ElasticNetAndLasso.R

harit

2020-10-14

rm(list = ls())  
set.seed(82)  
uscrime<- read.table("uscrime.txt", stringsAsFactors = FALSE, header = TRUE)  
  
uscrime[1:3,]

## 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  
## Time Crime  
## 1 26.2011 791  
## 2 25.2999 1635  
## 3 24.3006 578

# Scale the data  
Scaleduscrime <- as.data.frame(scale(uscrime[,c(1,3:15)]))  
Scaleduscrime <- cbind(uscrime[,2],Scaleduscrime,uscrime[,16])  
colnames(Scaleduscrime)[1] <- "So"  
colnames(Scaleduscrime)[16] <- "Crime"  
  
Scaleduscrime[1:3,]

## So M Ed Po1 Po2 LF M.F  
## 1 1 0.9886930 -1.3085099 -0.9085105 -0.8666988 -1.2667456 -1.1206050  
## 2 0 0.3521372 0.6580587 0.6056737 0.5280852 0.5396568 0.9834175  
## 3 1 0.2725678 -1.4872888 -1.3459415 -1.2958632 -0.6976051 -0.4758239  
## Pop NW U1 U2 Wealth Ineq Prob  
## 1 -0.09500679 1.943738564 0.69510600 0.8313680 -1.3616094 1.679364 1.6497631  
## 2 -0.62033844 0.008483424 0.02950365 0.2393332 0.3276683 0.000000 -0.7693365  
## 3 -0.48900552 1.146296747 -0.08143007 -0.1158877 -2.1492481 1.403647 1.5969416  
## Time Crime  
## 1 -0.05599367 791  
## 2 -0.18315796 1635  
## 3 -0.32416470 578

# Split the data into Training and Test Datasets.  
library(caret)

## Loading required package: lattice

## Loading required package: ggplot2

randomrows <- createDataPartition(y=1:nrow(Scaleduscrime),p=0.7, list = FALSE)  
TrainingData = Scaleduscrime[randomrows,]  
TestData = Scaleduscrime[-randomrows,]  
dim(TrainingData)

## [1] 35 16

dim(TestData)

## [1] 12 16

#b. Lasso Regression :   
  
  
library(glmnet)

## Loading required package: Matrix

## Loaded glmnet 4.0-2

set.seed(82)  
  
# The cv part means we want to use Cross validation to   
#obtain the optimalvalues for Lambda.  
model\_lasso <- cv.glmnet(x=as.matrix(TrainingData[,-16]),  
 y = as.matrix(TrainingData[,16]),  
 alpha = 1 ,  
 nfolds = 8,  
 nlambda = 20,  
 type.measure = "mse",  
 family ="gaussian",  
 standardize = TRUE)  
model\_lasso.predicted<-predict(model\_lasso,s=model\_lasso$lambda.1se,newx=as.matrix(TestData[,-16]))  
#Lambda.1se is the value of lambda,that resulted in the simplest model(model with few non zero parameters)  
#and was within 1 standard error of the lambda that had the smallest sum.  
model\_lasso.predicted

## 1  
## 2 895.0571  
## 5 895.0571  
## 8 895.0571  
## 18 895.0571  
## 21 895.0571  
## 22 895.0571  
## 25 895.0571  
## 26 895.0571  
## 27 895.0571  
## 42 895.0571  
## 43 895.0571  
## 45 895.0571

mean((TestData[,16] - model\_lasso.predicted)^2)

## [1] 275313.2

# Find the accuracy   
sse = sum((model\_lasso.predicted - TestData[,16])^2)  
totalSumofSquares = sum((TestData[,16]-mean(TestData[,16]))^2)  
RSquared = 1- (sse/totalSumofSquares)  
RSquared

## [1] -0.005634715

#Elastic.Net Regression   
  
# The cv part means we want to use Cross validation to   
#obtain the optimalvalues for Lambda.  
model\_elasticnet\_alpha0.5 <- cv.glmnet(x=as.matrix(TrainingData[,-16]),  
 y = as.matrix(TrainingData[,16]),  
 alpha = 0.5 ,  
 nfolds = 8,  
 nlambda = 20,  
 type.measure = "mse",  
 family ="gaussian",  
 standardize = TRUE)  
model\_elasticnet\_alpha0.5.predicted<-predict(model\_elasticnet\_alpha0.5,s=model\_elasticnet\_alpha0.5$lambda.1se,newx=as.matrix(TestData[,-16]))  
#Lambda.1se is the value of lambda,that resulted in the simplest model(model with few non zero parameters)  
#and was within 1 standard error of the lambda that had the smallest sum.  
model\_elasticnet\_alpha0.5.predicted

## 1  
## 2 1111.1050  
## 5 971.0996  
## 8 1087.6278  
## 18 691.4622  
## 21 928.0997  
## 22 743.8468  
## 25 568.1519  
## 26 1292.4630  
## 27 631.7364  
## 42 357.4658  
## 43 1015.9012  
## 45 772.1347

# Find the accuracy   
sse = sum((model\_elasticnet\_alpha0.5.predicted - TestData[,16])^2)  
totalSumofSquares = sum((TestData[,16]-mean(TestData[,16]))^2)  
RSquared = 1- (sse/totalSumofSquares)  
AdjustedRSqaured = RSquared - (1-RSquared)\*15/(nrow(TestData)-15-1)  
AdjustedRSqaured

## [1] 2.25108

RSquared

## [1] 0.5450617

#Lets try more values of alpha  
# We create the Elastic.NET fit using the cv.glmnet() function,  
#which takes alpha values from 0.0,0.1,..1.0.  
list\_of\_fits <- list()  
for(i in 0:10)  
{  
 fit.name <- paste0("alpha",i/10)  
 list\_of\_fits[[fit.name]] <- cv.glmnet(x=as.matrix(TrainingData[,-16]),  
 y = as.matrix(TrainingData[,16]),  
 alpha = i/10 ,  
 nfolds = 8,  
 nlambda = 20,  
 type.measure = "mse",  
 family ="gaussian",  
 standardize = FALSE)  
}  
  
results <- data.frame()  
# This for loop will give us the error values for each model from above.  
for(i in 0:10)  
{  
 fit.name <- paste0("alpha",i/10)  
 predicted <- predict(list\_of\_fits[[fit.name]],  
 s=list\_of\_fits[[fit.name]]$lambda.1se,newx=as.matrix(TestData[,-16]))  
   
 # Find the accuracy   
 sse = sum((predicted - TestData[,16])^2)  
 totalSumofSquares = sum((TestData[,16]-mean(TestData[,16]))^2)  
 RSquared = 1- (sse/totalSumofSquares)  
 temp <- data.frame(alpha=i/10, Rsqaured=RSquared, fit.name)  
 results <- rbind(results, temp)  
}  
  
results

## alpha Rsqaured fit.name  
## 1 0.0 0.192802377 alpha0  
## 2 0.1 0.151978925 alpha0.1  
## 3 0.2 -0.005634715 alpha0.2  
## 4 0.3 -0.005634715 alpha0.3  
## 5 0.4 -0.005634715 alpha0.4  
## 6 0.5 0.281999903 alpha0.5  
## 7 0.6 0.151110822 alpha0.6  
## 8 0.7 -0.005634715 alpha0.7  
## 9 0.8 -0.005634715 alpha0.8  
## 10 0.9 0.218213133 alpha0.9  
## 11 1.0 -0.005634715 alpha1