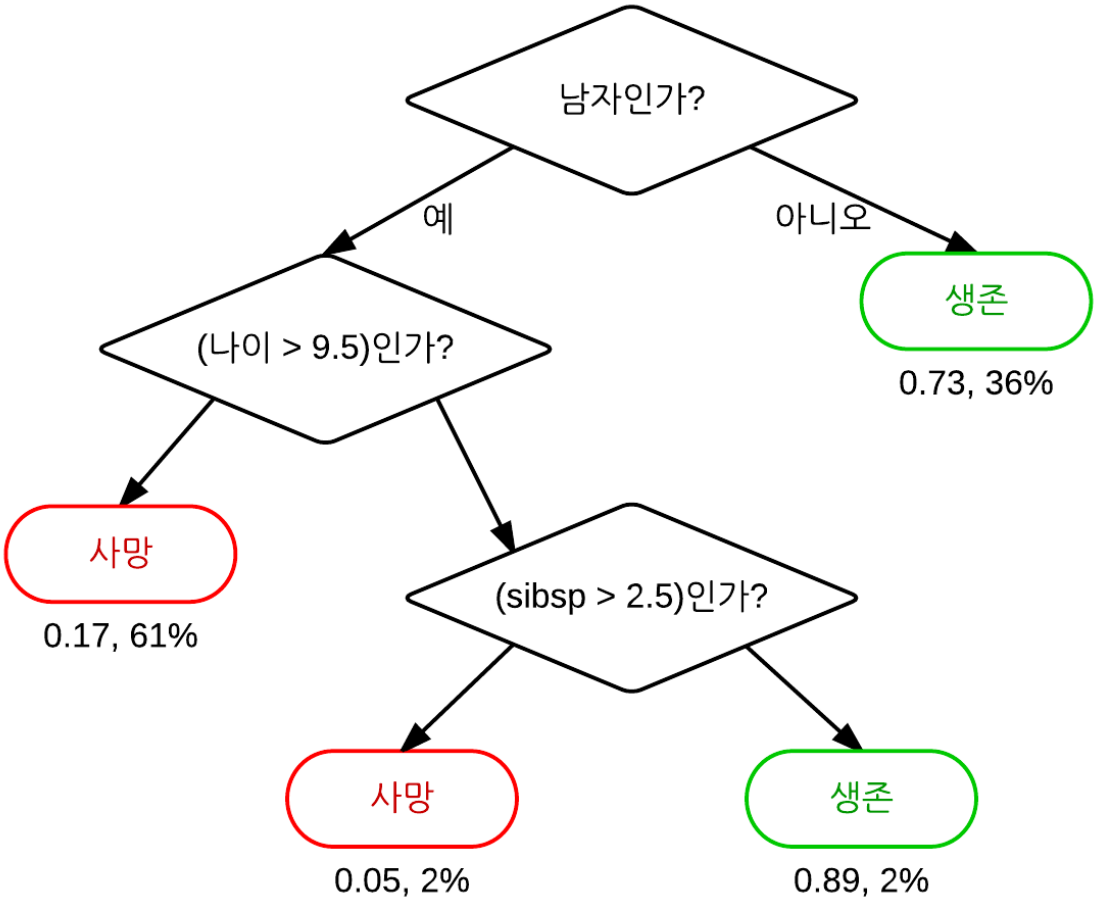
결정 트리(Decision Tree)

결정 트리란?

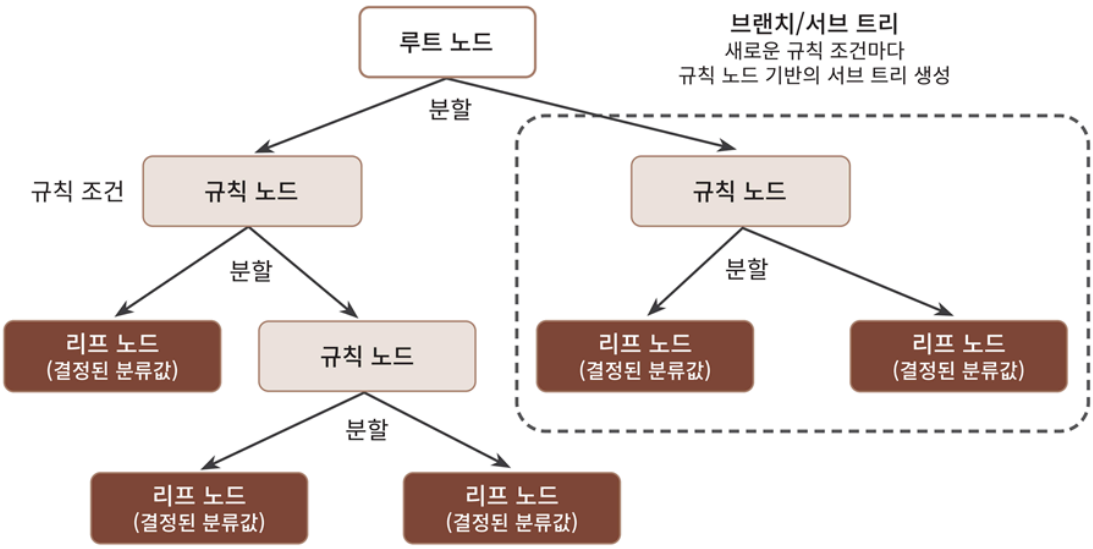
- ML 알고리즘 중 직관적으로 이해하기 쉬운 알고리즘으로 분류와 회귀 문제에서 가장 널리 사용
- 데이터 스케일링이나 정규화, 이상치 등의 전처리 영향이 적음
- 특정기준(질문)에 따라서 데이터를 구분
- 가장 쉬운 규칙 표현 방법 : if/else 기반 (스무고개 게임과 유사)
- tree : 전체적인 모양이 나무를 뒤집어 놓은 것과 닮았다고 해서 붙여진 이름

- 예. 타이타닉 데이터에 대한 결정트리



출처:
https://ko.wikipedia.org/wiki/%EA%B2%B0%EC%A0%95_%ED%8A%B8%EB%A6%AC_%ED%95%99%E

결정 트리 구조



파이썬 머신러닝 완벽 가이드, 권철민, 위키북스 , P185

관련 용어

- node : 결정트리의 질문
- root node : 모든 자료를 포함하고 있는 결정트리의 출발점(첫 질문)
- leaf node : 최종 결과를 나타내는 노드로 더 이상 자료를 나누지 않음(결정된 클래스 값)
- parent/child node : 상위 노드/하위 노드
- splitting : 노드를 주어진 기준에 따라 하위 노드로 나누는 것
- sub tree/branch : tree를 나누어 생성된 하위 tree (새로운 규칙 조건마다 Sub Tree 생성)
- decision node(규칙 노드) : 규칙 조건
- depth : root node 부터 leaf node까지의 중간 node들의 수

결정 트리 모델

- 분류 트리
  - 범주형 목표 변수를 기준으로 마디를 나눔
  - 끝마디에 포함된 자료의 범주가 분류 결과 값이 됨
- 회귀 트리
  - 연속형 목표 변수를 기준으로 마디를 나눔
  - 끝마디에 포함된 자료의 평균값이 각 끝마디의 회귀 값이 됨

결정트리에서 중요한 이슈

- 데이터 세트의 피처를 결합해 규칙 조건을 만들 때마다 규칙노드가 만들어짐
- `규칙이 많아지면` 결정 방식이 복잡해지고 `과대적합(overfitting)` 발생
  - 즉, `depth가 길어질수록` 결정트리의 예측 성능이 저하될 가능성이 높음
- 가능한 적은 결정노드로 높은 예측 정확도를 가지려면
  - 데이터를 분류할 때 최대한 많은 데이터 세트가 해당 분류에 속할 수 있도록 결정 노드의 규칙이 정해져야 함
- `어떻게 트리를 분할할 것인가`**가 중요**

- `어떤 기준으로` 규칙을 만들어야 `가장 효율적인 분류`가 될 것인가가 `알고리즘의 성능`을 크게 좌우
- 최대한 **균일한 데이터 세트**를 구성할 수 있도록 분할하는 것이 필요

정지규칙(Stopping rule)

- 결정트리를 결정하는 하이퍼파라미터를 조절
- max depth 제한
- leaf의 sample 수 제한
- split을 위한 sample 수 제한

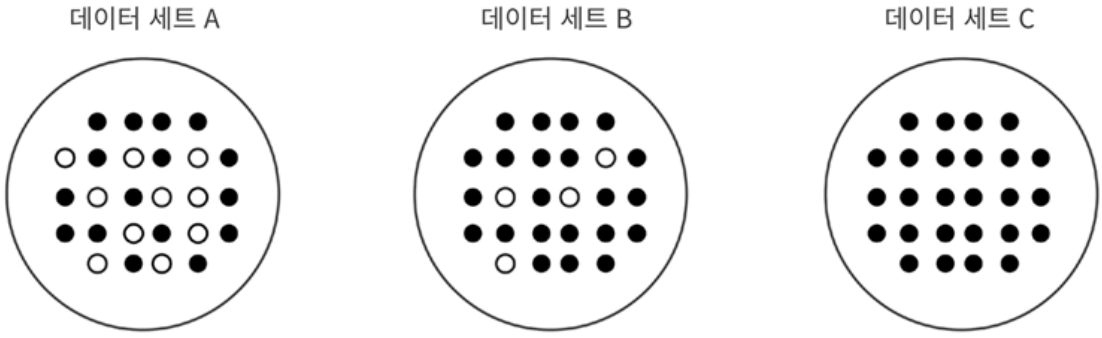
가지치기 (pruning)

- 특정 노드 밑의 하부 트리를 제거하여 일반화 성능을 높이는 방법
- 깊이가 줄어들고 결과의 개수가 줄어듦
- 과대적합(overfitting)을 막기 위한 방법

균일도(불순도: impurity)

- 규칙 조건 생성에서 중요한 것
- 정보 균일도가 높은 데이터 세트를 먼저 선택할 수 있도록 규칙 조건을 만드는 것이 중요

- 데이터 세트의 균일도는 데이터를 구분하는 데 필요한 정보의 양에 영향을 미침



파이썬 머신러닝 완벽 가이드, 권철민, 위키북스 , P186

균일도(불순도) 측정 지표

범주형 목표변수 (각 범주에 속하는 빈도에 기초하여 분리, Classification)

- Entropy Reduction
- Gini Reduction
- Chi-square test

연속형 목표변수 (목표 변수의 평균에 기초하여 분리, Regression)

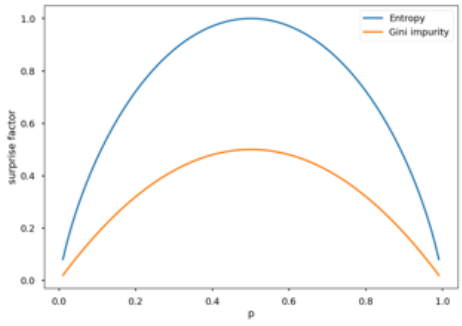
- F-test
- Reduction of Variance

- Gini Impurity

$$I_G(n) = 1 - \sum_{i=1}^{J} (p_i)^2$$

- Entropy

$$Entropy = \sum_{i=1}^{C} -p_i * \log_2(p_i)$$



- Variance

$$\text{Variance} = \frac{\Sigma(X - \overline{X})^2}{n}$$

엔트로피(Entropy)

- 데이터 분포의 불순도(impurity)를 수치화한 지표
- 서로 다른 데이터가 섞여 있으면 엔트로피가 높고 같은 값이 섞여 있으면 엔트로피 낮음
  - 수치 0: 모든 값이 동일 (분류하지 않아도 됨)
  - 수치 1: 불순도 최대

정보 이득 (Information Gain) 지수

- `1 - 엔트로피 지수` or `1 - 지니계수`
- 결정 트리는 정보 이득 지수가 높은 속성을 기준으로 분할

지니(Gini) 계수

- 불순도(impurity)를 수치화한 지표
  - 경제학에서 불평등 지수를 나타낼 때 사용하는 계수
  - 0이 가장 평등하고, 1로 갈수록 불평등
- 머신러닝에 적용될 때는 지니 계수가 낮을수록 데이터 균일도가 높은 것으로 해석
- 지니 계수가 낮은 속성을 기준으로 분할

- 지니 불순도가 엔트로피 보다 좀 더 계산이 빠름.
- 지니 불순도가 가장 빈도 높은 클래스를 한쪽 가지로 고립시키는 경향이 있음
- 엔트로피는 조금 더 균형 잡힌 트리를 생성

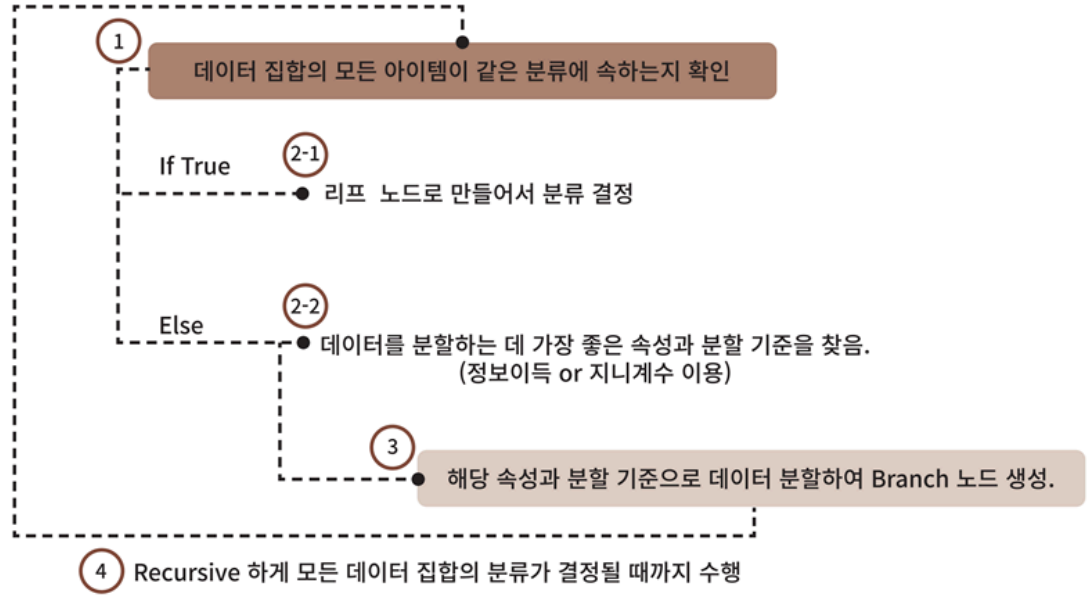
사이킷런의 결정 트리 알고리즘의 균일도 지표

- DecisionTreeClassifier의 criterion 기본값은 'gini'
- DecisionTreeRegressor는 'mse'가 기본값

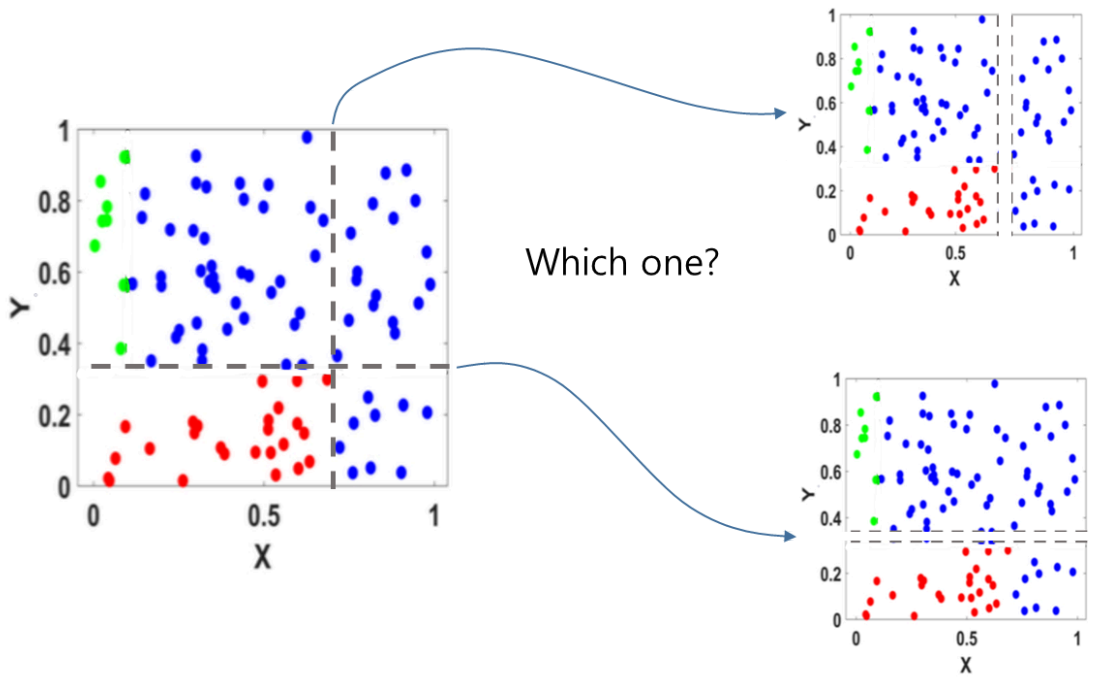
결정 트리 알고리즘에서 분류를 결정하는 과정

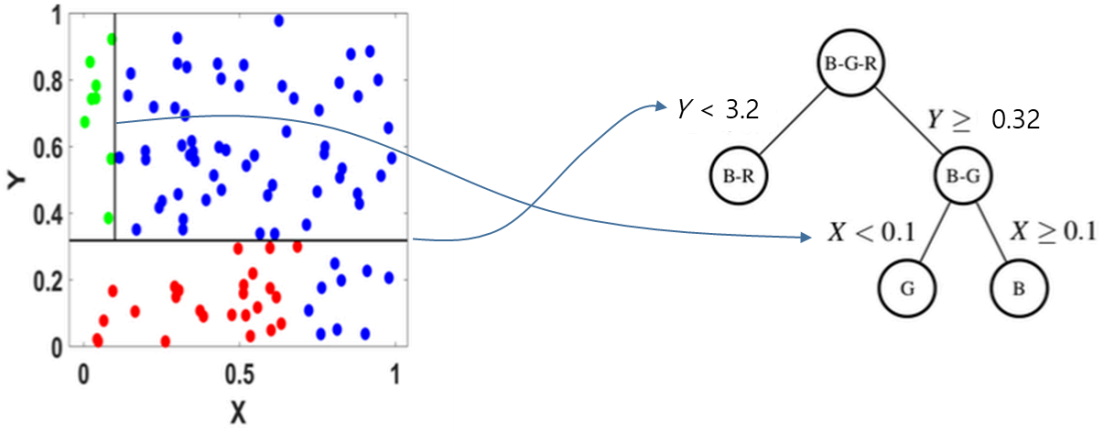
- 정보 이득이 높거나 지니 계수가 낮은 조건을 찾아서
- 자식 트리 노드에 걸쳐 반복적으로 분할한 뒤
- 데이터가 모두 특정 분류에 속하게 되면 분할을 멈추고 분류 결정

[분류 결정 과정]



파이썬 머신러닝 완벽 가이드, 권철민, 위키북스 , P187





▪ 예. Gini 계수를 이용하여 분리 기준 설정

X1 (성별)	X2 (학력)	Y (구매여부)
M	대졸	1
M	대졸	0
F	고졸	1
F	대졸	0
F	고졸	1

- 부모마디

$$imp(t) = 1 - \left(\frac{2}{5}\right)^2 - \left(\frac{3}{5}\right)^2 = 0.48$$

- 자식마디(X1으로 분리 : s1(M vs F))

$$imp(t_L) = 1 - \left(\frac{1}{2}\right)^2 - \left(\frac{1}{2}\right)^2 = 0.5$$

$$imp(t_R) = 1 - \left(\frac{1}{3}\right)^2 - \left(\frac{2}{3}\right)^2 = 0.4444$$

- 자식마디(X2로 분리 : s2(고졸 vs 대졸))

$$imp(t_L) = 1 - \left(\frac{0}{2}\right)^2 - \left(\frac{2}{2}\right)^2 = 0$$

$$imp(t_R) = 1 - \left(\frac{2}{3}\right)^2 - \left(\frac{1}{3}\right)^2 = 0.4444$$

X1 (성별)	X2 (학력)	Y (구매여부)
M	대졸	1
M	대졸	0
F	고졸	1
F	대졸	0
F	고졸	1

- 불순도 향상 정도(Goodness of Split) G(s,t)

$$G(s1, t) = 0.48 - \frac{2}{5} \times 0.5 - \frac{3}{5} \times 0.4444 = 0.01336$$

$$G(s2, t) = 0.48 - \frac{2}{5} \times 0 - \frac{3}{5} \times 0.4444 = 0.21336$$

→ 불순도 향상 정도가 더 큰 s2를 기준으로 분리

결정트리의 주요 알고리즘

	분리기준	특징
CART	• 분류나무 : 지니불순도 • 회귀나무 : 분산감소량	• 항상 이진분리 • 개별 특성변수 및 특성변수의 선형결합 형태의 분리기준도 가능
C4.5 C5.0	• 엔트로피 불순도로 구한 정보이득	• 범주형 특성변수는 다진분리 • 연속형 특성변수는 이진분리
CHAID	• 분류나무: 카이제곱통계량 • 회귀나무: ANOVA, F통계량	• 다진분리 • 변수간 통계적 관계에 기반

결정트리의 장단점

장점

- 시각화를 통한 쉬운 이해(if-then-else)
- 자료 가공이 거의 필요 없음
- 수치형 및 범주형 데이터 모두 적용 가능
- 선형성, 정규성 등의 가정이 필요 없는 비모수적 방법
- 대량 데이터 처리에도 적합
- 스케일에 민감하지 않으며, 이상치에 대해 상대적으로 덜 민감
- 일부 알고리즘(CART 등)의 경우 결측치 대응 방법 포함

단점

- 휴리스틱에 근거한 실용적 알고리즘으로 전역 최적화를 얻지 못할 수 있음
- 과적합이 쉽게 발생
- 자료에 따라 불안정함 (특히 적은 자료, Class 수에 비교하여 학습 데이터가 적은 규모이면 높은 분류 에러 발생)
- Feature간의 복잡한 관계를 처리하지는 못함
- 자료가 복잡하면 실행 시간이 급격히 증가
- 모든 분할은 축에 수직

사이킷런의 결정트리 알고리즘 클래스

- 분류와 회귀모델 모두 CART(Classification And Regression Tree) 알고리즘 기반

분류모델을 위한 `DecisionTreeClassifier`

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)

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

회귀모델을 위한 `DecisionTreeRegressor`

DecisionTreeRegressor(*, criterion='squared_error', 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, ccp_alpha=0.0)

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

결정 트리를 위한 클래스의 매개변수

- 분류, 회귀 모두 동일하게 적용

파라미터 명	설명
min_samples_split	• 노드를 분할하기 위한 최소한의 샘플 데이터 수로 과적합을 제어하는 데 사용됨. • 디폴트는 2이고 작게 설정할수록 분할되는 노드가 많아져서 과적합 가능성 증가 • 과적합을 제어. 1로 설정할 경우 분할되는 노드가 많아져서 과적합 가능성 증가
min_samples_leaf	• 말단 노드(Leaf)가 되기 위한 최소한의 샘플 데이터 수 • Min_samples_split와 유사하게 과적합 제어 용도. 그러나 비대칭적(imbalanced) 데이터의 경우 특정 클래스의 데이터가 극도로 작을 수 있으므로 이 경우는 작게 설정 필요.
max_features	• 최적의 분할을 위해 고려할 최대 피처 개수. 디폴트는 None으로 데이터 세트의 모든 피처를 사용해 분할 수행. • int 형으로 지정하면 대상 피처의 개수. float 형으로 지정하면 전체 피처 중 대상 피처의 퍼센트임 • 'sqrt'는 전체 피처 중 sqrt(전체 피처 개수), 즉 $\sqrt{\text{전체 피처 개수}}$만큼 선정 • 'auto'로 지정하면 sqrt와 동일 • 'log'는 전체 피처 중 log2(전체 피처 개수) 선정 • 'None'은 전체 피처 선정
max_depth	• 트리의 최대 깊이를 규정. • 디폴트는 None. None으로 설정하면 완벽하게 클래스 결정 값이 될 때까지 깊이를 계속 키우며 분할하거나 노드가 가지는 데이터 개수가 min_samples_split보다 작아질 때까지 계속 깊이를 증가시킴. • 깊이가 깊어지면 min_samples_split 설정대로 최대 분할하여 과적합할 수 있으므로 적절한 값으로 제어 필요.
max_leaf_nodes	• 말단 노드(Leaf)의 최대 개수

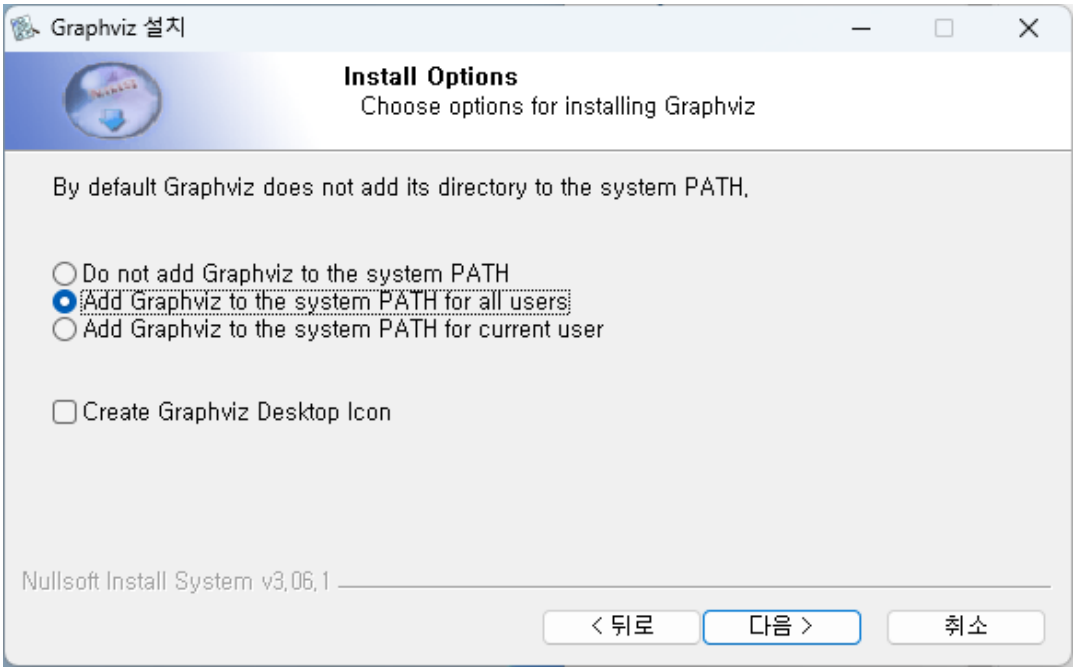
> # 결정 트리 모델의 시각화(Decision Tree Visualization)

Graphviz 패키지 사용

- 사이킷런에서 export_graphviz() API 제공
- 학습된 결정 트리 규칙을 실제 트리 형태로 시각화

Graphviz 설치

1. Graphviz 파이선 래퍼 모듈 설치 (윈도우버전)
    - https://graphviz.org/download/
    - graphviz-...exe 다운로드 (최신버전)



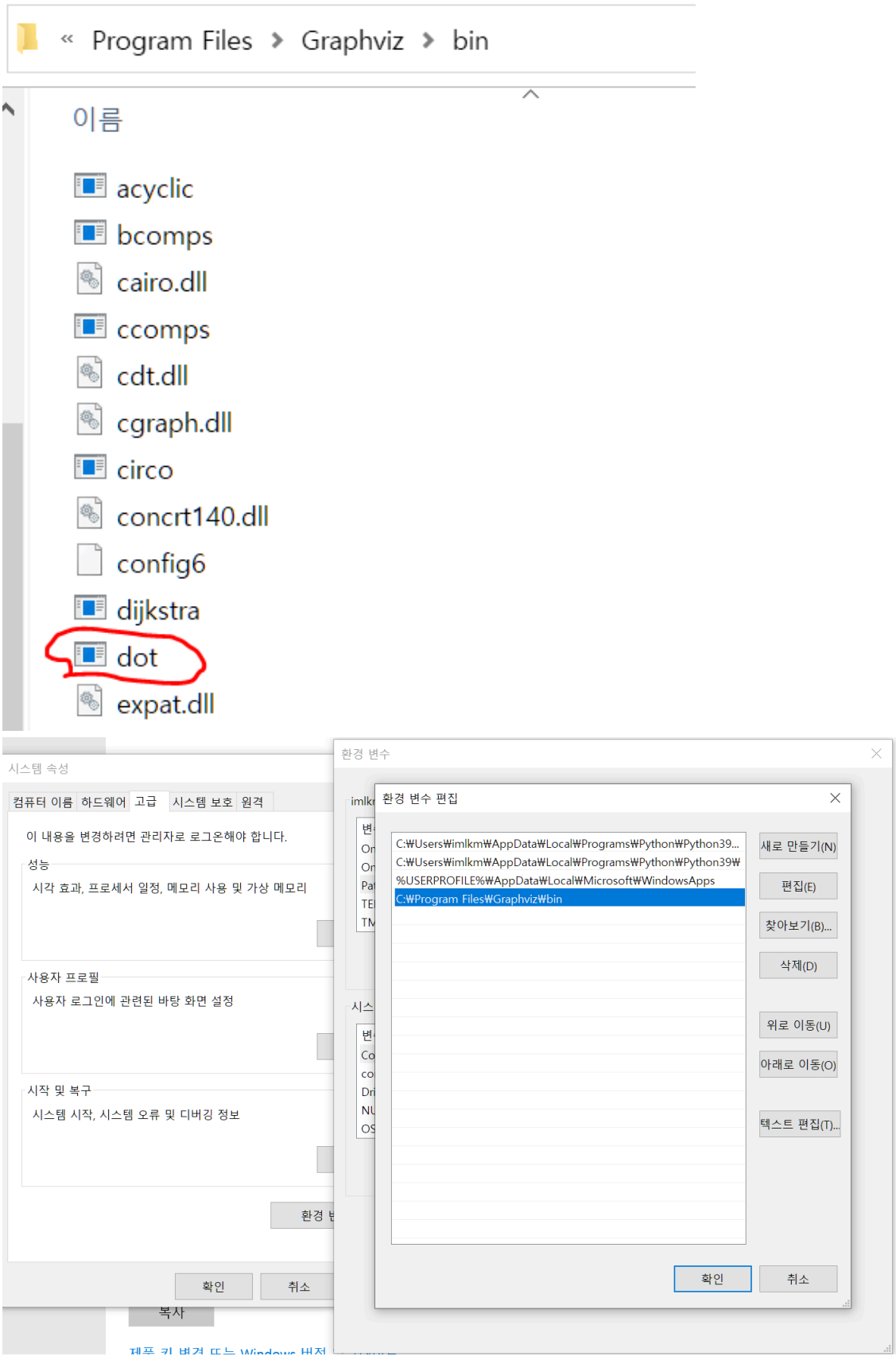
2. Graphviz 설치 후 Graphviz 파이썬 래퍼 모듈을 pip로 설치
    - pip install graphviz
    - (아나콘다 Command 콘솔 생성 시 관리자권한으로 실행)
        - conda install graphviz

```
(base) C:₩windows₩system32>pip install graphviz
Collecting graphviz
  Downloading https://files.pythonhosted.org/packages/1f/e2/ef2581b5b86625657afd32030f90cf2717456c1d2b711ba074bf007c0f1a
/graphviz-0.10.1-py2.py3-none-any.whl
Installing collected packages: graphviz
Successfully installed graphviz-0.10.1
```

3. 환경변수 설정

    - Graphviz가 설치된 경로 : C:₩Program Files₩Graphviz₩bin
    - '내 PC'이 속성 -> 고급시스템설정 -> 환경변수 클릭
    - Path와 시스템 변수 Path에 경로 추가

4. 주피터노트북 서버 프로그램 다시 실행



In [6]:
```
!pip install graphviz
```

```
Defaulting to user installation because normal site-packages is not writeable
Collecting graphviz
  Downloading graphviz-0.20.3-py3-none-any.whl.metadata (12 kB)
Downloading graphviz-0.20.3-py3-none-any.whl (47 kB)
   ---------------------------------------- 0.0/47.1 kB ? eta -:--:--
   ------------------------ ------------- 30.7/47.1 kB 660.6 kB/s eta 0:00:01
   ---------------------------------------- 47.1/47.1 kB 594.9 kB/s eta 0:00:00
Installing collected packages: graphviz
Successfully installed graphviz-0.20.3
```

In [3]:
```
# !pip list
```

sklearn.tree.export_graphviz()함수

- 사이킷런 tree 모듈에 Graphviz를 이용하기 위한 export_graphviz() 함수 제공
- https://scikit-learn.org/stable/modules/generated/sklearn.tree.export_graphviz.html

export_graphviz(decision_tree, out_file=None, *, max_depth=None, feature_names=None, class_names=None, label='all', filled=False, leaves_parallel=False, impurity=True, node_ids=False, proportion=False, rotate=False, rounded=False, special_characters=False, precision=3, fontname='helvetica')

- 매개변수
  - decision_tree : 학습이 완료된 estimator
  - out_file : output 파일명
  - class_names : 결정 클래스 명칭
  - feature_names : 피처 이름
  - impurity=True : 각 노드에 불순도 표시 (gini 계수) (True: 디폴트)
  - filled=True : 노드 색상 표시 (False: 디폴트)

In [8]:
```
from sklearn.tree import DecisionTreeClassifier
from sklearn.datasets import load_iris
from sklearn.model_selection import train_test_split
import warnings
warnings.filterwarnings('ignore')

from sklearn.tree import export_graphviz
```

In [9]:
```
iris = load_iris()
X = iris.data
y = iris.target

dt_clf = DecisionTreeClassifier(random_state=156)

X_train, X_test, y_train, y_test = train_test_split(X, y,
                                                    test_size=0.2,
                                                    random_state=11)

dt_clf.fit(X_train, y_train)
```

Out[9]:
```
▾ DecisionTreeClassifier
DecisionTreeClassifier(random_state=156)
```

In [10]:
```
export_graphviz(dt_clf, out_file='output/tree.dot',
                class_names=iris.target_names,
                feature_names=iris.feature_names,
                impurity=True, filled=True)
```

In [11]:
```
import graphviz

with open('output/tree.dot') as f:
    dot_graph = f.read()
graphviz.Source(dot_graph)
```

Out[11]: