CS3-end1-p1

1-1

1-1-1 2クラスのカテゴリー変数 / 2-class category variable
1-1-2 活性化関数がシグモイド関数の単純パーセプトロン / Single layer perceptron whose activation function is a sigmoid function
1-1-3 的中率 / hit rate

1-2

Import libraries

In [13]:
```
import numpy as np
import pandas as pd
import statsmodels.api as sm
from sklearn.preprocessing import scale
```

Read CSV file

In [14]:
```
csv_in = 'end1-p1.csv'
df = pd.read_csv(csv_in, sep=',', skiprows=2, header=0)
print(df.shape)
print(df.info())
display(df.head())
```

```
(150, 7)
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 150 entries, 0 to 149
Data columns (total 7 columns):
 #   Column  Non-Null Count  Dtype
---  ------  --------------  -----
 0   c1      150 non-null    float64
 1   c2      150 non-null    float64
 2   c3      150 non-null    float64
 3   c4      150 non-null    float64
 4   c5      150 non-null    float64
 5   c6      150 non-null    float64
```

```
 6   y       150 non-null    float64
dtypes: float64(7)
memory usage: 8.3 KB
None
```

	c1	c2	c3	c4	c5	c6	y
0	2.01	2.40	2.34	1.90	1.06	-0.38	7.40
1	3.28	-0.04	1.86	0.32	1.90	0.24	4.42
2	0.23	-0.48	1.96	2.85	1.34	2.30	27.17
3	2.73	-1.44	1.90	-0.02	0.46	1.51	5.56
4	0.11	1.99	-1.54	-0.66	-0.24	1.88	-1.76

Separate explanatory variables and objective variable

説明変数と目的変数を分ける

In [15]:

```
cols = ['c1', 'c2', 'c4']
X = df[cols]  # explanatory variables, 2D
y = df['y']  # objective variable, 1D
print('X:', X.shape)
display(X.head())
print('y:', y.shape)
print(y.head())
```

```
X: (150, 3)
```

	c1	c2	c4
0	2.01	2.40	1.90
1	3.28	-0.04	0.32
2	0.23	-0.48	2.85
3	2.73	-1.44	-0.02
4	0.11	1.99	-0.66

```
y: (150,)
0      7.40
1      4.42
2     27.17
3      5.56
4     -1.76
Name: y, dtype: float64
```

MLR calculation without standardization

標準化なしで線形重回帰分析

In [16]:

```
X_c = sm.add_constant(X)
model = sm.OLS(y, X_c)
results = model.fit()
print(results.summary())
```

```
                            OLS Regression Results
==============================================================================
Dep. Variable:                      y   R-squared:                       0.848
Model:                            OLS   Adj. R-squared:                  0.845
Method:                 Least Squares   F-statistic:                     270.9
Date:                Sat, 16 Jul 2022   Prob (F-statistic):           1.95e-59
Time:                        17:56:42   Log-Likelihood:                -414.50
No. Observations:                 150   AIC:                             837.0
Df Residuals:                     146   BIC:                             849.0
Df Model:                           3
Covariance Type:            nonrobust
==============================================================================
                 coef    std err          t      P>|t|      [0.025      0.975]
------------------------------------------------------------------------------
const         10.3675      0.515     20.112      0.000       9.349      11.386
c1            -3.9070      0.196    -19.894      0.000      -4.295      -3.519
c2            -2.0897      0.216     -9.677      0.000      -2.516      -1.663
c4             4.6121      0.294     15.694      0.000       4.031       5.193
==============================================================================
Omnibus:                        0.262   Durbin-Watson:                   1.966
Prob(Omnibus):                  0.877   Jarque-Bera (JB):                0.324
Skew:                          -0.098   Prob(JB):                        0.851
Kurtosis:                       2.883   Cond. No.                         4.05
==============================================================================
```

```
Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.
```

R2 and Adjusted R2

決定係数と自由度調整済み決定係数

In [17]:
```
print('R2:', results.rsquared)
print('Adj R2:', results.rsquared_adj)
```

```
R2: 0.8476880530061681
Adj R2: 0.844558355465199
```

Ans 2. 0.845

Partial regression coefficients

偏回帰係数

In [18]:
```
print(results.params)
```

```
const    10.367487
c1       -3.906958
c2       -2.089675
c4        4.612139
dtype: float64
```

Ans 7. c4, 4.612

MLR calculation with standardization

全説明変数と目的変数を標準化して線形重回帰分析

In [19]:
```
X_scaled_ar = scale(X)
y_scaled_ar = scale(y)
```

In [20]:

```
# make DataFrames corresponding to X_scaled and y_scaled.
X_scaled = pd.DataFrame(X_scaled_ar, columns=X.columns)
y_scaled = pd.Series(y_scaled_ar, name=y.name)
model = sm.OLS(y_scaled, X_scaled)
results_scaled = model.fit()
print(results_scaled.summary())
```

```
                                 OLS Regression Results
=======================================================================================
Dep. Variable:                          y   R-squared (uncentered):                   0.848
Model:                                OLS   Adj. R-squared (uncentered):              0.845
Method:                     Least Squares   F-statistic:                              272.7
Date:                    Sat, 16 Jul 2022   Prob (F-statistic):                    7.65e-60
Time:                            17:56:42   Log-Likelihood:                         -71.704
No. Observations:                     150   AIC:                                      149.4
Df Residuals:                         147   BIC:                                      158.4
Df Model:                               3
Covariance Type:                nonrobust
==============================================================================
                 coef    std err          t      P>|t|      [0.025      0.975]
------------------------------------------------------------------------------
c1            -0.6477      0.032    -19.962      0.000      -0.712      -0.584
c2            -0.3127      0.032     -9.710      0.000      -0.376      -0.249
c4             0.5110      0.032     15.747      0.000       0.447       0.575
==============================================================================
Omnibus:                        0.262   Durbin-Watson:                   1.966
Prob(Omnibus):                  0.877   Jarque-Bera (JB):                0.324
Skew:                          -0.098   Prob(JB):                        0.851
Kurtosis:                       2.883   Cond. No.                         1.14
==============================================================================

Notes:
[1] R² is computed without centering (uncentered) since the model does not contain a constant.
[2] Standard Errors assume that the covariance matrix of the errors is correctly specified.
```

R2 and Adjusted R2

決定係数と自由度調整済み決定係数

In [21]:
```
print('R2:', results_scaled.rsquared)
print('Adj R2:', results_scaled.rsquared_adj)
```

```
R2: 0.8476880530061681
Adj R2: 0.8445796459246613
```

Ans 3. 0.845

Compare standardized partial regression coefficients

標準化偏回帰係数を比較

In [22]:
```
print(results_scaled.params.sort_values(key=np.abs, ascending=False))
```

```
c1   -0.647679
c4    0.511014
c2   -0.312669
dtype: float64
```

Ans 5. c1, c4, c2

Ans 6. c4

Do prediction with obtained model

得られたモデルを用いて、予測を行う。

In [23]:
```
X_test = pd.DataFrame([[1, 1, 1],
                       [2, 2, 2],
                      ],
                      columns=X.columns)  # example
print('X for prediction:')
display(X_test)
```

X for prediction:

	c1	c2	c4
0	1	1	1

	c1	c2	c4
1	2	2	2

In [24]:

```
X_test_c = sm.add_constant(X_test)
y_test = results.predict(X_test_c)
print('Predicted y:')
print(y_test)
```

```
Predicted y:
0    8.982992
1    7.598498
dtype: float64
```

Ans 4. y1 8.983

Ans 4. y2 7.598