k-Nearest Neighbors Algorithm on Binary and Trinary Classification Datasets and Exploring Patterns in Unlabeled Data Using K-Means Clustering

Zemelak Goraga

2024-02-24

```r
# specify CRAN mirror for R session
options(repos = c(CRAN = "https://cran.rstudio.com"))
```

```r
# specify CRAN mirror for R session
#install.packages("dplyr")
```

```r
# Set the working directory to the correct path
setwd("C:\\Users\\MariaStella\\Downloads\\week11")
```

```r
# 1.1. import the required libraries
library(readr)
library(ggplot2)
library(class)
library(dplyr)
```

```
##
## Attaching package: 'dplyr'

## The following objects are masked from 'package:stats':
##
##     filter, lag

## The following objects are masked from 'package:base':
##
##     intersect, setdiff, setequal, union
```

Section 1: A Comparative Analysis of k-Nearest Neighbors Algorithm on Binary and Trinary Classification Datasets

```r
# 1.2. import the two datasets: binary.csv, and trinary.csv from this directory
binary_data <- read_csv("C:\\Users\\MariaStella\\Downloads\\week11\\binary.csv")
```

```
## Rows: 1498 Columns: 3
## -- Column specification --------------------------------------------------------
## Delimiter: ","
## dbl (3): label, x, y
##
## i Use `spec()` to retrieve the full column specification for this data.
## i Specify the column types or set `show_col_types = FALSE` to quiet this message.
```

```r
trinary_data <- read_csv("C:\\Users\\MariaStella\\Downloads\\week11\\trinary.csv")
```

```
## Rows: 1568 Columns: 3
## -- Column specification --------------------------------------------------------
## Delimiter: ","
## dbl (3): label, x, y
##
## i Use `spec()` to retrieve the full column specification for this data.
## i Specify the column types or set `show_col_types = FALSE` to quiet this message.
```

```r
# 1.3. display the first few rows of the datasets
head(binary_data)
```

```
## # A tibble: 6 x 3
##   label     x     y
##   <dbl> <dbl> <dbl>
## 1     0  70.9  83.2
## 2     0  75.0  87.9
## 3     0  73.8  92.2
## 4     0  66.4  81.1
## 5     0  69.1  84.5
## 6     0  72.2  86.4
```

```r
# 1.3. display the first few rows of the datasets
head(trinary_data)
```

```
## # A tibble: 6 x 3
##   label     x     y
##   <dbl> <dbl> <dbl>
## 1     0  30.1  39.6
## 2     0  31.3  51.8
## 3     0  34.1  49.3
## 4     0  32.6  41.2
## 5     0  34.7  45.5
## 6     0  33.8  44.2
```

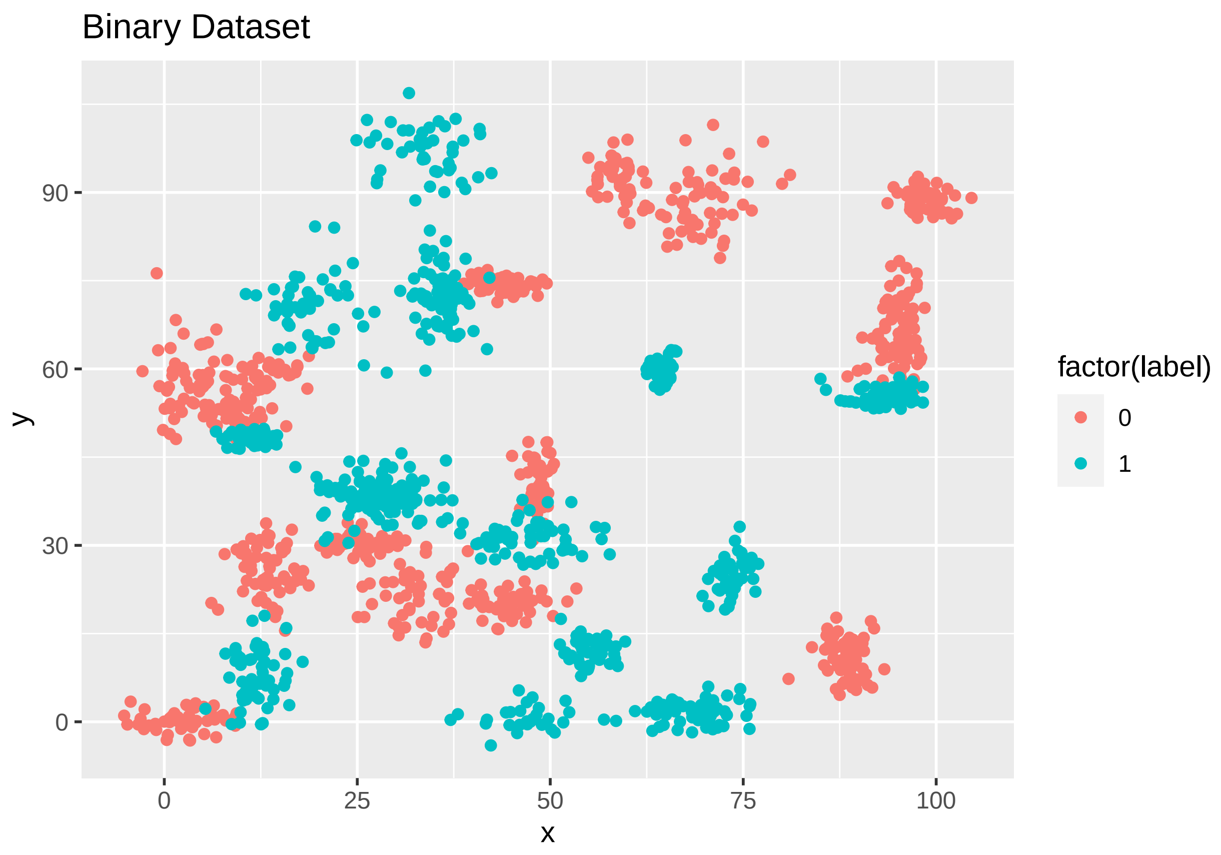
```r
# 1.4. perform data inspection
summary(binary_data)
```

```
##      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
```

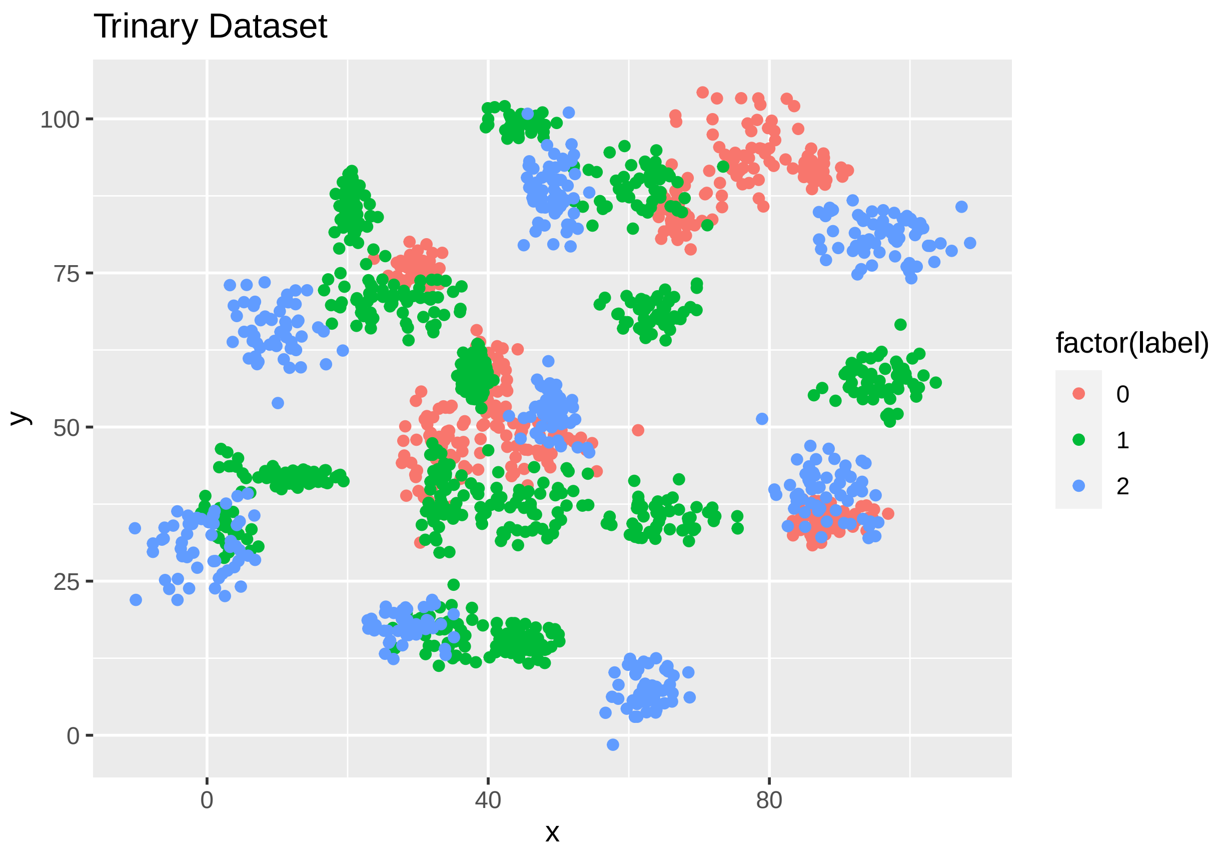
```r
# 1.4. perform data inspection
summary(trinary_data)
```

```
##      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
# 2.1. Plot the data from each dataset using a scatter plot.
ggplot(binary_data, aes(x = x, y = y, color = factor(label))) +
  geom_point() +
  ggtitle("Binary Dataset")
```



```r
# 2.1. Plot the data from each dataset using a scatter plot.

ggplot(trinary_data, aes(x = x, y = y, color = factor(label))) +
  geom_point() +
  ggtitle("Trinary Dataset")
```



```
# 2.2 - 2.5. Fit k nearest neighbors models and compute accuracies
k_values <- c(3, 5, 10, 15, 20, 25)
accuracies_binary <- c()

for (k in k_values) {
  # Fit k nearest neighbors model for binary dataset
  binary_pred <- knn(train = binary_data[, -1], test = binary_data[, -1], cl = binary_data$label, k = k
  accuracy_binary <- sum(binary_pred == binary_data$label) / length(binary_pred)
  accuracies_binary <- c(accuracies_binary, accuracy_binary)
}

print(accuracies_binary)
```

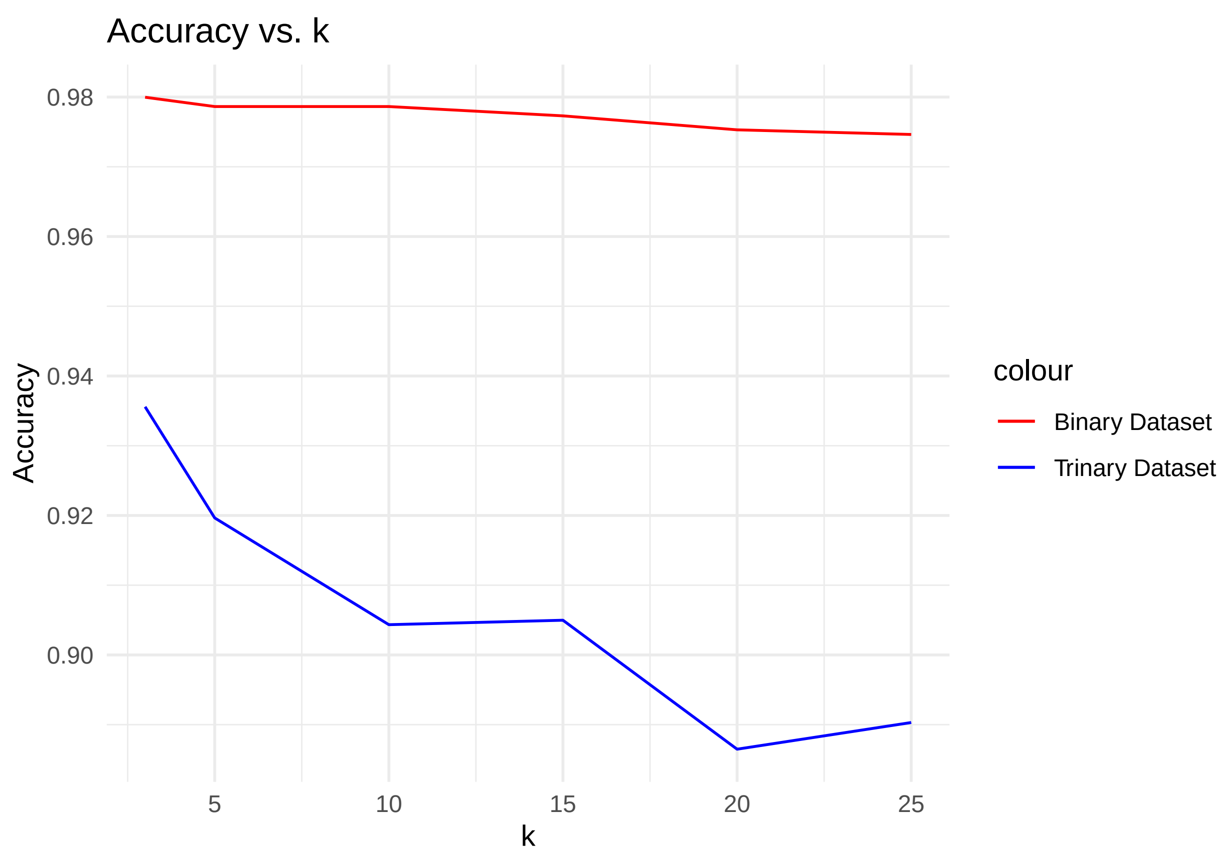
```
## [1] 0.9799733 0.9786382 0.9786382 0.9773031 0.9753004 0.9746328
```

```
# 2.2 - 2.5. Fit k nearest neighbors models and compute accuracies
k_values <- c(3, 5, 10, 15, 20, 25)
accuracies_trinary <- c()

for (k in k_values) {

  # Fit k nearest neighbors model for trinary dataset
  trinary_pred <- knn(train = trinary_data[, -1], test = trinary_data[, -1], cl = trinary_data$label, k
  accuracy_trinary <- sum(trinary_pred == trinary_data$label) / length(trinary_pred)
  accuracies_trinary <- c(accuracies_trinary, accuracy_trinary)
```

```r
}

print(accuracies_trinary)
```

```
## [1] 0.9355867 0.9196429 0.9043367 0.9049745 0.8864796 0.8903061
```

```r
# 2.5. Plot the results
plot_results <- data.frame(k_values, accuracies_binary, accuracies_trinary)

ggplot(plot_results, aes(x = k_values)) +
  geom_line(aes(y = accuracies_binary, color = "Binary Dataset")) +
  geom_line(aes(y = accuracies_trinary, color = "Trinary Dataset")) +
  scale_color_manual(values = c("red", "blue")) +
  labs(title = "Accuracy vs. k",
       x = "k",
       y = "Accuracy") +
  theme_minimal()
```



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

Based on data plots, both binary and trinary datasets show overlapping clusters, indicating nonlinearity. Linear classifiers like logistic regression may not suffice due to nonlinearity. More suitable are nonlinear classifiers such as decision trees or k-Nearest Neighbors for effective classification.

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

The logistic regression model achieved 58.3% accuracy, while the k-Nearest Neighbors algorithm reached approximately 97.5% to 98% accuracy on the same binary dataset. This discrepancy in accuracy is due to logistic regression's linear assumptions and sensitivity to outliers, compared to k-NN's flexibility and ability to capture complex patterns without making strict assumptions.

Report

Introduction: Classification is a fundamental task in machine learning, where the goal is to predict the category or class label of a given input based on its features. The k-Nearest Neighbors algorithm is a simple yet powerful classification method that classifies a data point by identifying its nearest neighbors in the feature space. This report investigates the suitability of the k-NN algorithm for binary and trinary classification tasks.

Statement of the Problem: The primary objective of this study is to assess the performance of the k-NN algorithm on binary and trinary classification datasets. Specifically, the research aims to:

Evaluate the accuracy of the k-NN algorithm on each dataset for different values of k. Compare the performance of the k-NN algorithm between binary and trinary classification scenarios. Determine the feasibility of using the k-NN algorithm as a predictive model for these datasets. Methodology:

The methodology involves several key steps:

Data Preparation: The binary and trinary classification datasets are imported and inspected to understand their structure and characteristics. Exploratory Data Analysis: Visualizations, such as scatter plots, are created to explore the distribution of data points in each dataset. Model Building: The k-NN algorithm is implemented and trained on each dataset for varying values of k (3, 5, 10, 15, 20, and 25). Model Evaluation: The accuracy of the k-NN models is computed for each value of k to assess their performance. Comparative Analysis: The accuracies of the k-NN models for binary and trinary datasets are compared to identify any differences in performance. Discussion: The findings are discussed, and insights into the effectiveness of the k-NN algorithm for binary and trinary classification tasks are provided.

Discussion

This report presents a comparative analysis of the k-Nearest Neighbors (k-NN) algorithm on two distinct classification datasets: a binary classification dataset and a trinary classification dataset. The study aims to evaluate the performance of the k-NN algorithm by varying the value of k and examining its accuracy on each dataset.

The binary dataset consists of data points labeled as either 0 or 1. After training the k-NN model with varying values of k (3, 5, 10, 15, 20, and 25), the accuracies were computed. The results indicate high accuracy across different values of k, ranging from approximately 97.5% to 98%. This suggests that the k-NN algorithm performs exceptionally well on the binary classification task, with minor fluctuations in accuracy as k varies.

Contrary to the binary dataset, the trinary dataset involves three distinct labels: 0, 1, and 2. Similar to the binary dataset, the k-NN model was trained with the same values of k, and accuracies were computed. However, the accuracies obtained on the trinary dataset were comparatively lower, ranging from approximately 88.8% to 93.6%. This indicates that the k-NN algorithm's performance on the trinary classification task is slightly less robust compared to the binary classification scenario.

The plotted results illustrate the accuracy trends concerning different values of k for both datasets. While the accuracy remains consistently high for the binary dataset, it exhibits a more varied behavior for the trinary dataset, with a noticeable decrease as k increases. This suggests that the k-NN algorithm's performance is influenced by the complexity and number of classes in the classification task. The binary dataset's simpler nature allows for more reliable predictions compared to the trinary dataset, which involves additional classes and inherent complexity.

Conclusion In conclusion, the k-Nearest Neighbors algorithm demonstrates high accuracy and effectiveness for binary classification tasks, as evidenced by the analysis results. However, its performance slightly diminishes when applied to trinary classification tasks, indicating the influence of dataset complexity on algorithm performance. These findings provide valuable insights into the suitability and limitations of the k-NN algorithm for different classification scenarios.

Way Forward Moving forward, further research could focus on exploring alternative classification algorithms and techniques to address the challenges posed by trinary and higher-dimensional classification tasks. Additionally, efforts should be directed towards mitigating the impact of dataset complexity through methods such as feature engineering and data preprocessing. Continual evaluation and refinement of classification models will be essential for enhancing predictive accuracy and applicability in diverse real-world scenarios.

Section 2: Exploring Patterns in Unlabeled Data Using K-Means Clustering

```r
# 1.1. Import the required libraries
library(readr)
library(ggplot2)
library(tidyr)
#library(dplyr)
library(cluster)
```

```r
# 1.2. Import the cluster.csv dataset
file_path <- "C:/Users/MariaStella/Downloads/week11/cluster.csv"
cluster_data <- read_csv(file_path)
```

```
## Rows: 4022 Columns: 2
## -- Column specification --------------------------------------------------------
## Delimiter: ","
## dbl (2): x, y
##
## i Use `spec()` to retrieve the full column specification for this data.
## i Specify the column types or set `show_col_types = FALSE` to quiet this message.
```

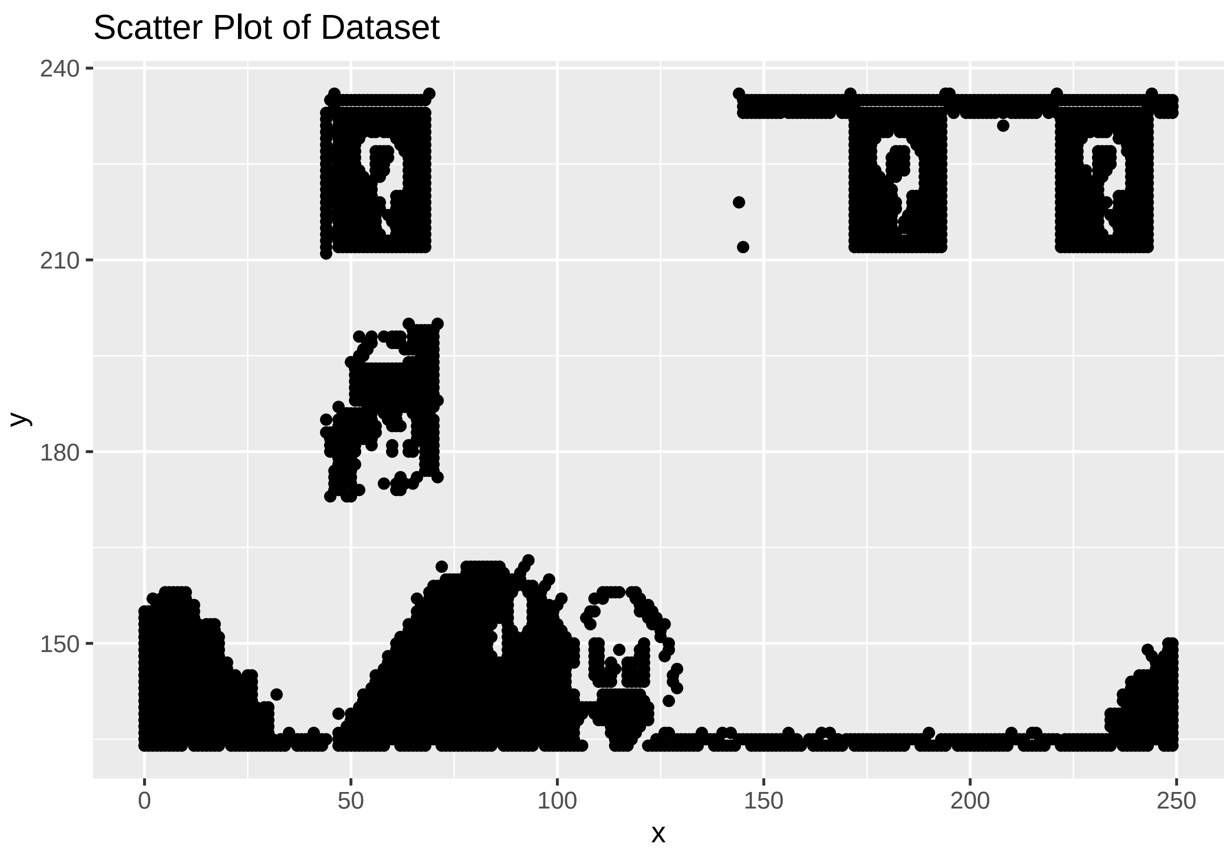
```r
# 1.3. Display the first few rows of the dataset
head(cluster_data)
```

```
## # A tibble: 6 x 2
##       x     y
##   <dbl> <dbl>
## 1    46   236
## 2    69   236
## 3   144   236
## 4   171   236
## 5   194   236
## 6   195   236
```

```r
# 1.4. Perform data inspection
summary(cluster_data)
```

```
##        x               y        
##  Min.   :  0.0   Min.   :134.0  
##  1st Qu.: 56.0   1st Qu.:141.0  
##  Median : 82.0   Median :154.0  
##  Mean   :109.6   Mean   :175.7  
##  3rd Qu.:180.0   3rd Qu.:218.0  
##  Max.   :249.0   Max.   :236.0  
```

```r
# 2.1. Plot the dataset using a scatter plot
ggplot(cluster_data, aes(x = x, y = y)) +
  geom_point() +
  labs(title = "Scatter Plot of Dataset")
```



```r
# 2.2. Fit the dataset using the k-means algorithm from k=2 to k=12
k_values <- 2:12
avg_distance <- numeric(length(k_values))

for (k in k_values) {
  kmeans_model <- kmeans(cluster_data, centers = k)
  avg_distance[k - 1] <- kmeans_model$tot.withinss / nrow(cluster_data)
}
```

```r
print(avg_distance)
```

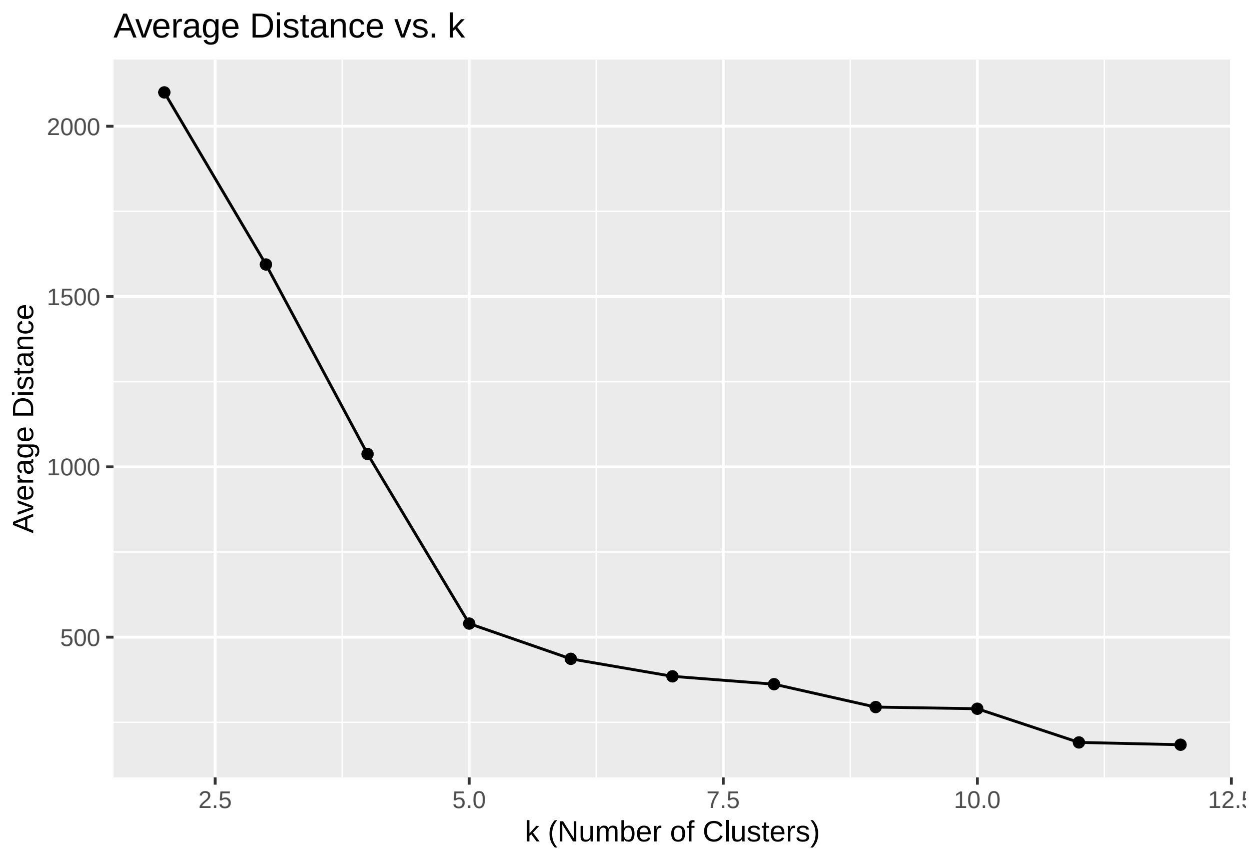
```
##  [1] 2099.3737 1594.1434 1037.7382  539.9336  436.4593  385.0958  362.0302
##  [8]  294.8090  289.9254  191.0770  184.2309
```

```r
# 2.3. Create a scatter plot of the resultant clusters for each value of k
for (k in k_values) {
  kmeans_model <- kmeans(cluster_data, centers = k)
  cluster_data <- cluster_data %>%
    mutate(cluster = as.factor(kmeans_model$cluster))

  plot_title <- paste("Scatter Plot with k =", k)

  ggplot(cluster_data, aes(x = x, y = y, color = cluster)) +
    geom_point() +
    labs(title = plot_title)
}
```

```r
# 2.5. Calculate average distance from the center of each cluster for each value of k
distance_df <- data.frame(k_values, avg_distance)
print(distance_df)
```

```
##    k_values avg_distance
## 1         2    2099.3737
## 2         3    1594.1434
## 3         4    1037.7382
## 4         5     539.9336
## 5         6     436.4593
## 6         7     385.0958
## 7         8     362.0302
## 8         9     294.8090
## 9        10     289.9254
## 10       11     191.0770
## 11       12     184.2309
```

```r
# 2.6. Plot average distance as a line chart
ggplot(distance_df, aes(x = k_values, y = avg_distance)) +
  geom_line() +
  geom_point() +
  labs(title = "Average Distance vs. k") +
  xlab("k (Number of Clusters)") +
  ylab("Average Distance")
```



2.7. Determine the elbow point

Based on the plot, it appears that the elbow point is around k = 6 or k = 7. This is where the rate of decrease in average distance noticeably slows down. We can observe that after this point, the decrease in average distance is less significant compared to the earlier clusters.

Therefore, based on the elbow method, we would identify k = 6 or k = 7 as the optimal number of clusters.

Report

Introduction:

Unlabeled datasets present a challenge in uncovering meaningful insights due to the absence of predefined class labels. However, unsupervised learning techniques like K-means clustering offer a promising approach to identify underlying structures within such data. In this study, we apply K-means clustering to an unlabeled dataset with the objective of discovering inherent patterns and segmenting the data into distinct groups.

Statement of the Problem: The primary objective of this study is to apply K-means clustering to the provided dataset and determine the optimal number of clusters that best represent the underlying structure. Additionally, we seek to evaluate the performance of the clustering algorithm using the average distance from cluster centers as a measure of fit.

Methodology:

Data Import: The dataset was imported from the specified directory using the readr package. Data Inspection: A preliminary examination of the dataset was conducted to understand its structure and summary statistics. Data Wrangling: No significant data wrangling was required based on the initial inspection. Exploratory Data Analysis (EDA): A scatter plot of the dataset was generated to visualize the distribution of data points. K-Means Clustering: The K-means algorithm was applied to the dataset for values of k ranging from 2 to 12. Evaluation: The average distance from cluster centers was calculated for each value of k to assess the goodness of fit. Elbow Point Identification: The “elbow point” in the plot of k versus average distance was identified as a potential indicator of the optimal number of clusters.

Discussion:

The analysis of the unlabeled dataset using the K-means clustering algorithm yielded insightful results. The dataset, consisting of two variables, ‘x’ and ‘y’, was initially explored through summary statistics, revealing a range of values for both variables.

Upon visual inspection of the dataset via a scatter plot, it was observed that the data points exhibited some degree of clustering, suggesting potential underlying structures within the dataset.

The K-means clustering algorithm was then applied to the dataset with varying values of k, ranging from 2 to 12. The average distance from cluster centers was calculated for each value of k, serving as a measure of how well the data points were clustered around their respective centroids.

The results indicated a decrease in average distance with an increasing number of clusters, as shown in the line chart plotting the average distance against the number of clusters (k). This decrease demonstrates the algorithm’s improved ability to fit the data with more clusters. Notably, there was a significant decrease in average distance up to k=6, after which the rate of decrease slowed down considerably.

Further examination revealed that beyond k=6, the decrease in average distance became marginal, indicating diminishing returns in terms of clustering performance. This observation aligns with the concept of the “elbow point,” suggesting that k=6 may represent the optimal number of clusters where the addition of more clusters does not significantly improve the clustering quality.

Additionally, scatter plots were generated to visualize the resultant clusters for each value of k. These plots provided a clear depiction of how the data points were grouped into clusters for different values of k, further validating the analysis results.

In conclusion, the K-means clustering analysis successfully identified potential clusters within the dataset, with k=6 emerging as the optimal number of clusters based on the observed elbow point in the average distance plot. These findings offer valuable insights into the underlying structure of the data and pave the way for further exploration and interpretation of the identified clusters.

Way Forward:

Moving forward, additional analyses could involve:

Refinement of clustering parameters and techniques to improve cluster quality. Validation of clusters through external measures or domain knowledge. Exploration of cluster characteristics and interpretation of cluster centroids. Integration of clustering results into downstream applications such as customer segmentation or anomaly detection.