STAT 457 Project 1 — Predicting Trip Duration

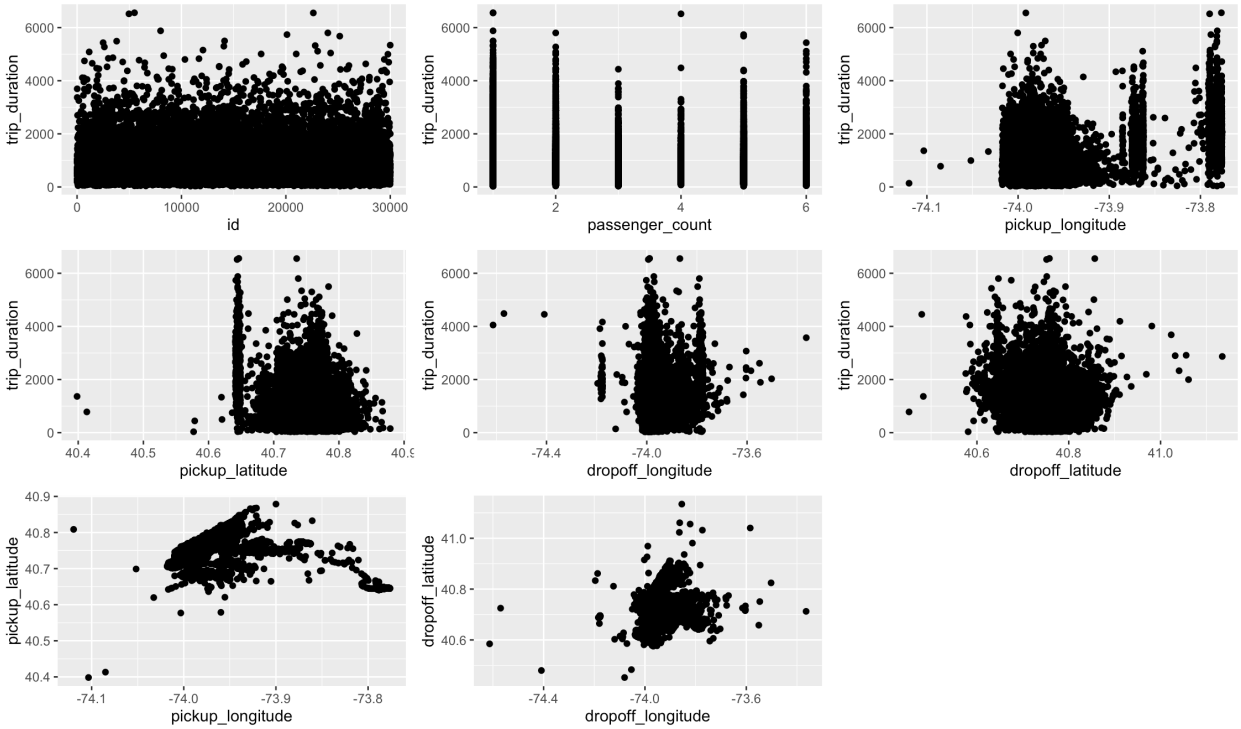
Bofan Sun
20108143
17bs48@queensu.ca

Introduction

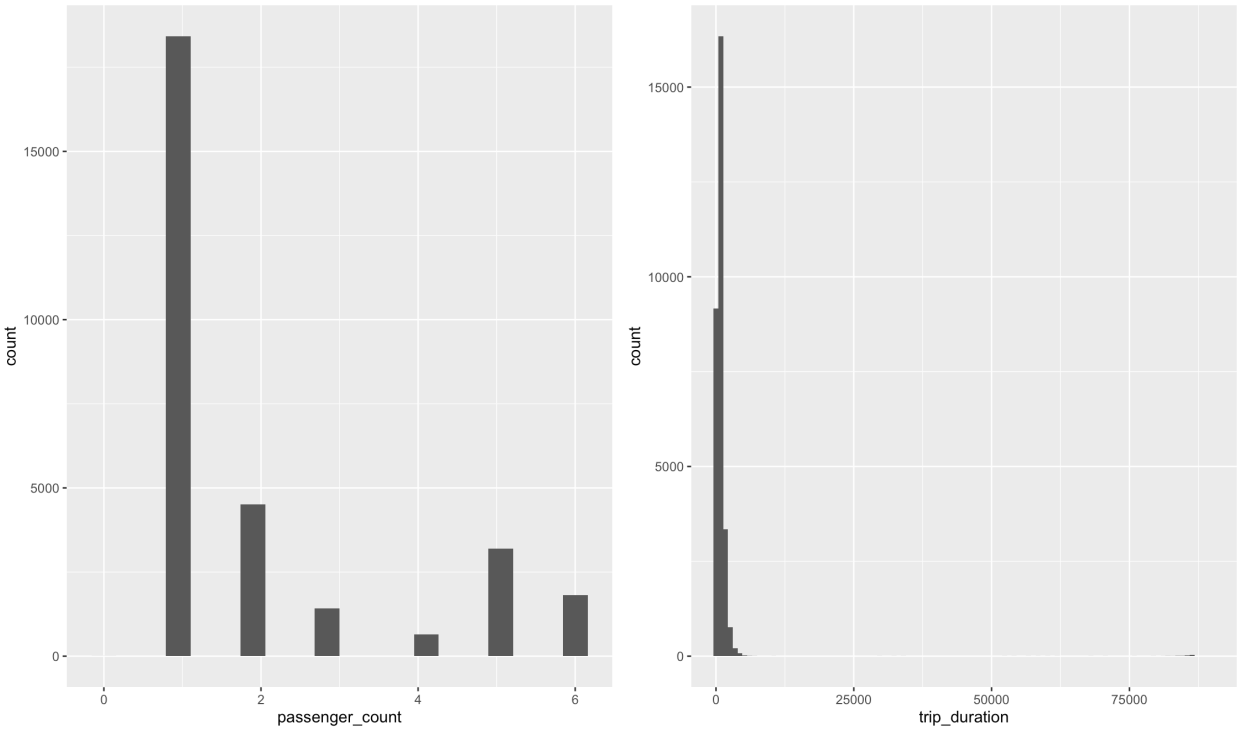
The goal of this project is to analyze a subset of data derived from the "New York City Taxi Trip Duration" competition on Kaggle, and predict the trip duration of each trip in the test dataset based on the attributes provided in the train dataset by building models. I start by using glimpse() and summary() to have a brief summary of the dataset.

```
> glimpse(train_dat)
Rows: 30,000
Columns: 9
$ id                <int> 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23, …
$ pickup_date       <chr> "2016-01-07", "2016-01-27", "2016-01-31", "2016-01-19", "2016-01-25", "2016-01-05", "2…
$ pickup_time       <chr> "19:32:15", "08:07:32", "13:52:55", "08:00:19", "23:32:14", "09:53:39", "15:26:03", "2…
$ passenger_count   <int> 1, 1, 1, 3, 1, 1, 1, 2, 1, 3, 2, 1, 1, 1, 1, 1, 1, 2, 1, 3, 1, 2, 1, 1, 1, 3, 1, 2, 1,…
$ pickup_longitude  <dbl> -73.98639, -73.95604, -73.97600, -73.96012, -73.98743, -73.98118, -73.97827, -73.97845…
$ pickup_latitude   <dbl> 40.75661, 40.76761, 40.75114, 40.78195, 40.76014, 40.72090, 40.74566, 40.74131, 40.749…
$ dropoff_longitude <dbl> -73.99979, -73.96820, -74.00185, -73.97197, -73.99098, -73.97739, -73.96526, -73.97649…
$ dropoff_latitude  <dbl> 40.76163, 40.78669, 40.73523, 40.75504, 40.74486, 40.72609, 40.75829, 40.73246, 40.756…
$ trip_duration     <int> 520, 989, 657, 1035, 621, 204, 734, 299, 235, 966, 587, 605, 918, 3695, 616, 414, 3370…
> summary(train_dat)
       id         pickup_date        pickup_time        passenger_count pickup_longitude pickup_latitude
 Min.   :    0   Length:30000       Length:30000       Min.   :0.000   Min.   :-74.20   Min.   :40.40
 1st Qu.: 7500   Class :character   Class :character   1st Qu.:1.000   1st Qu.:-73.99   1st Qu.:40.74
 Median :15000   Mode  :character   Mode  :character   Median :1.000   Median :-73.98   Median :40.75
 Mean   :15000                                         Mean   :2.037   Mean   :-73.97   Mean   :40.75
 3rd Qu.:22499                                         3rd Qu.:2.000   3rd Qu.:-73.97   3rd Qu.:40.77
 Max.   :29999                                         Max.   :6.000   Max.   :-73.55   Max.   :41.69
 dropoff_longitude dropoff_latitude trip_duration
 Min.   :-74.61    Min.   :40.45    Min.   :    2.0
 1st Qu.:-73.99    1st Qu.:40.74    1st Qu.:  386.0
 Median :-73.98    Median :40.75    Median :  634.0
 Mean   :-73.97    Mean   :40.75    Mean   :  995.5
 3rd Qu.:-73.96    3rd Qu.:40.77    3rd Qu.: 1019.0
 Max.   :-72.67    Max.   :41.69    Max.   :86356.0
>
```

Then, in order to closely observe the dataset, I did data visualization on the train dataset by plotting some graphs on different variables. First, I plot each variables (except pickup_date and pickup_time in character type) versus the response trip_duration, as well as the graphs for pickup and dropoff spots with longitude as x-axis and latitude as y-axis to observe their relationships. This is what I got:



We can see that there's no obvious relationship between the variable "id" and "trip_duration". We may consider drop this variable from later analysis. By the plot between latitude and longitude, it seems that most of the pickup spots and dropoff spots are very close. They may all pickup or dropoff within one area. Next, I also draw histograms of passenger_count and trip_duration. This is what I got:



From the first histogram we can see that most of the trips have 1 passenger, and there are 1 trip with 0 passenger. This looks strange, and we may consider to remove this from our dataset. From the second histogram of trip_duration, it is obvious that most of the trips are shorter than 10,000 seconds, and there are only a few trips longer than 10,000 seconds.

Exploratory Data Analysis

Next, I did some data transformations to the dataset. First I extract hour from pickup_time, pickup date and week of day from pickup_date. I also added a new variable called weekend which is 1 if the trip happened during the weekend, and 0 otherwise. Then I use disHaversine() function from "geosphere" package to calculate the shortest distance (in meters) between pickup point and dropoff point formed by pickup and dropoff longitude and latitude matrices. Then, I removed outliers with passenger_count is 0 from the training dataset.

```
#Modeling
#remove outliners
train_dat <- train_dat %>%
  filter(passenger_count >= 1)
```

Then, I prepared training dataset and testing dataset for the following modelling by removing some unnecessary variables, and split the original train dataset into train and test by about 30% as test and rest of them as train. The following is the dataset used in the first model which we will discuss later. By fitting different datasets with different variables to the model, the dataset with id, pickup_dat, pickup_time, pickup_longitude, pickup_latitude, dropoff_longitude and dropoff_latitude removed performs best.

```
#dataset for model 1
drops <- c('id', 'pickup_date', 'pickup_time', 'pickup_longitude',
           'pickup_latitude', 'dropoff_longitude', 'dropoff_latitude')
train <- train_dat[, !names(train_dat) %in% drops]
test <- test_dat[, !names(test_dat) %in% drops]
trip_Duration <- train$trip_duration

n = nrow(train)
index = sample(1:n, round(0.3*n))
test_1 = train[index,]
train_1 = train[-index,]
```

The following is the dataset used in the second and third model. By fitting different datasets with different variables to the model, the dataset with id, pickup_date, and pickup_time removed performs best which we will discuss later.

```r
#dataset for model 2 and 3
drops <- c('id', 'pickup_date', 'pickup_time')
train1 <- train_dat[, !names(train_dat) %in% drops]
test1 <- test_dat[, !names(test_dat) %in% drops]
trip_Duration <- train$trip_duration

n = nrow(train_dat)
index = sample(1:n, round(0.3*n))
test_2 = train[index,]
train_2 = train[-index,]
```

Prediction Models and Kaggle Scores

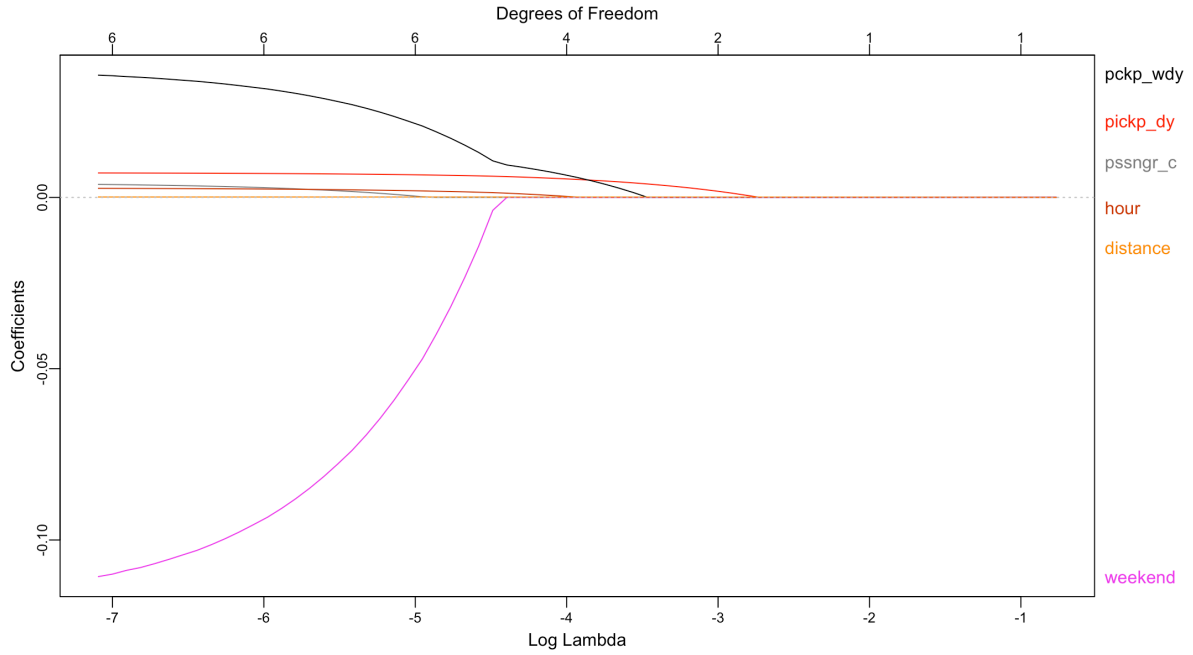
I first defined my own Root Mean Squared Logarithmic Error function for evaluating my model performance later.

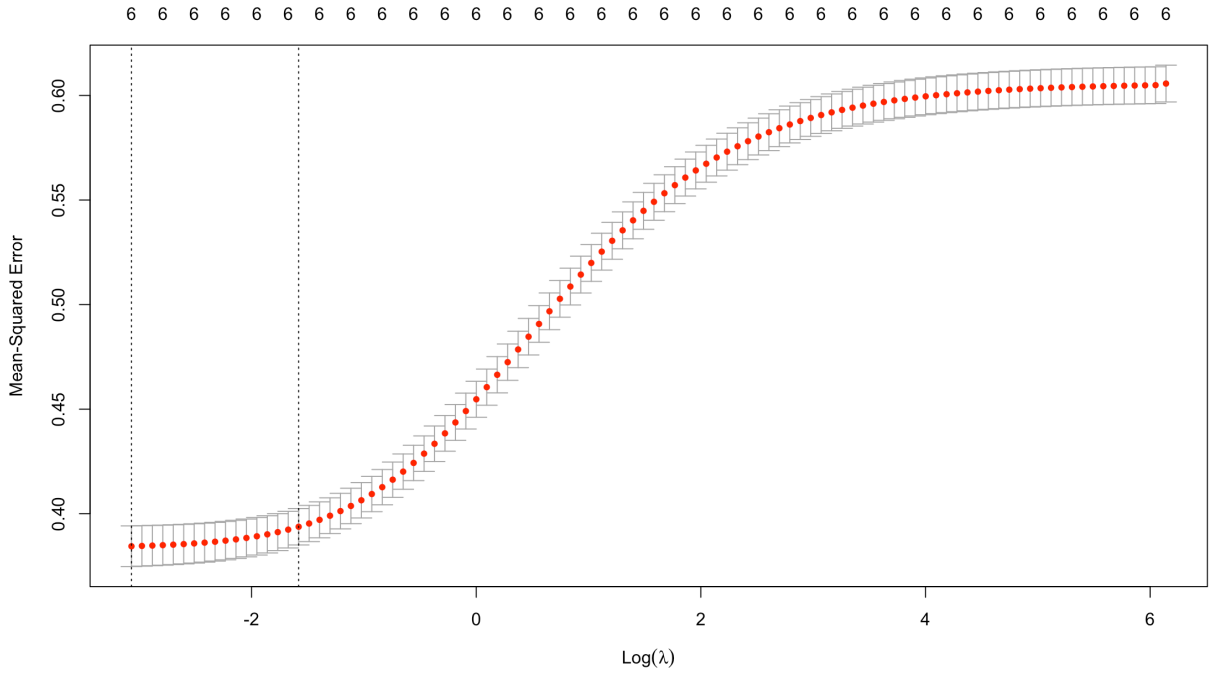
```r
rmsle <- function(pi, ai){
  index = which(ai!=0)
  term = (log(pi[index]+1) - log(ai[index]+1))^2
  result = sqrt(mean(term))
  return(result)
}
```

First Model —— Combination of Lasso regression and Ridge regression

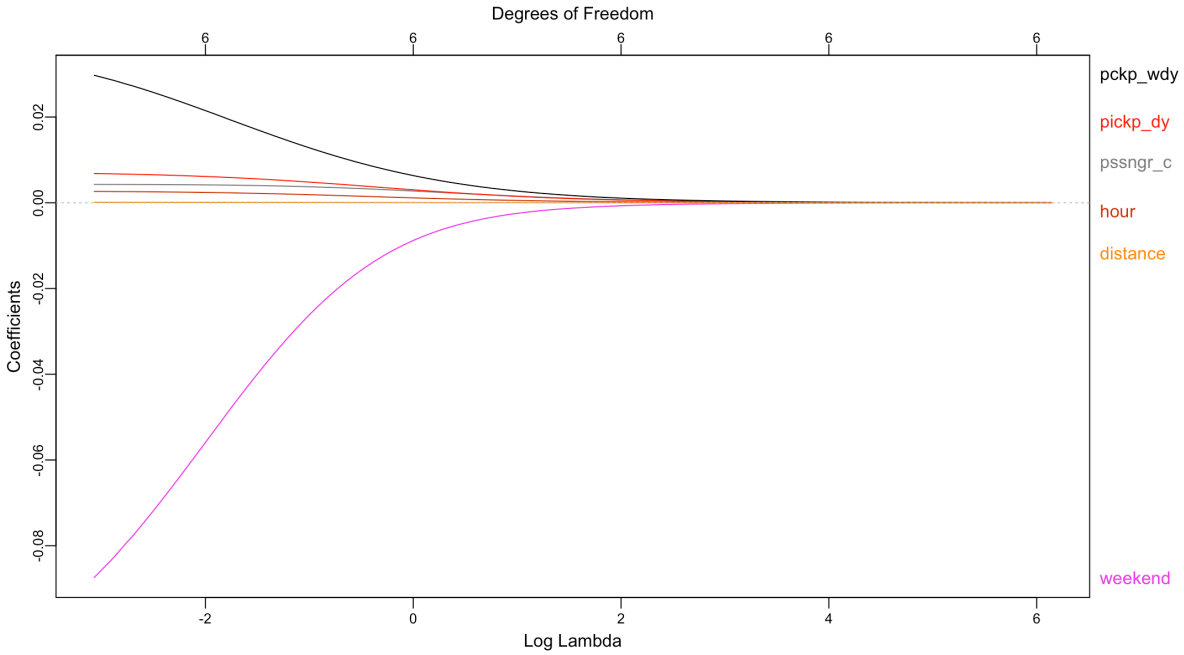
First, I fit the dataset prepared in the last step into lasso regression, and plot the solution path for lambda. Then I use cross validation to select lambda, and plot the graph. Next, I evaluate prediction performance with lambda chosen by one-standard deviation rule and smallest cross validation error using the error function I wrote above. Then do the same thing for ridge regression. Since we are using root mean squared logarithmic error as evaluation metric, in order to get a better performance, I perform logarithm with base e to trip_duration, and then perform exponential function to the prediction value when evaluating the error.

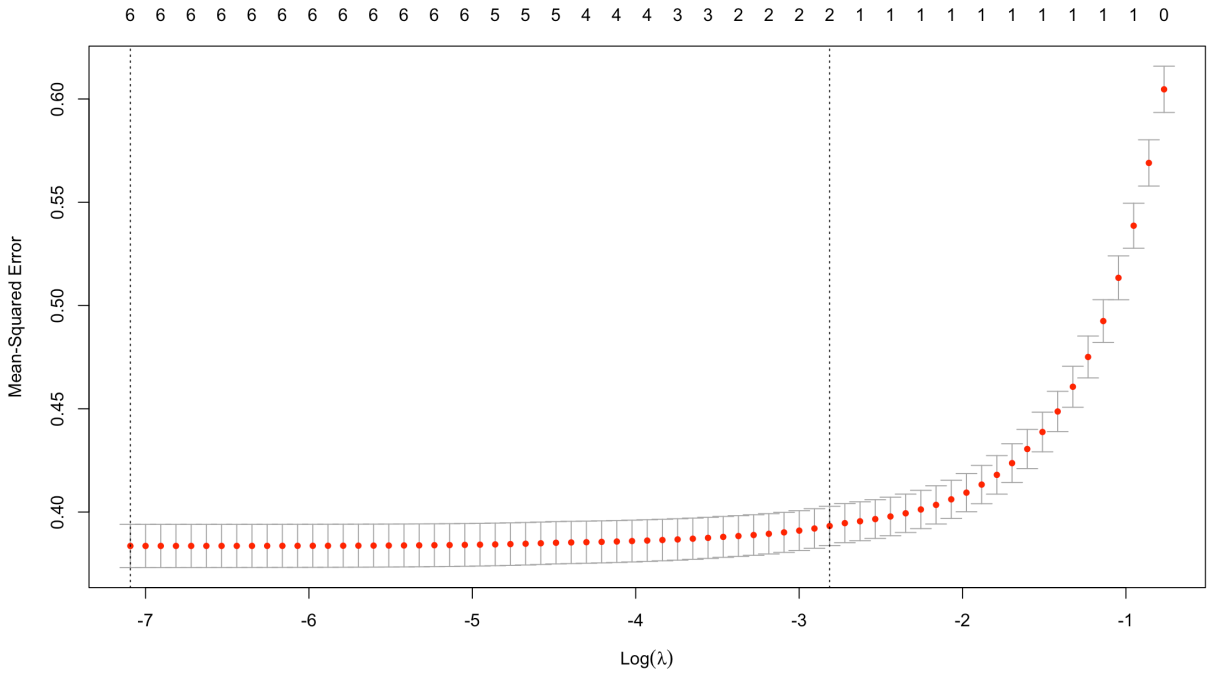
```r
#Model1: ridge regression + lasso regression
#Lasso regression
mylasso <- glmnet(as.matrix(train_1[,-2]), log(train_1$trip_duration), alpha = 1)
plot_glmnet(mylasso, label = TRUE, xvar = "lambda")
lasso.cv.out <- cv.glmnet(as.matrix(train_1[,-2]), log(train_1$trip_duration), alpha = 1)
plot(lasso.cv.out)
#lambda one-standard deviation
Dtest.1se <- predict(lasso.cv.out, s = lasso.cv.out$lambda.1se, newx=as.matrix(test_1[,-2]))
rmsle(exp(Dtest.1se), trip_Duration[index])
#[1] 0.6316224
#lambda smallest cross validation error
lass.min <- predict(lasso.cv.out, s = lasso.cv.out$lambda.min, newx = as.matrix(test_1[,-2]))
rmsle(exp(lass.min), trip_Duration[index])
#[1] 0.6229381
```






```r
#ridge regression
myridge <- glmnet(as.matrix(train_1[-2]), log(train_1$trip_duration), alpha = 0)
plot_glmnet(myridge, label = TRUE, xvar = "lambda")
ridge.cv.out <- cv.glmnet(as.matrix(train_1[-2]), log(train_1$trip_duration), alpha = 0)
plot(ridge.cv.out)
#lambda one-standard deviation
ridge.1se <- predict(ridge.cv.out, s = ridge.cv.out$lambda.1se, newx=as.matrix(test_1[-2]))
rmsle(exp(ridge.1se), trip_Duration[index])
#[1] 0.6342622
#lambda smallest cross validation error
ridge.min <- predict(ridge.cv.out, s = ridge.cv.out$lambda.min, newx = as.matrix(test_1[-2]))
rmsle(exp(ridge.min), trip_Duration[index])
#[1] 0.6244022
```





Then I combined the two regression models together with different weights. After trying different combinations of weights, I got the one with best result. Then I generate a csv file and submit it to Kaggle. My final score for this model is 0.59186.

```r
#combine lasso and ridge regression together
a=0.1
b=0.4
c=0.1
d=0.4
combine <- Dtest.1se*a+lass.min*b+ridge.1se*c+ridge.min*d
rmsle(exp(combine), trip_Duration[index])
#[1] 0.6244389
lasso0 <- predict(lasso.cv.out, s = lasso.cv.out$lambda.1se, newx = as.matrix(test))
lasso1<- predict(lasso.cv.out, s = lasso.cv.out$lambda.min, newx = as.matrix(test))
ridge0 <- predict(ridge.cv.out, s = ridge.cv.out$lambda.1se, newx=as.matrix(test))
ridge1 <- predict(ridge.cv.out, s = ridge.cv.out$lambda.min, newx=as.matrix(test))
combine0 <- lasso0*a+lasso1*b+ridge0*c+ridge1*d
out_Dat = data.frame(id = test_dat$id, trip_duration = exp(combine0))
colnames(out_Dat) = c('ID', 'trip_duration')
write.csv(out_Dat, "Desktop/w22proj1/W22P1_submission.csv", row.names = FALSE)
```

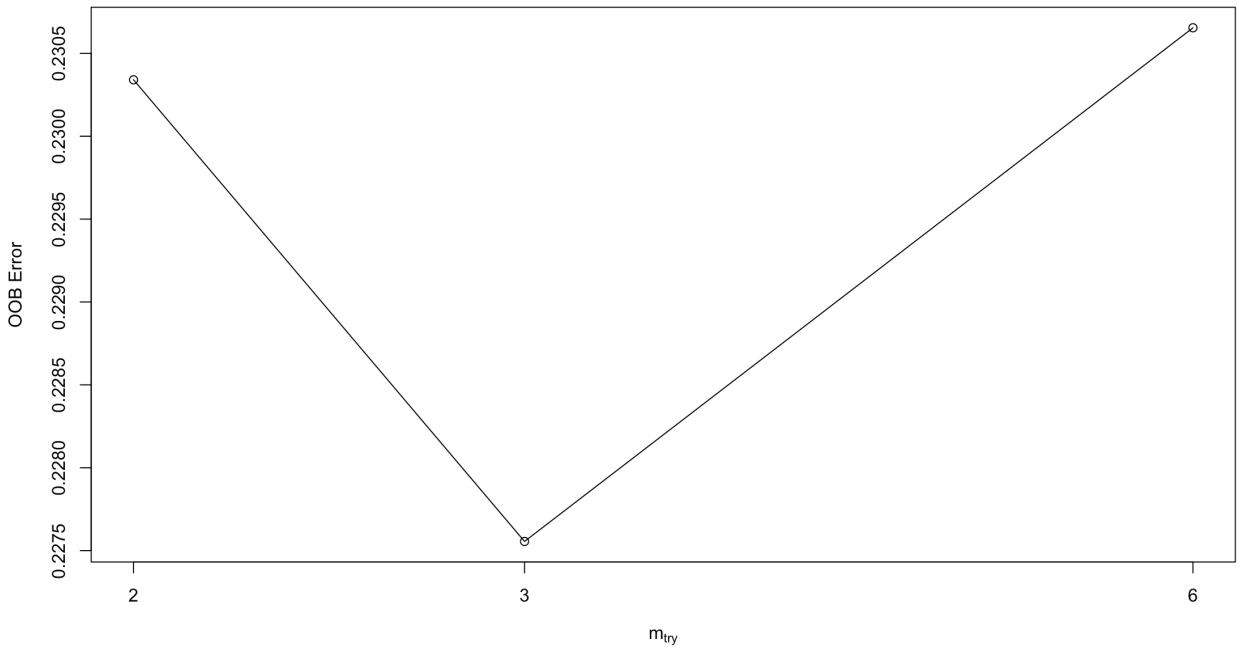
W22P1_submission.csv 0.59186

a few seconds ago by Monnnnasun

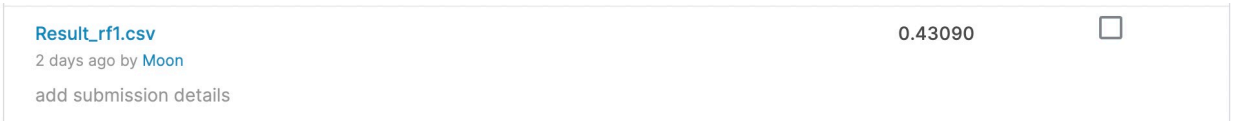
lasso+ridge

Second Model —— Random Forest

The second model is random forest. First, I used tuneRF() function to select the best tuning parameter mtry. By trying different stepFactors (1.5, 2, or 3), we got the best mtry as 3 for most of the times, and 4 some times.



Then use ranger() function from ranger package to implement random forest, with parameter num.trees = 500, and mtry = 3. Since we are using root mean squared logarithmic error as evaluation metric, in order to get a better performance, I perform logarithm with base e to trip_duration, and then perform exponential function to the prediction value when evaluating the error. Then applying random forest to the test file, and generate a csv file and submit it to Kaggle. I got my best result 0.43090.



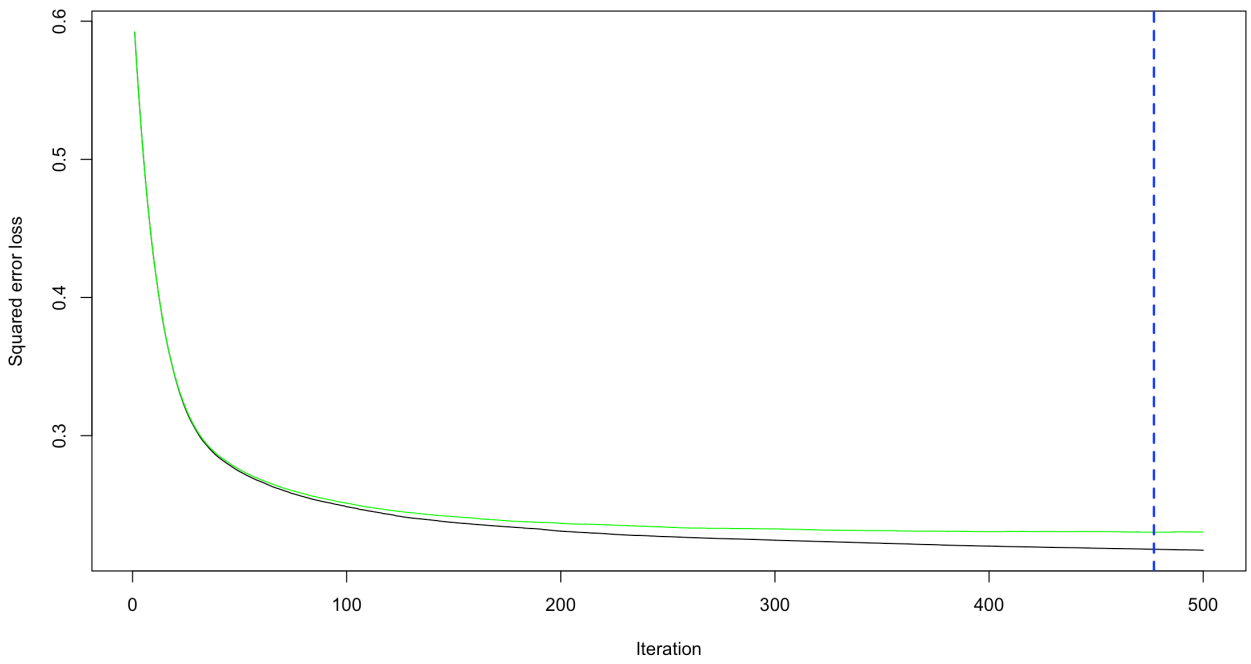
Third Model —— Boosting

My third model is Boosting. First, I specified the grid, and use "caret" package to train the boosting model and select best tuning parameter, and then use cross validation to choose number of iterations and evaluate prediction performance. Since we are using root mean squared logarithmic error as evaluation metric, in order to get a better performance, I perform logarithm with base e to trip_duration, and then perform exponential function to the prediction value when evaluating the error. The optimal model have tuning parameters n.trees = 500, interaction.depth = 2, and shrinkage = 0.05. This is what I got:

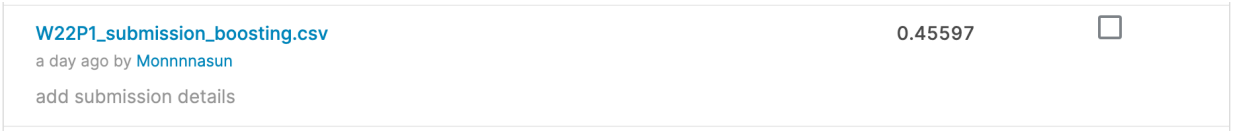
```
#Model 3: boosting
#select key parameters
ctrl <- trainControl(method = "repeatedcv",
                     number = 10,
                     repeats = 5)

gbm.Grid = expand.grid(interaction.depth = c(2,3,4,5,6),
                       n.trees = (1:5)*200,
                       shrinkage = c(0.15, 0.1, 0.05),
                       n.minobsinnode = 10)
gbm.cv.model <- train(log(trip_duration)~., data=train_3, method = "gbm", trControl = ctrl,
                      tuneGrid = gbm.Grid, verbose = FALSE)
gbm.cv.model
cv.num = gbm.perf(boost.trip)
boost.pred=exp(redict(gbm.cv.model,newdata=test_2[-6], n.trees = cv.num))
rmsle(boost.pred, trip_Duration[index])

boosting_pred <- predict(gbm.cv.model, as.matrix(test1), n.trees = cv.num)
out_Dat_3 = data.frame(id = test_dat$id, trip_duration = exp(boosting_pred))
colnames(out_Dat_3) = c('id', 'trip_duration')
write.csv(out_Dat_3, "Desktop/w22proj1/W22P1_submission_boosting.csv", row.names = FALSE)
```



My best score for boosting is 0.45597.



Summary

Model	Best Kaggle Score
Lasso regression + Ridge regression	0.59186
Random Forest	0.43090
Boosting	0.45597

By the comparison above, we can see that random forest has the best performance.