소개
- 모델 평가
- 모델 구축
- 해석

05. Clustering

- 추가 전처리
- PCA
- K-means
- 해석

06. Conclusion

- 결론
- 신용 점수 관리 서비스 제안
- 군집 별 메시지 제안

01

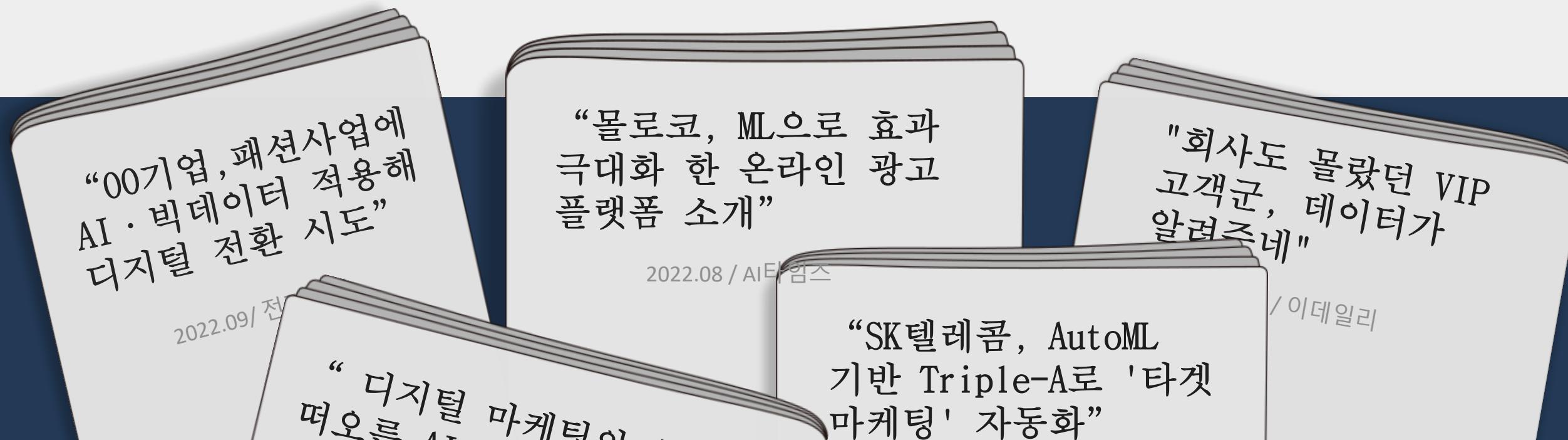
Introduction

- 분석 배경
- 분석 목표

분석 배경

- 대출 가입자가 늘어남과 동시에 대출 절차 간소화, 대출 시장 확장 및 금리 상승 등 시장의 변화가 일어나면서 개인의 피해를 줄이기 위해서는 상품 정보에 대한 접근성, 선택권이 보장되어야 한다는 공감대가 형성되고 있음
- 금융업계에서는 빅데이터를 활용해 고객 및 시장 이해, 서비스 프로세스 개선 등 향상된 의사결정에 드는 자원을 절약해 빠르게 목표 달성을 중
- 사용자들이 최적의 상품을 찾도록 돋는 핀다는 이런 대출 시장의 변화를 반영하고 있는 금융 플랫폼 모델링과 군집화를 통해 보다 더 효율적인 고객 관리와 마케팅 활동을 진행 가능

→ 이러한 흐름에 따라 우리는 **대출 신청 여부 예측 모델링**과 **어플 사용자 군집화**를 통해 대출 신청 군집에게 중요하게 작용하는 특성을 파악해 서비스를 제안하고, 어플 사용자 중 잠재 고객들에게 더 나은 서비스 메시지를 제안하고자 함



분석 목표

머신러닝 & 마케팅 자동화를 통한
효율적인 시간 자원 활용과 전략적 마케팅 관리에 투자

모델링을 통해 얻은 **SHAP Value 해석**을 통해 핀다의 프로모션 방향을 구체화

대출서비스를 이용한 고객과 이용하지 않은 고객으로 나누고, 군집화를 통해
군집을 나눈 후 각 군집 별 특성을 도출해내어 군집 별 메시지 제안

02

Data Exploration

-
- 데이터 병합
 - train 및 test 데이터 분리
 - 데이터 분포 확인
 - 상관관계

02 Data exploration

데이터 병합

user_spec

application_id	user_id	birth_year	gender	insert_time	...	credit_score
863239	605761	1968	0	2022-05-26 23:27:36		640.0
1762559	700573	1978	1	2022-05-26 21:31:23		650.0
1515483	504552	1993	1	2022-05-26 17:23:11		620.0
683930	571531	1988	1	2022-04-23 08:32:21		670.0

loan_result

application_id	loanapply_insert_time	bank_id	product_id	loan_limit	loan_rate	is_applied
1762559	2022-05-26 21:31:25	32	56	30000000.0	16.5	0
1762559	2022-05-26 21:31:25	23	236	19000000.0	12.5	1
1762559	2022-05-26 21:31:25	33	110	10000000.0	17.9	0
1762559	2022-05-26 21:31:31	40	129	20000000.0	16.0	0

user_spec에는 application_id가 '1762559'인 행이 1개

loan_result에는 application_id가 '1762559'인 행이 bank_id 개수만큼 존재



신청서 번호	유저 번호	유저 생년월일	유저 성별	생성일시	한도조회 일시	금융사 번호	상품 번호	승인한도	승인금리	신청 여부 (타겟)	
application_id	user_id	birth_year	gender	insert_time	...	loanapply_insert_time	bank_id	product_id	loan_limit	loan_rate	is_applied
1762559	700573	1978	1	2022-05-26 21:31:23	2022-05-26 21:31:25	32	56	30000000.0	16.5	0	
1762559	700573	1978	1	2022-05-26 21:31:23	2022-05-26 21:31:25	23	236	19000000.0	12.5	1	
1762559	700573	1978	1	2022-05-26 21:31:23	2022-05-26 21:31:31	40	129	20000000.0	16.0	0	

02 Data exploration

데이터 병합

- 데이터 설명

공통된 application_id를 가지고 있는 user_spec과 loan_result의 데이터들을 하나의 dataframe으로 합침

→ target data인 is_applied 칼럼이 loan_result에만 있으므로 loan_result를 기준으로 병합

이때 데이터는 각각의 application_id가 신청 가능한 bank_id 개수만큼 존재

user_spec					loan_result						
신청서 번호	유저 번호	유저 생년월일	유저 성별	생성일시	한도조회 일시	금융사 번호	상품 번호	승인한도	승인금리	신청 여부 (타겟)	
application_id	user_id	birth_year	gender	insert_time	loanapply_insert_time	bank_id	product_id	loan_limit	loan_rate	is_applied	
1945260	836762	1979.00000	1.00000	2022-04-20 01:23:09	2022-04-20 01:23:10	42	216	3000000.00000	14.50000	0.00000	
1945260	836762	1979.00000	1.00000	2022-04-20 01:23:09	2022-04-20 01:23:09	13	123	1000000.00000	19.90000	0.00000	
1019382	186886	1979.00000	1.00000	2022-04-20 00:38:18	2022-04-20 00:38:20	15	141	11000000.00000	15.10000	0.00000	
1019382	186886	1979.00000	1.00000	2022-04-20 00:38:18	2022-04-20 00:38:19	1	61	15000000.00000	9.90000	0.00000	
1019382	186886	1979.00000	1.00000	2022-04-20 00:38:18	2022-04-20 00:38:20	49	39	3000000.00000	15.90000	0.00000	

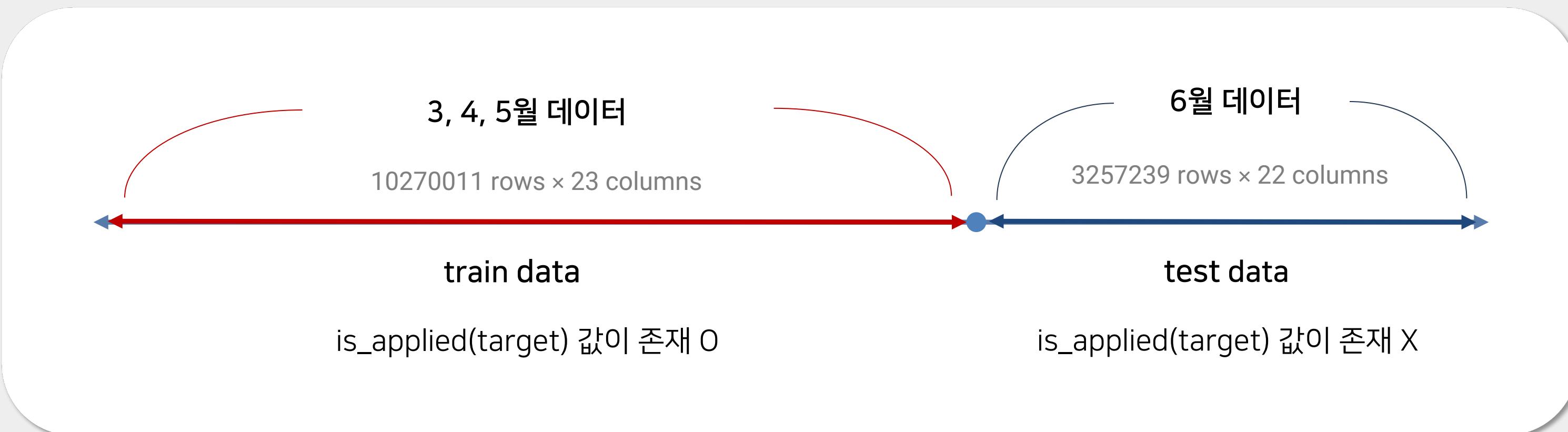
02 Data exploration

train,test 데이터 분리

- 병합한 데이터를 가지고 *train, test* 분리 후, 새로운 csv 생성

2022년 3,4,5월 데이터를 train data로 분리 (bigcon_train_origin.csv)

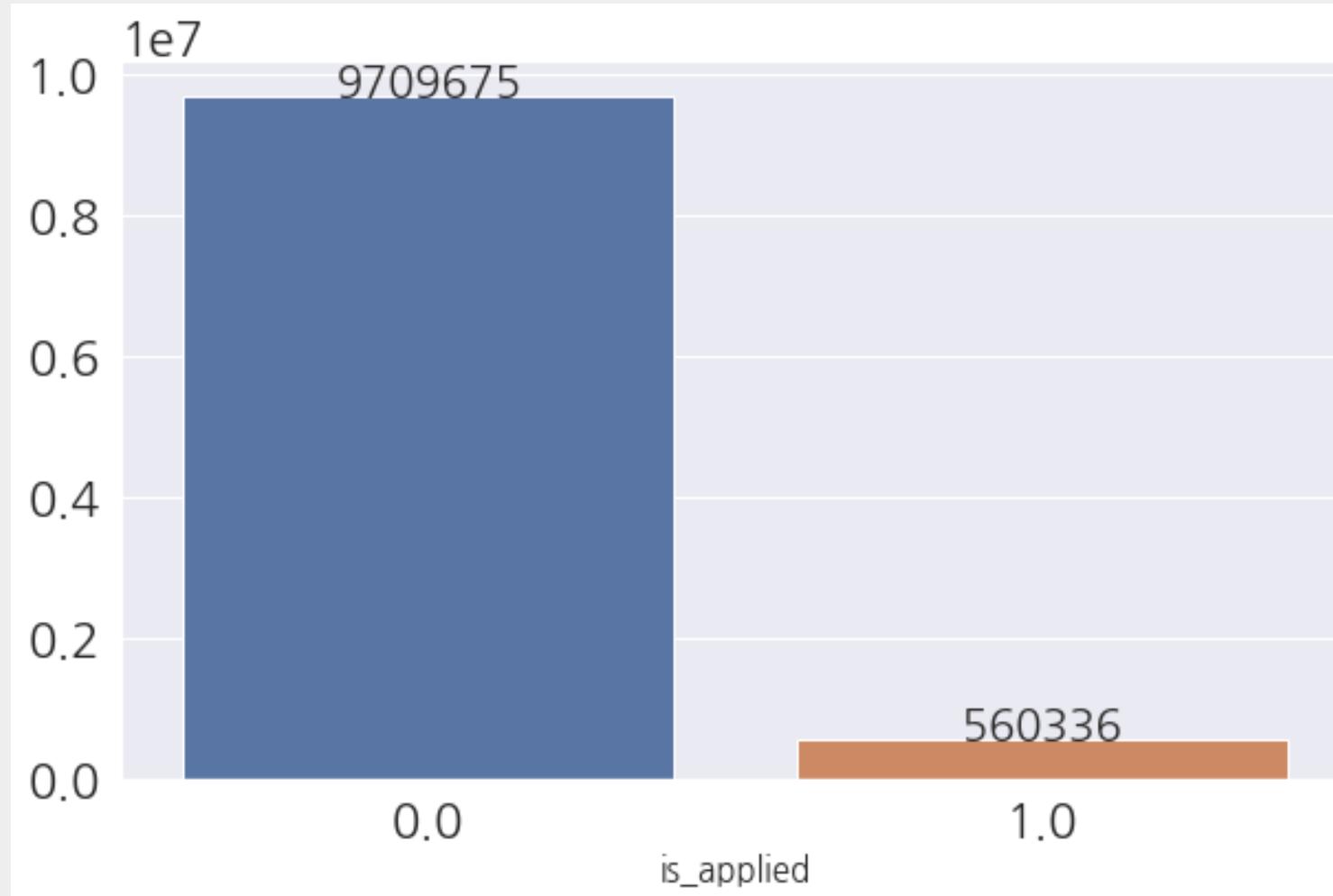
2022년 6월 데이터(is_applied가 결측치)를 test 데이터로 분리 (bigcon_test_origin.csv)



02 Data exploration

데이터 분포 확인

- *target 데이터 분포 확인*



전체 승인된 대출 상품 중 신청한 상품 수(1) : 9,709,675명

전체 승인된 대출 상품 중 신청하지 않은 상품 수(0) : 560,336명

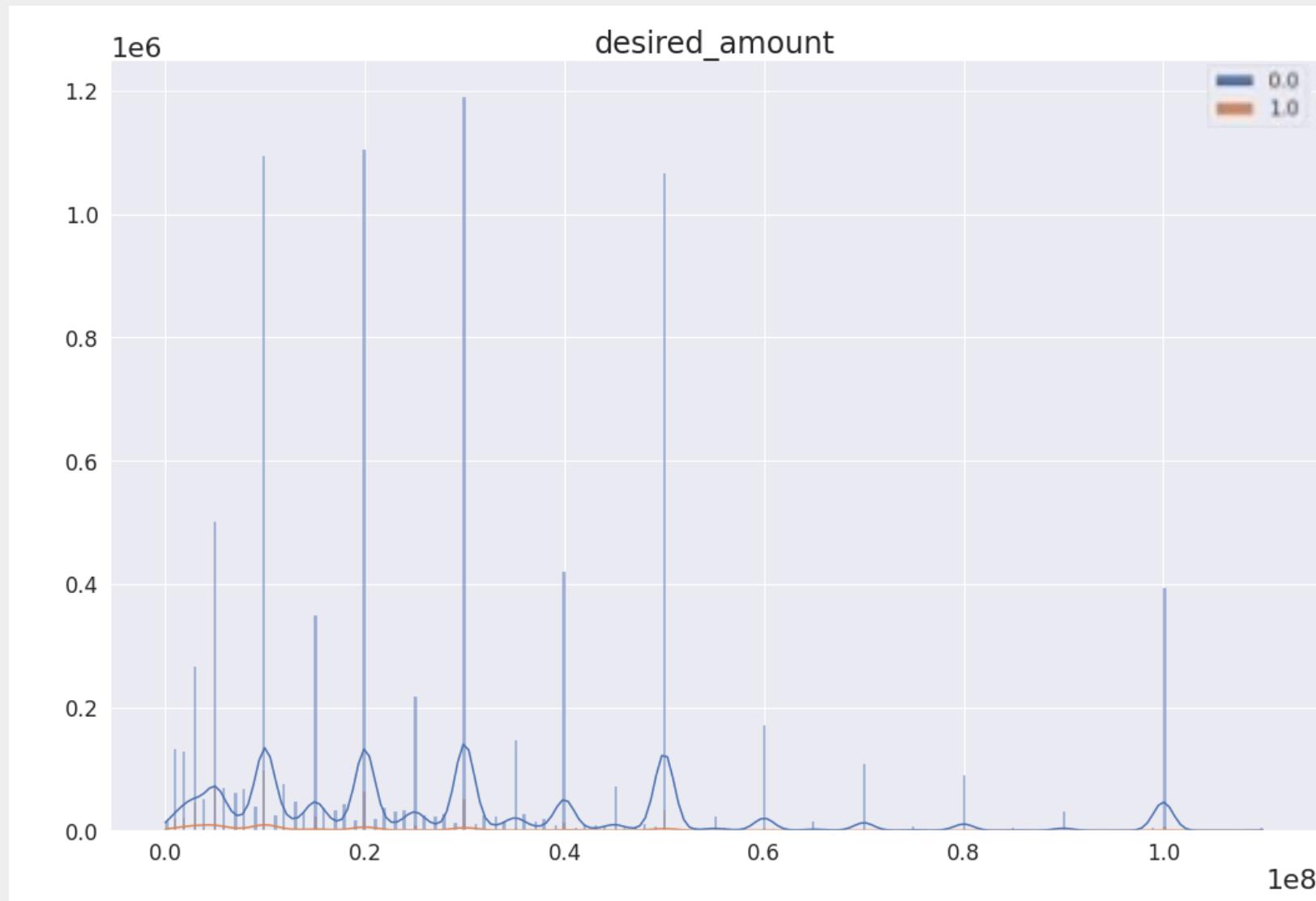
→ target 데이터가 약 19:1로 불균형한 분포를 보임
추후 모델링 시, 불균형 데이터 처리 필요

02 Data exploration

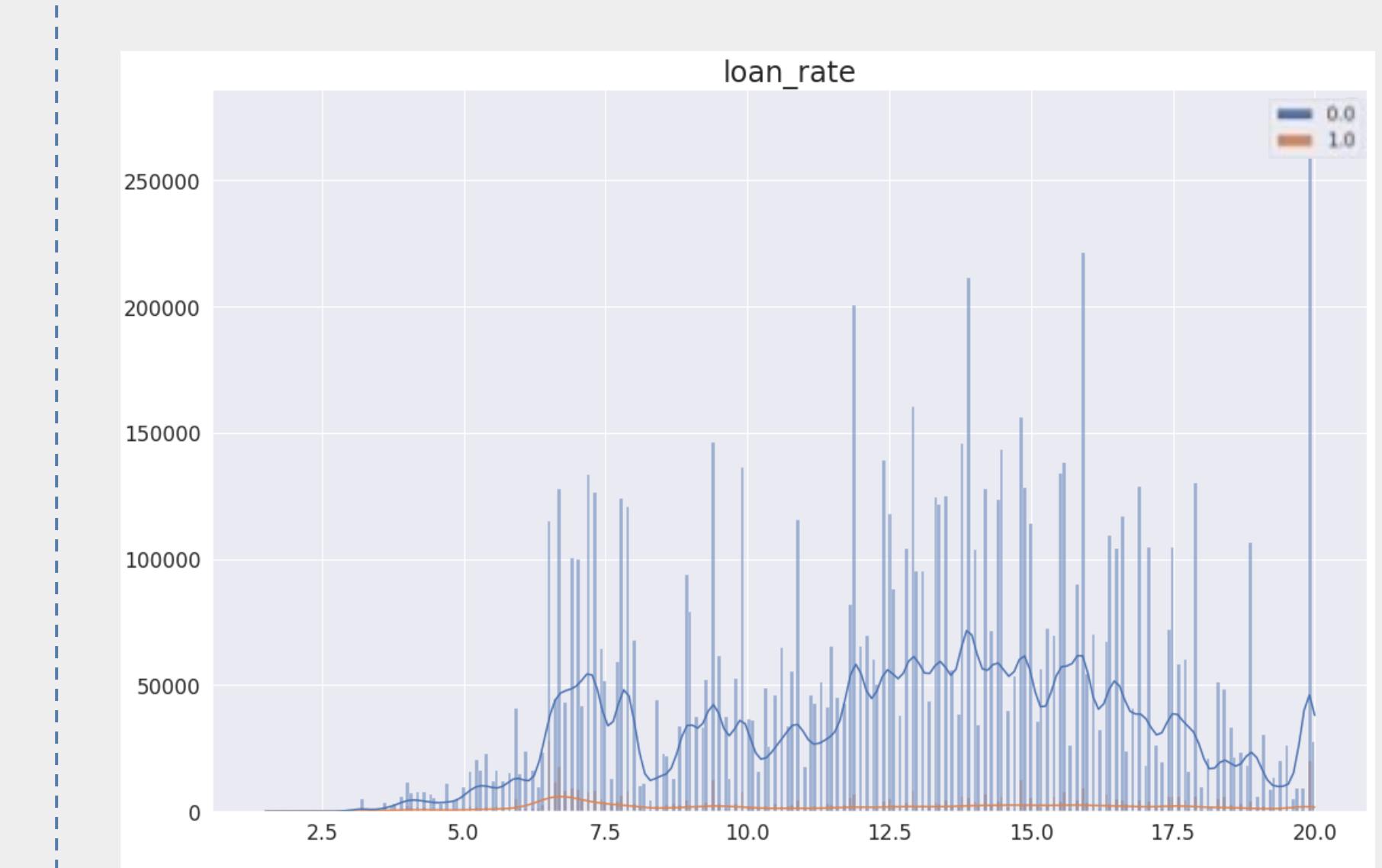
데이터 분포 확인

- 수치형 데이터

수치형 Feature당 is_applied 여부를 시각화



✓ Desired_amount : 1,000~5,000만원 구간이 많은 것을 확인할 수 있음

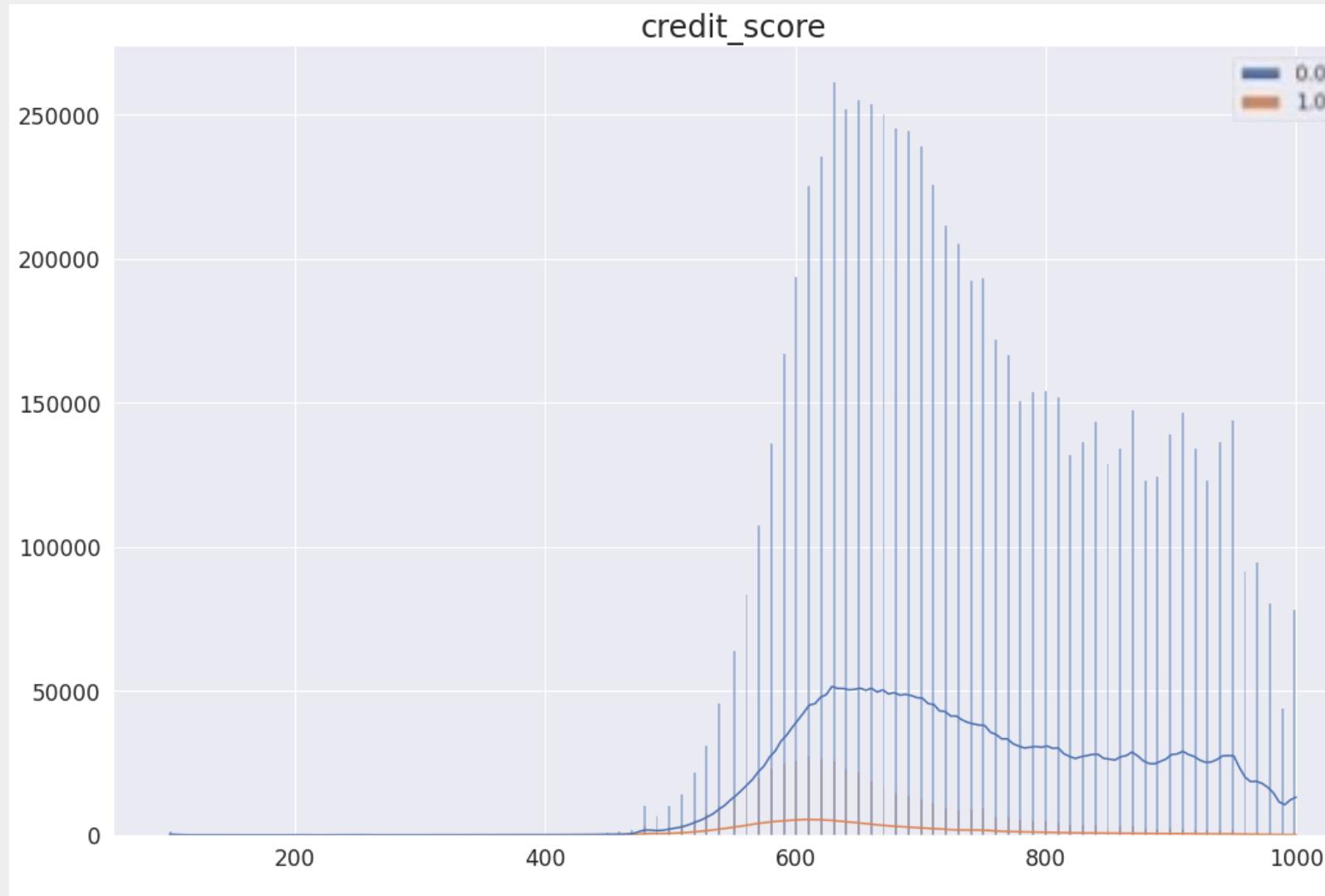


✓ loan_rate : 특정 금리 구간(6~7)에서 신청률이 높은 모습을 보임

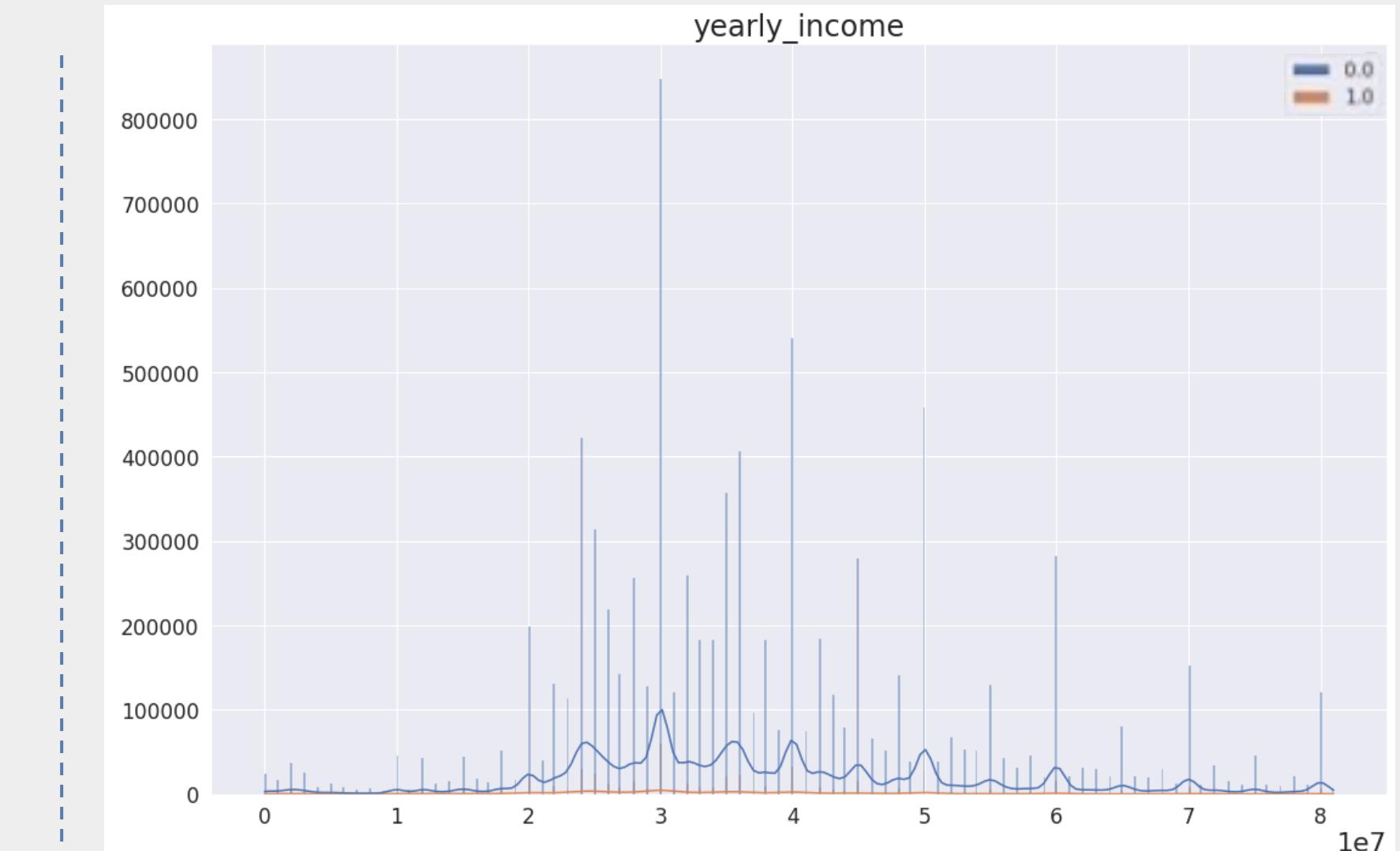
02 Data exploration

데이터 분포 확인

- 수치형 데이터



✓ credit_score : 600~700점 구간의 고객이 가장 많고,
신청률이 다른 구간에 비해 조금 높은 모습을 보임



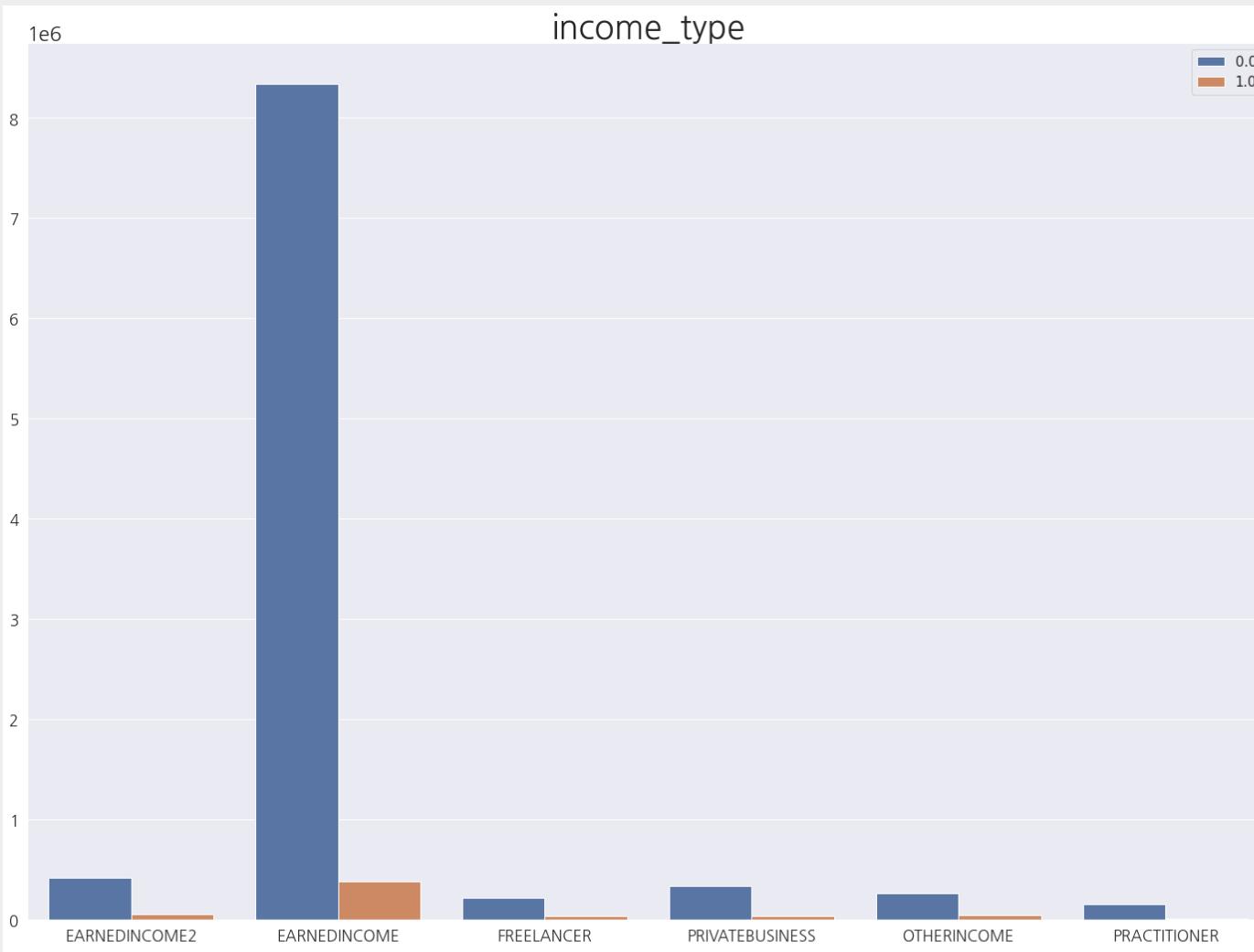
✓ yearly_income : 평균 연봉 2,500~5,000만원 구간의 고객이
많은 것을 확인할 수 있음

02 Data exploration

데이터 분포 확인

- 범주형 데이터

범주형 Feature당 is_applied 여부를 시각화



✓ income_type : EARNEDINCOME 타입의 고객이 눈에
띄게 많음을 확인할 수 있음

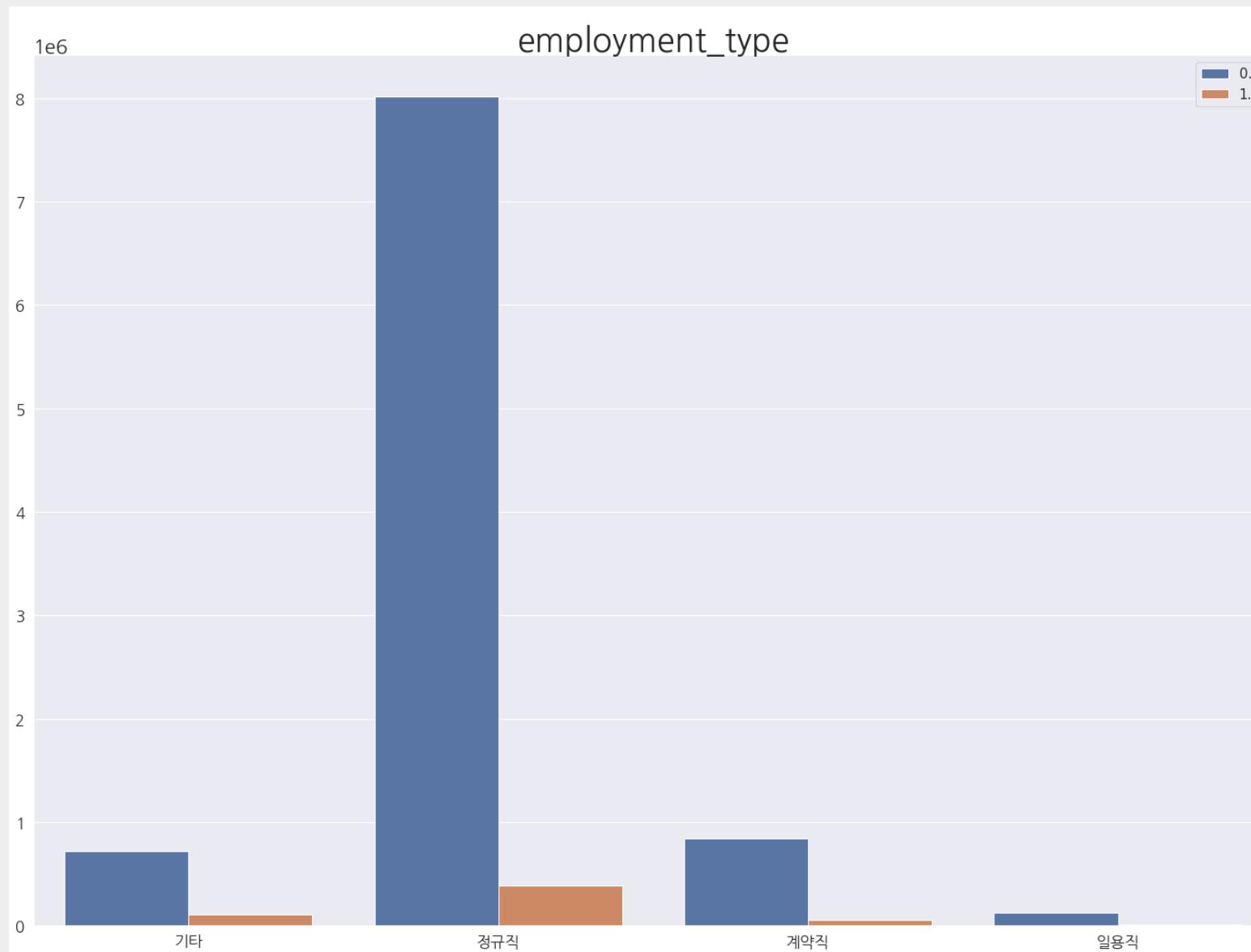


✓ product_id : 특정 상품(150 등)의 비율이 높음

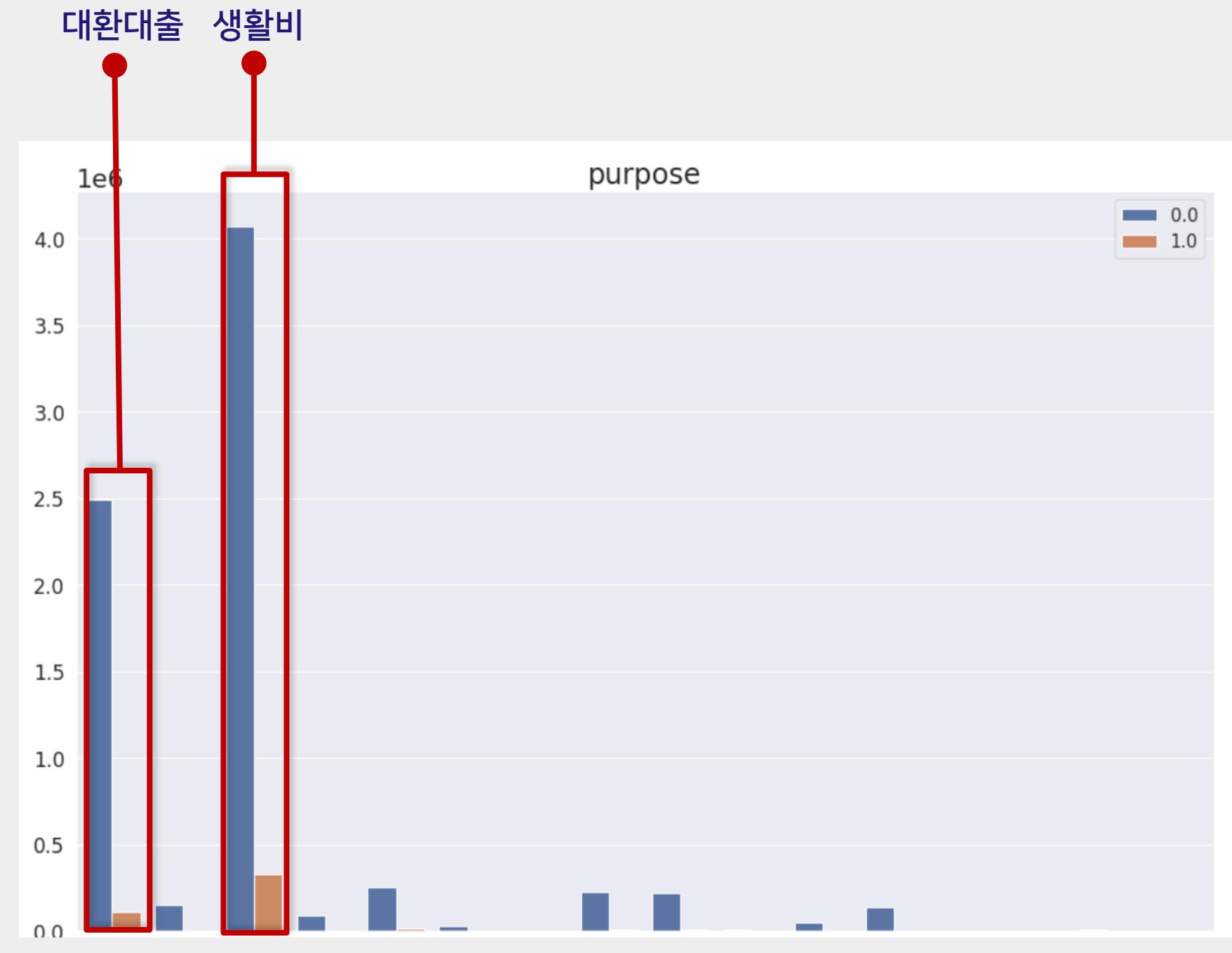
02 Data exploration

데이터 분포 확인

- 범주형 데이터



✓ employment_type : 정규직의 비율이 많음을 확인할 수 있음

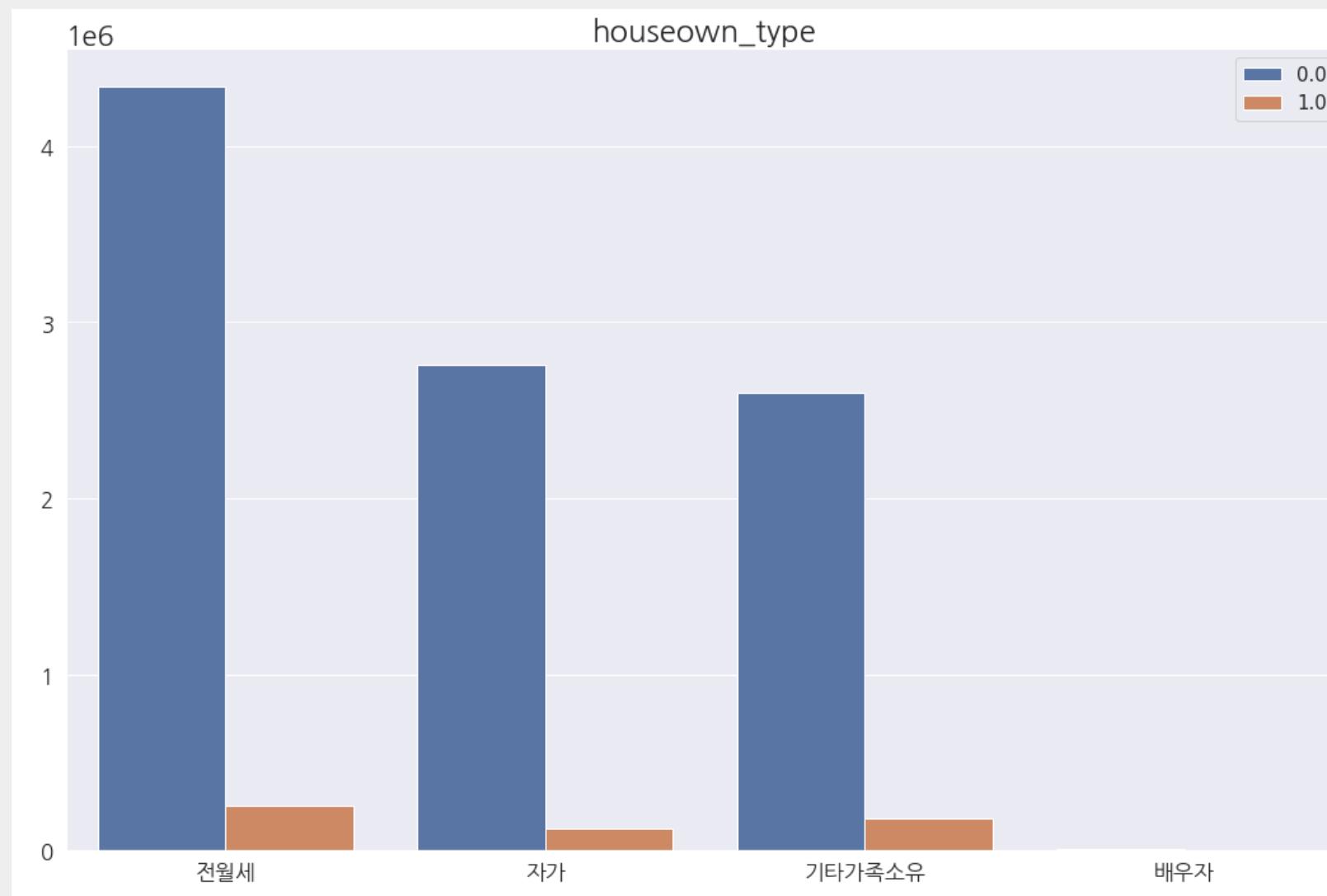


✓ purpose : 생활비와 대환대출 목적이 눈에 띄게 많음

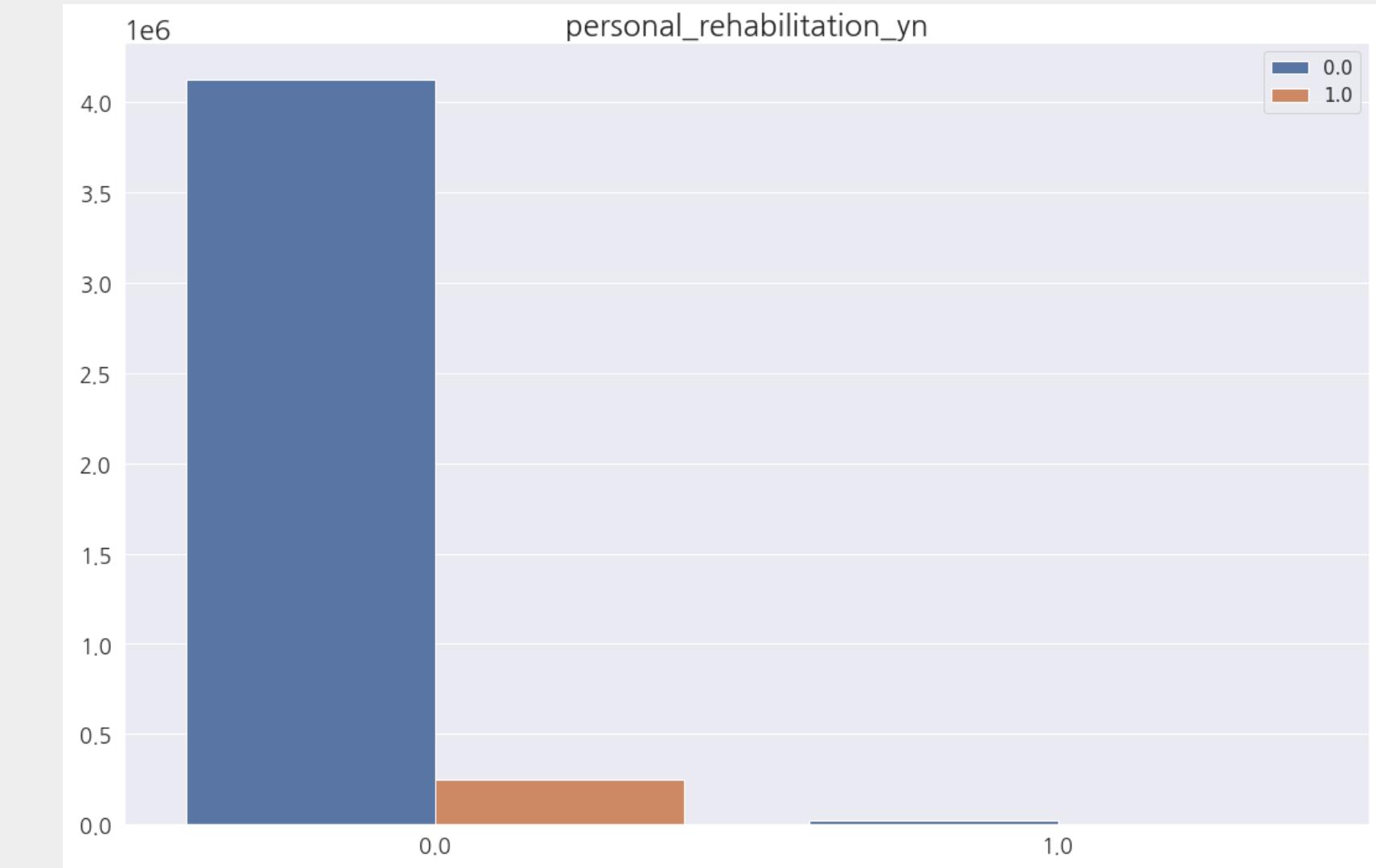
02 Data exploration

데이터 분포 확인

- 범주형 데이터



✓ houseown_type : 전월세 비율이 가장 높고, 배우자 비율이 가장 낮음

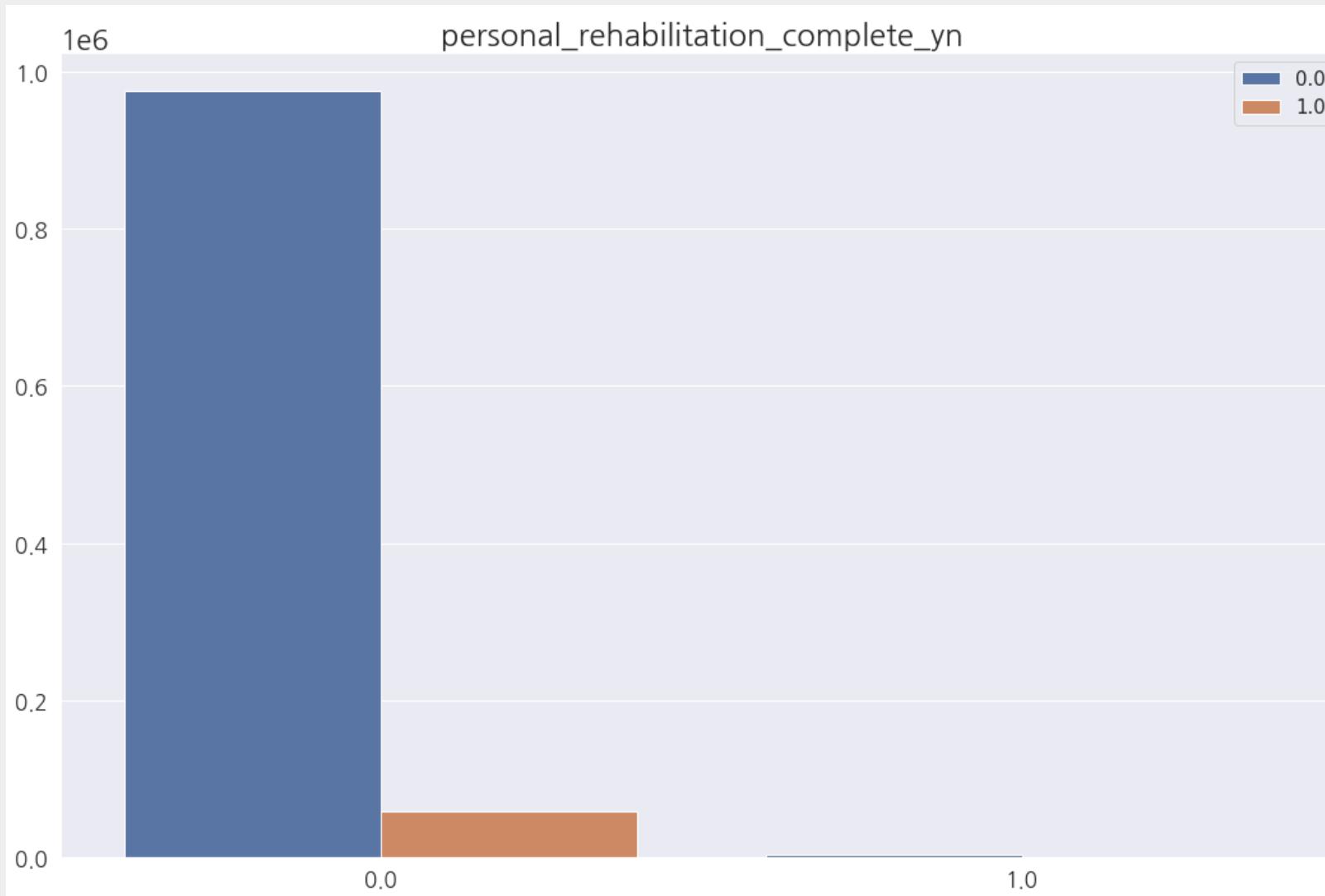


✓ personal_rehabilitation_yn : 개인회생자 고객(1)은 매우 적어
그래프 상에 나타나지 않음

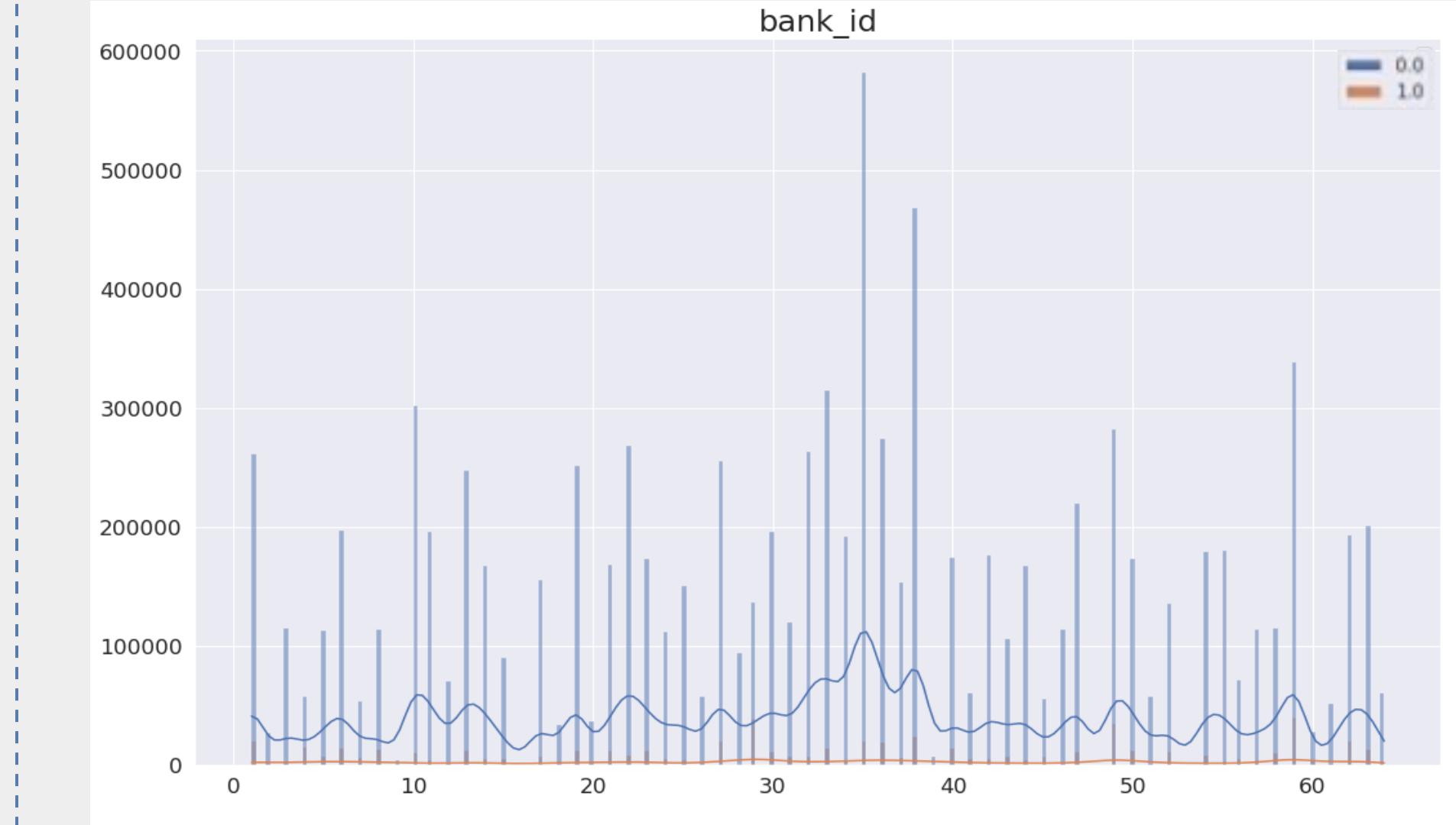
02 Data exploration

데이터 분포 확인

- 범주형 데이터



✓ personal_rehabiltiation_complete_yn : 개인 회생자가 납입을 완료한 비율은 매우 적어 그래프 상에 나타나지 않음

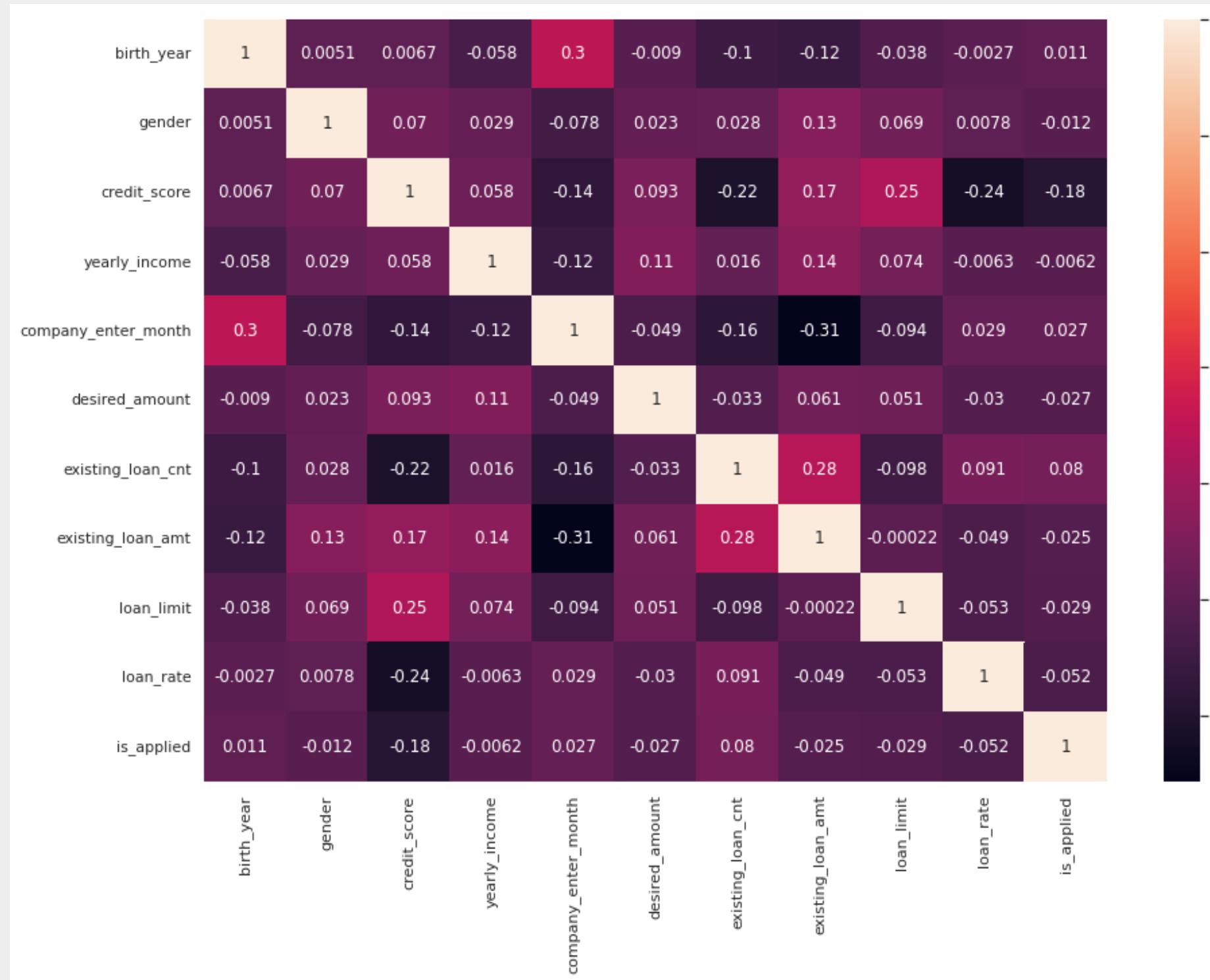


✓ bank_id : 특정 은행사의 비율이 높은 모습을 보임

02 Data exploration

상관관계

- 상관관계



✓ target과의 관계

모든 column들이 target과 -0.03 ~ 0.03 사이의 유의미하지 않은 상관관계를 가짐

✓ 독립변수들간의 관계 (loan_result, user_spec 병합 데이터)

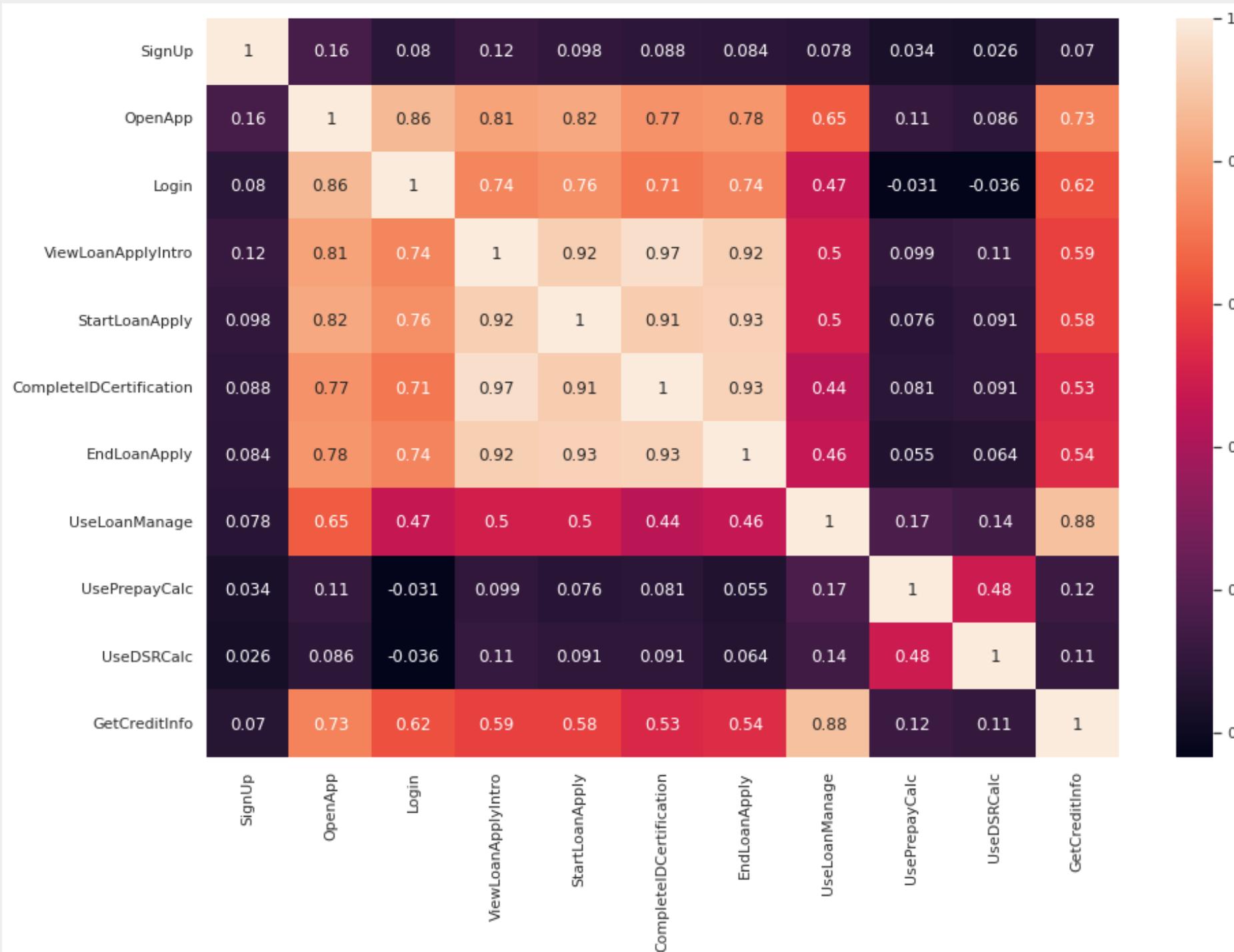
생년(birth_year)과 입사년월(company_enter_month)이 0.3,
기대출수(existing_loan_cnt)와 기대출금액(existing_loan_amt)이 0.28,
신용점수(credit_score)와 승인한도(loan_limit)가 0.25의 관계를 가짐

전반적으로 아주 약한 상관관계를 가지거나 유의미한 상관관계를 보이지 않음

02 Data exploration

상관관계

- 상관관계



✓ 독립변수들간의 관계 (log_data)

OpenApp, Login, ViewLoanApplyIntro, StartLoanApply, CompleteIDCertification, EndLoanApply 간의 상관관계가 0.7~0.9 사이를 띠고 있음

→ 다중공선성을 의심할 수 있음

03

Preprocessing

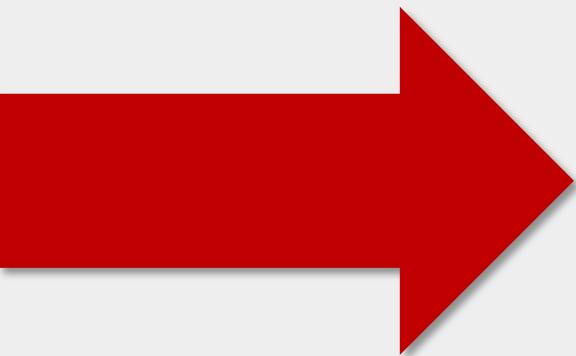
- 불필요한 변수 제거
- 파생변수 생성
- 결측치 처리
- 이상치 처리
- 인코딩
- 스케일링

03 Preprocessing

불필요한 변수 제거

삭제하는 변수

SignUp
OpenApp
ViewLoanApplyIntro
StartLoanApply
CompleteIDCertification
insert_time
loanapply_insert_time
company_enter_month
gender
personal_rehabilitation_complete_yn
personal_rehabilitation_yn



03 Preprocessing

파생 변수 생성

- ✓ 총 5개의 파생변수

bank_rank (대출 신청 은행사 개수 순위)	ls_applied가 1인 데이터 중 bank_id의 개수가 제일 많은 순서대로 그룹화하여 순위를 결정한 뒤 bank_rank 생성
admission_count (승인 상품 개수)	각 application_id 당 승인된 상품 개수 특정 application_id의 개수가 곧 product_id의 개수(승인된 상품 개수)이므로 중복되어 있는 application_id의 개수를 셈
over_desired (희망 금액 대비 대출 한도 초과 여부)	희망 대출 금액과 대출 한도 금액을 비교하여 desired_amount(희망 대출 금액)보다 loan_limit(대출 한도 금액)가 클 경우 1, 그렇지 않을 경우 0의 값을 할당
age_level (Life Time Event별 연령)	birth_year를 사용하여 2022년을 기준으로 해당 고객의 age(연령)을 구한 뒤 고객의 Life Time Event를 이용해 age를 범주화하여 age_level생성
credit_loan_rate (신용점수별 평균금리 초과 여부)	'여신금융협회의 신용점수별 평균금리현황'에서 신용점수 별 대출 금리의 평균을 구한 뒤, 각 신용점수 구간에 해당하는 loan_rate와 비교하여 loan_rate가 더 높을 경우 0, 낮을 경우 1의 값을 할당

03 Preprocessing

파생 변수 생성

1) bank_rank (대출 신청 은행사 개수 순위)

bank_id의 개수가 많은 순서대로 정렬

bank_id 값	bank_id 개수
59	39225
29	38721
49	34081
38	23731
62	19407
...	...
61	546
39	387
9	216
28	185

bank_id가 개수가 많은 순서대로 높은 값을 부여

bank_rank	bank_id
bank_rank : 6	[59, 29, 49, 38]
bank_rank : 5	[2, 35, 27, 1, 36]
bank_rank : 4	[4, 33, 6, 40, 8, 63, 50, 13, 19, 21, 23, 30, 47, 52]
bank_rank : 3	[58, 10, 54, 22, 32, 17, 5, 31, 43, 45, 7, 37]
bank_rank : 2	[14, 56, 41, 42, 15, 24, 34, 57, 44, 60, 11, 25, 55, 2, 64, 18, 3, 26, 20, 46, 51, 12]
bank_rank : 1	[61, 39, 9, 28, 16]

가장 높은 순위

가장 낮은 순위

bank_id	bank_rank
59	6
33	4
10	3
61	1
39	1
14	2
37	3

03 Preprocessing

파생 변수 생성

2) admission_count (승인 상품 개수)

application_id	loanapply_insert_time	bank_id	product_id	loan_limit	loan_rate	is_applied	admission_count
1850179	2022-04-02 16:01:01	38	223	10000000.0	17.1	0	4
1850179	2022-04-02 16:01:13	38	16	10000000.0	17.1	0	4
1850179	2022-04-02 16:01:03	63	226	6000000.0	18.7	1	4
1850179	2022-04-02 16:01:03	37	206	45000000.0	19.9	0	4
961500	2022-05-26 13:41:16	58	175	12000000.0	16.8	0	3
961500	2022-05-26 13:41:14	10	149	10000000.0	16.3	0	3
961500	2022-05-26 13:41:14	10	65	10000000.0	16.3	0	3

ex) application_id 가 '1850179' 인 사람의 데이터프레임

- 승인된 상품 개수가 총 4개 → application_id의 중복된 개수

ex) application_id 가 '961500' 인 사람의 데이터프레임

- 승인된 상품 개수가 총 3개 → application_id의 중복된 개수

03 Preprocessing

파생 변수 생성

3) over_desired (희망 금액 대비 대출 한도 초과 여부)

ex) application_id 가 '1362052' 인 사람의 데이터프레임

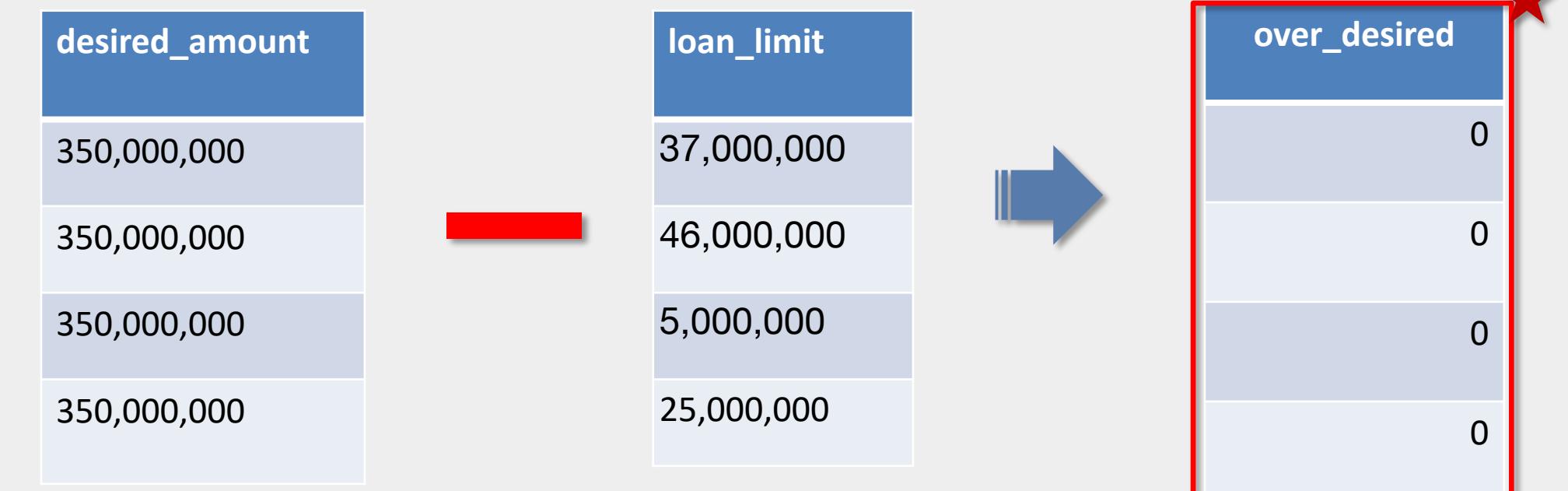
application_id	user_id	gender	credit_score	yearly_income	desired_amount	loan_limit	bank_id	product_id	is_applied
1362052	21535	1.0	790.0	46000000.0	350000000.0	37000000.0	33	110	0
1362052	21535	1.0	790.0	46000000.0	350000000.0	46000000.0	10	65	0
1362052	21535	1.0	790.0	46000000.0	350000000.0	5000000.0	12	35	0
1362052	21535	1.0	790.0	46000000.0	350000000.0	25000000.0	45	119	1

✓ 'desired_amount' – 'loan_limit' < 0

희망 대출 금액보다 실제로 승인된 대출 한도가 더 높게 나왔으므로 1 부여

✓ 'desired_amount' – 'loan_limit' > 0

희망 대출 금액보다 실제로 승인된 대출 한도가 더 높게 나왔으므로 0 부여

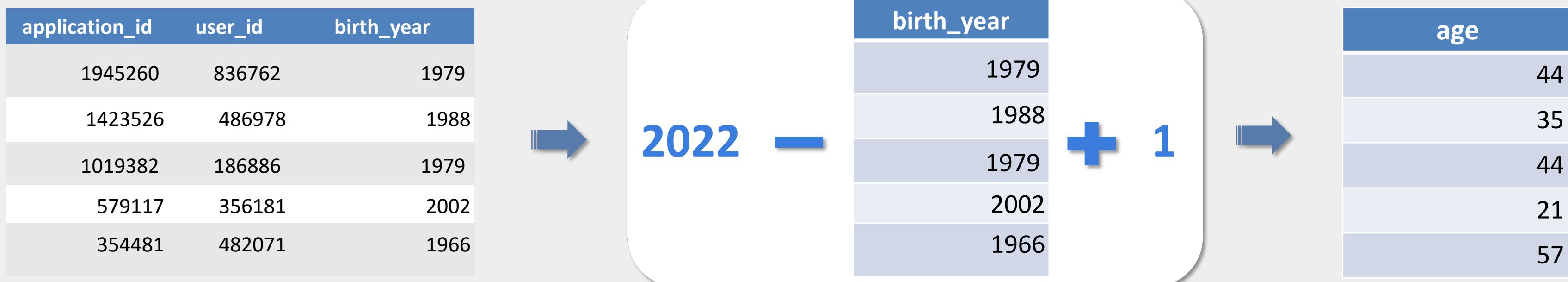


03 Preprocessing

파생 변수 생성

4) age_level

- ✓ 'birth_year' 칼럼은 태어난 연도로 보는 것보다 나이를 보는 것이 더 중요하다고 판단 → 현재 연도와 출생 연도와의 차이를 통해 age 칼럼 생성



- ✓ 금융데이터는 고객의 Life Time Event에 따라 소비 성향이 크게 달라지기 때문에, 대출을 하는 데에 영향이 있을 것이라고 판단하여 'age'를 그대로 반영하지 않고 범주화

1. 평균 취업 연령 : 28세
2. 평균 결혼 및 출산 연령 : 32세
3. 평균 내 집 마련 연령 : 40세
4. 평균 은퇴 연령 : 60세

The diagram shows the mapping of age ranges to age_level categories. A red star points to the age_level column.

age	age_level
20~27	0
28~31	1
32~39	2
40~59	3
60~	4

03 Preprocessing

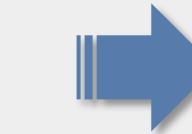
파생 변수 생성

5) credit_loan_rate

핀다와 연결된 제2금융권의 신용점수 별 평균 금리

- 대부분의 은행에서 신용점수가 높을수록 평균 금리가 낮아지는 것을 알 수 있음
- 신용점수가 500점 이하이면 대출 승인이 잘 발생하지 않는다는 것을 알 수 있음

회사명	900점 초과	801점~ 900점	701점~ 800점	601점~ 700점	501점~ 600점	401점~ 500점	301점~ 400점	300점 이하	평균 금리
롯데카드	12.62	13.90	16.18	19.59	19.90	-	-	-	16.92
비씨카드	11.26	11.87	11.71	-	-	-	-	-	11.73
KB국민카드	9.28	10.70	13.03	14.29	14.37	-	-	-	12.84
한국캐피탈	13.05	14.54	15.57	16.71	18.27	-	19.90	-	15.31
BNK캐피탈	12.22	12.69	13.59	14.89	15.78	-	-	-	13.99
DGB캐피탈	10.45	12.75	13.02	14.53	17.90	-	-	16.99	13.69
KB캐피탈	11.29	13.51	15.58	16.81	17.57	-	-	-	13.68
롯데캐피탈	10.60	11.83	12.82	14.14	14.65	-	-	-	13.03
우리금융캐피 탈	12.21	15.75	17.53	18.74	19.13	-	-	-	16.55
하나캐피탈	10.25	10.71	11.65	12.25	10.90	-	-	-	11.22
현대캐피탈	12.92	15.53	17.55	18.25	18.50	19.18	-	18.45	17.05
JB우리캐피탈	11.43	12.89	14.68	16.12	16.86	19.90	-	-	15.14
NH농협캐피탈	11.50	12.75	14.26	16.12	18.79	19.90	-	-	15.38



application_id	user_id	loan_rate	credit_loan_rate
1850179	376060	18.7	0
1850179	376060	17.1	1
52259	867773	10.8	1
52259	867773	10.6	1

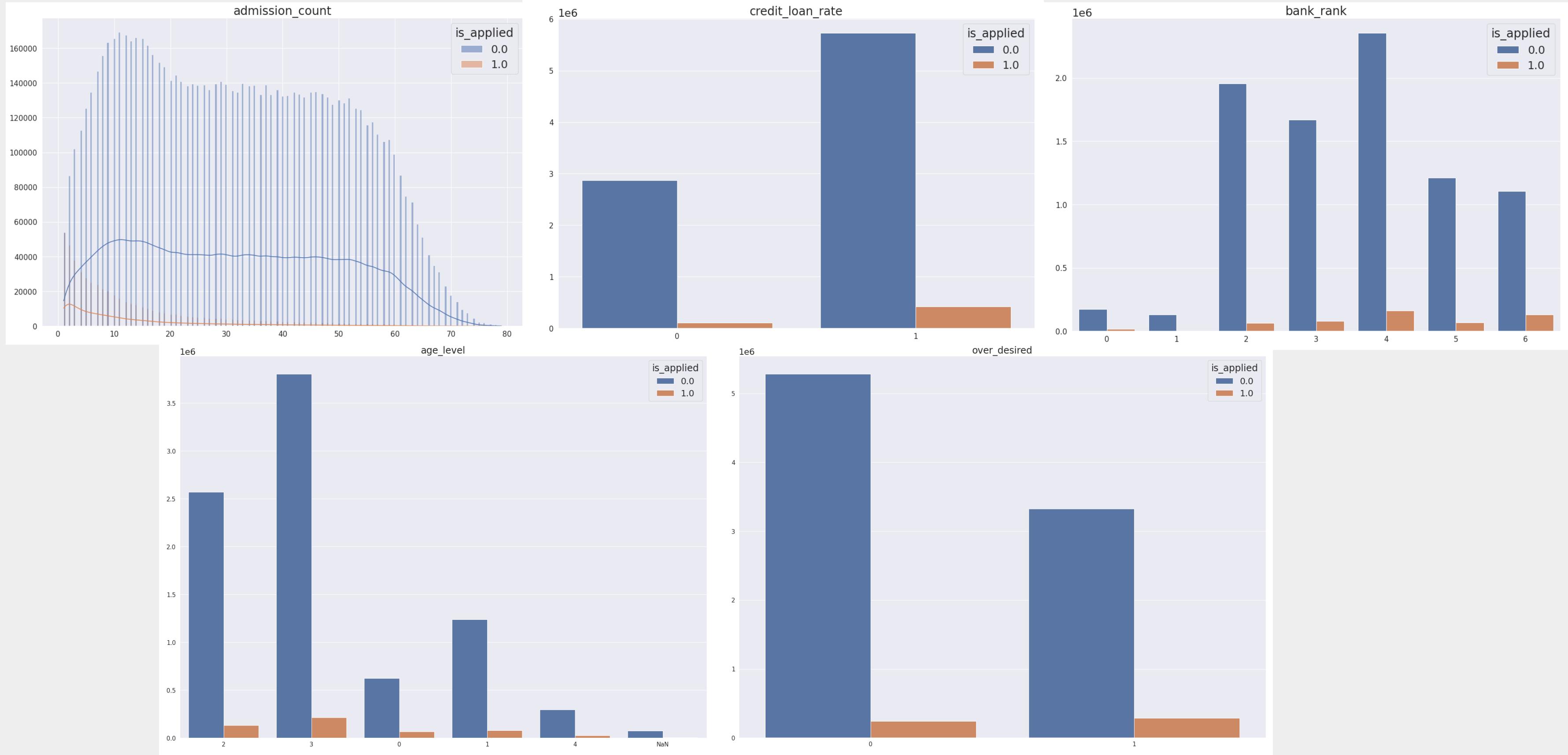
대출 선택 조건에 중요하게 작용한다고 판단하여, 실제 신용점수별로 할당된 평균 금리 대비 loan_rate 초과 여부 칼럼 생성

평균 금리 < loan_rate → 0
평균 금리 ≥ loan_rate → 1

03 Preprocessing

파생 변수 생성

파생 변수 분포 확인



03 Preprocessing

결측치 처리

- 결측치 개수 확인

train

★ credit_score : 1040769개

existing_loan_cnt : 1807898개

existing_loan_amt : 2598633개

loan_limit : 5358개

loan_rate : 5358개

age : 78052개

test

★ credit_score : 25523 개

yearly_income : 6개

existing_loan_cnt : 48698 개

existing_loan_amt : 74134 개

loan_limit : 390 개

loan_rate : 390 개

age : 4336개

* credit_score → credit_score와 상관계수가 0.2인 column들을 선택하여 따로 결측치 채움

03 Preprocessing

결측치 처리

- *train, test* 공통 적용

- ✓ 0으로 채우기

- `existing_loan_cnt` : 최솟값이 1이기 때문에 결측치는 기대출이 없는 값이라고 판단하여 0으로 채워줌
- `existing_loan_amt` : 기대출수가 결측치이면 기대출금액도 결측치이기 때문에 0으로 채움

- ✓ 결측치 처리 모델 [Iterative Imputer](#) 사용

- `credit_score` : 결측치가 많았고, `is_applied`과 연관성이 있다고 판단하여 credit score와 상관계수가 0.2 이상인 칼럼들 선별 후 `IterativeImputer` 이용
- `existing_loan_cnt` : 0으로 채우고 나머지 900292개의 결측치는 `IterativeImputer`를 이용해 채움



Iterative Imputer : MICE를 이용한 자동 대치

- Round robin 방식을 반복하여 결측 값을 회귀하는 방식으로 결측치를 처리
- 결측 값을 회귀하는 방식으로 처리하기 때문에 수치형 변수에만 사용
- 수치에 민감하므로 스케일링 처리를 한 후 적용

03 Preprocessing

결측치 처리

- *train*만 적용

- ✓ 행 제거

- loan_rate, loan_limit : 결측치(nan)가 있는 경우는 금융사에서 값을 보내주지 않은 경우로, 해당 경우는 채점에서 제외할 예정이니 해당 행 제거
 - 이 외의 특정 값으로 채우기 어렵다고 판단되는 결측치가 있는 행(existing_loan_amt, age) 제거

- *test*만 적용

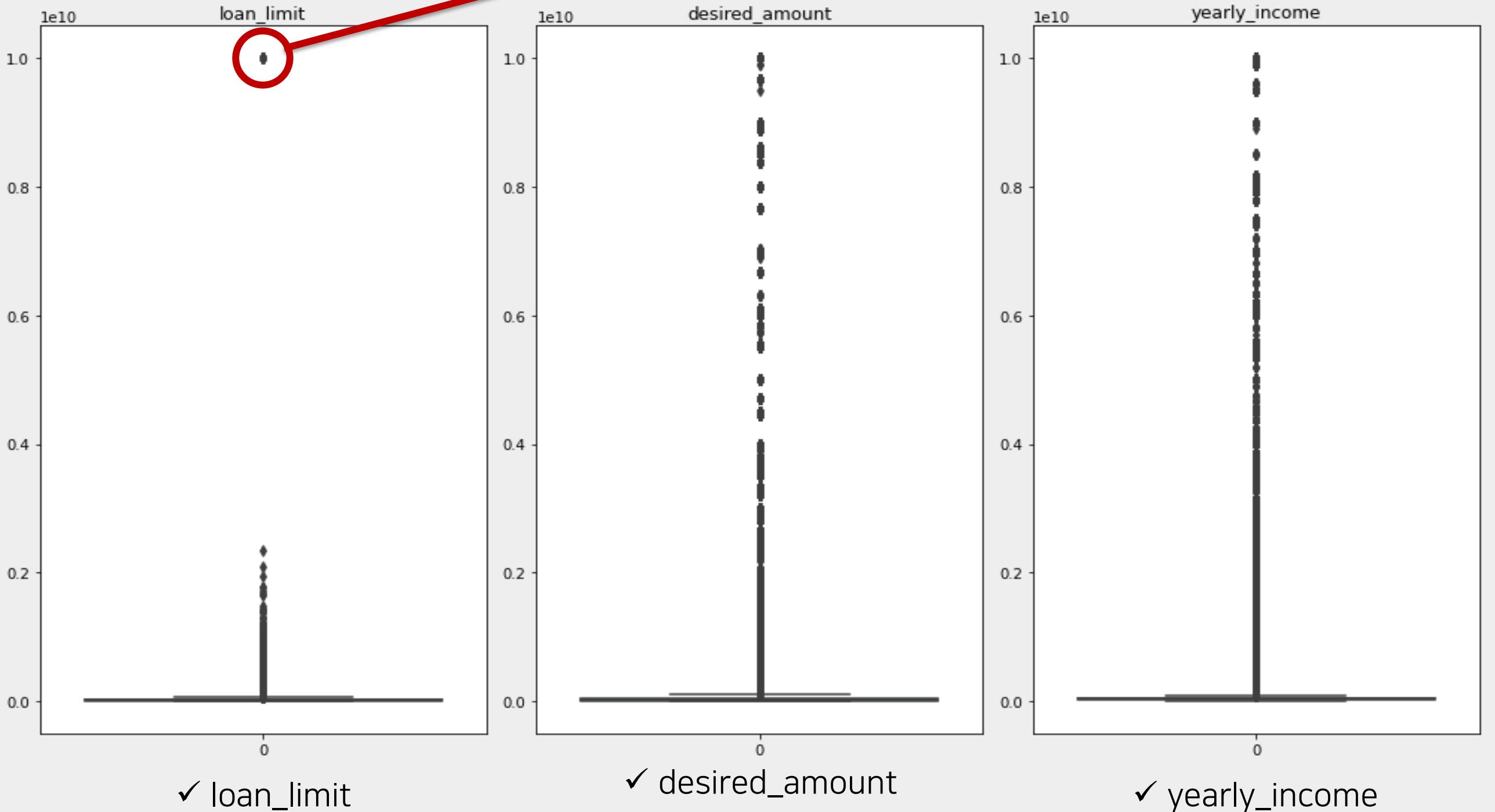
- ✓ Imputer 사용

- 이 외의 특정 값으로 채우기 어렵다고 판단되는 결측치가 있는 행(existing_loan_amt, yearly_income, loan_rate, loan_limit, age) 을 IterativeImputer를 이용해 채움

03 Preprocessing

이상치 처리

- *boxplot을 통한 이상치 확인*



03 Preprocessing

인코딩

- *Label Encoder*

- ✓ 범주형 데이터(income_type, employment_type, houseown_type, purpose)를 수치화하기 위해 Label encoder 사용



application_id	user_id	...	income_type	employment_type	houseown_type	purpose	is_applied
1850179	376060		0	3	2	11	1
1850179	376060		0	3	2	11	0
52259	867773		0	3	0	11	0
52259	867773		0	3	0	11	0

인코딩 결과

03 Preprocessing

스케일링

✓ log변환

범위가 큰 변수인 credit_score, yearly_income, desired_amount, existing_loan_amt, loan_limit, Login, EndLoanApply, UseLoanManage, UsePrepayCalc, UseDSRCalc, GetCreditInfo에 대해

log변환을 통해 큰 수를 작게 만들어 가중치 효과를 줄이고,
왜도와 첨도를 줄여 데이터 분석 시 의미있는 결과를 도출할 수 있게 해줌



application_id	user_id	credit_score	yearly_income	desired_amount	existing_loan_amt	loan_limit	is_applied
1850179	376060	6.41510	17.52908	16.11810	18.32637	15.60727	1
1850179	376060	6.41510	17.52908	16.11810	18.32637	16.11810	0
2167826	640786	6.32972	17.28125	15.42495	18.18496	15.42495	0
2167826	640786	6.32972	17.28125	15.42495	18.18496	16.90655	0

스케일링 후 데이터

04

Modeling

- 모델 소개
- 모델 평가
- 모델 구축
- 해석

04 Modeling

모델 소개

데이터의 특징

- 1) 불균형한 target값
Is_applied의 0과 1 비율이 19 : 1로 심각한
unbalanced dataset
- 2) 과적합 가능성
target 값이 1인 수가 매우 적어 Overfitting의 가능성 존재
- 3) 대용량 데이터셋
총 학습 데이터셋 10,270,011 rows × 23 columns



Machine Learning Tree Model

1. 이상치와 노이즈에 큰 영향을 받지 않음
2. 복잡한 비선형문제를 빠른 시간에 풀 수 있고, 모델의 해석력이 비교적 높음

CatBoost

LightGBM

Random Forest

모델 소개

Machine Learning Tree Model

Random Forest

LightGBM

CatBoost

✓ 결측치를 다루기 쉬움

✓ 빠른 학습 속도와 높은 성능
✓ 데이터가 많은 경우 효과적

✓ 기존 Boosting모델의 단점을 보완한
Ordered-Boosting 방식으로
과적합 우려 적음
✓ 범주형 자료가 많은 데이터셋에 효과적

04 Modeling

모델 평가

- ✓ F1-score를 주된 평가 지표로 보는 이유

$$F1 = \frac{2}{\frac{1}{precision} + \frac{1}{recall}} = 2 \times \frac{precision * recall}{precision + recall}$$

Precision(정밀도) : True라고 분류한 것들 중에서 실제로 True인 것의 비율

Recall(재현율) : 실제 True인 것 중에서 True라고 예측한 것의 비율

이 데이터셋은 0이 95%, 1이 5%로 레이블 값이 불균형한 특징을 갖고 있음

모든 예측값을 0으로 예측하는 '맹목 예측'을 할 경우, accuracy는 95%라는 높은 값을 보여줌

→ 따라서 불균형 데이터에서 모델 성능을 판단할 경우에는 accuracy가 적합한 평가 지표가 아님

→ 정밀도와 재현율의 조화평균인 F1-score를 평가지표로 사용

04 Modeling

모델 평가

- 3가지의 모델 성능 비교

*각 모델 별 최고의 성능을 낼 수 있는 전처리와 모델 튜닝을 진행한 것을 기준

	Accuracy	Precision	Recall	F1-score	ROC-AUC
XGboost	0.9229	0.3607	0.5293	0.4290	0.7375
LightGBM	0.9134	0.3421	0.6310	0.4436	0.7804
Catboost	0.9337	0.4201	0.5581	0.4793	0.7567

Voting을 통한 앙상블 방법 또한 CatBoost와 유사한 성능을 보였지만, 모델들의 예측 값을 가중평균 할 경우 설명력이 비교적 떨어질 수 있기에 과적합 우려가 적고, 범주형 변수가 많은 데이터에 효과적인 단일 모델인 **CatBoost 채택**

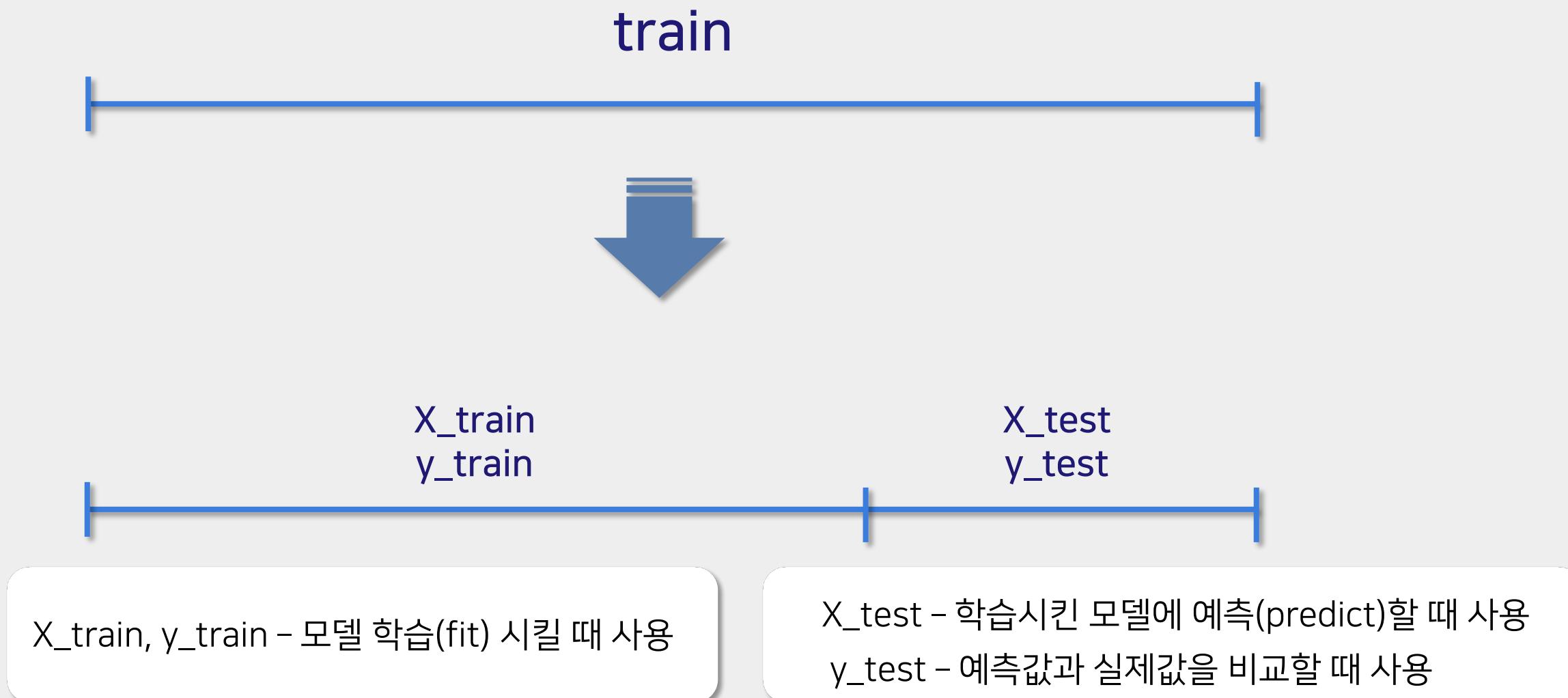
Voting을 통한 앙상블 방법 또한 CatBoost와 유사한 성능을 보였지만, 모델들의 예측 값을 가중평균 할 경우 설명력이 비교적 떨어질 수 있기에 과적합 우려가 적고, 범주형 변수가 많은 데이터에 효과적인 단일 모델인 **CatBoost 채택**

04 Modeling

모델 구축

- *train_test_spilt*

전체 데이터(train)를 8:2의 비율로 train과 test로 나누어 모델의 성능 파악



모델 구축

- **불균형 데이터 처리**

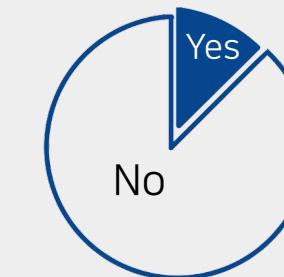
- ✓ **클래스 불균형**

어떤 데이터에서 각 Class (주로 범주형 반응 변수) 가 갖고 있는 데이터의 양에 차이가 큰 경우

현실 데이터에는 클래스 불균형 (class imbalance) 문제가 자주 있음



어떠한 경우 클래스 균형이 필요한가?
소수의 클래스에 특별히 더 큰 관심이 있는 경우



대출을 **신청한다고 예측하는 것과 신청하지 않는다고 예측하는 것은 중요도가 다름**

→ 잘못된 마케팅 비용 투자는 손실로 이루어질 수 있기 때문에 가입한다고 예측하는 것은 더 큰 리스크 수반

따라서 '대출을 신청한다' 라고 예측하는 것에 대해서는 **더 큰 성능을 보여야 함**

하지만 데이터가 '대출을 신청하지 않는다' class에 몰려있는 경우, '대출을 신청하지 않는다'에 대한 예측에 있어서는 높은 성능을 보일 수 있어도 '대출을 신청한다'에 대한 예측에 있어서는 예측 성능이 좋지 않게 될 수 있음

04 Modeling

모델 구축

- *Hyperparameter Tuning*

불균형 처리

✓ class_weights

Class 불균형이 있는 경우, Class에 따라 모델의 성능이 달라지게 됨

해결책

'대출 신청' Class에는 더욱 큰 비중(weight)를 두고 정확한 예측을 할 수 있도록 만들어야 함
class 별로 가중치를 다르게 부여하는 Tree기반 모델의 class_weights 파라미터 설정을 통해 불균형 문제를 해소

04 Modeling

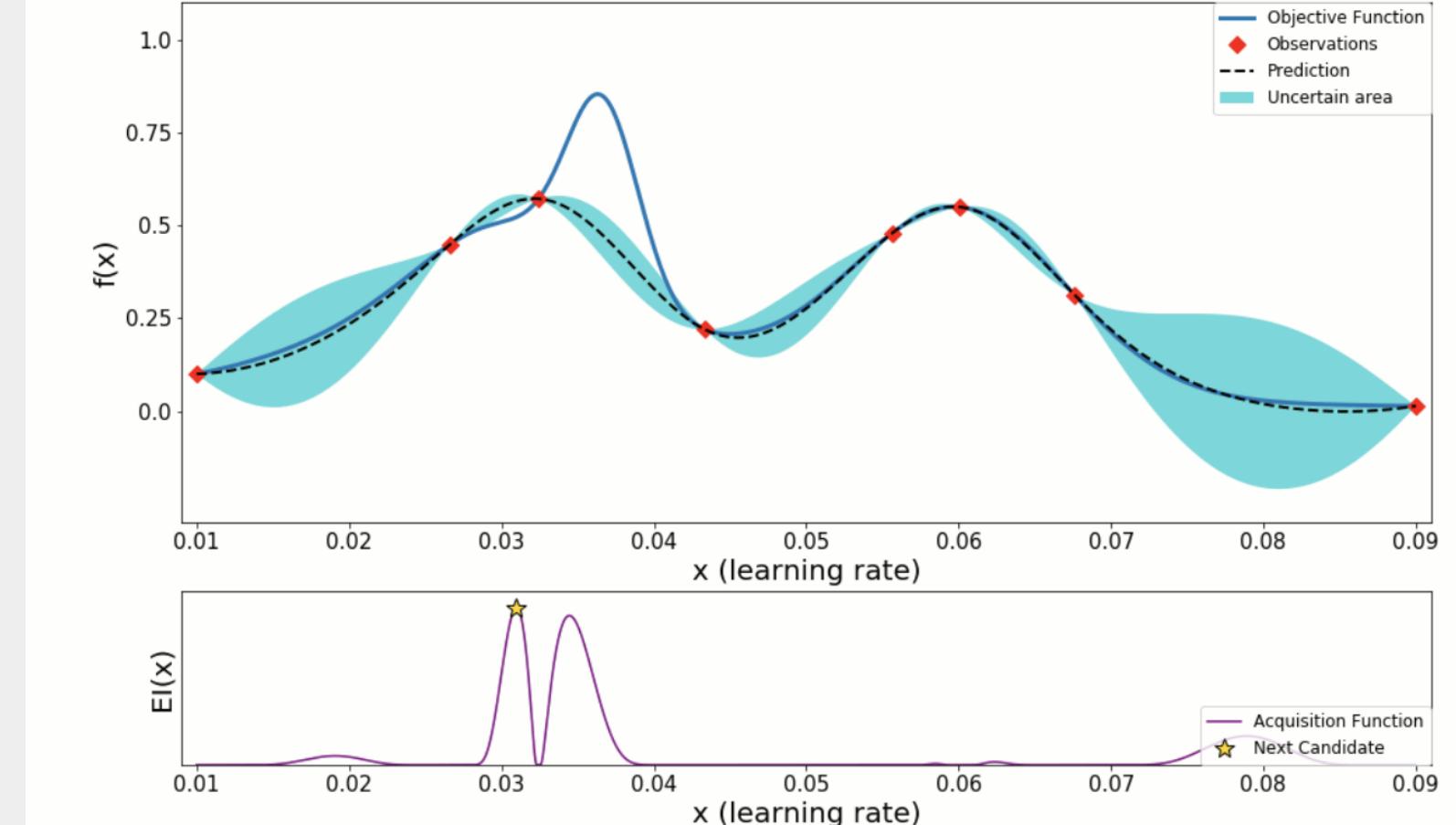
모델 구축

- *Hyperparameter Tuning*

Bayesian optimization : 어느 입력값(x)을 받는 미지의 목적 함수를 상정하여 해당 함숫값($f(x)$)을 최대로 만드는 최적해를 찾는 것을 목적으로 함

매 회 새로운 hyperparameter 값에 대한 조사를 수행할 시
‘사전 지식’을 충분히 반영하면서, 동시에 전체적인
탐색 과정을 체계적으로 수행할 수 있는 방법론

→ 하이퍼 파라미터 튜닝 후 2%p 성능 향상



Bayesian Optimization 수행 과정

04 Modeling

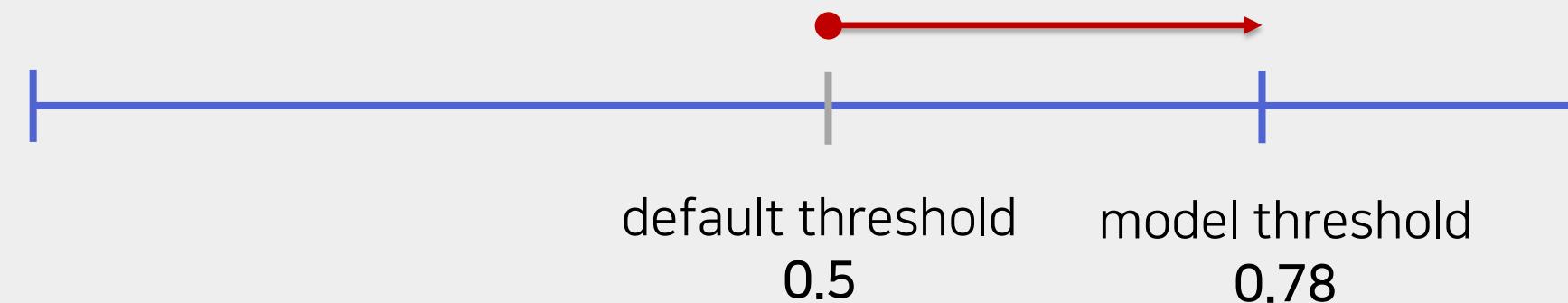
모델 구축

- *Threshold*

- ✓ threshold값 변경을 통해 조금 더 보수적으로 대출 신청 여부 판단

비즈니스 관점에서 회사의 이익을 극대화하기 위한 방법

대출을 신청하지 않는 사람들에 비해 대출을 신청하는 사람들의 비율은 소수이지만 비즈니스 관점에서는 신청하는 고객이 더 중요
모델의 기본 threshold인 0.5는 불균형한 데이터에서 너무 많은 사람을 신청한다고 예측하기 때문에 손실이 많아질 수 있음
따라서 threshold를 0.78로 설정함으로써 precision을 높이고 recall을 낮춤 → 이를 통해 f1-score가 0.4에서 0.48로 상승



* 실제 대출 신청 당 회사의 이익을 계산할 수 있는 비즈니스 로직을 구축할 수 있다면
model metric을 기반으로 threshold 최적화를 실행하는 대신 실제로 비즈니스 결과를 기반으로 최적화할 수 있음

04 Modeling

해석

- *SHAP Value*

SHAP Value(SHapley Additive exPlanations)는 모든 기계 학습 모델의 결과(출력)를 설명하기 위한 게임 이론적 접근 방식

- ✓ 양상블에서 변수 해석의 문제



Tree기반 양상블들은 전반적으로 우수한 성능을 내는 모델들로 알려져 있지만,
양상블 기법을 사용하면서 Decision Tree들의 결합과 반복되는 학습과정에서 **Decision Tree의 뛰어난 직관성이 사라짐**
변수 및 모델의 설명력을 위해서 Tree를 사용하는 것인데, 성능을 높이려고 양상블 기법을 추가하다 보니 원래의 목적을 잃는 것
이처럼 **모델의 성능과 설명력은 모델 선택에 있어서 trade-off 관계가 존재**

→ 이러한 이유 때문에 대부분의 **Feature Importance** 지표는 **Inconsistency** 함

But, SHAP Value는 여러 번 수행하며 반복적인 결과물의 평균이기에 **Consistency**가 잘 유지

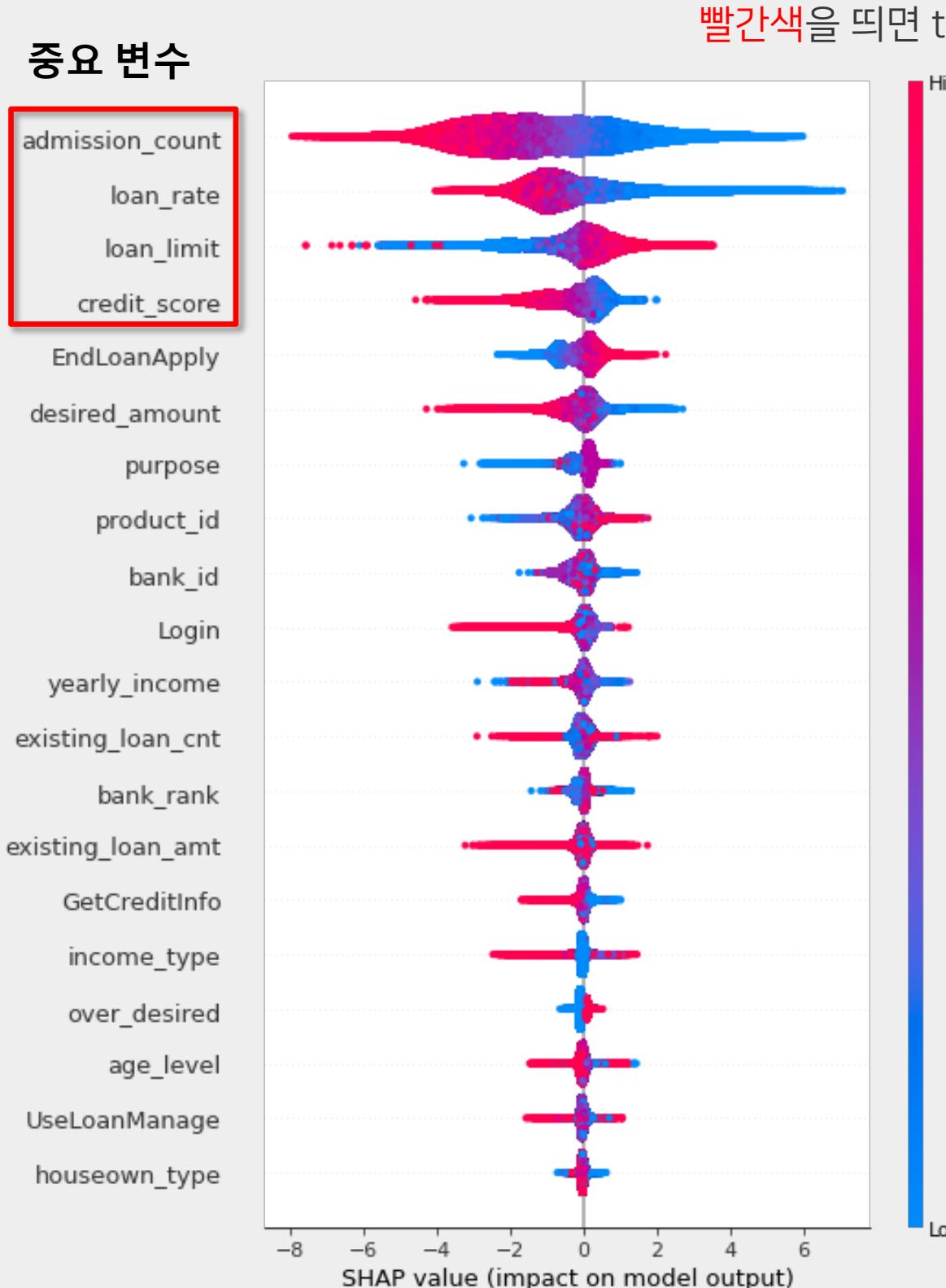
SHAP Value는 영향에 대한 방향성(positive or negative) 또한 설명

→ **Feature Importance**에서 단점이 보완

04 Modeling

해석

- *SHAP Value*



- 1) admission_count(승인 상품 개수) : 신용 점수가 높을수록 승인되는 상품의 개수가 많아짐
- 2) loan_rate(대출 금리) : 신용 점수가 높을수록 대출 금리가 낮은 경향이 있음
- 3) loan_limit(대출 한도) : 신용 점수가 높을수록 대출 한도가 높아짐

→ 따라서 중요 변수 4개가 전부 연관성이 있다고 판단

추후 credit_score를 중심으로 서비스 제안

05

Clustering

- 추가 전처리
- PCA
- K-means
- 해석

05 Clustering

추가 전처리

✓ application_id당 하나로 데이터 정리

이전에 전처리한 데이터는 하나의 application_id에 대해 admission_count 개수만큼 데이터가 존재

이때 admission_count가 많은 사람의 정보가 군집화 할 때 많이 반영되는 것을 방지하기 위하여 신청서 당
사용자 개인정보 & 신청서 입력 정보가 중복되는 데이터들을 삭제

How?

- 신청 이력이 있는(is_applied가 1인 데이터가 존재하는) 신청서의 경우 is_applied가 1인 행만 남김
- is_applied가 0만 있는 신청서의 경우 결국 신청하지 않은 사람이므로 한 행만 남기고 나머지 제거

✓ 파생변수 service_dummy 생성

각 user_id 당 log_data의 event 칼럼의 각 행동명들의 행동횟수를 센 뒤 대출조회 외의 기능인

'UseLoanManage', 'UsePrepayCalc', 'UseDSRCalc', 'GetCreditInfo' 값을 더함

이후 service_dummy에 이상치가 있음을 확인하고 이상치 제거

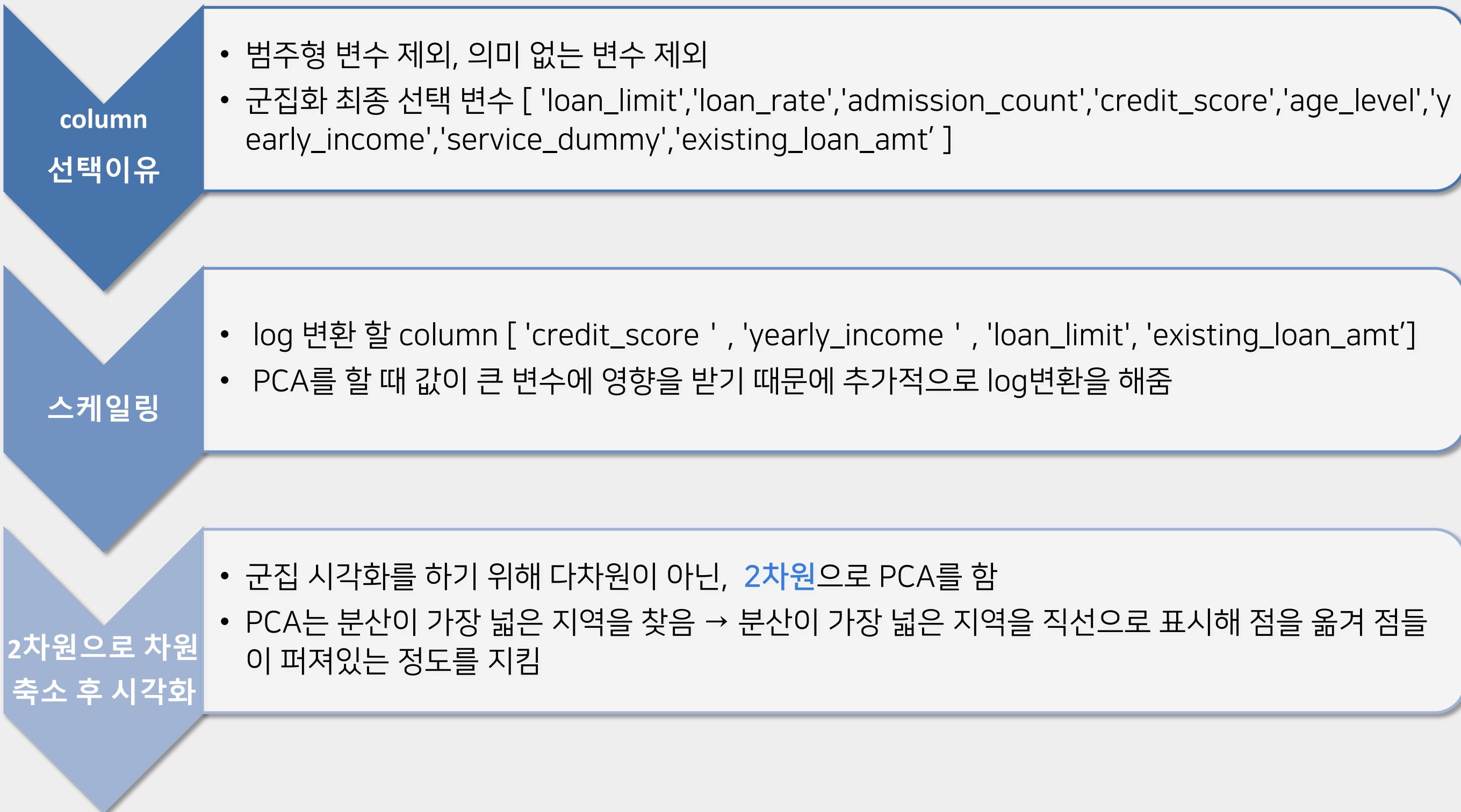


대출 신청한 사람들(is_applied = 1)과 대출 신청하지 않은 사람들(is_applied = 0)을 분리하여 군집화

05 Clustering

PCA

- *column 선택 후 스케일링*

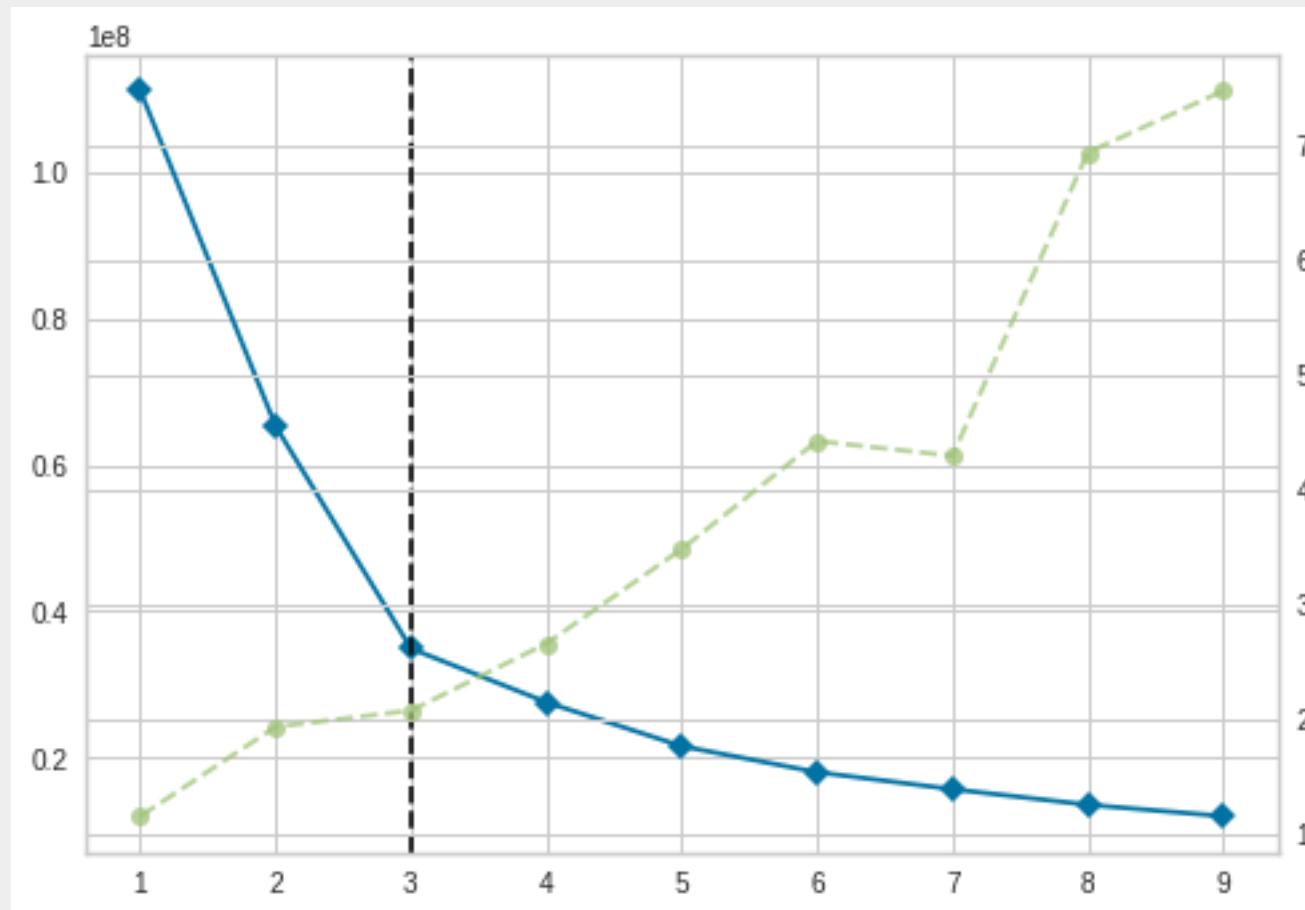


05 Clustering

K-Means

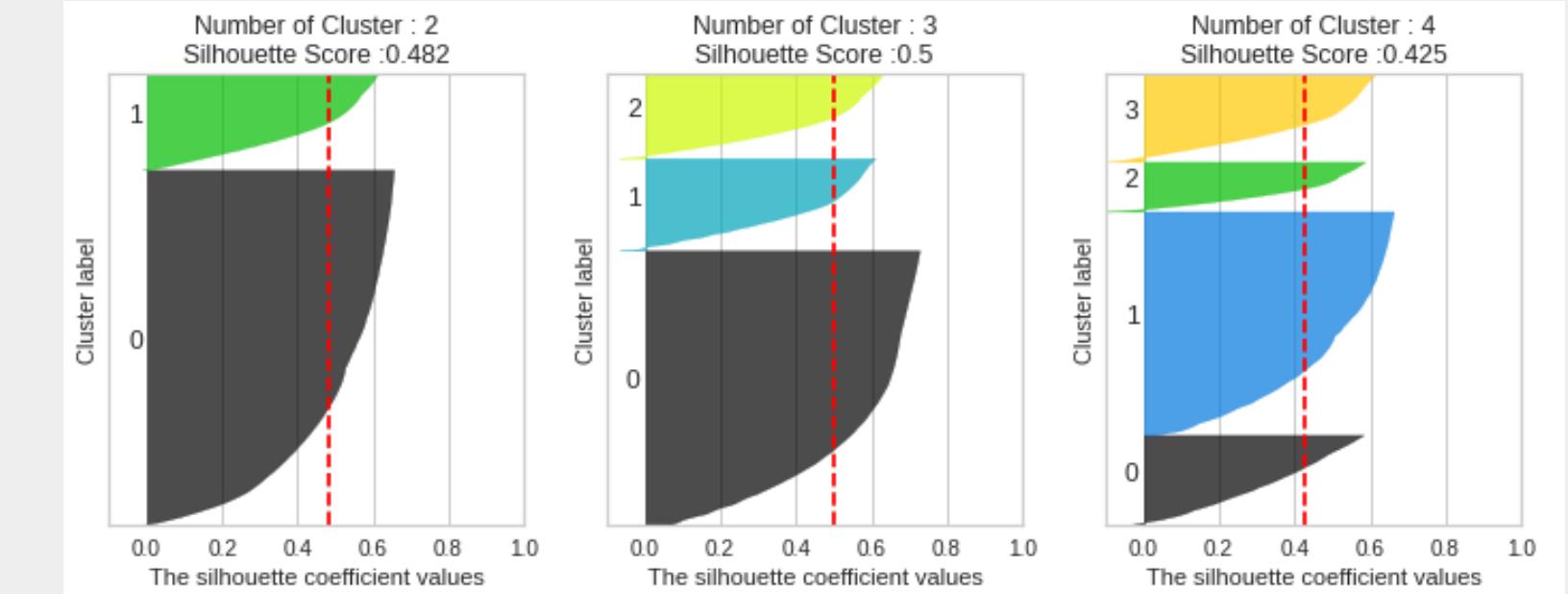
- *Elbow point*

군집수에 따라 군집내 총 제곱합(SSE)을 확인하여 가장 그래프가 꺾이는 지점(팔꿈치의 위치)을 통해 적절한 군집 수를 선택하는 방법



k = 3일 때 군집내 총 제곱합이 가장 높으며 군집수가 높아지면서 감소 경우에 따라서 날카로운 팔꿈치가 없을 수도 있는 단점이 존재

- 실루엣 계수



빨간 점선 : 전체 실루엣 계수의 평균

- 1) 군집 개수 = 2, 평균 실루엣 계수 0.482

군집별 실루엣 계수 평균값이 0보다 크고 편차가 크지 않지만, 두 군집의 데이터 수의 차이가 큼

- 2) 군집 개수 = 3, 평균 실루엣 계수 0.5

1, 2번 군집에 실루엣 계수가 음수인 데이터들이 존재함
따라서 2번 군집의 실루엣 계수와 전체 실루엣 계수에 편차가 조금 큼

- 3) 군집 개수 = 4, 평균 실루엣 계수 0.425

군집의 데이터 비율이 더 일정하지 않아졌으며, 실루엣 계수가 음수인 데이터들이 존재

→ 두 가지 평가 방법을 종합적으로 고려하여 최적 군집 개수 $n = 3$ 선택

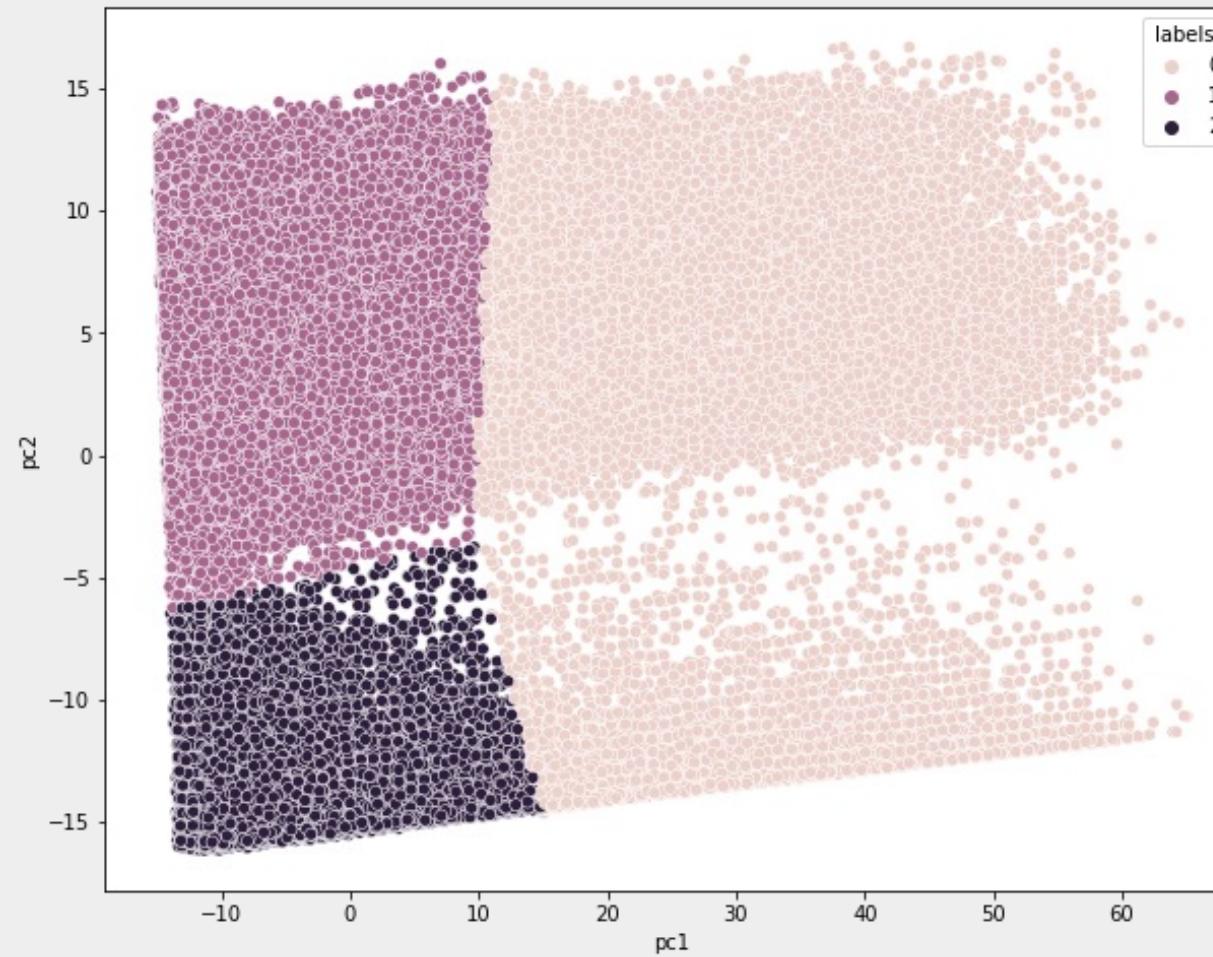
05 Clustering

K-Means

- ✓ 군집화 방법 : K-Means

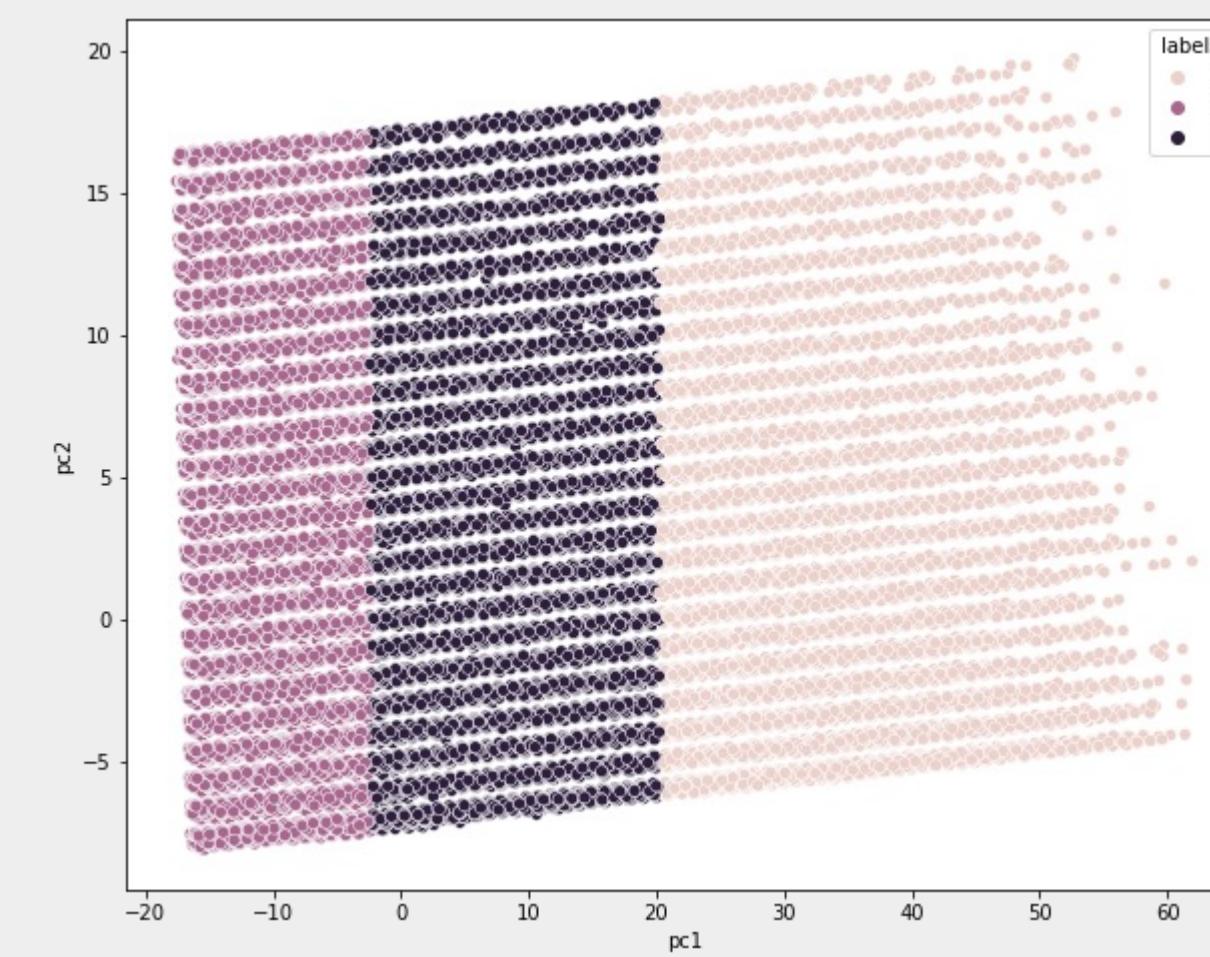
n개의 중심점을 찍은 후에, 이 중심점에서 각 점 간의 거리의 합이 가장 최소화가 되는 중심점 n의 위치를 찾고,
이 중심점에서 가까운 점들을 중심점으로 묶는 알고리즘

- ✓ is_applied = 1인 군집



0번 : 19%. 1번 : 63% 2번 : 18%

- ✓ is_applied = 0인 군집



is_applied = 0인 군집에서
군집화를 진행하였으나 군집 별
특성에서 유의미한 특징을
찾아내지 못하여 is_applied=0인
군집은 하나의 군집으로 설정

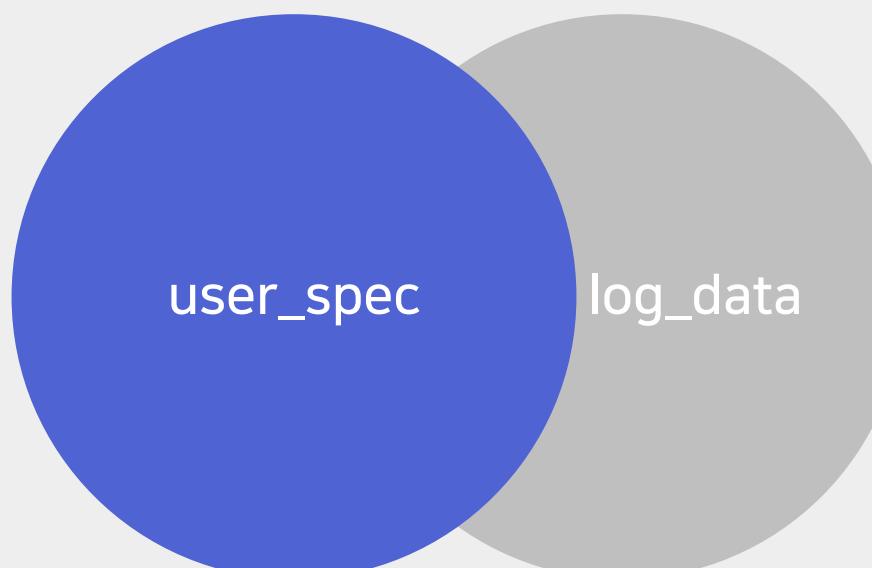
<군집 시각화 결과>

05 Clustering

해석

- 군집 별 특성

평균	is_applied = 1				is_applied = 0
	A	B	C	D	
승인상품(개수)	7.2	40.43	8.33	17.04	
기대출금액(원)	65,320,278	52,660,589	0	56,381,519	
기대출수(개수)	5.11	2.80	0.01	3.37	
서비스 이용(횟수)	8.68	6.42	1.61	6.92	
신용 점수(점수)	636	768	663	699.33	



기대출금액이 제일 많은 군집
앱 내 다른 서비스를
가장 많이 이용함

중간 정도의 어플 사용량
승인된 상품이 제일 많고
신용점수가 제일 높음

다른 서비스를 잘 이용하지 않음
기대출수가 거의 없고
평균 기대출 금액이 50만원 미만
(10만단위에서 반올림 처리되어
0원 기록)

보통 기대출은 존재하지만
핀다로 대출을 신청하지 않은 군집
B와 비슷한 추세를 보이지만
승인 상품이 더 적고
신용점수가 더 낮음

< 대출 한도 조회 서비스를 이용하지 않은 군집 E >

log_data의 584,636명의 유저 중 214,200명(36%)이 대출 한도 조회 서비스를 이용하지 않음
이 중 68,936명이 다른 서비스 조차 이용하지 않음(11%)

* user_spec에는 없고 log_data에만 있는 데이터

06

Conclusion

- 결론
- 신용 점수 관리 서비스 제안
- 군집 별 메시지 제안

06 Conclusion

결론

Modeling

admission_count, loan_limit, loan_rate, credit_score 등이 모델링에서 중요하게 작용

위 4개의 변수는 결국 credit_score와 관련이 깊은 것으로 판단하여
credit_score 변수에 초점을 맞추어 마케팅 전략 수립 및 진행

Clustering

대출을 신청한 3개의 군집에서는 군집 별 특징이 잘 나타나고,
대출을 신청하지 않은 3개의 군집에서는 유의미한 특징이 나타나지 않음

대출을 신청한 군집 3개, 대출을 신청하지 않은 군집 1개, 대출 조회를 하지 않은 군집 1개
총 5개의 군집 별 특성을 반영하여 핀다를 효율적으로 이용하기 위한 메시지 제안

06 Conclusion

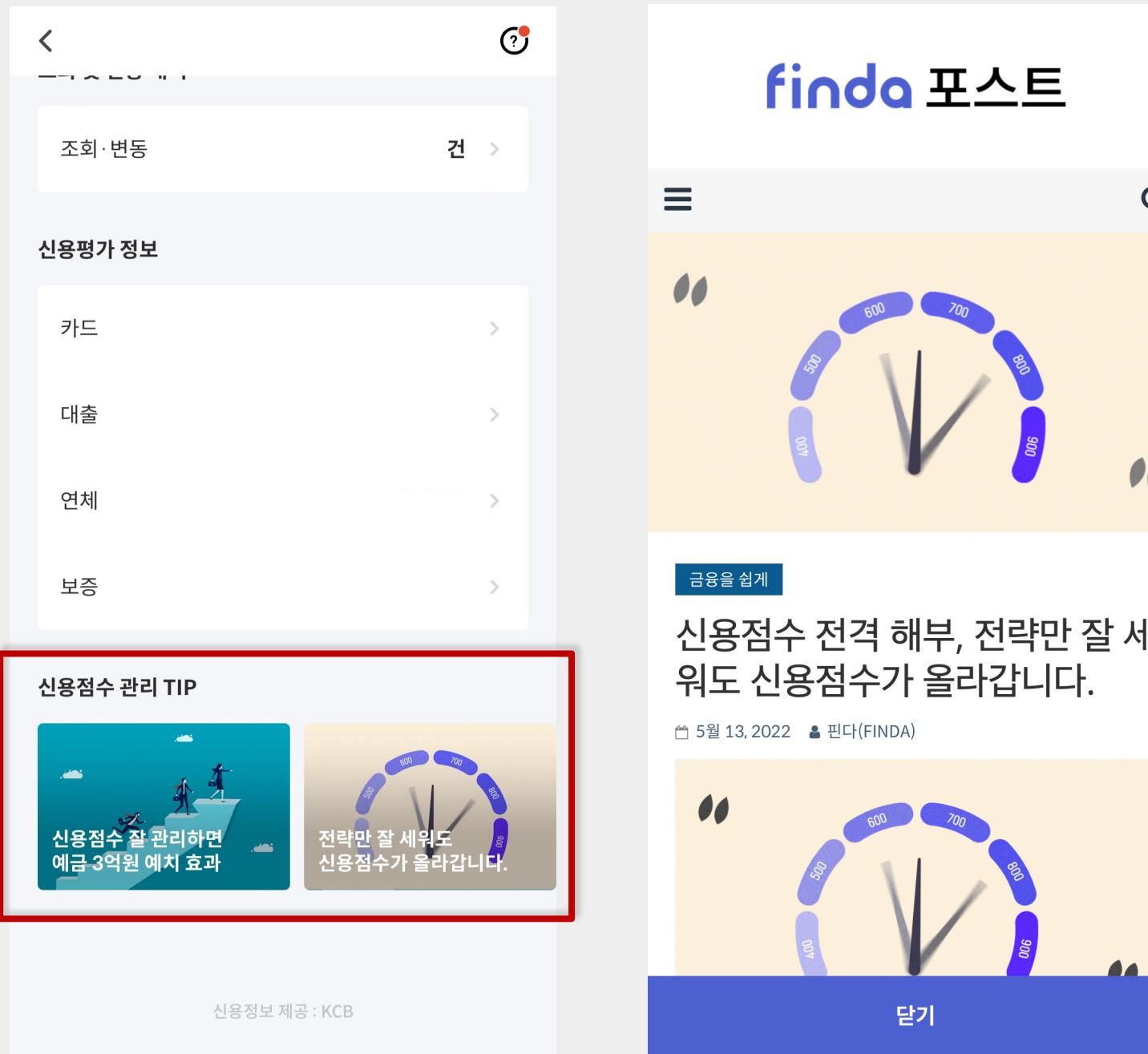
신용 점수 관리 서비스 제안

• 기존 서비스

✓ Finda 포스트를 통한 정보 제공

모델링의 SHAP value를 통해 얻은 중요도 상위 4개의 변수들이 모두 credit_score와 관련이 높은 것으로 판단

→ [기존의 서비스를 개선한 신용점수 관리 서비스를 제안](#)



신용 점수 조회 카테고리에서 “finda 포스트”를 통해
관리 방법 관련 정보 제공



간접적인 정보 제공으로 앱 기능 내에서
전반적인 신용 점수를 관리하기에는 한계 O

06 Conclusion

신용 점수 관리 서비스 제안

• 관리 서비스

- ✓ 신용점수 관리 페이지 생성 & 마이데이터 활용

신용점수에 영향을 주는 요소들을 관리할 수 있는 기능들을 모은 페이지 생성

마이데이터를 활용해 대출 관리를 제외한 다른 신용점수를 높일 수 있는 요소들을 앱 내에서 직접 관리

1) 카드 사용액과 한도 조회 및 가계부

- 진입 페이지에서 해당 월 카드 총 사용액, 목표금액 대비 사용액 비율, 사용 중인 카드의 한도 대비 사용액 비율을 제공
- 이어진 페이지에서 목표 사용금액 기능 등이 탑재된 가계부 기능 제공

2) 공공요금 · 통신비 납부 내역 제출

- 진입 페이지에서 납부 내역 제출 기록 존재 여부를 제공
- 이어진 페이지에서 납부 방법 설명, 납부 페이지로 연결 기능 제공

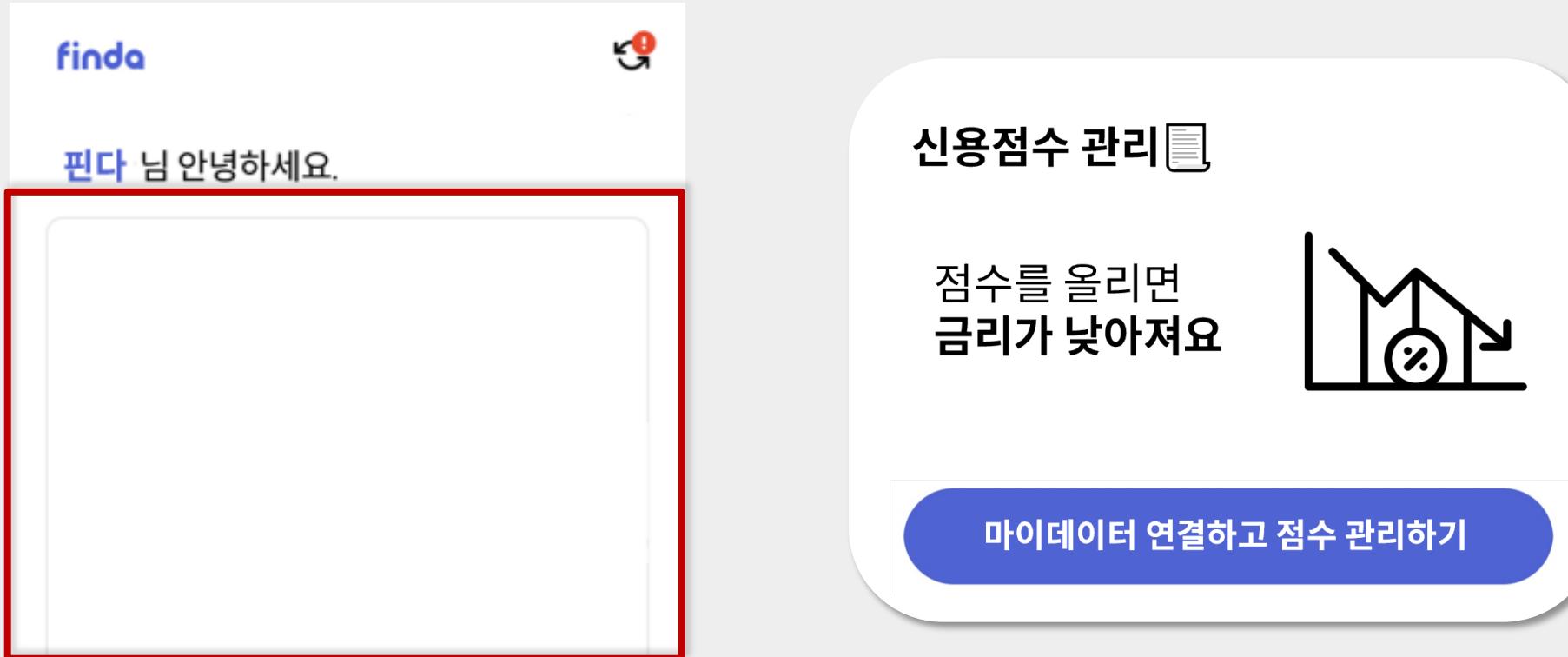
3) 대출 관리 페이지 연결

- 관리 페이지에서도 나의 대출 관리 기능을 연결해주는 진입페이지 제공

06 Conclusion

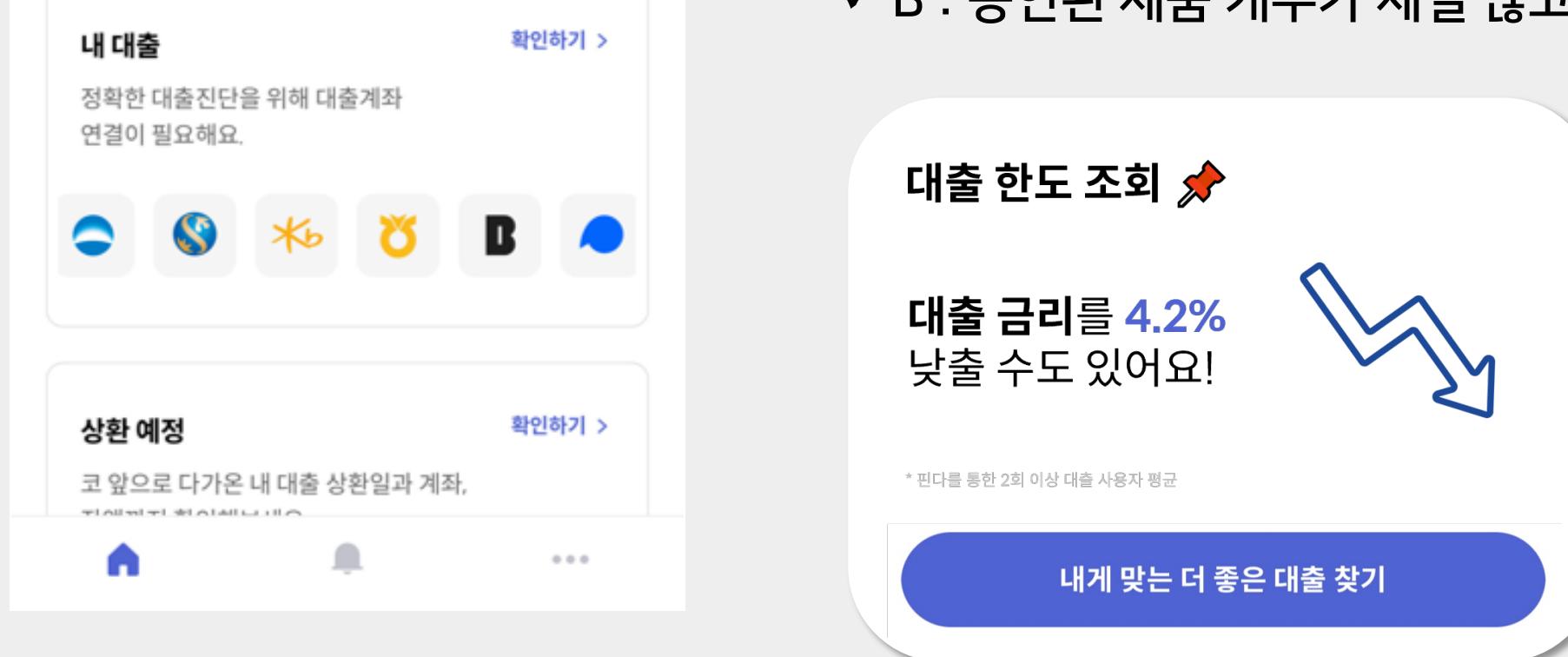
군집 별 메시지 제안

- ✓ A : 다른 서비스를 많이 이용하고 기대출금액이 제일 많은 군집



현재 대출을 활발히 이용하고 있고, finda를 통해서도 대출 서비스를 이용하며 다른 서비스를 많이 활용하고 있는 군집
→ 이미 어플리케이션을 잘 활용하고 있어 기존 서비스 내에서 추천할 필요성 감소
→ 보다 대출 환승 구조가 활발해지도록 새로 제안한 [신용점수 관리 서비스](#)를 이용하도록 유도하는 메시지 출력

- ✓ B : 승인된 제품 개수가 제일 많고 신용점수가 제일 높은 군집

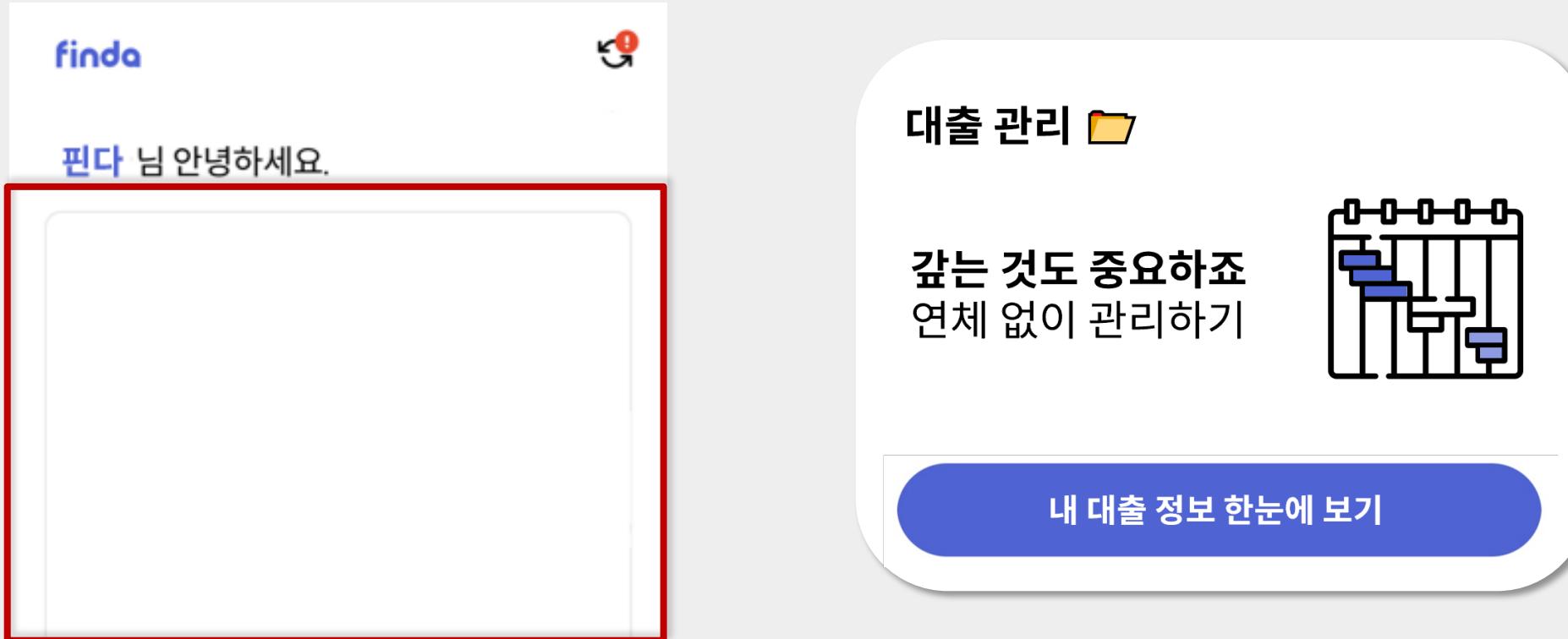


어느 정도 대출을 이용하고 있고, finda를 통해서도 대출을 하려 하지만 다른 서비스 이용이 높은 편은 아닌 군집
→ finda의 주 목적인 대출 환승이 활발하게 이루어지도록 대출 한도 조회를 유도하는 메시지 출력
→ 실제 finda 사용자들이 [서비스 사용으로 낮춘 대출 금리](#) 포인트 평균을 제시하거나 사용자가 해당하는 [군집의 평균 대출금리 대비 낮출 수 있는 금리](#) 포인트 제시

06 Conclusion

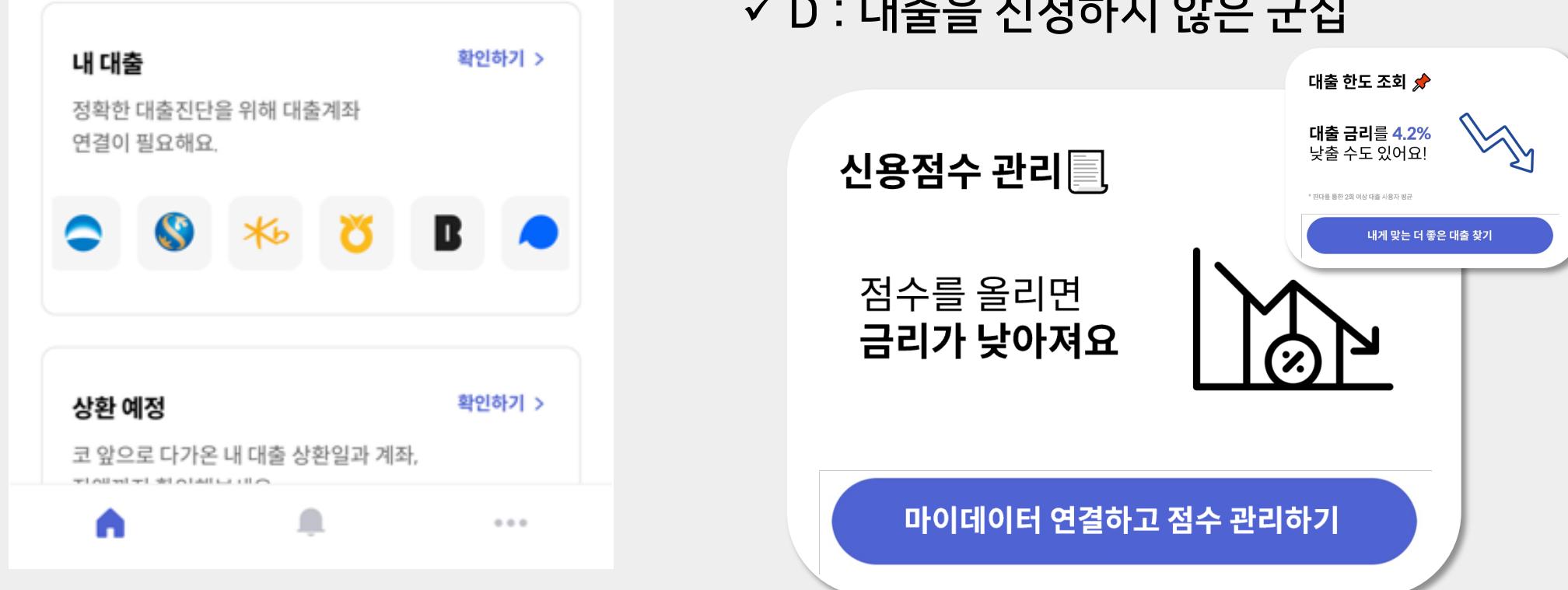
군집 별 메시지 제안

✓ C : 다른 서비스를 잘 이용하지 않는 군집 중 기대출수가 거의 없고 평균 기대출 금액 0



핀다를 통해서 처음 대출하거나 대출 경험이 적으면서 핀다의 다른 서비스를 잘 이용하지 않은 집단
핀다의 여러 서비스를 제안해볼 수 있는 가장 잠재성이 높은 집단이므로 가장 주력 기능으로 밀고 있는 [대출 관리 서비스를 사용해보도록 유도](#)하는 메시지 출력

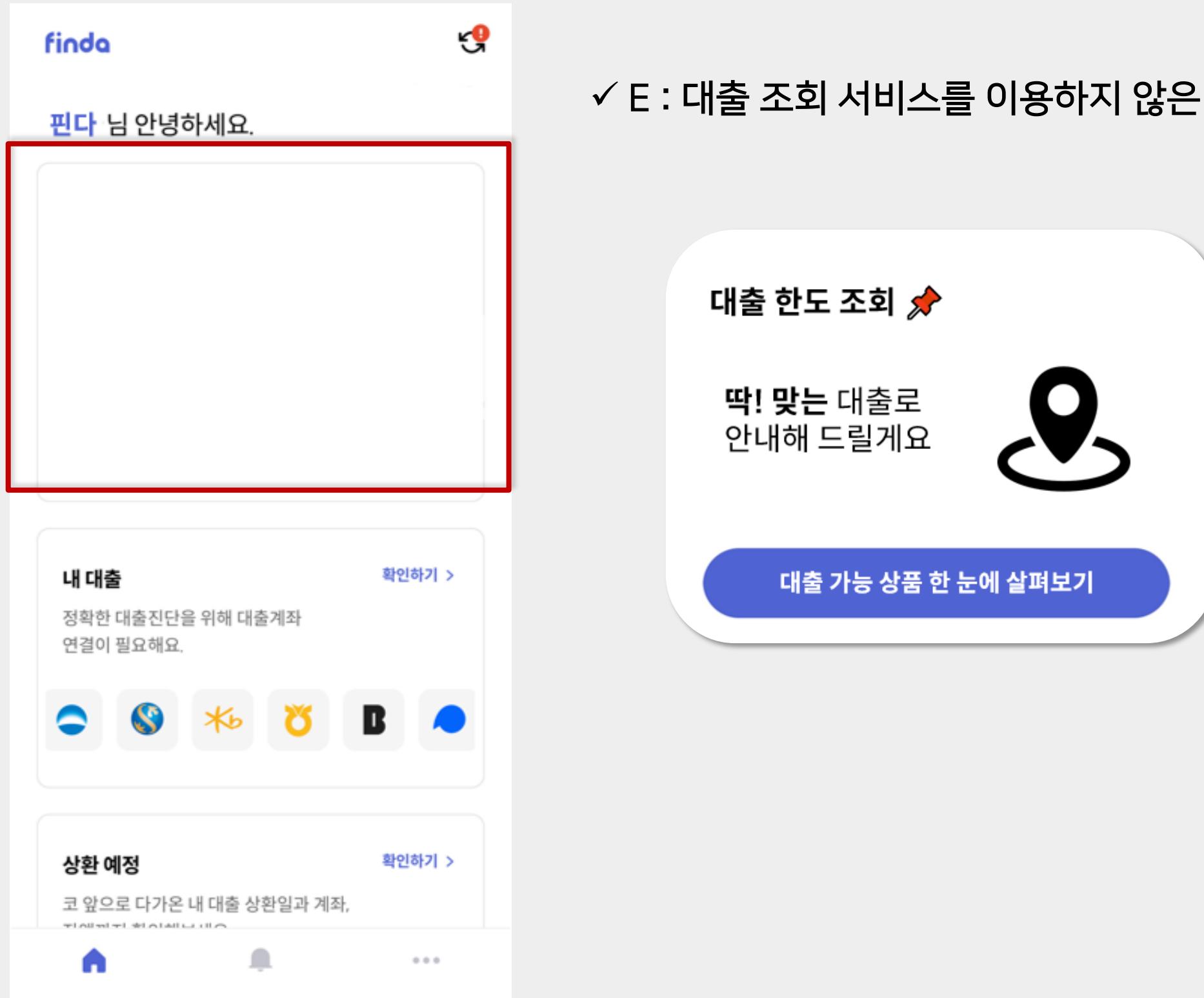
✓ D : 대출을 신청하지 않은 군집



B 군집과 비교하여 신용점수가 낮고 승인 상품이 적은 집단
B 군집과 같이 대출 환승이 활발하게 이루어지도록 하기 위해 B군집으로 유도하고자 [신용점수 관리 서비스 또는 대출 한도 조회 서비스를 이용해보도록 유도](#)하는 메시지 출력

06 Conclusion

군집 별 메시지 제안



✓ E : 대출 조회 서비스를 이용하지 않은 군집

다양한 이유로 대출에 관심이 있어 어플을 사용하기 시작했지만, 앱 실행, 회원가입 외에 뚜렷한 활동이 없는 군집
→ 사용자의 대출 현황을 알 수 없어 보다 자세한 서비스 제안이 어려우므로 정보 수집을 위해 **대출 한도 조회 서비스를 사용하도록 유도**

대출 이용 경험 여부를 알 수 없기 때문에 공통적으로 적용할 수 있도록 하기 위해서는 구체적인 수치 제안이 효과적이지 못할 것이라고 판단
→ finda의 주 마케팅 컨셉으로 사용하는 “환승”이라는 컨셉을 극대화하며 첫 대출 조회 서비스 이용을 유도하기 위해 “**길찾기**”라는 컨셉으로 소개

참고문헌

여신 금융 협회(외부 데이터) : <https://gongsi.crefia.or.kr/portal/creditloan/creditloanDisclosureDetail12?clgcMode=13>

기사 출처

[이필재가 만난 사람(29) 이혜민 핀다 공동대표] 최적화된 금융상품 추천하는 '온라인 PB' <https://jmagazine.joins.com/economist/view/327037>

[스타트] "핀테크 미래, '사람 냄새'에 있다" <https://www.ajunews.com/view/20210204134954551>

[아하 인터뷰] "대출비교 플랫폼 알면 빚 부담 허리 핀다" <https://www.asiatime.co.kr/article/20220701500221>

핀다 어플 사용 캡쳐

Bayesian Search 이미지 출처

<https://wooono.tistory.com/102>

아이콘 이미지 출처

"<https://www.flaticon.com>"

웃음꽃핀다

감사합니다