HW16

110077443

6/2/2022

Credit: 110077421 for assisting with the function in 2(b)

Please note that all code in this document is presented in a grey box and the output reflected below each box

- The below code allows lengthy lines of comments to display neatly within the grey box (wrapping it)

```
knitr::opts_chunk$set(tidy.opts = list(width.cutoff = 60), tidy = TRUE)
```

Setting up all the models we will need:

```
# 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
cars <- na.omit(cars)

# 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(horsepower), 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), data = cars_log)

# Linear model of log mpg over all the log variables,
```

```r
# 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)
```

1) Split the data into train and test sets (70:30) and try to predict log.mpg. for the smaller test set:

```r
# Split Sample validation

# Training
set.seed(27935752)  # same seed as professor
train_indices <- sample(1:nrow(cars_log), size = 0.7 * nrow(cars_log))
train_set <- cars_log[train_indices, ]
test_set <- cars_log[-train_indices, ]
```

a) Retrain the cars_log_lm model on just the training dataset

```r
# Retrain cars_log_lm model
lm_trained <- lm(log.mpg. ~ log.weight. + log.acceleration. +
    model_year + factor(origin), data = train_set)

# Show coefficients
require(knitr)  # Used for creating tables with kable function
kable(lm_trained$coefficients |>
    round(4), caption = "Coefficients", align = "c")  # Print coefficients in a table
```

Table 1: Coefficients

	x
(Intercept)	7.3274
log.weight.	-0.8723
log.acceleration.	0.0828
model_year	0.0324
factor(origin)2	0.0522
factor(origin)3	0.0277

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

```r
# Predicting
mpg_predicted <- predict(lm_trained, test_set)  # y-hat
mpg_actual <- test_set$log.mpg.  # y

# MSE(IS)
mse_is <- mean((train_set$log.mpg. - lm_trained$fitted.values)^2)
mse_is
```

ANSWER > [1] 0.01249181

```r
# MSE(OOS)
mse_oos <- mean((mpg_predicted - mpg_actual)^2)
mse_oos  # Print
```

ANSWER > [1] 0.01559438

c) Show a data frame of the test set's actual log.mpg., the predicted values, and the difference of the two (predictive error)

```r
# Predicted error
pred_err <- mpg_actual - mpg_predicted

# Creating data frame with first six rows
pred_data <- head(data.frame(mpg_actual, mpg_predicted, pred_err))

kable(pred_data |>
    round(4), caption = "Split Sample Validation", align = "c")  # Print first six rows in a table
```

Table 2: Split Sample Validation

	mpg_actual	mpg_predicted	pred_err
3	3.6738	3.4667	0.2071
8	2.9444	2.7895	0.1550
16	2.8904	2.9077	-0.0174
18	3.2581	3.3911	-0.1330
19	3.2581	3.3543	-0.0962
20	2.8904	2.9568	-0.0664

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

a) Report the MSE(IS) 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?

```r
# MSE_IS of cars_lm
mse_is_cars <- mean((cars$mpg - cars_lm$fitted.values)^2)

# MSE_IS of cars_log_lm
mse_is_log <- mean((cars_log$log.mpg. - cars_log_lm$fitted.values)^2)

# MSE_IS of cars_log_full_lm
mse_is_log_full <- mean((cars_log$log.mpg. - cars_log_full_lm$fitted.values)^2)
```

```r
# Creating data frame to compare the three model's MSE_IS
mse_is_df <- data.frame(mse_is_cars, mse_is_log, mse_is_log_full)

kable(mse_is_df |>
    round(4), caption = "MSE_is Comparison", align = "c")  # Print the three MSE_IS in a table
```

Table 3: MSE_is Comparison

mse_is_cars	mse_is_log	mse_is_log_full
10.9716	0.0133	0.0125

ANSWER >

- From the table we see that the MSE_is from the **cars_log_full_lm** model has the **lowest** mean-square fitting error.
- The MSE_is from the **cars_lm** model has the **worst**.

b) Writing a function that performs k-fold cross-validation

```r
k <- 10

k_fold_mse = function(model, dataset, k) {
    fold_pred_errors = sapply(1:k, function(i) {
        fold_i_pe(i, k, dataset, model)
    })
    pred_errors = unlist(fold_pred_errors)
    mean(pred_errors^2)
}
# Calculates prediction error for fold i out of k
fold_i_pe = function(i, k, dataset, model) {
    folds = cut(1:nrow(dataset), k, labels = FALSE)
    test_indices = which(folds == i)
    test_set = dataset[test_indices, ]
    train_set = dataset[-test_indices, ]
    trained_model = lm(model, data = train_set)
    test_set[, 1] - predict(trained_model, test_set)  #actuals - predictions
}
```

i) Use/modify your k-fold cross-validation function to find and report the MSEOOS for cars_lm

```r
mse_oos_cars_lm <- data.frame(MSEOOS = k_fold_mse(cars_lm, cars,
    10))
kable(mse_oos_cars_lm |>
    round(4), caption = "MSE_oos - cars_lm", align = "c")
```

Table 4: MSE_oos - cars_lm

MSEOOS
11.4159

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?

```
mse_oos_cars_log_lm <- data.frame(MSEOOS = k_fold_mse(cars_log_lm,
    cars_log, 10))
kable(mse_oos_cars_log_lm |>
    round(4), caption = "MSE_oos - cars_log_lm", align = "c")
```

Table 5: MSE_oos - cars_log_lm

MSEOOS
0.0139

ANSWER >

- It predicts **better than cars_lm** because log transforming the data helps to deal with non-linearity which could harm predictions if not dealt with.

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?

```
mse_oos_cars_log_full_lm <- data.frame(MSEOOS = k_fold_mse(cars_log_full_lm,
    cars_log, 10))
kable(mse_oos_cars_log_full_lm |>
    round(4), caption = "MSE_oos - cars_log_full_lm", align = "c")
```

Table 6: MSE_oos - cars_log_full_lm

MSEOOS
0.0133

ANSWER >

- Multicollinearity **does not** seem to affect the predictions, but we know that the individual independent variable's effect on mpg could be calculated wrongly because of it.

##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.

```r
fold_test <- data.frame(MSEOOS = k_fold_mse(cars_log_lm, cars_log,
    392))
kable(fold_test |>
    round(4), caption = "MSE_oos fold test: k = 392", align = "c")
```

Table 7: MSE_oos fold test: k = 392

MSEOOS
0.0138

ANSWER >

- k_fold_mse function **can** do as many folds as there are rows in the data