HW2 Report

인대영(2022311154)
2022-09-19

1. We first write a series of code in order to prompt the user for necessary

Prompt

In [2]:
```
train=input('Please enter the name of the training data file (e.g. boston_tr.csv): ')
```
```
Please enter the name of the training data file (e.g. boston_tr.csv): boston_tr.csv
```

In [3]:
```
test=input('Please enter the name of the test data file: (e.g. boston_tst.csv): ')
```
```
Please enter the name of the test data file: (e.g. boston_tst.csv): boston_tst.csv
```

Next, we use the input data to formulate the response variable and predictor variables. The fitted values for both train and test data are obtained, respectively, as shown below. We use the matrix approach such that $\hat{Y} = Zb$ where $b = (Z'Z)^{-1}Z'Y$. R^2 and MSE can each be calculated as $R^2 = 1 - \frac{SSE}{SST}$ and $MSE = \frac{SSE}{n-p}$, respectively, where p is the number of predictor variables.

In [9]:
```
#Fitted values for train data
Y_train_hat = Z_train.dot(B_train_hat)
Y_train_hat
```

Out[9]:

	0
0	30.756786
1	20.422682
2	27.174554
3	24.692692
4	14.717025
...	...
338	23.150192
339	31.805005
340	15.239591
341	33.552140
342	22.070635

In [12]:
```
#Fitted values for test data
Y_test_hat = Z_test.dot(B_train_hat)
Y_test_hat
```

Out[12]:

	0
0	24.345907
1	29.880133
2	19.446135
3	19.123273
4	21.119080
...	...
142	22.320327
143	15.878247
144	20.996992
145	27.194955
146	22.596097

Now, we calculate the prediction performance based on the fitted model and test data. The following formulas are utilized:

- $R^2(y, \hat{y}) = 1 - \frac{\sum (y-\hat{y})^2}{\sum (y-\bar{y})^2}$
- $MAE = \frac{1}{n}\sum_{i=1}^{n} |y-\hat{y}|$
- $MAPE = \frac{1}{n}\sum_{i=1}^{n} \frac{|y-\hat{y}|}{|y|}$
- $MSE = \frac{1}{n}\sum_{i=1}^{n} (y-\hat{y})^2$

Notice that the MSE for prediction uses n instead of $n-p$. The given formulas are calculated via the following code. The output file is also shown.

Prediction Performance

```
In [15]: #Calculate Predictive R-square
         SSE_test = sum((Y_test-Y_test_hat[0])**2)
         SST_test = sum((Y_test-Y_test.mean())**2)
         R_square_pred = round(1-SSE_test/SST_test,4)
         R_square_pred

Out[15]: 0.7637

In [16]: #Calculate MAE
         n_test = len(df_test)
         MAE = round(sum(np.abs(Y_test-Y_test_hat[0]))/n_test,3)
         MAE

Out[16]: 2.911

In [17]: #Calculate MAPE
         MAPE = round(1/n_test*sum(abs(Y_test-Y_test_hat[0])/abs(Y_test)),3)
         MAPE

Out[17]: 0.158

In [18]: #Calculate RMSE
         RMSE = round((SSE_test/n_test)**0.5,3)
         RMSE

Out[18]: 3.971
```

```
Coefficients
-------------
Constant: 23.685
Beta1: -0.074
Beta2: 0.03
Beta3: -0.075
Beta4: 1.109
Beta5: -5.275
Beta6: 4.001
Beta7: -0.036
Beta8: -1.08
Beta9: -0.005
Beta10: -0.677
Beta11: 0.007
Beta12: -0.365

Model Summary
-------------
R-square = 0.7648
MSE = 14.562

Prediction Performance
---------------------
Predictive R-square = 0.7637
MAE = 1.248
MAPE = 0.068
RMSE = 6.759
```

2. We used the 'statsmodels' package to run linear regression as shown below.

```
In [23]: #Linear Regression
         model = sm.OLS(y, x).fit()
         predictions = model.predict(x)
         print(model.summary())

                            OLS Regression Results
==============================================================================
Dep. Variable:                   medv   R-squared:                       0.765
Model:                            OLS   Adj. R-squared:                  0.756
Method:                 Least Squares   F-statistic:                     89.42
Date:                Sun, 18 Sep 2022   Prob (F-statistic):           5.74e-96
Time:                        22:16:39   Log-Likelihood:                -939.39
No. Observations:                 343   AIC:                             1905.
Df Residuals:                     330   BIC:                             1955.
Df Model:                          12
Covariance Type:            nonrobust
==============================================================================
                 coef    std err          t      P>|t|      [0.025      0.975]
------------------------------------------------------------------------------
const         23.6846      4.697      5.043      0.000      14.445      32.924
crim          -0.0737      0.032     -2.337      0.020      -0.136      -0.012
zn             0.0300      0.014      2.173      0.031       0.003       0.057
indus         -0.0753      0.055     -1.360      0.175      -0.184       0.034
chas           1.1091      0.898      1.236      0.217      -0.657       2.875
```

We also calculated predicted R^2, MAE, MAPE, RMSE using 'sklearn' package. We can observe that the values are the same as the values obtained from the code in #1.

```
In [25]: #Prediction Performance using Python Package
         print('Predictive R-square: ', round(r2(Y_test, Y_test_hat[0]),3))
         print('MAE: ', round(mae(Y_test, Y_test_hat[0]),3))
         print('MAPE: ', round(mape(Y_test, Y_test_hat[0]),3))
         print('RMSE: ', round((mse(Y_test, Y_test_hat[0]))**0.5,3))

         Predictive R-square:  0.764
         MAE:  2.911
         MAPE:  0.158
         RMSE:  3.971
```