**Project Title: Customer Segmentation Model**

**Executive Summary:**

Using transaction data from customers, this project conducts a Recency, Frequency, and Monetary (RFM) analysis, followed by k-means clustering to segment customers. The resulting insights are utilized to create personalized email marketing campaigns for customers who have previously engaged but have not made recent purchases.

**The Business Challenge:**

To get the most out of marketing campaigns, it's important to reach out to potential customers who have shown interest but haven't made a recent purchase. Our project offers a strategic solution to reconnect with these customers by developing a customer segmentation model and personalized marketing approach.

**Data and Methodology:**

The data is sourced from UCI's Machine Learning Repository, “Online Retail Store,” which comprises transactional information from a UK-based e-commerce store over a year. The methodology involves RFM analysis, which determines customers' recency, frequency, and monetary attributes, followed by k-means clustering to segment customers into different groups. Each customer's profile includes their most common stock code, country, and most frequent purchase item.

**Findings and Implications:**

After analyzing customer data, six unique segments were identified based on RFM values. By utilizing this model, personalized engagement with each segment can be optimized, leading to potential increases in customer conversion and loyalty. This segmentation allows for more focused marketing efforts, resulting in reduced costs and improved customer satisfaction through tailored content.

**Future Directions:**

The findings from this project will help create better marketing campaigns in the future. One possible next step is to conduct A/B testing to see if personalized emails or generalized campaigns are more effective. In the future, we can also include customer demographics to further improve segmentation.

**Appendix**

**References:**

"UCI Machine Learning Repository: Online Retail Data Set." UCI Machine Learning Repository. Web. [[https://archive.ics.uci.edu/ml/datasets/Online+Retail](https://archive.ics.uci.edu/ml/datasets/Online+Retail)](https://archive.ics.uci.edu/ml/datasets/Online+Retail%5D(https://archive.ics.uci.edu/ml/datasets/Online+Retail))

**Final Code**:

| # Data Loading library(tidyverse) library(readxl)   online\_retail\_data <- read\_excel("Online Retail.xlsx") %>%  na.omit(online\_retail\_data)  # Data Cleaning and Pre-processing library(dplyr)  # We will assume the 'InvoiceDate' column is the data of purchase,  # and the 'CustomerID' is the ID of the customer. # Lastly, the 'UnitPrice' is the cost of the price per unit.  str(online\_retail\_data) sum(is.na(online\_retail\_data))  # All of our data seems to be formatted correctly, and free of any missing values. # Since our time is POSIXct, there is no need to convert the date. # We will now store the last date of the data last\_date <- max(online\_retail\_data$InvoiceDate)  # We will also create a new column for total amount online\_retail\_data <- mutate(online\_retail\_data, TotalAmount = UnitPrice \* Quantity)   #Calculating RFM metrics customers <- online\_retail\_data %>%  group\_by(CustomerID) %>%  summarise(Recency = as.numeric(last\_date - max(InvoiceDate)),  Frequency = n(),  Monetary = sum(TotalAmount))  # Now that we have created RFM metrics of our customers, we will now segment them. # We will use a k-means clustering algorithm # Firstly, we need to determine an optimal numbers of cluster. #So we will create a scree plot.  library(purrr) library(ggplot2) library(cluster) library (NbClust) library (clustertend) library (factoextra)  # Now that we have created RFM metrics of our customers, we will now segment them. # We will use a k-means clustering algorithm # Firstly, we need to determine an optimal numbers of cluster. #So we will create a scree plot.  # Scaling the RFM Data rfm\_data <- scale(customers[,2:4])  # The range of k values that will be run k\_values <- 2:15  #Calculating the within-cluster sum of squares (WSS) wss <- wss <- map\_dbl(k\_values, function(k) {  kmeans(rfm\_data, centers = k)$tot.withinss })  # Scree Plot # Create an elbow plot elbow\_plot <- tibble(k = k\_values, wss = wss) %>%  ggplot(aes(x = k, y = wss)) +  geom\_line() +  geom\_point() +  labs(x = "Number of clusters (k)", y = "Within-cluster Sum of Squares (WSS)") +  ggtitle("Elbow Plot") +  theme\_minimal()  # Display the elbow plot print(elbow\_plot)  # It seems that 6 clusters might be best, but I want to do a silhouette analysis to make sure.  set.seed(456) k\_means <- kmeans(rfm\_data, centers = 6) customers$cluster <- k\_means$cluster  # Define function to calculate mode calculate\_mode <- function(x) {  ux <- unique(x)  ux[which.max(tabulate(match(x, ux)))] }  # Add cluster assignment back to original data online\_retail\_data <- online\_retail\_data %>%  inner\_join(customers, by = "CustomerID")  # Group by CustomerID and cluster, then summarise the other columns customer\_cluster\_info <- online\_retail\_data %>%  group\_by(CustomerID, cluster) %>%  summarise(  most\_common\_stockcode = calculate\_mode(StockCode),  most\_common\_country = calculate\_mode(Country),  item = names(which.max(table(Description))),  Recency = first(Recency),  Frequency = first(Frequency),  Monetary = first(Monetary),  .groups = "drop"  )  customer\_cluster\_info  ## MOCK MAILCHIMP EMAIL DELIVERY  # Install required packages install.packages("httr") install.packages("jsonlite")  # Load required packages library(httr) library(jsonlite)  # Function to send email to customers send\_email\_to\_customers <- function(df) {  # Mailchimp API URL  url <- "https://usX.api.mailchimp.com/3.0/campaigns" # replace "usX" with your datacenter    # Mailchimp API key  api\_key <- "your\_api\_key" # replace with your actual Mailchimp API key    for(i in 1:nrow(df)) {  # Define the email body  body <- list(  recipients = list(list(email\_address = df[i, 'Email'],   email\_type = "html")),  settings = list(subject\_line = "Personalized Recommendations Based on Your Purchases",   from\_name = "Your Name",   reply\_to = "your-email@example.com"),  template = list(  id = "your\_template\_id",   sections = list(  item = df[i, 'item'],  Recency = df[i, 'Recency'],  Frequency = df[i, 'Frequency'],  Monetary = df[i, 'Monetary']  )  )  )    # Make the POST request  response <- POST(url, body = body, encode = "json",   authenticate("anystring", api\_key, type = "basic"),  add\_headers("Content-Type" = "application/json"))    # Check the status of the request  if(response$status\_code == 200) {  print(paste("Email sent successfully to", df[i, 'Email']))  } else {  print(paste("Failed to send email to", df[i, 'Email']))  }  } }  # Call the function send\_email\_to\_customers(customer\_cluster\_info)  ## Please note that you must: ## 1. Replace "your\_api\_key" with an actual Mailchimp API key ## 2. Replace "usX" in the URL with the datacenter that corresponds to the account. ## 3. Replace "your-email@example.com" with the email to send from. ## 4. Replace "your\_template\_id" with the actual id of the template. ## 5. The original data frame does not include any actual emails. |
| --- |