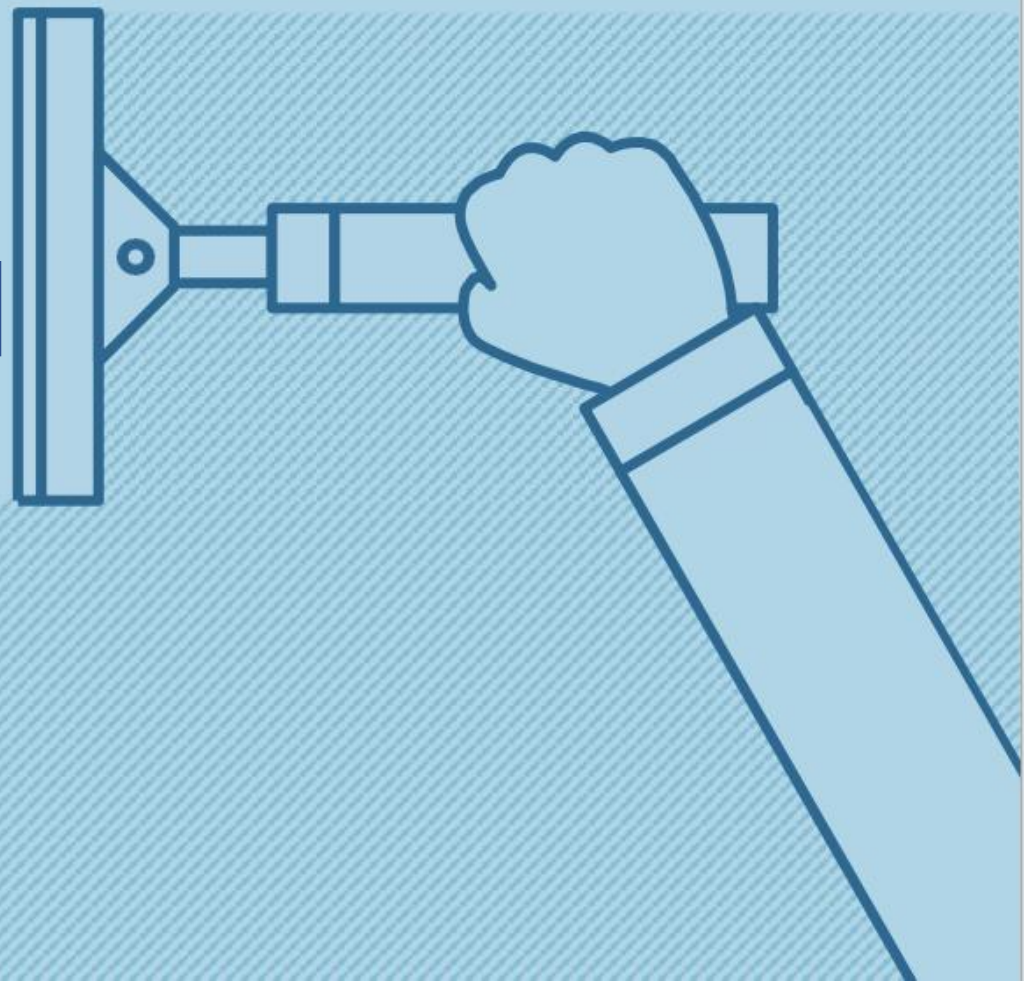
프로젝트 기말발표

3팀 강소영, 백진우, 우현수, 정현우, 정희영



CONTENTS

1

전처리 과정

PCA, Null 값 처리
및 column 재구성

2

사용 모델 및 성능평가

사용 모델 선정 및 성능

데이터 전처리

Null 값 처리 및 column 재구성

1. 전처리 과정

처음 사용했던 columns

원 칼럼 그대로	변형한 칼럼	PCA	새로 추가
isFraud	addr1_na	V_columns	C_fraud
TransactionDT	addr2_count	email_pca1	hour
TransactionAmt	addr2_fraud_ratio	email_pca2	repeated
	dist1_na	id_columns	TransactionAmt_residue
	dist2_na		
	TransactionAmt_log		

1. 전처리 과정

addr1_na

addr1	Percent	num
272.0	0.000000	1
348.0	0.071429	84
356.0	0.133333	90
426.0	0.156250	32
536.0	0.161812	309
171.0	0.166667	12
479.0	0.230769	13
432.0	0.289474	38

Addr1은 국가의 세부 지역

➡ 지역마다 사기율이 전부 다름

특정 지역에 사기거래가 몰림

Binary column 추가

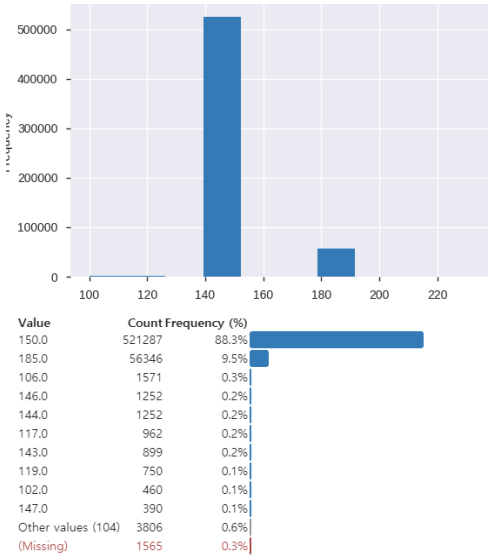
➡ 더불어 NA의 값들이 사기거래에서 빈번하게 나타남

이를 반영하고자 addr1_na column을 추가했다

1. 전처리 과정

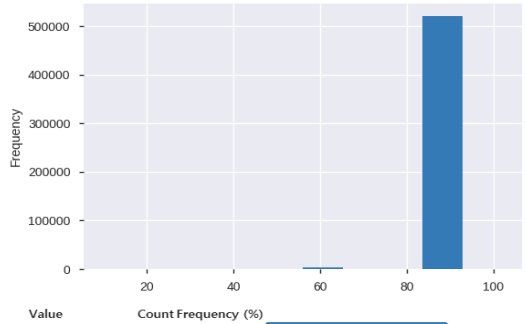
addr2_fraud_ratio / addr2_count

card3와 addr2의 분포가 가장 유사



Value	Count	Frequency (%)
150.0	521287	88.3%
185.0	56346	9.5%
106.0	1571	0.3%
146.0	1252	0.2%
144.0	1252	0.2%
117.0	962	0.2%
143.0	899	0.2%
119.0	750	0.1%
102.0	460	0.1%
147.0	390	0.1%
Other values (104)	3806	0.6%
(Missing)	1565	0.3%

card3

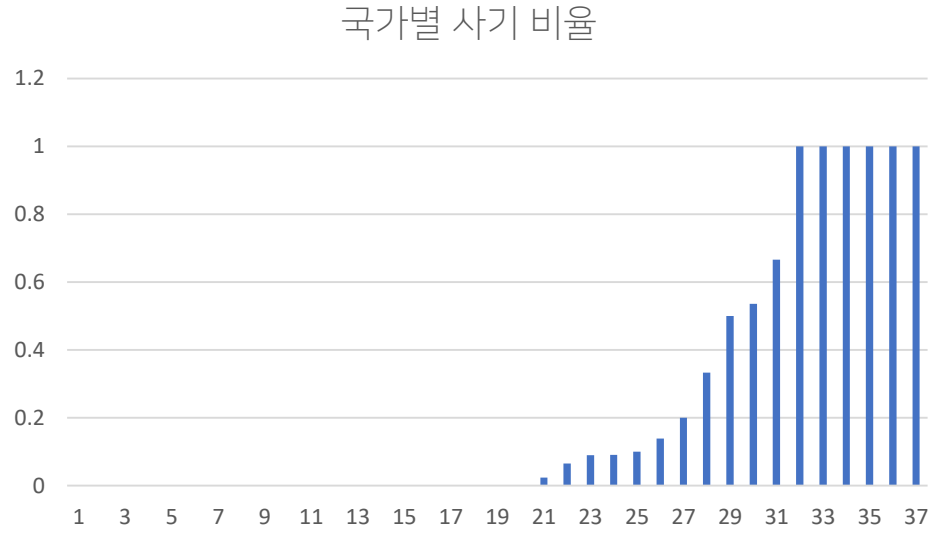


Value	Count	Frequency (%)
87.0	520481	88.1%
60.0	3084	0.5%
96.0	638	0.1%
32.0	91	0.0%
65.0	82	0.0%
16.0	55	0.0%
31.0	47	0.0%
19.0	33	0.0%
26.0	25	0.0%
27.0	20	0.0%
Other values (64)	278	0.0%
(Missing)	65706	11.1%

addr2

➡ card3, addr2 = 국가 columns

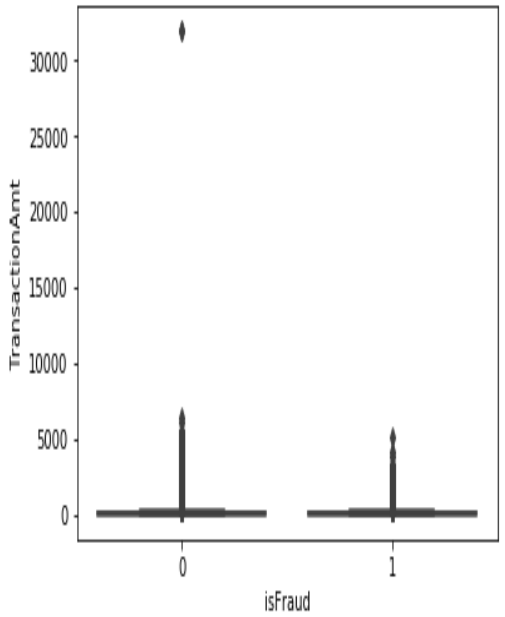
특정 국가에서 높은 사기비율



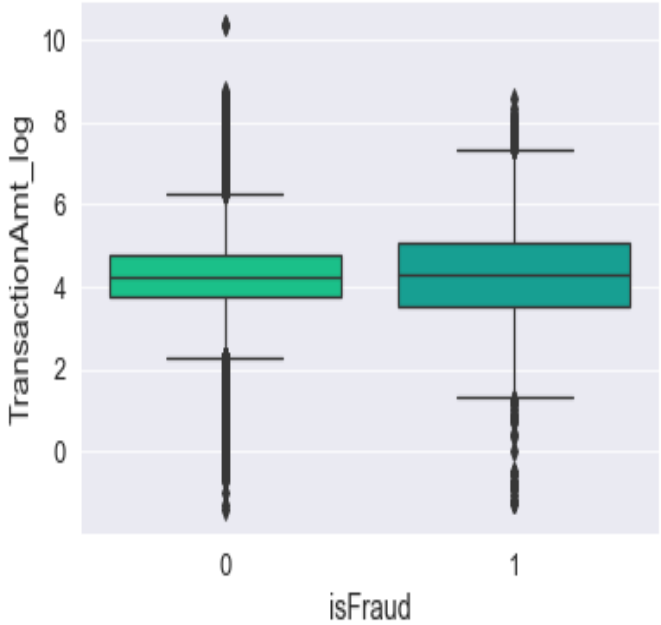
➡ ratio 칼럼, 빈도를 나타내주는 count 칼럼도 추가

1. 전처리 과정

TransactionAmt_log



기존 Amt 분포는
심하게 skewed 됨



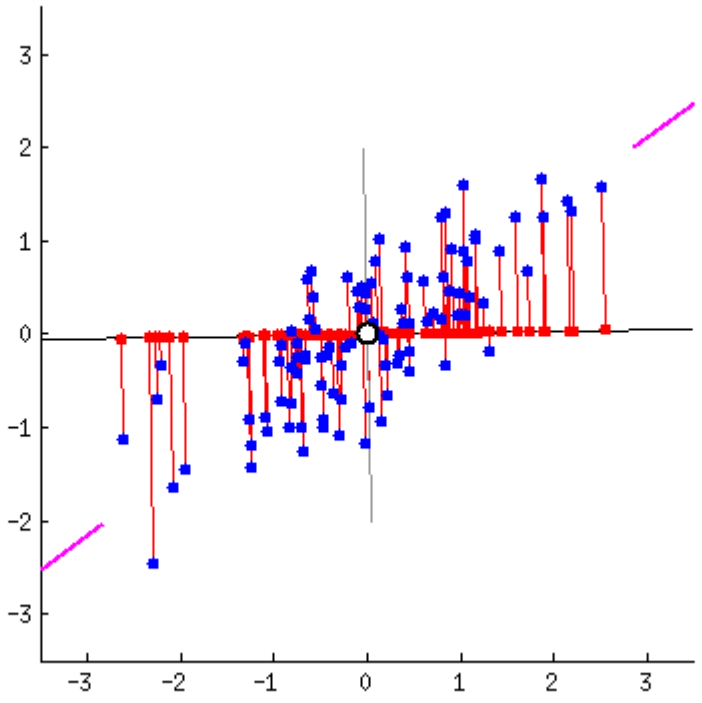
outlier 제거 후
Log scale로 변환

dist1_na / dist2_na

Null 값들이 너무 많아서 Binary로 만들어줌

1. 전처리 과정

PCA 진행한 columns



- Emaildomain
- V_columns
- Id_columns
- C-columns

- PCA를 사용한 이유

  PCA 사용한 칼럼들이 어떤 column인지 모르는 상황이어서

  PCA를 사용해도 괜찮겠다는 판단에 차원을 축소시킴.

1. 전처리 과정

TransactionAmt_residue

사기인 경우의 거래금액에 소숫점 이하 15자리 이상으로 나오는 경우가 많음

거래 금액이 소숫점 이하 15자리 이상인 경우 1, 아닌 경우 0

1. 전처리 과정

hour

```
In [0]: temp["TransactionDT"] = temp["TransactionDT"]/24/3600

In [0]: temp["day"] = temp["TransactionDT"]//1

In [0]: temp["hour"] = temp["TransactionDT"]%1*24//1

In [0]: temp["min"] = (temp["TransactionDT"]%1*24)%1*60//1

In [0]: temp["dayW"]=temp["day"]%7
```

➡ TransactionDT로 hour를 계산함

각 국가별로 시간별 사기 비율이 다름

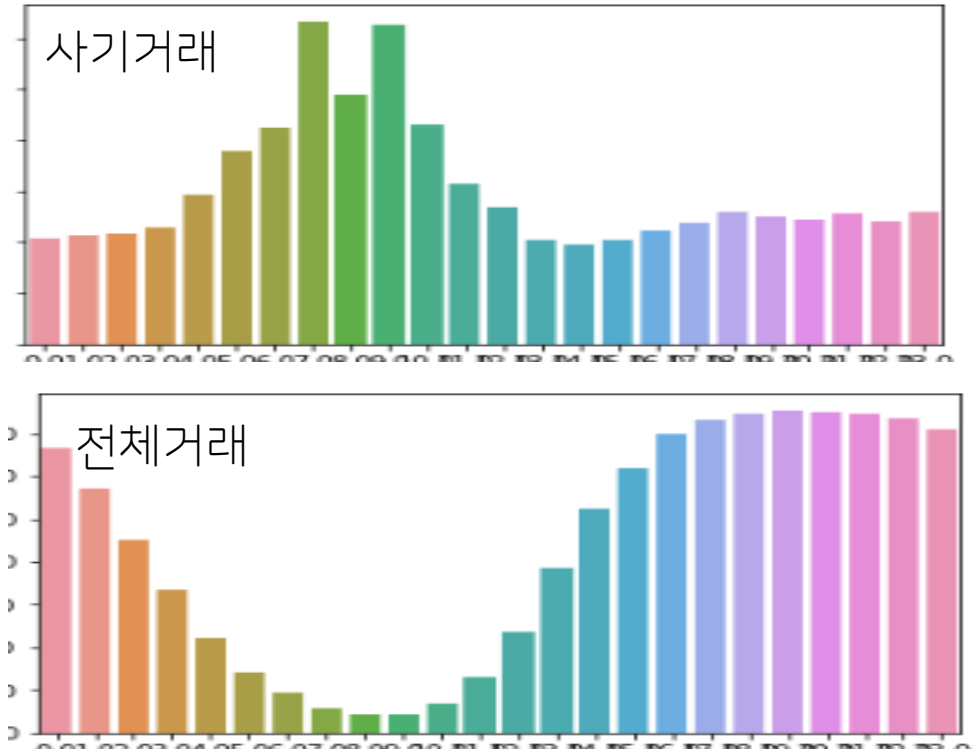
➡ 특정 시간대에 사기거래들이 몰림

➡ 이를 반영하기 위해

국가별 / 시간대별 사기비율을 새 column으로 만듦

1. 전처리 과정

C_fraud



V338	V339	hour	card3	c_fraud
NaN	NaN	0.0	150.0	2.088482
NaN	NaN	0.0	150.0	2.088482
NaN	NaN	0.0	150.0	2.088482
NaN	NaN	0.0	150.0	2.088482
0.0	0.0	0.0	150.0	2.088482

특정 국가 시간대별 사기비율을 넣어줌

1. 전처리 과정

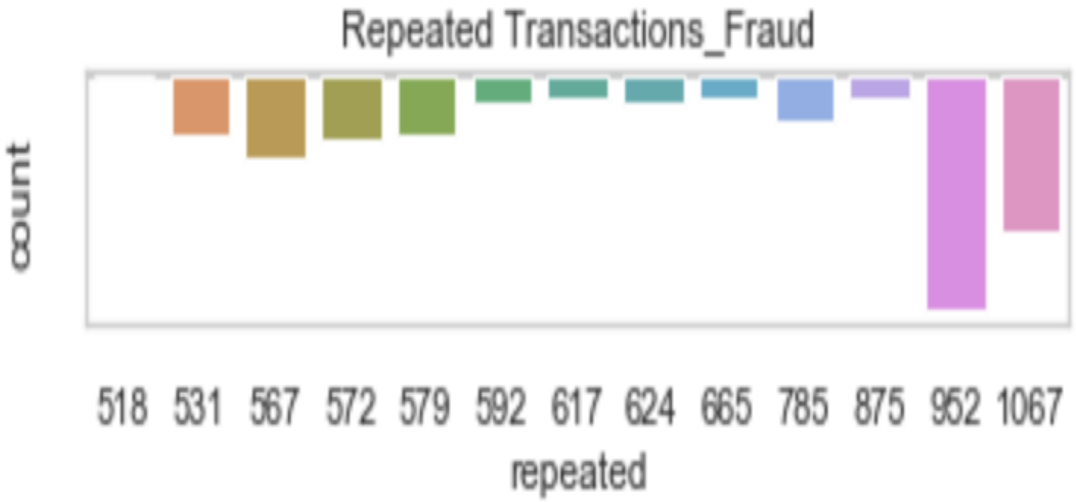
repeated

TransactionID	isFraud	TransactionDT	TransactionAmt	ProductCD	card1	card2	ca
2987203	1	89760	445.000	W	18268	583.0	15
2987240	1	90193	37.098	C	13413	103.0	18
2987243	1	90246	37.098	C	13413	103.0	18
2987245	1	90295	37.098	C	13413	103.0	18
2987288	1	90986	155.521	C	16578	545.0	18
2987367	1	92350	225.000	R	4425	562.0	15
2987405	1	92999	90.570	C	4504	500.0	18
2987630	1	97843	12.326	C	5812	408.0	18
2987683	1	99584	124.344	C	5812	408.0	18
2987736	1	100591	100.000	W	15063	NaN	15
2987779	1	102154	10.000	S	7481	364.0	14
2987780	1	102188	10.000	S	8732	360.0	15
2987781	1	102193	10.000	S	8732	360.0	15
2987869	1	106603	83.380	C	9026	545.0	18
2987923	1	108912	774.000	W	5033	269.0	15

TransactionAmt

➡ 카드 사기거래의 특이점을 발견

"사기 카드가 제대로 작동하는지 test하는 경향"



사용모델 및 성능평가

모델 사용 및 ROC-score

2. 모델 적용

사용모델

Xgboost, LGBM

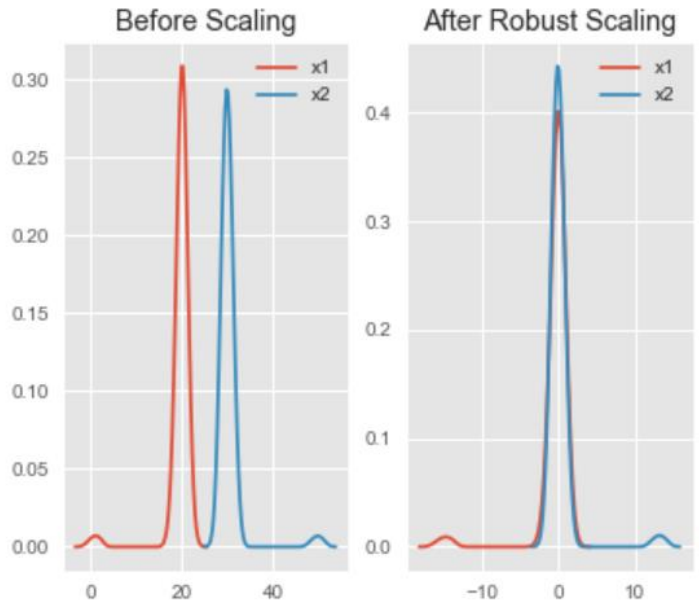
Xgboost_plot_importance

각 column들의 성능 기여도를 파악

Grid search

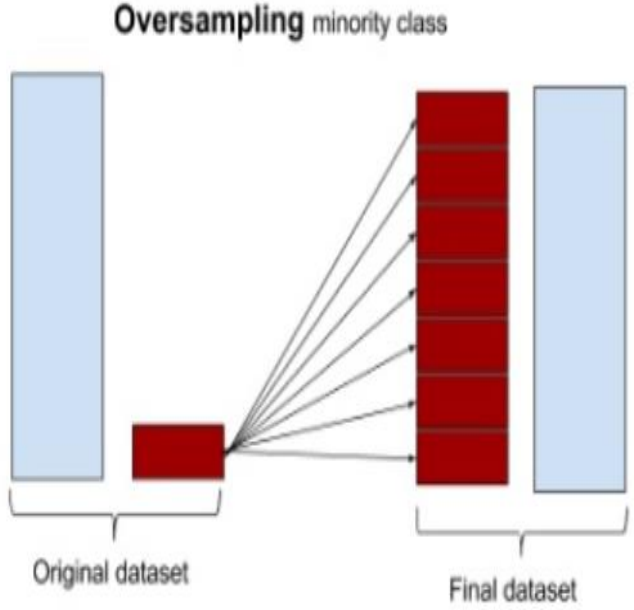
최적의 파라미터를 계산

Robust Scaling



➡ Robust scaling으로 X 변환

리샘플링(SMOTE)



➡ oversampling 방식으로 데이터 불균형 해소

3. 성능평가

GridSearchCV

```
[ ] xgb_model = xgb.XGBClassifier(learning_rate =0.001,
                                  n_jobs=-1,
                                  n_estimators=100,
                                  max_depth=15,
                                  min_child_weight=2,
                                  gamma=0,
                                  #n_estimators=2000,
                                  subsample=0.9,
                                  colsample_bytree=0.9,
                                  missing=-999,
                                  tree_method='gpu_hist')
```

```
[ ] gamma_test = {'gamma':[i/100.0 for i in range(0,6)]}

    gsearch2 = GridSearchCV(estimator = xgb_model, param_grid = gamma_test, scoring='roc_auc', cv=5)

    gsearch2.fit(X_train,y_train)

    gsearch2.cv_results_['params'], gsearch2.best_params_, gsearch2.best_score_
```

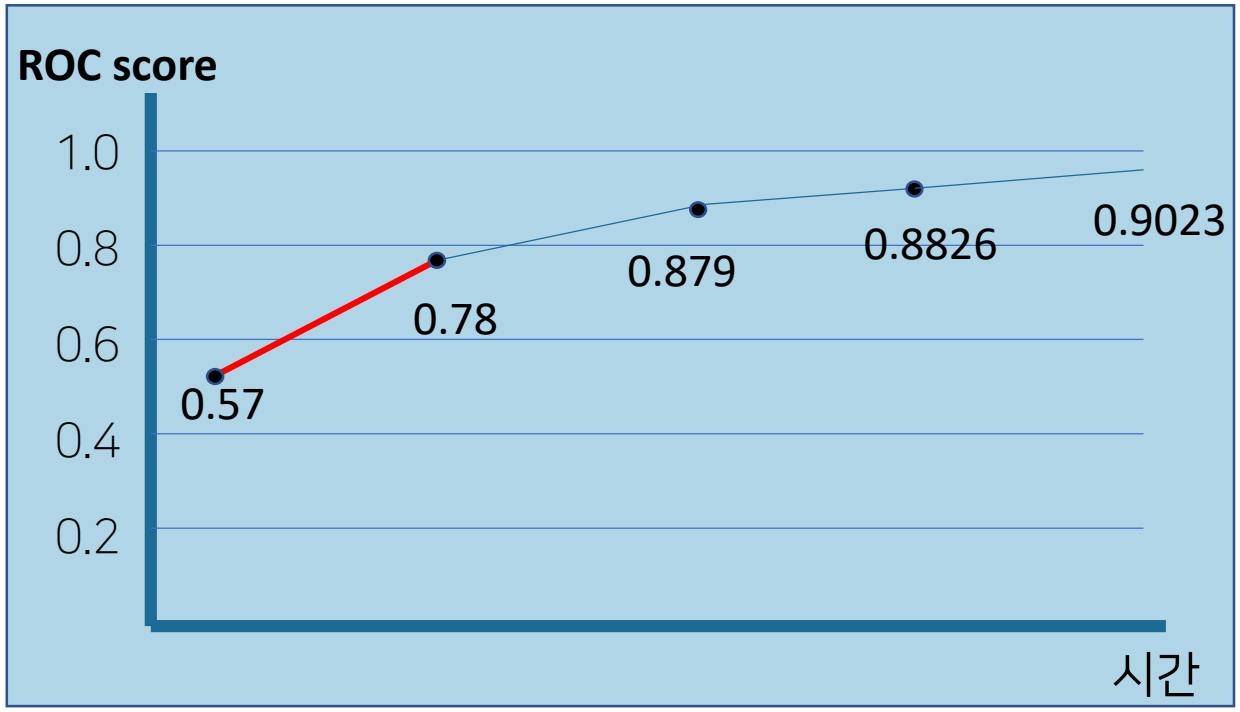
```
([{'gamma': 0.0},
  {'gamma': 0.01},
  {'gamma': 0.02},
  {'gamma': 0.03},
  {'gamma': 0.04},
  {'gamma': 0.05}],
 {'gamma': 0.02},
 0.8769353034565008)
```

Grid Search를 통해서
parameter의 최적값을 계산

```
xgb_model =
xgb.XGBClassifier(n_estimators=2000,
                  n_jobs=4,
                  max_depth=15,
                  learning_rate=0.001,
                  gamma = 0.02,
                  subsample = 0.9,
                  colsample_bytree=0.9,
                  missing=-999,)
```

3. 성능평가

Column 수정

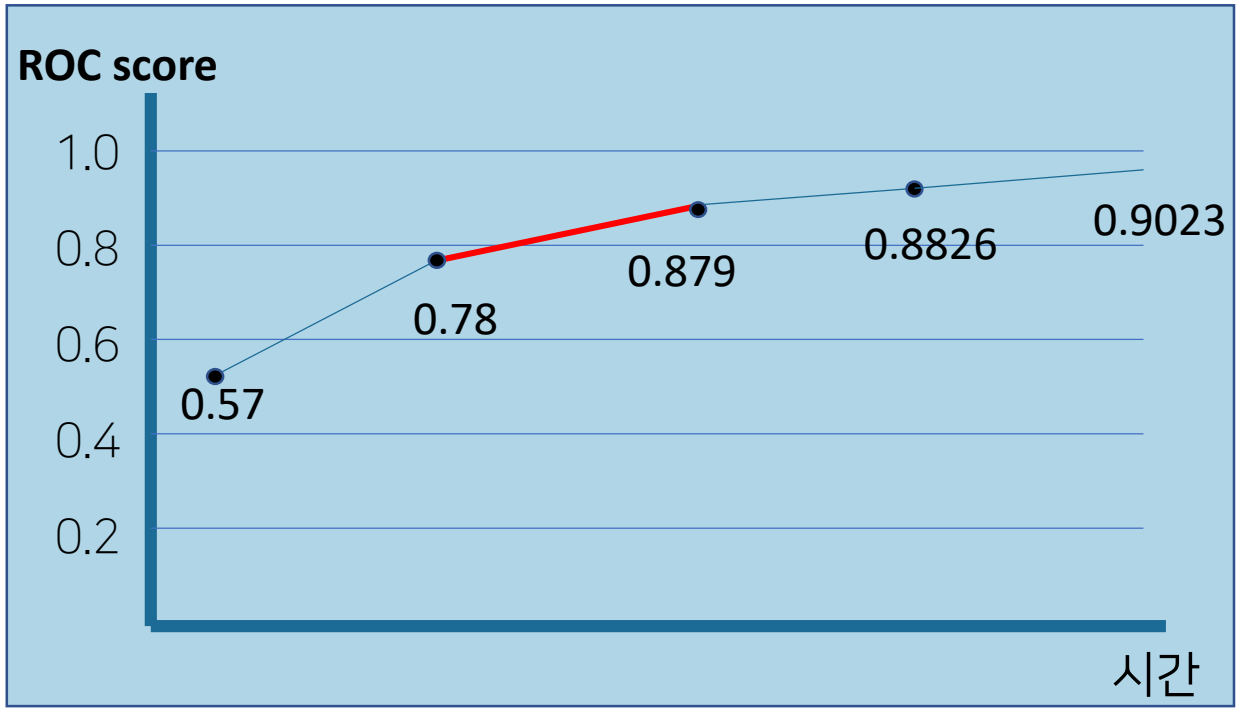


ROC_score 0.57 → 0.78

- 새로 만든 c_fraud가 문제인 것을 발견, c_fraud 삭제
- 사기비율 대신 **빈도수**로 대체

3. 성능평가

Column 수정

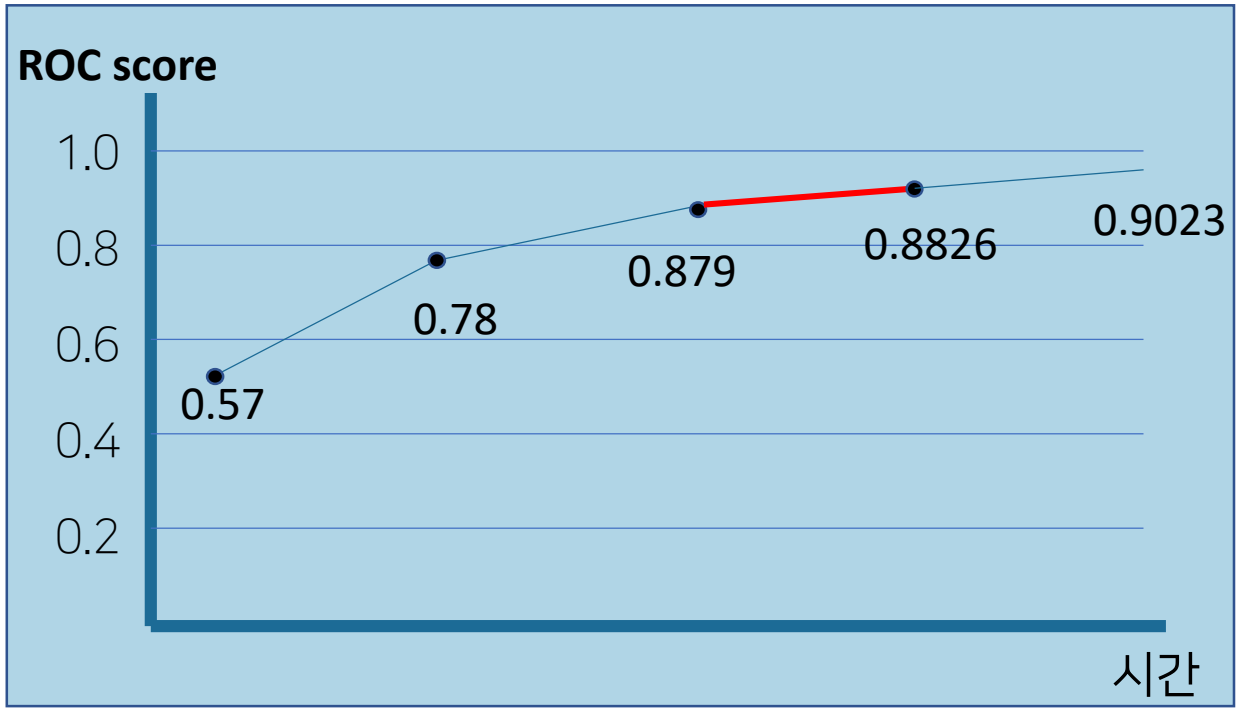


ROC_score 0.78 → 0.879

- Robust scaling과 SMOTE 처리 안 함
- Id, V column 추가
- train + test 묶어서 한 번에 PCA

3. 성능평가

Column 수정



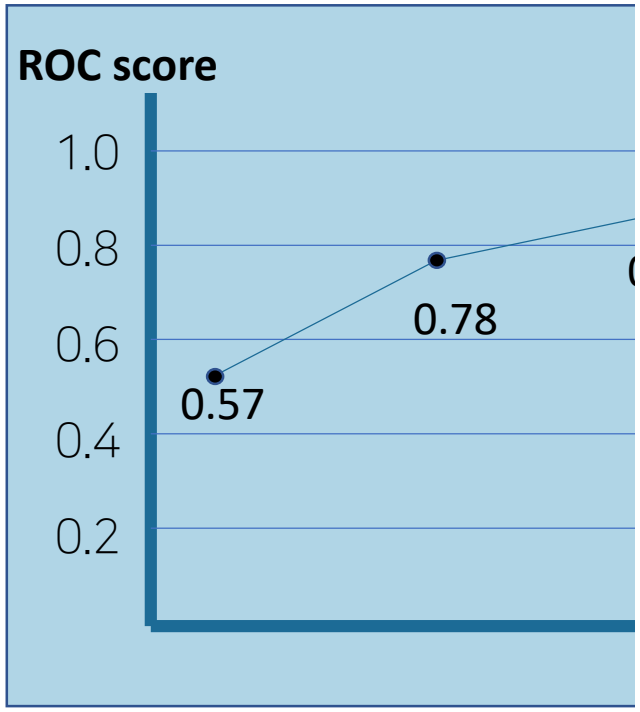
ROC_score 0.879 → 0.8826

➡ Xgboost_plot_importance을 사용하여

기여도가 낮은 column을 삭제

3. 성능평가

Column 수정




```
['C7_count', 0.12233107]
['ProductCD_C', 0.12139196]
['1_y', 0.08275286]
['C12_count', 0.036915515]
['C1_count', 0.03255757]
['C14_count', 0.022720547]
['card2_na', 0.019391662]
['0_y', 0.019241158]
['addr1_na', 0.01908694]
['ProductCD_W', 0.018294046]
['M4_na', 0.017781274]
['C13_count', 0.01772197]
['8', 0.01364583]
['6_y', 0.01294714]
['card6_count', 0.010793573]
['ProductCD_H', 0.010618027]
['C11_count', 0.009359405]
['C6_count', 0.0090164915]
['ProductCD_R', 0.008361908]
['C10_count', 0.008287751]
['C9_count', 0.007935721]
['D1', 0.0077629266]
['24', 0.007522261]
['2_y', 0.00743796]
['0_x', 0.0069390135]
['C4_count', 0.006814515]
['C8_count', 0.0066217026]
['D10', 0.0065505304]
```

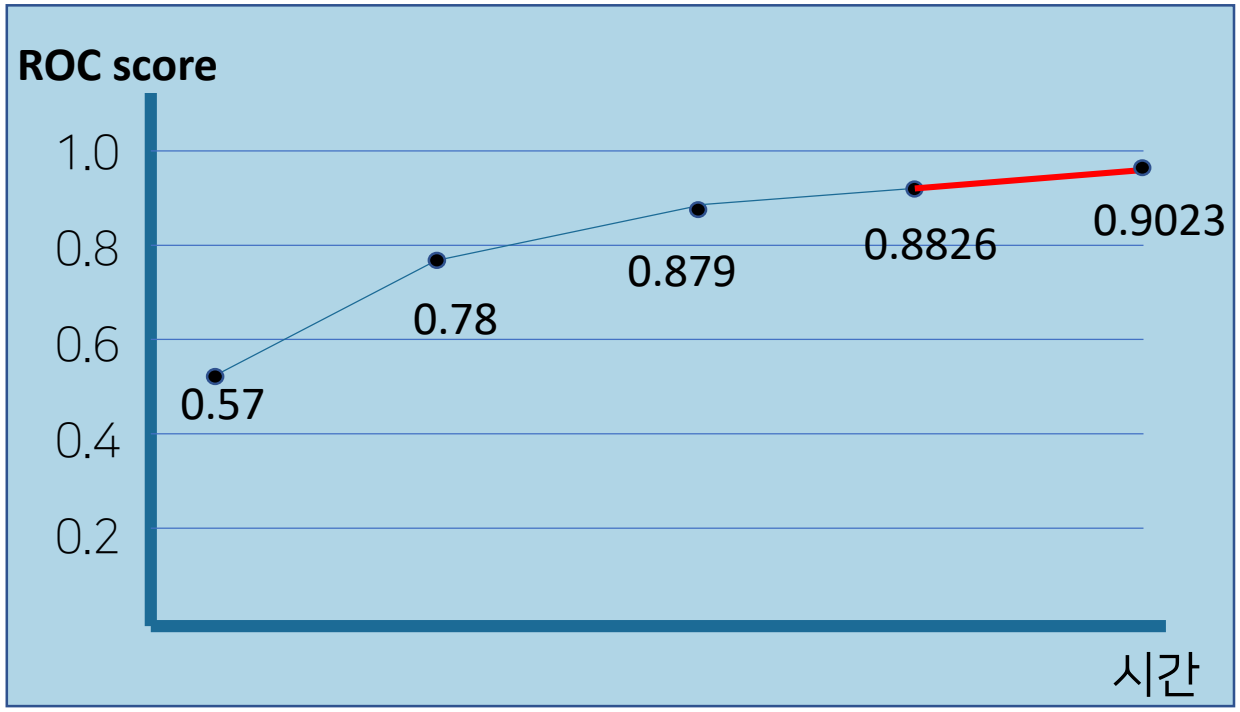
ROC_score 0.879 → 0.8826

➡ Xgboost_plot_importance을 사용하여

기여도가 낮은 column을 삭제

3. 성능평가

Column 수정



ROC_score 0.8826 → 0.9023

- D_columns, M_columns 추가
- LGBM 모델 사용

3. 성능평가

최종 성능 0.9023, 모델 LGBM
최종 사용 컬럼

TransactionDT, Hour

TransactionAmt_log,

TransactionAmt_residue

Repeated

ProductCD_dummies

Card1, 3, 5, 6_count

Card2_na

Card4_dummies

Addr2_count

Dist1, 2_na

C_count

email_pca

D1, D4, D10, D15

M1_na, M4_na, M6_na, M7_na

감사합니다