2022 제10회 빅콘테스트

데이터분석리그 퓨처스부문 최종보고서

앱 사용성 데이터를 통한 대출신청 예측분석

Team KUBIG B

김상옥 quadrat1c@korea.ac.kr

노연수 1020nys@korea.ac.kr

이수찬 eliot1113@korea.ac.kr

INDEX

Chapter A.
데이터 활용 및 EDA

Ch. A | 00

메타데이터 요약

사용자 신용정보

user_spec.csv

- 생년월일, 성별 등 개인사항
- 소득, 근로형태 등 직업 변수
- 기존 대출, 개인회생 등 신용 변수

대출 결과

loan_result.csv

- **신청서 및 상품 id (Key)**
- **신청 여부(Yes/No) (Target)**
- 승인한도 및 금리 등 대출 정보

사용자 로그

log_data.csv

- **Finda 어플리케이션 사용 정보**
- 앱 실행, 로그인, 신용정보 및 한도조회 등 활동 기록

Ch. A | 01 활용 라이브러리

수치 및 데이터프레임 연산

머신러닝 모델 피팅 및 평가

데이터 EDA &결과 시각화

Ch. A | 02 데이터셋 임포트, EDA

사용자 신용정보

user_spec.csv

140만개 가량의 row, 일부 열에서
10만개~100만개 정도의 결측치

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1394216 entries, 0 to 1394215
Data columns (total 17 columns):
 #   Column                          Non-Null Count    Dtype
---  ------                          --------------    -----
 0   application_id                  1394216 non-null  int64
 1   user_id                         1394216 non-null  int64
 2   birth_year                      1381255 non-null  float64
 3   gender                          1381255 non-null  float64
 4   insert_time                     1394216 non-null  object
 5   credit_score                    1289101 non-null  float64
 6   yearly_income                   1394126 non-null  float64
 7   income_type                     1394131 non-null  object
 8   company_enter_month             1222456 non-null  float64
 9   employment_type                 1394131 non-null  object
 10  houseown_type                   1394131 non-null  object
 11  desired_amount                  1394131 non-null  float64
 12  purpose                         1394131 non-null  object
 13  personal_rehabilitation_yn      806755 non-null   float64
 14  personal_rehabilitation_complete_yn  190862 non-null   float64
 15  existing_loan_cnt               1195660 non-null  float64
 16  existing_loan_amt               1080442 non-null  float64
dtypes: float64(10), int64(2), object(5)
memory usage: 180.8+ MB
```

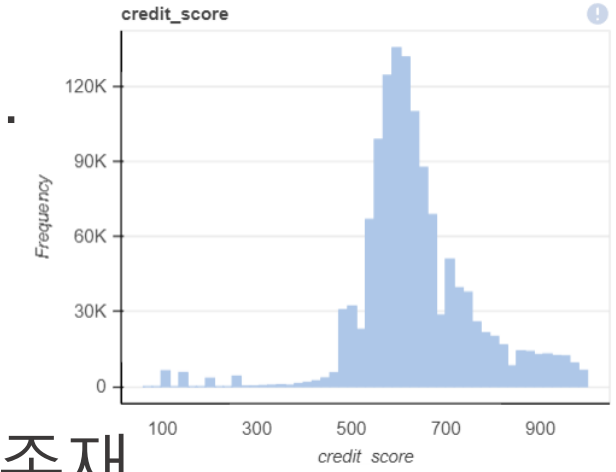
Ch. A | 02

데이터셋 임포트, EDA

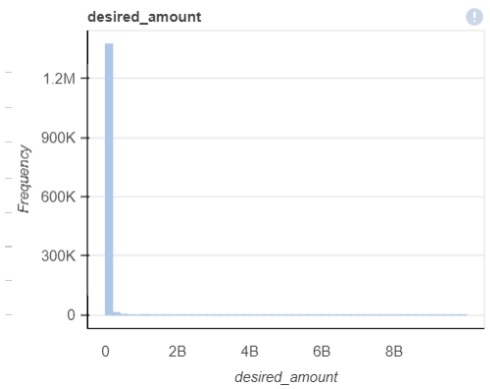
사용자 신용정보

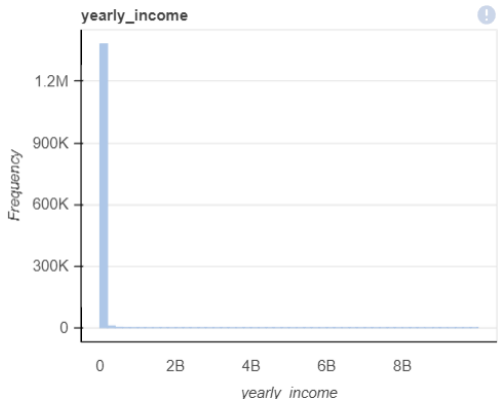
user_spec.csv

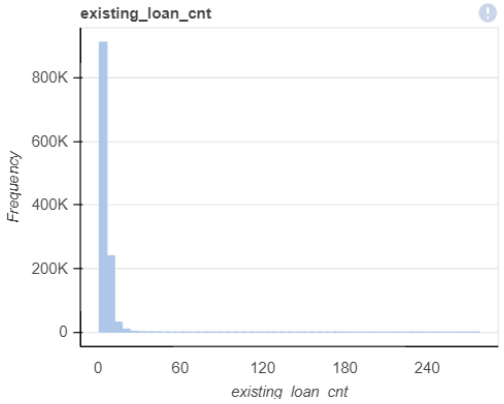
고르게 분포하는 변수도 있으나…
(credit_score 등)



금액 관련 변수는 극단적 이상치 존재
(yearly_income, desired_amount, existing_loan_amt 등)

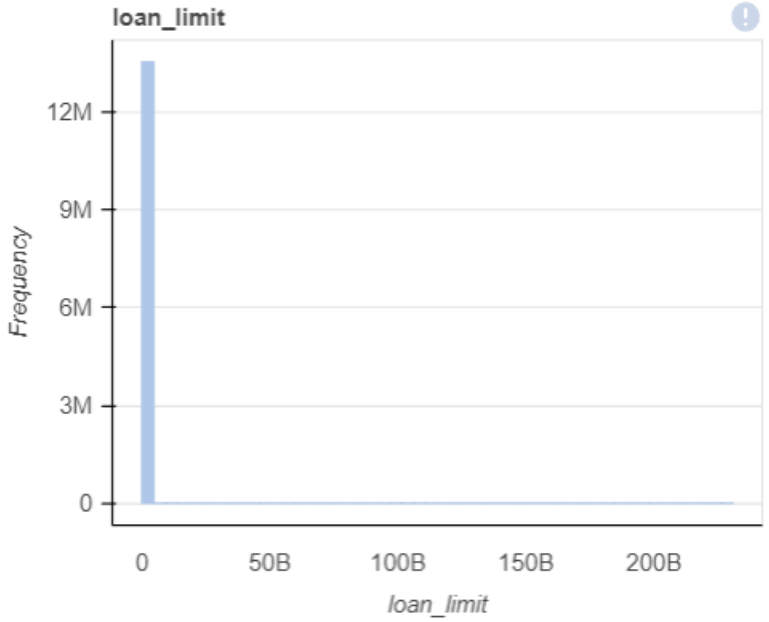


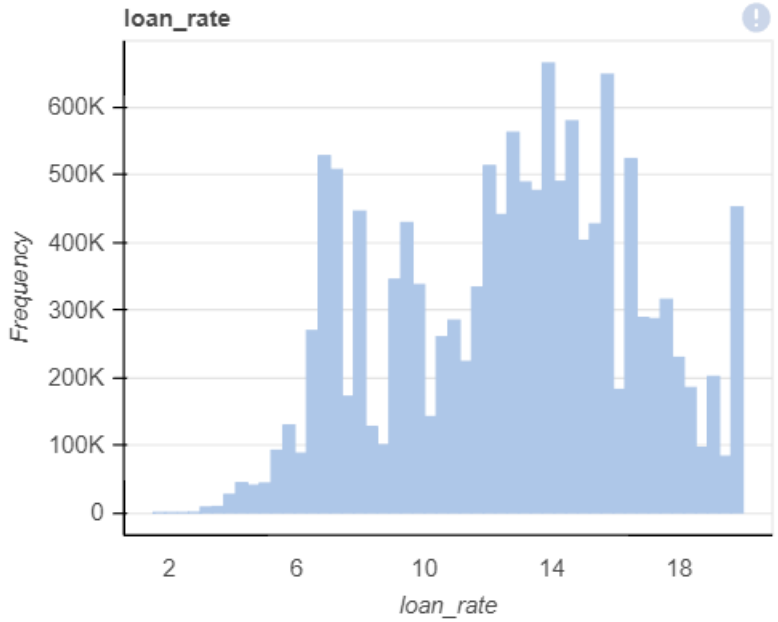




Ch. A | 02 데이터셋 임포트, EDA

대출 결과
loan_result.csv





타겟 변수(is_applied)와 **대출 정보(loan_limit/_rate),**
마찬가지로 **금액 변수의 극단값** 확인

Ch. A | 02

데이터셋 임포트, EDA

사용자 로그

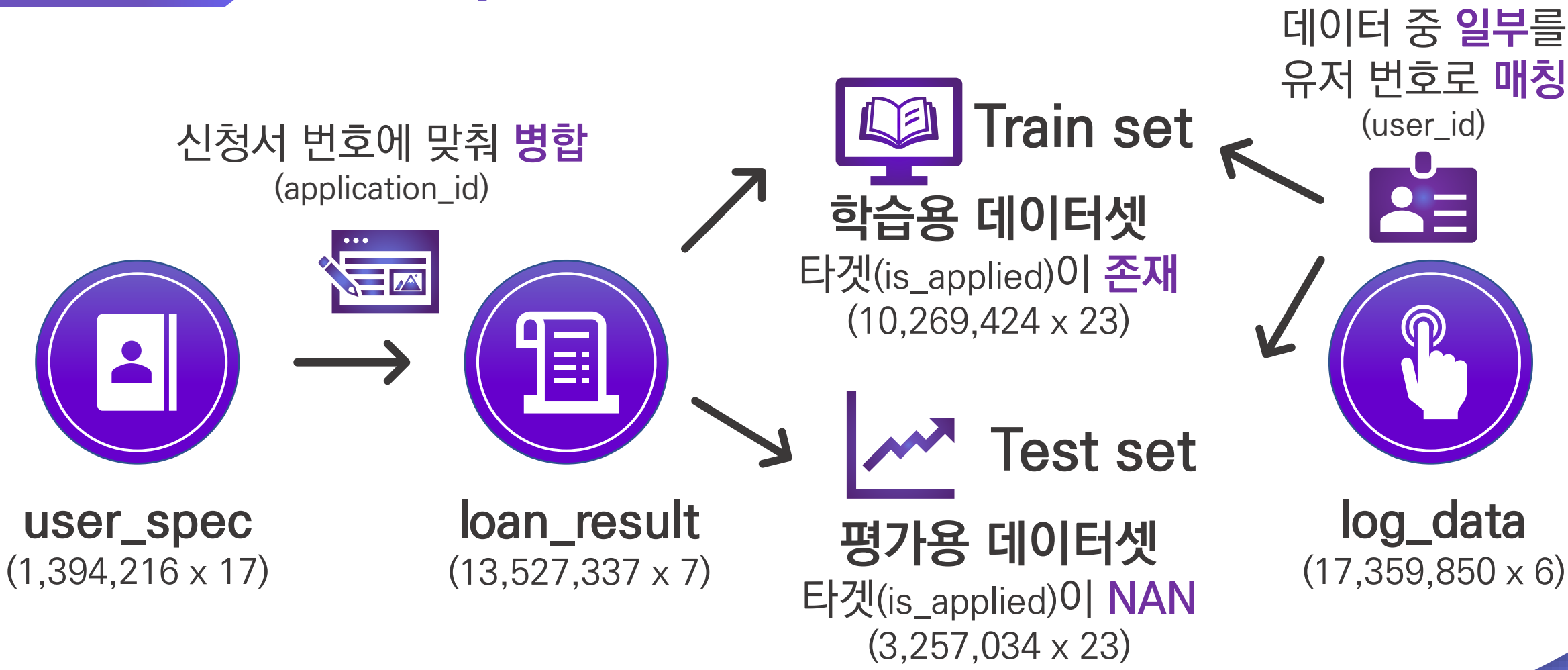
log_data.csv

	event	timestamp	mp_os	mp_app_version	date_cd
count	17843993	17843993	17843013	17183396	17843993
unique	11	6879764	4	259	122
top	OpenApp	2022-04-11 11:40:30	Android	3.14.0	2022-06-27
freq	3460762	23	12331688	2339899	267738

Finda 앱에서 **유저 사용 기록** 및 시간, 기기 OS와 앱 버전 등
1,700만개 가량의 row가 있는 **대형 데이터**

Ch. A | 03

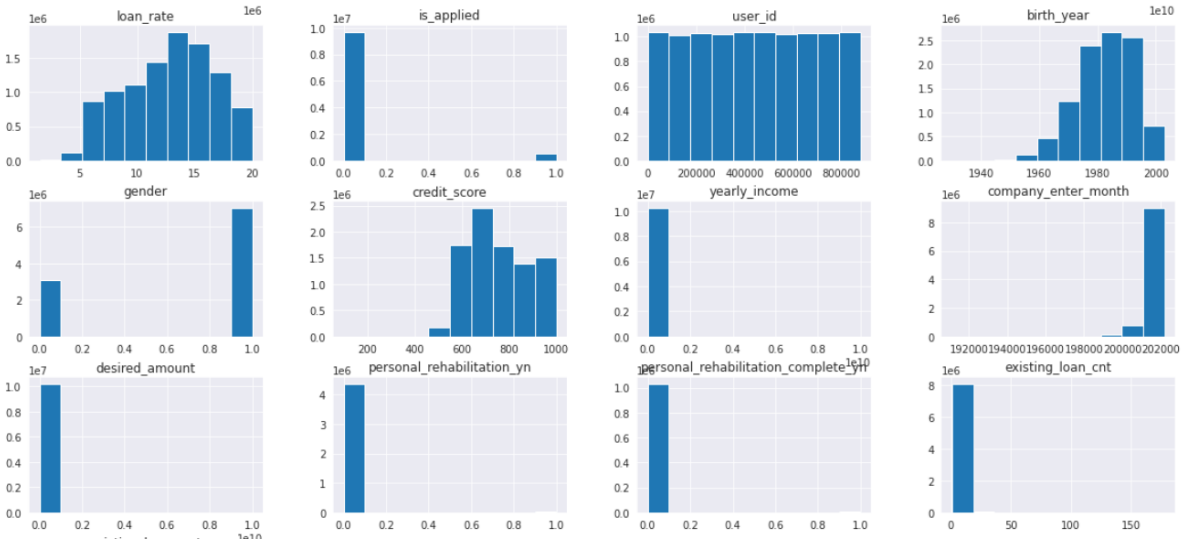
Train / Test 데이터 추출

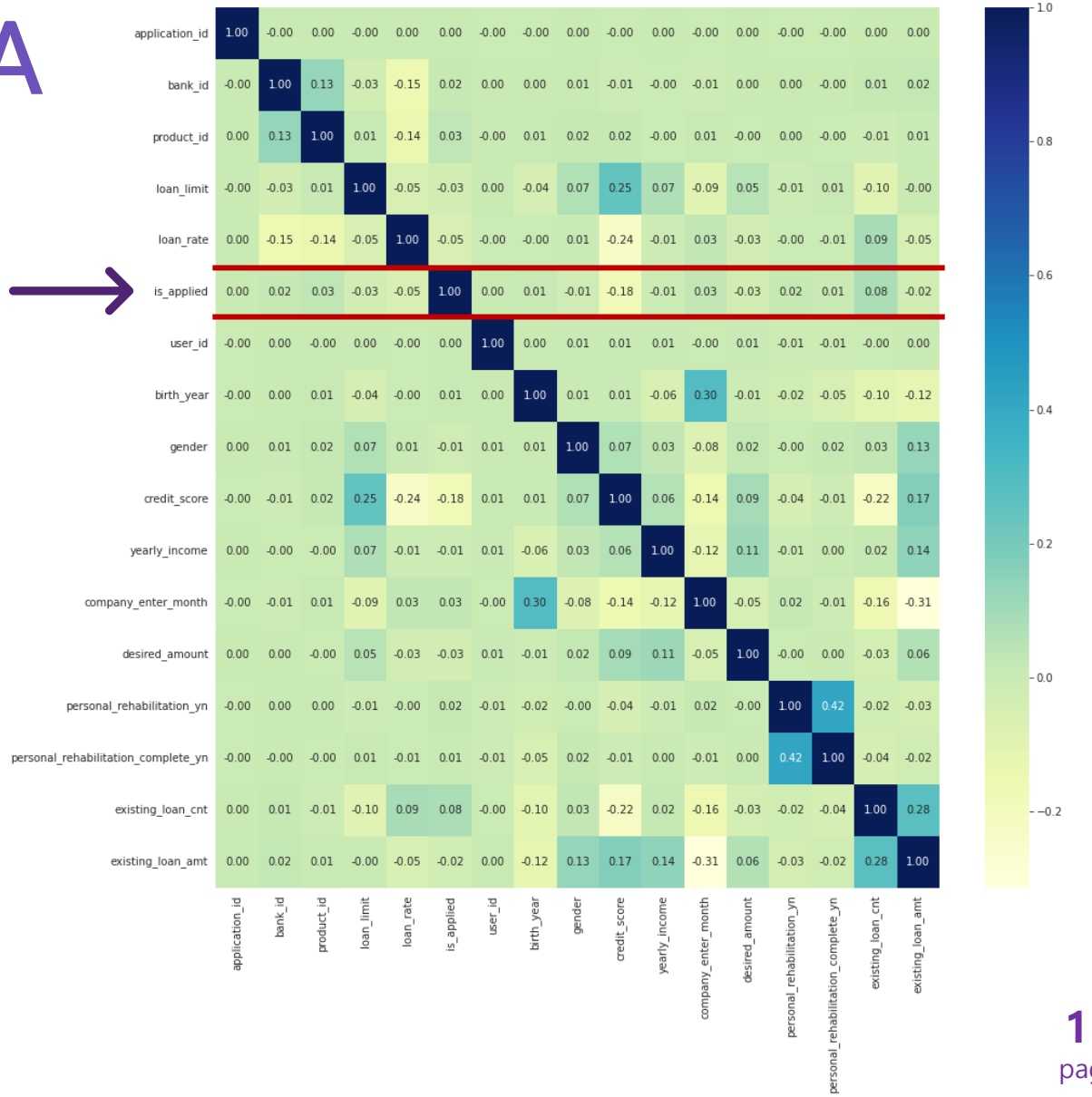


Ch. A | 03 Train 데이터 EDA

Correlation Matrix 상에서
타겟과의 상관계수 전반적으로 **낮은 경향**

loan_result&user_spec 병합 후에도
일부 극단적 분포





Chapter B.
데이터 전처리

Ch. B | 04 Train 결측치 제거, '무응답' 범주 추가

학습용 데이터셋(Train)에만
결측치가 있는 경우,
비율이 매우 낮으므로 결측행 제거

```
[ ]  #train에만 있는 결측치 비율
     pd.options.display.float_format = '{:.6f}'.format
     train_loan[na_train].isnull().sum()/len(train_loan)
```

```
insert_time        0.000011
houseown_type      0.000011
employment_type    0.000011
purpose            0.000011
income_type        0.000011
user_id            0.000011
desired_amount     0.000011
dtype: float64
```

개인회생 관련 변수 더미화,
결측치의 경우 기록 없음으로 간주해 0으로 처리

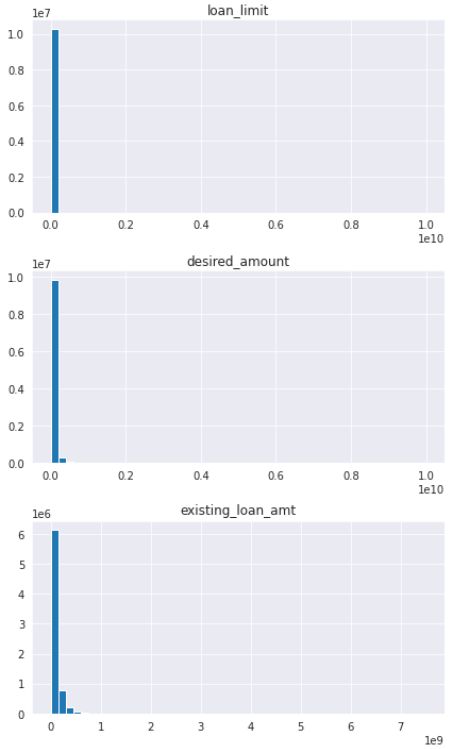
```
#trian 데이터에서 'rehabilitation_complete','rehabilitation_incomplete' 변수 고유값 확인
train_loan2[['rehabilitation_complete','rehabilitation_incomplete']].value_counts()
```

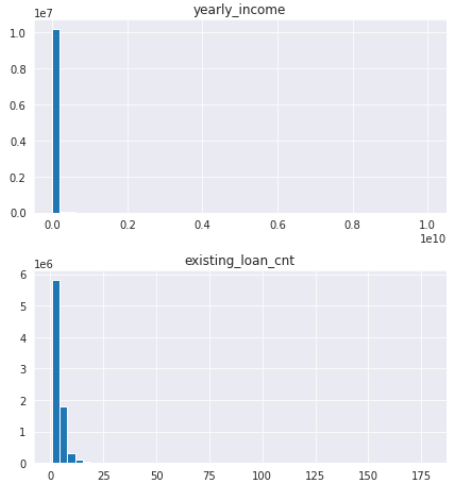
```
rehabilitation_complete  rehabilitation_incomplete
0                        0                          9226739
                         1                          1033045
1                        0                             4110
dtype: int64
```

9226739 ← 개인회생 해당 없음
1033045 ← 개인회생 진행 중
4110 ← 개인회생 완료

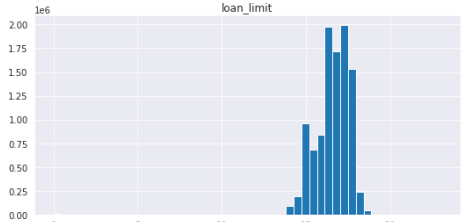
Ch. B | 04 로그변환, 극단적 이상치 완화

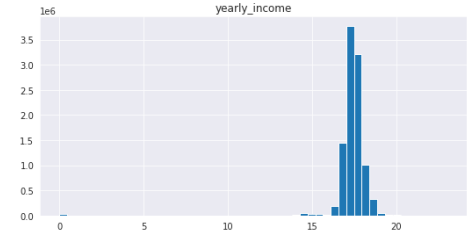
기하급수적 극단값 및 skewness 문제 있는 변수는 로그변환
(loan_limit,yearly_income,desired_amount,existing_loan_cnt,existing_loan_amt)

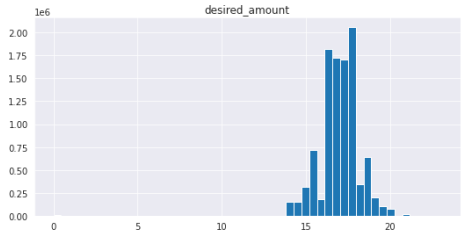


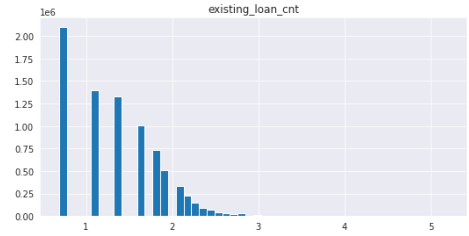


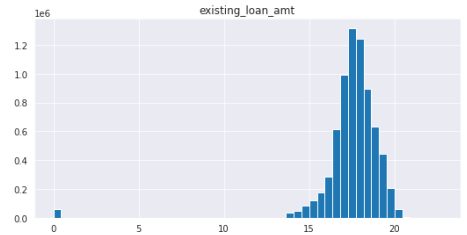
$ln(1+p)$로 변환
(np.log1p)











Ch. B | 04 더미변수 One-Hot Encoding

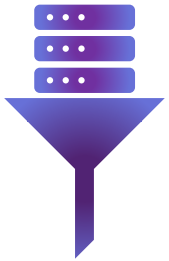
범주형 변수인 **소득/고용/주거유형, 대출 목적** 모두
(income_type, employment_type, houseown_type, purpose)

범주 간 순서가 존재하지 않으므로.
One-hot encoding이 적절하다고 판단, 실행

One-hot encoding:
n개의 범주를 n개의 비트(0,1) 벡터로 표현,
서로 다른 범주를 독립적인 의미로 사용 가능

그러나 부작용으로 **Feature 수가 과하게 늘어**나는 문제 발생

application_id
loanapply_insert_time
bank_id
product_id
loan_limit
loan_rate
is_applied
user_id
birth_year
gender
insert_time
credit_score
yearly_income
company_enter_month
desired_amount
existing_loan_cnt
existing_loan_amt
rehabilitation_complete
rehabilitation_incomplete
income_type_EARNEDINCOME
income_type_EARNEDINCOME2
income_type_FREELANCER
income_type_OTHERINCOME
income_type_PRACTITIONER
income_type_PRIVATEBUSINESS
employment_type_계약직
employment_type_기타
employment_type_일용직
employment_type_정규직
houseown_type_기타가족소유
houseown_type_배우자
houseown_type_자가
houseown_type_전월세
purpose_BUSINESS
purpose_BUYCAR
purpose_BUYHOUSE
purpose_ETC
purpose_HOUSEDEPOSIT
purpose_INVEST
purpose_LIVING
purpose_SWITCHLOAN
purpose_기타
purpose_대환대출
purpose_사업자금
purpose_생활비
purpose_자동차구입
purpose_전월세보증금
purpose_주택구입
purpose_투자



차원의 저주 방지 위해
타겟과의 **상관계수 0.03 미만 필터링,**
19개 변수 제거

Ch. B | 04 K-Means 클러스터링 및 결측치 대치

	index	total
13	existing_loan_cnt	1.935517
11	company_enter_month	1.853272
14	existing_loan_amt	1.718721
6	birth_year	1.523486
9	credit_score	1.499850
7	gender	1.254627
25	houseown_type_자가	1.035266
10	yearly_income	0.833370
12	desired_amount	0.751290
24	houseown_type_기타가족소유	0.561244
2	loan_limit	0.555600

결측치 존재 변수와의
상관계수 절대값 합
상위 5개 변수 선택,
(회색: 결측치 존재 변수 자신)

K-Means 클러스터링 후
군집별 중앙값으로 결측치 대치

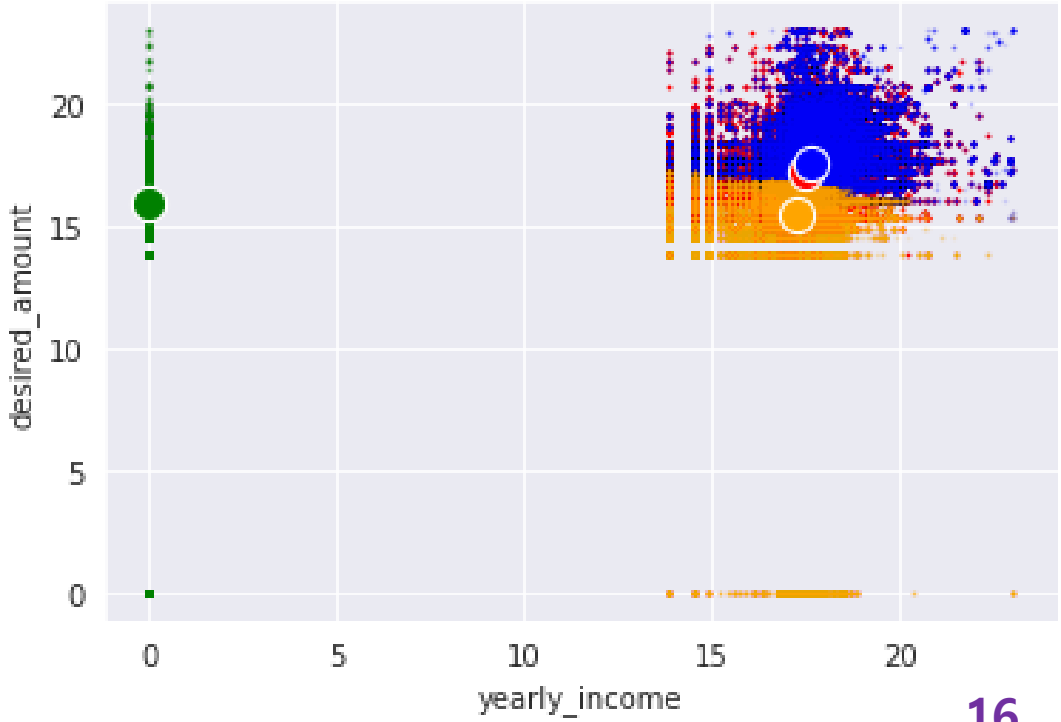
```
cols = ['yearly_income','desired_amount','houseown_type_자가','houseown_type_기타가족소유','loan_limit']
groups.fit(train_loan4[cols])
```

```
KMeans(n_clusters=4)
```

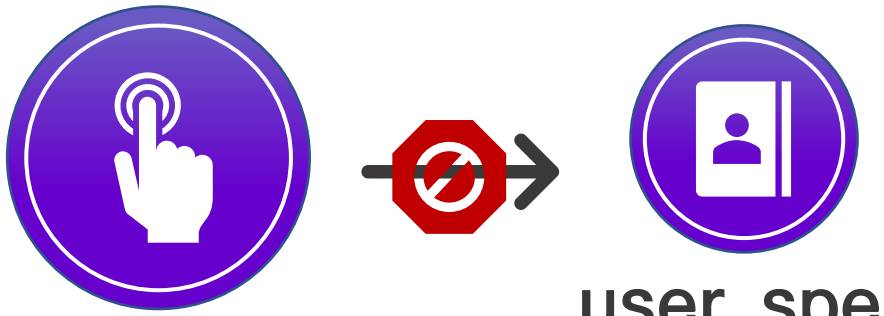
```
train_loan4['group'] = groups.labels_
train_loan4['group'].value_counts()
```

```
1    4578224
2    2830815
0    2826319
3      28536
Name: group, dtype: int64
```

군집 시각화, 5개 변수 이용했으므로
2차원 상에서는 **경향성 차이만** 드러남



Ch. B | 05 로그데이터 축소



log_data
(17,359,850 x 6)

user_spec
(1,394,216 x 17)

로그 데이터 특성상 대부분의 경우
복수의 행이 한개의 user_id에 대응하므로
user_id 기준 병합 사실상 불가

	user_id	event	timestamp	mp_os	mp_app_version	date_cd
0	576409	StartLoanApply	2022-03-25 11:12:09	Android	3.8.2	2022-03-25
1	576409	ViewLoanApplyIntro	2022-03-25 11:12:09	Android	3.8.2	2022-03-25
2	72878	EndLoanApply	2022-03-25 11:14:44	Android	3.8.4	2022-03-25
3	645317	OpenApp	2022-03-25 11:15:09	iOS	3.6.1	2022-03-25
4	645317	UseLoanManage	2022-03-25 11:15:11	iOS	3.6.1	2022-03-25
5	640185	UseLoanManage	2022-03-25 11:41:53	iOS	3.6.1	2022-03-25
6	640185	ViewLoanApplyIntro	2022-03-25 11:42:38	iOS	3.6.1	2022-03-25
7	640185	UsePrepayCalc	2022-03-25 11:43:07	iOS	3.6.1	2022-03-25
8	640185	UseLoanManage	2022-03-25 11:43:57	iOS	3.6.1	2022-03-25
9	640185	UseLoanManage	2022-03-25 11:44:04	iOS	3.6.1	2022-03-25

Ch. B | 05 로그데이터 축소

따라서 병합하는 대신,
user_id별 event 실행 여부 추출 →

↓ **event별 타겟과의 상관계수** 체크
전반적으로 높지는 않은 경향…

```
cor3 = train_loan5[list(log_data['event'].unique())].corrwith(other = train_loan5['is_applied'])
cor3 = cor3.reset_index()
cor3.sort_values(0, ascending=False)
```

	index	0
0	StartLoanApply	0.041596
2	EndLoanApply	0.039095
3	OpenApp	0.036774
4	UseLoanManage	0.035475
1	ViewLoanApplyIntro	0.025215
9	SignUp	0.024246
7	CompleteIDCertification	0.024220
10	GetCreditInfo	0.023731
6	Login	0.020620
5	UsePrepayCalc	0.000309
8	UseDSRCalc	-0.007152

```
[ ]  for i in list(log_data['event'].unique()):
       print(i)
       log_limit = log_data[log_data['event']==i]
       log_limit = log_limit[['user_id']]
       log_limit[i] = 1
       log_limit = log_limit.drop_duplicates(ignore_index = True)
       log_limit = log_limit.drop(log_limit[~log_limit['user_id'].isin(train_user['user_id'])].index)
       print(log_limit.shape)
       train_user = train_user.merge(log_limit, on='user_id', how='left')
       train_user = train_user.fillna(0)
       train_loan5[i] = train_user[i]
       train_user = train_loan5[['user_id']]
```

```
StartLoanApply
/usr/local/lib/python3.7/dist-packages/ipykernel_launcher.py:5: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html
  """
(205694, 2)
ViewLoanApplyIntro
(208407, 2)
EndLoanApply
(211758, 2)
OpenApp
(199218, 2)
UseLoanManage
(159545, 2)
UsePrepayCalc
(3098, 2)
Login
(169612, 2)
CompleteIDCertification
(203151, 2)
UseDSRCalc
(2373, 2)
SignUp
(12463, 2)
GetCreditInfo
(205325, 2)
```

Ch. B | 05 로그데이터 축소

타겟과의 상관계수가 그나마 있는
대출 신청 완료 여부 (EndLoanApply),

그 다음으로 유의하면서 상호 상관성이 과하지 않은
대출관리 서비스 이용 여부 (UseLoanManage) 를 최종 활용

```
[ ]  Use = ['EndLoanApply', 'UseLoanManage']
     Not_Use = list(set(log_data['event'].unique()) - set(Use))
     train_loan6 = train_loan5.drop(Not_Use, axis=1)
     train_loan6.info()
```

↑

```
[ ]  for i in Use:
       print(i)
       test_user = test_loan4[['user_id']]
       log_limit = log_data[log_data['event']==i]
       log_limit = log_limit[['user_id']]
       log_limit[i] = 1
       log_limit = log_limit.drop_duplicates(ignore_index = True)
       log_limit = log_limit.drop(log_limit[~log_limit['user_id'].isin(train_user['user_id'])].index)
       print(log_limit.shape)
       test_user = test_user.merge(log_limit, on='user_id', how='left')
       test_user = test_user.fillna(0)
       test_loan4[i] = test_user[i]
```

```
EndLoanApply
(211758, 2)
UseLoanManage
(159545, 2)
```

← 활용할 event를
평가용 데이터에도 매치

Chapter C.
활용 알고리즘과 예측 결과

Ch. C | 06 기초 예측과 모델 검토

모델 검토용 데이터 분할

Train 세트 내부에서 자체 train, 자체 test 분리(train_test_split)
자체 train에만 불균형 해소(SMOTE)

로지스틱 회귀 검토 및 피팅

피팅이 간단하고, 확률 계산에 장점 있음
자체 검토 시 f1 score 0.168 수준…**기각**

```
print(lr_conf_matrix)
print("정확도: ", lr_acc_score*100, "₩nF1 score:", lr_f1_score)
```

```
[[1776334 1136689]
 [  46381  119765]]
정확도:  61.57826998128391
F1 score: 0.16837480669197244
```

랜덤 포레스트 검토 및 피팅

고차원 데이터에 강한 비모수적 방식인 결정 트리를 앙상블,
오버피팅 가능성이 낮고 비선형적 데이터에 강점
자체 검토 시 f1 score 0.256 수준

```
print(rf_conf_matrix)
print("정확도: ", rf_acc_score*100, "₩nF1 score:", rf_f1_score)
```

```
[[2870639   42384]
 [ 135468   30678]]
정확도:  94.22402602780166
F1 score: 0.25649643824621243
```

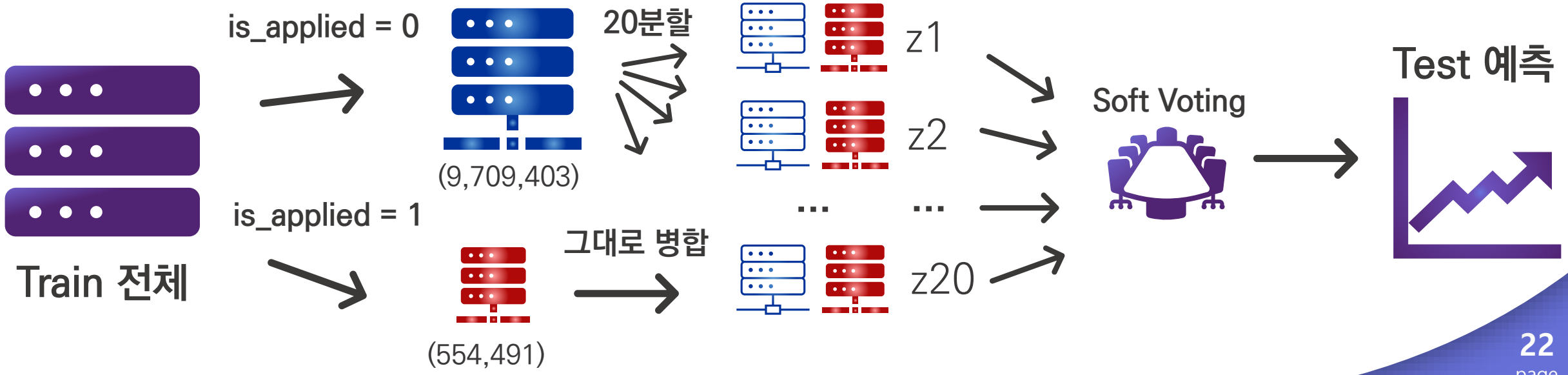
*모델의 장단점은 Wang, Y., Zhang, Y., Lu, Y., & Yu, X. (2020). A Comparative Assessment of Credit Risk Model Based on Machine Learning ——a case study of bank loan data. *Procedia Computer Science*, *174*, 141–149. https://doi.org/10.1016/j.procs.2020.06.069 을 참고하였음

Ch. C | 07 데이터 분할과 AutoML 모델링

자체 검토 결과로부터, **랜덤 포레스트 비롯한 결정 트리 계열의 모델** 사용 결정

모델 자체평가, 블렌딩, f1 점수 최적화 기능이 존재하는 **AutoML로 최종 모델** 도출

하드웨어 문제로 **데이터를 분할해서** 학습시킨 다음 각 모델의 예측 결과(확률)을 Soft Voting
불균형 완화 위해 타겟이 No인 데이터 20분할, Yes인 경우를 각 분할에 그대로 병합



Ch. C | 07 예측 결과 및 해석

Decision Tree, Random Forest, Extra Tree 블렌딩해 예측

분할 일부 예측 결과:

```
best_3 = compare_models(sort = 'f1', include = ['dt', 'rf', 'et'], n_select = 3, fold=3)
```

	Model	Accuracy	AUC	Recall	Prec.	F1	Kappa	MCC	TT (Sec)
et	Extra Trees Classifier	0.8862	0.9610	0.8629	0.9188	0.8900	0.7725	0.7741	129.6800
rf	Random Forest Classifier	0.8802	0.9583	0.8637	0.9072	0.8849	0.7602	0.7612	169.1967
dt	Decision Tree Classifier	0.8320	0.8320	0.8328	0.8492	0.8409	0.6630	0.6632	10.6300

```
blended = blend_models(estimator_list = best_3, fold = 3, method = 'soft',optimize = 'f1')
```

Fold	Accuracy	AUC	Recall	Prec.	F1	Kappa	MCC
0	0.8584	0.9534	0.8430	0.8859	0.8639	0.7165	0.7175
1	0.8595	0.9535	0.8428	0.8881	0.8648	0.7189	0.7199
2	0.8570	0.9524	0.8420	0.8842	0.8626	0.7138	0.7147
Mean	0.8583	0.9531	0.8426	0.8861	0.8638	0.7164	0.7174
Std	0.0010	0.0005	0.0004	0.0016	0.0009	0.0021	0.0021

이후 분할별 예측
Soft Voting

Ch. C | 08

결론 및 제언

대형 데이터에 대한 효율적 병합 Method 필요

loan_result의 1,000만개 타겟을 학습하고 300만개를 예측하는 대형 문제,
application_id, user_id를 **키로 매칭, 키 중복으로 인한 거대화 문제를 완화**했으나
추후 더 진행된다면 보다 근본적인 효율화 방법 모색할 필요

고차원 데이터의 특성에 적합한 모델 탐색

고차원 데이터 특성에 따라 **비모수적 Tree 모델이 실제로 성능이 더 나은 경향** 확인
지속적인 예측 시스템은 비모수적 가정 하에서 구성되는 것이 적절하다고 사료됨

앱에서 사용자의 이벤트 발생 주목

특정 이벤트 발생 여부 뿐 아니라 빈도, 시간 등 다른 변인 또한 주목해야 할 것으로 파악됨

이상으로 발표를 마칩니다.
감사합니다!