WK8_HW8

Anon Skywalker

10/13/2020

Question 11.1 Stepwise, Lasso, and Elastic Net Variable Selection

Base Model

To begin, I start with our most default regression model using the uscrime data with all predictors. This will serve as a base model for which we can begin to compare our future models with that will utilize different variable selection tactics.

Base Model Stats:

1. All Predictors
2. AIC of 650
3. R-squared of: 0.353

```
set.seed(1)
lm.crime.default <- lm(Crime~., data = uscrime)
#summary(lm.crime.default)

# cross validate using 4 folds
cv.lm.default <- cv.lm(lm.crime.default, data = uscrime, m = 4, plotit = FALSE)
```

```
## Analysis of Variance Table
##
## Response: Crime
##           Df  Sum Sq Mean Sq F value  Pr(>F)
## M          1   55084   55084    1.26  0.2702
## So         1   15370   15370    0.35  0.5575
## Ed         1  905668  905668   20.72 7.7e-05 ***
## Po1        1 3076033 3076033   70.38 1.8e-09 ***
## Po2        1  153024  153024    3.50  0.0708 .
## LF         1   61134   61134    1.40  0.2459
## M.F        1  111000  111000    2.54  0.1212
## Pop        1   42649   42649    0.98  0.3309
## NW         1   14197   14197    0.32  0.5728
## U1         1    7065    7065    0.16  0.6904
## U2         1  269663  269663    6.17  0.0186 *
## Wealth     1   34748   34748    0.79  0.3795
## Ineq       1  547423  547423   12.52  0.0013 **
## Prob       1  222620  222620    5.09  0.0312 *
## Time       1   10304   10304    0.24  0.6307
## Residuals 31 1354946   43708
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
##
## fold 1
## Observations in test set: 11
##                 2   9   14   16     20   22     26    38  41   44   47
## Predicted   1474 689  780 1006 1227.8  657 1977.4 562.7 824 1121  992
## cvpred      1535 706  867 1100 1298.9  931 2043.3 602.8 757 1257 1159
## Crime       1635 856  664  946 1225.0  439 1993.0 566.0 880 1030  849
## CV residual  100 150 -203 -154  -73.9 -492  -50.3 -36.8 123 -227 -310
##
## Sum of squares = 512057    Mean square = 46551    n = 11
##
## fold 2
## Observations in test set: 12
##                 1   3    6   11   19    25      28   29  30   33   35    39
## Predicted   755.0 322  793 1161 1146 605.9 1258.48 1287 703  841  738 839.3
## cvpred      727.7 265  920 1082 1449 535.1 1219.78 1534 634  784  886 868.7
## Crime       791.0 578  682 1674  750 523.0 1216.00 1043 696 1072  653 826.0
## CV residual  63.3 313 -238  592 -699 -12.1   -3.78 -491  62  288 -233 -42.7
##
## Sum of squares = 1382466    Mean square = 115205    n = 12
##
## fold 3
## Observations in test set: 12
##                4    5    10  12   13   15  17    34   37     40    42   45
## Predicted   1791 1167 736.5 722  733  903 393 971.5  971 1131.5 326.3  617
## cvpred      1576 1021 745.1 824  912 1050 103 823.4 1392 1186.8 -85.5  848
## Crime       1969 1234 705.0 849  511  798 539 923.0  831 1151.0 542.0  455
## CV residual  393  213 -40.1  25 -401 -252 436  99.6 -561  -35.8 627.5 -393
##
## Sum of squares = 1491541    Mean square = 124295    n = 12
```

```
##
## fold 4
## Observations in test set: 12
##                  7    8   18    21   23  24    27    31   32   36   43   46
## Predicted    934.2 1362  844 774.9  958 869 279.5 388.0  808 1138 1134  827
## cvpred      1055.1 1123 1189 725.3  922 851 272.7 433.1  953  852 1324  984
## Crime        963.0 1555  929 742.0 1216 968 342.0 373.0  754 1272  823  508
## CV residual  -92.1  432 -260  16.7  294 117  69.3 -60.1 -199  420 -501 -476
##
## Sum of squares = 1065774    Mean square = 88814    n = 12
##
## Overall (Sum over all 12 folds)
##    ms
## 94720
```

```
#A function for calculating r-squared for each improved regression model
R2 <- function(ms) {

  sse <- ms*nrow(uscrime)
  sst<- sum((uscrime$Crime - mean(uscrime$Crime))^2)
  R2 <- 1 - sse/sst
  #0.615 R-SQUARED
  return(R2)

}

#calculate for r-squared for base model
R2(94720)
```

```
## [1] 0.353
```

```
#650
AIC(lm.crime.default)
```

```
## [1] 650
```

Stepwise begins here:

Now that we have a base model to compare with, we can dive into the different types of variable selection methods. Beginning with Stepwise regression, where predictive variables are either added or subtracted at each step of the selection process. To achieve stepwise efficiently, I utilized the CARET packages train() function, which sets up a grid of tuning parameters for a number of classification and regression routines, fits each model and calculates a resampling based performance measure. The method= “leapSeq” is the stepwise regression method. The trainControl is a parameter used to apply cross validation with 10 folds onto each model train() comes up with.

After running the stepwise approach, the resulting model seemed to be best fitted with 6 predictors as can be seen by the $bestTune provided by the model. The final model returned by the StepWise approach utilized the M + Ed + Po1 + U2 + Ineq + Prob predictors of which I used in the step.improv model. Comparing this to our base model, it is a improvement in model quality that is notable, with a AIC of 10 points lower. Note: While the coefficients are not the same via this approach, we standardize the method of using the stepwise/lasso/elastic net variable selection processes to take their chosen predictors and use them with lm() to be able to test their AIC and r-squared for better model comparison. This method will be applied to each model.

Stepwise Model Results:

```
1. Preds in improv model: M + Ed + Po1 + U2 + Ineq + Prob
2. AIC = 640
3. R-Squared = 0.671
```

```
set.seed(1)
# Set up repeated k-fold cross-validation
train.control <- trainControl(method = "cv", number =10)
#"leapSeq", to fit linear regression with stepwise selection
step.model <- train(Crime~., data = uscrime, method = 'leapSeq',
                    tuneGrid = data.frame(nvmax = 1:8), trControl = train.control)

#r-Squared of 0.61 for 6 vars!
step.model$results
```

```
##   nvmax RMSE Rsquared MAE RMSESD RsquaredSD MAESD
## 1     1  271    0.584 226  110.5      0.337  96.7
## 2     2  254    0.606 202   84.8      0.281  64.1
## 3     3  239    0.703 186   88.7      0.101  59.2
## 4     4  276    0.664 216  104.7      0.251  79.0
## 5     5  252    0.565 207   80.6      0.280  62.7
## 6     6  233    0.785 185  102.3      0.148  81.6
## 7     7  240    0.729 191   95.4      0.182  74.0
## 8     8  259    0.699 203  112.0      0.203  90.3
```

```
#best # of variables
step.model$bestTune
```

```
##   nvmax
## 6     6
```

```
#best model appears to be with 6 predictors!
step.final <- step.model$finalModel


#* = Significance. Model Chose: M + Ed  Po1  M.F U1  U2 Ineq  Prob
summary(step.final)
```

```
## Subset selection object
## 15 Variables  (and intercept)
##        Forced in Forced out
## M          FALSE      FALSE
## So         FALSE      FALSE
## Ed         FALSE      FALSE
## Po1        FALSE      FALSE
## Po2        FALSE      FALSE
## LF         FALSE      FALSE
## M.F        FALSE      FALSE
## Pop        FALSE      FALSE
## NW         FALSE      FALSE
## U1         FALSE      FALSE
## U2         FALSE      FALSE
## Wealth     FALSE      FALSE
## Ineq       FALSE      FALSE
## Prob       FALSE      FALSE
## Time       FALSE      FALSE
## 1 subsets of each size up to 6
## Selection Algorithm: 'sequential replacement'
##          M   So  Ed  Po1 Po2 LF  M.F Pop NW  U1  U2  Wealth Ineq Prob Time
## 1  ( 1 ) " " " " " " "*" " " " " " " " " " " " " " " " "    " "  " "  " "
## 2  ( 1 ) " " " " " " "*" " " " " " " " " " " " " " " " "    "*"  " "  " "
## 3  ( 1 ) " " " " "*" "*" " " " " " " " " " " " " " " " "    "*"  " "  " "
## 4  ( 1 ) "*" "*" "*" "*" " " " " " " " " " " " " " " " "    " "  " "  " "
## 5  ( 1 ) "*" " " "*" "*" " " " " " " " " " " " " " " " "    "*"  "*"  " "
## 6  ( 1 ) "*" " " "*" "*" " " " " " " " " " " " " "*" " "    "*"  "*"  " "
```

```
#display final stepwise model coefficients
coef(step.final, 6)
```

```
## (Intercept)           M          Ed         Po1          U2        Ineq 
##     -5040.5       105.0       196.5       115.0        89.4        67.7 
##        Prob 
##     -3801.8
```

```
#create LM model with newfound variable combination
step.improv <- lm(Crime~M + Ed + Po1 + U2 + Ineq + Prob, data = uscrime)
summary(step.improv)
```

```
##
## Call:
## lm(formula = Crime ~ M + Ed + Po1 + U2 + Ineq + Prob, data = uscrime)
##
## Residuals:
##    Min     1Q Median     3Q    Max
## -470.7  -78.4  -19.7  133.1  556.2
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)  -5040.5      899.8   -5.60  1.7e-06 ***
## M              105.0       33.3    3.15   0.0031 **
## Ed             196.5       44.8    4.39  8.1e-05 ***
## Po1            115.0       13.8    8.36  2.6e-10 ***
## U2              89.4       40.9    2.18   0.0348 *
## Ineq            67.7       13.9    4.85  1.9e-05 ***
## Prob         -3801.8     1528.1   -2.49   0.0171 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 201 on 40 degrees of freedom
## Multiple R-squared:  0.766,  Adjusted R-squared:  0.731
## F-statistic: 21.8 on 6 and 40 DF,  p-value: 3.42e-11
```

```
# ms of 48203
cv.lm.step <- cv.lm(step.improv, data = uscrime, m=4, plotit = FALSE)
```

```
## Analysis of Variance Table
##
## Response: Crime
##           Df  Sum Sq Mean Sq F value  Pr(>F)
## M          1   55084   55084    1.37 0.24914
## Ed         1  725967  725967   18.02 0.00013 ***
## Po1        1 3173852 3173852   78.80 5.3e-11 ***
## U2         1  217386  217386    5.40 0.02534 *
## Ineq       1  848273  848273   21.06 4.3e-05 ***
## Prob       1  249308  249308    6.19 0.01711 *
## Residuals 40 1611057   40276
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
##
## fold 1
## Observations in test set: 11
##                 2   9    14     16     20   22   26    38  41   44   47
## Predicted    1388 719 713.6 1004.4 1203.0  728 1789 544.4 796 1178  976
## cvpred       1355 731 731.1 1023.2 1187.6  771 1720 588.4 763 1150  970
## Crime        1635 856 664.0  946.0 1225.0  439 1993 566.0 880 1030  849
## CV residual  280 125 -67.1  -77.2   37.4 -332  273 -22.4 117 -120 -121
##
## Sum of squares = 334042    Mean square = 30367    n = 11
##
## fold 2
## Observations in test set: 12
##                  1   3    6   11   19    25     28   29    30   33   35    39
## Predicted    810.8 386  730 1118 1221 579.1 1259.0 1495 668.0  874  808 786.7
## cvpred       716.9 296  888 1241 1363 504.3 1208.7 1711 614.2  792  919 736.6
## Crime        791.0 578  682 1674  750 523.0 1216.0 1043 696.0 1072  653 826.0
## CV residual  74.1 282 -206  433 -613  18.7    7.3 -668  81.8  280 -266  89.4
##
## Sum of squares = 1300449    Mean square = 108371    n = 12
##
## fold 3
## Observations in test set: 12
##                   4      5    10  12   13  15    17    34   37     40  42   45
## Predicted    1897.2 1269.8 787.3 673  739 828 527.4 997.5  992 1140.8 369  622
## cvpred       1916.6 1282.8 791.8 680  778 867 483.3 998.2 1037 1190.7 317  656
## Crime        1969.0 1234.0 705.0 849  511 798 539.0 923.0  831 1151.0 542  455
## CV residual   52.4  -48.8 -86.8 169 -267 -69  55.7 -75.2 -206  -39.7 225 -201
##
## Sum of squares = 261503    Mean square = 21792    n = 12
##
## fold 4
## Observations in test set: 12
##                 7    8  18  21   23    24    27  31  32   36   43   46
## Predicted    733 1354 800 783  938 919.4 312.2 440 774 1102 1017  748
## cvpred       708 1319 771 759  909 896.3 316.2 426 740 1093 1027  723
## Crime        963 1555 929 742 1216 968.0 342.0 373 754 1272  823  508
## CV residual 255  236 158 -17  307  71.7  25.8 -53  14  179 -204 -215
##
```

```
## Sum of squares = 369549    Mean square = 30796    n = 12
##
## Overall (Sum over all 12 folds)
##    ms
## 48203
```

```
#Calculate Rsquared for Stepwise model
R2(48203)
```

```
## [1] 0.671
```

```
#compare AICs between default and improv model: improv wins!
AIC(step.improv)
```

```
## [1] 640
```

Lasso Model

To begin with the lasso model, it is vital that the data is scaled due to the glmnet's need for matrix type data in the x and y parameters. This presented the challenge of not destroying the response column "Crime" and categorical column "So". To prevent this issue, I scaled all columns except for these two and then added them back in. Because of this, the parameter "standardize" was not used in my cv.glmnet model. The lasso approach requires finding the best value of lambda, which I found by both plotting and assessing the model itself. The lasso model chooses 6 more predictors than the stepwise approach. Yet, the predictors chosen tend to show less significance than the original 6, and removing them would just leave you back at where stepwise was with an AIC of 640, 8 points lower (better) than the current LASSO results at an AIC of 648. Comparing this to our previous stepwise and base models, it would appear that the lasso approach is only slightly better than the base model, while being 8 points in AIC worse than the stepwise model.

Lasso Model Results:

```
1. Preds: All but PO2, LF, and Time. 12 predictors, seems high!
2. AIC = 648
3. R-Squared = 0.615
4. Alpha = 1, Lambda = 6
```

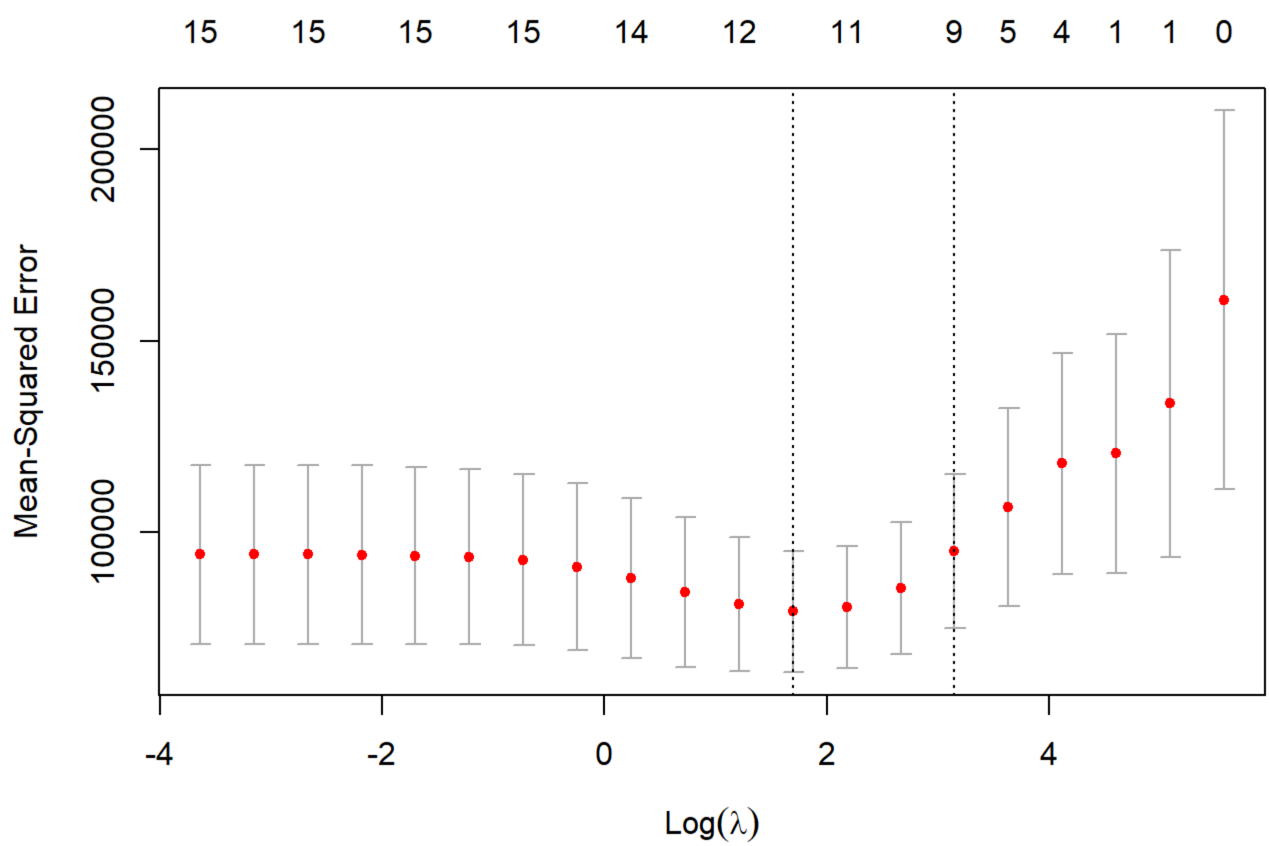
```
set.seed(1)
#scale all data except categorical So and response
uscrime.scaled = as.data.frame(scale(uscrime[ , -c(2,16)]))
uscrime.scaled <- cbind(uscrime[,2], uscrime.scaled, uscrime$Crime)
# re-insert column for So and Crime
colnames(uscrime.scaled)[1] <- "So"
colnames(uscrime.scaled)[16] <- "Crime"


scaled.matrix = data.matrix(uscrime.scaled[,-16])
crime.matrix = data.matrix(uscrime.scaled$Crime)



lasso.model = cv.glmnet( x = scaled.matrix,
                         y = crime.matrix,
                         alpha = 1,
                         nfolds = 6,
                         nlambda = 20 ,
                         type.measure = "mse",
                         family = "gaussian")

lasso.model
```

```
##
## Call:  cv.glmnet(x = scaled.matrix, y = crime.matrix, type.measure = "mse",      nfolds = 6,
alpha = 1, nlambda = 20, family = "gaussian")
##
## Measure: Mean-Squared Error
##
##     Lambda Measure    SE Nonzero
## min   5.44   79430 15855      12
## 1se  23.31   95141 20106       9
```

```
summary(lasso.model)
```

```
##            Length Class  Mode
## lambda     20     -none- numeric
## cvm        20     -none- numeric
## cvsd       20     -none- numeric
## cvup       20     -none- numeric
## cvlo       20     -none- numeric
## nzero      20     -none- numeric
## call        8     -none- call
## name        1     -none- character
## glmnet.fit 12     elnet  list
## lambda.min  1     -none- numeric
## lambda.1se  1     -none- numeric
```

```
#plot mean squared error vs lambda. Helps determine which lambda is the best to use.
#appears to be close to a value of 6.
plot(lasso.model)
```




```
#Run again to test those coefficients - a few that were low or negative values
lasso.improv <- lm(Crime~So+M+Ed+Po1+M.F+U2+Ineq, data = uscrime.scaled)
summary(lasso.improv)
```

```
##
## Call:
## lm(formula = Crime ~ So + M + Ed + Po1 + M.F + U2 + Ineq, data = uscrime.scaled)
##
## Residuals:
##    Min     1Q Median     3Q    Max
## -441.1  -96.8  -23.6  106.8  623.0
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)    911.4       49.4   18.44  < 2e-16 ***
## So             -18.5      111.6   -0.17  0.86917
## M              122.5       48.5    2.53  0.01570 *
## Ed             200.0       65.5    3.05  0.00408 **
## Po1            372.3       44.1    8.44  2.5e-10 ***
## M.F             26.7       40.2    0.66  0.51002
## U2              70.0       39.0    1.80  0.08022 .
## Ineq           250.8       66.6    3.76  0.00055 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 217 on 39 degrees of freedom
## Multiple R-squared:  0.733,  Adjusted R-squared:  0.685
## F-statistic: 15.3 on 7 and 39 DF,  p-value: 1.93e-09
```

```
# cross validate using 4 folds. MS: 56355
cv.lm.lasso <- cv.lm(lasso.improv, data = uscrime.scaled, m = 4, plotit = FALSE)
```

```
## Analysis of Variance Table
##
## Response: Crime
##           Df  Sum Sq Mean Sq F value  Pr(>F)
## So         1   56527   56527    1.20 0.27978
## M          1   13927   13927    0.30 0.58951
## Ed         1  905668  905668   19.25 8.5e-05 ***
## Po1        1 3076033 3076033   65.37 7.3e-10 ***
## M.F        1  209271  209271    4.45 0.04143 *
## U2         1  117573  117573    2.50 0.12202
## Ineq       1  666809  666809   14.17 0.00055 ***
## Residuals 39 1835119   47054
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
##
## fold 1
## Observations in test set: 11
##                 2     9  14     16     20   22   26    38  41     44   47
## Predicted   1355 750.3 711  977.5 1196.9  796 1926 576.1 716 1136.3 1061
## cvpred      1335 762.5 728 1022.5 1205.9  844 1858 603.1 674 1119.8 1047
## Crime       1635 856.0 664  946.0 1225.0  439 1993 566.0 880 1030.0  849
## CV residual  300  93.5 -64  -76.5   19.1 -405  135 -37.1 206  -89.8 -198
##
## Sum of squares = 381988    Mean square = 34726    n = 11
##
## fold 2
## Observations in test set: 12
##                 1   3    6   11   19    25   28   29    30   33   35  39
## Predicted   863.5 459  745 1051 1136 643.4 1233 1484 719.6  823  720 718
## cvpred      843.8 357  866 1166 1355 613.8 1202 1725 659.8  752  861 685
## Crime       791.0 578  682 1674  750 523.0 1216 1043 696.0 1072  653 826
## CV residual -52.8 221 -184  508 -605 -90.8   14 -682  36.2  320 -208 141
##
## Sum of squares = 1351124    Mean square = 112594    n = 12
##
## fold 3
## Observations in test set: 12
##                4      5   10  12   13    15    17     34    37     40    42
## Predicted   1844 1270.5  804 611  706 857.3 621.5  998.9 915.4 1075.9 492.4
## cvpred      1818 1259.1  818 622  732 889.7 633.8 1005.2 925.9 1089.5 512.4
## Crime       1969 1234.0  705 849  511 798.0 539.0  923.0 831.0 1151.0 542.0
## CV residual  151  -25.1 -113 227 -221 -91.7 -94.8  -82.2 -94.9   61.5  29.6
##               45
## Predicted    583
## cvpred       628
## Crime        455
## CV residual -173
##
## Sum of squares = 204236    Mean square = 17020    n = 12
##
## fold 4
## Observations in test set: 12
```

```
##               7    8   18    21   23     24    27   31    32   36   43   46
## Predicted   707 1326 1118 692.3  864 938.32 274.8  467 764.5  981 1020  809
## cvpred      708 1280 1179 671.4  823 959.63 317.5  623 704.3 1002 1078  890
## Crime       963 1555  929 742.0 1216 968.00 342.0  373 754.0 1272  823  508
## CV residual 255  275 -250  70.6  393   8.37  24.5 -250  49.7  270 -255 -382
##
## Sum of squares = 711356    Mean square = 59280    n = 12
##
## Overall (Sum over all 12 folds)
##    ms
## 56355
```

```
#615
R2(56355)
```

```
## [1] 0.615
```

```
#648
AIC(lasso.improv)
```

```
## [1] 648
```

Elastic Net Model

For the elastic net approach, the challenge was to find the best value of alpha as well. To do this, I made a basic for-loop to run through the model for different values of alpha. This resulted in an alpha value of 0.42 That I then used to find the value of lambda by creating another elastic model with the best chosen value of alpha. This value came out to be a lambda of 13. The resulting r-squared ended up being 0.42 which should cause for some suspicion. Upon further digging, I have come to believe that the insignificant p-values are the reason behind this low r-squared score. If one were to include only the significant values based off of the p-value significance, you would likely end up with a r-squared closer to what was achieved in lasso or stepwise.

In summary, the best approach for this data set is stepwise! It is also likely that the elastic/lasso models are overfit due to the number of predictors that was chosen by for each approach compared to the data set size. Despite this, it may be that the Lasso and Elastic net methods perform better when predicting a new point/city despite the current model results. One would be able to verify this by splitting the data into test and training sets and then using it to predict on a new point as we have done in past lessons, but this is beyond the scope of this homework. Additionally, I wanted to use as much data as possible for the variable selection process due to the relatively small dataset we are working with.

Elastic Net Results:

```
1. Preds: So+M+Ed+Po1+M.F+Pop+NW+U1+U2+Wealth+Ineq+Prob. Everything but Time. Far more than the
previous approaches
2. AIC = 645
3. R-squared = 0.477
4. Alpha = 0.42 and Lambda = 13
```

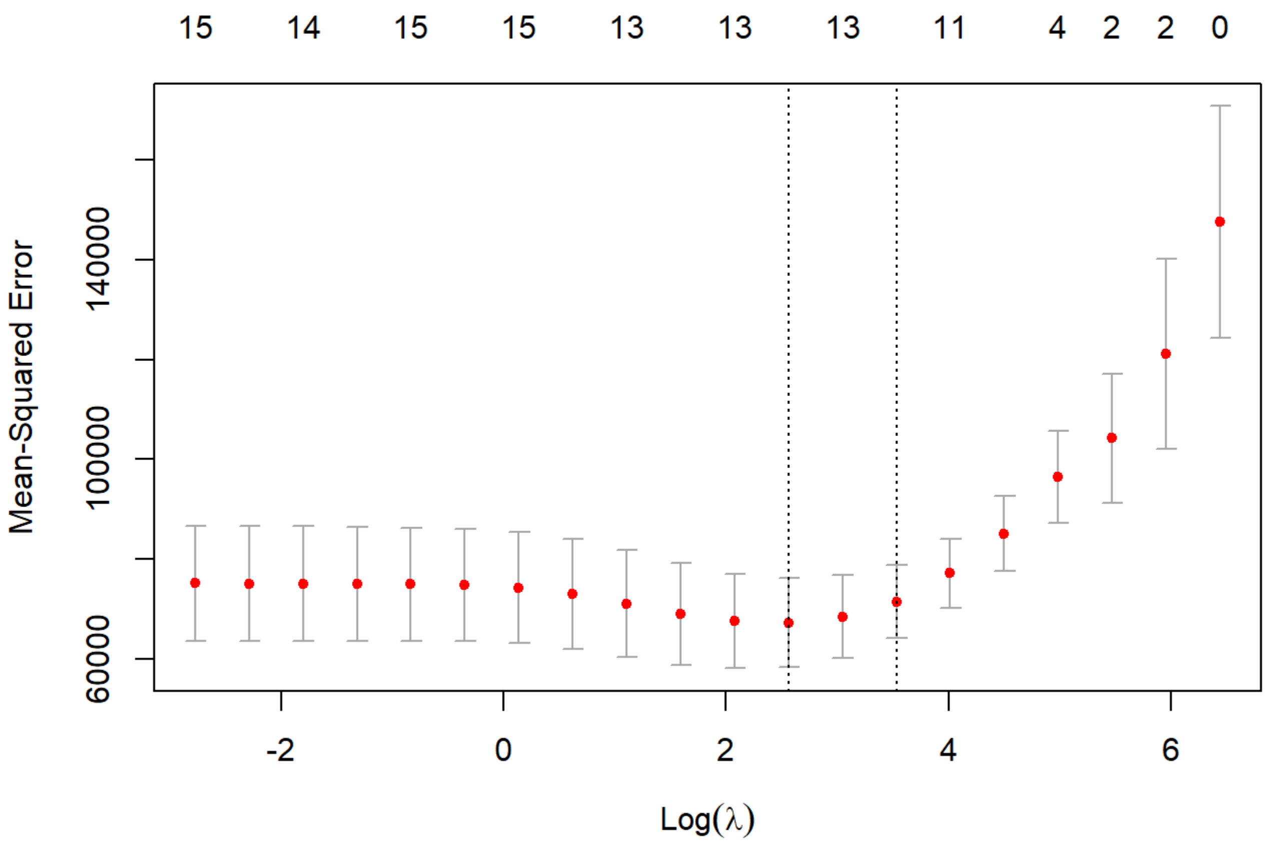
```r
set.seed(1)
r.squared <- c()

for (i in 0:100) {

  elastic.model <- cv.glmnet( x = scaled.matrix,
                             y = crime.matrix,
                             alpha = i/100,
                             nfolds = 6 ,
                             type.measure = "mse",
                             family = "gaussian")

  fit.select = which(elastic.model$glmnet.fit$lambda == elastic.model$lambda.min)
  #Calc for r-squared
  r.squared <- cbind(r.squared, elastic.model$glmnet.fit$dev.ratio[fit.select])

}
r.squared
```

```
##       [,1]  [,2]  [,3] [,4]  [,5]  [,6]  [,7]  [,8]  [,9] [,10] [,11] [,12]
## [1,] 0.774 0.708 0.766 0.76 0.748 0.769 0.755 0.761 0.771 0.745 0.765 0.748
##      [,13] [,14] [,15] [,16] [,17] [,18] [,19] [,20] [,21] [,22] [,23] [,24]
## [1,] 0.729 0.766 0.747 0.766 0.764 0.766 0.767 0.769  0.75 0.785 0.766 0.759
##      [,25] [,26] [,27] [,28] [,29] [,30] [,31] [,32] [,33] [,34] [,35] [,36]
## [1,] 0.779 0.741 0.758 0.767 0.756 0.765 0.792 0.776 0.774 0.772 0.765 0.731
##      [,37] [,38] [,39] [,40] [,41] [,42] [,43] [,44] [,45] [,46] [,47] [,48]
## [1,] 0.779  0.74 0.771 0.785 0.755 0.689 0.803 0.766 0.783 0.789 0.764 0.795
##      [,49] [,50] [,51] [,52] [,53] [,54] [,55] [,56] [,57] [,58] [,59] [,60]
## [1,] 0.781 0.749 0.794 0.719 0.783  0.79   0.7 0.721 0.781 0.788 0.764 0.769
##      [,61] [,62] [,63] [,64] [,65] [,66] [,67] [,68] [,69] [,70] [,71] [,72]
## [1,] 0.774 0.716  0.78 0.778 0.788 0.767  0.75 0.779 0.772 0.793 0.752 0.721
##      [,73] [,74] [,75] [,76] [,77] [,78] [,79] [,80] [,81] [,82] [,83] [,84]
## [1,]  0.73 0.759 0.712 0.789 0.796 0.748 0.789 0.792 0.734 0.792 0.789 0.793
##      [,85] [,86] [,87] [,88] [,89] [,90] [,91] [,92] [,93] [,94] [,95] [,96]
## [1,] 0.792 0.749 0.762 0.762 0.775 0.775 0.767 0.763 0.793 0.664 0.784 0.763
##      [,97] [,98] [,99] [,100] [,101]
## [1,] 0.787 0.751 0.682  0.768  0.776
```

```r
#now to determine the best value for alpha
alpha.best <- (which.max(r.squared)-1) /100

alpha.best
```

```
## [1] 0.42
```

```
elastic.model <- cv.glmnet( x = scaled.matrix,
                            y = crime.matrix,
                            alpha = alpha.best,
                            nfolds = 6 ,
                            nlambda = 20,
                            type.measure = "mse",
                            family = "gaussian")

plot(elastic.model)
```




```
#13 seems to be the best value of lambda according to the model/plot
elastic.model
```

```
##
## Call:  cv.glmnet(x = scaled.matrix, y = crime.matrix, type.measure = "mse",      nfolds = 6,
alpha = alpha.best, nlambda = 20, family = "gaussian")
##
## Measure: Mean-Squared Error
##
##     Lambda Measure   SE Nonzero
## min   13.0   67336 8853      14
## 1se   34.2   71561 7290      12
```

```
#selects 13 compared, much more than the previous lasso and stepwise approach
coefficients(elastic.model, s=elastic.model$lambda.min)
```

```
## 16 x 1 sparse Matrix of class "dgCMatrix"
##                     1
## (Intercept)  8.88e+02
## So           5.09e+01
## M            9.16e+01
## Ed           1.41e+02
## Po1          2.27e+02
## Po2          5.66e+01
## LF           3.06e+00
## M.F          6.35e+01
## Pop         -6.68e-04
## NW           1.72e+01
## U1          -5.46e+01
## U2           9.13e+01
## Wealth       3.00e+01
## Ineq         1.95e+02
## Prob        -8.72e+01
## Time         .
```

```
elastic.model.improv <- lm(Crime~So+M+Ed+Po1+M.F+Pop+NW+U1+U2+Wealth+Ineq+Prob, data = uscrime)
summary(elastic.model.improv)
```

```
##
## Call:
## lm(formula = Crime ~ So + M + Ed + Po1 + M.F + Pop + NW + U1 +
##     U2 + Wealth + Ineq + Prob, data = uscrime)
##
## Residuals:
##    Min     1Q Median     3Q    Max
## -434.2 -107.0   18.6  115.9  470.3
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept) -6.39e+03   1.41e+03   -4.52  7.1e-05 ***
## So           2.29e+01   1.25e+02    0.18   0.8562
## M            8.97e+01   3.93e+01    2.28   0.0288 *
## Ed           1.75e+02   5.63e+01    3.11   0.0038 **
## Po1          9.87e+01   2.19e+01    4.51  7.3e-05 ***
## M.F          1.66e+01   1.63e+01    1.02   0.3166
## Pop         -8.74e-01   1.20e+00   -0.73   0.4711
## NW           1.86e+00   5.61e+00    0.33   0.7419
## U1          -4.98e+03   3.64e+03   -1.37   0.1807
## U2           1.67e+02   7.91e+01    2.11   0.0424 *
## Wealth       8.63e-02   9.90e-02    0.87   0.3893
## Ineq         7.16e+01   2.14e+01    3.35   0.0020 **
## Prob        -4.08e+03   1.81e+03   -2.26   0.0307 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 203 on 34 degrees of freedom
## Multiple R-squared:  0.797,  Adjusted R-squared:  0.726
## F-statistic: 11.1 on 12 and 34 DF,  p-value: 1.52e-08
```

```
# cross validate using 4 folds
cv.lm.elastic <- cv.lm(elastic.model.improv, data = uscrime, m = 4, plotit = FALSE)
```

```
## Analysis of Variance Table
##
## Response: Crime
##           Df  Sum Sq Mean Sq F value  Pr(>F)
## So         1   56527   56527    1.38 0.24880
## M          1   13927   13927    0.34 0.56412
## Ed         1  905668  905668   22.06 4.2e-05 ***
## Po1        1 3076033 3076033   74.92 4.1e-10 ***
## M.F        1  209271  209271    5.10 0.03050 *
## Pop        1   66764   66764    1.63 0.21088
## NW         1   15839   15839    0.39 0.53866
## U1         1      17      17    0.00 0.98400
## U2         1  299228  299228    7.29 0.01074 *
## Wealth     1   43104   43104    1.05 0.31277
## Ineq       1  589797  589797   14.37 0.00059 ***
## Prob       1  208855  208855    5.09 0.03065 *
## Residuals 34 1395898   41056
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
##
## fold 1
## Observations in test set: 11
##                 2   9   14   16   20   22     26    38  41   44   47
## Predicted   1442 705  770  991 1224  659 1942.9 539.0 785 1189 1063
## cvpred      1517 711  843 1088 1343  829 2061.4 594.2 754 1235 1188
## Crime       1635 856  664  946 1225  439 1993.0 566.0 880 1030  849
## CV residual  118 145 -179 -142 -118 -390  -68.4 -28.2 126 -205 -339
##
## Sum of squares = 431883    Mean square = 39262    n = 11
##
## fold 2
## Observations in test set: 12
##                  1   3    6   11   19    25     28   29  30   33   35    39
## Predicted   762.9 345  790 1204 1184 592.6 1236.1 1310 677  888  698 790.8
## cvpred      718.9 316  892 1207 1464 545.6 1167.2 1648 588  835  881 779.3
## Crime       791.0 578  682 1674  750 523.0 1216.0 1043 696 1072  653 826.0
## CV residual  72.1 262 -210  467 -714 -22.6   48.8 -605 108  237 -228  46.7
##
## Sum of squares = 1337662    Mean square = 111472    n = 12
##
## fold 3
## Observations in test set: 12
##                 4    5    10    12   13   15  17    34   37     40  42   45
## Predicted   1796 1145 753.3 730.5  746  919 393 996.6 1026 1101.6 296  629
## cvpred      1646  970 658.8 757.7  761 1107 165 949.4 1095 1138.2 103  808
## Crime       1969 1234 705.0 849.0  511  798 539 923.0  831 1151.0 542  455
## CV residual  323  264  46.2  91.3 -250 -309 374 -26.4 -264   12.8 439 -353
##
## Sum of squares = 870267    Mean square = 72522    n = 12
##
## fold 4
## Observations in test set: 12
```

```
##                 7    8   18    21   23  24    27   31   32   36   43   46
## Predicted   889.6 1354  863 789.3  954 855 306.5  412  774 1125 1107  789
## cvpred      876.5 1061 1193 771.1  921 830 312.4  517  881  895 1245  923
## Crime       963.0 1555  929 742.0 1216 968 342.0  373  754 1272  823  508
## CV residual  86.5  494 -264 -29.1  295 138  29.6 -144 -127  377 -422 -415
##
## Sum of squares = 957736    Mean square = 79811    n = 12
##
## Overall (Sum over all 12 folds)
##    ms
## 76544
```

```
AIC(elastic.model.improv)
```

```
## [1] 645
```

```
#0.477 R-SQUARED
R2(76544)
```

```
## [1] 0.477
```