Analysis

```
## Loading required package: rJava
```

```
## Loading required package: xlsxjars
```

```
maindata <- read.xlsx("C:/Users/jaceline.preval/Desktop/717/BreastTissuedata.xl
sx", 1)

#Overview of data.
head(maindata)
```

```
##   Case.. Class       I0     PA500        HFS       DA      Area      ADA
## 1      1     1 524.7941 0.1874484 0.03211406 228.8002  6843.598 29.91080
## 2      2     1 330.0000 0.2268928 0.26529005 121.1542  3163.239 26.10920
## 3      3     1 551.8793 0.2324779 0.06352998 264.8049 11888.392 44.89490
## 4      4     1 380.0000 0.2408554 0.28623400 137.6401  5402.171 39.24852
## 5      5     1 362.8313 0.2007129 0.24434610 124.9126  3290.462 26.34213
## 6      6     1 389.8730 0.1500983 0.09773844 118.6258  2475.557 20.86862
##      MaxIP        DR        P
## 1 60.20488 220.73721 556.8283
## 2 69.71736  99.08496 400.2258
## 3 77.79330 253.78530 656.7694
## 4 88.75845 105.19857 493.7018
## 5 69.38939 103.86655 424.7965
## 6 49.75715 107.68616 429.3858
```

```
#Overall labels
Y2 <- as.matrix(maindata[,2])


#Overall data
Y1 <- as.matrix(maindata[,3:11])

#simple analysis
summary(Y1)
```

```
##        IO              PA500              HFS                  DA
##  Min.   : 103.0   Min.   :0.01239   Min.   :-0.06632   Min.   :  19.65
##  1st Qu.: 250.0   1st Qu.:0.06741   1st Qu.: 0.04398   1st Qu.:  53.85
##  Median : 384.9   Median :0.10542   Median : 0.08657   Median : 120.78
##  Mean   : 784.3   Mean   :0.12013   Mean   : 0.11469   Mean   : 190.57
##  3rd Qu.:1488.0   3rd Qu.:0.16960   3rd Qu.: 0.16650   3rd Qu.: 255.33
##  Max.   :2800.0   Max.   :0.35832   Max.   : 0.46775   Max.   :1063.44
##       Area                ADA               MaxIP              DR
##  Min.   :    70.43   Min.   :  1.596   Min.   :  7.969   Min.   : -9.258
##  1st Qu.:   409.65   1st Qu.:  8.180   1st Qu.: 26.894   1st Qu.: 41.781
##  Median :  2219.58   Median : 16.134   Median : 44.216   Median : 97.833
##  Mean   :  7335.16   Mean   : 23.474   Mean   : 75.381   Mean   :166.711
##  3rd Qu.:  7615.20   3rd Qu.: 30.953   3rd Qu.: 83.672   3rd Qu.:232.990
##  Max.   :174480.48   Max.   :164.072   Max.   :436.100   Max.   :977.552
##        P
##  Min.   : 125.0
##  1st Qu.: 270.2
##  Median : 454.1
##  Mean   : 810.6
##  3rd Qu.:1301.6
##  Max.   :2896.6
```

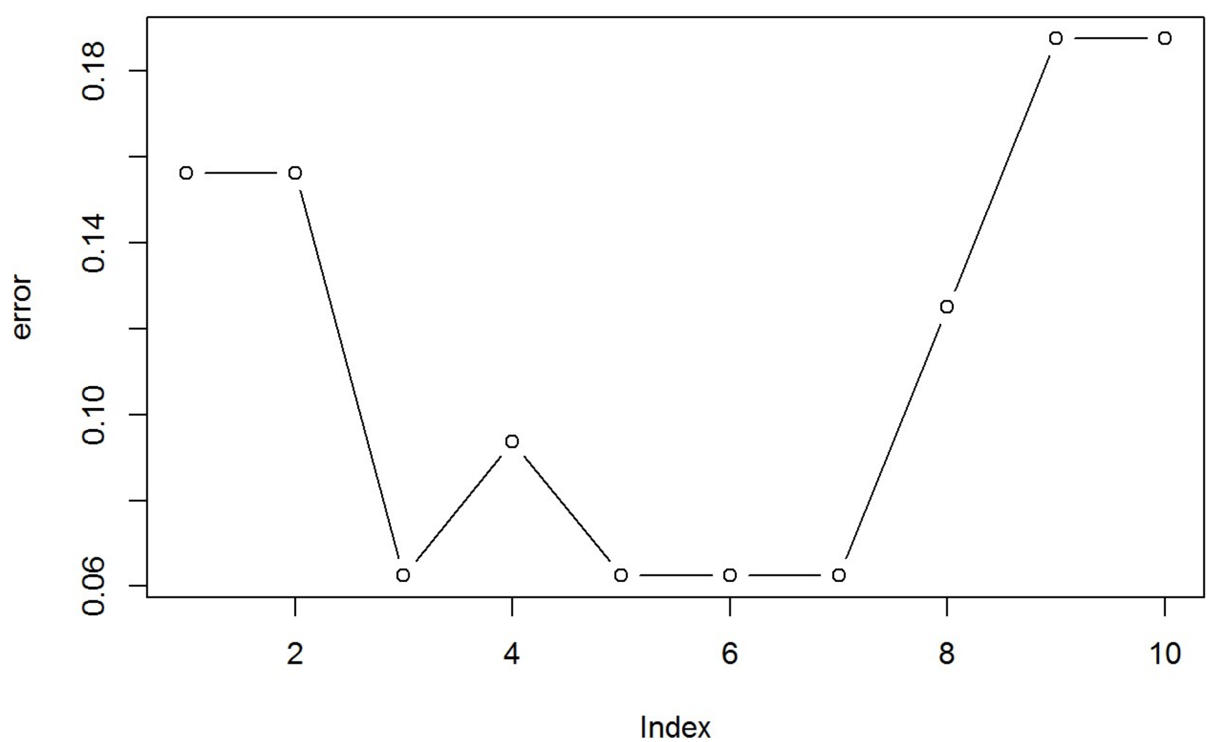
First Step - Classification

```
##training data
data <- maindata[,2:11]

#Dividing training set with 70% of data and Test with 30%.
smp_size <- floor(0.70 * nrow(data))
set.seed(123)
train_ind <- sample(seq_len(nrow(data)), size = smp_size)

train <- scale(as.matrix(data[train_ind, ]))
test <- scale(as.matrix(data[-train_ind, ]))

#Training data
trainlabel  <- as.matrix(train[,1])
traindata  <- train[,2:10]


#test data
testlabel  <- as.matrix(test[,1])
testdata  <- test[,2:10]
```

I will then create 10 K-Nearest Neighbor Classifier with k equals from 1 to 10.

```
library(class)

for(i in 1:10){
  #model
  assign(paste("knnmodel", i, sep = ""), knn(train=traindata, test = testdata, cl=trainlabel, k=i))
}
```

Plotting the apparent error rate results with this graph.



Clearly, k=3 or 5, 6,7 is the best one with the lowest error rate at 0.06

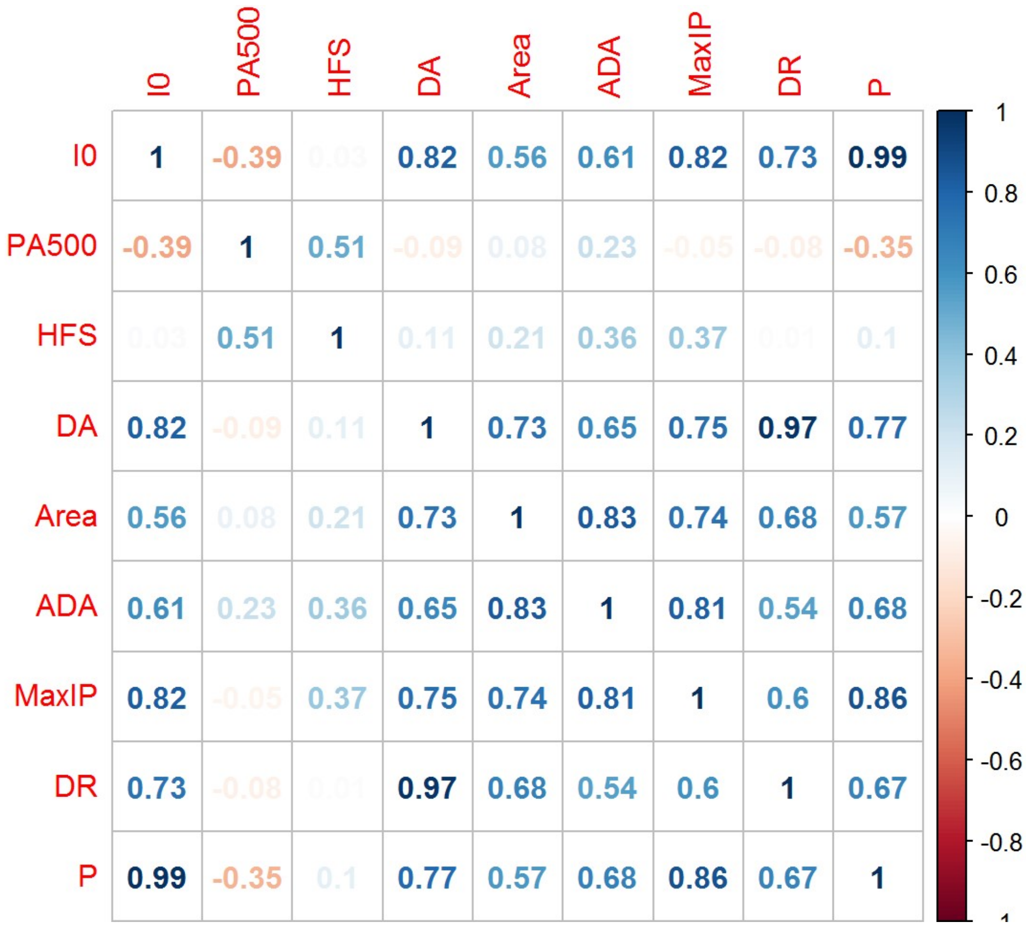
Second Step - Principal Component Analysis

I will perform PCA on my overall data and test four different method to decide how many of my 9 components should I retain to effectively summarize my data.For Each Method test, I will perform 2-Nearest Neighbor Classifier to see if my classification rate was improved.

```
# Preprocessing data by centering
y.bar <- apply(Y1, 2, mean)
Y <- t(t(Y1)-y.bar)



#overview after preprocessing
head(Y)
```

```
##              I0      PA500         HFS        DA       Area       ADA
## [1,] -259.4575 0.06731572 -0.08257673  38.23159  -491.5567  6.437019
## [2,] -454.2516 0.10676017  0.15059926 -69.41444 -4171.9157  2.635418
## [3,] -232.3723 0.11234522 -0.05116080  74.23629  4553.2367 21.421119
## [4,] -404.2516 0.12072280  0.17154321 -52.92853 -1932.9840 15.774740
## [5,] -421.4204 0.08058023  0.12965531 -65.65608 -4044.6927  2.868342
## [6,] -394.3786 0.02996568 -0.01695235 -71.94283 -4859.5981 -2.605164
##           MaxIP        DR         P
## [1,] -15.176379  54.02664 -253.8098
## [2,]  -5.663897 -67.62561 -410.4124
## [3,]   2.412038  87.07472 -153.8687
## [4,]  13.377187 -61.51201 -316.9363
## [5,]  -5.991869 -62.84402 -385.8416
## [6,] -25.624110 -59.02441 -381.2523
```

```
#checking how correlated is the data
library(corrplot)
corrplot(cor(Y), method="number")
```



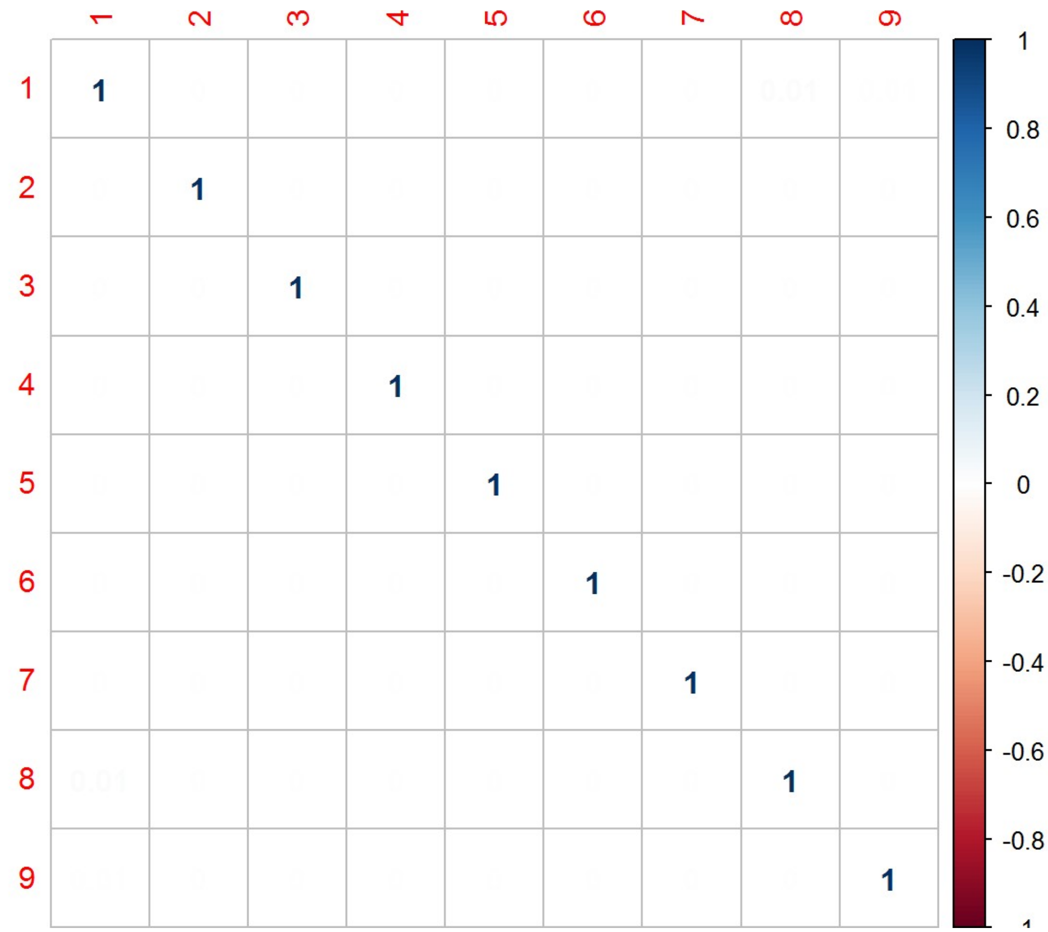

```
#getting all the principal components
S <- cov(Y)
C = matrix(round(eigen(S)$vectors, digits = 7),nrow=9,ncol=9)
A = t(C)

C
```

```
##            [,1]       [,2]       [,3]       [,4]       [,5]       [,6]
##  [1,] 0.0227685  0.7007091 -0.2193015 -0.5895295 -0.3075227  0.0003377
##  [2,] 0.0000003 -0.0000390 -0.0000217  0.0007528  0.0001727 -0.0013386
##  [3,] 0.0000011 -0.0000073  0.0001281  0.0011740 -0.0004305 -0.0008730
##  [4,] 0.0075117  0.1020608 -0.5611537  0.4437489 -0.2254843  0.6530514
##  [5,] 0.9994061 -0.0339408  0.0046135 -0.0037133  0.0002891  0.0003822
##  [6,] 0.0010433  0.0055121  0.0309353  0.1580544  0.1183112 -0.0107986
##  [7,] 0.0032208  0.0473363  0.0813509  0.4243511 -0.7574393 -0.4869511
##  [8,] 0.0065973  0.0810894 -0.6979450  0.1439556  0.3825269 -0.5798804
##  [9,] 0.0236150  0.6989933  0.3772132  0.4793202  0.3471997  0.0046603
##             [,7]       [,8]       [,9]
##  [1,] -0.1351980 -0.0011181 -0.0000585
##  [2,] -0.0021257 -0.2585212  0.9660020
##  [3,]  0.0037134 -0.9659991 -0.2585143
##  [4,]  0.0200221 -0.0001106  0.0006054
##  [5,]  0.0004506 -0.0000018  0.0000023
##  [6,] -0.9797443 -0.0027936 -0.0030619
##  [7,] -0.0148059  0.0013880 -0.0005274
##  [8,]  0.0542329  0.0008199 -0.0006578
##  [9,]  0.1350655  0.0009453  0.0001577
```

```
#eigen values
V <- matrix(round(eigen(S)$values, digits = 7),nrow=1,ncol=9)
V
```

```
##           [,1]     [,2]     [,3]     [,4]     [,5]    [,6]    [,7]
## [1,] 345637606 788042.2 25486.66 2939.197 950.0759 96.7578 35.5838
##           [,8]     [,9]
## [1,] 0.0048272 0.001191
```

```
# Rotated data
Z = Y%*%t(A)

# Getting uncorrelated data.
corrplot(cor(Z), method="number")
```



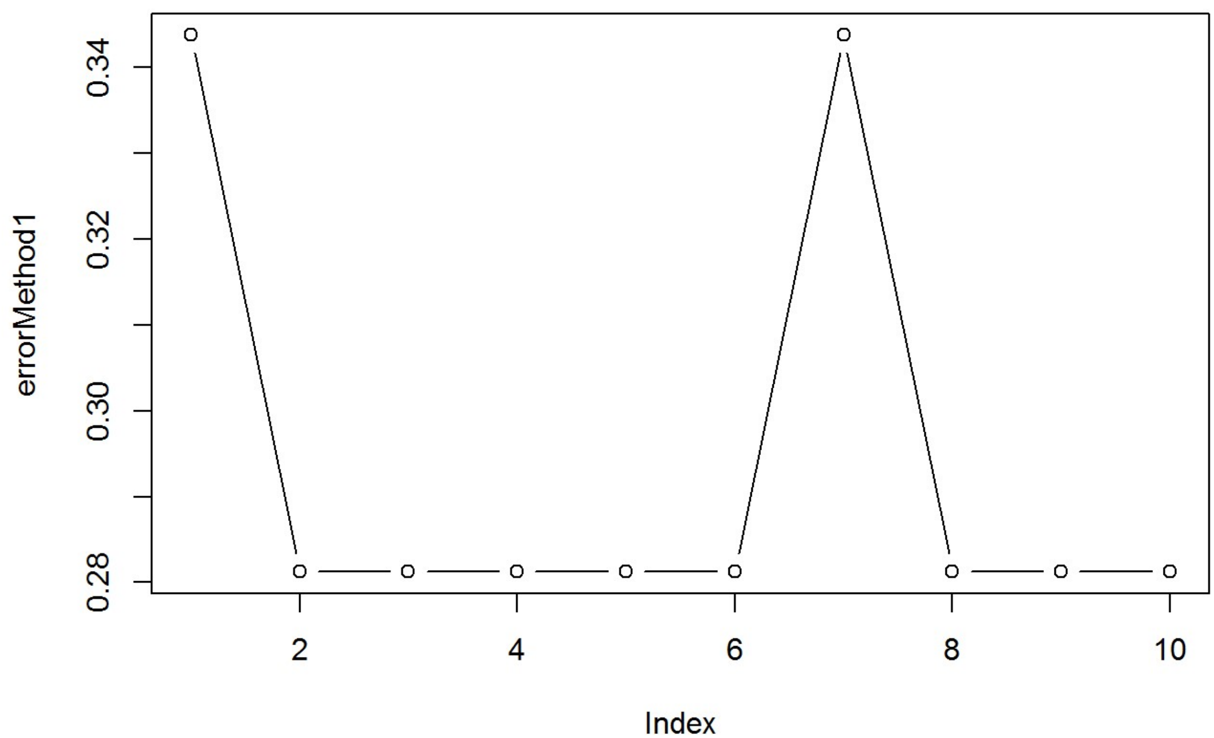
Method 1 - Getting the eigenvector with the highest eigenvalues

```
#getting the principal component witht the highest eigenvalue.
a1 <- C[,1]

#combining labels and the new data.
finaldataMethod1 <- cbind(Y2,Y%*%a1)
```

Perform another K-Nearest Neighbor Classifier with k equals from 1 to 10 on this new data given by this method. I will check to see if error was lower for k=2.

```
##train data
data <- finaldataMethod1

#Dividing training set with 70% of data and Test with 30%.
smp_size <- floor(0.70 * nrow(data))
set.seed(123)
train_ind <- sample(seq_len(nrow(data)), size = smp_size)

train <- scale(as.matrix(data[train_ind, ]))
test <- scale(as.matrix(data[-train_ind, ]))

#Train data
trainlabel  <- as.matrix(train[,1])
traindata  <- as.matrix(train[,2])



#test data
testlabel  <- as.matrix(test[,1])
testdata  <- as.matrix(test[,2])


library(class)

for(i in 1:10){

  #model
  assign(paste("knnmodel", i, sep = ""), knn(train=traindata, test = testdata,
cl=trainlabel, k=i))
}
```

Plotting the apparent error rate results with this graph.



Clearly with this method, my error rate for k= 2 did not improved. My error rate increased from .06 to .28. This method failed completly to improve my error rate with the lowest rate being at .28.

Method 2 - getting the eigenvectors with the eigenvalues greater than the average

```
S <- cov(Y)
lambda <-eigen(S)$value

average <- sum(lambda)/9


lambda[1]> average
```

```
## [1] TRUE
```

```
lambda[2]> average
```

```
## [1] FALSE
```

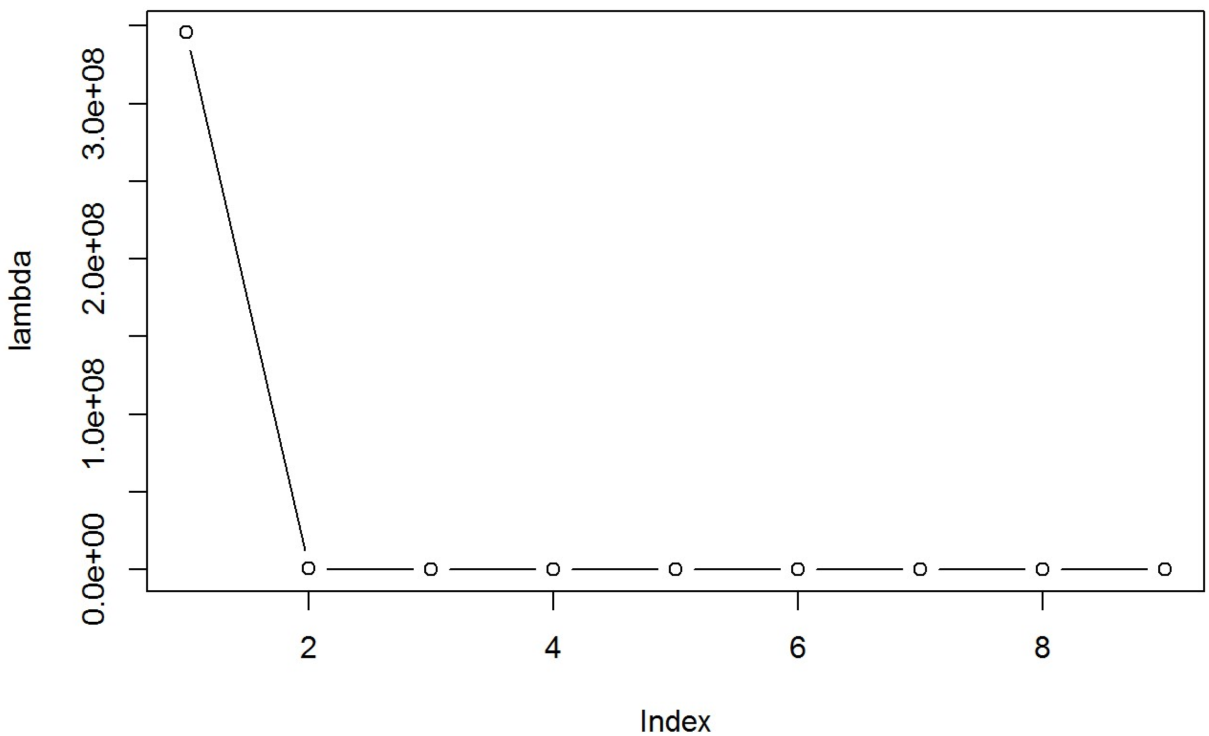
```
lambda[3]> average
```

```
## [1] FALSE
```

Since This method results with the same eigenvector as Method 1, We skip to metod #3.

Method 3 - use scree graph.

```
plot(lambda, type = "b")
```



Looking at the graph the first two eigenvectors should be retained.

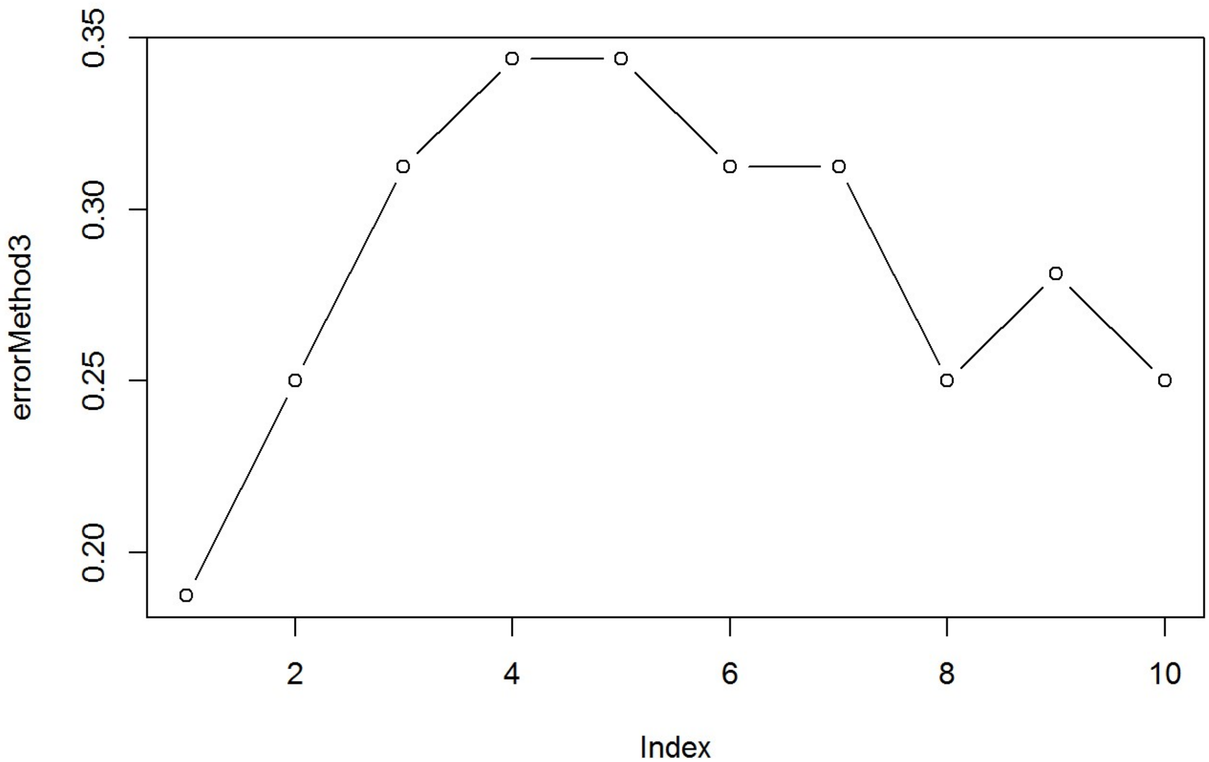
```
a3 <- C[,2]

finalMethod3<- cbind(Y2,Y%*%a3)
```

Perform another K-Nearest Neighbor Classifier with k equals from 1 to 10 on this new data given by this method. I will check to see if error was lower for k=2.

```
                          # Redoing first sep again

##train data
data <- finalMethod3

#Dividing training set with 70% of data and Test with 30%.
smp_size <- floor(0.70 * nrow(data))
set.seed(123)
train_ind <- sample(seq_len(nrow(data)), size = smp_size)


train <- scale(as.matrix(data[train_ind, ]))
test <- scale(as.matrix(data[-train_ind, ]))

#Train data
trainlabel  <- as.matrix(train[,1])
traindata  <- as.matrix(train[,2])


#test data
testlabel  <- as.matrix(test[,1])
testdata  <- as.matrix(test[,2])
```

```
library(class)

for(i in 1:10){

  #model
  assign(paste("knnmodel", i, sep = ""), knn(train=traindata, test = testdata, 
cl=trainlabel, k=i))
}
```

Plotting the apparent error rate results with this graph.



With Method 3, my error rate for k= 2 did not improved. The error rate also increased from .06 to .25. This method failed completly to improve my error rate with the lowest rate being again at .19.