Lending Club Predict Loan Payment

Fri. Aug. 25, 2017

Jeon Sohyun

Fast Campus
Data Science School

Table of contents

- 1. 프로젝트 요약
- 2. 전처리, 시각화, 변수 생성

3. 다중 공선성, PCA

4. 모델링

5. 결론

1. Summary of Project

Lending Club Loan 데이터를 이용한 채무상환 예측

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

기본적이지만, 의사 결정과 문제 해결에 가장 핵심으로 사용되는

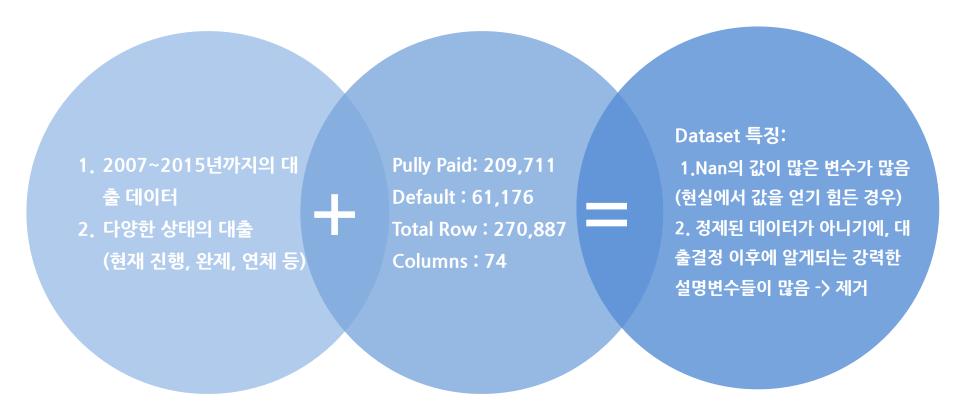
'분류문제'

- -물건 구매예측
- -대출 상환예측
- -마케팅 성공여부

- 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

현재 국내의 금융은 엄격한 규제로 비식별 데이터를 구하기 힘든 상황이기에 Kaggle의 Dataset을 이용. (Competition은 아님)

1. Summary of Project



2. Preprocessing, Visualize

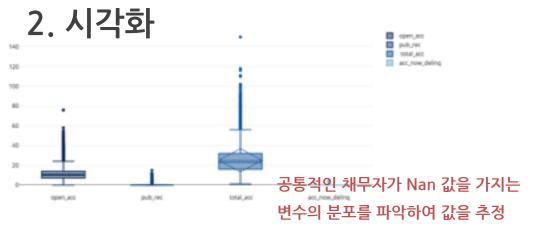
1. 데이터 전처리

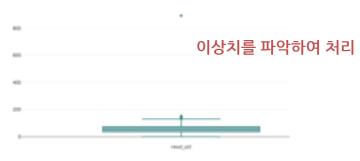
initial_list_status	0	id	0
collections_12_mths_ex_med	97	loan_amnt	0
mths_since_last_major_derog	164247	funded_amnt	0
policy_code	0	funded_amnt_inv	0
application_type	0	term	0
annual inc joint	203162	int_rate	0
dti joint	203162	installment	0
verification status joint	203162	grade	0
acc now deling	17	sub_grade	0
tot_coll_amt	49996	emp_title	11317
tot_cur_bal	49996	emp_length	0
open acc 6m	203056	home_ownership	0
open il 6m	203056	annual_inc	1
open il 12m	203056	verification_status	0
open il 24m	203056	issue_d	0
mths since rent il	203059	pymnt_plan	0
total bal il	203056	url	0
il_util	203068	desc	133865
_	203056	purpose	0
open_rv_12m		title	10
open_rv_24m	203056	zip_code	0
max_bal_bc	203056	addr_state	0
all_util	203056	dti	0
total_rev_hi_lim	49996	delinq_2yrs	17
inq_fi	203056	earliest or line	17
total_cu_tl	203056	ing last 6mths	17
inq_last_12m	203056	mths since last deling	111956
loan_status	0	mths_since_last_record	176929
dtype: int64			

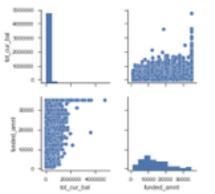


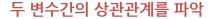
각 변수의 특징을 파악, 추론하여 Nan 값을 평균값, 0, 4분위수 등으로 처리

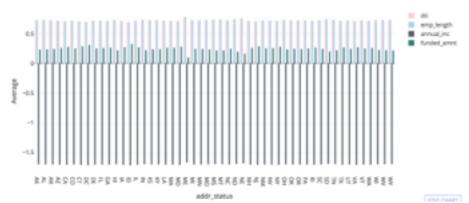
2. Preprocessing, Visualize





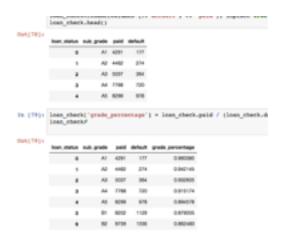






2. Preprocessing, Visualize

3. 변수 분석 및 생성



Stringtype의 신용등급에 대한 가중치계산하여 변수값 할당

ユ	0	오	의	변	수	생	성	과
질	적	벼.	수의	2	더디	기호	<u>}</u>	

드 니 다 나 시 시 시 기 가

norm	annual_inc	emp_length	
0.205355	47968.955389	0	0
0.287961	67266.308913	1	1
0.299701	70008.602745	2	2
0.303140	70812.048165	3	э
0.298662	68765.991235	4	4
0.305205	71294.267054	5	5
0.305791	71431.200015	6	
0.312060	72895.635531	7	7
0.317809	74238.551182	8	8
0.319519	74638.035644	9	
0.341848	79853:937696	10	10

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

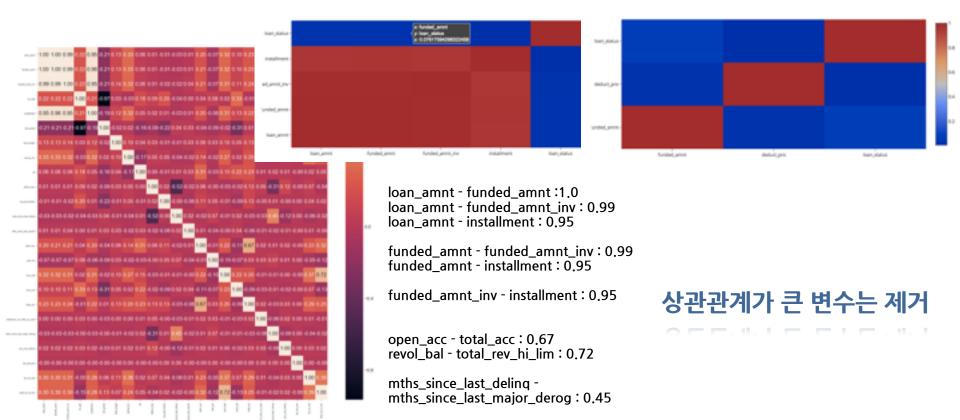
	issue_d	earliest or line	credit period
0	24124	29983	141
1	24164	24008	156
2	24158	24003	155
3	24161	23883	278
4	24158	24000	158
5	24184	23982	202
6	24156	24000	156
7	24176	23997	179
8	24170	24005	165
9	24154	23743	411
0	24185	24006	179

최초 신용개설일과 대출시작시점을 계산 하여 신용 기간 추정

columns: 74 -> 142

3. Correlation, PCA

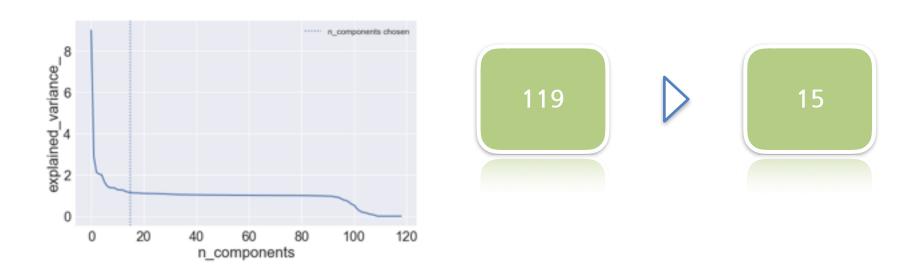
1.다중 공선성 확인



3. Correlation, PCA

2. PCA

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



3. Correlation, PCA

3. 결과

```
그라디언트 부스트
: from sklearn.ensemble import GradientBoo
  gbrt-GradientBoostingClassifier(n estima
| gbrt.fit(% train, y train)
print (gbrt.score(K train, y train))
  print (ghrt.score(K_wal, y_wal))
  0.777178371496
  0.774590486693
: y pred = qbrt.predict.proba(X val)
  #y pred = spbr.predict proba(% val)
  y true - y val
  pred random-[]
  TERESHOLD = 0.23
  for val in y_pred:
      if wal[1] > TERRESHOLD:
         pred random, append(1)
      alif val(1) - THRESHOLD:
         pred random_append(0)
  score = fl score(y true, pred random)
  print ("fl score: ()".format( score))
  fl moore: 0.4601449275362318
```

```
Light gbm.
123): gbm = 1gb.LGBMRegressor(objectiv
                               Searning
                               n_estime
124): gbm.fit(X_train, y_train,
              eval set "[(K_val, y_val)
               eval metric-'11',
              early stopping rounds-1
125]: print (gbm.score(K_train, y_tra)
      print (gbm.score(K wal, y wal))
      0.131356963125
      0.105339628872
176): y pred = qbm.predict(X wal, num
      y true = y val
      pred_random=[]
      THRESHOLD = 0.24
       for val in y_pred:
          if val > TERRESHOLD-
              pred_random.append(1)
          elif val <- THRESHOLD:
              pred random.append(0)
       score - fl score/y true, pred re
      print ("fl score: ()".format( sc
       fl score: 0.46600651713119684
```

```
Random Forest
(4): model random - HandomForestClassifier;n estimators-100
[7]: model random, fit(X train, y train)
[7]: RandomForestClassifier(bootstrap-True, class_weight-No
                 max depth-Bone, max features-'sgrt', max 1
                 min impurity split+le-07, min samples leaf
                 min sumples splits2, min weight fraction 1
                 n setimators-500, n jobs--1, ook score-Fall
                 verbose-0, warm start-Palse)
183: y pred - model random, predict probe(X val)
    y_true - y_vel
     pred random-()
     THRESHOLD = 0.15
     for val in y pred-
         of val(1) > THRESHOLDS
            pred random.append(1)
         elif val[1] - THRESHOLD:
             pred random, append(T)
     random fi = fi_scorecy_true, pred_random)
     random train score - model random.score(X train, y tra
     random val score = model random.score X val, y val
(0): print("train moore: ()".format(random train_moore))
     ((woons for motors) () .format(random val accord))
     print("fl score: ()'-format(random_fl))
     train soore: 1.0
     val seems: 0.7779077823279256
     fl scores 0.47182413998557504
```

그라디언트 부스트

```
| | from sklears.ensemble import GradientBoostingCla
   gbrt=GradientBoostingClassifier(n_estimators=200
prt.fit(X_train_scaled, y_train_scaled)
|| GradientBoostingClassifier(criterion='friedman_magnets)
                 learning rate=0.01, loss="deviance
                 max features-None, max leaf nodes-t
                 min impurity split-le-07, min samp.
                 min samples split=2, min weight fro
                 n estimators-200, presort-'suto', i
                 subeample=1.0, verbose=0, warm sta:
| | print (ghrt.score(X train scaled, y train scaled
   print (gbrt.score(K val scaled, y val scaled))
   0.777604954259
  0,77464955111
| y pred = mgbr.predict_proba(X_val_scaled)
   y true = y val scaled
   pred random-[]
   THRESHOLD = 0.24
   for val is y pred:
       if wal(1) > THRESHOLD:
           pred random.append(1)
       elif wal[1] <- TERRESHOLD:
           pred_random.append(0)
   score = fl_score(y_true, pred_random)
   print ("fl score: ()".format( score))
```

fl score: 0.4691500997233482

다중공선성 변수제외, PCA 적용 전

다중공선성 변수제외, PCA 적용 후

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

1. y값의 비대칭 문제 발견

Model	scaled	hyperparameter	threshold	train score	val score	F1 score
Random Forest		max_features='sqrf; n_estimators=500; n_abbe=-1; random_state=20	0.25	1.0	0.776	0.469
		n_jobs=-1, max_features='sgrt', random_state=20, n_estimators=200	0.25	0.944	0.943	0.8298
		n.jobs=-1, max.features="sqrt; random_state=20, n.estimators=200	0.32	0.944	0.945	0.85325
	×	n_jobs=1, max_features='sqrf', random_state=20, n_estimators=200	0.42	0.944	0.944	0.8648
	*	n_jobs=-1, max_features='sqrf', 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

Pully Paid: 157,229

Default: 45,936

상환과 부채의 비율이 7:3정도로 모델이 불균형데이터에 학습, 적절한 예측을 하지 못함

적절한 예측을 하지 못함

2. 해결책

(2) Dataframe sampling and shuffle

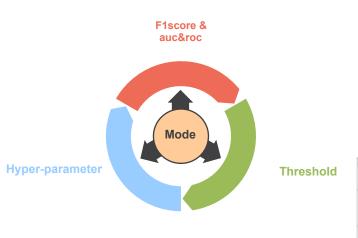
(3) Cross Validation

```
def cross(model, X_0, X_1, thres):
                                               Cross Validation
     X, y = shuffle data(X 0, X 1)
     train list - []
     test list = []
     fl list =[]
     aue 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)
         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)))
```



3. Hyper Parameter Tunning

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



Gradient Booting	n,extmators-200, learning,rate-	0.05, max, (kg/th-5	0.3	0.68788		66005	0.68	100			
	n_extimators=500), learning_rate=0.81, max_(legith=0. 		0.3	0.65649	0	69041	0.68	09			
			0.3	0.6344218		0.6301	0.6676	164			
			0.3	0.64300		0.6363	0.6655	89			
Envelope and	alpha-0.0001.max.iter- 300.	Random Forest	njobni i mex. njedmetoro di	Seatures "logit", random, 00	state=20.		0.42	10	0.0544	0.691	
Neural network	hidden, layer, sizen-2003			Seatureer'sgrt', random,	state=20).		0.29	10	0.65485	0.69167	
	alpha-0.0001,max,itan=100, helden,layer,spen=200		n, artimators-500 n jobic – 1, max flutures rispit, random, state-2,			0.29 14		10	0.6563	0.6905	
	diphor-0:0001,max, kerr 500, hidden (over scien-300)		numbered may		mess?					0.69706	
	alpha=0.0001,max, item 500.		n_obs-1, max_fautures-'sgrt', random_state=2, n_estimatory-200			0.36 1.0		0.00033	0.00033 0.00706		
	hidden, layer , sizes -1001 alpha=0.5001, max, item 900, hidden, layer , sizes -300		nunber-1, max. nunber-2	features* log2", random, 00	state=20.		0.33	10	0.68812	0.69604	
			random,state=2	70, n,estimators=200			0.30	10	0.6528	0.694763	
Light GBM	Objectiver ingression, num, lauren = 60, learning ; rain-0-000, n_sellmeton=1000 objectiver ingression, num, leavest=40, learning ; rain-0-000, n, sellmeton=2000, n, sellmeton=2000,		random,staten) max,features/)), nuestimutory-501), nuje kpří	der-I.		0.30	10	0.654536	0.69197	045491
			random, state+) max, features+)	l, n, erometon-50to, n, jo spri	der-1.		0.36	10	04558	0.698557	04607
			random, state-0 max_featureo-1 ,solb_score = TI		84*-1.		0.16	10	0.60009	0.70054	0.00051
	objective-ingressioni, learning_sate=0.000, n_extensions=0.000,			0.00546	0	5481	0.6831				
	objective-Ingressioni, num, leaves-ridit, learning, rate-0:001, n, estimators-0000,		0.36	0.676600	040	1445	0.69629	0.63	800		
	objectives* (agression), num_bases*50; laurang_cata+0:001, n_estimalar+1000, objectives* (agression), num_bases**000, laurang_cata+0:001, n_estimalar+1000,		0.36	0.000000	0.60	moo	0.69601	0.60	ine.		
			0.36	0.70343	0.64	1068	0.70036	0.00	047		
	objective 'ingression', hum javes-200, learning rate-0.001, n autimators-2000,		0.36	0.74348		9621	0.70188	2.06	200		

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
 fl score : (0.70339495798319329, 0.70520542413943932, 0
  & mean : 0.6997029222750076
                                       〈Random Forest〉
  auciroc score : [0.66452068899785643,
 3011 & mean : 0.6616537758496717
```

3-2) GradientBoosting

F1: 0.699

Gradient Boosting takes too long time, gridsearch v

auc&roc: 0.661

Non-scaled model_gradient= GradientBoostingClassifier(n_ec

model gradient final- cross(model gradien 〈Gradient Boosting〉 3] & mean : 0.6876668988737954 test score: [0.66044061302681989, 0.65726] F1: 0.690

4 mean : 0.6586903517938001 fl score : [0.69061913118438456, 0.6888866 mean : 0.6902023225046221

#UCEFOC #COT# 1 10.66036652730344669, 0.61 auc&roc: 0.658 443] 4 mean + 0.6586488693910831

3-3) Neural network models

```
random_states(0),alpha(0.0001 -> 1) :weight to be stronge
model_mlp = MLSClassifier(alpha=0.0001,max
fmodel mlp.fit(X train, y train)
model as final- cross(model mlp, x 0, x 1,
train score : [0.55557006850110302, 0.53250
91 4 mean : 0,5573725763380937
test score: [0.55725357018460464. 0.5307384
  & mean : 0.55698362939
 s mean : 0,637768114724
auckroc score : [0.55447
014] & mean : 0.55569437 F1: 0.637
```

auc&roc: 0.555

3-4) XG Boost

```
agbr - mgb.MiBClassifler (n_estimators-10, ga
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_mg_final, X_train_mg= cross(mgbr, X_0
                                           ⟨XGBoost⟩
train score : [0.76471612678509227, 0.762981
1) a mean + 0.7643881342157204
                                           F1: 0.696
 4 mean : 0.6521159874608151
fl score : [0.70075858397657698, 0.696215205
```

sucaroc score : [0.65269277339022636, 0.6484

1431 a mean : 0.6521130617535718

3-5) Light gbm

```
gbm = lgb.LGBMClassifier(objective='regression
                       num_leaves=200,
                        learning rate=0.001,
                       n_estimators=3000,
                        # early stopping roun
  lel gbm final = cross(gbm, X 0, X 1, 0.36)
  in score : [0.74336758388482527, 0.7445286]
  4 mean : 0.7437826541274818
  t acore: [0.65952629745733193, 0.6578282826
  man : 0.6574538488331592
  score : [0.69997243660418951, 0.69800559952
  mean | 0.6988261990347963
  aroc score : [0.65956065090604821, 0.657834
  1 4 mean + 0.65746417
```

(LightGBM)

F1: 0 698

auc&roc: 0.657

속도와 성능을 고려 시. LightGBM이 적절할 것으로 판단

auc&roc: 0.652

5. Conclusion

fl score : 0.4350461047212538

1. Test set Score

1) Random Forest 1 test(model_random_final, X_test, y_test, 0.35) aucaroc score : 0.656774332377025 fl score : 0.4235437819680892 2) Gradient Boosting test(model_gradient_final, X_test, y_test, 0.25) test score: 0.6570095389976669 auciroc score : 0.6582475146334387 fl score : 0.3997114978974729 3) Neural network model test(model_nn_final, X_test, y_test, 0.33) test score: 0.6462597888095449 auciroc score : 0.5806297067130928 fl score : 0.37103221255584296 Test set에서도 가장 좋은 4) XGBoost 점수가 나옴을 알 수 있음 test(model_xg_final, X_test, y_test, 0.33) test score: 0.654957030211748 auciroc score : 0.6550374639133867 fl score : 0.42015710776136606 LightGBM test(model_gbm_final, X_test, y_test, 0

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

tot_cnr_pal 0.022250

5. Conclusion

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

16

Thank You