BACS-hw16-107070004

```
# 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(wei
# 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, 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)
n <- nrow(cars_log)
shuffled_cars_log <- cars_log[sample(n), ]
train_indices <- 1:round(0.7 * n)
test_indices <- (round(0.7 * n) + 1):n
train_set <- shuffled_cars_log[train_indices,]
test_set <- shuffled_cars_log[test_indices,]
lm_trained <- lm(log.mpg. ~ log.weight. + log.acceleration. + model_year + factor(origin),
                 data=train_set)
lm_trained
```

```
##
## Call:
```

```
## lm(formula = log.mpg. ~ log.weight. + log.acceleration. + model_year +
##     factor(origin), data = train_set)
##
## Coefficients:
##       (Intercept)        log.weight.  log.acceleration.         model_year
##           7.32743           -0.87233            0.08277            0.03244
##   factor(origin)2    factor(origin)3
##           0.05215            0.02768
```

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?

```
mean((train_set$log.mpg. - lm_trained$fitted.values)^2)
```

```
## [1] 0.01249181
```

- What is the out-of-sample mean-square prediction error (MSEOOS) of the test dataset?

```
mpg_actual <- test_set$log.mpg.
mpg_predicted <- predict(lm_trained, test_set)
pred_err <- mpg_actual - mpg_predicted
mse_oos <- mean( (mpg_predicted-mpg_actual)^2 )
mse_oos
```

```
## [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); Just show us the first several rows

```
# actual log.mpg.
head(mpg_actual)
```

```
## [1] 3.025291 2.639057 3.005683 3.258097 3.583519 2.639057
```

```
# predicted values
head(mpg_predicted)
```

```
##       88      213      286      177      392      313
## 2.997752 2.465390 3.188372 3.151940 3.643497 2.553277
```

```
# predictive error
head(pred_err)
```

```
##          88         213         286         177         392         313
##  0.02753901  0.17366739 -0.18268925  0.10615685 -0.05997850  0.08578048
```

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?

```
# cars_lm
mean((cars$mpg - cars_lm$fitted.values)^2)
```

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

```
# cars_log_lm
mean((cars_log$log.mpg. - cars_log_lm$fitted.values)^2)
```

```
## [1] 0.01332245
```

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

```
## [1] 0.01246619
```

b. Try writing a function that performs k-fold cross-validation (see class notes and ask in Teams for hints!). Name your function k_fold_mse(dataset, k=10, ...) – it should return the MSEOOS of the operation. Your function may must accept a dataset and number of folds (k) but can also have whatever other parameters you wish.

```
fold_i_pe <- function(i, k, dataset, lm, actuals) {
  folds <- cut(1:nrow(dataset), k, labels = FALSE)
  test_indices <- which(folds == i)
  test_set <- dataset[test_indices,]
  trained_model <- lm
  predictions <- predict(trained_model, test_set)
  actuals[test_indices] - predictions
}
k_fold_mse <- function(dataset, k, lm, actuals) {
  fold_pred_errors <- sapply(1:k,\(i) {
    fold_i_pe(i, k, dataset, lm, actuals)
  })
  pred_errors <- unlist(fold_pred_errors)
  mean(pred_errors^2)
}
```

(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

```
k_fold_mse(cars, 10, cars_lm, cars$mpg)
```

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

(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, 10, cars_log_lm, cars_log$log.mpg.)
```

```
## [1] 0.01332245
```

It doesn't predict better than cars_lm. But predict better than The non-linearity is silghtly harming 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, 10, cars_log_full_lm, cars_log$log.mpg.)
```

```
## [1] 0.01246619
```

Yes. The non-linearity is harming 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, 392, cars_log_lm, cars_log$log.mpg.)
```

```
## [1] 0.01332245
```

It doesn't predict better than cars_lm. The non-linearity is harming predictions.