프로젝트 기반 데이터 과학자 양성과정(Data Science)
Machine Learning 및 분석실습

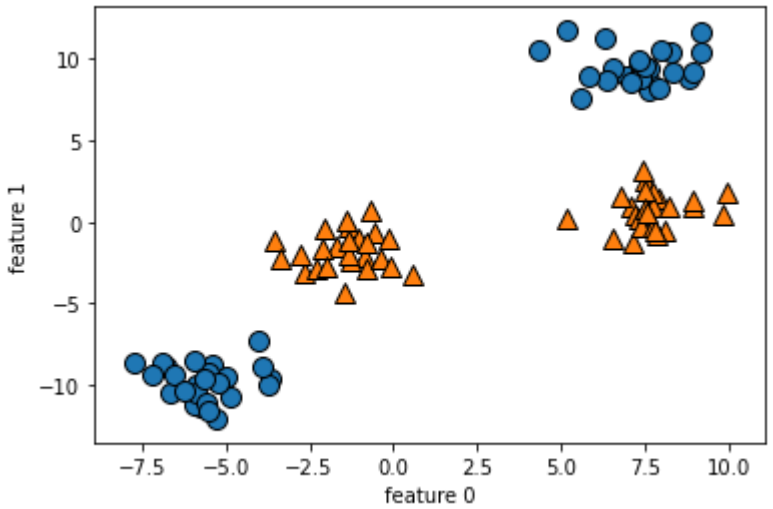
5주차
지도 학습
의사결정나무(Decision Tree)
RandomForest

강사 : 최영진

0. Support Vector Machine

❖ SVM 실습

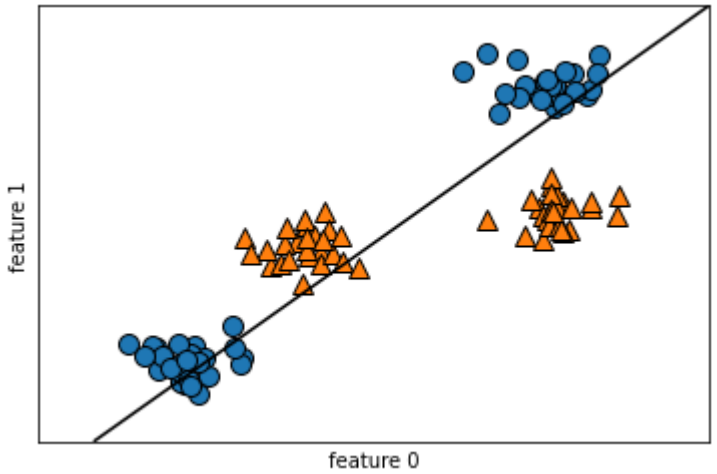
```
from sklearn.datasets import make_blobs
import mglearn
import matplotlib.pyplot as plt
import numpy as np

X, y = make_blobs(centers=4, random_state=8)
y = y % 2
print(y)

mglearn.discrete_scatter(X[:, 0], X[:, 1], y)
plt.xlabel("feature 0")
plt.ylabel("feature 1")
```



0. Support Vector Machine

❖ SVM 실습

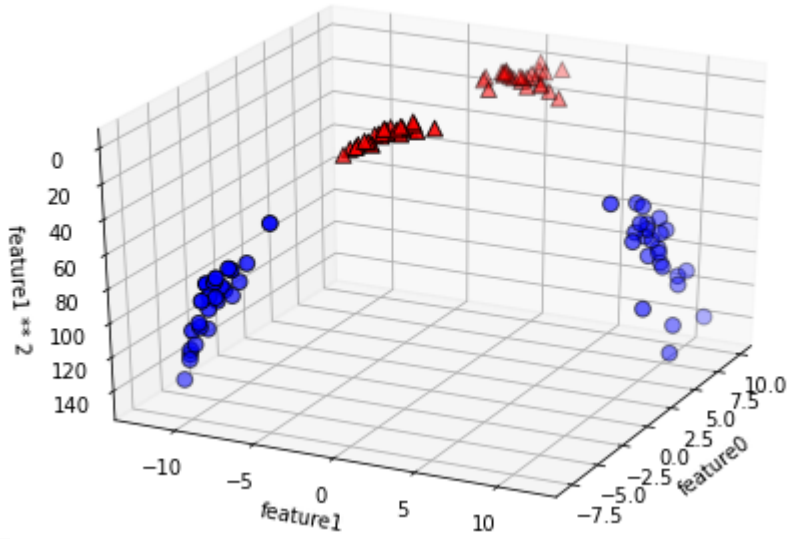
```
from sklearn.svm import LinearSVC
linear_svm = LinearSVC().fit(X, y)

mglearn.plots.plot_2d_separator(linear_svm, X)
mglearn.discrete_scatter(X[:, 0], X[:, 1], y)
plt.xlabel("feature 0")
plt.ylabel("feature 1")
```



0. Support Vector Machine

❖ SVM 실습

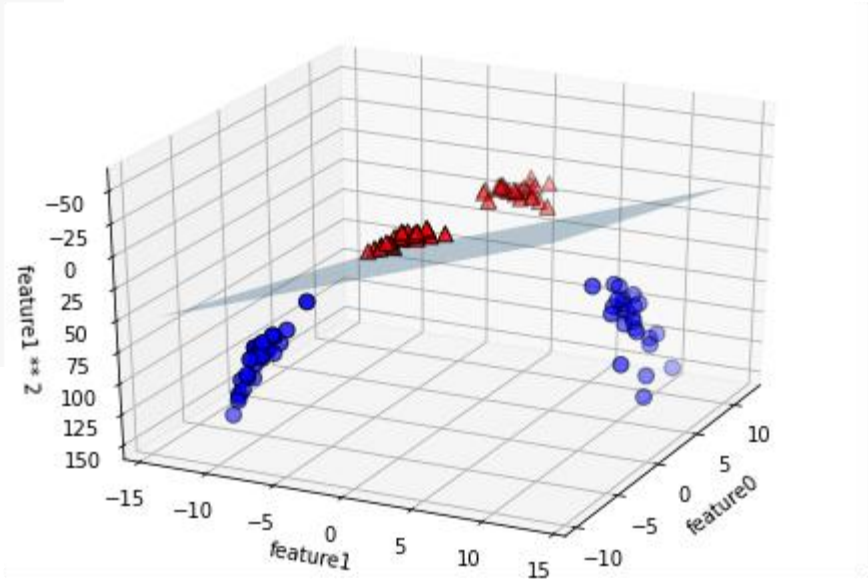
```
#3차원
X_new = np.hstack([X, X[:, 1:] ** 2])
print(X_new)


# 3차원 공간 그래프 그리기
from mpl_toolkits.mplot3d import Axes3D, axes3d
figure = plt.figure()

ax = Axes3D(figure, elev=-152, azim=-26)

# class 0, class 1 인것 구분하여 순서대로 그림

mask = y == 0
ax.scatter(X_new[mask, 0], X_new[mask, 1], X_new[mask, 2], c='b',
           cmap=mglearn.cm2, s=60, edgecolor='k')
ax.scatter(X_new[~mask, 0], X_new[~mask, 1], X_new[~mask, 2], c='r', marker='^',
           cmap=mglearn.cm2, s=60, edgecolor='k')
ax.set_xlabel("feature0")
ax.set_ylabel("feature1")
ax.set_zlabel("feature1 ** 2")
plt.show()
```



0. Support Vector Machine

❖ SVM 실습

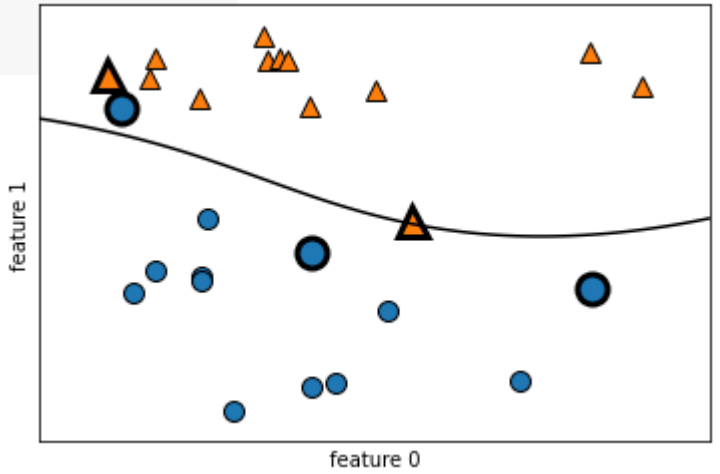
```
linear_svm_3d = LinearSVC(max_iter=5000).fit(X_new, y)
coef, intercept = linear_svm_3d.coef_.ravel(), linear_svm_3d.intercept_

# 선형 결정 경계 그리기
figure = plt.figure()
ax = Axes3D(figure, elev=-152, azim=-26)
xx = np.linspace(X_new[:, 0].min() - 2, X_new[:, 0].max() + 2, 50)
yy = np.linspace(X_new[:, 1].min() - 2, X_new[:, 1].max() + 2, 50)

XX, YY = np.meshgrid(xx, yy)
ZZ = (coef[0] * XX + coef[1] * YY + intercept) / -coef[2]
ax.plot_surface(XX, YY, ZZ, rstride=8, cstride=8, alpha=0.3)
ax.scatter(X_new[mask, 0], X_new[mask, 1], X_new[mask, 2], c='b',
           cmap=mglearn.cm2, s=60, edgecolor='k')
ax.scatter(X_new[~mask, 0], X_new[~mask, 1], X_new[~mask, 2], c='r', marker='^',
           cmap=mglearn.cm2, s=60, edgecolor='k')

ax.set_xlabel("feature0")
ax.set_ylabel("feature1")
ax.set_zlabel("feature1 ** 2")
```



0. Support Vector Machine

❖ SVM 실습

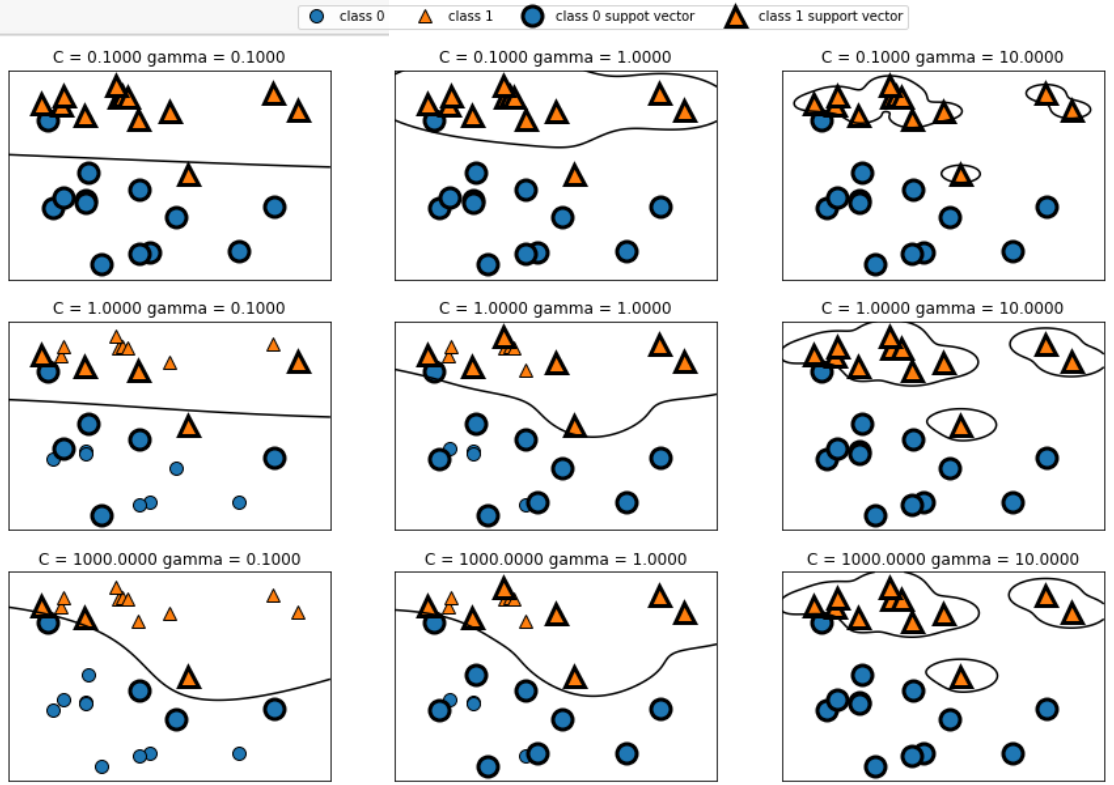
```
from sklearn.svm import SVC

X, y = mglearn.tools.make_handcrafted_dataset()
svm = SVC(kernel='rbf', C=10, gamma=0.1).fit(X, y)
mglearn.plots.plot_2d_separator(svm, X, eps=.5)
# 데이터 포인트 그리기
mglearn.discrete_scatter(X[:, 0], X[:, 1], y)
# 서포트 벡터
sv = svm.support_vectors_
# dual_coef_ 의 부호에 의해 서포트 벡터의 클래스 레이블이 결정됩니다
sv_labels = svm.dual_coef_.ravel() > 0
mglearn.discrete_scatter(sv[:, 0], sv[:, 1], sv_labels, s=15, markeredgewidth=3)
plt.xlabel("feature 0")
plt.ylabel("feature 1")
```



0. Support Vector Machine

❖ SVM 실습

```
fig, axes = plt.subplots(3, 3, figsize=(15, 10))

for ax, C in zip(axes, [-1, 0, 3]):
    for a, gamma in zip(ax, range(-1, 2)):
        mglearn.plots.plot_svm(log_C=C, log_gamma=gamma, ax=a)

axes[0, 0].legend(["class 0", "class 1", "class 0 suppot vector", "class 1 support vector"],
                  ncol=4, loc=(.9, 1.2))
```



zip 함수 예시

```
Number = [1,2,3,4]
Name = ['hong','gil','dong','nim']
Number_Name = list(zip(Number,Name))
print(Number_Name)
```

```
[(1, 'hong'), (2, 'gil'), (3, 'dong'), (4, 'nim')]
```

```
Number = [1,2,3,4]
Name = ['hong','gil','dong','nim']
dic = {}
for i in range(len(Number)) :
    dic[Number[i]] = Name[i]
print(dic)
```

```
{1: 'hong', 2: 'gil', 3: 'dong', 4: 'nim'}
```

0. Support Vector Machine

❖ SVM 실습

```
from sklearn.datasets import load_breast_cancer
breast_cancer_data = load_breast_cancer()


import pandas as pd
X_Data = pd.DataFrame(breast_cancer_data.data)
y = pd.DataFrame(breast_cancer_data.target)
```

```
# X_Data.info()
X_Data.describe()
```

	0	1	2	3	4	5	6	7	8	9	.
count	569.000000	569.000000	569.000000	569.000000	569.000000	569.000000	569.000000	569.000000	569.000000	569.000000	.
mean	14.127292	19.289649	91.969033	654.889104	0.096360	0.104341	0.088799	0.048919	0.181162	0.062798	.
std	3.524049	4.301036	24.298981	351.914129	0.014064	0.052813	0.079720	0.038803	0.027414	0.007060	.
min	6.981000	9.710000	43.790000	143.500000	0.052630	0.019380	0.000000	0.000000	0.106000	0.049960	.
25%	11.700000	16.170000	75.170000	420.300000	0.086370	0.064920	0.029560	0.020310	0.161900	0.057700	.
50%	13.370000	18.840000	86.240000	551.100000	0.095870	0.092630	0.061540	0.033500	0.179200	0.061540	.
75%	15.780000	21.800000	104.100000	782.700000	0.105300	0.130400	0.130700	0.074000	0.195700	0.066120	.
max	28.110000	39.280000	188.500000	2501.000000	0.163400	0.345400	0.426800	0.201200	0.304000	0.097440	.

0. Support Vector Machine

❖ SVM 실습

```
1  import sklearn.svm as svm
2  import sklearn.metrics as mt
3  from sklearn.model_selection import cross_val_score, cross_validate
4
5  # SVM, kernel = 'linear'로 선형분리 진행
6
7  svm_clf =svm.SVC(kernel = 'linear')
8
9  # 교차검증
10 scores = cross_val_score(svm_clf, X_Data, y, cv = 5)
11 scores
12
13 print('교차검증 평균: ', scores.mean())
14 print(pd.DataFrame(cross_validate(svm_clf, X_Data, y, cv =5)))
```

```
교차검증 평균:  0.9455364073901569
```

	fit_time	score_time	test_score
0	0.795871	0.000999	0.947368
1	1.850051	0.000999	0.929825
2	1.197795	0.000999	0.973684
3	0.604382	0.001996	0.921053
4	1.052186	0.000998	0.955752

0. Support Vector Machine

❖ SVM 실습

```
# SVM, kernel = 'rbf'로 비선형분리 진행

svm_clf =svm.SVC(kernel = 'rbf')

# 교차검증

scores = cross_val_score(svm_clf, X_Data, y, cv = 5)
scores

print(pd.DataFrame(cross_validate(svm_clf, X_Data, y, cv =5)))

print('cross Mean: ', scores.mean())

```

```
   fit_time  score_time  test_score
0  0.005983    0.001975    0.850877
1  0.005964    0.001024    0.894737
2  0.005984    0.001967    0.929825
3  0.005984    0.002016    0.947368
4  0.005963    0.002017    0.938053
cross Mean:  0.9121720229777983
```

0. Support Vector Machine

❖ SVM 실습

```
1  from sklearn.preprocessing import StandardScaler
2
3  scaler = StandardScaler()
4  scaler.fit(X_Data)
5  X_scaled = scaler.transform(X_Data)
6
7  from sklearn.model_selection import train_test_split
8  X_train, X_test, y_train, y_test = train_test_split(X_scaled, y, test_size = 0.3, random_state = 42)
9
10 import sklearn.svm as svm
11 import sklearn.metrics
12 from sklearn.model_selection import cross_val_score, cross_validate
13
14 svm_clf =svm.SVC(kernel = 'linear')
15
16 scores = cross_val_score(svm_clf, X_scaled, y, cv = 5)
17 scores
18
19 print(pd.DataFrame(cross_validate(svm_clf, X_scaled, y, cv =5)))
20
21 print('교차검증 평균: ', scores.mean())
22
```

```
   fit_time  score_time  test_score
0  0.003988    0.000997    0.956140
1  0.003984    0.000997    0.982456
2  0.003997    0.001000    0.964912
3  0.003987    0.000000    0.964912
4  0.002994    0.000995    0.982301
교차검증 평균:  0.9701443875174661
```

0. Support Vector Machine

❖ SVM 실습

```
from sklearn.model_selection import GridSearchCV

# 테스트하고자 하는 파라미터 값들을 사전타입으로 정의

svm_clf = svm.SVC(kernel = 'linear',random_state=42)
parameters = {'C': [0.001, 0.01, 0.1, 1, 10, 25, 50, 100]}

grid_svm = GridSearchCV(svm_clf,
                        param_grid = parameters, cv = 5)

grid_svm.fit(X_train, y_train)
print(grid_svm.best_params_)          # 좋은 파라미터를 보여줌,
print(grid_svm.best_score_)


result = pd.DataFrame(grid_svm.cv_results_['params'])
result['mean_test_score'] = grid_svm.cv_results_['mean_test_score']
result.sort_values(by='mean_test_score', ascending=False)

```

	C	mean_test_score
3	1.000	0.974810
2	0.100	0.972310
4	10.000	0.964778
7	100.000	0.959810
5	25.000	0.959778
6	50.000	0.957310
1	0.010	0.952215
0	0.001	0.927057

```
model=grid_svm.best_estimator_  # 최적의 파라미터로 모델 생성
y_pred=model.predict(X_test)

from sklearn import metrics

print("Accuracy:", metrics.accuracy_score(y_test, y_pred))

```

```
Accuracy: 0.9766081871345029
```

0. Support Vector Machine

❖ SVM 실습

```python
from sklearn.model_selection import GridSearchCV

# 테스트하고자 하는 파라미터 값들을 사전타입으로 정의

svm_clf = svm.SVC(kernel = 'rbf',random_state=100)
parameters = {'C': [0.001, 0.01, 0.1, 1, 10, 25, 50, 100],
              'gamma':[0.001, 0.01, 0.1, 1, 10, 25, 50, 100]}

grid_svm = GridSearchCV(svm_clf,
                        param_grid = parameters, cv = 5)


grid_svm.fit(X_train, y_train)
print(grid_svm.best_params_)          # 좋은 파라미터를 보여줌,
print(grid_svm.best_score_)

model=grid_svm.best_estimator_  # 최적의 파라미터로 모델 생성
y_pred=model.predict(X_test)

from sklearn import metrics

print("Accuracy:", metrics.accuracy_score(y_test, y_pred))


result = pd.DataFrame(grid_svm.cv_results_['params'])
result['mean_test_score'] = grid_svm.cv_results_['mean_test_score']
result.sort_values(by='mean_test_score', ascending=False)
```

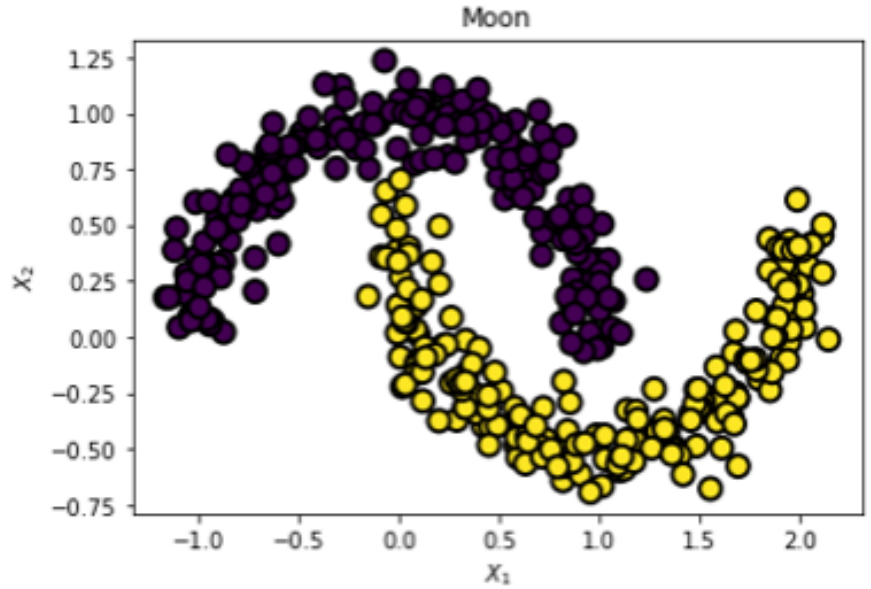
```
{'C': 50, 'gamma': 0.001}
0.9748417721518987
Accuracy: 0.9824561403508771
```

	C	gamma	mean_test_score
48	50.0	0.001	0.974842
40	25.0	0.001	0.972310
33	10.0	0.010	0.967278
57	100.0	0.010	0.967278
56	100.0	0.001	0.967247
...	...	...	...
28	1.0	10.000	0.625633
29	1.0	25.000	0.625633
30	1.0	50.000	0.625633
31	1.0	100.000	0.625633
63	100.0	100.000	0.625633

0. Support Vector Machine

❖ SVM 실습

```
1  from sklearn.datasets import make_moons
2  import matplotlib.pyplot as plt
3
4  plt.title("Moon")
5  X, y = make_moons(n_samples=400, noise=0.1, random_state=0)
6  plt.scatter(X[:, 0], X[:, 1], marker='o', c=y, s=100,
7              edgecolor="k", linewidth=2)
8  plt.xlabel("$X_1$")
9  plt.ylabel("$X_2$")
10 plt.show()
```



0. Support Vector Machine

❖ SVM 실습

```python
import sklearn.svm as svm
import sklearn.metrics as mt
from sklearn.model_selection import cross_val_score, cross_validate
from sklearn.model_selection import train_test_split, cross_val_score
import pandas as pd

X_train, X_test, y_train, y_test = train_test_split(X, y,
                                                    test_size = 0.3, random_state = 100)


svm_clf =svm.SVC(kernel = 'linear', random_state=100)

scores = cross_val_score(svm_clf, X, y, cv = 5)
scores



print(pd.DataFrame(cross_validate(svm_clf, X, y, cv =5)))
print('linear: ', scores.mean())


svm_clf =svm.SVC(kernel = 'rbf')

scores = cross_val_score(svm_clf, X, y, cv = 5)
scores

print(pd.DataFrame(cross_validate(svm_clf, X, y, cv =5)))
print('nonlinear: ', scores.mean())

```

```
   fit_time  score_time  test_score
0  0.000997    0.000998      0.8500
1  0.000998    0.000000      0.8875
2  0.000998    0.000000      0.8375
3  0.000997    0.000997      0.8625
4  0.000998    0.000000      0.9500
linear:  0.8775000000000001
   fit_time  score_time  test_score
0  0.000998    0.000000      1.0000
1  0.000999    0.000000      1.0000
2  0.000000    0.000998      1.0000
3  0.000997    0.000000      0.9750
4  0.000000    0.001000      0.9875
nonlinear:  0.9825
```

0. Support Vector Machine

❖ SVM 실습

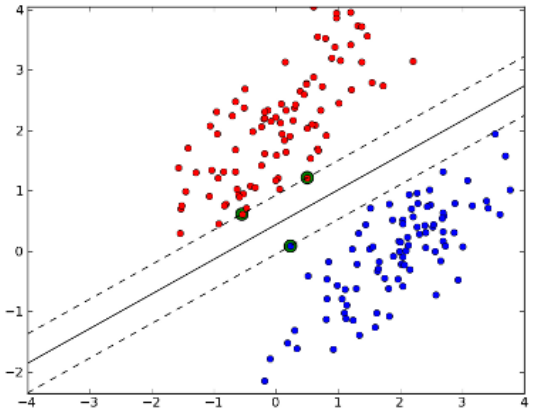
SVM

- *Objective Function*

  $\text{Min}\ \frac{1}{2} \| w \|^2 + c \sum_{i=1}^{N} \xi_i$

- *Constraints*

  $\text{s.t. } y_i(w^T x_i + b) \geq 1 - \xi_i, \xi_i \geq 0$



SVR

- *Estimate a linear regression*

  $f(X) = < W, X > + b$

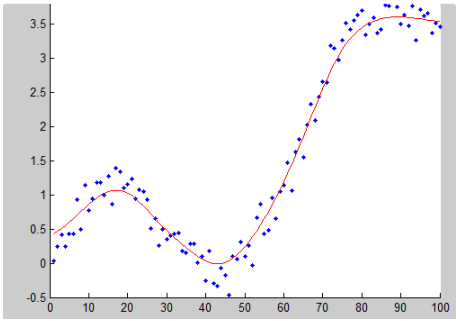
- *Objective Function*

  $\text{Min}\ \frac{1}{2} \| w \|^2 + c \sum_{i=1}^{N} (\xi_i + \xi_i^*)$

- *Constraints*

  $\text{s.t.} (< w, x_i > + b) - y_i \leq \varepsilon + \xi_i$

  $y_i - (< w, x_i > + b) \leq \varepsilon + \xi_i^*$



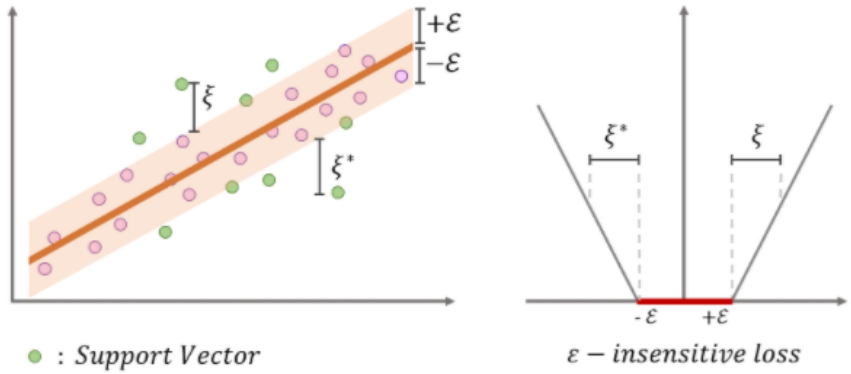
0. Support Vector Machine

❖ SVM 실습

구분	Loss Function	Objective Function
ε-insensitive	$c(\xi) = \lvert\xi\rvert_\varepsilon$ $\quad c(\xi) = \begin{cases} 0, & \lvert\xi\rvert < \varepsilon; \\ \lvert\xi\rvert - \varepsilon, & \lvert\xi\rvert \geq \varepsilon. \end{cases}$	$\min_{w,b} \frac{1}{2}\lVert w\rVert^2 + \frac{C}{l}\sum_{i=1}^{l}(\xi_i + \xi_i^*)$
Laplacian	$c(\xi) = \lvert\xi\rvert$	$\min_{w,b} \frac{1}{2}\lVert w\rVert^2 + C\sum_{i=1}^{l}(\xi_i + \xi_i^*)$
Gaussian	$c(\xi) = \frac{1}{2}\xi^2$	$\min_{w,b} \frac{1}{2}\lVert w\rVert^2 + C\sum_{i=1}^{l}(\xi_i^2 + \xi_i^{*2})$
Polynomial	$c(\xi) = \frac{1}{p}\lvert\xi\rvert^p$	$\min_{w,b} \frac{1}{2}\lVert w\rVert 2 + C\sum_{i=1}^{l}(\xi_i^p + \xi_i^{*p})$

- Structural Risk = Capacity Term + Training Error

  ☞ Loss Function으로 측정



0. Support Vector Machine

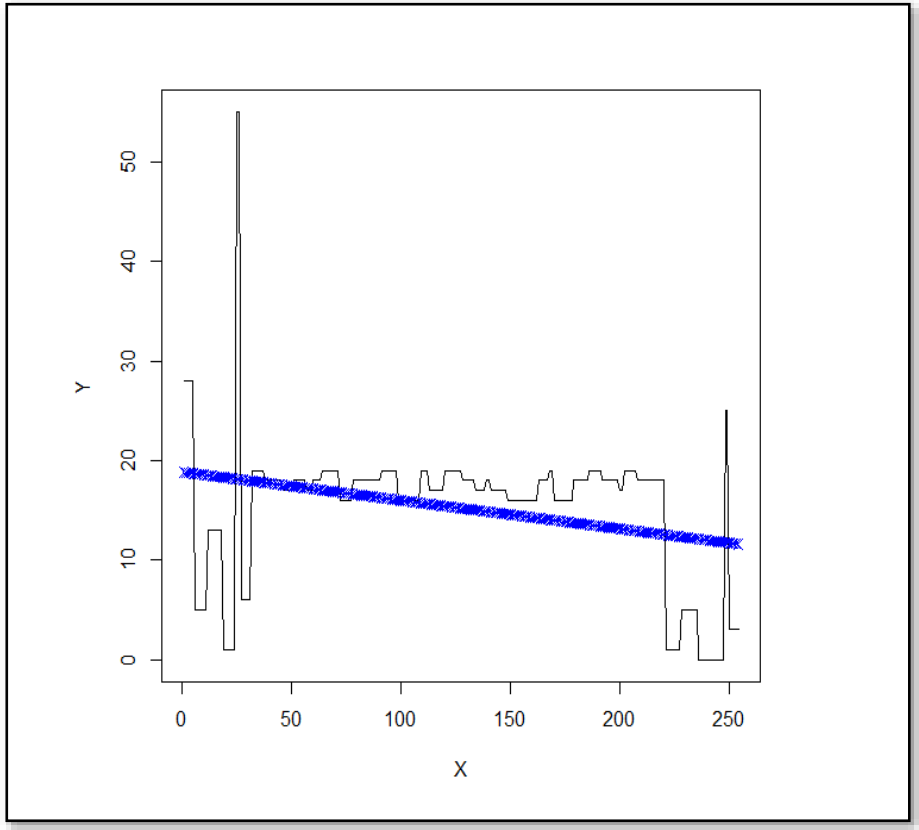
❖ SVM 실습

Kernel	Function	Ksvm(kpar='')
Gaussian Radial Basis Function	$k(\mathbf{x}, \mathbf{x}') = \exp(-\sigma\|\mathbf{x} - \mathbf{x}'\|^2)$	Sigma
Polynomial	$k(\mathbf{x}, \mathbf{x}') = \left(\text{scale} \cdot \langle \mathbf{x}, \mathbf{x}' \rangle + \text{offset}\right)^{\text{degree}}$	Scale, offset, degree
Hyperbolic tangent	$k(\mathbf{x}, \mathbf{x}') = \tanh\left(\text{scale} \cdot \langle \mathbf{x}, \mathbf{x}' \rangle + \text{offset}\right)$	Scale, offset
Laplace Radial Basis Function	$k(\mathbf{x}, \mathbf{x}') = \exp(-\sigma\|\mathbf{x} - \mathbf{x}'\|)$	Sigma
ANOVA radial basis	$k(\mathbf{x}, \mathbf{x}') = \left(\sum_{k=1}^{n} \exp(-\sigma(x^k - x'^k)^2)\right)^d$	Sigma, degree
Linear splines	$k(x, x') = 1 + xx' \min(x, x') - \frac{x + x'}{2}(\min(x, x')^2 + \frac{(\min(x, x')^3)}{3}$	
Bessel	$k(x, x') = \frac{J_{degree+1}(\sigma\|x - x'\|)}{\|x - x'\|^{-order(degree+1)}}$	Sigma, order, degree

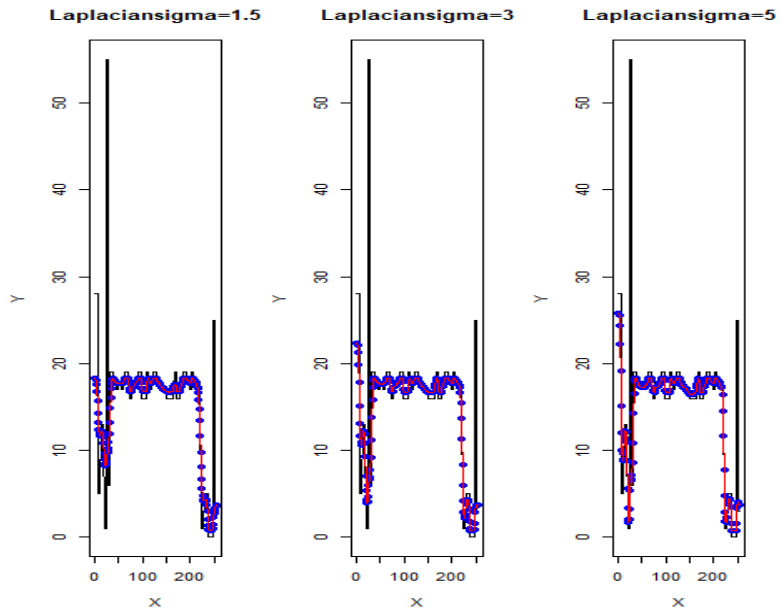
0. Support Vector Machine

❖ SVM 실습

Basic regression



Support Vector Regression



0. Support Vector Machine

❖ SVM 실습

- from sklearn.svm import SVR
- svr_rbf = SVR(kernel='rbf', C=100, gamma=0.1, epsilon=.1)
- svr_lin = SVR(kernel='linear', C=100, gamma='auto')
- svr_poly = SVR(kernel='poly', C=100, gamma='auto', degree=3, epsilon=.1, coef0=1)
- y_rbf = svr_rbf.fit(X, y).predict(X)
- y_lin = svr_lin.fit(X, y).predict(X)
- y_poly = svr_poly.fit(X, y).predict(X)

0. Support Vector Machine

❖ SVR 실습

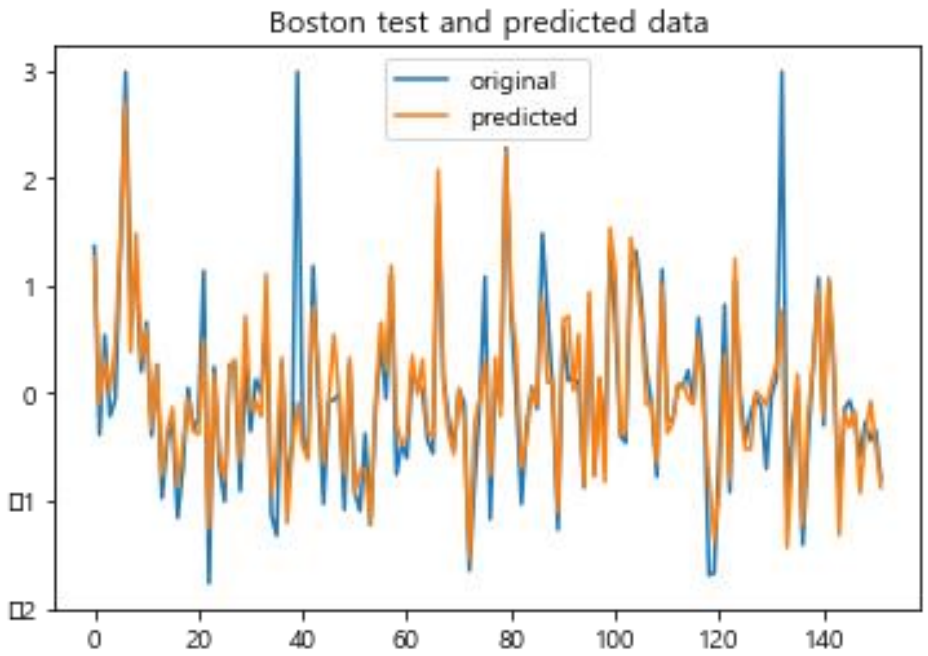
```
from sklearn.svm import LinearSVR
from sklearn.datasets import load_boston
from sklearn.datasets import make_regression
from sklearn.metrics import mean_squared_error
from sklearn.model_selection import train_test_split
from sklearn.model_selection import cross_val_score
from sklearn.preprocessing import scale
import matplotlib.pyplot as plt
boston = load_boston()
x, y = boston.data, boston.target

x = scale(x)
y = scale(y)
xtrain, xtest, ytrain, ytest = train_test_split(x, y, test_size=.3)
# lsvr=SVR(kernel='linear')
svr =SVR(kernel='rbf', gamma="auto")
svr.fit(xtrain, ytrain)
```

0. Support Vector Machine

❖ SVR 실습

```
19 score = svr.score(xtrain, ytrain)
20 print("R-squared:", score)
21
22 cv_score = cross_val_score(svr, x, y, cv=5)
23 print("CV mean score: ", cv_score.mean())
24
25 ypred = svr.predict(xtest)
26
27 mse = mean_squared_error(ytest, ypred)
28 print("MSE: ", mse)
29
30 x_ax = range(len(ytest))
31 plt.plot(x_ax, ytest, label="original")
32 plt.plot(x_ax, ypred, label="predicted")
33 plt.title("Boston test and predicted data")
34 plt.legend()
35 plt.show()
```

```
R-squared: 0.8955733007678928
CV mean score:  0.4911830321485834
MSE:  0.16994096509757747
```



0. Support Vector Machine

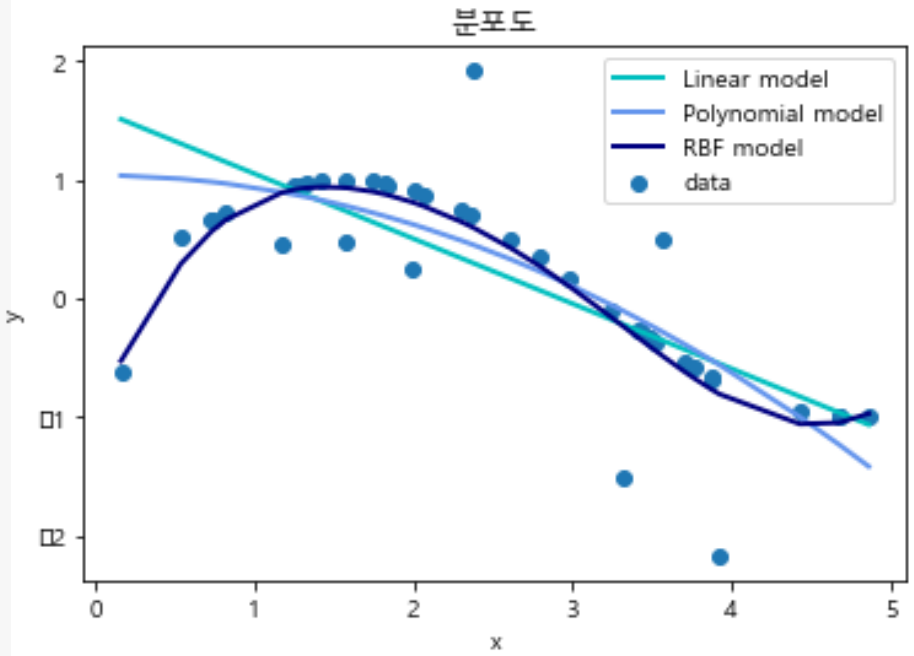
❖ SVR 실습

```
1  import numpy as np
2  from sklearn.svm import SVR
3  import matplotlib.pyplot as plt
4
5  x_data = np.sort(5 * np.random.rand(40, 1), axis=0)
6  y_data = np.sin(x_data).ravel()
7  y_data[::5] += 3 * (0.5 - np.random.rand(8)) # Add noise to targets
8
9  import matplotlib.pylab as plt
10 import matplotlib as mpl
11 plt.scatter(x_data, y_data, label='data')
12
13 clf_svr_linear = SVR(kernel='linear', C=1e3)
14 clf_svr_poly = SVR(kernel='poly', C=1e3, degree=2)
15 clf_svr_rbf = SVR(kernel='rbf', C=1e3, gamma=0.1)
16
17 clf_svr_linear.fit(x_data, y_data)
18 clf_svr_poly.fit(x_data, y_data)
19 clf_svr_rbf.fit(x_data, y_data)
20
```

0. Support Vector Machine

❖ SVR 실습

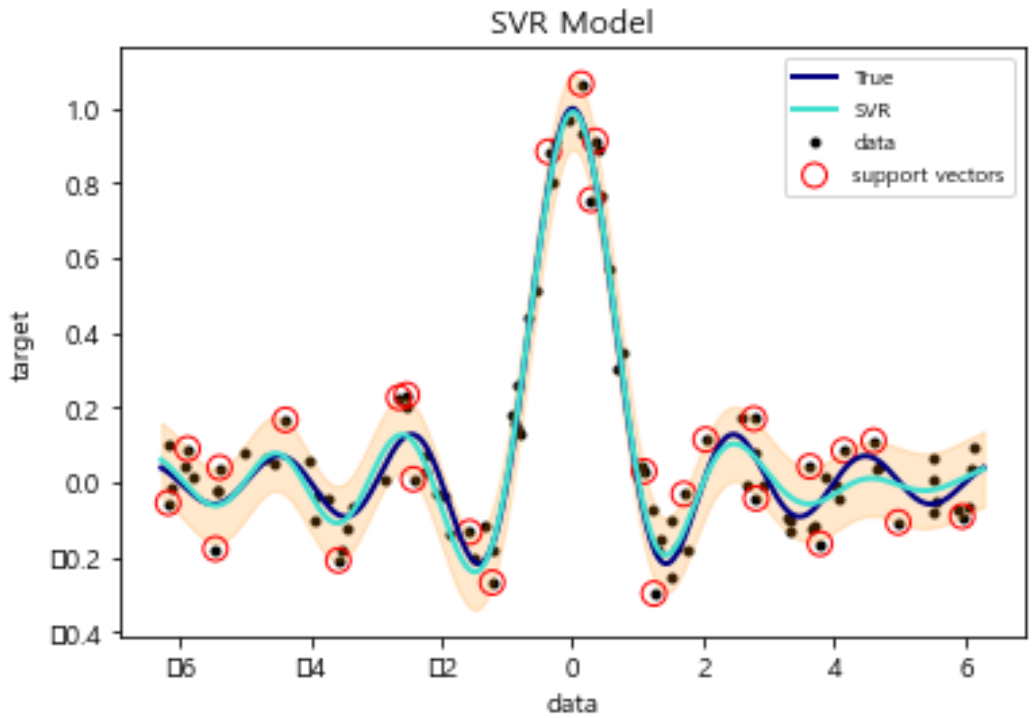
```
results_lin = clf_svr_linear.predict(x_data[:2])
results_poly = clf_svr_poly.predict(x_data[:2])
results_rbf = clf_svr_rbf.predict(x_data[:2])

results_lin = clf_svr_linear.predict(x_data)
results_poly = clf_svr_poly.predict(x_data)
results_rbf = clf_svr_rbf.predict(x_data)

import matplotlib.pylab as plt
import matplotlib as mpl
mpl.rc('font', family='Malgun Gothic') #한글 폰트 설정
lw = 2
plt.plot(x_data, results_lin, color='c', lw=lw, label='Linear model')
plt.plot(x_data, results_poly, color='cornflowerblue', lw=lw, label='Polynomial model')
plt.plot(x_data, results_rbf, color='navy', lw=lw, label='RBF model')
plt.title('분포도')
plt.xlabel('x')
plt.ylabel('y')
plt.legend() #범례
plt.show()
```



0. Support Vector Machine

❖ SVR 실습

```
import time
from sklearn.utils.testing import ignore_warnings
import matplotlib.pyplot as plt
import numpy as np
from sklearn.svm import SVR
# from sklearn_rvm import EMRVR

np.random.seed(8)
rng = np.random.RandomState(0)

# Generate sample data
X = 4 * np.pi * np.random.random(100) - 2 * np.pi
y = np.sinc(X)
y += 0.25 * (0.5 - rng.rand(X.shape[0]))  # add noise

X = X[:, None]

# Fit SVR
svr = SVR(kernel="rbf", gamma="auto",epsilon=0.1)
stime = time.time()
svr.fit(X, y)
print("Time for SVR fitting: %.3f" % (time.time() - stime))

X_plot = np.linspace(-2 * np.pi, 2 * np.pi, 10000)[:, None]
# Predict using SVR
stime = time.time()
y_svr = svr.predict(X_plot)
print("Time for SVR prediction: %.3f" % (time.time() - stime))

# Plot results
lw = 2

plt.scatter(X, y, marker=".", c="k", label="data")
plt.plot(X_plot, np.sinc(X_plot), color="navy", lw=lw, label="True")

plt.plot(X_plot, y_svr, color="turquoise", lw=lw, label="SVR")
plt.fill_between(X_plot[:, 0], y_svr - .1, y_svr + .1, color="darkorange", alpha=0.2)
support_vectors_idx = svr.support_
plt.scatter(X[support_vectors_idx], y[support_vectors_idx], s=80, facecolors="none", edgecolors="r",
            label="support vectors")
plt.ylabel("target")
plt.xlabel("data")
plt.legend(loc="best", scatterpoints=1, prop={"size": 8})
plt.title("SVR Model")
plt.show()
```

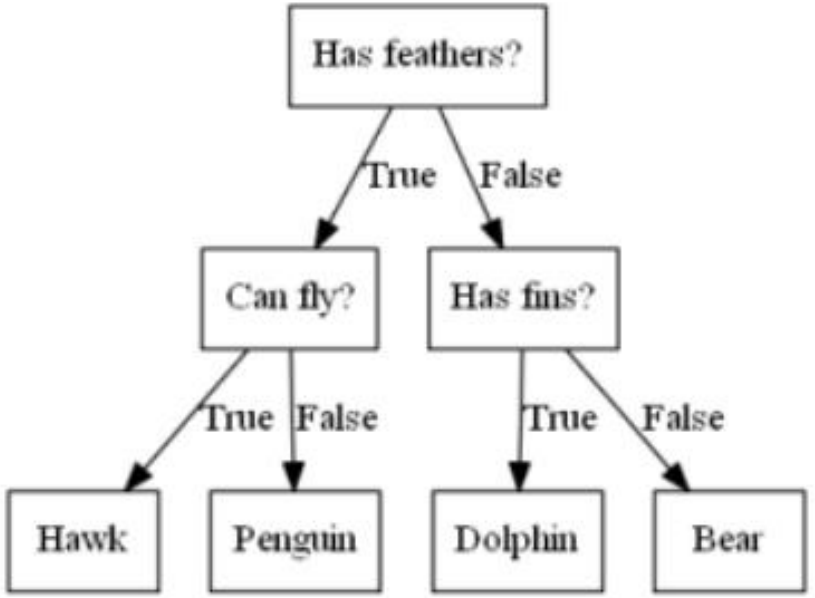


https://sklearn-rvm.readthedocs.io/en/latest/auto_examples/plot_compare_rvr_svr.html

1. 의사결정트리

❖ 의사결정트리(decision tree)

- 어떤 항목에 대한 관측값과 목표값을 연결시켜주는 예측 모델로서 결정 트리를 사용
- 의사결정 트리는 지도학습 방법으로 머신러닝에서 분류와 회귀에 사용
- 매우 직관적인 방법중 하나임
- 의사 결정에 이르기까지 yes/No로 분류하여 사용하고 질문을 던져 대상을 좁혀 스무고개 놀이와 비슷한 개념



1. 의사결정트리

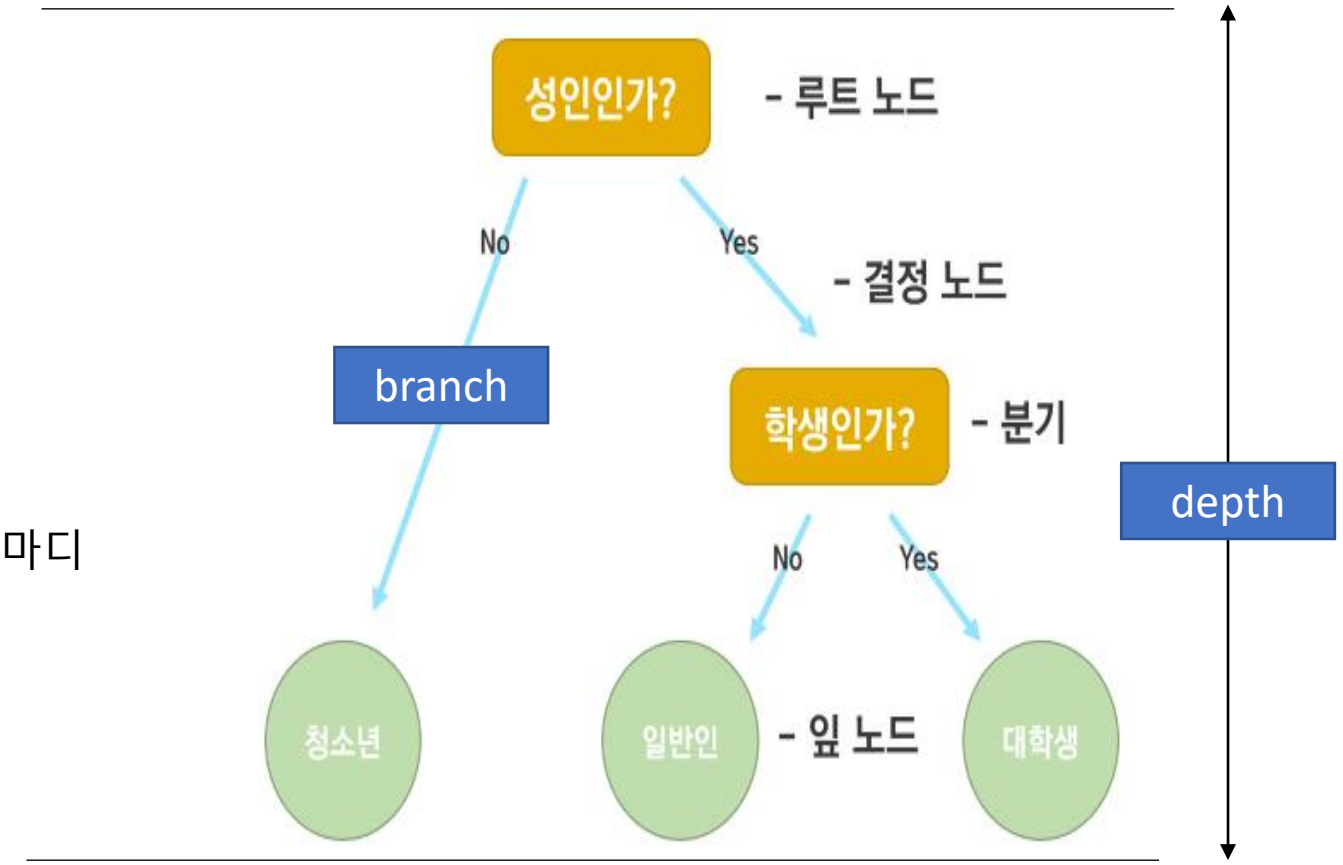
❖ 의사결정트리(decision tree)

- **classification Tree**
  - 종속 변수가 이산형인경우, 각각의 범주에 속하는 빈도에 기초해 분리발생
  - 분류 트리 분석은 예측된 결과로 입력 데이터가 분류되는 클래스를 출력
- **Regression Tree**
  - 종속 변수가 연속형인 경우, 평균과 표준편차에 의해 노드 분리
  - 회귀 트리 분석은 예측된 결과로 특정 의미를 지니는 실수 값을 출력
  - (예: 주택의 가격, 환자의 입원 기간)

1. 의사결정트리

❖ 의사결정트리(decision tree) 구성요소

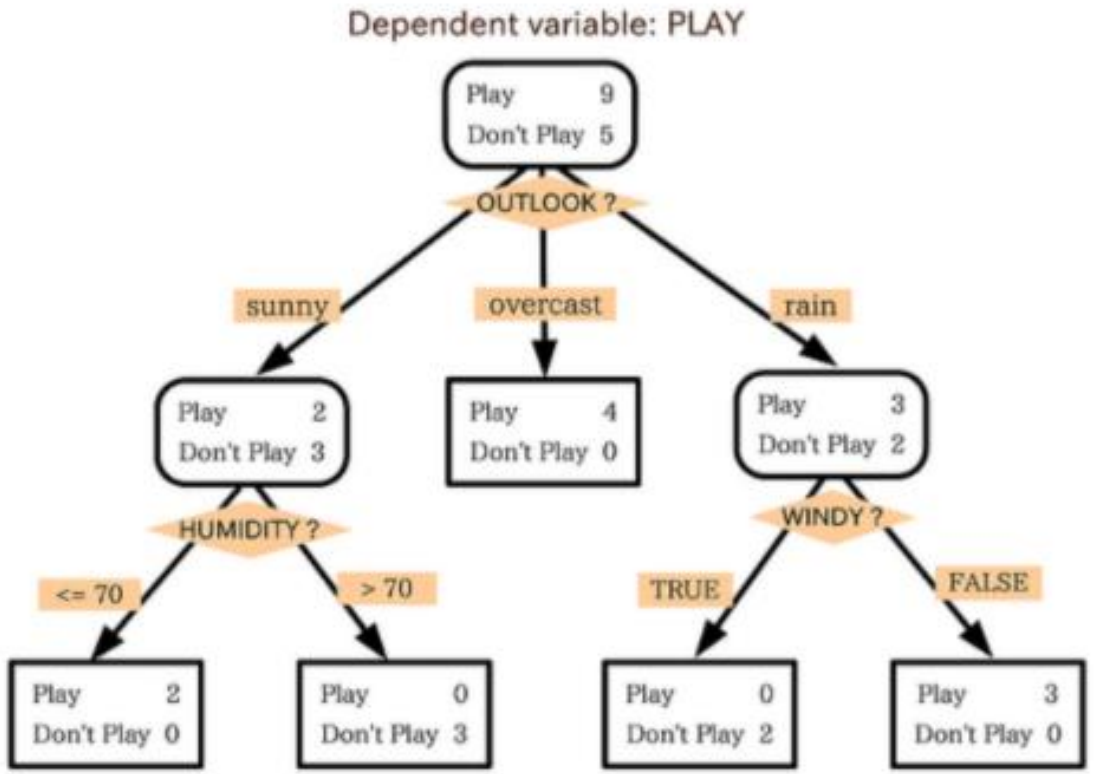
- 나무에서 분할되는 부분을 노드(node)
- 처음 노드 : root node
- 마지막 노드 : terminal node/ leaf node
  - 분리된 집합의 개수
- 부모 노드(parent node)
- 자식노드(child node)
- 가지(Branch) : 뿌리 마디로부터 끝마디까지 연결된 마디
- 깊이(Depth) : 뿌리마디로부터 끝마디를 이루는 층



1. 의사결정트리

❖ 의사결정트리(decision tree)

- 데이터에 내재되어 있는 패턴을 변수의 조합으로 나타내는 예측 분류 모델을 나무의 형태로 만드는 것

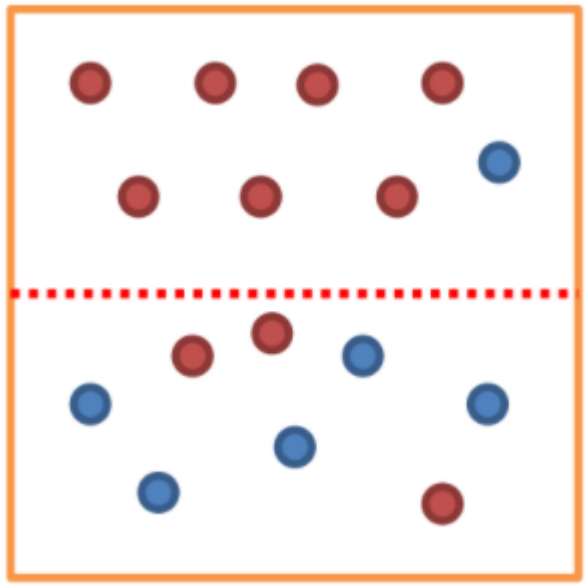


1. 의사결정트리

❖ 의사결정트리(decision tree) 구성요소

▪ 지니 불순도

- 집합에 이질적인 것이 얼마나 섞였는지를 측정하는 지표이며 CART 알고리즘에서 사용
- 집합에서 한 항목을 뽑아 무작위로 라벨을 추정할 때 틀릴 확률
- 집합에 있는 항목이 모두 같다면 지니 불순도는 최솟값(0)을 갖게 되며 이 집합은 완전히 순수



$$I(A) = 1 - \sum_{k=1}^{m} p_k^2$$

$$I(A) = 1 - \sum_{k=1}^{m} p_k^2$$

$$= 1 - \left(\frac{6}{16}\right)^2 - \left(\frac{10}{16}\right)^2$$

$$\approx 0.47$$

$$I(A) = \sum_{i=1}^{d} \left(R_i \left(1 - \sum_{k=1}^{m} p_{ik}^2\right)\right)$$

$$I(A)$$

$$= 0.5 \times \left(1 - \left(\frac{7}{8}\right)^2 - \left(\frac{1}{8}\right)^2\right) + 0.5 \times \left(1 - \left(\frac{3}{8}\right)^2 - \left(\frac{5}{8}\right)^2\right)$$

$$= 0.34$$

정보 획득량 (Information Gain) 은 0.47 - 0.34 = 0.13 이다

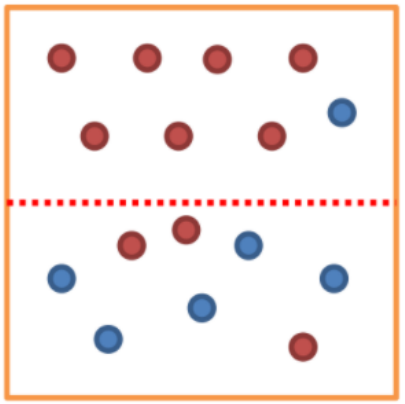
1. 의사결정트리

❖ 의사결정트리(decision tree) 구성요소

- **엔트로피(entropy)**
- m개의 레코드가 속하는 A영역에 대한 엔트로피(log)

$$Entropy(A) = -\sum_{k=1}^{m} p_k \log_2 (p_k)$$

- 16개(m=16) 가운데 빨간색 동그라미(범주=1)는 10개, 파란색(범주=2)은 6개



$$Entropy(A) = -\frac{10}{16}\log_2 (\frac{10}{16}) - \frac{6}{16}\log_2 (\frac{6}{16}) \approx 0.95$$

엔트로피 감소(=불확실성 감소=순도 증가=정보획득)

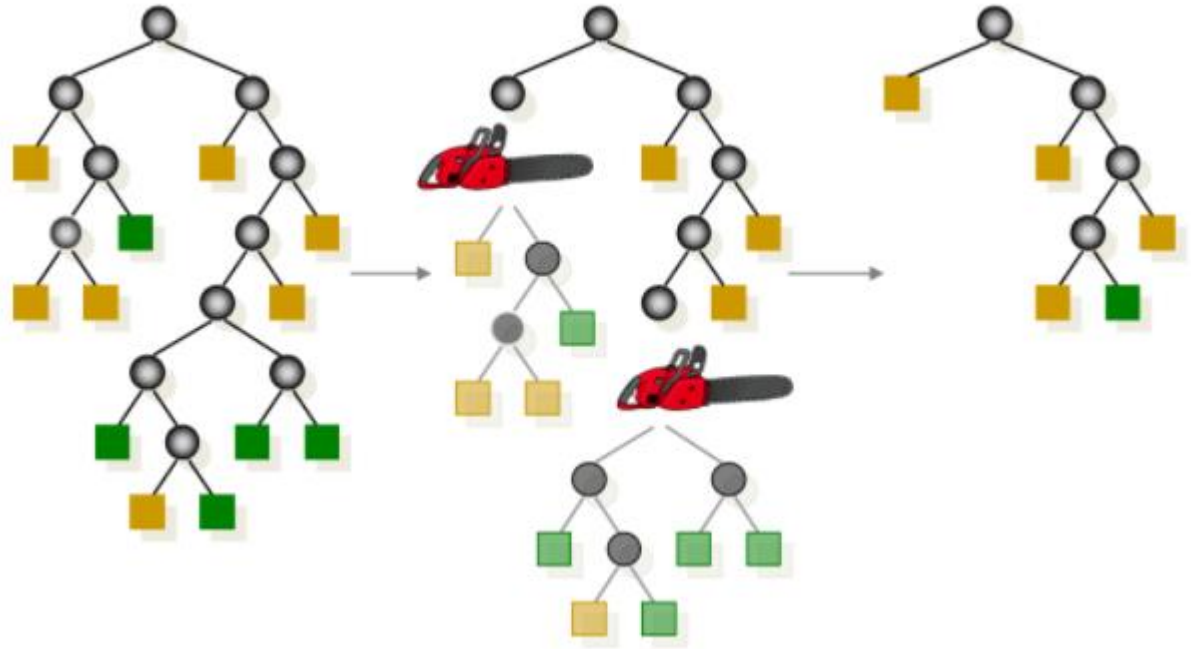
$$Entropy(A) = 0.5 \times \left(-\frac{7}{8}\log_2 (\frac{7}{8}) - \frac{1}{8}\log_2 (\frac{1}{8})\right) + 0.5 \times \left(-\frac{3}{8}\log_2 (\frac{3}{8}) - \frac{5}{8}\log_2 (\frac{5}{8})\right) \approx 0.75$$

각 영역의 순도(homogeneity)가 증가/불확실성(엔트로피)가 최대한 감소하도록 하는 방향으로 학습을 진행

1. 의사결정트리

❖ 의사결정트리(decision tree) 구성요소

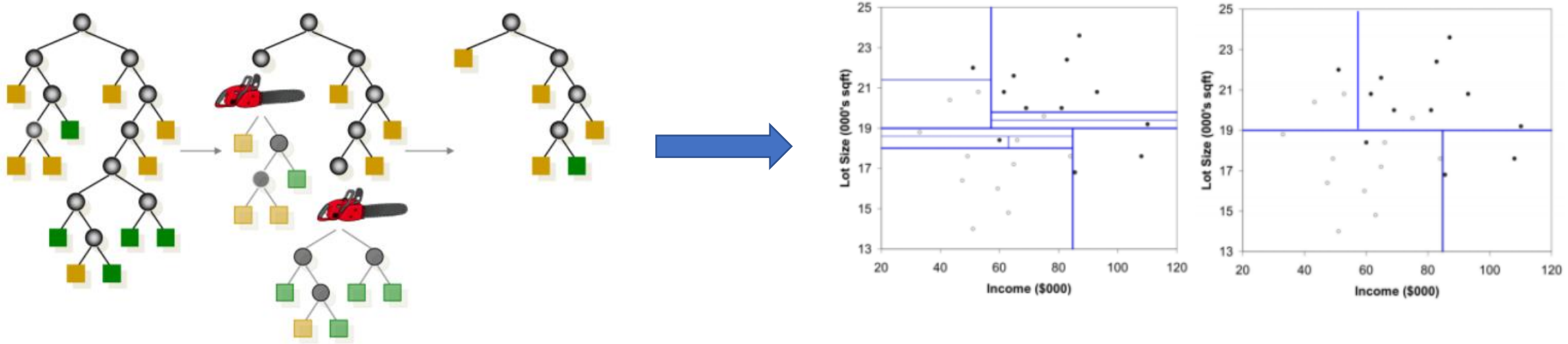
- **가지치기(Pruning) :** 오버피팅을 막기 위한 전략으로 불필요한 가지 제거
- Full tree를 생성한 뒤 적절한 수준에서 terminal node를 결합이 필요
- 분기 수가 증가할 때 처음에는 새로운 데이터에 대한 오분류율이 감소하나
  일정 수준 이상이 되면 오분류율이 되레 증가하는 현상



1. 의사결정트리

❖ 의사결정트리(decision tree) 구성요소

- **가지치기(Pruning) :** 오버피팅을 막기 위한 전략으로 불필요한 가지 제거
- Full tree를 생성한 뒤 적절한 수준에서 terminal node를 결합이 필요
- 분기 수가 증가할 때 처음에는 새로운 데이터에 대한 오분류율이 감소하나 일정 수준 이상이 되면 오분류율이 되레 증가하는 현상



1. 의사결정트리

❖ 의사결정트리(decision tree) – 재귀적 분기

- **특정 영역인 하나의 노드 내에서 하나의 변수 값을 기준으로 분기하여 새로 생성된 자식 노드들의 동질성이 최대화 되도록 분기점을 선택(동질성이 최대화 == 불순도는 최소화)**
  - 범주형 변수 : 지니 계수 (Gini Index)
  - 수치형 변수 : 분산 (Variance)

구매가	유지비	문의 수	탑승 가능인원	안전	평가
Vhigh	Vhigh	2	2	Low	Unacc
Vhigh	Vhigh	2	2	Med	Acc
Vhigh	High	2	2	High	Unacc
Vhigh	High	4	2	Low	Acc
High	High	4	2	Med	Unacc
High	Med	4	2	High	Acc
Med	Med	4	2	Low	Unacc
Med	Low	5	2	Med	Acc
Med	Low	5	2	High	Acc
Med	Low	5	4	Low	Acc

$$E(S) = -\frac{4}{10}\log_2\frac{4}{10} - \frac{6}{10}\log_2\frac{6}{10} \approx 0.971$$

$$E(S') = -\frac{1}{10}\log_2 1 - \frac{9}{10}\left(\frac{6}{9}\log_2\frac{6}{9} + \frac{3}{9}\log_2\frac{3}{9}\right) \approx 0.826$$

$$G(S) = E(S) - E(S') \approx 0.145$$

1. 의사결정트리

❖ 의사결정트리(decision tree)

- **장점**
  - 자료를 가공할 필요가 거의 없고 다른 기법들의 경우 자료를 정규화하거나 임의의 변수를 생성하거나 값이 없는 변수를 제거해야 하는 경우
  - 수치 자료와 범주 자료 모두에 적용
  - 다른 기법들은 일반적으로 오직 한 종류의 변수를 갖는 데이터 셋을 분석하는 것에 특화
  - 대규모의 데이터 셋에서도 잘 동작한다. 방대한 분량의 데이터를 일반적인 컴퓨터 환경에서 합리적인 시간 안에 분석

- **단점**
  - 결정 트리 학습자가 훈련 데이터를 제대로 일반화하지 못할 경우 너무 복잡한 결정 트리
  - 데이터의 특성이 특정 변수에 수직/수평적으로 구분되지 못할 때 분류율이 떨어지고, 트리가 복잡해지는 문제
  - 연속형 변수들의 대한 분리 경계점에서 예측오류가 클 가능성

1. 의사결정트리

❖ 의사결정트리(decision tree) 실습

- from sklearn.tree import DecisionTreeClassifier
- tree = DecisionTreeClassifier(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)))

1. 의사결정트리

❖ 의사결정트리(decision tree) 실습

- **max depth**: 최대 깊이 설정
- **min_samples_split**: 분할되기 위해 노드가 가져야하는 최소샘플 수
- **min_samples_leaf**: 리프 노드가 가지고있어야 하는 최소 샘플 수
- **min_weight_fraction_leaf**: min_samples_leaf와 비슷하지만 가중치가 부여된 전체 샘플 수에서의 비율
- **max_leaf_nodes**: 리프 노드의 최대수
- **max_features**: 각 노드에서 분할에 사용할 특성의 최대 수

1. 의사결정트리

❖ 의사결정트리(decision tree)

파라미터 명	설명
min_samples_split	- 노드를 분할하기 위한 최소한의 샘플 데이터수 → 과적합을 제어하는데 사용 - Default = 2 → 작게 설정할 수록 분할 노드가 많아져 과적합 가능성 증가
min_samples_leaf	- 리프노드가 되기 위해 필요한 최소한의 샘플 데이터수 - min_samples_split과 함께 과적합 제어 용도 - 불균형 데이터의 경우 특정 클래스의 데이터가 극도로 작을 수 있으므로 작게 설정 필요
max_features	- 최적의 분할을 위해 고려할 최대 feature 개수 - Default = None → 데이터 세트의 모든 피처를 사용 - int형으로 지정 →피처 갯수 / float형으로 지정 →비중 - sqrt 또는 auto : 전체 피처 중 √(피처개수) 만큼 선정 - log : 전체 피처 중 log2(전체 피처 개수) 만큼 선정
max_depth	- 트리의 최대 깊이 - default = None → 완벽하게 클래스 값이 결정될 때 까지 분할 또는 데이터 개수가 min_samples_split보다 작아질 때까지 분할 - 깊이가 깊어지면 과적합될 수 있으므로 적절히 제어 필요
max_leaf_nodes	리프노드의 최대 개수

1. 의사결정트리

❖ 의사결정트리(decision tree) 실습

```
from sklearn import tree
X = [[0, 0], [2, 3],[2, 1],[4, 7],[5, 4],[3, 2]]
Y = [0, 0,0,1,1,1]
clf = tree.DecisionTreeClassifier()
clf = clf.fit(X, Y)
```

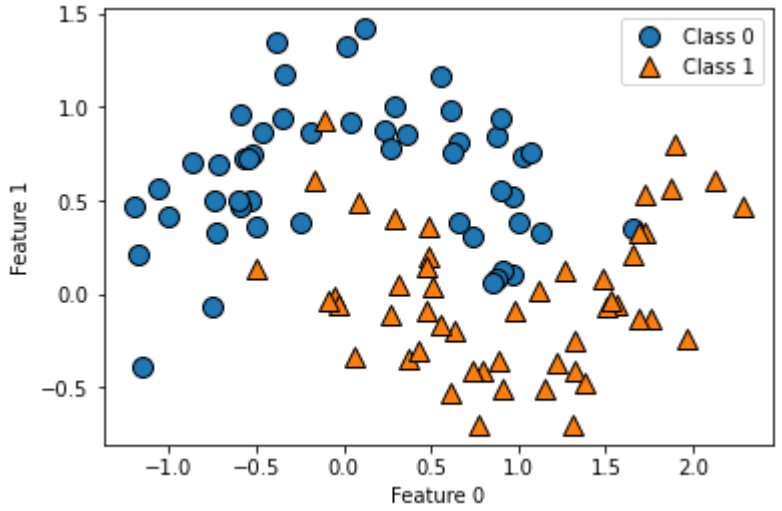
```
1  clf.predict([[2, 2]])
2
```

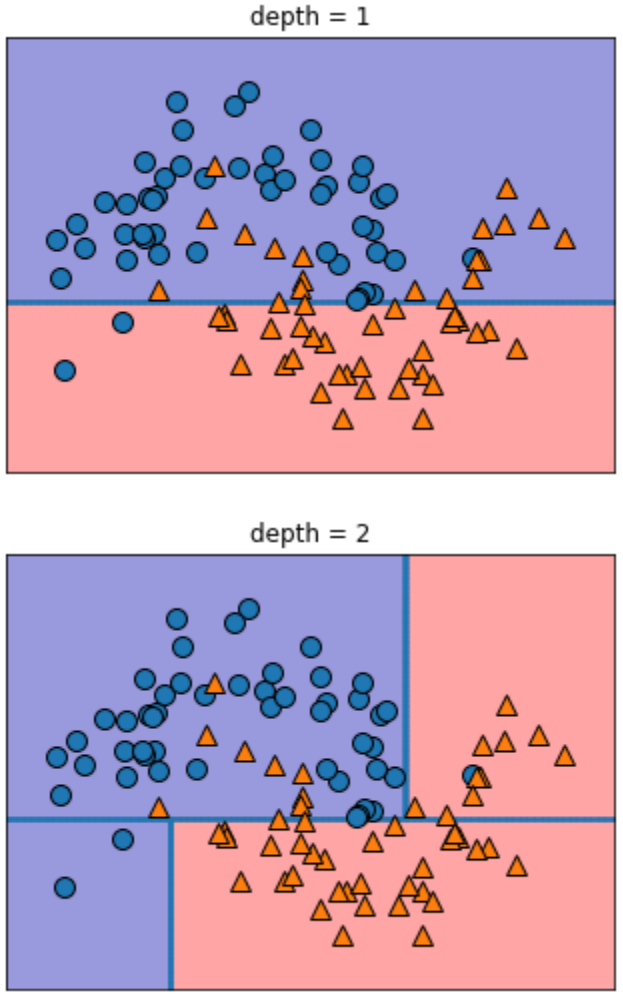
```
array([0])
```

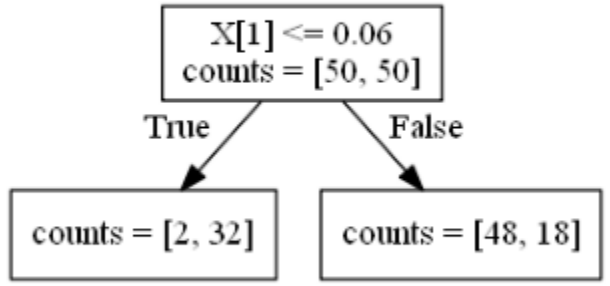
1. 의사결정트리

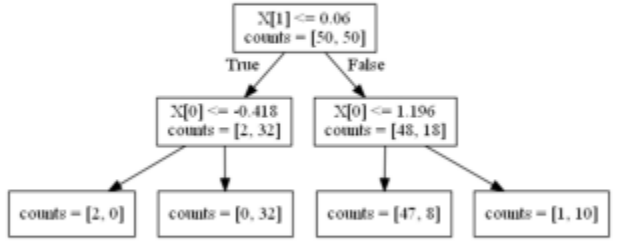
❖ 의사결정트리(decision tree) 실습

- **mglearn.plots.plot_tree_progressive()**





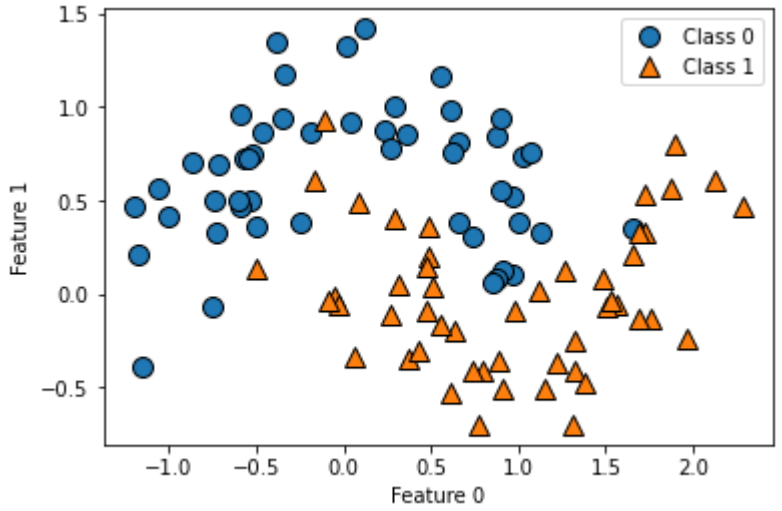


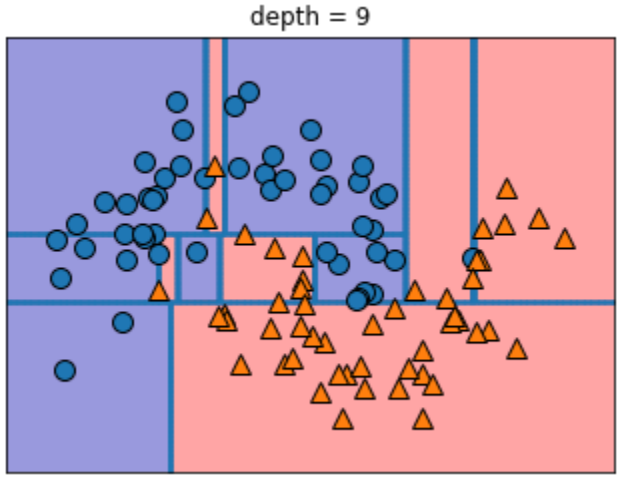


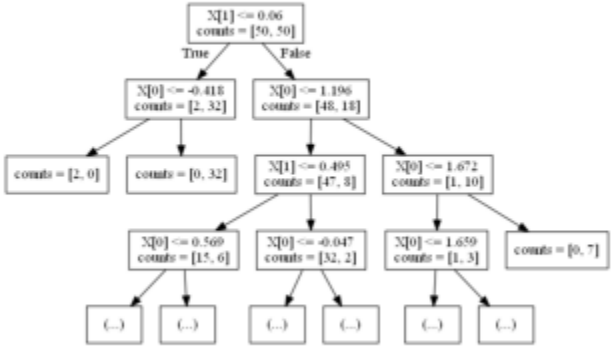
1. 의사결정트리

❖ 의사결정트리(decision tree) 실습

- **mglearn.plots.plot_tree_progressive()**







1. 의사결정트리

❖ 의사결정트리(decision tree) 실습

```
from sklearn import datasets
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
import numpy as np
from sklearn import tree

iris = datasets.load_iris()
# print(iris)
X = iris.data[:, :4]
y = iris.target

# 자동으로 데이터셋을 분리해주는 함수
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=0)

# 데이터 표준화 작업
sc = StandardScaler()
sc.fit(X_train)

# 표준화된 데이터셋
X_train_std = sc.transform(X_train)
X_test_std = sc.transform(X_test)
```

1. 의사결정트리

❖ 의사결정트리(decision tree) 실습

```
1  iris_tree = tree.DecisionTreeClassifier(criterion='entropy', max_depth=3, random_state=0)
2  iris_tree.fit(X_train, y_train)
```

```
DecisionTreeClassifier(ccp_alpha=0.0, class_weight=None, criterion='entropy',
                       max_depth=3, max_features=None, max_leaf_nodes=None,
                       min_impurity_decrease=0.0, min_impurity_split=None,
                       min_samples_leaf=1, min_samples_split=2,
                       min_weight_fraction_leaf=0.0, presort='deprecated',
                       random_state=0, splitter='best')
```

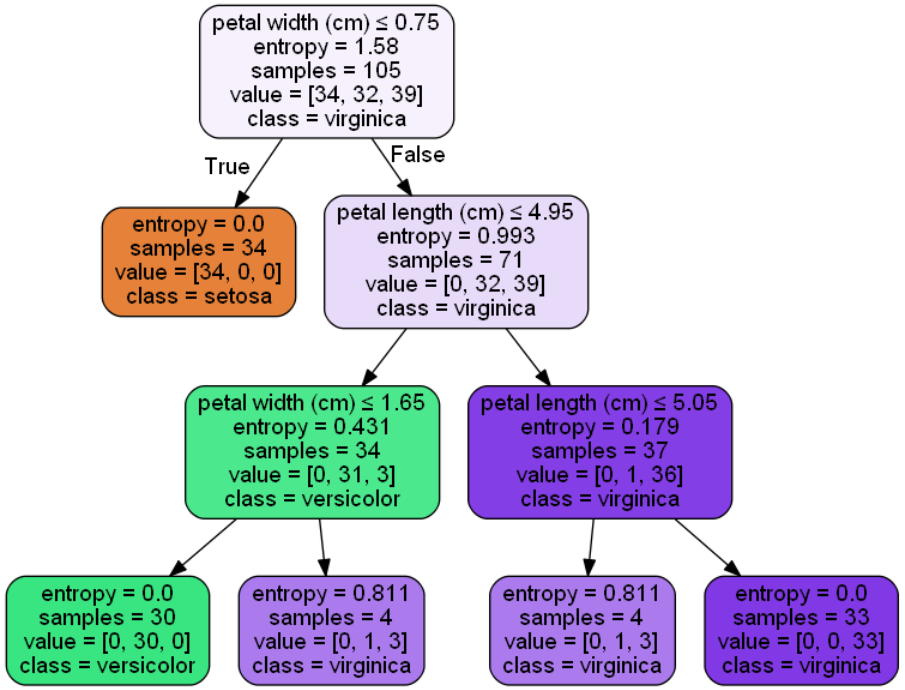
```
1  from sklearn.metrics import accuracy_score
2
3  y_pred_tr = iris_tree.predict(X_test)
4  print('Accuracy: %.2f' % accuracy_score(y_test, y_pred_tr))
```

```
Accuracy: 0.98
```

1. 의사결정트리

❖ 의사결정트리(decision tree) 실습

```
from sklearn.tree import export_graphviz
dot_data = export_graphviz(iris_tree, out_file=None, feature_names=['sepal length (cm)', 'sepal width (cm)', 'petal length (cm)', 'peta
                           class_names=iris.target_names, filled=True, rounded=True, special_characters=True)
graph = pydotplus.graph_from_dot_data(dot_data)
Image(graph.create_png())
```



1. 의사결정트리

❖ 의사결정트리(decision tree) 실습

```
from sklearn.tree import DecisionTreeClassifier
from sklearn.datasets import load_breast_cancer
from sklearn.model_selection import train_test_split, cross_val_score

breast_cancer_data = load_breast_cancer()


cancer = load_breast_cancer()
X_train, X_test, y_train, y_test = train_test_split(
    cancer.data, cancer.target, stratify=cancer.target, random_state=42)
tree = DecisionTreeClassifier(random_state=0)
tree.fit(X_train, y_train)
print("train set acc: {:.3f}".format(tree.score(X_train, y_train)))
print("test set acc: {:.3f}".format(tree.score(X_test, y_test)))
```

```
train set acc: 1.000
test set acc: 0.937
```

```
tree = DecisionTreeClassifier(max_depth=4, 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)))
```

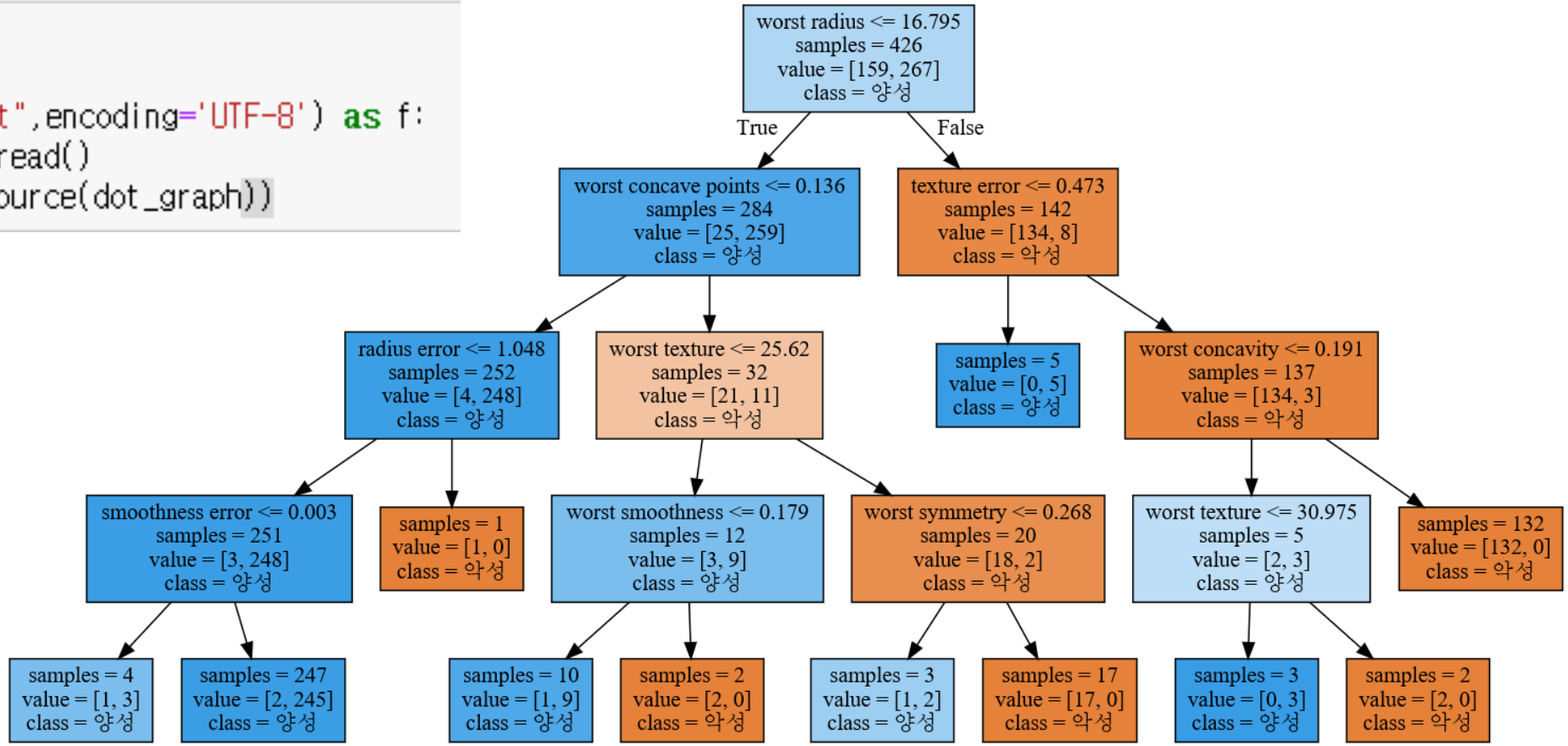
```
훈련 세트 정확도: 0.988
테스트 세트 정확도: 0.951
```

1. 의사결정트리

❖ 의사결정트리(decision tree) 실습

```
from sklearn.tree import export_graphviz
export_graphviz(tree, out_file="tree.dot", class_names=["악성", "양성"],
                feature_names=cancer.feature_names, impurity=False, filled=True)
```

```
import graphviz

with open("tree.dot",encoding='UTF-8') as f:
    dot_graph = f.read()
display(graphviz.Source(dot_graph))
```



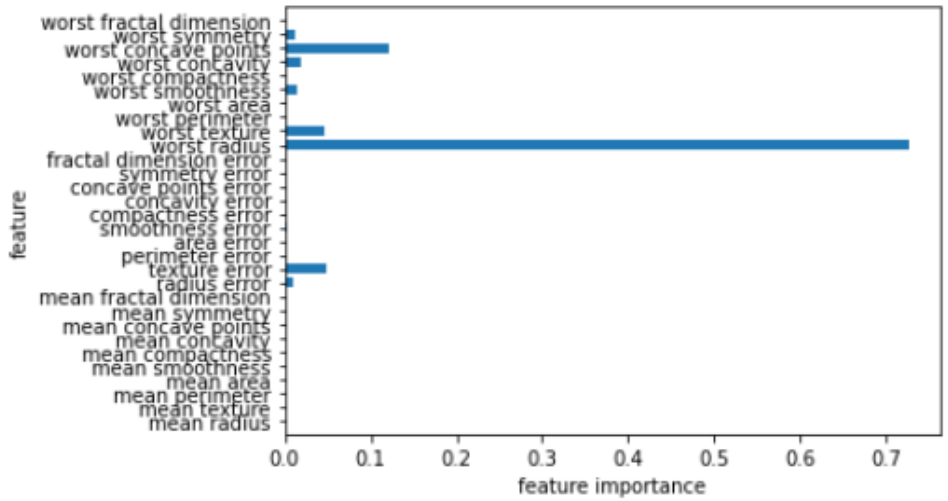
1. 의사결정트리

❖ 의사결정트리(decision tree) 실습

```
1  print("특성 중요도:₩n", tree.feature_importances_)
2
```

```
특성 중요도:
 [0.         0.         0.         0.         0.         0.
0.         0.         0.         0.         0.01019737 0.04839825
0.         0.         0.0024156  0.         0.         0.
0.         0.         0.72682851 0.0458159  0.         0.
0.0141577  0.         0.018188   0.1221132  0.01188548 0.        ]
```

```
1  import matplotlib.pyplot as plt
2  import numpy as np
3
4
5
6  def plot_feature_importances_cancer(model):
7      n_features = cancer.data.shape[1]
8      plt.barh(np.arange(n_features), model.feature_importances_, align='center')
9      plt.yticks(np.arange(n_features), cancer.feature_names)
10     plt.xlabel("feature importance")
11     plt.ylabel("feature")
12     plt.ylim(-1, n_features)
13
14 plot_feature_importances_cancer(tree)
15
```



1. 의사결정트리

❖ 의사결정트리(decision tree) 실습

```
from sklearn.model_selection import train_test_split
from sklearn.datasets import load_boston
from sklearn.metrics import mean_squared_error
```

```
X, y = load_boston(return_X_y=True)
X.shape # (506, 13)
```

```
(506, 13)
```

```
boston = load_boston()
X = boston.data
y = boston.target
colnames = boston.feature_names # 13개 칼럼 이름 가져올때
colnames
```

```
array(['CRIM', 'ZN', 'INDUS', 'CHAS', 'NOX', 'RM', 'AGE', 'DIS', 'RAD',
       'TAX', 'PTRATIO', 'B', 'LSTAT'], dtype='<U7')
```

1. 의사결정트리

❖ 의사결정트리(decision tree) 실습

```
model = DecisionTreeRegressor(max_depth=3)
model.fit(X = x_train, y = y_train)
```

```
DecisionTreeRegressor(ccp_alpha=0.0, criterion='mse', max_depth=3,
                      max_features=None, max_leaf_nodes=None,
                      min_impurity_decrease=0.0, min_impurity_split=None,
                      min_samples_leaf=1, min_samples_split=2,
                      min_weight_fraction_leaf=0.0, presort='deprecated',
                      random_state=None, splitter='best')
```

```
y_pred = model.predict(X)
y_true = y
```

```
mse = mean_squared_error(y_true, y_pred)
print('mse=', mse)
rmse = (np.sqrt(mse))
print("rmse :", rmse)

```

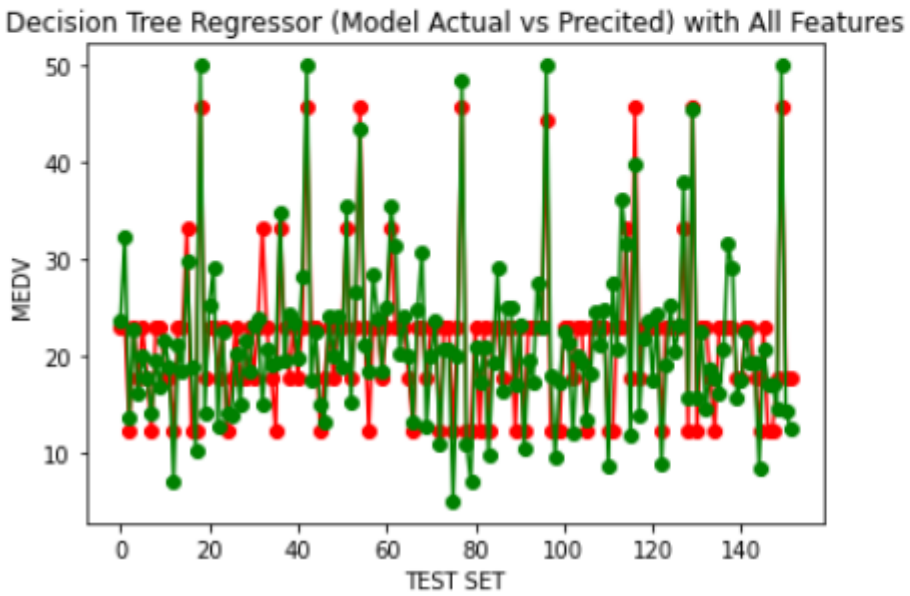
```
mse= 16.491246233330898
rmse : 4.06094154517262
```

1. 의사결정트리

❖ 의사결정트리(decision tree) 실습

```
# y_pred = regression.predict(X_test)
plt.figure()
plt.title("Decision Tree Regressor (Model Actual vs Precited) with All Features")
plt.xlabel('TEST SET')
plt.ylabel('MEDV')
plt.plot(y_pred, 'o-', color="r", label="Predicted MEDV")
plt.plot(y_test, 'o-', color="g", label="Actual MEDV")
```

```
[<matplotlib.lines.Line2D at 0x26208a0e518>]
```



1. 의사결정트리

❖ 의사결정트리(decision tree) 실습

```
from sklearn.datasets import load_wine

from sklearn.model_selection import train_test_split

from sklearn.tree import DecisionTreeClassifier

wine = load_wine()

x_train, x_test, y_train, y_test = train_test_split(wine.data, wine.target,random_state=0)

tree_5 = DecisionTreeClassifier(random_state=0, max_depth=5)
tree_5.fit(x_train, y_train)

score_tr = tree_5.score(x_train, y_train)
score_te = tree_5.score(x_test, y_test)

print('{:.3f}'.format(score_tr))
print('{:.3f}'.format(score_te))

```

```
1.000
0.933
```

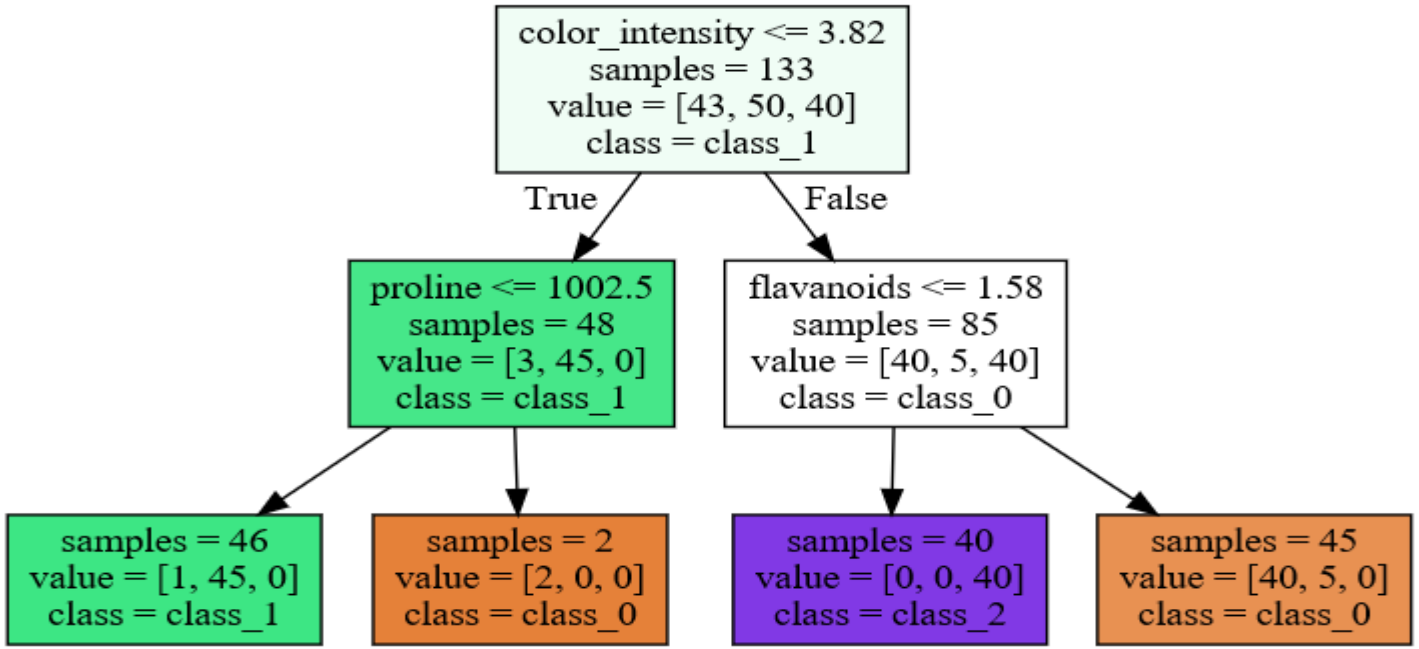
```
tree_2 = DecisionTreeClassifier(max_depth=2, random_state=0)
tree_2.fit(x_train, y_train)

score_tr = tree_2.score(x_train, y_train)
score_te = tree_2.score(x_test, y_test)

print('{:.3f}'.format(score_tr))
print('{:.3f}'.format(score_te))
```

```
0.955
0.844
```

1. 의사결정트리

❖ 의사결정트리(decision tree) 실습

```
import graphviz

from sklearn.tree import export_graphviz
from sklearn.tree import export_graphviz
export_graphviz(tree_2, out_file='aaaaa.dot',
                class_names=wine.target_names,
                feature_names=wine.feature_names,
                impurity=False, # gini 미출력
                filled=True) # filled: node의 색깔을 다르게

with open("aaaaa.dot",encoding='UTF-8') as f:
    dot_graph = f.read()
display(graphviz.Source(dot_graph))
```



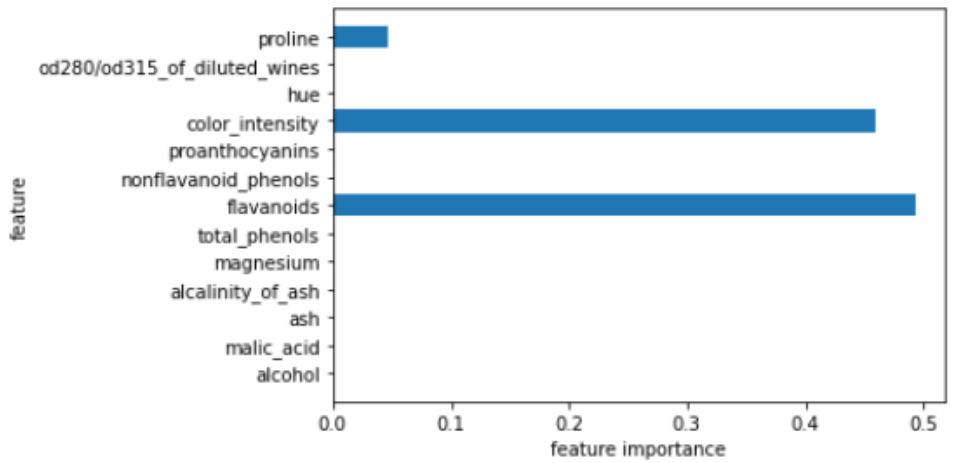
1. 의사결정트리

❖ 의사결정트리(decision tree) 실습

```
feature_imp = tree.feature_importances_
print('{}'.format(feature_imp))
```

```
[0.         0.         0.         0.         0.         0.
 0.49299068 0.         0.         0.45962843 0.         0.
 0.04738089]
```

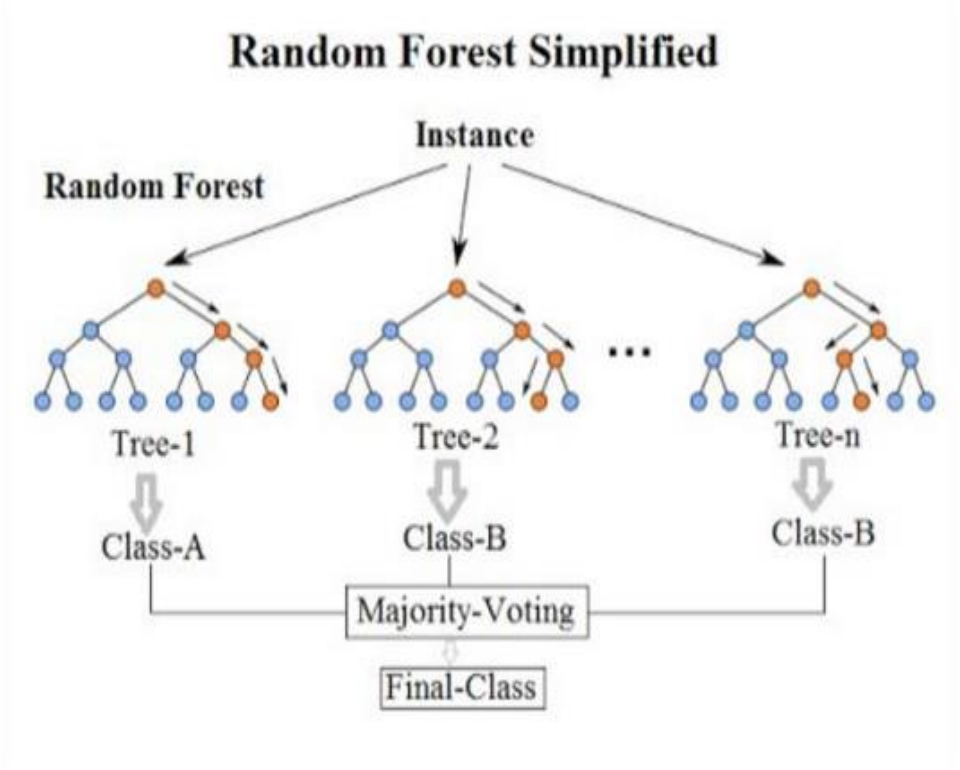
```
def plot_feature_importances_cancer(model):
    n_features = wine.data.shape[1]
    plt.barh(np.arange(n_features), model.feature_importances_, align='center')
    plt.yticks(np.arange(n_features), wine.feature_names)
    plt.xlabel("feature importance")
    plt.ylabel("feature")
    plt.ylim(-1, n_features)

plot_feature_importances_cancer(tree)
```



2. 랜덤포레스트

❖ Random Forest

- 2001년에 Leo Breiman에 의해 처음으로 소개된 기법
- Decision Tree의 단점을 개선하기 위한 알고리즘
- Random Forest는 훌륭한 데이터 분석 알고리즘
  - 데이터분류(classification)
  - 데이터군집(clustering)
  - Feature의 중요성확인
  - 데이터예측



출처 : https://medium.com/@williamkoehrsen/random-forest-simple-explanation-377895a60d2d

2. 랜덤포레스트

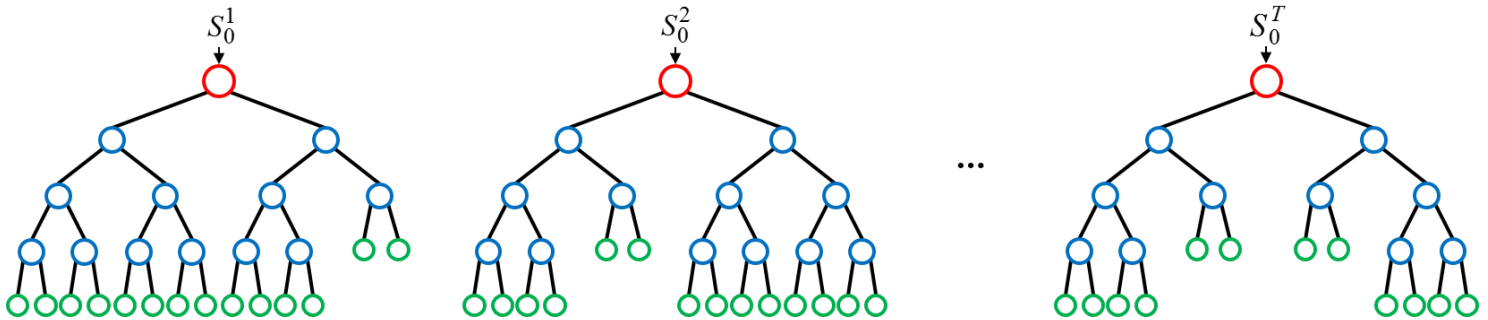
❖ Random Forest

- 의사결정 트리의 오버피팅 한계를 극복하기 위한 전략으로 **랜덤 포레스트(Random Forest) 등장**
- 데이터에 의사결정나무 여러 개를 동시에 적용해서 학습성능을 높이는 앙상블 기법
- 동일한 데이터로부터 복원추출을 통해 30개 이상의 데이터 셋을 만들어 각각에 의사결정나무를 적용한 뒤 학습 결과를 취합하는 방식

2. 랜덤포레스트

❖ Random Forest

- 배깅(bagging): bootstrap aggregating의 약자로, 부트스트랩(bootstrap)을 통해 조금씩 다른 훈련 데이터에 대해 훈련된 기초 분류기(base learner)들을 결합(aggregating)시키는 방법
- bootstrap sampling(복원추출)을 사용하며 decision tree 생성으로 algorithm으로 진행
- 트리들의 편향은 그대로 유지하면서, 분산은 감소시키기 때문에 포레스트의 성능을 향상

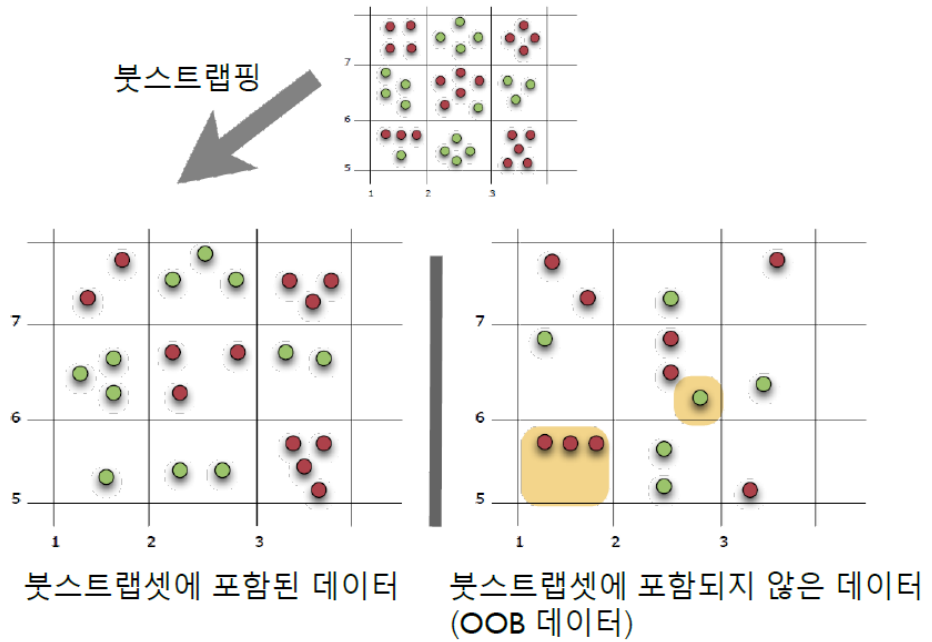


부트스트랩 방법을 통해 T}개의 훈련 데이터셋을 생성
T개의 기초 분류기(트리)들을 훈련
기초 분류기(트리)들을 하나의 분류기(랜덤 포레스트)로 결합(평균 또는 과반수투표 방식 이용).

2. 랜덤포레스트

❖ 변수의 중요도

- 선형 회귀모델/로지스틱 회귀모델과는 달리 개별 변수가 통계적으로 얼마나 유의한지에 대한 정보를 제공하지 않음
- **Out-Of-Bag(OOB)**
  - Oob 샘플은 위 흐름도에서 붓트스트랩 샘플링 과정에서 추출되지 않은 관측치
  - 샘플들은 주로 평가용 데이터에서의 오분류율을 예측하는 용도 및 변수 중요도를 추정하는 용도로 많이 이용



2. 랜덤포레스트

❖ Random Forest

- 장점
  - 다양성을 극대화 하여 예측력이 상당히 우수한 편
  - 다수의 트리의 예측 결과를 종합하여 의사결정을 진행하기 때문에 안정성도 상당히 높음
  - 랜덤화(randomization)는 포레스트가 노이즈가 포함된 데이터에 대해서도 강인
- 단점
  - 다수의 트리를 이용한 의사결정 기법을 이용하기 때문에 기존의 트리가 갖는 장점 중 하나인 설명력을 잃음

2. 랜덤포레스트

❖ Random Forest

- from sklearn.ensemble import RandomForestClassifier
- forest = RandomForestClassifier(n_estimators=5, random_state=2)
- forest.fit(X_train, y_train)
- forest.score(X_train,y_train)
- y_pred = forest.predict(x_test)
- metrics.accuracy_score(y_test, y_pred)

2. 랜덤포레스트

❖ Random Forest

파라미터 명	설명
n_estimators	bootstrap sampling은 random forest의 tree가 조금씩 다른 데이터셋으로 만들어지도록 함 max_feature에서 각 node는 특성의 일부를 무작위로 추출하여 사용
max_features	**max_features 값이 크면** random forest의 tree들은 같은 특성을 고려하므로 tree들이 매우 비슷해지고 가장 두드러진 특성을 이용해 데이터에 잘 맞춰짐 **max_features를 낮추면** random forest tree들은 많이 달라지고 각 tree는 데이터에 맞추기 위해 tree의 깊이가 깊어짐
max_Depth	트리의 깊이를 의미
Min_samples_leaf	리프노드가 되기 위한 최소한의 샘플 데이터의 수
Min_samples_split	노드를 분할하기 위한 최소한의 데이터수

2. 랜덤포레스트

❖ Random Forest

```
1  import pandas as pd
2  import numpy as np
3  from sklearn.ensemble import RandomForestClassifier
4  from sklearn.metrics import accuracy_score
5
```

2. 랜덤포레스트

❖ Random Forest

```
import numpy as np
from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestClassifier

x_data = np.array([
    [2, 1],
    [3, 2],
    [3, 4],
    [5, 6],
    [7, 5],
    [2, 1],
    [8, 9],
    [9, 10],
    [6, 12],
    [7, 2],
    [6, 10],
    [3, 4]
])
y_data = np.array([0, 0, 1, 1, 1, 0, 1, 1, 1, 1, 1, 0])

Label = ['Y', 'N']
```

2. 랜덤포레스트

❖ Random Forest

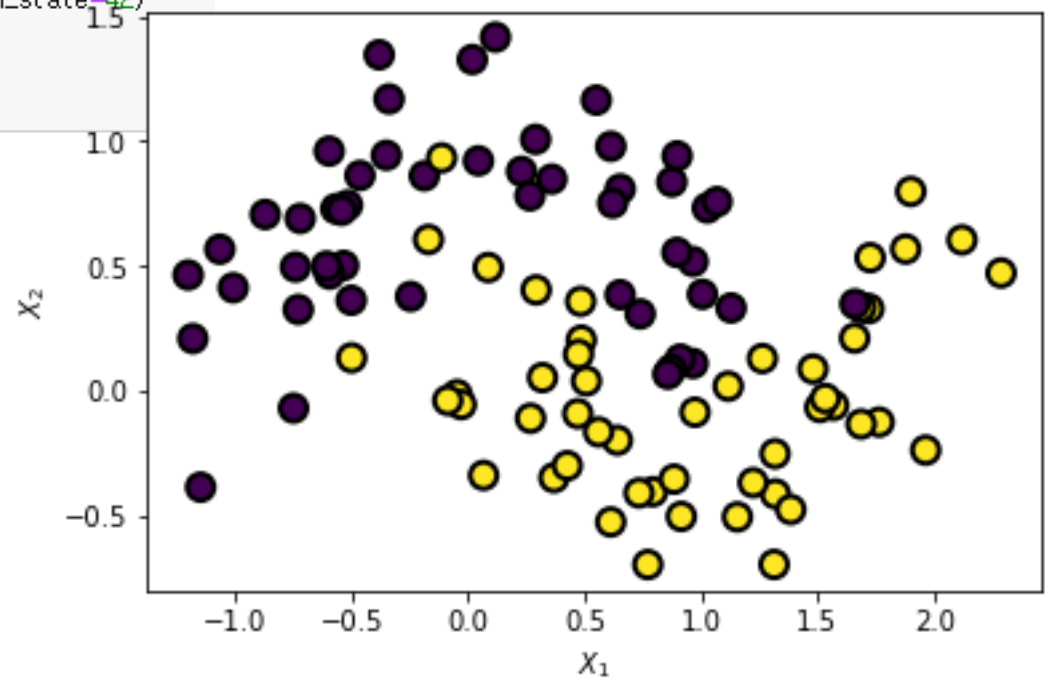
```
22
23 x_train, x_test, y_train, y_test = train_test_split(x_data, y_data, test_size=0.3, random_state=4)
24
25
26 model = RandomForestClassifier()
27 model.fit(x_train, y_train)
28
29 print(model.score(x_train, y_train)) #
30 print(model.score(x_test, y_test)) #1.0
31
32 x_test = np.array([
33     [2, 2]
34 ])
35 y_predict = model.predict(x_test)
36 print(Label[y_predict[0]]) #fail
```

```
0.875
0.75
Y
```

2. 랜덤포레스트

❖ Random Forest

```
from sklearn.ensemble import RandomForestClassifier
from sklearn.datasets import make_moons
from sklearn.model_selection import train_test_split, cross_val_score
X, y = make_moons(n_samples=100, noise=0.25, random_state=3)

plt.scatter(X[:, 0], X[:, 1], marker='o', c=y, s=100,
            edgecolor="k", linewidth=2)
plt.xlabel("$X_1$")
plt.ylabel("$X_2$")
plt.show()

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)
forest.fit(X_train, y_train)
```

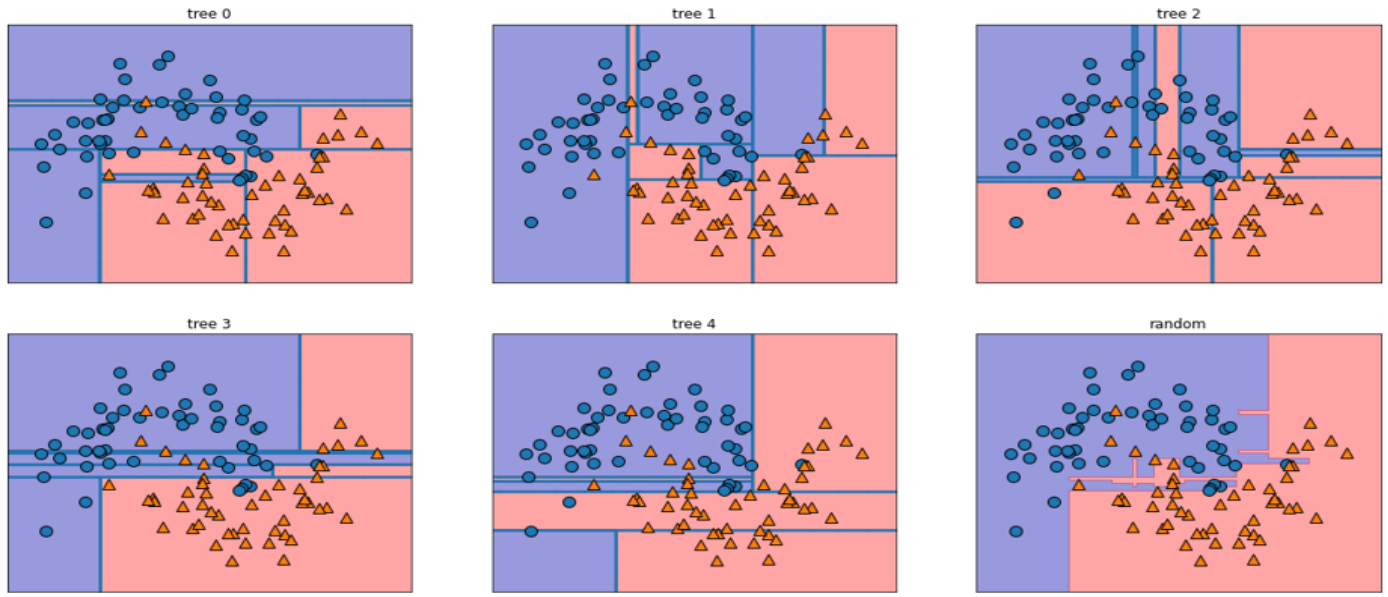


2. 랜덤포레스트

❖ Random Forest

```
import matplotlib.pyplot as plt
import numpy as np
import mglearn

fig, axes = plt.subplots(2, 3, figsize=(20, 10))
for i, (ax, tree) in enumerate(zip(axes.ravel(), forest.estimators_)):
    ax.set_title("tree {}".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("random")
mglearn.discrete_scatter(X[:, 0], X[:, 1], y)
```

```
[<matplotlib.lines.Line2D at 0x2605d91b588>,
 <matplotlib.lines.Line2D at 0x2605d91bc18>]
```



2. 랜덤포레스트

❖ Random Forest

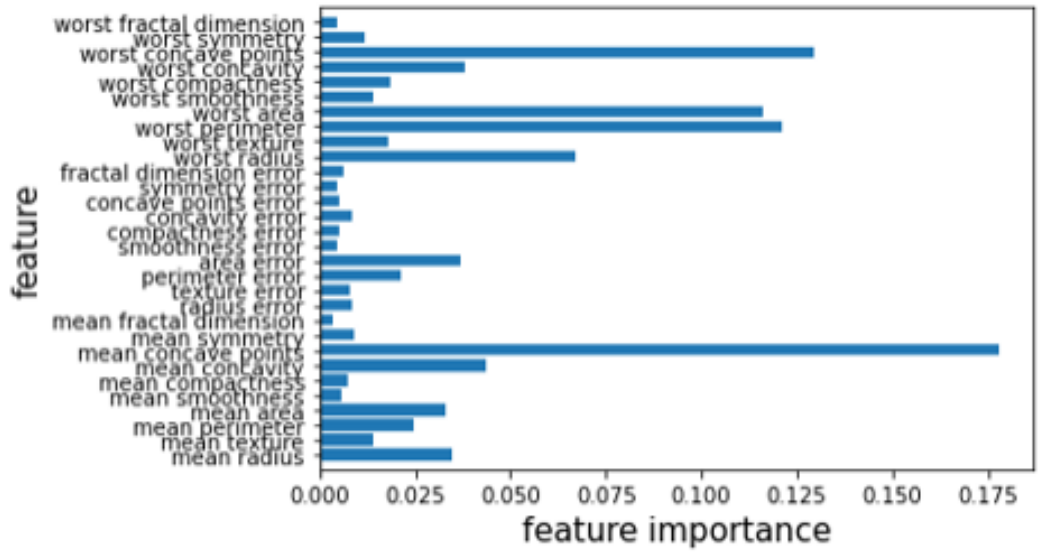
```
from sklearn.datasets import load_breast_cancer
from sklearn.model_selection import train_test_split, cross_val_score

cancer = load_breast_cancer()


X_train, X_test, y_train, y_test = train_test_split(
    cancer.data, cancer.target, random_state=0)
forest = RandomForestClassifier(n_estimators=100, random_state=0)
forest.fit(X_train, y_train)

print("train acc: {:.3f}".format(forest.score(X_train, y_train)))
print("test acc: {:.3f}".format(forest.score(X_test, y_test)))
```

```
train acc: 1.000
test acc: 0.972
```

2. 랜덤포레스트

❖ Random Forest

```
n_feature = cancer.data.shape[1]
index = np.arange(n_feature)

forest = RandomForestClassifier(n_estimators=100, n_jobs=-1)
forest.fit(X_train, y_train)
plt.barh(index, forest.feature_importances_, align='center')
plt.yticks(index, cancer.feature_names)
plt.ylim(-1, n_feature)
plt.xlabel('feature importance', size=15)

plt.ylabel('feature', size=15)

plt.show()
```



2. 랜덤포레스트

❖ Random Forest

```
#Import scikit-learn dataset library
from sklearn import datasets

#Load dataset
iris = datasets.load_iris()
```

```
# print the label species(setosa, versicolor,virginica)
print(iris.target_names)

# print the names of the four features
print(iris.feature_names)
```

```
['setosa' 'versicolor' 'virginica']
['sepal length (cm)', 'sepal width (cm)', 'petal length (cm)', 'petal width (cm)']
```

```
import pandas as pd
data=pd.DataFrame({
    'sepal length':iris.data[:,0],
    'sepal width':iris.data[:,1],
    'petal length':iris.data[:,2],
    'petal width':iris.data[:,3],
    'species':iris.target
})
data.head()
```

	sepal length	sepal width	petal length	petal width	species
0	5.1	3.5	1.4	0.2	0
1	4.9	3.0	1.4	0.2	0
2	4.7	3.2	1.3	0.2	0
3	4.6	3.1	1.5	0.2	0
4	5.0	3.6	1.4	0.2	0

2. 랜덤포레스트

❖ Random Forest

```
#Import Random Forest Model
from sklearn.ensemble import RandomForestClassifier

#Create a Gaussian Classifier
clf=RandomForestClassifier(n_estimators=100,random_state=4)

#Train the model using the training sets y_pred=clf.predict(X_test)
clf.fit(X_train,y_train)

y_pred=clf.predict(X_test)
```

```
#Import scikit-learn metrics module for accuracy calculation
from sklearn import metrics
# Model Accuracy, how often is the classifier correct?
print("Accuracy:",metrics.accuracy_score(y_test, y_pred))
```

```
Accuracy: 0.9333333333333333
```

```
import pandas as pd
feature_imp = pd.Series(clf.feature_importances_,index=iris.feature_names).sort_values(ascending=False)
feature_imp
```

```
petal width (cm)     0.452748
petal length (cm)    0.411520
sepal length (cm)    0.108719
sepal width (cm)     0.027014
dtype: float64
```

2. 랜덤포레스트

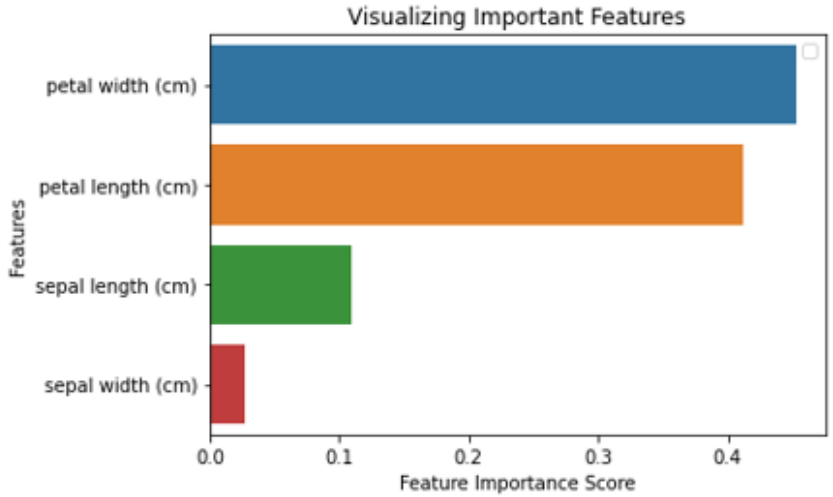
❖ Random Forest

```
import pandas as pd
feature_imp = pd.Series(clf.feature_importances_,index=iris.feature_names).sort_values(ascending=False)
feature_imp
```

```
petal width (cm)     0.452748
petal length (cm)    0.411520
sepal length (cm)    0.108719
sepal width (cm)     0.027014
dtype: float64
```

```
import matplotlib.pyplot as plt
import seaborn as sns
%matplotlib inline
# Creating a bar plot
sns.barplot(x=feature_imp, y=feature_imp.index)
# Add labels to your graph
plt.xlabel('Feature Importance Score')
plt.ylabel('Features')
plt.title("Visualizing Important Features")
plt.legend()
plt.show()

```

```
No handles with labels found to put in legend.
```



2. 랜덤포레스트

❖ Random Forest

```
from sklearn.model_selection import GridSearchCV

params = { 'n_estimators' : [10, 100],
           'max_depth' : [6, 8, 10, 12],
           'min_samples_leaf' : [8, 12, 18],
           'min_samples_split' : [8, 16, 20]
           }

# RandomForestClassifier 객체 생성 후 GridSearchCV 수행
rf_clf = RandomForestClassifier(random_state = 4, n_jobs = -1)
grid_cv = GridSearchCV(rf_clf, param_grid = params, cv = 5, n_jobs = -1)
grid_cv.fit(X_train, y_train)

print('최적 하이퍼 파라미터: ', grid_cv.best_params_)
print('최고 예측 정확도: {:.4f}'.format(grid_cv.best_score_))
```

```
최적 하이퍼 파라미터:  {'max_depth': 6, 'min_samples_leaf': 8, 'min_samples_split': 8, 'n_estimators': 10}
최고 예측 정확도: 0.9619
```

2. 랜덤포레스트

❖ Random Forest

```
from sklearn.ensemble import RandomForestRegressor # 회귀트리(모델)
from sklearn.model_selection import train_test_split # train/test
from sklearn.datasets import fetch_california_housing, load_boston # dataset
from sklearn.metrics import mean_squared_error # 평균제곱오차
```

```
X, y = load_boston(return_X_y=True)
X.shape # (506, 13)
```

```
(506, 13)
```

```
boston = load_boston()
X = boston.data
y = boston.target
colnames = boston.feature_names # 13개 칼럼 이름 가져올때
colnames
```

```
array(['CRIM', 'ZN', 'INDUS', 'CHAS', 'NOX', 'RM', 'AGE', 'DIS', 'RAD',
       'TAX', 'PTRATIO', 'B', 'LSTAT'], dtype='<U7')
```

```
x_train, x_test, y_train, y_test = train_test_split(X, y, test_size=0.3)
x_train.shape # (354, 13)
```

```
(354, 13)
```

2. 랜덤포레스트

❖ Random Forest

```
boston = load_boston()
X = boston.data
y = boston.target
colnames = boston.feature_names # 13개 칼럼 이름 가져올때
colnames
```

```
array(['CRIM', 'ZN', 'INDUS', 'CHAS', 'NOX', 'RM', 'AGE', 'DIS', 'RAD',
       'TAX', 'PTRATIO', 'B', 'LSTAT'], dtype='<U7')
```

```
x_train, x_test, y_train, y_test = train_test_split(X, y, test_size=0.3)
x_train.shape # (354, 13)
```

```
(354, 13)
```

```
model = RandomForestRegressor()
model.fit(X = x_train, y = y_train)
```

```
RandomForestRegressor(bootstrap=True, ccp_alpha=0.0, criterion='mse',
                      max_depth=None, max_features='auto', max_leaf_nodes=None,
                      max_samples=None, min_impurity_decrease=0.0,
                      min_impurity_split=None, min_samples_leaf=1,
                      min_samples_split=2, min_weight_fraction_leaf=0.0,
                      n_estimators=100, n_jobs=None, oob_score=False,
                      random_state=None, verbose=0, warm_start=False)
```

2. 랜덤포레스트

❖ Random Forest

```
modelrf = RandomForestRegressor(max_depth=3)
modelrf.fit(X = x_train, y = y_train)
```

```
RandomForestRegressor(bootstrap=True, ccp_alpha=0.0, criterion='mse',
                      max_depth=3, max_features='auto', max_leaf_nodes=None,
                      max_samples=None, min_impurity_decrease=0.0,
                      min_impurity_split=None, min_samples_leaf=1,
                      min_samples_split=2, min_weight_fraction_leaf=0.0,
                      n_estimators=100, n_jobs=None, oob_score=False,
                      random_state=None, verbose=0, warm_start=False)
```

```
y_pred = modelrf.predict(x_test)

mse = mean_squared_error(y_test, y_pred)
print('mse=', mse)
rmse = (np.sqrt(mse))
print("rmse :", rmse)

```

```
mse= 13.454873184042807
rmse : 3.6680884918500545
```

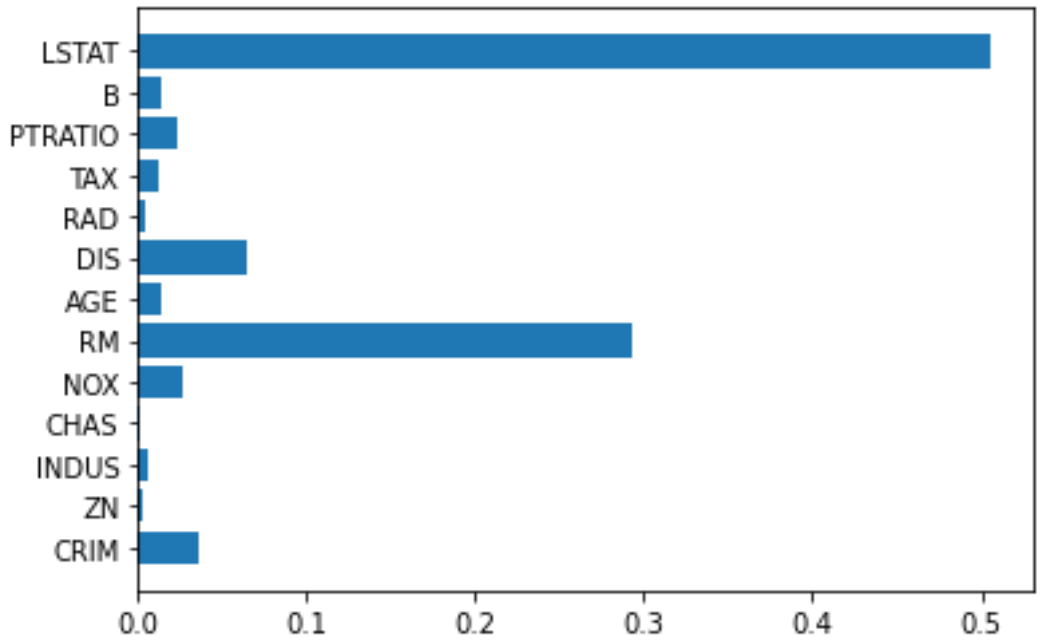
2. 랜덤포레스트

❖ Random Forest

```
y_pred = model.predict(x_test)

mse = mean_squared_error(y_test, y_pred)
print('mse=', mse)
rmse = (np.sqrt(mse))
print("rmse :", rmse)

```

```
mse= 9.2907813157894 63
rmse : 3.048078298828536
```

```
import matplotlib.pyplot as plt
plt.barh(range(13), imp) # (x, y) # 중요도 (y에 얼마나 영향을 미치는지)
plt.yticks(range(13), colnames) # 축 이름
```



2. 랜덤포레스트

❖ Random Forest

```
import pandas as pd
import statsmodels.api as sm
from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestRegressor # 회귀트리(모델)

wine_data = pd.read_csv('D:/big_data/winequality-white.csv',delimiter=';',dtype=float)
wine_data.head(10)
```

	fixed acidity	volatile acidity	citric acid	residual sugar	chlorides	free sulfur dioxide	total sulfur dioxide	density	pH	sulphates	alcohol	quality
0	7.0	0.27	0.36	20.7	0.045	45.0	170.0	1.0010	3.00	0.45	8.8	6.0
1	6.3	0.30	0.34	1.6	0.049	14.0	132.0	0.9940	3.30	0.49	9.5	6.0
2	8.1	0.28	0.40	6.9	0.050	30.0	97.0	0.9951	3.26	0.44	10.1	6.0
3	7.2	0.23	0.32	8.5	0.058	47.0	186.0	0.9956	3.19	0.40	9.9	6.0
4	7.2	0.23	0.32	8.5	0.058	47.0	186.0	0.9956	3.19	0.40	9.9	6.0
5	8.1	0.28	0.40	6.9	0.050	30.0	97.0	0.9951	3.26	0.44	10.1	6.0
6	6.2	0.32	0.16	7.0	0.045	30.0	136.0	0.9949	3.18	0.47	9.6	6.0
7	7.0	0.27	0.36	20.7	0.045	45.0	170.0	1.0010	3.00	0.45	8.8	6.0
8	6.3	0.30	0.34	1.6	0.049	14.0	132.0	0.9940	3.30	0.49	9.5	6.0
9	8.1	0.22	0.43	1.5	0.044	28.0	129.0	0.9938	3.22	0.45	11.0	6.0

```
x_data = wine_data.iloc[:,0:-1]
y_data = wine_data.iloc[:,-1]
```

2. 랜덤포레스트

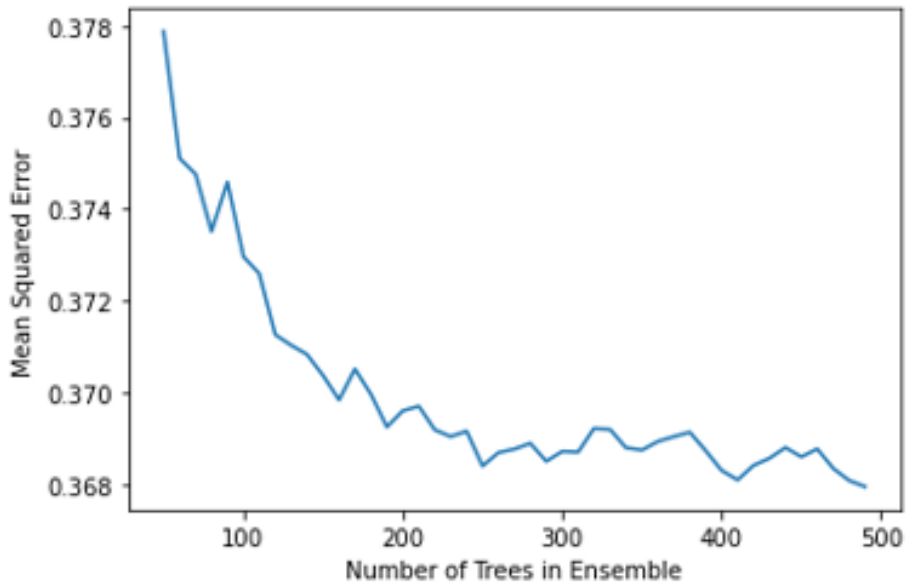
❖ Random Forest

```
#데이터 행의 30%로 고정된 홀드 아웃 세트 구성
xTrain, xTest, yTrain, yTest = train_test_split(x_data, y_data, test_size = 0.3, random_state = 531)

#MSE의 변화를 확인하기 위하여 앙상블의 크기 범위에서 랜덤 포레스트 트레이닝
mseOos = []
nTreeList = range(50, 500, 10)
for iTrees in nTreeList:
    depth = None
    maxFeat = 4 #조정해볼 것
    wineRFModel = RandomForestRegressor(n_estimators=iTrees,
                    max_depth=depth, max_features=maxFeat,
                    oob_score=False, random_state=531)
    wineRFModel.fit(xTrain, yTrain)
    #데이터 세트에 대한 MSE 누적
    prediction = wineRFModel.predict(xTest)
    mseOos.append(mean_squared_error(yTest, prediction))
```

2. 랜덤포레스트

❖ Random Forest

```
1  import numpy as np
2  import matplotlib.pyplot as plt
3
4  plt.plot(nTreeList, mseOos)
5  plt.xlabel('Number of Trees in Ensemble')
6  plt.ylabel('Mean Squared Error')
7  plt.show()
8
9
```

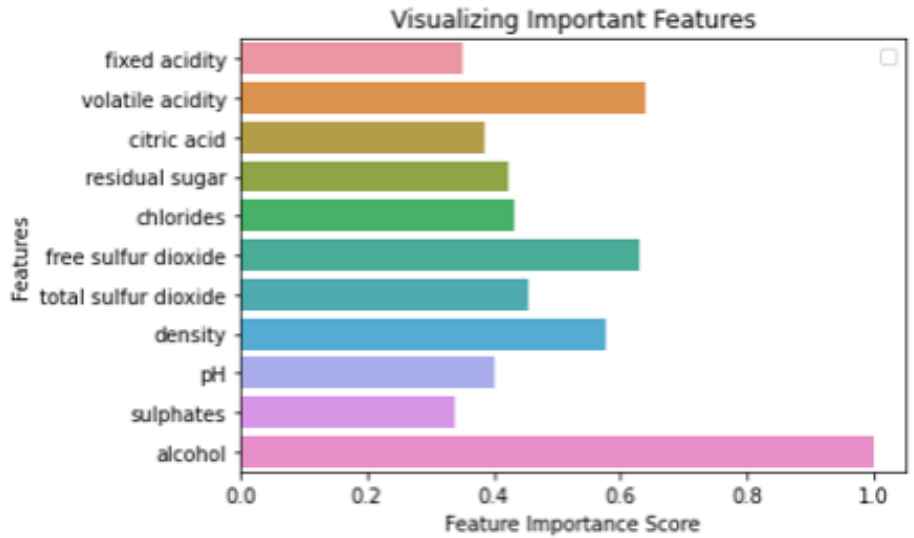


2. 랜덤포레스트

❖ Random Forest

```
1  import matplotlib.pyplot as plt
2  import seaborn as sns
3  %matplotlib inline
4  # Creating a bar plot
5  sns.barplot(x=featureImportance, y=x_data.columns)
6  # Add labels to your graph
7  plt.xlabel('Feature Importance Score')
8  plt.ylabel('Features')
9  plt.title("Visualizing Important Features")
10 plt.legend()
11 plt.show()
12
```

```
No handles with labels found to put in legend.
```



reference

- 모든 강의자료는 고려대 강필성 교수님 강의와 김성범 교수님 강의를 참고했음
- ratsgo’s blog ,https://ratsgo.github.io/
- 안드레아스 뮐러, 세라 가이도 지음, 박해선 옮김, "파이썬 라이브러리를 활용한 머신러닝", 한빛미디어(2017)
- 김의중 지음, "알고리즘으로 배우는 인공지능, 머신러닝, 딥러닝 입문", 위키북스(2016)
- https://en.wikipedia.org/wiki/ID3_algorithm, 위키피디아(영문), "ID3 algorithm"
- https://jihoonlee.tistory.com/16, 이지훈님의 블로그, 호옹호옹, "Decision Tree + ID3알고리즘"
- https://nittaku.tistory.com/277?category=745644, 동신한의 조재성, "3. 머신러닝 알고리즘: 의사결정 트리(Decision Tree) 알고리즘의 수학적 접근 - ID3 알고리즘"
- https://seamless.tistory.com/20, Data Engineer, "의사 결정 트리 (Decision Tree)"
- https://leedakyeong.tistory.com/entry/Decision-Tree란-ID3-알고리즘
- https://gomguard.tistory.com/86
- https://m.blog.naver.com/gksshdk8003/220914969026
- https://ko.wikipedia.org/wiki/%EB%9E%9C%EB%8D%A4_%ED%8F%AC%EB%A0%88%EC%8A%A4%ED%8A%B8
- 출처: https://hoony-gunputer.tistory.com/entry/핸즈온-머신러닝-6강-결정트리-Decision-tree [후니의 컴퓨터]

감사합니다