Lab01_Block2

Sreenand.S

03/12/2019

Assignment1

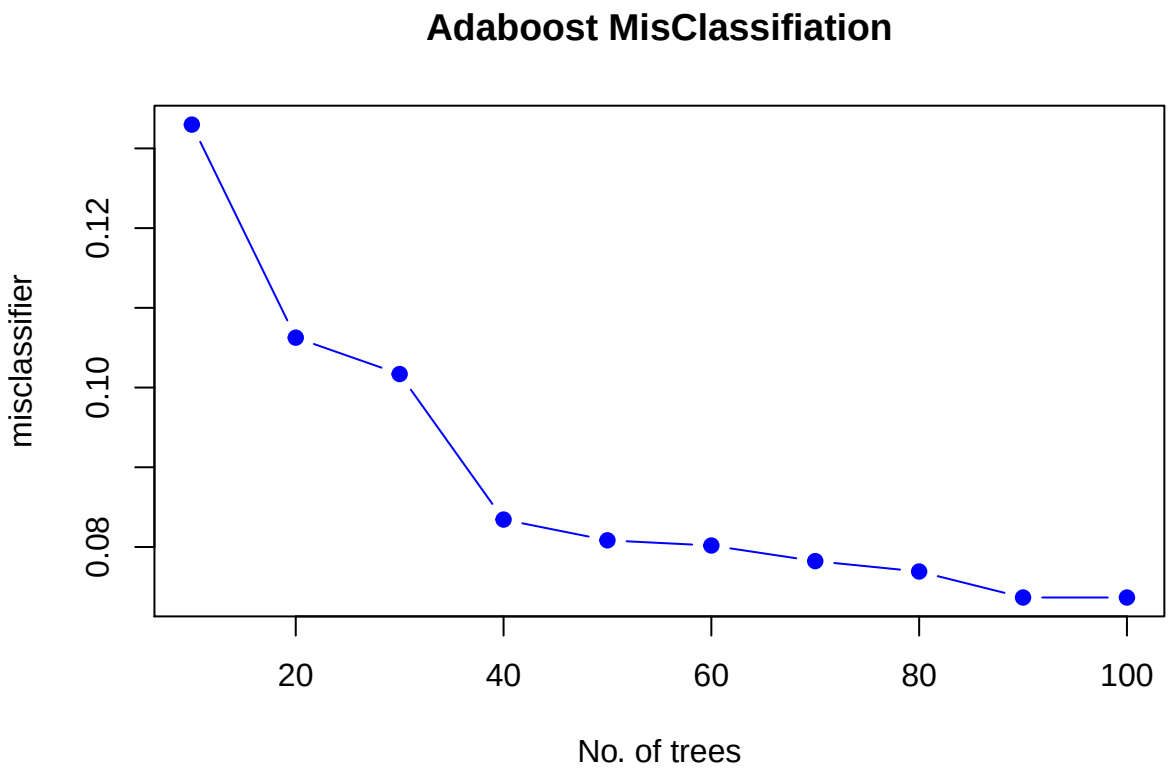
The spambases dataset is used and 2/3 percent of the data is used for training set and 1/3rd of it is used as the test set.

1.1-Adaboost Classifiation

```
## Warning in RNGkind("Mersenne-Twister", "Inversion", "Rounding"): non-uniform
## 'Rounding' sampler used
```

```
##
##     FALSE TRUE
##   0   884   38
##   1    75  537
```

```
##  [1] 0.13298566 0.10625815 0.10169492 0.08344198 0.08083442 0.08018253
##  [7] 0.07822686 0.07692308 0.07366362 0.07366362
```



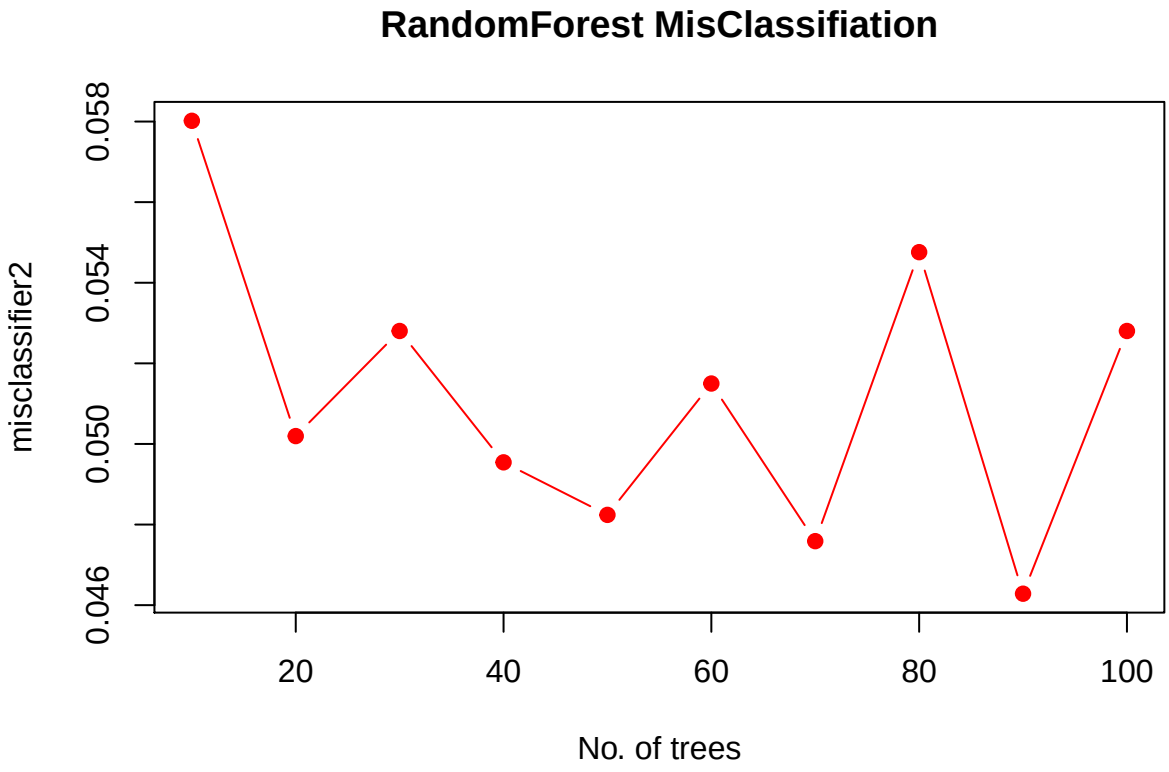
Adaboost classification is used and the no. of trees are taken as 10,20,30,40,50,60,70,80,90,100. Here the least misclassification rate is when the no. of trees is 90.

1.2 - RandomForest Classification

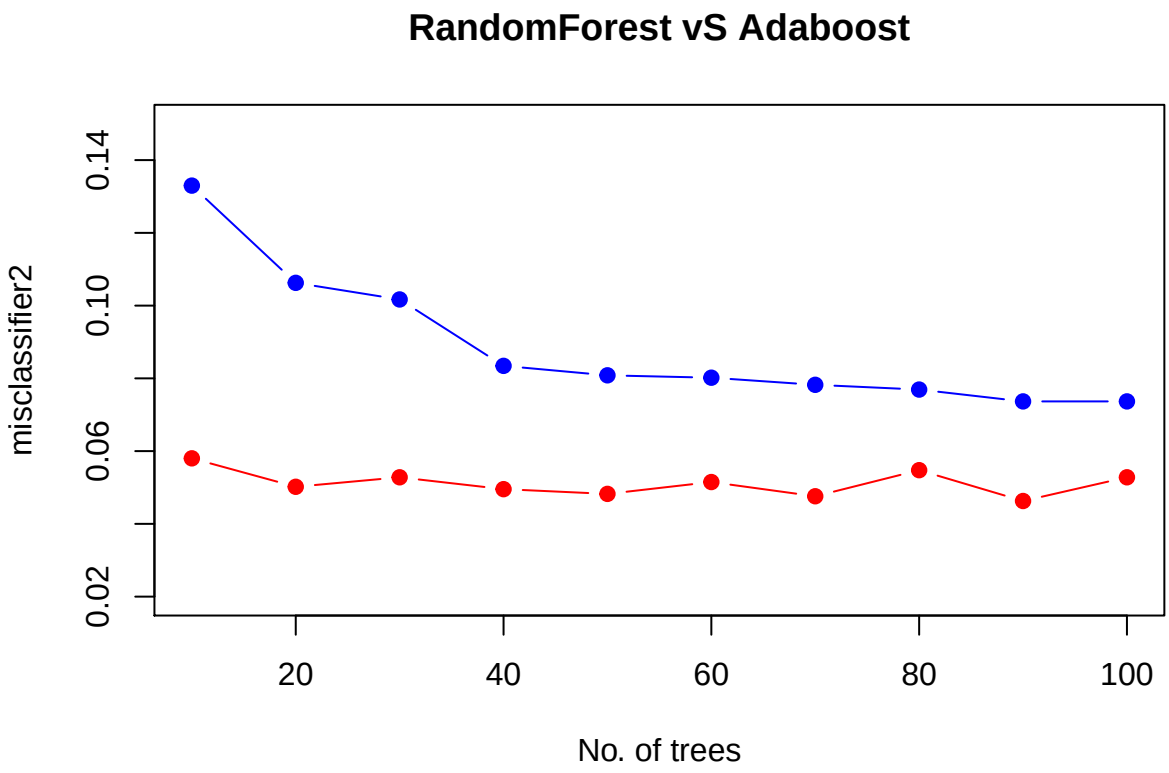
```
## Warning in RNGkind("Mersenne-Twister", "Inversion", "Rounding"): non-uniform
## 'Rounding' sampler used
```

```
##    prediction3
##       0   1
##   0 885  37
##   1  44 568
```

```
##  [1] 0.05801825 0.05019557 0.05280313 0.04954368 0.04823990 0.05149935
##  [7] 0.04758801 0.05475880 0.04628422 0.05280313
```

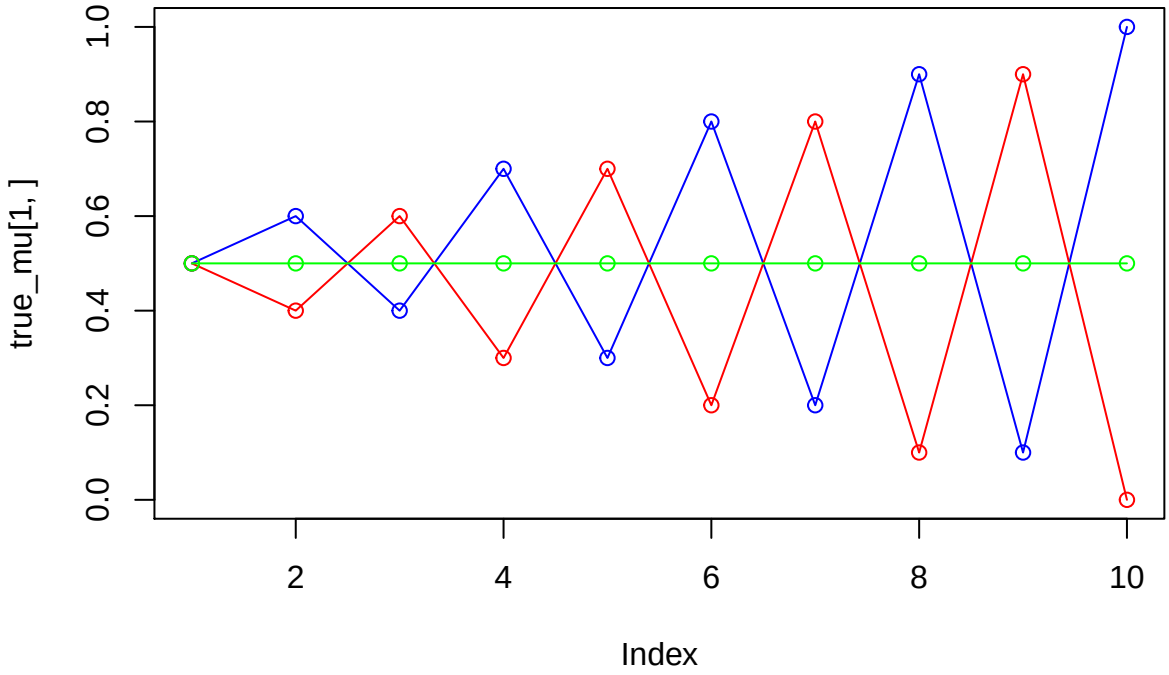


Adaboost Vs RandomForest

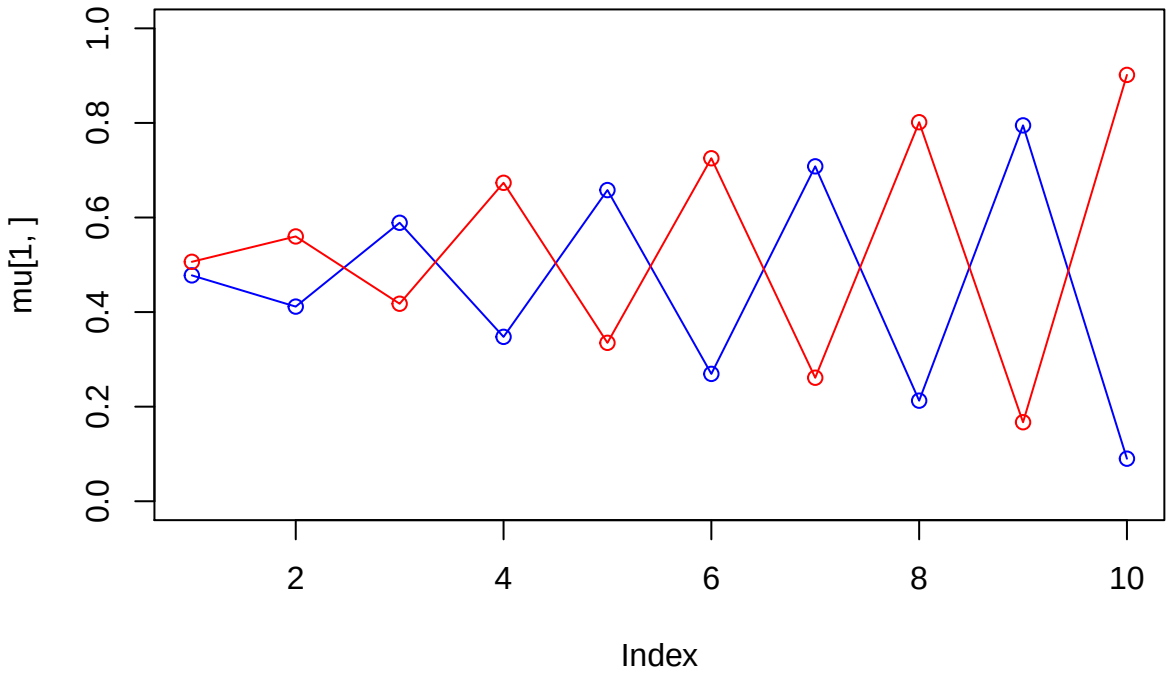


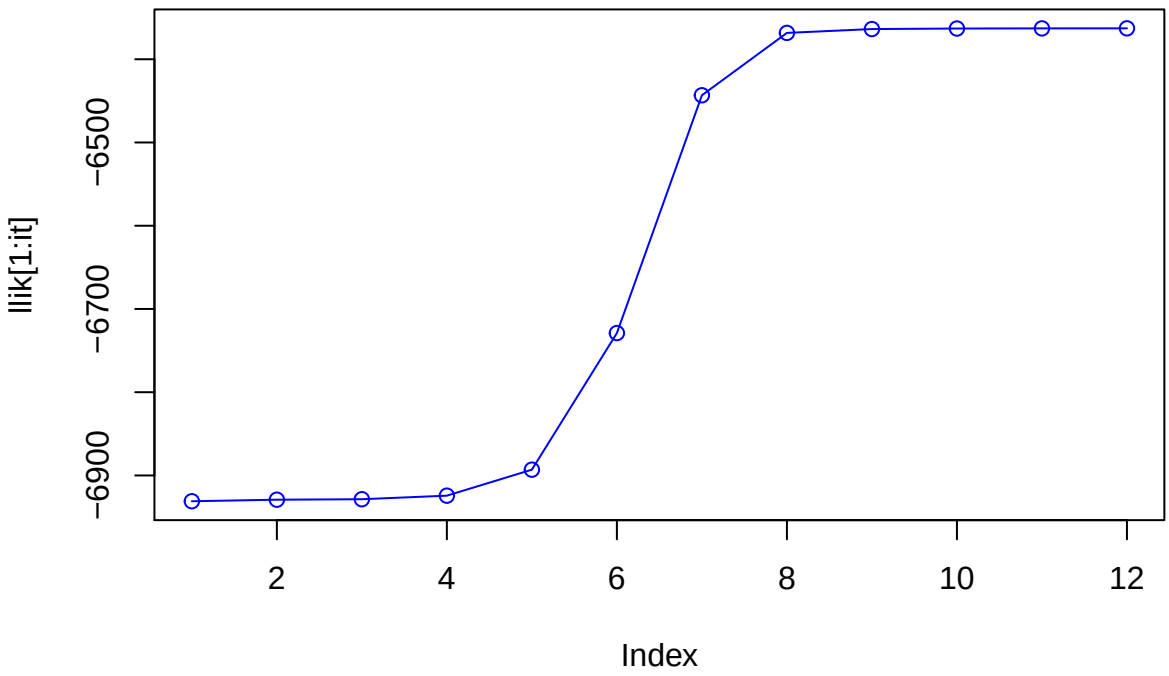
From the graph above it is observed that Randomforest Classifier has a lesser miscalculation rate than the adaboost classifier by a significant value and hence randomforest can be said to be the better classifier.

Assignment2-Mixture Models

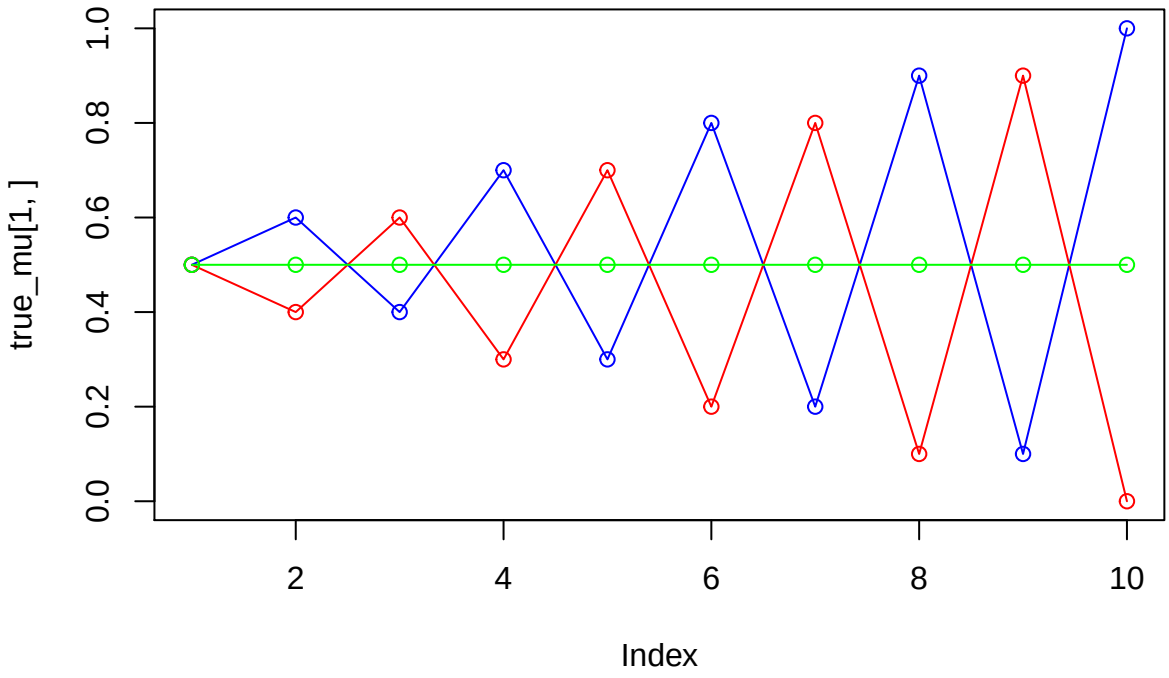



```
## [1] "Convergence Level"
## [1] 12
## [1] "Maximum Liklihood"
## [1] -6362.898
## [1] "pi Values"
## [1] 0.4984871 0.5015129
## [1] "Mu Values"
##           [,1]      [,2]      [,3]      [,4]      [,5]      [,6]      [,7]
## [1,] 0.4775923 0.4115816 0.5888739 0.3476034 0.6579934 0.2693230 0.7081895
## [2,] 0.5063208 0.5599694 0.4176442 0.6734108 0.3349839 0.7252974 0.2611631
##           [,8]      [,9]      [,10]
## [1,] 0.2127042 0.7948133 0.08999477
## [2,] 0.8015142 0.1670861 0.90154965
```

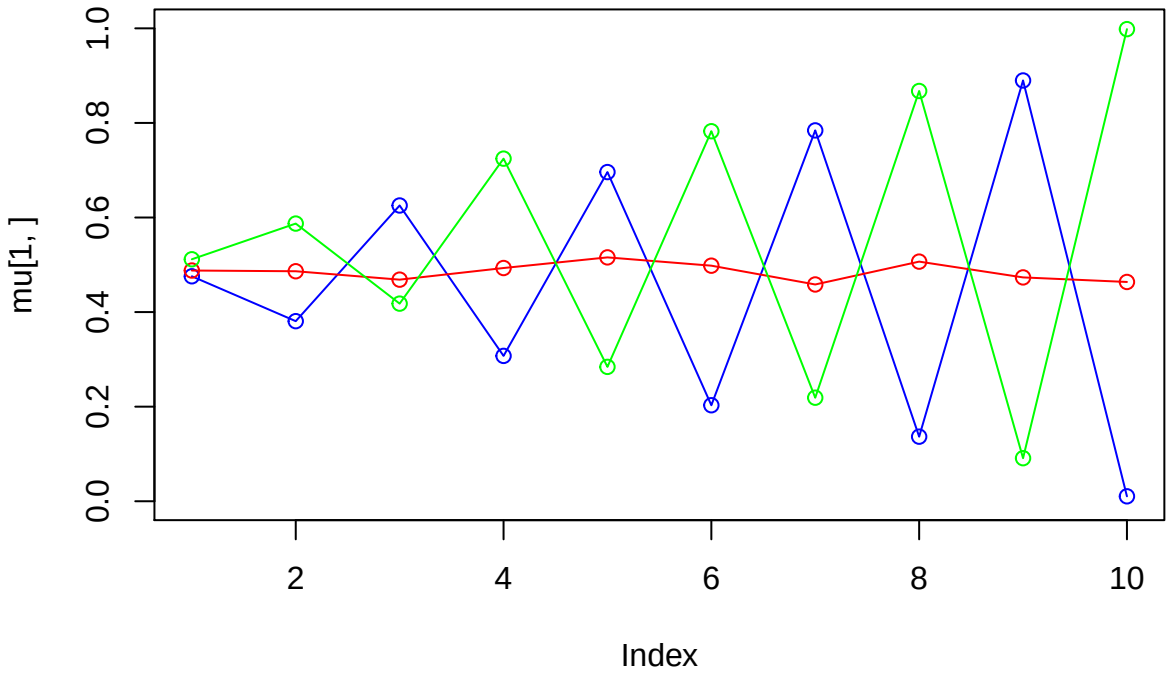

mu[1,]
0.0
0.2
0.4
0.6
0.8
1.0
2
4
6
8
10
Index

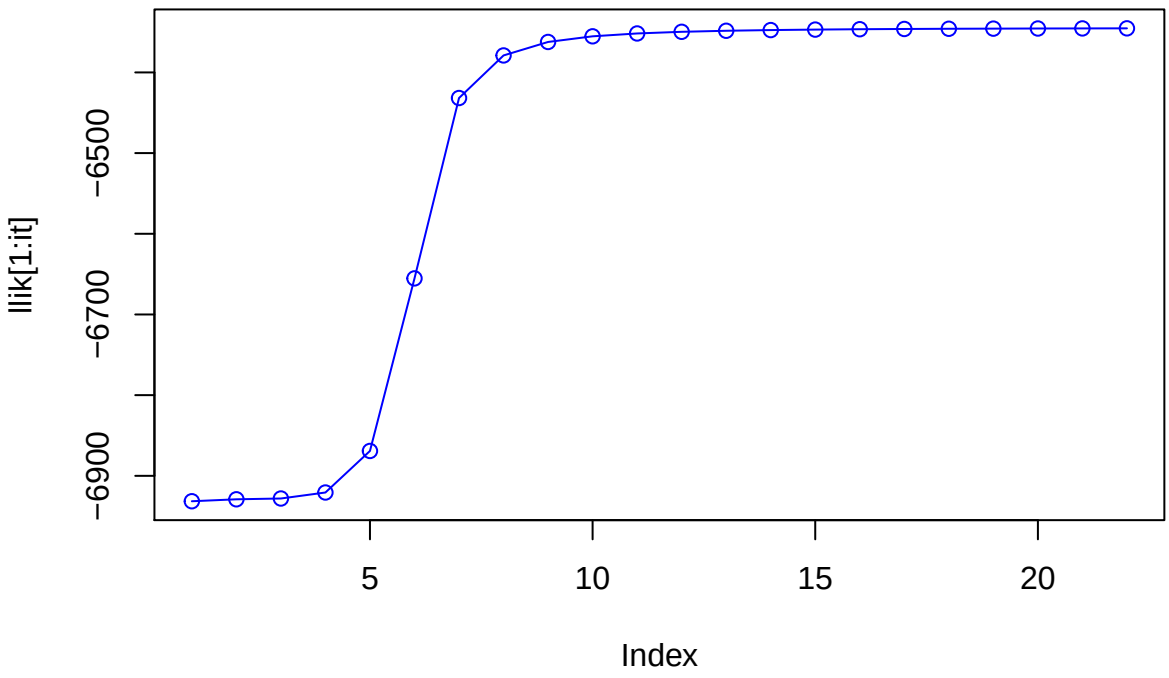


Here when the number of components are only 2 then the liklihood converges after 12 iterations and the maximum liklihood is observed to be -6362.898.

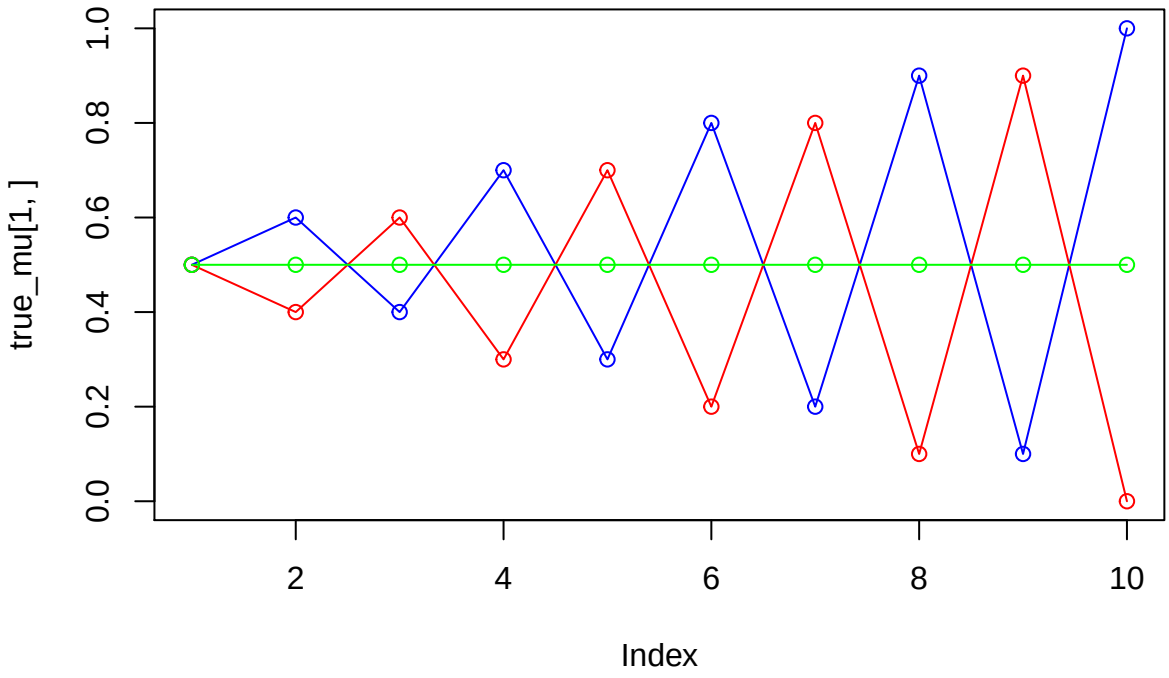



```
## [1] "Convergence Level"
## [1] 22
## [1] "Maximum Liklihood"
## [1] -6345.393
## [1] "pi Values"
## [1] 0.3301220 0.3276797 0.3421983
## [1] "Mu Values"
##           [,1]      [,2]      [,3]      [,4]      [,5]      [,6]      [,7]
## [1,] 0.4756868 0.3807664 0.6254036 0.3073594 0.6960711 0.2029757 0.7841952
## [2,] 0.4878853 0.4863472 0.4685707 0.4932018 0.5156753 0.4981670 0.4583882
## [3,] 0.5116776 0.5871875 0.4178847 0.7244971 0.2841490 0.7824529 0.2189240
##           [,8]       [,9]      [,10]
## [1,] 0.1365602 0.88976292 0.01039218
## [2,] 0.5067311 0.47335207 0.46369297
## [3,] 0.8675466 0.09106362 0.99832918
```

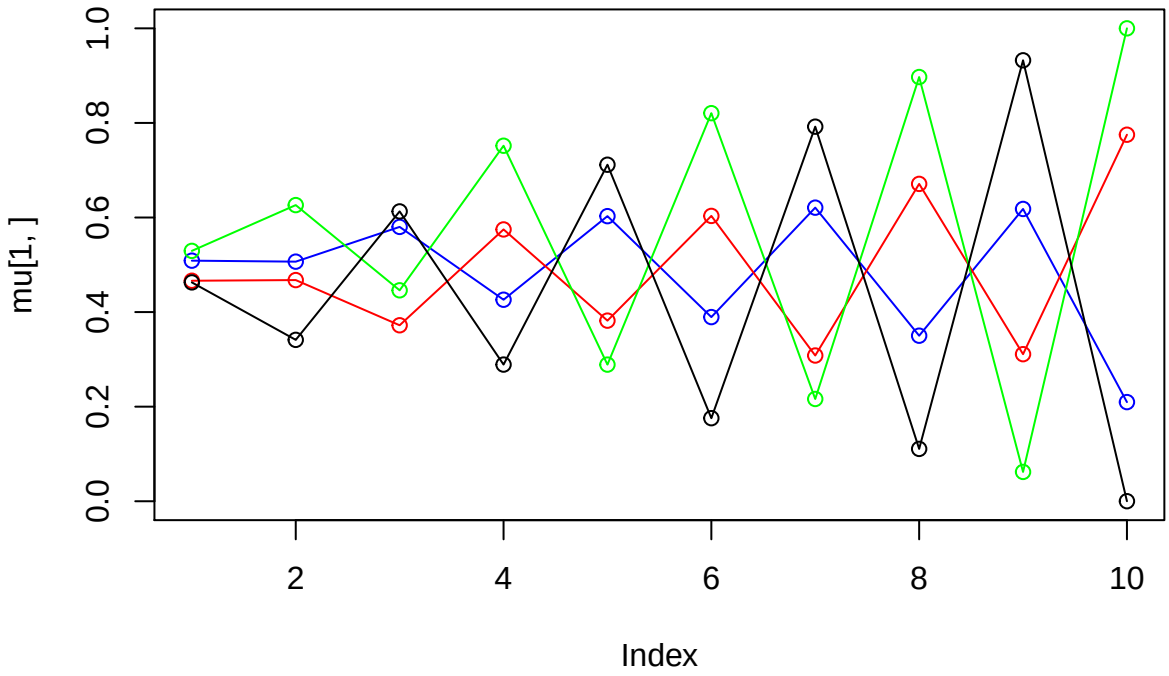

mu[1,]
0.0
0.2
0.4
0.6
0.8
1.0
2
4
6
8
10
Index

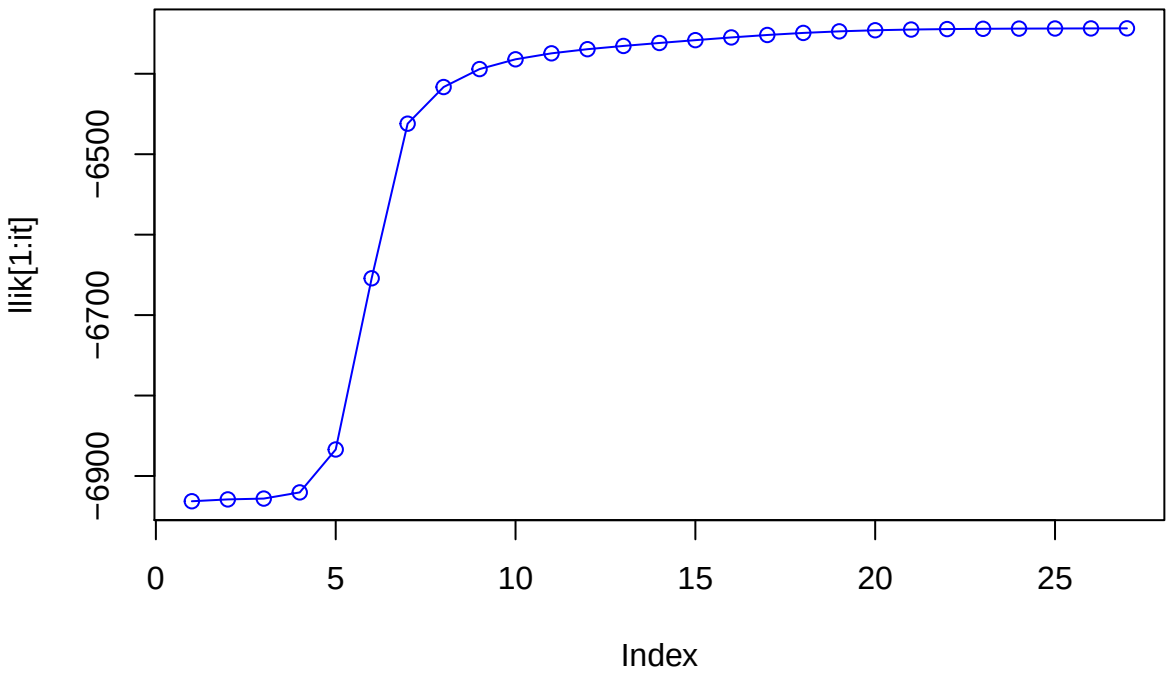


When the number of components is 3 then the maximum liklihood has increased from when K was only 2. The liklihood converges after 22 iterations and the maximum liklihood here is -6345.393 which is a lot higher than the previous step which showed -6362.This looks very similar to the given model.




```
## [1] "Convergence Level"
## [1] 27
## [1] "Maximum Liklihood"
## [1] -6343.567
## [1] "pi Values"
## [1] 0.2501201 0.2469146 0.2531561 0.2498092
## [1] "Mu Values"
##           [,1]      [,2]      [,3]      [,4]      [,5]      [,6]      [,7]
## [1,] 0.5087419 0.5066354 0.5801461 0.4261089 0.6028506 0.3893984 0.6208431
## [2,] 0.4662542 0.4677062 0.3721016 0.5747524 0.3819012 0.6033428 0.3081692
## [3,] 0.5294206 0.6262544 0.4460304 0.7517667 0.2889553 0.8206528 0.2162520
## [4,] 0.4627628 0.3412873 0.6128723 0.2889906 0.7116117 0.1756391 0.7921151
##           [,8]       [,9]        [,10]
## [1,] 0.3500757 0.61805851 0.2096617802
## [2,] 0.6710697 0.31098481 0.7750684454
## [3,] 0.8970426 0.06200737 0.9999910780
## [4,] 0.1106859 0.93241912 0.0001195492
```


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



When the no. of components are 4 then the Maximum liklihood is even more but it is only slightly grater hence 3 components seem to be more ideal no. of components to be taken for the distributions. Here the liklihood converges after 27 iterations and the maximum liklihood is -6343 whichshows that there has only been a slight increase in the maximum liklihood. Hence considering all the cases where K=2,3,4 , K=3 appears to be the ideal number of components.

Appendix

```
RNGversion('3.5.1')
knitr::opts_chunk$set(echo = TRUE)
library(caret)
library(adabag)
library(mboost)
library(randomForest)
sp <- read.csv2("~/Machine Learning/Lab1-Block2/spambase.csv")
 sp$Spam <- as.factor(sp$Spam)
n = dim(sp)[1]
set.seed(12345)
id = sample(1:n, floor(n*2/3))
train = sp[id,]
test = sp[-id,]
k<-1
misclassifier<-numeric(10)
misclassifier2<-numeric(10)
skip<-seq(10,100,10)
```

```
RNGversion('3.5.1')
for (i in skip) {
  model<-blackboost(Spam~.,data = train,weights = NULL,na.action = na.pass,family = AdaExp(),control = b
  prediction<-predict.mboost(object = model, newdata = test)
  confusionmatrix<-table(test$Spam,prediction>0)

  misclassifier[i/10]<-1-(sum(diag(confusionmatrix))/sum(confusionmatrix))

  }
confusionmatrix
misclassifier
plot(skip,misclassifier,type = "b",pch = 19, col = "blue",main = "Adaboost MisClassifiation",xlab = "No

RNGversion('3.5.1')
set.seed(12345)
for (i in skip) {
  model3<-randomForest(Spam~.,data = train,ntree= i)
  prediction3<-predict(object = model3, newdata = test)
  confusionmatrix3<-table(test$Spam,prediction3)
  confusionmatrix3
  misclassifier2[i/10]<-1-(sum(diag(confusionmatrix3))/sum(confusionmatrix3))

}
confusionmatrix3
misclassifier2
plot(skip,misclassifier2,type = "b",col = "red",pch = 19,main = "RandomForest MisClassifiation",xlab =



plot(skip,misclassifier2,type = "b",col = "red",pch = 19,main = "RandomForest vS Adaboost",xlab = "No.
points(skip,misclassifier,type = "b",pch = 19, col = "blue")
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")
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])
  }
}
```

```r
#for k=2
K=2 # number of guessed components
z <- matrix(nrow=N, ncol=K) # fractional 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)
}

for(it in 1:max_it)
{
  probability_x <- exp(x %*% log(t(mu)) + (1 - x) %*% log(1 - t(mu)))

  new_pi <- matrix(pi,ncol = K, nrow = N, byrow = TRUE)
  prob_pi <- probability_x * pi

  prob_z <- prob_pi / rowSums(prob_pi)

  llik[it] <- sum(log(rowSums(prob_pi)))
  if((llik[it]-llik[it-1]<= min_change) && (it >1))
   {
     break
  }

    pi <- colSums(prob_z)/N
    mu <- (t(prob_z) %*% x)/colSums(prob_z)
}

print("Convergence Level")
print(it)
print("Maximum Liklihood")
print(max(llik[1:it]))
print("pi Values")
print(pi)
print("Mu Values")
print(mu)

 plot(mu[1,], type="o", col="blue", ylim=c(0,1))
 points(mu[2,], type="o", col="red")
plot(llik[1:it],type ='o', col = "blue")
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")
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])
  }
}
#for k=2
K=3 # number of guessed components
z <- matrix(nrow=N, ncol=K) # fractional 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)
}

for(it in 1:max_it)
{
  probability_x <- exp(x %*% log(t(mu)) + (1 - x) %*% log(1 - t(mu)))

  new_pi <- matrix(pi,ncol = K, nrow = N, byrow = TRUE)
  prob_pi <- probability_x * pi

  prob_z <- prob_pi / rowSums(prob_pi)

  llik[it] <- sum(log(rowSums(prob_pi)))
  if((llik[it]-llik[it-1]<= min_change) && (it >1))
   {
     break
  }

    pi <- colSums(prob_z)/N
    mu <- (t(prob_z) %*% x)/colSums(prob_z)
}

print("Convergence Level")
print(it)
print("Maximum Liklihood")
print(max(llik[1:it]))
print("pi Values")
print(pi)
print("Mu Values")
print(mu)
```

```
 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")
plot(llik[1:it],type ='o', col = "blue")
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")
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])
  }
}
#for k=2
K=4 # number of guessed components
z <- matrix(nrow=N, ncol=K) # fractional 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)
}

for(it in 1:max_it)
{
  probability_x <- exp(x %*% log(t(mu)) + (1 - x) %*% log(1 - t(mu)))

  new_pi <- matrix(pi,ncol = K, nrow = N, byrow = TRUE)
  prob_pi <- probability_x * pi

  prob_z <- prob_pi / rowSums(prob_pi)

  llik[it] <- sum(log(rowSums(prob_pi)))
  if((llik[it]-llik[it-1]<= min_change) && (it >1))
   {
     break
  }
```

```
    pi <- colSums(prob_z)/N
    mu <- (t(prob_z) %*% x)/colSums(prob_z)
}

print("Convergence Level")
print(it)
print("Maximum Liklihood")
print(max(llik[1:it]))
print("pi Values")
print(pi)
print("Mu Values")
print(mu)

 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="black")
plot(llik[1:it],type ='o', col = "blue")
```