

신용카드 사용자의 신용도 예측

전공 프로젝트

5조 연체 멈춰! 

김혜림 최서영 현재웅 황영민

01
프로젝트 개요

-
프로젝트 주제
팀 구성 및 역할
데이터 설명

02
EDA와 전처리

-
EDA
Feature Engineering
데이터 스케일링

03
모델링

-
과적합 방지
기본 분류 모델
앙상블 모델

04
결론

-
최종 선정모델
보완점

프로젝트 주제

- 기존 신용카드 사용자의 개인 신상정보 데이터를 통해, 사용자의 대금 연체 정도(신용도)를 예측하는 알고리즘 개발

데이터 분석 목표

- 올바른 신용도 예측을 통해 모델의 출력값과 오차를 정의하는 함수인 예측 모델의 **logloss**를 최소화


신용카드 사용자 연체 예측 AI 경진대회


월간 데이콘 14 | 금융 | 정형 | Logloss

💰 상금 : 100만원

🕒 2021.04.05 ~ 2021.05.24 17:59 [+ Google Calendar](#)

👥 781팀 📅 D-35



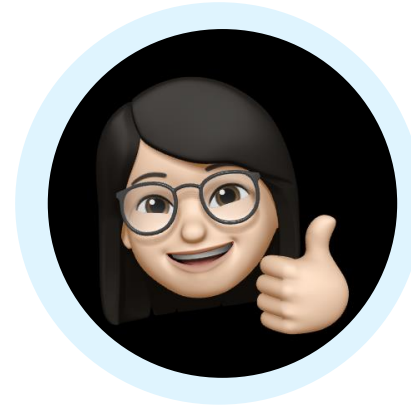


참여



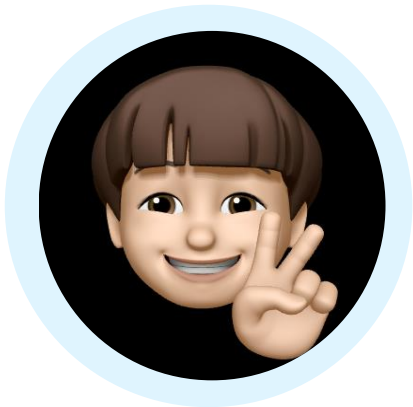
현재웅

- 앙상블 모델
- 데이터 분석



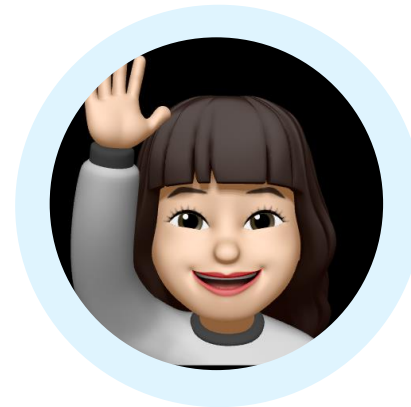
최서영

- EDA
- 통계 분석



황영민

- 머신러닝 모델
- 데이터 분석



김혜림

- 데이터 전처리
- ML/DL 모델링

데이터 설명

- 데이콘 제공: 신용카드 사용자들의 개인 신상정보 데이터
- 설명 변수(18개)
 - 12개의 범주형 변수
 - 6개의 수치형 변수
- 목표 변수 : 숫자가 작을수록 신용도가 좋음을 의미. Credit(0, 1, 2) -> 다중 분류 문제

변수의 종류		변수명	특징
설명변수	수치형 변수	child_num, income_total, DAYS_BIRTH, DAYS_EMPLOYED, family_size, begin_month	
	범주형 변수	gender, car, reality, FLAG_MOBIL, work_phone, phone, email,	이진변수
		income_type, edu_type, family_type, house_type, occyp_type	다중변수
목표변수		credit	0, 1, 2로 구성

01
프로젝트 개요

-
프로젝트 주제
팀 구성 및 역할
데이터 설명

02
EDA와 전처리

-
EDA
Feature Engineering
데이터 스케일링

03
모델링

-
과적합 방지
기본 분류 모델
앙상블 모델

04
결론

-
최종 선정모델
보완점

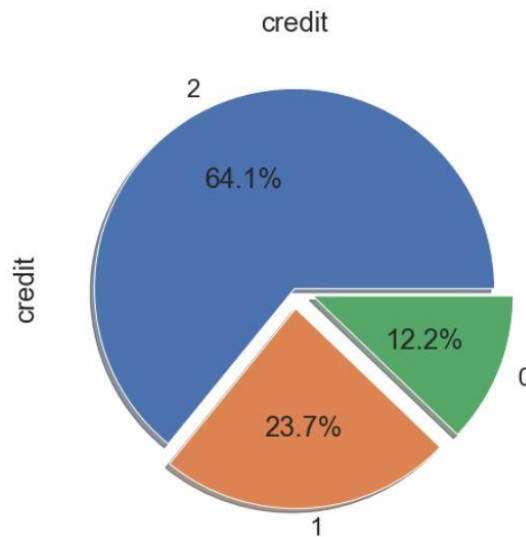
1. 데이터 탐색

- 행과 열 개수 : (26457, 19)
- 범주형 변수와 수치형 변수로 구성

```
RangeIndex: 26457 entries, 0 to 26456
Data columns (total 19 columns):
#   Column                Non-Null Count  Dtype
---  -
0   gender                26457 non-null  object
1   car                   26457 non-null  object
2   reality               26457 non-null  object
3   child_num             26457 non-null  int64
4   income_total          26457 non-null  float64
5   income_type           26457 non-null  object
6   edu_type              26457 non-null  object
7   family_type           26457 non-null  object
8   house_type            26457 non-null  object
9   DAYS_BIRTH            26457 non-null  int64
10  DAYS_EMPLOYED         26457 non-null  int64
11  FLAG_MOBIL            26457 non-null  int64
12  work_phone            26457 non-null  int64
13  phone                 26457 non-null  int64
14  email                 26457 non-null  int64
15  occyp_type            18286 non-null  object
16  family_size           26457 non-null  int64
17  begin_month           26457 non-null  int64
18  credit                26457 non-null  int64
dtypes: float64(1), int64(10), object(8)
```

2. 목표변수 확인

- credit 0 : 신용도 높음 / credit 2 : 신용도 낮음.
- 신용도가 제일 낮은 사람들이 과반수가 넘는 64.1%를 차지함.
- 신용도가 제일 높은 사람들은 12.2%에 불과함.



```
f, ax = plt.subplots(1, 2, figsize=(18, 8))
data['credit'].value_counts().plot.pie(explode=[0, 0.1, 0.1],
                                       autopct='%1.1f%%',
                                       ax=ax[0],
                                       shadow=True)

ax[0].set_title('credit')
ax[0].set_ylabel('credit')
plt.show()
```

3. 결측치 처리

✓ 결측치 확인

- Train Data의 직업 유형 열에서 8171개의 결측치
- Test Data의 직업 유형 열에서 3152개의 결측치

훈련데이터 결측치 합	테스트데이터 결측치 합
gender	gender
car	car
reality	reality
child_num	child_num
income_total	income_total
income_type	income_type
edu_type	edu_type
family_type	family_type
house_type	house_type
DAYS_BIRTH	DAYS_BIRTH
DAYS_EMPLOYED	DAYS_EMPLOYED
FLAG_MOBIL	FLAG_MOBIL
work_phone	work_phone
phone	phone
email	email
occyp_type 8171	occyp_type 3152
family_size	family_size
begin_month	begin_month
credit	credit
dtype: int64	dtype: int64

결측치 처리 전

✓ 결측치 처리 방법

- 직업이 Null 값인 사람 중 "소득 형태 = 연금"인 사람의 비율 54.3%
- 소득 형태(income type)가 연금(Pensioner) 이고, 직업(occyp type)이 Null인 사람은 은퇴자(Retired)로 처리
- 그 외의 직업이 Null인 사람은 NaN으로 처리

```
# 결측치 처리
```

```
cond = (train['income_type'] == 'Pensioner')
train['occyp_type'] = train['occyp_type'].fillna(cond.map({True: 'Retired', False: 'NaN'}))
```

```
cond = (test['income_type'] == 'Pensioner')
test['occyp_type'] = test['occyp_type'].fillna(cond.map({True: 'Retired', False: 'NaN'}))
```

```
# 확인
```

```
print('훈련데이터 결측치 합 \n', train.isnull().sum())
print('테스트데이터 결측치 합 \n', test.isnull().sum())
```

occyp_type 0	email 0
family_size 0	occyp_type 0
begin_month 0	family_size 0
credit 0	begin_month 0
dtype: int64	dtype: int64

결측치 처리 후

4. 이상치 처리

- EDA 중 "child_num", "family_size" 열에서 이상치 발견
- Ex) 자녀 수가 14명, 가족 수가 20명

1) 자녀 수

```
print('train childnum unique:', train['child_num'].unique())
```

```
train childnum unique: [ 0  1  2  3  4  5 14 19  7]
```

```
# 아이가 3명 이상인 데이터는 2로 변경
```

```
train.loc[train['child_num'] >= 3, 'child_num'] = 2
```

```
print('childnum unique:', train['child_num'].unique())
```

```
test.loc[test['child_num'] >= 3, 'child_num'] = 2
```

```
childnum unique: [0 1 2]
```

2) 가족 규모

```
print('train familysize unique:', train['family_size'].unique())
```

```
train familysize unique: [ 2.  3.  4.  1.  5.  6.  7. 15. 20.  9.]
```

```
# 가족규모가 5명 이상인 데이터는 4로 변경
```

```
train.loc[train['family_size'] >= 5, 'family_size'] = 4
```

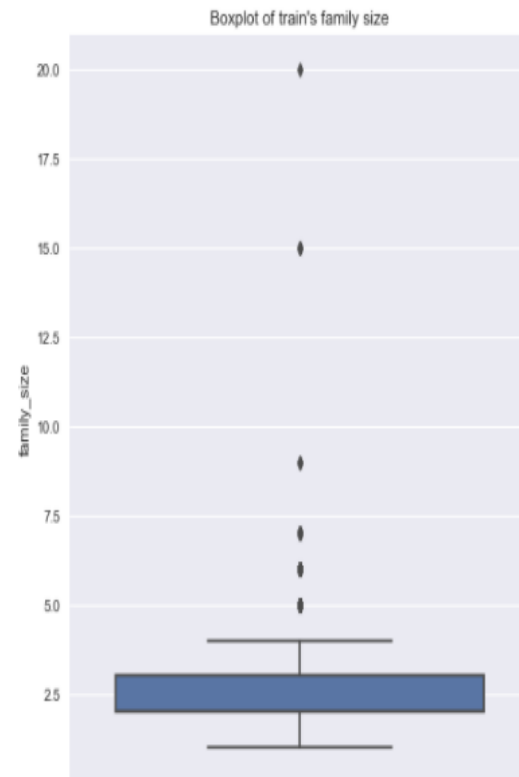
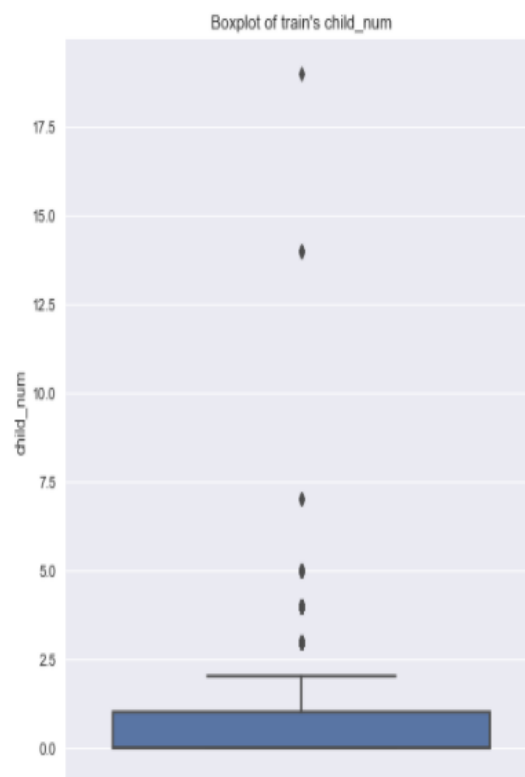
```
print('train familysize unique:', train['family_size'].unique())
```

```
test.loc[test['family_size'] >= 5, 'family_size'] = 4
```

```
train familysize unique: [2. 3. 4. 1.]
```

✓ IQR 방식을 통한 이상치 제거

- child_num이 3 이상인 값은 2로 변환
- family_size이 5 이상인 값은 4로 변환

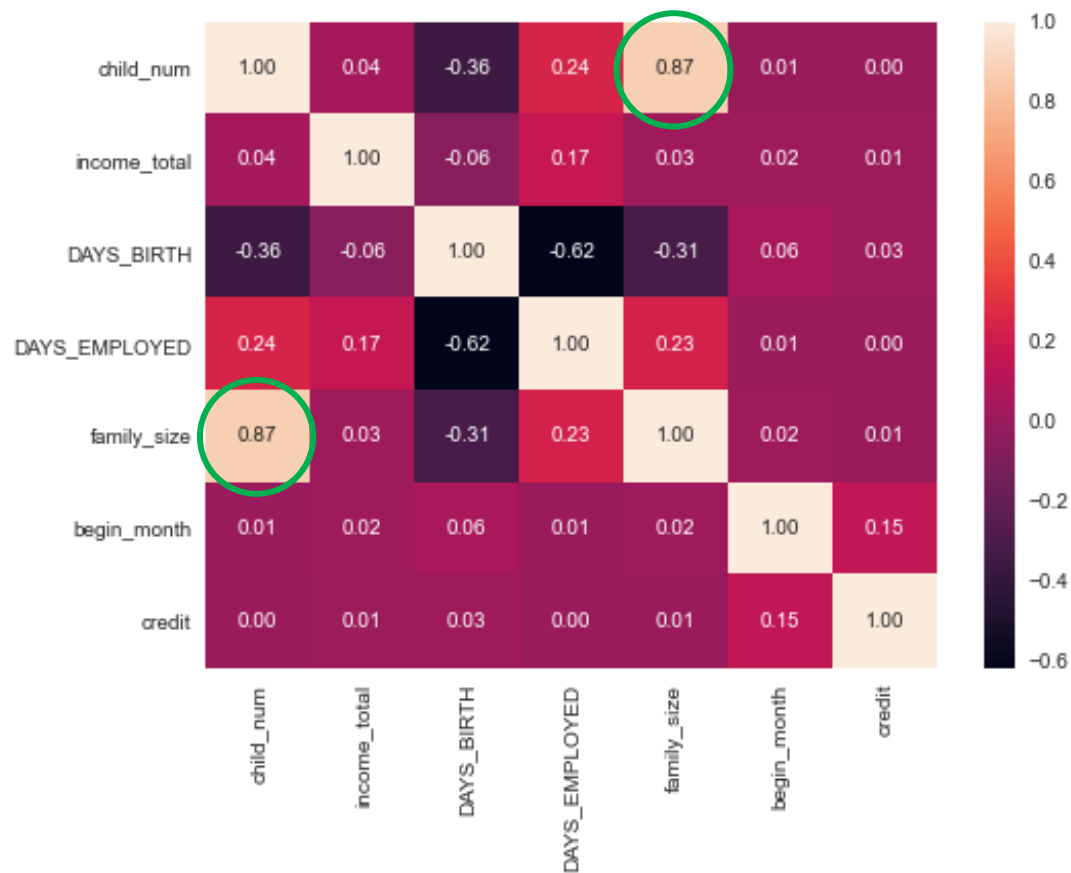


1. 다중 공선성 제거

- 상관 관계 Heatmap 결과, child_num과 family_size의 상관 관계가 상당히 높게 나옴
- 다중 공선성 제거를 위해 child_num 열 삭제

```
# child_num 변수 제거
train = train.drop(['child_num'], axis=1)
test = test.drop(['child_num'], axis=1)
```

```
display(train.head(5))
display(test.head(5))
```



2. 범주형 변수 인코딩

- 범주형 변수: gender, car, reality, Income_type, edu_type, family_type, house_type, occyp_type -> 총 8개

One- Hot Encoding 사용

- Label Encoding 시 숫자의 특성으로 인해 특정 모델에서 가중치의 차이가 발생
- 즉, 실제 각 value의 중요도는 똑같은 반면, Encoding 후 중요도의 차이가 발생
- 따라서 숫자의 특성(중요도)을 없애기 위해 **One-Hot Encoding** 실행

11	gender_F	26457	non-null	float64
12	gender_M	26457	non-null	float64
13	car_N	26457	non-null	float64
14	car_Y	26457	non-null	float64
15	reality_N	26457	non-null	float64
16	reality_Y	26457	non-null	float64
17	income_type_Commercial associate	26457	non-null	float64
18	income_type_Pensioner	26457	non-null	float64
19	income_type_State servant	26457	non-null	float64
20	income_type_Student	26457	non-null	float64
21	income_type_Working	26457	non-null	float64
22	edu_type_Academic degree	26457	non-null	float64
23	edu_type_Higher education	26457	non-null	float64
24	edu_type_Incomplete higher	26457	non-null	float64
25	edu_type_Lower secondary	26457	non-null	float64
26	edu_type_Secondary / secondary special	26457	non-null	float64
27	family_type_Civil marriage	26457	non-null	float64
28	family_type_Married	26457	non-null	float64
29	family_type_Separated	26457	non-null	float64
30	family_type_Single / not married	26457	non-null	float64
31	family_type_Widow	26457	non-null	float64
32	house_type_Co-op apartment	26457	non-null	float64
33	house_type_House / apartment	26457	non-null	float64
34	house_type_Municipal apartment	26457	non-null	float64
35	house_type_Office apartment	26457	non-null	float64
36	house_type_Rented apartment	26457	non-null	float64
37	house_type_With parents	26457	non-null	float64
38	occyp_type_Accountants	26457	non-null	float64
39	occyp_type_Cleaning staff	26457	non-null	float64
40	occyp_type_Cooking staff	26457	non-null	float64
41	occyp_type_Core staff	26457	non-null	float64
42	occyp_type_Drivers	26457	non-null	float64
43	occyp_type_HR staff	26457	non-null	float64
44	occyp_type_High skill tech staff	26457	non-null	float64
45	occyp_type_IT staff	26457	non-null	float64
46	occyp_type_Laborers	26457	non-null	float64
47	occyp_type_Low-skill Laborers	26457	non-null	float64
48	occyp_type_Managers	26457	non-null	float64
49	occyp_type_Medicine staff	26457	non-null	float64
50	occyp_type_NaN	26457	non-null	float64
51	occyp_type_Private service staff	26457	non-null	float64
52	occyp_type_Realty agents	26457	non-null	float64
53	occyp_type_Retired	26457	non-null	float64
54	occyp_type_Sales staff	26457	non-null	float64
55	occyp_type_Secretaries	26457	non-null	float64
56	occyp_type_Security staff	26457	non-null	float64
57	occyp_type_Waiters/barmen staff	26457	non-null	float64

스케일러 종류 (Scikit-Learn)

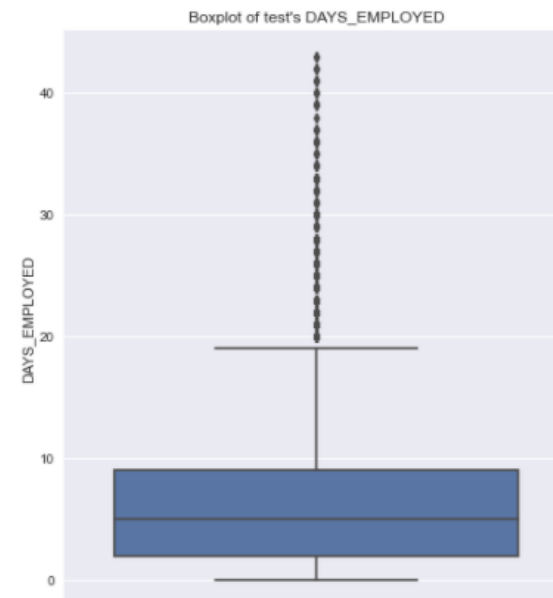
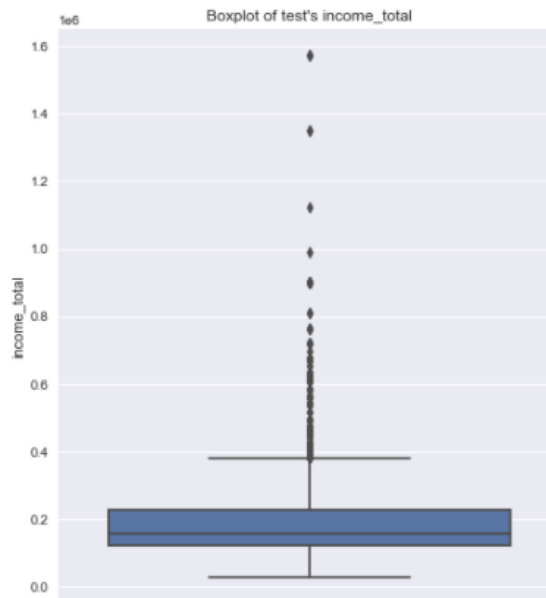
- MinMaxScaler
- MaxAbsScaler
- StandardScaler
- RobustScaler

선택 RobustScaler

- 중앙값과 IQR 사용
- > 이상치(Outlier)의 영향 최소화한 스케일러 기법

	child_num	income_total	DAYS_BIRTH	DAYS_EMPLOYED	family_size	begin_month
0	0	202500.0	13899	4709	2.0	6.0
1	1	247500.0	11380	1540	3.0	5.0
2	0	450000.0	19087	4434	2.0	22.0
3	0	202500.0	15088	2092	2.0	37.0
4	0	157500.0	15037	2105	2.0	26.0
...
26452	2	225000.0	12079	1984	4.0	2.0
26453	1	180000.0	15291	2475	2.0	47.0
26454	0	292500.0	10082	2015	2.0	25.0
26455	0	171000.0	10145	107	1.0	59.0
26456	0	81000.0	19569	1013	2.0	9.0

26457 rows × 6 columns



train data

	child_num	income_total	DAYS_BIRTH	DAYS_EMPLOYED	family_size	begin_month
	0	202500.0	13899	4709	2.0	6.0
	1	247500.0	11380	1540	3.0	5.0
	2	450000.0	19087	4434	2.0	22.0
	3	202500.0	15088	2092	2.0	37.0
	4	157500.0	15037	2105	2.0	26.0
...
26452	2	225000.0	12079	1984	4.0	2.0
26453	1	180000.0	15291	2475	2.0	47.0
26454	0	292500.0	10082	2015	2.0	25.0
26455	0	171000.0	10145	107	1.0	59.0
26456	0	81000.0	19569	1013	2.0	9.0

26457 rows × 6 columns

fit, transform



	child_num	income_total	DAYS_BIRTH	DAYS_EMPLOYED	family_size	begin_month	
	0	0.0	0.434783	-0.235934	1.154406	0.0	-0.666667
	1	1.0	0.869565	-0.596564	0.000364	1.0	-0.703704
	2	0.0	2.826087	0.506800	1.054261	0.0	-0.074074
	3	0.0	0.434783	-0.065712	0.201384	0.0	0.481481
	4	0.0	0.000000	-0.073014	0.206118	0.0	0.074074
...
26452	2.0	0.652174	-0.496492	0.162054	2.0	-0.814815	
26453	1.0	0.217391	-0.036650	0.340859	0.0	0.851852	
26454	0.0	1.304348	-0.782391	0.173343	0.0	0.037037	
26455	0.0	0.130435	-0.773372	-0.521486	-1.0	1.296296	
26456	0.0	-0.739130	0.575805	-0.191551	0.0	-0.555556	

26457 rows × 6 columns

test data

	child_num	income_total	DAYS_BIRTH	DAYS_EMPLOYED	family_size	begin_month
	0	112500.0	21990	0	2.0	60.0
	1	135000.0	18964	8671	2.0	36.0
	2	69372.0	15887	217	2.0	40.0
	3	112500.0	19270	2531	2.0	41.0
	4	225000.0	17822	9385	2.0	8.0
...
9995	0	202500.0	18593	5434	2.0	19.0
9996	0	202500.0	10886	1315	2.0	34.0
9997	0	292500.0	21016	14018	2.0	55.0
9998	0	180000.0	16541	1085	2.0	33.0
9999	0	270000.0	9154	187	2.0	11.0

10000 rows × 6 columns

transform



	child_num	income_total	DAYS_BIRTH	DAYS_EMPLOYED	family_size	begin_month	
	0	0.0	-0.434783	0.922405	-0.560452	0.0	1.333333
	1	0.0	-0.217391	0.489191	2.597232	0.0	0.444444
	2	0.0	-0.851478	0.048676	-0.481428	0.0	0.592593
	3	0.0	-0.434783	0.532999	0.361253	0.0	0.629630
	4	0.0	0.652174	0.325698	2.857247	0.0	-0.592593
...
9995	0.0	0.434783	0.436077	1.418427	0.0	-0.185185	
9996	0.0	0.434783	-0.667287	-0.081573	0.0	0.370370	
9997	0.0	1.304348	0.782963	4.544428	0.0	1.148148	
9998	0.0	0.217391	0.142305	-0.165331	0.0	0.333333	
9999	0.0	1.086957	-0.915247	-0.492353	0.0	-0.481481	

10000 rows × 6 columns

01
프로젝트 개요

-
프로젝트 주제
팀 구성 및 역할
데이터 설명

02
EDA와 전처리

-
EDA
Feature Engineering
데이터 스케일링

03
모델링

-
과적합 방지
기본 분류 모델
앙상블 모델

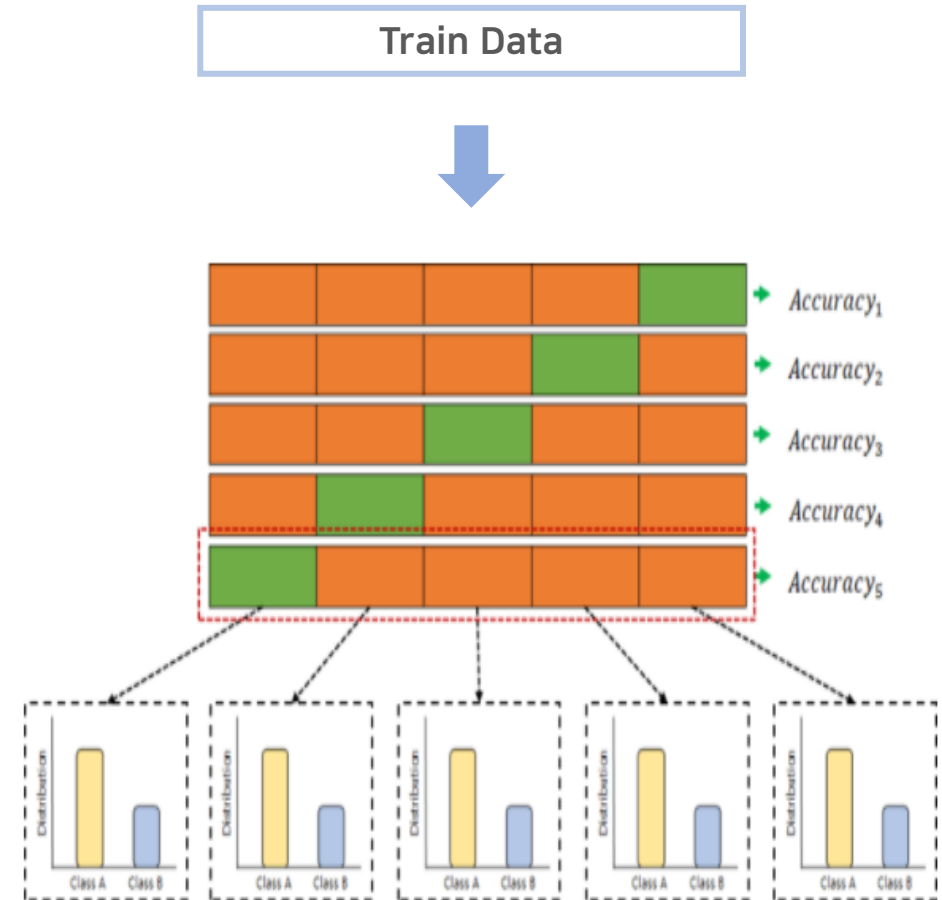
04
결론

-
최종 선정모델
보완점

1. 과적합 방지

- 고정된 train data와 test data로 평가하고,
반복적으로 모델을 튜닝하면 과적합 문제 발생
- 일반적인 k-fold CV 방법을 사용한다면, 각 fold마다
목표 변수의 분포가 달라지는 문제 발생

-> 목표 변수의 불균형한 분포까지 고려하여 data fold를
나누는 계층별 k-겹 교차 검증 방법 사용



계층별 k-겹 교차 검증

2. 기본 분류 모델

- 지도 학습 모델 중 분류 문제에서 자주 쓰이는 로지스틱 회귀, KNN, SVM 사용
- 앞서, k=5로 설정하여 교차 검증을 수행했기 때문에 총 5개의 검증 결과가 도출됨.
- 5번의 검증 시 도출한 혼동행렬, logloss의 평균을 통해 모델의 성능을 평가하기로 함.

```
skf = StratifiedKFold(n_splits=5, shuffle=True, random_state=42)
folds=[]

# 계층별 k-겹 교차검증을 위해 인덱스를 나누는 코드
for train_index, valid_index in skf.split(train, train['credit']):
    folds.append((train_index, valid_index))
display(folds)
```

```
[(array([ 0,  1,  3, ..., 26453, 26454, 26456]),
  array([ 2, 26, 45, ..., 26446, 26452, 26455])),
 (array([ 2,  3,  6, ..., 26454, 26455, 26456]),
  array([ 0,  1,  4, ..., 26435, 26447, 26450])),
 (array([ 0,  1,  2, ..., 26453, 26454, 26455]),
  array([ 3, 12, 16, ..., 26438, 26448, 26456])),
 (array([ 0,  1,  2, ..., 26454, 26455, 26456]),
  array([ 6, 14, 29, ..., 26440, 26442, 26453])),
 (array([ 0,  1,  2, ..., 26453, 26455, 26456]),
  array([ 7,  9, 10, ..., 26449, 26451, 26454]))]
```

```
# 로지스틱 회귀
logmodel = LogisticRegression()
logmodel.fit(X_train, Y_train)
y_hat = logmodel.predict(X_valid)
```

```
# knn 모델
knn = KNeighborsClassifier(n_neighbors=75) # 75로 성능을
knn.fit(X_train, Y_train)
y_hat = knn.predict(X_valid)
```

```
# 로지스틱 회귀
svm_model = svm.SVC(C=1, gamma = 10, kernel='rbf', probability=True)
svm_model.fit(X_train, Y_train)
```


• 모델별 검증 결과

로지스틱 회귀		분류 결과		
		0	1	2
실제 정답	0	0	8	637
	1	0	19	1234
	2	0	10	3383

Logloss: 0.8630

KNN		분류 결과		
		0	1	2
실제 정답	0	0	11	634
	1	0	33	1220
	2	0	25	3369

Logloss: 0.8618

SVM		분류 결과		
		0	1	2
실제 정답	0	99	52	496
	1	33	314	907
	2	60	189	3144

Logloss: 0.8094



로지스틱 회귀, KNN은 목표 변수 중 분포가 작은 0을 제대로 예측하지 못하는 문제점 발생
SVM이 가장 좋은 성능을 보임

3. 앙상블 모델

- 앙상블 학습을 통한 분류는 여러 개의 분류기를 생성하고, 그 예측을 결합함으로써 보다 정확한 최종 예측을 도출하는 기법
- 앙상블이 정형 데이터 분류 시 딥러닝보다 강력한 성능을 보여줌
- 앙상블 모델 중 랜덤포레스트, LightGBM, XGBoost 사용.

```
# LightGBM 모델
lgb = LGBMClassifier(n_estimators=1000)
lgb.fit(X_train, Y_train,
        eval_set=[(X_train, Y_train), (X_valid, Y_valid)],
        early_stopping_rounds=30,
        verbose=100)
```

```
# XGBoost
xgb = XGBClassifier()
xgb.fit(X_train, Y_train,
        eval_set=[(X_train, Y_train), (X_valid, Y_valid)],
        early_stopping_rounds=30,
        verbose=300)
```

```
# 랜덤포레스트
rfc = RandomForestClassifier(n_estimators=200, random_state=30, max_depth=10, n_jobs=-1)
rfc.fit(X_train, Y_train)
y_hat = rfc.predict(X_valid)
```

• 모델별 검증 결과

LightBGM		분류 결과		
		0	1	2
실제 정답	0	77	80	488
	1	24	415	814
	2	39	103	3251

Logloss: 0.7387

랜덤포레스트		분류 결과		
		0	1	2
실제 정답	0	1	75	569
	1	0	276	977
	2	0	4	3389

Logloss: 0.7933

XGBoost		분류 결과		
		0	1	2
실제 정답	0	59	82	504
	1	19	388	846
	2	25	81	3287

Logloss: 0.7420



전반적으로 앞서 설명한 기본 분류 모델보다 뛰어난 성능을 보여줌.
그 중에서도 **LightBGM**이 가장 좋은 성능을 보임.

01
프로젝트 개요

-
프로젝트 주제
팀 구성 및 역할
데이터 설명

02
EDA와 전처리

-
EDA
Feature Engineering
데이터 스케일링

03
모델링

-
과적합 방지
기본 분류 모델
앙상블 모델

04
결론

-
최종 선정모델
느낀점

1. 최종 선정 모델

- 앞서 Validation 결과를 통해 여러 모델 중 LightGBM, XGBoost의 좋은 성능을 확인함
- 두 모델로 Test data의 신용도를 예측한 결과,

- ① LightGBM의 logloss : 0.7290
 - ② XGBoost의 logloss : 0.7393




✓ “최종 모델로 logloss 값이 작은 LGBM 선정”

✓ 또한 계층별 k-겹 교차 검증 방법을 통해 과적합을 방지하여 두 모델 모두 Validation보다 좋은 결과를 얻을 수 있었음.

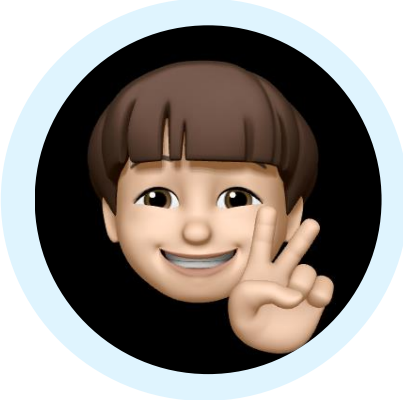
#2. 느낀점

현재웅



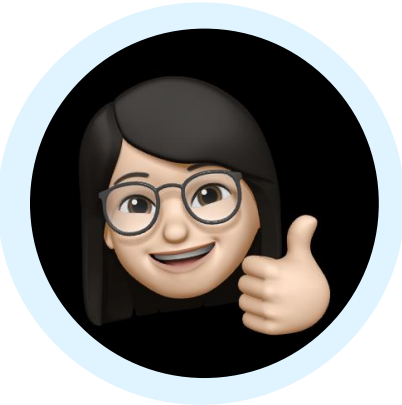
“사실 모델 성능을 높이기 위해 EDA 가설과 SMOTE 알고리즘 적용 등의 여러 시도를 팀원들과 했는데, 쉽게 모델의 성능이 오르지 않아 PPT에 담지 못해 아쉽습니다. 공모전 마감기한까지 모델을 고도화 시키기 위해 노력해서 더 많은 것을 배우고 싶습니다.”

황영민



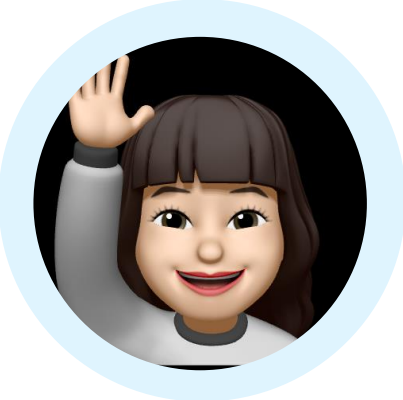
“데이터 분석을 처음부터 끝까지 다 적용해보면서 전체적인 흐름을 상세히 알게되었습니다. 각 예측모델의 하이퍼 파라미터 값을 변경해보면서 향상된 예측률을 얻어보고 싶습니다.”

최서영



“책에서만 보던 개념들을 실제로 확인하여 신기했습니다. 직접 전처리와 모델링을 하며 데이터의 특성에 맞는 방법을 고민하다보니 정말 많은 것을 배웠습니다. 이후 부스팅, 인공지능경망 등 복잡하지만 성능이 보완된 모델을 도전해보고 싶습니다.”

김혜림



“혼자 공부했을 때 보다 많은 것을 배울 수 있었습니다. 팀원들과 서로 모르는 내용을 가르쳐주고, 공부하면서 2배의 속도로 발전할 수 있는 시간이었습니다. 이후에 중복 데이터 처리에 대해서 더 고민하여 창의적인 아이디어를 통해 좋은 결과를 얻고 싶습니다.”

감사합니다 :)