Home Work 8

ISYE 6501

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.

Answer:

```
library(kernlab)
library(kknn)
library(lattice)
library(ggplot2)

library(caret)# an aggregator package for performing many machine learning mo
dels

library(corrplot) #graphical display of correlation matrix

library(rsample)  # data splitting

library (MASS) #Stepwise and model selection using AIC
library(glmnet)

library(leaps)
```

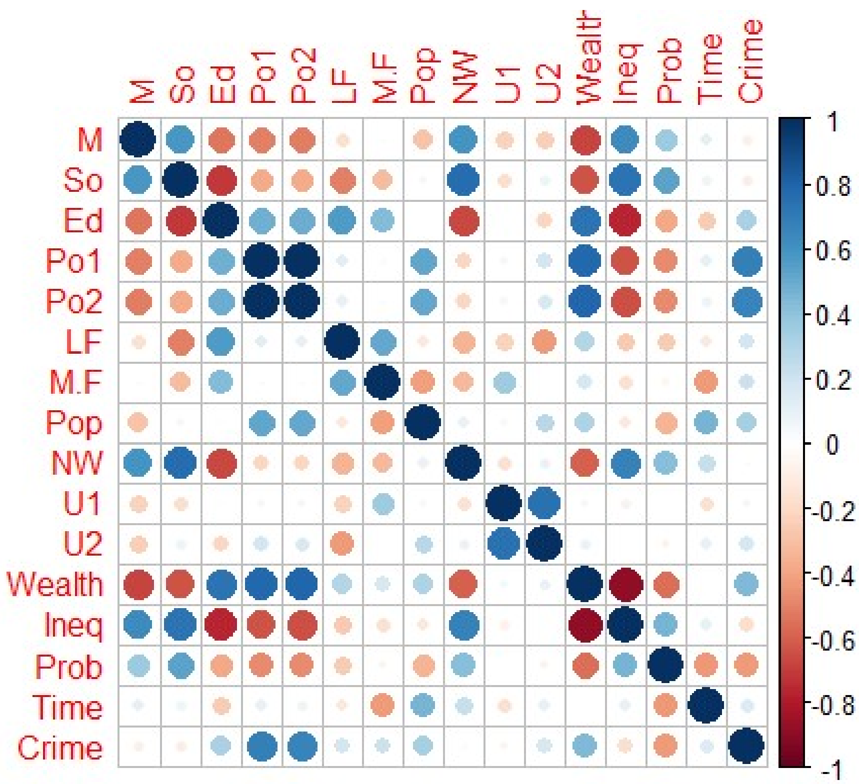
Next we will load the data and look at the data structure.

```
rm(list=ls())
uscrime <- read.table("uscrime.txt",stringsAsFactors = FALSE, header = TRUE)
head(uscrime)

##      M So   Ed  Po1  Po2    LF   M.F Pop   NW    U1  U2 Wealth Ineq     Pr
ob
## 1 15.1  1  9.1  5.8  5.6 0.510  95.0  33 30.1 0.108 4.1   3940 26.1 0.0846
02
## 2 14.3  0 11.3 10.3  9.5 0.583 101.2  13 10.2 0.096 3.6   5570 19.4 0.0295
99
## 3 14.2  1  8.9  4.5  4.4 0.533  96.9  18 21.9 0.094 3.3   3180 25.0 0.0834
01
## 4 13.6  0 12.1 14.9 14.1 0.577  99.4 157  8.0 0.102 3.9   6730 16.7 0.0158
01
## 5 14.1  0 12.1 10.9 10.1 0.591  98.5  18  3.0 0.091 2.0   5780 17.4 0.0413
99
```

```
## 6 12.1  0 11.0 11.8 11.5 0.547  96.4  25  4.4 0.084 2.9   6890 12.6 0.0342
01
##       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
```

Lets examine the data for correlations using a visualisation. This will help us understand which features are most useful to us.

```
#uscrime$So <- NULL
#head(uscrime)
cormatrix <- cor(uscrime) #calculate correlation matrix
corrplot(cormatrix, method = "circle") #plot correlation matrix
```



The correlation plot may not be needed here but it gives us an idea for which features to look out for tree branching. For instance Po1/Po2(but not both) are very important for Crime.

Scale the data. Except So as it is binary.

```
colNames <- colnames(uscrime[,-2])[1:14]

normalize <- function(df, cols) {
  result <- df # make a copy of the input data frame
```

```
  for (j in cols) { # each specified col
    m <- mean(df[,j]) # column mean
    std <- sd(df[,j]) # column (sample) sd

    result[,j] <- sapply(result[,j], function(x) (x - m) / std)
  }
  return(result)
}

#normalize predictors except 'So'
datanorm <- normalize(uscrime, colNames)

head(datanorm)

##            M So         Ed        Po1        Po2         LF         M.F
## 1  0.9886930  1 -1.3085099 -0.9085105 -0.8666988 -1.2667456 -1.12060499
## 2  0.3521372  0  0.6580587  0.6056737  0.5280852  0.5396568  0.98341752
## 3  0.2725678  1 -1.4872888 -1.3459415 -1.2958632 -0.6976051 -0.47582390
## 4 -0.2048491  0  1.3731746  2.1535064  2.1732150  0.3911854  0.37257228
## 5  0.1929983  0  1.3731746  0.8075649  0.7426673  0.7376187  0.06714965
## 6 -1.3983912  0  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
```

Stepwise regression assumes that the predictor variables are not highly correlated. As shown above there is no major correlation except for between Po1 and Po2. During each step in stepwise regression, a variable is considered for addition to or subtraction from the set of predictor variables based on some pre-specified criterion (e.g. adjusted R-squared). The two main approaches involve forward selection, starting with no variables in the model, and backwards selection, starting with all candidate predictors.

```
#Basic Stepwise Regression

# Fit the full model
full.model <- lm(Crime ~., data = uscrime)
# Stepwise regression model
# stepAIC(), which choose the best model by AIC.
```

```r
step.model <- stepAIC(full.model, direction = "both",trace = FALSE)
summary(step.model)
```

```
## 
## Call:
## lm(formula = Crime ~ M + Ed + Po1 + M.F + U1 + U2 + Ineq + Prob, 
##     data = uscrime)
## 
## 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) -6426.10    1194.61  -5.379 4.04e-06 ***
## M              93.32      33.50   2.786  0.00828 ** 
## Ed            180.12      52.75   3.414  0.00153 ** 
## Po1           102.65      15.52   6.613 8.26e-08 ***
## M.F            22.34      13.60   1.642  0.10874    
## U1          -6086.63    3339.27  -1.823  0.07622 .  
## U2            187.35      72.48   2.585  0.01371 *  
## Ineq           61.33      13.96   4.394 8.63e-05 ***
## Prob        -3796.03    1490.65  -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
```

The above result is an out of the box stepwise regression selecting the best model using AIC. Develop a new model with the eight variables found with stepwise regression.

```r
mod_Step8 = lm(Crime ~ M.F+U1+Prob+U2+M+Ed+Ineq+Po1, data = datanorm)
```

Lets try our cross validation with stepwise regression to see if we can get a better model.

The function starts by searching different best models of different size, up to the best 10-variables model. The number of features to be added is specified by nvmax. We specify stepwise selection by “leapSeq”.

```r
# Set seed for reproducibility
set.seed(123)
# Set up repeated k-fold cross-validation
train.control <- trainControl(method = "cv", number = 10)
# Train the model
step.model <- train(Crime ~., data = datanorm,
                    method = "leapSeq",
                    tuneGrid = data.frame(nvmax = 4:10),
                    trControl = train.control)
step.model$results
```

```
##   nvmax     RMSE  Rsquared      MAE    RMSESD RsquaredSD    MAESD
## 1     4 271.9430 0.6397970 222.7241 104.89255  0.2758066 74.98079
## 2     5 260.3804 0.5930318 209.8459  91.58884  0.2992502 69.13846
## 3     6 232.9240 0.7379318 188.7075  97.13428  0.2237860 70.61600
## 4     7 237.8892 0.6602768 197.7499  90.42968  0.2886561 67.34118
## 5     8 276.3432 0.5354045 226.0451 107.67461  0.3665290 78.50840
## 6     9 246.8846 0.6975119 201.3635  92.02137  0.2958348 67.53324
## 7    10 257.4864 0.6694090 212.3040  96.25332  0.3222326 70.34452

#Summary of best model from cross validation
#summary(step.model$finalModel)
```

From above, it can be seen that the model with 6 variables (nvmax = 6) is the one that has the lowest RMSE and high R squared. The regression coefficients of the final model (id = 4) can be accessed as follow.

```
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
```

Develop a new model with the six variables found with cross validation with stepwise regression.

```
mod_Step6 = lm(Crime ~ M+Prob+U2+Ed+Ineq+Po1, data = datanorm)
summary(mod_Step6)

##
## Call:
## lm(formula = Crime ~ M + Prob + U2 + Ed + Ineq + Po1, data = datanorm)
##
## 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 **
## Prob          -86.44      34.74  -2.488  0.01711 *
## U2             75.47      34.55   2.185  0.03483 *
## Ed            219.79      50.07   4.390 8.07e-05 ***
## Ineq          269.91      55.60   4.855 1.88e-05 ***
## Po1           341.84      40.87   8.363 2.56e-10 ***
## ---
## 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
```

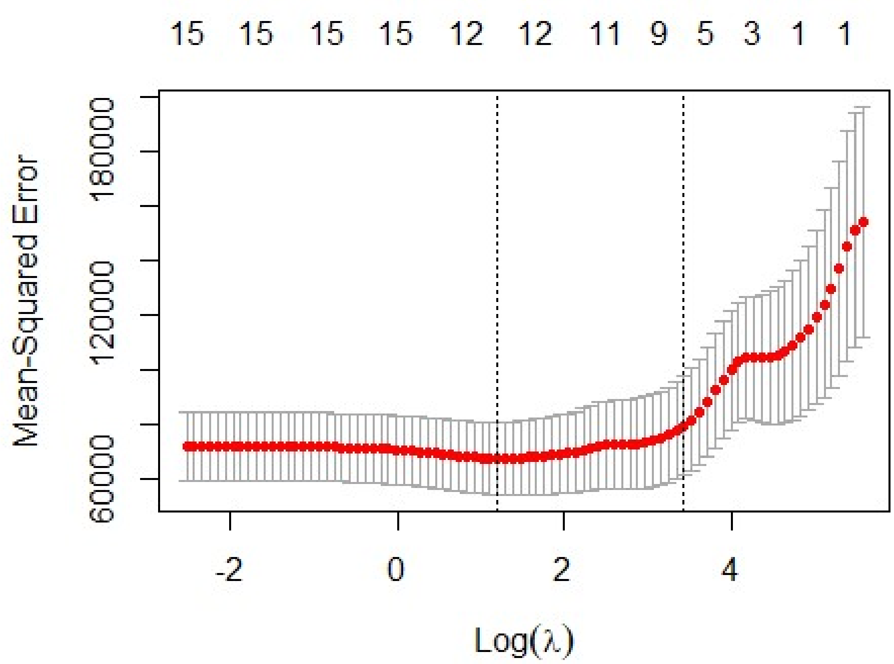
Since the previous AIC model with 8 features has a higer adjusted R squared of 0.7444, we will select that as our final model.

```
coef(mod_Step8)

## (Intercept)         M.F          U1        Prob          U2           M
##   905.08511    65.82935  -109.73459   -86.31027   158.22138   117.28311
##          Ed        Ineq         Po1
##   201.50034   244.70226   305.07465
```

LASSO Regression

Prepare the data for use in lasso regression.

```
# Dumy code categorical predictor variables
x=data.matrix(datanorm[,-16])
y=data.matrix(datanorm$Crime)

library(glmnet)
set.seed(123)
cv.lasso <- cv.glmnet(x, y, alpha = 1)
plot(cv.lasso)
```



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

```
cv.lasso$lambda.min

## [1] 3.319887
```

Using lambda.min as the best lambda, gives the following regression coefficients:

```
coef(cv.lasso, cv.lasso$lambda.min)

## 16 x 1 sparse Matrix of class "dgCMatrix"
##                       1
## (Intercept) 894.31532
## M           104.46095
## So           31.63626
## 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          .
```

Compute the final model using lambda.min:

```
mod_lassomin = lm(Crime ~ M+So+Pop+NW+U1+U2+Wealth+Prob+M.F+Ed+Ineq+Po1, data
= datanorm)
summary(mod_lassomin)

##
## Call:
## lm(formula = Crime ~ M + So + Pop + NW + U1 + U2 + Wealth + Prob +
##     M.F + Ed + Ineq + Po1, data = datanorm)
##
## 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    
## 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    
## Prob          -92.75      41.12  -2.255  0.03065 *  
## M.F            48.92      48.12   1.017  0.31656    
## Ed            195.70      62.94   3.109  0.00378 ** 
## Ineq          285.77      85.19   3.355  0.00196 ** 
## Po1           293.18      64.99   4.511 7.32e-05 ***
## ---
## 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
```

The function cv.glmnet() finds also the value of lambda that gives the simplest model but also lies within one standard error of the optimal value of lambda. This value is called lambda.1se.

```
cv.lasso$lambda.1se

## [1] 30.96138
```

Using lambda.1se as the best lambda, gives the following regression coefficients:

```
coef(cv.lasso, cv.lasso$lambda.1se)

## 16 x 1 sparse Matrix of class "dgCMatrix"
##                       1
## (Intercept) 905.08510638
## M            44.78226128
## So            .         
## Ed            0.07286797
## Po1         282.78158591
## Po2           .         
## LF            .         
## M.F          54.26395269
## Pop           .         
## NW            .         
## U1            .         
## U2            .         
## Wealth        .         
## Ineq         82.41002075
## Prob        -51.91991834
## Time          .         
```

Compute the final model using features from lambda.1se:

```r
mod_lassolse = lm(Crime ~ M+Prob+M.F+Ed+Ineq+Po1, data = datanorm)
summary(mod_lassolse)
```

```
##
## Call:
## lm(formula = Crime ~ M + Prob + M.F + Ed + Ineq + Po1, data = datanorm)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -484.68 -106.59   -2.44  135.89  505.69
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)   905.09      30.24  29.934  < 2e-16 ***
## M              91.75      40.96   2.240   0.0307 *
## Prob          -89.69      35.90  -2.498   0.0167 *
## M.F            50.71      36.16   1.402   0.1685
## Ed            140.01      55.54   2.521   0.0158 *
## Ineq          259.59      58.15   4.464 6.41e-05 ***
## Po1           364.24      41.41   8.797 6.80e-11 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 207.3 on 40 degrees of freedom
## Multiple R-squared:  0.7502, Adjusted R-squared:  0.7127
## F-statistic: 20.02 on 6 and 40 DF,  p-value: 1.2e-10
```

Since the lambda.1se model has far fewer features, 6 compared to lambda.min having 12 features, and almost identical adj R squared, it is preferable to select the lse model as the final LASSO model.

Elastic Net

We'll test the combination of 10 different values for alpha and lambda. This is specified using the option tuneLength.

The best alpha and lambda values are those values that minimize the cross-validation error.

```r
# Build the model using the training set
set.seed(123)
model_net <- train(Crime ~., data = datanorm, method = "glmnet",
                trControl = trainControl("cv", number = 10),
                tuneLength = 10)
```

```
## Warning in nominalTrainWorkflow(x = x, y = y, wts = weights, info = trainI
nfo, :
## There were missing values in resampled performance measures.
```

```r
# Best tuning parameter
model_net$bestTune
```

```
##    alpha   lambda
## 95     1 3.461984
```

```r
# Coefficient of the final model. You need
# to specify the best lambda
coef(model_net$finalModel, model_net$bestTune$lambda)
```

```
## 16 x 1 sparse Matrix of class "dgCMatrix"
##                       1
## (Intercept) 894.19825
## M           104.10991
## So           31.98013
## Ed          174.22823
## Po1         296.45399
## Po2            .
## LF             .
## M.F          52.48181
## Pop         -18.33992
## NW           14.11213
## U1          -70.27007
## U2          115.18367
## Wealth       52.51298
## Ineq        248.69439
## Prob        -89.03940
## Time           .
```

The Elastic Net selects 12 variables compared to 6 in Lasso and 8 in Step Wise. Next we compare how this new model performs compared to the Lasso and Step Wise models

```r
mod_Elastic_net = lm(Crime ~So+M+Ed+Po1+M.F+Pop+NW+U1+U2+Wealth+Ineq+Prob, da
ta = datanorm)
summary(mod_Elastic_net)
```

```
## 
## Call:
## lm(formula = Crime ~ So + M + Ed + Po1 + M.F + Pop + NW + U1 + 
##     U2 + Wealth + Ineq + Prob, data = datanorm)
## 
## 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
```

The R-SQuared value is similar using LASSO model with only 6 variables. Therefore the Elastic net may not be doing a good job as it selects 6 more variables for a similar RSquared value