Grouping Loan Clients by Repayment Behaviour

To: Quant Team at Crestline Bank: Asanda Ngcobo, Tseke Masemola

From: Head of Credit Risk

Date: 1 December 2025

Contents

1. Scenario & Problem Statement

Crestline Bank's current credit rating system for business clients is too detailed and complicated. It uses too many rating levels (like AAA, AA+, AA, etc.), which makes it hard to effectively issue loans, assign appropriate interest rates, and group clients by creditworthiness. This over-complex system causes two main problems:

- **Confusing for Staff & Costing Us Money:** With so many small rating differences, it is difficult for bank staff to decide what interest rate to charge or what loan amount to give a client. We are not matching our loan offers and interest rates to the true risk level of each client. This means we might offer low rates to risky clients (and lose money) or high rates to safe clients (and lose their business).
- **Wastes Time and Resources:** The bank spends too much time and resources maintaining and explaining tiny differences between ratings. This work does not help us better understand who will repay their loan and who will not. Our credit analysts debate whether a client is an `A-` or a `B+`, and our Client Managers spend hours justifying these subtle changes to clients, all for a distinction that doesn't change how we actually manage the relationship or price the loan.

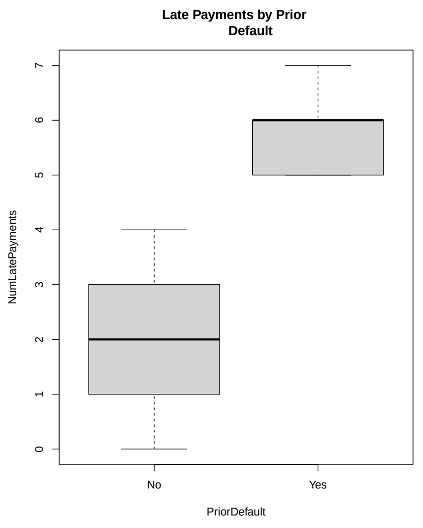
Crestline Bank needs a simpler, practical way to sort clients by their actual repayment behavior.

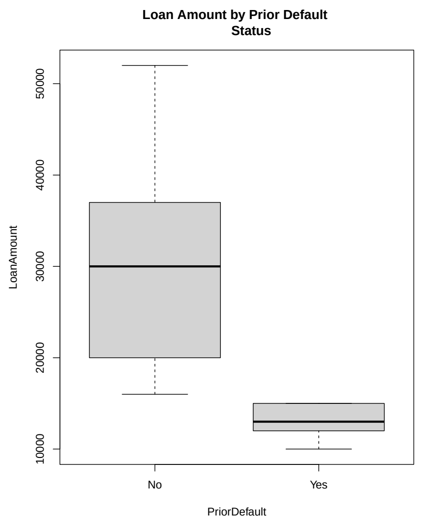
2. Objective

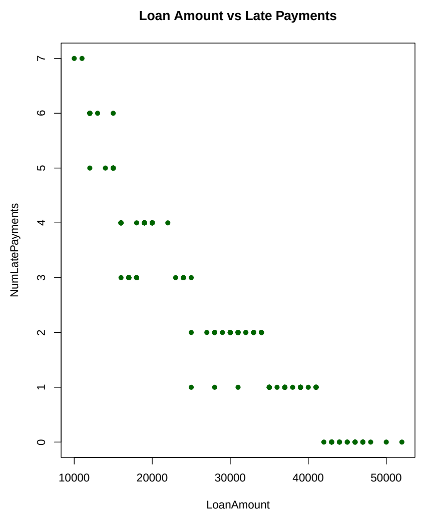
1. To increase revenue and decrease losses for Crestline Bank by reshaping how we assess client creditworthiness.
2. Replace the current complex rating framework with a simpler, practical way to group clients by creditworthiness.

3. Exploratory Data Analysis (EDA)

We collected data on 200 Crestline Bank clients to understand their repayment behavior. We focused on three key variables: Prior Default status, Number of Late Payments, and Loan Amount. Below are the exploratory plots and our observations:







- **Late Payments vs. Prior Default:** Clients with a prior default have significantly more late payments (mean ~6) than those without (mean ~2). The number of late payments acts as a trust barometer: clients with 5 or more late payments are very likely to have defaulted before. This shows a clear pattern: past defaulters are consistently late payers.
- **Loan Amount vs. Prior Default:** Prior defaulters consistently request smaller loan amounts. Clients without a prior default request a wider range of, and generally much larger, loan amounts. This makes practical sense: if a client has a high risk of default, the bank would approve a smaller loan to limit potential losses, and the client may also lack the confidence to ask for more.
- **Late Payments vs. Loan Amount:** Clients with fewer late payments tend to request larger loan amounts. This suggests that good repayment habits build trust, which allows clients to confidently request, and the bank to consider approving, larger loans.

Clearly, the plots tell a consistent story. Before any further analysis, 2 key risk drivers are:

1. having 5+ late payments is a strong warning sign of potential default.
2. a prior default history makes a client significantly riskier now.

4. Approach

We apply K-means clustering to uncover natural groupings among the bank's clients and to understand how the three key factors—prior default history, number of late payments, and loan amount—interact with overall credit risk. In our setting, K-means partitions the feature space into K distinct clusters, with each cluster representing a different combination or "profile" of these characteristics.

Before performing the clustering, we must determine an appropriate value for K. Our exploratory data analysis suggested the presence of two broad groups, driven mainly by whether a client has previously defaulted. To validate and refine this hypothesis, we use two complementary methods:

Elbow Plot (explanation)

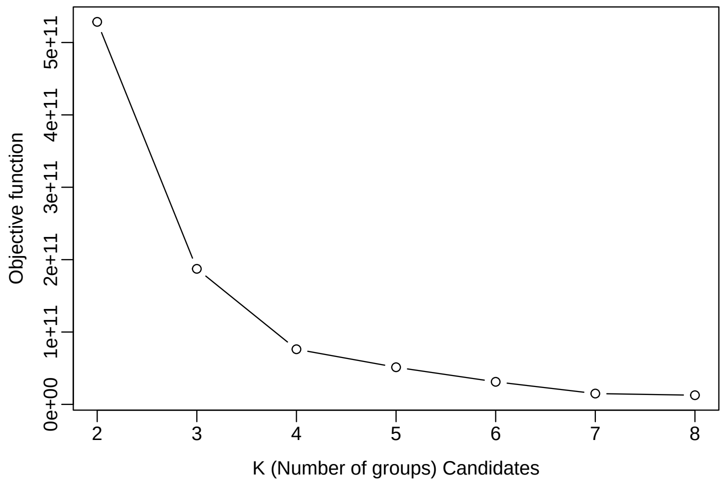
The elbow method evaluates the objective function, which measures the within-cluster variation (how tightly grouped the observations inside each cluster are). By plotting this value across several choices of K, we look for the point at which adding more clusters yields diminishing returns—i.e., where the objective function stops decreasing substantially. This helps balance interpretability with cluster separation.

Silhouette Score Analysis (explanation)

The silhouette method assesses how well each individual observation is assigned under different values of K. It does this by comparing the observation's distance to its own cluster with its distance to the nearest alternative cluster. Higher silhouette scores indicate clearer, more meaningful cluster assignments. Because the silhouette method evaluates both the suitability of K and the quality of each assignment, it generally provides stronger support for selecting the appropriate number of clusters than the elbow method alone.

Elbow plot

We will begin with the elbow plot. Plotting the objective function against various K candidates we got:



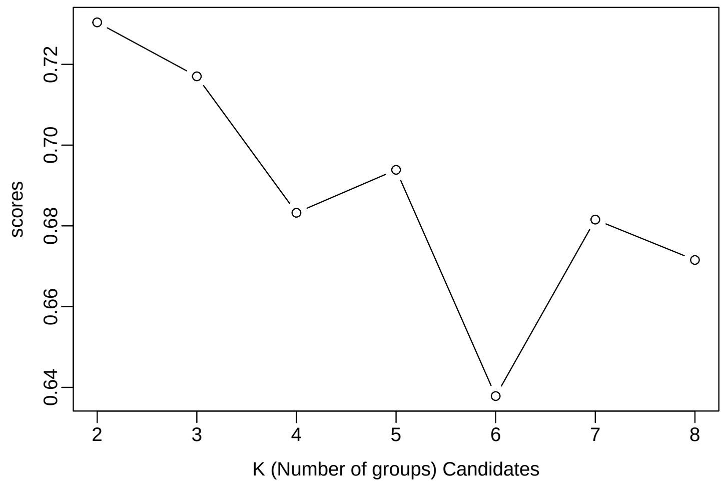
We observed in this plot that significant reductions in the objective function occur from 2 to 4 groups. Beyond 4 groups (K = 4), the reductions in the objective function are insignificant. Selecting any of those values for our model would result it a model that is unnecessarily complex.

Silhouette score analysis

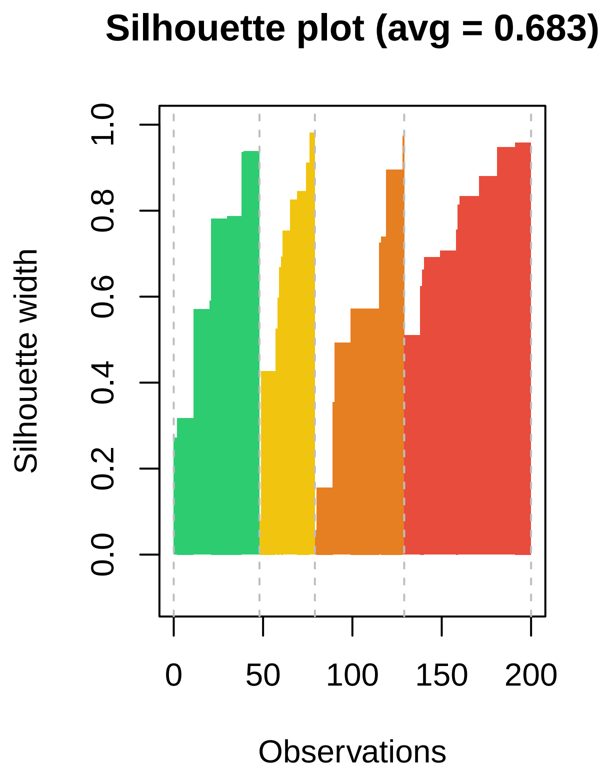
We then calculate how well each observation fits into its assigned cluster by computing silhouette scores for different values of K.

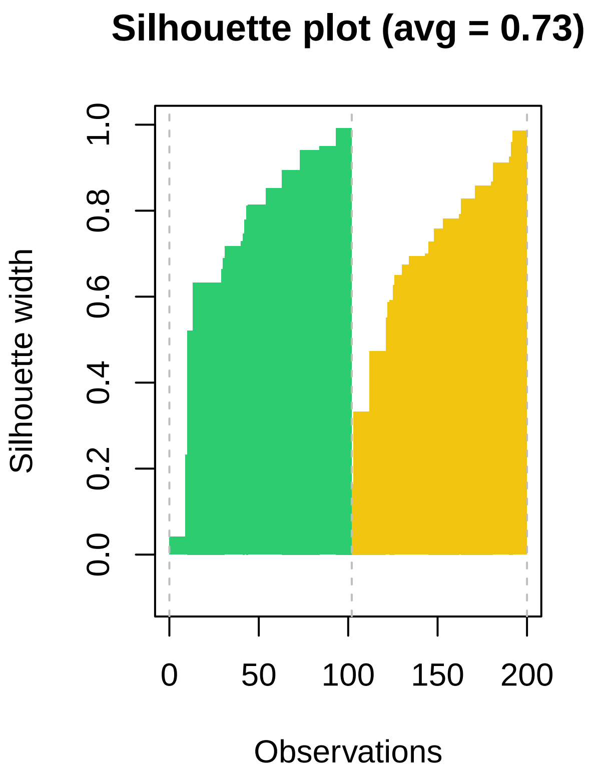
If the score is close to **1**, the point clearly belongs where it was placed. If it's close to **0**, the point sits between two clusters and could have gone either way.

By averaging these scores for each value of K, we get the plot shown below, which helps us decide how many clusters describe the data best.:



The silhouette average is highest (closets to 1) for a k value of 2 (suggesting 2 groups are sufficient for our model). This lines up nicely with our hypothesis but contradicts the results from the elbow plot. We will conduct a deeper analysis of the silhoutte scores, looking at how individual clusters fair under 2 and 4 as values of K. We begin with $K = 4$, plotting the individual silhouette scores under each cluster, with the mean of each cluster given on top.





4 clusters:
We see that three of the clusters are relatively poorly assigned compared to the one. The third cluster is particularly bad.

2 clusters:
The silhouette scores indicate better assignment. This tells us that 2 is the optimal value for K, meaning the data naturally forms two major groups. This matches what we already observed in our exploratory data analysis. Running the algorithm with two means produces two broad clusters:

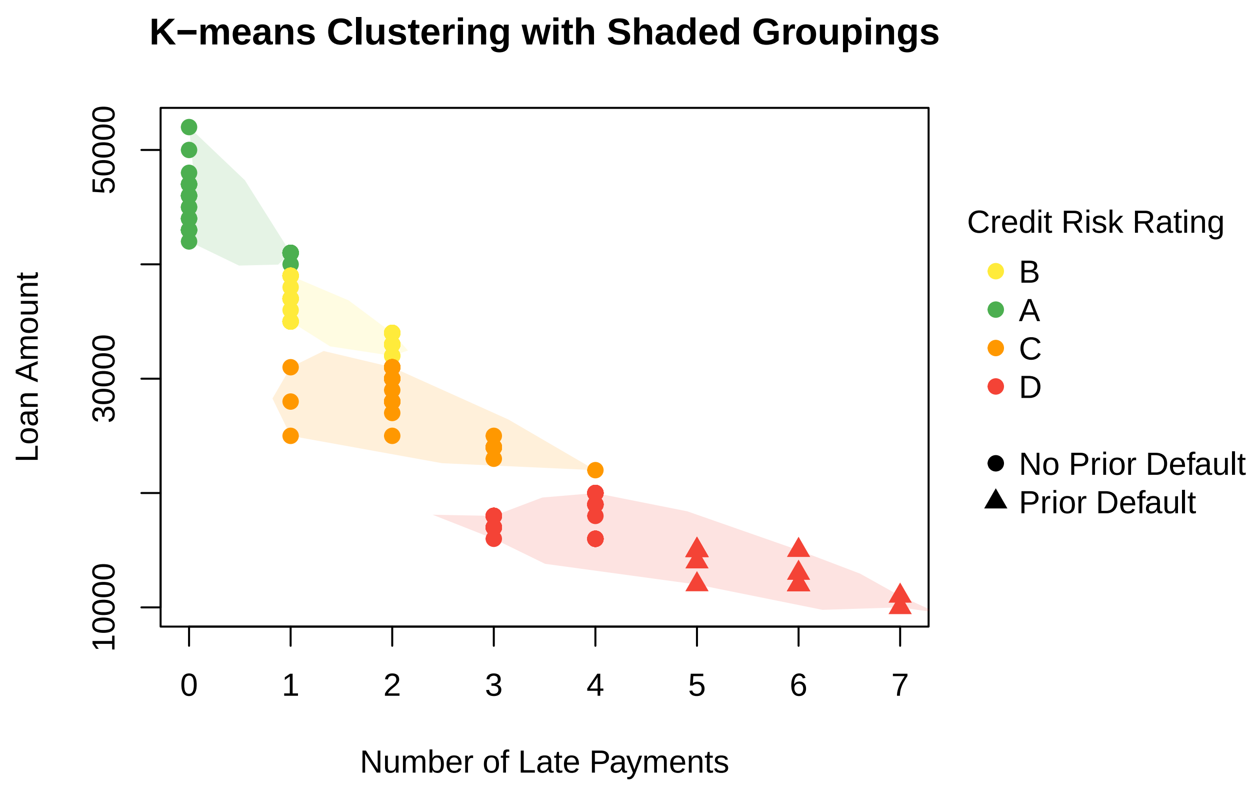
- High loan amounts, few late payments, and mostly no prior default
- Low loan amounts, many late payments, and a history of prior default

The silhouette analysis suggests that any further splits would occur *within* these broad groups. However, using only two clusters is too coarse, as it forces all clients into one of two categories. This can lead to the loss of good business because it does not distinguish between degrees of risk within each group.

By allowing for four clusters, we obtain a more refined ranking structure that gives each client a risk "grade" based on their characteristics. This grading system is detailed enough to give the bank flexibility in setting loan amounts and interest rates, while still remaining simple enough to implement without becoming overly complex or inefficient

Running final K-means

Running the final iteration with 4 means gives us:



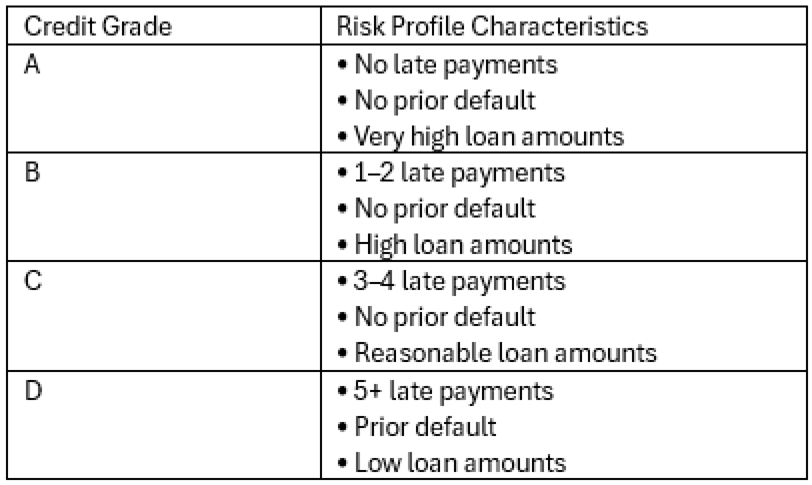
How we interpret the algorithm's output

Once K-means assigns each client to one of the four clusters, we examine the average values of the key variables in each group.

For each cluster, the algorithm's cluster centre (mean) shows the typical behaviour of clients in that group. By comparing these centres across the clusters, we can see clear patterns:

- one cluster has very high loan amounts and no negative behaviour
- another shows high loan amounts but a small number of late payments
- another shows moderate loan amounts with several late payments
- the last has low loan amounts and consistently high late payments with prior default

These patterns allow us to label the clusters in a meaningful way (A–D grades), moving from safest behaviour to highest risk.



Credit Grade	Risk Profile Characteristics
A	• No late payments • No prior default • Very high loan amounts
B	• 1–2 late payments • No prior default • High loan amounts
C	• 3–4 late payments • No prior default • Reasonable loan amounts
D	• 5+ late payments • Prior default • Low loan amounts

5. Key Insights

Our findings show that the number of late payments and whether a client has previously defaulted are two of the most important predictors of credit risk. The likelihood that a current client will default increases proportionally with the number of late payments they have accumulated. In addition, past defaulters consistently display riskier behavior—typically having more late payments than clients with no history of default.

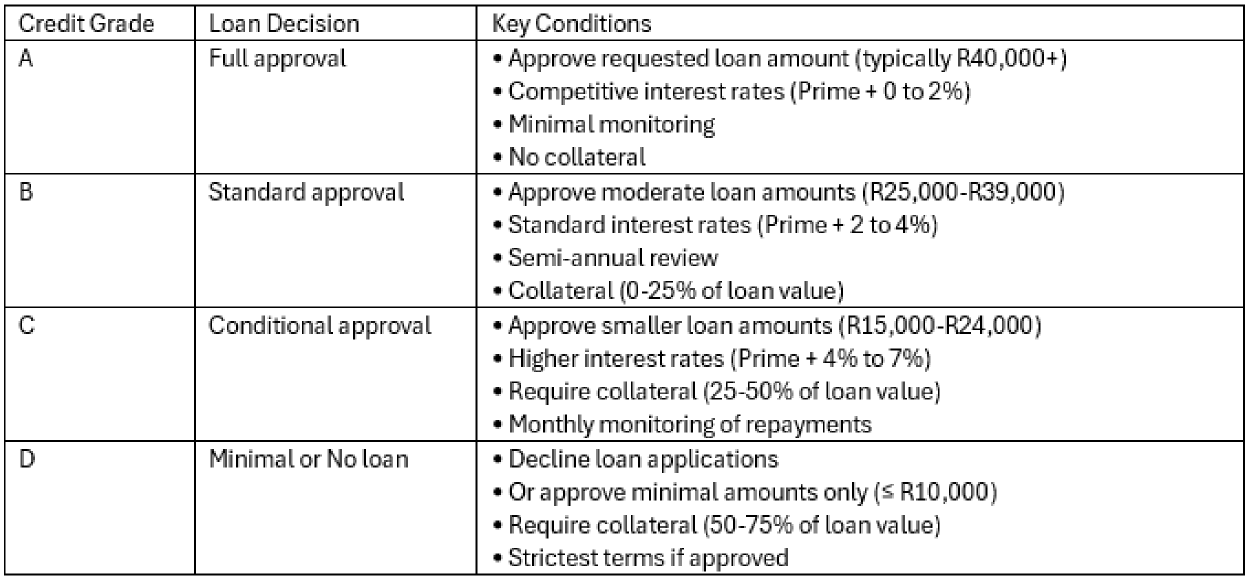
The bank may achieve better results with a credit decision system that is flexible but not overly complex. Our model groups clients based on shared characteristics and uses these groupings to determine loan approval and lending conditions. With four categories, the system is broad enough to handle a wide range of customer profiles while still being simple to apply operationally.

- **Category 1:** Clients with excellent repayment discipline, even at high loan amounts ("A-grade" clients).
- **Category 2:** Low-risk clients with strong repayment behaviour, no history of default, and generally high loan amounts.
- **Category 3:** Moderate-risk clients with average repayment behaviour, no prior defaults, and medium-sized loan amounts. These clients may be good candidates for cautious loan increases, but they should still be monitored.
- **Category 4:** High-risk clients with poor repayment habits, a history of default, and consistently low loan amounts.

This classification system can help the bank improve credit decisions by balancing simplicity with predictive power.

6. Quant's Recommendations

Crestline Bank's new credit decision framework:



Credit Grade	Loan Decision	Key Conditions
A	Full approval	• Approve requested loan amount (typically R40,000+) • Competitive interest rates (Prime + 0 to 2%) • Minimal monitoring • No collateral
B	Standard approval	• Approve moderate loan amounts (R25,000-R39,000) • Standard interest rates (Prime + 2 to 4%) • Semi-annual review • Collateral (0-25% of loan value)
C	Conditional approval	• Approve smaller loan amounts (R15,000-R24,000) • Higher interest rates (Prime + 4% to 7%) • Require collateral (25-50% of loan value) • Monthly monitoring of repayments
D	Minimal or No loan	• Decline loan applications • Or approve minimal amounts only (≤ R10,000) • Require collateral (50-75% of loan value) • Strictest terms if approved

7. Appendix

```r
# ------------------------------------------------------------------
# LOADING THE DATA

knitr::opts_chunk$set(echo = TRUE)

dat1 = read.table('Crestline Bank Clients Dataset.txt', header = TRUE, sep = ",")
attach(dat1)
dat2= data.frame(PriorDefault= PriorDefault, NumLatePayments=NumLatePayments,
                 LoanAmount=LoanAmount)
attach(dat2)


# ------------------------------------------------------------------

# EXPLORATOTY DATA ANALYSIS (EDA)

boxplot(NumLatePayments ~ PriorDefault, data = dat2, main = "Late Payments by Prior
        Default")

boxplot(LoanAmount ~ PriorDefault, data = dat2, main = "Loan Amount by Prior Default
        Status")

plot(NumLatePayments~LoanAmount, pch = 16, col = 'darkgreen', data = dat2,
     main = 'Loan Amount vs Late Payments')

# ------------------------------------------------------------------

# OBJECTIVE FUNCTION

library(dplyr)
#Within cluster sums
sum.clust.k = function(X,Mean.k){
  vec = c()
  for (i in 1:nrow(X)  ){
    sq.dis = t((X[i,]-Mean.k))%*%(X[i,]-Mean.k)
    vec[i] = sq.dis
  }
return(sum(vec))
}

#Total sum across clusters
sum.cross.clust = function(clust.sums, clust.nums ){
  obj = 0
  for (i in 1:length(clust.sums)){
    obj = obj +  clust.sums[i]*clust.nums[i]
  }
  return(obj)
}

# ------------------------------------------------------------------
# A FUNCTION FOR LLOYD'S ALGORITHM

Lloyd.means <- function(X, K, Iterations) {
  set.seed(5050)

  # Initialise means randomly
  means.ind <- sample(1:nrow(X), K)
  means <- X[means.ind, , drop = FALSE]

  labels <- numeric(nrow(X))  # <-- FIX 1: store all labels

  for (t in 1:Iterations) {

    # Empty cluster lists
    clust.list <- vector("list", K)
    for (i in 1:K) clust.list[[i]] <- matrix(nrow = 0, ncol = ncol(X))

    # Assign points
    for (row in 1:nrow(X)) {
      dis <- numeric(K)
      for (k in 1:K) {
        diff <- X[row, ] - means[k, ]
        dis[k] <- sum(diff^2)
      }
      clus.ind <- which.min(dis)
      clust.list[[clus.ind]] <- rbind(clust.list[[clus.ind]], X[row, ])

      labels[row] <- clus.ind   # <-- FIX 2: save label
    }

    # Handle empty clusters
    for (m in 1:K) {
      if (nrow(clust.list[[m]]) == 0) {

        # compute distance for each point to nearest mean
        all_dists <- numeric(nrow(X))

        for (i in 1:nrow(X)) {
          diffs <- sweep(means, 2, X[i, ], FUN = "-")
          dists <- rowSums(diffs^2)
          all_dists[i] <- min(dists)
        }

        furthest <- which.max(all_dists)
        clust.list[[m]] <- matrix(X[furthest, ], nrow = 1)
      }
    }

    # Update means
    for (m in 1:K) {
      means[m, ] <- colMeans(clust.list[[m]])
    }
  }
  return(list(means = means, clusters = clust.list, labels = labels))
}

# ------------------------------------------------------------------

# DATA MANIPULATION

dat = read.csv('Crestline Bank Clients Dataset.txt',header = TRUE)
attach(dat)
X.frame = dat[,c(7,8,9)]
Def.col = c()
for (i in PriorDefault){
  if (i == 'Yes'){
    Def.col = c(Def.col,1)
  }else{
    Def.col = c(Def.col,0)
  }
}
X.frame$PriorDefault = Def.col
X = as.matrix(X.frame)


# ------------------------------------------------------------------


# ELBOW PLOT

par(mar = c(3.5, 3.5, 1.5, 1), mgp = c(2, 0.5, 0))

k.values = c(2,3,4,5,6,7,8)
obj = c()
for (i in k.values){
  Results = Lloyd.means(X,i,100)
  means = Results$means
  clusters = Results$clusters
  clust.sums = c()
  clust.nums = c()
  for (k in 1:i){
   summ = sum.clust.k(clusters[[k]],means[k,])
    clust.sums = c(clust.sums,summ)
    clust.nums = c(clust.nums, nrow(clusters[[k]]))
  }
  obj = c(obj,sum.cross.clust(clust.sums,clust.nums))

}
plot(obj~k.values, type = 'b', xlab = 'K (Number of groups) Candidates',
     ylab = 'Objective function')

# ------------------------------------------------------------------

# FUNCTION FOR SILHOUETTE SCORES

ov.sil.score <- function(X, Means, K) {
```

```r
  sils <- numeric(nrow(X))  # preallocate

  for (i in 1:nrow(X)) {
    # Compute distances from point i to all cluster centers
    distances <- numeric(K)
    for (k in 1:K) {
      diff <- X[i, ] - Means[k, ]
      distances[k] <- sqrt(sum(diff^2))
    }

    # Sort distances to get closest and second closest cluster
    ord.dis <- sort(distances)      # ascending order
    a <- ord.dis[1]                 # distance to own cluster (closest)
    b <- ord.dis[2]                 # distance to next closest cluster

    # Silhouette-like score: (b - a) / b
    sils[i] <- (b - a) / b
  }

  return(list(avg.sil.score = mean(sils), sil.scores = sils))
}

# ------------------------------------------------------------------


# AVERAGE SILHOUETTE SCORES PLOT

par(mar = c(3.5, 3.5, 1.5, 1), mgp = c(2, 0.5, 0))

k.values = c(2,3,4,5,6,7,8)
scores = c()
for (i in k.values){
  means = Lloyd.means(X,i,100)$means
  score = ov.sil.score(X, means,i)$avg.sil.score
  scores = c(scores,score)
}
plot(scores~k.values, type = 'b',xlab = 'K (Number of groups) Candidates')

# ------------------------------------------------------------------


# INDIVIDUAL CLUSTER SILHOUETTE PLOTS (2 VS 4 COMPARISON)

par(mfrow = c(1, 2))

# Run Lloyd's algorithm
results <- Lloyd.means(X, 4, 100)
means <- results$means
clusters <- results$clusters
labels <- results$labels  # track cluster of each observation

# Compute silhouette scores for all points
sil_scores <- ov.sil.score(X, means, 4)$sil.scores
```

```r
avg_sil <- ov.sil.score(X, means, 4)$avg.sil.score

# Prepare a data frame for plotting
sil_df <- data.frame(
  cluster = factor(labels, levels = 1:4),
  sil_score = sil_scores
)

# Sort silhouette scores within each cluster
sil_df <- sil_df %>%
  group_by(cluster) %>%
  arrange(sil_score, .by_group = TRUE) %>%
  mutate(obs_in_cluster = row_number()) %>%
  ungroup()

# --- Classic silhouette plot in base R ---
# Compute cumulative x positions
cluster_sizes <- table(sil_df$cluster)
cum_sizes <- c(0, cumsum(cluster_sizes))
colors <- c("#2ECC71", "#F1C40F", "#E67E22", "#E74C3C")  # cluster colors

plot(NULL, xlim = c(0, nrow(X)), ylim = c(-0.1,1),
     xlab = "Observations", ylab = "Silhouette width",
     main = paste0("Silhouette plot (avg = ", round(avg_sil, 3), ")"))

# Draw bars for each cluster
for (i in 1:4) {
  idx <- which(sil_df$cluster == i)
  x_start <- cum_sizes[i]
  x_end <- cum_sizes[i+1]
  rect(xleft = x_start + 0:(length(idx)-1),
       ybottom = 0,
       xright = x_start + 1: length(idx),
       ytop = sil_df$sil_score[idx],
       col = colors[i],
       border = NA)
}

# Add vertical lines between clusters
abline(v = cum_sizes, lty = 2, col = "gray")

# Run Lloyd's algorithm
results <- Lloyd.means(X, 2, 100)
means <- results$means
clusters <- results$clusters
labels <- results$labels  # track cluster of each observation

# Compute silhouette scores for all points
sil_scores <- ov.sil.score(X, means, 2)$sil.scores
avg_sil <- ov.sil.score(X, means, 2)$avg.sil.score

# Prepare a data frame for plotting
sil_df <- data.frame(
  cluster = factor(labels, levels = 1:2),
  sil_score = sil_scores
)

# Sort silhouette scores within each cluster
sil_df <- sil_df %>%
  group_by(cluster) %>%
  arrange(sil_score, .by_group = TRUE) %>%
  mutate(obs_in_cluster = row_number()) %>%
  ungroup()

# --- Classic silhouette plot in base R ---
# Compute cumulative x positions
cluster_sizes <- table(sil_df$cluster)
cum_sizes <- c(0, cumsum(cluster_sizes))
colors <- c("#2ECC71", "#F1C40F", "#E67E22", "#E74C3C")  # cluster colors

plot(NULL, xlim = c(0, nrow(X)), ylim = c(-0.1,1),
     xlab = "Observations", ylab = "Silhouette width",
     main = paste0("Silhouette plot (avg = ", round(avg_sil, 3), ")"))

# Draw bars for each cluster
for (i in 1:2) {
  idx <- which(sil_df$cluster == i)
  x_start <- cum_sizes[i]
  x_end <- cum_sizes[i+1]
  rect(xleft = x_start + 0:(length(idx)-1),
       ybottom = 0,
       xright = x_start + 1: length(idx),
       ytop = sil_df$sil_score[idx],
       col = colors[i],
       border = NA)
}

# Add vertical lines between clusters
abline(v = cum_sizes, lty = 2, col = "gray")

par(mfrow = c(1, 1))

# -------------------------------------------------------------------


# FINAL K-MEANS PLOT (4 MEANS)

# Run K-means with 4 clusters to get results
results = Lloyd.means(X, 4, 100)

# Create separate vectors for defaulters and non-defaulters
default_indices <- which(X[,1] == 1)   # PriorDefault = 1 (Yes)
nondefault_indices <- which(X[,1] == 0)   # PriorDefault = 0 (No)

# Define color palette: Green -> Yellow -> Orange -> Red for credit risk gradient
cluster_colors <- c("#FFEB3B",
                    "#4CAF50",
                    "#FF9800",
                    "#F44336")

# Alternative approach using layout
layout(matrix(c(1, 2), ncol = 2), widths = c(4, 1.2))  # Increased legend width to 1.2

# First plot area
par(mar = c(5, 4, 4, 1))  # Normal margins for plot
plot(NA, NA,
     xlim = range(X[,2]), ylim = range(X[,3]),
     xlab = "Number of Late Payments", ylab = "Loan Amount",
     main = "K-means Clustering with Shaded Groupings")

# Draw shaded regions and points
for (k in 1:4) {
  cluster_points <- X[results$labels == k, 2:3]

  if (nrow(cluster_points) > 2) {
    hull <- chull(cluster_points)
    boundary <- cluster_points[hull, ]
    n_points <- nrow(boundary)
    organic_boundary <- matrix(0, nrow = n_points * 2, ncol = 2)

    for (i in 1:n_points) {
      organic_boundary[i*2 - 1, ] <- boundary[i, ]
      next_i <- ifelse(i == n_points, 1, i + 1)
      mid_x <- (boundary[i,1] + boundary[next_i,1]) / 2
      mid_y <- (boundary[i,2] + boundary[next_i,2]) / 2
      center_x <- mean(boundary[,1])
      center_y <- mean(boundary[,2])
      dir_x <- mid_x - center_x
      dir_y <- mid_y - center_y
      push <- runif(1, 1.1, 1.4)
      organic_boundary[i*2, 1] <- center_x + dir_x * push + runif(1, -0.2, 0.2)
      organic_boundary[i*2, 2] <- center_y + dir_y * push + runif(1, -200, 200)
    }

    organic_boundary <- rbind(organic_boundary, organic_boundary[1, ])

    # Fill with light color using new color gradient
    polygon(organic_boundary[,1], organic_boundary[,2],
            col = adjustcolor(cluster_colors[k], alpha.f = 0.15), border = NA)
  }
}

# Plot points with new color gradient
if (length(nondefault_indices) > 0) {
  # Map cluster labels to new colors for non-defaulters
  point_colors <- cluster_colors[results$labels[nondefault_indices]]
  points(X[nondefault_indices, 2], X[nondefault_indices, 3],
         col = point_colors, pch = 19, cex = 1)
}
```

```r
if (length(default_indices) > 0) {
  # Map cluster labels to new colors for defaulters
  point_colors <- cluster_colors[results$labels[default_indices]]
  points(X[default_indices, 2], X[default_indices, 3],
         col = point_colors, pch = 17, cex = 1.2)
}

# Second area for legend
par(mar = c(5, 0, 4, 0))  # Remove right margin for legend area
plot(1, type = "n", axes = FALSE, xlab = "", ylab = "")

# Add legend in the empty plot area with cleaner labels
legend("left",
       legend = c("B", "A", "C", "D",
                  "",  # Empty line for spacing
                  "No Prior Default", "Prior Default"),
       col = c(cluster_colors, NA, "black", "black"),
       pch = c(19, 19, 19, 19, NA, 19, 17),
       bty = "n", cex = 1,
       pt.cex = c(1, 1, 1, 1, NA, 1, 1.2),
       x.intersp = 0.8,  # Adjust horizontal spacing
       title = "Credit Risk Rating",  # Updated title to match color semantics
       title.adj = 0)


# -------------------------------------------------------------------
```