assignment_11.2.1_ChattapadhyayKausik

Kausik Chattapadhyay

2022-11-10

Assignment

In this problem, you will use the nearest neighbors algorithm to fit a model on two simplified datasets. The first dataset (found in binary-classifier-data.csv) contains three variables; label, x, and y. The label variable is either 0 or 1 and is the output we want to predict using the x and y variables. The second dataset (found in trinary-classifier-data.csv) is similar to the first dataset except that the label variable can be 0, 1, or 2. Note that in real-world datasets, your labels are usually not numbers, but text-based descriptions of the categories (e.g. spam or ham). In practice, you will encode categorical variables into numeric values.

Question A:

Plot the data from each dataset using a scatter plot.

Answer for A

Code

```
## Set the working directory to the root of your DSC 520 directory
setwd("/Users/kausik/desktop/MS Data Science/DSC 520/dsc520-stats-r-assignments")
classifierdata.binary <- read.csv("data/binary-classifier-data.csv")
head(classifierdata.binary)
```

```
##   label        x        y
## 1     0 70.88469 83.17702
## 2     0 74.97176 87.92922
## 3     0 73.78333 92.20325
## 4     0 66.40747 81.10617
## 5     0 69.07399 84.53739
## 6     0 72.23616 86.38403
```

```
summary(classifierdata.binary)
```

```
##      label             x               y
##  Min.   :0.000   Min.   : -5.20   Min.   : -4.019
##  1st Qu.:0.000   1st Qu.: 19.77   1st Qu.: 21.207
##  Median :0.000   Median : 41.76   Median : 44.632
```

```
##  Mean   :0.488   Mean   : 45.07   Mean   : 45.011
##  3rd Qu.:1.000   3rd Qu.: 66.39   3rd Qu.: 68.698
##  Max.   :1.000   Max.   :104.58   Max.   :106.896
```

```r
## Load the `trinary-classifier-data.csv`
classifierdata.trinary <- read.csv("data/trinary-classifier-data.csv")
head(classifierdata.trinary)
```

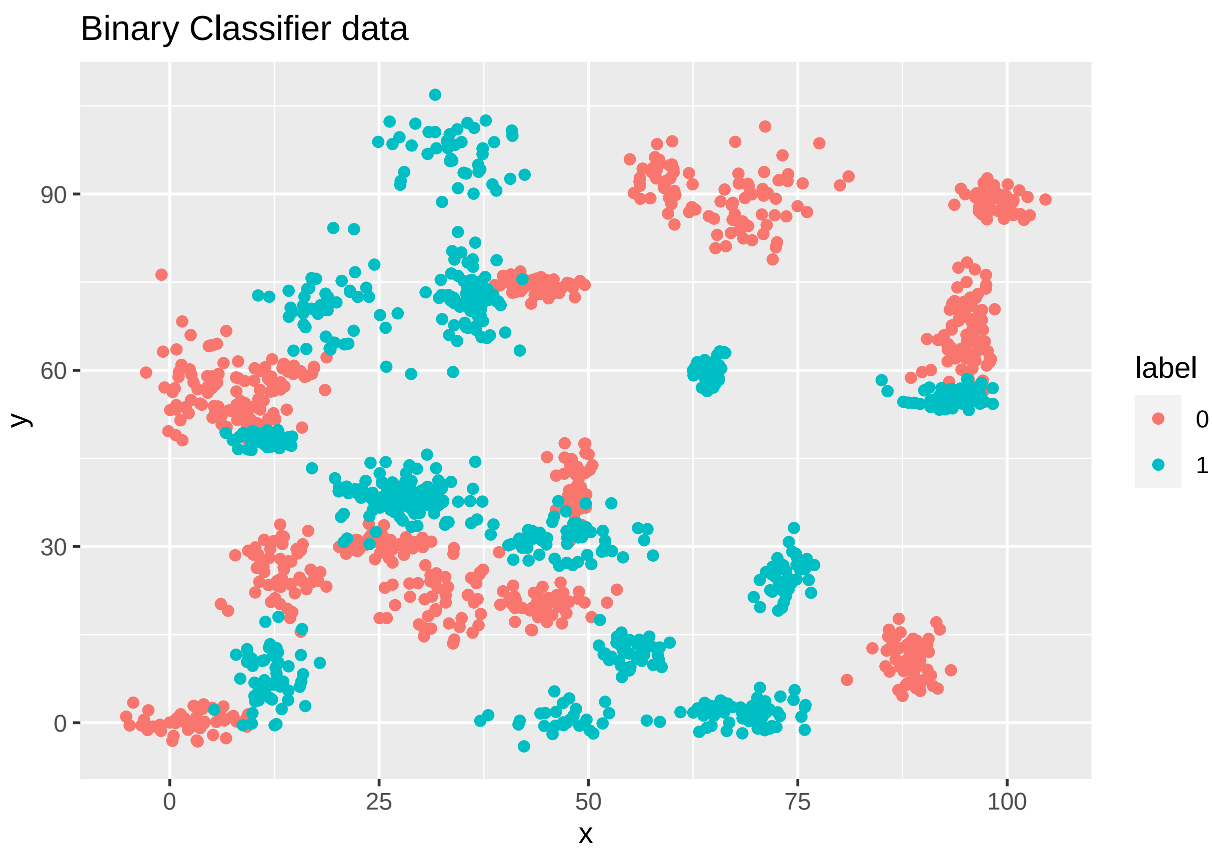
```
##   label        x        y
## 1     0 30.08387 39.63094
## 2     0 31.27613 51.77511
## 3     0 34.12138 49.27575
## 4     0 32.58222 41.23300
## 5     0 34.65069 45.47956
## 6     0 33.80513 44.24656
```

```r
summary(classifierdata.trinary)
```

```
##      label             x                y
##  Min.   :0.000   Min.   :-10.26   Min.   : -1.541
##  1st Qu.:0.000   1st Qu.: 31.15   1st Qu.: 35.906
##  Median :1.000   Median : 45.59   Median : 55.073
##  Mean   :1.037   Mean   : 48.86   Mean   : 55.282
##  3rd Qu.:2.000   3rd Qu.: 66.27   3rd Qu.: 77.403
##  Max.   :2.000   Max.   :108.56   Max.   :104.293
```

```r
classifierdata.binary$label <- as.factor(classifierdata.binary$label)
classifierdata.trinary$label <- as.factor(classifierdata.trinary$label)

library(ggplot2)
ggplot(data = classifierdata.binary, aes(y = y, x = x, color = label)) +
  geom_point() + ggtitle("Binary Classifier data")
```




```r
ggplot(data = classifierdata.trinary, aes(y = y, x = x, color = label)) +
  geom_point() + ggtitle("Trinary Classifier data")
```

Question B.

Fit a k nearest neighbors model for each dataset for k=3, k=5, k=10, k=15, k=20, and k=25. Compute the accuracy of the resulting models for each value of k. 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.

Answer for B:

Binary Classifier - Select data points, normalize and Split 80-20 into train and test datasets.

```
# normalize function
normalize<-function(x){
  return (
    (x - min(x))/max(x)-min(x)
   )
 }
classifierdata.normalizedbinary<-as.data.frame(lapply(classifierdata.binary[,c(2:3)], normalize))
head(classifierdata.normalizedbinary)
```

```
##          x        y
## 1 5.928057 4.835151
```

```
## 2 5.967139 4.879608
## 3 5.955775 4.919591
## 4 5.885243 4.815779
## 5 5.910742 4.847877
## 6 5.940980 4.865152
```

```
str(classifierdata.normalizedbinary)
```

```
## 'data.frame':    1498 obs. of  2 variables:
##  $ x: num  5.93 5.97 5.96 5.89 5.91 ...
##  $ y: num  4.84 4.88 4.92 4.82 4.85 ...
```

```
summary(classifierdata.normalizedbinary)
```

```
##        x               y
##  Min.   :5.200   Min.   :4.019
##  1st Qu.:5.439   1st Qu.:4.255
##  Median :5.650   Median :4.475
##  Mean   :5.681   Mean   :4.478
##  3rd Qu.:5.885   3rd Qu.:4.700
##  Max.   :6.250   Max.   :5.057
```

```
#Select train and test data
data.d = sample(1:nrow(classifierdata.normalizedbinary), size= nrow(classifierdata.normalizedbinary) *0
traindata.binary <- classifierdata.binary[data.d,] #80% training data
testdata_binary <- classifierdata.binary[-data.d,] #20% training data

#Create separate dataframes for train and test data for label
traindata_binary_df<-classifierdata.binary[data.d,1]
testdata_binary_df<-classifierdata.binary[-data.d,1]
k_valuebinary<-round(sqrt(nrow(classifierdata.binary)))
library(class)
```

```
## Warning: package 'class' was built under R version 4.0.5
```

```
# find no of observations
NROW(traindata_binary_df)
```

```
## [1] 1198
```

```
#define result data frames
k_binarydata <- c()
accurary_binarydata <- c()
# Calculate KNN for K=3
knn.3 <- knn(train = traindata.binary , test = testdata_binary, cl = traindata_binary_df, k = 3)
# Calculate Accuracy
acc.3 <- 100* sum(testdata_binary_df == knn.3)/NROW(testdata_binary_df)
acc.3
```

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

```r
k_binarydata <- c(k_binarydata, 3)
accurary_binarydata<- c(accurary_binarydata, acc.3)
# Calculate KNN for K=5
knn.5 <- knn(train = traindata.binary , test = testdata_binary, cl = traindata_binary_df, k = 5)
# Calculate Accuracy
acc.5 <- 100* sum(testdata_binary_df == knn.5)/NROW(testdata_binary_df)
acc.5
```

```
## [1] 98.33333
```

```r
k_binarydata <- c(k_binarydata, 5)
accurary_binarydata<- c(accurary_binarydata, acc.5)
# Calculate KNN for K=10
knn.10 <- knn(train = traindata.binary , test = testdata_binary, cl = traindata_binary_df, k = 10)
# Calculate Accuracy
acc.10 <- 100* sum(testdata_binary_df == knn.10)/NROW(testdata_binary_df)
acc.10
```

```
## [1] 98.33333
```

```r
k_binarydata <- c(k_binarydata, 10)
accurary_binarydata<- c(accurary_binarydata, acc.10)
# Calculate KNN for K=15
knn.15 <- knn(train = traindata.binary , test = testdata_binary, cl = traindata_binary_df, k = 15)
# Calculate Accuracy
acc.15 <- 100* sum(testdata_binary_df == knn.15)/NROW(testdata_binary_df)
acc.15
```

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

```r
k_binarydata <- c(k_binarydata, 15)
accurary_binarydata<- c(accurary_binarydata, acc.15)
# Calculate KNN for K=20
knn.20 <- knn(train = traindata.binary , test = testdata_binary, cl = traindata_binary_df, k = 20)
# Calculate Accuracy
acc.20 <- 100* sum(testdata_binary_df == knn.20)/NROW(testdata_binary_df)
acc.20
```

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

```r
k_binarydata <- c(k_binarydata, 20)
accurary_binarydata<- c(accurary_binarydata, acc.20)
# Calculate KNN for K=25
knn.25 <- knn(train = traindata.binary , test = testdata_binary, cl = traindata_binary_df, k = 25)
# Calculate Accuracy
acc.25 <- 100* sum(testdata_binary_df == knn.25)/NROW(testdata_binary_df)
acc.25
```

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

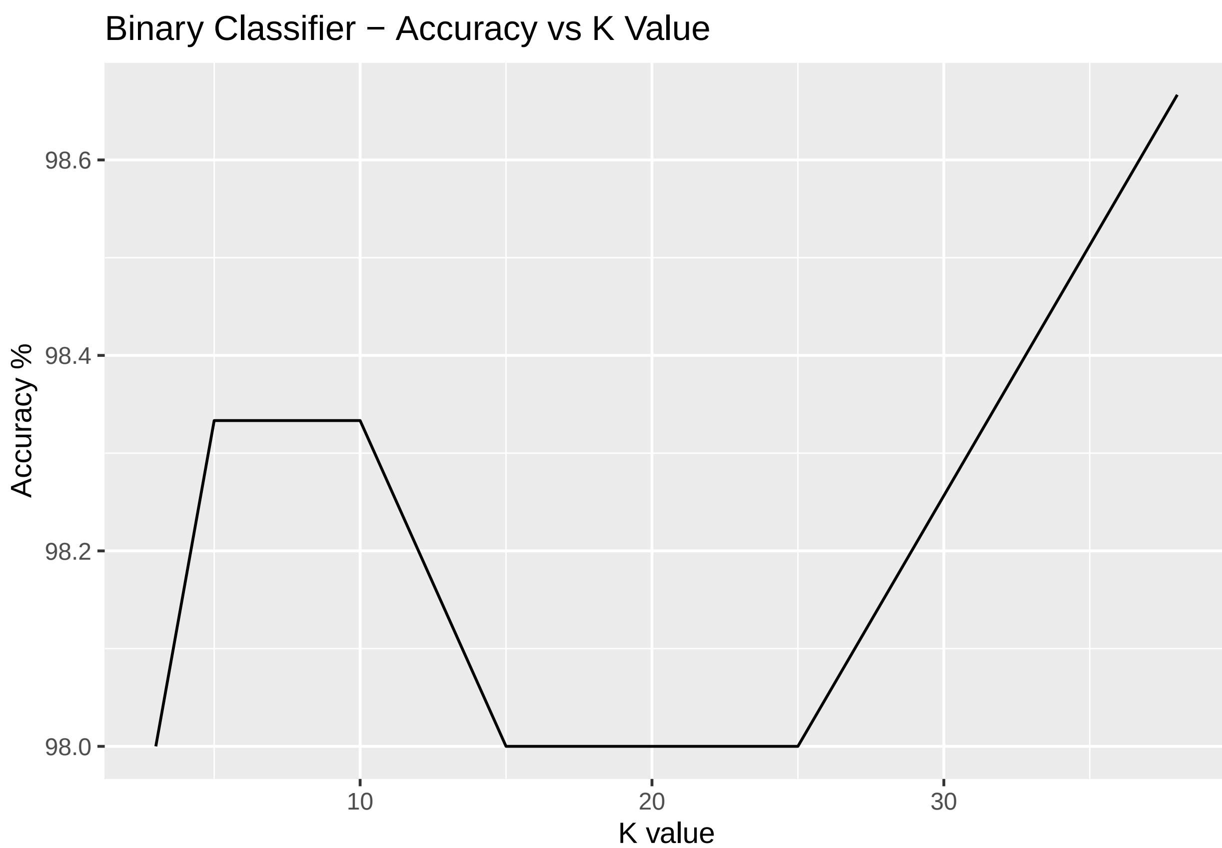
```r
k_binarydata <- c(k_binarydata, 25)
accurary_binarydata<- c(accurary_binarydata, acc.25)
# Calculate KNN for K=38 (square root of observations)
knn.38 <- knn(train = traindata.binary , test = testdata_binary, cl = traindata_binary_df, k = 38)
# Calculate Accuracy
acc.38 <- 100* sum(testdata_binary_df == knn.38)/NROW(testdata_binary_df)
acc.38
```

```
## [1] 98.66667
```

```r
k_binarydata <- c(k_binarydata, 38)
accurary_binarydata<- c(accurary_binarydata, acc.38)
accuracy_binarydata_df <- data.frame(k_binarydata, accurary_binarydata)
accuracy_binarydata_df
```

```
##   k_binarydata accurary_binarydata
## 1            3            98.00000
## 2            5            98.33333
## 3           10            98.33333
## 4           15            98.00000
## 5           20            98.00000
## 6           25            98.00000
## 7           38            98.66667
```

```r
## K Values vs Accuracy Plotting
library(ggplot2)
ggplot(data = accuracy_binarydata_df, aes(y = accurary_binarydata, x = k_binarydata)) +
    geom_line() + ggtitle("Binary Classifier - Accuracy vs K Value") +
    ylab("Accuracy %")  + xlab("K value")
```



Trinary Classifier - Select data points, normalize and Split 80-20 into train

and test datasets.

```
#normalize function
normalize<-function(x){
   return (
     (x - min(x))/max(x)-min(x)
   )
 }
classifierdata.normalizedtrinary<-as.data.frame(lapply(classifierdata.trinary[,c(2:3)], normalize))

head(classifierdata.normalizedtrinary)
```

```
##          x        y
## 1 10.63355 1.935708
## 2 10.64453 2.052150
## 3 10.67074 2.028185
## 4 10.65656 1.951069
## 5 10.67562 1.991786
## 6 10.66783 1.979964
```

```
str(classifierdata.normalizedtrinary)
```

```
## 'data.frame':    1568 obs. of  2 variables:
##  $ x: num  10.6 10.6 10.7 10.7 10.7 ...
##  $ y: num  1.94 2.05 2.03 1.95 1.99 ...
```

```
summary(classifierdata.normalizedtrinary)
```

```
##        x               y
##  Min.   :10.26   Min.   :1.541
##  1st Qu.:10.64   1st Qu.:1.900
##  Median :10.78   Median :2.084
##  Mean   :10.81   Mean   :2.086
##  3rd Qu.:10.97   3rd Qu.:2.298
##  Max.   :11.36   Max.   :2.556
```

```
#Select train and test data
data.d = sample(1:nrow(classifierdata.normalizedtrinary), size= nrow(classifierdata.normalizedtrinary)
traindata.trinary <- classifierdata.trinary[data.d,] #80% training data
testdata_trinary <- classifierdata.trinary[-data.d,] #20% training data
#Create separate dataframes for train and test data for label
traindata_trinary_df<-classifierdata.trinary[data.d,1]
testdata_trinary_df<-classifierdata.trinary[-data.d,1]
k_valuetrinary<-round(sqrt(nrow(classifierdata.trinary)))
library(class)
# find no of observations
NROW(traindata_trinary_df)
```

```
## [1] 1254
```

```
#define result data frames
k_trinarydata <- c()
accurary_trinarydata <- c()
# Calculate KNN for K=3
knn.3 <- knn(train = traindata.trinary , test = testdata_trinary, cl = traindata_trinary_df, k = 3)
# Calculate Accuracy
acc.3 <- 100* sum(testdata_trinary_df == knn.3)/NROW(testdata_trinary_df)
acc.3
```

```
## [1] 93.63057
```

```
k_trinarydata <- c(k_trinarydata, 3)
accurary_trinarydata<- c(accurary_trinarydata, acc.3)
# Calculate KNN for K=5
knn.5 <- knn(train = traindata.trinary , test = testdata_trinary,
             cl = traindata_trinary_df, k = 5)
# Calculate Accuracy
acc.5 <- 100* sum(testdata_trinary_df == knn.5)/NROW(testdata_trinary_df)
acc.5
```

```
## [1] 93.94904
```

```r
k_trinarydata <- c(k_trinarydata, 5)
accurary_trinarydata<- c(accurary_trinarydata, acc.5)
# Calculate KNN for K=10
knn.10 <- knn(train = traindata.trinary , test = testdata_trinary,
              cl = traindata_trinary_df, k = 10)
# Calculate Accuracy
acc.10 <- 100* sum(testdata_trinary_df == knn.10)/NROW(testdata_trinary_df)
acc.10
```

```
## [1] 91.71975
```

```r
k_trinarydata <- c(k_trinarydata, 10)
accurary_trinarydata<- c(accurary_trinarydata, acc.10)
# Calculate KNN for K=15
knn.15 <- knn(train = traindata.trinary , test = testdata_trinary,
              cl = traindata_trinary_df, k = 15)
# Calculate Accuracy
acc.15 <- 100* sum(testdata_trinary_df == knn.15)/NROW(testdata_trinary_df)
acc.15
```

```
## [1] 89.80892
```

```r
k_trinarydata <- c(k_trinarydata, 15)
accurary_trinarydata<- c(accurary_trinarydata, acc.15)
# Calculate KNN for K=20
knn.20 <- knn(train = traindata.trinary , test = testdata_trinary,
              cl = traindata_trinary_df, k = 20)
# Calculate Accuracy
acc.20 <- 100* sum(testdata_trinary_df == knn.20)/NROW(testdata_trinary_df)
acc.20
```

```
## [1] 89.80892
```

```r
k_trinarydata <- c(k_trinarydata, 20)
accurary_trinarydata<- c(accurary_trinarydata, acc.20)
# Calculate KNN for K=25
knn.25 <- knn(train = traindata.trinary , test = testdata_trinary, cl = traindata_trinary_df, k = 25)
# Calculate Accuracy
acc.25 <- 100* sum(testdata_trinary_df == knn.25)/NROW(testdata_trinary_df)
acc.25
```

```
## [1] 89.80892
```

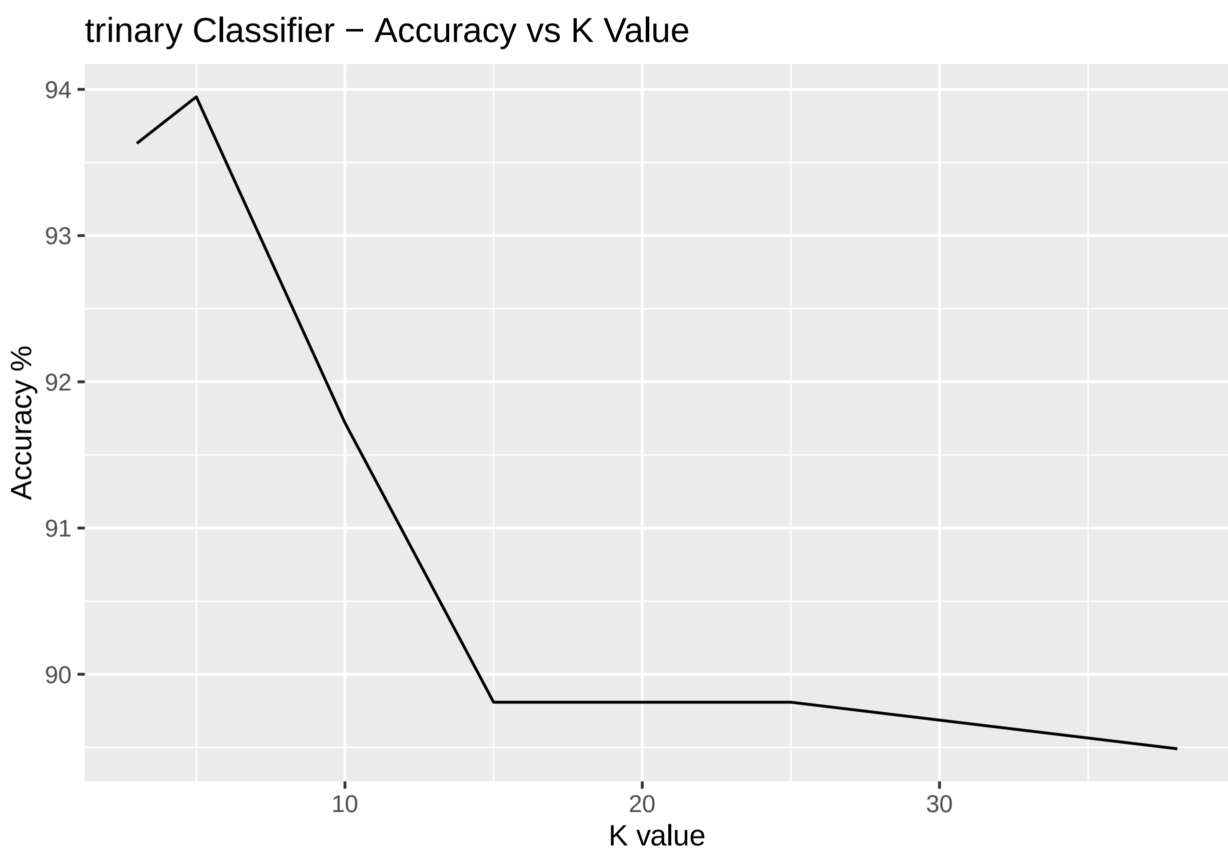
```r
k_trinarydata <- c(k_trinarydata, 25)
accurary_trinarydata<- c(accurary_trinarydata, acc.25)
# Calculate KNN for K=38 (square root of observations)
knn.38 <- knn(train = traindata.trinary , test = testdata_trinary, cl = traindata_trinary_df, k = 38)
# Calculate Accuracy
acc.38 <- 100* sum(testdata_trinary_df == knn.38)/NROW(testdata_trinary_df)
acc.38
```

```
## [1] 89.49045
```

```r
k_trinarydata <- c(k_trinarydata, 38)
accurary_trinarydata<- c(accurary_trinarydata, acc.38)
accuracy_trinarydata_df <- data.frame(k_trinarydata, accurary_trinarydata)
accuracy_trinarydata_df
```

```
##   k_trinarydata accurary_trinarydata
## 1             3             93.63057
## 2             5             93.94904
## 3            10             91.71975
## 4            15             89.80892
## 5            20             89.80892
## 6            25             89.80892
## 7            38             89.49045
```

```r
## K Values vs Accuracy Plotting
library(ggplot2)
ggplot(data = accuracy_trinarydata_df, aes(y = accurary_trinarydata, x = k_trinarydata)) + geom_line() +
```



Question C:

Looking back at the plots of the data, do you think a linear classifier would work well on these datasets?

Answer For C

Looking at the distribution of data in the plot, it is spread in different clusters so a linear classifier may not work with this data.