2

20249132

Question 1

1. resign.csv 자료는 어느 회사의 인사관리에 관한 데이터로 각 변수의 의미는 다음과 같다. 물음에 답하여라.

타겟 변수
- resign : 각 직원의 퇴사여부(0: 퇴사하지 않음, 1:퇴사함)

특성 변수
- satisfaction : 업무만족도
- evaluation : 인사평점
- project : 프로젝트 수행횟수
- workhour : 월평균업무시간
- years : 근속연수
- accident : 사고 여부
- promotion : 최근 5년 이내 승진여부

```
import numpy as np
import pandas as pd
from sklearn.preprocessing import StandardScaler
from sklearn.model_selection import GridSearchCV
from sklearn.linear_model import LogisticRegression

data = pd.read_csv("  hw2_resign.csv")
```

(1) 각 특성변수 중 0 또는 1의 값을 가지는 더미 변수들을 제외한 나머지 모든 숫자형 변수들은 모두 표준화(standardized) 변환을 적용하여라.

(2) (1)에서 전처리된 데이터 전체가 훈련용 데이터셋이라고 가정하고, sklearn을 이용하여 resign 를 목표변수로 하는 이항 로지스틱 회귀모형을 훈련하여라. 단, ridge 규제를 적용하되 규제의 정도를 나타내는 하이퍼파라미터(sklearn의 LogisticRegression 알고리즘에서의 'C')는 [0.0001, 0.0005, 0.001, 0.003, 0.005] 중 하나의 값을 5-fold 교차검증 방식으로 선택해야 함. 또한 튜닝 시 sklearn의 GridSearchCV 함수를 활용해야 하며, 교차검증에서의 평가지표는 분류정확도('accuracy')를 적용해야 함.

(3) (2)에서 추정된 파라미터를 이용하여, 훈련된 이항 로지스틱 회귀모형을 식으로 표현하여라.

(4) (2)의 훈련 결과를 이용하여, 아래와 같은 특성을 가지는 어느 직원이 퇴사할 확률을 구하여라.

satisfaction = 0.5, evaluation = 0.5, project = 5, workhour = 250, years = 5, accident = 0, promotion = 0

(5) 다른 모든 특성변수 값이 동일하다고 가정할 때, 최근 5년 이내 승진했던 적이 있는 직원은 그렇지 않은 직원에 비해 퇴사가능성에 대한 오즈(odds)가 몇 배인지 구하여라.

```
# (1)
print('----- Before Standardization -----')
print(pd.DataFrame(pd.concat([data.min(),data.max(),data.mean(),data.std()],axis=1)))

scaler = StandardScaler()
standardization = ['satisfaction', 'evaluation', 'project', 'workhour', 'years']
data[standardization] = scaler.fit_transform(data[standardization])

print('----- After Standardization -----')
print(pd.DataFrame(pd.concat([data.min(),data.max(),data.mean(),data.std()],axis=1)))
```

```
----- Before Standardization -----
```

```
                    0      1           2          3
satisfaction     0.09    1.0    0.612834   0.248631
evaluation       0.36    1.0    0.716102   0.171169
project          2.00    7.0    3.803054   1.232592
workhour        96.00  310.0  201.050337  49.943099
years            2.00   10.0    3.498233   1.460136
accident         0.00    1.0    0.144610   0.351719
resign           0.00    1.0    0.238083   0.425924
promotion        0.00    1.0    0.021268   0.144281
----- After Standardization -----
                      0         1             2         3
satisfaction -2.102922  1.557247  2.880259e-16  1.000033
evaluation   -2.080478  1.658639 -3.903508e-16  1.000033
project      -1.462863  2.593763  5.589976e-17  1.000033
workhour     -2.103471  2.181549 -8.716572e-17  1.000033
years        -1.026126  4.452998 -6.063702e-17  1.000033
accident      0.000000  1.000000  1.446096e-01  0.351719
resign        0.000000  1.000000  2.380825e-01  0.425924
promotion     0.000000  1.000000  2.126808e-02  0.144281
```

StandardScaler .

```
# (2)           (Ridge    )
x = data.drop(columns=['resign'])
y = data['resign']

params = {'C': [0.0001, 0.0005, 0.001, 0.003, 0.005]}
model = LogisticRegression(penalty='l2')
gridCV = GridSearchCV(model, params, cv=5, scoring='accuracy')
gridCV.fit(x, y)
print(gridCV.best_estimator_)
```

```
LogisticRegression(C=0.001)
```

Ridge , .

GridSearchCV .

```
# (3)
best_model = gridCV.best_estimator_
print("Intercept:", best_model.intercept_)
print("Coefficients:", best_model.coef_)
```

```
Intercept: [-1.25192337]
Coefficients: [[-0.66725713  0.03758406 -0.1368043   0.11213996  0.2043499  -0.24128311
  -0.05154287]]
```

, Intercept , coefficients .

$\frac{1}{1+e^{1.25+0.67*sati-0.04*eval-\cdots+0.05*promot}}$.

```
# (4)
newX_numeric = scaler.transform([[0.5, 0.5, 5, 250, 5]])
newX_dummy = [[0,0]]
newX = np.concatenate([newX_numeric,newX_dummy], axis=1)
predict_Y = best_model.predict_proba(newX)
print("   :", predict_Y[0][1])
```

   : 0.3080382797323505

```
/Users/hwan/.pyenv/versions/hwan/lib/python3.10/site-packages/sklearn/base.py:493: UserWarni

X does not have valid feature names, but StandardScaler was fitted with feature names

/Users/hwan/.pyenv/versions/hwan/lib/python3.10/site-packages/sklearn/base.py:493: UserWarni

X does not have valid feature names, but LogisticRegression was fitted with feature names
```

31% .

```
# (5)    (promotion)
odds = np.exp(best_model.coef_[0][-1])
print("  :", odds)
```

  : 0.9497629280080885

, , exponential .

0.95 , 5 .

Question 2

2. 다음 코드를 실행하면 아래와 같은 array를 생성할 수 있다. 이 array의 각 열은 순서대로 절편항 1, 특징변수 X_1, X_2, X_3, X_4, 목표변수 Y를 나타내며, 목표변수의 범주는 3개(K=3)인 훈련 데이터셋이라고 가정해 보자.

```
np.random.seed(123)
traindt = np.hstack([
    np.ones((5,1)),
    np.around(np.random.randn(5,4), 3),
    np.random.randint(1, 4, (5,1)) ])
traindt
```

```
array([[ 1.   , -1.086,  0.997,  0.283, -1.506,  2.   ],
       [ 1.   , -0.579,  1.651, -2.427, -0.429,  1.   ],
       [ 1.   ,  1.266, -0.867, -0.679, -0.095,  1.   ],
       [ 1.   ,  1.491, -0.639, -0.444, -0.434,  1.   ],
       [ 1.   ,  2.206,  2.187,  1.004,  0.386,  3.   ]])
```

(1) 다음 행렬 $\boldsymbol{\Theta_1}$는 특징변수가 4개, 목표변수의 범주가 3개인 경우에 대한 소프트맥스 회귀모형 가설에서의 파라미터 행렬 $\boldsymbol{\Theta_1}$이다. $\boldsymbol{\Theta_1}$의 각 행은 목표변수의 각 범주 별 소프트맥스 파라미터 $\boldsymbol{\theta}^{(1)}, \boldsymbol{\theta}^{(2)}, \boldsymbol{\theta}^{(3)}$를 나타낸다. 이 $\boldsymbol{\Theta_1}$이 주어진 훈련 자료를 분류하는데 적절한 파라미터라고 가정해 보자. 이 파라미터 행렬을 이용하여 주어진 훈련자료의 각 관찰치가 Y의 각 범주에 속할 확률을 계산한 뒤, 가장 확률이 높은 범주로 분류하여라.

$$\boldsymbol{\Theta_1} = \begin{bmatrix} -\left(\boldsymbol{\theta}^{(1)}\right)^T - \\ -\left(\boldsymbol{\theta}^{(2)}\right)^T - \\ -\left(\boldsymbol{\theta}^{(3)}\right)^T - \end{bmatrix} = \begin{bmatrix} 5 & 2 & 3 & 1 & 4 \\ 2 & 4 & 3 & 1 & 2 \\ 3 & 4 & 1 & 5 & 4 \end{bmatrix}$$

```
# (1)
data = pd.DataFrame(traindt,columns=['x0','x1','x2','x3','x4','target'])
theta = np.array([[5,2,3,1,4],[2,4,3,1,2],[3,4,1,5,4]])

y = data['target']
x = data.drop(columns='target')
```

```
s1 = (x * theta[0,:]).sum(axis=1)
s2 = (x * theta[1,:]).sum(axis=1)
s3 = (x * theta[2,:]).sum(axis=1)

total = np.exp(s1) + np.exp(s2) + np.exp(s3)

p1 = np.exp(s1) / total
p2 = np.exp(s2) / total
p3 = np.exp(s3) / total

result1 = np.concatenate([[p1],[p2],[p3]])

print('--------- Result of theta ---------')
print('Prediction : ',result1.argmax(axis=0)+1)
print('Prob : ', result1.max(axis=0))
print('Real : ', np.array(y))
```

```
--------- Result of theta ---------
Prediction :  [1 1 1 2 3]
Prob :  [0.89139444 0.96442872 0.41754961 0.47149401 0.72918736]
Real :  [2. 1. 1. 1. 3.]
```

x, y theta ,

5 1,1,1,2,3 .

2,1,1,1,3 3 , 2 .

(2) 다음 행렬 $\boldsymbol{\Theta_2}$도 $\boldsymbol{\Theta_1}$과 같은 형식으로 정의된 특징변수가 4개, 목표변수의 범주가 3개인 경우에 대한 소프트맥스 회귀모형 가설에서의 파라미터 행렬이다. (1)의 $\boldsymbol{\Theta_1}$과 (2)의 $\boldsymbol{\Theta_2}$ 중에서 주어진 훈련자료에 보다 더 적절한 파라미터 행렬은 무엇인가? 주어진 훈련자료에 대한 크로스 엔트로비 비용함수를 계산한 뒤 이를 이용하여 비교하여라.

$$\boldsymbol{\Theta_2} = \begin{bmatrix} 5.5 & 2 & 3 & 1.5 & 4 \\ 2 & 3.5 & 2.5 & 1 & 1.5 \\ 3 & 4 & 1 & 5 & 4 \end{bmatrix}$$

```
# (2)
theta2 = np.array([[5.5,2,3,1.5,4],[2,3.5,2.5,1,1.5],[3,4,1,5,4]])
```

```
s1_2 = (x * theta2[0,:]).sum(axis=1)
s2_2 = (x * theta2[1,:]).sum(axis=1)
s3_2 = (x * theta2[2,:]).sum(axis=1)

total_2 = np.exp(s1_2) + np.exp(s2_2) + np.exp(s3_2)

p1_2 = np.exp(s1_2) / total_2
p2_2 = np.exp(s2_2) / total_2
p3_2 = np.exp(s3_2) / total_2

result2 = np.concatenate([[p1_2],[p2_2],[p3_2]])

print('--------- Result of theta2 ---------')
print('Prediction : ', result2.argmax(axis=0)+1)
print('Prob : ', result2.max(axis=0))
print('Real : ', np.array(y))
```

```
--------- Result of theta2 ---------
Prediction :  [1 1 1 2 3]
Prob :  [0.87856171 0.94824032 0.47684229 0.39211219 0.72911211]
Real :  [2. 1. 1. 1. 3.]
```

5 1,1,1,2,3 .

.

```
from tensorflow.keras.utils import to_categorical
y_matrix = to_categorical( np.array(y)-1 )

cross_entropy1 = -(np.log(result1.T) * y_matrix).sum() / 5
cross_entropy2 = -(np.log(result2.T) * y_matrix).sum() / 5
print(cross_entropy1, cross_entropy2)
```

```
1.020442594489133 0.9084641929026332
```

, - .

y (one-hot) .

Θ_1 1.02, Θ_2 0.91 .

Θ_2 .