신용카드 연체 예측

우리가 평균 연령 막내 조?

1조 임상훈, 윤영주, 조은서, 채윤길

Contents

프로젝트 개요

주제

Dacon [신용카드 사용자 연체 예측 Ai 경진 대회]

기간

2021년 10월 18일 ~ 2021년 11월 12일

훈련내용

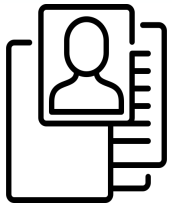
고객의 개인정보를 이용하여 신용점수 등급 분류 데이터 전처리/모델링/시각화

목표

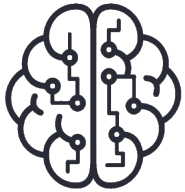
신규 사용자 혹은 재가입자의 향후 채무 불이행 여부 예측

I. 프로젝트 설명

고객의 개인정보 데이터를 이용하여 ,
카드 신청자의 향후 신용카드 대금연체에 따른 신용 등급을 예측하는 모델링을 제시

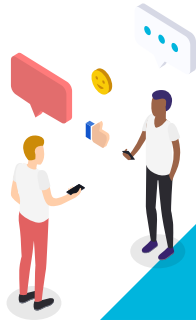


사용자 개인 정보



인공지능 모델

신용등급 예측



Ⅱ. 프로젝트 팀 구성 및 역할

임상훈

프로젝트 기획

윤영주

데이터 EDA
전처리

조은서

전처리
모델링

채윤길

모델링

III. 프로젝트 수행 절차

2021/10/18 ~
2021/10/23

프로젝트 주제 이해,
방향성 수립,
데이터 이해 및 EDA

2021/11/3~

데이터 전처리,
(이상치 처리, 정제, 파생변수 생성)
모델 탐색

2021/11/11~

모델링,
성능 개선을 위한
하이퍼파라미터 조정,
발표 준비

EDA

Exploratory Data Analysis

EDA

데이터 소개

Feature Column

Gender	남/여 구분	Work Phone	회사 휴대폰 소유 여부	Edu Type	최종 학력	Family Size	최종 학력	Child Num	자녀 숫자
Car	차량 소유 여부	Phone	전화 소유 여부	Family Type	가족 구성	Begin Month	카드 발급 월 수	Income Total	총 수입
Reality	부동산 소유 여부	Email	메일 소유 여부	House Type	생활 방식	Days Birth	태어난 일 수		
Flag Mobil	개인 휴대폰 소유 여부	Occyp Type	직업 유형	Income Type	소득 분류	Days Employed	고용 일 수		

Binary Category Multi Category Numeric Value

Label Column

Credit

0 - 가장 좋은 등급의 신용도

1 - 두 번째로 좋은 등급의 신용도

2 - 가장 안 좋은 등급의 신용도

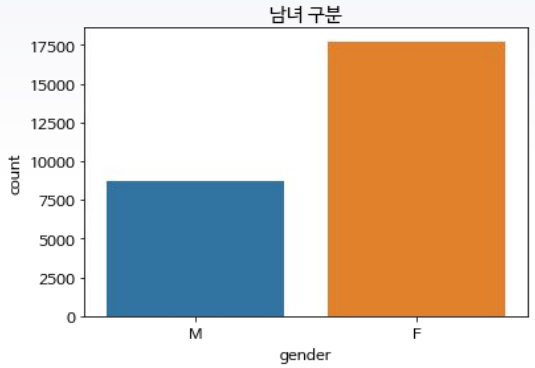
```
1 df.shape
```

(26457, 20)

EDA

데이터 탐색

1.




```
df.groupby(['gender'])['income_total'].mean()
```

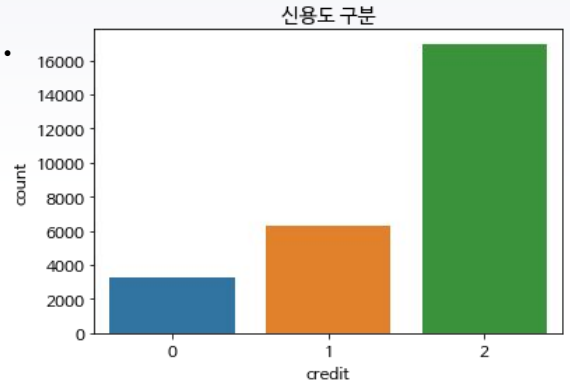
gender	mean_income
F	173158.791196
M	216000.336328

2.

EDA

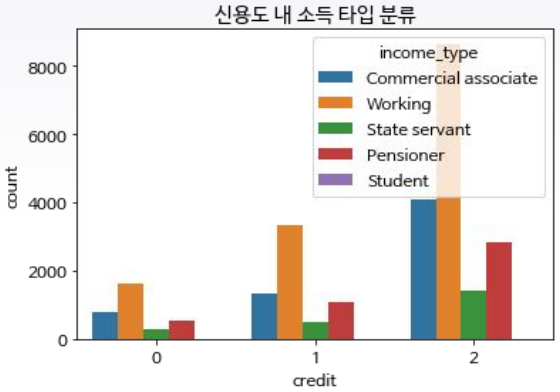
데이터 탐색

1.



credit	count
0.0	3222
1.0	6267
2.0	16968

2.

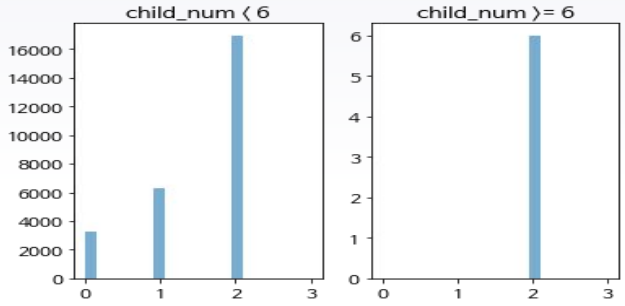


income_type	credit	count
Commercial associate	0.0	782
Commercial associate	1.0	1344
Commercial associate	2.0	4076
Pensioner	0.0	536
Pensioner	1.0	1084
Pensioner	2.0	2829
State servant	0.0	265
State servant	1.0	489
State servant	2.0	1400
Student	1.0	3
Student	2.0	4
Working	0.0	1639
Working	1.0	3347
Working	2.0	8659

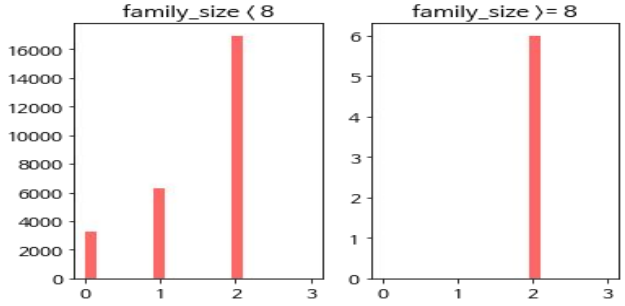
EDA

데이터 탐색

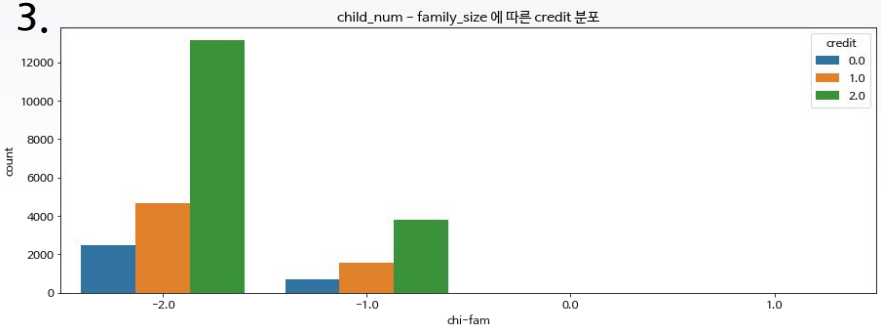
1.



2.



3.

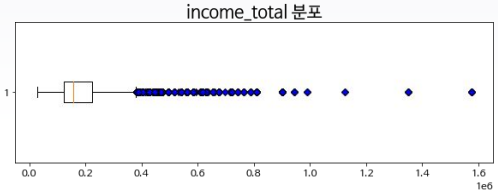


credit	0.0	1.0	2.0
chi-fam			
-2.0	2501.0	4679.0	13151.0
-1.0	721.0	1588.0	3811.0
0.0	0.0	0.0	5.0
1.0	0.0	0.0	1.0

EDA

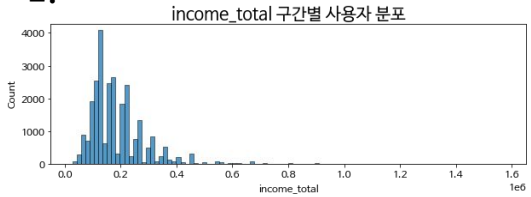
데이터 탐색

1.



index	income_total
count	26457
mean	187307
std	101878
min	27000
25%	121500
50%	157500
75%	225000
max	1575000

2.



3. Income total outlier 처리 전

credit	count	mean	std	min	25%	50%	75%	max
0.0	3222.0	190807.582402	102139.548208	29250.0	126000.0	162000.0	225000.0	945000.0
1.0	6267.0	181122.701532	92548.262258	27000.0	117000.0	157500.0	225000.0	1125000.0
2.0	16968.0	188925.666991	104981.727565	27000.0	121500.0	157500.0	225000.0	1575000.0

4. Income total outlier 처리 후

credit	count	mean	std	min	25%	50%	75%	max
0.0	3222.0	185254.509777	82918.399772	29250.0	126000.0	162000.0	225000.0	380250.0
1.0	6267.0	177519.613930	78813.358972	27000.0	117000.0	157500.0	225000.0	380250.0
2.0	16968.0	183049.022896	81963.662707	27000.0	121500.0	157500.0	225000.0	380250.0

Data Preprocessing

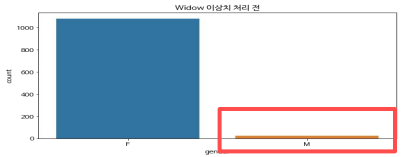
Data Preprocessing

이상치 처리

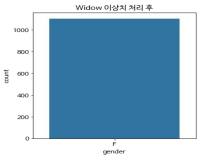
- 과부를 뜻하는 Widow는 Male이 존재할 수 없으므로 변경

```
df.loc[(df['family_type']=='Widow') & (df['gender']=='M'),'gender'] = 'F'
```

gender	count
F	1080
M	23



gender	count
F	1103



- Family_size보다 child_num이 큰 있을 수 없는 경우의 이상치 존재

child_num	family_size
2	1.0

```
df.loc[df['chi-fam'] > 0 , ['chi-fam']] = 0
```

Data Preprocessing

결측치 처리

- occyp_type 변수에만 결측치 존재

```
df.loc[pd.isna(df['occyp_type']),['occyp_type']]='NaN'
```

DAYS_BIRTH	DAYS_EMPLOYED	FLAG_MOBIL	work_phone	phone	email	occyp_type	family_size	begin_month	credit
-13899	-4709	1	0	0	0	NaN	2.0	-6.0	1.0
-15131	-1466	1	0	0	1	NaN	3.0	-38.0	2.0
-22922	365243	1	0	1	0	NaN	1.0	-41.0	2.0
-23113	365243	1	0	0	0	NaN	2.0	-37.0	2.0
-13727	-6031	1	0	0	0	NaN	2.0	-7.0	2.0
...	...	...	...	...	...	...	...	...	...

Data Preprocessing

범주형 변수 처리

- Label Encoder: *순서를 고려하지 않아도 되는* 범주형 변수들을 수치화할 때 쓰는 함수
  - ‘Income_type’ , ‘family_type’, ‘house_type’

```
from sklearn.preprocessing import LabelEncoder
le = LabelEncoder()
df['occyp_type_le'] = le.fit_transform(df['occyp_type'])
df['house_type_le'] = le.fit_transform(df['house_type'])
df['family_type_le'] = le.fit_transform(df['family_type'])
df['income_type_le'] = le.fit_transform(df['income_type'])
```

- Ordinal Encoder: 순서가 있는 범주형 변수들을 수치화할 때 쓰는 함수
  - ‘Edu_type’

```
edu_dict = {'Lower secondary':0,  'Secondary / secondary special':0,
            'Incomplete higher':1,
            'Higher education':2, 'Academic degree':2}

df['edu_type_le'] = df.edu_type.map(edu_dict)
```

Data Preprocessing

변수 제거 및 변환

- 모든 행이 1이므로 의미 없기에 삭제

```
df['FLAG_MOBIL'].unique()
```

```
array([1])
```

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

- 음수인 값 양수로 변환

```
feats = ['DAYS_BIRTH', 'begin_month', 'DAYS_EMPLOYED']
for feat in feats:
    df[feat]=np.abs(df[feat])
```

Data Preprocessing

중복 데이터 처리

'begin_month'와 'credit' 변수를 제외한 중복 데이터가 절반 이상

- 동일 인물이 다른 시기에 카드를 개설했다고 가정
- 한 행으로 줄이는 대신 'card_count (발급 카드 개수)'와 'credit_change_count (신용 변화 횟수)' 열 추가

동일 인물이 카드를 발급하는 시기에 따라 'credit'이 다를 수 있으므로
가장 최신의 카드를 발급했을 당시의 credit을 남겨두고 나머지 데이터를 제거

	begin_month	credit	chi-fam	occyp_type_le	house_type_le	family_type_le	income_type_le	edu_type_le	Age	career_year
3134	7.0	1.0	-2.0	3	1	1	4	2	37	6.0
342	12.0	0.0	-2.0	3	1	1	4	2	37	6.0
4118	12.0	1.0	-2.0	3	1	1	4	2	37	6.0
7195	35.0	2.0	-2.0	3	1	1	4	2	37	6.0
9895	42.0	2.0	-2.0	3	1	1	4	2	37	6.0
13677	42.0	0.0	-2.0	3	1	1	4	2	37	6.0
22155	49.0	0.0	-2.0	3	1	1	4	2	37	6.0

```
2 df_.shape

(26457, → (8759, 21)
```

Data Preprocessing

파생 변수 생성

- ‘Age’: DAYS_BIRTH / 365
- ‘career_year’: DAYS_EMPLOYED / 365
- ‘career_start_age’ : Age - career_year
- ‘Ability’: Income_total / (DAYS_BIRTH + DAYS_EMPLOYED)
- ‘Ability2’ : Income_total / DAYS_EMPLOYED
- ‘Card_count’: 한 사람이 생성한 카드의 개수
- ‘credit_change_count’ :한 사람의 신용도 변화 횟수
- ‘Chi-fam’: Child_num - family_size

DAYS_BIRTH	DAYS_EMPLOYED	Age	career_year
11153	1281	30	4.0
15964	385	43	2.0
11627	1449	31	4.0
15262	1674	41	5.0
11509	3506	31	10.0

Data Preprocessing

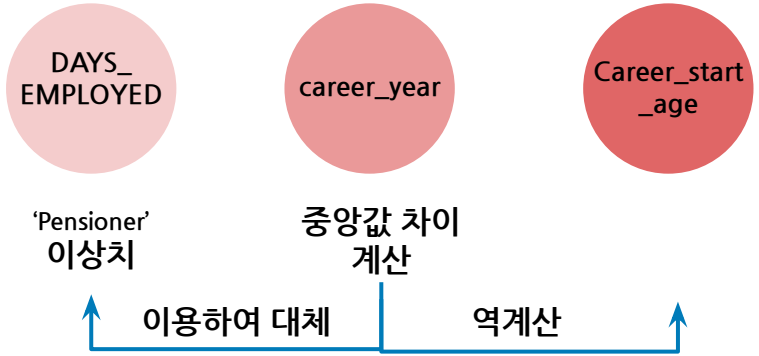
이상치 처리

'income_type' 변수값이 'Pensioner'인 행의 'DAYS_EMPLOYED' 변수값이 대부분 이상치(365243일)로 설정되어 있음
=> 이 이상치를 'Pensioner'가 아닌 사람들의 중앙값으로 대체

	income_type	DAYS_EMPLOYED
10	Working	-2213
11	Commercial associate	-91
12	Commercial associate	-2162
13	Working	-2474
14	Pensioner	365243
15	Working	-4056
16	Working	-4553
17	Commercial associate	-984
18	Pensioner	365243

1. ('Pensioner'의 'Age'의 중앙값) - ('Pensioner'가 아닌 사람들 'career_start_age'의 중앙값)을 'Pensioner'의 'career_year'로 대체
2. 새롭게 생성된 'Pensioner'의 'career_year'에 맞게 그들의 'career_start_year'와 'DAYS_EMPLOYED' 변수를 역으로 계산하여 변환

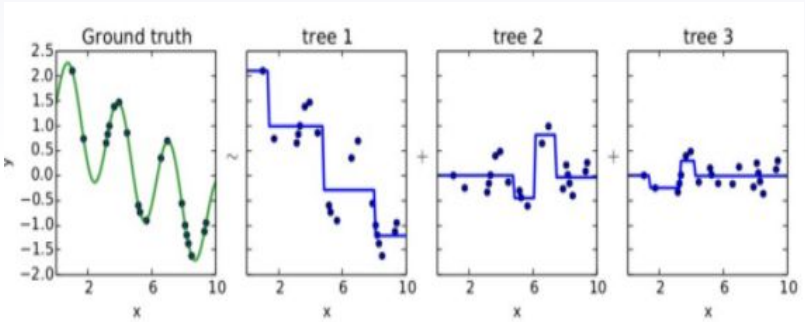
- 'career_year': DAYS_EMPLOYED / 365
- 'career_start_age' : Age - career_year



Modeling

Modeling

GBM(Gradient Boosting Machine)의 개념

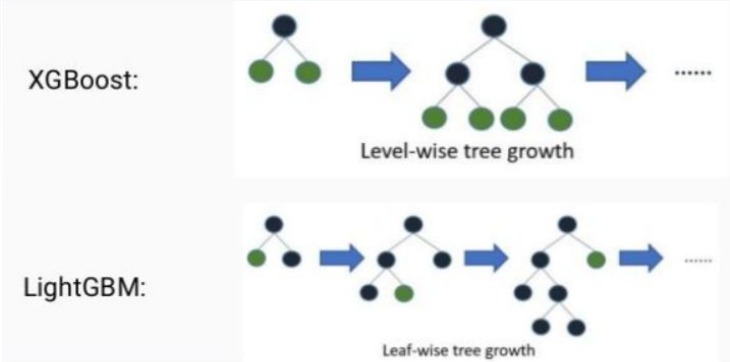


- tree1을 통해 예측하고 남은 잔차를 tree2를 통해 예측하고 이것을 반복하여 점차 잔차를 줄여나가는 방식.
- 이런 식으로 계속 학습이 되면 training set을 잘 설명하는 예측 모형이 만들어지지만 over-fitting & run-time이 오래 걸릴 수 있다는 단점.

  ➔ 이 단점을 보완한 것이 lightGBM

Modeling

LGBM(light gradient boosting machine) 개념



- 기존 boosting 모델들이 tree depth(깊이)를 줄이기 위해 level wise(균형 트리)분할을 사용하는 것과 다르게 LGBM은 leaf node를 지속적으로 분할하면서 진행
- 비대칭적이고 깊은 트리가 생성되지만 동일한 leaf를 생성할 때 level wise보다 손실을 줄일 수 있고 연산이 추가되는 것을 방지할 수 있다는 장점

  ➔ LGBM은 깊이 우선 GBM은 너비 우선

Modeling

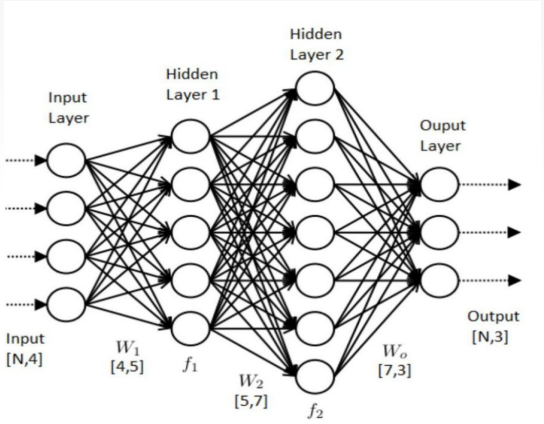
CATBOOST 개념

- GBM의 한계점인 과적합과 저속 학습 속도를 개선하는데 집중하여 개발된 모델

- CatBoost는 다른 모델들과 달리 대칭 트리 구현 ➔ runtime 단축
- Cat은 Categorical의 약자를 의미하며, 범주형 변수에 OneHot-encoding이 필요한 모델들과 달리, 모든 범주형 특징을 벡터화하기에, OneHot-encoding이 따로 필요없다는 장점

Modeling

ANN(Artificial Neural Network) 개념



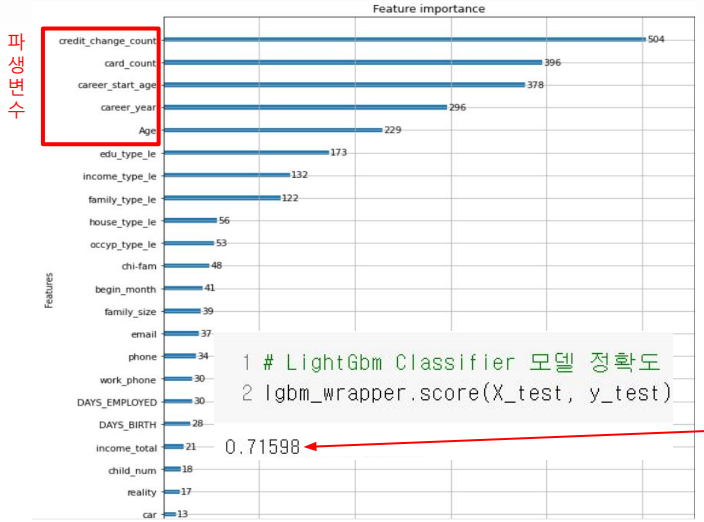
- 사람의 신경망 원리와 구조를 모방하여 적용한 모델
- 입력층에서 출력층으로 진행시, hidden layer(은닉층)에서 weight와 bias에 의해 값이 훈련되는 모델
  ➔ weight와 bias 설정이 매우 중요
  ➔ 최적의 파라미터 값을 찾기 힘들고, 학습시간이 너무 느리다는 한계가 존재.

Modeling

LGBM 모델링 결과

- LGBM (다중분류)

LGBM Features Importance/Confusion Matrix




```
evals = [(X_test, y_test)]
lgbm_wrapper.fit(X_train, y_train, early_stopping_rounds=100,
                 eval_metric="multi_logloss",
                 eval_set=evals, verbose=True)
preds = lgbm_wrapper.predict(X_test)
```

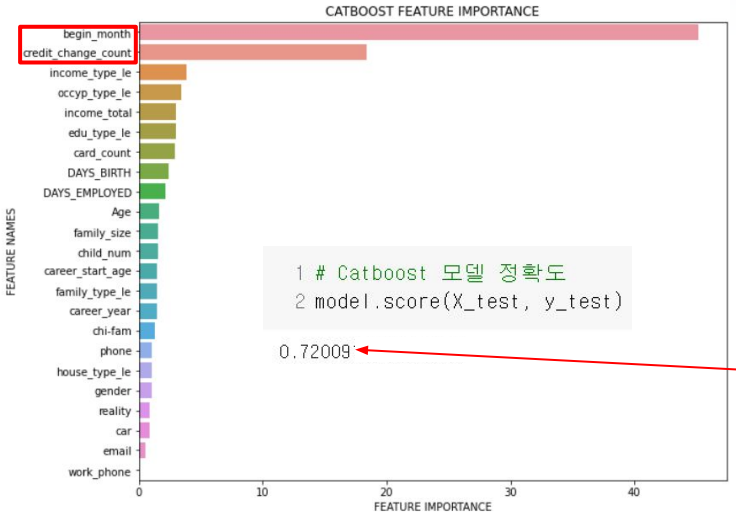
	precision	recall	f1-score	support
0.0	0.25	0.01	0.02	307
1.0	0.73	0.49	0.59	572
2.0	0.72	0.98	0.83	1311
accuracy			0.72	2190
macro avg	0.57	0.49	0.48	2190
weighted avg	0.65	0.72	0.65	2190

Modeling

CATBOOST 모델링 결과

- CATBOOST (다중분류)

CATBOOST Features Importance/Confusion Matrix




```
model = cb.CatBoostClassifier(loss_function='MultiClass', eval_metric='Accuracy')
```

```
grid = {'depth': [4,6,10],
'l2_leaf_reg': [1, 3, 5,],
'iterations': [5, 10, 30]}
```

```
model.grid_search(grid,train_dataset)
```

```
1 # Catboost 모델 정확도
2 model.score(X_test, y_test)
```

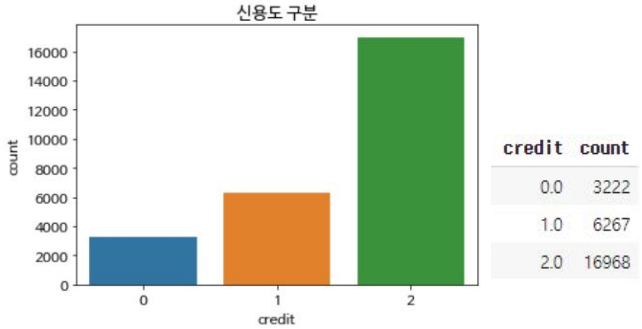
0.72009

	precision	recall	f1-score	support
0.0	0.00	0.00	0.00	307
1.0	0.72	0.49	0.58	572
2.0	0.71	0.98	0.83	1311
accuracy			0.72	2190
macro avg	0.48	0.49	0.47	2190
weighted avg	0.62	0.72	0.65	2190

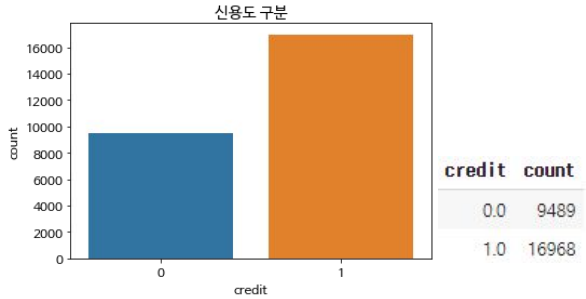
Modeling

다중분류를 이진분류로 바꾸어 진행

- 신용도를 예측하여 연체 가능성이 높은 사용자를 제대로 필터링 하는것이 본 프로젝트의 목표.
- Credit 등급 0,1을 합친 사용자가 2에 비해 현저히 적은 데이터 불균형 문제가 있었음



credit	count
0.0	3222
1.0	6267
2.0	16968



credit	count
0.0	9489
1.0	16968

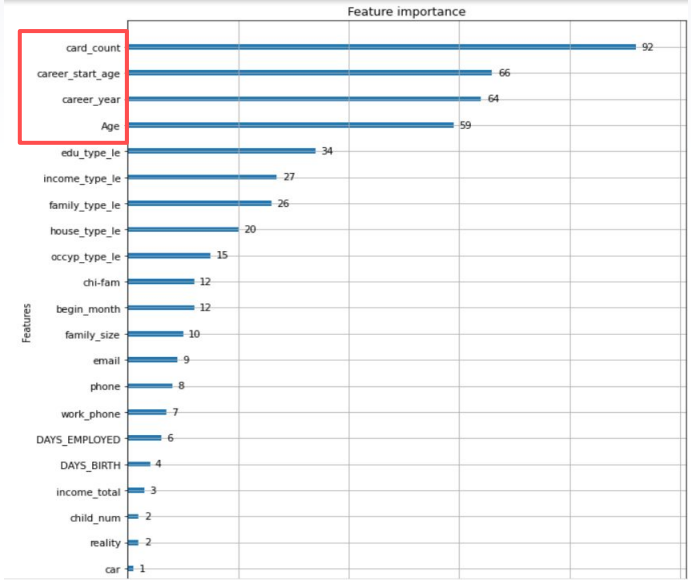
Modeling

LGBM (light gradient boosting machine) 모델링 결과

- **LGBM (이진분류)**

LGBM Features Importance/Confusion Matrix

```
오차행렬:
 [[   0  307]
 [   0 1883]]

정확도: 0.8598
정밀도: 0.8598
재현율: 1.0000
F1: 0.9246
AUC: 0.5000
```



Modeling

ANN(Artificial Neural Network) 모델링 결과

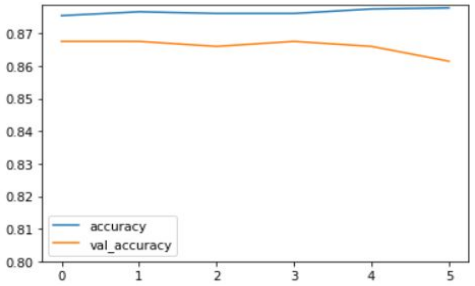
- **ANN (이진분류)**

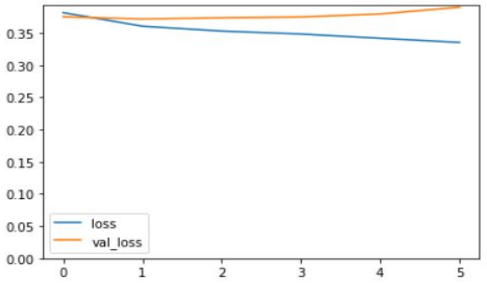
```
model = Sequential()
model.add(Dense(256, activation='relu', input_dim=X.shape[1]))
model.add(Dense(128, activation='relu'))
model.add(Dense(1, activation='sigmoid'))
model.summary()
```

```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 dense (Dense)               (None, 256)               6144

 dense_1 (Dense)             (None, 128)               32896

 dense_2 (Dense)             (None, 1)                 129

=================================================================
Total params: 39,169
Trainable params: 39,169
Non-trainable params: 0
_________________________________________________________________
```

```
model.compile(optimizer='adam', loss='binary_crossentropy', metrics=['accuracy'])
from tensorflow.keras.callbacks import EarlyStopping, ModelCheckpoint
es = EarlyStopping(monitor='val_accuracy', patience=5, restore_best_weights=False)
mc = ModelCheckpoint(filepath='best_model.h5', monitor='val_accuracy', save_best_only=True)
```

ANN Accuracy / Loss






```
1 # ANN 모델 정확도
2 np.round(score[1],4)
```

```
0.8548
```

Modeling

Adapted Model Accuracy Comparison

	다중분류	이진분류
ANN		0.8548
LGBM	0.7160	0.8598
CATBOOST	0.7201	

Ⅳ. 프로젝트 결론/한계점

연체 위험성 사전 감지

카드사는 위 예측 모델을 통해서, 신규/추가 신용카드 발급자의 CREDIT을 산정할수 있으며,

향후의 연체 위험성을 사전에 감지할 수 있는 기능을 갖고있음.

데이터 부족

데이터 전처리 과정에서 중복, 결측치, 이상치를 제외한 데이터는 사실상 8천여개 정도 확인되었음.

이는 예측 모델 학습 관점에서 **충분하지 않은 데이터 양**이기 때문에 정확한 분석에 한계점이 있음.

매출 손실 방지 및 성실 고객 확보

연체 가능성이 높은 사용자 군집에 안내문을 보내는 등 다양한 비지니스 방법으로 카드사의 연체로 인한 **매출손실 방지 및 성실 고객을 확보**할수 있음.

외부 데이터 활용 한계

외부 변수 활용으로 모델의 성능을 높이고자 하였으나, 개인정보이기 때문에 **활용할 외부 변수가 많지 않았음**

THANKS!