ISYE 6501

Homework 8

Artur Cabral, Marta Bras, Pedro Pinto, Katie Price

2019-10-12

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.

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.

1. Importing and initial analysis of the data

```
#reading the data
crime_data <- read.table("uscrime.txt", header = TRUE)

# Overall statistics
summary_table <- crime_data %>%
  summarize(number_states = nrow(crime_data),
            total_crime_rate = sum(Crime),
            average_crime_rate = mean(Crime),
            min_crime_rate = min(Crime),
            max_crime_rate = max(Crime))

kable(summary_table, caption = "Crime Rate - overall statistics") %>%
  kable_styling(bootstrap_options = c("striped", "hover", "condensed", "bordered"))
```

Table 1: Crime Rate - overall statistics

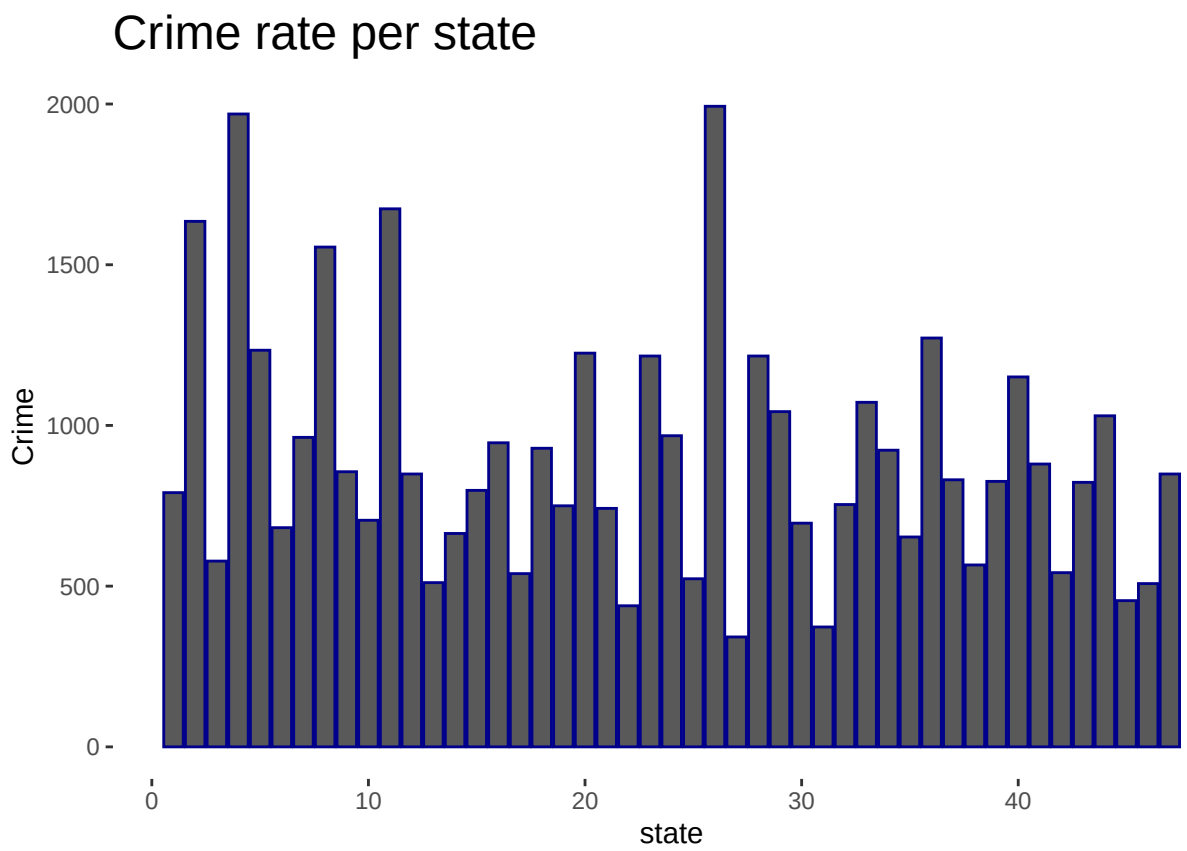
number_states	total_crime_rate	average_crime_rate	min_crime_rate	max_crime_rate
47	42539	905.0851	342	1993

Table 2: Crime Rate per state

So	number_states	total_crime_rate	average_crime_rate	min_crime_rate	max_crime_rate
0	47	28830	930.0000	342	1993
1	47	13709	856.8125	439	1555

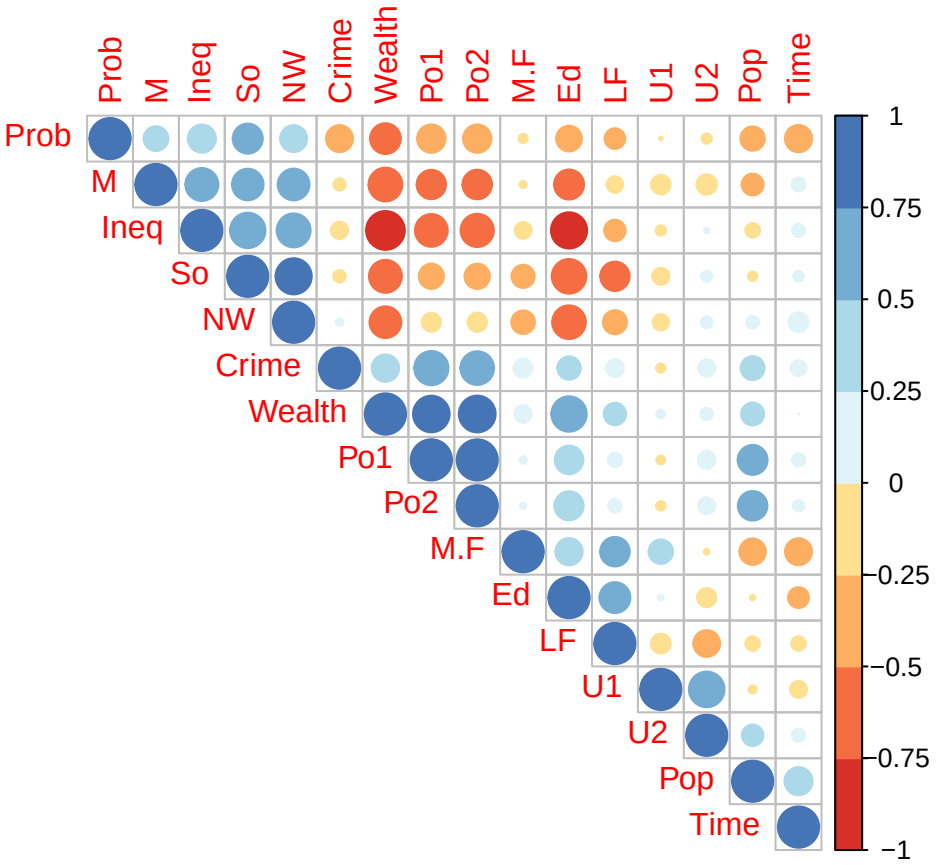
```
# Statistics per state
summary_table_state <- crime_data %>% group_by(So) %>%
  summarize(number_states = nrow(crime_data),
            total_crime_rate = sum(Crime),
            average_crime_rate = mean(Crime),
            min_crime_rate = min(Crime),
            max_crime_rate = max(Crime))

kable(summary_table_state, caption = "Crime Rate per state") %>%
  kable_styling(bootstrap_options = c("striped", "hover", "condensed", "bordered"))
```

```
# Crime rate per state
## adding an index variable
state <- seq(1, length(crime_data$So))
crime_data_2 <- cbind(state, crime_data)
state <- factor(crime_data_2$state)

ggplot(crime_data_2, aes(x=state, y = Crime)) +
  geom_col(color='darkblue') +
  ggtitle("Crime rate per state") +
  theme_bw() + theme(panel.border = element_blank(), panel.grid.major = element_blank(),
  panel.grid.minor = element_blank()) +
  theme(plot.title = element_text(size=18))
```



From the table above, we can see that the minimum crime rate between the 47 states is 342 crimes per 100,000 people. The maximum crime rate was 1993 per 100,000 people. The minimum and maximum crime rate are different from northern to southern states. We can also see some states have much higher crime rates than others.

```
#Correlations between variables

M <-cor(crime_data)
corrplot(M, type="upper", order="hclust",
         col=brewer.pal(n=8, name="RdYlBu"))
```

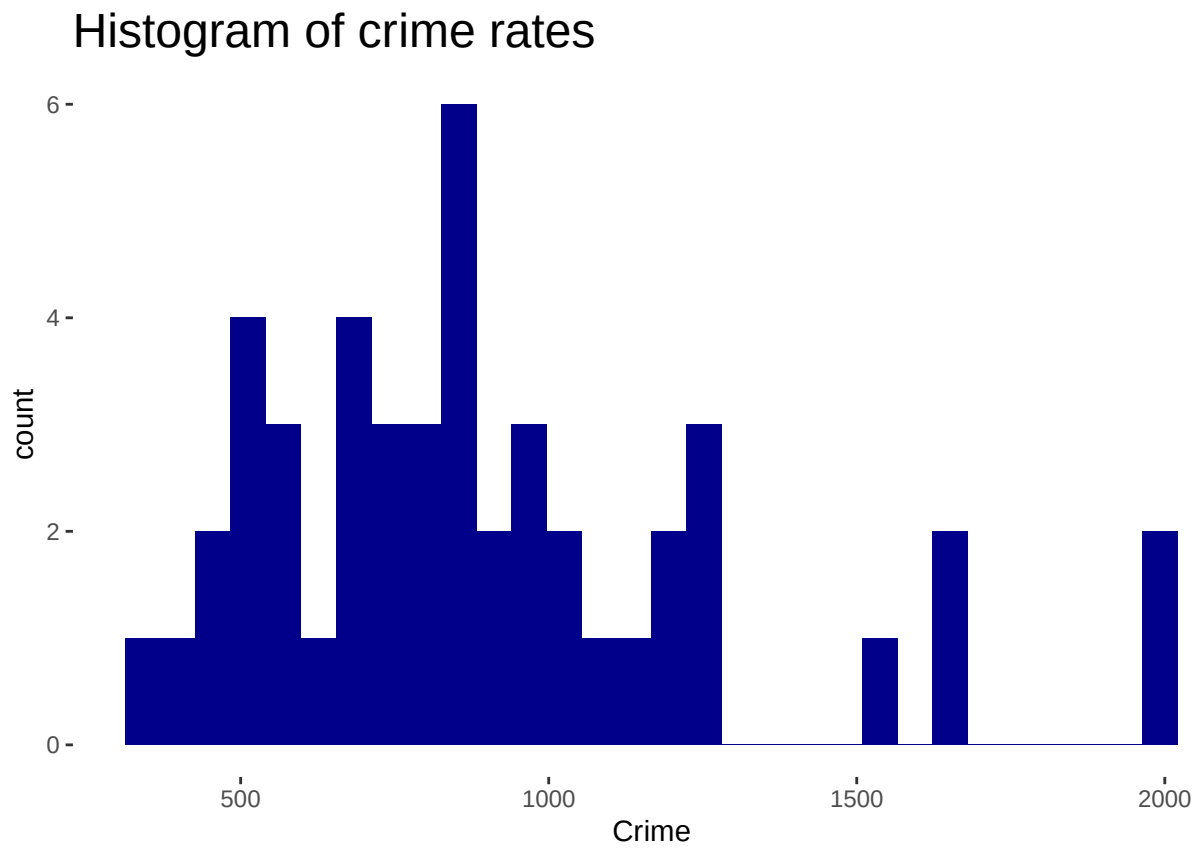


As we can see from the correlation matrix above, a lot of variables have correlations higher than 50% and some variables have correlations higher than 75%. Considering the small number of data points, the risk of over-fitting is especially high. For instances, variables Inequality and Wealth and variables Inequality and Education have correlations above 75%.

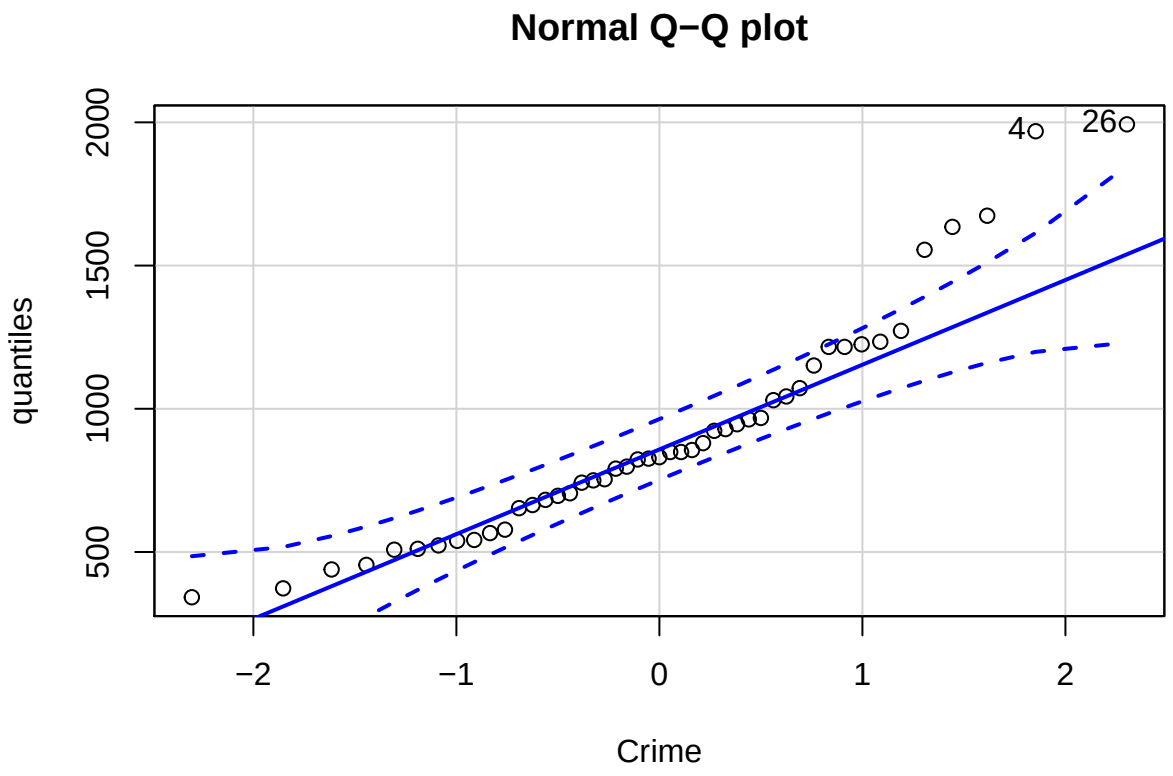
After doing a preliminary analysis we know things we should be aware when building the regression model: 1. Y variable is not normally distributed and there are outliers in the data, 2. There is strong correlation between possible explanatory variables.

```
#distribution of the y variable
# Histogram of crime
ggplot(crime_data, aes(x=Crime)) +
  geom_histogram(fill = "darkblue") +
  ggtitle("Histogram of crime rates") +
  theme_bw() + theme(panel.border = element_blank(), panel.grid.major = element_blank(),
  panel.grid.minor = element_blank()) +
  theme(plot.title = element_text(size=18))
```

```
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```

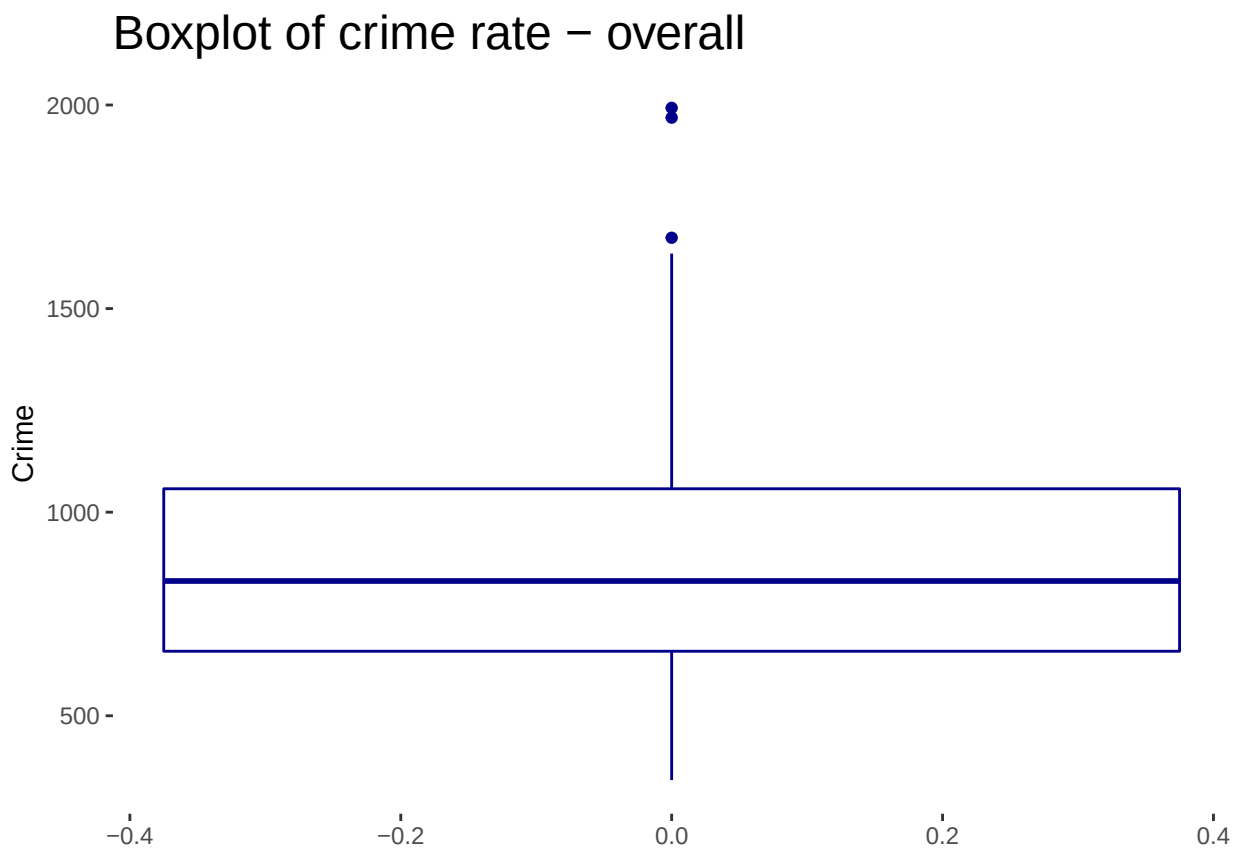



```
##QQplot graph of crime rate overall
qqPlot(crime_data$Crime, main = "Normal Q-Q plot", xlab = "Crime", ylab = "quantiles")
```




```
## [1] 26  4
```

```
# Boxplots of crime rate
## changing to factor variable to use in the histogram
crime_data$So <- factor(crime_data$So)

ggplot(crime_data, aes(y=Crime)) +
  geom_boxplot(color = "darkblue") +
  ggtitle("Boxplot of crime rate - overall") +
  theme_bw() + theme(panel.border = element_blank(), panel.grid.major = element_blank(),
  panel.grid.minor = element_blank()) +
  theme(plot.title = element_text(size=18))
```



The crime variable has a distribution with heavy tales on the right side, which is visible in both the qqplot and the histogram.

2. Stepwise regression

To perform variable selection, we can use stepwise regression. Stepwise regression can be performed forward - starting with no variables and adding variable by variable if they enhnace a specified criteria - or backward - starting with all variables and removing variable by variable if the removal enhances or has no impact on a specified criteria.

2.1. Leap package

One of the ways to perform stepwise regression in R is to use the regsubsets function under the leaps package.

It returns multiple models with different sizes up to the maximum number of variables defined. It can also be used in combination with the caret package in R, allowing to perform cross-validation with the train() function.

Since the algorithm returns a best model of each size, the results do not depend on a penalty model for model size: it doesn't make any difference to have an objective function (such as AIC, BIC,..)

```
# Set seed for reproducibility
set.seed(123)
# Set up repeated k-fold cross-validation
train.control <- trainControl(method = "cv", number = 10)
# Train the model
```

```r
step.model <- train(Crime ~., data = crime_data,
                    method = "leapForward",
                    tuneGrid = data.frame(nvmax = 1:15),
                    trControl = train.control
                    )

#results from the mddel for different number of variables
step.model$results
```

```
##    nvmax     RMSE  Rsquared      MAE    RMSESD RsquaredSD    MAESD
## 1      1 294.0919 0.6113474 234.5452  89.54470  0.3090369 68.56939
## 2      2 277.0405 0.7054625 221.7078 106.59142  0.2854178 80.58463
## 3      3 278.4634 0.6037461 227.3419 102.87138  0.2725414 88.68794
## 4      4 276.3418 0.6110869 225.2899 100.51338  0.2687307 74.23143
## 5      5 264.2073 0.6460780 213.6617 102.16483  0.2342649 77.70452
## 6      6 234.5125 0.7020166 193.9621 105.83208  0.2454595 77.42269
## 7      7 241.5600 0.6555468 202.1906  90.33335  0.2947141 67.65517
## 8      8 239.0965 0.6163437 203.4197  89.61133  0.3371332 67.33715
## 9      9 238.3927 0.5839176 201.1751  96.72091  0.3332297 68.92041
## 10    10 241.2421 0.6539396 201.7397  95.34818  0.2598476 68.44124
## 11    11 254.3227 0.6556660 208.1510 103.31189  0.3019813 75.21954
## 12    12 256.9434 0.6609192 211.8125 104.79749  0.2949388 78.13149
## 13    13 255.5330 0.6600182 207.0720 107.18448  0.3008627 79.15495
## 14    14 253.7609 0.6629858 204.6719 107.04647  0.2976913 80.26961
## 15    15 255.2604 0.6620912 207.1917 107.49455  0.2940859 81.54819
```

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

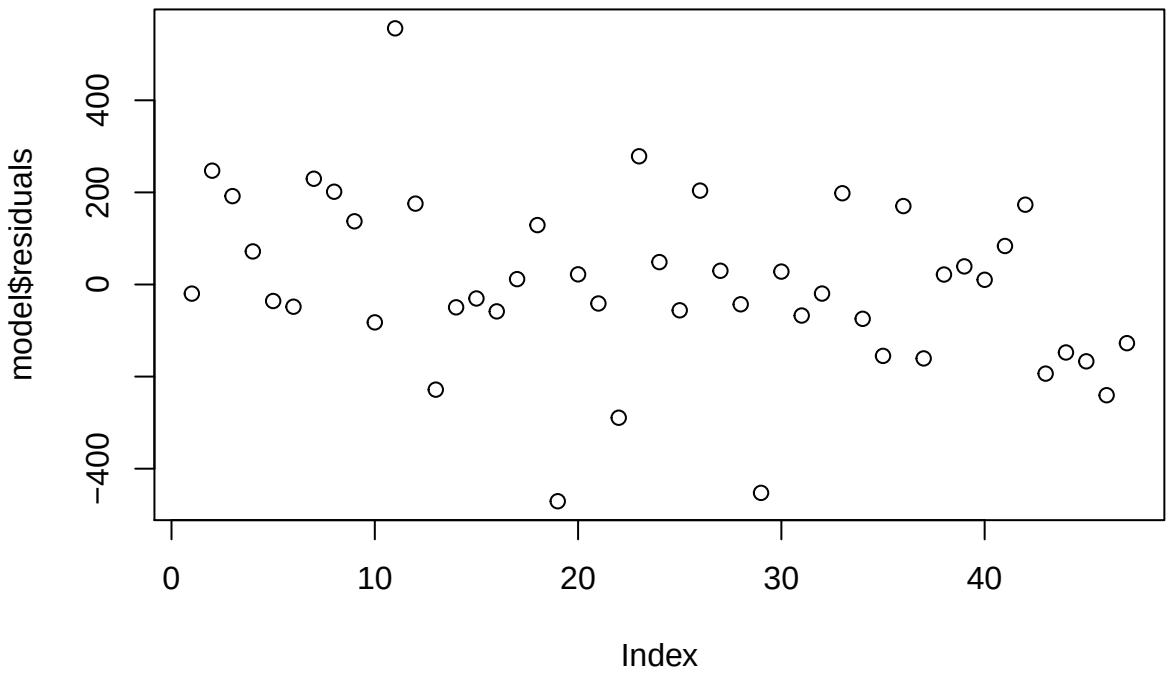
```
## Subset selection object
## 15 Variables  (and intercept)
##        Forced in Forced out
## M          FALSE      FALSE
## So1        FALSE      FALSE
## Ed         FALSE      FALSE
## Po1        FALSE      FALSE
## Po2        FALSE      FALSE
## LF         FALSE      FALSE
## M.F        FALSE      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: forward
##          M   So1 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 ) "*" " " "*" "*" " " " " " " " " " " " " " " " " " " "*" " "    "*"  "*"  " "
```

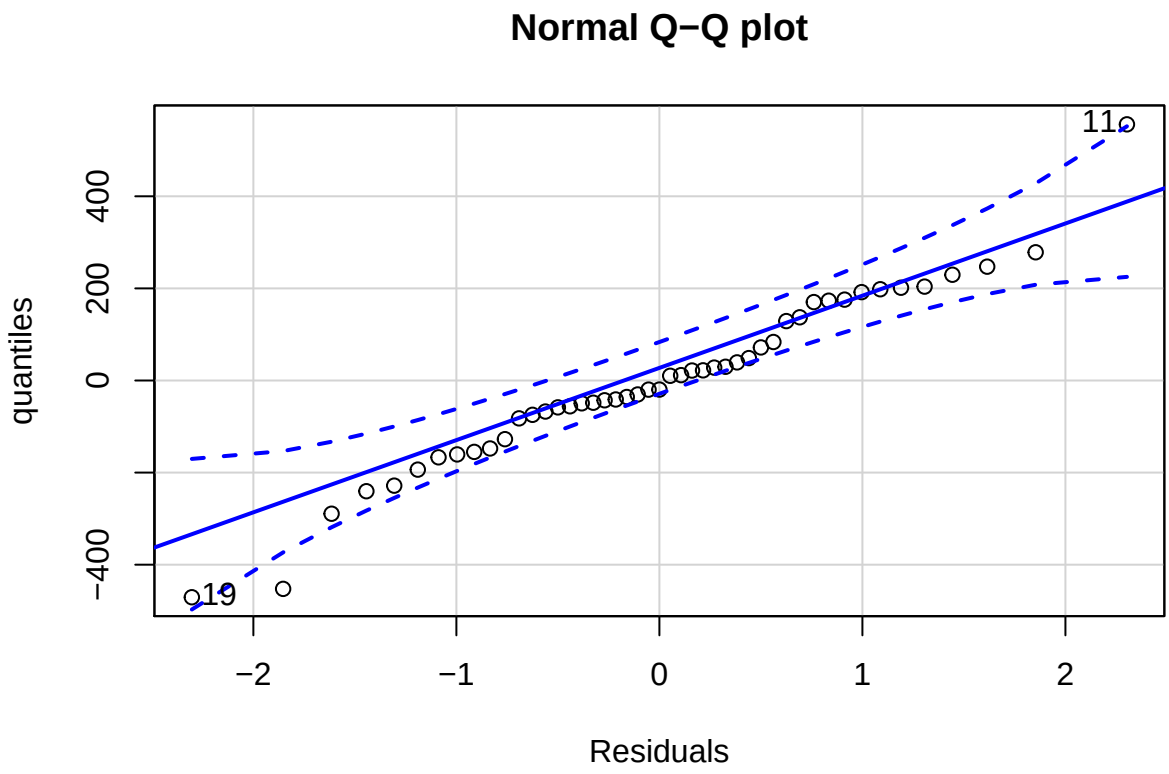
```
#model with selected variables
model <- lm(Crime ~ M + Ed + Po1 + U2 + Ineq + Prob, data = crime_data)
#summary
summary(model)
```

```
##
## Call:
## lm(formula = Crime ~ M + Ed + Po1 + U2 + Ineq + Prob, data = crime_data)
##
## 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) -5040.50     899.84  -5.602 1.72e-06 ***
## M             105.02      33.30   3.154  0.00305 **
## Ed            196.47      44.75   4.390 8.07e-05 ***
## Po1           115.02      13.75   8.363 2.56e-10 ***
## U2             89.37      40.91   2.185  0.03483 *
## Ineq           67.65      13.94   4.855 1.88e-05 ***
## Prob        -3801.84    1528.10  -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
```

```
#plot of the results
plot(model$residuals)
```




```
qqPlot(model$residuals, main = "Normal Q-Q plot", xlab = "Residuals", ylab = "quantiles")
```




```
## [1] 11 19
```

Using the leap function with forward selection, we can see the best model has 6 variables as this is the model that minimizes the MAE on the testing datasets. The R2 from this model is 76% and all the variables in this model are statistically significant. We can see that the resdiuals seem to be in line with the assumptions of constant variance, independence and normal distribution of the results.

The model with 6 variables has a RMSE of 220.2582 and a MAE of 178.4304 in the testing database.

We tried the same model using Backward selection for variable selection but it resulted in the same variables being picked up so we are not repeating it here.

2.2. MASS package

We can use the stepwise regression on the MASS package in combination with Caret package as well. With this stepwise function we have an objective function which in this case is AIC. Since the function has backward stepwise as default, the algorithm tries multiple variables and finds the one that does not increase AIC when removed. It removes variables until there is no variable to remove that doesn't consequently increase AIC.

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

# Train the model
step.model_2 <- train(Crime ~., data = crime_data,
                      method = 'lmStepAIC',
```

```
                    trControl = train.control
                    )


#results from the mddel for different number of variables
step.model_2$results

#summary
summary(step.model_2$finalModel)
```

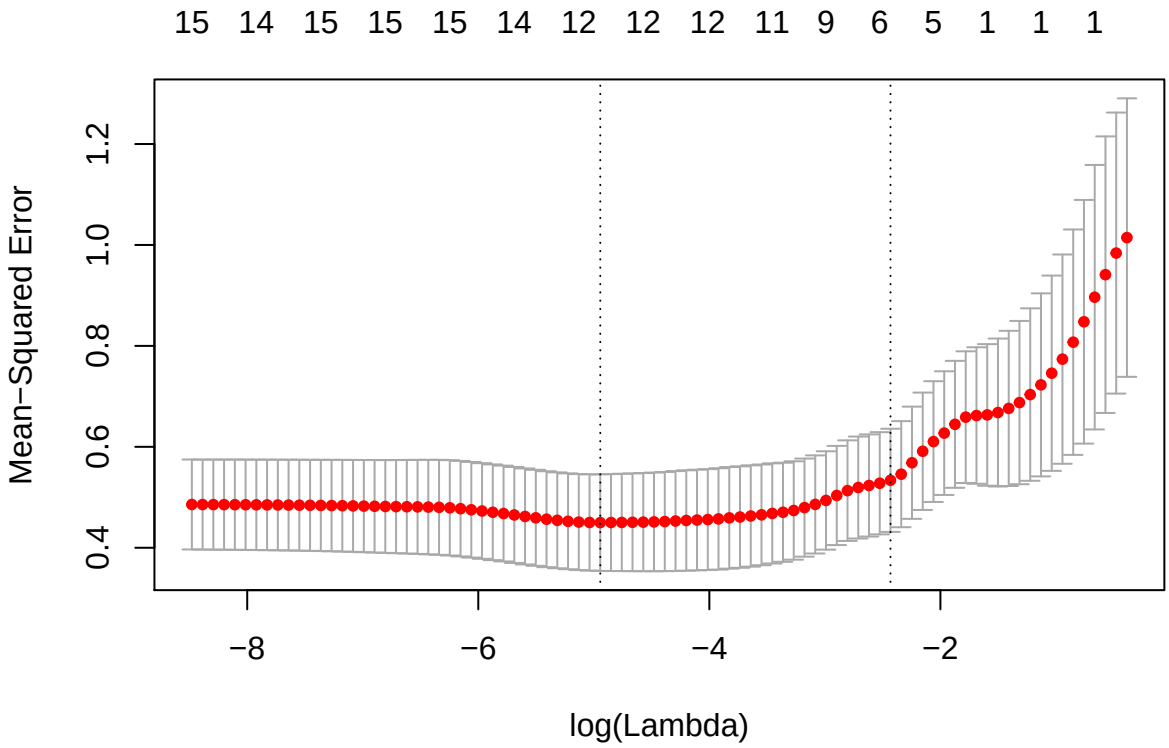
Using the MASS packkage to perform a stepwise function selection by AIC, we can see the best model has 8 variables as this model minimizes the AIC.The selection is done by testing different numbers of paramenters and the maximum likelyhood value that results in the lowest set of paramenters with best statistical significance for the model. The R2 from this model is 78%.

The model has a RMSE of 268.0678 and a MAE of 214.3536.

3. Lasso Regression

Lasso Regression performs a type L1 regularization, adding a penalty equal to the absolute value of magnitude of coefficients. Using Lasso regression, some coefficients might become zero and get eliminated from the model. The lasso regression adds a tunning parameter that controls the strength of the L1 penalty. The higher the penalty, more coefficients are eliminated from the model.

```
#converting data to dataframe and scaling
crime_data <- read.table("uscrime.txt", header = TRUE)

crime_data_matrix <- as.matrix(scale(crime_data))

predictors <- crime_data_matrix[,1:15]
response <-  crime_data_matrix[,16]


#Using cross validation for the Lasso regression
model_lasso <- cv.glmnet(predictors, response, alpha = 1)

#Finding optimal value of lambda that minimizes cross-validation errors
plot(model_lasso)
```



The plot displays the cross-validation error according to the log of r λ The left dashed vertical line indicates that the log of the optimal value of r λ is approximately -3, which is the one that minimizes the prediction error. This r λ value will give the most accurate model. The exact value of r λ can be viewed as follow:s

```
model_lasso$lambda.min
```

```
## [1] 0.007126408
```

The function cv.glmnet() finds also the value of r λ that gives the simplest model but also lies within one standard error of the optimal value of r λ This value is called lambda.lse.

```
model_lasso$lambda.1se
```

```
## [1] 0.08785769
```

And the coefficients can be retrieved by doing:

```
coef(model_lasso, model_lasso$lambda.1se)
```

```
## 16 x 1 sparse Matrix of class "dgCMatrix"
##                            1
## (Intercept) -3.504148e-16
## M            1.076123e-01
## So           .
```

```
## Ed           .
## Po1          7.105297e-01
## Po2          .
## LF           .
## M.F          1.297381e-01
## Pop          .
## NW           .
## U1           .
## U2           .
## Wealth       .
## Ineq         1.903715e-01
## Prob        -1.232049e-01
## Time         .
```

We can see that by using Lasso, with the optimal lambda value, our final model has 5 variables, less than the resultant from stepwise regression.

Computing the Lasso regression model with the optimal parameter for λ:

```
# Split data into train (70%) and test (30%)
crime_data_matrix_2 <- as.matrix((crime_data))

ind <- sample(1:nrow(crime_data_matrix),floor(nrow(crime_data_matrix)*0.70))
crime_data_matrix_train <- crime_data_matrix[ind,]
crime_data_matrix_test <- crime_data_matrix[-ind,]

yhat <- predict(model_lasso, s=model_lasso$lambda.min, newx=crime_data_matrix_test[,1:15])
MSE <- mean((crime_data_matrix_test[,16] - yhat)^2)
MSE
```

```
## [1] 0.2145095
```

4. Elastic Net

First, we have to find the best value for alpha. So, we loop through different values, and in each iteration we store the minimum values of MSE and lambda.min

```
mse_array <- numeric()
find_alpha <- function(num, scaled_data){
    alpha <- num
    elastic_net <- cv.glmnet(predictors,
                         response,
                         alpha = alpha,
                         nfolds=5,
                         type.measure="mse",
                         family="gaussian",
                         standardize=FALSE)

    mse_array <<- cbind(mse_array, c(alpha, min(elastic_net$cvm),elastic_net$lambda.min))
}

# Loop over different values of alpha
```

```r
for (i in seq(.01,1,by = .01)){find_alpha(i,crime_data_matrix)}

minIdx <- which.min(mse_array[2,])
mse_array[2,minIdx]
```

```
## [1] 0.3320627
```

```r
mse_array[1, minIdx]
```

```
## [1] 0.89
```

Since we have found the best value of alpha to be 0.8, we will now run the elastic net.

```r
elastic_result <- cv.glmnet(predictors,
                        response,
                        alpha = 0.8,
                        nfolds=5,
                        type.measure="mse",
                        family="gaussian",
                        standardize=FALSE)

coef(elastic_result, s = elastic_result$lambda.min)
```

```
## 16 x 1 sparse Matrix of class "dgCMatrix"
##                            1
## (Intercept) -3.010517e-16
## M            2.171088e-01
## So           6.221975e-02
## Ed           3.265191e-01
## Po1          7.792288e-01
## Po2          .
## LF           4.233760e-03
## M.F          1.431730e-01
## Pop          .
## NW           2.286064e-02
## U1          -7.686799e-02
## U2           1.641064e-01
## Wealth       4.316240e-05
## Ineq         4.522345e-01
## Prob        -2.171403e-01
## Time         .
```

```r
#So elastic_net selects 14 variables. Let's rerun lm using those features.

elastic.lm <- lm(Crime ~ ., data = as.data.frame(crime_data_matrix[, -4]))

summary(elastic.lm)
```

```
##
## Call:
## lm(formula = Crime ~ ., data = as.data.frame(crime_data_matrix[,
```

```
##     -4]))
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -1.22520 -0.34438  0.00482  0.35053  1.10215
##
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept) -5.240e-16  8.163e-02   0.000  1.00000
## M            2.932e-01  1.403e-01   2.090  0.04462 *
## So           4.420e-03  1.907e-01   0.023  0.98165
## Ed           4.918e-01  1.835e-01   2.680  0.01152 *
## Po2          7.123e-01  1.986e-01   3.586  0.00110 **
## LF          -9.507e-03  1.553e-01  -0.061  0.95157
## M.F          1.645e-01  1.595e-01   1.031  0.31005
## Pop         -5.973e-02  1.312e-01  -0.455  0.65210
## NW           5.328e-02  1.753e-01   0.304  0.76307
## U1          -2.669e-01  2.032e-01  -1.314  0.19828
## U2           3.937e-01  1.855e-01   2.122  0.04164 *
## Wealth       2.573e-01  2.676e-01   0.962  0.34341
## Ineq         7.619e-01  2.419e-01   3.150  0.00353 **
## Prob        -2.303e-01  1.347e-01  -1.710  0.09701 .
## Time         8.241e-03  1.296e-01   0.064  0.94969
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.5597 on 32 degrees of freedom
## Multiple R-squared:  0.7821, Adjusted R-squared:  0.6868
## F-statistic: 8.205 on 14 and 32 DF,  p-value: 4.875e-07
```

We are able to notice that the most siginificant variables are M, Ed, Po1, U2, Ineq, Prob.

5. Conclusion

We saw that all models agreed on the final 6 variables - M, Ed, Po1, U2, Ineq, Prob. Also, we were able to build a simpler model with just 5 features which had decent performance considering the number of features. It is interesting to notice that all the other models with more than 5 features did not have a much greater performance. In conclusion, we would choose the simpler models whenever performance was not affected because it provides the same result but easier interpretatio of results, and each feature's impact on prediction.