The significant variable in elastic net regression is M, Ed, Po1, U2, Ineq, Prob, and adjusted r2 is 0.7078. The r2 for cross-validation is 0.6662.

In conclusion, Variable selection seems affect the results of model greatly. Besides, removing insignicant variables can also greatly affect the model quality.

hw8_revised.R

zhuoxun.yang001

2022-10-20

```r
# import required package
library(caret)
library(dplyr)
library(leaps)
library(tidyverse)
library(glmnet)
library(elasticnet)
# import data assign to variable df
df <- read.delim("C:/Users/zhuoxun.yang001/Desktop/hw8-SP22 (1)/data 11.1/uscrime.txt")
# check the structure of df
str(df)
```

```
## 'data.frame':    47 obs. of  16 variables:
##  $ M     : num  15.1 14.3 14.2 13.6 14.1 12.1 12.7 13.1 15.7 14 ...
##  $ So    : int  1 0 1 0 0 0 1 1 1 0 ...
##  $ Ed    : num  9.1 11.3 8.9 12.1 12.1 11 11.1 10.9 9 11.8 ...
##  $ Po1   : num  5.8 10.3 4.5 14.9 10.9 11.8 8.2 11.5 6.5 7.1 ...
##  $ Po2   : num  5.6 9.5 4.4 14.1 10.1 11.5 7.9 10.9 6.2 6.8 ...
##  $ LF    : num  0.51 0.583 0.533 0.577 0.591 0.547 0.519 0.542 0.553 0.632 ...
##  $ M.F   : num  95 101.2 96.9 99.4 98.5 ...
##  $ Pop   : int  33 13 18 157 18 25 4 50 39 7 ...
##  $ NW    : num  30.1 10.2 21.9 8 3 4.4 13.9 17.9 28.6 1.5 ...
##  $ U1    : num  0.108 0.096 0.094 0.102 0.091 0.084 0.097 0.079 0.081 0.1 ...
##  $ U2    : num  4.1 3.6 3.3 3.9 2 2.9 3.8 3.5 2.8 2.4 ...
##  $ Wealth: int  3940 5570 3180 6730 5780 6890 6200 4720 4210 5260 ...
##  $ Ineq  : num  26.1 19.4 25 16.7 17.4 12.6 16.8 20.6 23.9 17.4 ...
##  $ Prob  : num  0.0846 0.0296 0.0834 0.0158 0.0414 ...
##  $ Time  : num  26.2 25.3 24.3 29.9 21.3 ...
##  $ Crime : int  791 1635 578 1969 1234 682 963 1555 856 705 ...
```

```r
# check the head of df
head(df)
```

```
##      M So   Ed  Po1  Po2    LF   M.F Pop   NW    U1  U2 Wealth Ineq     Prob    Time Crime
## 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 26.2011   791
## 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 25.2999  1635
## 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 24.3006   578
## 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 29.9012  1969
## 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 21.2998  1234
## 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 20.9995   682
```

```
########Stepwise regression########
#
library(dplyr)
df_scaled <- df %>%
  mutate_at(c(1,3,4,5,6,7,8,9,10,11,12,13,14,15), funs(c(scale(.))))
# check the head of dataframe
head(df_scaled)
```

```
##            M So         Ed        Po1        Po2         LF         M.F         Pop           NW          U1
U2     Wealth
## 1  0.9886930  1 -1.3085099 -0.9085105 -0.8666988 -1.2667456 -1.12060499 -0.09500679  1.943738564  0.69510600
0.8313680 -1.3616094
## 2  0.3521372  0  0.6580587  0.6056737  0.5280852  0.5396568  0.98341752 -0.62033844  0.008483424  0.02950365
0.2393332  0.3276683
## 3  0.2725678  1 -1.4872888 -1.3459415 -1.2958632 -0.6976051 -0.47582390 -0.48900552  1.146296747 -0.08143007 -
0.1158877 -2.1492481
## 4 -0.2048491  0  1.3731746  2.1535064  2.1732150  0.3911854  0.37257228  3.16204944 -0.205464381  0.36230482
0.5945541  1.5298536
## 5  0.1929983  0  1.3731746  0.8075649  0.7426673  0.7376187  0.06714965 -0.48900552 -0.691709391 -0.24783066 -
1.6551781  0.5453053
## 6 -1.3983912  0  0.3898903  1.1104017  1.2433590 -0.3511718 -0.64550313 -0.30513945 -0.555560788 -0.63609870 -
0.5895155  1.6956723
##         Ineq       Prob        Time Crime
## 1  1.6793638  1.6497631 -0.05599367   791
## 2  0.0000000 -0.7693365 -0.18315796  1635
## 3  1.4036474  1.5969416 -0.32416470   578
## 4 -0.6767585 -1.3761895  0.46611085  1969
## 5 -0.5013026 -0.2503580 -0.74759413  1234
## 6 -1.7044289 -0.5669349 -0.78996812   682
```

```
# set seed
set.seed(9876)
```

```r
# using regsubsets function to setup model
models <- regsubsets(Crime~., data = df_scaled, nvmax = 15, method = "seqrep")
summary(models)
```

```
## Subset selection object
## Call: regsubsets.formula(Crime ~ ., data = df_scaled, nvmax = 15, method = "seqrep")
## 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 15
## 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 )  "*" " " "*" "*" " " " " " " " " " " " " "*" " "    "*"  "*"  " " 
## 7  ( 1 )  "*" " " "*" "*" " " " " " " " " " " " " "*" "*"    "*"  "*"  " " 
## 8  ( 1 )  "*" " " "*" "*" " " " " "*" " " " " "*" "*" " "    "*"  "*"  " " 
## 9  ( 1 )  "*" " " "*" "*" " " " " "*" " " " " "*" "*" "*"    "*"  "*"  " " 
## 10  ( 1 ) "*" "*" "*" "*" "*" "*" "*" "*" "*" "*" " " " "    " "  " "  " " 
## 11  ( 1 ) "*" " " "*" "*" "*" " " "*" "*" " " "*" "*" "*"    "*"  "*"  " " 
## 12  ( 1 ) "*" " " "*" "*" "*" " " "*" "*" "*" "*" "*" "*"    "*"  "*"  " " 
## 13  ( 1 ) "*" " " "*" "*" "*" "*" "*" "*" "*" "*" "*" "*"    "*"  "*"  " " 
## 14  ( 1 ) "*" " " "*" "*" "*" "*" "*" "*" "*" "*" "*" "*"    "*"  "*"  "*" 
## 15  ( 1 ) "*" "*" "*" "*" "*" "*" "*" "*" "*" "*" "*" "*"    "*"  "*"  "*"
```

```r
res.sum <- summary(models)
data.frame(
  Adj.R2 = which.max(res.sum$adjr2),
  CP = which.min(res.sum$cp),
  BIC = which.min(res.sum$bic)
)# build up metrics
```

```
##   Adj.R2 CP BIC
## 1      8  6   6
```

```r
### key takeaway here is according to the metrics, the model with highest r_square
### is model 8, but cp & BIC score suggest model 6 is best model.It required more
### code to choose the optimal model. But I will choose model 6 prudently. Besides,
### i suggest another methods.
# set up repeated k-fold & step.model

# set up repeated k-fold & step.model
train.control <- trainControl(method = 'repeatedcv', number = 10, repeats = 5)
step.model <- train(Crime ~., data = df_scaled,
                    method = "leapSeq",
                    tuneGrid = data.frame(nvmax = 1:15),
                    trControl = train.control
)
step.model$results
```

```
##    nvmax     RMSE  Rsquared      MAE    RMSESD RsquaredSD     MAESD
## 1      1 278.6080 0.5920004 228.8714 106.65507  0.3015803  94.23435
## 2      2 268.6844 0.5926224 217.5776 123.77217  0.3284238 102.65366
## 3      3 242.9004 0.6354933 194.6751 104.08904  0.2976705  84.04550
## 4      4 273.2774 0.5653649 217.8379  94.33497  0.3157835  75.83168
## 5      5 249.4396 0.6116294 203.7182  91.95104  0.2734303  74.41041
## 6      6 228.3556 0.6627348 185.6701 104.05373  0.2889544  90.97370
## 7      7 250.7114 0.6384285 201.1505  89.66404  0.2923203  73.83384
## 8      8 251.4190 0.6066894 205.7348 101.50416  0.3075454  85.07687
## 9      9 273.5836 0.5546636 224.1775  89.55636  0.3254680  76.54735
## 10    10 270.6959 0.5834046 223.7683 100.36276  0.3020473  91.07227
## 11    11 261.3546 0.5682360 214.4411  92.74517  0.3022907  79.37151
## 12    12 264.7770 0.5393702 221.0624  95.73593  0.3326214  87.99210
## 13    13 254.8483 0.5550906 210.4418  93.37880  0.3311902  81.49800
```

```
## 14    14 256.3582 0.5892507 210.8387  94.15603  0.3016167  84.53106
## 15    15 260.1745 0.5766254 213.6233  95.68297  0.3079019  84.88635
```

```
step.model$bestTune
```

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

```
# check the summary
summary(step.model$finalModel)
```

```
## 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 ) "*" " " "*" "*" " " " " " " " " " " " " "*" " "    "*"  "*"  " "
```

```
# check the coefficients
coef(step.model$finalModel, 6)
```

```
## (Intercept)           M          Ed         Po1          U2        Ineq        Prob
##   905.08511   131.98475   219.79230   341.84009    75.47364   269.90968   -86.44225
```

```
# loop through rows to check the accuracy
sst_sw <- sum((df$Crime - mean(df$Crime))^2)
sse_sw <- 0
for(i in 1:nrow(df_scaled)) {
  step_model = lm(Crime ~ M + Ed + Po1 + U2 + Ineq + Prob, data = df_scaled[-i,])
  pred_i <- predict(step_model, newdata = df_scaled[i,])
  sse_sw <- sse_sw + ((pred_i - df[i,16])^2)
}
r2_sw <- 1 - sse_sw/sst_sw
r2_sw
```

```
##         1
## 0.6661638
```

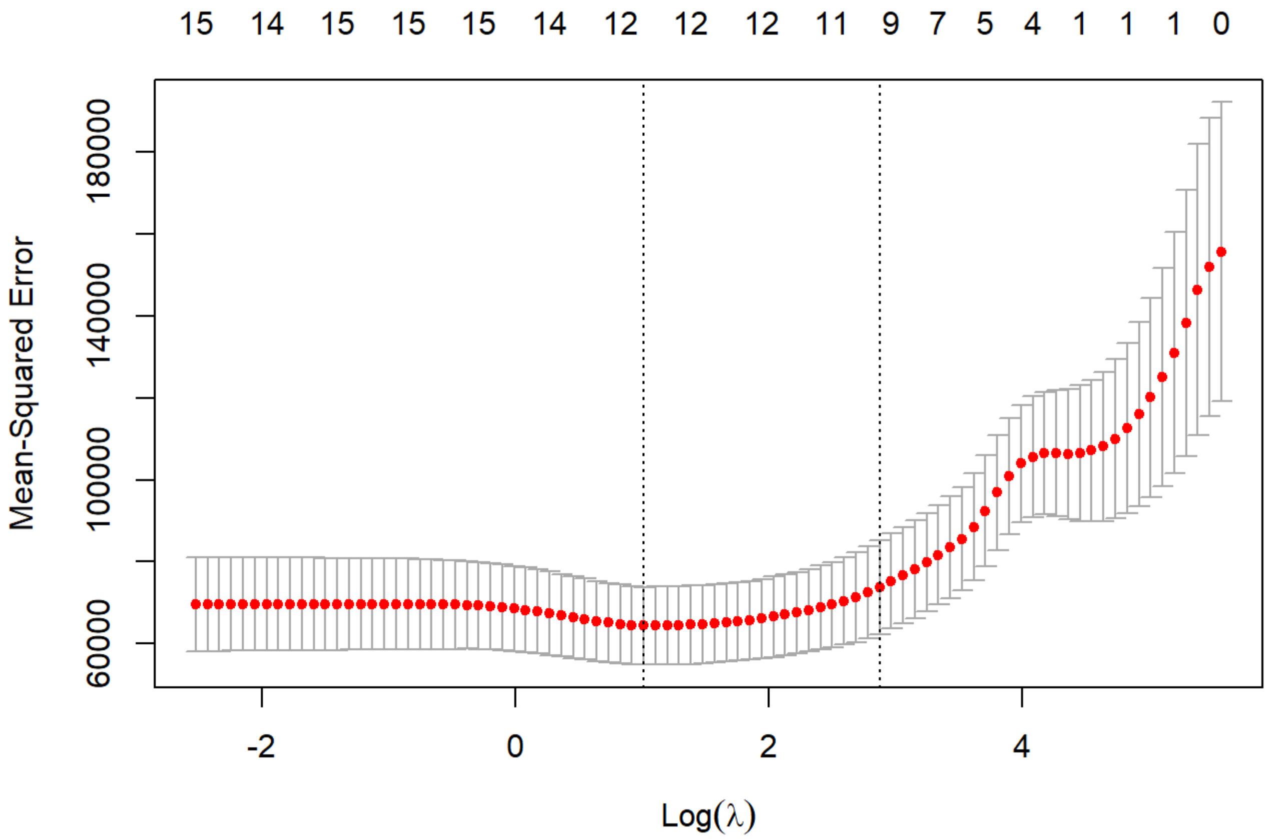
```
###########summary########

# we can see that the significant variables for stepwise regression is M,Ed,Po1,
# U2,Ineq and Prob.The adjusted r2 for the model is 0.7307, r2 for cross-validation
# is 0.6676.

###########
############Lasso regression###############
#
library(glmnet)
y_variable <- as.matrix(df_scaled$Crime)
x_variable <- as.matrix(df_scaled[, -16])
# set up cross validation lasso model with glmnet package, setting up folds of
#cross validation to 10
cv_lasso <- cv.glmnet(x_variable,y_variable, alpha = 1, nfolds = 10,
                      type.measure = "mse", family = "gaussian")
# filter out the minimum lambda
best_lambda <- cv_lasso$lambda.min
best_lambda
```

```
## [1] 2.756229
```

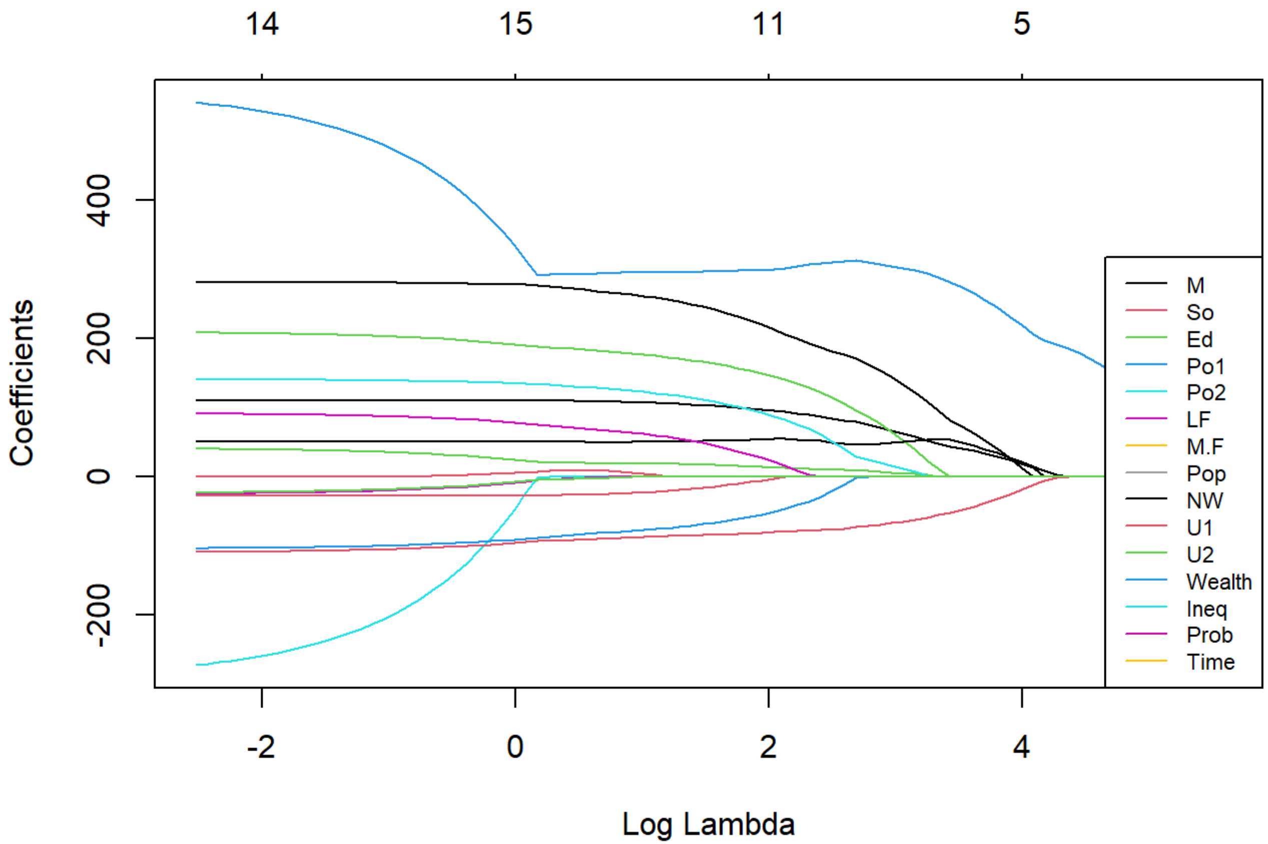
```
# find coefficients of lasso model
coef(cv_lasso, cv_lasso$lambda.min)
```

```
## 16 x 1 sparse Matrix of class "dgCMatrix"
##                      s1
## (Intercept) 894.76981
## M           105.82170
## So           30.30119
## Ed          178.57199
## Po1         295.87511
## Po2           .
## LF            .
## M.F          51.79120
## Pop         -21.38095
## NW           15.06974
## U1          -74.24490
## U2          120.38557
## Wealth       58.62421
## Ineq        256.18504
## Prob        -89.82648
## Time          .
```

```
# visualization & coefficients shrink
plot(cv_lasso)
```




```r
res <- glmnet(x_variable, y_variable, alpha = 1, lambda = cv_lasso$lambda, standardize = FALSE)
plot(res, xvar = "lambda")
legend("bottomright", lwd = 1, col = 1:15, legend = colnames(x_variable), cex = .7)
```




```
#
model_lasso <- lm(Crime~ M + So + Ed + Po1 + M.F + Pop + NW + U1 + U2 +
                    Wealth + Ineq + Prob, data = df_scaled)
summary(model_lasso)
```

```
##
## Call:
## lm(formula = Crime ~ M + So + Ed + Po1 + M.F + Pop + NW + U1 +
##     U2 + Wealth + Ineq + Prob, data = df_scaled)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -434.18 -107.01   18.55  115.88  470.32
##
## Coefficients:
```

```
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)   897.29      51.91  17.286  < 2e-16 ***
## M             112.71      49.35   2.284  0.02876 *
## So             22.89     125.35   0.183  0.85621
## Ed            195.70      62.94   3.109  0.00378 **
## Po1           293.18      64.99   4.511 7.32e-05 ***
## M.F            48.92      48.12   1.017  0.31656
## Pop           -33.25      45.63  -0.729  0.47113
## NW             19.16      57.71   0.332  0.74195
## U1            -89.76      65.68  -1.367  0.18069
## U2            140.78      66.77   2.108  0.04245 *
## Wealth         83.30      95.53   0.872  0.38932
## Ineq          285.77      85.19   3.355  0.00196 **
## Prob          -92.75      41.12  -2.255  0.03065 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 202.6 on 34 degrees of freedom
## Multiple R-squared:  0.7971, Adjusted R-squared:  0.7255
## F-statistic: 11.13 on 12 and 34 DF,  p-value: 1.52e-08
```

```r
# loop through rows to check the accuracy
sst_la <- sum((df$Crime - mean(df$Crime))^2)
sse_la <- 0
for(i in 1:nrow(df_scaled)) {
  lasso_model = lm(Crime ~ M + So + Ed + Po1 + M.F + NW + U2 + Ineq + Prob,
                   data = df_scaled[-i,])
  pred_i <- predict(lasso_model, newdata = df_scaled[i,])
  sse_la <- sse_la + ((pred_i - df[i,16])^2)
}
r2_lasso <- 1 - sse_la/sst_la
r2_lasso
```

```
##        1
## 0.620291
```

```r
#
model_rebuid = lm(Crime ~ M + Ed + Po1 + U2 + Ineq + Prob, data = df_scaled)
summary(model_rebuid)
```

```
##
## Call:
## lm(formula = Crime ~ M + Ed + Po1 + U2 + Ineq + Prob, data = df_scaled)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -470.68  -78.41  -19.68  133.12  556.23
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)   905.09      29.27  30.918  < 2e-16 ***
## M             131.98      41.85   3.154  0.00305 **
## Ed            219.79      50.07   4.390 8.07e-05 ***
## Po1           341.84      40.87   8.363 2.56e-10 ***
## U2             75.47      34.55   2.185  0.03483 *
## Ineq          269.91      55.60   4.855 1.88e-05 ***
## Prob          -86.44      34.74  -2.488  0.01711 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 200.7 on 40 degrees of freedom
## Multiple R-squared:  0.7659, Adjusted R-squared:  0.7307
## F-statistic: 21.81 on 6 and 40 DF,  p-value: 3.418e-11
```

```
######summary######

# we find the best lambda value, and using it to build up model. After we check the
# statistic significance, we using significant variables to build up model again.
# the significant variable for lasso is M, Ed, Po1, U2, Ineq and Prob. The Adjusted
# r-squared is 0.7255. accuracy for cross validation is 0.6203.
##############Elastic Net####################
# Elastic Net variable selection can be too dependent on data and thus unstable.
# The solution here is to combine the penalties of ridge regression and lasso to
# get the best of both.
# set up train control
train_control <- trainControl(method = "repeatedcv", number = 10, repeats = 5,
                              search = "random")
# train the model & print model
elastic_net_model <- train(Crime ~ .,data = df_scaled, method = "glmnet",
                           tuneLength = 15, trControl = train_control)
elastic_net_model
```

```
## glmnet
##
## 47 samples
## 15 predictors
##
## No pre-processing
## Resampling: Cross-Validated (10 fold, repeated 5 times)
## Summary of sample sizes: 44, 42, 42, 42, 42, 42, ...
## Resampling results across tuning parameters:
##
##   alpha       lambda       RMSE      Rsquared   MAE
##   0.04417930  0.066279634  264.3572  0.6005013  216.9645
##   0.04577691  1.095520159  264.0146  0.6018994  217.1242
##   0.07932199  0.002733406  264.8585  0.5994550  217.1952
##   0.09024189  1.108367119  263.9014  0.6021405  217.0619
##   0.25024358  0.650932027  264.2254  0.6010796  216.9603
##   0.31159158  0.001803115  264.8776  0.5990746  216.9046
##   0.61651379  0.100392753  264.9054  0.5993766  216.9615
##   0.70366666  0.858565096  263.3968  0.6034202  216.6188
##   0.71575780  0.005957316  264.8266  0.5992271  216.8352
##   0.72890090  0.509734599  264.2187  0.6012688  216.9023
##   0.76278761  0.014897101  264.9767  0.5993288  216.9484
##   0.82407300  0.003020130  264.7622  0.5993170  216.7902
##   0.88432834  0.141341155  264.9317  0.5993562  217.0562
##   0.98952189  0.291294504  264.5552  0.6004869  216.9696
##   0.99725219  0.008699334  264.7238  0.5994551  216.7361
##
## RMSE was used to select the optimal model using the smallest value.
## The final values used for the model were alpha = 0.7036667 and lambda = 0.8585651.
```

```
# filtered out the best alpha & lambda
elastic_net_model$bestTune
```

```
##       alpha    lambda
## 8 0.7036667 0.8585651
```

```
# print the full results
elastic_net_model$results
```

```
##         alpha      lambda     RMSE  Rsquared      MAE   RMSESD RsquaredSD    MAESD
## 1  0.04417930 0.066279634 264.3572 0.6005013 216.9645 104.2882  0.3143327 89.95938
## 2  0.04577691 1.095520159 264.0146 0.6018994 217.1242 102.2425  0.3110255 88.39143
## 3  0.07932199 0.002733406 264.8585 0.5994550 217.1952 105.8059  0.3168430 91.01478
## 4  0.09024189 1.108367119 263.9014 0.6021405 217.0619 102.1464  0.3108060 88.28210
## 5  0.25024358 0.650932027 264.2254 0.6010796 216.9603 103.9151  0.3137913 89.50878
## 6  0.31159158 0.001803115 264.8776 0.5990746 216.9046 107.5382  0.3183452 92.36235
## 7  0.61651379 0.100392753 264.9054 0.5993766 216.9615 107.3551  0.3180607 92.09460
## 8  0.70366666 0.858565096 263.3968 0.6034202 216.6188 102.4289  0.3109393 88.06947
## 9  0.71575780 0.005957316 264.8266 0.5992271 216.8352 107.5847  0.3183212 92.33185
## 10 0.72890090 0.509734599 264.2187 0.6012688 216.9023 104.6266  0.3144445 89.81462
## 11 0.76278761 0.014897101 264.9767 0.5993288 216.9484 107.5480  0.3182169 92.22603
## 12 0.82407300 0.003020130 264.7622 0.5993170 216.7902 107.5563  0.3182746 92.29525
## 13 0.88432834 0.141341155 264.9317 0.5993562 217.0562 107.1396  0.3182183 91.83823
## 14 0.98952189 0.291294504 264.5552 0.6004869 216.9696 106.2406  0.3164402 90.98895
## 15 0.99725219 0.008699334 264.7238 0.5994551 216.7361 107.5183  0.3182675 92.22483
```

```r
#
coef(elastic_net_model$finalModel, elastic_net_model$bestTune$lambda)
```

```
## 16 x 1 sparse Matrix of class "dgCMatrix"
##                      s1
## (Intercept)  901.08032
## M            110.22590
## So            11.76407
## Ed           196.71779
## Po1          388.72142
## Po2         -107.44398
## LF           -11.86208
## M.F           51.80040
## Pop          -26.97103
## NW            28.98884
## U1           -93.68767
## U2           137.02191
## Wealth        82.42043
## Ineq         276.96394
## Prob        -100.17899
## Time         -11.73740
```

```r
#### will shrink to zero
enm <- glmnet(x_variable, y_variable, alpha = 0.7036667, lambda = 0.8585651,
              family = 'gaussian')
enm
```

```
##
## Call:  glmnet(x = x_variable, y = y_variable, family = "gaussian", alpha = 0.7036667,      lambda = 0.8585651)
##
##   Df  %Dev Lambda
## 1 15 80.06 0.8586
```

```r
####set up linear regression
enm_lm = lm(Crime ~ M + So + Ed + Po1 + Po2 + LF + M.F + Pop + NW + U1 + U2 +
              Wealth + Ineq + Prob + Time, data = df_scaled)
summary(enm_lm)
```

```
##
## Call:
## lm(formula = Crime ~ M + So + Ed + Po1 + Po2 + LF + M.F + Pop +
##     NW + U1 + U2 + Wealth + Ineq + Prob + Time, data = df_scaled)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -395.74  -98.09   -6.69  112.99  512.67
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)  906.380     59.113  15.333 5.08e-16 ***
## M            110.382     52.424   2.106  0.04344 *
## So            -3.803    148.755  -0.026  0.97977
## Ed           210.678     69.458   3.033  0.00486 **
## Po1          572.995    315.347   1.817  0.07889 .
## Po2         -305.958    328.483  -0.931  0.35883
## LF           -26.826     59.394  -0.452  0.65465
## M.F           51.293     59.977   0.855  0.39900
## Pop          -27.906     49.095  -0.568  0.57385
## NW            43.234     66.642   0.649  0.52128
## U1          -105.056     75.906  -1.384  0.17624
## U2           141.714     69.536   2.038  0.05016 .
## Wealth        92.792    100.028   0.928  0.36075
```

```
## Ineq          281.954     90.630   3.111  0.00398 **
## Prob         -110.394     51.667  -2.137  0.04063 *
## Time          -24.655     50.780  -0.486  0.63071
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 209.1 on 31 degrees of freedom
## Multiple R-squared:  0.8031, Adjusted R-squared:  0.7078
## F-statistic: 8.429 on 15 and 31 DF,  p-value: 3.539e-07
```

```
# print three key indicator
predict_train <- enm %>% predict(x_variable)
data.frame( R2 = R2(predict_train, y_variable),
            RMSE = RMSE(predict_train, y_variable),
            MAE = MAE(predict_train, y_variable))
```

```
##          s0     RMSE      MAE
## 1 0.8006471 170.8779 133.6791
```

```
# write function to find the metrics of best tuning parameters
get_result = function(caret_fit) {
  best = which(rownames(caret_fit$results) == rownames(caret_fit$bestTune))
  best_result = caret_fit$results[best, ]
  rownames(best_result) = NULL
  best_result
}
# apply the function
get_result(elastic_net_model)
```

```
##       alpha    lambda     RMSE  Rsquared      MAE   RMSESD RsquaredSD    MAESD
## 1 0.7036667 0.8585651 263.3968 0.6034202 216.6188 102.4289  0.3109393 88.06947
```

```
#
enm_cv <- train(Crime ~ ., data = df_scaled, method="glmnet",
                trControl = train_control,
                tuneGrid = expand.grid(alpha = 0.7036667,
                                       lambda = 0.8585651))
print(enm_cv)
```

```
## glmnet
##
## 47 samples
## 15 predictors
##
## No pre-processing
## Resampling: Cross-Validated (10 fold, repeated 5 times)
## Summary of sample sizes: 43, 41, 43, 44, 42, 41, ...
## Resampling results:
##
##   RMSE      Rsquared   MAE
##   260.3687  0.6304321  215.8293
##
## Tuning parameter 'alpha' was held constant at a value of 0.7036667
## Tuning parameter 'lambda' was held constant at a value of 0.8585651
```

```
#
sst_enm <- sum((df$Crime - mean(df$Crime))^2)
sse_enm <- 0
for(i in 1:nrow(df_scaled)) {
  model_enm = lm(Crime ~ M + Ed + Po1 + U2 + Ineq + Prob, data = df_scaled[-i,])
  pred_i <- predict(model_enm, newdata = df_scaled[i,])
  sse_enm <- sse_enm + ((pred_i - df[i,16])^2)
}
R2_enm <- 1 - sse_enm/sst_enm
R2_enm
```

```
##         1
## 0.6661638
```

```
##### summary########
# the key takeaway here is i didn't trying to find the alpha in the first place.
# What i trying to do here is to tune both λ & α the same time. Briefly, Under
# elastic net regression, there are two parameters to tune: λ and α.
# The glmnet package allows to tune λ via cross-validation for a fixed α, but
# it does not support α-tuning, so i will turn to caret for this job.
# Referring to our model, we are trying to find the best alpha and lambda from 10
# fold. The metrics we are using is RMSE.The significant variable in elastic net
# regression is M, Ed, Po1, U2, Ineq, Prob, and adjusted r2 is 0.7078. The r2 for
# cross-validation is 0.6662.
```