



Optiver 데이터를 활용한 주식 종가 가격 예측 모델

3조_ 권영찬
차석희
서준범
견규원
정재우
김혜규

Overview

주제선정 이유

01

데이터 분석 전처리

02

모델 분석

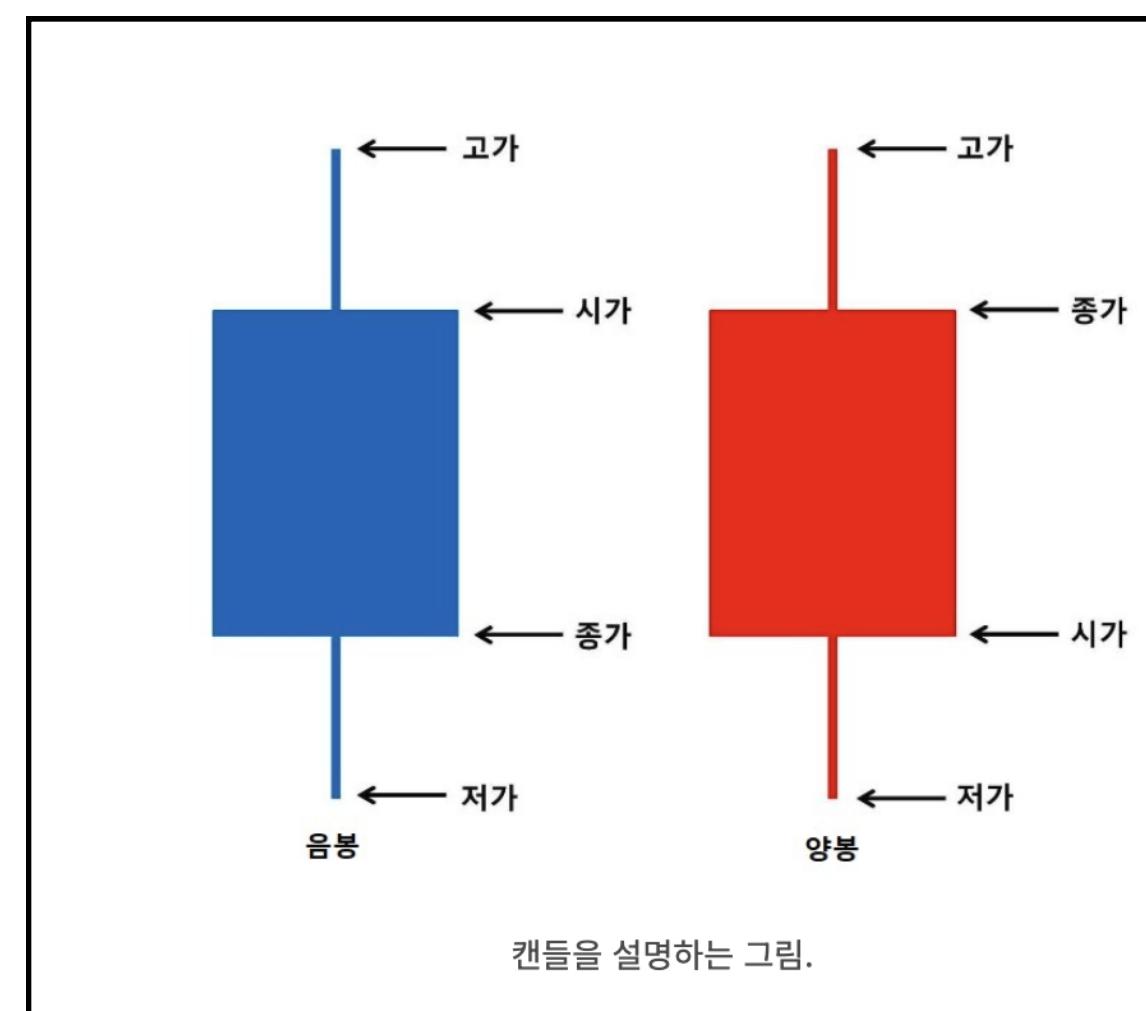
03

1. 주제 선정 이유

주제 선정 배경

fdr 라이브러리를 통해, 불러온 주식 데이터중,
종가를 기준으로 프로젝트를 진행

주식 데이터의 다양한 지표에 사용되는 '종가'



- 종가는 주식 시장에서 그날 마지막으로 형성된 가격
- 종가는 다양한 보조지표에 사용
- 대다수의 주식관련 데이터와 보조지표는 '종가'가 기준인 경우가 많음



우리가 생각하는 고래의 색은 파란색

하지만, 실제로 고래를 가까이서 보게 된다면,
색은 초록색에 가깝다.

이와 비슷하게, 우리가 보는 종가도

멀리서 볼때는 단순한 가격이지만,

종가가 만들어지는 원리를 이해하면서

자세히 들여다 보게 되면, 종가는 과연 무슨색일까?

주제 선정 배경

그럼, 이렇게 중요한
종가는 어떻게 만들어지는 거지?

1. 종가가 만들어 지는 원리

종가가 만들어지는
방식

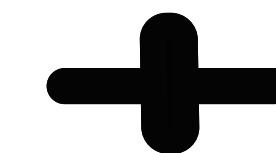
=
=



1. 종가가 만들어 지는 원리

주식 사는 사람의 주문서

메뉴			
메뉴	가격	수량	금액
소갈비찜	9,000		
공기밥+단장김치찌개 포함 * 소갈비찜 2리본 이상 포함됩니다 *			
갈비탕 칼국수	6,500		
갈비찜 비빔면 (면 종류 선택)	5,500		
어린이 돈까스	5,500		
동치미 국수	4,500		
간장비빔 달면	4,500		
콩기밥	1,000		
된장 추가	2,000		



주식 파는 사람의 주문서

메뉴			
메뉴	가격	수량	금액
소갈비찜	9,000		
공기밥+단장김치찌개 포함 * 소갈비찜 2리본 이상 포함됩니다 *			
갈비탕 칼국수	6,500		
갈비찜 비빔면 (면 종류 선택)	5,500		
어린이 돈까스	5,500		
동치미 국수	4,500		
간장비빔 달면	4,500		
콩기밥	1,000		
된장 추가	2,000		

45,950 ▼	250	-0.54%	6,722,834 ↗	82.15%
증감	45,950	45,900	308,388	0.11%
	35,524	46,400	KOSPI 200	투
	7,712	46,350	45,850 시	거
	50,762	46,300	46,150 고	외
	26,130	46,250	45,450 저	일
	79,862	46,200	46,200 기준	차
	46,499	46,150	60,000 상	뉴
	53,062	46,100	32,350 하	권
	77,829	46,050	153 비용	
	126,481	46,000	45,950 예상	
	99,627	45,950	545,874 수량	
			▼ 250 -0.54%	기
45,950	545,874	45,900	15,513	
45,850	20	45,850	10,660	
45,850	1	45,800	64,223	
45,850	4	45,750	109,271	매수물량
45,850	2	45,700	119,031	
45,850	1,000	45,650	157,243	
45,850	20	45,600	109,478	
45,850	100	45,550	77,680	
45,850	10	45,500	207,124	
45,850	20	45,450	103,833	
	603,488	15:35:27	974,056	
10	2,344	시간외	11,254	

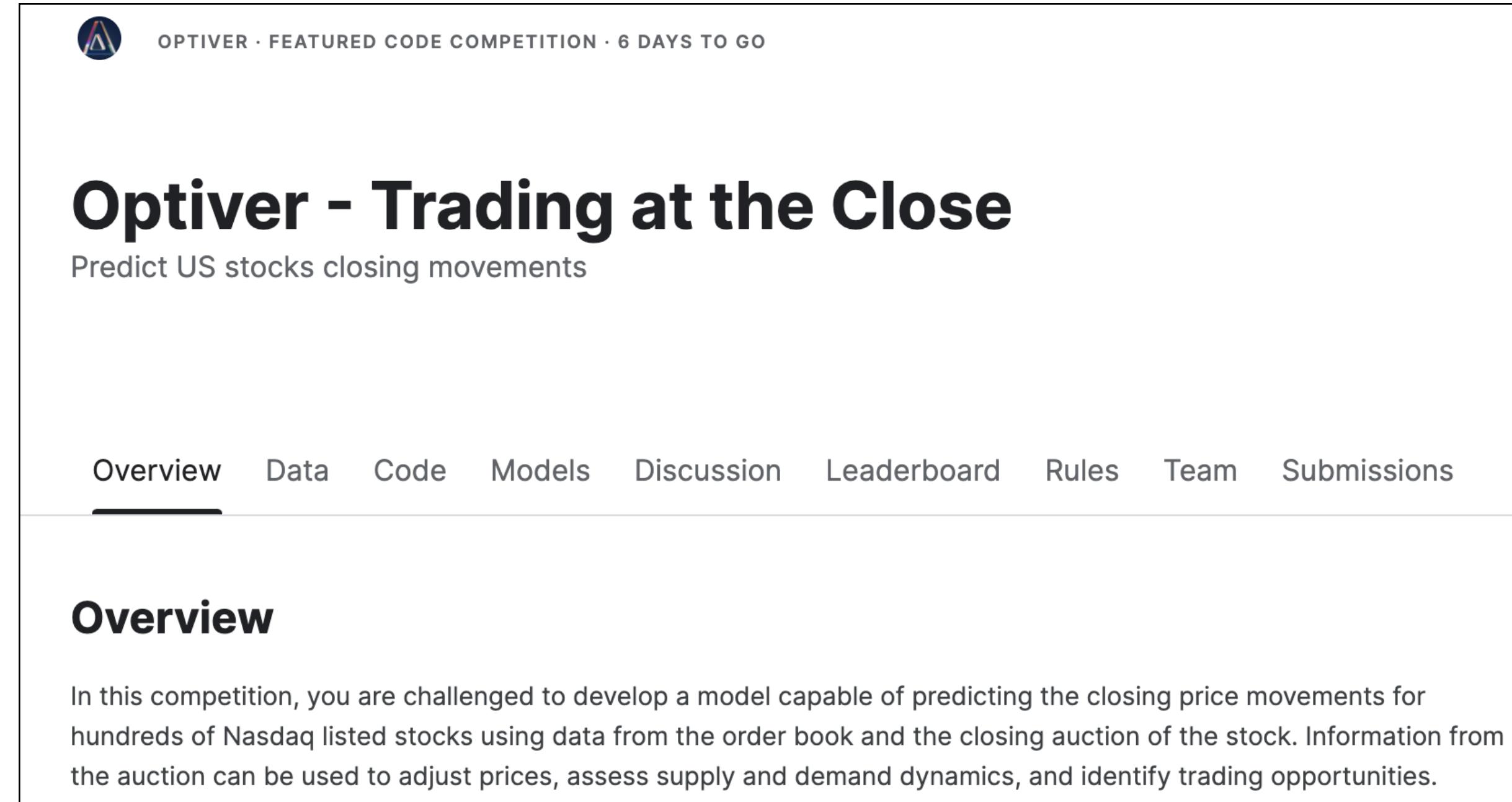
차이점

1. 가격을 직접 적어야 한다.

2. 사는 사람, 파는 사람 둘 다
가격과 수량을 적어야 한다.

2.데이터 분석 및 전처리

데이터 개요 및 출처



The screenshot shows the homepage of the Optiver - Trading at the Close competition. At the top left is the Optiver logo, followed by the text "OPTIVER · FEATURED CODE COMPETITION · 6 DAYS TO GO". Below this is the main title "Optiver - Trading at the Close" in large bold letters, with the subtitle "Predict US stocks closing movements" underneath. A horizontal navigation bar follows, with "Overview" underlined and other options: Data, Code, Models, Discussion, Leaderboard, Rules, Team, and Submissions. The "Overview" section contains the following text: "In this competition, you are challenged to develop a model capable of predicting the closing price movements for hundreds of Nasdaq listed stocks using data from the order book and the closing auction of the stock. Information from the auction can be used to adjust prices, assess supply and demand dynamics, and identify trading opportunities."

- 17 columns
- 비식별화된 장외 거래 데이터
- 당일 주식의 마감가 기준 경매

데이터 살펴보기 / columns

	stock_id	date_id	seconds_in_bucket	imbalance_size	imbalance_buy_sell_flag	reference_price	matched_size	far_price	near_price	bid_price	bid_size	ask_price	ask_size	wap	target	time_id	row_id
0	0	0	0	3180602.69	1	0.999812	13380276.64	0.0	0.0	0.999812	60651.50	1.000026	8493.03	1.000000	-3.029704	0	0_0_0
1	0	0	10	1299772.70	1	1.000026	15261106.63	0.0	0.0	0.999812	13996.50	1.000026	23519.16	0.999892	0.389814	1	0_10_0
2	0	0	20	1299772.70	1	0.999919	15261106.63	0.0	0.0	0.999812	4665.50	0.999919	12131.60	0.999842	4.220009	2	0_20_0
3	0	0	30	1299772.70	1	1.000133	15261106.63	0.0	0.0	1.000026	55998.00	1.000133	46203.30	1.000085	5.450249	3	0_30_0
4	0	0	40	1218204.43	1	1.000455	15342674.90	0.0	0.0	1.000241	14655.95	1.000455	26610.45	1.000317	3.169775	4	0_40_0
5	0	0	50	1218204.43	1	1.000455	15342674.90	0.0	0.0	1.000348	42012.00	1.000455	9897.22	1.000434	0.599623	5	0_50_0
6	0	0	60	1218204.43	1	1.000562	15342674.90	0.0	0.0	1.000455	14005.50	1.000562	10085.04	1.000517	-0.200272	6	0_60_0
7	0	0	70	1264494.89	1	1.000455	15352380.96	0.0	0.0	1.000348	37904.16	1.000455	17366.82	1.000421	2.410412	7	0_70_0
8	0	0	80	1189832.86	1	1.000241	15427043.00	0.0	0.0	1.000133	9427.34	1.000241	61984.40	1.000148	-0.389814	8	0_80_0
9	0	0	90	1189272.89	1	1.000562	15427602.97	0.0	0.0	1.000348	23340.00	1.000562	40433.54	1.000426	-4.339814	9	0_90_0
10	0	0	100	1249282.50	1	1.000348	15427602.97	0.0	0.0	1.000241	9801.75	1.000348	42572.16	1.000261	-2.049804	10	0_100_0

시간 데이터

- **date_id**
- **seconds_in_bucket**
- **time_id**
- **row_id**

금액 데이터

- **reference_price**
- **ask_price**
- **bid_price**

주문수량 데이터

- **matched_size**
- **imbalance_size**
- **ask_size**
- **bid_size**

데이터프레임 확인

	stock_id	date_id	seconds_in_bucket	imbalance_size	imbalance_buy_sell_flag	reference_price	matched_size	far_price	near_price	bid_price	bid_size	ask_price	ask_size	wap	target	time_id	row_id
0	0	0	0	3180602.69	1	0.999812	13380276.64	0.0	0.0	0.999812	60651.50	1.000026	8493.03	1.000000	-3.029704	0	0_0_0
191	0	0	10	1299772.70	1	1.000026	15261106.63	0.0	0.0	0.999812	13996.50	1.000026	23519.16	0.999892	0.389814	1	0_10_0
382	0	0	20	1299772.70	1	0.999919	15261106.63	0.0	0.0	0.999812	4665.50	0.999919	12131.60	0.999842	4.220009	2	0_20_0
573	0	0	30	1299772.70	1	1.000133	15261106.63	0.0	0.0	1.000026	55998.00	1.000133	46203.30	1.000085	5.450249	3	0_30_0
764	0	0	40	1218204.43	1	1.000455	15342674.90	0.0	0.0	1.000241	14655.95	1.000455	26610.45	1.000317	3.169775	4	0_40_0
955	0	0	50	1218204.43	1	1.000455	15342674.90	0.0	0.0	1.000348	42012.00	1.000455	9897.22	1.000434	0.599623	5	0_50_0
1146	0	0	60	1218204.43	1	1.000562	15342674.90	0.0	0.0	1.000455	14005.50	1.000562	10085.04	1.000517	-0.200272	6	0_60_0
1337	0	0	70	1264494.89	1	1.000455	15352380.96	0.0	0.0	1.000348	37904.16	1.000455	17366.82	1.000421	2.410412	7	0_70_0
1528	0	0	80	1189832.86	1	1.000241	15427043.00	0.0	0.0	1.000133	9427.34	1.000241	61984.40	1.000148	-0.389814	8	0_80_0
1719	0	0	90	1189272.89	1	1.000562	15427602.97	0.0	0.0	1.000348	23340.00	1.000562	40433.54	1.000426	-4.339814	9	0_90_0
1910	0	0	100	1249282.50	1	1.000348	15427602.97	0.0	0.0	1.000241	9801.75	1.000348	42572.16	1.000261	-2.049804	10	0_100_0
2101	0	0	110	1277280.77	1	1.000133	15399604.70	0.0	0.0	1.000026	5039.82	1.000133	28375.36	1.000042	0.020266	11	0_110_0
2292	0	0	120	1216057.90	1	1.000133	15460827.57	0.0	0.0	0.999812	43482.46	1.000026	68224.23	0.999895	2.290011	12	0_120_0
2483	0	0	130	1216057.90	1	1.000026	15460827.57	0.0	0.0	0.999812	32658.50	1.000026	13999.50	0.999962	1.000166	13	0_130_0
2674	0	0	140	1104904.79	1	0.999919	15571980.68	0.0	0.0	0.999705	33028.20	0.999919	25196.40	0.999826	-1.699925	14	0_140_0

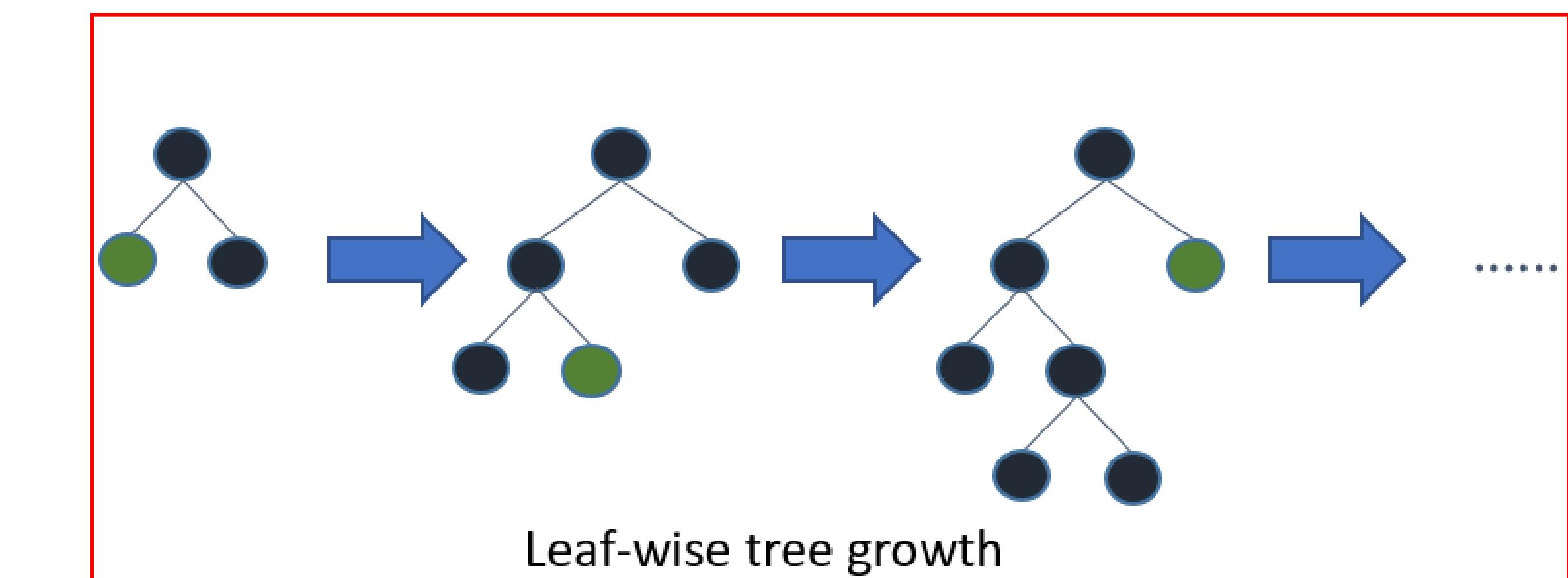
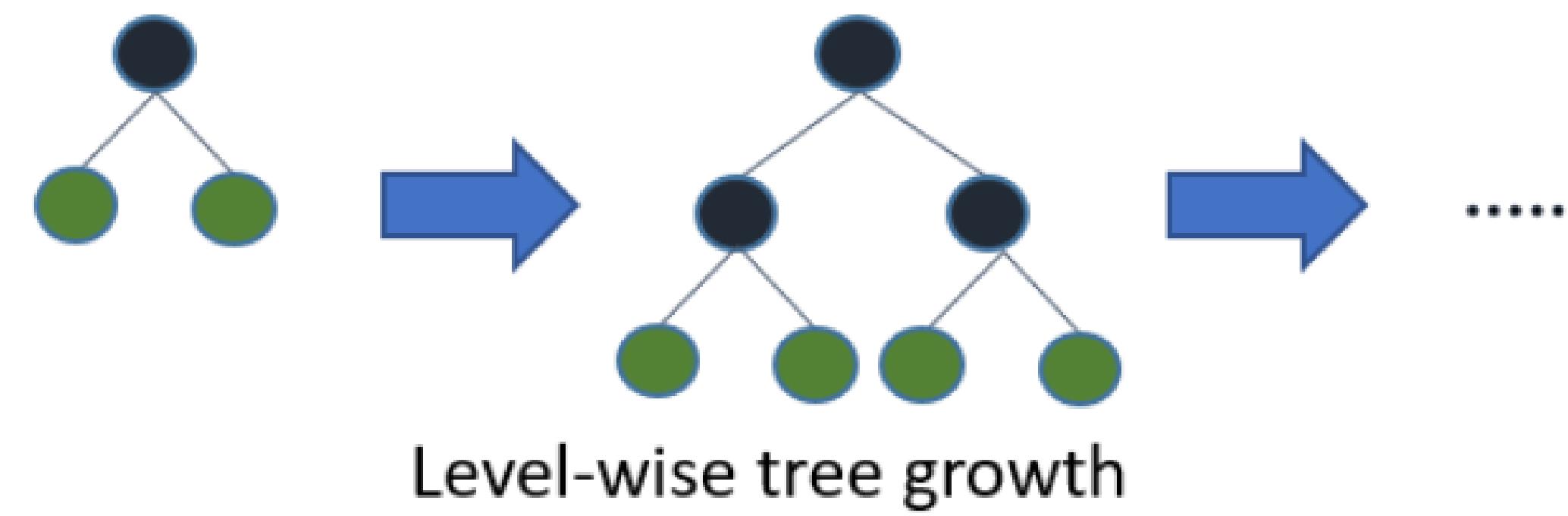
데이터 프레임 id별로 찍어보기

결측치 처리

```
# 결측치 대체  
df = df.fillna(0)
```

fillna(0) 함수를 이용해 0으로 대체해줌

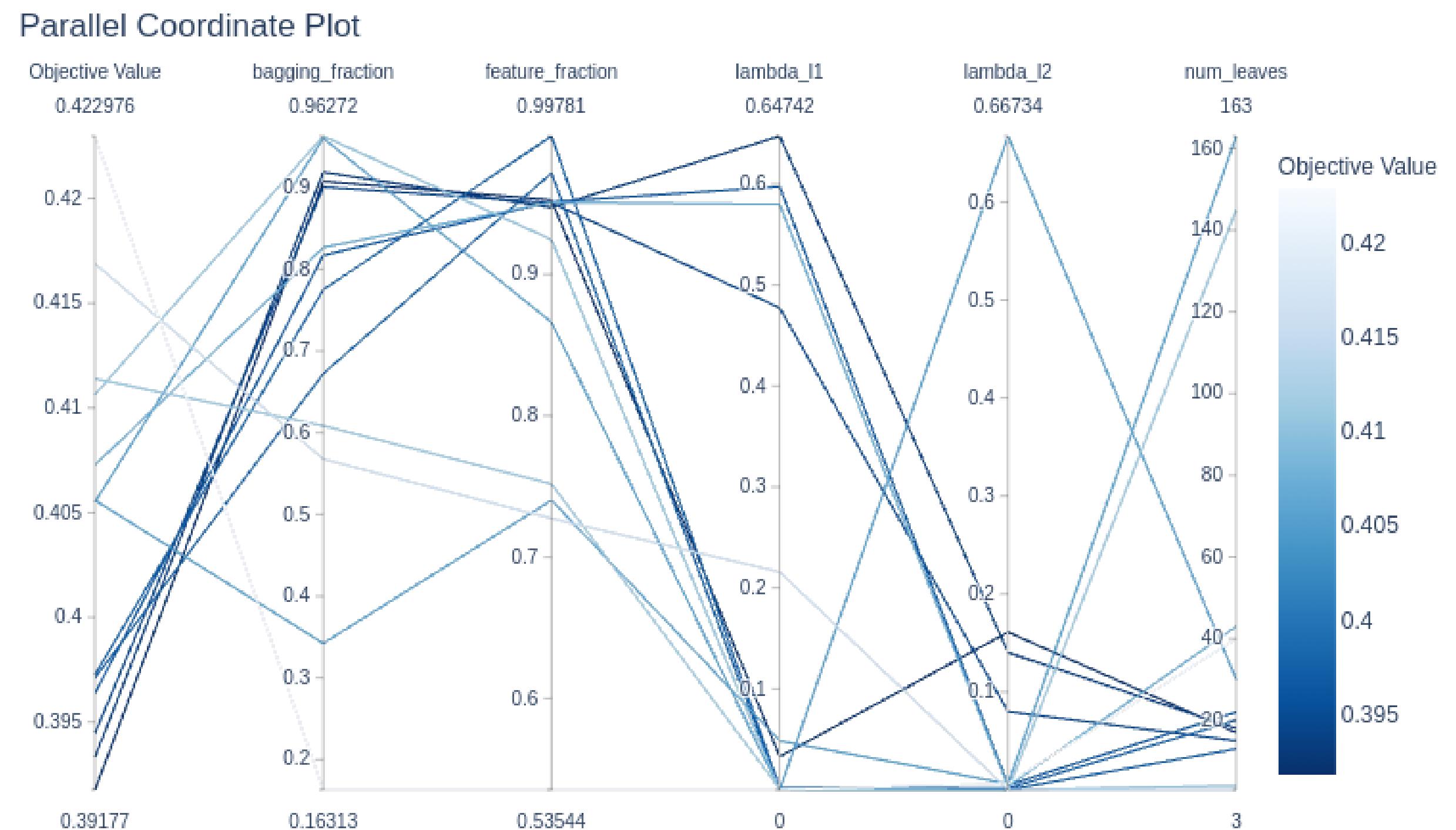
LGBM (LightGBM)



LGBM (LightGBM)

- Tree 기반 학습 알고리즘
- Gradient Boost -발전-> XGBoost -속도향상-> LightGBM
- XGBoost에 비해 훈련 시간이 짧고, 성능이 개선된 모델
- Tree 기준 분할이 아닌 Leaf 기준 분할 (수직 확장)
- 동일한 leaf를 확장할 때, leaf-wise 알고리즘은 level-wise 알고리즘보다 더 많은 loss, 손실을 줄일 수 있음
- 트리의 균형을 맞추지 않고 최대 손실 값을 갖는 리프 노드를 지속적으로 분할하면서 깊고 비대칭적인 트리를 생성

Optimize parameter with Optuma



Optuna란?

- 기계 학습 모델의 성능을 최대화하기 위해 하이퍼파라미터 튜닝을 자동화하는 것을 목표로 하는 AutoML 기법 중 하나
 - 베이지안 최적화, 유전 알고리즘, 그리드 서치 등 다양한 최적화 알고리즘을 사용하여 모델의 성능을 향상시키는데 사용

Optimize parameter result with Optuna

```
def objective(trial):
    params = {
        'random_seed': 123,
        'n_estimators': trial.suggest_int('n_estimators', 300, 1000),
        'num_leaves': trial.suggest_int('num_leaves', 4, 32),
        'max_depth': trial.suggest_int("max_depth", 1, 10)}

    model = lgbm.LGBMRegressor(**params)
    model.fit(x, y)
    y_pred = model.predict(x)
    score = mean_absolute_error(y, y_pred)

    return score
```

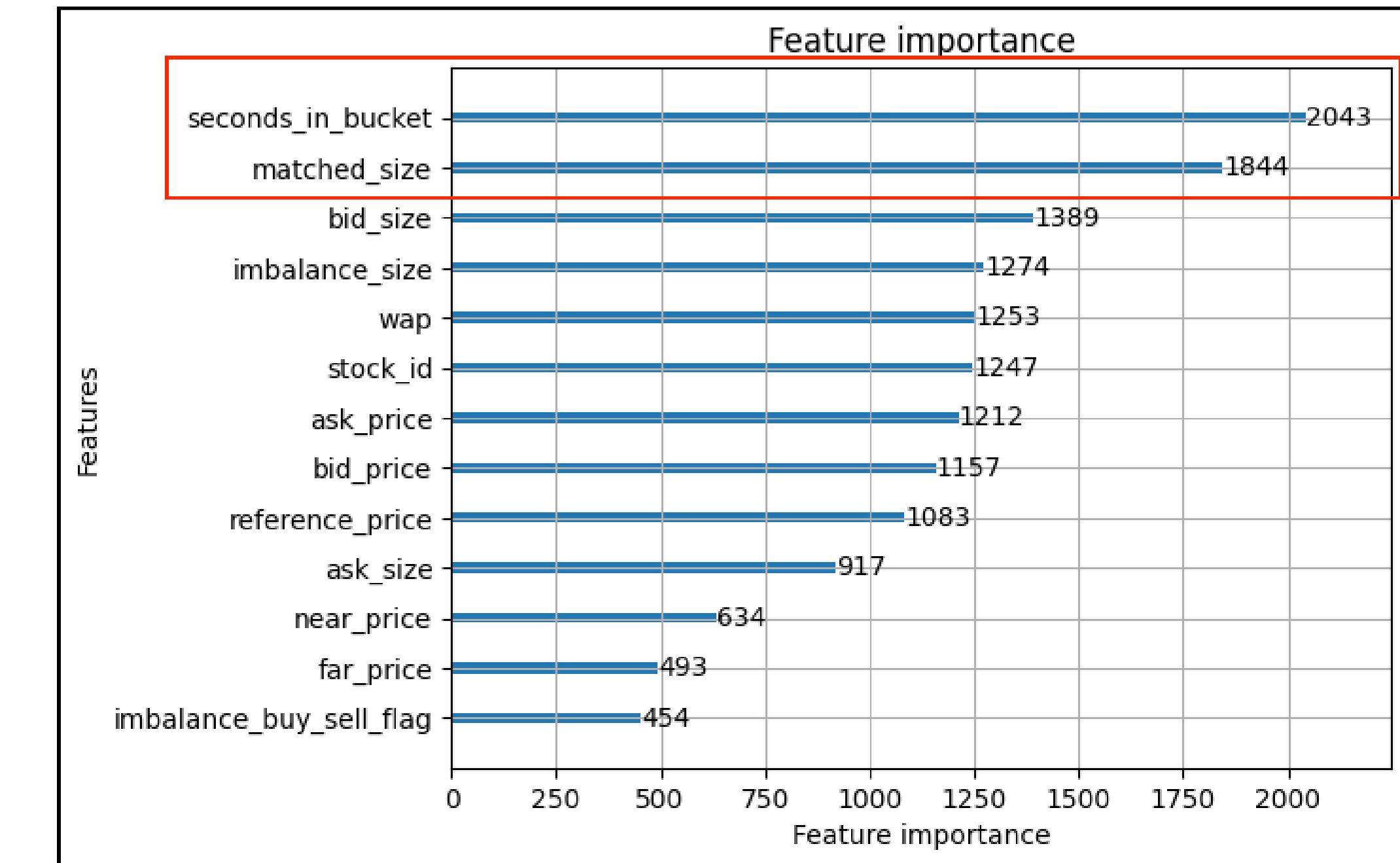
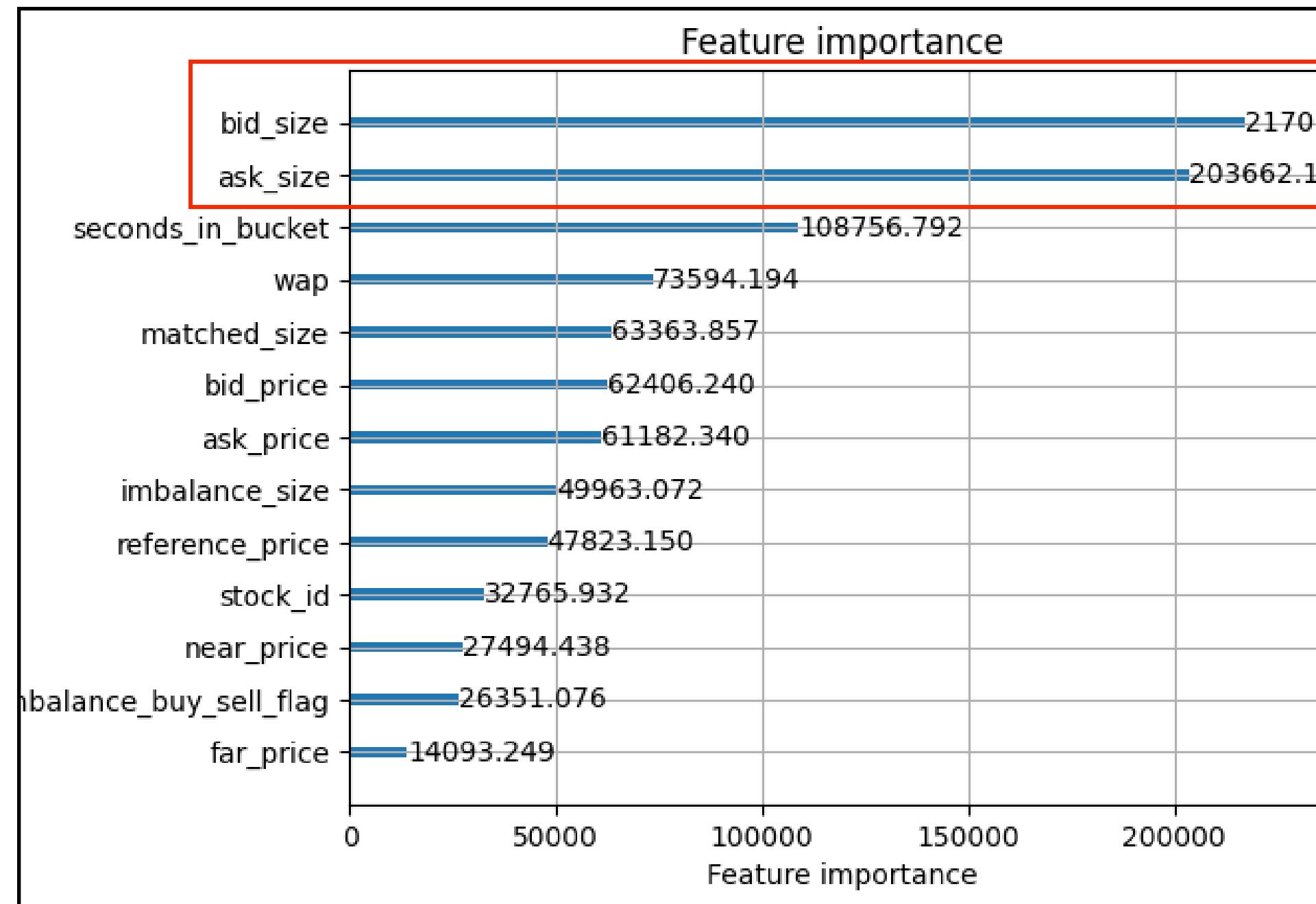
파라미터

- `n_estimators`: 트리 모델에서 생성할 트리의 개수
- `num_leaves`: 트리의 각 수준(level)에서의 최대 리프(leaf) 개수
- `max_depth`: 각 결정 트리의 최대 깊이를 제한하는 파라미터

장점

- 사용자가 원하는 최적화 전략을 선택할 수 있음
- 병렬 처리를 통해 여러 하이퍼 파라미터 조합을 동시에 평가하여 최적화 속도를 높임

변수 중요도



- 각 특성이 분할에서 제공한 정보 획득량(Information Gain)을 기준으로 했을 때
: **bid_size, ask_size**
- 모델 내의 모든 트리에서 데이터를 분할하는데 사용된 특정 피처의 빈도수를 기준으로 했을 때
: **seconds_in_bucket, matched_size**

모델 성능을 높이기 위한 보조지표 추가

```
def pre_process1(df):
    df['imbalance_ratio'] = df['imbalance_size'] / df['matched_size']
    df['imbl_size1'] = (df['bid_size']-df['ask_size']) / (df['bid_size']+df['ask_size'])
    df['imbl_size2'] = (df['imbalance_size']-df['matched_size']) / (df['imbalance_size']+df['matched_size'])
    df['bid_size_diff'] = df[["stock_id", "date_id", "bid_size"]].groupby(["stock_id", "date_id"]).diff()
    df['ask_size_diff'] = df[["stock_id", "date_id", "ask_size"]].groupby(["stock_id", "date_id"]).diff()
    df["bid_size_over_ask_size"] = df["bid_size"].div(df["ask_size"])
    df["bid_price_over_ask_price"] = df["bid_price"].div(df["ask_price"])
    df['ShortEMA'] = df['reference_price'].ewm(span=short_period, min_periods=1).mean()
    df['LongEMA'] = df['reference_price'].ewm(span=long_period, min_periods=1).mean()
    df['MACD'] = df['ShortEMA'] - df['LongEMA']
    df['SignalLine'] = df['MACD'].ewm(span=signal_period, min_periods=1).mean()
    df['MACD_Histogram'] = df['MACD'] - df['SignalLine']
    df['new_size'] = df['imbalance_size'] * df['imbalance_buy_sell_flag'] * abs(df['ask_price'] - df['bid_price'])
    return df
```

MACD(Moving Average Convergence Divergence):

주가의 기술적 분석에 사용되는 지표

파라미터 재조정 및 각 모델의 결과 확인

```
params = {  
    'task': 'train',  
    'boosting_type': 'gbdt',  
    'objective': 'regression',  
    'metric': ['l1', 'l2'],  
    'learning_rate': 0.005,  
    'feature_fraction': 0.9,  
    'bagging_fraction': 0.7,  
    'bagging_freq': 10,  
    'verbose': 0,  
    "max_depth": 8,  
    "num_leaves": 20,  
    "max_bin": 512,  
    "num_iterations": 1000,  
    "force_col_wise": 'true'  
}  
  
lgbm_model = lgbm.LGBMRegressor(**params)  
lgbm_model.fit(x_train, y_train)
```

LightGBM 모델의 MAE: 4.0618

XGBoost 모델의 MAE: 5.9139

3. 모델 분석

3. Model

개요

1) Training Data Set Splitting

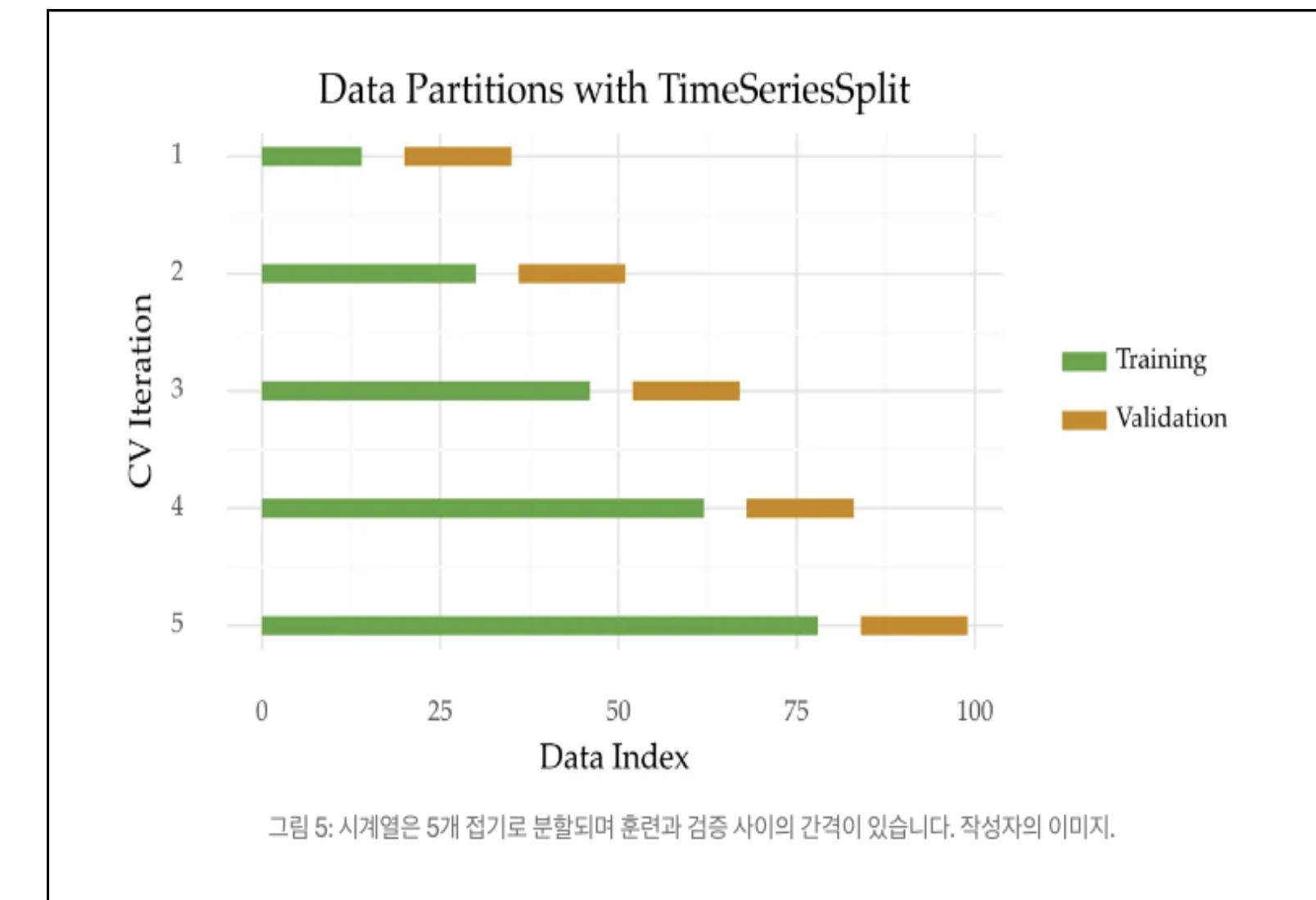
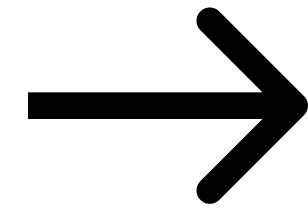
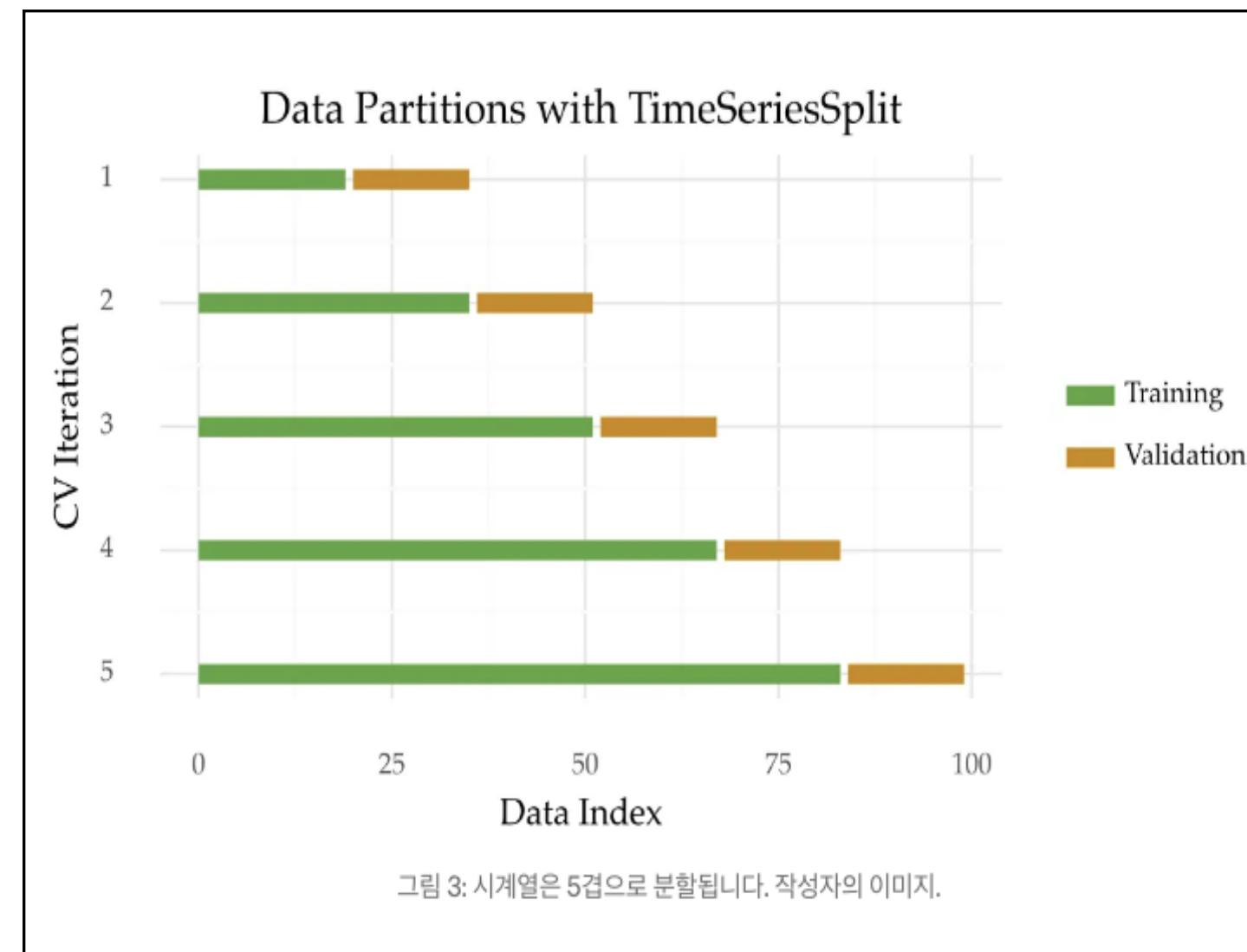
2) Modeling

3) Training

4) Evaluation

3. Model

1) Training Data Set Splitting



(1) Types of Split Data Sets : [Train, Validation, Test]

(2) Group TimeSeries Split : [stock_id]

(3) K-Fold : K개의 데이터 분할

3. Model

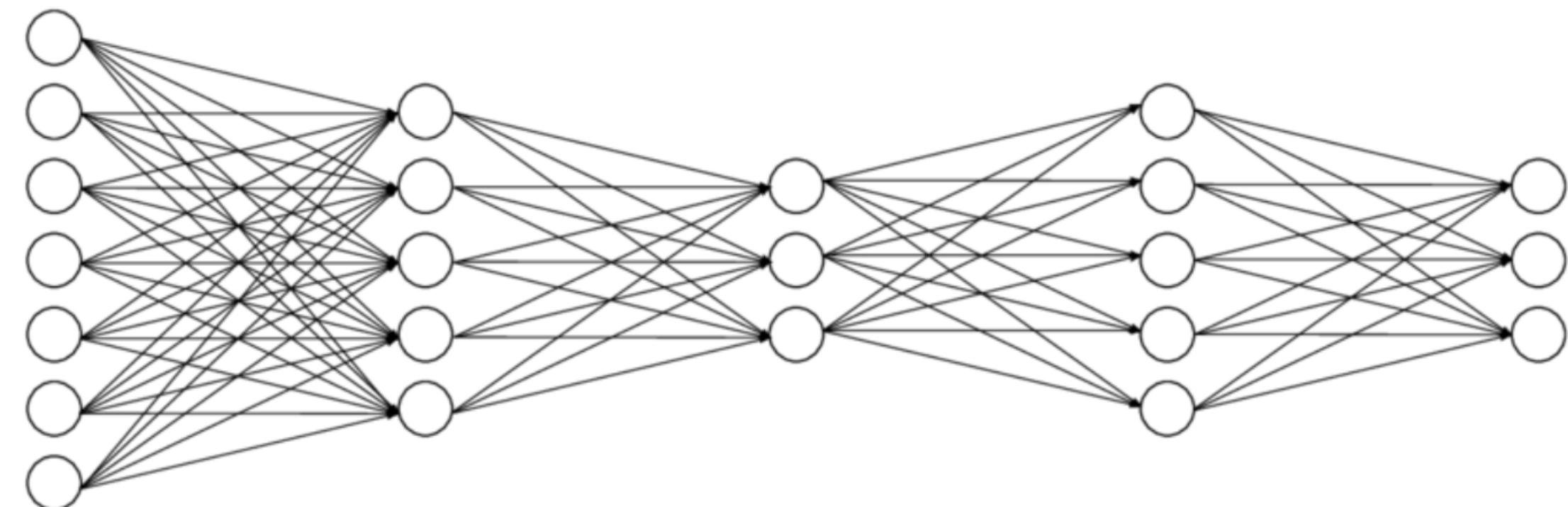
2) Modeling

Modeling

(1) Select : [RNN, Decision Tree, Feed-Forward NN with Embedding]

(2) Decision Tree

(3) Feed-Forward NN Modeling



- *Embedding*을 필두로 한 *FeedForwardNeuralNetwork(FNN, 순방향 신경망)*
- *Embedding*이 범주형 feature를 밀집 벡터로 변환한 후, (*one-hot-encoding*의 개선)
- 변환된 feature 혹은 feature들과 함께 *FNN*모델에 기입한다.

3. Model

3) *Training*

(1) K-Fold

(2) Early Stop

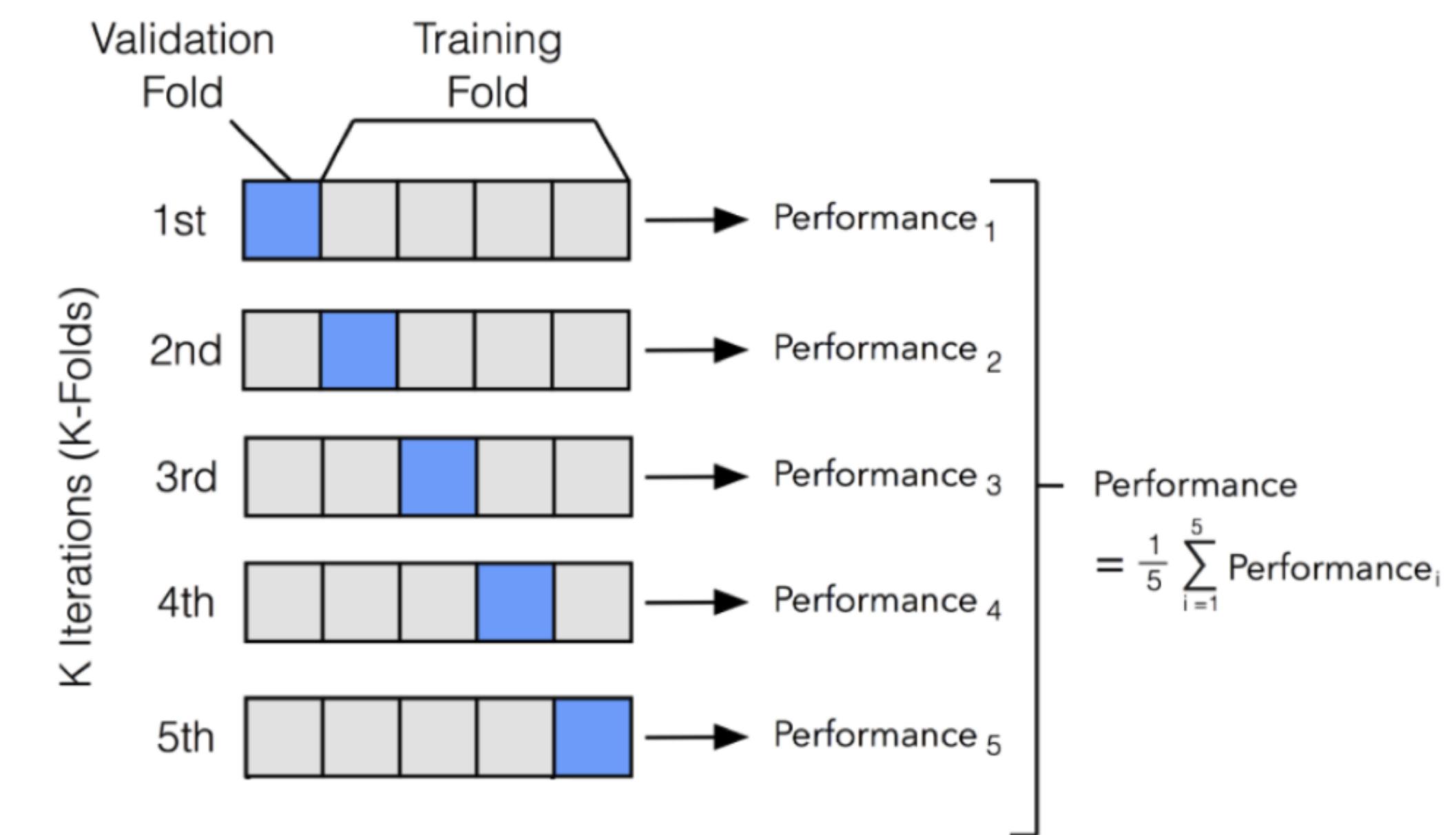
(3) Hyper-Parameter Tuning

3. Model

3) *Training*

데이터를 k 개의 부분집합으로 분할하고, 각 부분집합을 한 번씩 검증 데이터로 사용하여 모델을 k 번 학습하고 평가하는 교차 검증

K-Fold

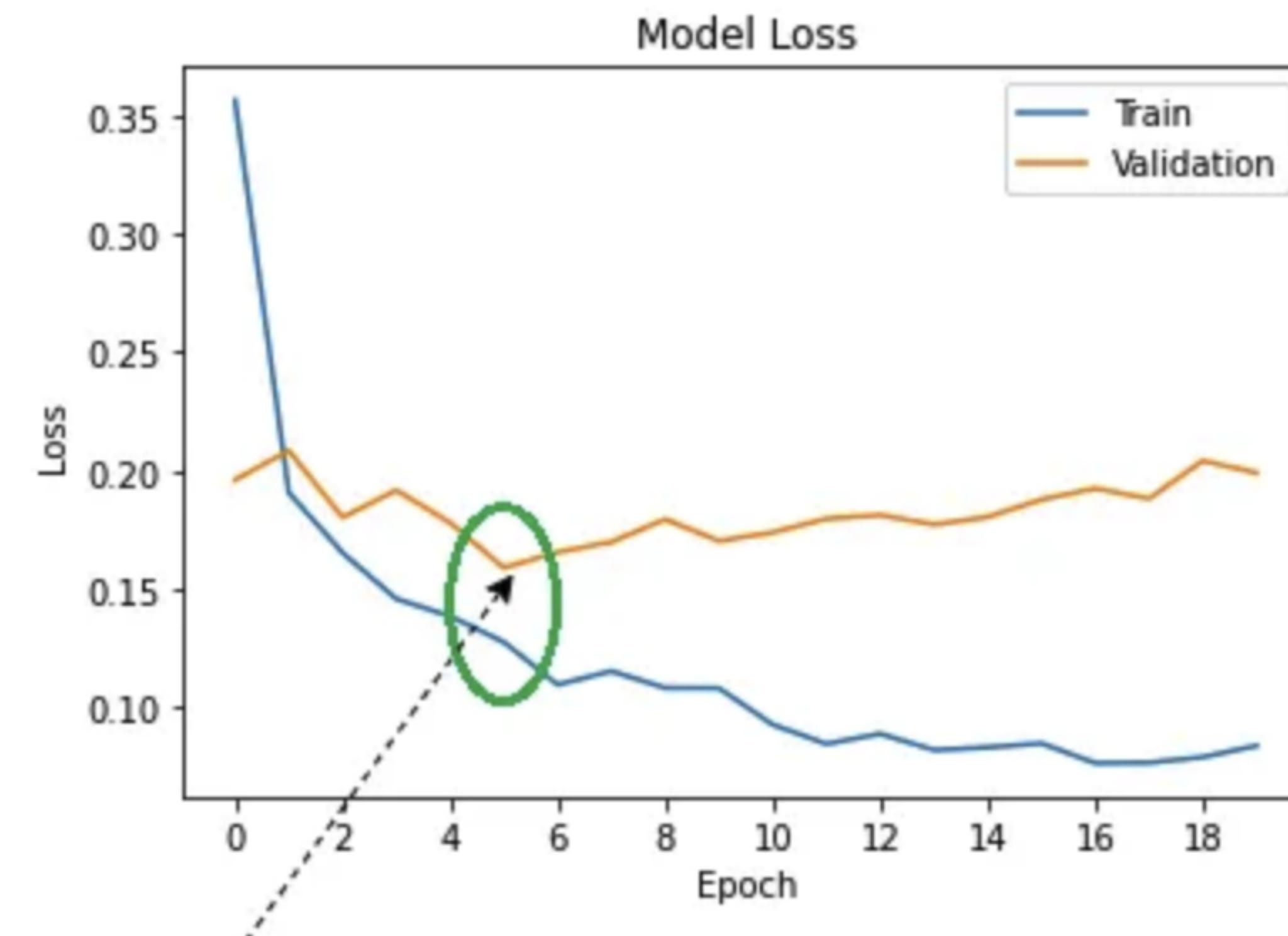


3. Model

3) *Training*

모델 학습 중 검증 성능이 더 이상 향상되지 않을 때 학습을 조기 종료시켜 과적합을 방지

Early Stop



3. Model

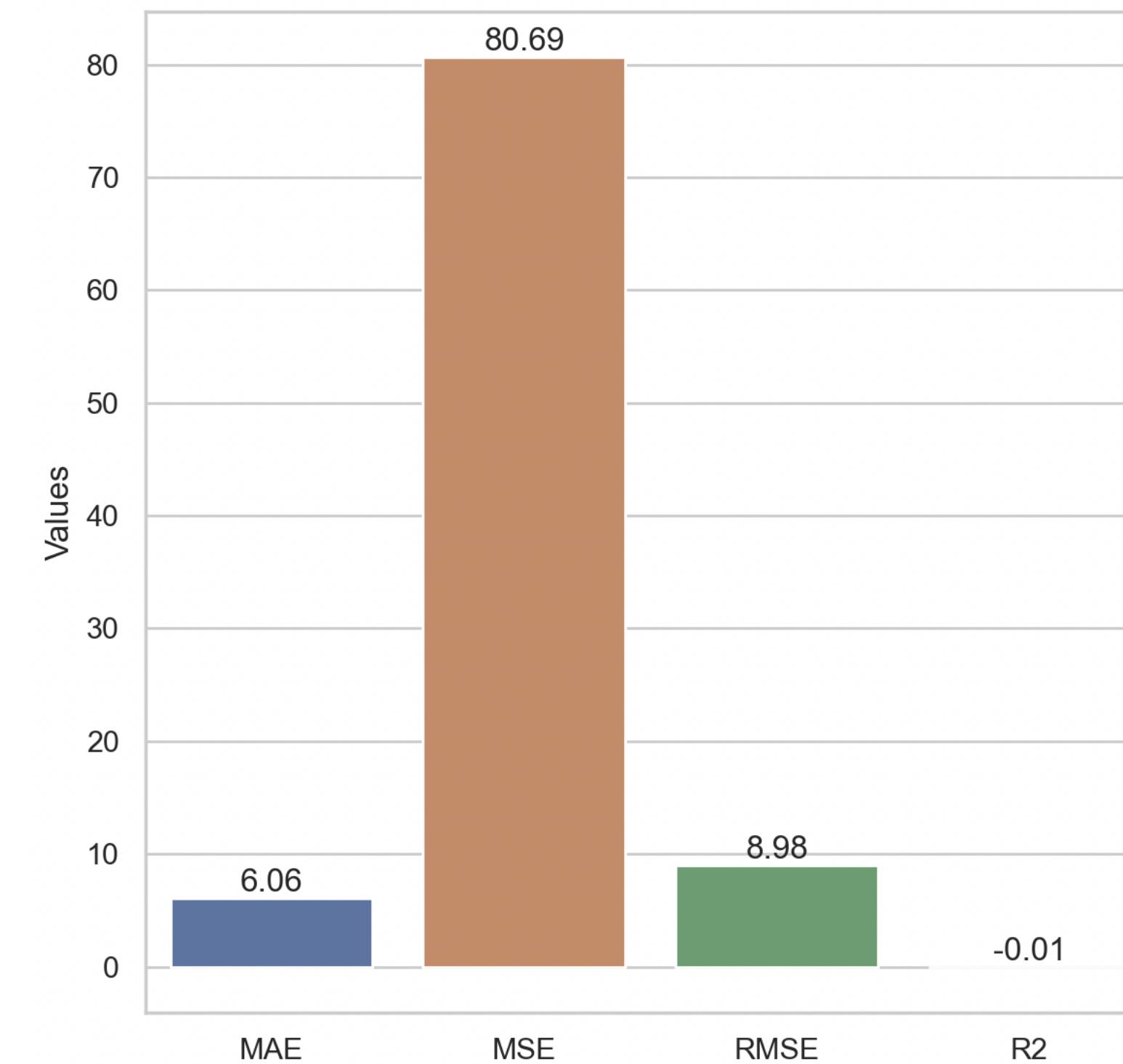
4) Evaluation

- (1) MAE(Mean Absolute Error): 예측값과 실제값 간의 절대적인 평균 오차
- (2) MAPE(Mean Absolute Percentage Error): 백분율 기준의 평균 절대 오차
- (3) MSE(Mean Squared Error): 예측값과 실제값 간의 제곱 평균 오차
- (4) RMSE(Root Mean Squared Error): MSE의 제곱근으로 예측 오차의 평균 제곱근
- (5) R2(R-squared): 모델이 설명하는 변동의 비율

3. Model

4) Total_Evaluation_best

Validation Set



Test Set

