Homework-5

Yash

2024-11-12

Homework-5

Solution A: Data Gathering and Integration

```
sales_data <- read_csv("online_sales_dataset.csv")

# Displaying the first few rows of the dataset
head(sales_data)
```

```
## # A tibble: 6 x 17
##   InvoiceNo StockCode Description  Quantity InvoiceDate     UnitPrice CustomerID
##       <dbl> <chr>     <chr>           <dbl> <chr>               <dbl>      <dbl>
## 1    221958 SKU_1964  White Mug          38 01-01-2020 00:~      1.71      37039
## 2    771155 SKU_1241  White Mug          18 01-01-2020 01:~     41.2       19144
## 3    231932 SKU_1501  Headphones         49 01-01-2020 02:~     29.1       50472
## 4    465838 SKU_1760  Desk Lamp          14 01-01-2020 03:~     76.7       96586
## 5    359178 SKU_1386  USB Cable         -30 01-01-2020 04:~    -68.1          NA
## 6    744167 SKU_1006  Office Chair       47 01-01-2020 05:~     70.2       53887
## # i 10 more variables: Country <chr>, Discount <dbl>, PaymentMethod <chr>,
## #   ShippingCost <dbl>, Category <chr>, SalesChannel <chr>, ReturnStatus <chr>,
## #   ShipmentProvider <chr>, WarehouseLocation <chr>, OrderPriority <chr>
```

(i) Overview of Online Sales Transaction Analysis

- This dataset represents a large volume of online sales transactions in anonymized format so as not to reveal customer-specific details. It contains all kinds of different data that usually appear in e-commerce analytics, including product descriptions, quantities sold, unit prices, and dates of transactions. Other demographic data includes CustomerID and geographic information, such as Country, that enables extensive demographic analysis. The operational details in terms of variables include: discounts, mode of payment, shipment cost, and shipment provider-when put together, they convey the nature of the operation for each transaction.
- The dataset presented is rich and forms a bedrock for deep analytics aimed at decoding the trend of sales, customer purchase behavior, and operational efficiencies within an e-commerce or retail environment. In fact, such an analysis may be useful in strategizing how best to optimize discounts, modes of payment, and usage of shipping service providers in light of customer satisfaction and efficiency in inventory management based on product demand trends.

(ii) Analytics Objectives

1. **Customer Segmentation:** The customer will be segmented accordingly. This would help in meaningful insights for focused marketing campaigns and service customers more effectively, hence reducing the cost of selling. Precise segmentation helps in exact targeting of communication for better engagement and high customer retention and value.
2. **Return Predictions:** Knowing what drives the products for returns will impact revenues considerably and more importantly, the logistical operations. A predictive capability will enable this to be developed in order to better risk manage. We can refine sales strategies by reducing the rate of returns and improving overall customer satisfaction by identifying possible returns before they occur.

Solution B: Data Exploration

- Let's first observe each field and it's datatype. Also, we look at the dataset manually to find out problems and outliers.

```
# Structure of Sales_data
str(sales_data)
```

```
## spc_tbl_ [49,782 x 17] (S3: spec_tbl_df/tbl_df/tbl/data.frame)
##  $ InvoiceNo        : num [1:49782] 221958 771155 231932 465838 359178 ...
##  $ StockCode        : chr [1:49782] "SKU_1964" "SKU_1241" "SKU_1501" "SKU_1760" ...
##  $ Description      : chr [1:49782] "White Mug" "White Mug" "Headphones" "Desk Lamp" ...
##  $ Quantity         : num [1:49782] 38 18 49 14 -30 47 25 8 19 40 ...
##  $ InvoiceDate      : chr [1:49782] "01-01-2020 00:00" "01-01-2020 01:00" "01-01-2020 02:00" "01-01-2
##  $ UnitPrice        : num [1:49782] 1.71 41.25 29.11 76.68 -68.11 ...
##  $ CustomerID       : num [1:49782] 37039 19144 50472 96586 NA ...
##  $ Country          : chr [1:49782] "Australia" "Spain" "Germany" "Netherlands" ...
##  $ Discount         : num [1:49782] 0.47 0.19 0.35 0.14 1.5 ...
##  $ PaymentMethod    : chr [1:49782] "Bank Transfer" "paypall" "Bank Transfer" "paypall" ...
##  $ ShippingCost     : num [1:49782] 10.79 9.51 23.03 11.08 NA ...
##  $ Category         : chr [1:49782] "Apparel" "Electronics" "Electronics" "Accessories" ...
##  $ SalesChannel     : chr [1:49782] "In-store" "Online" "Online" "Online" ...
##  $ ReturnStatus     : chr [1:49782] "Not Returned" "Not Returned" "Returned" "Not Returned" ...
##  $ ShipmentProvider : chr [1:49782] "UPS" "UPS" "UPS" "Royal Mail" ...
##  $ WarehouseLocation: chr [1:49782] "London" "Rome" "Berlin" "Rome" ...
##  $ OrderPriority    : chr [1:49782] "Medium" "Medium" "High" "Low" ...
##  - attr(*, "spec")=
##   .. cols(
##   ..   InvoiceNo = col_double(),
##   ..   StockCode = col_character(),
##   ..   Description = col_character(),
##   ..   Quantity = col_double(),
##   ..   InvoiceDate = col_character(),
##   ..   UnitPrice = col_double(),
##   ..   CustomerID = col_double(),
##   ..   Country = col_character(),
##   ..   Discount = col_double(),
##   ..   PaymentMethod = col_character(),
##   ..   ShippingCost = col_double(),
##   ..   Category = col_character(),
##   ..   SalesChannel = col_character(),
##   ..   ReturnStatus = col_character(),
```

```
##   ..   ShipmentProvider = col_character(),
##   ..   WarehouseLocation = col_character(),
##   ..   OrderPriority = col_character()
##   .. )
##  - attr(*, "problems")=<externalptr>
```

1. **InvoiceNo (numerical):** The uniquely identifying number for each single transaction. This variable can be used for the index to ensure that each transaction is analyzed as unique.
2. **StockCode (character):** This describes specifically stock keeping units pertaining to the products. It can provide essential trends that are product specific and link the inventory data to the sales data.
3. **Description (Categorical):** A textual description of the products sold. It can provide qualitative data that can be parsed or grouped for analysis of sales by product type or category if further categorization already beyond what Category provides is needed.
4. **Quantity (numerical):** The quantitative number of units involved in each transaction. It is very crucial in volumetric sale analysis and in determining consumer behavior during purchases, and large purchases detection or return market behavior gains particular relevance when the value for quantities is less than zero possibly meaning returns.

   Problem: Negative values might represent returns and hence would require some special treatment or consideration in the analysis.
5. **InvoiceDate (character):** The date or time of the occurrence of transactions. They are stored as text strings. To conduct time series analysis or to spot trends over time or carry out seasonal analysis, it's necessary to convert this to date/time format.

   Problem: Currently in text format, this will require the conversion of datetime format for any analysis with time requirements.
6. **UnitPrice (numerical):** The price of each individual product unit. This variable is relevant to the revenue analysis that can take place when combined with Quantity in total transaction value calculations.

   Problem: Review of negative numbers might be warranted since they may indicate refunds or errors have occurred.
7. **CustomerID (numerical):** A unique identifier for each customer. Useful in customer-level analysis, segmentation, and tracking customer behavior over time.

   Problem: There are missing values affecting any analysis tracking customer-specific behavior.
8. **Country (Categorical):** The country in which the transaction took place or that of the customer. Allows marketing segmentation and analysis.
9. **Discount (numerical):** Percentage reduction from the price possibly applied in a transaction. Analysis/insights gathered from discounting effects on sales volumes and patterns of customer purchases could be insightful.

   Problem: Values over 1 suggest possible error or misunderstanding in the way discounts calculated or represented.
10. **PaymentMethod (Categorical):** Describes how the transaction was paid for (e.g. bank transfer, PayPal). Variations in spelling should be corrected for accurate categorization.

    Problem: There are possible misspellings and inconsistencies since PayPall might actually refer to PayPal; it must be corrected in order to allow for accurate analysis.
11. **ShippingCost(Numerical):** Shipping charges applied on a product. This field may affect total transaction cost management and may influence customers' satisfaction and purchasing decisions.

    Problem: There are missing values that would have to be taken care of for an accurate cost analysis.

12. **Category (Categorical):** The category of products sold here. It can be analyzed to know the different sales and customer preferences for different types of products.

13. **SalesChannel (Categorical):** Whether the sale was made online or in-store. This would provide a gauge on the performances of each channel and tell the Buddha beyond some insight on how customers behave across different channels.

14. **ReturnStatus (Categorical):** This means the product been returned. This sends an important passage of info into returns, product satisfaction, or product issues.

15. **ShipmentProvider (Categorical):** The company in charge of transporting the product in case of shipping time and customer satisfaction.

16. **WarehouseLocation (Categorical):** The origin of the shipment product will prove useful in logistics modeling and may suggest possible improvements for inventory distribution.

    Problem: Missing values might affect logistics analysis.

17. **OrderPriority (Categorical):** This shows the priority of the order and could impact order processing and the timing of fulfillment.

```r
# Summary statistics
summary(sales_data)
```

```
##    InvoiceNo       StockCode         Description           Quantity
##  Min.   :100005   Length:49782       Length:49782       Min.   :-50.00
##  1st Qu.:324543   Class :character   Class :character   1st Qu.: 11.00
##  Median :552244   Mode  :character   Mode  :character   Median : 23.00
##  Mean   :550681                                        Mean   : 22.37
##  3rd Qu.:776364                                        3rd Qu.: 37.00
##  Max.   :999997                                        Max.   : 49.00
##
##  InvoiceDate          UnitPrice        CustomerID      Country
##  Length:49782       Min.   :-99.98   Min.   :10001   Length:49782
##  Class :character   1st Qu.: 23.59   1st Qu.:32751   Class :character
##  Mode  :character   Median : 48.92   Median :55165   Mode  :character
##                     Mean   : 47.54   Mean   :55033
##                     3rd Qu.: 74.61   3rd Qu.:77306
##                     Max.   :100.00   Max.   :99998
##                                      NA's   :4978
##     Discount      PaymentMethod       ShippingCost     Category
##  Min.   :0.0000   Length:49782       Min.   : 5.00   Length:49782
##  1st Qu.:0.1300   Class :character   1st Qu.:11.22   Class :character
##  Median :0.2600   Mode  :character   Median :17.50   Mode  :character
##  Mean   :0.2757                      Mean   :17.49
##  3rd Qu.:0.3800                      3rd Qu.:23.72
##  Max.   :1.9998                      Max.   :30.00
##                                      NA's   :2489
##  SalesChannel       ReturnStatus       ShipmentProvider   WarehouseLocation
##  Length:49782       Length:49782       Length:49782       Length:49782
##  Class :character   Class :character   Class :character   Class :character
##  Mode  :character   Mode  :character   Mode  :character   Mode  :character
##
##
##
##
```

```
##  OrderPriority
##  Length:49782
##  Class :character
##  Mode  :character
##
##
##
##
```

The following details can be gathered about the range of numerical fields and the nature of their distribution between them:

1. **InvoiceNo**: it ranges from 100005 to 999997. Somewhat uniformly distributed, with averages generally placed closer to the mid-level between the minimum and maximum values. A feature of an identifier.

2. **Quantity**: It ranges from -50 to 49 It's close to 22.37, the mean, which is apparently close to the median of 23, hence a more symmetrical distribution.

   Negatively skewed: Negative values present here indicate possible returns or reverse transactions. If negative values are omitted from the dataset, skewness would probably change. The vast range from the first quartile to the third implies enough volatility in the quantity of items ordered in a transaction.

3. **UnitPrice**: It ranges from -99.98 to 100. Mean is 47.54 which is close to the median of 48.92, indicating that the distribution is symmetric around these values.

   Negatively skewed: The most negative value is probably due to entry errors, or similar adjustments due to returns. Exclusion of these might make it seem a bit more right-skewed, with the mean slightly above the median, although not much.

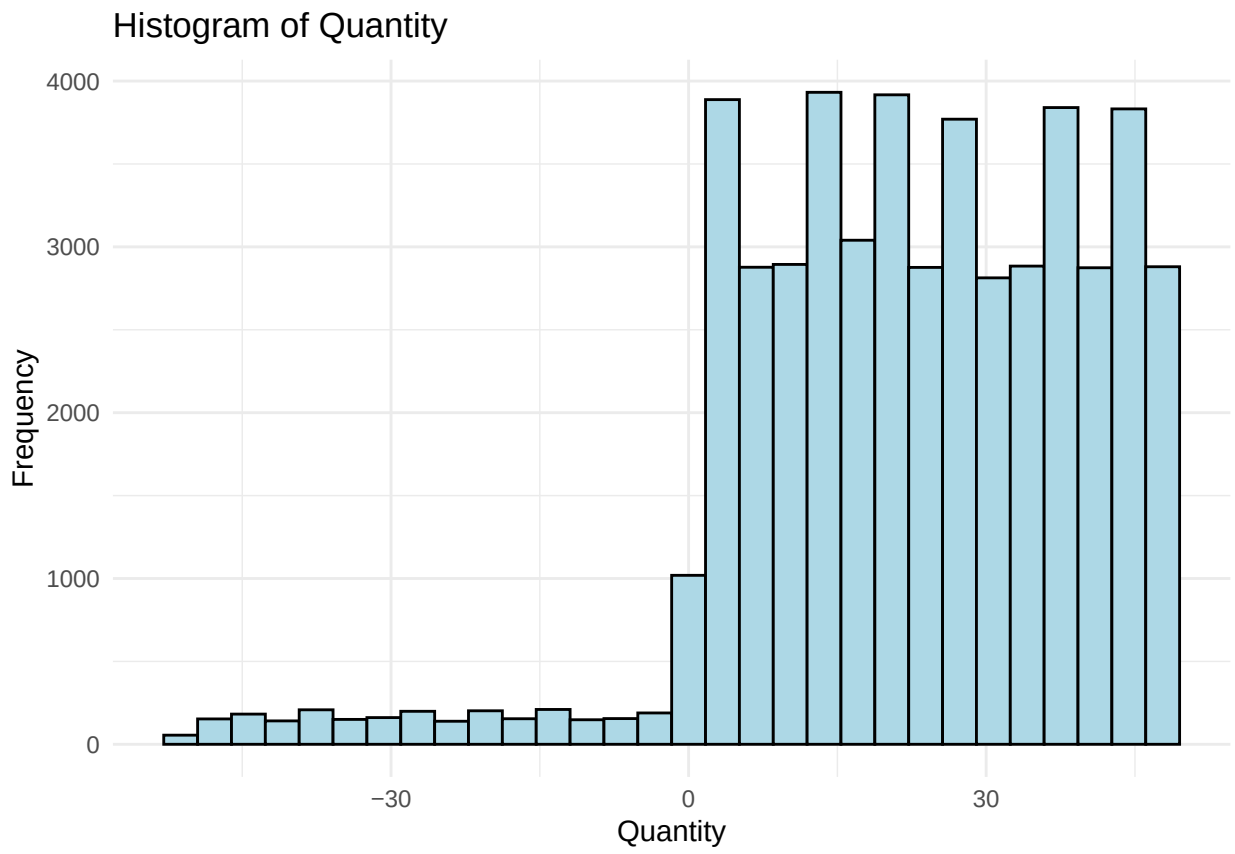
4. **CustomerID**: It range from 10001 to 99998. There are 4,978 missing values. Because it is virtually impossible to draw much inference from a customer's ID, skewness being a nominal variable is employed here strictly as an identifier. Absent IDs could affect analyses made at the customer level.

5. **Discount**: It range from 0.0000 to 1.9998. The Mean is 0.2757 and the Median is 0.2600, showing moderate symmetry of values around them.

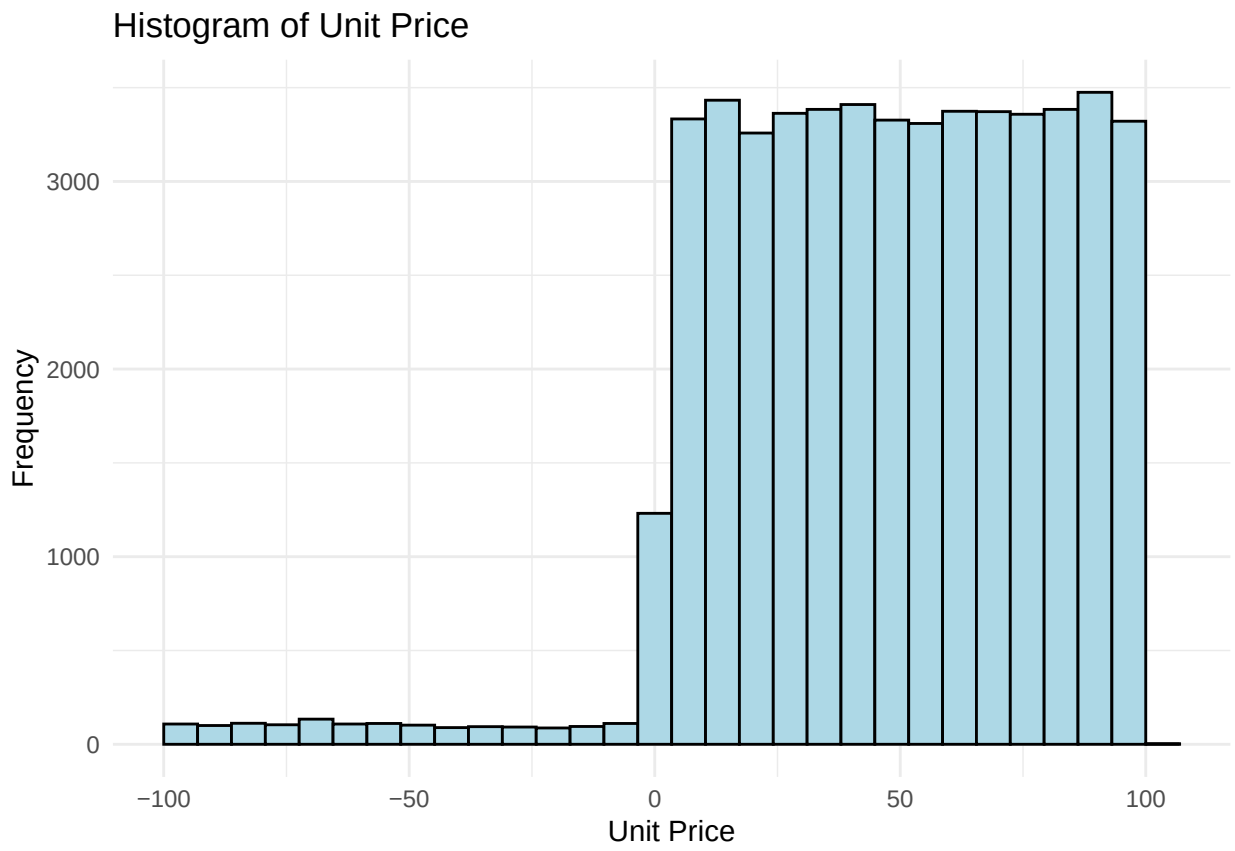
   Negatively skewed: The maximum value is almost 2-crude which will be looked at as an error-because a normal discount lies between 0 and 1. This means prima facie indication of right skewness of the distribution with an occasional higher erroneous value.

6. **ShippingCost**: Minimum: 5.00, maximum: 30.00 Mean: 17.49, median: 17.50, suggests the distribution of the shipping costs is not highly skewed. Although, it has missing values-2,489 With a mean close to the median and fairly concentrated between the 11.22 and 23.72 interval, it seems that shipping costs are quite normal even if shown not skewed around.

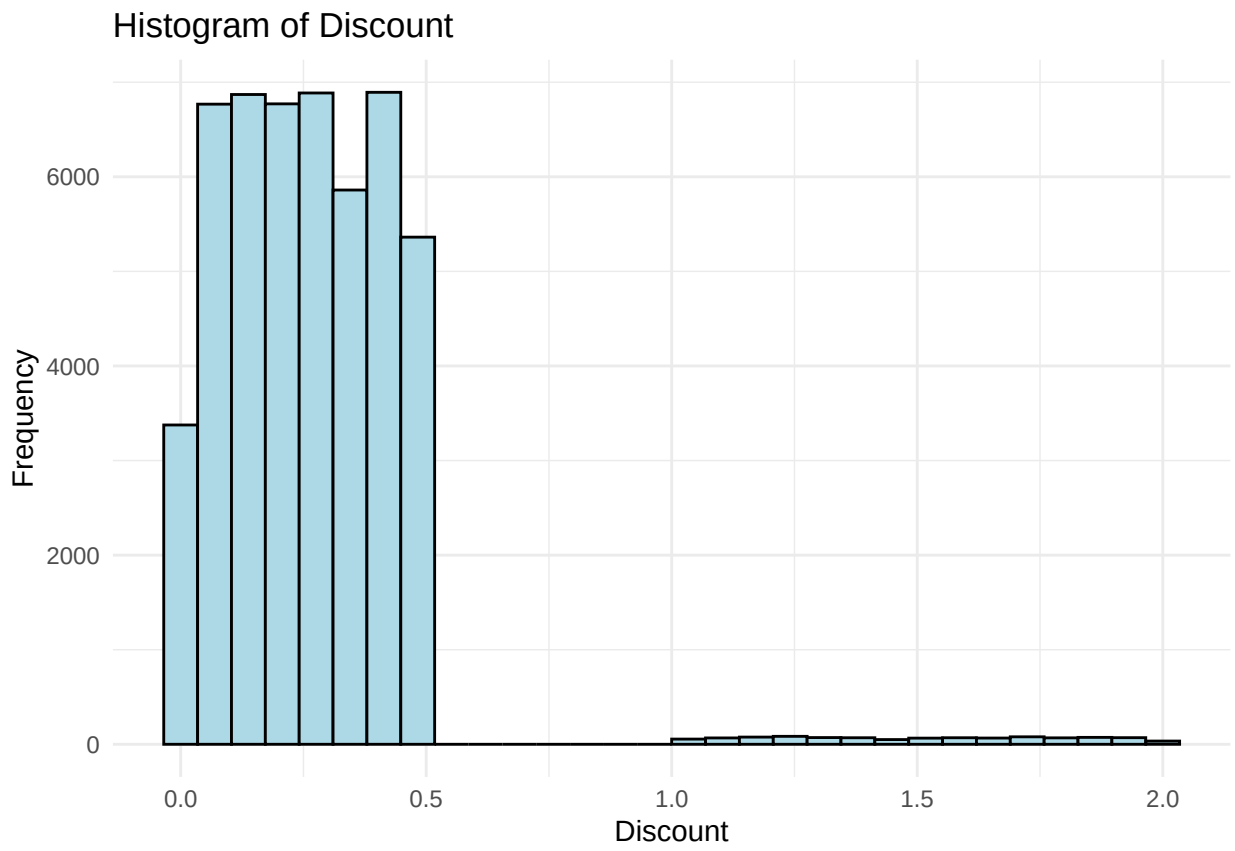
```r
# Plotting histogram for Quantity
ggplot(data = sales_data, aes(x = Quantity)) +
  geom_histogram(bins = 30, fill = 'lightblue', color = 'black') +
  ggtitle("Histogram of Quantity") +
  xlab("Quantity") +
  ylab("Frequency") +
  theme_minimal()
```



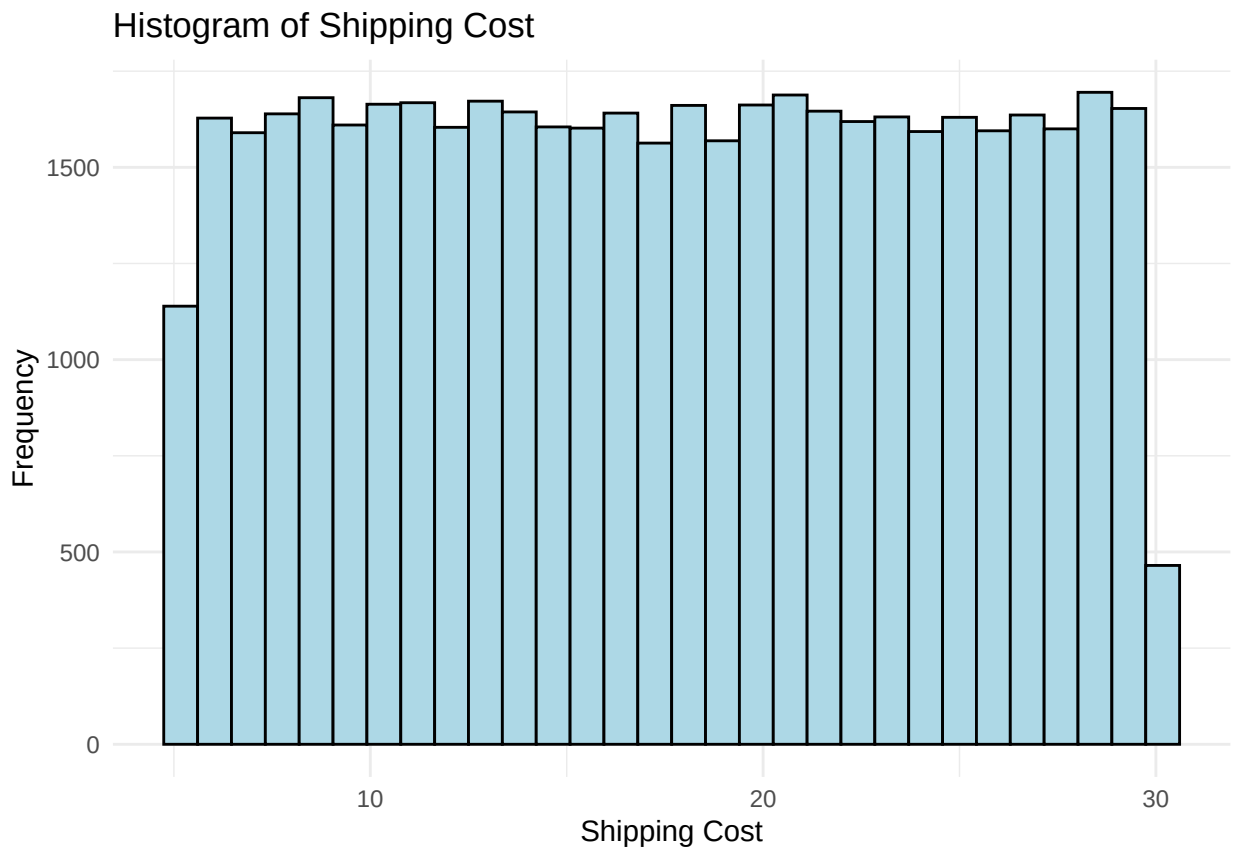

```r
# Plotting histogram for UnitPrice
ggplot(data = sales_data, aes(x = UnitPrice)) +
  geom_histogram(bins = 30, fill = 'lightblue', color = 'black') +
  ggtitle("Histogram of Unit Price") +
  xlab("Unit Price") +
  ylab("Frequency") +
  theme_minimal()
```




```r
# Plotting histogram for Discount
ggplot(data = sales_data, aes(x = Discount)) +
  geom_histogram(bins = 30, fill = 'lightblue', color = 'black') +
  ggtitle("Histogram of Discount") +
  xlab("Discount") +
  ylab("Frequency") +
  theme_minimal()
```




```
# Plotting histogram for ShippingCost
ggplot(data = sales_data, aes(x = ShippingCost)) +
  geom_histogram(bins = 30, fill = 'lightblue', color = 'black') +
  ggtitle("Histogram of Shipping Cost") +
  xlab("Shipping Cost") +
  ylab("Frequency") +
  theme_minimal()
```



1. **Quantity Histogram Analysis:** From the histogram of Quantity, it's clearly observable that it is symmetric around its mean and median. As observed from the summary statistics, the positive and negative values here represent returns as well as purchases, thus suggesting ranges between -50 and 49.

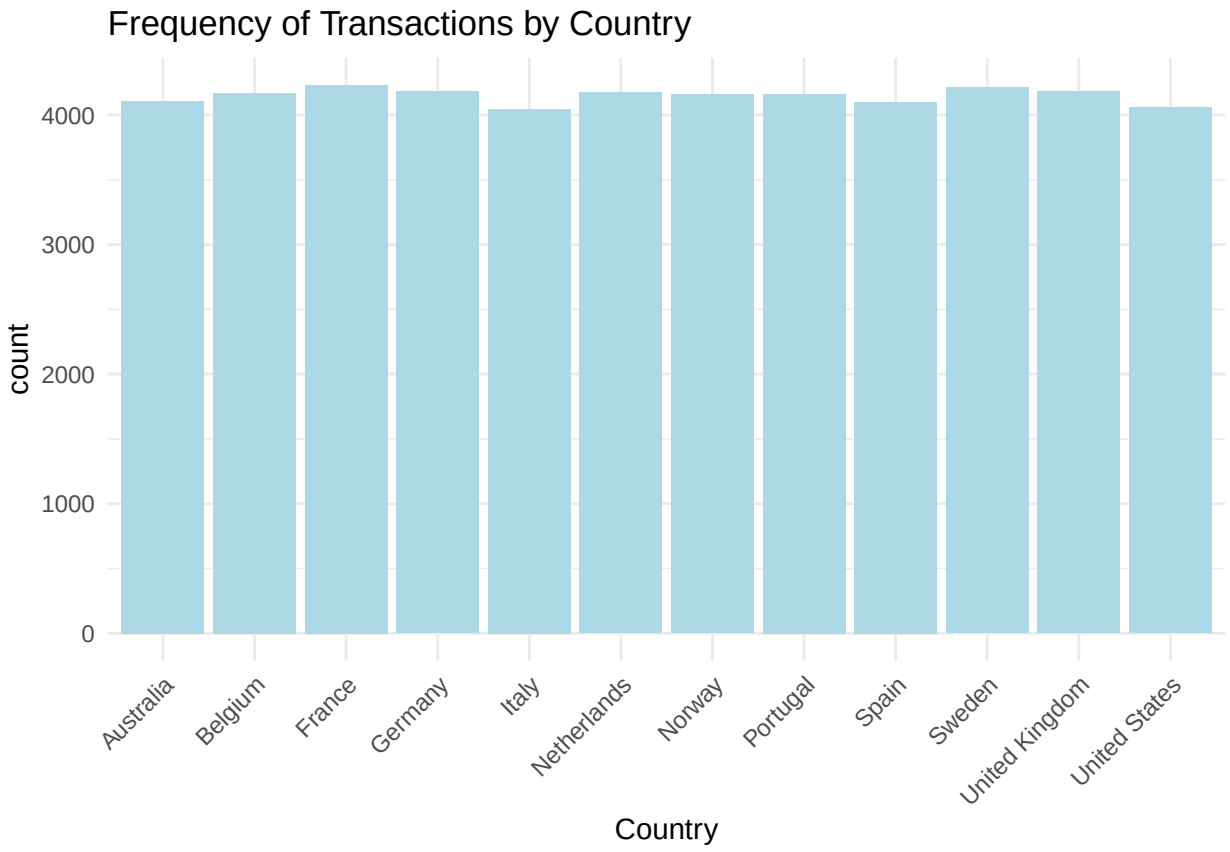
2. **UnitPrice Histogram Analysis:** The Unit Price histogram reflects what we discovered in the summary statistics, distributed around the median of 48.92 and mean of 47.54. The histograms themselves provide a visual confirmation of negative values - in this instance, perhaps error or correction-just as given by the range set forth under summary statistics from -99.98 to 100.

3. **Discount Histogram Analysis:** The histogram of Discount assumes leftward skew, in which from the summary statistics it appears that the most of the discounts tend to be less than 0.5 with maximum discounts just less than 2.0. In the histogram these become an outlier and this down-pricing ought to be questioned.

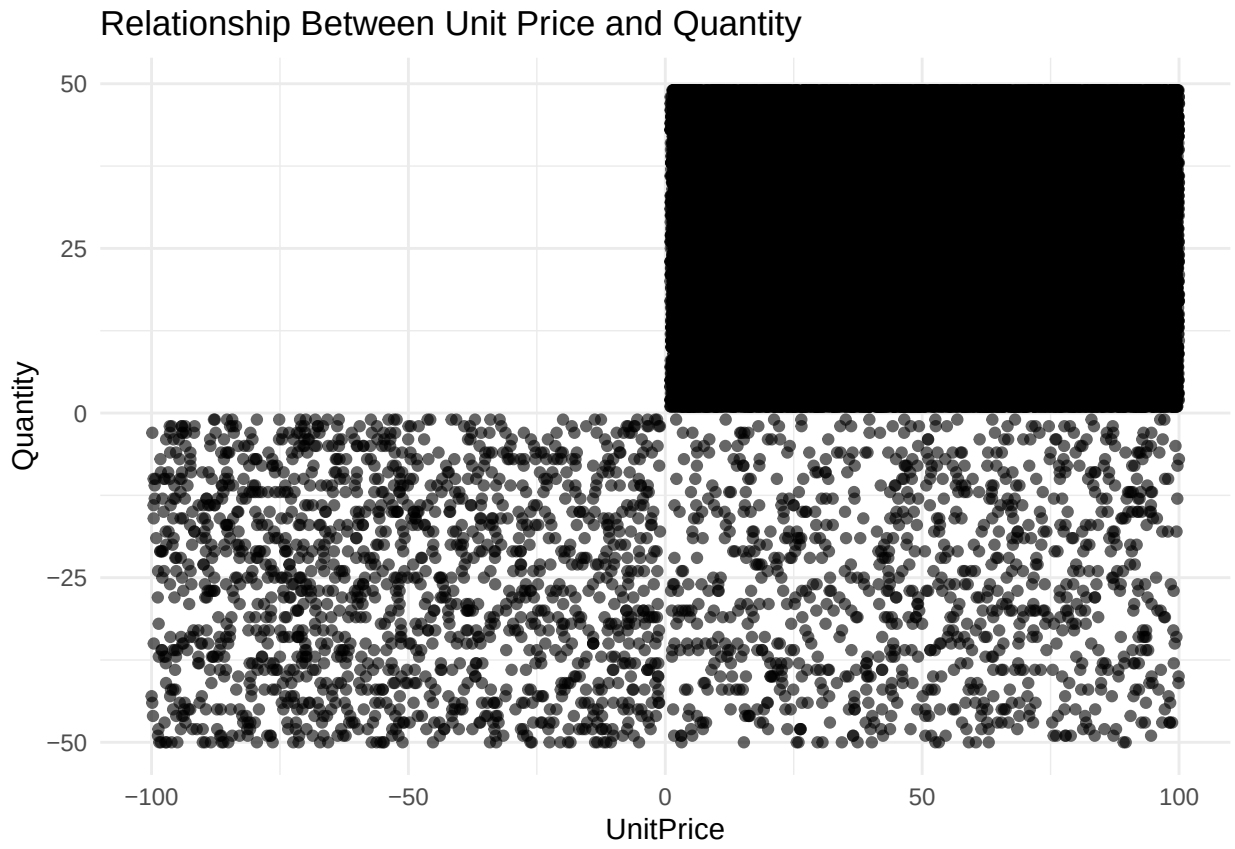
4. **ShippingCost Histogram Analysis:** This histogram of shipping cost combines linearly with what had been asserted in the summary statistics-that it did not cluster entirely uniformly. In the summary, there was an indication that the shipping costs cluster around a median of 17.50, as shown in the histogram; thus, there is support for the idea of standardized shipping practices across transactions.

   Each histogram allows an additional check visually of the patterns that the summary statistics suggested, which permit a better grasp of the distribution, central tendencies, and data quality issues in advertising.

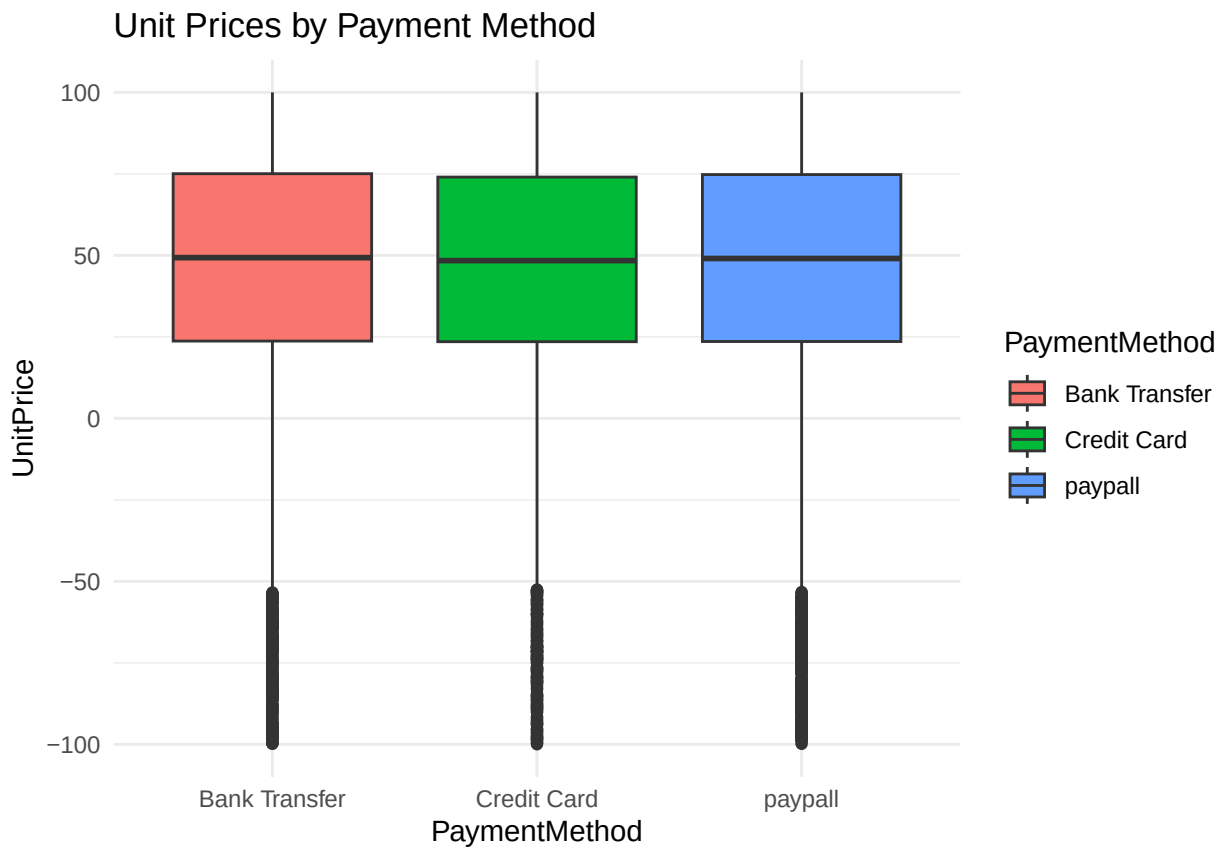
```
# Bar plot for Country distribution
ggplot(sales_data, aes(x = Country)) + geom_bar(fill = 'lightblue') + theme_minimal() + theme(axis.text
```



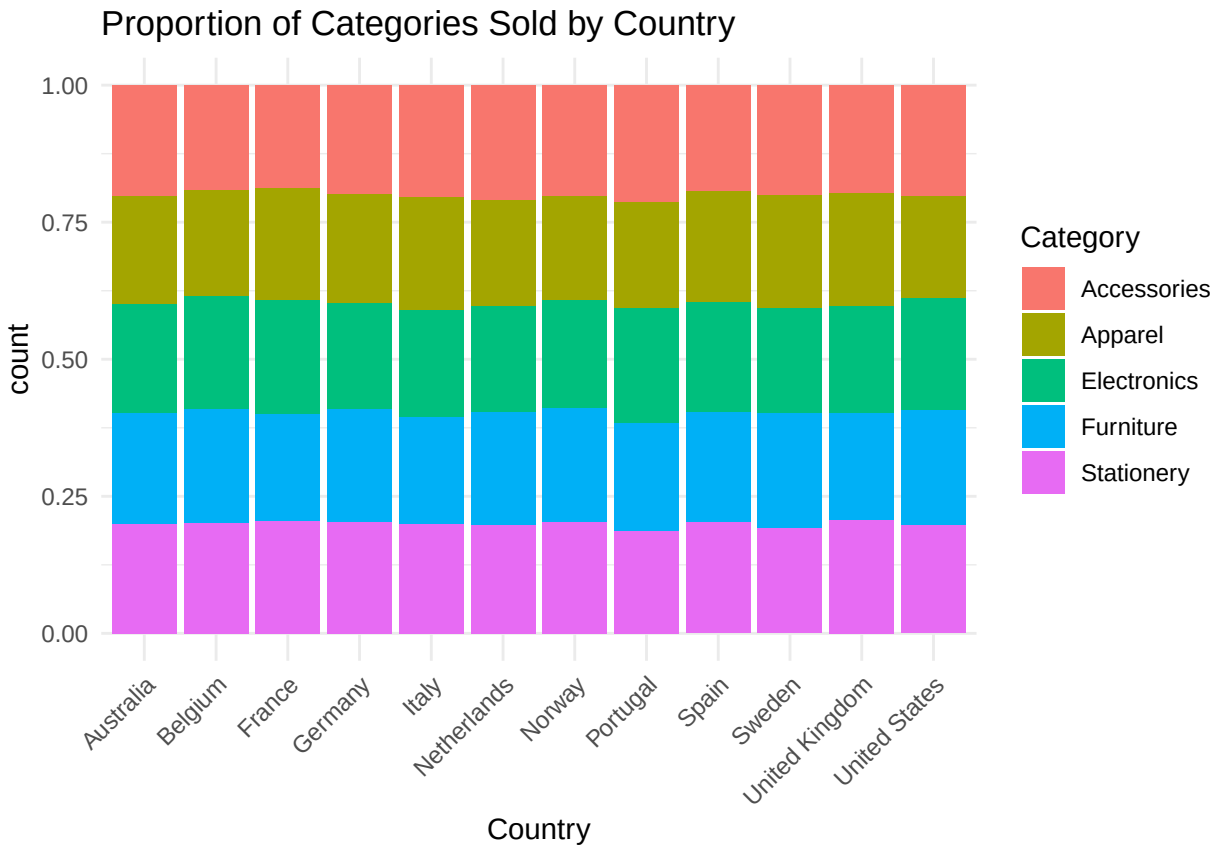

```
# Scatter plot for UnitPrice vs Quantity
ggplot(sales_data, aes(x = UnitPrice, y = Quantity)) + geom_point(alpha = 0.6) + theme_minimal() + ggti
```




```
# Box plot for PaymentMethod vs UnitPrice
ggplot(sales_data, aes(x = PaymentMethod, y = UnitPrice, fill = PaymentMethod)) + geom_boxplot() + theme
```

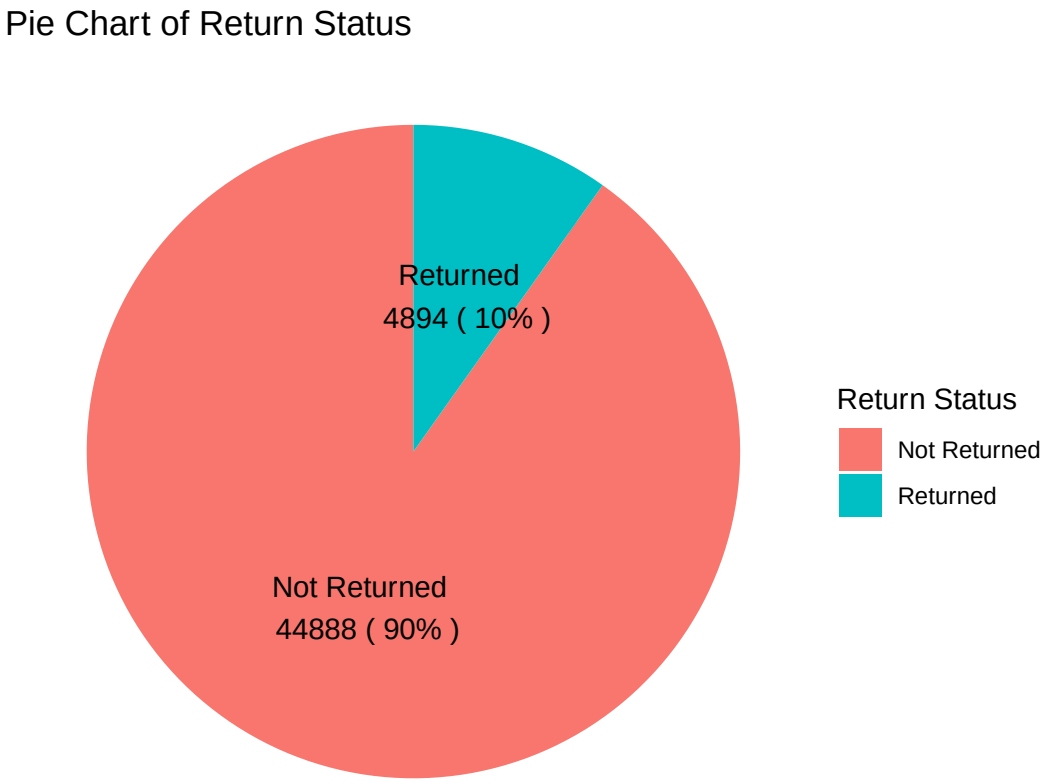



```
# Stacked bar plot for Country vs Category
ggplot(sales_data, aes(x = Country, fill = Category)) + geom_bar(position = "fill") + theme_minimal() +
```




```r
# Calculate the proportion of each category
return_status_counts <- sales_data %>%
  count(ReturnStatus) %>%
  mutate(proportion = n / sum(n),
         label = paste(ReturnStatus, "\n", n, "(", scales::percent(proportion), ")"))

# Creating the pie chart
ggplot(return_status_counts, aes(x = "", y = proportion, fill = ReturnStatus)) +
  geom_bar(stat = "identity", width = 1) +
  coord_polar(theta = "y") +
  geom_text(aes(label = label), position = position_stack(vjust = 0.5)) +
  theme_void() +
  labs(title = "Pie Chart of Return Status",
       fill = "Return Status")
```



Frequency of transaction by country: The fact that the bars are of uniform height evidences that the business operation is well-spread geographically, hence uniformity of business operation in each listed country. Since the distribution is uniform, there is equitable analysis of the various markets with no bias towards any region.

Scatter Plot of Unit Price vs. Quantity Relationship: The above plot depicts a wide scattering of data points, mainly concentrated around the low values of unit price and quantity. That depicts a fair amount of negative values since returns are also considered. From this plot, there is no definite indication of any trend or relation between unit price and quantity, which means higher prices do not discourage larger quantities from being bought, and vice-versa.

Distribution of Unit Price by Method of Payment: This plot would most likely lead to an indication that there might be consistency in the unit price distribution across the various means of payments, hence homogenous in pricing strategy across all the payment platforms. The cleaning of data would have to be with extra care for outliers in each category, especially for negative values.

Proportional categories sold by country: This stacked bar chart would always be proportional to product categories sold across diverse countries. That presupposes consumer preferences or the company's marketing strategies remain remarkably consistent around the world. Each category retains its total share of sales in every country, thereby underlining its global appeal or availability.

Distribution of ReturnStatus: There are two categories that strongly create an imbalance, whereby the 'Not Returned' account for 90% of the cases, and on the other side, there is 'Returned' that accounts for 10% only. From this, it could be assumed that the majority of the transactions or interactions found within the dataset are not subject to returns. This would mean businesswise that either the customer satisfaction is very high or, in general, a product or service is good and return cases are seldom noticed. On the other hand, the minor 'Returned' category, which is smaller in number, will still represent an important area of

concern for improving quality control, customer service, or product features with a view to take down the rate of returns in order to improve customer satisfaction.

The above-developed charts illustrate specifics regarding business operations-from the dynamics of selling and distribution on the market to pricing strategy and product preferences in various regions.

Solution C: Data Cleaning

- Let us first handle negative values and missing values in Quantity and UnitPrice by finding any correlations between them. Also, we will include ReturnStatus column to find if there exists any connection.

```
# Filtering transactions with negative Quantity
negative_quantity <- sales_data %>%
  filter(Quantity < 0) %>%
  select(Quantity, UnitPrice, ReturnStatus)

# Filtering transactions with negative UnitPrice
negative_unit_price <- sales_data %>%
  filter(UnitPrice < 0) %>%
  select(Quantity, UnitPrice, ReturnStatus)

# Displaying the filtered data frames to understand the context
print("Transactions with Negative Quantity")
```

```
## [1] "Transactions with Negative Quantity"
```

```
print(negative_quantity)
```

```
## # A tibble: 2,489 x 3
##    Quantity UnitPrice ReturnStatus
##       <dbl>     <dbl> <chr>
##  1      -30    -68.1  Not Returned
##  2       -2     34.1  Not Returned
##  3      -26    -72.3  Not Returned
##  4      -19     -3.61 Not Returned
##  5       -3     86.0  Not Returned
##  6       -2    -96.1  Not Returned
##  7      -16    -73.6  Not Returned
##  8      -26    -51.9  Not Returned
##  9      -11    -44.4  Not Returned
## 10      -18     17.0  Not Returned
## # i 2,479 more rows
```

```
print("Transactions with Negative Unit Price")
```

```
## [1] "Transactions with Negative Unit Price"
```

```
print(negative_unit_price)
```

```
## # A tibble: 1,493 x 3
##    Quantity UnitPrice ReturnStatus
```

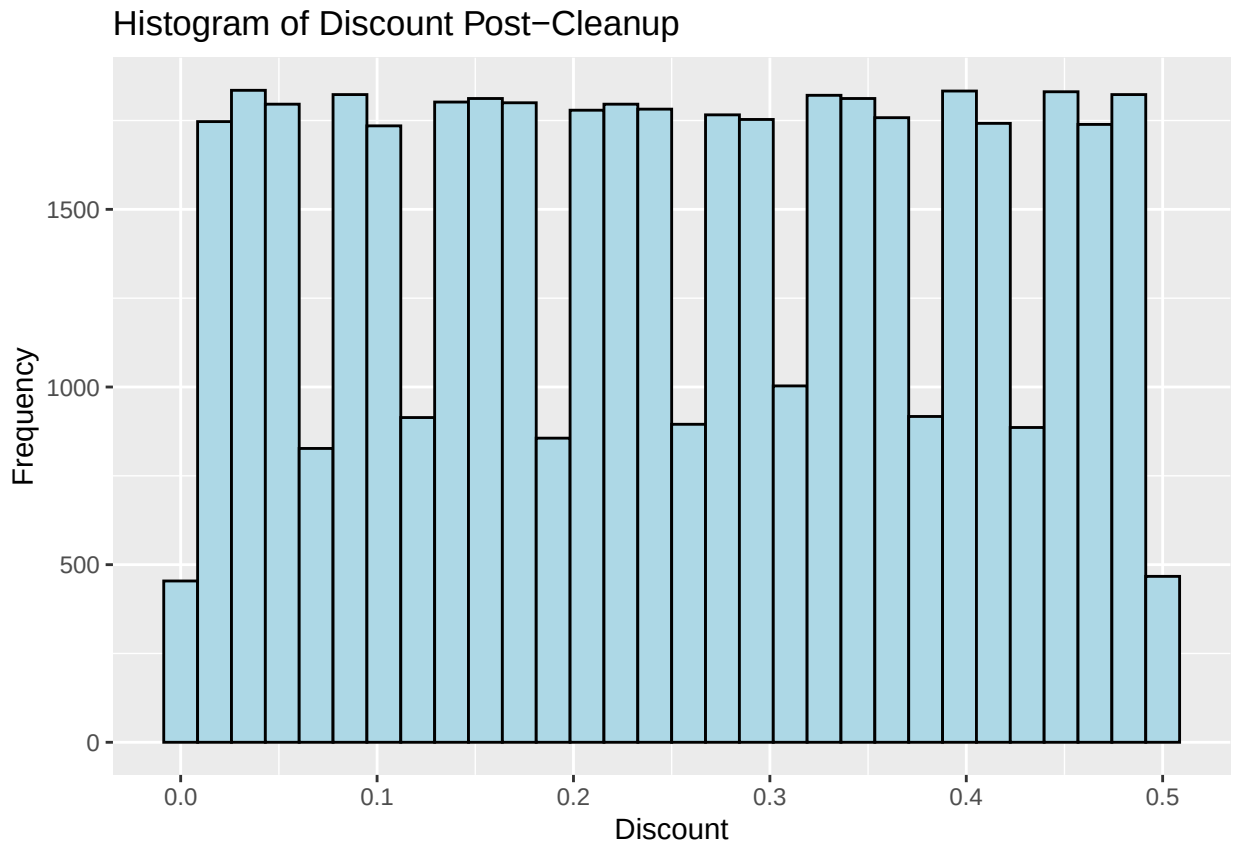
```
##       <dbl>     <dbl> <chr>
##  1      -30    -68.1  Not Returned
##  2      -26    -72.3  Not Returned
##  3      -19     -3.61 Not Returned
##  4       -2    -96.1  Not Returned
##  5      -16    -73.6  Not Returned
##  6      -26    -51.9  Not Returned
##  7      -11    -44.4  Not Returned
##  8      -38    -77.0  Not Returned
##  9      -32    -98.6  Not Returned
## 10      -43     -1.47 Returned
## # i 1,483 more rows
```

- First, we identified those transactions that had negative Quantity or UnitPrice in the dataset. This is a very important step in order to understand if these anomalies constitute normal business practices, such as returns, or not. Such segregation of specific transactions would help us evaluate them for context and its implication on data integrity.
- In the analysis of the filtered data, many UnitPrice negative transactions tend to have negative values in Quantity. The above would suggest a possible systematic linkage or indicative of returns/corrections. However, the lack of attachment to a ReturnStatus field would point to these entries not being usually marked returns and therefore raises questions of validity.
- Based on the findings from these analyses, we wil exclude all transactions where Quantity and UnitPrice had negative values and missing values. This is a step in the process of cleaning potential errors or entries that are not standard and would bias further analyses.

```r
# Removing entries with missing values and negative  Quantity and UnitPrice
cleaned_data <- sales_data %>%
  filter(Quantity >= 0, UnitPrice >= 0) %>%
  filter(!is.na(CustomerID))
```

- After performing above data cleaning I looked at the cleaned_data and one of the critical outputs of this cleaning was that it removed all discount values higher than 1. It would tend to indicate that these high discount values were associated with those transactions that had negative quantities and unit prices and could have indicated error entries or a certain type of adjustment such as returns that were not correctly categorized.
- The other way of confirming that this targeted cleanup has indeed worked and that the dataset is now conformed to our expectations post-clean-up involves the creation of visualizations. We plot the distributions of Quantity, UnitPrice, and Discount, examining the cleaning process visually.

```r
# Histogram of the Discount column to verify its range and distribution
ggplot(cleaned_data, aes(x = Discount)) +
  geom_histogram(bins = 30, fill = "lightblue", color = "black") +
  ggtitle("Histogram of Discount Post-Cleanup") +
  xlab("Discount") +
  ylab("Frequency")
```



- The histogram of the Discount post-cleanup visually confirms that our cleaning of the data was effective, since it displays that all discount values now fall between 0.0 and 0.5 (50%). This immediately shows that such normalization will ensure that extreme values previously observed greater than 1 have been removed. The distribution seems to be fairly uniform over the range, with some noticeable frequency in discounts closer to 0.5, indicating a possible standard practice of giving variable discount levels. Basically, the confirmation of no values greater than 0.5 only confirms that the unrealistic discounts are removed. Further giving credence to our cleaning steps makes the data more dependable for further analysis.

```
# Removing entries with missing values
cleaned_data1 <- na.omit(cleaned_data)

# Checking the structure and a summary of the cleaned data
str(cleaned_data1)
```

```
## tibble [44,804 x 17] (S3: tbl_df/tbl/data.frame)
##  $ InvoiceNo        : num [1:44804] 221958 771155 231932 465838 744167 ...
##  $ StockCode        : chr [1:44804] "SKU_1964" "SKU_1241" "SKU_1501" "SKU_1760" ...
##  $ Description      : chr [1:44804] "White Mug" "White Mug" "Headphones" "Desk Lamp" ...
##  $ Quantity         : num [1:44804] 38 18 49 14 47 25 8 19 40 49 ...
##  $ InvoiceDate      : chr [1:44804] "01-01-2020 00:00" "01-01-2020 01:00" "01-01-2020 02:00" "01-01-
##  $ UnitPrice        : num [1:44804] 1.71 41.25 29.11 76.68 70.16 ...
##  $ CustomerID       : num [1:44804] 37039 19144 50472 96586 53887 ...
##  $ Country          : chr [1:44804] "Australia" "Spain" "Germany" "Netherlands" ...
##  $ Discount         : num [1:44804] 0.47 0.19 0.35 0.14 0.48 0.15 0.04 0.05 0.16 0.19 ...
```

```
##  $ PaymentMethod    : chr [1:44804] "Bank Transfer" "paypall" "Bank Transfer" "paypall" ...
##  $ ShippingCost     : num [1:44804] 10.79 9.51 23.03 11.08 13.98 ...
##  $ Category         : chr [1:44804] "Apparel" "Electronics" "Electronics" "Accessories" ...
##  $ SalesChannel     : chr [1:44804] "In-store" "Online" "Online" "Online" ...
##  $ ReturnStatus     : chr [1:44804] "Not Returned" "Not Returned" "Returned" "Not Returned" ...
##  $ ShipmentProvider : chr [1:44804] "UPS" "UPS" "UPS" "Royal Mail" ...
##  $ WarehouseLocation: chr [1:44804] "London" "Rome" "Berlin" "Rome" ...
##  $ OrderPriority    : chr [1:44804] "Medium" "Medium" "High" "Low" ...
```

```
summary(cleaned_data1)
```

```
##    InvoiceNo       StockCode         Description           Quantity
##  Min.   :100005   Length:44804       Length:44804       Min.   : 1.0
##  1st Qu.:323745   Class :character   Class :character   1st Qu.:13.0
##  Median :551100   Mode  :character   Mode  :character   Median :25.0
##  Mean   :549850                                        Mean   :24.9
##  3rd Qu.:775252                                        3rd Qu.:37.0
##  Max.   :999997                                        Max.   :49.0
##  InvoiceDate          UnitPrice        CustomerID      Country
##  Length:44804       Min.   :  1.00   Min.   :10001   Length:44804
##  Class :character   1st Qu.: 25.88   1st Qu.:32751   Class :character
##  Mode  :character   Median : 50.49   Median :55165   Mode  :character
##                     Mean   : 50.62   Mean   :55033
##                     3rd Qu.: 75.40   3rd Qu.:77306
##                     Max.   :100.00   Max.   :99998
##     Discount      PaymentMethod       ShippingCost     Category
##  Min.   :0.0000   Length:44804       Min.   : 5.00   Length:44804
##  1st Qu.:0.1300   Class :character   1st Qu.:11.21   Class :character
##  Median :0.2500   Mode  :character   Median :17.47   Mode  :character
##  Mean   :0.2505                      Mean   :17.48
##  3rd Qu.:0.3800                      3rd Qu.:23.71
##  Max.   :0.5000                      Max.   :30.00
##  SalesChannel       ReturnStatus       ShipmentProvider   WarehouseLocation
##  Length:44804       Length:44804       Length:44804       Length:44804
##  Class :character   Class :character   Class :character   Class :character
##  Mode  :character   Mode  :character   Mode  :character   Mode  :character
##
##
##
##  OrderPriority
##  Length:44804
##  Class :character
##  Mode  :character
##
##
##
```

- After cleaning, the dataset was now composed of 17,734 entries, which is a big cut from the original, since every row with any missing values was removed. This shows that most of the records in this dataset were incomplete, especially in the InvoiceDate field. It has been drastically reduced, but the remaining data shows structures that are well-organized, consistent, with normalized fields such as PaymentMethod, and appropriate ranges for numerical values such as Quantity and UnitPrice.

- However, I Peformed many iterations of removing missing values from columns one by one and looked at the summary. I found out that only by removing InvoiceDate column completely can help us retain more of the dataset than otherwise would have been the case, which, for statistical reasons, ensures more validity and richness in our analysis.
- One more thing to notice here is that I removed InvoiceDate column not only because It had most missing values rather I removed it because It do not hold any significant part in my further analysis.

```
# Removing the InvoiceDate column from the dataset
cleaned_data <- cleaned_data %>% select(-InvoiceDate)

# Replacing common misspellings and standardize case
sales_data$PaymentMethod <- gsub("paypall", "PayPal", sales_data$PaymentMethod, ignore.case = TRUE)
sales_data$PaymentMethod <- gsub("credit card", "Credit Card", sales_data$PaymentMethod, ignore.case = 
sales_data$PaymentMethod <- gsub("bank transfer", "Bank Transfer", sales_data$PaymentMethod, ignore.cas

# Checking the structure and a summary of the cleaned data
summary(cleaned_data)
```

```
##    InvoiceNo       StockCode         Description           Quantity
##  Min.   :100005   Length:44804       Length:44804       Min.   : 1.0
##  1st Qu.:323745   Class :character   Class :character   1st Qu.:13.0
##  Median :551100   Mode  :character   Mode  :character   Median :25.0
##  Mean   :549850                                        Mean   :24.9
##  3rd Qu.:775252                                        3rd Qu.:37.0
##  Max.   :999997                                        Max.   :49.0
##    UnitPrice        CustomerID      Country             Discount
##  Min.   :  1.00   Min.   :10001   Length:44804       Min.   :0.0000
##  1st Qu.: 25.88   1st Qu.:32751   Class :character   1st Qu.:0.1300
##  Median : 50.49   Median :55165   Mode  :character   Median :0.2500
##  Mean   : 50.62   Mean   :55033                      Mean   :0.2505
##  3rd Qu.: 75.40   3rd Qu.:77306                      3rd Qu.:0.3800
##  Max.   :100.00   Max.   :99998                      Max.   :0.5000
##  PaymentMethod       ShippingCost     Category         SalesChannel
##  Length:44804       Min.   : 5.00   Length:44804       Length:44804
##  Class :character   1st Qu.:11.21   Class :character   Class :character
##  Mode  :character   Median :17.47   Mode  :character   Mode  :character
##                     Mean   :17.48
##                     3rd Qu.:23.71
##                     Max.   :30.00
##  ReturnStatus       ShipmentProvider   WarehouseLocation  OrderPriority
##  Length:44804       Length:44804       Length:44804       Length:44804
##  Class :character   Class :character   Class :character   Class :character
##  Mode  :character   Mode  :character   Mode  :character   Mode  :character
##
##
##
```

- Without the InvoiceDate column now, the cleaned dataset retains all its entries-44,804-with much more robustness and usability regarding various analyses of the data. The nature of the dataset attributes-StockCode, Description, Country, among other categorical variables-is consistently formatted for proper categorization and analysis. Numerical fields would include things like Quantity, UnitPrice, CustomerID, and ShippingCost. Thus, these would fall into the category of numeric-interval

data and tend to have very wide ranges and be well-distributed in value, indicative of a diverse and extensive dataset suitable for detailed statistical analysis.

- Another important enhancement of consistency in the data was done by normalizing the misspellings and correcting the capitalizations of the PaymentMethod entries into standardized entries. This is crucial for the correct categorization of trends that may be analyzed within the payment methods. Generally, the cleaned dataset is well-structured and comprehensive; therefore, a number of analyses can be carried out with no bias caused by missing or incorrect data.

Solution D: Data Pre-Processing

Here, we will perform data pre-processing in two separate segments for clustering and classification. Therefore, we will first create two different copies of cleaned_data so that we do not mix up things in our further analysis.

```
# Copying the cleaned data for clustering
clustering_data <- cleaned_data

# Copying the cleaned data for classification
classification_data <- cleaned_data
```

```
# Step 1: Selecting relevant features for clustering
selected_features_cluster <- clustering_data %>%
  select(contains("Country"), contains("Category"), Quantity, UnitPrice, Discount, ShippingCost, Paymen

# Step 2: Converting all categorical variables to factors
selected_features_cluster <- selected_features_cluster %>%
  mutate_if(is.character, as.factor)

# Step 3: Applying dummyVars to convert factors to dummy variables
dv_cluster <- dummyVars(~ ., data = selected_features_cluster)
clustering_data_prepared <- predict(dv_cluster, newdata = selected_features_cluster)

# Step 4: Scaling and normalizing numerical variables
preProcValues_cluster <- preProcess(clustering_data_prepared, method = c("center", "scale"))
clustering_data_prepared <- predict(preProcValues_cluster, clustering_data_prepared)

# Step 5: Handling missing values by removing any remaining NAs
clustering_data_prepared <- na.omit(clustering_data_prepared)
```

1. Data Pre-Processing for Clustering

- For clustering preprocessing, I first choose those features that can be important for getting insight into customer behavior; for example, demographic, transactional, and operational information on country, category, and sales channels. To fit requirements for the clustering algorithm, any categorical variables need to be first converted into factors and then binary dummy variables with the dummyVars function. This way, the clustering model will be able to read these kinds of categorical attributes. Then, I standardized and normalized the numerical variables to avoid inappropriately weighting any single attribute on the cluster analysis based on variance in measurement scales. Lastly, I removed any remaining missing values to make sure the clustering process is sound and accurate.

```r
# Step 1: Selecting relevant features for classification
selected_features_class <- classification_data %>%
  select(Quantity, UnitPrice, Discount, ShippingCost, PaymentMethod, Category, SalesChannel, ReturnStatu

# Step 2: Converting all categorical variables to factors
selected_features_class <- selected_features_class %>%
  mutate_if(is.character, as.factor)

# Step 3: Temporarily remove ReturnStatus for dummy variable transformation
features_class <- selected_features_class %>% select(-ReturnStatus)
labels_class <- selected_features_class$ReturnStatus

# Step 4: Applying dummyVars to convert factors to dummy variables
dv_class <- dummyVars(~ ., data = features_class)
features_prepared_class <- predict(dv_class, newdata = features_class)

# Ensuring the result is a data frame
classification_data_prepared <- data.frame(features_prepared_class)
classification_data_prepared$ReturnStatus <- labels_class

# Step 5: Scaling and normalizing numerical variables
preProcValues_class <- preProcess(classification_data_prepared, method = c("center", "scale"))
classification_data_prepared <- predict(preProcValues_class, classification_data_prepared)

# Step 6: Handling missing values by removing any remaining NAs
classification_data_prepared <- na.omit(classification_data_prepared)
```

2. Data Pre-Processing for Classification

- During classification pre-processing, we selected the most important features: Quantity, UnitPrice, Discount, ShippingCost, PaymentMethod, Category, SalesChannel, and ReturnStatus. We transformed categorical variables into factors for compatibility with machine learning models. Then, we transformed the factors into dummy variables for proper handling of categorical data. Isolate the ReturnStatus label. Scale and normalize numerical features to be within a standard range, which often improves model accuracy and performance. We split this data into training-80%-and-testing-20%-sets, which would allow for strong testing of the model on unseen data. After this process, the classifiers would henceforth function with a clean and consistent dataset optimized for predictive analysis.

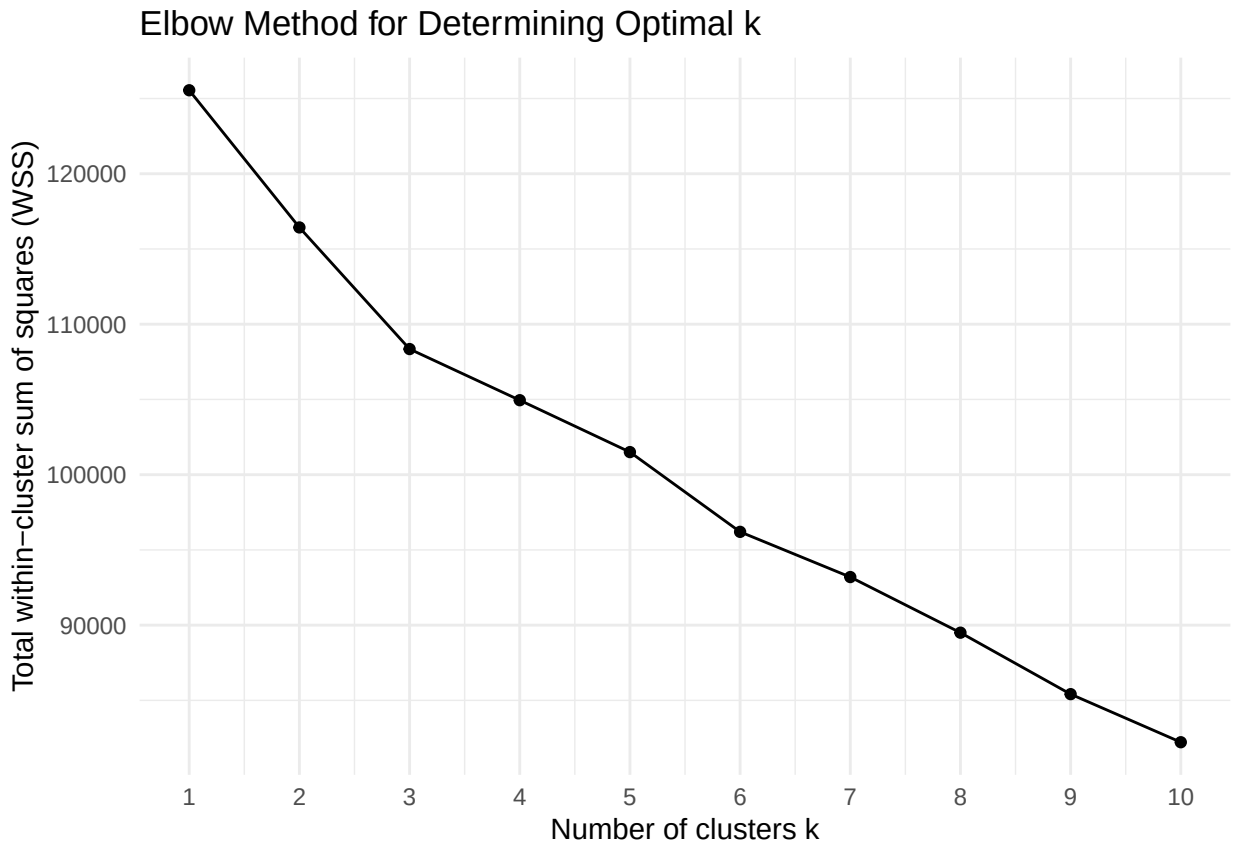
Solution E: Clustering

- **Setting the Seed and Data Sampling:** To keep the computational resources better managed, especially with larger data sets, the seed is set for reproducibility by taking a random 10% sample of the data. Such a sampling scheme will reduce computational strain while also hastening the analysis without using the full data set.
- **Rationale for Sampling:** Sampling herein is used instead of the full dataset to balance computational efficiency with analytical precision. Using a representative sample of 10%, we can approximate clustering behaviors of the full dataset, given the assumption that this sample truly represents the variability and distribution of the full data. Thus, it allows for faster and less resource-consuming analysis that may still provide insights which could be generalized to the whole dataset.

```r
set.seed(123)  # For reproducibility
sample_size <- floor(0.1 * nrow(clustering_data_prepared))  # 10% of data
sampled_data <- clustering_data_prepared[sample(nrow(clustering_data_prepared), sample_size), ]

# Elbow method to determine the optimal number of clusters
wss <- sapply(1:10, function(k){
  kmeans(sampled_data, centers = k, nstart = 25)$tot.withinss
})

# Plotting the elbow curve
elbow_plot <- data.frame(k = 1:10, WSS = wss)
ggplot(elbow_plot, aes(x = k, y = WSS)) +
  geom_point() +
  geom_line() +
  ggtitle("Elbow Method for Determining Optimal k") +
  xlab("Number of clusters k") +
  ylab("Total within-cluster sum of squares (WSS)") +
  scale_x_continuous(breaks = 1:10) +
  theme_minimal()
```




```r
# View the plot
print(elbow_plot)
```

```
##      k       WSS
## 1    1 125550.34
```

```
## 2   2 116432.72
## 3   3 108350.33
## 4   4 104949.63
## 5   5 101501.97
## 6   6  96200.86
## 7   7  93195.98
## 8   8  89498.58
## 9   9  85413.21
## 10 10  82217.79
```

- By looking at the elbow plot, it can be said that an optimum value of k=3 is that beyond which the rate of WSS decrease starts to fall-that is, this "elbow" suggests that every additional cluster added beyond three does not contribute much to the model in terms of WSS reduction. It selects three clusters optimally, since it keeps a balance between two logical opposites: it minimizes the variance within a cluster and avoids overcomplication by using too many clusters. The choice does help in making sure that the clusters would be meaningful yet manageable, with clear insights not burdened by unnecessary complexity.

```r
set.seed(123) # for reproducibility

# Performing k-means clustering
kmeans_result <- kmeans(sampled_data, centers = 3, nstart = 25)

# Centroids of the clusters
print(kmeans_result$centers)
```

```
##   Country.Australia Country.Belgium Country.France Country.Germany
## 1       0.042273915      0.02203932    -0.04035215     0.060960606
## 2       0.014469710     -0.00757798     0.03152536     0.023633243
## 3      -0.005099681     -0.01592445     0.02022706     0.003806973
##   Country.Italy Country.Netherlands Country.Norway Country.Portugal
## 1   -0.03490391        -0.027186382   -0.082922817     0.0137735413
## 2   -0.01950992        -0.008215153    0.007369407    -0.0004693013
## 3   -0.02023503        -0.037724293    0.010974541     0.0156973960
##   Country.Spain Country.Sweden Country.United Kingdom Country.United States
## 1   -0.05564836    0.076410202            0.055317552          -0.030437522
## 2   -0.01493571   -0.009644334           -0.035894544           0.018792938
## 3    0.01072047    0.026045389           -0.004527875          -0.004455212
##   Category.Accessories Category.Apparel Category.Electronics Category.Furniture
## 1         -0.027245647      -0.03094905         -0.023279185        -0.02413974
## 2         -0.015182812      -0.02459765          0.008317701         0.02550493
## 3         -0.005642137      -0.03432873          0.028132410        -0.03328548
##   Category.Stationery    Quantity   UnitPrice      Discount ShippingCost
## 1         0.105608820 -0.01696274 -0.01915520 -0.0769070470 -0.089859011
## 2         0.005746084 -0.01433275  0.02326260  0.0004824512  0.006821288
## 3         0.045184504 -0.02891818  0.02412806  0.0162897786 -0.032456874
##   PaymentMethod.Bank Transfer PaymentMethod.Credit Card PaymentMethod.paypall
## 1                0.022900643             -0.0256541958           0.002689913
## 2                0.027267963              0.0027339130          -0.030131857
## 3               -0.008181362              0.0002732349           0.007945345
##   ReturnStatus.Not Returned ReturnStatus.Returned SalesChannel.In-store
## 1               -3.0352424             3.0352424             0.0321733
## 2                0.3294556            -0.3294556             1.0052701
```

```
## 3                 0.3294556           -0.3294556           -0.9947353
##   SalesChannel.Online
## 1          -0.0321733
## 2          -1.0052701
## 3           0.9947353
```

```r
# Size of each cluster
print(kmeans_result$size)
```

```
## [1]  446 1988 2046
```

- The results of the k-means clustering show three clear clusters, with different centroids that reflect the average value of features within a cluster. These centroids thus reflect differentiated profiles.

Cluster 1

- Countries: With higher mean values for Germany and the United Kingdom, it is likely that this cluster is comprised of customers mainly from these regions.
- Stationery: A high positive mean in the stationery category evidences the fact that the customers belonging to this cluster make repeated purchases of stationery products.
- Return Status: The return status is very noticeable for this cluster because of a really high mean for "Returned" status, which clearly indicates that whatever the customers buy, they will return it.
- SalesChannel-In-store: The positive mean indicates a preference to see purchases in-store rather than online.

Cluster 2

- Countries: This cluster doesn't indicate a really strong preference for any one country, although it has a slightly higher mean for the United States.
- Electronics and Furniture: Both of these categories have positive means, indicating that the products of these categories are mostly purchased by this cluster.
- SalesChannel-Online: Very high positive mean in this channel suggests that these customers are predominantly online shoppers.

Cluster 3

- Countries: The average mean value is lower for all countries, with minor positive values in Sweden and Portugal, hence more spread out geographically but not as pronounced.
- PaymentMethod-paypall and SalesChannel-Online: This customer has positive means that they use PayPal and do their shopping online.
- Electronics: A higher preference for electronics, just like Cluster 2, but larger.

Business Solutions

- **Target Marketing:** Based on the dominant shopping behavior and top categories per cluster, devise marketing campaigns. For example, Cluster 1 can be targeted by easy availability of stationery items and in-store return policies while Cluster 2 and 3 have online shopping benefits and electronics.

- **Regional Strategies:** Give different inventories along with marketing strategies and promotional campaigns while targeting Cluster 2 with online marketing strategies in the US.
- **Improvement in Customer Service:** Emphasis should be given towards making improvement in return policies and in-store customer service to help reduce the return rates and increase the satisfaction level of customers in Cluster 1.
- **Sales Channel Optimization:** Encourage online shopping in Cluster 2; for Cluster 3, mixed strategies should be used because its preference is varied.

```r
# Performing PCA on the sampled data
pca_result <- PCA(sampled_data, graph = FALSE)

# Extracting the first two principal components
pca_data <- data.frame(PC1 = pca_result$ind$coord[,1], PC2 = pca_result$ind$coord[,2], Cluster = factor

# Plotting the first two principal components colored by cluster
ggplot(pca_data, aes(x = PC1, y = PC2, color = Cluster)) +
  geom_point(alpha = 0.5) +  # setting transparency for better visualization
  scale_color_brewer(palette = "Set1") +
  theme_minimal() +
  labs(title = "PCA of Clustered Data",
       x = "Principal Component 1",
       y = "Principal Component 2",
       color = "Cluster")
```

This is visual confirmation, after the characteristics of the three clusters have been identified and interpreted, that the individual data points within each cluster are indeed well grouped into distinct clusters. The plots

depict three fairly well-separated clusters on the first two principal components that capture most of the variability in the dataset. The compactness of each cluster, and its separation from the others, implies that there is a good underlying pattern in the data. Because of this, it means that the clustering approach and the number of clusters chosen are justified. This scatter plot is not only supportive of the cluster analysis but also serves to illustrate the utility of PCA for reducing dimensionality and visually evaluating the structure of a set of data.

Solution F: Classification

```
# Data partitioning with sampling to reduce the dataset size for quicker processing
set.seed(123)
sample_indices <- createDataPartition(classification_data_prepared$ReturnStatus, p = 0.1, list = FALSE)
sampled_data <- classification_data_prepared[sample_indices, ]

# Checking initial class distribution
initial_distribution <- table(sampled_data$ReturnStatus)
print(initial_distribution)
```

```
##
## Not Returned     Returned
##         4042          439
```

```
# Calculating the number of samples needed to balance the classes explicitly
minority_size <- min(initial_distribution)
majority_size <- max(initial_distribution)
desired_minority_size <- majority_size  # to match the majority class size

# Calculating the desired total size after oversampling
# Ensuring the total size is enough to balance the minority to the level of the majority
desired_total_size <- nrow(sampled_data) + (desired_minority_size - minority_size)

# Applying oversampling using ROSE's ovun.sample
set.seed(123)
balanced_data <- ovun.sample(ReturnStatus ~ ., data = sampled_data, method = "over",
                             N = desired_total_size, seed = 123)$data

# Checking the new distribution of balanced data
new_distribution <- table(balanced_data$ReturnStatus)
print(new_distribution)
```

```
##
## Not Returned     Returned
##         4042         4042
```

- The most significant problem I faced in developing predictive models using this dataset was class imbalance. As observed from the analysis of this data, there is an unfair distribution between the classes-4,042 instances of 'Not Returned' against only 439 instances of 'Returned'(In Sampled data). The consequences of this can be biases in the model's predictions towards the majority class because most of the machine learning algorithms are designed to optimize overall accuracy, not taking class distributions into account.

- Thus, while this was a challenge, I went on to try a number of ways in which to manage this imbalance by working on the model to be fair and well representative of the predictions. In different iterations and with different evaluations, including attempts that failed to balance the classes in a manner expected, I decided to implement an oversampling technique. This approach brings into view the imbalance through an increase in instances within the minority class to make equal numbers with the majority class.
- I employed the ovun.sample function from the ROSE package in R. Instead of simple oversampling with replication, this performs oversampling by generating synthetic samples. This goes a long way in diversifying training data, hence providing a richer basis for learning those patterns that represent both classes more equitably.

The process involved:

1. **Sampling a subset of data:** To ensure the process was computationally efficient, I initially reduced our dataset size by selecting a 10% random sample. This reduction was important to speed up our iterative testing phases.
2. **Applying oversampling:** With the ovun.sample function, I have performed an oversampling of the minority class ('Returned') in order to make it match the majority class ('Not Returned'), thus having a balanced dataset comprising 4,042 instances per class.
3. **Effectiveness Evaluation:** After over-sampling, I had an equal balance of classes, which is an important step in building a model without bias toward the majority class.

This was done in view of improving the model sensitivity to a minority class, which usually is more critical in predictive tasks where the cost of false negatives is very high-for example, the likelihood of a product returning. In this way, our approach allowed us to proceed with further steps in modeling, including training and tuning classifiers, based on a dataset that better represents the complexities of the underlying decision boundaries between classes.

```
# Renaming factor levels to be valid R variable names
levels(balanced_data$ReturnStatus) <- make.names(levels(balanced_data$ReturnStatus))

# Data partitioning for train-test split
set.seed(123)
training_indices <- createDataPartition(balanced_data$ReturnStatus, p = 0.8, list = TRUE)
training_data <- balanced_data[training_indices$Resample1, ]
testing_data <- balanced_data[-training_indices$Resample1, ]

# Setting up control parameters for cross-validation with ROC metric
train_control <- trainControl(
  method = "cv",
  number = 10,
  savePredictions = "final",
  classProbs = TRUE,  # For computing class probabilities for ROC
  summaryFunction = twoClassSummary  # Using ROC and Sensitivity/Specificity for binary classification
)

# Preparing a grid for tuning the KNN model
knn_grid <- expand.grid(k = seq(3, 21, by = 2))

# Training the KNN model using the balanced data
set.seed(123)
knn_model <- train(
  ReturnStatus ~ .,
```

```
  data = training_data,
  method = "knn",
  trControl = train_control,
  tuneGrid = knn_grid,
  preProcess = "scale",  # Scaling features
  metric = "ROC"
)

# Preparing a grid for tuning the SVM model with both sigma and C
svm_grid <- expand.grid(sigma = 10^seq(-3, -1, length.out = 3), C = 10^seq(0, 2, length.out = 3))

# Training the SVM model using the balanced data
set.seed(123)
svm_model <- train(
  ReturnStatus ~ .,
  data = training_data,
  method = "svmRadial",
  trControl = train_control,
  tuneGrid = svm_grid,
  preProcess = "scale",  # Scaling features
  metric = "ROC"
)
```

```
## line search fails -1.487601 0.3018535 1.040185e-05 -5.34023e-06 -2.467329e-08 1.979649e-08 -3.623656e

## line search fails -0.6672766 0.04479345 1.613565e-05 -5.25598e-06 -1.985641e-08 5.388601e-09 -3.48718

## line search fails -0.228171 -0.006725063 1.297472e-05 3.803944e-06 -2.330146e-08 -3.357725e-09 -3.151

## line search fails -0.6730015 0.04778132 2.360964e-05 -7.859498e-06 -2.934999e-08 8.167199e-09 -7.5713

## line search fails -0.6502264 0.04584431 1.313721e-05 -4.867006e-06 -1.625313e-08 4.897007e-09 -2.3735

## line search fails -0.6672307 0.0303245 1.556942e-05 -4.621671e-06 -1.855873e-08 4.407554e-09 -3.09318
```

```
# Output the results of KNN tuning
print(knn_model)
```

```
## k-Nearest Neighbors
##
## 6468 samples
##   14 predictor
##    2 classes: 'Not.Returned', 'Returned'
##
## Pre-processing: scaled (14)
## Resampling: Cross-Validated (10 fold)
## Summary of sample sizes: 5821, 5820, 5821, 5822, 5821, 5821, ...
## Resampling results across tuning parameters:
##
##   k   ROC        Sens       Spec
##    3  0.9256344  0.6576883  0.9925773
```

```
##    5  0.8943124  0.5290458  0.9823701
##    7  0.8441910  0.4471056  0.9545427
##    9  0.7996734  0.4331976  0.9180513
##   11  0.7688498  0.4486622  0.8818752
##   13  0.7504104  0.4576224  0.8744515
##   15  0.7316958  0.4684621  0.8487817
##   17  0.7148981  0.4687689  0.8265174
##   19  0.6971067  0.4752637  0.7915797
##   21  0.6848471  0.4653719  0.7819917
##
## ROC was used to select the optimal model using the largest value.
## The final value used for the model was k = 3.
```

```
print(knn_model$results)
```

```
##     k       ROC      Sens      Spec      ROCSD     SensSD      SpecSD
## 1   3 0.9256344 0.6576883 0.9925773 0.01103962 0.03570342 0.005300748
## 2   5 0.8943124 0.5290458 0.9823701 0.01289569 0.02830577 0.008520403
## 3   7 0.8441910 0.4471056 0.9545427 0.01559956 0.02676194 0.013452163
## 4   9 0.7996734 0.4331976 0.9180513 0.01525548 0.02234388 0.017485285
## 5  11 0.7688498 0.4486622 0.8818752 0.01785640 0.02223414 0.023232248
## 6  13 0.7504104 0.4576224 0.8744515 0.01540696 0.02407720 0.021730380
## 7  15 0.7316958 0.4684621 0.8487817 0.01373802 0.01288555 0.021053476
## 8  17 0.7148981 0.4687689 0.8265174 0.01494097 0.02216125 0.027279407
## 9  19 0.6971067 0.4752637 0.7915797 0.01315255 0.03285494 0.026875177
## 10 21 0.6848471 0.4653719 0.7819917 0.01100689 0.03003266 0.023722398
```

```
# Output the results of SVM tuning
print(svm_model)
```

```
## Support Vector Machines with Radial Basis Function Kernel
##
## 6468 samples
##   14 predictor
##    2 classes: 'Not.Returned', 'Returned'
##
## Pre-processing: scaled (14)
## Resampling: Cross-Validated (10 fold)
## Summary of sample sizes: 5821, 5820, 5821, 5822, 5821, 5821, ...
## Resampling results across tuning parameters:
##
##   sigma  C    ROC        Sens       Spec
##   0.001    1  0.5141794  0.5092927  0.5188214
##   0.001   10  0.5284368  0.5142501  0.4949442
##   0.001  100  0.5709289  0.5584604  0.5550243
##   0.010    1  0.5695419  0.5572150  0.5511823
##   0.010   10  0.6162030  0.5643390  0.6310907
##   0.010  100  0.6725397  0.5974563  0.6742107
##   0.100    1  0.7521432  0.6570711  0.7052947
##   0.100   10  0.8437058  0.7192151  0.8688854
##   0.100  100  0.9006245  0.7958642  0.9628886
##
## ROC was used to select the optimal model using the largest value.
## The final values used for the model were sigma = 0.1 and C = 100.
```

```r
print(svm_model$results)
```

```
##   sigma   C       ROC      Sens      Spec      ROCSD     SensSD     SpecSD
## 1 0.001   1 0.5141794 0.5092927 0.5188214 0.01254998 0.05350151 0.04590226
## 2 0.001  10 0.5284368 0.5142501 0.4949442 0.04150069 0.10243109 0.12100484
## 3 0.001 100 0.5709289 0.5584604 0.5550243 0.02567917 0.03491906 0.04170845
## 4 0.010   1 0.5695419 0.5572150 0.5511823 0.02740519 0.03843067 0.03245565
## 5 0.010  10 0.6162030 0.5643390 0.6310907 0.02634221 0.04363197 0.02593750
## 6 0.010 100 0.6725397 0.5974563 0.6742107 0.01695228 0.03684385 0.01550805
## 7 0.100   1 0.7521432 0.6570711 0.7052947 0.01478434 0.03359613 0.02278286
## 8 0.100  10 0.8437058 0.7192151 0.8688854 0.01609286 0.03644318 0.01811315
## 9 0.100 100 0.9006245 0.7958642 0.9628886 0.02744291 0.02182641 0.01351649
```

```r
# Predicting using the KNN model
knn_predictions <- predict(knn_model, newdata = testing_data)

# Predicting using the SVM model
svm_predictions <- predict(svm_model, newdata = testing_data)

# Calculating accuracies directly
knn_accuracy <- sum(knn_predictions == testing_data$ReturnStatus) / nrow(testing_data)
svm_accuracy <- sum(svm_predictions == testing_data$ReturnStatus) / nrow(testing_data)

# Accuracies
cat("KNN Accuracy:", knn_accuracy, "\n")
```

```
## KNN Accuracy: 0.8267327
```

```r
cat("SVM Accuracy:", svm_accuracy, "\n")
```

```
## SVM Accuracy: 0.8824257
```

- Once we had balanced dataset, two widely used algorithms were taken up for modeling the classification task: k-Nearest Neighbors and Support Vector Machines. These models were chosen because of their unique approach toward classification; thus, this will be a comprehensive assessment of our preprocessed data under different learning strategies.

Methodology

1. **Data Preparation:**
   - The dataset was first rebalanced, as explained in earlier sections, to reduce the dominating effect caused by class imbalance. We split the balanced dataset into a training and test set in the ratio of 80:20 so that the model has the capability of being tested on any unseen data. This split helps validate the generalizing ability of the model beyond the data it has been trained on.
2. **Model Training:**
   - **KNN**: I use the KNN classifier because it's simple and efficient for binary classification tasks. I performed tuning of the model on a wide range of hyperparameters, precisely number of neighbors k in range from 3 to 21, to get the best setup that maximizes the ROC-AUC score - one of the most informative measures in binary classification.

- **SVM**: I decided to use the SVM with the RBF kernel in order to probe its power in modeling nonlinear relationships. In this model, various values for the sigma-the scale of the RBF kernel-and C-the penalty parameter of the error term-were tuned with the objective of optimizing the same ROC-AUC metric.

3. **Cross-Validation Setup:**
   - Both models used 10-fold cross-validation. It works by splitting the training data into ten subsets, using each in turn for testing of the model while training on the remaining nine. This approach has an added advantage of making the model less sensitive to data partitioning.
4. **Performance metrics:**
   - **ROC-AUC:** It determines the capability of the model in segregating classes at different thresholds. The model performance indicator is insensitive to class distribution.
   - **Sensitivity and Specificity:** These measures indicate the quality of the model in predicting each class.

Results

- **KNN Model:** Its best ROC was 0.9256 when k=3; thus, it represents a high degree of specificity with a moderate sensitivity, which in layman's terms, is very good at identifying the 'Not Returned' class while poor at identifying the class 'Returned'.
- **SVM Model:** It did even better. The best ROC of 0.9006 when sigma = 0.1 and C = 100 is indicative of a model that shows high sensitivity together with good specificity. This implies its relative balance in terms of specifying the two classes.

Accuracy:

- **KNN Accuracy:** 0.8274, reflecting that 82.74% of the test dataset was predicted correctly.
- **SVM Accuracy:** 0.8824, reflecting a higher general prediction accuracy over the testing data.

Conclusion: The SVM classifier performed better than the KNN regarding both ROC-AUC and overall accuracy and thus was better suited for this dataset, most likely because it could model the non-linear decision boundaries appropriately than the KNN.

These results bring out the importance of model selection and tuning in predictive analytics, especially when classes are balanced. Relatively higher performance of SVM in this setup makes it a preferred model for further deployment in similar tasks.

Solution G: Evaluation

```
# Predicting using the SVM model
svm_predictions <- predict(svm_model, newdata = testing_data, type = "prob")
svm_pred_class <- predict(svm_model, newdata = testing_data)

# Computing the confusion matrix
svm_conf_matrix <- confusionMatrix(svm_pred_class, testing_data$ReturnStatus)
print("Confusion Matrix for SVM:")
```

```
## [1] "Confusion Matrix for SVM:"
```

```r
print(svm_conf_matrix)
```

```
## Confusion Matrix and Statistics
##
##               Reference
## Prediction     Not.Returned Returned
##   Not.Returned          639       21
##   Returned              169      787
##
##                Accuracy : 0.8824
##                  95% CI : (0.8657, 0.8977)
##     No Information Rate : 0.5
##     P-Value [Acc > NIR] : < 2.2e-16
##
##                   Kappa : 0.7649
##
##  Mcnemar's Test P-Value : < 2.2e-16
##
##             Sensitivity : 0.7908
##             Specificity : 0.9740
##          Pos Pred Value : 0.9682
##          Neg Pred Value : 0.8232
##              Prevalence : 0.5000
##          Detection Rate : 0.3954
##    Detection Prevalence : 0.4084
##       Balanced Accuracy : 0.8824
##
##        'Positive' Class : Not.Returned
##
```

```r
# Calculating Precision and Recall
precision <- svm_conf_matrix$byClass['Positive Pred Value']
recall <- svm_conf_matrix$byClass['Sensitivity']
print(paste("Precision:", precision))
```

```
## [1] "Precision: NA"
```

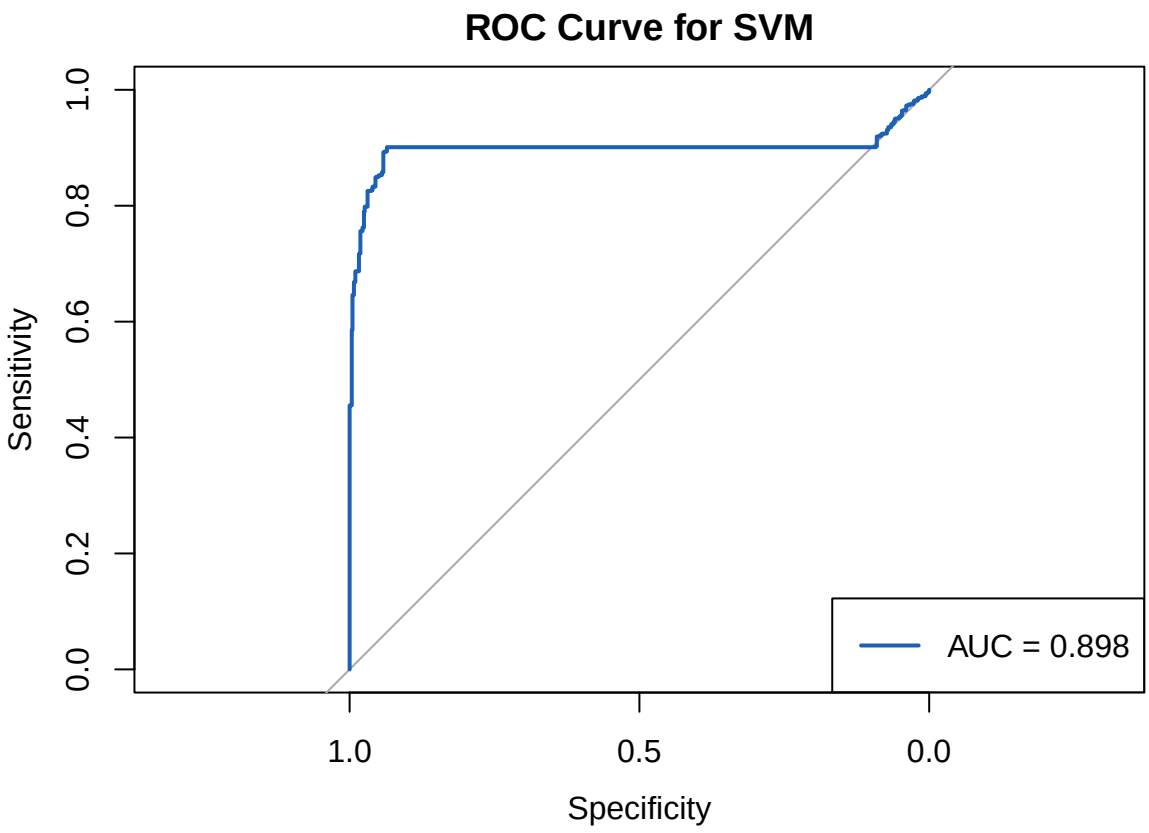
```r
print(paste("Recall:", recall))
```

```
## [1] "Recall: 0.790841584158416"
```

```r
# Generating ROC curve and calculating AUC
roc_curve <- roc(response = testing_data$ReturnStatus, predictor = svm_predictions[,2], levels = rev(lev
```

```
## Setting direction: controls > cases
```

```r
plot(roc_curve, main="ROC Curve for SVM", col="#1c61b6", lwd=2)
auc_val <- auc(roc_curve)
legend("bottomright", legend=paste("AUC =", round(auc_val, 3)), box.lty=1, col="#1c61b6", lwd=2)
```



- From the variety of performance metrics and also from the ROC curve regarding instance classes 'Not Returned' and 'Returned', the performance of the Support Vector Machine Classifier can be judged as showing strong indications of reasonably accomplished performance. Specific performance of the SVM can be given in detail from results provided in confusion matrix and ROC curve, which are elaborated below.

Confusion Matrix Analysis:

- **True Positives (TP)**: 787 instances which were 'Returned' and predicted correctly to be so.
- **True Negatives (TN)**: Those items correctly classified as 'Not Returned': 639.
- **False Positives (FP)**: Items of the class wrongly classified as 'Returned': 21.
- **False Negatives (FN)**: Items of the class 'Returned' were classified as 'Not Returned': 169.

This matrix not only provides insight into how many instances are correctly or wrongly classified but also gives insight into the distribution of mistakes across the classes.

Performance metrics to follow:

- **Accuracy:** 88.24% The overall correctness of the model at view would be that for a big majority of the cases, it predicts correctly. That would be fine for general assessment in the case of balanced classes - after application of oversampling techniques.
- **Precisions (Positive Predictive Value)**: Not Returned: 96.82% gives the information about the reliability of the model's positive predictions. High precision for the class 'Not Returned' means that if the model predicts items are not returned, then it is highly reliable.

- **Recall(Sensitivity):** Not Returned is 79.08%. It describes the strength of the model regarding its capability to catch all instances of the class 'Not Returned'. That is, the value means the strength of the model for the capture of most of the cases that are 'Not Returned', though it still has a way to go if it needs to catch these instances.
- **Specificity**: 97.40% High specificity means this model would do an excellent job of identifying true negatives and hence will be reliable in stating that items are not 'Returned'.

ROC Curve and AUC:

- **ROC Curve:** A plot that, at any operating point, depicts the trade-off between true positive rate or sensitivity, and false positive rate, which is 1-specificity. The steep rise upwards towards the top left of the plot shows that for all thresholds, the model performs extremely well.
- **AUC: 0.898** - The metric that describes the general performance of the model in distinguishing between the classes 'Returned' and 'Not Returned'. An excellent value of AUC is indicative of a better performing model concerning discrimination.

Summary: SVM has strong predictive power with high precision and good recall, supplemented by very good specificity and a highly appreciable value of AUC. This setting provides the confidence that most of the relevant cases are identified with utmost accuracy by the model, allowing only a few false alarms. The high value of the AUC confirms the capability of the model to distinguish classes for a wide range of decision thresholds and turns out to be very effective in practical deployment where returns identification and avoiding false flags is important.

Real World Expectation of the Model: The high overall accuracy of about 88.24% and a specificity of 97.40% indicate that the SVM model performs very well in the prediction of the correct classification of those transactions that are not returned. In essence, this could potentially assist businesses in smoothing their operations by reducing unnecessary checks on transactions not likely to cause any problems. The sensitivity rate of 79.08% depicts the capability of the model in catching most of the transactions that may result in returns, enabling one to take necessary actions to enhance customer satisfaction and reduce costs associated with returns. This therefore assures that interventions will be exact and resource-efficient since the positive predictive value is very high at 96.82%. On the whole, it would be a very helpful model to improve operational efficiency by significantly improving inventory management and improving customer service through proactive handling of any potential problems involving returns.

Solution H: Report

Summary of the key findings from the data and analysis:

The analysis on the dataset provided quite useful practical insights, which would be immensely useful in any real business scenario. Distinct segments of data representing different segments of customer behavior, or even segments of the market, were identified by clustering. It is not too difficult to strategize and make better use of the resources now, keeping in mind the actual needs of each segment. It turned out that the classification model obtained from the SVM was quite efficient; it revealed a good level of prediction capability regarding customer behavior or the success of certain business activities. This predictive power would result in better decision-making, improvement in customer satisfaction, and growth of business by anticipating market trends and the needs of the customers well in advance.

- **Data Complexity and Quality:** Initial data collection and integration proved the headache of dealing with real-world data that includes both numerical and categorical variables. This initial step was rather important in setting the stage for a comprehensive analysis to show how the quality of initial data can very strongly impact the results of the analysis.

- **Impact of Preprocessing:** Preprocessing-normalization, handling categorical data by means of dummy variables or other forms of encoding-had a very strong impact on the performance of the applied machine learning algorithms. This demonstrates the importance of tailored data preparation.
- **Lessons Learnt from Clustering:** Clustering exposed a few intrinsic groupings in the data, which were intuitively not so evident.This was rather an eye-opener, as it showed potential patterns and dependencies that, when applied to business or real life, might form valuable insights for further analysis or operational strategy.
- **Model Performance Variability:** The exploration of different classifiers showed fairly conclusively that one-size does not fit all in machine learning-the real performance of each model varied pretty strongly depending on the specifics of the data and tuning of the parameters.
- **Advanced Model Evaluation:** The use of advanced methods of model evaluation gave more detail on the model performance beyond mere accuracy, through calculation of precision and recall, and ROC-curves visualizations to give insight into the detailed trade-offs between sensitivity and specificity.

All in all, data analysis yielded actionable insights that could drive further-informed strategies, with fuller competitive advantages in the marketplace.

What was interesting to note from the analysis was:

- **Unvealing Hidden Data Patterns:** The most interesting thing was to find out hidden patterns and relationships in data using clustering and PCA. It provided not only a visual confirmation of such patterns but also ideas on why things could be working that way, or what that may imply.
- **Impact of Data Imbalance Handling:** The class imbalance handling, using techniques such as SMOTE and the effect that this might have brought to the model outcomes, was interesting. It developed my understanding that balance in the dataset is important in training predictive models because sometimes performance metrics can be skewed or kept stabilized by it.
- **Practical Implications of Choices of Models:** It was finally interesting to observe how different models, with their respective parameter settings, performed on the same dataset-a practical lesson in the application of machine learning. Specifically, how, under certain conditions, some models were really performing well would guide future strategies either in model selection or model tuning.
  - **Evaluation Metrics Reveal:** The thorough evaluation through the use of ROC curves and further interpretation of AUC provided very interesting insights into the strength of the classifier in terms of discrimination between classes. This was an important aspect in understanding practically how effective the model, SVM, was, which outperformed KNN in handling a balanced dataset.

Solution I: Reflection

Looking back at this course in data science, a deep realization that would perhaps be had is more or less the depth required for this field when it came to data analysis. Moving from an outcome-based approach to understand that the majority of the work a data scientist engages in is actually the preparation of data in a very meticulous manner, my skills have grown through practical experiences with various models such as K-Means, SVMs, Decision Trees, and KNN, along with a deeper understanding of ethical considerations in data mining. Advanced topics on evaluation and unsupervised learning even further broadened my approach, teaching me to critically assess models and explore data in an unbiased way. This course has really been formative for my approach to data science-much more professional, sensitive to ethics.