CH03 비지도 학습과 데이터 전처리

학습 내용

- 1. 비지도 학습의 종류에 대해 알아봅니다.
- 2. 데이터 전처리와 스케일 조정에 대해 알아봅니다.
- 3. cancer 데이터 셋을 이용하여 정규화, 표준화를 수행해 봅니다.

목차

01. 비지도 학습이란?

목차로 이동하기

- 우리가 예측해야 하는 값(target)의 출력값이 없이 입력 데이터 만을 이용하여 학습 알고리즘을 가르쳐야 하는 모든 종류의 머신러닝

02. 비지도 학습의 종류

목차로 이동하기

> 비지도 변환(unsupervised transformation)과 군집(clustering), 연관 알고리즘

비지도 변환(unsupervised transformation)

> A. 비지도 변환은 데이터를 새롭게 표현하여,
> B. 사람이나 다른 머신러닝 알고리즘이 원래 데이터보다 쉽게 해석할 수 있도록 만드는 알고리즘이다.

사용 분야

많이 사용되는 분야는 특성이 많은 고차원 데이터의 **특성(feature)의 수**를 줄이면서 꼭 필요한 특징을 포함한 데이터로 표현하는 방법인 **차원 축소(dimensionality reduction)**이다.

현재 어려운 부분

(가) 알고리즘이 좋은지 나쁜지를 학습한 내용에 대한 평가
(나) 우리가 뭔가 원하는 것을 알려줄 수 없다.
-> 비지도 학습 알고리즘은 데이터 과학자가 데이터를 잘 이해하고 싶을 때 분석 단계에서 많이 사용

소셜 미디어에서 선거, 총기, 팝스타 같은 주제로 일어나는 토론을 추적, 텍스트 문서에서 주제를 추출

군집 알고리즘(Clustering)

데이터를 비슷한 것끼리 그룹(클러스터)으로 묶는 것.

03. 데이터 전처리와 스케일 조정

목차로 이동하기

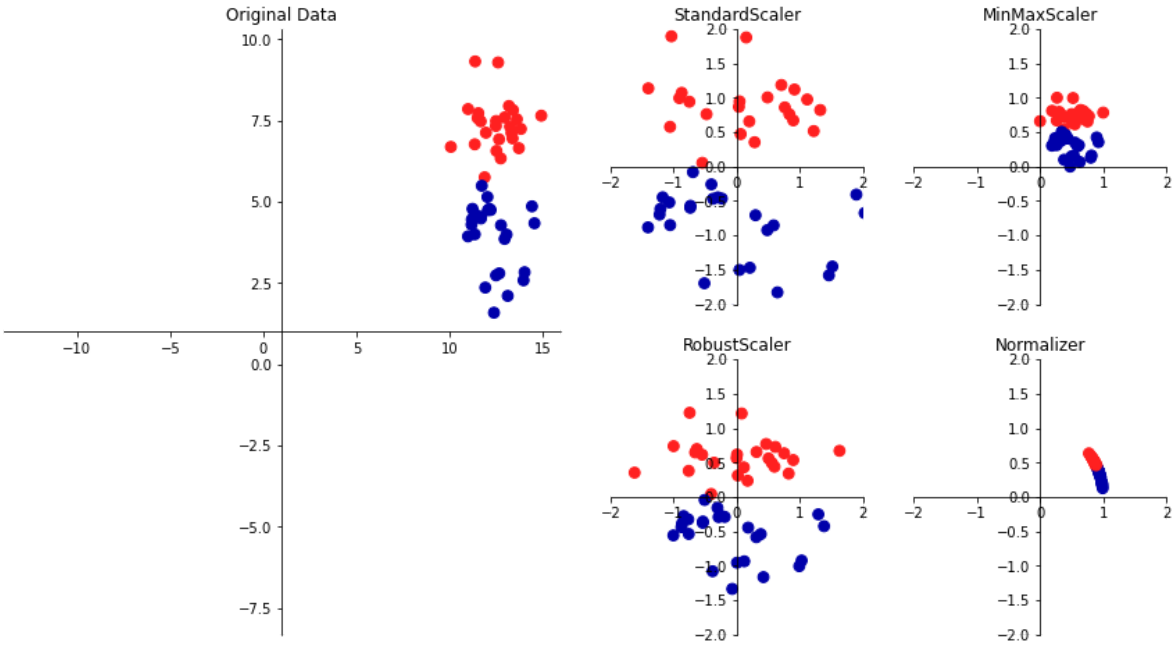
- 신경망과 서포트벡터 머신(SVM)같은 알고리즘은 데이터 스케일(범위)에 매우 민감하다.
- 이런 알고리즘에 맞게 특성(feature)를 조정한다. 각 feature별로 변경한다.

In [1]:

```
import mglearn
import warnings

warnings.filterwarnings('ignore')
```

In [2]:

```
mglearn.plots.plot_scaling()
```



In [3]:

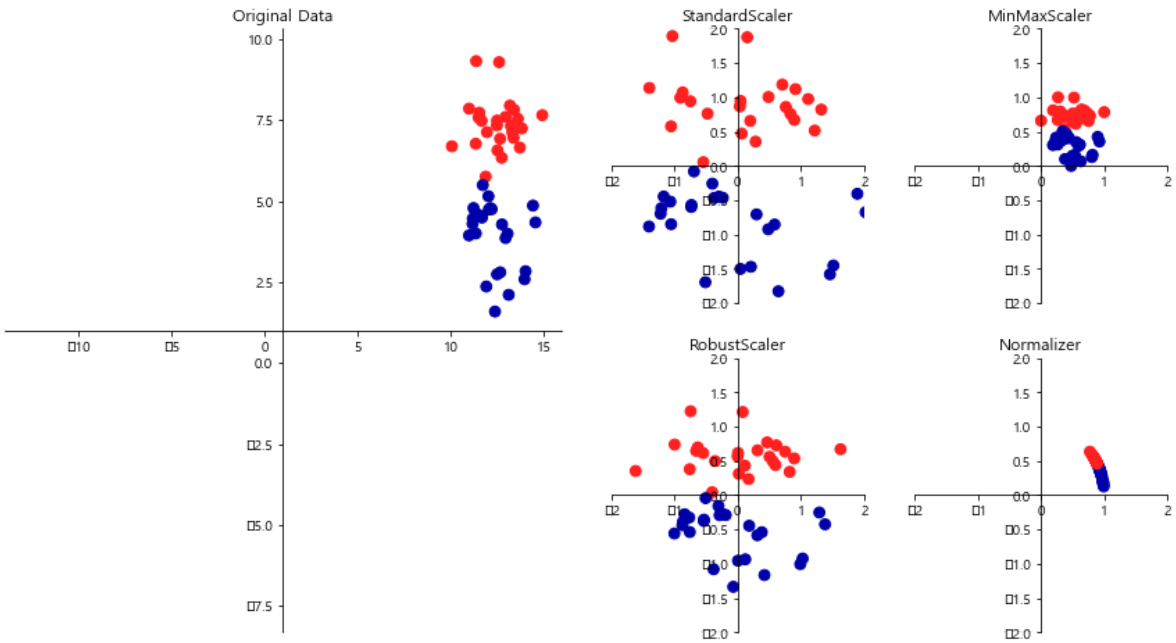
```
### 한글 폰트 설정
from matplotlib import font_manager, rc
import matplotlib.pyplot as plt
import platform
import matplotlib

path = "C:/Windows/Fonts/malgun.ttf"
if platform.system() == "Windows":
    font_name = font_manager.FontProperties(fname=path).get_name()
    rc('font', family=font_name)
elif platform.system()=="Darwin":
    rc('font', family='AppleGothic')
else:
    print("Unknown System")
```

In [4]:

```
import mglearn
```

In [5]:

```
mglearn.plots.plot_scaling()
```



(가) StandardScaler - 표준화

(가) 각 특성(feature)의 평균을 0, 분산을 1로 변경
(나) 이 방법은 feature(특성)의 최솟값과 최댓값 크기를 제한하지 않는다.

(나) RobustScaler

(가) 같은 스케일을 갖는다. StandardScaler과 비슷
(나) 이 방법은 평균과 분산 대신 중간 값(median)과 사분위 값(quantile)을 사용.
--> 중앙값을 선택하므로 전체 데이터와 아주 동떨어진 데이터 포인트에 영향을 받지 않음.

(다) MinMaxScaler - 정규화

(가) 모든 특성이 정확하게 0과 1사이에 위치하도록 데이터를 변경.

(라) Normalizer

(가) 유클리디안 길이가 1이 되도록 데이터 포인트를 조정.
(나) 지름이 1인 원에 데이터 포인트를 투영한다.
--> 정규화는 특성 벡터의 길이는 상관없고 데이터의 방향(또는 각도)만이 중요할 때 많이 사용.

04. 데이터 변환 실습 (cancer 데이터)

목차로 이동하기

데이터 전처리와 스케일 조정

- 데이터 셋 : cancer 데이터 (569개, 30열)

In [6]:

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

cancer = load_breast_cancer()
print("[cancer.keys()]  ₩n{}".format(cancer.keys()))
print("유방암 데이터의 형태 : {}".format(cancer.data.shape))
print("유방암 데이터의 피처명 : ", cancer.feature_names)
```

```
[cancer.keys()]
dict_keys(['data', 'target', 'frame', 'target_names', 'DESCR', 'feature_names', 'fil
ename'])
유방암 데이터의 형태 : (569, 30)
유방암 데이터의 피처명 :  ['mean radius' 'mean texture' 'mean perimeter' 'mean area'
 'mean smoothness' 'mean compactness' 'mean concavity'
 'mean concave points' 'mean symmetry' 'mean fractal dimension'
 'radius error' 'texture error' 'perimeter error' 'area error'
 'smoothness error' 'compactness error' 'concavity error'
 'concave points error' 'symmetry error' 'fractal dimension error'
 'worst radius' 'worst texture' 'worst perimeter' 'worst area'
 'worst smoothness' 'worst compactness' 'worst concavity'
 'worst concave points' 'worst symmetry' 'worst fractal dimension']
```

In [7]:

```
# 데이터 셋 나누기
X_train, X_test, y_train, y_test = train_test_split(cancer.data,
                                                    cancer.target,
                                                    stratify=cancer.target,
                                                    random_state=0)

print(X_train.shape , X_test.shape)
print(y_train.shape, y_test.shape)
```

```
(426, 30) (143, 30)
(426,) (143,)
```

MinMaxScaler(정규화)를 이용

In [8]:

```
from sklearn.preprocessing import MinMaxScaler
scaler = MinMaxScaler()
scaler.fit(X_train)
```

Out[8]:

```
MinMaxScaler()
```

In [9]:

```
import numpy as np
# np.set_printoptions(precision=3)
# 소숫점 3자리까지 표현
np.set_printoptions(formatter={'float_kind': lambda x: "{0:0.3f}".format(x)})
```

실제로 학습 데이터의 스케일을 조정하려면, 스케일 객체의 transform 메서드를 사용.

In [10]:

```
X_train_s = scaler.transform(X_train)
print("변환전 크기 : {}".format(X_train.shape))
print("변환전 값의 최소, 최대 : ₩n {}, {}".format(X_train.min(axis=0),
                                        X_train.max(axis=0)))
print()
print("변환 후 크기 : {}".format(X_train_s.shape))
print("변환후 값의 최소, 최대 : ₩n {}, {}".format(X_train_s.min(axis=0),
                                        X_train_s.max(axis=0)))
```

```
변환전 크기 : (426, 30)
변환전 값의 최소, 최대 :
 [7.691 9.710 47.920 170.400 0.053 0.019 0.000 0.000 0.106 0.050 0.112
 0.360 0.757 6.802 0.003 0.002 0.000 0.000 0.010 0.001 8.678 12.020 54.490
 223.600 0.081 0.034 0.000 0.000 0.157 0.055], [28.110 39.280 188.500 2501.000 0.142
0.345 0.375 0.191 0.304 0.097 2.873
 3.896 21.980 542.200 0.031 0.135 0.304 0.041 0.079 0.022 36.040 49.540
 251.200 4254.000 0.223 0.938 1.252 0.290 0.664 0.173]

변환 후 크기 : (426, 30)
변환후 값의 최소, 최대 :
 [0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000
 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000 0.000
 0.000 0.000 0.000 0.000 0.000 0.000], [1.000 1.000 1.000 1.000 1.000 1.000 1.000 1.
000 1.000 1.000 1.000 1.000
 1.000 1.000 1.000 1.000 1.000 1.000 1.000 1.000 1.000 1.000 1.000 1.000
 1.000 1.000 1.000 1.000 1.000 1.000]
```

변환전 후, 시각화

In [11]:

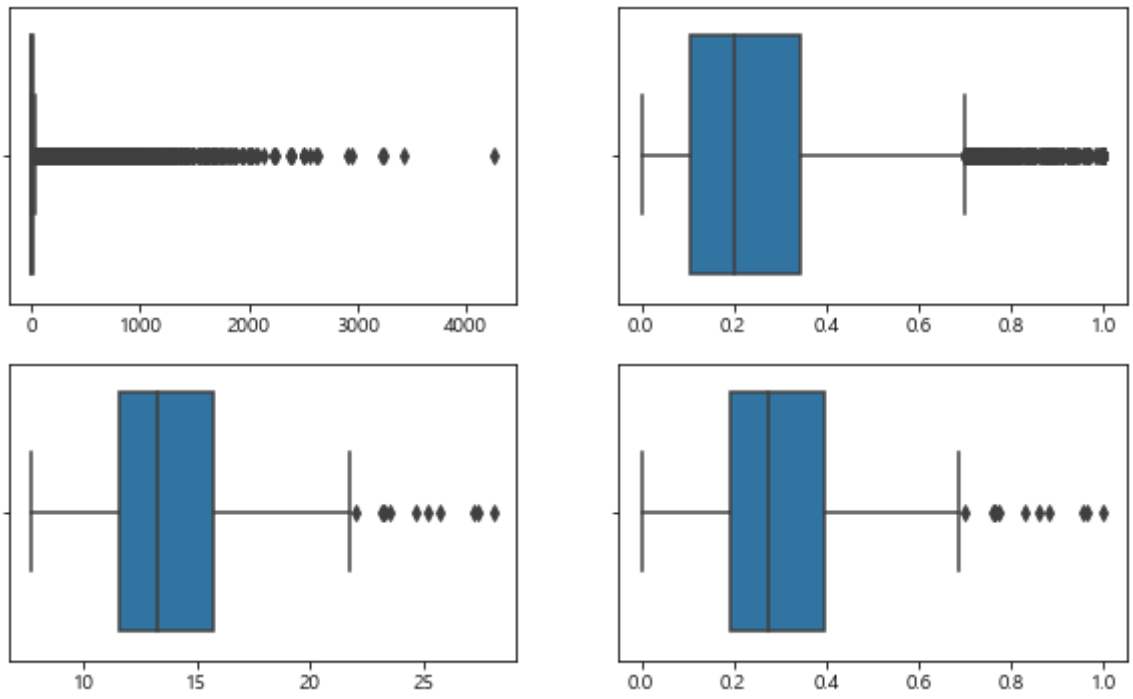
```
import seaborn as sns
import matplotlib.pyplot as plt
```

In [12]:

```
plt.figure(figsize=(10,6))
plt.subplot(2,2,1)
sns.boxplot(X_train)

plt.subplot(2,2,2)
sns.boxplot(X_train_s)

# 첫번째 열(mean radius)의 데이터 확인
plt.subplot(2,2,3)
sns.boxplot(X_train[:, 0])

plt.subplot(2,2,4)
sns.boxplot(X_train_s[:, 0])
```

Out[12]:

<AxesSubplot:>



메서드 단축해서 사용

- fit, transform를 -> fit_transform 메서드 사용하기

표준화 수행

- 특징의 값을 평균 0, 분산 1로 변경

In [13]:

```
from sklearn.preprocessing import StandardScaler
scaler = StandardScaler()

# 같은 내용 다른 방법
X_train_scaler = scaler.fit(X_train).transform(X_train)
X_train_scaler_d = scaler.fit_transform(X_train)

### fit().transform()과 fit_transform()과 같다.
print("변경전 :", X_train.max(), X_train.min() )
print("변경후 :", X_train_scaler.max(), X_train_scaler.min() )
print("변경후 :", X_train_scaler_d.max(), X_train_scaler_d.min() )
```

변경전 : 4254.0 0.0
변경후 : 10.57748988678996 -3.131829324965022
변경후 : 10.57748988678996 -3.131829324965022

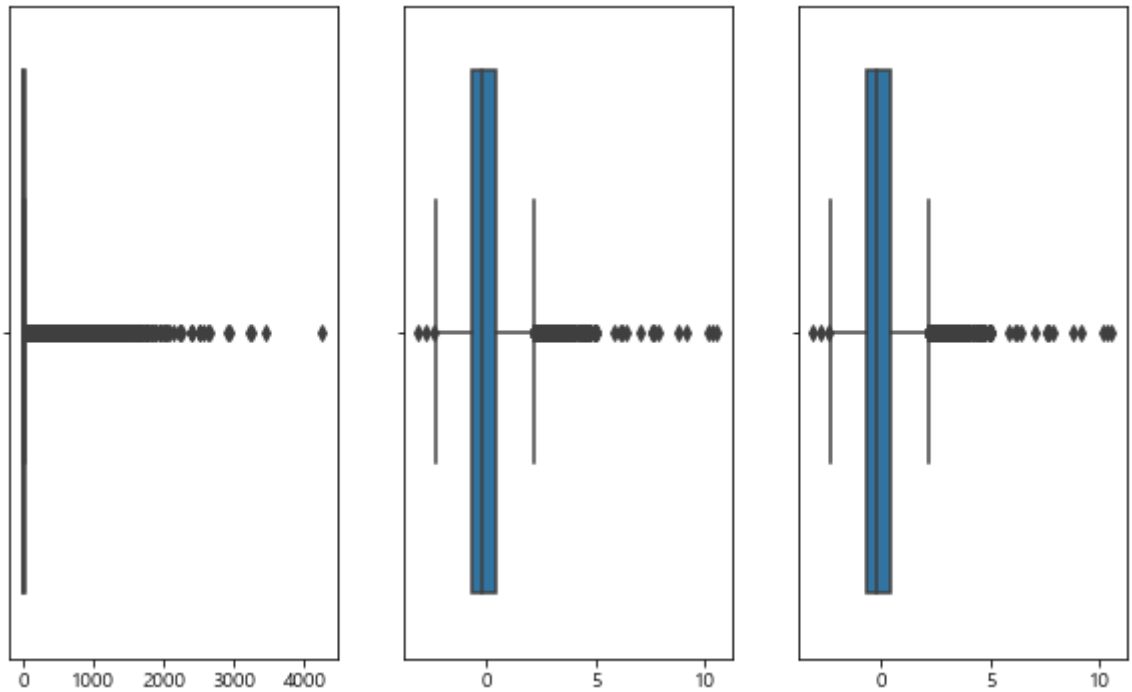
시각화 한 결과, 2번째, 3번째 결과는 같다.

In [14]:

```
fig, axes = plt.subplots(1, 3, figsize=(10,6))
sns.boxplot(X_train, ax=axes[0])
sns.boxplot(X_train_scaler, ax=axes[1])
sns.boxplot(X_train_scaler_d, ax=axes[2])
```

Out[14]:

<AxesSubplot:>



주의해야할 것

- 학습용 데이터 셋과 테스트 데이터 셋을 같은 스케일을 적용해야 함.

In [15]:

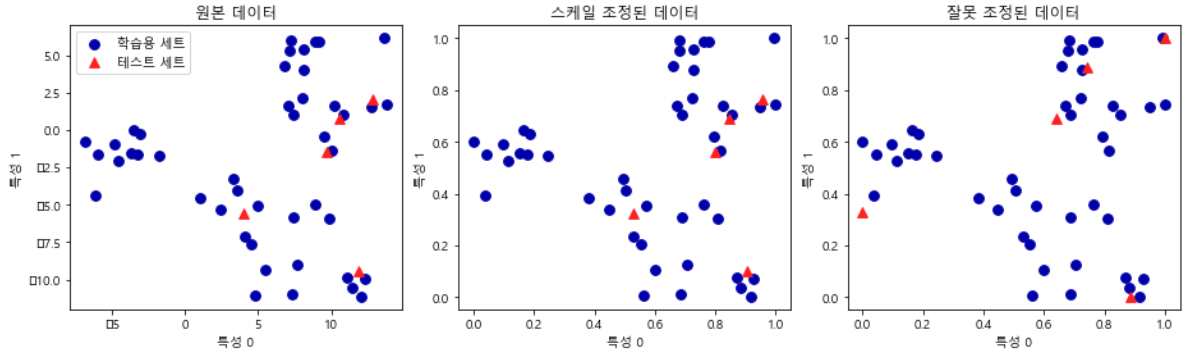
```
from sklearn.datasets import make_blobs

# 인위적인 데이터셋 생성
X, _ = make_blobs(n_samples=50, centers=5,
                  random_state=4, cluster_std=2)

# 학습용 세트와 테스트 세트로 나눕니다
X_train, X_test = train_test_split(X, random_state=5,
                                   test_size=.1)
```

In [16]:

```
# 학습용 세트와 테스트 세트의 산점도를 그립니다
fig, axes = plt.subplots(1, 3, figsize=(13, 4))
axes[0].scatter(X_train[:, 0], X_train[:, 1],
                c=mglearn.cm2(0), label="학습용 세트", s=60)
axes[0].scatter(X_test[:, 0], X_test[:, 1], marker='^',
                c=mglearn.cm2(1), label="테스트 세트", s=60)
axes[0].legend(loc='upper left')
axes[0].set_title("원본 데이터")

# MinMaxScaler를 사용해 스케일을 조정합니다
scaler = MinMaxScaler()
scaler.fit(X_train)
X_train_scaled = scaler.transform(X_train)
X_test_scaled = scaler.transform(X_test)

# 스케일이 조정된 데이터의 산점도를 그립니다
axes[1].scatter(X_train_scaled[:, 0], X_train_scaled[:, 1],
                c=mglearn.cm2(0), label="학습용 세트", s=60)
axes[1].scatter(X_test_scaled[:, 0], X_test_scaled[:, 1], marker='^',
                c=mglearn.cm2(1), label="테스트 세트", s=60)
axes[1].set_title("스케일 조정된 데이터")

# 테스트 세트의 스케일을 fit() 실행시, 따로 조정합니다
# 테스트 세트의 최솟값은 0, 최댓값은 1이 됩니다
# 이는 예제를 위한 것으로 절대로 이렇게 사용해서는 안됩니다
test_scaler = MinMaxScaler()
test_scaler.fit(X_test)
X_test_scaled_badly = test_scaler.transform(X_test)

# 잘못 조정된 데이터의 산점도를 그립니다
axes[2].scatter(X_train_scaled[:, 0], X_train_scaled[:, 1],
                c=mglearn.cm2(0), label="training set", s=60)
axes[2].scatter(X_test_scaled_badly[:, 0], X_test_scaled_badly[:, 1],
                marker='^', c=mglearn.cm2(1), label="test set", s=60)
axes[2].set_title("잘못 조정된 데이터")

for ax in axes:
    ax.set_xlabel("특성 0")
    ax.set_ylabel("특성 1")
fig.tight_layout()
```

```
*c* argument looks like a single numeric RGB or RGBA sequence, which should be av
oided as value-mapping will have precedence in case its length matches with *x* &
*y*.  Please use the *color* keyword-argument or provide a 2-D array with a singl
e row if you intend to specify the same RGB or RGBA value for all points.
*c* argument looks like a single numeric RGB or RGBA sequence, which should be av
oided as value-mapping will have precedence in case its length matches with *x* &
*y*.  Please use the *color* keyword-argument or provide a 2-D array with a singl
e row if you intend to specify the same RGB or RGBA value for all points.
*c* argument looks like a single numeric RGB or RGBA sequence, which should be av
oided as value-mapping will have precedence in case its length matches with *x* &
*y*.  Please use the *color* keyword-argument or provide a 2-D array with a singl
e row if you intend to specify the same RGB or RGBA value for all points.
*c* argument looks like a single numeric RGB or RGBA sequence, which should be av
oided as value-mapping will have precedence in case its length matches with *x* &
*y*.  Please use the *color* keyword-argument or provide a 2-D array with a singl
e row if you intend to specify the same RGB or RGBA value for all points.
```

```
*c* argument looks like a single numeric RGB or RGBA sequence, which should be av
oided as value-mapping will have precedence in case its length matches with *x* &
*y*.  Please use the *color* keyword-argument or provide a 2-D array with a singl
e row if you intend to specify the same RGB or RGBA value for all points.
*c* argument looks like a single numeric RGB or RGBA sequence, which should be av
oided as value-mapping will have precedence in case its length matches with *x* &
*y*.  Please use the *color* keyword-argument or provide a 2-D array with a singl
e row if you intend to specify the same RGB or RGBA value for all points.
```



- 첫번째 그래프 원본 데이터
- 두번째 그래프 정상
- 세번째 그래프 - 각각 train, test에 대해서 fit를 이용하여 적용시킨다.
  - MinMaxScaler()이 되면
  - scaler = MinMaxScaler()
  - test_scaler = MinMaxScaler()
    - scaler.fit(X_train)
    - test_scaler.fit(X_test)
  - 원본 데이터와 다르게 표현됨.