BACS - HW (Week 16)

Let's return yet again to the cars dataset we now understand quite well. Recall that it had several interesting issues such as *non-linearity* and *multicollinearity*. How do these issues affect prediction? We are also interested in *model complexity* and the difference in fit error versus prediction error. Let's **setup** what we need for this assignment (note: we will *not* use the cars_log dataset; we will return to the original, raw data for cars):

```
# 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)
```

A simple description of each **formula** described above and needed for the models below:

`cars_full`: The full formula with all IVs in our original dataset

`cars_reduced`: The reduced formula after stepwise-VIF to eliminate collinear terms

`cars_full_poly2`: The full formula with quadratic terms

`cars_reduced_poly2`: The reduced formula with quadratic terms

`cars_reduced_poly6`: The reduced formula with upto 6th degree higher-order terms

Here are seven **models** (formula + estimation/training method) you must create and test in this project:

i. `lm_full`: A full model (`cars_full`) using *linear regression*
ii. `lm_reduced`: A reduced model (`cars_reduced`) using *linear regression*
iii. `lm_poly2_full`: A full quadratic model (`cars_full_poly2`) using *linear regression*
iv. `lm_poly2_reduced`: A reduced quadratic model (`cars_reduced_poly2`) using *linear regression*
v. `lm_poly6_reduced`: A reduced 6th order polynomial (`cars_reduced_poly6`) using *linear regression*
vi. `rt_full`: A full model (`cars_full`) using a *regression tree*
vii. `rt_reduced`: A reduced model (`cars_reduced`) using a *regression tree*

(questions begin on next page)

Question 1) Compute and report the in-sample *fitting error* (MSE_{in}) of all the models described above. It might be easier to first write a function called `mse_in(...)` that returns the fitting error of a single model; you can then apply that function to each model (feel free to ask us for help!). We will discuss these results later.

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*:

a. Split the data into 70:30 for training:test (did you remember to shuffle the data earlier?)
b. Retrain the `lm_reduced` model on just the training dataset (call the new model: `trained_model`); Show the coefficients of the trained model.
c. Use the `trained_model` model to predict the `mpg` of the test dataset
   What is the in-sample mean-square fitting error (MSE_{in}) of the trained model?
   What is the out-of-sample mean-square prediction error (MSE_{out}) of the test dataset?
d. Show a data frame of the test set's actual `mpg values`, the predicted `mpg` values, and the difference of the two (ε_{out} = predictive error); *Just show us the first several rows of this dataframe.*

Question 3) Let's use k-fold cross validation (k-fold CV) to see how all these models perform predictively!

a. Write a function that performs k-fold cross-validation (see class notes and ask us online for hints!). Name your function `k_fold_mse(model, dataset, k=10, ...)` – it should return the MSE_{out} of the operation. Your function must accept a model, dataset and number of folds (k) but can also have whatever other parameters you wish.
   i. Use your `k_fold_mse` function to find and report the 10-fold CV MSE_{out} for all models.
   ii. For all the models, which is bigger — the fit error (MSE_{in}) or the prediction error (MSE_{out})? *(optional: why do you think that is?)*
   iii. Does the 10-fold MSE_{out} 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!)
b. Make sure your `k_fold_mse()` function can accept as many folds as there are rows (i.e., `k=392`).
   i. How many rows are in the training dataset and test dataset of each iteration of k-fold CV when k=392?
   ii. Report the k-fold CV MSE_{out} for all models using `k=392`.
   iii. When k=392, does the MSE_{out} 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!)
   iv. Looking at the fit error (MSE_{in}) and prediction error (MSE_{out}; 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? *(optional: if not, then when/why are analysts so scared of multicollinearity?)*
   v. Look at the fit error and prediction error (k=392) of the reduced quadratic versus 6^{th} order polynomial regressions — did adding more higher-order terms hurt the fit and/or predictions? *(optional: What does this imply? Does adding complex terms improve fit or prediction?)*