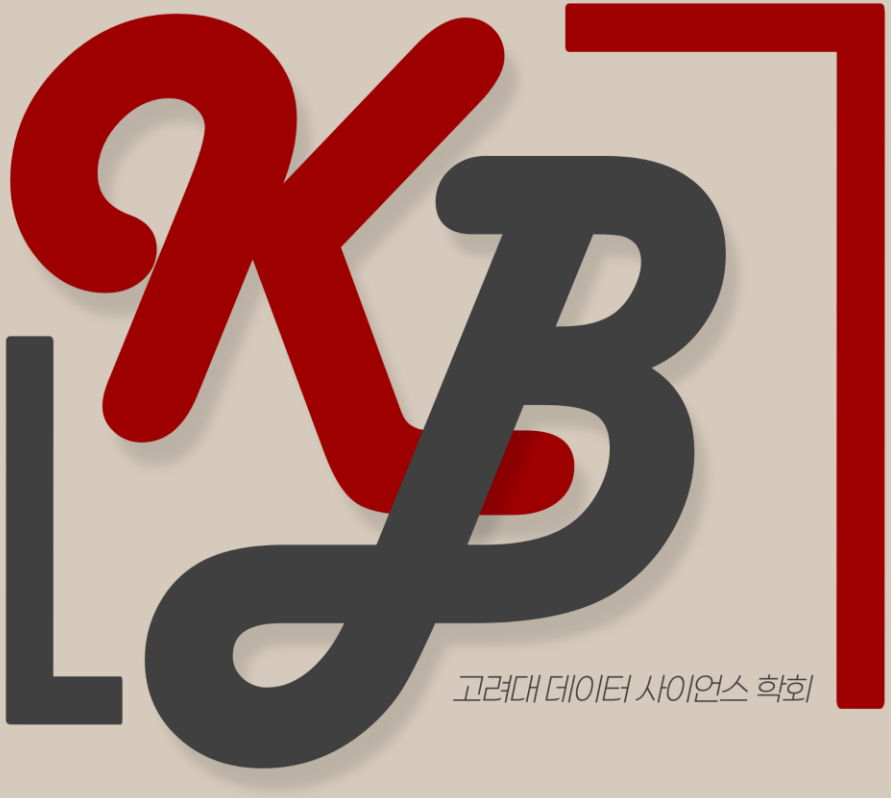
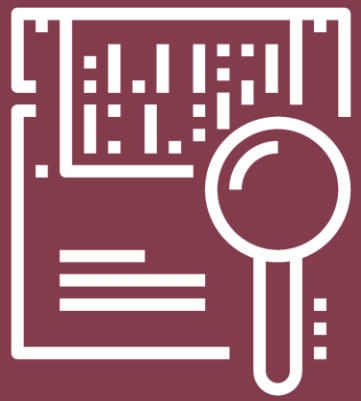


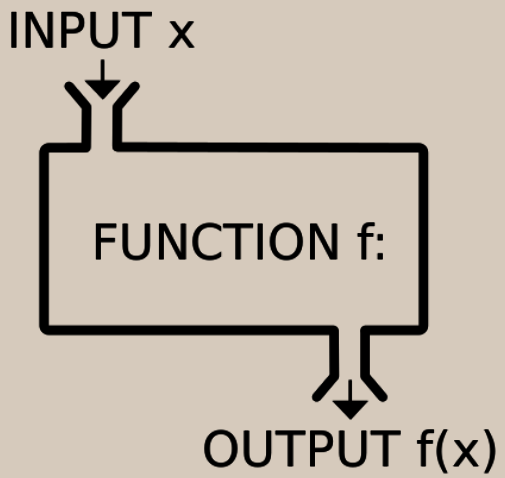
2023-1 KUBIG / ML 분반
신용카드 사용자 연체 예측 프로젝트

강민채, 김예은, 목진휘, 백서경

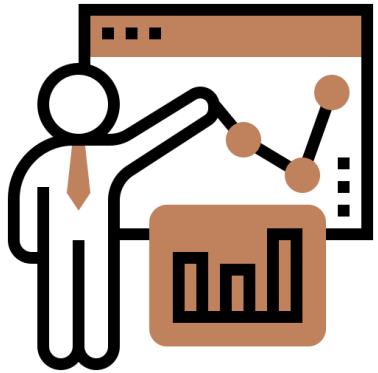
1. EDA & 전처리



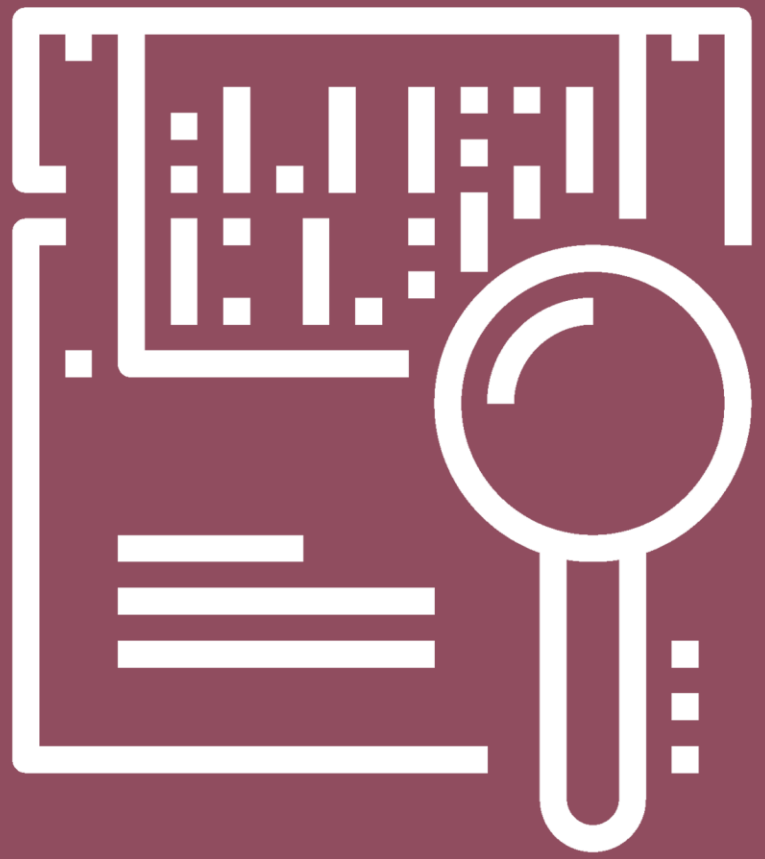
2. 모델



3. 결과



1. EDA & 전처리



1. EDA – 데이터 변수 설명

변수명	설명
gender	성별
car	차량 소유 여부
Reality	부동산 소유 여부
income_type	소득분류
edu_type	교육 수준
famliy_type	결혼 여부
house_type	주거 방식
FLAG_MOBIL	핸드폰 소유 여부
work_phone	업무용 전화 소유 여부
phone	전화 소유 여부
email	이메일 소유 여부
occyp_type	직업 유형
child_num	자녀 수
income_total	연간 소득
DAYS_BIRTH	데이터 수집일로부터 출생일까지 날짜 수
DAYS_EMPLOYED	데이터 수집일로부터 업무 시작일까지 날짜 수
family_size	가족 규모
begin_month	데이터 수집일로부터 시용카드 발급일까지 월 수
credit	신용등급(0,1,2)

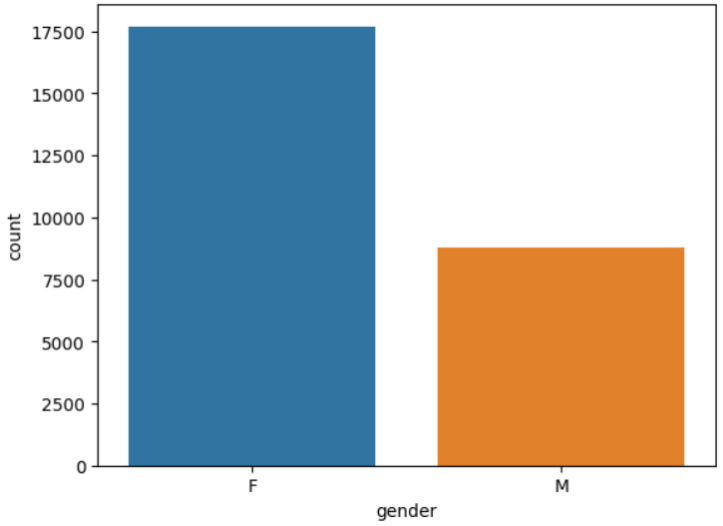
[목표]

총 16개의 feature를 가진 dataset으로
credit의 class(0,1,2)를 예측하는
classification 문제

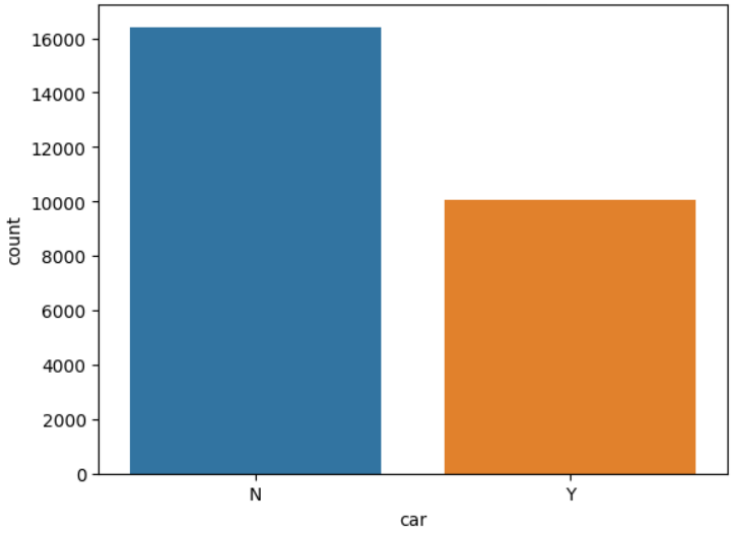
➡ TARGET

1. EDA – 범주형 변수

gender



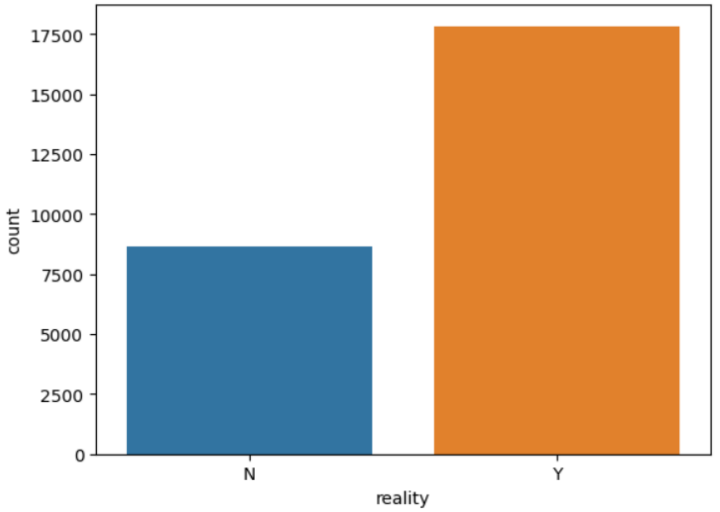
car



N: 차량을 소유하지 않음

Y: 차량을 소유함

reality

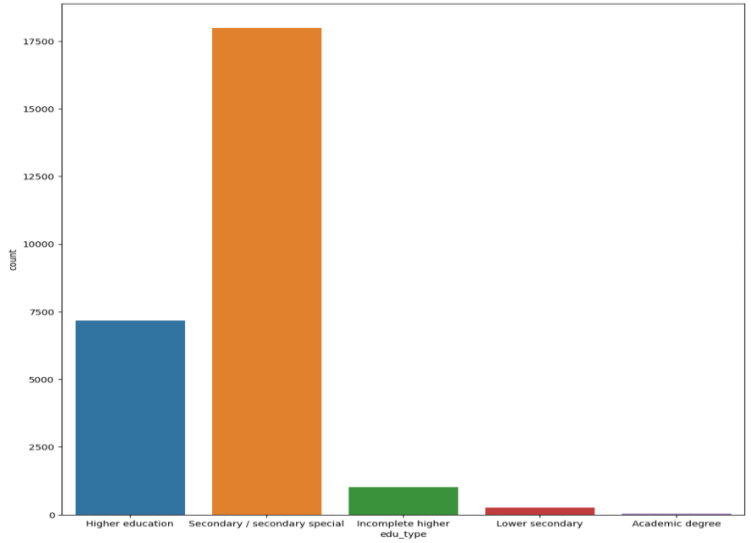


N: 부동산을 소유하지 않음

Y: 부동산을 소유함

1. EDA – 범주형 변수

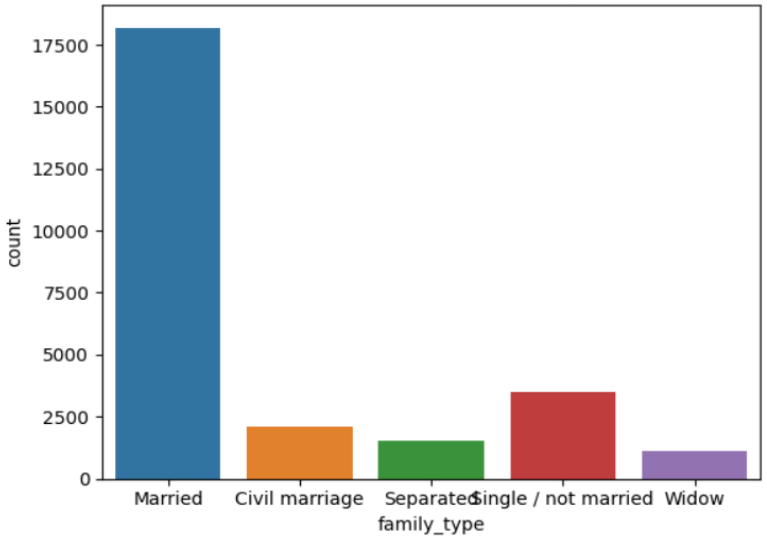
Edu_type



- Secondary/secondary special
- Higher education
- Incomplete higher edu_type
- Lower secondary
- Academic degree

순으로 분포

Family_type

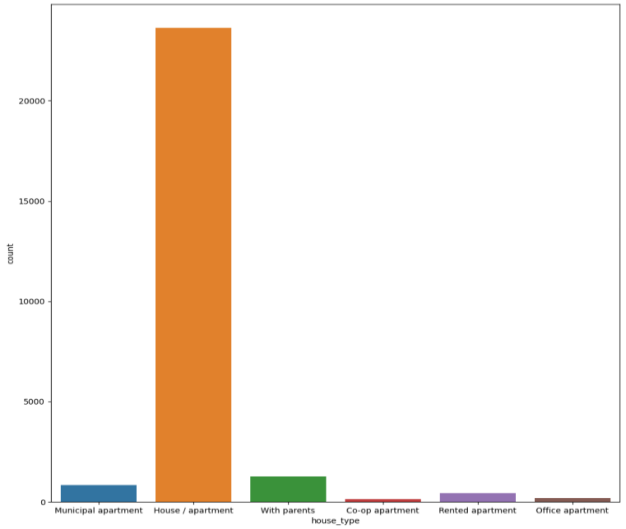


- Married
- Single/not married
- Civil marriage
- Separated family_type
- Widow

순으로 분포

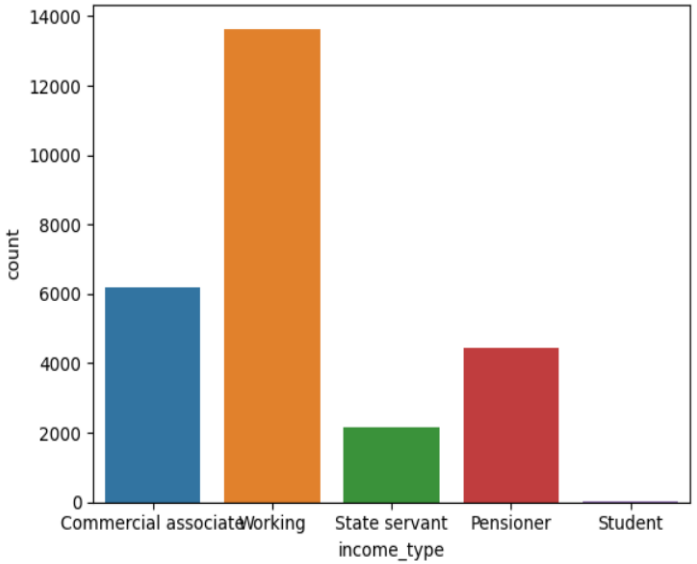
1. EDA – 범주형 변수

House_type



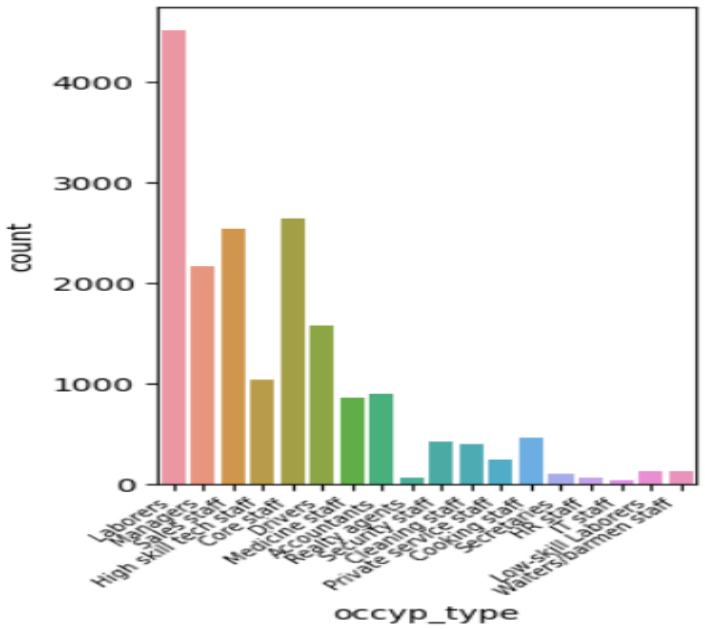
- House / apartment가 대부분

Income_type



- Working
- Commercial associate
- Pensioner
- State_servant income_type 순으로 분포

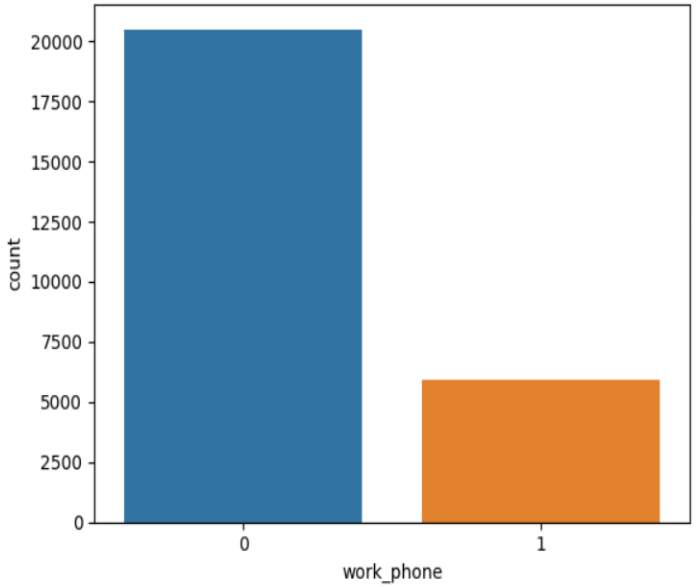
Occyp_type



- Laborers가 가장 많음
- 그 외에는 Core staff, Sales staff, Managers
- 결측치 존재

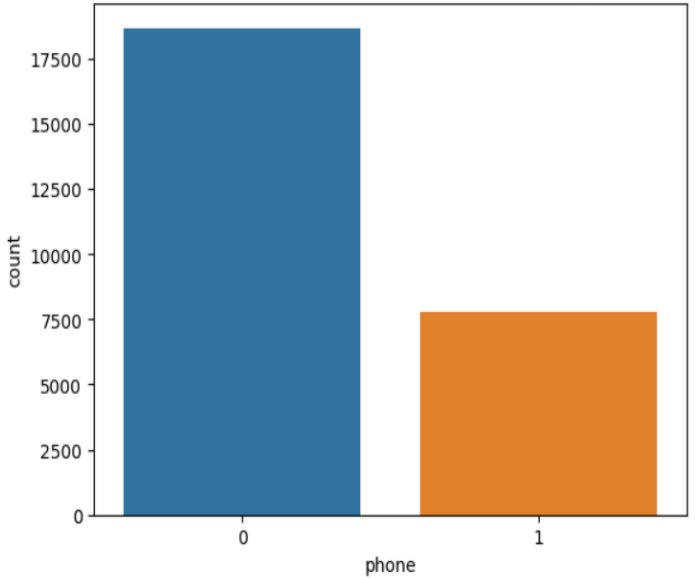
1. EDA – 범주형 변수

Work_phone



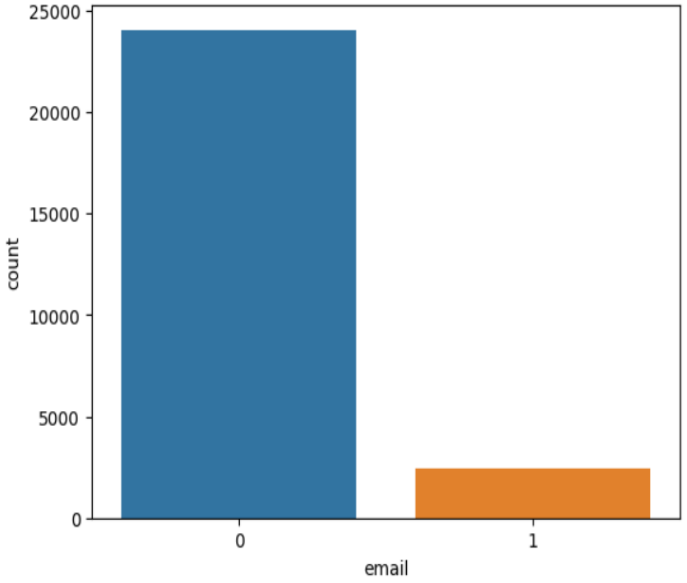
- 인코딩 된 형태(0/1)

Phone



- 인코딩 된 형태(0/1)

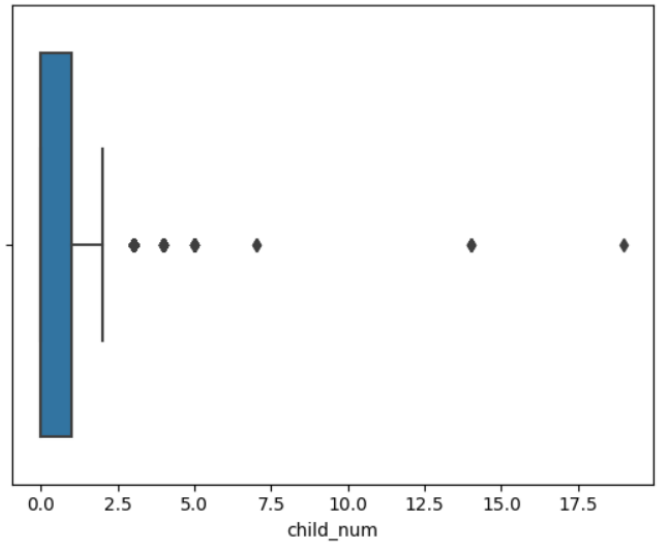
Email



- 인코딩 된 형태(0/1)

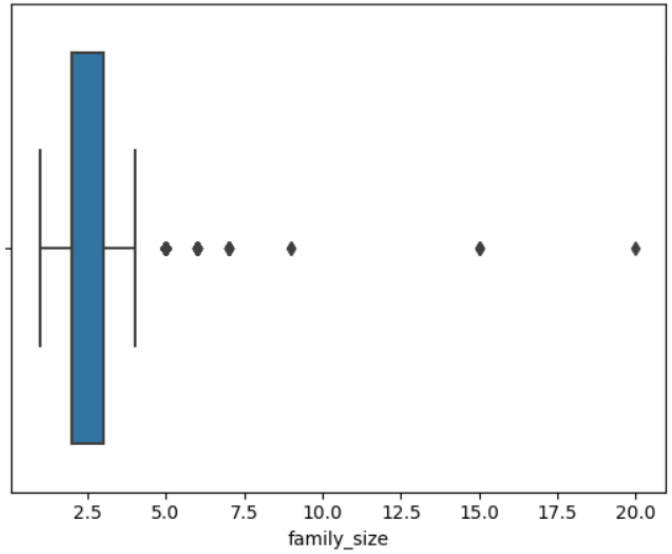
1. EDA – 수치형 변수

Child_num



이상치 처리 필요

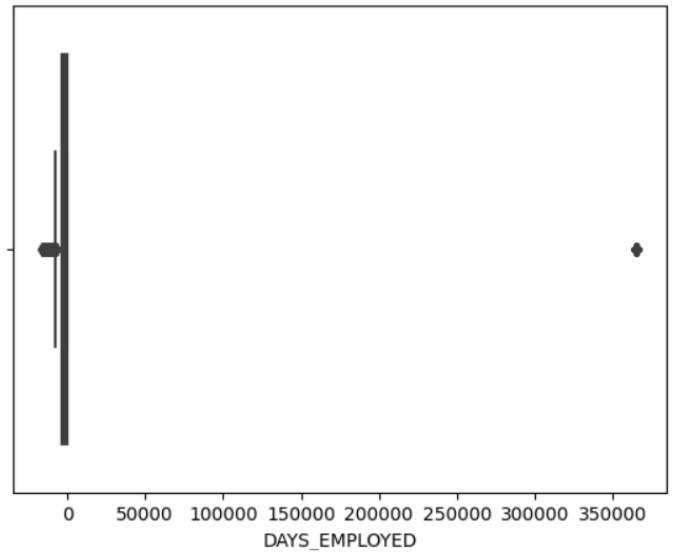
Family_size



이상치 처리 필요

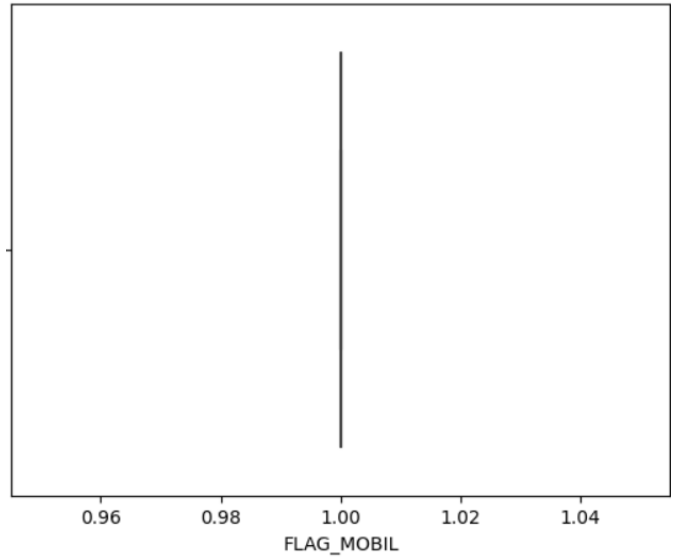
1. EDA – 수치형 변수

Days_employed



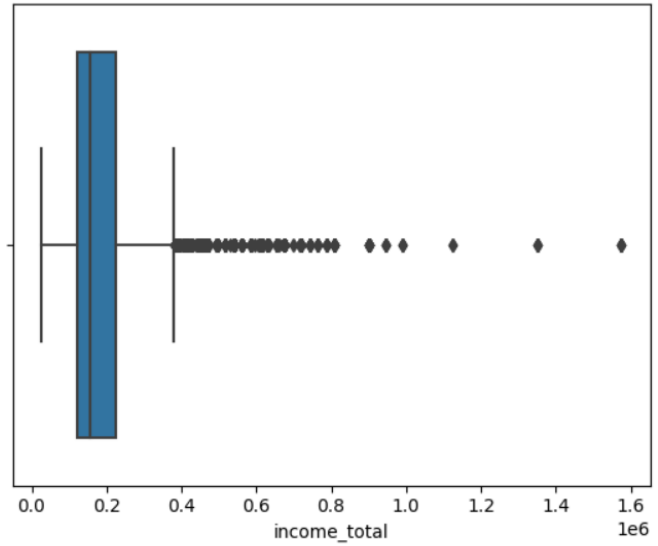
양수값 처리 필요

Flag_mobil



모든 값이 1 → 변수 제거

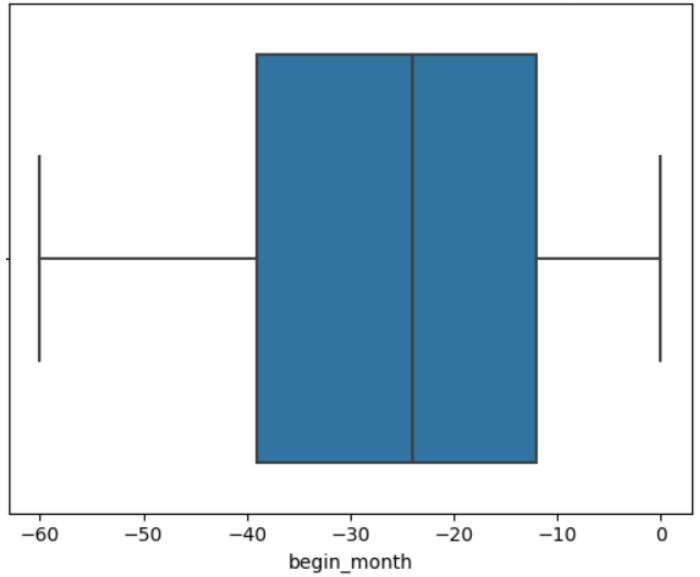
Income_total



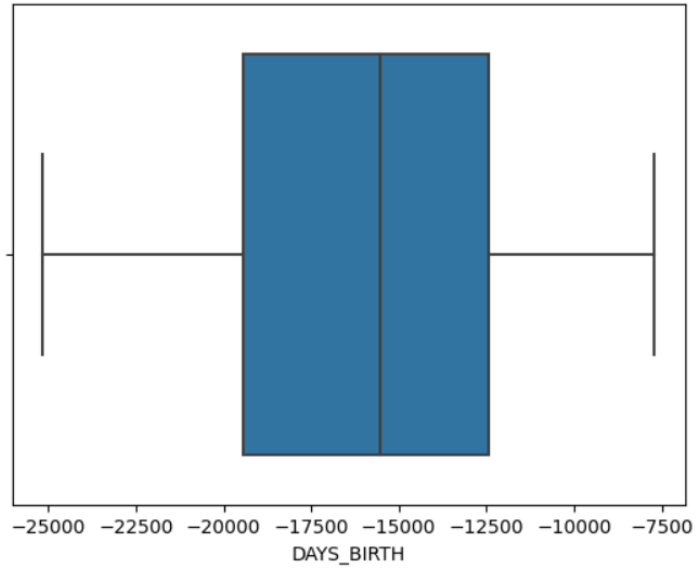
표준화/정규화 필요

1. EDA – 수치형 변수

Begin_month



Days_birth



1. 전처리 – 불필요 변수 제거

불필요변수 제거: Index, FLAG_MOBIL(모든 값이 1로 동일)

```
train.drop(['index','FLAG_MOBIL'],axis=1,inplace=True)
```

1. 전처리 – 결측치

occyp_type 〉〉〉 8171개의 결측치

Days_Employed 값이 음수

직업은 있으나 기입 X: 최빈값

```
employed_na=train[(train['DAYS_EMPLOYED']<0)&(train['occyp_type'].isna()==True)]
train['occyp_type']=train['occyp_type'].fillna('Laborers')
```

Days_Employed 값이 양수

직업이 없는 것으로 간주: 'no job'

```
employed_nojob=train[(train['DAYS_EMPLOYED']>=0)&(train['occyp_type'].isna()==True)
employed_nojob.fillna({'occyp_type':'No job'},inplace=True)
```

---

occyp_type 제거

모든 데이터에서 occyp_type 변수를 제거 후 분석

1. 전처리 - 중복데이터 issue

(1) 모든 feature에 관해 중복되는 경우

=〉 Train Set 전체 obs 26457개 중 1634개에 해당

=〉 중복되는 그룹에서 unique한 obs 한 가지만 남기고 나머지 drop

(2) begin_month(카드 발급일) 를 제외한 나머지 feature에 관해 중복되는 경우

=〉 Train Set 전체 obs 26457개 중 18535개에 해당

=〉 credit(신용도)를 포함한 나머지 모든 변수가 같고, 카드 발급일만 다른 경우이므로, 한 사람이 다양한 카드를 발급받은 것으로 파악.

=〉 중복되는 데이터의 개수를 이용하여 NumCards (카드의 개수)라는 파생변수 생성

gender	car	reality	income_total	income_type	edu_type	family_type	house_type	DAYS_BIRTH	DAYS_EMPLOYED	work_phone	phone	email	family_size	begin_month	credit
F	Y	Y	29250.0	Pensioner	Secondary / secondary special	Married	House / apartment	-20086	0	0	0	0	2.0	-2.0	1.0
F	Y	Y	29250.0	Pensioner	Secondary / secondary special	Married	House / apartment	-20086	0	0	0	0	2.0	-24.0	1.0
F	Y	Y	29250.0	Pensioner	Secondary / secondary special	Married	House / apartment	-20086	0	0	0	0	2.0	-6.0	1.0

예시) begin_month를 제외한 나머지 값이 중복인 경우

1. 전처리 - 중복데이터 issue

(3) Credit을 제외한 나머지 feature에 관해 중복

=〉 Train Set 전체 obs 26457개 중 1425개에 해당

=〉 모든 input이 같음에도 불구하고 Target이 다른 경우이므로, 학습에 어려움. 조치가 필요

=〉 일부를 Drop을 한 뒤 학습하거나, 새로운 파생변수를 만들어 학습해야 할 필요성

gender	car	reality	income_total	income_type	edu_type	family_type	house_type	DAYS_BIRTH	DAYS_EMPLOYED	work_phone	phone	email	family_size	begin_month	credit
F	Y	N	31500.0	Commercial associate	Secondary / secondary special	Married	House / apartment	-17762	-7734	1	1	0	2.0	-34.0	1.0
F	Y	N	31500.0	Commercial associate	Secondary / secondary special	Married	House / apartment	-17762	-7734	1	1	0	2.0	-34.0	0.0

예시) credit을 제외한 나머지 값이 중복인 경우

gender	car	reality	child_num	income_total	income_type	edu_type	family_type	house_type	DAYS_BIRTH	DAYS_EMPLOYED	FLAG_MOBIL	work_phone	phone	email	occyp_type	family_size	begin_month
F	N	Y	0	32400.0	State servant	Secondary / secondary special	Married	House / apartment	-19200	-5637	1	0	0	0	Managers	2.0	-9.0
F	N	Y	0	32400.0	State servant	Secondary / secondary special	Married	House / apartment	-19200	-5637	1	0	0	0	Managers	2.0	-9.0

예시) Test Set 또한, credit 을 제외한 나머지 값이 중복인 경우 (즉, 모든 값이 중복인 경우)가 존재

(4) 이 외에도, begin_month와 credit을 제외한 나머지 feature에 관해 중복인 경우 또한 존재

1. 전처리 – 범주형 변수 인코딩, 변수변환, 이상치 처리

인코딩 :

- 사용하려는 모델의 특성, 변수의 특성에 따라 다양하게 인코딩 가능

(One-Hot Encoding, Label Encoding, Ordinal Encoding)

변수 변환 :

- 수치형 feature들을 표준화 혹은 정규화 후 성능 비교 가능 (MinMax Scaling, Standard Scaling, Robust Scaling)
- Skewed 된 feature에 대해서 변환 가능 (Log transform)

이상치 처리:

- 이상치라 판단될 경우 제거 가능

1. 전처리 – 파생변수의 생성

파생변수 : 기존의 feature를 이용하여 새로운 feature를 생성

(1) 'Kids' (자녀 유무) : child_num과 family_size는 feature의 근본적 특성상 상관관계가 높을 수 밖에 없음

- child_num 제거 후 자녀유무를 나타내는 범주형 변수 'kids' 생성

(2) NumCards (카드의 개수) : 똑같은 사람이 begin_month를 기반으로 여러 카드를 발급받았다고 판단되는 경우

- 중복데이터 개수를 이용하여 카드의 개수를 나타내는 변수 'NumCards' 생성

(3) 시간변수의 재조정 : begin_month, DAYS_EMPLOYED, DAYS_BIRTH 과 같이 시간을 나타내는 변수의 경우

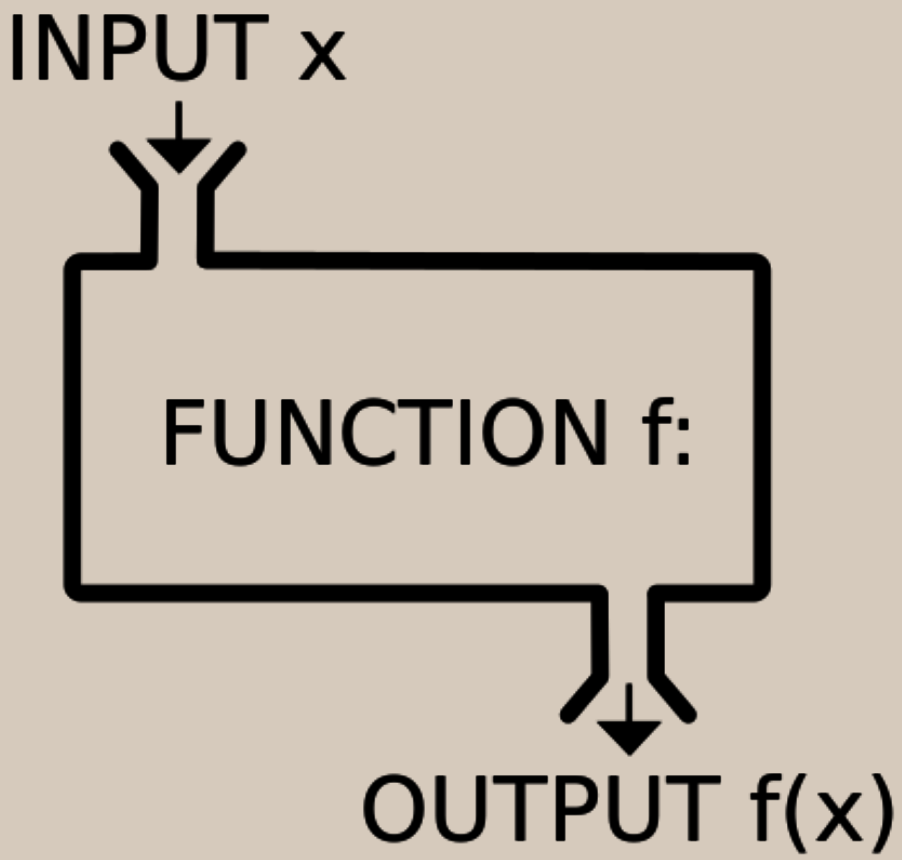
- 년 / 월 / 일 등을 이용하여 재표현

(4) ID : 직접 중복 데이터를 제거하는 대신, 모델이 스스로 중복 데이터임을 구별할 수 있도록 함

- 개인의 신상을 나타내는 여러 변수를 결합하여 생성

목표 : 언급된 여러 전처리를 적용해보고 비교하면서 최적의 성능을 내는 모델을 탐구

2. 평가지표에 대한 설명 및 모델링



2. 평가지표에 대한 설명 – log loss

Loss function : 모델의 출력값과 정답의 오차를 정의하는 함수

- 최종적으로 답으로만 성능을 평가할 경우, 얼만큼의 확률로 해당 답을 얻은 것인지 평가가 불가능.

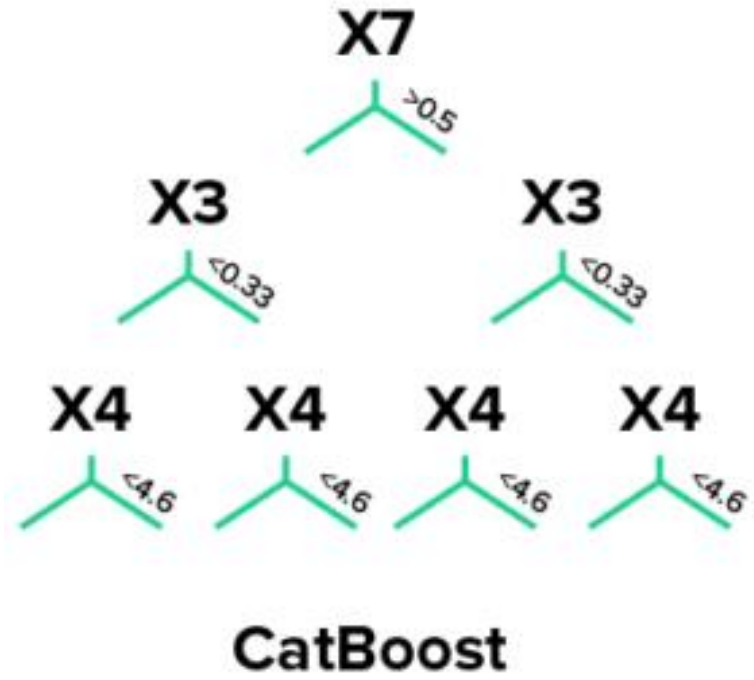
- Log loss는 모델이 예측한 확률 값을 직접적으로 반영하여 평가.

- 확률 값을 음의 log함수에 넣어 변환을 시킨 값으로 평가하여, 잘못 예측할 수록 페널티를 부여하기 위함.
  (100%의 확률 : -log(1.0) = 0 / 60%의 확률 : -log(0.6) = 0.51082)

- 답을 맞추더라도, 높은 확률로 답을 맞추는 모델을 찾기 위함. i.e. loss를 줄임

2. 모델 – CatBoost

속도 개선 로직과 정규화 방법을
보유하고 있는 Boosting 기반 모델



Ordered Boosting

- 데이터 일부만 가지고 잔차 계산을 한 뒤, 이를 이용하여 모델을 만들고, 그후 데이터 잔차는 이 모델로 예측한 값을 사용

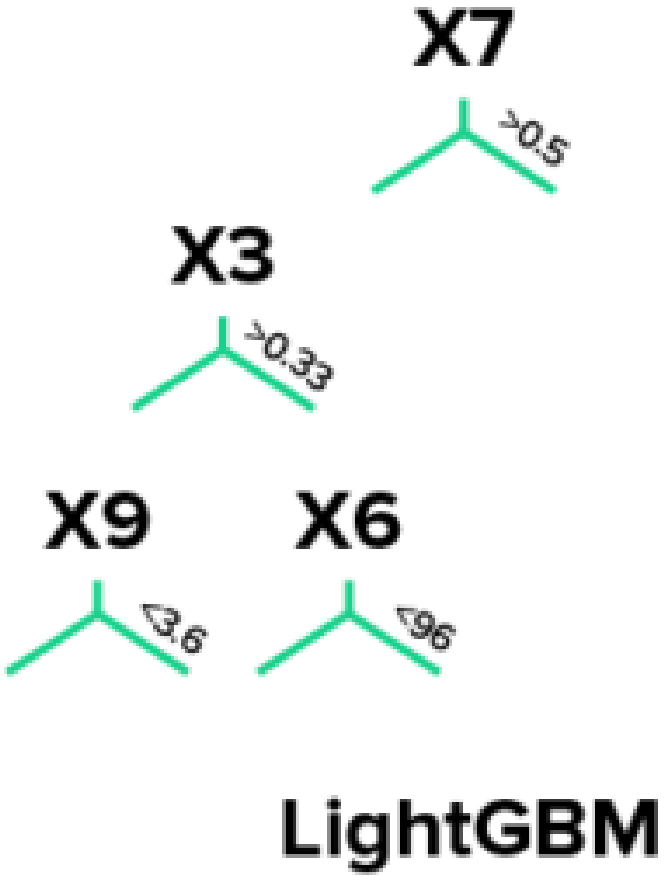
Categorial Feature Handling

- 각 단계별 다른 무작위 순열을 사용하여 과거 데이터로부터 평균을 내어 현재 target value를 추정

-동일한 대상을 대표하는 것으로 보이는 특징 검토 후 묶는 categorical feature combination 기법 활용

2. 모델 – LightGBM

Gradient Boosting 프레임워크로
Tree 기반 학습 알고리즘



Leaf Wise Tree 분할

- 최대 손실 값을 가지는 leaf node를 분할해 비대칭적인 tree 생성

GBM Boosting

- 틀린 부분에 가중치를 더하면서 진행하는 방식

- 정답지와 오답지 간의 차이를 훈련에 다시 투입하여 gradient를 적극 이용해서 모델을 개선하는 방식

2. 모델 – PyCaret을 이용하여 대략적인 성능 비교

*PyCaret 패키지 사용

	Model	Accuracy	AUC	Recall	Prec.	F1	Kappa	MCC	LogLoss	TT (Sec)
lightgbm	Light Gradient Boosting Machine	0.7117	0.7090	0.4389	0.7002	0.6431	0.2665	0.3419	0.7479	0.837
catboost	CatBoost Classifier	0.7121	0.7133	0.4501	0.6920	0.6509	0.2823	0.3434	0.7491	16.248
gbc	Gradient Boosting Classifier	0.7047	0.6705	0.4171	0.6814	0.6242	0.2293	0.3208	0.7742	9.941
lda	Linear Discriminant Analysis	0.6540	0.6201	0.3458	0.5858	0.5335	0.0357	0.0873	0.8442	0.216
nb	Naive Bayes	0.6579	0.6119	0.3450	0.5932	0.5317	0.0352	0.1133	0.8507	0.049
lr	Logistic Regression	0.6504	0.6064	0.3333	0.4230	0.5126	0.0000	0.0000	0.8529	1.439
dummy	Dummy Classifier	0.6504	0.5000	0.3333	0.4230	0.5126	0.0000	0.0000	0.8695	0.047
rf	Random Forest Classifier	0.7112	0.7513	0.5306	0.6862	0.6906	0.3732	0.3825	0.8722	3.541
ada	Ada Boost Classifier	0.7038	0.6378	0.4126	0.6577	0.6201	0.2214	0.3196	1.0767	1.052
et	Extra Trees Classifier	0.6825	0.7150	0.5066	0.6567	0.6630	0.3153	0.3224	1.4269	2.863
knn	K Neighbors Classifier	0.6084	0.6374	0.4401	0.5833	0.5929	0.1772	0.1793	4.0519	0.319
dt	Decision Tree Classifier	0.6183	0.6386	0.4872	0.6246	0.6213	0.2604	0.2606	13.1824	0.598
qda	Quadratic Discriminant Analysis	0.1216	0.5620	0.3380	0.5723	0.0356	0.0025	0.0213	30.2417	0.085

BEST!

Train Set을 10-fold CV를 이용하여 logloss 순으로 정렬 (LGBM logloss 0.7479)

2. 모델 – PyCaret을 이용하여 대략적인 성능 비교

눈 여겨 볼 점

- 앞에서 언급된 여러 전처리들을 활용하여 많은 시도를 해보았지만, PyCaret 패키지 결과상 유의미한 성능 변화는 없었음.

- 고전적인 Logistic Regression, LDA, Naïve Bayes 등에 비해 GBM, LGBM, CatBoost의 성능이 좋음

- Logloss 뿐만 아니라, F1 Score, Precision과 같은 평가지표에 따르면 Tree기반 부스팅 계열의 성능이 좋음

2. 모델 – PyCaret을 이용한 Voting Ensemble

*성능이 가장 좋았던 세 모델 (LGBM, GBM. CatBoost)을 활용하여 Soft Voting Ensemble

```
1 blend_tune = blend_models(estimator_list=tuned_best3, fold=10, optimize='logloss', method = 'soft')
```

Fold	Accuracy	AUC	Recall	Prec.	F1	Kappa	MCC	LogLoss
0	0.7161	0.7062	0.4596	0.7160	0.6577	0.2893	0.3551	0.7460
1	0.7204	0.7304	0.4564	0.7043	0.6593	0.3054	0.3666	0.7306
2	0.7155	0.7261	0.4620	0.6980	0.6598	0.3003	0.3548	0.7313
3	0.7198	0.7367	0.4602	0.7008	0.6612	0.3058	0.3670	0.7298
4	0.7149	0.7364	0.4511	0.6866	0.6536	0.2920	0.3520	0.7369
5	0.7112	0.7470	0.4503	0.6825	0.6512	0.2830	0.3417	0.7252
6	0.7088	0.7214	0.4526	0.6923	0.6502	0.2783	0.3345	0.7454
7	0.7259	0.7428	0.4672	0.7163	0.6679	0.3200	0.3852	0.7134
8	0.7221	0.7397	0.4706	0.7059	0.6682	0.3165	0.3740	0.7255
9	0.7111	0.7368	0.4584	0.7054	0.6536	0.2808	0.3408	0.7340
Mean	0.7166	0.7324	0.4589	0.7008	0.6583	0.2971	0.3572	0.7318
Std	0.0052	0.0113	0.0063	0.0107	0.0060	0.0140	0.0152	0.0092

Ensemble 및 Hyperparameter tuning 과정

	Model	Accuracy	AUC	Recall	Prec.	F1	Kappa	MCC	LogLoss
0	Voting Classifier	0.7155	0.728	0.4552	0.7014	0.6555	0.2911	0.3538	0.7338

Train Set 자체 split을 이용하여
logloss 산출 (Voting Ensemble logloss 0.7479)

=>앞의 단일 모델에 비해 logloss가 0.14정도 줄었음을 확인 가능

2. 모델 – CatBoost 단일 모델

*범주형 변수가 많고, 중복 데이터가 많아 이를 구별하기 위해, obs별 ID 를 만들 필요에 의해 CatBoost 단일모델 적합

NumCards	ID_personal	ID_income	ID_family	income_total	income_type	edu_type	family_type	house_type	DAYS_BIRTH	DAYS_EMPLOYED	credit
2	F_N_N_-13899	Commercial associate_Higher education_Unknown	Married_Municipal apartment_2.0_N	12.218495	Commercial associate	Higher education	Married	Municipal apartment	-13899	-4709	1.0
0	F_N_Y_-11380	Commercial associate_Secondary / secondary spe...	Civil marriage_House / apartment_3.0_Y	12.419166	Commercial associate	Secondary / secondary special	Civil marriage	House / apartment	-11380	-1540	1.0
5	M_Y_Y_-19087	Working_Higher education_Managers	Married_House / apartment_2.0_N	13.017003	Working	Higher education	Married	House / apartment	-19087	-4434	2.0

예시) ID 파생변수 생성

	Model	Accuracy	AUC	Recall	Prec.	F1	Kappa	MCC	LogLoss
0	CatBoost Classifier	0.7637	0.8817	0.5623	0.7402	0.7329	0.4828	0.5064	0.5782

Train Set 자체 split을 이용하여 logloss 산출한 결과 (logloss : 0.5782)

=〉 logloss가 확연히 줄었으나 overfitting을 의심해 보아야함

2. 모델 – CatBoost 단일 모델

눈 여겨 볼 점

- 범주형 처리에 유연한 CatBoost를 이용하여, 많은 파생변수를 만들어 fitting 시도

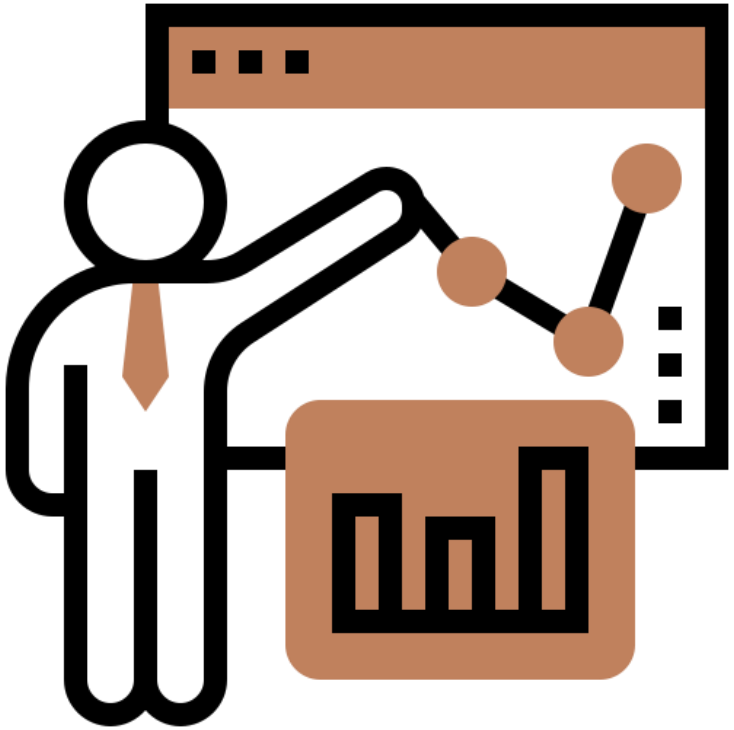
- CatBoost의 경우 hyperparameter을 tuning 하였을 때, 오히려 성능 하락

- Train set 자체 Split을 이용하여 logloss를 산출한 결과, logloss가 확연히 줄었으나 overfitting을 의심 필요

- 실제로 Dacon 제출결과 logloss Score가 1.11로 overfitting이 매우 심함

- 파생변수가 많고 복잡하여, 모델의 complexity가 증가했기 때문에 과적합 됨

3. 결과



결과- 결과 요약

LightGBM 단일 모델

=〉 가장 성능이 좋았던 모델

전처리 방식

=〉 사용자를 식별할 수 있는 고유 id와 NumCards 등 파생 변수의 추가

=〉 occyp_type의 결측치를 최빈값으로 처리

=〉 child_num과 family_size feature의 이상치를 제거

결과- 한계점

=〉 모든 feature들이 같고, credit이 다른 경우가 다수 존재

⇒ 16개의 feature가 같다는 점에서 동일인물이라 볼 수 있겠지만, 동일인물임에도 target이 다르다는 것은 데이터의 큰 결함

한계점

Train Data의 근본적 문제점

⇒ 특히, DAYS_BIRTH (데이터 수집일로부터 출생일까지 날짜 수)가 같다는 점에서, 동일인물의 credit변화라고 볼 수 없음.
(시계열 자료일 경우, 동일인물이어도 데이터 수집일이 달라야한다는 점에서, DAYS_BIRTH가 달라야 함)

⇒ 해당 자료를 전부 drop 시키기에는 Data의 손실이 크고, 일부만 drop 시키기에는 관측자의 임의성이 반영된다는 점에서 문제가 되며 학습이 어려움

감사합니다