assignment_week11_KanaparthiVenkata

Venkata Kanaparthi

5/31/2021

Regression algorithms are used to predict numeric quantity while classification algorithms predict categorical outcomes. A spam filter is an example use case for a classification algorithm. The input dataset is emails labeled as either spam (i.e. junk emails) or ham (i.e. good emails). The classification algorithm uses features extracted from the emails to learn which emails fall into which category.

```
library(TSdist)
library(ggm)
library(ggplot2)
library(readr)
library(pastecs)
library(readxl)
library(plyr)
library(dplyr)
library(magrittr)
library(purrr)
library(stringr)
library(QuantPsyc)
library(caTools)
library(survival)
library(class)
library(spatstat)
```

Nearest Neighbors Algorithm on two data sets trinary-classifier-data.csv, binary-classifier-data.csv

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 (You worked with this dataset last week!). 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.

```
accuracy <- function(x){sum(diag(x)/(sum(rowSums(x)))) * 100}
```

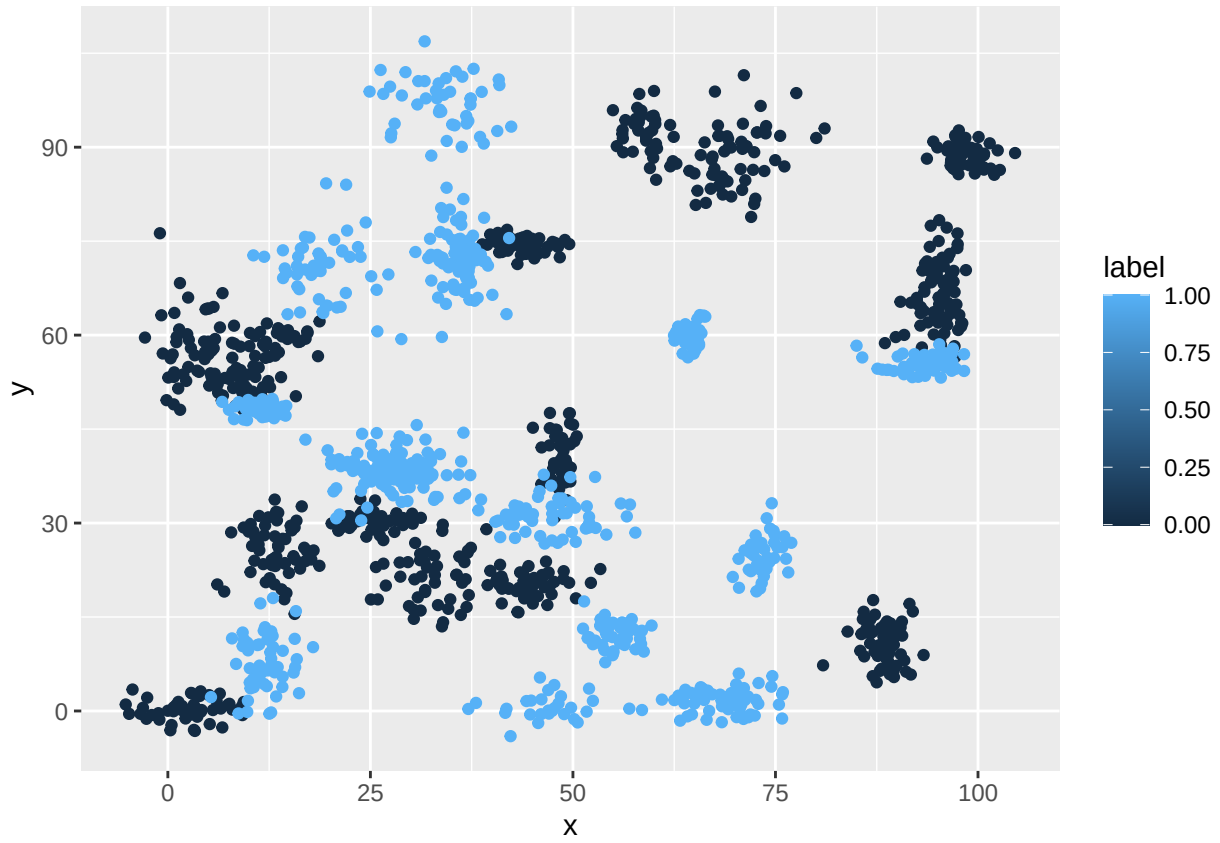
```
setwd('E:/MSDS-SEM2/DSC520/CodingAssignments/DSC520KANAPARTHI')
binaryClassifier_df <- read.csv("data/binary-classifier-data.csv", stringsAsFactors = FALSE)

a <- binaryClassifier_df[,c(2,3)]
dat <- sample(1:nrow(binaryClassifier_df), size = nrow(binaryClassifier_df)*0.7)

nor <-function(x) { (x -min(x))/(max(x)-min(x))   }
```

```
binaryClassifier_norm <- as.data.frame(lapply(a, nor))

bin_train_set <- binaryClassifier_norm[dat,]
bin_train_label <- binaryClassifier_df[dat, 1]

bin_test_set <- binaryClassifier_norm[-dat,]
bin_test_label <- binaryClassifier_df[-dat, 1]

bin_pr <- knn(bin_train_set, bin_test_set, cl = bin_train_label, k=32)
#bin_pr
```

```
setwd('E:/MSDS-SEM2/DSC520/CodingAssignments/DSC520KANAPARTHI')
trinaryClassifier_df <- read.csv("data/trinary-classifier-data.csv", stringsAsFactors = FALSE)

a <- trinaryClassifier_df[,c(2,3)]
dat <- sample(1:nrow(trinaryClassifier_df), size = nrow(trinaryClassifier_df)*0.7)

nor <-function(x) { (x -min(x))/(max(x)-min(x))   }
trinaryClassifier_norm <- as.data.frame(lapply(a, nor))
trin_train_set <- trinaryClassifier_norm[dat,]
trin_train_label <- trinaryClassifier_df[dat, 1]

trin_test_set <- trinaryClassifier_norm[-dat,]
trin_test_label <- trinaryClassifier_df[-dat, 1]
trin_pr <- knn(trin_train_set, trin_test_set, cl = trin_train_label, k=32)
#trin_pr
```

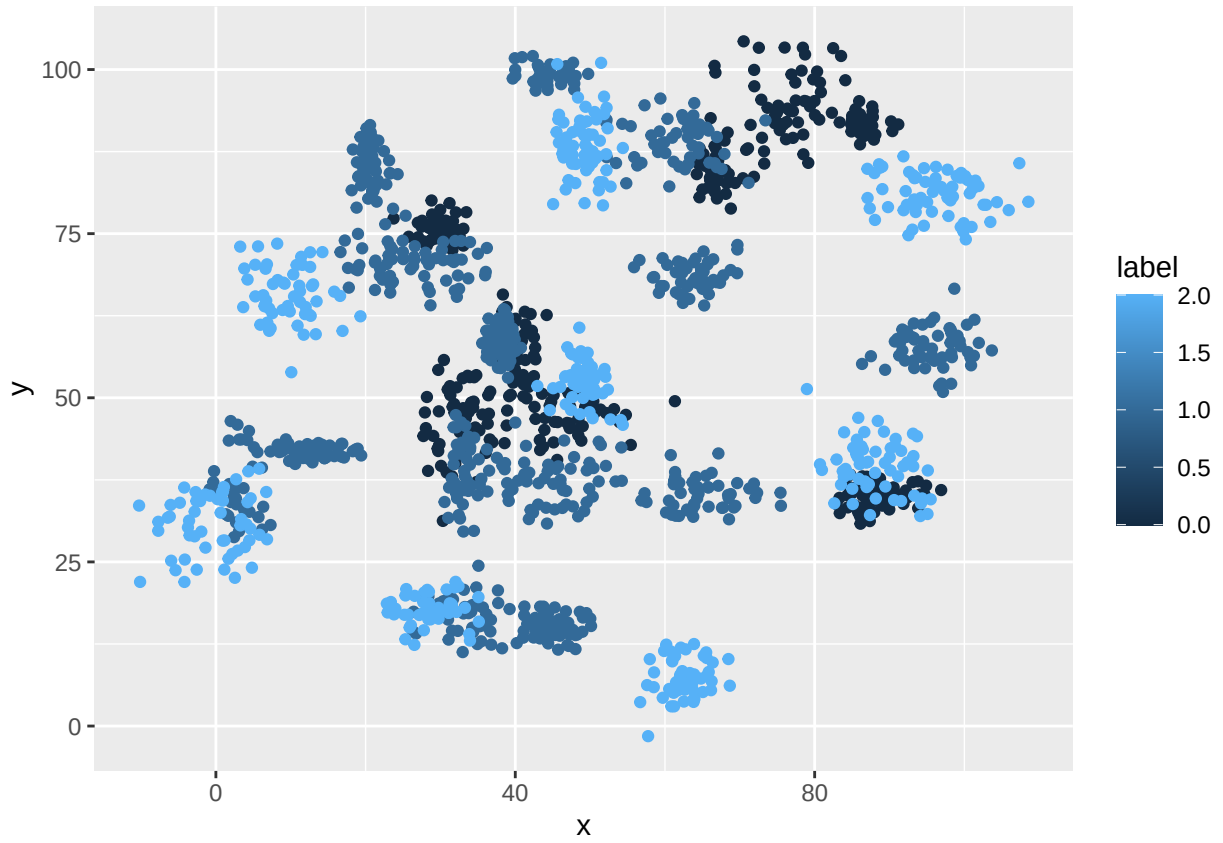
Plot Scatter plots on Binary Classifer Data Set

```
ggplot(data = binaryClassifier_df, aes(x=x, y=y, color=label)) +  geom_point()
```



Plot Scatter plots on Trinary Classifer Data Set

```
ggplot(data = trinaryClassifier_df, aes(x=x, y=y, color=label)) +  geom_point()
```



EuclideanDistance on binary Classifer Data Set

```
dist <- nndist(binaryClassifier_df)
```

EuclideanDistance on trinary Classifer Data Set

```
dist <- nndist(trinaryClassifier_df)
```

Model Accuracy calculation on Binary Classifer Data Set

```
bin_tab <- table(bin_pr, bin_test_label)
accuracy(bin_tab)
```

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

Model Accuracy calculation on Trinary Classifer Data Set

```
trin_tab <- table(trin_pr, trin_test_label)
accuracy(trin_tab)
```

```
## [1] 87.68577
```

Accuracy Model for kValue = 3

```
bin_pr_3 <- knn(bin_train_set, bin_test_set, cl = bin_train_label, k=3)
bin_3_tab <- table(bin_pr_3, bin_test_label)
accr_bin_3 <- accuracy(bin_3_tab)
```

Accuracy Model for kValue = 5

```
bin_pr_5 <- knn(bin_train_set, bin_test_set, cl = bin_train_label, k=5)
bin_5_tab <- table(bin_pr_5, bin_test_label)
accr_bin_5 <- accuracy(bin_5_tab)
```

Accuracy Model for kValue = 10

```
bin_pr_10 <- knn(bin_train_set, bin_test_set, cl = bin_train_label, k=10)
bin_10_tab <- table(bin_pr_10, bin_test_label)
accr_bin_10 <- accuracy(bin_10_tab)
```

Accuracy Model for kValue = 15

```
bin_pr_15 <- knn(bin_train_set, bin_test_set, cl = bin_train_label, k=15)
bin_15_tab <- table(bin_pr_15, bin_test_label)
accr_bin_15 <- accuracy(bin_15_tab)
```

Accuracy Model for kValue = 20

```
bin_pr_20 <- knn(bin_train_set, bin_test_set, cl = bin_train_label, k=20)
bin_20_tab <- table(bin_pr_20, bin_test_label)
accr_bin_20 <- accuracy(bin_20_tab)
```

Accuracy Model for kValue = 25

```
bin_pr_25 <- knn(bin_train_set, bin_test_set, cl = bin_train_label, k=25)
bin_25_tab <- table(bin_pr_25, bin_test_label)
accr_bin_25 <- accuracy(bin_25_tab)
```

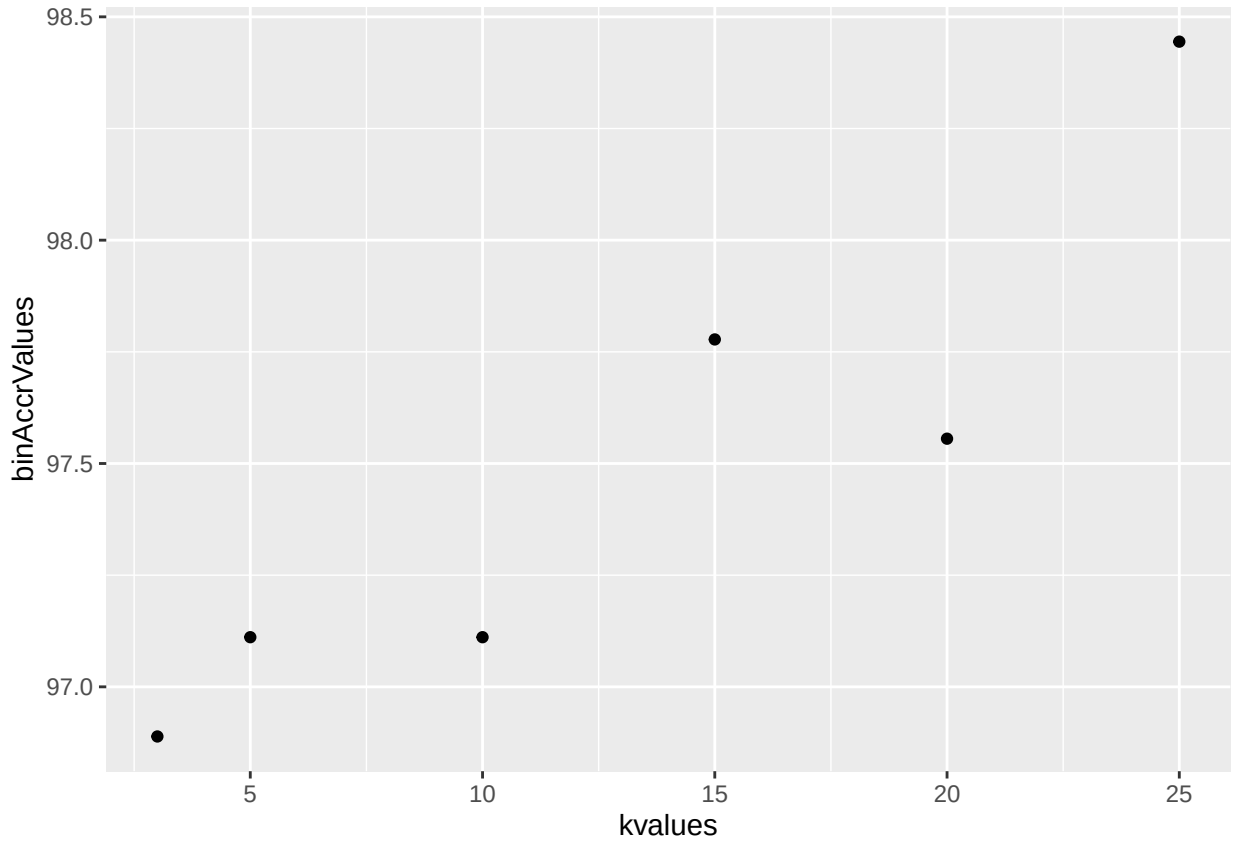
Creating a Dataset for kValues and Accuracy for Binary Set Data

```
kvalues <- c(3,5,10,15,20,25)
binAccrValues <- c(accr_bin_3,accr_bin_5,accr_bin_10,accr_bin_15,accr_bin_20,accr_bin_25)
dataSet <- data.frame(kvalues,binAccrValues)
dataSet
```

```
##   kvalues binAccrValues
## 1       3      96.88889
## 2       5      97.11111
## 3      10      97.11111
## 4      15      97.77778
## 5      20      97.55556
## 6      25      98.44444
```

Plotting Scatters for kValues and Accuracy for Binary Set Data

```
ggplot(data = dataSet, aes(x=kvalues, y=binAccrValues)) +  geom_point()
```




```
#geom_line(aes(y = cases, colour = "Florida")) +
```

Accuracy Model for kValue = 3

```
trin_pr_3 <- knn(trin_train_set, trin_test_set, cl = trin_train_label, k=3)
trin_3_tab <- table(trin_pr_3, trin_test_label)
accr_trin_3 <- accuracy(trin_3_tab)
```

Accuracy Model for kValue = 5

```
trin_pr_5 <- knn(trin_train_set, trin_test_set, cl = trin_train_label, k=5)
trin_5_tab <- table(trin_pr_5, trin_test_label)
accr_trin_5 <- accuracy(trin_5_tab)
```

Accuracy Model for kValue = 10

```
trin_pr_10 <- knn(trin_train_set, trin_test_set, cl = trin_train_label, k=10)
trin_10_tab <- table(trin_pr_10, trin_test_label)
accr_trin_10 <- accuracy(trin_10_tab)
```

Accuracy Model for kValue = 15

```
trin_pr_15 <- knn(trin_train_set, trin_test_set, cl = trin_train_label, k=15)
trin_15_tab <- table(trin_pr_15, trin_test_label)
accr_trin_15 <- accuracy(trin_15_tab)
```

Accuracy Model for kValue = 20

```
trin_pr_20 <- knn(trin_train_set, trin_test_set, cl = trin_train_label, k=20)
trin_20_tab <- table(trin_pr_20, trin_test_label)
accr_trin_20 <- accuracy(trin_20_tab)
```

Accuracy Model for kValue = 25

```
trin_pr_25 <- knn(trin_train_set, trin_test_set, cl = trin_train_label, k=25)
trin_25_tab <- table(trin_pr_25, trin_test_label)
accr_trin_25 <- accuracy(trin_25_tab)
```

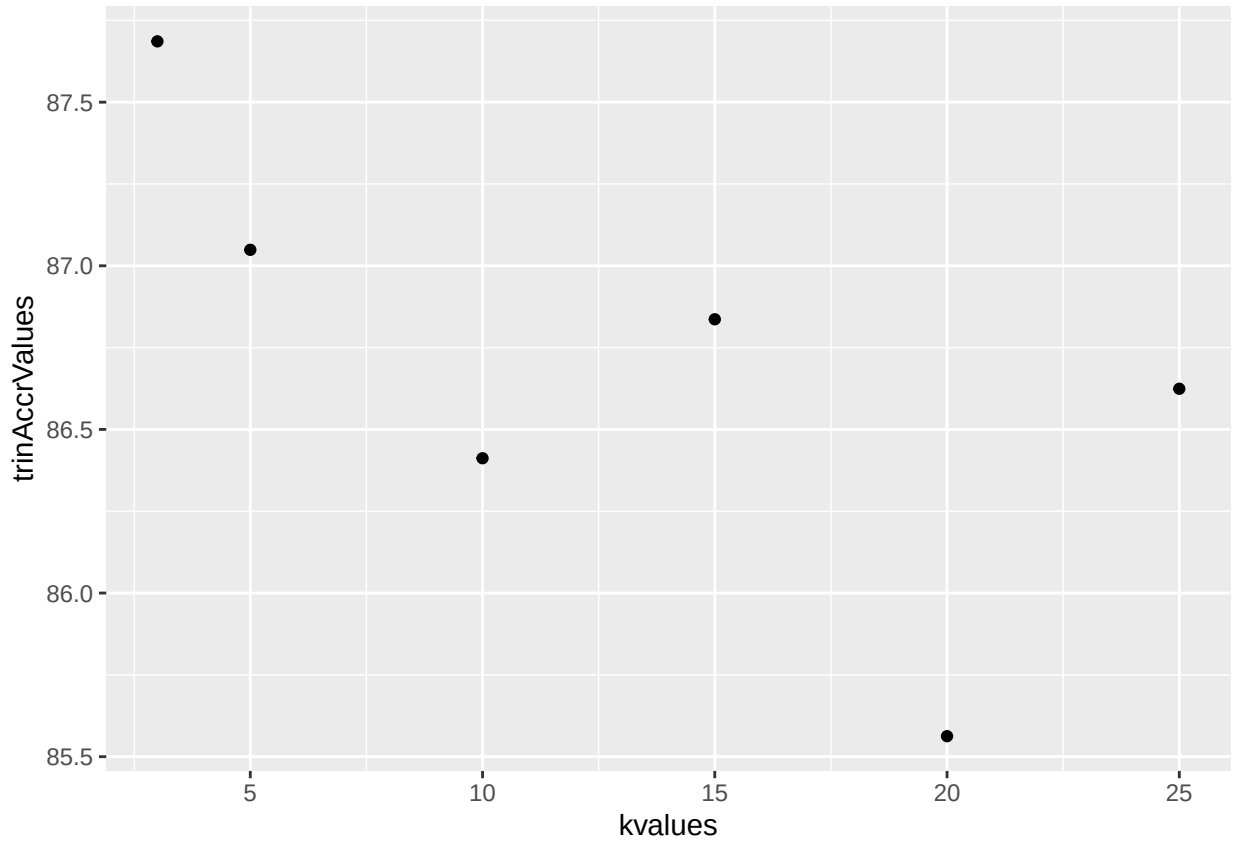
Creating a Dataset for kValues and Accuracy for Trinary Set Data

```
kvalues <- c(3,5,10,15,20,25)
trinAccrValues <- c(accr_trin_3,accr_trin_5,accr_trin_10,accr_trin_15,accr_trin_20,accr_trin_25)
dataSet <- data.frame(kvalues,trinAccrValues)
dataSet
```

```
##   kvalues trinAccrValues
## 1       3       87.68577
## 2       5       87.04883
## 3      10       86.41189
## 4      15       86.83652
## 5      20       85.56263
## 6      25       86.62420
```

Plotting Scatters for kValues and Accuracy for Trinary Set Data

```
ggplot(data = dataSet, aes(x=kvalues, y=trinAccrValues)) +  geom_point()
```



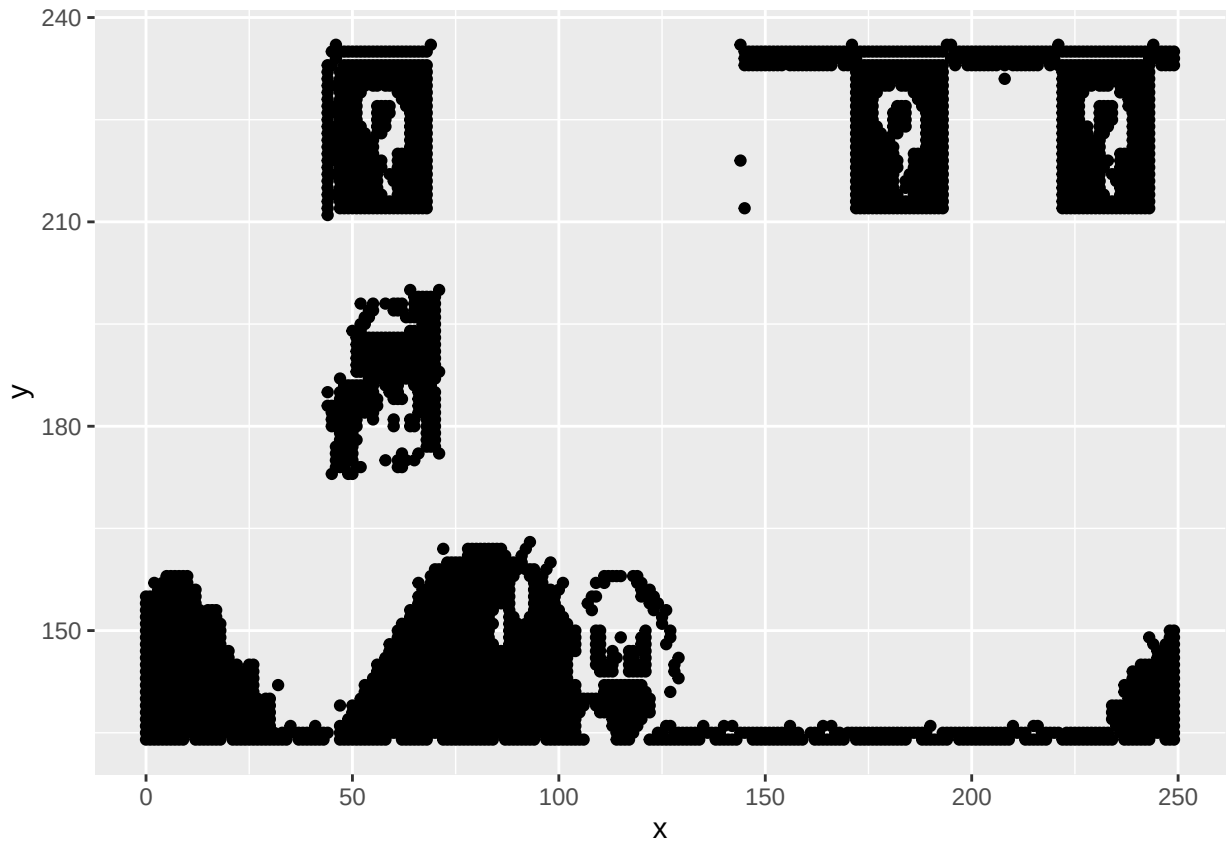
Looking back at the plots of the data, do you think a linear classifier would work well on these datasets? yes

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

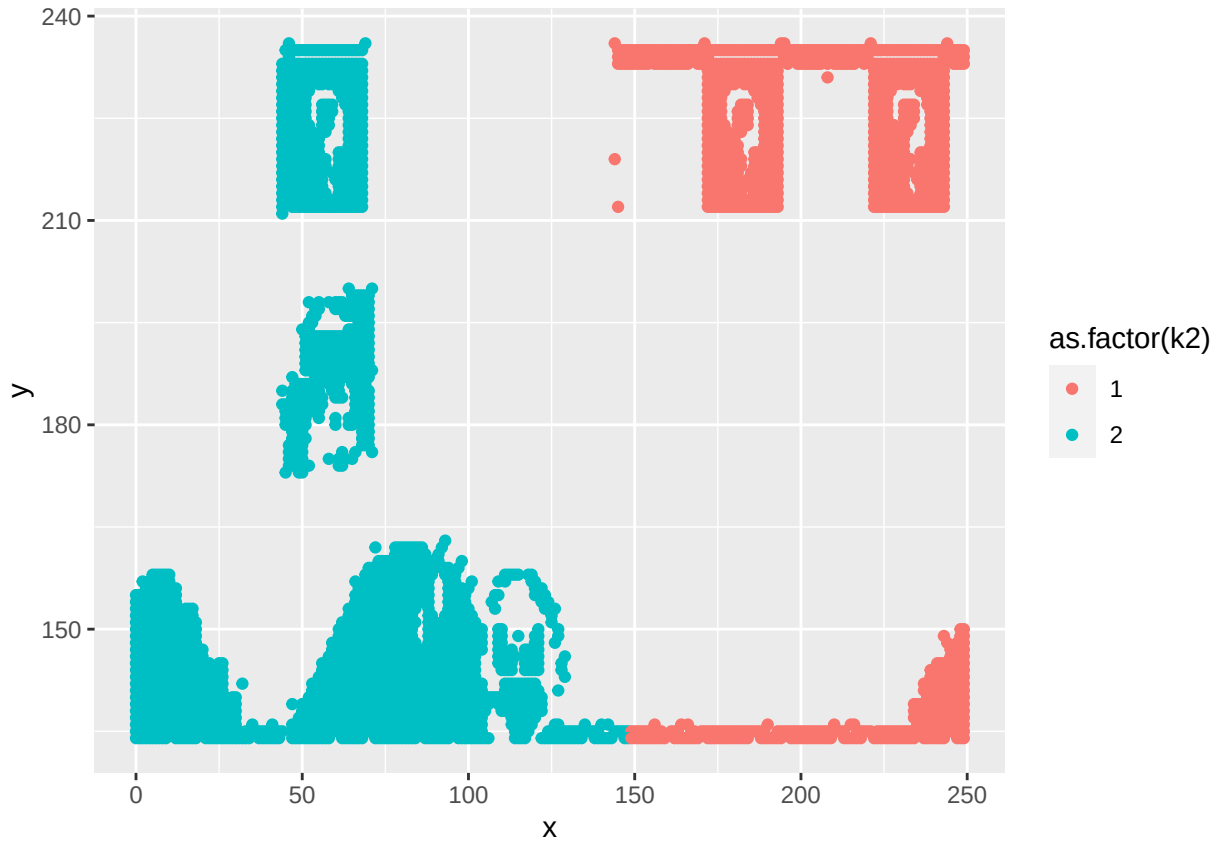
Accuracy increased by 39%.

KNN is a deterministic algorithm, meaning if we keep the value of K and run the algorithm n times, the results will be the same. But logistic regression is a stochastic algorithm. It means the algorithm use some random values to achieve it's goal. If we run the algorithm many times we will see the results varying.

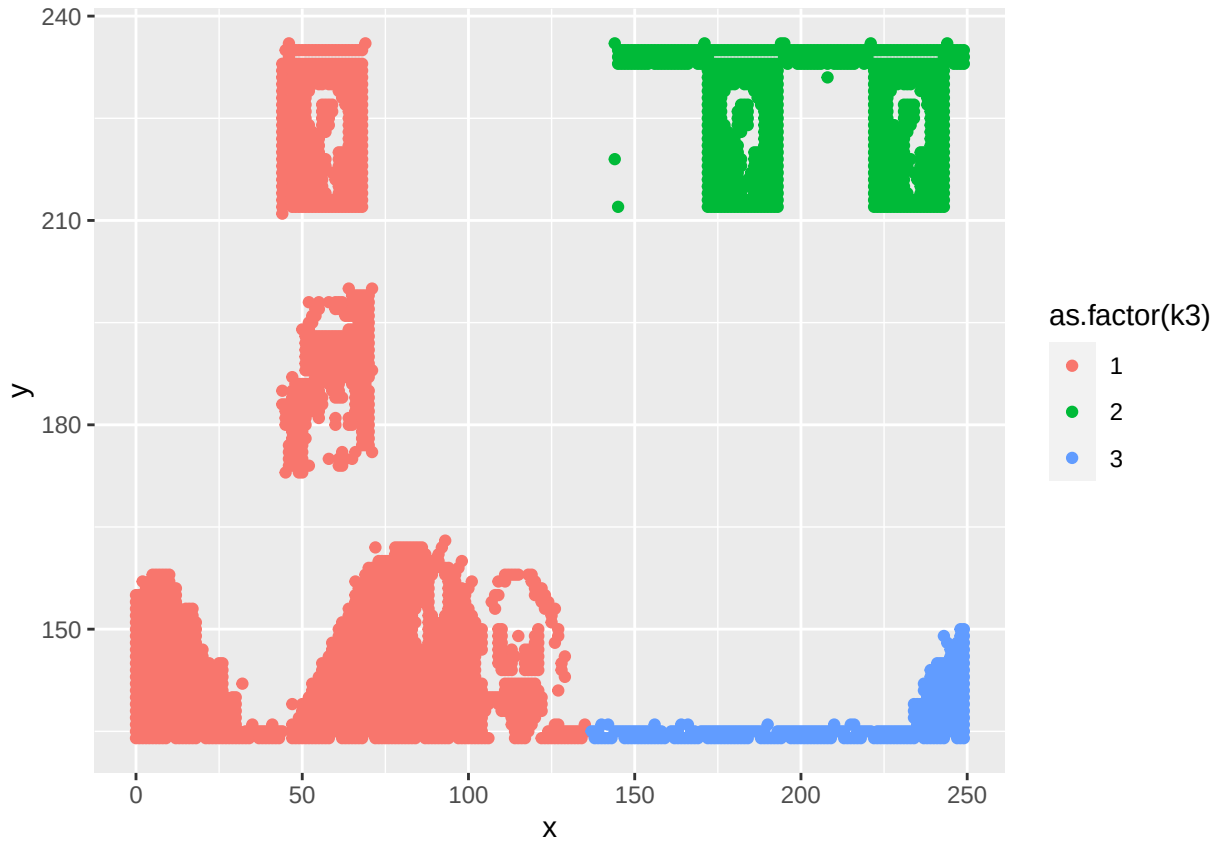
```
setwd('E:/MSDS-SEM2/DSC520/CodingAssignments/DSC520KANAPARTHI')
clustering_df <- read.csv("data/clustering-data.csv", stringsAsFactors = FALSE)
ggplot(data = clustering_df, aes(x=x, y=y)) +  geom_point()
```



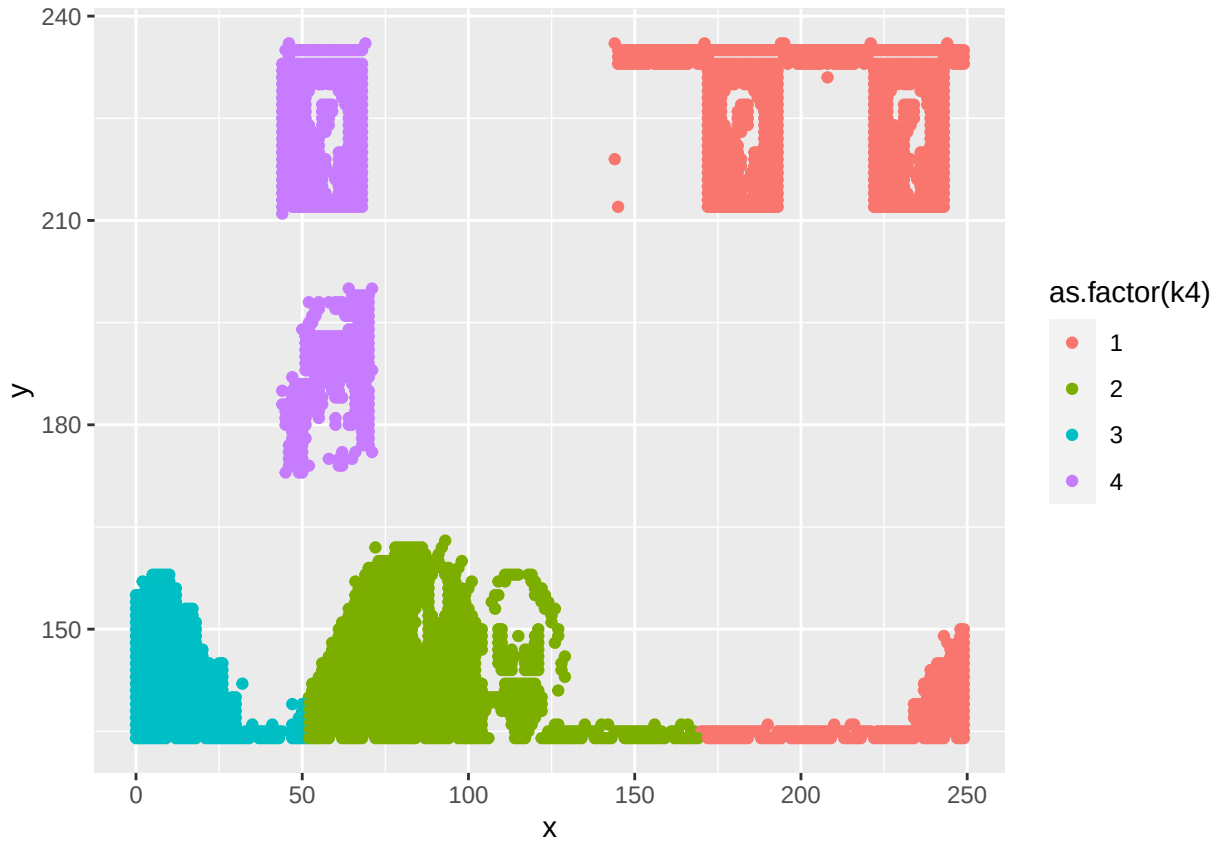

```
clusters2 <- kmeans(clustering_df, 2)
clustering_df$k2 <- as.factor(clusters2$cluster)
ggplot(data = clustering_df, aes(x=x, y=y)) + geom_point(aes(x = x, y = y, colour = as.factor(k2)),data
```



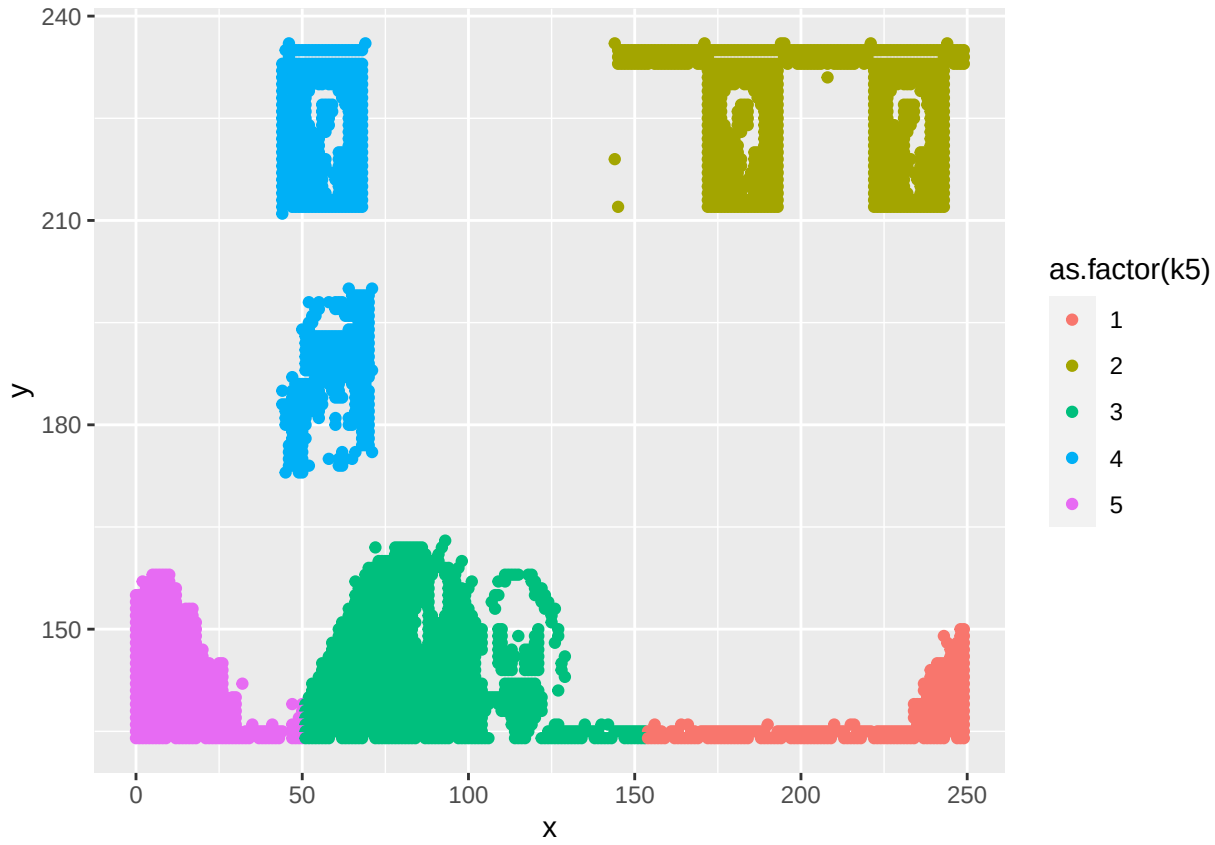

```
clusters3 <- kmeans(clustering_df, 3)
clustering_df$k3 <- as.factor(clusters3$cluster)
ggplot(data = clustering_df, aes(x=x, y=y)) + geom_point(aes(x = x, y = y, colour = as.factor(k3)),data
```



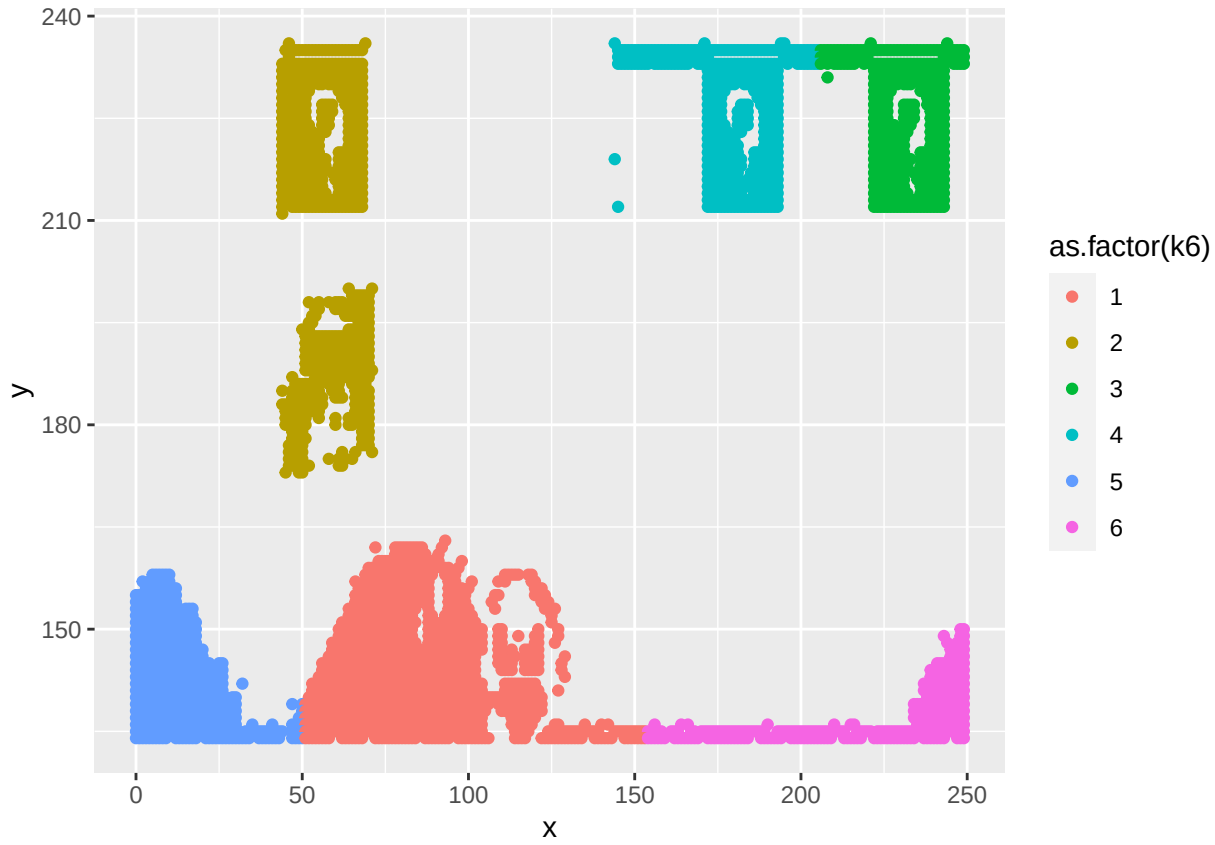

```
clusters4 <- kmeans(clustering_df, 4)
clustering_df$k4 <- as.factor(clusters4$cluster)
ggplot(data = clustering_df, aes(x=x, y=y)) + geom_point(aes(x = x, y = y, colour = as.factor(k4)),data
```



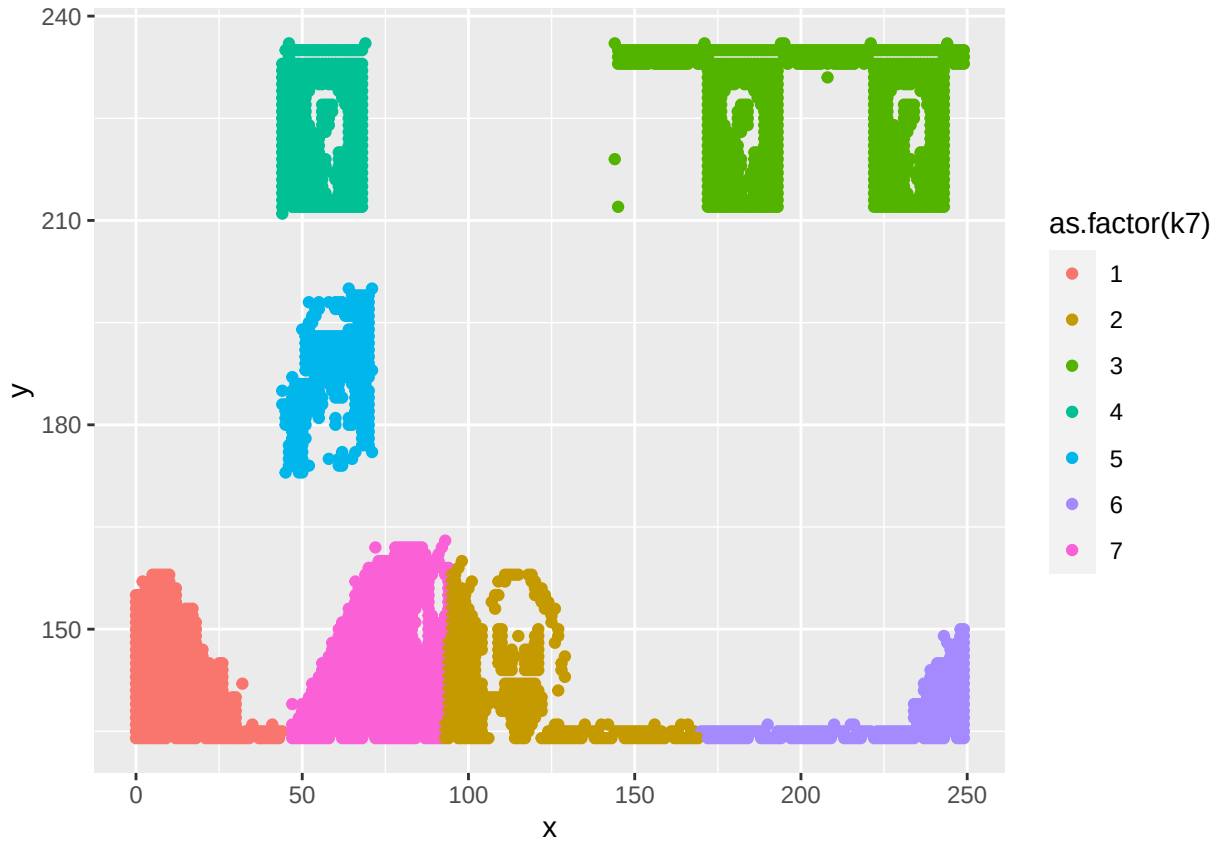

```
clusters5 <- kmeans(clustering_df, 5)
clustering_df$k5 <- as.factor(clusters5$cluster)
ggplot(data = clustering_df, aes(x=x, y=y)) + geom_point(aes(x = x, y = y, colour = as.factor(k5)),data
```



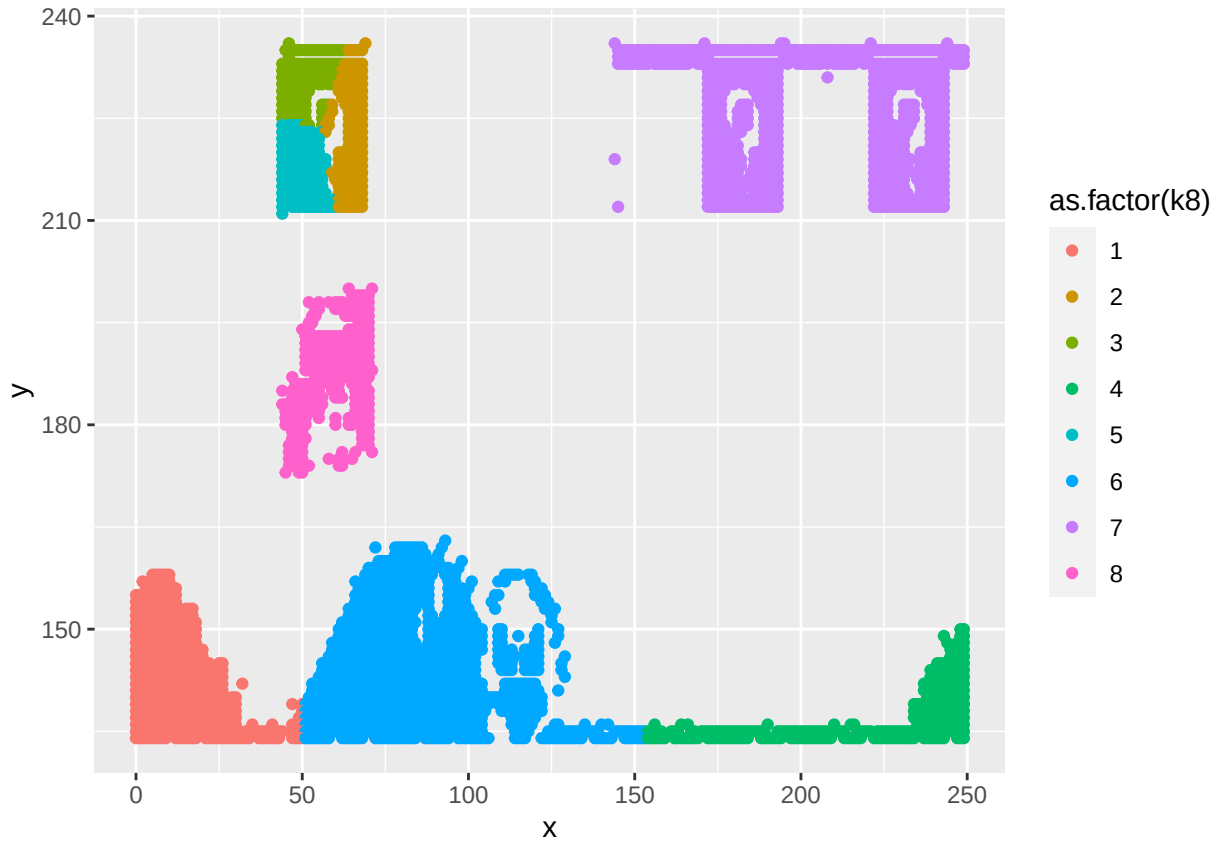

```
clusters6 <- kmeans(clustering_df, 6)
clustering_df$k6 <- as.factor(clusters6$cluster)
ggplot(data = clustering_df, aes(x=x, y=y)) + geom_point(aes(x = x, y = y, colour = as.factor(k6)),data
```



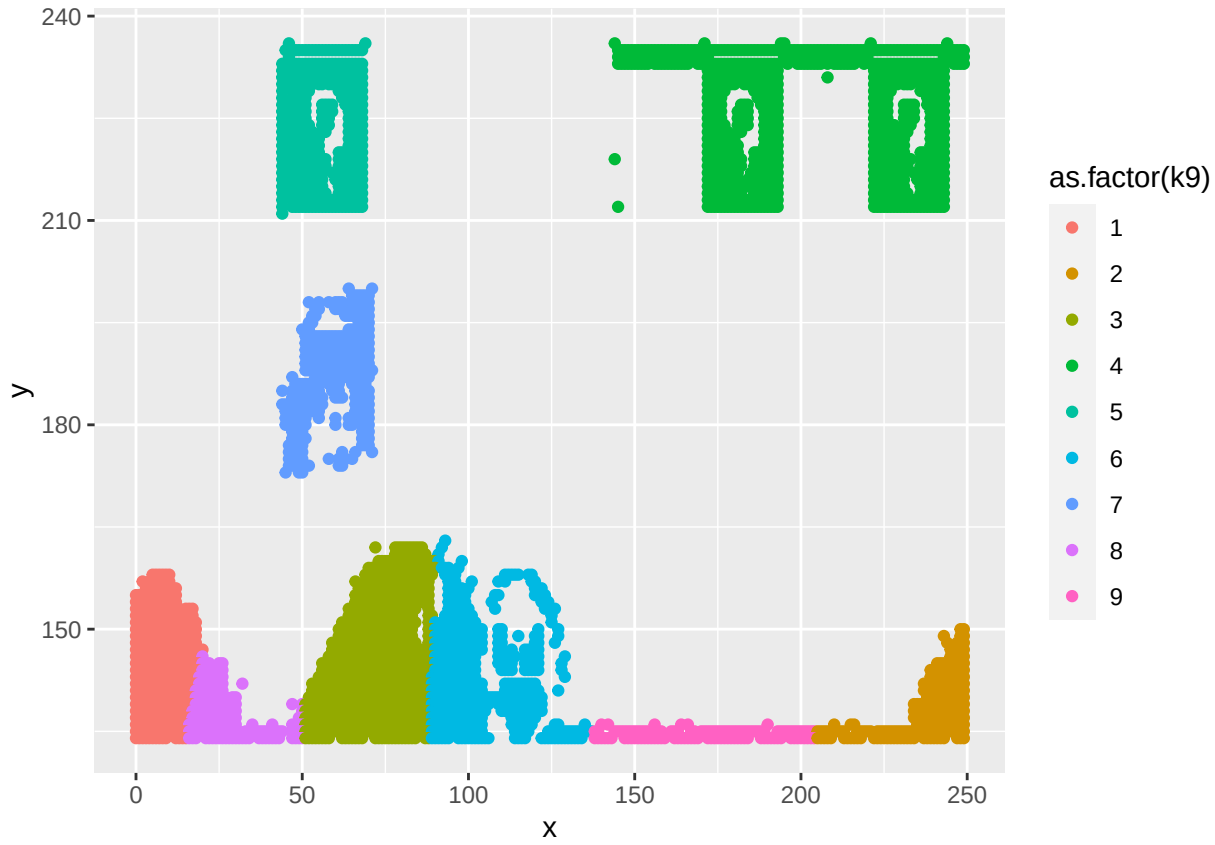

```
clusters7 <- kmeans(clustering_df, 7)
clustering_df$k7 <- as.factor(clusters7$cluster)
ggplot(data = clustering_df, aes(x=x, y=y)) + geom_point(aes(x = x, y = y, colour = as.factor(k7)),data
```



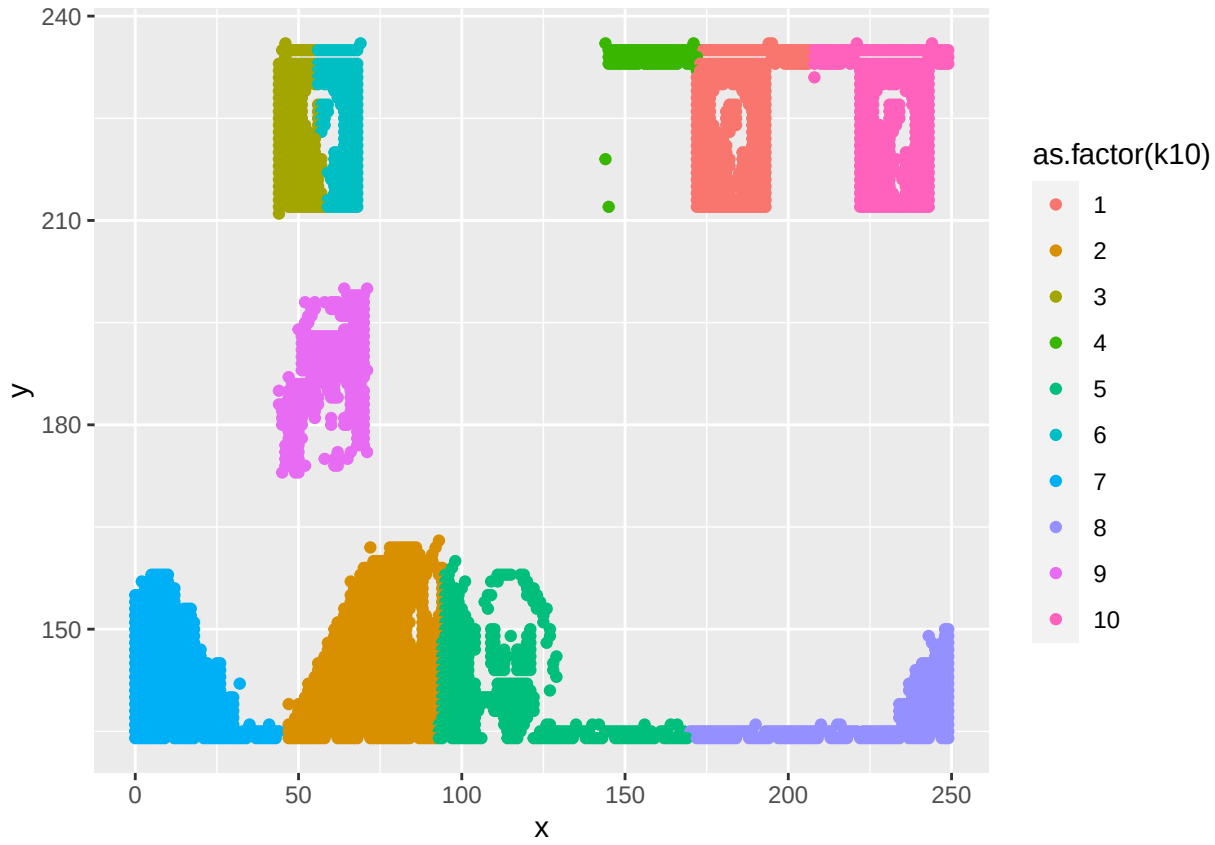

```
clusters8 <- kmeans(clustering_df, 8)
clustering_df$k8 <- as.factor(clusters8$cluster)
ggplot(data = clustering_df, aes(x=x, y=y)) + geom_point(aes(x = x, y = y, colour = as.factor(k8)),data
```



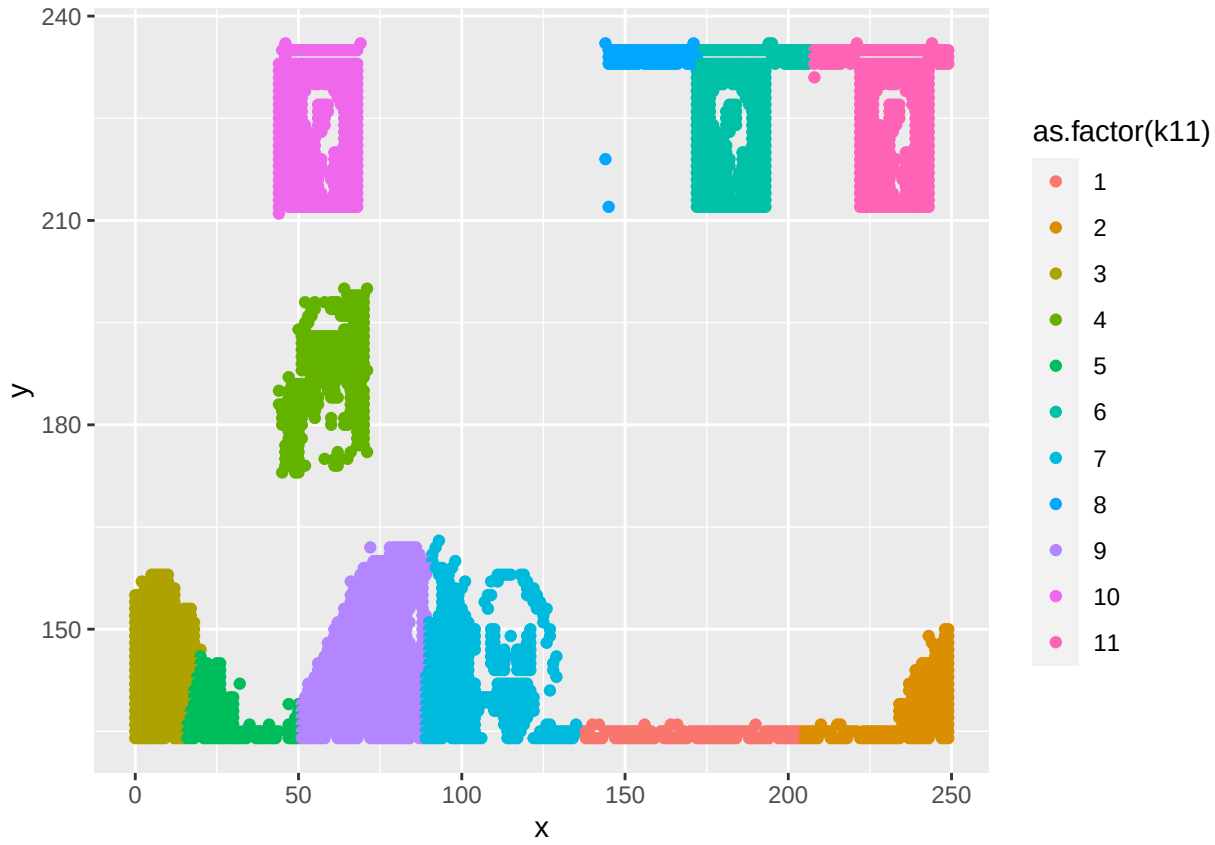

```
clusters9 <- kmeans(clustering_df, 9)
clustering_df$k9 <- as.factor(clusters9$cluster)
ggplot(data = clustering_df, aes(x=x, y=y)) + geom_point(aes(x = x, y = y, colour = as.factor(k9)),data
```



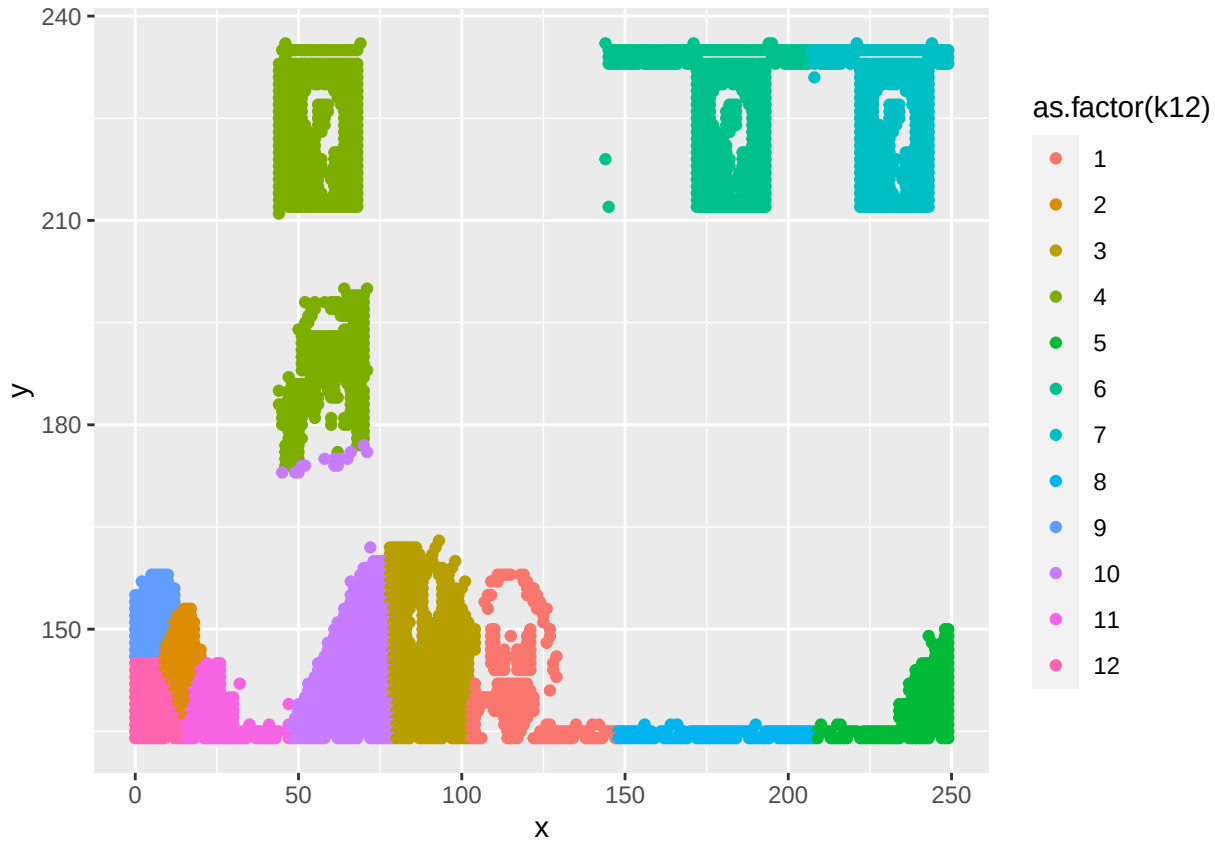

```
clusters10 <- kmeans(clustering_df, 10)
clustering_df$k10 <- as.factor(clusters10$cluster)
ggplot(data = clustering_df, aes(x=x, y=y)) + geom_point(aes(x = x, y = y, colour = as.factor(k10)),data
```



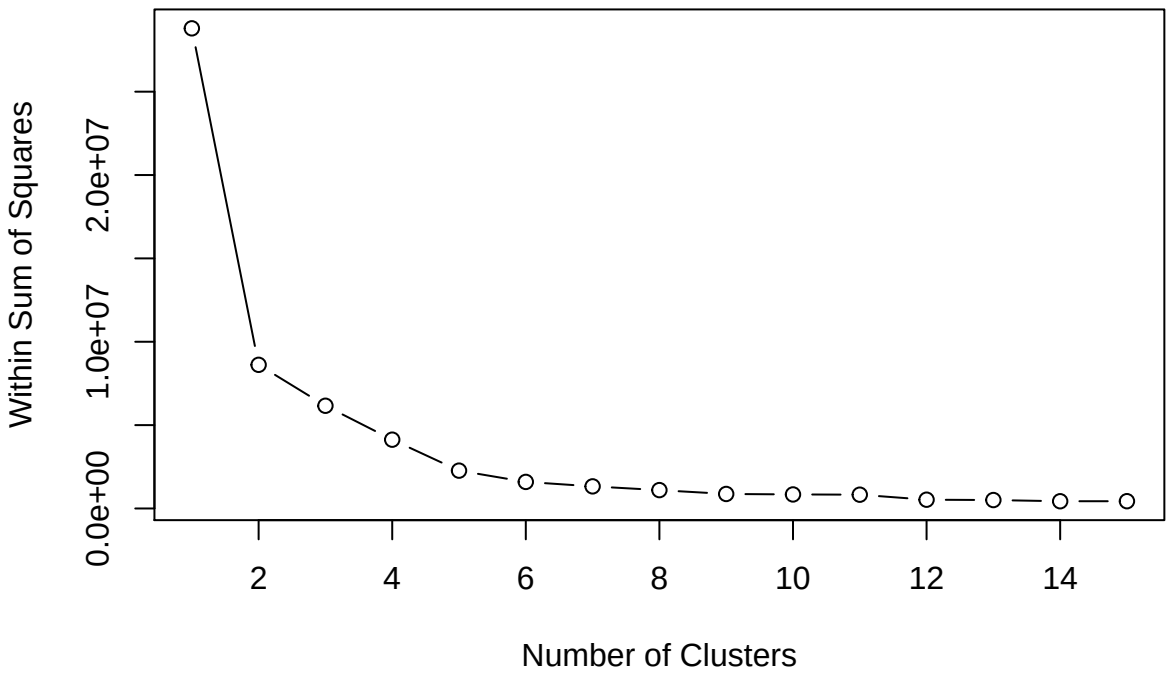

```
clusters11 <- kmeans(clustering_df, 11)
clustering_df$k11 <- as.factor(clusters11$cluster)
ggplot(data = clustering_df, aes(x=x, y=y)) + geom_point(aes(x = x, y = y, colour = as.factor(k11)),data
```




```
clusters12 <- kmeans(clustering_df, 12)
clustering_df$k12 <- as.factor(clusters12$cluster)
ggplot(data = clustering_df, aes(x=x, y=y)) + geom_point(aes(x = x, y = y, colour = as.factor(k12)),data
```




```
withInSumSq <- (nrow(clustering_df)-1)*sum(apply(clustering_df,2,var))
for (i in 2:15) withInSumSq[i] <- sum(kmeans(clustering_df, centers=i)$withinss)
plot(1:15, withInSumSq, type = "b", xlab="Number of Clusters", ylab = "Within Sum of Squares")
```



We can consider 5 as the K-value because there isnt any change after that