Predict the credit risk

by using Gradient Boosting methods

1조 Final Project

in Financial Data Analysis with Big Data for Financial Engineering

20213836 MFE 김재용 20213838 MFE 김창균 20213843 MFE 배준환

20213858 MFE 조규영 20213861 MFE 한석호

Abstract

이번 프로젝트의 목적은 은행에서 대출 상환 능력이 있다고 판단되어 대출 승인이 난 후의 데이터를 가지고 차주의 채무 불이행을 예측하는 것을 목표를 하고 있다. 즉, 차주의 신용 리스크 예측, 채무 불이행을 예측하는데 초점을 맞추고 있다. 이러한 프로젝트 목표는 은행 관점에서는 은행 내부의 대출 승인 조건에 대해 평가할 수 있을 뿐만 아니라 은행의 리스크 관리에 중요한 부분이다.

은행의 대출 승인을 받은 후의 예측이기에 데이터의 불균형은 어느정도 예상되는 부분이었고 불균형의 정도에 관해서 논의를 해본 결과 불균형이라고 판단하는 정도가 개개인의 주관이 들어가고 이로 인해 모델의 성과가 달라질 수 있을 것이라 생각했다.

위의 논의 결과 Over 혹은 Under Sampling을 사용한 모델과 사용하지 않은 모델의 필요성을 비교하면서 성과가 개선되는지 혹은 어떠한 차이가 발생하는지에 대한 의구심을 가지게 되어서 위의 방법론이 실제 데이터 분석에 어느 정도의 중요성을 가지고 있는지 평가 및 비교하는데 초점을 두고 프로젝트를 진행했다.

Table of contents

I. Introduction and summary explanation of feature variables.

II. Data processing

III. Model selection with pure data

IV. Results of pure data

V. Analysis with over/under sampling data

VI. PCA Analysis

VII. Catboost

VIII. Conclusion

I. Introduction and summary explanation of feature variables

i. Feature varibales

- person_age : 나이
- person_income : 연 소득
- persiongomeownership : 집 소유 유무(rent, mortgage, own, other)
- persion emp length : 근속연수
- loan_intent : 대출 목적(education, medical, venture, personal, debt consolidation, home improvement)
- loan_grade : 대출 등급 (A,B,C,D,E,F,G)
- loan_amnt : 대출금
- loan_int_rate : 대출 이자(%)
- loan_percent_income : 대출이자 / 월수입
- cb_person_default_on_file : 과거 연체 유무(N ,Y)
- cb_person_cred_hist_length : 연체 시 신용불량 기간(day)

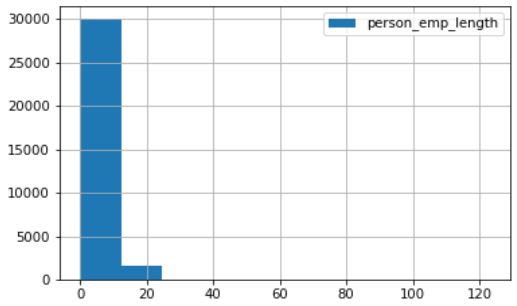
ii. Target variable

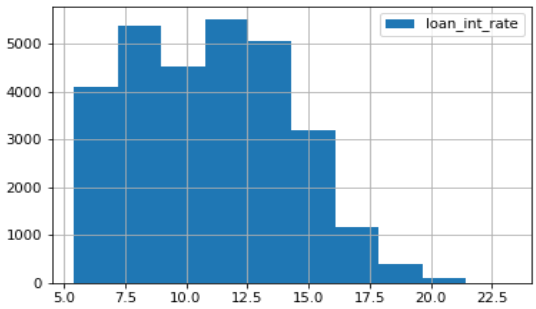
- Loan status : Loan status
- (0 is non default 1 is default)

II. Data Preprocessing

i. 결측치 처리

각 특성변수마다 약 3만2천개의 데이터를 포함하고 있다. 이 중 근속연수(persion emp length) 데이터에서 7.4%(약 2400개)와 대출 이자(loan_int_rate) 데이터에서 7.4%(약 2400개)의 결측치가 존재하는 것을 확인했다. 결측치를 채우는 방법으로는 근속연수와 대출 이자에 대한 히스토그램 그래프를 확인 후 대출 이자의 결측치는 평균값으로, 근속연수의 결측치는 최빈값으로 대체해 진행했다.

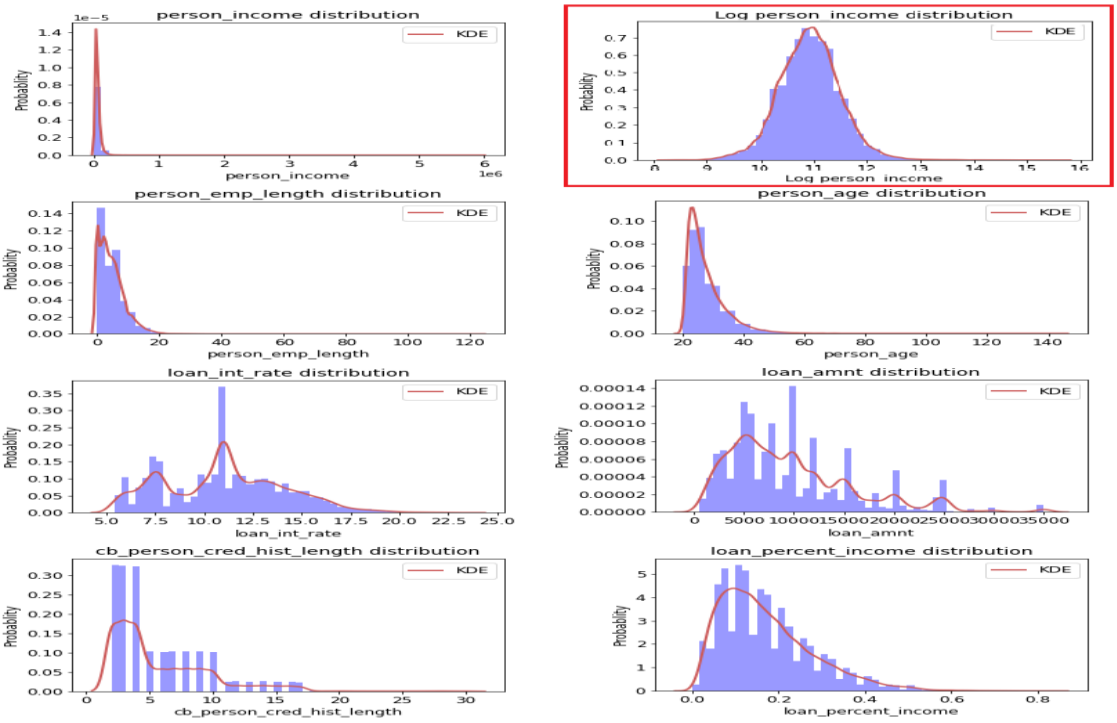


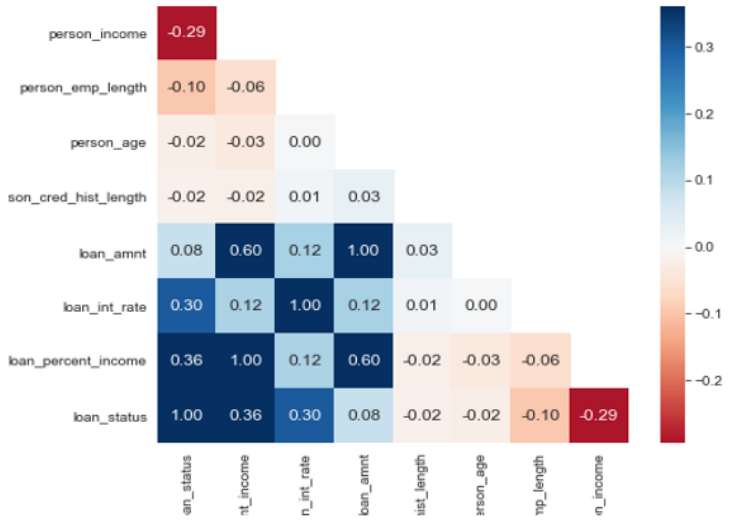


ii. 연속형 데이터(분포)

전체 특성변수 중 연속형 자료는 나이(person_age), 연 소득(person_income), 근속연수(persion emp length), 대출금(loan_amnt), 대출 이자(loan_int_rate), 월 소득에 대한 대출 이자 비중(loan_percent_income) 과거 연체 시 연체기간으로 총 7개의 연속형 데이터가 있고 각 특성변수에 대해서 각각의 분포를 확인 후 연 소득에 대한 분포를 확인해 보았을 때, 오른쪽 긴 꼬리 분포를 나타남을 확인했고 가격 데이터의 왜도(skewness)와 첨도(Kurtosis)를 줄여 정규성을 높이기 위해 로그 변환을 취해 적용했다. 나머지 연속형 특성변수에 대해서도 로그 변환을 시도해 보았지만 눈에 띄게 정규성이 커지는 효과가 없었을 뿐만 아니라 큰 의미를 찾거나 성과를 보지 못하여 최종적으로 연 소득에 대해서만 로그 변환를 취해서 진행했다.

이후, 히트맵을 통해서 연속형 특성변수들 간의 상관계수를 확인한 결과 대출금과 월 소득에 대한 대출 이자 비중 변수들 간의 상관계수가 0.60으로 가장 크게 나타나는 것을 확인할 수 있고 나머지 특성변수들 간의 상관계수는 크지 않다는 것을 확인했다. VIF확인 결과 다중공산성 문제가 크게 유의하지 않아 특성변수를 특별히 제거하지 않고 진행했다.

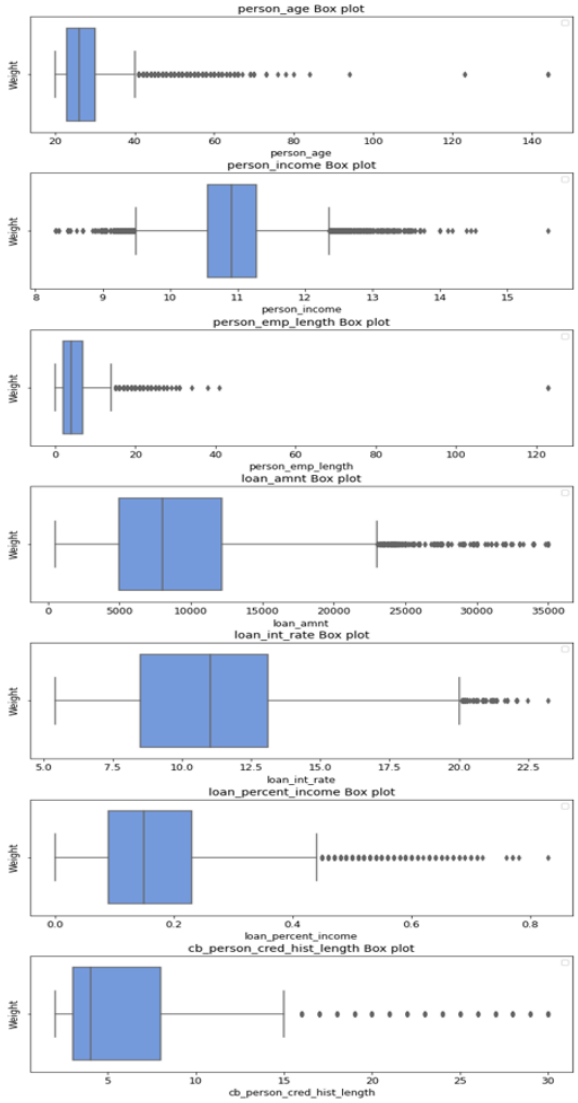




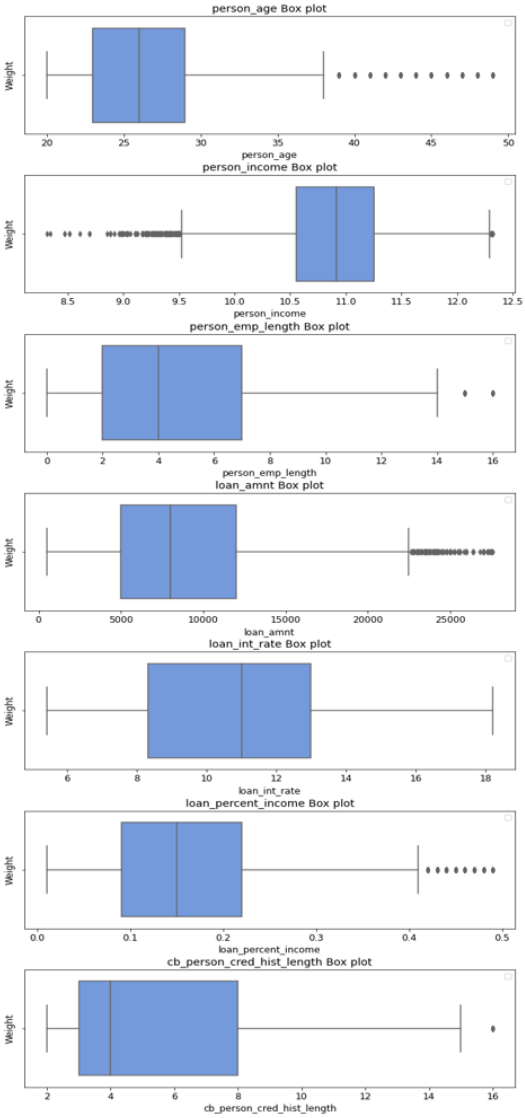
iii. 이상치 제거

모든 연속형 특성변수에 대해 상자그림을 확인해 보면 이상치의 존재를 확인할 수 있다. 대부분의 연속형 특성변수를 보면 상위 극단 값에 대한 이상치가 존재하고 상위 1%을 기준으로 극단 값을 제거했다. 왼쪽의 그래프은 극단 값을 제거하기 전의 상자 그림이고 오른쪽 그래프는 극단 값을 제거한 뒤의 상자그림이다.

극단 값 제거 전



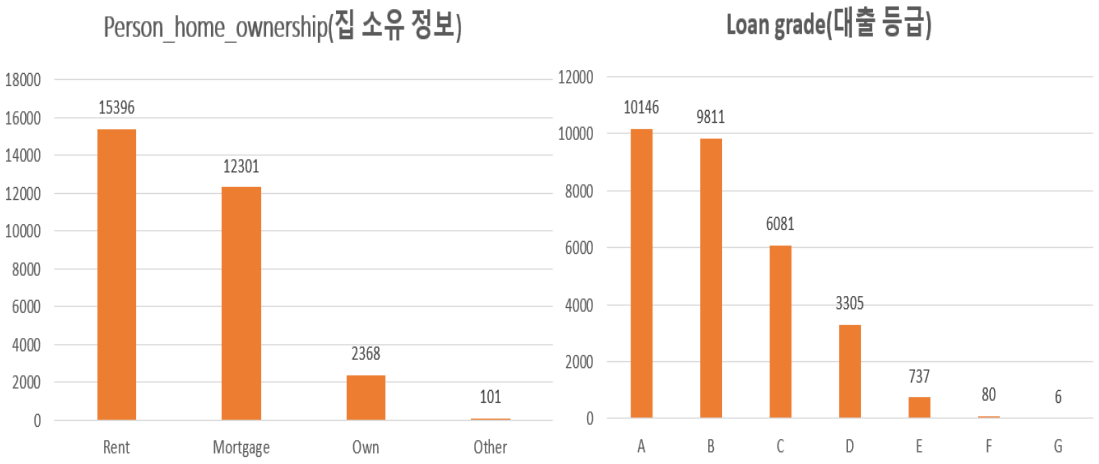
극단 값 제거 후

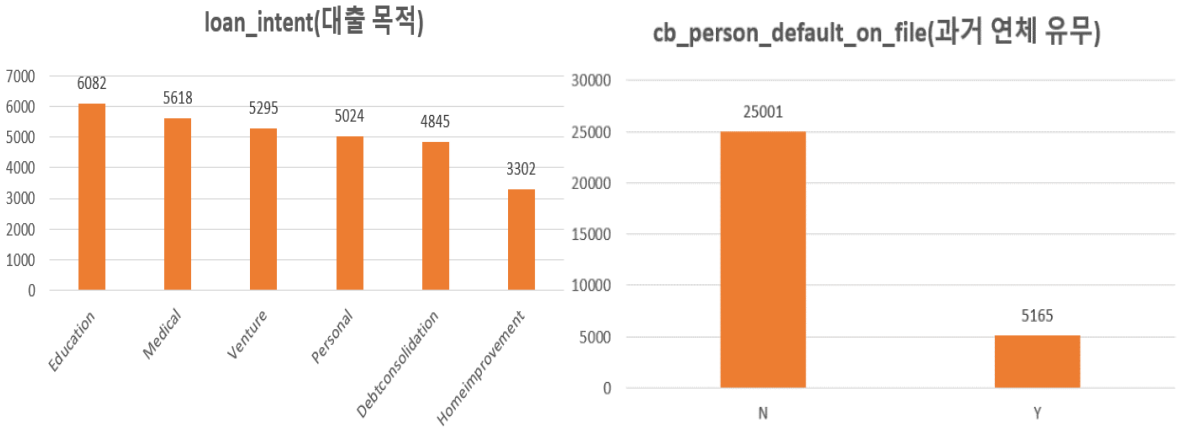


iv. 범주형 자료

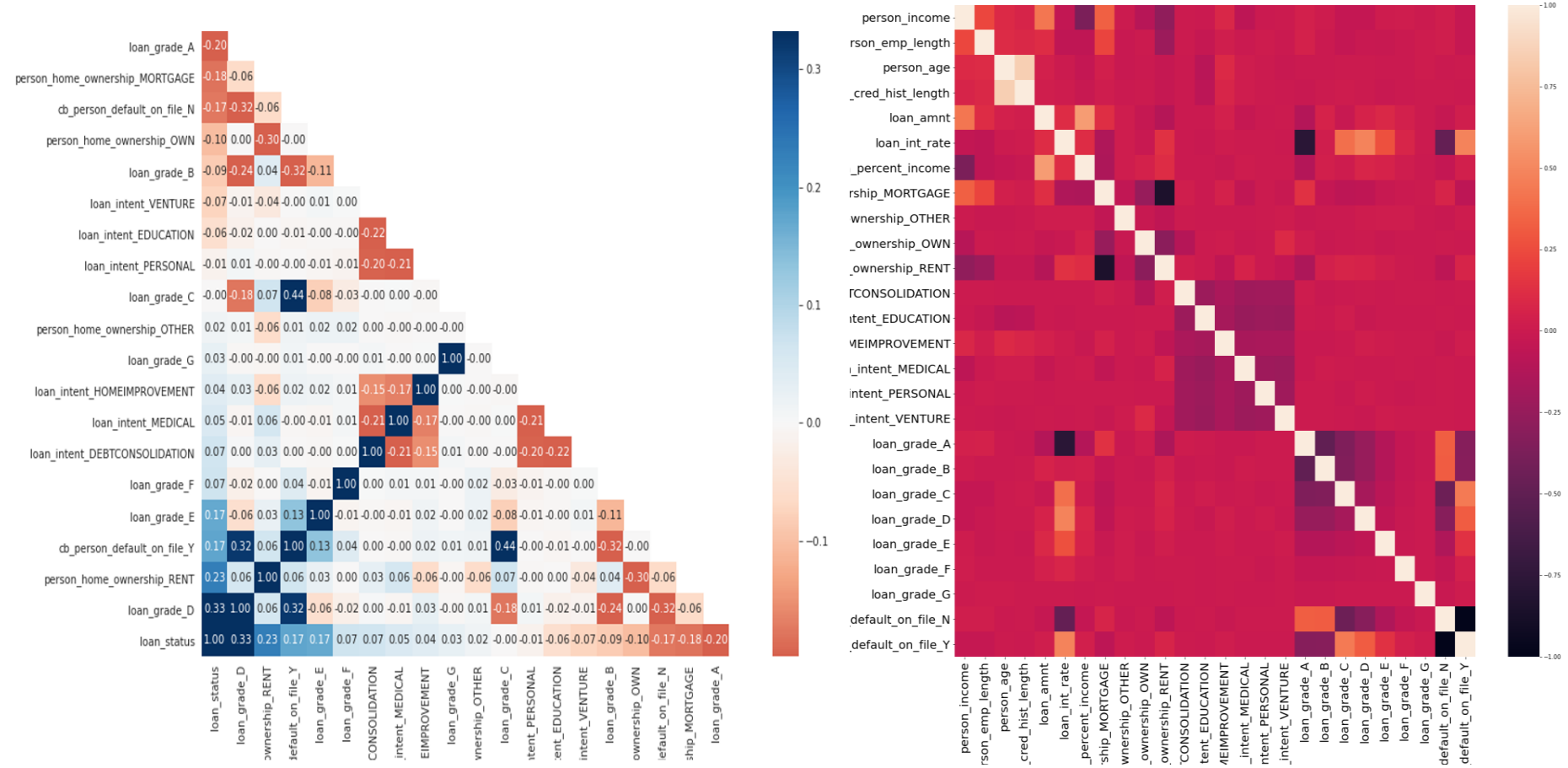
전체 특성 변수 중에서 범주형 데이터는 집 소유 여부, 대출목적, 대출 등급, 과거 연체 유무로 총 4개의 범주형 데이터가 존재한다. 4개의 범주형 데이터는 연속적인 특징을 갖지 않다고 판단하여 4개의 범주형 데이터 모두 One Hot Encoding 방식을 이용한 더미변수(Get dummy)로 처리 후 모델에 적용했다. 다음 장의 오른쪽 그래프는 범주형 자료 간의 heatmap이고 왼쪽은 모든 변수에 대한 heatmap이다.

대출등급(C)와 과거 연체(Y) 간의 상관계수가 0.44로 가장 높게 나타났고 신용 등급(A)와 대출이자, 집 소유(Rent)와 집 소유(Mortage), 나이와 신용불량 기간이 상관계수가 크게 나타난 것을 확인할 수 있다. Feature Selection과 제거 후 모델을 비교해 보았을 때, 크게 유의미한 결과를 띄지 않아서 최종 모델에서는 제거하지 않고 진행했다.



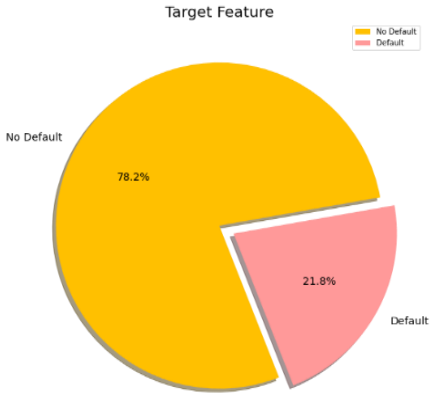


범주형 & 모든 특성변수 상관계수 Plot



iv. 목표 변수

목표 변수인 loan status는 채무 불이행의 유무에 대한 변수이고 파이 차트를 보면 대략 No default에 대한 비중은 약 78%, default는 22%로 목표 변수는 불균형한 데이터임을 확인 할 수 있다.



v. 범주형 자료

변수 selection 과정에 있어서 selectKbest를 사용하였다.
일반적으로 Score(F-score)가 5이상이면 통계학적으로 유의미하다 알려져 있기에 이 데이터에 한해서도 5 이하의 변수들은 제거하는 것이 일반적인 방법이다.

하지만 9페이지의 heatmap correlation과 일반적인 대출 관점을 미루어 보았을 때, score 5 이하의 4개의 변수들도 충분히 유의미하다 판단하여 (정량적으로는 제거하는 것이 맞지만 해석력 측면이나 정성적인 관점에서는 쓸모있는 변수라고 생각) 27개의 변수 모두 가져가는 방향으로 설정하였다.

	Spec	Score
19	loan_grade_C	2.103233
8	person_home_ownership_OTHER	2.919411
3	cb_person_cred_hist_length	2.967964
15	loan_intent_PERSONAL	3.330518
2	person_age	6.724770
23	loan_grade_G	19.143842
13	loan_intent_HOMEIMPROVEMENT	31.613307
14	loan_intent_MEDICAL	34.076647
12	loan_intent_EDUCATION	49.219726
11	loan_intent_DEBTCONSOLIDATION	62.653396
16	loan_intent_VENTURE	84.036839
22	loan_grade_F	89.815251
18	loan_grade_B	100.031772
4	loan_amnt	122.911683
1	person_emp_length	128.444049
9	person_home_ownership_OWN	161.907202
21	loan_grade_E	400.980164
24	cb_person_default_on_file_N	526.357218
25	cb_person_default_on_file_Y	526.357218
7	person_home_ownership_MORTGAGE	534.940761
17	loan_grade_A	651.543856
10	person_home_ownership_RENT	887.682714
0	person_income	1344.229330
5	loan_int_rate	1555.812042
20	loan_grade_D	1887.949932
6	loan_percent_income	2229.218963

III. Model selection with pure data

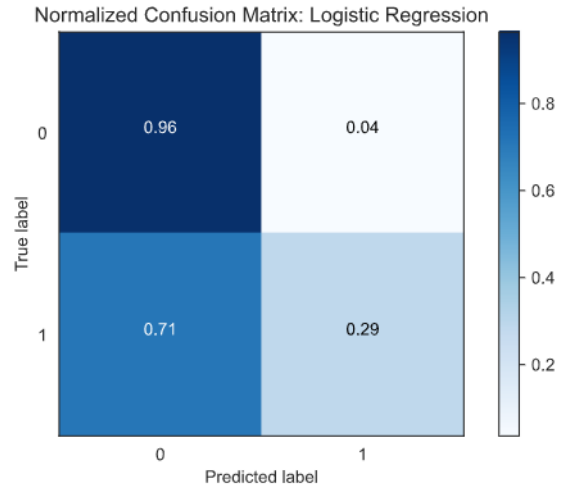
최적의 모델 선택에 있어서 타겟 변수의 특성을 먼저 살펴보았다. 지금까지 실습이나 오픈소스에서 접한 데이터들은 target variable이 imbalance data가 많았는데 이 데이터는 그다지 큰 imbalance data의 특징을 가지고 있지 않았기에 Oversampling과 Undersampling을 먼저 적용하기 보다는 Data pre-processing 과정으로 얻은 데이터로 Over 및 Under를 적용하지 않은 pure data로 첫 번째 분석을 진행했다. 이후 Oversampling과 Undersampling을 적용한 데이터에 대해 같은 방식으로 분석을 진행하였으며 한 가지 참고할 사항은 Oversampling이나 Undersampling을 적용할 시에 test자료에는 영향이 가면 안되므로 사전에 train_test_split을 이용하여 train자료와 test자료를 분리시켜 진행했다.

어느 모델이 데이터에 가장 적합할지 고려하기 위해 수업시간에 배운 모델 및 개선된 모델들을 가지고 model fitting 작업을 시행하였고, Fitting에 사용된 모델들은 Logistic Regression, DecisionClassifier, RandomForest, Adaboost, XGBoost, LightGBM 총 6개 모델을 사용했다. 추가적으로 모델 fitting에는 사용하지 않았지만 6개의 모델과 함께 범주형 데이터에 최적화 되어있는 Catboost도 함께 소개해보려 한다. Catboost에 관한 내용은 추후 언급하겠다.

각 모델들에 대해 모두 GridSearchCV를 적용하지 않고 간단한 parameter들만 셋팅 후에 fitting한 결과를 바탕으로 GridSearchCV를 진행하고자 했다. 총 6개 모델들에 대한 Confusion matirx, Accuracy, F1 score는 다음과 같다.

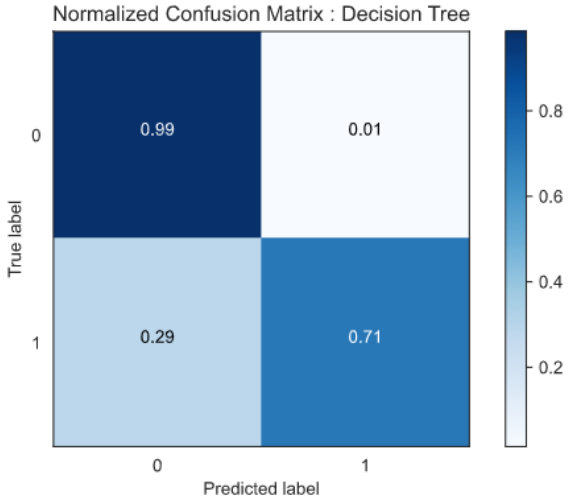
Logistic Regression

```
Accuracy Score is 0.82354
F1-Score is 0.40254
      0    1
0  6915  258
1  1339  538
[[0.96403179 0.03596821]
 [0.7133724  0.2866276 ]]
```



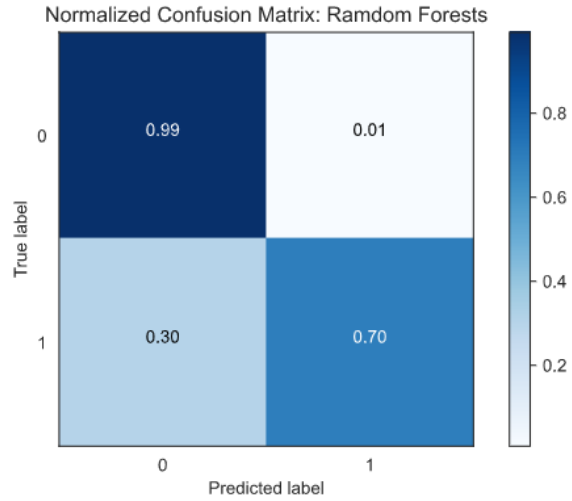
Decision Tree

```
Accuracy Score is 0.92906
F1-Score is 0.80663
      0     1
0  7069   104
1   538  1339
[[0.98550118 0.01449882]
 [0.2866276  0.7133724 ]]
```



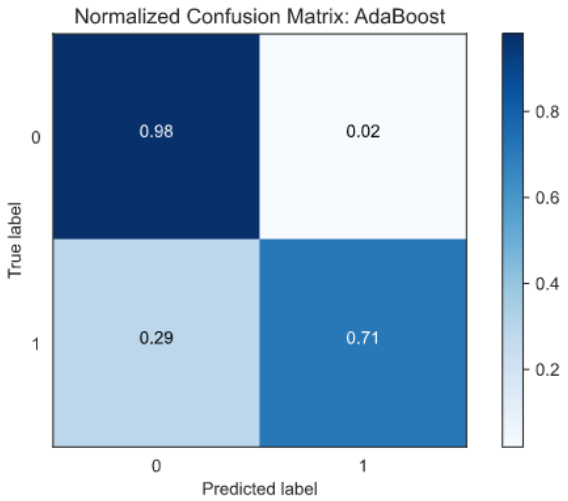
Random Forest

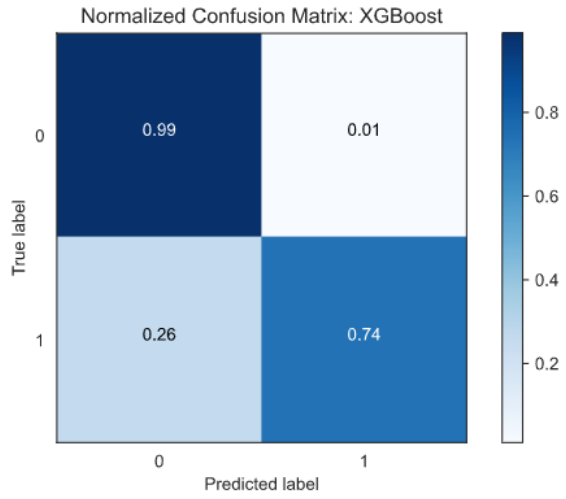
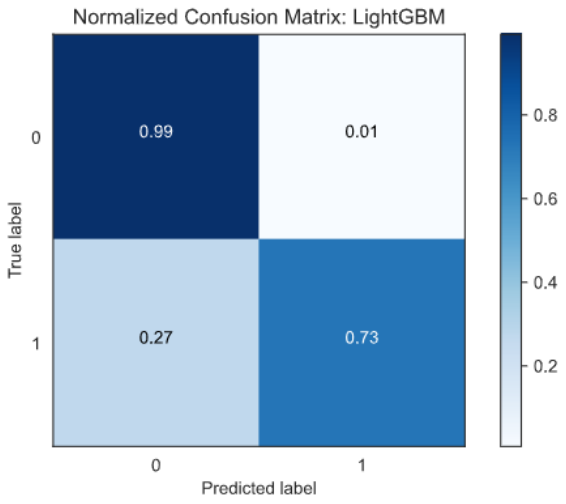
```
Accuracy Score is 0.93083
F1-Score is 0.80655
      0     1
0  7119    54
1   572  1305
[[0.99247177 0.00752823]
 [0.30474161 0.69525839]]
```



AdaBoost

```
AdaBoost Accuracy Score is 0.92486
F1-Score is 0.79738
      0     1
0  7032   141
1   539  1338
[[0.98034295 0.01965705]
 [0.28716036 0.71283964]]
```



XGBoost	LightGBM
XGboost Accuracy Score is 0.93691 F1-Score is 0.8292 0 1 0 7093 80 1 491 1386 [[0.98884707 0.01115293] [0.26158764 0.73841236]]	Accuracy Score is 0.93812 F1-Score is 0.83041 0 1 0 7119 54 1 506 1371 [[0.99247177 0.00752823] [0.26957912 0.73042088]]
	

모델 fitting결과 LightGBM의 F1 Score가 가장 우수하게 나왔으며 XGBoost도 Light GBM에 못지않게 우수한 결과를 도출한 것을 확인할 수 있다.

새롭게 소개하는 CatBoost도 두 개의 모델(XGBoost, LightGBM)에 못지않게 좋은 결과를 보여주었으나(Accuracy = 0.9329, F1 score = 0.8139) 이 보고서에는 CatBoost를 중점으로 소개하는 것이 아니기에 CatBoost에 관한 자세한 사항은 후반부에 다시 설명하도록 하겠다. 이해를 돕기 위한 CatBoost를 사용한 confusion matrix, Accuracy, F1 score를 보여주는 시각자료는 후반부에 작성하였다.

F1_score를 바탕으로 GridSearchCV를 진행하게 되는 모델은 RandomForest, XGBoost, Light GBM으로 선정하게 되었다. 모델 퍼포먼스가 가장 좋은 LightGBM 1개 만을 최종 모델로 바로 선정하는 것 보다 수업시간에 배운 XGBoost와 LightGBM이 어떤 차이점이 있고 각자의 장단점이 무엇인지 알아보는 과정과 Bagging & Decision Tree를 섞은 RandomForest도 함께 GridSearchCV를 통하여 최적의 parameter를 찾은 후, fitting한 결과를 비교해보며 교차 분석을 해보는 것이 합리적이라 생각했다. 결과는 아래와 같으며 Logistic Regression에 대한 결과는 선정한 모델들의 개선된 성과를 보여주기 위해 임의로 넣은 것이다.

	Accuracy	Roc	Recall	Precision	F1
Logistic	0.8266	0.6322	0.2999	0.6883	0.4178
RF	0.9334	0.8457	0.6958	0.9761	0.8124
XGBoost	0.9354	0.8548	0.7171	0.9614	0.8215
LGBoost	0.9362	0.8567	0.7208	0.9623	0.8242
Cat	0.9347	0.8502	0.7059	0.9714	0.8176

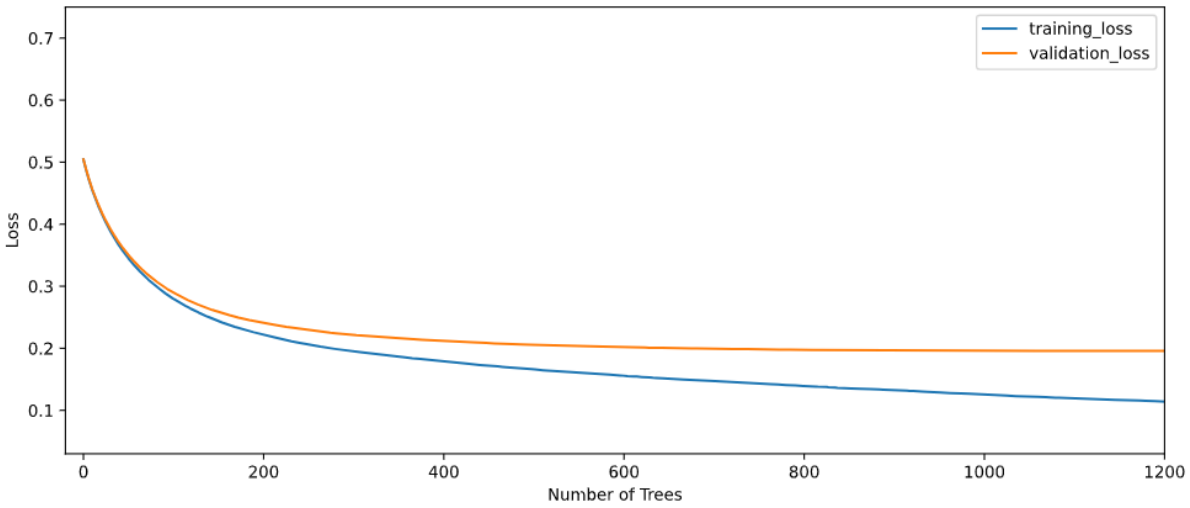
Accuracy, Roc, Recall과 Precision의 조화평균인 F1 score를 미루어 보았을 때 최종적으로 LightGBM이 가장 우수한 모델이라 판단하였고 LightGBM를 통한 최종 모델 fitting과정을 진행하였다. 결과는 아래와 같다.

	Accuracy	Roc	Recall	Precision	F1
LightGBM	0.9366	0.8585	0.7251	0.9591	0.8258

왼쪽 결과는 최종 모델인 LightGBM을 이용하여 full parameter tuning 및 iteration 횟수와 learning rate을 극대화시킨 결과이다.

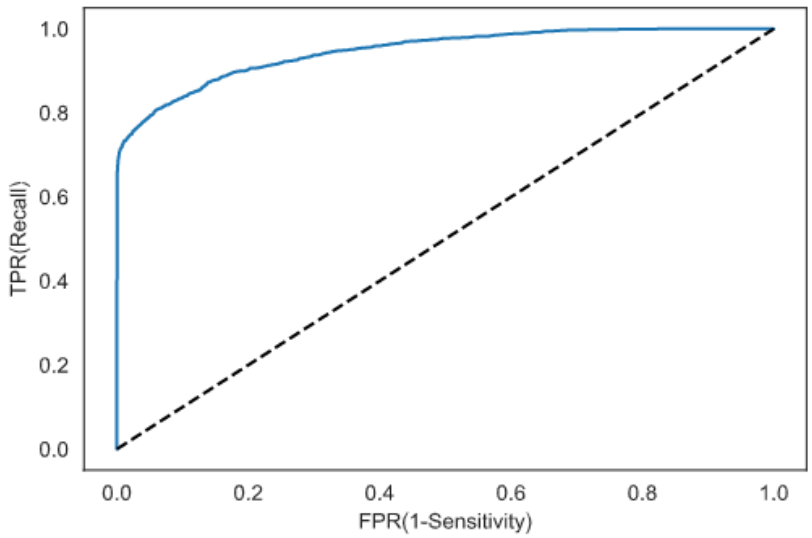
마지막으로 Learning curve를 통해 training과 validation의 관계를 알아본 결과, 이론적내용들과 일맥상통하게 흘러가는 것을 관찰할 수 있었다. Learning curve를 통한 pure data를 분석한 최종 결과는 다음 장에 이어지겠다.

IV. Results of pure data



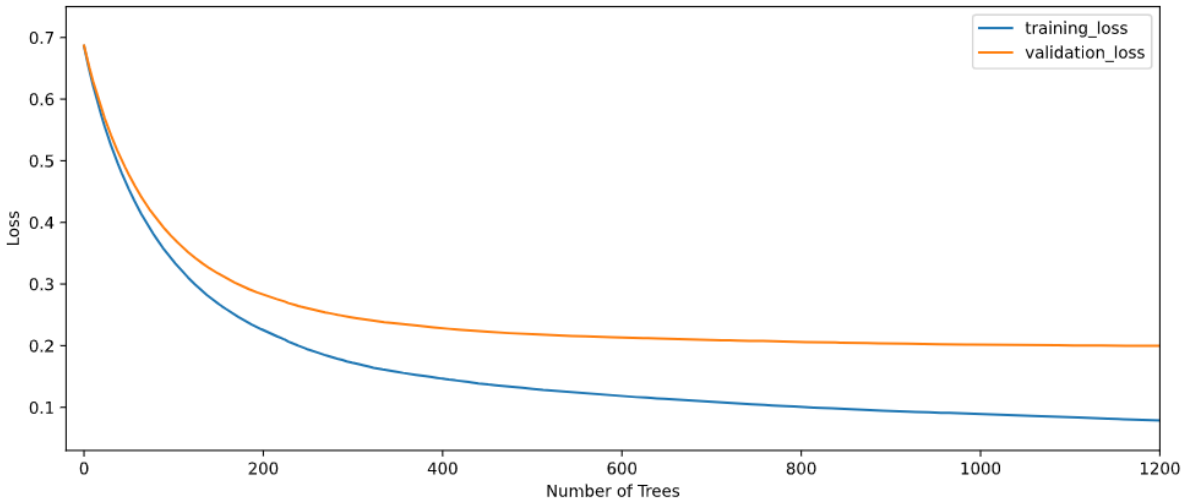
	Num of Trees 100	Num of Trees 200	Num of Trees 400	Num of Trees 600	Num of Trees 800	Num of Trees 1000	Num of Trees 1200
Train_Loss	0.279775	0.221695	0.179122	0.155688	0.139164	0.125229	0.114049
Validation_Loss	0.289783	0.240886	0.211825	0.201821	0.197538	0.196057	0.195701
Gap	0.010007	0.019192	0.032703	0.046134	0.058373	0.070828	0.081651

	Accuracy	Roc	Recall	Precision	F1
Logistic	0.8266	0.6322	0.2999	0.6883	0.4178
RF	0.9334	0.8457	0.6958	0.9761	0.8124
XGBoost	0.9354	0.8548	0.7171	0.9614	0.8215
LGBoost	0.9362	0.8567	0.7208	0.9623	0.8242
Cat	0.9347	0.8502	0.7059	0.9714	0.8176



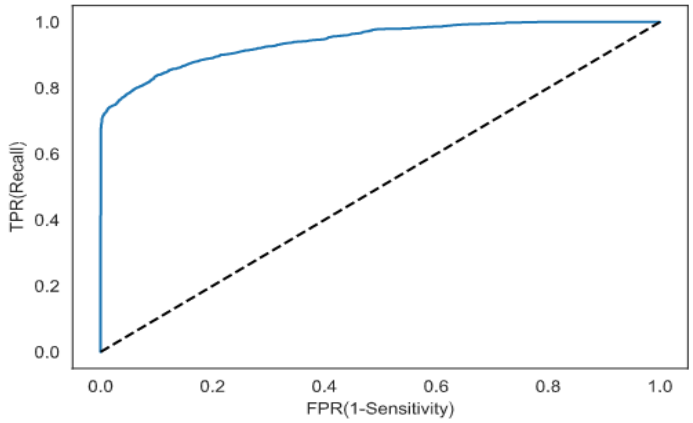
V. Analysis with over/under sampling data

OverSampling에 대한 결과

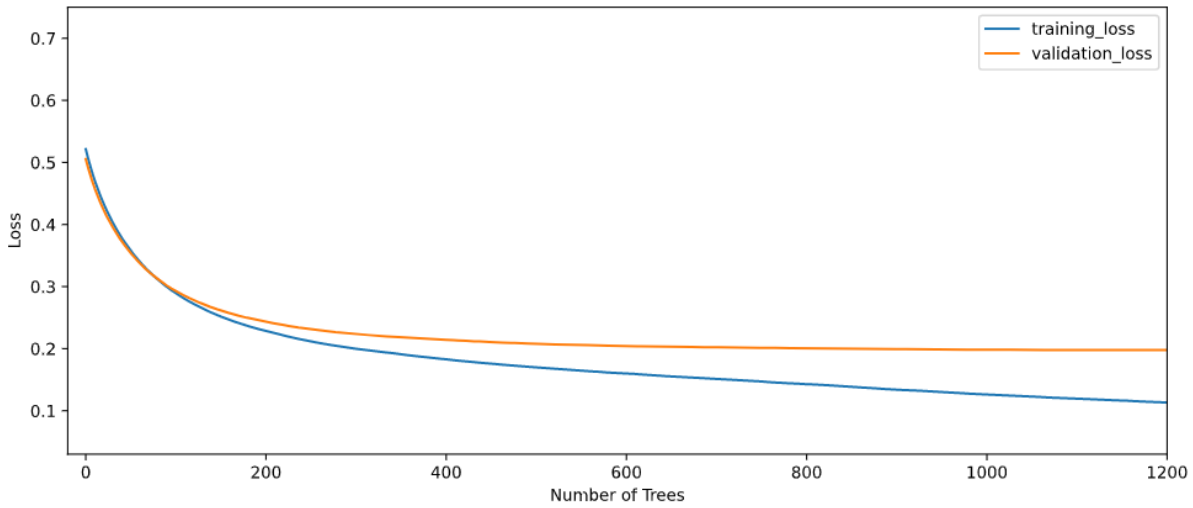


	Num of Trees 100	Num of Trees 200	Num of Trees 400	Num of Trees 600	Num of Trees 800	Num of Trees 1000	Num of Trees 1200
Train_Loss	0.3387	0.2264	0.1456	0.1175	0.0998	0.0879	0.0773
Validation_Loss	0.3752	0.2831	0.2279	0.2131	0.2069	0.2036	0.2009
Gap	0.0365	0.0567	0.0823	0.0956	0.1071	0.1157	0.1236

	Accuracy	Roc	Recall	Precision	F1
Logistic	0.7621	0.7081	0.6159	0.4467	0.5178
RF	0.9270	0.8442	0.7027	0.9276	0.7996
XGBoost	0.9349	0.8575	0.7251	0.9491	0.8221
LGBoost	0.9346	0.8596	0.7315	0.9398	0.8226
Cat	0.9337	0.8545	0.7192	0.9487	0.8182

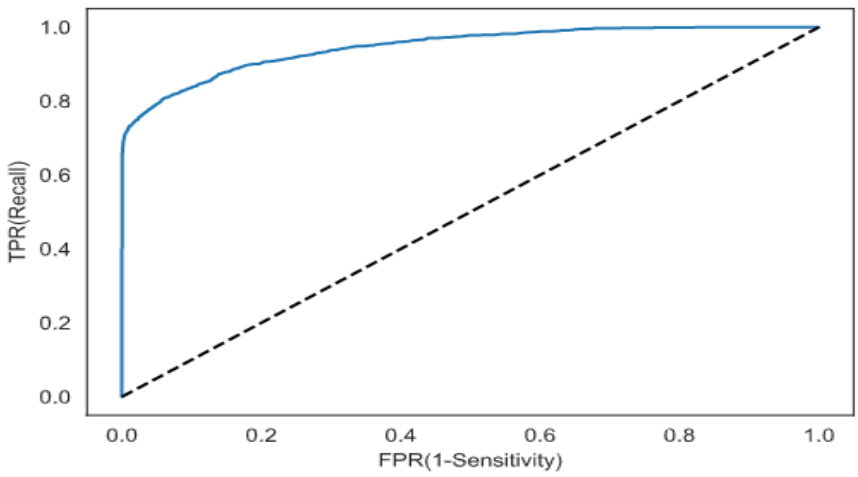


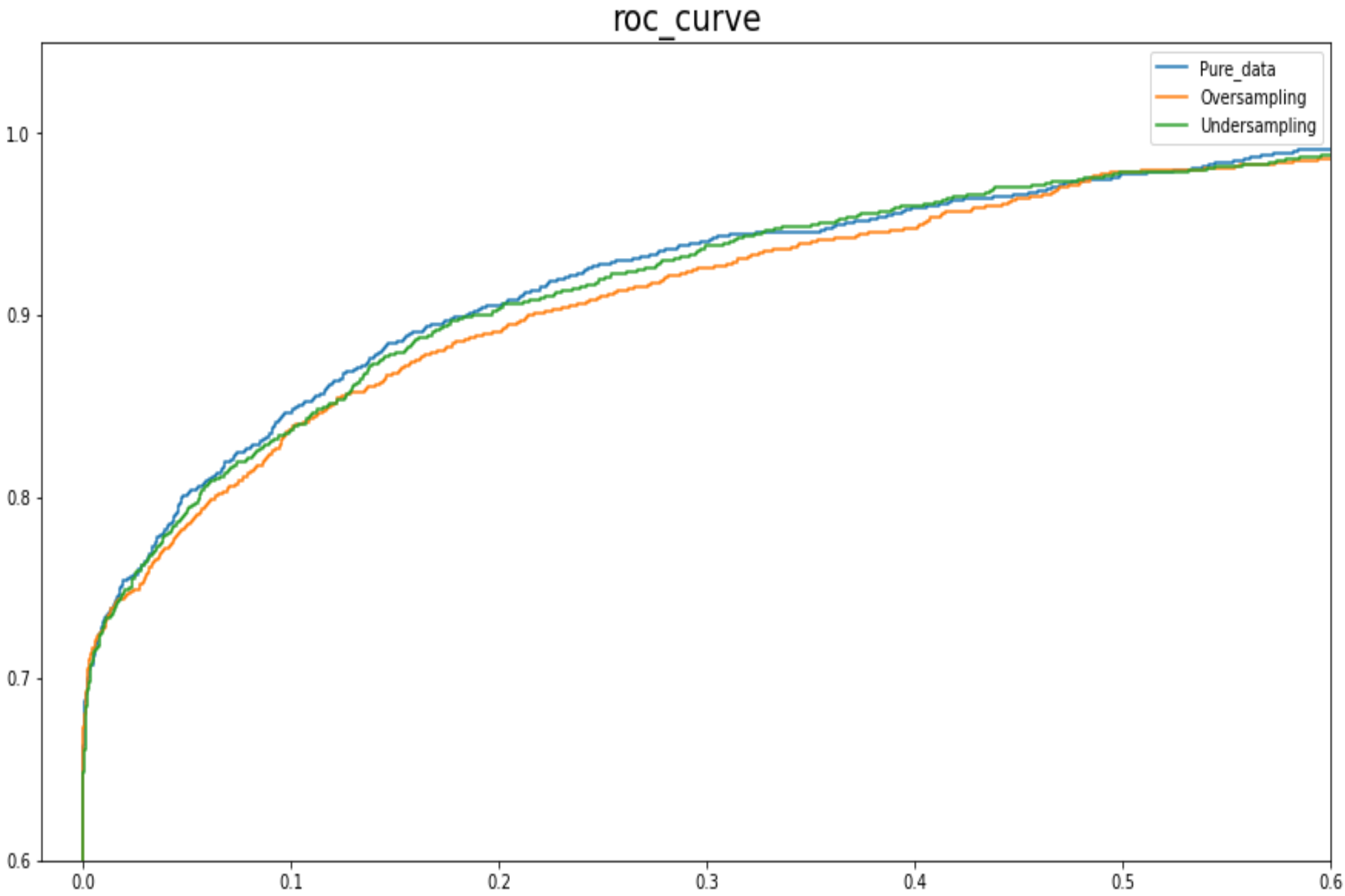
Undersampling에 대한 결과



	Num of Trees 100	Num of Trees 200	Num of Trees 400	Num of Trees 600	Num of Trees 800	Num of Trees 1000	Num of Trees 1200
Train_Loss	0.2892	0.2284	0.1826	0.1599	0.1426	0.1292	0.1163
Validation_Loss	0.2929	0.2434	0.2144	0.2048	0.2014	0.1995	0.1985
Gap	0.0037	0.0151	0.0318	0.0449	0.0588	0.0703	0.0822

	Accuracy	Roc	Recall	Precision	F1
Logistic	0.8260	0.6477	0.3431	0.6531	0.4499
RF	0.9330	0.8468	0.6995	0.9690	0.8125
XGBoost	0.9354	0.8577	0.7251	0.9517	0.8231
LGBoost	0.9351	0.8582	0.7267	0.9485	0.8229
Cat	0.9327	0.8508	0.7107	0.9529	0.8142

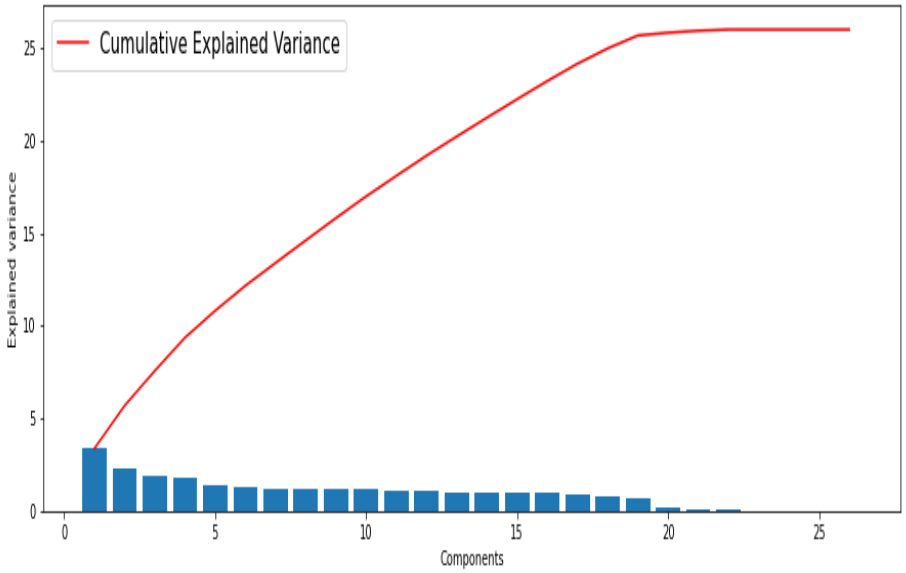



roc_curve
Pure_data
Oversampling
Undersampling
1.0
0.9
0.8
0.7
0.6
0.0
0.1
0.2
0.3
0.4
0.5
0.6

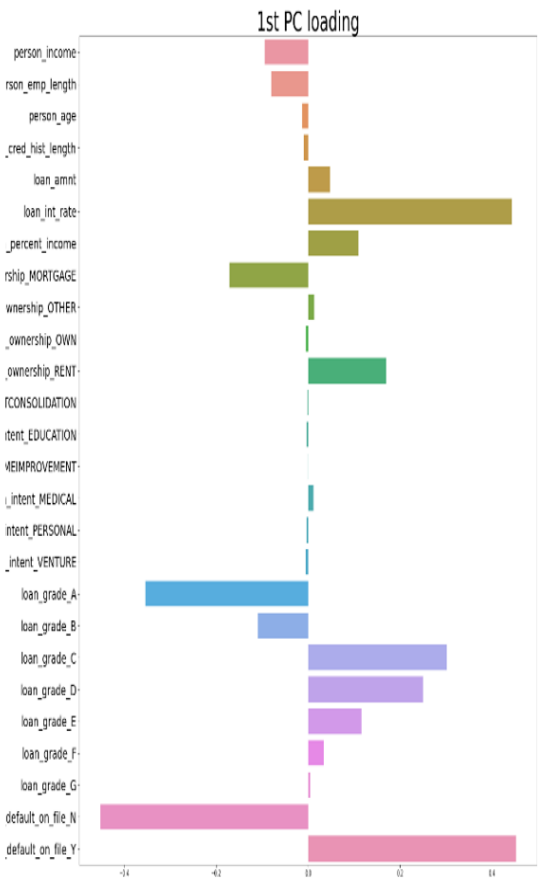
VI. PCA Analysis

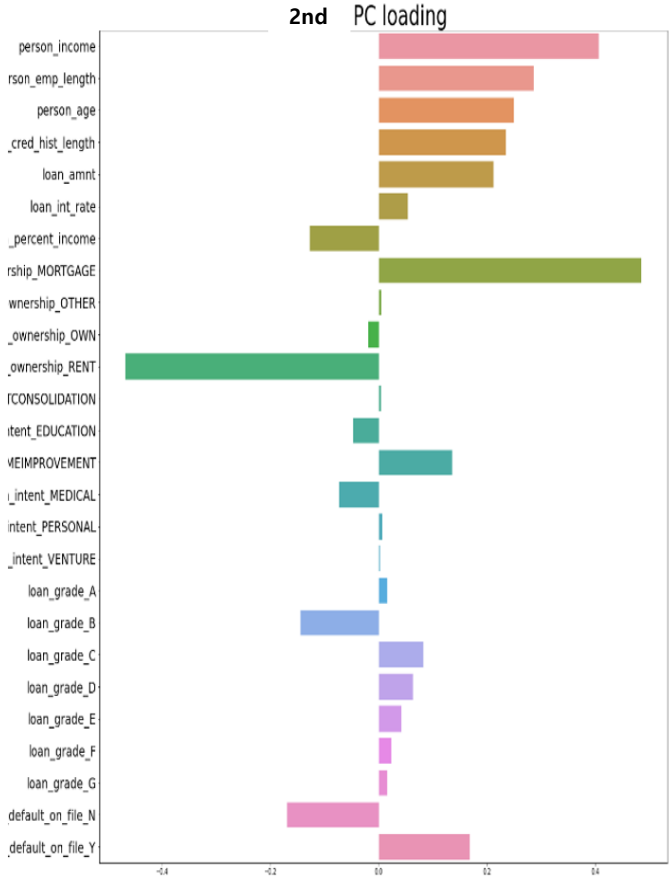
추가적으로 PCA분석을 통해 전체 변동성의 약 70%를 설명하는 고유벡터를 선택하여 위와 동일한 과정을 거쳐 모델을 생성했지만 큰 성과를 내지는 못했다. 최종 모델에 직접적으로 사용하지는 못했지만 대략적으로 각 고유벡터가 나타내는 로딩(loading)값이 생각해 볼 만한 의미를 줄 수 있다고 생각해서 분석을 진행했다. 전체적으로는 하나의 고유벡터가 전체 변동성의 대부분을 설명하지는 못했고 전체 변동성에 대해서 약 70%를 설명하기 위해서 사용해야하는 고유벡터의 개수는 10개였음을 확인했다. 원래의 특성변수가 11개인 점과 비교를 해보면 차원 축소가 가능하는 장점으로는 본 주제에는 해당하지 않았음을 알 수 있었고 이는 각 변수들의 상관계수가 높지 않기 때문에 발생한 것이라 판단된다.

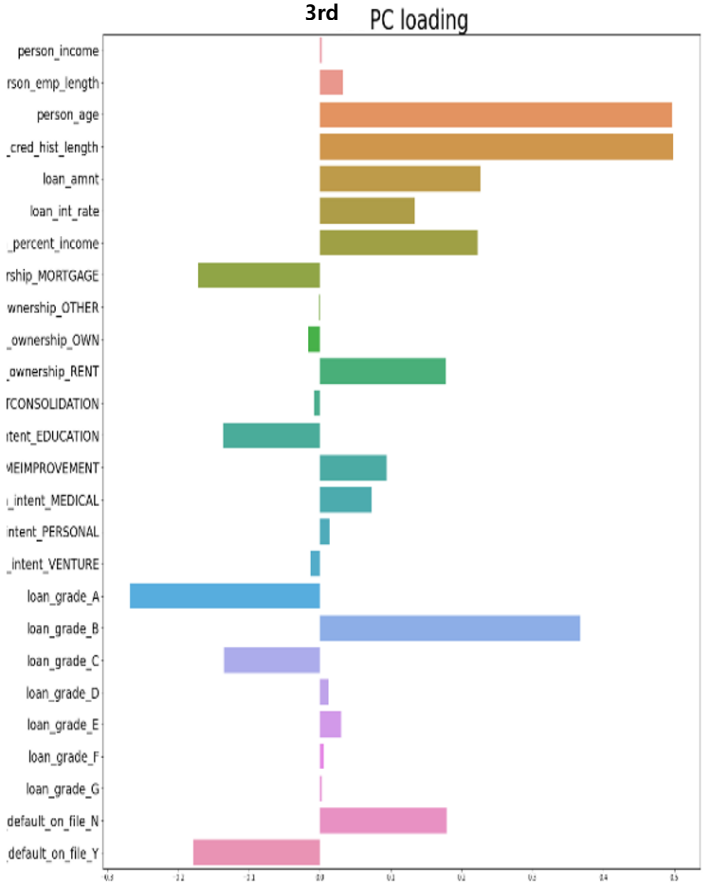
또한 PC1~PC3까지 로딩(loading)값을 보고 분석한 결과 PC1에 대해서는 대출이자, 과거 연체 유무에 대해서 가장 많은 값을 차지하고 있음을 확인하여 대출이자 또한 과거의 대출이력에 관련이 있다고 생각해 PC1에 대해서는 과거 연체에 대한 정보라고 판단했다. 두 번째로 PC2에 대해서는 차주의 연 소득, 차주의 집에 대한 정보(Mortgage, Rent)에 대한 정보의 중요도가 높게 나와서 차주의 원리금 상환능력 대한 지표라고 판단했다. 마지막으로 PC3은 차주의 나이와 과거 연체기간의 정보가 중요하다고 판단했는데 PC3부터는 지표에 대한 설명이 명확하지 않다고 생각했고 PCA에 대한 분석을 마쳤다.



PCA Loading Plot







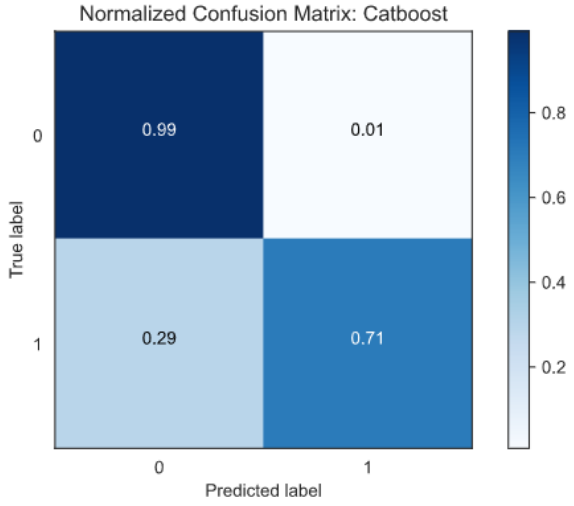
VII. Catboost

Catboost는 2017년 Yandex에서 공개한 부스팅 계열의 알고리즘이며 Category + Boosting의 합성어로 그래디언 부스팅을 기반으로 하는 알고리즘이다. 이름에서도 알 수 있듯이 카테고리 변수들의 전처리에 특정 방식을 적용하여 예측력을 높이는 방법을 적용하고 있으며 일반적으로는 수치형 자료보다는 범주형 자료에 더 좋은 예측력을 보여주는 것으로 알려져 있다.

Catboost의 특징으로는 XGBoost와 동일하게 Level-wise로 트리를 만들어 나간다는 것이다. 딥러닝과 다르게 적은 데이터로도 좋은 결과를 얻을 수 있는 효율적인 알고리즘이며 범주형(카테고리)형 변수를 사용자가 작업을 하지 않아도 자동으로 변환하여 사용한다. 또한 하이퍼 파라미터 튜닝을 하지 않더라도 기본 세팅만으로도 충분히 훌륭한 성능을 가진다.

보고서 도입부에 언급했듯이 Catboost가 분석의 main model은 아니지만 우리 데이터에서는 30% 수치형 자료, 70% 범주형 자료로 이루어져 있었기에 범주형 자료에 널리 쓰이는 Catboost도 적용시켜 보았다.

```
CatBoost Accuracy Score is 0.93293
F1-Score is 0.81397
      0     1
0  7115    58
1   549  1328
[[0.99191412 0.00808588]
 [0.29248801 0.70751199]]
```



자세한 하이퍼파라미터 튜닝 과정을 거치지 않고 기본 셋팅만으로 모델을 돌렸을 때의 Catboost 결과는 왼쪽과 같다.

비교를 위해 Model selection 과정에서 보여 주었던 confusion matrix를 catboost도 적용시켜 보았는데, LightGBM이 가장 좋은 F1 score를 보여주지만 catboost도 그에 못지 않은 좋은 성능을 보여주는 것이 확인되었다.

성능이 비슷한 이유를 모델이 가지는 특징을 토대로 분석한 결과 Catboost는 XGBoost와 동일하게 Level-wise로 트리를 만들어 나간다는 것이다.

Level-wise(균형분할)와 LightGBM에서 사용하는 Leaf-wise(잎 중심 분할)을 간략히 설명하자면, Level-wise는 트리 깊이를 최대한 줄이기 위해 초점이 맞추어져 있다면 Leaf-wise는 각 node마다 최대손실값을 갖는 node를 계속 분할하여 손실값을 줄이는데에 초점이 맞추어져 있다. 이 때문에 Leaf-wise는 비대칭적으로 어느 트리는 깊이가 아주 깊어지게 되고 이 방식은 균형 트리 분할보다 오류를 줄일 수 있으며 메모리 사용과 러닝시간을 줄이는데 큰 도움이 된다.

하지만 Catboost에도 한계점은 존재한다. 우선 Sparse Matrix (희소행렬, 행렬의 원소에 비교적 0이 많은 행렬)은 처리하지 못한다는 단점이 있다. 또한 데이터 대부분이 수치형 변수인 경우 LightGBM보다 학습 속도가 느리다는 것이다. 따라서 데이터의 변수가 100% 범주형 자료로 이루어져 있거나 수치형 자료를 여러 level로 나눌 수 있는 자료의 경우에 한해서 사용하는 것이 좋다.

VIII. Conclusion

이번 프로젝트의 목적은 은행에서 대출 상환 능력이 있다고 판단되어 대출 승인이 난 후의 데이터를 가지고 차주의 채무 불이행을 예측하는 것을 목표를 하고 있다. 즉. 차주의 신용 리스크 예측, 채무 불이행을 예측하는데 초점을 맞추고 있다. 은행 입장에서는 은행 내부의 대출 승인 조건에 대해 평가할 수 있을 뿐만 아니라 은행의 리스크 관리에 중요한 부분이다. 은행의 대출 승인을 받은 후의 예측이기에 데이터의 불균형은 어느정도 예상되는 부분이었고 그로 인해 분석에 오버 혹은 언더 샘플링의 필요성과 중요성에 대한 분석의 필요성을 느꼈다. 또한 이러한 방법론을 사용한 모델과 사용하지 않은 모델과 비교했을 시 어느정도 성과가 개선되는지 혹은 위의 방법론이 실제 데이터 분석에 어느 정도의 중요성을 가지고 있는지 평가 및 비교하는데 초점을 두고 프로젝트를 진행했다.

결과적으로는 오버 및 언더 샘플링을 사용한 모델보다 사용하지 않은 모델의 성과가 좀 더 좋은 성과를 나타냈습니다. 다만, 방법론의 사용 유무에 대한 성과 차이는 유의하다고 말할 수 준으로 차이가 나는 것은 아니였다. 때문에 이번 프로젝트에 사용한 데이터와 모델에 대해서는 방법론의 차이에 대한 뚜렸한 차이는 발견하지 못했다.

이를 분석했을 때, 본래 가지고 있던 원 데이터의 정보가 분석 목표에 대한 정보를 충분히 가지고 있고 이 정보들이 차주의 채무 불이행을 예측하는데 핵심적인 정보를 포함하고 있다고 판단했다. 이러한 결과로 목표 변수의 데이터가 적은 불균형을 가지거나 혹은 큰 불균형을 가지더라도 원 데이터의 정보가 충분히 유의하고 핵심적인 내용은 내포하고 있다면 충분히 유의한 성과를 낼 수 있다는 것을 확인할 수 있는 결론을 내리고 확인할 수 있는 계기가 될 수 있다.

또한 모델이 복잡해질수록 성과는 높아졌지만 해석에 대한 부분에 한계점을 확인할 수 있었다. 머신러닝, 딥러닝 사용으로 인한 모델의 해석력이 감소함에 따라 결과에 대한 명확한 분석을 할 수 없기에 실제 사용하기에 많은 어려움이 있다고 판단했다. 복잡한 모델을 통해 대출 승인을 판단한다면 승인 유무에 대해서 알 수 없는 차별이 발생할 가능성 때문에 아직까지는 현실적으로 사용하기에는 어려 문제가 발생할 것이라 생각했고 때문에 위의 주제를 현실적으로 적용하고 큰 의미를 가지려면 좋은 예측력 뿐만 아니라 강한 해석력을 가지는 모델의 필요성을 느꼈다.