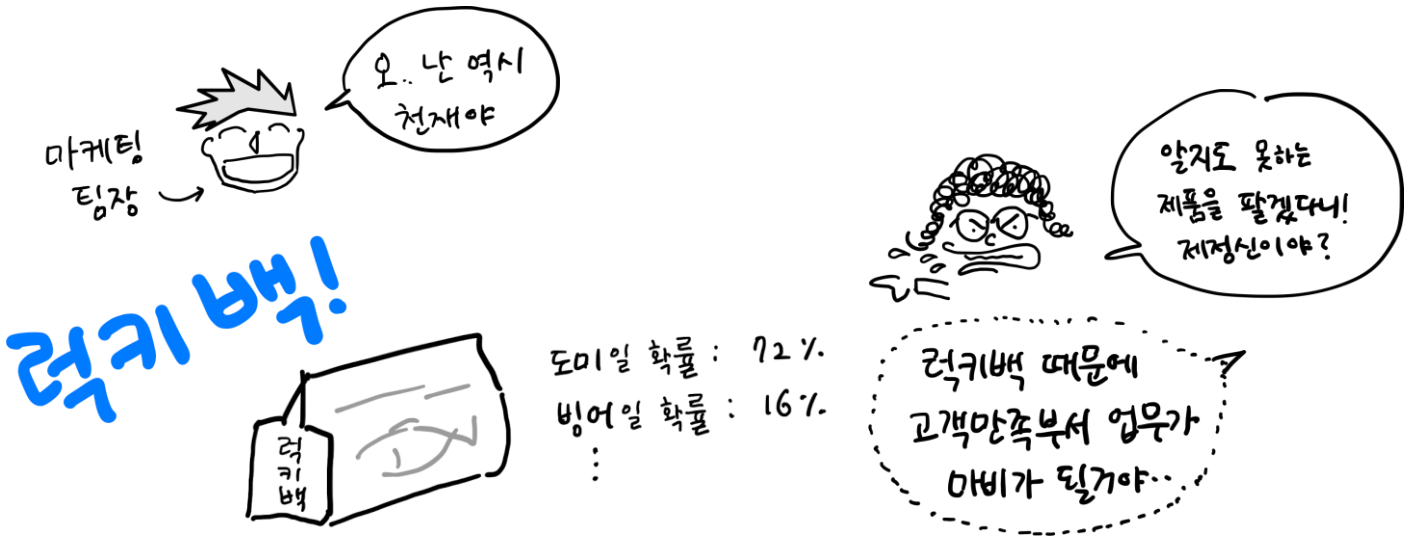
로지스틱 회귀분석-이진분류

로지스틱 회귀분석

Python

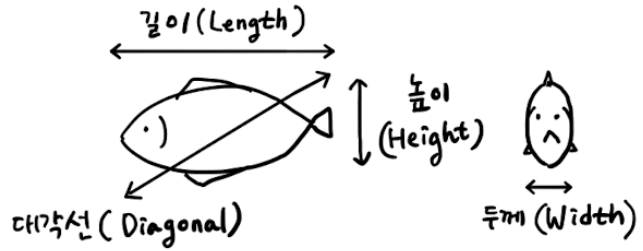
내용물은 모르는 제품 판매

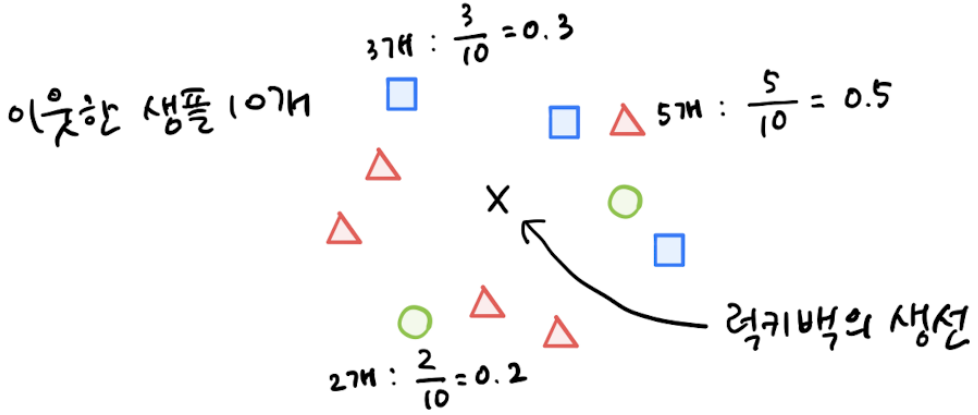


회귀 알고리즘

Python

확률 계산하기





회귀 알고리즘

Python

데이터 준비

```
import pandas as pd

fish = pd.read_csv('https://bit.ly/fish_csv_data')
fish.head()
```

	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

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

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

회귀 알고리즘

Python

데이터 전처리

```
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)


from sklearn.preprocessing import StandardScaler

ss = StandardScaler()
ss.fit(train_input)
train_scaled = ss.transform(train_input)
test_scaled = ss.transform(test_input)
```

회귀 알고리즘

Python

K-최근접 이웃의 다중 분류

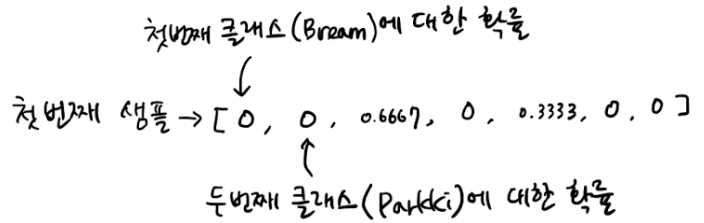
```
from sklearn.neighbors import KNeighborsClassifier

kn = KNeighborsClassifier(n_neighbors=3)
kn.fit(train_scaled, train_target)

print(kn.classes_)
['Bream' 'Parkki' 'Perch' 'Pike' 'Roach' 'Smelt' 'Whitefish']

print(kn.predict(test_scaled[:5]))
['Perch' 'Smelt' 'Pike' 'Perch' 'Perch']

proba = kn.predict_proba(test_scaled[:5])
print(np.round(proba, decimals=4))
[[0.     0.     1.     0.     0.     0.     0.    ]
 [0.     0.     0.     0.     0.     1.     0.    ]
 [0.     0.     0.     1.     0.     0.     0.    ]
 [0.     0.     0.6667 0.     0.3333 0.     0.    ]
 [0.     0.     0.6667 0.     0.3333 0.     0.    ]]
```



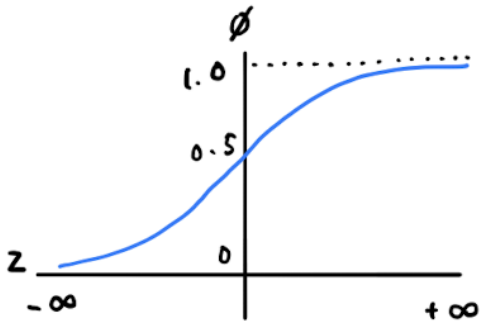
회귀 알고리즘

Python

로지스틱 회귀

$$z = a \times \text{무게} + b \times \text{길이} + c \times \text{대각선} + d \times \text{높이} + e \times \text{두께} + f$$

$$\phi = \frac{1}{1 + e^{-z}}$$



회귀 알고리즘

Python

로지스틱 회귀(이진 분류)

```
bream_smelt_indexes = (train_target == 'Bream') | (train_target == 'Smelt')
train_bream_smelt = train_scaled[bream_smelt_indexes]
target_bream_smelt = train_target[bream_smelt_indexes]

from sklearn.linear_model import LogisticRegression

lr = LogisticRegression()
lr.fit(train_bream_smelt, target_bream_smelt)

print(lr.predict(train_bream_smelt[:5]))
['Bream' 'Smelt' 'Bream' 'Bream' 'Bream']

print(lr.predict_proba(train_bream_smelt[:5]))
[[0.99759855 0.00240145]
 [0.02735183 0.97264817]
 [0.99486072 0.00513928]
 [0.98584202 0.01415798]
 [0.99767269 0.00232731]]
```

회귀 알고리즘

Python

로지스틱 회귀 계수 확인

```
print(lr.coef_, lr.intercept_)
[[-0.4037798  -0.57620209 -0.66280298 -1.01290277 -0.73168947]] [-2.16155132]
```

$$z = -0.404 \times \text{무게} - 0.576 \times \text{길이} - 0.663 \times \text{대각선} - 0.013 \times \text{높이} - 0.732 \times \text{두께} - 2.161$$

```
decisions = lr.decision_function(train_bream_smelt[:5])
print(decisions)
[-6.02927744  3.57123907 -5.26568906 -4.24321775 -6.0607117 ]

from scipy.special import expit
print(expit(decisions))
[0.00240145 0.97264817 0.00513928 0.01415798 0.00232731]
```

감사합니다.