CREDIT PREDICTION

신용카드 사용자 연체 예측

18기 분석 문다정
18기 분석 이소연
18기 분석 한상범
18기 엔지니어링 김인섭

CREDIT PREDICTION

INDEX

CREDIT PREDICTION

01

프로젝트 소개

❶ 주제 소개
❷ 데이터 및 변수.

01

주제 소개

- 프로젝트 주제 : 신용카드 사용자 연체 예측
- 목적 : 가장 좋은 분류예측을 하는 모델 파악을 통한 신용카드 사용자 연체 예측
- 사용 데이터 : dacon 신용카드 사용자 개인 신상정보
- 사용 방법론 : LGBM, CatBoost, XGBoost
- 성능 평가 : Log loss함수

01

데이터 및 변수

GENDER,
CAR,
REALTY,
FLAG
MOBIL,
WORK
PHONE,
PHONE,
EMAIL

OCCUPY
TYPE,
INCOME
TYPE,
EDU
TYPE,
FAMILY
TYPE,
HOUSE
TYPE

INCOME
TOOTAL,
FAMILY
SIZE,
BEGIN
MONTH,
DAYS
BIRTH,
DAYS
EMPLOYED,
CHILD NUM

01

데이터 및 변수

0 가장 좋은 등급 신용도

1 2번째로 좋은 등급 신용도

2 3번째로 좋은 등급 신용도

CREDIT PREDICTION

02

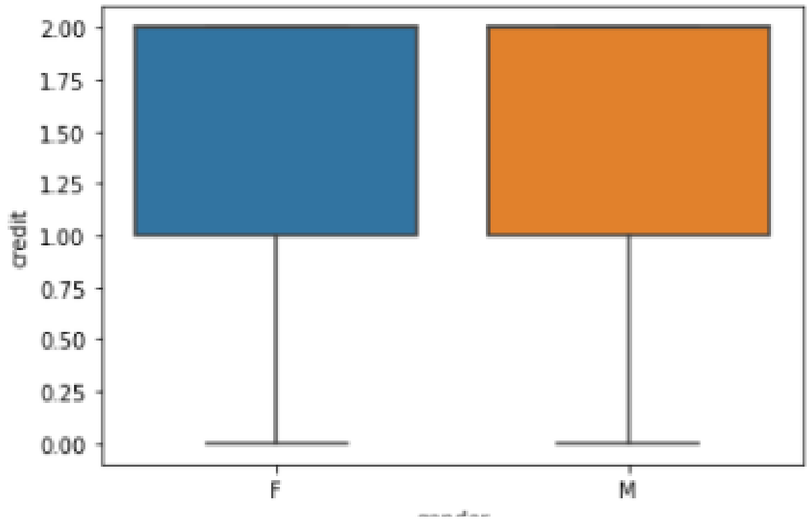
데이터 전처리

02

데이터 전처리

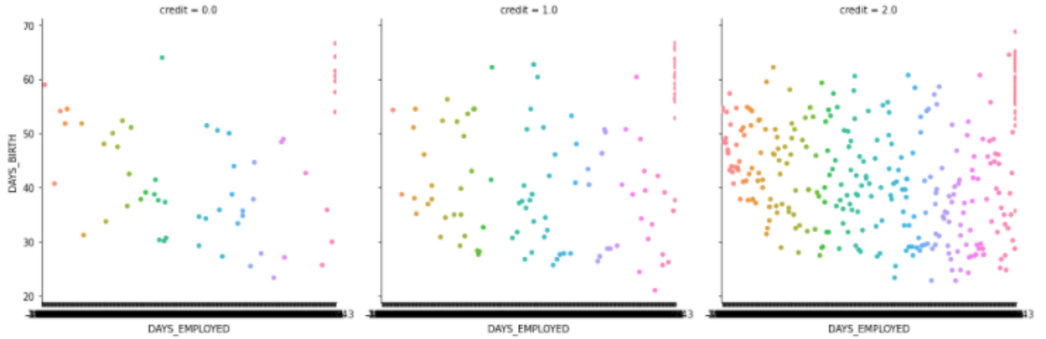
```
# 결측치 확인
df_Train.isnull().sum()
```

```
index              0
gender             0
car                0
reality            0
child_num          0
income_total       0
income_type        0
edu_type           0
family_type        0
house_type         0
DAYS_BIRTH         0
DAYS_EMPLOYED      0
FLAG_MOBIL         0
work_phone         0
phone              0
email              0
occyp_type      8171
family_size        0
begin_month        0
credit             0
```




```
sns.boxplot(x='gender', y='credit', data=df_Train)
print(df_Train[df_Train.gender=='F'].credit.describe())
print()
print(df_Train[df_Train.gender=='M'].credit.describe())

#유의미한 차이는 없음
```

```
rand = np.random.choice(df_Train.shape[0], 500)

# 1. 전반적으로 나이가 어림
# 2. 나이가 많을수록 업무일 수도 많음
sns.catplot(x = 'DAYS_EMPLOYED', y = 'DAYS_BIRTH', data = df_Train.loc[rand], col = 'credit', col_wra
plt.show()
```



02

데이터 전처리

GENDER,
CAR,
REALTY,
FLAG
MOBIL,
WORK
PHONE,
PHONE,
EMAIL

BINARY CATEGORY

```
:  1  # 초록색 전처리
   2  df_Train = pd.get_dummies(df_Train, columns = ['gender','car','reality'])
```

```
In [17]:  1  train['FLAG_MOBIL'].value_counts() # 모두 똑같이 다 1개를 갖고있다.

Out[17]: 1    26457
         Name: FLAG_MOBIL, dtype: int64
```

02

데이터 전처리

MULTI CATEGORY

OCCUPY TYPE, INCOME TYPE, EDU TYPE, FAMILY TYPE, HOUSE TYPE

:

```
label_encoder=preprocessing.LabelEncoder()
df_Train['income_type']=label_encoder.fit_transform(df_Train['income_type'])
##################################################################################
df_Train['edu_type']=label_encoder.fit_transform(df_Train['edu_type'])
##################################################################################
df_Train['family_type']=label_encoder.fit_transform(df_Train['family_type'])
##################################################################################
df_Train['house_type']=label_encoder.fit_transform(df_Train['house_type'])
##################################################################################
df_Train['occyp_type']=label_encoder.fit_transform(df_Train['occyp_type'])
```

02

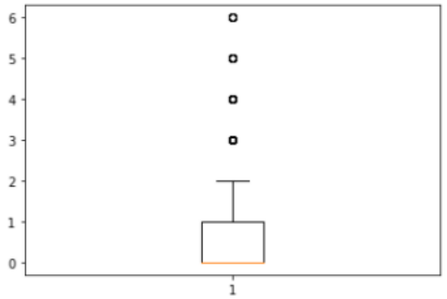
데이터 전처리

NUMERICAL VALUE

INCOME TOOTAL,
FAMILY SIZE,
BEGIN MONTH,
DAYS BIRTH,
DAYS EMPLOYED,
CHILD NUM

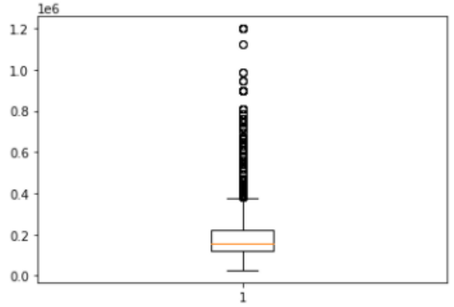
```
plt.boxplot(df.child_num)

#결국 자녀수가 7 이상인 샘플은 이상치로 판단되고, 6으로 치환해준다.
```

```
{'whiskers': [<matplotlib.lines.Line2D at 0x1e1b00e0250>,
  <matplotlib.lines.Line2D at 0x1e1b00e05e0>],
 'caps': [<matplotlib.lines.Line2D at 0x1e1b00e0970>,
  <matplotlib.lines.Line2D at 0x1e1b00e0d00>],
 'boxes': [<matplotlib.lines.Line2D at 0x1e1b00d0e80>],
 'medians': [<matplotlib.lines.Line2D at 0x1e1b00eb0d0>],
 'fliers': [<matplotlib.lines.Line2D at 0x1e1b00eb460>],
 'means': []}
```




```
In [7]: plt.boxplot(df.income_total)
```

```
Out[7]: {'whiskers': [<matplotlib.lines.Line2D at 0x1e1af3e8910>,
  <matplotlib.lines.Line2D at 0x1e1af3e8be0>],
 'caps': [<matplotlib.lines.Line2D at 0x1e1af3e8f70>,
  <matplotlib.lines.Line2D at 0x1e1af462340>],
 'boxes': [<matplotlib.lines.Line2D at 0x1e1af3d9370>],
 'medians': [<matplotlib.lines.Line2D at 0x1e1af4626d0>],
 'fliers': [<matplotlib.lines.Line2D at 0x1e1af462a60>],
 'means': []}
```




```
In [8]: df.at[df.income_total>1200000,'income_total'] = 1200000
```

```
df.at[df.income_total>1200000,'income_total'] = 1200000 #연간소득
df.at[df.family_size>8, 'family_size'] = 8              #가족규모
df.begin_month = (-1)*df.begin_month                    #신용카드발급월
df.DAYS_BIRTH = (-1)*df.DAYS_BIRTH /365                 #출생일
df.at[df.DAYS_EMPLOYED>0,'DAYS_EMPLOYED'] = 0           #업무시작일
df.DAYS_EMPLOYED = (-1)*df.DAYS_EMPLOYED/30
df.at[df.child_num>=6, 'child_num'] = 6                 #자녀수
```

CREDIT PREDICTION

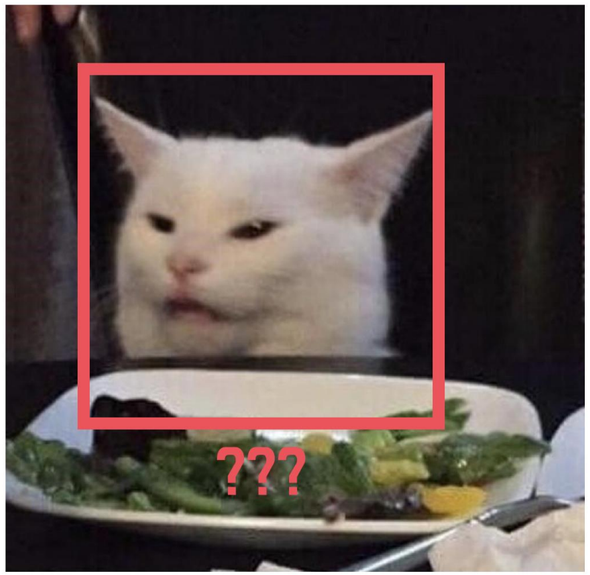
03

모델링 & 성능평가

❶ 성능평가 지표 : Log loss
❷ 모델링 결과

03 성능평가지표 : Log loss

1. 다음 사진의 동물로 알맞은 것은 무엇인가요?



① 고양이
② 사자
③ 강아지
④ 기린
⑤ 래서팬더

1. 영희	
① 고양이	0.9
② 사자	0.1
③ 강아지	0
④ 기린	0
⑤ 래서팬더	0

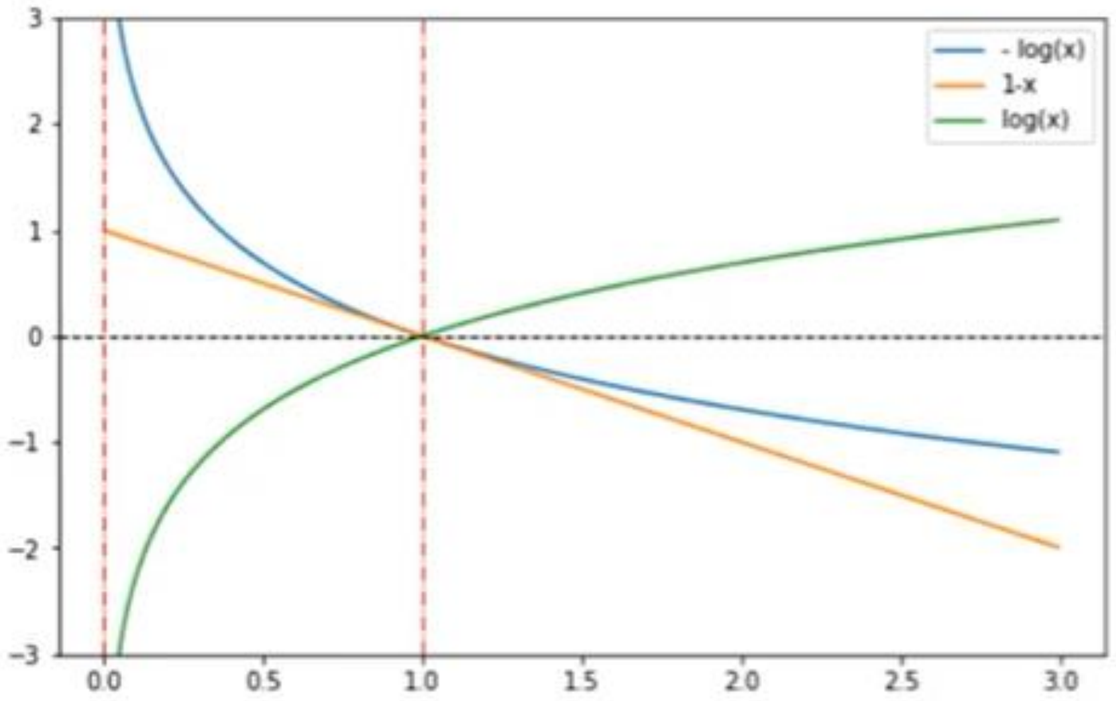
2. 철수	
① 고양이	0.4
② 사자	0.15
③ 강아지	0.15
④ 기린	0.15
⑤ 래서팬더	0.15

둘 다 정답이지만, 더 높은 확률로
정답을 예측한 영희에게 더 높은
점수를 주는 것 = **Log loss**

logloss 알아보기 (velog.io)

03

성능평가지표 : Log loss



파란색 음의 로그 함수

확률 1 일 때, -log(1.0) = 0

확률 0.8 일 때, -log(0.8) = 0.22314

확률 0.6 일 때, -log(0.6) = 0.51082

Log loss => 낮을수록 좋은 모델

03

성능평가지표 : Log loss

Log loss : 다중 클래스 분류 모델 (target 3개 이상)을 평가하는 방법

이진 분류 과제

에서 주로 사용하는 성능 평가지표

Accuracy (정확도), precision (정밀도), recall (재현율), f1 score 등

다항 분류 과제 √

성능평가지료 계산을 위해 3*3 정오행렬이 생성된다.

예측이 틀린 정도에 따라 패널티가 바뀌는 지표를 사용하여 판단하고자 했다.

=> Log Loss를 성능평가지표로 선택

03

모델링 결과

LightGBM

0.73212

(1) Light GBM

```
from lightgbm import LGBMClassifier

lgbm_model = LGBMClassifier(n_estimators=400)
np.random.seed(2020)
lgbm_model.fit(train_x, train_y,verbose=True)
result_lgbm = lgbm_model.predict_proba(test_x)
```

```
result_lgbm
```

```
array([[0.04215235, 0.11727961, 0.84056804],
       [0.2554878 , 0.13525832, 0.60925388],
       [0.03598366, 0.05209257, 0.91192377],
       ...,
       [0.02640005, 0.0967554 , 0.87684456],
       [0.09001884, 0.18555992, 0.72442124],
       [0.07369236, 0.3162416 , 0.61006604]])
```

```
submi_lgbm = pd.DataFrame(result_lgbm)
submi_lgbm['index'] = np.arange(26457, 36457)
submi_lgbm = submi_lgbm[['index',0,1,2]]
submi_lgbm
```

	index	0	1	2
0	26457	0.042152	0.117280	0.840568
1	26458	0.255488	0.135258	0.609254
2	26459	0.035984	0.052093	0.911924
3	26460	0.137361	0.140733	0.721906
4	26461	0.087158	0.128532	0.784309
...	...	...	...	...
9995	36452	0.098218	0.197139	0.704643
9996	36453	0.148934	0.260566	0.590501
9997	36454	0.026400	0.096755	0.876845
9998	36455	0.090019	0.185560	0.724421
9999	36456	0.073692	0.316242	0.610066

10000 rows × 4 columns

03 모델링 결과

(2) Cat Boost

```
from catboost import CatBoostClassifier

cat = CatBoostClassifier(learning_rate=0.05, iterations=5000)
np.random.seed(2020)
cat.fit(train_x, train_y, early_stopping_rounds=100, verbose=300)
result = cat.predict_proba(test_x)
```

```
0:      learn: 1.0697049        total: 171ms    remaining: 14m 16s
300:    learn: 0.7534762        total: 7.72s    remaining: 2m
600:    learn: 0.7098401        total: 15s      remaining: 1m 49s
900:    learn: 0.6770550        total: 22.6s    remaining: 1m 43s
1200:   learn: 0.6494836        total: 29.8s    remaining: 1m 34s
1500:   learn: 0.6267634        total: 36.9s    remaining: 1m 26s
1800:   learn: 0.6047805        total: 44.2s    remaining: 1m 18s
2100:   learn: 0.5867632        total: 52.5s    remaining: 1m 12s
2400:   learn: 0.5700053        total: 60s      remaining: 1m 4s
2700:   learn: 0.5545074        total: 1m 7s    remaining: 57.7s
3000:   learn: 0.5407359        total: 1m 15s   remaining: 50s
3300:   learn: 0.5272693        total: 1m 22s   remaining: 42.3s
3600:   learn: 0.5147162        total: 1m 29s   remaining: 34.7s
3900:   learn: 0.5031963        total: 1m 36s   remaining: 27.3s
4200:   learn: 0.4914841        total: 1m 43s   remaining: 19.8s
4500:   learn: 0.4807274        total: 1m 50s   remaining: 12.3s
4800:   learn: 0.4709178        total: 1m 58s   remaining: 4.89s
4999:   learn: 0.4646923        total: 2m 2s    remaining: 0us
```

CatBoost

0.73823

```
submi_cat = pd.DataFrame(result)
submi_cat['index'] = np.arange(26457, 36457)
submi_cat = submi_cat[['index',0,1,2]]
submi_cat
```

	index	0	1	2
0	26457	0.017754	0.027215	0.955031
1	26458	0.215099	0.133738	0.651162
2	26459	0.043745	0.132273	0.823982
3	26460	0.076875	0.078490	0.844635
4	26461	0.087195	0.270207	0.642598
...	...	...	...	...
9995	36452	0.113252	0.186596	0.700152
9996	36453	0.203086	0.256848	0.540066
9997	36454	0.007769	0.051064	0.941167
9998	36455	0.093728	0.303976	0.602296
9999	36456	0.032848	0.240059	0.727094

10000 rows × 4 columns

03 모델링 결과

(3) XGBoost

```
from xgboost import XGBClassifier

xgb_model = XGBClassifier(n_estimators=400, objective = 'multi:softprob', learning_rate=0.1, max_dept
np.random.seed(2020)
xgb_model.fit(train_x, train_y)
result11 = xgb_model.predict_proba(test_x)
```

```
submi_xgb = pd.DataFrame(result11)
submi_xgb['index'] = np.arange(26457, 36457)
submi_xgb = submi_xgb[['index',0,1,2]]
submi_xgb
```

	index	0	1	2
0	26457	0.076772	0.200647	0.722580
1	26458	0.126175	0.167108	0.706716
2	26459	0.108816	0.184138	0.707046
3	26460	0.111668	0.146806	0.741525
4	26461	0.084213	0.157079	0.758708
...	...	...	...	...
9995	36452	0.104150	0.230961	0.664890
9996	36453	0.103007	0.247734	0.649259
9997	36454	0.034301	0.091433	0.874266
9998	36455	0.071080	0.171955	0.756965
9999	36456	0.102270	0.177523	0.720206

10000 rows × 4 columns

XGBoost

0.78253

03

모델링 결과

LightGBM

0.73212

<

CatBoost

0.73823

<

XGBoost

0.78253

03

모델링 결과

LightGBM

0.73212

Log loss 값이 가장 낮은
LightGBM 최종 선택

CREDIT PREDICTION

04

분석 한계

❶ 분석의 한계 & 개선점

04 분석의 한계 & 개선점

01

자녀수, 결혼여부/가족수 관계 이상치 샘플 처리

적절한 전처리 방향을 잡지 못함에 대한 아쉬움.

02

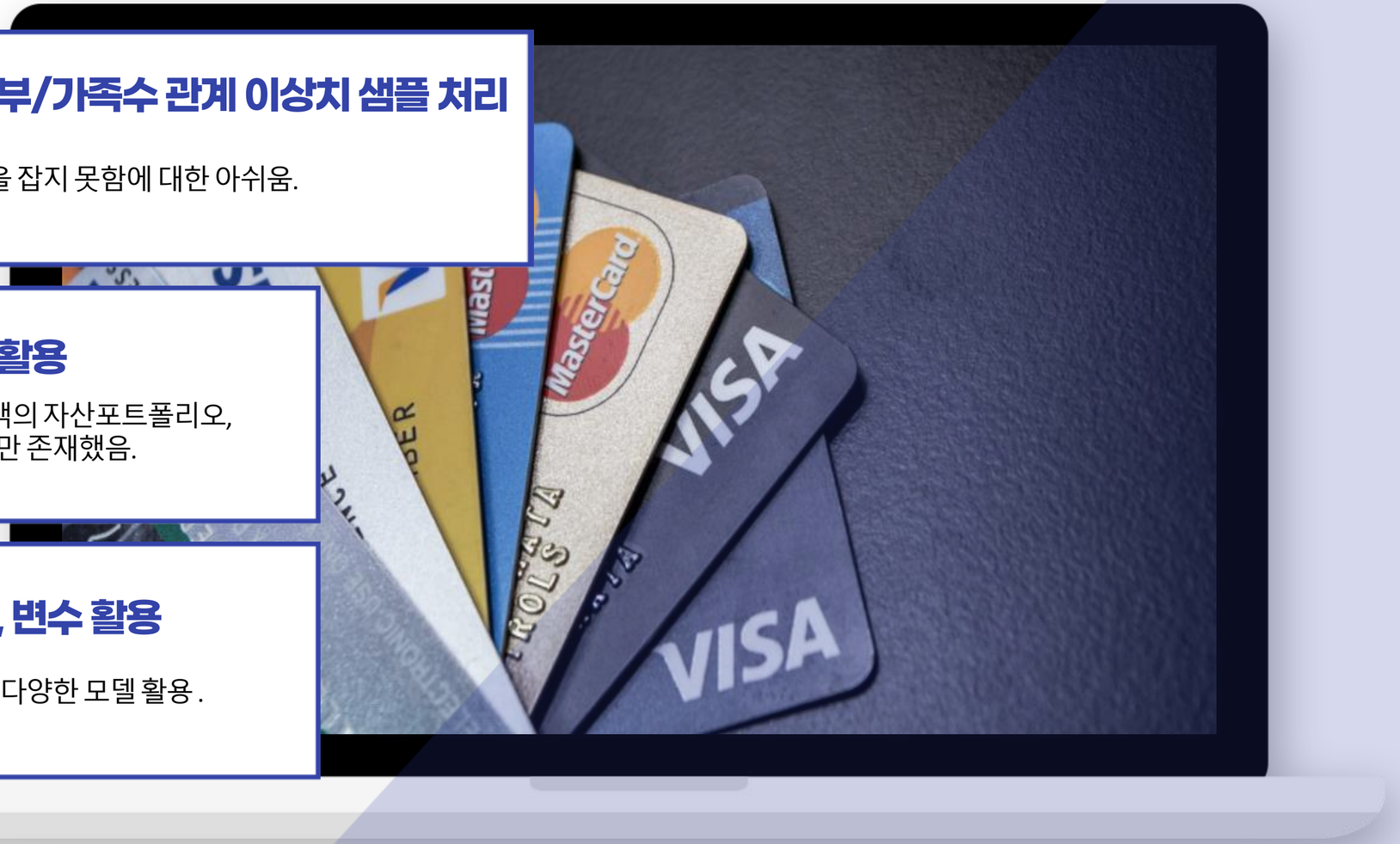
비금융 데이터 활용

활용한 데이터는 고객의 자산포트폴리오,
Demo와 관련된 컬럼만 존재했음.

03

더 다양한 모델, 변수 활용

추가 변수 활용 및 더 다양한 모델 활용 .



CREDIT PREDICTION

감사합니다