앱 사용성 데이터를 활용한 대출 신청 고객 예측 및 군집 맞춤형 서비스 제안

2022 bigcontest

빅콘치즈

황영우 hyw612@naver.com (조장)
정정룡 fyd0277@naver.com
정성훈 sung121011@naver.com
김평진 kimpung127@naver.com

finda

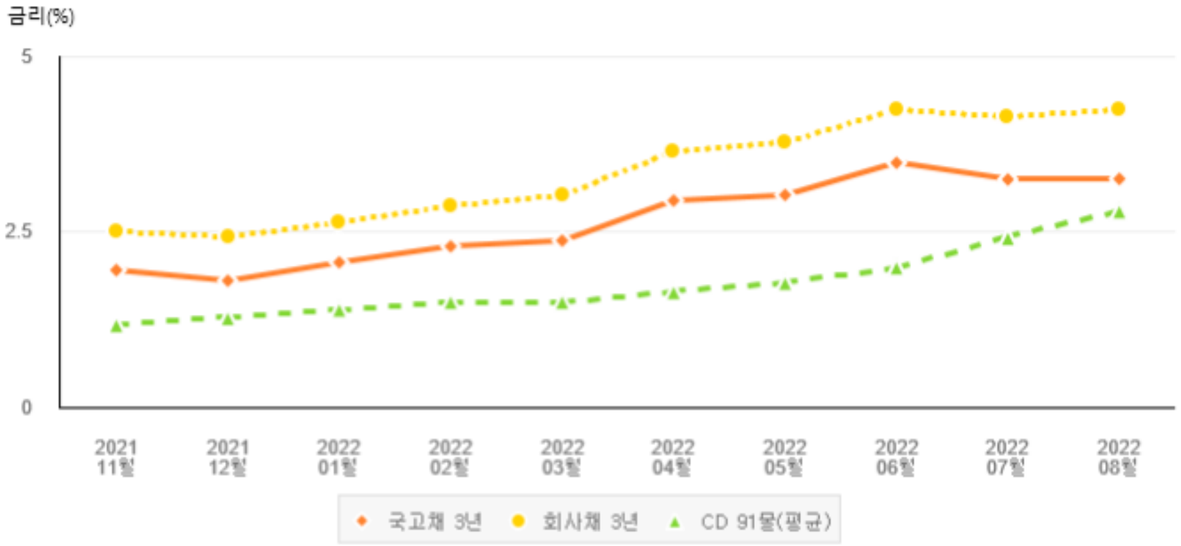
목차 CONTENTS

00. 분석 배경

「현 대출 환경」

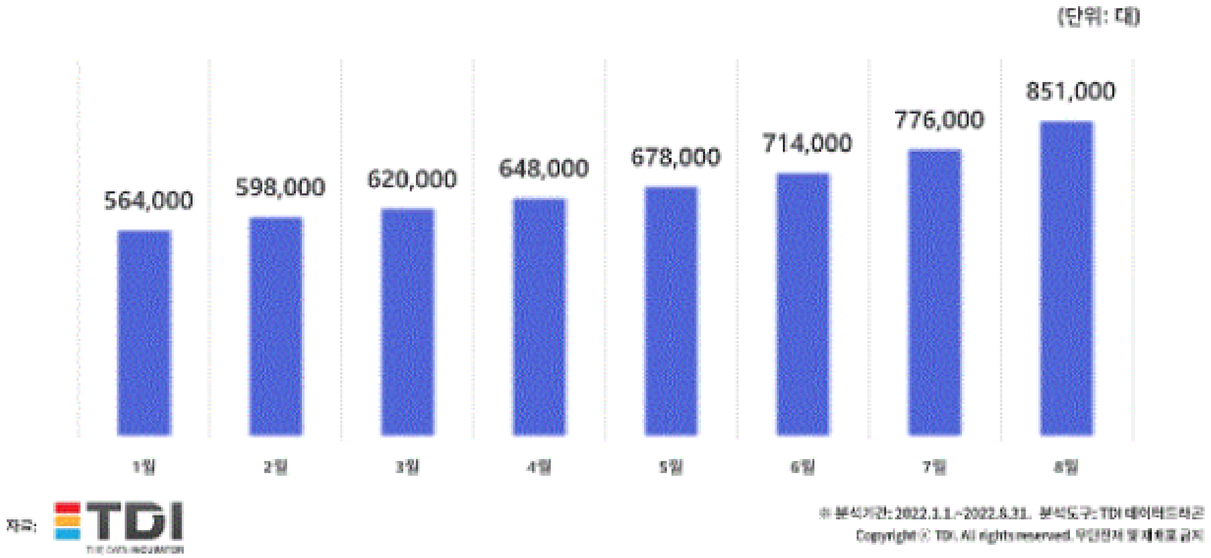
- 금리 상승기에 따라 핀다 앱 사용자가 늘고 있는 추세

˙주요 금리 추이 현황



출처 : 한국은행 경제통계시스템 > 4. 금리

˙핀다 기기설치수 추이



00. 분석 배경

「핀다 대표 서비스」

- 비교대출과 대출관리, 자동차 금융서비스를 주 사업으로 하는 플랫폼

고객에게 더 낮은 금리의 상품을 제공하는'대환보장제'
서비스 업계 최초로 출시

출처 : 핀다

'나의 대출관리' 서비스를 통해 신용도 조회, 기대출금액 관리 등
고객의 편의를 위한 서비스를 제공

01. 데이터 전처리 - 데이터 탐색

「loan_result」

- 대출상품결과 테이블

"Target"

신청서 번호	한도 조회 일시	금융사 번호	상품 번호	승인 한도	승인 금리	신청 여부
application_id	loanapply_insert_time	bank_id	product_id	loan_limit	loan_rate	is_applied
1748340	2022-06-07 13:05:41	7	191	42000000	13.6	NA
1748340	2022-06-07 13:05:41	25	169	24000000	17.9	NA
1748340	2022-06-07 13:05:41	2	7	24000000	18.5	NA
1748340	2022-06-07 13:05:41	4	268	59000000	10.8	NA
1748340	2022-06-07 13:05:41	11	168	5000000	16.4	NA

- loanapply_insert_time : 2022년 3월 ~ 2022년 6월

- loan_limit : 백만 단위에서 반올림

- is_applied : 0,1 => train data
  NA => test data

01. 데이터 전처리 - 데이터 탐색

「log_data」

- 유저로그 데이터

유저 번호	행동명	행동 일시	기기 종류	앱 버전	일 코드
user_id	event	timestamp	mp_os	mp_app_version	date_cd
576409	StartLoanApply	2022/03/25 11:12	Android	3.8.2	2022-03-25
576409	ViewLoanApplyIntro	2022/03/25 11:12	Android	3.8.2	2022-03-25
72878	EndLoanApply	2022/03/25 11:14	Android	3.8.4	2022-03-25
645317	OpenApp	2022/03/25 11:15	iOS	3.6.1	2022-03-25
645317	UseLoanManage	2022/03/25 11:15	iOS	3.6.1	2022-03-25

「user_spec」

- 유저스펙 테이블

신청서 번호	유저 번호	생년월일	성별	생성일시	신용점수	연 소득	근로 형태	입사연월	고용형태	주거 소유 형태
application_id	user_id	birth_year	gender	insert_time	credit_score	yearly_income	income_type	company_enter_month	employment_type	houseown_type
1249046	118218	1985	1	2022/06/07 6:28	660	1.08E+08	PRIVATEBUSINESS	20151101	기타	자가
954900	553686	1968	1	2022/06/07 14:29	870	3.00E+07	PRIVATEBUSINESS	20070201	정규직	기타가족소유
137274	59516	1997	1	2022/06/07 21:40	710	3.00E+07	FREELANCER	20210901	기타	기타가족소유
1570936	167320	1989	1	2022/06/07 9:40	820	6.20E+07	EARNEDINCOME	20170101	정규직	자가
967833	33400	2000	1	2022/06/07 8:55	630	3.60E+07	EARNEDINCOME	20210901	정규직	기타가족소유

유저 기본 정보

유저 스펙 정보

「user_spec」

- 유저스펙 테이블

대출 희망 금액	대출 목적	개인회생자 여부	개인회생자 납입 완료 여부	기대출 수	기대출 금액
desired_amount	purpose	personal_rehabilitation_yn	personal_rehabilitation_complete_yn	existing_loan_cnt	existing_loan_amt
1.00E+06	기타	0	NA	4	1.62E+08
3.00E+07	대환대출	0	NA	1	2.70E+07
1.00E+07	생활비	0	NA	5	1.50E+07
2.00E+06	생활비	0	NA	7	3.44E+08
5.00E+06	생활비	0	0	1	1.60E+07

희망 대출

유저 대출 현황

01. 데이터 전처리 - 변수 변환? 및 NA 대체

「purpose」

■ 대출 목적

purpose
Business
Buycar
ETC
자동차구입
전월세보증금

→

purpose
사업자금
자동차구입
기타
자동차구입
전월세보증금

영문으로 된 관측값과 한글로 된 관측값이 **일맥상통**(?)
하기 때문에 소수인 **영문 관측값을 한글로** 변경

「yearly_income」

■ 연소득

연수입 변수의 결측치 값은 **총 1개** 존재

같은 user_id에 있는 값으로 대체

user_id	yearly_income
670502	NA
670502	0

→

user_id	yearly_income
670502	0
670502	0

「loan_rate, loan_limit」

■ 승인금리, 승인한도

loan_rate와 loan_limit에서의 결측치는 항상 같이 존재한다.

loan_rate와 loan_limit에서의 결측치는 **금융사에서 보내주지 않은 값**이기 때문에 제외하고 진행

01. 데이터 전처리 - 변수 변환? 및 NA 대체

「gender」

user_id	gender
3341	NA
3341	1
3341	1
49072	0
49072	NA
877983	NA
877983	NA

같은 user_id에서 성별에 결측값과 함께
0 또는 1이 존재할 경우 그 값으로 대체

결측값 대체가 불가능 할 경우,
'알 수 없음'으로 변경

user_id	gender
3341	1
3341	1
3341	1
49072	0
49072	0
877983	알 수 없음
877983	알 수 없음

01. 데이터 전처리 - 변수 변환? 및 NA 대체

「existing_loan_cnt, amt」

existing_loan_amt	existing_loan_cnt	obs 수
1	0	3495
1	N	87355
1	NA	78753
N	0	375
N	N	652462
N	NA	0
NA	0	0
NA	N	0
NA	NA	146287

▪ N = 2 이상의 수

▪ 기대출 수= NA

기대출 금액의 단위가 백만 이므로
(기대출이 없는 것이 아닌) 50만원 미만
의 기대출 금액으로 판단

▪ 기대출 금액 = 0

0인 관측값이 존재하지 않으므로
NA = 0으로 판단

▪ 기대출 금액 = NA

금융사에서의 정보제공 누락
NA = 0으로 판단

01. 데이터 전처리 - 변수 변환? 및 NA 대체

「company_enter_month」

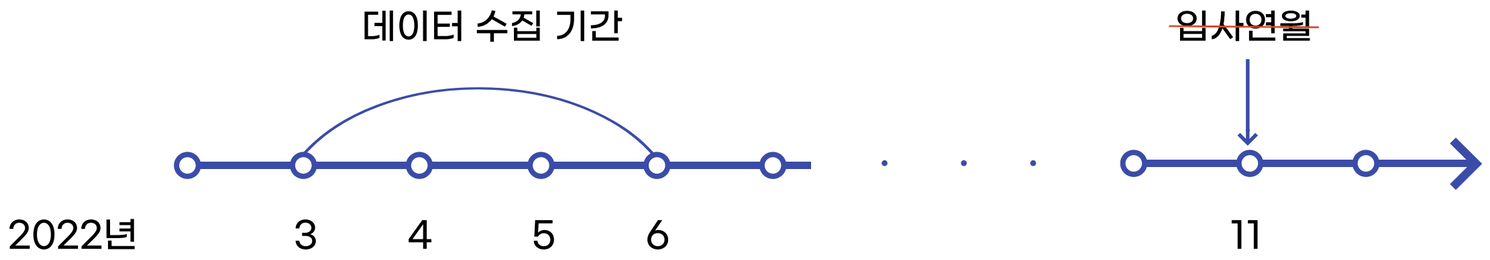
■ 입사연월

company_enter_month
20200501
20200201
201210
20160601
202110

→

company_enter_month
202005
202002
201210
201606
202110

입사 **일**에 대한 정보가 있는 것과 없는 것이
혼합되어있어 **연도, 월**까지 표기하는 것으로 통일



- 입사연월이 데이터 수집 기간 이후인것은 불가능
  같은 이용자에 '202201' 이 존재하므로
  오기로 판단하여 대체

(입사연월 - 출생연도) ≥ 20 ⇒ NA

- 미성년자의 경우 취직이 불가능 하므로
  입사연월과 출생연도의 비교를 통해
  결측치로 처리(변경)

01. 데이터 전처리 - 파생변수 생성 in user_spec

「age」

■ 출생연도

birth_year
1979
2000
1985
1993
1971

출생연도를 모델링 및 분석에 용이하게
나이로 변환

■ 나이

age		
44	.....	40 ~ 49세
23	.....	20 ~ 29세
56	.....	50 ~ 59세
81	.....	80세 ~
NA	.....	NA

각 나이를 연령대로 **범주화**
80세 이상은 초고령, NA는 알 수 없음으로 범주화 진행

age
40대
20대
50대
초고령
알 수 없음

01. 데이터 전처리 - 파생변수 생성 in user_spec

「*rehabilitation*」

personal_rehabilitation_yn	personal_rehabilitation_complete_yn	obs 수
0	0	119897
0	1	3
0	NA	426217
1	0	4223
1	1	635
NA	NA	417752

→

rehabilitation
해당없음
해당없음
해당없음
납입중
납입완료
알 수 없음

01. 데이터 전처리 - 파생변수 생성 in user_spec

「work_year」

■ 입사연월

company_enter_month
202005
202202
201210
NA
NA

입사연월을 연도만 추출해 그 값에
현재 연도를 뺀 경력으로 변환

→

work_year		
2	.....	N년차
0	.....	0년차
10	.....	N년차
NA	.....	NA
NA	.....	NA

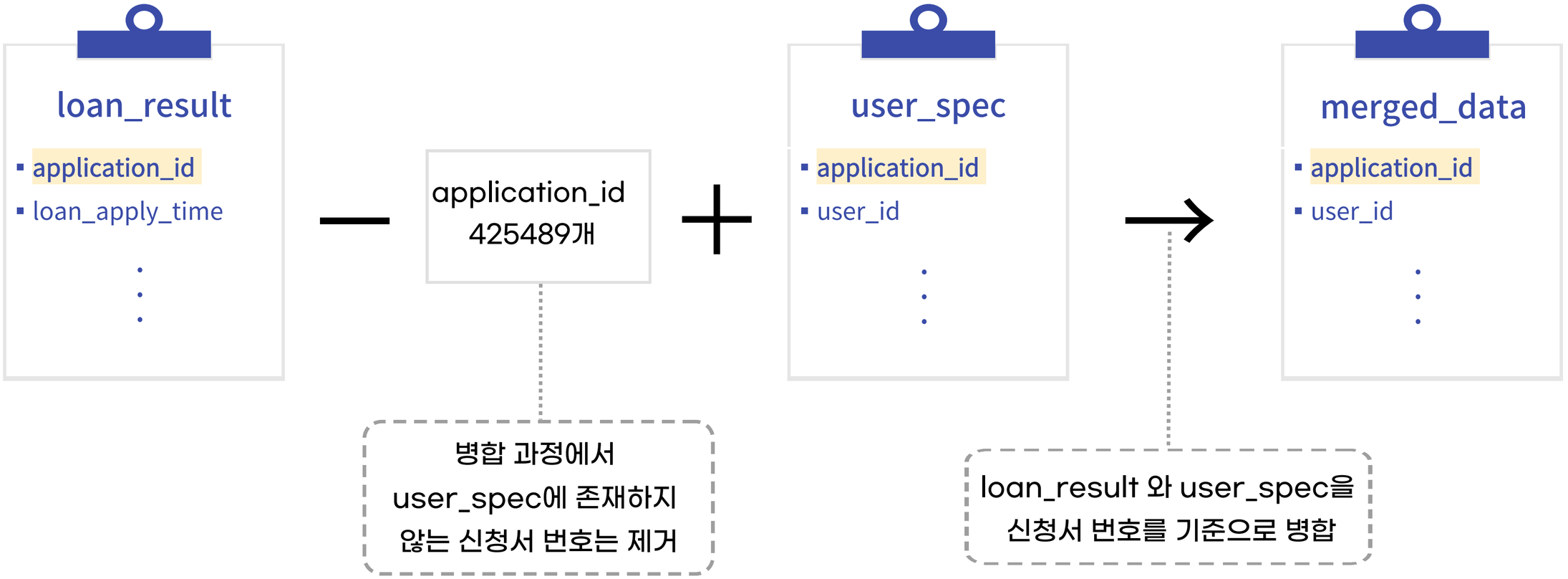
경력을 0년차, N년차, NA로 구분 무직의 경우 입사연월은
결측치이므로 NA = 기타로 범주화

→

age
N년차
0년차
N년차
기타
기타

01. 데이터 전처리 - 파생변수 생성 in merged_data

「데이터 병합」



01. 데이터 전처리 - 파생변수 생성

「rate_rank」

■ 한 신청서 내 금리 순위

application_id	loan_rate
954900	6.9
954900	6.9
954900	7.5
954900	10.6
954900	10.6

+

rate_rank
1
1
3
4
4

각 신청서 내의 상품들을
승인금리 기준으로 **순위화**

「product_per_app」

■ 신청서 별 상품수

index	application_id
1	954900
2	954900
⋮	⋮
11	954900
12	954900

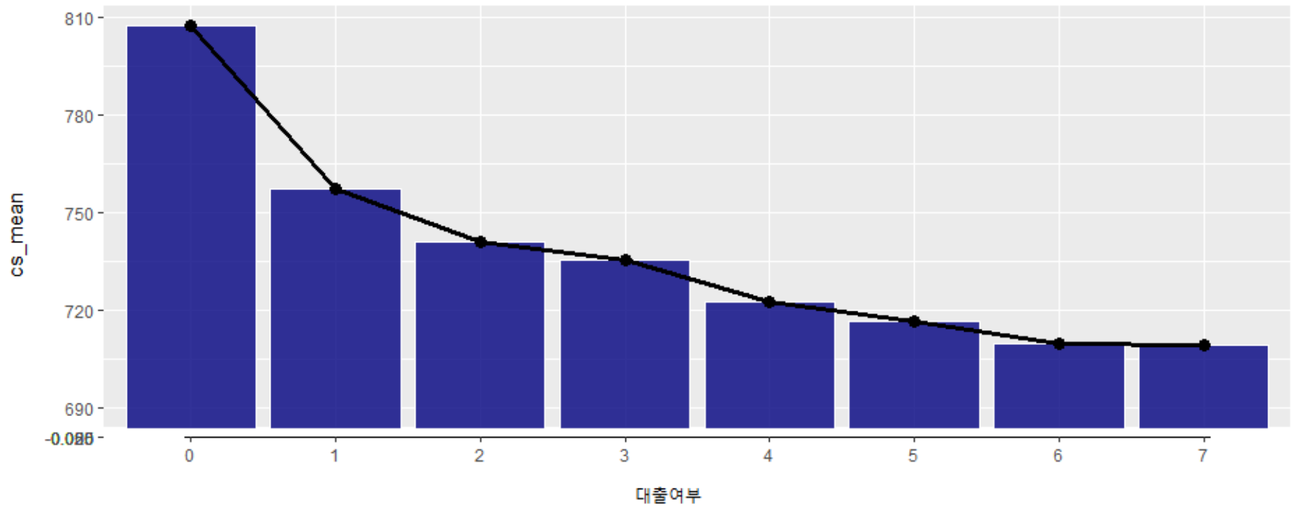
+

product_per_app
12
12
⋮
12
12

각 신청서 별로 조회되는 **총 상품수**를
나타내는 변수 생성

01. 데이터 전처리 - 파생변수 생성

「*loan_per*」



이용자들을 총 대출 신청 횟수로 그룹지어 각 그룹별 신용점수의 평균을 비교한 결과, 대출 횟수가 4회가 되는 이후부터는 경사가 완만하다고 판단하여 4회 이상은 '대출신청 다수'로 범주화

총 대출 횟수 (대출 여부)		loan_per
0	→	대출 안함
1	→	대출 처음
2	→	대출 2회
3	→	대출 3회
4	→	대출 신청 다수

01. 데이터 전처리 - 파생변수 생성

「*user_spec 최종 변수*」

application_id	user_id	~~birth_year~~ age
1614045	860410	40대
954900	553686	50대
137274	59516	20대
1570936	167320	30대
967833	33400	20대

. . .

~~company_enter_month~~ work_year
기타
N년차
N년차
N년차
N년차

. . .

~~personal_rehabilitation_complete_yn~~ ~~personal_rehabilitation_yn~~ rehabilitation
해당없음
해당없음
해당없음
해당없음
해당없음

+

rate_rank	product_per_app	loan_per
해당없음	1	대출
해당없음	12	대출안함
해당없음	11	대출안함
해당없음	25	대출2회
해당없음	2	대출3회

01. 데이터 전처리 - 파생변수 생성

「*log_data*」

user_id	event	timestamp
576409	StartLoanApply	2022/03/25 11:12
576409	ViewLoanApplyIntro	2022/03/25 11:12
72878	EndLoanApply	2022/03/25 11:14
645317	OpenApp	2022/03/25 11:15
645317	UseLoanManage	2022/03/25 11:15

— 6월(test) 데이터

— mp_os, mp_app_version, date_cd

log_data에서 test데이터인 6월 데이터와 이후 분석에 쓰지 않을 변수 제거

+

loan_per	product_per_app	application_n
대출 2회	9	2
대출 2회	11	2
대출 안함	3	4
대출 신청 다수	4	7
대출 신청 다수	6	7

↓

이용자별 한도 조회 수를 신청서 수를 통해 파생변수 생성

01. 데이터 전처리 - 파생변수 생성

「*log_data*」

user_id	event
576409	StartLoanApply
576409	ViewLoanApplyIntro
72878	EndLoanApply
645317	OpenApp
645317	UseLoanManage

user_id	StartLoan Apply	ViewLoan ApplyIntro
576409	33	30
576409	33	30
72878	5	4
645317	25	38
645317	25	38

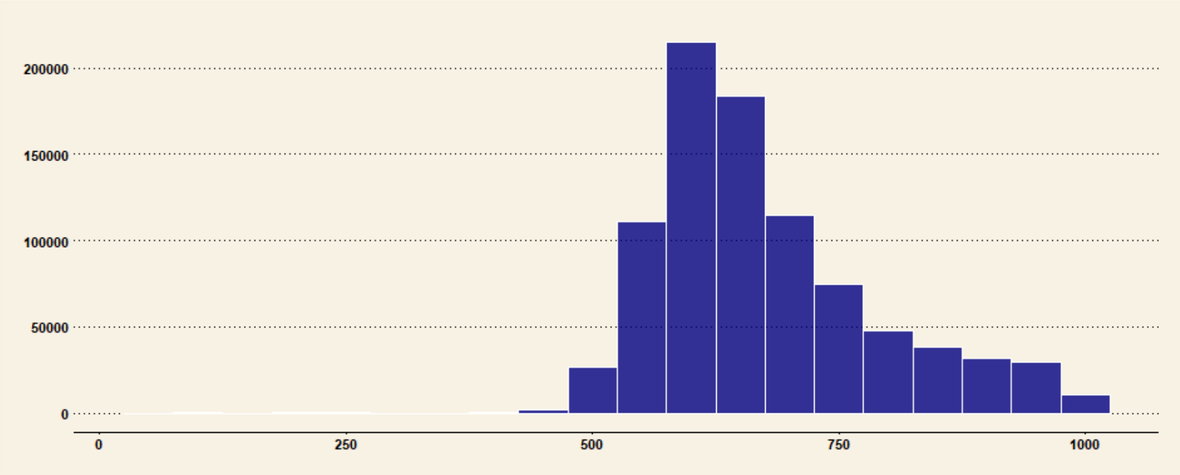
user_id	StartLoan Apply	ViewLoan ApplyIntro
576409	33	30
72878	5	4
645317	25	38

이용자마다 핀다 앱 내에서의 각 행동 별 총 누적수를 파생변수로 생성하기 위해 pivot_wider를 실시

02. 신용점수 결측치 대체 - 파생변수 생성

「결측치 대체 배경」

신용점수 분포



신용점수의 결측치는 전산상 오류로 인해 불러오지 못한 결측치로
MCAR(완전무작위결측)에 해당

결측치 존재하는 obs 제거 ✕

- NA : 81768개
  데이터 손실을 고려하여 진행 X

단순 통계값으로 대체 ✕

- 추정량 표준오차의 과소추정
  및 편향 등으로 인한 왜곡 발생

단일 대체법

1. Lasso 회귀
2. Knn-imputation
3. Missforest

02. 신용점수 결측치 대체 - 모델 소개

「Lasso 회귀」

: 결측치가 있는 변수를 종속변수로,
나머지 변수들을 독립변수로 한 회귀식을 통해
결측치를 예측하는 기법

■ L1-norm 패널티를 가진 선형 회귀 방법

$$MSE + penalty$$

$$= MSE + \alpha L_1 - norm$$

$$= \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y_i})^2 + \alpha \sum_{j=1}^{m}|w_j|$$

「KNN-imputation」

: 기존 knn과 같이 인접한 데이터의 최빈값 혹은 통계값으로 결측치를
대체하는 기법

: 거리지표로 Euclidean distance 사용

: 결측치가 있는 관측치 간 거리는 결측치를
제외한 거리에 가중치를 주어 계산

weight = total coordianates(전체변수) /no-NA coordianates(결측치가 없는 변수)

d^2= weight * squared distance of non - NA coordianates

02. 신용점수 결측치 대체 - 모델 소개

「*missforest*」

: random forest 알고리즘의 예측값으로 결측치를 대체하는 기법

: knn에 비해 계산량이 적으며 변수들의 scale을 고려하지 않아도 됨

■ 알고리즘 과정

1) 결측치를 중위값이나 최빈값으로 임의대체

2) 결측치 비율이 낮은 변수부터 결측치가 없는 데이터를 랜덤포레스트로 학습하여 결측치 예측 진행

3) 결측치가 채워진 데이터로 2) 과정과 동일한 학습과정으로 결측치를 다시 한 번 예측

4) 모든 iter가 다 돌아가거나 결측값의 변화량이 일정 값으로 수렴하면 종료

02. 신용점수 결측치 대체 - 파생변수 생성

「하이퍼 파라미터 선정」

: Lasso와 Missforest는 Grid Search를 통해
최적의 하이퍼 파라미터 조합을 선정
Knn은 k값을 조정하여 진행

모델	hyperparameter
Lasso	CV = 3
Knn	K = {5, 9, 11}
Missforest	CV = 3, MAx_iter={5} n_estimators={200}

「성능 비교」

: 결측치 대체 성능을 비교하기 위해 결측값을 가진 행을 제외한
886,959 행에서 임의로 신용점수에 결측값을 부여한 후 대체한 결과와
RMSE 비교

모델	최저 RMSE
Lasso	78.31
Knn	81.4
Missforest	73.2

⇒ Missforest로 모델 선정하여 결측치 대체

모델링

「1. RandomForest」

: 결정트리 기반 배깅 모델

: 변수 간 차이를 쉽게 볼 수 있어 중요한 변수를 선정하기 용이
결정트리에 비해 정확도가 높으며 과적합 방지에 유리

「2. Logistic Regression + Lasso」

: 이진분류 모델로 L1 규제를 적용하여 변수의 영향력을 없애 설명력을 높이면서 동시에 변수선택의 효과도 줄 수 있는 비선형회귀모형

: 일반적인 머신러닝 모델에 비해 해석이 용이하며,
중요도가 다른 변수들에서 중요한 변수들만을
뽑아내기 위해 L1 규제를 사용

「3. CatBoost」

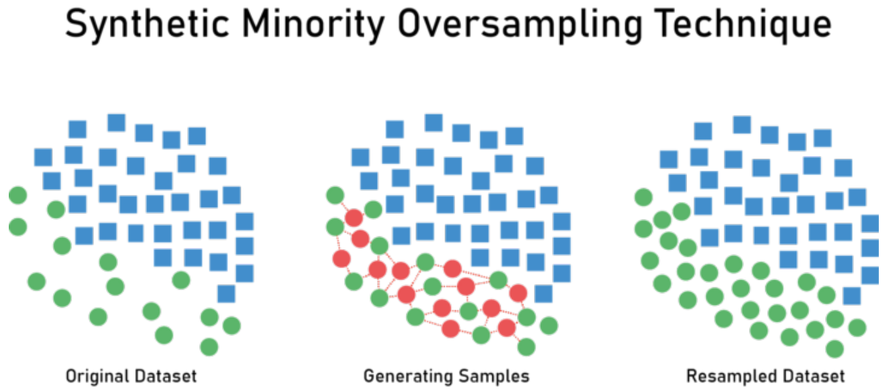
: 부스팅 모델로 Ordered boosting, mean encoding등의 기법 사용

: 범주형 변수가 많을 때 연산속도와 성능이 뛰어남

모델링 - 데이터 불균형 처리

「SMOTE」
(Synthetic Minority Over-sampling Technique)

KNN기반 오버샘플링기법으로
임의의 소수 클래스 데이터 사이에 새로운 데이터를 생성하는 방법.
데이터 생성을 위해 KNN 알고리즘을 활용하여
소수 클래스의 데이터 중 특정 표본에 가장 가까운 k개의 이웃 벡터를 선정한 뒤 ,
기준 벡터와 선정한 벡터 사이를 선분으로 이어 선분 위의 임의의 점을 새로운 샘플로 만드는 기법.



「Class_weight」

사이킷런 내 모델들에 적용할 수 있는 하이퍼파라미터로 불균형 데이터 내 적은 집단에 대해 가중치를 주어 영향력을 높이는 방식. 이진 분류에서는 Class 별 샘플의 역수가 가중치가 된다.

$$w_{n,c} = \frac{1}{Number\ of\ Samples\ in\ Class\ c}$$

「*하이퍼 파라미터 선정*」

: GridSearchCV를 사용하여
하이퍼 파라미터 선정

SMOTE

Model	Hyper Parameter
RandomForest	max_depth = 21, min_samples_leaf = 9, n_estimators = 190
Logistic	C = 10
CatBoost	iteration =1000, l2_leaf_reg = 0.07, leaf_estimation_iter = 15, depth = 16

Class_weight

Model	Hyper Parameter
RandomForest	max_depth = 14, min_samples_leaf = 4, n_estimators = 150
Logistic	C = 1
CatBoost	iteration =1000, l2_leaf_reg = 0.07, leaf_estimation_iter = 15, depth = 16

모델링 - 예측 결과 및 최종 모델 선정

「*예측 결과 및 최종 모델 선정*」

: 각 cv 별 F1 score의 평균 점수를 비교하여 최종 모델 선정

SMOTE

모델	cv1,2,3 평균점수
RandomForest	0.450
Logistic	0.276
CatBoost	0.40

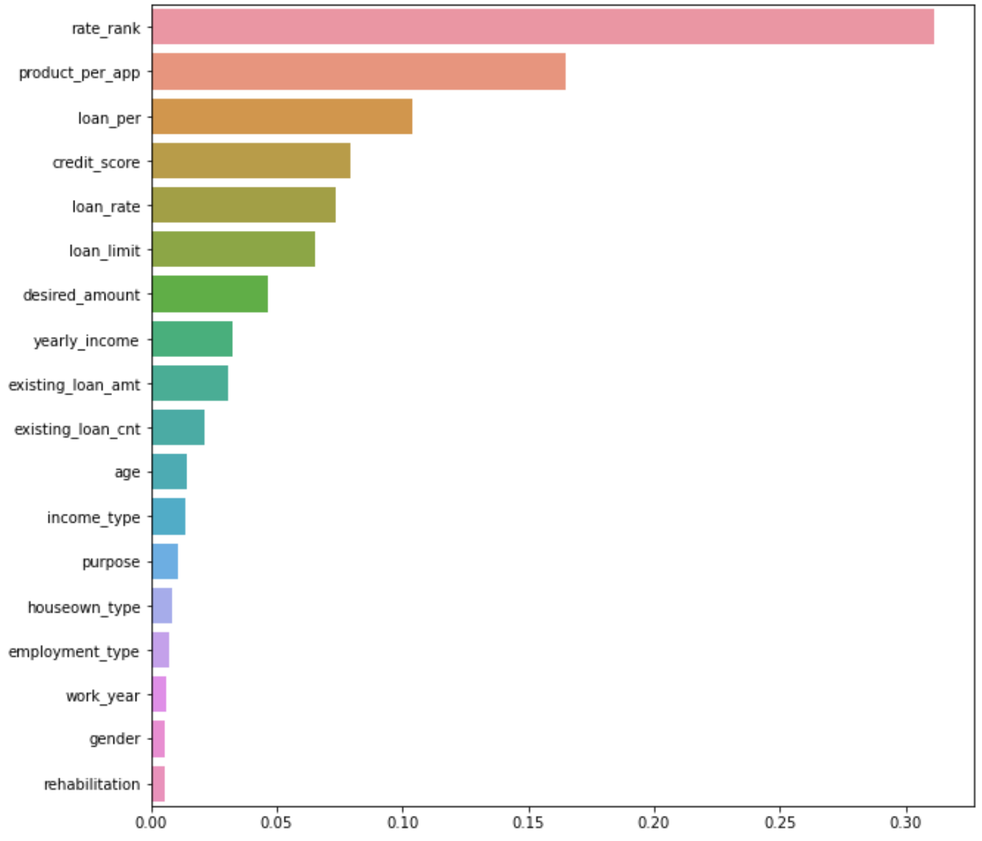
Class_weight

모델	cv1,2,3 평균점수
RandomForest	0.452
Logistic	0.16
CatBoost	0.43

최종 모델을 "RandomForest" 로
선정하여 예측 모델 기반 EDA 진행

모델기반 EDA

「Feature importance」



「Permutation Feature importance」

Weight	Feature
0.0136 ± 0.0011	loan_limit
0.0021 ± 0.0017	rate_rank
0.0016 ± 0.0013	loan_per
0.0005 ± 0.0006	age
0.0004 ± 0.0003	work_year
0.0004 ± 0.0003	gender
0.0003 ± 0.0005	rehabilitation
0.0003 ± 0.0004	houseown_type
-0.0001 ± 0.0002	income_type
-0.0003 ± 0.0005	existing_loan_cnt
-0.0004 ± 0.0004	employment_type
-0.0007 ± 0.0005	yearly_income
-0.0008 ± 0.0003	existing_loan_amt
-0.0011 ± 0.0004	purpose
-0.0025 ± 0.0012	desired_amount
-0.0037 ± 0.0006	loan_rate
-0.0085 ± 0.0004	credit_score
-0.0139 ± 0.0006	product_per_app

군집화

군집화 알고리즘 선정 배경

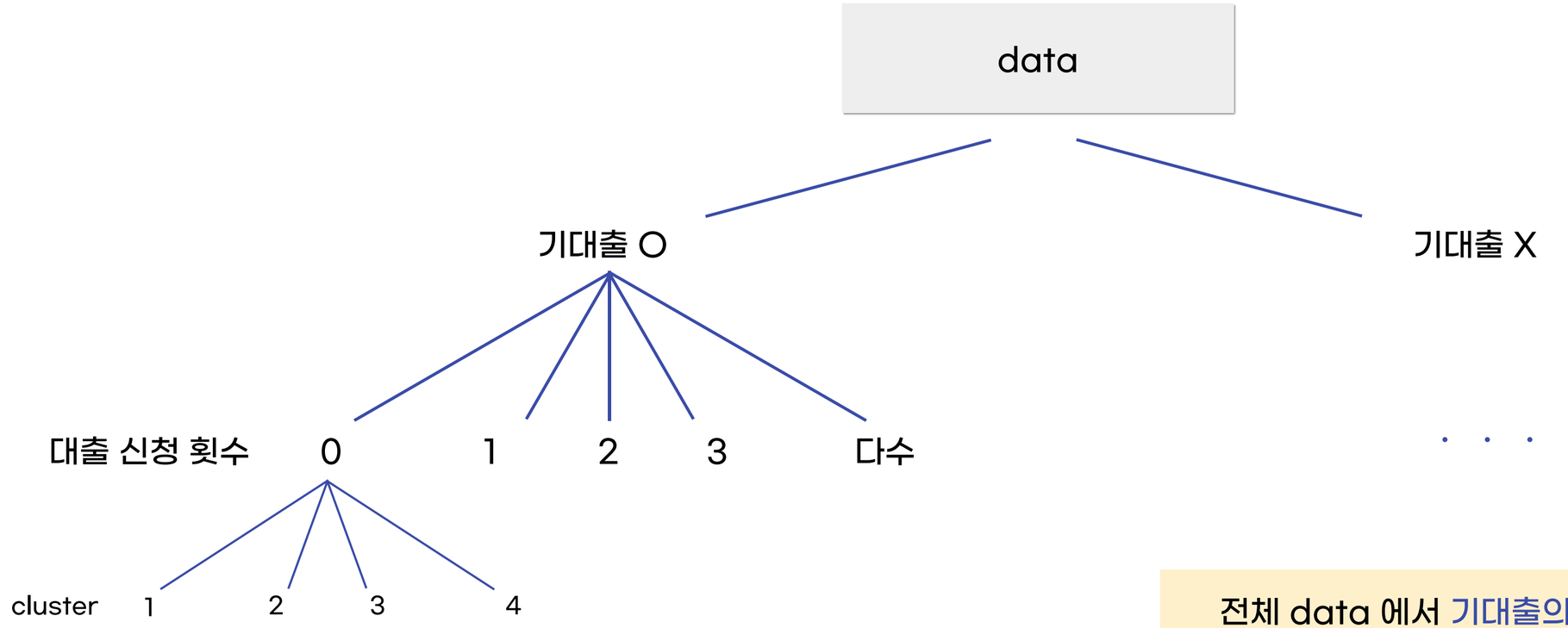
log_data 내 event들에 대해 유저 별 누적 로그 데이터로 였으며,
로그 별 상관계수와 특정 행동이 발생했을 때 연쇄되어 일어나는 행동을 본 결과
특정 로그들이 가지는 패턴을 확인할 수 있었다.
따라서 유사한 로그들을 하나의 패턴으로 두고 각각의 평균값으로 변환하여 군집화를 진행하였다

군집화 알고리즘 소개

과도하게 많은 로그를 처리하기 위해 99 quantile로 이상치를 제거 하였으며 패턴 및 유저 간 scale 차이를 고려하여 지수변환 및 정규화를 동시에 진행하였으며, 군집화 기법은 DBSCAN 으로 진행하였다.

대출신청 횟수에 따른 고객 세분화를 위해 대출신청 0,1,2,3,n번의 횟수로 구분 하였으며, 기대출 유무에 따라 계산기 사용이 제한 되므로 기대출 유무로 집단을 나눈 뒤 패턴에 따른 군집화를 진행하였다.

군집화



전체 data 에서 기대출의 유무로 이분화 후, loan_per 변수인 대출 신청 횟수로 그룹 세분화

그 후 log 데이터의 행동 패턴의 누적 횟수를 변수로 군집화 실시

군집화

「기대출 존재 O」

* 표 내의 값은 각 cluster의 평균

대출 신청 : 0

	0	1	2	3	4
GetCreditInfo	0.748	0.786	0.434	0.506	0.497
UseLoanManage	1.031	0.502	0.463	0.555	0.540
UsePrepayCalc	0.829	0.000	0.000	1.018	0.915
UseDSRCalc	0.487	0.000	1.024	0.000	0.953
start	0.996	0.646	0.315	0.347	0.328
loanApply	0.277	0.663	0.440	0.471	0.413

대출 신청 : 1

	1	2	3
GetCreditInfo	0.547	0.796	0.457
UseLoanManage	0.571	0.555	0.463
UsePrepayCalc	1.017	0.000	0.318
UseDSRCalc	0.000	0.000	1.010
start	0.404	0.730	0.346
loanApply	0.525	0.785	0.461

군집화

「기대출 존재 O」

* 표 내의 값은 각 cluster의 평균

대출 신청 : 2

	1	2	3
GetCreditInfo	0.785	0.466	0.544
UseLoanManage	0.617	0.480	0.575
UsePrepayCalc	0.000	0.367	1.019
UseDSRCalc	0.000	1.014	0.000
start	0.804	0.343	0.399
loanApply	0.866	0.452	0.532

대출 신청 : 3

	0	1	2	3
GetCreditInfo	0.00	0.782	0.594	0.565
UseLoanManage	0.00	0.637	0.626	0.571
UsePrepayCalc	0.00	0.000	1.019	0.346
UseDSRCalc	0.00	0.000	0.000	1.000
start	1.03	0.837	0.423	0.422
loanApply	0.00	0.907	0.565	0.539

군집화

「기대출 존재 O」

* 표 내의 값은 각 cluster의 평균

대출 신청 : 다수

	0	1	2	3
GetCreditInfo	0.515	0.761	0.565	0.499
UseLoanManage	0.360	0.665	0.594	0.513
UsePrepayCalc	0.146	0.000	0.975	0.000
UseDSRCalc	0.147	0.000	0.250	1.025
start	0.828	0.892	0.451	0.431
loanApply	0.255	0.939	0.577	0.559

군집화

「기대출 존재 X」

대출 신청 : 0

	0	1	2	3	4
GetCreditInfo	0.000	0.394	0.642	0.654	0.582
UseLoanManage	0.515	0.122	0.636	0.649	0.563
UsePrepayCalc	0.464	0.000	1.010	0.932	0.000
UseDSRCalc	0.515	0.000	0.000	0.951	1.025
start	0.507	0.601	0.330	0.362	0.353
loanApply	0.577	0.778	0.528	0.504	0.538

대출 신청 : 1

	1	2	3	4
GetCreditInfo	0.334	0.649	0.571	0.586
UseLoanManage	0.122	0.642	0.552	0.554
UsePrepayCalc	0.000	1.005	0.000	0.858
UseDSRCalc	0.000	0.000	1.018	1.006
start	0.710	0.451	0.428	0.410
loanApply	0.821	0.554	0.499	0.487

군집화

「기대출 존재 X」

대출 신청 : 2

	1	2	3
GetCreditInfo	0.306	0.516	0.601
UseLoanManage	0.136	0.495	0.620
UsePrepayCalc	0.000	0.306	1.013
UseDSRCalc	0.000	1.017	0.006
start	0.789	0.467	0.499
loanApply	0.906	0.559	0.572

대출 신청 : 3

	1	2	3
GetCreditInfo	0.370	0.646	0.552
UseLoanManage	0.184	0.630	0.534
UsePrepayCalc	0.012	0.870	0.000
UseDSRCalc	0.000	1.024	1.030
start	0.829	0.482	0.398
loanApply	0.882	0.549	0.493

군집화

「기대출 존재 X」

대출 신청 : 다수

	1	2	3
GetCreditInfo	0.402	0.631	0.714
UseLoanManage	0.231	0.592	0.705
UsePrepayCalc	0.000	0.302	0.978
UseDSRCalc	0.000	1.009	0.004
start	0.874	0.540	0.622
loanApply	0.917	0.598	0.710

감사합니다