ASSIGNMENT 10.2

Brian Reppeto

2023-08-07

Markdown Basics

4

```r
# Load the required libraries
library(class)

# Split binary dataset into features (x and y) and labels (label)
binary_features <- binary_data[, c("x", "y")]
binary_labels <- binary_data$label

# Split binary data into training and testing sets
set.seed(42)
binary_train_indices <-
  sample(1:nrow(binary_features), 0.8 * nrow(binary_features))
binary_features_train <- binary_features[binary_train_indices,]
binary_features_test <- binary_features[-binary_train_indices,]
binary_labels_train <- binary_labels[binary_train_indices]
binary_labels_test <- binary_labels[-binary_train_indices]

# Fit the nearest neighbors model for binary classification
binary_k <- 3  # You can adjust the number of neighbors
binary_classifier <-
  knn(
    train = binary_features_train,
    test = binary_features_test,
    cl = binary_labels_train,
    k = binary_k
  )

# Calculate accuracy for binary classification
binary_accuracy <-
  sum(binary_classifier == binary_labels_test) / length(binary_labels_test)
print(paste("Binary Classifier Accuracy:", binary_accuracy))
```

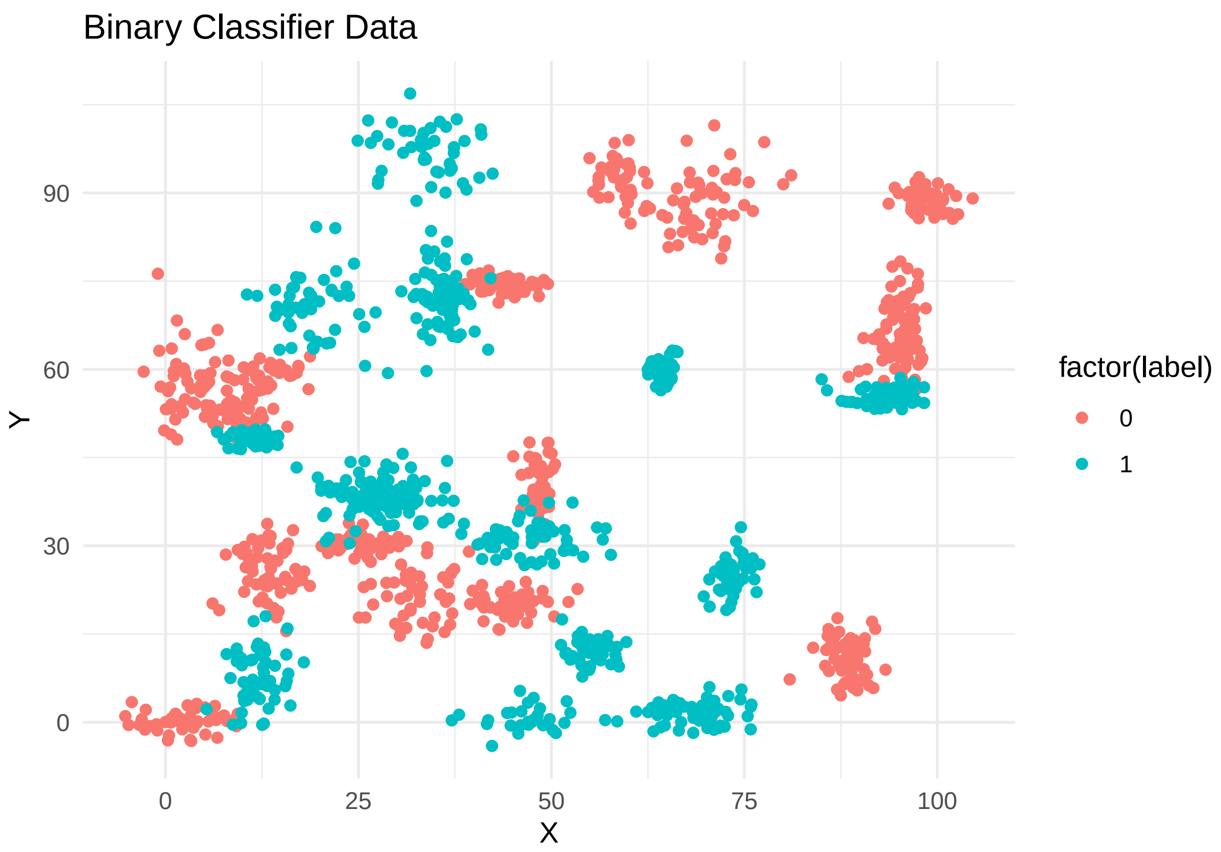
```
## [1] "Binary Classifier Accuracy: 0.973333333333333"
```

```r
# Split dataset into features (x and y) and label
trinary_features <- trinary_data[, c("x", "y")]
trinary_labels <- trinary_data$label

# Split data into training and testing sets
set.seed(42)
```

```r
trinary_train_indices <-
  sample(1:nrow(trinary_features), 0.8 * nrow(trinary_features))
trinary_features_train <- trinary_features[trinary_train_indices,]
trinary_features_test <- trinary_features[-trinary_train_indices,]
trinary_labels_train <- trinary_labels[trinary_train_indices]
trinary_labels_test <- trinary_labels[-trinary_train_indices]

# Fit the nearest neighbors model for classification
trinary_k <- 3  # You can adjust the number of neighbors
trinary_classifier <-
  knn(
    train = trinary_features_train,
    test = trinary_features_test,
    cl = trinary_labels_train,
    k = trinary_k
  )

# Calculate accuracy for classification
trinary_accuracy <-
  sum(trinary_classifier == trinary_labels_test) / length(trinary_labels_test)
print(paste("Trinary Classifier Accuracy:", trinary_accuracy))
```

```
## [1] "Trinary Classifier Accuracy: 0.89171974522293"
```

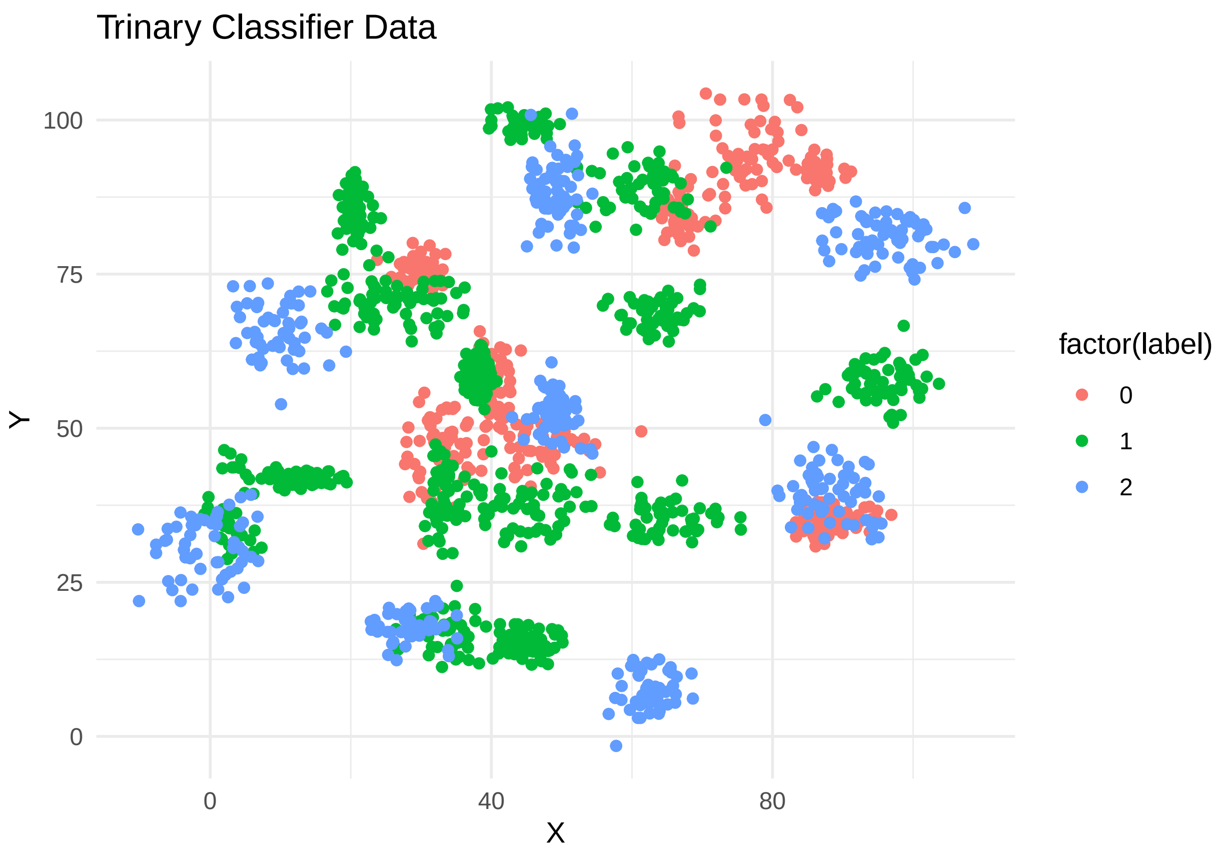
```r
# Load the required libraries
library(ggplot2)
library(stats)

# Scatter plot for binary data
ggplot(binary_data, aes(x = x, y = y, color = factor(label))) +
  geom_point() +
  labs(title = "Binary Classifier Data", x = "X", y = "Y") +
  theme_minimal()
```



```r
# Calculate Euclidean distance between two points (p1 and p2)
euclidean_distance <- function(p1, p2) {
  sqrt(sum((p1 - p2) ^ 2))
}

# Example usage of Euclidean distance function
point1 <- c(1, 2)
point2 <- c(3, 4)
distance <- euclidean_distance(point1, point2)
print(paste("Euclidean distance between point1 and point2:", distance))
```

```
## [1] "Euclidean distance between point1 and point2: 2.82842712474619"
```

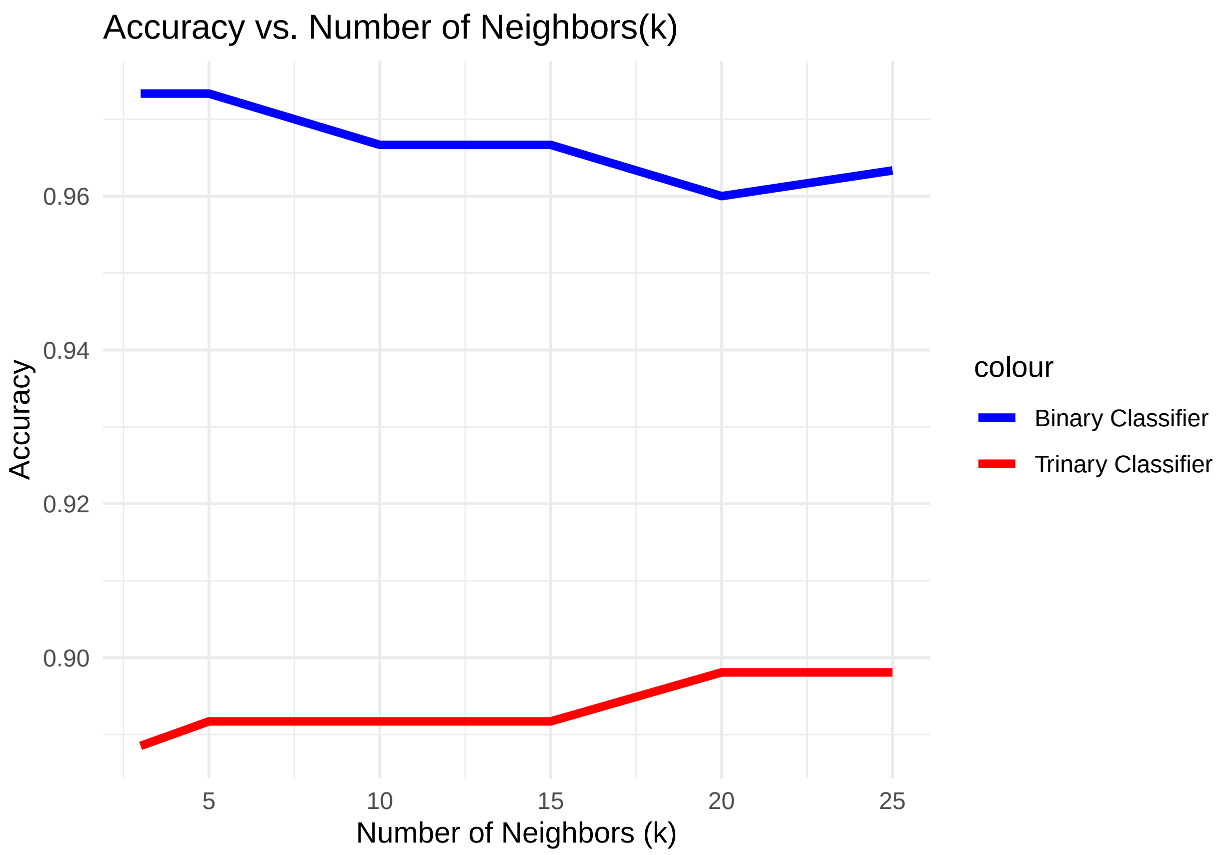
```r
# Scatter plot for trinary data
ggplot(trinary_data, aes(x = x, y = y, color = factor(label))) +
  geom_point() +
  labs(title = "Trinary Classifier Data", x = "X", y = "Y") +
  theme_minimal()
```



```
# Load the required libraries
library(class)
library(ggplot2)

calculate_accuracy <-
  function(features_train,
           labels_train,
           features_test,
           labels_test,
           k) {
    predictions <-
      knn(
        train = features_train,
        test = features_test,
        cl = labels_train,
        k = k
      )
    accuracy <- sum(predictions == labels_test) / length(labels_test)
    return(accuracy)
}

k_values <- c(3, 5, 10, 15, 20, 25)

binary_accuracies <- sapply(k_values, function(k) {
  calculate_accuracy(
    binary_features_train,
    binary_labels_train,
```

```r
    binary_features_test,
    binary_labels_test,
    k
  )
})


trinary_accuracies <- sapply(k_values, function(k) {
  calculate_accuracy(
    trinary_features_train,
    trinary_labels_train,
    trinary_features_test,
    trinary_labels_test,
    k
  )
})

# Create a data frame for plotting
accuracy_df <-
  data.frame(K = k_values,
             Binary_Accuracy = binary_accuracies,
             Trinary_Accuracy = trinary_accuracies)

# Plotting
ggplot(accuracy_df, aes(x = K)) +
  geom_line(aes(y = Binary_Accuracy, color = "Binary Classifier"), size = 1.5) +
  geom_line(aes(y = Trinary_Accuracy, color = "Trinary Classifier"), size = 1.5) +
  labs(title = "Accuracy vs. Number of Neighbors(k)",x = "Number of Neighbors (k)", y = "Accuracy") +
  scale_color_manual(values = c(
    "Binary Classifier" = "blue",
    "Trinary Classifier" = "red"
  )) +
  theme_minimal()
```

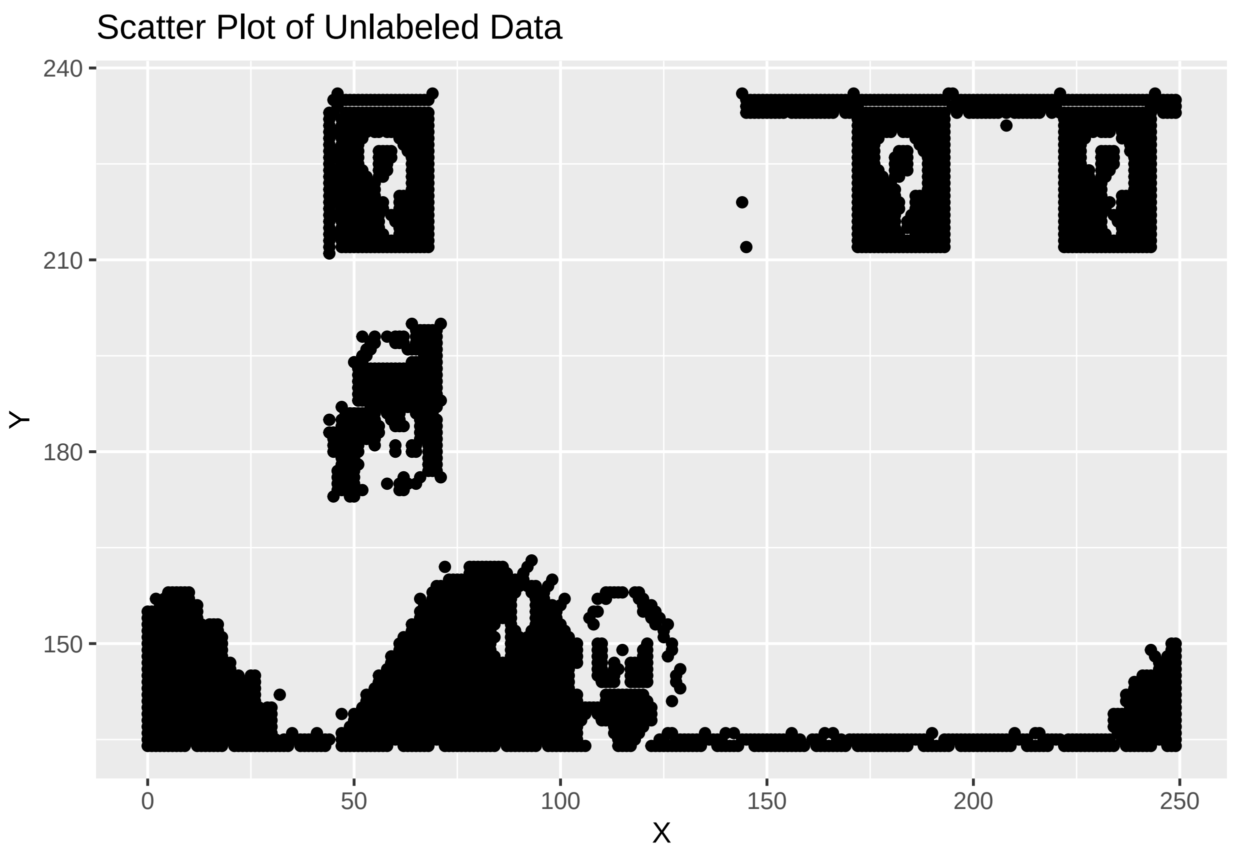
```
## Warning: Using `size` aesthetic for lines was deprecated in ggplot2 3.4.0.
## i Please use `linewidth` instead.
## This warning is displayed once every 8 hours.
## Call `lifecycle::last_lifecycle_warnings()` to see where this warning was
## generated.
```



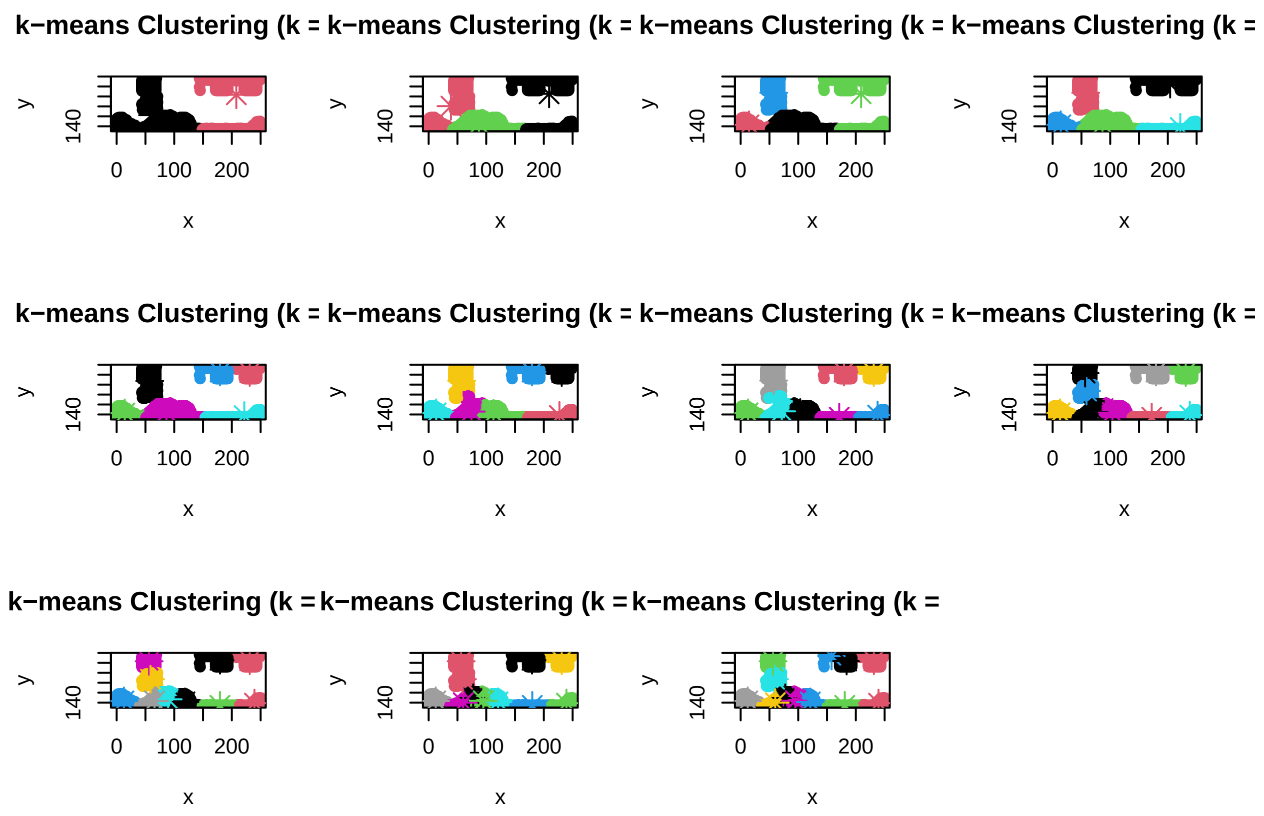
Looking at the plots, they do not seem like they would be easily split by a straight line. The plot from last week had more correlated data.

```r
library(readr)
library(ggplot2)
setwd("~/DSC520/Week 10")

cluster_df <- read.csv("clustering-data.csv")
```

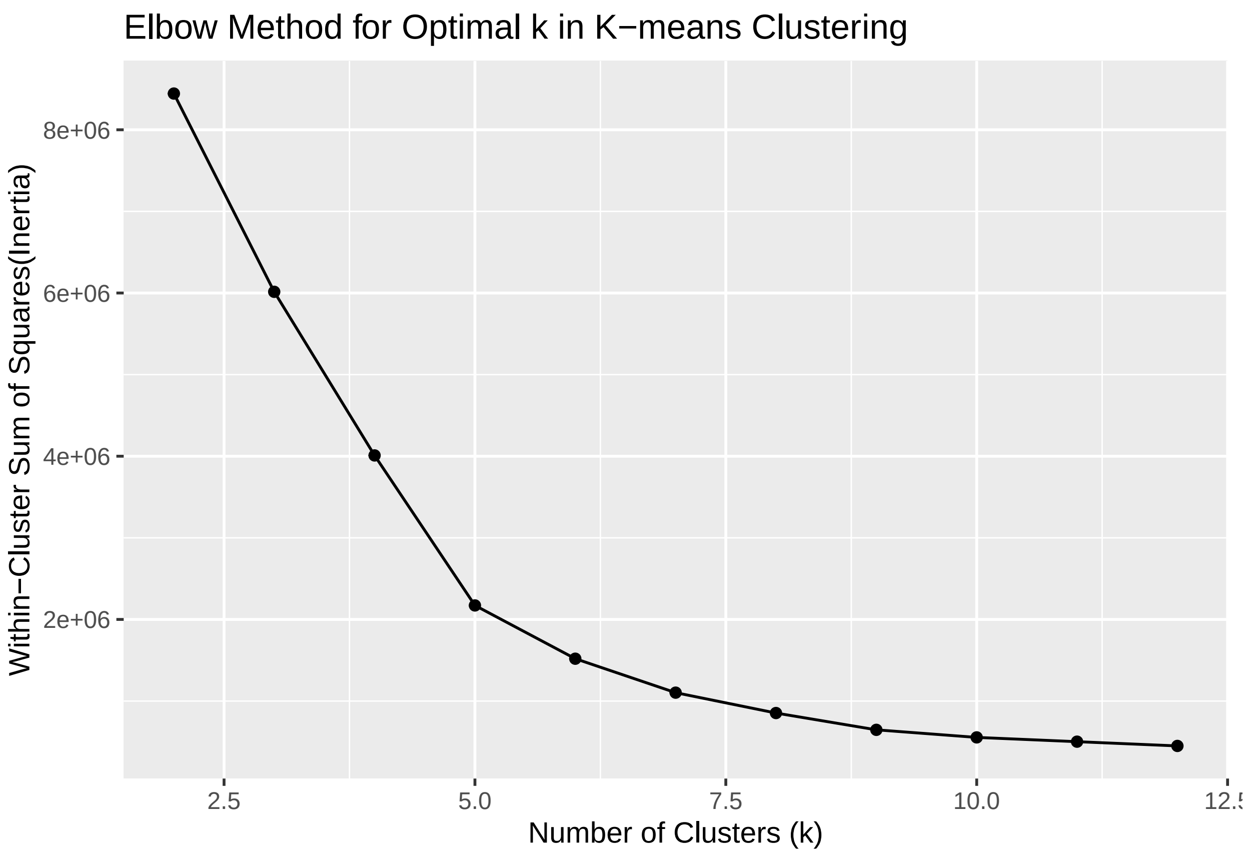
```r
ggplot(cluster_df, aes(x = x, y = y)) +
  geom_point() +
  labs(title = "Scatter Plot of Unlabeled Data", x = "X", y = "Y")
```



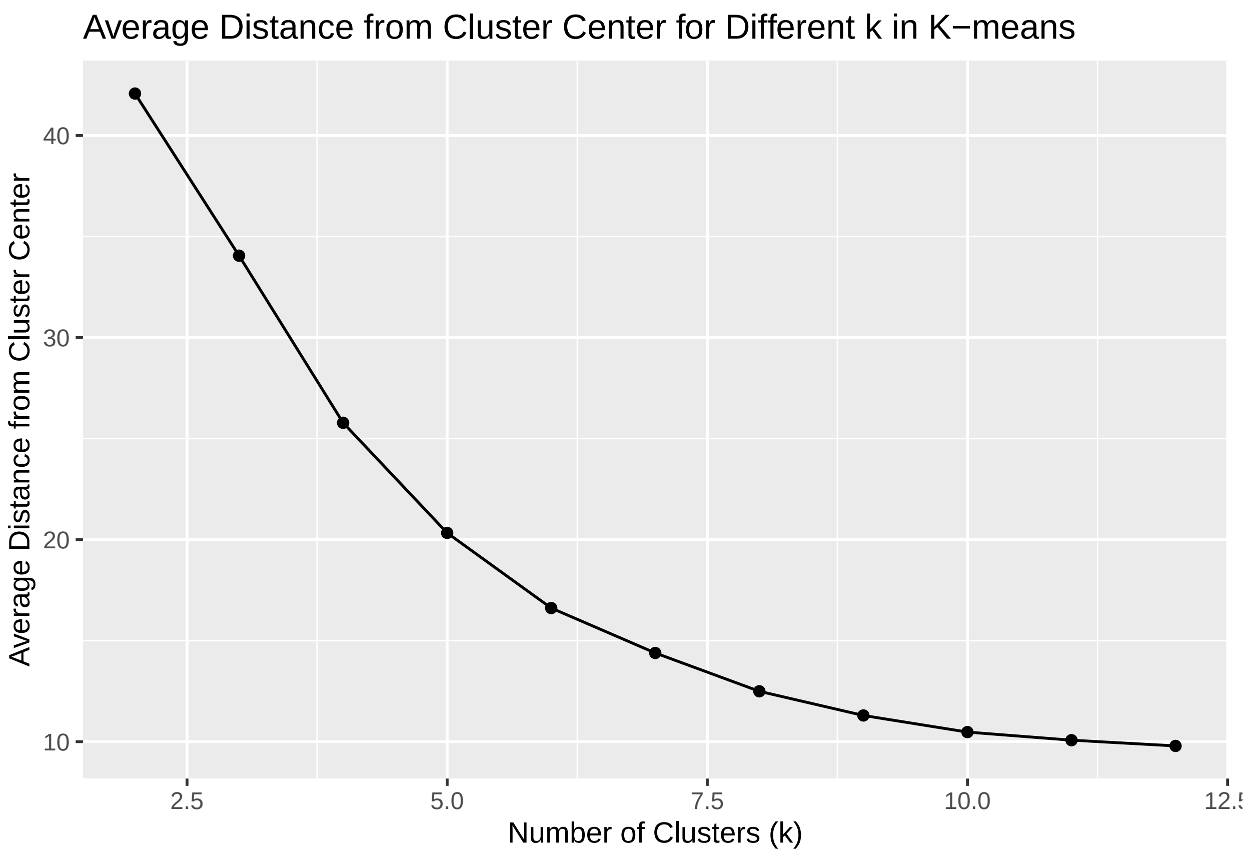
```
library(cluster)
library(ggplot2)
perform_kmeans <- function(cluster_df , k) {
  kmeans_result <- kmeans(cluster_df , centers = k, nstart = 25)
  cluster_centers <- kmeans_result$centers
  cluster_assignment <- kmeans_result$cluster
  plot(cluster_df , col = cluster_assignment, pch = 19,
       main = paste("k-means Clustering (k =", k, ")"))
  points(cluster_centers, col = 1:k, pch = 8, cex = 2)
}


par(mfrow = c(3, 4))
for (k in 2:12) {
  perform_kmeans(cluster_df , k)
}
```



```
 library(cluster)
library(ggplot2)

inertia <- numeric(0)
for (k in 2:12) {
  kmeans_result <- kmeans(cluster_df, centers = k, nstart = 25)
  inertia[k - 1] <- kmeans_result$tot.withinss
}
plot_data <- data.frame(K = 2:12, Inertia = inertia)


ggplot(plot_data, aes(x = K, y = Inertia)) +
  geom_line() +
  geom_point() +
  labs(x = "Number of Clusters (k)", y="Within-Cluster Sum of Squares(Inertia)",
       title = "Elbow Method for Optimal k in K-means Clustering")
```



```
k_values <- 2:12
average_distances <- numeric(length(k_values))

for (i in 1:length(k_values)) {
  k <- k_values[i]
  kmeans_result <- kmeans(cluster_df, centers = k, nstart = 25)
  average_distances[i] <-
    mean(apply(cluster_df - kmeans_result$centers[kmeans_result$cluster, ], 1, function(x)
      sqrt(sum(x ^ 2))))
}

# Create a data frame for plotting
plot_data <- data.frame(K = k_values, Average_Distance = average_distances)

# Plot average distance from center for different values of k
ggplot(plot_data, aes(x = K, y = Average_Distance)) +
  geom_line() +
  geom_point() +
  labs(x = "Number of Clusters (k)",y ="Average Distance from Cluster Center",
       title ="Average Distance from Cluster Center for Different k in K-means")
```



Elbow point

Looking at the graph abouve, it appears the elbow point is 10 as this the point that the graph appears to start to flatten.