In [603]:
```
import numpy as np
import pandas as pd
import matplotlib as mpl
import matplotlib.pyplot as plt
import mglearn
%matplotlib inline
import seaborn as sns
import platform
from matplotlib import font_manager , rc

if platform.system() == 'Darwin':
  rc('font' , family = 'AppleGothic')
elif platform.system() == 'Windows':
  path = 'C:/Windows/Fonts/malgun.ttf'
  font_name = font_manager.FontProperties(fname = path).get_name()
  rc('font' , family = font_name)
else:
  print('모름')
plt.rcParams['axes.unicode_minus'] = False
import warnings
warnings.filterwarnings('ignore')
```

executed in 22ms, finished 17:31:35 2023-10-24

1 당뇨병 진행도 예측

In [604]:
```
from sklearn.datasets import load_diabetes
diabetes = load_diabetes()
```

executed in 13ms, finished 17:31:35 2023-10-24

In [605]:
```
df = pd.DataFrame(diabetes.data , columns = diabetes.feature_names)
```

executed in 14ms, finished 17:31:35 2023-10-24

In [606]:
```
raw = df.to_numpy()
```

executed in 14ms, finished 17:31:35 2023-10-24

In [607]:
```
df
```
executed in 15ms, finished 17:31:35 2023-10-24

Out[607]:

	age	sex	bmi	bp	s1	s2	s3	s4
0	0.038076	0.050680	0.061696	0.021872	-0.044223	-0.034821	-0.043401	-0.002592
1	-0.001882	-0.044642	-0.051474	-0.026328	-0.008449	-0.019163	0.074412	-0.039493
2	0.085299	0.050680	0.044451	-0.005670	-0.045599	-0.034194	-0.032356	-0.002592
3	-0.089063	-0.044642	-0.011595	-0.036656	0.012191	0.024991	-0.036038	0.034309
4	0.005383	-0.044642	-0.036385	0.021872	0.003935	0.015596	0.008142	-0.002592
...	...	...	...	...	...	...	...	...
437	0.041708	0.050680	0.019662	0.059744	-0.005697	-0.002566	-0.028674	-0.002592
438	-0.005515	0.050680	-0.015906	-0.067642	0.049341	0.079165	-0.028674	0.034309
439	0.041708	0.050680	-0.015906	0.017293	-0.037344	-0.013840	-0.024993	-0.011080
440	-0.045472	-0.044642	0.039062	0.001215	0.016318	0.015283	-0.028674	0.026560
441	-0.045472	-0.044642	-0.073030	-0.081413	0.083740	0.027809	0.173816	-0.039493

442 rows × 10 columns

In [608]:
```
from sklearn.model_selection import train_test_split
```
executed in 15ms, finished 17:31:35 2023-10-24

In [609]:
```
train_input , test_input , train_target , test_target = train_test_split(raw , di
```
executed in 14ms, finished 17:31:35 2023-10-24

In [610]:
```
len(train_input)
```
executed in 13ms, finished 17:31:35 2023-10-24

Out[610]: 331

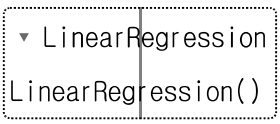
In [611]:
```
len(train_target)
```
executed in 15ms, finished 17:31:35 2023-10-24

Out[611]: 331

In [612]:
```
len(test_input)
```
executed in 12ms, finished 17:31:35 2023-10-24

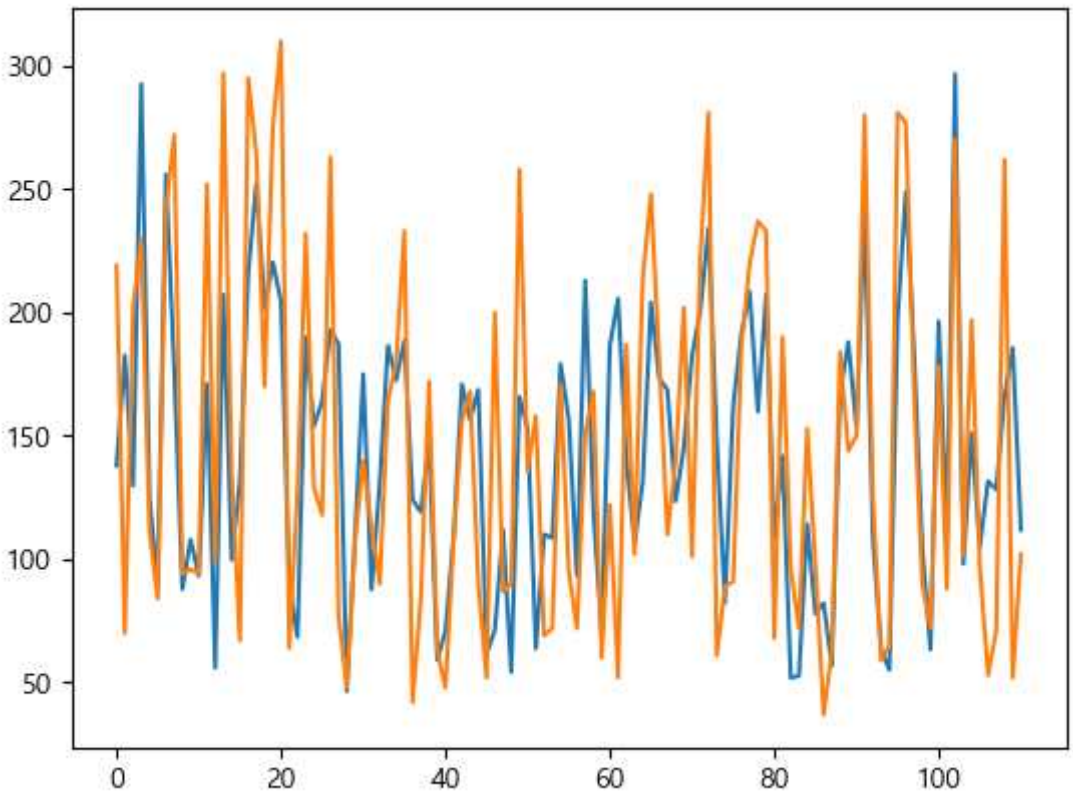
Out[612]: 111

In [613]:
```
#회귀모델 만들기
from sklearn.linear_model import LinearRegression
lr = LinearRegression()
lr.fit(train_input , train_target)
```
executed in 13ms, finished 17:31:35 2023-10-24

Out[613]:


In [614]:
```
test = lr.predict(test_input)
```
executed in 14ms, finished 17:31:35 2023-10-24

In [615]:
```
# 모델을 적용해서 test_input의 target을 예측한 값을 test에 담고 , 실제 target과 t
test = lr.predict(test_input)
plt.plot(test)
plt.plot(test_target)
plt.show()
```
executed in 146ms, finished 17:31:36 2023-10-24



- 너무 차이나는데...

In [616]:
```
lr.score(train_input , train_target)
```
executed in 21ms, finished 17:31:36 2023-10-24

Out[616]: 0.5190341891679049

In [617]:
```
lr.score(test_input , test_target)
```
executed in 14ms, finished 17:31:36 2023-10-24

Out[617]: 0.4849058889476756

- 점수가 너무 낮다.

In [618]:
```
from sklearn.metrics import mean_squared_error
mse = mean_squared_error(lr.predict(test_input) , test_target)
```
executed in 14ms, finished 17:31:36 2023-10-24

In [619]:
```
mse
```
executed in 12ms, finished 17:31:36 2023-10-24

Out[619]: 2848.3106508475053

1.1 mse가 너무 높다. 성능이 나쁘다

In [620]:
```
df
```
executed in 14ms, finished 17:31:36 2023-10-24

Out[620]:

	age	sex	bmi	bp	s1	s2	s3	s4
0	0.038076	0.050680	0.061696	0.021872	-0.044223	-0.034821	-0.043401	-0.002592
1	-0.001882	-0.044642	-0.051474	-0.026328	-0.008449	-0.019163	0.074412	-0.039493
2	0.085299	0.050680	0.044451	-0.005670	-0.045599	-0.034194	-0.032356	-0.002592
3	-0.089063	-0.044642	-0.011595	-0.036656	0.012191	0.024991	-0.036038	0.034309
4	0.005383	-0.044642	-0.036385	0.021872	0.003935	0.015596	0.008142	-0.002592
...	...	...	...	...	...	...	...	...
437	0.041708	0.050680	0.019662	0.059744	-0.005697	-0.002566	-0.028674	-0.002592
438	-0.005515	0.050680	-0.015906	-0.067642	0.049341	0.079165	-0.028674	0.034309
439	0.041708	0.050680	-0.015906	0.017293	-0.037344	-0.013840	-0.024993	-0.011080
440	-0.045472	-0.044642	0.039062	0.001215	0.016318	0.015283	-0.028674	0.026560
441	-0.045472	-0.044642	-0.073030	-0.081413	0.083740	0.027809	0.173816	-0.039493

442 rows × 10 columns

In [621]:
```
df.corr()
```
executed in 13ms, finished 17:31:36 2023-10-24

Out[621]:

	age	sex	bmi	bp	s1	s2	s3	s4
age	1.000000	0.173737	0.185085	0.335428	0.260061	0.219243	-0.075181	0.203841
sex	0.173737	1.000000	0.088161	0.241010	0.035277	0.142637	-0.379090	0.332115
bmi	0.185085	0.088161	1.000000	0.395411	0.249777	0.261170	-0.366811	0.413807
bp	0.335428	0.241010	0.395411	1.000000	0.242464	0.185548	-0.178762	0.257650
s1	0.260061	0.035277	0.249777	0.242464	1.000000	0.896663	0.051519	0.542207
s2	0.219243	0.142637	0.261170	0.185548	0.896663	1.000000	-0.196455	0.659817
s3	-0.075181	-0.379090	-0.366811	-0.178762	0.051519	-0.196455	1.000000	-0.738493
s4	0.203841	0.332115	0.413807	0.257650	0.542207	0.659817	-0.738493	1.000000
s5	0.270774	0.149916	0.446157	0.393480	0.515503	0.318357	-0.398577	0.617859
s6	0.301731	0.208133	0.388680	0.390430	0.325717	0.290600	-0.273697	0.417212

- 상관성이 각각 너무 낮다.

In [622]:
```
df1 = df.iloc[:,[0,3,9]]
```
executed in 13ms, finished 17:31:36 2023-10-24

In [623]:
```
raw2 = df1.to_numpy()
```
executed in 14ms, finished 17:31:36 2023-10-24

In [624]:
```
train_input , test_input , train_target , test_target = train_test_split(raw , di
```
executed in 14ms, finished 17:31:36 2023-10-24

In [625]:
```
lr = LinearRegression()
lr.fit(train_input , train_target)
```
executed in 14ms, finished 17:31:36 2023-10-24

Out[625]:
```
▾ LinearRegression

LinearRegression()
```

In [626]:
```
lr.score(train_input , train_target)
```
executed in 14ms, finished 17:31:36 2023-10-24

Out[626]:
```
0.5190341891679049
```

In [627]:
```
# 변수를 늘려보기
poly = PolynomialFeatures(include_bias = False , degree = 3)
train_poly = poly.fit_transform(train_input)
test_poly = poly.transform(test_input)
```
executed in 15ms, finished 17:31:36 2023-10-24

In [628]:
```
lr.fit(train_poly , train_target)
print(lr.score(train_poly , train_target))
print(lr.score(test_poly , test_target))
```
executed in 29ms, finished 17:31:36 2023-10-24

```
0.90983817573718
-55.921870574233004
```

- 과대적합 , 릿지와 라쏘 활용해보기

In [629]:
```
from sklearn.linear_model import Ridge
ridge = Ridge()
ridge.fit(train_input , train_target)
print(ridge.score(train_input , train_target))
print(ridge.score(test_input , test_target))
```
executed in 12ms, finished 17:31:36 2023-10-24

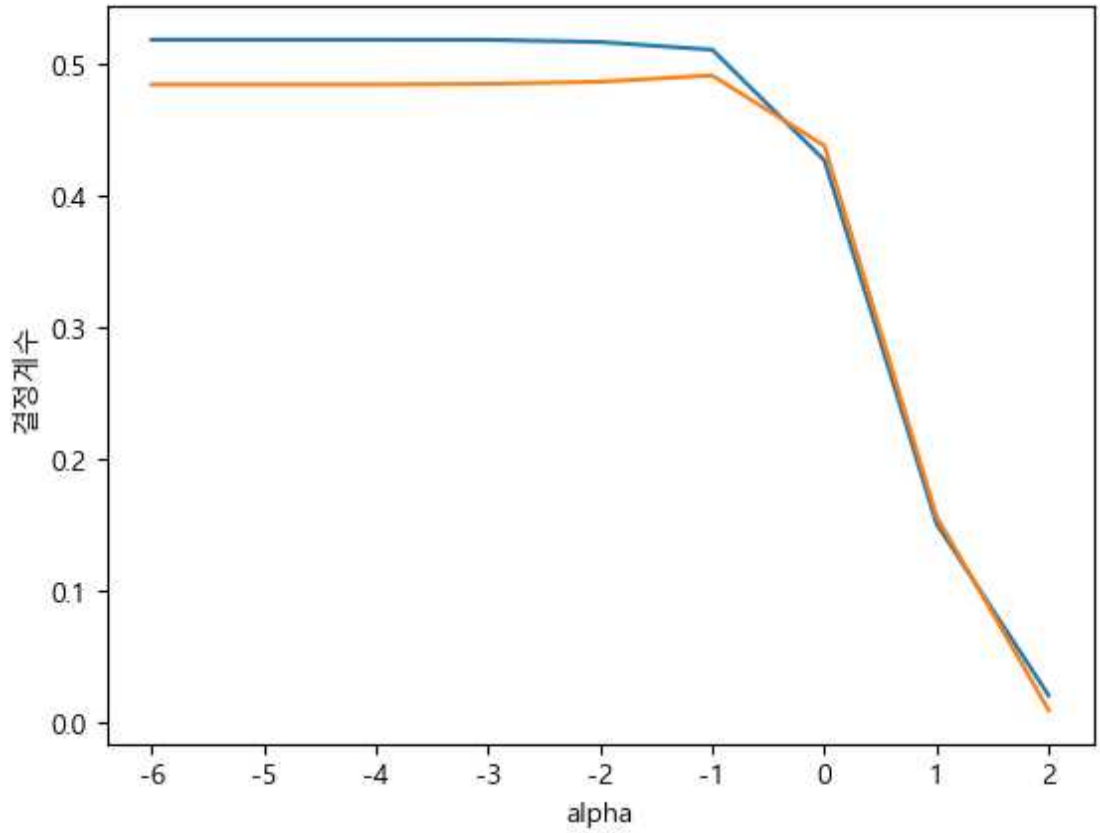
```
0.42730211000433205
0.4384002973142447
```

In [630]:
```
train_score = []
test_score = []
for i in [0.000001 , 0.00001 , 0.0001 , 0.001 , 0.01 , 0.1 , 1 , 10 , 100]:
    ridge = Ridge(alpha = i)
    ridge.fit(train_input , train_target)
    train_score.append(ridge.score(train_input , train_target))
    test_score.append(ridge.score(test_input , test_target))
```

executed in 23ms, finished 17:31:36 2023-10-24

In [631]:
```
plt.plot(np.log10([0.000001 , 0.00001 , 0.0001 , 0.001 , 0.01 , 0.1 , 1 , 10 , 10
plt.plot(np.log10([0.000001 , 0.00001 , 0.0001 , 0.001 , 0.01 , 0.1 , 1 , 10 , 10
plt.xlabel('alpha')
plt.ylabel('결정계수')
plt.show()
```

executed in 154ms, finished 17:31:36 2023-10-24



In [632]:
```
from sklearn.linear_model import Lasso
lasso = Lasso()
lasso.fit(train_input , train_target)
print(lasso.score(train_input , train_target))
print(lasso.score(test_input , test_target))
```

executed in 13ms, finished 17:31:36 2023-10-24

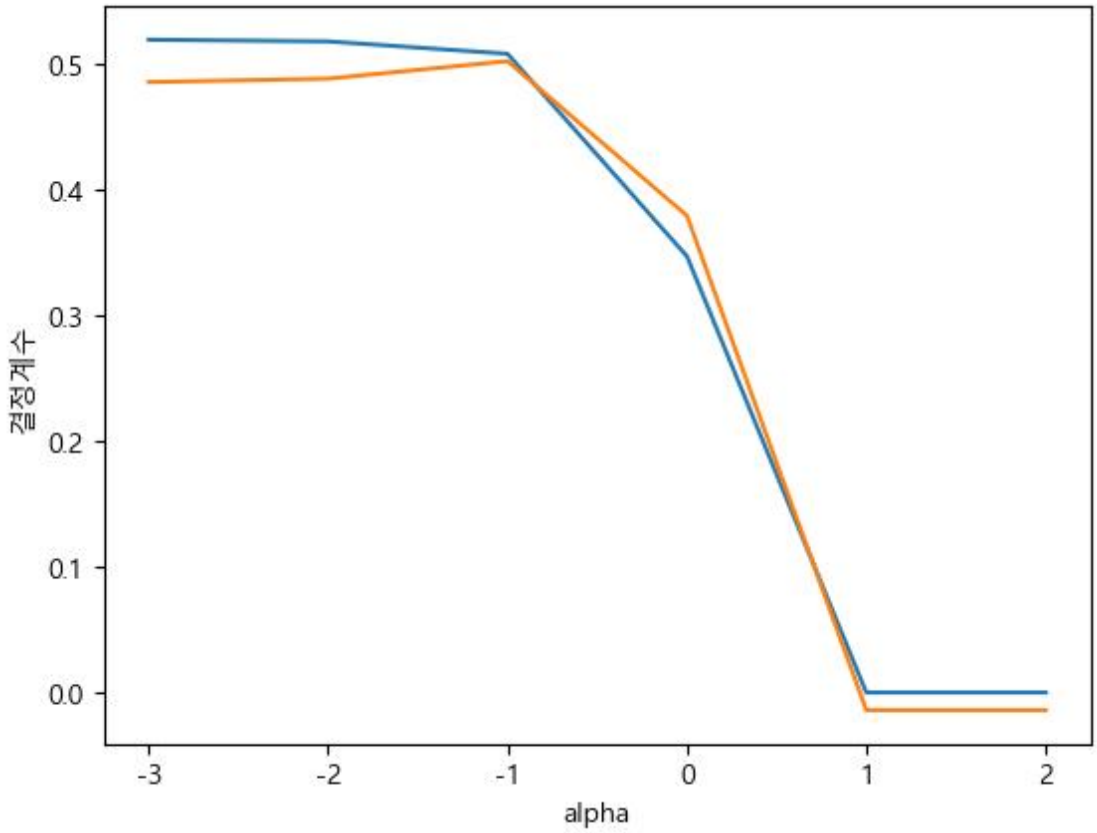
```
0.34687336241711
0.3791413953419158
```

In [633]:
```
train_score = []
test_score = []
for i in [0.001 , 0.01 , 0.1 , 1 , 10 , 100]:
    lasso = Lasso(alpha = i)
    lasso.fit(train_input , train_target)
    train_score.append(lasso.score(train_input , train_target))
    test_score.append(lasso.score(test_input , test_target))
```

executed in 29ms, finished 17:31:36 2023-10-24

In [634]:
```
plt.plot(np.log10([0.001 , 0.01 , 0.1 , 1 , 10 , 100]) , train_score)
plt.plot(np.log10([0.001 , 0.01 , 0.1 , 1, 10 , 100]) , test_score)
plt.xlabel('alpha')
plt.ylabel('결정계수')
plt.show()
```

executed in 123ms, finished 17:31:36 2023-10-24



In [635]:
```
from sklearn.linear_model import Lasso
lasso = Lasso(alpha = 0.1)
lasso.fit(train_input , train_target)
print(lasso.score(train_input , train_target))
print(lasso.score(test_input , test_target))
```

executed in 17ms, finished 17:31:36 2023-10-24

```
0.507826468555518
0.5019753487835408
```

In [636]:
```
df.corr()
```
executed in 27ms, finished 17:31:36 2023-10-24

Out[636]:

	age	sex	bmi	bp	s1	s2	s3	s4
age	1.000000	0.173737	0.185085	0.335428	0.260061	0.219243	-0.075181	0.203841
sex	0.173737	1.000000	0.088161	0.241010	0.035277	0.142637	-0.379090	0.332115
bmi	0.185085	0.088161	1.000000	0.395411	0.249777	0.261170	-0.366811	0.413807
bp	0.335428	0.241010	0.395411	1.000000	0.242464	0.185548	-0.178762	0.257650
s1	0.260061	0.035277	0.249777	0.242464	1.000000	0.896663	0.051519	0.542207
s2	0.219243	0.142637	0.261170	0.185548	0.896663	1.000000	-0.196455	0.659817
s3	-0.075181	-0.379090	-0.366811	-0.178762	0.051519	-0.196455	1.000000	-0.738493
s4	0.203841	0.332115	0.413807	0.257650	0.542207	0.659817	-0.738493	1.000000
s5	0.270774	0.149916	0.446157	0.393480	0.515503	0.318357	-0.398577	0.617859
s6	0.301731	0.208133	0.388680	0.390430	0.325717	0.290600	-0.273697	0.417212

2 보스턴 집값 예측

In [637]:
```
X , y = mglearn.datasets.load_extended_boston()
train_input , test_input , train_target , test_target = train_test_split(X , y ,
```
executed in 954ms, finished 17:31:37 2023-10-24

In [638]:
```
# 데이터를 보았을 때 , 0의 개수가 너무 많은 것들을 제거
x = pd.DataFrame(X).drop([1,3,8] , axis = 1).to_numpy()
```
executed in 16ms, finished 17:31:37 2023-10-24

In [639]:
```
#상관계수 확인
pd.DataFrame(x).corr()
```
executed in 33ms, finished 17:31:37 2023-10-24

Out[639]:

	0	1	2	3	4	5	6	7
0	1.000000	0.406583	0.420972	-0.219247	0.352734	-0.379670	0.582764	0.289946
1	0.406583	1.000000	0.763651	-0.391676	0.644779	-0.708027	0.720760	0.383248
2	0.420972	0.763651	1.000000	-0.302188	0.731470	-0.769230	0.668023	0.188933
3	-0.219247	-0.391676	-0.302188	1.000000	-0.240265	0.205246	-0.292048	-0.355501
4	0.352734	0.644779	0.731470	-0.240265	1.000000	-0.747881	0.506456	0.261515
...	...	...	...	...	...	...	...	...
96	-0.027768	0.110998	-0.062746	-0.236286	0.052092	-0.015479	0.087699	0.719030
97	0.506050	0.600477	0.495391	-0.542275	0.558305	-0.481751	0.630946	0.658179
98	-0.372994	-0.387507	-0.411609	0.153335	-0.298176	0.310894	-0.454404	-0.164704
99	0.220731	0.423440	0.390434	-0.569364	0.469398	-0.345454	0.293140	0.270669
100	0.458726	0.521482	0.521016	-0.534571	0.511949	-0.440135	0.486444	0.292706

101 rows × 101 columns

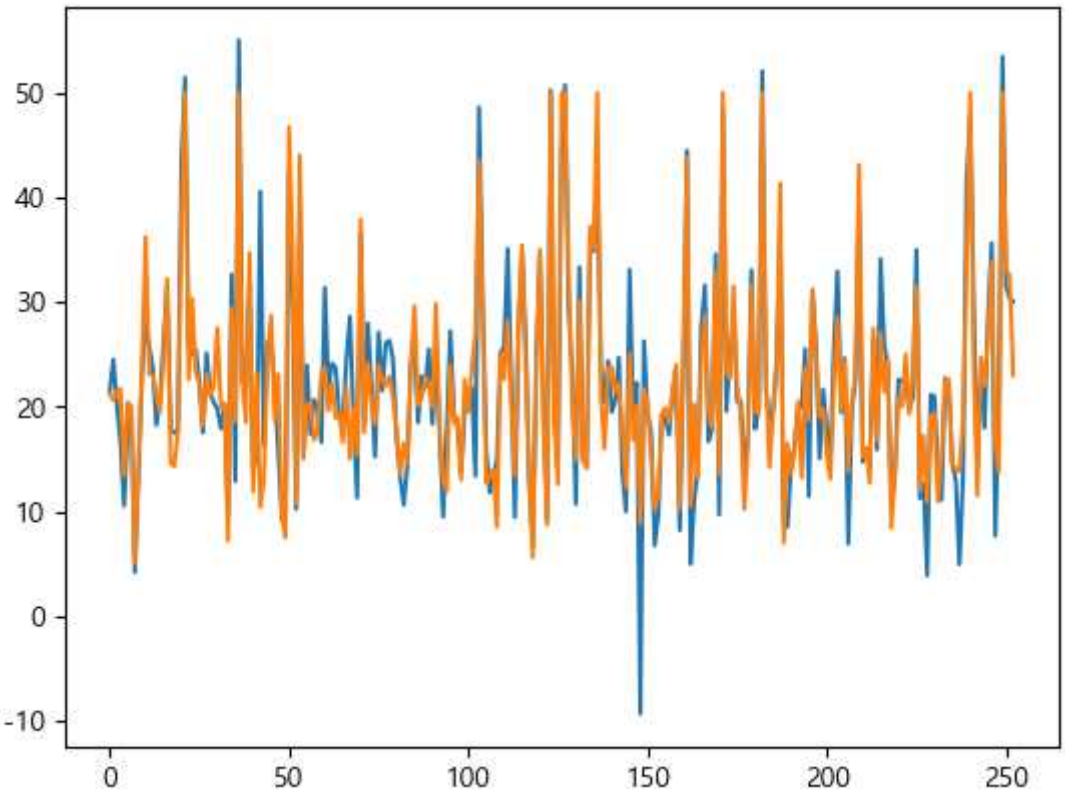
In [640]:
```
#test 데이터의 점수가 너무 낮아 , size를 0.5로 올려보았다.
train_input , test_input , train_target , test_target = train_test_split(x , y ,
```
executed in 13ms, finished 17:31:37 2023-10-24

In [641]:
```
from sklearn.linear_model import LinearRegression
lr = LinearRegression()
lr.fit(train_input , train_target)
```
executed in 28ms, finished 17:31:37 2023-10-24

Out[641]:
LinearRegression
LinearRegression()

In [642]:
```
test = lr.predict(test_input)
plt.plot(test)
plt.plot(test_target)
plt.show()
```

executed in 142ms, finished 17:31:37 2023-10-24



In [643]:
```
lr.score(train_input , train_target)
```

executed in 7ms, finished 17:31:37 2023-10-24

Out[643]: 0.9446400898336168

In [644]:
```
lr.score(test_input , test_target)
```

executed in 15ms, finished 17:31:37 2023-10-24

Out[644]: 0.7955911953641852

In [645]:
```
poly = PolynomialFeatures(include_bias = False)
train_poly = poly.fit_transform(train_input)
test_poly = poly.transform(test_input)
```

executed in 43ms, finished 17:31:37 2023-10-24

In [646]:
```
lr.fit(train_poly , train_target)
print(lr.score(train_poly , train_target))
print(lr.score(test_poly , test_target))
```

executed in 53ms, finished 17:31:38 2023-10-24

```
1.0
-1.8050261418672213
```

In [647]:
```
from sklearn.linear_model import Ridge
ridge = Ridge()
ridge.fit(train_input , train_target)
print(ridge.score(train_input , train_target))
print(ridge.score(test_input , test_target))
```

executed in 13ms, finished 17:31:38 2023-10-24

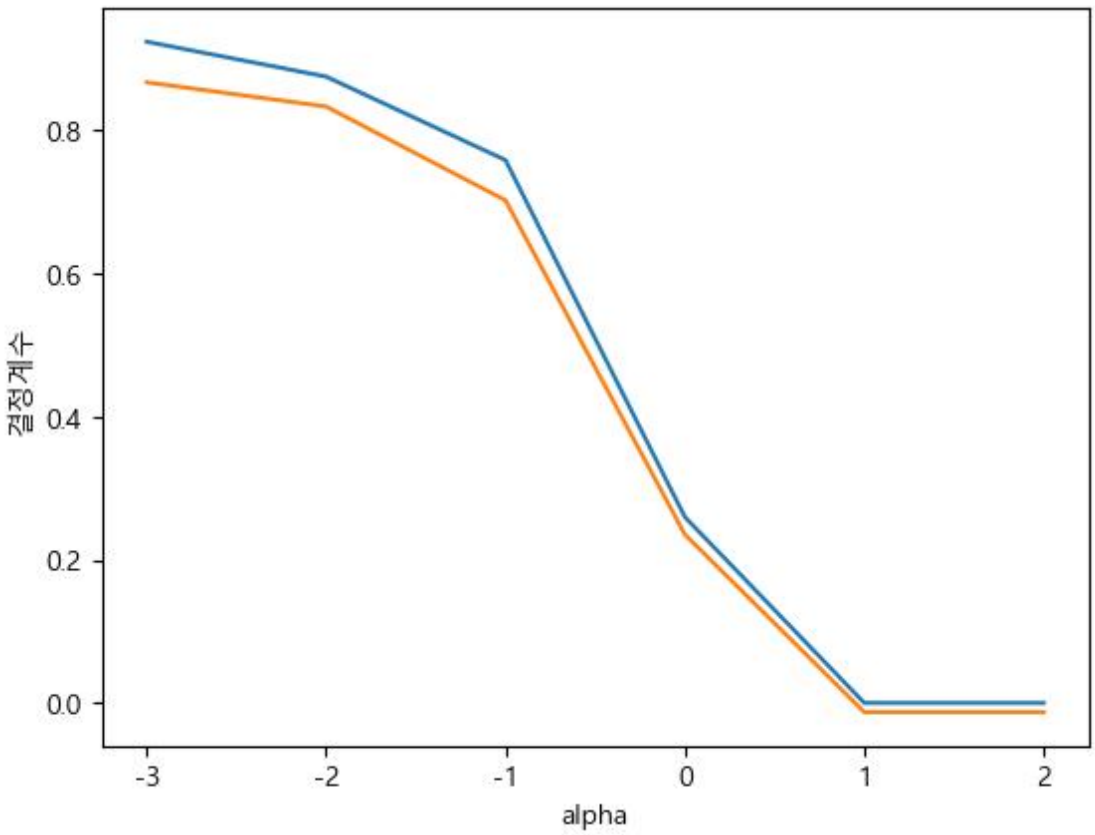
```
0.8533197390538276
0.8107040970157519
```

In [648]:
```
train_score = []
test_score = []
for i in [0.001 , 0.01 , 0.1 , 1 , 10 , 100]:
    lasso = Lasso(alpha = i)
    lasso.fit(train_input , train_target)
    train_score.append(lasso.score(train_input , train_target))
    test_score.append(lasso.score(test_input , test_target))
```

executed in 51ms, finished 17:31:38 2023-10-24

In [649]:
```
plt.plot(np.log10([0.001 , 0.01 , 0.1 , 1 , 10 , 100]) , train_score)
plt.plot(np.log10([0.001 , 0.01 , 0.1 , 1, 10 , 100]) , test_score)
plt.xlabel('alpha')
plt.ylabel('결정계수')
plt.show()
```

executed in 130ms, finished 17:31:38 2023-10-24



In [650]:
```
from sklearn.linear_model import Lasso
lasso = Lasso()
lasso.fit(train_input , train_target)
print(lasso.score(train_input , train_target))
print(lasso.score(test_input , test_target))
```

executed in 19ms, finished 17:31:38 2023-10-24

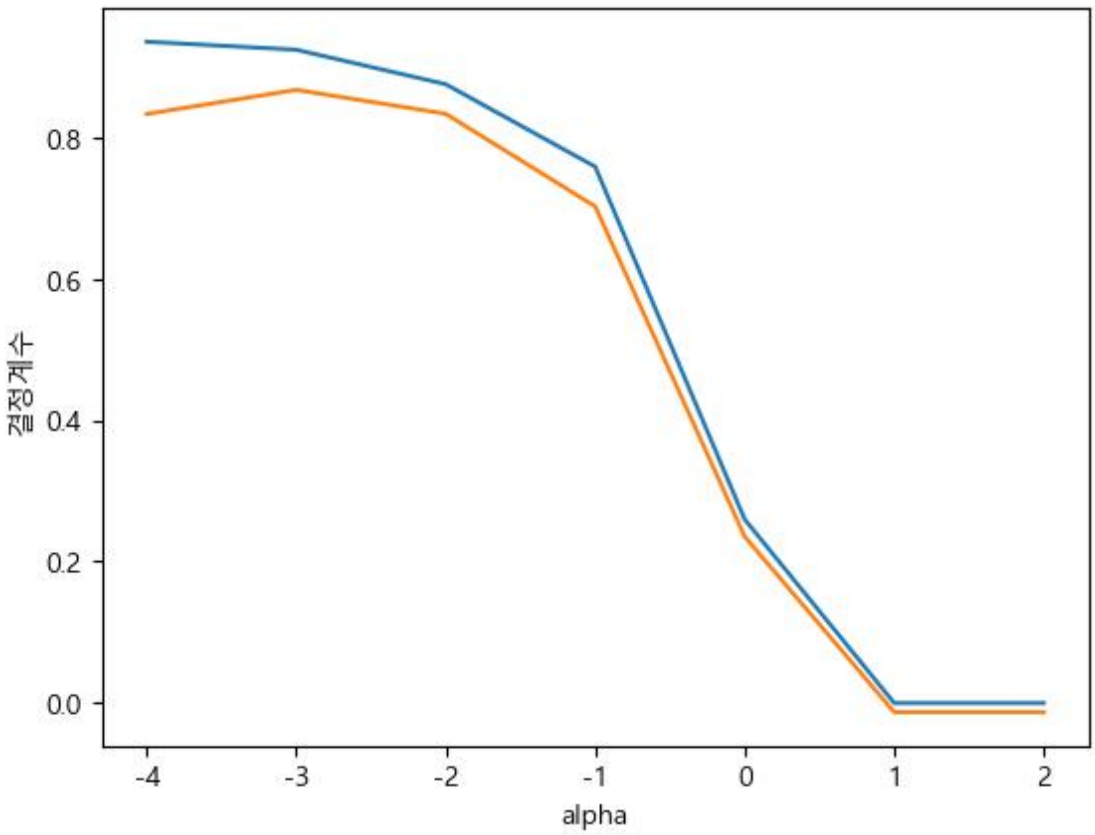
```
0.25971564078597364
0.23526920018529962
```

In [651]:
```
train_score = []
test_score = []
for i in [0.0001 , 0.001 , 0.01 , 0.1 , 1 , 10 , 100]:
    lasso = Lasso(alpha = i)
    lasso.fit(train_input , train_target)
    train_score.append(lasso.score(train_input , train_target))
    test_score.append(lasso.score(test_input , test_target))
```

executed in 49ms, finished 17:31:38 2023-10-24

In [652]:
```
plt.plot(np.log10([0.0001 , 0.001 , 0.01 , 0.1 , 1 , 10 , 100]) , train_score)
plt.plot(np.log10([0.0001 , 0.001 , 0.01 , 0.1 , 1, 10 , 100]) , test_score)
plt.xlabel('alpha')
plt.ylabel('결정계수')
plt.show()
```

executed in 134ms, finished 17:31:38 2023-10-24



In [653]:
```
from sklearn.linear_model import Lasso
lasso = Lasso(alpha = 0.001)
lasso.fit(train_input , train_target)
print(lasso.score(train_input , train_target))
print(lasso.score(test_input , test_target))
```

executed in 32ms, finished 17:31:38 2023-10-24

```
0.9237251470881582
0.8669762750880734
```

3 결론

- 당뇨병 수치 판단 자료는 점수가 너무 낮게 나온다.(최대점수 0.5)
- 보스턴은 0.9237251470881582 , 0.8669762750880734의 점수였다. 과대적합이 일어났다고 볼 수 있다.

In [654]:
```
#boston 데이터 불러오기
boston = pd.read_csv('BostonHousing.csv')
```

executed in 16ms, finished 17:31:38 2023-10-24

In [655]:
```
#집값을 제외한 것들
boston.iloc[:,:-1]
```

executed in 43ms, finished 17:31:38 2023-10-24

Out[655]:

	crim	zn	indus	chas	nox	rm	age	dis	rad	tax	ptratio	b	lstat
0	0.00632	18.0	2.31	0	0.538	6.575	65.2	4.0900	1	296	15.3	396.90	4.98
1	0.02731	0.0	7.07	0	0.469	6.421	78.9	4.9671	2	242	17.8	396.90	9.14
2	0.02729	0.0	7.07	0	0.469	7.185	61.1	4.9671	2	242	17.8	392.83	4.03
3	0.03237	0.0	2.18	0	0.458	6.998	45.8	6.0622	3	222	18.7	394.63	2.94
4	0.06905	0.0	2.18	0	0.458	7.147	54.2	6.0622	3	222	18.7	396.90	5.33
...	...	...	...	...	...	...	...	...	...	...	...	...	...
501	0.06263	0.0	11.93	0	0.573	6.593	69.1	2.4786	1	273	21.0	391.99	9.67
502	0.04527	0.0	11.93	0	0.573	6.120	76.7	2.2875	1	273	21.0	396.90	9.08
503	0.06076	0.0	11.93	0	0.573	6.976	91.0	2.1675	1	273	21.0	396.90	5.64
504	0.10959	0.0	11.93	0	0.573	6.794	89.3	2.3889	1	273	21.0	393.45	6.48
505	0.04741	0.0	11.93	0	0.573	6.030	80.8	2.5050	1	273	21.0	396.90	7.88

506 rows × 13 columns

In [656]:
```
#집값을 제외한 것들과 집값을 훈련 , 테스트로 나누기
train_input , test_input , train_target , test_target = train_test_split(boston.i
```

executed in 7ms, finished 17:31:38 2023-10-24

In [657]:
```
poly = PolynomialFeatures(include_bias = False , degree = 5)
train_poly = poly.fit_transform(train_input)
test_poly = poly.transform(test_input)
```
executed in 31ms, finished 17:31:38 2023-10-24

In [658]:
```
lr.fit(train_poly , train_target)
print(lr.score(train_poly , train_target))
print(lr.score(test_poly , test_target))
```
executed in 132ms, finished 17:31:38 2023-10-24

```
1.0
-544.4273353229181
```

In [659]:
```
# 평균과 표준편차를 이용하여 특성의 스케일을 표준점수로 변경
from sklearn.preprocessing import StandardScaler
ss = StandardScaler()
ss.fit(train_poly)
train_scaled = ss.transform(train_poly)
test_scaled = ss.transform(test_poly)
```
executed in 67ms, finished 17:31:38 2023-10-24

In [660]:
```
from sklearn.linear_model import LinearRegression
lr = LinearRegression()
lr.fit(train_scaled , train_target)
```
executed in 147ms, finished 17:31:38 2023-10-24

Out[660]:
```
▾ LinearRegression

LinearRegression()
```

In [661]:
```
lr.score(train_scaled , train_target)
```
executed in 13ms, finished 17:31:38 2023-10-24

Out[661]:
```
1.0
```

In [662]:
```
lr.score(test_scaled , test_target)
```
executed in 14ms, finished 17:31:38 2023-10-24

Out[662]:
```
-7.858387605502939
```