2022 빅콘테스트 퓨처스 부문

앱 사용성 데이터를 통한 대출신청 예측분석

수퍼루키

팀장 장우솔 (jus6886@naver.com)
팀원 김예은 (rladpdms921@naver.com)
팀원 박선홍 (psh56750855@gmail.com)
팀원 신재욱 (jwshin0908@gmail.com)
팀원 우아라 (woo.ara00@gmail.com)

CONTENTS

I

문제 배경

Ⅰ. 문제 배경

대출 중개 플랫폼, 핀다(finda)

대출조건을 조회하여 신청하고 향후 상환까지 이루어지는 대출의 모든 과정을 포괄하는 대출 통합 관리 서비스 제공

기업의 방향성 : 혁신적인 핀테크 기술과 빅데이터를 통해 개인이 쉽고 편리한 금융생활을 영위하게 하는 것

⟶ 핀다 앱을 사용한 수많은 고객 데이터를 활용한다면 더 많은 사람들에게 더 나은 서비스를 제공할 수 있다는 아이디어에서 시작

[출처 : finda]

finda

[출처 : finda 홈페이지]

Ⅰ. 문제 배경

대출의 종류

기관, 지급 방식, 금리 유형, 대출 상환 방식 등 다양한 기준에 따라 나눌 수 있지만, 가장 대표적으로 신용대출과 담보대출 두 가지가 존재

신용대출

개인의 신용도를 근거로 돈을 빌려주는 대출

개인의 신용등급과 소득 유무, 재직상태, 기존 대출 현황, 연체 이력, 카드 사용률 등 여러 가지 요건을 고려

대출 한도 : 본인의 연소득에 기반하여 결정하며 은행에 따라 상이

대출 금리 : 본인의 신용도와 은행 거래 내역에 따라 산출하며 은행에 따라 상이

담보대출

부동산, 토지 등을 담보로 돈을 빌려주는 대출

대출 한도 : 담보물의 가치에 따라 결정하며 은행에 따라 상이

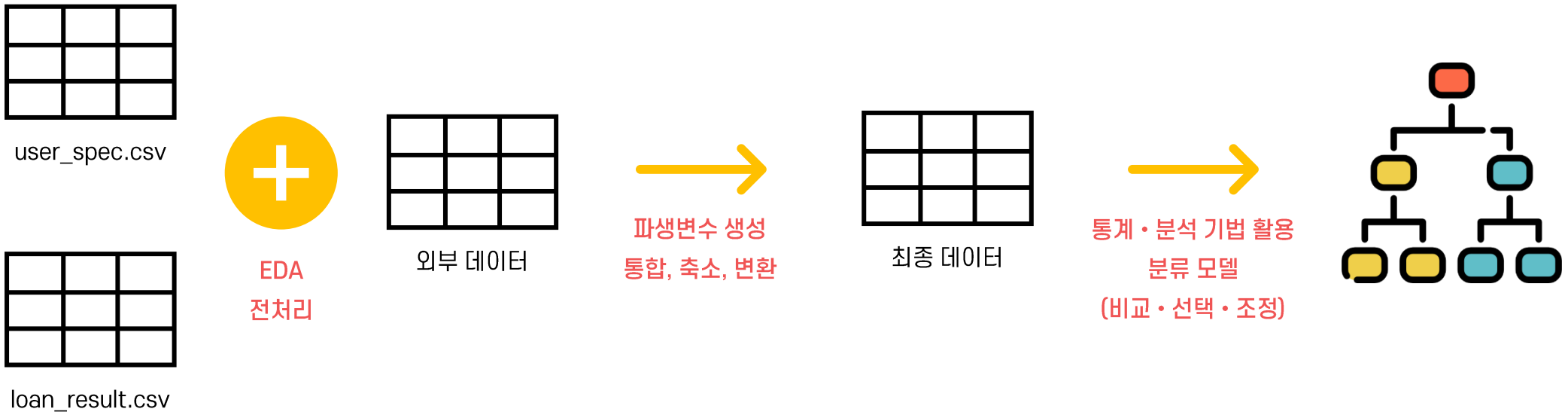
대출 금리 : 본인의 신용도와 은행 거래 내역에 따라 산출하며 은행에 따라 상이

Ⅰ. 문제 배경

핀다 홈화면 진입 고객 중 특정기간 안에 대출신청 고객 예측

과거의 대출신청 고객 데이터를 기반으로 새로운 고객들의 대출신청 여부를 예측하여 수요 예측 가능

모델에 기반해 대출신청 유무에 따른 고객들의 패턴을 파악한다면 고객에 맞는 대출을 세분화하여 추천함으로써 고객 니즈를 충족 가능

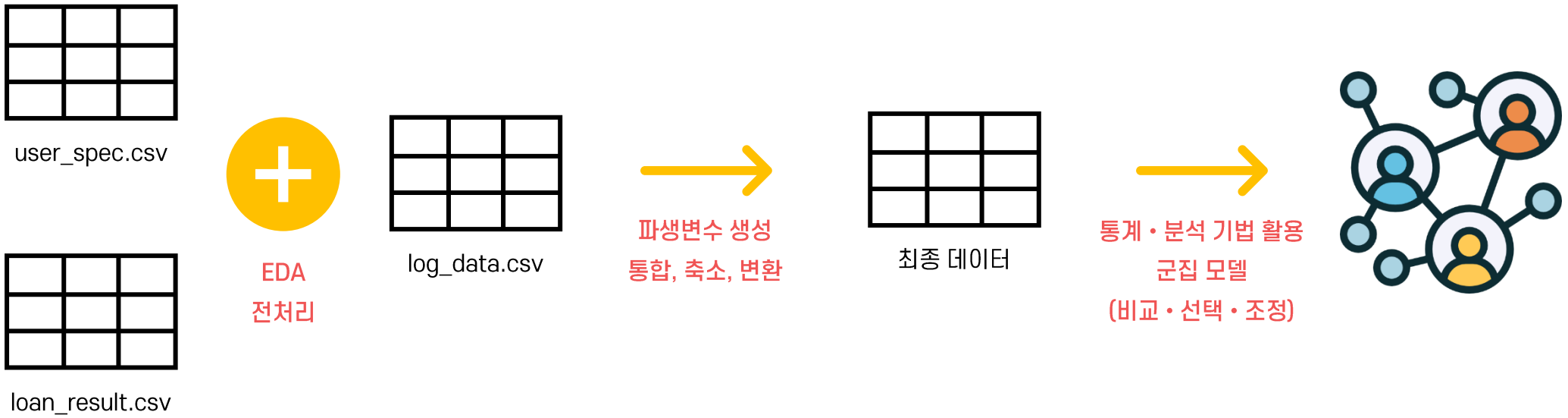


Ⅰ. 문제 배경

핀다 홈화면 진입 고객의 모델 기반 고객 군집 분석 & 군집 별 서비스 메시지 제안

핀다 앱의 사용자 로그를 기반으로 군집 분석을 수행하여 맞춤형 앱 서비스를 제공해 사용자 경험 향상에 도움

고객 세분화를 앱 사용성 향상뿐만 아니라 PR 과 같은 핀다 서비스의 다양한 분야의 성장에 활용 가능



II

데이터 탐색

기존 데이터 - 가명화된 데이터(2022년 3~6월 데이터)

① 유저스펙테이블(user_spec)

- 사용자에 대한 정보를 담은 데이터
- 한 유저 번호(user_id)당 대출 신청서(application_id)는 여러 번 생성 가능

컬럼ID	컬럼명	example 1	example 2	example 3
application_id	신청서 번호	1249046	954900	137274
user_id	유저 번호	118218	553686	59516
birth_year	유저 생년월일	1985.0	1968.0	1997.0
gender	유저 성별	1.0	1.0	1.0
insert_time	생성일시	2022-06-07 06:28:18	2022-06-07 14:29:03	2022-06-07 21:40:22
credit_score	한도조회 당시 유저 신용점수	660.0	870.0	710.0
yearly_income	연소득	108000000.0	30000000.0	30000000.0
income_type	근로형태	PRIVATEBUSINESS	PRIVATEBUSINESS	FREELANCER
company_enter_month	입사연월	20151101.0	20070201.0	20210901.0
employment_type	고용형태	기타	정규직	기타
houseown_type	주거소유형태	기타	기타가족소유	기타가족소유
desired_amount	대출희망금액	1000000.0	30000000.0	30000000.0
purpose	대출 목적	기타	대환대출	생활비
personal_rehabilitation_yn	개인회생자 여부	0.0	0.0	0.0
personal_rehabilitation_complete_yn	개인회생자 납입 완료 여부	NaN	NaN	NaN
existing_loan_cnt	기대출수	4.0	1.0	5.0
existing_loan_amt	기대출금액	162000000.0	27000000.0	15000000.0

기존 데이터 - 가명화된 데이터(2022년 3~6월 데이터)

② 대출상품결과테이블(loan_result)

- 승인된 신청서의 대출 상품 정보에 대한 정보를 담은 데이터
- 사용자는 대출 상품 정보를 보고 신청 여부(is_applied)를 결정

컬럼ID	컬럼명	example 1	example 2	example 3
application_id	신청서 번호	1748340	1748340	1748340
loanapply_insert_time	한도조회 일시	2022-06-07 13:05:41	2022-06-07 13:05:41	2022-06-07 13:05:41
bank_id	금융사 번호	7	25	2
product_id	상품 번호	191	169	7
loan_limit	승인한도	42000000.0	24000000.0	24000000.0
loan_rate	승인금리	13.6	17.9	18.5
is_applied	신청 여부(타겟)	NaN	NaN	NaN

기존 데이터 - 가명화된 데이터(2022년 3~6월 데이터)

③ 유저로그데이터(log_data)

- 사용자의 앱 로그 정보를 담은 데이터
- 각 행동(event)에 대한 행동 일시(timestamp)는 초 단위로 기록

컬럼ID	컬럼명	example 1	example 2	example 3
user_id	유저 번호	576409	576409	72878
event	행동명	StartLoanApply	ViewLoanApplyIntro	EndLoanApply
timestamp	행동일시	2022-03-25 11:12:09	2022-03-25 11:12:09	2022-03-25 11:14:44
mp_os	접속 OS	Android	Android	Android
mp_app_version	앱 버전	3.8.2	3.8.2	3.8.4
date_cd	일 코드	2022-03-25	2022-03-25	2022-03-25

외부 데이터

① 콜금리

- 금융기관 사이에 단기적인 자금 거래가 이루어지는 시장인 콜시장에서 결정되는 금리 / 일별 데이터
- 콜 금리가 오르면 대출의 금리도 오름

② 기준금리

- 현재 금리 수준을 판단할 수 있는 다양한 금융기관과 금융상품을 포괄하는 대표성 있는 금리지표

③ 생활물가지수

- 생활물가지수는 전체 소비자물가대상 상품과 서비스를 포괄하는 월별 데이터

④ 소비자심리지수

- 소비자들의 경제상황에 대한 심리를 종합적으로 나타냄

⑤ 네이버 데이터랩

- 네이버 검색 조회량에 대한 일별 데이터
- '핀다', '대출', '금리' 단어 조회 수를 추가해 조회 수가 증가할수록 관심도가 많아져 대출 신청을 많이 할 수 있다는 아이디어에서 기인
- 각 데이터의 나이대, 성별, 날짜 기준으로 추가

핀다 대표 고객층 특징(변수별 최빈값) : user_spec

① birth_year - 1980년대생

② gender - 남자

③ employment_type - 정규직

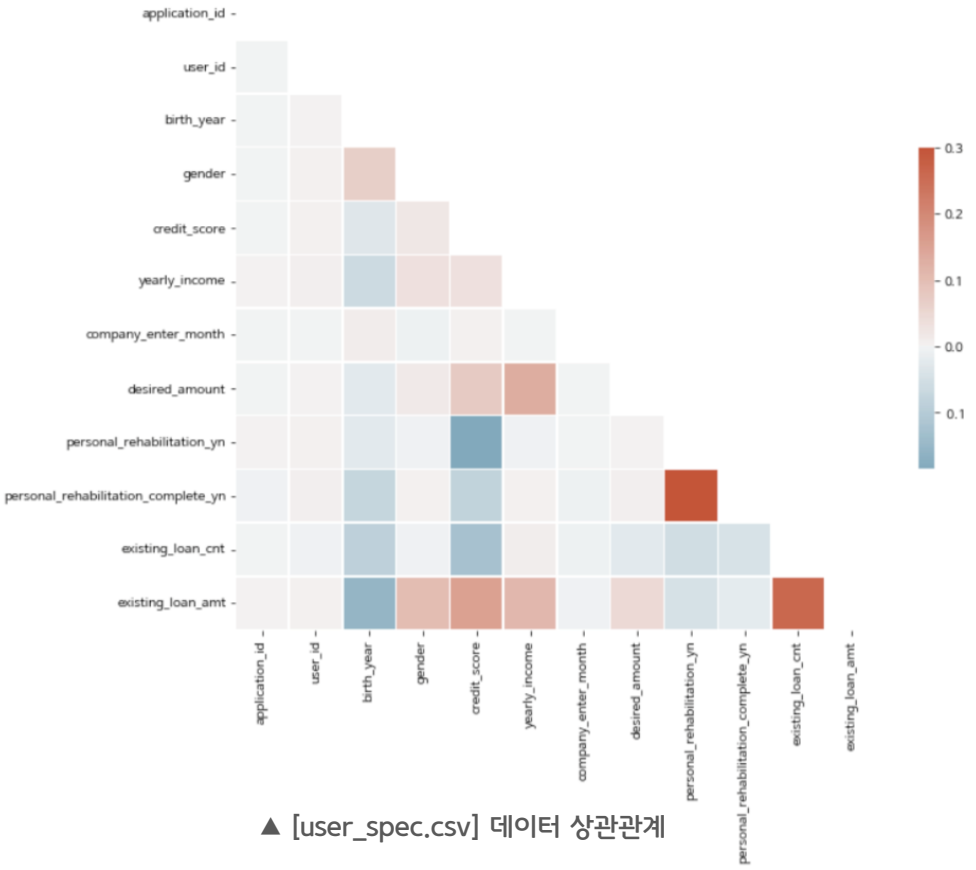
④ income_type - 4대보험이 있는 직장 가입자

⑤ houseown_type - 주거소유형태 전월세

⑥ purpose - 대출 목적은 생활비

⑦ existing_loan_cnt - 기대출수 10~20회 사이

⑧ personal_rehabilitation_yn - 개인회생자가 아닌 사람



▲ [user_spec.csv] 데이터 상관관계

상관관계

- 대출을 한 적이 있는 사람의 신용등급이 낮음
- 나이가 많을수록 기대출금액도 많음
- 연소득이 높을수록 대출 희망금액 높음

대출 상품 결과 특징 : loan_result

① loanapply_insert_time

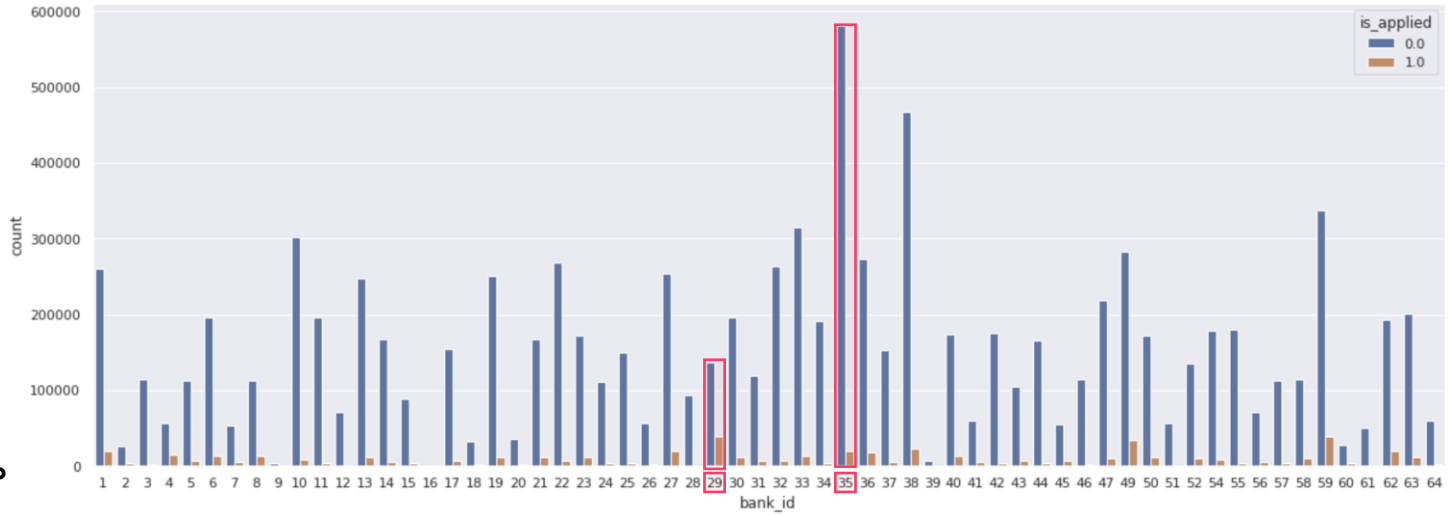
- 오전(특히 10시)에 주로 신청

② bank_id

- 35번 은행에서 가장 많은 신청서가 발급되었지만,
실제 대출은 29번 은행에서 가장 많이 이루어짐

③ is_applied

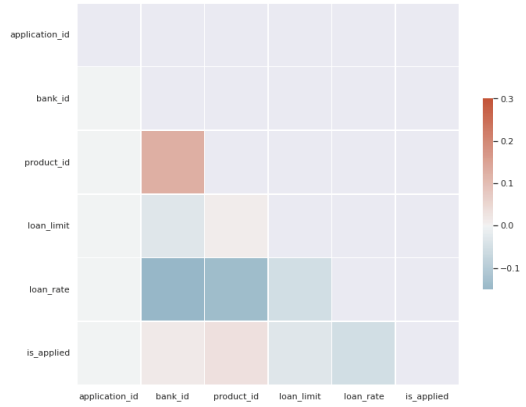
- 발급된 신청서 중, 실제로 대출이 이루어진 수는 약 5%



▲ bank_id 별 is_applied 개수 확인

상관관계

- 승인금리가 낮을수록 대출이 이루어짐
- 승인한도가 낮을수록 승인 금리가 낮음



▲ [loan_result.csv] 데이터 상관관계

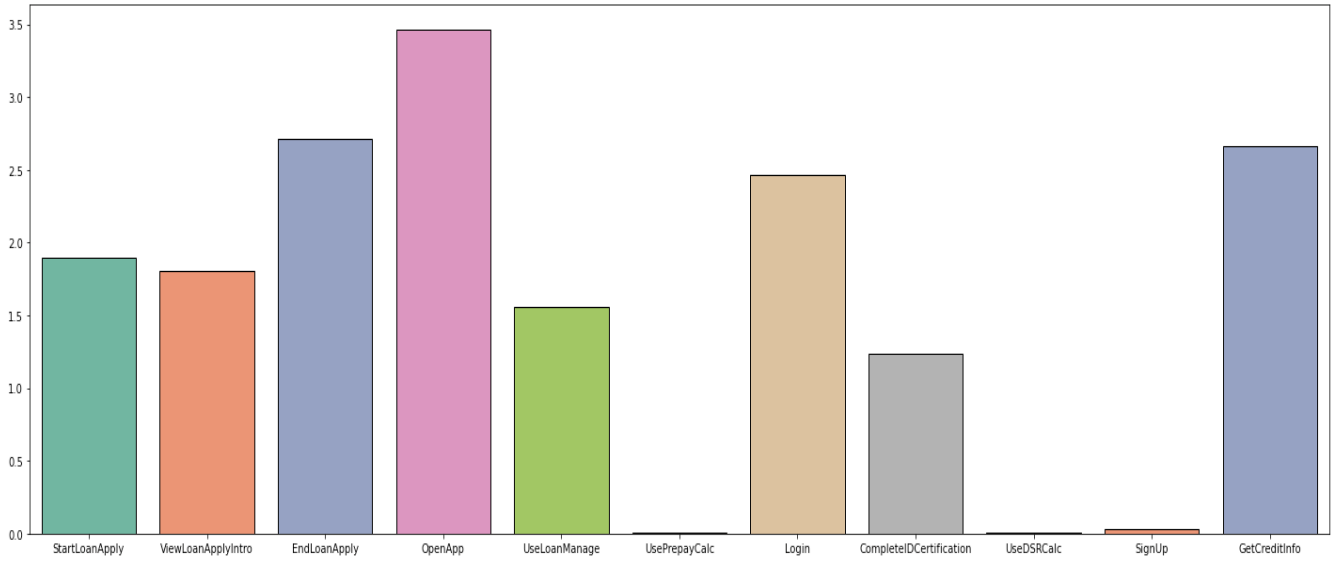
앱 로그 정보 특징 : log_data

① event - 앱 사용중 한도조회 결과를 조회하거나 KCB 신용정보를 조회한 경우가 가장 많고, 계산기를 사용한 경우가 가장 적음

② timestamp - 오전(특히 11시)에 접속자 수가 가장 많음

③ mp_os - 접속한 OS는 Android가 가장 많음

④ date_cd - 매월 16일에 가장 많은 사용자가 이용, 31일에 가장 적은 사용자가 이용



▲ event 열 데이터 분포도

III

예측 문제

01. 데이터 정제

① 각 테이블 값들의 형태 통일

company_enter_month	purpose
202109.0	생활비
20070201.0	Living
20151101.0	Living

↓ '년+월+일' → '년+월' 로 통일
영어 or 한글 → 한글로 통일

company_enter_month	purpose
202109.0	생활비
200702.0	생활비
201511.0	생활비

② 이상치 파악

- 만 19세 이하 데이터 삭제
- IQR 기준으로 이상치 및 결측치 대체

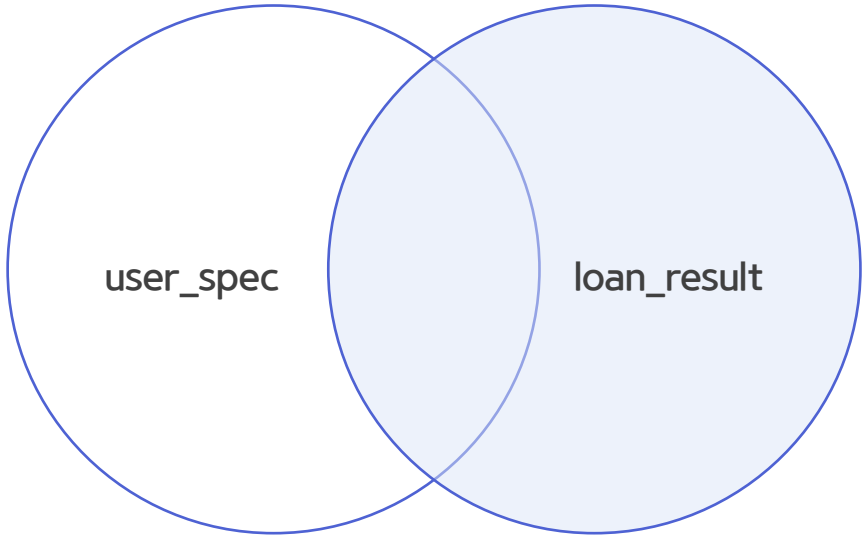
③ 결측치 대체

- 다중대체법 사용
- 동일 column의 값으로 대체
- 카테고리별 최빈값으로 대체

02. 데이터 통합

[user_spec] + [loan_result] 테이블 [loan_result] 방향으로 left join

+ 외부데이터 [콜금리, 기준금리, 생활물가지수, 소비자심리지수, 네이버 데이터랩]



03. 파생변수 생성

① loanapply_insert_time(한도조회 일시) → loanapply_month, loanapply_day, loanapply_time, loanapply_week

- loanapply_insert_time 을 월별, 일별, 시간대별, 요일별로 나누어 만든 파생변수

loanapply_insert_time
2022-06-21 8:31

→

loanapply_month	loanapply_day	loanapply_time	loanapply_week
06	21	8:31	Tuesday

② insert_time(생성일시), company_enter_month(입사연월) → time_at_work (근로 기간)

- 대출 신청을 위한 한도조회 당시의 insert_time 연도에서 company_enter_month 연도를 빼서 만든 근로 기간(단위 : 년) 파생변수

insert_time	company_enter_month
2022-03-06 1:00:29	201901

→

time_at_work
3

03. 파생변수 생성

③ birth_year(유저 생년월일) → 나이대

- '20대, 30대, 40대, 50대, 60대, 60대 이상'으로 10년 단위로 나이대를 구분해놓은 파생변수

birth_year
1971

→

나이대
50

④ personal_rehabilitation_yn(개인회생자 여부), personal_rehabilitation_complete_yn(개인회생자 납입 완료 여부) → rehabilitation_yn

- 두 변수의 (0, 0), (1, 0), (0, 1), (1, 1) 4가지 조합 중 (0, 1)을 이상치로 판단하여 제거 후 나머지 3가지 조합을 0, 1, 2로 범주화하는 파생변수

personal_rehabilitation_yn	personal_rehabilitation_complete_yn
0	0
1	0
1	1

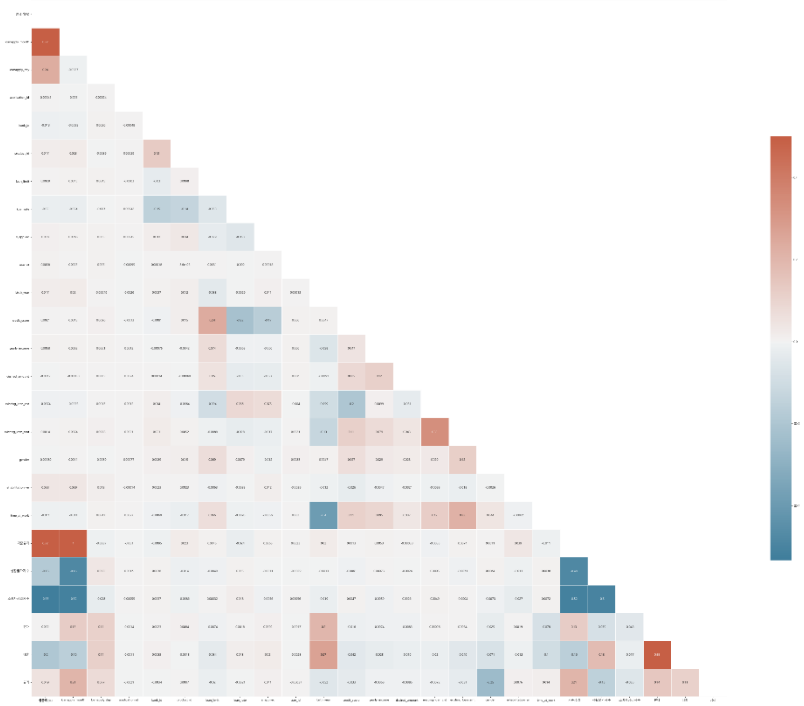
→

rehabilitation_yn
0
1
2

04. 데이터 축소

① 상관계수 분석(heatmap)

- 상관계수가 0.7 이상일 경우 다중공선성의 우려가 있다고 판단
- 두 변수의 조합 중 is_applied 와의 상관관계가 더 작은 변수를 삭제
- '기준금리', 'loanapply_month' 삭제



② VIF(분산팽창요인) 활용

- 10 이상일 경우 다중공선성의 우려가 있다고 판단
- 수치형 자료들의 경우 그보다 작은 값을 가짐

features	VIF Factor
콜금리(%)	3.05057
loanapply_day	1.29765
application_id	1.00009
bank_id	1.04682
product_id	1.03422
loan_limit	1.08890
loan_rate	1.11654
user_id	1.00043
birth_year	1.53558
credit_score	1.31057
yearly_income	1.03392
desired_amount	1.03957
existing_loan_cnt	1.22556
existing_loan_amt	1.34394
gender	1.14946
rehabilitation_yn	1.00356
time_at_work	1.36430
생활물가지수	2.71554
소비자심리지수	3.56461
핀다	1.98141
대출	3.39181
금리	1.49575

04. 데이터 축소

③ 추가적 변수 제거 작업

- ‘rehabilitation_yn’ : 개인회생자 여부를 이용해 만든 파생변수로, credit_score 계산에 이미 포함되어있다 판단하고 제거
- ‘나이대’ : 연속형 변수를 범주형 변수로 변환해 만든 파생변수로, 성능 향상에 도움이 되지 않아 birth_year을 다시 사용
- ‘loan_apply_time : 시계열 데이터를 split해서 만든 파생변수로, 시계열적 특징이 나타나지 않아 제거

01. 모델링 목표

성능 평가

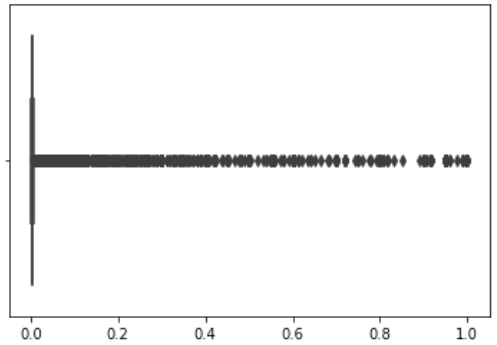
- 교차검증을 통해 안정적으로 성능을 평가
- 모델 성능을 위해 다양한 외부 변수 추가
- 단순히 수치의 높고 낮음이 절대적인 성능 평가의 기준이 되지 않도록 함
- 여러 분류 모델의 비교와 튜닝을 통해 최종 모델의 F1-score 획득

해석력

- 모든 변수를 사용하여 성능을 향상시키기 보다는 상관관계 및 통계적 검증절차 분석을 통해 변수를 선별하여 가치있는 데이터를 사용하고자 함

변수의 다양성 고려

- 데이터의 양이 많아 표기 형태가 다른 데이터나 이상치에 유의해야 함



02. 모델 소개

① Logistic Regression

- 가볍고 빠르지만 이진 분류 예측 성능도 뛰어남
- 모든 독립변수의 증감에 대해 대출 신청여부가 1이 되는 확률로 해석이 가능
- 모델결과를 이해하기 편리

② LightGBM

- 이전에 만들어진 트리들로부터 정보를 사용하여 만드는 부스팅 기법
- 메모리 사용량과 학습에 걸리는 시간이 다른 부스팅 기법에 비해 상대적으로 적어 본 데이터에 적합
- 많은 데이터 세트에 적용하기 때문에 과적합 가능성 적음
- 최대 손실 값을 가지는 리프 노드를 분할하면서 비대칭적인 규칙 트리를 지속적으로 생성 → 예측 오류 손실 최소화
- 순차적으로 학습을 진행하면서 예측이 틀린 데이터에 대해 가중치 부여 → 이전에 예측이 틀린 데이터에 대해 보다 높은 정확도로 예측할 수 있음
- 하이퍼파라미터가 너무 많고 튜닝을 위한 시간이 많이 소모

02. 모델 소개

③ Random Forest

- 양이 방대하여 알고리즘 중 비교적 빠른 수행 속도를 가지고 있어 데이터에 알맞은 모델
- 결정트리의 쉽고 직관적인 장점을 그대로 사용하되, 각 분할에서 특정 개수의 설명변수들의 새로운 표본이 선택
- 트리들의 상관성을 제거하는 간단한 방법 → 배깅된 트리들보다 더 나은 성능 제공
- 분할을 수행할 때마다 설명변수들의 일부분만 고려하여 상관성을 제거 가능
- 결과 트리들의 평균은 변동성이 적어지고 더 안정적으로 예측(= 분산을 줄일 수 있음)
- 지니 지수의 평균 감소를 사용하여 변수 중요도를 확인할 수 있고 시각화를 통해 변수들을 확인할 수 있어 편리
- 하이퍼파라미터가 너무 많고 튜닝을 위한 시간이 많이 소모

03. 최종 모델 소개

앙상블(Ensemble)

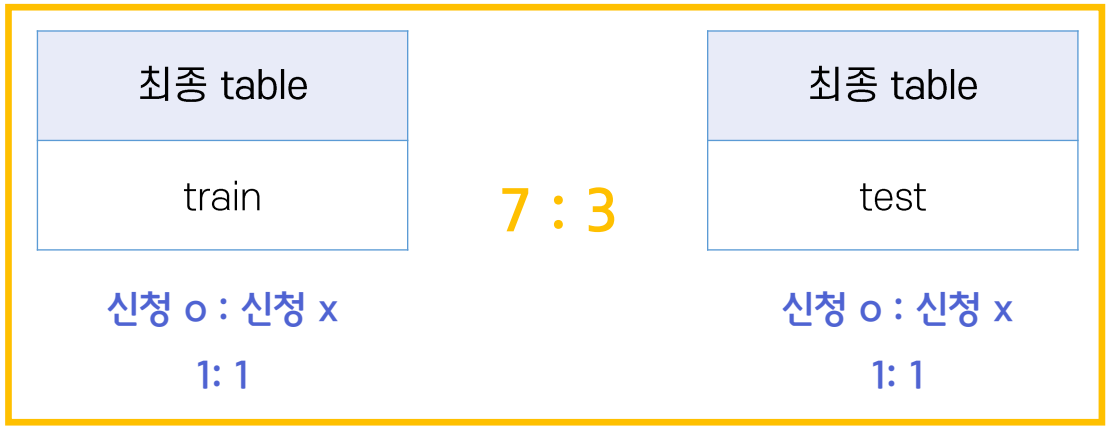
- 다양한 분류기의 예측 결과를 결합함으로써 단일 분류기보다 신뢰성이 높은 예측값을 얻을 수 있음
- 그 중 서로 다른 알고리즘을 가진 분류기를 결합하는 'Hard Voting'을 사용해 최종 예측 결과를 선정
- Y값이 불균형한 데이터로 양성(1)이 상대적으로 적어 음성(0)으로 예측할 횟수가 늘어남
- 세 개의 알고리즘에 분류 결정 임곗값을 변화시켜 분류 결과를 조정하고 'Hard Voting'을 통해 최종 예측 결과를 선정

앱 사용성 데이터를 이용해 대출 신청 여부를 분류하여 고객에게 알맞은 대출 상품을 추천하여,
고객의 니즈를 충족시키는 것으로 같은 f1 score 여도 재현율이 높은 결과값이 중요하다고 판단

대출 신청을 하지 않으려는 고객에게 대출 서비스를 제공하는 것은 괜찮지만, 대출을 신청하려는 고객에게 서비스를 제공해주지 않는다면
고객뿐만 아니라 핀다에서도 엄청난 손해를 보기 때문
이런 측면에서 임계치 조정 이전 재현율은 평균적으로 0.8 정도 되어 가치가 있는 분류 결과라고 생각한다.

04. 모델 학습 및 평가

① 3,4,5월 데이터를 is_applied = 1인 데이터와 0인 데이터의 비율이 1:1이 되도록 7:3(train:test)으로 나눔



데이터의 신청 o : 신청 x 비율을 1 : 1로 맞춤

- is_applied = 1인 데이터 약 38만개
- is_applied = 0인 데이터 중 약 38만개 비복원 추출

→ 약 76만개의 데이터가 5개 생성

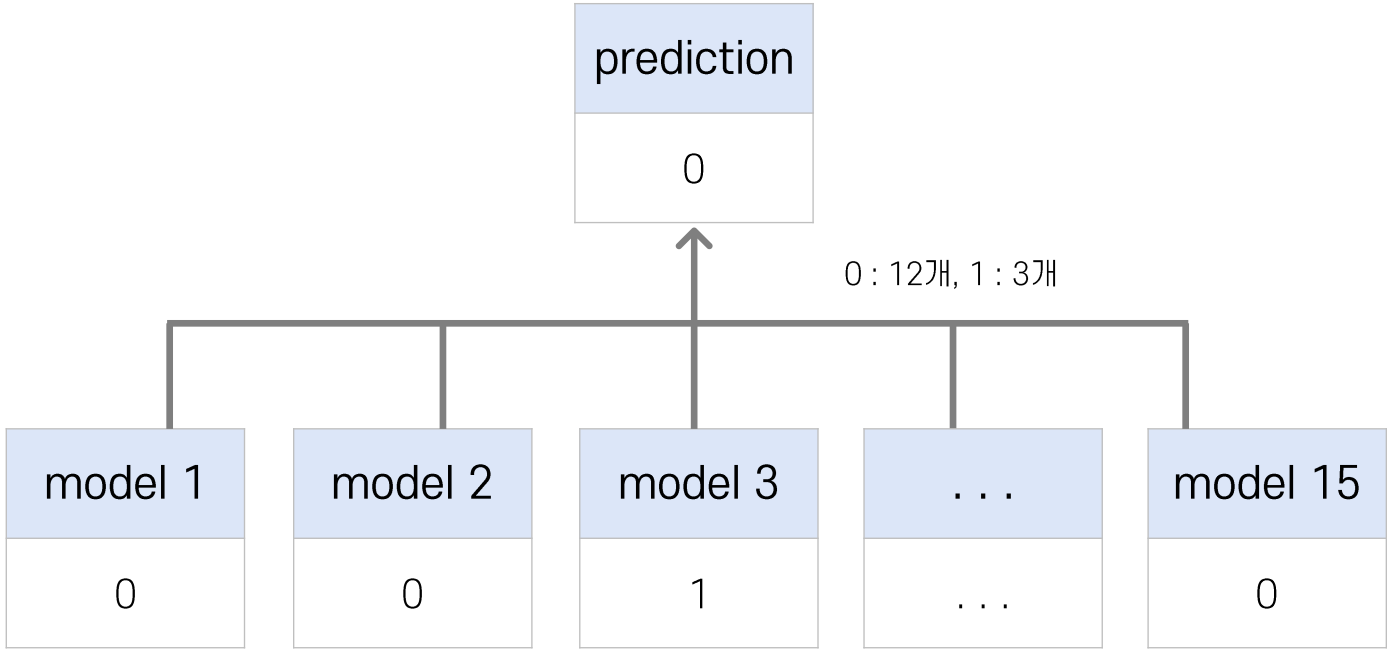
04. 모델 학습 및 평가

② StandardScaler 를 통해 정규화 진행

③ 각 5개의 데이터에 Logistic Regression, LightGBM, Random forest 모델 학습 후 하이퍼파라미터 튜닝

④ 5개의 데이터, 3개의 모델(총 15개)에 분류 결정 임계값을 변화시켜 분류 결과를 조정

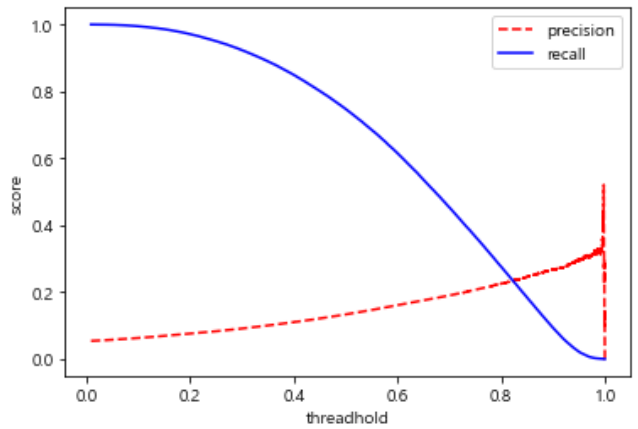
⑤ Hard voting을 통해 최종 예측 결과를 선정



01. 모델 결과

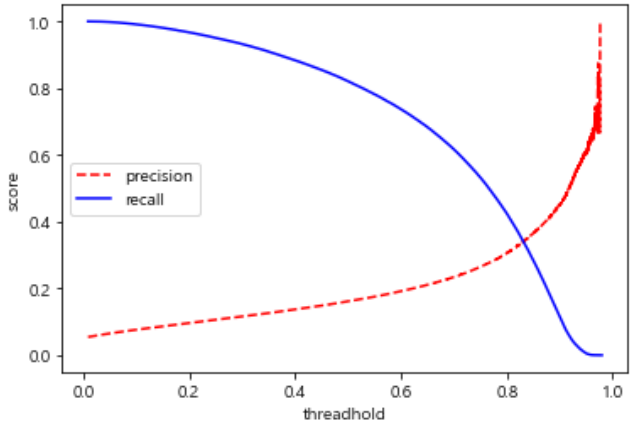
① Logistic Regression

	임계치 조정 이전	임계치 조정 이후
Accuracy	0.72	0.87
Precision	0.13	0.19
Recall	0.75	0.45
F1 Score	0.23	0.27
AUC	0.73	0.67



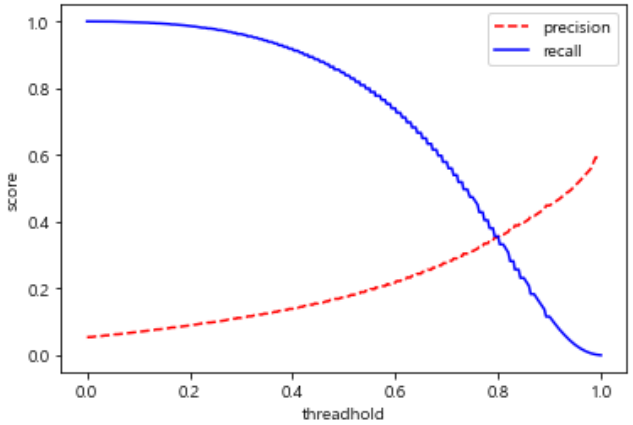
② LightGBM

	임계치 조정 이전	임계치 조정 이후
Accuracy	0.78	0.90
Precision	0.18	0.28
Recall	0.84	0.56
F1 Score	0.29	0.38
AUC	0.81	0.74



③ Random Forest

	임계치 조정 이전	임계치 조정 이후
Accuracy	0.76	0.92
Precision	0.16	0.31
Recall	0.82	0.43
F1 Score	0.27	0.37
AUC	0.79	0.6



01. 모델 결과

② LightGBM

① Logistic Regression

③ Random Forest

Hard-Voting을 통해 생성된 최종 모델 성능

Accuracy : 0.91, Precision: 0.29,

Recall : 0.49, F1 : 0.37, AUC : 0.71

```
: final['result'].value_counts()

: 0    3023941
  1     231253
  Name: result, dtype: int64
```

02. 모델 기반 패턴 분석

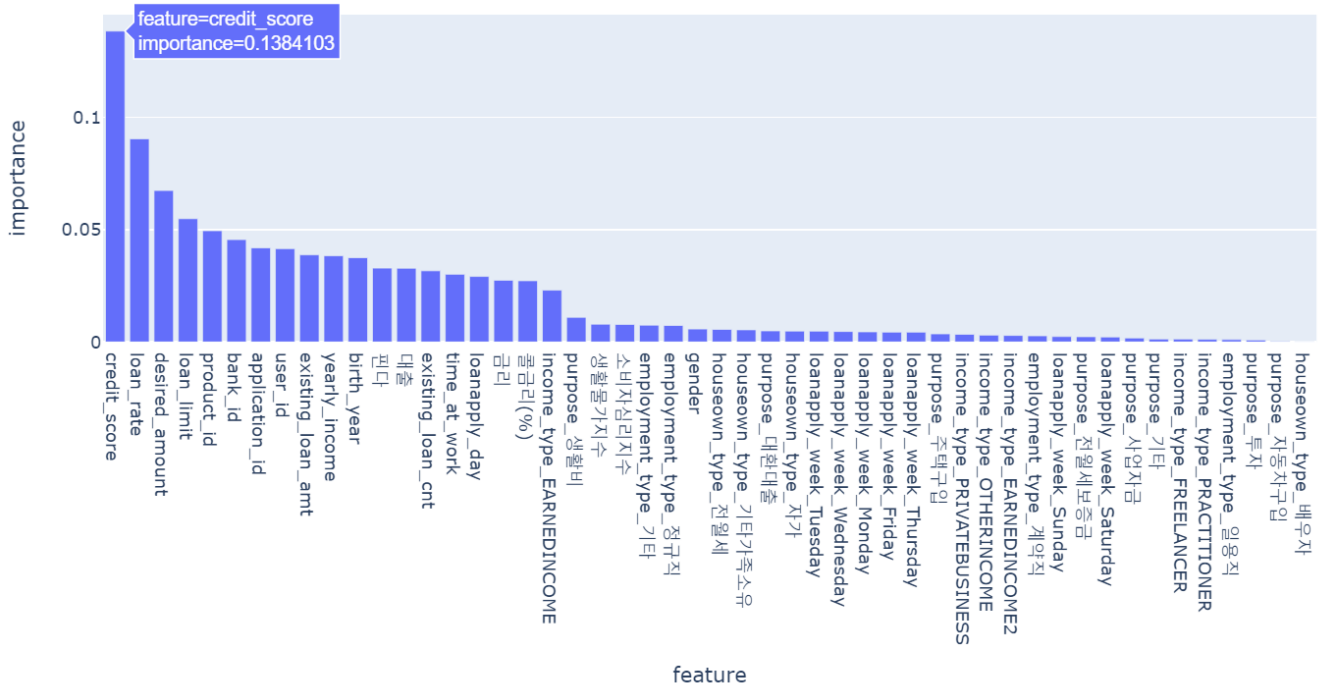
① Logistic Regression 기반

- 로짓 변환 후 summary 메서드를 통해 분석
- credit_score, loan_rate, desired_amount가 is_applied에 유의미한 영향을 끼치는 걸 확인

변수	coef	std err	z	P >|z|	[0.025	0.975]
credit_score	-0.9795	0.004	-272.326	0.000	-0.987	-0.972
loan_rate	-0.5954	0.003	-196.764	0.000	-0.601	-0.589
desired_amount	-0.4112	0.004	-114.756	0.000	-0.418	-0.404

② Random Forest 기반

- feature_importance 를 통해 분석
- credit_score, loan_rate, desired_amount



Ⅳ

군집 문제

01. 데이터 정제

- [mp_os], [mp_app_version] 열 제거
- 실제로 Android, IOS 등 환경에서 사용해 본 결과, OS 버전 등의 기기 정보는 어플리케이션을 사용하는 데 큰 영향을 미치지 않는다고 판단

user_id	date_cd	event	timestamp	mp_os	my_app_version
1	2022-05-03	GetCreditInfo	2022-05-03 14:52:28	Android	3.8.2
1	2022-05-03	GetCreditInfo	2022-05-03 14:52:35	Android	3.8.2
7	2022-06-16	UseLoanManage	2022-06-16 23:58:41	Android	3.8.4
7	2022-06-16	Login	2022-06-16 23:58:41	iOS	3.6.1

- 각 event 간격이 15분 이내인 경우 같은 접속이라고 판단
- 같은 접속의 event, timestamp 값을 한 행으로 모음

user_id	date_cd	events	timestamp
1	2022-05-03	GetCreditInfo,GetCreditInfo	2022-05-03 14:52:28,2022-05-03 14:52:35
7	2022-06-16	UseLoanManage,Login	2022-06-16 23:58:41,2022-06-16 23:58:41

01. 데이터 정제

- birth_year : 연속형 변수 → 범주형 변수(나이대)
- application_id 등 7개의 변수 제거

application_id	loanapply_insert_time	…	birth_year	personal_rehabilitation_complete_yn
1748340	2022-06-07 13:05:41	…	27	0.00000
1748340	2022-06-07 13:05:41	…	27	0.00000
1748340	2022-06-07 13:05:41	…	27	0.00000
1748340	2022-06-07 13:05:41	…	27	0.00000

application_id, insert_time, is_applied, company_enter_month, rehabilitation_yn,
personal_rehabilitation_yn, personal_rehabilitation_complete_yn

birth_year
20
20
20
20

02. 파생변수 생성

동일한 user_id는 한 데이터프레임에 여러 개 존재하지 않도록 한다

- 동일한 user_id 데이터가 여러 개 존재하는 기존의 [log_data] 데이터를 그대로 사용하면 병합 및 user_id의 대표성에서 문제가 생길 것이라고 판단
- 따라서 추가적인 작업과 user_id를 통한 파생변수 생성

02. 파생변수 생성

각 event 간격이 15분(900초) 이내인 경우 같은 접속이라고 가정

같은 접속의 event, timestamp 값들을 한 행으로 모음

user_id	date_cd	event	timestamp
1	2022-05-03	GetCreditInfo	2022-05-03 14:52:28
1	2022-05-03	GetCreditInfo	2022-05-03 14:52:35
7	2022-06-16	UseLoanManage	2022-06-16 23:58:41
7	2022-06-16	Login	2022-06-16 23:58:41

user_id	date_cd	events	time
1	2022-05-03	GetCreditInfo,GetCreditInfo	2022-05-03 14:52:28,2022-05-03 14:52:35
7	2022-06-16	UseLoanManage,Login	2022-06-16 23:58:41,2022-06-16 23:58:41

02. 파생변수 생성

events 변수 중에서 'OpenApp' 문자열 이벤트를 제거 (어플을 켜는 것 자체가 OpenApp을 수반하기 때문)

일종의 불용어 처리

user_id	date_cd	events	time
201	2022-05-16	OpenApp, GetCreditInfo,GetCreditInfo	2022-05-16 14:52:28,2022-05-16 14:52:35
202	2022-06-07	OpenApp,OpenApp	2022-06-07 14:52:28,2022-06-07 14:52:35
205	2022-04-23	UseLoanManage	2022-04-23 14:52:28,2022-04-23 14:52:35
207	2022-06-14	Login	2022-06-14 14:52:28,2022-06-14 14:52:35

user_id	date_cd	events	time
201	2022-05-16	GetCreditInfo,GetCreditInfo	2022-05-16 14:52:28,2022-05-16 14:52:35
202	2022-06-07		2022-06-07 14:52:28,2022-06-07 14:52:35
205	2022-04-23	UseLoanManage	2022-04-23 14:52:28,2022-04-23 14:52:35
207	2022-06-14	Login	2022-06-14 14:52:28,2022-06-14 14:52:35

02. 파생변수 생성

timestamp → time

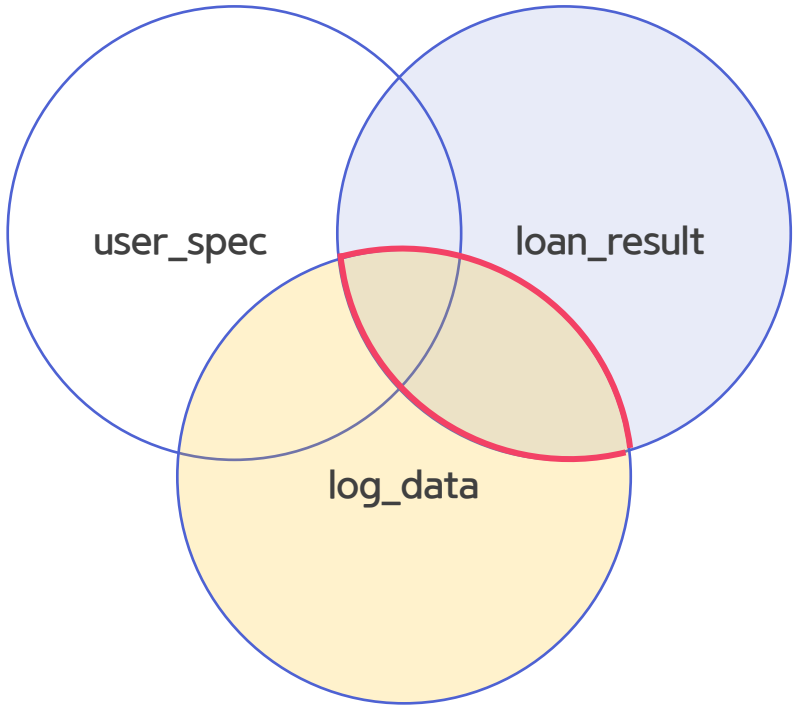
- 전처리된 timestamp 에 여러 값들이 존재할 경우, 최종 접속 시점에서 최초 접속 시간을 빼서 만든 시간(단위 : 초) 파생변수
- 전처리된 timestamp 에 값이 한 개만 존재할 경우, time 은 0으로 처리

timestamp
2022-05-03 14:52:28,2022-05-03 14:52:35
2022-05-22 16:39:49

time
7
0

03. 데이터 통합

[user_spec] + [loan_result] 테이블 [loan_result] 방향으로 left join 한 데이터에 [log_data] 테이블 inner join



01. 모델링 목표

성능 평가

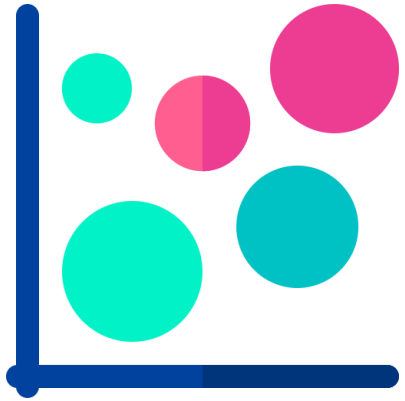
- 집단 내 응집도와 분리도를 최대화하는 알고리즘을 선택해 그룹의 대표성 도출

해석력

- 데이터의 특징이나 형태에 따라 적절한 군집 알고리즘을 사용

변수의 다양성 고려

- 집단의 특징 파악을 위해 변수 선택과 군집 개수에 유의함



02. 모델 소개

수퍼루키님

저녁 11시 12분

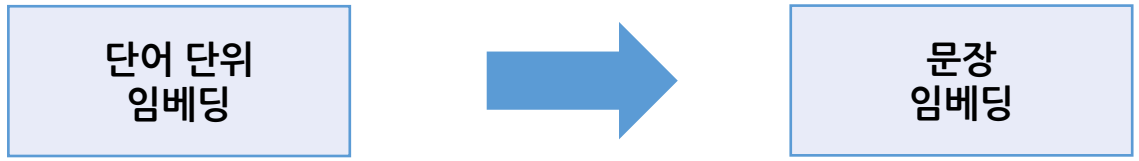
events 변수를 군집화 할 때 어떻게 최적화할까??

나

events의 문자열은 10여개의 단어(SignUp, StartLoanApply 등)로만 구성된 문자열이고,
일반적인 자연어와는 다르게 적은 양의 단어가 반복해서 나온다.
events를 일일이 전처리하기 보다는, embedding 벡터를 활용하면 간단하면서 효율적으로 할 수 있지 않을까?

저녁 11시 12분

GloVe : 카운트 기반과 예측 기반을 모두 사용하는 방법론, Word2Vec의 예측 기반, LSA기반의 카운트 기반을 둘 다 사용



02. 모델 소개

① 단어 단위 embedding

- ‘Corpus’ 라이브러리를 활용 : events 열의 문자열들을 window size = 5로 두고 corpus화
- 각 단어들을 32차원으로 벡터화

ViewLoanApplyIntro → *([0.52122394, -0.53870755, -0.50408432, 0.51495922, -0.53594435, ... ,-0.531547 , 0.50260563, -0.53763296, 0.52180368, -0.53376842]),*

② 문장 embedding

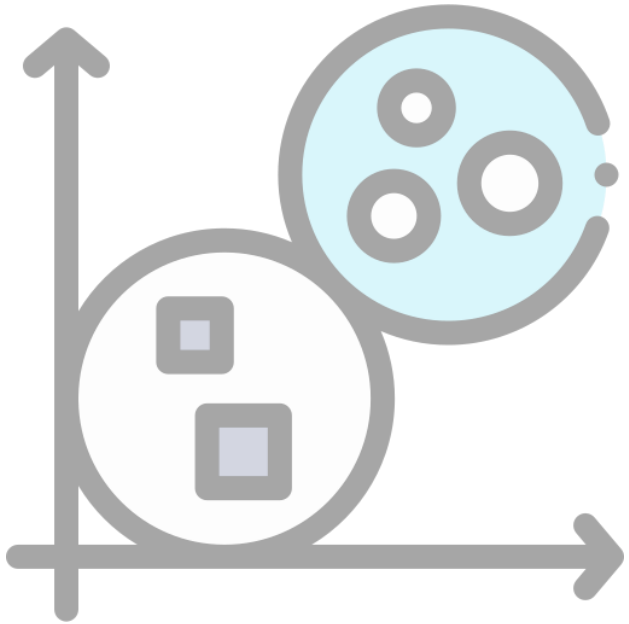
- Glove 모델을 활용해 학습한 단어 embedding을 가지고, 문장을 구성하는 embedding 벡터들의 열(column) 평균을 활용해서 문장 embedding을 진행
- 문장 또한 단어 embedding vector과 동일한 32차원으로 embedding을 진행

OpenApp,Login,StartLoanApply → *([-0.51256538, -0.51179434, -0.49176959, 0.51354897, -0.51007658, ... ,-0.50146263, 0.48732218, -0.51015109, 0.52351302, -0.5303776]),*

02. 모델 소개

③ 군집화 알고리즘 : K-means++

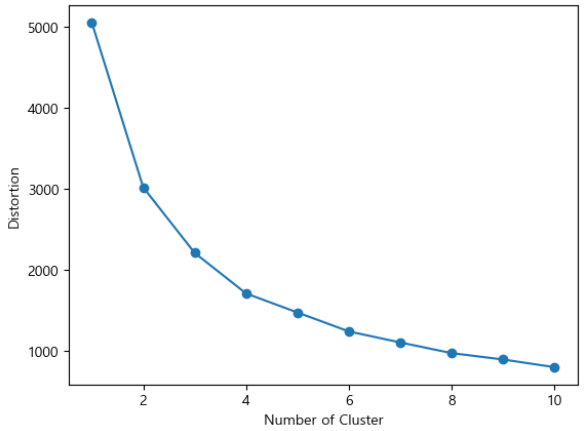
- 비지도학습 모델 중 하나로, K개의 Centroid를 기반으로 K개의 클러스터를 만들어 주는 것을 목표로 함
- 각 클러스터와의 거리 차이의 분산을 최소화하는 방식으로 작동
- Centroid를 선택하는 방법은 주로 랜덤하게 할당
- Centroid를 배치하는 방법을 거리비례 확률에 따라 선택하는 K-means++ 선택



02. 모델 소개

④ K값 설정

- K값을 1~10까지 설정한 후, Elbow Method 확인을 통해 k값을 2~5 사이로 결정
- k가 2~5일 경우를 모두 군집화 한 후 실제로 events들이 어떻게 군집 되었는지 확인한 결과, k=5일 때 상대적으로 잘 군집화 되었다는 것을 확인



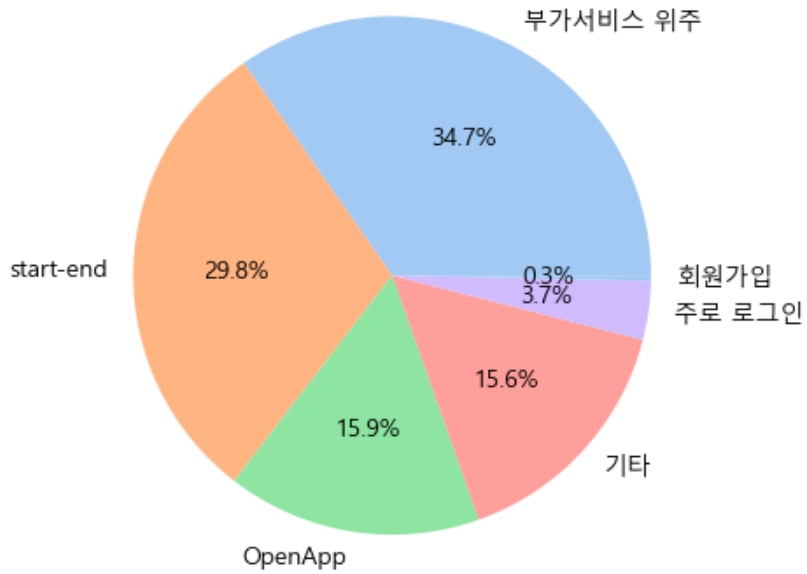
events	timestamp	time	cluster
GetCreditInfo,GetCreditInfo	2022-05-03 14:52:28,2022-05-03 14:52:35	7	1
UseLoanManage,Login,GetCreditInfo	2022-06-16 23:58:41,2022-06-16 23:58:41,2022-0...	1	1
GetCreditInfo	2022-05-22 16:39:49	0	1
GetCreditInfo,GetCreditInfo,GetCreditInfo	2022-05-21 23:37:58,2022-05-21 23:43:33,2022-0...	354	1
GetCreditInfo,UseLoanManage,GetCreditInfo,UseP...	2022-03-24 10:53:59,2022-03-24 10:54:07,2022-0...	692	1

events 데이터를 활용한 log데이터 클러스터링 결과가 0~6 으로 표현

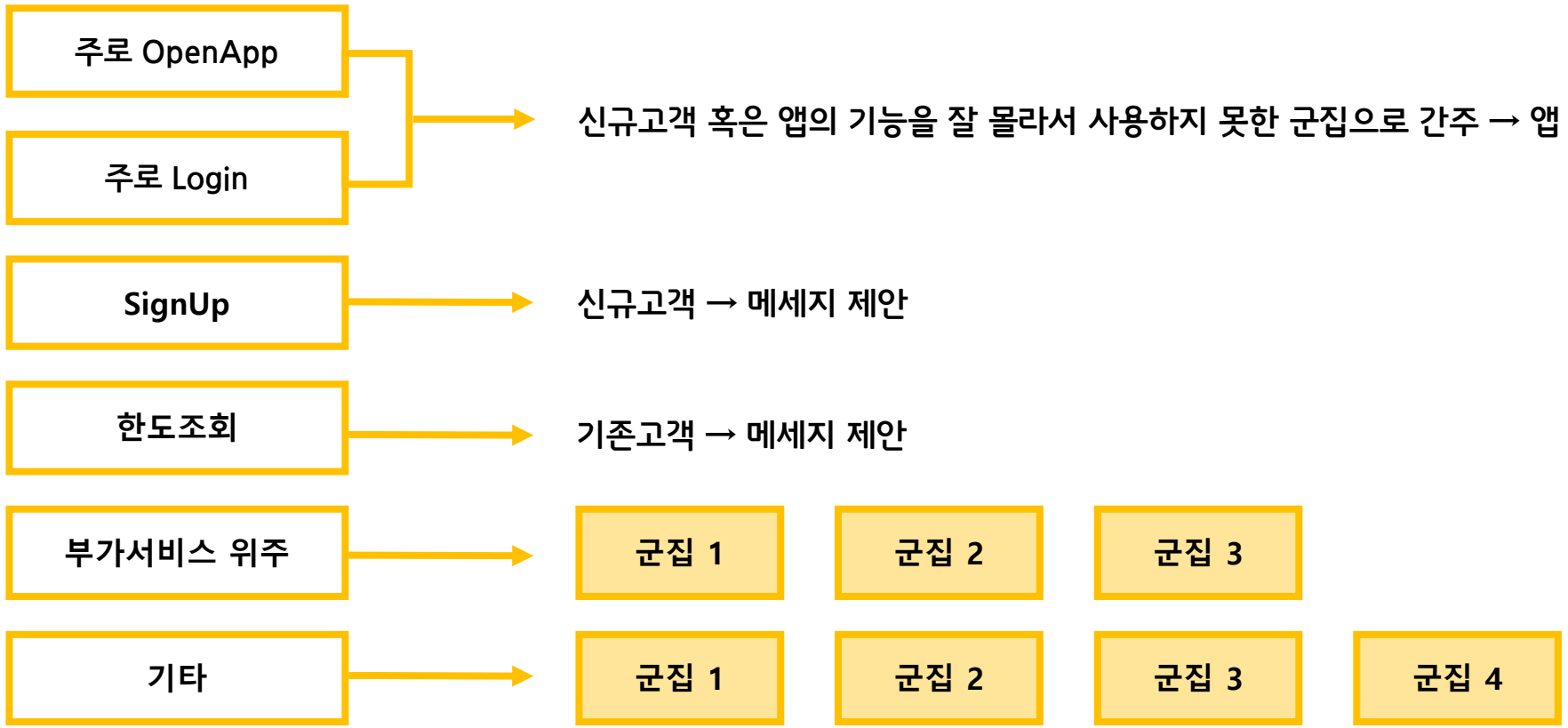
03. 모델 구현

events 데이터를 활용한 log데이터 클러스터링 결과

- 0번 : 기타
- 1번 : 부가서비스 위주
- 2번 : start - end loan apply
- 3번 : 회원가입
- 4번 : 주로 로그인 event만
- 5번 : OpenApp event만



03. 모델 구현


주로 OpenApp
주로 Login
신규고객 혹은 앱의 기능을 잘 몰라서 사용하지 못한 군집으로 간주 → 앱 기능 추천
SignUp
신규고객 → 메세지 제안
한도조회
기존고객 → 메세지 제안
부가서비스 위주
군집 1
군집 2
군집 3
기타
군집 1
군집 2
군집 3
군집 4

03. 모델 구현

user_loan 데이터와 결합 후 추가적인 군집 분석

- user_loan 데이터에 존재하지만 log data에 없는 데이터 → 로그 데이터 누락으로 이벤트로 1차 군집화 불가
- log data에 존재하지만 user_loan에 없는 데이터 → 유저 정보는 한도 조회시 생성되는데 date도 없기 때문에 log data와 join 불가능

‘ inner join 수행’

user_cluster_final.csv

user_id	date	time	cluster	count
1	2022-06-16	8	1	2
7	2022-05-22	0	1	1
9	2022-05-21	354	1	1
11	2022-06-13	1687	1	5

+

user_log_merge.csv

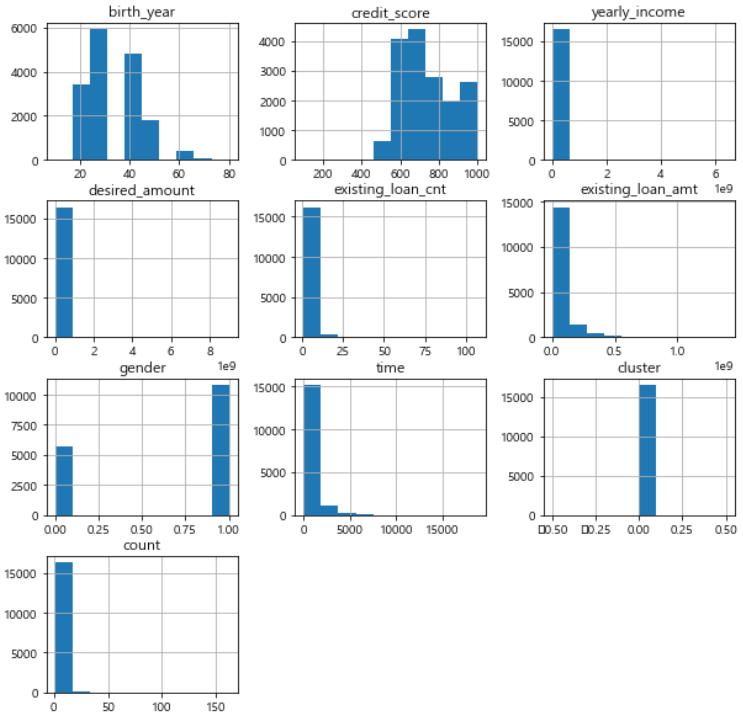
user_id	loanapply_insert_time	...	existing_loan_amt	gender
1	2022-05-16	...	15000000.00000	1.00000
7	2022-05-21	...	86000000.00000	1.00000
9	2022-06-03	...	23000000.00000	1.00000
11	2022-06-04	...	43000000.00000	1.00000

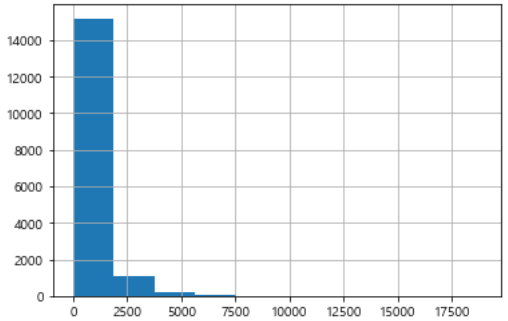
=

user_id	loanapply_insert_time	...	cluster	count
197454	2022-06-07	...	0	1
82141	2022-06-07	...	0	1
158340	2022-06-07	...	2	1

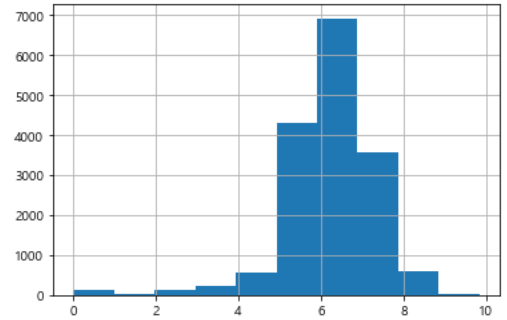
03. 모델 구현

군집분석을 위한 데이터 분포 확인 및 변환





ex) ‘time’ 로그변환 전



ex) ‘time’ 로그변환 후

히스토그램 및 첨도와 왜도를 보고 각각의 변수가 정규분포를 따르는지 확인
→ yearly_income, desired_amount, existing_loan_amt, existing_loan_cnt, time 변환 후 scaling 진행

03. 모델 구현

동일한 user_id 가 있으면 가장 마지막 user 의 정보를 선택 → user_id 를 하나로 통합

user_id	loanapply_insert_time	…	credit_score	personal_rehabilitation_complete_yn
101209	2022-03-16	…	650	0.00000
101209	2022-04-08	…	650	0.00000
101209	2022-04-08	…	650	0.00000
101209	2022-05-20	…	660	0.00000

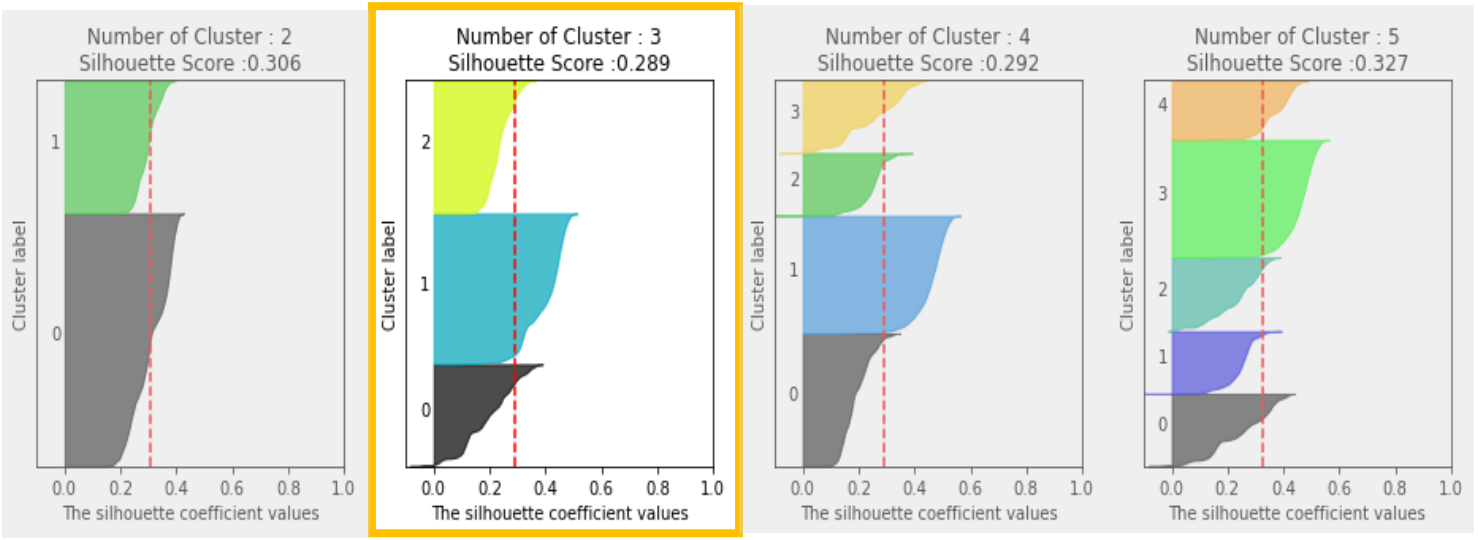
user_id	loanapply_insert_time	…	credit_score	personal_rehabilitation_complete_yn	count
101209	2022-03-16	…	650	0.00000	4

하나로 통합하기 전 user_id의 행 개수를 ‘count’ 열에 저장

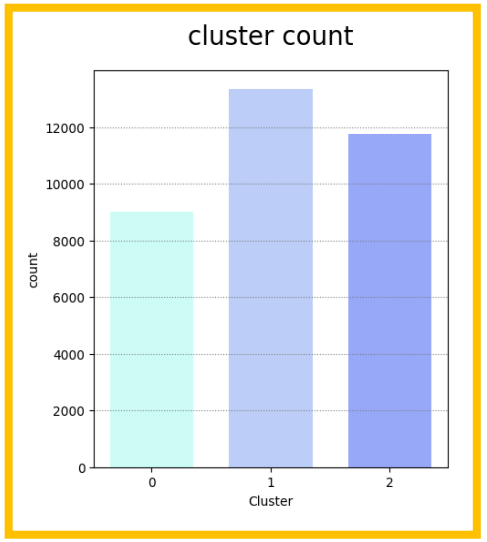
03. 모델 구현

부가서비스 위주 → 군집 1 / 군집 2 / 군집 3

주로 부가서비스를 이용하는 고객을 3개의 군집으로 세분화



실루엣 계수



군집별 개수

Elbow 숫자를 기준으로 군집의 개수를 지정하고, 실루엣 점수를 바탕으로 분할된 군집들의 특성을 직접 눈으로 확인

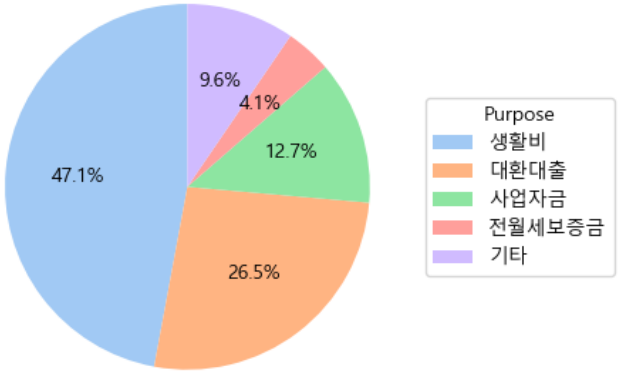
가장 특성이 두드러지게 나타나며, 많은 대상이 한쪽에 치우쳐져 있지 않은 군집을 형성한다.

03. 모델 구현

부가서비스 위주 → 군집 1 군집 2 군집 3

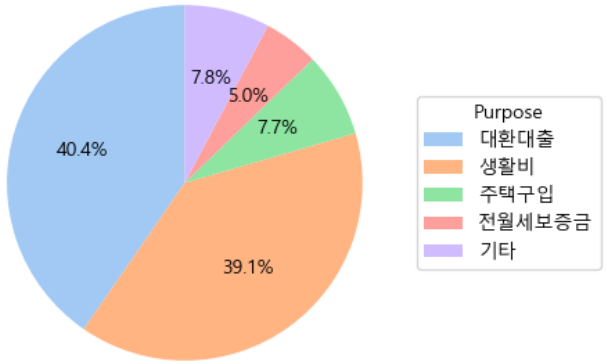
시각화를 통해 군집 간의 차이를 살펴보고 가설이 유의한지 검정을 통해 확인

부가 서비스 이용자는 대부분 3~40대로 이루어져있고 모든 군집에서 생활비, 대환대출이 목적인 사람이 70%이상



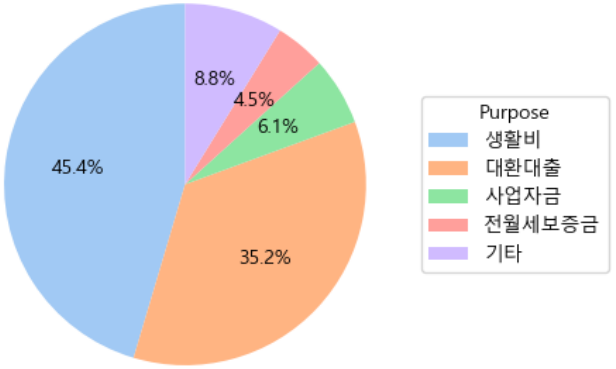
군집 1

주로 기타소득자로 이루어져 있고 생활비 대출이 목적이며 앱에 접속한 시간과 사용횟수가 가장 적음



군집 2

연소득, 신용점수, 대출 희망금액, 기대출수, 기대출금액이 가장 높고 앱 사용 시간과 방문 횟수가 가장 많음



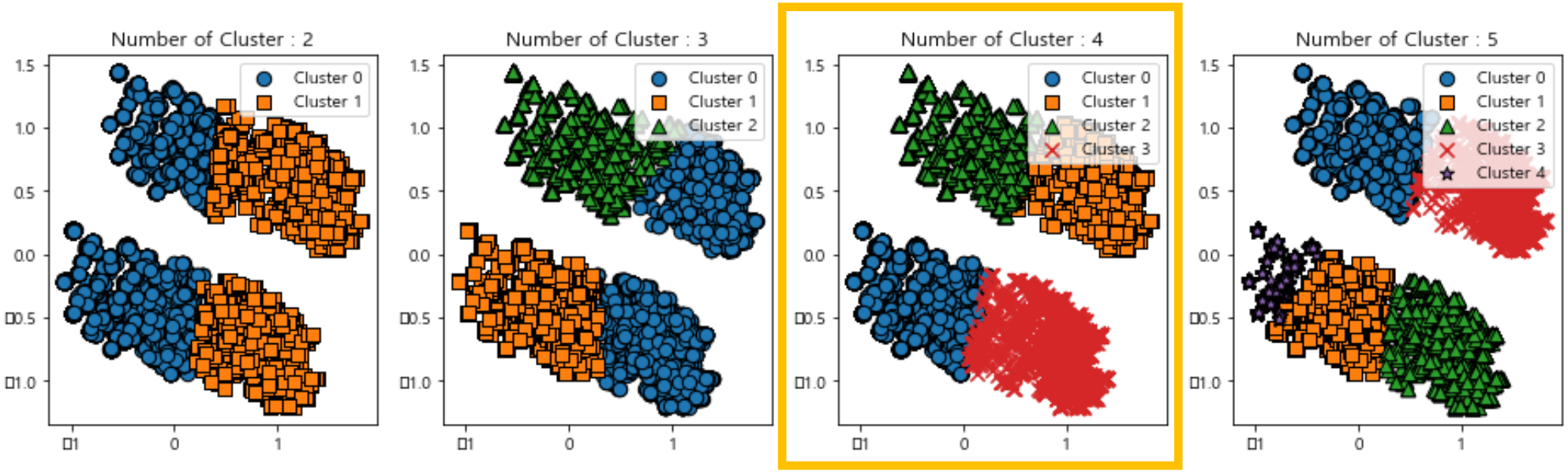
군집 3

대출 목적으로 생활비, 대환대출의 비율이 가장 높고, 여성 고객만 존재, 연소득, 신용점수, 대출 희망 금액이 가장 낮음

03. 모델 구현

기타 이용자 적절한 k 선택

군집 개수 정하기 위해 시각적으로 확인한 결과 군집 간의 차이가 가장 두드러지는 군집 개수 4개로 선택

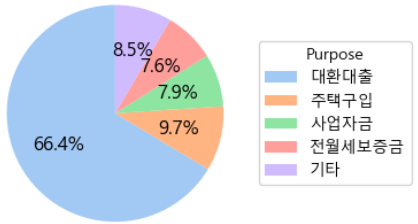


Ⅳ. 군집 문제

03. 모델 구현

기타 → 군집 1 | 군집 2 | 군집 3 | 군집 4

기타 이용자는 대부분 30~40대로 이루어져있고 모든 군집에서 대출 목적으로 생활비, 대환대출이 70%넘게 차지

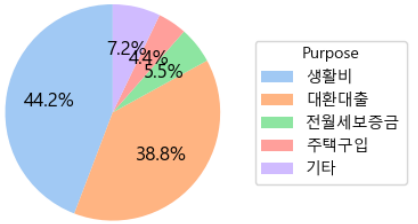


군집 1

직장가입자(4대보험) 80% 이상

정규직 90% 이상,

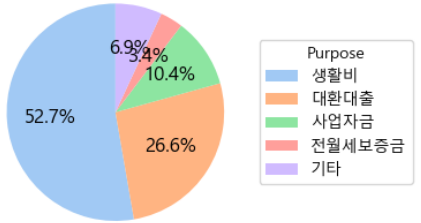
신용점수 평균, 연 소득 평균, 대출희망금액, 기대출수 등이 가장 높음



군집 2

'기타소득 - 프리랜서 - 개인사업자' 순서로 많은 비율 차지

'기타' 고용형태가 99%이상의 비율을 차지



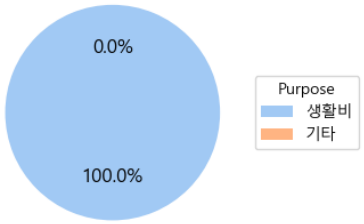
군집 3

직장가입자(4대보험) 80%

정규직 85~90%

연 소득 평균, 기대출수, 기대출금액 평균값 가장 적음

앱 접속시간 가장 짧음



군집 4

01. 모델 결과 분석

분산분석-군집간의 유의미한 차이가 있는가?

H_0 : 군집에 따라 연소득 평균의 차이가 없다.

H_1 : 군집에 따라 연소득 평균의 차이가 있다.

분산분석의 가정

① 독립성 : 독립변수의 그룹 군은 서로 독립적이어야 한다.

→ 일반적으로 독립성은 독립되어 있다하고 판단.

② 정규성: 각 군집간의 데이터는 정규분포를 따라야 한다.

ex) ShapiroResult(statistic=0.3892717957496643, pvalue=0.0)
ShapiroResult(statistic=0.0765770673751831, pvalue=0.0)

→ p=0.0 < 유의수준 = 0.05 이므로 각 군집은 정규분포를 따르지 않는다.

BUT, **중심극한정리로 n > 30이면 데이터는 정규분포를 따른다고 가정할 수 있다.**

01. 모델 결과 분석

③ 등분산성: 각 군집은 서로 같은 분산을 가져야 한다.

ex) LeveneResult(statistic=29.588474417880306, pvalue=4.531377588158256e-19)

→ levene 검증을 통한 등분산성 확인 → 모든 변수에서 등분산 가정이 성립되지 않음

등분산 가정이 성립되지 않는 경우, Welch's ANOVA 검증을 통해 모든 수치형 변수에서 통계적으로 유의미한 차이 발견

→ 유의확률 P=0.000 < 유의수준 = 0.05 이므로 각 군집은 유의한 차이가 있다.

Source	ddof1	ddof2	F	p-unc	np2
kmeans_minmax	3	8818.24017	74.81344	0.00000	0.01199

01. 모델 결과 분석

회원가입, OpenApp, Login

군집 1
(OpenApp, Login)

→ 계좌 잔액 조회와 원리금 납부 알림 서비스로 당장 신용점수 높여보세요. 번거롭게 은행까지 가지 말고 핀다에서 해결하세요! #주말에도 대출 가능!

군집 2
(SignUp)

→ 핀다, 처음이신가요? 핀다 사용에 얼굴꽃 핀다. 핀다 사용방법 바로가기 →

군집 3
(한도조회)

→ 금융이 쉬워진다. 대출 상환일 잊지마세요. 핀다가 매달 알려드릴께요.

01. 모델 결과 분석

부가서비스 위주 군집

군집 4
(부가서비스 -1)

→ 대출 관리 저희가 해드릴게요. 대출 관리 대신 해주는 서비스 바로 가기. 대출 신청도 실행단계에서부터 단 1분

군집 5
(부가서비스 -2)

→ 핀다로 신용점수 관리해서 더 이자율 낮은 대출로 갈아타자! 딱 맞는 대출관리 서비스 알아보러 가기 (월별 대환진단기능 + 대출상속안전장치, 대출 상환계좌 연동 등)

군집 6
(부가서비스 -3)

→ 요즘 물가도 오르는데 이자도 오르고.. 걱정 마세요! 대출기간 동안 원금과 이자 바로 알아보고 효율적으로 관리하는 꿀팁 대공개!

01. 모델 결과 분석

기타 군집

군집 7
(기타 -1)

이제 혼자하지마세요 핀다가 알아서 대출금 계산해서 신용점수 자동 관리 가능!

군집 8
(기타 -2)

요새 누가 내 돈써? 은행 돈으로 먹고 살자! 여러 제휴사 한번에 대출 조회하고 매일 웃음 꽃 핀다! 핀다하면? 인생 핀다.

군집 9
(기타 -3)

직장 다니느라 바쁘시죠? 핀다로 시간 절약해봐요~!

V

의의 및 한계

Ⅴ. 의의 및 한계

1. 전처리 과정에서 이상치와 누락된 값이 많아 전처리에 어려움이 있었음
2. 데이터의 양이 방대하여 모델 분석을 위한 사용 기기 환경에 제약이 있었음
3. 대량의 데이터를 접해볼 수 있었다는 점이 좋았음
4. 어려움에 부딪히고 이를 극복해나가는 과정에서 큰 성장과 경험을 이룰 수 있었음

VI

부록

Ⅵ. 부록

파일명에 약간의 차이가 있을 수 있지만, ppt 중간중간에 차이가 있는 부분은 언급해놓았습니다.

코드 및 파일 정리
• code 폴더 밑에 코드 정리(e.g 'data/loan_user_merge.csv' 불러오기)
• data 폴더 밑에 csv 파일들 정리
• 외부데이터 폴더 밑에 외부데이터 다운 받은 것들 저장

기존 제공 데이터 → EDA 코드 생성
1.loan_result.ipynb : 시각화 코드+특성 설명
2.user_spec.ipynb : 시각화 코드+특성 설명
3.log_data.ipynb : 시각화 코드+특성 설명

merge 데이터 - 예측
1.loan_user_merge.ipynb : loan_result, user_spec 각각 전처리하고 병합한 코드 → loan_user_merge_first.csv 생성
2.loan_user_merge(loan_user_merge_first에서 전처리한 코드).ipynb → loan_user_merge.csv 생성
3.외부데이터 폴더에 외부데이터 파일 따로 옮겨놓음.
4.include_all_merge(파생변수, 외부데이터 추가).ipynb → include_all_merge.csv 생성
5.data_reducing.ipynb → 데이터 변수 축소 과정 내용
6.RandomSampling_Model.ipynb → custom_pred 폴더 내의 파일들 생성
7.final_result.ipynb : 최종 분류 결과 생성(데이터분석분야_퓨처스부문_수퍼루키_평가데이터.csv)

merge 데이터 - 군집
1.log_events.ipynb → log_events.csv
2.log_include_all_merge.ipynb → clu_final.csv - (몇개 변수 제외)
3.log_cluster.ipynb
4.(특성분석한 것).ipynb

감 사 합 니 다