Bryson Cook
ISYE6501, Spring 2018
HW8

Question 11.1

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

I created the model using the step() function, which is part of the R stats package. This function chooses a model by AIC in a Stepwise Algorithm and can be set to forward regression, backward regression, or both (stepwise) regression. Using the functions stepwise regression, I got the following model and coefficients:

```
Call:
lm(formula = Crime ~ Po1 + Ineq + Ed + M + Prob + U2, data = mydata)

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 ***
Po1           115.02      13.75   8.363 2.56e-10 ***
Ineq           67.65      13.94   4.855 1.88e-05 ***
Ed            196.47      44.75   4.390 8.07e-05 ***
M             105.02      33.30   3.154  0.00305 ** 
Prob        -3801.84    1528.10  -2.488  0.01711 *  
U2             89.37      40.91   2.185  0.03483 *  
---
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
(Intercept)         Po1        Ineq           Ed            M        Prob 
U2 
-5040.50498   115.02419    67.65322   196.47120   105.01957 -3801.83628    89
.36604 
```

To test the function, I then manually stepped through the process using the “forward” regression method while checking after each step for any factors with a p-value > 0.05 which would signal the factor needs to be removed. After 7 total steps, the function stated that the AIC would not decrease by adding any of the remaining factors. No factors were removed and the model ended up equivalent to the original stepwise regression model.

2. Lasso

I created my Lasso model using the cv.glmnet function in the glmnet package. I first scaled the data, as this is very important in global approaches since otherwise a factor that is larger than the rest can artificially dominate the coefficient sizes. I then used all of the data in the cv.glmnet function to create the lasso. The lambda min and 1se values were 17.71724 and 40.92912, respectively and their corresponding factors are shown below, with the unused factors removed.

```
> coef(lasso, s = lasso$lambda.min)
(Intercept) -483.35043
M            469.88508
So            34.24302
Ed           526.94068
Po1         1210.52937
M.F          248.14372
Pop          -62.41065
NW            52.20859
U1          -259.51651
U2           487.08970
Wealth       185.40526
Ineq         898.79317
Prob        -436.95775
> coef(lasso, s = lasso$lambda.1se)
(Intercept)  345.6484
M            192.0794
Po1         1118.8669
M.F          233.2874
Ineq         276.8265
Prob        -236.6188
```

I was curious which model would prove to be better, so I split the data into training (80% of the data) and test (20% of the data) sets and created two separate models using the lm() function, created from the above factors for each model, on the training data. I then used the predict() function to apply the models to the test data and calculated the R2 of each. The .min model R2 being 0.7595 and the .1se R2 being 0.7987. The .1se model is slightly better, but both should be considered good quality.

3. Elastic net

Similar to the lasso model, I created my elastic net model using the cv.glmnet function in the glmnet package. I first scaled the data, as this is very important in global approaches since otherwise a factor that is larger than the rest can artificially dominate the coefficient sizes. I then used all of the data as inputs to the cv.glmnet function. I then set up a for loop to try alphas from 0.05 to 0.95 by 0.05 increments, recording the deviance explained by the lambda.min and lambda.1se models at each step. The results of the sweep are shown below:

	alpha	lambda.min	alpha	lambda.1se
[1,]	0.05	0.7600097	0.05	0.6112904
[2,]	0.10	0.7535046	0.10	0.6510103
[3,]	0.15	0.7719830	0.15	0.6424259
[4,]	0.20	0.7446501	0.20	0.5150635
[5,]	0.25	0.7521779	0.25	0.5962289
[6,]	0.30	0.7849256	0.30	0.6079146
[7,]	0.35	0.7728548	0.35	0.5976618
[8,]	0.40	0.7723386	0.40	0.4698605
[9,]	0.45	0.7675174	0.45	0.6930689
[10,]	0.50	0.7262310	0.50	0.4532806
[11,]	0.55	0.7837526	0.55	0.7010773
[12,]	0.60	0.7870555	0.60	0.7319783
[13,]	0.65	0.6951777	0.65	0.4001932
[14,]	0.70	0.7205195	0.70	0.5835926
[15,]	0.75	0.7770456	0.75	0.7010699
[16,]	0.80	0.7039714	0.80	0.5408690
[17,]	0.85	0.7665348	0.85	0.6603967
[18,]	0.90	0.7432739	0.90	0.6166792
[19,]	0.95	0.7866508	0.95	0.7208391

Coincedentally, the maximum deviance explained is at alpha = 0.60 for both models. I then re-ran the cv.glmnet() function with alpha = 0.60 and extracted the coefficients for the .min and.1se models, which are shown below.

```
> coef(enet1se, s = enet1se$lambda.1se)
(Intercept) -293.041269
M            414.868525
So            46.301462
Ed           439.007442
Po1         1088.142151
Po2           87.404209
LF             0.177451
M.F          284.530645
NW            60.880746
U1          -195.708421
U2           379.046332
Wealth        93.871025
Ineq         726.763583
Prob        -427.527884
> coef(enet1se, s = enet1se$lambda.1se)
(Intercept)  335.94369
M            188.16200
Ed            22.37253
Po1          758.72949
Po2          332.62180
M.F          257.77357
NW            57.20927
Ineq         278.27325
Prob        -284.9983
```

I was again curious which model would prove to be better, so I split the data into training (80% of the data) and test (20% of the data) sets and created two separate models using the lm() function, created from the above factors for each model, on the training data. I then used the predict() function to apply the models to the test data and calculated the R2 of each. The .min model R2 being 0.4987 and the .1se R2 being 0.4419. The .min model is slightly better, but it's interesting that the lasso models had a much higher R2 for both cases.