Course 3, Task 1: Report to Blackwell

Linear Regression Modeling in R

Anders Dowd - Spring 2022 Data Analytics Certification Program

(A) Installing and Learning R and R Studio

Downloading and installing R and R Studio were reasonably straight forward. The links provided in the tutorial all functioned as expected and the downloads were relatively quick. Installation only took a few minutes and the RStudio was up an running with little effort. Over the past couple of months I've settled into Python pretty comfortably, so it was a bit of a gear shift working in R. I did find a number of similarities that helped ease the transition in the language, but the interface was significantly different enough from Jupyter Notebook that it did require an adjustment and plenty of background reading.

Much of the functionality of R seemed familiar enough having previously worked in Python (reading data sets, preprocessing, EDA, train/test/split, etc.), but the processes, formatting and syntax took some adjusting to. For instance, the concept of installing and importing packages was not new, but RStudio allowed for both the installation and import within the same interface, whereas Jupyter Notebook requires the installation to take place within the command module before allowing one to import a library into a notebook. Also, the format of dataframe functions is flipped in R. The function is listed first, followed by the dataframe in parentheses. I'm used to the Python way of listing the name of the dataframe followed by the command (e.g. summary(df) vs. df.describe()). Some new characters made appearances as well. '$' and '<-' had new functionality previously associated with other characters altogether in Python. These are all minor differences in the grand scheme of things, but certainly required a restructuring of my thought process in writing the code.

(B) R Tutorial Using the Cars Data Set

The tutorial was very helpful in guiding me through the process and made the transition considerably smoother. It featured an exercise utilizing a dataset consisting of three columns and 50 rows. The data included car brand names, speeds, and distances traveled. The goal was to predict the distance traveled by a particular vehicle based solely on the speed at which the vehicle moved. This was a straight-forward regression problem solved with the use of a linear regression algorithm. The data was split into a training set of 35 entries and a testing set of 15 entries. The model proved quite efficient and accurate in its predictions with a Multiple R-squared metric of 0.9572 and a p-value of <2.2e-16, though it did return an unexpected negative value for one of the rows. Table 1 displays the predictions of the model vs the ground truth; Figure 1 graphs the results in a scatterplot.

Index	Distance	Predictions
2	4	-9.284179
7	17	17.950448
10	20	22.489552
13	26	27.028657
15	26	27.028657
20	32	36.106866
23	34	36.106866
29	42	49.724179
30	46	49.724179
31	46	49.724179
39	60	63.341493
40	61	63.341493
42	68	63.341493
44	76	72.419701
50	120	86.037015

Table 1:
True Distance vs. Linear Regression Predictions

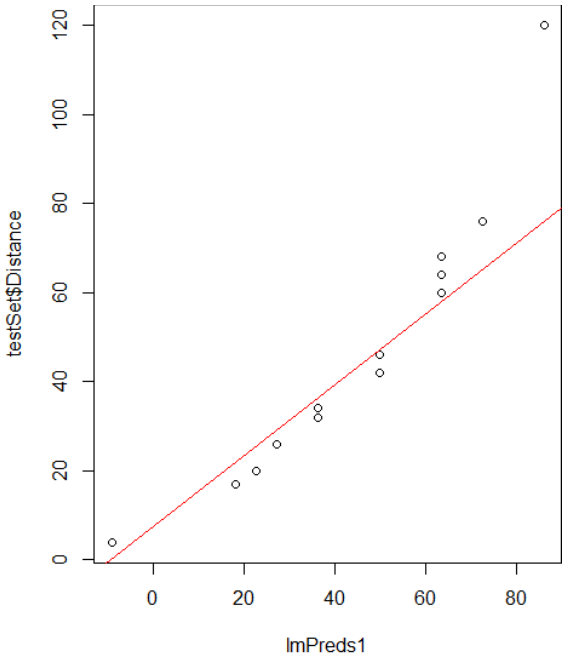

Figure 2:
Linear Regression Predictions vs. True Distance

(*C) Finding Errors in an R Script Using the Iris Data Set*

The provided script for building another regression model, this time using the well-known Iris data set, proved to be full of errors. Even having run through the tutorial but once prior to tackling this next task, most of the errors were glaringly obvious just reading through the code. For the sake of thoroughness, I first inputted the code as presented just to see what error messages it would output. The following is a list of the code as originally written along with the errors returned and the corrected code:

```
install.packages(readr)
Error in install.packages : object 'readr' not found
```

Missing quotes. Change to: `install.packages("readr")`

```
library("readr")
```

No error or warning. Per the formatting in the tutorial, I removed the quotes and it worked both ways.

```
IrisDataset <- read.csv(iris.csv)
Error in read.table(file = file, header = header, sep = sep, quote = quote,  :
  object 'iris.csv' not found
Error in file(file, "rt") : cannot open the connection
In addition: Warning message:
In file(file, "rt") :
  cannot open file 'iris.csv': No such file or directory
```

I had to add the file to my working directory first, then I put the file name in quotes.

`read.csv("iris.csv")`

```
attributes(IrisDataset)
```

No error or warning.

```
summary(risDataset)
Error in summary(risDataset) : object 'risDataset' not found
```

Missing the first letter of the dataset name, "I".

```
str(IrisDatasets)

Error in str(IrisDatasets) : object 'IrisDatasets' not found
```

An extra “s” is tacked on to the end of the dataset name.

```
names(IrisDataset)
```

No error or warning.

```
hist(IrisDataset$Species)

Error in hist.default(IrisDataset$Species) : 'x' must be numeric
```

The class of the Species column is “character”. I had to first change the names to numbers using this code:

```
IrisDataset['Species'][ IrisDataset['Species'] == 'setosa'] <- 1
IrisDataset['Species'][ IrisDataset['Species'] == 'versicolor'] <- 2
IrisDataset['Species'][ IrisDataset['Species'] == 'virginica'] <- 3
```

Then I had to reclassify it as.numeric.

```
plot(IrisDataset$Sepal.Length
```

This returned no error, but it also returned no plot. It is missing its closing parentheses. Once the line is completed, it plots the Sepal.Length variable against the index, which is mostly useless for our purposes. A more informative graph plots the Petal.Length against the Petal.Width, which is achieved with the following code:

```
plot(IrisDataset$Petal.Length, IrisDataset$Petal.Width)
```

This format can also be used to plot the Sepal attributes, but that’s not what we were looking to analyze in this exercise.

```
qqnorm(IrisDataset)

Error in xy.coords(x, y, xlabel, ylabel, log) :

  'x' and 'y' lengths differ

In addition: Warning message:

  In xtfrm.data.frame(x) : cannot xtfrm data frames
```

This needs an attribute from the dataframe to plot. Here it is with Sepal.Width:

```
qqnorm(IrisDataset$Sepal.Width)
```

```
IrisDataset$Species<- as.numeric(IrisDataset$Species)
```

This won't work before changing the names of the classes in the variable ('setosa', 'versicolor', 'virginica') to numbers (1, 2, 3). I did this already when I was fixing the code to make the histogram work.

```
set.seed(123)
```

No error or warning. (*Note: The figures and plots included in this report use seed(2395).)

```
trainSize <- round(nrow(IrisDataset) * 0.2)
```

No error code, but using a training set of only 20% of the data doesn't make sense here. I changed it to 0.7, or 70%.

```
testSize <- nrow(IrisDataset) - trainset

Error: object 'trainSet' not found
```

This should have read: `testSize <- nrow(IrisDataset) -trainSize`

```
trainSizes

Error: object 'trainSizes' not found
```

This was just another typo. Dropped the 's'.

```
testSize
```

No error or warning.

```
trainSet <- IrisDataset[training_indices, ]

testSet <- IrisDataset[-training_indices, ]

Error in `[.data.frame`(IrisDataset, training_indices, ) :

  object 'training_indices' not found
```

The training_indices were not defined before attempting to call them. This needs to be included before defining the train and test sets:

`training_indices<-sample(seq_len(nrow(IrisDataset)),size =trainSize)`

```
set.seed(405)

trainSet <- IrisDataset[training_indices, ]

testSet <- IrisDataset[-training_indices, ]
```

No error or warning, but this is superfluous. We already set a seed and the train/test sets.

```
LinearModel<- lm(trainSet$Petal.Width ~ testingSet$Petal.Length)

Error in eval(predvars, data, env) : object 'testingSet' not found

Error in model.frame.default(formula = trainSet$Petal.Width ~ testSet$Petal.Length,  :

  variable lengths differ (found for 'testSet$Petal.Length')
```

Everything is wrong with this. As written, this code attempts to predict the Petal.Width of the trainset using the Petal.Length of the testSet (which is mistakenly written as "testingSet"). We need to be using the Petal.Width of the trainSet to predict the Petal.Length of the same trainSet in order to properly train our model. This corrected code does that: `LinearModel<-lm(Petal.Length~ Petal.Width, trainSet)`

```
summary(LinearModel)
```

No error or warning.

```
prediction<-predict(LinearModeltestSet)

Error in predict(LinearModeltestSet) :

  object 'LinearModeltestSet' not found
```

The original line of code is missing a comma between the LinearModel name and the testSet.

```
predictions

Error: object 'predictions' not found
```

Another typo. Drop the 's'.

Finally, I plotted the predictions vs. the ground truth:

```
plot(prediction, testSet$Petal.Length)

abline(lm(prediction ~ testSet$Petal.Length), col = "red") # Draw regression line
```

Once all the errors in the code were corrected, the model proved to be an effective one. Nearly as accurate as the cars model, the linear regression model for the iris dataset returned a Multiple R-squared metric of 0.9259 and a p-value of <2.2e-16. The predictions and ground truth values are displayed in Table 2 and Figure 2 on the following page.

Index	Petal.Legth	Predictions
1	1.4	1.50372
4	1.5	1.50372
6	1.7	1.955102
10	1.5	1.278029
14	1.1	1.278029
15	1.2	1.50372
16	1.5	1.955102
25	1.9	1.50372
26	1.6	1.50372
29	1.4	1.50372
31	1.6	1.50372
34	1.4	1.50372
35	1.5	1.50372
37	1.3	1.50372
42	1.3	1.729411
48	1.4	1.50372
57	4.7	4.663392
60	3.9	4.212011
66	4.4	4.212011
67	4.5	4.437701
68	4.1	3.309247
81	3.8	3.534938
88	4.4	3.98632
93	4.0	3.760629
94	3.3	3.309247
95	4.2	3.98632
97	4.2	3.98632
98	4.3	3.98632
100	4.1	3.98632
102	5.1	5.340465
105	5.8	6.017538
106	6.6	5.791847
109	5.8	5.114774
112	5.3	5.340465
115	5.1	6.468919
120	5.0	4.437701
124	4.9	5.114774
125	5.7	5.791847
126	6.0	5.114774
130	5.8	4.663392
137	5.6	6.468919
141	5.6	6.468919
143	5.1	5.340465
144	5.9	6.243228
149	5.4	6.243228

Table 2:
True Petal Length vs. Linear Regression Predictions

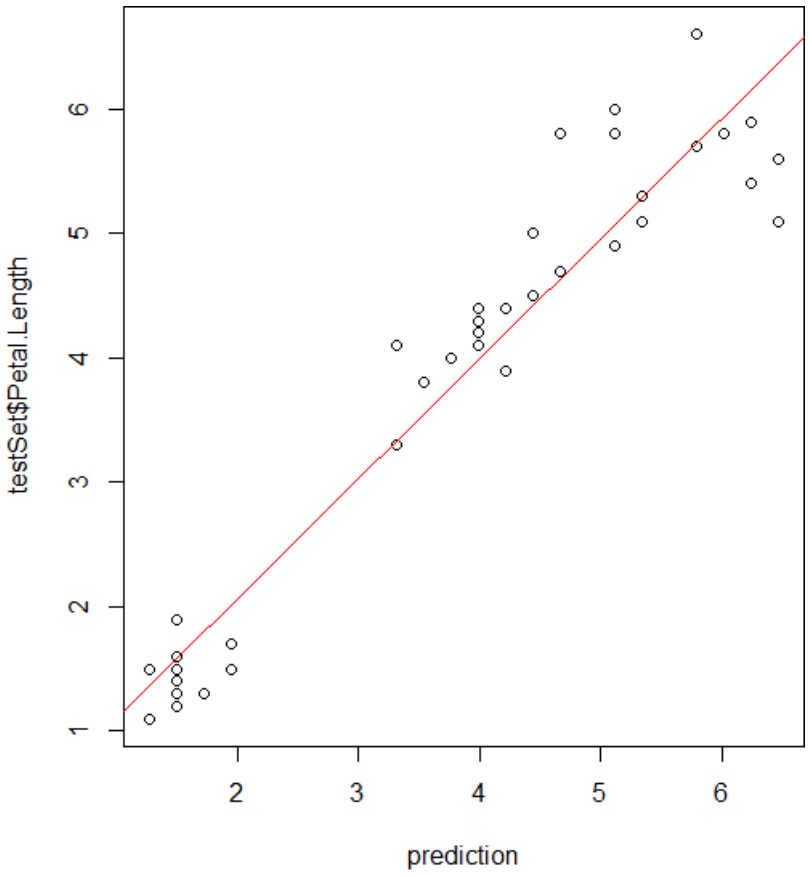

Figure 2:
Linear Regression Predictions vs. True Petal Length

(D) Conclusions and Recommendations

For those dabbling in R for the first time who only have prior experience in Python, they will find many similarities but should be prepared to restructure their thinking regarding syntax.

While R has the tools to deliver great statistical models, the graphic output leaves a little to be desired. If one is looking for appealing plots and graphs to spice up reports or presentations, Python's MatPlotLib and Seaborn have much more attractive output capabilities.

One advantage RStudio has over Jupyter Notebook is display of datasets. RStudio automatically provides a way to easily view the entire dataframe whenever needed after importing and it updates in real time when making changes in the code. Jupyter on the other hand will only display a limited view of a large dataset unless certain parameters are set, and it must be called back each time it is edited in order to see any changes.