

**Assessment Report**

**on**

**“Customer Segmentation in E-commerce”**

**submitted as partial fulfillment for the award of**

**BACHELOR OF TECHNOLOGY**

**DEGREE**

**SESSION 2024-25**

**in**

**Name of discipline**

**By**

**Yogesh Kumar Verma (202401100300289)**

**Under the supervision of**

**“MR.ABHISHEK SHUKLA”**

**KIET Group of Institutions, Ghaziabad**

**Affiliated to**

**Dr. A.P.J. Abdul Kalam Technical University, Lucknow**

**(Formerly UPTU)**

**May,2025**

Introduction

* **Customer Segmentation in E-commerce**
* In today’s data-driven e-commerce industry, businesses serve a vast number of customers with diverse behaviors and preferences. However, not all customers contribute equally to the company’s revenue. Some may shop frequently and spend a lot, while others may purchase rarely or only once. To target customers more effectively, businesses must understand and analyze these differences. This is where **customer segmentation** becomes essential.
* **Customer segmentation** is the process of dividing a company’s customer base into distinct groups or "segments" based on common characteristics. In e-commerce, these characteristics often relate to **purchasing behavior**, such as:
* **How often** a customer makes purchases
* **How recently** they have shopped
* **How much** money they typically spend
* By identifying these segments, companies can:
* Design **personalized marketing campaigns**
* Improve **customer retention**
* Enhance **customer satisfaction**
* Offer **targeted promotions**
* Increase **overall profitability**
* **🎯 Objective of the Case Study**
* The main objective of this project is to apply **data analytics and machine learning techniques** to perform customer segmentation based on purchasing behavior. We use a well-known marketing model called **RFM Analysis**, which segments customers by examining:
* **Recency** – How recently a customer made a purchase
* **Frequency** – How often they make purchases
* **Monetary** – How much money they spend
* Using these three metrics, we create a profile for each customer. Then, we apply **KMeans clustering**, an unsupervised machine learning algorithm, to group customers into similar clusters.
* **📊 Real-World Relevance**
* This kind of segmentation is widely used by top e-commerce platforms like Amazon, Flipkart, and others to:
* Identify VIP or loyal customers
* Re-engage inactive customers
* Recognize potential churn
* Find discount-driven customers
* Optimize inventory and advertising efforts
* **🔧 What We Do in This Project**
* **Load and clean the customer transaction data**
* **Compute RFM metrics for each customer**
* **Normalize the data for clustering**
* **Use the Elbow Method to choose the optimal number of clusters**
* **Apply KMeans clustering to group similar customers**

Methodology

The methodology outlines the step-by-step process followed to achieve customer segmentation using transactional data. The approach is primarily based on **RFM analysis** combined with **K-Means clustering** to group customers based on their purchasing behavior.

**📁 1. Data Collection**

* The dataset used for this project is a transactional log of an e-commerce company.
* It includes features like InvoiceNo, StockCode, Description, Quantity, InvoiceDate, UnitPrice, CustomerID, and Country.

**🧹 2. Data Preprocessing**

* **Handling missing values**: Rows with missing CustomerID are removed, as they cannot be uniquely identified.
* **Data type conversion**: The InvoiceDate column is converted to datetime format.
* **Feature engineering**: A new column TotalPrice is created using Quantity × UnitPrice to represent the total spending per transaction.

**📊 3. RFM Analysis**

To evaluate customer value, we compute:

* **Recency**: Days since the customer's last purchase.
* **Frequency**: Number of unique purchases made (Invoice count).
* **Monetary**: Total amount spent by the customer.

These values are calculated by grouping data by CustomerID.

**⚖️ 4. Data Normalization**

* The Recency, Frequency, and Monetary values are scaled using **StandardScaler** to ensure all features contribute equally to clustering.
* This step is necessary because the three features are on different scales (e.g., days vs. dollar amounts).

**🔍 5. Optimal Cluster Selection (Elbow Method)**

* The **Elbow Method** is used to determine the ideal number of clusters (k) by plotting the **Within-Cluster Sum of Squares (WCSS)**.
* The point where the WCSS starts to level off is selected as the optimal number of clusters.

**📌 6. K-Means Clustering**

* The **KMeans algorithm** is applied to the normalized RFM data using the chosen value of k.
* Each customer is assigned to a cluster, forming distinct customer groups based on purchasing behavior.

**📈 7. Visualization & Analysis**

* The resulting customer segments are visualized using **pair plots**, **scatter plots**, or **bar graphs** to interpret and analyze the behavioral patterns of each segment.
* This helps identify groups such as:
  + **High-value customers**
  + **Frequent buyers**
  + **Infrequent or inactive users**
  + **Bargain hunters**

**📤 8. Business Insights & Applications**

* The segmentation results can be used to create targeted marketing strategies:
  + Rewarding loyal customers
  + Re-engaging inactive ones
  + Offering deals to low-spending groups
  + Creating loyalty programs for high-value users

Code

# Step 1: Upload the CSV file

from google.colab import files

uploaded = files.upload()

# Step 2: Import necessary libraries

import pandas as pd

import numpy as np

from sklearn.cluster import KMeans

from sklearn.preprocessing import StandardScaler

import matplotlib.pyplot as plt

import seaborn as sns

# Step 3: Read the uploaded file (replace with actual file name if needed)

file\_name = list(uploaded.keys())[0]

df = pd.read\_csv(file\_name)

# Step 4: Data Cleaning

df.dropna(subset=['CustomerID'], inplace=True)

df['InvoiceDate'] = pd.to\_datetime(df['InvoiceDate'])

df['TotalPrice'] = df['Quantity'] \* df['UnitPrice']

# Step 5: Create RFM (Recency, Frequency, Monetary) table

snapshot\_date = df['InvoiceDate'].max() + pd.Timedelta(days=1)

rfm = df.groupby('CustomerID').agg({

    'InvoiceDate': lambda x: (snapshot\_date - x.max()).days,  # Recency

    'InvoiceNo': 'nunique',                                   # Frequency

    'TotalPrice': 'sum'                                       # Monetary

}).reset\_index()

rfm.columns = ['CustomerID', 'Recency', 'Frequency', 'Monetary']

# Step 6: Normalize RFM features

scaler = StandardScaler()

rfm\_scaled = scaler.fit\_transform(rfm[['Recency', 'Frequency', 'Monetary']])

# Step 7: Elbow Method to find optimal number of clusters

wcss = []

for i in range(1, 11):

    kmeans = KMeans(n\_clusters=i, random\_state=42)

    kmeans.fit(rfm\_scaled)

    wcss.append(kmeans.inertia\_)

# Plot the elbow graph

plt.figure(figsize=(8, 4))

plt.plot(range(1, 11), wcss, marker='o')

plt.title('Elbow Method for Optimal k')

plt.xlabel('Number of Clusters')

plt.ylabel('WCSS')

plt.grid(True)

plt.show()

# Step 8: Apply KMeans (choose k=4 for example)

kmeans = KMeans(n\_clusters=4, random\_state=42)

rfm['Cluster'] = kmeans.fit\_predict(rfm\_scaled)

# Step 9: Visualize Clusters

sns.pairplot(rfm, hue='Cluster', palette='tab10', diag\_kind='kde')

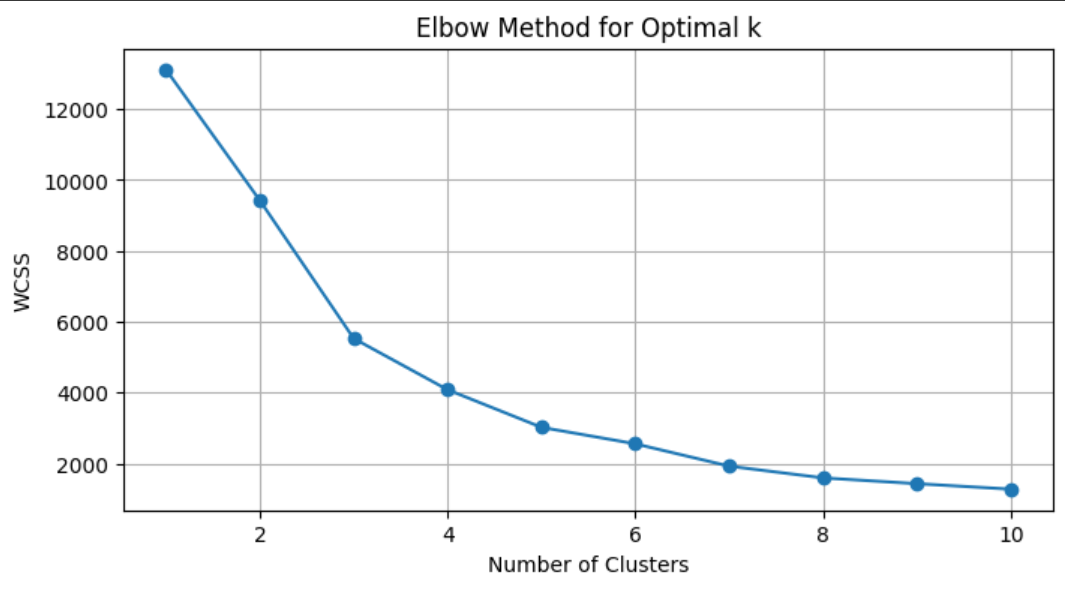
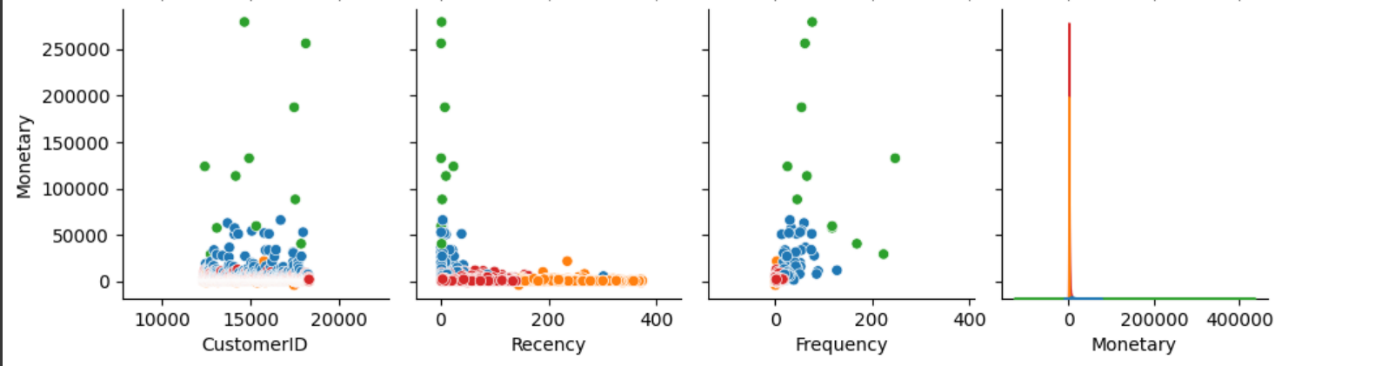
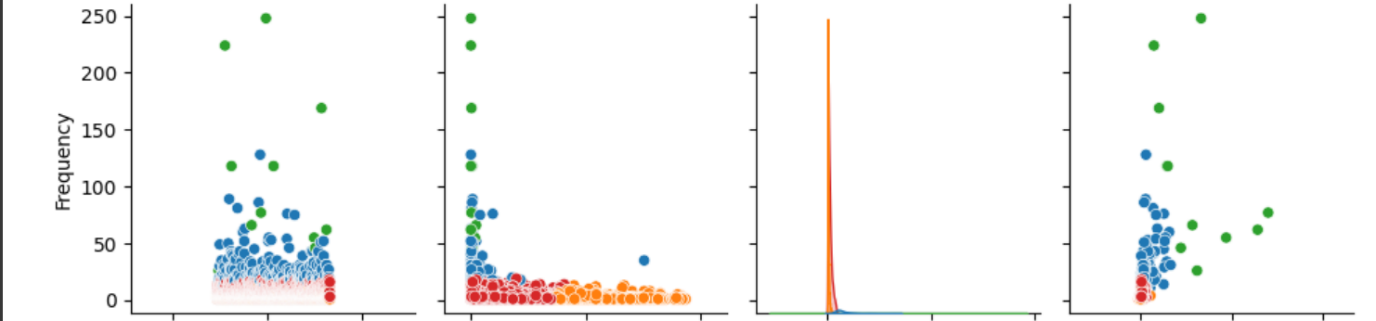
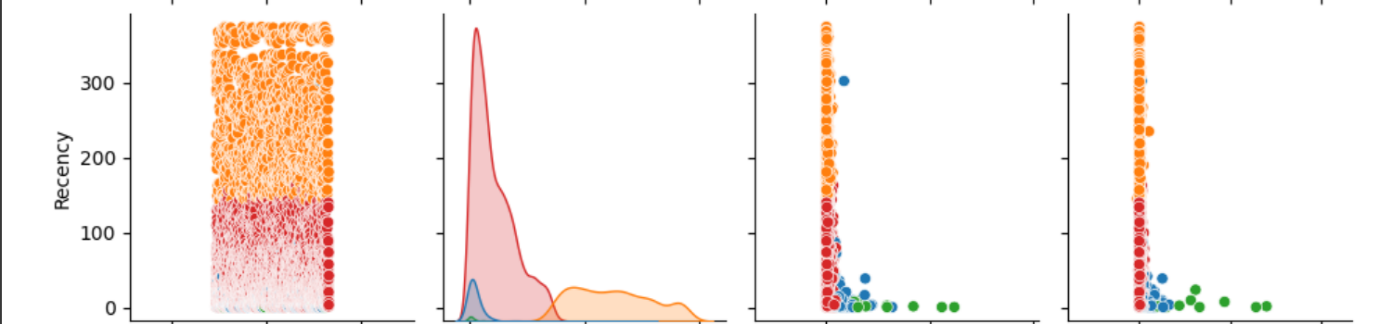
plt.suptitle("Customer Segments by RFM", y=1.02)

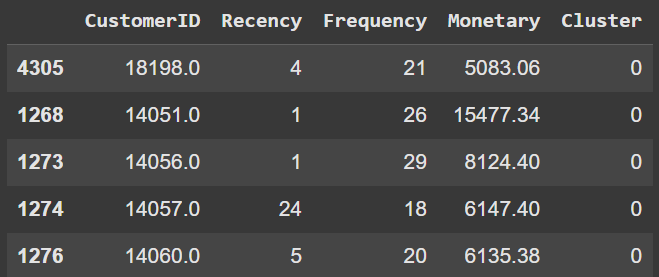
plt.show()

# Optional: Display RFM table with clusters

rfm.sort\_values('Cluster').head()

Output



References

* Hughes, A. M. (1994). *Strategic Database Marketing*.
* Kumar, V., & Reinartz, W. (2016). *Customer Relationship Management*. Springer.
* Jain, A. K. (2010). *Data clustering: 50 years beyond K-means*. Pattern Recognition Letters.
* Han, J., Kamber, M., & Pei, J. (2011). *Data Mining: Concepts and Techniques*.
* Scikit-learn Documentation: <https://scikit-learn.org>
* Kaggle - *Online Retail Dataset*: https://www.kaggle.com/datasets/vijayuv/onlineretail
* UCI Machine Learning Repository: <https://archive.ics.uci.edu/ml>
* Seaborn, Matplotlib, Pandas – Official Python library docs