**Lab 3: Cross Validation**

**First Name**: yuan  **Last Name**: he **NetID**: yh689

**Lab 3 is due March 17 (Friday) by 4:30pm in the homework box at 2nd floor of Rhodes Hall. For Lab 3, submit your code and plots of the validation errors as well as your answers to the problems.**

In this lab, we will learn to use the Cross-Validation techniques to select the degrees of polynomials in regression. We will consider the validation set approach, LOOCV and k-fold CV, and apply them to the ToyotaCorolla data set. Some of the commands in this lab may take a while to run on your computer.

Download the ToyotaCorolla.csv data set from blackboard. Our goal is to use linear regression model with high order terms of different degrees to predict the price of a vehicle for resell in the market using the variable Age and KM, and find the appropriate order of the polynomial of

Before analyzing the data, do the same procedures to clean it as we did previously, i.e.

1. Delete all other columns except for Age 08 04 and KM, which we will be using.
2. Check if there is any missing data. If so, deal with it.
3. Check the classes of the variables.

# The Validation Set Approach

We pick half of the observations to be our training data.

> set.seed(1)

> train\_id = sample(nrow(corollas), nrow(corollas)/2)

Then we fit a linear regression model using only the data points in the training set.

> lm.fit = lm(Price~Age\_08\_04+KM, data=corollas, subset=train\_id)

Now we can calculate the MSE over the validation set. How to do that?

mean((test$y-predict(model\_Q, test))^2)

We next fit models with quadratic and cubic terms of the predictors, making used of the poly() function.

> lm.fit2 = lm(Price~poly(Age\_08\_04, 2)+poly(KM, 2), data=corollas, subset=train\_id)

> lm.fit3 = lm(Price~poly(Age\_08\_04, 3)+poly(KM, 3), data=corollas, subset=train\_id)

Calculate the MSE for quadratic and cubic regression models. What do you observe?



It decreases from 1st order to 2nd order and then increase.

# Leave-One-Out Cross-Validation

We then consider Leave-One-Out Cross-Validation (LOOCV). The LOOCV estimate can be automatically computed for any generalized linear model using the glm() and cv.glm() functions. Here we fit and validate models with degrees up to 5.

> library(boot)

> cv.error=rep(0,5)

> for (i in 1:5) {

+ glm.fit = glm(Price~poly(Age\_08\_04, i)+poly(KM,i), data = corollas)

+ cv.error[i] = cv.glm(corollas, glm.fit)$delta[1]

+ }

> cv.error

> plot(cv.error, type="b")

According to the LOOCV estimates, which model do you recommend? Why?

The model with 2nd order performs better than others. Since the error of this model is less than the error of others.

Check *p*-values in the summary for the cubic model. Describe your observations and compare them with your previous findings.

Next, we still use the cv.glm() function to perform *k*-fold CV, with *k* = 10. We will consider polynomials of degrees one to five, and then plot the corresponding cross-validation errors. > cv.error=rep(0,5)

> for (i in 1:5) {

+ glm.fit = glm(Price~poly(Age\_08\_04, i)+poly(KM,i), data = corollas)

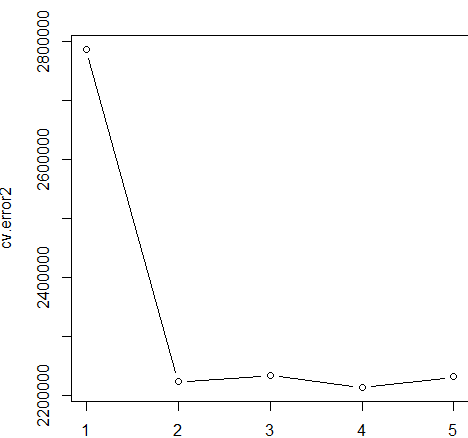
+ cv.error[i] = cv.glm(corollas, glm.fit, K=10)$delta[1]

+ }

> cv.error

> plot(cv.error, type="b")

Report your findings.



The error decreases at the beginning and swing slightly around a certain value.

Compare the running time of the three CV methods.

The LOOCV method is the slowest. And the first simple method is the fastest. The k-fold CV is in the middle.

Do you get different estimates for different runs of the validation approach? How about LOOCV? How about k-fold CV? Why?

The result of the LOOCV is the same for different runs. And the K-fold CV show different estimations since the LOOCV go through the whole set and K-fold CV split the dataset and the choose are random action. So the sets is different for different runs.

# Take Home Questions

**Part 1:**

If we use higher degree polynomials, how does it affect the bias, variance and test error of the model? Explain your answer.

The bias would decrease, since as the order increases, the model better fit the data.

The variance would increase, since as the degree increases, the model would become more fit for the test set, therefore, the model tends to overfit the certain test set (which do harm to the prediction of other sets).

The test error would decrease firstly and then increase after certain points. Because, at the beginning, the model underfit the whole set. With the help of test points, it better fit the set. Thus, the error would decrease. However, as the order increase, the model may be overfit the certain test set, which may derive from the reality.

The validation set, LOOCV and *k*-fold CV approaches are three different ways for estimating an unknown quantity. What is that quantity.

It is an estimator of how a certain trained model fit the real response function.

**Part 2:**

We will apply LOOCV and *k*-fold CV to logistic regression over synthetic data.

Firstly, run the following code to create a synthetic data set DF and make a plot.

> n = 1000

> x1 = runif(n)

> x2 = runif(n, -2, 1)

> z = (x1-0.2)\*(x1-0.5)\*(x1-0.9) \* 25 - x2\*(x2+1.2)\*(x2-0.8) + rnorm(n)/3

> y = as.integer(z>0)

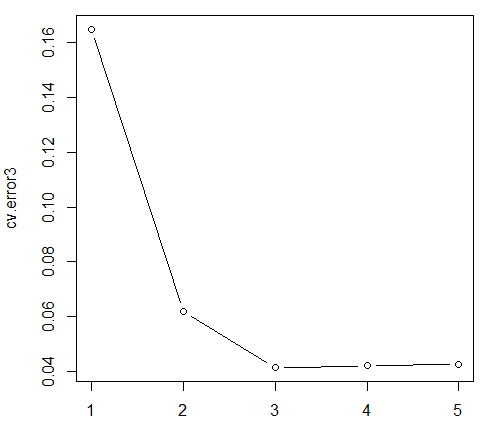
> plot(x1, x2, col=c("red", "blue")[y+1])

> DF = data.frame(x1,x2,y)

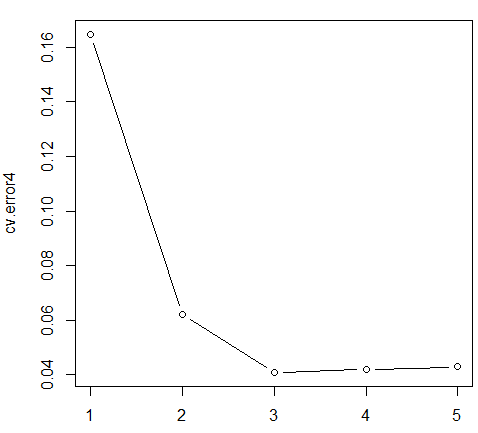
Then fit logistic regression models with different degrees and check the validation errors using LOOCV. (You need to write the R code yourself.)

Is the validation result consistent with how the data is created? Explain your answer.

Yes, it is consistent with how the data is created, since as the order of the function reaches 3, the error decreases dramatically and remain around a certain low value. Also the weight of the x1^3 and x2^3 is also high.



**Optional:** Use K-fold CV for logistic regression models with different degrees, and plot their validation errors. Compare the result with that from LOOCV.



The result of the K-fold is similar to the result of the LOOCV