SNU 빅데이터 핀테크 과정 통계데이터사이언스

Lending club 부도예측 모형

권예준, 김경빈, 원소현, 이아연, 전우근

INDEX

01

분석 목적

Lending club

- **2007년 설립된 미국의 P2P(Peer to Peer) 대출업체**
- 금융기관을 거치지 않고 온라인상에서 직접 자금 수요자와 공급자를 연결시켜주는 방식
- 대출 이자의 일부를 수수료로 수익을 얻고 나머지 금액은 투자자에게 투자 수익으로 제공
- 개인 대출 서비스를 넘어 자산 운용, 대출 채권, 기업 대출까지 사업 영역을 확장

코로나19, 금융위기 등의 주요 발생 원인으로 금융환경이 급속하게 변화함에 따라 신용 부도 사건은 여전히 자주 발생하고 있다.

특히 기업은 부도가 발생할 경우 관련 기업의 근무자, 유관기업 및 개인, 금융기관을 비롯한 주요 투자자 및 채권자 등 모든 연관된 경제 주체에 연쇄적인 피해가 발생할 수 있다.

기업 부도 위험을 보다 정확히 예측하고 평가하고자 **lending club** 데이터를 기반으로 분석 연구를 하였다.

정확한 부도 예측을 통해 기대수익을 극대화하고 채무 불이행 손실을 최소화하고자 하는 것이 본 연구의 목적이다.

02

데이터 전처리

1) 데이터 구조파악

> 1092919개 행, 333개 열로 이루어진 dataframe

```
df.shape
```

```
(1092919, 333)
```

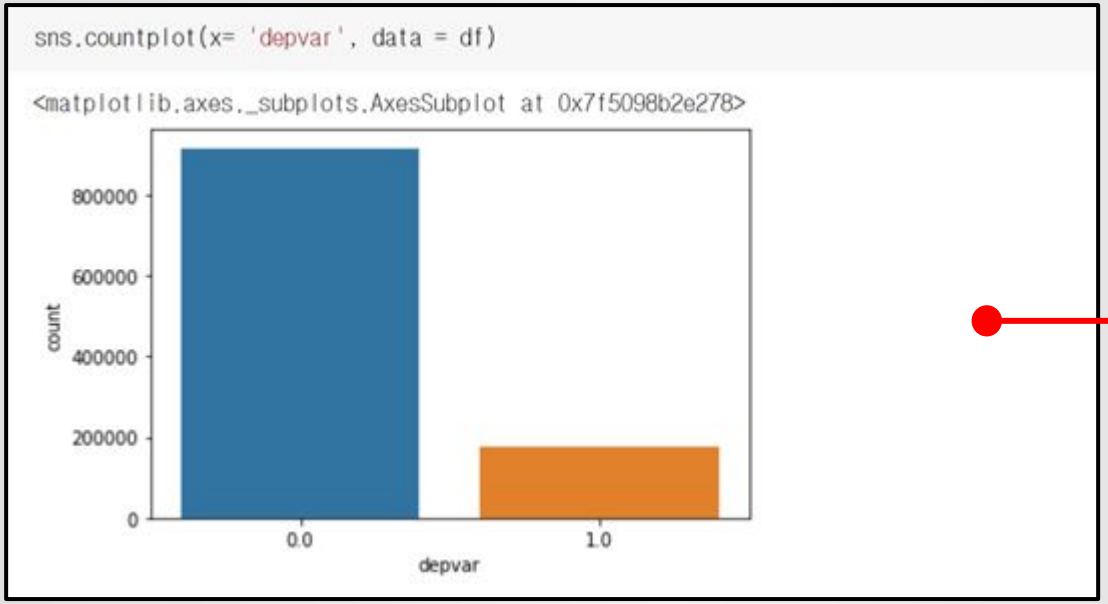
```
df.head()
```

	loan_amnt	funded_amnt	funded_amnt_inv	int_rate	installment	annual_inc	dti	delinq_2yrs	fico_range_low	fico_rang
0	19000	19000	19000.0	0.0916	605.62	65000.0	16.36	1	670	
1	10000	10000	10000.0	0.0789	312.86	58000.0	5.03	0	690	
2	6000	6000	6000.0	0.1147	197.78	46900.0	24.23	2	665	
3	25200	25200	25200.0	0.1199	836.89	76280.0	32.87	0	685	
4	8000	8000	8000.0	0.1299	269.52	29000.0	20.28	0	770	

5 rows × 333 columns

1) 데이터 구조파악

> 종속변수 'depvar'= 0일 경우 '부도아님' , 'depvar'=1일 경우 '부도'



약 16.2%의
부도확률

2) 변수 제거

> 대출 해주는 시점에서 관측 불가능한 변수와 가능한 변수 구분

```
post_attr = ['out_prncp','out_prncp_inv', 'total_pymnt','total_pymnt_inv', 'total_rec_prncp','total_rec_int',
'recoveries', 'total_rec_late_fee', 'collection_recovery_fee', 'last_pymnt_amnt', 'debt_settlement_flag',
'installment','elapsed_t','term']

for i in range(1,119):
    col = 'issue_d' + str(i)
    post_attr.append(col)

cat_lists = ['mths_since_last_delinq','mths_since_last_major_derog','mths_since_last_record',
             'mths_since_rcnt_il','mths_since_recent_bc','mths_since_recent_bc_dlq','mths_since_recent_revol_delinq']

for col in cat_lists:
    for i in range(1,12):
        coli = col + str(i)
        post_attr.append(coli)

for i in range(1,11):
    col = 'mths_since_recent_inq' + str(i)
    post_attr.append(col)
```

> 관측 불가능한 변수를 제거하여 114개의 열로 축소

```
pre_attr = list(set(df.columns) - set(post_attr))
pre_attr.sort()

df = df[pre_attr]
df.shape
```

```
(1092919, 114)
```

3) Splitting data

> train set : validate set : test set = 6 : 2 : 2 로 분할

```
X = df.drop('depvar', axis=1)
y = df['depvar']

X_train, X_test, y_train, y_test = train_test_split(X, y, train_size=0.8, test_size=0.2, random_state=47)
X_train, X_valid, y_train, y_valid = train_test_split(X_train,y_train,test_size=0.25,random_state=73)
```

> validation set의 경우 optimal threshold를 찾기위한 수단으로 사용

4) Optimal Threshold 기준 - 기대수익 극대화

가정:

① class 0과 class 1인 사람들의 평균 대출금액은 같다.
② (prior) class0와 class 1은 주어진 자료의 비율을 고려해 4.5:1이라고 가정.
③ 1종 오류율 : 실제 class 0인 사람을 class 1으로 판단하는 비율
2종 오류율 : 실제 class 1인 사람을 class 0으로 판단하는 비율

설정원리 :

① 부도확률은 추후에 소개될 기계학습의 모델들을 활용해 도출을 한다.
② (threshold) 임의의 데이터에 대해 모델이 측정한 부도확률이 특정한 값(theshold)을 넘으면 class 1로, 넘지 않으면 class 0로 판단한다.

threshold ↑ ▸ 예측 class 0 ↑ ▸ 부도비율 ↑

③ (최적의 threshold) threshold를 1단위 증가시켰을 때 증가하는 한계 이자 수익 = 증가하는 한계 부도 금액

: 한계이자수익 > 한계부도금액 ► threshold를 1단위 높임으로써 수익 증대
한계이자수익 < 한계부도금액 ► threshold를 1단위 낮춤으로써 수익 증대

4) Optimal Threshold 기준 - 기대수익 극대화

④ **(한계 이자 수익)**

class 0인 사람들의 평균 이자율 = 약 12.2%

total_rec_prncp(대출 상환 원금) > 원금의 규모 도출, 약 $13,500

→ 추정 한계이자수익 = “12.2%•$13,500•1종 오류 감소의 절대치"

⑤ **(한계 부도 금액)**

class 1인 사람들의 평균 total_pymnt_inv = 약 $9,100

class 1인 사람들의 평균 원리금 회수율은 약 67%(=9,100/13,500) (가정①)

부도를 낸 사람들이 평균적으로 원리금의 67%를 상환했고 원금의 33%를 갚지 못했다.

→ 추정 한계 부도 금액 = “33%•$13,500•2종 오류 증가의 절대치"

⑥ 가정 ①과 ②에 따라 class 0이 대략 class 1보다 4.5배 많으므로 최적의 threshold를 도출하는 균형조건

“-12.2%•4.5•한계 1종 오류율" = “33%•1•한계 2종 오류율"

5) 기대수익 극대화 균형조건

$$-int_0 \times n_0 \times \frac{de_1}{dt} = int_1 \times n_1 \times \frac{de_2}{dt}$$

$int_0 : interest\ rate\ for\ class\ 0$
$int_1 : interest\ rate\ for\ class\ 1$
$n_0 : total\ number\ of\ class\ 0$
$n_1 : total\ number\ of\ class\ 1$
$e_1 : type\ 1\ error\ rate$
$e_2 : type\ 2\ error\ rate$
$t : threshold$

6) Optimal Threshold 탐색 코드

```
# finding optimal threshold
for i in threshold_interval:
    valid_result_temp0 = pd.DataFrame(log_reg_dt.predict_proba(X_valid_dt)[:,1],columns=['pred_prob'])
    valid_result_temp0['binary'] = np.where(valid_result_temp0['pred_prob']>=(i-0.01),1,0)

    valid_result_temp1 = pd.DataFrame(log_reg_dt.predict_proba(X_valid_dt)[:,1],columns=['pred_prob'])
    valid_result_temp1['binary'] = np.where(valid_result_temp1['pred_prob']>=i,1,0)

    cm_temp0 = confusion_matrix(y_test,valid_result_temp0['binary'])
    cm_norm_temp0 = cm_temp0 / cm_temp0.sum(axis=1).reshape(-1,1)

    cm_temp1 = confusion_matrix(y_test,valid_result_temp1['binary'])
    cm_norm_temp1 = cm_temp1 / cm_temp1.sum(axis=1).reshape(-1,1)
    # print("1-fpr : tpr = {} : {}".format(a*5/100,1-(a*5/100)))
    # print("optimal threshold : {}".format(np.ceil(optimal_threshold_logit_temp*1000)/1000))
    print(i)
    print(-4.5*12*(cm_norm_temp1[0,1]-cm_norm_temp0[0,1])>33*(cm_norm_temp1[1,0]-cm_norm_temp0[1,0]))
    print("accuracy is :", (cm_temp1[0,0]+cm_temp1[1,1])/(cm_temp1[0,0]+cm_temp1[0,1]+cm_temp1[1,0]+cm_temp1[1,1]))
    print('')
```

- threshold를 0.0부터 1.0까지 0.01단위로 움직이며 앞에서 설정한 기준 충족 확인

03

사용 분류 모델 설명

1) 사용 분류 모델

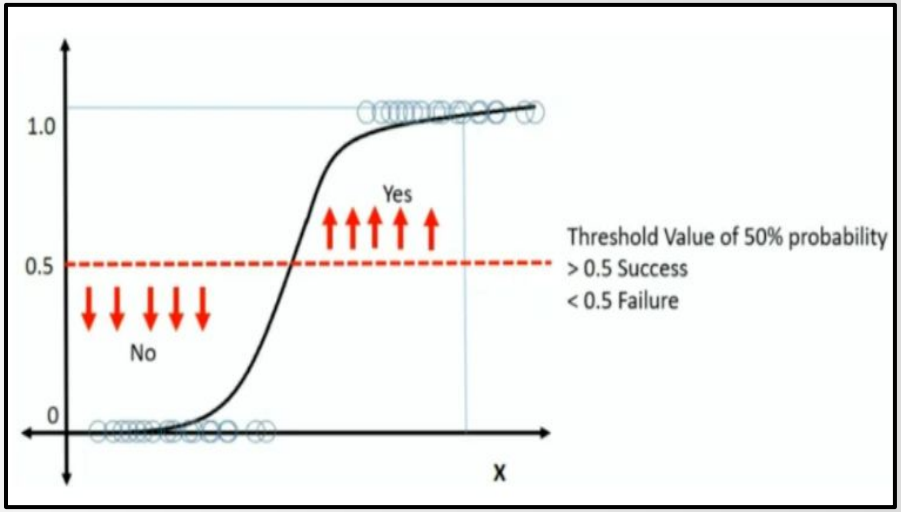
1. Decision Tree + Logistic Regression

2. Lasso + Logistic Regression

3. Random Forest Classifier

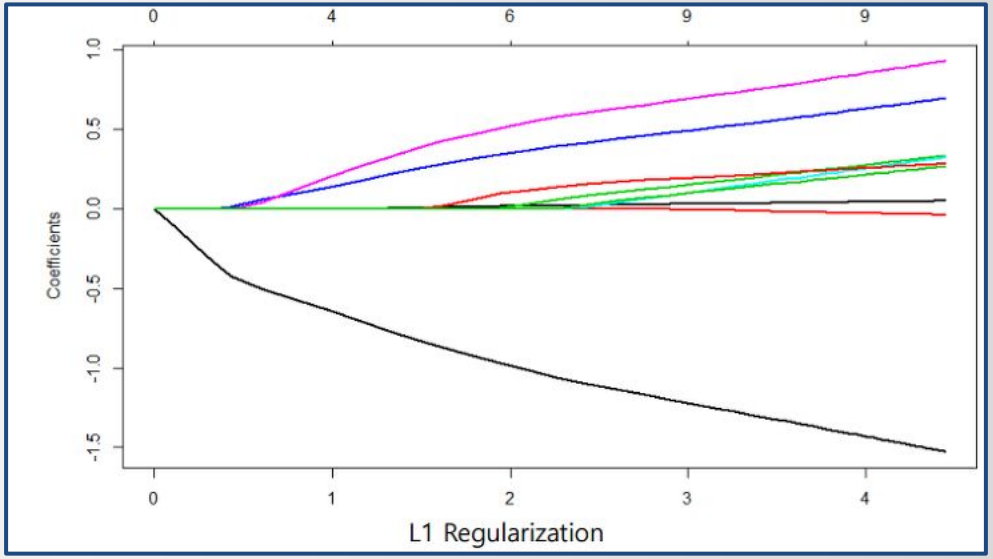
1. Decision Tree + Logistic Regression

- **Logistic Reg** : 반응변수가 1 또는 0인 이진형 변수에서 쓰이는 회귀분석 방법
- **Decision Tree Classifier** : 고차원 데이터에 적용이 가능하며, 데이터 전처리가 거의 필요 없다는 장점이 있음.
  > distribution에대한 가정이 필요 없다는 장점으로 Feature Selection에 가장 보편적으로 쓰이는 기법이다.
- 하지만 과적합의 가능성이 있기 때문에 Feature Selection 이후 과정은 Logistic Regression을 이용함



2. Lasso + Logistic Regression

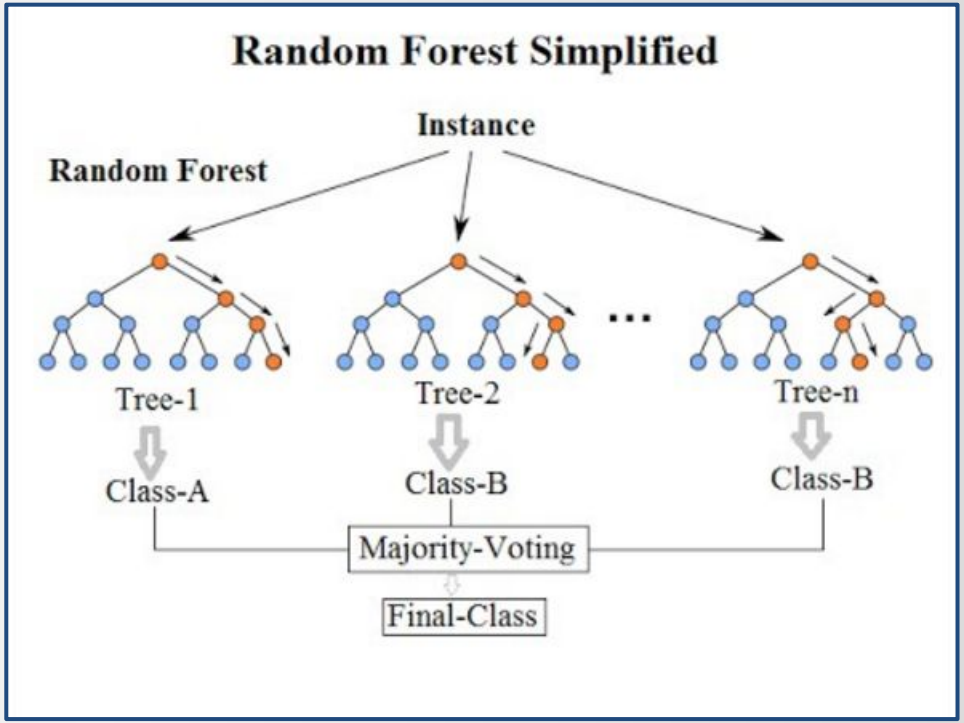
- **Lasso :** 모델의 복잡도를 줄이기 위해 사용하는 regularization tool. 중요도가 낮은 변수들의 coefficient를 0으로 만들어주기 때문에 feature selection의 기능도 가짐. 람다(hyperparameter)의 값을 조정함으로써 복잡도 조절.
- Lasso를 이용하기 위해 solver로는 saga를 사용하였으며, C = 0.001을 적용하여 모델의 복잡도를 낮춤



이미지 출처 : https://sosal.kr/1104

3. Random Forest Classifier

- **Random Forest Classifier** : 랜덤으로 선택한 feature을 가지고 decision tree를 평가하는 앙상블 기법을 활용한 분류기. decision tree의 과적합(over-fitting) 문제점을 해결할 수 있음.
- Random Forest Classifier를 이용하여 중요한 feature를 확인하였으며, Validation과 test set을 이용하여 결과를 분석함



이미지 출처 : https://en.wikipedia.org/wiki/Random_forest

04

모델 적용 및 결과 해석(1)
-Decision Tree + Logistic Regression-

- Decision Tree + Logistic Regression

1) Feature Selection

```
fi_col1 = []
fi1 = []

for i, column in enumerate(X):
  fi_col1.append(column)
  fi1.append(dt_model.feature_importances_[i])

fi_df1 = pd.DataFrame(zip(fi_col1, fi1), columns=['features', 'feature importance'])
fi_df1 = fi_df1.sort_values('feature importance', ascending=False).reset_index()
idx_to_drop = fi_df1.index[fi_df1['feature importance'] == 0].tolist()
fi_df1.drop(index = idx_to_drop,inplace=True) # feature importance 가 0인 변수들 제거
print(fi_df1.shape) # 113개중에 75개 살아남음
fi_df1.head()
```

(75, 3)

	index	features	feature importance
0	1	last_fico_range_low	0.868731
1	0	last_fico_range_high	0.059833
2	2	int_rate	0.009749
3	13	emp_length12	0.007599
4	12	loan_amnt	0.007371

> Decision tree를 이용하여 분류후 feature importance 가 0인 변수들 제거

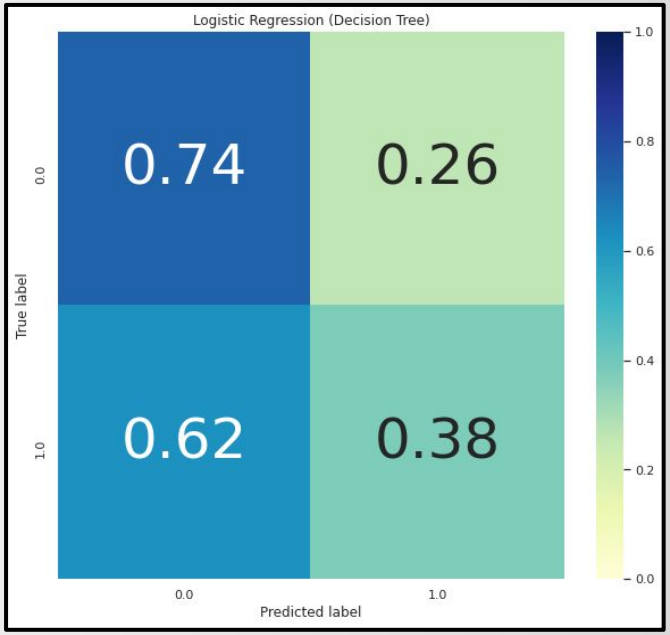
> 113개에서 75개로 줄어들었음을 확인할 수 있음(depvar는 타겟변수로 제외)

- Decision Tree + Logistic Regression

2) Selected Feature를 이용한 모델링

```
w = {0:1, 1:4.5}
log_reg_dt = LogisticRegression(random_state=37, solver='saga', class_weight=w)
log_reg_dt.fit(X_train_dt, y_train)
```

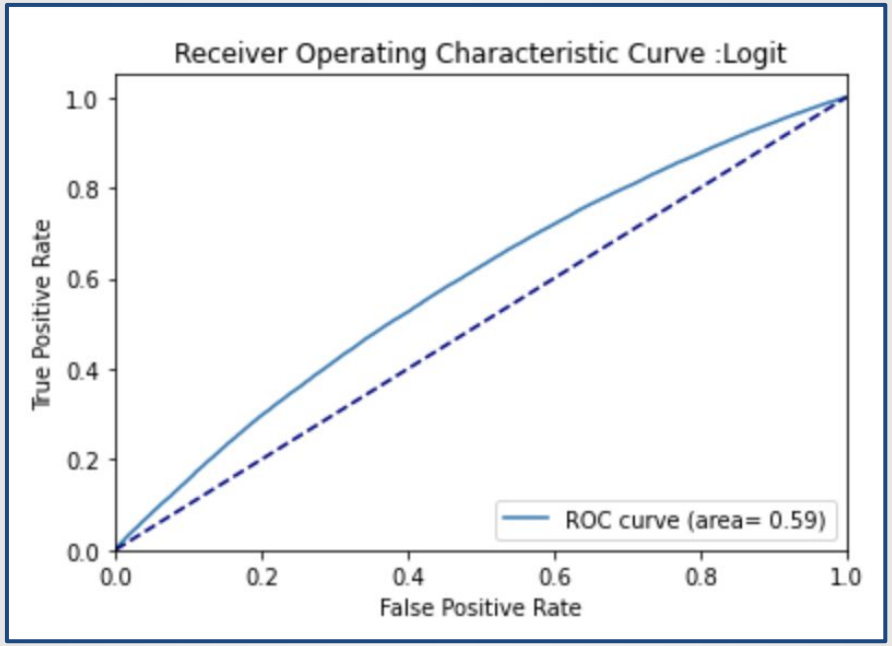
```
Train accuracy is:  0.6793249266871114
Test accuracy is:  0.6798484793031512
              precision    recall  f1-score   support

         0.0       0.86      0.74      0.79    549698
         1.0       0.22      0.38      0.28    106053

    accuracy                           0.68    655751
   macro avg       0.54      0.56      0.53    655751
weighted avg       0.76      0.68      0.71    655751
```



> LogisticRegression의 경우 class_weight를 주어야 편향되지 않은 결과를 도출하기 때문에 class0과 class1의 비중에 맞춰서 w를 주었음.

- Decision Tree + Logistic Regression

3) Validation Set을 적용하여 auc & roc curve 확인



- auc = 0.59
  : 1에 가까울수록 좋은 모델 , 예측 성능↓

- Decision Tree + Logistic Regression

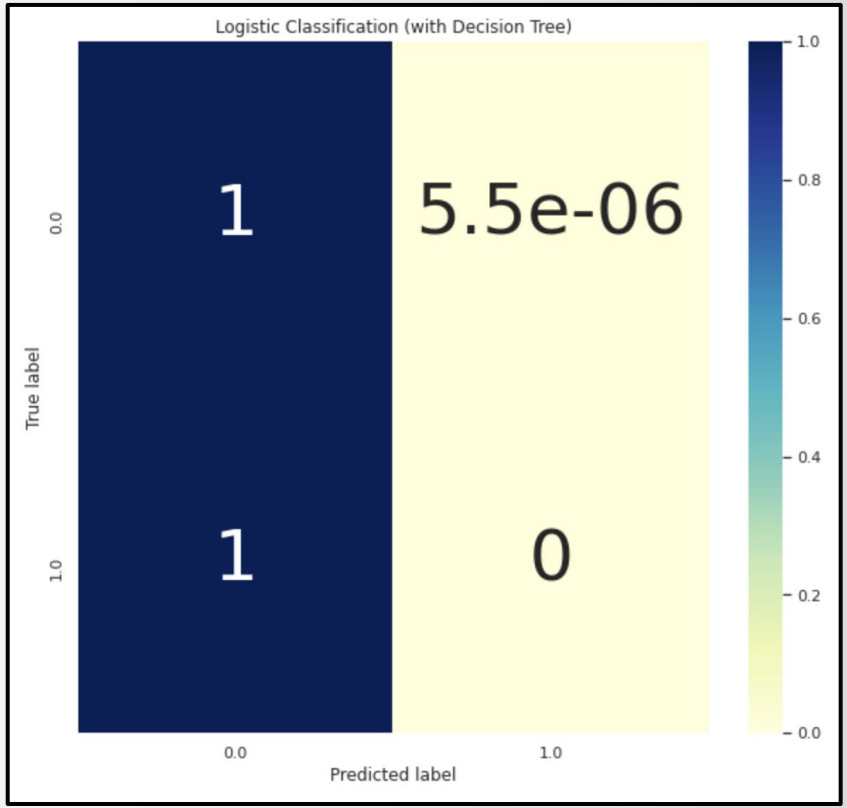
4) Optimal Threshold 선택 & Test 검증

optimal threshold = 0.62
accuarcy = 0.8384465468652783

```
cm2 = confusion_matrix(y_test,test_result_lg1['dt_binary'])
cm2
```

```
array([[183271,      1],
       [ 35312,      0]])
```

수익을 극대화하는 optimal threshold는 비록 높은 accuracy를 달성했으나 recall이 형편 없어 대출 심사 모델로 사용하기 부적합하다고 판단함.



04

모델 적용 및 결과 해석(2)

- Lasso + Logistic Regression-

- Lasso + Logistic Regression

1) Lasso Penalty 적용

```
w = {0:1, 1:4.5}
log_reg_ls = LogisticRegression(penalty = 'l1',random_state=37, solver='saga', class_weight=w, C=0.001)
log_reg_ls.fit(X_train, y_train)
```

```
# 살아남은 변수 확인하기
fi_col4 = []
fi4 = []

for i, column in enumerate(X):
  fi_col4.append(column)
  fi4.append(log_reg_ls.coef_[0][i])

fi_df4 = zip(fi_col4, fi4)
fi_df4 = pd.DataFrame(fi_df4, columns=['features', 'feature importance'])
fi_df4 = fi_df4.sort_values('feature importance', ascending=False).reset_index()

idx_to_drop3 = fi_df4.index[fi_df4['feature importance'] == 0].tolist()
fi_df4.drop(index = idx_to_drop3,inplace=True) # feature importance 가 0인 변수들 제거
print(fi_df4.shape) # 113개중에 41개 살아남음
fi_df4.iloc[:,1:]
```

(41, 3)

	features	feature importance
0	funded_amnt	6.849508e-06
1	loan_amnt	6.849282e-06
2	funded_amnt_inv	6.847025e-06
3	tot_coll_amt	2.081605e-07
4	delinq_amnt	1.525486e-07

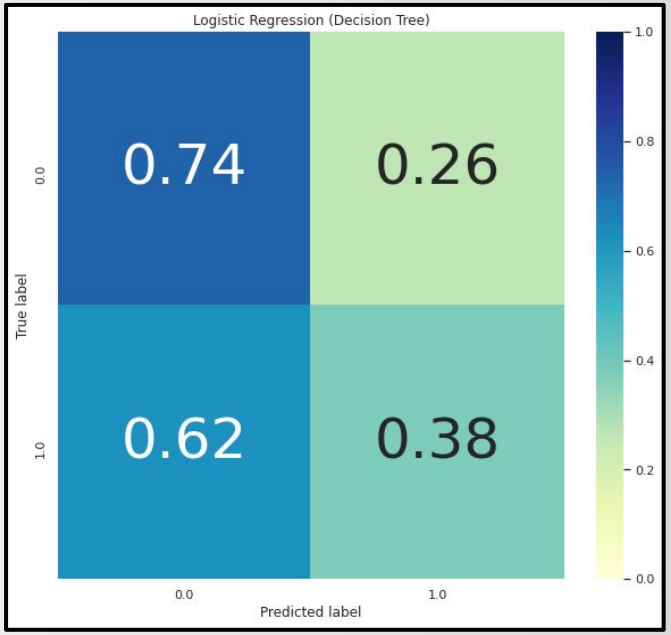
> LogisticRegression의 경우 class_weight를 주어야 편향되지 않은 결과를 도출하기 때문에 class0과 class1의 비중에 맞춰서 w를 주었음.

> 113개에서 41개의 Feature가 의미를 가짐

- Lasso + Logistic Regression

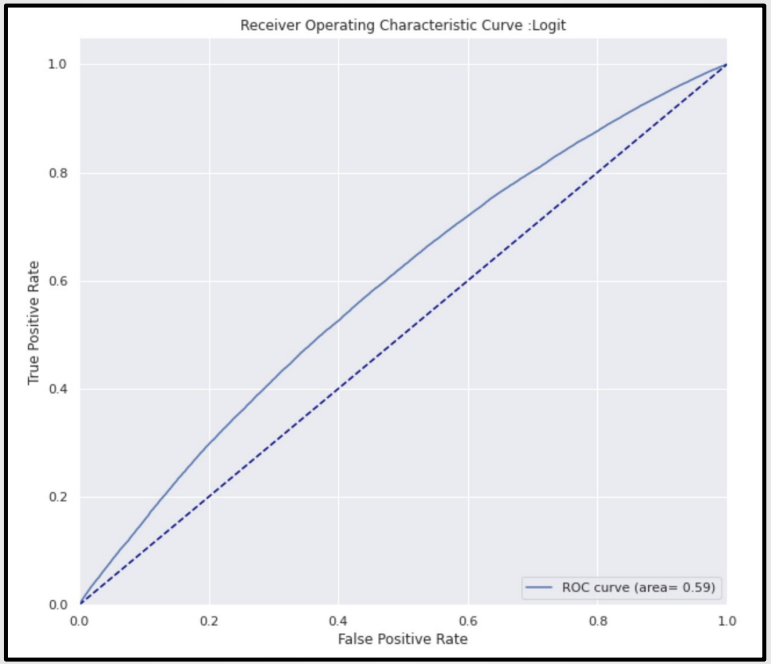
2) Selected Feature를 이용한 모델링

```
Train accuracy is:  0.6793249266871114
Test accuracy is:  0.6798484793031512
              precision    recall  f1-score   support

         0.0       0.86      0.74      0.79    549698
         1.0       0.22      0.38      0.28    106053

    accuracy                           0.68    655751
   macro avg       0.54      0.56      0.53    655751
weighted avg       0.76      0.68      0.71    655751
```



- Lasso + Logistic Regression

3) Validation Set을 적용하여 auc & roc curve 확인



- auc = 0.59
  : 1에 가까울수록 좋은 모델, 예측 성능↓

- Lasso + Logistic Regression

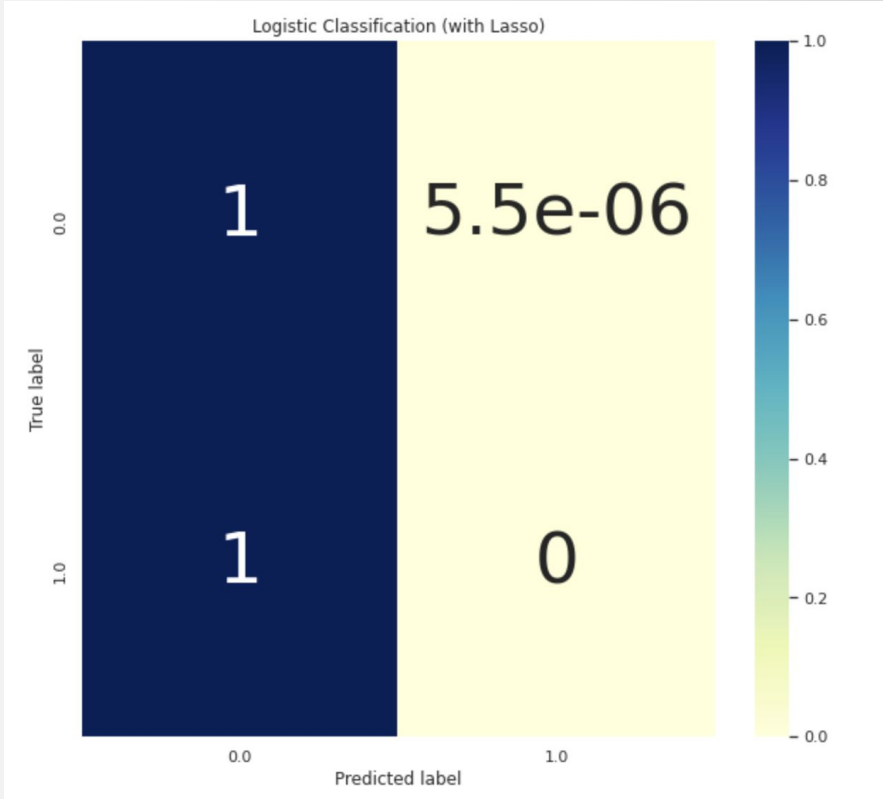
4) Optimal Threshold 선택 & Test 검증

- optimal_threshold_ls_lg = 0.62
- accuracy : 0.8384465468652783

```
cm2 = confusion_matrix(y_test,test_result_lg2['ls_binary'])
cm2
```

```
array([[183271,      1],
       [ 35312,      0]])
```

앞선 decision tree model과 마찬가지로
recall이 형편 없어 대출 심사 모델로 사용하기
부적합하다고 판단함.



04

모델 적용 및 결과 해석(3)
-Random Forest Classifier-

1) Feature Selection

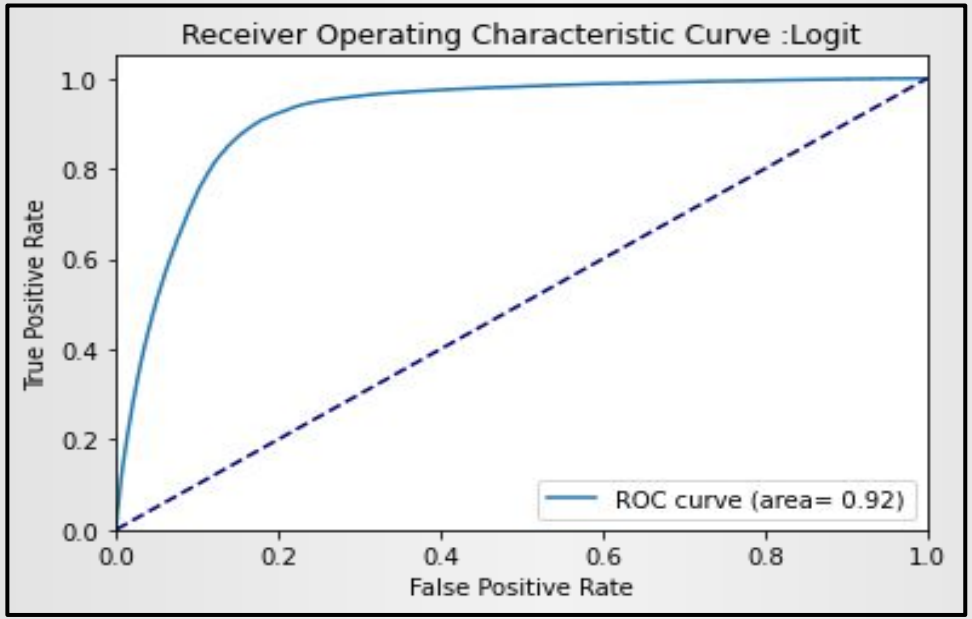
```
fi_col2 = []
fi2 = []

for i, column in enumerate(X):
  fi_col2.append(column)
  fi2.append(rf_model.feature_importances_[i])

fi_df2 = zip(fi_col2, fi2)
fi_df2 = pd.DataFrame(fi_df2, columns=['features', 'feature importance'])
fi_df2 = fi_df2.sort_values('feature importance', ascending=False).reset_index()

idx_to_drop2 = fi_df2.index[fi_df2['feature importance'] == 0].tolist()
fi_df2.drop(index = idx_to_drop2,inplace=True) # feature importance 가 0인 변수들 제거
print(fi_df2.shape) # 113개 중에 111개 생존
fi_df2.head()
```

(111, 3)

	index	features	feature importance
0	84	last_fico_range_high	0.446779
1	85	last_fico_range_low	0.438049
2	83	int_rate	0.046423
3	70	fico_range_high	0.011368
4	71	fico_range_low	0.010357

- (Random Forest) 앙상블 방식을 통해 Decision Tree의 과적합 문제를 해결
- 113개 중 111개의 feature가 0이 아님
  > 무작위성 때문에 Decision Tree보다 많은 feature에 대해 평가할 수 있음.

2) Validation Set을 적용하여 auc & roc curve 확인



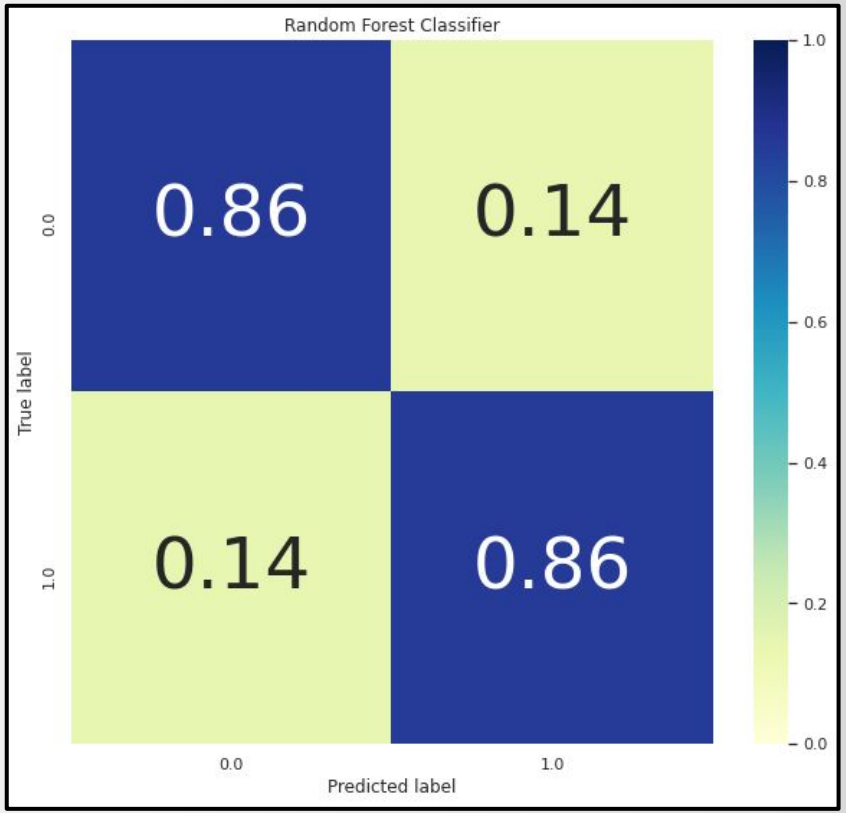
- auc = 0.92
  : 1에 가까울수록 좋은 모델

4) Test set 검증

- optimal_threshold_rf = 0.65
- accuracy : 0.8581735168173333
- bayes decision rule에 따른, 즉 1-fpr+tpr를 극대화하는 threshold(=0.593)보다 높음.

```
cm2 = confusion_matrix(y_test,test_result2['rf_binary'])
cm2
```

```
array([[157390,  25882],
       [  5119,  30193]])
```



결과 해석

- Random Forest 모델을 사용하여 평가하였을 때 Logistic Regression보다 월등히 좋은 예측 성능을 가지고 있음을 알 수 있다.
- class0과 class1을 구분하는 threshold를 **0.65**로 설정했을 때 우리가 목표하는 기대수익률을 최대화 시킬 수 있고, 1종과 2종 오류율 또한 낮게 나오는 것을 확인할 수 있다.
- < 모델 성능 >

  TPR : 85.50%

  FPR : 14.12%

  Precision : 53.84%

  Accuracy : 85.82%

	precision	recall	f1-score	support
0.0	0.97	0.86	0.91	183272
1.0	0.54	0.86	0.66	35312
accuracy			0.86	218584
macro avg	0.75	0.86	0.79	218584
weighted avg	0.90	0.86	0.87	218584

05

결론

conclusion

01. 이전에는 사후변수를 포함하고 부도확률을 예측했다. 그 결과 1종 오류율과 2종 오류율을 모두 적절히 유지하는 logistic regression model을 학습시킬 수 있었다. 그러나 사전변수만으로 학습을 한 결과 feature selection등 여러 조정에도 불구하고 지나치게 높은 2종 오류율을 보였다. 그래서 random forest classifier로 모델을 바꾸었고 보다 나은 결과를 얻었다.

02. 기대수익을 고려하지 않은 경우에 비해 기대수익을 고려했을 때 optimal threshold가 더 높아지는 것을 확인할 수 있었다. 이는 Bayes decision rule에 따라 1종 오류의 면적과 2종 오류의 면적의 합을 최소화하는 경우보다 적극적으로 대출을 해주는 것이 수익을 더 높일 수 있음을 시사한다.

03. 주어지지 않은 데이터(부실고객의 대출금 등)의 uniform 분포를 가정하였다. 차후 정확한 데이터가 추가된다면 앞서 설정한 정확한 수식을 이용함으로써 더 성능 좋은 모델을 가질 수 있을 것이라 기대한다.

THE END

Thank you