Computer lab 1

Emil K Svensson

9 November 2016

Assignment 1

1.1

```
#setwd("Documents/732A95/lab1")

data <- read.csv2("spambase.csv",sep = ",",
                  header = TRUE, stringsAsFactors = FALSE) # Reading data

set.seed(12345)

# Dividing data in to a test and traning data set
n     <- dim(data)[1]
id    <- sample( 1:n, floor(n*0.5) )
train <- data[id,]
test  <- data[-id,]
```

1.2

As the instruction was a bit unclear on what features the function should be returning and what features the functions should had so my function returns a listobject with Distance-matrix, classifications, predicted values, confusion matrix and missclassification-rates for both train and test data. The function requires that the target value is located at the last column.

```
knearest <- function(data, K, newdata){

train <- data
test <- newdata

# i Compute xhat
xhat <- as.matrix( data[,-length(data)] )
xhat <- xhat / sqrt(rowSums( xhat^2 ))

# ii Compute yhat
yhat <- newdata[,-length(newdata)]
yhat <- yhat / sqrt(rowSums(yhat^2))

# iii Compute matrix C as abs(c(Xi,Ymj))
C <-  xhat %*% t(yhat)

# iv Compute the distance

D <- 1 - C

#Returns orders for each column (the lowest value is the fist value)
myOrders <- apply(D,MARGIN = 2, FUN = order)

#Keeps the K lowest values in the distance matrix
myOrders <- matrix(myOrders[1:K,], nrow=K)

#Extracts the K number of values of the observed y-variables
myData<-train[myOrders[1:K,],length(train)]

#puts the y-observations in a matrix where each column represents a column
myData<- matrix(myData,nrow=K,byrow = FALSE)

#Majority voting
myClass<-apply(myData,MARGIN = 2, FUN =function(X) round(mean(X)))

#Generating predictions for 1.6
myPredict<-apply(myData,MARGIN = 2, FUN =function(X) mean(X))


#missclassification rates for training and test
mcr<- c(1 - mean(myClass == test[,length(test)]))
#mcrTrain <- 1 - mean(myClass == train[,length(train)])
#mcr<-c(Test = mcrTest, Train = mcrTrain)

#Confusion-matrix for the training and test dataset
minRejm           <- data.frame(myClass = myClass)
minRejm$myTest    <- test[,length(test)]
```

```
minRejm$myTrain   <- train[,length(train)]
cmTest <- table(myClass = minRejm$myClass,myTest  = minRejm$myTest)
cmTrain<- table(myClass = minRejm$myClass,myTrain = minRejm$myTrain)


returnlist<-list(D = D, myClass = myClass,
                 cmTest = cmTest,cmTrain = cmTrain,
                 mcr=mcr,myPredict=myPredict)

return(returnlist)


}
```

1.3

```
mytest<-knearest(train, K = 5, test)
```

```
## [1] "For the traning-set"
```

```
## [1] 0.2021898
```

```
## [1] "For the test-set"
```

```
## [1] 0.3175182
```

A missclassification rate around 30 % can be considerd a good fit for the test data set.

First the confusion matrix for the training dataset

```
##        myTrain
## myClass   0   1
##       0 597 291
##       1 348 134
```

Here are the confusion matrix for the test dataset.

```
##        myTest
## myClass   0   1
##       0 695 193
##       1 242 240
```

1.4

And now when K equals 1.

```
myKN<-knearest(train, K = 1, test)
```

```
## Missclassification rate for the traning-set
```

```
## 0.2021898
```

```
## For the traning-set
```

```
## 0.3474453
```

First the confusion matrix for the training dataset

```
##         myTrain
## myClass   0   1
##       0 553 264
##       1 392 161
```

and here is the confusion matrix for the test dataset.

```
##         myTest
## myClass   0   1
##       0 639 178
##       1 298 255
```

The K = 1 preforms worse than the K = 5, which isn't that strange as a voting process with one observation is a complex model and is over-fitting the data as the voting process would be basicly assuming the closest neighbouhr has the same value as the one it's calculated from, which probably isn't a good asumption to make.

1.5

```
testKNN <- function(KKNpred,pi){
  a       <- data.frame(pred = as.numeric( KKNpred > pi ))
  a$test  <- test[,length(test)]
  return(table(a))
}

library(kknn)
kknnearest <- kknn(Spam~., train, test, distance = 2, k = 5)

tab1<-testKNN(predict(kknnearest),0.5)
print(tab1)
```

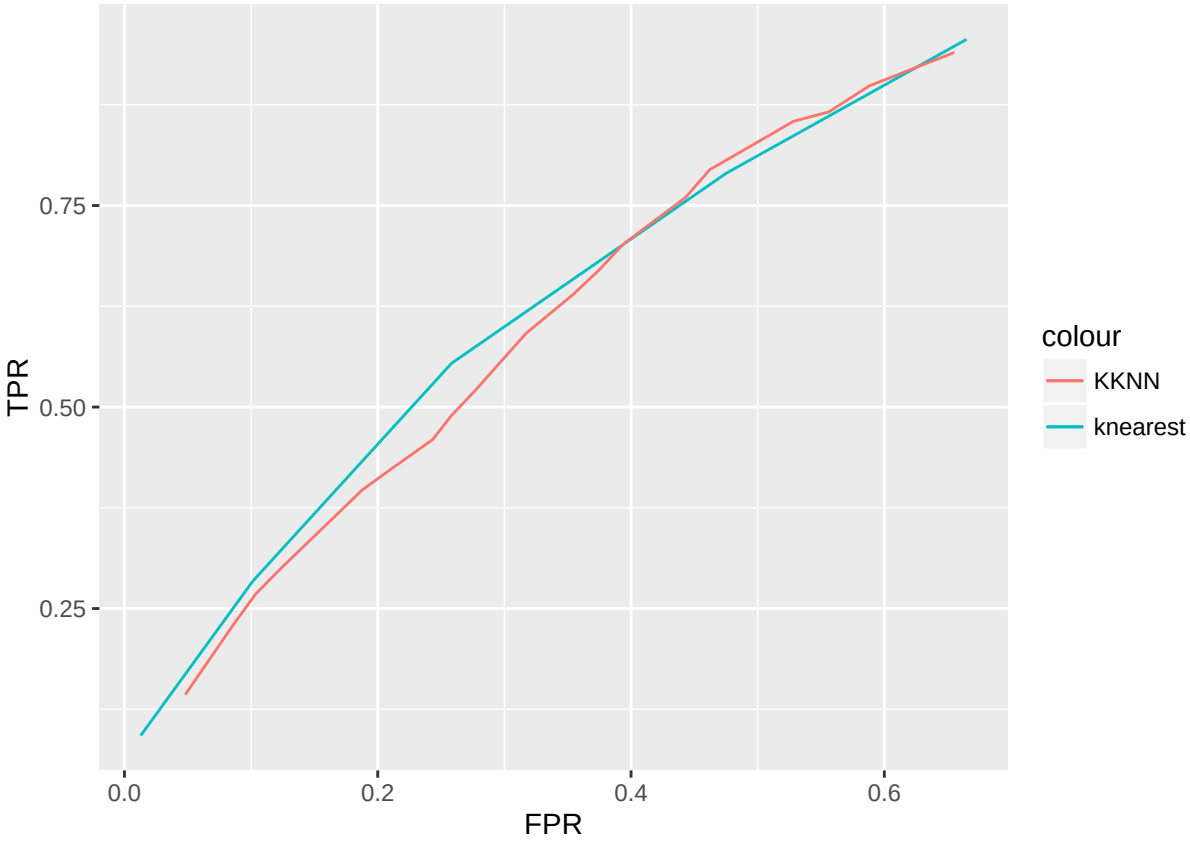
```
##     test
## pred   0   1
##    0 640 177
##    1 297 256
```

```
print((1 -sum(diag(tab1))/(sum(tab1[,1])+sum(tab1[,2]))))
```

```
## [1] 0.3459854
```

The missclassification rate is higher for the test-data set compared to knearest with K = 5. Which is notable, the kknn-model has a missclassification rate comparable with the knearest-model with K = 1, although this dosn't mean the kknn-model is overfitted as the knearest(K=1), the decision boundaries for the kknn model is although not good for this data set.

1.6



My own function knearest seems to have an larger area than the kknn function and should be the prefered model in this case

Assignment 2

2.1

```
machines <- data.frame(read.csv2("machines.csv"))
machines<-as.vector(unlist(machines$Length))
```

Read the data and transform it in to a vector.

2.2

The distrubution given is a exponential distrubution. The pdf for the exponetial distrubution is:

$p(x|\theta) = \theta e^-x\theta$

The likelihood is the following:

$\prod p(x|\theta) = \theta e^-\theta\sum x_i$

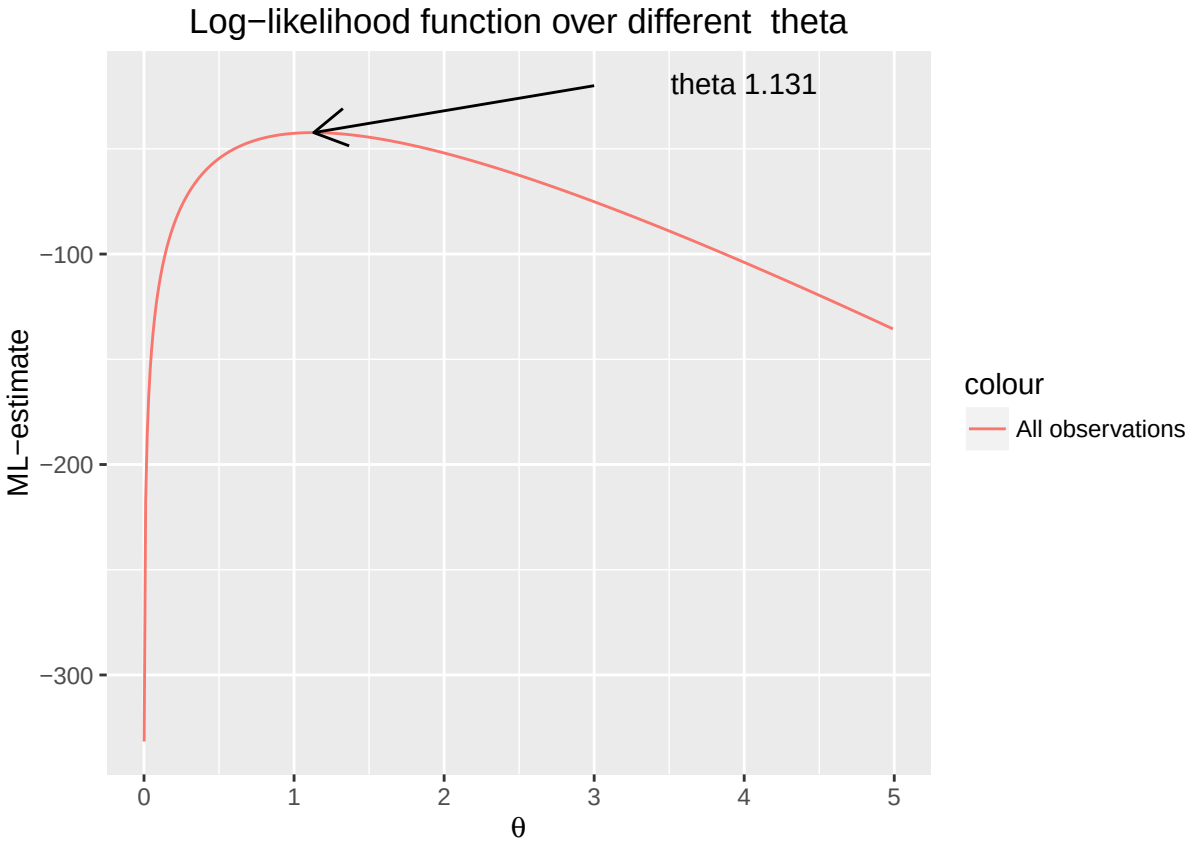
Finaly the log-likelihood was computed to:

```
  $log\,p(x|\theta)=n\,log(\theta)-\theta\sum_{n=1}^N\mathrm{x_i}$
```

From this expression an R function was created called explog.

```
explog <-function(theta,data){

  return(     length(data)*log(theta,base = exp(1)) - (theta*sum(data))    )
}
```

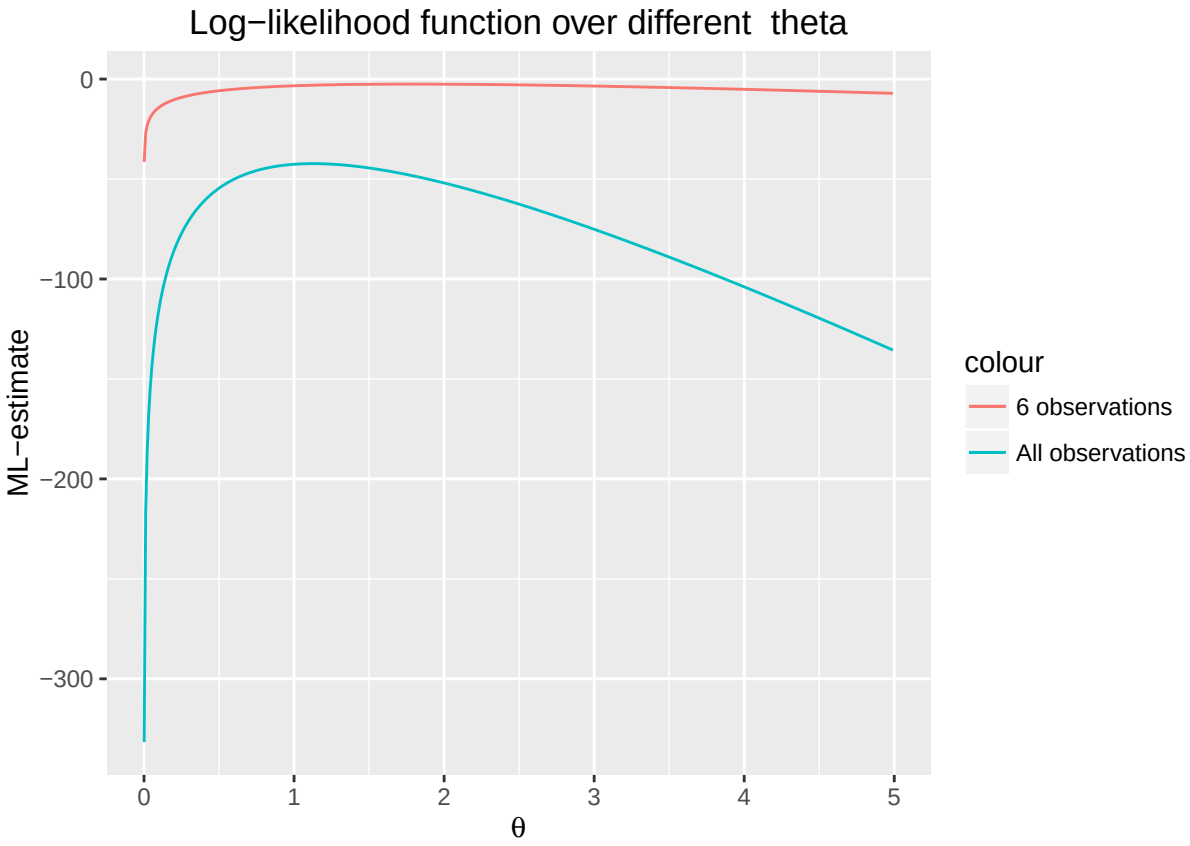
```
theta <- seq(0.001,5,by=0.01) #Generates a sequence of different
thetaML <- sapply( X = theta, FUN = explog, data = machines)
#apply the different thetas to the log-likelihood with the machine data-set.
plotData <- data.frame(x=theta,y=thetaML) #Put the data in a frame.


#Plotting
library(ggplot2)
p <- ggplot() + geom_line(data=plotData,aes(x=x,y=y,col="All observations")) +
                labs(title=paste("Log-likelihood function over different  theta"),
       x=expression(theta),y="ML-estimate")

a <- p + geom_segment(aes(x = 3, y = -20,
                          xend = theta[thetaML == max(thetaML)], yend = max(thetaML)),
                  arrow = arrow(length = unit(0.5, "cm"))) +
  annotate("text",x=4,y=-19,label = paste("theta",theta[thetaML == max(thetaML)]))

plot(a)
```



The maximum in the plot for the log-likelihood of the different thetas is estimated to 1.131.

2.3

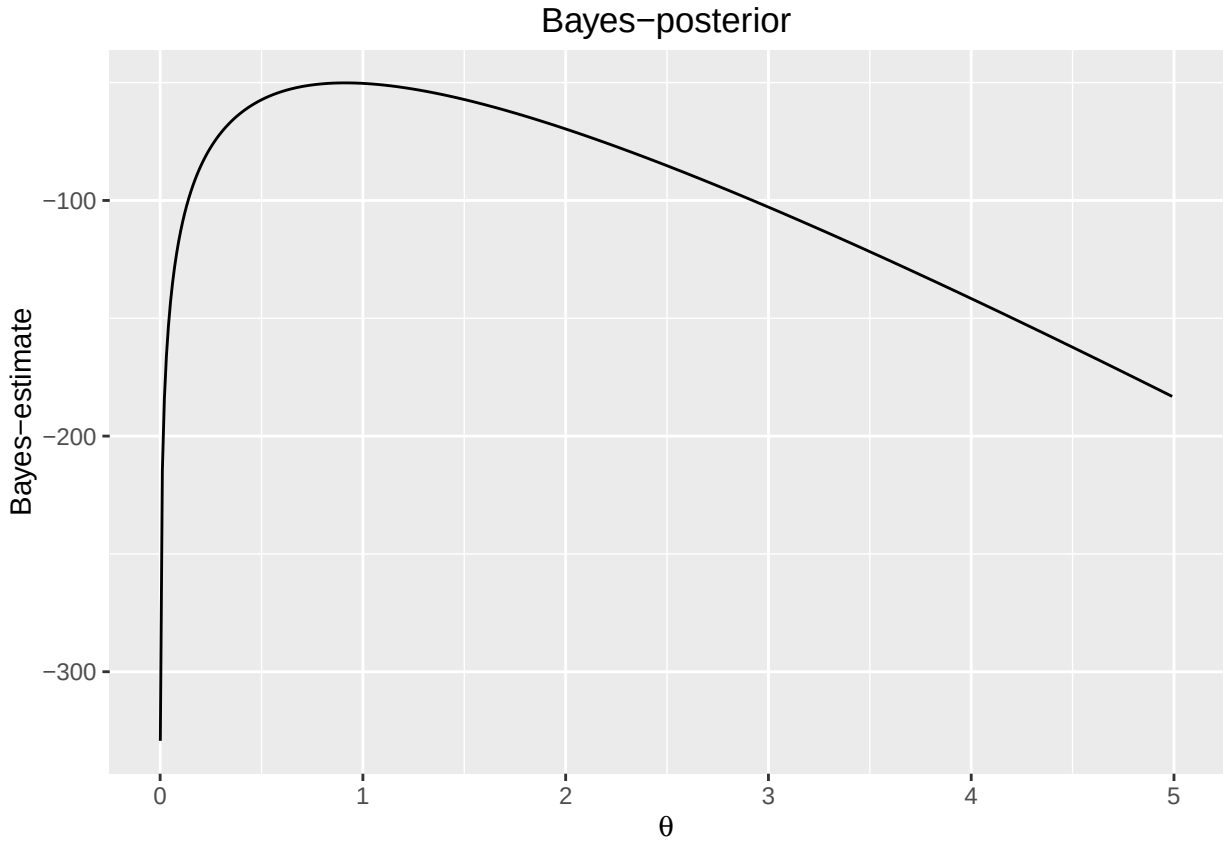
```
thetaML3 <- sapply( X = theta, FUN = explog, data = machines[1:6])
#Using only the 6 first observations of machines
plotData$y2 <- thetaML3

p + geom_line(data = plotData, aes(x=x,y=y2,col="6 observations"))
```



The red line represents the log likelihood function of different theta using six observations. These observations seem to have a higher intercept than the teal line which represents the function using all observations in the machine dataset. The function using less data seem to reach a max-point at a higher theta value and the max isn't that defined in the function compared to the function using all data. This show the dependence of data with these estimators.

2.4

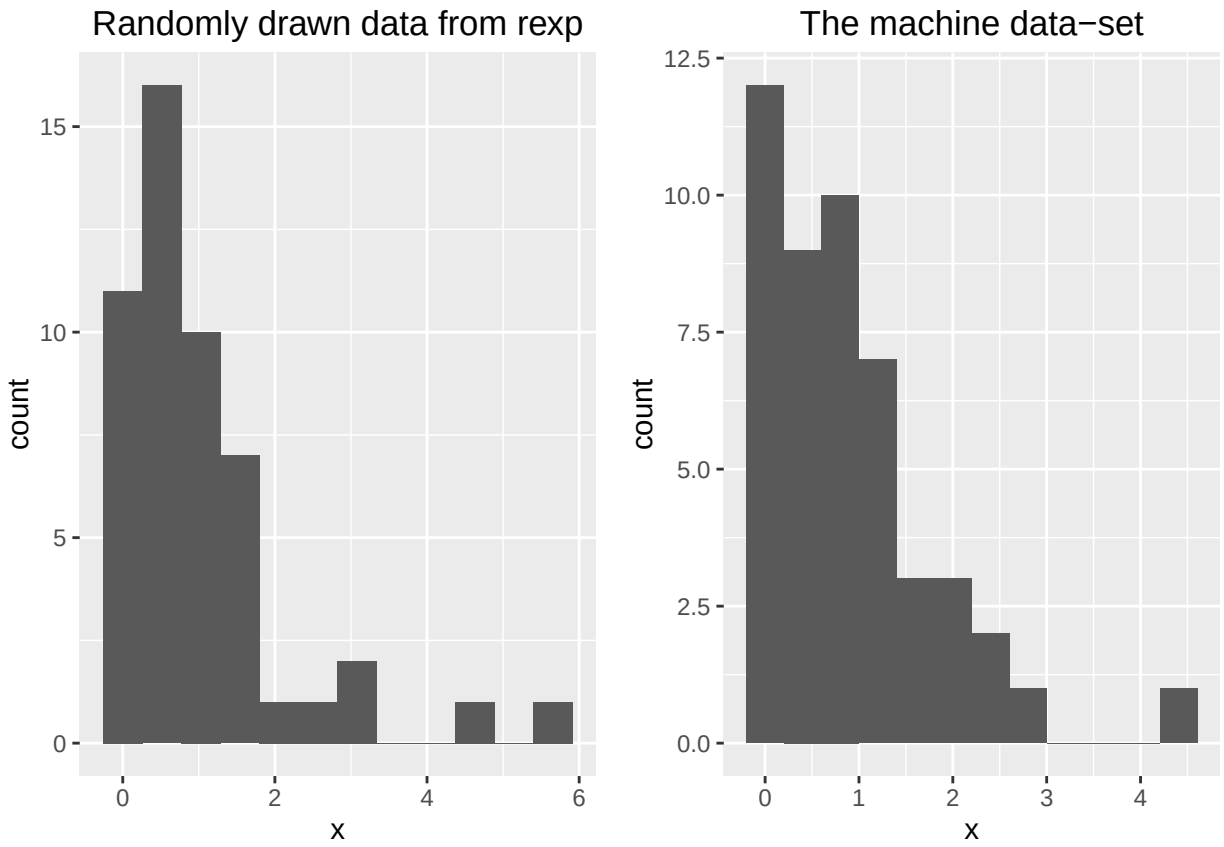
```
postBayes <- function(data,theta){
  lambda <- 10
  return(length(data) * log(theta) - theta * sum(data) +
           log(lambda) - (lambda*theta))

}

thetaBayes <- sapply(theta,FUN = postBayes,data = machines)

plotDataBayes <- data.frame(x=theta,y=thetaBayes)
B <- ggplot() + geom_line(data=plotDataBayes,aes(x=x,y=y)) +
  labs(title="Bayes-posterior",
       x=expression(theta),y="Bayes-estimate")
plot(B)
```



This function calculates the posterior for different thetas. The max-value here is about 0.9 and thats a lower estimate of theta.

2.5

```
library(gridExtra)
#extract the exact theta for the  ML-estimate
maxTheta <- theta[thetaML == max(thetaML)]
# Generate 50 random variables from the exponential distrubution with
# the ML-estimate extracted
set.seed(12345)
randomExpData<- data.frame(x=rexp(50,rate = maxTheta))

#Plotting
ab<- ggplot(data = randomExpData,aes(x=x)) +
  geom_histogram(bins = 12) +
  labs(title = "Randomly drawn data from rexp")
abc<- ggplot(data =data.frame(x=machines),aes(x=x)) +
  geom_histogram(bins = 12) +
  labs(title = "The machine data-set")

 grid.arrange( ab, abc, ncol=2)
```



The distribution of the data-set machines have similar properties as the randomly drawn values from the exponential distribution with the Maximum Likelihood estimaded theta. The most noteworthy difference is that the machines data set dosn't seem to have the initial peak as the exponential-distrubution should have. It decreases constantly instead of a initial increase and then decreasing. But over all it seems resonable to assume its an exponential distrubution because of the low amount of observations it's hard to demand a better fit.

Code

```
knitr::opts_chunk$set(echo = TRUE)

#setwd("Documents/732A95/lab1")

data <- read.csv2("spambase.csv",sep = ",",
                  header = TRUE, stringsAsFactors = FALSE) # Reading data

set.seed(12345)

# Dividing data in to a test and traning data set
n     <- dim(data)[1]
id    <- sample( 1:n, floor(n*0.5) )
train <- data[id,]
test  <- data[-id,]

knearest <- function(data, K, newdata){

train <- data
test <- newdata

# i Compute xhat
xhat <- as.matrix( data[,-length(data)] )
xhat <- xhat / sqrt(rowSums( xhat^2 ))

# ii Compute yhat
yhat <- newdata[,-length(newdata)]
yhat <- yhat / sqrt(rowSums(yhat^2))

# iii Compute matrix C as abs(c(Xi,Ymj))
C <-  xhat %*% t(yhat)

# iv Compute the distance

D <- 1 - C

#Returns orders for each column (the lowest value is the fist value)
myOrders <- apply(D,MARGIN = 2, FUN = order)

#Keeps the K lowest values in the distance matrix
myOrders <- matrix(myOrders[1:K,], nrow=K)

#Extracts the K number of values of the observed y-variables
myData<-train[myOrders[1:K,],length(train)]

#puts the y-observations in a matrix where each column represents a column
myData<- matrix(myData,nrow=K,byrow = FALSE)

#Majority voting
myClass<-apply(myData,MARGIN = 2, FUN =function(X) round(mean(X)))

#Generating predictions for 1.6
```

```
myPredict<-apply(myData,MARGIN = 2, FUN =function(X) mean(X))


#missclassification rates for training and test
mcr<- c(1 - mean(myClass == test[,length(test)]))
#mcrTrain <- 1 - mean(myClass == train[,length(train)])
#mcr<-c(Test = mcrTest, Train = mcrTrain)

#Confusion-matrix for the training and test dataset
minRejm           <- data.frame(myClass = myClass)
minRejm$myTest    <- test[,length(test)]
minRejm$myTrain   <- train[,length(train)]
cmTest <- table(myClass = minRejm$myClass,myTest  = minRejm$myTest)
cmTrain<- table(myClass = minRejm$myClass,myTrain = minRejm$myTrain)


returnlist<-list(D = D, myClass = myClass,
                 cmTest = cmTest,cmTrain = cmTrain,
                 mcr=mcr,myPredict=myPredict)

return(returnlist)


}

mytest<-knearest(train, K = 5, test)
print("For the traning-set")
print(knearest(train, K = 5, train)$mcr)
print("For the test-set")
print(mytest$mcr)
print(mytest$cmTrain)
print(mytest$cmTest)

myKN<-knearest(train, K = 1, test)
cat("Missclassification rate for the traning-set")
cat(knearest(train, K = 5, train)$mcr)
cat("For the traning-set")
cat(myKN$mcr)
print(myKN$cmTrain)

print(myKN$cmTest)


testKNN <- function(KKNpred,pi){
  a       <- data.frame(pred = as.numeric( KKNpred > pi ))
  a$test  <- test[,length(test)]
  return(table(a))
}

library(kknn)
kknnearest <- kknn(Spam~., train, test, distance = 2, k = 5)

tab1<-testKNN(predict(kknnearest),0.5)
```

```
print(tab1)
print((1 -sum(diag(tab1))/(sum(tab1[,1])+sum(tab1[,2]))))


rocKalk <- function(predictions,pi = 0.5){
  a       <- data.frame(pred = as.numeric( predictions > pi ))
  a$test  <- test[,length(test)]
  confmat<-table(a)
  TPR<-sum(confmat[2,2])/sum(confmat[,2])
  FPR<-1 - sum(confmat[1,1])/sum(confmat[,1])
  return(data.frame(TPR = TPR, FPR = FPR))
}


mypi<-seq(0.05,0.95,0.05)
i <- 1
rocMyFunk <- data.frame(matrix(ncol= 2 ,nrow = length(mypi)))
for (n in mypi){
rocMyFunk[i,1]<-rocKalk(predictions = mytest$myPredict, pi=n)[[1]]
rocMyFunk[i,2]<-rocKalk(predictions = mytest$myPredict, pi=n)[[2]]
i <- i + 1

  }
rocMyFunk$X1<- as.numeric(rocMyFunk$X1)
rocMyFunk$X2<- as.numeric(rocMyFunk$X2)
colnames(rocMyFunk) <- c("TPR","FPR")


mypi<-seq(0.05,0.95,0.05)
i <- 1
rocKKNN <- data.frame(matrix(ncol= 2 ,nrow = length(mypi)))
for (n in mypi){
  rocKKNN[i,1]<-rocKalk(predictions = predict(kknnearest), pi=n)[[1]]
  rocKKNN[i,2]<-rocKalk(predictions = predict(kknnearest), pi=n)[[2]]
  i <- i + 1

}
rocKKNN$X1<- as.numeric(rocKKNN$X1)
rocKKNN$X2<- as.numeric(rocKKNN$X2)
colnames(rocKKNN) <- c("TPR","FPR")

library(ggplot2)

ggplot() + geom_line(data=rocMyFunk,aes(x = FPR, y = TPR,col="knearest")) + geom_line(data=rocKKNN,aes(


machines <- data.frame(read.csv2("machines.csv"))
machines<-as.vector(unlist(machines$Length))


explog <-function(theta,data){

  return(     length(data)*log(theta,base = exp(1)) - (theta*sum(data))    )
```

```
}


theta <- seq(0.001,5,by=0.01) #Generates a sequence of different
thetaML <- sapply( X = theta, FUN = explog, data = machines)
#apply the different thetas to the log-likelihood with the machine data-set.
plotData <- data.frame(x=theta,y=thetaML) #Put the data in a frame.


#Plotting
library(ggplot2)
p <- ggplot() + geom_line(data=plotData,aes(x=x,y=y,col="All observations")) +
                labs(title=paste("Log-likelihood function over different  theta"),
       x=expression(theta),y="ML-estimate")

a <- p + geom_segment(aes(x = 3, y = -20,
                          xend = theta[thetaML == max(thetaML)], yend = max(thetaML)),
                  arrow = arrow(length = unit(0.5, "cm"))) +
  annotate("text",x=4,y=-19,label = paste("theta",theta[thetaML == max(thetaML)]))

plot(a)


thetaML3 <- sapply( X = theta, FUN = explog, data = machines[1:6])
#Using only the 6 first observations of machines
plotData$y2 <- thetaML3

p + geom_line(data = plotData, aes(x=x,y=y2,col="6 observations"))


postBayes <- function(data,theta){
  lambda <- 10
  return(length(data) * log(theta) - theta * sum(data) +
           log(lambda) - (lambda*theta))

}

thetaBayes <- sapply(theta,FUN = postBayes,data = machines)

plotDataBayes <- data.frame(x=theta,y=thetaBayes)
B <- ggplot() + geom_line(data=plotDataBayes,aes(x=x,y=y)) +
  labs(title="Bayes-posterior",
       x=expression(theta),y="Bayes-estimate")
plot(B)

library(gridExtra)
#extract the exact theta for the  ML-estimate
maxTheta <- theta[thetaML == max(thetaML)]
# Generate 50 random variables from the exponential distrubution with
# the ML-estimate extracted
set.seed(12345)
randomExpData<- data.frame(x=rexp(50,rate = maxTheta))
```

```
#Plotting
ab<- ggplot(data = randomExpData,aes(x=x)) +
  geom_histogram(bins = 12) +
  labs(title = "Randomly drawn data from rexp")
abc<- ggplot(data =data.frame(x=machines),aes(x=x)) +
  geom_histogram(bins = 12) +
  labs(title = "The machine data-set")

 grid.arrange( ab, abc, ncol=2)
```