Lending Club Predict Loan Payment

Fri. Aug. 25, 2017

Jeon Sohyun

Fast Campus Data Science School

Table of contents

Lending Club Loan 데이터를 이용한 채무상환 예측

- Lending Club은 미국 최대의 P2P회사
- 데이터 출처: https://www.kaggle.com/wendykan/lending-club-loan-data

1

기본적이지만, 의사 결정과 문제 해결에 가장 핵심으로 사용되는

'분류문제'

-물건 구매예측

-대출 상환예측

-마케팅 성공여부

2

1. 대출 상환 여부는 금융권에 있어서 가장 중요한 문제.

2. 부실은 과도한 충당금으로 건전성에 영향을 줄 수 있음.

3. 효과적인 상환예측모델은 비용감소를 기대할 수 있음

3

현재 국내의 금융은 엄격한 규제로 비식별 데이터를 구하기 힘든 상황이기에

Kaggle의 Dataset을 이용.

(Competition은 아님)

1. Summary of Project

1. 2007~2015년까지의 대출 데이터
2. 다양한 상태의 대출 (현재 진행, 완제, 연체 등)

+

Pully Paid: 209,711

Default : 61,176

Total Row : 270,887

Columns : 74

=

Dataset 특징:

1.Nan의 값이 많은 변수가 많음 (현실에서 값을 얻기 힘든 경우)

2. 정제된 데이터가 아니기에, 대출결정 이후에 알게되는 강력한 설명변수들이 많음 -> 제거

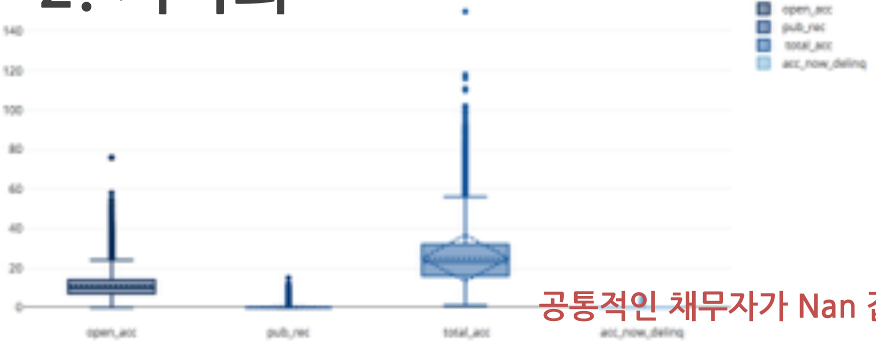
1. 데이터 전처리

```
initial_list_status                  0
collections_12_mths_ex_med          97
mths_since_last_major_derog     164247
policy_code                          0
application_type                     0
annual_inc_joint                203162
dti_joint                       203162
verification_status_joint       203162
acc_now_delinq                      17
tot_coll_amt                     49996
tot_cur_bal                      49996
open_acc_6m                     203056
open_il_6m                      203056
open_il_12m                     203056
open_il_24m                     203056
mths_since_rcnt_il              203059
total_bal_il                    203056
il_util                         203068
open_rv_12m                     203056
open_rv_24m                     203056
max_bal_bc                      203056
all_util                        203056
total_rev_hi_lim                 49996
inq_fi                          203056
total_cu_tl                     203056
inq_last_12m                    203056
loan_status                          0
dtype: int64
```

```
id                                   0
loan_amnt                            0
funded_amnt                          0
funded_amnt_inv                      0
term                                 0
int_rate                             0
installment                          0
grade                                0
sub_grade                            0
emp_title                        11317
emp_length                           0
home_ownership                       0
annual_inc                           1
verification_status                  0
issue_d                              0
pymnt_plan                           0
url                                  0
desc                            133865
purpose                              0
title                               10
zip_code                             0
addr_state                           0
dti                                  0
delinq_2yrs                         17
earliest_cr_line                    17
inq_last_6mths                      17
mths_since_last_delinq          111956
mths_since_last_record          176929
```

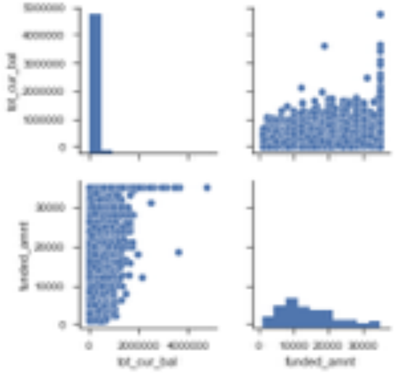
각 변수의 특징을 파악,
추론하여 Nan 값을
평균값, 0, 4분위수 등으로 처리

2. 시각화

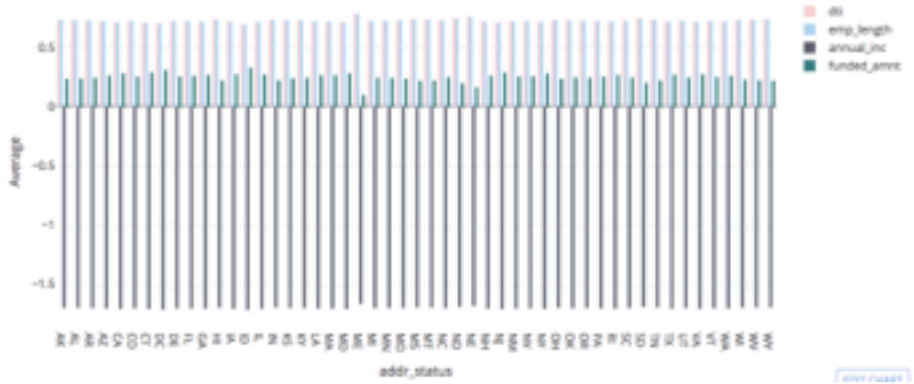


이상치를 파악하여 처리

공통적인 채무자가 Nan 값을 가지는
변수의 분포를 파악하여 값을 추정



두 변수간의 상관관계를 파악



3. 변수 분석 및 생성

```
loan_check.head()
```

Out[78]:

	loan_status	sub_grade	paid	default
0	A1	4291	177
1	A2	4462	274
2	A3	5037	364
3	A4	7768	720
4	A5	8299	978

In [79]:

```
loan_check['grade_percentage'] = loan_check.paid / (loan_check.d
loan_check
```

Out[79]:

	loan_status	sub_grade	paid	default	grade_percentage
0	A1	4291	177	0.960385
1	A2	4462	274	0.942145
2	A3	5037	364	0.932605
3	A4	7768	720	0.915174
4	A5	8299	978	0.894578
5	B1	8202	1129	0.879005
6	B2	9739	1556	0.862480

Stringtype의 신용등급에 대한 가중치 계산하여 변수값 할당

	emp_length	annual_inc	norm
0	0	47969.955389	0.205355
1	1	67266.308913	0.287961
2	2	70008.602745	0.299701
3	3	70812.048165	0.303140
4	4	69765.991236	0.298662
5	5	71294.267054	0.305205
6	6	71431.280815	0.305791
7	7	72895.635531	0.312060
8	8	74238.551182	0.317809
9	9	74638.035644	0.319519
10	10	79853.937696	0.341848

재직기간에 따라 One-hot-encoding 외에 연소득과의 관계로 가중치 할당

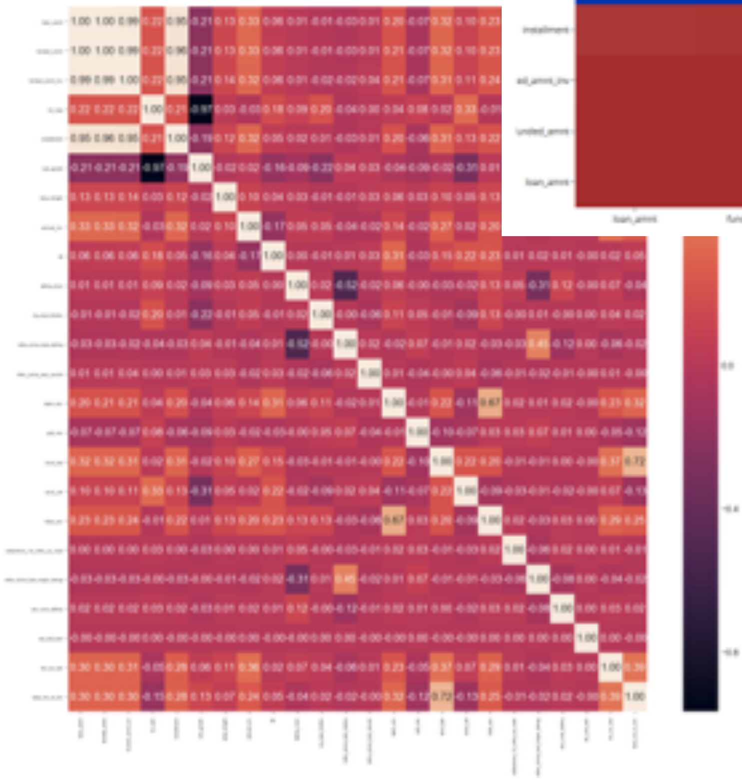
	issue_d	earliest_cr_line	credit_period
0	24124	23983	141
1	24164	24008	156
2	24158	24003	155
3	24161	23883	278
4	24158	24000	158
5	24184	23982	202
6	24156	24000	156
7	24176	23997	179
8	24170	24005	165
9	24154	23743	411
10	24185	24006	179

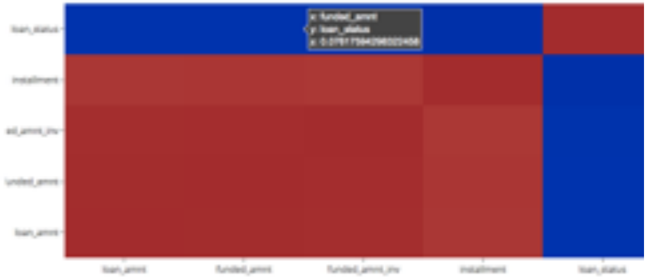
최초 신용개설일과 대출시작시점을 계산하여 신용 기간 추정

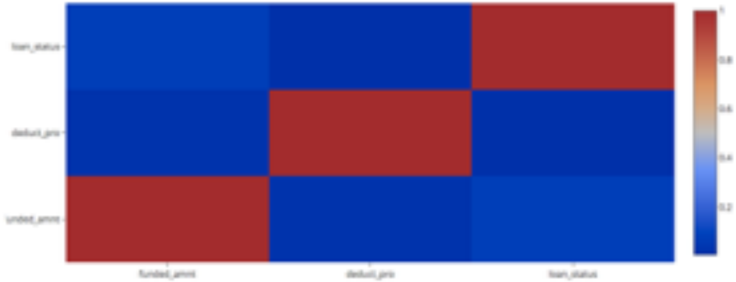
그 이외의 변수 생성과 질적변수의 더미화

columns : 74 -> 142

1.다중 공선성 확인







loan_amnt - funded_amnt :1.0
loan_amnt - funded_amnt_inv : 0.99
loan_amnt - installment : 0.95

funded_amnt - funded_amnt_inv : 0.99
funded_amnt - installment : 0.95

funded_amnt_inv - installment : 0.95

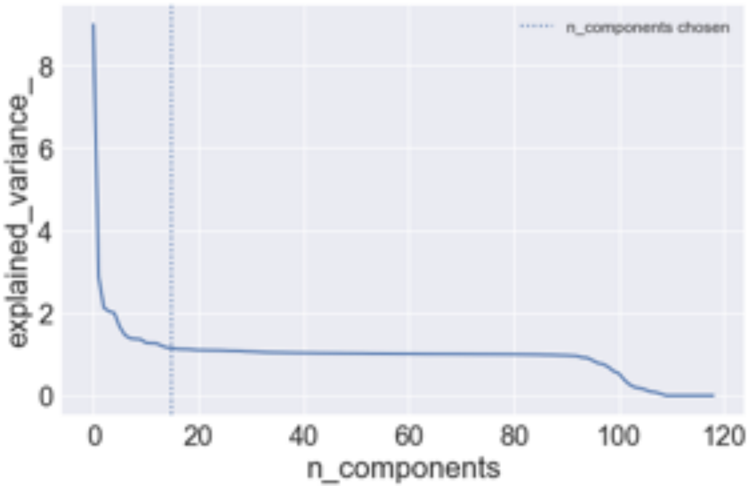
open_acc - total_acc : 0.67
revol_bal - total_rev_hi_lim : 0.72

mths_since_last_delinq -
mths_since_last_major_derog : 0.45

상관관계가 큰 변수는 제거

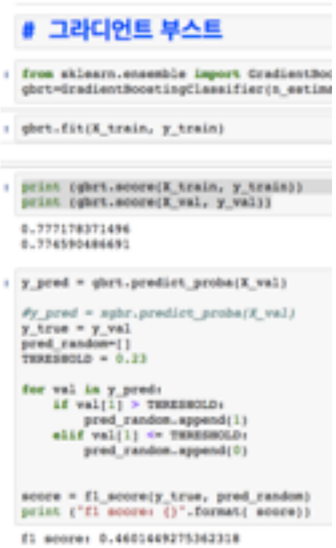
2. PCA

- 0의 값이 전체의 80%를 초과하는 변수와 더미화한 변수를 PCA로 차원축소 한다 (과적합 우려)

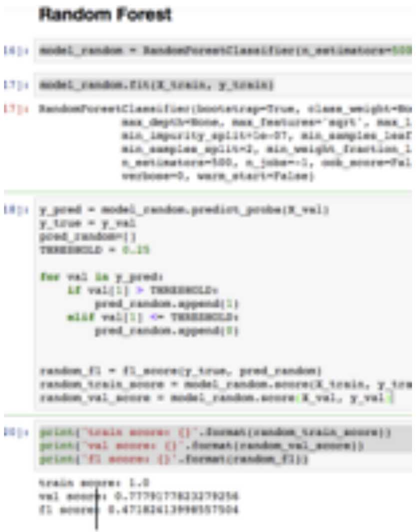


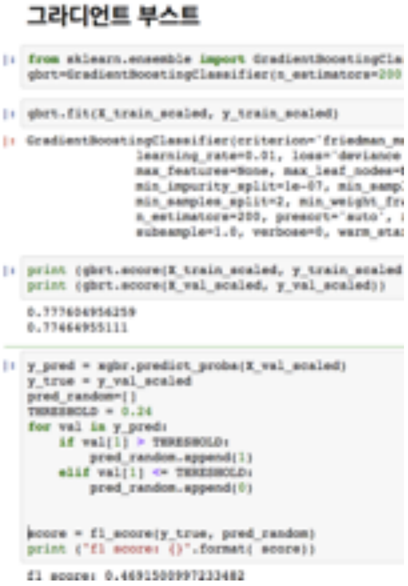
119 ▷ 15

3. 결과



다중공선성 변수제외, PCA 적용 후





다중공선성 변수제외, PCA 적용 전

적용전이 적용후 보다 점수가 좀더 높은 경향으로 보여 원 데이터를 살리기로 함

4. Modelling

1. y값의 비대칭 문제 발견

Model	scaled	hyperparameter	threshold	train score	val score	F1 score
Random Forest	x	max_features='sqrt', n_estimators=500, n_jobs=-1, random_state=20	0.25	1.0	0.776	0.469
	x	n_jobs=-1, max_features='sqrt', random_state=20, n_estimators=200	0.25	0.944	0.943	0.8298
	x	n_jobs=-1, max_features='sqrt', random_state=20, n_estimators=200	0.32	0.944	0.945	0.85325
	x	n_jobs=-1, max_features='sqrt', random_state=20, n_estimators=200	0.42	0.944	0.944	0.8648
	x	n_jobs=-1, max_features='sqrt', random_state=10, n_estimators=200	0.42	0.944	0.944	0.8646

하이퍼 파라미터를 튜닝하면서 획기적으로 높은 스코어를 얻음

하지만 Test set을 적용 시 F1 Score가 0.21이 되는 기형적 현상을 발견

〈Training Set〉

Pully Paid: 157,229

Default : 45,936

상환과 부채의 비율이 7:3정도로

모델이 불균형데이터에 학습,

적절한 예측을 하지 못함

2. 해결책

(2) Dataframe sampling and shuffle

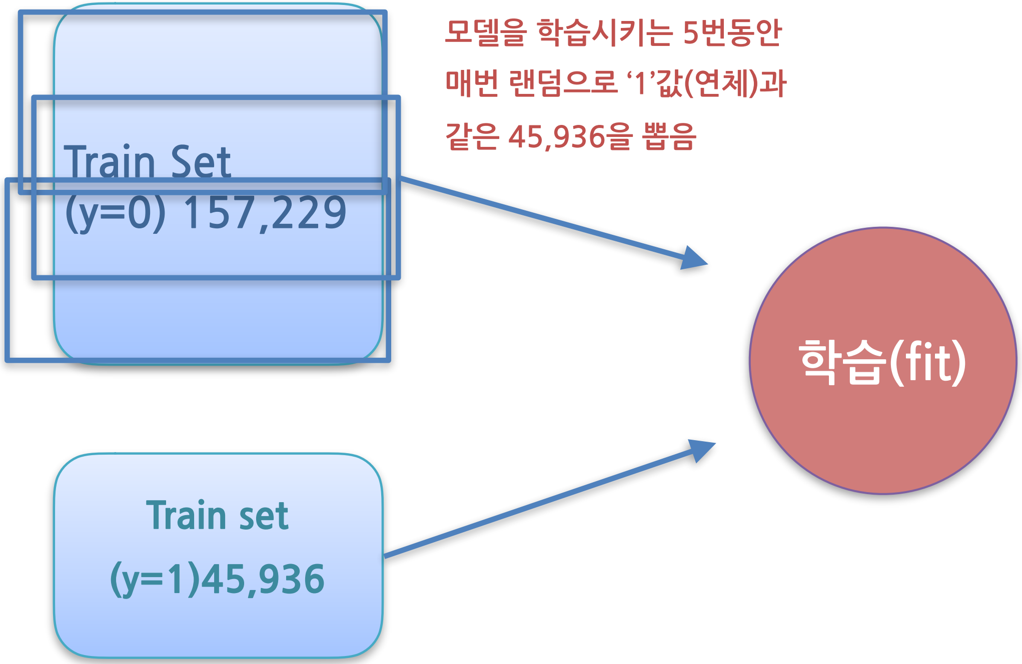
```
def shuffle_data(X_0, X_1): # select same number of sample with X_1 from X_0, and shuffle

    X_sample = X_0.sample(n=45936)

    X = pd.concat([X_sample, X_1])

    X = X.sample(frac=1).reset_index(drop=True)

    y = X['loan_status']
    X = X.drop('loan_status', axis=1)

    return X, y
```

Shuffle 함수

(3) Cross Validation

```
def cross(model, X_0, X_1, thres):

    X, y = shuffle_data(X_0, X_1)

    train_list = []
    test_list = []
    f1_list =[]
    auc_list=[]

    # 5 times modeling and take score
    for i in range(5):
        X_train, X_val, y_train, y_val = train_test_split(X, y)

        # Training
        fitted_model = model.fit(X_train, y_train)

        # accuracy
        train_list.append(model.score(X_train, y_train))
        test_list.append(model.score(X_val, y_val))

        # AUC $ ROC
        auc_list.append(roc_auc_score(y_val, model.predict(X_val)))
```

Cross Validation

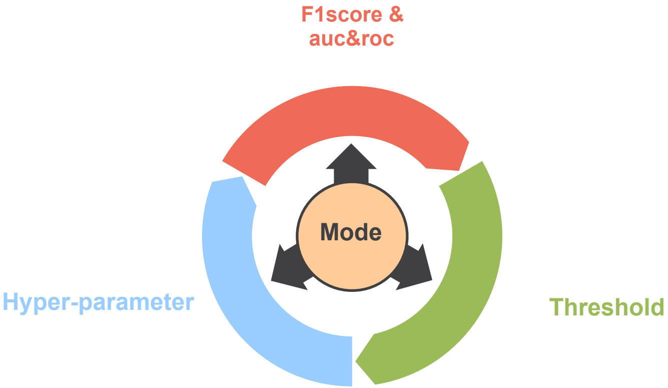


모델을 학습시키는 5번동안
매번 랜덤으로 '1'값(연체)과
같은 45,936을 뽑음

4. Modelling

3. Hyper Parameter Tunning

- Grid Search를 이용하여 자동 Hyper Parameter를 뽑아 보았으나, 수기로 조정 시 보다 Performance가 좋지 않음을 확인함



4. Model Selection

3-1) Random Forest

*Predicting overfiting, tuning hypter parameter

```
model_random2 = RandomForestClassifier( random_state=2,
```

```
model_random_final= cross(model_random2, X_0, X_1, 0.35
```

```
train score : [1.0, 1.0, 1.0, 1.0, 1.0] & mean : 1.0
test score: [0.66457680250783702, 0.66004876349703934,
  & mean : 0.6616509926854756
f1 score : [0.70339495798319329, 0.70520542413943932, 0
  & mean : 0.6997029222750076
auc&roc score : [0.66452068899785643,
301] & mean : 0.6616537758496717
```

〈Random Forest〉

F1: 0.699

auc&roc: 0.661

3-2) GradientBoosting

- Gradient Boosting takes too long time, gridsearch

Non-scaled

```
model_gradient= GradientBoostingClassifier(n_es
```

```
model_gradient_final= cross(model_gradient
```

〈Gradient Boosting〉

F1: 0.690

auc&roc: 0.658

3-3) Neural network models

random_state(0),alpha(0.0001 -> 1) :weight to be strong

```
model_mlp = MLPClassifier(alpha=0.0001,max_
#model_mlp.fit(X_train, y_train)
```

```
model_nn_final= cross(model_mlp, X_0, X_1,
```

〈NN〉

F1: 0.637

auc&roc: 0.555

3-4) XG Boost

```
xgbr = xgb.XGBClassifier (n_estimators=30,
```

```
X_0.sort_index(axis=1, inplace = True)
X_1.sort_index(axis=1, inplace = True)
X_test.sort_index(axis=1,inplace=True)
```

```
model_xg_final, X_train_xg= cross(xgbr, X_0,
```

〈XGBoost〉

F1: 0.696

auc&roc: 0.652

3-5) Light gbm

```
gbm = lgb.LGBMClassifier(objective='regression
                         num_leaves=200,
                         learning_rate=0.001,
                         n_estimators=3000,
                         # early_stopping_roun
                         )
```

```
model_gbm_final = cross(gbm, X_0, X_1, 0.36)
```

〈LightGBM〉

F1: 0.698

auc&roc: 0.657

속도와 성능을 고려 시, LightGBM이 적절할 것으로 판단

1. Test set Score

1) Random Forest ¶

```
test(model_random_final, X_test, y_test, 0.35)
```

```
test score: [illegible]
auc&roc score : 0.656774332377025
f1 score : 0.42354378196800892
```

2) Gradient Boosting

```
test(model_gradient_final, X_test, y_test, 0.25)
```

```
test score: 0.6570095389976669
auc&roc score : 0.6582475146336387
f1 score : 0.3997114978974729
```

3) Neural network model

```
test(model_nn_final, X_test, y_test, 0.33)
```

```
test score: 0.6462597088095449
auc&roc score : 0.5806297067130928
f1 score : 0.37103221255584296
```

4) XGBoost

```
test(model_xg_final, X_test, y_test, 0.33)
```

```
test score: 0.654957030211748
auc&roc score : 0.6550374639133867
f1 score : 0.42015710776136606
```

5) LightGBM

```
test(model_gbm_final, X_test, y_test, 0.36)
```

```
test score: 0.6529045214258291
auc&roc score : 0.6562043019173878
f1 score : 0.4350461047212538
```

Test set에서도 가장 좋은
점수가 나옴을 알 수 있음

Main Hyper Parameter

num_leaves=200,
earning_rate=0.001,
n_estimators=3000,

상위 주요 변수 5위

annual_inc	0.073342
dti	0.072496
revol_util	0.066707
credit_period	0.058479
tot_cur_bal	0.057720

2. Conclusion

하버드대학의 Dr. Asim Khwaja의 최근 논문에 따르면, 금융 정보 이외의 비금융 정보 (신청서 작성 시 자필 및 맞춤법, 통신 및 보험 정보, SNS의 평판 등)를 가지고 좀 더 정확하고 강력한 대출 상환 예측을 할 수 있다는 연구가 있었다.

현재 국내에서는 다양한 규제로 인해 비식별 데이터를 얻기 힘든 실정이긴 하지만 이런 비금융 정보를 이용하여 효과적인 모형을 만들어 보고 싶다

Thank You