Project 1: Predict the Housing Prices in Ames (PART II)

You are asked to implement Lasso using Coordinate Descent on the Ames housing data.

Download the dataset, "Ames_data.csv", from the Resouces page. The dataset has 2930 rows (i.e., houses) and 83 columns. Column 1 is "PID", the Parcel identification number, the last column is the response variable, "Sale_Price", and the remaining 81 columns are explanatory variables describing (almost) every aspect of residential homes.

The goal is to predict the final price of a home in Log scale.

Coordinate Descent for Lasso

- First write your own function or use the following function to solve the one-step Lasso for beta_j, see page 37 of [lec_W3_VariableSelection.pdf].

```
one_step_lasso = function(r, x, lam){
  xx = sum(x^2)
  xr = sum(r*x)
  b = (abs(xr) -lam/2)/xx
  b = sign(xr)*ifelse(b>0, b, 0)
  return(b)
}
```

- Then write a function to implement CD for n.iter steps.

```
mylasso = function(X, y, lam, n.iter = 50, standardize  = TRUE)
{
  # X: n-by-p design matrix without the intercept
  # y: n-by-1 response vector
  # lam: lambda value
  # n.iter: number of iterations
  # standardize: if True, center and scale X and y.



  # YOUR CODE
  # If standardize  = TRUE, center and scale X and Y
               record the corresponding means and sd


  # Initial values for residual and coefficient vector b
  b = rep(0, p)
  r = y

  for(step in 1:n.iter){
    for(j in 1:p){

      # YOUR CODE

      # 1) Update the residual vector
      # r <-- r + X[, j] * b[j]
      # r on the left: residual in blue on p37 of [lec_W3_VariableSelection.pdf]
      # r on the right: current residual

      # 2) Apply one_step_lasso to update beta_j
      # b[j] = one_step_lasso(r, X[, j], lam)

      # 3) Update the current residual vector
```

```
        # r <-- r - X[, j] * b[j]
      }
    }

    # YOUR CODE: scale back b and add intercept b0
    # For b0, check p13 of [lec_W3_VariableSelection.pdf].
    return(c(b0, b))
}
```

Note: in the script above, we run a fixed number of steps. n.iter = 50 is just for illustration. You may need to set a bigger number. You can change it to a "while" loop to stop when some covergence criterion is satisified.

- In practice, you need to add another loop outside to compute the Lasso solution for a range of lambda values.

  How to pick the lambda sequence? First think about how to find the largest lambda value (denoted by lambda_0), i.e., when lambda is bigger than lambda_0, all coefficients are zero, and once lambda is smaller than lambda_0, one variable will enter the model. Then, set a sequence of lambda from lambda_0 to a small number times lambda_0, equally spaced in log-scale.

  In "mylasso" function above, the initial value of beta is always set to be all zero. You can try to use the fitted beta from a larger lambda value as the initial value for the next (smaller) lambda.

  For this assignment, students are allowed to "cheat": you can use "glmnet" to figure out a proper range of lambda values, try them, and then fix a particular lambda value in your submission.

- Split the data into training and test with the test.id generated as follows. Fit your Lasso algorithm on the training data and pick the best lambda value by minimizing the RMSE of the logarithm of Sale_Price on the test data.

  ```
  test.id = seq(1, 2930, by=3)
  ```

- You can choose to apply your Lasso algorithm only on a subset of features. For example, we removed the following variables when trying our algorithm: Street, Utilities, Land_Slope, Condition_2, Roof_Matl, Heating, Pool_QC, Misc_Feature, Low_Qual_Fin_SF, Three_season_porch, Pool_Area, Misc_Val, Longitude and Latitude.

What you need to submit?

Before the deadline (Thursday, Oct 18, 11:30pm, Pacific Time) please submit your R/Python code
(.R or .py or zip).

How we evaluate your code?

Name your main file as **mymain_Lasso.R**. If you have multiple R files, upload the zip file. After unzipping your file, we will run the command "source(mymain_Lasso.R)" in a directory, in which there are only two files: train.csv and test.csv, where training and test are split using test.id described above.

- **train.csv**: subset the whole data "Ames_data.csv" with exactly the same 83 columns;
- **test.csv**: subset of whole data without the last column "Sale_Price", i.e., it has 82 columns.

After running your Rcode, we should see **ONE txt files** in the same directory named "mysubmission3.txt". **Submission File Format** and **Evaluation Procedure** are the same as Part I.

Your RMSE should be below 0.125.

The evaluation process for Python code is the same.
