통계 데이터 사이언스

서울대학교 빅데이터 핀테크 과정

Lending Club 데이터 분석 보고서

LPM, Logistic/Probit, SVM을 중심으로

1조 | 김서현 김단아 김연성 서문홍 조건우 최지은

피드백 정리

변수 설정

- 스케일을 통해 여러 변수들의 단위를 맞추어야 한다.
- 기업의 이익 측면에서 어떤 변수를 사용할지 정해야 한다.
- 3차원 공간에 점을 찍고, 1과 0의 점의 색을 구분해서 logit의 hyper plane 그림을 그려서 시각화하는 것이 필요하다.

전처리

- 데이터를 생각해보면 대출을 신청했지만 거절당한 사람들의 데이터는 없다.
- 실제 랜딩클럽이 활용할 때 현재 모델은 selected된 샘플만 가지고 개발된 것이기 때문에, 분석 목적을 적용하기 곤란하다.
- 모델 자체의 설명력을 높이기 위해 변수를 선택하였으나, 유의미한 변수를 제거한 부분 또한 존재했다.

Thresholds

- 기존의 threshold를 찾는 방식은 accuracy 최대화를 목표로 구간별 최적화 방법을 적용하여 최적의 threshold를 도출하는 것이었다.
- 이러한 방식으로 도출한 threshold값을 바탕으로 대출 승인 여부를 결정하면 대부분의 경우가 대출이 거부되는 문제가 발생했다.

기타

- 계수들의 통계적 의미가 크게 중요하지 않다. 다시 말해서 x와 y사이의 인과관계를 규명하는 것에 포인트를 두지 말아야 한다.

피드백 반영내용

변수 설정

- 스케일을 통해 여러 변수들의 단위를 맞추었다.
- 모형 우수성 평가 지표인 '비용 최소화' 여부를 판단하기 위해서는 비용을 구성하는 변수들에 대한 고려가 필요하다. Lendingclub 관점에서의 손익계산 모델을 만들어, 이에 맞는 두 가지 변수를 기준으로 분석을 진행하였다.

전처리

- 투자자와 차입자 간의 거래가 성사되기 이전에 알 수 없는 변수들은 분석에서 제외했다.
- Lendingclub 이익 극대화를 목표로, 일련의 가정 하에서의 installment와 int_rate를 중요한 설명변수로 선택했다.

Thresholds

- 앞에서 설정한 Lendingclub의 손익에 대한 모델을 바탕으로 이를 극대화하는 방식으로 thresholds를 구하였다.
- 합리적인 thresholds 도출을 통해 매우 제한적인 대출 승인 방식을 극복하고, 회사의 이익을 극대화하고자 했다.

INDEX

01. Introduction

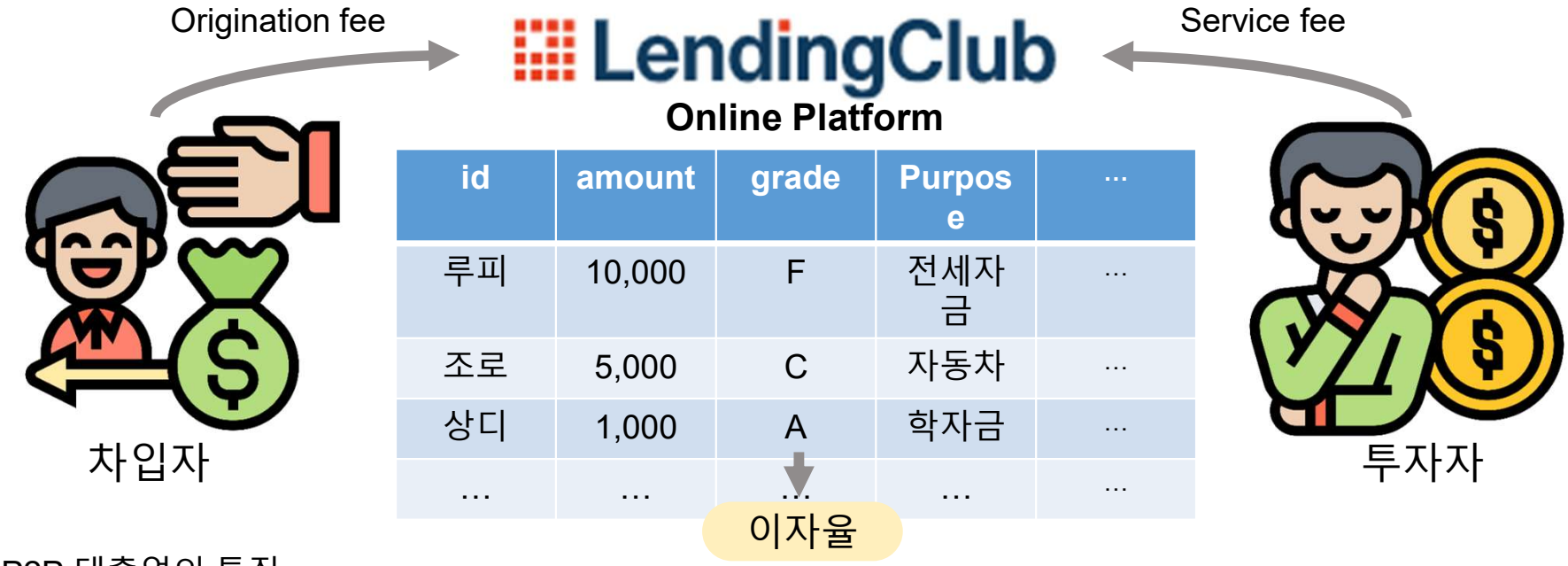
은 행 업 : 간접금융

수익원: 예대마진

은행업의 특징 및 한계

> 은행은 규모의 경제를 통해 예금자와 차입자를 연결하여 자금 배분의 효율성을 달성한다.
은행업은 규제가 가장 강한 산업 중 하나이며, 전염효과를 고려하여 수익성 뿐만 아니라 유동성과 안정성 역시 고려한다.
신용등급은 낮지만 자금에 대한 수요가 강한 금융취약계층 여전히 금융 사각시대에 놓이게 된다.

P2P 대출업 : 직접금융



id	amount	grade	Purpose	…
루피	10,000	F	전세자금	…
조로	5,000	C	자동차	…
상디	1,000	A	학자금	…
…	…	…	…	…

P2P 대출업의 특징

> P2P 대출업체는 차입자의 개인 정보를 바탕으로 이자율을 산출하여 이를 투자자에게 제공한다.
P2P 대출업체 수익원은 차입자에게 받은 origination fee와 투자자에게 받은 service fee로 구성된다.
금융취약계층에 금융에 대한 접근성이 높여 포용적 금융을 실현하게 한다.

부도예측모형의 유용성

금융업에 대한 이해

금융업의 내재적 불안정성과 전염효과

뱅크런

- ✔ 한 은행의 도산이 정상적으로 영업하던 다른 은행으로 삽시간으로 퍼진다.
- ✔ 예측과 달리 실제 개인 파산이 더 많이 이루어진다면 큰 사회적 혼란이 야기된다.

부도예측모형의 예측력은 미시적으로, 거시적으로 매우 중요함

부도예측모형의 목적

모형의 잠재적 수요자

미시적 관점

- ✔ **P2P 대출업체**
  : 모형을 통한 선별을 통해 역선택을 방지할 수 있다.
- ✔ **투자자**
  : 합리적인 투자의사결정에 도움을 준다.
- ✔ **차입자**
  : 기준 미달인 차입자의 장래의 고통을 덜게 해준다.

거시적 관점

- ✔ **금융당국**
  : 보편적으로 P2P 대출은 예금보호상품이 아니다.
  제도적 안전장치가 기술 발전을 따라가지 못하는 형국이다.
  합리적인 규제감독 방안을 마련하는 데 도움이 될 것이다.

→ 1) 예측의 정확도(Accuracy) 최대화

2) 시간 효율 최적화

부도예측모형의 '구체적' 목적

※ 기존 발표의 문제 중 하나는 모형의 '목적'이 불분명하다는 점이었다. 이를 개선하기 위해, 부도예측모형이 어떤 행위자를 위한 것인지를 구체화했다. 1조는 P2P에 직접적으로 참여하는 세 행위자 중 P2P 업체(Lending Club)를 부도예측모형 활용의 주체로 선정했다. 이후, 변수 선택 및 모델 우수성 평가를 철저히 'P2P 업체'의 관점에서 진행했다.

P2P 대출의 주요 참여자

- ✔ **P2P 대출업체** **=> 타겟으로 선정!**
- ✔ **투자자**
- ✔ **차입자**

** P2P 대출업체는 투자자와 차입자에게 대출 플랫폼을 제공하고, 그 대가로 투자자와 차입자로부터 '수수료'를 받는다. 차입자의 부도로 인해 연체된 원금 자체가 P2P 대출업체에게 손실이 되지 않는다는 점이 P2P 대출업체의 가장 특징적인 이해관계이다.*

부도예측모형의 '구체적' 목적

✔ **P2P 대출업체** **=> 타겟으로 선정!**

P2P 대출업체의 목적

- *P2P 대출업체의 수입원은 차입자와 투자자가 매달 지불하는 '수수료'이다.*
- *따라서 P2P 업체는 자신의 고객인 차입자와 투자자를 최대한 많이 확보해야 한다. 그러나 두 고객을 유치하기 위한 전략이 상충한다는 점에서 P2P 대출업체의 목적을 두 가지로 구분할 수 있다.*

1) 단기: 많은 차입자에 대한 대출 활성화

(대출)건당 수수료를 주 수입원으로 삼는 P2P업체는 최대한 많은 차입자에게 대출을 허가해야, 대출 건수를 높일 수 있다. 부도 정확도가 너무 높은 모형은 신용이 확정적으로 좋은 차입자에 대한 대출만을 허락해주어, 많은 고객 유치라는 목적에 어긋난다.

2) 장기: 안정적인 투자자 유치를 위한 부도 예측의 정확도 높이기

P2P 업체에서 대출을 허가한 차입자로부터 부도가 발생할 경우, 투자자는 자신이 기대한 원금 및 이자를 받지 못하게 된다. P2P 업체의 부도 발생 빈도가 높다면, 투자자들은 해당 업체에 대한 신뢰를 잃고, 안정적인 수익 창출을 위해 다른 P2P 업체에 투자할 것이다. 따라서 P2P 업체의 안정적인 운영을 위해 정확한 부도예측모형을 개발해야 한다.

=> 1과 2는 trade-off의 관계!

부도예측모형의 목표

: 차입자와 투자자를 모두 최대로 확보할 수 있는 '적당히 정확한' 부도 예측 모형을 만들자!

데이터 분석 과정

※ 기존 변수 선정 과정에서 Correlation의 절댓값만을 고려한 결과, 사전 대출 심사 과정에서 파악하기 힘들거나 종속변수(depvar)와의 양의 관계가 당연한 변수들이 선정되었다. (예: Funded_amnt_inv(투자자가 약정한 총 금액), Total_rec_prncp(현재까지 받은 원금))
따라서 모형의 우수성을 평가할 수 있는 지표를 우선적으로 설정한 후, 지표를 도출하기 위해 필요한 변수를 선정하는 것으로, 변수 설정 방식을 수정했다.

P2P 업체의 수익 구조

- *모형 우수성 평가 지표를 제시하기 위해, P2P 대출업체의 수익 구조에 대한 이해가 필요하다.*
- *P2P 업체의 일반적 수수료: 차입자의 월 지급액 x 이자율에 비례*

P2P 대출업체의 정상 수익
= 대출을 해주었고 실제로 부도가 안 난 경우(TN)
= 차입자의 월 지급액 x 이자율

부도예측 실패 시, P2P 업체의 비용
= 대출해줬는데 부도난 경우(FN) + 부도 안 날 사람에게 대출을 안 해준 경우(FP)
= FN의 비용(차입자의 월 지급액)
+ FP의 비용(차입자의 월 지급액 x 이자율)

부도 예측 모형의 우수성 평가 지표

: 부도 예측 실패로 인한 P2P 업체의 비용을 최소화할 수 있는 예측 모형

데이터 분석 과정

LendingClub 관점에서의 손익계산

= 대출을 해줬고 실제로 완납한 경우(TN)
- 대출해줬는데 부도난 경우(FN)
- 부도 안 날 사람에게 대출을 안 해준 경우(FP)

= TN의 수익 (installment x int_rate)
- FN의 비용 (installment)
- FP의 비용 (installment x int_rate)

■ LendingClub의 수익을 최대화하는 Optimal Thresholds를 찾는 것을 목표로 하였다.
■ TP의 경우 대출을 진행하지 않았고 또한 실제로 부도난 경우이므로, LendingClub 입장에서 추가적인 수익이나 비용이 발생하지 않는다. 따라서 손익계산에서 제외하기로 하였다.

선택한 변수 설정

※ 모형 우수성 평가 지표인 '비용 최소화' 여부를 판단하기 위해서는 '비용'을 구성하는 변수들에 대한 고려가 필요하다. 따라서 다음 두 가지 변수를 분석에 포함했다.

Installment	대출이 시작된 경우 차용자가 지불해야 하는 월별 지불액
Int_rate	대출금리 (이자율)

02. Backgrounds

Lending Club 데이터 구조 파악

Lending Club DataFrame

> 1092919개 행, 333개 열을 갖는 데이터프레임

```
lcd.shape
```

```
(1092919, 333)
```

> 333개 열에 대한 기술통계량

```
lcd.describe()
```

	loan_amnt	funded_amnt	funded_amnt_inv	int_rate	installment	annual_inc	dti	delinq_2yrs	fico_range_low
count	1.092919e+06	1.092919e+06	1.092919e+06	1.092919e+06	1.092919e+06	1.092919e+06	1.092919e+06	1.092919e+06	1.092919e+06
mean	1.364630e+04	1.364628e+04	1.364033e+04	1.263927e-01	4.312534e+02	7.562999e+04	1.818105e+01	3.394543e-01	6.942867e+02
std	8.441163e+03	8.441145e+03	8.438033e+03	4.361999e-02	2.652486e+02	8.994245e+04	8.321510e+00	9.065449e-01	3.061425e+01
min	1.000000e+03	1.000000e+03	7.750000e+02	5.320000e-02	4.930000e+00	3.000000e+03	-1.000000e+00	0.000000e+00	6.600000e+02
25%	7.125000e+03	7.125000e+03	7.100000e+03	9.170000e-02	2.369800e+02	4.500000e+04	1.192000e+01	0.000000e+00	6.700000e+02
50%	1.200000e+04	1.200000e+04	1.195000e+04	1.229000e-01	3.627800e+02	6.400000e+04	1.768000e+01	0.000000e+00	6.850000e+02
75%	1.900000e+04	1.900000e+04	1.900000e+04	1.531000e-01	5.732100e+02	9.000000e+04	2.410000e+01	0.000000e+00	7.100000e+02
max	4.000000e+04	4.000000e+04	4.000000e+04	3.099000e-01	1.715420e+03	6.100000e+07	4.996000e+01	3.000000e+01	8.450000e+02

8 rows × 333 columns

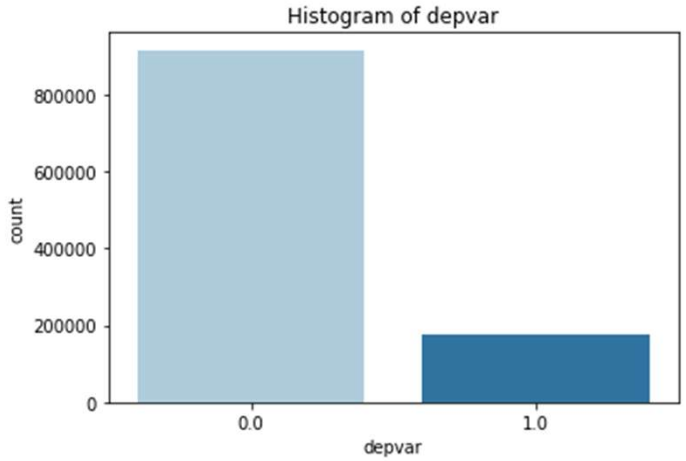
현실적인 데이터 분석의 효율성과
적정 수준의 예측력을 확보하기 위해
Column 개수를 줄여야 함

Lending Club 데이터 구조 파악

종속변수 depvar

> depvar == 0 (부도 아님), depvar == 1 (부도)
> 부도 비율 = 16.18 %

```
sns.countplot(x="depvar", data=lcd, palette='Paired')
plt.title("Histogram of depvar")
plt.show()
```



결측치 확인

```
np.sum(lcd.isnull().sum())
```

```
0
```

모든 데이터 값이 0인 Column 조회

```
del_list = []
for i in range(df.shape[1]):
    if df.iloc[:,i].mean() == 0:
        del_list.append(i)
print (del_list)
```

```
[125, 126, 127, 128, 129, 130, 131, 132, 133, 134, 135, 136, 137,
1, 152, 153, 154, 155, 156, 157, 158, 159, 160, 161, 162, 163, 16
178, 179, 180, 181, 182, 183, 184, 185, 186]
```

모든 행의 값이 0인 컬럼(열) 조회

-> 62개 확인 (전처리 단계에서 제거)

Lending Club 데이터 구조 파악

Column Variable List 조회

> 333개의 변수 이름의 리스트 확인

```python
import re

col_list2 = []

for i in lcd.columns:
    p = re.compile("[^0-9]")
    new = "".join(p.findall(i))
    col_list2.append(new)

print(col_list2)
```

```
['loan_amnt', 'funded_amnt', 'funded_amnt_inv', 'int_rate', 'installment', 'annual_inc', 'dti', 'delinq_yrs', 'fico_range_low', 'fico_rang
e_high', 'inq_last_mths', 'open_acc', 'pub_rec', 'revol_bal', 'revol_util', 'total_acc', 'out_prncp', 'out_prncp_inv', 'total_pymnt', 'tot
al_pymnt_inv', 'total_rec_prncp', 'total_rec_int', 'total_rec_late_fee', 'recoveries', 'collection_recovery_fee', 'last_pymnt_amnt', 'last
_fico_range_high', 'last_fico_range_low', 'collections__mths_ex_med', 'acc_now_delinq', 'tot_coll_amt', 'tot_cur_bal', 'chargeoff_within__
mths', 'delinq_amnt', 'pub_rec_bankruptcies', 'tax_liens', 'emp_length', 'emp_length', 'emp_length', 'emp_length', 'emp_length', 'emp_leng
th', 'emp_length', 'emp_length', 'emp_length', 'emp_length', 'emp_length', 'emp_length', 'home_ownership', 'home_ownership', 'home_ownersh
ip', 'home_ownership', 'home_ownership', 'home_ownership', 'verification_status', 'verification_status', 'verification_status', 'purpose',
'purpose', 'purpose', 'purpose', 'purpose', 'purpose', 'purpose', 'purpose', 'purpose', 'purpose', 'purpose', 'purpose', 'purpose', 'purpo
se', 'addr_state', 'addr_state', 'addr_state', 'addr_state', 'addr_state', 'addr_state', 'addr_state', 'addr_state', 'addr_state', 'addr_s
tate', 'addr_state', 'addr_state', 'addr_state', 'addr_state', 'addr_state', 'addr_state', 'addr_state', 'addr_state', 'addr_state', 'addr
_state', 'addr_state', 'addr_state', 'addr_state', 'addr_state', 'addr_state', 'addr_state', 'addr_state', 'addr_state', 'addr_state', 'ad
dr_state', 'addr_state', 'addr_state', 'addr_state', 'addr_state', 'addr_state', 'addr_state', 'addr_state', 'addr_state', 'addr_state',
'addr_state', 'addr_state', 'addr_state', 'addr_state', 'addr_state', 'addr_state', 'addr_state', 'addr_state', 'addr_state', 'addr_stat
e', 'addr_state', 'addr_state', 'initial_list_status', 'initial_list_status', 'elapsed_t', 'issue_d', 'issue_d', 'issue_d', 'issue_d', 'is
sue_d', 'issue_d', 'issue_d', 'issue_d', 'issue_d', 'issue_d', 'issue_d', 'issue_d', 'issue_d', 'issue_d', 'issue_d', 'issue_d', 'issue_
d', 'issue_d', 'issue_d', 'issue_d', 'issue_d', 'issue_d', 'issue_d', 'issue_d', 'issue_d', 'issue_d', 'issue_d', 'issue_d', 'issue_d', 'i
ssue_d', 'issue_d', 'issue_d', 'issue_d', 'issue_d', 'issue_d', 'issue_d', 'issue_d', 'issue_d', 'issue_d', 'issue_d', 'issue_d', 'issue_
d', 'issue_d', 'issue_d', 'issue_d', 'issue_d', 'issue_d', 'issue_d', 'issue_d', 'issue_d', 'issue_d', 'issue_d', 'issue_d', 'issue_d', 'i
ssue_d', 'issue_d', 'issue_d', 'issue_d', 'issue_d', 'issue_d', 'issue_d', 'issue_d', 'issue_d', 'issue_d', 'issue_d', 'issue_d', 'issue_
d', 'issue_d', 'issue_d', 'issue_d', 'issue_d', 'issue_d', 'issue_d', 'issue_d', 'issue_d', 'issue_d', 'issue_d', 'issue_d', 'issue_d', 'i
ssue_d', 'issue_d', 'issue_d', 'issue_d', 'issue_d', 'issue_d', 'issue_d', 'issue_d', 'issue_d', 'issue_d', 'issue_d', 'issue_d', 'issue_
d', 'issue_d', 'issue_d', 'issue_d', 'issue_d', 'issue_d', 'issue_d', 'issue_d', 'issue_d', 'issue_d', 'issue_d', 'issue_d', 'issue_d', 'i
ssue_d', 'issue_d', 'issue_d', 'issue_d', 'issue_d', 'issue_d', 'issue_d', 'issue_d', 'issue_d', 'issue_d', 'issue_d', 'issue_d', 'issue
d', 'issue_d', 'debt_settlement_flag', 'term', 'depvar', 'mths_since_last_delinq', 'mths_since_last_delinq', 'mths_since_last_delinq', 'mt
```

다수의 열이 더미변수 임을 확인

-> 비슷한 분포를 갖는 변수들을 전처리 과정에서 제거

모형 선정

분류(classification) : 독립 변수 값이 주어졌을 때 그 값과 가장 연관성이 큰 종속 변수 값(클래스)을 예측하는 문제

분류모형의 구분

확률적 모형 > 확률적 생성모형

주어진 데이터에 대해(conditionally) 각 카테고리 혹은 클래스가 정답일 조건부확률(conditional probability)를 계산하고,
베이즈 정리를 사용하여 간접적으로 조건부확률을 구함

판별함수 모형

주어진 데이터를 카테고리에 따라 서로 다른 영역으로 나누는 경계면(decision boundary)을 찾아낸 다음 이 경계면으로부터 주어진 데이터가 어느 위치에 있는지를 계산하는 판별함수(discriminant function)를 이용

모형 선정 기준

1) 수업시간에 배운 모형을 다양하게 활용하고, 각각의 특징을 비교 및 확인

1) 분류 모형의 두 종류인 확률적 모형과 판별함수 모형을 모두 사용

출처 | 데이터사이언스 스쿨, <5.3 분류모형>

모형 선정

선정한 분류 모델

종속변수가 Binary일 때 주로 사용 확률적 모형		Binary와 Multi Class 분류 모두에서 사용 판별함수 모형
LPM (Linear Probability Model)	Probit / Logistic	SVM (Support Vector Machine)
Binary한 종속변수가 독립변수 x에 따라 선형적으로 변화한다는 가정에 따라 선택 확률을 계산하는 모형	- Probit: 독립변수와 확률 간 비선형의 관계를 표준정규분포의 누적확률분포함수(프로빗 함수)를 이용하여 표현하는 모형 - Logit: Probit 모형과 유사하지만, 프로빗 함수 대신 로지스틱 분포의 확률분포함수 사용	데이터를 분류하기 위한 결정 경계를 정의하여, 이에 따라 새로운 데이터의 class를 분류하는 모형

1) 예측력과 2) 효율성(시간)을 기준으로 부도를 가장 잘 예측하는 최종 모델을 선택

03. Experiment

데이터 전처리

제거한 변수 목록

col_del_list

'total_pymnt'
'total_pymnt_inv'
'total_rec_int'
'total_rec_prncp'
'funded_amnt'
'total_rec_late_fee'
'last_pymnt_amnt'
'funded_amnt_inv'
'recoveries'
'collection_recovery_fee'
'last_fico_range_high'
'last_fico_range_low'

- 분석을 진행하기 전, 변수들에 대해서 실제 거래가 진행되기 전/후에 얻은 결과들을 나누어서 생각해보았다.
- 이유) 우리가 부도인지 아닌지에 대한 판단을 하는데 사용되는 변수들이 “거래가 진행된 이후의 결과값을 갖고 분석한다”는 방법과 모순이라고 판단 하였고, 앞서 언급한 거래 이후의 변수들은 분석에 사용하지 않기로 하였다.
- 그리고 앞서 말한 내용에 해당하는 변수들을 col_del_list라고 하였고, 그들은 왼쪽과 같다.

데이터 전처리

모든 데이터 값의 일치율이 높은 Column 제거

```
for i in list(lcd.columns):
    check = ""
    if list(lcd[i].value_counts(normalize=True))[0]*100 > 99.5: # 99.5 % 이상
        check = "   <<< del"
        col_del_list.append(i)
    print("{0} : {1:0.3f}%{2}".format(i, list(lcd[i].value_counts(normalize=True))[0]*100, check))
col_del_list = list(set(col_del_list))
```

```
loan_amnt : 7.638%
funded_amnt : 7.638%
funded_amnt_inv : 7.220%
int_rate : 3.353%
installment : 0.301%
annual_inc : 3.848%
dti : 0.076%
delinq_2yrs : 79.637%
fico_range_low : 9.524%
fico_range_high : 9.524%
inq_last_6mths : 58.169%
open_acc : 8.953%
pub_rec : 82.592%
revol_bal : 0.280%
revol_util : 0.344%
total_acc : 3.652%
out_prncp : 99.984%   <<< del
out_prncp_inv : 99.984%   <<< del
total_pymnt : 0.039%
```

모든 데이터 값의 일치율이 99.5% 이상인 행은 변수로써의 유의미한 차이가 없다고 가정하여 제거 (88개)

데이터 전처리

비슷한 분포를 갖는 Column 제거

```
np.array(del_list)
```

```
array([  0,   1,  12,  16,  17,  22,  28,  29,  32,  33,  35,  36,  38,
        39,  40,  41,  42,  43,  44,  45,  46,  47,  48,  50,  51,  57,
        60,  61,  62,  63,  64,  65,  66,  67,  68,  69,  70,  71,  72,
        73,  74,  76,  77,  78,  79,  80,  81,  82,  83,  84,  85,  86,
        87,  88,  89,  90,  91,  92,  93,  94,  95,  96,  97,  98,  99,
       100, 101, 102, 103, 104, 105, 106, 107, 108, 109, 110, 111, 112,
       113, 114, 115, 116, 117, 118, 119, 120, 121, 125, 126, 127, 128,
       129, 130, 131, 132, 133, 134, 135, 136, 137, 138, 139, 140, 141,
       142, 143, 144, 145, 146, 147, 148, 149, 150, 151, 152, 153, 154,
       155, 156, 157, 158, 159, 160, 161, 162, 163, 164, 165, 166, 167,
       168, 169, 170, 171, 172, 173, 174, 175, 176, 177, 178, 179, 180,
       185, 186, 187, 188, 189, 190, 191, 192, 193, 194, 196, 197, 198,
       199, 200, 201, 202, 203, 204, 205, 207, 208, 209, 210, 211, 212,
       213, 214, 215, 216, 218, 219, 220, 221, 222, 223, 224, 225, 226,
       227, 228, 231, 240, 241, 242, 243, 244, 245, 246, 247, 248, 249,
       253, 254, 261, 262, 263, 264, 265, 266, 267, 268, 269])
```

특정 두 행의 값이 95% 이상 일치한다면, 다중공선성 문제가 발생
이를 통해 제거된 행은 149개

(88 + 149)개의 변수를 제거한 84개의 변수를 이용하여 데이터 분석 진행

데이터 전처리

모든 데이터 값이 0인 Column 제거

```
np.array(del_list)
```

```
array([128, 129, 130, 131, 132, 133, 134, 135, 136, 137, 138, 139, 140,
       141, 142, 143, 144, 145, 146, 147, 148, 149, 150, 151, 152, 153,
       154, 155, 156, 157, 158, 159, 160, 161, 162, 163, 164, 165, 166,
       167, 168, 169, 170, 171, 172, 173, 174, 175, 176, 177, 178, 179,
       180, 181, 182, 183, 184, 185, 186, 125, 126, 127])
```

모든 값이 0이 되는 행은 변수로써 의미가 없음
이를 통해 제거된 행은 62개

데이터 전처리

Train / Test set 나누기

```
lcd2.shape, x_train1.shape, x_test1.shape
```

```
((1092919, 64), (819689, 64), (273230, 64))
```

Train / Test set의 비율을 8:2로 하여 진행

데이터 분석 과정

변수 설정

```
cor_dep = lcd2.corr()['depvar']
cor_dep
```

```
funded_amnt_inv                 0.024779
int_rate                        0.223721
installment                     0.026392
annual_inc                     -0.034461
dti                             0.085891
                                   ...
mths_since_recent_inq8         -0.010725
mths_since_recent_inq9         -0.015108
mths_since_recent_inq10        -0.025423
mths_since_recent_revol_delinq -0.006205
mths_since_recent_revol_delinq1 0.002052
```

종속변수인 'depvar'에 대한 correlation 행렬을 구하고 correlation의 절댓값이 큰 변수를 순서대로 모형에 대입

이 때 초기 모형은 LPM, Logit, Probit 세 개의 모형을 선택하고 변수를 추가하면서 다음을 측정

1) 각 모형에 대한 Accuracy의 최댓값

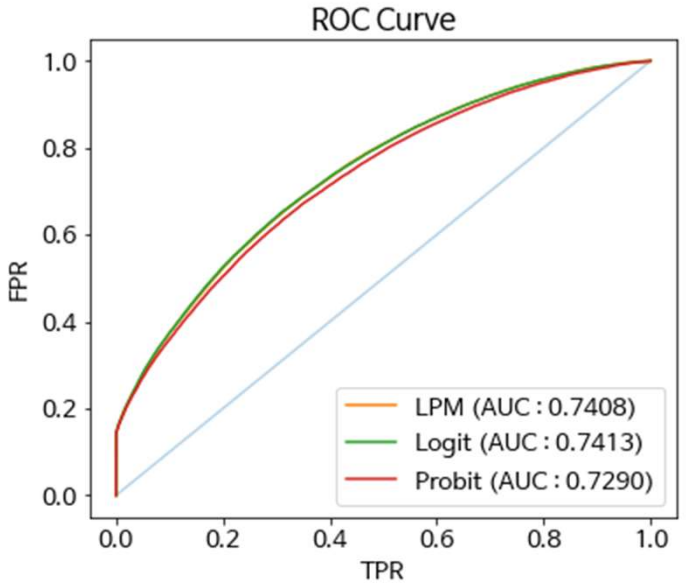
2) 각 모형이 걸리는 시간

```
def corr_col_list(df, n):

    # 주어진 'depvar'에 대한 correlation 벡터를 절댓값 크기가 큰 순서대로 n개의 행 정렬

    abs_desc_cor = sorted(df, key=abs, reverse=True)

    col_list = ['depvar']
    for i in range(1, n):
        col_list.append(cor_dep[cor_dep == abs_desc_cor[i]].index.values[0])

    return col_list
```

ROC Curve

ROC Curve

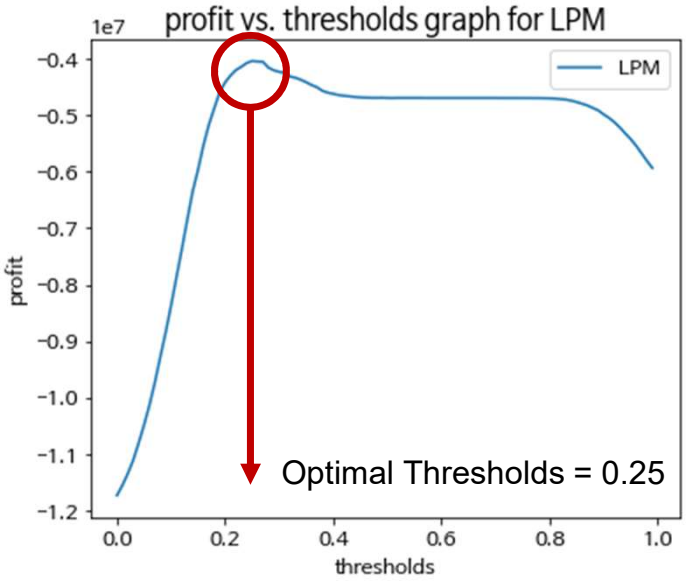
```python
# 각 모형(LPM, Logit, Probit)에 대한 ROC Curve와 각각의 AUC

plt.rcParams['figure.figsize'] = (6, 5)
plt.rcParams['font.size'] = 14

fpr1, tpr1, thresholds1 = roc_curve(y_test, pred_m1)
fpr2, tpr2, thresholds2 = roc_curve(y_test, pred_m2)
fpr3, tpr3, thresholds3 = roc_curve(y_test, pred_m3)
plt.plot([0,1], [0,1], alpha=0.3)
plt.plot(fpr1, tpr1, label='LPM (AUC : {0:3f})'.format(roc_auc_score(y_test, pred_m1)))
plt.plot(fpr2, tpr2, label='Logit (AUC : {0:3f})'.format(roc_auc_score(y_test, pred_m2)))
plt.plot(fpr3, tpr3, label='Probit (AUC : {0:3f})'.format(roc_auc_score(y_test, pred_m3)))

plt.xlabel('TPR')
plt.ylabel('FPR')
plt.title('ROC Curve')
plt.legend()

plt.show()
```



Optimal Thresholds

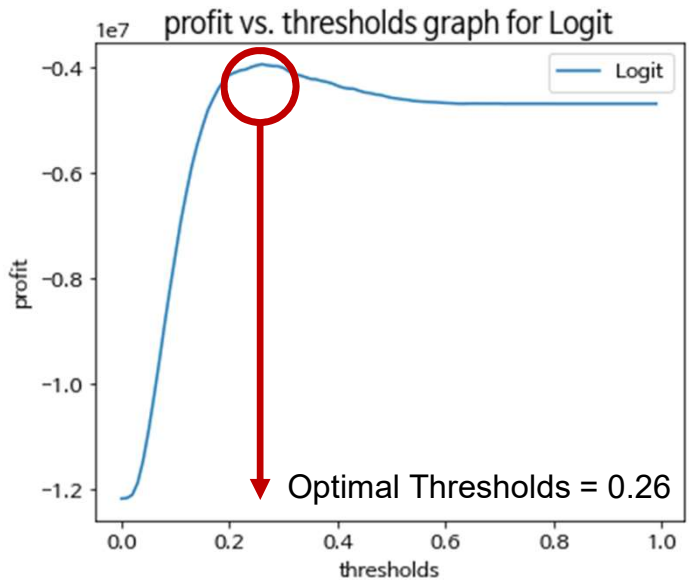
Profit vs. Thresholds graphs for LPM

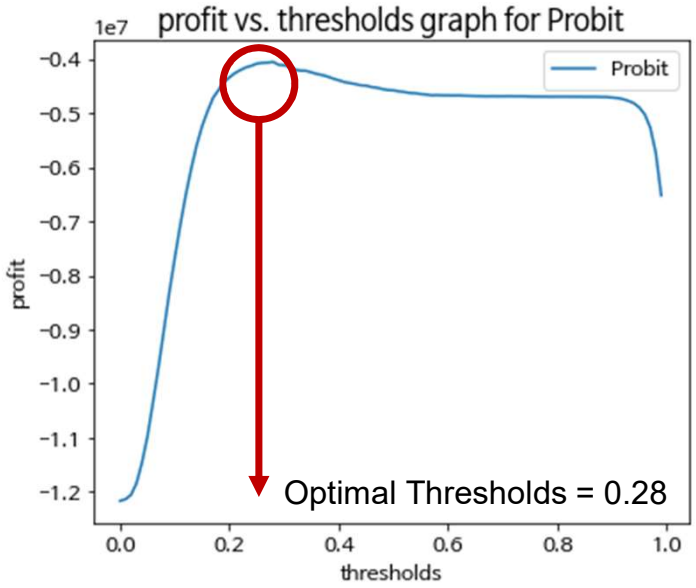
```
x_xx1 = []
y_yy1 = []
for i in np.linspace(0, 0.99, 100):
    x_xx1.append(i)
    check = [1 if j >= i else 0 for j in pred_m1]
    total_loss1 = 0
    for j in range(len(pred_m1)):
        if check[j] == 0 and y_test.iloc[j] == 1:
            total_loss1 -= x_test['installment'].iloc[j]
        if check[j] == 0 and y_test.iloc[j] == 0:
            total_loss1 += x_test['installment'].iloc[j] * x_test['int_rate'].iloc[j]
        if check[j] == 1 and y_test.iloc[j] == 0:
            total_loss1 -= x_test['installment'].iloc[j] * x_test['int_rate'].iloc[j]
    y_yy1.append(total_loss1)
    print(i, total_loss1)
plt.plot(x_xx1, y_yy1)
```



Optimal Thresholds

Profit vs. Thresholds graphs for Logit, Probit





데이터 분석 과정

변수 설정

```
def max_acc_rate(real_m, pred_m):

    # thresolds인 p의 값을 0부터 1까지 4등분하여 각각의 값에 대하여 accuracy를 계산
    # 다시 주어진 값이 포함된 구간을 4등분하여 각각의 값에 대하여 accuracy를 계산하고
    # accuracy가 최댓값을 갖도록 10번 반복
    # 이 때 가장 큰 accuracy 값과 그 때의 thresholds인 p 값을 반환

    max = 0
    max_thre = 0
    a = 0
    b = 1
    rep_num = 10

    for i in range(rep_num):
        for i in np.linspace(a, b, 5):
            if acc_rate(real_m, pred_m, i) > max:
                max = acc_rate(real_m, pred_m, i)
                max_thre = i
        if max_thre <= a + (b-a)/4:
            a = a
            b = a + (b-a)/2
        elif max_thre == a + (b-a)/2:
            a = a + (b-a)/4
            b = a + 3*(b-a)/4
        else:
            a = a + (b-a)/2
            b = b

    return (max, max_thre)
```

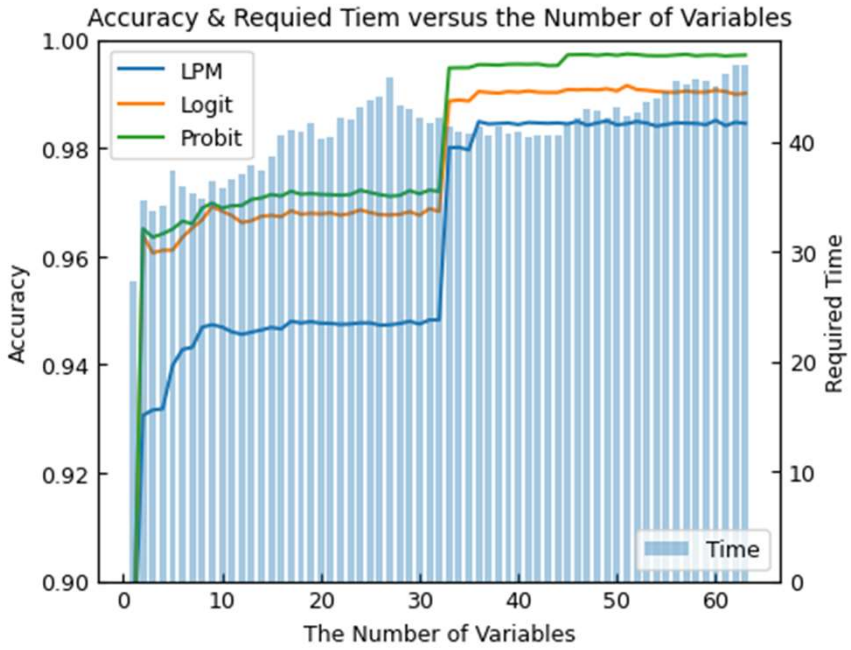
Accuracy를 최대로 하는 threshold 탐색

일반적으로 threshold인 p = 0.5 를 기준으로 0/1 범주를 나누지만, 최적점을 찾기 위해 다음과 같은 알고리즘을 이용

1) 구간 [0,1]을 최초 4등분하여 최적의 p를 p_1
2) 이후에 업데이트된 구간을 또 다시 4등분하여 최적의 p를 p_2
3) 이와 같은 방법을 10번 반복하여 최대의 threshold 값을 채택

데이터 분석 과정

변수 개수에 따른 모형 별 Accuracy & Time



시간에 대해서는 약간의 오차는 있지만 변수의 개수에 비례함을 확인

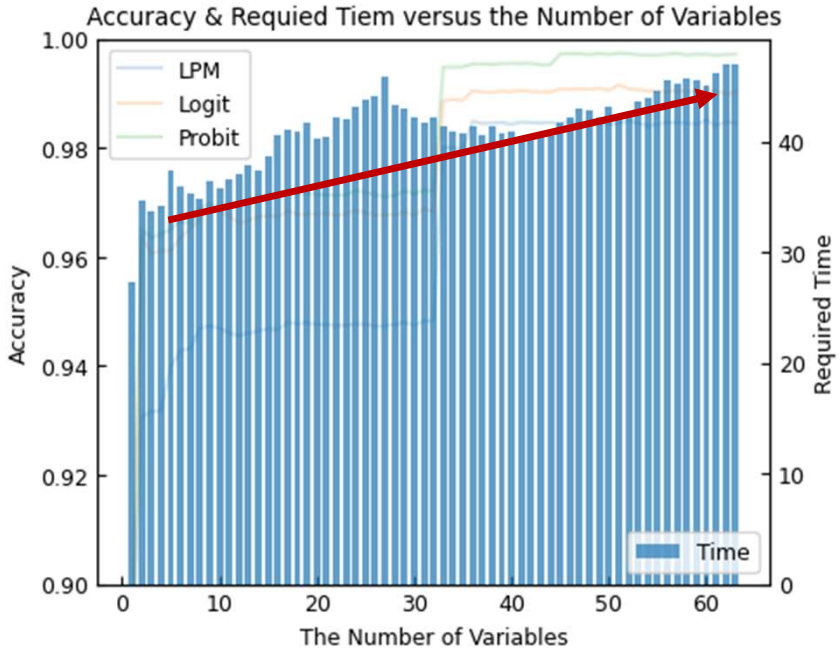
-> 효율성을 위해서는 Accuracy에 영향을 미치는 주요한 변수들에 대한 취사선택이 필요

몇 가지 변수에 대해 Accuracy 증가율이 유의미하게 높아짐을 확인

-> Accuracy에 영향을 유의미하게 주는 변수를 찾기 위하여 위의 과정을 반복하여 Accuracy 증가율이 높은 변수들을 순서대로 구함

데이터 분석 과정

변수 개수에 따른 모형 별 Accuracy & Time



시간에 대해서는 약간의 오차는 있지만 변수의 개수에 비례함을 확인

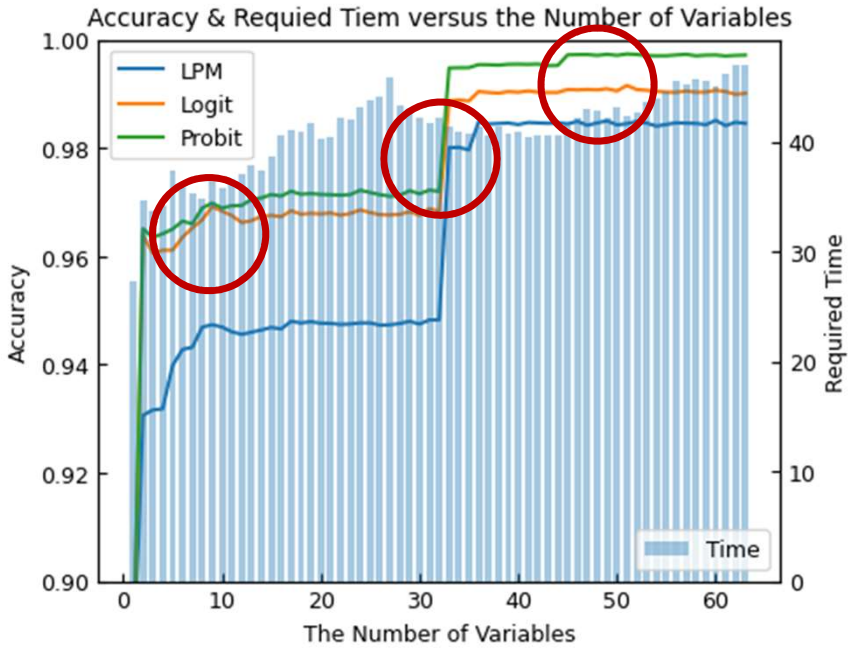
-> 효율성을 위해서는 Accuracy에 영향을 미치는 주요한 변수들에 대한 취사선택이 필요

몇 가지 변수에 대해 Accuracy 증가율이 유의미하게 높아짐을 확인

-> Accuracy에 영향을 유의미하게 주는 변수를 찾기 위하여 위의 과정을 반복하여 Accuracy 증가율이 높은 변수들을 순서대로 구함

데이터 분석 과정

변수 개수에 따른 모형 별 Accuracy & Time



시간에 대해서는 약간의 오차는 있지만 변수의 개수에 비례함을 확인

-> 효율성을 위해서는 Accuracy에 영향을 미치는 주요한 변수들에 대한 취사선택이 필요

몇 가지 변수에 대해 Accuracy 증가율이 유의미하게 높아짐을 확인

-> Accuracy에 영향을 유의미하게 주는 변수를 찾기 위하여 위의 과정을 반복하여 Accuracy 증가율이 높은 변수들을 순서대로 구함

데이터 분석 과정

Accuracy에 critical한 변수

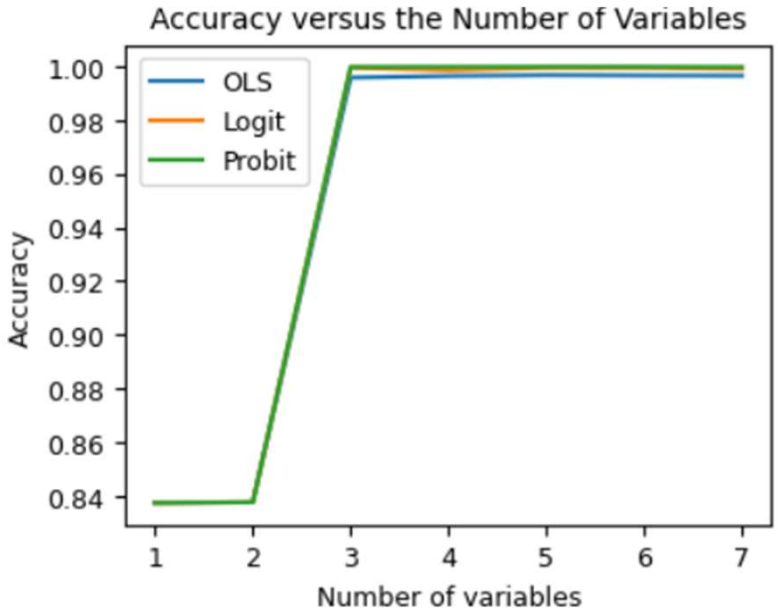
```
# accuracy를 가장 많이 높히는 variable을 7개를 순서대로 구함

var_list_num = []
for i in range(1, 6):
    var_list_num.append(ols4[i][1]+2)
    var_list_num.append(logit4[i][1]+2)
    var_list_num.append(probit4[i][1]+2)
var_list_num = list(set(var_list_num))
var_list_num
```

```
[33, 36, 5, 6, 8, 9, 45]
```

위의 과정을 통해 구한
['Installment', 'funded_amnt_inv', 'total_rec_prncp', 'total_pymnt'] …
와 같은 변수들을 차례대로 대입하며
"변수 개수에 따른 모형 별 Accuracy 그래프"를 구함

데이터 분석 과정

Accuracy에 critical한 변수



변수가 3개 이상인 경우 유의미하게 Accuracy를 높이지 못한다고 판단

-> Accuracy에 가장 critical하게 영향을 주는 변수 3개를
['Installment', 'funded_amnt_inv', 'total_rec_prncp']
와 같이 선택하여 각 모형에 대해 데이터 분석을 반복함

데이터 분석 과정

변수 설명

Installment	대출이 시작된 경우 차용자가 지불해야 하는 월별 지불액
Funded_amnt_inv	해당 시점에 투자자가 해당 대출을 위해 약정한 총 금액
Total_rec_prncp	현재까지 받은 원금

```
# 위에서 구한 variable 중 3개를 선택하여 각각 OLS, Logit, Probit 모형에 적용함

op_col_list = ['depvar', 'installment', 'funded_amnt_inv', 'total_rec_prncp']
lcd4 = lcd2[op_col_list]

predictor = lcd4.depvar
variables = lcd4.drop('depvar', axis=1)
x_train, x_test, y_train, y_test = train_test_split(variables, predictor)
```

LPM

1. LPM

```
result1 = sm.OLS(y_train, sm.add_constant(x_train)).fit(disp=False)
pred_m1 = result1.predict(sm.add_constant(x_test))
print("thresholds : %0.6f 일때 accuracy : %0.6f 로 최대값을 가짐"
      % (max_acc_rate(y_test, pred_m1)[1], max_acc_rate(y_test, pred_m1)[0]))
result1.summary()
```

```
thresholds : 0.152466 일때 accuracy : 0.996977 로 최대값을 가짐
```

OLS Regression Results

Dep. Variable:	depvar	**R-squared:**	0.609
Model:	OLS	**Adj. R-squared:**	0.609
Method:	Least Squares	**F-statistic:**	4.261e+05
Date:	Mon, 18 Jan 2021	**Prob (F-statistic):**	0.00
Time:	22:46:44	**Log-Likelihood:**	41112.
No. Observations:	819689	**AIC:**	-8.222e+04
Df Residuals:	819685	**BIC:**	-8.217e+04
Df Model:	3		
Covariance Type:	nonrobust		

	coef	std err	t	P>|t|	[0.025	0.975]
const	0.1584	0.000	325.241	0.000	0.157	0.159
installment	9.724e-05	3.95e-06	24.620	0.000	8.95e-05	0.000
funded_amnt_inv	5.933e-05	1.35e-07	439.698	0.000	5.91e-05	5.96e-05
total_rec_prncp	-6.94e-05	6.14e-08	-1129.990	0.000	-6.95e-05	-6.93e-05

Omnibus:	268371.385	**Durbin-Watson:**	1.999
Prob(Omnibus):	0.000	**Jarque-Bera (JB):**	1294176.298
Skew:	1.524	**Prob(JB):**	0.00
Kurtosis:	8.349	**Cond. No.**	4.15e+04

LPM

1. LPM

```
result1 = sm.OLS(y_train, sm.add_constant(x_train)).fit(disp=False)
pred_m1 = result1.predict(sm.add_constant(x_test))
print("thresholds : %0.6f 일때 accuracy : %0.6f 로 최대값을 가짐"
      % (max_acc_rate(y_test, pred_m1)[1], max_acc_rate(y_test, pred_m1)[0]))
result1.summary()
```

thresholds : 0.152466 일때 accuracy : 0.996977 로 최대값을 가짐

OLS Regression Results

Dep. Variable:	depvar	R-squared:	0.609
Model:	OLS	Adj. R-squared:	0.609
Method:	Least Squares	F-statistic:	4.261e+05
Date:	Mon, 18 Jan 2021	Prob (F-statistic):	0.00
Time:	22:46:44	Log-Likelihood:	41112.
No. Observations:	819689	AIC:	-8.222e+04
Df Residuals:	819685	BIC:	-8.217e+04
Df Model:	3		
Covariance Type:	nonrobust		

	coef	std err	t	P>|t|	[0.025	0.975]
const	0.1584	0.000	325.241	0.000	0.157	0.159
installment	9.724e-05	3.95e-06	24.620	0.000	8.95e-05	0.000
funded_amnt_inv	5.933e-05	1.35e-07	439.698	0.000	5.91e-05	5.96e-05
total_rec_prncp	-6.94e-05	6.14e-08	-1129.990	0.000	-6.95e-05	-6.93e-05

Omnibus:	268371.385	Durbin-Watson:	1.999
Prob(Omnibus):	0.000	Jarque-Bera (JB):	1294176.298
Skew:	1.524	Prob(JB):	0.00
Kurtosis:	8.349	Cond. No.	4.15e+04

Logistic

2. Logistic Model

```
result2 = sm.Logit(y_train, sm.add_constant(x_train)).fit(disp=False)
pred_m2 = result2.predict(sm.add_constant(x_test))
print("thresholds : %0.6f 일때 accuracy : %0.6f 로 최대값을 가짐"
      % (max_acc_rate(y_test, pred_m2)[1], max_acc_rate(y_test, pred_m2)[0]))
result2.summary()
```

```
thresholds : 0.250000 일때 accuracy : 0.999945 로 최대값을 가짐
```

Logit Regression Results

Dep. Variable:	depvar	**No. Observations:**	819689
Model:	Logit	**Df Residuals:**	819685
Method:	MLE	**Df Model:**	3
Date:	Mon, 18 Jan 2021	**Pseudo R-squ.:**	0.9985
Time:	22:46:57	**Log-Likelihood:**	-526.20
converged:	True	**LL-Null:**	-3.6270e+05
Covariance Type:	nonrobust	**LLR p-value:**	0.000

	coef	std err	z	P>|z|	[0.025	0.975]
const	-8.9077	0.254	-35.096	0.000	-9.405	-8.410
installment	0.0064	0.004	1.729	0.084	-0.001	0.014
funded_amnt_inv	1.0619	0.100	10.605	0.000	0.866	1.258
total_rec_prncp	-1.0622	0.100	-10.606	0.000	-1.258	-0.866

Logistic

2. Logistic Model

```
result2 = sm.Logit(y_train, sm.add_constant(x_train)).fit(disp=False)
pred_m2 = result2.predict(sm.add_constant(x_test))
print("thresholds : %0.6f 일때 accuracy : %0.6f 로 최대값을 가짐"
      % (max_acc_rate(y_test, pred_m2)[1], max_acc_rate(y_test, pred_m2)[0]))
result2.summary()
```

thresholds : 0.250000 일때 accuracy : 0.999945 로 최대값을 가짐

Logit Regression Results

Dep. Variable:	depvar	No. Observations:	819689
Model:	Logit	Df Residuals:	819685
Method:	MLE	Df Model:	3
Date:	Mon, 18 Jan 2021	Pseudo R-squ.:	0.9985
Time:	22:46:57	Log-Likelihood:	-526.20
converged:	True	LL-Null:	-3.6270e+05
Covariance Type:	nonrobust	LLR p-value:	0.000

	coef	std err	z	P>|z|	[0.025	0.975]
const	-8.9077	0.254	-35.096	0.000	-9.405	-8.410
installment	0.0064	0.004	1.729	0.084	-0.001	0.014
funded_amnt_inv	1.0619	0.100	10.605	0.000	0.866	1.258
total_rec_prncp	-1.0622	0.100	-10.606	0.000	-1.258	-0.866

Probit

3. Probit Model

```
result3 = Probit(y_train, sm.add_constant(x_train)).fit(method='bfgs', disp=False)
pred_m3 = result3.predict(sm.add_constant(x_test))
print("thresholds : %0.6f 일때 accuracy : %0.6f 로 최대값을 가짐"
      % (max_acc_rate(y_test, pred_m3)[1], max_acc_rate(y_test, pred_m3)[0]))
result3.summary()
```

```
thresholds : 0.007812 일때 accuracy : 0.999876 로 최대값을 가짐
```

Probit Regression Results

Dep. Variable:	depvar	**No. Observations:**	819689
Model:	Probit	**Df Residuals:**	819685
Method:	MLE	**Df Model:**	3
Date:	Mon, 18 Jan 2021	**Pseudo R-squ.:**	0.9950
Time:	22:47:14	**Log-Likelihood:**	-1797.2
converged:	False	**LL-Null:**	-3.6270e+05
Covariance Type:	nonrobust	**LLR p-value:**	0.000

	coef	std err	z	P>|z|	[0.025	0.975]
const	-2.4368	0.032	-76.299	0.000	-2.499	-2.374
installment	-0.0067	0.001	-10.942	0.000	-0.008	-0.006
funded_amnt_inv	0.0168	0.000	62.418	0.000	0.016	0.017
total_rec_prncp	-0.0167	0.000	-62.899	0.000	-0.017	-0.016

Probit

3. Probit Model

```
result3 = Probit(y_train, sm.add_constant(x_train)).fit(method='bfgs', disp=False)
pred_m3 = result3.predict(sm.add_constant(x_test))
print("thresholds : %0.6f 일때 accuracy : %0.6f 로 최대값을 가짐"
      % (max_acc_rate(y_test, pred_m3)[1], max_acc_rate(y_test, pred_m3)[0]))
result3.summary()
```

thresholds : 0.007812 일때 accuracy : 0.999876 로 최대값을 가짐

Probit Regression Results

Dep. Variable:	depvar	No. Observations:	819689
Model:	Probit	Df Residuals:	819685
Method:	MLE	Df Model:	3
Date:	Mon, 18 Jan 2021	Pseudo R-squ.:	0.9950
Time:	22:47:14	Log-Likelihood:	-1797.2
converged:	False	LL-Null:	-3.6270e+05
Covariance Type:	nonrobust	LLR p-value:	0.000

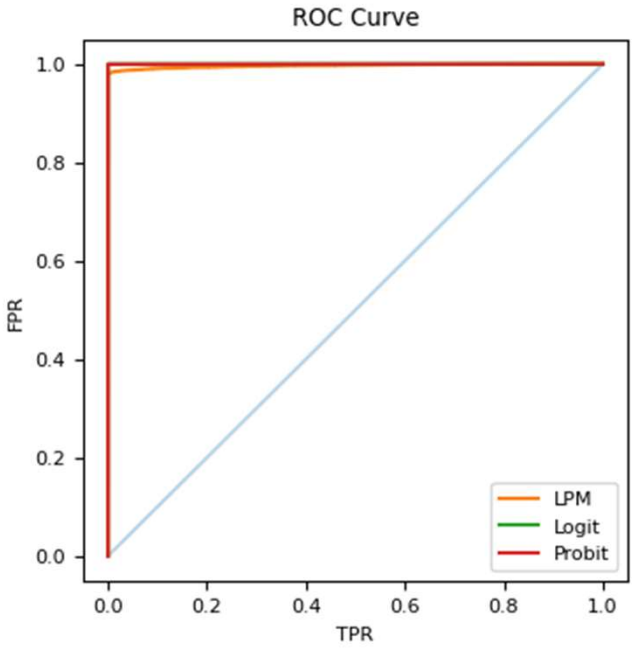
	coef	std err	z	P>|z|	[0.025	0.975]
const	-2.4368	0.032	-76.299	0.000	-2.499	-2.374
installment	-0.0067	0.001	-10.942	0.000	-0.008	-0.006
funded_amnt_inv	0.0168	0.000	62.418	0.000	0.016	0.017
total_rec_prncp	-0.0167	0.000	-62.899	0.000	-0.017	-0.016

ROC Curve

ROC Curve

```
# 각 모형(OLS, Logit, Probit)에 대한 ROC Curve와 각각의 AUC

plt.rcParams['figure.figsize'] = (4, 4)
plt.rcParams['font.size'] = 8

fpr1, tpr1, thresholds1 = roc_curve(y_test, pred_m1)
fpr2, tpr2, thresholds2 = roc_curve(y_test, pred_m2)
fpr3, tpr3, thresholds3 = roc_curve(y_test, pred_m3)
plt.plot([0,1], [0,1], alpha=0.3)
plt.plot(fpr1, tpr1, label='LPM')
plt.plot(fpr2, tpr2, label='Logit')
plt.plot(fpr3, tpr3, label='Probit')
print("LPM 모형의 AUC 최댓값 : %0.6f, 이 때의 thresolds : %0.5f, Accuracy : %0.6f"
      % (roc_auc_score(y_test, pred_m1), max_ftt(fpr1, tpr1, thresholds1)[2], acc_rate(y_test, pred_m1, max_ftt(fpr1, tpr1, thresholds1)[2]
print("Logit 모형의 AUC 최댓값 : %0.6f, 이 때의 thresolds : %0.5f, Accuracy : %0.6f"
      % (roc_auc_score(y_test, pred_m2), max_ftt(fpr2, tpr2, thresholds2)[2], acc_rate(y_test, pred_m2, max_ftt(fpr2, tpr2, thresholds2)[2]
print("Probit 모형의 AUC 최댓값 : %0.6f, 이 때의 thresolds : %0.5f, Accuracy : %0.6f"
      % (roc_auc_score(y_test, pred_m2), max_ftt(fpr3, tpr3, thresholds3)[2], acc_rate(y_test, pred_m3, max_ftt(fpr3, tpr3, thresholds3)[2]

plt.xlabel('TPR')
plt.ylabel('FPR')
plt.title('ROC Curve')
plt.legend()

plt.show()
```

```
LPM 모형의 AUC 최댓값 : 0.996372, 이 때의 thresolds : 0.15213, Accuracy : 0.996959
Logit 모형의 AUC 최댓값 : 0.999933, 이 때의 thresolds : 0.00050, Accuracy : 0.999960
Probit 모형의 AUC 최댓값 : 0.999933, 이 때의 thresolds : 0.00815, Accuracy : 0.999872
```



SVM

알고리즘 과정

> 데이터의 variable 중 3개를 선택
: 'depvar ' 가 0일 때와 1일 때를 구분하여, [' installment ' , ' funded_amnt_inv ' , ' total_rec_prncp']데이터를 수집

> SVM 모델로 fitting 진행
: 데이터를 depvar의 value에 따라 라벨링하여 SVM 모델에 fitting, 이후 예측 함수를 사용해 accuracy_score 측정

하지만, SVM fitting 단계에서 output이 나오지 않고 계속 run되는 문제가 발생하여 accuracy를 확인할 수 없었음

code

SVM 함수

```
from sklearn import svm
from sklearn.svm import SVC
model = svm.SVC(kernel='linear')
model.fit(result,Y)
```

Predict 함수

```
predict1 = model.predict(xpca)
predict2 = model.predict(xpc)
import sklearn
print(sklearn.metrics.accuracy_score(y1,predict1,normalize=True,sample_weight=None))
print(sklearn.metrics.accuracy_score(y2,predict2,normalize=True,sample_weight=None))
```

4. Results & Conclusion

Results & Conclusion

Conclusion

	LPM	Logit	Probit
AUC	0.7408	0.7413	0.7290
Optimal Thresholds	0.25	0.26	0.28
Accuracy	0.8252	0.8271	0.8350

- **AUC :** Probit < LPM < Logit
- **Optimal Thresholds에서 Accuracy**
  : LPM < Logit < Probit

AUC와 Accuracy를 모두 고려할 때
Logit 모형 선택

Results & Conclusion

결과

- **Logit 모형 선택**
  Accuracy가 가장 높은 모형을 선택

- **Accuracy/ AUC 측정**
  Threshold = 0.0005, 이 때의 Accuracy = 0.99996, AUC = 0.999933

- **Variables 선택**
  시간 효율을 위해 'installment','funded_amnt_inv','total_rec_prncp'의 3개의 변수 선택

$$Pr[y=1|x] = \frac{exp(-8.907683+0.006370x_{installment}+1.061942x_{funded\_amnt\_inv}-1.062195x_{total\_rec\_prncp})}{1+exp(-8.907683+0.006370x_{installment}+1.061942x_{funded\_amnt\_inv}-1.062195x_{total\_rec\_prncp})} \geq 0.0005$$

Results & Conclusion

Discussion

$$Pr[y=1|x] = \frac{exp(-8.907683+0.006370x_{installment}+1.061942x_{funded\_amnt\_inv}-1.062195x_{total\_rec\_prncp})}{1+exp(-8.907683+0.006370x_{installment}+1.061942x_{funded\_amnt\_inv}-1.062195x_{total\_rec\_prncp})} \geq 0.0005$$

Installment

Funded_amnt_inv

Total_rec_prncp

Results & Conclusion

Discussion

```
lcd[['depvar', 'installment','funded_amnt_inv' ,'total_rec_prncp']].head()
```

	depvar	installment	funded_amnt_inv	total_rec_prncp
0	0.0	605.62	19000.0	19000.00
1	0.0	312.86	10000.0	10000.00
2	0.0	197.78	6000.0	6000.00
3	1.0	836.89	25200.0	12882.27
4	0.0	269.52	8000.0	8000.00

- **각 변수의 단위 크기가 다름**

  'installment'의 경우 scale이 10^2

  'funded_amnt_inv'와'total_rec_prncp'의 경우 scale이 10^4

Results & Conclusion

Discussion

lcd.installment.describe()

count	1.092919e+06
mean	4.312534e+02
std	2.652486e+02
min	4.930000e+00
25%	2.369800e+02
50%	3.627800e+02
75%	5.732100e+02
max	1.715420e+03

lcd.funded_amnt_inv.describe()

count	1.092919e+06
mean	1.364033e+04
std	8.438033e+03
min	7.750000e+02
25%	7.100000e+03
50%	1.195000e+04
75%	1.900000e+04
max	4.000000e+04

lcd.total_rec_prncp.describe()

count	1.092919e+06
mean	1.221735e+04
std	8.515152e+03
min	0.000000e+00
25%	5.780600e+03
50%	1.000000e+04
75%	1.692500e+04
max	4.000000e+04

Results & Conclusion

Discussion

$$Pr[y=1|x] = \frac{exp(-8.907683+0.006370x_{installment}+1.061942x_{funded\_amnt\_inv}-1.062195x_{total\_rec\_prncp})}{1+exp(-8.907683+0.006370x_{installment}+1.061942x_{funded\_amnt\_inv}-1.062195x_{total\_rec\_prncp})} \geq 0.0005$$

Installment

Funded_amnt_inv

Total_rec_prncp

Results & Conclusion

Discussion

$$Pr[y=1|x] = \frac{exp(-8.907683+0.006370x_{installment}+1.061942x_{funded\_amnt\_inv}-1.062195x_{total\_rec\_prncp})}{1+exp(-8.907683+0.006370x_{installment}+1.061942x_{funded\_amnt\_inv}-1.062195x_{total\_rec\_prncp})} \geq 0.0005$$

- **낮은 threshold 값**
  'funded_amnt_inv'와'total_rec_prncp'의 경우 scale이 10^4
  Exponential 함수 값의 조정이 큼

감사합니다