상류 BOD 측정값을 통한 하류의 BOD 수치 예측

2020108246 컴퓨터공학과 고해인

목차

BOD와 예측 필요성

BOD에 대해 / BOD 예측 필요성 / 관련 연구

BOD가 뭐지?

생화학적 산소 요구량
(Biochemical Oxygen Demand)은

물 속에 있는 호기성 미생물이
유기물을 분해하는 데 필요한
산소 소모량을 말한다.

"밥은 잘 먹고 사니?"

어떻게 측정하지?

5일이나 걸린다!

물 속의 산소량을 측정하고
5일 뒤에 얼마나 줄었는지
측정하는 방법
(BOD5)

이걸 왜 예측해야해?

내려가서 물 한 컵 더 뜨면 되잖아?

상류 데이터를 통한 예측값과
실측값이 크게 다른 것을 발견하면
자연 환경의 변화나 인위적인 오염을
예상할 수 있는 단서가 된다

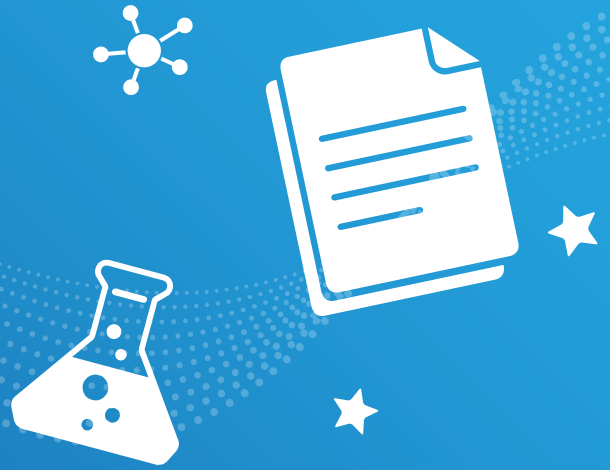
>> 수질 모형 구축

수질 모형을 위한 연구들

1차원 수질 예측 모형의 검보정 자동화 시스템 개발 및
낙동강에서의 적용 (손아롱 외)

신경망 모형을 이용한 달천의 수질예측 시스템 구축 (이원호 외)

인공신경망기법을 이용한 하천수질인자의 예측모델링
- BOD와 DO를 중심으로 (조현경)



머신러닝 모델 비교와 예측

요약 / 분석 / 시각화 / 전처리 / 학습과 테스트 / 예측

과제 요약

- 데이터 분석과 전처리
- 5가지 머신러닝 모델 성능 비교
- 데이터 학습과 테스트

데이터를 분석하고 시각화하는 라이브러리의 사용법과

머신러닝의 성능을 평가하는 방법에 대해 탐구한다

데이터 분석

Id : 각 측정값을 구분하는 ID

target : 예측 목표 지점의 BOD

1~7 : 예측 목표 지점에서 강을 따라 올라가며 매긴 번호
제일 가까운 곳은 1,
제일 먼 곳은 7

	Id	target	1	2	3	4	5	6	7
0	0	5.85	4.80	5.85	NaN	NaN	NaN	NaN	NaN
1	3	4.28	5.88	6.84	NaN	NaN	NaN	NaN	NaN
2	4	3.97	3.20	2.70	NaN	NaN	NaN	NaN	NaN
3	5	5.95	7.70	7.06	NaN	NaN	NaN	NaN	NaN
4	6	4.70	5.50	5.30	NaN	NaN	NaN	NaN	NaN
5	7	4.36	4.91	4.48	NaN	NaN	NaN	NaN	NaN
6	8	3.74	3.71	3.76	NaN				
7	9	7.70	3.75	3.60	NaN				
8	10	3.34	2.13	5.40	NaN				
9	11	3.96	2.24	4.22	NaN				

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 147 entries, 0 to 146
Data columns (total 9 columns):
 #   Column  Non-Null Count  Dtype
---  ------  --------------  -----
 0   Id      147 non-null    int64
 1   target  147 non-null    float64
 2   1       145 non-null    float64
 3   2       145 non-null    float64
 4   3       32 non-null     float64
 5   4       31 non-null     float64
 6   5       33 non-null     float64
 7   6       37 non-null     float64
 8   7       37 non-null     float64
dtypes: float64(8), int64(1)
memory usage: 10.5 KB
```

데이터 분석

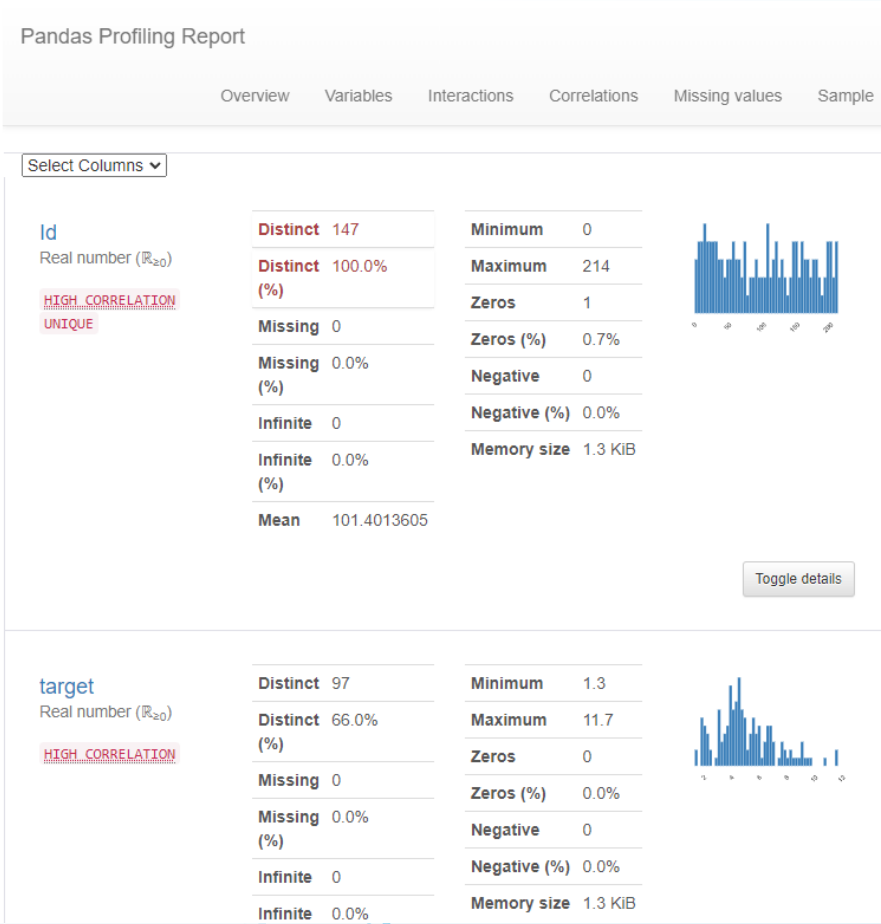
데이터 읽기, 그래프 표시,
머신러닝 모델들, 전처리 모듈..

수 많은 모듈 중에
pandas_profilling
모듈을 이용

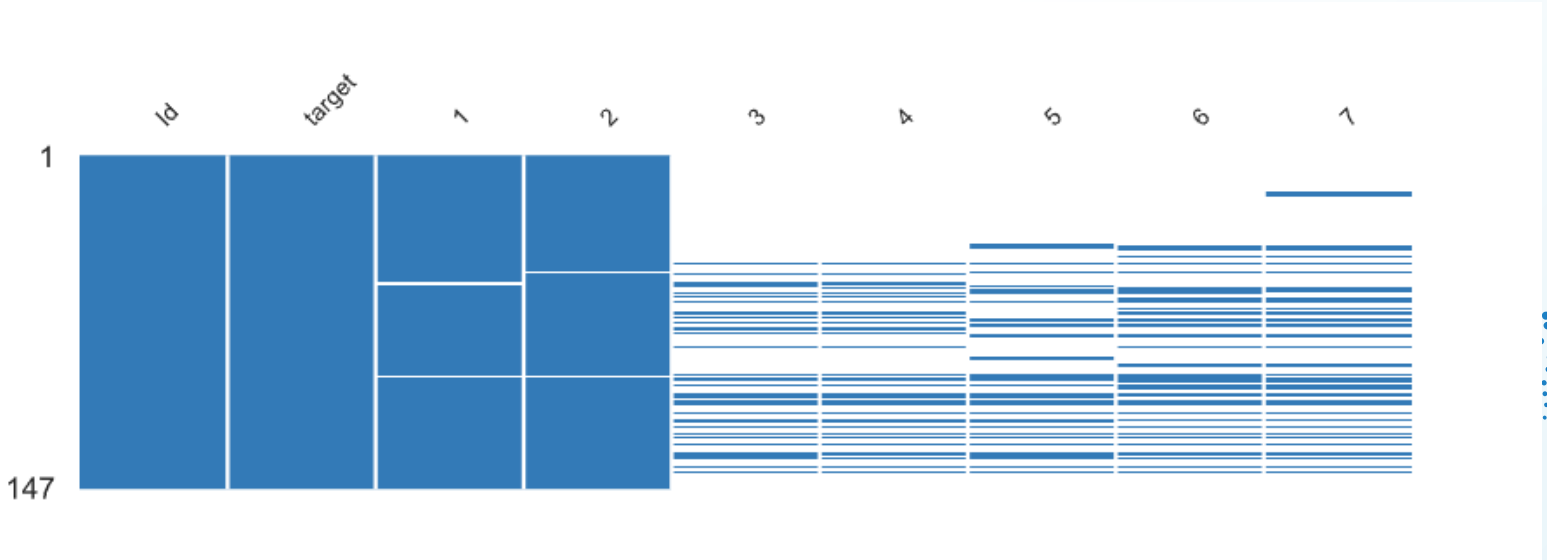
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
%matplotlib inline

# preprocessing
from sklearn.preprocessing import LabelEncoder, StandardScaler
from sklearn.model_selection import train_test_split, cross_val_score, GridSearchCV, StratifiedKFold
import pandas_profiling as pp

# models
from sklearn.linear_model import LinearRegression, SGDRegressor, RidgeCV
from sklearn.svm import SVR, LinearSVR
from sklearn.ensemble import RandomForestRegressor, GradientBoostingRegressor, ExtraTreesRegressor
from sklearn.ensemble import BaggingRegressor, AdaBoostRegressor, VotingRegressor
from sklearn.neural_network import MLPRegressor
from sklearn.tree import DecisionTreeRegressor
from sklearn.neighbors import RadiusNeighborsRegressor
import sklearn.model_selection
from sklearn.model_selection import cross_val_predict as cvp
from sklearn import metrics
from sklearn.metrics import r2_score, mean_absolute_error, mean_squared_error
import xgboost as xgb
import lightgbm as lgb

# model tuning
from hyperopt import STATUS_OK, Trials, fmin, hp, tpe, space_eval

import warnings
warnings.filterwarnings("ignore")
```

데이터 시각화

각 항목의 대표값, 분포도,
상관도, 히트맵 등을 정리해서
한 눈에 볼 수 있는 모듈

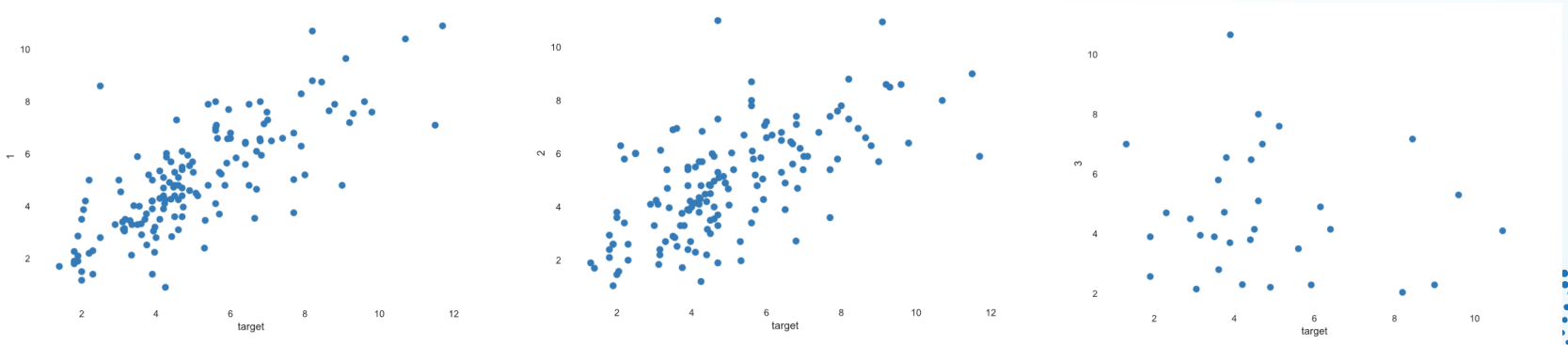


데이터 시각화 – Nullity Matrix



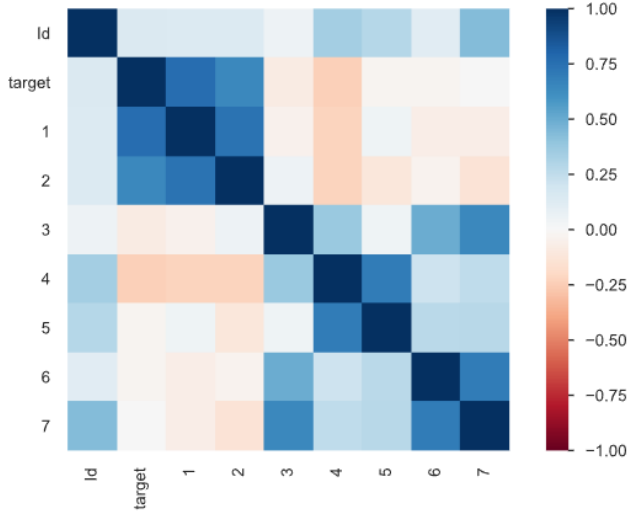
각 항목에서 빈 값들을 시각적으로 표현하는 매트릭스
3~7은 누락된 데이터가 너무 많아 사용하기 어렵다

데이터 시각화 – Plot Chart



target과 1, 2 에 대한 연관성은 선형에 가깝게 나오지만
3 이후 부터는 연관성을 찾기가 어렵다

데이터 시각화 – Heatmap



target과 1, 2 항목에 대한
연관성이 높게 나타난다

데이터 전처리

측정값 구분자와
Null값이 많은 열을 지우고
Null값을 가지고 있는 행 또한
제거하여 학습용 데이터 준비

```
In [7]: train0 = train0.drop(['Id','3','4','5','6','7'], axis = 1)
        train0 = train0.dropna()
        train0.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 144 entries, 0 to 146
Data columns (total 3 columns):
 #   Column  Non-Null Count  Dtype
---  ------  --------------  -----
 0   target  144 non-null    float64
 1   1       144 non-null    float64
 2   2       144 non-null    float64
dtypes: float64(3)
memory usage: 4.5 KB
```

```
In [8]: train0.head(3)
```

Out [8]:

	target	1	2
0	5.85	4.80	5.85
1	4.28	5.88	6.84
2	3.97	3.20	2.70

데이터 전처리/분할

```
In [9]: target_name = 'target'

In [10]: # For boosting model
         train0b = train0
         train_target0b = train0b[target_name]
         train0b = train0b.drop([target_name], axis=1)
         # Synthesis valid as test for selection models
         trainb, testb, targetb, target_testb = train_test_split(train0b, train_target0b, test_size=valid_part, random_state=0)

In [11]: train_target0 = train0[target_name]
         train0 = train0.drop([target_name], axis=1)

In [12]: #For models from Sklearn
         scaler = StandardScaler()
         train0 = pd.DataFrame(scaler.fit_transform(train0), columns = train0.columns)

In [15]: # Synthesis valid as test for selection models
         train, test, target, target_test = train_test_split(train0, train_target0, test_size=valid_part, random_state=0)
```

부스팅 모델용으로 분리, 왜?

```
In [2]: valid_part = 0.3
```

일반 모델용만 정규화, 왜?

데이터의 30%는 테스트용으로 분리

일반 모델에 사용할 데이터는 평균 0, 분산 1로 정규화

데이터 전처리/분할

```
In [16]: train.head(3)
```

Out[16]:

	1	2
95	-0.804207	1.018669
92	1.284809	0.238010
50	1.064655	0.640931

```
In [17]: test.head(3)
```

Out[17]:

	1	2
7	-0.598730	-0.668561
89	-1.993038	-1.877323
97	-0.466638	-0.482211

```
In [18]: train.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 100 entries, 95 to 47
Data columns (total 2 columns):
 #   Column  Non-Null Count  Dtype
---  ------  --------------  -----
 0   1       100 non-null    float64
 1   2       100 non-null    float64
dtypes: float64(2)
memory usage: 2.3 KB
```

```
In [19]: test.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 44 entries, 7 to 112
Data columns (total 2 columns):
 #   Column  Non-Null Count  Dtype
---  ------  --------------  -----
 0   1       44 non-null     float64
 1   2       44 non-null     float64
dtypes: float64(2)
memory usage: 1.0 KB
```

학습용, 테스트용 데이터가 100개, 44개씩
무작위로 선정되었다

학습과 테스트 - 평가지표

R2 Score	Relative Error	RMSE
우리가 그동안 사용한 score 함수로 나오는 점수	오차를 참값으로 나눠 오차의 상대적 비율을 나타내는 점수	편차를 제곱하고 평균을 낸 후 제곱근을 씌운 값
높을수록 좋다!	낮을수록 좋다!	낮을수록 좋다!

학습과 테스트 - 평가지표

In [21]:
```python
def acc_d(y_meas, y_pred):
    # Relative error between predicted y_pred and measured y_meas values
    return mean_absolute_error(y_meas, y_pred)*len(y_meas)/sum(abs(y_meas))

def acc_rmse(y_meas, y_pred):
    # RMSE between predicted y_pred and measured y_meas values
    return (mean_squared_error(y_meas, y_pred))**0.5
```

acc_d : Relative Error를 계산하는 함수

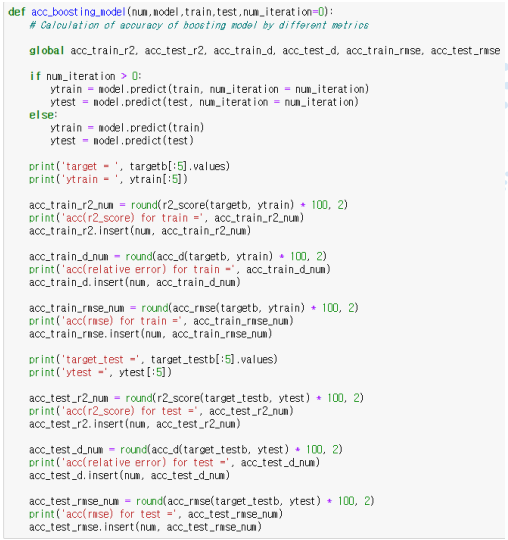
acc_rmse : RMSE 점수를 계산하는 함수

acc_boosting_model

부스팅 모델에 점수를 매기는 함수

acc_model

일반 모델에 점수를 매기는 함수



```python
def acc_boosting_model(num,model,train,test,num_iteration=0):
    # Calculation of accuracy of boosting model by different metrics

    global acc_train_r2, acc_test_r2, acc_train_d, acc_test_d, acc_train_rmse, acc_test_rmse

    if num_iteration > 0:
        ytrain = model.predict(train, num_iteration = num_iteration)
        ytest = model.predict(test, num_iteration = num_iteration)
    else:
        ytrain = model.predict(train)
        ytest = model.predict(test)

    print('target = ', targetb[:5].values)
    print('ytrain = ', ytrain[:5])

    acc_train_r2_num = round(r2_score(targetb, ytrain) * 100, 2)
    print('acc(r2_score) for train =', acc_train_r2_num)
    acc_train_r2.insert(num, acc_train_r2_num)

    acc_train_d_num = round(acc_d(targetb, ytrain) * 100, 2)
    print('acc(relative error) for train =', acc_train_d_num)
    acc_train_d.insert(num, acc_train_d_num)

    acc_train_rmse_num = round(acc_rmse(targetb, ytrain) * 100, 2)
    print('acc(rmse) for train =', acc_train_rmse_num)
    acc_train_rmse.insert(num, acc_train_rmse_num)

    print('target_test =', target_testb[:5].values)
    print('ytest =', ytest[:5])

    acc_test_r2_num = round(r2_score(target_testb, ytest) * 100, 2)
    print('acc(r2_score) for test =', acc_test_r2_num)
    acc_test_r2.insert(num, acc_test_r2_num)

    acc_test_d_num = round(acc_d(target_testb, ytest) * 100, 2)
    print('acc(relative error) for test =', acc_test_d_num)
    acc_test_d.insert(num, acc_test_d_num)

    acc_test_rmse_num = round(acc_rmse(target_testb, ytest) * 100, 2)
    print('acc(rmse) for test =', acc_test_rmse_num)
    acc_test_rmse.insert(num, acc_test_rmse_num)
```

```python
def acc_model(num,model,train,test):
    # Calculation of accuracy of model 일반 모델 Sklearn by different metrics

    global acc_train_r2, acc_test_r2, acc_train_d, acc_test_d, acc_train_rmse, acc_test_rmse

    ytrain = model.predict(train)
    ytest = model.predict(test)

    print('target = ', target[:5].values)
    print('ytrain = ', ytrain[:5])

    acc_train_r2_num = round(r2_score(target, ytrain) * 100, 2)
    print('acc(r2_score) for train =', acc_train_r2_num)
    acc_train_r2.insert(num, acc_train_r2_num)

    acc_train_d_num = round(acc_d(target, ytrain) * 100, 2)
    print('acc(relative error) for train =', acc_train_d_num)
    acc_train_d.insert(num, acc_train_d_num)

    acc_train_rmse_num = round(acc_rmse(target, ytrain) * 100, 2)
    print('acc(rmse) for train =', acc_train_rmse_num)
    acc_train_rmse.insert(num, acc_train_rmse_num)

    print('target_test =', target_test[:5].values)
    print('ytest =', ytest[:5])

    acc_test_r2_num = round(r2_score(target_test, ytest) * 100, 2)
    print('acc(r2_score) for test =', acc_test_r2_num)
    acc_test_r2.insert(num, acc_test_r2_num)

    acc_test_d_num = round(acc_d(target_test, ytest) * 100, 2)
    print('acc(relative error) for test =', acc_test_d_num)
    acc_test_d.insert(num, acc_test_d_num)

    acc_test_rmse_num = round(acc_rmse(target_test, ytest) * 100, 2)
    print('acc(rmse) for test =', acc_test_rmse_num)
    acc_test_rmse.insert(num, acc_test_rmse_num)
```

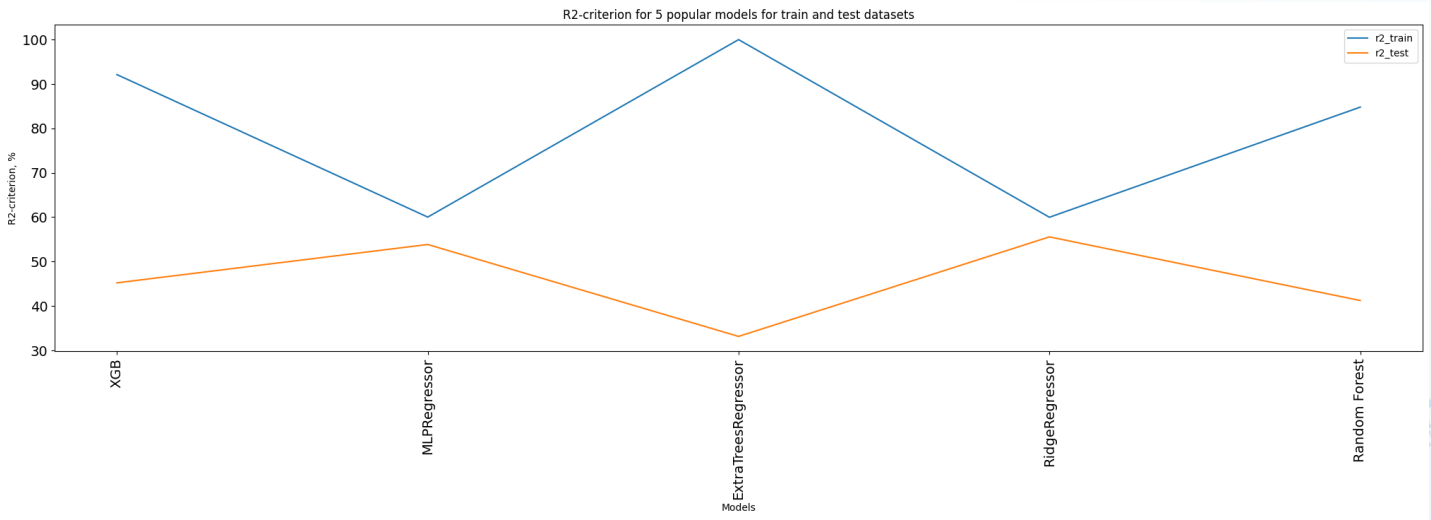
모델을 학습시키고 평가하자

랜덤 포레스트 / 릿지 회귀 / 엑스트라 트리 / MLP / XGB

학습과 테스트

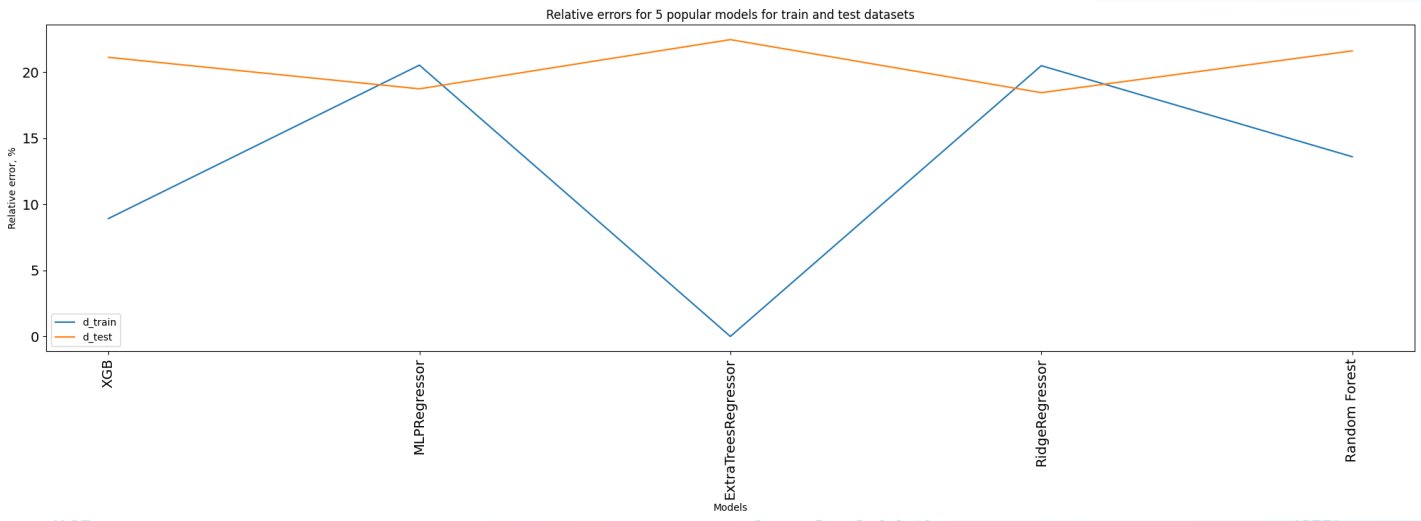
5가지 모델에 학습데이터를 주고 공부시키고
3가지 점수 측정법으로 채점

학습과 테스트 – 그래프 (R2)



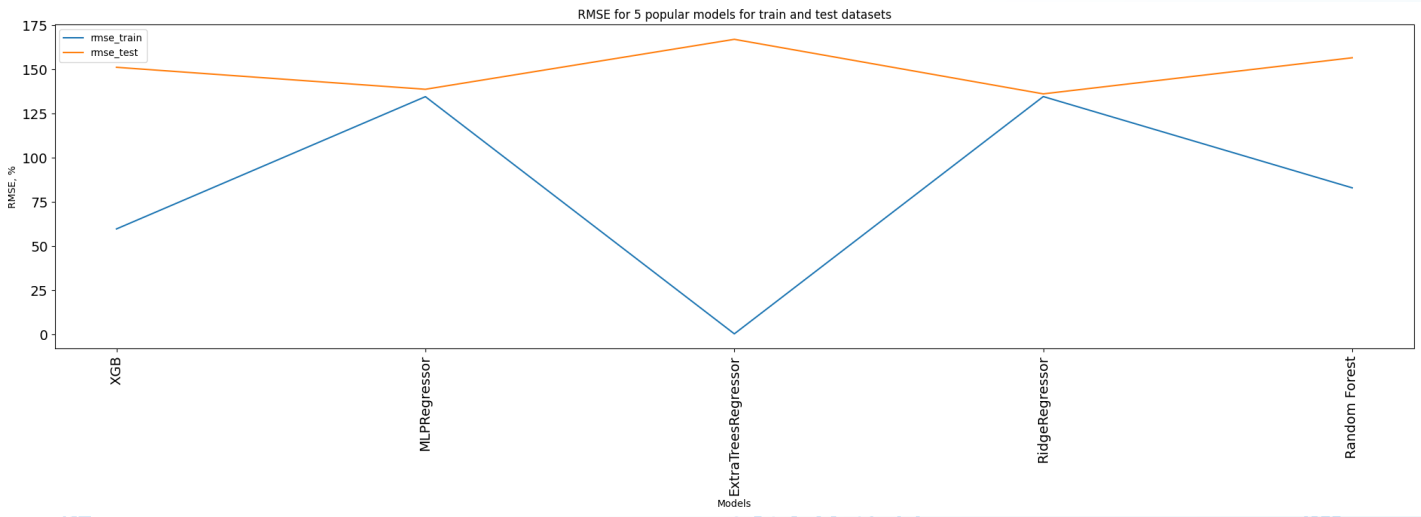
엑스트라 트리는 시험 문제만 외워버린 모양이다..
MLP와 Ridge는 다른 모델보다 문제를 잘 풀었다

학습과 테스트 – 그래프 (상대오차)



엑스트라 트리는 시험 문제는 잘 외웠지만 정답은 너무 차이가 난다
Ridge 회귀가 테스트에서 상대오차가 제일 낮다

학습과 테스트 – 그래프 (RMSE)

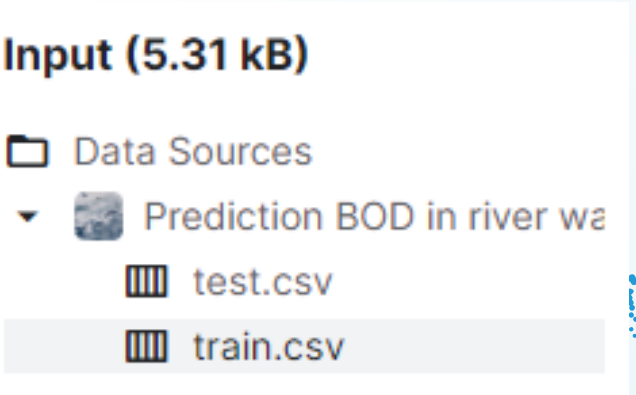


나머지 점수들과 양상이 비슷하다
시험 문제만 외운 엑스트라 트리는 “과적합” 되었다고 볼 수 있다

예측

지금까지 모의고사로 어느 모델이 예측을 잘 하는 지 평가했다

이제는 진짜 시험을 칠 시간!



예측

```
testn = pd.read_csv('test.csv')
testn.info()
```

```
testn = testn.drop(['Id','3','4','5','6','7'], axis = 1)
```

```
testn.head()
```

```
#For models from Sklearn
testn = pd.DataFrame(scaler.transform(testn), columns = testn.columns)
```

```
mlp_GS.fit(train0, train_target0)
mlp_GS.predict(testn)[:5]
```

```
xgb_reg.fit(train0, train_target0)
xgb_reg.predict(testn)[:5]
```

```
random_forest.fit(train0, train_target0)
random_forest.predict(testn)[:5]
```

```
etr.fit(train0, train_target0)
etr.predict(testn)[:5]
```

```
ridge.fit(train0, train_target0)
ridge.predict(testn)[:5]
```

test.csv를 읽는다
(target 값이 원래 없는 파일)

Null 값이 많은 열은 지운다

학습 데이터를 쪼개기 전의
train 파일로 학습시킨다

각 모델에게 test 정보로
예측시킨다

예측 - 결과비교

시험문제

	1	2
0	6.80	5.40
1	4.71	4.20
2	2.10	3.40
3	5.35	5.85
4	4.80	5.30

MLP

```
array([6.09303094, 3.94754094, 1.21875865, 4.54306964, 3.98969117])
```

XGB

```
array([6.210833 , 4.81564  , 2.397621 , 4.9732246, 5.2233343],
      dtype=float32)
```

Rnd Forest

```
array([6.13695, 4.31025, 2.24162, 4.90642, 5.83687])
```

ExTree

```
array([5.8813, 4.221 , 2.1812, 4.9056, 5.8896])
```

Ridge

```
array([6.32349671, 4.73803635, 2.88768171, 5.46142929, 5.00075999])
```

테스트 점수를 기준으로 판단하면 Ridge나 MLP가 신뢰도가 높다

결론

결론

데이터의 종류나 양에 따라
다양한 모델들의 학습 능력이
달라지는 것을 볼 수 있었다.

특히, ExTree는 과적합 문제를
보완한 알고리즘이라는데,
이 예측에서는 완전히
과적합되어 있는 모습을
볼 수 있었다.

감사합니다