ISYE HW8

11.1]

1)

The dataset crime.txt was first divided into 70% train and 30% test by randomization. An initial model was trained with summary of model as shown in Fig [1]

```
> summary(model_1)

Call:
lm(formula = Crime ~ ., data = train_data)

Residuals:
    Min      1Q  Median      3Q     Max 
-239.23  -59.16  -11.66   75.96  200.66 

Coefficients:
              Estimate Std. Error t value Pr(>|t|)    
(Intercept) -8.971e+03  1.744e+03  -5.144  9.8e-05 ***
M            3.788e+01  3.952e+01   0.959 0.352068    
So          -4.780e+01  1.300e+02  -0.368 0.717824    
Ed           9.898e+01  5.471e+01   1.809 0.089270 .  
Po1          2.348e+02  1.064e+02   2.208 0.042221 *  
Po2         -1.694e+02  1.162e+02  -1.458 0.164158    
LF          -2.241e+03  1.296e+03  -1.729 0.103126    
M.F          7.051e+01  2.179e+01   3.236 0.005176 ** 
Pop          8.538e-01  1.110e+00   0.769 0.453018    
NW           8.469e+00  6.284e+00   1.348 0.196559    
U1          -1.369e+04  4.913e+03  -2.786 0.013219 *  
U2           1.783e+02  1.021e+02   1.746 0.099982 .  
Wealth       2.389e-01  9.477e-02   2.521 0.022691 *  
Ineq         8.579e+01  2.041e+01   4.204 0.000673 ***
Prob        -6.911e+03  1.984e+03  -3.484 0.003065 ** 
Time        -1.185e+00  6.359e+00  -0.186 0.854492    
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 155.8 on 16 degrees of freedom
Multiple R-squared:  0.9319,	Adjusted R-squared:  0.8681 
F-statistic:  14.6 on 15 and 16 DF,  p-value: 1.368e-06
```

Fig[1]

The model has an adjusted R squared value on 86.81%. This was used as a baseline to predict if the model gets better using the stepwise regression modelling method. A new model called step_model was then generated using the stepAIC() function. The summary of the model is shown in Fig[2]

```
> summary(step_model)

Call:
lm(formula = Crime ~ Ed + Po1 + Po2 + LF + M.F + NW + U1 + U2 + 
    Wealth + Ineq + Prob, data = train_data)

Residuals:
    Min      1Q  Median      3Q     Max 
-243.85  -42.67  -11.93   97.90  171.44 

Coefficients:
              Estimate Std. Error t value Pr(>|t|)    
(Intercept) -8.361e+03  1.397e+03  -5.986 7.46e-06 ***
Ed           8.813e+01  4.930e+01   1.788 0.089009 .  
Po1          2.435e+02  8.859e+01   2.749 0.012378 *  
Po2         -1.708e+02  9.425e+01  -1.812 0.085074 .  
LF          -1.994e+03  1.030e+03  -1.936 0.067136 .  
M.F          6.912e+01  1.645e+01   4.202 0.000439 ***
NW           1.026e+01  4.630e+00   2.216 0.038466 *  
U1          -1.235e+04  4.245e+03  -2.909 0.008672 ** 
U2           1.401e+02  9.004e+01   1.556 0.135480    
Wealth       2.290e-01  8.444e-02   2.712 0.013413 *  
Ineq         8.675e+01  1.761e+01   4.927 8.13e-05 ***
Prob        -7.768e+03  1.492e+03  -5.205 4.30e-05 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 145.8 on 20 degrees of freedom
Multiple R-squared:  0.9255,	Adjusted R-squared:  0.8845 
F-statistic: 22.59 on 11 and 20 DF,  p-value: 6.765e-09
```

Fig[2]

The summary of step_model shows that the variables M,So,Pop and Time were removed based on stepAIC() function based on the p values that do not appear to be good for the model prediction. The adjusted R squared value for step_model was 88.45% which indicates that the fit was improved from my original model. The coefficients on the step_model were as shown in Fig[3]

```
> step_model

Call:
lm(formula = Crime ~ Ed + Po1 + Po2 + LF + M.F + NW + U1 + U2 +
    Wealth + Ineq + Prob, data = train_data)

Coefficients:
(Intercept)           Ed          Po1          Po2           LF          M.F           NW
  -8360.565       88.130      243.523     -170.759    -1993.898       69.121       10.259
         U1           U2       Wealth         Ineq         Prob
 -12350.360      140.066        0.229       86.747    -7767.710
```

Fig[3]

The step_model were then used to predict the values on the test data set with summary as shown in Fig[4]

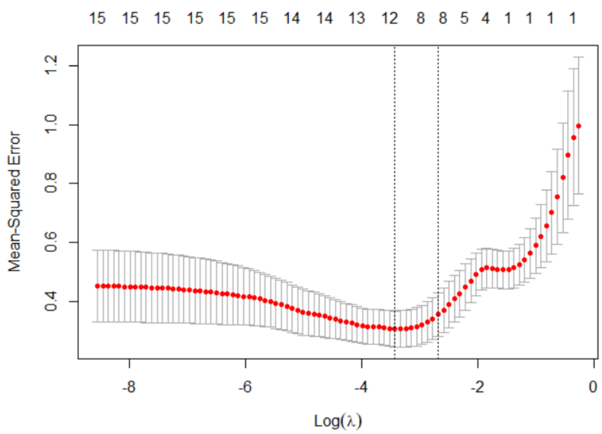
```
> summary(model_2)

Call:
lm(formula = Crime ~ Ed + Po1 + Po2 + LF + M.F + NW + U1 + U2 +
    Wealth + Ineq + Prob, data = test_data)

Residuals:
       1        3        5        9       15       17       24       27       29       32       33
-116.436  -71.034    1.636  173.921   39.211  -11.445   39.454  -28.096   22.053  -15.203    5.034
      42       43       46       47
  62.033  -11.716  -81.843   -7.570

Coefficients:
              Estimate Std. Error t value Pr(>|t|)
(Intercept) -2.362e+03  1.909e+03  -1.238   0.3039
Ed           3.504e+02  1.466e+02   2.390   0.0968 .
Po1          9.812e+01  1.844e+02   0.532   0.6316
Po2         -6.005e+01  1.957e+02  -0.307   0.7791
LF          -8.473e+01  1.876e+03  -0.045   0.9668
M.F         -2.135e+01  2.258e+01  -0.945   0.4142
NW           2.257e+00  7.228e+00   0.312   0.7753
U1          -6.225e+02  7.614e+03  -0.082   0.9400
U2           4.789e+01  1.494e+02   0.321   0.7696
Wealth      -7.014e-02  2.597e-01  -0.270   0.8046
Ineq         9.206e+01  3.525e+01   2.612   0.0796 .
Prob        -4.777e+03  4.773e+03  -1.001   0.3906
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 146.5 on 3 degrees of freedom
Multiple R-squared:  0.9225,    Adjusted R-squared:  0.6385
F-statistic: 3.248 on 11 and 3 DF,  p-value: 0.1807
```

Fig[4]

On the test data set the model had an adjust R squared value on 63.85%.

2)

Similar to the first part of this problem, the data was divided into training and testing data set. For the lasso method the data set needs to be in a matric form so the variables and responses were converted into a matrix format. The data was also scaled as a part of converting them into a matrix. To model using the lasso method, first the cv.glmnet() function was used on the training data so multiple values of the threshold lambda can be used in modelling to find the lowest mean squared error for our model. The visual results of this run and the min lamda value as shown in Fig[5]




```
> best_lamda
[1] 0.03231453
```

Fig[5]

Using the best lambda value, a new model was generated to drop the variables that were identified to not be as critical in providing a better model. The coefficients on the variables used in modelling the best_lasso_model are as shown in Fig[6]

```
> coef(best_lasso_model)
16 x 1 sparse Matrix of class "dgCMatrix"
                        s1
(Intercept) -2.391440e-16
M            6.120543e-02
So           .
Ed           .
Po1          8.094835e-01
Po2          .
LF           .
M.F          1.312972e-01
Pop          3.862504e-03
NW           .
U1          -1.737874e-02
U2           .
Wealth       .
Ineq         2.322744e-01
Prob        -1.602647e-01
Time         .
```

Fig[6]

It shows that the variables M,Po1,M.F<U1,Ineq and Prob were identified as the most critical in building a regression model using the lasso method. The new model was then used to measure the R squared

values and the root mean squared error (RMSE) values on training data and test data. The values of these as shown in Fig[7] and Fig[8]

```
> R_square
[1] 0.7708589
> RMSE
[1] 0.471148
```

Fig[7]: Train Data

```
> R_square_2
[1] 0.2786975
> RMSE_2
[1] 0.8204972
```

Fig[8]: Test Data

Comparing the RMSE the model provides a great fit with an RMSE value of 0.4711 on the training data but a comparatively poor fit on the test data with an RMSE value of 0.820. This can be attributed to the lower sample size of the dataset which potentially increase the randomness in the regression model.

3)

Using the elastic net method, similar data preparation techniques were used as in the lasso method. The model was trained using caret function with the glmnet method. Cross verification was set to 10 folds as the standard. The model was then used to find the best alpha and lambda values using the $bestTune command. The best prediction values were as shown in Fig[9]

```
> eng_model$bestTune
   alpha     lambda
97     1 0.05365617
```

Fig[9]

The alpha value of 1 indicates that the best lambda value was found when the prediction reverted back to the lasso method. To verify if using this model the quality of fit changes, the R squared value and the RMSE value on training and testing dataset was measured. The results are as shown in Fig[10] and Fig[11]

```
> R_square
[1] 0.8247221
> RMSE
[1] 0.4120685
```

Fig[10]: Train Data

```
> R_square_2
[1] 0.2645312
> RMSE_2
[1] 0.8285153
```

Fig[11]: Test Data

We notice that the model performs better on the training dataset with an RMSE of 0.412 Vs testing dataset with an RMSE of 0.829. In comparison to the RMSE values from models using the strictly the lasso method, the RMSE values are very similar on both the training and the testing dataset. Which supports that our best method for this dataset should be using the lasso regression model.

R Codes for Reference:

1)

```
rm(list=ls())
set.seed(100)
#Read dataset
crime_data=read.table("M:/OMSA/ISYE6501/HW5/crime.txt",header=TRUE)
head(crime_data)

#Divide data in training and testing data
random_row=sample(1:nrow(crime_data),as.integer(0.7*nrow(crime_data),replace=FALSE))
train_data=crime_data[random_row,]
test_data=crime_data[-random_row,]

#converting variables and response into matrix for lasso method
xtrain=scale(as.matrix(train_data)[,-16],center=TRUE,scale=TRUE)
ytrain=scale(as.matrix(train_data)[,16],center=TRUE,scale=TRUE)
xtest=scale(as.matrix(test_data)[,-16],center=TRUE,scale=TRUE)
ytest=scale(as.matrix(test_data)[,16],center=TRUE,scale=TRUE)

#use lasso method
library("glmnet")
cv.lasso=cv.glmnet(xtrain,ytrain,alpha=1,family="gaussian")
plot(cv.lasso)

#find best lamda
best_lamda=cv.lasso$lambda.min
best_lamda

#generate model using best lamda
best_lasso_model=cv.glmnet(xtrain,ytrain,alpha=1,family="gaussian",lamda=best_lamda)
coef(best_lasso_model)
predict_train=predict(best_lasso_model,xtrain)
SSE <- sum((predict_train-ytrain)^2)
SST <- sum((ytrain - mean(ytrain))^2)
R_square <- 1 - SSE / SST
RMSE = sqrt(SSE/nrow(train_data))
R_square
RMSE

#prediction on test model
predict_test=predict(best_lasso_model,xtest)
SSE_2 <- sum((predict_test-ytest)^2)
SST_2 <- sum((ytest - mean(ytest))^2)
R_square_2 <- 1 - SSE_2 / SST_2
RMSE_2 = sqrt(SSE_2/nrow(test_data))
R_square_2
RMSE_2
```

2)

```
rm(list=ls())
set.seed(100)

#Read dataset
crime_data=read.table("M:/OMSA/ISYE6501/HW5/crime.txt",header=TRUE)
head(crime_data)

#Divide data in training and testing data
random_row=sample(1:nrow(crime_data),as.integer(0.7*nrow(crime_data),replace=FALSE))
train_data=crime_data[random_row,]
test_data=crime_data[-random_row,]

#stepwise regression
library(MASS)

#initial model
model_1=lm(Crime~.,data=train_data)
summary(model_1)

#stepwise function model
step_model=stepAIC(model_1,direction="both",trace=FALSE)
step_model
summary(step_model)

#check model fit on test data
model_2=lm(Crime~Ed+Po1+Po2+LF+M.F+NW+U1+U2+Wealth+Ineq+Prob,data=test_data)
summary(model_2)

```

3)

```
rm(list=ls())
set.seed(100)
library("caret")

#Read dataset
crime_data=read.table("M:/OMSA/ISYE6501/HW5/crime.txt",header=TRUE)
head(crime_data)

#Divide data in training and testing data
random_row=sample(1:nrow(crime_data),as.integer(0.7*nrow(crime_data),replace=FALSE))
train_data=crime_data[random_row,]
test_data=crime_data[-random_row,]

#converting variables and response into matrix for eng method
xtrain=scale(as.matrix(train_data)[,-16],center=TRUE,scale=TRUE)
ytrain=scale(as.matrix(train_data)[,16],center=TRUE,scale=TRUE)
xtest=scale(as.matrix(test_data)[,-16],center=TRUE,scale=TRUE)
ytest=scale(as.matrix(test_data)[,16],center=TRUE,scale=TRUE)

#make model
eng_model=train(Crime~.,data=as.matrix(scale(train_data)),method="glmnet",trControl=trainControl("cv",number=10),preProcess=c("center","scale"),tuneLength=10)
eng_model$bestTune



#prediction on train
predict_train=predict(eng_model, xtrain)
predict_train=predict(eng_model,xtrain)
SSE <- sum((predict_train-ytrain)^2)
SST <- sum((ytrain - mean(ytrain))^2)
R_square <- 1 - SSE / SST
RMSE = sqrt(SSE/nrow(train_data))
R_square
RMSE

#prediction on test
predict_test=predict(eng_model,xtest)
SSE_2 <- sum((predict_test-ytest)^2)
SST_2 <- sum((ytest - mean(ytest))^2)
R_square_2 <- 1 - SSE_2 / SST_2
RMSE_2 = sqrt(SSE_2/nrow(test_data))
R_square_2
RMSE_2
```