Tree와 랜덤 포레스트

Tree와 랜덤 포레스트

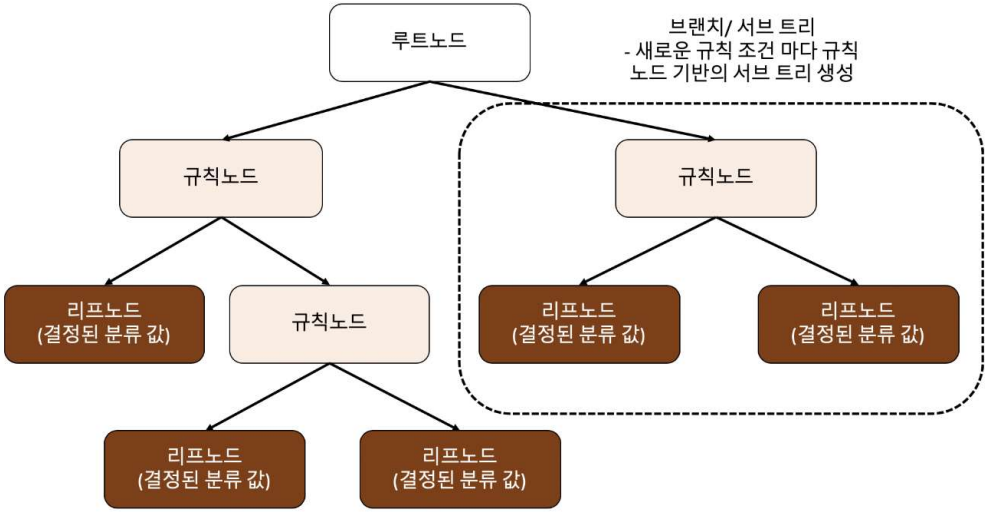
- 분류
  - 다양한 머신러닝 알고리즘으로 구현할 수 있습니다

  - 베이즈(Bayes)통계와 생성 모델에 기반한 나이브 베이즈(Naive Bayes)
  - 독립변수와 종속변수의 선형 관계성에 기반한 로지스틱 회귀(Logistic Regression)
  - 데이터 균일도에 따른 규칙 기반의 결정 트리(Decision Tree)
  - 개별 클래스 간의 최대 분류 마진을 효과적으로 찾아주는 서포트 벡터 머신(Support Vector Machine)
  - 근접 거리를 기준으로 하는 최소 근접(Nearest Neighbor) 알고리즘
  - 심층 연결 기반의 신경망(Neural Network)
  - 서로 다른 (또는 같은) 머신러닝 알고리즘을 결합한 앙상블(Ensemble)

  - 결정 트리는 매우 쉽고 유연하게 적용될 수 있는 알고리즘입니다.
  - 데이터의 스케일링이나 정규화 등의 사전 가공의 영향이 매우 적습니다.
  - 예측 성능을 향상시키기 위해 복잡한 규칙 구조를 가져야 하며, 이로 인한 과적합(overfitting)이 발생해 반대로 예측 성능이 저하될 수 있다는 단점이 있습니다.

Tree와 랜덤 포레스트

- 결정 트리(Decision Tree)
  - 데이터에 있는 규칙을 학습을 통해 자동으로 찾아내 트리(tree) 기반의 분류 규칙을 만드는 것입니다.
  - if/else 기반으로 예측을 위한 규칙을 만드는 알고리즘
  - 데이터의 어떤 기준을 바탕으로 규칙을 만들어야 가장 효율적인 분류가 될 것인가가 알고리즘의 성능을 크게 좌우한다
  - 의사결정나무(decision tree)는 여러 가지 규칙을 순차적으로 적용하면서 독립 변수 공간을 분할
  - 데이터 불순도란 데이터가 제대로 분류되지 않고 섞여 있는 정도를 의미합니다.
  - 정보이득 IG는 자식노드의 데이터 불순도가 작으면 작을수록 커지게 됩니다.
  - 계산을 단순화하기 위해 보통 의사결정트리에서 분기되는 자식 노드의 개수는 2개로 하는 이진 의사결정트리(Binary decision tree)라고 부릅니다.



Tree와 랜덤 포레스트

- 결정 트리(Decision Tree)
  - 최대한 균일한 데이터 세트를 구성할 수 있도록 분할 하는 것이 필요
  - 결정 노드는 정보 균일도가 높은 데이터 세트를 먼저 선택할 수 있도록 규칙 조건을 만듭니다.
  - 정보의 균일도를 측정하는 대표적인 방법은 엔트로피를 이용한 정보 이득 지수와 지니 계수가 있습니다.
  - 정보 이득은 엔트로피라는 개념을 기반으로 합니다.
  - 엔트로피는 주어진 데이터 집합의 혼잡도를 의미하며,
  - 서로 다른 값이 섞여 있으면 엔트로피가 높고, 같은 값이 섞여 있으면 엔트로피가 낮습니다.
  - 정보 이득 지수는 1에서 엔트로피 지수를 뺀 값입니다.
  - 1-엔트로피 지수입니다. 결정 트리는 이 정보 이득 지수로 분할 기준을 정합니다. 즉, 정보 이득이 높은 속성을 기준으로 분할합니다.

Tree와 랜덤 포레스트

- 결정 트리(Decision Tree)
  - 확률론에서의 엔트로피는 확률분포의 모양을 설명하는 특정값이며 확률분포가 가지고 있는 정보의 양을 나타내는 값
  - 확률분포가 가지는 정보의 확신도 혹은 정보량을 수치로 표현한 것
  - 확률 또는 확률밀도가 특정값에 몰려 있으면 엔트로피가 작다고 하고 반대로 여러가지 값에 골고루 퍼져 있다면 엔트로피가 크다고 한다.

  - 지니계수는 경제학에서 불평등 지수를 나타낼 때 사용하는 계수입니다.
  - 지니 계수는 0이 가장 평등하고 1로 갈수록 불평등합니다.
  - 머신러닝에 적용될 때는 지니 계수가 낮을 수록 데이터 균일도가 높은 것으로 해석해 지니 계수가 낮은 속성을 기준으로 분할합니다.

  - 지니 계수를 이용해 데이터 세트를 분할합니다.
  - 정보 이득이 높거나 지니 계수가 낮은 조건을 찾아서 지식 트리 노드에 걸쳐 반복적으로 분할한 뒤 데이터가 모두 특정 분류에 속하게 되면 분할을 멈추고 분류를 결정합니다.

  - 장점 - 해석력이 높음. 직관적.범용성. 피처의 스케일링이나 정규화 등의 사전 가공 영향도가 크지 않습니다.
  - 단점 - 높은 변동성. 샘플에 민감할 수 있음.

  - 과적합으로 알고리즘 성능이 떨어진다. 이를 극복하기 위해 트리의 크기를 사전에 제한하는 튜닝 필요

Tree와 랜덤 포레스트

- 결정 트리(Decision Tree)
  - 결정 트리 파라미터

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(Leaf) 노드가 되기 위한 최소한의 샘플 데이터 수 과적합 제어 용도 비대칭 데이터의 경우 특정 클래스의 데이터가 극도록 작을 수 있으므로 이 경우 작게 설정 필요
max_features	최적의 분할을 위해 고려할 최대 피처 개수 디폴트 None으로 데이터 세트의 모든 피처를 사용해 분할 수행 int형으로 지정하면 대상 피처의 개수 float 형으로 지정하면 전체 피처 중 대상 피처의 퍼센트 sqrt, auto, log, None
max_depth	트리의 최대 깊이를 규정 디폴트는 None, None으로 설정하면 완벽하게 클래스 결정 값이 될 때까지 깊이를 계속 기우며 분할하거나 노드가 가지는 데이터 개수 깊이가 깊어지면 min_samples_split 설정대로 최대 분할하여 과적합할 수 있으므로 적절한 값으로 제어 필요
max_leaf_nodes	말단 노드(Leaf)의 최대 개수

Tree와 랜덤 포레스트

- 결정 트리 분류기 훈련
  - 의사결정트리 분류기는 일련의 질문에 근거하여 주어진 데이터를 분류해주는 알고리즘입니다.
  - 의사결정트리 학습은 트레이닝 데이터를 이용해 데이터를 최적으로 분류해주는 질문들을 학습하는 머신러닝입니다.
  - 의사결정트리 학습에서 각 노드에서 분기하기 위한 최적의 질문은 정보이득(Information Gain)이라는 값이 최대가 되도록 만들어주는 것이 핵심입니다.
  - 어느 특정 노드에서 m개의 자식 노드로 분기되는 경우 정보이득은 다음의 식으로 정의합니다.
  - $$IG(D_p, f) = I(D_p) - \sum_{j=1}^{m} \frac{N_j}{N_p} I(D_j)$$
  - 데이터 불순도란 데이터가 제대로 분류되지 않고 섞여 있는 정도를 의미합니다.
  - 정보이득 IG는 자식노드의 데이터 불순도가 작으면 작을수록 커지게 됩니다.
  - 계산을 단순화하기 위해 보통 의사결정트리에서 분기되는 자식 노드의 개수는 2개로 하는 이진 의사결정트리(Binary decision tree)라고 부릅니다.

$$IG(D_p, f) = I(D_p) - \frac{N_L}{N_p} I(D_L) - \frac{N_R}{N_p} I(D_R)$$

  - 데이터 불순도를 측정하는 방법 - 지니 인덱스(Gini Index), 엔트로피(entropy), 분류오류(classification error)

Tree와 랜덤 포레스트

- 결정 트리 분류기 훈련
  - 결정 트리 학습기는 노드에서 불순도가 가장 크게 감소하는 결정 규칙을 찾습니다.
  - DecisionTreeClassifier는 기본적으로 지니 불순도를 사용합니다.
  - 불순도를 낮추는 결정 규칙을 찾는 과정은 모든 리프 노드(leaf node)가 순수해지거나(즉, 한 클래스만 남거나) 어떤 임곗값에 도달할 때까지 반복됩니다.

```
from sklearn.tree import DecisionTreeClassifier
from sklearn import datasets

iris = datasets.load_iris()  # 데이터 로드
features = iris.data
target = iris.target

decisiontree = DecisionTreeClassifier(random_state=0)  # 결정 트리 분류기 객체 생성
model = decisiontree.fit(features, target)  # 모델 훈련

observation = [[ 5,  4,  3,  2]]  # New 샘플 데이터
model.predict(observation)  # 샘플 데이터의 클래스 예측
model.predict_proba(observation)  # 세 개의 클래스에 대한 예측 확률을 확인

# 엔트로피를 사용해 결정 트리 분류기를 훈련합니다.
decisiontree_entropy = DecisionTreeClassifier( criterion='entropy', random_state=0)
model_entropy = decisiontree_entropy.fit(features, target)   # 모델 훈련
```

Tree와 랜덤 포레스트

- 결정 트리 분류기 훈련
  - 데이터 불순도를 측정하는 방법 - 지니 인덱스(Gini Index), 엔트로피(entropy), 분류오류(classification error)

지니 인덱스

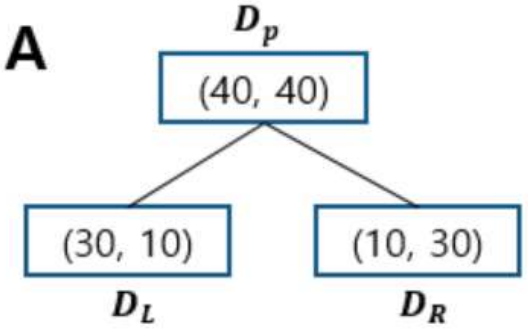
$$I_G(t) = 1 - \sum_{i=1}^{c} p(i\backslash t)^2$$

엔트로피

$$I_H(t) = -\sum_{i=1}^{c} p(i\backslash t)\log_2 p(i\backslash t)$$

분류오류

$$I_E(t) = 1 - \max\{p(i\backslash t)\}$$



의사결정트리 A의 경우 불순도 *I(Dp), I(DL), I(DR)* 및 정보이득 *IG* 계산

$$I_E(D_p) = 1 - 0.5 = 0.5$$

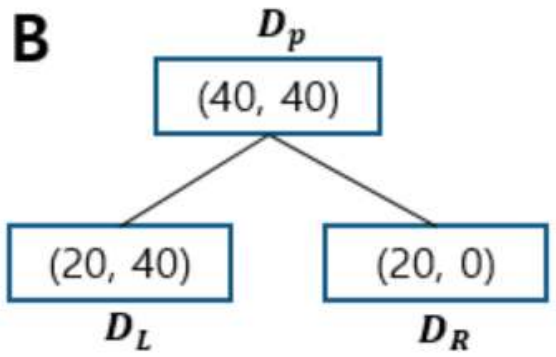
$$I_E(D_L) = 1 - \frac{3}{4} = 0.25$$

$$I_E(D_R) = 1 - \frac{3}{4} = 0.25$$

$$IG_E = 0.5 - \frac{40}{80} \times 0.25 - \frac{40}{80} \times 0.25 = 0.25$$

Tree와 랜덤 포레스트

➢ 결정 트리 분류기 훈련



의사결정트리 B의 경우 불순도 *I(Dp), I(DL), I(DR)* 및 정보이득 *IG* 계산

$$I_E(D_p) = 1 - 0.5 = 0.5$$

$$I_E(D_L) = 1 - \frac{4}{6} = \frac{1}{3}$$

$$I_E(D_R) = 1 - 1 = 0$$

$$IG_E = 0.5 - \frac{60}{80} \times \frac{1}{3} - 0 = 0.25$$

지니 인덱스로 계산한 경우
A에서 정보이득 : 0.125
B에서 정보이득 : 약 0.16

엔트로피로 계산한 경우
A에서 정보이득 : 0.19
B에서 정보이득 : 0.31

- 의사결정트리에서 정보이득이 최대가 되는 것을 채택해야 하므로, 분류오류를 적용하였을 때는 A, B가 모두 같은 값이 나와서 어떤 것을 선택하더라도 무관하지만, 지니 인덱스로 계산하거나 엔트로피로 계산한 경우에는 B가 A보다 정보 이득 값이 크므로 B를 선택하게 될것입니다.

Tree와 랜덤 포레스트

- 결정 트리 분류기 훈련
  - 지니 불순도는 클래스가 균등하게 분포되어 있을 때 최대가 됩니다.
  - 이진 클래스일 경우 클래스 샘플 비율이 0.5일 때 가장 큰 값이 됩니다.
  - 엔트로피도 클래스 샘플 비율이 균등할 때 가장 큰 값이 됩니다.

  - 지니 인덱스, 엔트로피, 분류오류 3가지 불순도 계산 방법 모두 0 또는 1에 가까와질 수록 불순도가 낮아지고, 0.5일 때 분순도가 가장 높게 나옵니다.
  - 데이터 세트에 여러 가지 부류에 속하는 데이터가 섞여 있는 경우, 특정 부류에 속하는 멤버입장에서 고려할 때 그 멤버가 차지하는 비율이 0에 가까와지거나 1에 가까와 질 때 가장 순도가 높고, 0.5일 때 분순도가 가장 높다..고 해석 됩니다.
  - criterion 매개변수는 불순도 계산 방법을 설정합니다. ('entropy' , 'gini')

Tree와 랜덤 포레스트

- 결정 트리 분류기 훈련
  - 트리 기반 학습 알고리즘은 분류와 회귀에서 사용되는 비모수 지도 학습 방법입니다.
  - 트리 기반 학습기의 기본은 일련의 결정 규칙이 연결된 결정 트리입니다.
  - 결정 규칙이 맨 위에 있고 이어지는 결정 규칙이 아래로 퍼져 있습니다.
  - 결정 트리에서 모든 결정 규칙은 결정 노드에서 일어납니다
  - 결정 규칙이 없는 마지막 가지를 리프라고 부릅니다.
  - 트리 기반 모델은 이해하기 쉽습니다.
  - 사이킷런의 DecisionTreeClassifier를 사용합니다.
  - 결정 트리 학습기는 노드에서 불순도가 가장 크게 감소하는 결정 규칙을 찾습니다.
  - DecisionTreeClassifier는 지니 불순도를 기본으로 사용합니다.

$$G(t) = 1 - \sum_{i=1}^{c} p_i^2$$

  - G(t)는 노드 t에서 지니 불순도이고 pi는 노드 t에서 클래스 c의 샘플 비율입니다
  - 불순도를 낮추는 결정 규칙을 찾는 과정은 모든가 순수해지거나 어떤 임계값에 도달할 때까지 반복됩니다.
  - 지니 불순도는 클래스가 균등하게 분포도어 있을 때 최대가 됩니다.
  - 예] 이진 클래스일 경우 클래스 샘플 비율이 0.5일때 가장 큰 값이 됩니다
  - 엔트로피 불순도 공식

$$H(t) = -\sum_{i=1}^{c} p_i \log_2 p_i$$

  - 엔트로피도 클래스 샘플 비율이 균등할 때 가장 큰 값이 됩니다.

  - 주어진 학습 데이터에 따라 생성되는 의사결정트리가 매우 달라져서 일반화하여 사용하기 어렵고 의사 결정 트리를 이용한 학습 결과 역시 성능과 변동의 폭이 크다는 단점을 가지고 있습니다.

Tree와 랜덤 포레스트

➢ 결정 트리 회귀 훈련

- 결정 트리 회귀는 지니 불순도나 엔트로피를 감소하는 대신 기본적으로 얼마나 평균 제곱 오차(MSE)를 감소시키는지에 따라 분할합니다.
- 사이킷런에서는 DecisionTreeRegressor를 사용하여 결정 트리 회귀를 수행할 수 있습니다
- criterion매개변수를 사용하여 분할 품질의 측정 방식을 선택할 수 있습니다.

$$MSE = \frac{1}{n}\sum_{i=1}^{n}(y_i - \bar{y})^2$$

```
from sklearn.tree import DecisionTreeRegressor
from sklearn import datasets

boston = datasets.load_boston()   # 데이터 로드
features = boston.data[:,0:2]   #두 개의 특성만 선택
target = boston.target

decisiontree = DecisionTreeRegressor(random_state=0)  # 결정 트리 회귀 모델 객체 생성
model = decisiontree.fit(features, target)   # 모델 훈련

observation = [[0.02, 16]]   #New 샘플 데이터
model.predict(observation)   # 샘플 데이터의 타깃을 예측

# 평균 제곱 오차를 사용한 (평균 절댓값 오차MAE가 감소되는) 결정 트리 회귀 모델 객체 생성
decisiontree_mae = DecisionTreeRegressor(criterion="mae", random_state=0)
model_mae = decisiontree_mae.fit(features, target)   # 모델 훈련
```

$$MAE = \frac{1}{n}\sum_{i=1}^{n}|y_i - \bar{y}|$$

Tree와 랜덤 포레스트

- 결정 트리 회귀 훈련
  - 사이킷런의 DecisionTreeRegressor를 사용합니다.
  - 트리 기반 학습 알고리즘은 분류와 회귀에서 사용되는 비모수 지도 학습 방법입니다
  - 지니 불순도나 엔트로피를 감소하는 대신 기본적으로 얼마나 평균 제곱 오차(MSE)를 감소시키는지에 따라 분할합니다.
  - criterion 매개변수 - 품질의 측정 방식을 선택할 수 있습니다.

$$\mathrm{MSE} = \frac{1}{n}\sum_{i=1}^{n}\left(y_i - \bar{y}\right)^2$$

  - 평균 절댓값

$$\mathrm{MAE} = \frac{1}{n}\sum_{i=1}^{n}\left|y_i - \bar{y}\right|$$

  - friendman_mse 방식 - 왼쪽 노드와 오른쪽 노드의 평균을 비교합니다.

$$\mathrm{FriedmanMSE} = \frac{n_{left} \times n_{right}\left(\mu_{left} - \mu_{right}\right)^2}{n_{left} + n_{right}}$$

Tree와 랜덤 포레스트

- 결정 트리 모델 시각화
  - Graphviz 패키지
  - export_graphviz() API를 제공
  - https://graphviz.gitlab.io/_pages/Download/Download_windows.html
  - 환경변수 설정

  - 사이킷런의 export_graphviz()는 함수 인자로 학습이 완료된 Estimator, 피처의 이름 리스트, 레이블 이름 리스트를 입력하면 학습된 결정 트리 규칙을 실제 트리 형태로 시각화

  - 피처의 중요한 역할 지표를 DecisionTreeClassifier 객체의 feature_importances_속성으로 제공합니다.

Tree와 랜덤 포레스트

➢ 결정 트리 모델 시각화

- 결정 트리 분류기의 장점은 훈련된 전체 모델을 시각화할 수 있다는 것이다.
- 훈련된 모델을 DOT 포맷으로 변환한 다음 그래프를 그립니다.

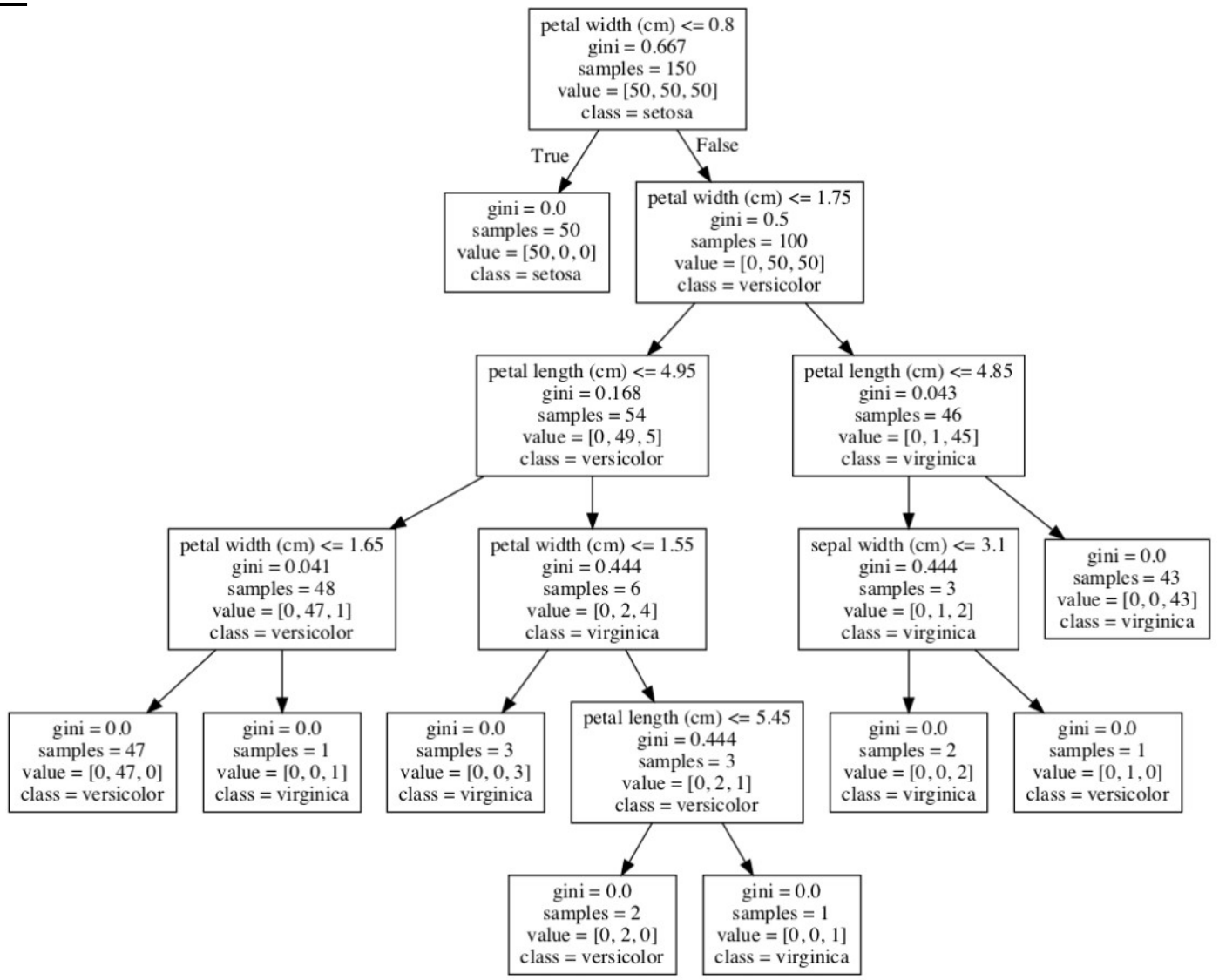
```
import pydotplus
from sklearn.tree import DecisionTreeClassifier
from sklearn import datasets
from IPython.display import Image
from sklearn import tree
iris = datasets.load_iris() # 데이터 로드
features = iris.data
target = iris.target
decisiontree = DecisionTreeClassifier(random_state=0) # 결정 트리 분류기를 만듭니다.

model = decisiontree.fit(features, target) # 모델 훈련
dot_data = tree.export_graphviz(decisiontree,
                                out_file=None,
                                feature_names=iris.feature_names,
                                class_names=iris.target_names) # DOT 데이터를 만듭니다

graph = pydotplus.graph_from_dot_data(dot_data) # 그래프를 그립니다.
Image(graph.create_png()) # 그래프 출력
graph.write_pdf("iris.pdf") # PDF를 만듭니다.
graph.write_png("iris.png") # PNG 파일을 만듭니다
```

Tree와 랜덤 포레스트

- 결정 트리 모델 시각화



Tree와 랜덤 포레스트

- 결정 트리 모델 시각화
  - export_graphviz()의 filled 매개변수를 True로 지정하면 노드마다 다수의 클래스에 따라 색이 채워집니다.
  - round 매개변수를 True로 지정하면 노드의 모서리를 라운드 처리합니다.
  - matplotlib 기반의 트리 그래프를 그려주는 plot_tree() 는 적절한 그래프 크기를 정의합니다.

```
import matplotlib.pyplot as plt

plt.figure(figsize=(20, 15))
tree.plot_tree(model, filled=True,
               feature_names=iris.feature_names,
               class_names=iris.target_names,
               rounded=True, fontsize=14)
plt.show()
```

Tree와 랜덤 포레스트

- 결정 트리 모델 시각화