CS 422: Data Mining

Department of Computer Science
Illinois Institute of Technology
Vijay K. Gurbani, Ph.D.

Fall 2022: Homework 2 (10 points)

Due date: Sat, September 24, 2022, 11:59:59 PM Chicago Time

Please read all of the parts of the homework carefully before attempting any question. If you detect any ambiguities in the instructions, please let me know right away instead of waiting until after the homework has been graded.

1. Exercises (2 points)

1.1 [1 point] ISLR 2e (Gareth James, et al.)

Section 3.7 (Exercises), page 123: Exercise 6. (Hint: The least squares line is given by the equation below.)

$$y_i = \hat{\beta}_0 + \hat{\beta}_1 x_i$$

1.2 [1 point] Section 3.7 (Exercises), page 120: Exercises 1, 3, 4-a.

2. Programming Problem (8 points divided evenly by each subsection)

Please label your answers clearly. Each answer must be preceded by the R markdown as shown in the Homework 0 R notebook (### Part 2.1-A-ii, for example). Failure to clearly label the answers in the submitted R notebook will lead to a loss of 2 points.

Round up all decimal numbers to two significant digits.

2.1 Install package ISLR. This package contains a dataset called Auto.

In this problem, you will tackle multi-variate linear regression using the Auto dataset from Problem 2.2. You will set aside a portion of this dataset (5%) for testing, and use the remaining portion (95%) to train your OLS model. To divide the dataset into training and testing sets, issue the following commands **exactly as shown:**

```
> set.seed(1122)
> index <- sample(1:nrow(Auto), 0.95*dim(Auto)[1])
> train.df <- Auto[index,]
> test.df  <- Auto[-index, ]
```

The set.seed() command seeds the pseudo-random number generator for reproducibility. The sample() command picks 352 random numbers between 1 and the total number of rows in the Auto dataset, without replacement. These numbers correspond to the indices of the Auto dataset, and all observations at these indices constitute the training dataset (the third line of code above). The remaining numbers (note the use of -index in the fourth line

of code above) correspond to indices not chosen through random sampling, and thus, they become the test set of the dataset. Once you have divided your dataset as shown above, you will train on the train.df dataframe and test your trained model on the test.df dataframe.

(a) From the training dataset, create a model using all the predictors except name to predict mpg.

(i) Why is using name as a predictor not a reasonable thing to do?

(ii) Print the summary of the regression model, and comment on how well the model fits the data by studying the R^2 , RSE and RMSE. To do so, first print out the values of R^2, RSE, and RMSE as follows:

R-sq value is XXX.XX
Adjusted R-sq value is XXX.XX
RSE is XXX.XX
RMSE is XXX.XX

Hint: You can extract the values of R^2 and adjusted R^2 from the object returned to you by the summary() method, or you can code it in R. You will need to write R code to get the values of RSE and RMSE. Recall that:

$$RSE = \sqrt{\frac{1}{n-p-1}RSS} \qquad \text{RMSE} = \sqrt{\frac{\sum(y_i - \hat{y}_i)^2}{N}} \qquad \text{RSS: } \sum_{i=1}^{n}(y_i - \hat{y}_i)^2$$

(iii) Plot the residuals of the model.

(iv) Plot a histogram of the residuals of the model. Does the histogram follow a Gaussian distribution? What can you say about the distribution of the residuals?

(b) Using the regression model you have created in (a), your aim is to narrow down the features to the 3 attributes will act as the best predictors for regressing on **mpg**. To do so, start with all predictors in the model; then:

(i) Determine which predictors are statistically significant and which are not. Eliminate those that are not statistically significant and create a new model using only those 3 predictors that you believe are statistically significant.

(ii) Print the summary of the regression model created in (b)(i) and comment on how well the model fits the data by studying the R2 , RSE and RMSE. (Print out the values of R2 , RSE, and RMSE.)

(iii) Plot the residuals of the model.

(iv) Plot a histogram of the residuals of the model. Does the histogram follow a Gaussian distribution? What can you say about the distribution of the residuals?

(v) Comparing the summaries of the model produced in (a) and in (b), including residual analysis of each model. Which model do you think is better, and why?

(c) Using the predict() method, fit the **test dataset** to the model you created in (b) and perform the analysis below.

To best assist you with the analysis, create a new data frame as follows: get the predictions (fitted values) and put them in a new dataframe as a column vector. Put the test response variable as the second column vector in the new dataframe.

(d) Count how many of the fitted values matched the **mpg** in the **test** dataset at a 95% confidence level by creating confidence intervals. To be considered a match, the **response** value of each observation in in the test dataset should be in the CI created by predict(). Note that we have the ground truth in our test dataset, so we can count the matches as one measure of accuracy.

To help facilitate this counting, add a column (called **Matches**) in the data frame you created above that contains 1 if the response value was in the CI, and 0 otherwise. To find out the total observations correctly predicted in our test dataset, simply count the number of 1's in the Matches column. Print this data frame, and following it, print the number of 1's in the **Matches** column. Your output should look like:

Prediction	Response	Lower	Upper	Matches
56.23	56.10	55.11	57.21	1
35.10	34.19	34.98	38.38	0

…

Total observations correctly predicted: XX

(Hint: To compute Matches, you must use the apply() function on the above data frame.)

(e) Follow the same instructions in (d) except this time, you will be using a **prediction** interval.

Count how many of the fitted values matched the **mpg** in the **test** dataset at a 95% confidence level by creating **prediction** intervals. To be considered a match, the **response** value of each observation in in the test dataset should be in the prediction interval created by predict(). Note that we have the ground truth in our test dataset, so we can count the matches as one measure of accuracy.

To help facilitate this counting, add a column (called **Matches**) in the data frame you created above that contains 1 if the response value was in the CI, and 0 otherwise. To find out the total observations correctly predicted in our test dataset, simply count the number of 1's in the **Matches** column.

Print this data frame, and following it, print the number of 1's in the **Matches** column. Your output should look like:

Prediction	Response	Lower	Upper	Matches
56.23	56.10	50.54	62.20	1
35.10	34.19	29.92	40.44	1

…

Total observations correctly predicted: XX

(f) Comment on the results of (d) and (e):

(i) Which of (d) or (e) results in more matches?

(ii) Why? (1 – 2 sentences.)