HW5

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

Let's start with loading the data....

```
cr_data <- read.table("/Users/chintan/Downloads/6501/crimedata.txt",
                      stringsAsFactors = FALSE, header = TRUE)
head(cr_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
```

Stepwise Regression.... We will try forward, backward, and combine stepwise regression for AIC and BIC...

Forward Regression - AIC

```
library(MASS)
model_null <- lm(Crime~1,data= cr_data)
model_fw <- stepAIC(model_null, direction = "forward",trace =FALSE,
                    scope = ~M + So + Ed + Po1 + Po2 + LF + M.F +
                      Pop + NW + U1 + U2 + Wealth + Ineq + Prob +Time)
summary(model_fw)
```

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

Backward Regression -AIC

```
model_full <- lm(Crime~., data =cr_data)
model_bk <- stepAIC(model_full,direction = "backward", trace = FALSE,
                    scope = ~M + So + Ed + Po1 + Po2 + LF + M.F + Pop
                    + NW + U1 + U2 + Wealth + Ineq + Prob +Time)
summary(model_bk)
```

```
##
## Call:
## lm(formula = Crime ~ M + Ed + Po1 + M.F + U1 + U2 + Ineq + Prob,
##     data = cr_data)
##
## 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
```

Using both direction AIC

```
model_both <- stepAIC(model_null,direction = "both", trace = FALSE,
                      scope = ~M + So + Ed + Po1 + Po2 + LF + M.F + Pop
                      + NW + U1 + U2 + Wealth + Ineq + Prob +Time)
summary(model_both)
```

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

Above methods suggest us there are 6 to 8 important factors. We can build our LM model using these factors.

Let's try with 8 factores ...

```
model_final <- lm(Crime~M + Ed + Po1 + U2 + Ineq + Prob + M.F+ U1, data = cr_data)
summary(model_final)
```

```
##
## Call:
## lm(formula = Crime ~ M + Ed + Po1 + U2 + Ineq + Prob + M.F +
##     U1, data = cr_data)
##
## 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 ***
## 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 *
## M.F            22.34      13.60   1.642  0.10874
## U1          -6086.63    3339.27  -1.823  0.07622 .
## ---
## 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
```

Let’s try with 6 factores

```
model_final1 <- lm(Crime~M + Ed + Po1 + U2 + Ineq + Prob, data = cr_data)
summary(model_final1)
```

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

When we compare LM model with 6 and 8 factors, it seems that model with 8 factors (as suggested by backward and both direction regression) is better than other. This is a good start and gives us the rough estimate of which factors are important. As said in the lectures this method don't perform well and suggests the variables that more fit to random effect and give higher R2 values. Hence, we need to test other regression method as well.....

```
library(glmnet)
```

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

```
## Loaded glmnet 4.0
```

```
set.seed(43)
```

First we need to scale the data ....

```
set.seed(41)
x <- as.matrix(cr_data[,-16])
y <- as.double(as.matrix(cr_data[,16]))
scale_x <- scale(x)
scale_y <- scale(y)

model_lasso <- cv.glmnet(scale_x, y = scale_y,alpha =1, nfolds =5, type.measure = "mse",
  family = "gaussian")
coef(model_lasso,s = model_lasso$lambda.min)
```

```
## 16 x 1 sparse Matrix of class "dgCMatrix"
##                          1
## (Intercept) -3.046715e-16
## M            2.247676e-01
## So           5.738635e-02
## Ed           3.407168e-01
## Po1          7.955957e-01
## Po2          .
## LF           3.021196e-04
## M.F          1.397577e-01
## Pop          .
## NW           1.340837e-02
## U1          -7.733692e-02
## U2           1.665824e-01
## Wealth       .
## Ineq         4.787490e-01
## Prob        -2.148360e-01
## Time         .
```

Let’s built our LM model using these 11 variables suggested by Lasso model...

```
lm_lasso <- lm(Crime~M+So+Ed+Po1+LF+M.F+NW+U1+U2+Ineq+Prob, data = cr_data)
summary(lm_lasso)
```

```
##
## Call:
## lm(formula = Crime ~ M + So + Ed + Po1 + LF + M.F + NW + U1 +
##     U2 + Ineq + Prob, data = cr_data)
##
## Residuals:
##    Min     1Q Median     3Q    Max
## -443.2 -101.4    4.1  120.5  486.2
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept) -6434.101   1253.263  -5.134 1.07e-05 ***
## M              84.825     39.221   2.163  0.03747 *
## So             36.573    139.615   0.262  0.79489
## Ed            186.954     58.101   3.218  0.00278 **
## Po1            99.463     18.338   5.424 4.44e-06 ***
## LF           -264.646   1339.041  -0.198  0.84447
## M.F            25.438     17.353   1.466  0.15159
## NW              1.265      5.783   0.219  0.82814
## U1          -6050.130   3977.786  -1.521  0.13725
## U2            179.349     78.140   2.295  0.02783 *
## Ineq           58.402     16.962   3.443  0.00151 **
## Prob        -4222.327   1740.886  -2.425  0.02059 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 202.9 on 35 degrees of freedom
## Multiple R-squared:  0.7906, Adjusted R-squared:  0.7248
## F-statistic: 12.01 on 11 and 35 DF,  p-value: 6.965e-09
```

Let's built Elastic net.....

```
model_elasticnet <- cv.glmnet(scale_x, y = scale_y,alpha =.5, nfolds =5,
                              type.measure = "mse", family = "gaussian")
coef(model_elasticnet,s = model_elasticnet$lambda.min)
```

```
## 16 x 1 sparse Matrix of class "dgCMatrix"
##                          1
## (Intercept) -3.488284e-16
## M            2.332935e-01
## So           6.110828e-02
## Ed           3.587069e-01
## Po1          6.230021e-01
## Po2          1.169887e-01
## LF           5.775715e-03
## M.F          1.606956e-01
## Pop          .
## NW           4.097378e-02
## U1          -1.308733e-01
## U2           2.240888e-01
## Wealth       6.540920e-02
## Ineq         4.969616e-01
## Prob        -2.237242e-01
## Time         .
```

Let's built our LM model using these 13 variables suggested by Elastic net model and if it increase or reduce the R2 errors...

```
lm_elasticnet <- lm(Crime~M+So+Ed+Po1+Po2+LF+M.F+NW+U1+U2+Ineq+Prob+Wealth, data = cr_dat
  a)
summary(lm_elasticnet)
```

```
##
## Call:
## lm(formula = Crime ~ M + So + Ed + Po1 + Po2 + LF + M.F + NW +
##     U1 + U2 + Ineq + Prob + Wealth, data = cr_data)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -362.46 -111.81  -11.25  110.56  470.07
##
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept) -6.679e+03  1.374e+03  -4.860 2.78e-05 ***
## M            8.598e+01  4.035e+01   2.131  0.04064 *
## So           8.722e+00  1.445e+02   0.060  0.95225
## Ed           1.904e+02  6.083e+01   3.130  0.00365 **
## Po1          1.675e+02  9.828e+01   1.705  0.09767 .
## Po2         -8.570e+01  1.077e+02  -0.796  0.43185
## LF          -7.438e+02  1.425e+03  -0.522  0.60514
## M.F          2.547e+01  1.757e+01   1.450  0.15659
## NW           3.107e+00  6.067e+00   0.512  0.61196
## U1          -5.995e+03  4.044e+03  -1.483  0.14767
## U2           1.644e+02  8.029e+01   2.047  0.04869 *
## Ineq         6.688e+01  2.155e+01   3.103  0.00391 **
## Prob        -4.012e+03  1.813e+03  -2.213  0.03396 *
## Wealth       8.178e-02  1.003e-01   0.816  0.42059
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 205 on 33 degrees of freedom
## Multiple R-squared:  0.7984, Adjusted R-squared:  0.719
## F-statistic: 10.05 on 13 and 33 DF,  p-value: 4.781e-08
```

Well, this model has very slightly increased the R2 value compare to LASSO model. But this elastic net model also suggests that we should use 13 variables instead of 11. I would choose lasso model as it uses two less variables and there is not much difference in the R2 value.

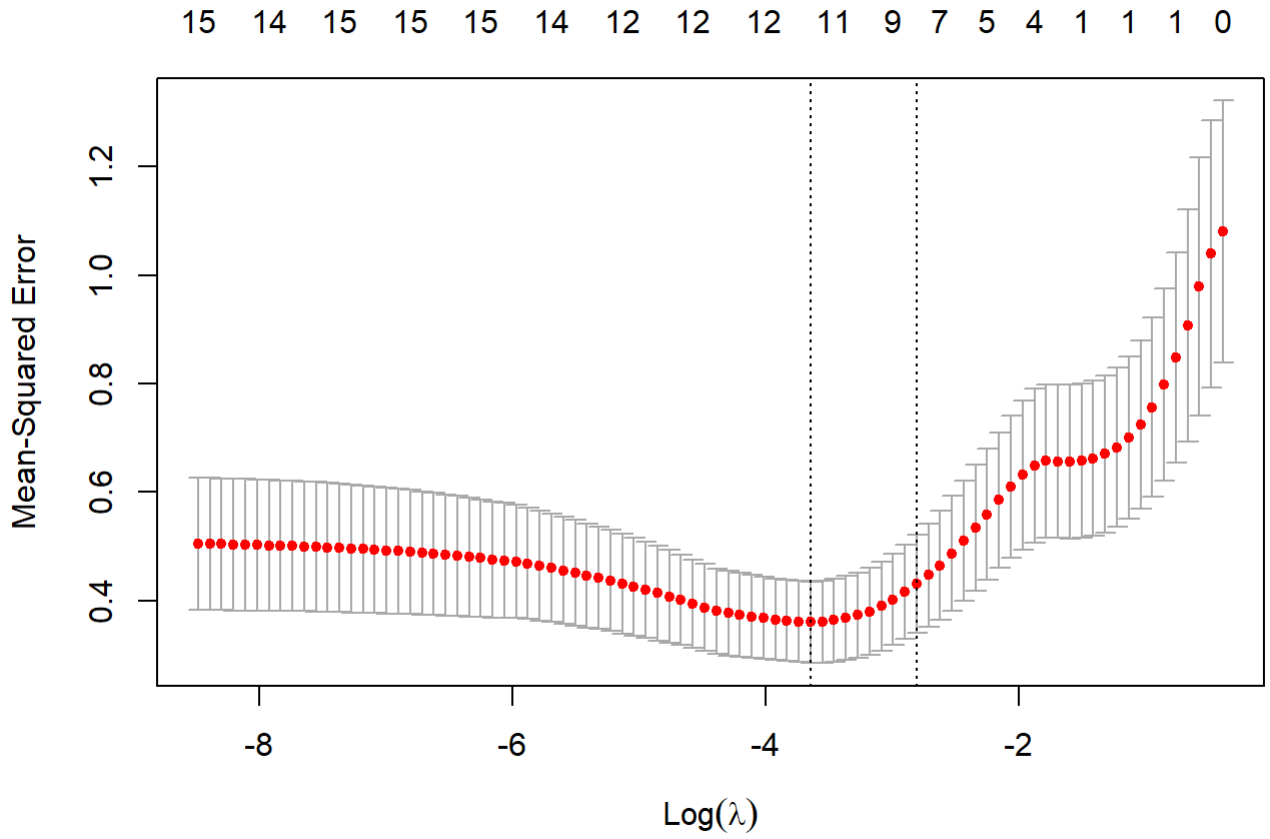
```
model_ridge <- cv.glmnet(scale_x, y_ = scale_y,alpha_ =0, nfolds_ =5,
                         type.measure_ = "mse", family_ = "gaussian")
coef(model_ridge,s_ = model_ridge$lambda.min)
```

```
## 16 x 1 sparse Matrix of class "dgCMatrix"
##                          1
## (Intercept) -3.610205e-16
## M            1.947157e-01
## So           1.130185e-01
## Ed           2.467624e-01
## Po1          3.624676e-01
## Po2          2.838860e-01
## LF           6.211988e-02
## M.F          1.800398e-01
## Pop          9.610372e-03
## NW           8.935072e-02
## U1          -1.123656e-01
## U2           2.011091e-01
## Wealth       9.083300e-02
## Ineq         3.258374e-01
## Prob        -2.180393e-01
## Time         1.359824e-02
```

```
lm_ridge <- lm(Crime~., data = cr_data)
summary(lm_ridge)
```

```
##
## Call:
## lm(formula = Crime ~ ., data = cr_data)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -395.74  -98.09   -6.69  112.99  512.67
##
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept) -5.984e+03  1.628e+03  -3.675 0.000893 ***
## M            8.783e+01  4.171e+01   2.106 0.043443 *
## So          -3.803e+00  1.488e+02  -0.026 0.979765
## Ed           1.883e+02  6.209e+01   3.033 0.004861 **
## Po1          1.928e+02  1.061e+02   1.817 0.078892 .
## Po2         -1.094e+02  1.175e+02  -0.931 0.358830
## LF          -6.638e+02  1.470e+03  -0.452 0.654654
## M.F          1.741e+01  2.035e+01   0.855 0.398995
## Pop         -7.330e-01  1.290e+00  -0.568 0.573845
## NW           4.204e+00  6.481e+00   0.649 0.521279
## U1          -5.827e+03  4.210e+03  -1.384 0.176238
## U2           1.678e+02  8.234e+01   2.038 0.050161 .
## Wealth       9.617e-02  1.037e-01   0.928 0.360754
## Ineq         7.067e+01  2.272e+01   3.111 0.003983 **
## Prob        -4.855e+03  2.272e+03  -2.137 0.040627 *
## Time        -3.479e+00  7.165e+00  -0.486 0.630708
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 209.1 on 31 degrees of freedom
## Multiple R-squared:  0.8031, Adjusted R-squared:  0.7078
## F-statistic: 8.429 on 15 and 31 DF,  p-value: 3.539e-07
```

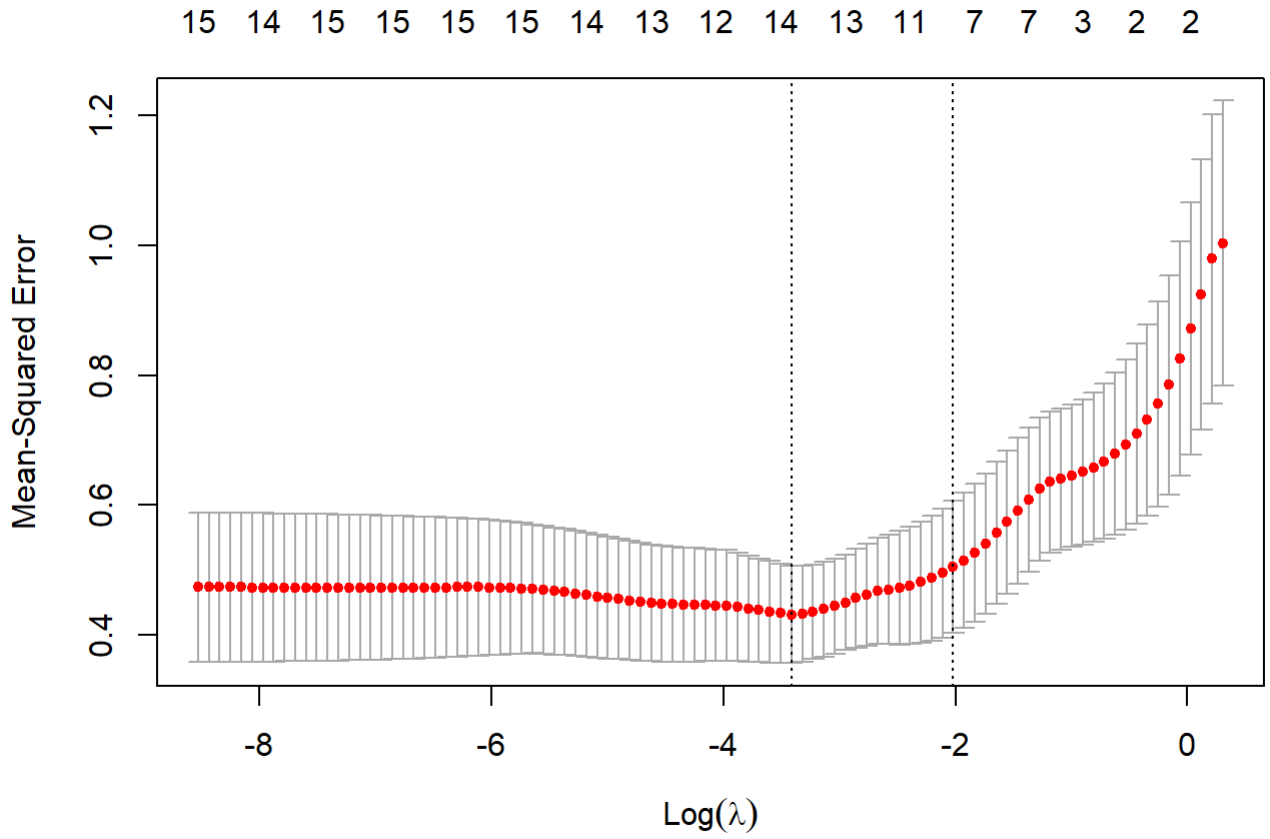
```
plot(model_lasso)
```




```
lambda1 <- model_lasso$lambda.min
model_lasso$cvm[model_lasso$lambda == lambda1]
```

```
## [1] 0.3602918
```

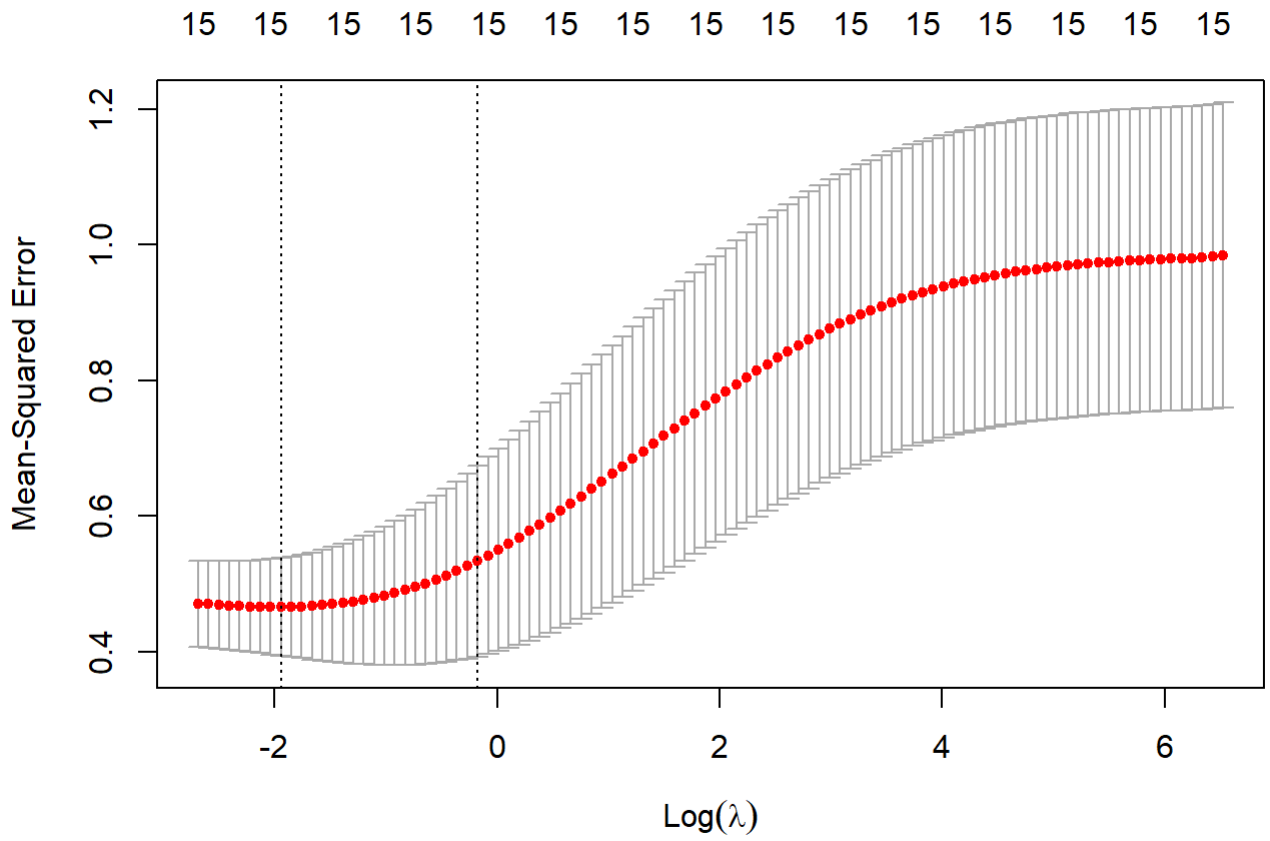
```
plot(model_elasticnet)
```




```
lambda <- model_elasticnet$lambda.min
model_elasticnet$cvm[model_elasticnet$lambda == lambda]
```

```
## [1] 0.4316991
```

```
plot(model_ridge)
```




```
lambda_r <- model_ridge$lambda.min
model_ridge$cvm[model_ridge$lambda == lambda_r]
```

```
## [1] 0.4658285
```

Above three plot compares the lambda min values which gives minimum mean cross- validated error. In the above case lasso model performs the best. Below is the link that gives more detail about lambda..
https://stats.stackexchange.com/questions/77546/how-to-interpret-glmnet
(https://stats.stackexchange.com/questions/77546/how-to-interpret-glmnet)

#Conclusion

From the below matrix we can say that there is not much difference in all of these regression models but LASSO, Elastic net and ridge gives slightly better R2 values than the step regression models as discussed in the video lectures. However if we compare the R2 and Adj. R2 value then the step regression models are doing better as they have less variance hence not over fitted. It is hard to tell as our data set is very small and little randomness will have big impact on the results.

```
R           Squared Adj R Squred    Variables   Diff
```

AIC forward 0.7659 0.7307 6 0.0352

AIC backward 0.7888 0.7444 8 0.0444

AIC both 0.7659 0.7307 6 0.0352

LASSO 0.7906 0.7248 11 0.0658

Elastic net 0.7984 0.719 13 0.0794

Ridge 0.8031 0.7078 15 0.0953

Optional Testing…

To get better understaind between lasso, ridge and elastic net let's divdie the data in to training and test

```
set.seed(42)
n<- nrow(cr_data)
train.rows <- sample(1:n,.70*n)
x.train <- scale_x[train.rows,]
x.test <- scale_x[-train.rows,]
y.train <- scale_y[train.rows,]
y.test <- scale_y[-train.rows,]
```

Now let's apply this to Ridge Regression….

```
set.seed(42)
alpha0.fit <- cv.glmnet(x.train, y.train, type.measure = "mse", alpha =0, family = "gauss
  ian")

alpha0.predict <- predict(alpha0.fit, s= alpha0.fit$lambda.1se, newx=x.test)
```

Lasso Regression

```
set.seed(42)
alpha1.fit <- cv.glmnet(x.train, y.train, type.measure = "mse", alpha =1,
                        family = "gaussian")

alpha1.predict <- predict(alpha1.fit, s= alpha1.fit$lambda.1se, newx=x.test)
```

Elastic net Regression

```
set.seed(42)
alpha0.5.fit <- cv.glmnet(x.train, y.train, type.measure = "mse", alpha =0.5,
                          family = "gaussian")

alpha0.5.predict <- predict(alpha0.5.fit, s= alpha0.5.fit$lambda.1se, newx=x.test)
```

Lets calculate the mean squred error of the difference between the true values and the predicted values…

```
#ridge regression
mean <- ((y.test - alpha0.predict)^2)
sum(mean)
```

```
## [1] 8.730022
```

```
# lasso regression
mean1 <- ((y.test - alpha1.predict)^2)
sum(mean1)
```

```
## [1] 10.17396
```

```
#elastic net regression
mean0.5 <- ((y.test - alpha0.5.predict)^2)
sum(mean0.5)
```

```
## [1] 10.74953
```

Based on the above analysis it looks like Ridge model has lowest mean squared errors and out performs other two models. It matches to our previous analysis of R2 errors. We can also see that there is not much difference between Lasso and Elastic net model.

Question 12.1

Describe a situation or problem from your job, everyday life, current events, etc., for which a design of experiments approach would be appropriate.

What I have observed is if you go to any movie browsing website they have thumbnail images of movies and that images are changed after some time and you will see new thumbnail images for that same movies. These the one way they are trying to use DOE to understand by keeping which image people will click/watch that movie.

Other example is let's say if I want to introduce new perfume to the market. The shape of the bottle and the packaging is really important for this kind of products. I will try to create different kind of bottles and packaging to see which one is more appealing to customers. I will use DOE on a set of customer to understand their buying habits.

Another good example of DOE is found in the chemical industry where they try to optimize the outcome of a certain process when working with the chemical reactions. There is lot of variables involved in the process and slight change in one could make big impact. This is where they used DOE to indentify the exact balance between different variables involved and find tune them until the desired outcome is met.

Question 12.2

To determine the value of 10 different yes/no features to the market value of a house (large yard, solar roof, etc.), a real estate agent plans to survey 50 potential buyers, showing a fictitious house with different combinations of features. To reduce the survey size, the agent wants to show just 16 fictitious houses. Use R's FrF2 function (in the FrF2 package) to find a fractional factorial design for this experiment: what set of features should each of the 16 fictitious houses have? Note: the output of FrF2 is "1" (include) or "-1" (don't include) for each feature.

```
set.seed(1)
library(FrF2)
```

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

```
## Loading required package: grid
```

```
## Loading required package: conf.design
```

```
## Registered S3 method overwritten by 'DoE.base':
##   method           from
##   factorize.factor 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
```

nruns = 16 fictious houses nfactors= 10 features

```
FrF2(nruns= 16, nfactors = 10)
```

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

Question 13.1

For each of the following distributions, give an example of data that you would expect to follow this distribution (besides the examples already discussed in class).

```
        1) Weather prediction like will it rain tomorrow? Will have answer as Yes or No
```

b. Geometric : 1)You can throw the dart at a bulls eye until you hit the center.. 2) You can play a card game where you have to flip the card until you get Jack or Queen

c. Poisson : Number of events happing in a time period. FOr example number of car passing the toll booth can follow the Poisson distribution.

d. Exponential: It is used to model the time elapsed between events 1) Time between the car passes through the toll booth is an example of exponential distribution 2) The Auto insurance company uses exponential distribution to check the risk of a client getting in an accident by observing the time between two accidents. This can help in determining the price for insurance policy.

e. Weibull: Mainly used in analysis of time to failure.

1. Many companies that give warranty on their products can use Weibull distribution to measure the amount time before the product or its parts break. Based on this they can find out the probability of parts failing after 1year, 2 year....