HW8_11_1.R

gH0$t

2021-03-17

```
# Install required library package
#install.packages("glmnet")

# Clear environment
rm(list = ls())

# Load required libraries
library(glmnet)
```

```
## Warning: package 'glmnet' was built under R version 4.0.4

## Loading required package: Matrix

## Loaded glmnet 4.1-1
```

```
# Load data from uscrime.txt into a table
uscrime <- read.table("uscrime.txt", stringsAsFactors = FALSE, header = TRUE)

# Optional check to make sure the data is read correctly
head(uscrime)
```

```
##      M So   Ed  Po1  Po2    LF   M.F Pop   NW    U1  U2 Wealth Ineq     Prob
## 1 15.1  1  9.1  5.8  5.6 0.510  95.0  33 30.1 0.108 4.1   3940 26.1 0.084602
## 2 14.3  0 11.3 10.3  9.5 0.583 101.2  13 10.2 0.096 3.6   5570 19.4 0.029599
## 3 14.2  1  8.9  4.5  4.4 0.533  96.9  18 21.9 0.094 3.3   3180 25.0 0.083401
## 4 13.6  0 12.1 14.9 14.1 0.577  99.4 157  8.0 0.102 3.9   6730 16.7 0.015801
## 5 14.1  0 12.1 10.9 10.1 0.591  98.5  18  3.0 0.091 2.0   5780 17.4 0.041399
## 6 12.1  0 11.0 11.8 11.5 0.547  96.4  25  4.4 0.084 2.9   6890 12.6 0.034201
##      Time Crime
## 1 26.2011   791
## 2 25.2999  1635
## 3 24.3006   578
## 4 29.9012  1969
## 5 21.2998  1234
## 6 20.9995   682
```

```
# Setting the random number generator seed so that our results are reproducible
set.seed(1)

##### Part 1 #####
```

```r
# Perform backward elimination
model_back <- lm(Crime~., data = uscrime)
# step(model_back, direction = "backward")
step(model_back, direction = "backward", trace = 0)
```

```
##
## Call:
## lm(formula = Crime ~ M + Ed + Po1 + M.F + U1 + U2 + Ineq + Prob,
##     data = uscrime)
##
## Coefficients:
## (Intercept)            M           Ed          Po1          M.F           U1
##    -6426.10        93.32       180.12       102.65        22.34     -6086.63
##          U2         Ineq         Prob
##      187.35        61.33     -3796.03
```

```r
# Perform forward selection
model_forward <- lm(Crime~1, data = uscrime)
# step(model_forward, direction = "forward")
step(model_forward, direction = "forward", trace = 0)
```

```
##
## Call:
## lm(formula = Crime ~ 1, data = uscrime)
##
## Coefficients:
## (Intercept)
##       905.1
```

```r
# Perform Stepwise Regression
model_both <- lm(Crime~., data = uscrime)
step(model_both,
     scope = list(lower = formula(lm(Crime~1, data = uscrime)),
                  upper = formula(lm(Crime~., data = uscrime))),
     direction = "both")
```

```
## Start:  AIC=514.65
## Crime ~ M + So + Ed + Po1 + Po2 + LF + M.F + Pop + NW + U1 +
##     U2 + Wealth + Ineq + Prob + Time
##
##          Df Sum of Sq     RSS    AIC
## - So      1        29 1354974 512.65
## - LF      1      8917 1363862 512.96
## - Time    1     10304 1365250 513.00
## - Pop     1     14122 1369068 513.14
## - NW      1     18395 1373341 513.28
## - M.F     1     31967 1386913 513.74
## - Wealth  1     37613 1392558 513.94
## - Po2     1     37919 1392865 513.95
## <none>                1354946 514.65
## - U1      1     83722 1438668 515.47
```

```
## - Po1     1    144306 1499252 517.41
## - U2      1    181536 1536482 518.56
## - M       1    193770 1548716 518.93
## - Prob    1    199538 1554484 519.11
## - Ed      1    402117 1757063 524.86
## - Ineq    1    423031 1777977 525.42
##
## Step:  AIC=512.65
## Crime ~ M + Ed + Po1 + Po2 + LF + M.F + Pop + NW + U1 + U2 +
##     Wealth + Ineq + Prob + Time
##
##          Df Sum of Sq     RSS    AIC
## - Time    1     10341 1365315 511.01
## - LF      1     10878 1365852 511.03
## - Pop     1     14127 1369101 511.14
## - NW      1     21626 1376600 511.39
## - M.F     1     32449 1387423 511.76
## - Po2     1     37954 1392929 511.95
## - Wealth  1     39223 1394197 511.99
## <none>                1354974 512.65
## - U1      1     96420 1451395 513.88
## + So      1        29 1354946 514.65
## - Po1     1    144302 1499277 515.41
## - U2      1    189859 1544834 516.81
## - M       1    195084 1550059 516.97
## - Prob    1    204463 1559437 517.26
## - Ed      1    403140 1758114 522.89
## - Ineq    1    488834 1843808 525.13
##
## Step:  AIC=511.01
## Crime ~ M + Ed + Po1 + Po2 + LF + M.F + Pop + NW + U1 + U2 +
##     Wealth + Ineq + Prob
##
##          Df Sum of Sq     RSS    AIC
## - LF      1     10533 1375848 509.37
## - NW      1     15482 1380797 509.54
## - Pop     1     21846 1387161 509.75
## - Po2     1     28932 1394247 509.99
## - Wealth  1     36070 1401385 510.23
## - M.F     1     41784 1407099 510.42
## <none>                1365315 511.01
## - U1      1     91420 1456735 512.05
## + Time    1     10341 1354974 512.65
## + So      1        65 1365250 513.00
## - Po1     1    134137 1499452 513.41
## - U2      1    184143 1549458 514.95
## - M       1    186110 1551425 515.01
## - Prob    1    237493 1602808 516.54
## - Ed      1    409448 1774763 521.33
## - Ineq    1    502909 1868224 523.75
##
## Step:  AIC=509.37
## Crime ~ M + Ed + Po1 + Po2 + M.F + Pop + NW + U1 + U2 + Wealth +
##     Ineq + Prob
```

```
##
##          Df Sum of Sq     RSS    AIC
## - NW      1     11675 1387523 507.77
## - Po2     1     21418 1397266 508.09
## - Pop     1     27803 1403651 508.31
## - M.F     1     31252 1407100 508.42
## - Wealth  1     35035 1410883 508.55
## <none>                1375848 509.37
## - U1      1     80954 1456802 510.06
## + LF      1     10533 1365315 511.01
## + Time    1      9996 1365852 511.03
## + So      1      3046 1372802 511.26
## - Po1     1    123896 1499744 511.42
## - U2      1    190746 1566594 513.47
## - M       1    217716 1593564 514.27
## - Prob    1    226971 1602819 514.54
## - Ed      1    413254 1789103 519.71
## - Ineq    1    500944 1876792 521.96
##
## Step:  AIC=507.77
## Crime ~ M + Ed + Po1 + Po2 + M.F + Pop + U1 + U2 + Wealth + Ineq +
##     Prob
##
##          Df Sum of Sq     RSS    AIC
## - Po2     1     16706 1404229 506.33
## - Pop     1     25793 1413315 506.63
## - M.F     1     26785 1414308 506.66
## - Wealth  1     31551 1419073 506.82
## <none>                1387523 507.77
## - U1      1     83881 1471404 508.52
## + NW      1     11675 1375848 509.37
## + So      1      7207 1380316 509.52
## + LF      1      6726 1380797 509.54
## + Time    1      4534 1382989 509.61
## - Po1     1    118348 1505871 509.61
## - U2      1    201453 1588976 512.14
## - Prob    1    216760 1604282 512.59
## - M       1    309214 1696737 515.22
## - Ed      1    402754 1790276 517.74
## - Ineq    1    589736 1977259 522.41
##
## Step:  AIC=506.33
## Crime ~ M + Ed + Po1 + M.F + Pop + U1 + U2 + Wealth + Ineq +
##     Prob
##
##          Df Sum of Sq     RSS    AIC
## - Pop     1     22345 1426575 505.07
## - Wealth  1     32142 1436371 505.39
## - M.F     1     36808 1441037 505.54
## <none>                1404229 506.33
## - U1      1     86373 1490602 507.13
## + Po2     1     16706 1387523 507.77
## + NW      1      6963 1397266 508.09
## + So      1      3807 1400422 508.20
```

```
## + LF      1      1986 1402243 508.26
## + Time    1       575 1403654 508.31
## - U2      1    205814 1610043 510.76
## - Prob    1    218607 1622836 511.13
## - M       1    307001 1711230 513.62
## - Ed      1    389502 1793731 515.83
## - Ineq    1    608627 2012856 521.25
## - Po1     1   1050202 2454432 530.57
##
## Step:  AIC=505.07
## Crime ~ M + Ed + Po1 + M.F + U1 + U2 + Wealth + Ineq + Prob
##
##          Df Sum of Sq     RSS    AIC
## - Wealth  1     26493 1453068 503.93
## <none>                1426575 505.07
## - M.F     1     84491 1511065 505.77
## - U1      1     99463 1526037 506.24
## + Pop     1     22345 1404229 506.33
## + Po2     1     13259 1413315 506.63
## + NW      1      5927 1420648 506.87
## + So      1      5724 1420851 506.88
## + LF      1      5176 1421398 506.90
## + Time    1      3913 1422661 506.94
## - Prob    1    198571 1625145 509.20
## - U2      1    208880 1635455 509.49
## - M       1    320926 1747501 512.61
## - Ed      1    386773 1813348 514.35
## - Ineq    1    594779 2021354 519.45
## - Po1     1   1127277 2553852 530.44
##
## Step:  AIC=503.93
## Crime ~ M + Ed + Po1 + M.F + U1 + U2 + Ineq + Prob
##
##          Df Sum of Sq     RSS    AIC
## <none>                1453068 503.93
## + Wealth  1     26493 1426575 505.07
## - M.F     1    103159 1556227 505.16
## + Pop     1     16697 1436371 505.39
## + Po2     1     14148 1438919 505.47
## + So      1      9329 1443739 505.63
## + LF      1      4374 1448694 505.79
## + NW      1      3799 1449269 505.81
## + Time    1      2293 1450775 505.86
## - U1      1    127044 1580112 505.87
## - Prob    1    247978 1701046 509.34
## - U2      1    255443 1708511 509.55
## - M       1    296790 1749858 510.67
## - Ed      1    445788 1898855 514.51
## - Ineq    1    738244 2191312 521.24
## - Po1     1   1672038 3125105 537.93
```

```
##
## Call:
## lm(formula = Crime ~ M + Ed + Po1 + M.F + U1 + U2 + Ineq + Prob,
```

```
##     data = uscrime)
##
## Coefficients:
## (Intercept)            M           Ed          Po1          M.F           U1
##    -6426.10        93.32       180.12       102.65        22.34     -6086.63
##          U2         Ineq         Prob
##      187.35        61.33     -3796.03
```

```
# Fit Regression Model using identified coefficients
model_step <- lm(formula = Crime ~ M + Ed + Po1 + M.F + U1 + U2 + Ineq + Prob,
                 data = uscrime)
summary(model_step)
```

```
##
## Call:
## lm(formula = Crime ~ M + Ed + Po1 + M.F + U1 + U2 + Ineq + Prob,
##     data = uscrime)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -444.70 -111.07    3.03  122.15  483.30
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept) -6426.10    1194.61  -5.379 4.04e-06 ***
## M              93.32      33.50   2.786  0.00828 **
## Ed            180.12      52.75   3.414  0.00153 **
## Po1           102.65      15.52   6.613 8.26e-08 ***
## M.F            22.34      13.60   1.642  0.10874
## U1          -6086.63    3339.27  -1.823  0.07622 .
## U2            187.35      72.48   2.585  0.01371 *
## Ineq           61.33      13.96   4.394 8.63e-05 ***
## Prob        -3796.03    1490.65  -2.547  0.01505 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 195.5 on 38 degrees of freedom
## Multiple R-squared:  0.7888, Adjusted R-squared:  0.7444
## F-statistic: 17.74 on 8 and 38 DF,  p-value: 1.159e-10
```

```
# Scale the data and convert it to a matrix for LASSO and Elastic Net
scaled_data <- as.data.frame(scale(uscrime[,c(1,3:15)]))
scaled_data <- cbind(uscrime[,2],scaled_data,uscrime[,16])
colnames(scaled_data)[1] <- "So"
colnames(scaled_data)[16] <- "Crime"
data_mx <- as.matrix(scaled_data)
predictors = data_mx[,1:15]
response = data_mx[, 16]

# Split uscrime into training and test data sets
r = nrow(scaled_data)
set = sample(1:r, size = round(r * .8), replace = FALSE)
train = scaled_data[set,]
```

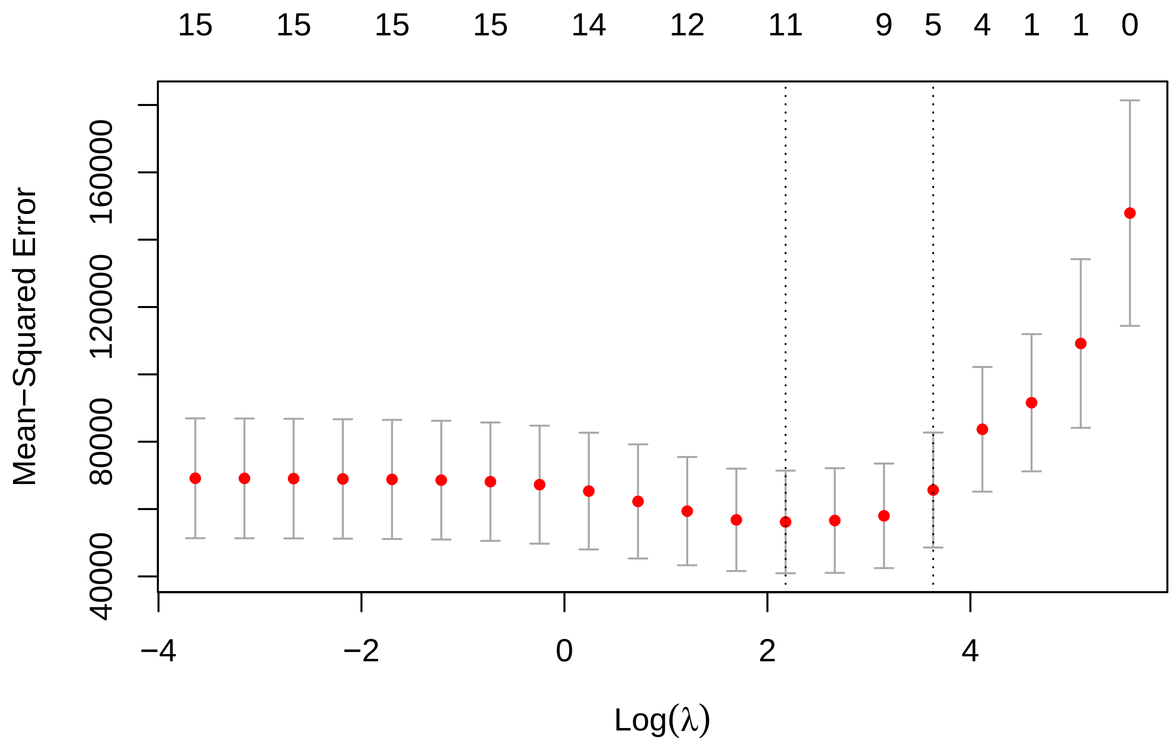
```r
test = scaled_data[-set,]


##### Part 2 #####

# Perform LASSO
model_lasso <- cv.glmnet(x = predictors,
                         y = response,
                         alpha = 1,
                         nfolds = 8,
                         nlambda = 20,
                         type.measure = "mse",
                         family = "gaussian",
                         standardize = TRUE)
model_lasso
```

```
##
## Call:  cv.glmnet(x = predictors, y = response, type.measure = "mse",      nfolds = 8, alpha = 1, nla
##
## Measure: Mean-Squared Error
##
##     Lambda Index Measure    SE Nonzero
## min   8.84     8   56179 15236      11
## 1se  37.84     5   65655 17071       5
```

```r
plot(model_lasso)
```




```
model_lasso$lambda.min
```

```
## [1] 8.839527
```

```
model_lasso$lambda.1se
```

```
## [1] 37.84495
```

```
cbind(model_lasso$lambda, model_lasso$cvm, model_lasso$nzero)
```

```
##              [,1]      [,2] [,3]
## s0  263.09539664 147889.32    0
## s1  162.02682877 109170.53    1
## s2   99.78393228  91580.65    1
## s3   61.45175597  83684.15    4
## s4   37.84495384  65655.02    5
## s5   23.30674704  57990.46    9
## s6   14.35341842  56597.94   10
## s7    8.83952702  56178.58   11
## s8    5.44380688  56797.67   12
## s9    3.35255872  59387.37   12
## s10   2.06466728  62271.26   13
## s11   1.27152165  65342.32   14
## s12   0.78306433  67241.25   15
```

```
## s13   0.48224876  68134.60   15
## s14   0.29699204  68586.41   15
## s15   0.18290201  68802.90   15
## s16   0.11263988  68946.19   14
## s17   0.06936907  69041.67   15
## s18   0.04272082  69112.65   15
## s19   0.02630954  69150.22   15
```

```
coef(model_lasso, s = model_lasso$lambda.min)
```

```
## 16 x 1 sparse Matrix of class "dgCMatrix"
##                      1
## (Intercept) 889.854605
## So           44.739597
## M            90.279192
## Ed          140.289856
## Po1         304.140909
## Po2           .
## LF            .
## M.F          55.640579
## Pop           .
## NW            6.487469
## U1          -38.645259
## U2           74.618077
## Wealth        7.441720
## Ineq        194.791647
## Prob        -83.865228
## Time          .
```

```
coef(model_lasso, s = model_lasso$lambda.1se)
```

```
## 16 x 1 sparse Matrix of class "dgCMatrix"
##                     1
## (Intercept) 905.08511
## So            .
## M            37.58244
## Ed            .
## Po1         264.54147
## Po2           .
## LF            .
## M.F          44.90060
## Pop           .
## NW            .
## U1            .
## U2            .
## Wealth        .
## Ineq         62.38884
## Prob        -42.18236
## Time          .
```

```
# Using the lambda.min model:
# Calculate R-squared by first fitting a linear regression model using training
```

```r
# data set and then making a prediction model using the test data set
model_lmin = lm(Crime~ So + M + Ed + Po1 + M.F + NW + U1+ U2 + Wealth + Ineq + Prob,
           as.data.frame(train))
summary(model_lmin)
```

```
##
## Call:
## lm(formula = Crime ~ So + M + Ed + Po1 + M.F + NW + U1 + U2 +
##     Wealth + Ineq + Prob, data = as.data.frame(train))
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -229.36 -136.67   17.18   91.91  305.50
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)  897.899     50.757  17.690 5.12e-16 ***
## So           -54.019    129.694  -0.417 0.680456
## M             99.661     50.268   1.983 0.058072 .
## Ed           261.915     63.402   4.131 0.000332 ***
## Po1          220.041     61.401   3.584 0.001371 **
## M.F           -2.891     45.235  -0.064 0.949533
## NW            75.091     63.274   1.187 0.246053
## U1            10.249     64.234   0.160 0.874463
## U2            60.923     63.405   0.961 0.345474
## Wealth       140.073    106.515   1.315 0.199974
## Ineq         371.525     83.883   4.429 0.000152 ***
## Prob         -41.990     38.793  -1.082 0.289010
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 171.4 on 26 degrees of freedom
## Multiple R-squared:  0.7945, Adjusted R-squared:  0.7076
## F-statistic: 9.141 on 11 and 26 DF,  p-value: 2.066e-06
```

```r
pred = predict.lm(model_lmin, as.data.frame(test))
sse = sum((pred - test[,16]) ^ 2)
sst = sum((test[,16] - mean(test[,16])) ^ 2) #total sum of squares
1 - sse / sst
```

```
## [1] 0.5455591
```

```r
# Using the .1se model, which is the largest value of lambda
# such that error is within 1 standard error of the minimum:
model_se = lm(Crime ~ M + Po1 + M.F + Ineq + Prob, as.data.frame(train))
summary(model_se)
```

```
##
## Call:
## lm(formula = Crime ~ M + Po1 + M.F + Ineq + Prob, data = as.data.frame(train))
##
## Residuals:
```

```
##      Min       1Q   Median       3Q      Max
## -449.05 -123.35   33.02  116.40  392.94
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)   887.05      35.59  24.922  < 2e-16 ***
## M              85.77      47.78   1.795   0.0821 .
## Po1           308.78      54.94   5.620 3.27e-06 ***
## M.F            82.33      39.91   2.063   0.0473 *
## Ineq          140.06      60.66   2.309   0.0276 *
## Prob          -76.83      40.95  -1.876   0.0697 .
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 217.3 on 32 degrees of freedom
## Multiple R-squared:  0.5935, Adjusted R-squared:   0.53
## F-statistic: 9.345 on 5 and 32 DF,  p-value: 1.452e-05
```

```
predse = predict.lm(model_se, as.data.frame(test))
ssese = sum((predse - test[,16]) ^ 2)
sstse = sum((test[,16] - mean(test[,16])) ^ 2) #total sum of squares
1 - ssese / sstse
```

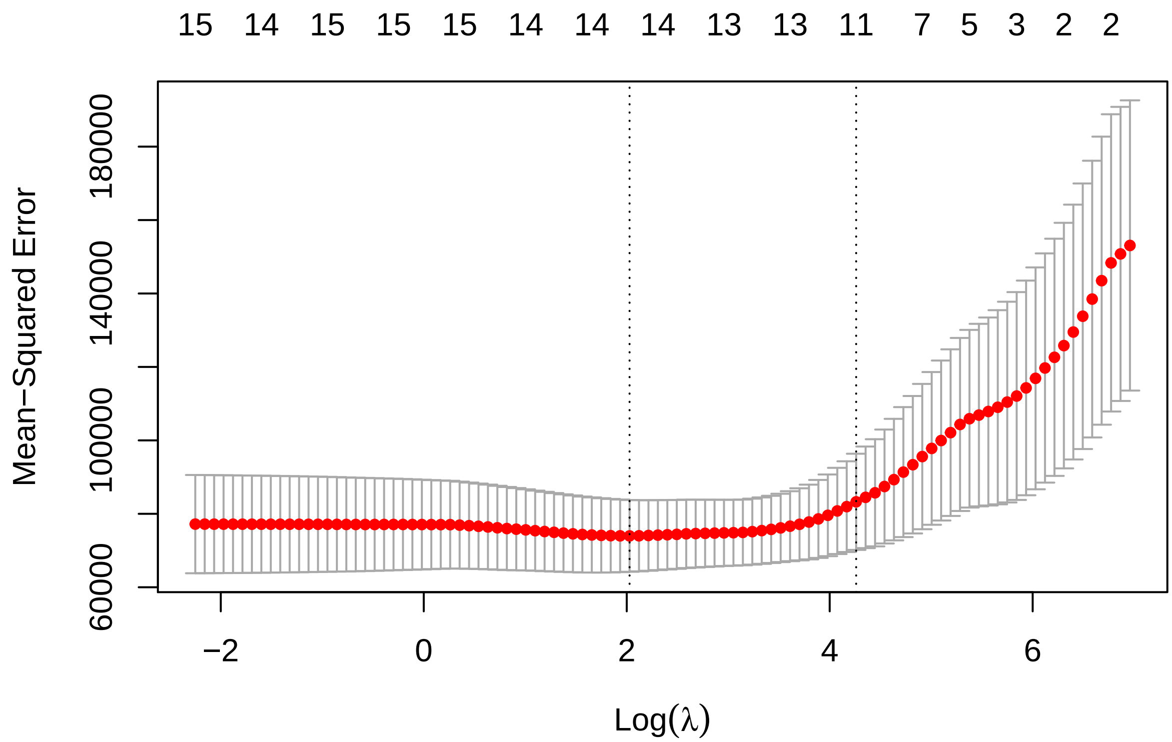
```
## [1] 0.7664472
```

```
###### Part 3 #####

# Perform Elastic Net
model_ent <- cv.glmnet(x = predictors,
                   y = response,
                   alpha = 0.25,
                   nfolds = 8,
                   type.measure = "mse",
                   family = "gaussian")
model_ent
```

```
##
## Call:  cv.glmnet(x = predictors, y = response, type.measure = "mse",      nfolds = 8, alpha = 0.25, 
##
## Measure: Mean-Squared Error
##
##     Lambda Index Measure    SE Nonzero
## min   7.60    54   73982  9774      13
## 1se  70.87    30   83264 13100      11
```

```
plot(model_ent)
```




```r
model_ent$lambda.min
```

```
## [1] 7.599046
```

```r
model_ent$lambda.1se
```

```
## [1] 70.86896
```

```r
cbind(model_ent$lambda, model_ent$cvm, model_ent$nzero)
```

```
##              [,1]      [,2] [,3]
## s0  1052.3815866 153082.77    0
## s1   958.8909070 150781.09    2
## s2   873.7056817 148334.67    2
## s3   796.0880770 143494.43    2
## s4   725.3658064 138478.13    2
## s5   660.9263074 133805.82    2
## s6   602.2114359 129500.14    2
## s7   548.7126318 125809.98    2
## s8   499.9665139 122623.71    2
## s9   455.5508668 119709.53    3
## s10  415.0809834 116892.44    3
## s11  378.2063329 114286.05    3
## s12  344.6075247 112077.85    3
```

```
## s13  313.9935420 110431.73    3
## s14  286.0992212 109019.65    4
## s15  260.6829549 107866.48    4
## s16  237.5245997 106887.04    4
## s17  216.4235689 105895.77    5
## s18  197.1970955 104331.53    6
## s19  179.6786491 102106.99    7
## s20  163.7164931  99960.16    7
## s21  149.1723711  97810.31    7
## s22  135.9203088  95589.22    7
## s23  123.8455232  93370.84   10
## s24  112.8434283  91359.72   11
## s25  102.8187291  89312.91   11
## s26   93.6845966  87436.80   11
## s27   85.3619153  85725.77   11
## s28   77.7785980  84491.42   11
## s29   70.8689617  83264.15   11
## s30   64.5731585  81950.42   11
## s31   58.8366571  80783.38   12
## s32   53.6097706  79589.79   12
## s33   48.8472263  78604.54   12
## s34   44.5077734  77761.70   13
## s35   40.5538255  77136.06   13
## s36   36.9511354  76626.66   13
## s37   33.6684984  76157.30   13
## s38   30.6774818  75755.80   13
## s39   27.9521788  75423.15   13
## s40   25.4689843  75144.75   13
## s41   23.2063899  74933.06   13
## s42   21.1447982  74847.87   13
## s43   19.2663526  74806.66   13
## s44   17.5547830  74745.26   13
## s45   15.9952644  74674.41   14
## s46   14.5742892  74615.42   14
## s47   13.2795495  74543.35   14
## s48   12.0998310  74400.84   14
## s49   11.0249153  74300.49   14
## s50   10.0454922  74197.12   14
## s51    9.1530784  74094.56   14
## s52    8.3399441  74018.72   14
## s53    7.5990465  73982.35   13
## s54    6.9239681  74003.05   13
## s55    6.3088619  74043.50   14
## s56    5.7484000  74121.15   14
## s57    5.2377280  74235.46   14
## s58    4.7724227  74361.41   14
## s59    4.3484538  74546.84   13
## s60    3.9621491  74744.26   14
## s61    3.6101627  74962.33   14
## s62    3.2894458  75178.03   14
## s63    2.9972205  75401.09   14
## s64    2.7309557  75619.67   14
## s65    2.4883451  75814.78   15
## s66    2.2672874  75986.43   15
```

```
## s67    2.0658678  76197.67   15
## s68    1.8823418  76418.72   15
## s69    1.7151198  76601.12   15
## s70    1.5627533  76770.39   15
## s71    1.4239226  76916.74   15
## s72    1.2974252  77043.54   15
## s73    1.1821655  77052.61   15
## s74    1.0771452  77057.96   15
## s75    0.9814546  77069.92   15
## s76    0.8942649  77083.49   15
## s77    0.8148208  77094.16   15
## s78    0.7424344  77096.39   15
## s79    0.6764786  77094.03   15
## s80    0.6163821  77090.51   15
## s81    0.5616244  77089.24   15
## s82    0.5117312  77094.01   15
## s83    0.4662704  77105.48   15
## s84    0.4248483  77124.41   15
## s85    0.3871059  77140.84   15
## s86    0.3527165  77146.87   15
## s87    0.3213821  77148.61   15
## s88    0.2928314  77151.26   15
## s89    0.2668171  77153.49   15
## s90    0.2431138  77157.72   14
## s91    0.2215162  77161.50   14
## s92    0.2018373  77164.83   14
## s93    0.1839067  77169.46   14
## s94    0.1675689  77174.04   15
## s95    0.1526826  77178.05   15
## s96    0.1391187  77181.50   15
## s97    0.1267597  77185.06   15
## s98    0.1154988  77191.12   15
## s99    0.1052382  77195.83   15
```

```
coef(model_ent, s = model_ent$lambda.min)
```

```
## 16 x 1 sparse Matrix of class "dgCMatrix"
##                      1
## (Intercept) 891.18837
## So           40.82167
## M           100.53196
## Ed          166.34645
## Po1         242.78251
## Po2          40.08840
## LF             .
## M.F          60.83399
## Pop         -15.42574
## NW           21.86336
## U1          -75.46051
## U2          117.52461
## Wealth       57.10100
## Ineq        233.60507
## Prob        -91.92210
## Time           .
```

```r
coef(model_ent, s = model_ent$lambda.1se)
```

```
## 16 x 1 sparse Matrix of class "dgCMatrix"
##                     1
## (Intercept) 888.63603
## So           48.31917
## M            49.38293
## Ed           39.57069
## Po1         147.11644
## Po2         107.71176
## LF           14.77275
## M.F          51.54757
## Pop            .
## NW           24.95346
## U1             .
## U2           20.96674
## Wealth         .
## Ineq         74.63311
## Prob        -69.64644
## Time           .
```

```r
# Using the lambda.min model (alpha = 0.25)
# Calculate R-squared by first fitting a linear regression model using training
# data set and then making a prediction model using the test data set
elnet_model <- lm(formula = Crime ~ So + M + Ed + Po1 + Po2 + LF + M.F +
                    Pop + NW + U1 + U2 + Wealth + Ineq + Prob, data = train)
summary(elnet_model)
```

```
##
## Call:
## lm(formula = Crime ~ So + M + Ed + Po1 + Po2 + LF + M.F + Pop +
##     NW + U1 + U2 + Wealth + Ineq + Prob, data = train)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -255.161 -101.947    3.633   86.278  293.980
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)   919.88      61.12  15.051 2.13e-13 ***
## So            -89.73     156.91  -0.572 0.572970
## M              91.44      53.15   1.721 0.098748 .
## Ed            288.50      71.32   4.045 0.000503 ***
## Po1           558.22     370.51   1.507 0.145520
## Po2          -316.53     341.78  -0.926 0.363997
## LF            -35.29      54.31  -0.650 0.522300
## M.F            11.82      55.79   0.212 0.834057
## Pop           -31.88      55.64  -0.573 0.572240
## NW            100.69      73.47   1.370 0.183775
## U1            -21.32      74.06  -0.288 0.776017
## U2             82.28      69.89   1.177 0.251124
## Wealth        104.82     117.86   0.889 0.383045
```

```
## Ineq          361.57      90.29   4.005 0.000556 ***
## Prob          -55.44      42.68  -1.299 0.206777
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 177.8 on 23 degrees of freedom
## Multiple R-squared:  0.8045, Adjusted R-squared:  0.6855
## F-statistic: 6.761 on 14 and 23 DF,  p-value: 3.275e-05
```

```
eln_pred = predict.lm(elnet_model, as.data.frame(test))
eln_sse = sum((pred - test[,16]) ^ 2)
eln_sst = sum((test[,16] - mean(test[,16])) ^ 2)
1 - eln_sse / eln_sst
```

```
## [1] 0.5455591
```

```
# Using the .1se model, which is the largest value of lambda
# such that error is within 1 standard error of the minimum:
model_ent_se1 = lm(Crime ~ So + M + Po1 + Po2 + LF + M.F + NW + U2 + Ineq
                   + Prob, as.data.frame(train))
summary(model_ent_se1)
```

```
##
## Call:
## lm(formula = Crime ~ So + M + Po1 + Po2 + LF + M.F + NW + U2 +
##     Ineq + Prob, data = as.data.frame(train))
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -435.19 -126.00   24.25   98.64  379.99
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)  883.566     71.428  12.370 1.23e-12 ***
## So             5.818    179.034   0.032   0.9743
## M            104.426     67.818   1.540   0.1352
## Po1          308.980    394.336   0.784   0.4401
## Po2            3.083    389.636   0.008   0.9937
## LF            45.094     61.985   0.728   0.4732
## M.F           62.604     53.514   1.170   0.2523
## NW           -10.127     90.505  -0.112   0.9117
## U2            38.473     49.756   0.773   0.4461
## Ineq         137.192     77.845   1.762   0.0893 .
## Prob         -66.138     49.693  -1.331   0.1943
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 233.1 on 27 degrees of freedom
## Multiple R-squared:  0.6055, Adjusted R-squared:  0.4594
## F-statistic: 4.145 on 10 and 27 DF,  p-value: 0.001539
```

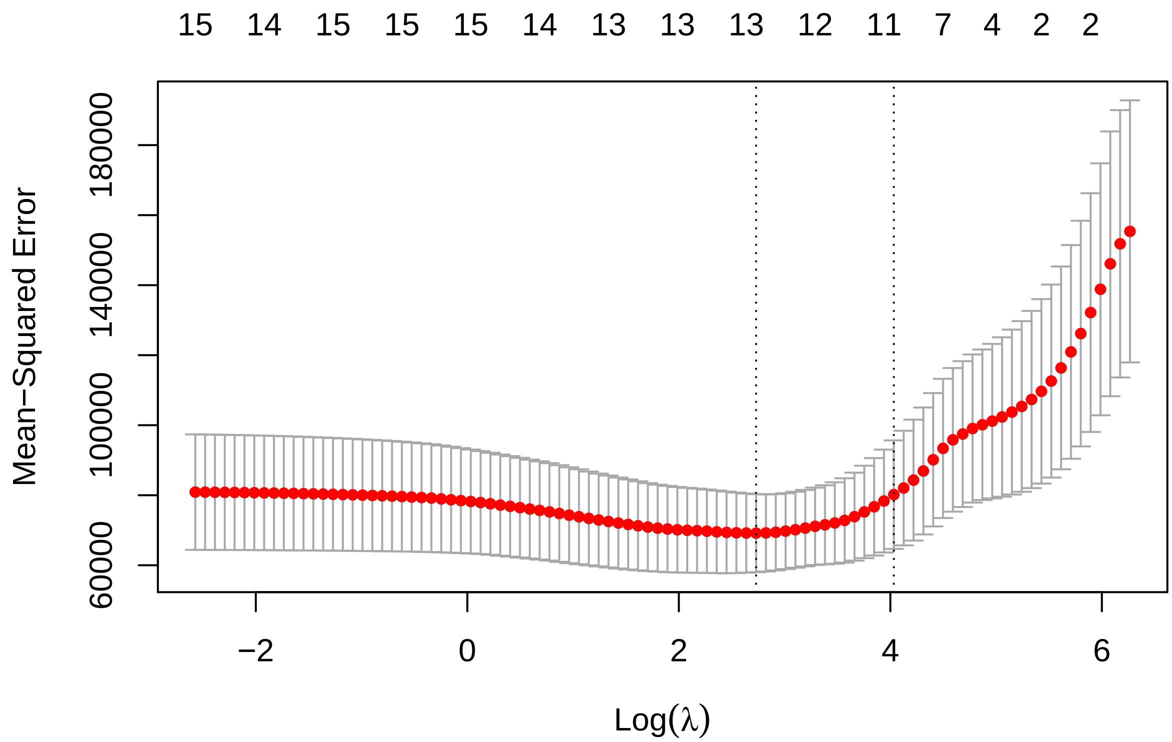
```
eln_pred2 = predict.lm(model_ent_se1, as.data.frame(test))
eln_sse2 = sum((eln_pred2 - test[,16]) ^ 2)
eln_sst2 = sum((test[,16] - mean(test[,16])) ^ 2)
1 - eln_sse2 / eln_sst2
```

```
## [1] 0.777308
```

```
# Alpha = 0.50
model_ent2 <- cv.glmnet(x = predictors,
                        y = response,
                        alpha = 0.50,
                        nfolds = 8,
                        type.measure = "mse",
                        family = "gaussian")
model_ent2
```

```
##
## Call:  cv.glmnet(x = predictors, y = response, type.measure = "mse",      nfolds = 8, alpha = 0.5, fa
##
## Measure: Mean-Squared Error
##
##     Lambda Index Measure    SE Nonzero
## min  15.34    39   69168 11148      13
## 1se  56.42    25   80167 15492       9
```

```
plot(model_ent2)
```




```
model_ent2$lambda.min
```

```
## [1] 15.33874
```

```
model_ent2$lambda.1se
```

```
## [1] 56.42171
```

```
cbind(model_ent2$lambda, model_ent2$cvm, model_ent2$nzero)
```

```
##             [,1]      [,2] [,3]
## s0  526.19079328 155355.92    0
## s1  479.44545348 151819.44    2
## s2  436.85284084 146090.96    2
## s3  398.04403851 138820.52    2
## s4  362.68290322 132167.20    2
## s5  330.46315372 126170.61    2
## s6  301.10571797 120929.45    2
## s7  274.35631589 116365.65    2
## s8  249.98325696 112600.78    2
## s9  227.77543342 109676.60    2
## s10 207.54049171 107334.60    2
## s11 189.10316644 105353.43    2
## s12 172.30376234 103748.81    3
```

```
## s13 156.99677100 102347.62    3
## s14 143.04961058 101168.63    4
## s15 130.34147745 100127.77    4
## s16 118.76229984  99053.63    6
## s17 108.21178445  97464.40    6
## s18  98.59854777  95818.40    7
## s19  89.83932455  93365.11    7
## s20  81.85824657  90137.51    7
## s21  74.58618556  86927.47    7
## s22  67.96015441  84312.14    7
## s23  61.92276160  82056.26    7
## s24  56.42171413  80166.93    9
## s25  51.40936456  78339.83   11
## s26  46.84229831  76693.69   11
## s27  42.68095763  75227.68   11
## s28  38.88929899  73881.31   11
## s29  35.43448086  72806.25   11
## s30  32.28657924  72070.56   11
## s31  29.41832853  71548.75   11
## s32  26.80488531  71116.67   12
## s33  24.42361317  70603.46   12
## s34  22.25388670  70148.19   12
## s35  20.27691274  69746.56   13
## s36  18.47556770  69414.29   13
## s37  16.83424919  69233.75   13
## s38  15.33874089  69167.94   13
## s39  13.97608942  69189.67   13
## s40  12.73449216  69263.73   13
## s41  11.60319497  69376.49   14
## s42  10.57239911  69538.78   13
## s43   9.63317632  69725.75   13
## s44   8.77739148  69866.72   13
## s45   7.99763222  69988.76   13
## s46   7.28714461  70129.29   13
## s47   6.63977477  70340.99   12
## s48   6.04991548  70590.54   12
## s49   5.51245766  70896.51   12
## s50   5.02274612  71276.44   12
## s51   4.57653920  71651.64   12
## s52   4.16997207  72055.71   13
## s53   3.79952324  72492.41   13
## s54   3.46198407  72926.24   13
## s55   3.15443095  73377.31   13
## s56   2.87420000  73854.67   13
## s57   2.61886399  74306.65   13
## s58   2.38621133  74770.02   14
## s59   2.17422689  75246.50   14
## s60   1.98107457  75657.13   14
## s61   1.80508136  76043.21   15
## s62   1.64472291  76448.77   15
## s63   1.49861026  76821.82   15
## s64   1.36547786  77173.17   15
## s65   1.24417257  77552.60   15
## s66   1.13364371  77908.53   15
```

```
## s67   1.03293392  78211.95   15
## s68   0.94117092  78459.59   15
## s69   0.85755989  78701.24   15
## s70   0.78137663  78937.72   15
## s71   0.71196129  79168.46   15
## s72   0.64871261  79328.61   15
## s73   0.59108277  79472.45   15
## s74   0.53857260  79608.44   15
## s75   0.49072730  79731.11   15
## s76   0.44713244  79832.48   15
## s77   0.40741042  79928.41   15
## s78   0.37121720  80019.01   15
## s79   0.33823929  80105.46   15
## s80   0.30819105  80186.29   15
## s81   0.28081220  80262.07   15
## s82   0.25586562  80332.64   15
## s83   0.23313522  80399.58   15
## s84   0.21242413  80460.79   14
## s85   0.19355296  80518.27   14
## s86   0.17635825  80571.75   14
## s87   0.16069107  80620.52   14
## s88   0.14641572  80666.62   14
## s89   0.13340855  80709.21   14
## s90   0.12155690  80747.82   15
## s91   0.11075812  80783.05   15
## s92   0.10091867  80816.08   15
## s93   0.09195334  80845.49   15
## s94   0.08378446  80871.52   15
## s95   0.07634128  80896.44   15
```

```
coef(model_ent2, s = model_ent2$lambda.min)
```

```
## 16 x 1 sparse Matrix of class "dgCMatrix"
##                      1
## (Intercept) 886.931351
## So           53.326656
## M            86.292769
## Ed          127.632559
## Po1         231.389529
## Po2          57.601221
## LF            5.083053
## M.F          60.187655
## Pop           .
## NW           14.672514
## U1          -41.489965
## U2           76.953274
## Wealth       16.598743
## Ineq        178.902825
## Prob        -85.594059
## Time          .
```

```
coef(model_ent2, s = model_ent2$lambda.1se)
```

```
## 16 x 1 sparse Matrix of class "dgCMatrix"
```

```
##                     1
## (Intercept) 904.440418
## So            1.893772
## M            38.182553
## Ed            6.250363
## Po1         166.932228
## Po2          91.717677
## LF            .
## M.F          54.861464
## Pop           .
## NW           17.177934
## U1            .
## U2            .
## Wealth        .
## Ineq         66.004866
## Prob        -57.294877
## Time          .
```

```r
# Using the lambda.min model (alpha = 0.50)
# Calculate R-squared by first fitting a linear regression model using training
# data set and then making a prediction model using the test data set
elnet_model2 <- lm(formula = Crime ~ So + M + Ed + Po1 + Po2 + LF + M.F +
                     Pop + NW + U1 + U2 + Wealth + Ineq + Prob, data = train)
summary(elnet_model2)
```

```
##
## Call:
## lm(formula = Crime ~ So + M + Ed + Po1 + Po2 + LF + M.F + Pop +
##     NW + U1 + U2 + Wealth + Ineq + Prob, data = train)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -255.161 -101.947    3.633   86.278  293.980
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)   919.88      61.12  15.051 2.13e-13 ***
## So            -89.73     156.91  -0.572 0.572970
## M              91.44      53.15   1.721 0.098748 .
## Ed            288.50      71.32   4.045 0.000503 ***
## Po1           558.22     370.51   1.507 0.145520
## Po2          -316.53     341.78  -0.926 0.363997
## LF            -35.29      54.31  -0.650 0.522300
## M.F            11.82      55.79   0.212 0.834057
## Pop           -31.88      55.64  -0.573 0.572240
## NW            100.69      73.47   1.370 0.183775
## U1            -21.32      74.06  -0.288 0.776017
## U2             82.28      69.89   1.177 0.251124
## Wealth        104.82     117.86   0.889 0.383045
## Ineq          361.57      90.29   4.005 0.000556 ***
## Prob          -55.44      42.68  -1.299 0.206777
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
```

```
## Residual standard error: 177.8 on 23 degrees of freedom
## Multiple R-squared:  0.8045, Adjusted R-squared:  0.6855
## F-statistic: 6.761 on 14 and 23 DF,  p-value: 3.275e-05
```

```r
eln_pred3 = predict.lm(elnet_model2, as.data.frame(test))
eln_sse3 = sum((pred - test[,16]) ^ 2)
eln_sst3 = sum((test[,16] - mean(test[,16])) ^ 2)
1 - eln_sse3 / eln_sst3
```

```
## [1] 0.5455591
```

```r
# Using the .1se model, which is the largest value of lambda
# such that error is within 1 standard error of the minimum:
model_ent_se2 = lm(Crime ~ So + M + Po1 + Po2 + LF + M.F + NW + U2 + Ineq
                   + Prob, as.data.frame(train))
summary(model_ent_se1)
```

```
##
## Call:
## lm(formula = Crime ~ So + M + Po1 + Po2 + LF + M.F + NW + U2 +
##     Ineq + Prob, data = as.data.frame(train))
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -435.19 -126.00   24.25   98.64  379.99
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)  883.566     71.428  12.370 1.23e-12 ***
## So             5.818    179.034   0.032   0.9743
## M            104.426     67.818   1.540   0.1352
## Po1          308.980    394.336   0.784   0.4401
## Po2            3.083    389.636   0.008   0.9937
## LF            45.094     61.985   0.728   0.4732
## M.F           62.604     53.514   1.170   0.2523
## NW           -10.127     90.505  -0.112   0.9117
## U2            38.473     49.756   0.773   0.4461
## Ineq         137.192     77.845   1.762   0.0893 .
## Prob         -66.138     49.693  -1.331   0.1943
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 233.1 on 27 degrees of freedom
## Multiple R-squared:  0.6055, Adjusted R-squared:  0.4594
## F-statistic: 4.145 on 10 and 27 DF,  p-value: 0.001539
```

```r
eln_pred4 = predict.lm(model_ent_se2, as.data.frame(test))
eln_sse4 = sum((eln_pred2 - test[,16]) ^ 2)
eln_sst4 = sum((test[,16] - mean(test[,16])) ^ 2)
1 - eln_sse4 / eln_sst4
```

```
## [1] 0.777308
```