Homework8

N/A

2023-10-15

```r
library(caret)
```

```
## Loading required package: ggplot2
```

```
## Loading required package: lattice
```

```r
library(glmnet)
```

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

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

```r
library(ggplot2)
library(leaps)
library(MASS)
library(olsrr)
```

```
##
## Attaching package: 'olsrr'
```

```
## The following object is masked from 'package:MASS':
##
##     cement
```

```
## The following object is masked from 'package:datasets':
##
##     rivers
```

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.

I read in the dataset and run some basic summary statistics.

```r
crime <- read.table("uscrime.csv", header = TRUE)

summary(crime)
```

```
##        M               So               Ed             Po1
##  Min.   :11.90   Min.   :0.0000   Min.   : 8.70   Min.   : 4.50
##  1st Qu.:13.00   1st Qu.:0.0000   1st Qu.: 9.75   1st Qu.: 6.25
##  Median :13.60   Median :0.0000   Median :10.80   Median : 7.80
##  Mean   :13.86   Mean   :0.3404   Mean   :10.56   Mean   : 8.50
##  3rd Qu.:14.60   3rd Qu.:1.0000   3rd Qu.:11.45   3rd Qu.:10.45
##  Max.   :17.70   Max.   :1.0000   Max.   :12.20   Max.   :16.60
##       Po2              LF              M.F              Pop
##  Min.   : 4.100   Min.   :0.4800   Min.   : 93.40   Min.   :  3.00
##  1st Qu.: 5.850   1st Qu.:0.5305   1st Qu.: 96.45   1st Qu.: 10.00
##  Median : 7.300   Median :0.5600   Median : 97.70   Median : 25.00
##  Mean   : 8.023   Mean   :0.5612   Mean   : 98.30   Mean   : 36.62
##  3rd Qu.: 9.700   3rd Qu.:0.5930   3rd Qu.: 99.20   3rd Qu.: 41.50
##  Max.   :15.700   Max.   :0.6410   Max.   :107.10   Max.   :168.00
##        NW              U1                U2            Wealth
##  Min.   : 0.20   Min.   :0.07000   Min.   :2.000   Min.   :2880
##  1st Qu.: 2.40   1st Qu.:0.08050   1st Qu.:2.750   1st Qu.:4595
##  Median : 7.60   Median :0.09200   Median :3.400   Median :5370
##  Mean   :10.11   Mean   :0.09547   Mean   :3.398   Mean   :5254
##  3rd Qu.:13.25   3rd Qu.:0.10400   3rd Qu.:3.850   3rd Qu.:5915
##  Max.   :42.30   Max.   :0.14200   Max.   :5.800   Max.   :6890
##       Ineq            Prob              Time           Crime
##  Min.   :12.60   Min.   :0.00690   Min.   :12.20   Min.   : 342.0
##  1st Qu.:16.55   1st Qu.:0.03270   1st Qu.:21.60   1st Qu.: 658.5
##  Median :17.60   Median :0.04210   Median :25.80   Median : 831.0
##  Mean   :19.40   Mean   :0.04709   Mean   :26.60   Mean   : 905.1
##  3rd Qu.:22.75   3rd Qu.:0.05445   3rd Qu.:30.45   3rd Qu.:1057.5
##  Max.   :27.60   Max.   :0.11980   Max.   :44.00   Max.   :1993.0
```

```r
head(crime)
```

```
##      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
```

The code defines a function called split_data for dividing a dataset into training and testing subsets. It takes the input data and a specified training size (defaulting to 80%). The function calculates the split point based on the training size, then uses the split function to split the data into "train" and "test" subsets. The function returns either the training or testing subset based on the is_training parameter.

Subsequently, this function is applied to the crime dataset, creating two subsets: crime_train for training data and crime_test for testing data. The training data is used to build the model, while the testing data is used to assess the model's performance on unseen examples.

```r
split_data <- function(input_data, train_size = 0.8, is_training = TRUE) {
  n_rows <- nrow(input_data)
  split_row <- train_size * n_rows
  split_index <- round(split_row)

  split_data <- split(input_data, ifelse(seq_len(n_rows) <= split_index, "train", "test"))

  if (is_training) {
    return(split_data[["train"]])
  } else {
    return(split_data[["test"]])
  }
}

crime_train <- split_data(crime, 0.8, is_training = TRUE)
crime_test <- split_data(crime, 0.8, is_training = FALSE)
```

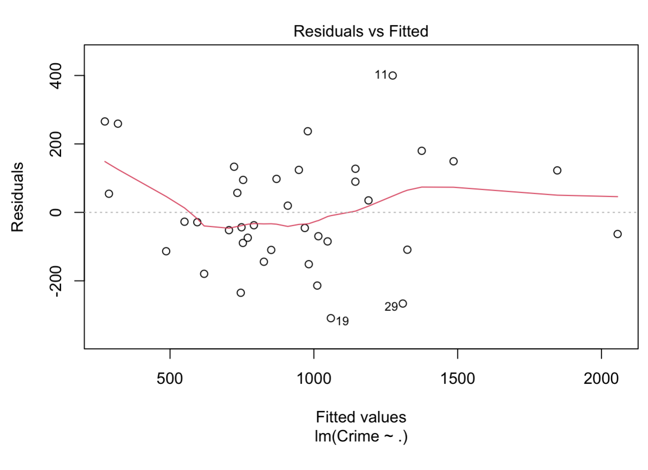
Here I run a linear reg model without doing anything:

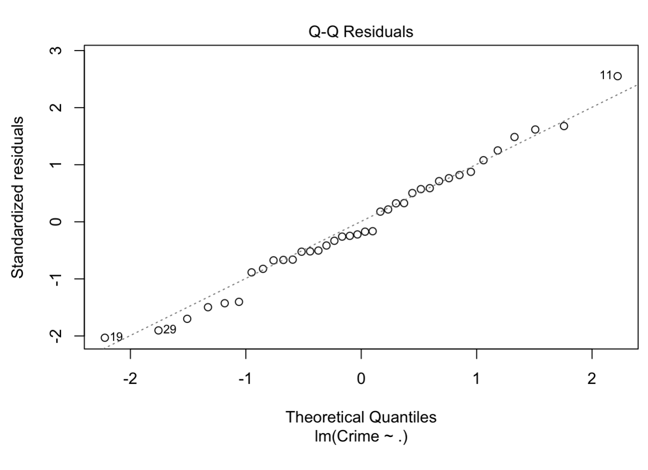
```r
uscrime_model <- lm(Crime~., data = crime_train)
summary(uscrime_model)
```

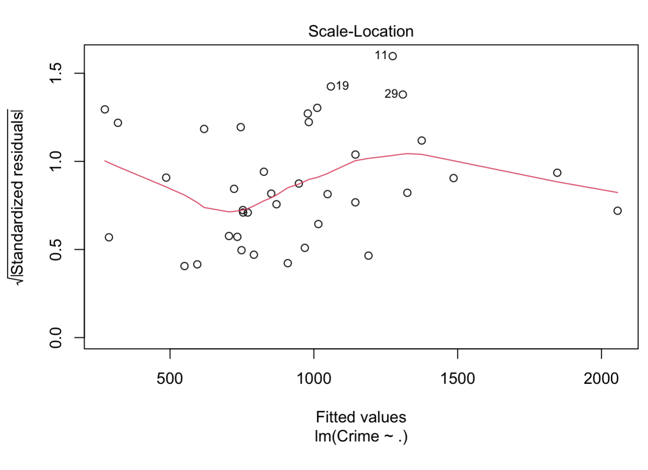
```
##
## Call:
## lm(formula = Crime ~ ., data = crime_train)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -309.13 -104.13  -33.14  116.47  399.91
##
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept) -7594.8378  2164.2749  -3.509  0.00198 **
## M              74.5493    44.8335   1.663  0.11054
## So            100.5641   174.7852   0.575  0.57089
## Ed            150.1609    70.5539   2.128  0.04476 *
## Po1           182.6350   157.0649   1.163  0.25737
## Po2          -103.8188   174.9515  -0.593  0.55896
## LF          -1109.8071  1693.0149  -0.656  0.51893
## M.F            37.0280    23.1726   1.598  0.12432
## Pop            -0.4669     1.4560  -0.321  0.75147
## NW              2.1385     7.4178   0.288  0.77582
## U1          -6068.4033  4644.5716  -1.307  0.20486
## U2            131.5777    91.6661   1.435  0.16524
## Wealth          0.1801     0.1097   1.641  0.11504
## Ineq           81.9726    24.7287   3.315  0.00315 **
## Prob        -5406.1560  2638.0973  -2.049  0.05255 .
## Time           -0.6905     9.3250  -0.074  0.94164
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 205.2 on 22 degrees of freedom
## Multiple R-squared:  0.8521, Adjusted R-squared:  0.7512
## F-statistic: 8.448 on 15 and 22 DF,  p-value: 5.794e-06
```

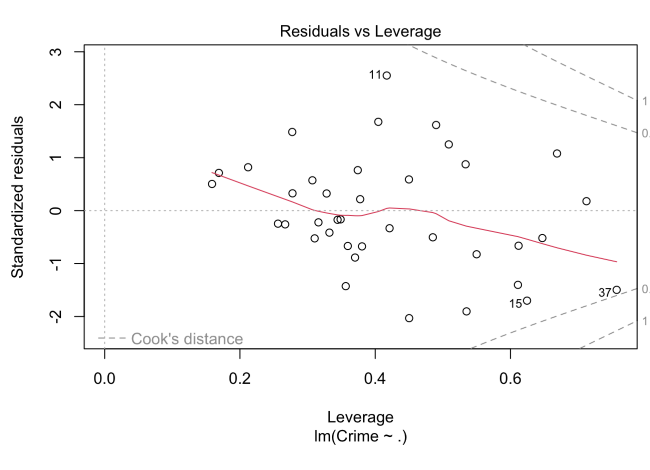
The R^2 of this model is 0.8 which isn't too bad. Observing some basic statistics about our dataset.

```r
plot(uscrime_model)
```









1) Stepwise Regression.

```r
stepwise_model <- train(Crime ~., data = crime_train,
                        method = "lmStepAIC",
                        trControl = trainControl(),
                        trace = FALSE)


stepwise_model$results
```

```
##   parameter    RMSE  Rsquared      MAE   RMSESD RsquaredSD    MAESD
## 1      none 382.168 0.4565835 298.6403 109.1995  0.1956438 88.66819
```

```r
summary(stepwise_model$finalModel)
```

```
##
## Call:
## lm(formula = .outcome ~ M + Ed + Po1 + M.F + U1 + U2 + Wealth +
##     Ineq + Prob, data = dat)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -344.45  -90.59  -32.53  124.69  355.91
##
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept) -8.079e+03  1.367e+03  -5.911 2.32e-06 ***
## M            8.929e+01  3.819e+01   2.338 0.026750 *
## Ed           1.274e+02  5.925e+01   2.150 0.040345 *
## Po1          9.277e+01  1.864e+01   4.977 2.95e-05 ***
## M.F          3.460e+01  1.481e+01   2.336 0.026875 *
## U1          -6.295e+03  3.494e+03  -1.802 0.082381 .
## U2           1.659e+02  7.825e+01   2.121 0.042937 *
## Wealth       1.581e-01  9.836e-02   1.608 0.119112
## Ineq         8.303e+01  1.943e+01   4.274 0.000201 ***
## Prob        -3.659e+03  1.607e+03  -2.277 0.030630 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 189.8 on 28 degrees of freedom
## Multiple R-squared:  0.8389, Adjusted R-squared:  0.7872
## F-statistic: 16.21 on 9 and 28 DF,  p-value: 6.605e-09
```

From the summary of our model, our R squared value has decreased slightly. I now move on to lasso and elastic net.

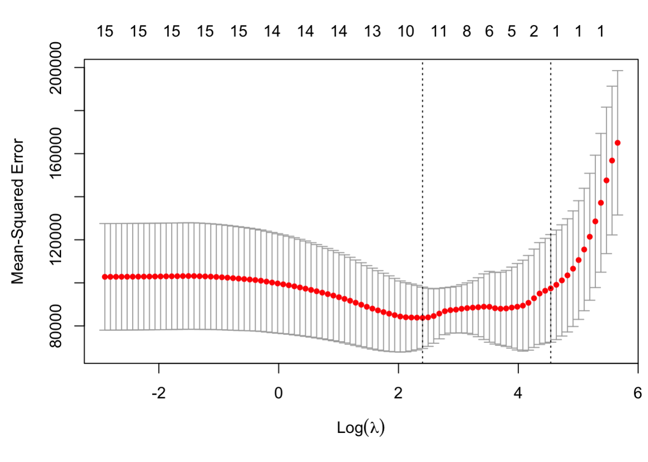
2) Lasso Model

Now I experiment with the lasso model. First we need to scale our values, as stated in the instructions.

```r
uscrime_scale <- cbind(as.data.frame(scale(crime_train[,1])),
                       as.data.frame(crime_train[,2]),
                       as.data.frame(scale(crime_train[,c(3,4,5,6,7,8,9,10,11,12,13,14,15)])),
                       as.data.frame(crime_train[,16]))
```

Adjust column names

```r
colnames(uscrime_scale) <- colnames(crime_train)
uscrime_lasso <- cv.glmnet(x=as.matrix(uscrime_scale[,-16]), y=as.matrix(uscrime_scale$Crime),
                           alpha=1, nfolds = 5, type.measure="mse", family="gaussian")
plot(uscrime_lasso)
```



Here, I attempt to find the lowest lambda value because that would be the one with the lowest

```r
best_lambda_min <- uscrime_lasso$lambda.min
best_lambda_1se <- uscrime_lasso$lambda.1se
```

Here I find display the coefficients and use the scaled coefficients in another linear model.

```r
coefficients(uscrime_lasso, best_lambda_min)
```

```
## 16 x 1 sparse Matrix of class "dgCMatrix"
##                    s1
## (Intercept) 900.92446
## M            66.98995
## So          103.32255
## Ed           84.98754
## Po1         318.69762
## Po2           .
## LF            .
## M.F          98.11665
## Pop           .
## NW            .
## U1          -43.71647
## U2           53.25360
## Wealth       71.10063
## Ineq        224.77158
## Prob        -99.41184
## Time          .
```

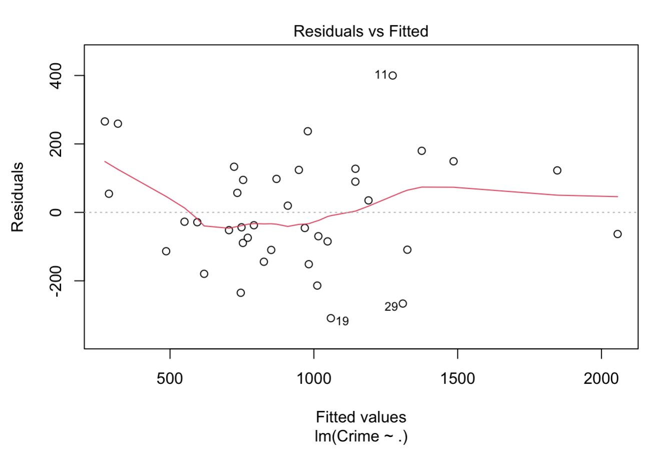
```r
lasso_coef <- predict(uscrime_lasso, s = "lambda.min", type = "coefficients")
uscrime_lasso_lm <- lm(Crime ~., data = crime_train)
```

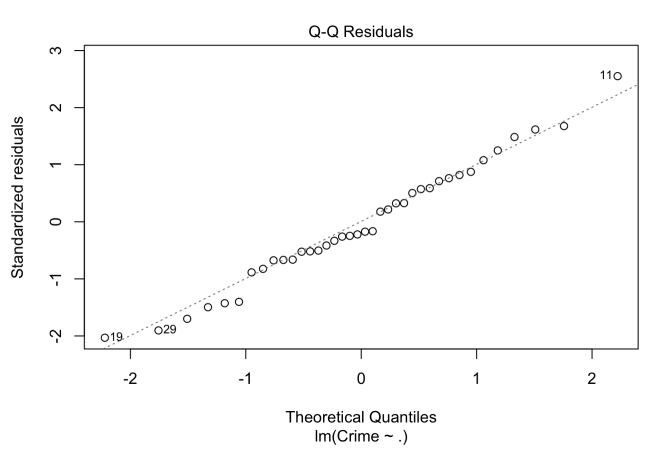
We can see that the R^2 value has improved and we obtain a value of 0.85

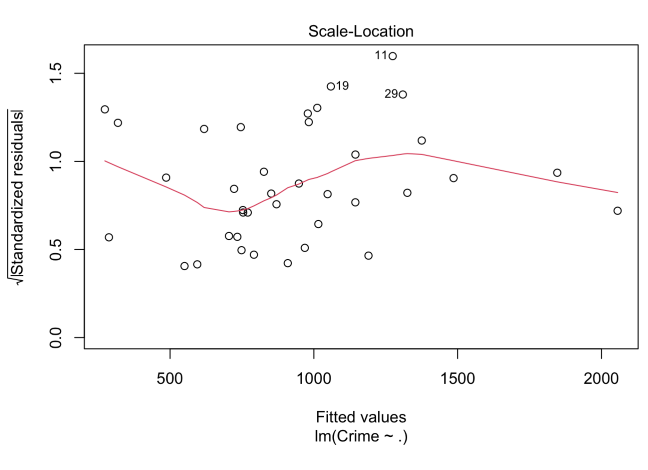
```r
summary(uscrime_lasso_lm)
```

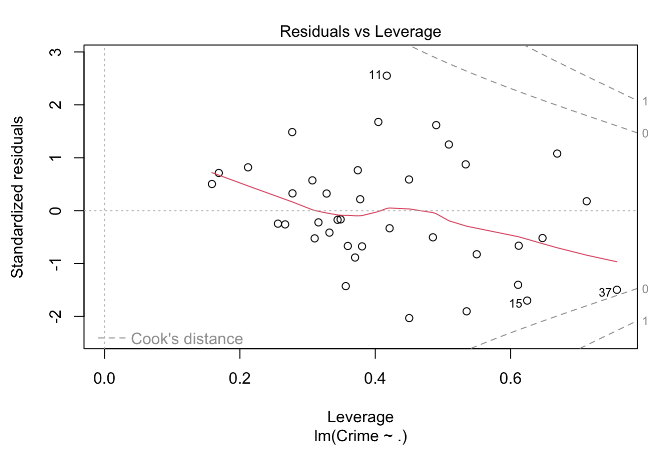
```
##
## Call:
## lm(formula = Crime ~ ., data = crime_train)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -309.13 -104.13  -33.14  116.47  399.91
##
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept) -7594.8378  2164.2749  -3.509  0.00198 **
## M              74.5493    44.8335   1.663  0.11054
## So            100.5641   174.7852   0.575  0.57089
## Ed            150.1609    70.5539   2.128  0.04476 *
## Po1           182.6350   157.0649   1.163  0.25737
## Po2          -103.8188   174.9515  -0.593  0.55896
## LF          -1109.8071  1693.0149  -0.656  0.51893
## M.F            37.0280    23.1726   1.598  0.12432
## Pop            -0.4669     1.4560  -0.321  0.75147
## NW              2.1385     7.4178   0.288  0.77582
## U1          -6068.4033  4644.5716  -1.307  0.20486
## U2            131.5777    91.6661   1.435  0.16524
## Wealth          0.1801     0.1097   1.641  0.11504
## Ineq           81.9726    24.7287   3.315  0.00315 **
## Prob        -5406.1560  2638.0973  -2.049  0.05255 .
## Time           -0.6905     9.3250  -0.074  0.94164
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 205.2 on 22 degrees of freedom
## Multiple R-squared:  0.8521, Adjusted R-squared:  0.7512
## F-statistic: 8.448 on 15 and 22 DF,  p-value: 5.794e-06
```

```r
plot(uscrime_lasso_lm)
```









3). Elastic Net

First I standardize both the predictor variables and the response variable from the training data. Then set up a loop to build a set of Lasso regression models, each with a different alpha value ranging from 0 to 1 in increments of 0.1. For each alpha, the code constructs a model name, performs cross-validated Lasso regression using the standardized data, and stores the results in a list. This allows for the creation of multiple Lasso models with varying levels of L1 regularization, which can later be compared to select the most suitable alpha value for optimal model performance and feature selection. The code here iterates over a range of alpha values from 0 to 1 in steps of 0.1. For each alpha value, it constructs a model name, performs Lasso regression, and calculates the Mean Squared Error (MSE) on a test dataset. The results, including alpha values, MSE, model names, and the optimal lambda (lambda.min) values, are collected in a data frame named results. This allows for the comparison of Lasso regression models with varying degrees of L1 (Lasso) regularization, making it possible to identify which alpha value produces the best-fitting model in terms of minimizing prediction errors.

```r
alphas <- seq(0, 1, by = 0.1)
r2 <- numeric(length(alphas))

cv_results <- lapply(alphas, function(alpha) {
  mod_uscrime_elastic <- cv.glmnet(
    x = as.matrix(uscrime_scale[, -16]),
    y = as.matrix(uscrime_scale$Crime),
    alpha = alpha,
    nfolds = 5,
    type.measure = "mse",
    family = "gaussian"
  )
  best_lambda_index <- which.min(mod_uscrime_elastic$glmnet.fit$dev.ratio)
  r2 <- mod_uscrime_elastic$glmnet.fit$dev.ratio[best_lambda_index]
  return(list(alpha = alpha, r2 = r2))
})

cv_results_df <- do.call(rbind, cv_results)
best_alpha <- cv_results_df$alpha[which.max(cv_results_df$r2)]



uscrime_elastic <- cv.glmnet(x=as.matrix(uscrime_scale[,-16]),
                             y=as.matrix(uscrime_scale$Crime), alpha=0.05,
                             nfolds = 5, type.measure="mse", family="gaussian")

uscrime_elastic_lm = lm(Crime ~M+So+Ed+Po1+Po2+LF+M.F+NW+U1+U2+Wealth+Ineq+Prob
                        +Time, data = uscrime_scale)
summary(uscrime_elastic_lm)
```

```
##
## Call:
## lm(formula = Crime ~ M + So + Ed + Po1 + Po2 + LF + M.F + NW +
##     U1 + U2 + Wealth + Ineq + Prob + Time, data = uscrime_scale)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -298.65  -99.86  -37.07   99.23  398.57
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)   901.13      63.16  14.268  6.5e-13 ***
## M              98.69      56.59   1.744  0.09450 .
## So            102.67     171.22   0.600  0.55461
## Ed            163.81      74.75   2.192  0.03880 *
## Po1           598.88     479.17   1.250  0.22393
## Po2          -341.66     490.75  -0.696  0.49328
## LF            -48.52      62.76  -0.773  0.44733
## M.F           119.21      62.00   1.923  0.06700 .
## NW             26.05      78.13   0.333  0.74182
## U1           -114.37      77.89  -1.468  0.15557
## U2            107.99      74.27   1.454  0.15942
## Wealth        179.73     108.78   1.652  0.11208
## Ineq          317.99      92.31   3.445  0.00221 **
## Prob         -131.65      60.81  -2.165  0.04101 *
## Time          -11.56      60.21  -0.192  0.84940
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 201.2 on 23 degrees of freedom
## Multiple R-squared:  0.8514, Adjusted R-squared:  0.7609
## F-statistic: 9.411 on 14 and 23 DF,  p-value: 1.901e-06
```

After conducting an Elastic Net regression, the next steps involve evaluating the model's performance using metrics like Mean Squared Error or R-squared.As we see here, we still recieve a R squared value of around 0.85. This was not much different than our last few models so we can assume that we've maximized out model out and its most likely performing at its peak.