Lab 5

Start Assignment

Due Monday by 11:59pm **Points** 30 **Submitting** a text entry box or a file upload

EE5373: Data Modeling Using R

Summer, 2023

Department of Electrical and Computer Engineering

University of Minnesota

Lab 5: Training, testing, and predicting.

-

Due date: See the due date shown on the class web page.

Goal: This lab explores the training and testing process for multiple linear regression models and predicting responses across data sets.

What to do:

Part 1 – Training and testing using a single data set.

In the discussion of data splitting, the value f was defined to be the fraction of the complete data set used for the training set. You then test the predictive capability of your model trained on this portion of the data set using the remaining (1-f) of the data set. Comparing the predicted values with the actual values in the test set, you obtain the vector delta_1 = (actual values) - (predicted values). This vector shows how well the model predicted the actual results for one partitioning of the data set using the sample() function. The total number of elements in delta_1 is N = (1-f)n, where n is the total number of values in the original data set.

If you run the experiment again with the same value of f, the sample() function will allocate a different subset of the data to the testing and training sets (do NOT set the random number seed for this lab – see p. 50 in the textbook for more information about the random number seed). Call the resulting second vector of difference values delta_2.

If you repeat this process k times, you will have k different delta_i vectors, each with N elements. Now concatenate all these delta_i vectors into a single vector called D_f. This vector will have Nk elements. The mean and confidence interval of D_f shows how well the model predicted the actual results for k different training-testing permutations provided by the sample() function.

For the benchmarks int95, int06, fp95, and fp06, use the process described above to plot the mean and 95 percent confidence interval (e.g. using error bars around the mean) for D_f for f = {0.1, 0.2, 0.3, ..., 0.9}. Note that you can use the models you developed in Lab 4. When you are done, you will have a graph for each benchmark showing how the mean and confidence intervals of D_f vary as f changes. Use k=100 to ensure that you have a reasonable number of data points for each confidence interval. (See the code below for an example of one possible way to plot data with error bars.)

Discuss what your results show. For example, what is the value of f that tends to give the best results for each benchmark? Is it the same value of f for every benchmark? Why or why not? What happens when f is at the extreme values? Why? What other interesting observations can you make?

(only if you have time do part 2)

Part 2 – Predicting responses across data sets.

Use the regression model that you previously developed for int95 to predict the performance values for fp95, and your previous model for int06 to predict the performance values for fp06. Compute the mean and the 95 percent confidence interval for each prediction. What can you say about the models' predictive abilities, based on these results?

What to turn in for grading:

Write a short lab report that includes your results and answers the questions above.

Some useful information:

Here is an example of R code showing how to extract the endpoints of a confidence interval computed with the t-test.

```
# Create a new vector with some test data.
x = 1:1000
# Find the mean and confidence interval for this vector of example data.
ci_of_x = t.test(x, conf.level = 0.90)
# Extract the mean of this vector from the confidence interval calculation
ci_of_x$estimate
# Show the two confidence intervals
ci_of_x$conf.int
# Extract the lower bound of the confidence interval
ci_of_x$conf.int[1]
# Extract the upper bound of the confidence interval
ci_of_x$conf.int[2]
```

-

Here is some example code showing one method for plotting error bars.

```
# Make up some data for the plot.
# (You will, of course, want to use your own data.)
x <- 1:5
y <- c(23, 14, 15, 45, 8)
ylow <- y-2*x
yhigh <- y+2.3*x

# Plot the data with error bars
plot(x, y, ylim = range(c(ylow, yhigh)))
arrows(x, ylow, x, yhigh, length=0.05, angle=90, code=3)
```