Homework_8

Claire Kraft

2024-10-15

Question 11.1

Using the crime data set uscrime.txt from Questions 8.2, 9.1, and 10.1, build a regression model using:

1. **Stepwise regression**
2. **Lasso**
3. **Elastic net**

For Parts 2 and 3, remember to scale the data first – otherwise, the regression coefficients will be on different scales and the constraint won't have the desired effect.

For Parts 2 and 3, use the glmnet function in R.

In Stepwise regression I first created a regression that takes in all factors to find highly correlative factors. I think in a previous homework we did a similar exercise to extract the best factors. I don't recall what those factors were but based on the results from this code snippet, the best factors were *Ed* and *M.f* which have p-values of 0.1 or better. After running the R Squares on the forward and backward models, the results came out to be 0.1446551 meaning there is a 14.47% variance. I'm going to pull in a few more factors (Ed + M.F + Po1 + Po2 + Ineq) to see how the performance compares. The R Squared result did jump up to 0.7304136 or 73.04% variance. This lm_broad model seems more robust.

```
# Helper
#install.packages("glmnet")
library(ggplot2)
library(glmnet)
```

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

```
## Loaded glmnet 4.1-8
```

```
library(tree)
```

```
# Set up
rm(list = ls())

# Read in data
uscrime <- read.table("~/GitHub/omsa/ISYE 6501/Homework 08/uscrime.txt", stringsAsFactors = FALS
E, header = TRUE)
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
```

```
# Set seed for reproducibility
set.seed(123)
# 70% of the data
train_size <- floor(0.70 * nrow(uscrime))

# Randomly sample indices for the training set
train_indices <- sample(seq_len(nrow(uscrime)), size = train_size)

# Split the data into training and test sets - 70/30
train_data <- uscrime[train_indices, ]
test_data <- uscrime[-train_indices, ]

# Display the dimensions of the training and test sets
dim(train_data) # Should be approximately 33 x 16
```

```
## [1] 32 16
```

```
dim(test_data)  # Should be approximately 14 x 16
```

```
## [1] 15 16
```

```
#-------------------------- Stepwise reg - lm_slim
# Fit model
lm_everything <- lm(train_data$Crime ~., train_data[1:15])
# Find high correlations by taking in all factors
lm_everything_summary <- summary(lm_everything)
# Look for twinkle twinkle stars
print(lm_everything_summary)
```

```
##
## Call:
## lm(formula = train_data$Crime ~ ., data = train_data[1:15])
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -303.31  -81.48  -17.43   95.90  436.22
##
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept) -1.046e+04  2.265e+03  -4.616 0.000286 ***
## M            8.164e+01  4.929e+01   1.656 0.117145
## So           3.870e+01  2.295e+02   0.169 0.868161
## Ed           2.636e+02  7.791e+01   3.383 0.003791 **
## Po1          1.654e+02  1.285e+02   1.287 0.216427
## Po2         -1.171e+02  1.429e+02  -0.819 0.424632
## LF          -4.137e+03  2.300e+03  -1.799 0.090949 .
## M.F          7.947e+01  2.924e+01   2.718 0.015214 *
## Pop          1.861e+00  1.922e+00   0.968 0.347469
## NW           1.474e+01  1.027e+01   1.434 0.170715
## U1          -1.032e+04  5.619e+03  -1.838 0.084775 .
## U2           1.221e+02  1.033e+02   1.182 0.254357
## Wealth       1.026e-01  1.370e-01   0.749 0.464925
## Ineq         4.807e+01  2.979e+01   1.614 0.126162
## Prob         1.456e+03  3.926e+03   0.371 0.715603
## Time         1.209e+01  1.016e+01   1.190 0.251322
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 202.3 on 16 degrees of freedom
## Multiple R-squared:  0.8825, Adjusted R-squared:  0.7724
## F-statistic: 8.012 on 15 and 16 DF,  p-value: 7.895e-05
```

```
# Forward and backward
# Look at reference 1
lm_slim <- lm(Crime ~ Ed + M.F, data = train_data)
# Should be highly correlative
lm_slim_summary <- summary(lm_slim)
# Look for twinkle twinkle stars
print(lm_slim_summary)
```

```
##
## Call:
## lm(formula = Crime ~ Ed + M.F, data = train_data)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -526.48 -340.32  -83.95  218.82  903.34
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept) -2206.04    2420.87  -0.911   0.3697
## Ed            120.10      67.23   1.787   0.0845 .
## M.F            18.29      25.80   0.709   0.4839
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 405.4 on 29 degrees of freedom
## Multiple R-squared:  0.1447, Adjusted R-squared:  0.08567
## F-statistic: 2.452 on 2 and 29 DF,  p-value: 0.1038
```

```
forward <- step(lm_slim, direction = 'forward', scope = formula(~ .))
```

```
## Start:  AIC=387.16
## Crime ~ Ed + M.F
```

```
forward
```

```
##
## Call:
## lm(formula = Crime ~ Ed + M.F, data = train_data)
##
## Coefficients:
## (Intercept)           Ed          M.F
##    -2206.04       120.10        18.29
```

```
backward <- step(lm_slim, direction = 'backward', trace = 0)
backward
```

```
##
## Call:
## lm(formula = Crime ~ Ed, data = train_data)
##
## Coefficients:
## (Intercept)           Ed
##      -560.5        134.5
```

```
# Predict on the training data
predictions_slim <- predict(lm_slim, newdata = train_data)

# Calculate the total sum of squares (SST)
# Look at reference 2
sst_slim <- sum((train_data$Crime - mean(train_data$Crime))^2)

# Calculate the sum of squared errors (SSE)
# Look at reference 2
sse_slim <- sum((train_data$Crime - predictions_slim)^2)

# Calculate R-squared
# Look at reference 2
r2_slim <- 1 - (sse_slim / sst_slim)
r2_slim
```

```
## [1] 0.1446551
```

```
#-------------------------- Stepwise reg - lm_broad
# Forward and backward
# Look at reference 1
lm_broad <- lm(Crime ~ Ed + M.F + Po1 + Po2 + Ineq, data = train_data)
# Should be highly correlative
lm_broad_summary <- summary(lm_broad)
# Look for twinkle twinkle stars
print(lm_broad_summary)
```

```
##
## Call:
## lm(formula = Crime ~ Ed + M.F + Po1 + Po2 + Ineq, data = train_data)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -461.19 -126.61   12.86  148.78  557.18
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept) -5317.08    1623.77  -3.275 0.002993 **
## Ed            137.84      65.63   2.100 0.045551 *
## M.F            22.30      16.65   1.339 0.192106
## Po1            61.85     129.20   0.479 0.636142
## Po2            65.50     140.28   0.467 0.644435
## Ineq           77.90      20.51   3.799 0.000788 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 240.4 on 26 degrees of freedom
## Multiple R-squared:  0.7304, Adjusted R-squared:  0.6786
## F-statistic: 14.09 on 5 and 26 DF,  p-value: 1.043e-06
```

```
forward <- step(lm_broad, direction = 'forward', scope = formula(~ .))
```

```
## Start:  AIC=356.21
## Crime ~ Ed + M.F + Po1 + Po2 + Ineq
```

```
forward
```

```
##
## Call:
## lm(formula = Crime ~ Ed + M.F + Po1 + Po2 + Ineq, data = train_data)
##
## Coefficients:
## (Intercept)           Ed          M.F          Po1          Po2         Ineq
##    -5317.08       137.84        22.30        61.85        65.50        77.90
```

```
backward <- step(lm_broad, direction = 'backward', trace = 0)
backward
```

```
##
## Call:
## lm(formula = Crime ~ Ed + Po1 + Ineq, data = train_data)
##
## Coefficients:
## (Intercept)           Ed          Po1         Ineq
##    -3563.50       173.47       120.40        82.46
```

```
# Predict on the training data
predictions_broad <- predict(lm_broad, newdata = train_data)

# Calculate the total sum of squares (SST)
# Look at reference 2
sst_broad <- sum((train_data$Crime - mean(train_data$Crime))^2)

# Calculate the sum of squared errors (SSE)
# Look at reference 2
sse_broad <- sum((train_data$Crime - predictions_broad)^2)

# Calculate R-squared
# Look at reference 2
r2_broad <- 1 - (sse_broad / sst_broad)
r2_broad
```

```
## [1] 0.7304136
```

```r
# Set up
rm(list = ls())
# Clear the workspace by removing all objects from the environment.

# Read in data
uscrime <- read.table("~/GitHub/omsa/ISYE 6501/Homework 08/uscrime.txt", stringsAsFactors = FALS
E, header = TRUE)
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
```

```r
# Scale data
uscrime_scaled <- data.frame(scale(uscrime[1:15]), Crime = uscrime$Crime)
head(uscrime_scaled)
```

```
##            M         So         Ed        Po1        Po2         LF         M.F
## 1  0.9886930  1.3770536 -1.3085099 -0.9085105 -0.8666988 -1.2667456 -1.12060499
## 2  0.3521372 -0.7107373  0.6580587  0.6056737  0.5280852  0.5396568  0.98341752
## 3  0.2725678  1.3770536 -1.4872888 -1.3459415 -1.2958632 -0.6976051 -0.47582390
## 4 -0.2048491 -0.7107373  1.3731746  2.1535064  2.1732150  0.3911854  0.37257228
## 5  0.1929983 -0.7107373  1.3731746  0.8075649  0.7426673  0.7376187  0.06714965
## 6 -1.3983912 -0.7107373  0.3898903  1.1104017  1.2433590 -0.3511718 -0.64550313
##           Pop           NW          U1         U2     Wealth       Ineq
## 1 -0.09500679  1.943738564  0.69510600  0.8313680 -1.3616094  1.6793638
## 2 -0.62033844  0.008483424  0.02950365  0.2393332  0.3276683  0.0000000
## 3 -0.48900552  1.146296747 -0.08143007 -0.1158877 -2.1492481  1.4036474
## 4  3.16204944 -0.205464381  0.36230482  0.5945541  1.5298536 -0.6767585
## 5 -0.48900552 -0.691709391 -0.24783066 -1.6551781  0.5453053 -0.5013026
## 6 -0.30513945 -0.555560788 -0.63609870 -0.5895155  1.6956723 -1.7044289
##         Prob        Time Crime
## 1  1.6497631 -0.05599367   791
## 2 -0.7693365 -0.18315796  1635
## 3  1.5969416 -0.32416470   578
## 4 -1.3761895  0.46611085  1969
## 5 -0.2503580 -0.74759413  1234
## 6 -0.5669349 -0.78996812   682
```

```
# Set seed for reproducibility
set.seed(123)

# 70% of the data
train_size <- floor(0.70 * nrow(uscrime_scaled))

# Randomly sample indices for the training set
train_indices <- sample(seq_len(nrow(uscrime_scaled)), size = train_size)

# Split the data into training and test sets - 70/30
train_data <- uscrime_scaled[train_indices, ]
test_data <- uscrime_scaled[-train_indices, ]


# Display the dimensions of the training and test sets
dim(train_data) # Should be approximately 33 x 16
```

```
## [1] 32 16
```

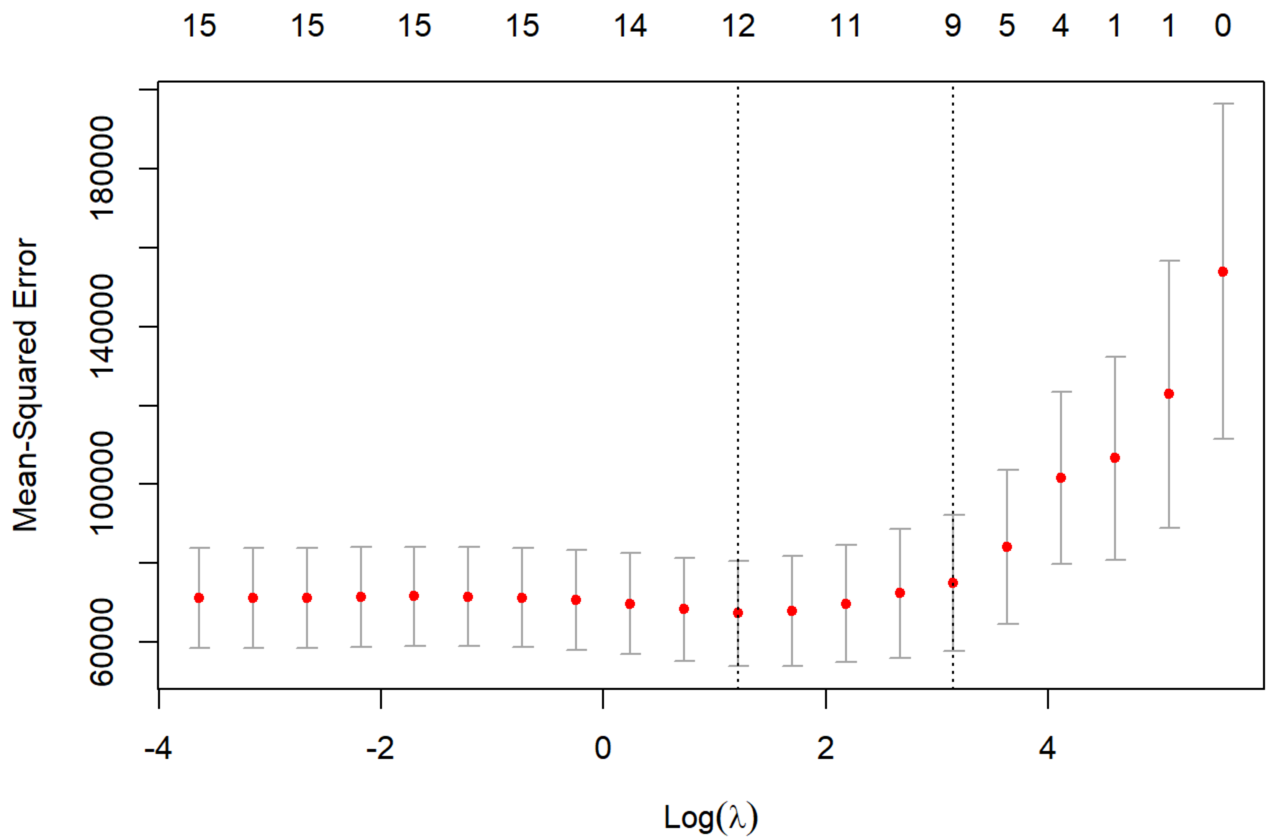
```
dim(test_data)  # Should be approximately 14 x 16
```

```
## [1] 15 16
```

```
# Load necessary library
library(glmnet)

# Set seed for reproducibility
set.seed(123)

# Prepare the data
x <- as.matrix(uscrime_scaled[,-16]) # Features
y <- as.matrix(uscrime_scaled[,16])  # Response (Crime)

# Fit the LASSO model with cross-validation
lasso_model <- cv.glmnet(x = x,
                         y = y,
                         alpha = 1,             # Alpha = 1 for LASSO
                         nfolds = 10,           # 10-fold cross-validation
                         nlambda = 20,          # Number of lambda values
                         type.measure = 'mse',  # Mean Squared Error
                         family = 'gaussian',   # Regression (as opposed to classification)
                         standardize = TRUE)    # Standardize features

# Display the best lambda value
best_lambda <- lasso_model$lambda.min
best_lambda
```

```
## [1] 3.352559
```

```
# Produce a plot of test MSE by lambda value
plot(lasso_model)
```




```
# Display the best lambda value
lasso_model$lambda.min
```

```
## [1] 3.352559
```

```
# Display lambda, cross-validated MSE, and number of non-zero coefficients
cbind(lasso_model$lambda, lasso_model$cvm, lasso_model$nzero)
```

```
##             [,1]      [,2] [,3]
## s0  263.09539664 154014.48    0
## s1  162.02682936 122823.25    1
## s2   99.78393301 106600.67    1
## s3   61.45175663 101680.03    4
## s4   37.84495439  84119.78    5
## s5   23.30674746  74897.75    9
## s6   14.35341873  72323.40   10
## s7    8.83952725  69680.14   11
## s8    5.44380704  67782.43   12
## s9    3.35255883  67287.64   12
## s10   2.06466736  68282.92   13
## s11   1.27152170  69677.77   14
## s12   0.78306436  70721.44   15
## s13   0.48224879  71242.19   15
## s14   0.29699205  71492.98   15
## s15   0.18290202  71556.62   15
## s16   0.11263988  71398.22   14
## s17   0.06936907  71258.69   15
## s18   0.04272082  71174.59   15
## s19   0.02630954  71132.39   15
```

```
# Display coefficients at the best lambda value
coef(lasso_model, s = lasso_model$lambda.min)
```

```
## 16 x 1 sparse Matrix of class "dgCMatrix"
##                    s1
## (Intercept) 905.08511
## M           104.33521
## So           15.23279
## Ed          174.95444
## Po1         296.14026
## Po2           .
## LF            .
## M.F          52.35011
## Pop         -18.84810
## NW           14.34342
## U1          -70.88764
## U2          115.98159
## Wealth       53.91208
## Ineq        250.09818
## Prob        -89.16550
## Time          .
```

```
# Set up
rm(list = ls())
# Clear the workspace by removing all objects from the environment.

# Read in data
uscrime <- read.table("~/GitHub/omsa/ISYE 6501/Homework 08/uscrime.txt", stringsAsFactors = FALS
E, header = TRUE)
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
```

```
# Scale data
uscrime_scaled <- data.frame(scale(uscrime[1:15]), Crime = uscrime$Crime)
head(uscrime_scaled)
```

```
##            M         So         Ed        Po1        Po2         LF         M.F
## 1  0.9886930  1.3770536 -1.3085099 -0.9085105 -0.8666988 -1.2667456 -1.12060499
## 2  0.3521372 -0.7107373  0.6580587  0.6056737  0.5280852  0.5396568  0.98341752
## 3  0.2725678  1.3770536 -1.4872888 -1.3459415 -1.2958632 -0.6976051 -0.47582390
## 4 -0.2048491 -0.7107373  1.3731746  2.1535064  2.1732150  0.3911854  0.37257228
## 5  0.1929983 -0.7107373  1.3731746  0.8075649  0.7426673  0.7376187  0.06714965
## 6 -1.3983912 -0.7107373  0.3898903  1.1104017  1.2433590 -0.3511718 -0.64550313
##           Pop           NW          U1         U2     Wealth       Ineq
## 1 -0.09500679  1.943738564  0.69510600  0.8313680 -1.3616094  1.6793638
## 2 -0.62033844  0.008483424  0.02950365  0.2393332  0.3276683  0.0000000
## 3 -0.48900552  1.146296747 -0.08143007 -0.1158877 -2.1492481  1.4036474
## 4  3.16204944 -0.205464381  0.36230482  0.5945541  1.5298536 -0.6767585
## 5 -0.48900552 -0.691709391 -0.24783066 -1.6551781  0.5453053 -0.5013026
## 6 -0.30513945 -0.555560788 -0.63609870 -0.5895155  1.6956723 -1.7044289
##         Prob        Time Crime
## 1  1.6497631 -0.05599367   791
## 2 -0.7693365 -0.18315796  1635
## 3  1.5969416 -0.32416470   578
## 4 -1.3761895  0.46611085  1969
## 5 -0.2503580 -0.74759413  1234
## 6 -0.5669349 -0.78996812   682
```

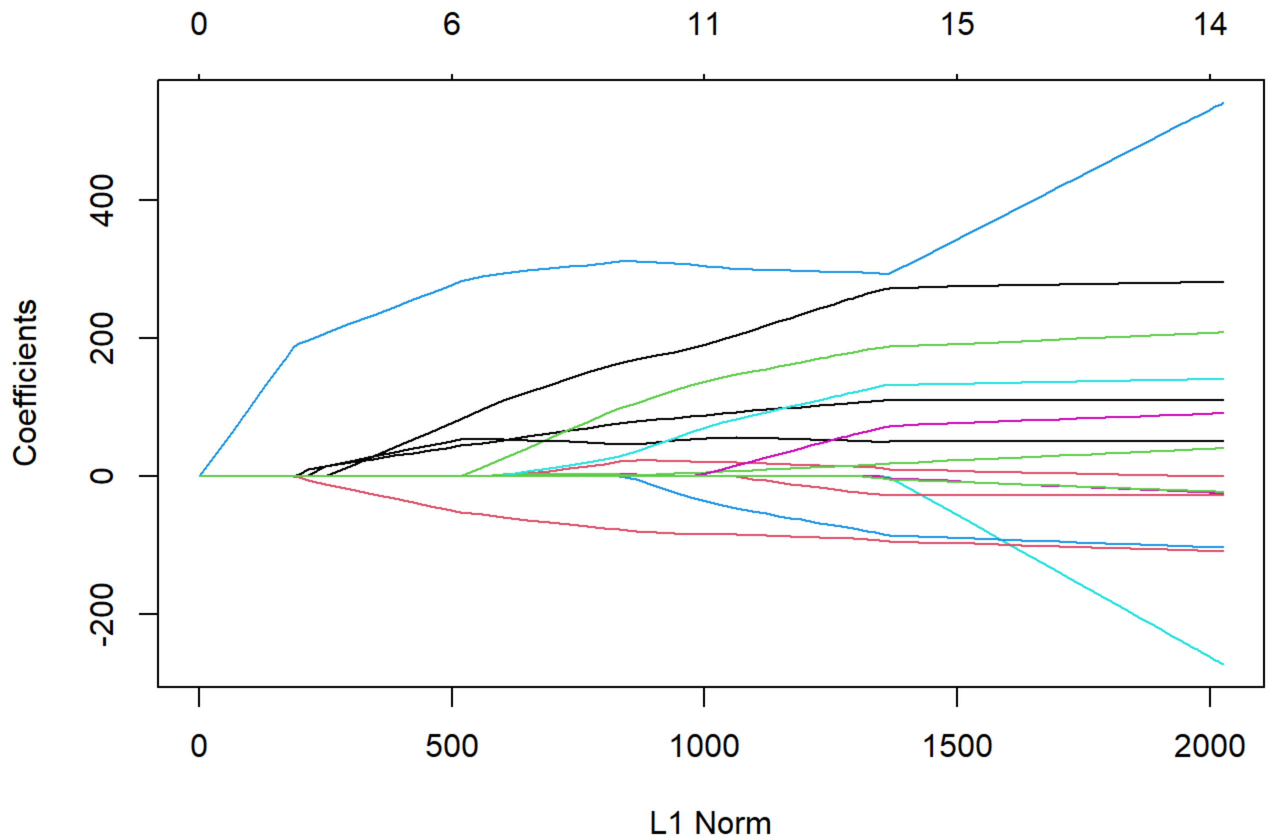
```
# Set seed for reproducibility
set.seed(123)

# 70% of the data
train_size <- floor(0.70 * nrow(uscrime_scaled))

# Randomly sample indices for the training set
train_indices <- sample(seq_len(nrow(uscrime_scaled)), size = train_size)

# Split the data into training and test sets - 70/30
train_data <- uscrime_scaled[train_indices, ]
test_data <- uscrime_scaled[-train_indices, ]


# Display the dimensions of the training and test sets
dim(train_data) # Should be approximately 33 x 16
```

```
## [1] 32 16
```

```
dim(test_data)  # Should be approximately 14 x 16
```

```
## [1] 15 16
```

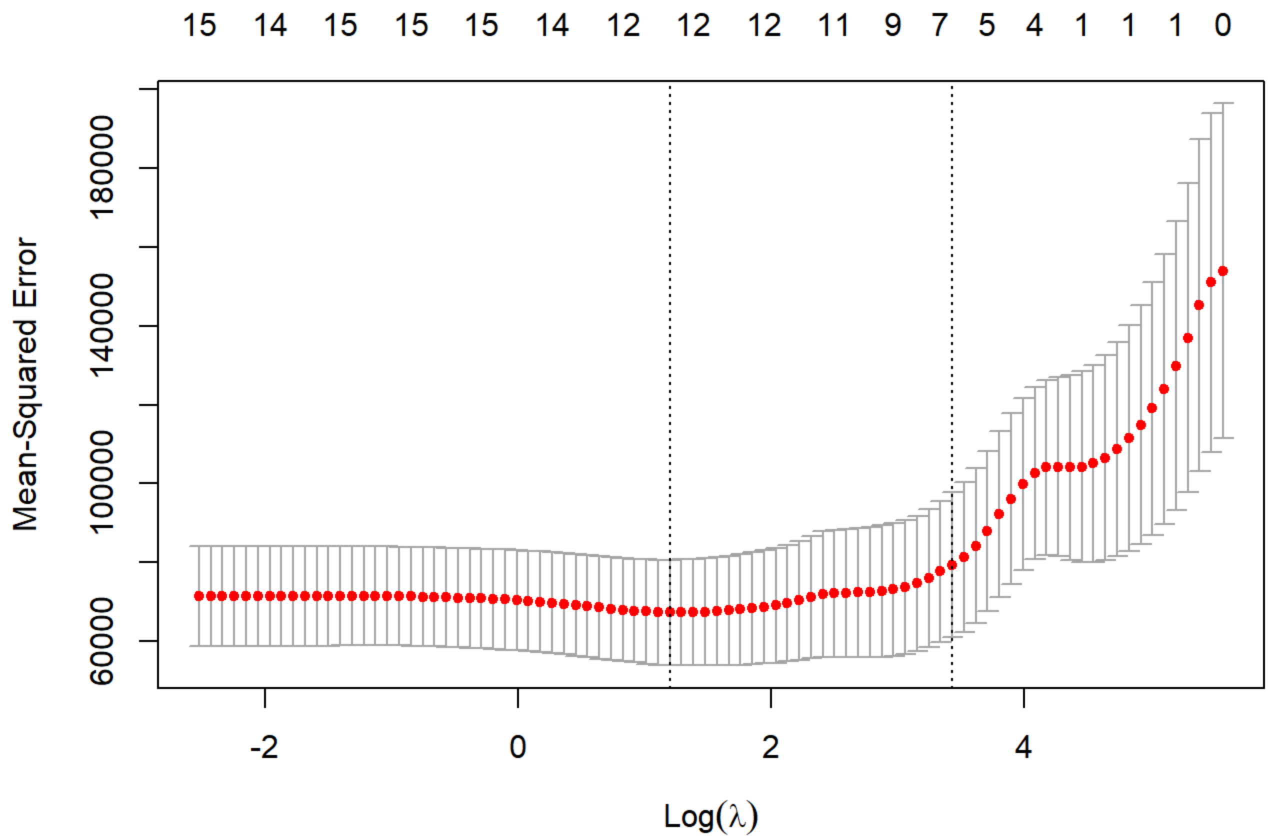
```
# Load necessary library
library(glmnet)

# Set seed for reproducibility
set.seed(123)

# Prepare the data
x <- as.matrix(uscrime_scaled[,-16]) # Features
y <- as.matrix(uscrime_scaled[,16])  # Response (Crime)

# Look at reference 4
elastic_net_model <- glmnet(x, y)
plot(elastic_net_model)
```




```
coef(elastic_net_model, s = 0.1)
```

```
## 16 x 1 sparse Matrix of class "dgCMatrix"
##                       s1
## (Intercept)  905.08511
## M            110.41769
## So             .
## Ed           208.30437
## Po1          537.28311
## Po2         -267.60142
## LF           -23.90589
## M.F           51.30889
## Pop          -27.77932
## NW            40.30472
## U1          -102.92284
## U2           140.96599
## Wealth        91.17590
## Ineq         281.36290
## Prob        -108.49278
## Time         -22.18052
```

```
#set.seed(123)
#nx <- matrix(rnorm(5 * 15), 5, 15)
#predict(elastic_net_model, newx = nx, s = c(0.1, 0.05))

elastic_net_model_cv <- cv.glmnet(x, y)
plot(elastic_net_model_cv)
```




```
# Display the best lambda value
best_lambda <- elastic_net_model_cv$lambda.min
best_lambda
```

```
## [1] 3.319887
```

```
# Display the best lambda value
elastic_net_model_cv$lambda.min
```

```
## [1] 3.319887
```

```
# Display lambda, cross-validated MSE, and number of non-zero coefficients
cbind(elastic_net_model_cv$lambda, elastic_net_model_cv$cvm, elastic_net_model_cv$nzero)
```

```
##             [,1]      [,2] [,3]
## s0  263.09539664 154014.48    0
## s1  239.72272672 151019.28    1
## s2  218.42642038 145332.71    1
## s3  199.02201920 137010.01    1
## s4  181.34145155 129868.15    1
## s5  165.23157679 123937.47    1
## s6  150.55285890 119012.46    1
## s7  137.17815786 114922.49    1
## s8  124.99162839 111525.88    1
## s9  113.88771662 108705.01    1
## s10 103.77024576 106524.62    1
## s11  94.55158313 105089.28    1
## s12  86.15188108 104243.33    1
## s13  78.49838541 104055.54    1
## s14  71.52480520 104261.50    3
## s15  65.17073864 104039.91    4
## s16  59.38114984 102662.68    4
## s17  54.10589215  99818.68    5
## s18  49.29927381  96100.68    5
## s19  44.91966220  92232.33    5
## s20  40.92912321  87955.97    5
## s21  37.29309271  84229.96    5
## s22  33.98007714  81427.10    5
## s23  30.96138074  79404.18    6
## s24  28.21085701  77703.93    7
## s25  25.70468222  76026.40    9
## s26  23.42114910  74610.75    9
## s27  21.34047877  73769.37    9
## s28  19.44464945  73095.61    9
## s29  17.71724038  72694.47    9
## s30  16.14328958  72419.01   10
## s31  14.70916422  72342.72   10
## s32  13.40244262  72278.29   11
## s33  12.21180655  72128.83   11
## s34  11.12694332  71943.44   11
## s35  10.13845634  71056.98   11
## s36   9.23778382  70287.93   11
## s37   8.41712457  69649.50   11
## s38   7.66937042  69128.46   12
## s39   6.98804469  68716.57   12
## s40   6.36724606  68396.18   12
## s41   5.80159747  68041.99   12
## s42   5.28619954  67789.02   12
## s43   4.81658814  67618.00   12
## s44   4.38869572  67470.43   12
## s45   3.99881609  67369.69   12
## s46   3.64357229  67310.55   12
## s47   3.31988737  67282.09   12
## s48   3.02495773  67312.32   12
## s49   2.75622882  67493.21   12
## s50   2.51137305  67721.29   12
```

```
## s51   2.28826959  67971.54   12
## s52   2.08498602  68235.52   13
## s53   1.89976161  68519.12   13
## s54   1.73099203  68812.44   13
## s55   1.57721547  69096.77   13
## s56   1.43709999  69396.45   14
## s57   1.30943199  69709.80   14
## s58   1.19310566  69989.94   15
## s59   1.08711344  70159.89   15
## s60   0.99053728  70370.93   15
## s61   0.90254068  70564.09   15
## s62   0.82236145  70694.14   15
## s63   0.74930513  70798.09   15
## s64   0.68273892  70894.94   15
## s65   0.62208628  70995.05   15
## s66   0.56682185  71100.86   15
## s67   0.51646696  71201.76   15
## s68   0.47058546  71289.98   15
## s69   0.42877994  71367.76   15
## s70   0.39068831  71421.39   15
## s71   0.35598064  71453.05   15
## s72   0.32435630  71470.53   15
## s73   0.29554138  71482.28   15
## s74   0.26928630  71482.34   15
## s75   0.24536365  71482.04   15
## s76   0.22356622  71483.27   15
## s77   0.20370521  71488.25   15
## s78   0.18560860  71491.74   15
## s79   0.16911964  71493.78   15
## s80   0.15409552  71497.49   14
## s81   0.14040610  71501.90   14
## s82   0.12793281  71501.23   14
## s83   0.11656761  71497.21   14
## s84   0.10621207  71489.96   14
## s85   0.09677648  71485.84   14
## s86   0.08817912  71492.84   14
## s87   0.08034553  71499.45   15
```

```
# Display coefficients at the best lambda value
coef(elastic_net_model_cv, s = elastic_net_model_cv$lambda.min)
```

```
## 16 x 1 sparse Matrix of class "dgCMatrix"
##                    s1
## (Intercept) 905.08511
## M           104.46095
## So           15.15298
## Ed          175.11702
## Po1         296.32198
## Po2           .
## LF            .
## M.F          52.34557
## Pop         -18.94369
## NW           14.32038
## U1          -71.07987
## U2          116.24494
## Wealth       53.72200
## Ineq        250.17012
## Prob        -89.19999
## Time          .
```

References:

[1] Sanderson, S. P. (2023, December 6). Steve’s Data Tips and Tricks - A Complete Guide to Stepwise Regression in R. Steve’s Data Tips and Tricks. https://www.spsanderson.com/steveondata/posts/2023-12-06/index.html (https://www.spsanderson.com/steveondata/posts/2023-12-06/index.html)

[2] Function to calculate R2 (R-squared) in R. (n.d.). Stack Overflow. https://stackoverflow.com/questions/40901445/function-to-calculate-r2-r-squared-in-r (https://stackoverflow.com/questions/40901445/function-to-calculate-r2-r-squared-in-r)

[3] Microsoft. (2024). Copilot: AI companion. Accessed 2024-10-15. Prompt: ‘Add comments to my lines of code.’ Generated using https://www.microsoft.com (https://www.microsoft.com).”

[4] Hastie , T., Qian, J., & Tay, K. (2023, March 27). An Introduction to `glmnet` . Glmnet.stanford.edu. https://glmnet.stanford.edu/articles/glmnet.html (https://glmnet.stanford.edu/articles/glmnet.html)

[5] APA citation