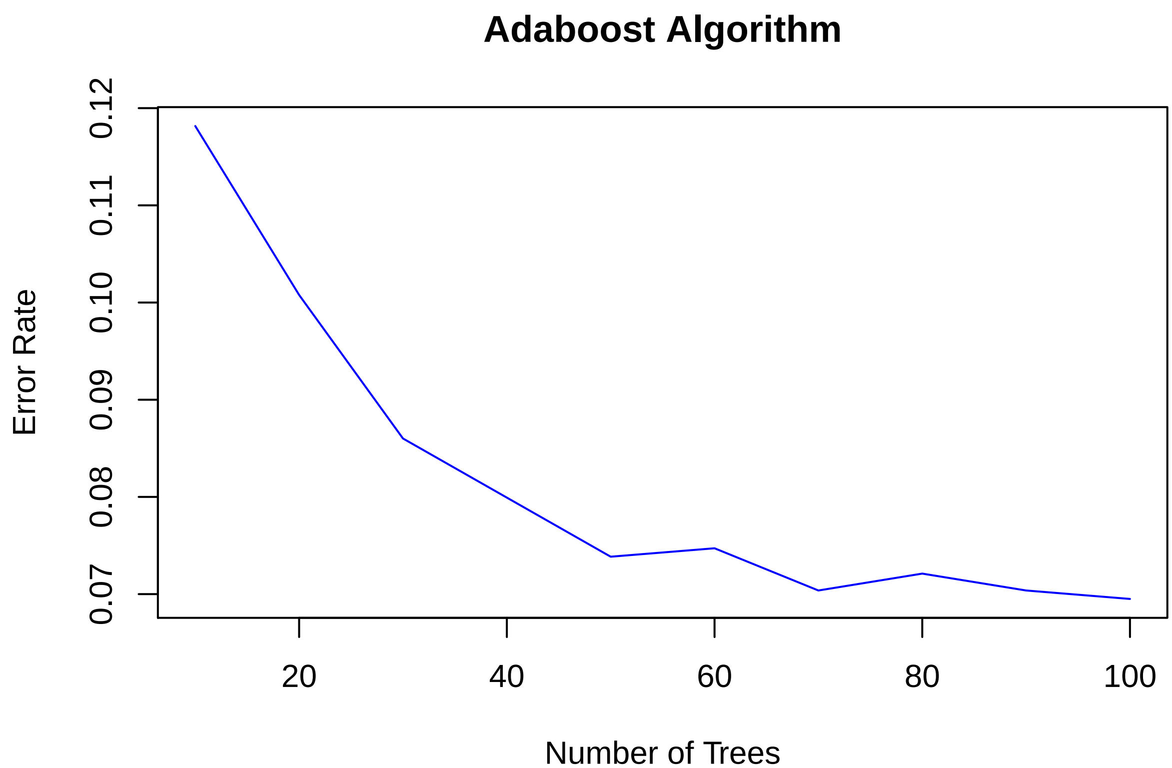
Block 2 Lab 1 Report

Mohammed Bakheet

04/12/2019

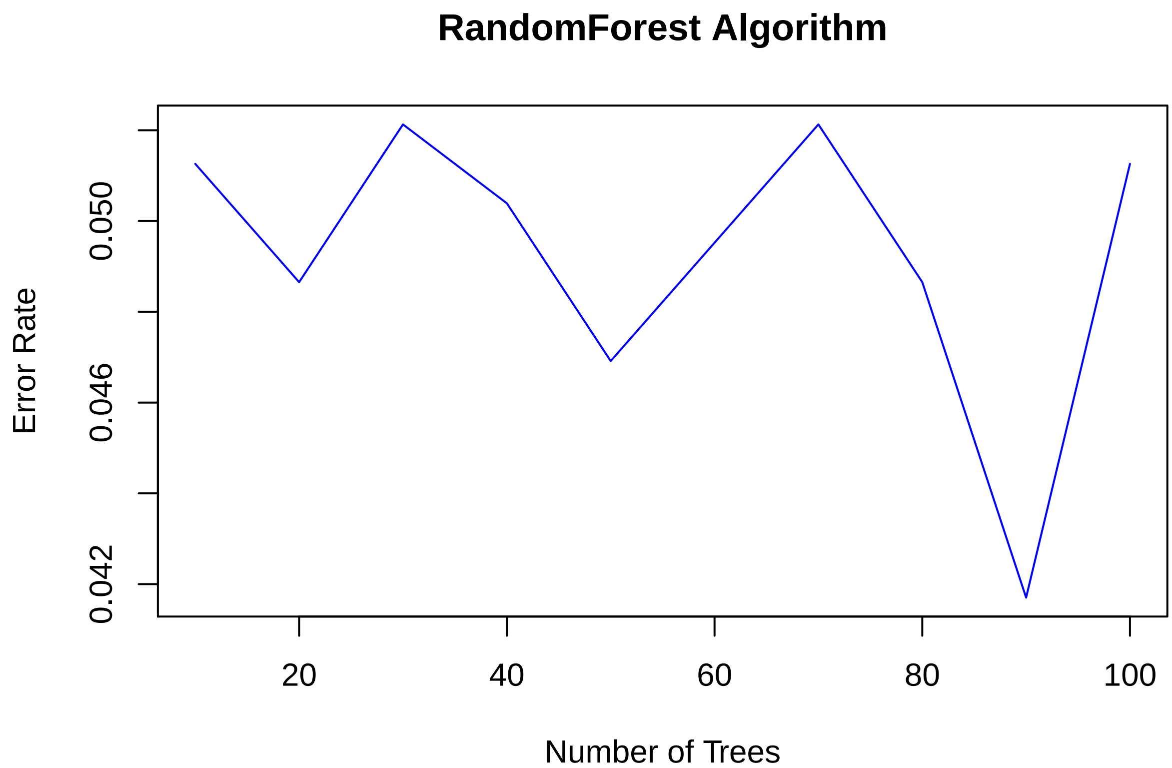
First Task (ENSEMBLE METHODS)

In this task the performance of Adaboost classification trees and random forests are evaluated on the spam data. Plots showing Adaboost and random forests errors rates are also provided.



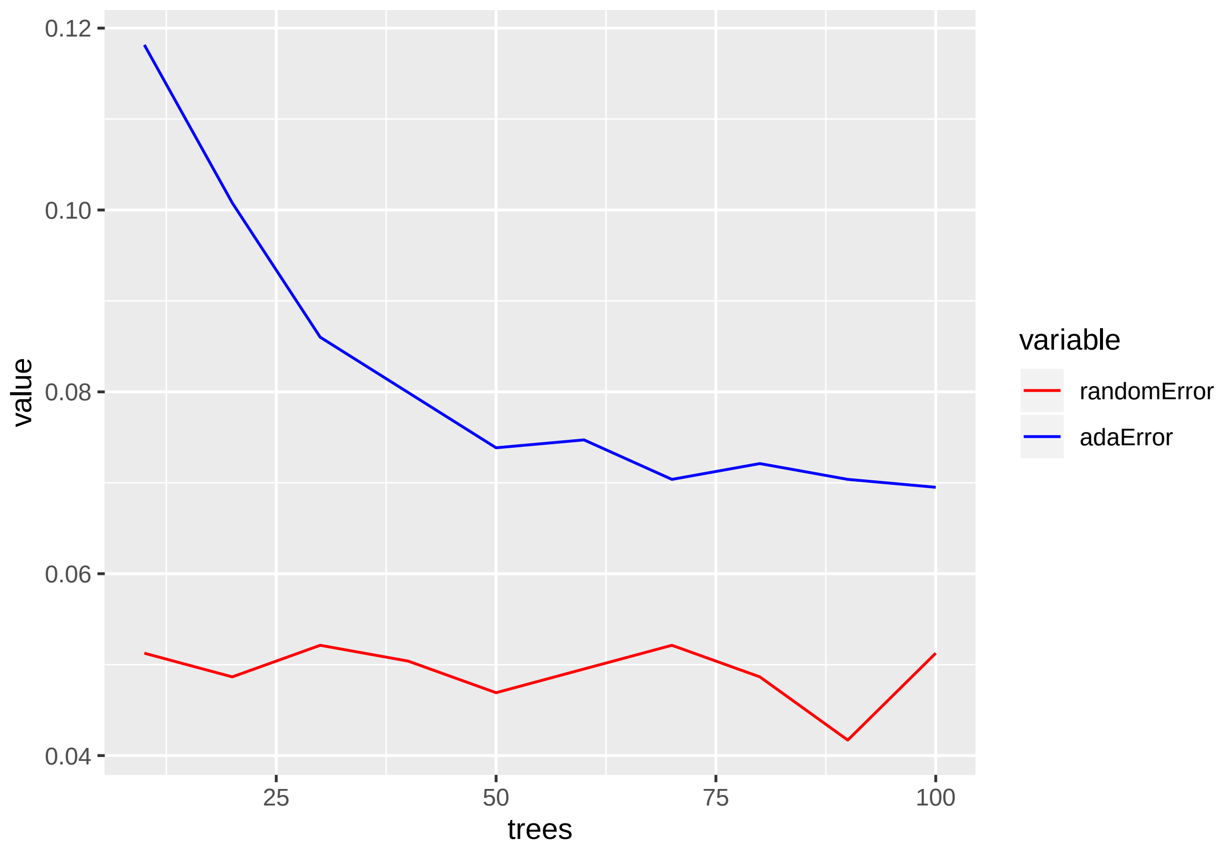

```
## integer(0)
```

#The error rate is highest when using 10 trees and it decreases as we use more trees.




```
## integer(0)
```

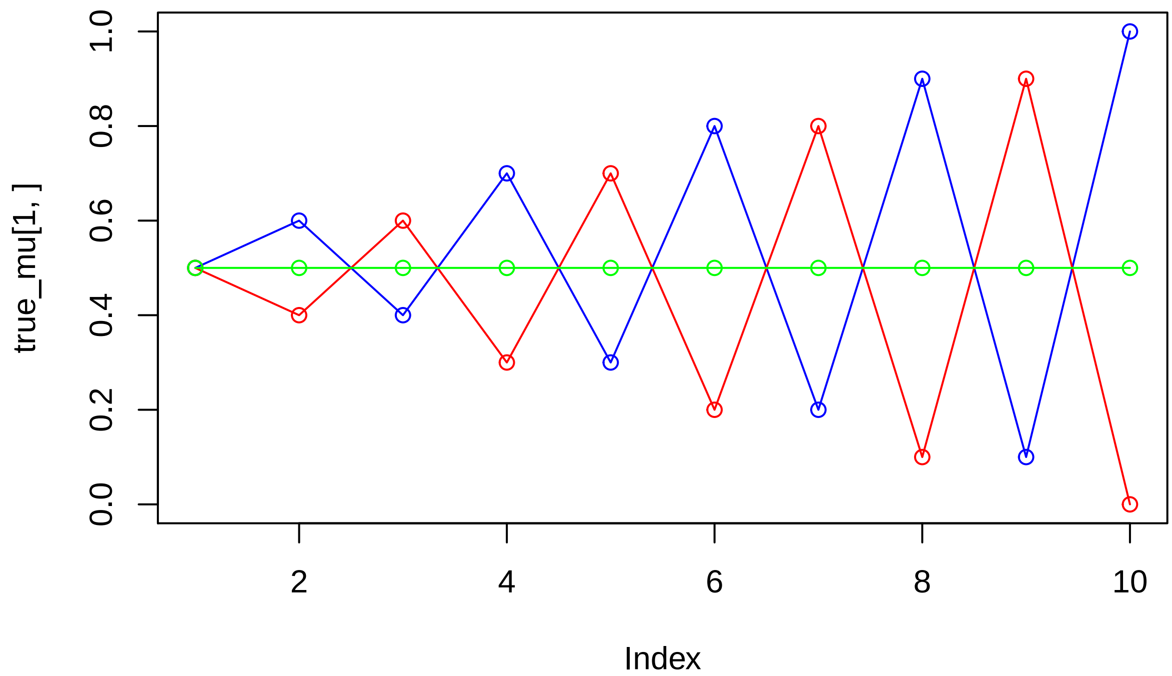
#In case of randomForest, the error rate is also highest when using 10 trees and it also decreases when using more trees.



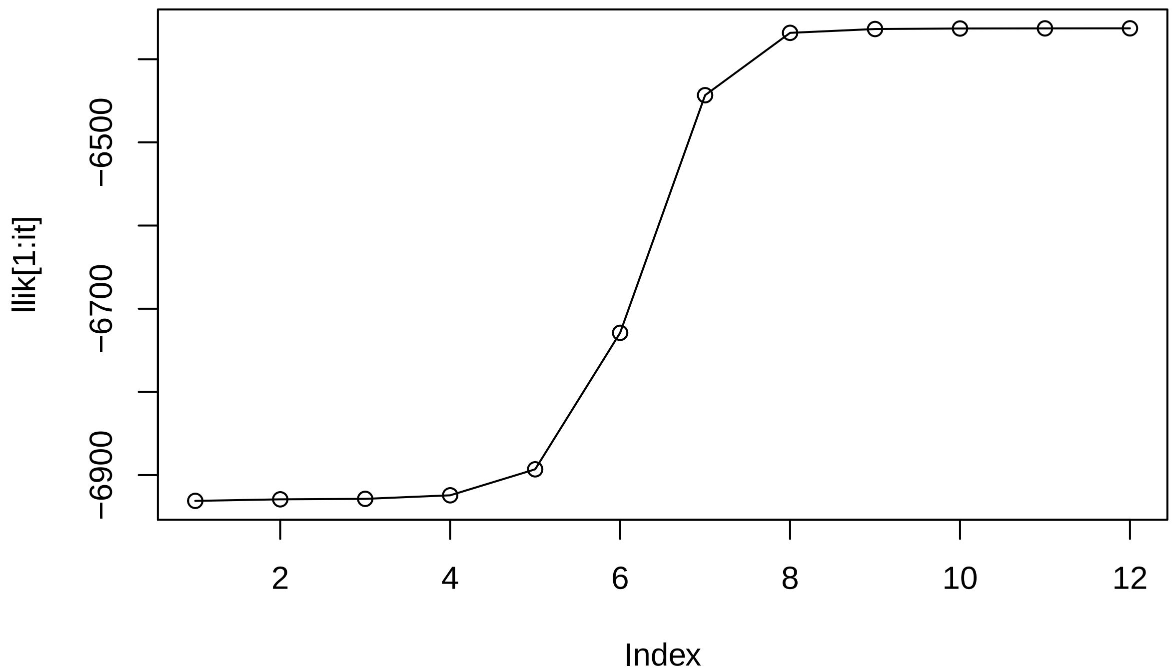
#When comparing the error rate for Random Forest and Adaboost, it's clear from the plot that Random Forests always give a lower value of error than Adaboost.

Second Task (EM Algorithm)

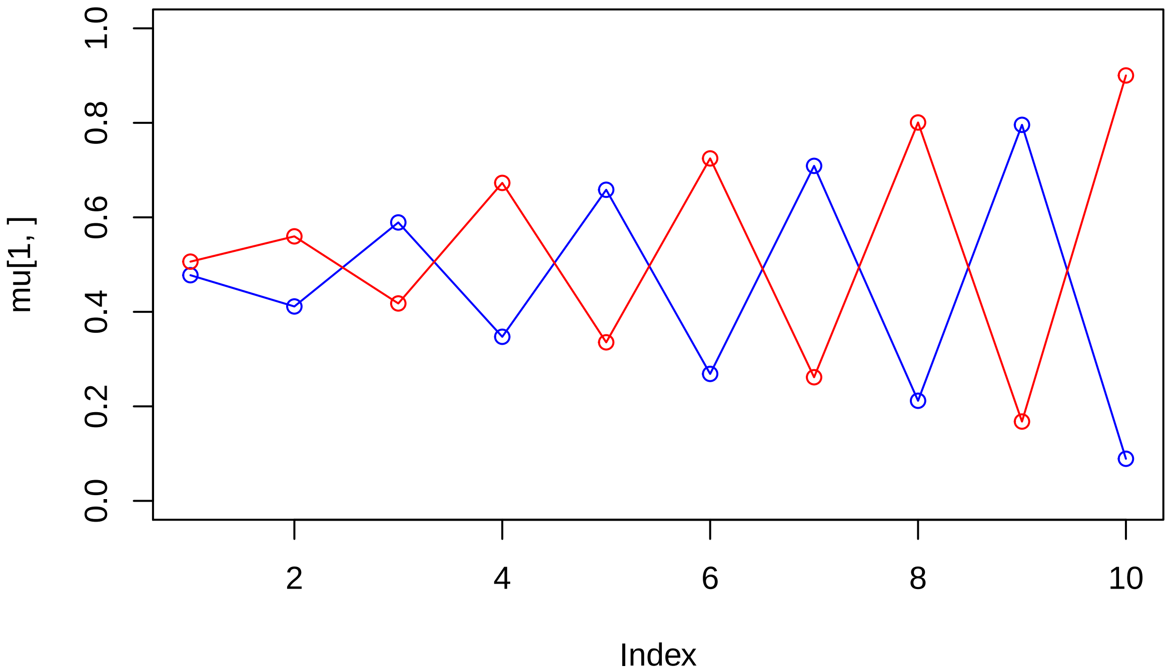
When K=2



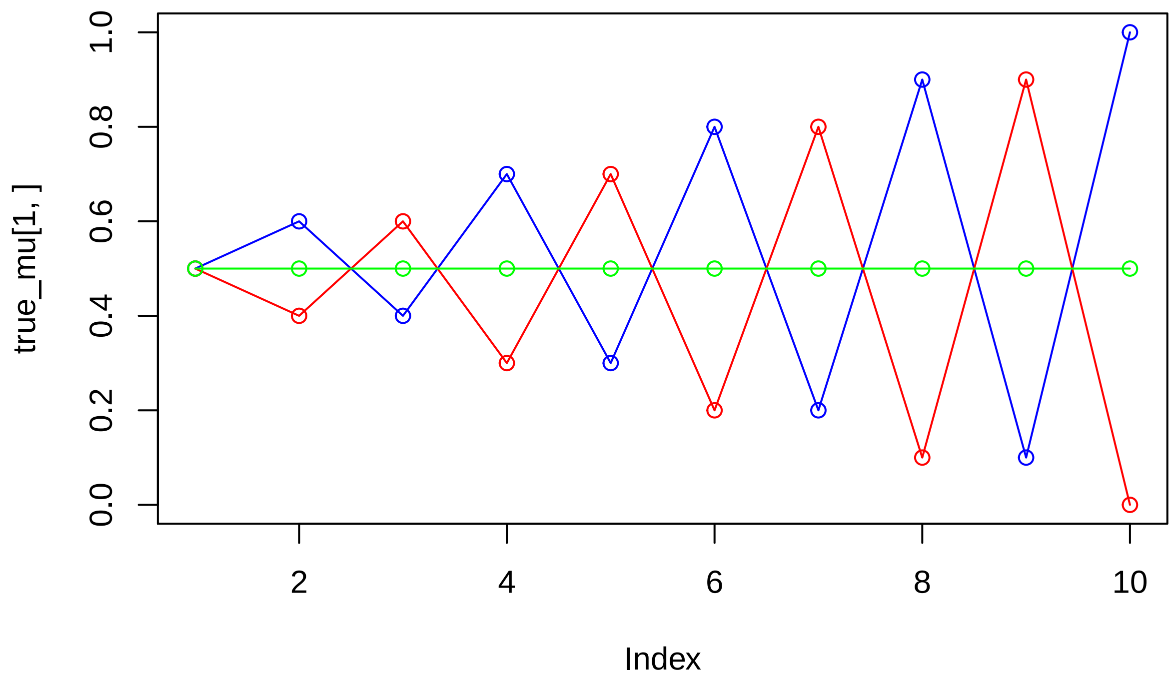

```
## iteration:  1 log likelihood:  -6930.975
## iteration:  2 log likelihood:  -6929.125
## iteration:  3 log likelihood:  -6928.562
## iteration:  4 log likelihood:  -6924.281
## iteration:  5 log likelihood:  -6893.055
## iteration:  6 log likelihood:  -6728.948
## iteration:  7 log likelihood:  -6443.28
## iteration:  8 log likelihood:  -6368.318
## iteration:  9 log likelihood:  -6363.734
## iteration:  10 log likelihood:  -6363.109
## iteration:  11 log likelihood:  -6362.947
## iteration:  12 log likelihood:  -6362.897
```




```
## [1] "co 2"
```

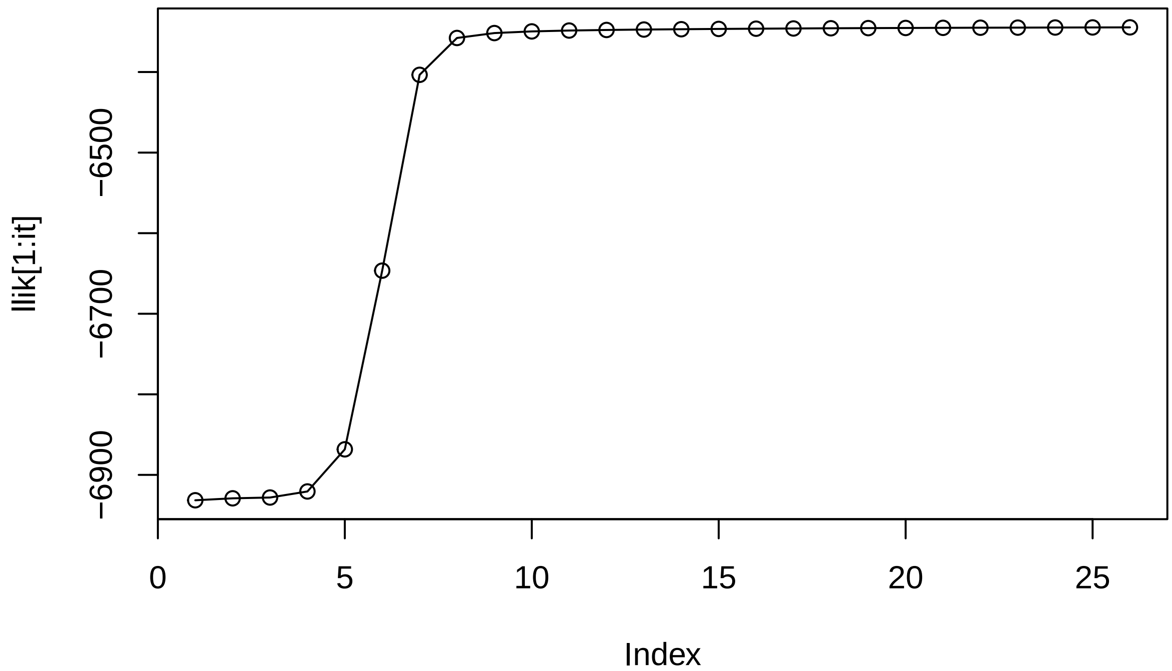

mu[1,]
0.0
0.2
0.4
0.6
0.8
1.0
2
4
6
8
10
Index

When K=3

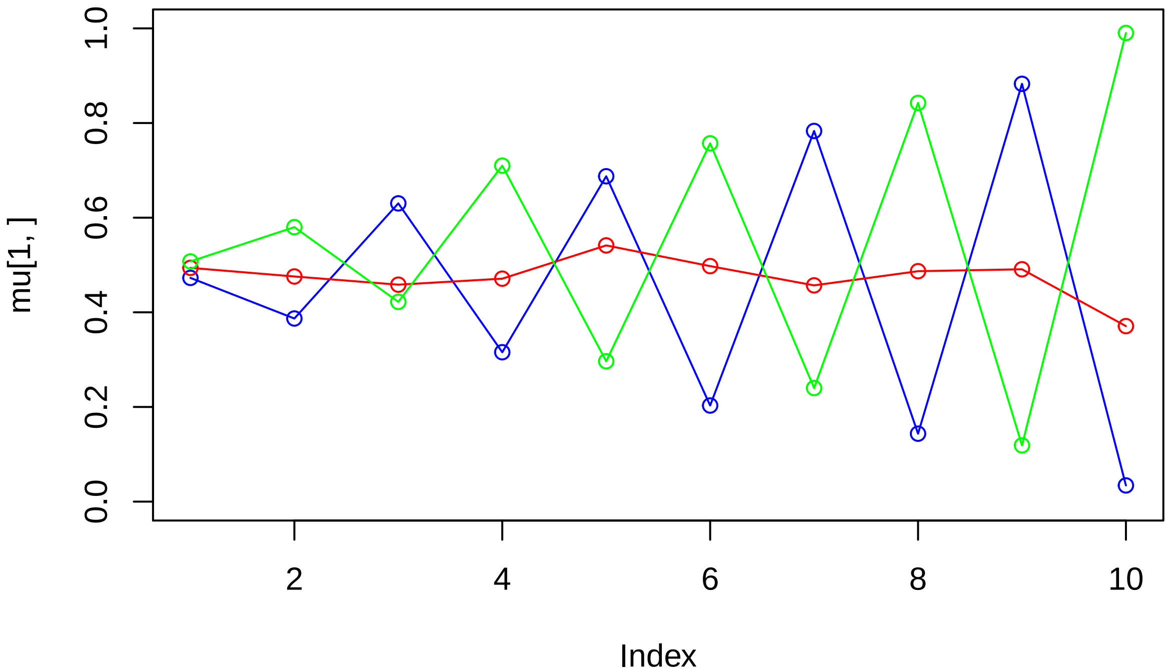



```
## iteration:  1 log likelihood:  -6931.482
## iteration:  2 log likelihood:  -6929.074
## iteration:  3 log likelihood:  -6928.081
## iteration:  4 log likelihood:  -6920.57
## iteration:  5 log likelihood:  -6868.29
## iteration:  6 log likelihood:  -6646.505
## iteration:  7 log likelihood:  -6403.476
## iteration:  8 log likelihood:  -6357.743
## iteration:  9 log likelihood:  -6351.637
## iteration:  10 log likelihood:  -6349.59
## iteration:  11 log likelihood:  -6348.513
## iteration:  12 log likelihood:  -6347.809
## iteration:  13 log likelihood:  -6347.284
## iteration:  14 log likelihood:  -6346.861
## iteration:  15 log likelihood:  -6346.506
## iteration:  16 log likelihood:  -6346.2
## iteration:  17 log likelihood:  -6345.934
## iteration:  18 log likelihood:  -6345.699
## iteration:  19 log likelihood:  -6345.492
## iteration:  20 log likelihood:  -6345.309
## iteration:  21 log likelihood:  -6345.147
## iteration:  22 log likelihood:  -6345.003
## iteration:  23 log likelihood:  -6344.875
## iteration:  24 log likelihood:  -6344.762
```

```
## iteration:  25 log likelihood:  -6344.66
## iteration:  26 log likelihood:  -6344.57
```

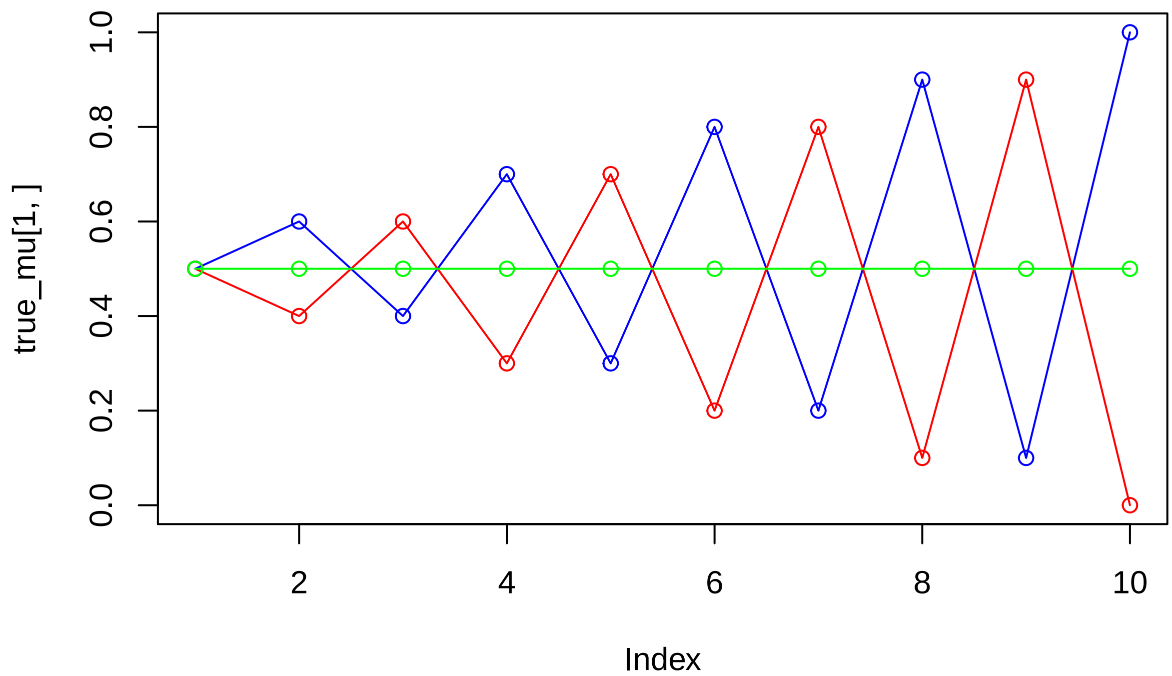



```
## [1] "co 3"
```



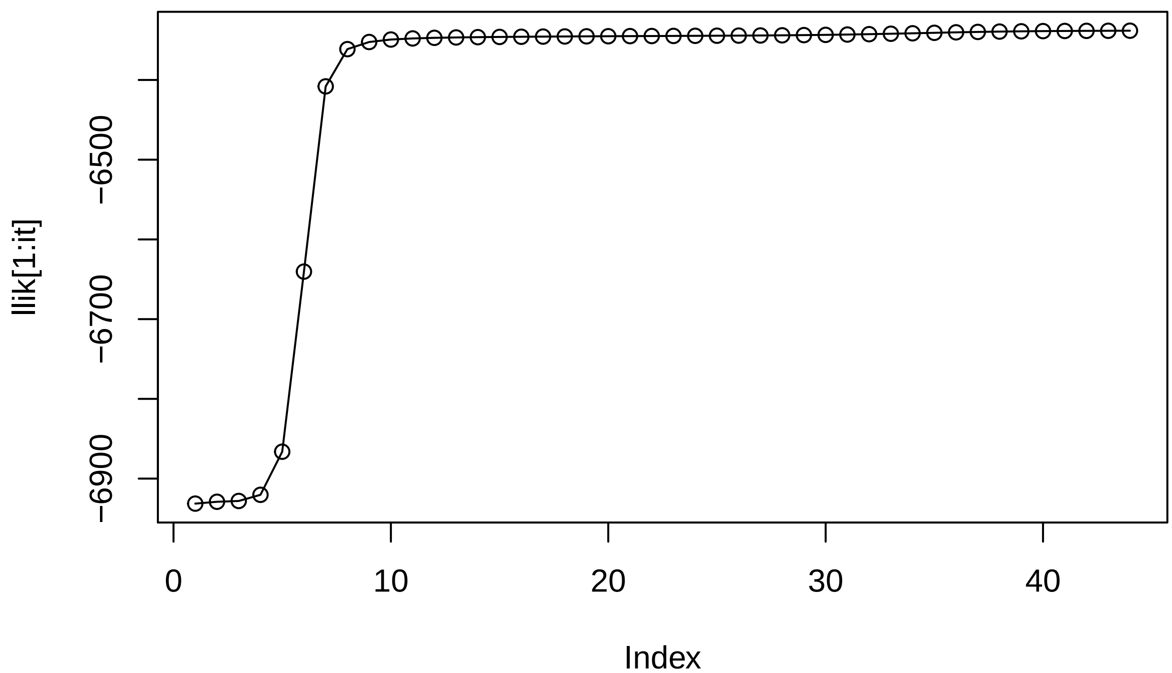
mu[1,]
0.0
0.2
0.4
0.6
0.8
1.0
2
4
6
8
10
Index

When K=4

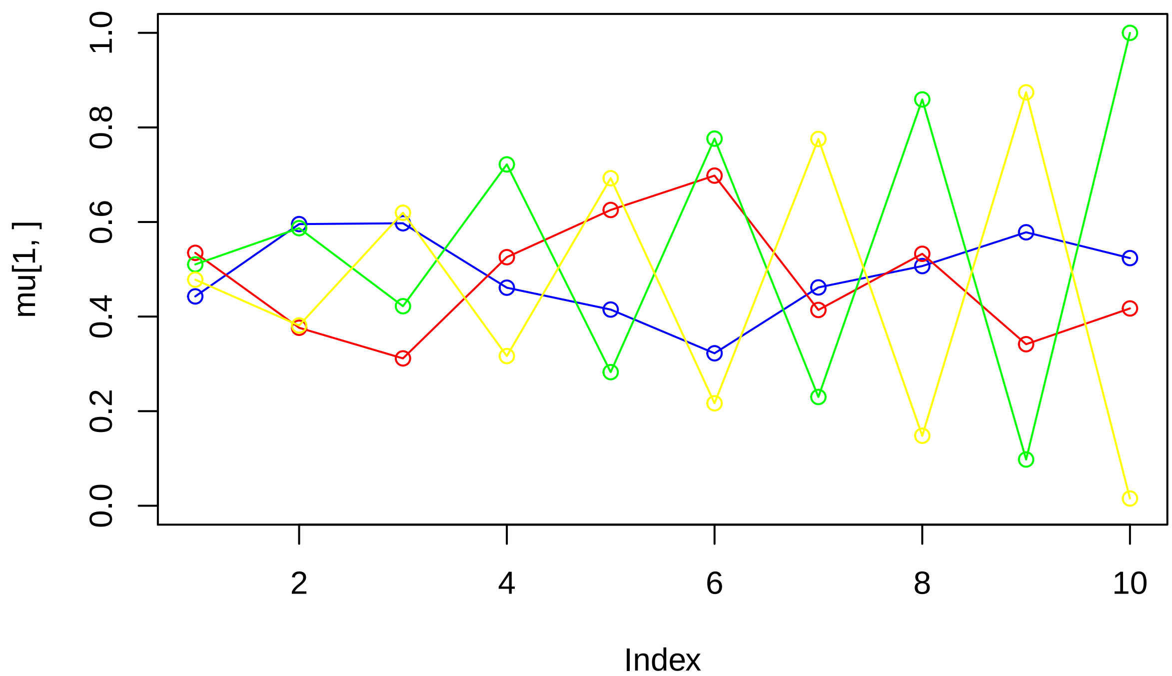



```
## iteration:  1 log likelihood:  -6931.372
## iteration:  2 log likelihood:  -6929.087
## iteration:  3 log likelihood:  -6928.057
## iteration:  4 log likelihood:  -6920.335
## iteration:  5 log likelihood:  -6866.277
## iteration:  6 log likelihood:  -6640.396
## iteration:  7 log likelihood:  -6408.058
## iteration:  8 log likelihood:  -6361.322
## iteration:  9 log likelihood:  -6352.413
## iteration:  10 log likelihood:  -6349.293
## iteration:  11 log likelihood:  -6347.902
## iteration:  12 log likelihood:  -6347.148
## iteration:  13 log likelihood:  -6346.663
## iteration:  14 log likelihood:  -6346.308
## iteration:  15 log likelihood:  -6346.028
## iteration:  16 log likelihood:  -6345.797
## iteration:  17 log likelihood:  -6345.601
## iteration:  18 log likelihood:  -6345.43
## iteration:  19 log likelihood:  -6345.279
## iteration:  20 log likelihood:  -6345.142
## iteration:  21 log likelihood:  -6345.015
## iteration:  22 log likelihood:  -6344.894
## iteration:  23 log likelihood:  -6344.775
## iteration:  24 log likelihood:  -6344.652
```

```
## iteration:  25 log likelihood:  -6344.52
## iteration:  26 log likelihood:  -6344.373
## iteration:  27 log likelihood:  -6344.2
## iteration:  28 log likelihood:  -6343.992
## iteration:  29 log likelihood:  -6343.737
## iteration:  30 log likelihood:  -6343.421
## iteration:  31 log likelihood:  -6343.033
## iteration:  32 log likelihood:  -6342.57
## iteration:  33 log likelihood:  -6342.036
## iteration:  34 log likelihood:  -6341.451
## iteration:  35 log likelihood:  -6340.849
## iteration:  36 log likelihood:  -6340.272
## iteration:  37 log likelihood:  -6339.757
## iteration:  38 log likelihood:  -6339.327
## iteration:  39 log likelihood:  -6338.988
## iteration:  40 log likelihood:  -6338.732
## iteration:  41 log likelihood:  -6338.544
## iteration:  42 log likelihood:  -6338.406
## iteration:  43 log likelihood:  -6338.304
## iteration:  44 log likelihood:  -6338.228
```




```
## [1] "co 4"
```



#The EM algorithm takes the value ot two,three, and four componenst and calculates the Bernoulli probability and the maximum likelihood, in addition to the log liklihood, until the change is less than 0.001. In each iteration the conversion rate decreases as the algorithm tries to make better classification to the points included in the matrix.
When we have three components, the mu is closest to the true mu.

Code Appendix

```
packages <- c("ggplot2", "plotly","readxl","kknn")
options(tinytex.verbose = TRUE)
knitr::opts_chunk$set(
    echo = TRUE,
    message = FALSE,
    warning = FALSE
)
packages <- c("ggplot2", "plotly","readxl","kknn")
options(tinytex.verbose = TRUE)
RNGversion("3.5.1")
library(mboost)
library(randomForest)

#Importing data from csv file
sp <- read.csv2("D:/Desktop/Machine Learning/Machine Learning/lab02/spambase.csv")
sp$Spam <- as.factor(sp$Spam)
```

```
#Dividing data into training and testing
n=dim(sp)[1]
set.seed(12345)
id=sample(1:n, floor(n*0.75))
train=sp[id,]
test=sp[-id,]
trainSequence <- seq(from = 10, to = 100, by = 10)

#------------------------------------------------------------------------
adaboostResult <- sapply(trainSequence,FUN = function(i){
  adaBoostModel <- blackboost(Spam ~ .,
                              data = train,
                              control = boost_control(mstop = i),
                              family = Binomial(type="adaboost"))
  modelTrainingAccuracy <- predict(adaBoostModel, train)
  modelTestingAccuracy <- predict(adaBoostModel,test, type = "class")
  c(length(which(modelTestingAccuracy != test$Spam))/length(test$Spam),i)
})
plot(adaboostResult[2,],adaboostResult[1,], xlab = "Number of Trees", ylab = "Error Rate", type = "l", c
#-------------------------------------------------------------------------

#Calculating for randomForest
randomforestResult <- sapply(trainSequence, function(i){
randomForestModel <- randomForest(Spam ~., data = train, ntree = i)
randomforestTrainingAccuracy <- predict(randomForestModel, train)
randomforestTestingAccuracy <- predict(randomForestModel,test, type = "class")
c(length(which(randomforestTestingAccuracy != test$Spam))/length(test$Spam),i)
})
plot(randomforestResult[2,],randomforestResult[1,], xlab = "Number of Trees", ylab = "Error Rate", type

library(reshape2)
library(ggplot2)
adaForest <- data.frame(randomError =randomforestResult[1,], adaError = adaboostResult[1,], trees = ranc
dd = melt(adaForest, id=c("trees"))

ggplot(dd) + geom_line(aes(x=trees, y=value, colour=variable)) +
  scale_colour_manual(values=c("red","blue"))
#https://stackoverflow.com/questions/10349206/add-legend-to-ggplot2-line-plot
em <- function(k_){
set.seed(1234567890)
max_it <- 100 # max number of EM iterations
min_change <- 0.1 # min change in log likelihood between two consecutive EM iterations
N=1000 # number of training points
D=10 # number of dimensions
x <- matrix(nrow=N, ncol=D) # training data
true_pi <- vector(length = 3) # true mixing coefficients
true_mu <- matrix(nrow=3, ncol=D) # true conditional distributions
true_pi=c(1/3, 1/3, 1/3)
true_mu[1,]=c(0.5,0.6,0.4,0.7,0.3,0.8,0.2,0.9,0.1,1)
true_mu[2,]=c(0.5,0.4,0.6,0.3,0.7,0.2,0.8,0.1,0.9,0)
true_mu[3,]=c(0.5,0.5,0.5,0.5,0.5,0.5,0.5,0.5,0.5,0.5)
plot(true_mu[1,], type="o", col="blue", ylim=c(0,1))
points(true_mu[2,], type="o", col="red")
```

```r
points(true_mu[3,], type="o", col="green")

# Producing the training data
for(n in 1:N) {
  k <- sample(1:3,1,prob=true_pi)
  for(d in 1:D) {
    x[n,d] <- rbinom(1,1,true_mu[k,d])
  }
}
K=k_ # number of guessed components
z <- matrix(nrow=N, ncol=K) # summal component assignments
pi <- vector(length = K) # mixing coefficients
mu <- matrix(nrow=K, ncol=D) # conditional distributions
llik <- vector(length = max_it) # log likelihood of the EM iterations
# Random initialization of the paramters
pi <- runif(K,0.49,0.51)
pi <- pi / sum(pi)
for(k in 1:K){
  mu[k,] <- runif(D,0.49,0.51)
}
pi
mu
for(it in 1:max_it) {
  Sys.sleep(0.5)
  # E-step: Computation of the summal component assignments
  #Calculating the probabilities
  for (n in 1:N) {
    summ = c()
    for (k in 1:K) {
      probability <- prod((mu[k,]^x[n,]) ,((1-mu[k,])^(1-x[n,])))
      summ = c(summ, probability)
    }

    z[n,] = pi*summ/sum(pi*summ)
  }
  # Your code here
  #Log likelihood computation.
  #Looping through components to calculate the log likelihood
    total = matrix(nrow = N, ncol = K)
    llik[it] = 0
  sapply(1:N, function(c){
    #Looping through dimensions
    sapply(1:K, function(d,c){
      total[c,d] <<- pi[d]*prod(((mu[d,]^x[c,])*((1-mu[d,])^(1-x[c,]))))
    },c=c)
    })
  llik[it]<- sum(log(rowSums(total)))

  # Your code here
  cat("iteration: ", it, "log likelihood: ", llik[it], "\n")
  flush.console()
  # Stop if the lok likelihood has not changed significantly
  if(it > 1){
```

```
    if(abs(llik[it]-llik[it-1]) < min_change){
      break
    }
    }
  # Your code here
  #M-step: ML parameter estimation from the data and summal component assignments
  # Your code here
  zcol = colSums(z)
  pi =zcol/N

#updating mu
  for (k in 1:K) {
    for (i in 1:D) {
      mu[k,i] <- sum(z[,k]*x[,i])/sum(z[,k])
    }
  }
}
pi
mu
plot(llik[1:it], type="o")
K <- as.character(K)
switch (K,
  "2"={print("co 2")
    plot(mu[1,], type="o", col="blue", ylim=c(0,1))
      points(mu[2,], type="o", col="red")},
  "3" = {print("co 3")
    plot(mu[1,], type="o", col="blue", ylim=c(0,1))
      points(mu[2,], type="o", col="red")
      points(mu[3,], type="o", col="green")},
  "4" = {print("co 4")
    plot(mu[1,], type="o", col="blue", ylim=c(0,1))
      points(mu[2,], type="o", col="red")
      points(mu[3,], type="o", col="green")
      points(mu[4,], type="o", col="yellow")}
)
}

em(2)
em(3)
em(4)
```

References:

https://stats.stackexchange.com
https://stackoverflow.com