

# **Customer Segmentation and Marketing Strategy Report**

Using RFM & K-Means Algorithm



Table of Contents

[Introduction 3](#_Toc156160698)

[Requirements Specification 3](#_Toc156160699)

[Data Requirements 3](#_Toc156160700)

[Software and Libraries 3](#_Toc156160701)

[RFM Analysis Implementation 3](#_Toc156160702)

[Customer Segmentation 4](#_Toc156160703)

[Marketing and Recommendations 4](#_Toc156160704)

[Functional and Non-Functional Requirements 4](#_Toc156160705)

[Functional Requirements 4](#_Toc156160706)

[Non-Functional Requirements 4](#_Toc156160707)

[System Design and Design Implementation 5](#_Toc156160708)

[Data Processing and Exploration 5](#_Toc156160709)

[Exploratory Data Analysis (EDA) 5](#_Toc156160710)

[RFM Analysis Implementation 5](#_Toc156160711)

[Customer Segmentation 5](#_Toc156160712)

[Marketing and Recommendations 5](#_Toc156160713)

[System Testing and Maintenance 5](#_Toc156160714)

[System Testing 5](#_Toc156160715)

[Maintenance 6](#_Toc156160716)

[Architecture of Proposed Machine Learning Model 6](#_Toc156160717)

[Algorithm Selection and Computation 7](#_Toc156160718)

[Flowchart of Overall Algorithm Implementation 8](#_Toc156160719)

[Algorithm Implementation 9](#_Toc156160720)

[**1.** **Exploratory Data Analysis (EDA)** 9](#_Toc156160721)

[**2. Data Visualization** 12](#_Toc156160722)

[**3. Data Cleaning** 13](#_Toc156160723)

[**4. RFM Based Customer Clustering** 14](#_Toc156160724)

[Insights from RFM algorithm implementation 17](#_Toc156160725)

[K-Means Clustering 18](#_Toc156160726)

[Insights from K-Means algorithm implementation 21](#_Toc156160727)

[Stakeholders 21](#_Toc156160728)

[Critical Reflection 22](#_Toc156160729)

[**RFM Analysis** 22](#_Toc156160730)

[**K-means Clustering** 22](#_Toc156160731)

[**Objectives Achievement** 22](#_Toc156160732)

[Conclusion 23](#_Toc156160733)

[**Key Achievements** 23](#_Toc156160734)

[**Future Work** 23](#_Toc156160735)

[Appendix 24](#_Toc156160736)

[**Algorithm Code Implementation** 24](#_Toc156160737)

[Data Visualization 35](#_Toc156160738)

[**Distribution of Quantity Ordered** 35](#_Toc156160739)

[**Distribution of Sales** 35](#_Toc156160740)

[**Distribution of Transactions Per Customer** 36](#_Toc156160741)

[**Monthly Active Customers** 36](#_Toc156160742)

[**Customers Distribution by Country** 37](#_Toc156160743)

[**Monthly Sales Trend** 37](#_Toc156160744)

[**Correlation Heatmap** 38](#_Toc156160745)

[**Top Selling Products** 38](#_Toc156160746)

[**Invoice Count by Hour of the Day** 39](#_Toc156160747)

[**Customer Repeat Rate** 39](#_Toc156160748)

[**Product Price Distribution by Country** 40](#_Toc156160749)

[**Geospatial Distribution of Total Sales by Country** 40](#_Toc156160750)

[**Total Sales Over Time** 41](#_Toc156160751)

[**Pair Plot for Key Numerical Features** 41](#_Toc156160752)

## 

## Introduction

Customer segmentation plays an important role in marketing strategies, enabling businesses to understand the diverse needs of their customers. Based on customers' characteristics like demographics, and purchasing behavior, companies can adapt marketing strategies that enhance customer experience.

In this analysis, we performed a comprehensive customer segmentation study with RFM (Recency, Frequency, Monetary) analysis technique. The dataset contains information about customer transactions, like product purchases, dates, and monetary values. The main objective of this analysis is to formulate targeted marketing strategies by identifying distinct segments within the customer base.

Using exploratory data analysis (EDA), data visualizations, data cleaning, data scaling, outlier detection and removal and modeling techniques, our goal is to get actionable insights that can help in decision-making processes. By the end of this analysis, we anticipate offering valuable recommendations for marketing campaigns, and product recommendations thus contributing to the optimization of marketing efforts and the enhancement of the overall customer experience.

## Requirements Specification

Implementation of Customer Segmentation and Marketing Strategy includes a number of prerequisites and specifications to ensure a seamless analysis and interpretation of results.

### Data Requirements

* **Dataset:** Dataset contains customer transactions, including columns such as 'Invoice,' 'StockCode,' 'Description,' 'Quantity,' 'InvoiceDate,' 'Price,' 'Customer ID,' and 'Country.'
* **Data Quality:** The dataset should be preprocessed to handle missing values, outliers, and ensure data quality for accurate results.

### Software and Libraries

* **Python Environment:** The analysis is conducted using the Python programming language.
* **Libraries:** Ensure the installation of essential libraries such as pandas, numpy, matplotlib, seaborn, scikit-learn, statsmodels, and kneed for data manipulation, analysis, and visualization.

### RFM Analysis Implementation

* **RFM Scores:** The successful calculation of Recency, Frequency, and Monetary scores for each customer using the provided code.
* **RFM Visualization:** The visualization of RFM scores to understand the distribution and patterns within the dataset.

### Customer Segmentation

* **Segmentation Algorithm (K-Means):** Utilization of the RFM scores to implement customer segmentation.
* **Segment Interpretation:** Clear interpretation of the segments obtained, including their characteristics and distribution percentages.

### Marketing and Recommendations

* **Strategy Formulation:** Based on the segmentation results, formulate targeted marketing strategies for each customer segment.
* **Product Suggestions:** Offer insights into product recommendations and promotional campaigns for different segments.

## Functional and Non-Functional Requirements

### Functional Requirements

|  |  |
| --- | --- |
| Data Reading | The system must be able to read and load the provided CSV file containing relevant customer transaction data. |
| Data Preprocessing | Missing values and outliers in the dataset must be handled systematically to ensure data quality. |
| RFM Analysis | The system should calculate Recency, Frequency, and Monetary scores for each customer based on their transactions. |
| Customer Segmentation | Utilize the RFM scores to implement a robust customer segmentation algorithm for categorizing customers. |
| Visualization | Generate a variety of visualizations (line plots, bar charts, etc.) to interpret and communicate analysis results. |
| Marketing Recommendations | Formulate targeted marketing strategies and recommendations for each customer segment based on RFM analysis. |

### Non-Functional Requirements

|  |  |
| --- | --- |
| Performance | The system must handle the analysis efficiently, even with large datasets, providing results in a reasonable time. |
| Reliability | The system should be reliable, producing consistent and accurate results across multiple runs. |
| Scalability | The code should be scalable, allowing for future enhancements and accommodating changes in the dataset or requirements. |
| Maintainability | Code must be documented and easily maintainable, allowing for updates and bug fixes. |

## System Design and Design Implementation

### Data Processing and Exploration

* **CSV File Reading:** The Python code utilizes pandas for reading and exploring the CSV file, providing a foundation for subsequent analysis.
* **Data Cleaning:** Missing values are handled, and the dataset is cleaned to ensure data quality and reliability.

### Exploratory Data Analysis (EDA)

* **Visualization**: Matplotlib and Seaborn libraries are employed to create visualizations, ranging from basic to advanced levels, offering a profound understanding of the dataset.
* **Insights Extraction:** EDA is conducted to extract meaningful insights, trends, and patterns from the data.

### RFM Analysis Implementation

* **Code Structure:** Python functions are created to implement RFM analysis efficiently, with clear code structure and documentation.
* **Recency, Frequency, and Monetary Calculation:** The analysis involves calculating Recency, Frequency, and Monetary scores for each customer using pandas’ functionalities.

### Customer Segmentation

* **Segmentation Algorithm:** The RFM scores are leveraged to implement customer segmentation, employing quantiles and labels for Recency, Frequency, and Monetary values.
* **Segment Interpretation:** Segments are interpreted and visualized, providing a clear understanding of the distribution and characteristics of each segment.

### Marketing and Recommendations

* **Segment-Specific Strategies:** Based on the segmentation results, segment-specific marketing strategies are formulated, and product recommendations.

## System Testing and Maintenance

### System Testing

* **Data Integrity:** The integrity of the dataset is verified through data quality checks, ensuring that outliers, missing values, and anomalies are appropriately handled.
* **Visualization Testing:** Visualizations are rigorously examined to confirm their accuracy in representing insights and trends derived from the data.
* **Segmentation Accuracy:** The accuracy of customer segmentation is tested by validating whether the RFM scores effectively categorize customers into meaningful segments.

### Maintenance

* **Code Documentation Updates:** Documentation is essential for maintaining code clarity and helping in future modifications. Code comments reflect the logic.
* **Algorithm Enhancements:** Continuous evaluation of segmentation algorithms and data preprocessing techniques allows improvement of accuracy and relevance of customer segmentation.
* **Iterative Development:** Agile methodology is adopted for system maintenance, allowing for iterative development cycles. Changes and improvements are implemented incrementally, ensuring that each iteration adds value to the system.

## Architecture of Proposed Machine Learning Model

The proposed AI model design is based upon the combination of the RFM (Recency, Recurrence, Monetary) method for customer segmentation and the K-Means clustering algorithm. The architecture is intended to examine customer transaction data, recognize significant fragments, and provide advertising strategies.

In the first phase of the architecture, the RFM analysis is used to calculate three critical measurements for each customer: Recency, Frequency, and Monetary values. The Recency metric defines the time since a customer's last purchase, Frequency measures how frequently customers buy products, and Monetary represents the total amount spent by each customer.

Following the RFM technique, the K-Means clustering algorithm is used for customer segmentation. K-Means is a centroid-based clustering strategy that segments the dataset into 'k' different groups. About customer segmentation, 'k' defines the number of fragments or groups that the algorithm recognizes. The RFM scores acquired before act as information highlights for the K-Means calculation, allowing group customers based on similar purchasing behaviors.

The K-Means algorithm iteratively assigns customers to clusters by minimizing the within-cluster sum of squares. Each cluster represents a distinct segment of customers who share common traits in terms of Recency, Frequency, and Monetary values. The result is a segmentation of the customer base into groups that can be interpreted and targeted with specific marketing strategies.

By employing these techniques, businesses can gain a comprehensive understanding of their customers, identifying segments such as 'Champions' who are highly valuable customers, or 'About to Sleep' customers who may require targeted efforts to re-engage.

## Algorithm Selection and Computation

The selection of the RFM (Recency, Frequency, Monetary) technique for customer segmentation and the K-Means clustering algorithm is driven by their inherent advantages and suitability for achieving our marketing and recommendation goals.

The RFM technique is chosen because it provides a straightforward yet powerful approach to categorize customers based on their transaction history. Recency, Frequency, and Monetary values encapsulate distinct facets of customer behavior, offering a holistic view of their engagement with the business. This method allows for a understanding of customer segments, providing actionable insights into their preferences and loyalty.

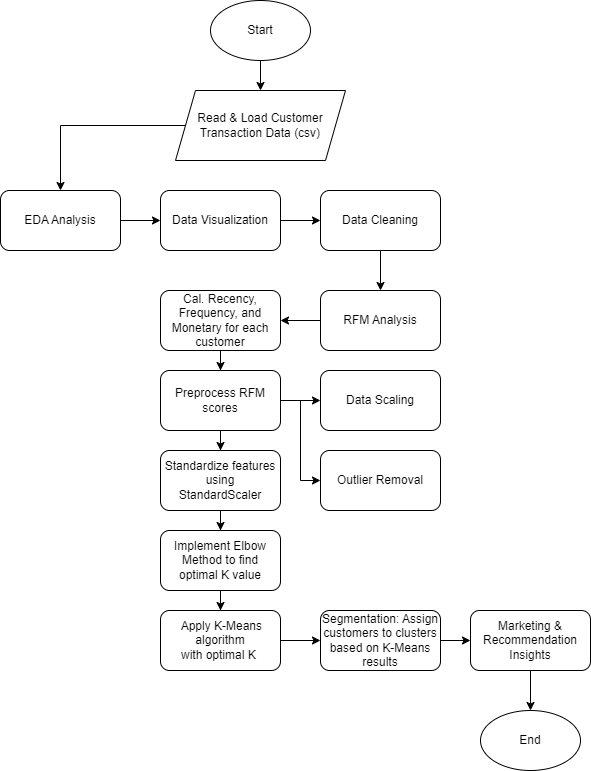
K-Means clustering is employed as it efficiently partitions customers into distinct groups based on similarity in RFM scores. The algorithm is particularly effective for its simplicity and ability to handle large datasets. By iteratively assigning customers to clusters and optimizing for within-cluster homogeneity, K-Means identifies natural groupings within the data. The choice of K-Means aligns with our goal of uncovering meaningful segments that can inform targeted marketing strategies.

The Elbow Method is employed to determine the optimal number of clusters (k) in K-Means. This method involves running the K-Means algorithm with different values of k and plotting the sum of squared distances within each cluster. The 'elbow' point on the graph, where the rate of decrease in the sum of squared distances slows down, indicates the optimal number of clusters. This approach aids in selecting a value for k that maximizes the homogeneity within clusters while maintaining distinctiveness between them.

The benefit of using the RFM-K-Means approach lies in its ability to deliver actionable customer segments. These segments, such as 'Champions,' 'Potential Loyalists,' and 'About to Sleep,' are easily interpretable and can directly influence marketing strategies. For instance, tailored campaigns can be designed to retain loyal customers, re-engage inactive ones, or welcome new customers. Additionally, the granularity of the segments allows for personalized recommendations and promotions, enhancing the overall customer experience.

This method empowers businesses with a strategic framework to allocate resources effectively, optimize marketing efforts, and foster customer loyalty. By leveraging the insights derived from RFM-K-Means clustering, businesses can craft targeted and personalized marketing initiatives, resulting in improved customer satisfaction, increased retention, and ultimately, enhanced business growth.

## Flowchart of Overall Algorithm Implementation



## Algorithm Implementation

### **Exploratory Data Analysis (EDA)**

1. **Data Information**

* **df.info():** Displays general information about the dataset, including data types and non-null values.

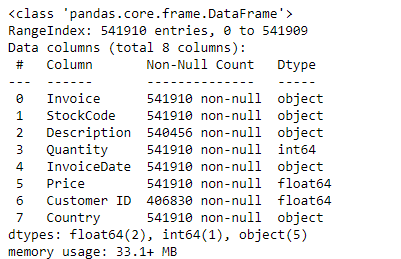


Figure 1 - Dataset details along with data type of each column

* **df.describe():** Provides statistical summary, including count, mean, std deviation, min, 25th percentile, median, 75th percentile, and max for numerical columns.

A table with numbers and letters

Description automatically generated

Figure 2 - Stats of Numerical columns in dataset

* **df.nunique():** Shows the number of unique values in each column.

A screenshot of a computer code

Description automatically generated

Figure 3 - Extract the unique values each volume has.

As per results there are 38 countries with 4373 clients.

* **missing\_values:** Counts the number of missing values in each column.

A screen shot of a computer code

Description automatically generated

Figure 4 - Missing Or Null Values in Each Column

1. **Frequent Values in 'StockCode'**

Displays the top frequent values in the 'StockCode' column, indicating the most commonly occurring product codes.

A number and numbers on a white background

Description automatically generated

Figure 5 - Top 5 Selling Stock Code in Dataset

1. **Average Price per Item**

Calculates the average price for each unique item in the 'StockCode' column.

A number on a white background

Description automatically generated

Figure 6 - First 5 stock code with their average price

1. **Date and Total Sales Calculations**

* Converts the 'InvoiceDate' column to datetime format.
* Creates a new column 'TotalSales' by multiplying 'Quantity' and 'Price' columns.

1. **Total Sales per Country**

Groups the data by 'Country' and calculates the total sales in each country.

A black text on a white background

Description automatically generated

Figure 7 - Sales with respect to countries

United Kingdom has the highest sale as per dataset.

1. **Monthly Sales Trends**

* Extracts the month from the 'InvoiceDate' column and adds a new 'Month' column to the DataFrame.
* Groups the data by month and calculates the total sales in each month.

A screenshot of a computer

Description automatically generated

Last 4 months of a year are highest sales month as per dataset while first 8 months have an average sale.

1. **Total Quantity per Product**

* Groups the data by 'StockCode' and calculates the total quantity sold for each product.
* Identifies the top-selling products based on the total quantity sold.

A white background with black text

Description automatically generated

Highest demanding product as per dataset is StockCode with ID 22197

1. **Average Invoice Value and Quantity per Transaction**

Calculates the average total sales value per invoice and the average quantity per transaction.





### **2. Data Visualization**

**Distribution of 'Quantity' and 'Total Sales'**

* Histograms are created to show the distribution of 'Quantity' and 'Total Sales.'
* The purpose is to identify patterns and the range of values for these two key variables.

**Distribution of Transactions per Customer**

* Histogram depicting the distribution of the number of transactions per customer.
* Provides insights into customer transaction behavior.

**Monthly Active Customers**

* Line plot showing the number of active customers each month.
* Helps in understanding customer engagement trends over the months.

**Customer Distribution by Country:**

* Bar chart illustrating the distribution of customers across different countries.
* Offers insights into the geographical distribution of customers.

**Intermediate Level Visualization: Monthly Sales Trends**

* Line plot showing the monthly sales trends over different years.
* Allows for the observation of any recurring patterns or seasonality.

**Correlation Heatmap**

* Heatmap representing the correlation matrix of numerical features.
* Assists in identifying relationships between variables.

**Top Selling Products (Horizontal Bar Chart)**

* Horizontal bar chart displaying the top-selling products based on the total quantity sold.
* Helps in identifying the most popular products.

**Invoice Count by Hour of the Day:**

* Line plot showing the count of invoices based on the hour of the day.
* Offers insights into the peak hours of business activity.

**Customer Repeat Rate**

* Pie chart illustrating the distribution of customers based on their repeat purchase behavior.
* Aids in understanding customer loyalty.

**Product Price Distribution by Country**

* Violin plot showcasing the distribution of product prices across different countries.
* Provides insights into price variations.

**Temporal Heatmap for Monthly Sales**

* Heatmap representing the total sales on a monthly basis over different years.
* Visualizes the temporal trends in sales.

**Geospatial Distribution of Total Sales by Country (Choropleth Map)**

* World map colored by total sales per country.
* Offers a geographical perspective on sales distribution.

**ARIMA Model Diagnostic Plots**

* Diagnostic plots for an ARIMA time series model fitted to the 'TotalSales' data.
* Assesses the model's goodness-of-fit.

**Multivariate Time Series Plot with Subplots**

* Subplots showing the total sales and quantity sold over time.
* Helps in visualizing the trends and patterns in sales and quantity.

**Pairplot for Key Numerical Features**

* Pairplot displaying relationships between key numerical features (Quantity, Price, TotalSales).
* Offers a comprehensive view of pairwise relationships.

### **3. Data Cleaning**

**Drop Null Values**

df.dropna(inplace=True) is used to remove rows with null values to ensures missing data does not impact the analysis.

**Check and Drop Duplicates**

* df[df.duplicated()].shape checks for duplicate rows in the dataset.
* df = df.drop\_duplicates(keep='first') removes duplicate rows, keeping the first occurrence.

**Handling Cancelled Invoices**

* df[df['Invoice'].str.contains('C')] identifies rows with invoices marked as 'C,' indicating cancellations.
* df = df[~df['Invoice'].str.contains('C')] drops rows associated with cancelled invoices.
* Eliminates the impact of cancelled transactions on the analysis.

**Check and Remove Rows with Price as 0**

* df[df['Price']==0] checks for rows where the price is zero.
* df = df[df['Price']>0] removes rows with zero prices.
* Ensures that only valid transactions with non-zero prices are considered.

### **4. RFM Based Customer Clustering**

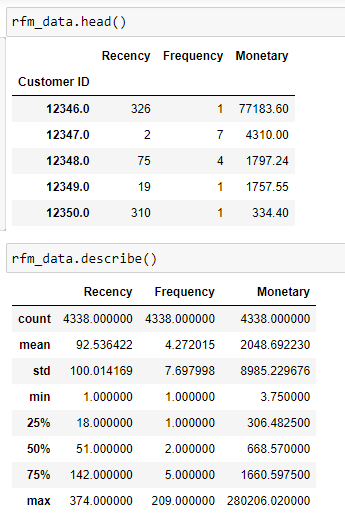
**RFM - Recency, Frequency, and Monetary**

Calculate RFM Metrics:

* The code calculates Recency, Frequency, and Monetary values for each customer.
* reference\_date is set as the maximum invoice date plus one day.
* rfm\_data is generated using the groupby function to aggregate metrics per customer.

Descriptive Statistics:

* Descriptive statistics for the RFM metrics are displayed using rfm\_data.describe().



The figure demonstrates the recency, frequency, and monetary value of each customer along with statistical values for RFM metrics. Customer with id 12350 have high recency value along with low frequency and low monetary value which overall classify this customer as low value customer while customer with id 12347 have low recency and good frequency and monetary value indicating a valuable customer.

**Customer Segmentation - Using Scoring RFM Metrics**

Check for Outliers:

* Boxplots are created to visualize the distribution of Recency, Frequency, and Monetary values.
* Outliers are identified, indicating extreme values in the data.

A graph of a number of objects

Description automatically generated with medium confidence

Dots in the images outside of the max value of RFM metrics highlights the outliers in those parameters.

Replace Outliers:

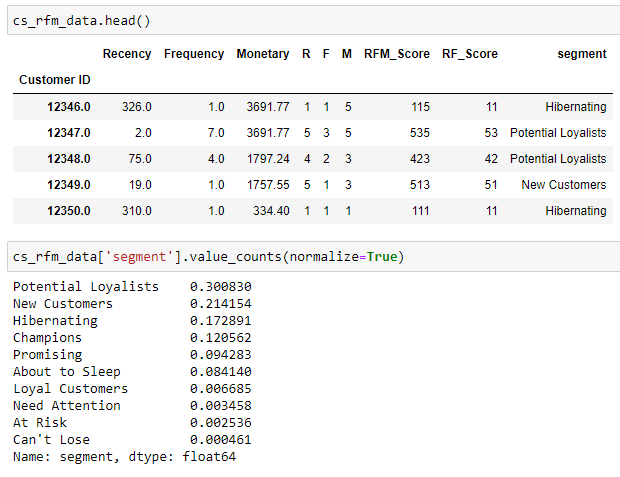
* Outliers are replaced with lower and upper bounds to mitigate their impact on subsequent analysis.
* This ensures that extreme values do not skew the results.

Create RFM Scores:

* Quantiles are created for Recency, Frequency, and Monetary values.
* R, F, and M scores are assigned to each customer based on their quartile.
* The RFM score is then generated by combining the R, F, and M scores.

Customer Segmentation:

* A segmentation map (seg\_map) is defined to map RF scores to customer segments.
* The cs\_rfm\_data DataFrame is updated with the customer segments.
* The distribution of customer segments is displayed using cs\_rfm\_data['segment'].value\_counts(normalize=True).



This image demonstrate how to map recency, frequency and monetary values to R,F and M columns with values range between 1 to 5 While RF and RFM score columns are just string concatenation of those columns and based on RF score we map a particular client to segment like Hibernating, New Customer etc. In the figure customer with recency 326 get a R as 1 because high recency is bad while customer with recency 2 get a R value of 5.

### Insights from RFM algorithm implementation

**Targeted Marketing Campaigns:**

1. **Potential Loyalists and New Customers (51.55%):** Focus on attracting and retaining these segments. Run special promotions, loyalty programs, and personalized offers to encourage repeat purchases and build brand loyalty.
2. **Hibernating and About to Sleep (25.02%):** Implement reactivation campaigns to win back these customers. Provide incentives, discounts, or exclusive offers to re-engage them.

**Product Recommendations:**

1. **Champions (12.06%):** These are highly valuable customers. Suggest premium products, exclusive deals, or early access to new collections to maintain their loyalty.
2. **Promising and Loyal Customers (10.08%):** Continue providing personalized recommendations

**Need Attention, At Risk, and Can't Lose (1.1%):** These segments require different approaches.

1. **Need Attention:** Reach out with targeted communications, offering assistance and addressing concerns.
2. **Feedback from At-Risk Customers:** Gather feedback from the 'At Risk' segment to understand their concerns and improve the customer experience. Implement retention strategies, such as personalized discounts or special offers.
3. **Can't Lose:** Ensure these valuable customers are satisfied and consider personalized gestures to maintain their loyalty.
4. **About to Sleep:** Time-limited promotions or reminders about products they have shown interest in can be effective in rekindling their engagement.

### K-Means Clustering

**Outlier Replacement**

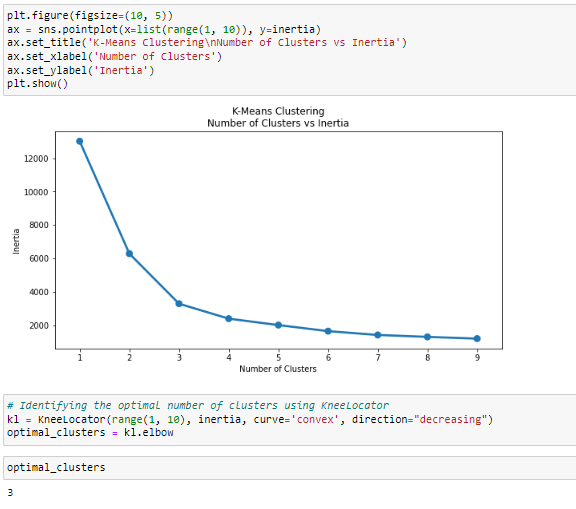
Outliers in the RFM data are replaced with lower and upper bounds to ensure a robust clustering process.

**Standard Scaling**

The RFM data is standardized using StandardScaler to transform the features into a standard normal distribution.

**Elbow Method**

* The Elbow method is applied to determine the optimal number of clusters.
* The inertia (within-cluster sum of squares) is calculated for a range of cluster numbers, and a plot is generated to visualize the "elbow" point.



**KneeLocator**

KneeLocator is utilized to automatically identify the optimal number of clusters based on the elbow point in the plot.

**K-Means Clustering**

* K-means clustering is performed with the optimal number of clusters obtained from the elbow method.
* Cluster labels are assigned to each data point.

**Cluster Analysis**

* A DataFrame is created to store the cluster labels for each customer.
* The cluster labels are joined with the original RFM data.

A screenshot of a graph

Description automatically generated

This figure demonstrates how using k means and RFM metrics we assign a client to one of the 3 clusters so we can group clients and this can help to target specific audience and we can manage our marketing strategies accordingly along with that we can recommend products based on same cluster grouping.

**Cluster Distribution**

The distribution of customers across clusters is displayed using cluster\_distribution.

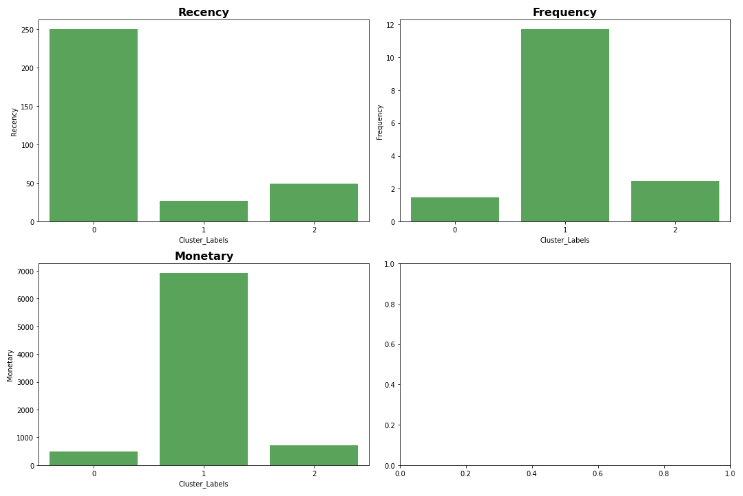
A black text on a white background

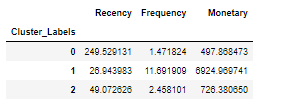
Description automatically generated

Cluster 2 roughly contains 53% of data while cluster 0 and 1 contains 47% of data.

**Cluster Means Visualization**

The mean values of Recency, Frequency, and Monetary metrics for each cluster are calculated and visualized using a bar plot.





These images illustrate distribution of RFM values via bar chart and tabular format with respect to three clusters.

### Insights from K-Means algorithm implementation

**Cluster labeling indicates the grouping of customer based on data set and using cluter label we can recommend customers with stock(s)**

1. Cluster 0 with highest recency rate and lowest frequency and monetary as compared to cluster 1 and 2 is low value customers.
2. Cluster 1 with lowest recency along with highest frequency and monetary rate is high value customers.
3. Cluster 2 with normal recency rate along with good frequency and monetary rate is middle value customers.

## Algorithm Testing and Evaluation

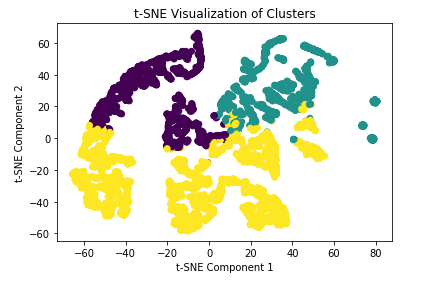
The evaluation and testing phase is crucial in understanding the performance and validity of the clustering model. Let's delve into the key aspects of the evaluation based on the provided results:

**Silhouette Score**

The Silhouette Score is a measure of how well-separated the clusters are. Silhouette scores vary better -1 to 1. A score of 0.51 indicates a good level of separation among the clusters. This suggests that the clusters are reasonably well-defined.

**t-SNE Visualization**

t-SNE visualization provides a two-dimensional representation of the clusters. In the scattered plot, clusters are distinct and well-separated. The plot seems to show relatively distinct clusters.



This diagram shows how K mean algorithm implementation have distributed the data in 3 different clusters.

**Cluster Profiles**

Analyzing the cluster profiles, we observe distinct patterns in terms of Recency, Frequency, and Monetary values for each cluster. Cluster 1, for example, has significantly lower recency, higher frequency, and higher monetary values compared to others.

A number and a number on a white background

Description automatically generated

**Cluster Distribution**

Examining the distribution of customers across clusters is vital. In this case, Cluster 2 dominates with approximately 54% of customers, followed by Cluster 0 (24%) and Cluster 1 (22%).

A white background with black text

Description automatically generated

## Stakeholders

**Marketing Teams**

* Primary stakeholders seeking insights into customer behavior for targeted marketing campaigns.
* Benefit from customer segmentation to tailor promotions and communication strategies.

**Sales Teams**

* Utilize customer segmentation to prioritize leads, focusing efforts on high-potential segments.
* Gain insights into purchasing patterns for effective sales strategies.

**Product Managers**

* Leverage customer clusters to inform product recommendations and inventory planning.
* Align product offerings with the preferences of different customer segments.

**Customer Support**

* Tailor support services based on the needs and behaviors of specific customer groups.
* Enhance customer satisfaction by addressing segment-specific concerns.

**Customers**

* Indirect stakeholders benefiting from personalized experiences and relevant offerings.
* Experience improved interactions with the business based on targeted approaches.

## Critical Reflection

Critical Reflection on Code Implementation and Objectives

The code implementation for customer segmentation using RFM analysis and K-means clustering reflects a thoughtful approach towards understanding customer behavior. Let's critically reflect on the code and its alignment with the stated objectives of customer segmentation for marketing and product recommendations.

### **RFM Analysis**

**Strengths:**

* The RFM analysis effectively captures three crucial aspects of customer behavior: Recency, Frequency, and Monetary value.
* The code implements outlier handling, ensuring that extreme values do not unduly influence the segmentation process.
* The use of quantiles for scoring allows for a robust and intuitive segmentation approach.

**Considerations:**

The manual creation of bins and scoring might be subjective; incorporating more dynamic approaches, such as percentile-based scoring, could enhance objectivity.

### **K-means Clustering**

**Strengths:**

* The implementation of the Elbow method and KneeLocator for determining the optimal number of clusters demonstrates a data-driven approach.
* Standardization of features improves the clustering algorithm's performance.
* The code provides clear visualization of cluster characteristics through bar plots.

**Considerations:**

The choice of the number of clusters remains somewhat subjective, even with the Elbow method.

### **Objectives Achievement**

**Marketing:**

* Identification of distinct customer segments allows for targeted marketing strategies.
* Understanding the RFM characteristics of each cluster enables personalized communication and promotions.

**Product Recommendations:**

* Clusters provide insights into varying purchasing behaviors, aiding in tailored product recommendations.
* Analysis of customer segments can inform inventory planning based on preferences.

The code implementation aligns well with the objectives of customer segmentation for marketing and product recommendations. It successfully leverages RFM analysis and K-means clustering to identify meaningful customer segments. However, continuous refinement and exploration of advanced techniques could further enhance the accuracy and depth of insights derived from customer behavior data.

## Conclusion

The implementation of customer segmentation using RFM analysis and K-means clustering presents a valuable framework for understanding and categorizing customer behavior. The code successfully translates business objectives into actionable insights, providing a foundation for targeted marketing strategies and personalized product recommendations.

### **Key Achievements**

* Recency, Frequency, and Monetary metrics have been effectively utilized to capture different facets of customer engagement.
* Outlier handling and scoring mechanisms contribute to the robustness of the segmentation process.
* The Elbow method and KneeLocator have been applied judiciously to determine the optimal number of clusters.
* Standardization of features enhances the performance of the clustering algorithm.
* The segmentation enables targeted marketing efforts by identifying distinct customer groups with varying behaviors.
* Insights derived from customer clusters can inform product recommendations, aiding in inventory planning and personalized customer experiences.

### **Future Work**

While the current implementation meets the specified objectives, continuous improvement and exploration of advanced techniques are recommended. Future iterations could consider:

* Integrating additional customer features or external data sources for a more comprehensive understanding.
* Exploring advanced clustering algorithms or machine learning models for improved accuracy.
* Regularly monitoring customer behavior trends and adapting segmentation strategies to evolving market dynamics.

By understanding of customer behavior, businesses can tailor their strategies to meet the diverse needs of their customer base, fostering long-term relationships and driving sustainable growth.

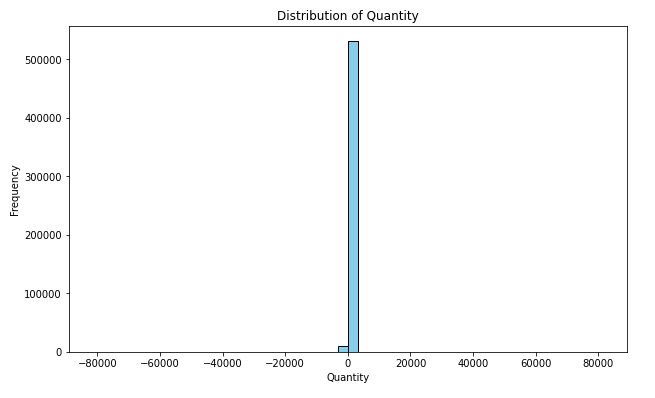
## Appendix

### **Algorithm Code Implementation**

|  |
| --- |
| # -\*- coding: utf-8 -\*-  """Customer Segmentation.ipynb  Automatically generated by Colaboratory.  Original file is located at  https://colab.research.google.com/drive/1gM3Yy\_VSa\_MZJxJGQx77jY5F6cMbNmwW  ## Building a Machine Learning Model for predicting and Enhancing Customer Segmentation and Personalized Marketing in E-Commerce  ### Libraries  """  import pandas as pd  import numpy as np  import matplotlib.pyplot as plt  import seaborn as sns  import datetime as dt  from statsmodels.tsa.seasonal import seasonal\_decompose  import geopandas as gpd  from mpl\_toolkits.axes\_grid1 import make\_axes\_locatable  from statsmodels.tsa.arima.model import ARIMA  from pandas.plotting import autocorrelation\_plot  import matplotlib.dates as mdates  import warnings  warnings.filterwarnings("ignore")  """### Dataset"""  file\_path = 'online\_retail\_II.csv'  df = pd.read\_csv(file\_path, encoding='ISO-8859-1')  df.head()  """### Dataset Analysis"""  df.info()  df.describe()  # find unique values in each column  df.nunique()  missing\_values = df.isnull().sum()  missing\_values  # Frequent values in the 'StockCode' column  df['StockCode'].value\_counts().head()  # Average price per item  average\_price\_per\_item = df.groupby('StockCode')['Price'].mean()  average\_price\_per\_item.head()  # InvoiceDate to datetime format  df['InvoiceDate'] = pd.to\_datetime(df['InvoiceDate'])  # New column for total sales per transaction  df['TotalSales'] = df['Quantity'] \* df['Price']  df.head()  # Determine total sales per country  total\_sales\_per\_country = df.groupby('Country')['TotalSales'].sum()  total\_sales\_per\_country.sort\_values(ascending=False).head()  # Extract the Month from InvoiceDate column  df['Month'] = df['InvoiceDate'].dt.month  # Monthly sales trends  monthly\_sales = df.groupby('Month')['TotalSales'].sum()  print(monthly\_sales)  # Get total quantity sold per StockCode  total\_quantity\_per\_product = df.groupby('StockCode')['Quantity'].sum()  # Top-selling products  top\_selling\_products = total\_quantity\_per\_product.sort\_values(ascending=False).head()  top\_selling\_products  average\_invoice\_value = df.groupby('Invoice')['TotalSales'].sum().mean()  print(f"Average Invoice Value: {average\_invoice\_value}")  # Average quantity per transaction  average\_quantity\_per\_transaction = df.groupby('Invoice')['Quantity'].mean()  print(f"Average Quantity per Transaction: {average\_quantity\_per\_transaction.mean()}")  """### Dataset Visualization"""  # distribution of 'Quantity'  plt.figure(figsize=(10, 6))  plt.hist(df['Quantity'], bins=50, color='skyblue', edgecolor='black')  plt.title('Distribution of Quantity')  plt.xlabel('Quantity')  plt.ylabel('Frequency')  plt.show()  # Distribution plot for TotalSales  plt.figure(figsize=(12, 6))  plt.hist(df['TotalSales'], bins=50, color='skyblue', edgecolor='black')  plt.title('Distribution of Total Sales')  plt.xlabel('Total Sales')  plt.ylabel('Frequency')  plt.show()  """\*\*Distribution of quantity & total sales:\*\*  Maximum transactions have a total quantity and sales close to zero which indicates large small transactions  """  # number of transactions per customer  transactions\_per\_customer = df.groupby('Customer ID')['Invoice'].nunique()  plt.figure(figsize=(10, 6))  plt.hist(transactions\_per\_customer, bins=50, color='orange', edgecolor='black')  plt.title('Distribution of Transactions per Customer')  plt.xlabel('Number of Transactions')  plt.ylabel('Number of Customers')  plt.show()  """\*\*Distribution of Transactions per Customer:\*\*  Maximum Customers have a total transactions close to zero and very less clients have transactions between 25 to 50  """  # Active customers per month  monthly\_active\_customers = df.groupby('Month')['Customer ID'].nunique()  # Plot the monthly active customers  plt.figure(figsize=(10, 6))  plt.plot(monthly\_active\_customers, marker='o', color='green')  plt.title('Monthly Active Customers')  plt.xlabel('Month')  plt.ylabel('Number of Active Customers')  plt.show()  # Unique customers per country  customers\_per\_country = df.groupby('Country')['Customer ID'].nunique()  # Plot the customer distribution by country  plt.figure(figsize=(12, 8))  customers\_per\_country.sort\_values(ascending=False).plot(kind='bar', color='purple')  plt.title('Customer Distribution by Country')  plt.xlabel('Country')  plt.ylabel('Number of Customers')  plt.show()  # Intermediate Level Visualization: Monthly Sales Trends  df['Year'] = df['InvoiceDate'].dt.year  plt.figure(figsize=(12, 6))  sns.lineplot(x='Month', y='TotalSales', data=df.groupby(['Month', 'Year'])['TotalSales'].sum().reset\_index(), marker='o', color='green')  plt.title('Monthly Sales Trends')  plt.xlabel('Month')  plt.ylabel('Total Sales')  plt.show()  # Correlation Heatmap  correlation\_matrix = df.corr()  plt.figure(figsize=(10, 8))  sns.heatmap(correlation\_matrix, annot=True, cmap='coolwarm', fmt='.2f', linewidths=0.5)  plt.title('Correlation Heatmap')  plt.show()  # Top Selling Products (Horizontal Bar Chart)  plt.figure(figsize=(12, 8))  top\_selling\_products = df.groupby('StockCode')['Quantity'].sum().sort\_values(ascending=False).head(10)  top\_selling\_products.sort\_values().plot(kind='barh')  plt.title('Top Selling Products')  plt.xlabel('Total Quantity Sold')  plt.ylabel('Product Code')  plt.show()  # Invoice Count by Hour of the Day  df['Hour'] = df['InvoiceDate'].dt.hour  plt.figure(figsize=(12, 8))  sns.lineplot(x='Hour', y='Invoice', data=df.groupby('Hour')['Invoice'].nunique().reset\_index(), marker='o', color='purple')  plt.title('Invoice Count by Hour of the Day')  plt.xlabel('Hour of the Day')  plt.ylabel('Number of Invoices')  plt.show()  # Customer Repeat Rate  repeat\_customers = df['Customer ID'].value\_counts().value\_counts().reset\_index().rename(columns={'index': 'Number of Purchases', 'Customer ID': 'Count'})  plt.figure(figsize=(10, 10))  plt.pie(repeat\_customers['Count'], labels=repeat\_customers['Number of Purchases'], autopct='%1.1f%%', startangle=90, colors=sns.color\_palette('pastel')[0:5])  plt.title('Customer Repeat Rate')  plt.show()  # Product Price Distribution by Country  plt.figure(figsize=(12, 8))  sns.violinplot(x='Country', y='Price', data=df, palette='viridis')  plt.title('Product Price Distribution by Country')  plt.xlabel('Country')  plt.ylabel('Price')  plt.xticks(rotation=45, ha='right')  plt.show()  # Temporal Heatmap for Monthly Sales  plt.figure(figsize=(16, 8))  # Pivot the data for heatmap  heatmap\_data = df.pivot\_table(index='Month', columns='Year', values='TotalSales', aggfunc='sum')  # Plotting the heatmap  sns.heatmap(heatmap\_data, cmap='viridis', annot=True, fmt=".1f", linewidths=.5, cbar\_kws={'label': 'Total Sales'})  plt.title('Temporal Heatmap for Monthly Sales')  plt.xlabel('Year')  plt.ylabel('Month')  plt.show()  # Load world map  world = gpd.read\_file(gpd.datasets.get\_path('naturalearth\_lowres'))  # Merge the world map with total sales per country  sales\_by\_country = df.groupby('Country')['TotalSales'].sum().reset\_index()  world = world.merge(sales\_by\_country, how='left', left\_on='name', right\_on='Country')  # Plot the geospatial distribution of sales (Choropleth Map)  fig, ax = plt.subplots(1, 1, figsize=(15, 10))  divider = make\_axes\_locatable(ax)  cax = divider.append\_axes("right", size="5%", pad=0.1)  world.plot(column='TotalSales', ax=ax, legend=True, cax=cax, legend\_kwds={'label': "Total Sales"})  ax.set\_title('Geospatial Distribution of Total Sales by Country')  plt.show()  # ARIMA Model Diagnostic Plots  plt.figure(figsize=(16, 8))  # Fit ARIMA model  model = ARIMA(df.set\_index('InvoiceDate')['TotalSales'], order=(5, 1, 2))  fit\_model = model.fit()  # Plot diagnostics  fit\_model.plot\_diagnostics(figsize=(14, 10))  plt.show()  # Multivariate Time Series Plot with Subplots  plt.figure(figsize=(16, 12))  # Set up subplots  ax1 = plt.subplot(2, 1, 1)  ax2 = plt.subplot(2, 1, 2, sharex=ax1)  # Plot Total Sales  ax1.plot(df.set\_index('InvoiceDate')['TotalSales'], label='Total Sales', color='blue')  ax1.set\_title('Total Sales Over Time')  ax1.legend()  # Plot Quantity Sold  ax2.plot(df.set\_index('InvoiceDate')['Quantity'], label='Quantity Sold', color='green')  ax2.set\_title('Quantity Sold Over Time')  ax2.legend()  # Format x-axis to show months  ax2.xaxis.set\_major\_locator(mdates.MonthLocator())  ax2.xaxis.set\_major\_formatter(mdates.DateFormatter('%b %Y'))  plt.tight\_layout()  plt.show()  # Pairplot for Key Numerical Features  numerical\_features = ['Quantity', 'Price', 'TotalSales']  # Subsample the data for better visualization (optional)  subset\_data = df.sample(1000)  # Plotting the pairplot  sns.pairplot(subset\_data, vars=numerical\_features, hue='Country', markers='.', plot\_kws={'alpha':0.5})  plt.suptitle('Pairplot for Key Numerical Features')  plt.show()  """#### Dataset Correlation Analysis"""  # Display the correlation matrix  correlation\_matrix = df.corr()  correlation\_matrix  """### Dataset Cleaning"""  # drop null values  df.dropna(inplace=True)  df.isna().sum()  df[df.duplicated()].shape  df = df.drop\_duplicates(keep='first')  df.shape  # invoice with 'C' means cancelled  df[df['Invoice'].str.contains('C')]  # Drop data with "C" in Invoice  df = df[~df['Invoice'].str.contains('C')]  df.shape  # checking if any data row with price as 0  df[df['Price']==0]  # Remove rows with Price as 0  df = df[df['Price']>0]  df.shape  """### RFM - Recency, Frequency, and Monetary"""  # Calculate Recency, Frequency, and Monetary values for each customer  # Set a reference date for recency calculation  reference\_date = df['InvoiceDate'].max() + dt.timedelta(days=1)  # Calculate Recency, Frequency, and Monetary values  rfm\_data = df.groupby('Customer ID').agg({  'InvoiceDate': lambda InvoiceDate: (reference\_date - InvoiceDate.max()).days,  'Invoice': lambda Invoice: Invoice.nunique(),  'TotalSales': lambda TotalSales: TotalSales.sum()  }).rename(columns={  'InvoiceDate': 'Recency',  'Invoice': 'Frequency',  'TotalSales': 'Monetary'  })  rfm\_data.head()  rfm\_data.describe()  """### Customer Segmentation - Using Scoring RFM Metrics"""  cs\_rfm\_data = rfm\_data.copy()  # Check for Outliers using Boxplots  plt.figure(figsize=(16, 5))  # Recency  plt.subplot(1, 3, 1)  sns.boxplot(x=cs\_rfm\_data['Recency'], color='#baf808')  plt.title('Recency')  # Frequency  plt.subplot(1, 3, 2)  sns.boxplot(x=cs\_rfm\_data['Frequency'], color='#baf808')  plt.title('Frequency')  # Monetary  plt.subplot(1, 3, 3)  sns.boxplot(x=cs\_rfm\_data['Monetary'], color='#baf808')  plt.title('Monetary')  plt.show()  # Replace outliers with the Bound values  for col in cs\_rfm\_data.columns:  # calculate Q1 and Q3  Q1 = cs\_rfm\_data[col].quantile(0.25)  Q3 = cs\_rfm\_data[col].quantile(0.75)  # Calculate IQR  IQR = Q3 - Q1  # Identify outliers using boolean masks  lower\_bound\_mask = (cs\_rfm\_data[col] < (Q1 - 1.5 \* IQR))  upper\_bound\_mask = (cs\_rfm\_data[col] > (Q3 + 1.5 \* IQR))  # Replace outliers with lower and upper bounds  cs\_rfm\_data[col] = cs\_rfm\_data[col].mask(lower\_bound\_mask, Q1 - 1.5 \* IQR)  cs\_rfm\_data[col] = cs\_rfm\_data[col].mask(upper\_bound\_mask, Q3 + 1.5 \* IQR)  # Create quantiles for Recency, Frequency, and Monetary value  r\_bins = np.linspace(0, cs\_rfm\_data['Recency'].max()+1, 6, dtype=int)  f\_bins = np.linspace(0, cs\_rfm\_data['Frequency'].max()+1, 6, dtype=int)  m\_bins = np.linspace(0, cs\_rfm\_data['Monetary'].max()+1, 6, dtype=int)  # Assign R, F, and M scores to each customer based on their quartile  cs\_rfm\_data['R'] = pd.cut(cs\_rfm\_data['Recency'], bins=r\_bins, labels=[5, 4, 3, 2, 1])  cs\_rfm\_data['F'] = pd.cut(cs\_rfm\_data['Frequency'], bins=f\_bins, labels=[1, 2, 3, 4, 5])  cs\_rfm\_data['M'] = pd.cut(cs\_rfm\_data['Monetary'], bins=m\_bins, labels=[1, 2, 3, 4, 5])  # Create the RFM score using the R, F, and M scores.  cs\_rfm\_data["RFM\_Score"] = cs\_rfm\_data["R"].astype(str) + cs\_rfm\_data["F"].astype(str) + cs\_rfm\_data["M"].astype(str)  cs\_rfm\_data.head()  cs\_rfm\_data["RF\_Score"] = cs\_rfm\_data["R"].astype(str) + cs\_rfm\_data["F"].astype(str)  cs\_rfm\_data.head()  seg\_map = {  r'[1-2][1-2]': 'Hibernating',  r'[1-2][3-4]': 'At Risk',  r'[1-2]5': 'Can\'t Lose',  r'3[1-2]': 'About to Sleep',  r'33': 'Need Attention',  r'[3-4][4-5]': 'Loyal Customers',  r'41': 'Promising',  r'51': 'New Customers',  r'[4-5][2-3]': 'Potential Loyalists',  r'5[4-5]': 'Champions'  }  cs\_rfm\_data["segment"] = cs\_rfm\_data["RF\_Score"].replace(seg\_map, regex = True)  cs\_rfm\_data.head()  cs\_rfm\_data['segment'].value\_counts(normalize=True)  !pip install kneed  from sklearn.cluster import KMeans  from sklearn.preprocessing import StandardScaler  from kneed import KneeLocator  kmean\_clus = rfm\_data.copy()  # Replace outliers with the Bound values  for col in kmean\_clus.columns:  # calculate Q1 and Q3  Q1 = kmean\_clus[col].quantile(0.25)  Q3 = kmean\_clus[col].quantile(0.75)  # Calculate IQR  IQR = Q3 - Q1  # Identify outliers using boolean masks  lower\_bound\_mask = (kmean\_clus[col] < (Q1 - 1.5 \* IQR))  upper\_bound\_mask = (kmean\_clus[col] > (Q3 + 1.5 \* IQR))  # Replace outliers with lower and upper bounds  kmean\_clus[col] = kmean\_clus[col].mask(lower\_bound\_mask, Q1 - 1.5 \* IQR)  kmean\_clus[col] = kmean\_clus[col].mask(upper\_bound\_mask, Q3 + 1.5 \* IQR)  scaler = StandardScaler()  scaled\_features = scaler.fit\_transform(kmean\_clus)  scaled\_df = pd.DataFrame(scaled\_features, columns=kmean\_clus.columns)  scaled\_df.info()  # Elbow method to find the optimal number of clusters  inertia = []  for k in range(1, 10):  kmeans = KMeans(n\_clusters=k, max\_iter=300, random\_state=42)  kmeans.fit(scaled\_features)  inertia.append(kmeans.inertia\_)  plt.figure(figsize=(10, 5))  ax = sns.pointplot(x=list(range(1, 10)), y=inertia)  ax.set\_title('K-Means Clustering\nNumber of Clusters vs Inertia')  ax.set\_xlabel('Number of Clusters')  ax.set\_ylabel('Inertia')  plt.show()  # Identifying the optimal number of clusters using KneeLocator  kl = KneeLocator(range(1, 10), inertia, curve='convex', direction="decreasing")  optimal\_clusters = kl.elbow  optimal\_clusters  # Applying K-means clustering with the optimal number of clusters  kmeans\_optimal = KMeans(n\_clusters=optimal\_clusters, max\_iter=300, random\_state=42)  cluster\_labels = kmeans\_optimal.fit\_predict(scaled\_features)  # Creating a DataFrame with cluster labels  cluster\_df = pd.DataFrame(cluster\_labels, index=rfm\_data.index, columns=['Cluster\_Labels'])  cluster\_df.head()  # Joining the cluster labels with the original data  clustered\_rfm = rfm\_data.join(cluster\_df)  clustered\_rfm.head()  # Displaying the distribution of customers across clusters  cluster\_distribution = clustered\_rfm['Cluster\_Labels'].value\_counts(normalize=True)  cluster\_distribution  clus\_mean = clustered\_rfm.groupby('Cluster\_Labels').mean()  fig, axes = plt.subplots(nrows=2, ncols=2, figsize=(15, 10))  for i, col in enumerate(clus\_mean.columns):  ax = axes[i // 2, i % 2] # Adjusting subplot indexing  sns.barplot(x=clus\_mean.index, y=clus\_mean[col], data=clus\_mean, color='#4eaf4e', ax=ax)  ax.set\_title(col, fontweight='bold', fontsize=16)  plt.tight\_layout()  plt.show()  clus\_mean |

### Data Visualization

### **Distribution of Quantity Ordered**



This diagram shows in most of the orders quantity of stockcode ordered by customer are low.

### **Distribution of Sales**

A graph of a bar graph

Description automatically generated with medium confidence

This diagram shows sales value range is low which also demonstrates the products we are selling have a low price.

### **Distribution of Transactions Per Customer**

A graph with numbers and lines

Description automatically generated

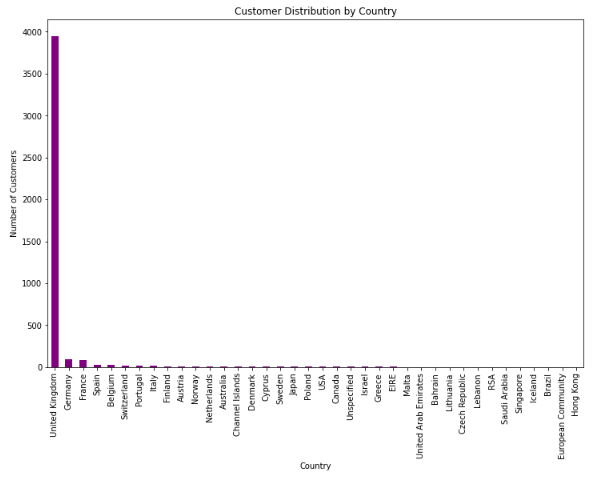
This Diagrams illustrate there are very few customers whose orders are more than 25 in 2010-11 time period.

### **Monthly Active Customers**



This diagram illustrates customers get more active towards the end of year.

### **Customers Distribution by Country**



This diagram illustrates Maximum sales order are from UK while rest 27 countries contributes very little.

### **Monthly Sales Trend**

A green line graph with white text

Description automatically generated

This diagram illustrates sales per month as customers get more active towards the end of year so the sale.

### **Correlation Heatmap**

A diagram of a heatmap

Description automatically generated

This diagram Shows the correlation factor of each column with every other column in the dataset including that column as well.

### **Top Selling Products**

A bar graph with blue and white bars

Description automatically generated

This horizontal bar chart illustrate top 10 stock code like 22197 stock code is the top selling product

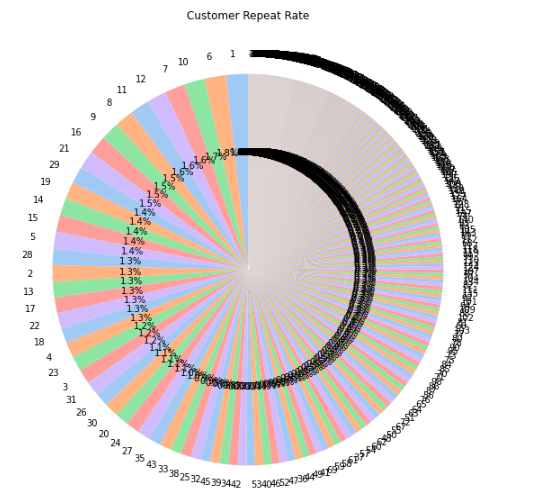
### **Invoice Count by Hour of the Day**

A graph with purple lines

Description automatically generated

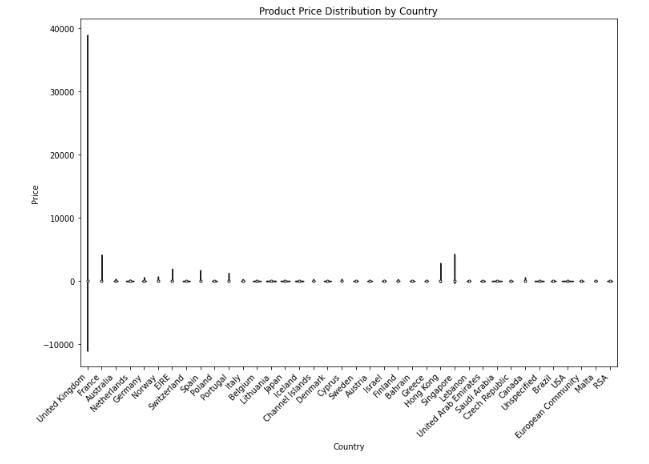
This image illustrates frequency of order with respect to hour of the day. As per diagram peak hours are between 10 to 14.

### **Customer Repeat Rate**



This diagram illustrates repeat rate of a customer to place a new order. Like highest repeat rate is around 2% as per pie chart.

### **Product Price Distribution by Country**



This diagram illustrates products stock code price variation with respect to country. Like 99% of invoice are from UK that’s why UK distribution show the highest variation.

### **Geospatial Distribution of Total Sales by Country**

A map of the world

Description automatically generated

This diagram highlights the countries customers order from in geospatial distribution.

### **Total Sales Over Time**

A screenshot of a computer screen

Description automatically generated

This diagram highlights quantity ordered and sales distribution from dec 2010 till dec 2011.

### **Pair Plot for Key Numerical Features**

A graph of a number of data

Description automatically generated with medium confidence

These diagrams illustrate the pair plot of numerical columns in the dataset. Like first three graph represent quantity vs quantity then quantity order and price we get and finally quantity order vs total sale we gain.