ch04 비선형 변환

학습 내용

- 01 비선형 변환
- 02 실습을 통해 확인해 보기
- 03 데이터를 리지 회귀(L1규제)에 적용

01 비선형 변환

- 제곱항이나 세제곱 항을 추가하면 선형 회귀 모델에 도움이 된다.
- log, exp, sin 같은 수학 함수를 적용하는 방법도 특성 변환에 유용.
- 선형 모델과 신경망은 각 특성의 스케일과 분포에 밀접하게 연관되어 있음.
- log, exp 함수는 데이터의 스케일을 변경하여 선형 모델과 신경망을 올리는데 도움이 된다.

In [1]:

```
import os, warnings
# 경고 메시지 무시하거나 숨길때(ignore), 다시보이게(default)
# warnings.filterwarnings(action='default')
warnings.filterwarnings(action='ignore')
```

In [2]:

```
# 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
```

02 실습을 통해 확인해 보기

In [3]:

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

In [10]:

```
rnd = np.random.RandomState(0)
X_org = rnd.normal(size=(1000,3))  # 정규분포를 따르는 임의의 행렬 1000,3 만들기

# np.random.normal()  # 정규분포를 따르는 임의의 값을 가져오겠다.
# np.random.uniform()  # 균등 분포를 따르는 임의의 값을 가져오겠다.
print(X_org.shape)

w = rnd.normal(size=3)
print(w.shape)
```

```
(1000, 3)
(3,)
```

In [11]:

```
# 포아송 분포를 따르는 값
X = rnd.poisson(10 * np.exp(X_org))
y = np.dot(X_org, w)
print("X, w : ", X.shape, w.shape)
print("y : ", y.shape)
print(X[:, 0])
```

```
X, w :  (1000, 3) (3,)
y :  (1000,)
[ 56  81  25  20  27  18  12  21 109   7  15   1  27   4   1   7   2  11
   6  18   1   2   2   7  31   7   7  28  37   9   8  21   7  20   3  71
  69  27   9   7  12  43   9  18   4  21  16  12 120  33   9   6  30   7
  25  27  40  15   2   5   7   3   5  42  10   7  87   5  19   4  24   8
  10  11   2   9   2   7   1   0   4   8  32   2  11  16   2   5   1   8
  13  19  13  68   2  21  20  12  10  16   5   7   1   3   1   9  10   3
   4 112  26   3   3  45   7  11  18   2   4   3  11  13  10   2  28  10
   8  14  14   3  23   3  24  28  18  17  13  27  56  10  23  11  24   5
  34  13  28   0  32   5   1  12   6  14   8  84   1   5  45   1  22   9
  47  14  29   9   2   9  10   1  25   2  17  60   0   6   0  12   0  10
  11  51  31   3  22  36   2  14   4  19  16  24   8  27   2   5   6  17
  11   6   1   9  15   5  25  15  21   8   8  53   6   4   4  38  28   5
   6   1  12   4   5  17   5  21   3  16  53  34   9   1  15   2   8  29
   1  84  36  26  43   5   8  13   1  14   4   1   6   5   3  19  14  59
  18   5  26   2  55   5   6  11  43   2  17   7   6   7   7  27  31  14
   6   4   4  13   3  12  44  21  11   4  19  93  13   7   4   4   3  12
  18   8  28   2   7   1  10   3  17  16   2   2   2  20   4  34  29  21
  15  39   9   3  52   5   2   6  18  14  15  15  15   6   7  31   7  18
  17   7  12  11   5  21  13  10  11   6   5  18  15  46   2   0  24   2
   3   6   3  28  14   6  43   3   2  34   2   3  12  15  19   2  16   5
   5   1   3   9   3   6  23   4   3  22   8   3   3  13   4  20   9   5
   7  10   0   2   6   6  17  44  56  45  27   7  15   7  21  18   6  13
  28  24  36   7   8  16  24  20   7  10   1   9   2   8  30  58  27   2
  18  16   8  19   4  27  11  69   2  75  45  16   2  10   2   3  14   2
  13   1  10   0  16   3   2   5   9   9  47  17  14   4   3   6  17   8
   5  88   7   5  10   8  12  18   8  20  35  45   4  27  16   4  31  17
   5  20  15   2  69   4  26  17   6   1   5   4  12  16   0  19  11   5
   1  22  12  28  28   5   5  13   4  10 140   2   0   6   4   7  10   2
   3  12   9   9   9  45   4  11   6   4  24  22   9  12   9  14  63  23
   5   3   0  18   3  24   4 125   2  27   6   5   5  17  11 102   8  48
   2  52   7  41  16   7   7  45  11  12  22  22  47  11   1  20   9  29
  18   7   9  85  10   7  94  20   8  10   2   4  31   9  13  21  47  11
  19  55  10   2   4   0   0   6  20  11  35   9  12  11  12  17  24  11
   2  14  11  19   5  14  19  48   5  17  33   2   3  13   6  12   5   7
  17   1   2   9  16   8  12  12   5   4  14   6   4   6  27   4   5  37
   0  18  59  25   5  15   7   0  35   2  51  16  10  29   8  43  25  23
  12 108   3   5  12  54  15   0  10   5   8   9   4   5  16  15  45  16
   4   1   7  11  19  14  16  15   7   6  13  13  54  50   0   3   3  26
   2  11  13  18   0  42  38  12   9  27   3   1  64   8  33   1  10  11
  15   1  14   4   6  20   2  14   0   3  19   8   3   0  16  10  12   4
   2  29   7  13  16  28   4   6   7   2  10   1   9  25  33  13   6  46
   6   7  44   5  23   2   2  20   6  19   1  15  24  22  10  21   2  24
  17   9  22  47   4   2   5  11  13  15   4   9  19   7  16   8  10   7
   0  20  14  12  13   1  15  26  34  13  25  21  18   2   6   7  41   6
   5  51   5   4   9   7   2  20  12  30   5  23  21  73  21   8   7  19
  12  17   4  16  20   7  15  50   5   4   4   0   4  11   2  12  19  49
   2   6   4  38  23   0  63   2   1   4   9  22  13  15  24  19   2  17
  13   8  85  31   0  10  18  21  36   8   2  16   5  16   5  29   0  63
  20   9   5   3   5  18   2  32  11  12   4   9  76   2  20   4  24  11
```

```
 26   3  24   8  10   6  11   1   8  10  24   2   1  21   6  18  16  19
  0   7  15  19  12   7  23  13   1   1  19   6  15   5   5  28   5   7
 11   1  17  38   3  19  15   2  12  12   3  21   9   4   2  14   7  21
 10  73   4   3  11  10   8  41   4   7   3   6  11  18  24  32  17   3
  3  14  33   8   6   4   4   9   3   3  31  55   7   4   7   8   4  25
  4  52   8   4  10  18   8   4  12  21   0  16   9  21   5  14  12   9
  5   6  19  10   0   7   5   2   7   7]
```

In [13]:

```
### 각 값이 가지는 것에 대해 확인해 보기
print("값 출현 횟수(0~) :₩n",np.bincount(X[:,0]))
```

```
값 출현 횟수(0~) :
 [28 38 68 48 61 59 45 56 37 40 35 34 36 26 23 26 27 21 23 23 18 21 10  9
  17  9  7 14 12  7  3  8  4  5  5  3  4  2  4  1  1  3  2  5  3  8  2  5
   2  1  2  3  3  2  2  3  3  0  1  2  1  0  0  3  1  0  0  0  1  3  0  1
   0  2  0  1  1  0  0  0  0  1  0  0  2  2  0  1  1  0  0  0  0  1  1  0
   0  0  0  0  0  0  1  0  0  0  0  0  1  1  0  0  1  0  0  0  0  0  0  0
   1  0  0  0  0  1  0  0  0  0  0  0  0  0  0  0  0  0  0  0  1]
```

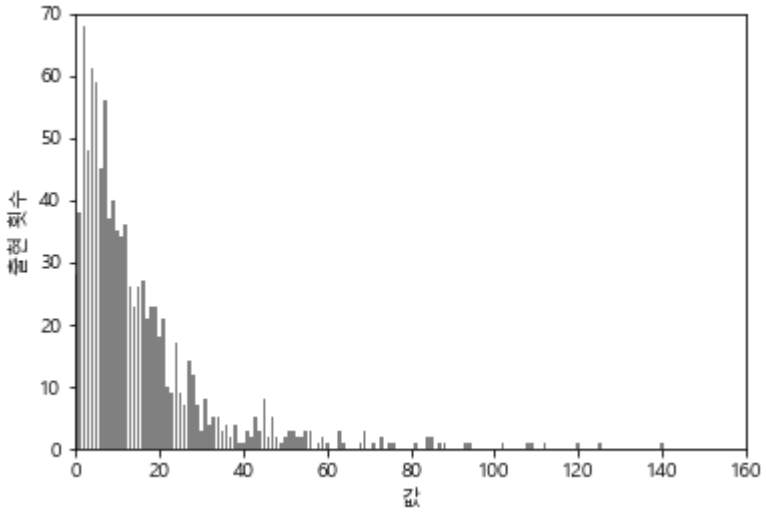
- 2가 68번으로 가장 많이 나타나며. 큰 값의 수는 빠르게 줄어든다.
- 85, 86처럼 아주 큰 값도 약간은 있음

In [14]:

```
plt.xlim(0, 160)
plt.ylim(0, 70)
bins = np.bincount(X[:, 0])
plt.bar(range(len(bins)), bins, color='grey')
plt.ylabel("출현 횟수")
plt.xlabel("값")
```

Out[14]:

```
Text(0.5, 0, '값')
```



In [15]:

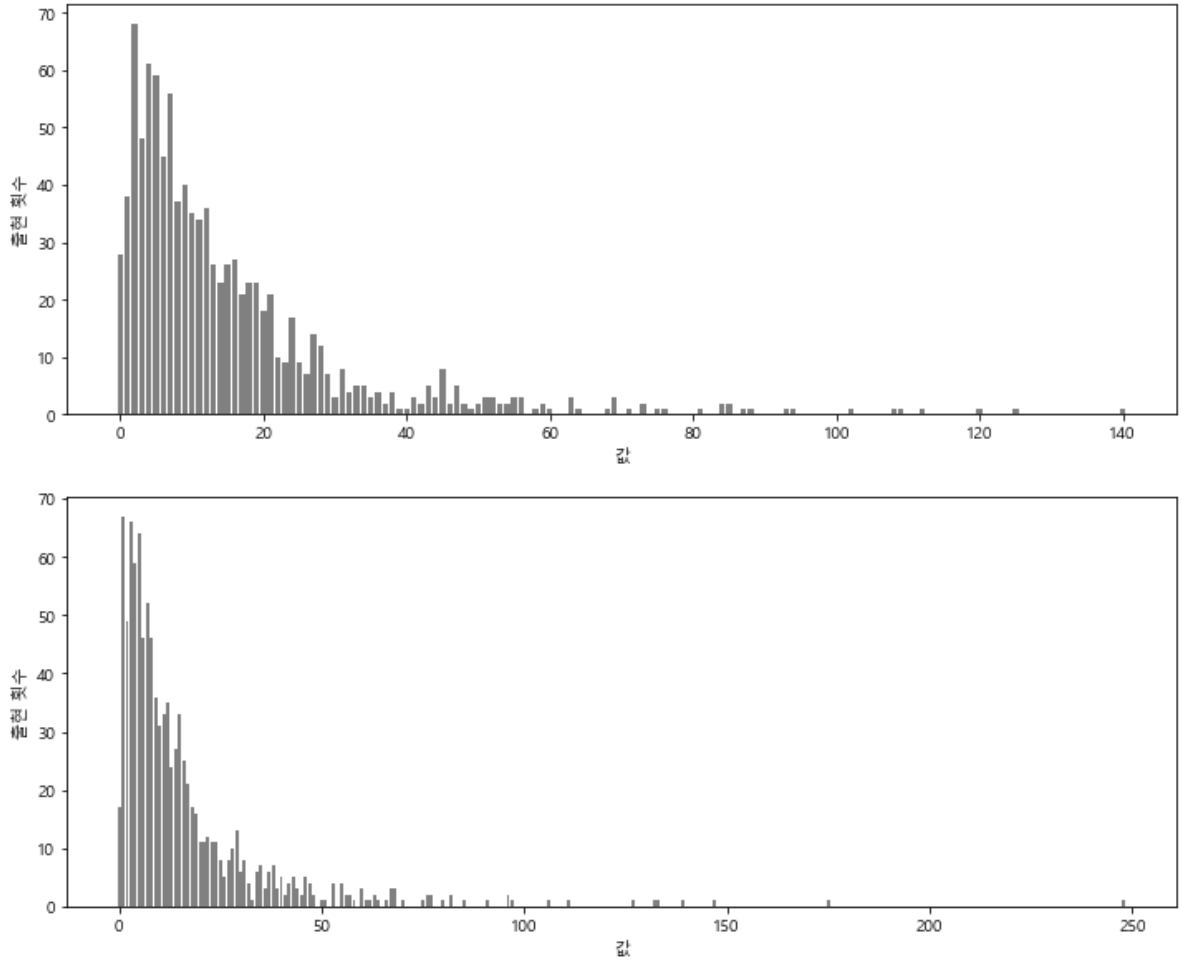
```
plt.figure(figsize=(12, 10))

plt.subplot(2,1,1)
bins = np.bincount(X[:, 0])
plt.bar(range(len(bins)), bins, color='grey')
plt.ylabel("출현 횟수")
plt.xlabel("값")

plt.subplot(2,1,2)
bins = np.bincount(X[:, 1])
plt.bar(range(len(bins)), bins, color='grey')
plt.ylabel("출현 횟수")
plt.xlabel("값")
```

Out[15]:

```
Text(0.5, 0, '값')
```



- X[:, 1]과 X[:, 2]의 특성도 비슷하다.

03 주어진 데이터 활용한 리지 회귀(L1규제) 구현

In [17]:

```
from sklearn.linear_model import Ridge
from sklearn.model_selection import train_test_split
```

In [18]:

```
X.shape, y.shape
```

Out[18]:

```
((1000, 3), (1000,))
```

In [20]:

```
X_train, X_test, y_train, y_test = train_test_split(X, y, random_state=0)

print(X_train.shape, X_test.shape, y_train.shape, y_test.shape)
score = Ridge().fit(X_train, y_train).score(X_test, y_test)
print("테스트 점수 : {:.3f}".format(score))
```

```
(750, 3) (250, 3) (750,) (250,)
테스트 점수 : 0.622
```

In [21]:

```
X_train_log = np.log(X_train + 1)
X_test_log = np.log(X_test + 1)
```

In [25]:

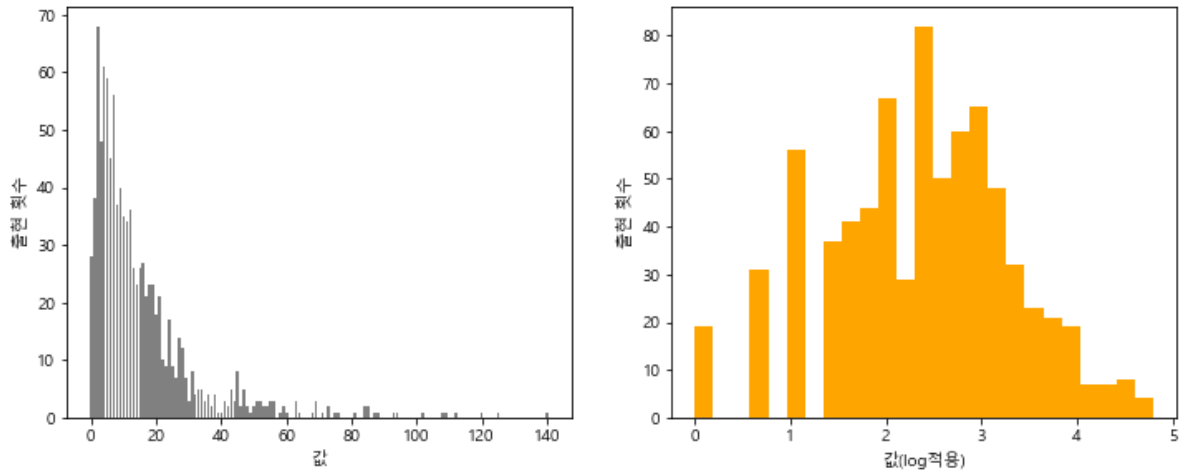
```
plt.figure(figsize=(12, 10))

plt.subplot(2,2,1)
bins = np.bincount(X[:, 0])
plt.bar(range(len(bins)), bins, color='grey')
plt.ylabel("출현 횟수")
plt.xlabel("값")

plt.subplot(2,2,2)
plt.hist( X_train_log[:, 0], bins=25, color='orange')
plt.ylabel("출현 횟수")
plt.xlabel("값(log적용)")
```

Out[25]:

```
Text(0.5, 0, '값(log적용)')
```



In [26]:

```
score = Ridge().fit(X_train_log, y_train).score(X_test_log, y_test)
print("테스트 점수 : {:.3f}".format(score))
```

```
테스트 점수 : 0.875
```

- 결정계수의 값이 0.622에서 0.875로 향상되었다.
- 선형모델, 나이브 베이즈 모델 같은 덜 복잡한 모델에서 구간 분할, 다항식, 상호작용은 데이터가 주어진 상황에서 모델의 성능에 큰 영향을 줄 수 있다.