HW7 - Q11.1

Yu Fung David Wang

Read in Data

```r
# Goal is to Predict " Crime"
crime_data <- read.table("uscrime.txt", header = TRUE)
head(crime_data)
```

```
##      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
```

Split Data into Train and Test

```r
set.seed("1234")
sample <- sample(c(TRUE, FALSE), nrow(crime_data), replace = TRUE, prob = c(0.7, 0.3))
train <- crime_data[sample,]
test <- crime_data[!sample,]
```

11.1 (1) Stepwise Regression

```r
# Will Build a Both Direction Stepwise Regression Model

# define intercept-only model
intercept <- lm(Crime ~ 1, data = train)
# define model with predictors
step_model <- lm(Crime~., data = train)
# perform forward stepwise reg
step_forward <- step(intercept, direction = "both", scope = formula(step_model), trace = 0) # trace = 0

step_forward$coefficients
```

```
## (Intercept)         Po1        Ineq          Ed        Prob           M
## -4581.88282   127.56461    74.23826   178.13660 -3595.04811    75.59325
##          U2
##    62.20142
```

```
step_forward$anova
```

```
##     Step Df   Deviance Resid. Df Resid. Dev      AIC
## 1         NA         NA        37    5085064 450.5608
## 2  + Po1 -1 2251139.28        36    2833925 430.3443
## 3 + Ineq -1  763711.24        35    2070214 420.4119
## 4   + Ed -1  443356.80        34    1626857 413.2538
## 5 + Prob -1  162409.77        33    1464447 411.2573
## 6    + M -1  110203.35        32    1354244 410.2844
## 7   + U2 -1   82972.09        31    1271272 409.8818
```

```
# Predict
step_predictions <- predict(step_forward, test)

# RMSE
RMSE_step <- sqrt(mean(as.matrix((test$Crime - step_predictions)^2)))
paste("RMSE of Stepwise Regression Model:", RMSE_step)
```

```
## [1] "RMSE of Stepwise Regression Model: 215.016974452619"
```

Based on the given coefficients of the model, the model will look like - Crime ~ 115.02xPo1 + 67.65xIneq + 196.47xEd + (-3801.84)xProb +105.02x M + 89.37xU2

Based on the ANOVA test, we can see that the akaike info criterion (AIC) is lowest with predictors (Po1, Ineq, Ed, M, Prob, U2). As a lower AIC indicates a better fit model, it shows that all predictors used will produce the most reduction in AIC. All other features were deemed not important.

The trained model gives a RMSE value of 215.017 on the test data.

Set Predictor, Response Values (Train) and Test Predictor Values for Lasso and Elastic

```
library(glmnet) # in glmnet, there is a standardize function to scale the data
```

```
## Loading required package: Matrix
## Loaded glmnet 4.1-8
```

```
response <- train$Crime

# predictors have to be in R's matrix format rather than data frame format
predictors <- data.matrix(train[,c('M', 'So', 'Ed', 'Po1', 'Po2', 'LF', 'M.F', 'Pop', 'NW', 'U1', 'U2', 

test_predictors <- data.matrix(test[,c('M', 'So', 'Ed', 'Po1', 'Po2', 'LF', 'M.F', 'Pop', 'NW', 'U1', '
```

11.1 (2) Lasso

```
# First, perform k-fold CV to identify lambda for lowest RMSE
cv_lasso <- cv.glmnet(predictors, response , alpha = 1) # alpha = 1 is the lasso penalty (alpha = 0 is 
opt_lambda <- cv_lasso$lambda.min
paste("Optimal lambda is:", opt_lambda)
```

```
## [1] "Optimal lambda is: 11.2973361554184"
```

```
# Make sure to scale (standardize) in glmnet
lasso_model <- glmnet(predictors, response, lambda = opt_lambda, standardize = TRUE, family="gaussian")
coef(lasso_model)
```

```
## 16 x 1 sparse Matrix of class "dgCMatrix"
##                        s0
## (Intercept) -3713.9153520
```

```
## M              38.9350005
## So             55.1676972
## Ed            115.5901674
## Po1           102.9905871
## Po2             4.8646787
## LF              .
## M.F             9.5861386
## Pop             0.9478213
## NW              0.9619140
## U1              .
## U2             23.3844976
## Wealth          .
## Ineq           51.7147433
## Prob        -3105.5178265
## Time            .
```

```
# Predict
lasso_prediction <- predict(lasso_model, s = opt_lambda, newx = test_predictors)

# RMSE
RMSE_lasso <- sqrt(mean(as.matrix((test$Crime - lasso_prediction)^2)))
paste("RMSE of Lasso Regression Model:", RMSE_lasso)
```

```
## [1] "RMSE of Lasso Regression Model: 246.602004174076"
```

Based on the given coefficients of the model, the model will look like - Crime ~ -3582.998 + 37.12xM + 51.36xSo + 109.44xEd + 102.19xPo1 + 4.99xPo2 + 9.56xM.F + 0.95xPop + 0.96xNW + 20.89xU2 + 50.33xIneq + (-3035.22)xProb

Based on the coefficients, LF, U1, and Wealth were not utilized/not important features.

11.1 (3) Elastic Net

```
# Have to combine ridge and lasso regression

# Have to tune hyperparameter alpha (run cv.glmnet function over wide range of alphas)

elastic_models <- list()

for (i in 0:29) { # looking through 30 different alphas from 0 to 1
  curr_alpha <- paste("alpha", round(i/29, 2))
  elastic_models[[curr_alpha]] <- cv.glmnet(predictors, response, alpha = i/29, standardize = TRUE, fami
}

# Look for best alpha based on evaluation of rmse on test data

elastic_table <- data.frame()
for (i in 0:29) {
  curr_alpha <- paste("alpha", round(i/29, 2))
  curr_elastic_model <- elastic_models[[curr_alpha]]

  # predictions
  elastic_prediction <- predict(curr_elastic_model, curr_elastic_model$lambda.1se, newx = test_predicto

  # RMSE
RMSE_elastic <- sqrt(mean(as.matrix((test$Crime - elastic_prediction)^2)))
```

```r
# Store into elastic_table results
  elastic_table <- rbind(elastic_table, data.frame(Alpha = round(i/29, 2), 2, RMSE = RMSE_elastic))

}

elastic_table
```

```
##    Alpha X2     RMSE
## 1   0.00  2 332.8455
## 2   0.03  2 324.5594
## 3   0.07  2 358.8645
## 4   0.10  2 391.1069
## 5   0.14  2 323.2978
## 6   0.17  2 371.5414
## 7   0.21  2 324.1561
## 8   0.24  2 320.1245
## 9   0.28  2 327.9942
## 10  0.31  2 331.2957
## 11  0.34  2 363.3935
## 12  0.38  2 319.4727
## 13  0.41  2 317.1067
## 14  0.45  2 309.4465
## 15  0.48  2 319.2187
## 16  0.52  2 297.1754
## 17  0.55  2 322.7308
## 18  0.59  2 321.0543
## 19  0.62  2 359.3735
## 20  0.66  2 334.7673
## 21  0.69  2 333.3433
## 22  0.72  2 361.9061
## 23  0.76  2 321.7231
## 24  0.79  2 358.9947
## 25  0.83  2 357.8118
## 26  0.86  2 300.8366
## 27  0.90  2 348.5540
## 28  0.93  2 284.3908
## 29  0.97  2 314.9414
## 30  1.00  2 305.3935
```

```r
min_RMSE <- min(elastic_table$RMSE)
min_alpha <- elastic_table$Alpha[elastic_table$RMSE == min_RMSE]

paste("The min RMSE of ", min_RMSE, "is found with a alpha value of", min_alpha)
```

```
## [1] "The min RMSE of  284.390835546979 is found with a alpha value of 0.93"
```

```r
# Use the alpha that provides min RMSE to look at coefficients
table_alpha_name <- paste("alpha", min_alpha)
predict(elastic_models[[table_alpha_name]], type ="coef")
```

```
## 16 x 1 sparse Matrix of class "dgCMatrix"
##                lambda.1se
## (Intercept) -1960.6412102
## M              15.8420663
## So              0.2230406
```

```
## Ed            35.8809645
## Po1           67.9939224
## Po2           29.4904825
## LF             .
## M.F            8.7428824
## Pop            1.0014388
## NW             1.0911256
## U1             .
## U2             .
## Wealth         .
## Ineq          32.5777796
## Prob       -2114.5374081
## Time           .
```

The coefficients are seen above, as the results/regression formula constantly change, will not write it into equation form. However, generally, feautres LF, U1, U2, Wealth, and Time are not utilized for the regression based on Elastic Net Regression.