Lending Club Data
Default Prediction

함께하조 (1조)

김민환 김소희 장수빈 전민규 정명훈

Contents

배경 소개

Lending Club 소개

Lending Club은 초기에 P2P 대출로 부상했으나 상장 후 2016년에 위기를 맞았다.
그러나 은행 charter를 획득한 후 **현재는 대출 플랫폼보다는 은행의 형태**로 운영되고 있다.

Lending Club

2007년

- P2P 대출 중개 플랫폼으로 출범
- 채무자의 신용 등급, 대출 금액 등을 게시하고 채권자가 원하는 만큼 fund 하는 방식

2016년

- 투자자 유치 어려움에서 비롯된 연이은 대출 이자율 상승으로 주가 하락
- 정보 공개 관련 논란으로 CEO 해임

2020년

- Radius Bank 인수 통해 예금 등 상품 발행 능력 확보, 수익률 향상 도모
- 현재는 P2P 대출 발행 중단

P2P 대출의 수입 구조

Lending Club이 운영하는 Personal Loan의 주 수입원은

1) 채무자에게서 대출을 발행한 것에 대해 받는 수수료와 2) 채권자에게서 받는 서비스 수수료이다.

Loan origination fee (대출 발행 수수료)
Loan amount의 3~6%

채무자

Lending Club

Service fee (서비스 이용 수수료)
채무자에게 받는 월 상환금의 약 1%

채권자

이때 채무자는 중금리 대출이 가능하고, 채권자는 은행에 비해 높은 기대수익률을 얻을 수 있어 시스템이 형성된다.

전처리 및 EDA

Dataset 소개

Dataset을 살펴본 결과 다음과 같은 사실을 확인할 수 있었다.

특히 **불균형 데이터**라는 점과 결측치가 없어 보이나 실제로는 일부 존재한다는 점이 특징적이다.

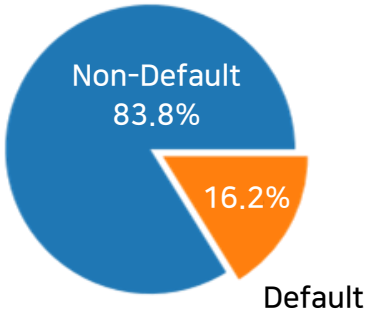
```
# 데이터 크기 확인
df.shape
```

```
(874335, 334)
```

행이 874335개, 열이 334개인 데이터이다.

```
# 종속변수의 Value 확인
df['depvar'].value_counts()
```

```
0.0    732652
1.0    141683
Name: depvar, dtype: int64
```



종속변수 depvar의 값이 약 8:2로 나타나는 불균형 데이터이다.

(Default: 1, Fully Paid: 0)

→ 비율을 유지한 채로 train set과 test set을 추출함

```
# Null 값의 비율
(df.isnull().sum().sum())/(df.shape[0]*df.shape[1])
```

```
0.0
```

```
'emp_length8': 'emp_length==7 years',
'emp_length9': 'emp_length==8 years',
'emp_length10': 'emp_length==9 years',
'emp_length11': 'emp_length==< 1 year',
'emp_length12': 'emp_length==n/a'
```

표면적으로는 결측치가 존재하지 않지만, 열의 맥락을 살펴보면 emp_length의 경우 n/a 값을 포함하는 열이 확인된다.

사용 변수 및 제거 변수

모델링에 사용할 독립 변수는 다음과 같은 과정을 통해 선별하였다.

1. Data 정의서를 확인하여 각 feature 가 무엇을 의미하는지 파악

2. trivial 한 변수를 제거하고, 유사한 의미를 갖는 변수의 경우 하나만 선택

ex) index, mths_since_recent_inq: trivial하므로 제거
revol_bal: revol_bal보다 revol_util을 통해 신용 상태를 평가하는 것이 더욱 적절하므로 revol_bal 제거
total_acc: open_acc이 total_acc에 포함되는 개념이며 더욱 구체적이므로 total_acc 제거

3. 시점에 따라 변수들을 분류하고 모든 사후 변수를 제거

ex) funded_amnt, installment: 대출 승인이 확정된 시점에 알 수 있는 변수이므로 제거
int_rate: 이미 한 차례 신용 평가가 된 상태에서 산정된 이자율일 것이므로 모델링 과정에서 제거
out_prncp, total_pymnt: 대출 승인 후 시간이 지난 뒤, 데이터 수집 시에 도출하는 변수이므로 제거

최종적으로 사용한 21개의 변수 목록:

loan_amnt	inq_last_6mths	last_fico_range_high	pub_rec_bankruptcies
annual_inc	open_acc	last_fico_range_low	tax_liens
dti	pub_rec	acc_now_delinq	purpose
delinq_2yrs	revol_util	chargeoff_within_12_mths	emp_length
fico_range_low		delinq_amnt	home_ownership
fico_range_high			verification_status

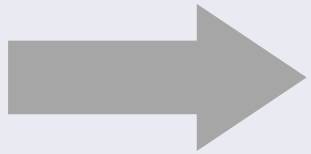
Categorical 변수 처리

EDA에 앞서, categorical 변수를 dummy 변수로 바꾼 다수의 행을 발견하여 이를 다시 categorical 변수로 변환하였다.

ex)

purpose1	purpose2	purpose3	purpose4	purpose5	purpose6	purpose7	purpose8
0	0	1	0	0	0	0	0
0	0	1	0	0	0	0	0
0	0	1	0	0	0	0	0
0	0	1	0	0	0	0	0
0	0	1	0	0	0	0	0
0	1	0	0	0	0	0	0
0	1	0	0	0	0	0	0
0	0	0	0	0	0	0	1
0	1	0	0	0	0	0	0
0	0	1	0	0	0	0	0

...



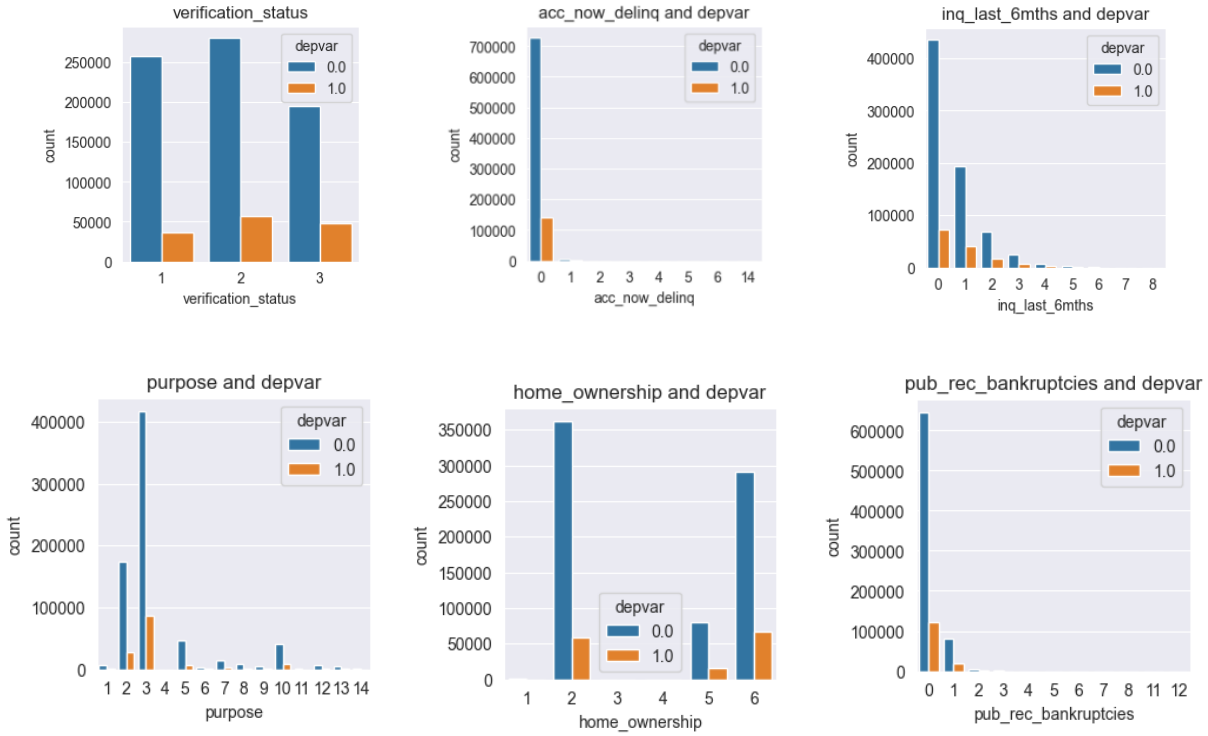
purpose
3
3
3
3
3
2
2
8
2
3

purpose 외에도 emp_length, home_ownership, verification_status 변수들에 동일한 작업을 거쳤다.

Exploratory Data Analysis

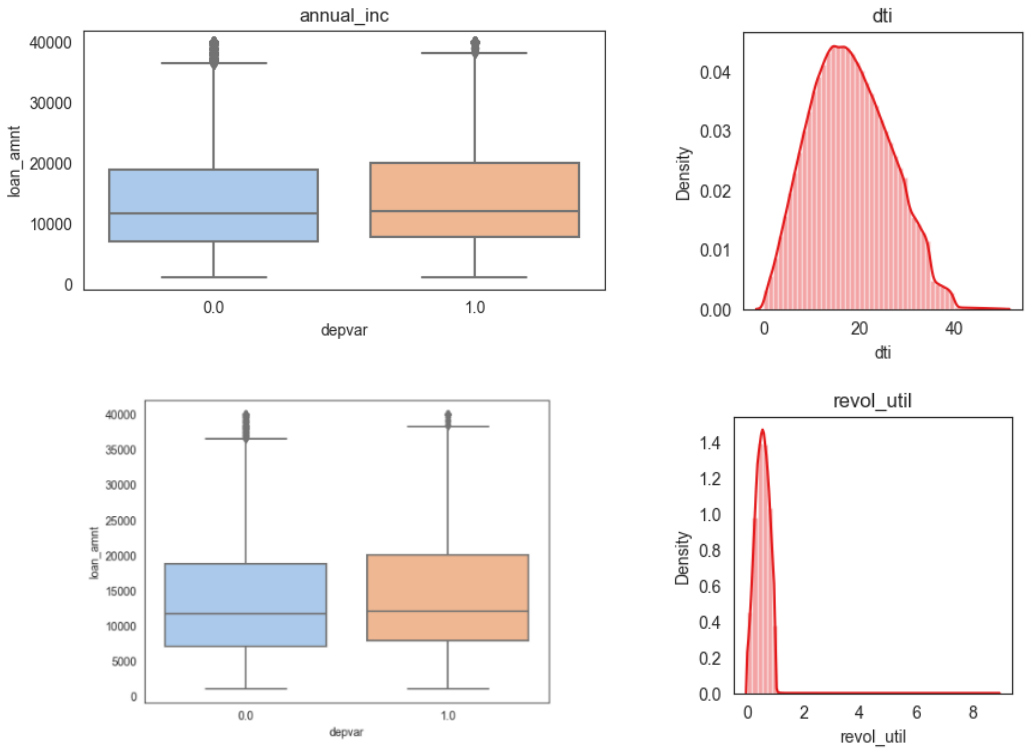
선별한 독립변수에 대해 **depvar 값에 따른 분포**를 확인하였다.

EDA의 목적은 1차적으로 독립변수 별 Default 양상을 확인하여 대략적인 관계를 파악하고자 한 것에 있다.



범주형 변수

(Blue: Fully Paid, Orange: Default)

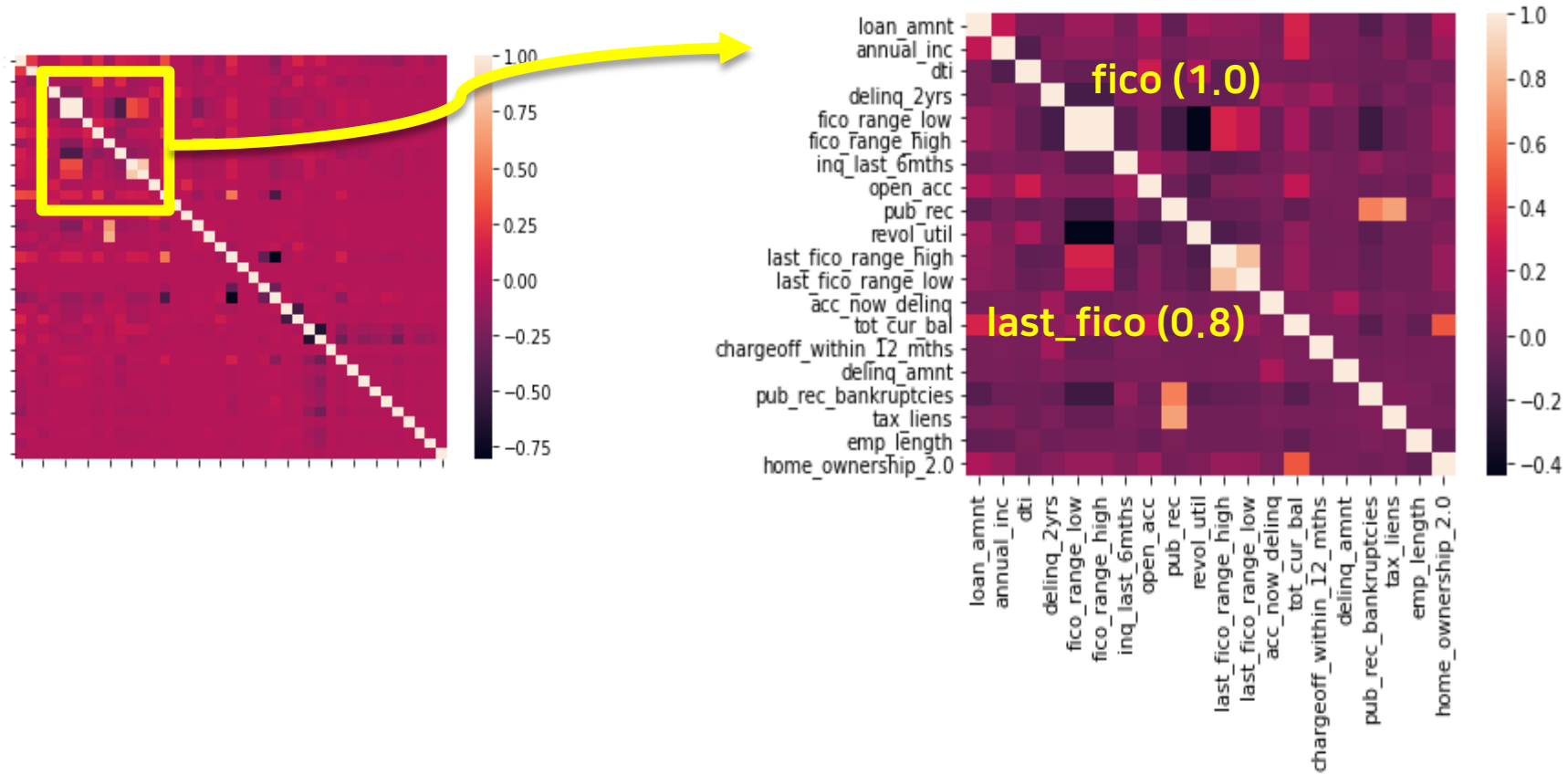


연속형 변수

파생변수 생성: fico, last_fico

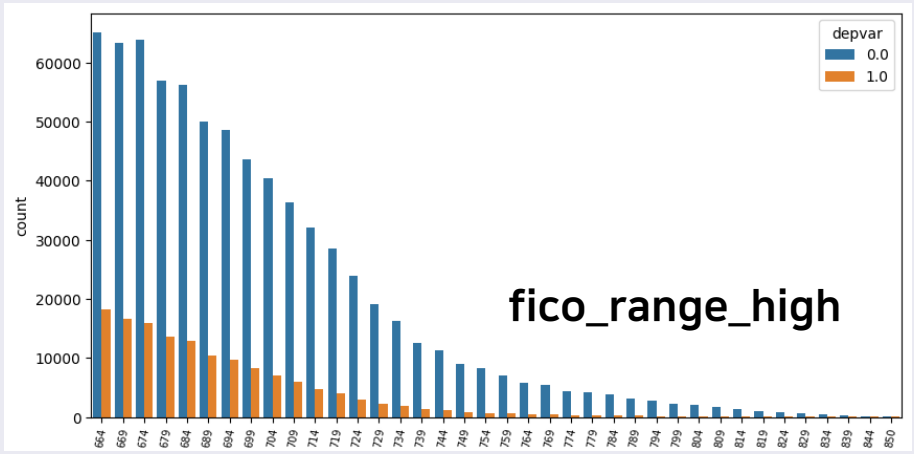
변수 간 상관관계를 확인하기 위해 **correlation heatmap**을 그려보았다.

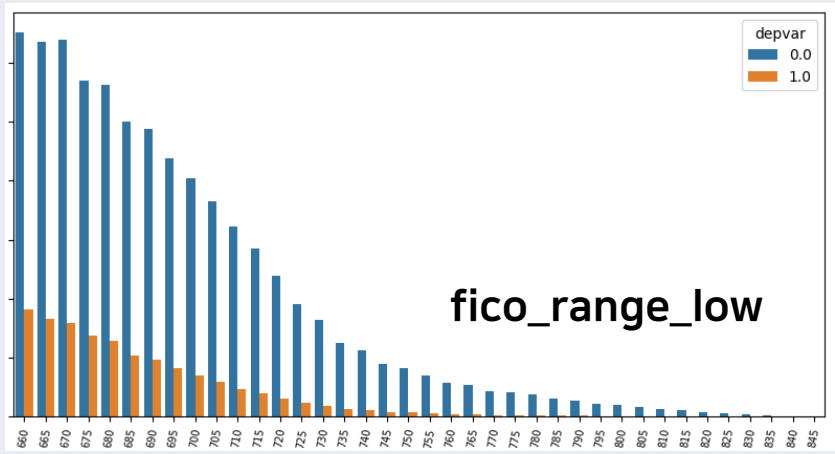
fico_range의 상하한값과 last_fico_range의 상하한값 사이의 상관계수가 매우 높아서 조정이 필요함을 확인하였다.

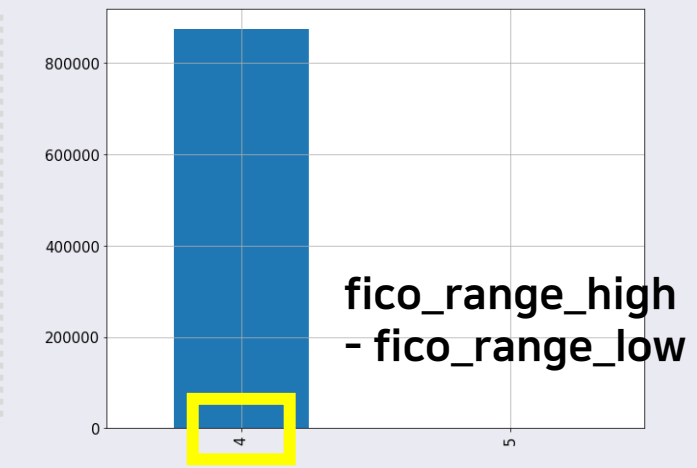


파생변수 생성: fico_average

fico_range_high와 low의 분포를 확인해본 결과, 둘은 값만 다를 뿐 모양이 거의 동일하여 상관계수기 1로 나타났으며
이때 fico_range_high와 low의 **차이값은 4가 지배적임**을 알 수 있었다.
따라서 중복을 방지하되 기존 정보는 손실 없이 이용하기 위해 **fico_average 라는 새로운 열을 생성**한 뒤 기존 열을 삭제하였다.








```
fico_average = (fico_range_high + fico_range_low) / 2
```

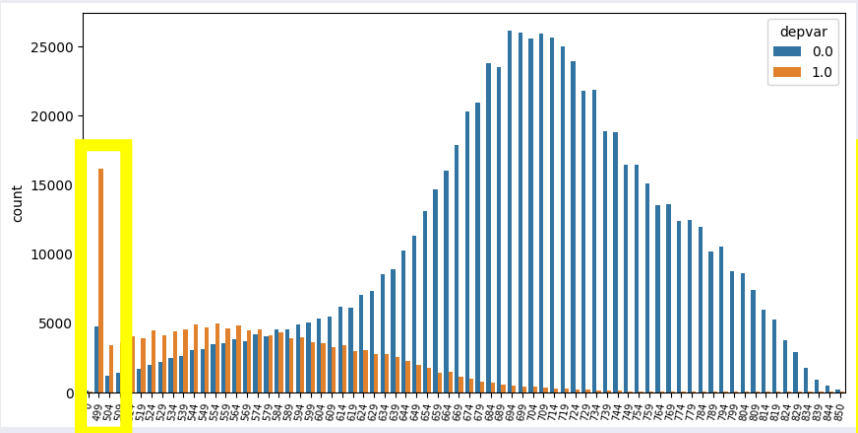
추가 변수 생성: last_fico_range_big

last_fico_range도 0.8의 높은 correlation을 보였고, last_fico_range_**high – low = 499** 인 특징적인 데이터를 파악하였다.

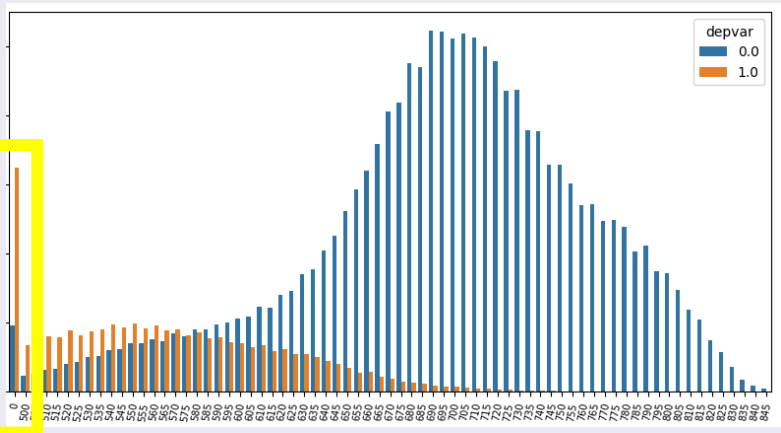
해당 경우의 depvar 값을 확인해보니 default가 약 77%, fully paid가 약 23%로 나타났고,
이에 따라 high– low = 499인 데이터는 **일종의 고위험군**으로 해석 가능하다고 판단하였다.

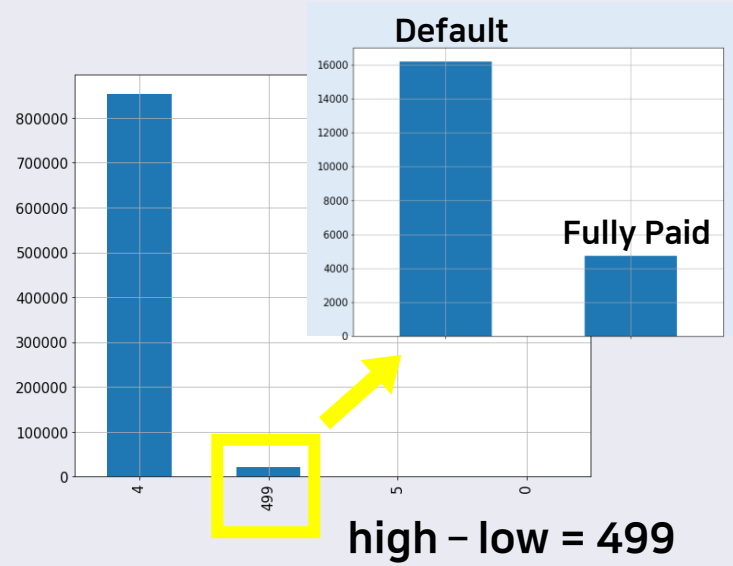
따라서 새로운 열 **last_fico_gap_big**을 만들어 차이값이 499인 경우에는 1, 그렇지 않은 경우 0을 할당하여 처리하였다.

last_fico_range_high



last_fico_range_low

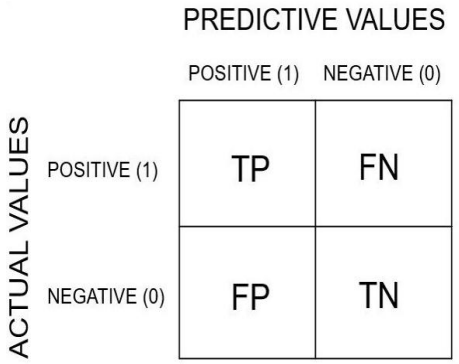


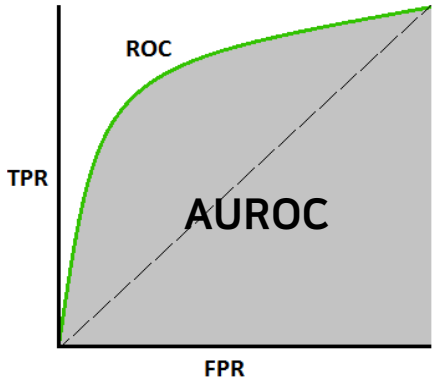


모델링

성능 평가 지표: AUROC

모델이 Default를 얼마나 잘 예측하는지 파악하기 위해 **AUROC score**를 성능 평가 지표로 사용하였다.
FPR이 낮은 상태에서 큰 TPR을 얻을수록 score가 1에 가까워지고 모델 성능이 좋아진다는 것에 초점을 맞췄으며,
이러한 AUROC score를 이용해 **optimal threshold**를 찾아 비즈니스 목적에 맞는 모델을 만들고자 하였다.





TPR (True Positive Rate) = Recall

실제 Default 중 모델이 Default로 예측한 비율 $\frac{TP}{FN + TP}$

FPR (False Positive Rate)

실제 Fully Paid 중 모델이 Default로 잘못 예측한 비율 $\frac{FP}{TN + FP}$

대출을 상환할 사람을 Default로 잘못 예측한 경우를 줄이고 (FPR ↓)
채무불이행자는 정확하게 Default 할 것이라고 예측하여 (TPR ↑)
AUROC score가 1에 가까운 모델을 얻고자 함

목적 함수 정의

발표 자료에는 Lending Club의 이익을 극대화하는 것을 목적으로 아래와 같은 목적함수를 사용하였다.
초기의 목적함수는 Lending Club의 이익을 총체적으로 파악하는 과정에서 기회 비용을 비롯한 모든 이익과 손실을 고려했다는 점에서 의의가 있으나, 발표 피드백 후 **더 간결하고 효과적인 목적함수**를 고민하게 되었다.

$$f(x) = \mathrm{L}ending\ Club\ profit\ -\ Lending\ Club\ loss$$

$$profit = TN * avg(loan\ amnt) * (ir + cm) + TP * avg(loan\ amnt) * avgpayoff * (ir + cm)$$

$$loss = FP * avg(loan\ amnt) * (ir + cm) + FN * avg(loan\ amnt) * (1 - avgpayoff) * (ir + cm)$$

ir = 채무자 이자율 평균
cm = 채권자 수수료율 평균
$avgpayoff$ = Default 발생 시 채무자가 빌린 돈 중 갚은 돈의 비율의 평균

목적 함수 정의

목적함수 재설정을 위해 Lending Club의 운영 구조를 거시적으로 이해하고자 했고,
그 결과 Lending Club의 이익 극대화보다는 **투자자들의 이익 극대화를 목적**으로 하는 것으로 목적함수를 수정하였다.
결정 근거 2가지는 다음과 같다.

1. Lending Club은 이미 구조적으로 최대한의 수입을 확보하는 방향으로 운영되고 있다.
아래의 운영 방식과 APR(annual percentage rate) 산정 방식 등을 보면 Lending Club은 이미 수입을 최대한 확보하고 있다.
따라서 Lending Club의 이익을 극대화하는 것의 효과는 크지 않을 것임을 알 수 있다.

How do you pay the origination fee?

In most cases, the origination fee is included as part of your total loan amount. That means if you take out a loan for $10,000 with a 5% origination fee, you'll receive $9,500 but repay $10,000, not including interest.

대출에 대해 선불 origination fee를 받음으로써
채무 이행 여부와 관계 없이 일정 금액을 수입으로 확보함

When borrowers miss payments and loans become late, LendingClub makes a reasonable effort to recover the money owed to investors. If LendingClub is successful in recovering the owed payment(s), LendingClub charges investors one of the following collection fees, deducted from any amount recovered:

Up to 40% on all amounts collected on a delinquent loan (net of legal fees and expenses) to the extent any litigation has been initiated against the borrower, or; (이하 생략)

Default 시 채무자로부터 각종 fee와 penalty를 징수하며,
이러한 금액에 대해서도 일정 부분 수수료를 받음

목적 함수 정의

2. Lending Club은 P2P 대출이라는 상품보다 이를 통한 Scaling을 목적으로 한다.

아래에서 Lending Club이 발행한 전체 대출 중 개인 투자자들이 차지하는 비중이 매우 낮다는 것과, P2P 대출 중단 후 투자자 한정 고금리 예금 상품을 출시했다는 것을 알 수 있다. 이러한 내용을 통해 Lending Club에게는 P2P 대출 플랫폼으로 성공하는 것보다, 이를 수단으로 **투자자를 유치하고 사업 규모의 확장을 이루는 것**이 더욱 중요한 목표였음을 알 수 있다.

[Lending Club 2019 재무제표]

	December 31, 2019	September 30, 2019	June 30, 2019	March 31, 2019	December 31, 2018
Loan Originations by Investor Type:					
Banks	32 %	38 %	45 %	49 %	41 %
Other institutional investors	25 %	20 %	21 %	18 %	19 %
LendingClub inventory	23 %	23 %	13 %	10 %	18 %
Managed accounts	17 %	15 %	16 %	17 %	16 %
Self-directed retail investors	3 %	4 %	5 %	6 %	6 %
Total	100 %	100 %	100 %	100 %	100 %

[Personal Loan 중단 관련 기사]

The company specifically mentioned a high-yield savings account offering and plans to offer a version (known as the Founder Savings account) exclusively to investors on the platform. And in an email to its current retail investors, the company said that

(이하 생략)

https://www.nasdaq.com/articles/lendingclub-is-ending-its-p2p-lending-platform-now-what-2020-10-08

(이하 생략)

목적 함수 정의

즉, Lending Club에게는 장기적으로 보았을 때 투자자들을 끌어들여 규모를 확대하고 경쟁력을 확보하는 것이 관건임을 알 수 있다.
이에 따라 **투자자들의 이익 극대화에 초점**을 두고 구상한 목적함수는 다음과 같다.

```python
def optimal_threshold(cal_df):

    def cal_tn(tpi, fai):
        return_inv = tpi- fai
        return return_inv

    def cal_fn(fa, ir, tpi):
        loss_inv = fa*(ir+1) - tpi
        return loss_inv

    tn_df = cal_df.loc[(cal_df["depvar"]==0)&(cal_df["y_pred"]==0)]
    fn_df = cal_df.loc[(cal_df["depvar"]==1)&(cal_df["y_pred"]==0)]

    tn_df_ = tn_df.copy()
    fn_df_ = fn_df.copy()
    tn_df_["return"] = tn_df_.apply(lambda x: cal_tn(x["total_pymnt_inv"], x["funded_amnt_inv"]), axis=1)
    fn_df_["loss"] = fn_df_.apply(lambda x: cal_fn(x["funded_amnt"], x["int_rate"], x["total_pymnt_inv"]), axis=1)
    inv_profit = tn_df_["return"].sum() - fn_df_["loss"].sum()

    return inv_profit
```

$$f(x) = investor\ profit\ - investor\ loss$$

$$investor\ profit = TN\ \times\ (total\ pymnt\ inv\ - funded\ amnt\ inv)$$

$$investor\ loss = FN \times (funded\ amnt\ *(1 + interest\ rate))$$

Linear Models

Logistic Regression의 경우 **Lasso, Ridge, Elastic Net** 정규화를 사용하였다.

Note 1.

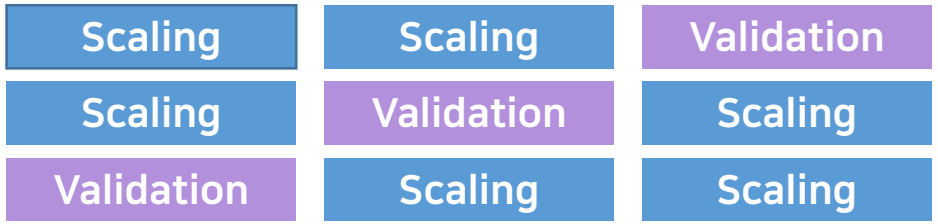
다중 공선성 문제를 고려하여 카테고리형 변수는 one-hot-encoding을 진행한 후 변수 간 독립성을 위해 더미로 변환한 칼럼 중 하나는 삭제하였다.

ex)
purpose를 purpose_1 ~ purpose_14로 인코딩 후 purpose_1은 drop, 나머지만 분석

Note 2.

Scaler의 경우 MinMax, Robust, StandardScaler를 사용하고 test set에 scaling이 적용되는 것을 방지하기 위해 cv가 split된 train set에서만 scaling이 되도록 학습시켰다.

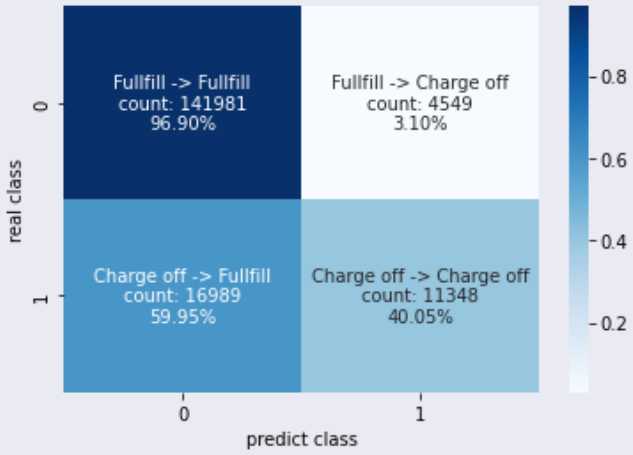
ex) 3-cv



Lasso Regression 결과

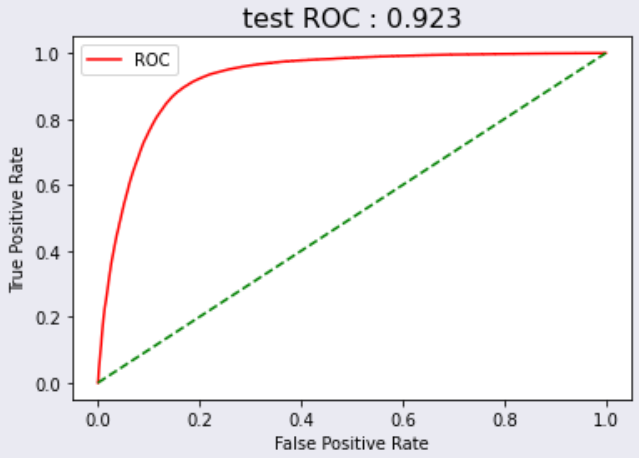
Lasso Regression의 결과는 다음과 같다.

[Confusion Matrix]



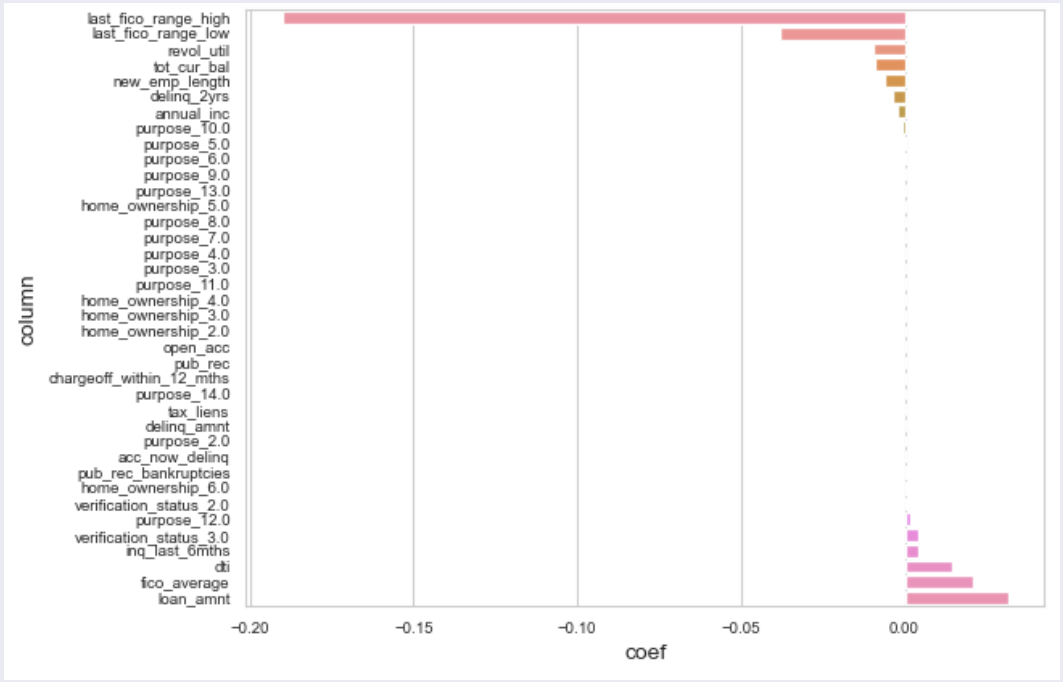
직관적인 이해를 위해 각각
TNR*100, FPR*100, FNR*100,
TPR*100으로 변환하여 도식화

[AUROC score]



정규화 방법에 따른
AUROC 차이는 없음
(0.923으로 동일)

[Coefficients]

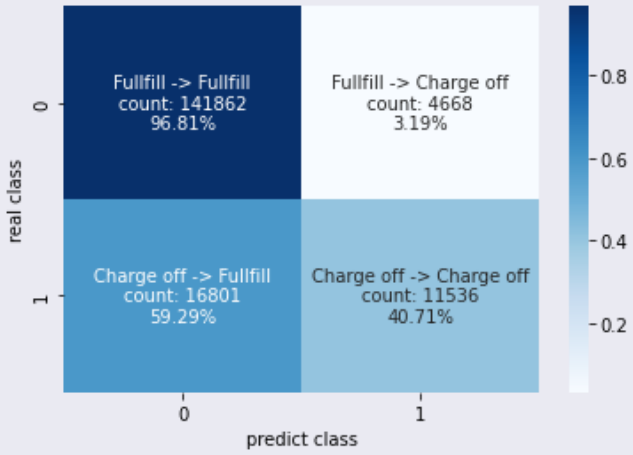


Lasso 정규화의 정의 상 계수가
0인 변수들이 다수 관찰됨

Ridge Regression 결과

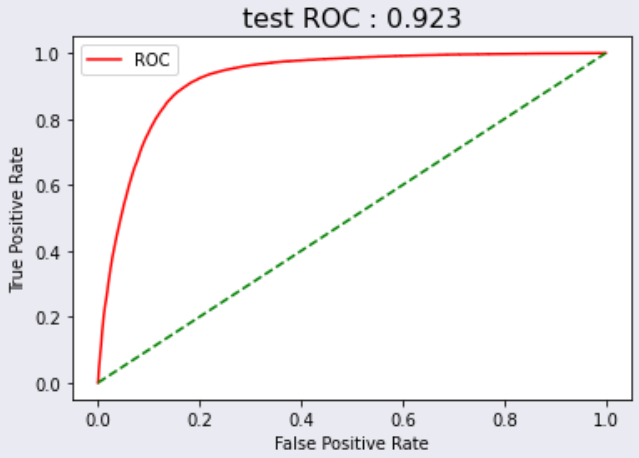
Ridge Regression의 결과는 다음과 같다.

[Confusion Matrix]



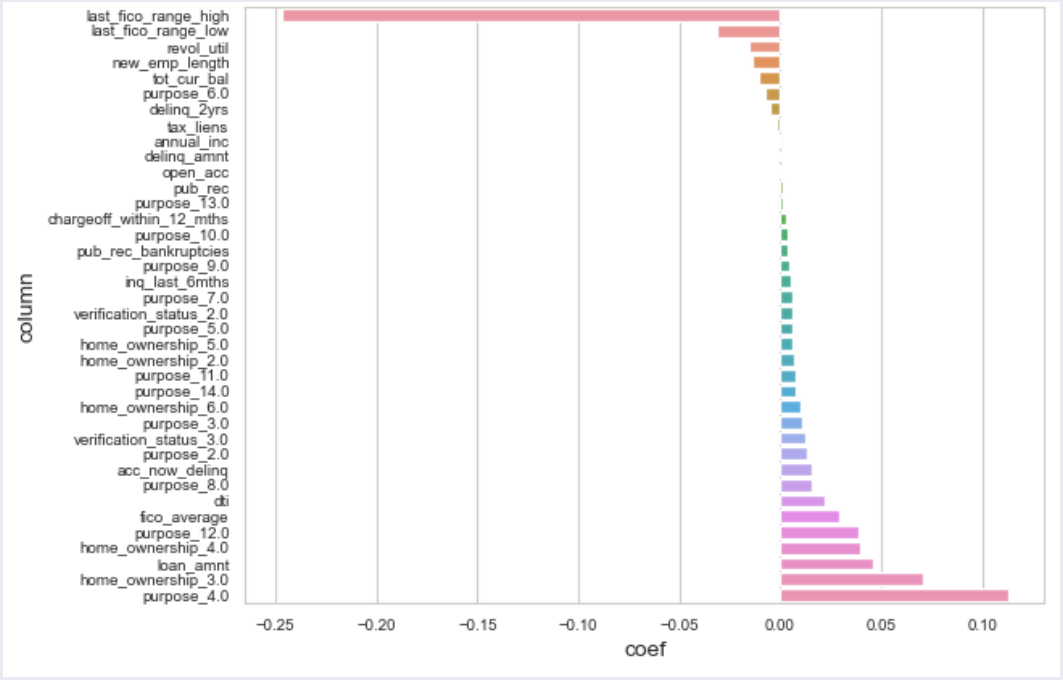
Lasso에 비해 1로 예측하는 비율 (TPR, FPR) 이 높음

[AUROC score]



AUROC는 0.923으로 동일

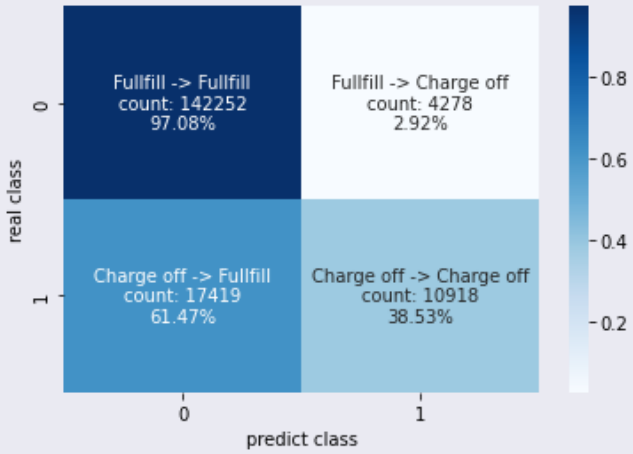
[Coefficients]



Elastic Net 결과

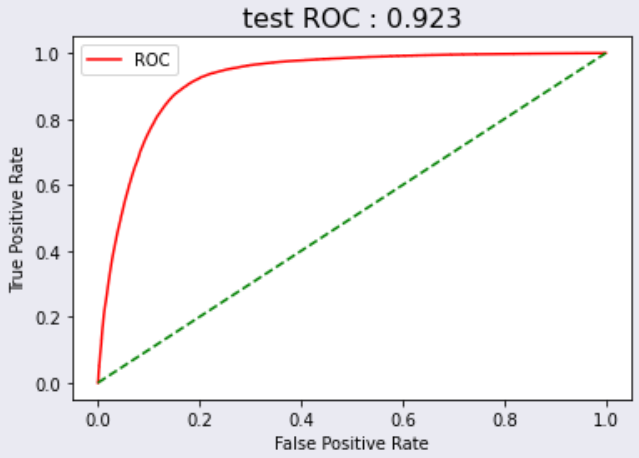
Elastic Net Regression의 결과는 다음과 같다.

[Confusion Matrix]



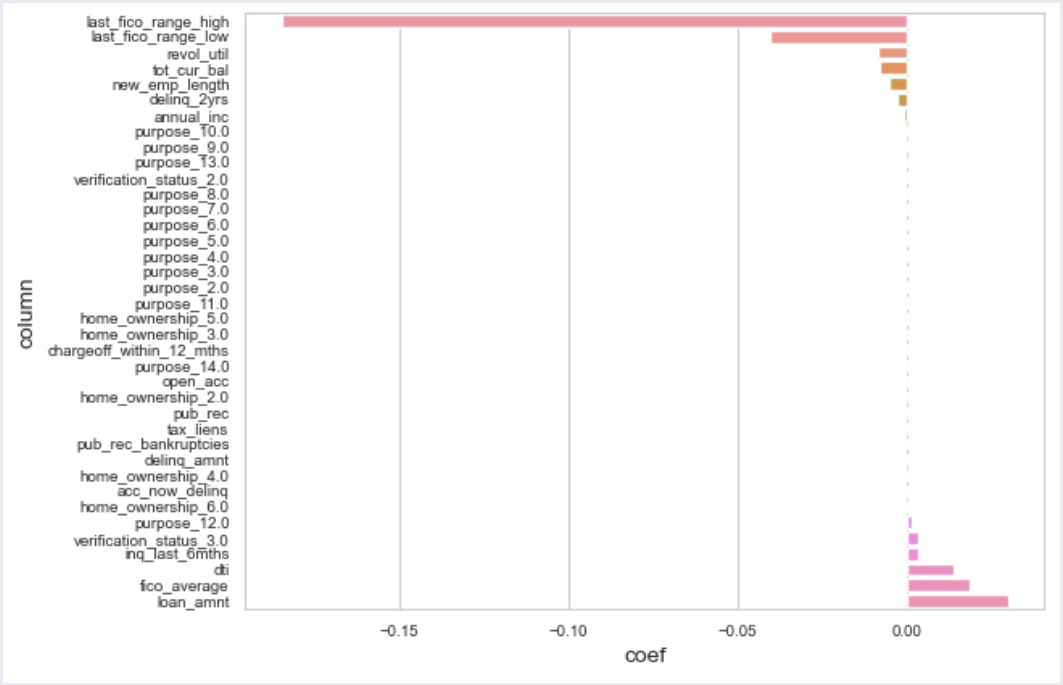
1로 예측하는 비율
(TPR, FPR) 이 가장 낮음

[AUROC score]



AUROC는 0.923으로 동일

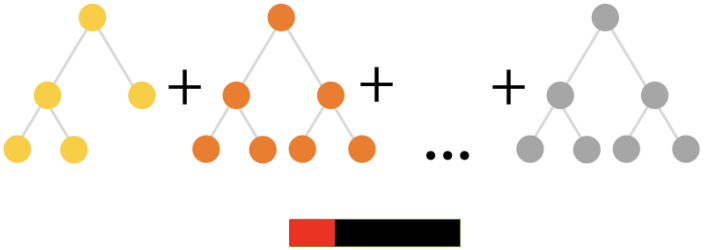
[Coefficients]



Tree Based Models

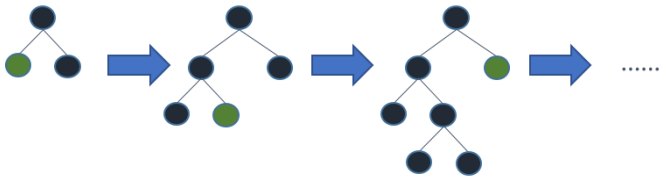
트리 기반 모델의 경우 **Catboost와 LGBM**을 사용하였다.
모델의 수가 적다는 피드백을 반영하여 기존 부스팅 모델보다 범주형 데이터에 특화된 **Catboost를 추가했**으나
LGBM과 AUROC 차이가 거의 없었고, 연산이 느리다는 단점이 있어 최종 모델은 LGBM으로 결정하였다.

CatBoost



nth tree with smaller error

LightGBM



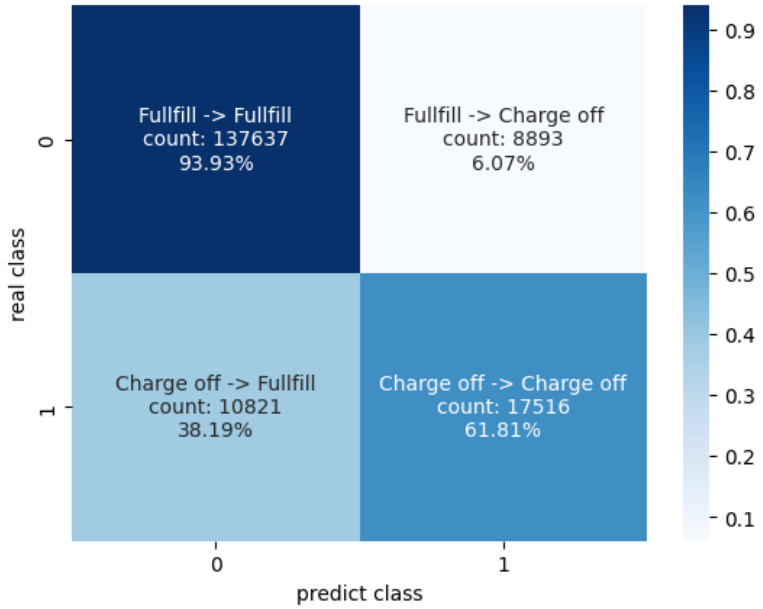
Leaf-wise tree growth

위 모델들은 카테고리형 변수를 그대로 입력 변수로 사용할 수 있어 one-hot-encoding을 사용하지 않고 학습시켰다.
최적 파라미터를 구하기 위해 베이지안 최적화를 사용하였으며 Logistic Regression과 동일하게 5-Fold Cross Validation을 수행하였다.

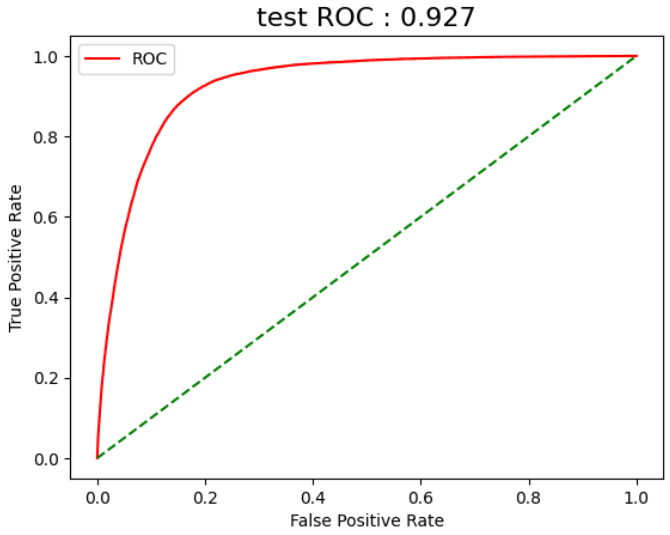
Catboost 결과

Catboost 결과는 다음과 같다.

[Confusion Matrix]



[AUROC score]



[Hyperparameters]

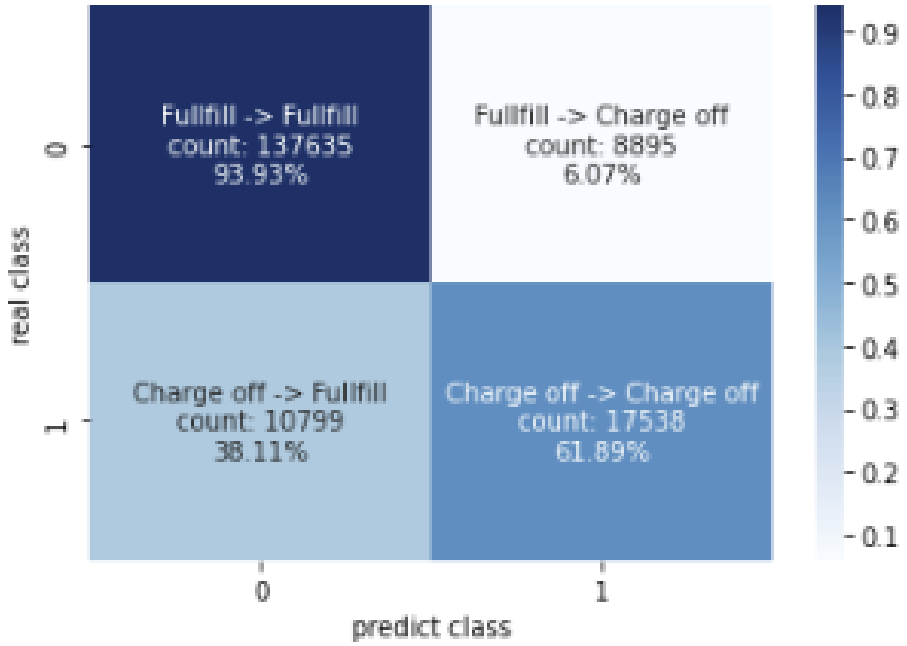
```
grid_cat.best_params_
```

```
{'depth': 6,
 'early_stopping_rounds': 100,
 'eval_metric': 'AUC',
 'iterations': 1000,
 'l2_leaf_reg': 10,
 'leaf_estimation_iterations': 10,
 'learning_rate': 0.1,
 'loss_function': 'Logloss',
 'random_seed': 42,
 'use_best_model': True,
 'verbose': 1}
```

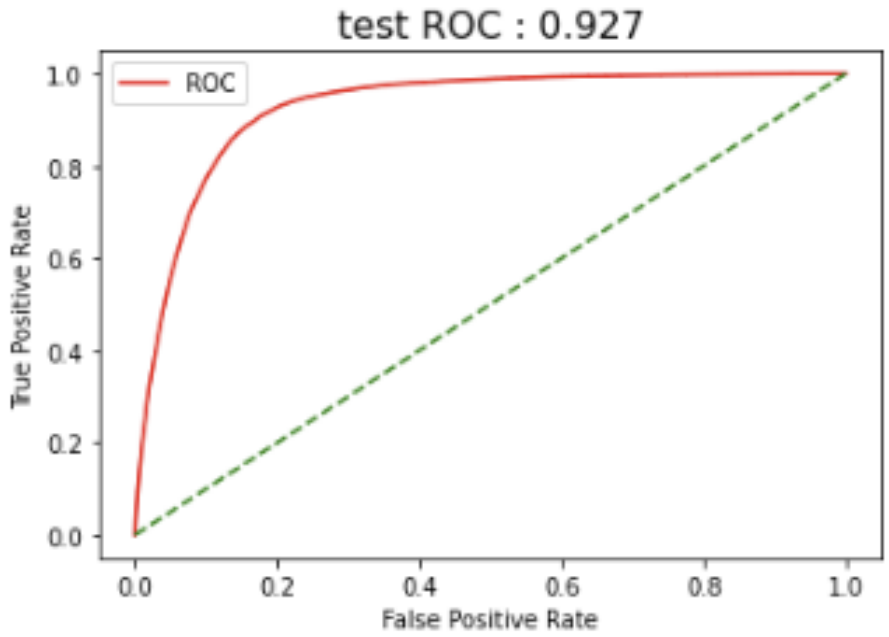
LGBM 결과

LGBM 결과는 다음과 같다.

[Confusion Matrix]



[AUROC score]



전체 모델 성능 비교

모든 모델의 성능을 비교한 결과, LGBM을 최종 모델로 선택되었다.

	Ridge	Lasso	Elastic Net
5-Fold mean AUROC	0.9233	0.9233	0.9233
Test AUROC	0.9229	0.9230	0.9230
Test F1 score	0.5178	0.5130	0.5015

Logistic Regression의 경우 AUROC가 동일할 때 순서대로 Test AUROC와 F1 score를 평가 지표로 사용하였다.
이 중에서는 Lasso Regression이 가장 좋은 성능을 보였다.

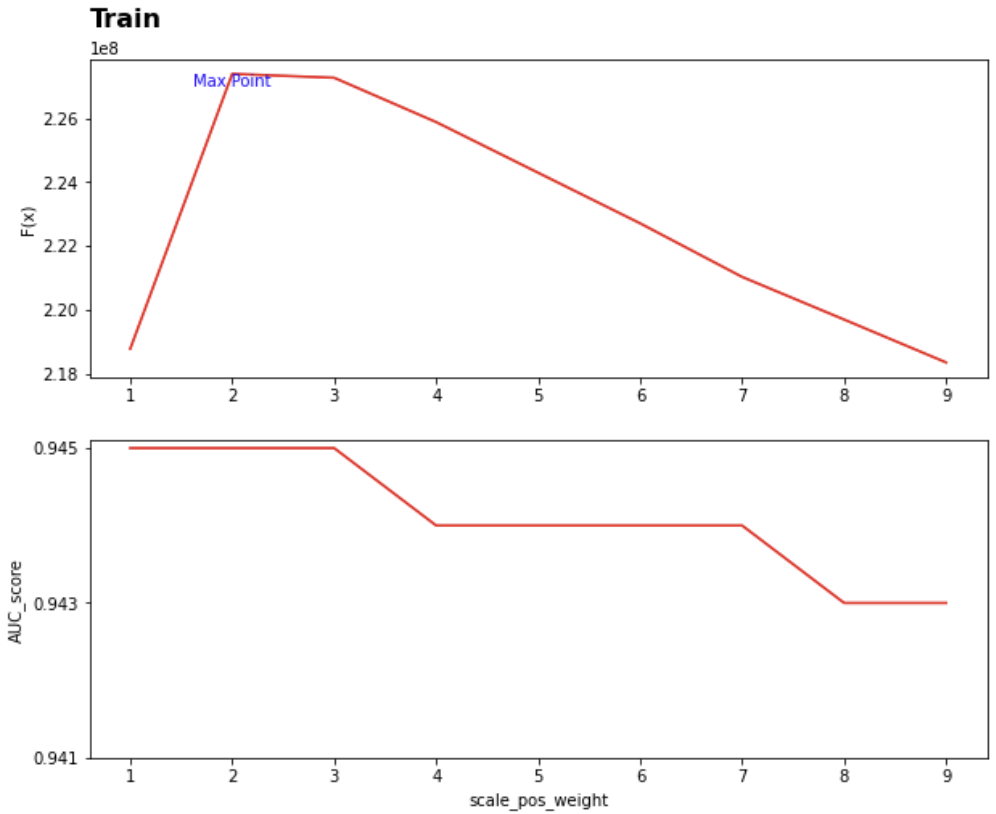
	Lasso	Catboost	LGBM
5-Fold mean AUROC	0.9233	0.9277	0.928
Test AUROC	0.9230	0.9273	0.9274

트리 기반 모델까지 포함했을 때에는 LGBM의 성능이 가장 좋아 최종 모델로 선정하였다.

결론 및 해석

최종 모델

LGBM에 대해 **목적함수 값을 최대화**하는 과정은 다음과 같다.



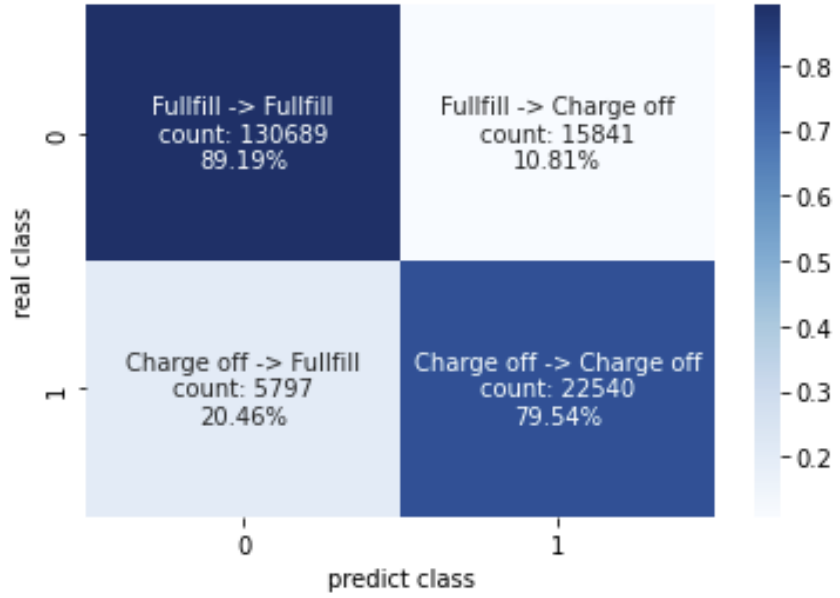
[scale_pos_weight에 따른 목적함수 값 및 AUROC score]

- AUROC score를 평가지표로 사용했던 것은 모델의 성능을 전체 threshold range에서 확인하여 가능한 모든 확률 스펙트럼에 대해 최고의 성능을 내는 모델을 도출하기 위함이었다.
- LGBM의 경우 1에 가중치를 주는 파라미터인 scale_pos_weight를 사용하여 임계값의 모든 변화를 확인하는 대신 해당 파라미터를 조정하여 동일한 효과를 얻었다.

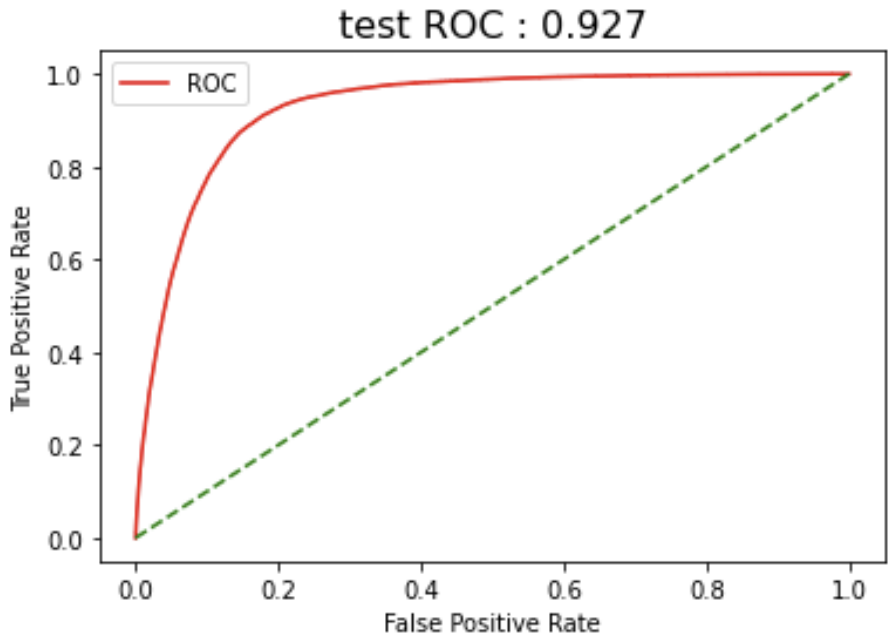
최종 모델

목적함수를 최대화하는 scale_pos_weight=2일 때의 결과는 다음과 같다.

[Confusion Matrix]



[AUROC score]



정확도: 0.8763
정밀도: 0.5873
재현율: 0.7954
F1: 0.6757

결과 해석: Shapley Value

Best Model을 찾은 뒤에는 결과를 해석하고자 어떤 변수들이 크게 작용했는지 확인하는 과정을 거쳤다.
초기에는 **Shapley Value**를 통해서만 접근하였으나, 발표 피드백을 반영하여 **자체적으로도 식을 만들어 변수 영향력을 확인**해보았다.

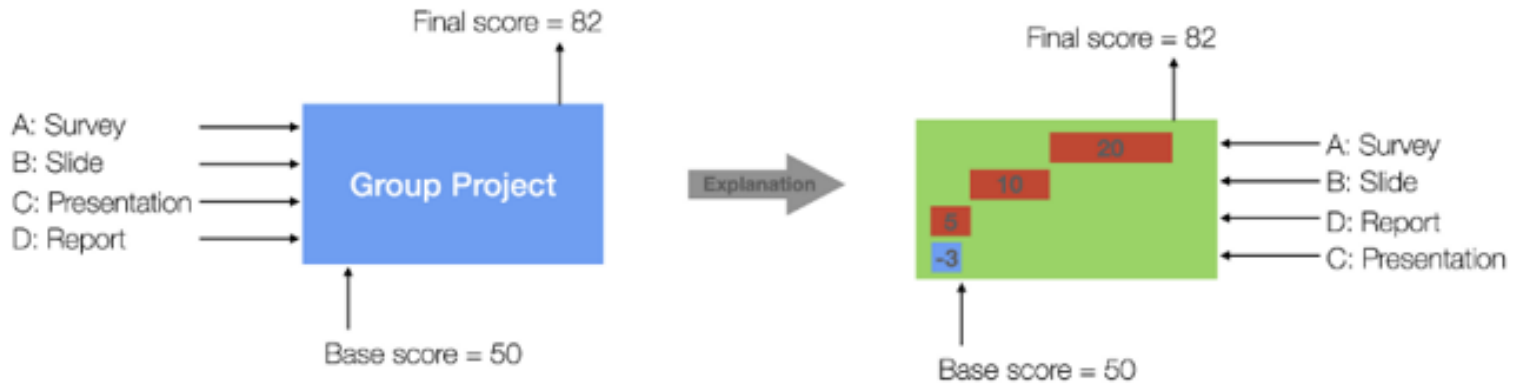
1. Shapley Value

Shapley Value는 노벨경제학상 수상자 Lloyd Shapley가 협력 게임에서 개별 플레이어들의 기여도를 수치화한 것에서 비롯되어
현재는 머신러닝 모델의 결과를 설명하는 데 사용되고 있는 게임이론적인 접근 방식이다.
특히 랜덤포레스트, XGBoost, 딥러닝 등 해석하기 어려운 blackbox 모델을 해석하는 데 주로 사용되고 있다.

결과 해석: Shapley Value

2. 예시를 통한 Shapley Value의 이해

Shapley Value는 다른 변수와의 상호작용을 고려하여 한 변수의 영향력을 측정한다.



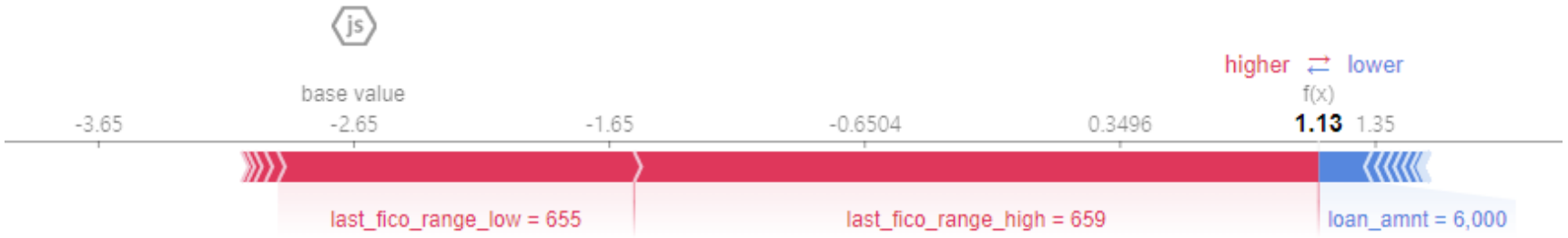
ex) 4명이 한 조를 이뤄 과제를 수행하여 최종 82점을 받은 상황:
각 사람이 82점에 기여한 정도를 A: +20, B:+10, D: +5, C:-3의 수치로 반환함

위와 같이 Shapley Value를 이용하면 A는 긍정적으로 +20만큼, C는 부정적으로 3만큼 기여했음을 알 수 있다.

변수 값에 따른 output의 변화를 측정할 수 있다는 점을 모델 해석에 적용하면 모든 변수에 대해 각각의 영향력을 확인할 수 있다.

결과 해석: Shapley Value

3. 1개 데이터에 대한 Shapley Value



y_pred[1]	y_pred_proba[1]
1.0	0.7557387605065783

Test data 중 Default가 발생한 1개의 레코드에 대한 Shapley Value와 예측 라벨 및 확률 값이다.

1의 방향으로 판단하는 데에는 last fico range low와 high가 크게 작용했으며, high의 영향력이 더 컸음을 알 수 있다.

반대로 loan amount는 1로 예측하는 확률 값을 낮추는 역할을 하고 있다고 해석할 수 있다.

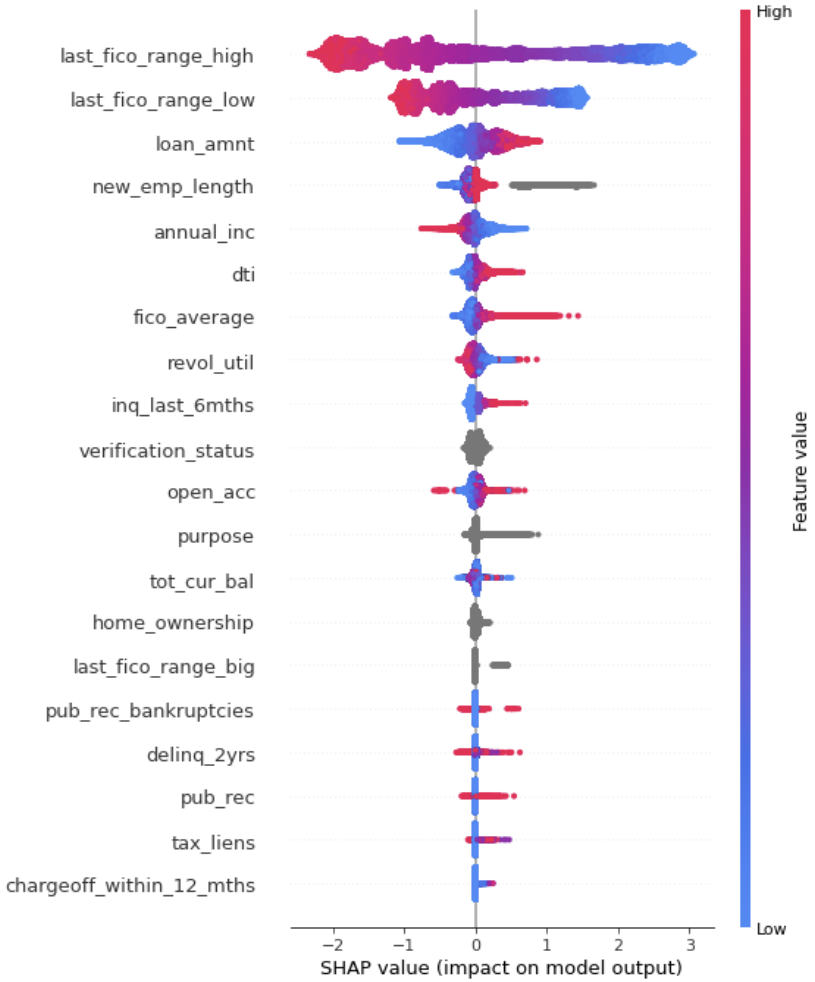
또한, 그림에 나타난 f(x)=1.13이라는 값은 다음의 식에 따라 p(x)가 0.5보다 크다는 것을 의미한다.

$$\text{odds} = \frac{p(x)}{1-p(x)} = e^{f(x)} \qquad logit = \log\left(\frac{p(x)}{1-p(x)}\right) = f(x)$$

$$f(x) = 0 \text{ 일 때 } p(x) = 0.5$$
$$f(x) > 0 \text{ 일 때 } p(x) > 0.5$$
$$f(x) < 0 \text{ 일 때 } p(x) < 0.5$$

결과 해석: Shapley Value

4. 전체 변수의 Shapley Value



여기서 붉은색은 큰 feature 값, 파란색은 작은 feature 값을 뜻한다.
또한 $x < 0$ 은 부정적 영향, $x > 0$ 은 긍정적 영향을 나타낸다.

즉, Last fico range high가 전체 변수 중
Default를 판단하는 데 가장 큰 역할을 하였으며
Last fico range high 값이 높을 수록 output을 0으로
예측하는 경향이 있었음을 알 수 있다.

추가로, 회색 영역은 카테고리형 변수 또는 결측치에 해당한다.

결과 해석: 자체 척도

Shapley Value가 변수의 영향력을 파악하기에 적절한 방법이 아닐 수 있다는 피드백을 고려하여
그와 별개로 다른 접근법으로 식을 구성하여 변수의 영향력을 산출한 뒤 결과를 비교해 보았다.

1. 자체 척도 원리

	Feature A	Feature B	Feature C	Feature D	Feature E	예측 확률값
0	12000	10.20	0	27898	692	0.47
1	14000	17.40	2	19286	672	0.68
2	24000	11.73	1	105032	687	0.35

1) 모든 행에 대해 A value – avg(A) 산출

	Feature A	Feature B	Feature C	Feature D	Feature E	예측 확률값
0	-4.667	10.20	0	27898	692	0.43
1	-2,667	17.40	2	19286	672	0.67
2	7,333	11.73	1	105032	687	0.51

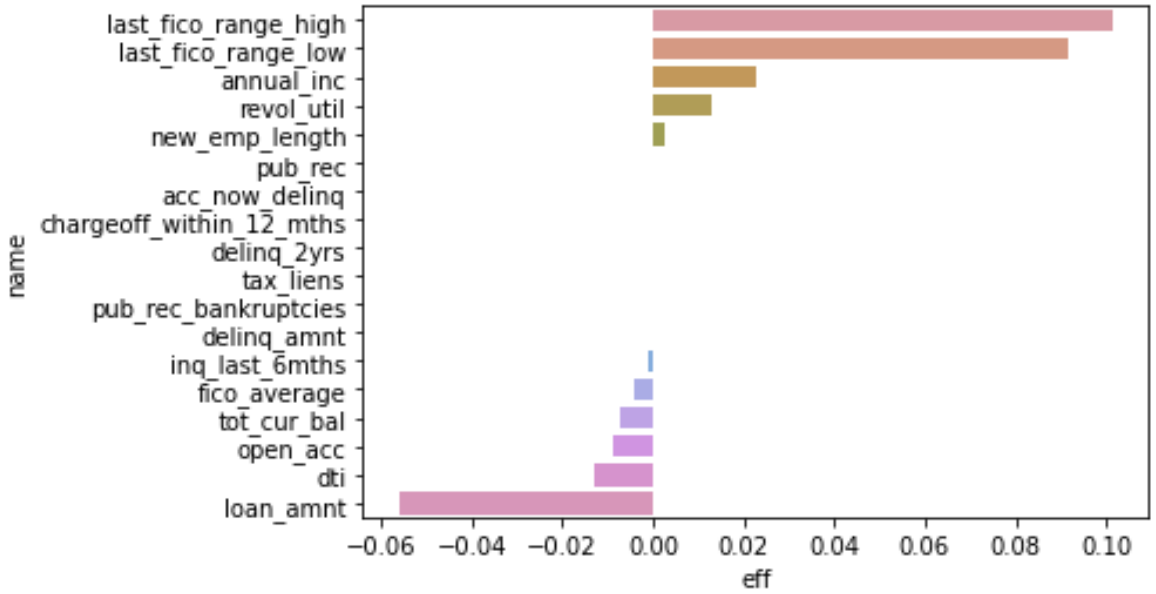
2) 예측 확률값의 변화를 통해 영향력 측정

이러한 작업을 모든 변수에 대해 반복하였다.
본 평가 방식은 연속형 변수에만 적용 가능한 한계가 있으나, 영향력을 평가하는 데에는 충분히 유효하다고 판단하였다.

결과 해석: 자체 척도

2. 자체 척도 평가 결과

자체적으로 평가한 결과와 Shapley Value를 통해 확인한 변수 영향력이 크게 다르지 않음을 알 수 있었다.




```
open_acc 영향력 -> -0.00865579420270746 5
loan_amnt 영향력 -> -0.05606442539959086
new_emp_length 영향력 -> 0.0025758511763987204
chargeoff_within_12_mths 영향력 -> 0.0
fico_average 영향력 -> -0.003981117698681927
delinq_2yrs 영향력 -> 0.0
last_fico_range_low 영향력 -> 0.09164777872776374
delinq_amnt 영향력 -> -1.6594011014075971e-06
revol_util 영향력 -> 0.012739833941041 84
dti 영향력 -> -0.012876436291555037
annual_inc 영향력 -> 0.022593033278773 82
tax_liens 영향력 -> 0.0
tot_cur_bal 영향력 -> -0.007118443243053 59
pub_rec_bankruptcies 영향력 -> 0.0
inq_last_6mths 영향력 -> -0.0008566135290851487
acc_now_delinq 영향력 -> 0.0
last_fico_range_high 영향력 -> 0.10137942515138 68
pub_rec 영향력 -> 0.0
```

Last fico range와 Annual income, Revolving utilization rate가 큰 양의 상관계수를 가지며
Loan amount, DTI, Open account 개수 등의 변수가 음의 상관계수 중 유의미했다는 것은 동일하다.
Fico average의 영향력의 크기는 다소 차이가 있으나 결국 중요한 변수를 식별하는 것과는 무관하였다.

최종 결과

최종적으로 진행한 **10-Fold Out of Sample Test**의 결과이다.
Out of Sample에 대해서도 높은 예측 성능을 보이는 것을 확인하여 유의미한 모델을 구축했다고 결론지었다.

	Fold1	Fold2	Fold3	Fold4	Fold5	Fold6	Fold7	Fold8	Fold9	Fold10	Mean	STD
accuracy	0.875886	0.876481	0.880781	0.875520	0.874279	0.877848	0.875286	0.876018	0.873776	0.873639	0.875952	0.001924
precision	0.583629	0.584792	0.596350	0.583054	0.579858	0.587588	0.582289	0.585605	0.579524	0.578301	0.584099	0.004692
recall	0.795390	0.798805	0.799659	0.792888	0.791406	0.805635	0.793398	0.782584	0.782869	0.790268	0.793290	0.006494
Auc_score	0.925837	0.926407	0.930358	0.925268	0.926375	0.928754	0.927953	0.927443	0.923557	0.924884	0.926684	0.001806
f1	0.673251	0.675247	0.683200	0.671971	0.669314	0.679549	0.671645	0.669915	0.666021	0.667869	0.672798	0.004793

감사합니다