Assignment: ASSIGNMENT 11.2

Name: Shekhar , Manish

Date: 2021-05-25

1. Introduction to Machine Learning

Reading the datasets binary classifier and trinary classifier

```
# Read the binary classifier data file into data frame
# I copied it to working directory
bin_cls_data <- read.csv('binary-classifier-data.csv')
# check the structure of the data
str(bin_cls_data)
```

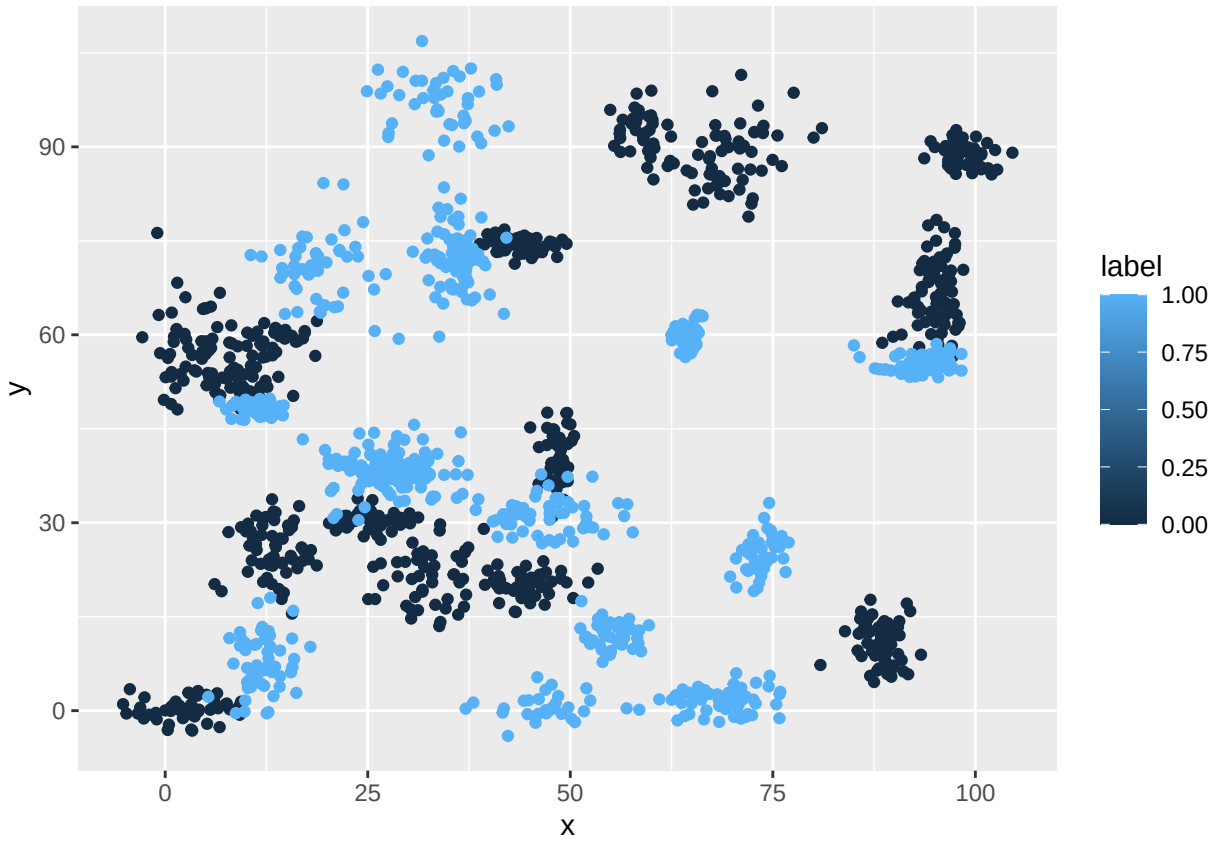
```
## 'data.frame':    1498 obs. of  3 variables:
##  $ label: int  0 0 0 0 0 0 0 0 0 0 ...
##  $ x    : num  70.9 75 73.8 66.4 69.1 ...
##  $ y    : num  83.2 87.9 92.2 81.1 84.5 ...
```

```
# Read the trinary classifier data file into data frame
# I copied it to working directory
tri_cls_data <- read.csv('trinary-classifier-data.csv')
# check the structure of the data
str(tri_cls_data)
```

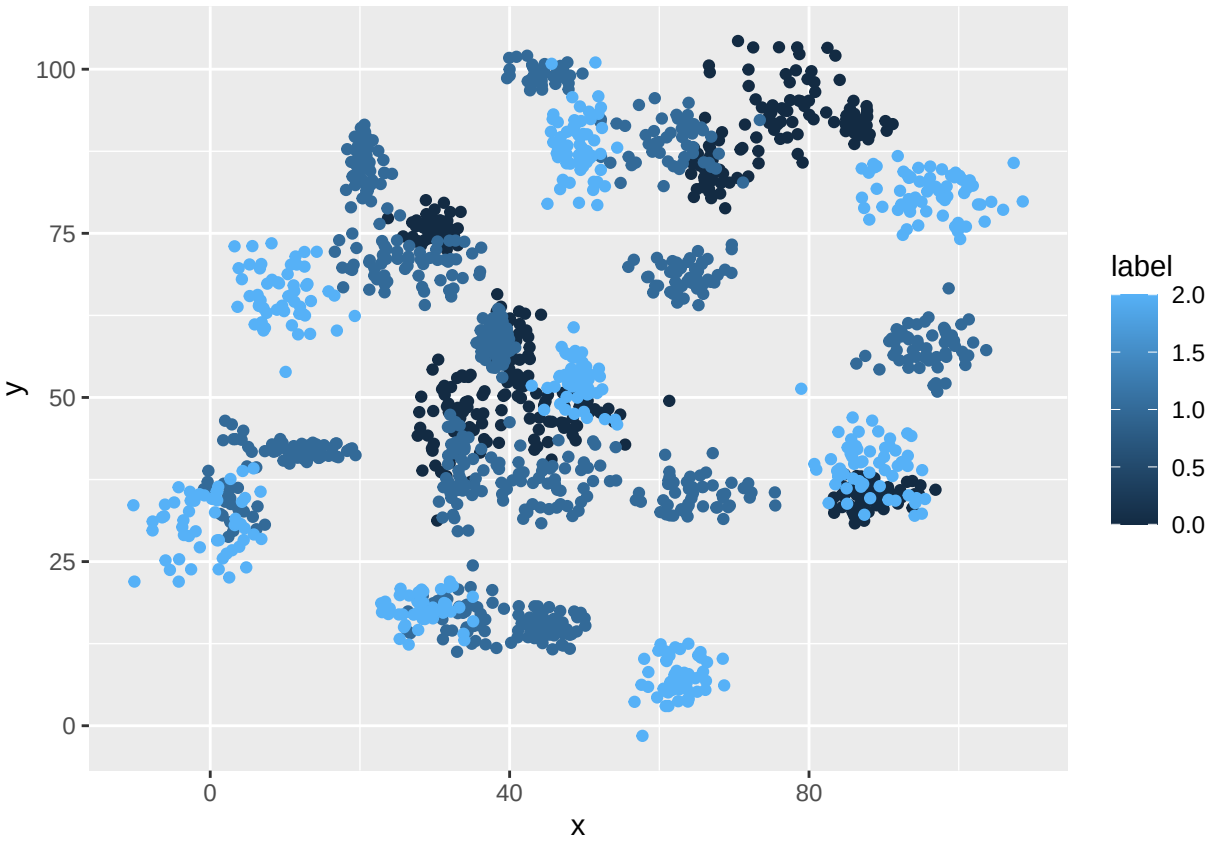
```
## 'data.frame':    1568 obs. of  3 variables:
##  $ label: int  0 0 0 0 0 0 0 0 0 0 ...
##  $ x    : num  30.1 31.3 34.1 32.6 34.7 ...
##  $ y    : num  39.6 51.8 49.3 41.2 45.5 ...
```

Plotting the data using scatterplot

```
library(ggplot2)
# plotting binary classifier data
ggplot(data = bin_cls_data, aes(x = x, y = y, colour = label)) +
    geom_point()
```




```
# plotting trinary classifier data
ggplot(data = tri_cls_data, aes(x = x, y = y, colour = label)) +
    geom_point()
```



Creating training and test set and fitting the model for each binary class dataset for k=3, k=5, k=10, k=20, and k=25.

```
# binary classifier data
# encode the target feature as factor
bin_cls_data$label = factor(bin_cls_data$label, levels = c(0,1))

# splitting the data into training and test set
library(caTools)
set.seed(123)
split = sample.split(bin_cls_data$label, SplitRatio = 0.9)
training_set = subset(bin_cls_data,split == TRUE)
test_set = subset(bin_cls_data,split == FALSE)

# feature scaling
training_set[-1] = scale(training_set[-1])
test_set[-1] = scale(test_set[-1])


# Fitting KNN classifier to training set and predicting the test set results
# installing and loading the class library for KNN classifier
# install.packages("class")
library(class)
ypred = knn(train = training_set[,-1],
            test = test_set[,-1],
```

```
            cl = training_set[,1],
            k = 3)

# Making the confusion matrix
cm = table(test_set[,1], ypred)
cm
```

```
##    ypred
##      0  1
##   0 69  8
##   1  9 64
```

```
# Accuracy of the model
# accuracy = total correct prediction / total number of observation
tot_obs = nrow(test_set)
accuracy_bin_k3 = (69+64)/tot_obs
accuracy_bin_k3
```

```
## [1] 0.8866667
```

```
# 88.67% accuracy using k=3

# create a data frame to store number of K data points used for model fitting and accuracy of the model
bin_data_to_plot <- data.frame(cbind(3, accuracy_bin_k3))
colnames(bin_data_to_plot) <- c("k", "accuracy")
bin_data_to_plot
```

```
##   k  accuracy
## 1 3 0.8866667
```

```
# -------------------------- k = 5 --------------------
# fitting the model and getting the prediction back
ypred_5 = knn(train = training_set[,-1],
            test = test_set[,-1],
            cl = training_set[,1],
            k = 5)

# Making the confusion matrix
cm_5 = table(test_set[,1], ypred_5)
cm_5
```

```
##    ypred_5
##      0  1
##   0 67 10
##   1  8 65
```

```
# Accuracy of the model
# accuracy = total correct prediction / total number of observation
accuracy_bin_k5 = (67+65)/tot_obs
accuracy_bin_k5
```

```
## [1] 0.88
```

```
# 88% accuracy using k=5

# appending to the data frame
bin_data_to_plot <- rbind(bin_data_to_plot, c(5, accuracy_bin_k5))
```

```
bin_data_to_plot
```

```
##   k  accuracy
## 1 3 0.8866667
## 2 5 0.8800000
```

```
# -------------------------- k = 10 --------------------
# fitting the model and getting the prediction back
ypred_10 = knn(train = training_set[,-1],
            test = test_set[,-1],
            cl = training_set[,1],
            k = 10)

# Making the confusion matrix
cm_10 = table(test_set[,1], ypred_10)
cm_10
```

```
##    ypred_10
##      0  1
##   0 67 10
##   1  8 65
```

```
# Accuracy of the model
# accuracy = total correct prediction / total number of observation
accuracy_bin_k10 = (67+65)/tot_obs
accuracy_bin_k10
```

```
## [1] 0.88
```

```
# 88% accuracy using k=10

# appending to the data frame
bin_data_to_plot <- rbind(bin_data_to_plot, c(10, accuracy_bin_k10))
bin_data_to_plot
```

```
##    k  accuracy
## 1  3 0.8866667
## 2  5 0.8800000
## 3 10 0.8800000
```

```
# -------------------------- k = 20 --------------------
# fitting the model and getting the prediction back
ypred_20 = knn(train = training_set[,-1],
            test = test_set[,-1],
            cl = training_set[,1],
            k = 20)

# Making the confusion matrix
cm_20 = table(test_set[,1], ypred_20)
cm_20
```

```
##    ypred_20
##      0  1
##   0 67 10
##   1  9 64
```

```r
# Accuracy of the model
# accuracy = total correct prediction / total number of observation
accuracy_bin_k20 = (67+64)/tot_obs
accuracy_bin_k20
```

```
## [1] 0.8733333
```

```r
# 87.33% accuracy using k=20

# appending to the data frame
bin_data_to_plot <- rbind(bin_data_to_plot, c(20, accuracy_bin_k20))
bin_data_to_plot
```

```
##    k  accuracy
## 1  3 0.8866667
## 2  5 0.8800000
## 3 10 0.8800000
## 4 20 0.8733333
```

```r
# -------------------------- k = 25 --------------------
# fitting the model and getting the prediction back
ypred_25 = knn(train = training_set[,-1],
            test = test_set[,-1],
            cl = training_set[,1],
            k = 25)

# Making the confusion matrix
cm_25 = table(test_set[,1], ypred_25)
cm_25
```

```
##    ypred_25
##      0  1
##   0 68  9
##   1  6 67
```

```r
# Accuracy of the model
# accuracy = total correct prediction / total number of observation
accuracy_bin_k25 = (68+67)/tot_obs
accuracy_bin_k25
```

```
## [1] 0.9
```

```r
# 90% accuracy using k=25

# appending to the data frame
bin_data_to_plot <- rbind(bin_data_to_plot, c(25, accuracy_bin_k25))
bin_data_to_plot
```

```
##    k  accuracy
## 1  3 0.8866667
## 2  5 0.8800000
## 3 10 0.8800000
## 4 20 0.8733333
## 5 25 0.9000000
```

Creating training and test set and fitting the model for each trinary class dataset for k=3, k=5, k=10, k=20, and k=25.

```
# trinary classifier data
# encode the target feature as factor
tri_cls_data$label = factor(tri_cls_data$label, levels = c(0,1,2))

# splitting the data into training and test set
library(caTools)
set.seed(123)
split = sample.split(tri_cls_data$label, SplitRatio = 0.9)
training_set = subset(tri_cls_data,split == TRUE)
test_set = subset(tri_cls_data,split == FALSE)

# feature scaling
training_set[-1] = scale(training_set[-1])
test_set[-1] = scale(test_set[-1])


# Fitting KNN classifier to training set and predicting the test set results
ypred_tri = knn(train = training_set[,-1],
            test = test_set[,-1],
            cl = training_set[,1],
            k = 3)

# Making the confusion matrix
cm_tri = table(test_set[,1], ypred_tri)
cm_tri
```

```
##    ypred_tri
##      0  1  2
##   0 32  4  3
##   1  2 69  1
##   2  2  2 41
```

```
# Accuracy of the model
# accuracy = total correct prediction / total number of observation
tot_obs = nrow(test_set)
accuracy_tri_k3 = (35+67+36)/tot_obs
accuracy_tri_k3
```

```
## [1] 0.8846154
```

```
# 88.47% accuracy using k=3

# create a data frame to store number of K data points used for model fitting and accuracy of the model
tri_data_to_plot <- data.frame(cbind(3, accuracy_tri_k3))
colnames(tri_data_to_plot) <- c("k", "accuracy")
tri_data_to_plot
```

```
##   k  accuracy
## 1 3 0.8846154
```

```
# ------------------------- k = 5 --------------------
# Fitting KNN classifier to training set and predicting the test set results
```

```r
ypred_tri_5 = knn(train = training_set[,-1],
            test = test_set[,-1],
            cl = training_set[,1],
            k = 5)

# Making the confusion matrix
cm_tri_5 = table(test_set[,1], ypred_tri_5)
cm_tri_5
```

```
##    ypred_tri_5
##      0  1  2
##   0 34  4  1
##   1  3 68  1
##   2  2  2 41
```

```r
# Accuracy of the model
# accuracy = total correct prediction / total number of observation
accuracy_tri_k5 = (33+66+39)/tot_obs
accuracy_tri_k5
```

```
## [1] 0.8846154
```

```r
# 88.46 % accuracy using k=5

# appending to the data frame
tri_data_to_plot <- rbind(tri_data_to_plot, c(5, accuracy_tri_k5))
tri_data_to_plot
```

```
##   k  accuracy
## 1 3 0.8846154
## 2 5 0.8846154
```

```r
# ------------------------- k = 10 -------------------
# Fitting KNN classifier to training set and predicting the test set results
ypred_tri_10 = knn(train = training_set[,-1],
            test = test_set[,-1],
            cl = training_set[,1],
            k = 10)

# Making the confusion matrix
cm_tri_10 = table(test_set[,1], ypred_tri_10)
cm_tri_10
```

```
##    ypred_tri_10
##      0  1  2
##   0 33  5  1
##   1  6 66  0
##   2  2  2 41
```

```r
# Accuracy of the model
# accuracy = total correct prediction / total number of observation
accuracy_tri_k10 = (35+67+36)/tot_obs
accuracy_tri_k10
```

```
## [1] 0.8846154
```

```r
# 88.46% accuracy using k=10

# appending to the data frame
tri_data_to_plot <- rbind(tri_data_to_plot, c(10, accuracy_tri_k10))
tri_data_to_plot
```

```
##    k  accuracy
## 1  3 0.8846154
## 2  5 0.8846154
## 3 10 0.8846154
```

```r
# ------------------------- k = 20 -------------------
# Fitting KNN classifier to training set and predicting the test set results
ypred_tri_20 = knn(train = training_set[,-1],
            test = test_set[,-1],
            cl = training_set[,1],
            k = 20)

# Making the confusion matrix
cm_tri_20 = table(test_set[,1], ypred_tri_20)
cm_tri_20
```

```
##    ypred_tri_20
##      0  1  2
##   0 31  7  1
##   1  5 67  0
##   2  1  3 41
```

```r
# Accuracy of the model
# accuracy = total correct prediction / total number of observation
accuracy_tri_k20 = (33+65+40)/tot_obs
accuracy_tri_k20
```

```
## [1] 0.8846154
```

```r
# 88.46% accuracy using k=20

# appending to the data frame
tri_data_to_plot <- rbind(tri_data_to_plot, c(20, accuracy_tri_k20))
tri_data_to_plot
```

```
##    k  accuracy
## 1  3 0.8846154
## 2  5 0.8846154
## 3 10 0.8846154
## 4 20 0.8846154
```

```r
# ------------------------- k = 25 -------------------
# Fitting KNN classifier to training set and predicting the test set results
ypred_tri_25 = knn(train = training_set[,-1],
            test = test_set[,-1],
            cl = training_set[,1],
            k = 25)

# Making the confusion matrix
cm_tri_25 = table(test_set[,1], ypred_tri_25)
cm_tri_25
```

```
##    ypred_tri_25
##       0  1  2
##   0 32  6  1
##   1  5 67  0
##   2  0  2 43
```

```
# Accuracy of the model
# accuracy = total correct prediction / total number of observation
accuracy_tri_k25 = (33+67+39)/tot_obs
accuracy_tri_k25
```

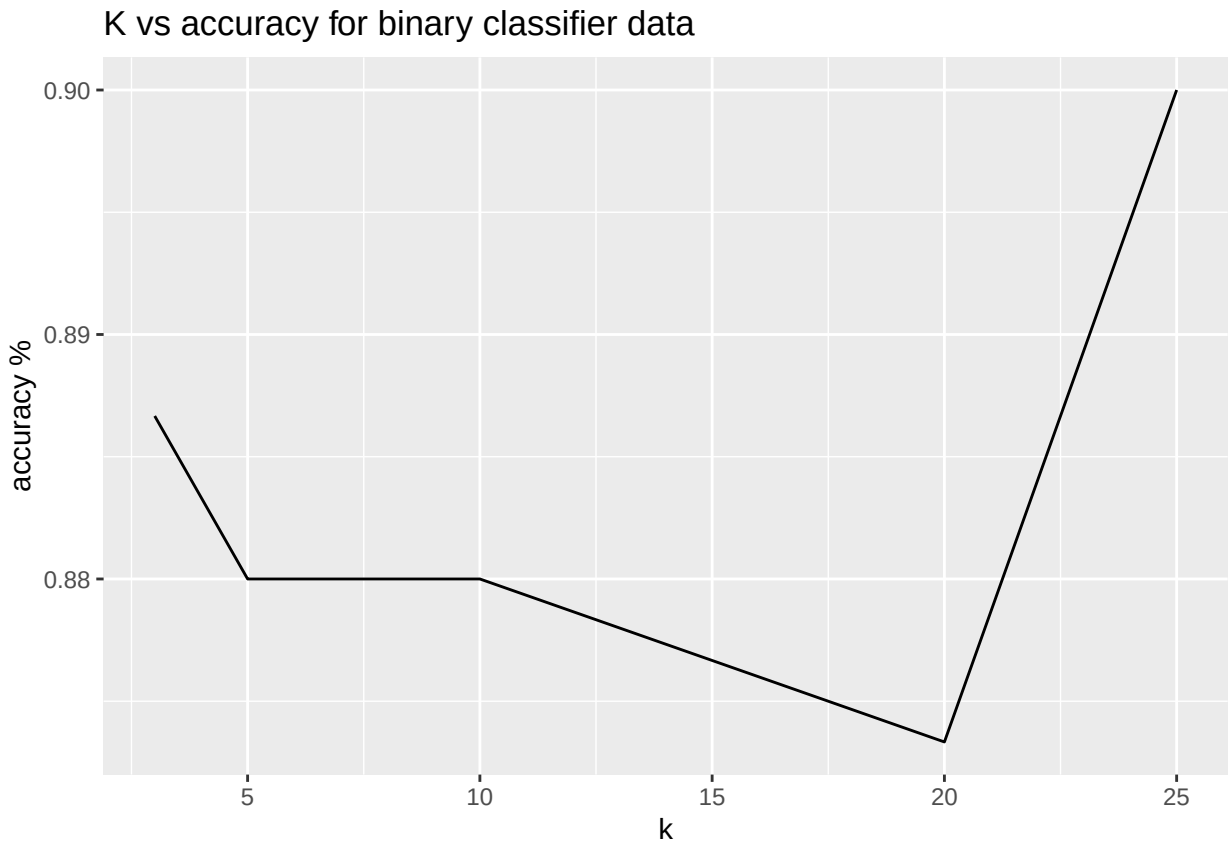
```
## [1] 0.8910256
```

```
# 89.1% accuracy using k=25

# appending to the data frame
tri_data_to_plot <- rbind(tri_data_to_plot, c(25, accuracy_tri_k25))
tri_data_to_plot
```

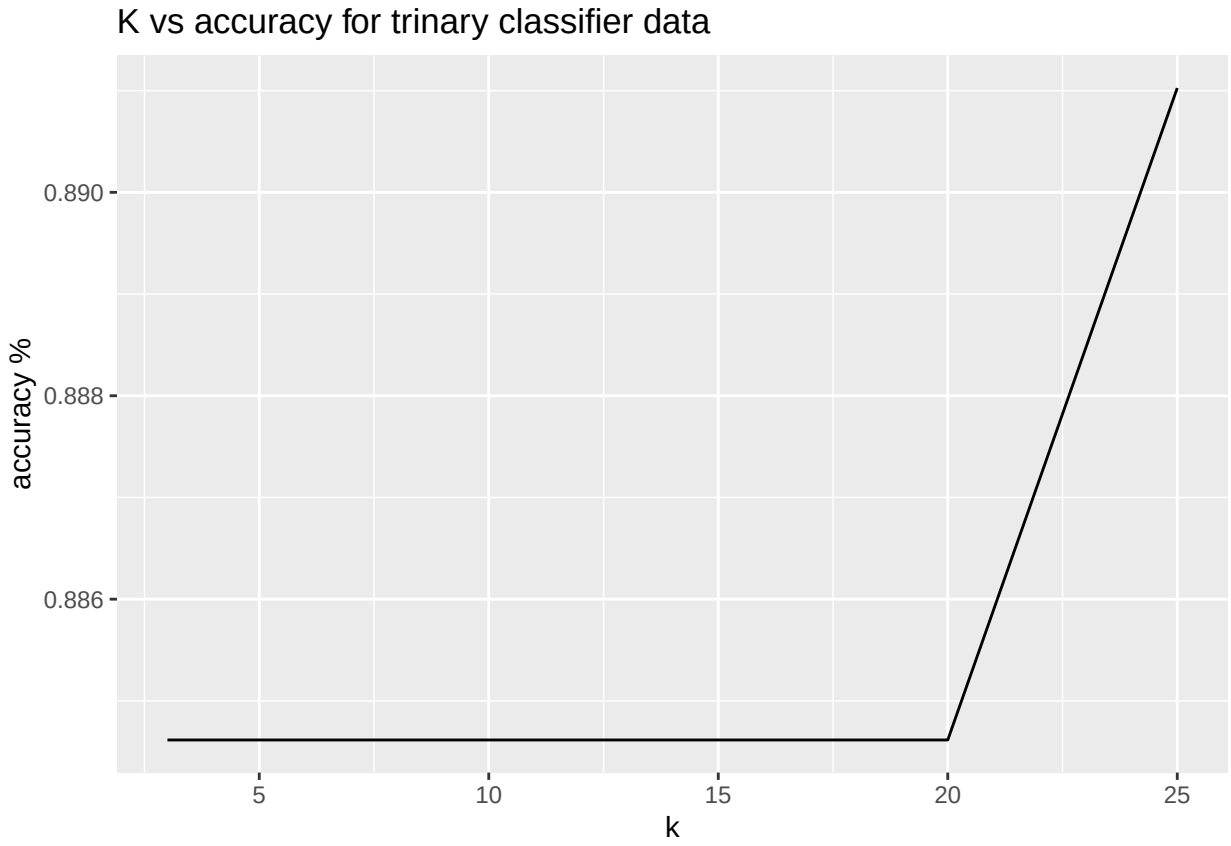
```
##    k  accuracy
## 1  3 0.8846154
## 2  5 0.8846154
## 3 10 0.8846154
## 4 20 0.8846154
## 5 25 0.8910256
```

Plot k vs accuracies

```
# plot k vs accuracies for binary classifier data set
ggplot(data = bin_data_to_plot, aes(x = k, y = accuracy)) +
    geom_line() +
    ggtitle("K vs accuracy for binary classifier data") +
    ylab("accuracy %")
```




```
# plot k vs accuracies for trinary classifier data set
ggplot(data = tri_data_to_plot, aes(x = k, y = accuracy)) +
    geom_line() +
    ggtitle("K vs accuracy for trinary classifier data") +
    ylab("accuracy %")
```



Looking back at the plots of data, linear classifier wouldn't work well as data clearly is not linearly separable and needs much more advanced technique to categorize.

Accuracy comparision between last week's logistic regression and this week's KNN for binary classifier data.

Accuracy obstained from logistic regression (using glm function) from last week was 54%.

Accuracy obtained this week using KNN is way higher and is about 90% for K = 25.

Thus with KNN we can see a huge gain with data at hand. Data plot clearly justifies the same.

2. Clustering

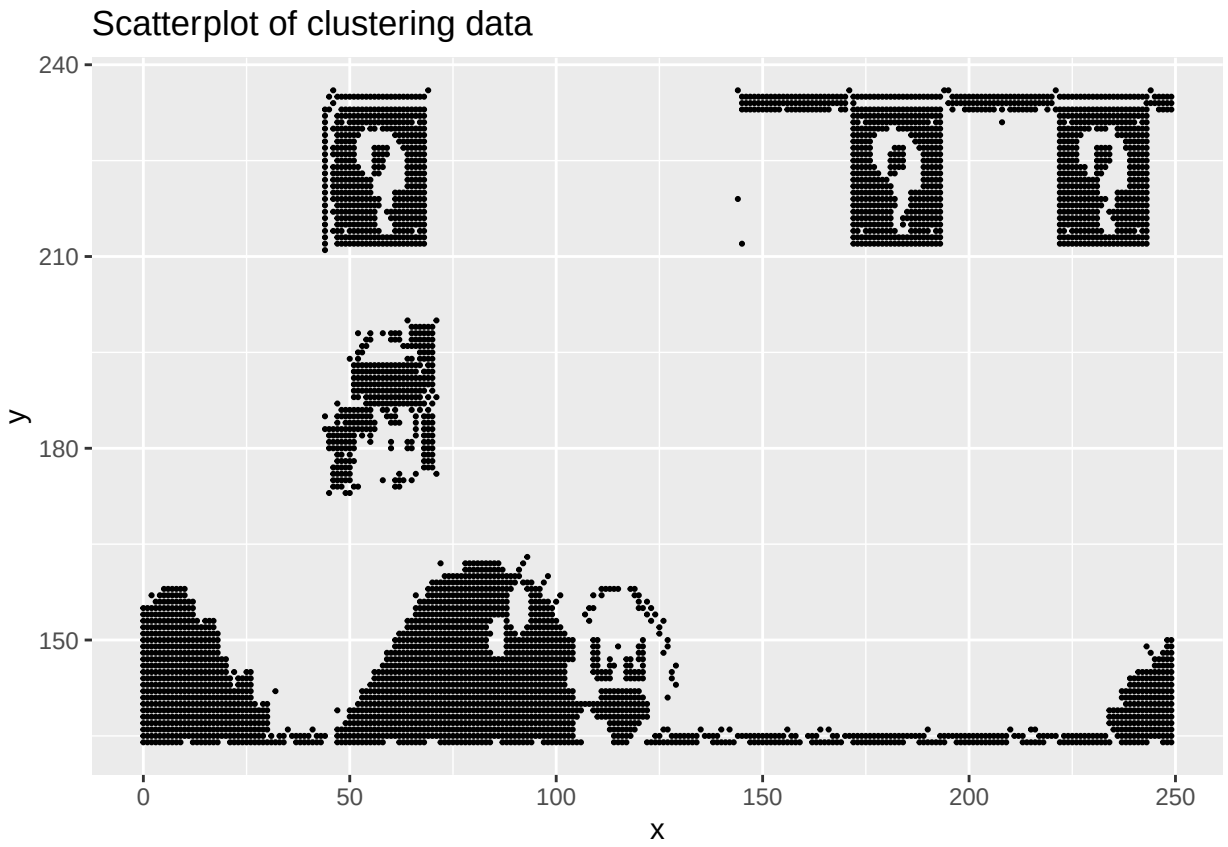
Read the data

```
# Read the data file
cluster_data <- read.csv("./clustering-data.csv")
# check the structure and some sample data
str(cluster_data)
```

```
## 'data.frame':    4022 obs. of  2 variables:
##  $ x: int  46 69 144 171 194 195 221 244 45 47 ...
##  $ y: int  236 236 236 236 236 236 236 236 235 235 ...
```

Plot the data using a scatter plot

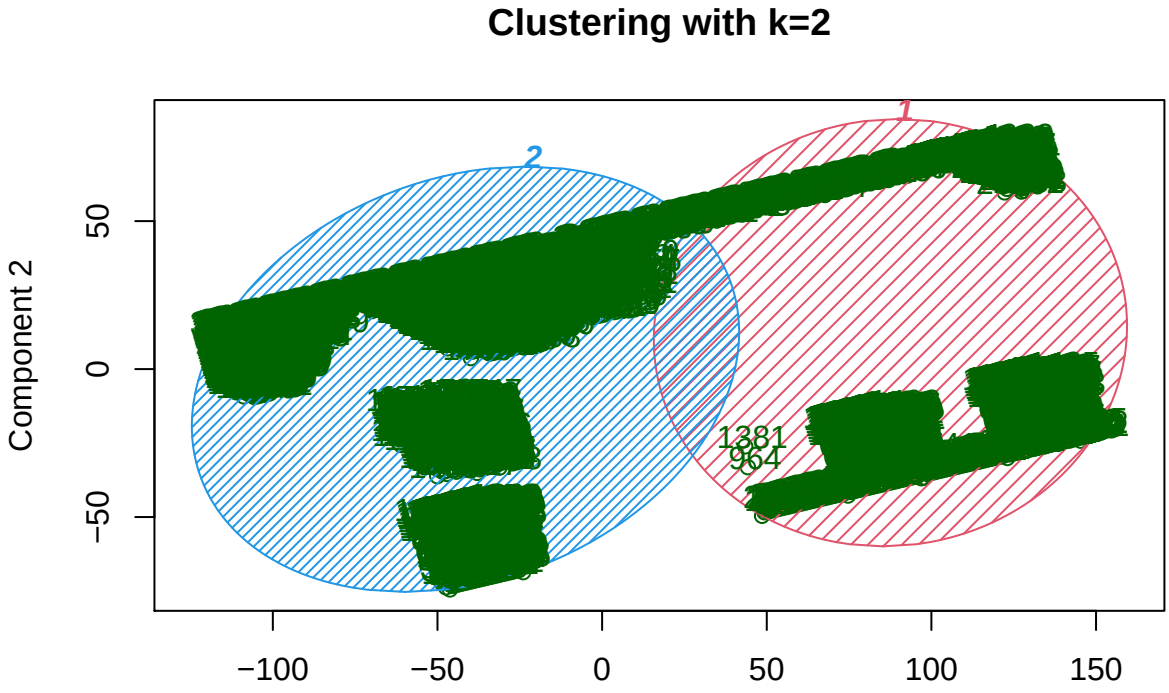
```
# scatter plot of data
library(ggplot2)
ggplot(data = cluster_data, aes(x=x, y=y)) +
    geom_point(size = 0.4) +
    ggtitle("Scatterplot of clustering data")
```



Fit the dataset using k-means algorithm from k=2 to k=12. Create a scatterplot of the resultant clusters for each value of k

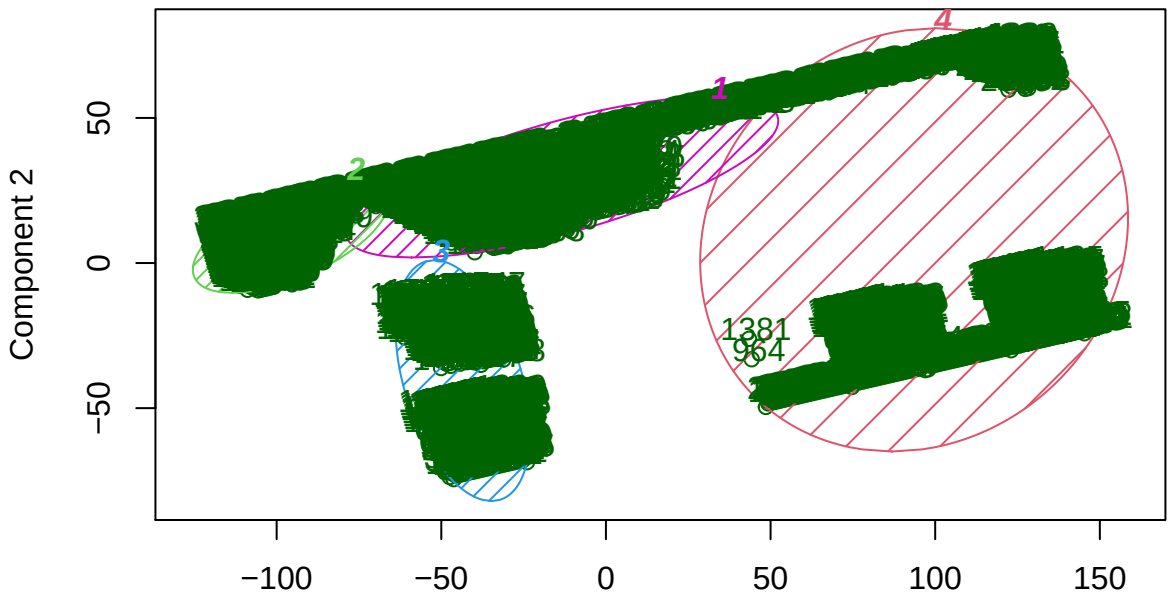
```
# Applying k-means with k = 2
set.seed(123)
kmeans_2 <- kmeans(cluster_data, 2, iter.max = 300, nstart = 10)

# Visualizing the clusters using cluster library
# library(useful)
# plot(kmeans_2, data = cluster_data)
# plot() from useful library is causing R-studio to hang and thus using another way to plot the cluster
```

```
library(cluster)
clusplot(cluster_data,
         kmeans_2$cluster, # which data point belongs to which cluster
         lines = 0, # so no distance lines appears on the plot
         shade = TRUE, # so clusters are shaded with respect to their density
         color = TRUE,
         labels = 2, # so we have all clusters and point labeled in the plot
         plotchar = FALSE, # we don't want different symbols to show in different clusters
         span = TRUE, # To plot eclipse around the clusters
         main = paste("Clustering with k=2")
         )
```



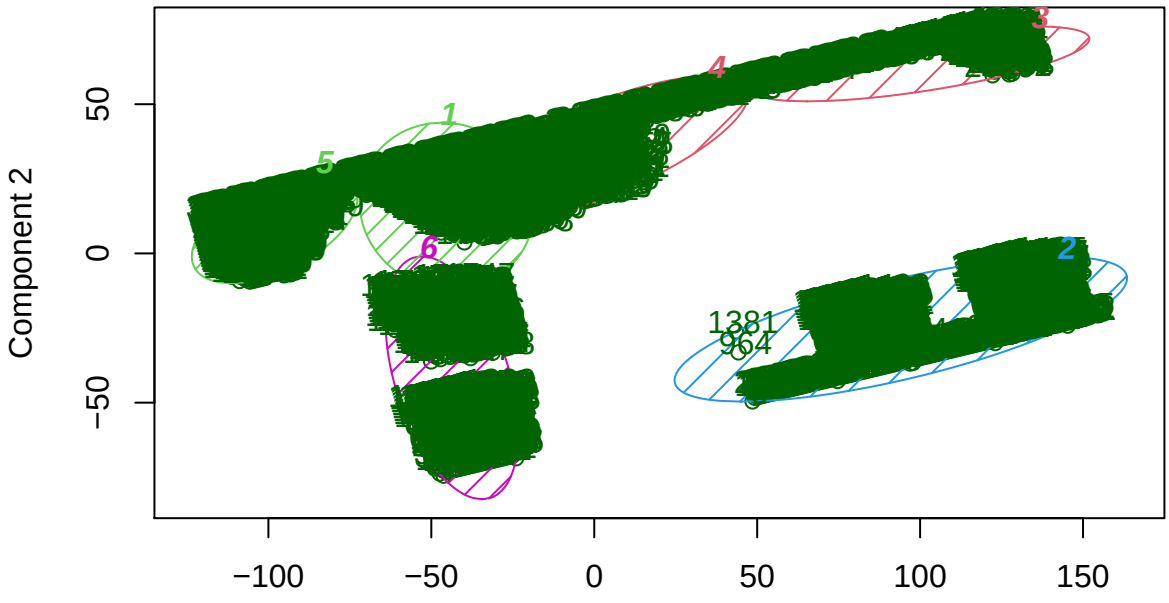

```
# Applying k-means with k = 4
set.seed(123)
kmeans_4 <- kmeans(cluster_data, 4, iter.max = 300, nstart = 10)



# Visualizing the clusters using cluster library
clusplot(cluster_data,
         kmeans_4$cluster, # which data point belongs to which cluster
         lines = 0, # so no distance lines appears on the plot
         shade = TRUE, # so clusters are shaded with respect to their density
         color = TRUE,
         labels = 2, # so we have all clusters and point labeled in the plot
         plotchar = FALSE, # we don't want different symbols to show in different clusters
         span = TRUE, # To plot eclipse around the clusters
         main = paste("Clustering with k=4")
```

```
)
```



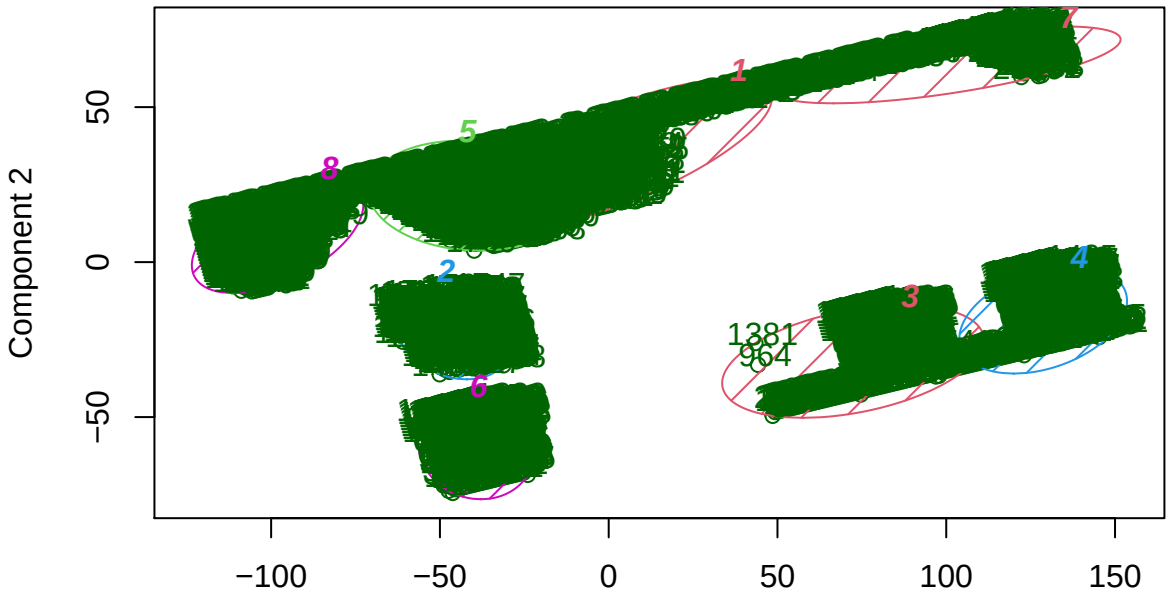
These two components explain 100 % of the point variability.

```
# Applying k-means with k = 6
set.seed(123)
kmeans_6 <- kmeans(cluster_data, 6, iter.max = 300, nstart = 10)

# Visualizing the clusters using cluster library
clusplot(cluster_data,
         kmeans_6$cluster, # which data point belongs to which cluster
         lines = 0, # so no distance lines appears on the plot
         shade = TRUE, # so clusters are shaded with respect to their density
         color = TRUE,
         labels = 2, # so we have all clusters and point labeled in the plot
         plotchar = FALSE, # we don't want different symbols to show in different clusters
         span = TRUE, # To plot eclipse around the clusters
         main = paste("Clustering with k=6")
         )
```

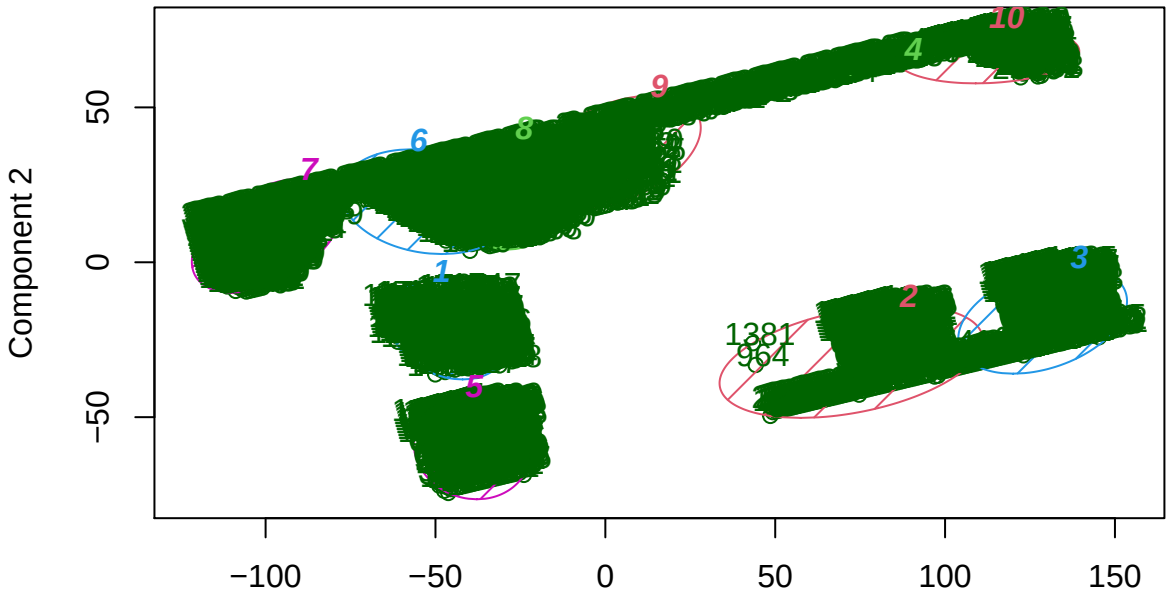


These two components explain 100 % of the point variability.

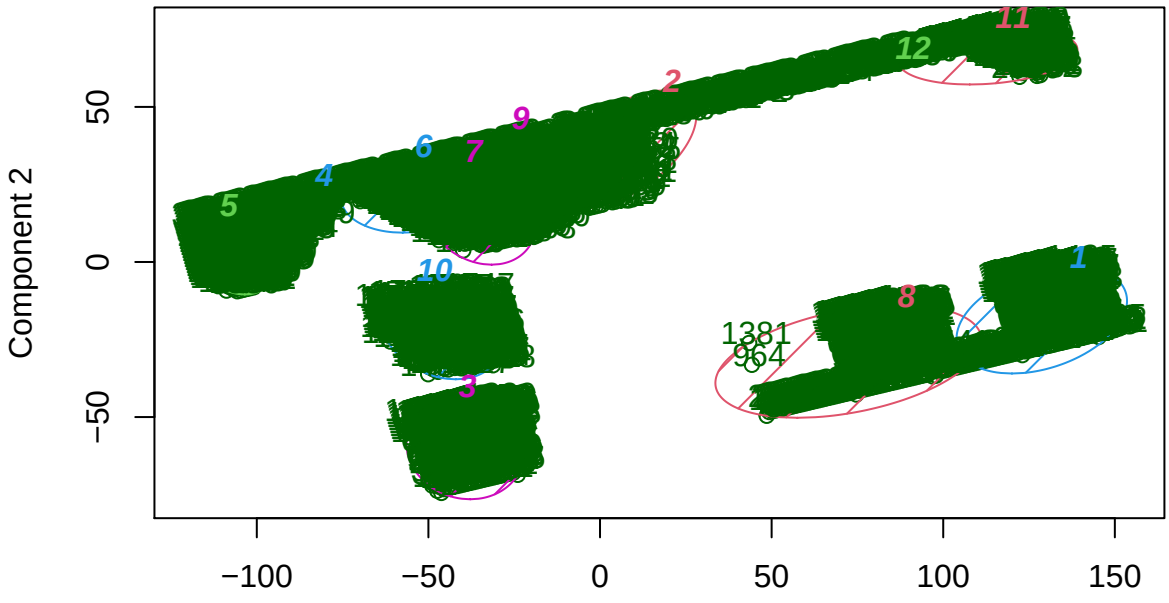
```
# Applying k-means with k = 8
set.seed(123)
kmeans_8 <- kmeans(cluster_data, 8, iter.max = 300, nstart = 10)

# Visualizing the clusters using cluster library
clusplot(cluster_data,
         kmeans_8$cluster, # which data point belongs to which cluster
         lines = 0, # so no distance lines appears on the plot
         shade = TRUE, # so clusters are shaded with respect to their density
         color = TRUE,
         labels = 2, # so we have all clusters and point labeled in the plot
         plotchar = FALSE, # we don't want different symbols to show in different clusters
         span = TRUE, # To plot eclipse around the clusters
         main = paste("Clustering with k=8")
         )
```




```
# Applying k-means with k = 10
set.seed(123)
kmeans_10 <- kmeans(cluster_data, 10, iter.max = 300, nstart = 10)

# Visualizing the clusters using cluster library
clusplot(cluster_data,
         kmeans_10$cluster, # which data point belongs to which cluster
         lines = 0, # so no distance lines appears on the plot
         shade = TRUE, # so clusters are shaded with respect to their density
         color = TRUE,
         labels = 2, # so we have all clusters and point labeled in the plot
         plotchar = FALSE, # we don't want different symbols to show in different clusters
         span = TRUE, # To plot eclipse around the clusters
         main = paste("Clustering with k=10")
         )
```




```
# Applying k-means with k = 12
set.seed(123)
kmeans_12 <- kmeans(cluster_data, 12, iter.max = 300, nstart = 10)

# Visualizing the clusters using cluster library
clusplot(cluster_data,
         kmeans_12$cluster, # which data point belongs to which cluster
         lines = 0, # so no distance lines appears on the plot
         shade = TRUE, # so clusters are shaded with respect to their density
         color = TRUE,
         labels = 2, # so we have all clusters and point labeled in the plot
         plotchar = FALSE, # we don't want different symbols to show in different clusters
         span = TRUE, # To plot eclipse around the clusters
         main = paste("Clustering with k=12")
         )
```



Calculating within cluster average within cluster sum of squares

```
# for clustering with k = 2
# getting vector of within-cluster sum of squares, one component per cluster
kmeans_2$withinss
```

```
## [1] 3057964 5385717
```

```
# getting the total distance within clusters (un squared)
tot_dis_per_cluster <- sqrt(kmeans_2$withinss)
# average distances for the whole model
avg_dis_kmeans_2 <- sum(tot_dis_per_cluster)/2
# add to a data frame for plotting
df_for_plot <- data.frame(cbind(2, avg_dis_kmeans_2))
colnames(df_for_plot) <- c("num_of_clusters", "Avg_tot_dis_within_cluster")
df_for_plot
```

```
##   num_of_clusters Avg_tot_dis_within_cluster
## 1               2                   2034.709
```

```
# for clustering with k = 4
# getting vector of within-cluster sum of squares, one component per cluster
kmeans_4$withinss
```

```
## [1]  778264.2  105897.1  334024.9 2791492.2
```

```
# getting the total distance within clusters (un squared)
tot_dis_per_cluster_4 <- sqrt(kmeans_4$withinss)
# average distances for the whole model
```

```r
avg_dis_kmeans_4 <- sum(tot_dis_per_cluster_4)/4
# add to a data frame for plotting
df_for_plot <- rbind(df_for_plot, c(4, avg_dis_kmeans_4))
df_for_plot
```

```
##   num_of_clusters Avg_tot_dis_within_cluster
## 1               2                   2034.709
## 2               4                    864.084
```

```r
# for clustering with k = 8
# getting vector of within-cluster sum of squares, one component per cluster
kmeans_8$withinss
```

```
## [1] 174149.81  35774.69 112959.43  65846.87 181245.94  47345.46 160239.56
## [8]  75619.60
```

```r
# getting the total distance within clusters (un squared)
tot_dis_per_cluster_8 <- sqrt(kmeans_8$withinss)
# average distances for the whole model
avg_dis_kmeans_8 <- sum(tot_dis_per_cluster_8)/8
# add to a data frame for plotting
df_for_plot <- rbind(df_for_plot, c(8, avg_dis_kmeans_8))
df_for_plot
```

```
##   num_of_clusters Avg_tot_dis_within_cluster
## 1               2                  2034.7091
## 2               4                   864.0840
## 3               8                   314.7206
```

```r
# for clustering with k = 10
# getting vector of within-cluster sum of squares, one component per cluster
kmeans_10$withinss
```

```
##  [1]  35774.69 112959.43  65846.87  35016.52  47345.46  60236.34  67391.59
##  [8]  73385.70  32322.91  24436.87
```

```r
# getting the total distance within clusters (un squared)
tot_dis_per_cluster_10 <- sqrt(kmeans_10$withinss)
# average distances for the whole model
avg_dis_kmeans_10 <- sum(tot_dis_per_cluster_10)/10
# add to a data frame for plotting
df_for_plot <- rbind(df_for_plot, c(10, avg_dis_kmeans_10))
df_for_plot
```

```
##   num_of_clusters Avg_tot_dis_within_cluster
## 1               2                  2034.7091
## 2               4                   864.0840
## 3               8                   314.7206
## 4              10                   229.8596
```

```r
# for clustering with k = 12
# getting vector of within-cluster sum of squares, one component per cluster
kmeans_12$withinss
```

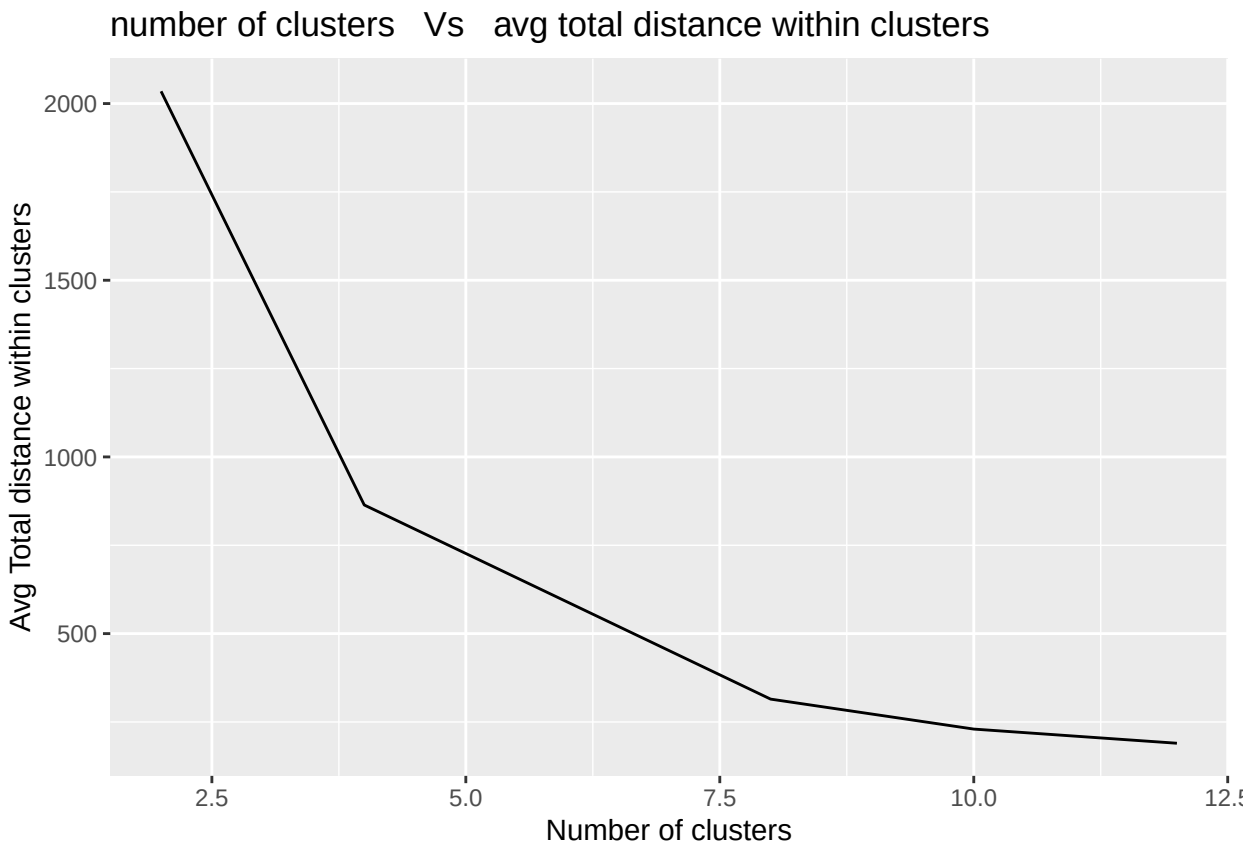
```
##  [1]  65846.87  28700.78  47345.46  11075.64  23805.34  24260.32  33990.66
##  [8] 112959.43  31168.01  35774.69  22577.30  33068.50
```

```r
# getting the total distance within clusters (un squared)
tot_dis_per_cluster_12 <- sqrt(kmeans_12$withinss)
# average distances for the whole model
avg_dis_kmeans_12 <- sum(tot_dis_per_cluster_12)/12
# add to a data frame for plotting
df_for_plot <- rbind(df_for_plot, c(12, avg_dis_kmeans_12))
df_for_plot
```

```
##   num_of_clusters Avg_tot_dis_within_cluster
## 1               2                  2034.7091
## 2               4                   864.0840
## 3               8                   314.7206
## 4              10                   229.8596
## 5              12                   189.7624
```

Plotting Elbow curve

```r
# Plotting the Elbow curve
ggplot(data = df_for_plot, aes(x = num_of_clusters, y = Avg_tot_dis_within_cluster)) +
  geom_line() +
  ggtitle("number of clusters   Vs   avg total distance within clusters") +
  xlab("Number of clusters") +
  ylab("Avg Total distance within clusters")
```




```r
# Looking at the curve it appears Elbow happening at k = 8, after that curve seems to settle down. Thus
```