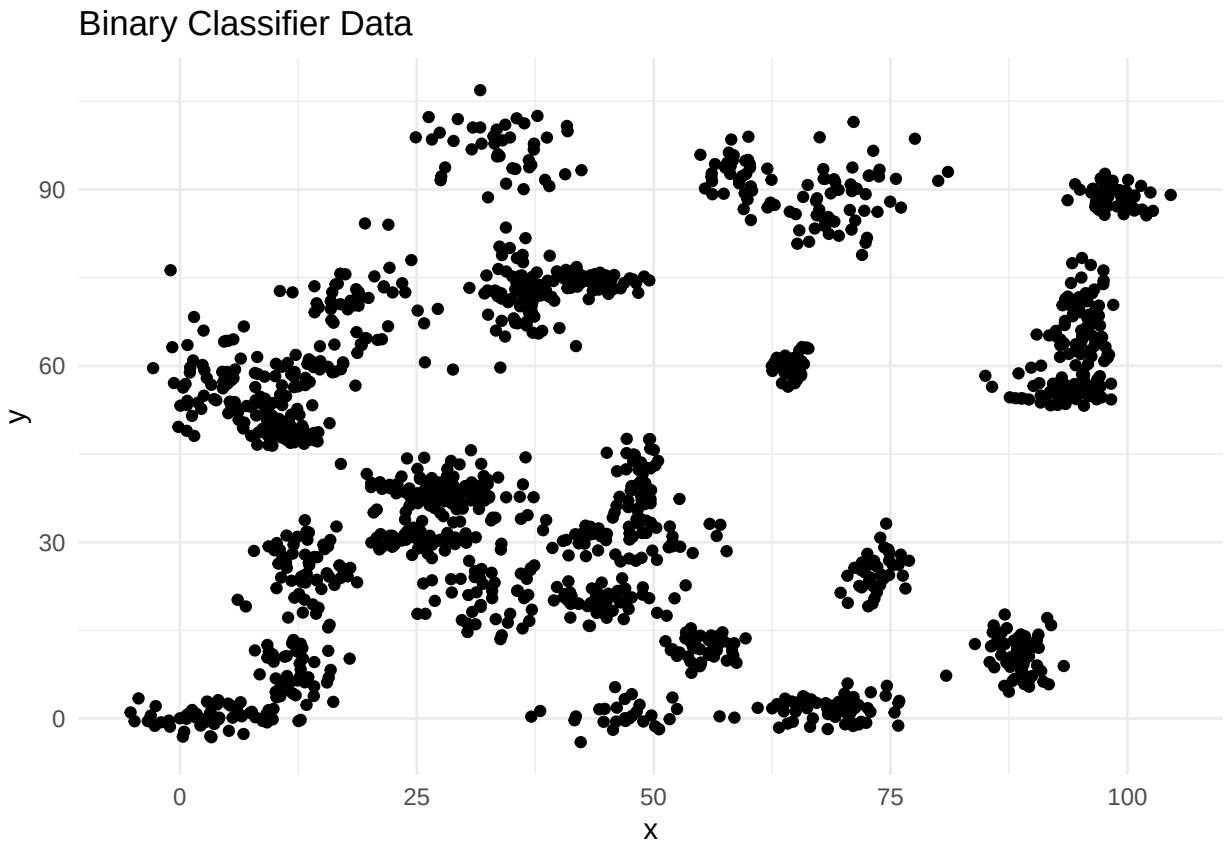
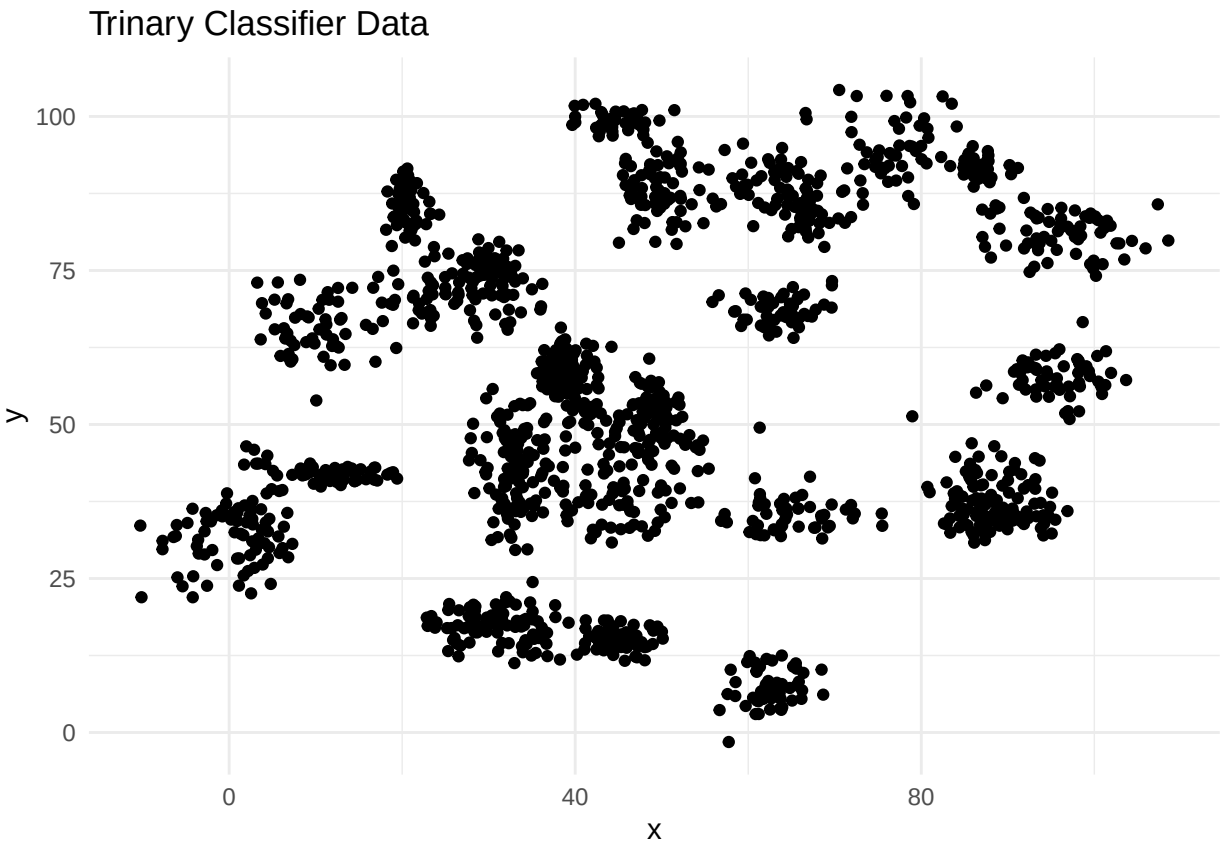
K-Nearest Neighbors Analysis

```r
# Load the datasets
binary_data <- read.csv("/Users/nickblackford/Desktop/R/binary-classifier-data (1).csv")
trinary_data <- read.csv("/Users/nickblackford/Desktop/R/trinary-classifier-data.csv")

# Plot the binary dataset
ggplot(binary_data, aes(x = x, y = y)) +
  geom_point() +
  labs(title = "Binary Classifier Data", color = "Class") +
  theme_minimal()
```

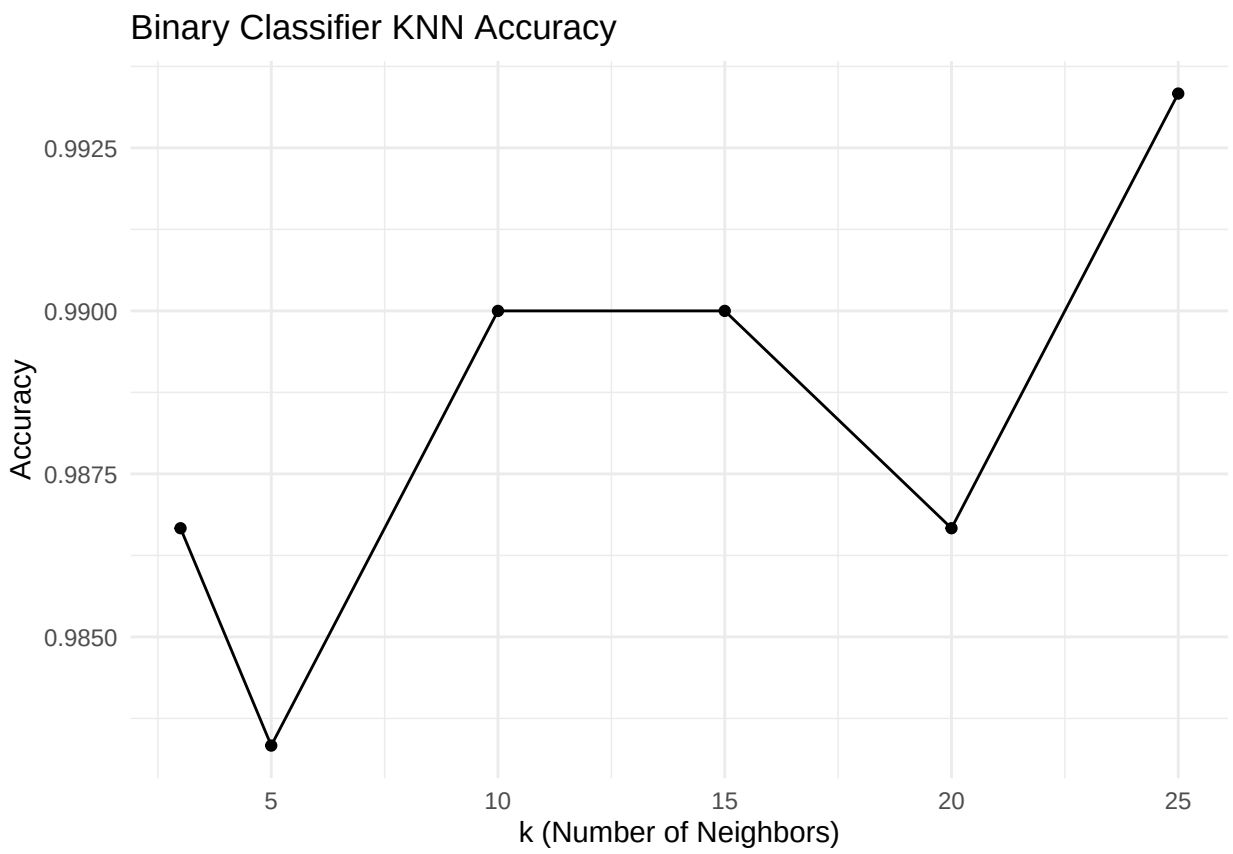



```r
# Plot the trinary dataset
ggplot(trinary_data, aes(x = x, y = y)) +
  geom_point() +
  labs(title = "Trinary Classifier Data", color = "Class") +
  theme_minimal()
```

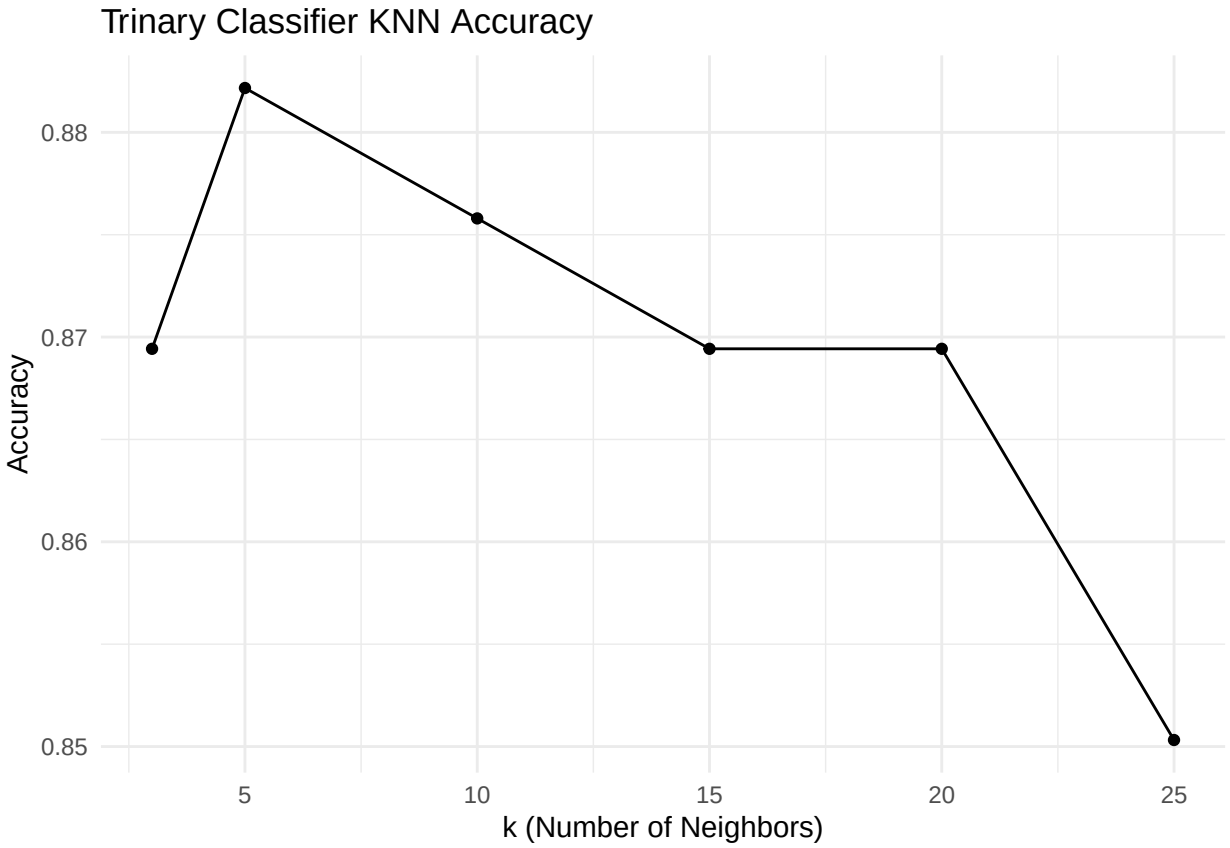



```r
# Function to compute KNN accuracy for a range of k values
compute_knn_accuracy <- function(data, k_values) {
  accuracy <- numeric(length(k_values))
  for (i in seq_along(k_values)) {
    k <- k_values[i]
    set.seed(123)  # For reproducibility
    # Create indices for a random split
    indices <- sample(1:nrow(data), size = 0.8 * nrow(data))
    train <- data[indices, ]
    test <- data[-indices, ]
    # Fit KNN model and predict
    predicted <- knn(train = train[, c("x", "y")], test = test[, c("x", "y")], cl = train[, "label"], k
    # Compute accuracy
    accuracy[i] <- sum(predicted == test[, "label"]) / length(predicted)
  }
  return(accuracy)
}

# Define k values to test
k_values <- c(3, 5, 10, 15, 20, 25)

# Compute accuracy for each k value
binary_accuracy <- compute_knn_accuracy(binary_data, k_values)
trinary_accuracy <- compute_knn_accuracy(trinary_data, k_values)

# Plot the results
plot_knn_accuracy <- function(accuracy, k_values, title) {
```

```
  data <- data.frame(k = k_values, Accuracy = accuracy)
  ggplot(data, aes(x = k, y = Accuracy)) +
    geom_line() +
    geom_point() +
    labs(title = title, x = "k (Number of Neighbors)", y = "Accuracy") +
    theme_minimal()
}

plot_knn_accuracy(binary_accuracy, k_values, "Binary Classifier KNN Accuracy")
```




```
plot_knn_accuracy(trinary_accuracy, k_values, "Trinary Classifier KNN Accuracy")
```

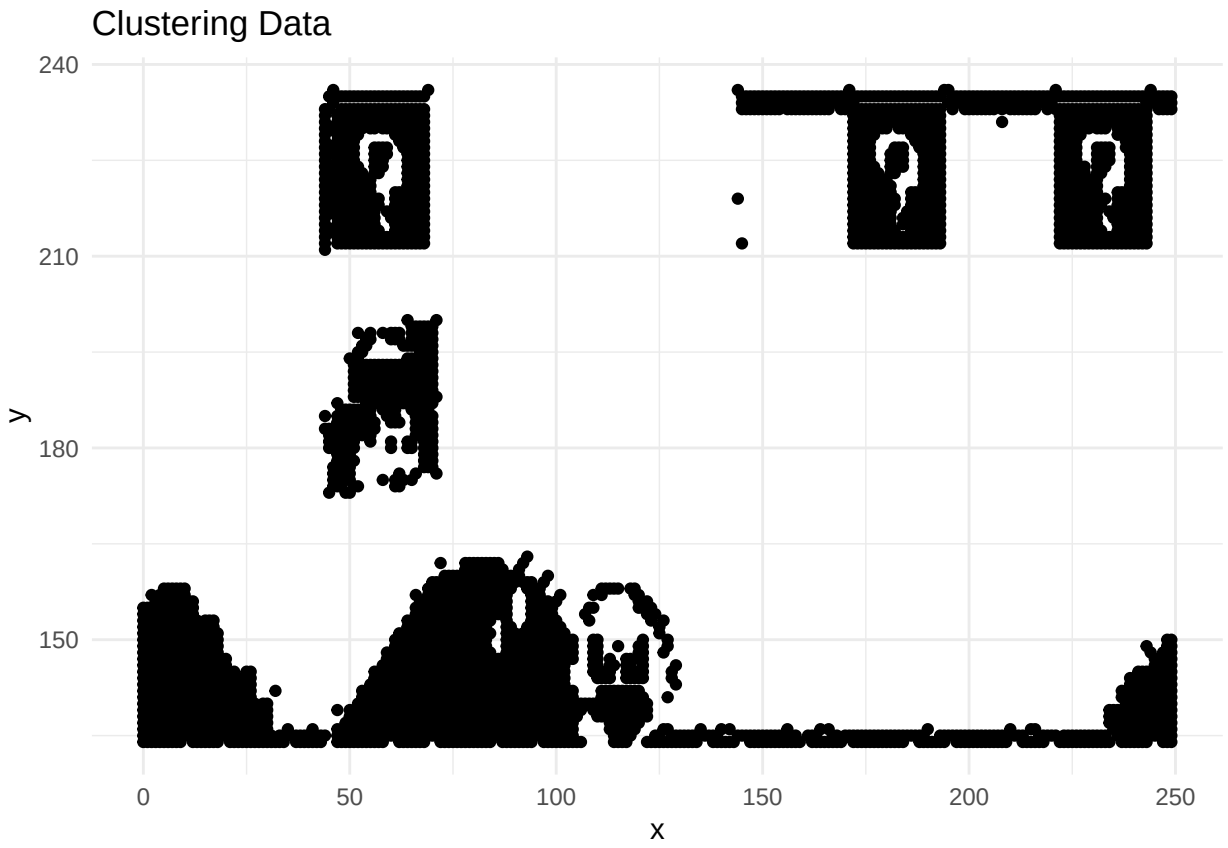


Looking back at the data, a linear classifier would not work well as the data points do not follow a linear trend.

The accuracy of my logistic regression was 84% for the binary dataset. For KNN on the binary dataset, my accuracy was upwards of 98%. The accuracies vary because KNN is a non-linear model capable of capturing more complex relationships between variables.

Part 2

```
library(ggplot2)

# Load the dataset
clustering_data <- read.csv("/Users/nickblackford/Downloads/clustering-data.csv")

# Plot the dataset
ggplot(clustering_data, aes(x = x, y = y)) +
  geom_point() +
  labs(title = "Clustering Data") +
  theme_minimal()
```




```r
# Fit k-means and plot clusters for k = 2 to 12
for (k in 2:12) {
  set.seed(123)  # For reproducibility
  kmeans_result <- kmeans(clustering_data, centers = k)
  clustering_data$cluster <- as.factor(kmeans_result$cluster)

  # Plot the clusters
  ggplot(clustering_data, aes(x = x, y = y, color = cluster)) +
    geom_point() +
    labs(title = paste("K-Means Clustering with k =", k), color = "Cluster") +
    theme_minimal()
}
```

I was unable to compute the distance of each data point to the center of the cluster it is assigned to and take the average value of all of those distances. I couldn't do this because I couldn't get a function that would do this for each data point without erroring out every time. Thus, I was also unable to plot a line with k on the x axis and average value of distances on the y axis, but I did do some research myself on elbow points and what to look for in k means clustering.