Tae Coding
Introduction to Data Science: CS61
Summer 2018
Class Exercise#6

Date Given: June 28, 2018 Due Date:

==

Old Faithful is a geyser located in Yellowstone National Park in Wyoming, US. It is a highly predictable geothermal feature and has erupted every 44 to 125 minutes since 2000.

The 'oldfaithful.csv' file contains waiting time between eruptions and the duration of the eruption. This file contains 272 observations on 2 variables.

Variable Name	Type	Semantics
Time Eruption	Numeric	Eruption duration time in mins
Time Waiting	Numeric	Waiting time between eruption in mins

Build your regression model.

- Use 'Time Eruption' as the predictor variable
- Use 'Time Waiting' as the response variable

Use Python/Scikit-Learn package for this homework assignment.

1. Compute the regression equation and the R-Square metrics of your regression model.

2. Split the dataset into training and testing set with 70/30 ratio randomly. Build a regression model using training data set. Compute the predicted value of 'Time Waiting' variable of training and testing data set using the model built. The Root Mean Square Error (RMSE) and the Mean Square Error (MSE) metrics are defined as follows.

$$RMSE = \sqrt{\frac{1}{N}\sum_{i=1}^{N}[f(x_i) - y_i)]^2} \qquad MSE = \frac{1}{N}\sum_{i=1}^{N}[f(x_i) - y_i)]^2$$

Here $f(x_i)$ is the computed value and y_i is the true (observed) value.

Compute the training error (RMSE or MSE) and the testing error (RMSE or MSE).
Which one is greater – RMSE(Training) or RMSE (Testing)?

Answer

All Data: Regression Equation:

$y\ (Time\ Waiting) = 10.7296 * x(Eruption\ Time) + 33.4744$

$R^2 = 0.8115$

After splitting data into training (70%) and testing (30%)

Model using training data

$y\ (Time\ Waiting) = 10.637 * x(Eruption\ Time) + 33.535$

$MSE\ (Training) = 34.68$
$MSE\ (Testing) = 35.04$

Python Code

```
# -*- coding: utf-8 -*-
"""
Created on Thu May 31 15:03:09 2018

@author: ash
"""
####################################################
# June 10, 2018
# Data: Old Faithful
#
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from sklearn import linear_model
from sklearn.cross_validation import train_test_split
from sklearn.metrics import mean_squared_error
from sklearn.metrics import r2_score

######################################
# 1. Read Data File + Show dimensions + Plot Data
#
data = np.genfromtxt('oldfaithful.csv',delimiter=',',skip_header=1)
#data = data[0:20]
print(data)
data.shape

eruptionX = data[:,0]
waitingY = data[:,1]

eruptionX.shape
waitingY.shape

plt.figure()
plt.plot(eruptionX, waitingY,'.b',label='Observation')
plt.xlabel('Eruption Duration (minutes)')
plt.ylabel('Time Between Eruptions (minutes)')


#####################################
# 2. Regression with all data
# Compute Regression Equation + + Plot Regression Line
#
reg = linear_model.LinearRegression()

df_x = pd.DataFrame(eruptionX)
df_y = pd.DataFrame(waitingY)

df_x.describe()
df_y.describe()

reg.fit(df_x,df_y)
reg.coef_
reg.intercept_
```

```
# Plot Regression Line
#
plot_x = np.array([1.5,5.5])
plot_y = reg.coef_[0]*plot_x + reg.intercept_

plt.figure()
plt.plot(eruptionX, waitingY,'.b',label='Observation')
plt.xlabel('Eruption Duration (minutes)')
plt.ylabel('Time Between Eruptions (minutes)')
plt.plot(plot_x, plot_y, 'r:', label='Linear Regression')

####################################################
# 2.1. Compute Predicted Values + RSquare
#
predicted_value = reg.predict(df_x)

r2_score(waitingY,predicted_value)
r2_score(df_y,predicted_value)

###################################################
# 3. Split data into Training / Testing
#
x_train, x_test, y_train, y_test = train_test_split(df_x, df_y,
test_size=0.3, random_state=0)

x_train
y_train

x_test
y_test

x_train.shape
y_train.shape
x_test.shape
y_test.shape

plt.plot(x_train, y_train,'.b')
plt.plot(x_test, y_test,'.b')
#################################################
# 4. Build Regression model using training data
#
reg_train = linear_model.LinearRegression()

reg_train.fit(x_train,y_train)
reg_train.coef_
reg_train.intercept_

################################################
# 5.1 Predict Using Training data
# Compute Training MSE
#
predcited_values_training = reg_train.predict(x_train)
mean_squared_error(y_train, predcited_values_training)
```

```
# 5.2 Predict Using Testing data
# Compute Testing MSE
#
predcited_values_testing = reg_train.predict(x_test)
mean_squared_error(y_test, predcited_values_testing)
```

R Code

```
> #############################################
> #  Homework: Old Faithful
> #
> rm(list=ls(all=TRUE))
> #################################################
> # 1. Read Data file + Show Dimensions + Plot
> #
> DataFile = read.csv("oldFaithful.csv")
>
> #fix(DataFile)
> (dim(DataFile))
[1] 272   2
> (nDataFile = dim(DataFile)[1])
[1] 272
> (nrow(DataFile))
[1] 272
> #(DataFile = DataFile[1:20,])
>
> plot(DataFile$TimeEruption, DataFile$TimeWaiting,
+      pch=21, col="blue", bg="red")
> ###################################################
> # 2. Regression with all data
> # Compute Regression Equation + Plot the Regression line
> #
> # 2.1 Compute RSquare
> #
> lm.fit.all = lm(TimeWaiting~TimeEruption, data=DataFile)
> summary(lm.fit.all)

Call:
lm(formula = TimeWaiting ~ TimeEruption, data = DataFile)

Residuals:
     Min       1Q   Median       3Q      Max 
-12.0796  -4.4831   0.2122   3.9246  15.9719 

Coefficients:
             Estimate Std. Error t value Pr(>|t|)    
(Intercept)   33.4744     1.1549   28.98   <2e-16 ***
TimeEruption  10.7296     0.3148   34.09   <2e-16 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 5.914 on 270 degrees of freedom
Multiple R-squared:  0.8115,	Adjusted R-squared:  0.8108 
F-statistic:  1162 on 1 and 270 DF,  p-value: < 2.2e-16

> plot(DataFile$TimeEruption, DataFile$TimeWaiting,
+      pch=21, col="blue", bg="red")
> abline(lm.fit.all, lwd=3, col="red")
>
>
> #####################################################
> # 3. Split the data into Training/Testing 70/30
> #
> set.seed(0)
> train = sample(nrow(DataFile),floor(nrow(DataFile) * 0.70))
> length(train)
[1] 190
>
> all = seq(1,nrow(DataFile),1)
> test = setdiff(all,train)
> length(test)
[1] 82
```

```
>
> ######################################################
> plot(DataFile$TimeEruption[train], DataFile$TimeWaiting[train],
+      pch=21, col="blue", bg="red")
>
> points(DataFile$TimeEruption[test],DataFile$TimeWaiting[test],
+        pch=21, col="blue", bg="green")
> ######################################################
> # 4. Build Regression model using Training data
> #
> lm.fit.train = lm(TimeWaiting~TimeEruption, data=DataFile, subset=train)
> summary(lm.fit.train)

Call:
lm(formula = TimeWaiting ~ TimeEruption, data = DataFile, subset = train)

Residuals:
     Min       1Q   Median       3Q      Max
-12.1465  -4.8556   0.1688   3.6368  16.1686

Coefficients:
             Estimate Std. Error t value Pr(>|t|)
(Intercept)    33.987      1.397   24.32   <2e-16 ***
TimeEruption   10.527      0.381   27.63   <2e-16 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 6.02 on 188 degrees of freedom
Multiple R-squared:  0.8024,   Adjusted R-squared:  0.8014
F-statistic: 763.5 on 1 and 188 DF,  p-value: < 2.2e-16

>
> ######################################################
> plot(DataFile$TimeEruption[train], DataFile$TimeWaiting[train],
+      pch=21, col="blue", bg="red",
+      xlim=c(1,5),ylim=c(40,90))
> abline(lm.fit.train, lwd=3, col="red")
>
> plot(DataFile$TimeEruption[test],DataFile$TimeWaiting[test],
+      pch=21, col="blue", bg="black",
+      xlim=c(1,5),ylim=c(40,90))
> abline(lm.fit.train, lwd=3, col="red")
>
> ##########################################################
> predcitedValues = predict(lm.fit.train, DataFile)
>
> predcitedValuesTrain = predcitedValues[train]
> length(predcitedValuesTrain)
[1] 190
>
> predcitedValuesTest = predcitedValues[-train]
> length(predcitedValuesTest)
[1] 82
>
> #######################################################
> # 5. Compute Mean Square - Training
> SquareDiff1 = (DataFile$TimeWaiting[train] - predcitedValuesTrain) ^2
> SETrain = sum(SquareDiff1)
> (MSETrain = SETrain/length(train))
[1] 35.85298
>
> #########################################################
> # 5.2 Compute Mean Square - Testing
> #
> SquareDiff1 = (DataFile$TimeWaiting[-train] - predcitedValuesTest) ^2
> SETest = sum(SquareDiff1)
```

```
> (MSETest = SETest/length(test))
[1] 32.39149

>
```