로지스틱 회귀

구글 코랩에서 실행하기

럭키백의 확률

데이터 준비하기

In [1]:

```
import pandas as pd

fish = pd.read_csv('https://bit.ly/fish_csv_data')
fish.head()
#Species -> y
```

Out[1]:

	Species	Weight	Length	Diagonal	Height	Width
0	Bream	242.0	25.4	30.0	11.5200	4.0200
1	Bream	290.0	26.3	31.2	12.4800	4.3056
2	Bream	340.0	26.5	31.1	12.3778	4.6961
3	Bream	363.0	29.0	33.5	12.7300	4.4555
4	Bream	430.0	29.0	34.0	12.4440	5.1340

In [2]:

```
print(pd.unique(fish['Species'])) # species 열에서 고유한 값을 출력
```

```
['Bream' 'Roach' 'Whitefish' 'Parkki' 'Perch' 'Pike' 'Smelt']
```

In [3]:

```
fish_input = fish[['Weight','Length','Diagonal','Height','Width']].to_numpy() # 나머지 열을 feature
```

In [4]:

```
print(fish_input[:5])
```

```
[[242.      25.4     30.      11.52     4.02  ]
 [290.      26.3     31.2     12.48     4.3056]
 [340.      26.5     31.1     12.3778   4.6961]
 [363.      29.      33.5     12.73     4.4555]
 [430.      29.      34.      12.444    5.134 ]]
```

In [5]:

```
fish_target = fish['Species'].to_numpy() # target
```

In [6]:

```
from sklearn.model_selection import train_test_split

train_input, test_input, train_target, test_target = train_test_split(
    fish_input, fish_target, random_state=42) # 학습, 테스트 데이터 분할
```

In [7]:

```
from sklearn.preprocessing import StandardScaler
# 훈련, 테스트 데이터 표준화 전처리 진행
ss = StandardScaler()
ss.fit(train_input)
train_scaled = ss.transform(train_input)
test_scaled = ss.transform(test_input)
print(train_scaled.shape)
print(test_scaled.shape)
```

```
(119, 5)
(40, 5)
```

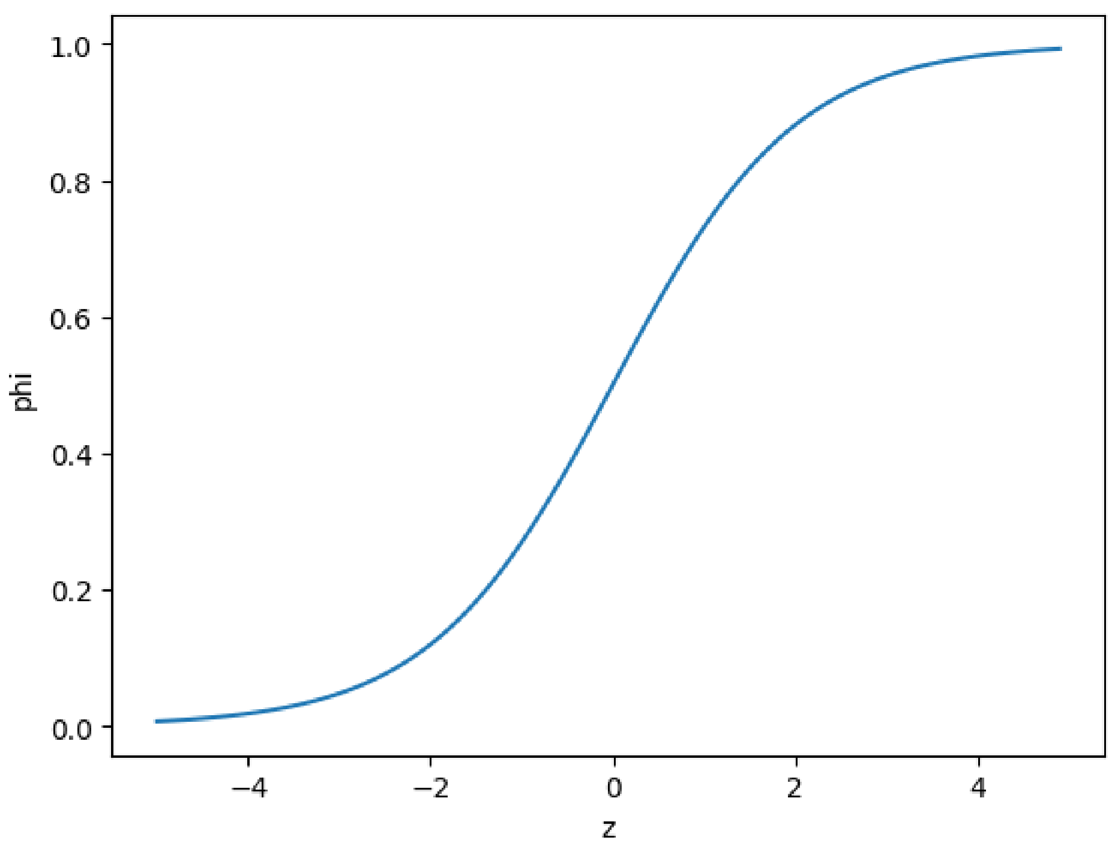
로지스틱 회귀

In [8]:

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

z = np.arange(-5, 5, 0.1)
phi = 1 / (1 + np.exp(-z))

plt.plot(z, phi)
plt.xlabel('z')
plt.ylabel('phi')
plt.show()
```



로지스틱 회귀로 이진 분류 수행하기

In [ ]:

```
char_arr = np.array(['A', 'B', 'C', 'D', 'E'])
print(char_arr[[True, False, True, False, False]])
```

```
['A' 'C']
```

In [10]:

```
bream_smelt_indexes = (train_target == 'Bream') | (train_target == 'Smelt')
# bream_smelt_indexes 비열은 훈련 세트 중 "Bream"또는  "Smelt" 일때 True 값 할 당 이외는 False
train_bream_smelt = train_scaled[bream_smelt_indexes]
target_bream_smelt = train_target[bream_smelt_indexes]
print('train_bream_smelt',train_bream_smelt.shape)
print('target_bream_smelt',target_bream_smelt.shape)
```

```
train_bream_smelt (33, 5)
target_bream_smelt (33,)
```

In [11]:

```
from sklearn.linear_model import LogisticRegression

lr = LogisticRegression()
lr.fit(train_bream_smelt, target_bream_smelt) # 모델 훈련
```

Out[11]:

```
▾ LogisticRegression
LogisticRegression()
```

In [12]:

```
print(lr.predict(train_bream_smelt[:5])) # train_bream_smelt의 처음 5개 샘플 출력
```

```
['Bream' 'Smelt' 'Bream' 'Bream' 'Bream']
```

In [13]:

```
print(lr.predict_proba(train_bream_smelt[:5])) # 처음 5개 샘플의 예측 확률 출력
# 첫번째 열이 클래스 0에 대한 확률, 두 번째 열이 클래스 1에 대한 확률
```

```
[[0.99759855 0.00240145]
 [0.02735183 0.97264817]
 [0.99486072 0.00513928]
 [0.98584202 0.01415798]
 [0.99767269 0.00232731]]
```

In [14]:

```
print(lr.classes_)
```

```
['Bream' 'Smelt']
```

In [15]:

```
print(lr.coef_, lr.intercept_)
```

```
[[-0.4037798  -0.57620209 -0.66280298 -1.01290277 -0.73168947]] [-2.16155132]
```

In [16]:

```
decisions = lr.decision_function(train_bream_smelt[:5]) # sigmoid 값에 들어가기전 z 값
print(decisions)
```

```
[-6.02927744  3.57123907 -5.26568906 -4.24321775 -6.0607117 ]
```

In [17]:

```
from scipy.special import expit

print(expit(decisions))
```

```
[0.00240145 0.97264817 0.00513928 0.01415798 0.00232731]
```

로지스틱 회귀로 다중 분류 수행하기

In [18]:

```
lr = LogisticRegression(C=20, max_iter=1000)
# max_iter=1000으로 반복 횟수를 1000으로 설정
# logisticRegression에서 규제를 제어하는 매개변수 C, C가 커지면 alpha값은 감소
lr.fit(train_scaled, train_target)

print(lr.score(train_scaled, train_target))
print(lr.score(test_scaled, test_target))
```

```
0.9327731092436975
0.925
```

In [20]:

```
print(lr.predict(test_scaled[:5]))
```

```
['Perch' 'Smelt' 'Pike' 'Roach' 'Perch']
```

In [21]:

```
proba = lr.predict_proba(test_scaled[:5]) # 확률값 출력
print(np.round(proba, decimals=3)) # 소수점 4째 자리에서 반올림
```

```
[[0.    0.014 0.841 0.    0.136 0.007 0.003]
 [0.    0.003 0.044 0.    0.007 0.946 0.   ]
 [0.    0.    0.034 0.935 0.015 0.016 0.   ]
 [0.011 0.034 0.306 0.007 0.567 0.    0.076]
 [0.    0.    0.904 0.002 0.089 0.002 0.001]]
```

In [22]:

```
print(lr.classes_)
```

```
['Bream' 'Parkki' 'Perch' 'Pike' 'Roach' 'Smelt' 'Whitefish']
```

In [23]:

```
print(lr.coef_.shape, lr.intercept_.shape)
```

```
(7, 5) (7,)
```

In [25]:

```
decision = lr.decision_function(test_scaled[:5]) # z값
print(np.round(decision, decimals=2)) # 소수점 3째 자리에서 반올림
```

```
[[ -6.5    1.03   5.16  -2.73   3.34   0.33  -0.63]
 [-10.86   1.93   4.77  -2.4    2.98   7.84  -4.26]
 [ -4.34  -6.23   3.17   6.49   2.36   2.42  -3.87]
 [ -0.68   0.45   2.65  -1.19   3.26  -5.75   1.26]
 [ -6.4   -1.99   5.82  -0.11   3.5   -0.11  -0.71]]
```

In [27]:

```
from scipy.special import softmax

proba = softmax(decision, axis=1) # 즉 샘플에 대해 소프트 맥스 계산
print(np.round(proba, decimals=3))
```

```
[[0.    0.014 0.841 0.    0.136 0.007 0.003]
 [0.    0.003 0.044 0.    0.007 0.946 0.   ]
 [0.    0.    0.034 0.935 0.015 0.016 0.   ]
 [0.011 0.034 0.306 0.007 0.567 0.    0.076]
 [0.    0.    0.904 0.002 0.089 0.002 0.001]]
```

In [ ]: