Spam_Email_.R

Furong Bai, Shuangqiu Liu, Danlu Pan

Fri Mar 17 23:47:47 2017

```
#########################################################################
######################## Input Data Set Into R ##########################
#########################################################################

# Uncommet to install packages
# install.packages("corrplot")
# install.packages("kernlab")
# if using Mac, uncomment following line to install package
# install.packages("doMC")
# if using Microsoft, uncommet following line to install package
# install.packages("doMC", repos="http://R-Forge.R-project.org")
library(corrplot)

## Warning: package 'corrplot' was built under R version 3.3.3

library(caret)

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

## Loading required package: lattice

## Loading required package: ggplot2

## Warning: package 'ggplot2' was built under R version 3.3.2

library(dplyr)

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

##
## Attaching package: 'dplyr'

## The following objects are masked from 'package:stats':
##
##     filter, lag

## The following objects are masked from 'package:base':
##
##     intersect, setdiff, setequal, union

library(ggplot2)
require(kernlab)

## Loading required package: kernlab
```

```
## Warning: package 'kernlab' was built under R version 3.3.2

##
## Attaching package: 'kernlab'

## The following object is masked from 'package:ggplot2':
##
##     alpha

require(doMC)

## Loading required package: doMC

## Loading required package: foreach

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

## Loading required package: iterators

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

## Loading required package: parallel

# input the data set
spamD <- read.table('http://archive.ics.uci.edu/ml/machine-learning-dat
abases/spambase/spambase.data',sep=',',header=F)
colnames(spamD) <- c(
  'word.freq.make', 'word.freq.address', 'word.freq.all',
  'word.freq.3d', 'word.freq.our', 'word.freq.over', 'word.freq.remove'
,
  'word.freq.internet', 'word.freq.order', 'word.freq.mail',
  'word.freq.receive', 'word.freq.will', 'word.freq.people',
  'word.freq.report', 'word.freq.addresses', 'word.freq.free',
  'word.freq.business', 'word.freq.email', 'word.freq.you',
  'word.freq.credit', 'word.freq.your', 'word.freq.font',
  'word.freq.000', 'word.freq.money', 'word.freq.hp', 'word.freq.hpl',
  'word.freq.george', 'word.freq.650', 'word.freq.lab',
  'word.freq.labs', 'word.freq.telnet', 'word.freq.857',
  'word.freq.data', 'word.freq.415', 'word.freq.85',
  'word.freq.technology', 'word.freq.1999', 'word.freq.parts',
  'word.freq.pm', 'word.freq.direct', 'word.freq.cs',
  'word.freq.meeting', 'word.freq.original', 'word.freq.project',
  'word.freq.re', 'word.freq.edu', 'word.freq.table',
  'word.freq.conference', 'char.freq.semi', 'char.freq.lparen',
  'char.freq.lbrack', 'char.freq.bang', 'char.freq.dollar',
  'char.freq.hash', 'capital.run.length.average',
  'capital.run.length.longest', 'capital.run.length.total',
  'spam'
)
View(spamD)
str(spamD)
```

```
## 'data.frame':    4601 obs. of  58 variables:
##  $ word.freq.make            : num  0 0.21 0.06 0 0 0 0 0 0.15 0.06
...
##  $ word.freq.address         : num  0.64 0.28 0 0 0 0 0 0 0 0.12 ...
##  $ word.freq.all             : num  0.64 0.5 0.71 0 0 0 0 0 0.46 0.7
7 ...
##  $ word.freq.3d              : num  0 0 0 0 0 0 0 0 0 0 ...
##  $ word.freq.our             : num  0.32 0.14 1.23 0.63 0.63 1.85 1.
92 1.88 0.61 0.19 ...
##  $ word.freq.over            : num  0 0.28 0.19 0 0 0 0 0 0 0.32 ...
##  $ word.freq.remove          : num  0 0.21 0.19 0.31 0.31 0 0 0 0.3
0.38 ...
##  $ word.freq.internet        : num  0 0.07 0.12 0.63 0.63 1.85 0 1.8
8 0 0 ...
##  $ word.freq.order           : num  0 0 0.64 0.31 0.31 0 0 0 0.92 0.
06 ...
##  $ word.freq.mail            : num  0 0.94 0.25 0.63 0.63 0 0.64 0 0
.76 0 ...
##  $ word.freq.receive         : num  0 0.21 0.38 0.31 0.31 0 0.96 0 0
.76 0 ...
##  $ word.freq.will            : num  0.64 0.79 0.45 0.31 0.31 0 1.28
0 0.92 0.64 ...
##  $ word.freq.people          : num  0 0.65 0.12 0.31 0.31 0 0 0 0 0.
25 ...
##  $ word.freq.report          : num  0 0.21 0 0 0 0 0 0 0 0 ...
##  $ word.freq.addresses       : num  0 0.14 1.75 0 0 0 0 0 0 0.12 ...
##  $ word.freq.free            : num  0.32 0.14 0.06 0.31 0.31 0 0.96
0 0 0 ...
##  $ word.freq.business        : num  0 0.07 0.06 0 0 0 0 0 0 0 ...
##  $ word.freq.email           : num  1.29 0.28 1.03 0 0 0 0.32 0 0.15
 0.12 ...
##  $ word.freq.you             : num  1.93 3.47 1.36 3.18 3.18 0 3.85
0 1.23 1.67 ...
##  $ word.freq.credit          : num  0 0 0.32 0 0 0 0 0 3.53 0.06 ...
##  $ word.freq.your            : num  0.96 1.59 0.51 0.31 0.31 0 0.64
0 2 0.71 ...
##  $ word.freq.font            : num  0 0 0 0 0 0 0 0 0 0 ...
##  $ word.freq.000             : num  0 0.43 1.16 0 0 0 0 0 0 0.19 ...
##  $ word.freq.money           : num  0 0.43 0.06 0 0 0 0 0 0.15 0 ...
##  $ word.freq.hp              : num  0 0 0 0 0 0 0 0 0 0 ...
##  $ word.freq.hpl             : num  0 0 0 0 0 0 0 0 0 0 ...
##  $ word.freq.george          : num  0 0 0 0 0 0 0 0 0 0 ...
##  $ word.freq.650             : num  0 0 0 0 0 0 0 0 0 0 ...
##  $ word.freq.lab             : num  0 0 0 0 0 0 0 0 0 0 ...
##  $ word.freq.labs            : num  0 0 0 0 0 0 0 0 0 0 ...
##  $ word.freq.telnet          : num  0 0 0 0 0 0 0 0 0 0 ...
##  $ word.freq.857             : num  0 0 0 0 0 0 0 0 0 0 ...
##  $ word.freq.data            : num  0 0 0 0 0 0 0 0 0.15 0 ...
##  $ word.freq.415             : num  0 0 0 0 0 0 0 0 0 0 ...
##  $ word.freq.85              : num  0 0 0 0 0 0 0 0 0 0 ...
```

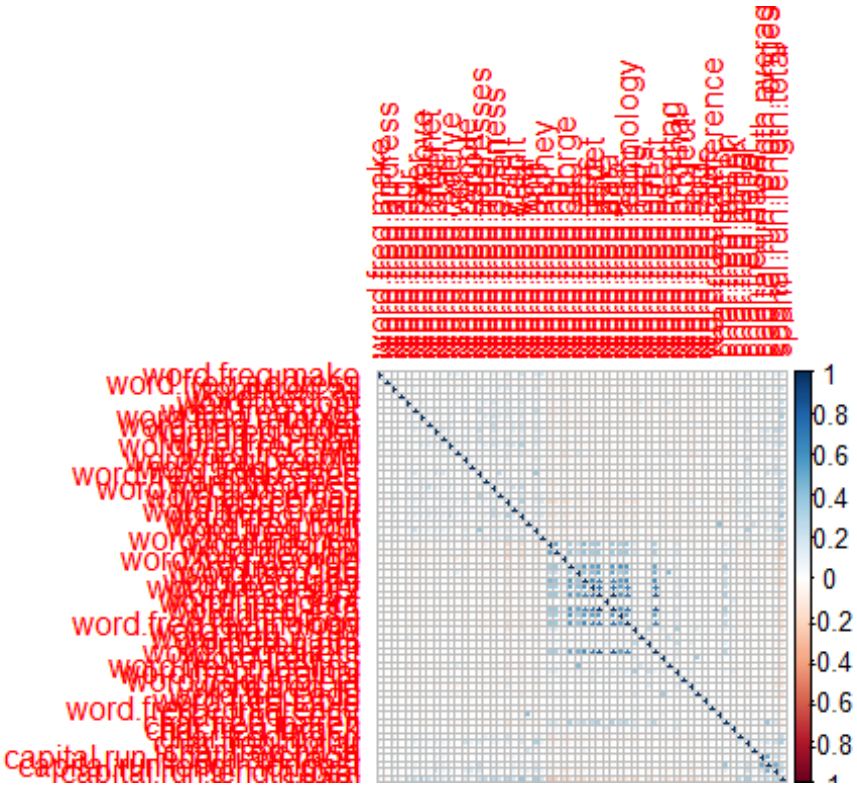
```
##  $ word.freq.technology      : num  0 0 0 0 0 0 0 0 0 0 ...
##  $ word.freq.1999            : num  0 0.07 0 0 0 0 0 0 0 0 ...
##  $ word.freq.parts           : num  0 0 0 0 0 0 0 0 0 0 ...
##  $ word.freq.pm              : num  0 0 0 0 0 0 0 0 0 0 ...
##  $ word.freq.direct          : num  0 0 0.06 0 0 0 0 0 0 0 ...
##  $ word.freq.cs              : num  0 0 0 0 0 0 0 0 0 0 ...
##  $ word.freq.meeting         : num  0 0 0 0 0 0 0 0 0 0 ...
##  $ word.freq.original        : num  0 0 0.12 0 0 0 0 0 0.3 0 ...
##  $ word.freq.project         : num  0 0 0 0 0 0 0 0 0 0.06 ...
##  $ word.freq.re              : num  0 0 0.06 0 0 0 0 0 0 0 ...
##  $ word.freq.edu             : num  0 0 0.06 0 0 0 0 0 0 0 ...
##  $ word.freq.table           : num  0 0 0 0 0 0 0 0 0 0 ...
##  $ word.freq.conference      : num  0 0 0 0 0 0 0 0 0 0 ...
##  $ char.freq.semi            : num  0 0 0.01 0 0 0 0 0 0 0.04 ...
##  $ char.freq.lparen          : num  0 0.132 0.143 0.137 0.135 0.223
0.054 0.206 0.271 0.03 ...
##  $ char.freq.lbrack          : num  0 0 0 0 0 0 0 0 0 0 ...
##  $ char.freq.bang            : num  0.778 0.372 0.276 0.137 0.135 0
0.164 0 0.181 0.244 ...
##  $ char.freq.dollar          : num  0 0.18 0.184 0 0 0 0.054 0 0.203
 0.081 ...
##  $ char.freq.hash            : num  0 0.048 0.01 0 0 0 0 0 0.022 0 .
..
##  $ capital.run.length.average: num  3.76 5.11 9.82 3.54 3.54 ...
##  $ capital.run.length.longest: int  61 101 485 40 40 15 4 11 445 43
...
##  $ capital.run.length.total  : int  278 1028 2259 191 191 54 112 49
1257 749 ...
##  $ spam                      : int  1 1 1 1 1 1 1 1 1 1 ...

################################################################################
############################## Data Cleaning ###################################
################################################################################

#Check for missing values
sapply(spamD, function(x) sum(is.na(x)))

##             word.freq.make          word.freq.address
##                          0                          0
##              word.freq.all               word.freq.3d
##                          0                          0
##              word.freq.our             word.freq.over
##                          0                          0
##           word.freq.remove         word.freq.internet
##                          0                          0
##            word.freq.order             word.freq.mail
##                          0                          0
##          word.freq.receive             word.freq.will
##                          0                          0
##           word.freq.people           word.freq.report
```

```
##                           0                           0
##        word.freq.addresses             word.freq.free
##                           0                           0
##         word.freq.business            word.freq.email
##                           0                           0
##              word.freq.you           word.freq.credit
##                           0                           0
##             word.freq.your             word.freq.font
##                           0                           0
##             word.freq.000            word.freq.money
##                           0                           0
##              word.freq.hp              word.freq.hpl
##                           0                           0
##           word.freq.george              word.freq.650
##                           0                           0
##              word.freq.lab             word.freq.labs
##                           0                           0
##           word.freq.telnet              word.freq.857
##                           0                           0
##             word.freq.data              word.freq.415
##                           0                           0
##              word.freq.85       word.freq.technology
##                           0                           0
##             word.freq.1999            word.freq.parts
##                           0                           0
##              word.freq.pm           word.freq.direct
##                           0                           0
##              word.freq.cs          word.freq.meeting
##                           0                           0
##         word.freq.original          word.freq.project
##                           0                           0
##              word.freq.re              word.freq.edu
##                           0                           0
##            word.freq.table       word.freq.conference
##                           0                           0
##             char.freq.semi           char.freq.lparen
##                           0                           0
##           char.freq.lbrack             char.freq.bang
##                           0                           0
##           char.freq.dollar             char.freq.hash
##                           0                           0
## capital.run.length.average capital.run.length.longest
##                           0                           0
##   capital.run.length.total                        spam
##                           0                           0

#Check the class of each var
sapply(spamD, function(x) class(x))
```

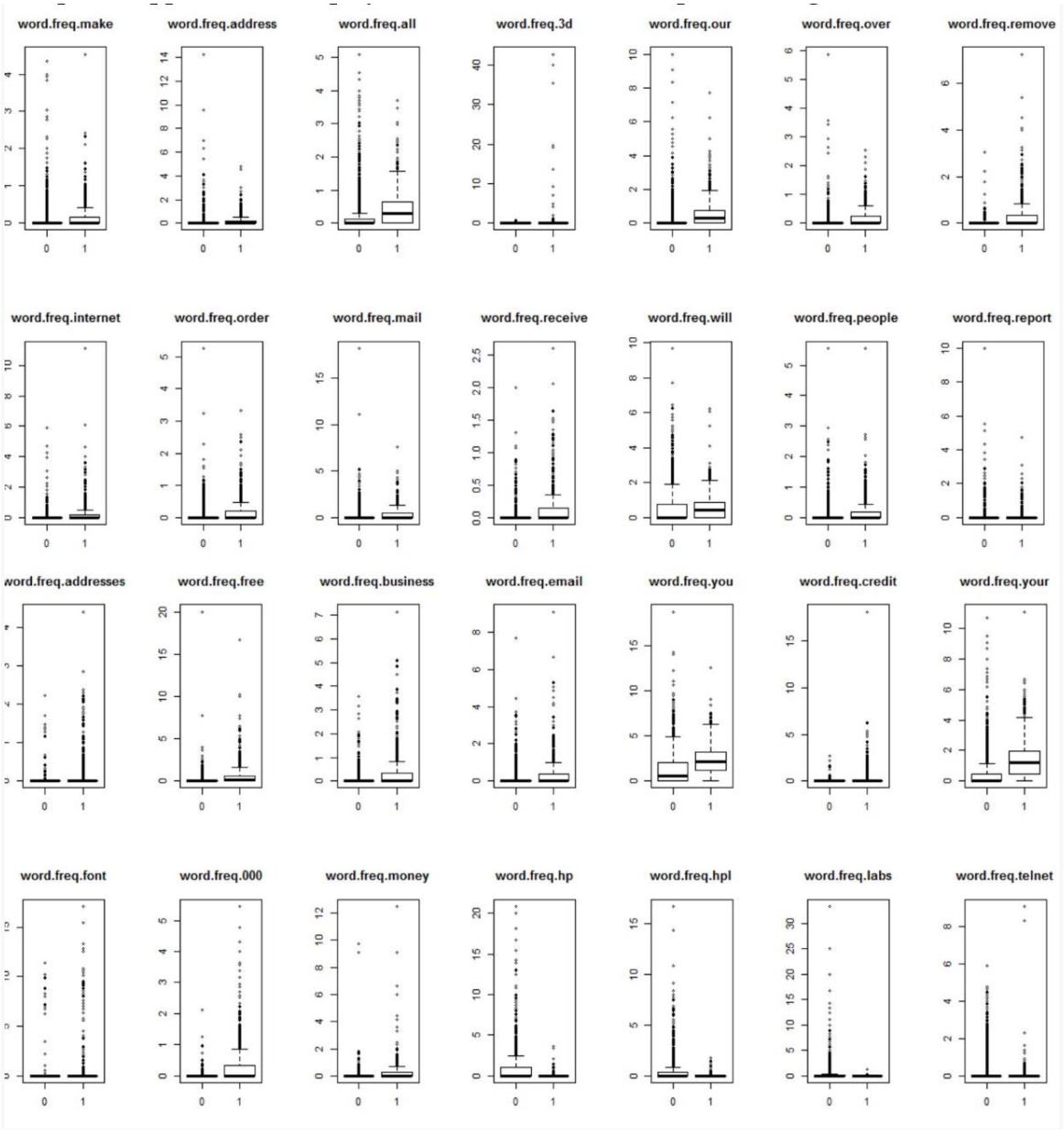
```
##             word.freq.make          word.freq.address
##                  "numeric"                  "numeric"
##              word.freq.all              word.freq.3d
##                  "numeric"                  "numeric"
##              word.freq.our             word.freq.over
##                  "numeric"                  "numeric"
##           word.freq.remove         word.freq.internet
##                  "numeric"                  "numeric"
##            word.freq.order             word.freq.mail
##                  "numeric"                  "numeric"
##           word.freq.receive             word.freq.will
##                  "numeric"                  "numeric"
##           word.freq.people           word.freq.report
##                  "numeric"                  "numeric"
##        word.freq.addresses             word.freq.free
##                  "numeric"                  "numeric"
##         word.freq.business            word.freq.email
##                  "numeric"                  "numeric"
##              word.freq.you           word.freq.credit
##                  "numeric"                  "numeric"
##             word.freq.your             word.freq.font
##                  "numeric"                  "numeric"
##              word.freq.000            word.freq.money
##                  "numeric"                  "numeric"
##               word.freq.hp              word.freq.hpl
##                  "numeric"                  "numeric"
##           word.freq.george             word.freq.650
##                  "numeric"                  "numeric"
##              word.freq.lab             word.freq.labs
##                  "numeric"                  "numeric"
##           word.freq.telnet             word.freq.857
##                  "numeric"                  "numeric"
##             word.freq.data             word.freq.415
##                  "numeric"                  "numeric"
##               word.freq.85       word.freq.technology
##                  "numeric"                  "numeric"
##             word.freq.1999             word.freq.parts
##                  "numeric"                  "numeric"
##               word.freq.pm            word.freq.direct
##                  "numeric"                  "numeric"
##               word.freq.cs           word.freq.meeting
##                  "numeric"                  "numeric"
##         word.freq.original           word.freq.project
##                  "numeric"                  "numeric"
##               word.freq.re               word.freq.edu
##                  "numeric"                  "numeric"
##            word.freq.table       word.freq.conference
##                  "numeric"                  "numeric"
##              char.freq.semi            char.freq.lparen
##                  "numeric"                  "numeric"
```

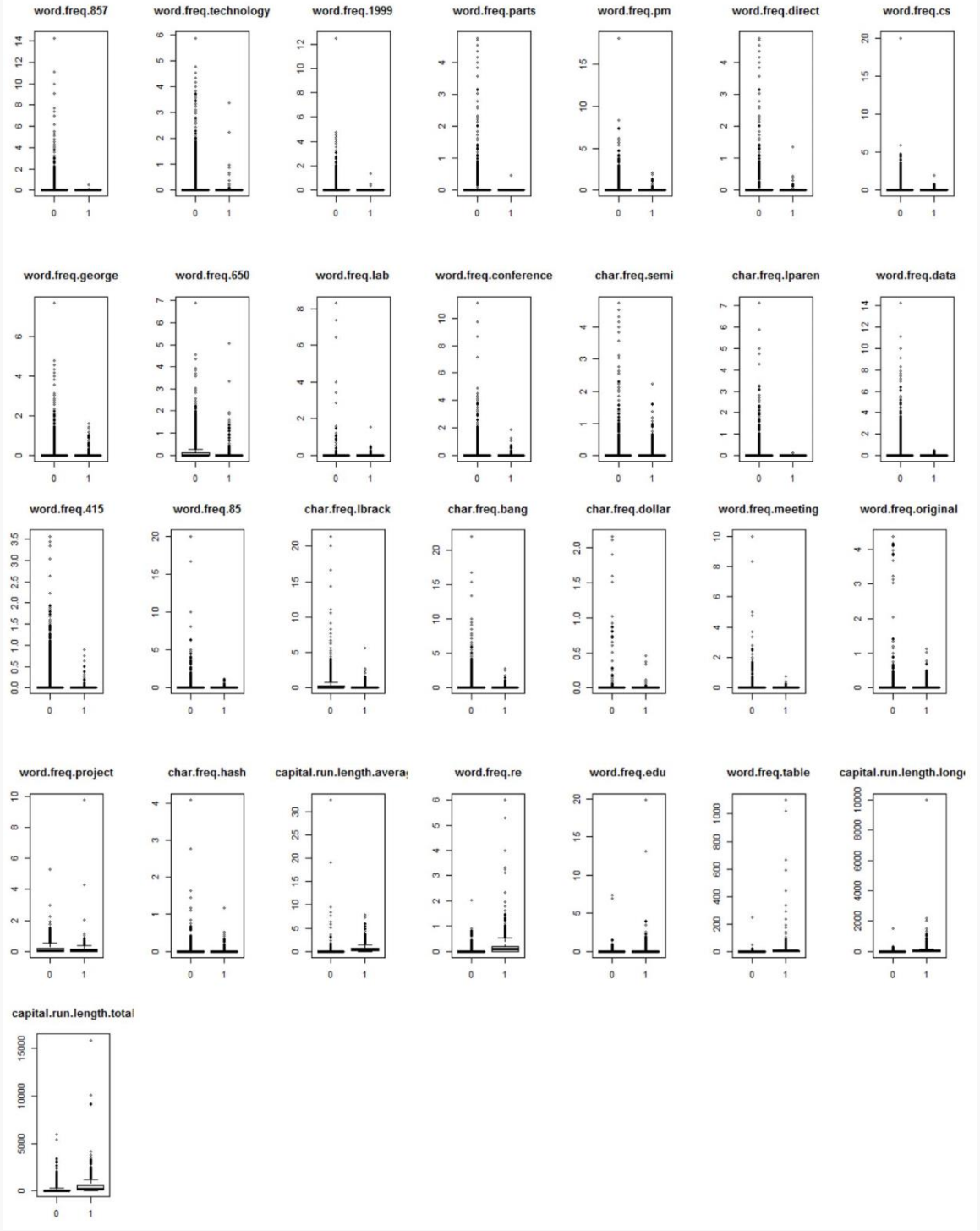
```
##           char.freq.lbrack             char.freq.bang
##                  "numeric"                  "numeric"
##           char.freq.dollar             char.freq.hash
##                  "numeric"                  "numeric"
## capital.run.length.average capital.run.length.longest
##                  "numeric"                  "integer"
##   capital.run.length.total                       spam
##                  "integer"                  "integer"

#Create my.summary function for numerical variables
my.summary <- function(input_df){
  summary_df <-
    data.frame(quantile_.01 = sapply(input_df, function(x) quantile(x,.
01)),
               quantile_.05 = sapply(input_df, function(x) quantile(x,.
05)),
               quantile_.1 = sapply(input_df, function(x) quantile(x,.1
)),
               quantile_.25 = sapply(input_df, function(x) quantile(x,.
25)),
               quantile_.5 = sapply(input_df, function(x) quantile(x,.5
)),
               quantile_.75 = sapply(input_df, function(x) quantile(x,.
75)),
               quantile_.95 = sapply(input_df, function(x) quantile(x,.
95)),
               quantile_.99 = sapply(input_df, function(x) quantile(x,.
99)),
               mean = sapply(input_df, mean),
               variance = sapply(input_df, var),
               min = sapply(input_df, min),
               max = sapply(input_df, max),
               percMissing = sapply(input_df, function(x) sum(is.na(x))
)/nrow(input_df)
    )
}

################################################################################
############################## Data Exploration ################################
################################################################################


#find all numeric variables
isnumeric <-sapply(spamD, function(x) is.numeric(x))

#run my.summary on all of the numeric varaibles
summary_numeric <- my.summary(spamD[,isnumeric])

write.csv(summary_numeric, file="summary_numeric.csv", row.names=TRUE,
quote = FALSE)
```

```r
#create correlation plot
cor_spamD<-cor(spamD[,isnumeric])
corrplot(cor_spamD, method="circle")
```




```r
#Boxplot for all numeric variables against spam_bool
par(mfrow=c(2, 7))

sapply(seq_along(spamD[,isnumeric]), function(i) {
  x <- spamD[,isnumeric][, i]
  boxplot(x ~ spamD$spam,
          main=names(spamD[isnumeric])[i]
  )})
```


word.freq.make
word.freq.address
word.freq.all
word.freq.3d
word.freq.our
word.freq.over
word.freq.remove
word.freq.internet
word.freq.order
word.freq.mail
word.freq.receive
word.freq.will
word.freq.people
word.freq.report
word.freq.addresses
word.freq.free
word.freq.business
word.freq.email
word.freq.you
word.freq.credit
word.freq.your
word.freq.font
word.freq.000
word.freq.money
word.freq.hp
word.freq.hpl
word.freq.labs
word.freq.telnet




```
################################################################################
############################ Data Prep (Question 2) ############################
################################################################################
```

```
###### Splitting to training & testing ########
set.seed(20)
dt = sort(sample(nrow(spamD), nrow(spamD)*0.8))
training1 <- spamD[dt,]
testing1 <- spamD[-dt,]

########### Standardization#################
values <- preProcess(training1[,1:57], method = c("center","scale"))
training2 <-predict(values, training1[ ,1:57])
training <- mutate(training2, spam=training1[,58])

testing2 <- predict(values, testing1[ ,1:57])
testing <- mutate(testing2, spam=testing1[,58])



################################################################################
############################### Adaline ########################################
################################################################################
adalineGD <- function(X, y, n.iter, eta) {

  # extend input vector and initialize extended weight
  X[, dim(X)[2] + 1] <- 1
  X <- as.matrix(X)
  w <- as.matrix(rep(0, dim(X)[2]))

  # initialize cost values - gets updated according to epochnums - numb
er of epochs
  cost <- rep(0, n.iter)
  errors <- rep(0, n.iter)
  TN = rep(0, n.iter)
  FN = rep(0, n.iter)
  FP = rep(0, n.iter)
  TP = rep(0, n.iter)
  Accuracy <- rep(0, n.iter)
  Error <- rep(0, n.iter)
  Precision <- rep(0, n.iter)
  Recall <- rep(0, n.iter)
  FPR <- rep(0, n.iter)

  # loop over the number of epochs
  for (i in 1:n.iter) {

    # find the number of wrong prediction before weight update
    for (j in 1:dim(X)[1]) {

      # compute net input
      z <- sum(w * X[j, ])
```

```
      # quantizer
      if (z < 0) {
        ypred <- -1
      }else {
        ypred <- 1
      }

      # comparison with actual values and counting error
      if(ypred != y[j]) {
        errors[i] <- errors[i] + 1
      }

      # metrices
      if(ypred==-1 && y[j]==-1){TN[i] = TN[i]+1}
      if(ypred==-1 && y[j]== 1){FN[i] = FN[i]+1}
      if(ypred== 1 && y[j]==-1){FP[i] = FP[i]+1}
      if(ypred== 1 && y[j]==1){TP[i] = TP[i]+1}

      Accuracy[i] <- (TP[i]+TN[i])/(FP[i]+FN[i]+TP[i]+TN[i])
      Error[i] <- 1-Accuracy[i]
      Precision[i] <- TP[i]/(TP[i]+FP[i])
      Recall[i] <- TP[i]/(TP[i]+FN[i])
      FPR[i] <- FP[i]/(FP[i]+TN[i])
    }


    # update cost function (SSE)
    cost[i] <- sum((y - X %*% w)^2)/2

    # update weight according to gradient descent
    p = t(X) %*% (y - X %*% w)
    w <- w + eta* p
  }

  # data frame consisting of cost and error info
  confusion_matrix <- matrix(c(TN[n.iter],FN[n.iter],FP[n.iter],TP[n.it
er]),nrow = 2)
  colnames(confusion_matrix) <- c("Negative","Positive")
  rownames(confusion_matrix) <- c("Negative","Positive")

  infomatrix <- matrix(rep(0, 8 * n.iter), nrow = n.iter, ncol = 8)
  infomatrix[, 1] <- 1:n.iter
  infomatrix[, 2] <- log(cost)
  infomatrix[, 3] <- errors
  infomatrix[, 4] <- Accuracy
  infomatrix[, 5] <- Error
  infomatrix[, 6] <- Precision
  infomatrix[, 7] <- Recall
  infomatrix[, 8] <- FPR
```

```
  infodf <- as.data.frame(infomatrix)
  names(infodf) <- c("No_iteration", "cost_function", "errors","Accurac
y","Error","Precision","Recall","FPR")

  infolist <- list(w,infodf,confusion_matrix)
  names(infolist) <- c("w","infomatrix","confusion_matrix")

  return(infolist)
}

predict.adaline <- function(w, X, y){
  # extend input vector and initialize extended weight
  X[, dim(X)[2] + 1] <- 1
  X <- as.matrix(X)


  # initialize metrics
  errors <- 0
  TN = 0
  FN = 0
  FP = 0
  TP = 0
  Accuracy <- 0
  Error <- 0
  Precision <- 0
  Recall <- 0
  FPR <- 0

  # find the number of wrong prediction
  for (j in 1:dim(X)[1]) {

    # compute net input
    z <- sum(w * X[j, ])

    # quantizer
    if (z < 0) {
      ypred <- -1
    }else {
      ypred <- 1
    }

    # comparison with actual values and counting error
    if(ypred != y[j]) {
      errors <- errors + 1
    }

    # metrices
    if(ypred==-1 && y[j]==-1){TN = TN+1}
```

```
    if(ypred==-1 && y[j]== 1){FN = FN+1}
    if(ypred== 1 && y[j]==-1){FP = FP+1}
    if(ypred== 1 && y[j]==1){TP = TP+1}

    Accuracy <- (TP+TN)/(FP+FN+TP+TN)
    Error <- 1-Accuracy
    Precision <- TP/(TP+FP)
    Recall <- TP/(TP+FN)
    FPR <- FP/(FP+TN)
  }

  # data frame consisting of cost and error info
  confusion_matrix <- matrix(c(TN,FN,FP,TP),nrow = 2)
  colnames(confusion_matrix) <- c("Negative","Positive")
  rownames(confusion_matrix) <- c("Negative","Positive")

  infomatrix <- matrix(rep(0, 7), nrow = 1, ncol = 7)
  infomatrix[, 1] <- "Matrics"
  infomatrix[, 2] <- errors
  infomatrix[, 3] <- Accuracy
  infomatrix[, 4] <- Error
  infomatrix[, 5] <- Precision
  infomatrix[, 6] <- Recall
  infomatrix[, 7] <- FPR

  infodf <- as.data.frame(infomatrix)
  names(infodf) <- c("Matrics", "errors","Accuracy","Error","Precision"
,"Recall","FPR")

  infolist <- list(infodf,confusion_matrix)
  names(infolist) <- c("infomatrix","confusion_matrix")

  return(infolist)

}

########## Apply to the training set. #############
y1 <- rep(1, nrow(training))
y1[training[,58]==0] <- -1
result.adalineGD <- adalineGD(training[,-58], y1 ,n.iter = 100, eta = 0
.00001)
# check the weights
result.adalineGD$w

##                                [,1]
## word.freq.make            -0.016092013
## word.freq.address         -0.022453967
## word.freq.all              0.031395917
## word.freq.3d               0.037159250
```

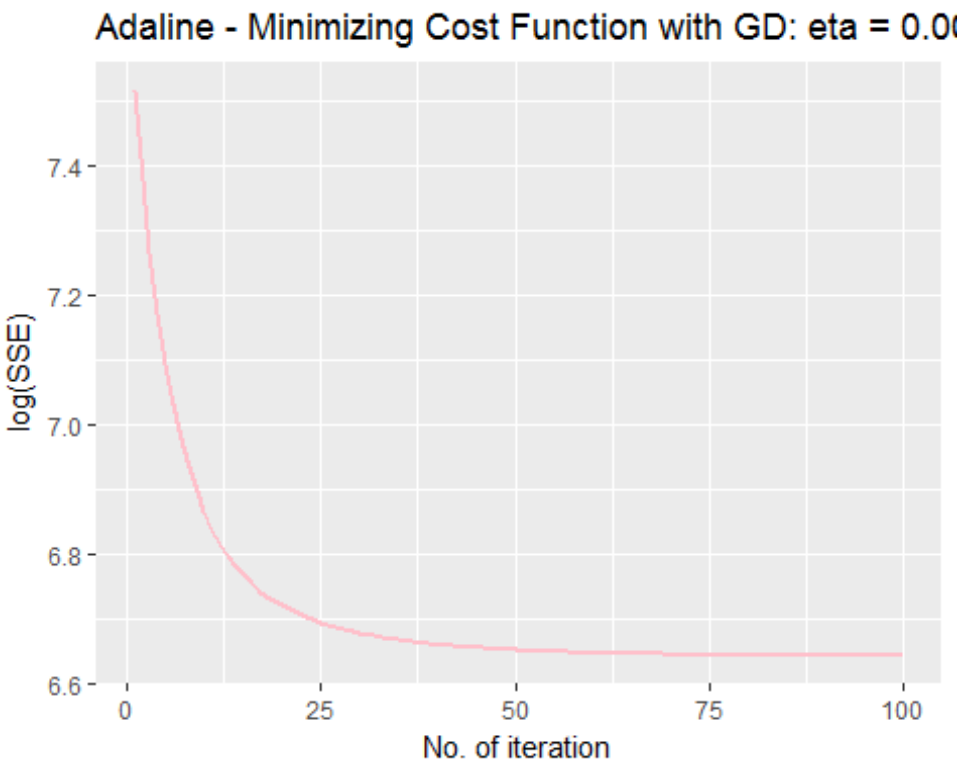
```
## word.freq.our               0.104496590
## word.freq.over              0.060959080
## word.freq.remove            0.154066905
## word.freq.internet          0.067026278
## word.freq.order             0.036184105
## word.freq.mail              0.015238185
## word.freq.receive           0.036489739
## word.freq.will             -0.047590505
## word.freq.people            0.011170719
## word.freq.report            0.002306505
## word.freq.addresses         0.012193955
## word.freq.free              0.115370202
## word.freq.business          0.052993809
## word.freq.email             0.070666060
## word.freq.you               0.056211414
## word.freq.credit            0.065496249
## word.freq.your              0.124652583
## word.freq.font              0.088931289
## word.freq.000               0.116685271
## word.freq.money             0.077642421
## word.freq.hp               -0.071915708
## word.freq.hpl              -0.041010933
## word.freq.george           -0.081889952
## word.freq.650              -0.016829319
## word.freq.lab              -0.020489754
## word.freq.labs             -0.043312766
## word.freq.telnet           -0.008556181
## word.freq.857               0.016737453
## word.freq.data             -0.043581506
## word.freq.415               0.017582877
## word.freq.85               -0.029067954
## word.freq.technology        0.020907370
## word.freq.1999             -0.028943961
## word.freq.parts            -0.018089757
## word.freq.pm               -0.014095638
## word.freq.direct            0.045354068
## word.freq.cs               -0.011440889
## word.freq.meeting          -0.050123597
## word.freq.original         -0.027687121
## word.freq.project          -0.042740310
## word.freq.re               -0.069382306
## word.freq.edu              -0.071447841
## word.freq.table            -0.029508582
## word.freq.conference       -0.032096434
## char.freq.semi             -0.052464511
## char.freq.lparen           -0.036282054
## char.freq.lbrack           -0.007155919
## char.freq.bang              0.141615621
## char.freq.dollar            0.103646870
## char.freq.hash              0.019884924
```

```
## capital.run.length.average  0.009822735
## capital.run.length.longest  0.031277263
## capital.run.length.total    0.093515095
## V58                        -0.202723377
```

```
# check confusion matrix
confusion_matrix = result.adalineGD$confusion_matrix
confusion_matrix
```

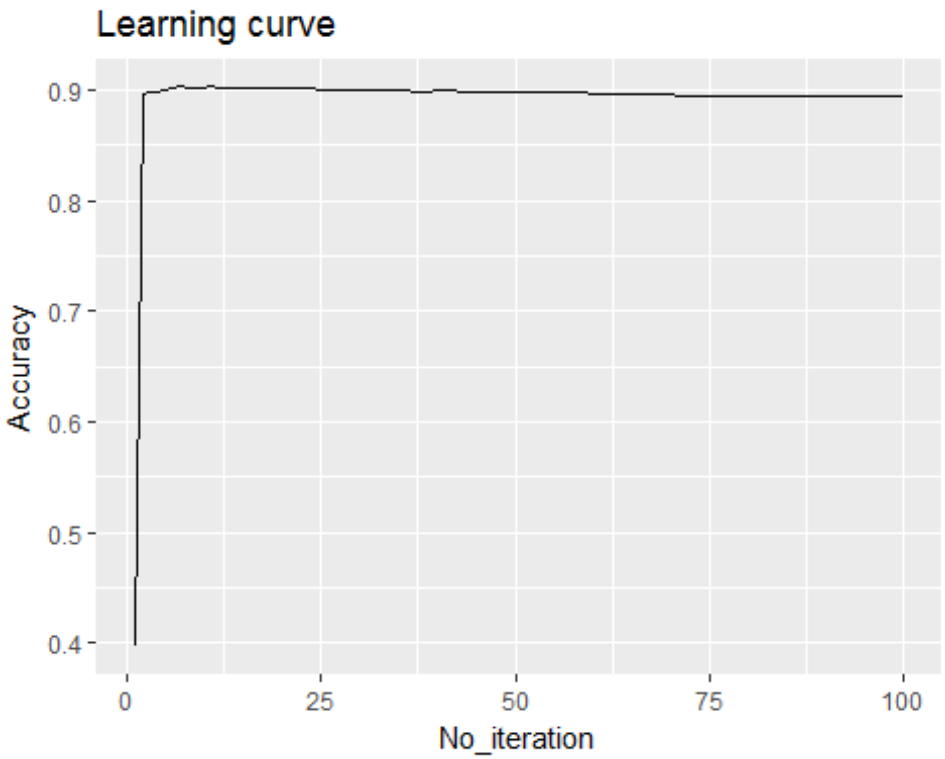
```
##          Negative Positive
## Negative     2119      103
## Positive      288     1170
```

```
# check error and other matrics in each iteration
View(result.adalineGD$infomatrix)

###### Model Evaluation ############
# plot cost function minimization process
ggplot(result.adalineGD$infomatrix, aes(x = No_iteration, y = cost_func
tion)) +
  geom_line(size = 0.8, col = "pink") +
  xlab("No. of iteration") +
  ylab("log(SSE)") +
  ggtitle("Adaline - Minimizing Cost Function with GD: eta = 0.00001")
```

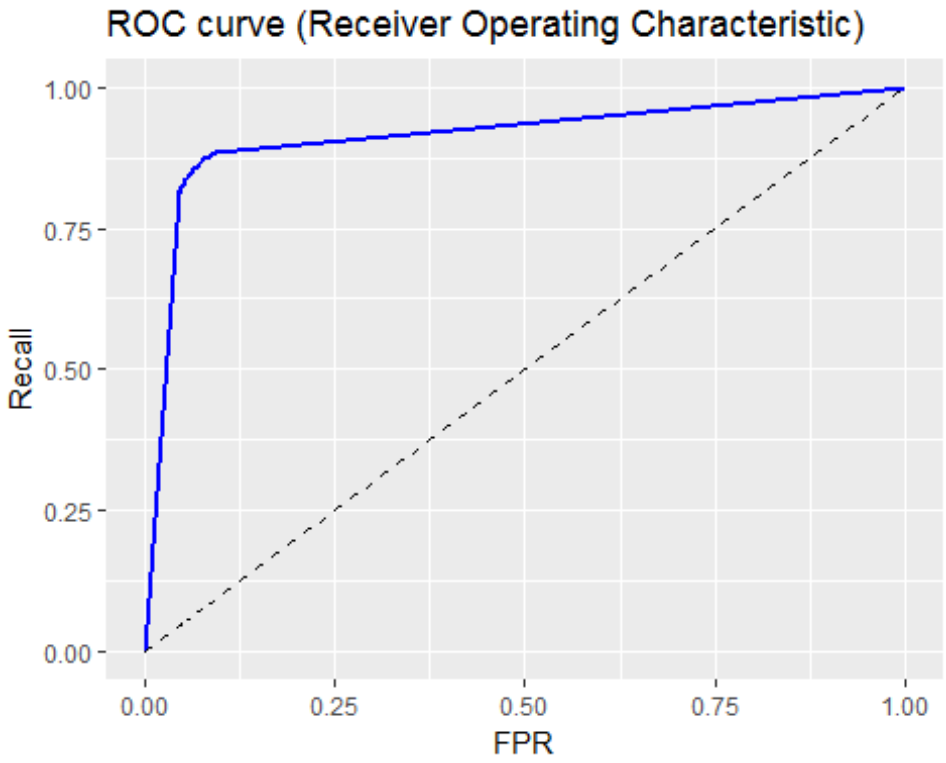



```
# plot accuracy as a function of learning effort
ggplot(result.adalineGD$infomatrix, aes(x = No_iteration, y = Accuracy)
```

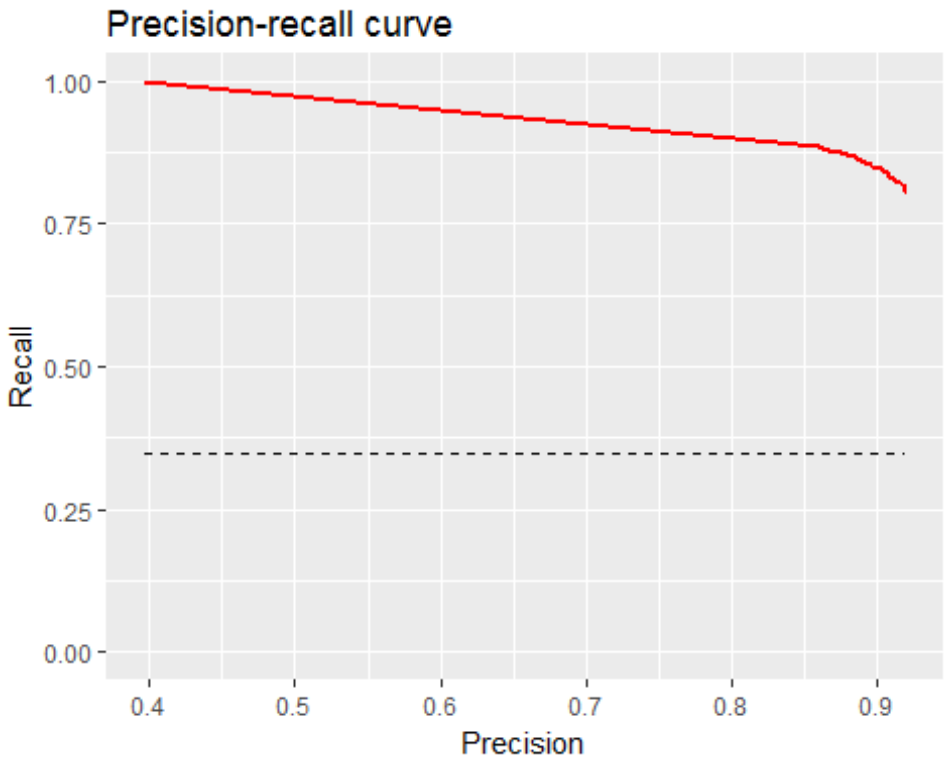
```
) +
  geom_line() +
  ggtitle("Learning curve")
```



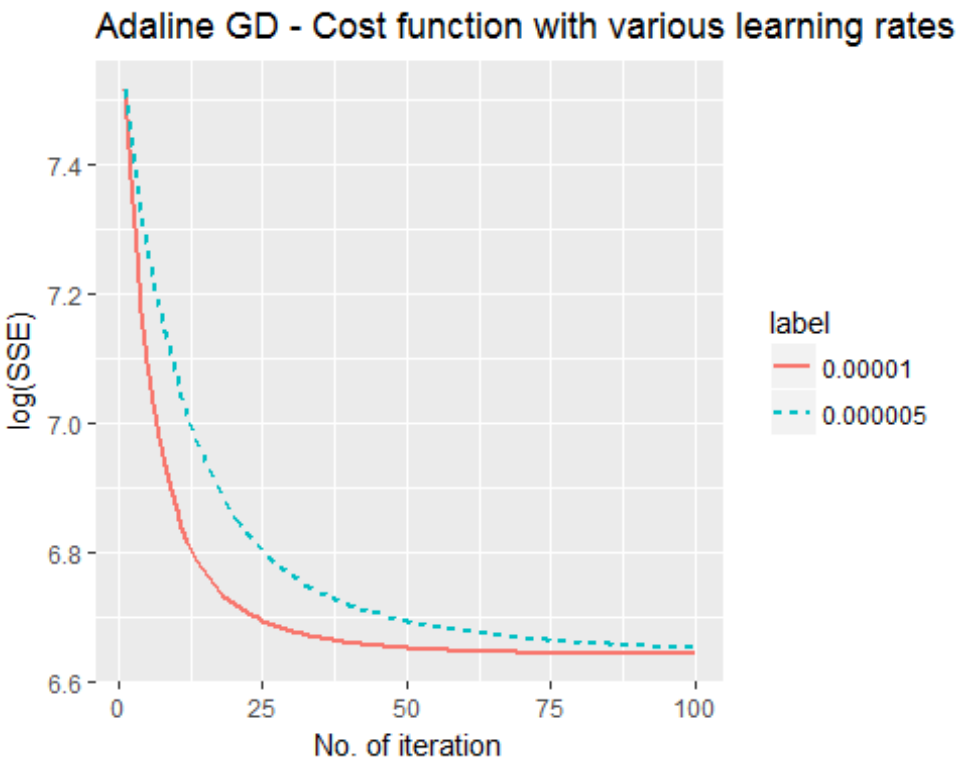

```
# plot ROC (Receiver Operating Characteristic) curve
c = rep(0,8)
ggplot(rbind(c,result.adalineGD$infomatrix), aes(x = FPR, y = Recall))
+
  geom_line(col="blue",size=1)+
  geom_line(aes(y=FPR),linetype=2,size=0.6 ) +
  ggtitle("ROC curve (Receiver Operating Characteristic)")
```




```
# plot precision vs. recall
baseline = rep(((confusion_matrix[1,2]+confusion_matrix[2,2])/nrow(trai
ning)),nrow(result.adalineGD$infomatrix))
ggplot(cbind(result.adalineGD$infomatrix, baseline), aes(x = Precision,
 y = Recall)) +
  geom_line(col="red",size=1) +
  geom_line(aes(y=baseline),linetype=2,size=0.6) +
  ylim(0, 1) +
  ggtitle("Precision-recall curve")
```




```r
# Plot cost function using various learning rate
result1 <- adalineGD(training[,-58], y1 ,n.iter = 100, eta = 0.00001)
result2 <- adalineGD(training[,-58], y1 ,n.iter = 100, eta = 0.000005)
result1 <- result1$infomatrix
result2 <- result2$infomatrix
label <- rep("0.00001", dim(result1)[1])
result1 <- cbind(label, result1)
label <- rep("0.000005", dim(result2)[1])
result2 <- cbind(label, result2)
df <- rbind(result1, result2)
ggplot(df, aes(x = No_iteration, y = cost_function)) +
  geom_line(aes(color=label, linetype=label), size = 1) +
  xlab("No. of iteration") +
  ylab("log(SSE)") +
  ggtitle("Adaline GD - Cost function with various learning rates")
```




```
######## Test model on testing set #########
y2 <- rep(1, nrow(testing))
y2[testing[,58]==0] <- -1
w = result.adalineGD$w
test.adalineGD = predict.adaline(w, testing[,-58], y2)
test.adalineGD

## $infomatrix
##   Matrics errors          Accuracy             Error         Precisi
on
## 1 Matrics     97 0.894679695982628 0.105320304017372 0.9271523178807
95
##              Recall              FPR
## 1 0.788732394366197 0.03886925795053
##
## $confusion_matrix
##          Negative Positive
## Negative      544       22
## Positive       75      280

##############################################################################
############################## Logistic Regression ###########################
##############################################################################

logisticGD <- function(X, y, n.iter, eta) {

  # extend input vector and initialize extended weight
```

```r
  X[, dim(X)[2] + 1] <- 1
  X <- as.matrix(X)
  w <- as.matrix(rep(0, dim(X)[2]))

  # initialize cost values - gets updated according to epochnums - numb
er of epochs
  cost <- rep(0, n.iter)
  errors <- rep(0, n.iter)

  # initialize various metrics
  TN = rep(0, n.iter)
  FN = rep(0, n.iter)
  FP = rep(0, n.iter)
  TP = rep(0, n.iter)
  Accuracy <- rep(0, n.iter)
  Error <- rep(0, n.iter)
  Precision <- rep(0, n.iter)
  Recall <- rep(0, n.iter)
  FPR <- rep(0, n.iter)

  #Sigmoid function
  sigmoid <- function(z)
  {
    g <- 1/(1+exp(-z))
    return(g)
  }

  # loop over the number of epochs
  for (i in 1:n.iter) {

    # find the number of wrong prediction before weight update
    for (j in 1:dim(X)[1]) {

      # compute net input
      z <- sigmoid(sum(w * X[j, ]))
      #z <- 1/(1+exp(-sum(w * X[j, ])))

      # quantizer
      if (z < 0.5) {
        ypred <- 0
        }else {
        ypred <- 1
        }

      # comparison with actual labels and counting error
      if(ypred != y[j]) {
        errors[i] <- errors[i] + 1
      }
```

```
      # metrics
      if(ypred==0 && y[j]==0){TN[i] = TN[i]+1}
      if(ypred==0 && y[j]== 1){FN[i] = FN[i]+1}
      if(ypred== 1 && y[j]==0){FP[i] = FP[i]+1}
      if(ypred== 1 && y[j]==1){TP[i] = TP[i]+1}

      Accuracy[i] <- (TP[i]+TN[i])/(FP[i]+FN[i]+TP[i]+TN[i])
      Error[i] <- 1-Accuracy[i]
      Precision[i] <- TP[i]/(TP[i]+FP[i])
      Recall[i] <- TP[i]/(TP[i]+FN[i])
      FPR[i] <- FP[i]/(FP[i]+TN[i])
    }

    # update the cost function(cost function is the formula given in cl
assnotes)
    # cost[i] <- sum((y - X %*% w)^2)/2
    g <- sigmoid(X %*% w)
    cost[i] <- sum((-y*log(g)) - ((1-y)*log(1-g)))

    # update weight according to gradient descent
    p = t(X) %*% (y - g)
    w <- w + eta* p
  }

  # data frame consisting of cost and error info
  confusion_matrix <- matrix(c(TN[n.iter],FN[n.iter],FP[n.iter],TP[n.it
er]),nrow = 2)
  colnames(confusion_matrix) <- c("Negative","Positive")
  rownames(confusion_matrix) <- c("Negative","Positive")

  infomatrix <- matrix(rep(0, 8 * n.iter), nrow = n.iter, ncol = 8)
  infomatrix[, 1] <- 1:n.iter
  infomatrix[, 2] <- log(cost)
  infomatrix[, 3] <- errors
  infomatrix[, 4] <- Accuracy
  infomatrix[, 5] <- Error
  infomatrix[, 6] <- Precision
  infomatrix[, 7] <- Recall
  infomatrix[, 8] <- FPR

  infodf <- as.data.frame(infomatrix)
  names(infodf) <- c("No_iteration", "cost_function", "errors","Accurac
y","Error","Precision","Recall","FPR")

  infolist <- list(w,infodf,confusion_matrix)
  names(infolist) <- c("w","infomatrix","confusion_matrix")

  return(infolist)
}
```

```r
predict.logistic <- function(w, X, y){
  # extend input vector and initialize extended weight
  X[, dim(X)[2] + 1] <- 1
  X <- as.matrix(X)

  # initialize metrics
  errors <- 0
  TN = 0
  FN = 0
  FP = 0
  TP = 0
  Accuracy <- 0
  Error <- 0
  Precision <- 0
  Recall <- 0
  FPR <- 0

  #Sigmoid function
  sigmoid <- function(z)
  {
    g <- 1/(1+exp(-z))
    return(g)
  }

  # find the number of wrong prediction
  for (j in 1:dim(X)[1]) {

    # compute net input
    z <- sigmoid(sum(w * X[j,]))

    # quantizer
    if (z < 0.5) {
      ypred <- 0
    }else {
      ypred <- 1
    }

    # comparison with actual values and counting error
    if(ypred != y[j]) {
      errors <- errors + 1
    }

    # metrics
    if(ypred==0 && y[j]==0){TN = TN+1}
    if(ypred==0 && y[j]== 1){FN = FN+1}
    if(ypred== 1 && y[j]==0){FP = FP+1}
    if(ypred== 1 && y[j]==1){TP = TP+1}
```

```
    Accuracy <- (TP+TN)/(FP+FN+TP+TN)
    Error <- 1-Accuracy
    Precision <- TP/(TP+FP)
    Recall <- TP/(TP+FN)
    FPR <- FP/(FP+TN)
  }

  # data frame consisting of cost and error info
  confusion_matrix <- matrix(c(TN,FN,FP,TP),nrow = 2)
  colnames(confusion_matrix) <- c("Negative","Positive")
  rownames(confusion_matrix) <- c("Negative","Positive")

  infomatrix <- matrix(rep(0, 7), nrow = 1, ncol = 7)
  infomatrix[, 1] <- "Matrics"
  infomatrix[, 2] <- errors
  infomatrix[, 3] <- Accuracy
  infomatrix[, 4] <- Error
  infomatrix[, 5] <- Precision
  infomatrix[, 6] <- Recall
  infomatrix[, 7] <- FPR

  infodf <- as.data.frame(infomatrix)
  names(infodf) <- c("Matrics", "errors","Accuracy","Error","Precision"
,"Recall","FPR")

  infolist <- list(infodf,confusion_matrix)
  names(infolist) <- c("infomatrix","confusion_matrix")

  return(infolist)

}

####### Apply to the training set. #########
y1=training[,58]
result.logGD <- logisticGD(training[,-58], y1 ,n.iter = 100, eta = 0.00
015)
# check weights
result.logGD$w
```

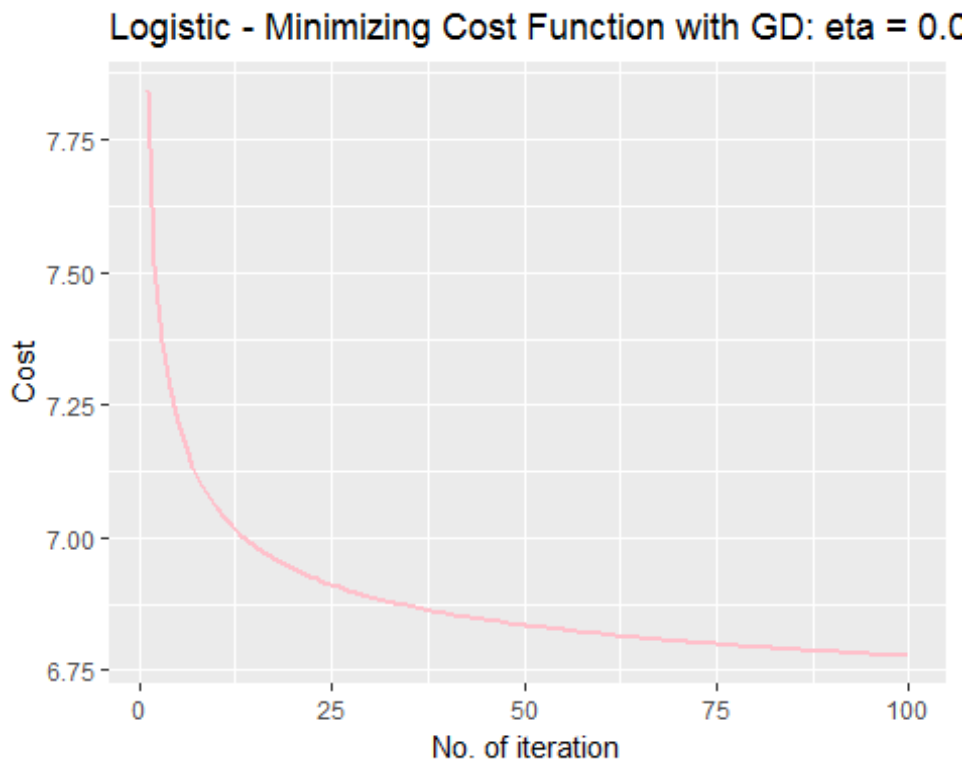
```
##                                  [,1]
## word.freq.make             -0.076581161
## word.freq.address          -0.104065464
## word.freq.all               0.067405048
## word.freq.3d                0.234358950
## word.freq.our               0.329385765
## word.freq.over              0.151354167
## word.freq.remove            0.825159810
## word.freq.internet          0.250189111
## word.freq.order             0.157182326
```

```
## word.freq.mail               0.072708653
## word.freq.receive            0.052482892
## word.freq.will              -0.149725733
## word.freq.people             0.004357380
## word.freq.report             0.031170687
## word.freq.addresses          0.245896982
## word.freq.free               0.406465429
## word.freq.business           0.385777356
## word.freq.email              0.197053688
## word.freq.you                0.198462983
## word.freq.credit             0.370058986
## word.freq.your               0.302223066
## word.freq.font               0.340453020
## word.freq.000                0.771427143
## word.freq.money              0.285879149
## word.freq.hp                -0.729663045
## word.freq.hpl               -0.490668127
## word.freq.george            -0.659631683
## word.freq.650               -0.029949419
## word.freq.lab               -0.277525392
## word.freq.labs              -0.237387092
## word.freq.telnet            -0.130514326
## word.freq.857               -0.087338219
## word.freq.data              -0.356187269
## word.freq.415               -0.092750345
## word.freq.85                -0.265786035
## word.freq.technology         0.130033137
## word.freq.1999              -0.140690221
## word.freq.parts             -0.079572351
## word.freq.pm                -0.217924172
## word.freq.direct            -0.003394096
## word.freq.cs                -0.254307548
## word.freq.meeting           -0.446355285
## word.freq.original          -0.180603341
## word.freq.project           -0.337191175
## word.freq.re                -0.490331307
## word.freq.edu               -0.575289906
## word.freq.table             -0.134934894
## word.freq.conference        -0.271691387
## char.freq.semi              -0.228397134
## char.freq.lparen            -0.054902845
## char.freq.lbrack            -0.075319964
## char.freq.bang               0.693764243
## char.freq.dollar             0.971022266
## char.freq.hash               0.208249584
## capital.run.length.average   0.163435994
## capital.run.length.longest   0.467332811
## capital.run.length.total     0.431154982
## V58                         -0.699199253
```

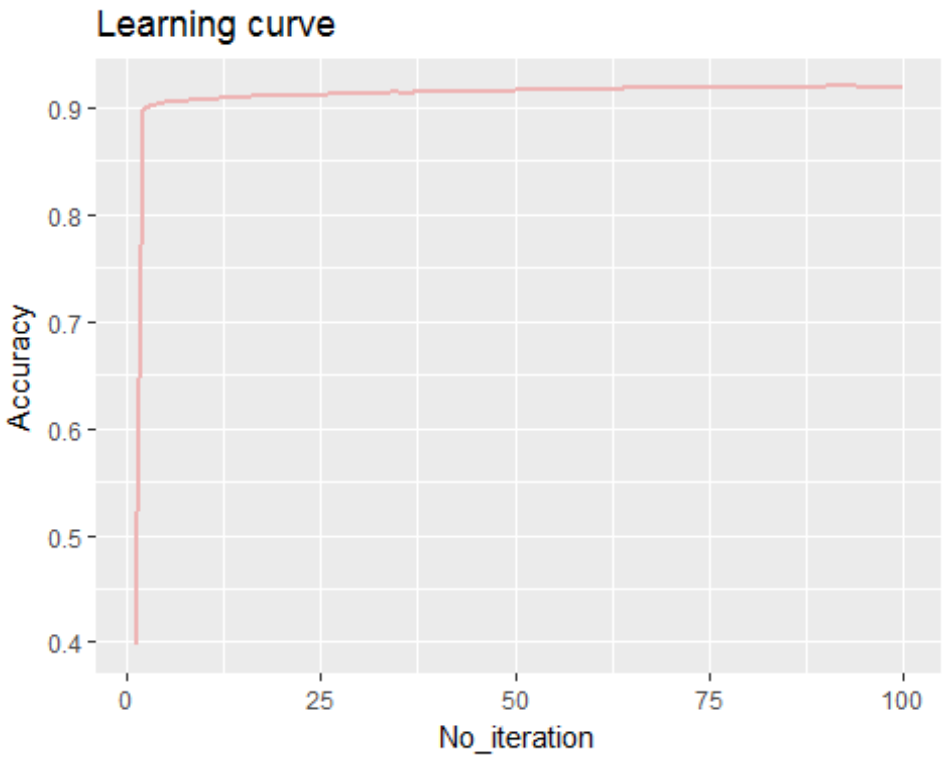
```r
# check confusion matrix
confusion_matrix = result.logGD$confusion_matrix
confusion_matrix
```

```
##          Negative Positive
## Negative     2114      108
## Positive      189     1269
```

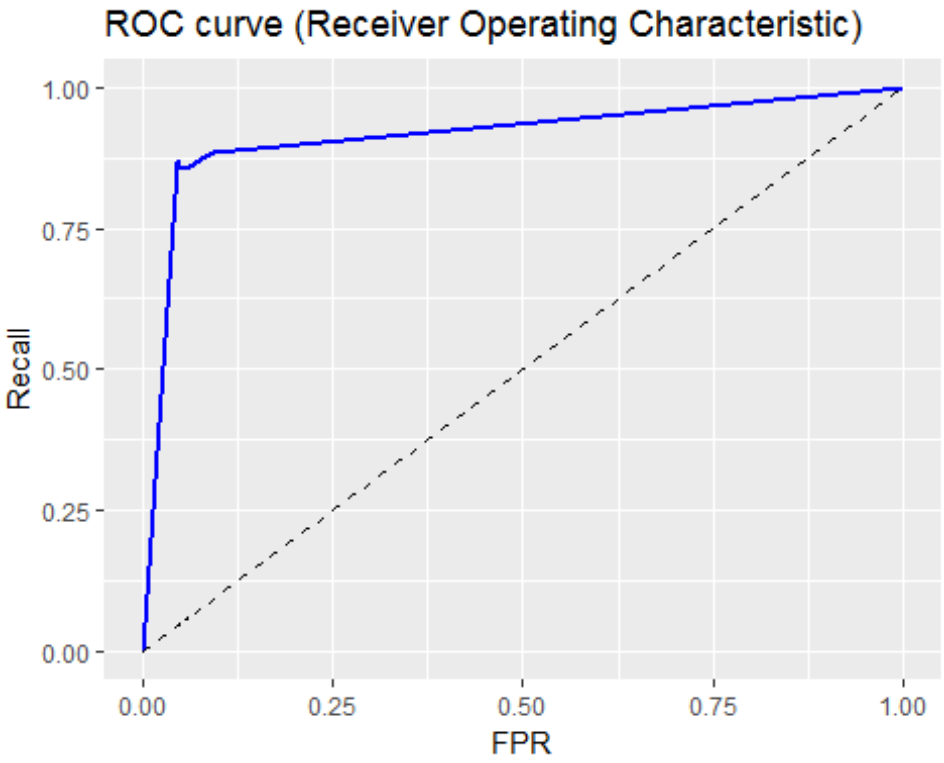
```r
# check error and cost function in each iteration
View(result.logGD$infomatrix)


###### Model Evaluation ############

# plot cost function minimization process
ggplot(result.logGD$infomatrix, aes(x = No_iteration, y = cost_function
)) +
  geom_line(size = 0.8, col = "pink") +
  xlab("No. of iteration") +
  ylab("Cost") +
  ggtitle("Logistic - Minimizing Cost Function with GD: eta = 0.00015")
```



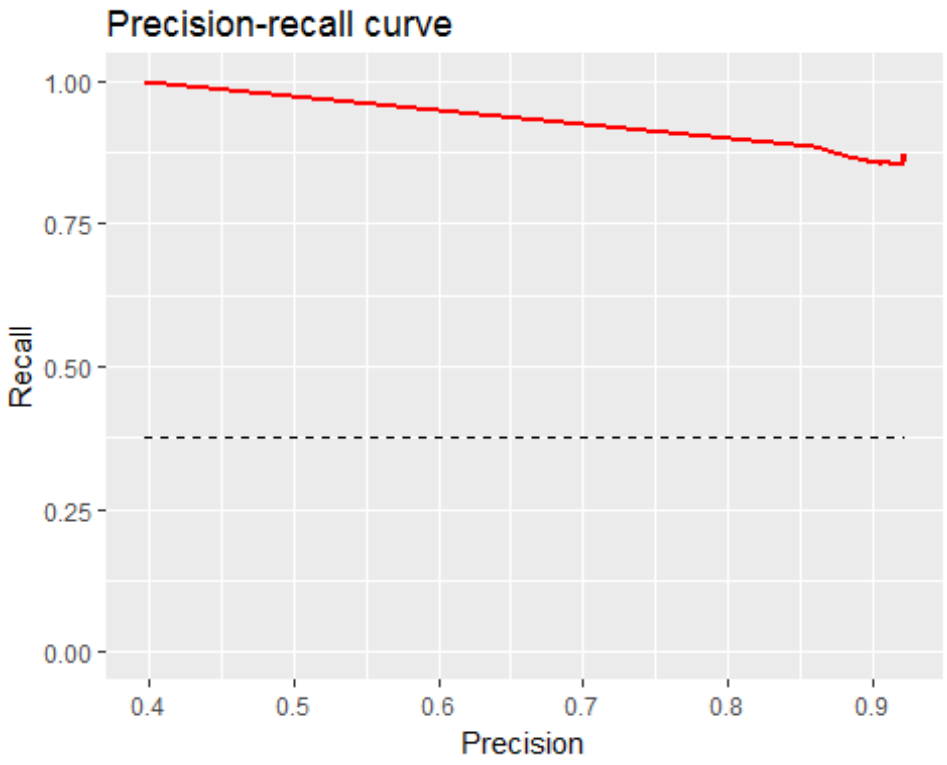

```r
# plot accuracy as a function of learning effort
ggplot(result.logGD$infomatrix, aes(x = No_iteration, y = Accuracy)) +
  geom_line(size = 0.8, col = "rosybrown2") +
  ggtitle("Learning curve")
```



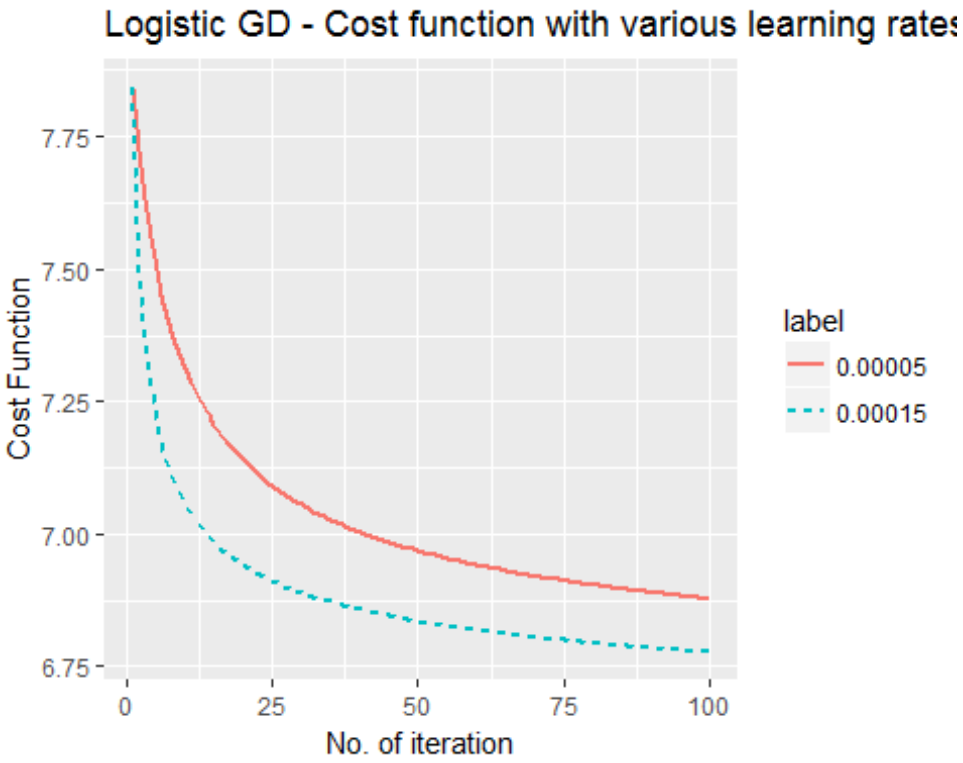

```
# plot ROC (Receiver Operating Characteristic) curve
c = rep(0,8)
ggplot(rbind(c,result.logGD$infomatrix), aes(x = FPR, y = Recall)) +
  geom_line(col="blue",size=1)+
  geom_line(aes(y=FPR),linetype=2,size=0.6 ) +
  ggtitle("ROC curve (Receiver Operating Characteristic)")
```




```
# plot precision vs. recall
baseline = rep(((confusion_matrix[1,2]+confusion_matrix[2,2])/nrow(trai
ning)),nrow(result.logGD$infomatrix))
ggplot(cbind(result.logGD$infomatrix, baseline), aes(x = Precision, y =
 Recall)) +
  geom_line(col="red",size=1) +
  geom_line(aes(y=baseline),linetype=2,size=0.6) +
  ylim(0, 1) +
  ggtitle("Precision-recall curve")
```




```r
# Plot cost function using various learning rate
result1 <- logisticGD(training[,-58], y1 ,n.iter = 100, eta = 0.00005)
result2 <- logisticGD(training[,-58], y1 ,n.iter = 100, eta = 0.00015)
result1 <- result1$infomatrix
result2 <- result2$infomatrix
label <- rep("0.00005", dim(result1)[1])
result1 <- cbind(label, result1)
label <- rep("0.00015", dim(result2)[1])
result2 <- cbind(label, result2)
df <- rbind(result1, result2)
ggplot(df, aes(x = No_iteration, y = cost_function)) +
  geom_line(aes(color=label, linetype=label), size = 1) +
  xlab("No. of iteration") +
  ylab("Cost Function") +
  ggtitle("Logistic GD - Cost function with various learning rates")
```




```
####### Apply to the testing set. ######
y2=testing[,58]
w = result.logGD$w
test.logGD = predict.logistic(w, testing[,-58], y2)
test.logGD
```

```
## $infomatrix
##   Matrics errors          Accuracy              Error         Precis
ion
## 1 Matrics     64 0.930510314875136 0.0694896851248643 0.919308357348
703
##              Recall               FPR
## 1 0.898591549295775 0.049469964664311
##
## $confusion_matrix
##          Negative Positive
## Negative      538       28
## Positive       36      319
```

```
## training.f <- training
```

```
> testing.f <- testing
```

```
> training.f$spam <- as.factor(training.f$spam)
```

```
> testing.f$spam <- as.factor(testing.f$spam)
```

```
> registerDoMC(cores=5)

> sigDist <- sigest(spam ~ ., data = training.f, frac = 1)

> svmTuneGrid <- data.frame(.sigma = sigDist[1], .C = 2^(-2:7))

Warning message:

In data.frame(.sigma = sigDist[1], .C = 2^(-2:7)) :

  row names were found from a short variable and have been discarded

> x <- train(spam ~ ., data = training.f,method = "svmRadial",tuneGrid
= svmTuneGrid,trControl = trainControl(method = "repeatedcv", repeats =
 5, classProbs =  FALSE))

> proc.time()

   user  system elapsed

618.762  25.425 507.282

> plot(x, col = "hotpink2", lwd = 1.6, cex = 1)

> pred <- predict(x,testing.f[,1:57])

> acc <- confusionMatrix(pred,testing.f$spam) > training.f <- training

> testing.f <- testing

> training.f$spam <- as.factor(training.f$spam)

> testing.f$spam <- as.factor(testing.f$spam)

> registerDoMC(cores=5)

> sigDist <- sigest(spam ~ ., data = training.f, frac = 1)

> svmTuneGrid <- data.frame(.sigma = sigDist[1], .C = 2^(-2:7))

Warning message:

In data.frame(.sigma = sigDist[1], .C = 2^(-2:7)) :

  row names were found from a short variable and have been discarded

> x <- train(spam ~ ., data = training.f,method = "svmRadial",tuneGrid
= svmTuneGrid,trControl = trainControl(method = "repeatedcv", repeats =
 5, classProbs =  FALSE))

> proc.time()

   user  system elapsed

618.762  25.425 507.282
```

```
> plot(x, col = "hotpink2", lwd = 1.6, cex = 1)
> pred <- predict(x,testing.f[,1:57])
> acc <- confusionMatrix(pred,testing.f$spam)
```