HW17

108048110

2022-06-01

BACS HW - Week 17

Prerequisite

```
library(dplyr)
```

Setup

```
# loading data and remove missing values
cars <- read.table("data/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(42)
cars <- cars[sample(1:nrow(cars)),]

# Create a log transform data set 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)
```

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

paste1 <- function(obj){
  paste(round(obj, 4))
}
```

Question 1) Test basic prediction

Split the data into train and test sets (7:3)

a. Retrain the `cars_log_lm` model on just the training data set. Show the coefficients of the trained model.

```r
set.seed(42)
train_indicies <- sample(1:nrow(cars), size=0.7*nrow(cars))
train_set <- cars_log[train_indicies,]

paste0("The size of the training set is ", round(nrow(train_set)/nrow(cars)*100, 2), "%")
```

```
## [1] "The size of the training set is 69.9%"
```

```r
lm_trained <- lm(log.mpg. ~ log.weight. +
                   log.acceleration. +
                   model_year +
                   factor(origin),
                 data=train_set)
knitr::kable(lm_trained$coefficients)
```

	x
(Intercept)	7.2487100
log.weight.	-0.8603015
log.acceleration.	0.0389111

	x
model_year	0.0338931
factor(origin)2	0.0580686
factor(origin)3	0.0363888

```r
knitr::kable(cars_log_lm$coefficients)
```

	x
(Intercept)	7.4109736
log.weight.	-0.8754990
log.acceleration.	0.0543770
model_year	0.0327866
factor(origin)2	0.0561110
factor(origin)3	0.0319369

b. Use the new model to predict the `mpg` of the test data set. - What is the MSE_{IS} of the trained model? - What is the MSE_{OOS} of the test data set?

```r
test_set = cars_log[-train_indicies,]
paste0("The size of the testing set is ", round(nrow(test_set)/nrow(cars)*100, 2), "%")
```

```
## [1] "The size of the testing set is 30.1%"
```

```r
log.mpg.predicted <- predict(lm_trained, test_set)
knitr::kable(head(log.mpg.predicted))
```

	x
7	3.099254
8	3.339727
9	3.390471
15	2.952343
17	3.331828
19	3.396228

```r
mpg_fitted <- fitted(lm_trained) # y hat
fit_error <- train_set$log.mpg. - mpg_fitted # residuals(lm_trained)
mse_is <- mean(fit_error^2)
paste1(mse_is)
```

```
## [1] "0.0126"
```

```r
mpg_actual <- test_set$log.mpg.
pred_error <- mpg_actual-log.mpg.predicted
mse_oos <- mean(pred_error^2)
paste1(mse_oos)
```

```
## [1] "0.0152"
```

c. Show a data frame of the test set's actual `mpg`, the predicted values, and the predictive error.

```
test_set %>%
  transmute(actual_mpg = mpg_actual,
         predicted_values = log.mpg.predicted,
         pred_error = pred_error) -> comparison_df
knitr::kable(comparison_df %>% head)
```

	actual_mpg	predicted_values	pred_error
7	3.178054	3.099254	0.0787999
8	3.433987	3.339727	0.0942604
9	3.433987	3.390471	0.0435164
15	2.564949	2.952343	-0.3873938
17	3.258097	3.331828	-0.0737318
19	3.367296	3.396228	-0.0289319

Question 2) How the 3 models described at the top perform predictively?

```
MSE_is <- function(model, data, log=FALSE){
  if(log==FALSE){
    fit_error <- model$fitted.values-data[,1]
  }else{
    fit_error <- exp(model$fitted.values)-exp(data[,1])
  }
  insample_mse <- mean(fit_error^2)
  return(insample_mse)
}
```

a. Report the MSE_{IS} of the 3 models described in the setup; Which model has the best mean-square fitting error? Which has the worst?

```
MSE_is(cars_lm, cars)
```

```
## [1] 10.97164
```

```
MSE_is(cars_log_lm, cars_log, log=TRUE)
```

```
## [1] 8.305048
```

```
MSE_is(cars_log_all_lm, cars_log, log=TRUE)
```

```
## [1] 7.982862
```

Ans. `cars_log_all_lm` has the best MSE, while `cars_lm` has the worst.

b. Write a function that performs k-fold cross-validation.

```r
# calculates prediction error for fold i out of k
fold_i_pred_err <- function(i, k, dataset, predictors){
  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(predictors, data=train_set)

  predictions <- predict(trained_model, test_set)
  test_set[,1]-predictions
}


# calculates mse_oos across all folds
k_fold_mse <- function(predictors, data, k=10){
  shuffled_indices = sample(1:nrow(data))
  data = data[shuffled_indices,]

  fold_pred_error <- sapply(1:k, \(i){
    fold_i_pred_err(i, k, data, predictors)
  })
  pred_error <- unlist(fold_pred_error)
  mse <- \(errs){mean(errs^2)}
  c(in_sample = mse(residuals(predictors)), out_of_sample = mse(pred_error))}
```

- ***i.*** Modify your k-fold cross-validation function to find and report the MSE_{OOS} for `cars_lm`.

  ```r
  k_fold_mse(cars_lm, data=cars, k=10)
  ```

  ```
  ##     in_sample out_of_sample
  ##      10.97164      11.58398
  ```

- ***ii.*** Modify your k-fold cross-validation function to find and report the MSE_{OOS} for `cars_log_lm` – does it predict better than `cars_lm`? Was non-linearity harming predictions?

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

  ```
  ##     in_sample out_of_sample
  ##    0.01332245    0.01399654
  ```

  - **Ans.** Non-linearity does not seem to harm predictions.

- ***iii.*** Modify your k-fold cross-validation function to find and report the MSE_{OOS} for `cars_log_all_lm` - does multicollinearity seem to harm the predictions?

  ```r
  k_fold_mse(cars_log_all_lm, data=cars_log)
  ```

  ```
  ##     in_sample out_of_sample
  ##    0.01246619    0.01312540
  ```

  - **Ans.** Multicollinearity does not seem to harm the predictions.

c. Check if your *k_fold_mse* can do as many folds as there are rows in the data. Report the MSE_{OOS} for the `cars_log_lm` model with k=392.

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

```
##     in_sample out_of_sample
##    0.01332245    0.01379209
```