2023 D&A ML competition

서울과 부산에 존재하는 아파트의 실제 거래가격 예측

머신러닝이목1래
김윤재 백경린 손아현 홍예진

CONTENTS

01
분석 및 모델링 전략

02
Feature Engineering & Cleansing

03
Encoding & Scaling

04
Modeling

01 분석 및 모델링 전략

아파트, 어린이집, 공원
+ 외부 데이터

>>>

서울 · 부산 아파트의
실제 거래 가격 예측

파이프라인 적용

단일모델

- Light GBM
- HistGradientBoostingRegressor

앙상블

- LightGBM model & HistGradientBoostingRegressor
- Catboost & XGboost
- Catboost & XGboost & HistGradientBoostingRegressor

=> 최종적으로 성능이 가장 잘 나온 모델 선택

02 Feature Engineering & Cleansing

> train 데이터

수치형 변수

transaction_id,
apartment_id,
exclusive_use_area,
year_of_completion,
transaction_year_month,
floor,
transaction_real_price

범주형 변수

city, dong, jibun,
apt, addr_kr,
transaction_date

02 Feature Engineering & Cleansing

> train 데이터

city & dong	‘gu’
apt	‘top10’ ‘transformed’ ‘apt_counts’
floor	‘floor_avg’
transaction_year_month	‘year’ ‘month’ ‘season’ ‘주상복합’
year_of_completion	‘apartment_age’
apartment_id	‘previous_transaction’

transformed: ‘top10 시공사'와 '대표 25개'에 해당 여부
apt_counts: 동일 아파트 수

02 Feature Engineering & Cleansing

> day_care_center 데이터

is_commuting_vehicle → 'vehicle_pop'

구별 차량 여부 비율

day_care_baby_num & day_care_type → 'baby'

어린이집 유형별 아이 비율

> park 데이터

dong & park_type → 'count'

동 별 공원 수

park_type → 'size'

공원 유형별 공원 수

count / size → 'population'

전체에서 동 별 공원 별 차지하는 비율

결측치는 주변 구별 평균으로 대체

02 Feature Engineering & Cleansing

<외부 데이터>

schools_df	subway_df	interest	seoul / busan money
‘dong_school’ ‘dong_highschool’	‘gu_subway’	‘대출금리’ ‘변화율’	‘평균소득’
동별 학교 수 동별 고등학교 수	구별 지하철 역 수	**변화율:** 이전 달 대출금리와 현재 달 대출금리 간 차이	동 별 월 평균 소득

02 Feature Engineering & Cleansing

앞서 생성한 수치형 피처들의 조합 → 18개의 추가 피처 생성

```
def feat(df):
    df['최강'] = df['평균소득'] * df['gu_subway'] * df['exclusive_use_area']
    df['최강'] = np.sqrt(df['최강'])
    df['부자'] = df['최강'] * df['top10'] * df['주상복합']
    df['subsubway'] = df['gu_subway'] ** 2
    df['exclusive_use_area2'] = df['exclusive_use_area'] ** 2
    df['대출금리2'] = df['대출금리'] **2
    df['층과_면적'] = df['floor'] * df['exclusive_use_area']
    df['학교_수'] = df['dong_highschool'] * df['dong_school']
    df['역학세권'] = df['학교_수'] * df['gu_subway']
    df['아파트'] = df['previous_transaction'] * df['apartment_age']
    df['운전학교'] = df['학교_수'] * df['vehicle_pop']
    df['종합'] = df['transformed'] * df['층과_면적'] * df['baby']
    df['상가'] = df['주상복합'] * df['floor_avg'] * df['exclusive_use_area']
    df['몰라'] =  df['number'] * df['gu_subway']
    df['돈'] = df['평균소득'] * df['층과_면적']
    df['층'] = df['floor_avg'] * df['대출금리'] * df['최강']
    df['주변'] = df['상가'] * df['역학세권']
    df['minus'] = df['vehicle_pop'] * df['대출금리']
    return df
```

03 Encoding& Scaling

- 값이 얼마 없는 city, transaction_date 는 원핫인코딩
- 만 개가 넘는 값들을 가진 dong, apt, gu는 이진인코딩 사용

‘city’
‘season’
‘transaction_date’

“OneHotEncoder”

‘dong’
‘apt’
‘gu’

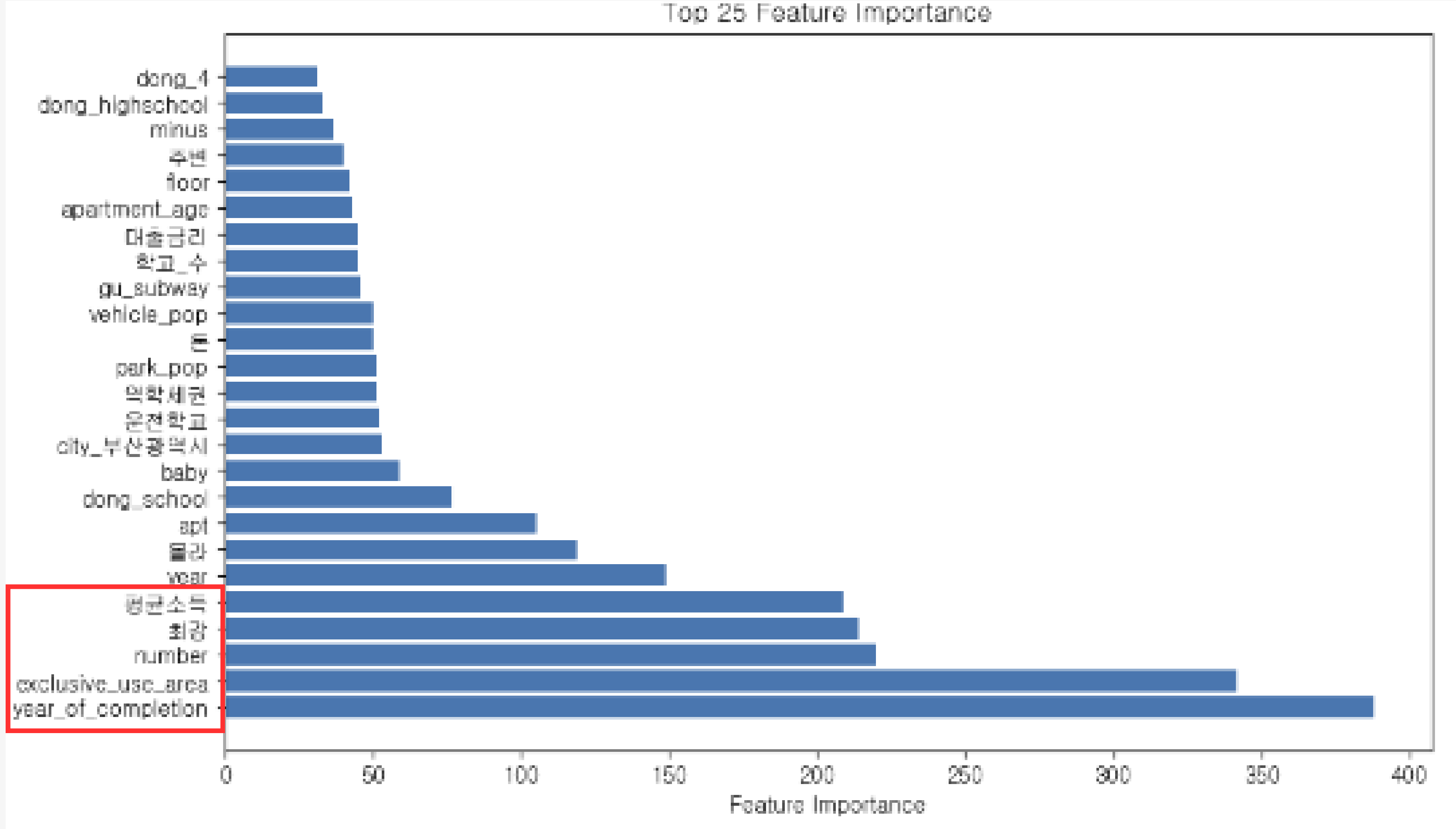
“BinaryEncoder”

Scaling은 후 모델링에서 파이프라인에 MinMaxScaler를 활용

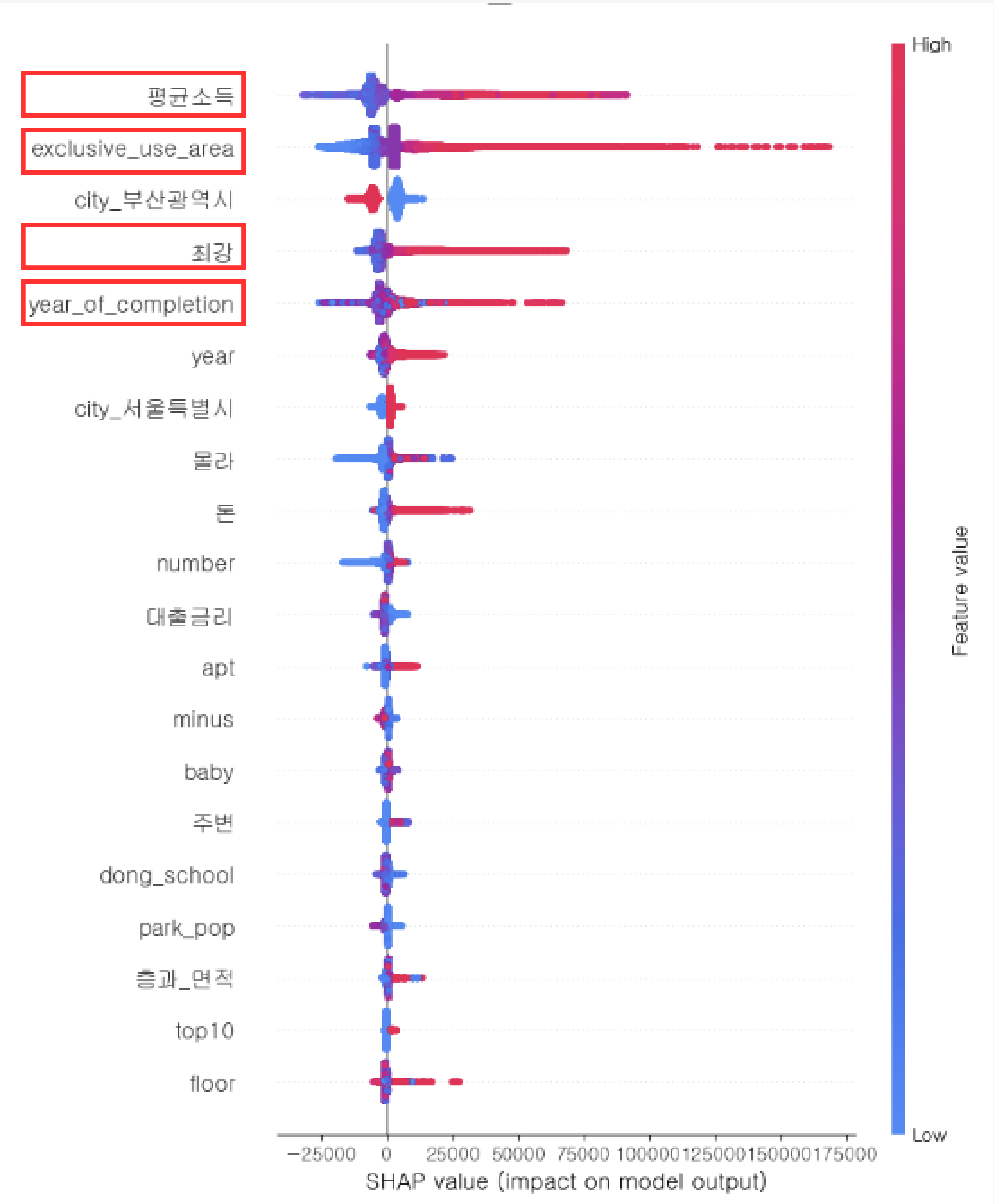
04. Modeling

04 Modeling

- Light GBM을 이용한 feature 중요도 확인



- shap을 이용한 feature 중요도 확인



'평균소득' , '대출금리', 'number(거래횟수)',
'최강(평균소득+구별 지하철 수+전용면적)' 등..
외부데이터를 사용한 열의 중요도가 높음

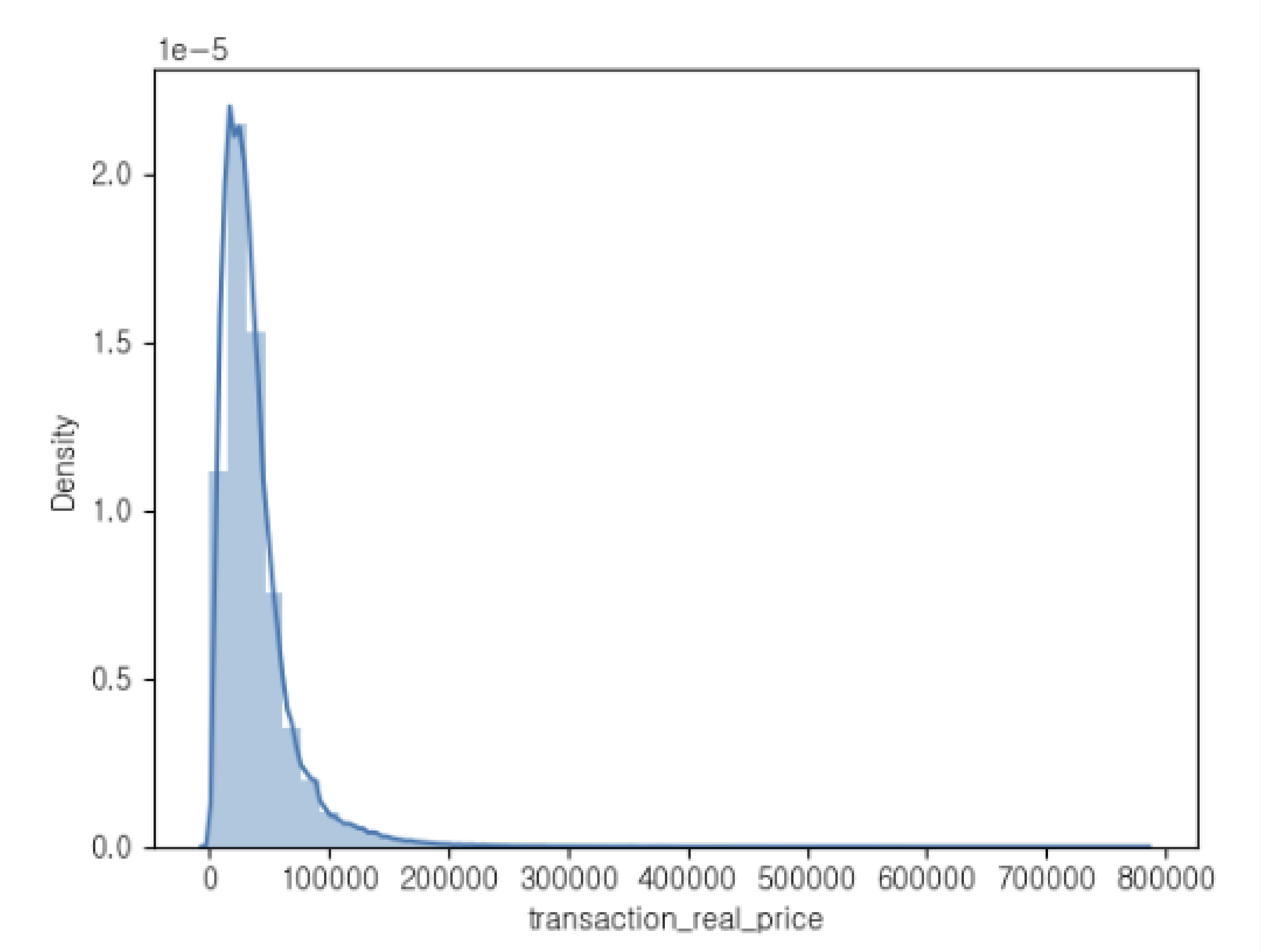
04 Modeling

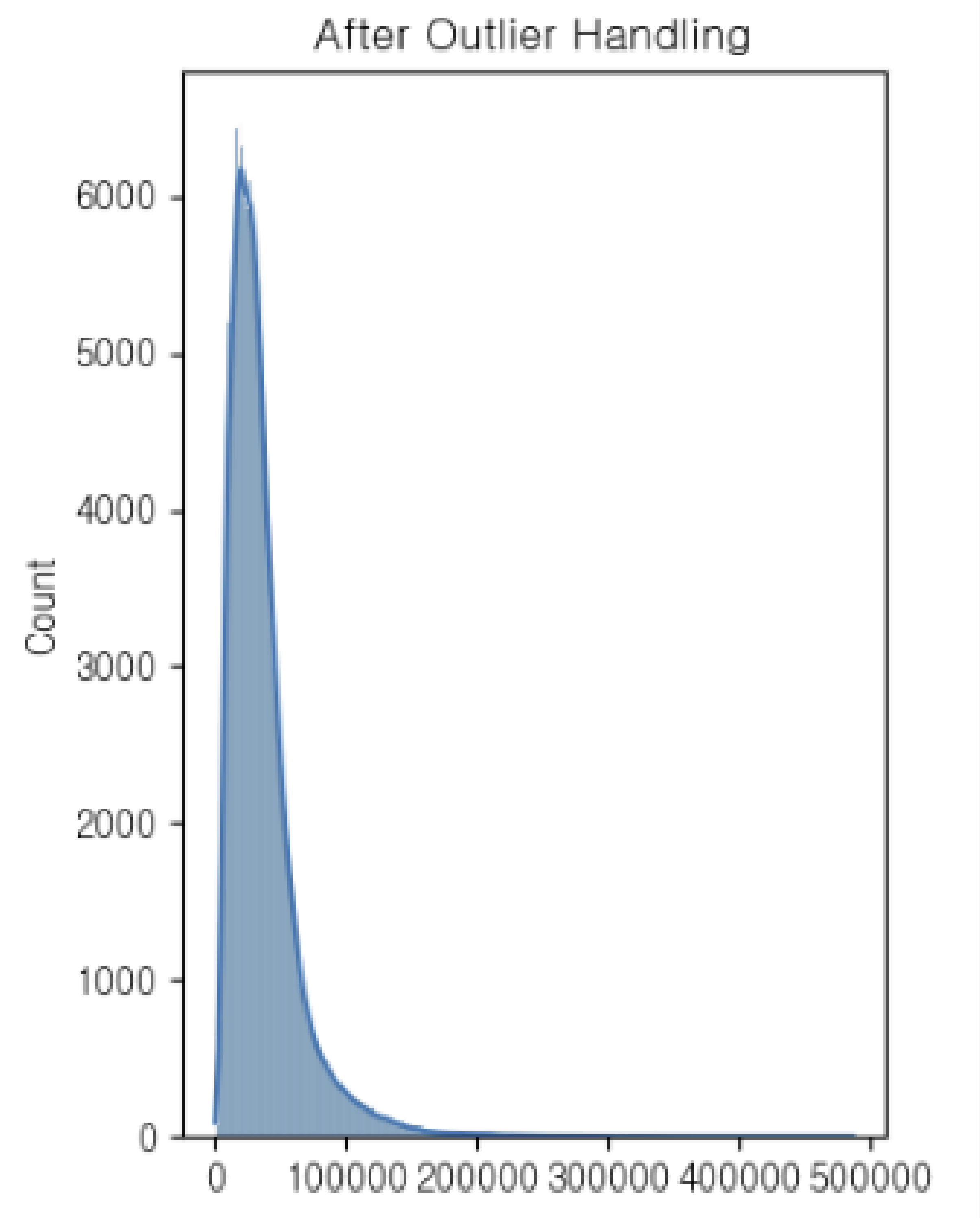
- Light GBM 단일모델 앙상블

```python
def rmse_score(y,y_pred):
    a = 0
    for i,j in zip(y,y_pred):
        a += (i-j)**2
    return np.sqrt(a/len(y))
from sklearn.model_selection import  KFold
from tqdm import tqdm
import lightgbm as lgb
lgbm = lgb.LGBMRegressor()
kf = KFold(n_splits=5, shuffle=True, random_state=42)
ensemble_predictions = []
scores = []
for train_idx, val_idx in tqdm(kf.split(features), total = 5, desc = "processing folds"):
    X_t, X_val = features.iloc[train_idx], features.iloc[val_idx]
    y_t, y_val = target.iloc[train_idx], target.iloc[val_idx]
    lgbm.fit(X_t,y_t)
    val_pred = lgbm.predict(X_val)
    #val_pred = np.exp(val_pred)
    #y_val = np.exp(y_val)
    scores.append(rmse_score(y_val, val_pred))

    lgbm_pred = lgbm.predict(test)
    #lgbm_pred = np.exp(lgbm_pred)
    lgbm_pred = np.where(lgbm_pred<0, 0, lgbm_pred)
    ensemble_predictions.append(lgbm_pred)
final_predictions = np.mean(ensemble_predictions,axis = 0)
print("Validation : RMSE scores for each fold:", scores)
print("Validation : RMSE", np.mean(scores))
```

04 Modeling

- Min Max Scaler 적용
- 이상치 처리





04 Modeling

- Hist Gradient Boosting Regressor model 적용

optuna튜닝을 통해 찾은 best parameter 를 적용

params = {'max_iter': 2414, 'max_leaf_nodes': 175, 'max_depth': 10, 'min_samples_leaf': 36, 'l2_regularization': 0.05704545550641109}

```
preprocessor = ColumnTransformer(
    transformers=[
        ('ss', MinMaxScaler(), features.columns),
    ], remainder='passthrough'
)

#Best Parameters using OPTUNA
params = {'max_iter': 2414, 'max_leaf_nodes': 175, 'max_depth': 10,
            'min_samples_leaf': 36, 'l2_regularization': 0.05704545550641109}

pipe = Pipeline(
    [
        ('MIN',preprocessor),
        ('HIST',HistGradientBoostingRegressor(random_state=42, loss='squared_error', **params))
    ]
)

pipe.fit(features,target)
```

04 Modeling

- Hist Gradient Boosting Regressor model & LightGBM model 앙상블

```
lgbm = lgb.LGBMRegressor()
kf = KFold(n_splits=5, shuffle=True, random_state=42)
ensemble_predictions = []
scores = []

for train_idx, val_idx in tqdm(kf.split(features), total=5, desc="processing folds"):
    X_t, X_val = features.iloc[train_idx], features.iloc[val_idx]
    y_t, y_val = target.iloc[train_idx], target.iloc[val_idx]

    lgbm.fit(X_t, y_t)
    val_pred_lgbm = lgbm.predict(X_val)
    scores.append(rmse_score(y_val, val_pred_lgbm))

    lgbm_pred = lgbm.predict(test)
    lgbm_pred = np.where(lgbm_pred < 0, 0, lgbm_pred)
    ensemble_predictions.append(lgbm_pred)

# lgbm 모델의 예측값에 대한 평균을 계산합니다.
final_predictions_lgbm = np.mean(ensemble_predictions, axis=0)

# HistGradientBoostingRegressor를 사용한 모델을 정의합니다.
preprocessor = ColumnTransformer(
    transformers=[
        ('ss', MinMaxScaler(), features.columns),
    ], remainder='passthrough'
)
```

```
params = {'max_iter': 1602, 'max_leaf_nodes': 166,
          'max_depth': 13, 'min_samples_leaf': 41,
          'l2_regularization': 0.07173583579993222}

hist = HistGradientBoostingRegressor(random_state=42, loss='squared_error', **params)

# HistGradientBoostingRegressor를 사용한 모델로 예측을 수행합니다.
pipe = Pipeline(
    [
        ('MIN', preprocessor),
        ('HIST', hist)
    ]
)

pipe.fit(features, target)
prediction_hist = pipe.predict(test).clip(0,)

# lgbm 모델과 HistGradientBoostingRegressor 모델의 예측값을 앙상블합니다.
final_predictions = (final_predictions_lgbm + prediction_hist) / 2

# Define the weights for each model
weight_lgbm = 0.7
weight_hist = 0.3

# Multiply the predictions by the respective weights
weighted_predictions_lgbm = final_predictions_lgbm * weight_lgbm
weighted_predictions_hist = prediction_hist * weight_hist

# Combine the weighted predictions with the specified weights
final_predictions_weighted = weighted_predictions_lgbm + weighted_predictions_hist
```

04 Modeling

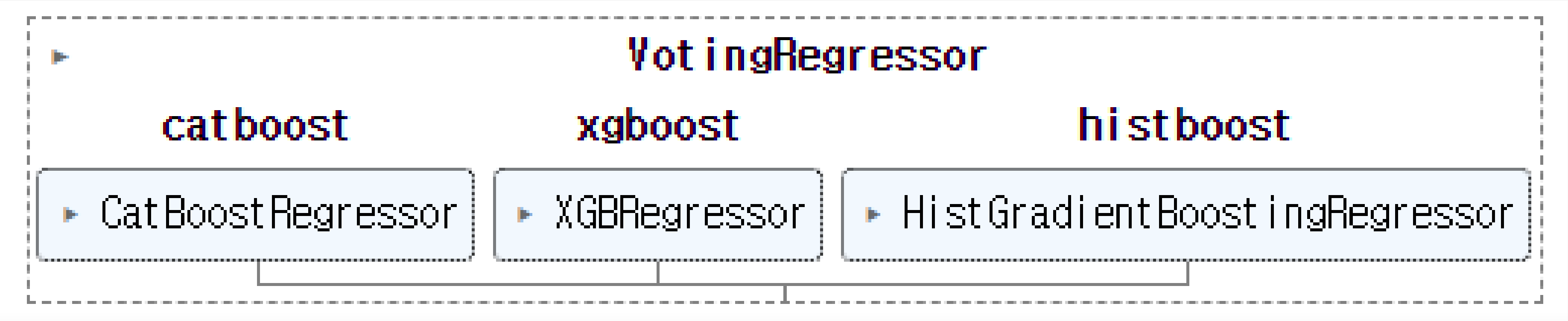
- catboost & XGboost 앙상블

```
from catboost import CatBoostRegressor
from xgboost import XGBRegressor
from sklearn.ensemble import VotingRegressor

# CatBoost 모델 정의
catboost_model = CatBoostRegressor(iterations=1000, depth=2, learning_rate=0.1, loss_function='RMSE', random_state=42)

# XGBoost 모델 정의
xgboost_model = XGBRegressor(n_estimators=1000, max_depth=2, learning_rate=0.1, objective='reg:squarederror', random_state=42)

# 앙상블 모델 생성
ensemble_model = VotingRegressor([('catboost', catboost_model), ('xgboost', xgboost_model)])

# 앙상블 모델 훈련
ensemble_model.fit(features_scaled, target)
```

04 Modeling

- catboost & XGboost & HistGradientBoostingRegressor 앙상블

```python
from catboost import CatBoostRegressor
from xgboost import XGBRegressor
from sklearn.ensemble import VotingRegressor
from sklearn.experimental import enable_hist_gradient_boosting
from sklearn.ensemble import HistGradientBoostingRegressor
from sklearn.preprocessing import MinMaxScaler
from sklearn.model_selection import train_test_split
import pandas as pd

# MinMaxScaler를 사용하여 데이터 스케일링
scaler = MinMaxScaler()
features_scaled = scaler.fit_transform(features)


# CatBoost 모델 정의
catboost_model = CatBoostRegressor(iterations=1000, depth=6, learning_rate=0.1, loss_function='RMSE', random_state=42)

# XGBoost 모델 정의
xgboost_model = XGBRegressor(n_estimators=1000, max_depth=6, learning_rate=0.1, objective='reg:squarederror', random_state=42)

# HistGradientBoostingRegressor 모델 정의
params = {'max_iter': 1669, 'max_leaf_nodes': 167, 'max_depth': 13,
          'min_samples_leaf': 38, 'l2_regularization': 0.06837989932119798}
histboost_model = HistGradientBoostingRegressor(random_state=42, loss='squared_error', **params)

# 앙상블 모델 생성
ensemble_model = VotingRegressor([('catboost', catboost_model), ('xgboost', xgboost_model), ('histboost', histboost_model)])

# 앙상블 모델 훈련
ensemble_model.fit(features_scaled , target)
```



04 Modeling

- 최종 모델 선택

"Hist Gradient Boosting Regressor model"

RMSE 값 : 6422.94058

감사합니다