Homework 6 Part 1

Instructions

Due: 5:00pm on Wednesday, November 3rd

1. Add your name between the quotation marks on the author line in the YAML above.
2. Compose your answer to each problem between the bars of red stars.
3. Commit your changes frequently.
4. Be sure to knit your .Rmd to a .pdf file.
5. Push both the .pdf and the .Rmd file to your repo on the class organization before the deadline.

Regression Competition III

We again return to the data set from Homework 3 to build a new multiple regression model using penalized regression techniques discussed recently in class.

As before, you will construct your model using a training data set with information on 66 variables recorded for 1808 houses. I've held back the data on 600 other houses which will serve as the test data set for assessing the predictive accuracy of your model.

You should record your answers in this .Rmd file. However, you are encouraged to use a separate .Rmd file for scratchwork. The assignment is divided into several **Components** to help organize your work. Put all work you want graded between the bars of red stars in the corresponding section.

Grading

This assignment restricts you to just using `glmnet` to build your model, so your overall score on this assignment will not be based on model accuracy (most models will have relatively similar accuracy). However, you will have optional opportunity at the end of this assignment to build a better model that synthesizes feature selection along with the other work you did on Homework 3, and I will run that model on test data as well and report the results.

The Data

The data set `house_train` can be found in the hw_3 repo and can be loaded by running the following code.

```r
house<-read_csv("house_train.csv")
```

Additionally, the `data_description.txt` file in the same repo gives a full description of the variables appearing in the data set.

There is one special column of note:

- `Sale_Price` is your response variable and should not be included as a predictor.

Components

Data Exploration

Create initial training and validation sets from the `house` data.

1. How many rows are in the training set? How many are in the validation set?
2. How many columns are in the training set? How many are in the validation set?
3. Create a model matrix for both training and validation data sets. How many columns are in the model matrix for the training set? How many for the validation set?
4. Explain why your answer to the previous part will pose a problem when using `glmnet` to make predictions on the validation model matrix using a model built on the training model matrix.

```r
set.seed(10)
library(rsample)
```

```
## Warning: package 'rsample' was built under R version 3.6.2
```

```r
house_split <- initial_split(house)
house_train <- training(house_split)
house_test <- testing(house_split)
```

1. The training set contains 1356 rows, while the validation set contains 452 rows.
2. The training set contains 66 columns, as does the valdidation set.
3. This particular training model matrix contains 196 columns, while the particular validation model matrix contains 185 columns (answers will vary depending on the random split, as not all levels of categorical predictors will be present in test and training sets).

```r
house_train_mm <- model.matrix(Sale_Price ~., data = house_train)[,-1]
ncol(house_train_mm)
```

```
## [1] 196
```

```r
house_test_mm <- model.matrix(Sale_Price ~., data = house_test)[,-1]
ncol(house_test_mm)
```

```
## [1] 185
```

4. The model.matrix function converts categorical variables to quantitative dummy variables. However, if a level of the categorical variable is present in the training set but not test set (or vice versa), some dummy variables may not be present in either the training or test set. But this is problematic for using `glmnet` to make predictions, since `glmnet` requires every predictor in the test set to be present in the training set.

Data Partitioning

In this section, create a training / validation split that solves the problem identified in the previous part.

1. Use `model.matrix` to create a model matrix for the full house data set. In order to leave the response variable in the matrix, do not include it on the left side of `~` in the `model.matrix` function.
2. Convert the model matrix to a data frame using `as.data.frame()`, apply `initial_split` function from the `rsample` package, and then create training and validation data frames.
3. Use `model.matrix` once again to create separate model matrices for the training and validation sets, along with response vectors for each set.
4. Verify that both training and validation model matrices have the same number of columns.

---

1. Below, we create the model matrix for the full house data and include the response variable. However, we remove the left-most column which consists of just 1's (and which corresponds to the intercept of the linear model).

```
house_full_mm <- model.matrix(~., data = house)[,-1]
```

2.

```
set.seed(10)
house_full_df <- as.data.frame(house_full_mm)
house_full_split <- initial_split(house_full_df)
house_train <- training(house_full_split)
house_test <- testing(house_full_split)
```

3. Below, we create model matrices for predictors, as well as vector for responses.

```
house_train_mm <- model.matrix(Sale_Price~., data = house_train)[,-1]
y_train <- house_train$Sale_Price

house_test_mm <-model.matrix(Sale_Price~., data = house_test)[,-1]
y_test <- house_test$Sale_Price
```

4. Both training and test model matrices have 197 columns.

```
ncol(house_train_mm)
```

```
## [1] 197
```

```
ncol(house_test_mm)
```

```
## [1] 197
```

Model Building

In this section, use `glmnet` to create both ridge regression and lasso models on the house training data. Do not perform any data processing, or include any transformations or interaction terms.

Based on coefficient path plots, it seems that lambda appropriate values of lambda range from 10^{-2} to 10^{8}.

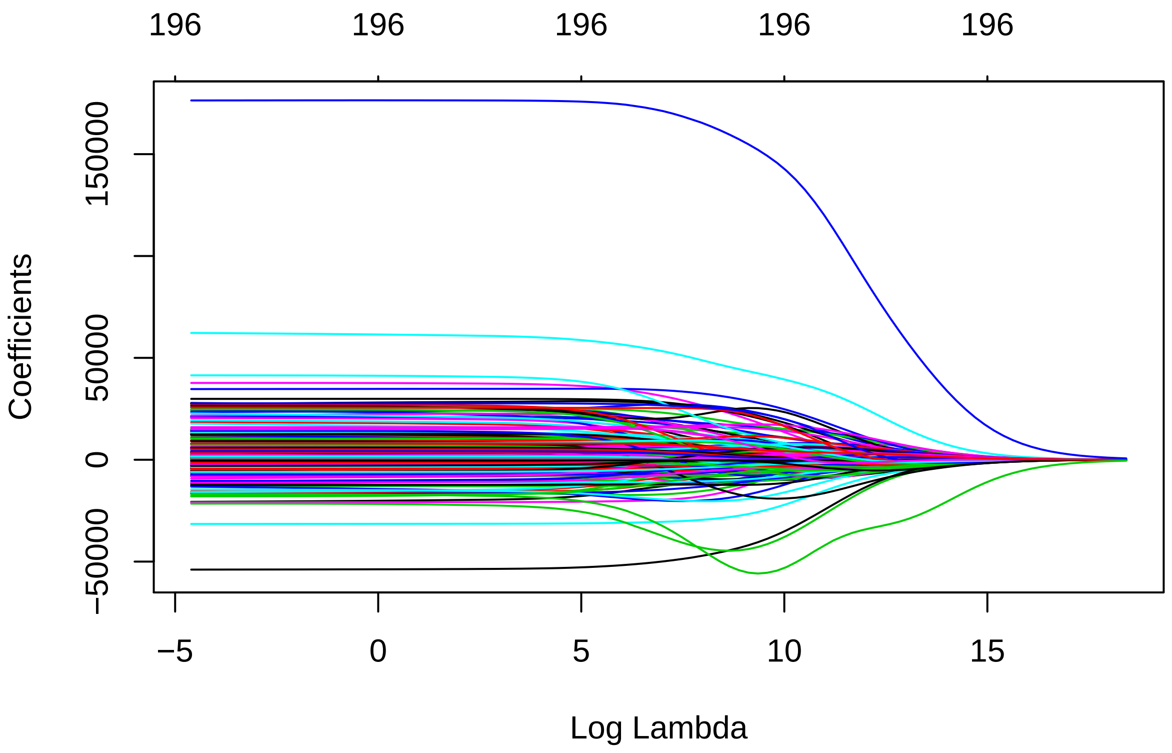
```
library(glmnet)
```

```
## Warning: package 'glmnet' was built under R version 3.6.2

## Loading required package: Matrix

## Warning: package 'Matrix' was built under R version 3.6.2

##
## Attaching package: 'Matrix'

## The following objects are masked from 'package:tidyr':
##
##     expand, pack, unpack

## Loaded glmnet 4.1-1
```

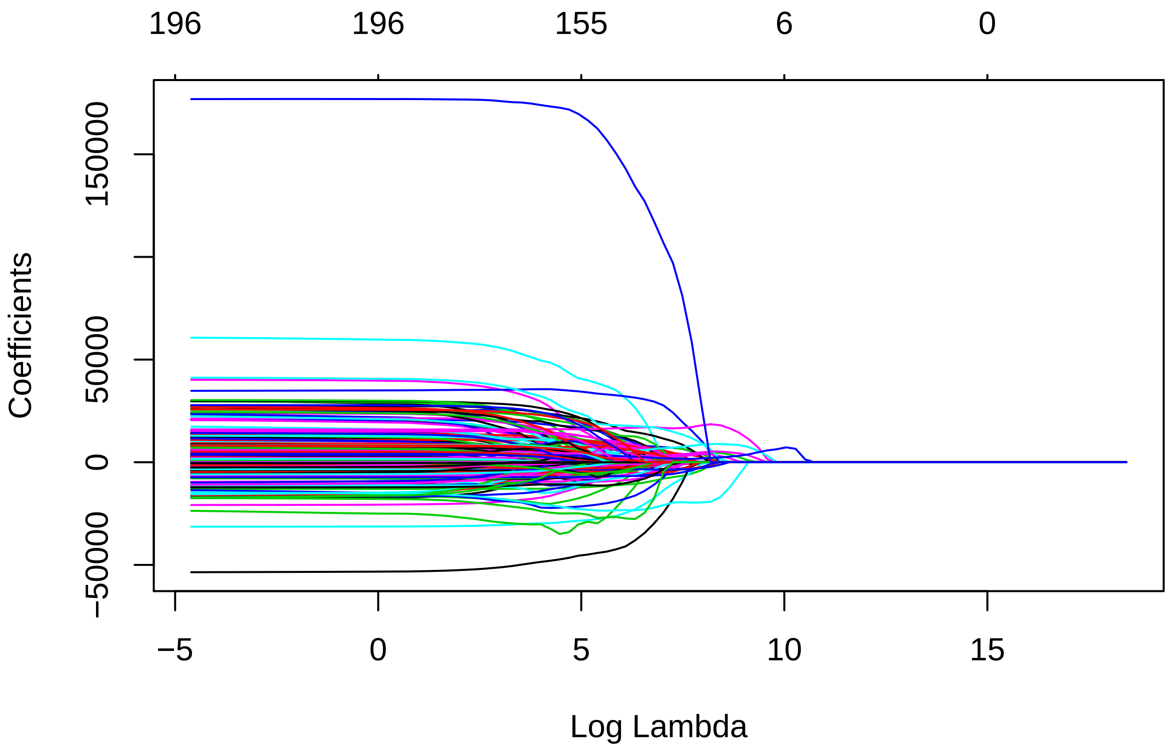
```
grid <- 10^seq(-2, 8, length = 100)
ridge_mod<-glmnet(x = house_train_mm, y = y_train, alpha = 0, lambda = grid )
lasso_mod<-glmnet(x = house_train_mm, y = y_train, alpha = 1, lambda = grid  )
```

Although not strictly necessary for this question, we create plots of the coefficient paths for both models.

```r
plot(ridge_mod, xvar = "lambda")
```



```r
plot(lasso_mod, xvar = "lambda")
```



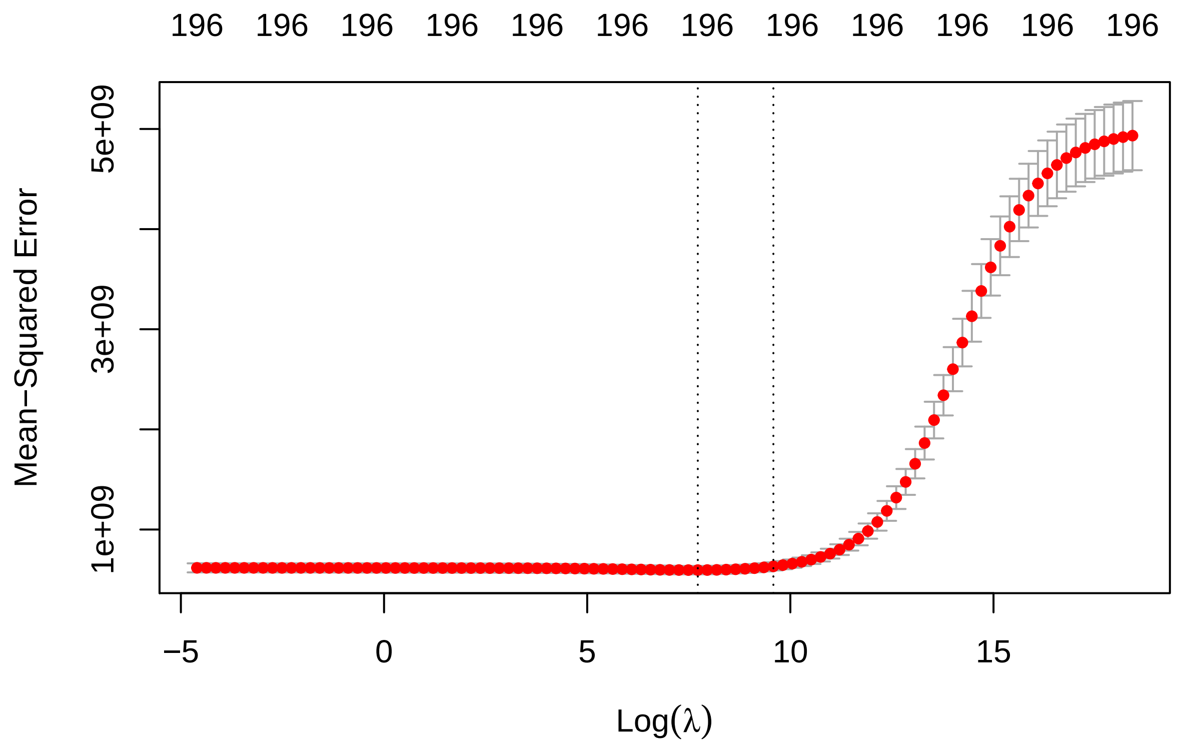
Model Diagnostics

In this section, use cross validation to assess the relationship between λ and model accuracy, for both ridge regression and LASSO. Display these metrics both graphically, and then explicitly compute optimal values. Which value of λ appears to produce the optimal model for each model type? How many variables are present in the optimal ridge regression model? How many are present in the optimal lasso model?

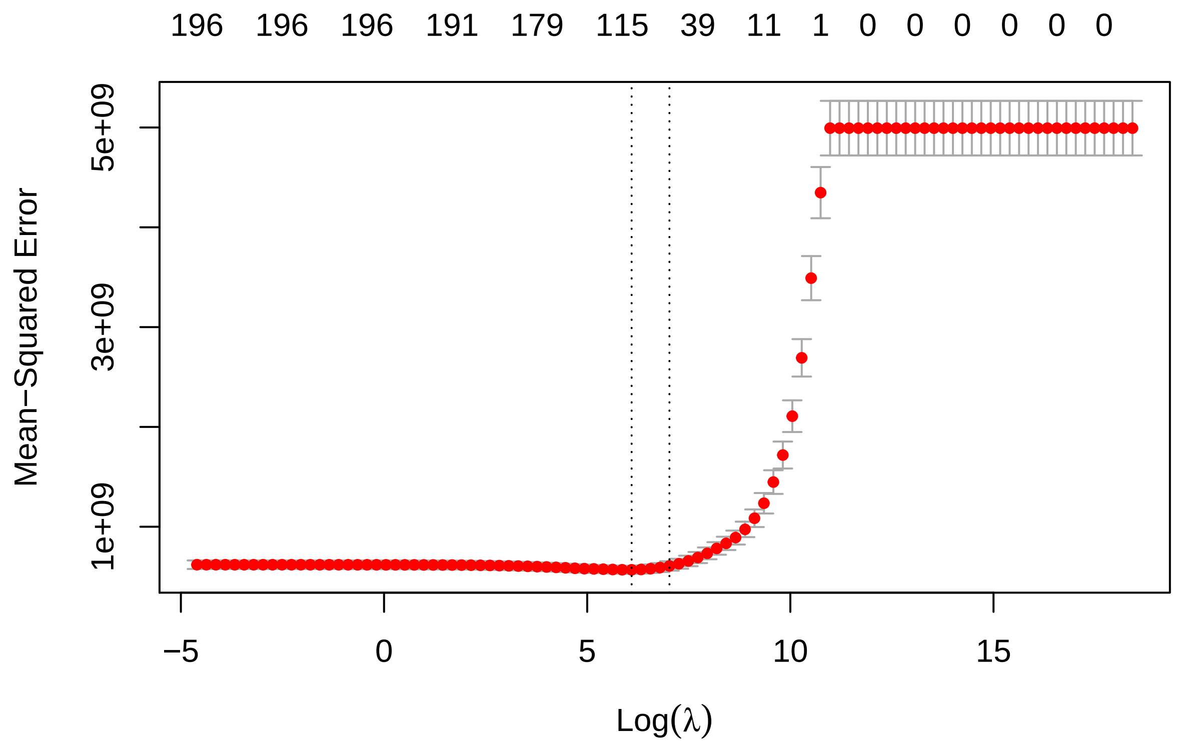
```r
set.seed(111)
ridge_cv <- cv.glmnet(x = house_train_mm, y = y_train, alpha = 0, lambda = grid )
lasso_cv <- cv.glmnet(x = house_train_mm, y = y_train, alpha = 1, lambda = grid  )
```

The plots below show cross-validated MSE for ridge regression and lasso as a function of the tuning parameter $\log \lambda$.

```r
plot(ridge_cv)
```



```r
plot(lasso_cv)
```



Test MSE is minimized for ridge regression at 2257.0197196 and for lasso at 443.0621458

```r
s_rr <- which(ridge_mod$lambda == ridge_cv$lambda.min)
n_nonzero_rr <- sum(coef(ridge_mod)[,s_rr]!=0)
n_nonzero_rr
```

```
## [1] 197
```

```r
s_lasso <- which(lasso_mod$lambda == lasso_cv$lambda.min)
n_nonzero_lasso <- sum(coef(lasso_mod)[,s_rr]!=0)
n_nonzero_lasso
```

```
## [1] 40
```

In the optimial ridge regression model, there are 197 non-zero coefficients, while the optimal lasso model, there are 40 non-zero coefficients. This is not surprising, since LASSO performs feature selection, while ridge regression does not.

Model Assessment

Choose 1 Ridge Regression and 1 LASSO model based on the information in the previous component, and then compute rMSE for each model on the *validation* set. Additionally, compare to the rMSE for the **full** model, as well as for the model you choose using feature selection in HW 5.

Which model performed best?

```r
preds_rr <- predict(ridge_mod, s = ridge_cv$lambda.min, newx = house_test_mm)
preds_lasso <- predict(lasso_mod, s = lasso_cv$lambda.min, newx = house_test_mm)
```

Here, we use the `rmse` function from the `yardstick` package to efficiently compute rmse for all four models. We find that the lasso model has lower rmse than the ridge regression model, but that both outperform the full model, as well as the best subset model (which doesn't use **any** categorical variables).

```
full_mod <- lm(Sale_Price ~., data = house_train)
full_preds <- predict(full_mod, house_test)
```

```
## Warning in predict.lm(full_mod, house_test): prediction from a rank-deficient
## fit may be misleading
```

```
best_mod <- lm(Sale_Price ~ Lot_Frontage + Lot_Area + Year_Built + Year_Remod_Add +
    Mas_Vnr_Area + BsmtFin_SF_2 + Bsmt_Unf_SF + Total_Bsmt_SF +
    Gr_Liv_Area + Bedroom_AbvGr + Kitchen_AbvGr + Garage_Area +
    Wood_Deck_SF + Screen_Porch + Latitude, data = house_train)
best_preds <- predict(best_mod, house_test)

results <- data.frame(
  ridge_regression = as.numeric(preds_rr),
  lasso = as.numeric(preds_lasso),
  full = as.numeric(full_preds),
  best_subset = as.numeric(best_preds),
  observed = y_test ) %>%
  rownames_to_column(var = "id") %>%
  pivot_longer(!c("id", "observed"), names_to = "model", values_to = "prediction")

library(yardstick)
```

```
## Warning: package 'yardstick' was built under R version 3.6.2

## For binary classification, the first factor level is assumed to be the event.
## Use the argument `event_level = "second"` to alter this as needed.

##
## Attaching package: 'yardstick'

## The following object is masked from 'package:readr':
##
##     spec
```

```
results %>% group_by(model) %>%
  rmse(truth = observed, estimate = prediction) %>%
  arrange(.estimate)
```

```
## # A tibble: 4 x 4
##   model            .metric .estimator .estimate
##   <chr>            <chr>   <chr>          <dbl>
## 1 lasso            rmse    standard      22201.
## 2 ridge_regression rmse    standard      23627.
## 3 full             rmse    standard      24232.
## 4 best_subset      rmse    standard      25831.
```

Model Interpretation

What are some advantages of Ridge Regression and LASSO in model building, compared to either the full model or feature selection using `regsubsets`? Are there any disadvantages to using Ridge Regression or LASSO?

Compared to the full model, both ridge regression and lasso have lower variance at the cost of slightly higher bias, and often will have a lower rMSE as a result. Coefficient estimates in both ridge regression and lasso tend to be smaller than in the full model. Additionally, lasso performs feature selection in addition to coefficient shrinkage.

Compared to the other method of feature selection (`regsubsets`), LASSO is capable of handling both quantitative and numeric predictors, is significantly faster to compute (and often to code!), and ultimately produces models that have lower rMSE.

One disadvantage of ridge regression and lasso relative to ordinary least squares regression is that we lose the ability to make inferential statements about model coefficients (this is also a problem of `regsubsets`)

Optional Model Enhancement

Optionally, you may use this space to build a model that improves on the one from Homework 3 by incorporating penalized regression, along with transformations, interaction terms, and feature engineering. I will assess your optional model on test data, alongside the model you made in the previous part. Use the following function to create **predictions** for your model. **NOTE that the number 3 should immediately follow your last name**

```
# This function will create a vector of predictions for the Sale_Price of houses.
# This function should take only test_data as an argument.
# The function must be self-contained, so needs to include all model building and data processin steps

FirstName_LastName3_predictions <- function(test_data){
  library(tidyverse)     ## Load whatever packages you need
  ## Create your model here
  my_preds <- predict(model, test_data)   ## Make predictions based on your model.
  my_preds<- my_preds*1 ## Transform your model predictions back to the original units for Sale_Price,
  my_preds      ##return your predictions as output
}
```

To verify that your function is working as desired, open a new .Rmd file, load the `house_train` data, copy the code for your function over to the new .Rmd, and then run the following code:

```
library(dplyr)
B <- sample_n(house, size = 100) # This creates a test dry
FirstName_LastName3_predictions(B) # Change to your First and Last Name
```
