HW8_ISYE_6501

10/11/2020

Loading libraries

```
require(ggthemes)
library(tidyverse)
library(magrittr)
library(TTR)
library(tidyr)
library(dplyr)
library(lubridate)
library(ggplot2)
library(plotly)
library(fpp2)
library(forecast)
library(caTools)
library(reshape2)
library(psych)
require(graphics)
require(Matrix)
library(corrplot)
library(mltools)
library(fBasics)
library(kableExtra)
library(DMwR)
library(caret)
library(gridExtra)
library(leaps)
library(MASS)
library(glmnet)
```

Load Data

```
df <- read.csv(file="uscrime.csv",stringsAsFactors = F, header=T)
head(df,2)
```

```
##      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
##      Time Crime
## 1 26.2011   791
## 2 25.2999  1635
```

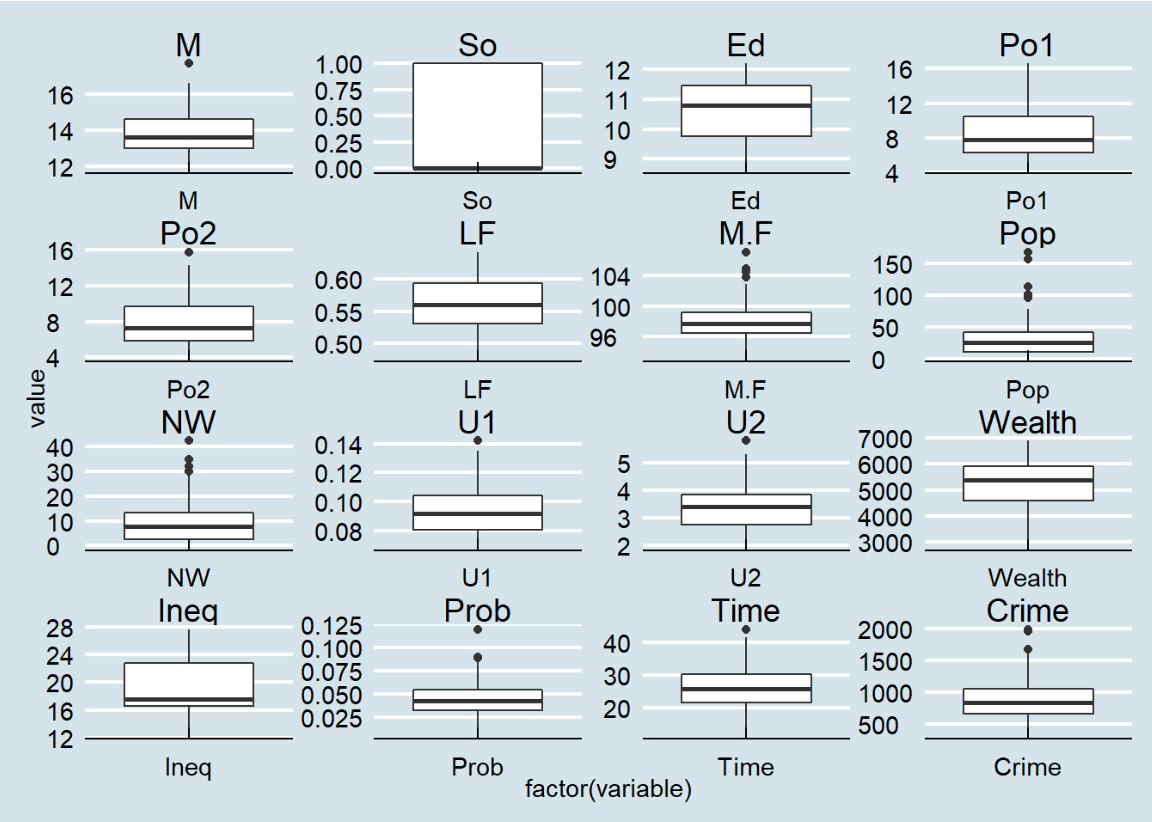
- Data exploration & Simple Visualizations

```
#Statistical Summary of original data
df_unistats<- basicStats(df)[c("Minimum", "Maximum", "1. Quartile", "3. Quartile", "Mean", "Median",
                               "Variance", "Stdev", "Skewness", "Kurtosis"), ] %>% t() %>% as.data.frame()

kable(df_unistats)
```

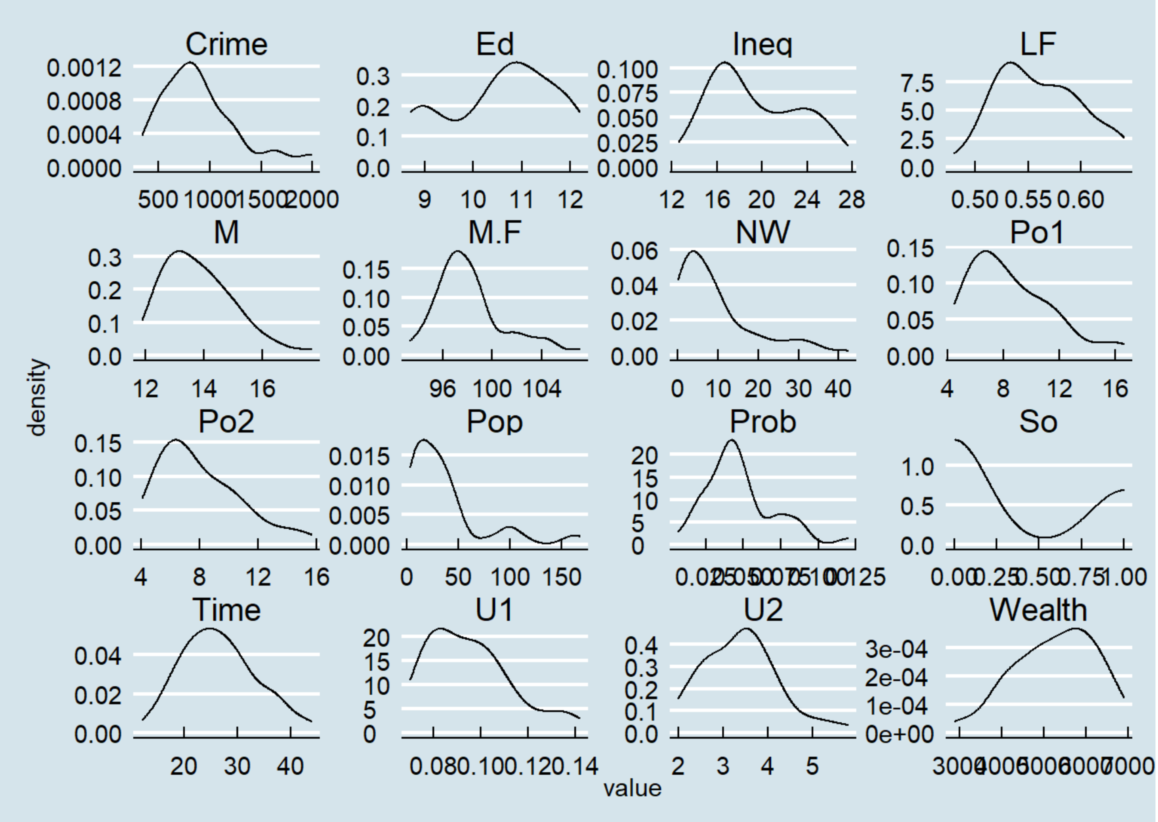
	Minimum	Maximum	1. Quartile	3. Quartile	Mean	Median	Variance	Stdev	Skewness	Kurtosis
M	11.9000	17.700000	13.000000	14.60000	13.857447	13.6000	1.579454e+00	1.256763	0.821917	0.377753
So	0.0000	1.000000	0.000000	1.00000	0.340426	0.0000	2.294170e-01	0.478975	0.652139	-1.607569
Ed	8.7000	12.200000	9.750000	11.45000	10.563830	10.8000	1.251489e+00	1.118700	-0.318987	-1.149253
Po1	4.5000	16.600000	6.250000	10.45000	8.500000	7.8000	8.832174e+00	2.971897	0.890124	0.162309
Po2	4.1000	15.700000	5.850000	9.70000	8.023404	7.3000	7.818353e+00	2.796132	0.844367	0.008590

	Minimum	Maximum	1. Quartile	3. Quartile	Mean	Median	Variance	Stdev	Skewness	Kurtosis
LF	0.4800	0.641000	0.530500	0.59300	0.561191	0.5600	1.633000e-03	0.040412	0.270675	-0.888732
M.F	93.4000	107.100000	96.450000	99.20000	98.302128	97.7000	8.683256e+00	2.946737	0.993223	0.652010
Pop	3.0000	168.000000	10.000000	41.50000	36.617021	25.0000	1.449415e+03	38.071188	1.854230	3.078936
NW	0.2000	42.300000	2.400000	13.25000	10.112766	7.6000	1.057377e+02	10.282882	1.379966	1.077648
U1	0.0700	0.142000	0.080500	0.10400	0.095468	0.0920	3.250000e-04	0.018029	0.774876	-0.131208
U2	2.0000	5.800000	2.750000	3.85000	3.397872	3.4000	7.132560e-01	0.844545	0.542264	0.173008
Wealth	2880.0000	6890.000000	4595.000000	5915.00000	5253.829787	5370.0000	9.310502e+05	964.909442	-0.381952	-0.613169
Ineq	12.6000	27.600000	16.550000	22.75000	19.400000	17.6000	1.591696e+01	3.989606	0.367063	-1.138824
Prob	0.0069	0.119804	0.032701	0.05445	0.047091	0.0421	5.170000e-04	0.022737	0.883336	0.749579
Time	12.1996	44.000400	21.600350	30.45075	26.597921	25.8006	5.022408e+01	7.086895	0.371275	-0.413474
Crime	342.0000	1993.000000	658.500000	1057.50000	905.085106	831.0000	1.495854e+05	386.762697	1.053927	0.777628

```
# Visualizations via Box plots
meltData <- melt(df)

p <- ggplot(meltData, aes(factor(variable), value))
p + geom_boxplot() + facet_wrap(~variable, scale="free")+theme_economist()+scale_colour_economist()
```




```
#density plots of original data
density <- df %>%
  gather() %>%
  ggplot(aes(value)) +
    facet_wrap(~ key, scales = "free") +
    geom_density()+theme_economist()+scale_colour_economist()
density
```



Stepwise Regression

```
# Set seed for reproducibility
set.seed(123)
#Generate a random sample of 90% of the rows
random_row<- sample(1:nrow(df ),as.integer(0.9*nrow(df),replace=F))
traindata = df[random_row,]
#Assign the test data set to the remaining 10% of the original set
testdata = df [-random_row,]
# Set up repeated k-fold cross-validation
train.control <- trainControl(method = "cv", number = 10)
# Stepwise regression model
# Train the model
step.model <- train(Crime ~., data = traindata ,
                    method = "lmStepAIC",
                    trControl = train.control,trace=F
                    )
#model accuracy
step.model$results
```

```
##   parameter     RMSE  Rsquared      MAE    RMSESD RsquaredSD    MAESD
## 1      none 285.9305 0.6092788 225.0695 105.3426  0.2387865 71.53864
```

- Best Model # of Predictors Combo

```
# Final model coefficients
step.model$finalModel
```

```
##
## Call:
## lm(formula = .outcome ~ M + Ed + Po1 + M.F + U1 + U2 + Ineq +
##     Prob, data = dat)
##
## Coefficients:
## (Intercept)            M           Ed          Po1          M.F           U1
##    -6557.63        87.10       173.41        98.34        27.00     -7394.41
##          U2         Ineq         Prob
##      206.41        56.57     -3489.88
```

```
# Summary of the model
summary(step.model$finalModel)
```

```
##
## Call:
## lm(formula = .outcome ~ M + Ed + Po1 + M.F + U1 + U2 + Ineq +
##     Prob, data = dat)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -439.19 -116.92   -4.76  127.15  474.12
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept) -6557.63    1359.17  -4.825 3.09e-05 ***
## M              87.10      34.76   2.506 0.017312 *
## Ed            173.41      55.87   3.104 0.003903 **
## Po1            98.34      16.22   6.064 7.99e-07 ***
## M.F            27.00      15.20   1.777 0.084851 .
## U1          -7394.41    3702.00  -1.997 0.054080 .
## U2            206.41      77.94   2.648 0.012312 *
## Ineq           56.57      15.02   3.766 0.000651 ***
## Prob        -3489.88    1578.65  -2.211 0.034100 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 200.7 on 33 degrees of freedom
## Multiple R-squared:  0.7873, Adjusted R-squared:  0.7358
## F-statistic: 15.27 on 8 and 33 DF,  p-value: 4.342e-09
```

- Evaluation on train model

```
#setting a "full model"
full.model <- lm(Crime ~ M + Ed + Po1 + U2+M.F+U1+U2+ Ineq + Prob,data = traindata) # on train data set
# Stepwise regression model- in both directions
stepfinal.model <- stepAIC(full.model, direction = "both",
                      trace = FALSE,k=2)
#model accuracy
summary(stepfinal.model)
```

```
##
## Call:
## lm(formula = Crime ~ M + Ed + Po1 + U2 + M.F + U1 + U2 + Ineq +
##     Prob, data = traindata)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -439.19 -116.92   -4.76  127.15  474.12
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept) -6557.63    1359.17  -4.825 3.09e-05 ***
## M              87.10      34.76   2.506 0.017312 *
## Ed            173.41      55.87   3.104 0.003903 **
## Po1            98.34      16.22   6.064 7.99e-07 ***
## U2            206.41      77.94   2.648 0.012312 *
## M.F            27.00      15.20   1.777 0.084851 .
## U1          -7394.41    3702.00  -1.997 0.054080 .
## Ineq           56.57      15.02   3.766 0.000651 ***
## Prob        -3489.88    1578.65  -2.211 0.034100 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 200.7 on 33 degrees of freedom
## Multiple R-squared:  0.7873, Adjusted R-squared:  0.7358
## F-statistic: 15.27 on 8 and 33 DF,  p-value: 4.342e-09
```

- lesser significant predictors

```
#setting a "smaller leaner model"
full1.model <- lm(Crime ~ M + Ed + Po1 + Ineq + Prob,data = (traindata)) # on train data set
# Stepwise regression model- in both directions
stepfinal1.model <- stepAIC(full1.model, direction = "both",
                      trace = FALSE,k=2)
#model accuracy
summary(stepfinal1.model)
```

```
##
## Call:
## lm(formula = Crime ~ M + Ed + Po1 + Ineq + Prob, data = (traindata))
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -520.80  -80.81   -9.98  156.62  511.52
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept) -3803.59     872.91  -4.357 0.000105 ***
## M              76.04      33.96   2.239 0.031423 *
## Ed            146.12      46.92   3.115 0.003604 **
## Po1           119.88      14.76   8.122 1.18e-09 ***
## Ineq           64.54      15.61   4.135 0.000203 ***
## Prob        -3622.43    1696.34  -2.135 0.039600 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 216.3 on 36 degrees of freedom
## Multiple R-squared:  0.7305, Adjusted R-squared:  0.693
## F-statistic: 19.51 on 5 and 36 DF,  p-value: 2.303e-09
```

- Evaluating on train & test data

```
#create the evaluation metrics function
eval_metrics = function(model, df, predictions, target){
    resids = df[,target] - predictions
    resids2 = resids**2
    N = length(predictions)
    r2 = as.character(round(summary(model)$r.squared, 2))
    adj_r2 = as.character(round(summary(model)$adj.r.squared, 2))
    print(adj_r2) #Adjusted R-squared
    print(as.character(round(sqrt(sum(resids2)/N), 2))) #RMSE
}
predictions.train = predict(stepfinal1.model, newdata = (traindata))
predictions.test = predict(stepfinal1.model, newdata = testdata)
#model accuracy
eval_metrics(stepfinal1.model, traindata, predictions.train, target = 'Crime')
```

```
## [1] "0.69"
## [1] "200.27"
```

```
eval_metrics(stepfinal1.model, testdata, predictions.test, target = 'Crime')
```

```
## [1] "0.69"
## [1] "162.05"
```

Observation1:

- Cross-validation to filter out unwanted predictors and fed that into our Stepwise (both directions) on training data
- The R-squares for both training and test data is the same while RSME was surprisingly better in test set than training

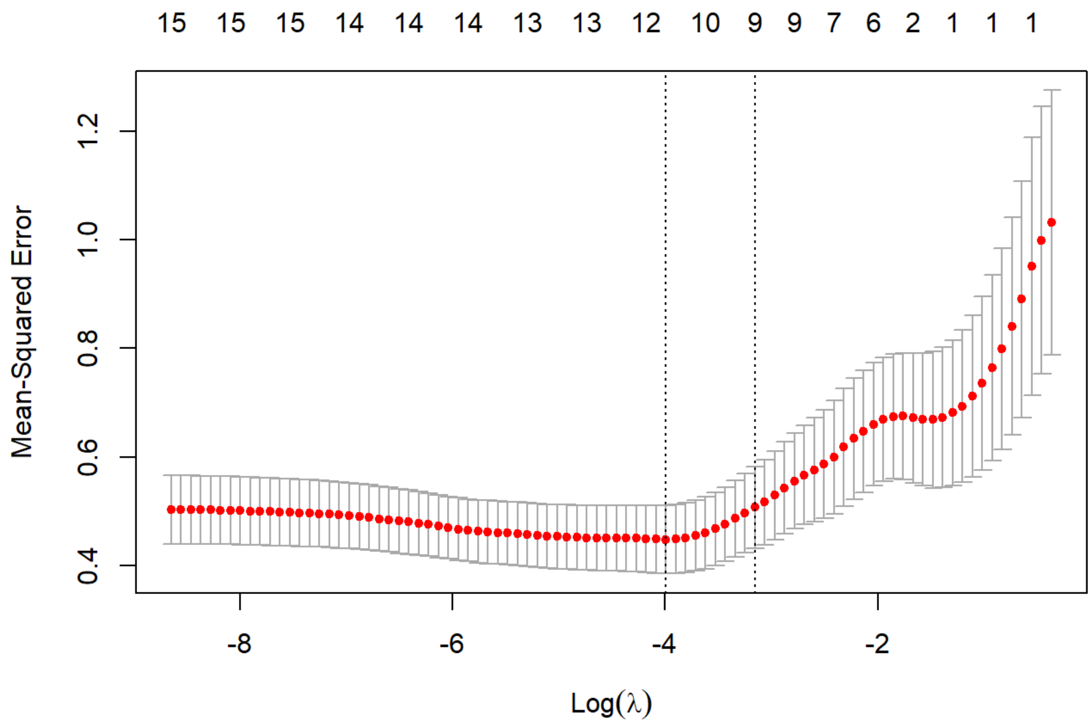
---

Lasso

```
#scale data set
xtrain<-scale(as.matrix(traindata)[,-16], center = TRUE, scale = TRUE)
ytrain<-scale(as.matrix(traindata)[,16], center = TRUE, scale = TRUE)
xtest<-scale(as.matrix(testdata)[,-16], center = TRUE, scale = TRUE)
ytest<-scale(as.matrix(testdata)[,16], center = TRUE, scale = TRUE)
```

- Defining the model

```
lasso_cv <- cv.glmnet(xtrain, ytrain, family="gaussian", alpha=1)
plot(lasso_cv)#plot lasso cv
```




```
coef(lasso_cv)
```

```
## 16 x 1 sparse Matrix of class "dgCMatrix"
##                         1
## (Intercept) -1.831403e-16
## M            1.632944e-01
## So           1.039512e-01
## Ed           1.902793e-01
## Po1          7.972784e-01
## Po2          .
## LF           3.783644e-02
## M.F          1.205711e-01
## Pop          .
## NW           .
## U1           .
## U2           6.527132e-02
## Wealth       .
## Ineq         3.319624e-01
## Prob        -1.857967e-01
## Time         .
```

```
best_lambda <- lasso_cv$lambda.min
cat(best_lambda)
```

```
## 0.01844124
```

- Model using best lambda value

```
lasso_mod = glmnet(xtrain, ytrain, family = "gaussian", alpha = 1, lambda = best_lambda)
coef(lasso_mod)
```

```
## 16 x 1 sparse Matrix of class "dgCMatrix"
##                        s0
## (Intercept) -2.304740e-16
## M            2.248093e-01
## So           9.661256e-02
## Ed           3.637371e-01
## Po1          7.720783e-01
## Po2          .
## LF           2.177071e-02
## M.F          1.563956e-01
## Pop          .
## NW           5.887326e-03
## U1          -1.563660e-01
## U2           2.607918e-01
## Wealth       3.499338e-02
## Ineq         4.674476e-01
## Prob        -2.103825e-01
## Time         .
```

- Prediction & evaluations (LASSO)

```
# Compute R^2 from true and predicted values
eval_results <- function(true, predicted, df) {
  SSE <- sum((predicted - true)^2)
  SST <- sum((true - mean(true))^2)
  R_square <- 1 - SSE / SST
  RMSE = sqrt(SSE/nrow(df))


  # Model performance metrics
data.frame(
  RMSE = RMSE,
  Rsquare = R_square
)

}

# Prediction and evaluation on train data
yhat.train = predict(lasso_mod, xtrain)
eval_results(ytrain, yhat.train, traindata)
```

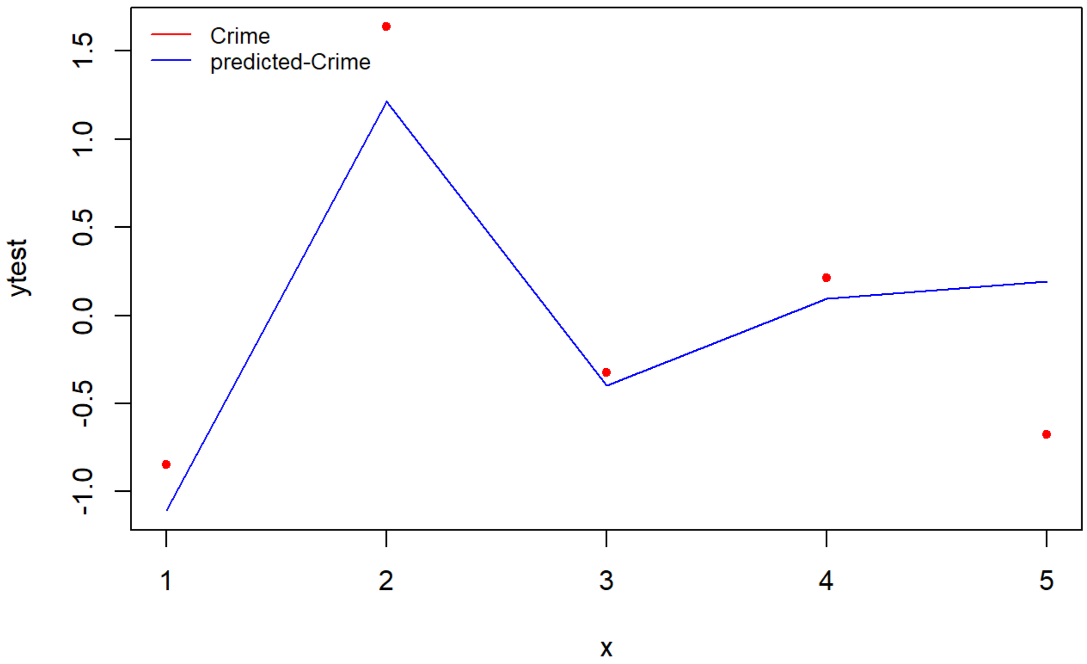
```
##        RMSE   Rsquare
## 1 0.4634677 0.7799586
```

```
# Prediction and evaluation on test data
yhat.test = predict(lasso_mod, xtest)
eval_results(ytest, yhat.test, testdata)
```

```
##        RMSE  Rsquare
## 1 0.4510459 0.745697
```

- Plot of Lasso Prediction

```
x = 1:length(ytest)
plot(x, ytest, ylim=c(min(yhat.test), max(ytest)), pch=20, col="red")
lines(x, yhat.test, lwd="1", col="blue")
legend("topleft", legend=c("Crime", "predicted-Crime"),
       col=c("red", "blue"), lty=1,cex = 0.8, lwd=1, bty='n')
```



Observation2:

- The optimal lambda was 0.02221254 from the plot
- RSME and R squares were fairly similar for both training and testing data sets and very comparable to linear regression
- Because Stepwise was done w/o scaling, the RSME metric cannot be compared with Lasso as its scaled. So we will be using R's as the metric of comparision going forward. As expected Lasso's R-square saw improvement

Elastic Net

```
# Set training control
train_cont <- trainControl(method = "repeatedcv",
                              number = 10,
                              repeats = 5,
                              search = "random",
                              verboseIter = F)


# Train the model
elastic_reg <- train(Crime ~ .,data = as.matrix(scale(traindata)),method = "glmnet",preProcess = c("center",
"scale"),
                           tuneLength = 10,#10 different combinations of values for alpha and lambda are to b
e tested
                           trControl = train_cont)
```

```
## Warning in nominalTrainWorkflow(x = x, y = y, wts = weights, info = trainInfo, :
## There were missing values in resampled performance measures.
```

```
# Best tuning parameter
elastic_reg$bestTune
```

```
##        alpha     lambda
## 1 0.00381962 0.09804711
```

- Predictions & Evaluations (Elastic Net)

```
# Make predictions on training set
predictions_train <- predict(elastic_reg, xtrain)
eval_results(ytrain, predictions_train, as.matrix(traindata))
```

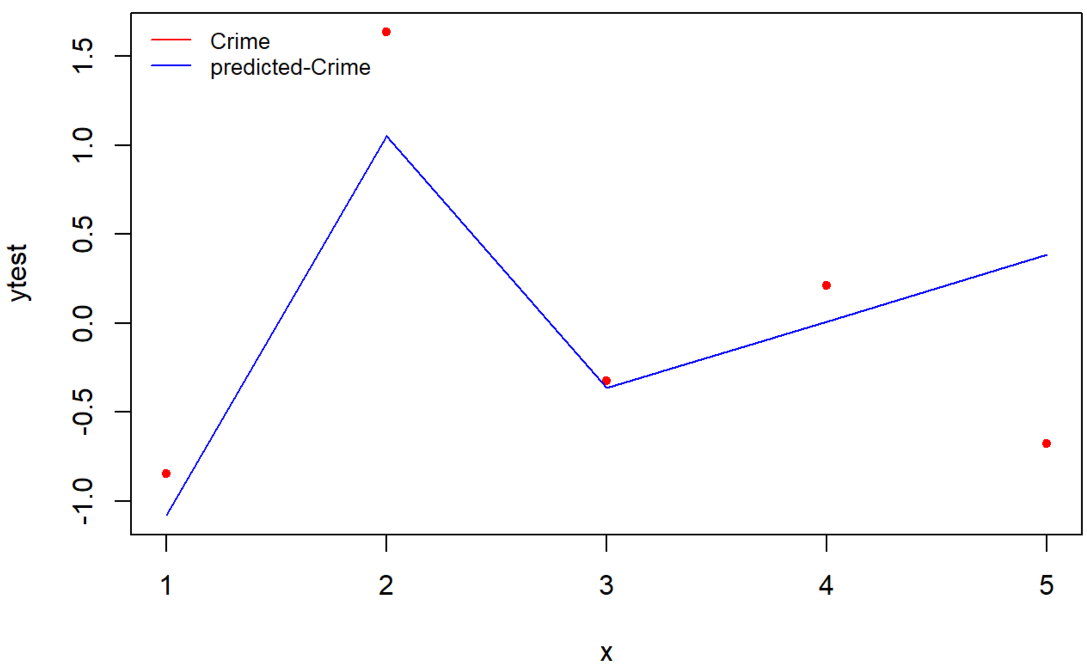
```
##       RMSE  Rsquare
## 1 0.477752 0.766186
```

```
# Make predictions on test set
predictions_test <- predict(elastic_reg, xtest)
eval_results(ytest, predictions_test, as.matrix(testdata))
```

```
##        RMSE   Rsquare
## 1 0.5595539 0.6086243
```

- Plot of Elastic Net Prediction

```
x = 1:length(ytest)
plot(x, ytest, ylim=c(min(predictions_test), max(ytest)), pch=20, col="red")
lines(x, predictions_test, lwd="1", col="blue")
legend("topleft", legend=c("Crime", "predicted-Crime"),
       col=c("red", "blue"), lty=1,cex = 0.8, lwd=1, bty='n')
```



Observation3:

- Since there's no definite alpha for Elastic net, using the argument tuneLength specifies that 10 different combinations of values for alpha and lambda are to be tested
- Based on the above iterations and output, best tuned alpha & best tuned lambda were listed above

Note: Potentially better results (or worst) would've been gotten if I had tried out different tune length

- From a quality perspective, regularized R-square dropped slightly from training to test set like it should
- Overall, all the models performed well with decent R-squared and stable RMSE values. Strangely enough for this data set, witnessed improvements going from traditional Linear Regression to regularization models in terms of R squares

---