분류형 선형 모델

학습 내용

- 이진 분류의 알고리즘에 대해 알아봅니다.
- 심플한 데이터를 활용하여 이진 분류 알고리즘을 실습해 봅니다.
- 로지스틱 회귀 모델과 서포트벡터 머신에 대해 알아보며, 규제 매개변수 C에 대해 알아봅니다.

- 분류용 선형 모델에서는 **결정경계**가 입력의 선형 함수
- 이진 선형 분류기는 선, 평면, 초평면을 사용해서 **두 개의 클래스를 구분하는 분류기**

이진 분류를 위한 예측 방정식

- y = w[0] * x[0] + w[1] * x[1] + w[2] * x[2] + ... + w[p] * x[p] + b > 0

- 특성들의 가중치 합을 사용하는 대신, 예측한 값을 임계치 0과 비교.
- **0보다 작으면 클래스를 -1, 0보다 크면 1**로 예측

이진 분류를 위한 잘 알려진 알고리즘

- 로지스틱 회귀(Logistic regression)
- 서포트 벡터 머신(support vector machine)

In [5]:

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

%matplotlib inline
```

라이브러리 불러오기

In [6]:

```
from sklearn.linear_model import LogisticRegression
from sklearn.svm import LinearSVC
import mglearn
import matplotlib.pyplot as plt
```

데이터셋 로드

- 26개의 행과 2개의 열의 입력
- 26개의 출력

In [7]:

```
X, y = mglearn.datasets.make_forge()
print(X.shape, y.shape)
print(X[:5], y[:5])  # 5개 데이터
```

```
(26, 2) (26,)
[[ 9.96346605  4.59676542]
 [11.0329545  -0.16816717]
 [11.54155807  5.21116083]
 [ 8.69289001  1.54322016]
 [ 8.1062269   4.28695977]] [1 0 1 0 0]

C:\Users\WJ\anaconda3\lib\site-packages\sklearn\utils\deprecation.py:86: FutureWarni
ng: Function make_blobs is deprecated; Please import make_blobs directly from scikit
-learn
  warnings.warn(msg, category=FutureWarning)
```

데이터 셋을 이용하여 로지스틱 회귀와 서포트벡터 머신 구현

- LogisticRegression과 LinearSVC모델 생성
- 데이터 셋 : forge 데이터 셋
- 모델 : LogisticRegression과 LinearSVC모델

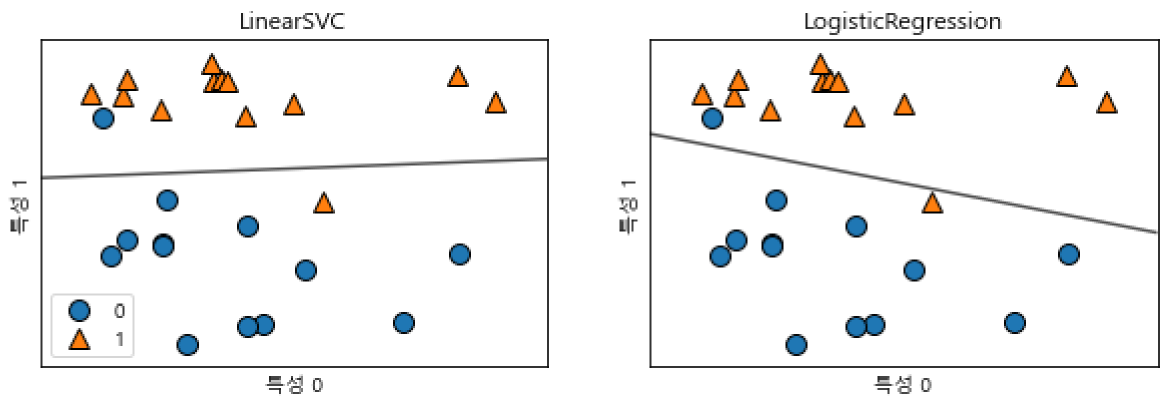
In [8]:

```
fig, axes = plt.subplots(1, 2, figsize=(10,3))

for model, ax in zip([LinearSVC(), LogisticRegression()], axes):
    clf = model.fit(X, y)
    mglearn.plots.plot_2d_separator(clf, X, fill=False, eps=0.5, ax=ax, alpha=0.7)
    mglearn.discrete_scatter(X[:,0], X[:,1], y, ax=ax)
    ax.set_title(clf. __class__.__name__)
    ax.set_xlabel("특성 0")
    ax.set_ylabel("특성 1")
axes[0].legend()
```

```
C:\Users\WJ\anaconda3\lib\site-packages\sklearn\svm\_base.py:976: ConvergenceWarnin
g: Liblinear failed to converge, increase the number of iterations.
  warnings.warn("Liblinear failed to converge, increase "
```

Out[8]:

```
<matplotlib.legend.Legend at 0x2bcfbac4f40>
```



- 위쪽은 클래스 1, 아래쪽은 클래스 0으로 분류.
  - 새로운 데이터가 **위쪽에 놓이면 클래스 1로 분류**
  - 새로운 데이터가 반대로 **직선 아래쪽에 놓이면 클래스 0으로 분류**
- 두개의 모델은 기본적으로 L2규제(계수를 0에 가깝게)를 이용

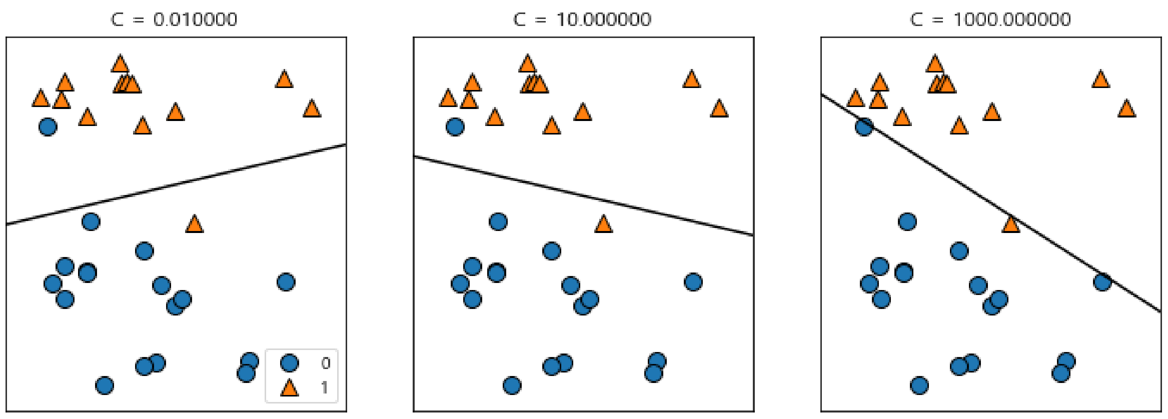
매개변수 C

- LogisticRegression과 LinearSVC에서 규제의 강도를 결정하는 매개변수 C
  - C의 값이 높아지면 규제가 감소
    - 높은 C의 값을 지정하면 LogisticRegression과 LinearSVC 학습용 데이터 세트에 최대한 맞추게 됨.
  - C의 값이 낮추면 모델 계수 벡터(w)가 0에 가까워지도록 함.

각기 다른 C값으로 만든 SVM 모델의 결정 경계

In [9]:

```
mglearn.plots.plot_linear_svc_regularization()
```



- 왼쪽 그림은 아주 작은 C값으로 인해 규제가 많이 적용됨.
- 규제가 비교적 강해진 모델은 비교적 수평에 가까운 결정
- C값을 아주 크게한 오른쪽은 마침내 클래스의 0의 모든 데이터를 올바르게 분류함.

유방암 데이터셋을 활용한 LogisticRegression

In [10]:

```
from sklearn.datasets import load_breast_cancer
cancer = load_breast_cancer()
from sklearn.model_selection import train_test_split
```

In [11]:

```
X_train, X_test, y_train, y_test = train_test_split(cancer.data, cancer.target,
                                                    stratify=cancer.target,
                                                    random_state=42)
# C의 기본값은 1
logreg = LogisticRegression(C=1).fit(X_train, y_train)

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

```
훈련 세트 점수 : 0.941
테스트 세트 점수 : 0.965

C:\Users\WJ\anaconda3\lib\site-packages\sklearn\linear_model\_logistic.py:762: Conve
rgenceWarning: lbfgs failed to converge (status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html (https://scikit-lear
n.org/stable/modules/preprocessing.html)
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression (h
ttps://scikit-learn.org/stable/modules/linear_model.html#logistic-regression)
  n_iter_i = _check_optimize_result(
```

- 기본값 C=1이 훈련세트와 테스트 양쪽이 95%로 정확도로 꽤 훌륭함.
  - 단, 훈련세트와 테스트 세트의 성능이 매우 비슷하므로 과소적합으로 판단됨.
    - 모델의 제약을 풀기 위해 C를 증가시켜봄.

In [12]:

```
logreg100 = LogisticRegression(C=100).fit(X_train, y_train)

print("훈련 세트 점수 : {:.3f}".format(logreg100.score(X_train, y_train)))
print("테스트 세트 점수 : {:.3f}".format(logreg100.score(X_test, y_test)))
```

```
훈련 세트 점수 : 0.951
테스트 세트 점수 : 0.958

C:\Users\WJ\anaconda3\lib\site-packages\sklearn\linear_model\_logistic.py:762: Conve
rgenceWarning: lbfgs failed to converge (status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html (https://scikit-lear
n.org/stable/modules/preprocessing.html)
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression (h
ttps://scikit-learn.org/stable/modules/linear_model.html#logistic-regression)
  n_iter_i = _check_optimize_result(
```

- 훈련 세트 점수가 증가 -> 복잡도가 높은 모델일 수록 성능이 좋음.
  - C의 값이 증가되면 복잡도가 증가
  - C의 값이 감소되면 제약이 많아 단순해짐

In [13]:

```
logreg001 = LogisticRegression(C=0.01).fit(X_train, y_train)

print("훈련 세트 점수 : {:.3f}".format(logreg001.score(X_train, y_train)))
print("테스트 세트 점수 : {:.3f}".format(logreg001.score(X_test, y_test)))
```

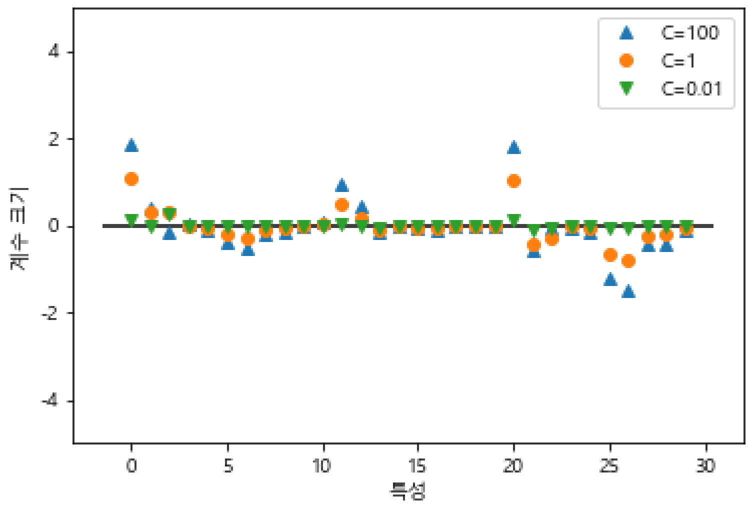
```
훈련 세트 점수 : 0.937
테스트 세트 점수 : 0.930

C:₩Users₩WJ₩anaconda3₩lib₩site-packages₩sklearn₩linear_model₩_logistic.py:762: Conve
rgenceWarning: lbfgs failed to converge (status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html (https://scikit-lear
n.org/stable/modules/preprocessing.html)
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression (h
ttps://scikit-learn.org/stable/modules/linear_model.html#logistic-regression)
  n_iter_i = _check_optimize_result(
```

In [14]:

```
plt.plot(logreg100.coef_.T, '^', label="C=100")
plt.plot(logreg.coef_.T, 'o', label="C=1")
plt.plot(logreg001.coef_.T, 'v', label="C=0.01")

xlims = plt.xlim()
plt.hlines(0, xlims[0], xlims[1])

plt.ylim(-5, 5)
plt.xlabel("특성")
plt.ylabel("계수 크기")
plt.legend()
```

Out[14]:

```
<matplotlib.legend.Legend at 0x2bcfbc1ba90>
```



- C의 값이 **클수록 규제를 풀고**, C의 값이 **적을수록 많은 규제**가 이루어진다.

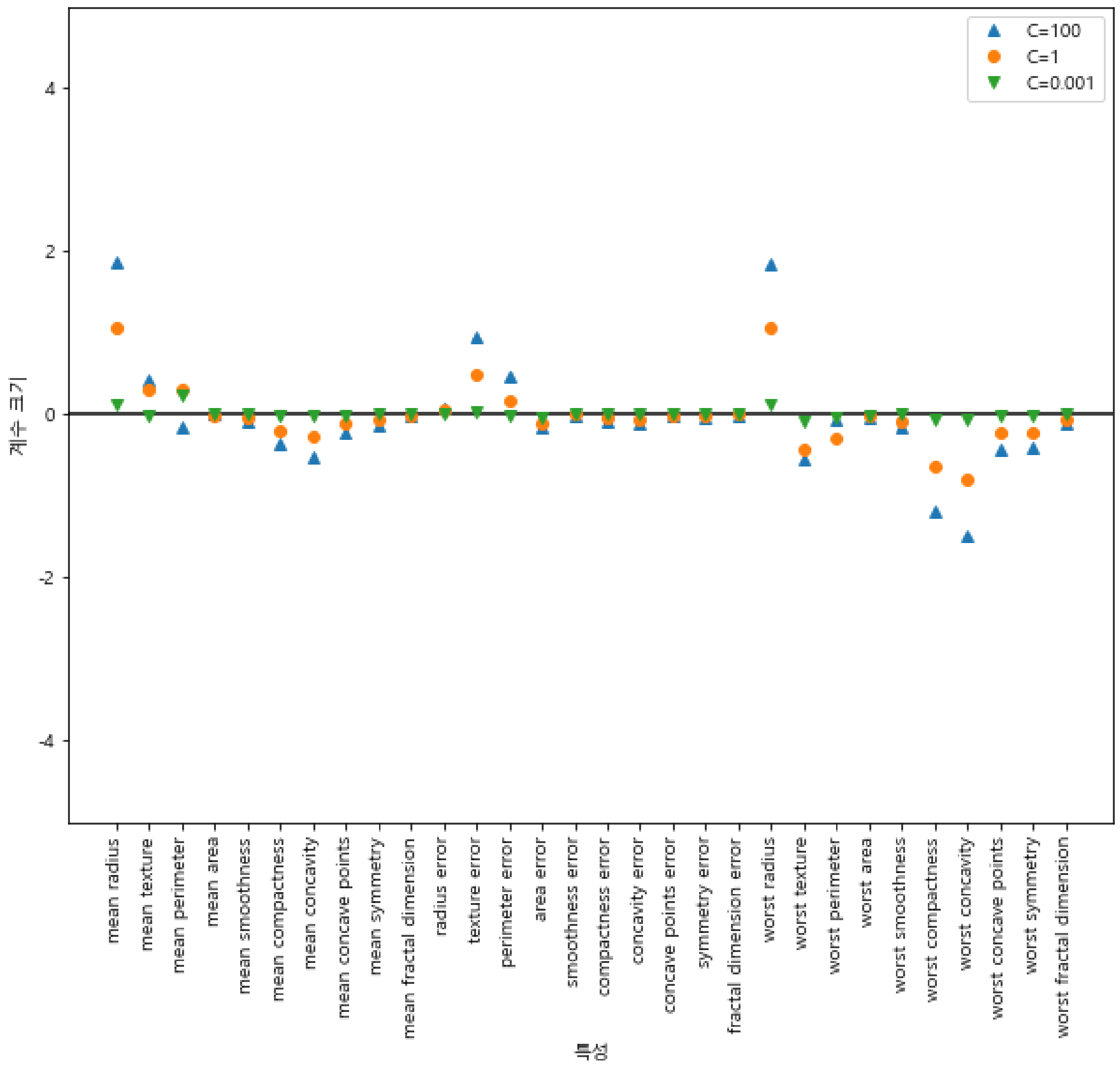
- 로지스틱 회귀(LogisticRegression)은 기본으로 L2규제를 적용하고 있음. (0에 가깝게, 0으로 만들지 않음.)

In [15]:

```
plt.figure(figsize=(10,8))
plt.plot(logreg100.coef_.T, '^', label="C=100")
plt.plot(logreg.coef_.T, 'o', label="C=1")
plt.plot(logreg001.coef_.T, 'v', label="C=0.001")
plt.xticks(range(cancer.data.shape[1]), cancer.feature_names, rotation=90)
xlims = plt.xlim()
plt.hlines(0, xlims[0], xlims[1])
plt.xlim(xlims)
plt.ylim(-5, 5)
plt.xlabel("특성")
plt.ylabel("계수 크기")
plt.legend()
```

Out[15]:

```
<matplotlib.legend.Legend at 0x2bcfbc95580>
```



L1규제 적용한 로지스틱 회귀 모델의 계수

- penalty의 매개변수를 이용하여 모든 특성을 사용할지(L2), 일부 특성만 사용할지(L1)을 결정

In [18]:

```
for C, marker in zip([0.001, 1, 100], ['o', '^', 'v']):
    lr_l1 = LogisticRegression(C=C,
                               penalty="none",
                               solver='lbfgs',
                               max_iter=100).fit(X_train, y_train)
    print("C={:.3f} 인 l1 로지스틱 회귀의 훈련 정확도: {:.2f}".format(
          C, lr_l1.score(X_train, y_train)))
    print("C={:.3f} 인 l1 로지스틱 회귀의 테스트 정확도: {:.2f}".format(
          C, lr_l1.score(X_test, y_test)))
    plt.plot(lr_l1.coef_.T, marker, label="C={:.3f}".format(C))

plt.xticks(range(cancer.data.shape[1]), cancer.feature_names, rotation=90)
xlims = plt.xlim()
plt.hlines(0, xlims[0], xlims[1])
plt.xlim(xlims)
plt.xlabel("특성")
plt.ylabel("계수 크기")

plt.ylim(-5, 5)
plt.legend(loc=3)
```

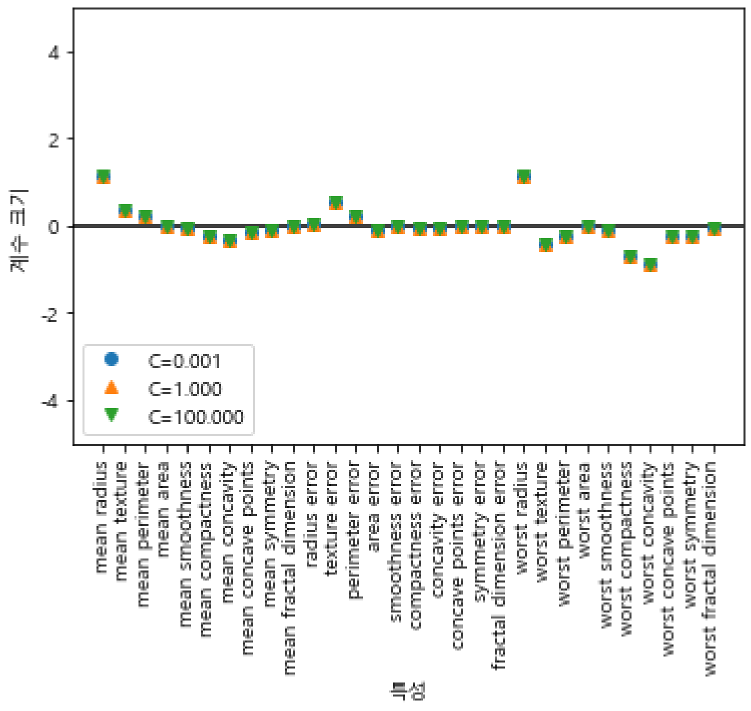
```
C=0.001 인 l1 로지스틱 회귀의 훈련 정확도: 0.94
C=0.001 인 l1 로지스틱 회귀의 테스트 정확도: 0.95
C=1.000 인 l1 로지스틱 회귀의 훈련 정확도: 0.94
C=1.000 인 l1 로지스틱 회귀의 테스트 정확도: 0.95
C=100.000 인 l1 로지스틱 회귀의 훈련 정확도: 0.94
C=100.000 인 l1 로지스틱 회귀의 테스트 정확도: 0.95
```

```
C:\Users\WJ\anaconda3\lib\site-packages\sklearn\linear_model\_logistic.py:1320: Us
erWarning: Setting penalty='none' will ignore the C and l1_ratio parameters
  warnings.warn(
C:\Users\WJ\anaconda3\lib\site-packages\sklearn\linear_model\_logistic.py:762: Con
vergenceWarning: lbfgs failed to converge (status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html (https://scikit-lea
rn.org/stable/modules/preprocessing.html)
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression
 (https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression)
  n_iter_i = _check_optimize_result(
C:\Users\WJ\anaconda3\lib\site-packages\sklearn\linear_model\_logistic.py:762: Con
vergenceWarning: lbfgs failed to converge (status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html (https://scikit-lea
rn.org/stable/modules/preprocessing.html)
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression
 (https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression)
  n_iter_i = _check_optimize_result(
C:\Users\WJ\anaconda3\lib\site-packages\sklearn\linear_model\_logistic.py:1320: Us
erWarning: Setting penalty='none' will ignore the C and l1_ratio parameters
  warnings.warn(
C:\Users\WJ\anaconda3\lib\site-packages\sklearn\linear_model\_logistic.py:762: Con
```

```
vergenceWarning: lbfgs failed to converge (status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html (https://scikit-lea
rn.org/stable/modules/preprocessing.html)
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression
  (https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression)
  n_iter_i = _check_optimize_result(
```

Out[18]:

```
<matplotlib.legend.Legend at 0x2bcfbd896d0>
```



history

- Machine Learning with sklearn @ DJ,Lim
- date : 2020/12/02 -- version 03