ch02 앙상블 기법- randomForest(2)

학습 내용

- 1. RandomForest를 활용하여 유방암 데이터 분석을 수행해 본다.

01. 랜덤 포레스트 분석

In [48]:

```
from sklearn.ensemble import RandomForestRegressor
from sklearn.datasets import make_moons
from sklearn.model_selection import train_test_split
import matplotlib.pyplot as plt
import numpy as np
import pandas as pd
```

실습해보기(1)

데이터 셋 : 유방암 데이터 셋
위의 데이터 셋을 이용하여 모델을 만들어보자.
(1) 랜덤 포레스트를 이용하여 훈련 세트 정확도, 테스트 세트 정확도를 확인해 보자.
(랜덤 포레스트 트리의 개수 = 5개, random_state=0, 최대 변수 선택 = 4)

실습 1 풀이

In [49]:

```
# 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=4
2)
forest = RandomForestClassifier(n_estimators=5, random_state=2)  # 5개의 트리
forest.fit(X_train, y_train)
```

Out[49]:

```
RandomForestClassifier(n_estimators=5, random_state=2)
```

In [50]:

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

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

In [51]:

```
# 5개의 모델에 대한 정확도 평가
for model in forest.estimators_:
    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
훈련 세트 정확도 : 1.000
테스트 세트 정확도 : 0.923
훈련 세트 정확도 : 1.000
테스트 세트 정확도 : 0.951
훈련 세트 정확도 : 1.000
테스트 세트 정확도 : 0.937
훈련 세트 정확도 : 1.000
테스트 세트 정확도 : 0.937
```

In [52]:

```
print(forest.feature_importances_)
print(forest.max_features)
print(forest.n_features_)
print(forest.oob_score)
print(forest)
```

```
[0.         0.01226443 0.00160554 0.12919727 0.01380144 0.01000016
 0.00240085 0.17205546 0.00389807 0.00304871 0.01171708 0.01206883
 0.         0.00712283 0.00188887 0.00302627 0.         0.00675665
 0.         0.0054606  0.28981717 0.03132555 0.15858617 0.01797894
 0.01314885 0.00636359 0.03506172 0.00768609 0.01757438 0.0261445 ]
auto
30
False
RandomForestClassifier(n_estimators=5, random_state=2)
```

In [53]:

```
forest1 = RandomForestClassifier(n_estimators=100, random_state=0)  # 100개의 트리
forest1.fit(X_train, y_train)
```

Out[53]:

```
RandomForestClassifier(random_state=0)
```

In [54]:

```
cancer.feature_names
```

Out[54]:

```
array(['mean radius', 'mean texture', 'mean perimeter', 'mean area',
       'mean smoothness', 'mean compactness', 'mean concavity',
       'mean concave points', 'mean symmetry', 'mean fractal dimension',
       'radius error', 'texture error', 'perimeter error', 'area error',
       'smoothness error', 'compactness error', 'concavity error',
       'concave points error', 'symmetry error',
       'fractal dimension error', 'worst radius', 'worst texture',
       'worst perimeter', 'worst area', 'worst smoothness',
       'worst compactness', 'worst concavity', 'worst concave points',
       'worst symmetry', 'worst fractal dimension'], dtype='<U23')
```

In [55]:

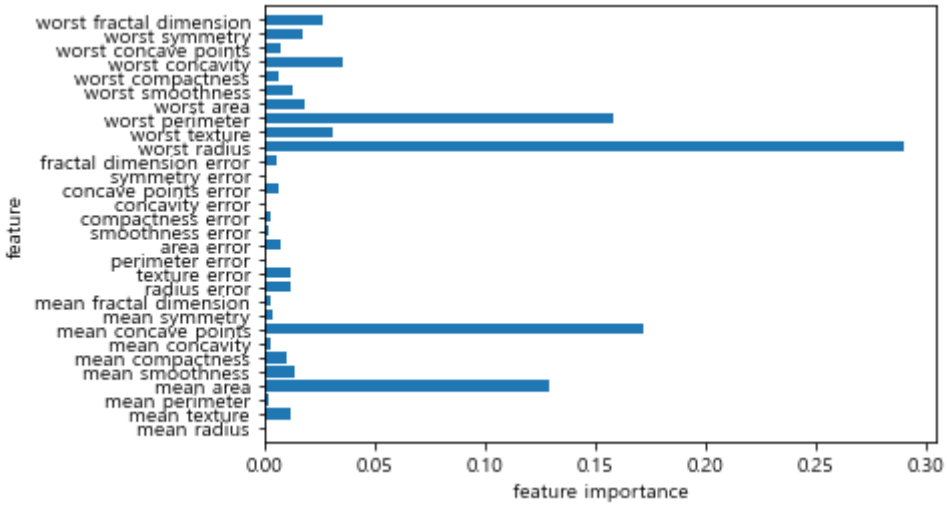
```
# 한글
import matplotlib
from matplotlib import font_manager, rc
font_loc = "C:/Windows/Fonts/malgunbd.ttf"
font_name = font_manager.FontProperties(fname=font_loc).get_name()
matplotlib.rc('font', family=font_name)
matplotlib.rcParams['axes.unicode_minus'] = False

%matplotlib inline
```

In [56]:

```
# model : 모델
# n_features : feature(변수의 개수)
# feature_names : 특성의 이름
def plot_feature_important_up(model, n_features, feature_names):
  imp = model.feature_importances_     # feature의 중요도
  plt.barh(range(n_features) , imp, align='center')  # 그래프(가로 막대 그래프)
  plt.yticks(np.arange(n_features), feature_names)  #y축의 축의 값
  plt.xlabel("feature importance")   # x축 레이블(제목)
  plt.ylabel("feature")              # y축 제목
  plt.ylim(-1, n_features)           # y축의 범위 지정
```

In [57]:

```
n_fea = cancer.data.shape[1]
plot_feature_important_up(forest, n_fea, cancer.feature_names)
```



In [58]:

```
print(forest1.feature_importances_)
print(forest1.max_features)
print(forest1.n_features_)
print(forest1.oob_score)
print(forest)
```

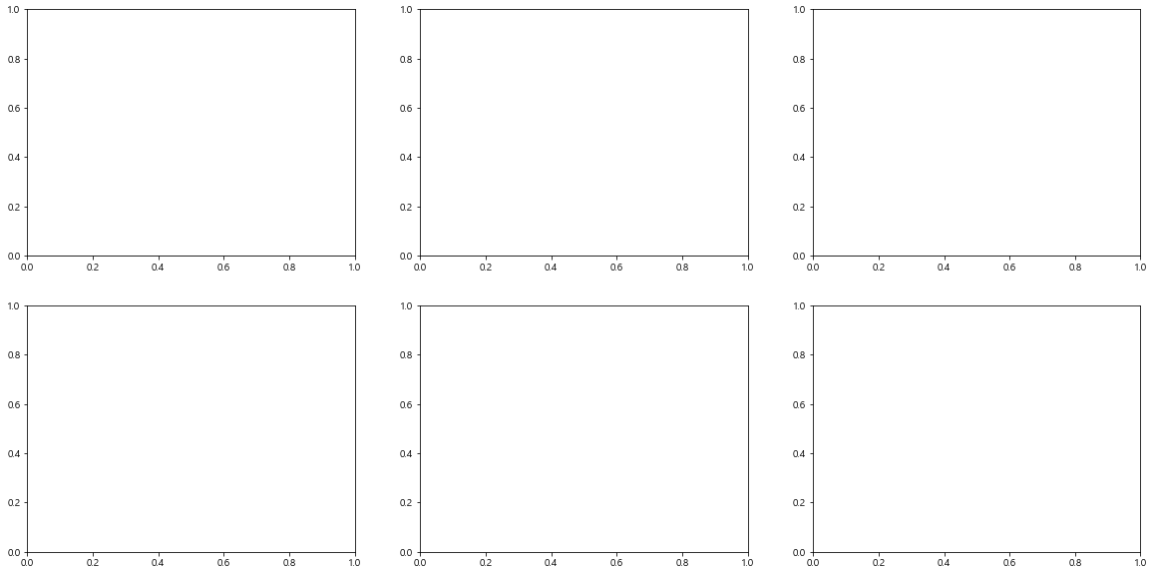
```
[0.03428109 0.01603486 0.07742074 0.04462701 0.00756603 0.00421043
 0.05773247 0.10465483 0.00561231 0.00310465 0.01828184 0.00550746
 0.01598658 0.02849289 0.00432991 0.00457692 0.00285285 0.00537969
 0.00521666 0.0038365  0.11200226 0.01973492 0.14912901 0.07116392
 0.00837159 0.01177922 0.02616743 0.13282975 0.00991387 0.00920232]
auto
30
False
RandomForestClassifier(n_estimators=5, random_state=2)
```

ravel() 함수 이해하기

In [59]:

```
fig1, axes1 = plt.subplots(2,3, figsize=(20,10)) # 2행, 3열의 사이즈 20, 10으로 그리기
# axes1.ravel()
import numpy as np
array = np.arange(15).reshape(3, 5)
print("Original array : ₩n", array)
print("₩nravel() : ", array.ravel())
```

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

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



enumerate 이해

In [60]:

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

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

zip이해

In [61]:

```
#print(zip(axes.ravel(), forest.estimators_))
for i1, i2 in zip([11,12,13], [4,5,6]):
    print(i1, i2)
```

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

- A simple toy dataset to visualize clustering and classification algorithms.
- [Parameters] n_samples : 발생시킬 데이터 수.
- [Parameters] noise : Standard deviation of Gaussian noise added to the data(가우시안 표준편차)

In [62]:

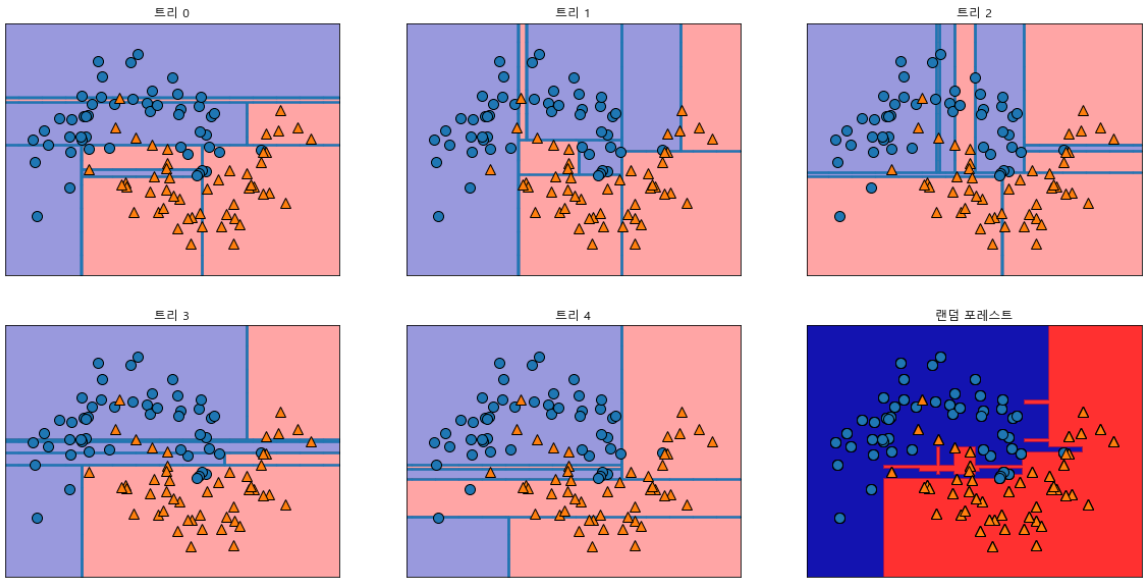
```
from sklearn.ensemble import RandomForestClassifier
from sklearn.datasets import make_moons

X, y = make_moons(n_samples=100, noise=0.25, random_state=3)

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

Out[62]:

```
RandomForestClassifier(n_estimators=5, random_state=2)
```

In [63]:

```
fig, axes = plt.subplots(2,3, figsize=(20,10)) # 2행, 3열의 사이즈 20, 10으로 그리기
# 각각의 트리에 각각의 트리 객체 대입
# 1단계 : zip에 의해 그래프객체1~6(ax), 모델객체1~6(tree)
# 2단계 : enumerate에 의해 인덱스 부여
for i, (ax, tree) in enumerate(zip(axes.ravel(), forest.estimators_)):
    ax.set_title("트리 {}".format(i))
    mglearn.plots.plot_tree_partition(X, y, tree, ax=ax)
    mglearn.plots.plot_2d_separator(forest, X, fill=True, ax=axes[-1,-1], alpha=.4)
    axes[-1, -1].set_title("랜덤 포레스트")
    mglearn.discrete_scatter(X[:, 0],X[:,1], y)
```



History

- 2020/12 업데이트 v11
- Machine Learning with sklearn @ DJ,Lim