Assignment 11.2.1

Chitramoy Mukherjee

02/24/2023

Install and Load required packages :

```
knitr::opts_chunk$set(echo = TRUE)
knitr::opts_chunk$set(warning = FALSE)
knitr::opts_chunk$set(fig.width = 12, fig.height = 10)
knitr::opts_chunk$set(tidy.opts = list(width.cutoff = 70), tidy = TRUE)

# Package names
# packages <- c("ggplot2","dplyr","tidyr","magrittr","tidyverse","purrr")
packages <- c("e1071","caTools","class","ggplot2","plotly")

# Install packages not yet installed
installed_packages <- packages %in% rownames(installed.packages())
if (any(installed_packages == FALSE)) {
  install.packages(packages[!installed_packages])
}

# Packages loading
invisible(lapply(packages, library, character.only = TRUE))
```

```
##
## Attaching package: 'plotly'

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

## The following object is masked from 'package:stats':
##
##     filter

## The following object is masked from 'package:graphics':
##
##     layout
```

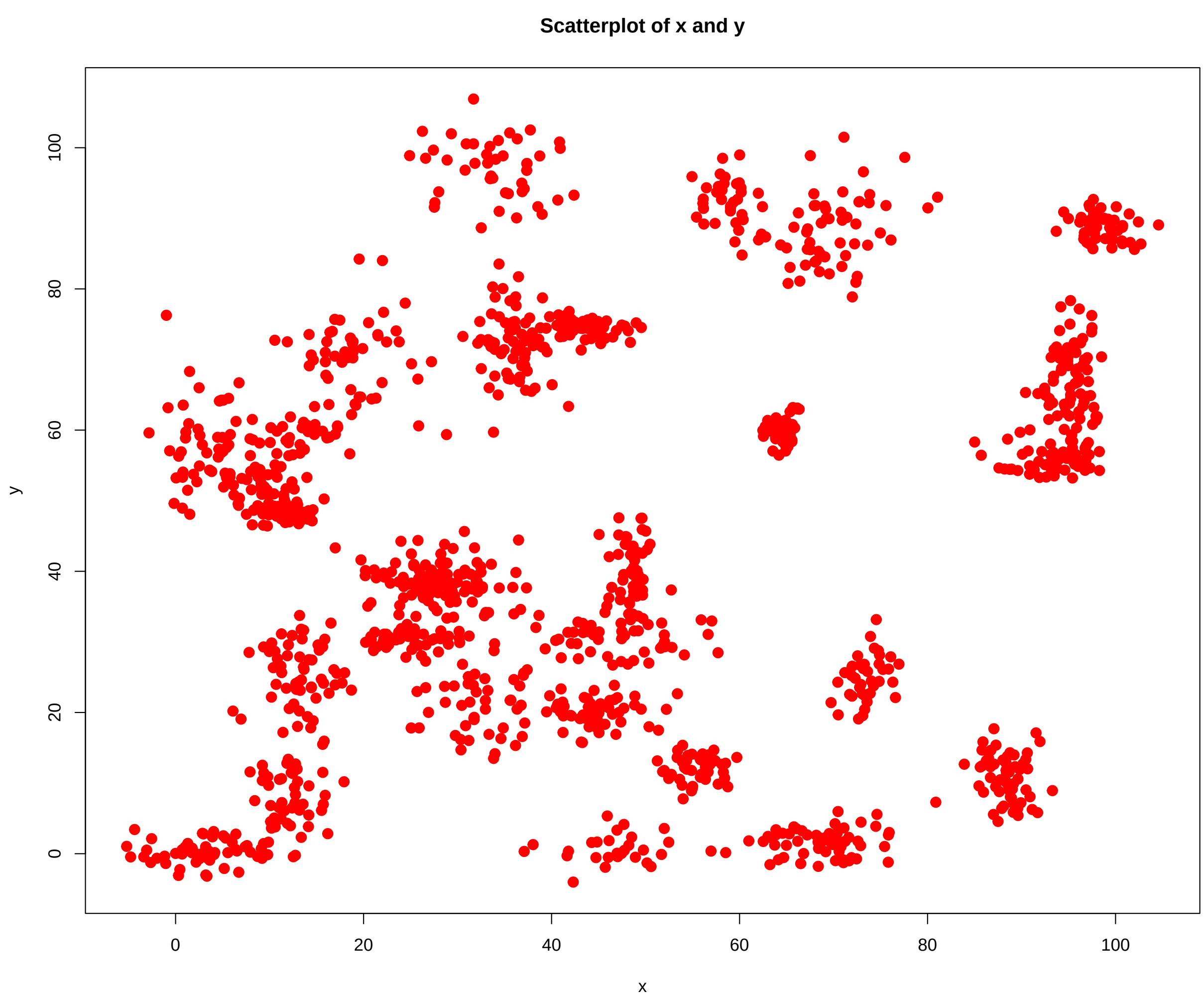
Set the working directory to the root of your DSC 520 directory

setwd("C:/Users/14024/Desktop/dsc520-fork-chitro")

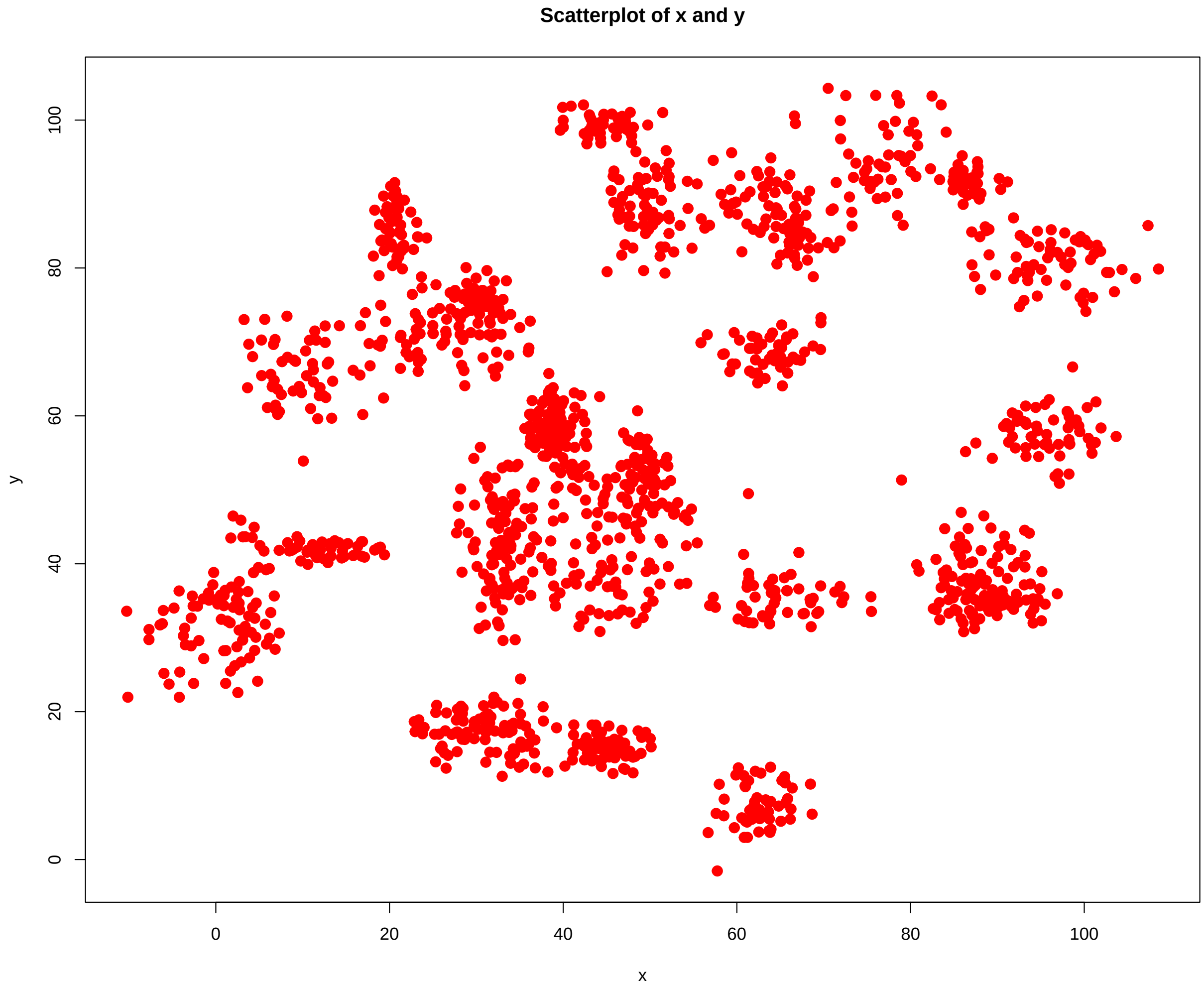
```r
## Set the working directory to the root of your DSC 520 directory
setwd("C:/Users/14024/Desktop/dsc520-fork-chitro")

## Load data from data/binary-classifier-data.csv
bc_data <- read.csv("data/binary-classifier-data.csv")
tc_data <- read.csv("data/trinary-classifier-data.csv")

#create scatterplot of x vs. y from binary-classifier-data
plot(x = bc_data$x , y = bc_data$y, col='red', pch=19, cex=1.3,
     xlab='x', ylab='y', main='Scatterplot of x and y')
```




```r
#create scatterplot of x vs. y from trinary-classifier-data
plot(x = tc_data$x , y = tc_data$y, col='red', pch=19, cex=1.3,
     xlab='x', ylab='y', main='Scatterplot of x and y')
```




```
# Applying KNN on binary-classifier-data

bc_data_sub <- as.data.frame(bc_data[,2:3])

set.seed(123)
dat.d <- sample(1:nrow(bc_data_sub),size=nrow(bc_data_sub)*0.7,replace = FALSE)
#random selection of 70% data.

# Data Splicing
train.bc_data <- bc_data[dat.d,] # 70% training data
test.bc_data <- bc_data[-dat.d,] # remaining 30% test data

train.bc_data_labels <- bc_data[dat.d,1]
test.bc_data_labels <-bc_data[-dat.d,1]

## Building a Machine Learning model
#Find the number of observation
NROW(train.bc_data)
```

```
## [1] 1048
```

```
NROW(test.bc_data)
```

```
## [1] 450
```

```
knn.32 <- knn(train=train.bc_data, test=test.bc_data, cl=train.bc_data_labels, k=32)
knn.33 <- knn(train=train.bc_data, test=test.bc_data, cl=train.bc_data_labels, k=33)

#Calculate the proportion of correct classification for k = 32, 33
ACC.32 <- 100 * sum(test.bc_data_labels == knn.32)/NROW(test.bc_data_labels)
ACC.33 <- 100 * sum(test.bc_data_labels == knn.33)/NROW(test.bc_data_labels)

ACC.32
```

```
## [1] 97.55556
```

```
ACC.33
```

```
## [1] 97.55556
```

```
# Fit a k nearest neighbors' model for each dataset for k=3, k=5, k=10, k=15, k=20, and k=25
knn.3 <- knn(train=train.bc_data, test=test.bc_data, cl=train.bc_data_labels, k=3)
knn.5 <- knn(train=train.bc_data, test=test.bc_data, cl=train.bc_data_labels, k=5)
knn.10 <- knn(train=train.bc_data, test=test.bc_data, cl=train.bc_data_labels, k=10)
knn.15 <- knn(train=train.bc_data, test=test.bc_data, cl=train.bc_data_labels, k=15)
knn.20 <- knn(train=train.bc_data, test=test.bc_data, cl=train.bc_data_labels, k=20)
knn.25 <- knn(train=train.bc_data, test=test.bc_data, cl=train.bc_data_labels, k=25)

#Compute the accuracy of the resulting models for each value of k.
ACC.3 <- 100 * sum(test.bc_data_labels == knn.3)/NROW(test.bc_data_labels)
ACC.5 <- 100 * sum(test.bc_data_labels == knn.5)/NROW(test.bc_data_labels)
ACC.10 <- 100 * sum(test.bc_data_labels == knn.10)/NROW(test.bc_data_labels)
ACC.15 <- 100 * sum(test.bc_data_labels == knn.15)/NROW(test.bc_data_labels)
ACC.20 <- 100 * sum(test.bc_data_labels == knn.20)/NROW(test.bc_data_labels)
ACC.25 <- 100 * sum(test.bc_data_labels == knn.25)/NROW(test.bc_data_labels)

ACC.3
```

```
## [1] 98
```

```
ACC.5
```

```
## [1] 97.55556
```

```
ACC.10
```

```
## [1] 98.44444
```

```
ACC.15
```

```
## [1] 97.55556
```

```r
ACC.20
```

```
## [1] 97.33333
```

```r
ACC.25
```

```
## [1] 98.44444
```

```r
# Applying KNN on trinary-classifier-data.csv

tc_data_sub <- as.data.frame(tc_data[,2:3])

set.seed(123)
dat.d <- sample(1:nrow(tc_data_sub),size=nrow(tc_data_sub)*0.7,replace = FALSE)
#random selection of 70% data.

# Data Splicing
train.tc_data <- tc_data[dat.d,] # 70% training data
test.tc_data <- tc_data[-dat.d,] # remaining 30% test data

train.tc_data_labels <- tc_data[dat.d,1]
test.tc_data_labels <-tc_data[-dat.d,1]

## Building a Machine Learning model
#Find the number of observation
NROW(train.tc_data)
```

```
## [1] 1097
```

```r
NROW(test.tc_data)
```

```
## [1] 471
```

```r
knn.33 <- knn(train=train.tc_data, test=test.tc_data, cl=train.tc_data_labels, k=33)
knn.34 <- knn(train=train.tc_data, test=test.tc_data, cl=train.tc_data_labels, k=34)

#Calculate the proportion of correct classification for k = 33, 34
ACC.33 <- 100 * sum(test.tc_data_labels == knn.33)/NROW(test.tc_data_labels)
ACC.34 <- 100 * sum(test.tc_data_labels == knn.34)/NROW(test.tc_data_labels)

ACC.33
```

```
## [1] 86.41189
```

```r
ACC.34
```

```
## [1] 85.98726
```

```r
# Fit a k nearest neighbors' model for each dataset for k=3, k=5, k=10, k=15, k=20, and k=25
knn.3 <- knn(train=train.tc_data, test=test.tc_data, cl=train.tc_data_labels, k=3)
knn.5 <- knn(train=train.tc_data, test=test.tc_data, cl=train.tc_data_labels, k=5)
knn.10 <- knn(train=train.tc_data, test=test.tc_data, cl=train.tc_data_labels, k=10)
knn.15 <- knn(train=train.tc_data, test=test.tc_data, cl=train.tc_data_labels, k=15)
knn.20 <- knn(train=train.tc_data, test=test.tc_data, cl=train.tc_data_labels, k=20)
knn.25 <- knn(train=train.tc_data, test=test.tc_data, cl=train.tc_data_labels, k=25)

#Compute the accuracy of the resulting models for each value of k.
ACCR.3 <- 100 * sum(test.tc_data_labels == knn.3)/NROW(test.tc_data_labels)
ACCR.5 <- 100 * sum(test.tc_data_labels == knn.5)/NROW(test.tc_data_labels)
ACCR.10 <- 100 * sum(test.tc_data_labels == knn.10)/NROW(test.tc_data_labels)
ACCR.15 <- 100 * sum(test.tc_data_labels == knn.15)/NROW(test.tc_data_labels)
ACCR.20 <- 100 * sum(test.tc_data_labels == knn.20)/NROW(test.tc_data_labels)
ACCR.25 <- 100 * sum(test.tc_data_labels == knn.25)/NROW(test.tc_data_labels)

ACCR.3
```

```
## [1] 93.20594
```

```r
ACCR.5
```

```
## [1] 92.14437
```

```r
ACCR.10
```

```
## [1] 89.38429
```

```r
ACCR.15
```

```
## [1] 89.17197
```

```r
ACCR.20
```

```
## [1] 86.41189
```

```r
ACCR.25
```

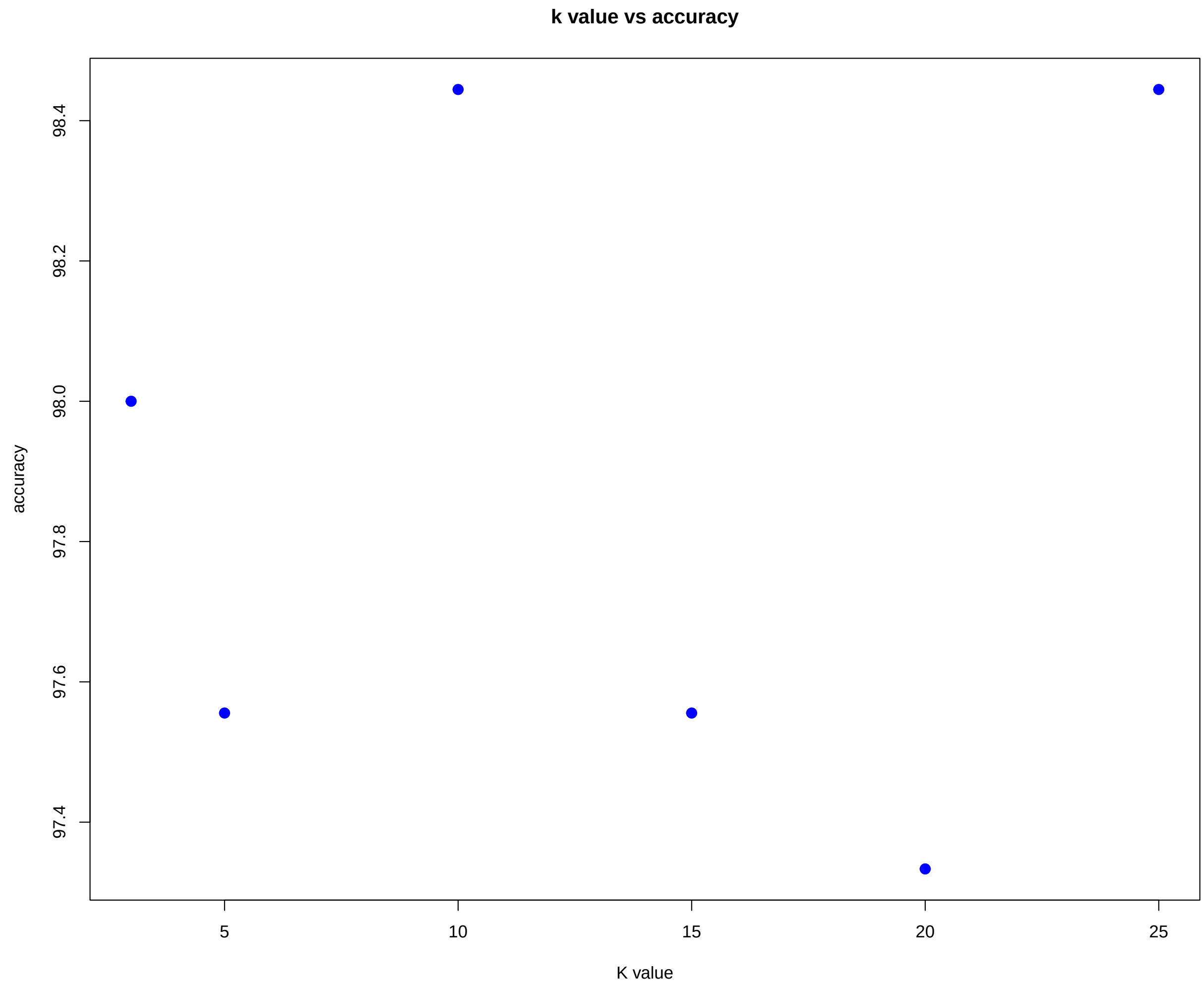
```
## [1] 86.83652
```

```r
# Plot the results in a graph where the x-axis is the different values of k and the
# y-axis is the accuracy of the model based on binary-classifier-data.

a <- c(3,5,10,15,20,25)
b <- c(ACC.3,ACC.5,ACC.10,ACC.15,ACC.20,ACC.25)

bc_df <- data.frame(a,b)

plot(x = bc_df$a , y = bc_df$b, col='blue', pch=19, cex=1.3,
     xlab='K value', ylab='accuracy', main='k value vs accuracy')
```

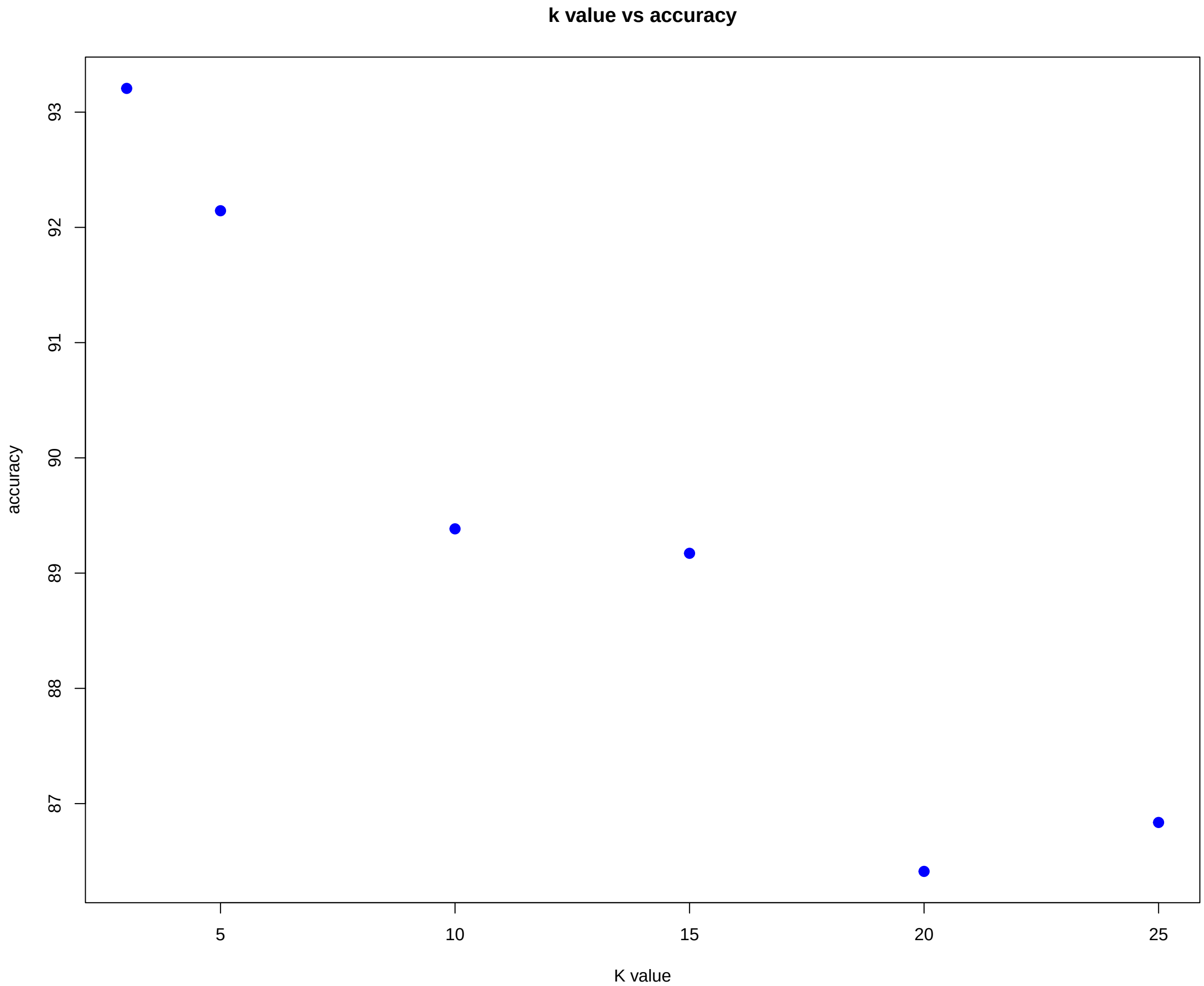



```
# Plot the results in a graph where the x-axis is the different values of k and the
# y-axis is the accuracy of the model based on trinary-classifier-data.

a <- c(3,5,10,15,20,25)
b <- c(ACCR.3,ACCR.5,ACCR.10,ACCR.15,ACCR.20,ACCR.25)

tc_df <- data.frame(a,b)

plot(x = tc_df$a , y = tc_df$b, col='blue', pch=19, cex=1.3,
     xlab='K value', ylab='accuracy', main='k value vs accuracy')
```



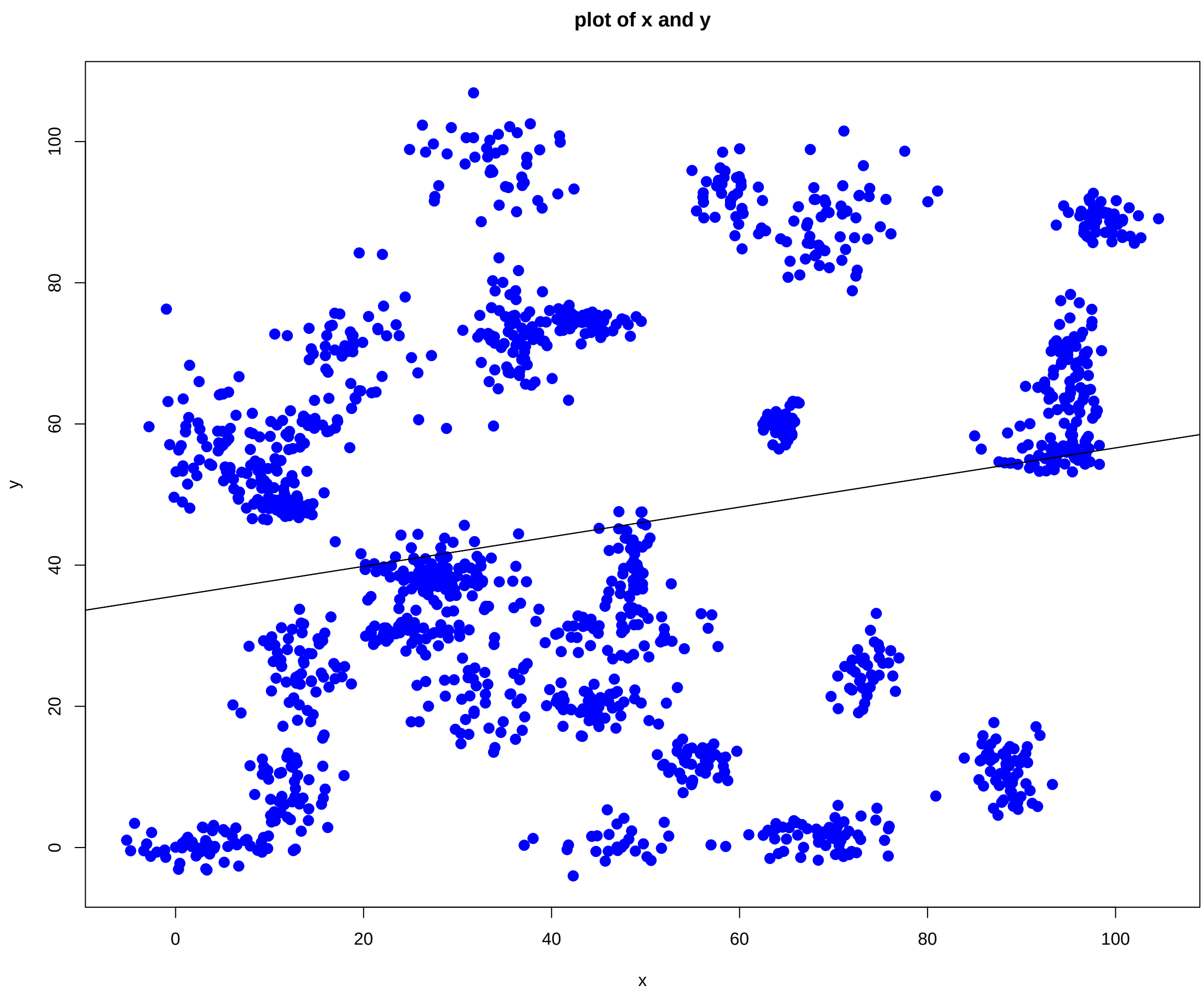

```r
# Looking back at the plots of the data, do you think a
# linear classifier would work well on binary-classifier-data  dataset?

plot(x = bc_data$x , y = bc_data$y, col='blue', pch=19, cex=1.3,
     xlab='x', ylab='y', main='plot of x and y')
lm(bc_data$x ~ bc_data$y)
```

```
##
## Call:
## lm(formula = bc_data$x ~ bc_data$y)
##
## Coefficients:
## (Intercept)    bc_data$y
##     35.6321       0.2098
```

```r
abline(35.6321,  0.2098)

abline(lm(bc_data$x ~ bc_data$y))
```



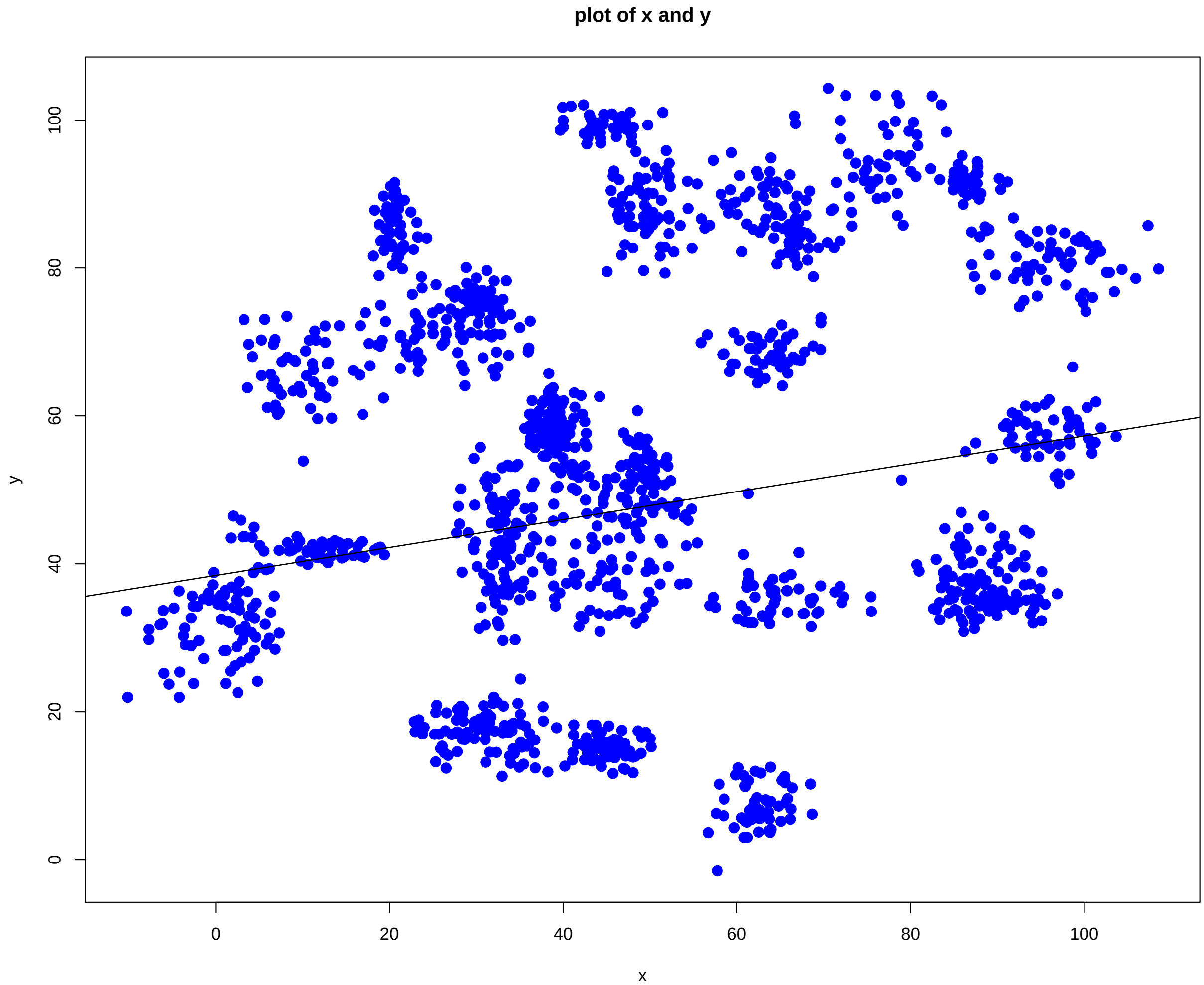

```
# Looking back at the plots of the data, do you think a
# linear classifier would work well on trinary-classifier-data dataset?

plot(x = tc_data$x , y = tc_data$y, col='blue', pch=19, cex=1.3,
     xlab='x', ylab='y', main='plot of x and y')
lm(tc_data$x ~ tc_data$y)
```

```
##
## Call:
## lm(formula = tc_data$x ~ tc_data$y)
##
## Coefficients:
## (Intercept)    tc_data$y
##     38.4345       0.1886
```

```
abline(38.4345,   0.1886)

abline(lm(tc_data$x ~ tc_data$y))
```



How does the accuracy of your logistic regression classifier from last week compare? Why is the accuracy different between these two methods?

accuracy of logistic regression classifier has been increased compare to last week. First of all, the KNN is a deterministic algorithm, it means if you keep the value of K and run the algorithm n times, the results will be the same.

On the other hand, the logistic regression is a stochastic algorithm. It means t he algorithm use some random values to achieve it's goal. If we run the algorithm many times will see the results varying. It's normal, although you wanna reduce this variation.