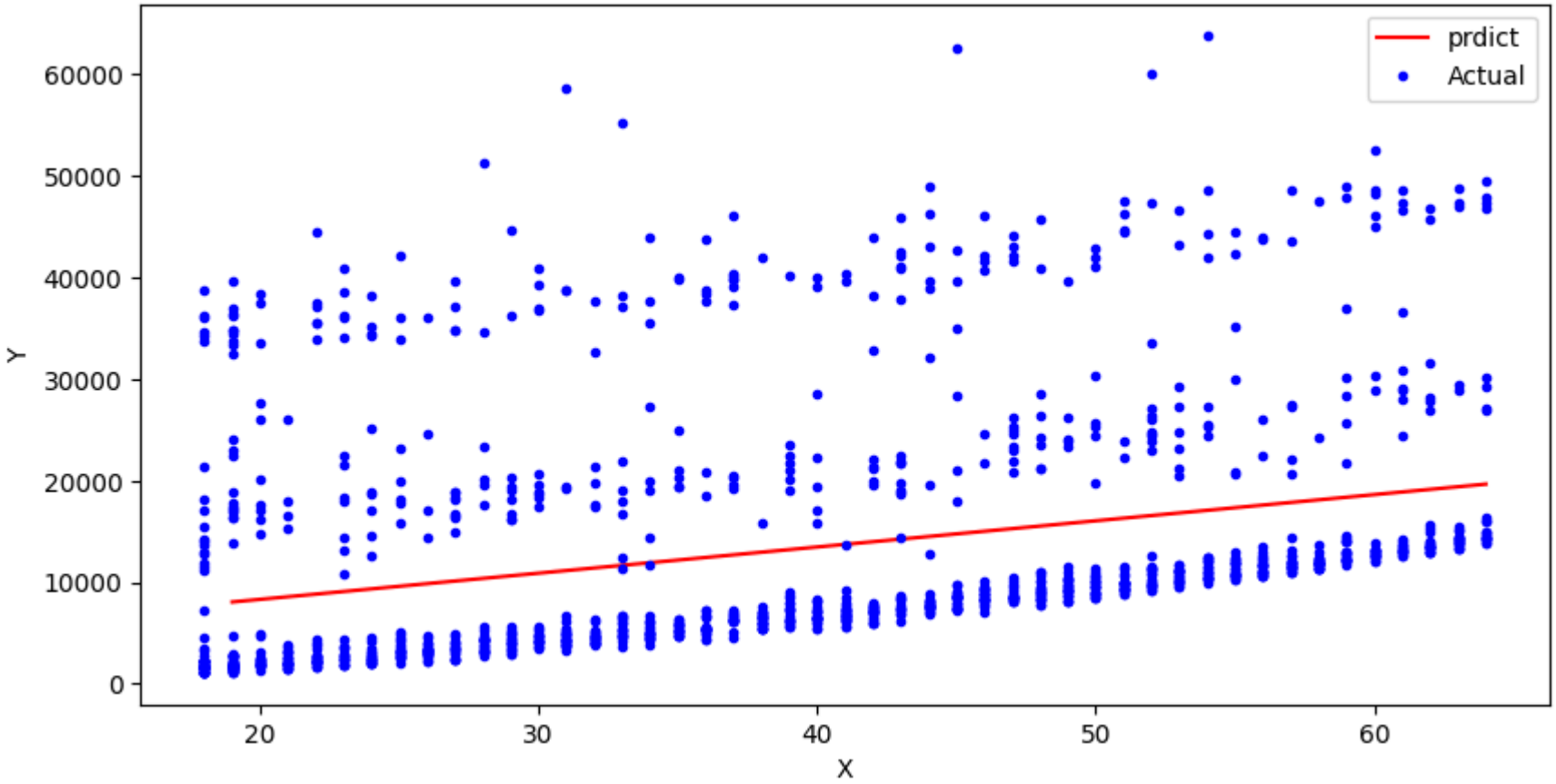
In [2]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from sklearn.linear_model import LinearRegression

data = pd.read_csv("https://raw.githubusercontent.com/ADPclass/ADP_book_ver01/main/data/insurance.csv")
x=data['age']
y=data['charges']
x=np.array(x)
y=np.array(y)
x=x.reshape(-1,1)
y=y.reshape(-1,1)
lr=LinearRegression()
lr.fit(x,y)
print('선형 회귀 모델 결과 ')
print('절편',lr.intercept_,'계수',lr.coef_)
print(lr.score(x,y))
```

```
선형 회귀 모델 결과
절편 [3165.88500606] 계수 [[257.72261867]]
0.08940589967885804
```

In [3]:
```
x_new=[[19],[64]]
y_hat=lr.predict(x_new)
print(y_hat)
```

```
[[ 8062.61476073]
 [19660.13260074]]
```

In [4]:
```
plt.figure(figsize=(10,5))
plt.plot(x_new,y_hat,'-r',label='prdict')
plt.plot(x,y,"b.",label='Actual')
plt.legend(loc='upper right')
plt.xlabel('X')
plt.ylabel('Y')
plt.show()
```



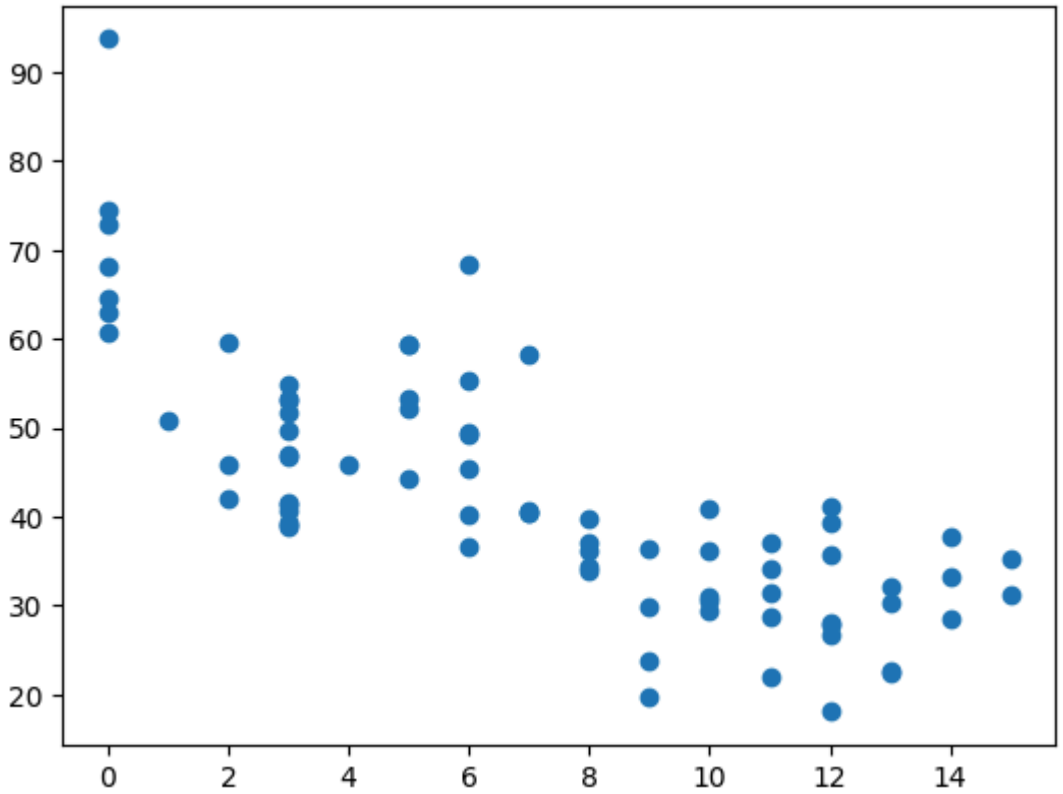
In [5]:
```
import pandas as pd
cereal = pd.read_csv("https://raw.githubusercontent.com/ADPclass/ADP_book_ver01/main/data/cereal.csv")
```

In [6]:
```
cereal=cereal[cereal.columns[3:]]
cereal=cereal[cereal>=0]
cereal.head()
```

Out[6]:

	calories	protein	fat	sodium	fiber	carbo	sugars	potass	vitamins	shelf	weight	cups	rating
0	70	4	1	130	10.0	5.0	6.0	280.0	25	3	1.0	0.33	68.402973
1	120	3	5	15	2.0	8.0	8.0	135.0	0	3	1.0	1.00	33.983679
2	70	4	1	260	9.0	7.0	5.0	320.0	25	3	1.0	0.33	59.425505
3	50	4	0	140	14.0	8.0	0.0	330.0	25	3	1.0	0.50	93.704912
4	110	2	2	200	1.0	14.0	8.0	NaN	25	3	1.0	0.75	34.384843

In [7]:
```
import matplotlib.pyplot as plt

cereal2= cereal[['sugars','rating']]
cereal2=cereal2.dropna(axis=0)
cereal2.sort_values(by=['sugars'],inplace=True)
cereal2.reset_index(drop=True,inplace=True)
x=cereal2['sugars'].values
y=cereal2['rating'].values
plt.scatter(x,y)
plt.show()
```



In [8]:
```
from sklearn.model_selection import train_test_split
X_train,X_test,Y_train,Y_test=train_test_split(x,y,test_size=0.3, random_state=1)
print(X_train.shape,X_test.shape)
print(Y_train.shape,Y_test.shape)

from sklearn.preprocessing import PolynomialFeatures
poly_reg=PolynomialFeatures(degree=2)
X_poly=poly_reg.fit_transform(X_train.reshape(-1,1))

from sklearn.linear_model import LinearRegression
```

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

```
(53,) (23,)
(53,) (23,)
```

Out[8]:

LinearRegression

LinearRegression()

In [9]:

```
X_test_poly = poly_reg.transform(X_test.reshape(-1,1))
pred=reg.predict(X_test_poly)
np.set_printoptions(precision=2)
#print(np.concatenate([pred.reshape(-1,1),Y_test.reshape(-1,1)],axis=1))
```

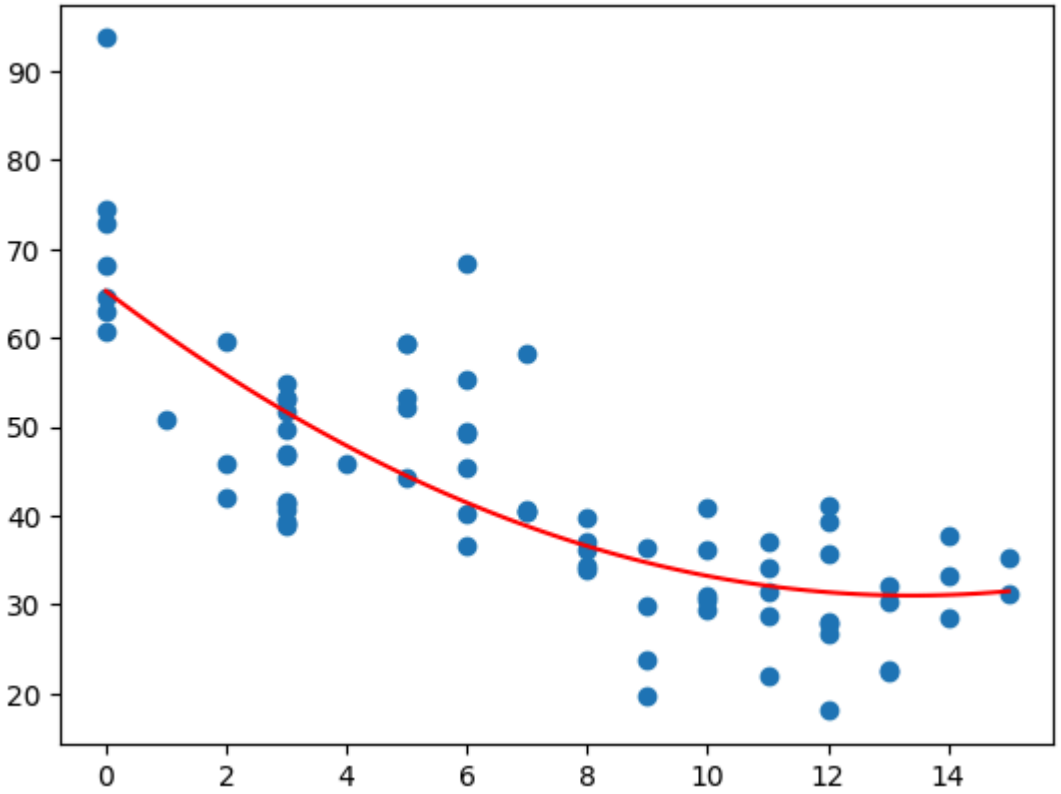
In [10]:

```
from sklearn.metrics import mean_squared_error, mean_absolute_error
mse=mean_squared_error(Y_test,pred)
mae=mean_absolute_error(Y_test,pred)
rmse=np.sqrt(mse)
acc=reg.score(poly_reg.transform(X_test.reshape(-1,1)),Y_test)
print('MAE',mae)
print('RMSE',rmse)
print('R2',acc*100,'%')
```

```
MAE 4.605784071295562
RMSE 5.793540436815039
R2 74.37569749767225 %
```

In [11]:

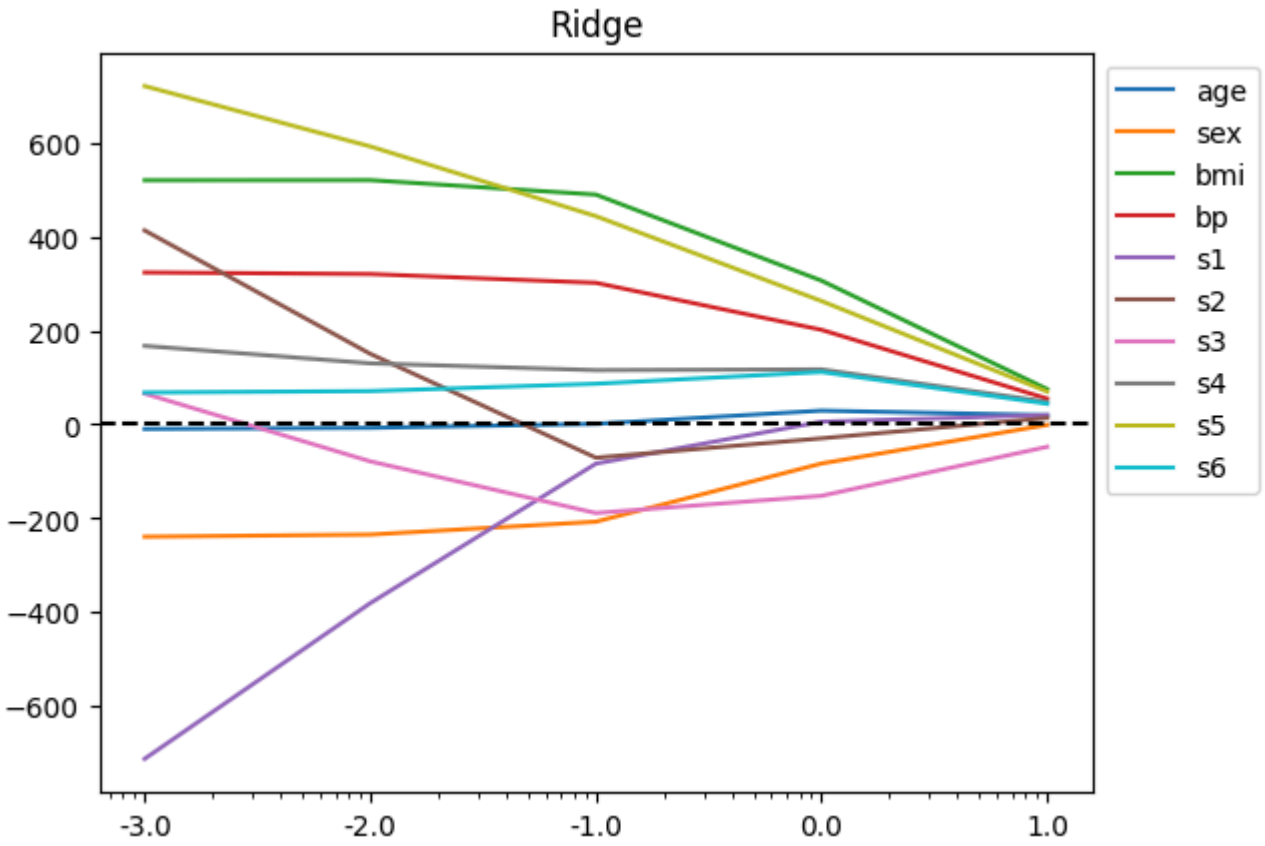
```
X_new=np.linspace(0,15,100).reshape(100,1)
X_new_poly=poly_reg.transform(X_new)
y_new=reg.predict(X_new_poly)

plt.plot(x,y,'o',label='Actual')
plt.plot(X_new,y_new,'r-',label='Prediction')
plt.show()
```



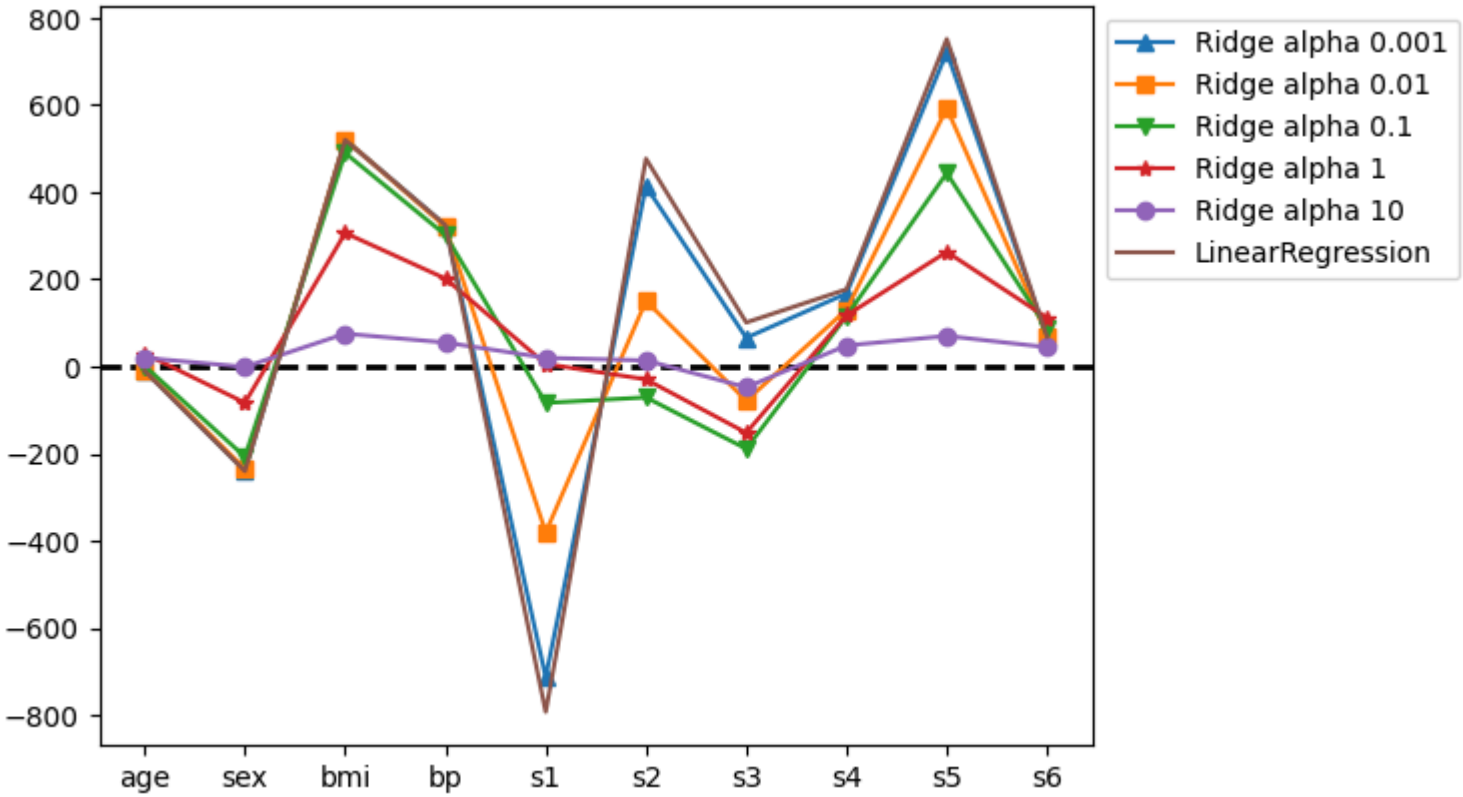
In [12]:

```
from sklearn.datasets import load_diabetes

diabetes=load_diabetes()
x=pd.DataFrame(diabetes.data,columns=diabetes.feature_names)
y=diabetes.target

from sklearn.linear_model import Ridge
import numpy as np

alpha=np.logspace(-3,1,5)
data=[]

for i,a in enumerate(alpha):
    ridge=Ridge(alpha=a,random_state=1)
    ridge.fit(x,y)
    data.append(pd.Series(np.hstack([ridge.coef_])))

df_ridge=pd.DataFrame(data,index=alpha)
df_ridge.columns=x.columns

plt.semilogx(df_ridge)
```

```
plt.xticks(alpha,labels=np.log10(alpha))
plt.legend(labels=df_ridge.columns,bbox_to_anchor=(1,1))
plt.title('Ridge')
plt.axhline(y=0,linestyle='--',color='black')
plt.show()
```

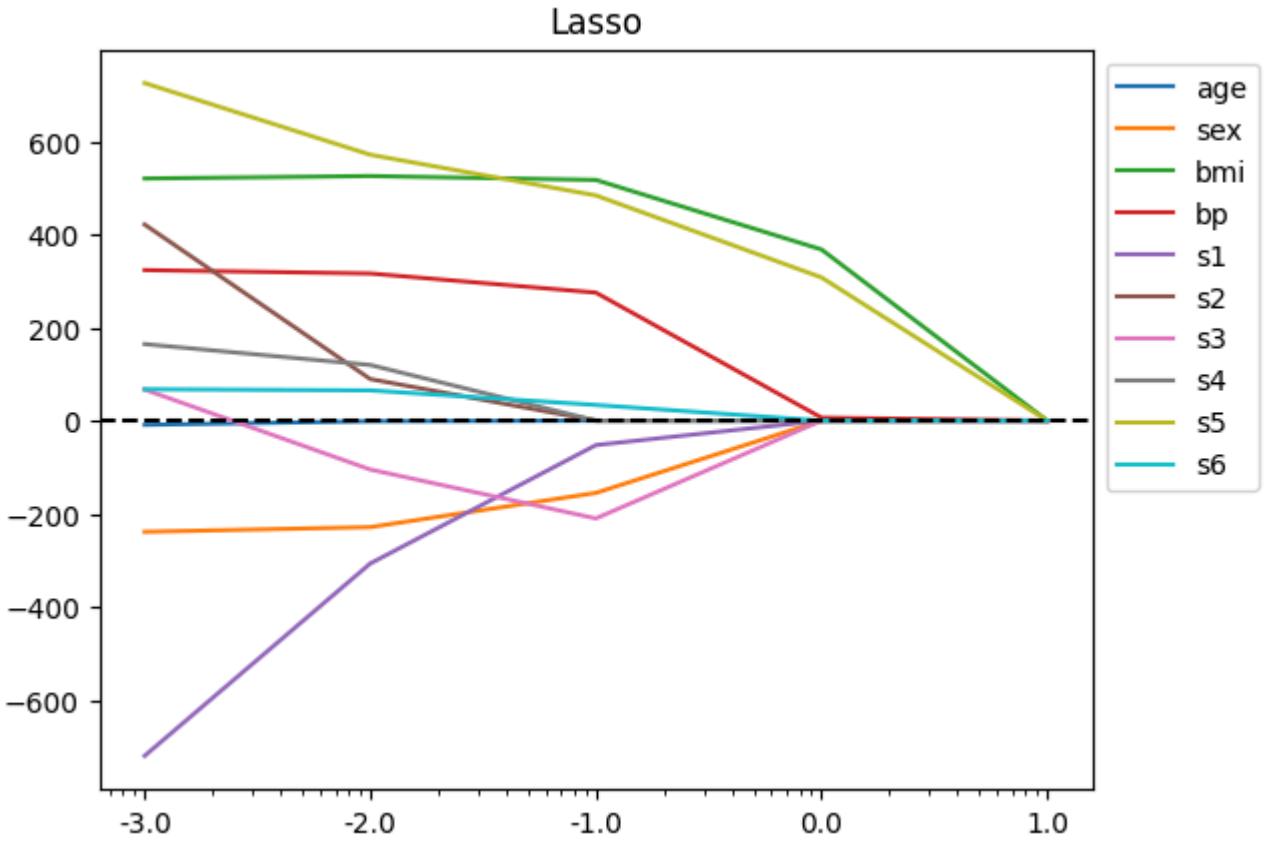


In [13]:
```
from sklearn.linear_model import LinearRegression

lr=LinearRegression()
lr.fit(x,y)
plt.axhline(y=0,linestyle='--',color='k',linewidth=2)
plt.plot(df_ridge.loc[0.001],'^-',label='Ridge alpha 0.001')
plt.plot(df_ridge.loc[0.01],'s-',label='Ridge alpha 0.01')
plt.plot(df_ridge.loc[0.1],'v-',label='Ridge alpha 0.1')
plt.plot(df_ridge.loc[1],'*-',label='Ridge alpha 1')
plt.plot(df_ridge.loc[10],'o-',label='Ridge alpha 10')
plt.plot(lr.coef_,label='LinearRegression')
plt.legend(bbox_to_anchor=(1,1))
plt.show()
```



In [14]:

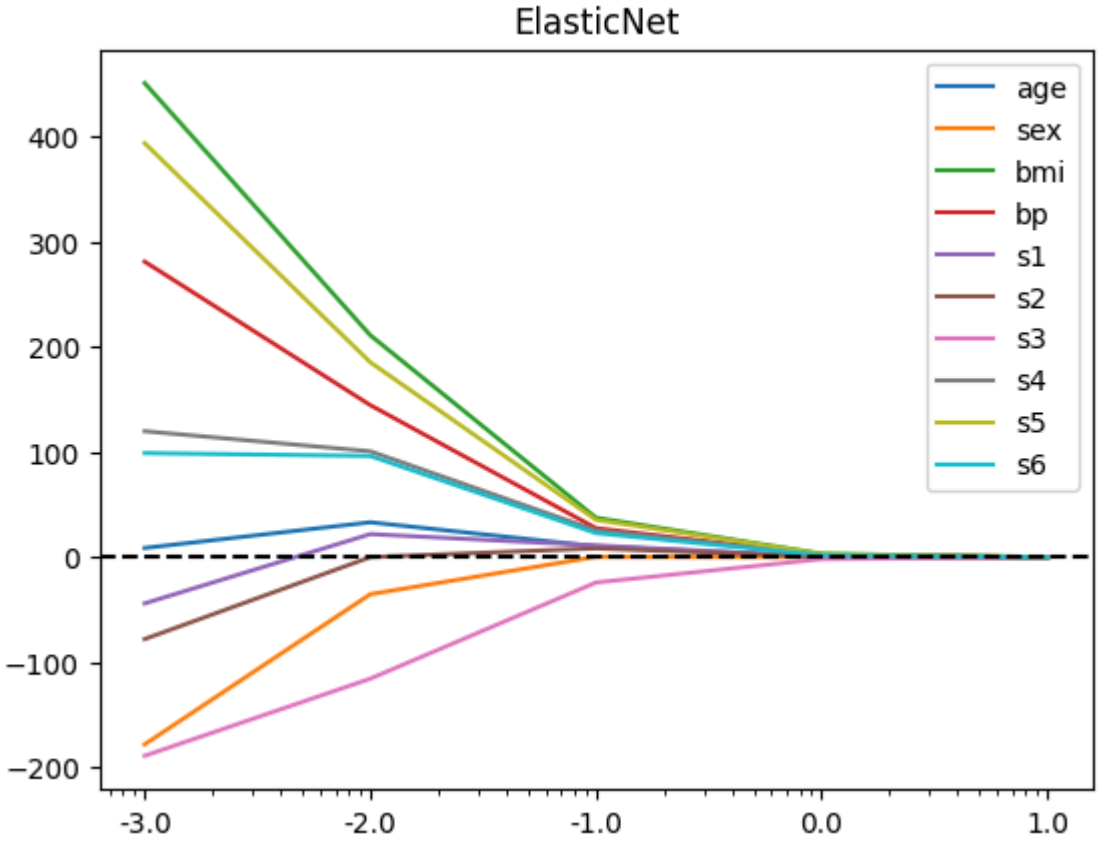
```
from sklearn.linear_model import Lasso
import numpy as np

alpha=np.logspace(-3,1,5)
data=[]

for i,a in enumerate(alpha):
    ridge=Lasso(alpha=a,random_state=1)
    ridge.fit(x,y)
    data.append(pd.Series(np.hstack([ridge.coef_])))

df_ridge=pd.DataFrame(data,index=alpha)
df_ridge.columns=x.columns

plt.semilogx(df_ridge)
plt.xticks(alpha,labels=np.log10(alpha))
plt.legend(labels=df_ridge.columns,bbox_to_anchor=(1,1))
plt.title('Lasso')
plt.axhline(y=0,linestyle='--',color='black')
plt.show()
```



In [15]:

```
from sklearn.linear_model import ElasticNet
import numpy as np

alpha=np.logspace(-3,1,5)
data=[]

for i,a in enumerate(alpha):
    ridge=ElasticNet(alpha=a,l1_ratio=0.5,random_state=1)
    ridge.fit(x,y)
    data.append(pd.Series(np.hstack([ridge.coef_])))

df_ridge=pd.DataFrame(data,index=alpha)
df_ridge.columns=x.columns

plt.semilogx(df_ridge)
plt.xticks(alpha,labels=np.log10(alpha))
plt.legend(labels=df_ridge.columns,bbox_to_anchor=(1,1))
plt.title('ElasticNet')
```

```
plt.axhline(y=0,linestyle='--',color='black')
plt.show()
```



In [16]:
```
import pandas as pd
import numpy as np
import warnings

warnings.filterwarnings('ignore')
body=pd.read_csv("https://raw.githubusercontent.com/ADPclass/ADP_book_ver01/main/data/bodyPerformance.csv")
```

In [17]:
```
body['gender']=np.where(body['gender']=='M',0,1)
body['class']=np.where(body['class']=='A',1,0)
```

In [18]:
```
from sklearn.model_selection import train_test_split

feature_columns= list(body.columns.difference(['class']))
x=body[feature_columns]
y=body['class']
```

```
X_train,X_test,Y_train,Y_test=train_test_split(x,y,stratify=y,train_size=0.7,random_state=1)

from sklearn.linear_model import LogisticRegression

logR= LogisticRegression(random_state=0)
logR.fit(X_train,Y_train)
```

Out[18]:

LogisticRegression

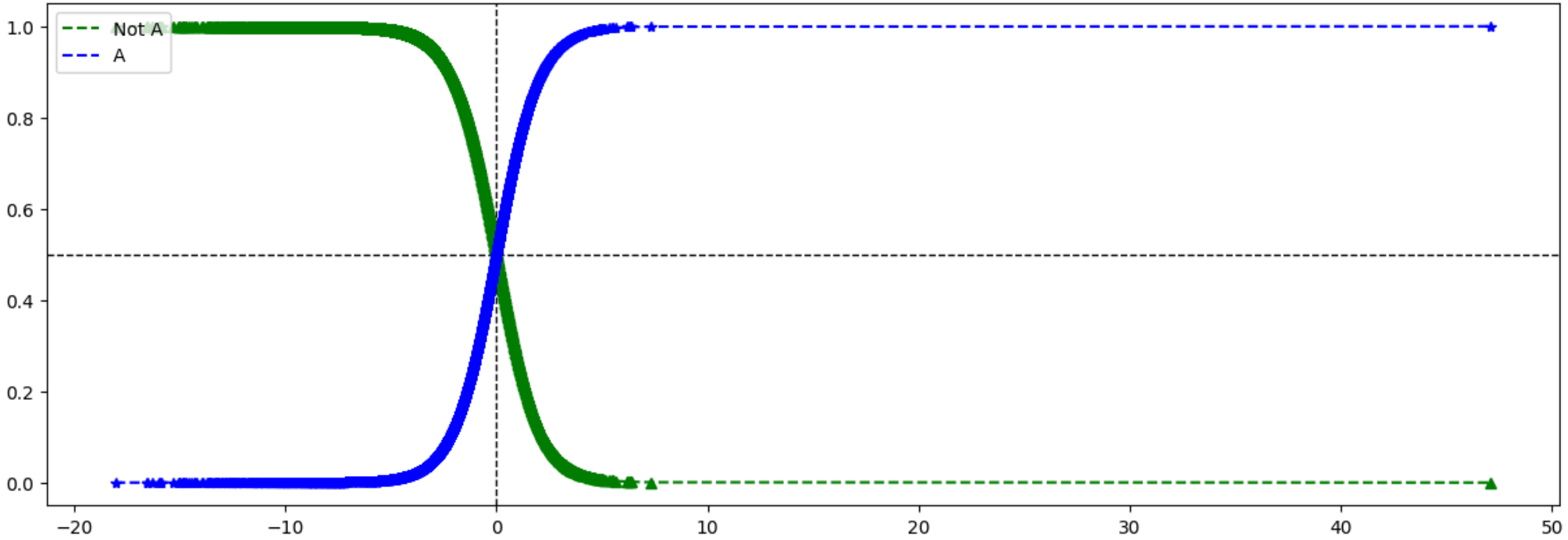
LogisticRegression(random_state=0)

In [19]:

```
proba=pd.DataFrame(logR.predict_proba(X_train))
confidence_score=logR.decision_function(X_train) # 로짓
df=pd.concat([proba,pd.DataFrame(confidence_score)],axis=1)
df.columns=['Not A','A','decision_function']
df.sort_values(['decision_function'],inplace=True)
df.reset_index(inplace=True,drop=True)
```

In [20]:

```
import matplotlib.pyplot as plt

plt.figure(figsize=(15,5))

plt.axhline(y=0.5 , linestyle='--', color = 'k', linewidth=1)
plt.axvline(x=0 , linestyle='--', color = 'k', linewidth=1)
plt.plot(df['decision_function'],df['Not A'],'g--',label='Not A')
plt.plot(df['decision_function'],df['Not A'],'g^')
plt.plot(df['decision_function'],df['A'],'b--',label='A')
plt.plot(df['decision_function'],df['A'],'b*')
plt.legend(loc='upper left')
plt.show()
```

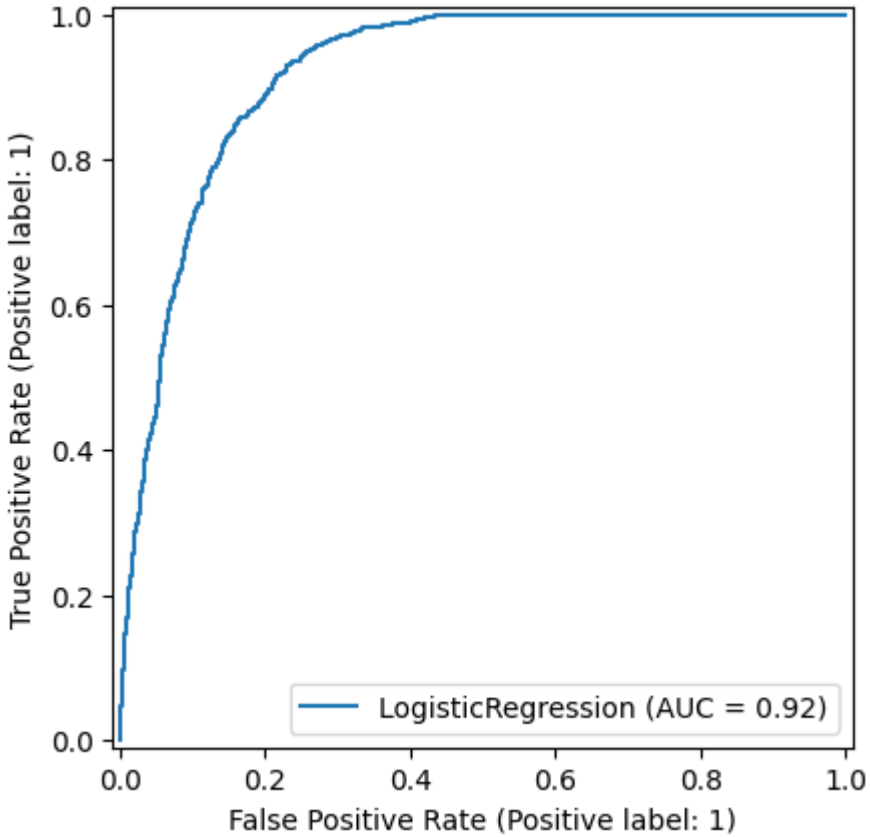


In [38]:
```
from sklearn.metrics import confusion_matrix,accuracy_score,precision_score,recall_score,f1_score

pred=logR.predict(X_test)

test_cm = confusion_matrix(Y_test,pred)
test_acc=accuracy_score(Y_test,pred)
test_prc=precision_score(Y_test,pred)
test_rcll=precision_score(Y_test,pred)
test_f1=f1_score(Y_test,pred)

print(test_cm)
```

```
[[2760  254]
 [ 353  651]]
```

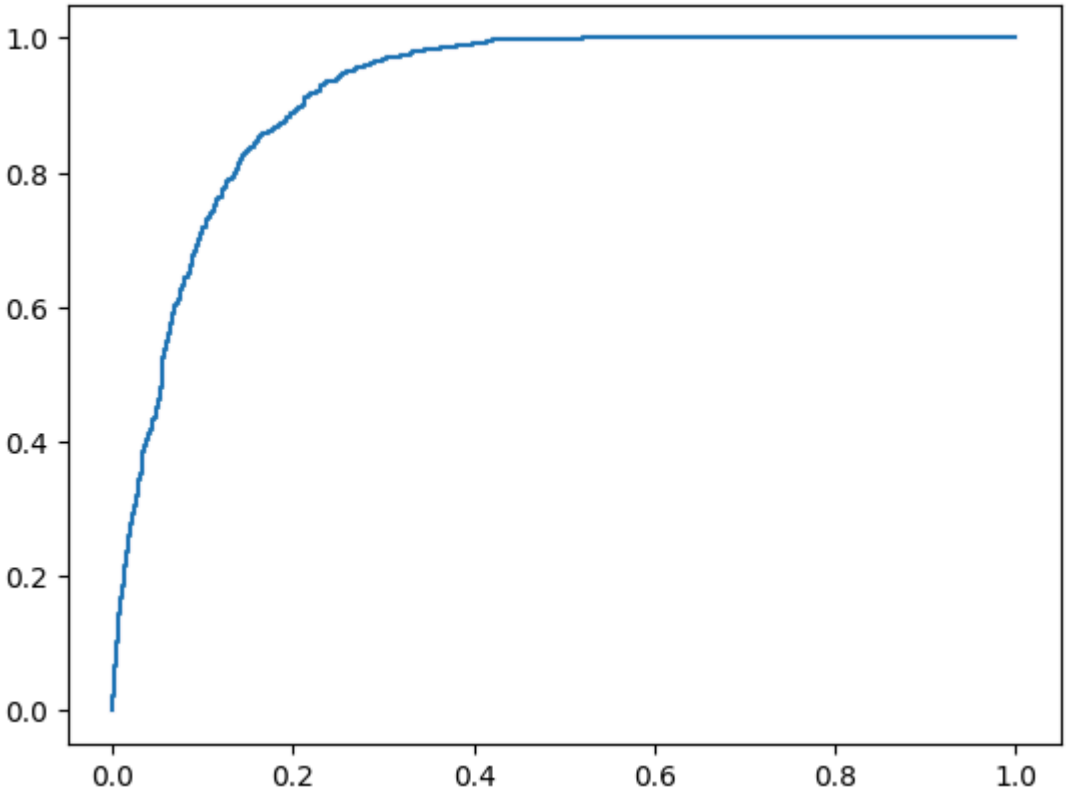
In [39]:
```
from sklearn.metrics import RocCurveDisplay
RocCurveDisplay.from_estimator(logR,X_test,Y_test)
```

Out[39]: <sklearn.metrics._plot.roc_curve.RocCurveDisplay at 0x350a85bd0>



In [49]:
```
from sklearn.metrics import roc_curve,roc_auc_score
fpr,tpr,thres= roc_curve(Y_test,logR.predict_proba(X_test)[:,1],pos_label=1)
print(roc_auc_score(Y_test,logR.predict_proba(X_test)[:,1]))
plt.plot(fpr,tpr)
plt.show()
```

```
0.9186495557253401
```



In [1]:

```python
from sklearn.linear_model import LogisticRegression
from sklearn.datasets import load_iris
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn.metrics import accuracy_score

iris = load_iris()
X, y = iris.data, iris.target

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

scaler = StandardScaler()
X_train = scaler.fit_transform(X_train)
X_test = scaler.transform(X_test)

# 로지스틱 회귀 + L2 정규화 (릿지)
logreg_l2 = LogisticRegression(penalty='l2', C=1.0, solver='lbfgs', multi_class='multinomial')
logreg_l2.fit(X_train, y_train)

y_pred = logreg_l2.predict(X_test)
```

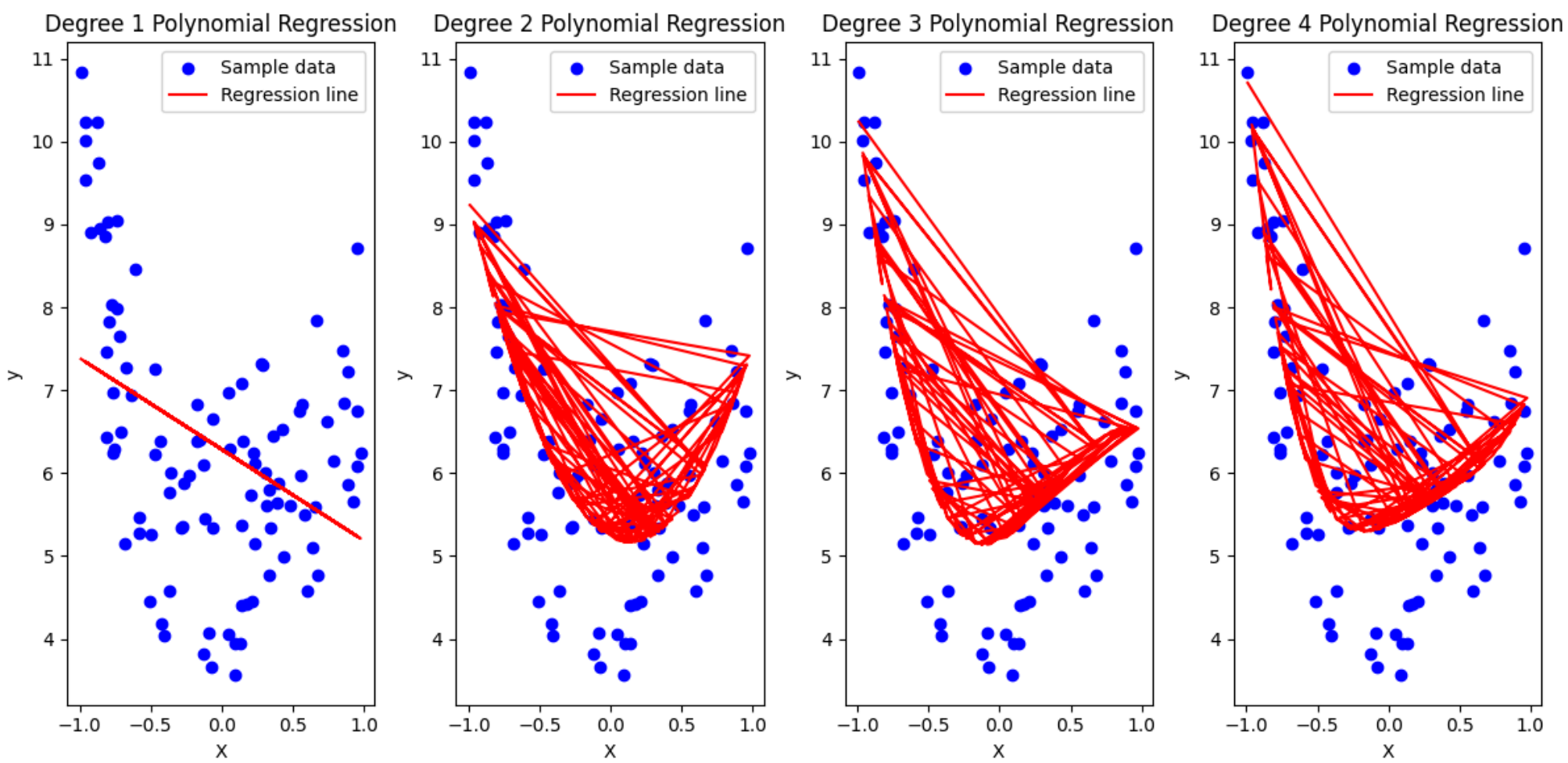
```
accuracy = accuracy_score(y_test, y_pred)
print(f"Accuracy with L2 regularization: {accuracy:.4f}")
```

```
Accuracy with L2 regularization: 1.0000
```

```
/opt/homebrew/Caskroom/miniforge/base/envs/general/lib/python3.11/site-packages/sklearn/linear_model/_logistic.py:1247: FutureWa
rning: 'multi_class' was deprecated in version 1.5 and will be removed in 1.7. From then on, it will always use 'multinomial'. L
eave it to its default value to avoid this warning.
  warnings.warn(
```

In [2]:

```
import numpy as np
import matplotlib.pyplot as plt
from sklearn.linear_model import LinearRegression
from sklearn.preprocessing import PolynomialFeatures

np.random.seed(0)
X = 2 * np.random.rand(100, 1) - 1
y = 2 * X**4 - 3 * X**3 + 2 * X**2 + 1 * X + 5 + np.random.randn(100, 1)

plt.figure(figsize=(12, 6))
degrees = [1, 2, 3, 4]  # 1차부터 4차까지 다항회귀 적용
for i, degree in enumerate(degrees):
    ax = plt.subplot(1, len(degrees), i + 1)
    polynomial_features = PolynomialFeatures(degree=degree)
    X_poly = polynomial_features.fit_transform(X)
    model = LinearRegression()
    model.fit(X_poly, y)
    y_pred = model.predict(X_poly)
    plt.scatter(X, y, color='blue', label='Sample data')
    plt.plot(X, y_pred, color='red', label='Regression line')
    plt.title(f'Degree {degree} Polynomial Regression')
    plt.xlabel('X')
    plt.ylabel('y')
    plt.legend()
plt.tight_layout()
plt.show()
```


Degree 1 Polynomial Regression
Degree 2 Polynomial Regression
Degree 3 Polynomial Regression
Degree 4 Polynomial Regression
Sample data
Regression line
X
y