**sklearn Decision Tree 기반 regression 모델:**

**parameter 정리 & 실험**

**Scope:**

**Input:** N차원의 continuous vector값

*categorical data는 one-hot encoding을 이용하여 기하적 특성 부여*

**output:** continuous scalar값

**참조:**

Decision Tree Regressor

<https://scikit-learn.org/stable/modules/generated/sklearn.tree.DecisionTreeRegressor.html>

Random Forest Regressor

[https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.RandomForestRegressor.html#sklearn.ensemble.RandomForestRegressor](https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.RandomForestRegressor.html" \l "sklearn.ensemble.RandomForestRegressor)

Gradient Boosting Regressor

<https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.GradientBoostingRegressor.html>

XGBoost

[https://xgboost.readthedocs.io/en/latest/python/python\_api.html#module-xgboost.sklearn](https://xgboost.readthedocs.io/en/latest/python/python_api.html" \l "module-xgboost.sklearn)

LightGBM

<https://lightgbm.readthedocs.io/en/latest/Parameters-Tuning.html>

CatBoost

<https://catboost.ai/docs/concepts/python-reference_catboostregressor.html>

+@: **Support Vector Regression**, DNN, time series 등등

**실험에 사용된 데이터: 1D & 2D continuous vector**

Target function: **f(x) = sin(x)**, 혹은 **f((x1,x2)) = x2\*sin(x1)**

현실의 **noisy data**구현을 위해X와 y에gaussian noise 추가 가능.

**Imbalanced data** 구현을 위해 X에 skewed distribution 설정 가능.

<https://en.wikipedia.org/wiki/Skewness>

y값에 **주기적인** **outlier** 설정 가능.

**Dense Data**의 경우를 상정했고, Sparse Data의 경우는 구현하지 않았음.

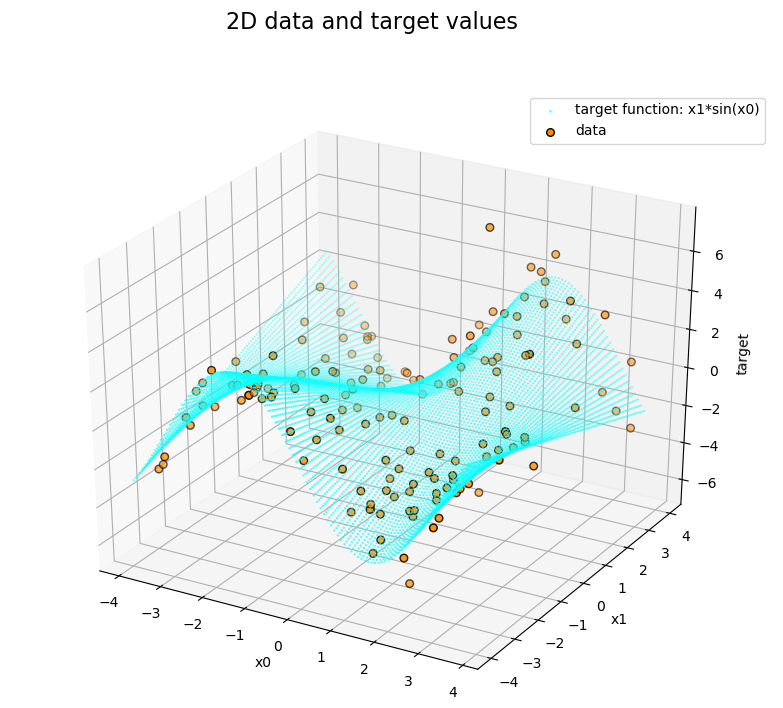
텍스트이(가) 표시된 사진

자동 생성된 설명

텍스트이(가) 표시된 사진

자동 생성된 설명

2차원 데이터 x1sin(x0) 예:



**전 학습모델 공통:** Decision Treeparameters

<https://scikit-learn.org/stable/modules/generated/sklearn.tree.DecisionTreeRegressor.html>

**노드 분할 Impurity (혹은 노드 별 Loss Function) 관련:**

criterion: {“mse”, “friedman\_mse”, “mae”, “poisson”}

**노드 분할 알고리즘 관련:**

splitter: {“random”, “best”}

max\_features {“auto”, “sqrt”, “log2”, “None”}

**Regularization 관련:**

max\_depth: {1,2,3,…,None}

min\_samples\_split: {2,3,4,… } OR {0.0~1.0}

min\_samples\_leaf: {1,2,3,… } OR {0.0~1.0}

min\_impurity\_split: float

~~ccp\_alpha: positive float <= classification 전용~~

**(번외) 각 샘플의 가중치를 이용한 Regularization:**

Min\_weight\_fractions\_leaf: {0.0~1.0}

**Criterion:** {“mse”, “friedman\_mse”, “mae”, “poisson”}, default: “mse”

각 internal node에서 분할에 사용될 loss function. (**H**는 주로 entropy에 사용되는 기호임)

**mse**: Mean Squared Error (L2 loss) 텍스트, 손목시계이(가) 표시된 사진

자동 생성된 설명

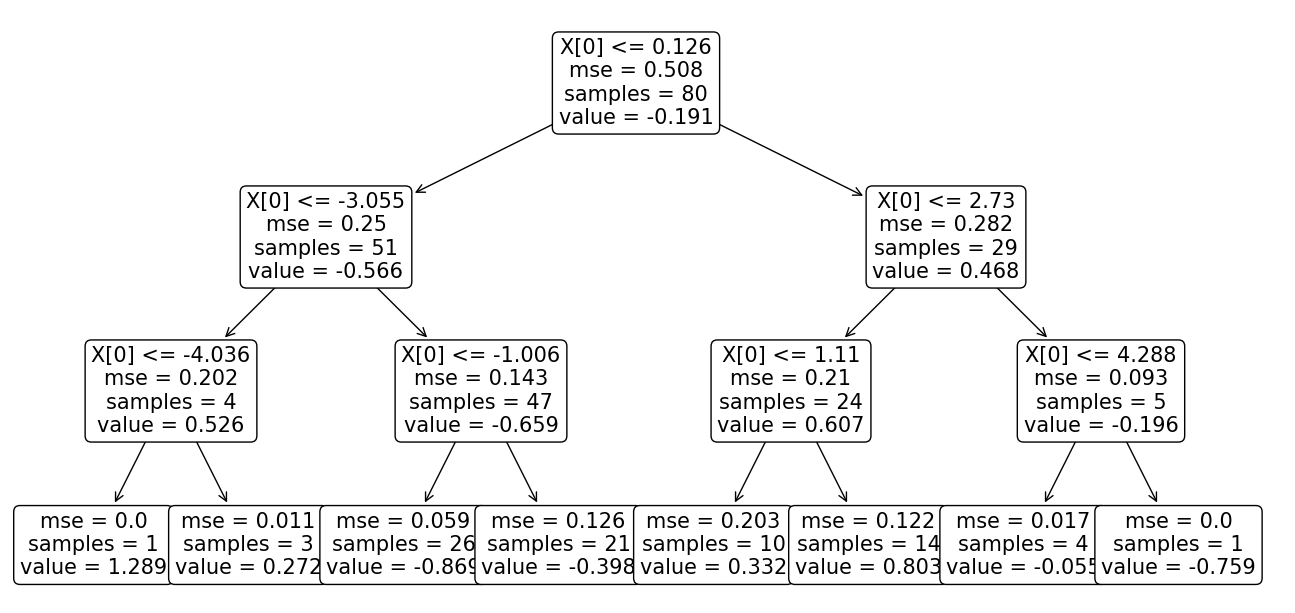
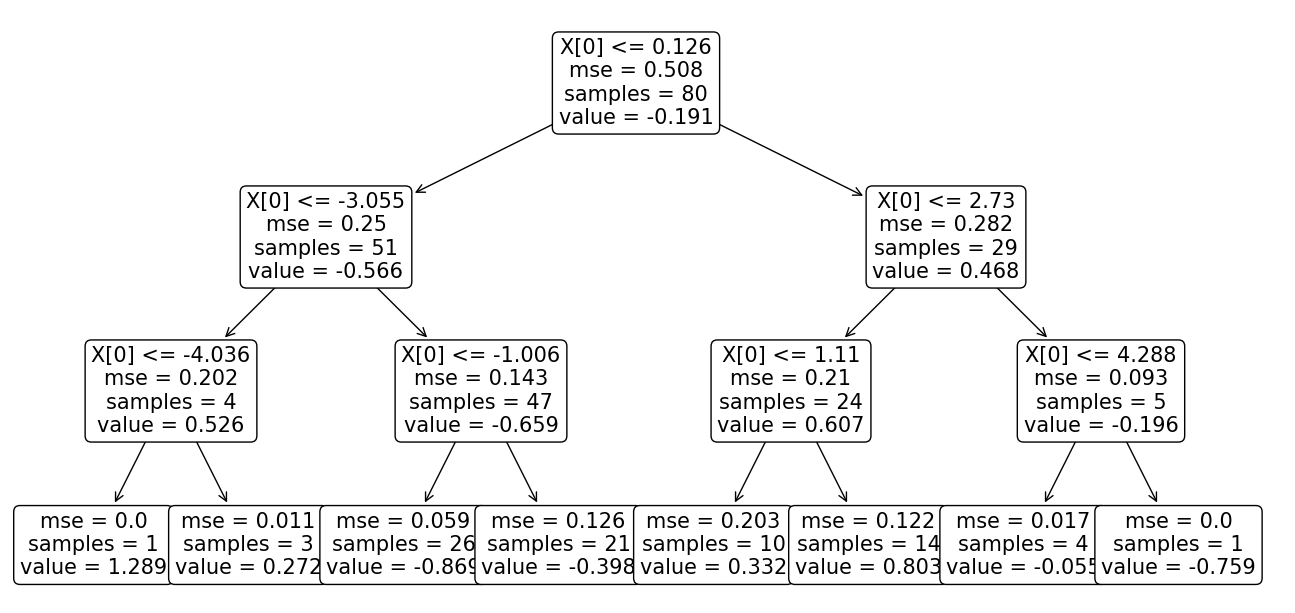
**mae**: Mean Absolute Error (L1 loss) 텍스트, 손목시계이(가) 표시된 사진

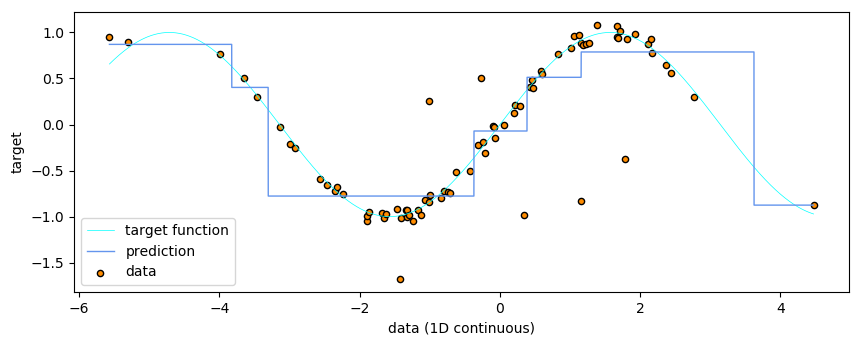
자동 생성된 설명

**Poisson**: Half Poisson Deviance 텍스트, 손목시계이(가) 표시된 사진

자동 생성된 설명  
**friedman\_mse**: <https://stackoverflow.com/questions/55113978/what-is-the-difference-between-freidman-mse-and-mse>

Splitter=”best”인 경우, 각각 feature마다 모든 샘플 정렬 후 linear scan하는 방식으로 가장 낮은 mse값의 분할조건이 선택됨.



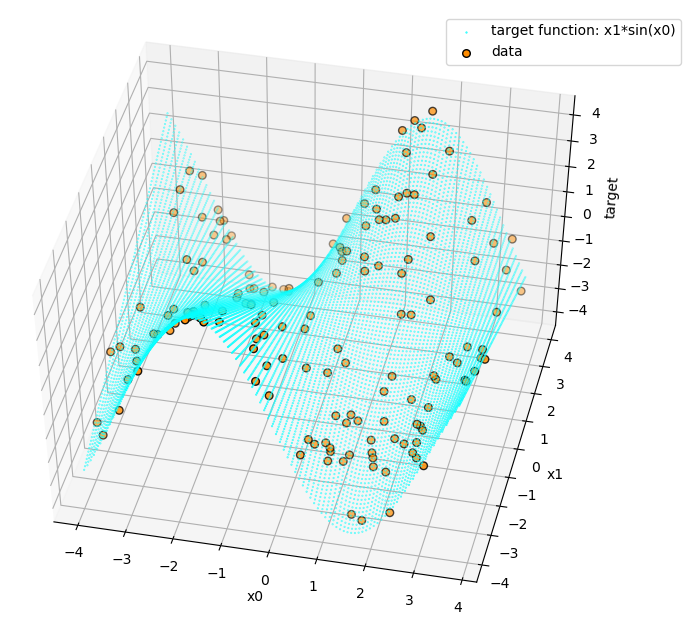
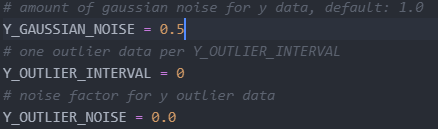


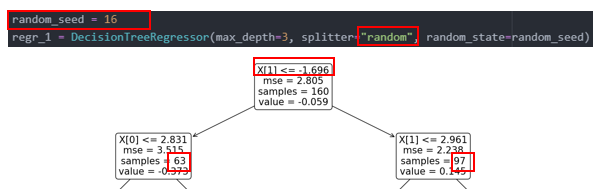
**Splitter:** {“best”, “random”}, default: “best”

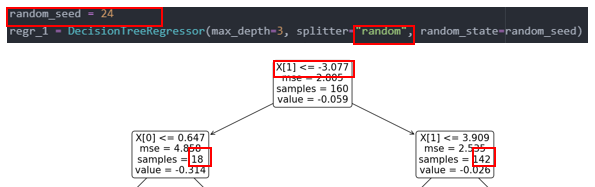
Best: 모든 분할조건을 탐색 후 최적의 loss를 내는 분할조건을 선택

Random: 최적의 loss를 내는 분할 조건 중에서 랜덤하게 (uniform distribution) 선택

[**https://stackoverflow.com/questions/46756606/what-does-splitter-attribute-in-sklearns-decisiontreeclassifier-do**](https://stackoverflow.com/questions/46756606/what-does-splitter-attribute-in-sklearns-decisiontreeclassifier-do) **참조!**







Comment:

random으로 설정할 시 overfitting에 대응하는데 도움이 된다고 알려짐.

**max\_features:** {“auto”, ”sqrt”, ”log2”} OR int OR float default: “auto”

노드를 분할하기 위한 최적의 조건을 찾을 때, 고려할 features (dimension)의 개수 (랜덤하게 선택)

예) 데이터Xi = (x1, x2, x3, x4), 그리고 max\_feautes = 2로 설정했을 때,

노드 A에서는 랜덤하게 선택한 x1, x2 features 내에서만 분할조건을 검색하고,

노드 B에서는 랜덤하게 선택한 x2, x4 features 내에서만 분할조건을 검색하고,

노드 C에서는 랜덤하게 선택한 x1, x4 features 내에서만 분할조건을 검색

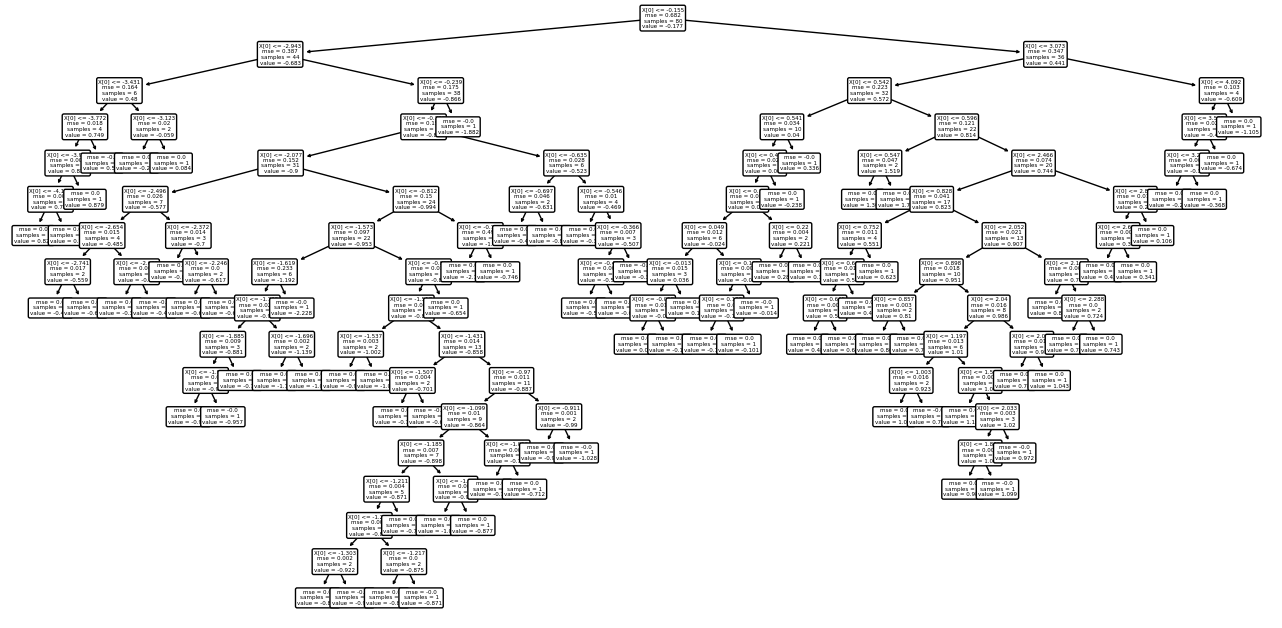
Comment:

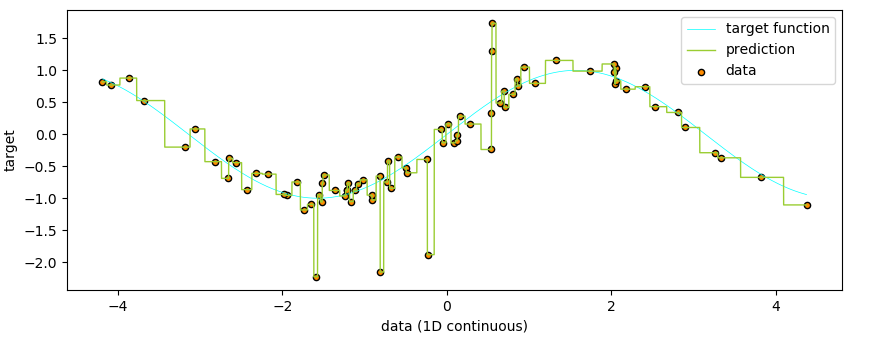
max\_features값이 낮을수록 모델의 variance가 낮아지고, bias가 높아지는 효과.

Regression tree의 경우에는 모든 features를 다 고려하는 것이 추천됨. (“auto”)

**Max\_depth:** positive integer {1, 2, 3, …, None}, default: None

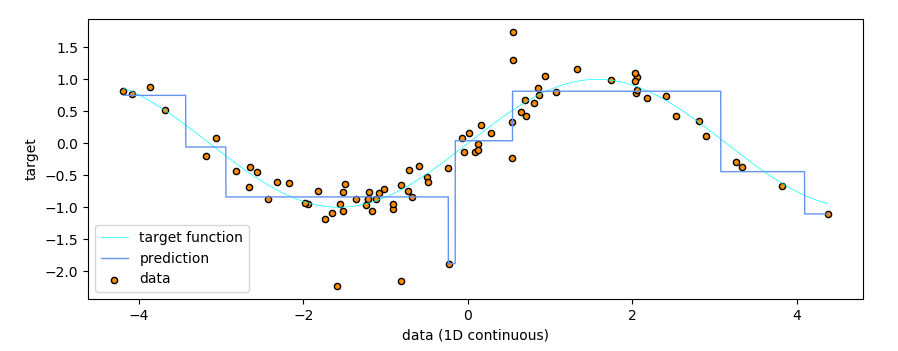
Tree의 최대 허용 가능 깊이



****

텍스트, 영수증이(가) 표시된 사진

자동 생성된 설명

****

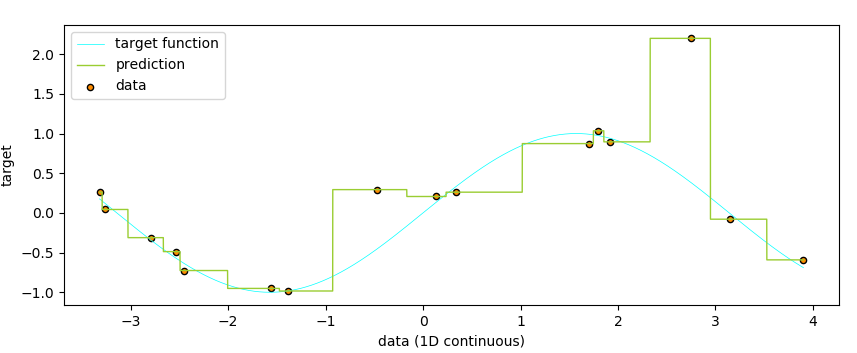
Comment: 위 그래프에서 확인할 수 있듯, max\_depth만 이용해서는 Regularization이 어려움. min\_samples\_split & min\_samples\_leaf의 값을 조합해야 함.

**min\_samples\_split:** positive integer {>= 2}, default: 2

혹은: positive float {0.0~1.0} (parent\_node에 대한 ratio)

**Internal node**의 분할을 진행하기 위해 필요한 최소 샘플 수. 특정 노드에 샘플 숫자가min\_samples\_split보다 적으면, recursion이 종료되고 그 노드는 자동으로 **leaf**가 됨. *(주의: min\_samples\_split을 어떤 숫자로 지정하든 leaf의 샘플 수가 1이 될 수 있음)*

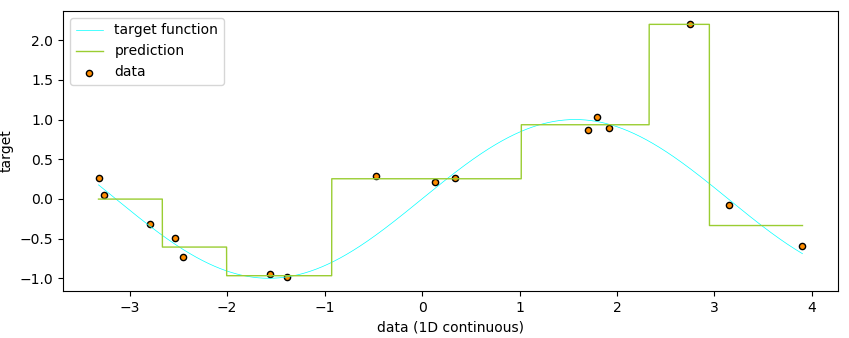


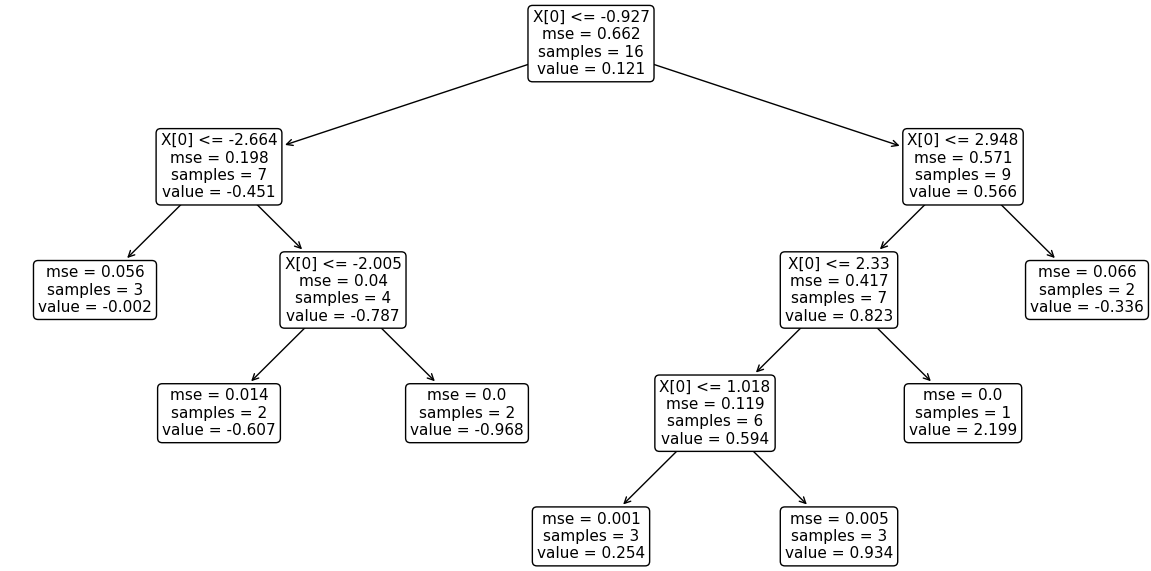


텍스트, 낱말맞추기게임, 영수증이(가) 표시된 사진

자동 생성된 설명





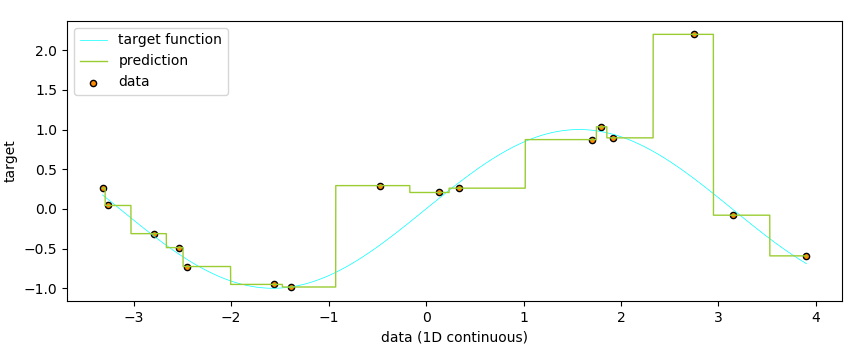


**min\_samples\_leaf:** positive integer (>=1), default: 1

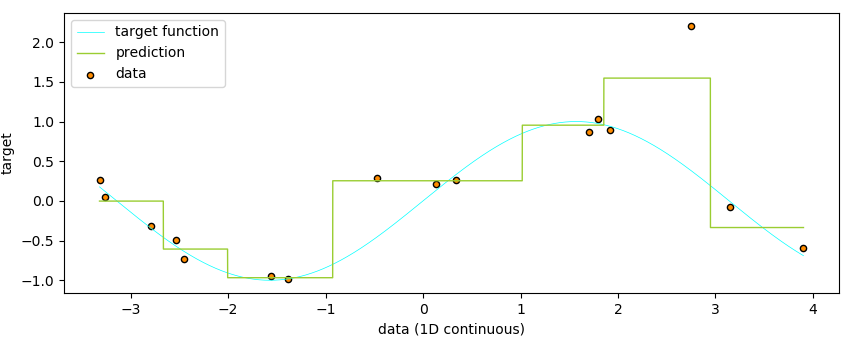
혹은: positive float {0.0~1.0} (parent\_node에 대한 ratio)

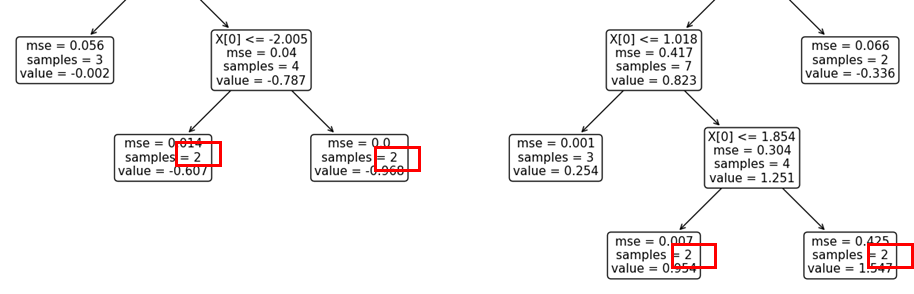
internal node에서 left child node, right child node로 분할을 하려고 할 때, left child node 혹은 right child node 둘 중에 한 노드 이상이 가지고 있는 샘플 수가 min\_samples\_leaf보다 작아지게 되면 그 분할을 진행하지 않음.











comment: 경험적으로 **min\_samples\_leaf = 2**로 설정하는 것이 가장 효과가 좋다고 알려짐. 주의: 데이터가 Imbalanced한 경우에는 특정 구역의 샘플 수가 굉장히 적은 경우가 있어서 충분히 낮은 값으로 세팅해야 함.

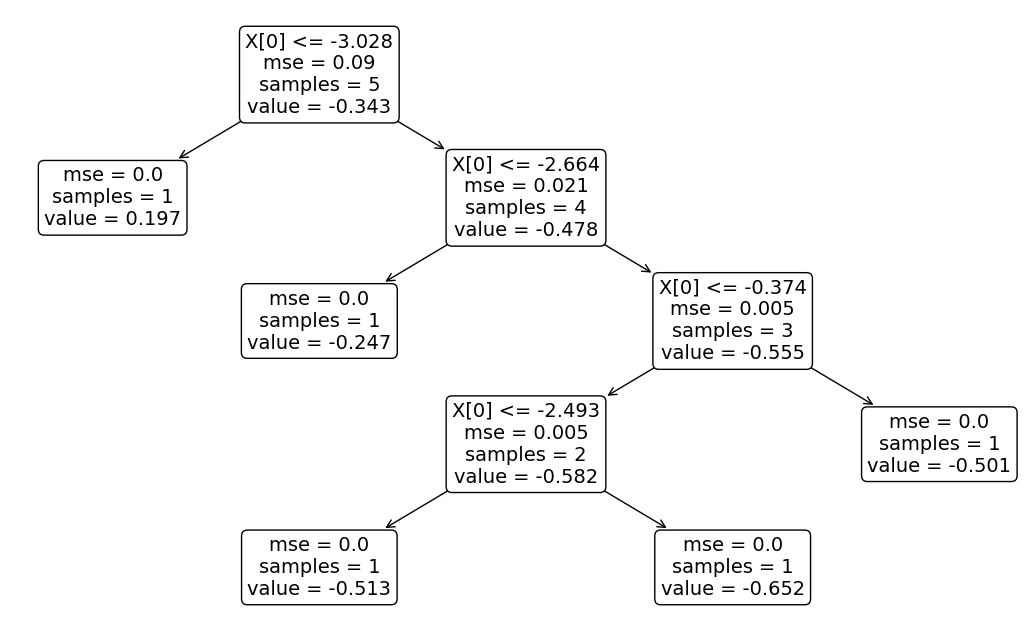
**min\_weight\_fraction\_leaf:** positive float, default: 0.0

leaf가 되기 위한 최소 (샘플 수) x (각 샘플 가중치의 합). 이 parameter를 사용하기 위해서는 각각의 데이터에 가중치가 따로 설정되어 있어야 함.

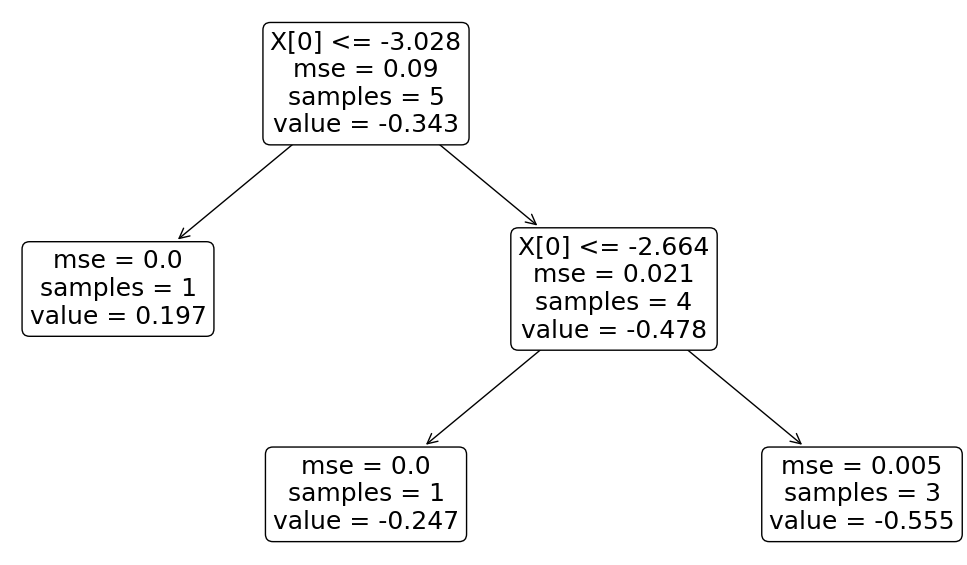
**min\_impurity\_split:** float, default: 0.0

노드 분할을 진행하기 위한 최소 impurity(loss) threshold값. impurity값이 보다 낮으면 그 노드는 leaf노드가 됨.









Comment:

i.i.d 조건 하에서 impurity(depth=**N**) > impurity(depth=**N+1**)를 만족함.

하지만, 각 feature 사이에 높은 차원의 연관성이 존재할 경우에

impurity(depth=**N**) < impurity(depth=**N+1 |** depth=**N**)

인 경우도 발생하기 때문에, min\_impurity\_split으로 regularization을 하는 것은 권장되는 기법이 아님. Parameter 값 Tuning도 어려움!!!!

**Sklearn.rfr에서 Deprecated됨 & 혹은 될 예정**

**~~ccp\_alpha:~~** ~~positive float, default: 0.0~~

[https://scikit-learn.org/stable/modules/tree.html#minimal-cost-complexity-pruning](https://scikit-learn.org/stable/modules/tree.html" \l "minimal-cost-complexity-pruning)

[https://scikit-learn.org/stable/auto\_examples/tree/plot\_cost\_complexity\_pruning.html#sphx-glr-auto-examples-tree-plot-cost-complexity-pruning-py](https://scikit-learn.org/stable/auto_examples/tree/plot_cost_complexity_pruning.html" \l "sphx-glr-auto-examples-tree-plot-cost-complexity-pruning-py)

Decision Tree를 완성한 이후에 진행하는 Pruning에 사용되는 값.

**Classification 전용.**

**Random Forest Regressor** parameters

[https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.RandomForestRegressor.html#sklearn.ensemble.RandomForestRegressor](https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.RandomForestRegressor.html" \l "sklearn.ensemble.RandomForestRegressor)

**Ensemble (Bagging) 관련:**

n\_estimators: positive integer

bootstrap: Boolean

max\_samples: positive integer OR positive float

**Regularization 관련:**

~~min\_impurity\_split: deprecated!!!~~

min\_impurity\_**decrease**: float

**속도 최적화 관련:**

n\_jobs: int

**Metrics 관련:**

oob\_score: Boolean

**그 외 나머지 parameter는 위 Decision Tree parameters 참조.**

**bootstrap:** boolean, default: True

**max\_samples:** integer or float, or None, default: None

Bootstrap (stochastic subsampling with replacement) 단계에서 데이터셋에서 데이터를 하나씩 랜덤하게 max\_samples 번 만큼 뽑음.

<https://www.youtube.com/watch?v=J4Wdy0Wc_xQ> 여기에 쉽게 설명되어 있음.

**min\_impurity\_decrease:** float, default: 0.0

텍스트이(가) 표시된 사진

자동 생성된 설명

위에서 min\_impurity\_split에 대해 기술된 대로, min\_impurity\_split은 효과적이지도 않고 parameter값 제어도 굉장히 어려움. 그에 반해서 min\_impurity\_decrease는 impurity값을 기준으로 모델을 더 안정적으로 regularization할 수 있음. Parameter값 튜닝도 상대적으로 쉬운 편.

**min\_impurity\_decrease:** Boolean, default: False

성능평가를 위해 out of bag score를 학습과 함께 계산할지에 대한 여부.

out of bag samples: bootstrap data에서 사용되지 않은 샘플. *(예, 데이터셋 d1, d2, d3에서 3번 random sampling with replacement을 하면 d1, d1, d1 이렇게 동일 데이터가 세번 뽑힐 가능성이 있고, 이 경우에 out of bag samples: {d2, d3}가 됨.)*

**K-Cross validation과 비슷한 개념으로 생각할 수 있음.**

**Gradient Boosting Regressor** parameters

<https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.GradientBoostingRegressor.html>

**Ensemble (Boosting) 관련:**

n\_estimators: positive integer

init: {estimator object, “zero”, None}

loss: {“ls”, “lad”, “huber”, “quantile”}

learning\_rate: float

**Regularization 관련:**

~~min\_impurity\_split: deprecated!!!~~

min\_impurity\_**decrease**: float

subsample: positive float {0~1}

**학습 종료 조건 관련:**

validation\_fraction: positive float {0~1}

n\_iter\_no\_change: positive int or None

tol: float

**그 외 나머지 parameter는 위 Decision Tree parameters 참조.**

**init:** {estimator object OR “zero” OR None}, default: None

Tree를 학습시키기 전 초기 예측값. None은 모든 데이터의 y 평균값을 내게 됨.

**loss:** {“ls”, “lad”, “huber”, “quantile”}, default: ls

ls: least squares

lad: least absolute deviation

각 Boosting 단계의 Tree를 평가하는 데 쓰일 loss function

**subsample:** {0.0 ~ 1.0}, default: 1.0

각 Decision Tree를 학습시킬 때 랜덤하게 쓸 데이터셋의 비율. 1.0이면 모든 데이터를 사용함.

**validation\_fraction:** {0.0 ~ 1.0}, default: 1.0

validation데이터 비율

**XGBoost** & **LightGBM** parameters

[http://machinelearningkorea.com/2019/09/29/lightgbm-%ED%8C%8C%EB%9D%BC%EB%AF%B8%ED%84%B0/](http://machinelearningkorea.com/2019/09/29/lightgbm-파라미터/) 참조. (주의: classification을 목적으로 작성되었기 때문에 regression과 다른 점이 있음.)

L1 regularization & L2 regularization을 제외한 나머지는 대부분 Gradient Boosting Regression과 동일.

: d 차원의 continuous vector값

: continuous scalar값