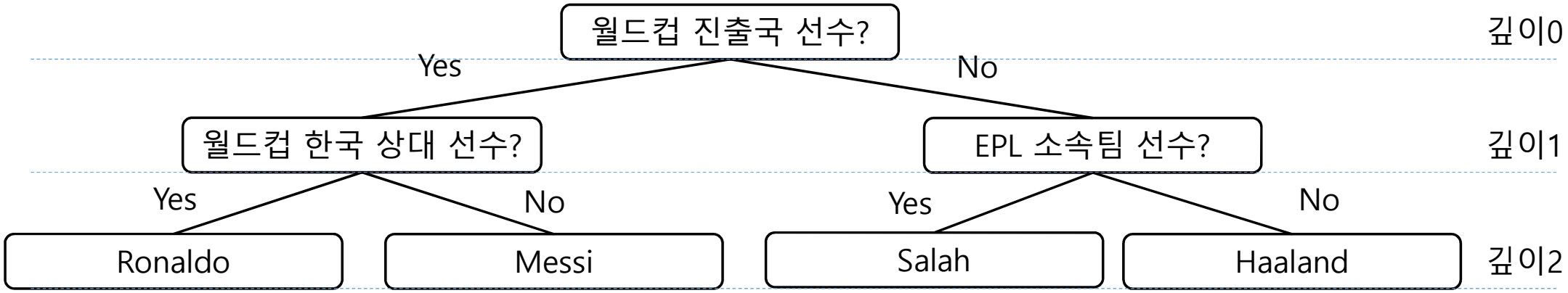
의사결정나무

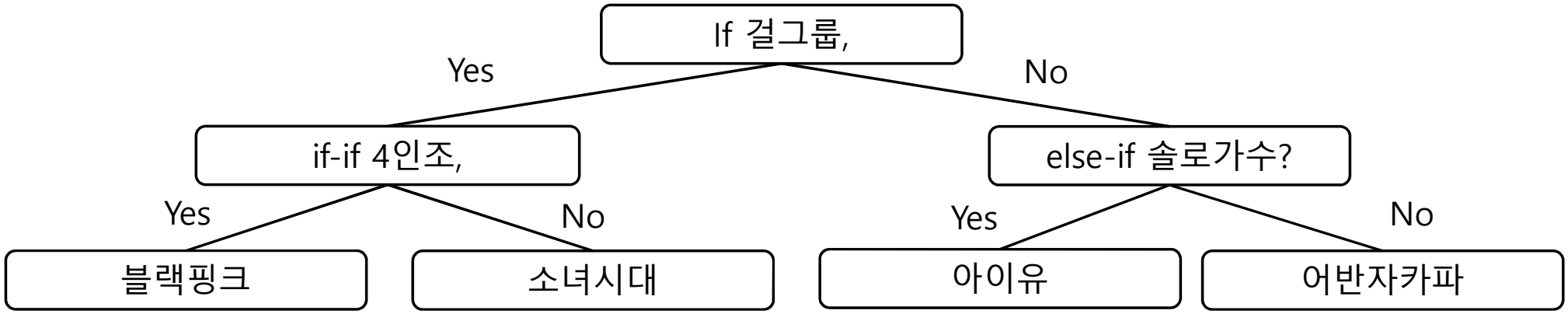
의사결정나무(Decision Tree)

- 학습 데이터 분석을 통해 데이터 속성으로부터 패턴을 찾아내서 분류 문제를 해결하는 모델
- 질문을 계속 던져 답을 결정하게 하는 방식
- 좋은 질문일수록 분류가 쉬워지고, 나쁜 질문일수록 정확한 분류가 어려워짐
- 의사결정방식 과정의 표현법이 나무와 비슷하여 의사결정나무라고 불림



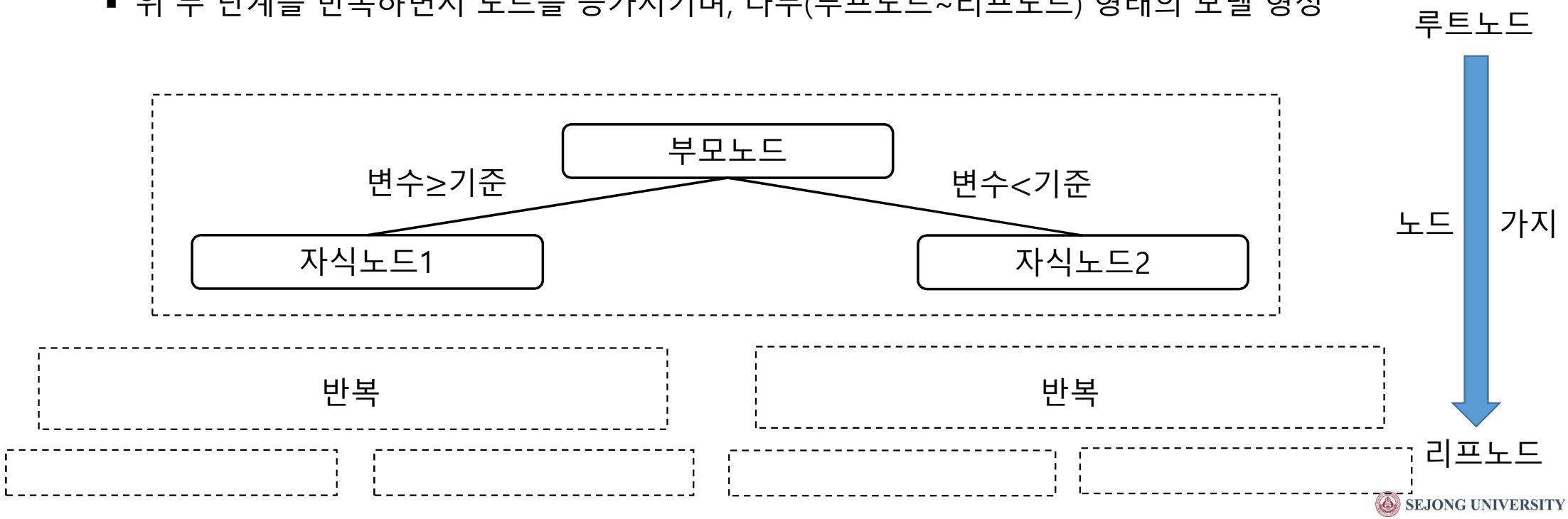
의사결정나무 장단점

- 조건문 형식으로 표현되기 때문에 모델을 쉽게 이해할 수 있고, 중요한 변수(특징) 선택에 유용
- CART(Classification and Regression Tree): 분류와 회귀 문제를 모두 해결
- 정확한 모델을 만들기 위하여 비교적 많은 데이터와 시간이 필요
- 데이터의 변화에 민감하기 때문에, 학습 데이터와 테스트 데이터의 도메인이 유사해야함



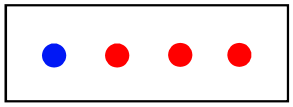
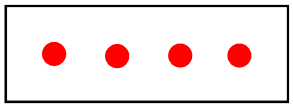
의사결정나무 과정

- 여러 독립변수 중 한 개의 변수 선택 후 분류를 위한 기준(규칙) 결정
- 기준에 따라서 복수 개의 자식 노드 구성
- 위 두 단계를 반복하면서 노드를 증가시키며, 나무(루프노드~리프노드) 형태의 모델 형성



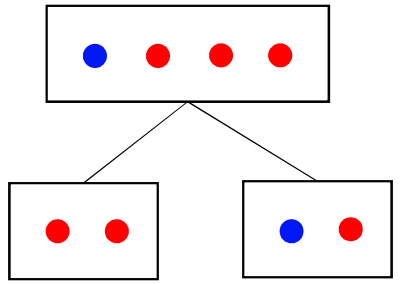
불순도(Impurity)

- 엔트로피(Entropy) 상태
- 순도가 높다는 것은 엔트로피가 낮은 상태
- 불순도가 높다는 것은 불확실성이 높다는 것이며, 이는 엔트로피가 높은 상태임을 의미
- 불순도가 낮아지는 방향으로, 엔트로피가 낮아지는 상태로 의사결정나무 모델 생성

	지니(Gini) 불순도	엔트로피 불순도
	$1-\left(\frac{1}{4}\right)^2-\left(\frac{3}{4}\right)^2=0.375$	$-\frac{1}{4}\log_2(\frac{1}{4})-\frac{3}{4}\log_2(\frac{3}{4})=0.811$
	$1-\left(\frac{0}{4}\right)^2-\left(\frac{4}{4}\right)^2=0$	$-\frac{0}{4}\log_2(\frac{0}{4})-\frac{4}{4}\log_2(\frac{4}{4})=0$

정보이득(Information Gain)

- 부모노드(이전단계)와 자식노드(다음단계) 간의 불순도 차이
- 정보이득이 최대가 되도록 데이터 분류 기준 결정
- 분할된 데이터들의 불순도가 작을수록 정복이득 증가



지니(Gini) 불순도

부모노드 $1-\left(\frac{1}{4}\right)^2-\left(\frac{3}{4}\right)^2=0.375$

자식노드 $\left(1-\left(\frac{0}{2}\right)^2-\left(\frac{2}{2}\right)^2\right)\times 0.5+\left(1-\left(\frac{1}{2}\right)^2-\left(\frac{1}{2}\right)^2\right)\times 0.5=0.25$

정보이득

$0.375-0.25=0.125$

sklearn.tree.DecisionTreeClassifier

class sklearn.tree.DecisionTreeClassifier(*, *criterion='gini'*, *splitter='best'*, *max_depth=None*, *min_samples_split=2*, *min_samples_leaf=1*, *min_weight_fraction_leaf=0.0*, *max_features=None*, *random_state=None*, *max_leaf_nodes=None*, *min_impurity_decrease=0.0*, *class_weight=None*, *ccp_alpha=0.0*) [source]

A decision tree classifier.

criterion : *{"gini", "entropy"}, default="gini"*

The function to measure the quality of a split. Supported criteria are "gini" for the Gini impurity and "entropy" for the information gain.

max_depth : *int, default=None*

The maximum depth of the tree. If None, then nodes are expanded until all leaves are pure or until all leaves contain less than min_samples_split samples.

https://scikit-learn.org/stable/modules/generated/sklearn.tree.DecisionTreeClassifier.html#sklearn.tree.DecisionTreeClassifier

의사결정나무 분류

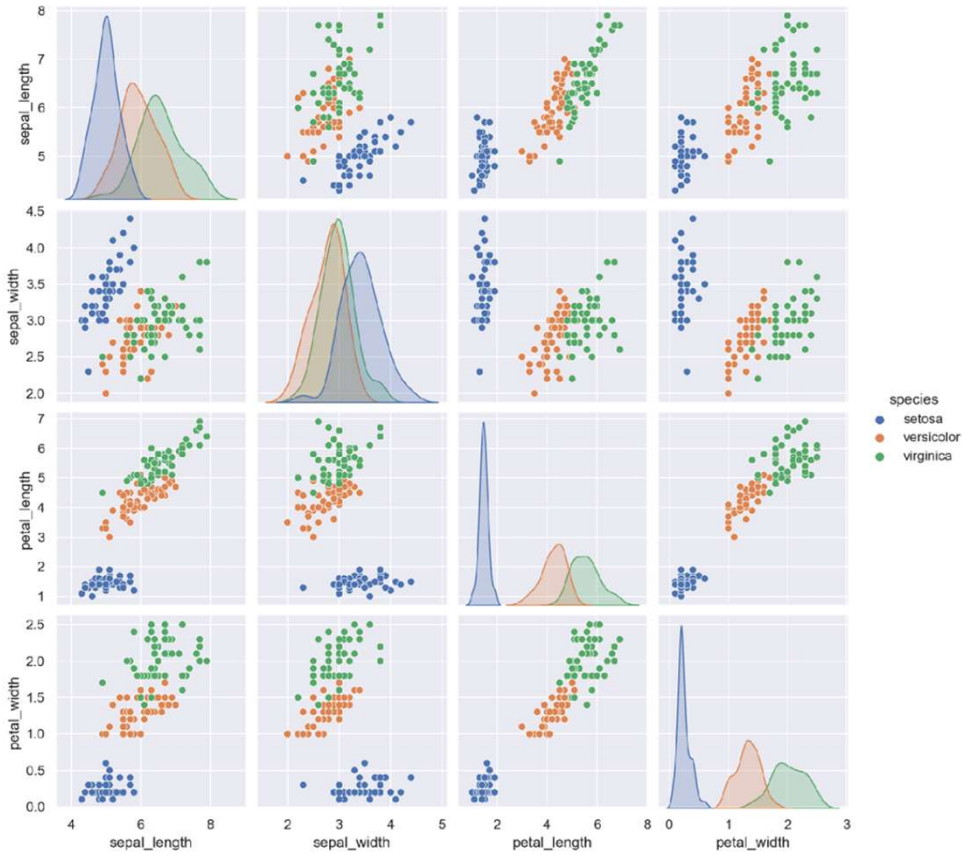
```
import seaborn as sns
iris = sns.load_dataset("iris")

data = iris.drop("species", axis = 1)
t = iris[["species"]].copy()
t[t["species"] == "setosa"] = 0
t[t["species"] == "versicolor"] = 1
t[t["species"] == "virginica"] = 2
t = t["species"].astype("int")
```

```
from sklearn.model_selection import train_test_split
train_data, test_data, train_target, test_target = train_test_split(
    data, t, test_size = 0.3, random_state = 42, stratify = t)

from sklearn.tree import DecisionTreeClassifier
dt = DecisionTreeClassifier(random_state = 42)
dt.fit(train_data, train_target)
print("Train-Eval:", dt.score(train_data, train_target))
print("Test-Eval :", dt.score(test_data, test_target))
```

```
from sklearn.metrics import confusion_matrix
conf = confusion_matrix(test_target, dt.predict(test_data))
print(conf)
```




```
Train-Eval: 1.0
Test-Eval : 0.9333333333333333
[[15  0  0]
 [ 0 12  3]
 [ 0  0 15]]
```

sklearn.tree.plot_tree

sklearn.tree.plot_tree(*decision_tree*, *, *max_depth=None*, *feature_names=None*, *class_names=None*, *label='all'*, *filled=False*, *impurity=True*, *node_ids=False*, *proportion=False*, *rounded=False*, *precision=3*, *ax=None*, *fontsize=None*) [source]

Plot a decision tree.

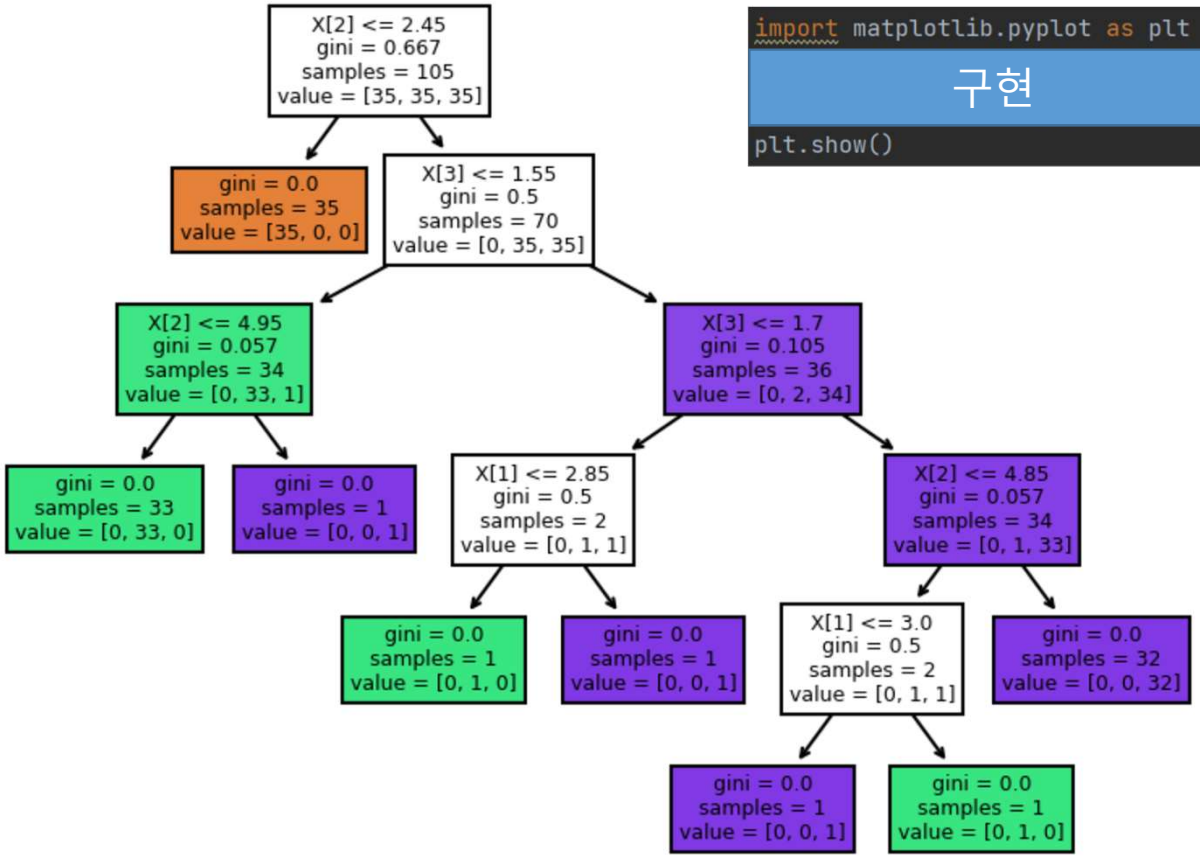
max_depth : *int, default=None*
The maximum depth of the representation. If None, the tree is fully generated.

feature_names : *list of strings, default=None*
Names of each of the features. If None, generic names will be used ("X[0]", "X[1]", ...).

filled : *bool, default=False*
When set to `True`, paint nodes to indicate majority class for classification, extremity of values for regression, or purity of node for multi-output.

https://scikit-learn.org/stable/modules/generated/sklearn.tree.DecisionTreeClassifier.html#sklearn.tree.DecisionTreeClassifier

의사결정나무 시각화




```
import matplotlib.pyplot as plt
```

구현

```
plt.show()
```

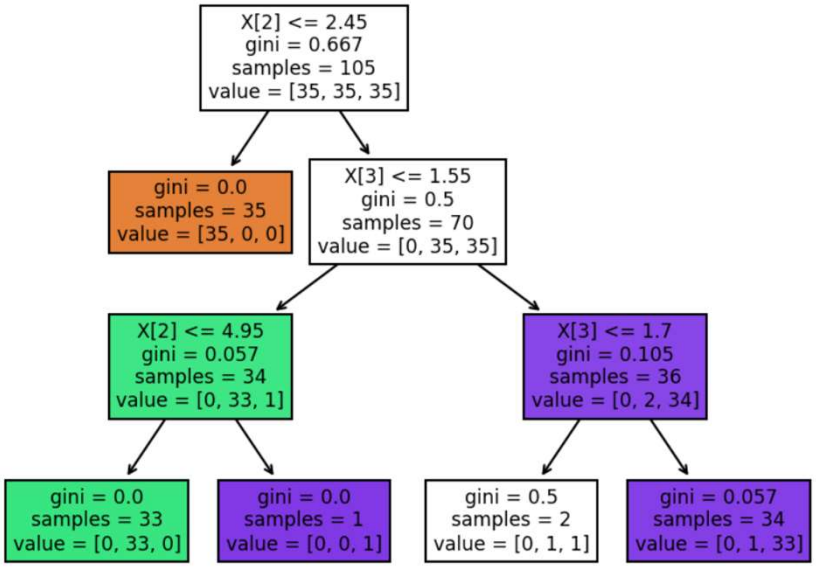
```
print(dt.feature_importances_)
```

```
[0.         0.02857143 0.54117647 0.4302521 ]
```

특징 중요도

가지치기(Pruning)

- 의사결정나무 모델의 깊이가 깊을수록 학습 데이터에 과대적합될 가능성이 높음
- 가장 간단한 방법은 나무의 최대 깊이 지정




```
Train-Eval: 1.0
Test-Eval : 0.9333333333333333
[[15  0  0]
 [ 0 12  3]
 [ 0  0 15]]
```

```
Train-Eval: 0.9809523809523809
Test-Eval : 0.9777777777777777
[[15  0  0]
 [ 0 14  1]
 [ 0  0 15]]
```

참고자료

- 지능기전공학부 최유경 교수님 자료, https://github.com/sejongresearch/2021.MachineLearning
- 코랩(Colab), https://colab.research.google.com/
- 파이썬(Python), https://www.python.org/doc/
- 사이킷런(sckit-learn), https://scikit-learn.org/stable/index.html
- 판다스(pandas), https://pandas.pydata.org/
- 맷플롯립(matplotlib), https://matplotlib.org/
- 씨본(seaborn), https://seaborn.pydata.org/
- 캐글(Kaggle), https://www.kaggle.com/
- 넘파이(numpy), https://numpy.org/doc/stable/
- 스택오퍼플러우(stackoverflow), https://stackoverflow.com/