

# 결정트리(decision tree)

## 학습 내용

- 01 의사결정트리 모델을 생성해 보기
- 02 트리의 특성 중요도 알아보고 시각화 해보기
- 03 의사결정트리의 범주형/연속형 적용해보기
- 04 과적합을 해결해 보기

## 01 의사결정트리 기본

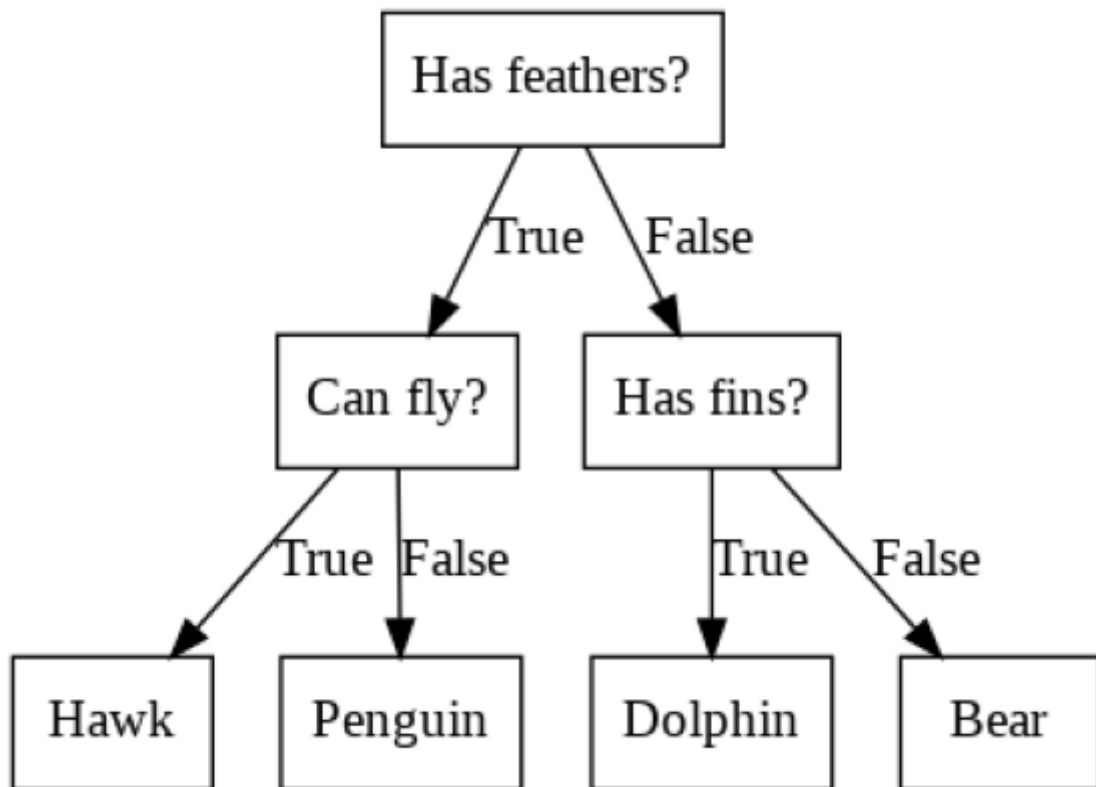
- (가) decision tree는 classification(분류)와 regression(회귀) 문제에 널리 사용하는 모델이다.
- (나) 스무고개 놀이의 질문과 비슷하다.

```
In [1]: # mglearn 설치 필요
        # !pip install mglearn

Collecting mglearn
  Downloading mglearn-0.1.9.tar.gz (540 kB)
    |████████████████████████████████████████| 540 kB 7.0 MB/s
Requirement already satisfied: numpy in /usr/local/lib/python3.7/dist-packages (from mglearn) (1.21.6)
Requirement already satisfied: matplotlib in /usr/local/lib/python3.7/dist-packages (from mglearn) (3.2.2)
Requirement already satisfied: scikit-learn in /usr/local/lib/python3.7/dist-packages (from mglearn) (1.0.2)
Requirement already satisfied: pandas in /usr/local/lib/python3.7/dist-packages (from mglearn) (1.3.5)
Requirement already satisfied: pillow in /usr/local/lib/python3.7/dist-packages (from mglearn) (7.1.2)
Requirement already satisfied: cyclo in /usr/local/lib/python3.7/dist-packages (from mglearn) (0.11.0)
Requirement already satisfied: imageio in /usr/local/lib/python3.7/dist-packages (from mglearn) (2.4.1)
Requirement already satisfied: joblib in /usr/local/lib/python3.7/dist-packages (from mglearn) (1.1.0)
Requirement already satisfied: kiwisolver>=1.0.1 in /usr/local/lib/python3.7/dist-packages (from matplotlib->mglearn) (1.4.2)
Requirement already satisfied: pyparsing!=2.0.4,!=2.1.2,!=2.1.6,>=2.0.1 in /usr/local/lib/python3.7/dist-packages (from matplotlib->mglearn) (3.0.8)
Requirement already satisfied: python-dateutil>=2.1 in /usr/local/lib/python3.7/dist-packages (from matplotlib->mglearn) (2.8.2)
Requirement already satisfied: typing-extensions in /usr/local/lib/python3.7/dist-packages (from kiwisolver>=1.0.1->matplotlib->mglearn) (4.2.0)
Requirement already satisfied: six>=1.5 in /usr/local/lib/python3.7/dist-packages (from python-dateutil>=2.1->matplotlib->mglearn) (1.15.0)
Requirement already satisfied: pytz>=2017.3 in /usr/local/lib/python3.7/dist-packages (from pandas->mglearn) (2022.1)
Requirement already satisfied: threadpoolctl>=2.0.0 in /usr/local/lib/python3.7/dist-packages (from scikit-learn->mglearn) (3.1.0)
Requirement already satisfied: scipy>=1.1.0 in /usr/local/lib/python3.7/dist-packages (from scikit-learn->mglearn) (1.4.1)
Building wheels for collected packages: mglearn
  Building wheel for mglearn (setup.py) ... done
  Created wheel for mglearn: filename=mglearn-0.1.9-py2.py3-none-any.whl size=582639 sha256=f205d10bbb5511c252273e58983f75aeb6975a0d870335da6ef8d39dc9684b4f
  Stored in directory: /root/.cache/pip/wheels/f1/17/e1/1720d6dcd70187b6b6c3750cb3508798f2b1d57c9d3214b08b
Successfully built mglearn
Installing collected packages: mglearn
Successfully installed mglearn-0.1.9
```

```
In [2]: import matplotlib.pyplot as plt
import mglearn
```

```
In [3]: plt.figure(figsize=(10,10))
mglearn.plots.plot_animal_tree()
```



- (가) 트리에 사용되는 세 개의 feature가 있음.
  - 'Has feathers?'(날개가 있나요?)
  - 'Can fly?'(날수 있나요?)
  - 'Has fins?'(지느러미가 있나요?)
- (나) 이 머신러닝 문제는 네 개의 클래스로 구분하는 모델을 생성
  - 네 개의 클래스 - 매, 펭귄, 돌고래, 곰
- (다) 노드 종류
  - 맨 위의 노드 - **Root Node(루트 노드)**
  - 맨 마지막 노드 - **Leaf Node(리프 노드)**
  - target가 하나로만 이루어진 Leaf Node(리프 노드) **순수 노드 (pure node)**
- (라) 노드 분기(각 노드)
  - 범주형은 데이터를 구분하는 질문을 통해 나눈다.
  - 연속형은 특성 **i**가 **a**보다 큰가?의 질문으로 나눈다.

## 02 의사결정 트리 구축

```
In [4]: from sklearn.tree import DecisionTreeClassifier
from sklearn.model_selection import train_test_split
from sklearn.datasets import load_breast_cancer
import seaborn as sns
```

```
In [5]: cancer = load_breast_cancer()
X = cancer.data
y = cancer.target
```

```
X_train, X_test, y_train, y_test = train_test_split(X, y,
                                                    stratify=cancer.target,
                                                    test_size = 0.3,
                                                    random_state=77)
```

```
In [6]: tree = DecisionTreeClassifier(max_depth=2, random_state=0)
        tree.fit(X_train, y_train)

        print("훈련 세트 정확도 : {:.3f}".format(tree.score(X_train, y_train)))
        print("테스트 세트 정확도 : {:.3f}".format(tree.score(X_test, y_test)))
```

```
훈련 세트 정확도 : 0.972
테스트 세트 정확도 : 0.912
```

### 03 의사 결정 트리 복잡도 변경

- 결정트리의 깊이를 제한하지 않으면 트리는 무작정 깊어지고 복잡해 질 수 있다.
- 첫번째는 트리가 일정 깊이에 도달하면 트리의 성장을 멈추게 하는 것.
  - max\_depth를 이용

```
In [7]: for i in range(1,7,1):
        tree = DecisionTreeClassifier(max_depth=i, random_state=0)
        tree.fit(X_train, y_train)
        print(f"max_depth : {i}")
        print("훈련 세트 정확도 : {:.3f}".format(tree.score(X_train, y_train)))
        print("테스트 세트 정확도 : {:.3f}".format(tree.score(X_test, y_test)))
```

```
max_depth : 1
훈련 세트 정확도 : 0.932
테스트 세트 정확도 : 0.883
max_depth : 2
훈련 세트 정확도 : 0.972
테스트 세트 정확도 : 0.912
max_depth : 3
훈련 세트 정확도 : 0.982
테스트 세트 정확도 : 0.906
max_depth : 4
훈련 세트 정확도 : 0.985
테스트 세트 정확도 : 0.906
max_depth : 5
훈련 세트 정확도 : 0.992
테스트 세트 정확도 : 0.889
max_depth : 6
훈련 세트 정확도 : 0.997
테스트 세트 정확도 : 0.901
```

```
In [8]: tree = DecisionTreeClassifier(max_depth=2, random_state=0)
        tree.fit(X_train, y_train)
        print(f"max_depth : {i}")
        print("훈련 세트 정확도 : {:.3f}".format(tree.score(X_train, y_train)))
        print("테스트 세트 정확도 : {:.3f}".format(tree.score(X_test, y_test)))
```

```
max_depth : 6
훈련 세트 정확도 : 0.972
테스트 세트 정확도 : 0.912
```

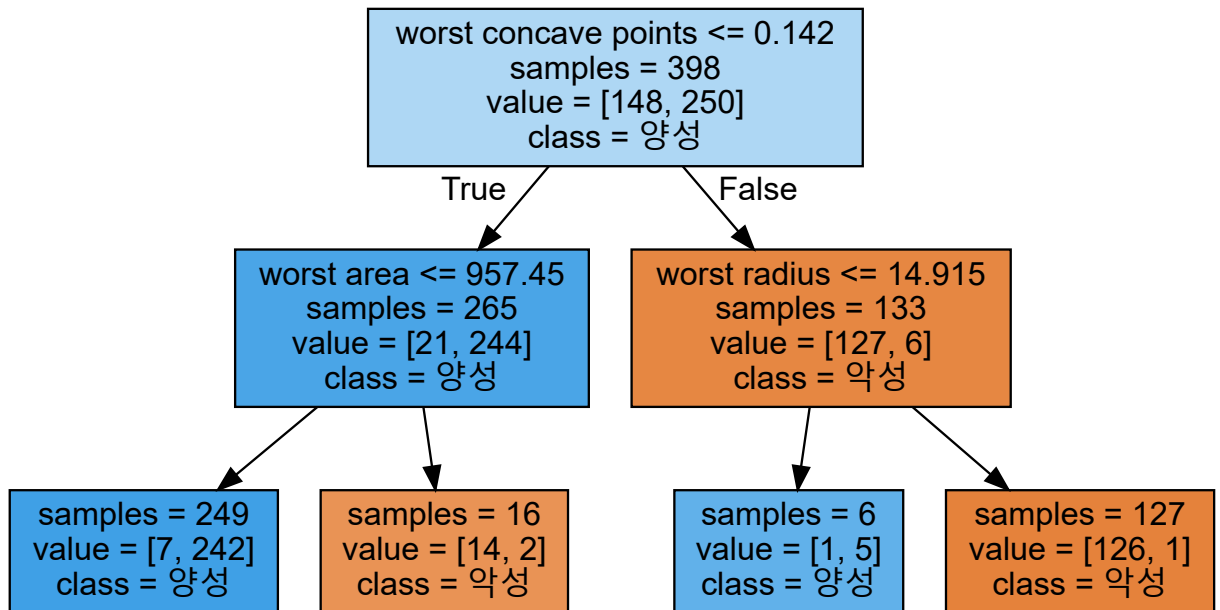
### 04 유방암 데이터 셋을 이용한 모델 구축 및 시각화

```
In [9]: from sklearn.tree import export_graphviz
        import graphviz
```

```
In [10]: export_graphviz(tree,
                        out_file="tree.dot",
                        class_names=['악성', '양성'],
```

```
feature_names = cancer.feature_names,
impurity = False, # gini 계수
filled=True)      # color
```

```
In [11]: with open("tree.dot") as f:
dot_graph = f.read()
display(graphviz.Source(dot_graph))
```



## 05 트리의 특성 중요도

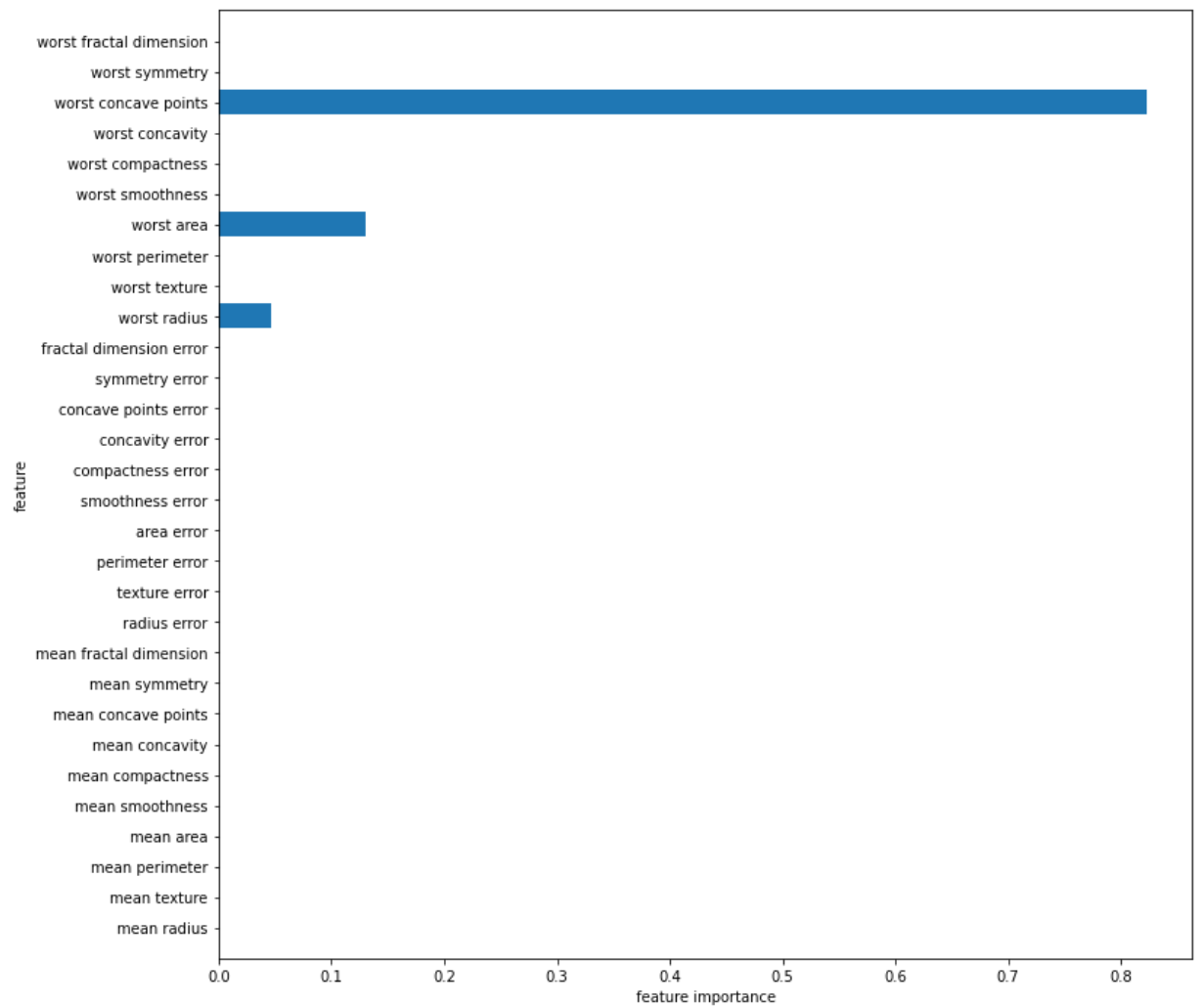
- 특성 중요도 : 이 값은 0과 1사이의 숫자.
  - 0은 트리에서 전혀 사용되지 않음.
  - 1은 트리에서 완벽하게 타깃 클래스를 예측했다.
  - 특성 중요도의 전체 합은 1이다.
- 특성의 *featureimportance* 값이 낮다고 해서 특성이 유용하지 않다는 것이 아니다.
- 단지 트리가 그 특성을 선택하지 않았다는 것.

```
In [12]: import numpy as np
```

```
In [13]: def plot_feature_imp_cancer(model):
n_features = cancer.data.shape[1]
imp = model.feature_importances_
plt.barh(range(n_features), imp, align='center')
plt.yticks(np.arange(n_features), cancer.feature_names)

plt.xlabel("feature importance")
plt.ylabel("feature")
plt.ylim(-1, n_features)
```

```
In [14]: plt.figure(figsize=(12,12))
plot_feature_imp_cancer(tree)
```



- worst\_concave\_points이 가장 중요한 특성으로 나타난다.
  - 첫번째 노드에서 두 클래스를 꽤 잘 나누고 있다.
- featureimportance 값이 낮다고 해서 특성이 유용하지 않다는 뜻이 아님.