ch02 앙상블 기법 (분류) - RandomForest(2)

- 유방암 데이터 셋 데이터 분석

학습 내용

- 1. RandomForest를 활용하여 유방암 데이터 분석을 수행해 본다.

목차

In [ ]:
```
import platform
import matplotlib
from matplotlib import font_manager, rc
import matplotlib
```

In [ ]:
```
# 한글 및 마이너스 표시 설정
path = "C:/Windows/Fonts/malgun.ttf"
if platform.system() == "Windows":
    font_name = font_manager.FontProperties(fname=path).get_name()
    matplotlib.rc('font', family=font_name)
elif platform.system()=="Darwin":
    rc('font', family='AppleGothic')
else:
    print("Unknown System")

matplotlib.rcParams['axes.unicode_minus'] = False
%matplotlib inline
```

01 앙상블을 활용한 유방암 유무 예측 모델 구축

In [ ]:
```
from sklearn.ensemble import RandomForestRegressor
from sklearn.model_selection import train_test_split
import matplotlib.pyplot as plt
import numpy as np
import pandas as pd
```

- 랜덤 포레스트는 **여러개의 모델 이용**이 가능하다.
    - n_estimators 를 이용
- 랜덤 포레스트는 각각의 **모델별 특징(변수) 선택을 제한**할 수 있다.

실습 1-1

- 데이터 셋 : 유방암 데이터 셋
- 랜덤 포레스트 알고리즘을 이용하여 모델을 만들어보자.
- (1) 모델의 학습용 세트 정확도, 테스트 세트 정확도를 확인해 보자.
    - 랜덤 포레스트 트리의 개수 = 5개, random_state=0, 최대 변수 선택 = 4

실습 1-1

In [ ]:
```
# 01 데이터 셋 불러오기
from sklearn.ensemble import RandomForestClassifier
from sklearn.datasets import load_breast_cancer
import mglearn
cancer = load_breast_cancer()

X = cancer.data
y = cancer.target

# 02 데이터 셋 나누기 및 학습
X_train, X_test, y_train, y_test = train_test_split(X,y ,
                                                    stratify=cancer.target, random_state=42)
```

In [ ]:
```
model = RandomForestClassifier(n_estimators=5, random_state=2)  # 5개의 트리
model.fit(X_train, y_train)

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

```
훈련 세트 정확도 : 1.000
테스트 세트 정확도 : 0.958
```

각각의 모델에 대한 정확도를 확인해 보자.

- 모델의 model.estimators_로 각각의 모델에 접근이 가능하다.

In [ ]:
```
model.estimators_
```

Out[ ]:
```
[DecisionTreeClassifier(max_features='auto', random_state=1872583848),
 DecisionTreeClassifier(max_features='auto', random_state=794921487),
 DecisionTreeClassifier(max_features='auto', random_state=111352301),
 DecisionTreeClassifier(max_features='auto', random_state=1853453896),
 DecisionTreeClassifier(max_features='auto', random_state=213298710)]
```

In [ ]:
```
# 5개의 모델에 대한 정확도 평가
for one_model in model.estimators_:
    print("학습용 세트 정확도 : {:.3f}".format(one_model.score(X_train, y_train)))
    print("테스트 세트 정확도 : {:.3f}".format(one_model.score(X_test, y_test)))
    print()
```

```
학습용 세트 정확도 : 0.986
테스트 세트 정확도 : 0.937

학습용 세트 정확도 : 0.981
테스트 세트 정확도 : 0.944

학습용 세트 정확도 : 0.962
테스트 세트 정확도 : 0.937

학습용 세트 정확도 : 0.986
테스트 세트 정확도 : 0.944

학습용 세트 정확도 : 0.965
테스트 세트 정확도 : 0.909
```

모델의 정보 확인

In [ ]:
```
print(model.feature_importances_)  # 모델의 중요도
print(model.n_features_)           # 모델 사용 특징
```

```
[0.01573926 0.01565104 0.00203568 0.10677515 0.00583498 0.00250098
 0.00279083 0.1518233  0.00170642 0.         0.00408493 0.00285406
 0.00165178 0.00553254 0.         0.01549629 0.00621975 0.
 0.00340706 0.00362224 0.32952352 0.04192876 0.02694543 0.03620429
 0.02041974 0.00820242 0.01165303 0.15871316 0.01608264 0.00260073]
30
```

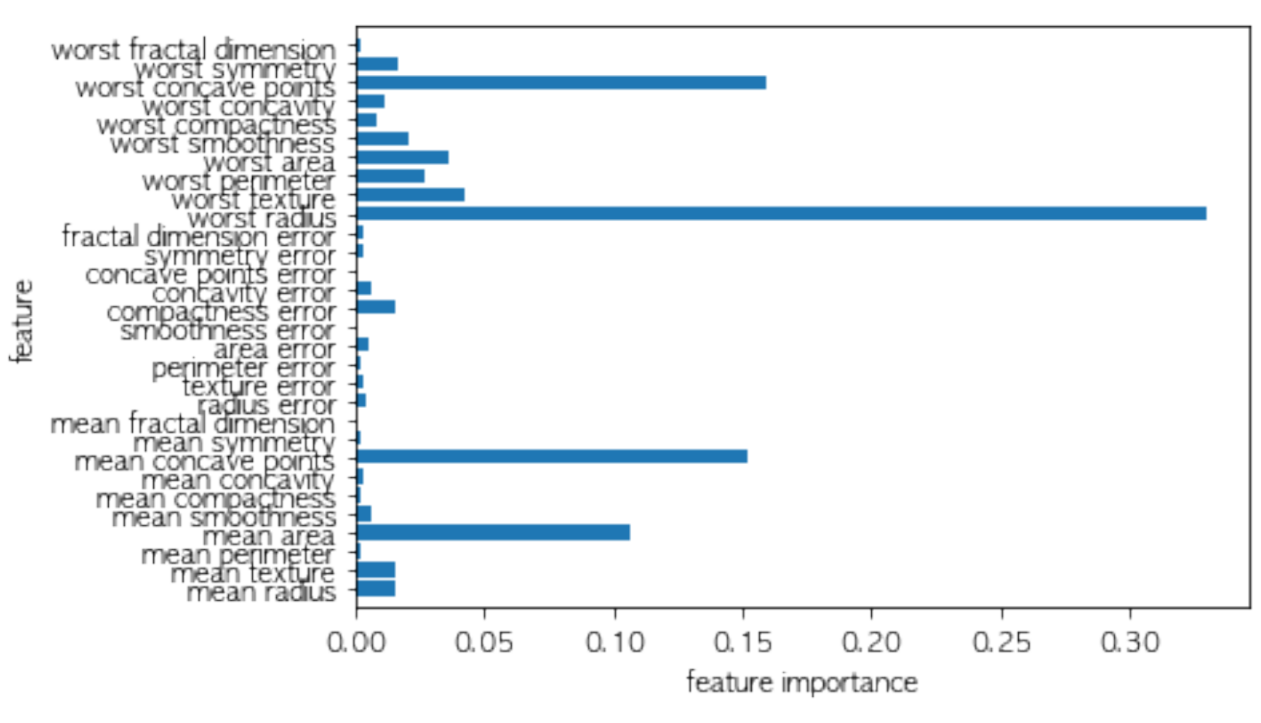
In [ ]:
```
# model : 모델
# 데이터 셋
def plot_feature_important_common(model, dataset, col_names):
  imp = model.feature_importances_                     # feature의 중요도
  n_features = dataset.shape[1]
  feature_names = col_names

  plt.barh(range(n_features) , imp, align='center')  # 그래프(가로 막대 그래프)

  plt.yticks(np.arange(n_features), feature_names)   # y축 값 지정

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

In [ ]:
```
n_fea = cancer.data.shape[1]
plot_feature_important_common(model, cancer.data, cancer.feature_names)
```



실습 1-2

- tree의 수를 100개로 해 보고, 모델 만들고, 정보확인해보기

02 모델 정보 시각화

- 입력 : 100개 2열
- 출력 : 100개 준비
    - 이 데이터 셋을 기준으로 모델을 만든다.

In [ ]:
```
from sklearn.ensemble import RandomForestClassifier
from sklearn.datasets import make_moons

# 100개 행, 2열
# make_moons() : 초승달 모양 클러스터 두 개의 형상의 데이터를 생성
# n_samples : 표본 데이터의 수, 기본 100개
# noise : 잡음의 크기. 0이면 정확한 반원을 이룬다.
X, y = make_moons(n_samples=100, noise=0.25, random_state=3)
print(X.shape, y.shape)

print(X[0:5])
print(y[0:5])
```

```
(100, 2) (100,)
[[ 1.87756309  0.56839425]
 [ 0.36877983 -0.34894509]
 [ 0.96515318  0.10921819]
 [ 0.48599685  0.20291313]
 [ 1.72532644  0.53367598]]
[1 1 0 1 1]
```

In [ ]:
```
X_train, X_test, y_train, y_test = train_test_split(X, y, stratify=y,
                                                    random_state=42)
m = RandomForestClassifier(n_estimators=5, random_state=2) # 5개의 트리
m.fit(X_train, y_train)
```

Out[ ]:
```
RandomForestClassifier(n_estimators=5, random_state=2)
```

코드 설명

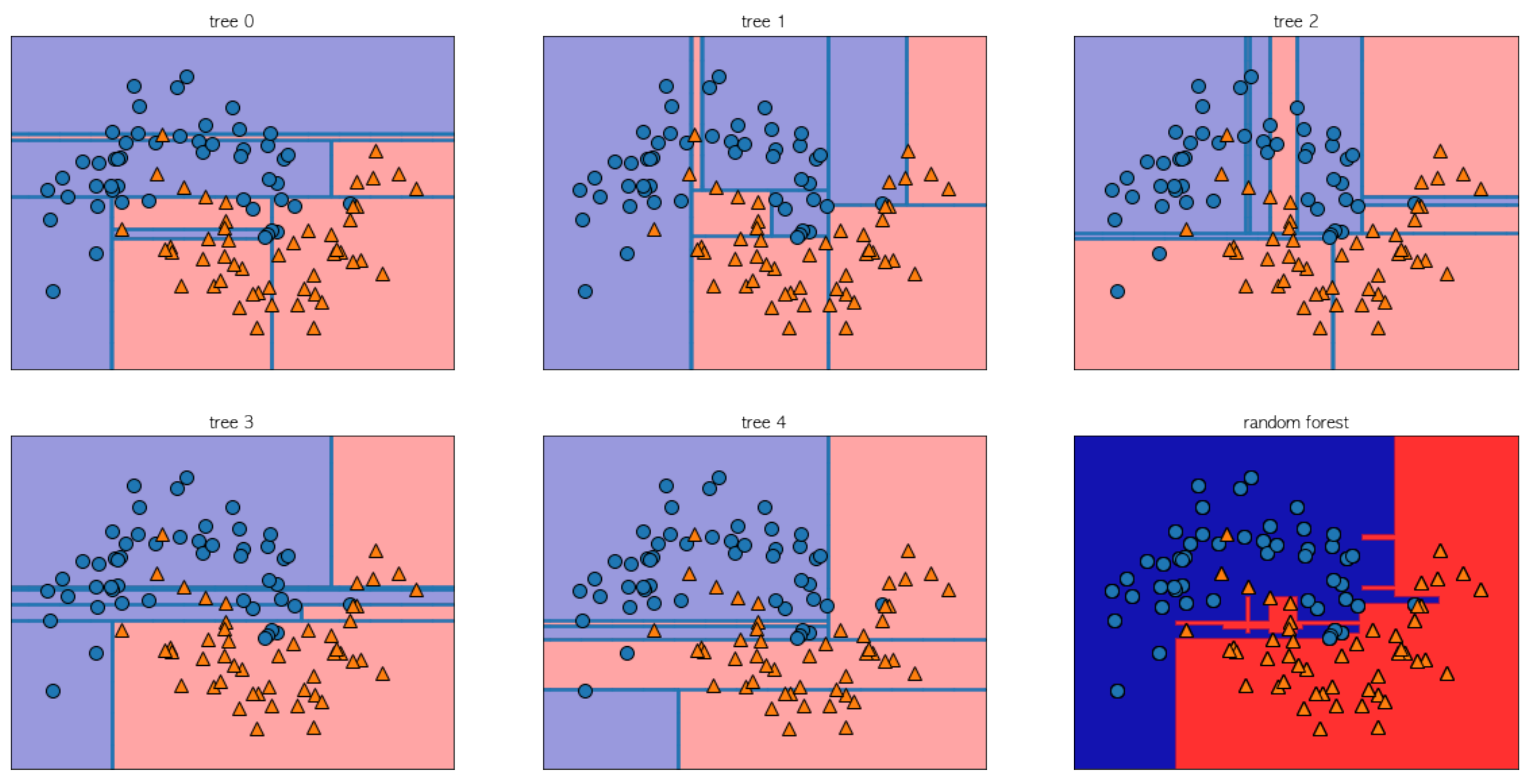
- 01 subplots로 2행 3열의 그래프를 기본 구조 지정
- 02 [].set_title : 각 해당 위치의 제목을 지정
- 03 tree을 보여주는 그래프를 그린다. 마지막은 random forest에 대한 그래프

In [ ]:
```
fig, axes = plt.subplots(2,3, figsize=(20,10))

for i, (ax, tree) in enumerate(zip(axes.ravel(), m.estimators_)):
    ax.set_title("tree {}".format(i))  # 각 그래프 제목

    # 그래프 그리기
    mglearn.plots.plot_tree_partition(X, y, tree, ax=ax)
    mglearn.plots.plot_2d_separator(m, X, fill=True, ax=axes[-1,-1], alpha=.4)
    axes[-1, -1].set_title("random forest")

    mglearn.discrete_scatter(X[:, 0],X[:,1], y)
```



추가 이해하기

- ravel() 함수 이해하기
    - ravel()함수를 이용하여 배열이 쫙 펴진다.

In [ ]:
```
import numpy as np
array = np.arange(15).reshape(3, 5)
print("원래 배열 : \n", array)
print("\n ravel() 함수 이용 : ", array.ravel())
```

```
원래 배열 :
 [[ 0  1  2  3  4]
 [ 5  6  7  8  9]
 [10 11 12 13 14]]

 ravel() 함수 이용 :  [ 0  1  2  3  4  5  6  7  8  9 10 11 12 13 14]
```

enumerate 이해

In [ ]:
```
for i, name in enumerate(['body', 'foo', 'bar']):
    print(i, name)
```

```
0 body
1 foo
2 bar
```

zip이해

In [ ]:
```
for i1, i2 in zip([11,12,13], [4,5,6]):
    print(i1, i2)
```

```
11 4
12 5
13 6
```