# 626 hw2

```
dat <- read.csv("C:/Users/linzh/Desktop/ECON 626/midterm1_trainingdata (1).csv")

dat$BUILT<-cut(dat$BUILT, c(1918,1942,1965,1988,2011), labels = c(1, 2, 3, 4))
dat$LUNITSF = log(dat$UNITSF)
dat$LLOT = log(dat$LOT)

# Use backward stepwise selection to reduce insignificant variables
temp <- lm(LOGVALUE~.+ REGION * METRO+ LUNITSF + LLOT, data = dat)
summary(temp)
```

```
##
## Call:
## lm(formula = LOGVALUE ~ . + REGION * METRO + LUNITSF + LLOT,
##     data = dat)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -12.9398  -0.3300   0.0714   0.4695   3.4297
##
## Coefficients:
##                Estimate Std. Error t value Pr(>|t|)
## (Intercept)   5.550e+00  2.232e-01  24.870  < 2e-16 ***
## BATHS         2.047e-01  1.221e-02  16.762  < 2e-16 ***
## BEDRMS        7.322e-03  1.281e-02   0.572   0.5675
## BUILT2       -3.858e-03  2.223e-02  -0.174   0.8622
## BUILT3       -1.857e-01  2.234e-02  -8.314  < 2e-16 ***
## BUILT4       -1.584e-01  2.302e-02  -6.879 6.21e-12 ***
## UNITSF       -5.861e-05  6.715e-06  -8.728  < 2e-16 ***
## LOT          -4.379e-07  8.233e-08  -5.318 1.06e-07 ***
## ROOMS         1.185e-01  7.506e-03  15.793  < 2e-16 ***
## REGION        5.271e-01  1.766e-02  29.841  < 2e-16 ***
## KITCHEN      -1.312e-01  9.214e-02  -1.424   0.1544
## FLOORS        1.130e-01  7.178e-03  15.747  < 2e-16 ***
## LAUNDY       -7.700e-02  1.324e-02  -5.815 6.17e-09 ***
## RECRM        -2.950e-02  2.671e-02  -1.105   0.2694
## METRO         1.275e-01  8.694e-03  14.669  < 2e-16 ***
## METRO3       -1.584e-02  7.551e-03  -2.098   0.0359 *
## LUNITSF       4.545e-01  2.814e-02  16.152  < 2e-16 ***
## LLOT          5.178e-02  7.579e-03   6.831 8.66e-12 ***
## REGION:METRO -3.894e-02  2.775e-03 -14.031  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.9469 on 19980 degrees of freedom
## Multiple R-squared:  0.2502, Adjusted R-squared:  0.2495
## F-statistic: 370.4 on 18 and 19980 DF,  p-value: < 2.2e-16
```

```
temp2 <- lm(LOGVALUE ~ . + REGION * METRO+ LUNITSF + LLOT - BEDRMS, data = dat)
summary(temp2)
```

```
##
## Call:
## lm(formula = LOGVALUE ~ . + REGION * METRO + LUNITSF + LLOT -
##     BEDRMS, data = dat)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -12.9436  -0.3299   0.0713   0.4695   3.4295
##
## Coefficients:
##                Estimate Std. Error t value Pr(>|t|)
## (Intercept)   5.548e+00  2.231e-01  24.865  < 2e-16 ***
## BATHS         2.056e-01  1.211e-02  16.981  < 2e-16 ***
## BUILT2       -3.054e-03  2.218e-02  -0.138   0.8905
## BUILT3       -1.853e-01  2.233e-02  -8.300  < 2e-16 ***
## BUILT4       -1.577e-01  2.300e-02  -6.860 7.11e-12 ***
## UNITSF       -5.885e-05  6.701e-06  -8.781  < 2e-16 ***
## LOT          -4.363e-07  8.229e-08  -5.302 1.16e-07 ***
## ROOMS         1.213e-01  5.712e-03  21.241  < 2e-16 ***
## REGION        5.269e-01  1.766e-02  29.836  < 2e-16 ***
## KITCHEN      -1.320e-01  9.213e-02  -1.433   0.1520
## FLOORS        1.129e-01  7.172e-03  15.737  < 2e-16 ***
## LAUNDY       -7.734e-02  1.323e-02  -5.846 5.13e-09 ***
## RECRM        -3.232e-02  2.625e-02  -1.231   0.2183
## METRO         1.275e-01  8.694e-03  14.663  < 2e-16 ***
## METRO3       -1.585e-02  7.551e-03  -2.100   0.0358 *
## LUNITSF       4.558e-01  2.804e-02  16.256  < 2e-16 ***
## LLOT          5.148e-02  7.561e-03   6.808 1.02e-11 ***
## REGION:METRO -3.891e-02  2.775e-03 -14.022  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.9469 on 19981 degrees of freedom
## Multiple R-squared:  0.2502, Adjusted R-squared:  0.2496
## F-statistic: 392.2 on 17 and 19981 DF,  p-value: < 2.2e-16
```

```
temp3 <- lm(LOGVALUE ~ . + REGION * METRO+ LUNITSF + LLOT - BEDRMS - RECRM, data = dat)
summary(temp3)
```

```
##
## Call:
## lm(formula = LOGVALUE ~ . + REGION * METRO + LUNITSF + LLOT -
##     BEDRMS - RECRM, data = dat)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -12.9399  -0.3295   0.0717   0.4692   3.4312
##
## Coefficients:
```

```
##                 Estimate Std. Error t value Pr(>|t|)
## (Intercept)   5.552e+00  2.231e-01  24.882  < 2e-16 ***
## BATHS         2.057e-01  1.211e-02  16.982  < 2e-16 ***
## BUILT2       -3.908e-03  2.217e-02  -0.176   0.8601
## BUILT3       -1.860e-01  2.232e-02  -8.333  < 2e-16 ***
## BUILT4       -1.585e-01  2.299e-02  -6.895 5.56e-12 ***
## UNITSF       -5.929e-05  6.692e-06  -8.860  < 2e-16 ***
## LOT          -4.349e-07  8.228e-08  -5.286 1.26e-07 ***
## ROOMS         1.198e-01  5.569e-03  21.505  < 2e-16 ***
## REGION        5.269e-01  1.766e-02  29.837  < 2e-16 ***
## KITCHEN      -1.328e-01  9.213e-02  -1.441   0.1496
## FLOORS        1.126e-01  7.169e-03  15.707  < 2e-16 ***
## LAUNDY       -7.830e-02  1.321e-02  -5.928 3.11e-09 ***
## METRO         1.273e-01  8.692e-03  14.642  < 2e-16 ***
## METRO3       -1.604e-02  7.549e-03  -2.124   0.0337 *
## LUNITSF       4.572e-01  2.802e-02  16.315  < 2e-16 ***
## LLOT          5.129e-02  7.560e-03   6.784 1.20e-11 ***
## REGION:METRO -3.885e-02  2.774e-03 -14.003  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.9469 on 19982 degrees of freedom
## Multiple R-squared:  0.2502, Adjusted R-squared:  0.2496
## F-statistic: 416.6 on 16 and 19982 DF,  p-value: < 2.2e-16
```

```
temp4 <- lm(LOGVALUE ~ . + REGION * METRO+ LUNITSF + LLOT - BEDRMS - RECRM - KITCHEN, data = dat)
summary(temp4)
```

```
##
## Call:
## lm(formula = LOGVALUE ~ . + REGION * METRO + LUNITSF + LLOT -
##     BEDRMS - RECRM - KITCHEN, data = dat)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -12.9385  -0.3289   0.0719   0.4692   3.4312
##
## Coefficients:
##                 Estimate Std. Error t value Pr(>|t|)
## (Intercept)   5.420e+00  2.034e-01  26.642  < 2e-16 ***
## BATHS         2.059e-01  1.211e-02  17.003  < 2e-16 ***
## BUILT2       -3.794e-03  2.217e-02  -0.171   0.8641
## BUILT3       -1.860e-01  2.232e-02  -8.334  < 2e-16 ***
## BUILT4       -1.585e-01  2.299e-02  -6.896 5.52e-12 ***
## UNITSF       -5.932e-05  6.692e-06  -8.865  < 2e-16 ***
## LOT          -4.321e-07  8.226e-08  -5.253 1.51e-07 ***
## ROOMS         1.200e-01  5.568e-03  21.550  < 2e-16 ***
## REGION        5.274e-01  1.766e-02  29.871  < 2e-16 ***
## FLOORS        1.123e-01  7.166e-03  15.675  < 2e-16 ***
## LAUNDY       -7.794e-02  1.321e-02  -5.902 3.65e-09 ***
## METRO         1.278e-01  8.685e-03  14.712  < 2e-16 ***
## METRO3       -1.603e-02  7.550e-03  -2.123   0.0338 *
## LUNITSF       4.570e-01  2.802e-02  16.310  < 2e-16 ***
## LLOT          5.083e-02  7.553e-03   6.729 1.75e-11 ***
```

```
## REGION:METRO -3.896e-02  2.773e-03 -14.049  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.9469 on 19983 degrees of freedom
## Multiple R-squared:  0.2501, Adjusted R-squared:  0.2495
## F-statistic: 444.2 on 15 and 19983 DF,  p-value: < 2.2e-16
```

```
# Predictor variables
func <- as.formula('LOGVALUE ~ . + REGION * METRO+ LUNITSF + LLOT- BEDRMS - RECRM - KITCHEN')
x = model.matrix(func ,dat)[,-1]
# Outcome variable
y = dat$LOGVALUE

# separate train data and test data
set.seed (1)
num_obs = nrow(dat)
train_index = sample(num_obs, size = trunc(0.50 * num_obs))# 50/50 split between test and train
train_data = dat[train_index, ]# Training sample
test_data = dat[-train_index, ]# Validation sample

# Predictor variables
x_train = model.matrix(func,train_data)[,-1]
x_test = model.matrix(func,test_data)[,-1]

# Outcome variable
y_train = train_data$LOGVALUE
y_test = test_data$LOGVALUE
```

```
########################
#The Lasso: alpha = 1##
########################
library(glmnet)
```

```
## Loading required package: Matrix
```

```
## Loaded glmnet 4.1
```

```
set.seed(1)
grid = 10 ^ seq (10,-2, length =100)
lasso.mod = glmnet(x, y, alpha = 1, lambda = grid)
dim(coef(lasso.mod))
```

```
## [1]  16 100
```

```
#use cross validation to get the tuning parameter lambda with the smallest cross validation error
set.seed (1)
cv.out = cv.glmnet(x_train, y_train, alpha = 1) # Fit lasso model on training data
bestlam = cv.out$lambda.min  # Select lambda that minimizes training MSE
lasso.pred = predict(lasso.mod, s = bestlam, newx = x_test) # Use best lambda to predict test data
out = glmnet(x, y, alpha = 1, lambda =grid) # Fit lasso model on the full data set
lasso.coef = predict(out, type = "coefficients", s = bestlam)[1:16, ]
lasso.coef
```

```
##   (Intercept)         BATHS        BUILT2        BUILT3        BUILT4
##  7.326310e+00  1.913078e-01  3.822775e-02 -9.511330e-02 -6.397120e-02
##        UNITSF           LOT         ROOMS        REGION        FLOORS
## -2.099681e-05 -1.473636e-07  1.215994e-01  2.832177e-01  1.055922e-01
##        LAUNDY         METRO        METRO3       LUNITSF          LLOT
## -5.320672e-02  6.283883e-03  0.000000e+00  3.130051e-01  2.732110e-02
##  REGION:METRO
## -3.441088e-04
```

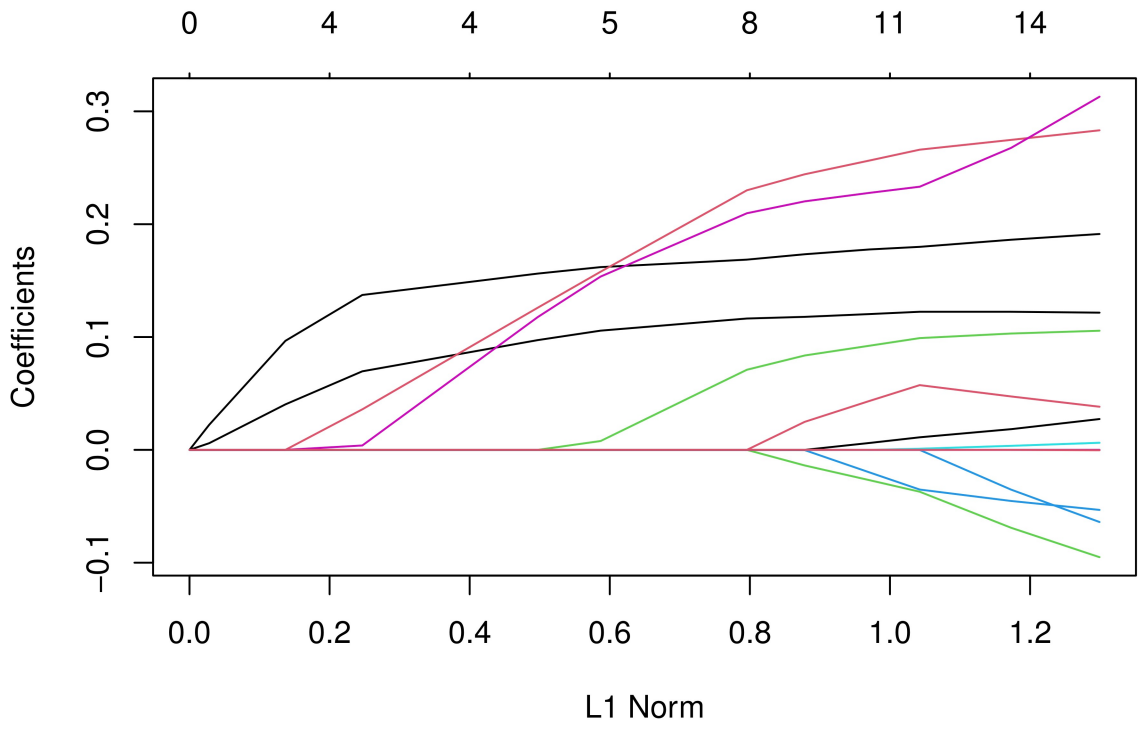
From the Lasso result, METRO3's coefficient is 0. Refit the model:

```
func <- as.formula('LOGVALUE ~ . + REGION * METRO+ LUNITSF + LLOT- BEDRMS - RECRM - KITCHEN - METRO3')
x = model.matrix(func ,dat)[,-1]
# Outcome variable
y = dat$LOGVALUE

# separate train data and test data
set.seed (1)
num_obs = nrow(dat)
train_index = sample(num_obs, size = trunc(0.50 * num_obs))# 50/50 split between test and train
train_data = dat[train_index, ]# Training sample
test_data = dat[-train_index, ]# Validation sample

# Predictor variables
x_train = model.matrix(func,train_data)[,-1]
x_test = model.matrix(func,test_data)[,-1]

# Outcome variable
y_train = train_data$LOGVALUE
y_test = test_data$LOGVALUE

########################
#The Lasso: refit##
########################
library(glmnet)

set.seed(1)
grid = 10 ^ seq (10,-2, length =100)
lasso.mod = glmnet(x, y, alpha = 1, lambda = grid)
dim(coef(lasso.mod))
```

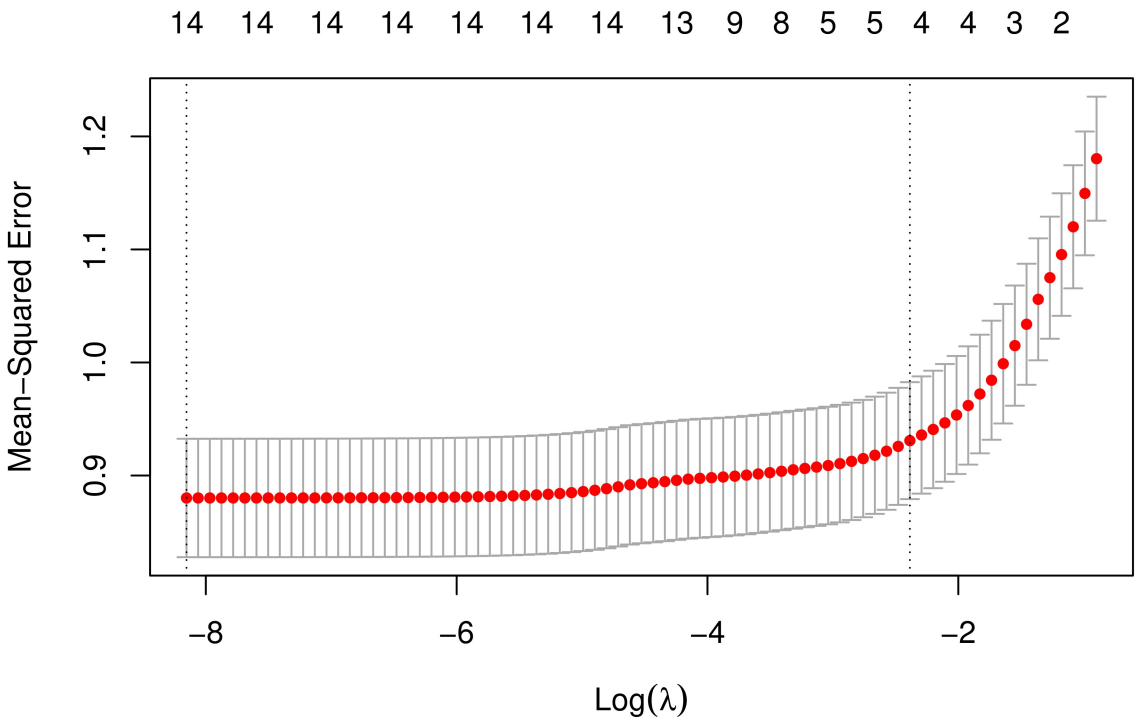
```
## [1]  15 100
```

```
plot(lasso.mod)    # Draw plot of coefficients
```

```
## Warning in regularize.values(x, y, ties, missing(ties), na.rm = na.rm):
## collapsing to unique 'x' values
```




```
#use cross validation to get the tuning parameter lambda with the smallest cross validation error
set.seed (1)
cv.out = cv.glmnet(x_train, y_train, alpha = 1) # Fit lasso model on training data
plot(cv.out)
```




```
bestlam = cv.out$lambda.min  # Select lambda that minimizes training MSE
bestlam
```

```
## [1] 0.0002875116
```

```
lasso.pred = predict(lasso.mod, s = bestlam, newx = x_test) # Use best lambda to predict test data
mean((lasso.pred - y_test)^2) # Calculate test MSE
```

```
## [1] 0.9276267
```

```
SSE = sum((lasso.pred - y_test)^2)
SST = sum((y_test - mean(y_test))^2)
R_square = 1 - SSE / SST #R_square
R_square
```

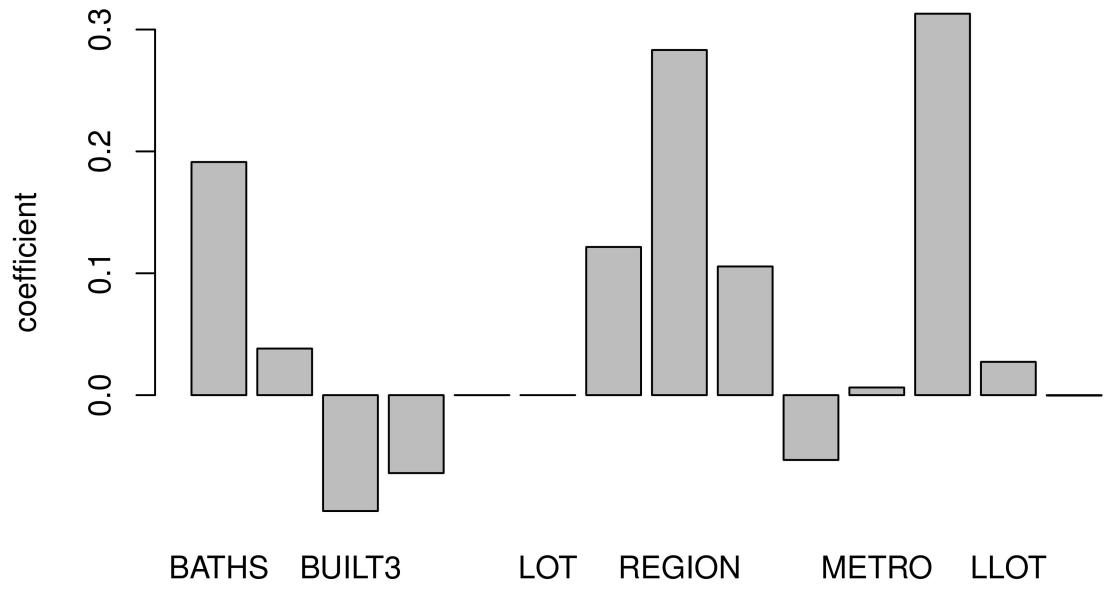
```
## [1] 0.2318339
```

```
out = glmnet(x, y, alpha = 1, lambda = grid) # Fit lasso model on the full data set
lasso.coef = predict(out, type = "coefficients", s = bestlam)[1:15, ]
lasso.coef
```

```
##   (Intercept)         BATHS        BUILT2        BUILT3        BUILT4
##  7.326310e+00  1.913078e-01  3.822775e-02 -9.511330e-02 -6.397120e-02
```

```
##        UNITSF           LOT         ROOMS        REGION        FLOORS
## -2.099681e-05 -1.473636e-07  1.215994e-01  2.832177e-01  1.055922e-01
##        LAUNDY         METRO       LUNITSF          LLOT  REGION:METRO
## -5.320672e-02  6.283883e-03  3.130051e-01  2.732110e-02 -3.441088e-04
```

```
barplot(lasso.coef[-1], ylab = 'coefficient', sub = "Relative Importance of Features in Lasso Model")
```

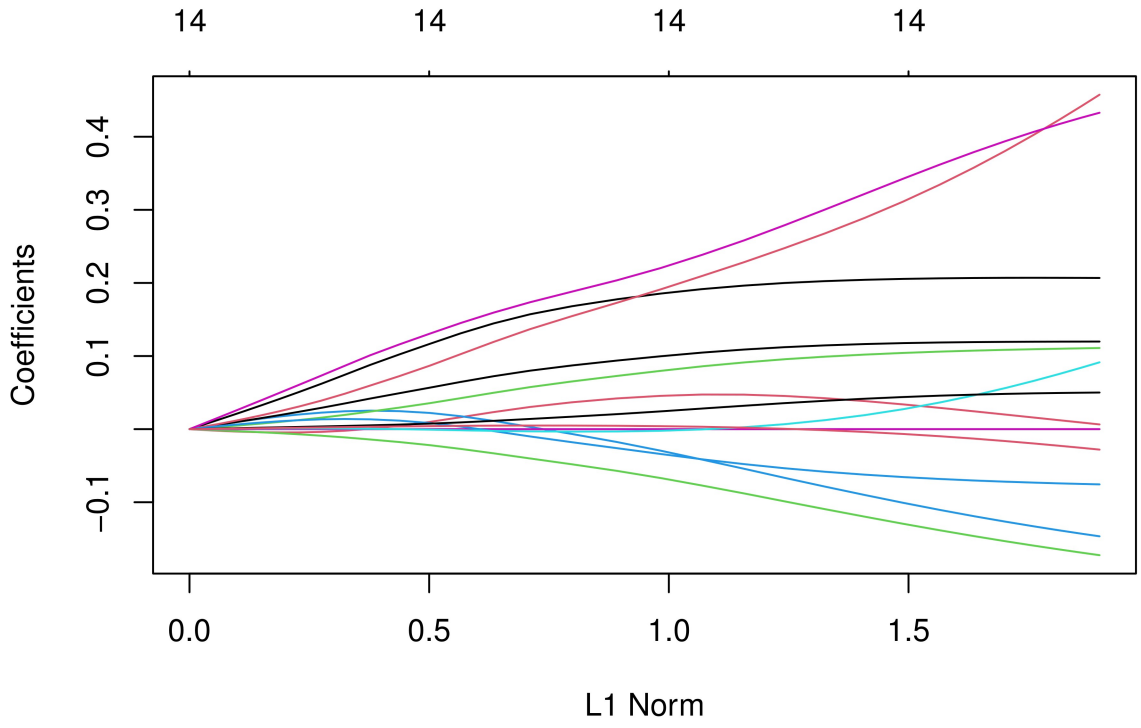



Relative Importance of Features in Lasso Model

```
##############################
#Ridge Regression: alpha = 0#
##############################
library(glmnet)

set.seed(1)
grid = 10 ^ seq (10, -2, length = 100)
ridge.mod = glmnet(x, y, alpha = 0, lambda = grid)
dim(coef(ridge.mod))
```

```
## [1]  15 100
```

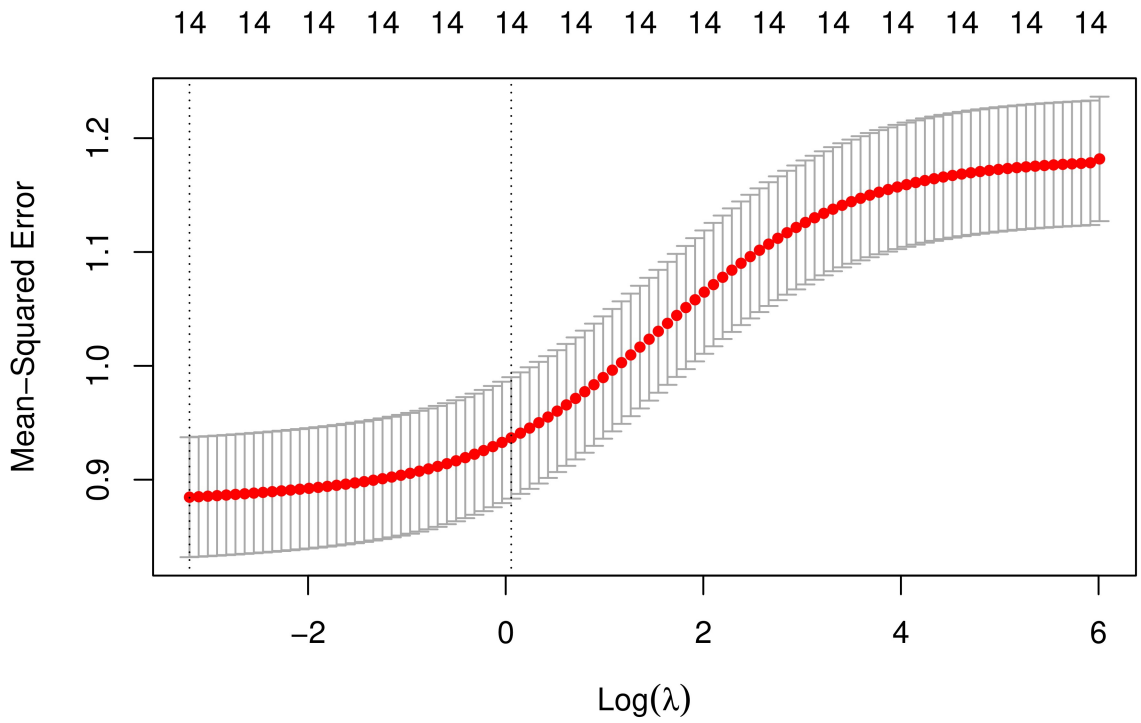
```
plot(ridge.mod)    # Draw plot of coefficients
```




```
#use cross validation to get the tuning parameter lambda with the smallest cross validation error
set.seed (1)
cv.out = cv.glmnet(x_train, y_train, alpha = 0) # Fit ridge regression model on training data
bestlam = cv.out$lambda.min  # Select lambda that minimizes training MSE
bestlam
```

```
## [1] 0.04075403
```

```
plot(cv.out) # Draw plot of training MSE as a function of lambda
```




```
ridge.pred = predict(ridge.mod, s = bestlam, newx = x_test) # Use best lambda to predict test data
mean((ridge.pred - y_test)^2) # Calculate test MSE  0.9205295
```

```
## [1] 0.9205295
```

```
SSE = sum((ridge.pred - y_test)^2)
SST = sum((y_test - mean(y_test))^2)
R_square = 1 - SSE / SST #R_square
R_square
```

```
## [1] 0.2377111
```

```
out = glmnet(x, y, alpha = 0, lambda = grid) # Fit ridge regression model on the full data set
ridge.coef = predict(out, type = "coefficients", s = bestlam)[1:15,]
ridge.coef
```

```
##   (Intercept)         BATHS        BUILT2        BUILT3        BUILT4
##  6.486044e+00  2.067565e-01  2.457050e-02 -1.460004e-01 -1.187315e-01
##        UNITSF           LOT         ROOMS        REGION        FLOORS
## -4.006802e-05 -3.792391e-07  1.190505e-01  3.568768e-01  1.073679e-01
##        LAUNDY         METRO       LUNITSF          LLOT  REGION:METRO
## -7.014779e-02  4.582733e-02  3.783920e-01  4.714921e-02 -1.284409e-02
```

```
barplot(ridge.coef[-1], ylab = 'coefficient', sub = "Relative Importance of Features in Ridge Model")
```