기대 수익 최대화 모델

Lending Club Model

4조 박석훈 박태준 윤성규 임동건

Contents

1. Lending Club 소개

Lending club이란?

Lending Club은 미국 최대의 P2P 대출 플랫폼으로, 돈이 필요한 대출 수요자와 여유 자금을 굴리고 싶은 투자자를 연결해주는 회사이다.

한 명의 차입자에 대해 다수의 투자자가 매칭되고 대출기간은 36 ~ 60개월이며 한도는 4만 달러까지다.

Lending Club은 대출 신청자 중 선별하여 7단계의 신용 등급을 매겨

온라인 대출 장터에 올려놓는다. 개인 투자자들은 대출 신청자 명단을 보고 자신이 원하는 사람에게 투자한다.

Lending Club은 대출금의 1~3%를 중개 수수료로 받는다.

2. 데이터 탐색 및 목표 설정

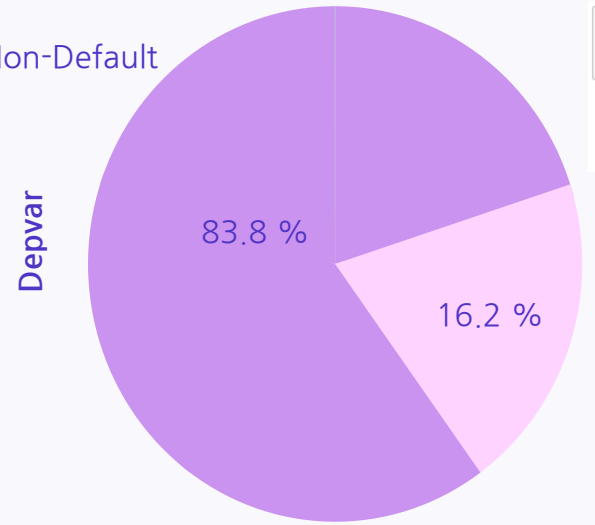
Dataset 소개

87만 개 데이터 x 334개의 특성변수로 이루어진 매우 큰 규모의 Narrow Dataset 이다.

	index	loan_amnt	funded_amnt	funded_amnt_inv	int_rate	installment	annual_inc
0	1027571	12000	12000	12000.0	0.0789	375.43	90641.0
1	46603	14000	14000	14000.0	0.1147	461.47	46000.0
2	40908	24000	24000	24000.0	0.1147	791.09	90000.0
3	182450	14000	14000	14000.0	0.0949	448.40	115000.0
4	868323	14000	14000	14000.0	0.0999	451.68	78000.0
...	...	...	...	...	...	...	...
874330	480114	32000	32000	32000.0	0.1875	825.71	90000.0
874331	1027649	24000	24000	24000.0	0.0692	740.18	76000.0
874332	773937	4000	4000	4000.0	0.0662	122.82	24960.0
874333	87618	8000	8000	8000.0	0.1288	269.09	53000.0
874334	296586	8000	8000	8000.0	0.1149	263.78	77000.0

874335 rows × 334 columns

기존 종속변수 depvar의 값이 약 8:2 비율로 나타나는 Imbalanced Data이다.




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

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

Data Features

대출 신청자의 소득, 신청금액에서부터 자가 상태까지 55종류의 서로 다른 특성변수를 포함한다.

연속형 변수
Continuous variables

loan_amnt

annual_inc

int_rate

dti

total_acc

이산형 변수
Discrete Variables

delinq_2yrs

open_acc

chargeoff_within_12_mths

tot_cur_bal

term1

명목형 변수
Categorical Variables

home_ownership

purpose

verification_status

initial_list_status

emp_length

목표 : 총 수익 극대화

```
print("수익률 :",((sum(df['total_pymnt'])/sum(df['funded_amnt']))-1)*100,"%")
print("총 받은 금액 :",sum(df['total_pymnt']))
print("총 투자한 금액 :",sum(df['funded_amnt']))
print("총 순이익:",sum(df['total_pymnt']) - sum(df['funded_amnt']))
```

```
수익률 : 9.083150808028329 %
총 받은 금액 : 13013512253.598719
총 투자한 금액 : 11929901325
총 순이익: 1083610928.5987186
```

전체 데이터에서의 수익 관련 통계량

수익률 : 약 9%

총 받은 금액 : 약 130억 달러

총 투자한 금액 : 약 119억 달러

총 순이익 : 약 11억 달러

```
print("수익률 :",((sum(df['total_pymnt'])/sum(df['funded_amnt']))-1)*100,"%")
print("총 받은 금액 :",sum(df['total_pymnt']))
print("총 투자한 금액 :",sum(df['funded_amnt']))
print("총 순이익:",sum(df['total_pymnt']) - sum(df['funded_amnt']))
```

```
수익률 : -35.5665135219272 %
총 받은 금액 : 1289520074.3908787
총 투자한 금액 : 2001319725
총 순이익: -711799650.6091213
```

부도가 난 경우의 수익 관련 통계량

수익률 : 약 -36%

총 받은 금액 : 약 13억 달러

총 투자한 금액 : 약 20억 달러

총 순이익 : 약 -7억 달러

목표 : 총 수익 극대화

```
df1 = df[df['depvar'] == 1]
df1 = df1[df1['total_pymnt'] > df1['funded_amnt']]
df1
```

	index	loan_amnt	funded_amnt	funded_amnt_inv	int_rate
46	332076	5375	5375	5375.0	0.1699
55	574374	22000	22000	22000.0	0.1144
72	191521	2700	2700	2700.0	0.1449
96	1026327	11975	11975	11975.0	0.2299
97	505231	20125	20125	20025.0	0.2363
...	...	...	...	...	...
873993	1072873	3000	3000	3000.0	0.1714
874097	10427	10000	10000	10000.0	0.0916
874151	581311	12000	12000	12000.0	0.1952
874282	74474	17000	17000	17000.0	0.1288
874300	440205	22400	22400	22400.0	0.2580

18890 rows × 334 columns

부도가 났음에도 부도 이전까지 낸 돈(total_pymnt)이 대출 금액(funded_amnt)보다 큰 경우가 왼쪽과 같이 총 18,890 건이 존재한다.

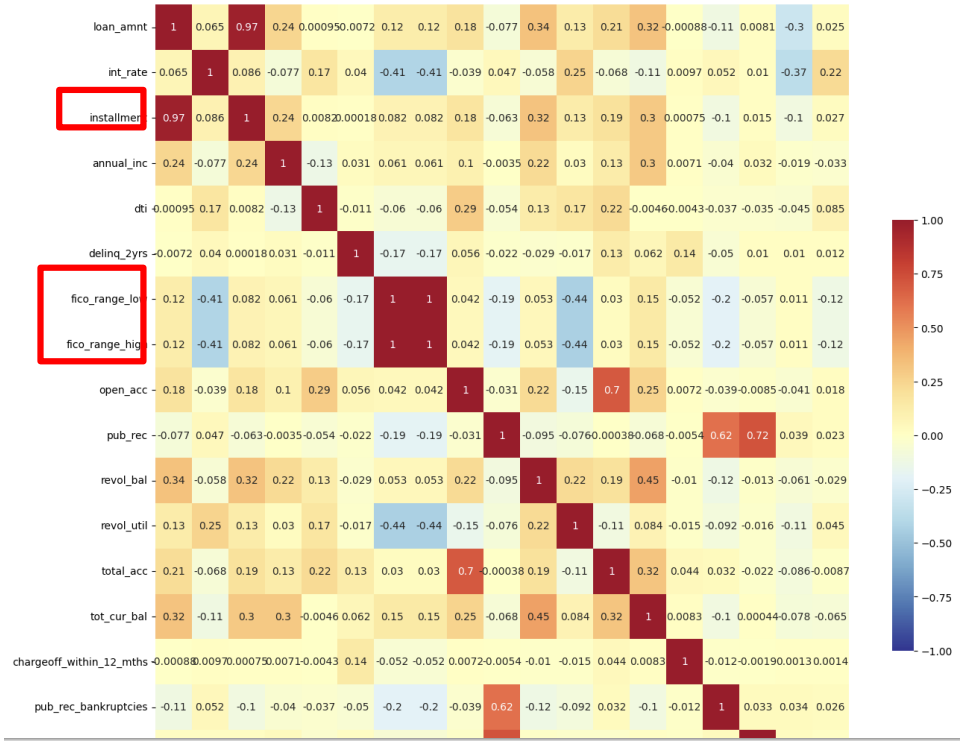
이를 고려하여 기존의 반응변수인 depvar를 사용하지 않고, 이익을 주는 고객들을 선별해 대출을 승인하는 새로운 반응변수를 정의하고, 이에 맞는 평가 척도를 세워 총 수익을 최대화 할 수 있는 모델을 만든다.

Correlation Matrix

변수 간 상관관계를 확인하기 위해 Correlation matrix를 활용했다.

상관계수가 높은 fico_range_high와 low, installment 등 중복되는 Feature들을 확인할 수 있었고,

loan_amnt와의 correlation 값이 0.97로 매우 높게 나타나는 installment를 제외했다.



3. 변수 설정과 데이터 전처리

설명변수 선정

투자자의 입장에서 최대의 기대 수익률을 예측하기 위해, 대출 승인 시점 이전에 대출자의 정보를 담은 변수만을 선택하였다. 설명변수로 사용할 변수는 (더미변수를 고려하지 않았을 때) 총 24개이다.

loan_amnt	fico_range_low	revol_bal	tot_cur_bal	verification_stauts(1~3)
int_rate	inq_last_6mnths	revol_until	chargeoff_within..	purpose(1~14)
annual_inc	open_acc	total_acc	pub_rec_bankruptcies	addr_state(1~51)
dti	pub_rec	acc_now_delinq	emp_length(1~12)	term
delinq_2yrs	delinq_2yrs	tot_coll_amt	home_ownership(1~6)	

반응변수

설명변수 선정

0 비율이 매우 높은 변수는 (영향을 주지 않는다는 결론 아래) 변수 선정에서 제외하였다. (기준 : 0.999)

```
for i in df.columns:
    print(f"{i} 변수의 0 비율 {len(df[df[i] == 0])/len(df[i])}\n")
```

```
loan_not_access 변수의 0 비율 0.859561838425775

loan_amnt 변수의 0 비율 0.0

int_rate 변수의 0 비율 0.0

annual_inc 변수의 0 비율 0.0

dti 변수의 0 비율 0.0002996563102243419

delinq_2yrs 변수의 0 비율 0.796181097634202

open_acc 변수의 0 비율 0.0

fico_range_low 변수의 0 비율 0.0

pub_rec 변수의 0 비율 0.8258241978189138

revol_bal 변수의 0 비율 0.0027941235338857532
```

```
for i in df.columns:
    if len(df[df[i] == 0])/len(df[i]) > 0.999:
        print(f"{i} 변수의 0 비율 {len(df[df[i] == 0])/len(df[i])}\n")
```

```
home_ownership1 변수의 0 비율 0.9996557383611545

home_ownership3 변수의 0 비율 0.9999599695768784

home_ownership4 변수의 0 비율 0.9999588258505036

purpose4 변수의 0 비율 0.9999988562736251

purpose11 변수의 0 비율 0.9993366387025568

purpose14 변수의 0 비율 0.9992131162540674

addr_state13 변수의 0 비율 0.9999977125472502

addr_state14 변수의 0 비율 0.9990873063528282
```

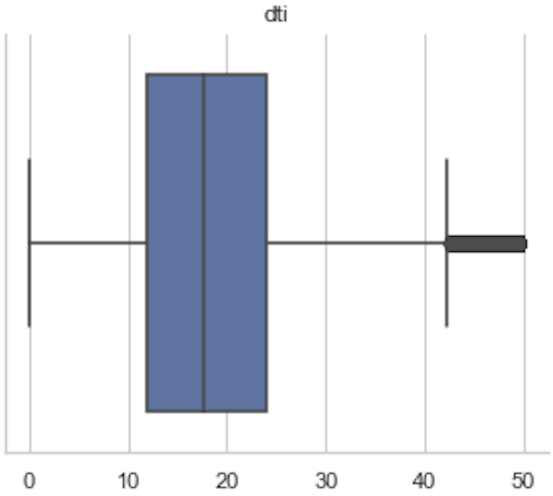
```
for i in df.columns:
    if len(df[df[i] == 0])/len(df[i]) > 0.999:
        df = df.drop(i, axis=1)
```

EDA Exploratory Data Analysis

더미 변수들은 하나의 열로 합친 후, 선별한 설명변수들에 대해 변수 종류별로 각각 boxplot, histogram, barplot 을 그려보고 분포를 확인해보았다.

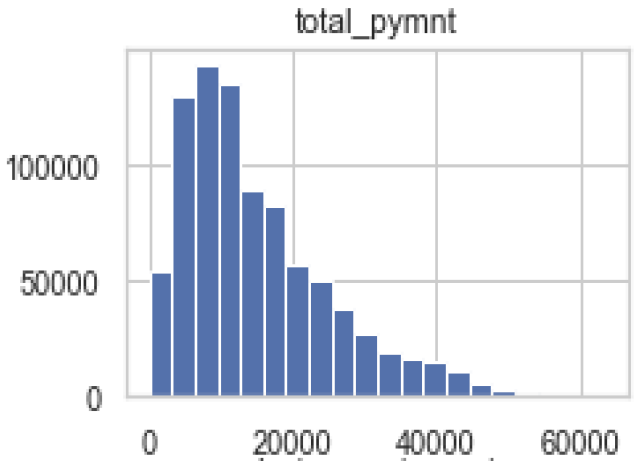
연속형 변수

Continuous variables



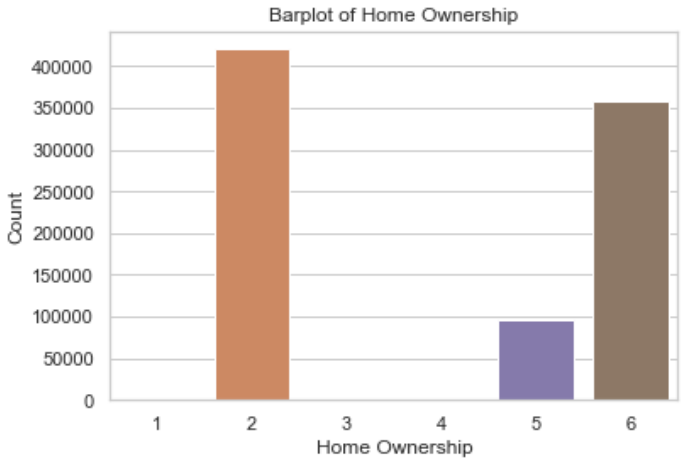
이산형 변수

Discrete Variables



명목형 변수

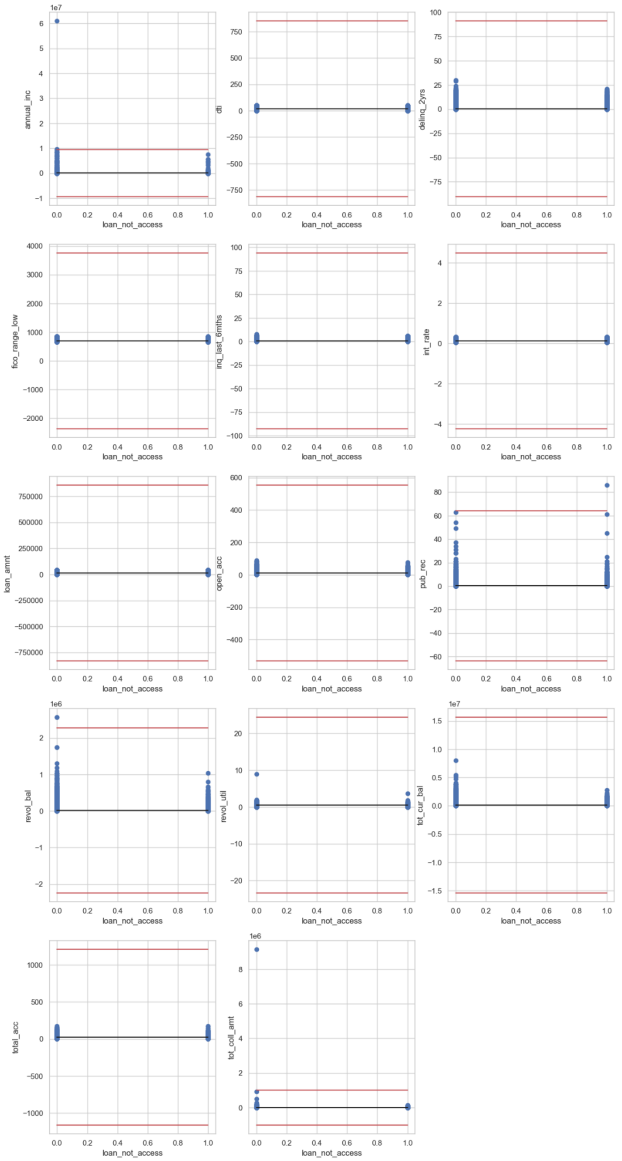
Categorical Variables



EDA Exploratory Data Analysis

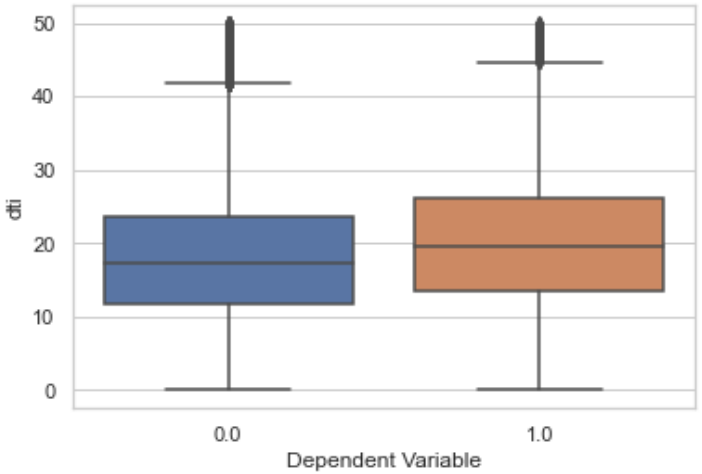
매우 큰 차이를 보이는 outlier가 존재하는 변수들이 있음을 확인할 수 있었고, 오른쪽 그림과 같이 검은선을 평균, 빨간선을 100*(sample standard deviation)으로 놓고 분포를 살펴보았다.

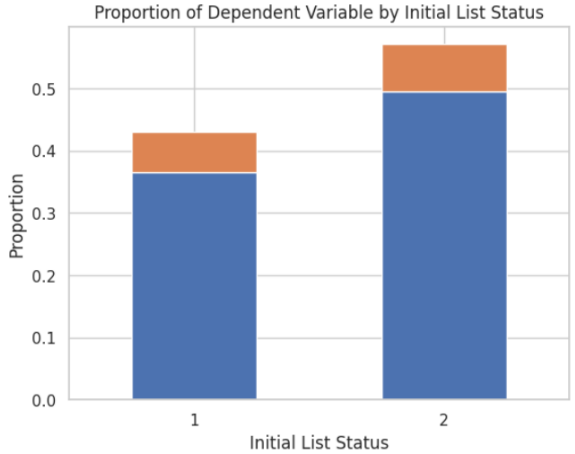
이 중 annual_inc, revol_bal, pub_rec, tot_coll_amt에서 각각 100*ssd를 벗어나는 outlier를 하나씩 찾을 수 있었고, 분석 결과에 부정적인 영향이 있을 것이라 판단해 데이터에서 제거하였다.

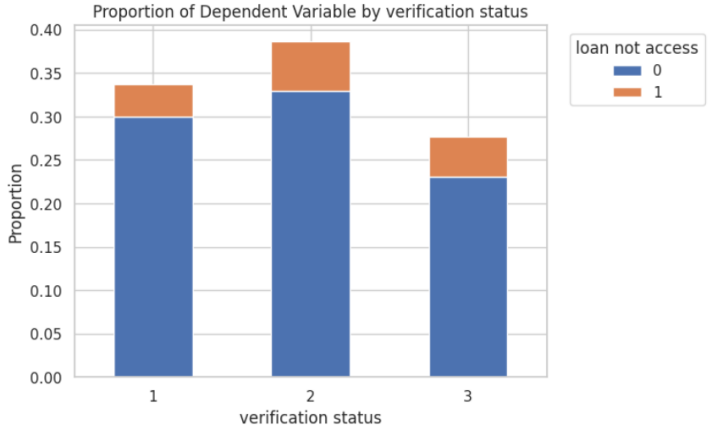


EDA Exploratory Data Analysis

설명변수별로 종속변수 depvar 값에 따른 분포를 그려보고, 그 차이를 시각적으로 살펴보았다.







설명변수와 반응변수의 독립성 검정

단순히 시각적으로 차이를 살펴보는 것이 그치지 않고, 설명변수와 독립변수 간에 독립성 검정을 시행하여

P-Value를 계산하였다. 표본이 매우 많기에 대부분 매우 유의하였다.

연속형 변수
Continuous variables

```
loan_amnt변수에 대한 가설검정 결과
T-statistic: -23.304617495743283
P-value: 5.7655922989563386e-120

annual_inc변수에 대한 가설검정 결과
T-statistic: 38.15976995291201
P-value: 0.0

int_rate변수에 대한 가설검정 결과
T-statistic: -207.67336618331794
P-value: 0.0

dti변수에 대한 가설검정 결과
T-statistic: -78.14422598656297
P-value: 0.0

tot_coll_amt변수에 대한 가설검정 결과
T-statistic: 0.4392218722313159
P-value: 0.6605008870966624
```

이산형 변수
Discrete Variables

```
pub_rec변수에 대한 가설검정 결과
T-statistic: -19.721432586250497
P-value: 1.7344027613339152e-86

revol_bal변수에 대한 가설검정 결과
T-statistic: 30.426458412776153
P-value: 6.244579597876126e-203

total_acc변수에 대한 가설검정 결과
T-statistic: 8.054003882964933
P-value: 8.056687701032075e-16

chargeoff_within_12_mths변수에 대한 가설검정 결과
T-statistic: -1.316919526320535
P-value: 0.1878671035453382

acc_now_delinq변수에 대한 가설검정 결과
T-statistic: -2.712175234000153
P-value: 0.006684920606149291
```

명목형 변수
Categorical Variables

```
home_ownership변수에 대한 가설검정 결과
F-statistic: 729.9793325539163
P-value: 0.0

verification_status변수에 대한 가설검정 결과
F-statistic: 2546.5863998060195
P-value: 0.0

purpose변수에 대한 가설검정 결과
F-statistic: 162.18333645293407
P-value: 0.0

initial_list_status변수에 대한 가설검정 결과
F-statistic: 917.2239212968465
P-value: 2.250155151651637e-201

emp_length변수에 대한 가설검정 결과
F-statistic: 161.52792789044898
P-value: 0.0
```

반응변수 정의

총 수익을 최대화하는 모델을 만들기 위해 'loan_not_access'라는 새로운 반응변수를 정의했다.

1.2.3 define response variable

```
print(len(df[df['depvar']==1]))

df['loan_not_access'] = np.where(df['funded_amnt'] > df['total_pymnt'], 1, 0)
len(df[df['loan_not_access']==1])
```

```
141683
```

```
122790
```

기존의 설명변수 'total_pymnt'(총 받은 금액) - 'funded_amnt'(대출 금액)이 0보다 클 경우

해당 고객으로부터 순이익을 얻은 경우이므로 0 값을 갖고, 0보다 작을 경우 1값을 갖는

'loan_not_access'를 만든다.

즉, 값이 0일 경우 해당 고객에게 순이익을 얻을 수 있기에 대출을 해줄 것이고, 1일 경우 순이익을 얻을 수 없기에 대출을 승인하지 않을 것임을 나타내는 변수이다.

반응변수 관련 수익 통계량

반응변수를 정확히 예측할 수 있다면, 102억 달러만을 투자해서 전체 순수익을 7억 5천만 달러 증가시킬 수 있을 것이다.

loan_not_access = 0 인 경우

수익률 : 약 18%

총 받은 금액 : 약 120억 달러

총 투자한 금액 : 약 102억 달러

총 순이익 : 약 18억 달러

loan_not_access = 1 인 경우

수익률 : 약 -44%

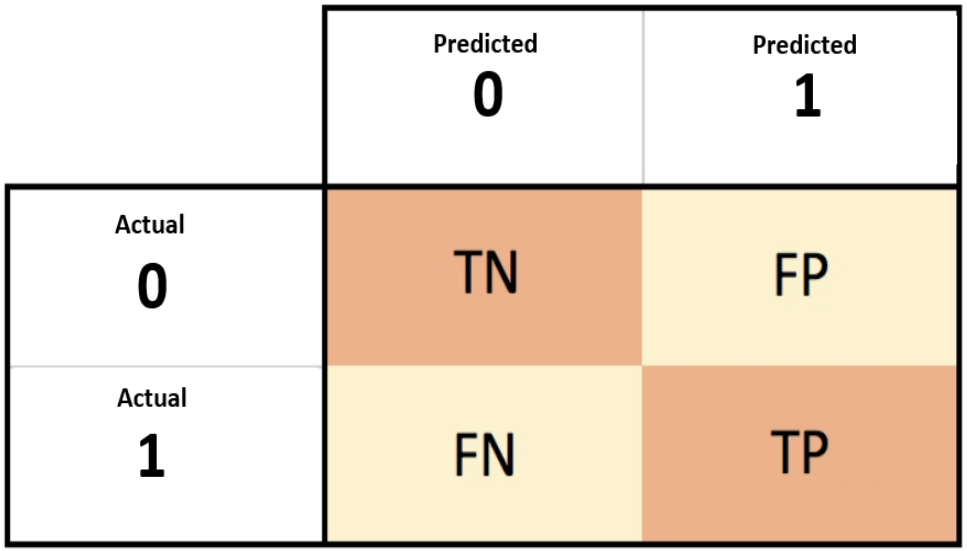
총 받은 금액 : 약 9억 7천만 달러

총 투자한 금액 : 약 17억 달러

총 순이익 : 약 -7억 5천만 달러

모델 평가 척도: Revenue_Risk

반응변수 값에 따른 전체 수익률 (0 : 18 %, 1 : -44%) 을 고려하여 다음과 같은 Measure를 만들어 모델 평가 척도로 사용한다. 동일한 test set에 대해 모델별로 크기를 비교할 수 있다.



Revenue_Risk = 0.18 x FP + 0.44 x FN

TN : 이익 + 줄 사람에게 대출 O

FP : 이익 + 줄 사람에게 대출 X

FN : 이익 – 줄 사람에게 대출 O

TP : 이익 – 줄 사람에게 대출 X

4. 모델링 및 평가

1) Logistic Regression

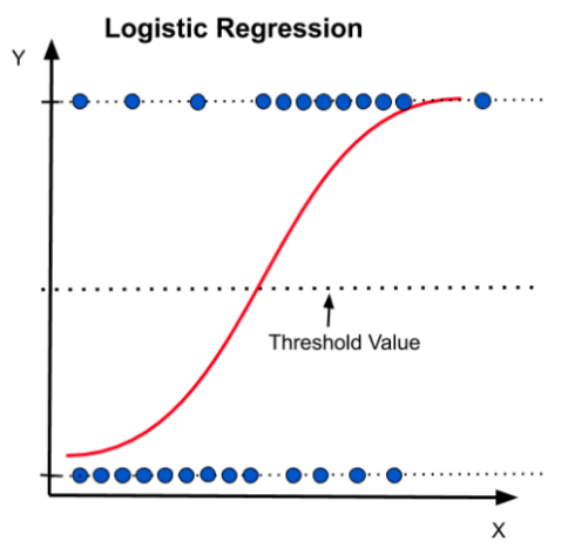
Logistic Regression

표준화한 Train set으로 학습된 Logistic Regression 모델과 Validation data에서 구한 optimal threshold를 이용해 Test set으로 검증한 결과는 다음과 같다.

데이터를 분리할 때 y=1 비율을 유지하게 나눠주었고, 비교를 쉽게 하기 위해 모든 모델에 대해 통일함.

Train set

모델 학습



Validation set

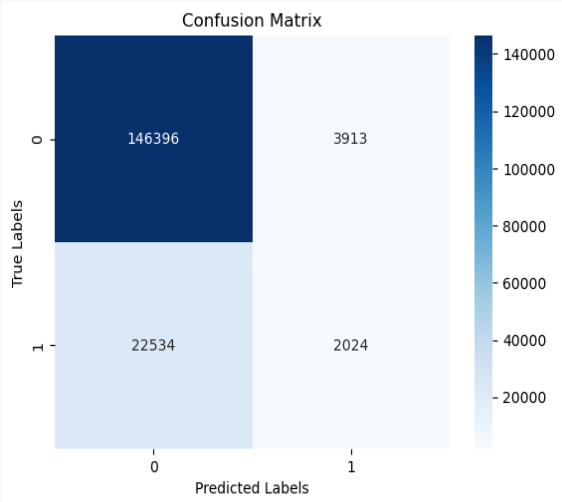
Optimal Threshold 찾기

Optimal Threshold : 0.3232

Min Revenue Risk : 0.0603

Test set

모델 평가



Logistic Regression

Logistic Regression Model 결과의 투자금, 수익금, 순수익, 수익률 지표는 다음과 같다.

	All	Test	Test → Modeling
투자금	119억 달러	23억 8천만 달러	22억 8천만 달러
수익금	130억 달러	26억 달러	24억 8천만 달러
순수익	10억 8천만 달러	2억 1천만 달러	2억 7백만 달러
수익률	9.08%	9.06%	9.10%

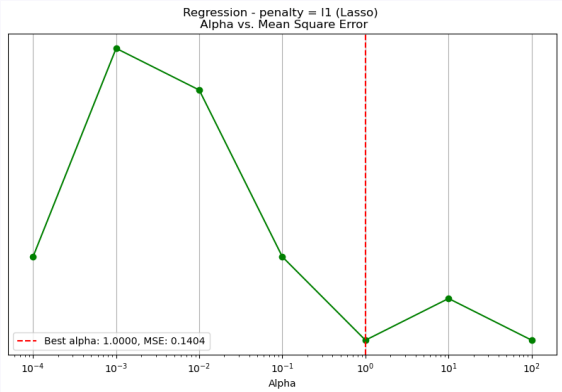
2) Logistic Regression - Lasso

Logistic Regression - Lasso

표준화한 Train set으로 학습된 Logistic Regression - Lasso 모델과 Validation data에서 구한 optimal threshold를 이용해 Test set으로 검증한 결과는 다음과 같다.

Train set

모델 학습



Best alpha : 1.0

Best MSE : 0.1404

Validation set

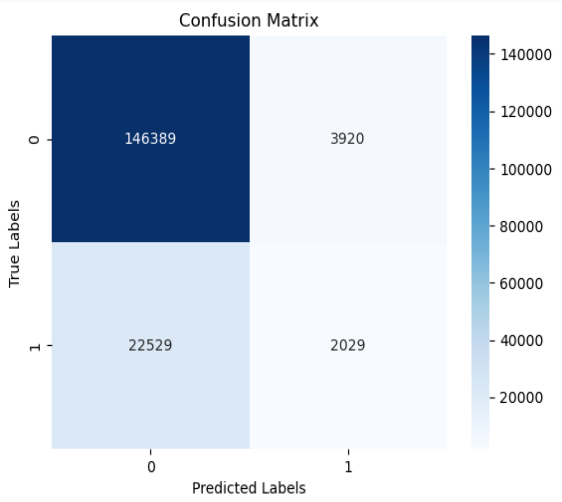
Optimal Threshold 찾기

Optimal Threshold : 0.3230

Min Revenue Risk : 0.0604

Test set

모델 평가



Logistic Regression - Lasso

Logistic Regression - Lasso Model 결과의 투자금, 수익금, 순수익, 수익률 지표는 다음과 같다.

	All	Test	Test → Modeling
투자금	119억 달러	23억 8천만 달러	22억 8천만 달러
수익금	130억 달러	26억 달러	24억 8천만 달러
순수익	10억 8천만 달러	2억 1천만 달러	2억 7백만 달러
수익률	9.08%	9.06%	9.10%

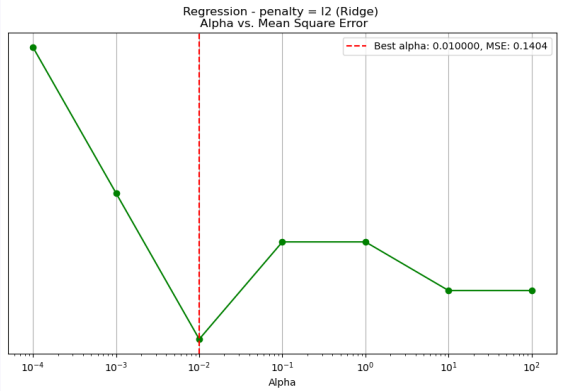
3) Logistic Regression - Ridge

Logistic Regression - Ridge

표준화한 Train set으로 학습된 Logistic Regression - Ridge 모델과 Validation data에서 구한 optimal threshold를 이용해 Test set으로 검증한 결과는 다음과 같다.

Train set

모델 학습



Best alpha : 0.01

Best MSE : 0.1404

Validation set

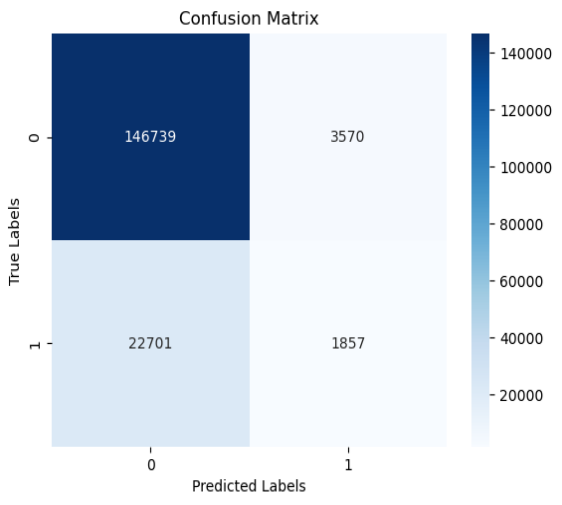
Optimal Threshold 찾기

Optimal Threshold : 0.3284

Min Revenue Risk : 0.0603

Test set

모델 평가



Ridge Regression

Logistic Regression - Ridge Model 결과의 투자금, 수익금, 순수익, 수익률 지표는 다음과 같다.

	All	Test	Test → Modeling
투자금	119억 달러	23억 8천만 달러	22억 9천만 달러
수익금	130억 달러	26억 달러	24억 9천만 달러
순수익	10억 8천만 달러	2억 1천만 달러	2억 8백만 달러
수익률	9.08%	9.06%	9.10%

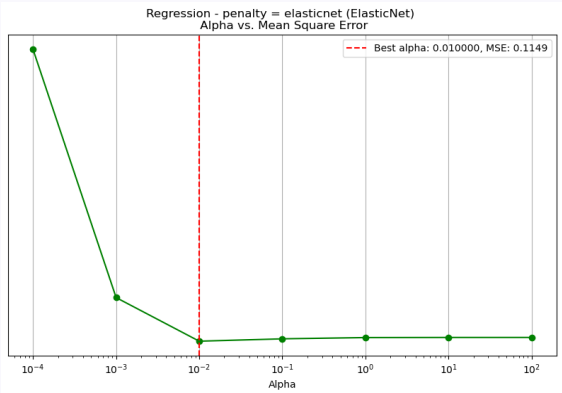
4) Logistic Regression - ElasticNet

Logistic Regression - ElasticNet

표준화한 Train set으로 학습된 Logistic Regression - ElasticNet 모델과 Validation data에서 구한 optimal threshold를 이용해 Test set으로 검증한 결과는 다음과 같다.

Train set

모델 학습



Best alpha : 0.01

Best MSE : 0.1149

Validation set

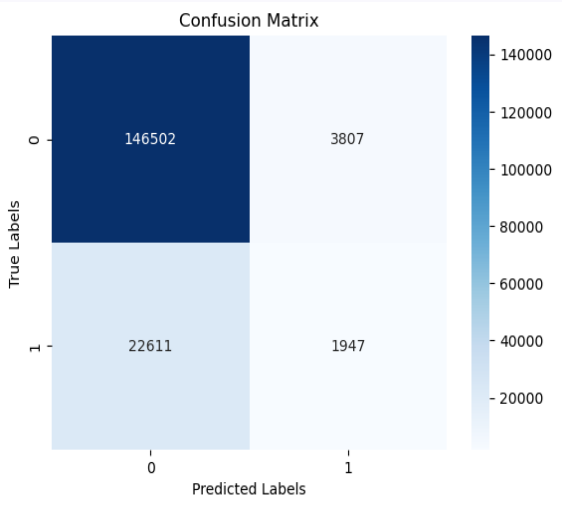
Optimal Threshold 찾기

Optimal Threshold : 0.3223

Min Revenue Risk : 0.0603

Test set

모델 평가



Logistic Net Regression - ElasticNet

Logistic Regression - ElasticNet Model의 투자금, 수익금, 순수익, 수익률 지표는 다음과 같다.

	All	Test	Test → Modeling
투자금	119억 달러	23억 8천만 달러	22억 8천만 달러
수익금	130억 달러	26억 달러	24억 9천만 달러
순수익	10억 8천만 달러	2억 1천만 달러	2억 7백만 달러
수익률	9.08%	9.06%	9.09%

5) Random Forest

Random Forest

Train set으로 학습된 Random Forest 모델과 Validation data에서 구한 optimal threshold를 이용해

Test set으로 검증한 결과는 다음과 같다.

Train set

모델 학습

random_state= 0

hyperparameter tuning 결과

n_estimators = 100

max_depth = 20

min_samples_split = 5

min_samples_leaf =2

Validation set

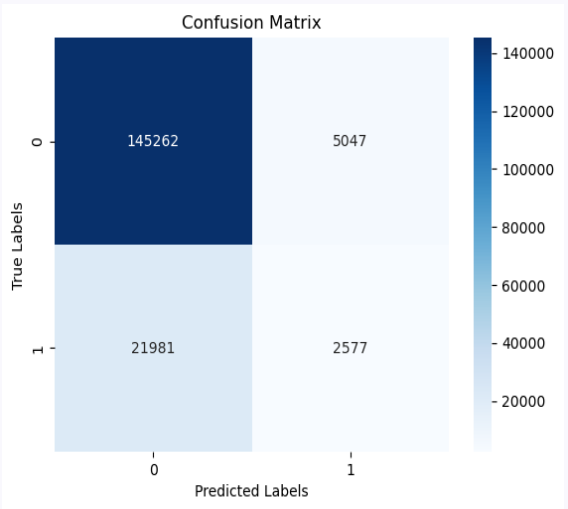
Optimal Threshold 찾기

Optimal Threshold : 0.2721

Min Revenue Risk: 0.0605

Test set

모델 평가



Random Forest

Random Forest Model의 투자금, 수익금, 순수익, 수익률 지표는 다음과 같다.

	All	Test	Test → Modeling
투자금	119억 달러	23억 8천만 달러	22억 4천만 달러
수익금	130억 달러	26억 달러	24억 5천만 달러
순수익	10억 8천만 달러	2억 1천만 달러	2억 6백만 달러
수익률	9.08%	9.06%	9.20%

5. 결론 및 함의

모델의 평가 지표 비교

5개 모델 중 RandomForest Model이 수익률을 가장 최대화하는 것으로 나타났다. 이 모델을 사용해 전체 투자금인 119억 달러를 투자금으로 사용할 경우 기대 수익이 1600만 달러만큼 증가한다.

	Logistic Regression	Logistic Regression - Lasso	Logistic Regression - Ridge	Logistic Regression - ElasticNet	RandomForest
Accuracy	0.8485	0.8487	0.8500	0.8489	0.8442
Sensitivity	0.0824	0.0826	0.0756	0.0793	0.1110
Specificity	0.9740	0.9739	0.9762	0.9747	0.9640
f1-score	0.5050	0.5052	0.5067	0.5023	0.4972
Revenue_risk	10619.30	10618.36	10631.04	10634.1	10580.06
수익률	9.09%	9.10%	9.10%	9.09%	9.20%

모델의 수익금, 수익률 비교

	Logistic Regression	Logistic Regression - Lasso	Logistic Regression - Ridge	Logistic Regression - ElasticNet	Random Forest
All	9.08% 10억 8천만 달러	9.08% 10억 8천만 달러	9.08% 10억 8천만 달러	9.08% 10억 8천만 달러	9.08% 10억 8천만 달러
Test	9.06% 2억 2천만달러	9.06% 2억 2천만달러	9.06% 2억 2천만달러	9.06% 2억 2천만달러	9.06% 2억 2천만달러
Modelling	9.10% 2억 7백만 달러	9.10% 2억 8백만 달러	9.08% 2억 8백만 달러	9.09% 2억 7백만 달러	9.20% 2억 6백만 달러

모델의 평가 지표 비교 - out of set 적용

Out-of-sample test set을 적용하여 모델링한 결과 기존의 데이터와 마찬가지로 Random Forest 가 가장 좋은 모델임을 확인할 수 있었으며 그 때 수익률이 9.28%를 갖는다

	Logistic Regression	Logistic Regression - Lasso	Logistic Regression - Ridge	Logistic Regression - ElasticNet	RandomForest
Accuracy	0.8508	0.8520	0.8508	0.8511	0.8456
Sensitivity	0.0819	0.0759	0.0820	0.0795	0.1092
Specificity	0.9745	0.9769	0.9745	0.9753	0.9641
f1-score	0.5061	0.5112	0.5058	0.5060	0.4906
Revenue_risk	13099.98	13101.74	13100.80	13107.44	13090.78
수익률	9.21%	9.22%	9.21%	9.20%	9.28%

모델의 수익금, 수익률 비교 - out of set

	Logistic Regression	Logistic Regression - Lasso	Logistic Regression - Ridge	Logistic Regression - ElasticNet	Random Forest
All	9.15% 2억 7천만 달러	9.15% 2억 7천만 달러	9.15% 2억 7천만 달러	9.15% 2억 7천만 달러	9.15% 2억 7천만 달러
Modelling	9.21% 2억 6천만 달러	9.22% 2억 6천만 달러	9.21% 2억 6천만 달러	9.20% 2억 6천만 달러	9.28% 2억 6천만 달러

- 총 투자금 119억 달러를 사용해 총 기대 수익을 최대 7.5 * 119/102 = 8억 7천만 달러까지 증가시킬 수 있는 상황에서 가장 성능이 좋은 모델조차 기대 수익을 크게 올리지는 못하였다.

- 자료에 담긴 내용 외에도 수익률을 최대로 높이기 위해 다양한 조합을 시도해보았다. 설명변수 선택, 반응변수 정의, xgboost knn 등 이외의 모델, x변수 scale 방식, hyperparameter tuning 방식, optimal threshold 구하는 방식, 새롭게 만든 평가 지표 등 하나하나 바꿔가며 다양하게 모델링해보았지만, 수익률을 최대 9.6%으로까지밖에 올리지 못하였다.

- 이 결과를 통해 설명변수들과 반응변수와의 상관성이 크지 않고, imbalanced data이고, 모델링의 한계로 인해서 모델 성능이 좋지 않게 나타나는 것이라고 결론지었다.

- 또, lending club 측에서 자체적인 고객 신용평가를 통해 대출을 승인해준 고객들에 대한 자료이고, 이미 전체 수익률 9%를 얻은 자료이기 때문에 이를 다른 변수들을 통해 모델링해 더 확연히 나은 기대 수익률을 얻기가 어려운 것이라 유추하였다.

- 그럼에도 위에서 시도한 여러 조합들 중 적절한 조합을 선택해 더 다양한 모델링 방식들을 시도해본다면 지금까지의 결과보다 더 나은 결과를 얻을 수 있을 것이라 기대한다.

6. 추가 사항

모델 평가 척도 추가 제시: 수익률, 수익 관점

우리가 대출해 준 금액과 돌려 받은 금액을 기반으로

수익과 수익률을 validation set에서 최대로 하는 threshold 값을 정해주어

동일한 test set에 적용시켜 모델별로 어느 모델의 성능을 비교할 수 있다고 파악하여 진행함

대출 해주는 대상 (= 학습한 모델의 결과가 threshold 값 이하를 나타내주는 사람)

수익 : (대출해주는 대상들이 갚은 총 금액) - (대출 해주는 대상들이 빌려간 총 금액)

수익률 : ((대출해주는 대상들이 갚은 총 금액) / (대출 해주는 대상들이 빌려간 총 금액) - 1) *100

모델의 평가 지표 비교 - 수익 최대화

모든 방법들이 대략 2억 2천만 달러의 수익금을 가질 수 있다.

	Logistic Regression	Logistic Regression - Lasso	Logistic Regression - Ridge	Logistic Regression - ElasticNet	RandomForest
Accuracy	0.8595	0.8594	0.8594	0.8594	0.8592
Sensitivity	0.0070	0.0070	0.0073	0.0056	0.0060
Specificity	0.9987	0.9987	0.9986	0.9989	0.9985
f1-score	0.6475	0.6464	0.6335	0.6239	0.5801
수익금	2억 2천만 달러	2억 2천만 달러	2억 2천만 달러	2억 2천만 달러	2억 2천만 달러
수익률	9.09%	9.09%	9.08%	9.09%	9.08%

모델의 평가 지표 비교 - 수익 최대화 - out of set

모든 방법들이 대략 2억 7천만 달러의 수익금을 가질 수 있다.

	Logistic Regression	Logistic Regression - Lasso	Logistic Regression - Ridge	Logistic Regression - ElasticNet	RandomForest
Accuracy	0.8610	0.8610	0.8610	0.8610	0.8609
Sensitivity	0.0063	0.0063	0.0066	0.0051	0.0055
Specificity	0.9986	0.9986	0.9985	0.9988	0.9986
f1-score	0.5972	0.5963	0.5914	0.5765	0.5568
수익금	2억 7천만 달러	2억 7천만 달러	2억 7천만 달러	2억 7천만 달러	2억 7천만 달러
수익률	9.17%	9.17%	9.17%	9.16%	9.16%

모델의 평가 지표 비교 - 수익률 최대화

수익률이 Logistic Regression - Ridge의 경우에 10.41% 로 가장 높은 수익률을 나타냈으나 수익금이 매우 적었다.

	Logistic Regression	Logistic Regression - Lasso	Logistic Regression - Ridge	Logistic Regression - ElasticNet	RandomForest
Accuracy	0.1409	0.1408	0.1408	0.1408	0.8330
Sensitivity	0.9999	0.9999	0.9999	0.9999	0.1621
Specificity	0.0005	0.0005	0.0004	0.0004	0.9426
f1-score	0.0010	0.0010	0.0009	0.0008	0.4730
수익금	17만 8천 달러	18만 4천 달러	17만 9천 달러	16만 7천 달러	2억 7십만 달러
수익률	10.37%	10.37%	10.41%	10.39%	9.22%

모델의 평가 지표 비교 - 수익률 최대화 - out of set

수익률이 Logistic Regression - Ridge의 경우에 10.68% 로 가장 높은 수익률을 나타냈으나 수익금이 매우 적었다

	Logistic Regression	Logistic Regression - Lasso	Logistic Regression - Ridge	Logistic Regression - ElasticNet	RandomForest
Accuracy	0.1390	0.1390	0.1390	0.1389	0.8336
Sensitivity	0.9999	0.9999	0.9999	0.9999	0.1608
Specificity	0.0004	0.0004	0.0004	0.0003	0.9419
f1-score	0.0009	0.0009	0.0008	0.0007	0.4647
수익금	22만 1천달러	22만 9백달러	21만 4천 달러	19만 7백 달러	2억 5천만 달러
수익률	10.62%	10.63%	10.68%	10.58%	9.28%

모델 평가 기준 변경 : remains 도입

우리가 대출해 준 금액과 돌려 받은 금액을 기반으로 수익이 생기는 경우 ‘loan_not_access’ 라는 변수를 도입하지 않고 수익액 ‘remains’를 도입하여 대출 받을 사람에대한 예상 수익액을 직접 계산해 보았습니다.

기존의 loan_not_access는 (total_pymnt - funded_amnt) > 0 이면 0 , (total_pymnt - funded_amnt) < 0 이면 1 이라 놓고 진행해 왔으나, 지금 제시한 remains는 total_pymnt - funded_amnt 로 대출 받는 사람이 주는 수익을 기반으로 학습하여 예상했을 때 수익액이 0보다 크면 돈을 빌려주는 형식으로 진행 하였다.

1차 회귀 모델 방식과 2차 회귀 모델 방식만 적용시켜 보았다.

remains 기준 결과 비교

	1차 회귀 모델	2차 회귀 모델
Test	9.05% 2억 2천만 달러	9.05% 2억 2천만 달러
Modelling	9.16% 2억 1천만 달러	9.19% 2억 1천만 달러

	1차 회귀 모델	2차 회귀 모델
Accuracy	0.8099	0.8368
Sensitivity	0.0817	0.0597
Specificity	0.9289	0.9638
f1-score	0.2700	0.3476

remains 기준 결과 비교 - out of set

	1차 회귀 모델	2차 회귀 모델
All	9.15% 2억 7천만 달러	9.15% 2억 7천만 달러
Modelling	9.23% 2억 7천만 달러	9.26% 2억 7천만 달러

	1차 회귀 모델	2차 회귀 모델
Accuracy	0.8126	0.8389
Sensitivity	0.0798	0.0615
Specificity	0.9305	0.9640
f1-score	0.2670	0.3528

- 1차 발표 이후, 기존 모델의 성능을 개선하기 위해 각 모델별로 새로운 방법을 시도하여 218,584개의 out-of-sample test set을 적용한 2차 최종 모델링 과정을 진행하였다.

- 기존 방법과 같이 out-of-sample test에서도 약간의 수익률 상승을 이루었고 기존 방법과 똑같이 최고 수익률은 Random Forest 모델로 9.28%의 수익률이 나오게 되었다.

- 교수님들께서 강조하셨던 최대 수익이나 수익률이 가장 좋을때에 대하여 모델에 적용시켜보았으나 예상했던 바와 같이 수익을 높이기 위해 threshold값을 정하면 매우 많은 인원들에게 돈을 빌려주게 되면 수익률이 적게 나오고, 수익률을 높이기 위해 threshold값을 정하면 매우 많은 인원들에게 빌려주지 않아 수익금이 매우 적은 문제가 생김을 확인하였다. 따라서 위에서 우리가 정한 Risk가 수익금을 어느정도 보장해주는 상태에서 높은 수익률을 가져다 주는 좋은 척도임을 확인하였다.

- 교수님들께서 피드백 주신 수익 금액 자체를 예측하는 것도 해보았을 때 수익률과 수익을 어느정도 보장해주는 성과를 나타내 주었다. 따라서 내가 직접 운영한다면 위에서 Random Forest 모델을 정한 Risk 척도에서 안전하고 수익금을 예측한 2차 회귀 모델에서 예측한 값이 0보다 큰 경우에 빌려주는 식으로 운영한다면 많은 수익과 안정성을 보장할 수 있을 것이라고 생각된다.