ML-C1

22.07.14

Supervised Learning(지도학습)

새로운 데이터에 대한 정확한 출력을 예측하는것.
라벨링(정답) 존재.

-> Classification / Regression

Classification(분류) : 미리 정의된 클래스 레이블 중 하나를 예측하는 것.

⇒Categorical Value ex) 붓꽃 품종 예측 -> Target_shape : (5,)

Regression(회귀) : 연속적인 숫자를 예측하는 것.

⇒Continuous Value ex) 2022년 옥수수 수확량 예측 -> 실수, 정수

일반화, 과대적합, 과소적합

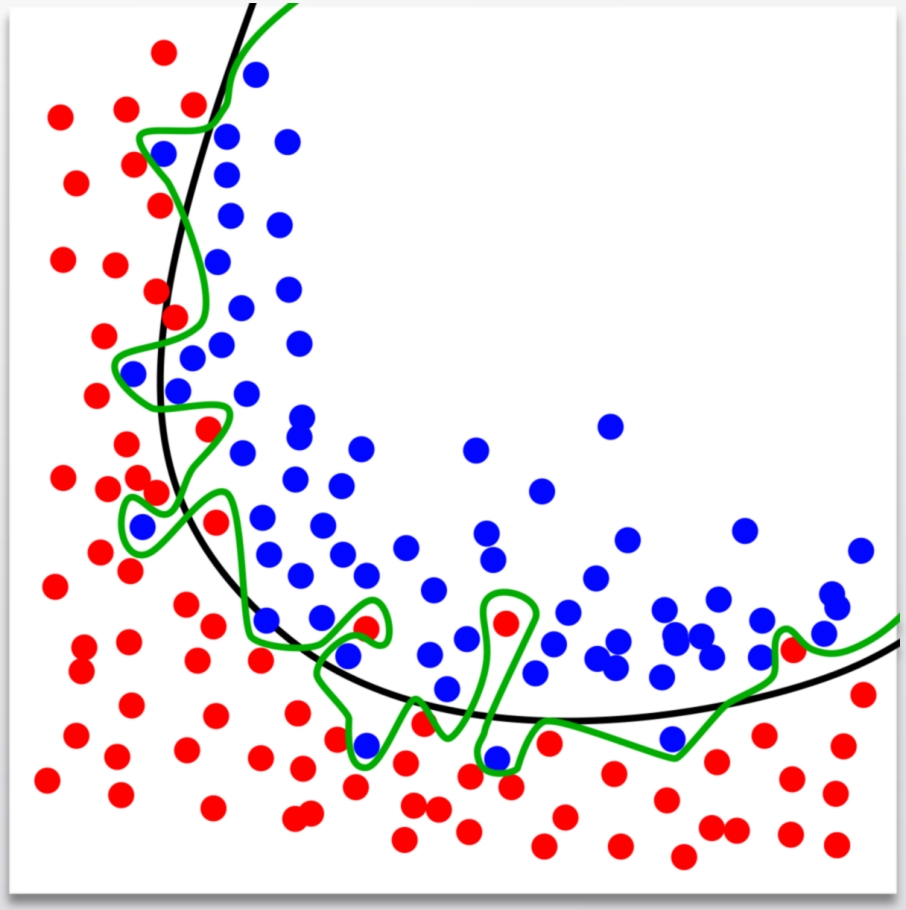
모델이 처음보는 데이터에 대해 정확하게
예측 할 수 있으면
훈련세트에서 테스트 세트로 일반화 되었다.

과대적합 : 너무 훈련데이터에 편향되었다.

Ex) 장미 (Train 데이터는 빨간 장미에 국한되었다.)

-> 빨갛고, 가시가 있고, 등등....

Test Data에는 빨간 장미외에도 파란장미, 노란장미,
가시가 없는 장미들이 포함되었다.



일반화, 과대적합, 과소적합

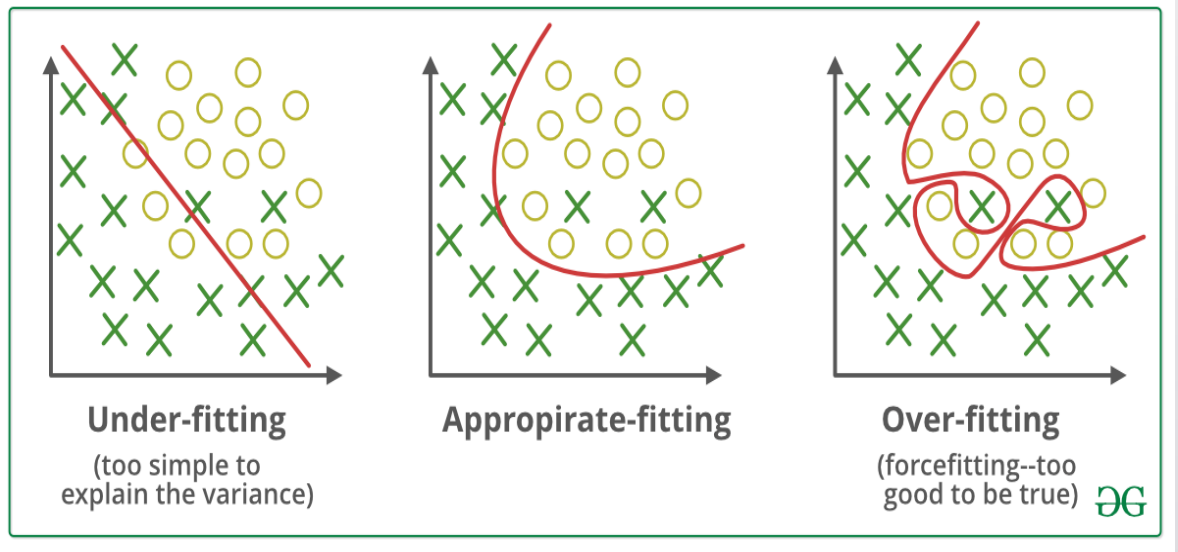
모델이 처음보는 데이터에 대해 정확하게
예측 할 수 있으면
훈련세트에서 테스트 세트로 일반화 되었다.

과소적합 : 너무 간단한 모델이 선택된다.

Ex) 장미

-> 꽃봉오리만 있으면 장미!

-> 과소적합과 과대적합 사이의 절충점을
찾는 것이 중요하다.



지도학습 알고리즘

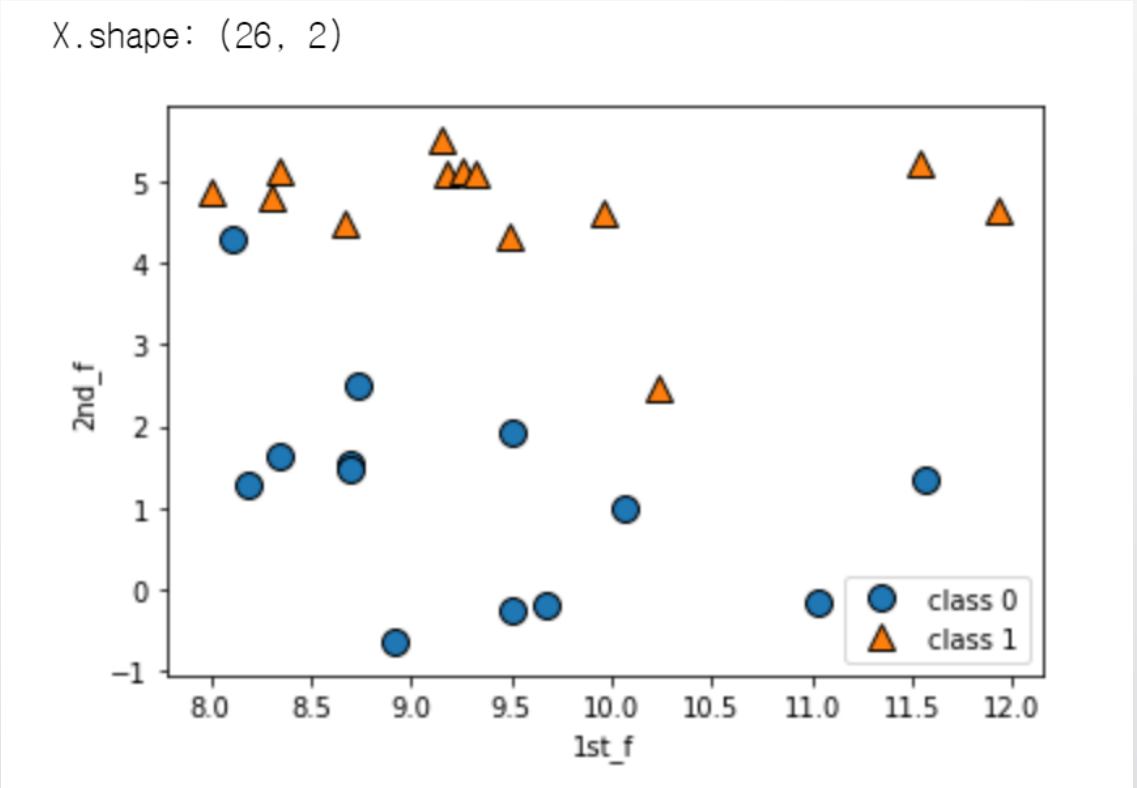
기본 포함 라이브러리

```
import numpy as np
from IPython.display import display
import matplotlib.pyplot as plt
import pandas as pd
import mglearn
```

Mglearn은 저자가 독자들의 학습을 위해 만든 라이브러리.

지도학습 알고리즘 예제1(Classification)

```
import numpy as np
from IPython.display import display
import matplotlib.pyplot as plt
import pandas as pd
import mglearn # mglearn은 저자가 만든 라이브러리로 임의의 데이터셋 만들기가 대부분.

# 랜덤으로 이중특성을 가진 이진분류 데이터셋을 만든다.
# X는 입력, y는 출력 즉 레이블이다.
X, y = mglearn. datasets.make_forge()

mglearn.discrete_scatter(X[:,0],X[:,1],y) # 첫번째 특성, 두번째 특성, target
plt.legend(["class 0","class 1"],loc=4) # loc=4는 lower right
plt.xlabel("1st_f") # x축_라벨
plt.ylabel("2nd_f") # y축_라벨
print("X.shape:",X.shape) # feature 크기
```



지도학습 알고리즘 예제1 추가설명

데이터셋을 만든다. X는 입력, y는 출력 즉 레이블이다.
X, y = mglearn. datasets.make_forge()

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

```
>>> print(x)
[[1 2]
 [3 4]
 [5 6]
 [7 8]]
>>> x[:,0]
array([1, 3, 5, 7])
>>> x[0:]
array([[1, 2],
       [3, 4],
       [5, 6],
       [7, 8]])
>>> x[0,:]
array([1, 2])
>>>
```

첫번째 특성, 두번째 특성, target
plt.legend(["class 0","class 1"],loc=4) # loc=4는 lower right
plt.xlabel("1st_f") # x축_라벨
plt.ylabel("2nd_f") # y축_라벨
print("X.shape:",X.shape) # feature 크기

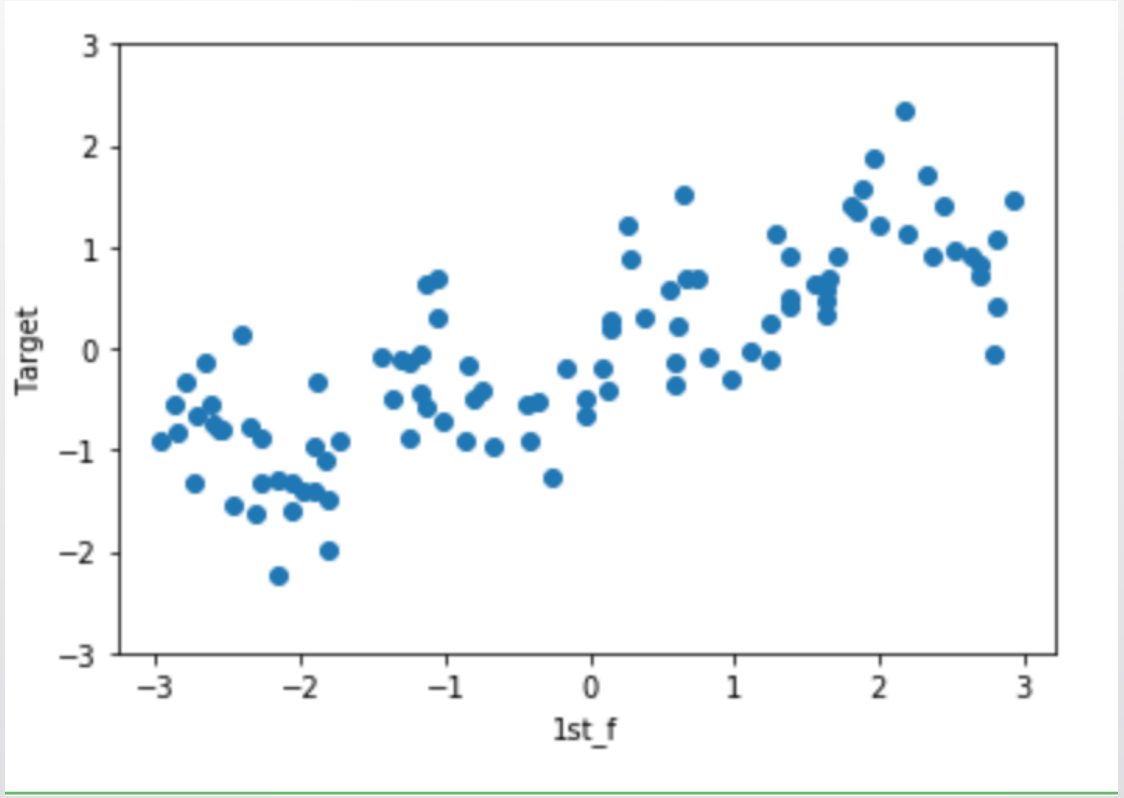
1. loc, 바운딩 박스 안

loc 인자는 미리 정해진 값들 중 하

Location String	Location Code
'best'	0
'upper right'	1
'upper left'	2
'lower left'	3
'lower right'	4
'right'	5
'center left'	6
'center right'	7
'lower center'	8
'upper center'	9
'center'	10

지도학습 알고리즘 예제2(Regression)

```
import numpy as np
from IPython.display import display
import matplotlib.pyplot as plt
import pandas as pd
import mglearn # mglearn은 저자가 만든 라이브러리로 임의의 데이터셋 만들기가 대부분.

# 데이터셋을 만든다. X는 입력, y는 출력 즉 레이블이다.
# Continuous Value가 y이다.
X, y = mglearn. datasets.make_wave()

plt.plot(X,y,'o') # x축 특징 / y축은 레이블 / o는 circle형태로 그래프
plt.ylim(-3,3)
plt.xlabel("1st_f") # x축_라벨
plt.ylabel("Target") # y축_라벨
```



지도학습 알고리즘 예제2(Regression) 추가설명

데이터셋을 만든다. X는 입력, y는 출력 즉 레이블이다.
Continuous Value가 y이다.
X, y = mglearn. datasets.make_wave()

plt.plot(X,y,'o')
# x축 특징 / y축은 레이블 / o는 circle형태로 그래프
plt.ylim(-3,3) #y축 값제한
plt.xlabel("1st_f") # x축_라벨
plt.ylabel("Target") # y축_라벨

Character	Description
'-'	Solid line style
'--'	Dashed line style
'-.'	Dash-dot line style
':'	Dotted line style
'.'	Point marker
','	Pixel marker
'o'	Circle marker
'v'	Triangle_down marker
'^'	Triangle_up marker
'<'	Triangle_left marker
'>'	Triangle_right marker
'1'	Tri_down marker
'2'	Tri_up marker
'3'	Tri_left marker
'4'	Tri_right marker
's'	Square marker
'p'	Pentagon marker
'*'	Star marker
'h'	Hexagon1 marker
'H'	Hexagon2 marker
'+'	Plus marker
'x'	X marker
'D'	Diamond marker
'd'	Thin_diamond marker
'|'	Vline marker
'_'	Hline marker

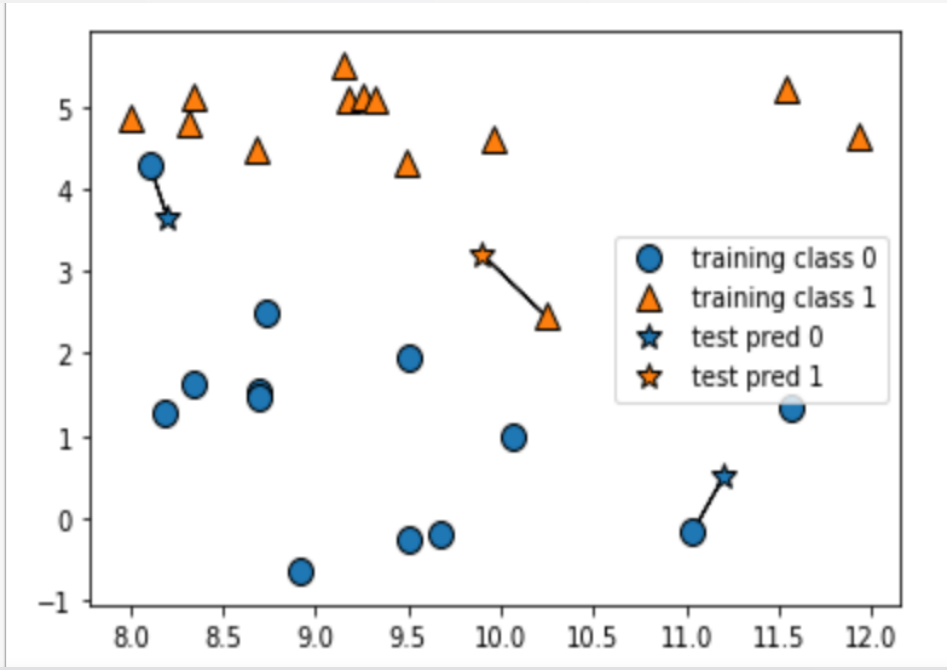
Character	Color
'b'	Blue
'g'	Green
'r'	Red
'c'	Cyan
'm'	Magenta
'y'	Yellow
'k'	Black
'w'	White

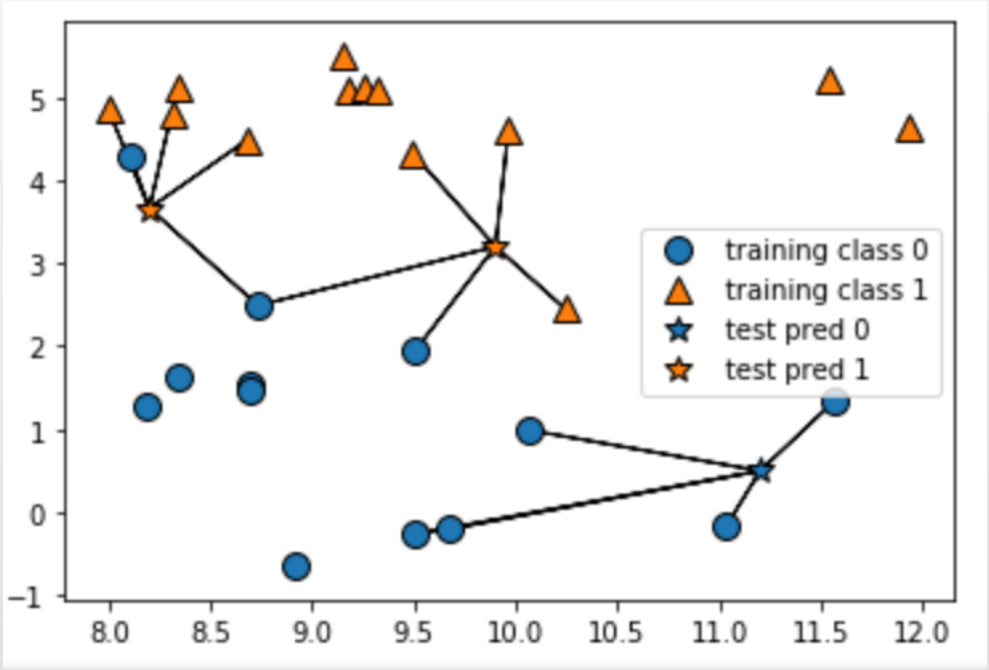
지도학습 알고리즘 예제 유방암 종양 임상데이터

```
import numpy as np
from IPython.display import display
import matplotlib.pyplot as plt
import pandas as pd
import mglearn # mglearn은 저자가 만든 라이브러리로
임의의 데이터셋 만들기가 대부분.
from sklearn.datasets import load_breast_cancer
#사이킷런 데이터셋 중 유방암데이터가져오기.
cancer=load_breast_cancer()
cancer.keys()
#dict_keys(['data', 'target', 'target_names', 'DESCR',
'feature_names', 'filename'])
# 사이킷런에 저장되어있는 데이터셋은 딕셔너리 형태가
아닌 번치형태로 저장되어있어,
#접근시 dot연산자를 통해 접근이 가능하다.
```

K-최근접 이웃 => 유방암데이터

K(Hyper Parameter)가 1일경우와 5일경우.
mglearn.datasets.make_forge()





K-최근접 이웃 => 유방암데이터

훈련데이터 셋에서 가장 가까운 데이터 포인트, 최근접 이웃을 찾는다.

```
import numpy as np
from IPython.display import display
from sklearn.neighbors import KNeighborsClassifier
from sklearn.model_selection import train_test_split
import matplotlib.pyplot as plt
import pandas as pd

import mglearn # mglearn은 저자가 만든 라이브러리로 임의의 데이터셋 만들기가 대부분.
from sklearn.datasets import load_breast_cancer #사이킷런 데이터셋 중
유방암데이터가져오기.
b_cancer = load_breast_cancer()

X_train, X_test, y_train, y_test =
train_test_split(b_cancer.data,b_cancer.target,random_state=0)
knn = KNeighborsClassifier(n_neighbors =5)
knn.fit(X_train,y_train)
a=knn.predict(X_test)
print("정확도 : {:.2f}".format(np.mean(a==y_test)))
```

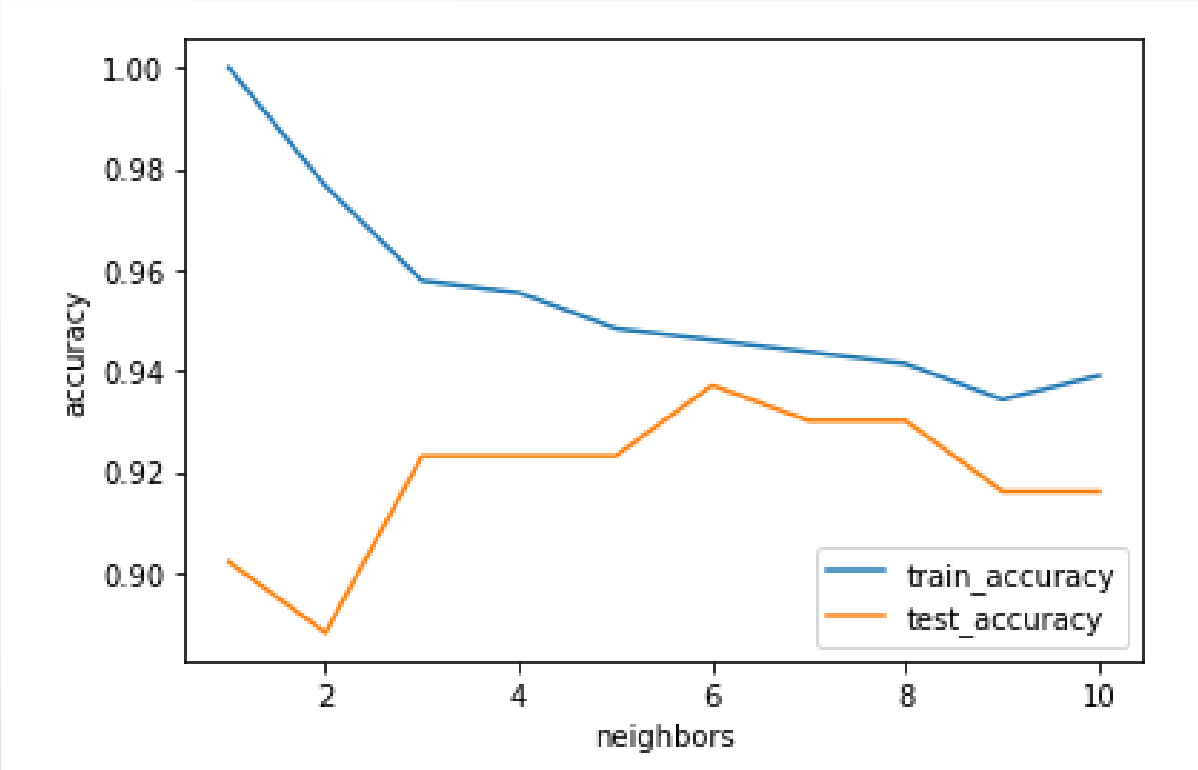
K-최근접 이웃 => 유방암데이터 분류 결과 분석

```
import numpy as np
from IPython.display import display
from sklearn.neighbors import KNeighborsClassifier
from sklearn.model_selection import train_test_split
import matplotlib.pyplot as plt
import pandas as pd

import mglearn # mglearn은 저자가 만든 라이브러리로 임의의 데이터셋 만들기가 대부분.
from sklearn.datasets import load_breast_cancer #사이킷런 데이터셋 중 유방암데이터가져오기.
b_cancer = load_breast_cancer() # Bunch class 타입으로 받아옴. => 딕셔녀리와 같이 [“target”].으로 접근하지 않아도됨.

X_train, X_test, y_train, y_test = train_test_split(b_cancer.data,b_cancer.target,stratify=b_cancer.target,random_state=66)
Training_ac = [] #훈련정확도를 저장할 리스트
Test_ac = [] #테스트 정확도를 저장할 리스트
Neighbors_set = range(1,11) # 이웃수 1~11

For n in neighbors_set :
    clf = KNeighborsClassifier(n_neighbors =n) #이웃수가 n인 knn모델 생성
    clf.fit(X_train,y_train) #훈련데이터 모델에 적용
    training_ac.append(clf.score(X_train, y_train)) # 이웃 n의 knn 훈련데이터 정확도를 저장함.
    test_ac.append(칠.score(X_test, y_test)) # 이웃 n의 knn 테스트데이터 정확도를 저장함.


Plt.plot(neighbors_set,training_ac,label= " train_accuracy " ) # x축 1~11 / y축 훈련테스트 정확도 / 라벨값의 텍스트 정의
Plt.plot(neighbors_set,test_ac,label="test_accuracy") # 윗라인과 이하동문
Plt.xlabel("neighbors") # x축 라벨
plt.ylabel("accuracy") # y축 라벨
plt.legend(loc=4) # 범주 위치 loc 생략시 우측 상단.
```

K-최근접 이웃 => 유방암데이터 분류 결과 분석

의미하는 바 :
이웃수가 적다는 것은 너무 복잡한 모델을 의미
-> 특징이 많다. -> 과대적합 (훈련 정확도는 높지만, 테스트 정확도는 낮다.)

이웃수가 많다는 것은 너무 간단한 모델을 의미
-> 특징이 적다. -> 과소적합 (훈련정확도도 낮고 테스트 정확도도 낮다.)



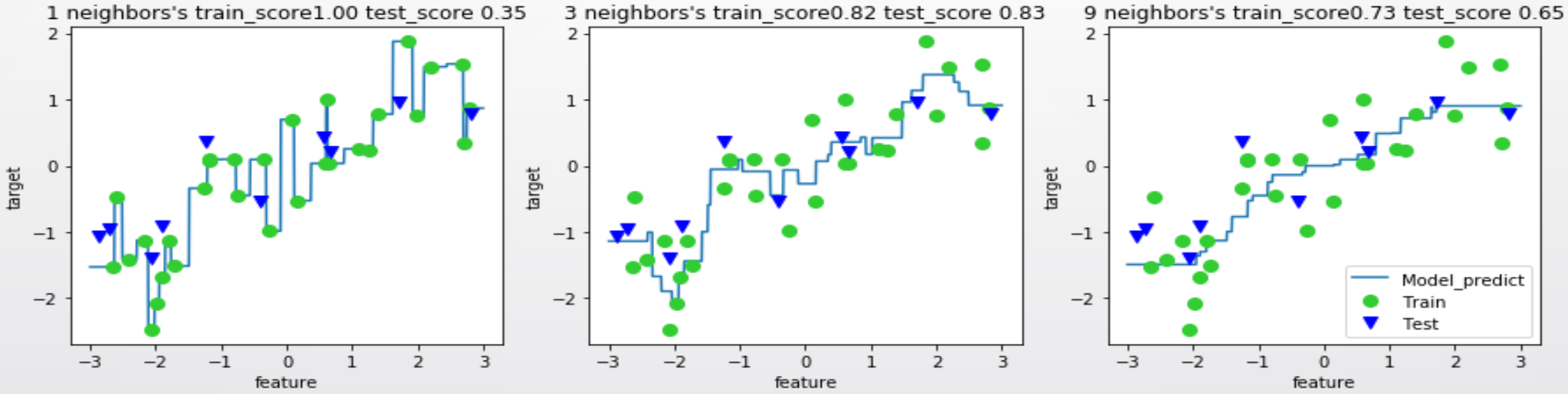
K-최근접 이웃 => 회귀

```
import numpy as np
from IPython.display import display
from sklearn.neighbors import KNeighborsRegressor #knn회귀는 KNeighborsRegressor에 구현되어있음.
From sklearn.model_selection import train_test_split
Import matplotlib.pyplot as plt
Import pandas as pd

Import mglearn # mglearn은 저자가 만든 라이브러리로 임의의 데이터셋 만들기가 대부분.
From sklearn.datasets import load_breast_cancer #사이킷런 데이터셋 중 유방암데이터가져오기.
X,y = mglearn.datasets.make_wave(n_samples=40) # 40개의 샘플로 제한.
X_train,X_test,y_train,y_test = train_test_split(X,y,random_state=0)
kn=KNeighborsRegressor(n_neighbors = 3)
kn.fit(X_train,y_train)
print(kn.score(X_test,y_test))
```

K-최근접 이웃 => 회귀 결과 분석

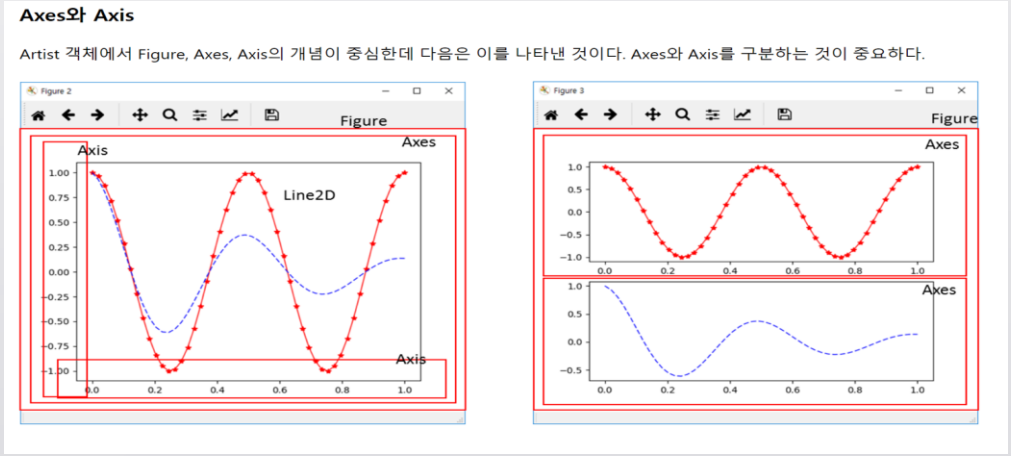
```
import numpy as np
from IPython.display import display
from sklearn.neighbors import KNeighborsRegressor
from sklearn.model_selection import train_test_split
import matplotlib.pyplot as plt
import pandas as pd

import mglearn # mglearn은 저자가 만든 라이브러리로 임의의 데이터셋 만들기가 대부분.
from sklearn.datasets import load_breast_cancer #사이킷런 데이터셋 중 유방암데이터가져오기.
X,y = mglearn.datasets.make_wave(n_samples=40)
X_train,X_test,y_train,y_test = train_test_split(X,y,random_state=0)
fig, axes = plt.subplots(1,3 ,figsize= (15,4))
line = np.linspace(-3,3,1000).reshape(-1,1) #행당 1개의 원소를 가진다.
for n , axe in zip([1,3,9],axes) :
    kn = KNeighborsRegressor(n_neighbors = n)
    kn.fit(X_train,y_train)
    axe.plot(line,kn.predict(line))
    axe.plot(X_train,y_train,'o',color='limegreen',markersize=8)
    axe.plot(X_test,y_test,'v',color='blue',markersize=8)
    axe.set_title("{} neighbors's train_score{:.2f} test_score {:.2f}".format(n,kn.score(X_train, y_train),kn.score(X_test,y_test)))
    axe.set_xlabel("feature")
    axe.set_ylabel("target")
axes[2].legend(["Model_predict","Train","Test"],loc=4)
```

K-최근접 이웃 => 회귀 결과 분석



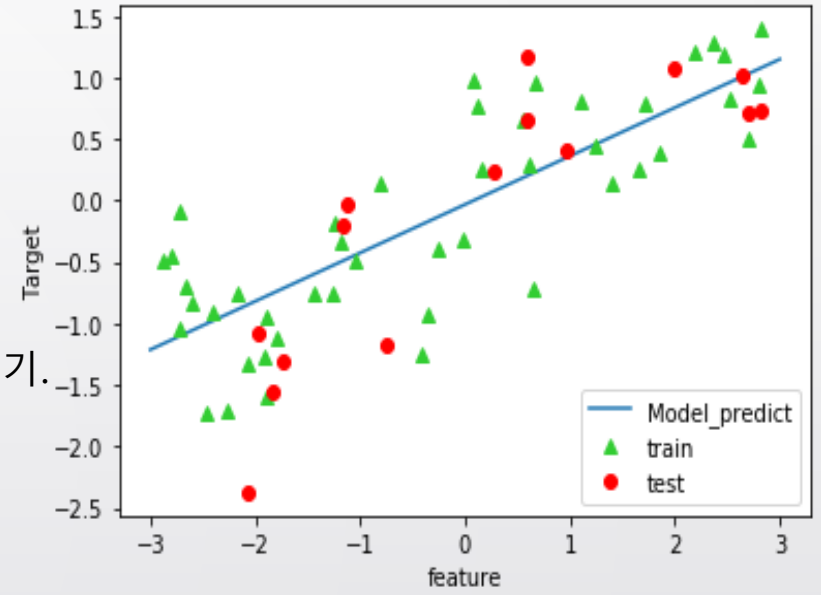
의미하는 바 :
이웃수가 적다는 것은 너무 복잡한 모델을 의미
-> 특징이 많다. -> 과대적합 (훈련 정확도는 높지만, 테스트 정확도는 낮다.)

이웃수가 많다는 것은 너무 간단한 모델을 의미
-> 특징이 적다. -> 과소적합 (훈련정확도도 낮고 테스트 정확도도 낮다.)



회귀 : 선형회귀(Linear_Regression)

```
import numpy as np
from IPython.display import display
from sklearn.neighbors import KNeighborsRegressor
from sklearn.linear_model import LinearRegression
from sklearn.model_selection import train_test_split
import matplotlib.pyplot as plt
import pandas as pd

import mglearn # mglearn은 저자가 만든 라이브러리로 임의의 데이터셋 만들기가 대부분.
from sklearn.datasets import load_breast_cancer #사이킷런 데이터셋 중 유방암데이터가져오기.
X,y = mglearn.datasets.make_wave(n_samples=60)
X_train,X_test,y_train,y_test = train_test_split(X,y,random_state=42)
lr = LinearRegression().fit(X_train,y_train)
```




```
line=np.linspace(-3,3,1000).reshape(-1,1) #1열짜리 벡터로 재정의함. -> 특성이 하나다. Wave_dataset은 그래서 1열
plt.plot(line,lr.predict(line)) # line(범위)에 해당하는 선형예측선(학습된 기울기와 절편에 특성값과 곱해서 사용.)
plt.plot(X_train,y_train,"^",color="limegreen")
plt.plot(X_test,y_test,"o",color="red")
plt.xlabel("feature")
plt.ylabel("Target")
plt.legend(["Model_predict" ,"train" ,"test"],loc=4)
```

회귀 : 선형회귀(Linear_Regression) 예제2 특성 106개

```
import numpy as np
from IPython.display import display
from sklearn.neighbors import KNeighborsRegressor
from sklearn.linear_model import LinearRegression
from sklearn.model_selection import train_test_split
import matplotlib.pyplot as plt
import pandas as pd

import mglearn # mglearn은 저자가 만든 라이브러리로 임의의 데이터셋
만들기가 대부분.
from sklearn.datasets import load_breast_cancer #사이킷런 데이터셋 중
유방암데이터가져오기.
X,y = mglearn.datasets.load_extended_boston()
X_train,X_test,y_train,y_test = train_test_split(X,y,random_state=0)
lr = LinearRegression().fit(X_train,y_train)
print("train_score : {:.2f}".format(lr.score(X_train,y_train)))
print("test_score : {:.2f}".format(lr.score(X_test,y_test)))
```

훈련데이터 성능은 높은 반면
테스트데이터 성능은 낮다. -> 과대적합을 의미한다. 특성이 너무 많다. ->
복잡도를 제어할 수 없는 단점이 존재한다.

```
train_score : 0.95 test_score : 0.61
```

회귀 : Ridge선형회귀(Ridge Linear_Regression) 예제2 특성 106개

```
import numpy as np
from IPython.display import display
from sklearn.neighbors import KNeighborsRegressor
from sklearn.linear_model import LinearRegression
from sklearn.model_selection import train_test_split
from sklearn.linear_model import Ridge
import matplotlib.pyplot as plt
import pandas as pd
```

일반 선형회귀와 다른점은 L2규제를 통해 기울기 w를 0에 가깝게 한다는 것이고 이는 특징이 출력에 주는 영향을 줄이는 것.

규제란 과대적합을 막기위해 강제적으로 모델을 제한하는것을 뜻함.

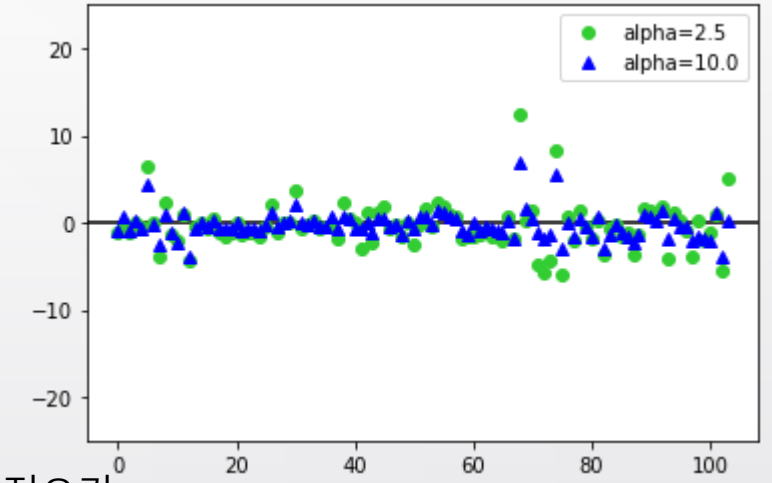
```
train_score : 0.89 test_score : 0.75
```

```
import mglearn # mglearn은 저자가 만든 라이브러리로 임의의 데이터셋 만들기가 대부분.
from sklearn.datasets import load_breast_cancer #사이킷런 데이터셋 중
유방암데이터가져오기.
X,y = mglearn.datasets.load_extended_boston()
X_train,X_test,y_train,y_test = train_test_split(X,y,random_state=0)
lr = Ridge().fit(X_train,y_train)
print("train_score : {:.2f}".format(lr.score(X_train,y_train)))
print("test_score : {:.2f}".format(lr.score(X_test,y_test)))
```

lr = Ridge(alpha=2.5).fit(X_train,y_train)
Alpha를 높일수록 계수를 0에 더 가깝게 만들어주어
특성의 영향력을 없애 성능은 떨어지지만, 일반화에는 도움을 준다.

회귀 : Ridge선형회귀(Ridge Linear_Regression) alpha

```
import numpy as np
from IPython.display import display
from sklearn.neighbors import KNeighborsRegressor
from sklearn.linear_model import LinearRegression
from sklearn.model_selection import train_test_split
from sklearn.linear_model import Ridge
import matplotlib.pyplot as plt
import pandas as pd

import mglearn # mglearn은 저자가 만든 라이브러리로 임의의 데이터셋 만들기가 대부분.
from sklearn.datasets import load_breast_cancer #사이킷런 데이터셋 중 유방암데이터가져오기.
X,y = mglearn.datasets.load_extended_boston()
X_train,X_test,y_train,y_test = train_test_split(X,y,random_state=0)
lr = Ridge(alpha=2.5).fit(X_train,y_train)
lr1 = Ridge(alpha=10).fit(X_train,y_train)
plt.plot(lr.coef_,"o",color="limegreen",label="alpha=2.5")
plt.plot(lr1.coef_,"^",color="blue",label="alpha=10.0")
xlims=plt.xlim() #lr.coef_ and lr1.coef_의 x_min 과 x_max값을 리스트로 반환한다.
Plt.xlim(xlims[0],xlims[1]) #반환받은 x_min과 x_max를 기준으로 limit을 설정한다.
Plt.hlines(0,xlims[0],xlims[1]) # 수평선을 그린다. 첫번째 원소는 y좌표 나머지는 최소 최대(x좌표)
plt.ylim(-25,25)
plt.legend()
```



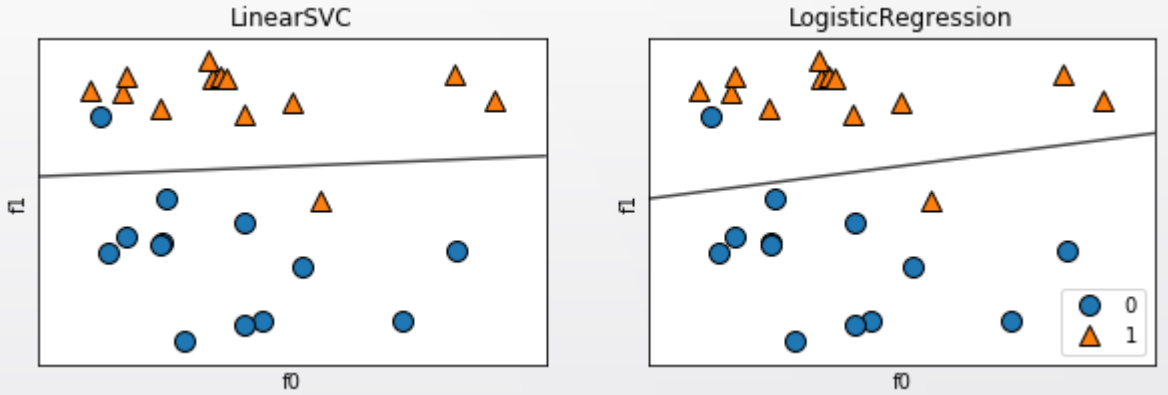
회귀 : Lasso선형회귀(Lasso Linear_Regression)

```
import numpy as np
from IPython.display import display
from sklearn.neighbors import KNeighborsRegressor
from sklearn.linear_model import LinearRegression
from sklearn.model_selection import train_test_split
from sklearn.linear_model import Lasso
import matplotlib.pyplot as plt
import pandas as pd

import mglearn # mglearn은 저자가 만든 라이브러리로 임의의 데이터셋 만들기가 대부분.
from sklearn.datasets import load_breast_cancer #사이킷런 데이터셋 중 유방암데이터가져오기.
X,y = mglearn.datasets.load_extended_boston()
X_train,X_test,y_train,y_test = train_test_split(X,y,random_state=0)
lr = Lasso(alpha=0.01,max_iter=1000000).fit(X_train,y_train)

print("train_ac : {:.2f} test_ac : {:.2f} using_fea_num : {}".format(lr.score(X_train,y_train),lr.score(X_test,y_test),np.sum(lr.coef_!=0)))
```

선형분류모델 ->y=ax+b가 결정경계이다!

선형회귀에서 y=ax+b -> y^이 예측값이였다면, 분류에서는 분류를 결정
짓는 경계이다!

```
import numpy as np
from IPython.display import display
from sklearn.neighbors import KNeighborsRegressor
from sklearn.linear_model import LinearRegression
from sklearn.linear_model import LogisticRegression
from sklearn.svm import LinearSVC
from sklearn.model_selection import train_test_split
from sklearn.linear_model import Lasso
import matplotlib.pyplot as plt
import pandas as pd

import mglearn # mglearn은 저자가 만든 라이브러리로 임의의 데이터셋 만들기가 대부분.
from sklearn.datasets import load_breast_cancer #사이킷런 데이터셋 중 유방암데이터가져오기.
X,y = mglearn.datasets.make_forge()
fig, axes = plt.subplots(1,2,figsize=(10,3))

for model, ax in zip([LinearSVC(), LogisticRegression()],axes):
    clf = model.fit(X,y)
    mglearn.discrete_scatter(X[:,0],X[:,1],y,ax=ax)
    mglearn.plots.plot_2d_separator(clf,X,fill=False,eps=0.5,ax=ax,alpha=.7)
    ax.set_title(clf.__class__.__name__)
    ax.set_xlabel("f0")
    ax.set_ylabel("f1")

axes[1].legend(loc=4)
```



선형분류모델의 하이퍼파라미터(C)

선형분류모델인 Logistic_Regression과 SVC는 C라는 하이퍼 파라미터가 존재한다. 이를 작게할 경우, L2규제를 강화하여, 분류경계를 완만하게 해준다. -> 과소적합의 방향(테스트 성능과 train성능이 비슷함.)
크게할경우, L2규제를 풀어 개별의 특성에 집중하지만 이는, 과대적합을 일으킬 수 있다.(train성능은 높지만, test성능이 현저히낮음.)

```
import numpy as np
from IPython.display import display
from sklearn.neighbors import KNeighborsRegressor
from sklearn.linear_model import LinearRegression
from sklearn.linear_model import LogisticRegression
from sklearn.svm import LinearSVC
from sklearn.model_selection import train_test_split
from sklearn.linear_model import Lasso
import matplotlib.pyplot as plt
import pandas as pd
```

```
C: 0.1 Train_ac : 0.951 Test_ac : 0.930 C: 100.0 Train_ac : 0.979 Test_ac : 0.944
```

```
import mglearn # mglearn은 저자가 만든 라이브러리로 임의의 데이터셋 만들기가 대부분.
from sklearn.datasets import load_breast_cancer #사이킷런 데이터셋 중 유방암데이터가져오기.
cancer= load_breast_cancer()
X_train,X_test, y_train, y_test = train_test_split(cancer.data,cancer.target,stratify = cancer.target , random_state =0 )
LogReg = LogisticRegression(C=0.1)
LogReg1 = LogisticRegression(C=100)
LogReg.fit(X_train,y_train)
LogReg1.fit(X_train,y_train)
print("C: 0.1 Train_ac : {:.3f} Test_ac : {:.3f}".format(LogReg.score(X_train,y_train),LogReg.score(X_test,y_test)))
print("C: 100.0 Train_ac : {:.3f} Test_ac : {:.3f}".format(LogReg1.score(X_train,y_train),LogReg1.score(X_test,y_test)) )
```

선형분류모델의 하이퍼파라미터(C) 계수 그래프

LogReg.coef_ # 각 특성별 기울기 값이 1행으로만 구성됨. => 전치행렬이 필요함.(벡터형태)

```
array([[ 0.56547593, 0.12193965, 0.36257521, -0.01571042, -
0.02188321, -0.09850069, -0.13771284, -0.05906787, -
0.02901748, -0.00670856, 0.01093659, 0.1768017 , 0.09896998,
-0.06547616, -0.00206195, -0.0214876 , -0.0306396 , -
0.00765427, -0.00672815, -0.00217135, 0.5152188 , -
0.24796213, -0.20907631, -0.01423568, -0.03852957, -
0.30113253, -0.36705277, -0.11598162, -0.08565021, -
0.02966569]])
```

(1, 30)

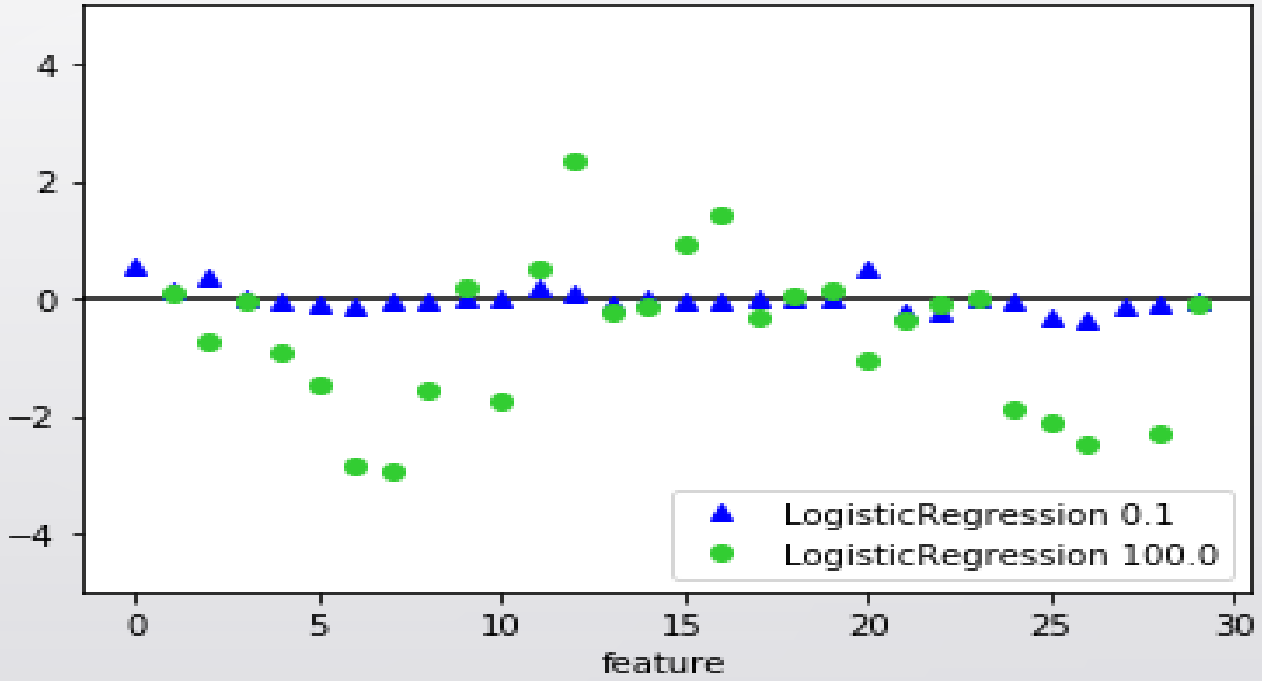
LogReg.coef_.T

```
array([[ 0.56547593], [ 0.12193965], [ 0.36257521],
[-0.01571042], [-0.02188321], [-0.09850069], [-
0.13771284], [-0.05906787], [-0.02901748], [-
0.00670856], [ 0.01093659], [ 0.1768017 ],
[ 0.09896998], [-0.06547616], [-0.00206195], [-
0.0214876 ], [-0.0306396 ], [-0.00765427], [-
0.00672815], [-0.00217135], [ 0.5152188 ], [-
0.24796213], [-0.20907631], [-0.01423568], [-
0.03852957], [-0.30113253], [-0.36705277], [-
0.11598162], [-0.08565021], [-0.02966569]])
```

(30, 1)

선형분류모델의 하이퍼파라미터(C) 계수 그래프

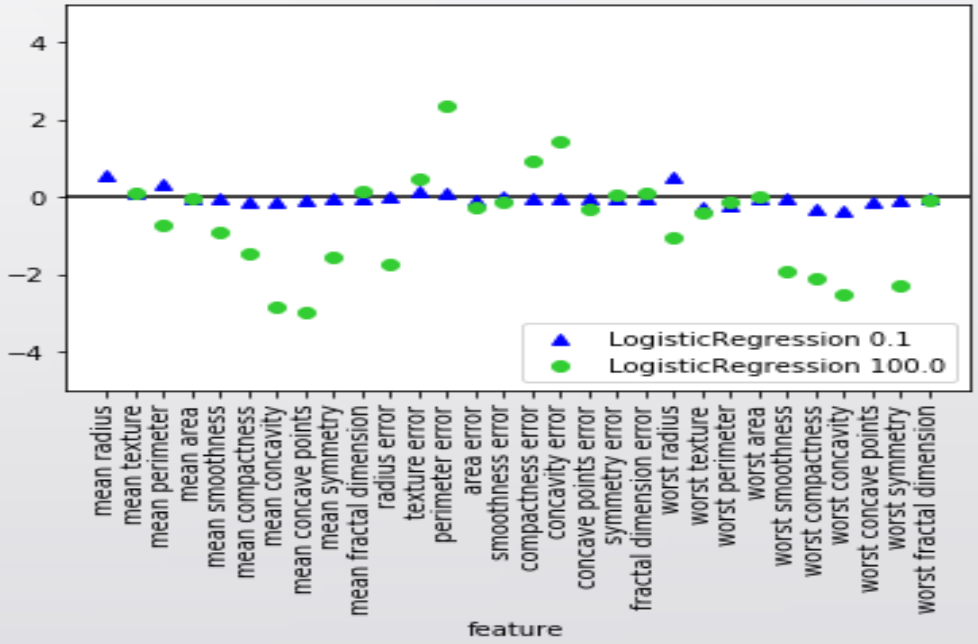
```
plt.plot(LogReg.coef_.T,"^",color="blue",label=LogRe
g.__class__.__name__ +" 0.1")
plt.plot(LogReg1.coef_.T,"o",color="limegreen",label=
LogReg.__class__.__name__ +" 100.0")
xlims=plt.xlim()
plt.xlim(xlims[0],xlims[1])

plt.hlines(0,xlims[0],xlims[1])
plt.ylim(-5,5)
plt.xlabel("feature")
plt.legend(loc=4)
```



=> X축은 feature에 해당한다. 즉 수치값을 feature_name으로 매핑시켜 가시성을 높여주자.

선형분류모델의 하이퍼파라미터(C) 계수 그래프

```
plt.plot(LogReg.coef_.T,"^",color="blue",label=LogReg.__class__.__name__ +" 0.1")
plt.plot(LogReg1.coef_.T,"o",color="limegreen",label=LogReg.__class__.__name__ +" 100.0")
plt.xticks(range(cancer.data.shape[1]),cancer.feature_names,rotation=90) #특징수는 30개이므로, data는 열개수가 특징의 개수.
#이름은 cancer.feature_names에 저장되어있고, 회전은 90도로 해준다.
xlims=plt.xlim()
plt.xlim(xlims[0],xlims[1])

plt.hlines(0,xlims[0],xlims[1])
plt.ylim(-5,5)
plt.xlabel("feature")
plt.legend(loc=4)
```

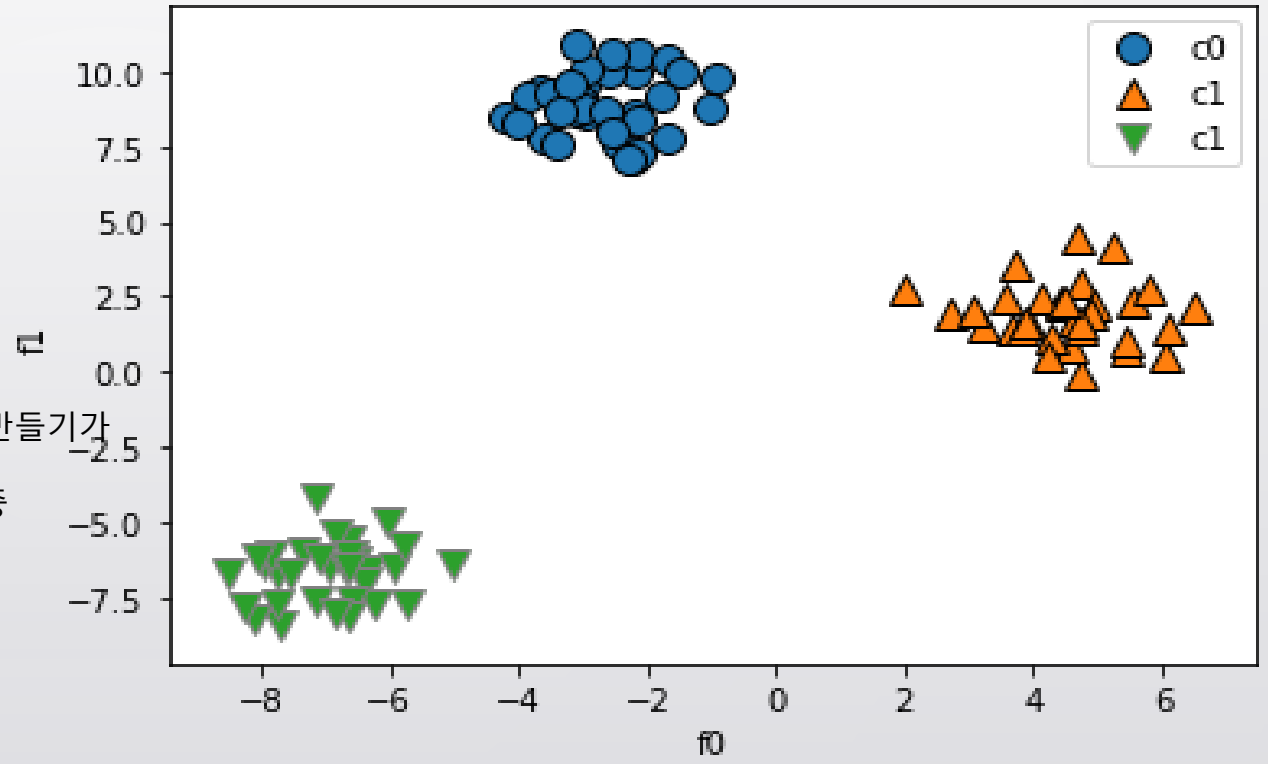


다중클래스 분류용 선형 모델

로지스틱 회귀를 제외한 많은 선형분류모델은 다중분류를 지원하지 않는다.(2진분류만을 지원)
그래서 1:N형태로 사용함. 이진분류기를.

즉, 각클래스별로 경계가 존재하고, 예측시에는 각 클래스별 분류모델중에 가장 가까운 MSE를 채택한다.

```
import numpy as np
from IPython.display import display
from sklearn.neighbors import KNeighborsRegressor
from sklearn.linear_model import LinearRegression
from sklearn.linear_model import LogisticRegression
from sklearn.svm import LinearSVC
from sklearn.model_selection import train_test_split
from sklearn.linear_model import Lasso
import matplotlib.pyplot as plt
import pandas as pd

import mglearn # mglearn은 저자가 만든 라이브러리로 임의의 데이터셋 만들기가 대부분.
from sklearn.datasets import load_breast_cancer #사이킷런 데이터셋 중 유방암데이터가져오기.
from sklearn.datasets import make_blobs
X,y = make_blobs(random_state = 42)
mglearn.discrete_scatter(X[:,0],X[:,1],y)
plt.xlabel("f0")
plt.ylabel("f1")
plt.legend(["c0","c1","c1"])
```



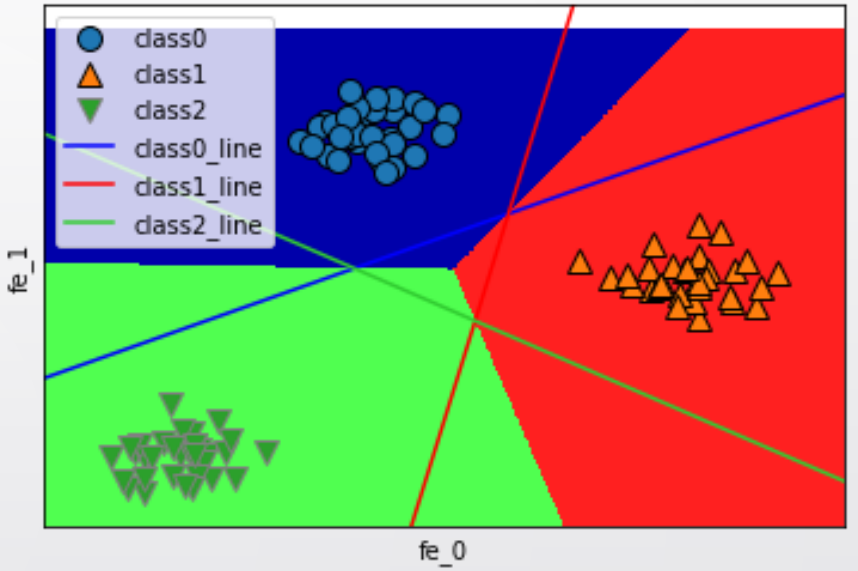
다중클래스 분류용 선형 모델

```
import numpy as np
from IPython.display import display
from sklearn.neighbors import KNeighborsRegressor
from sklearn.linear_model import LinearRegression
from sklearn.linear_model import LogisticRegression
from sklearn.svm import LinearSVC
from sklearn.model_selection import train_test_split
from sklearn.linear_model import Lasso
import matplotlib.pyplot as plt
import pandas as pd

import mglearn # mglearn은 저자가 만든 라이브러리로 임의의 데이터셋 만들기가
대부분.
from sklearn.datasets import load_breast_cancer #사이킷런 데이터셋 중
유방암데이터가져오기.
from sklearn.datasets import make_blobs
X,y = make_blobs(random_state = 42)


svm_lin = LinearSVC()
svm_lin.fit(X,y)
print(svm_lin.coef_.shape)
print(svm_lin.intercept_.shape)
```

```
(3, 2) (3,)
```

다중클래스 분류 예시.

```
import numpy as np
from IPython.display import display
from sklearn.neighbors import KNeighborsRegressor
from sklearn.linear_model import LinearRegression
from sklearn.linear_model import LogisticRegression
from sklearn.svm import LinearSVC
from sklearn.model_selection import train_test_split
from sklearn.linear_model import Lasso
import matplotlib.pyplot as plt
import pandas as pd

import mglearn # mglearn은 저자가 만든 라이브러리로 임의의 데이터셋 만들기가 대부분.
from sklearn.datasets import load_breast_cancer #사이킷런 데이터셋 중 유방암데이터가져오기.
from sklearn.datasets import make_blobs
X,y = make_blobs(random_state = 42)
```




```
svm_lin = LinearSVC()
svm_lin.fit(X,y)
mglearn.plots.plot_2d_classification(svm_lin,X,fill=True,eps=0.5,alpha=.7) #경계에 해당하는 배경색 칠해주기.
mglearn.discrete_scatter(X[:,0],X[:,1],y) # 각특성별로 이데이터셋에선 2개 / 산점도 행렬 그려주기.
line=np.linspace(-15,15) #범위
for co, inter, color in zip(svm_lin.coef_,svm_lin.intercept_,["blue","red","limegreen"]):
    plt.plot(line,-(line * co[0] + inter)/co[1],c=color) #각 특성별 기울기에 해당하는 직선 그려주기인데, 공식은 이해가 안됨.
    # -(line * class(n)특성1기울기 + class(n)절편)/class(n)특성2기울기
plt.ylim(-10,15)
plt.xlim(-10,8)
plt.xlabel("fe_0")
plt.ylabel("fe_1")
plt.legend(["class0", "class1", "class2", "class0_line", "class1_line", "class2_line"],loc=2)
```

선형회귀 / 선형 분류(다중까지) 정리.

선형회귀 -> LinearRegression() / Lasso(L1) / Ridge(L2) -> alpha => Hyperparameter

선형분류 -> SVC , LogisticRegression -> L2 -> C -> Hyperparameter

Alpha => 클수록 규제강화 , C=> 낮출수록 규제 강화.

(규제강화 = 모델 단순해짐을 의미함. -> 과소적합에 가까워진다.)

L1 규제는 몇몇 특성을 아예 0으로 만들어버리므로, 모델 해석이 중요할 때 사용하도록 한다.

선형 모델은 샘플에 비해 특성이 많을 때 잘 작동한다. 하지만 저차원 데이터셋에서는 일반화(과소적합)가 쉽게 되기때문에

실패할 수 도 있다.

추가정보 :Lr= LogisticRegression().fit(X_train,y_train)
Lr.fit(X_train,y_train).predict(X_test)로 쓰는 것도 가능하지만, 코드의 가독성 측면에서 보기 좋지않다.

나이브 베이즈 분류기

선형 모델과 매우 유사함.

훈련속도는 빠르지만, 일반화성능이 떨어짐.

효과적인 이유는 각 특성을 개별로 취급하여, 파라미터를 학습하고,(기울기) 각 특성에서 클래스별 통계를 단순하게 취합.

GaussianNB , BernoulliNB , MultinomialNB

Gaussian은 연속적인 어떤 데이터에도 적용할 수 있고, (고차원 데이터셋)
Bernoulli는 이진데이터를, Multinomial는 카운트 데이터 (ex 문장에 단어의 횟수) -> 두모델은 보통 텍스트데이터를 분류할 때 사용됨.

```
X = np.array([[0,1,0,1],
              [1,0,1,1],
              [0,0,0,1],
              [1,0,1,0],])
y=np.array([0,1,0,1])

counts = {}
for label in np.unique(y) :
    counts[label]= X[y==label].sum(axis=0) #0은 행끼리 더함, 1은 열끼리 합함.
print(counts)
```

```
{0: array([0, 1, 0, 2]), 1: array([2, 0, 2, 1])}
```

```
{0: array([2, 1]), 1: array([3, 2])}
```

결정트리 (Decision Tree) -> 2진분류 특성두개.

Yes or No! 질문 이름은 test라하고 보통 수치적으로 “>” “<“을 이용한 조건이 대부분이다.

```
import numpy as np
from IPython.display import display
from sklearn.neighbors import KNeighborsRegressor
from sklearn.linear_model import LinearRegression
from sklearn.linear_model import LogisticRegression
from sklearn.svm import LinearSVC
from sklearn.model_selection import train_test_split
from sklearn.tree import DecisionTreeClassifier
from sklearn.linear_model import Lasso
import matplotlib.pyplot as plt
import pandas as pd

import mglearn # mglearn은 저자가 만든 라이브러리로 임의의 데이터셋 만들기가 대부분.
from sklearn.datasets import load_breast_cancer #사이킷런 데이터셋 중 유방암데이터가져오기.
from sklearn.datasets import make_blobs
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_ac : {:.3f} , test_ac {:.3f}".format(tree.score(X_train,y_train),tree.score(X_test,y_test)))
```

```
train_ac : 1.000 , test_ac 0.937
```

모델이 굉장히 과적합됨. Why? Train 정확도가 100 %즉 리프노드가 순수 데이터포인트를 가진
트리이기 때문임. -> sklearn에서는 사전 가지치기를 지원함. 사후가지치기 방법도 존재함.

결정트리 (Decision Tree)

Yes or No!

```
import numpy as np
from IPython.display import display
from sklearn.neighbors import KNeighborsRegressor
from sklearn.linear_model import LinearRegression
from sklearn.linear_model import LogisticRegression
from sklearn.svm import LinearSVC
from sklearn.model_selection import train_test_split
from sklearn.tree import DecisionTreeClassifier
from sklearn.linear_model import Lasso
import matplotlib.pyplot as plt
import pandas as pd

import mglearn # mglearn은 저자가 만든 라이브러리로 임의의 데이터셋 만들기가 대부분.
from sklearn.datasets import load_breast_cancer #사이킷런 데이터셋 중 유방암데이터가져오기.
from sklearn.datasets import make_blobs
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_ac : {:.3f} , test_ac {:.3f}".format(tree.score(X_train,y_train),tree.score(X_test,y_test)))
```

```
train_ac : 1.000 , test_ac 0.937
```

모델이 굉장히 과적합됨. Why? Train 정확도가 100 %즉 리프노드가 순수 데이터포인트를 가진
트리이기 때문임. -> sklearn에서는 사전 가지치기를 지원함. 사후가지치기 방법도 존재함.

```
tree = DecisionTreeClassifier(max_depth=4,random_state =0)
```

트리의 높이를 제한함으로써, 과대적합을 막는다. 하지만 훈련세트의 정확성은 떨어질 수 밖에없다.

```
train_ac : 0.988 , test_ac 0.951
```

결정트리 (Decision Tree) 분석

Graphviz 난 왜 안되지 □ 알려주삼

```
import numpy as np
from IPython.display import display
from sklearn.neighbors import KNeighborsRegressor
from sklearn.linear_model import LinearRegression
from sklearn.linear_model import LogisticRegression
from sklearn.svm import LinearSVC
from sklearn.model_selection import train_test_split
from sklearn.tree import DecisionTreeClassifier
from sklearn.linear_model import Lasso
from sklearn.tree import export_graphviz
import matplotlib.pyplot as plt
import pandas as pd
```

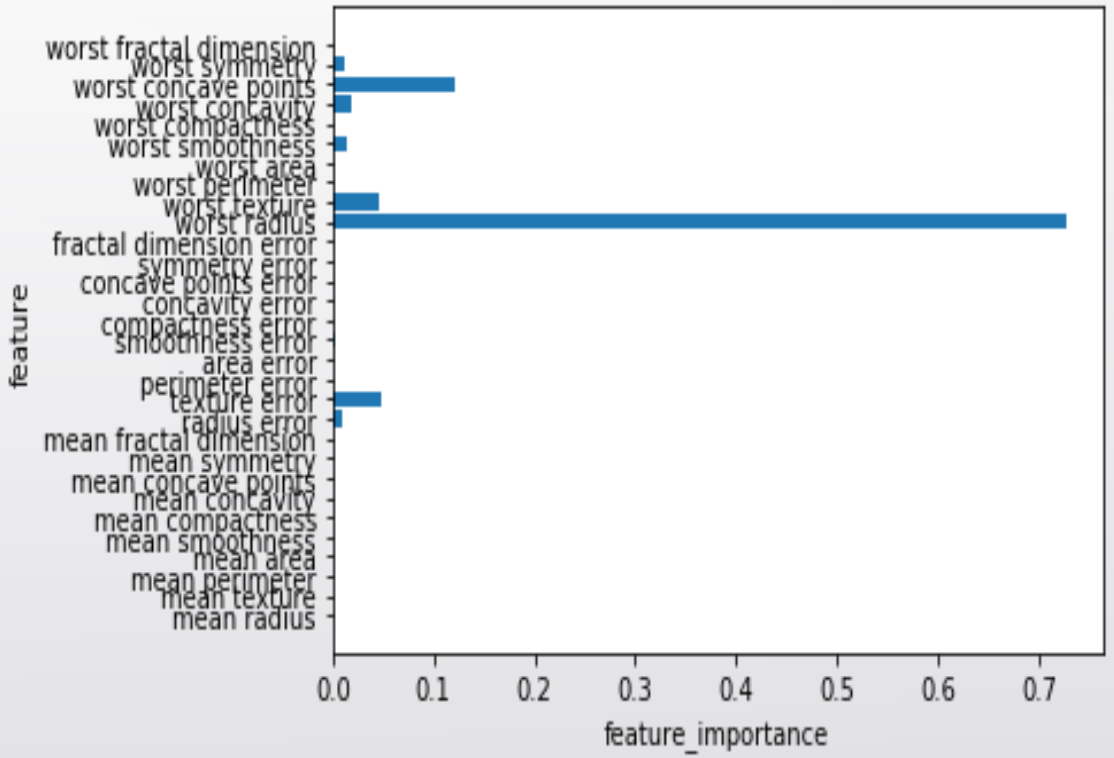
```
feature_importance [0. 0. 0. 0. 0. 0. 0. 0. 0. 0.01019737 0.04839825
0. 0. 0.0024156 0. 0. 0. 0. 0. 0.72682851 0.0458159 0. 0. 0.0141577 0.
0.018188 0.1221132 0.01188548 0. ]
```

```
import mglearn # mglearn은 저자가 만든 라이브러리로 임의의 데이터셋 만들기가 대부분.
from sklearn.datasets import load_breast_cancer #사이킷런 데이터셋 중 유방암데이터가져오기.
from sklearn.datasets import make_blobs
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(max_depth=4,random_state =0)
tree.fit(X_train, y_train)
print("feature_importance",tree.feature_importances_)
```

결정트리 (Decision Tree) 그래프

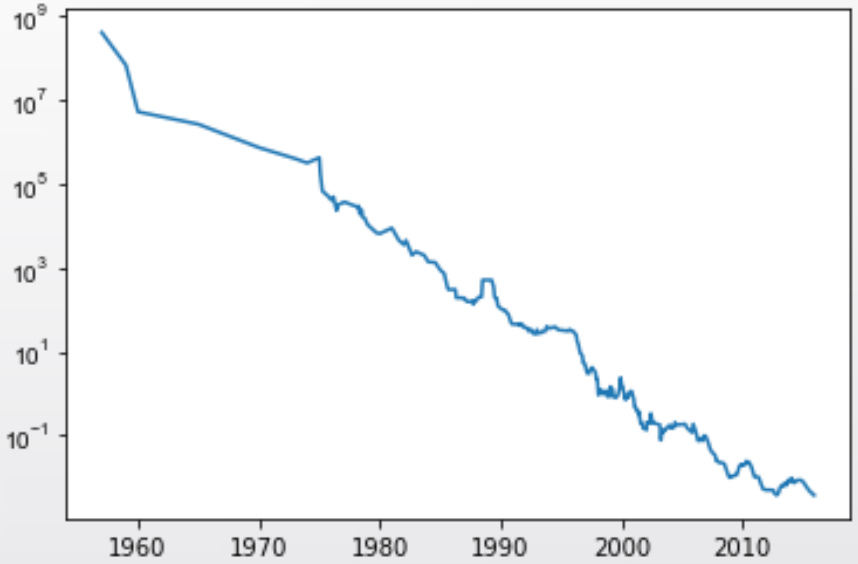
```
import numpy as np
from IPython.display import display
from sklearn.neighbors import KNeighborsRegressor
from sklearn.linear_model import LinearRegression
from sklearn.linear_model import LogisticRegression
from sklearn.svm import LinearSVC
from sklearn.model_selection import train_test_split
from sklearn.tree import DecisionTreeClassifier
from sklearn.linear_model import Lasso
from sklearn.tree import export_graphviz
import matplotlib.pyplot as plt
import pandas as pd
```

```
import mglearn # mglearn은 저자가 만든 라이브러리로 임의의 데이터셋 만들기가 대부분.
from sklearn.datasets import load_breast_cancer #사이킷런 데이터셋 중 유방암데이터가져오기.
from sklearn.datasets import make_blobs
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(max_depth=4,random_state =0)
tree.fit(X_train, y_train)

def plot_feature_importances_cancer(model):
    n_features = cancer.data.shape[1] #특징개수를 뜻함.
    plt.barh(np.arange(n_features),model.feature_importances_,align='center')
    plt.yticks(np.arange(n_features),cancer.feature_names)
    plt.xlabel("feature_importance")
    plt.ylabel("feature")
plot_feature_importances_cancer(tree)
```



결정트리 (Decision Tree) ex 램가격동향

```
import numpy as np
from IPython.display import display
from sklearn.neighbors import KNeighborsRegressor
from sklearn.linear_model import LinearRegression
from sklearn.linear_model import LogisticRegression
from sklearn.svm import LinearSVC
from sklearn.model_selection import train_test_split
from sklearn.tree import DecisionTreeClassifier
from sklearn.linear_model import Lasso
from sklearn.tree import export_graphviz
import matplotlib.pyplot as plt
import pandas as pd
import os
```

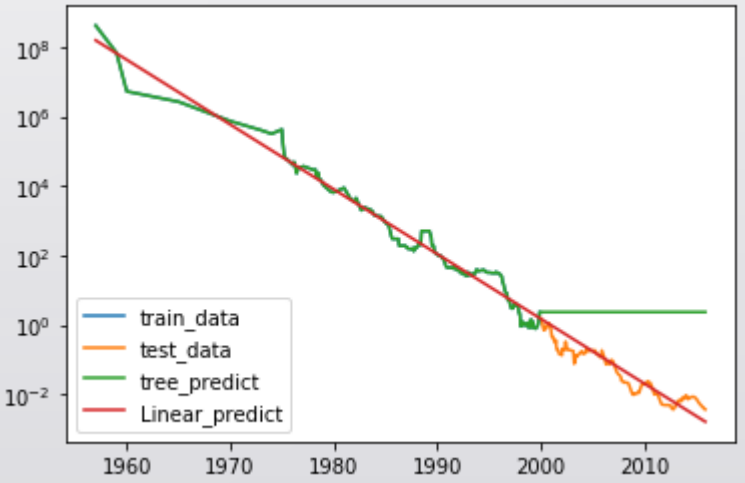



```
import mglearn # mglearn은 저자가 만든 라이브러리로 임의의 데이터셋 만들기가 대부분.
from sklearn.datasets import load_breast_cancer #사이킷런 데이터셋 중 유방암데이터가져오기.
from sklearn.datasets import make_blobs

ram_prices = pd.read_csv(os.path.join(mglearn.datasets.DATA_PATH,"ram_price.csv"))

plt.yticks(fontname = "Arial")
plt.semilogy(ram_prices.date,ram_prices.price)
```

램가격 동향 분석 (Linear_Regression / Tree_Decision compare)

```
import numpy as np
from IPython.display import display
from sklearn.neighbors import KNeighborsRegressor
from sklearn.linear_model import LinearRegression
from sklearn.linear_model import LogisticRegression
from sklearn.svm import LinearSVC
from sklearn.model_selection import train_test_split
from sklearn.tree import DecisionTreeClassifier
from sklearn.linear_model import Lasso
from sklearn.tree import export_graphviz
from sklearn.tree import DecisionTreeRegressor
import matplotlib.pyplot as plt
import pandas as pd
import os

import mglearn # mglearn은 저자가 만든 라이브러리로 임의의 데이터셋 만들기가 대부분.
from sklearn.datasets import load_breast_cancer #사이킷런 데이터셋 중
유방암데이터가져오기.
from sklearn.datasets import make_blobs

ram_prices = pd.read_csv(os.path.join(mglearn.datasets.DATA_PATH,"ram_price.csv"))
data_train = ram_prices[ram_prices.date < 2000]
data_test = ram_prices[ram_prices.date>=2000]

X_train =data_train.date[:,np.newaxis]

y_train = np.log(data_train.price)

tree = DecisionTreeRegressor().fit(X_train,y_train)
Linear_reg=LinearRegression().fit(X_train,y_train)
```

```
X_all = ram_prices.date[:,np.newaxis]

pred_tree = tree.predict(X_all)
pred_lr = Linear_reg.predict(X_all)

price_tree = np.exp(pred_tree)
price_lr = np.exp(pred_lr)

plt.semilogy(data_train.date,data_train.price, label="train_data")
plt.semilogy(data_test.date,data_test.price, label="test_data")

plt.semilogy(ram_prices.date,price_tree,label="tree_predict")
plt.semilogy(ram_prices.date,price_lr,label="Linear_predict")
plt.legend(loc=3)
```



Decision_Tree 같은 경우 복잡도 제어를 아무것도 하지 않았기 때문에, 리프노드에는 순수원소만 존재하기에, 훈련성능은 백프로가 나오고, 테스트 성능은 확연히 떨어짐.

외전 : Pandas 문법.

https://rfriend.tistory.com/250

Csv파일을 읽어올 때 사용하는 함수는 pandas.read_csv()이다. 파라미터로는 '경로', sep=' '이다.

위 예시에서는 아래와 같이 사용했는데, os.path.join은 파라미터로 받아온 경로를 이어주는 기능을 하는 메소드이다.

```
ram_prices = pd.read_csv(os.path.join(mglearn.datasets.DATA_PATH,"ram_price.csv")
```

```
print(mglearn.datasets.DATA_PATH)
print(os.path.join(mglearn.datasets.DATA_PATH,"ram_price.csv"))
```

```
C:\ProgramData\Anaconda3\lib\site-
packages\mglearn\data
C:\ProgramData\Anaconda3\lib\site-
packages\mglearn\data\ram_price.csv
```

결정트리의 장단점

사전 가지치기를 함으로써 과대적합을 막아줄 수 있지만, 충분치 않다.

장점으로는 시각화에 유용하며, 데이터 스케일에 영향을 받지 않고, 정규화나 표준화 같은 전처리 과정이 필요없다. (뒤에 뜻?)

결정트리의 앙상블(ensemble)

앙상블은 여러 머신러닝 모델을 연결하여 더 강력한 모델을 만드는 기법이다.

-> 랜덤 포레스트 / 그레이디언트 부스팅 결정트리는 모델을 구성하는 기본요소로 결정트리를 사용한다.

Random_Forest 기법

서로다른방향으로 과대적합된 트리를 많이 만든후 그 결과를 평균을 냄으로써, 과대적합양을 줄인다 !

랜덤 포레스트 구축 :

트리의 개수를 정해야한다. -> RandomForestRegressor / RandomForestClassifier의 n_estimators 매개변수 트리의 개수를 조정가능.

데이터의 부트스트랩 샘플을 생성한다. -> n개 샘플에서 n개 만큼을 무작위 추출(독립 시행으로. -> 중복허용)

또 특성 또한 무작위로 추출하여 테스트를 결정. (최선으로) -> 매개변수 max_features로 개수 조정가능.

각 노드는 다른 특성을 이용하여 테스트를 생성함.

Max_features = 전체 특성개수와 같으면, 의미가 없어짐. 모든 노드들이 각기다른 특성을 가지고 test를 하기 때문에.

회귀는 각 트리의 예측값의 평균을 통해 최종 예측을 만들고, 분류는 각 트리의 가능성있는 출력 레이블의 확률을 평균내어 가장 높은 확률을 가진 클래스가 예측 레이블이 된다.

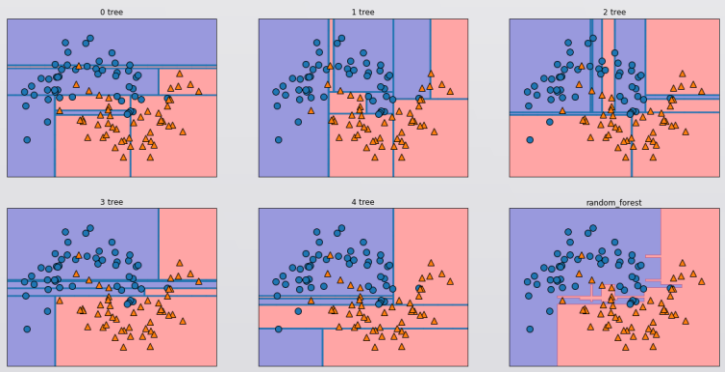
Random_Forest 기법

예제 보기전 파이썬 문법.

```
axes.ravel().shape # (2,3) -> (6,)

>>> np.arange(6).reshape(2,3)
array([[0, 1, 2],
       [3, 4, 5]])
>>> n_a=np.arange(6).reshape(2,3)
>>> n_a[-1,-1]
5
>>> n_a[-2,-2]
1
>>> n_a[-2,-3]
0
```

Random_Forest 기법 (2진 분류 2진특성 데이터 셋)

```
import numpy as np
from IPython.display import display
from sklearn.neighbors import KNeighborsRegressor
from sklearn.linear_model import LinearRegression
from sklearn.linear_model import LogisticRegression
from sklearn.svm import LinearSVC
from sklearn.model_selection import train_test_split
from sklearn.tree import DecisionTreeClassifier
from sklearn.linear_model import Lasso
from sklearn.tree import export_graphviz
from sklearn.tree import DecisionTreeRegressor
from sklearn.ensemble import RandomForestClassifier
from sklearn.datasets import make_moons
import matplotlib.pyplot as plt
import pandas as pd
import os
```

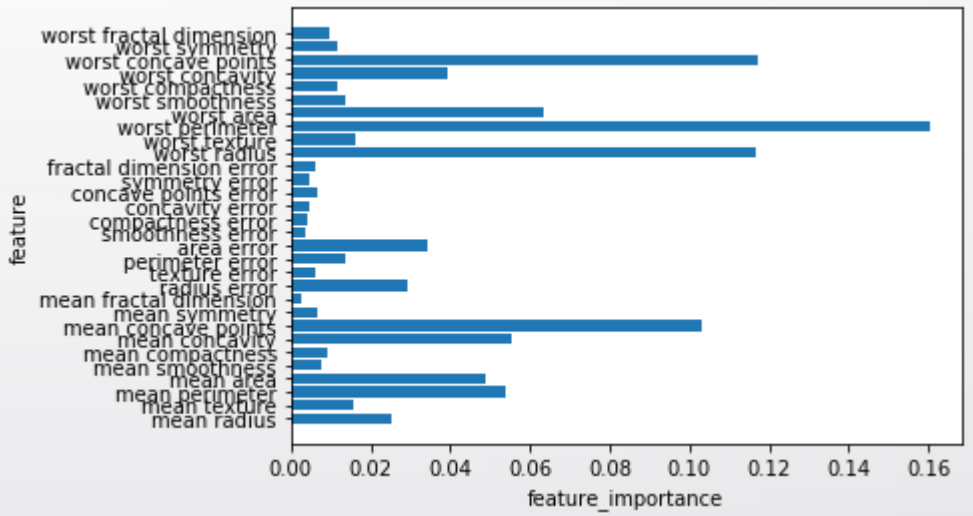



```
import mglearn # mglearn은 저자가 만든 라이브러리로 임의의 데이터셋 만들기가 대부분.
from sklearn.datasets import load_breast_cancer #사이킷런 데이터셋 중
유방암데이터가져오기.
from sklearn.datasets import make_blobs

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

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

Random_Forest 기법 (2진 분류 2진특성 데이터 셋)

```
import numpy as np
from IPython.display import display
from sklearn.neighbors import KNeighborsRegressor
from sklearn.linear_model import LinearRegression
from sklearn.linear_model import LogisticRegression
from sklearn.svm import LinearSVC
from sklearn.model_selection import train_test_split
from sklearn.tree import DecisionTreeClassifier
from sklearn.linear_model import Lasso
from sklearn.tree import export_graphviz
from sklearn.tree import DecisionTreeRegressor
from sklearn.ensemble import RandomForestClassifier
from sklearn.datasets import make_moons
import matplotlib.pyplot as plt
import pandas as pd
import os
```




```
train_ac : 1.000 / test_ac : 0.972
```

```
import mglearn # mglearn은 저자가 만든 라이브러리로 임의의 데이터셋 만들기가 대부분.
from sklearn.datasets import load_breast_cancer #사이킷런 데이터셋 중
유방암데이터가져오기.
from sklearn.datasets import make_blobs

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_ac : {:.3f} / test_ac :
{:.3f}".format(forest.score(X_train,y_train),forest.score(X_test,y_test)))
```

```
def plot_feature_importances_cancer(model):
    n_features = cancer.data.shape[1] #특징개수를 뜻함.
    plt.barh(np.arange(n_features),model.feature_importances_,align='center')
    plt.yticks(np.arange(n_features),cancer.feature_names)
    plt.xlabel("feature_importance")
    plt.ylabel("feature")
```

Gradient_Boosting -> 가중치를 수정 (Learning_rate)

무작위성은 없지만, 강력한 사전 가지치기가 이루어진다.

아이디어 : 얕은 트리들을 많이 연결한다. 각각 트리는 데이터 일부에 대해 잘 예측을 수행한다.

Learning_rate(학습률)이 성능에 영향을 준다. 다음 트리에 test의 임계치를 얼마나 강하게 바꿀것인지.

Gradient_Boosting -> 가중치를 수정 (Learning_rate)

```
import numpy as np
from IPython.display import display
from sklearn.neighbors import KNeighborsRegressor
from sklearn.linear_model import LinearRegression
from sklearn.linear_model import LogisticRegression
from sklearn.svm import LinearSVC
from sklearn.model_selection import train_test_split
from sklearn.tree import DecisionTreeClassifier
from sklearn.linear_model import Lasso
from sklearn.tree import export_graphviz
from sklearn.tree import DecisionTreeRegressor
from sklearn.ensemble import RandomForestClassifier
from sklearn.ensemble import GradientBoostingClassifier
from sklearn.datasets import make_moons
import matplotlib.pyplot as plt
import pandas as pd
import os
```

```
def plot_feature_importances_cancer(model):
    n_features = cancer.data.shape[1] #특징개수를 뜻함.
    plt.barh(np.arange(n_features),model.feature_importances_,align='center')
    plt.yticks(np.arange(n_features),cancer.feature_names)
    plt.xlabel("feature_importance")
    plt.ylabel("feature")

import mglearn # mglearn은 저자가 만든 라이브러리로 임의의 데이터셋 만들기가 대부분.
from sklearn.datasets import load_breast_cancer #사이킷런 데이터셋 중
유방암데이터가져오기.
from sklearn.datasets import make_blobs

cancer= load_breast_cancer()
X_train, X_test, y_train , y_test = train_test_split(cancer.data,cancer.target,
random_state = 0)
gbrt = GradientBoostingClassifier(random_state = 0)
gbrt.fit(X_train, y_train)

print("train_ac : {:.3f}, test_ac :
{:.3f}".format(gbrt.score(X_train,y_train),gbrt.score(X_test,y_test)))
```

```
train_ac : 1.000, test_ac : 0.965
```

과대적합이다. -> 사전가지치기를 해주자.

Gradient_Boosting ->사전가지치기 이용.

```
import numpy as np
from IPython.display import display
from sklearn.neighbors import KNeighborsRegressor
from sklearn.linear_model import LinearRegression
from sklearn.linear_model import LogisticRegression
from sklearn.svm import LinearSVC
from sklearn.model_selection import train_test_split
from sklearn.tree import DecisionTreeClassifier
from sklearn.linear_model import Lasso
from sklearn.tree import export_graphviz
from sklearn.tree import DecisionTreeRegressor
from sklearn.ensemble import RandomForestClassifier
from sklearn.ensemble import GradientBoostingClassifier
from sklearn.datasets import make_moons
import matplotlib.pyplot as plt
import pandas as pd
import os
```

```
def plot_feature_importances_cancer(model):
    n_features = cancer.data.shape[1] #특징개수를 뜻함.
    plt.barh(np.arange(n_features),model.feature_importances_,align='center')
    plt.yticks(np.arange(n_features),cancer.feature_names)
    plt.xlabel("feature_importance")
    plt.ylabel("feature")

import mglearn # mglearn은 저자가 만든 라이브러리로 임의의 데이터셋 만들기가 대부분.
from sklearn.datasets import load_breast_cancer #사이킷런 데이터셋 중
유방암데이터가져오기.
from sklearn.datasets import make_blobs

cancer= load_breast_cancer()
X_train, X_test, y_train , y_test = train_test_split(cancer.data,cancer.target,
random_state = 0)
gbrt = GradientBoostingClassifier(max_depth=1,random_state = 0)
gbrt.fit(X_train, y_train)

print("train_ac : {:.3f}, test_ac :
{:.3f}".format(gbrt.score(X_train,y_train),gbrt.score(X_test,y_test)))
```

```
train_ac : 0.991, test_ac : 0.972
```

테스트 정확도가 꽤 상승했다. 두번째 방법인 Learning_rate를 낮춰서 한번 보자.

Gradient_Boosting ->Learning_rate 낮추기

```
import numpy as np
from IPython.display import display
from sklearn.neighbors import KNeighborsRegressor
from sklearn.linear_model import LinearRegression
from sklearn.linear_model import LogisticRegression
from sklearn.svm import LinearSVC
from sklearn.model_selection import train_test_split
from sklearn.tree import DecisionTreeClassifier
from sklearn.linear_model import Lasso
from sklearn.tree import export_graphviz
from sklearn.tree import DecisionTreeRegressor
from sklearn.ensemble import RandomForestClassifier
from sklearn.ensemble import GradientBoostingClassifier
from sklearn.datasets import make_moons
import matplotlib.pyplot as plt
import pandas as pd
import os
```

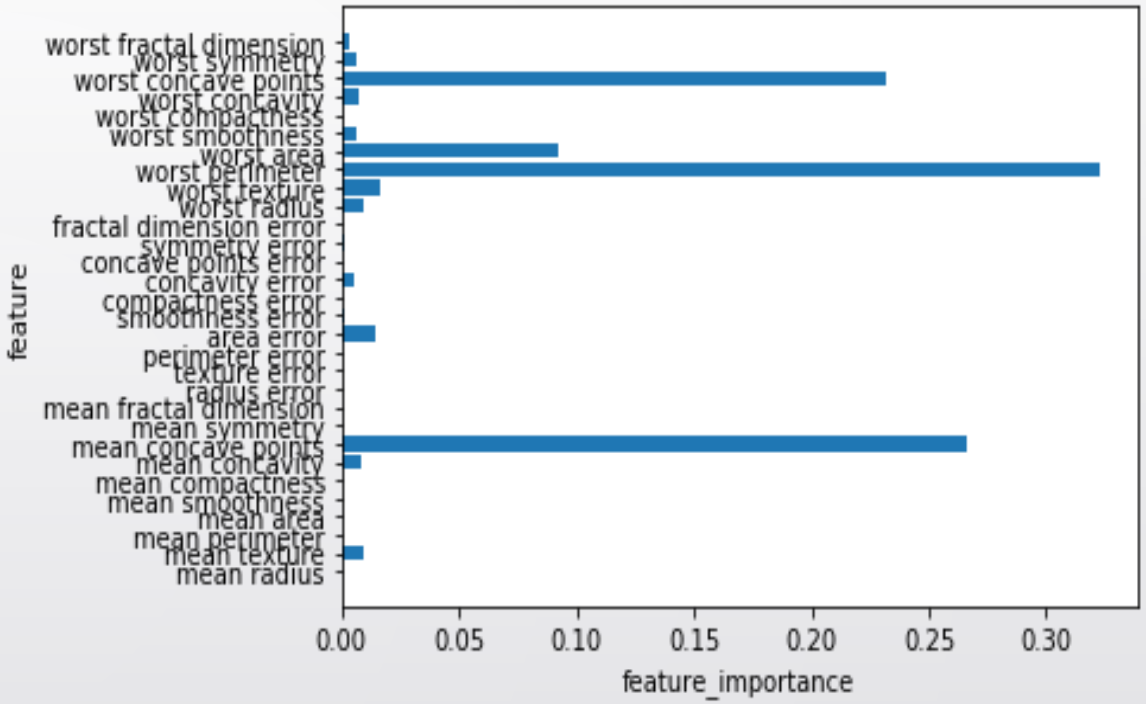
```
def plot_feature_importances_cancer(model):
    n_features = cancer.data.shape[1] #특징개수를 뜻함.
    plt.barh(np.arange(n_features),model.feature_importances_,align='center')
    plt.yticks(np.arange(n_features),cancer.feature_names)
    plt.xlabel("feature_importance")
    plt.ylabel("feature")

import mglearn # mglearn은 저자가 만든 라이브러리로 임의의 데이터셋 만들기가 대부분.
from sklearn.datasets import load_breast_cancer #사이킷런 데이터셋 중
유방암데이터가져오기.
from sklearn.datasets import make_blobs

cancer= load_breast_cancer()
X_train, X_test, y_train , y_test = train_test_split(cancer.data,cancer.target,
random_state = 0)
gbrt = GradientBoostingClassifier(learning_rate=0.01,random_state = 0)
gbrt.fit(X_train, y_train)

print("train_ac : {:.3f}, test_ac :
{:.3f}".format(gbrt.score(X_train,y_train),gbrt.score(X_test,y_test)))
```

```
train_ac : 0.988, test_ac : 0.965
```

위 유방암 데이터셋에서는 learning_rate를 낮추는 것 보단 사전 가지치기 방식이 더높은 테스트 정확도 상승률을 보였다.

Gradient_Boosting feature_importance

```
cancer= load_breast_cancer()
X_train, X_test, y_train , y_test =
train_test_split(cancer.data,cancer.target, random_state = 0)
gbrt = GradientBoostingClassifier(max_depth=1,random_state = 0)
#기본값 트리개수 100개
gbrt.fit(X_train, y_train)
plot_feature_importances_cancer(gbrt)
#print("train_ac : {:.3f}, test_ac :
{:.3f}".format(gbrt.score(X_train,y_train),gbrt.score(X_test,y_test)))
```



Gradient_Boosting기법은 지도학습에서 강력하고 널리 사용됨.
단점은 학습시간과 매개변수 조정.

매개변수 : n_estimators (트리의 개수)
랜덤포레스트와 달리 많을 수록 과적합이고,
Learning_rate(학습률)
학습률을 낮추면 그만큼 비슷한 복잡도의 트리가 많이 필요함.

즉 트리의 개수를 환경에 맞추어(Ram , Computing Env) 후
Learning_rate조정을 해야한다.

Bagging(Bootstrap aggregating)

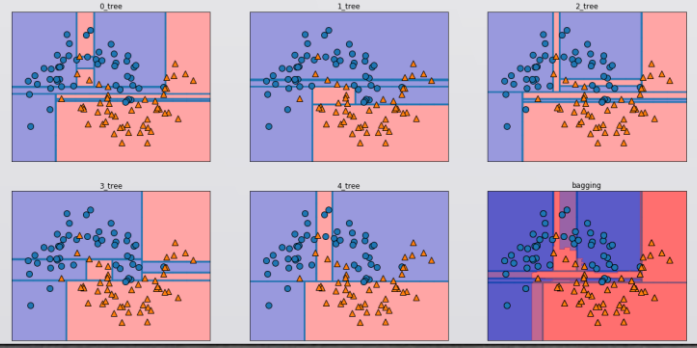
```
import numpy as np
from IPython.display import display
from sklearn.neighbors import KNeighborsRegressor
from sklearn.linear_model import LinearRegression
from sklearn.linear_model import LogisticRegression
from sklearn.svm import LinearSVC
from sklearn.model_selection import train_test_split
from sklearn.tree import DecisionTreeClassifier
from sklearn.linear_model import Lasso
from sklearn.tree import export_graphviz
from sklearn.tree import DecisionTreeRegressor
from sklearn.ensemble import BaggingClassifier
from sklearn.ensemble import RandomForestClassifier
from sklearn.ensemble import GradientBoostingClassifier
#from preamble import *
from sklearn.datasets import make_moons
import matplotlib.pyplot as plt

import pandas as pd
import os
```

```
def plot_feature_importances_cancer(model):
    n_features = cancer.data.shape[1] #특징개수를 뜻함.
    plt.barh(np.arange(n_features),model.feature_importances_,align='center')
    plt.yticks(np.arange(n_features),cancer.feature_names)
    plt.xlabel("feature_importance")
    plt.ylabel("feature")

import mglearn # mglearn은 저자가 만든 라이브러리로 임의의 데이터셋 만들기가 대부분.
from sklearn.datasets import load_breast_cancer #사이킷런 데이터셋 중 유방암데이터가져오기.
from sklearn.datasets import make_blobs

cancer= load_breast_cancer()
Xc_train, Xc_test, yc_train , yc_test = train_test_split(cancer.data,cancer.target, random_state = 0)
Xm, ym = make_moons(n_samples =100,noise=0.25,random_state=3)
Xm_train, Xm_test , ym_train, ym_test = train_test_split(Xm,ym,stratify=ym,random_state =0)

bagging = BaggingClassifier(LogisticRegression(),n_estimators =100, oob_score=True,n_jobs=1,random_state = 42)
#LogisticRegression으로 약한 분류기를 100개를 만들어서 각각 중복을 포함한 랜덤으로 특성으로 학습시키고 높은 빈도 클래스 레이블
예측값임.
bagging.fit(Xc_train, yc_train)

print("train_ac : {:.3f}".format(bagging.score(Xc_train,yc_train)))
print("test_ac : {:.3f}".format(bagging.score(Xc_test,yc_test)))
print("OOB_ac : {:.3f}".format(bagging.oob_score_))
```

```
train_ac : 0.962 test_ac : 0.958 OOB_ac : 0.948
```

Bagging(Bootstrap aggregating) -> tree 5개

```
import numpy as np
from IPython.display import display
from sklearn.neighbors import KNeighborsRegressor
from sklearn.linear_model import LinearRegression
from sklearn.linear_model import LogisticRegression
from sklearn.svm import LinearSVC
from sklearn.model_selection import train_test_split
from sklearn.tree import DecisionTreeClassifier
from sklearn.linear_model import Lasso
from sklearn.tree import export_graphviz
from sklearn.tree import DecisionTreeRegressor
from sklearn.ensemble import BaggingClassifier
from sklearn.ensemble import RandomForestClassifier
2]: from sklearn.ensemble import GradientBoostingClassifier
#from preamble import *
from sklearn.datasets import make_moons
import matplotlib.pyplot as plt

import pandas as pd
import os
```




```
def plot_feature_importances_cancer(model):
    n_features = cancer.data.shape[1] #특징개수를 뜻함.
    plt.barh(np.arange(n_features),model.feature_importances_,align='center')
    plt.yticks(np.arange(n_features),cancer.feature_names)
    plt.xlabel("feature_importance")
    plt.ylabel("feature")

import mglearn # mglearn은 저자가 만든 라이브러리로 임의의 데이터셋 만들기가 대부분.
from sklearn.datasets import load_breast_cancer #사이킷런 데이터셋 중 유방암데이터가져오기.
from sklearn.datasets import make_blobs

cancer= load_breast_cancer()
Xc_train, Xc_test, yc_train , yc_test = train_test_split(cancer.data,cancer.target, random_state = 0)
Xm, ym = make_moons(n_samples =100,noise=0.25,random_state=3)
Xm_train, Xm_test , ym_train, ym_test = train_test_split(Xm,ym,stratify=ym,random_state =0)

bagging = BaggingClassifier(DecisionTreeClassifier(),n_estimators =5, oob_score=True,n_jobs=-
1,random_state = 42)

bagging.fit(Xm_train, ym_train)

fig ,axes = plt.subplots(2,3,figsize = (20,10))
for i,(ax, tree) in enumerate(zip(axes.ravel(),bagging.estimators_)):
    ax.set_title("{}_tree".format(i))
    mglearn.plots.plot_tree_partition(Xm,ym,tree,ax=ax)
mglearn.plots.plot_2d_separator(bagging,Xm,fill=True,ax=axes[-1,-1],alpha=.4)
axes[-1,-1].set_title("bagging")
mglearn.discrete_scatter(Xm[:,0],Xm[:,1],ym)
```

Bagging(Bootstrap aggregating) -> tree 100개.

```
import numpy as np
from IPython.display import display
from sklearn.neighbors import KNeighborsRegressor
from sklearn.linear_model import LinearRegression
from sklearn.linear_model import LogisticRegression
from sklearn.svm import LinearSVC
from sklearn.model_selection import train_test_split
from sklearn.tree import DecisionTreeClassifier
from sklearn.linear_model import Lasso
from sklearn.tree import export_graphviz
from sklearn.tree import DecisionTreeRegressor
from sklearn.ensemble import BaggingClassifier
from sklearn.ensemble import RandomForestClassifier
2]: from sklearn.ensemble import GradientBoostingClassifier
#from preamble import *
from sklearn.datasets import make_moons
import matplotlib.pyplot as plt

import pandas as pd
import os
```

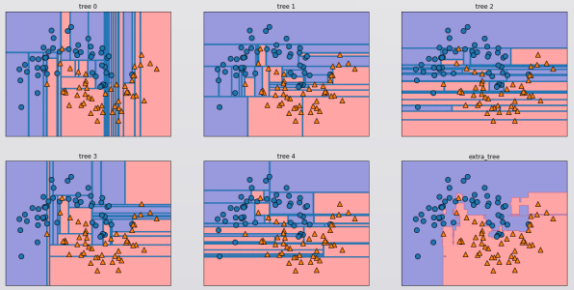
```
def plot_feature_importances_cancer(model):
    n_features = cancer.data.shape[1] #특징개수를 뜻함.
    plt.barh(np.arange(n_features),model.feature_importances_,align='center')
    plt.yticks(np.arange(n_features),cancer.feature_names)
    plt.xlabel("feature_importance")
    plt.ylabel("feature")

import mglearn # mglearn은 저자가 만든 라이브러리로 임의의 데이터셋 만들기가 대부분.
from sklearn.datasets import load_breast_cancer #사이킷런 데이터셋 중 유방암데이터가져오기.
from sklearn.datasets import make_blobs

cancer= load_breast_cancer()
Xc_train, Xc_test, yc_train , yc_test = train_test_split(cancer.data,cancer.target, random_state = 0)
Xm, ym = make_moons(n_samples =100,noise=0.25,random_state=3)
Xm_train, Xm_test , ym_train, ym_test = train_test_split(Xm,ym,stratify=ym,random_state =0)

bagging = BaggingClassifier(DecisionTreeClassifier(),n_estimators =100, oob_score=True,n_jobs=-
1,random_state = 42)

bagging.fit(Xc_train, yc_train)

print("train {:.3f}".format(bagging.score(Xc_train,yc_train)))
print("test {:.3f}".format(bagging.score(Xc_test,yc_test)))
print("oob {:.3f}".format(bagging.oob_score_))
```

```
train 1.000 test 0.965 oob 0.951
```

Extra_Trees(Random_forest와 달리 BootstrapSampling적용x)

랜덤포레스트방식과 다른 무작위성을 주입한다. (특성 무작위로 분할 후, 그중 가장 최적을 찾는다.

```
import numpy as np
from IPython.display import display
from sklearn.neighbors import KNeighborsRegressor
from sklearn.linear_model import LinearRegression
from sklearn.linear_model import LogisticRegression
from sklearn.svm import LinearSVC
from sklearn.model_selection import train_test_split
from sklearn.tree import DecisionTreeClassifier
from sklearn.linear_model import Lasso
from sklearn.tree import export_graphviz
from sklearn.tree import DecisionTreeRegressor
2]: from sklearn.ensemble import BaggingClassifier
from sklearn.ensemble import ExtraTreesClassifier
from sklearn.ensemble import RandomForestClassifier
from sklearn.ensemble import GradientBoostingClassifier
#from preamble import *
from sklearn.datasets import make_moons
import matplotlib.pyplot as plt

import pandas as pd
import os
```




```
def plot_feature_importances_cancer(model):
    n_features = cancer.data.shape[1] #특징개수를 뜻함.
    plt.barh(np.arange(n_features),model.feature_importances_,align='center')
    plt.yticks(np.arange(n_features),cancer.feature_names)
    plt.xlabel("feature_importance")
    plt.ylabel("feature")

import mglearn # mglearn은 저자가 만든 라이브러리로 임의의 데이터셋 만들기가 대부분.
from sklearn.datasets import load_breast_cancer #사이킷런 데이터셋 중
유방암데이터가져오기.
from sklearn.datasets import make_blobs

cancer= load_breast_cancer()
Xc_train, Xc_test, yc_train , yc_test = train_test_split(cancer.data,cancer.target,
random_state = 0)
Xm, ym = make_moons(n_samples =100,noise=0.25,random_state=3)
Xm_train, Xm_test , ym_train, ym_test = train_test_split(Xm,ym,stratify=ym,random_state =0)

xtree = ExtraTreesClassifier(n_estimators=5,n_jobs=-1,random_state=0)
xtree.fit(Xm_train,ym_train)

fig, axes = plt.subplots(2,3,figsize=(20,10))
for i , (ax , tree) in enumerate(zip(axes.ravel(),xtree.estimators_)):
    ax.set_title("tree {}".format(i))
    mglearn.plots.plot_tree_partition(Xm,ym,tree,ax=ax)
mglearn.plots.plot_2d_separator(xtree,Xm,fill=True,ax=axes[-1,-1],alpha=.4)
axes[-1,-1].set_title("extra_tree")
mglearn.discrete_scatter(Xm[:,0],Xm[:,1],ym)
```

Extra_Trees ->est 100, cancer_dataset (feature많음)

랜덤포레스트방식과 다른 무작위성을 주입한다. (특성 무작위로 분할 후, 그중 가장 최적을 찾는다.

```
import numpy as np
from IPython.display import display
from sklearn.neighbors import KNeighborsRegressor
from sklearn.linear_model import LinearRegression
from sklearn.linear_model import LogisticRegression
from sklearn.svm import LinearSVC
from sklearn.model_selection import train_test_split
from sklearn.tree import DecisionTreeClassifier
from sklearn.linear_model import Lasso
from sklearn.tree import export_graphviz
from sklearn.tree import DecisionTreeRegressor
from sklearn.ensemble import BaggingClassifier
from sklearn.ensemble import ExtraTreesClassifier
from sklearn.ensemble import RandomForestClassifier
from sklearn.ensemble import GradientBoostingClassifier
#from preamble import *
from sklearn.datasets import make_moons
import matplotlib.pyplot as plt

import pandas as pd
import os
```

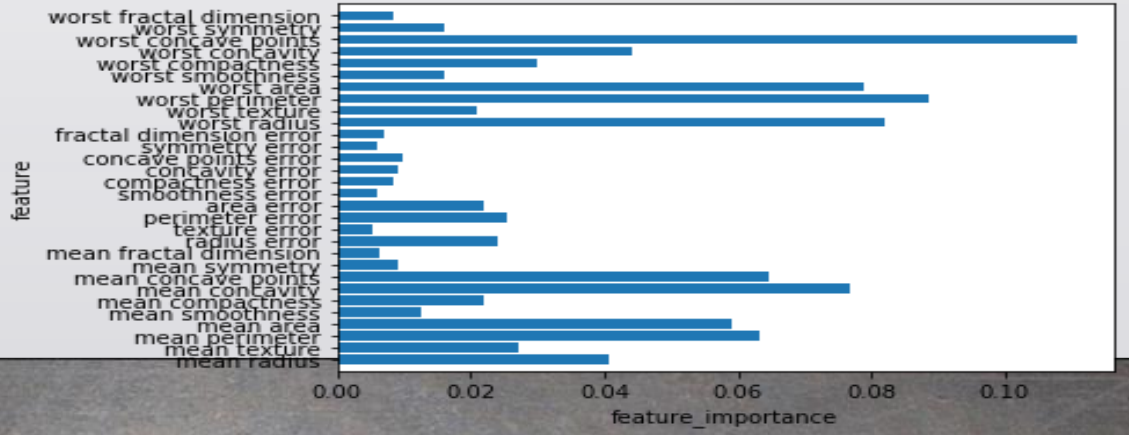
```
from sklearn.datasets import load_breast_cancer #사이킷런 데이터셋 중
유방암데이터가져오기.
from sklearn.datasets import make_blobs

cancer= load_breast_cancer()
Xc_train, Xc_test, yc_train , yc_test = train_test_split(cancer.data,cancer.target,
random_state = 0)
Xm, ym = make_moons(n_samples =100,noise=0.25,random_state=3)
Xm_train, Xm_test , ym_train, ym_test =
train_test_split(Xm,ym,stratify=ym,random_state =0)

xtree = ExtraTreesClassifier(n_estimators=100,n_jobs=-1,random_state=0)
xtree.fit(Xc_train,yc_train)

print("train_ac : {:.3f}".format(xtree.score(Xc_train,yc_train)))
print("test_ac : {:.3f}".format(xtree.score(Xc_test,yc_test)))
```

```
train_ac : 1.000 test_ac : 0.972
```

Extra_Trees ->est 100, cancer_dataset (feature많음)

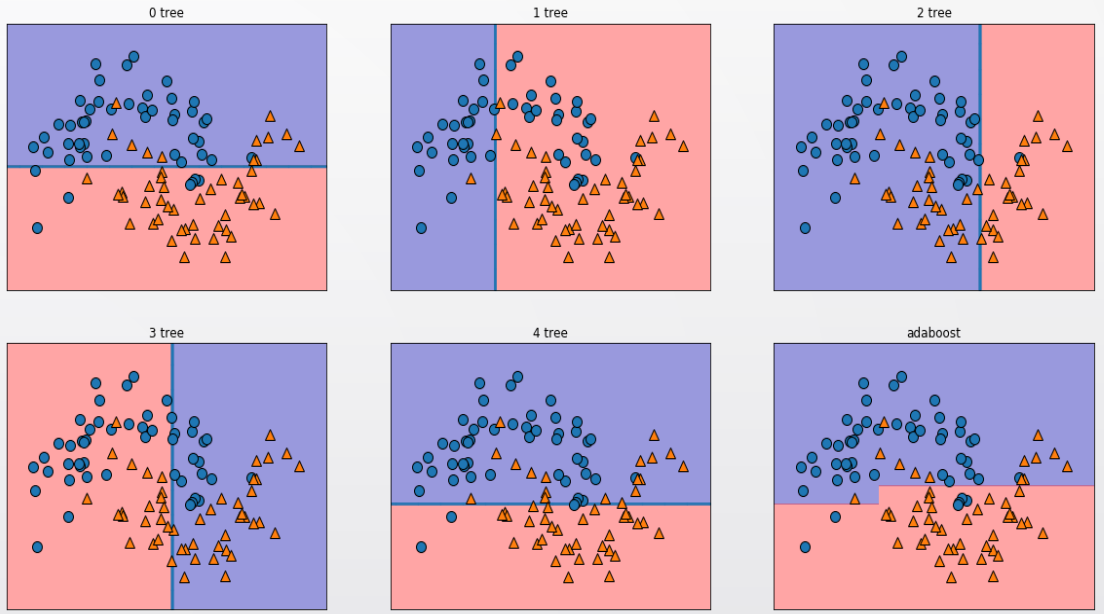
```
import numpy as np
from IPython.display import display
from sklearn.neighbors import KNeighborsRegressor
from sklearn.linear_model import LinearRegression
from sklearn.linear_model import LogisticRegression
from sklearn.svm import LinearSVC
from sklearn.model_selection import train_test_split
from sklearn.tree import DecisionTreeClassifier
from sklearn.linear_model import Lasso
from sklearn.tree import export_graphviz
from sklearn.tree import DecisionTreeRegressor
from sklearn.ensemble import BaggingClassifier
from sklearn.ensemble import ExtraTreesClassifier
from sklearn.ensemble import RandomForestClassifier
from sklearn.ensemble import GradientBoostingClassifier
#from preamble import *
from sklearn.datasets import make_moons
import matplotlib.pyplot as plt

import pandas as pd
import os


def plot_feature_importances_cancer(model):
    n_features = cancer.data.shape[1] #특징개수를 뜻함.
    plt.barh(np.arange(n_features),model.feature_importances_,align='center')
    plt.yticks(np.arange(n_features),cancer.feature_names)
    plt.xlabel("feature_importance")
    plt.ylabel("feature")
```

```
import mglearn # mglearn은 저자가 만든 라이브러리로 임의의 데이터셋 만들기가 대부분.
from sklearn.datasets import load_breast_cancer #사이킷런 데이터셋 중
유방암데이터가져오기.
from sklearn.datasets import make_blobs

cancer= load_breast_cancer()
Xc_train, Xc_test, yc_train , yc_test = train_test_split(cancer.data,cancer.target,
random_state = 0)
Xm, ym = make_moons(n_samples =100,noise=0.25,random_state=3)
Xm_train, Xm_test , ym_train, ym_test = train_test_split(Xm,ym,stratify=ym,random_state =0)

xtree = ExtraTreesClassifier(n_estimators=100,n_jobs=-1,random_state=0)
xtree.fit(Xc_train,yc_train)

n_features = cancer.data.shape[1]
plt.barh(range(n_features),xtree.feature_importances_)
plt.yticks(range(n_features),cancer.feature_names)
plt.xlabel("feature_importance")
plt.ylabel("feature")
plt.ylim(-1,n_features)
```



AdaBoost(Moon_dataset / esti = 5)

```
from sklearn.ensemble import AdaBoostClassifier

cancer= load_breast_cancer()
Xc_train, Xc_test, yc_train , yc_test = train_test_split(cancer.data,cancer.target,
random_state = 0)
Xm, ym = make_moons(n_samples =100,noise=0.25,random_state=3)
Xm_train, Xm_test , ym_train, ym_test =
train_test_split(Xm,ym,stratify=ym,random_state =0)

ada = AdaBoostClassifier(n_estimators=5, random_state=42)
ada.fit(Xm_train,ym_train)

fig , axes = plt.subplots(2,3,figsize=(20,10))
for i ,(ax, tree) in enumerate(zip(axes.ravel(), ada.estimators_)):
    ax.set_title("{} tree".format(i))
    mglearn.plots.plot_tree_partition(Xm,ym,tree,ax=ax)
mglearn.plots.plot_2d_separator(ada,Xm,fill=True,ax=axes[-1,-1],alpha=.4)
axes[-1,-1].set_title("adaboost")
mglearn.discrete_scatter(Xm[:,0],Xm[:,1],ym)
```



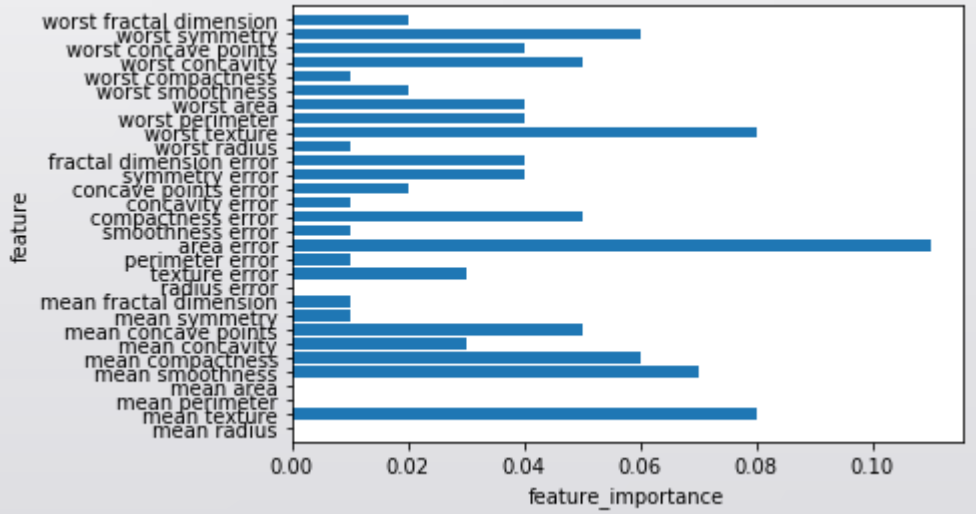
깊이가 1자리인 각각 Decision_Tree를 사용하기에 각트리의 경계는 선형으로 나온다.

AdaBoost(Cancer_dataset

```
ada = AdaBoostClassifier(n_estimators=100, random_state=42)
ada.fit(Xc_train,yc_train)

print("train_ac {:.3f} / test_ac
{:.3f}".format(ada.score(Xc_train,yc_train),ada.score(Xc_test,yc_test)))
```

```
train_ac 1.000 / test_ac 0.986
```

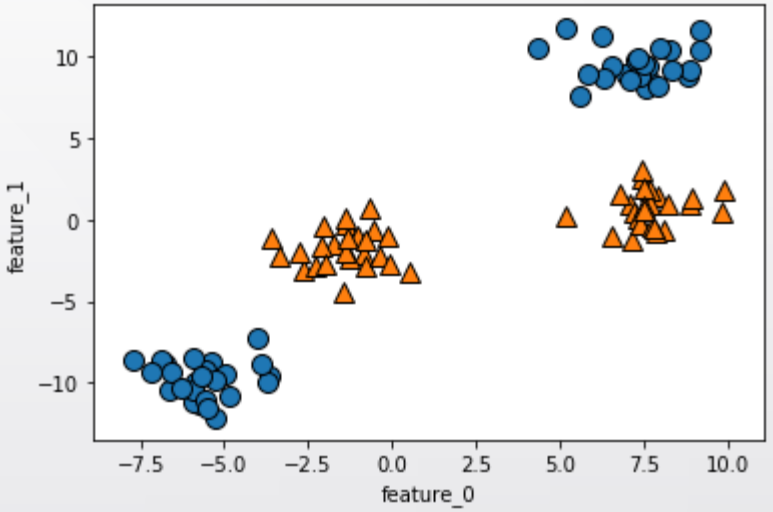
```
cancer= load_breast_cancer()
Xc_train, Xc_test, yc_train , yc_test = train_test_split(cancer.data,cancer.target,
random_state = 0)
Xm, ym = make_moons(n_samples =100,noise=0.25,random_state=3)
Xm_train, Xm_test , ym_train, ym_test =
train_test_split(Xm,ym,stratify=ym,random_state =0)

ada = AdaBoostClassifier(n_estimators=100, random_state=42)
ada.fit(Xc_train,yc_train)
plt.barh(np.arange(cancer.data.shape[1]),ada.feature_importances_)
plt.yticks(np.arange(cancer.data.shape[1]),cancer.feature_names)
plt.xlabel("feature_importance")
plt.ylabel("feature")
plt.ylim(-1,n_features)
```



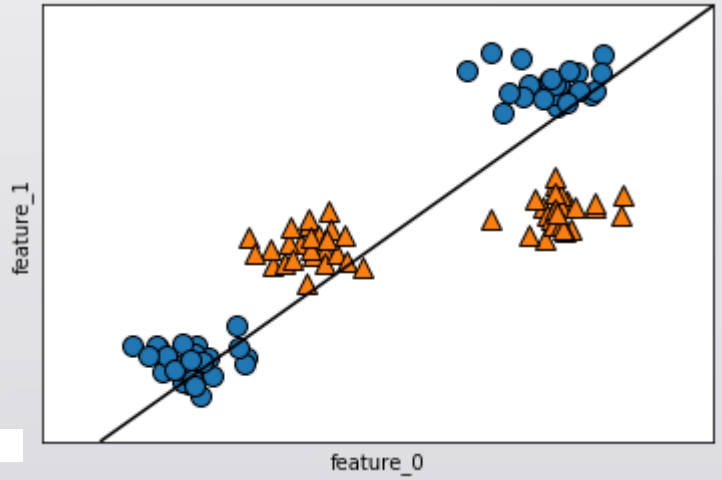
Kernelized support vector machines

Linear Model and unLinear feature

```
X,y = make_blobs(centers = 4 ,random_state=8)
y=y%2
mglearn.discrete_scatter(X[:,0],X[:,1],y)
plt.xlabel("feature_0")
plt.ylabel("feature_1")
```




```
X,y = make_blobs(centers = 4 ,random_state=8)

y=y%2 # 4개의 클래스를 2개로 만듬.
L_svc = LinearSVC().fit(X,y)
print("train_ac : {:.2f}".format(L_svc.score(X,y)))
mglearn.plots.plot_2d_separator(L_svc,X)
mglearn.discrete_scatter(X[:,0],X[:,1],y)
plt.xlabel("feature_0")
plt.ylabel("feature_1")
```

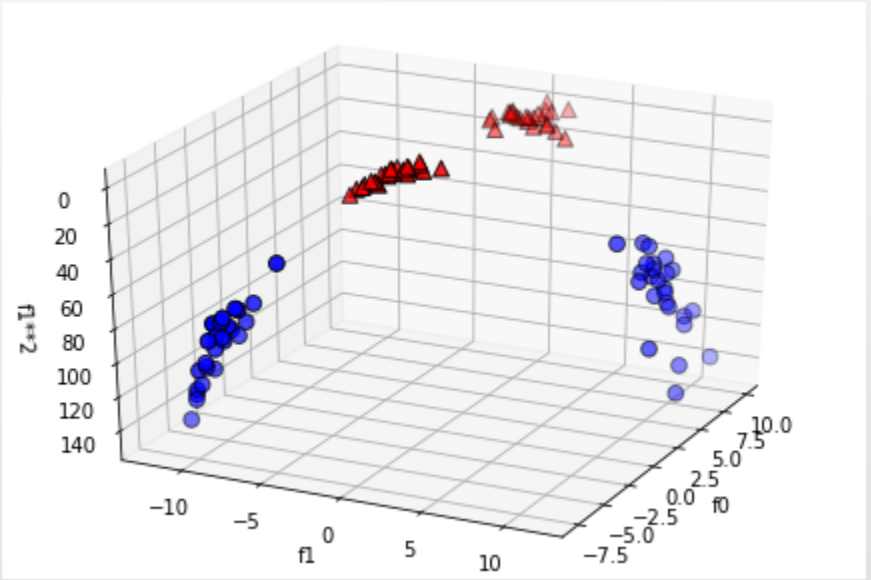



```
train_ac : 0.67
```

Kernelized support vector machines

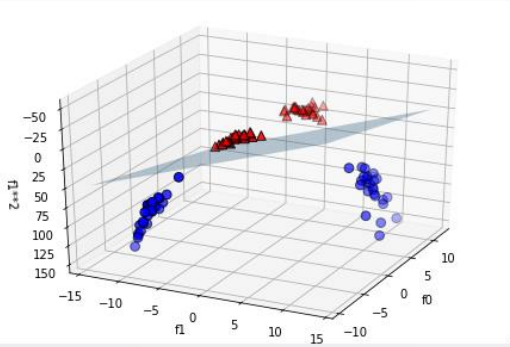
3d Scatter

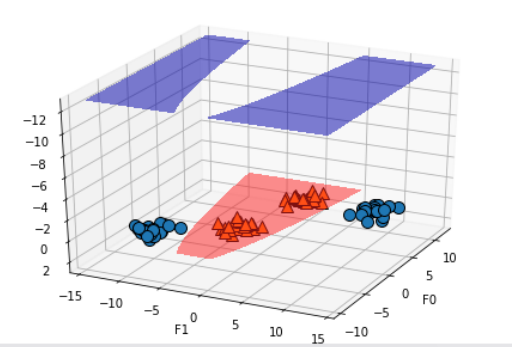
```
X,y = make_blobs(centers = 4 ,random_state=8)
y=y%2

X_new = np.hstack([X,X[:,1:]**2])
from mpl_toolkits.mplot3d import Axes3D, axes3d
figure = plt.figure()
ax = Axes3D(figure, elev=-152, azim=-26) #elev, azim -> 시야각?

mask = y ==0
ax.scatter(X_new[mask,0],X_new[mask,1],X_new[mask,2],c='b',s=60,edgecolor='k')
ax.scatter(X_new[~mask,0],X_new[~mask,1],X_new[~mask,2],c='r',s=60,marker='^',edgecolor='k')
ax.set_xlabel("f0")
ax.set_ylabel("f1")
ax.set_zlabel("f1**2")
```



Kernelized support vector machines

```
coef.shape : (3,)
```






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

X_new = np.hstack([X,X[:,1:]**2])

figure = plt.figure()
ax = Axes3D(figure, elev=-152, azim=-26) #elev, azim -> 시야각?
linear_svm_3d = LinearSVC().fit(X_new,y)
coef, intercept = linear_svm_3d.coef_.ravel(),linear_svm_3d.intercept_
print("coef.shape : {}".format(coef.shape))

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] # I dont know
ax.plot_surface(XX,YY,ZZ,rstride=8, cstride =8, alpha=.3) # rstride / cstride -> 높을수록 색변화율이 거칠다.
ax.scatter(X_new[mask,0],X_new[mask,1],X_new[mask,2],c='b',s=60,edgecolor='k')
ax.scatter(X_new[~mask,0],X_new[~mask,1],X_new[~mask,2],c='r',s=60,marker='^',edgecolor='k')
ax.set_xlabel("f0")
ax.set_ylabel("f1")
ax.set_zlabel("f1**2")

ZZ = YY ** 2
dec = linear_svm_3d.decision_function(np.c_[XX.ravel(),YY.ravel(),ZZ.ravel()])
plt.contourf(XX,YY,
dec.reshape(XX.shape),levels=[dec.min(),0,dec.max()],cmap=mglearn.cm2,alp
ha=.5)
mglearn.discrete_scatter(X[:,0],X[:,1],y)
plt.xlabel("F0")
plt.ylabel("F1")
```

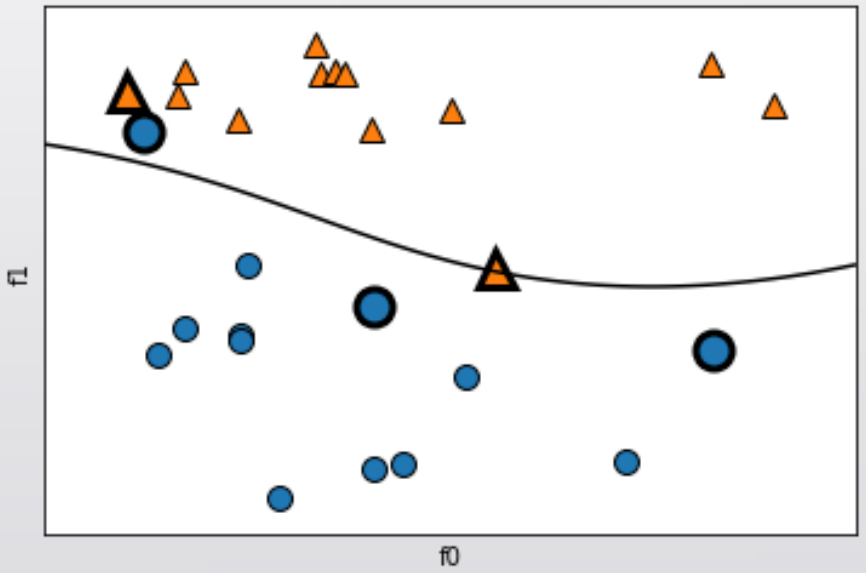
Kernel기법 이란?

Support_Vector란 두클래스 사이 경계에 위치한 데이터 포인트들을 의미한다.

새로운 데이터포인트에 대해 예측하려면 각 서포트 벡터와의 거리를 측정한다.

분류 결정은 서포트 벡터까지의 거리에 기반하며 서포트 벡터의 중요도는 훈련과정에서 학습합니다.(SVC객체의 dual_coef_속성에 저장됨.) -> 아래 그림의 굵은 그림이 서포트 벡터에 해당하며, 새로운데이터가 들어올시 학습된 서포트벡터와의 유크라디안 거리를 통해 클래스 레이블을 예측한다. -> 가우시안 커널(RBF)이다.

```
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)

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

sv = svm.support_vectors_

sv_labels = svm.dual_coef_.ravel()>0

mglearn.discrete_scatter(sv[:,0],sv[:,1],sv_labels, s=15, markeredgewidth=3)
plt.xlabel("f0")
plt.ylabel("f1")
```



SVM parameter_Tune

Gamma & C가 있다.

Gamma는 커널폭의 역수에 해당한다.

하나의 훈련 샘플이 미치는 영향의 범위를 결정한다.

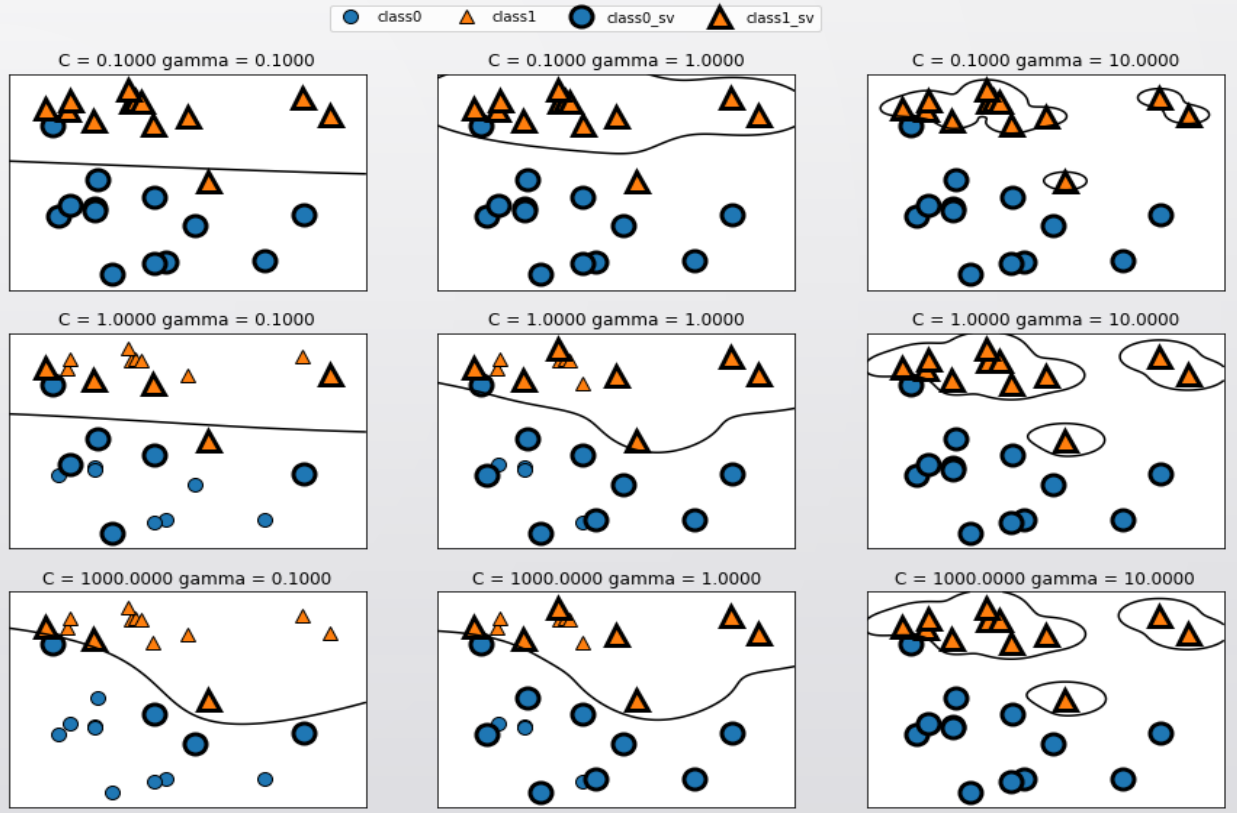
작은 값은 넓은 영역을 의미, 큰값이라면 영향을 미치는 범위가 제한적 (과대적합과 과소적합)

C매개 변수는 선형모델과 동일하게 규제 매개변수이다.

Dual_coef_를 제한한다.

SVM parameter_Tune

```
X,y = mglearn.tools.make_handcrafted_dataset()

fig , axes = plt.subplots(3,3,figsize=(15,10))
for ax , c in zip(axes,[-1,0,3]): # 10^(-1,0,3)
    for a, gamma in zip(ax, range(-1,2)): # 10^(-1,0,1)
        mglearn.plots.plot_svm(log_C =c, log_gamma=gamma, ax=a)


axes[0,0].legend(["class0", "class1", "class0_sv","class1_sv"],ncol=4,loc=(0.9,1.2))
```



SVM parameter_Tune (Cancer_dataset)

```
cancer= load_breast_cancer()
Xc_train, Xc_test, yc_train , yc_test = train_test_split(cancer.data,cancer.target,
random_state = 0)
Xm, ym = make_moons(n_samples =100,noise=0.25,random_state=3)
Xm_train, Xm_test , ym_train, ym_test = train_test_split(Xm,ym,stratify=ym,random_state =0)
print(cancer.feature_names.shape) # (30, )
svc = SVC() # basic gamma = 1/n_features / c = 1
svc.fit(Xc_train,yc_train)
print("train_ac : {:.2f}".format(svc.score(Xc_train,yc_train)))
print("test_ac : {:.2f}".format(svc.score(Xc_test,yc_test)))
```

```
(30,) train_ac : 1.00 test_ac : 0.63
```

Train 정확도는 100%에 Test 정확도 63프로로 봤을때 굉장히 과대적합되었다.

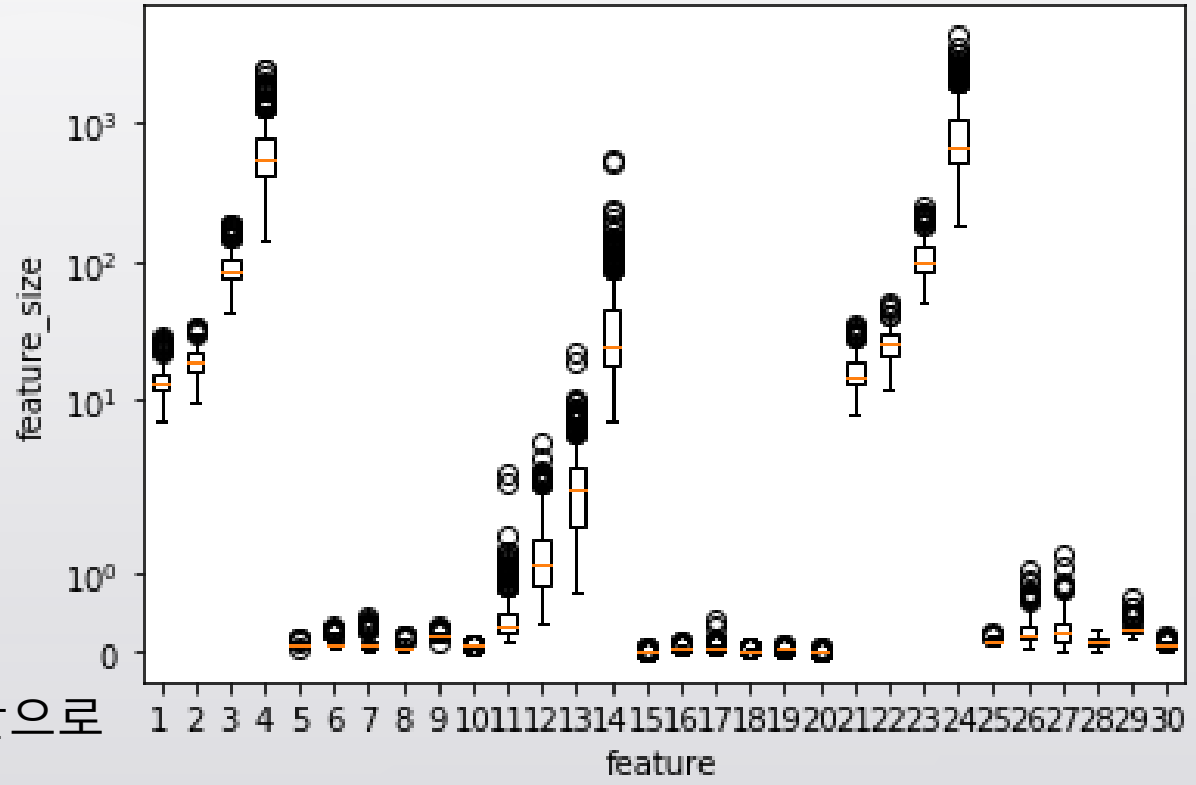
SVM parameter_Tune (Cancer_dataset)

각 특성의 최대 최솟값 로그스케일

```
plt.boxplot(Xc_train)
plt.yscale("symlog")
plt.xlabel("feature")
plt.ylabel("feature_size")
```

```
Xc_train[:,3].sum() # 280000
```

```
Xc_train[:,29].sum() #35.5
```



How solve ? 모든 특성값을 평균이 0인 단위분산으로 맞추거나 0과1사이로 맞추는 방법을 많이 사용함.

SVM parameter_Tune (Feature_Preprocessing)

```
cancer= load_breast_cancer()
Xc_train, Xc_test, yc_train , yc_test = train_test_split(cancer.data,cancer.target,
random_state = 0)
Xm, ym = make_moons(n_samples =100,noise=0.25,random_state=3)
Xm_train, Xm_test , ym_train, ym_test =
train_test_split(Xm,ym,stratify=ym,random_state =0)
svc = SVC() # basic gamma = 1/n_features / c = 1
svc.fit(Xc_train,yc_train)

min_on_training=Xc_train.min(axis=0) # 각 특성별 최솟값을 저장.
range_on_training = (Xc_train-min_on_training).max(axis=0)#최솟값에서
최댓값까지의 범위 계산

Xc_train_scaled = (Xc_train - min_on_training)/ range_on_training
print("feature_min ",Xc_train_scaled.min(axis=0))
print("feature_max ",Xc_train_scaled.max(axis=0))
```

```
feature_min [0. 0. 0. 0. 0. 0. 0.
0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.
0. 0. 0. 0. 0. 0. 0. 0. 0. 0. 0.
0.] feature_max [1. 1. 1. 1. 1.
1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1.
1. 1. 1. 1. 1. 1. 1. 1. 1. 1. 1.
1. 1. 1.]
```

(0~1)Scale공식 : (data-min)/(max-min)

SVM parameter_Tune (Feature_Preprocessing)

```
from sklearn.datasets import load_breast_cancer #사이킷런 데이터셋 중 유방암데이터가져오기.
from sklearn.datasets import make_blobs
from mpl_toolkits.mplot3d import Axes3D, axes3d
cancer= load_breast_cancer()
Xc_train, Xc_test, yc_train , yc_test = train_test_split(cancer.data,cancer.target, random_state = 0)
Xm, ym = make_moons(n_samples =100,noise=0.25,random_state=3)
Xm_train, Xm_test , ym_train, ym_test = train_test_split(Xm,ym,stratify=ym,random_state =0)

svc.fit(Xc_train,yc_train)

min_on_training=Xc_train.min(axis=0) # 각 특성별 최솟값을 저장.
range_on_training = (Xc_train-min_on_training).max(axis=0)#최솟값에서 최댓값까지의 범위 계산

Xc_train_scaled = (Xc_train - min_on_training)/ range_on_training

Xc_test_scaled = (Xc_test - min_on_training)/range_on_training

svc = SVC() # basic gamma = 1/n_features / c = 1
svc.fit(Xc_train_scaled,yc_train)

print("train_ac : {:.3f} / test_ac : {:.3f}".format(svc.score(Xc_train_scaled,yc_train),svc.score(Xc_test_scaled,yc_test)))
```

```
train_ac : 0.948 / test_ac : 0.951
```

과소적합이다.

SVM의 장단점 :

저차원 고차원 데이터 모두 잘작동하지만,
10000개정도의 샘플에서는 괜찮지만,
100000개정도부터는 메모리 및 속도 관점에서는 도전적이다.

전처리가 중요하다. 매개변수 설정이 성능에 큰영향을 미친다.

또한 시각적으로 표현하기 쉽지않다.
->데이터의 스케일이 비슷한 단위며 해볼만 하다.

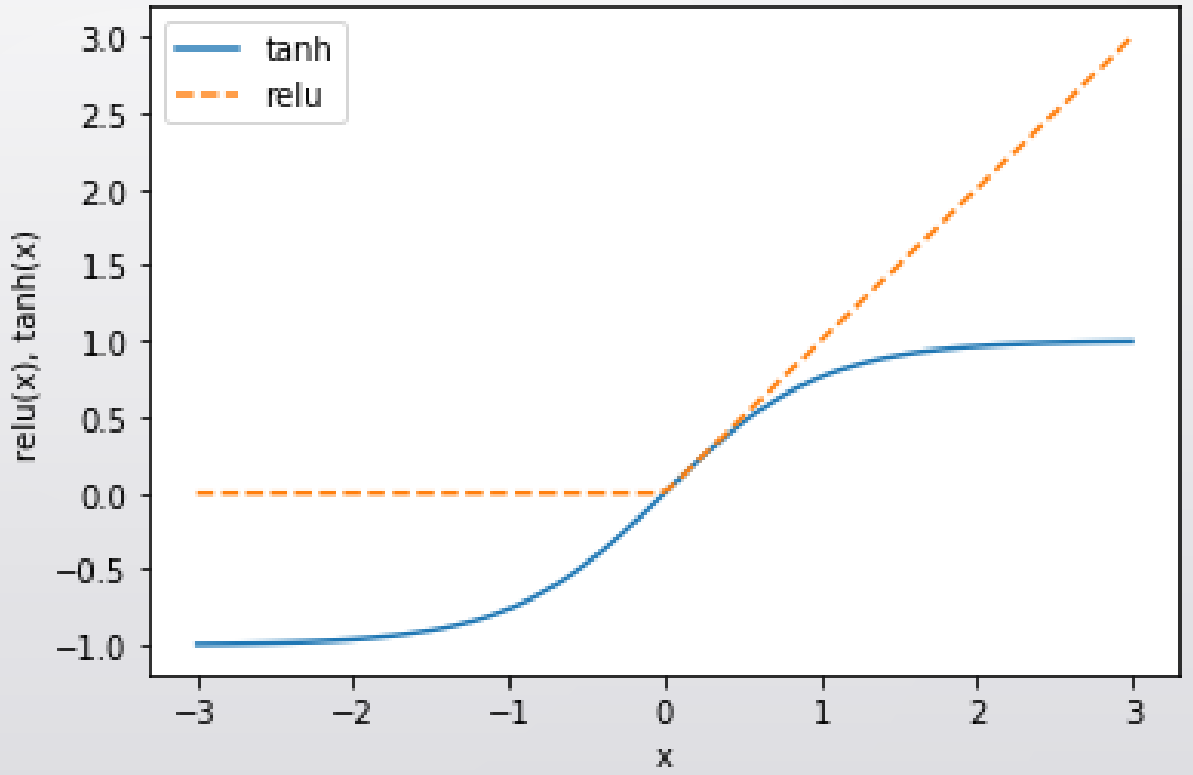
Hyperparameter를 증가시켜서 모델의 복잡도를 증가시키고, 괜찮은 성능을 얻었다.

```
svc = SVC(C=1000) # basic gamma = 1/n_features / c = 1
```

```
train_ac : 0.988 / test_ac : 0.972
```

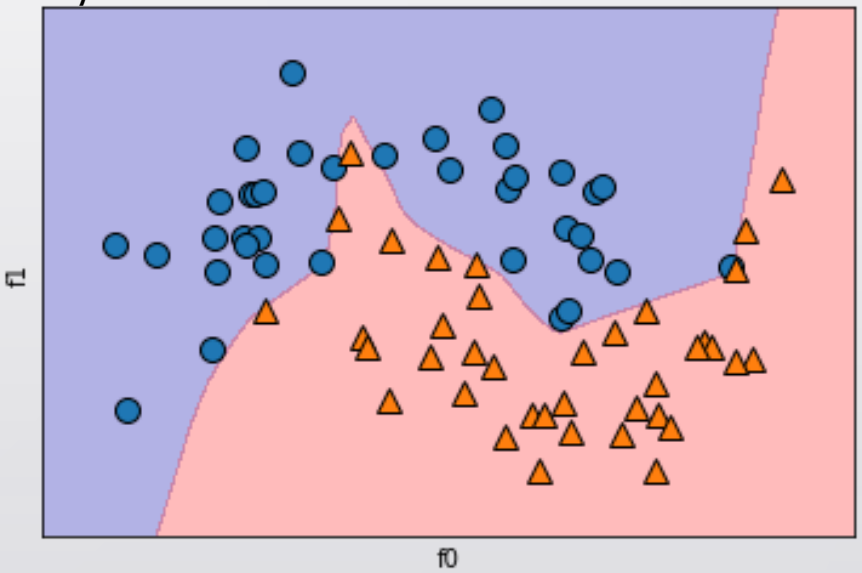
MLP(MultiLayerPercpetron)

```
line = np.linspace(-3,3,100)
plt.plot(line,np.tanh(line),label="tanh")
plt.plot(line,np.maximum(line,0),linestyle='--',label="relu")
plt.legend(loc="best")
plt.xlabel("x")
plt.ylabel("relu(x), tanh(x)")
```



MLP(MultiLayerPercpetron) tuning -> two moons dataset

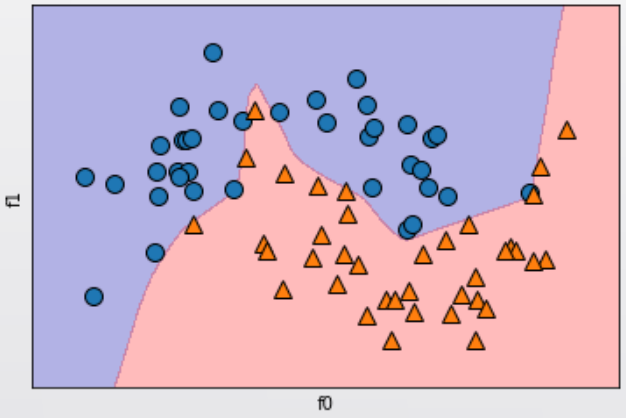
```
from sklearn.neural_network import MLPClassifier

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=0)
mlp = MLPClassifier(solver='lbfgs',random_state=0).fit(X_train,y_train)
mglearn.plots.plot_2d_separator(mlp,X_train,fill=True,alpha=.3)
mglearn.discrete_scatter(X_train[:,0],X_train[:,1],y_train)
plt.xlabel("f0")
plt.ylabel("f1")
```



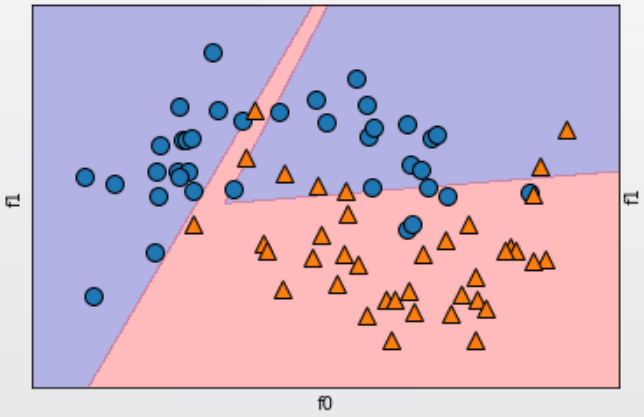
기본 은닉층 개수 : 100개 하지만 이 100개의 샘플에 100개의 은닉층은 과분하기에 10개로 줄여보자.

MLP(MultiLayerPercpetron) tuning -> two moons dataset

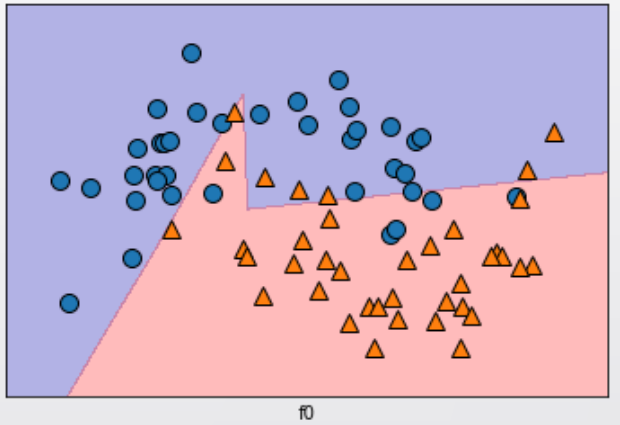
기본 은닉유닛 개수 : 100개 하지만 이 100개의 샘플에 100개의 은닉유닛은
과분하기에 10개로 줄여보자.



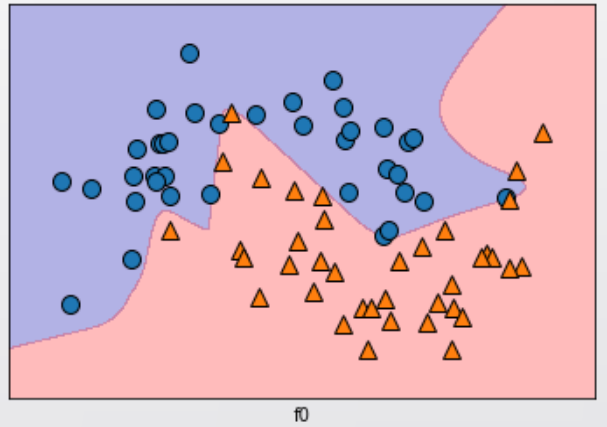
은닉유닛 : 100개
레이어 : 1층
활성함수 : Relu



은닉유닛 : 10개
레이어 : 1층
활성함수 : Relu



은닉유닛 : 10개
레이어 : 2층
활성함수 : Relu



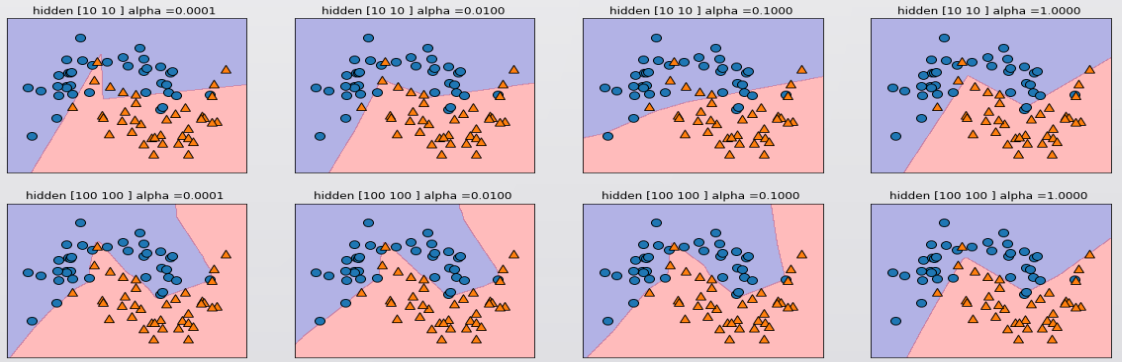
은닉유닛 : 10개
레이어 : 2층
활성함수: tanh

은닉유닛 10개 은닉층 : 2층 활성함수 : tanh 예시

```
mlp = MLPClassifier(solver='lbfgs',hidden_layer_sizes=[10,10],activation='tanh',random_state=0).fit(X_train,y_train)
```

MLP(MultiLayerPercpetron) tuning -> two moons dataset

```
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=0)
fig, axes = plt.subplots(2,4, figsize =(20,8))
for ax,hidden_node in zip(axes,[10,100]):
    for a , alpha in zip(ax,[0.0001,0.01,0.1,1]):
        mlp =
MLPClassifier(solver='lbfgs',random_state=0,hidden_layer_sizes=[hidden_node,hidden_node],alph
a=alpha)
        mlp.fit(X_train,y_train)
        mglearn.plots.plot_2d_separator(mlp,X_train,fill=True,alpha=.3,ax=a)
        mglearn.discrete_scatter(X_train[:,0],X_train[:,1],y_train,ax=a)
        a.set_title("hidden [{} {} ] alpha ={:.4f} ".format(hidden_node,hidden_node,alpha))
```

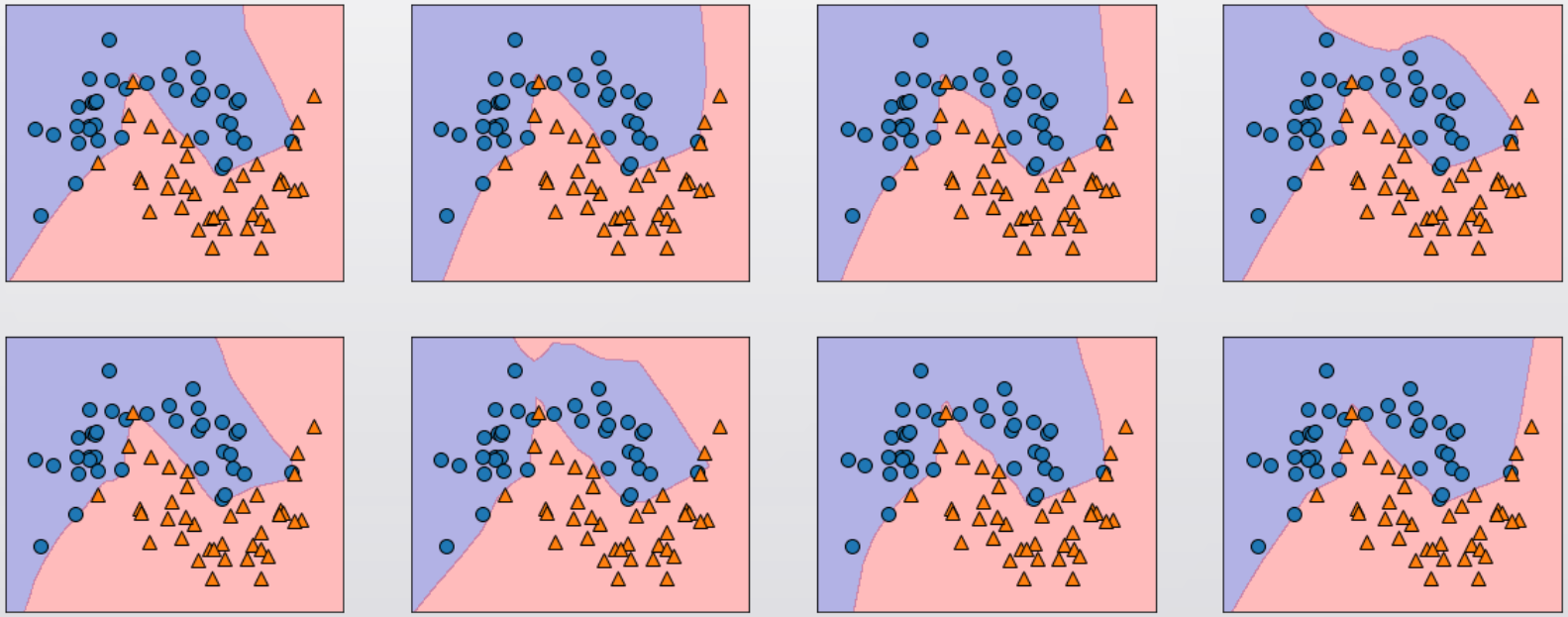


신경망의 초기 가중치 값은 무작위한 초기화로 이루어지는데,
모델의 학습에 영향을 준다.

같은 매개변수를 사용하더라도 초깃값이 다르면 모델이 많이 달라질수 있다.

MLP(MultiLayerPercpetron) tuning -> other init

```
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=0)
fig, axes = plt.subplots(2,4, figsize =(20,8))
for i, ax in enumerate(axes.ravel()):
    mlp = MLPClassifier(solver='lbfgs',random_state=i,hidden_layer_sizes=[100,100])
    mlp.fit(X_train, y_train)
    mglearn.plots.plot_2d_separator(mlp,X_train,fill=True,alpha=.3,ax=ax)
    mglearn.discrete_scatter(X_train[:,0],X_train[:,1],y_train,ax=ax)
```



MLP(MultiLayerPercpetron) Cancer_dataset

```
cancer= load_breast_cancer()
Xc_train, Xc_test, yc_train , yc_test = train_test_split(cancer.data,cancer.target, random_state = 0)
Xm, ym = make_moons(n_samples =100,noise=0.25,random_state=3)
Xm_train, Xm_test , ym_train, ym_test = train_test_split(Xm,ym,stratify=ym,random_state =0)

mlp=MLPClassifier(random_state = 42)
mlp.fit(Xc_train,yc_train)

print("train_ac : {:.3f}, test_ac : {:.3f}".format(mlp.score(Xc_train, yc_train),mlp.score(Xc_test,yc_test)))
```

```
train_ac : 0.939, test_ac : 0.916
```

=>다른 모델에 비해 성능이 좋지않다. 특성의 스케일을 0~1로 맞춰주도록 한다.

MLP(MultiLayerPercpetron) Cancer_dataset

```
cancer= load_breast_cancer()
Xc_train, Xc_test, yc_train , yc_test = train_test_split(cancer.data,cancer.target, random_state = 0)
Xm, ym = make_moons(n_samples =100,noise=0.25,random_state=3)
Xm_train, Xm_test , ym_train, ym_test = train_test_split(Xm,ym,stratify=ym,random_state =0)

mlp=MLPClassifier(random_state = 42)
mlp.fit(Xc_train,yc_train)

print("train_ac : {:.3f}, test_ac : {:.3f}".format(mlp.score(Xc_train, yc_train),mlp.score(Xc_test,yc_test)))
```

```
train_ac : 0.939, test_ac : 0.916
```

=>다른 모델에 비해 성능이 좋지않다. 특성의 스케일을 0~1로 맞춰주도록 한다.

MLP(MultiLayerPercpetron) Cancer_dataset

```
Xc_train, Xc_test, yc_train , yc_test = train_test_split(cancer.data,cancer.target, random_state = 0)
Xm, ym = make_moons(n_samples =100,noise=0.25,random_state=3)
Xm_train, Xm_test , ym_train, ym_test = train_test_split(Xm,ym,stratify=ym,random_state =0)

#z점수 공식 : (data-평균) / 표준편차
mean_on_train = Xc_train.mean(axis=0) # mean ;> 평균
std_on_train = Xc_train.std(axis=0) # std -> 표준편차

X_trained_scaled = (Xc_train-mean_on_train)/std_on_train
X_test_scaled = (Xc_test-mean_on_train)/std_on_train

mlp = MLPClassifier(random_state=0)
mlp.fit(X_trained_scaled, yc_train)

print("train_ac {:.2f} / test_ac
{:.2f}".format(mlp.score(X_trained_scaled,yc_train),mlp.score(X_test_scaled,yc_test)))
```

```
train_ac 0.99 / test_ac 0.97
```

MLP(MultiLayerPercpetron) (Max_iteration,alpha) reinforce

```
Xc_train, Xc_test, yc_train , yc_test = train_test_split(cancer.data,cancer.target, random_state = 0)
Xm, ym = make_moons(n_samples =100,noise=0.25,random_state=3)
Xm_train, Xm_test , ym_train, ym_test = train_test_split(Xm,ym,stratify=ym,random_state =0)

#z점수 공식 : (data-평균) / 표준편차
mean_on_train = Xc_train.mean(axis=0) # mean ;> 평균
std_on_train = Xc_train.std(axis=0) # std -> 표준편차

X_trained_scaled = (Xc_train-mean_on_train)/std_on_train
X_test_scaled = (Xc_test-mean_on_train)/std_on_train

mlp = MLPClassifier(max_iter=1000,random_state=0)
mlp.fit(X_trained_scaled, yc_train)

print("train_ac {:.2f} / test_ac
{:.2f}".format(mlp.score(X_trained_scaled,yc_train),mlp.score(X_test_scaled,yc_test)))
```

```
train_ac 1.00 / test_ac 0.97
```

=> 성능 향상은 되었지만, test성능과의 일반화를 위해서 규제를 강화시키는 alpha변수를 크게 올려보자.

```
mlp = MLPClassifier(max_iter=1000,alpha=1.0,random_state=0)
```

```
train_ac 0.99 / test_ac 0.97
```

MLP(MultiLayerPercpetron) 장점과 단점

학습이 오래걸린다.

데이터 전처리가 필요하다.

신경망 매개변수 튜닝이 까다롭다.

Pytorch , Keras, Tensorflow
-> 딥러닝 라이브러리

MLP모델 설계시 일단 과대적합이 되도록 설계한 후 매개변수 조정을 통해, 일반화를 시키는 방향으로 진행한다.

최적화 -> adam / lbfgs => adam은 데이터전처리가 필수이며(scaling) , lbfgs 큰데이터에는 시간이 오래걸린다.
추가적으로 : sgd

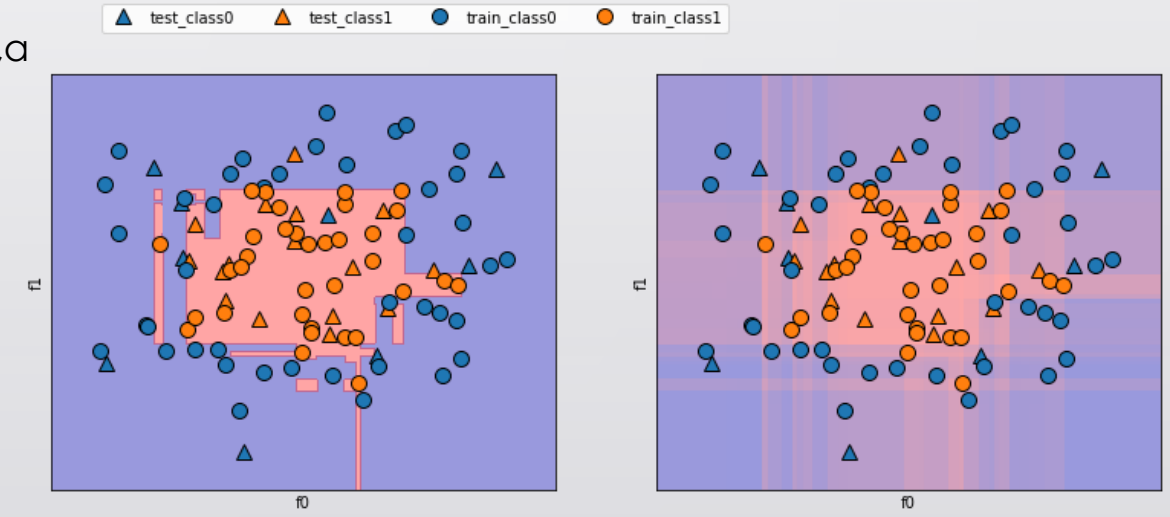
분류(Classification) 예측(Prediction) 불확실성 추정

```
X, y = make_circles(noise=0.25,factor=0.5,random_state=1)
y_named = np.array(["blue","red"])[y]
X_train, X_test , y_train_named, y_test_named , y_train, y_test
=train_test_split(X,y_named,y,random_state=0)

gbrt = GradientBoostingClassifier(random_state=0).fit(X_train,y_train_named)

greater_zero = (gbrt.decision_function(X_test)>0).astype(int)
pred = gbrt.classes_[greater_zero]
print(np.all(pred == gbrt.predict(X_test)))
```

```
True
```

분류(Classification) 예측(Prediction) 불확실성 추정 1.decision_function

```
fig, axes = plt.subplots(1,2,figsize=(13,5))
mglearn.tools.plot_2d_separator(gbrt,X,ax=axes[0],alpha=.4,fill=True,
cm=mglearn.cm2)
scores_image =
mglearn.tools.plot_2d_scores(gbrt,X,ax=axes[1],alpha=.4,cm=mglear
n.ReBl)

for ax in axes :

mglearn.discrete_scatter(X_test[:,0],X_test[:,1],y_test,markers='^',ax=
ax)

mglearn.discrete_scatter(X_train[:,0],X_train[:,1],y_train,markers='o',a
x=ax)

    ax.set_xlabel("f0")
    ax.set_ylabel("f1")
axes[0].legend(["test_class0", "test_class1", "train_class0",
"train_class1"],ncol=4,loc=(.1,1.1))
```



분류(Classification) 예측(Prediction) 불확실성 추정 2. predict_proba