HW16

110078509 Discussed with 110078514, 110078501

202206

```
# Load the data and remove missing values
cars <- read.table("auto-data .txt", header=FALSE, na.strings = "?")
names(cars) <- c("mpg", "cylinders", "displacement", "horsepower", "weight", "acceleration",
                 "model_year", "origin", "car_name")
cars$car_name <- NULL # Drop y
cars <- na.omit(cars) # Drop NA

# Shuffle the rows of cars
set.seed(27935752)
cars <- cars[sample(1:nrow(cars)),]


# Create a log transformed dataset also
cars_log <- with(cars, data.frame(log(mpg), log(cylinders), log(displacement), log(horsepowe
r), log(weight), log(acceleration), model_year, origin))
# Linear model of mpg over all the variables that don't have multicollinearity
cars_lm <- lm(mpg ~ weight + acceleration + model_year + factor(origin), data=cars)

# Linear model of log mpg over all the log variables that don't have multicollinearity
cars_log_lm <- lm(log.mpg. ~ log.weight. + log.acceleration. + model_year + factor(origin), d
ata=cars_log)

# Linear model of log mpg over all the log variables, including multicollinear terms!
cars_log_full_lm <- lm(log.mpg. ~ log.cylinders. + log.displacement. + log.horsepower. +
                       log.weight. + log.acceleration. + model_year + factor(origin),
                       data=cars_log)
```

Question 1) Let's work with the cars_log model and test some basic prediction. Split the data into train and test sets (70:30) and try to predict log.mpg. for the smaller test set:

a. Retrain the cars_log_lm model on just the training dataset (call the new model: lm_trained)

Show the coefficients of the trained model

```
set.seed(27935752) # split to train , test set

train_indices <- sample(1: nrow(cars_log), size = 0.7 * nrow(cars_log))
# sample index from 1 ~392

train_set <- cars_log[train_indices, ] # 0.7 data  = 274
test_set <- cars_log[-train_indices, ] # 0.3 data  = 118

lm_trained <- lm(log.mpg. ~ log.weight. + log.acceleration. + model_year + factor(origin),
data=train_set)

lm_trained$coefficients
```

```
##       (Intercept)       log.weight. log.acceleration.        model_year
##        7.32742954       -0.87233187        0.08276868        0.03243753
##   factor(origin)2   factor(origin)3
##        0.05215223        0.02767976
```

b. Use the lm_trained model to predict the log.mpg. of the test dataset

What is the in-sample mean-square fitting error (MSEIS) of the trained model? What is the out-of-sample mean-square prediction error (MSEOOS) of the test dataset?

```
# In-Sample
mse_is <- mean(residuals(lm_trained)^2)

# Out of Sample
mpg_predicted <-predict(cars_log_lm,test_set)
mpg_actual <- test_set$log.mpg.
mse_oos <- mean( (mpg_predicted - mpg_actual)^2 )

sprintf('In Sample MSE : % f, Out Sample MSE: % f', mse_is,mse_oos)
```

```
## [1] "In Sample MSE :  0.012492, Out Sample MSE:  0.015012"
```

c. Show a data frame of the test set's actual log.mpg., the predicted values, and the difference of the two (predictive error); Just show us the first several rows

```
c_ans <- data.frame(Actual = mpg_actual, Prediction = mpg_predicted)|>
    mutate(predictive_error = Actual - Prediction)

head(c_ans)
```

```
##      Actual Prediction predictive_error
## 3  3.673766   3.474516       0.19925024
## 8  2.944439   2.794318       0.15012078
## 16 2.890372   2.905463      -0.01509074
## 18 3.258097   3.393640      -0.13554362
## 19 3.258097   3.365669      -0.10757221
## 20 2.890372   2.961792      -0.07142018
```

Question 2) Let's see how our three large models described in the setup at the top perform predictively!

a. Report the MSEIS of the cars_lm, cars_log_lm, and cars_log_full_lm;

Which model has the best (lowest) mean-square fitting error? Which has the worst?

```
# In sample
is_mse.cars <- mean(residuals(cars_lm)^2)
is_mse.cars_log <- mean(residuals(cars_log_lm)^2)
is_mse.cars_full_log <- mean(residuals(cars_log_full_lm)^2)
cat("In-sample MSE","\ncars_lm:", is_mse.cars, "\ncars_log_lm:", is_mse.cars_log, "\ncars_log
_full_lm:", is_mse.cars_full_log)
```

```
## In-sample MSE
## cars_lm: 10.97164
## cars_log_lm: 0.01332245
## cars_log_full_lm: 0.01246619
```

The cars_full_log_lm is the best and the cars_lm is the worst.

---

b. Try writing a function that performs k-fold cross-validation.

Ans:

```
# Version 2

k_fold_mse <- function(dataset, k=10, model) {

  # shuffle the data\
  set.seed(27935752)
  data_shuffle <- dataset[sample(1:nrow(dataset),replace = F),]
  # The input: structual dataframe and the specific k value
  # it return the MSEOOS
    fold_pred_errors <- sapply(1:k, \(i) {
        fold_i_pe(i, k, data_shuffle, model)
    })
    pred_errors <- unlist(fold_pred_errors)
    mean(pred_errors^2)
}

fold_i_pe <- function(i, k, dataset, model) {
    folds <- cut(seq(1,nrow(dataset)),breaks=k,labels=FALSE)
    testIndexes <- which(folds==i,arr.ind=TRUE)  # find the index of the list that match the
 condition

    test_set <- dataset[testIndexes, ]
    train_set <- dataset[-testIndexes, ]

    f <- format(terms(model)) %>% paste(., collapse = " ") %>% as.formula()
    trained_model <- lm(f,train_set)
    formula <- trained_model$model %>% names() # 取出Y的 column name

    predictions <- predict(trained_model, test_set)
    real_value <- test_set[,colnames(test_set) == formula[1]]
    real_value - predictions
}
```

Ans:

i. Use/modify your k-fold cross-validation function to find and report the MSEOOS for cars_lm – recall that this non-transformed data/model has non-linearities

Ans:

```
cars_lm_mse <- k_fold_mse(cars, k=10, cars_lm)
cars_lm_mse
```

```
## [1] 11.37571
```

---

ii. Use/modify your k-fold cross-validation function to find and report the MSEOOS for cars_log_lm – does it predict better than cars_lm? Was non-linearity harming predictions?

```
k_fold_mse(cars_log, k=10, cars_log_lm)
```

```
## [1] 0.01381873
```

Ans: Yes. It's better, And yes, the non-linearity harm predictions.

---

iii. Use/modify your k-fold cross-validation function to find and report the MSEOOS for cars_log_lm_full this model has collinear terms; so does multicollinearity seem to harm the predictions?

```
k_fold_mse(cars_log, k=10, cars_log_full_lm)
```

```
## [1] 0.01322936
```

Ans: No, The multicollinearity does not seem to harm the predictions.

---

c. Check if your k_fold_mse function can do as many folds as there are rows in the data (i.e., k=392). Report the MSEOOS for the cars_log_lm model with k=392.

```
k_fold_mse(cars_log, k=392, cars_log_lm)
```

```
## [1] 0.01379209
```

Ans: Yes, it can.