Tree Models + AutoML

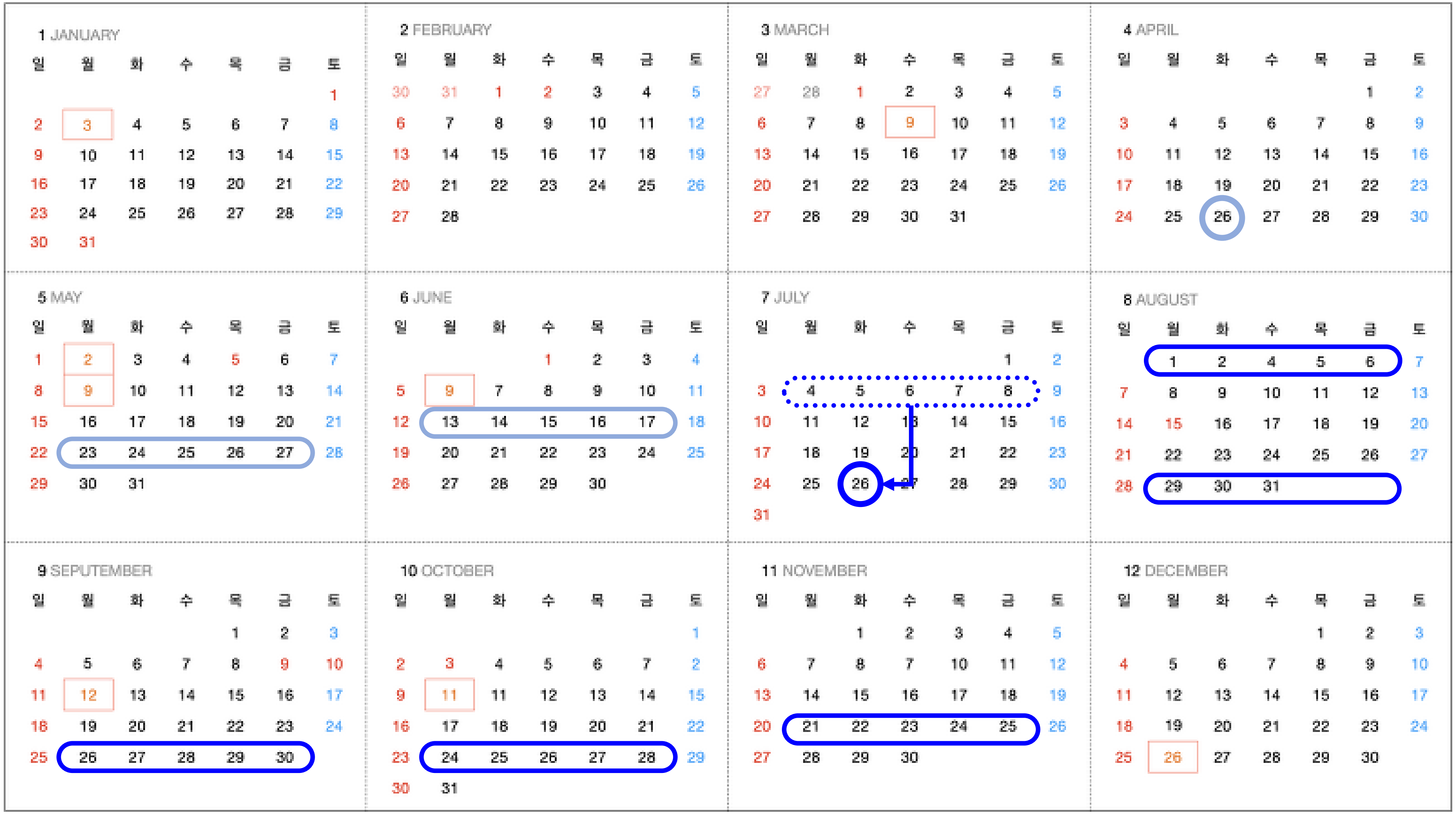
실습 코드 : https://bit.ly/3PkGBHc

2022.07.14.

한국에너지기술연구원 계산과학연구실

이제현

2022 에너지+AI 학습 조직 : 총 9회 + α



머신 러닝 강좌

- base : Scikit-learn MOOC @inria
  - 머신 러닝 위주. 딥 러닝은 skip (기회가 되면 한번쯤은 다룰 수도..?)
  - 소스 코드 포함 강의 자료 : 원내 게시판 공개
  - 강의 영상 : KIER-Tube & Youtube 공개

1차 모임 (4월)
머신러닝 기본 개념

2차 모임 (5월)
Modeling pipeline

3차 모임 (6월)
Best Model?

4차 모임 (7월)
Hyperparameter
→ Tree Models

5차 모임 (8월)
Linear Models
→ Hyperparameter

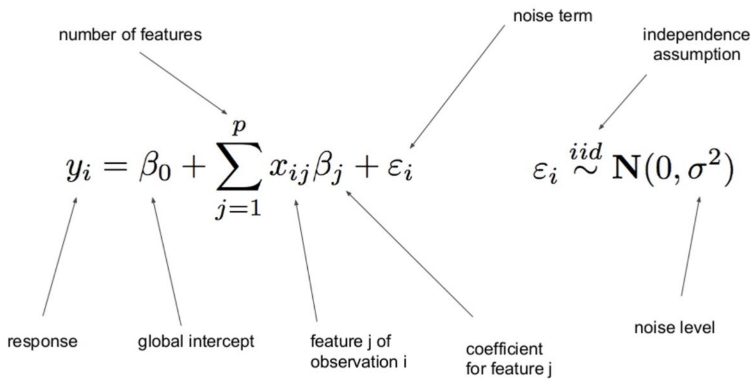
6차 모임 (8월)
Tree Models
→ Nonlinear Models

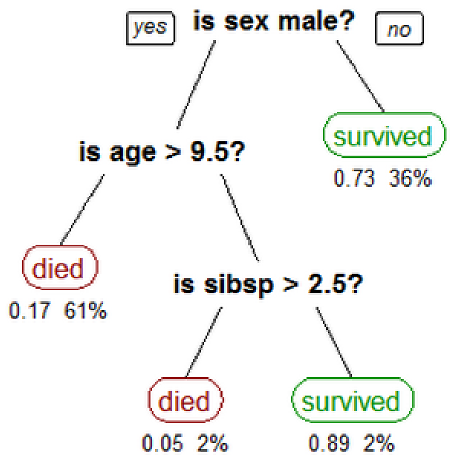
7차 모임 (9월)
Ensemble Models

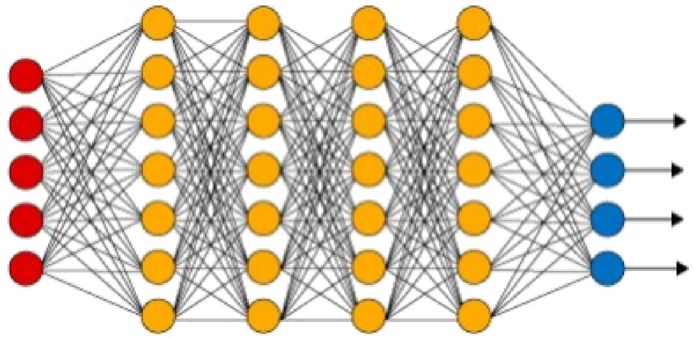
8차 모임 (10월)
CV & metrics
→ Neural Network (?)

9차 모임 (11월)
마무리

머신 러닝 Machine Learning



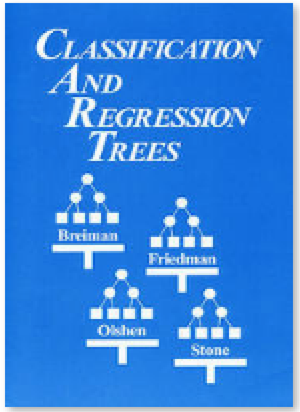




	선형 모델 Linear Model	트리 모델 Tree Model	신경망 모델 Neural Network
비선형성	X	O	◎
설명력	O	O	△
속도	O	△	X
비용	X	X	◎

1. Decision Tree

- CART (Classification and Regression Tree)



Book

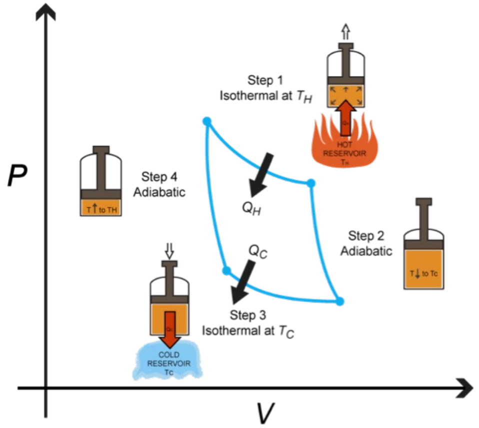
Classification And Regression Trees

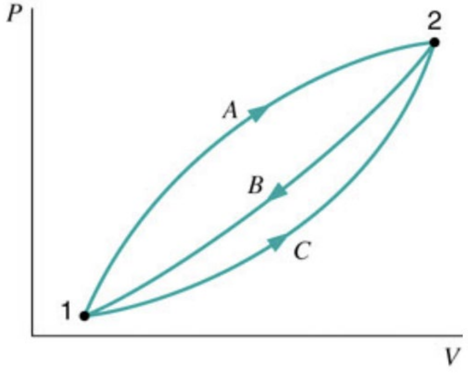
By Leo Breiman, Jerome H. Friedman, Richard A. Olshen, Charles J. Stone

Edition	1st Edition
First Published	1984
eBook Published	25 October 2017
Pub. Location	New York
Imprint	Routledge
DOI	https://doi.org/10.1201/9781315139470
Pages	368
eBook ISBN	9781315139470
Subjects	Mathematics & Statistics

Entropy by Clausius

- 자연계의 변화에 존재하는 방향성을 설명하기 위해 도입 (1865)
  - “쓸데 없는 에너지” ~ “주어진 열이 일로 전환되기 어려운 정도”





카르노 기관 1 cycle $\dfrac{Q_H}{Q_L} = \dfrac{T_H}{T_L}$ → Clausius 부등식 $\oint \dfrac{\delta Q}{T} \le 0$ Rev. : = Irrev. : <

이상 기체 가역 반응 A → B $\oint \dfrac{\delta Q}{T} = 0 = \int_1^2 \left(\dfrac{\delta Q}{T}\right)_A + \int_2^1 \left(\dfrac{\delta Q}{T}\right)_B$

C → B $\oint \dfrac{\delta Q}{T} = 0 = \int_1^2 \left(\dfrac{\delta Q}{T}\right)_C + \int_2^1 \left(\dfrac{\delta Q}{T}\right)_B$

두 식 차이 $\int_1^2 \left(\dfrac{\delta Q}{T}\right)_A = \int_1^2 \left(\dfrac{\delta Q}{T}\right)_C$

→ 경로 무관 상태 함수 $\boldsymbol{dS = \left(\dfrac{\delta Q}{T}\right)_{rev}}$ = 엔트로피

https://www.shmoop.com/study-guides/physics/thermodynamics/carnot-cycle
http://contents.kocw.or.kr/KOCW/document/2016/chungnam/leewanghee/14.pdf

Entropy by Boltzmann & Gibbs

- “어떤 계가 취할 수 있는 미시적인 상태의 수” = “무질서도” (1877)

열역학적 엔트로피

$$dS = \frac{dQ}{T}$$

에너지 보존법칙

$$dU = dQ + dW$$

: 이 둘을 합치면

$$dU = TdS - pdV$$

$$U(S, V)$$

$$dU = \left(\frac{\partial U}{\partial S}\right)_V dS - \left(\frac{\partial U}{\partial V}\right)_S dV$$

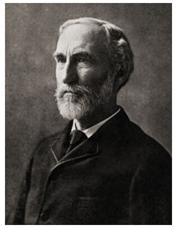
$$\boxed{\left(\frac{\partial S}{\partial U}\right)_V = \frac{1}{T}}$$

통계열역학적 엔트로피

$$S = k_B \ln \Omega$$

온도의 통계열역학적 정의

$$\frac{1}{T} = k_B \frac{d\ln\Omega}{dE}$$

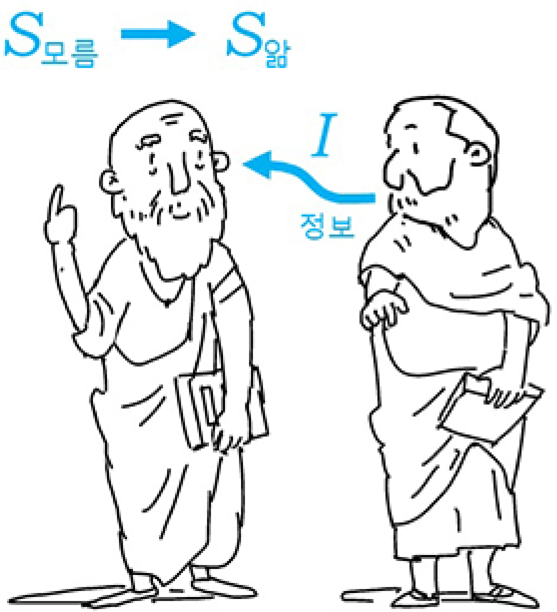


$$S = -k_B \sum_i p_i \ln(p_i)$$

i : microstate
p_i : probability of i

Information Entropy by Claude Shannon

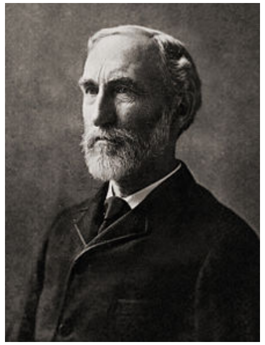
- "각 메시지에 포함된 정보의 기댓값(평균)" (1948, "A mathematical theory of communication")
  - 경우의 수가 많다 = 확정적이지 않다 = 엔트로피가 높다
  - 추가 정보가 주어진다 = 불확실한 상태에서 벗어난다 = 엔트로피가 낮아진다.

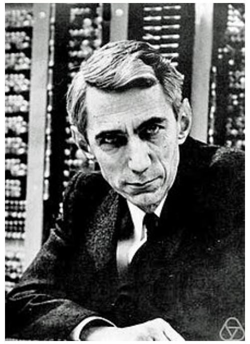


$S_{앎} = S_{모름} - I$

$S = \log_2 2^N = N = 2$

동전을 2개 던진다
→ 경우의 수는 4가지
→ 엔트로피 S = 2





"확률이 낮을수록 정보의 가치는 높다"
ex. 주사위 vs 동전 : 1/6 vs 1/2

$$H = -\sum_i p_i \log_2(p_i)$$

$$S = -k_B \sum_i p_i \ln(p_i)$$

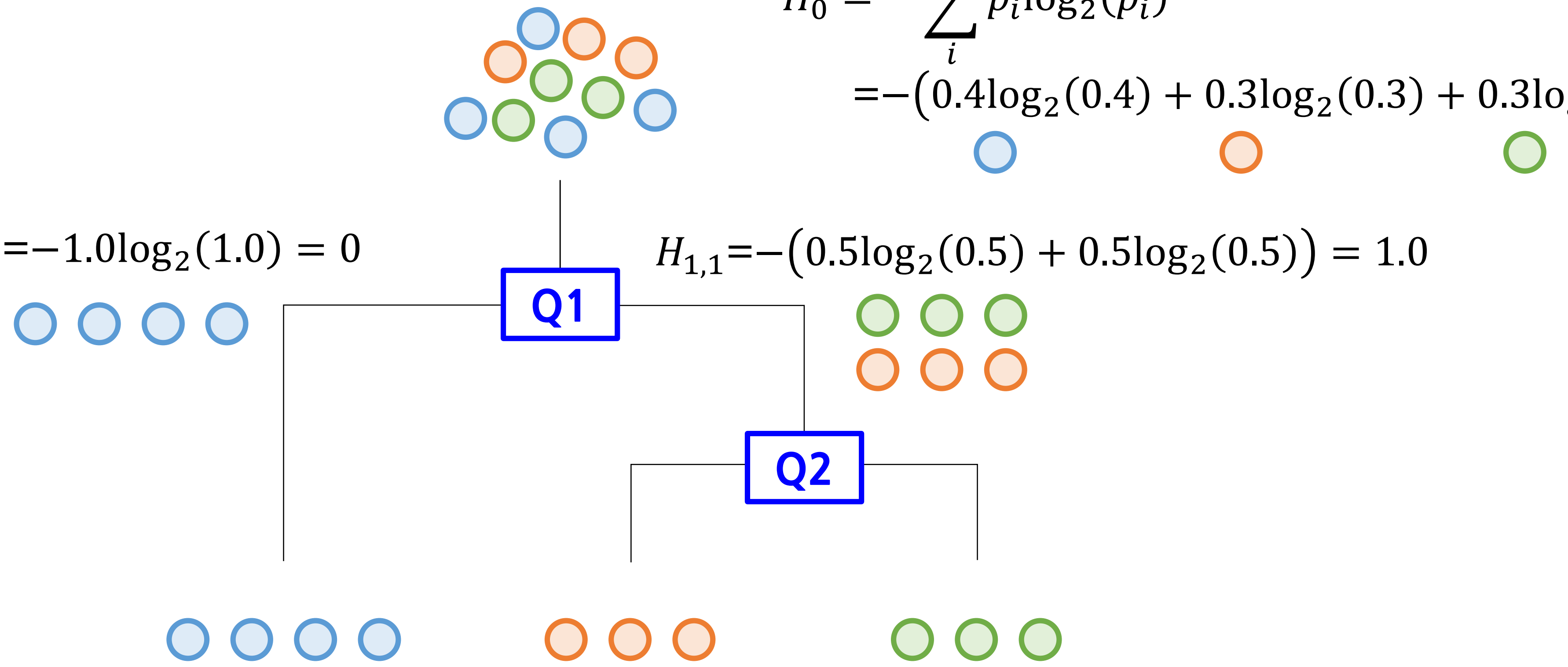
스무고개 : 반복적 정보 획득을 통해 엔트로피를 낮춰 가는 게임
→ 좋은 질문 : 한 번에 엔트로피를 많이 낮출 수 있는 질문

공학으로 세상을 말한다 http://www.yes24.com/Product/Goods/2498927

Decision Tree

- 스무고개 놀이의 질문 찾기

$$H_0 = -\sum_i p_i \log_2(p_i)$$
$$= -\big(0.4\log_2(0.4) + 0.3\log_2(0.3) + 0.3\log_2(0.3)\big) = 1.571$$



$$H_{1,0} = -1.0\log_2(1.0) = 0$$

$$H_{1,1} = -\big(0.5\log_2(0.5) + 0.5\log_2(0.5)\big) = 1.0$$

$$H_{2,0} = -1.0\log_2(1.0) = 0$$

$$H_{2,1} = -1.0\log_2(1.0) = 0$$

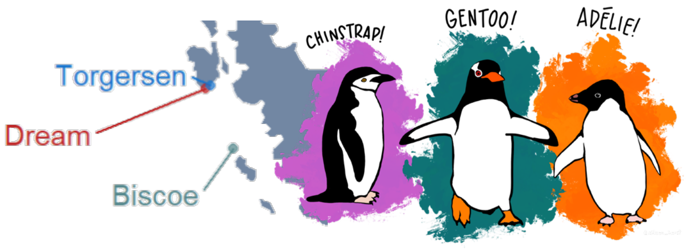
$$H_{2,2} = -1.0\log_2(1.0) = 0$$

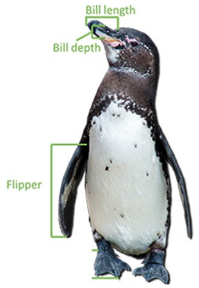
scikit-learn Decision Tree Regressor

- 펭귄 체중 예측 문제

- **pipeline with DecisionTreeRegressor**

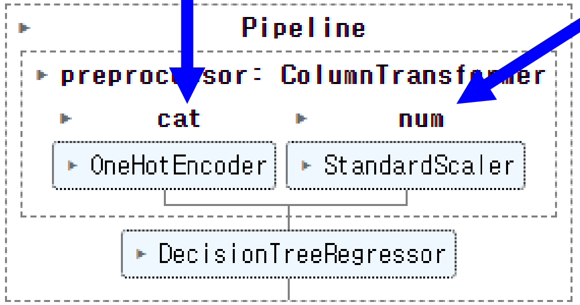
```python
from sklearn.preprocessing import OneHotEncoder
from sklearn.preprocessing import StandardScaler
from sklearn.compose import ColumnTransformer
from sklearn.pipeline import Pipeline
from sklearn.tree import DecisionTreeRegressor

def get_model(cat_features=["species", "island", "sex"],
              num_features=["bill_length_mm", "bill_depth_mm", "flipper_length_mm"],
              **kwargs):
    # 1-1.categorical feature에 one-hot encoding 적용
    cat_transformer = OneHotEncoder()

    # 1-2.numerical feature는 standard scaler 적용
    num_transformer = StandardScaler()

    # 2. 인자 종류별 전처리 적용
    preprocessor = ColumnTransformer([("cat", cat_transformer, cat_features),
                                      ("num", num_transformer, num_features)])

    # 3. 전처리 후 Decision Tree Regressor 적용
    pipeline = Pipeline(steps=[("preprocessor", preprocessor),
                               ("ml", DecisionTreeRegressor(**kwargs))])

    return pipeline
```





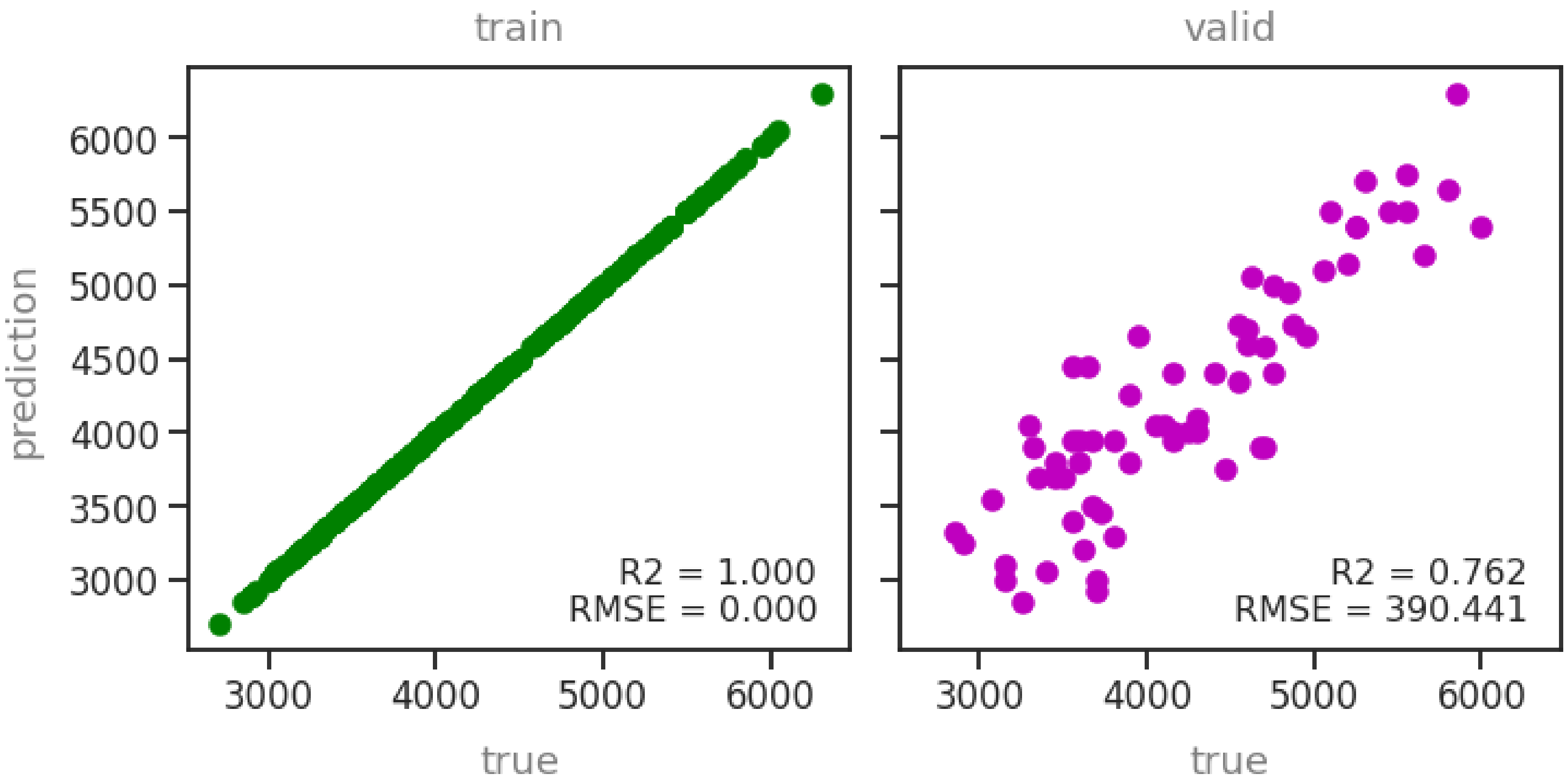
펭귄 서식지 3곳, 종 3가지, 성별

신체치수 3가지



scikit-learn Decision Tree Regressor

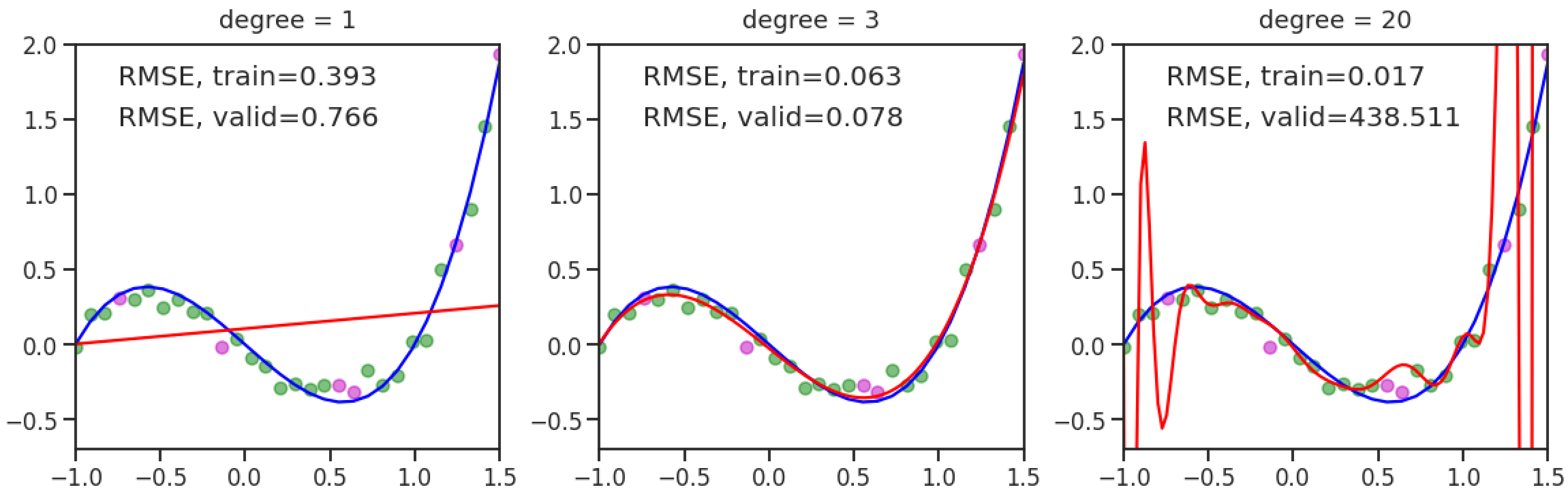
- 배운 데이터는 완벽. 안 배운 데이터는 ..?



scikit-learn Decision Tree Regressor

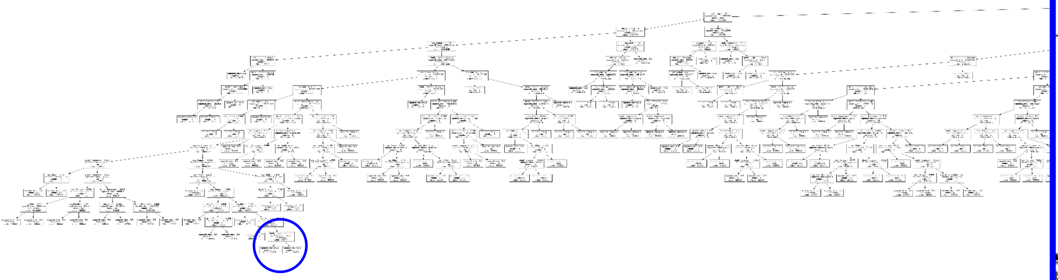
- 배운 데이터는 완벽. 안 배운 데이터는 ..?

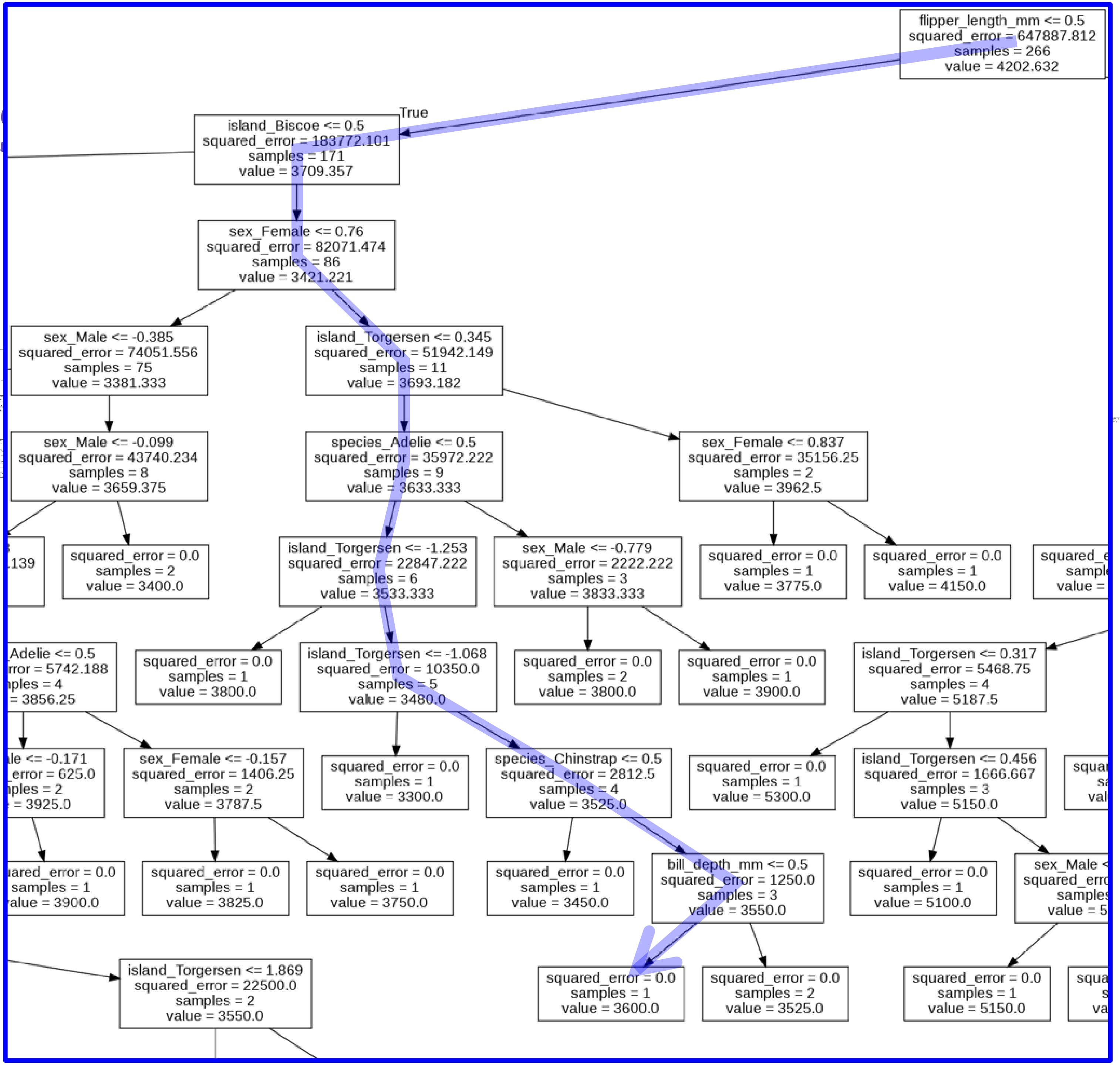
underfitting　　　Just fit　　　overfitting



scikit-learn Deci

- 이런 Tree가 만들어 졌습니다.



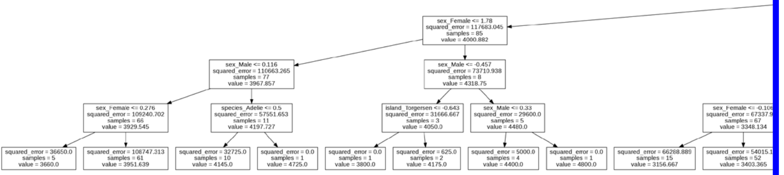


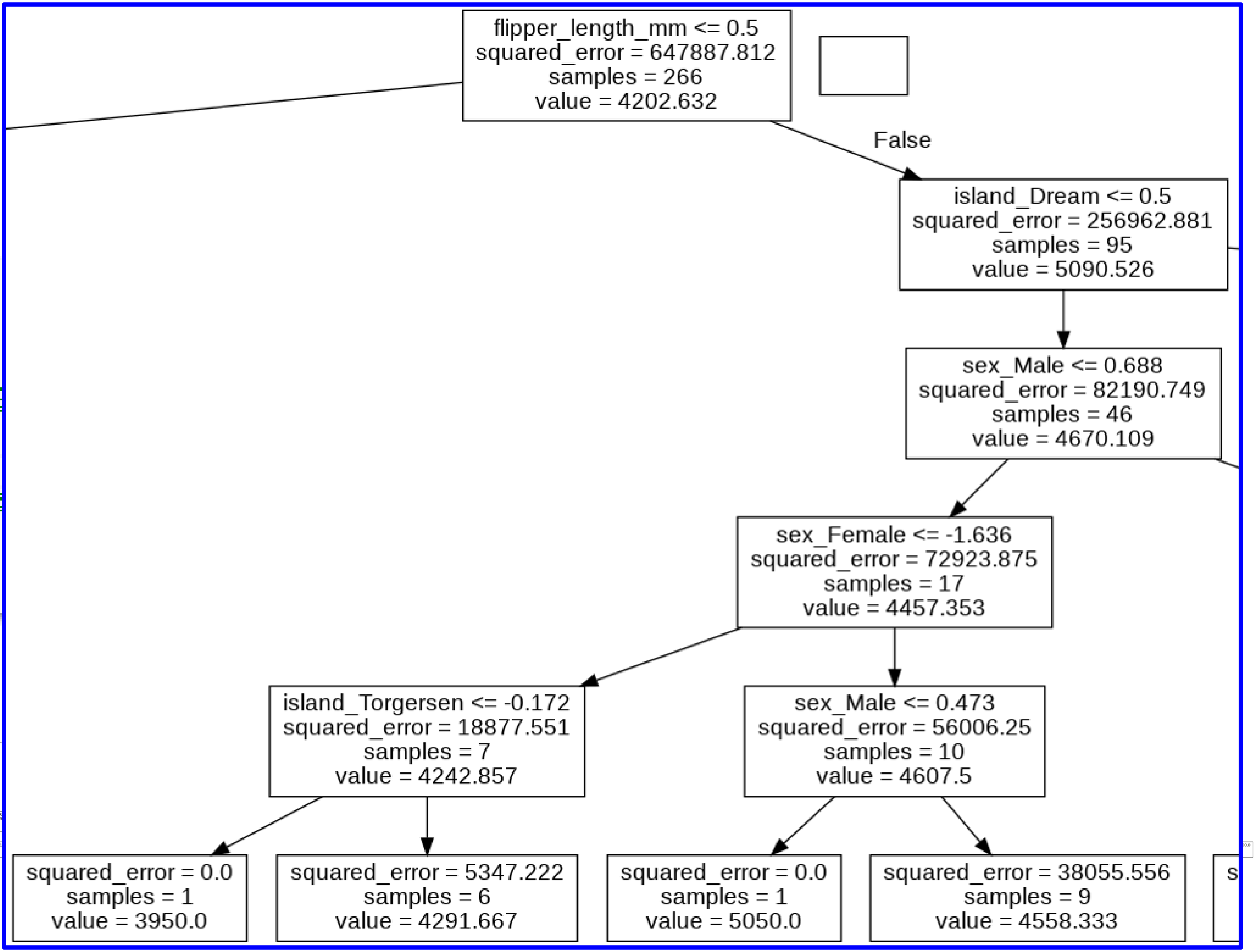
- 최대 18단계

scikit-learn

- 최대 깊이를 줄입니다 :

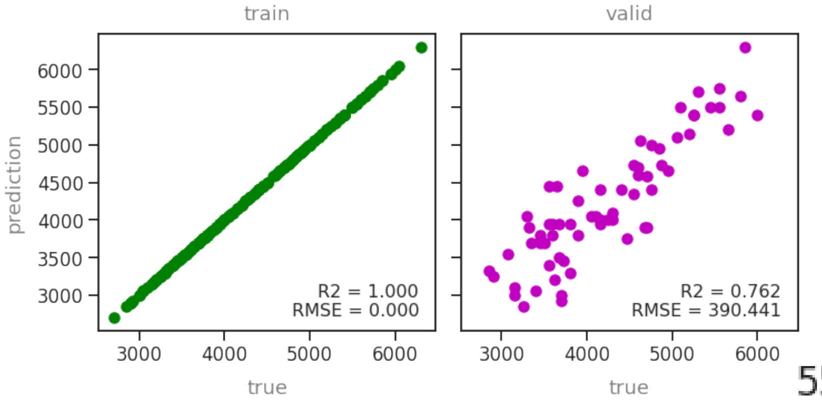
```
# pip
dt_ma

# 학습
dt_ma

# 예측
y_pre
y_pre
```

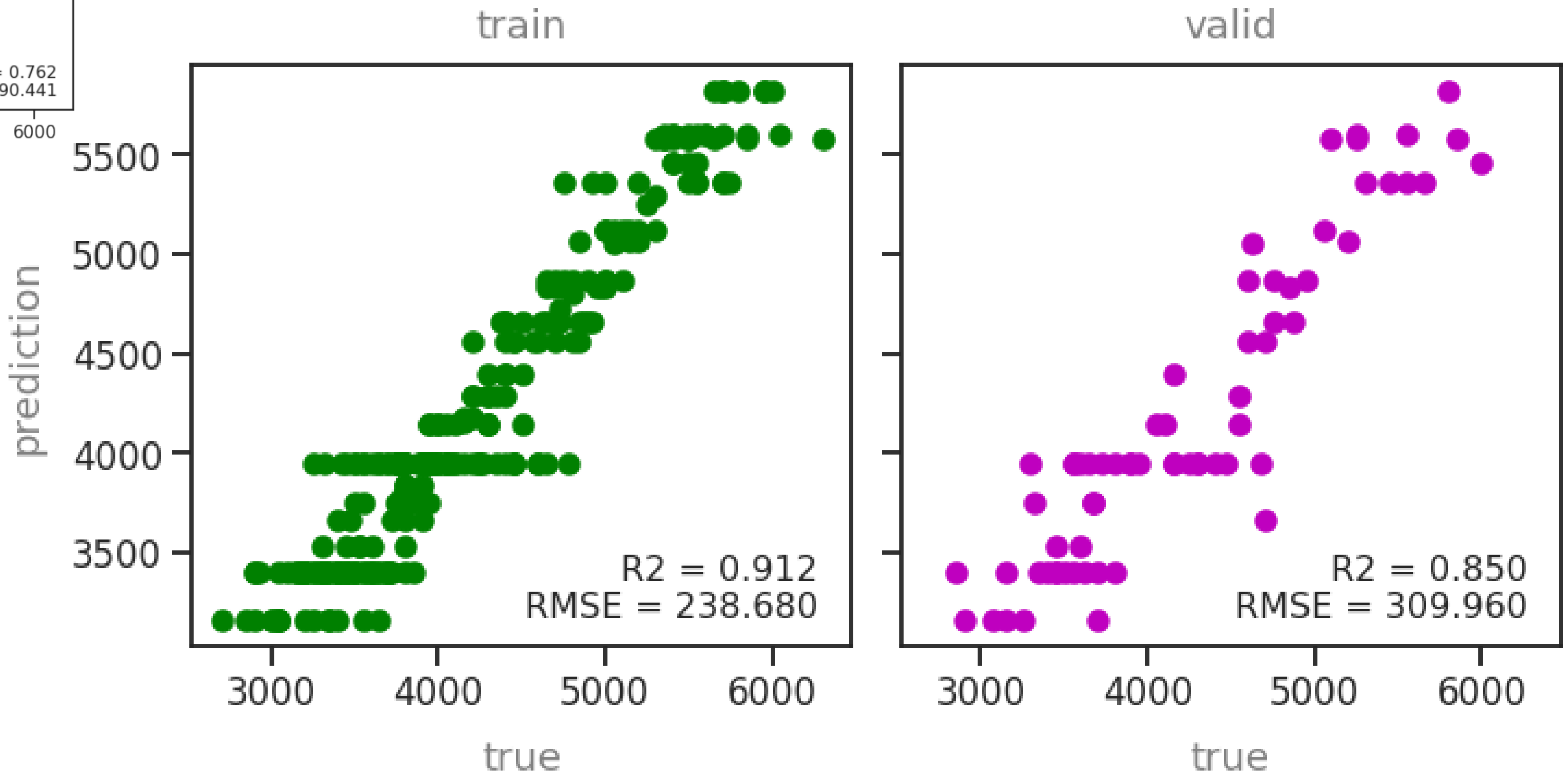




scikit-learn Decision Tree Regressor

- Trainset 성능은 저하, validation set 성능은 향상 → 둘이 가까워짐

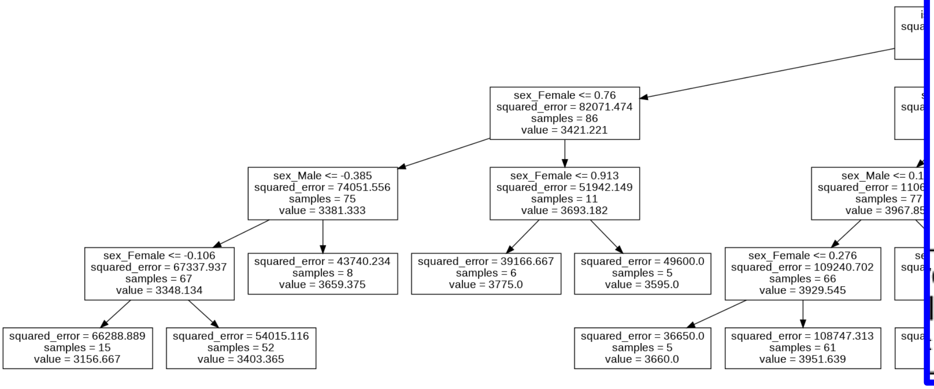


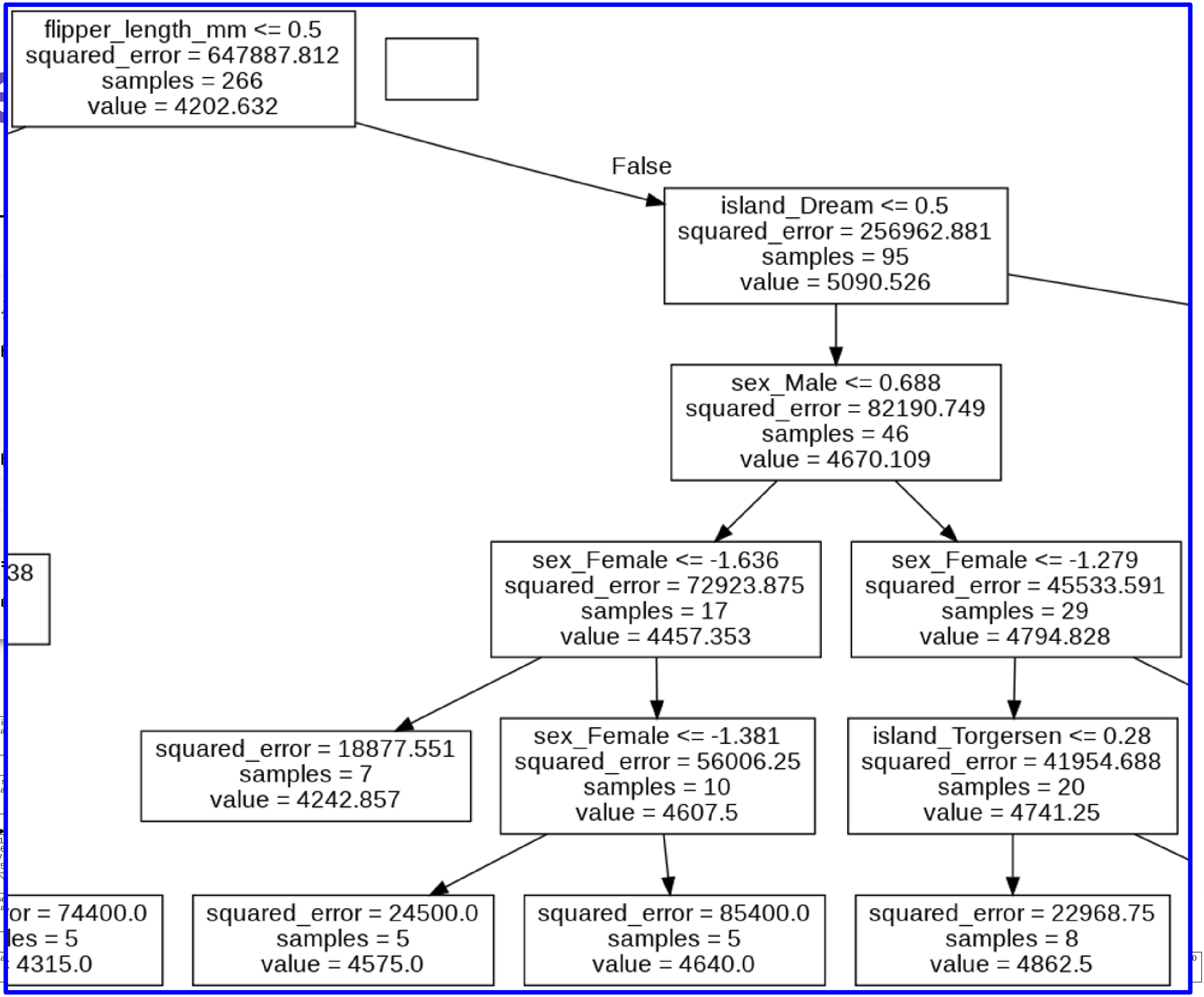


scikit-learn De

- Leaf의 최소 데이터 수를 늘립니

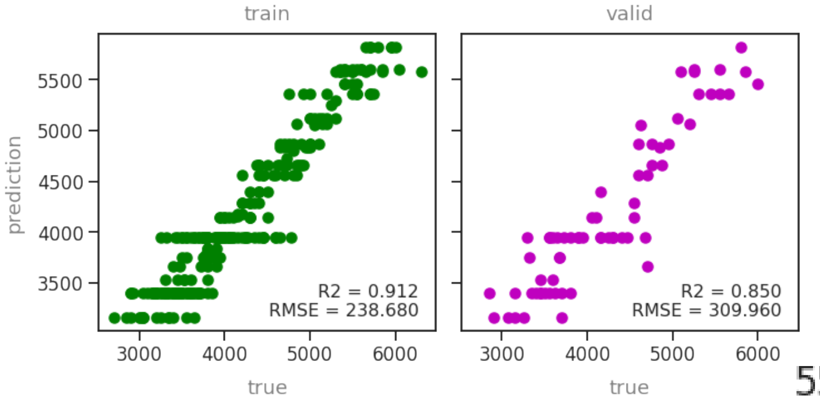
```
# pipeline 제
dt_minsamplel

# 학습
dt_minsamplel

# 예측
y_pred_train =
y_pred_val =
```

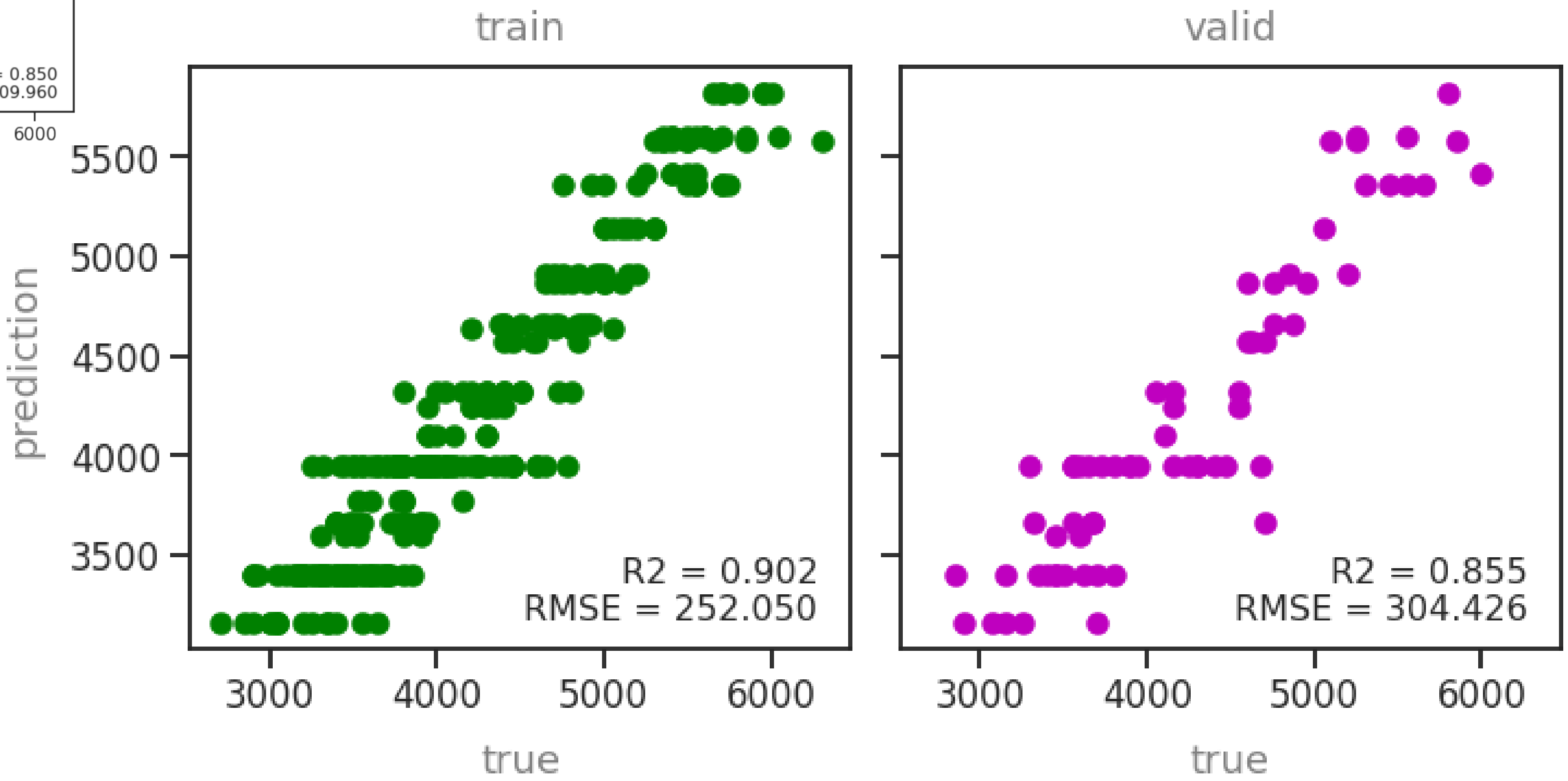




scikit-learn Decision Tree Regressor

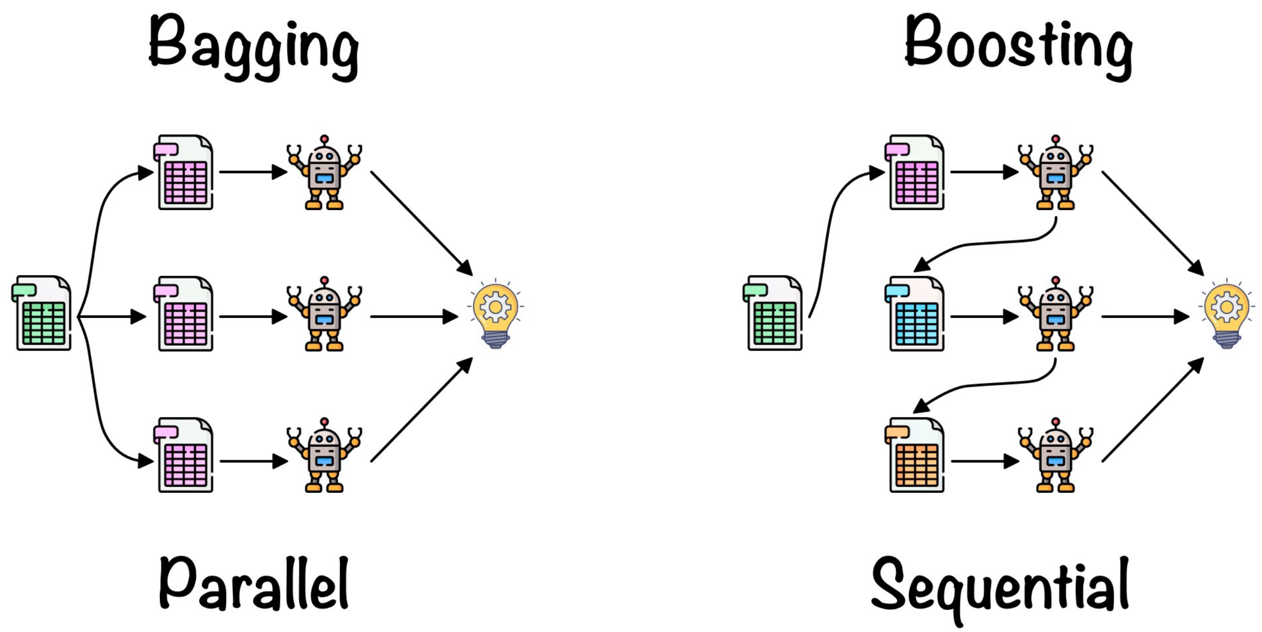
- Trainset 성능은 조금 더 저하, validation set 성능은 조금 더 향상 → 둘이 조금 더 가까워짐





Overfitting 방지: Ensemble Tree

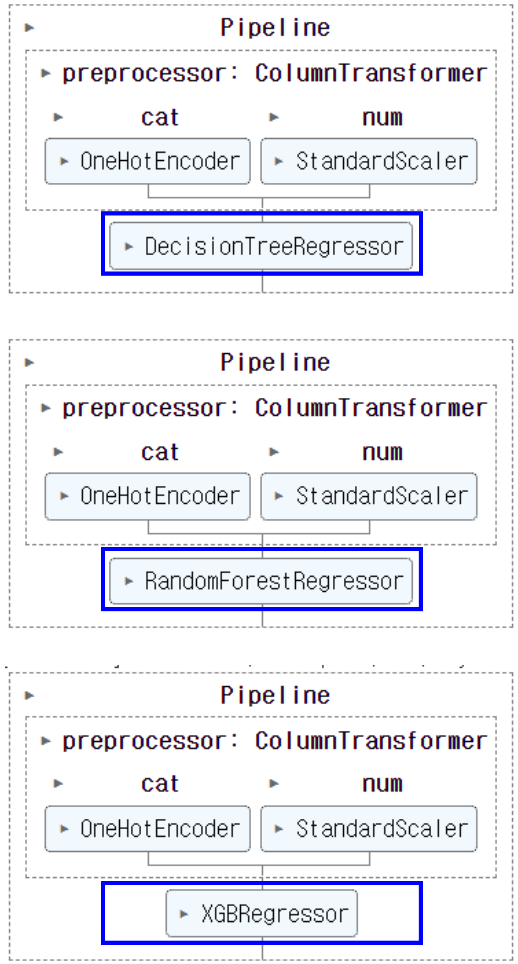
- Ensemble : 여러 model을 결합하여 과적합 방지 & 예측 성능 향상
  - Bagging : 학습 데이터를 중복을 허락해 랜덤하게 선택, 각각 모델을 취합 (voting, averaging)
    - Random Forest
  - Boosting : 하나의 model에서 시작, model을 추가하며 오차 감소
    - Gradient Boosting - LightGBM, XGBoost



https://buggyprogrammer.com/difference-between-bagging-and-boosting/

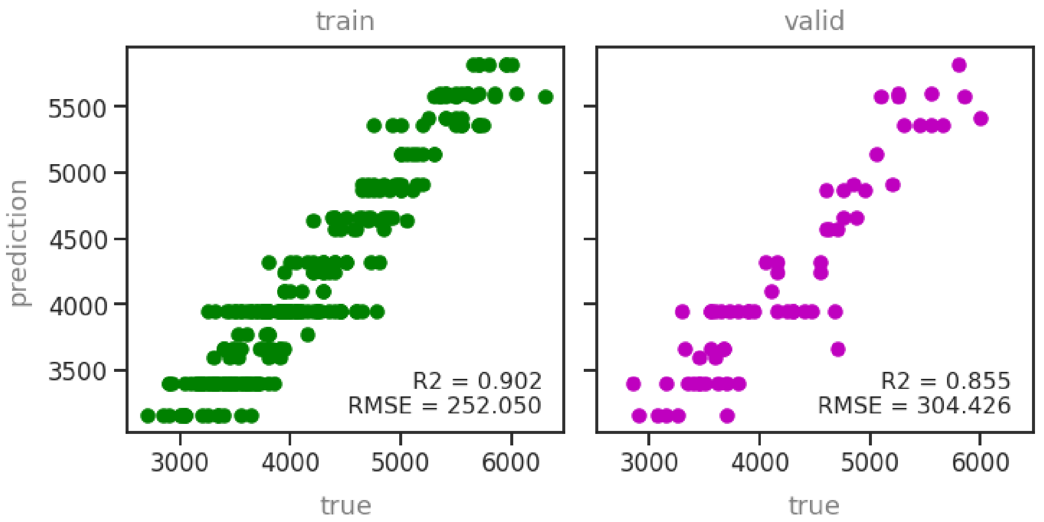
DT vs RF vs XGBoost

- 가변 모델 Pipeline 구축

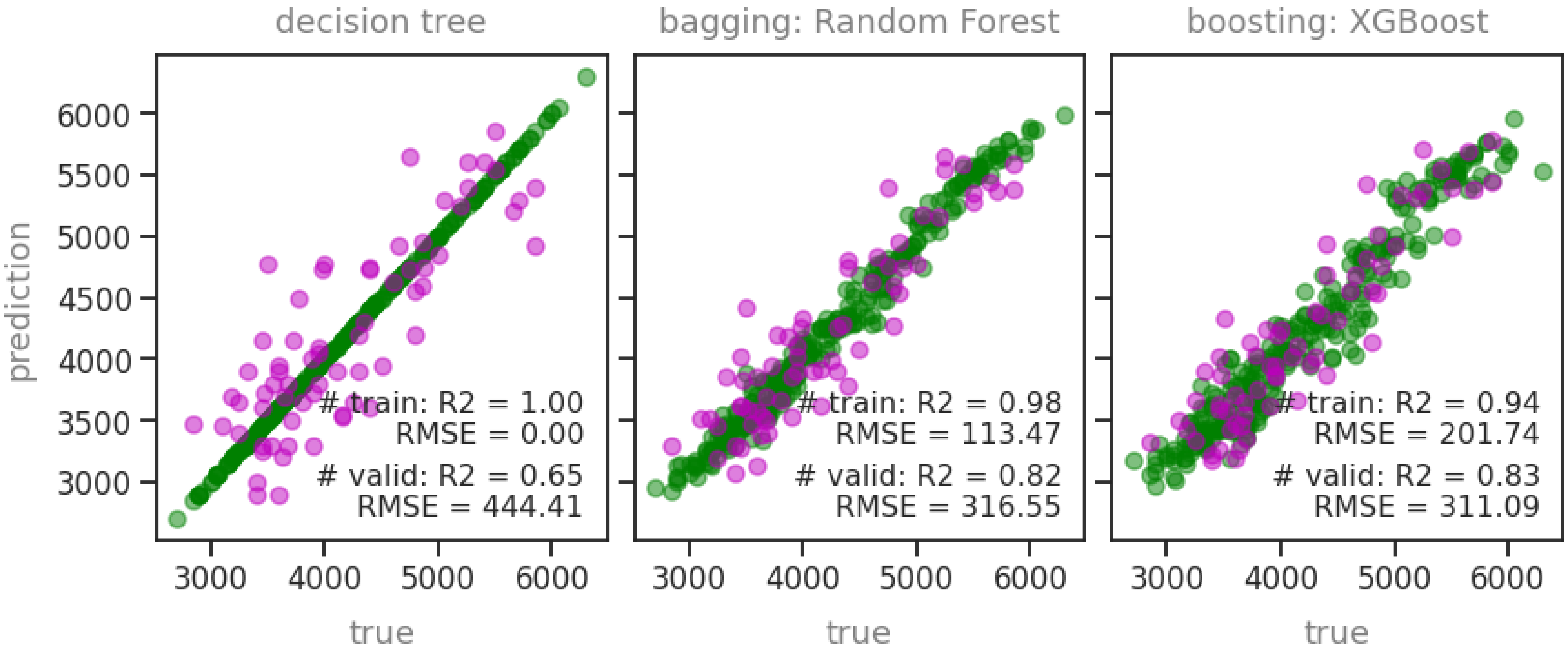



```python
from sklearn.ensemble import RandomForestRegressor
from lightgbm import LGBMRegressor
from xgboost import XGBRegressor

def get_model(method="dt",
              cat_features=["species", "island", "sex"],
              num_features=["bill_length_mm", "bill_depth_mm", "flipper_length_mm"],
              **kwargs):
    # 1-1.categorical feature에 one-hot encoding 적용
    cat_transformer = OneHotEncoder()

    # 1-2.numerical feature는 standard scaler 적용
    num_transformer = StandardScaler()

    # 2. 인자 종류별 전처리 적용
    preprocessor = ColumnTransformer([("cat", cat_transformer, cat_features),
                                      ("num", num_transformer, num_features)])

    # 3. 전처리 후 입력된 방법론 적용
    if method == "dt":
        ml = ("ml", DecisionTreeRegressor(**kwargs))
    elif method == "rf":
        ml = ("ml", RandomForestRegressor(**kwargs))
    elif method == "lgbm":
        ml = ("ml", LGBMRegressor(**kwargs))
    elif method == "xgb":
        ml = ("ml", XGBRegressor(**kwargs))

    pipeline = Pipeline(steps=[("preprocessor", preprocessor),
                               ml])

    return pipeline
```

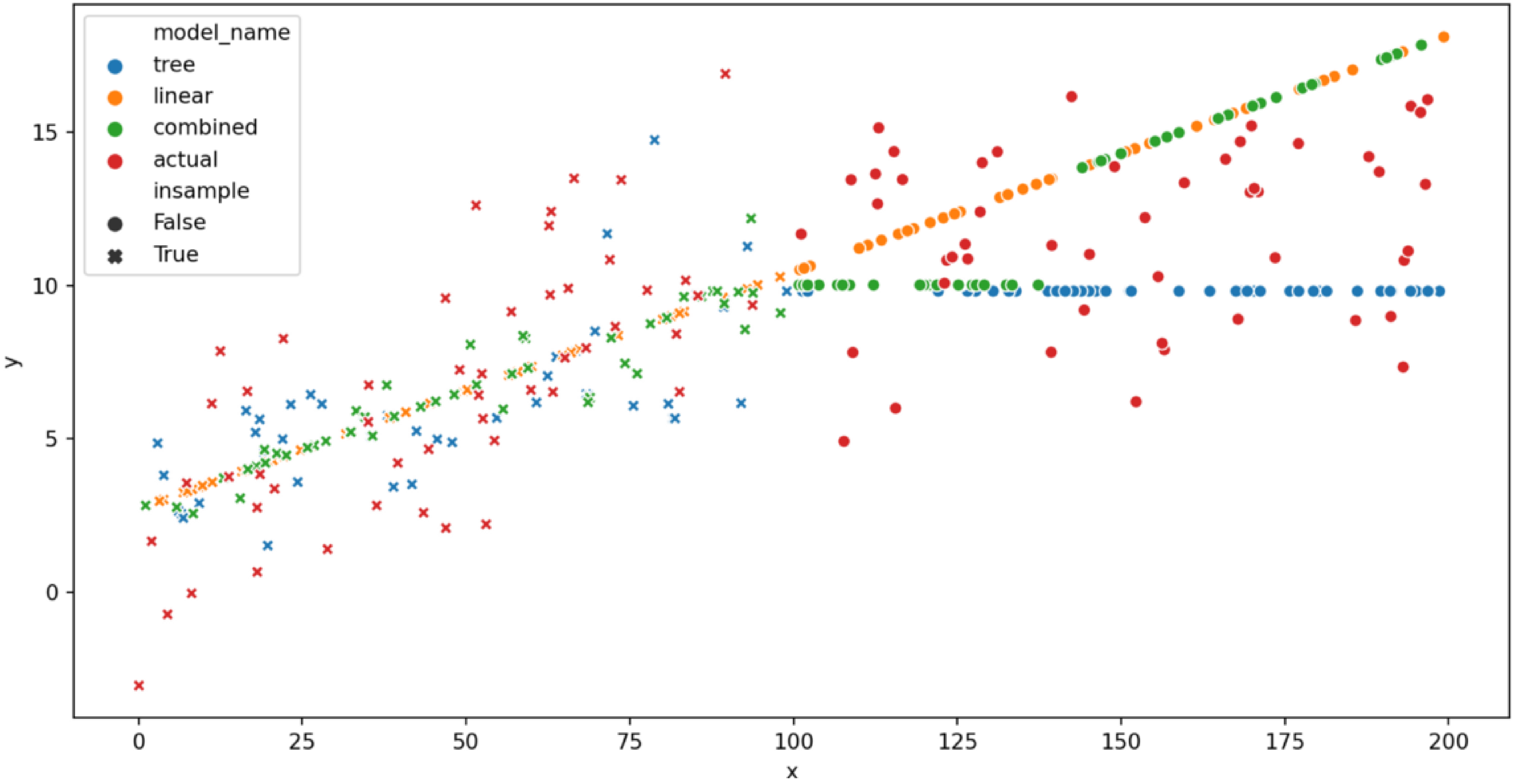
DT vs RF vs XGBoost



- 단일 모델 사용시보다 성능 향상, 과적합 방지
  - Hyperparameter tuning 성능 추가 향상 필요



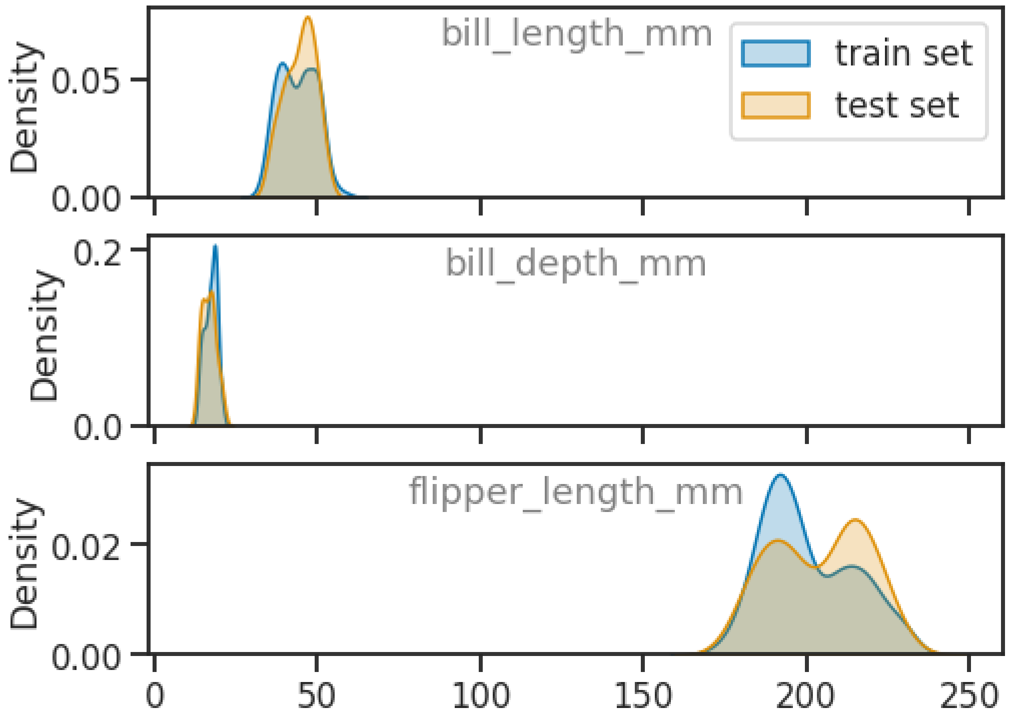
Tree model 유의점 : Extrapolation



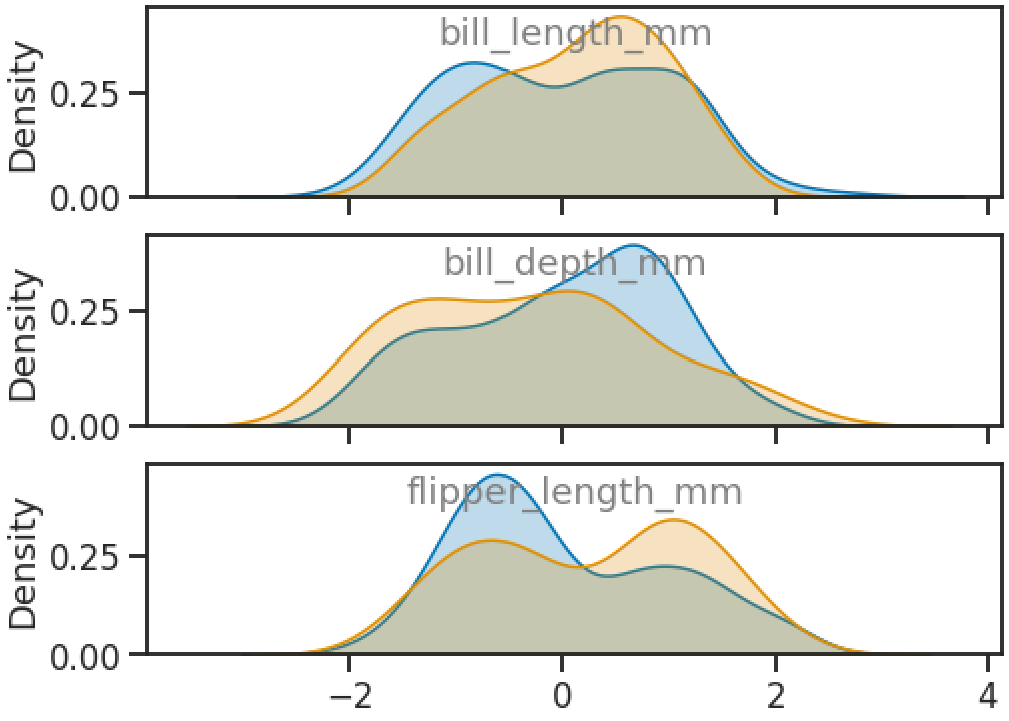
https://towardsdatascience.com/combining-tree-based-models-with-a-linear-baseline-model-to-improve-extrapolation-c100bd448628

Tree model 유의점 : Scaling 불필요

Standard Scaler 적용 전



Standard Scaler 적용 후



AutoML : PyCaret

https://pycaret.gitbook.io/docs/

AutoML : PyCaret

- Step 1. setup

```
from copy import deepcopy
data_pycaret = deepcopy(X_train)
data_pycaret["body_mass_g"] = y_train
```

```
from pycaret.regression import *
s = setup(data_pycaret, target='body_mass_g')
```

	Description	Value
0	session_id	7351
1	Target	body_mass_g
2	Original Data	(266, 7)
3	Missing Values	False
4	Numeric Features	3
5	Categorical Features	3
6	Ordinal Features	False
7	High Cardinality Features	False
8	High Cardinality Method	None
9	Transformed Train Set	(186, 10)
10	Transformed Test Set	(80, 10)
11	Shuffle Train-Test	True
12	Stratify Train-Test	False
13	Fold Generator	KFold
14	Fold Number	10

- Step 2. compare_models

```
best = compare_models()
```

	Model	MAE	MSE	RMSE	R2	RMSLE	MAPE	TT (Sec)
lasso	Lasso Regression	247.5016	96188.5188	306.4862	0.8448	0.0757	0.0610	0.014
lr	Linear Regression	247.2969	96329.0867	306.7353	0.8445	0.0758	0.0610	0.757
lar	Least Angle Regression	247.2862	96326.7976	306.7316	0.8445	0.0758	0.0610	0.014
ridge	Ridge Regression	249.0347	97004.0398	307.6309	0.8438	0.0760	0.0614	0.012
br	Bayesian Ridge	248.8887	96936.6859	307.5623	0.8438	0.0760	0.0614	0.013
llar	Lasso Least Angle Regression	253.2021	98480.0116	310.0893	0.8422	0.0766	0.0625	0.014
huber	Huber Regressor	247.8342	98608.8259	310.1308	0.8418	0.0767	0.0610	0.037
ada	AdaBoost Regressor	247.0875	101124.4603	313.4655	0.8382	0.0771	0.0609	0.067
lightgbm	Light Gradient Boosting Machine	259.4533	105001.3610	321.7189	0.8330	0.0798	0.0639	0.054
rf	Random Forest Regressor	259.2117	107942.8235	326.1563	0.8276	0.0813	0.0641	0.409
gbr	Gradient Boosting Regressor	262.3502	110425.7252	329.1704	0.8227	0.0815	0.0644	0.050
et	Extra Trees Regressor	262.3006	113926.3309	334.8787	0.8187	0.0842	0.0653	0.390
en	Elastic Net	291.7242	139113.2867	366.8851	0.7760	0.0901	0.0715	0.014
knn	K Neighbors Regressor	296.6944	141985.5773	373.1102	0.7682	0.0908	0.0726	0.062
dt	Decision Tree Regressor	326.4620	165929.0936	404.3962	0.7365	0.1010	0.0802	0.015
omp	Orthogonal Matching Pursuit	326.4182	171511.1888	408.0774	0.7244	0.1008	0.0803	0.012
par	Passive Aggressive Regressor	654.4641	629168.3378	765.3411	-0.0147	0.1825	0.1688	0.013
dummy	Dummy Regressor	695.2393	681039.0562	821.8434	-0.0803	0.1925	0.1681	0.011

https://pycaret.gitbook.io/docs/

AutoML : PyCaret

- Step 3. Evaluate