2025-2 데이터문제해결및실습1

11주차-1
[경진대회3] 안전 운전자 예측_1

세종대학교
인공지능데이터사이언스학과
박동현 교수

- 본 강의는 골든래빗 출판사의 머신러닝/딥러닝 문제해결전략이 제공하는 강의 교안에 기반함.

Chapter

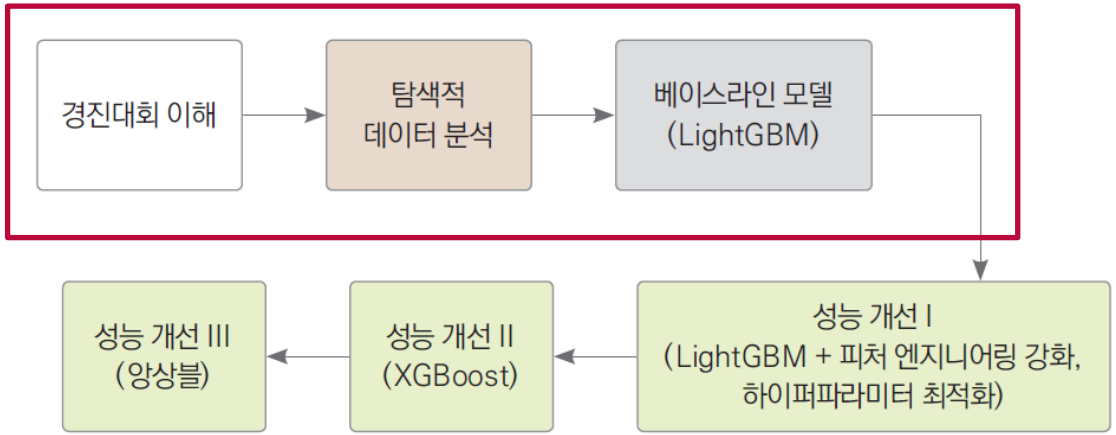
08

[경진대회]
안전 운전자 예측_1

□ 학습 목표

실제 기업 데이터를 활용한 안전 운전자 예측 경진대회에 참가한다. 먼저 탐색적 데이터 분석으로 모델링에 필요 없는 데이터를 찾아본다. 이번에도 베이스라인 모델에서 시작해 성능이 좋은 모델로 발전시킨다. 이 과정에서 캐글에서 실제로 많이 활용되는 여러 가지 고급 모델링 기법을 배울 수
있다.

□ 학습 순서



□ 학습 키워드

- **유형 및 평가지표** : 이진분류, 정규화된 지니계수
- **탐색적 데이터 분석** : 피처 요약표 응용, 결측값 시각화, 결측값 처리
- **머신러닝 모델** : OOF 예측, LightGBM, XGBoost, 앙상블
- **피처 엔지니어링** : 창의적 피처 엔지니어링
- **하이퍼파라미터 최적화** : 베이지안 최적화

□ 핵심 요약

- 머신러닝 평가지표로서의 **정규화 지니계수**는 '예측값에 대한 지니계수 / 예측이 완벽할 때의 지니계수'이다.
- **피처 요약표** : 데이터 성격에 맞게 조금씩 변형해 사용
- **missingno 패키지** : 결측값을 시각화해주는 유용한 함수를 제공함.'
- **세 가지 결측값 처리 방법** : 결측값이 많다면 해당 피처 자체를 제거하고, 많지 않다면 다른 값으로 대체한다. 때로는 결측값이 예측에 도움되는 경우도 있다. 이럴 때는 결측값을 하나의 고윳값으로 간주한다.
- **베이지안 최적화** : 그리드서치보다 최적 하이퍼파라미터를 더 빠르고 효율적으로 찾아준다. 코드가 직관적이라 사용하기 편하다.
- **OOF 예측** : K 폴드 교차 검증을 수행하면서 각 폴드마다 테스트 데이터로 예측하는 방식. 과대적합 방지 및 앙상블 효과가 있다.

□ 경진대회 정보

- **난이도** : ★★☆
- **경진대회명** : 포르투 세구로 안전 운전자 예측 경진대회
- **미션** : 포르투 세구로 보험사에서 제공한 고객 데이터를 활용해 운전자가 보험을 청구할 확률 예측
- **문제 유형** : 이진분류
- **평가지표** : 정규화된 지니계수
- **데이터 크기** : 288.7MB
- **참가팀 수** : 5,156팀
- **제출 시 사용한 모델** : LightGBM와 XGBoost의 앙상블
- **파이썬 버전** : 3.7.10
- **사용 라이브러리 및 버전**
    - numpy (numpy==1.19.5) / pandas (pandas==1.3.2)
    - seaborn (seaborn==0.11.2) / matplotlib (matplotlib==3.4.3)
    - sklearn (scikit-learn==0.23.2) / scipy (scipy==1.7.1)
    - missingno (missingno==0.4.2) / lightgbm (lightgbm==3.2.1)
    - xgboost (xgboost==1.4.2) / bayes_opt (bayesian-optimization==1.2.0)

8.1 경진대회 이해

이번에는 안전 운전자 예측 경진대회(Porto Seguro’s Safe Driver Prediction)에 참가한다. 본 경진대회는 2017년 9월 30일부터 2017년 11월 30일까지 두 달 동안 개최되었으며, 총 5,156팀이 참가했다.

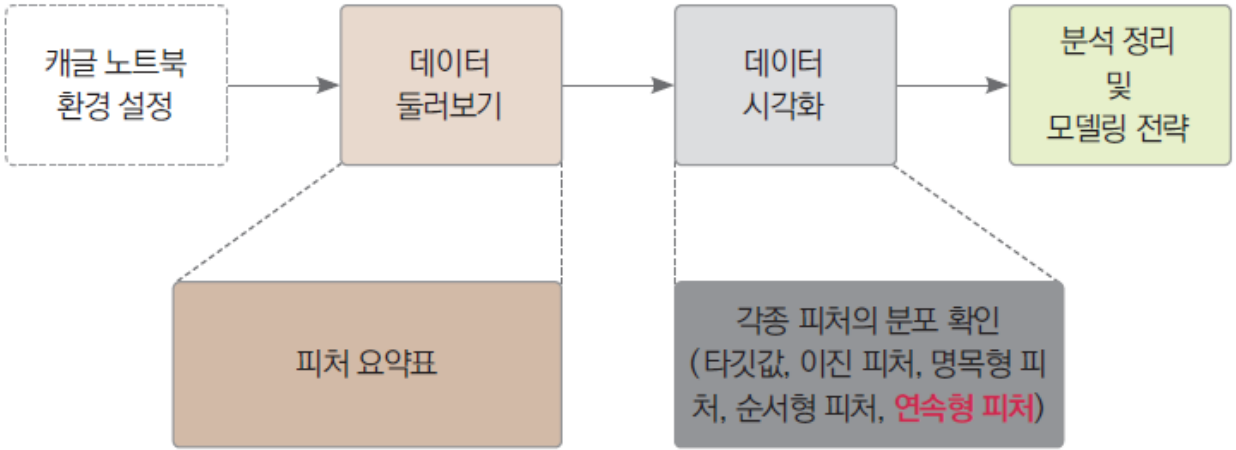
안전 운전자 예측 경진대회는 포르투 세구로라는 브라질 보험 회사에서 주최한 대회다. 운전자가 보험금을 청구할 확률을 정확히 예측하는 모델을 만드는 게 본 경진대회 목표이다.

주어진 데이터는 포르투 세구로가 보유한 고객 데이터다. 주어진 데이터에 결측값이 꽤 많은데, 결측값은 -1로 기록돼 있다.

마지막으로 타깃값은 0 또는 1이다. 값이 0이면 운전자가 보험금을 청구하지 않는다는 뜻이고, 1이면 청구한다는 뜻이다. 타깃값이 두 개이므로 본 경진대회는 이진분류 문제에 속한다.

8.2 탐색적 데이터 분석

캐글 검색창에서 “Porto Seguro’s Safe Driver Prediction”을 검색하여 경진대회에 접속한다.



탐색적 데이터 분석에 사용한 코드는 본 경진대회에서 추천수가 네 번째로 많은 코드를 참고해 작성했다.

- https://www.kaggle.com/bertcarremans/data-preparation-exploration

8.2 탐색적 데이터 분석

8.2.1 데이터 둘러보기

- 훈련 데이터 약 59만 개, 테스트 데이터 약 89만 개(훈련 데이터보다 테스트 데이터가 더 많다)
- 타깃값을 제외한 피처 : 총 57개
- 일괄로 입력된 타깃값 확률 : 0.0364
- 피처명은 비식별화되어 있어서 각 피처가 어떤 의미인지 알 수 없음
- 분류: ind, reg, car, calc
- 데이터 종류: bin(이진 피처), cat (명목형 피처)
- 결측값 시각화를 위해 missingno 패키지 사용

8.2 탐색적 데이터 분석

8.2.1 데이터 둘러보기

피처 요약표

	데이터 타입	결측값 개수	고윳값 개수	데이터 종류
target	int64	0	2	이진형
ps_ind_01	int64	0	8	순서형
ps_ind_02_cat	int64	216	5	명목형
ps_ind_03	int64	0	12	순서형
ps_ind_04_cat	int64	83	3	명목형
ps_ind_05_cat	int64	5809	8	명목형
ps_ind_06_bin	int64	0	2	이진형
ps_ind_07_bin	int64	0	2	이진형
ps_ind_08_bin	int64	0	2	이진형
ps_ind_09_bin	int64	0	2	이진형
ps_ind_10_bin	int64	0	2	이진형
ps_ind_11_bin	int64	0	2	이진형
ps_ind_12_bin	int64	0	2	이진형
ps_ind_13_bin	int64	0	2	이진형
ps_ind_14	int64	0	5	순서형
ps_ind_15	int64	0	14	순서형
ps_ind_16_bin	int64	0	2	이진형
ps_ind_17_bin	int64	0	2	이진형
ps_ind_18_bin	int64	0	2	이진형
ps_reg_01	float64	0	10	연속형

8.2 탐색적 데이터 분석

8.2.2 데이터 시각화

- 데이터 시각화를 통해 모델링에 필요한 피처는 무엇이고 필요 없는 피처는 무엇인지 선별한다.
- 먼저 타깃값 분포를 활용해 타깃값이 얼마나 불균형한지 알아본다.
- 이진 피처, 명목형 피처, 순서형 피처의 고윳값별 타깃값 비율을 알아본다.

시각화 라이브러리를 불러온다.

```
import seaborn as sns
import matplotlib as mpl
import matplotlib.pyplot as plt
%matplotlib inline
```

8.2 탐색적 데이터 분석

8.2.2 데이터 시각화

타깃값 분포

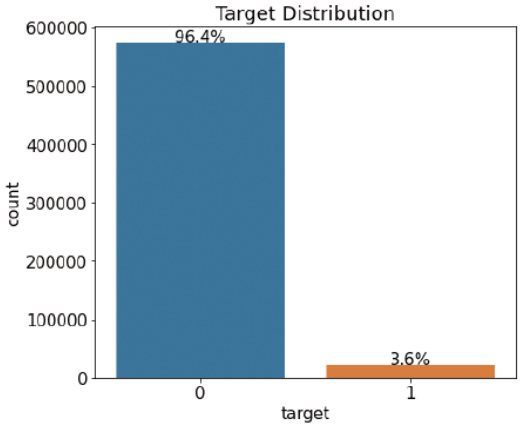
```
def write_percent(ax, total_size):
    '''도형 객체를 순회하며 막대 그래프 상단에 타깃값 비율 표시'''
    for patch in ax.patches:
        height = patch.get_height() # 도형 높이(데이터 개수)
        width = patch.get_width() # 도형 너비
        left_coord = patch.get_x() # 도형 왼쪽 테두리의 x축 위치
        percent = height/total_size*100 # 타깃값 비율

        # (x, y) 좌표에 텍스트 입력
        ax.text(left_coord + width/2.0, # x축 위치
                height + total_size*0.001, # y축 위치
                '{:1.1f}%'.format(percent), # 입력 텍스트
                ha='center') # 가운데 정렬

mpl.rc('font', size=15)
plt.figure(figsize=(7, 6))
ax = sns.countplot(x='target', data=train)
write_percent(ax, len(train)) # 비율 표시
ax.set_title('Target Distribution');
```

8.2 탐색적 데이터 분석

8.2.2 데이터 시각화

타깃값 분포

: 전체 운전자 중 3.6%만 보험금을 청구. 소수의 운전자만 보험금을 청구했으며, 타깃값이 불균형하다.

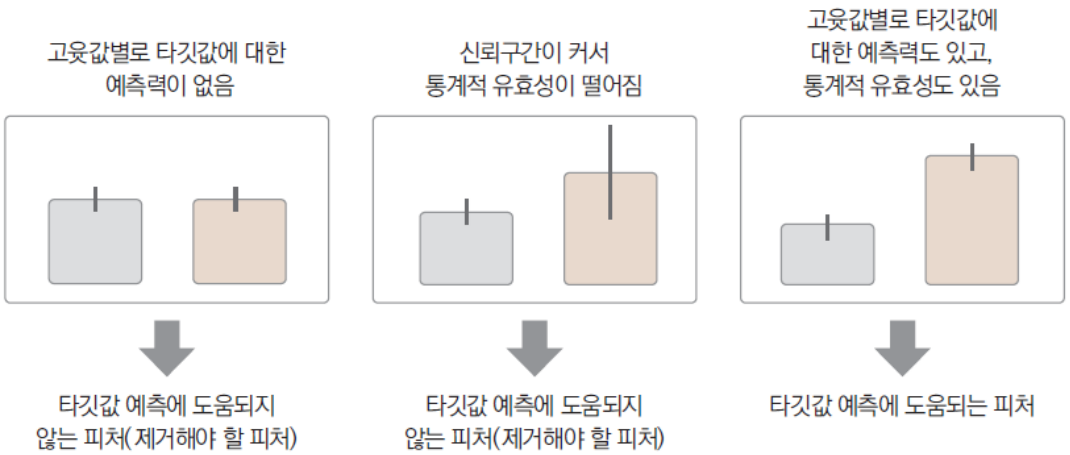


8.2 탐색적 데이터 분석

8.2.2 데이터 시각화

타깃값 분포

: 타깃값이 불균형하므로 비율이 작은 타깃값 1을 잘 예측하는 게 중요하다. 각 피처의 분포를 알아보기보다는 각 피처의 고윳값별 타깃값 1 비율을 알아본다. 고윳값별 타깃값 1 비율을 통해 해당 피처가 모델링에 필요한 피처인지 확인할 수 있다.



8.2 탐색적 데이터 분석

8.2.2 데이터 시각화

이진 피처

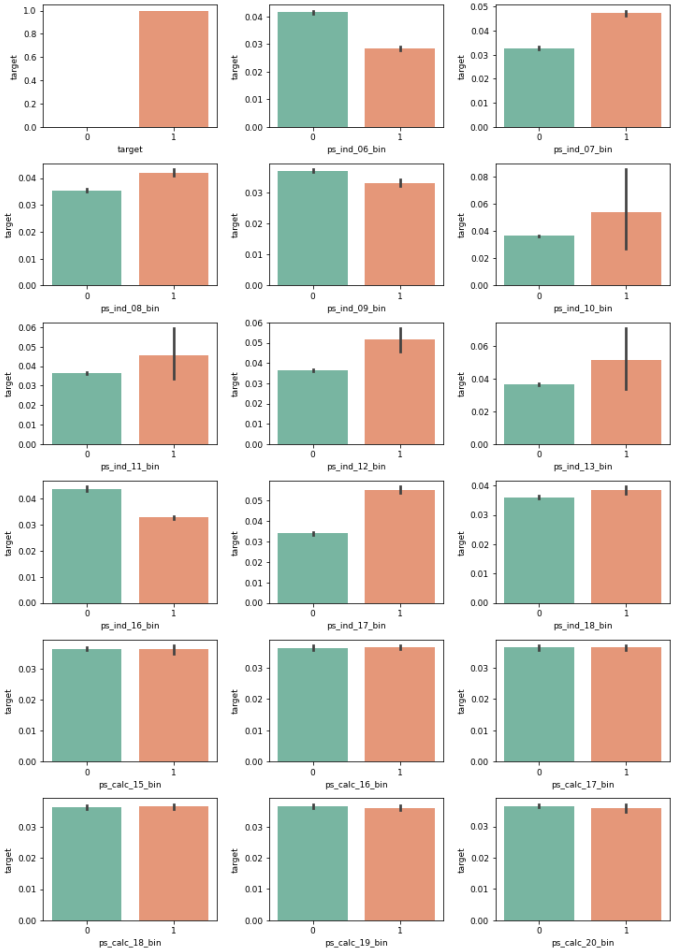
```
import matplotlib.gridspec as gridspec

def plot_target_ratio_by_features(df, features, num_rows, num_cols,
                                  size=(12, 18)):
mpl.rc('font', size=9)
plt.figure(figsize=size) # 전체 Figure 크기 설정
grid = gridspec.GridSpec(num_rows, num_cols) # 서브플롯 배치
plt.subplots_adjust(wspace=0.3, hspace=0.3) # 서브플롯 좌우/상하 여백 설정

for idx, feature in enumerate(features):
    ax = plt.subplot(grid[idx])
    # ax축에 고윳값별 타깃값 1 비율을 막대 그래프로 그리기
    sns.barplot(x=feature, y='target', data=df, palette='Set2', ax=ax)
```

```
ibin_features = summary[summary['데이터 종류'] == '이진형'].index # 이진 피처
# 이진 피처 고윳값별 타깃값 1 비율을 막대 그래프로 그리기
plot_target_ratio_by_features(train, bin_features, 6, 3) # 6행 3열 배치
```

8.2 탐색적 데이터 분석

8.2.2 데이터 시각화

이진 피처



8.2 탐색적 데이터 분석

8.2.2 데이터 시각화

이진 피처

서브플롯 위치	피처명	제거해야 하는 이유
❻~❾ (1행 2열~2행 2열)	ps_ind_10_bin ~ ps_ind_13_bin	신뢰구간이 넓어 통계적 유효성이 떨어짐
⓭~⓲ (4행 0열~5행 2열)	ps_calc_15_bin ~ ps_calc_20_bin	고윳값별 타깃값 비율 차이가 없어 타깃값 예측력이 없음

▲ (분석 결과) 제거해야 할 이진 피처

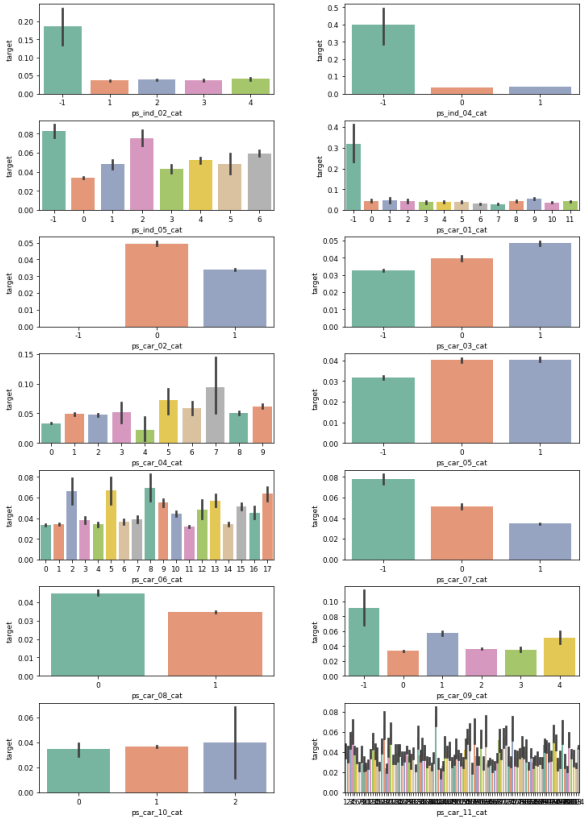
8.2 탐색적 데이터 분석

8.2.2 데이터 시각화

명목형 피처

```
inom_features = summary[summary['데이터 종류'] == '명목형'].index
# 명목형 피처
plot_target_ratio_by_features(train, nom_features,7, 2) # 7행 2열
```

- 결측값 자체에 타깃값 예측력이 있다면 고윳값으로 간주(결측값 처리 필요 없음)
- 명목형 피처 중 제거해야 할 피처 없음, 모두 모델링에 이용



8.2 탐색적 데이터 분석

8.2.2 데이터 시각화

순서형 피처

```
ord_features = summary[summary['데이터 종류'] == '순서형'].index # 순서형 피처

plot_target_ratio_by_features(train, ord_features, 8, 2, (12, 20)) # 8행 2열
```

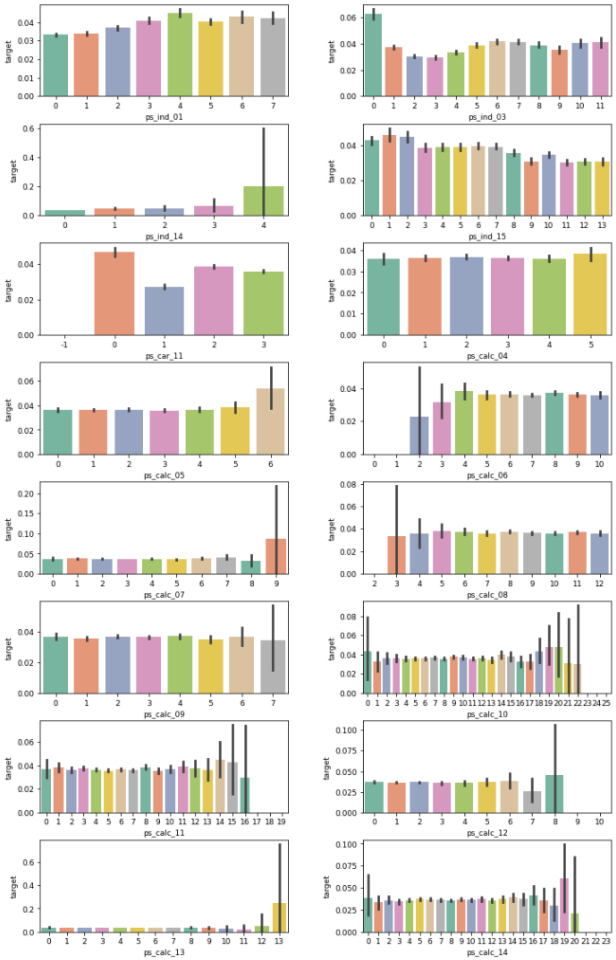
서브플롯 위치	피처명	제거해야 하는 이유
❶	ps_ind_14	타깃값 비율의 신뢰구간이 넓어 통계적 유효성이 떨어짐
❷	ps_calc_04 ~ ps_calc_14	고윳값별 타깃값 비율 차이가 없음. 타깃값 비율이 다르더라도 신뢰구간이 넓어 통계적 유효성이 떨어짐

▲ (분석 결과) 제거해야 할 순서형 피처

8.2 탐색적 데이터 분석

8.2.2 데이터 시각화

순서형 피처



8.2 탐색적 데이터 분석

8.2.2 데이터 시각화

연속형 피처

- 연속된 값이므로 고윳값이 굉장히 많다. 고윳값이 많을 때는 몇 개의 구간으로 나눠 구간별 비율을 알아본다.
- 판다스의 cut( ) 함수를 활용

서브플롯 위치	피처명	제거해야 하는 이유
⑧~⑩	ps_calc_01~ps_calc_03	구간별 타깃값 비율 차이가 없음

▲ (분석 결과) 제거해야 할 연속형 피처

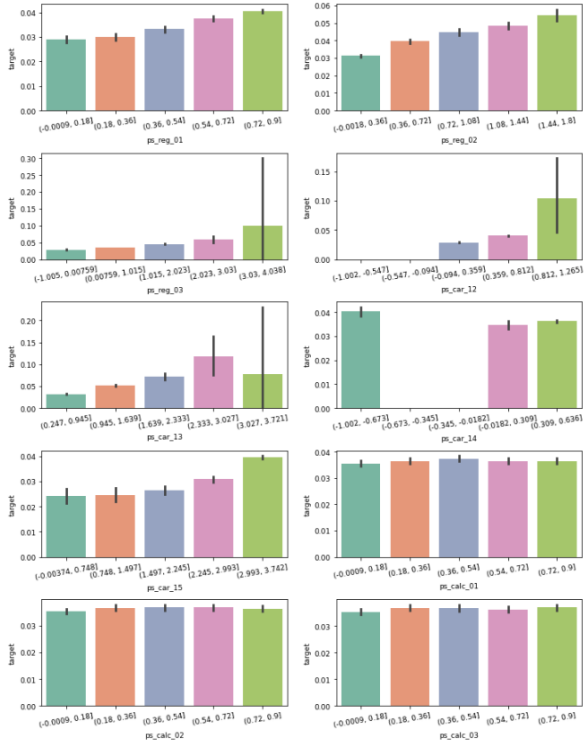
8.2 탐색적 데이터 분석

8.2.2 데이터 시각화

연속형 피처

서브플롯 위치	피처명	제거해야 하는 이유
⑧~⑩	ps_calc_01~ps_calc_03	구간별 타깃값 비율 차이가 없음

▲ (분석 결과) 제거해야 할 연속형 피처

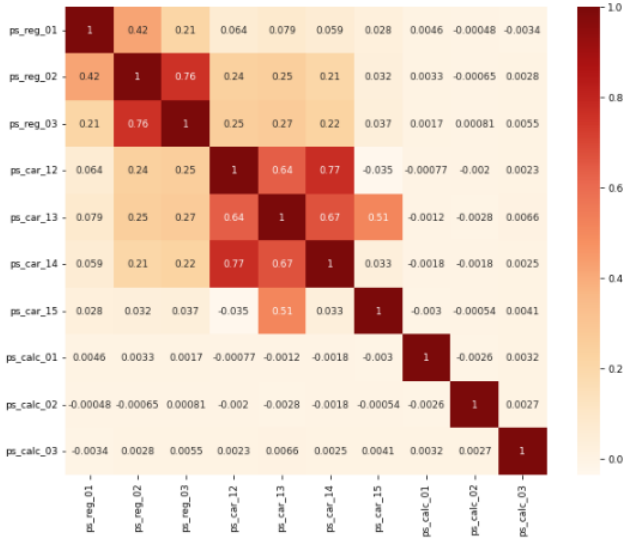


8.2 탐색적 데이터 분석

8.2.2 데이터 시각화

연속형 피처 2

- 일반적으로 강한 상관관계를 보이는 두 피처가 있으면 둘 중 하나를 제거하는 게 좋다. 상관관계가 강하면 타깃값 예측력도 비슷하다. 그런 피처가 있으면 모델 성능이 떨어질 수도 있다.
- 상관관계가 강한 피처 제거(ps_car_14)



분석 정리 및 모델링 전략

분석 정리

1. 데이터가 크고 피처 수도 많다.
2. 피처명만으로 분류별 혹은 데이터 종류별 피처들을 구분해 추출해낼 수 있다.
3. **결측값 처리** : 결측값 자체에 타깃값 예측력이 있다면 고윳값으로 간주한다.
4. **결측값 처리** : 피처 간 상관관계 분석은 결측값 제거 후 수행한다.
5. **피처 제거** : 신뢰구간이 넓으면 통계적 유효성이 떨어져 믿을 수 없다(ps_ind_14, ps_calc_04~ps_calc_14).
6. **피처 제거** : 고윳값별 타깃값 비율에 차이가 없다면 타깃값 예측력이 없다(ps_calc_04~ps_calc_14).
7. **피처 제거** : (연속형 데이터의 경우) 구간별 타깃값 차이가 거의 없다면 타깃값 예측력이 없다 (ps_calc_01~ps_calc_03).
8. **피처 제거** : 일반적으로 강한 상관관계를 보이는 두 피처가 있으면 둘 중 하나를 제거하는 게 좋다 (ps_car_14).

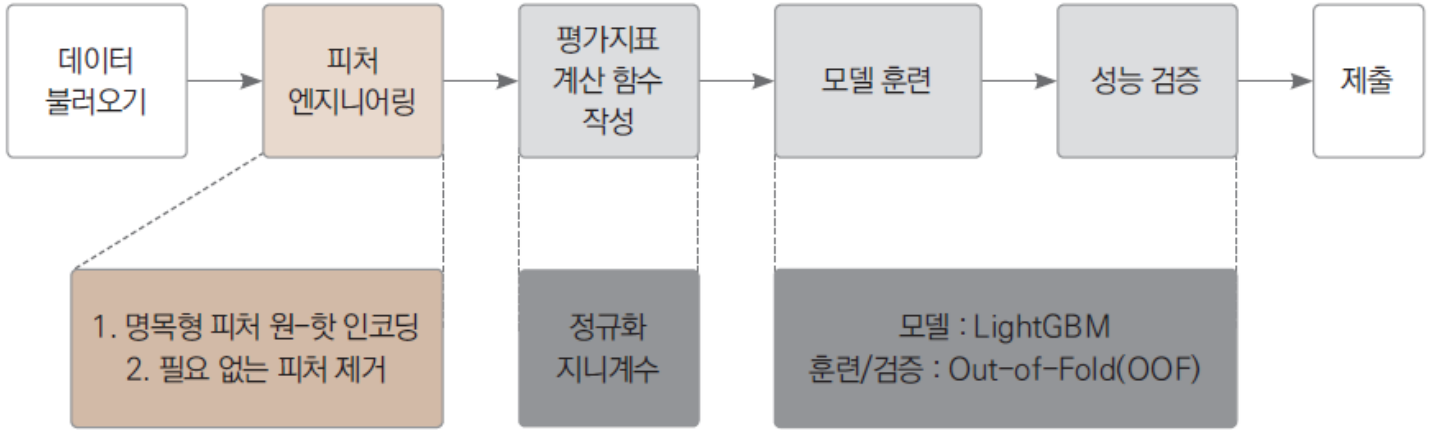
분석 정리 및 모델링 전략

모델링 전략

- **베이스라인 모델** : LightGBM
  - **훈련 및 예측** : OOF 예측(과대적합 방지 + 앙상블 효과)
- **성능 개선 I** : LightGBM 유지
  - 피처 엔지니어링 : 파생 피처 추가
  - 하이퍼파라미터 최적화 : 베이지안 최적화
- **성능 개선 II** : XGBoost (모델만 변경)
- **성능 개선 III** : LightGBM + XGBoost 앙상블

8.3 베이스라인 모델

앞서 선별한 피처들을 제거해 베이스라인 모델을 만든다. 베이스라인 모델로는 LightGBM을 사용한다. LightGBM은 마이크로소프트가 개발한 모델로, 빠르면서 성능이 좋다.



8.3 베이스라인 모델

8.3.1 피처 엔지니어링

데이터 합치기

- 먼저 훈련 데이터와 테스트 데이터를 합친다. 두 데이터에 동일한 인코딩을 적용하기 위해서다. 인코딩은 타깃값이 아닌 피처에만 적용해야 하므로 합친 데이터에서 타깃값은 제거한다.

```
all_data = pd.concat([train, test], ignore_index=True)
all_data = all_data.drop('target', axis=1) # 타깃값 제거
```

- all_data에 있는 모든 피처를 all_features 변수에 저장한다.

```
all_features = all_data.columns # 전체 피처
all_features
```

8.3 베이스라인 모델

8.3.1 피처 엔지니어링

명목형 피처 원-핫 인코딩

- 지능형 리스트를 활용해 명목형 피처를 추출한 다음 인코딩한다.

```
from sklearn.preprocessing import OneHotEncoder
# 명목형 피처 추출
cat_features = [feature for feature in all_features if 'cat' in feature]

onehot_encoder = OneHotEncoder() # 원-핫 인코더 객체 생성
# 인코딩
encoded_cat_matrix = onehot_encoder.fit_transform(all_data[cat_features])

encoded_cat_matrix
```

8.3 베이스라인 모델

8.3.1 피처 엔지니어링

필요 없는 피치 제거

- 원-핫 인코딩에 사용한 명목형 피처, calc 분류의 피처, 추가 제거할 6개 피처를 제외한 나머지 피처를 remaining_features에 저장한다.

```
# 추가로 제거할 피처
drop_features = ['ps_ind_14', 'ps_ind_10_bin', 'ps_ind_11_bin',
                 'ps_ind_12_bin', 'ps_ind_13_bin', 'ps_car_14']

# '1) 명목형 피처, 2) calc 분류의 피처, 3) 추가 제거할 피처'를 제외한 피처
remaining_features = [feature for feature in all_features
                      if ('cat' not in feature and
                      'calc' not in feature and
                      feature not in drop_features)]
```

8.3 베이스라인 모델

8.3.1 피처 엔지니어링

데이터 나누기

- 전체 데이터를 훈련 데이터와 테스트 데이터로 다시 나눈다. 타깃값도 y에 할당한다.

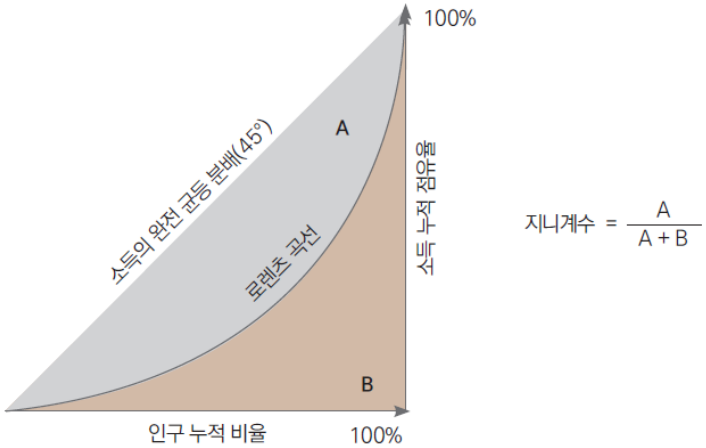
```
num_train = len(train) # 훈련 데이터 개수

# 훈련 데이터와 테스트 데이터 나누기
X = all_data_sprs[:num_train]
X_test = all_data_sprs[num_train:]

y = train['target'].values
```

8.3 베이스라인 모델

8.3.2 평가지표 계산 함수 작성

지니계수란?

- 경제학에서 지니계수는 소득 불평등 정도를 나타내는 지표. 지니계수가 작을수록 소득 수준이 평등하고, 클수록 불평등하다.
- 머신러닝에서 지니계수는 모델의 예측 성능을 측정하는 데 쓰인다.



$$\text{지니계수} = \frac{A}{A+B}$$

8.3 베이스라인 모델

8.3.2 평가지표 계산 함수 작성

정규화 지니계수 계산 함수

- 정규화란 값의 범위를 0~1 사이로 조정한다는 뜻이므로, 정규화 지니계수는 값이 0에 가까울수록 성능이 나쁘고, 1에 가까울수록 성능이 좋다.

$$\text{정규화 지니계수} = \frac{\text{예측 값에 대한 지니계수}}{\text{예측이 완벽할 때의 지니계수}}$$

- '예측값에 대한 지니계수: 예측값과 실젯값으로 구한 지니계수
- 예측이 완벽할 때의 지니계수: 실젯값과 실젯값으로 구한 지니계수

8.3 베이스라인 모델

8.3.3 모델 훈련 및 성능 검증

OOF (Out-Of-Fold) 예측 방식

- K 폴드 교차 검증을 수행하면서 각 폴드마다 테스트 데이터로 예측하는 방식
- OFF 예측 절차

1. 전체 훈련 데이터를 K개 그룹나눈다.
2. K개 그룹 중 한 그룹은 검증 데이터, 나머지 K-1개 그룹은 훈련 데이터로 지정한다.
3. 훈련 데이터로 모델을 훈련한다.
4. 훈련된 모델을 이용해 검증 데이터로 타깃 확률을 예측하고, 전체 테스트 데이터로도 타깃 확률을 예측한다.
5. 검증 데이터로 구한 예측 확률과 테스트 데이터로 구한 예측 확률을 기록한다.
6. 검증 데이터를 다른 그룹으로 바꿔가며 2~5번 절차를 총 K번 반복한다.
7. K개 그룹의 검증 데이터로 예측한 확률을 훈련 데이터 실제 타깃값과 비교해 성능 평가점수를 계산한다. 이 점수로 모델 성능을 가늠해볼 수 있습니다.
8. 테스트 데이터로 구한 K개 예측 확률의 평균을 구한다. 이 값이 최종 예측 확률이며, 제출해야 하는 값이다

8.3 베이스라인 모델

8.3.3 모델 훈련 및 성능 검증

OOF 예측 방식

- OOF 예측 방식의 장점
  - 과대적합 방지에 효과적이다.
  - 앙상블 효과가 있어 모델 성능이 좋아진다.

$$\frac{\left[\text{테스트 데이터 예측 확률 1} + \text{테스트 데이터 예측 확률 2} + \text{테스트 데이터 예측 확률 3} + \text{테스트 데이터 예측 확률 4} + \text{테스트 데이터 예측 확률 5}\right]}{5} = \text{최종 예측 확률}$$

8.3 베이스라인 모델

8.3.3 모델 훈련 및 성능 검증

OOF 방식으로 LightGMB 훈련

- 층화 K 폴드 교차 검증기 생성하기

```
from sklearn.model_selection import StratifiedKFold

# 층화 K 폴드 교차 검증기
folds = StratifiedKFold(n_splits=5, shuffle=True, random_state=1991)
```

- LightGBM의 하이퍼파라미터를 설정하기

```
params = {'objective': 'binary',
'learning_rate': 0.01,
'force_row_wise': True,
'random_state': 0}
```

8.3 베이스라인 모델

8.3.3 모델 훈련 및 성능 검증

OOF 방식으로 LightGMB 훈련

- 1차원 배열 두 개 만들기

```
f# OOF 방식으로 훈련된 모델로 검증 데이터 타깃값을 예측한 확률을 담을 1차원 배열
oof_val_preds = np.zeros(X.shape[0])
# OOF 방식으로 훈련된 모델로 테스트 데이터 타깃값을 예측한 확률을 담을 1차원 배열
oof_test_preds = np.zeros(X_test.shape[0])
```

8.3 베이스라인 모델

8.3.3 모델 훈련 및 성능 검증

OOF 방식으로 LightGMB 훈련

- LightGBM 모델을 훈련하기

```
import lightgbm as lgb


# OOF 방식으로 모델 훈련, 검증, 예측
for idx, (train_idx, valid_idx) in enumerate(folds.split(X, y)): #1
    # 각 폴드를 구분하는 문구 출력
    print('#'*40, f'폴드 {idx+1} / 폴드 {folds.n_splits}', '#'*40)

    # 훈련용 데이터, 검증용 데이터 설정 #2
    X_train, y_train = X[train_idx], y[train_idx] # 훈련용 데이터
    X_valid, y_valid = X[valid_idx], y[valid_idx] # 검증용 데이터

    # LightGBM 전용 데이터셋 생성 #3
    dtrain = lgb.Dataset(X_train, y_train) # LightGBM 전용 훈련 데이터셋
    dvalid = lgb.Dataset(X_valid, y_valid) # LightGBM 전용 검증 데이터셋
```

8.3 베이스라인 모델

8.3.3 모델 훈련 및 성능 검증

OOF 방식으로 LightGMB 훈련

- LightGBM 모델을 훈련하기

```
# LightGBM 모델 훈련 #4
lgb_model = lgb.train(params=params, # 훈련용 하이퍼파라미터
                      train_set=dtrain, # 훈련 데이터셋
                      num_boost_round=1000, # 부스팅 반복 횟수
                      valid_sets=dvalid, # 성능 평가용 검증 데이터셋
                      feval=gini, # 검증용 평가지표
                      early_stopping_rounds=100, # 조기종료 조건
                      verbose_eval=100) # 100번째마다 점수 출력


# 테스트 데이터를 활용해 OOF 예측 #5
oof_test_preds += lgb_model.predict(X_test)/folds.n_splits

# 모델 성능 평가를 위한 검증 데이터 타깃값 예측 #6
oof_val_preds[valid_idx] += lgb_model.predict(X_valid)

# 검증 데이터 예측 확률에 대한 정규화 지니계수 #7
gini_score = eval_gini(y_valid, oof_val_preds[valid_idx])
print(f'폴드 {idx+1} 지니계수: {gini_score}\n')
```

8.3 베이스라인 모델

8.3.3 모델 훈련 및 성능 검증

OOF 방식으로 LightGMB 훈련

- 모델을 훈련할 때 출력된 로그를 살펴보기

```
######################################## 폴드 5 / 폴드 5 ########################
################
[LightGBM] [Info] Number of positive: 17355, number of negative: 458815
[LightGBM] [Info] Total Bins 1098
[LightGBM] [Info] Number of data points in the train set: 476170, number of used
features: 200
[LightGBM] [Info] [binary:BoostFromScore]: pavg=0.036447 -> initscore=-3.274766
[LightGBM] [Info] Start traini                                    4766
Training until validation scores don't improve for 100 rounds
[100] valid_0's binary_logloss: 0.153483      valid_0's gini: 0.262106
[200] valid_0's binary_logloss: 0.152646      valid_0's gini: 0.273406
[300] valid_0's binary_logloss: 0.152291      valid_0's gini: 0.279805
[400] valid_0's binary_logloss: 0.152093      valid_0's gini: 0.284645
[5    0's binary_logloss: 0.152004      valid_0's gini: 0.28713
[600] valid_0's binary_logloss: 0.151982      valid_0's gini: 0.287668
Early stopping, best iteration is:
[583] valid_0's binary_logloss: 0.15198       valid_0's gini: 0.287804
폴드 5 지니계수 : 0.2878042213842625
```

❶ 성능 평가점수가 두 개

❷ 조기 종료

❸ 종료 시까지 최고 점수를 기록한 이터레이션과 점수

8.3 베이스라인 모델

8.3.4 예측 및 결과 제출

- 최종 예측 확률은 oof_test_preds에 담겨 있다. OOF 예측 방식으로 총 5개 폴드로 교차 검증한 확률값들의 평균이다.

```
submission['target'] = oof_test_preds
submission.to_csv('submission.csv')
```

- 커밋 후 제출한다.