소프트웨어학과 32204041 정다훈 4장 과제

In [1]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
```

실습1: Simple linear regression

1. Iris 데이터셋에서 꽃잎의 길이(Petal.Length)로 꽃잎의 폭(Petal.Width)을 예측하는 회귀 모델을 만드세요.
2. 회귀 모델의 mean_squared_error와 r2_score 값을 보이세요.
3. 회귀 모델의 coefficients와 Intercept 값을 보이세요.
4. 회귀 모델을 이용하여 꽃잎의 길이가 1.0, 1.2, 1.4일 때 꽃잎의 폭을 예측해 보세요.

In [2]:
```
iris = pd.read_csv('./data/iris.csv')
```

In [7]:
```
iris.columns
iris_X = iris['Petal.Length']
iris_Y = iris['Petal.Width']

# train test split
from sklearn.model_selection import train_test_split
X_train, X_test, Y_train, Y_test = train_test_split(iris_X, iris_Y, test_size=0.

# linear regression
from sklearn.linear_model import LinearRegression
reg = LinearRegression()
reg.fit(X_train.values.reshape(-1, 1), Y_train.values)

# predict
Y_pred = reg.predict(X_test.values.reshape(-1, 1))

# evaluate(mse, r2)
from sklearn.metrics import mean_squared_error, r2_score
MSE = mean_squared_error(Y_test, Y_pred)
print(f'MSE: {MSE}')

r2 = r2_score(Y_test, Y_pred)
print(f'R2: {r2}')

# coeeficient and intercept
print(f'Coefficient: {reg.coef_}')
print(f'Intercept: {reg.intercept_}')

# using the model
X_new = np.array([1.0, 1.2, 1.4]).reshape(-1, 1)
Y_new = reg.predict(X_new)
print(f'New Y: {Y_new}')
```

```
MSE: 0.038762659714382725
R2: 0.9292817886035307
Coefficient: [0.42133892]
Intercept: -0.3732225694780913
New Y: [0.04811635 0.13238413 0.21665191]
```

실습2: Multiple linear regression

1. 실습용으로 제공한 BostonHousing 데이터셋은 보스턴 지역의 지역정보 및 평균주택 가격 (medv) 정보를 담고 있다. 다른 변수를 이용하여 medv를 예측하는 모델을 만드세요 (feature 중 chas는 제외).
2. 회귀 모델의 mean_squared_error와 r2_score 값을 보이세요.
3. 회귀 모델의 coefficients와 Intercept 값을 보이세요.

In [4]:
```
BostonHousing = pd.read_csv("./data/BostonHousing.csv")
```

In [9]:
```
# 'medv' 컬럼이 타겟 컬럼, 나머지는 독립 변수
BostonHousing_X = BostonHousing.drop(['medv', 'chas'], axis=1)
BostonHousing_y= BostonHousing['medv']

# train test split
X_train, X_test, y_train, y_test = train_test_split(BostonHousing_X, BostonHousi

# linear regression
reg = LinearRegression()
reg.fit(X_train, y_train)

# predict
y_pred = reg.predict(X_test)

# evaluate(mse, r2)
MSE = mean_squared_error(y_test, y_pred)
print(f'MSE: {MSE}')

r2 = r2_score(y_test, y_pred)
print(f'R2: {r2}')

# coeeficient and intercept
print(f'Coefficient: {reg.coef_}')
print(f'Intercept: {reg.intercept_}')
```

```
MSE: 24.344456871929992
R2: 0.7359029307280057
Coefficient: [-9.79164542e-02  6.55190530e-02 -4.99609296e-03 -2.13172793e+01
  2.59729736e+00  7.68150802e-03 -1.89395459e+00  3.88424213e-01
 -1.55433784e-02 -1.11296742e+00  9.60981706e-03 -5.78193786e-01]
Intercept: 51.339602439179956
```

실습3: Rogistic regression

1. scikit-learn에 포함된 wine 데이터셋으로부터 wine의 등급을 분류하는 모델을 만드세요.
2. 만들어진 모델의 예측 정확도를 보이세요.

- train set에 대한 예측 정확도
- test set에 대한 예측 정확도

In [10]:
```
# prepare dataset
from sklearn import datasets
X, y = datasets.load_wine(return_X_y=True)

X = pd.DataFrame(X)

# train test split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_

# Rogistic regression
from sklearn.linear_model import LogisticRegression
logreg = LogisticRegression()
logreg.fit(X_train, y_train)

# predict
y_pred = logreg.predict(X_test)
print(y_pred)

# train set에 대한 예측 정확도 평가
from sklearn.metrics import accuracy_score
y_pred_train = logreg.predict(X_train)
accuracy_train = accuracy_score(y_train, y_pred_train)

# test set에 대한 예측 정확도 평가
accuracy = accuracy_score(y_test, y_pred)

print(f'Train set accuracy: {accuracy_train}')
print(f'Test set accuracy: {accuracy}')
```

```
[1 1 1 1 2 1 2 0 0 2 2 2 0 1 1 0 0 2 2 2 0 1 1 2 1 2 1 0 0 1 0 2 1 1 1 1 0
 2 0 0 2 0 1 2 1 1 0 2 1 2 0 2 2 1]
Train set accuracy: 0.9838709677419355
Test set accuracy: 0.9444444444444444
```

```
c:\Users\jdh25\AppData\Local\Programs\Python\Python310\lib\site-packages\sklearn
\linear_model\_logistic.py:469: ConvergenceWarning: lbfgs failed to converge (sta
tus=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
Please also refer to the documentation for alternative solver options:
    https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression
  n_iter_i = _check_optimize_result(
```