Student ID: 112077423

```r
library(ggplot2)
library(data.table)
library(tidyr)
library(rpart)
```

```
## Warning:     'rpart'          R     4.3.3
```

```r
# 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)
# IMPORTANT: Shuffle the rows of data in advance for this project!
set.seed(27935752) # use your own seed, or use this one to compare to next class notes
cars <- cars[sample(1:nrow(cars)),]
# DV and IV of formulas we are interested in
cars_full <- mpg ~ cylinders + displacement + horsepower + weight + acceleration +
                   model_year + factor(origin)
cars_reduced <- mpg ~ weight + acceleration + model_year + factor(origin)
cars_full_poly2 <- mpg ~ poly(cylinders, 2) + poly(displacement, 2) + poly(horsepower, 2) +
                         poly(weight, 2) + poly(acceleration, 2) + model_year +
                         factor(origin)
cars_reduced_poly2 <- mpg ~ poly(weight, 2) + poly(acceleration,2) + model_year +
                            factor(origin)
cars_reduced_poly6 <- mpg ~ poly(weight, 6) + poly(acceleration,6) + model_year +
                            factor(origin)
```

```r
lm_full <- lm(cars_full, data=cars)
lm_reduced <- lm(cars_reduced, data=cars)
lm_poly2_full <- lm(cars_full_poly2, data=cars)
lm_poly2_reduced <- lm(cars_reduced_poly2, data=cars)
lm_poly6_reduced <- lm(cars_reduced_poly6, data=cars)
rt_full <- rpart(cars_full, data=cars)
rt_reduced <- rpart(cars_reduced, data=cars)
```

Question 1

Compute and report the in-sample fitting error of all the models described above.

```r
mse_in <- function(mpg_lm, name) {
  mse_is <- mean(residuals(mpg_lm)^2)
  out1 <- paste('In-sample fitting error of', name)
  out2 <- paste('=', round(mse_is, 2))
  cat(out1, out2, sep=' ')
  cat('\n')
}

mse_in(lm_full, 'lm_full')
```

```
## In-sample fitting error of lm_full = 10.68
```

```
mse_in(lm_reduced, 'lm_reduced')
```

```
## In-sample fitting error of lm_reduced = 10.97
```

```
mse_in(lm_poly2_full, 'lm_poly2_full')
```

```
## In-sample fitting error of lm_poly2_full = 7.92
```

```
mse_in(lm_poly2_reduced, 'lm_poly2_reduced')
```

```
## In-sample fitting error of lm_poly2_reduced = 8.36
```

```
mse_in(lm_poly6_reduced, 'lm_poly6_reduced')
```

```
## In-sample fitting error of lm_poly6_reduced = 8.25
```

```
mse_in(rt_full, 'rt_full')
```

```
## In-sample fitting error of rt_full = 9.16
```

```
mse_in(rt_reduced, 'rt_reduced')
```

```
## In-sample fitting error of rt_reduced = 9.5
```

Question 2

Let's work with the lm_reduced model and test its predictive performance with split-sample testing.

```
train_indices <- sample(1:nrow(cars), size=0.70*nrow(cars))
train_set <- cars[train_indices,]
test_set <- cars[-train_indices,]

trained_model <- lm(cars_reduced, data=train_set)
coefficients(trained_model)
```

```
##     (Intercept)          weight    acceleration      model_year factor(origin)2
##   -19.884850373    -0.005691225     0.059620087     0.772666424     2.218726662
## factor(origin)3
##     2.148123177
```

```
mpg_predicted <- predict(trained_model, test_set)
mpg_actual <- test_set$mpg
mse_out <- mean( (mpg_actual - mpg_predicted)^2 )

mse_is <- mean((train_set$mpg - fitted(trained_model))^2)

out1 <- paste('In-sample mean-square fitting error is', round(mse_is, 2))
out2 <- paste('Out-of-sample mean-square prediction error is', round(mse_out, 2))
cat(out1, out2, sep='\n')
```

```
## In-sample mean-square fitting error is 12.11
## Out-of-sample mean-square prediction error is 8.51
```

```
tmp <- mpg_actual - mpg_predicted
df <- cbind(test_set$mpg, mpg_predicted, tmp)
colnames(df) <- c('actual mpg', 'predicted mpg', 'predictive error')
head(df)
```

```
##     actual mpg predicted mpg predictive error
## 372         29      30.05738      -1.05737552
## 214         13      16.47532      -3.47532289
## 81          22      23.07057      -1.07057048
## 384         38      35.33296       2.66703558
## 85          27      26.92741       0.07258502
## 103         26      28.89266      -2.89265898
```

Question 3(a)

Let's use k-fold cross validation (k-fold CV) to see how all these models perform predictively

(i) *Use your k_fold_mse function to find and report the 10-fold CV MSEout for all models.*

```
# Calculate prediction error for fold i out of k
fold_i_pe <- function(i, k, dataset, model, mode) {
  folds <- cut(1:nrow(dataset), k, labels = FALSE)
  test_indices <- which(folds==i)
  test_set <- dataset[test_indices,]
  train_set <- dataset[-test_indices,]
  if(mode) { trained_model <- rpart(dataset, data=train_set) }
  else { trained_model <- lm(model, data=train_set) }
  predictions <- predict(trained_model, test_set)
  return(test_set$mpg - predictions)
}
# Calculate mse_out across all folds
k_fold_mse <- function(model, dataset=cars, k=10, mode=0) {
  shuffled <- dataset[sample(nrow(dataset)),]
  fold_pred_errors <- sapply(1:k, \(i) { fold_i_pe(i, k, shuffled, model, mode) })
  pred_errors <- unlist(fold_pred_errors)
  return(mean(pred_errors^2))
}

out <- paste('MSEout of cars_full =', round(k_fold_mse(cars_full), 2))
cat(out, '\n')
```

```
## MSEout of cars_full = 11.34
```

```
out <- paste('MSEout of cars_reduced =', round(k_fold_mse(cars_reduced), 2))
cat(out, '\n')
```

```
## MSEout of cars_reduced = 11.4
```

```r
out <- paste('MSEout of cars_full_poly2 =', round(k_fold_mse(cars_full_poly2), 2))
cat(out, '\n')
```

```
## MSEout of cars_full_poly2 = 8.61
```

```r
out <- paste('MSEout of cars_reduced_poly2 =', round(k_fold_mse(cars_reduced_poly2), 2))
cat(out, '\n')
```

```
## MSEout of cars_reduced_poly2 = 8.91
```

```r
out <- paste('MSEout of cars_reduced_poly6 =', round(k_fold_mse(cars_reduced_poly6), 2))
cat(out, '\n')
```

```
## MSEout of cars_reduced_poly6 = 9.16
```

```r
out <- paste('MSEout of cars_full (rt) =', round(k_fold_mse(cars_full, mode=1), 2))
cat(out, '\n')
```

```
## MSEout of cars_full (rt) = 12.79
```

```r
out <- paste('MSEout of cars_reduced (rt) =', round(k_fold_mse(cars_reduced, mode=1), 2))
cat(out, '\n')
```

```
## MSEout of cars_reduced (rt) = 13.83
```

(ii) *For all the models, which is bigger — the fit error (MSEin) or the prediction error (MSEout)?*

Prediction error (MSEout) is larger. The possible reason is that to compute MSEin, we use a single dataset. Whereas to compute MSEout, we have two different datasets to train and test.

(iii) *Does the 10-fold MSEout of a model remain stable (same value) if you re-estimate it over and over again, or does it vary?*

```r
cat(round(k_fold_mse(cars_full), 2), '\n')
```

```
## 11.38
```

```r
cat(round(k_fold_mse(cars_full), 2), '\n')
```

```
## 11.21
```

```r
cat(round(k_fold_mse(cars_full), 2), '\n')
```

```
## 11.41
```

```r
cat(round(k_fold_mse(cars_full), 2), '\n')
```

```
## 11.31
```

Since we shuffle the data each time k_fold_mse function is called, results slightly vary each time.

Question 3(b)

(i) *How many rows are in the training dataset and test dataset of each iteration of k-fold CV when k=392?*

We split data into k folds. Since the number of observations we have is 392, therefore each fold has N/k = 392/392 = 1 row. We iteratively train on k-1 folds and test on 1 fold each time. So, the test set has 1 row (only 1 fold) and the train set has 391 rows (391 folds with 1 row each).

(ii) *Report the k-fold CV MSEout for all models using k=392.*

```r
out <- paste('MSEout of cars_full =', round(k_fold_mse(cars_full, k=392), 2))
cat(out, '\n')
```

```
## MSEout of cars_full = 11.29
```

```r
out <- paste('MSEout of cars_reduced =', round(k_fold_mse(cars_reduced, k=392), 2))
cat(out, '\n')
```

```
## MSEout of cars_reduced = 11.38
```

```r
out <- paste('MSEout of cars_full_poly2 =', round(k_fold_mse(cars_full_poly2, k=392), 2))
cat(out, '\n')
```

```
## MSEout of cars_full_poly2 = 8.61
```

```r
out <- paste('MSEout of cars_reduced_poly2 =', round(k_fold_mse(cars_reduced_poly2, k=392), 2))
cat(out, '\n')
```

```
## MSEout of cars_reduced_poly2 = 8.79
```

```r
out <- paste('MSEout of cars_reduced_poly6 =', round(k_fold_mse(cars_reduced_poly6, k=392), 2))
cat(out, '\n')
```

```
## MSEout of cars_reduced_poly6 = 9.18
```

```r
out <- paste('MSEout of cars_full (rt) =', round(k_fold_mse(cars_full, mode=1, k=392), 2))
cat(out, '\n')
```

```
## MSEout of cars_full (rt) = 12.77
```

```
out <- paste('MSEout of cars_reduced (rt) =', round(k_fold_mse(cars_reduced, mode=1, k=392), 2))
cat(out, '\n')
```

```
## MSEout of cars_reduced (rt) = 12.77
```

(iii) *When k=392, does the MSEout of a model remain stable (same value) if you re-estimate it over and over again, or does it vary?*

```
cat(round(k_fold_mse(cars_full, k=392), 2), '\n')
```

```
## 11.29
```

```
cat(round(k_fold_mse(cars_full, k=392), 2), '\n')
```

```
## 11.29
```

```
cat(round(k_fold_mse(cars_full, k=392), 2), '\n')
```

```
## 11.29
```

```
cat(round(k_fold_mse(cars_full, k=392), 2), '\n')
```

```
## 11.29
```

Values are the same.

(iv) *Looking at the fit error (MSEin) and prediction error (MSEout; k=392) of the full models versus their reduced counterparts (with the same training technique), does multicollinearity present in the full models seem to hurt their fit error and/or prediction error?*

The multicollinearity present in the full models doesn't seem to hurt their fit error and/or prediction error. But analysts are still scared of multicollinearity because it can lead to unreliable and unstable estimates of regression coefficients.

(v) *Look at the fit error and prediction error (k=392) of the reduced quadratic versus 6th order polynomial regressions — did adding more higher-order terms hurt the fit and/or predictions?*

As we can see from the results, adding more higher-order terms does hurt the fit. The possible reason is overfitting.