Homework 5

Hw-5

Q1

First we will load the data and build the basic models, one with every feature and one with none.

```
# Stepwise Regression
library(MASS)
library(caret)
```

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

```
crime <- read.table('../Hw-3/uscrime.txt', header = TRUE)
# sapply(crime, class)
crime$So <- as.factor(crime$So)
crime$Crime <- as.numeric(crime$Crime)

preprocessParams <- preProcess(crime[,1:15], method=c("scale","center"))
crime.lm <- crime
crime.lm[,1:15] <- predict(preprocessParams,crime.lm[,1:15])

## Stepwise Regression Model
crime.full <- lm(Crime~., data = crime.lm)
crime.null <- lm(Crime~1, data = crime.lm)
```

We will make forward selection, backwards selection, then bi-direction

```
# Forward Selection
step.forw <- stepAIC(crime.null, direction = "forward",trace = FALSE,
                     scope = ~ M + So + Ed + Po1 + Po2 + LF + M.F + Pop + NW + U1 + U2 + Wealth + Ineq +
summary(step.forw)
```

```
##
## Call:
## lm(formula = Crime ~ Po1 + Ineq + Ed + M + Prob + U2, data = crime.lm)
##
## 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 ***
## Po1           341.84      40.87   8.363 2.56e-10 ***
## Ineq          269.91      55.60   4.855 1.88e-05 ***
## Ed            219.79      50.07   4.390 8.07e-05 ***
## 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 *
## ---
```

```
## 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
```

```
# Backward Selection
step.back <- stepAIC(crime.full, direction = "backward", trace = FALSE)
summary(step.back)
```

```
##
## Call:
## lm(formula = Crime ~ M + Ed + Po1 + M.F + U1 + U2 + Ineq + Prob,
##     data = crime.lm)
##
## 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)   905.09      28.52  31.731  < 2e-16 ***
## M             117.28      42.10   2.786  0.00828 **
## Ed            201.50      59.02   3.414  0.00153 **
## Po1           305.07      46.14   6.613 8.26e-08 ***
## M.F            65.83      40.08   1.642  0.10874
## U1           -109.73      60.20  -1.823  0.07622 .
## U2            158.22      61.22   2.585  0.01371 *
## Ineq          244.70      55.69   4.394 8.63e-05 ***
## Prob          -86.31      33.89  -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
```

We can see that stepping back we have removed a number of coeddicants. Lets have a look at a model with both forwards and backwarads selection.

```
# Forward and Bakward
step.both <- stepAIC(crime.null, direction = "both",trace = FALSE,
                     scope = ~ M + So + Ed + Po1 + Po2 + LF + M.F + Pop + NW + U1 + U2 + Wealth + Ineq +
summary(step.both)
```

```
##
## Call:
## lm(formula = Crime ~ Po1 + Ineq + Ed + M + Prob + U2, data = crime.lm)
##
## 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 ***
```

```
## Po1           341.84      40.87   8.363 2.56e-10 ***
## Ineq          269.91      55.60   4.855 1.88e-05 ***
## Ed            219.79      50.07   4.390 8.07e-05 ***
## 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 *
## ---
## 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
```

These models look to be ok for a light touch approach to variable selection. The bi-directional selection produced the same model as the forward, and if we ran it with a full model and cut it back it would build the same as the backwards selection. I wouldn't expect more interplay and different models until it was a higher dimensional feature space I suppose.

Part 2: Elastic-Net and Lasso

For this problem I am going to feed it into a grid, and also feed in a grid from alpha = 0 to 1, and also play around with the regularisation term.

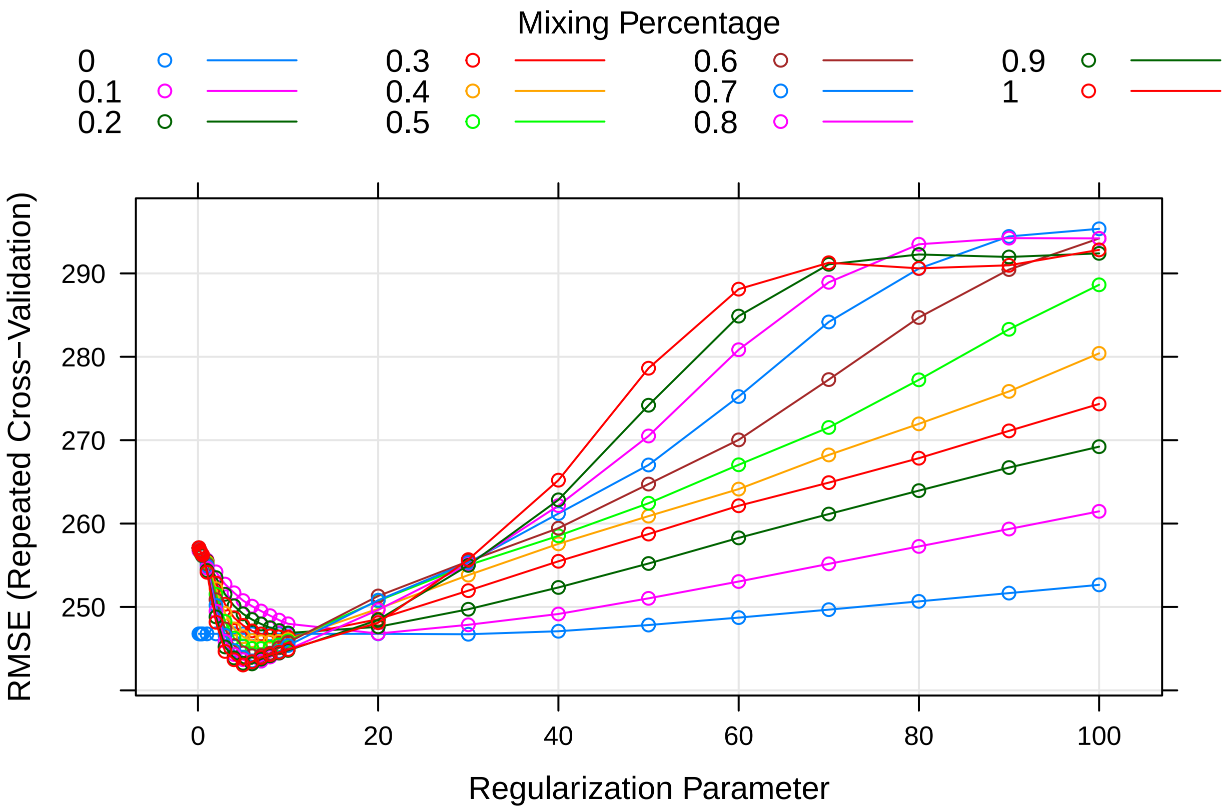
```r
# RIDGE, LASSO, ELASTICNET
library(caret)
library(glmnet)
```

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

```
## Loading required package: foreach
```

```
## Loaded glmnet 2.0-10
```

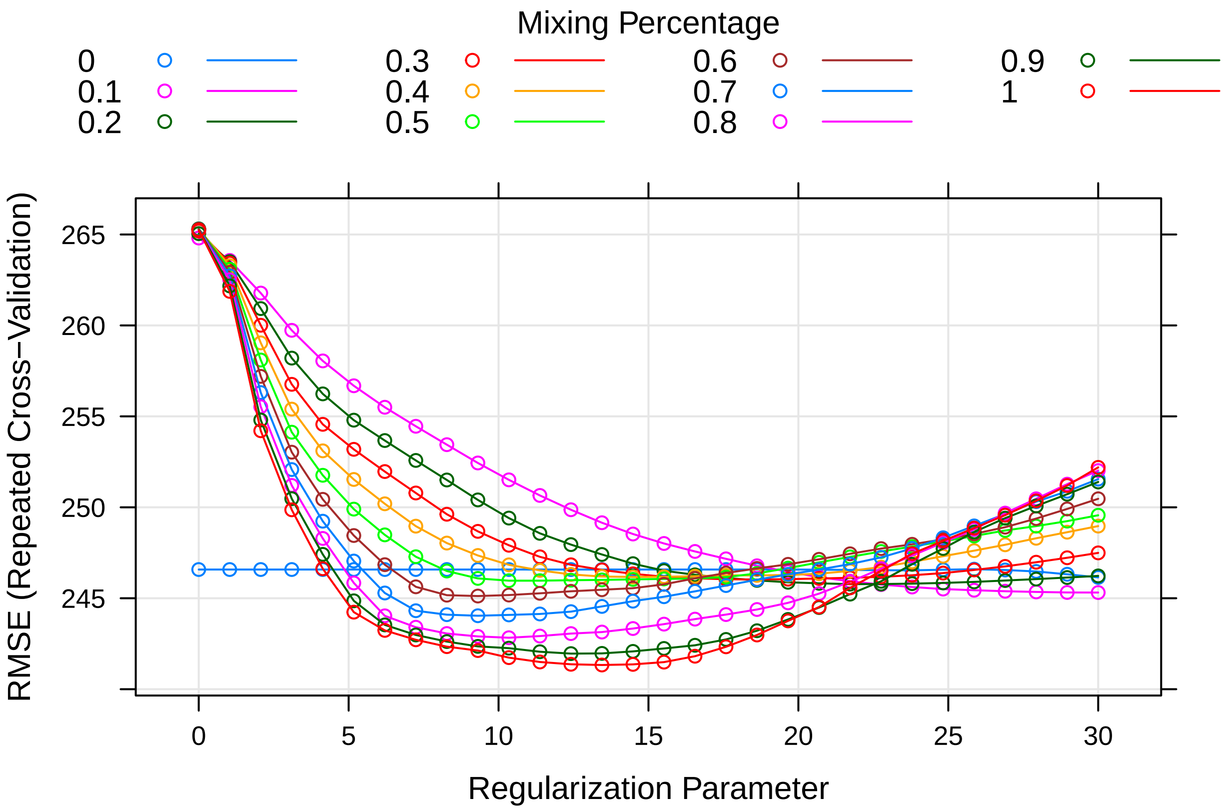
```r
control <- trainControl(method="repeatedcv", number=10, repeats=3)
metric <- "RMSE"
# GLMNET
set.seed(7)
grid <- expand.grid(.alpha = seq(0, 1, length = 11),.lambda = c((1:5)/10,(1:10),(2:10)*10))
fit.glmnetlong <- train(Crime~., data=crime, method="glmnet", family="gaussian", tuneGrid=grid, metric=
plot(fit.glmnetlong)
```



When I first ran this test I had it on much lower regularisation terms and saw no real sensitivity to it. we can wee that we are getting best performance at around 15, so lets run two models in that range, for the second we will do a BoxCos (log) transform on the data. We saw in previous work the BoxCox transform perfoming a bit better.

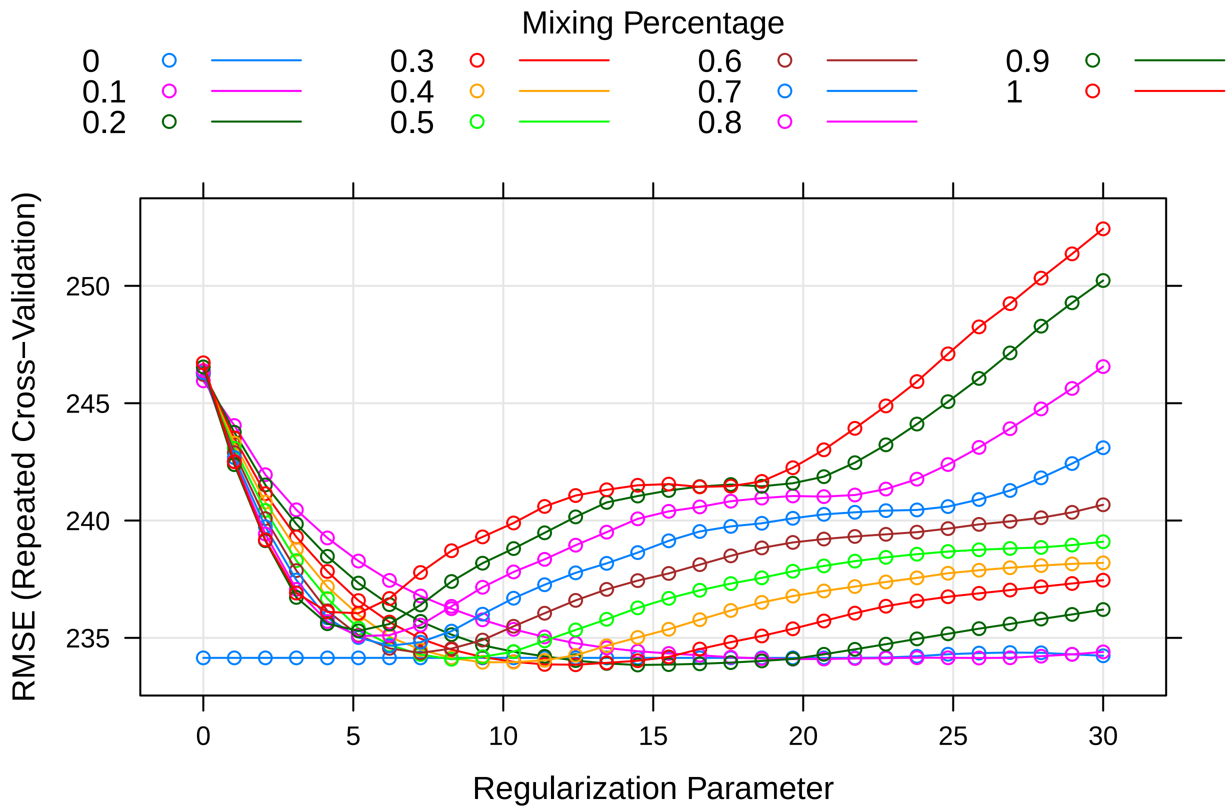
```
grid <- expand.grid(.alpha = seq(.0, 1, length = 11),.lambda = seq(0, 30, length = 30))
fit.glmnet <- train(Crime~., data=crime, method="glmnet", family="gaussian", tuneGrid=grid, metric=metri
fit.glmnetbc <- train(Crime~., data=crime, method="glmnet", family="gaussian", tuneGrid=grid, metric=me

plot(fit.glmnet)
```



We see our ridge regression model performing fairly consistently in terms of RMSE, and the LASSO actually performs best out of the set at around RMSE = 241.3, the Rsq value here is around 66%. Let's have a look at the model with the BoxCox transform:

```
plot(fit.glmnetbc)
```



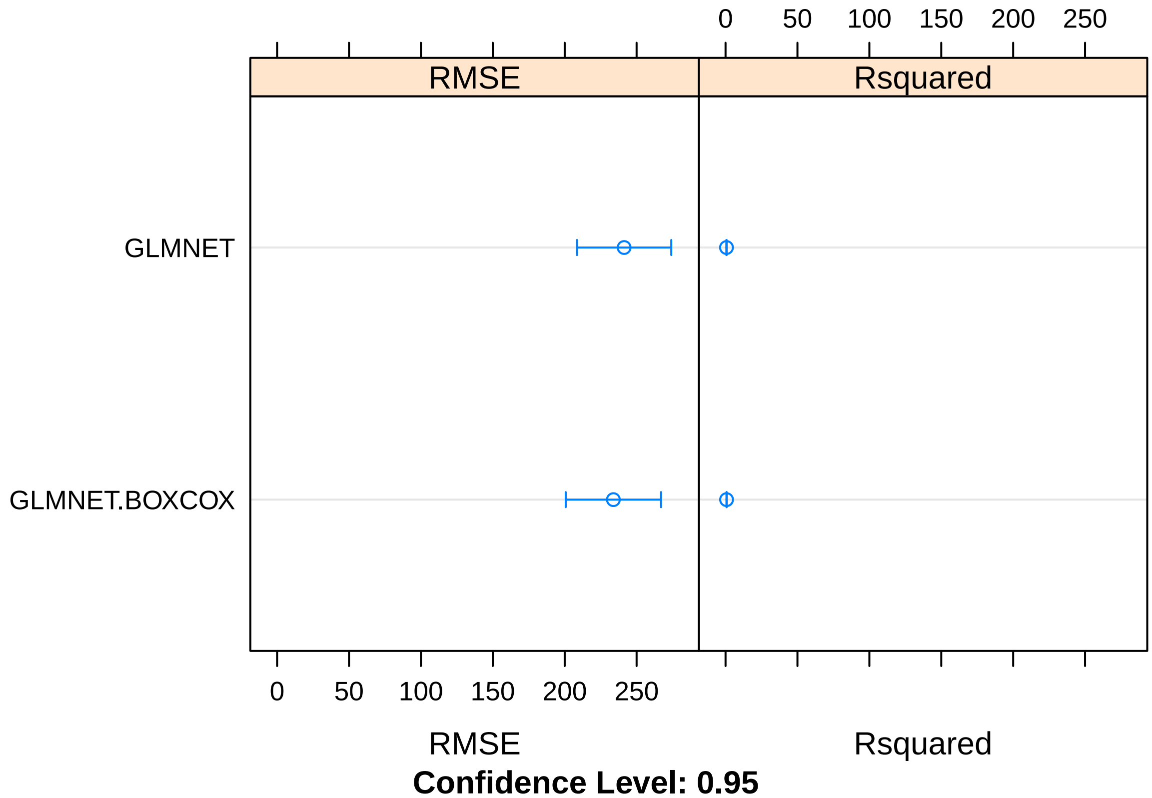
Here the ridge performs consistenly well, will the LASSO performs worse for higher values of regularisation.

Lets compare each set:

```
results <- resamples(list(GLMNET=fit.glmnet, GLMNET.BOXCOX=fit.glmnetbc))
summary(results)
```

```
##
## Call:
## summary.resamples(object = results)
##
## Models: GLMNET, GLMNET.BOXCOX
## Number of resamples: 30
##
## RMSE
##                   Min.  1st Qu.   Median     Mean  3rd Qu.     Max. NA's
## GLMNET        72.94855 188.3645 250.4506 241.3313 300.3241 376.5861    0
## GLMNET.BOXCOX 85.38793 184.3370 214.4836 233.8453 275.0196 395.9513    0
##
## Rsquared
##                     Min.   1st Qu.    Median      Mean   3rd Qu.      Max.
## GLMNET        0.01148215 0.4834443 0.7160835 0.6593070 0.8952263 0.9870348
## GLMNET.BOXCOX 0.29093304 0.6111540 0.7287743 0.7120925 0.8947086 0.9841534
##               NA's
## GLMNET           0
## GLMNET.BOXCOX    0
```

```r
dotplot(results)
```



So again boxcox looks like an overall slighly better fit. With Rsq going up to 98% that data looks over fit. Lets take the parameters from the best performing BC transform as our final model:

```r
results <- fit.glmnetbc$results
results[which.min(results$RMSE),]
```

```
##    alpha   lambda     RMSE  Rsquared   RMSESD RsquaredSD
## 75   0.2 14.48276 233.8453 0.7120925 88.73479  0.1980543
```

So this is an elastic net with an 0.2 weight, and an Rsq of 71. Nice one.

Q2

For facial recognition systems you may design an experiment to test the level of work produced given different configurations of algorithimn tuning and manual resolution. This would allow you to optimise the level of work and risk control / quality produced by the system.

Q3

Below we provide the following to the participants for the survey:

```r
library(FrF2)
```

```
## Loading required package: DoE.base
```

```
## Loading required package: grid

## Loading required package: conf.design

##
## Attaching package: 'DoE.base'

## The following objects are masked from 'package:stats':
##
##     aov, lm

## The following object is masked from 'package:graphics':
##
##     plot.design

## The following object is masked from 'package:base':
##
##     lengths
```

```
DoE <- FrF2(16,10, default.levels = c("In", "Out"),seed=42)
DoE
```

```
##      A   B   C   D   E   F   G   H   J   K
## 1   In Out Out Out  In  In Out  In Out  In
## 2  Out Out Out Out Out Out Out Out Out Out
## 3   In  In Out  In Out  In  In Out Out  In
## 4   In Out  In Out  In Out  In  In  In Out
## 5  Out Out Out  In Out Out Out  In  In  In
## 6  Out  In Out  In  In Out  In  In Out Out
## 7  Out Out  In Out Out  In  In Out  In  In
## 8  Out  In  In  In  In  In Out  In  In  In
## 9   In  In Out Out Out  In  In  In  In Out
## 10 Out  In Out Out  In Out  In Out  In  In
## 11  In Out  In  In  In Out  In Out Out  In
## 12 Out Out  In  In Out  In  In  In Out Out
## 13  In Out Out  In  In  In Out Out  In Out
## 14  In  In  In  In Out Out Out Out  In Out
## 15 Out  In  In Out  In  In Out Out Out Out
## 16  In  In  In Out Out Out Out  In Out  In
## class=design, type= FrF2
```

This will provide fractional factorials to model each's value.

Q4

a. Binomial = the chance my collegue is on facebook when I walk in tomorrow
b. Geometric = the number of times in a week I walk in and my colleague is on facebook
c. Poisson = The number of meteors greater than 1 meter diameter that strike Earth in a year
d. Exponential = can be used to model the time until a radio active particle decays
e. Weibull = can be used in weather forecasting for windspeeds

Looking forward to doing some simulation modelling with these distributions.