원-핫-인코딩 (가변수)

In [4]:

```
import pandas as pd
import mglearn
```

```
C:₩Anaconda₩lib₩site-packages₩sklearn₩externals₩six.py:31: DeprecationWarning: The
module is deprecated in version 0.21 and will be removed in version 0.23 since w
e've dropped support for Python 2.7. Please rely on the official version of six (h
ttps://pypi.org/project/six/).
  "(https://pypi.org/project/six/).", DeprecationWarning)
C:₩Anaconda₩lib₩site-packages₩sklearn₩externals₩joblib₩__init__.py:15: Deprecation
Warning: sklearn.externals.joblib is deprecated in 0.21 and will be removed in 0.2
3. Please import this functionality directly from joblib, which can be installed w
ith: pip install joblib. If this warning is raised when loading pickled models, yo
u may need to re-serialize those models with scikit-learn 0.21+.
  warnings.warn(msg, category=DeprecationWarning)
```

In [5]:

```
import os

data = pd.read_csv(
    os.path.join(mglearn.datasets.DATA_PATH, "adult.data"), header=None, index_col=False,
    names=['age', 'workclass', 'fnlwgt', 'education',  'education-num',
           'marital-status', 'occupation', 'relationship', 'race', 'gender',
           'capital-gain', 'capital-loss', 'hours-per-week', 'native-country',
           'income'])

data = data[['age', 'workclass', 'education', 'gender', 'hours-per-week',
             'occupation', 'income']]

display(data.head())
```

	age	workclass	education	gender	hours-per-week	occupation	income
0	39	State-gov	Bachelors	Male	40	Adm-clerical	<=50K
1	50	Self-emp-not-inc	Bachelors	Male	13	Exec-managerial	<=50K
2	38	Private	HS-grad	Male	40	Handlers-cleaners	<=50K
3	53	Private	11th	Male	40	Handlers-cleaners	<=50K
4	28	Private	Bachelors	Female	40	Prof-specialty	<=50K

문자열로 된 범주형 데이터 확인하기

In [6]:

```
print(data.gender.value_counts())
```

```
 Male      21790
 Female    10771
Name: gender, dtype: int64
```

In [7]:

```
print("원본 특성:₩n", list(data.columns), "₩n")
data_dummies = pd.get_dummies(data)
print("get_dummies 후의 특성:₩n", list(data_dummies.columns))
```

```
원본 특성:
 ['age', 'workclass', 'education', 'gender', 'hours-per-week', 'occupation', 'inco
me']

get_dummies 후의 특성:
 ['age', 'hours-per-week', 'workclass_ ?', 'workclass_ Federal-gov', 'workclass_ L
ocal-gov', 'workclass_ Never-worked', 'workclass_ Private', 'workclass_ Self-emp-i
nc', 'workclass_ Self-emp-not-inc', 'workclass_ State-gov', 'workclass_ Without-pa
y', 'education_ 10th', 'education_ 11th', 'education_ 12th', 'education_ 1st-4th',
'education_ 5th-6th', 'education_ 7th-8th', 'education_ 9th', 'education_ Assoc-ac
dm', 'education_ Assoc-voc', 'education_ Bachelors', 'education_ Doctorate', 'educ
ation_ HS-grad', 'education_ Masters', 'education_ Preschool', 'education_ Prof-sc
hool', 'education_ Some-college', 'gender_ Female', 'gender_ Male', 'occupation_
?', 'occupation_ Adm-clerical', 'occupation_ Armed-Forces', 'occupation_ Craft-rep
air', 'occupation_ Exec-managerial', 'occupation_ Farming-fishing', 'occupation_ H
andlers-cleaners', 'occupation_ Machine-op-inspct', 'occupation_ Other-service',
'occupation_ Priv-house-serv', 'occupation_ Prof-specialty', 'occupation_ Protecti
ve-serv', 'occupation_ Sales', 'occupation_ Tech-support', 'occupation_ Transport-
moving', 'income_ <=50K', 'income_ >50K']
```

In [8]:

```
display(data_dummies.head())
```

	age	hours-per-week	workclass_ ?	workclass_ Federal-gov	workclass_ Local-gov	workclass_ Never-worked	workclass_ Private	workclass_ Self-emp-inc
0	39	40	0	0	0	0	0	0
1	50	13	0	0	0	0	0	0
2	38	40	0	0	0	0	1	0
3	53	40	0	0	0	0	1	0
4	28	40	0	0	0	0	1	0

5 rows × 46 columns

In [9]:

```
features = data_dummies.loc[:, 'age':'occupation_ Transport-moving']

X = features.values
y = data_dummies['income_ >50K'].values
print("X.shape: {}  y.shape: {}".format(X.shape, y.shape))
```

```
X.shape: (32561, 44)  y.shape: (32561,)
```

In [10]:

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

X_train, X_test, y_train, y_test = train_test_split(X, y, random_state=0)
logreg = LogisticRegression()
logreg.fit(X_train, y_train)

print("테스트 점수: {:.2f}".format(logreg.score(X_test, y_test)))
```

```
테스트 점수: 0.81

C:\Anaconda\lib\site-packages\sklearn\linear_model\logistic.py:432: FutureWarning:
Default solver will be changed to 'lbfgs' in 0.22. Specify a solver to silence thi
s warning.
  FutureWarning)
```

숫자로 표현된 범주형 특성

In [11]:

```
demo_df = pd.DataFrame({'숫자 특성': [0, 1, 2, 1], '범주형 특성': ['양말', '여우', '양말', '상
자']})

display(demo_df)
```

	숫자 특성	범주형 특성
0	0	양말
1	1	여우
2	2	양말
3	1	상자

In [12]:

```
display(pd.get_dummies(demo_df))
```

	숫자 특성	범주형 특성_상자	범주형 특성_양말	범주형 특성_여우
0	0	0	1	0
1	1	0	0	1
2	2	0	1	0
3	1	1	0	0

In [13]:

```
demo_df['숫자 특성'] = demo_df['숫자 특성'].astype(str)

display(pd.get_dummies(demo_df, columns=['숫자 특성', '범주형 특성']))
```

	숫자 특성_0	숫자 특성_1	숫자 특성_2	범주형 특성_상자	범주형 특성_양말	범주형 특성_여우
0	1	0	0	0	1	0
1	0	1	0	0	0	1
2	0	0	1	0	1	0
3	0	1	0	1	0	0

구간 분할, 이산화 그리고 선형 모델, 트리 모델

In [17]:

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

In [19]:

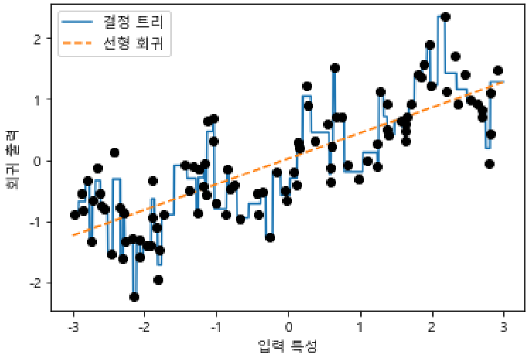
```
from matplotlib import font_manager, rc

font_name = font_manager.FontProperties(fname="C:/Windows/Fonts/malgun.ttf").get_name()
rc('font', family=font_name)

plt.rcParams['axes.unicode_minus'] = False
```

In [20]:

```
from sklearn.linear_model import LinearRegression
from sklearn.tree import DecisionTreeRegressor

X, y = mglearn.datasets.make_wave(n_samples=100)
line = np.linspace(-3, 3, 1000, endpoint=False).reshape(-1, 1)

reg = DecisionTreeRegressor(min_samples_split=3).fit(X, y)
plt.plot(line, reg.predict(line), label="결정 트리")

reg = LinearRegression().fit(X, y)
plt.plot(line, reg.predict(line), '--', label="선형 회귀")

plt.plot(X[:, 0], y, 'o', c='k')
plt.ylabel("회귀 출력")
plt.xlabel("입력 특성")
plt.legend(loc="best")
```

Out[20]:

```
<matplotlib.legend.Legend at 0x17a4eef9ef0>
```



In [21]:

```
bins = np.linspace(-3, 3, 11)

print("bins: {}".format(bins))
```

```
bins: [-3.  -2.4 -1.8 -1.2 -0.6  0.   0.6  1.2  1.8  2.4  3. ]
```

In [22]:

```
which_bin = np.digitize(X, bins=bins)

print("\n데이터 포인트:\n", X[:5])
print("\n데이터 포인트의 소속 구간:\n", which_bin[:5])
```

```
데이터 포인트:
 [[-0.75275929]
 [ 2.70428584]
 [ 1.39196365]
 [ 0.59195091]
 [-2.06388816]]

데이터 포인트의 소속 구간:
 [[ 4]
 [10]
 [ 8]
 [ 6]
 [ 2]]
```

In [23]:

```
from sklearn.preprocessing import OneHotEncoder

encoder = OneHotEncoder(sparse=False)
encoder.fit(which_bin)
X_binned = encoder.transform(which_bin)

print(X_binned[:5])
```

```
[[0. 0. 0. 1. 0. 0. 0. 0. 0. 0.]
 [0. 0. 0. 0. 0. 0. 0. 0. 0. 1.]
 [0. 0. 0. 0. 0. 0. 0. 1. 0. 0.]
 [0. 0. 0. 0. 0. 1. 0. 0. 0. 0.]
 [0. 1. 0. 0. 0. 0. 0. 0. 0. 0.]]

C:\Anaconda\lib\site-packages\sklearn\preprocessing\_encoders.py:415: FutureWarnin
g: The handling of integer data will change in version 0.22. Currently, the catego
ries are determined based on the range [0, max(values)], while in the future they
will be determined based on the unique values.
If you want the future behaviour and silence this warning, you can specify "catego
ries='auto'".
In case you used a LabelEncoder before this OneHotEncoder to convert the categorie
s to integers, then you can now use the OneHotEncoder directly.
  warnings.warn(msg, FutureWarning)
```

In [24]:

```
print("X_binned.shape: {}".format(X_binned.shape))
```

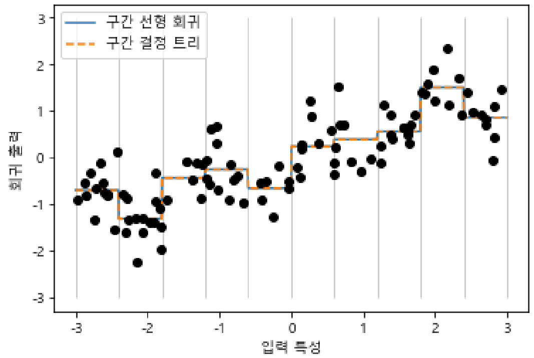
```
X_binned.shape: (100, 10)
```

In [25]:

```
line_binned = encoder.transform(np.digitize(line, bins=bins))

reg = LinearRegression().fit(X_binned, y)
plt.plot(line, reg.predict(line_binned), label='구간 선형 회귀')

reg = DecisionTreeRegressor(min_samples_split=3).fit(X_binned, y)
plt.plot(line, reg.predict(line_binned), '--', label='구간 결정 트리')
plt.plot(X[:, 0], y, 'o', c='k')
plt.vlines(bins, -3, 3, linewidth=1, alpha=.2)
plt.legend(loc="best")
plt.ylabel("회귀 출력")
plt.xlabel("입력 특성")
```

Out[25]:

```
Text(0.5, 0, '입력 특성')
```



교차항과 고차항

In [26]:

```
X_combined = np.hstack([X, X_binned])
print(X_combined.shape)
```

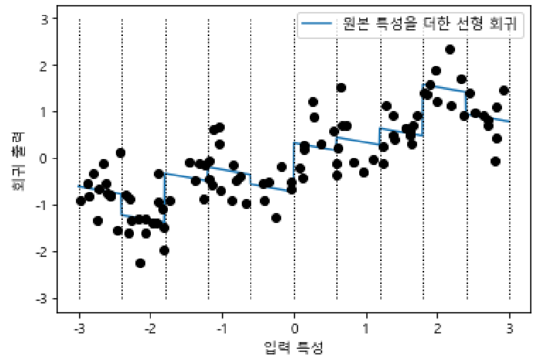
```
(100, 11)
```

In [27]:

```
reg = LinearRegression().fit(X_combined, y)

line_combined = np.hstack([line, line_binned])
plt.plot(line, reg.predict(line_combined), label='원본 특성을 더한 선형 회귀')

for bin in bins:
    plt.plot([bin, bin], [-3, 3], ':', c='k', linewidth=1)
plt.legend(loc="best")
plt.ylabel("회귀 출력")
plt.xlabel("입력 특성")
plt.plot(X[:, 0], y, 'o', c='k')
```

Out[27]:

```
[<matplotlib.lines.Line2D at 0x17a4f2b5828>]
```



In [28]:

```
X_product = np.hstack([X_binned, X * X_binned])
print(X_product.shape)
```

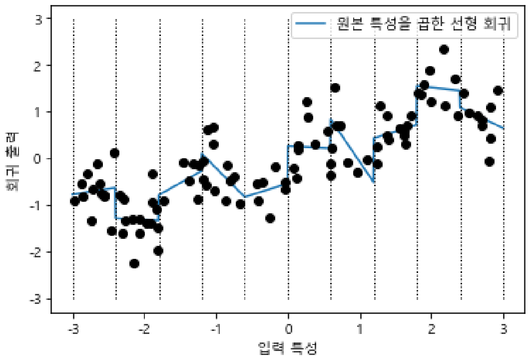
```
(100, 20)
```

In [29]:

```
reg = LinearRegression().fit(X_product, y)

line_product = np.hstack([line_binned, line * line_binned])
plt.plot(line, reg.predict(line_product), label='원본 특성을 곱한 선형 회귀')

for bin in bins:
    plt.plot([bin, bin], [-3, 3], ':', c='k', linewidth=1)

plt.plot(X[:, 0], y, 'o', c='k')
plt.ylabel("회귀 출력")
plt.xlabel("입력 특성")
plt.legend(loc="best")
```

Out[29]:

```
<matplotlib.legend.Legend at 0x17a4f3b2b38>
```



In [30]:

```
from sklearn.preprocessing import PolynomialFeatures

poly = PolynomialFeatures(degree=10, include_bias=False)
poly.fit(X)
X_poly = poly.transform(X)
```

In [31]:

```
print("X_poly.shape: {}".format(X_poly.shape))
```

```
X_poly.shape: (100, 10)
```

In [32]:

```
print("X 원소:₩n{}".format(X[:5]))
print("X_poly 원소:₩n{}".format(X_poly[:5]))
```

```
X 원소:
[[-0.75275929]
 [ 2.70428584]
 [ 1.39196365]
 [ 0.59195091]
 [-2.06388816]]
X_poly 원소:
[[-7.52759287e-01  5.66646544e-01 -4.26548448e-01  3.21088306e-01
  -2.41702204e-01  1.81943579e-01 -1.36959719e-01  1.03097700e-01
  -7.76077513e-02  5.84199555e-02]
 [ 2.70428584e+00  7.31316190e+00  1.97768801e+01  5.34823369e+01
   1.44631526e+02  3.91124988e+02  1.05771377e+03  2.86036036e+03
   7.73523202e+03  2.09182784e+04]
 [ 1.39196365e+00  1.93756281e+00  2.69701700e+00  3.75414962e+00
   5.22563982e+00  7.27390068e+00  1.01250053e+01  1.40936394e+01
   1.96178338e+01  2.73073115e+01]
 [ 5.91950905e-01  3.50405874e-01  2.07423074e-01  1.22784277e-01
   7.26822637e-02  4.30243318e-02  2.54682921e-02  1.50759786e-02
   8.92423917e-03  5.28271146e-03]
 [-2.06388816e+00  4.25963433e+00 -8.79140884e+00  1.81444846e+01
  -3.74481869e+01  7.72888694e+01 -1.59515582e+02  3.29222321e+02
  -6.79478050e+02  1.40236670e+03]]
```

In [33]:

```
print("항 이름:₩n{}".format(poly.get_feature_names()))
```

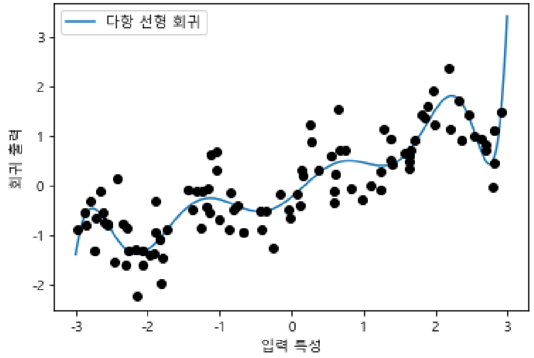
```
항 이름:
['x0', 'x0^2', 'x0^3', 'x0^4', 'x0^5', 'x0^6', 'x0^7', 'x0^8', 'x0^9', 'x0^10']
```

In [34]:

```
reg = LinearRegression().fit(X_poly, y)

line_poly = poly.transform(line)

plt.plot(line, reg.predict(line_poly), label='다항 선형 회귀')
plt.plot(X[:, 0], y, 'o', c='k')
plt.ylabel("회귀 출력")
plt.xlabel("입력 특성")
plt.legend(loc="best")
```

Out[34]:

<matplotlib.legend.Legend at 0x17a4f474ac8>

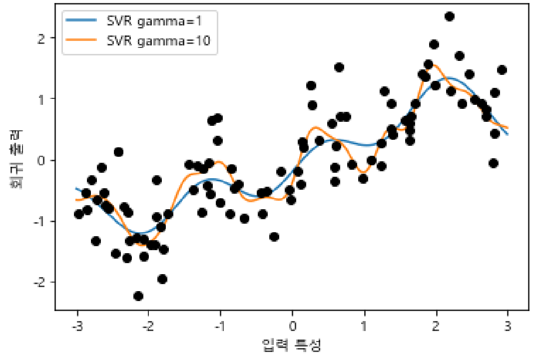


In [35]:

```
from sklearn.svm import SVR

for gamma in [1, 10]:
    svr = SVR(gamma=gamma).fit(X, y)
    plt.plot(line, svr.predict(line), label='SVR gamma={}'.format(gamma))

plt.plot(X[:, 0], y, 'o', c='k')
plt.ylabel("회귀 출력")
plt.xlabel("입력 특성")
plt.legend(loc="best")
```

Out[35]:

<matplotlib.legend.Legend at 0x17a4f4dff60>



In [36]:

```
from sklearn.datasets import load_boston
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import MinMaxScaler

boston = load_boston()
X_train, X_test, y_train, y_test = train_test_split(boston.data, boston.target, random_state=0)
scaler = MinMaxScaler()
X_train_scaled = scaler.fit_transform(X_train)
X_test_scaled = scaler.transform(X_test)
```

In [37]:

```
poly = PolynomialFeatures(degree=2).fit(X_train_scaled)
X_train_poly = poly.transform(X_train_scaled)
X_test_poly = poly.transform(X_test_scaled)

print("X_train.shape: {}".format(X_train.shape))
print("X_train_poly.shape: {}".format(X_train_poly.shape))
```

```
X_train.shape: (379, 13)
X_train_poly.shape: (379, 105)
```

In [38]:

```
print("다항 특성 이름:₩n{}".format(poly.get_feature_names()))
```

```
다항 특성 이름:
['1', 'x0', 'x1', 'x2', 'x3', 'x4', 'x5', 'x6', 'x7', 'x8', 'x9', 'x10', 'x11', 'x
12', 'x0^2', 'x0 x1', 'x0 x2', 'x0 x3', 'x0 x4', 'x0 x5', 'x0 x6', 'x0 x7', 'x0 x
8', 'x0 x9', 'x0 x10', 'x0 x11', 'x0 x12', 'x1^2', 'x1 x2', 'x1 x3', 'x1 x4', 'x1
x5', 'x1 x6', 'x1 x7', 'x1 x8', 'x1 x9', 'x1 x10', 'x1 x11', 'x1 x12', 'x2^2', 'x2
x3', 'x2 x4', 'x2 x5', 'x2 x6', 'x2 x7', 'x2 x8', 'x2 x9', 'x2 x10', 'x2 x11', 'x2
x12', 'x3^2', 'x3 x4', 'x3 x5', 'x3 x6', 'x3 x7', 'x3 x8', 'x3 x9', 'x3 x10', 'x3
x11', 'x3 x12', 'x4^2', 'x4 x5', 'x4 x6', 'x4 x7', 'x4 x8', 'x4 x9', 'x4 x10', 'x4
x11', 'x4 x12', 'x5^2', 'x5 x6', 'x5 x7', 'x5 x8', 'x5 x9', 'x5 x10', 'x5 x11', 'x
5 x12', 'x6^2', 'x6 x7', 'x6 x8', 'x6 x9', 'x6 x10', 'x6 x11', 'x6 x12', 'x7^2',
'x7 x8', 'x7 x9', 'x7 x10', 'x7 x11', 'x7 x12', 'x8^2', 'x8 x9', 'x8 x10', 'x8 x1
1', 'x8 x12', 'x9^2', 'x9 x10', 'x9 x11', 'x9 x12', 'x10^2', 'x10 x11', 'x10 x12',
'x11^2', 'x11 x12', 'x12^2']
```

In [39]:

```
from sklearn.linear_model import Ridge

ridge = Ridge().fit(X_train_scaled, y_train)

print("상호작용 특성이 없을 때 점수: {:.3f}".format(ridge.score(X_test_scaled, y_test)))

ridge = Ridge().fit(X_train_poly, y_train)

print("상호작용 특성이 있을 때 점수: {:.3f}".format(ridge.score(X_test_poly, y_test)))
```

```
상호작용 특성이 없을 때 점수: 0.621
상호작용 특성이 있을 때 점수: 0.753
```

In [40]:

```
from sklearn.ensemble import RandomForestRegressor

rf = RandomForestRegressor(n_estimators=100, random_state=0).fit(X_train_scaled, y_train)

print("상호작용 특성이 없을 때 점수: {:.3f}".format(rf.score(X_test_scaled, y_test)))

rf = RandomForestRegressor(n_estimators=100, random_state=0).fit(X_train_poly, y_train)

print("상호작용 특성이 있을 때 점수: {:.3f}".format(rf.score(X_test_poly, y_test)))
```

```
상호작용 특성이 없을 때 점수: 0.795
상호작용 특성이 있을 때 점수: 0.774
```

단변량 비선형 변환

In [41]:

```
rnd = np.random.RandomState(0)
X_org = rnd.normal(size=(1000, 3))
w = rnd.normal(size=3)
X = rnd.poisson(10 * np.exp(X_org))
y = np.dot(X_org, w)

print(X[:10, 0])
```

```
[ 56  81  25  20  27  18  12  21 109   7]
```

In [42]:

```
print("특성 출현 횟수:₩n{}".format(np.bincount(X[:, 0].astype('int'))))
```

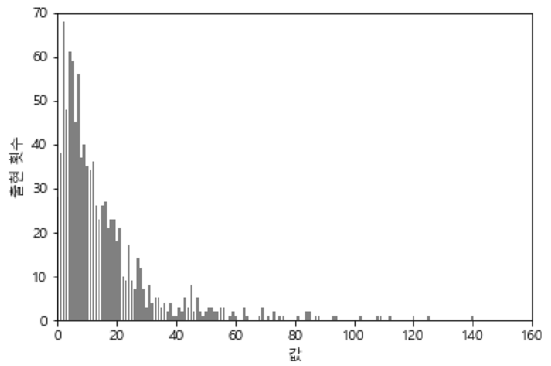
```
특성 출현 횟수:
[28 38 68 48 61 59 45 56 37 40 35 34 36 26 23 26 27 21 23 23 18 21 10  9
 17  9  7 14 12  7  3  8  4  5  5  3  4  2  4  1  1  3  2  5  3  8  2  5
  2  1  2  3  3  2  2  3  3  0  1  2  1  0  0  3  1  0  0  0  1  3  0  1
  0  2  0  1  1  0  0  0  0  1  0  0  2  2  0  1  1  0  0  0  0  1 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

In [43]:

```
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[43]:

```
Text(0.5, 0, '값')
```



In [44]:

```
from sklearn.linear_model import Ridge

X_train, X_test, y_train, y_test = train_test_split(X, y, random_state=0)
score = Ridge().fit(X_train, y_train).score(X_test, y_test)

print("테스트 점수: {:.3f}".format(score))
```

테스트 점수: 0.622

In [45]:

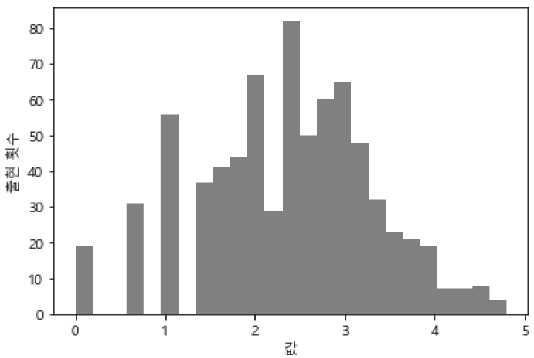
```
X_train_log = np.log(X_train + 1)
X_test_log = np.log(X_test + 1)
```

In [46]:

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

Out[46]:

Text(0.5, 0, '값')



In [47]:

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

테스트 점수: 0.875

자동 선택 단변량 통계

In [48]:

```
from sklearn.datasets import load_breast_cancer
from sklearn.feature_selection import SelectPercentile, f_classif
from sklearn.model_selection import train_test_split

cancer = load_breast_cancer()
rng = np.random.RandomState(42)
noise = rng.normal(size=(len(cancer.data), 50))
X_w_noise = np.hstack([cancer.data, noise])

X_train, X_test, y_train, y_test = train_test_split(X_w_noise, cancer.target, random_state=0, te
st_size=.5)
select = SelectPercentile(score_func=f_classif, percentile=50)
select.fit(X_train, y_train)
X_train_selected = select.transform(X_train)

print("X_train.shape: {}".format(X_train.shape))
print("X_train_selected.shape: {}".format(X_train_selected.shape))
```

```
X_train.shape: (284, 80)
X_train_selected.shape: (284, 40)
```

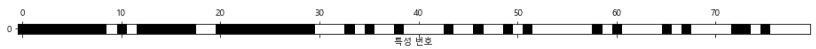
In [49]:

```
mask = select.get_support()

print(mask)

plt.matshow(mask.reshape(1, -1), cmap='gray_r')
plt.xlabel("특성 번호")
plt.yticks([0])
```

```
[ True  True  True  True  True  True  True  True  True False  True False
  True  True  True  True  True  True False False  True  True  True  True
  True  True  True  True  True  True False False False  True False  True
 False False  True False False False False  True False False  True False
 False  True False  True False False False False False False  True False
  True False False False False  True False  True False False False False
  True  True False  True False False False False]
```

Out[49]:

```
([<matplotlib.axis.YTick at 0x17a508599b0>],
 <a list of 1 Text yticklabel objects>)
```



In [50]:

```
from sklearn.linear_model import LogisticRegression

X_test_selected = select.transform(X_test)

lr = LogisticRegression()
lr.fit(X_train, y_train)

print("전체 특성을 사용한 점수: {:.3f}".format(lr.score(X_test, y_test)))

lr.fit(X_train_selected, y_train)

print("선택된 일부 특성을 사용한 점수: {:.3f}".format(lr.score(X_test_selected, y_test)))
```

```
전체 특성을 사용한 점수: 0.930
선택된 일부 특성을 사용한 점수: 0.940

C:\Anaconda\lib\site-packages\sklearn\linear_model\logistic.py:432: FutureWarning:
Default solver will be changed to 'lbfgs' in 0.22. Specify a solver to silence thi
s warning.
  FutureWarning)
C:\Anaconda\lib\site-packages\sklearn\linear_model\logistic.py:432: FutureWarning:
Default solver will be changed to 'lbfgs' in 0.22. Specify a solver to silence thi
s warning.
  FutureWarning)
```

모델 기반 특성 선택

In [51]:

```
from sklearn.feature_selection import SelectFromModel
from sklearn.ensemble import RandomForestClassifier

select = SelectFromModel(RandomForestClassifier(n_estimators=100, random_state=42), threshold="m
edian")
```

In [52]:

```
select.fit(X_train, y_train)
X_train_l1 = select.transform(X_train)

print("X_train.shape: {}".format(X_train.shape))
print("X_train_l1.shape: {}".format(X_train_l1.shape))
```

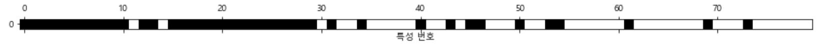
```
X_train.shape: (284, 80)
X_train_l1.shape: (284, 40)
```

In [53]:

```
mask = select.get_support()

plt.matshow(mask.reshape(1, -1), cmap='gray_r')
plt.xlabel("특성 번호")
plt.yticks([0])
```

Out[53]:

```
([<matplotlib.axis.YTick at 0x17a509059e8>],
 <a list of 1 Text yticklabel objects>)
```



In [54]:

```
X_test_l1 = select.transform(X_test)
score = LogisticRegression().fit(X_train_l1, y_train).score(X_test_l1, y_test)

print("Test score: {:.3f}".format(score))
```

```
Test score: 0.951

C:\Anaconda\lib\site-packages\sklearn\linear_model\logistic.py:432: FutureWarning:
Default solver will be changed to 'lbfgs' in 0.22. Specify a solver to silence thi
s warning.
  FutureWarning)
```

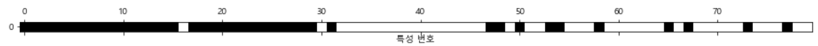
반복적 특성 선택

In [55]:

```
from sklearn.feature_selection import RFE
select = RFE(RandomForestClassifier(n_estimators=100, random_state=42), n_features_to_select=40)

select.fit(X_train, y_train)
mask = select.get_support()

plt.matshow(mask.reshape(1, -1), cmap='gray_r')
plt.xlabel("특성 번호")
plt.yticks([0])
```

Out[55]:

```
([<matplotlib.axis.YTick at 0x17a5091e940>],
 <a list of 1 Text yticklabel objects>)
```



In [56]:

```
X_train_rfe = select.transform(X_train)
X_test_rfe = select.transform(X_test)
score = LogisticRegression().fit(X_train_rfe, y_train).score(X_test_rfe, y_test)

print("테스트 점수: {:.3f}".format(score))
```

```
테스트 점수: 0.951

C:\Anaconda\lib\site-packages\sklearn\linear_model\logistic.py:432: FutureWarning:
Default solver will be changed to 'lbfgs' in 0.22. Specify a solver to silence thi
s warning.
  FutureWarning)
```

In [57]:

```
print("테스트 점수: {:.3f}".format(select.score(X_test, y_test)))
```

```
테스트 점수: 0.951
```

전문가 지식 활용

In [58]:

```
citibike = mglearn.datasets.load_citibike()
```

In [59]:

```
print("시티 바이크 데이터:\n{}".format(citibike.head()))
```

```
시티 바이크 데이터:
starttime
2015-08-01 00:00:00     3
2015-08-01 03:00:00     0
2015-08-01 06:00:00     9
2015-08-01 09:00:00    41
2015-08-01 12:00:00    39
Freq: 3H, Name: one, dtype: int64
```

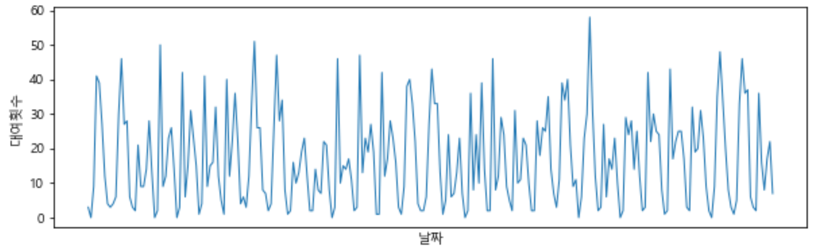
In [60]:

```
plt.figure(figsize=(10, 3))
xticks = pd.date_range(start=citibike.index.min(), end=citibike.index.max(), freq='D')
week = ["일", "월", "화","수", "목", "금", "토"]
xticks_name = [week[int(w)]+d for w, d in zip(xticks.strftime("%w"), xticks.strftime(" %m-%d"
))]

plt.xticks(xticks.astype(int), xticks_name, rotation=90, ha="left")
plt.plot(citibike, linewidth=1)
plt.xlabel("날짜")
plt.ylabel("대여횟수")
```

```
C:\Anaconda\lib\site-packages\pandas\plotting\_converter.py:129: FutureWarning: Us
ing an implicitly registered datetime converter for a matplotlib plotting method.
The converter was registered by pandas on import. Future versions of pandas will r
equire you to explicitly register matplotlib converters.

To register the converters:
        >>> from pandas.plotting import register_matplotlib_converters
        >>> register_matplotlib_converters()
  warnings.warn(msg, FutureWarning)
```

Out[60]:

```
Text(0, 0.5, '대여횟수')
```



In [62]:

```
y = citibike.values
X = citibike.index.astype("int64").values.reshape(-1, 1) // 10**9
```

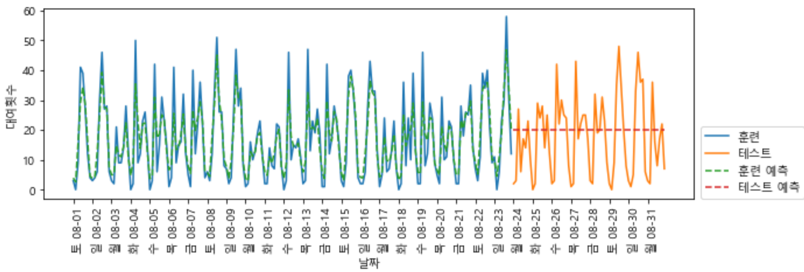
In [63]:

```
n_train = 184

def eval_on_features(features, target, regressor):
    X_train, X_test = features[:n_train], features[n_train:]
    y_train, y_test = target[:n_train], target[n_train:]
    regressor.fit(X_train, y_train)

    print("테스트 세트 R^2: {:.2f}".format(regressor.score(X_test, y_test)))

    y_pred = regressor.predict(X_test)
    y_pred_train = regressor.predict(X_train)

    plt.figure(figsize=(10, 3))
    plt.xticks(range(0, len(X), 8), xticks_name, rotation=90, ha="left")
    plt.plot(range(n_train), y_train, label="훈련")
    plt.plot(range(n_train, len(y_test) + n_train), y_test, '-', label="테스트")
    plt.plot(range(n_train), y_pred_train, '--', label="훈련 예측")
    plt.plot(range(n_train, len(y_test) + n_train), y_pred, '--', label="테스트 예측")
    plt.legend(loc=(1.01, 0))
    plt.xlabel("날짜")
    plt.ylabel("대여횟수")
```

In [64]:

```
regressor = RandomForestRegressor(n_estimators=100, random_state=0)
eval_on_features(X, y, regressor)
```

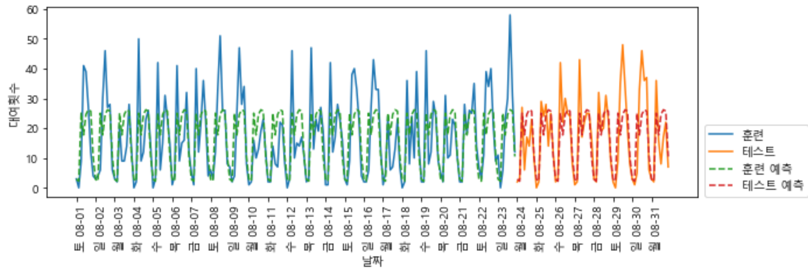
테스트 세트 R^2: -0.04



In [65]:

```
X_hour = citibike.index.hour.values.reshape(-1, 1)
eval_on_features(X_hour, y, regressor)
```

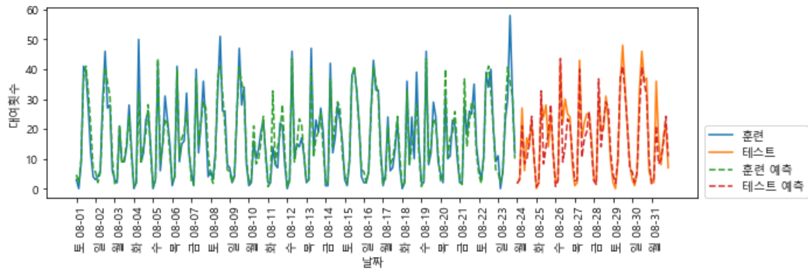
테스트 세트 R^2: 0.60



In [66]:

```
X_hour_week = np.hstack([citibike.index.dayofweek.values.reshape(-1, 1), citibike.index.hour.val
ues.reshape(-1, 1)])
eval_on_features(X_hour_week, y, regressor)
```

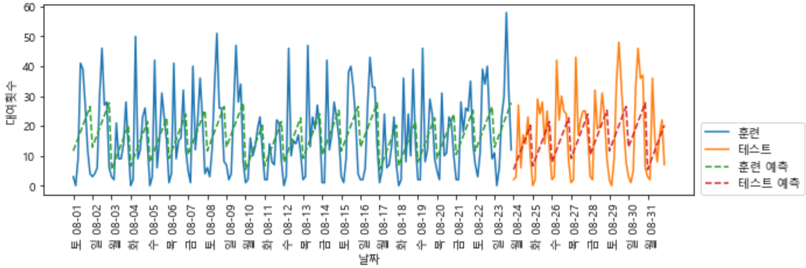
테스트 세트 R^2: 0.84



In [67]:

```
from sklearn.linear_model import LinearRegression

eval_on_features(X_hour_week, y, LinearRegression())
```

테스트 세트 R^2: 0.13



In [68]:

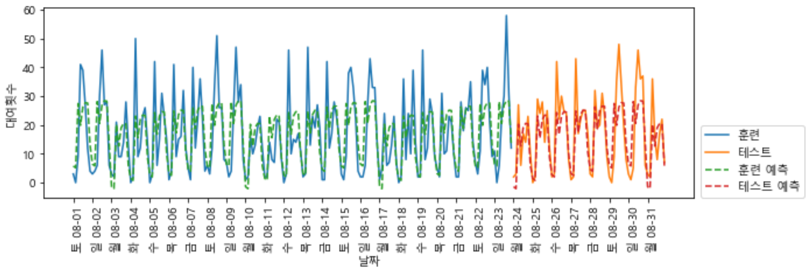
```
enc = OneHotEncoder()
X_hour_week_onehot = enc.fit_transform(X_hour_week).toarray()
```

```
C:₩Anaconda₩lib₩site-packages₩sklearn₩preprocessing₩_encoders.py:415: FutureWarnin
g: The handling of integer data will change in version 0.22. Currently, the catego
ries are determined based on the range [0, max(values)], while in the future they
will be determined based on the unique values.
If you want the future behaviour and silence this warning, you can specify "catego
ries='auto'".
In case you used a LabelEncoder before this OneHotEncoder to convert the categorie
s to integers, then you can now use the OneHotEncoder directly.
  warnings.warn(msg, FutureWarning)
```

In [69]:

```
eval_on_features(X_hour_week_onehot, y, Ridge())
```

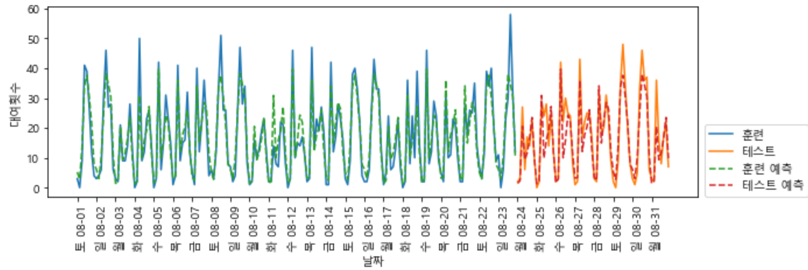
테스트 세트 R^2: 0.62



In [70]:

```
poly_transformer = PolynomialFeatures(degree=2, interaction_only=True, include_bias=False)
X_hour_week_onehot_poly = poly_transformer.fit_transform(X_hour_week_onehot)
lr = Ridge()
eval_on_features(X_hour_week_onehot_poly, y, lr)
```

테스트 세트 R^2: 0.85



In [71]:

```
hour = ["%02d:00" % i for i in range(0, 24, 3)]
day = ["월", "화", "수", "목", "금", "토", "일"]
features =  day + hour
```

In [72]:

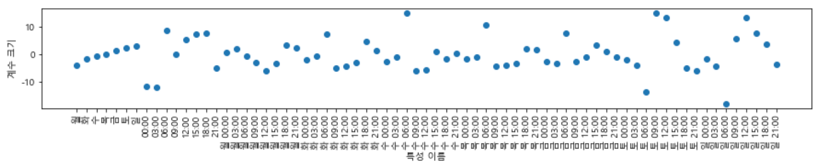
```
features_poly = poly_transformer.get_feature_names(features)
features_nonzero = np.array(features_poly)[lr.coef_ != 0]
coef_nonzero = lr.coef_[lr.coef_ != 0]
```

In [73]:

```
plt.figure(figsize=(15, 2))
plt.plot(coef_nonzero, 'o')
plt.xticks(np.arange(len(coef_nonzero)), features_nonzero, rotation=90)
plt.xlabel("특성 이름")
plt.ylabel("계수 크기")
```

Out[73]:

```
Text(0, 0.5, '계수 크기')
```



In [ ]: