UST 과학기술연합대학원대학교
UNIVERSITY OF SCIENCE & TECHNOLOGY

의사결정나무

www.ust.ac.kr

Contents

목 차

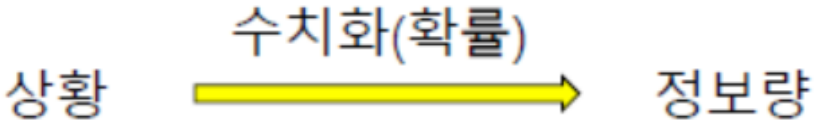
01.

결정나무 이론 소개

이홍석 (hsyi@kisti.re.kr)
www.ust.ac.kr

정보이론

- 상황 1)
  - 오늘 하루 종일 맑다. 뉴스에서 "내일은 맑다"라는 일기 예보를 듣는다.
- 상황 2)
  - 오늘 하루 종일 맑다. 뉴스에서 "내일은 큰 비가 온다"라는 일기 예보를 듣는다.
- 두 가지 상황을 비교해보자.
  - 어느 상황에서 더 많은 정보를 얻었을까?
  - Information Theory(정보 이론)는 주어진 상황에서 우리가 얻을 수 있는 정보량을 수치화해주는 기능을 제공한다.



자료: Jeonghun Yoon, Decision Tree (github)

정보이론

- 상황 1)
  - 오늘 하루 종일 맑다. 뉴스에서 "내일은 맑다"라는 일기 예보를 듣는다.
  - 직관적으로, 오늘 날씨가 맑기 때문에 내일의 날씨도 맑을 확률이 높다.
  - 이 상황에서 얻을 수 있는 정보(놀라운 정도)는 적다고 볼 수 있다.
- 상황 2)
  - 오늘 하루 종일 맑다. 뉴스에서 "내일은 큰 비가 온다"라는 일기 예보를 듣는다.
  - 직관적으로, 오늘 날씨가 맑기 때문에 내일 비가 올 확률은 상황 1에 비해 상대적으로 낮다.
  - 이 상황에서 얻을 수 있는 정보(놀라운 정도)는 상황 1에 비해 상대적으로 높다.

자료: Jeonghun Yoon, Decision Tree (github)

정보이론

- 정보이론(Information Theory)의 가장 중요한 원리
  - 어떤 사건의 확률과 그것이 전달하는 정보량은 반비례 관계임
  - 정보량은 놀라운 정도라고 할 수 있다.
- 일반적으로 확률이 낮은 사건일수록 더욱 놀랍고 정보량은 크다.
- 그러면, 정보량을 어떻게 수치화 할 수 있을까?

$$p(\text{내일} = \text{큰비} \mid \text{오늘} = \text{맑음})$$
$$p(\text{내일} = \text{맑음} \mid \text{오늘} = \text{맑음})$$

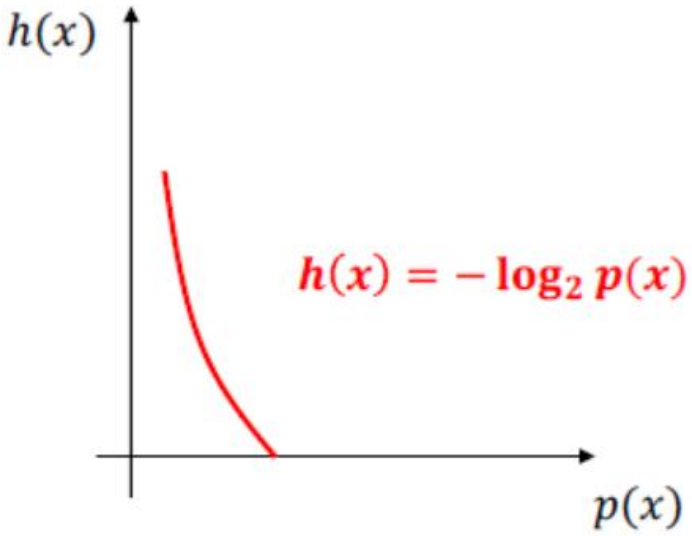
자료: Jeonghun Yoon, Decision Tree (github)

정보량

x : random variable(랜덤 변수)

$p(x)$: x가 특정한 값을 가질 때, 랜덤 변수 x의 확률

$h(x)$: x의 자기 정보량(self-information)



$h(x)$의 단위는 bit이다.

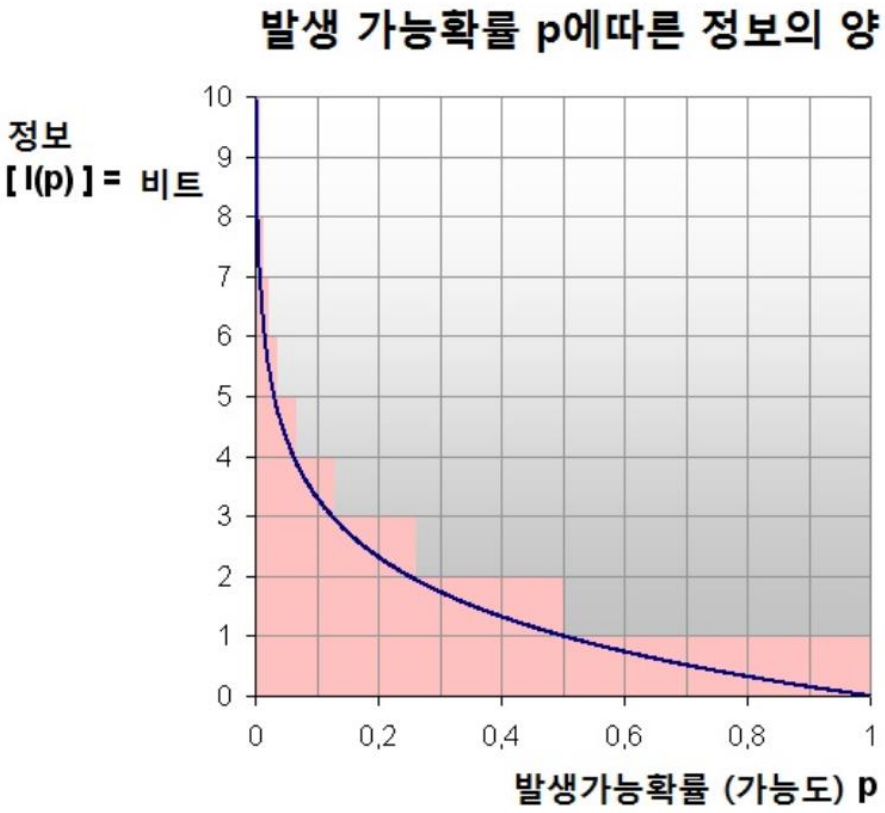
$p(a) = \frac{1}{1024}$ 라고 하자.

이 때 a의 정보량은 10bits가 된다.

자료: Jeonghun Yoon, Decision Tree (github)

정보량 (불확실성, 엔트로피)

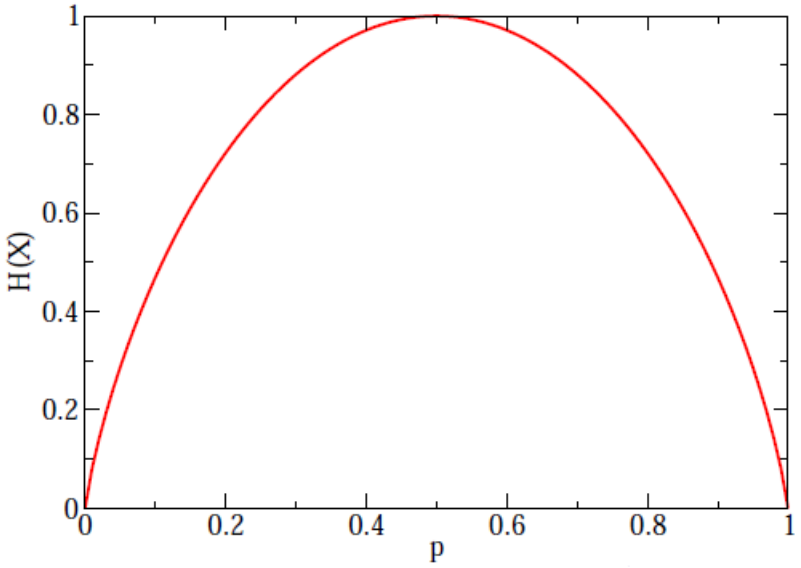
확률이 높을 수록 우리에게 별 도움이 안된다. (정보량이 낮다)



정보량 (불확실성, 엔트로피)

정보량 Quantities of information

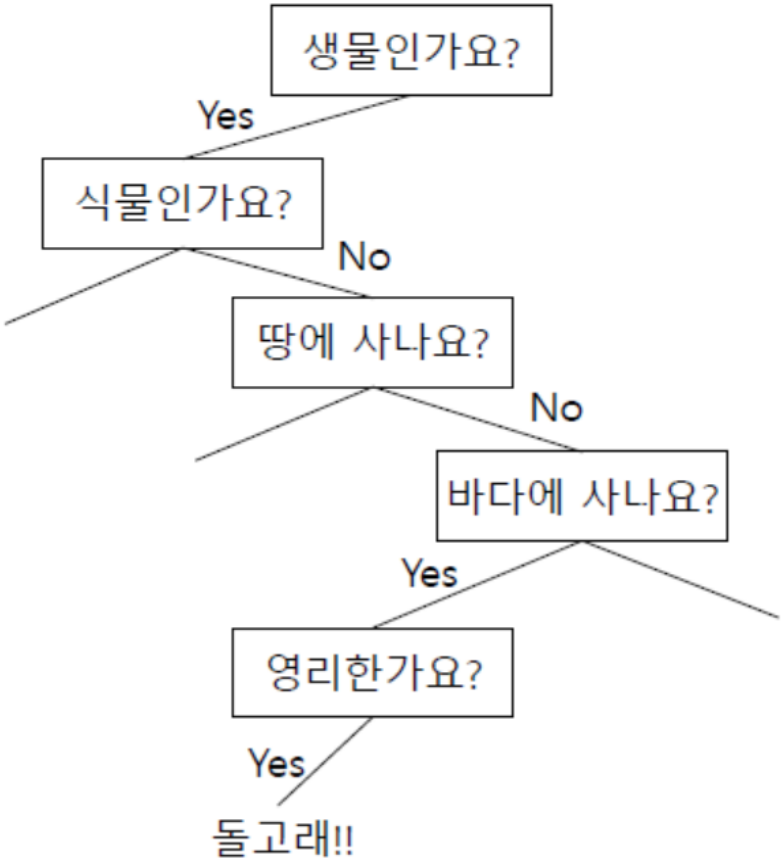
- $P(x)$: x의 확률
- $H(x) = \frac{1}{log_2 P(x)} = -log_2 P(x)$



불확실성 H(x)는 확률이 50% 일때
가장 높다. 이때, 엔트로피가 가장 크다.

의사 결정나무

스무고개!



자료: Jeonghun Yoon, Decision Tree (github)

Decision tree의 원리

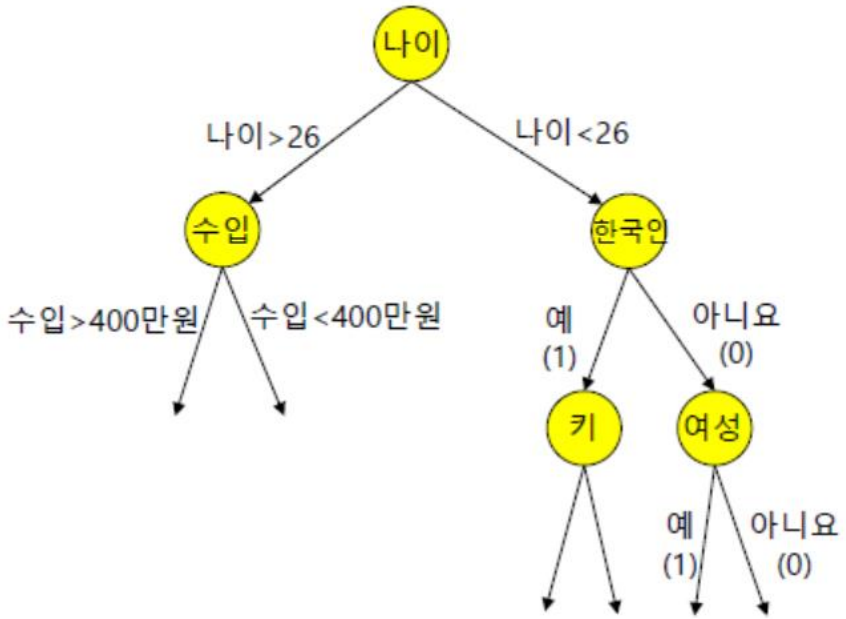
- 결정 트리는 계층구조를 가진 트리를 이용하여 데이터를 분류하는 분류기이다.
- 결정 트리는 스무고개와 유사한 원리를 사용한다.
  - 스무 고개는 사람(진행자)이 그 때마다 문제를 만들어낸다.
  - 결정 트리는 컴퓨터가 자동으로 질문을 생성해내야 한다.

- Question
  - 각 노드의 질문을 어떻게 만들 것인가?
  - 노드에서 몇 개의 가지로 나눌 것인가? (자식노드의 개수)
  - 언제 멈출 것인가? (leaf node)
  - leaf node를 어느 부류로 할당할(assign) 것인가?

자료: Jeonghun Yoon, Decision Tree (github)

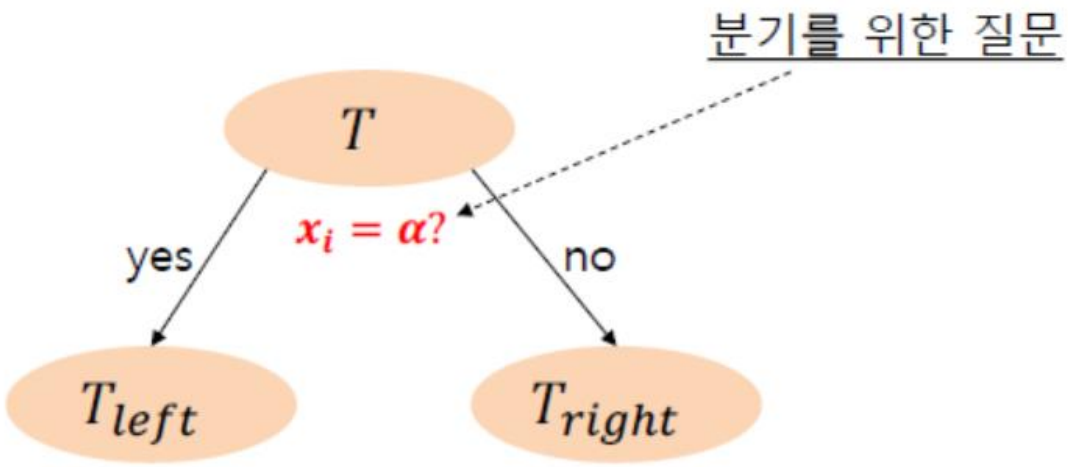
Decision tree의 구조

- internal node : attribute or feature (속성 or 특징)
- leaf node : classification 결과
- edge : assignment (or 조건)



자료: Jeonghun Yoon, Decision Tree (github)

각 노드의 질문을 어떻게 만들 것인가?



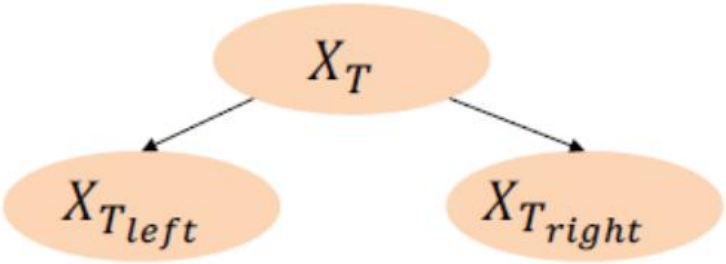
후보의 개수가 지수적이지 않기 때문에 모든 후보를 평가하여 가장 좋은 것을 선택(exhaustive search)

가장 좋은 것다는 것의 판단 기준은?
후보 질문을 평가하기 위한 기준 함수?

자료: Jeonghun Yoon, Decision Tree (github)

기준이 되는 함수

노드 T에 속하는 샘플 X_T에는 여러 부류에 속하는 샘플들이 혼재되어 있다.



분기를 반복하면 결국 leaf node에 도달하게 될 것이고, leaf node에는 같은 부류에 속하는 샘플들이 대부분이어야 한다.

따라서 X_T의 분기의 결과인 $X_{T_{left}}$와 $X_{T_{right}}$에는 각각 최대한 동질의 샘플이 담기는 것이 좋다. 동질의 샘플은 같은 부류에 속하는 샘플이라는 것이다.

동질성을 측정하는 기준 함수를 만들어

자료: Jeonghun Yoon, Decision Tree (github)

Impurity (불순도)

- 동질성을 측정하는 기준으로 Impurity (불순도):
  - 같은 부류에 속하는 샘플이 많을수록 불순도는 낮고,
  - 다른 부류에 속하는 샘플이 많이 섞여 있으면 불순도는 높다.
  - 불순도가 낮으면, 특정 부류에 속한다고 예측하기가 쉽다.
  - 불순도가 높으면, 특정 부류에 속한다고 예측하기가 어렵다.
- 불순도를 정의하는 방법
  - Entropy
  - Gini impurity (지니 불순도)

자료: Jeonghun Yoon, Decision Tree (github)

Entropy (노드 분기)

$$im(T) = -\sum_{i=1}^{M} P(y_i|T) \log_2 P(y_i|T)$$

엔트로피는 M개의 부류가 같은 확률을 가질 때 가장 큰 값을 가짐

- 같은 확률을 가진다는 의미는, 한 노드에 다른 부류에 속한 샘플들이 같은 빈도로 나타나는 것을 의미한다.
- 특정 샘플이 어떤 부류에 속하는지 예측하기 어렵다. ⇒ 불순도가 높다.

자료: Jeonghun Yoon, Decision Tree (github)

Gini impurity (노드 분기)

- 불순도를 정의하는 또 다른 방법으로 지지 불순도
  - **잘못된 분류를 측정하는 도구**로, 다부류 분류기에 적용된다.
  - (부류가 2개 이상인)
  - 엔트로피와 거의 동일하지만 훨씬 더 빨리 계산할 수 있다.

$$im(T) = 1 - \sum_{i=1}^{M} P(y_i|T)^2 = \sum_{i \neq j} P(y_i|T)P(y_j|T)$$

자료: Jeonghun Yoon, Decision Tree (github)

Cart 알고리즘

- Cart (Classification And Regression Tree)의 약자
  - Binary decision tree 를 만들 때 사용하는 알고리즘이다.
  - 앞에서 다루었던, 동질성을 체크하는 기준 함수를 이용하여 parent node 에서 child node 로 split분기 한다.
  - 기준 함수로는 entropy의 정보 이득 또는 gini impurity 값을 사용

자료: Jeonghun Yoon, Decision Tree (github)

Decision tree의 특성 요약

결정 트리의 특성

① 단순하게 문장으로 규칙을 만들 수 있어서, 상관에게 설명이 용이

② 결정 트리는 비-파라미터 모델로, 훈련되기 전에 파라미터 수가 결정되지 않는다.

③ 기저 분포에 관한 어떠한 가정도 하지 않는다.

④ 모델의 모양이 미리 정해지지 않고 최적 분류로 학습된다.

⑤ 모든 변수가 범주형(categorical) 변수 일 때 가장 잘 작동한다.

⑥ 파라미터 사이의 비선형 관계가 트리의 성능에 영향을 주지 않고 수치 데이터도 잘 처리한다.

⑦ Outlier 또는 missing value 를 잘 처리한다.

자료: Jeonghun Yoon, Decision Tree (github)

결정나무 핵심 요약

- **중요한 특징**
  - Non-paramentric 지도학습 (최소한의 하이퍼-파라미터)
  - 분류 문제 적용 가능
  - 회귀 문제도 적용 가능
- **목적은**
  - 데이터로 부터 유추되는 간단한 결정 규칙을 훈련하여 목적 값을 예측하는 모델을 만드는 것

결정트리의 장점과 단점

- 결정트리의 이점
  - 이해가 쉽다.
  - 데이터 정규화 과정 없이 적은 데이터를 사용할 수 있다.
  - 수치데이터와 범주화 데이터 둘 다 사용 가능
  - 멀티 출력을 다룰 수 있다.
- 결정트리의 단점
  - 오버피팅을 야기할 수 있다.
  - 데이터의 차이는 생성해야 할 전혀 다른 결정나무로 만들 수 있다.

02. IRIS

결정나무로 분류 문제에 적용해보다.

이홍석 (hsyi@kisti.re.kr)
www.ust.ac.kr

결정트리로 붓꽃(IRIS) 분류

붓꽃(IRIS) 데이터를 결정트리로 분류해보자! ¶

```
from sklearn.datasets import load_iris

from sklearn import tree
model = tree.DecisionTreeClassifier(max_depth=5)

iris = load_iris()

from sklearn.model_selection import train_test_split

trainX, testX, trainY, testY = train_test_split(iris.data, iris.target)
```

자료: Jeonghun Yoon, Decision Tree (github)

결정트리로 붓꽃(IRIS) 분류

```
model.fit(trainX, trainY)
```

```
DecisionTreeClassifier(ccp_alpha=0.0, class_weight=None, criterion='gini',
                       max_depth=5, max_features=None, max_leaf_nodes=None,
                       min_impurity_decrease=0.0, min_impurity_split=None,
                       min_samples_leaf=1, min_samples_split=2,
                       min_weight_fraction_leaf=0.0, presort='deprecated',
                       random_state=None, splitter='best')
```

```
model.score(testX,testY)
```

```
1.0
```

자료: Jeonghun Yoon, Decision Tree (github)

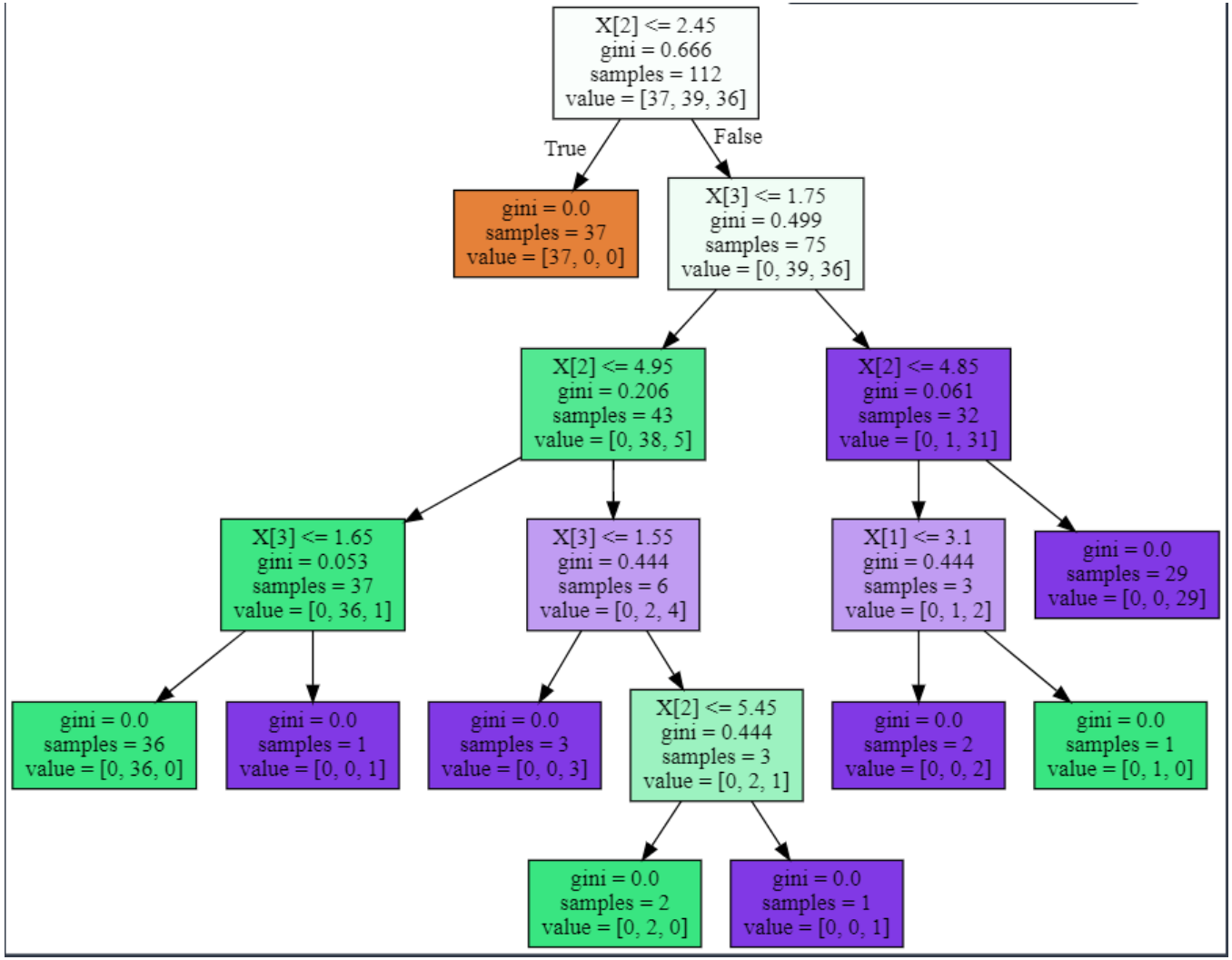
결정트리로 붓꽃(IRIS) 분류

```
import graphviz
graph = tree.export_graphviz(model, out_file=None, filled=True)
graphviz.Source(graph)
```

```
import graphviz, pydotplus
graph = tree.export_graphviz(model, out_file=None, filled=True)

pydot_graph = pydotplus.graph_from_dot_data(graph)
pydot_graph.set_size('"5,5!"')
graphviz.Source(pydot_graph.to_string())
```

자료: Jeonghun Yoon, Decision Tree (github)

결정트리로 붓꽃(IRIS) 분류



03. Boston house price

결정나무로 회귀 문제에 적용해보다.

이홍석 (hsyi@kisti.re.kr)
www.ust.ac.kr

결정트리로 보스턴 주택 가격 회귀

보스턴 주택을 회귀문제로 풀어보자 ¶

```
from sklearn.datasets import load_boston
```

```
boston = load_boston()
```

```
boston.feature_names
```

```
array(['CRIM', 'ZN', 'INDUS', 'CHAS', 'NOX', 'RM', 'AGE', 'DIS', 'RAD',
       'TAX', 'PTRATIO', 'B', 'LSTAT'], dtype='<U7')
```

결정트리로 붓꽃(IRIS) 분류

```
from sklearn import tree
model2 = tree.DecisionTreeRegressor(max_depth=4)
```

```
trainX, testX, trainY, testY = train_test_split(boston.data, boston.target)
```

```
model2.fit(trainX, trainY)
```

```
DecisionTreeRegressor(ccp_alpha=0.0, criterion='mse', max_depth=4,
                      max_features=None, max_leaf_nodes=None,
                      min_impurity_decrease=0.0, min_impurity_split=None,
                      min_samples_leaf=1, min_samples_split=2,
                      min_weight_fraction_leaf=0.0, presort='deprecated',
                      random_state=None, splitter='best')
```

결정트리로 붓꽃(IRIS) 분류

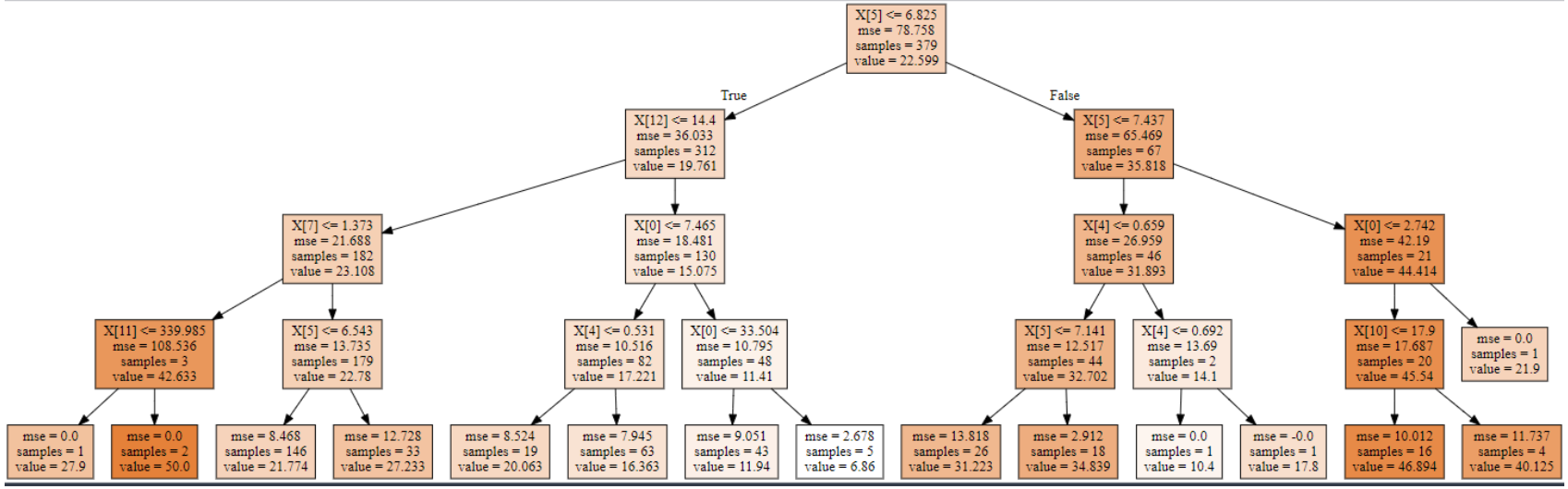
```
graph = tree.export_graphviz(model2, out_file=None, filled=True)
graphviz.Source(graph)
```

```
import graphviz, pydotplus
graph = tree.export_graphviz(model2, out_file=None, filled=True)
pydot_graph = pydotplus.graph_from_dot_data(graph)
pydot_graph.set_size('"9,9"')
graphviz.Source(pydot_graph.to_string())
```

```
model2.score(testX, testY)
```

```
0.7832363589796502
```

결정트리로 붓꽃(IRIS) 분류



04.

학생 60명중에 방과후에 축구하는 사람 분류

이홍석 (hsyi@kisti.re.kr)
www.ust.ac.kr

결정 트리 소개

1) 결정 트리 소개

결정 트리 용어

루트 노드 - 부모 노드로 알려짐. 데이터셋의 길이와 모든 가지가 여기서 출발 함.

브랜치 - 가지는 루트 노드의 서브 노드로 나눈다. 물론 데이터도 나눔

결정 노드 - 결정노드들은 서브노드를 더 깊게 나눔. 더 이상 나눌 것이 없으며 리프노느임.

리프노드: 더이상 나눌수 없는 노드.

결정 트리 소개

알고리즘 - '지니'(Gini)는 경제학에서 불평등 지수를 나타냄. 0이 가장 평등하며, 1이 불평등하다.

- 즉, 데이터가 다양한 값을 가질 경우 평등(0)하며
- 특정 값으로 쏠릴경우 불평등(1에 가까움)
- 엔트로피는 무질서도를 나타내며, 무질서도(혼잡도)는 서로 다른 값이 섞여 있으면 높다.
- 혼잡도가 높으면 1, 적으면 0

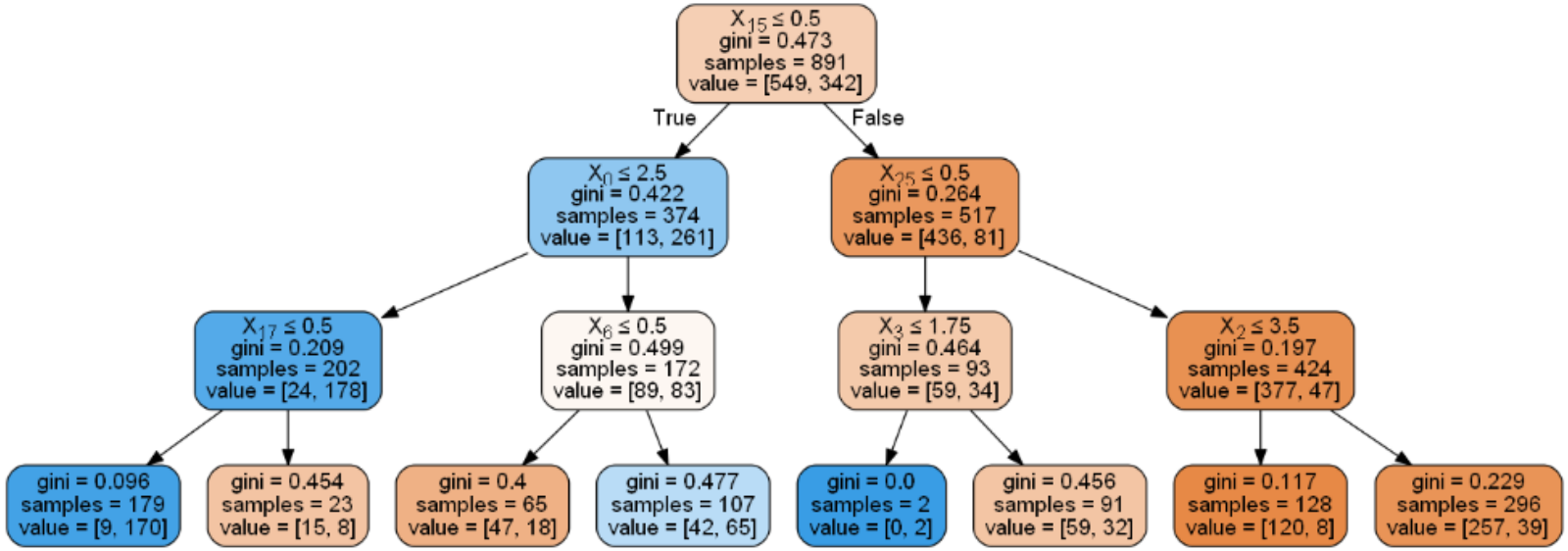
지니 지수: $G = 1 - \sum_{i}^{c} p_i^2 \quad , \quad 0 \leq G \leq 1/2$

엔트로피 지수: $E = -\sum_{i}^{c} p_i \log_2 p_i \quad , \quad 0 \leq E \leq 1$

결정트리 그래프 예제

루트노드, 리프노드, 가지노드, gini를 이해해보자

(예) 결정트리 : 타이타닉 생존자 예측



분류 문제 다루기 : 데이터 설명(1)

사용 데이터 설명 및 문제 설명

- 데이터는 60명이 학생이 있고,
- 성별 (M/F), 반 (IX/X), 키 (5/6 피트) 즉 150과 180cm, 몸무게 (50/58)
- 이중 30명은 방과후에 축구를 한다.
- 풀어햐 할 문제는 방과후에 누가 축구를 하는지 예측하라!

```
label = ['P', 'NP', 'P','NP', 'NP', 'NP', 'P','NP', 'NP', 'NP', 'NP','NP', 'P', 'NP', 'P','NP', 'P', 'NP', 'P','NP', 'P', 'NP', 'P','NP','P'

sex = ['M', 'M', 'M', 'M', 'M', 'M', 'M', 'M', 'M', 'M', 'M', 'M', 'M', 'M', 'M', 'F', 'M', 'F', 'M', 'F', 'M', 'F', 'M', 'F', 'M', 'F', 'F'

cls = ['IX', 'IX', 'IX', 'IX', 'IX', 'IX', 'IX', 'IX', 'IX', 'IX', 'IX', 'X', 'IX', 'X', 'X', 'X', 'X', 'X', 'X', 'X', 'X', 'X', 'X',

height = [5, 6, 5, 6, 5, 6, 5, 6,5, 6, 5, 6,6,6, 5, 5, 6, 5, 6, 5, 6, 5, 6,5, 6, 5, 6,6,6, 5,5, 6, 5, 6, 5, 6, 5, 6,5, 6, 5, 6,6,6, 5,5, 6,

weight = [50, 58, 50, 58, 50, 58, 50, 58, 50, 58, 50, 58, 50, 58, 50, 58, 50, 58, 50, 58,50, 58, 50, 58, 50, 58, 50, 58, 50, 58,50, 58, 50,

# Data frame created using the above list
df = pd.DataFrame({'Weight': weight, 'Height': height, 'Class': cls, 'Sex': sex, 'label': label})
```

결정나무 분류

```
# Data frame created using the above list
df = pd.DataFrame({'Weight': weight, 'Height': height,
                   'Class': cls, 'Sex': sex, 'label': label})
```

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 60 entries, 0 to 59
Data columns (total 5 columns):
 #   Column  Non-Null Count  Dtype
---  ------  --------------  -----
 0   Weight  60 non-null     int64
 1   Height  60 non-null     int64
 2   Class   60 non-null     object
 3   Sex     60 non-null     object
 4   label   60 non-null     object
dtypes: int64(2), object(3)
memory usage: 2.5+ KB
```

```
df.head()
```

	Weight	Height	Class	Sex	label
0	50	5	IX	M	P
1	58	6	IX	M	NP
2	50	5	IX	M	P
3	58	6	IX	M	NP
4	50	5	IX	M	NP

결정나무 분류

- 카테로리컬한 데이터를 숫자로 바꾸어라.
- 축구하면 (P=1), 안하면(NP)를 0
- new_label 변수로 1이면 방과후에 축구를 한 것

```
code = {'P': 1, 'NP': 0}
df['new_label'] = df['label'].map(code)
```

```
new_s = pd.get_dummies(df.Sex)
new_c = pd.get_dummies(df.Class)
df[new_s.columns] = new_s
df[new_c.columns] = new_c
```

결정나무 분류

df.get_dummies는 카타로그 데이터를 0과 1 값 만듬

```
df.head()
```

	Weight	Height	Class	Sex	label	new_label	F	M	IX	X
0	50	5	IX	M	P	1	0	1	1	0
1	58	6	IX	M	NP	0	0	1	1	0
2	50	5	IX	M	P	1	0	1	1	0
3	58	6	IX	M	NP	0	0	1	1	0
4	50	5	IX	M	NP	0	0	1	1	0

결정나무 분류

```
new_df = df[['F', 'M', 'IX', 'X', 'Weight', 'Height','new_label']]
new_df.head()
```

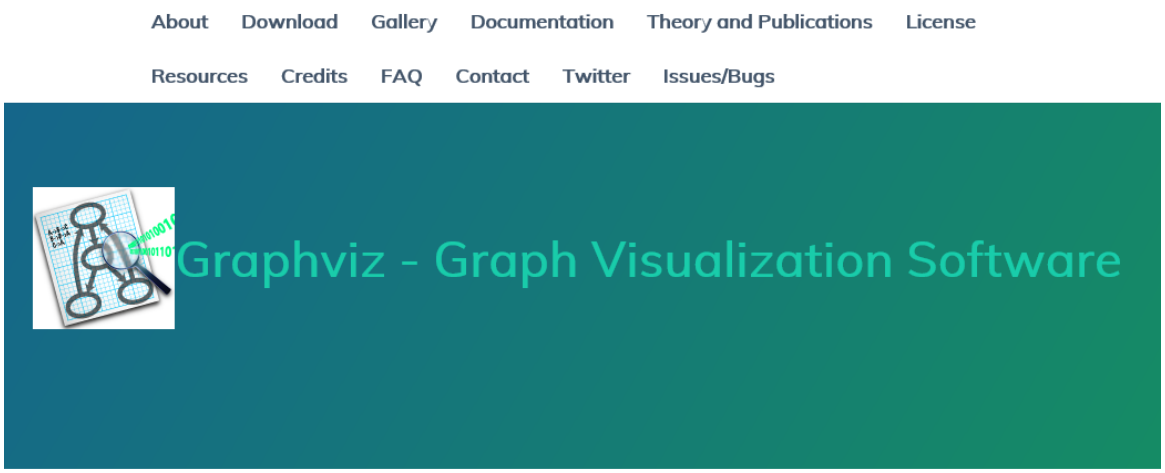
	F	M	IX	X	Weight	Height	new_label
0	0	1	1	0	50	5	1
1	0	1	1	0	58	6	0
2	0	1	1	0	50	5	1
3	0	1	1	0	58	6	0
4	0	1	1	0	50	5	0

결정나무 분류

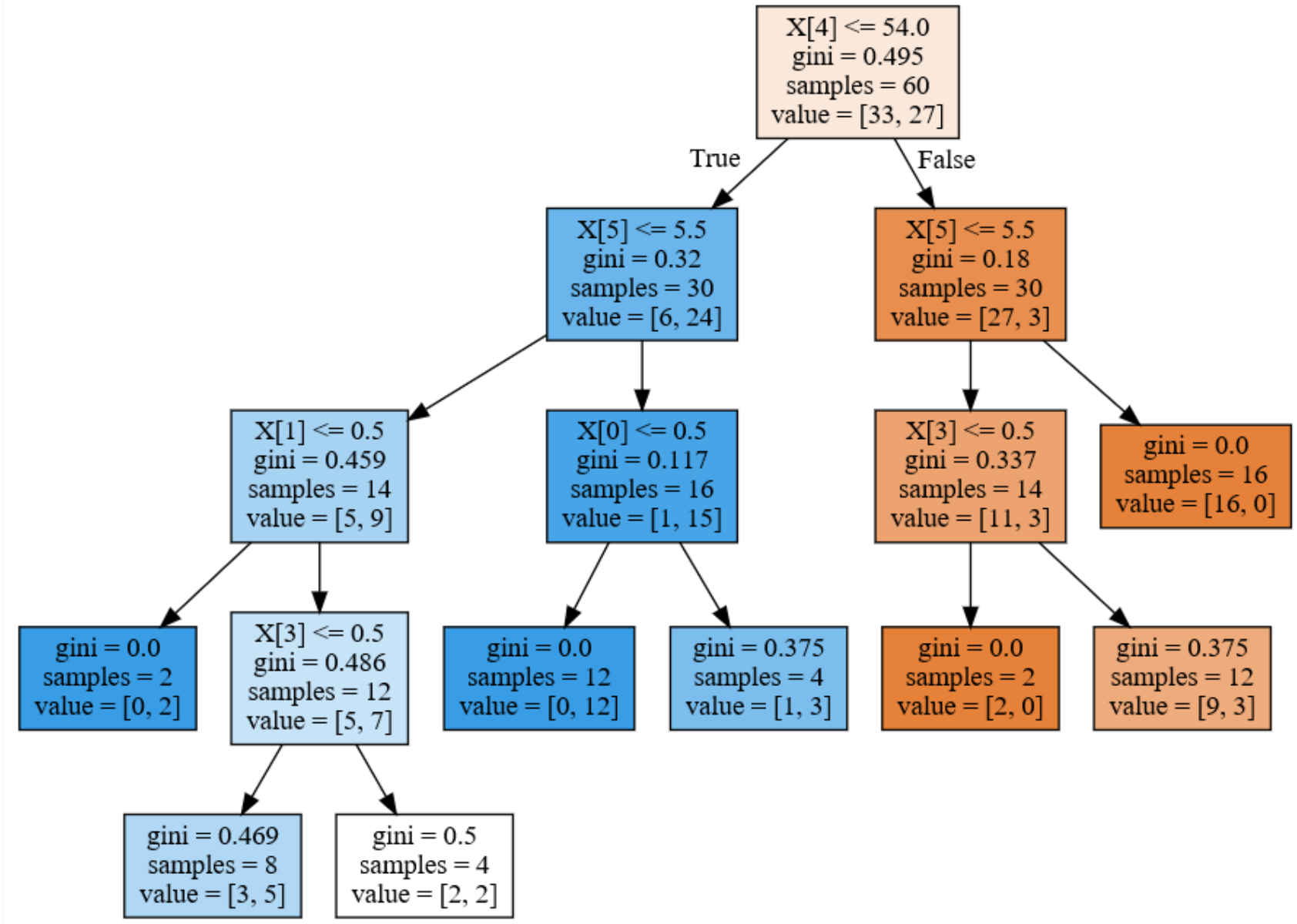
```
from sklearn import tree
model = tree.DecisionTreeClassifier(random_state = 23)
feature = new_df.drop(['new_label'], axis = 1)
label = new_df.new_label
model.fit(feature, label)
```

```
import graphviz
graph = tree.export_graphviz(model, out_file=None, filled=True)
graphviz.Source(graph)
```

결정나무 분류




```
import graphviz
graph = tree.export_graphviz(model, out_file=None, filled=True)
graphviz.Source(graph)
```


X[4] <= 54.0
gini = 0.495
samples = 60
value = [33, 27]
True
False
X[5] <= 5.5
gini = 0.32
samples = 30
value = [6, 24]
X[5] <= 5.5
gini = 0.18
samples = 30
value = [27, 3]
X[1] <= 0.5
gini = 0.459
samples = 14
value = [5, 9]
X[0] <= 0.5
gini = 0.117
samples = 16
value = [1, 15]
X[3] <= 0.5
gini = 0.337
samples = 14
value = [11, 3]
gini = 0.0
samples = 16
value = [16, 0]
gini = 0.0
samples = 2
value = [0, 2]
X[3] <= 0.5
gini = 0.486
samples = 12
value = [5, 7]
gini = 0.0
samples = 12
value = [0, 12]
gini = 0.375
samples = 4
value = [1, 3]
gini = 0.0
samples = 2
value = [2, 0]
gini = 0.375
samples = 12
value = [9, 3]
gini = 0.469
samples = 8
value = [3, 5]
gini = 0.5
samples = 4
value = [2, 2]

결정나무 분류

중요 변수의 이해

criterion: 지니 혹은 엔트로피 알고리즘 선택 'Gini' 혹은 'Entropy'.

max_depth: 위의 예제의 경우 4이다. 이 숫자가 크면 오퍼피팅 될 수 있다.

max_features: 최적의 분할을 고려할때 초대 피처 개수,

- 디폰트는 None으로 테이터 셋트를 사용하여 분할 수행

max_lead_nodes: 말단 리프 노드의 최대 갯수

min_samples_leaf: 말단 노드인 리프노드가 되기 위한 최소한의 샘플 데이터 수

min_samples_split: 노드를 분할 하기 위한 최소한의 샘플 데이터의 수.

- 과적합을 제어에 도움이 됨
- 디폴트는 2이고, 작게 설정할 수록 분할 노드가 많아져서 과적함 증가

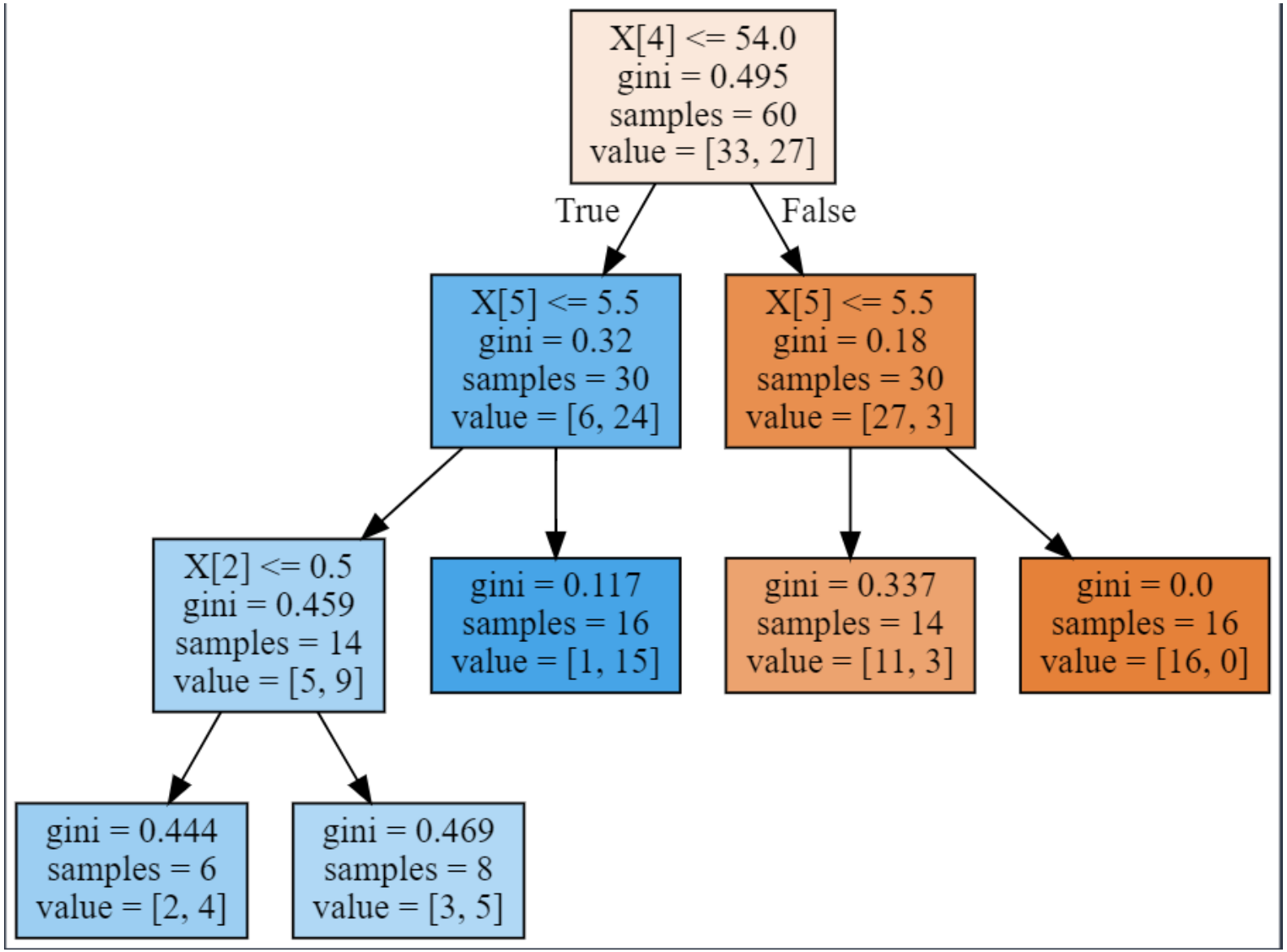
결정나무 분류

min_samples_leaf

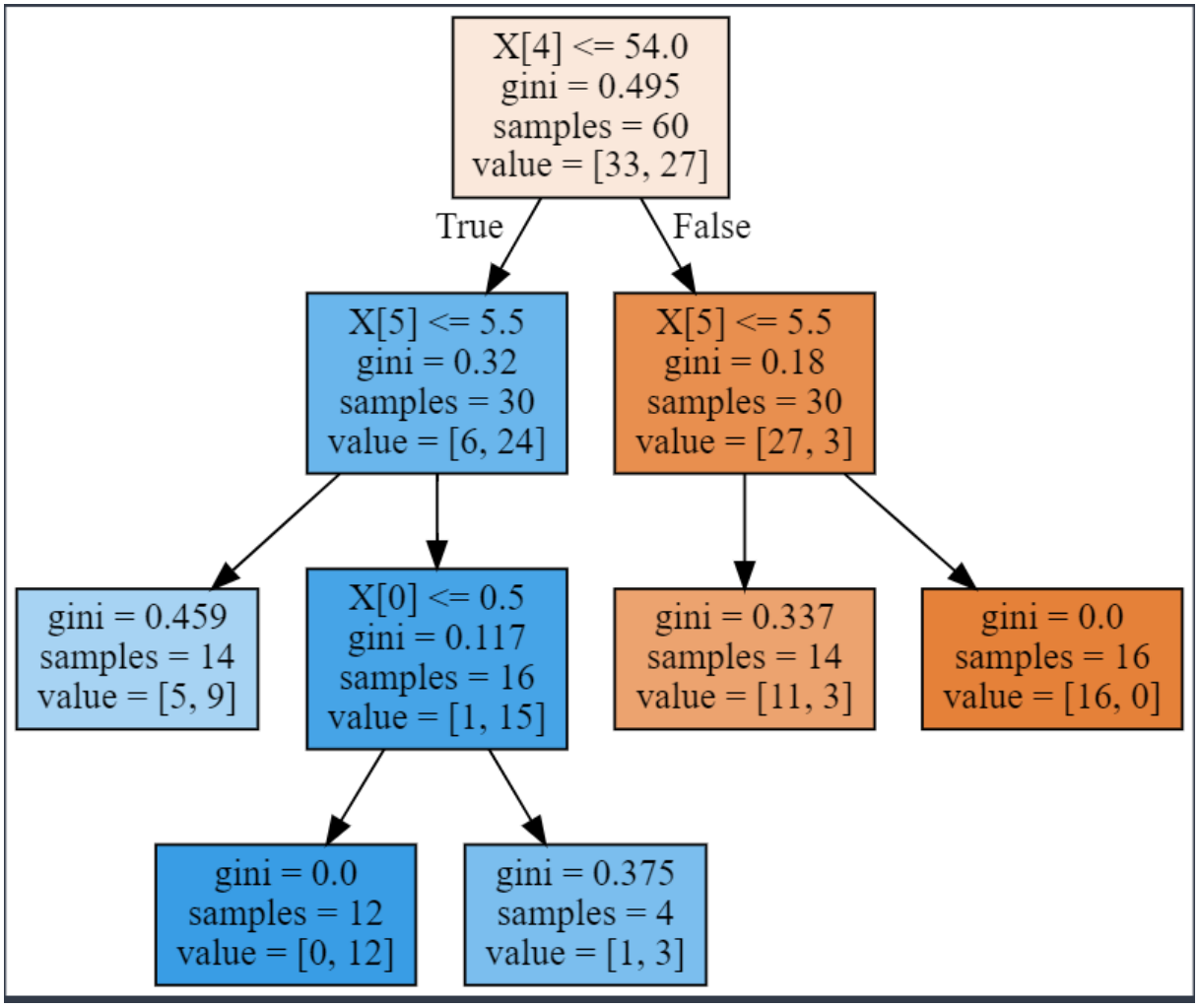
연습문제 min_samples_leaf 개수를 5로 했을때 차이점을 논의하시오

```
from sklearn import tree
model1 = tree.DecisionTreeClassifier(min_samples_leaf = 5,
                    random_state = 23)
model1.fit(feature, label)
```

```
DecisionTreeClassifier(ccp_alpha=0.0, class_weight=None, criterion='gini',
                       max_depth=None, max_features=None, max_leaf_nodes=None,
                       min_impurity_decrease=0.0, min_impurity_split=None,
                       min_samples_leaf=5, min_samples_split=2,
                       min_weight_fraction_leaf=0.0, presort='deprecated',
                       random_state=23, splitter='best')
```


X[4] <= 54.0
gini = 0.495
samples = 60
value = [33, 27]
True
False
X[5] <= 5.5
gini = 0.32
samples = 30
value = [6, 24]
X[5] <= 5.5
gini = 0.18
samples = 30
value = [27, 3]
X[2] <= 0.5
gini = 0.459
samples = 14
value = [5, 9]
gini = 0.117
samples = 16
value = [1, 15]
gini = 0.337
samples = 14
value = [11, 3]
gini = 0.0
samples = 16
value = [16, 0]
gini = 0.444
samples = 6
value = [2, 4]
gini = 0.469
samples = 8
value = [3, 5]

```
from sklearn import tree
model2 = tree.DecisionTreeClassifier(min_samples_split =16, random_state = 23)
model2.fit(feature, label)
```



결정나무 분류

연습문제 2. 그리드 서치를 통해 최적의 파라미터를 찾아라

Grid Search CV ¶

```
from sklearn.model_selection import GridSearchCV
param1 = {'min_samples_leaf': [2,3,4,5],
          'min_samples_split': [2,3,5,10,12,14],
          'max_depth': [2,3,4,5,6],
          'criterion': ['gini', 'entropy'],'max_features':[2,3,4]}
CV = GridSearchCV(model, param1)
CV.fit(feature, label)
```

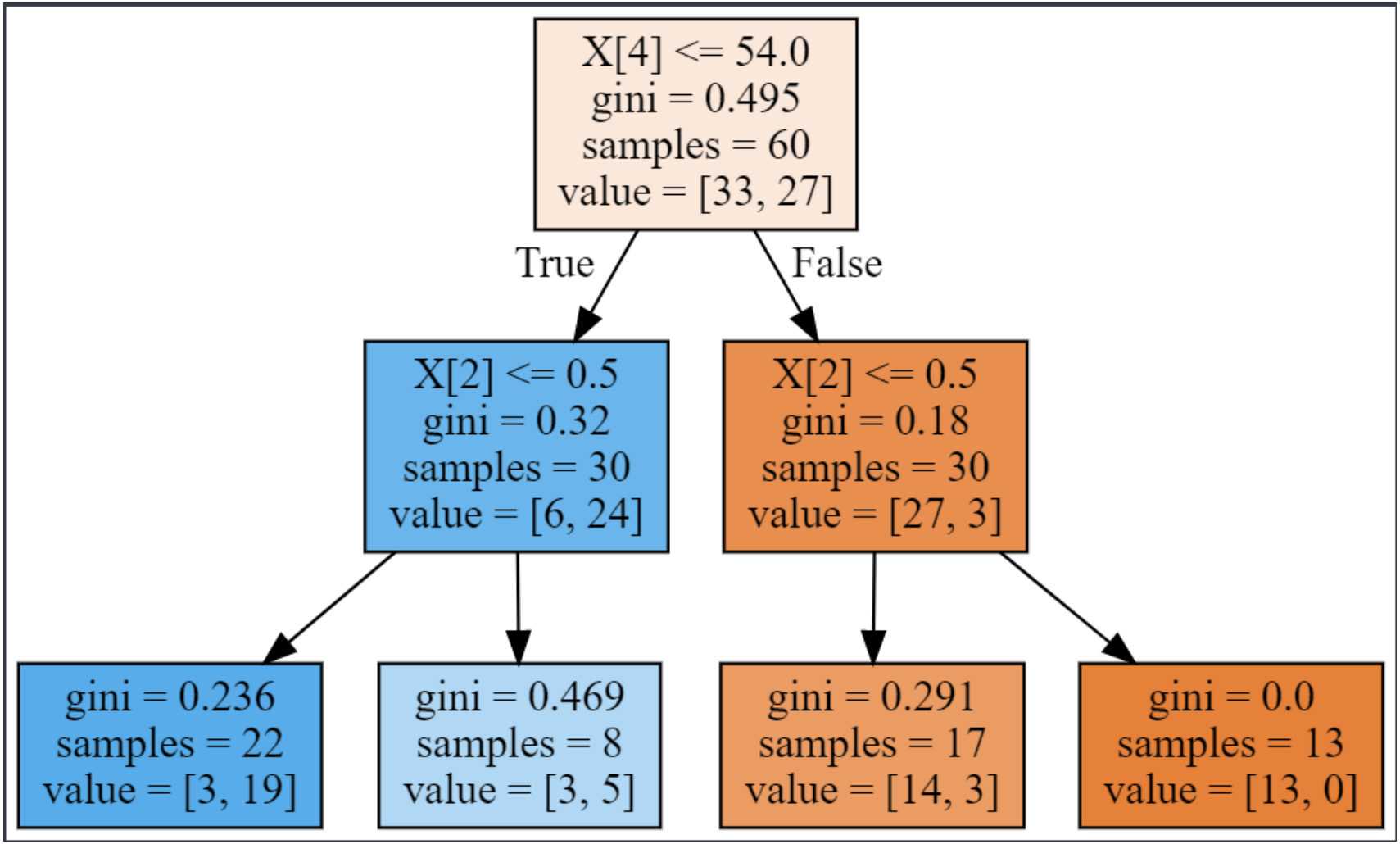
```
best = CV.best_estimator_
best
```

결정나무 분류

```
best = CV.best_estimator_
best
```

```
DecisionTreeClassifier(ccp_alpha=0.0, class_weight=None, criterion='gini',
                       max_depth=2, max_features=2, max_leaf_nodes=None,
                       min_impurity_decrease=0.0, min_impurity_split=None,
                       min_samples_leaf=2, min_samples_split=2,
                       min_weight_fraction_leaf=0.0, presort='deprecated',
                       random_state=23, splitter='best')
```

결정나무 분류


X[4] <= 54.0
gini = 0.495
samples = 60
value = [33, 27]
True
False
X[2] <= 0.5
gini = 0.32
samples = 30
value = [6, 24]
X[2] <= 0.5
gini = 0.18
samples = 30
value = [27, 3]
gini = 0.236
samples = 22
value = [3, 19]
gini = 0.469
samples = 8
value = [3, 5]
gini = 0.291
samples = 17
value = [14, 3]
gini = 0.0
samples = 13
value = [13, 0]

결정나무 분류

```
from sklearn.model_selection import cross_val_score
from sklearn.metrics import accuracy_score

accuracy_score(best.predict(feature), label)
```

```
 0.85
```

```
cross_val_score(best, feature, label, cv=5)
```

```
 array([0.83333333, 0.91666667, 0.91666667, 0.66666667, 0.91666667])
```

소결) 정확도 85%는 꽤 좋은 성적이고 교차 검증 성적도 좋다

분류 문제 다루기 : 주요변수 이해

중요 변수의 이해

> **criterion**: 지니 혹은 엔트로피 알고리즘 선택 'Gini' 혹은 'Entropy'.
>
> **max_depth**: 위의 예제의 경우 4이다. 이 숫자가 크면 오퍼피팅 될 수 있다.
>
> **max_features**: 최적의 분할을 고려할때 초대 피처 개수, 디폰트는 None으로 테이터 셋트를 사용하여 분할 수행
>
> **max_lead_nodes**: 말단 리프 노드의 최대 갯수
>
> **min_samples_leaf**: 말단 노드인 리프노드가 되기 위한 최소한의 샘플 데이터 수
>
> **min_samples_split**: 노드를 분할 하기 위한 최소한의 샘플 데이터의 수. 과적합을 제어에 도움이 됨
>
> - 디폴트는 20이고, 작게 설정할 수록 분할 노드가 많아져서 과적함 증가

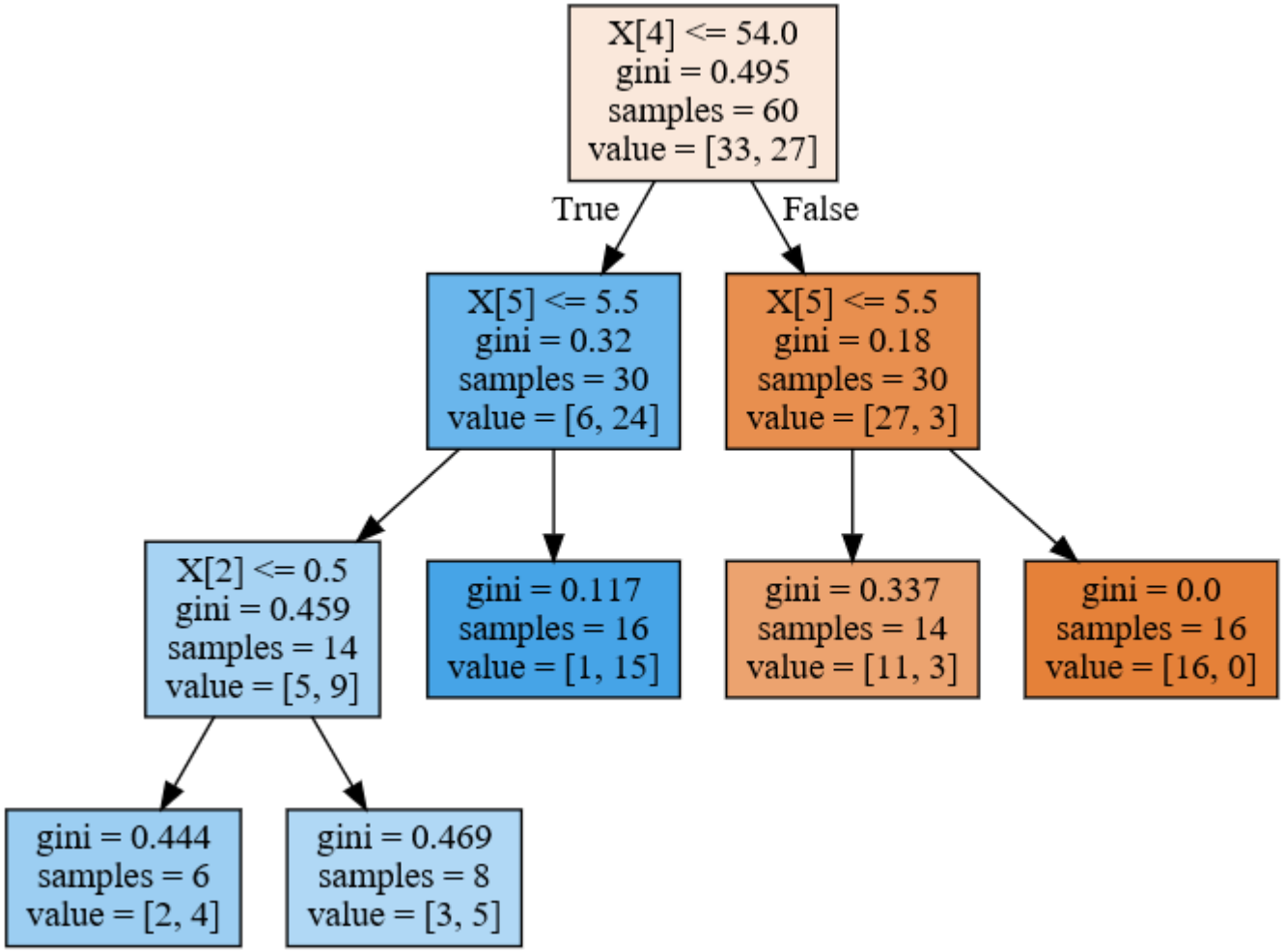
연습문제 : min_samples_leaf 이해

연습문제 min_samples_leaf 개수를 5로 했을때 차이점을 논의하시오

```
from sklearn import tree
model1 = tree.DecisionTreeClassifier(min_samples_leaf = 5, random_state = 23)
model1.fit(feature, label)
```

```
DecisionTreeClassifier(ccp_alpha=0.0, class_weight=None, criterion='gini',
                       max_depth=None, max_features=None, max_leaf_nodes=None,
                       min_impurity_decrease=0.0, min_impurity_split=None,
                       min_samples_leaf=5, min_samples_split=2,
                       min_weight_fraction_leaf=0.0, presort='deprecated',
                       random_state=23, splitter='best')
```

연습문제 : min_samples_leaf 이해



연습문제 : min_samples_split 이해

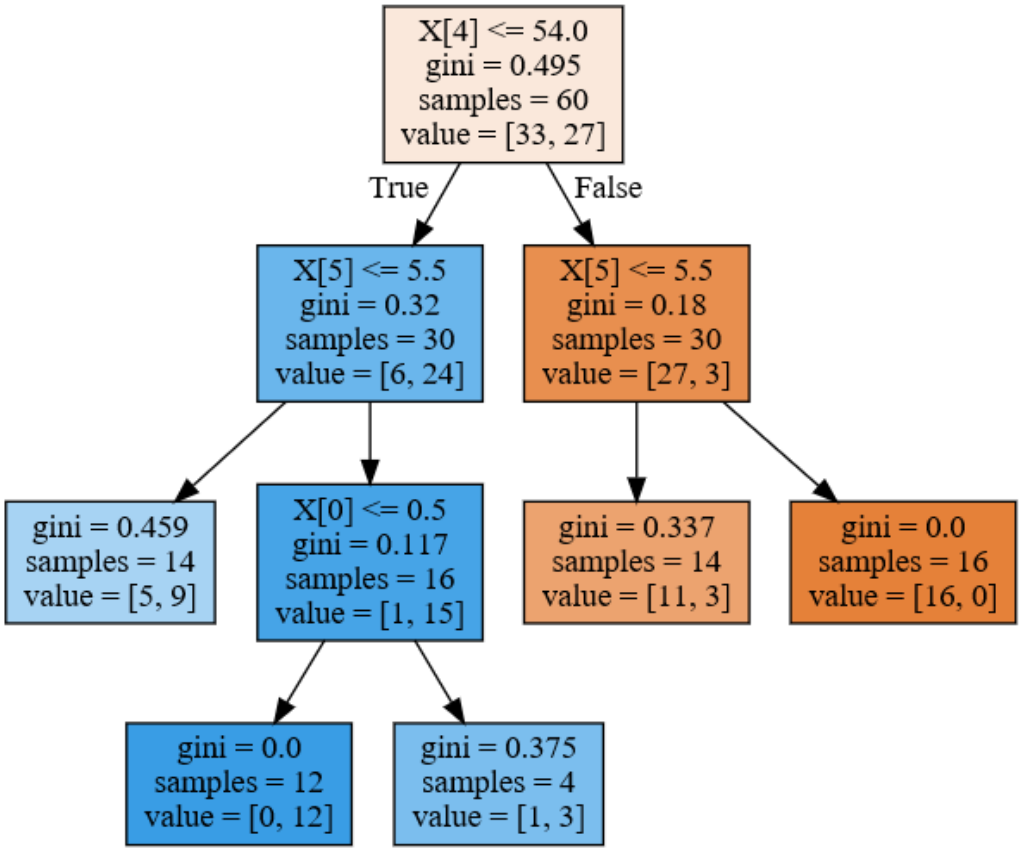
min_samples_split

```
from sklearn import tree
model2 = tree.DecisionTreeClassifier(min_samples_split =16, random_state = 23)
model2.fit(feature, label)
```

```
DecisionTreeClassifier(ccp_alpha=0.0, class_weight=None, criterion='gini',
                       max_depth=None, max_features=None, max_leaf_nodes=None,
                       min_impurity_decrease=0.0, min_impurity_split=None,
                       min_samples_leaf=1, min_samples_split=16,
                       min_weight_fraction_leaf=0.0, presort='deprecated',
                       random_state=23, splitter='best')
```

(실습 해보기)
min_samples_split를 4~20 사이를 변화시키면서,
그래프의 변화를 이해하라

연습문제 : min_samples_split 이해



트리의 변화를 확인하여라.

최적의 결정트리 파라미터 찾기(1)

Grid Search CV

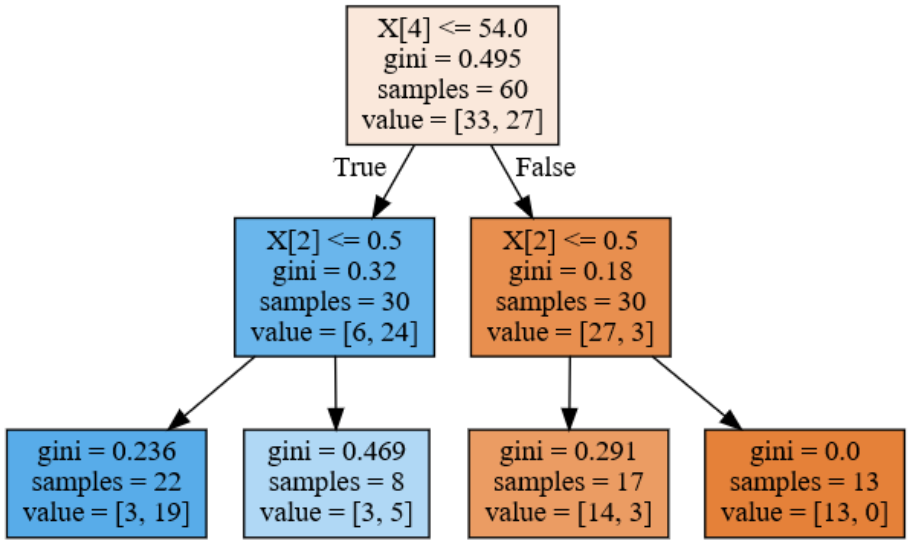
```
from sklearn.model_selection import GridSearchCV

param1 = {'min_samples_leaf': [2,3,4,5],
          'min_samples_split': [2,3,5,10,12,14],
          'max_depth': [2,3,4,5,6],
          'criterion': ['gini', 'entropy'],
          'max_features':[2,3,4]}

CV = GridSearchCV(model, param1)

CV.fit(feature, label)
```

```
GridSearchCV(cv=None, error_score=nan,
             estimator=DecisionTreeClassifier(ccp_alpha=0.0, class_weight=None,
                                              criterion='gini', max_depth=None,
                                              max_features=None,
                                              max_leaf_nodes=None,
                                              min_impurity_decrease=0.0,
                                              min_impurity_split=None,
                                              min_samples_leaf=1,
                                              min_samples_split=2,
                                              min_weight_fraction_leaf=0.0,
                                              presort='deprecated',
                                              random_state=23,
                                              splitter='best'),
             iid='deprecated', n_jobs=None,
             param_grid={'criterion': ['gini', 'entropy'],
                         'max_depth': [2, 3, 4, 5, 6],
                         'max_features': [2, 3, 4],
                         'min_samples_leaf': [2, 3, 4, 5],
                         'min_samples_split': [2, 3, 5, 10, 12, 14]},
             pre_dispatch='2*n_jobs', refit=True, return_train_score=False,
             scoring=None, verbose=0)
```

중요 :

1) Grid Search는 무엇인가?
2) 파라미터 공간을 조정해보자
3) 관심이 더 있으면 Random Seach
4) Bayesian Optimization Search
5) NAS (Neural architecture search)
6) Hyperparamenter search (HPO) 쪽으로 주제는 현제 AI에서 핫 이슈임.

최적의 결정트리 파라미터 찾기(2)




```
from sklearn.model_selection import cross_val_score
from sklearn.metrics import accuracy_score

accuracy_score(best.predict(feature), label)
```

```
0.85
```

```
cross_val_score(best, feature, label, cv=5)
```

```
array([0.83333333, 0.91666667, 0.91666667, 0.66666667, 0.91666667])
```

소결) 정확도 85%는 꽤 좋은 성적이고 교차 검증 성적도 좋다

중요:

1) Grid Search에서 얻은 각각 파라미터 최적의 값은 무엇인가?

2) 결정 트리 정확도 85%를 얻었다.

3) 정확도 85%의 의미와 데이터 양과의 관계를 생각해보아라

4) 교차검증은 83.3%을 얻었다. 의미는?

05.
1차 선형 곡선을 결정트리로 회귀하기

이홍석 (hsyi@kisti.re.kr)
www.ust.ac.kr

결정트리 회귀 : 1차 함수 예제

2) 회귀(Regression) 의사결정 회귀 문제
결정트리는 회귀 문제에도 사용할 수 있으며, 사이킷런의 **DecisionTreeRegressor**를 사용

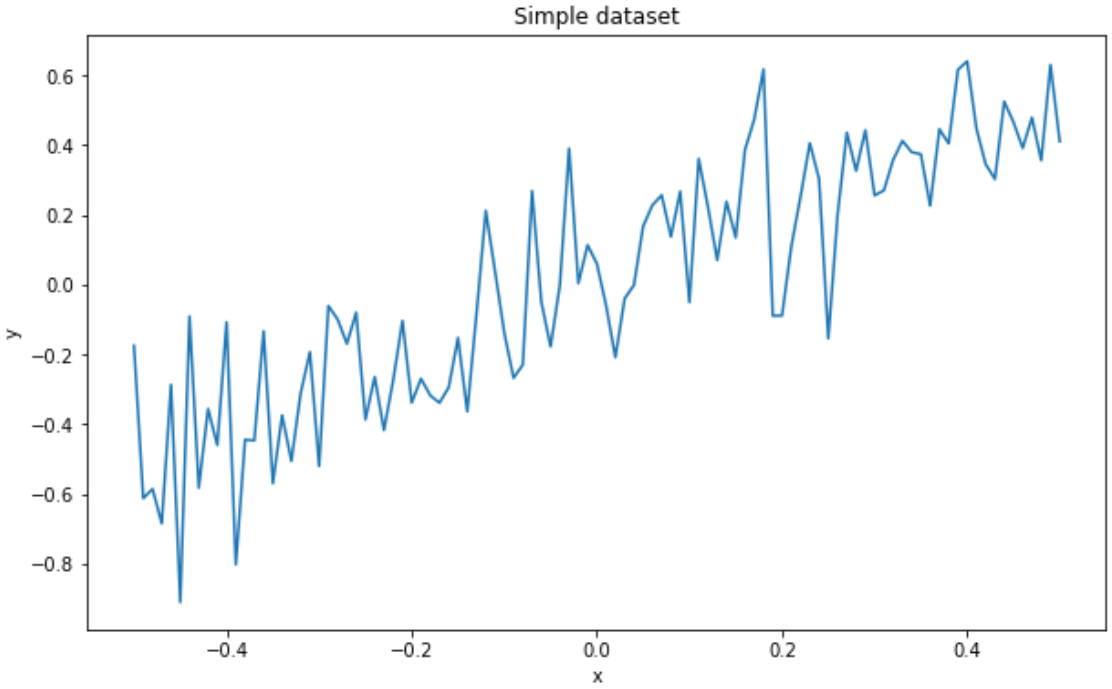
```
import numpy as np
import pandas as pd
```

```
from sklearn import tree
import graphviz

import matplotlib.pyplot as plt
import seaborn as sns
```

예제로 1차 함수 형태의 데이터셋에서 회귀 트리 만들어 보기

결정트리 회귀 : 1차 함수 예제

```
nPoints = 100
xPlot = [(float(i)/float(nPoints) - 0.5) for i in range(nPoints + 1)]
x = [[s] for s in xPlot]
np.random.seed(1)
y = [s + np.random.normal(scale=0.2) for s in xPlot]
```



결정트리 회귀 : 1차 함수 예제

max_depth=1 설정으로 간단한 트리 구조

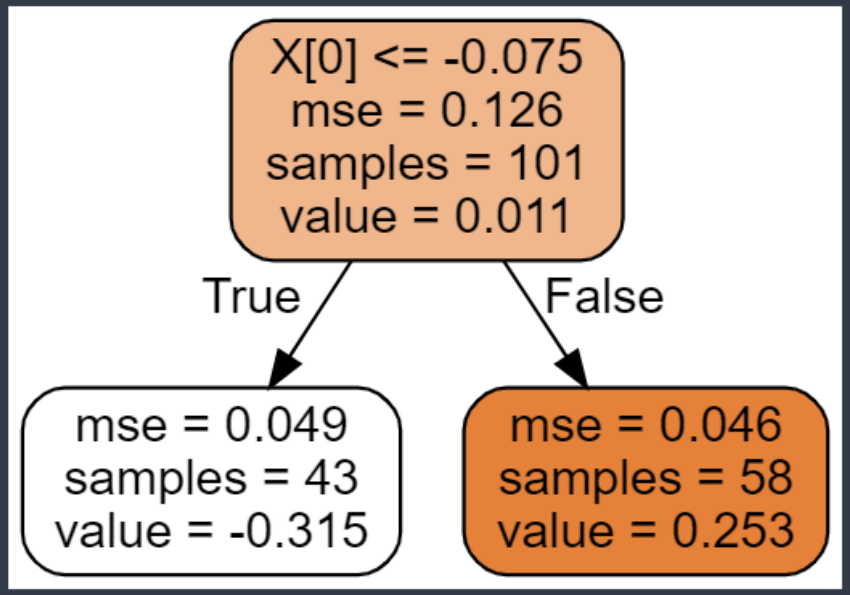
```
model = tree.DecisionTreeRegressor(max_depth=1)
model.fit(x, y)
```

```
DecisionTreeRegressor(ccp_alpha=0.0, criterion='mse', max_depth=1,
                      max_features=None, max_leaf_nodes=None,
                      min_impurity_decrease=0.0, min_impurity_split=None,
                      min_samples_leaf=1, min_samples_split=2,
                      min_weight_fraction_leaf=0.0, presort='deprecated',
                      random_state=None, splitter='best')
```

- 아래 블럭 다이아그램에서 루트 노드는 -0.075을로 기준 (split 점)
- 이 split 값을 기준으로 2개의 그룹으로 분류
- 아래 2개의 박스 중에서 왼쪽으로는 43개의 샘플, 오른쪽에는 58개의 샘풀로 분류
- 만일 테스트 값이 x=0.2이라면, 예측 값은 y=0.253 임

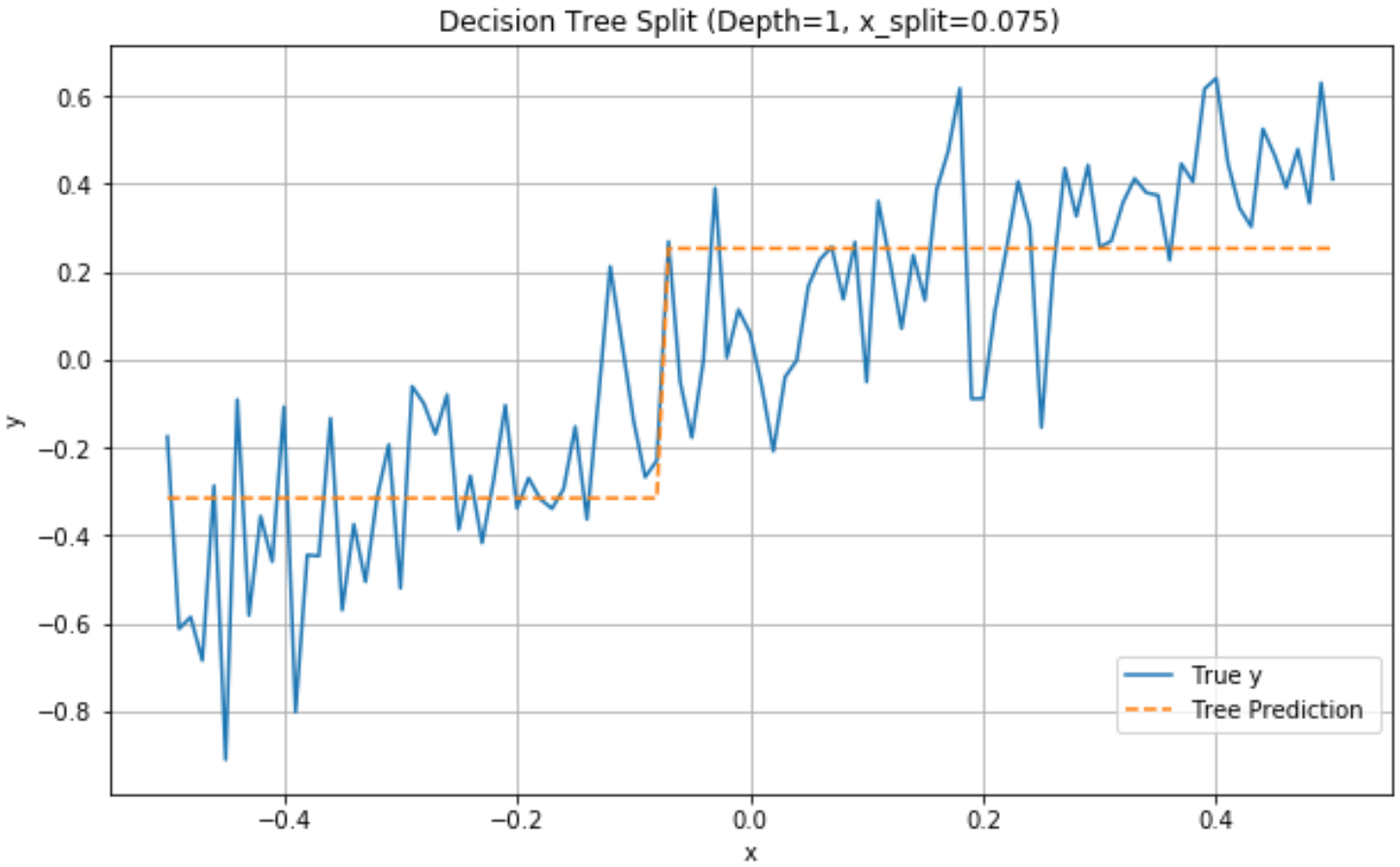
결정트리 회귀 : 1차 함수 예제

```
#draw the decision tree result with graphviz
graph = tree.export_graphviz(model, out_file = None,
                             rounded = True, filled = True)
graphviz.Source(graph)
```



결정트리 회귀 : 1차 함수 예제

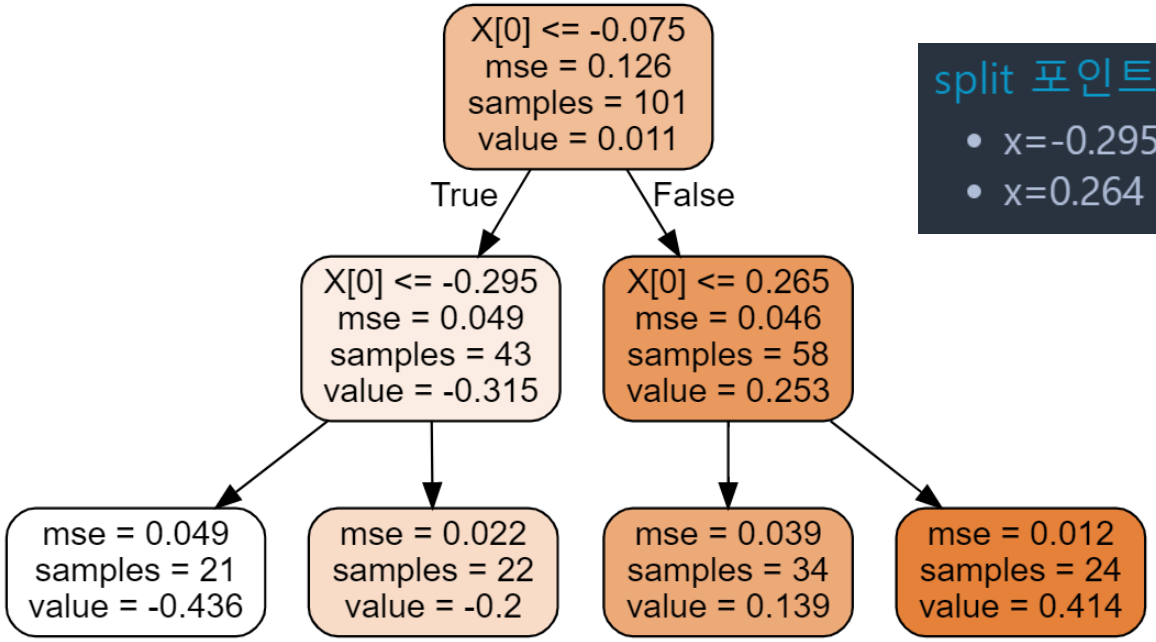
예측을 해보자



결정트리 회귀 : 1차 함수 예제

연습문제 : 의사결정 나무 depth=2로 증가 시켜라

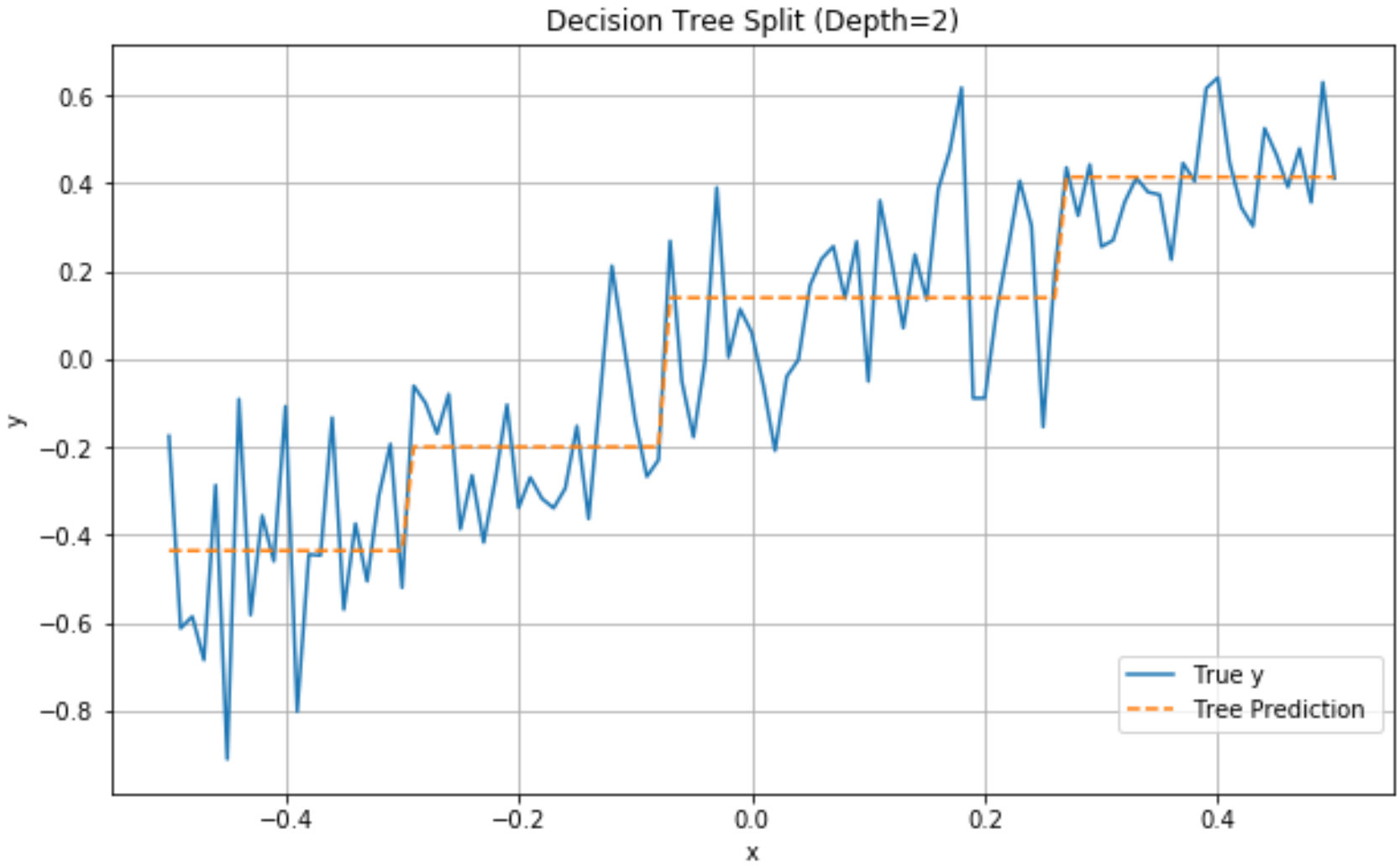
```
# 결정나무 다시 설정 max_depth=2
model2 = tree.DecisionTreeRegressor(max_depth=2)
model2.fit(x, y)
```



split 포인트는 2개가 더 생김

- x=-0.295
- x=0.264

결정트리 회귀 : 1차 함수 예제



결정트리 회귀 : 1차 함수 예제

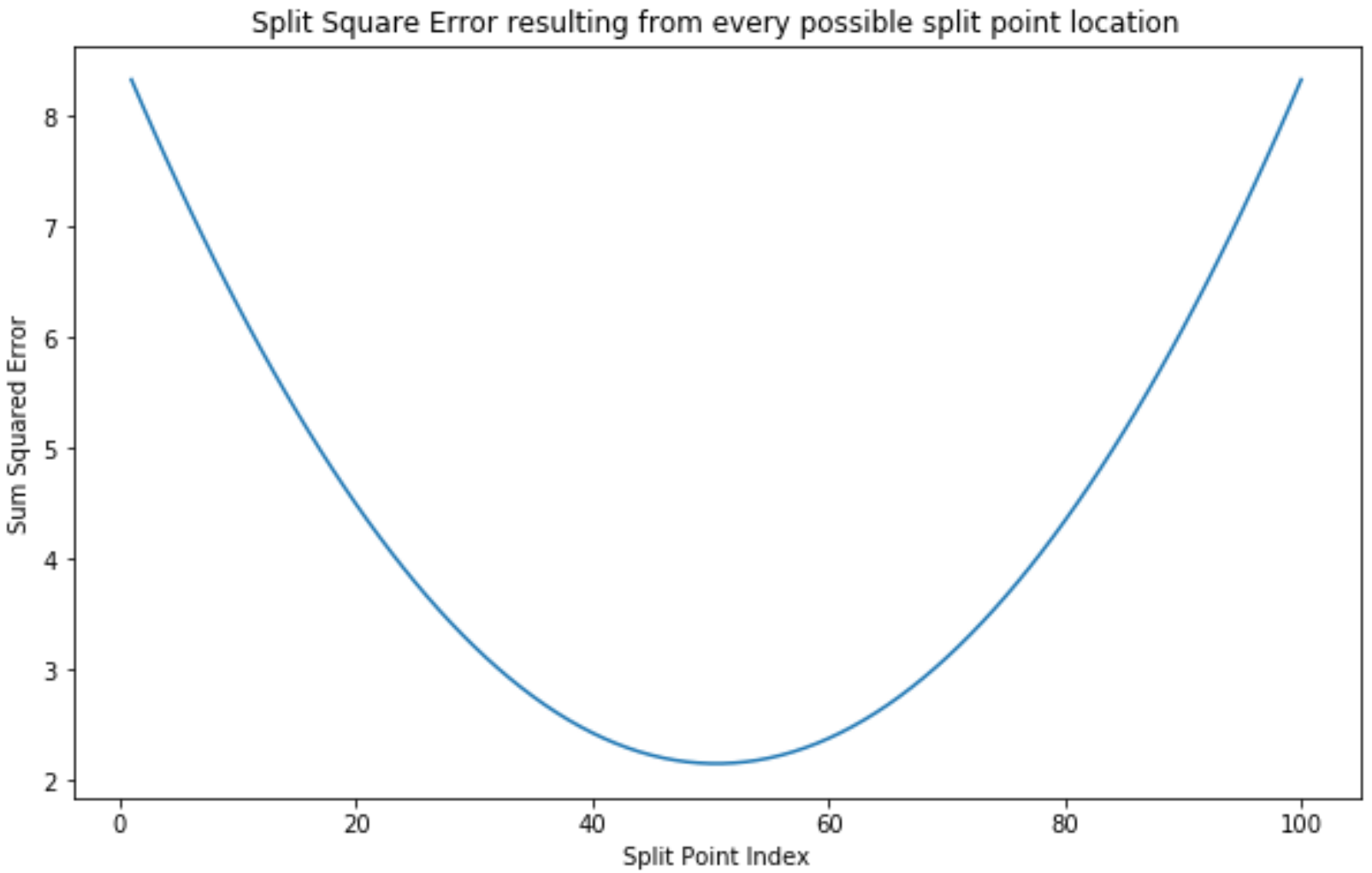
Split Points 찾기는 어떻게하나? ¶

- 트리는 예측값의 제곱 오차를 최소화(MSE)한다.
- 생각해보면, 임의의 split 값이 주어지면 2개의 그룹 중에 1개로 선택된다.
- 각각의 그룹의 평균은 MSE를 최소화하는 값이 된다.
- 아래 예제를 보자

결정트리 회귀 : 1차 함수 예제

```
sse = []
xMin = []
for i in range(1, len(xPlot)):
    lhList = list(xPlot[0:i])
    rhList = list(xPlot[i:len(xPlot)])
    lhAvg = sum(lhList) / len(lhList)
    rhAvg = sum(rhList) / len(rhList)
    lhSse = sum([(s - lhAvg) * (s - lhAvg) for s in lhList])
    rhSse = sum([(s - rhAvg) * (s - rhAvg) for s in rhList])
    sse.append(lhSse + rhSse)
    xMin.append(max(lhList))
```

결정트리 회귀 : 1차 함수 예제



결정트리 회귀 : 1차 함수 예제

SSE 함수에서 최소값을 찾고, 최소값의 위치를 찾은 다음, 그 값에 해당하는 최소값

```
minSse = min(sse)
idxMin = sse.index(minSse)
print(xMin[idxMin])
```

```
-0.010000000000000009
```

결정트리 회귀 : 1차 함수 예제

멀티 변수 트리 학습은 어떻게?

- 알고리듬은 MSE의 최소값에 기여하는 모든 가능한 split point을 찾는다.

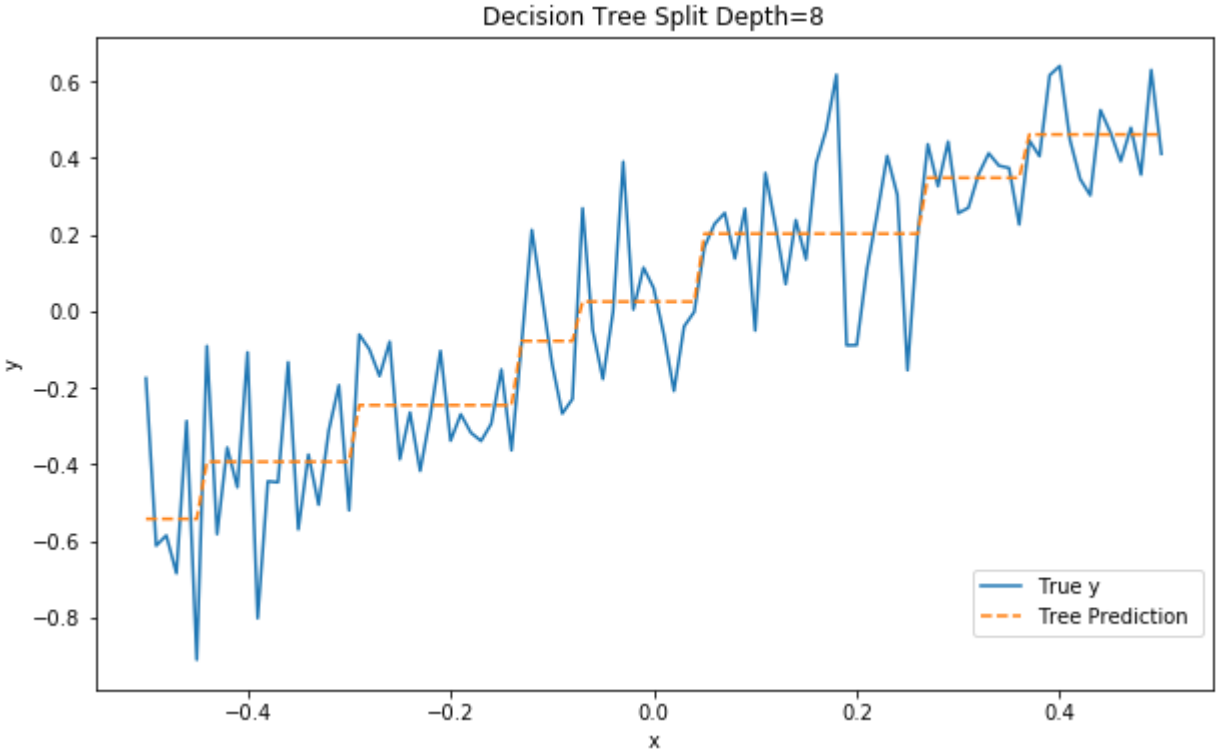
결정트리의 과적합 (Overfitting) 문제

- 데이터는 적은데 너무 많은 split point를 고려해보면
- 학습은 잘 되는데, 예측할때 틀릴 수가 많다.
- depth를 높여서 실습해보자

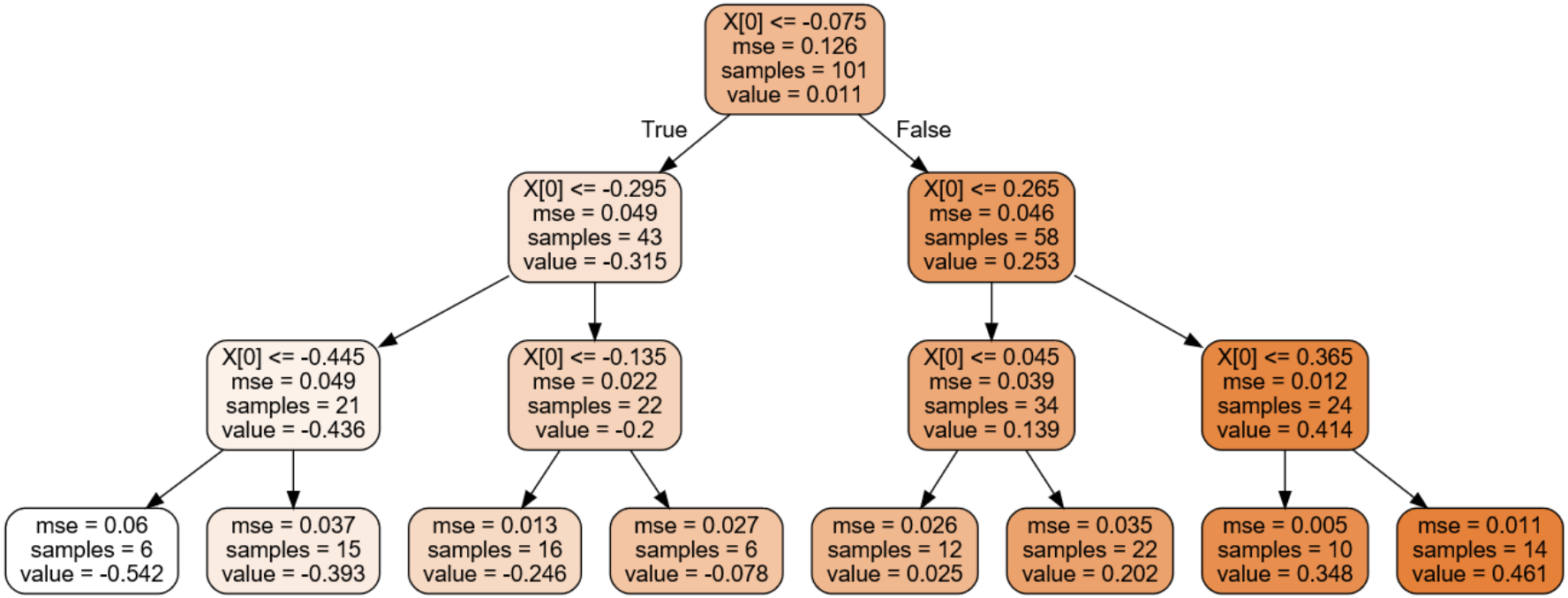
결정트리 회귀 : 1차 함수 예제

연습문제2: max_depth를 3로 높여서 예측해보기

```
model3 = tree.DecisionTreeRegressor(max_depth=3)
model3.fit(x, y);
```



결정트리 회귀 : 1차 함수 예제



결정트리 회귀 : 1차 함수 예제

숙제: 데이터 숫자를 늘려서 확인해보기

- 데이터 개수를 nPoints = 200로 2개 증가 시킴
- max_depth=1
- max_depth=2
- max_depth=4 일때 결정트리의 예측정확도를 구해라?

UST 과학기술연합대학원대학교
UNIVERSITY OF SCIENCE & TECHNOLOGY

Thank You!

www.ust.ac.kr