Question 11.1

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.

Notes on R:

- For the elastic net model, what we called λ in the videos, glmnet calls "alpha"; you can get a range of results by varying alpha from 1 (lasso) to 0 (ridge regression) [and, of course, other values of alpha in between].
- In a function call like glmnet(x,y,family="mgaussian",alpha=1) the predictors x need to be in R's matrix format, rather than data frame format. You can convert a data frame to a matrix using as.matrix – for example, x <- as.matrix(data[,1:n-1])
- Rather than specifying a value of T, glmnet returns models for a variety of values of T.

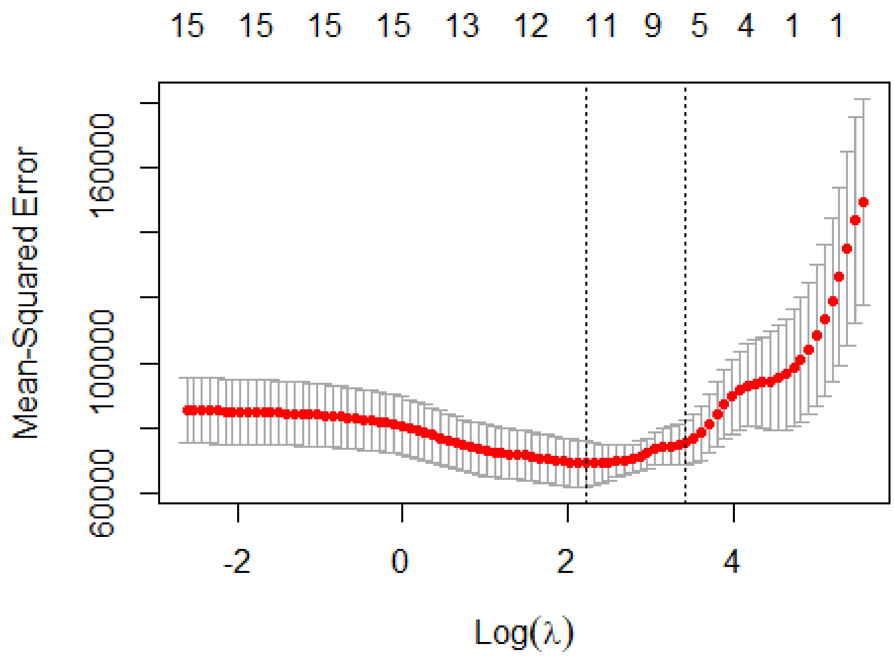
Using the crime data set uscrime.txt from Questions 8.2, 9.1, and 10.1, build a regression model using:

1. Stepwise regression

The code for this approach can be found in appendix 11.1.1. In order to approach the stepwise regression, I first began by scaling the data. Once that was done, I wanted to create the stepwise regression model, so I used the stepAIC function with direction set to both, so it would do a backward and forward search. In doing so, the predictors that the model identified were: M, Ed, Po1, M.F, U1, U2, Ineq, Prob. Then, I wanted to identify the best number of predictors. For this, I set up a 10-fold cross-validation approach, trained the model on the scaled data, and used the "leapSeq" method, which would fit the linear regression with stepwise selection. I also set nvmax values from 1:15 so that it would identify the best models with 1 predictor, 2 predictors, …, 15 predictors. Upon finishing this process, I was able to identify that the best model was one with six predictors as it had the lowest RMSE (232.8635). The predictors present in this model were M, Ed, Po1, U2, Ineq, and Prob. I then wanted to cross-validate and calculate the R-squared values for the model with the originally identified predictors and the model with only six predictors. In the end, the R-squared value for the model with predictors: M, Ed, Po1, M.F, U1, U2, Ineq, and Prob was 0.667621. The R-squared value for the model with predictors: M, Ed, Po1, U2, Ineq, and Prob was 0.6661638. Ultimately, both values were pretty similar here, and it seems that M.F and U1 were not too significant overall.

2. Lasso

The code for this approach can be found in appendix 11.1.2. In order to approach lasso regression, I also began by first scaling the data. Once that was done, I used the cv.glmnet function to perform the actual lasso regression and plotted the output:



The above graph displays the MSE against the log of lambda. The leftmost vertical line shows that the log of the optimal value of lambda is approximately two. In particular, the value of lambda is 9.237784. Following this, I wanted to find the non-zero coefficient variables of the model associated with the minimum (optimal) value of lambda. They are: So, M, Ed, Po1, M.F, NW, U1, U2, Wealth, Ineq, and Prob. I then fitted a new model with the aforementioned factors. Upon doing this, it showed me there were only six significant variables: M, Ed, Po1, U2, Ineq, and Prob. I then wanted to cross-validate and calculate the R-squared value of the original model and the model with only the six significant variables. The R-squared value for the original model, with predictors: So, M, Ed, Po1, M.F, NW, U1, U2, Wealth, Ineq, and Prob, was 0.6051577. The R-squared value for the model with only the significant predictors: M, Ed, Po1, U2, Ineq, and Prob was 0.6661638. Here, the R-squared value increased as we removed the predictors So, M.F, NW, U1, and Wealth, indicating that they are likely not essential to the model.

3. Elastic net

The code for this approach can be found in appendix 11.1.3. In order to approach the elastic net regression, I first began by scaling the data. Then, I wanted to vary the alpha values from 0 to 1, incrementing by 0.1 each time, to find the best value. In order to identify which was best, I looked at which alpha value resulted in the highest R-squared value among those iterated through. As a result of this, I found the best alpha value was 0.9. Once this was done, I built an elastic net model using the best value of alpha. The important variables identified were: So, M, Ed, Po1, M.F, Pop, NW, U1, U2, Wealth, Ineq, and Prob. I then used these variables to fit a new model. From this, there were six significant variables identified: M, Ed, Po1, U2, Ineq, and Prob.

Then, I used cross-validation to calculate the R-squared value of the original model and the model with only six significant variables. The R-squared value for the original model, with predictors: So, M, Ed, Po1, M.F, Pop, NW, U1, U2, Wealth, Ineq, and Prob, was 0.5903894. The R-squared value for the model with only the six significant predictors: M, Ed, Po1, U2, Ineq, and Prob was 0.6661638. Here, the R-squared values once again increased as we removed inessential variables such as So, M.F, Pop, NW, U1, and Wealth.

Summary

Model	**Relevant Variables**	**CV R-Squared**
Stepwise Regression, All Variables	M, Ed, Po1, M.F, U1, U2, Ineq, and Prob	0.667621
Stepwise Regression, Significant Variables	M, Ed, Po1, U2, Ineq, and Prob	0.6661638
LASSO Regression, All Variables	So, M, Ed, Po1, M.F, NW, U1, U2, Wealth, Ineq, and Prob	0.6051577
LASSO Regression, Significant Variables	M, Ed, Po1, U2, Ineq, and Prob	0.6661638
Elastic Net, All Variables	So, M, Ed, Po1, M.F, Pop, NW, U1, U2, Wealth, Ineq, and Prob	0.5903894
Elastic Net, Significant Variables	M, Ed, Po1, U2, Ineq, and Prob	0.6661638

There are a few interesting observations here. First of all, it appears that, when using all variables, elastic net regression performed the worst out of all three while stepwise regression performed the best. It is also interesting to note that elastic net, when using all variables, had the most at 12 while LASSO had 11, and stepwise had 8. Therefore, decreasing the number of variables selected does seem to improve the overall quality of the model. Furthermore, for each model, when I looked for the significant variables identified, the same six were identified each time: M, Ed, Po1, U2, Ineq, and Prob. In addition to this, more often than not, the six variables also produced the highest cross-validation R-squared value when fitted to a model (0.6661638) when compared to the R-squared value produced by fitting all identified variables. It is also likely that these variable-selection models were particularly helpful in this situation because there was a small number of data points. As per lecture, in these situations, we do not want to have too many factors as it can cause overfitting. Overall, it seems that variable-selection models are extremely helpful and can produce really good results, especially compared to some of the other models we have used in class.

Appendix

Question 11.1.1

```
#This is the stepwise regression approach

#Load Packages
library(tidyverse); library(caret); library(leaps); library(MASS)

## ── Attaching packages ─────────────────────────────────────── tidyverse 1.
3.2 ──
## ✓ ggplot2 3.4.0      ✓ purrr   1.0.1
## ✓ tibble  3.1.8      ✓ dplyr   1.0.10
## ✓ tidyr   1.2.1      ✓ stringr 1.5.0
## ✓ readr   2.1.3      ✓ forcats 0.5.2
## ── Conflicts ────────────────────────────────────────── tidyverse_conflict
s() ──
## ✕ dplyr::filter() masks stats::filter()
## ✕ dplyr::lag()    masks stats::lag()
## Loading required package: lattice
##
##
## Attaching package: 'caret'
##
##
## The following object is masked from 'package:purrr':
##
##     lift
##
##
##
## Attaching package: 'MASS'
##
##
## The following object is masked from 'package:dplyr':
##
##     select

#Load Data
data <- read.table("C:\\Users\\User\\OneDrive\\Desktop\\Data 11.1\\uscrime.tx
t", stringsAsFactors = F, header = T)

#Scale Data
scaledD <- as.data.frame(scale(data[,c(1,3,4, 5, 6, 7, 8, 9, 10, 11, 12, 13,
14, 15)]))
scaledD <- cbind(data[,2], scaledD, data[,16])
colnames(scaledD)[1] <- "So"
colnames(scaledD)[16] <- "Crime"

#Stepwise regression
```

```
data_model <- lm(Crime~., data = scaledD)
step_model <- stepAIC(data_model, direction = "both", trace = F)
summary(step_model)

##
## Call:
## lm(formula = Crime ~ M + Ed + Po1 + M.F + U1 + U2 + Ineq + Prob,
##     data = scaledD)
##
## 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)   905.09      28.52  31.731  < 2e-16 ***
## M             117.28      42.10   2.786  0.00828 **
## Ed            201.50      59.02   3.414  0.00153 **
## Po1           305.07      46.14   6.613 8.26e-08 ***
## M.F            65.83      40.08   1.642  0.10874
## U1           -109.73      60.20  -1.823  0.07622 .
## U2            158.22      61.22   2.585  0.01371 *
## Ineq          244.70      55.69   4.394 8.63e-05 ***
## Prob          -86.31      33.89  -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

#Identifying best number of predictors
set.seed(1)

#10-fold CV
train.control <- trainControl(method = "cv", number = 10)

#Training model
step.model <- train(Crime~., data = scaledD, method = "leapSeq", tuneGrid = d
ata.frame(nvmax = 1:15), trControl = train.control)

step.model$results

##    nvmax     RMSE  Rsquared      MAE    RMSESD RsquaredSD    MAESD
## 1      1 271.2008 0.5843691 226.2581 110.47396  0.3371720 96.68780
## 2      2 254.3833 0.6064511 202.4499  84.83446  0.2805282 64.11189
## 3      3 238.7279 0.7033882 186.3716  88.72517  0.1011618 59.18009
## 4      4 257.3026 0.7051499 200.2685 100.10472  0.1940803 76.05679
## 5      5 252.0782 0.6269684 201.9099  91.52018  0.2311683 70.23161
## 6      6 232.8635 0.7854913 184.9587 102.32196  0.1476856 81.55553
```

```
## 7      7 240.1809 0.7294113 190.5553  95.43075  0.1820599 74.00488
## 8      8 258.5924 0.6986070 202.8368 111.99607  0.2032220 90.26335
## 9      9 260.8053 0.6076826 210.0503 108.82448  0.1764839 89.03990
## 10    10 267.2403 0.6709714 216.7676  96.06399  0.2069373 76.76546
## 11    11 240.1841 0.6456568 201.6880  85.04252  0.2280547 71.83940
## 12    12 273.1258 0.6852765 222.0306 116.50459  0.2130656 93.46969
## 13    13 246.1732 0.5977355 199.2895  85.79255  0.2451306 69.47176
## 14    14 245.9982 0.5961171 195.7384  90.27009  0.2680203 68.68076
## 15    15 247.1168 0.5898489 196.0868  87.86030  0.2573792 67.76324
```

```
step.model$bestTune
```

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

```
summary(step.model$finalModel)
```

```
## Subset selection object
## 15 Variables  (and intercept)
##        Forced in Forced out
## So         FALSE      FALSE
## M          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'
##          So  M   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 ) " " "*" "*" "*" " " " " " " " " " " " " "*" " "    "*"  "*"  " "
```

```
coef(step.model$finalModel, 6)
```

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

```r
#Check CV with LOOC
#Using all variables
TotSS <- sum((data$Crime - mean(data$Crime))^2)
TotSSE <- 0

for (i in 1:nrow(scaledD)){
  fit_step_i = lm(Crime ~ M+Ed+Po1+M.F+U1+U2+Ineq+Prob, data=scaledD[-i,])
  pred_i <- predict(fit_step_i, newdata=scaledD[i,])
  TotSSE <- TotSSE + ((pred_i - data[i, 16])^2)
}

R2_all <- 1 - TotSSE/TotSS
R2_all

##        1
## 0.667621

#Using only significant variables
TotSS <- sum((data$Crime - mean(data$Crime))^2)
TotSSE <- 0

for (i in 1:nrow(scaledD)){
  fit_step_i = lm(Crime ~ M+Ed+Po1+U2+Ineq+Prob, data=scaledD[-i,])
  pred_i <- predict(fit_step_i, newdata=scaledD[i,])
  TotSSE <- TotSSE + ((pred_i - data[i, 16])^2)
}

R2_sig <- 1 - TotSSE/TotSS
R2_sig

##         1
## 0.6661638
```

Question 11.1.2

```r
#This is the lasso regression approach

#Load Packages
library(tidyverse); library(caret); library(leaps); library(MASS); library(gl
mnet)

## Loading required package: Matrix

##
## Attaching package: 'Matrix'

## The following objects are masked from 'package:tidyr':
##
##     expand, pack, unpack

## Loaded glmnet 4.1-6
```

```r
#Load Data
data <- read.table("C:\\Users\\User\\OneDrive\\Desktop\\Data 11.1\\uscrime.tx
t", stringsAsFactors = F, header = T)

#Scale Data
scaledD <- as.data.frame(scale(data[,c(1,3,4, 5, 6, 7, 8, 9, 10, 11, 12, 13,
14, 15)]))
scaledD <- cbind(data[,2], scaledD, data[,16])
colnames(scaledD)[1] <- "So"
colnames(scaledD)[16] <- "Crime"

set.seed(1)

#Lasso regression

lasso <- cv.glmnet(x=as.matrix(scaledD[,-16]), y=as.matrix(scaledD$Crime), al
pha=1, nfolds = 10, type.measure="mse", family = "gaussian")

plot(lasso)

lasso$lambda.min

## [1] 9.237784

#Use value of lambda that gives min. CVM
coef(lasso, s=lasso$lambda.min)

## 16 x 1 sparse Matrix of class "dgCMatrix"
##                     s1
## (Intercept) 889.944466
## So           44.475632
## M            89.288569
## Ed          138.032503
## Po1         305.238076
## Po2           .
## LF            .
## M.F          55.156773
## Pop           .
## NW            6.192810
## U1          -36.329654
## U2           71.884874
## Wealth        4.808914
## Ineq        191.893564
## Prob        -83.561850
## Time          .

#Fit new model with new variables
fit_lasso <- lm(Crime ~ So+M+Ed+Po1+M.F+NW+U1+U2+Wealth+Ineq+Prob, data=scale
dD)

summary(fit_lasso)
```

```
##
## Call:
## lm(formula = Crime ~ So + M + Ed + Po1 + M.F + NW + U1 + U2 +
##     Wealth + Ineq + Prob, data = scaledD)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -408.38  -96.14   -1.39  114.80  454.53
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)   893.73      51.33  17.411  < 2e-16 ***
## So             33.35     123.69   0.270  0.78905
## M             114.97      48.92   2.350  0.02454 *
## Ed            195.31      62.52   3.124  0.00357 **
## Po1           275.69      59.99   4.596 5.41e-05 ***
## M.F            64.50      42.82   1.506  0.14101
## NW             15.93      57.16   0.279  0.78209
## U1            -94.61      64.90  -1.458  0.15380
## U2            140.81      66.32   2.123  0.04089 *
## Wealth         73.59      93.96   0.783  0.43878
## Ineq          267.01      80.66   3.310  0.00217 **
## Prob          -87.64      40.25  -2.177  0.03627 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 201.3 on 35 degrees of freedom
## Multiple R-squared:  0.794,  Adjusted R-squared:  0.7292
## F-statistic: 12.26 on 11 and 35 DF,  p-value: 5.334e-09
```

```r
#Check CV with LOOC
#Using all variables
TotSS <- sum((data$Crime - mean(data$Crime))^2)
TotSSE <- 0

for (i in 1:nrow(scaledD)){
  fit_lasso_i = lm(Crime ~ So+M+Ed+Po1+M.F+NW+U1+U2+Wealth+Ineq+Prob, data=sc
aledD[-i,])
  pred_i <- predict(fit_lasso_i, newdata=scaledD[i,])
  TotSSE <- TotSSE + ((pred_i - data[i, 16])^2)
}

R2_all <- 1 - TotSSE/TotSS
R2_all
```

```
##         1
## 0.6051577
```

```r
#Using only significant variables
TotSS <- sum((data$Crime - mean(data$Crime))^2)
```

```
TotSSE <- 0

for (i in 1:nrow(scaledD)){
  fit_lasso_i = lm(Crime ~ M+Ed+Po1+U2+Ineq+Prob, data=scaledD[-i,])
  pred_i <- predict(fit_lasso_i, newdata=scaledD[i,])
  TotSSE <- TotSSE + ((pred_i - data[i, 16])^2)
}

R2_sig <- 1 - TotSSE/TotSS
R2_sig

##         1
## 0.6661638
```

Question 11.1.3

```
#This is the elastic net regression approach

#Load Packages
library(tidyverse); library(caret); library(leaps); library(MASS); library(gl
mnet)

#Load Data
data <- read.table("C:\\Users\\User\\OneDrive\\Desktop\\Data 11.1\\uscrime.tx
t", stringsAsFactors = F, header = T)

#Scale Data
scaledD <- as.data.frame(scale(data[,c(1,3,4, 5, 6, 7, 8, 9, 10, 11, 12, 13,
14, 15)]))
scaledD <- cbind(data[,2], scaledD, data[,16])
colnames(scaledD)[1] <- "So"
colnames(scaledD)[16] <- "Crime"

set.seed(1)
#Vary alpha values, calculate R-squared
R2_el <- c()

for (i in 0:10){
  fit_elastic <- cv.glmnet(x=as.matrix(scaledD[,-16]), y=as.matrix(scaledD$Cr
ime), alpha = i/10, nfolds = 10, type.measure="mse", family="gaussian")

  R2_el <- cbind(R2_el, fit_elastic$glmnet.fit$dev.ratio[which(fit_elastic$gl
mnet.fit$lambda==fit_elastic$lambda.min)])
}

R2_el

##           [,1]      [,2]      [,3]      [,4]      [,5]      [,6]      [,7]
## [1,] 0.7450772 0.7490443 0.7446379 0.7167649 0.7853621 0.7936385 0.7597881
##           [,8]      [,9]     [,10]     [,11]
## [1,] 0.7940936 0.7705495 0.7941416 0.7682889
```

```r
#The best alpha value is:
best_alpha <- (which.max(R2_el)-1)/10
best_alpha

## [1] 0.9

#Build model using best alpha
elastic <- cv.glmnet(x=as.matrix(scaledD[,-16]), y=as.matrix(scaledD$Crime),
alpha = best_alpha, nfolds = 10, type.measure = "mse", family="gaussian")

coef(elastic, s=elastic$lambda.min)

## 16 x 1 sparse Matrix of class "dgCMatrix"
##                     s1
## (Intercept) 894.24787
## So           31.83438
## M           104.63624
## Ed          175.94404
## Po1         295.00439
## Po2           .
## LF            .
## M.F          52.69543
## Pop         -19.58677
## NW           15.25481
## U1          -72.62608
## U2          117.92638
## Wealth       55.75815
## Ineq        251.09728
## Prob        -89.64831
## Time          .

#Use important variables to create new model

elastic_imp <- lm(Crime~So+M+Ed+Po1+M.F+Pop+NW+U1+U2+Wealth+Ineq+Prob, data=s
caledD)

summary(elastic_imp)

##
## Call:
## lm(formula = Crime ~ So + M + Ed + Po1 + M.F + Pop + NW + U1 +
##     U2 + Wealth + Ineq + Prob, data = scaledD)
##
## 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 ***
## So             22.89     125.35   0.183  0.85621
```

```
## M              112.71      49.35   2.284  0.02876 *  
## 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
```

```
#Check CV with LOOC
#Using all variables
TotSS <- sum((data$Crime - mean(data$Crime))^2)
TotSSE <- 0

for (i in 1:nrow(scaledD)){
  fit_elastic_i = lm(Crime~So+M+Ed+Po1+M.F+Pop+NW+U1+U2+Wealth+Ineq+Prob, dat
a=scaledD[-i,])
  pred_i <- predict(fit_elastic_i, newdata=scaledD[i,])
  TotSSE <- TotSSE + ((pred_i - data[i, 16])^2)
}

R2_all <- 1 - TotSSE/TotSS
R2_all
```

```
##         1 
## 0.5903894
```

```
#Using only significant variables
TotSS <- sum((data$Crime - mean(data$Crime))^2)
TotSSE <- 0

for (i in 1:nrow(scaledD)){
  fit_elastic_i = lm(Crime~M+Ed+Po1+U2+Ineq+Prob, data=scaledD[-i,])
  pred_i <- predict(fit_elastic_i, newdata=scaledD[i,])
  TotSSE <- TotSSE + ((pred_i - data[i, 16])^2)
}

R2_sig <- 1 - TotSSE/TotSS
R2_sig
```

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