bacs\_hw14

110071010

2024-05-23

# Setup

# 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(111555777) # 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)

library(rpart) # for regression trees  
  
lm\_full <- lm(formula = cars\_full, data=cars)  
lm\_reduced <- lm(formula = cars\_reduced, data=cars)  
lm\_poly2\_full <- lm(formula = cars\_full\_poly2, data=cars)  
lm\_poly2\_reduced <- lm(formula = cars\_reduced\_poly2, data=cars)  
lm\_poly6\_reduced <- lm(formula = cars\_reduced\_poly6, data=cars)  
rt\_full <- rpart(formula = cars\_full, data=cars)  
rt\_reduced <- rpart(formula = cars\_reduced, data=cars)

# Question 1

Compute and report the in-sample fitting error (MSEin) of all the models described above. It will be easier to first write a function called mse\_in(…) that returns the fitting error of a single estimated model; you can apply that function to each model (feel free to ask us for help!). We will discuss these results later.

mse\_in <- function(model){  
 mean(residuals(model)^2)  
 # mean((cars$mpg - fitted(model))^2)  
}  
  
mse\_lm\_full <- mse\_in(lm\_full)  
mse\_lm\_reduced <- mse\_in(lm\_reduced)  
mse\_lm\_poly2\_full <- mse\_in(lm\_poly2\_full)  
mse\_lm\_poly2\_reduced <- mse\_in(lm\_poly2\_reduced)  
mse\_lm\_poly6\_reduced <- mse\_in(lm\_poly6\_reduced)  
mse\_rt\_full <- mse\_in(rt\_full)  
mse\_rt\_reduced <- mse\_in(rt\_reduced)  
  
cat("mse\_lm\_full : ", mse\_lm\_full, "\n")  
cat("mse\_lm\_reduced : ", mse\_lm\_reduced, "\n")  
cat("mse\_lm\_poly2\_full : ", mse\_lm\_poly2\_full, "\n")  
cat("mse\_lm\_poly2\_reduced : ", mse\_lm\_poly2\_reduced, "\n")  
cat("mse\_lm\_poly6\_reduced : ", mse\_lm\_poly6\_reduced, "\n")  
cat("mse\_rt\_full : ", mse\_rt\_full, "\n")  
cat("mse\_rt\_reduced : ", mse\_rt\_reduced, "\n")

## mse\_lm\_full : 10.68212   
## mse\_lm\_reduced : 10.97164   
## mse\_lm\_poly2\_full : 7.91903   
## mse\_lm\_poly2\_reduced : 8.364546   
## mse\_lm\_poly6\_reduced : 8.254377   
## mse\_rt\_full : 9.155146   
## mse\_rt\_reduced : 9.501344

# Question 2

Let’s try some simple evaluation of prediction error. Let’s work with the lm\_reduced model and test its predictive performance with split-sample testing:

## 2a

Split the data into 70:30 for training:test (did you remember to shuffle the data earlier?)

set.seed(111555777)  
  
# Split the data into 70:30 for training:test  
train\_indices <- sample(1:nrow(cars), size = 0.70\*nrow(cars))  
train\_set <- cars[train\_indices,]   
test\_set <- cars[-train\_indices,]

## 2b

Retrain the lm\_reduced model on just the training dataset (call the new model: trained\_model). Show the coefficients of the trained model.

# Retrain the lm\_reduced model on just the training dataset  
trained\_model <- lm(formula = cars\_reduced, data = train\_set)  
  
# Show the coefficients of the trained model  
summary(trained\_model)$coefficients

## Estimate Std. Error t value Pr(>|t|)  
## (Intercept) -16.774070479 4.9272440811 -3.4043514 7.647307e-04  
## weight -0.005964948 0.0003428965 -17.3957686 6.983693e-46  
## acceleration 0.013635648 0.0849423083 0.1605283 8.725858e-01  
## model\_year 0.747895113 0.0593500574 12.6014219 6.607599e-29  
## factor(origin)2 1.939320199 0.6393551468 3.0332441 2.656790e-03  
## factor(origin)3 2.168933983 0.6300771348 3.4423309 6.689422e-04

## 2c

Use the trained\_model model to predict the mpg of the test dataset. What is the in-sample mean-square fitting error (MSEin) of the trained model? What is the out-of-sample mean-square prediction error (MSEout) of the test dataset?

# Function to calculate in-sample MSE  
mse\_in <- function(model) {  
 mean(residuals(model)^2)  
}  
  
# Use the trained\_model model to predict the mpg of the test dataset  
mpg\_predicted <- predict(trained\_model, newdata = test\_set)  
  
# Calculate the in-sample MSE (MSEin) of the trained model  
cat("MSEin : ", mse\_in(trained\_model), "\n")  
  
# Calculate the out-of-sample MSE (MSEout) of the test dataset  
mpg\_actual <- test\_set$mpg  
pred\_err <- mpg\_actual - mpg\_predicted  
cat("MSEout : ", mean((mpg\_predicted - mpg\_actual)^2), "\n")

## MSEin : 11.1779   
## MSEout : 10.6634

## 2d

Show a data frame of the test set’s actual mpg values, the predicted mpg values, and the difference of the two (εout = predictive error); Just show us the first several rows of this dataframe.

data.frame(  
 "actual mpg" = mpg\_actual,  
 "predicted mpg" = mpg\_predicted,  
 "predictive error" = pred\_err   
) |> head()

## actual.mpg predicted.mpg predictive.error  
## 144 26.0 26.98782 -0.98782442  
## 214 13.0 16.04172 -3.04172145  
## 368 28.0 29.28190 -1.28189777  
## 304 31.8 32.69119 -0.89118678  
## 185 25.0 24.92728 0.07271717  
## 109 20.0 26.65617 -6.65616742

# Question 3

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

## 3a

Write a function that performs k-fold cross-validation. Name your function k\_fold\_mse(model, dataset, k=10, …) – it should return the MSEout of the operation. Your function must accept a model, dataset and number of folds (k) but can also have whatever other parameters you wish.

# Function to calculate prediction errors for fold i  
fold\_i\_pe <- function(i, k, dataset, model\_function, formula) {  
 # Split the data into k folds  
 folds <- cut(seq(1, nrow(dataset)), breaks = k, labels = FALSE)  
   
 # Identify test and training indices for the ith fold  
 test\_indices <- which(folds == i, arr.ind = TRUE)  
 test\_set <- dataset[test\_indices, ]  
 train\_set <- dataset[-test\_indices, ]  
   
 # Train the model on the training set  
 trained\_model <- model\_function(formula, data = train\_set)  
   
 # Predict on the test set  
 predictions <- predict(trained\_model, newdata = test\_set)  
   
 # Calculate prediction errors  
 actuals <- test\_set$mpg  
 prediction\_errors <- actuals - predictions  
   
 return(prediction\_errors)  
}  
  
# Function to calculate mean squared error across all folds  
k\_fold\_mse <- function(dataset, k = 10, model\_function, formula) {  
 # Calculate prediction errors for each fold  
 fold\_pred\_errors <- sapply(1:k, function(i) {  
 fold\_i\_pe(i, k, dataset, model\_function, formula)  
 })  
   
 # Combine all prediction errors into a single vector  
 pred\_errors <- unlist(fold\_pred\_errors)  
   
 # Calculate and return the mean squared error  
 mse <- mean(pred\_errors^2)  
 return(mse)  
}

### i

Use your k\_fold\_mse function to find and report the 10-fold CV MSEout for all models.

mse\_full <- k\_fold\_mse(dataset=cars, k=10,lm, formula=cars\_full)   
mse\_reduced <- k\_fold\_mse(dataset=cars, k=10,lm, formula=cars\_reduced)  
mse\_full\_poly2 <- k\_fold\_mse(dataset=cars, k=10,lm, formula=cars\_full\_poly2)  
mse\_reduced\_poly2 <- k\_fold\_mse(dataset=cars, k=10,lm, formula=cars\_reduced\_poly2)  
mse\_reduced\_poly6 <- k\_fold\_mse(dataset=cars, k=10,lm, formula=cars\_reduced\_poly6)  
mse\_rpart\_reduced\_poly2 <- k\_fold\_mse(dataset=cars, k=10,rpart, formula=cars\_reduced\_poly2)  
mse\_rpart\_reduced\_poly6 <- k\_fold\_mse(dataset=cars, k=10,rpart, formula=cars\_reduced\_poly6)  
  
data.frame(Model = c("lm\_full", "lm\_reduced", "lm\_full\_poly2", "lm\_reduced\_poly2", "lm\_reduced\_poly6", "rt\_reduced\_poly2", "rt\_reduced\_poly6"),  
 MSEout = c(mse\_full, mse\_reduced, mse\_full\_poly2, mse\_reduced\_poly2,   
 mse\_reduced\_poly6, mse\_rpart\_reduced\_poly2, mse\_rpart\_reduced\_poly6)  
)

## Model MSEout  
## 1 lm\_full 11.227264  
## 2 lm\_reduced 11.350700  
## 3 lm\_full\_poly2 8.677378  
## 4 lm\_reduced\_poly2 8.807537  
## 5 lm\_reduced\_poly6 9.379082  
## 6 rt\_reduced\_poly2 11.865570  
## 7 rt\_reduced\_poly6 11.806473

### ii

**Question**

For all the models, which is bigger — the fit error (MSEin) or the prediction error (MSEout)? (optional: why do you think that is?)

**Answer**

MSEout > MSEin, as there are overfitting problems for in-sample.

### iii

**Question**

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? (show a few repetitions for any model and decide!)

set.seed(111555777)  
  
n\_repeats <- 5  
mse\_out\_repeats <- replicate(n\_repeats, {  
 k\_fold\_mse(cars, k = 10, model\_function = lm, formula = cars\_full)  
})  
  
# Display the results  
data.frame(Repetition = 1:n\_repeats, MSEout = mse\_out\_repeats)

## Repetition MSEout  
## 1 1 11.22726  
## 2 2 11.22726  
## 3 3 11.22726  
## 4 4 11.22726  
## 5 5 11.22726

**Answer**

Yes, the 10-fold MSEouts remain the same after re-estimations

## 3b

Make sure your k\_fold\_mse() function can accept as many folds as there are rows (i.e., k=392).

### i

**Question**

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

**Answer**

Test dataset: 1 observation

Training dataset: 392 - 1 = 391 observations

### ii

Report the k-fold CV MSEout for all models using k=392.

mse\_full <- k\_fold\_mse(dataset=cars, k=392,lm, formula=cars\_full)   
mse\_reduced <- k\_fold\_mse(dataset=cars, k=392,lm, formula=cars\_reduced)  
mse\_full\_poly2 <- k\_fold\_mse(dataset=cars, k=392,lm, formula=cars\_full\_poly2)  
mse\_reduced\_poly2 <- k\_fold\_mse(dataset=cars, k=392,lm, formula=cars\_reduced\_poly2)  
mse\_reduced\_poly6 <- k\_fold\_mse(dataset=cars, k=392,lm, formula=cars\_reduced\_poly6)  
mse\_rpart\_reduced\_poly2 <- k\_fold\_mse(dataset=cars, k=392,rpart, formula=cars\_reduced\_poly2)  
mse\_rpart\_reduced\_poly6 <- k\_fold\_mse(dataset=cars, k=392,rpart, formula=cars\_reduced\_poly6)  
  
data.frame(Model = c("lm\_full", "lm\_reduced", "lm\_full\_poly2", "lm\_reduced\_poly2", "lm\_reduced\_poly6", "rt\_reduced\_poly2", "rt\_reduced\_poly6"),  
 MSEout = c(mse\_full, mse\_reduced, mse\_full\_poly2, mse\_reduced\_poly2,   
 mse\_reduced\_poly6, mse\_rpart\_reduced\_poly2, mse\_rpart\_reduced\_poly6)  
)

## Model MSEout  
## 1 lm\_full 11.293439  
## 2 lm\_reduced 11.380040  
## 3 lm\_full\_poly2 8.610385  
## 4 lm\_reduced\_poly2 8.787013  
## 5 lm\_reduced\_poly6 9.177932  
## 6 rt\_reduced\_poly2 13.303589  
## 7 rt\_reduced\_poly6 13.270311

### iii

**Question**

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? (show a few repetitions for any model and decide!)

set.seed(111555777)  
  
n\_repeats <- 5  
mse\_out\_repeats <- replicate(n\_repeats, {  
 k\_fold\_mse(cars, k = 392, model\_function = lm, formula = cars\_full)  
})  
  
# Display the results  
data.frame(Repetition = 1:n\_repeats, MSEout = mse\_out\_repeats)

## Repetition MSEout  
## 1 1 11.29344  
## 2 2 11.29344  
## 3 3 11.29344  
## 4 4 11.29344  
## 5 5 11.29344

**Answer**

Yes, the 392-fold MSEouts remain the same after re-estimations

### iv

**Question**

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?

# Calculate in-sample MSE for full and reduced models  
mse\_lm\_full\_in <- mse\_in(lm\_full)  
mse\_lm\_reduced\_in <- mse\_in(lm\_reduced)  
  
# Set seed for reproducibility  
set.seed(111555777)  
  
# Calculate out-of-sample MSE for full and reduced models  
mse\_lm\_full\_out <- k\_fold\_mse(cars, 392, lm, cars\_full)  
mse\_lm\_reduced\_out <- k\_fold\_mse(cars, 392, lm, cars\_reduced)  
  
# Display results  
data.frame(Model = c("Full Model", "Reduced Model"),  
 MSEin = c(mse\_lm\_full\_in, mse\_lm\_reduced\_in),  
 MSEout = c(mse\_lm\_full\_out, mse\_lm\_reduced\_out)  
)

## Model MSEin MSEout  
## 1 Full Model 10.68212 11.29344  
## 2 Reduced Model 10.97164 11.38004

**Answer**

The reduced model has a lower in-sample MSE compared to the full model, suggesting that removing collinear terms improved the fit of the model on the training data. The reduced model also has a lower out-of-sample MSE compared to the full model. This indicates that the reduced model performs better in predicting new data.

### v

**Question**

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?

# Calculate in-sample MSE for both models  
mse\_poly2\_reduced\_in <- mse\_in(lm\_poly2\_reduced)  
mse\_poly6\_reduced\_in <- mse\_in(lm\_poly6\_reduced)  
  
# Set seed for reproducibility  
set.seed(111555777)  
  
# Calculate out-of-sample MSE for both models using k-fold CV  
mse\_poly2\_reduced\_out <- k\_fold\_mse(cars, 392, lm, cars\_reduced\_poly2)  
mse\_poly6\_reduced\_out <- k\_fold\_mse(cars, 392, lm, cars\_reduced\_poly6)  
  
# Display results  
data.frame(Model = c("Reduced Quadratic", "Reduced 6th Order Polynomial"),  
 MSEin = c(mse\_poly2\_reduced\_in, mse\_poly6\_reduced\_in),  
 MSEout = c(mse\_poly2\_reduced\_out, mse\_poly6\_reduced\_out)  
)

## Model MSEin MSEout  
## 1 Reduced Quadratic 8.364546 8.787013  
## 2 Reduced 6th Order Polynomial 8.254377 9.177932

**Answer**

Adding more higher-order terms may slightly improve the fit to the training data (lower MSEin) but can hurt the model’s ability to generalize to new data (higher MSEout). The higher MSEout of the 6th order polynomial might stem from overfitting.