Penalized Regression

```
setwd("C:/Users/akash/Desktop/Spring 2017/Advance R/Exam1")
getwd()

## [1] "C:/Users/akash/Desktop/Spring 2017/Advance R/Exam1"
```

1. Load the dataset into R.

```
mydata <- read.csv("exam1.csv", header=TRUE, sep=",")
```

2. Center and scale numerical predictors.

```
sc <- scale(mydata[sapply(mydata, is.numeric)], center = TRUE, scale = TRUE)
sc <- cbind.data.frame(as.data.frame(sc), as.data.frame(mydata$X15))
head(sc)

##            X0         X1         X2         X3         X4          X5
## 1  1.38162065 -0.5852751  1.3577944 -0.5975150  0.5703694 -0.06469633
## 2 -1.56241660 -0.3580871  0.6811227 -0.4719006  0.7505207  0.14539129
## 3 -0.19884214  0.2703088 -1.4210064  0.8028125  1.1659734 -1.73838751
## 4 -1.16158464 -0.6221013  1.0742198 -1.3521837 -0.7687214 -0.54758265
## 5 -0.05927866 -1.0997133 -0.4569793 -1.6009201 -0.1422798  0.91642468
## 6  0.11180996 -0.4569426 -2.9033014  0.3662817 -0.3441083  1.35518753
##           X6           X7         X8           X9        X11         X12
## 1 -0.2192142  1.332037606  2.0881376 -1.768388062 -0.7795136 -0.38284320
## 2  2.1730191  1.521699533 -1.5106584 -1.155575951 -1.2099636 -1.51853753
## 3  0.9294798 -0.002486363 -0.4190991 -0.602812834 -0.3757958 -0.06513682
## 4  0.2611978 -0.457903838 -0.5847360  0.735045193  0.2404122  0.44265588
## 5  0.6185339 -1.070999417 -0.4020839 -0.004904503 -0.7362294  0.04865450
## 6  0.2487695  0.620745584 -0.9163592 -0.259813570 -1.3366131  0.26361913
##           X13         X14           y mydata$X15
## 1  0.41282067 -1.39453397 -0.86391656       True
## 2  0.30790277  1.13590230 -0.70634478      False
## 3 -0.29228272  0.83857008 -0.34671627       True
## 4  0.33684205 -0.23564971 -0.30349688       True
## 5 -0.79376489  1.04065697 -0.03099074       True
## 6 -0.07521933  0.01437285 -0.91015485       True
```

3. Create dummy variables for any categorical predictors.

```
dX15 <- as.numeric(mydata$X15 == "True")
sc <- cbind.data.frame(as.data.frame(sc), as.data.frame(dX15))
colnames(sc)[colnames(sc)=="mydata$X15"] <- "X15"
head(sc)

##            X0         X1         X2         X3         X4          X5
## 1  1.38162065 -0.5852751  1.3577944 -0.5975150  0.5703694 -0.06469633
## 2 -1.56241660 -0.3580871  0.6811227 -0.4719006  0.7505207  0.14539129
## 3 -0.19884214  0.2703088 -1.4210064  0.8028125  1.1659734 -1.73838751
## 4 -1.16158464 -0.6221013  1.0742198 -1.3521837 -0.7687214 -0.54758265
```

```
## 5 -0.05927866 -1.0997133 -0.4569793 -1.6009201 -0.1422798  0.91642468
## 6  0.11180996 -0.4569426 -2.9033014  0.3662817 -0.3441083  1.35518753
##           X6          X7         X8           X9        X11          X12
## 1 -0.2192142  1.332037606  2.0881376 -1.768388062 -0.7795136 -0.38284320
## 2  2.1730191  1.521699533 -1.5106584 -1.155575951 -1.2099636 -1.51853753
## 3  0.9294798 -0.002486363 -0.4190991 -0.602812834 -0.3757958 -0.06513682
## 4  0.2611978 -0.457903838 -0.5847360  0.735045193  0.2404122  0.44265588
## 5  0.6185339 -1.070999417 -0.4020839 -0.004904503 -0.7362294  0.04865450
## 6  0.2487695  0.620745584 -0.9163592 -0.259813570 -1.3366131  0.26361913
##           X13         X14           y   X15 dX15
## 1  0.41282067 -1.39453397 -0.86391656  True    1
## 2  0.30790277  1.13590230 -0.70634478 False    0
## 3 -0.29228272  0.83857008 -0.34671627  True    1
## 4  0.33684205 -0.23564971 -0.30349688  True    1
## 5 -0.79376489  1.04065697 -0.03099074  True    1
## 6 -0.07521933  0.01437285 -0.91015485  True    1
```

4. Split the data into a training and test set. Set aside the test set until the end.

```
sSize = nrow(sc)*0.8
Splitteddata <- sample(nrow(sc), sSize, replace = TRUE)
trainData<- sc[Splitteddata,]
head(trainData)

##             X0         X1         X2          X3         X4          X5
## 83 -0.76423851  0.6992221  0.3265128  0.55622983  0.3905870  0.54316201
## 16  1.49502859  0.5190137  1.0720188  1.65912488  0.5449716 -1.46994384
## 41  0.08525963  0.1638406 -1.1776309 -0.62726992 -0.6761618 -2.12058610
## 37  1.21015290 -0.3794330 -1.5505619 -0.01258304  2.3321893 -0.04507707
## 99 -0.02925187 -1.3356394  0.5510720 -2.06124922 -0.5388880 -0.25249589
## 60  1.13173292  0.3020160 -0.3272747 -1.02099347  1.3831452  0.50386863
##            X6         X7         X8         X9          X11        X12
## 83 -1.1469889 -0.5906124 -0.1683684 -0.9045194 -2.119760727  1.0747030
## 16  0.7868164 -0.4548442  1.8034383  0.8819827  0.004817771 -2.5548724
## 41  0.5298044 -2.5394467 -0.3159878 -0.6429335 -0.748503677  0.6823144
## 37  0.1164359 -1.2050372  0.9008890 -0.5737284 -0.210632712  0.1792938
## 99  0.5162076 -0.4256776  0.4409269 -1.2038705  1.144273463 -1.1507188
## 60  0.8293151  0.6630730 -1.0024032  2.1987600 -0.793545568  0.4901609
##            X13        X14           y   X15 dX15
## 83  1.45621130  0.1611993 -0.26726921 False    0
## 16  0.50204953 -0.9376641  1.25789906 False    0
## 41  0.03769852 -1.8839302 -2.29269423 False    0
## 37  1.44117502  0.4011789  0.08259715 False    0
## 99 -0.10867593 -0.3334010 -1.79546408 False    0
## 60  0.12252447  0.5419100  2.12590560 False    0

testData<- sc[-Splitteddata,]
head(testData)

##            X0         X1         X2         X3        X4          X5
## 1  1.38162065 -0.5852751  1.3577944 -0.5975150 0.5703694 -0.06469633
```

```
## 2 -1.56241660 -0.3580871  0.6811227 -0.4719006  0.7505207  0.14539129
## 3 -0.19884214  0.2703088 -1.4210064  0.8028125  1.1659734 -1.73838751
## 4 -1.16158464 -0.6221013  1.0742198 -1.3521837 -0.7687214 -0.54758265
## 5 -0.05927866 -1.0997133 -0.4569793 -1.6009201 -0.1422798  0.91642468
## 9 -0.56386157 -0.4238898  0.2707242  0.4166060  1.0058743  0.55756009
##            X6           X7         X8           X9        X11         X12
## 1 -0.21921416  1.332037606  2.0881376 -1.768388062 -0.7795136 -0.38284320
## 2  2.17301906  1.521699533 -1.5106584 -1.155575951 -1.2099636 -1.51853753
## 3  0.92947985 -0.002486363 -0.4190991 -0.602812834 -0.3757958 -0.06513682
## 4  0.26119778 -0.457903838 -0.5847360  0.735045193  0.2404122  0.44265588
## 5  0.61853389 -1.070999417 -0.4020839 -0.004904503 -0.7362294  0.04865450
## 9  0.09952756 -2.460742908 -0.2356271 -1.498561180  2.2626796  0.12586559
##          X13        X14           y   X15 dX15
## 1  0.4128207 -1.3945340 -0.86391656  True    1
## 2  0.3079028  1.1359023 -0.70634478 False    0
## 3 -0.2922827  0.8385701 -0.34671627  True    1
## 4  0.3368420 -0.2356497 -0.30349688  True    1
## 5 -0.7937649  1.0406570 -0.03099074  True    1
## 9 -2.4471108 -0.1358160 -0.76848140  True    1
```

5. Split the training data using 4 fold cross validation.

```
library(caret)

## Warning: package 'caret' was built under R version 3.3.2

## Loading required package: lattice

## Loading required package: ggplot2

trainDataFolds <- createFolds(trainData$X15, k = 4, list = TRUE)
```

6. Fit ridge regression models for a range of "lambda"2 values. Be sure to include large enough values of "lambda"2 that you see a decrease in performance.

7. For each value of "lambda"2, you will have 4 models (1 for each fold). Evaluate the RMSE of all models on the fold not used to train. Use a loop for this.

```
library(AppliedPredictiveModeling)

## Warning: package 'AppliedPredictiveModeling' was built under R version
## 3.3.2

library(penalized)

## Warning: package 'penalized' was built under R version 3.3.2

## Loading required package: survival

##
## Attaching package: 'survival'
```

```
## The following object is masked from 'package:caret':
##
##     cluster

## Welcome to penalized. For extended examples, see vignette("penalized").

library(glmnet)

## Warning: package 'glmnet' was built under R version 3.3.2

## Loading required package: Matrix

## Loading required package: foreach

## Loaded glmnet 2.0-5

library(elasticnet)

## Warning: package 'elasticnet' was built under R version 3.3.2

## Loading required package: lars

## Warning: package 'lars' was built under R version 3.3.2

## Loaded lars 1.2

library(MASS)

## Warning: package 'MASS' was built under R version 3.3.2

rmse <- function(a,b){
  x= (a-b)^2
  y= sqrt(mean(x))
  return(y)
}

  perf.df <- data.frame("q"=numeric(0))
  average.root <- data.frame("t"= numeric(0))
  sequence <- seq(0, 1,0.01 )
  for (i in 1:length(sequence)) {
    for (j in length(trainDataFolds)) {
  crossvalidationTest <- trainData[trainDataFolds[[j]],]
  crossvalidationTrain <- trainData[-trainDataFolds[[j]],]
  crossvalidationTestCopy <- crossvalidationTest
  crossvalidationTest$X15 <- NULL
  crossvalidationTest$y <- NULL
  ridge.reg <- penalized(y ~
X0+X1+X2+X3+X4+X5+X6+X7+X8+X9+X11+X12+X13+X14+dX15, data =
crossvalidationTrain , lambda2 =sequence[[i]], standardize = TRUE)
  predict.p<-predict(ridge.reg,crossvalidationTest)
  rootmean <- rmse(predict.p, crossvalidationTestCopy$y)
  average.root <- rbind(average.root, c(rootmean))
    }
```

```
  final.mean <- lapply(average.root, mean)
  perf.df <- rbind(perf.df, c(final.mean))
  }
```

```
## 12
## 12
## 12
## 12
## 12
## 12
## 12
## 12
## 12
## 12
## 12
## 12
## 12
## 12
## 12
## 12
## 12
## 12
## 12
## 12
## 12
## 12
## 12
## 12
## 12
## 12
## 12
## 12
## 12
## 12
## 12
## 12
## 12
## 12
## 12
## 12
## 12
## 12
## 12
## 12
## 12
## 12
## 12
## 12
## 12
## 12
```

```
## 12
## 12
## 12
## 12
## 12
## 12
## 12
## 12
## 12
## 12
## 12
## 12
## 12
## 12
## 12
## 12
## 12
## 12
## 12
## 12
## 12
## 12
## 12
## 12
## 12
## 12
## 12
## 12
## 12
## 12
## 12
## 12
## 12
## 12
## 12
## 12
## 12
## 12
## 12
## 12
## 12
## 12
## 12
## 12
## 12
## 12
## 12
## 12
## 12
## 12
```

```
## 12
## 12
## 12
## 12
## 12
```

```
perf.df
```

```
##     X0.781123932381918
## 1            0.7811239
## 2            0.7811233
## 3            0.7811227
## 4            0.7811221
## 5            0.7811216
## 6            0.7811210
## 7            0.7811205
## 8            0.7811200
## 9            0.7811195
## 10           0.7811191
## 11           0.7811186
## 12           0.7811182
## 13           0.7811178
## 14           0.7811174
## 15           0.7811170
## 16           0.7811167
## 17           0.7811164
## 18           0.7811161
## 19           0.7811158
## 20           0.7811155
## 21           0.7811152
## 22           0.7811150
## 23           0.7811148
## 24           0.7811146
## 25           0.7811144
## 26           0.7811142
## 27           0.7811141
## 28           0.7811140
## 29           0.7811139
## 30           0.7811138
## 31           0.7811137
## 32           0.7811136
## 33           0.7811136
## 34           0.7811136
## 35           0.7811136
## 36           0.7811136
## 37           0.7811137
## 38           0.7811137
## 39           0.7811138
## 40           0.7811139
## 41           0.7811140
```

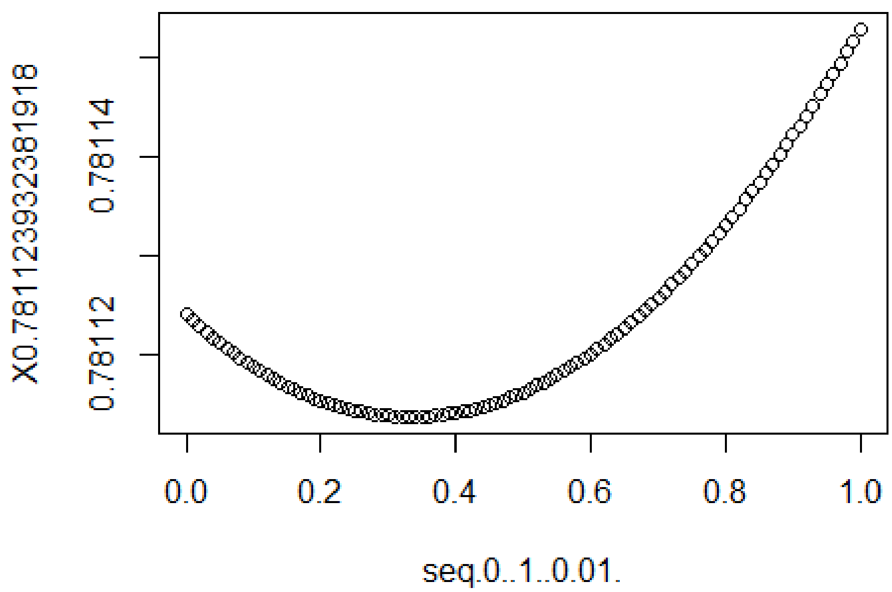
```
## 42             0.7811141
## 43             0.7811143
## 44             0.7811144
## 45             0.7811146
## 46             0.7811148
## 47             0.7811151
## 48             0.7811153
## 49             0.7811155
## 50             0.7811158
## 51             0.7811161
## 52             0.7811164
## 53             0.7811167
## 54             0.7811171
## 55             0.7811175
## 56             0.7811178
## 57             0.7811182
## 58             0.7811187
## 59             0.7811191
## 60             0.7811195
## 61             0.7811200
## 62             0.7811205
## 63             0.7811210
## 64             0.7811215
## 65             0.7811221
## 66             0.7811226
## 67             0.7811232
## 68             0.7811238
## 69             0.7811244
## 70             0.7811250
## 71             0.7811256
## 72             0.7811263
## 73             0.7811270
## 74             0.7811277
## 75             0.7811284
## 76             0.7811291
## 77             0.7811299
## 78             0.7811306
## 79             0.7811314
## 80             0.7811322
## 81             0.7811330
## 82             0.7811339
## 83             0.7811347
## 84             0.7811356
## 85             0.7811364
## 86             0.7811373
## 87             0.7811383
## 88             0.7811392
## 89             0.7811401
## 90             0.7811411
## 91             0.7811421
```

```
## 92           0.7811431
## 93           0.7811441
## 94           0.7811451
## 95           0.7811462
## 96           0.7811473
## 97           0.7811483
## 98           0.7811494
## 99           0.7811506
## 100          0.7811517
## 101          0.7811528
```

8. Make a plot with "lambda"2 on the x-axis and the mean RMSE (average over the 4 folds) on the y-axis.

```r
rmse.plot <- data.frame(seq(0, 1, 0.01), perf.df)

plot(rmse.plot)
```



9. Using this plot, select "lambda"2 for your model. Explain your reasoning. From the plot it can be observed that the value ofroot mean square error first increases as lambda increases and then decreases. It is minimum near to 0.4.Since we want RMSE to be minimized to get a good model,So, the value of lambda2 can be taken as 0.4.

10. Fit a model on the complete training data using your selected value for "lambda"2.

```r
Final.regression <- penalized(y ~
X0+X1+X2+X3+X4+X5+X6+X7+X8+X9+X11+X12+X13+X14+dX15, data = trainData ,
lambda2 = 0.4, standardize = TRUE)
```

```
## 12
```

```
Final.regression
```

```
## Penalized linear regression object
## 16 regression coefficients
##
## Loglikelihood =   168.951
## L2 penalty =  0.2302737  at lambda2 =  0.4
```

11. Evaluate the R2 and RMSE of your model on the test set.

```
testDataCopy <- testData
testData$X15 <- NULL
testData$y <- NULL
Final.predict <- predict(Final.regression,testData)
Final.rmse <- rmse(Final.predict, testDataCopy$y)
Final.rmse
```

```
## [1] 0.6780158
```

```
mean.testdata <- lapply(testDataCopy$y, mean)
SumofSquares.Total = sum((testDataCopy$y - mean(testDataCopy$y))^2)
sumofSquares.Errors = sum((testDataCopy$y - Final.predict[ ,"mu"])^2)
Rsquared = 1-(sumofSquares.Errors/SumofSquares.Total)
Rsquared
```

```
## [1] 0.9962385
```