Assignment 2

2021321148
INSIK CHO
조인식

1.

a,b.c.

```
[4]  train_data_name=input("Enter the name of train data file [(ex) harris.dat] : ") # data name
     test_data_name=input("Enter the name of test data file [(ex) harris.dat] : ")
     coding_fm=int(input("Select the data coding format(1 = 'a b c' or 2 = 'a,b,c'): ")) # data separator
     separator_fm={coding_fm ==1 : " "}.get(True, ",")
     res_pos=int(input("Enter the column position of the response variable : [from 1 to p] : "))
     header=input("Does the data have column header? (y/n) : ")
     if(header=="y") : trdata=pd.read_csv(train_data_name, sep=separator_fm) # loading data
     else : trdata=pd.read_csv(train_data_name, sep=separator_fm, header=None) # loading data
     if(header=="y") : tstdata=pd.read_csv(test_data_name, sep=separator_fm) # loading data
     else : tstdata=pd.read_csv(test_data_name, sep=separator_fm, header=None) # loading data
     out_name=input("Enter the output file name to export [(ex) result.txt] : ")
```

```
Enter the name of train data file [(ex) harris.dat] : boston_tr.csv
Enter the name of test data file [(ex) harris.dat] : boston_tst.csv
Select the data coding format(1 = 'a b c' or 2 = 'a,b,c'): 2
Enter the column position of the response variable : [from 1 to p] : 13
Does the data have column header? (y/n) : y
Enter the output file name to export [(ex) result.txt] : result.txt
```

```
# Extracting X and y
data = pd.DataFrame(trdata)
C = pd.DataFrame(np.ones(shape = (data.shape[0],)))
data = pd.concat([C,data], axis = 1, ignore_index= True)
X= data.drop([data.columns[res_pos]], axis=1 )
Y= data.iloc[:, res_pos ].values

#Extracting test X and y
tstdata = pd.DataFrame(tstdata)
tstC = pd.DataFrame(np.ones(shape = (tstdata.shape[0],)))
tstdata = pd.concat([tstC,tstdata], axis = 1, ignore_index= True)
tstX= tstdata.drop([tstdata.columns[res_pos]], axis=1 )
tstY= tstdata.iloc[:, res_pos ].values
```

HW1의 코드를 조금 수정해서 사용했습니다.
기존에 data_name으로 되있는 것을 train_data_name과 test_data_name으로 바꾸고 Input으로 입력가능하도록 설정했습니다.
주어진 boston 데이터로 연습을 진행했고, 'medv'열이 마지막 열(13번째)에 있어서 column position으로 13을 설정했습니다.
그 외에 test data회귀분석을 위해 test 데이터에서 X,Y 행렬 추출하는 코드를 따로 설정했습니다.

```
#Implementing Multiple Linear regression

beta = np.linalg.inv(X.T@ X)@(X.T @ Y)
haty = X @ beta
residual = (Y- haty )
RSS = ((residual**2).mean())*(data.shape[0])
TSS = (((Y - (Y.mean()))**2).mean())*(data.shape[0])
Rsquared = 1- (RSS/TSS )
MSE = RSS / (data.shape[0] - data.shape[1]+1)

# test data
tsthaty = tstX @ beta
tstresidual = (tstY-tsthaty)
tstRSS = ((tstresidual**2).mean())*(tstdata.shape[0])
tstTSS = (((tstY - (tstY.mean()))**2).mean())*(tstdata.shape[0])
tstRsquared = 1- (RSS/TSS )
tstMSE = tstRSS / (tstdata.shape[0])
tstRMSE = np.sqrt(tstMSE)
tstMAE = np.abs(tstY -tsthaty).mean()
tstMAPE = (np.abs(tstY -tsthaty)/np.abs(tstY)).mean()
```

d. train 데이터에서 얻은 추정치로 test데이터의 X를 추정했습니다. (tstX @ beta)
이를 이용해 주어진 지표들을 계산했습니다.
특이사항은 train data의 mse는 n-p로 나눴으나, test data는 n으로 나눴습니다.

```
import sys
sys.stdout = open(out_name,'w')
print("coefficients")
print("-----------------")

for i in range(beta.size):
  if i == 0 :
    print('constant:', beta[0])
  else:
    print('beta%d :'%i, beta[i])

print("              ")
print("model summmary")
print("---------------")
print("R-squared =  ", Rsquared)
print("MSE = ", MSE)

print("              ")
print("Prediction Performance")
print("---------------")
print("Predictive R-squared =  ", tstRsquared)
print("MAE = ", tstMAE)
print("MAPE =  ", tstMAPE)
print("RMSE = ", tstRMSE)
```

위 코드를 사용해 다음과 같은 result.txt파일의 내용을 얻었습니다.

coefficients
---
constant: 23.684594637179014
beta1 : -0.07366072897988217
beta2 : 0.030035081773484705
beta3 : -0.07529551828526891
beta4 : 1.1090747161736099
beta5 : -5.274829874185233
beta6 : 4.000848011134115
beta7 : -0.03626454160925006
beta8 : -1.079554806747001
beta9 : -0.004959667486739615
beta10 : -0.6774621226803804
beta11 : 0.006923668741425964
beta12 : -0.36502856092933667

model summmary

R-squared = 0.7647913348497996
MSE = 14.559974273870113

Prediction Performance

Predictive R-squared = 0.7647913348497996
MAE = 2.904847388891414
MAPE = 0.15773228176518989
RMSE = 3.9720352849818013

2. 비교를 위해 사용한 sklearn 코드는 다음과 같습니다.

```
from sklearn.linear_model import LinearRegression
from sklearn.metrics import r2_score as r2
from sklearn.metrics import mean_squared_error  as MSE
from sklearn.metrics import mean_absolute_error as MAE
def MAPE(tstY,predy):
  return (np.abs(tstY -predy)/np.abs(tstY)).mean()
def RMSE(tstY,predy):
  M = MSE(tstY,predy)
  return np.sqrt(M)

fitter = LinearRegression()
fitter.fit(X, Y)

predy = fitter.predict(tstX)

print("               ")
print("Prediction Performance by sklearn")
print("---------------")

print("Predictive R-squared =  ", r2(tstY,predy))
print("MAE = ", MAE(tstY,predy))
print("MAPE =  ", MAPE(tstY,predy))
print("RMSE = ", RMSE(tstY,predy) )
```

3. 1과 2에서 얻은 두 지표를 비교 시,

```
Prediction Performance
---------------
Predictive R-squared =   0.7647913348497996
MAE =  2.904847388891414
MAPE =   0.15773228176518989
RMSE =  3.9720352849818013

Prediction Performance by sklearn
---------------
Predictive R-squared =   0.7635849634495481
MAE =  2.9048473888913016
MAPE =   0.1577322817651805
RMSE =  3.972035284981572
```

(by sklearn이 2.에서 얻은 지표입니다.)

소숫점 2자리 까지는 모두 동일한 것을 확인할 수 있습니다. 아마 그 아래는 rounding error 때문에 차이가 벌어진 것으로 보입니다.