

2024학년도 2학기

문제해결프로그래밍 강의 5주차

조기필

2024.10.10



문제 1 : 자전거 대여 수요 예측

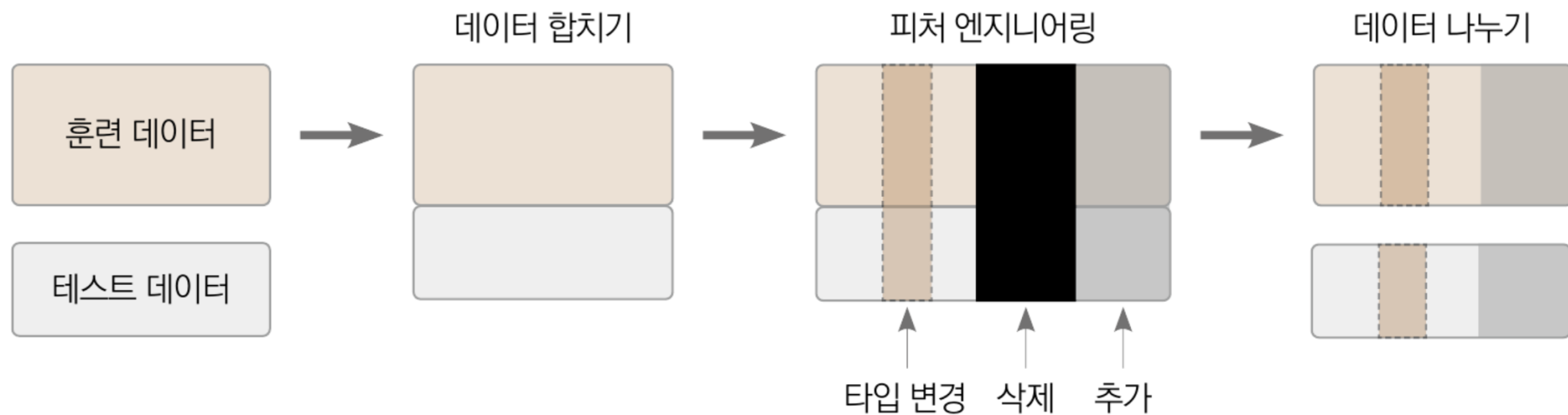
지난시간 복습

자전거 대여 수요 예측

1. 다양한 회귀 모델 훈련 및 성능 검증

문제 1 : 자전거 대여 수요 예측

▼ 피처 엔지니어링 전후의 데이터 합치기 및 나누기



문제 1 : 자전거 대여 수요 예측

지난번 데이터를 살펴본 정보를 바탕으로 데이터 정리를 요약해보자.

1. 시간 변수에서 새로운 특성 추출 : 연도, 월, 시간, 요일
2. 이상치 제거 : 날씨 4 ← 데이터가 하나뿐임
3. 필요 없는 특성 제거 : casual, registered, datetime, windspeed
4. 중복성을 띄는 특성 제거 : season 과 month 둘 중 하나

문제 1 : 자전거 대여 수요 예측

피처 선택이란?

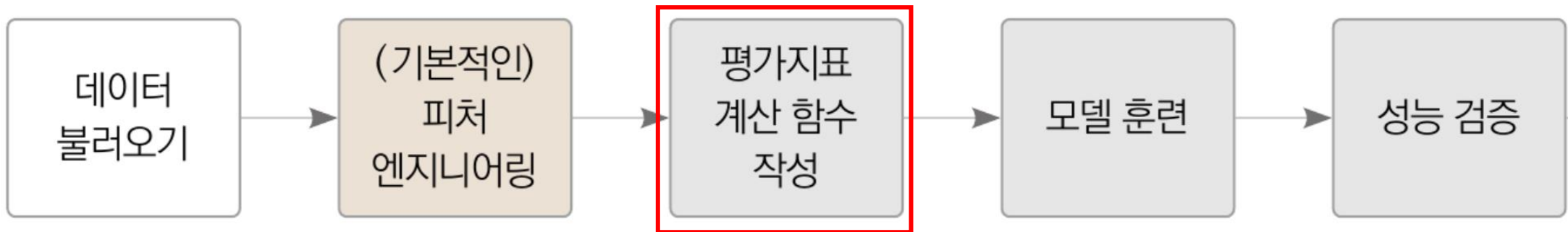
모델링 시 데이터의 특징을 잘 나타내는 주요 피처만 선택하는 작업을 **피처 선택**^{feature selection}이라고 합니다. 피처 선택은 머신러닝 모델 성능에 큰 영향을 줍니다. 타깃값 예측과 관련 없는 피처가 많다면 오히려 예측 성능이 떨어집니다. 피처가 많다고 무조건 좋은 게 아니라는 말입니다. 예측 성능을 높이려면 타깃값과 관련 있는 피처가 필요합니다.

피처 선택 방법에 정답은 없습니다. 어떤 피처를 선택해야 성능이 가장 좋을지 바로 알 방법은 없습니다. 탐색적 데이터 분석, 피처 중요도^{feature importance}, 상관관계 매트릭스 등을 활용해 종합적으로 판단해야 합니다.

문제 1 : 자전거 대여 수요 예측

평가지표를 만들어보자.

▼ 베이스라인 모델 전체 프로세스



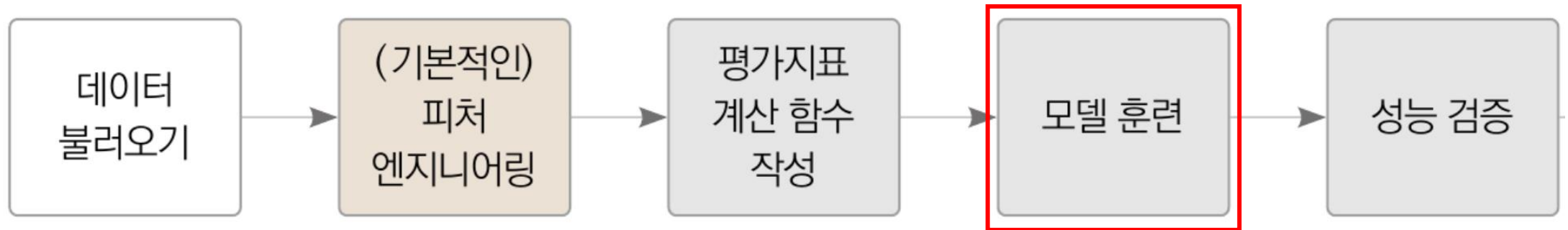
문제 1 : 자전거 대여 수요 예측

| 회귀 평가지표 | 수식 | 설명 |
|----------------|---|---|
| MAE | $\frac{1}{N} \sum_{i=1}^N y_i - \hat{y} $ | 평균 절대 오차Mean Absolute Error. 실제 타깃값과 예측 타깃값 차의 절댓값 평균 |
| MSE | $\frac{1}{N} \sum_{i=1}^N (y_i - \hat{y})^2$ | 평균 제곱 오차Mean Squared Error. 실제 타깃값과 예측 타깃값 차의 제곱의 평균 |
| RMSE | $\sqrt{\frac{1}{N} \sum_{i=1}^N (y_i - \hat{y})^2}$ | 평균 제곱근 오차Root Mean Squared Error. MSE에 제곱근을 취한 값 |
| MSLE | $\frac{1}{N} \sum_{i=1}^N (\log(y_i + 1) - \log(\hat{y}_i + 1))^2$ | Mean Squared Log Error. MSE에서 타깃값에 로그를 취한 값 |
| RMSLE | $\sqrt{\frac{1}{N} \sum_{i=1}^N (\log(y_i + 1) - \log(\hat{y}_i + 1))^2}$ | Root Mean Squared Log Error. MSLE에 제곱근을 취한 값 |
| R ² | $\frac{\hat{\sigma}^2}{\sigma^2}$ | 결정계수. 예측 타깃값의 분산 / 실제 타깃값의 분산 * 다른 지표와 다르게 1에 가까울수록 모델 성능이 좋습니다. |

문제 1 : 자전거 대여 수요 예측

먼저 기본 선형회귀 모델을 사용하여 훈련 및 성능 검증을 해보자.

▼ 베이스라인 모델 전체 프로세스



문제 1 : 자전거 대여 수요 예측

알고 있는 값

훈련 : $Y = \theta_0 + \theta_1 x_1 + \theta_2 x_2 + \theta_3 x_3$

구하려는 값

Y 종속변수(타깃값)

θ 회귀계수(가중치)

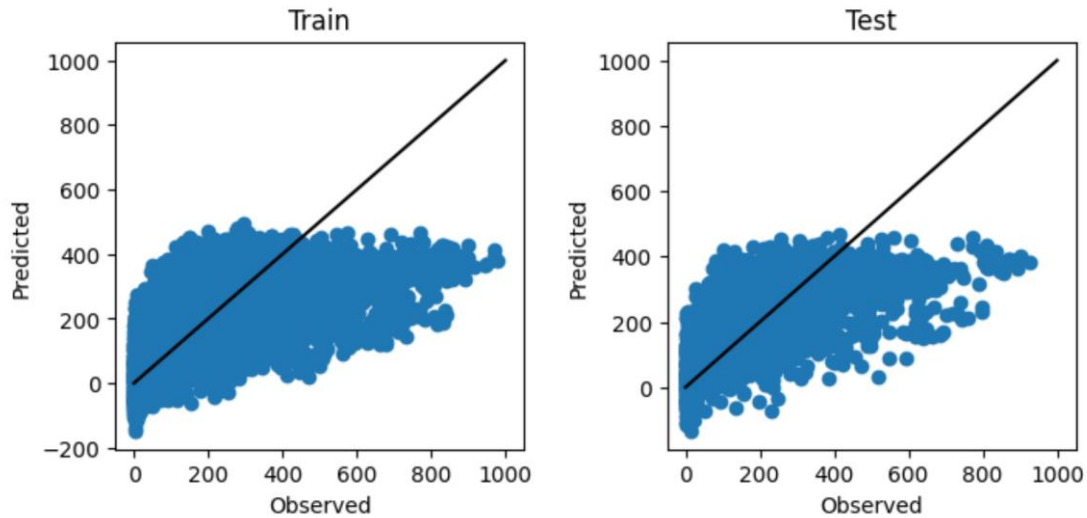
x 독립변수(피쳐)

예측 : $Y = \theta_0 + \theta_1 x_1 + \theta_2 x_2 + \theta_3 x_3$

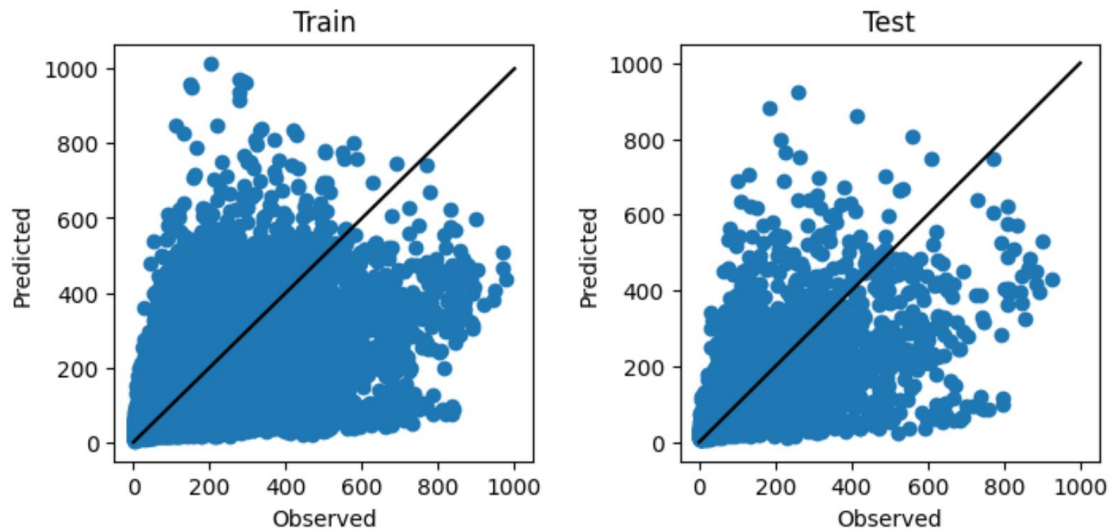
알고 있는 값

구하려는 값

문제 1 : 자전거 대여 수요 예측

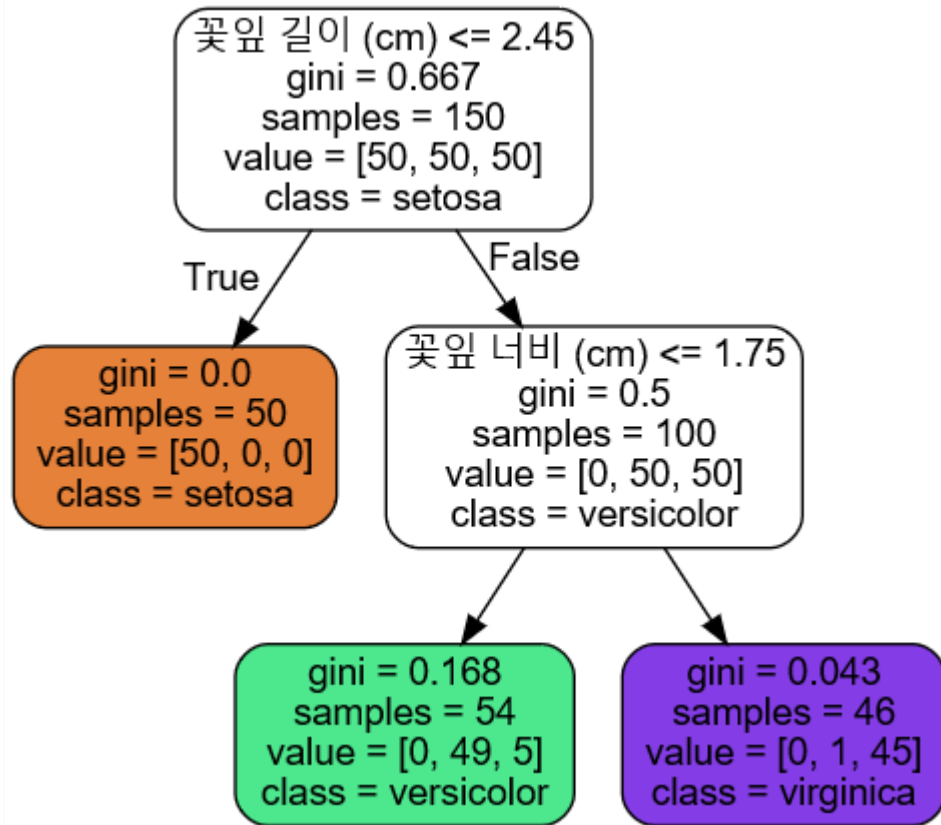


그대로 예측한 것과 로그 변환후 예측한 것이 단순히 정확도만 차이 나는 것이 아니라 예측에 대한 방향성에도 차이가 있음을 알 수 있다.

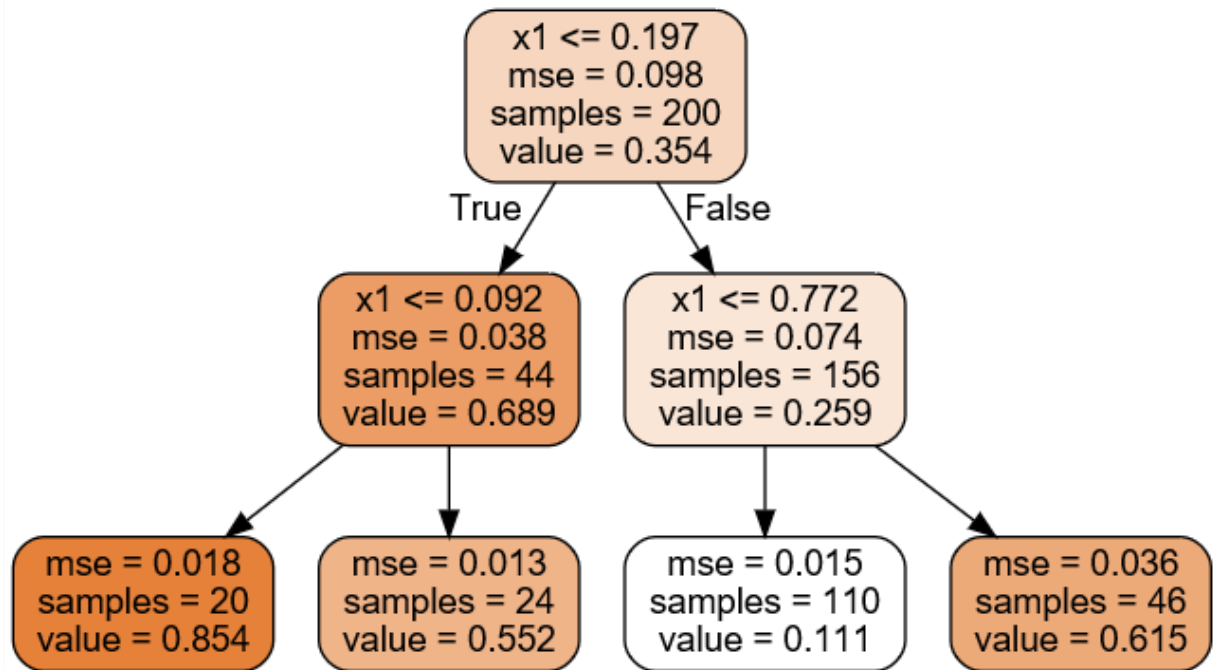


문제 1 : 자전거 대여 수요 예측

- 분류와 회귀 모두 사용 가능한 알고리즘



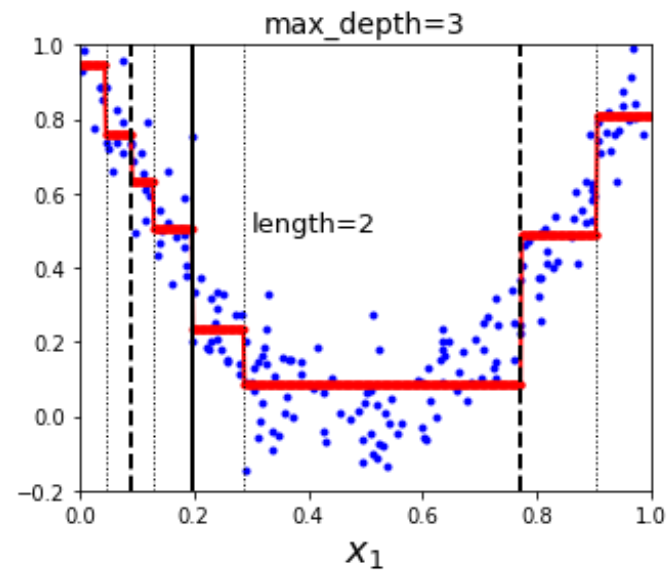
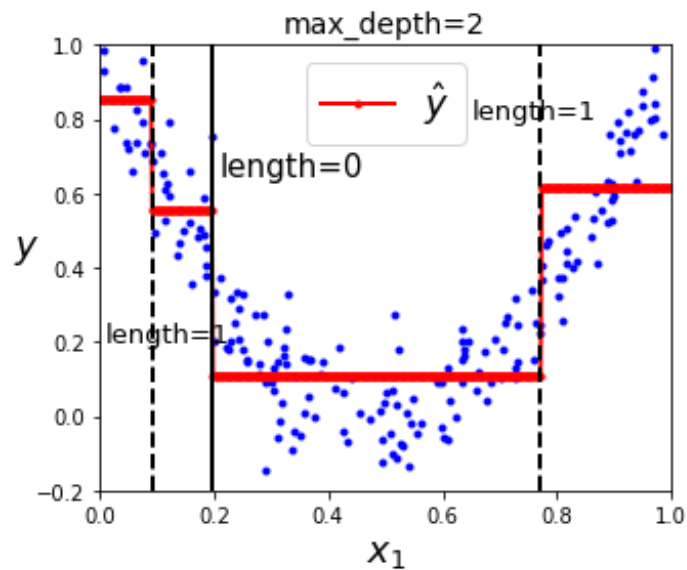
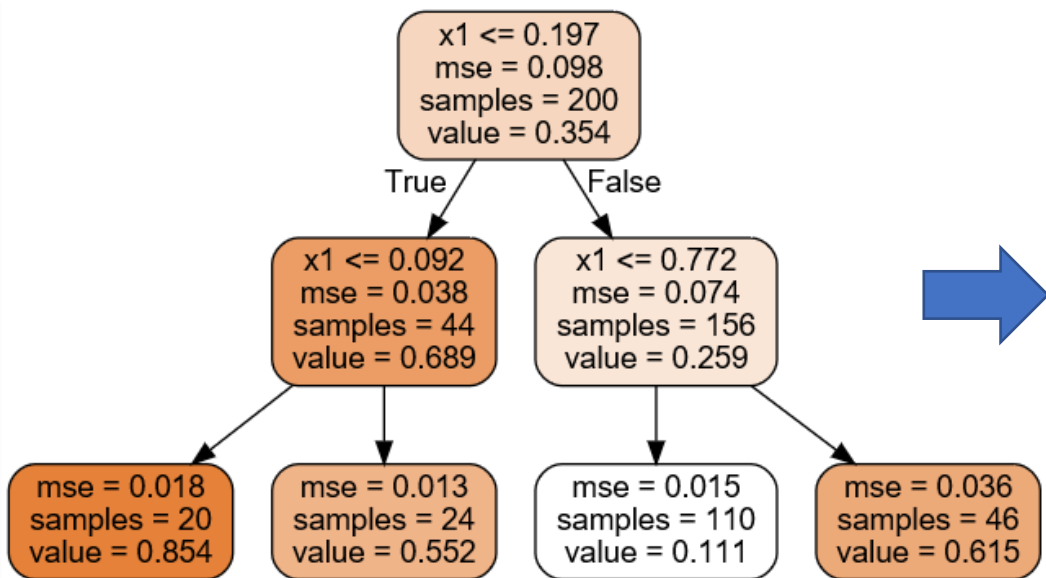
분류형



회귀형

문제 1 : 자전거 대여 수요 예측

회귀의 형태



문제 1 : 자전거 대여 수요 예측

선형회귀처럼 각 특성에 대한 계수값이 나오지 않나?

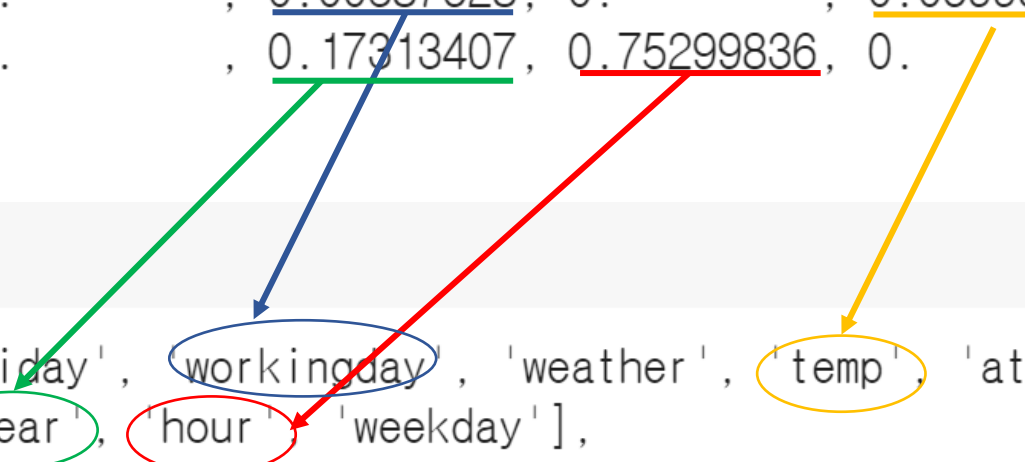
결정트리는 계수 대신 각 특성에 대한 중요도를 출력할 수 있다 (모든 특성의 중요도 합은 1).

```
Tree_model.feature_importances_
```

```
array([0.          , 0.          , 0.00387628, 0.          , 0.06999129,  
       0.          , 0.          , 0.17313407, 0.75299836, 0.          ])
```

```
X_train.columns
```

```
Index(['season', 'holiday', 'workingday', 'weather', 'temp', 'atemp',  
      'humidity', 'year', 'hour', 'weekday'],  
      dtype='object')
```

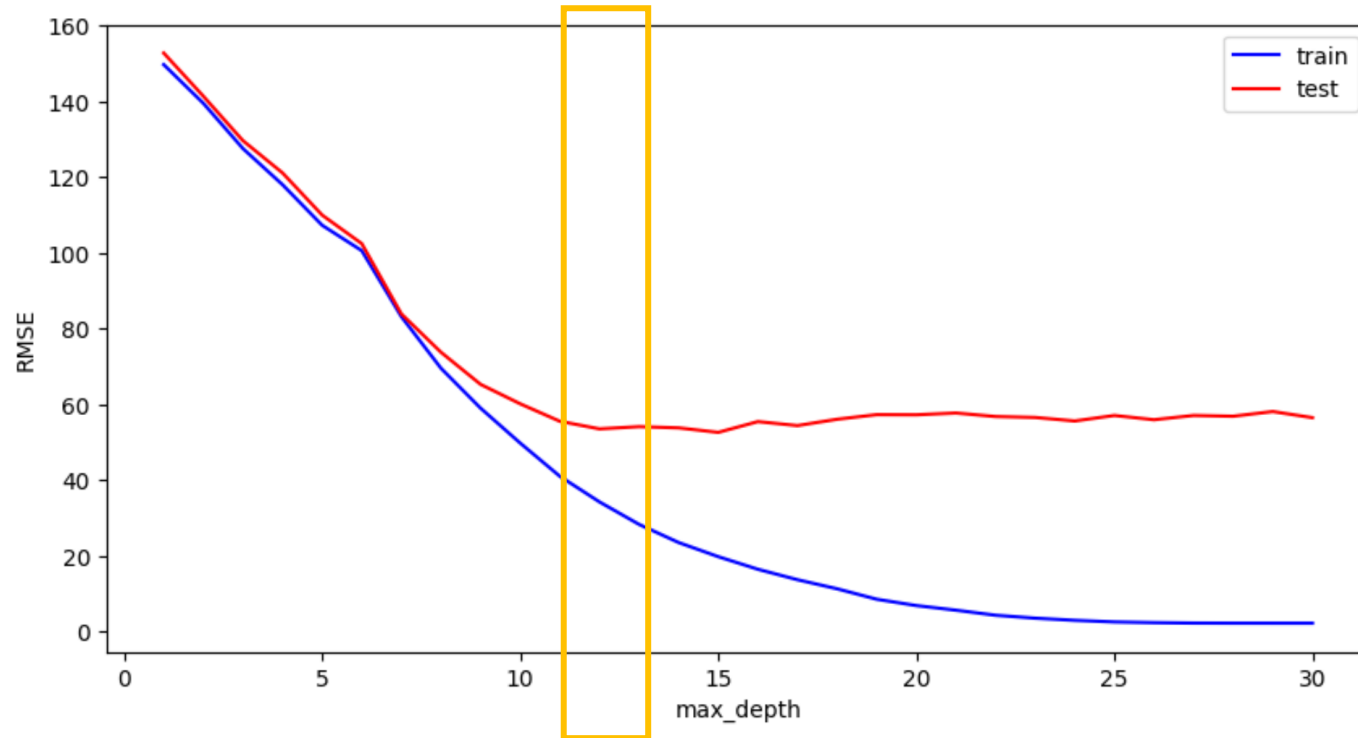


시간(75.3%), 연도(17.3%), 온도(7.0%), 근무일(0.4%)로만 자전거 대여 수요를 예측

문제 1 : 자전거 대여 수요 예측

```
plt.figure(figsize=(10,5))
max_depth_num = np.linspace(1,30,30)
plt.plot(max_depth_num,train_RMSE,'b',max_depth_num,test_RMSE,'r')
plt.xlabel('max_depth')
plt.ylabel('RMSE')
plt.legend(['train','test'])
plt.show()
```

Train은 트리의 깊이가 깊을수록 정확도가 높아지지만
Test는 일정 깊이 이후에 더 이상 정확도가 높아지지 않음.
Test의 정확도가 더 이상 좋아지지 않기 시작하는 지점의 깊이(11~12)가 적당함



문제 1 : 자전거 대여 수요 예측

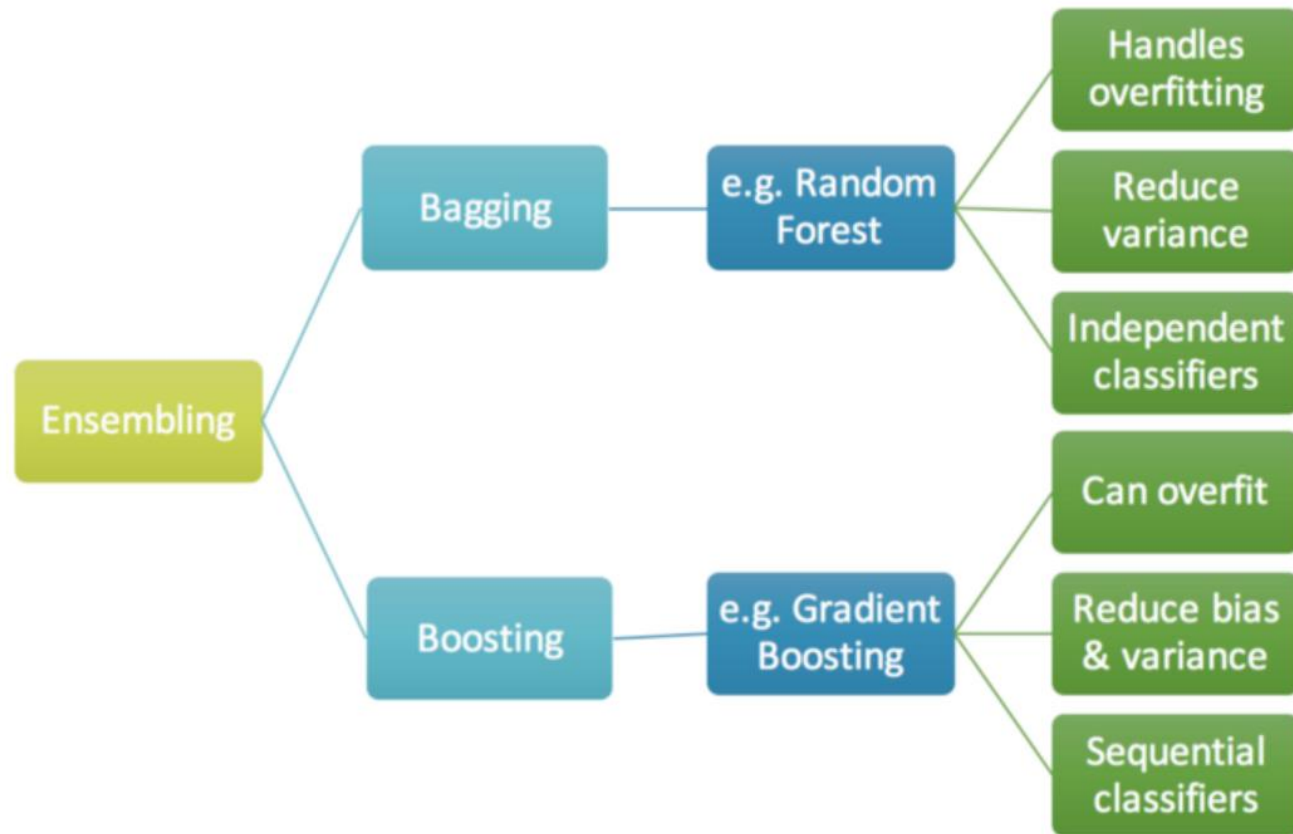
오늘의 학습내용

자전거 대여 수요 예측

1. 다양한 회귀 모델 훈련 및 성능 검증 (계속)
2. 하이퍼파라메타 최적화

문제 1 : 자전거 대여 수요 예측

앙상블(Ensemble) : 여러 개의 모델을 조화롭게 학습시켜 더 정확한 예측값을 도출하는 방법



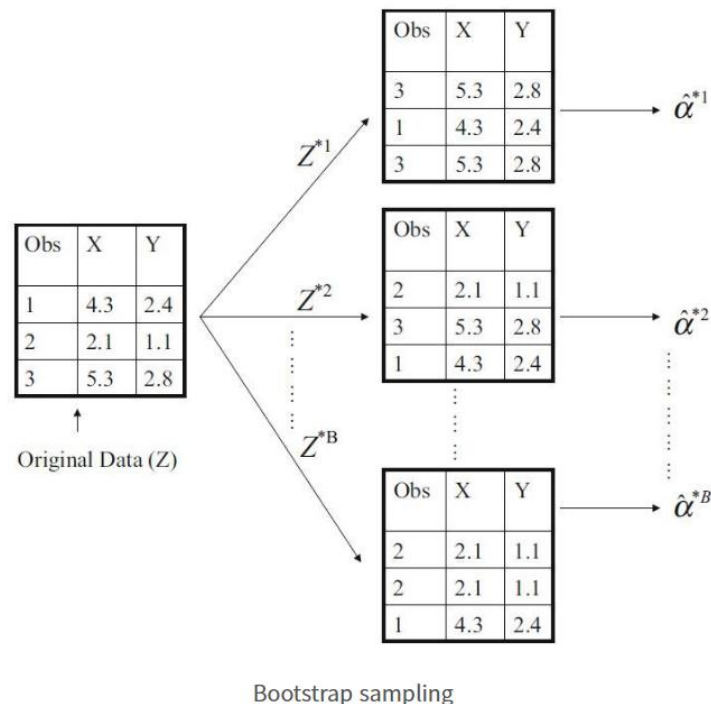
문제 1 : 자전거 대여 수요 예측

Bagging은 Bootstrap Aggregation의 약자

샘플을 여러 번 뽑아(Bootstrap) 각 모델을 학습시켜 결과물을 집계(Aggregation)하는 방법

Bootstrap sampling을 하는 방법은 매우 간단하다. N개의 sample data를 가지고 있을 때 1000개의 bootstrap samples를 만들고자 하면, 복원 추출을 N번 실행하여 새로운 sample data set을 만들고 이 작업을 1000번 반복하면 된다.

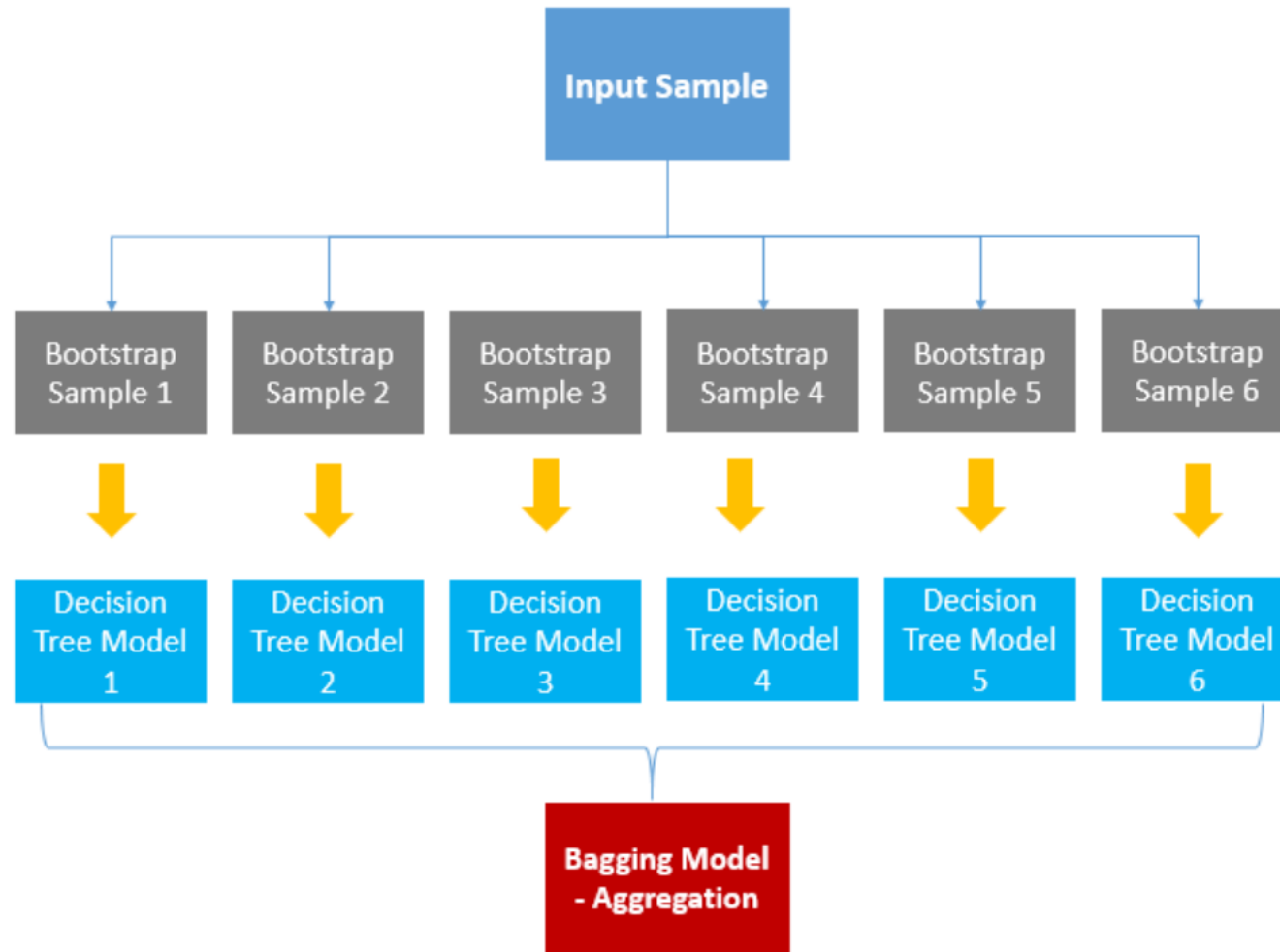
아래 그림은 N=3인 original data로, N번의 복원 추출을 각각 시행하여 총 B개의 bootstrap data set을 만든 것을 볼 수 있다.



문제 1 : 자전거 대여 수요 예측

Bagging은 Bootstrap Aggregation의 약자

샘플을 여러 번 뽑아(Bootstrap) 각 모델을 학습시켜 결과물을 집계(Aggregation)하는 방법



문제 1 : 자전거 대여 수요 예측

▶ Advantage

- 회귀와 분류에서 가장 널리 사용되는 알고리즘
- 뛰어난 성능, 매개변수 튜닝 부담 적음(기본값)
- 데이터 스케일 조정 불필요
- 큰 데이터셋 적용 가능

▶ Disadvantage

- 많은 트리가 생성되므로 자세한 분석이 어렵다
- 트리가 깊어지는 경향이 있음
- 차원이 크고 희소한 데이터에 성능 안 좋음(예: 텍스트 데이터)
- 선형 모델 보다 메모리 사용량이 많다
- 훈련과 예측이 느림

문제 1 : 자전거 대여 수요 예측

랜덤포레스트를 적용해보자.

```
y_train = np.array(y_train).reshape(-1)
y_test = np.array(y_test).reshape(-1)
```

← 랜덤포레스트의 경우 output이 1개일 경우
1차원 벡터를 입력하므로 1차원 벡터로 변경

```
from sklearn.ensemble import RandomForestRegressor
```

```
RF_model = RandomForestRegressor(n_estimators = 10)
RF_model.fit(X_train, y_train)
```

← 랜덤포레스트 모델을 정의하자
(몇 개를 샘플링하여 집계할지 결정하자).
← train 특성과 타깃을 랜덤포레스트 학습.

```
RF_pred_train = RF_model.predict(X_train)
```

← Predict함수로 train 결과 확인

```
RF_pred_test = RF_model.predict(X_test)
```

← Predict함수로 test 결과 확인

문제 1 : 자전거 대여 수요 예측

`RMSE(y_train, RF_pred_train)` ← 모델 평가 (train)

19.222317926285648

`RMSE(y_test, RF_pred_test)` ← 모델 평가 (test)

44.7899113426495

결정트리보다도 **RMSE가 낮아졌다.**

랜덤포레스트가 자전거 대여 수요 예측에서 **결정트리(최대깊이=11)보다 더 정확도가 좋은 모델이다.**

랜덤포레스트는 Bootstrap의 랜덤 샘플링 과정이 있기 때문에 결과가 항상 조금씩 달라짐

문제 1 : 자전거 대여 수요 예측

Decision Tree 기반 모델이기 때문에 특성중요도를 계산할 수 있다.

```
RF_model.feature_importances_
```

```
array([0.05101952, 0.0031291 , 0.04339639, 0.01815571, 0.07247466,  
       0.03395196, 0.03436899, 0.08789539, 0.61087164, 0.04473665])
```

```
X_train.columns
```

```
Index(['season', 'holiday', 'workingday', 'weather', 'temp', 'atemp',  
       'humidity', 'year', 'hour', 'weekday'],  
      dtype='object')
```

문제 1 : 자전거 대여 수요 예측

얼마나 정확해졌을까? 그림으로 비교해보자.

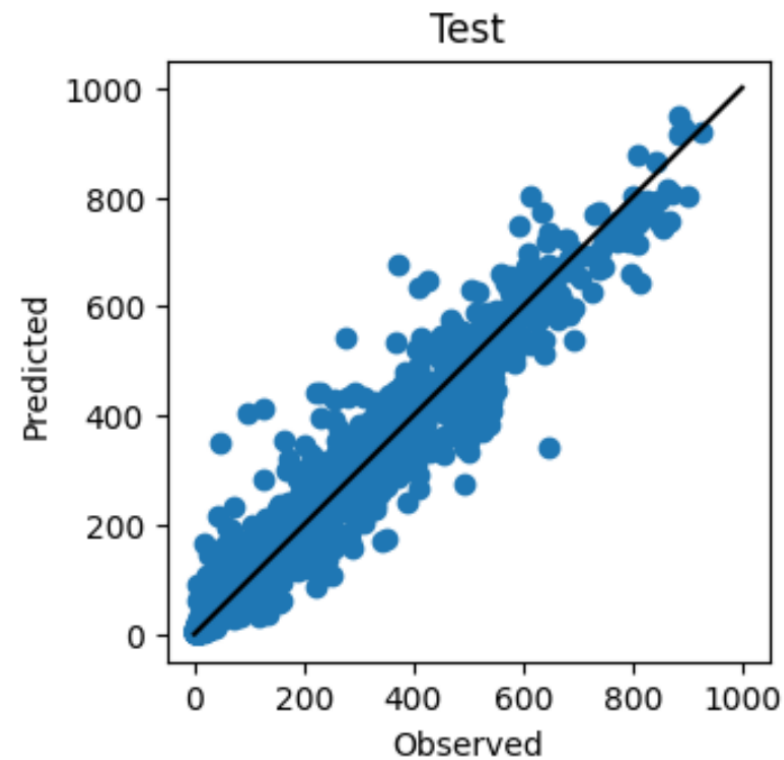
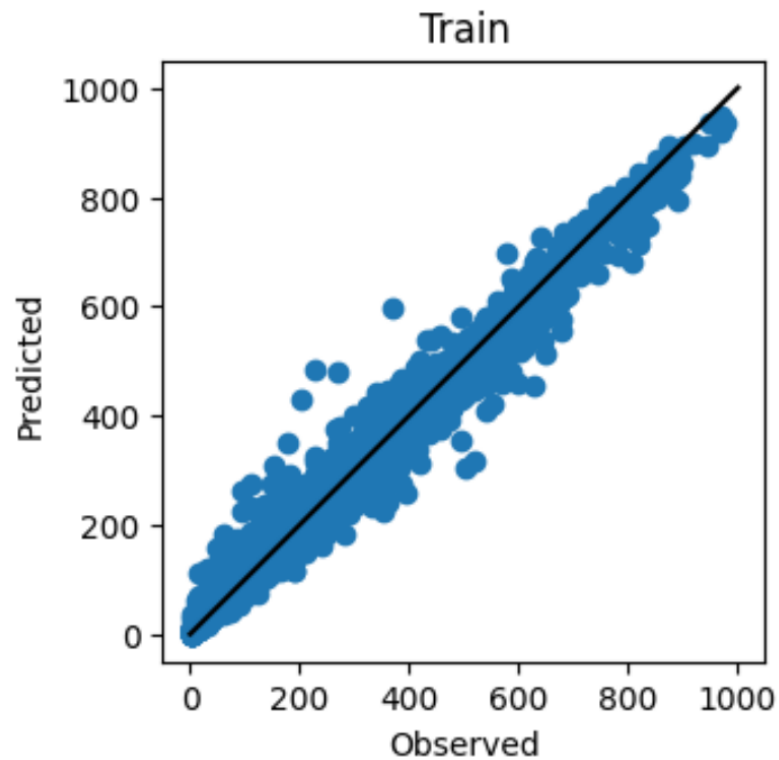
```
fig, axs = plt.subplots(1,2)
plt.tight_layout(h_pad=3, w_pad=3)
fig.set_size_inches(7,3)

x = np.linspace(0,1000)
y=x
axs[0].plot(y_train,RF_pred_train,'o',x,y,'k')
axs[0].set_title('Train')

axs[0].set_xlabel('Observed')
axs[0].set_ylabel('Predicted')

axs[1].plot(y_test,RF_pred_test,'o',x,y,'k')
axs[1].set_title('Test')

axs[1].set_xlabel('Observed')
axs[1].set_ylabel('Predicted')
plt.show()
```



문제 1 : 자전거 대여 수요 예측

추가적으로 고려해야할 문제

1. 과대적합 문제
2. 모델의 최적의 하이퍼파라메타를 찾는 문제

문제 1 : 자전거 대여 수요 예측

지금까지처럼 모델을 훈련만 하고, 성능 검증없이 바로 테스트 한다면?

1. 모델이 과대적합 될 수 있다.

- train 데이터에 대한 성능에 비해 test 데이터에 대한 성능이 많이 낮게 나옴

2. 모델 성능을 미리 확인하기 어렵다.

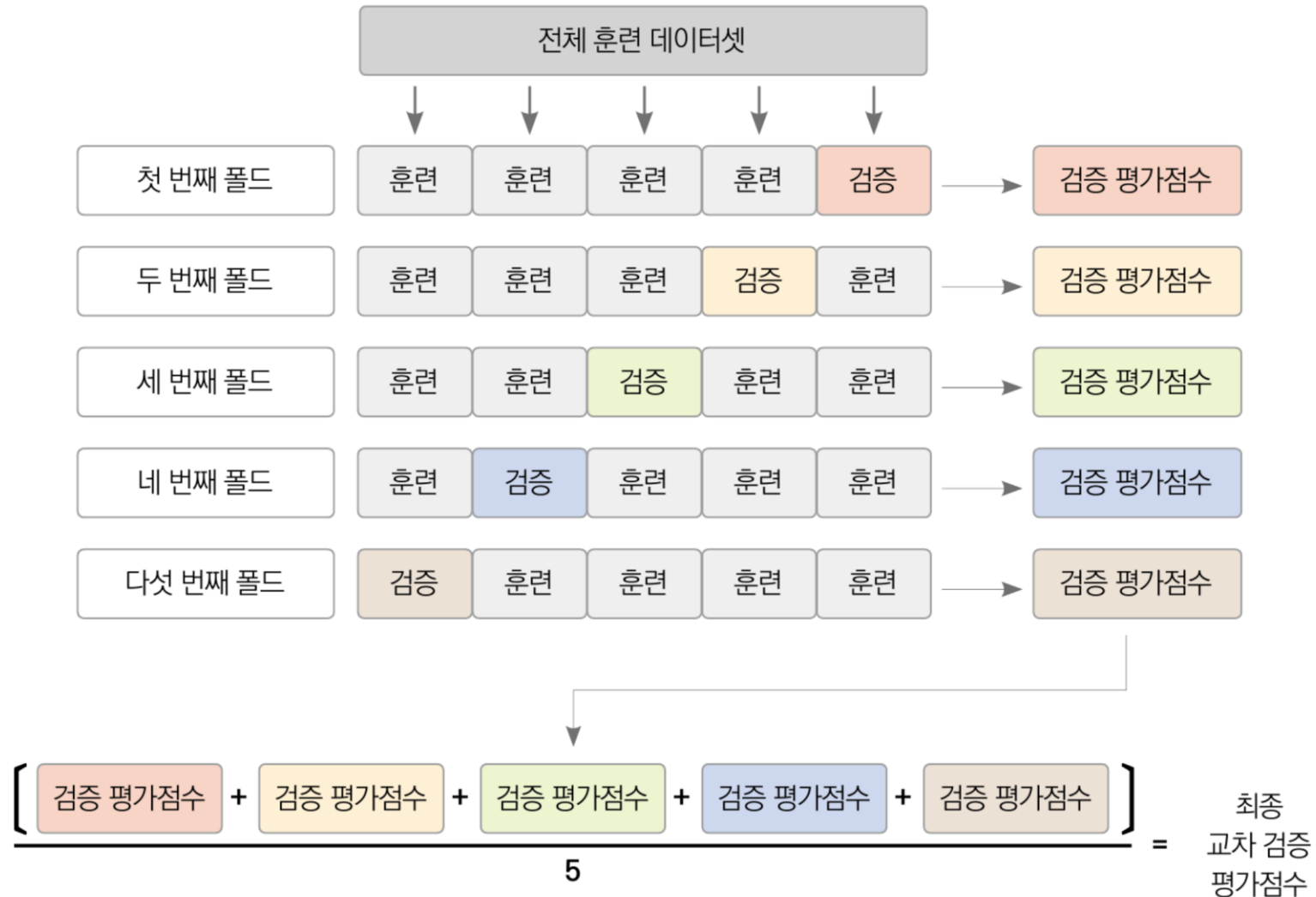
- 만약 예측을 한다고 했을때, 모든 데이터를 사용하여 훈련에 사용한다면 예측에 대한 성능을 확인하기 어려움

이를 개선하기 위한 방법은? **교차 검증!!**

문제 1 : 자전거 대여 수요 예측

K 폴드 교차 검증 (K-Fold Cross Validation)

▼ K 폴드 교차 검증(K = 5일 때)



문제 1 : 자전거 대여 수요 예측

K 폴드 라이브러리 추가

```
from sklearn.model_selection import KFold
```

K 폴드 교차 검증을 사용할 때, 데이터가 어떻게 되는지 살펴보자.

```
data = np.array(range(0,10))
```

← 교차 검정은 np.array 형식으로 입력되어야 한다.
0~9까지를 data로 입력하자.

```
data
```

```
array([0, 1, 2, 3, 4, 5, 6, 7, 8, 9])
```

```
exam_folds_1 = KFold(n_splits=5, shuffle=False)
```

← K 폴드를 정의하자.

↑ ↑
폴드 수 (나누는 수) 셔플 유무

문제 1 : 자전거 대여 수요 예측

섞었을때 데이터의 위치를 정의한다.

데이터를 앞서 정의한 폴드 수로 섞는다.

```
for train_index, valid_index in exam_folds_1.split(data):  
    print('훈련데이터 : ',data[train_index], ', 검증데이터 : ',data[valid_index])
```

훈련데이터 : [2 3 4 5 6 7 8 9] , 검증데이터 : [0 1]

훈련데이터 : [0 1 4 5 6 7 8 9] , 검증데이터 : [2 3]

훈련데이터 : [0 1 2 3 6 7 8 9] , 검증데이터 : [4 5]

훈련데이터 : [0 1 2 3 4 5 8 9] , 검증데이터 : [6 7]

훈련데이터 : [0 1 2 3 4 5 6 7] , 검증데이터 : [8 9]

셔플을 하지 않았기 때문에 검증데이터가 차례대로 선택된다.

문제 1 : 자전거 대여 수요 예측

셔플을 한다면?



```
exam_folds_2 = KFold(n_splits=5, shuffle=True)
```

```
for train_index, valid_index in exam_folds_2.split(data):  
    print('훈련데이터 : ', data[train_index], ', 검증데이터 : ', data[valid_index])
```

```
훈련데이터 : [0 1 2 3 5 6 7 9] , 검증데이터 : [4 8]  
훈련데이터 : [0 1 3 4 5 6 8 9] , 검증데이터 : [2 7]  
훈련데이터 : [0 1 2 4 6 7 8 9] , 검증데이터 : [3 5]  
훈련데이터 : [0 2 3 4 5 6 7 8] , 검증데이터 : [1 9]  
훈련데이터 : [1 2 3 4 5 7 8 9] , 검증데이터 : [0 6]
```

셔플이 되어 검증데이터가 무작위로 차례로 선택된 것을 볼 수 있다.

문제 1 : 자전거 대여 수요 예측

셔플은 하는데 셔플의 결과를 고정하고 싶다면?



```
exam_folds_3 = KFold(random_state = 42, n_splits = 5, shuffle = True)
```

```
for train_index, valid_index in exam_folds_3.split(data):  
    print('훈련데이터 : ', data[train_index], ', 검증데이터 : ', data[valid_index])
```

```
훈련데이터 : [0 2 3 4 5 6 7 9] , 검증데이터 : [1 8]  
훈련데이터 : [1 2 3 4 6 7 8 9] , 검증데이터 : [0 5]  
훈련데이터 : [0 1 3 4 5 6 8 9] , 검증데이터 : [2 7]  
훈련데이터 : [0 1 2 3 5 6 7 8] , 검증데이터 : [4 9]  
훈련데이터 : [0 1 2 4 5 7 8 9] , 검증데이터 : [3 6]
```

다시 셔플을 실행해도 동일한 훈련데이터와 검증데이터가 도출되는 것을 볼 수 있다.

문제 1 : 자전거 대여 수요 예측

자전거 대여 데이터에 대해 적용해보자.

```
 folds = KFold(random_state = 42, n_splits = 5, shuffle = True)
```

← 5 폴드 교차 검증 정의

```
 X = np.array(X_train)
 y = np.array(y_train)
```

교차 검증은 앞서 말한대로 np.array타입이어야 하기 때문에
각각의 train 데이터를 np.array타입으로 변경

문제 1 : 자전거 대여 수요 예측

RMSE_valid = [] ← 검증 결과를 저장한 공간 마련

for train_index, valid_index in folds.split(X):

X_train_2, X_valid = X[train_index], X[valid_index] ← X, y에 대한 train, validation 데이터
y_train_2, y_valid = y[train_index], y[valid_index]

RF_valid_model = RandomForestRegressor(random_state = 42, n_estimators = 10)

RF_valid_model.fit(X_train_2, y_train_2) ← train 데이터에 대한 랜덤포레스트 학습

y_valid_pred = RF_valid_model.predict(X_valid)

RMSE_valid.append(RMSE(y_valid, y_valid_pred)) ← validation 데이터에 대한 예측 및 평가

문제 1 : 자전거 대여 수요 예측

```
RMSE_valid
```

5개의 교차 검증 결과

```
[47.64103020524864,  
50.00930155736101,  
44.65274852336477,  
45.07234114801049,  
48.91942859356319]
```

```
np.mean(RMSE_valid)
```

5개의 교차 검증 결과 평균

```
47.258970005509624
```

```
RMSE(y_test, RF_pred_test)
```

```
44.7899113426495
```



과대 적합된 RMSE가 아닌 실제 test 결과에 대한 RMSE와 유사한 RMSE를 얻을 수 있다.

문제 1 : 자전거 대여 수요 예측

각 모델별로 정확도가 가장 높은 변수를 찾는 방법은 없을까?

문제 1 : 자전거 대여 수요 예측

하이퍼파라미터 (Hyperparameter) 최적화

1. 그리드 서치 : 주어진 하이퍼파라미터를 모두 순회하며 가장 좋은 성능을 내는 값을 찾는 방법
2. 랜덤서치 : 하이퍼파라미터를 무작위로 탐색해 가장 좋은 성능을 찾는 방법
3. 베이지안 최적화 : 사전 정보를 바탕으로 최적 하이퍼파라미터 값을 확률적으로 추정하며 탐색하는 방법

문제 1 : 자전거 대여 수요 예측

그리드 서치 (Grid search)

```
from sklearn.model_selection import GridSearchCV ← 그리드 서치 라이브러리
from sklearn import metrics ← 평가지표 판단을 위한 라이브러리
```

```
RF_model = RandomForestRegressor()
```

평가지표 함수

```
RMSE_scorer = metrics.make_scorer(RMSE, greater_is_better = False)
```

평가지표가 큰것이 좋나?

우리는 평가지표가 오차를 나타내므로

False!!

평가함수

```
RF_params = {'random_state' : [42], 'n_estimators' : [5, 10, 15]}
```

서치할 변수와 값들

문제 1 : 자전거 대여 수요 예측

```
GS_RF_model = GridSearchCV(estimator = RF_model, ← 학습 모델  
                           param_grid = RF_params, ← 서치할 파라미터  
                           scoring = RMSE_scorer, ← 평가지표  
                           cv = 5) ← 교차 검증
```

```
GS_RF_model.fit(X_train,y_train) ← 그리드 서치로 모델 학습
```

랜덤포레스트의 그리드 서치는 학습하는데 상당한 시간이 걸린다.

문제 1 : 자전거 대여 수요 예측

`GS_RF_model.best_params_` ← 서치한 파라미터중 가장 성능이 좋은 파라미터

```
{'n_estimators': 15, 'random_state': 42}
```

`GS_RF_model.best_estimator_.predict(X_train)` ← train data의 학습 결과

```
array([411.73333333, 409.6          , 21.26666667, ..., 286.2
        678.2          , 72.6          ])
```

`GS_pred_train = GS_RF_model.best_estimator_.predict(X_train)`
`GS_pred_test = GS_RF_model.best_estimator_.predict(X_test)` ← 예측 결과 저장

`RMSE(y_train, GS_pred_train)` ← RMSE로 모델 평가 (train)

```
18.164863512012037
```

`RMSE(y_test, GS_pred_test)` ← RMSE로 모델 평가 (test)

```
42.11580277640049
```

문제 1 : 자전거 대여 수요 예측

서치할 파라미터의 범위를 더 넓혀보자.

```
list(range(5,50,5))
```

```
[5, 10, 15, 20, 25, 30, 35, 40, 45]
```

예제

```
RF_params = {'random_state' : [42], 'n_estimators' : list(range(5,50,5))}
```

문제 1 : 자전거 대여 수요 예측

서치할 새로운 파라미터를 추가해보자.

| 파라미터 명 | 설명 |
|--------------------------|---|
| n_estimators | <ul style="list-style-type: none">- 결정트리의 갯수를 지정- Default = 10- 무작정 트리 갯수를 늘리면 성능 좋아지는 것 대비 시간이 걸릴 수 있음 |
| min_samples_split | <ul style="list-style-type: none">- 노드를 분할하기 위한 최소한의 샘플 데이터수→ 과적합을 제어하는데 사용- Default = 2 → 작게 설정할 수록 분할 노드가 많아져 과적합 가능성 증가 |
| min_samples_leaf | <ul style="list-style-type: none">- 리프노드가 되기 위해 필요한 최소한의 샘플 데이터수- min_samples_split과 함께 과적합 제어 용도- 불균형 데이터의 경우 특정 클래스의 데이터가 극도로 작을 수 있으므로 작게 설정 필요 |
| max_features | <ul style="list-style-type: none">- 최적의 분할을 위해 고려할 최대 feature 개수- Default = 'auto' (결정트리에서는 default가 none이었음)- int형으로 지정 → 피쳐 갯수 / float형으로 지정 → 비중- sqrt 또는 auto : 전체 피쳐 중 $\sqrt{(\text{피쳐개수})}$ 만큼 선정- log : 전체 피쳐 중 $\log_2(\text{전체 피쳐 개수})$ 만큼 선정 |
| max_depth | <ul style="list-style-type: none">- 트리의 최대 깊이- default = None→ 완벽하게 클래스 값이 결정될 때 까지 분할또는 데이터 개수가 min_samples_split보다 작아질 때까지 분할- 깊이가 깊어지면 과적합될 수 있으므로 적절히 제어 필요 |
| max_leaf_nodes | 리프노드의 최대 개수 |

예제

```
RF_params = {'random_state' : [42], 'n_estimators' : list(range(5,50,5)), 'min_samples_split' : list(range(2,10,2))}
```


문제 1 : 자전거 대여 수요 예측

- 각자 랜덤포레스트 모델의 최적의 하이퍼파라미터를 Grid search로 찾아보자.
- 비슷하게 결정트리 모델의 최적의 하이퍼파라미터도 Grid search로 찾아보자.
- 옆에 친구와 본인이 찾은 파라미터와 RMSE를 비교해보자.

문제 1 : 자전거 대여 수요 예측

*참고 : 결정트리 하이퍼파라미터

- max_depth : 최대 깊이
- max_features : 각 노드에서 분할에 사용할 특성의 최대수
- max_leaf_nodes : 리프 노드의 최대수
- min_samples_leaf : 리프 노드가 가지고 있어야 할 최소 샘플 수
- min_samples_split : 분할되기 위해 노드가 가져야 하는 최소 샘플의 수

Thank you

