**HỌC VIỆN HÀNG KHÔNG VIỆT NAM**

**KHOA CÔNG NGHỆ THÔNG TIN**

----- 📖 -----

# BÁO CÁO TIỂU LUẬN

# LẬP TRÌNH PYTHON

## PHÂN KHÚC KHÁCH HÀNG DỰA TRÊN MÔ HÌNH RFM VÀ THUẬT TOÁN KMEANS

**HỌC KỲ 2 – NĂM HỌC: 2023 - 2024**

**MÃ LỚP HỌC PHẦN: 010100087202**

**Giảng viên hướng dẫn: ThS.Nguyễn Thanh Hiếu**

**Nhóm sinh viên thực hiện: Nhóm 3**

| *Họ tên* | *Mã sinh viên* |
|---|---|
| 1. Lê Cao Tấn Lộc | 2154810086 |
| 2. Nguyễn Thị Thúy Hà | 2154810061 |
| 3. Trịnh Vinh Qui | 2154810110 |
| 4. Phan Tường Bảo Trâm | 2154810077 |
| 5. Nguyễn Nhật Hoàng | 2154810019 |

**TPHCM, tháng 3 năm 2024**

NHẬN XÉT CỦA GIẢNG VIÊN:

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

.....

Điểm: Bằng số Bằng chữ

Chữ ký của giảng viên

# MỤC LỤC

# CHƯƠNG I. GIỚI THIỆU

## 1. Giới thiệu đề tài

Trong thời đại số hóa ngày nay, thị trường bán lẻ trực tuyến đang trỗi dậy với sự phát triển nhanh chóng của công nghệ và internet. Để nắm bắt cơ hội trong môi trường kinh doanh này, việc hiểu rõ hơn về hành vi của khách hàng và tối ưu hóa chiến lược tiếp thị là rất quan trọng. Việc hiểu sâu hơn về khách hàng và đáp ứng đúng nhu cầu của họ là mục tiêu hàng đầu của mọi doanh nghiệp. Vì thế, việc phân loại khách hàng thành các nhóm riêng biệt và tạo ra các chiến lược phù hợp là điều cần thiết.

Ngoài ra, việc sử dụng khoa học dữ liệu và các công cụ phân tích dữ liệu là vô cùng quan trọng đối với doanh nghiệp đang kinh doanh ở mọi lĩnh vực cũng như các nhà buôn bán sỉ lẻ. Nhóm sẽ chuyển đổi dữ liệu giao dịch thành dữ liệu tập trung vào khách hàng bằng cách tạo ra các thông tin mới để chia khách hàng thành các nhóm khác nhau bằng phương pháp phân cụm K-means và một số phương pháp phân tích. Trong đề tài này, nhóm sẽ phát triển một hệ thống gợi ý sản phẩm sẽ đề xuất những mặt hàng bán chạy nhất cho mỗi nhóm khách hàng, nhằm tăng cường hiệu quả tiếp thị và doanh số bán hàng.

## 2. Lý do chọn đề tài

Lựa chọn đề tài này bắt nguồn từ việc nhận thức sâu sắc về sự phát triển đáng kể của thị trường bán lẻ trực tuyến và tầm quan trọng của việc hiểu biết khách hàng. Phân tích sâu về dữ liệu giao dịch không chỉ giúp nhóm khám phá các xu hướng và sở thích của khách hàng mà còn tạo ra cơ hội để tối ưu hóa chiến lược tiếp thị và tăng cường doanh số bán hàng. Bằng cách sử dụng công nghệ phân tích dữ liệu như K-means clustering, nhóm hy vọng có thể mở ra những khía cạnh mới trong việc cải thiện hiểu biết về khách hàng và phát triển mô hình kinh doanh.

Nhóm cũng sẽ tập trung vào việc phát triển một hệ thống đề xuất sản phẩm. Điều này giúp gợi ý những mặt hàng phổ biến nhất cho từng nhóm khách hàng. Qua đó, không chỉ tối ưu hóa trải nghiệm mua sắm của khách hàng mà còn nâng cao hiệu suất kinh doanh của doanh nghiệp trong một môi trường cạnh tranh như thị trường bán lẻ trực tuyến.

## 3. Ứng dụng thực tiễn

Đề tài này cũng có thể được ứng dụng rộng rãi trong nhiều lĩnh vực khác nhau của thị trường và doanh nghiệp. Hiểu về hành vi mua hàng của khách hàng có thể giúp doanh nghiệp tùy chỉnh trải nghiệm khách hàng, từ đó tạo ra cảm giác hài lòng và trung thành. Ngoài ra, phân tích dữ liệu cũng giúp dự báo xu hướng thị trường, điều này giúp doanh nghiệp chuẩn bị và điều chỉnh chiến lược kinh doanh một cách linh hoạt để đáp ứng nhu cầu thị trường và cạnh tranh hiệu quả.

Trong nhiều lĩnh vực thực tế như: Marketing và Quảng cáo, Dịch vụ khách hàng và Chăm sóc khách hàng,.vv..Các công ty, các doanh nghiệp có thể sử dụng phân khúc khách hàng để cải thiện dịch vụ và chăm sóc khách hàng,,tối ưu hóa chiến lược giá cả,...Ngoài ra, cũng có thể được sử dụng để phát triển sản phẩm mới, tạo ra cơ hội mở rộng thị trường và tiếp cận các nhóm khách hàng mới một cách hiệu quả. Họ có thể tập trung vào việc phát triển các sản phẩm và dịch vụ độc đáo để thu hút và giữ chân khách hàng. Tóm lại, việc áp dụng thuật toán phân cụm K-means vào phân khúc khách hàng có thể mang lại nhiều lợi ích thực tiễn cho doanh nghiệp, từ tối ưu hóa chiến lược tiếp thị đến việc cải thiện dịch vụ và phát triển sản phẩm.

## 4. Mục tiêu đạt được

Mục tiêu của đề tài là tăng cường hiệu suất kinh doanh và cải thiện trải nghiệm mua sắm trực tuyến cho khách hàng. Nhóm mong muốn phát triển một hệ thống đề xuất sản phẩm chính xác, giúp gợi ý các mặt hàng phổ biến nhất cho từng nhóm khách hàng. Điều này không chỉ tối ưu hóa trải nghiệm mua sắm mà còn nâng cao hiệu suất kinh doanh và tạo ra một môi trường mua sắm trực tuyến cá nhân hóa và thú vị hơn cho người tiêu dùng. Đồng thời, nhóm hy vọng mang lại lợi ích lâu dài và tăng cường sự cạnh tranh của doanh nghiệp trong thị trường bán lẻ trực tuyến ngày nay.

# CHƯƠNG II. LÝ THUYẾT/ CÔNG NGHỆ ỨNG DỤNG

## 1. Công nghệ ứng dụng

### 1.1. Python

Python là ngôn ngữ lập trình khá phổ biến hiện nay và được sử dụng rộng rãi trong các lĩnh vực lập trình, phát triển web, machine learning và data science. Python là ngôn ngữ lập trình được các lập trình viên lựa chọn sử dụng, vì thế không quá khó hiểu khi Python đã vượt qua Java - ngôn ngữ lập trình hàng đầu. Do tính phổ biến và khả năng chạy trên gần như mọi kiến trúc hệ thống, Python là một ngôn ngữ phổ quát được tìm thấy trong nhiều ứng dụng khác nhau. Python hỗ trợ các mô-đun và packages khác nhau, cho phép sử dụng mô đun chương trình và tái sử dụng mã. Nhờ vào tính linh hoạt cùng với sự thân thiện dành cho người mới bắt đầu, đã khiến Python trở thành một trong những ngôn ngữ lập trình được sử dụng nhiều nhất hiện nay.

Một số lợi ích mà ngôn ngữ Python mang lại:

- Dễ học và dễ sử dụng: Python có cú pháp đơn giản, dễ hiểu và dễ học, giúp người mới bắt đầu có thể tiếp cận nhanh chóng. Hơn nữa, Python có một loạt các thư viện và module tiêu chuẩn, giúp người lập trình tiết kiệm thời gian và công sức khi phát triển ứng dụng.
- Đa năng: Python được sử dụng rộng rãi trong nhiều lĩnh vực như khoa học dữ liệu, trí tuệ nhân tạo, phát triển web, tự động hóa, phân tích và xử lý văn bản,...
- Mã nguồn mở: Python cho phép người dùng có thể sử dụng, chỉnh sửa và phân phối mã nguồn một cách tự do.
- Hiệu suất cao: Mặc dù Python không phải là ngôn ngữ lập trình nhanh nhất, nhưng với sự phát triển của các thư viện như NumPy, Pandas và TensorFlow, Python đã trở thành một trong những ngôn ngữ lập trình hiệu suất cao.
- Hỗ trợ cộng đồng: Python có một cộng đồng lớn với nhiều người dùng và nhà phát triển trên khắp thế giới đóng góp vào việc phát triển và cải tiến Python. Cộng đồng này cũng cung cấp hỗ trợ và tài liệu để giúp người dùng Python giải quyết các vấn đề một cách nhanh chóng và hiệu quả.

### 1.2. Thư viện sử dụng

- Xử lý dữ liệu và mô hình hoá

Python cung cấp hai thư viện chính để xử lý dữ liệu đó là Pandas và NumPy. Trong đó:
- Pandas cung cấp cho chúng ta một số công cụ hữu ích nhất để khám phá, làm sạch và phân tích dữ liệu. Với Pandas, lập trình viên có thể dễ dàng, nhanh chóng chỉnh sửa, tổng hợp và trực quan hoá dữ liệu. Thư viện này cho phép chúng ta thao tác với các bảng số và chuỗi thời gian bằng cách sử dụng các cấu trúc và phép toán dữ liệu.
- NumPy là thư viện được ứng dụng chủ yếu trong phân tích dữ liệu, tính toán dữ liệu. Thư viện này hỗ trợ việc tính toán các mảng đa chiều, có kích thước lớn và là một trong số thư viện khoa học dữ liệu chủ chốt trong Python. Ngoài ra, NumPy còn đặc biệt hữu ích khi thực hiện các hàm liên quan tới đại số tuyến tính.

- Trực quan hoá dữ liệu

Trực quan hoá dữ liệu là việc trình bày thông tin định lượng (những danh sách các chữ số dài) dưới dạng đồ thị, giúp người xem dễ hiểu và xử lý hơn. Hai thư viện hỗ trợ được sử dụng rộng rãi để trực quan hoá dữ liệu Python chính là Matplotlib và Seaborn. Trong đó:
- Matplotlib cho phép người dùng tạo biểu đồ nhiều dạng, đồ thị theo thời gian và những đồ thị thông số chuyên nghiệp khác. Với Matplotlib, người dùng có thể tùy chỉnh mọi khía cạnh của số liệu, và thư viện này cũng sở hữu nhiều tính năng tương tác như phóng to/thu nhỏ. lên kế hoạch và lưu biểu đồ dưới dạng đồ hoạ.
- Seaborn là thư viện mở rộng được xây dựng trên nền tảng Matplotlib, giúp cho người dùng dễ dàng trực quan hoá dữ liệu chỉ qua một vài bước đơn giản. Với Seaborn, ta có thể thực hiện mọi tác vụ thống kê quan trọng, giúp tạo ra các biểu đồ tóm lược đầy đủ thông tin.

### 1.3. Google Colab

Google Colab, một môi trường phát triển dựa trên trình duyệt, là sản phẩm của Google nhằm hỗ trợ việc viết và chia sẻ mã nguồn Python một cách thuận lợi và không đòi hỏi cài đặt phần mềm trên máy tính cá nhân. Colab không chỉ cung cấp môi trường lập trình, mà còn cung cấp nguồn lực máy tính mạnh mẽ như GPU và TPU, giúp người dùng thực hiện các tác vụ tính toán phức tạp mà không làm chậm máy tính cá nhân của họ.

Ngôn ngữ lập trình Python, với sự đơn giản và dễ đọc, đã trở thành một trong những ngôn ngữ phổ biến nhất trong lĩnh vực khoa học dữ liệu, máy học và trí tuệ nhân tạo. Python không chỉ linh hoạt mà còn có một cộng đồng lớn, và nó được hỗ trợ bởi nhiều thư viện mạnh mẽ như NumPy, Pandas, Matplotlib và TensorFlow, giúp người lập trình xử lý và phân tích dữ liệu một cách hiệu quả.

Trong việc tích hợp giữa Google Colab và Python, Colab không chỉ miễn phí mà còn tích hợp tốt với Google Drive, cho phép lưu trữ và chia sẻ dự án dễ dàng. Colab cũng cung cấp tài nguyên GPU và TPU miễn phí, giúp tăng tốc độ tính toán cho các tác vụ máy học và deep learning.


## 2. Cơ sở lý thuyết

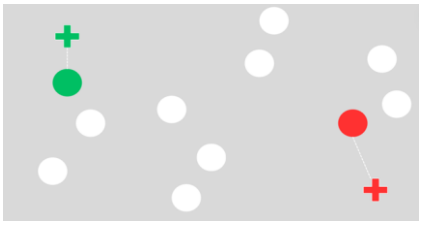
### 2.1. Thuật toán KMeans

Thuật toán phân cụm k-means là một phương pháp được sử dụng trong phân tích tính chất cụm của dữ liệu. Nó đặc biệt được sử dụng nhiều trong khai phá dữ liệu và thống kê. Nó phân vùng dữ liệu thành k cụm khác nhau. Giải thuật này giúp chúng ta xác định được dữ liệu của chúng ta nó thực sự thuộc về nhóm nào.

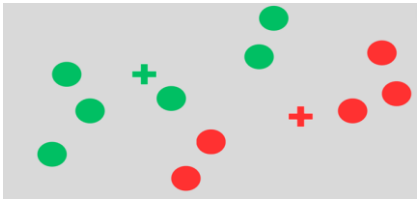
- Các bước thực hiện thuật toán

– Khởi Tạo Trung Tâm: Bắt đầu bằng cách chọn ngẫu nhiên K điểm từ tập dữ liệu làm trung tâm của các cụm.

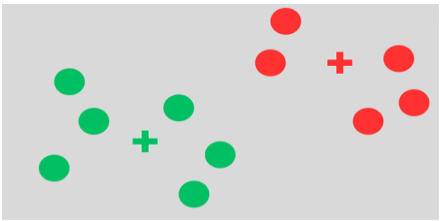
– Phân Cụm: Gán mỗi điểm dữ liệu vào cụm có trung tâm gần nhất. Điều này thường được đo bằng khoảng cách Euclidean.



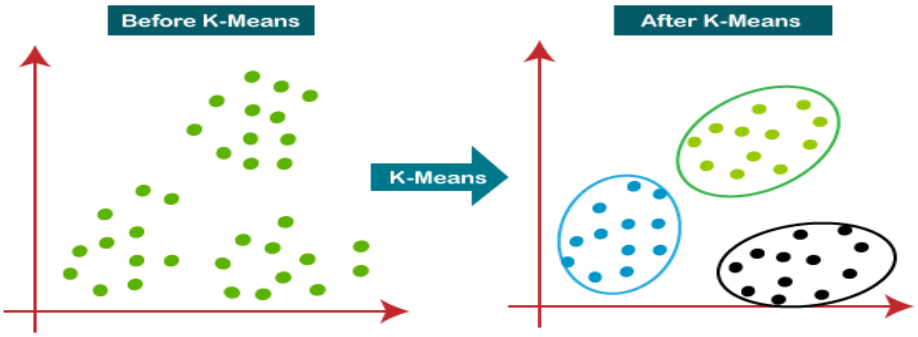
– Cập Nhật Trung Tâm: Tính toán trung tâm mới của mỗi cụm bằng cách lấy trung bình của tất cả các điểm dữ liệu thuộc cụm đó.



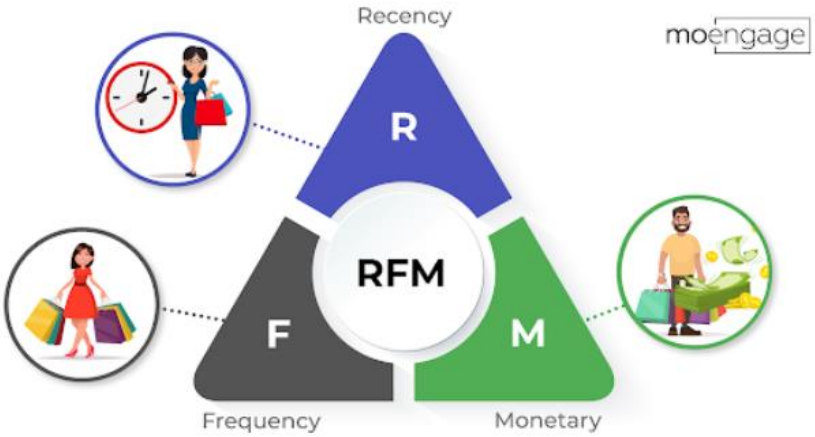
– Lặp Lại: Bước b, c được lặp lại cho đến khi không có sự thay đổi đáng kể nào trong việc gán nhãn của các điểm dữ liệu.



- **Ví dụ cho thuật toán Kmeans:**



## 2.2. Mô hình RFM




Mô hình RFM (Recency, Frequency, Monetary) là một mô hình phân tích được ứng dụng trong quy trình phân tích giá trị khách hàng và phân khúc cơ sở khách hàng. Mô hình RFM đánh giá các khía cạnh quan trọng của hành vi mua hàng của khách hàng dựa trên 3 yếu tố:

- Recency (Tần suất mua hàng):

Recency: Khoảng thời gian giữa giao dịch gần nhất tới hiện tại là bao lâu? Khoảng thời gian này càng lớn thì khách hàng càng có khả năng cao rời bỏ, đồng nghĩa với việc cửa hàng sẽ mất càng nhiều thời gian để chăm sóc và thuyết phục họ quay lại. Ngược lại, khoảng thời gian này càng nhỏ thì khả năng tiếp cận và upsell, cross-sell sẽ càng cao.

- Frequency (Tần suất mua hàng):

Frequency: Khách hàng mua hàng càng thường xuyên sẽ càng có khả năng phản hồi với các chiến dịch của thương hiệu và trở thành khách hàng trung thành. Khách hàng mua sản phẩm hàng tuần hoặc hàng tháng thì khả năng bán upsell sẽ cao hơn những khách hàng 3 tháng mua hàng 1 lần. Tần suất giao dịch cũng đo lường mức độ mối quan hệ của khách hàng với doanh nghiệp.

- Monetary (Giá trị mua hàng):

Monetary: Mức độ chi tiêu sẽ cho biết khả năng chỉ tiêu của khách hàng cho thương hiệu nằm ở đâu. Dựa vào con số này và tần suất giao dịch, bạn cũng có thể tính được chỉ tiêu trung bình của mỗi giao dịch của khách hàng. Thông thường, những khách hàng đã mua nhiều lần, có sự tin tưởng nhất định vào thương hiệu mới có thể mạnh tay mua những đơn giá trị cao.

- **Ví dụ thực tế của mô hình RFM:**

Công ty bán lẻ thời trang X muốn phân tích hành vi mua hàng của khách hàng để triển khai chương trình marketing hiệu quả. Công ty sử dụng mô hình RFM để đánh giá khách hàng dựa trên dữ liệu mua hàng trong 6 tháng qua.

Dựa vào điểm RFM, công ty phân loại khách hàng thành 5 nhóm:

1. Khách hàng VIP (20%): Mua hàng thường xuyên, giá trị cao.
2. Khách hàng trung thành (30%): Mua hàng thường xuyên, giá trị trung bình.
3. Khách hàng tiềm năng (30%): Mua hàng không thường xuyên, giá trị trung bình.
4. Khách hàng mới (10%): Mua hàng ít lần, giá trị thấp.
5. Khách hàng ngủ quên (10%): Ít mua hàng hoặc đã ngừng mua hàng.

Dựa vào phân loại này, công ty xây dựng các chiến lược marketing phù hợp:

- Khách hàng VIP: Gửi ưu đãi đặc biệt, quà tặng cao cấp, chương trình tri ân khách hàng.
- Khách hàng trung thành: Gửi thông tin sản phẩm mới, chương trình khuyến mãi hấp dẫn.
- Khách hàng tiềm năng: Gửi email giới thiệu sản phẩm, khuyến mãi theo sở thích.
- Khách hàng mới: Gửi voucher giảm giá, chương trình chào mừng khách hàng mới.
- Khách hàng: Gửi email nhắc nhở, chương trình khuyến mãi hấp dẫn để thu hút mua hàng trở lại.

# CHƯƠNG III: SẢN PHẨM ĐỀ TÀI

## 1. Mô tả bộ dữ liệu

DataFrame chứa 541 909 dòng và 8 cột, với mỗi cột mang lại thông tin quan trọng về đơn hàng, sản phẩm, và khách hàng.

Hình ảnh 1 phần dữ liệu:

| voiceNo | StockCode | Description | Quantity | InvoiceDate | UnitPrice | CustomerID | Country |
|---|---|---|---|---|---|---|---|
| 536365 | 85123A | WHITE HANGING HE | 6 | 01/12/2010 8:26 | 2,55 | 17850 | United Kingdom |
| 536365 | 71053 | WHITE METAL LANT | 6 | 01/12/2010 8:26 | 3,39 | 17850 | United Kingdom |
| 536365 | 84406B | CREAM CUPID HEAR | 8 | 01/12/2010 8:26 | 2,75 | 17850 | United Kingdom |
| 536365 | 84029G | KNITTED UNION FLA | 6 | 01/12/2010 8:26 | 3,39 | 17850 | United Kingdom |
| 536365 | 84029E | RED WOOLLY HOTTI | 6 | 01/12/2010 8:26 | 3,39 | 17850 | United Kingdom |
| 536365 | 22752 | SET 7 BABUSHKA NE | 2 | 01/12/2010 8:26 | 7,65 | 17850 | United Kingdom |
| 536365 | 21730 | GLASS STAR FROSTE | 6 | 01/12/2010 8:26 | 4,25 | 17850 | United Kingdom |
| 536366 | 22633 | HAND WARMER UN | 6 | 01/12/2010 8:28 | 1,85 | 17850 | United Kingdom |
| 536366 | 22632 | HAND WARMER RED | 6 | 01/12/2010 8:28 | 1,85 | 17850 | United Kingdom |
| 536367 | 84879 | ASSORTED COLOUR | 32 | 01/12/2010 8:34 | 1,69 | 13047 | United Kingdom |

Sau đây là mô tả thông tin chi tiết những trường dữ liệu cụ thể, bao gồm:

- **OrderID**: Mã đặt hàng cho mỗi giao dịch riêng biệt.
- **ProductID**: Mã sản phẩm được mua trong giao dịch.
- **Description**: Mô tả chi tiết về sản phẩm được mua.
- **Quantity**: Số lượng của sản phẩm được mua trong giao dịch.
- **InvoiceDate**: Thời điểm giao dịch được thực hiện.
- **Price**: Giá của mỗi sản phẩm.
- **CustomerID**: Mã khách hàng, đại diện cho mỗi khách hàng.
- **Country**: Quốc gia mà khách hàng đến từ.

Dựa vào trên, chúng ta có thể nhận thấy rằng tập dữ liệu chứa thông tin về các giao dịch mua hàng cụ thể, bao gồm sản phẩm, số lượng, giá cả, thời gian giao dịch, thông tin về khách hàng và quốc gia. Điều này có thể giúp phân tích hành vi mua hàng của khách hàng và hiểu rõ hơn về xu hướng mua sắm và hoạt động kinh doanh của cửa hàng trực tuyến.

## 2. Tiến hành phân tích dữ liệu

### 2.1. Kết nối dữ liệu và import các thư viện

– Bắt đầu bằng việc kết nối với Google Drive thông qua Google Colab, mã nguồn sử dụng thư viện google colab để thiết lập liên kết. Bước này là quan trọng để có thể truy cập và tương tác với dữ liệu được lưu trữ trên Google Drive. Mã xác

minh được yêu cầu là một phần quan trọng của quá trình này để xác nhận quyền truy cập từ phía người dùng.

```
[11] #Colab connect drive
     from google.colab import drive
     drive.mount('/content/drive')

     Mounted at /content/drive
```

– Import các thư viện: pandas, numpy, seaborn, datetime và matplotlib.pyplot là quan trọng để thực hiện các phân tích dữ liệu và tạo đồ thị để hỗ trợ quá trình hiểu biết dữ liệu. Ngoài ra còn có một số thư viện khác.

```
# Import thu vien
import pandas as pd
import numpy as np
import datetime as dt
import seaborn as sns
import matplotlib
%matplotlib inline
import matplotlib.pyplot as plt
import matplotlib.gridspec as gridspec
import plotly.graph_objects as go
from matplotlib.colors import LinearSegmentedColormap
from matplotlib import colors as mcolors
from scipy.stats import linregress
from sklearn.ensemble import IsolationForest
from sklearn.decomposition import PCA
from sklearn.cluster import KMeans
from sklearn.preprocessing import StandardScaler
from yellowbrick.cluster import KElbowVisualizer, SilhouetteVisualizer
from sklearn.metrics import silhouette_score, calinski_harabasz_score, davies_bouldin_score
from tabulate import tabulate
from collections import Counter
from numpy import math
import warnings
warnings.simplefilter(action='ignore', category=FutureWarning)
```

– Sử dụng thư viện pandas để đọc dữ liệu từ một tệp Excel được lưu trữ trên Google Drive thông qua Google Colab.

```
✓ 1 phút
[18] #read data
     df = pd.read_excel('/content/drive/MyDrive/Dataset/data.xlsx')
```

– Để biết tổng quan về dữ liệu như số lượng dòng trong DataFrame, chúng ta thực hiện df.info() được sử dụng để hiển thị thông tin tổng quan về một Data Frame trong thư viện pandas. Thông tin này bao gồm:

- Số lượng hàng và cột trong DataFrame.
- Tên và kiểu dữ liệu của mỗi cột.
- Số lượng giá trị không bị thiếu (non-null) của mỗi cột.
- Tổng số bộ nhớ được sử dụng bởi DataFrame.

```
# 2: Initial Data Analysis: Initial Data Analysis (IDA) là quá trình khám phá và hiểu về dữ liệu trong giai đoạn ban đầu
# 2.1  Dataset Overview: Tổng quan về dữ liệu
print("DATA INFO".center(125,'-'))
print(df.info())
```

```
-----------------------------------------------------------DATA INFO-----------------------------------------------------------
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 541909 entries, 0 to 541908
Data columns (total 8 columns):
 #   Column       Non-Null Count   Dtype
---  ------       --------------   -----
 0   OderlID      541909 non-null  object
 1   ProductID    541909 non-null  object
 2   Description  540455 non-null  object
 3   Quantity     541909 non-null  int64
 4   InvoiceDate  541909 non-null  datetime64[ns]
 5   Price        541909 non-null  float64
 6   CustomerID   406829 non-null  float64
 7   Country      541909 non-null  object
dtypes: datetime64[ns](1), float64(2), int64(1), object(4)
memory usage: 33.1+ MB
None
```

– Sau khi biết được tổng quan về thông tin của tệp dữ liệu, chúng ta tiến hành bước quan trọng tiếp theo bằng cách sử dụng df.head(10). Hàm này giúp hiển thị 10 dòng đầu tiên của dữ liệu, mang lại cái nhìn tổng quan về cấu trúc và đặc điểm của dữ liệu.

```
# In thu 10 dong du lieu
df.head(10).style.set_properties(**{"background-color": "#cd5c5c","color": "black", "border-color": "black"})
```

| | OderlID | ProductID | Description | Quantity | InvoiceDate | Price | CustomerID | Country |
|---|---|---|---|---|---|---|---|---|
| 0 | 536365 | 85123A | WHITE HANGING HEART T-LIGHT HOLDER | 6 | 2010-12-01 08:26:00 | 2.550000 | 17850.000000 | United Kingdom |
| 1 | 536365 | 71053 | WHITE METAL LANTERN | 6 | 2010-12-01 08:26:00 | 3.390000 | 17850.000000 | United Kingdom |
| 2 | 536365 | 84406B | CREAM CUPID HEARTS COAT HANGER | 8 | 2010-12-01 08:26:00 | 2.750000 | 17850.000000 | United Kingdom |
| 3 | 536365 | 84029G | KNITTED UNION FLAG HOT WATER BOTTLE | 6 | 2010-12-01 08:26:00 | 3.390000 | 17850.000000 | United Kingdom |
| 4 | 536365 | 84029E | RED WOOLLY HOTTIE WHITE HEART. | 6 | 2010-12-01 08:26:00 | 3.390000 | 17850.000000 | United Kingdom |
| 5 | 536365 | 22752 | SET 7 BABUSHKA NESTING BOXES | 2 | 2010-12-01 08:26:00 | 7.650000 | 17850.000000 | United Kingdom |
| 6 | 536365 | 21730 | GLASS STAR FROSTED T-LIGHT HOLDER | 6 | 2010-12-01 08:26:00 | 4.250000 | 17850.000000 | United Kingdom |
| 7 | 536366 | 22633 | HAND WARMER UNION JACK | 6 | 2010-12-01 08:28:00 | 1.850000 | 17850.000000 | United Kingdom |
| 8 | 536366 | 22632 | HAND WARMER RED POLKA DOT | 6 | 2010-12-01 08:28:00 | 1.850000 | 17850.000000 | United Kingdom |
| 9 | 536367 | 84879 | ASSORTED COLOUR BIRD ORNAMENT | 32 | 2010-12-01 08:34:00 | 1.690000 | 13047.000000 | United Kingdom |

- Ta có được một số thông tin sau:

- DATA TYPES: Đếm số lượng các loại dữ liệu trong DataFrame.
- MISSING VALUES: Đếm số lượng giá trị thiếu (null) trong mỗi cột của Data Frame và sắp xếp chúng theo thứ tự giảm dần.
- DUPLICATE VALUES: Đếm số lượng hàng trùng lặp trong DataFrame.
- STATISTICS OF DATA: Hiển thị các thống kê mô tả của dữ liệu, bao gồm cả các biến phân loại và liên tục.

```
-----------------------------------------------------------DATA TYPES----------------------------
object           4
float64          2
int64            1
datetime64[ns]   1
dtype: int64
------------------------------------------------------ MISSING VALUES ----------------------
CustomerID     135080
Description      1454
dtype: int64
------------------------------------------------------ DUPLICATED VALUES ----------------------
5268
```

Kết quả hiển thị kiểu dữ liệu:

- Có 5 cột có kiểu dữ liệu là object.
- Có 2 cột có kiểu dữ liệu là float64.
- Có 1 cột có kiểu dữ liệu là int64.
- "Customer ID": Có 135,080 giá trị thiếu. Điều này ngụ ý rằng có 135,080 dòng trong cột "Customer ID" không có giá trị hoặc bị thiếu thông tin.
- "Description": Có 1,454 giá trị thiếu. Tương tự như trên, có 1,454 dòng trong cột "Description" không có giá trị hoặc bị thiếu thông tin.

| | count | mean | std | min | 25% | 50% | 75% | max |
|---|---|---|---|---|---|---|---|---|
| **Quantity** | 541909.0 | 9.552250 | 218.081158 | -80995.00 | 1.00 | 3.00 | 10.00 | 80995.0 |
| **Price** | 541909.0 | 4.611114 | 96.759853 | -11062.06 | 1.25 | 2.08 | 4.13 | 38970.0 |
| **CustomerID** | 406829.0 | 15287.690570 | 1713.600303 | 12346.00 | 13953.00 | 15152.00 | 16791.00 | 18287.0 |

```
[ ]  # Summary statistics for categorical variables
     df.describe(include='object').T
```

| | count | unique | top | freq |
|---|---|---|---|---|
| **OderIID** | 541909 | 25900 | 573585 | 1114 |
| **ProductID** | 541909 | 4070 | 85123A | 2313 |
| **Description** | 540455 | 4223 | WHITE HANGING HEART T-LIGHT HOLDER | 2369 |
| **Country** | 541909 | 38 | United Kingdom | 495478 |

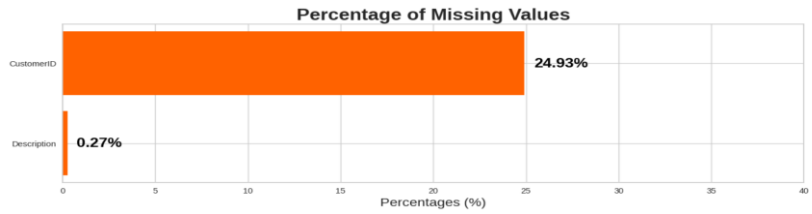
## 2.2 Tiền xử lý dữ liệu (Data Cleaning & Transformation)

Quá trình tiền xử lý dữ liệu đóng một vai trò quan trọng trong quá trình chuẩn bị dữ liệu cho các phân tích và mô hình hóa.

Data Cleaning & Transformation (làm sạch và tinh chỉnh dữ liệu): Bước này là quá trình làm sạch và biến đổi tập dữ liệu. Nhóm sẽ xử lý các giá trị thiếu, loại bỏ các mục trùng lặp, sửa các bất thường trong mã sản phẩm và mô tả, và thực hiện các điều chỉnh cần thiết để chuẩn bị dữ liệu cho phân tích và mô hình hóa. Dưới đây là mô tả chi tiết cho bước tiền xử lý dữ liệu được thực hiện trong đoạn mã:

- **Bước 1: Handling Missing Values (xử lý các giá trị thiếu)**

```
# 3: Data Cleaning & Transformation: làm sạch và tinh chỉnh tập dữ liệu
# 3.1 Handling Missing Values: Xử lý các giá trị thiếu
# Calculating the percentage of missing values for each column: Tính phần trăm giá trị thiếu cho mỗi cột
missing_data = df.isnull().sum()
missing_percentage = (missing_data[missing_data > 0] / df.shape[0]) * 100

# Prepare values: chuẩn bị
missing_percentage.sort_values(ascending=True, inplace=True)

# Plot the barh chart: Vẽ biểu đồ thanh ngang
fig,/ ax = plt.subplots(figsize=(15, 4))
ax.barh(missing_percentage.index, missing_percentage, color='#ff6200')

# Annotate the values and indexes: Ghi chú các giá trị và chỉ mục
for i, (value, name) in enumerate(zip(missing_percentage, missing_percentage.index)):
    ax.text(value+0.5, i, f"{value:.2f}%", ha='left', va='center', fontweight='bold', fontsize=18, color='black')

# Set x-axis limit: Đặt giới hạn trục x
ax.set_xlim([0, 40])

# Add title and xlabel: Thêm tiêu đề và nhãn trục x
plt.title("Percentage of Missing Values", fontweight='bold', fontsize=22)
plt.xlabel('Percentages (%)', fontsize=16)
plt.show()
```




**Ta thấy:**

**- CustomerID (chiếm 24.93%):**

- Cột CustomerID chiếm tỷ lệ hơi lớn, gần một phần tư của tập dữ liệu.
- Cột này quan trọng để phân cụm khách hàng
- Tỷ lệ lớn như vậy có thể làm cho phân tích nhiễu hoặc sai số đáng kể.
- Vì phân cụm dựa vào hành vi và sở thích của khách hàng, dữ liệu chính xác về nhận dạng khách hàng là rất quan trọng. Do đó, việc loại bỏ các hàng có CustomerID bị thiếu dường như là phương pháp hợp lý nhất để duy trì tính toàn vẹn của các cụm và phân tích.

**- Description (0.27%):**

- Mặc dù chiếm tỷ lệ thấp nhưng vẫn có sự không nhất quán trong dữ liệu.
- Sự không nhất quán xuất hiện khi cùng OrderID luôn không có cùng một Mô tả, cho thấy vấn đề về chất lượng dữ liệu và có thể là lỗi trong mô tả sản phẩm.
- Với tỷ lệ thiếu thấp, việc loại bỏ là một cách tốt để tránh lỗi.

- Tiếp đó sẽ làm sạch giá trị thiếu của 2 cột CustomerID và Description

```
# Extracting rows with missing values in 'CustomerID' or 'Description' columns
df[df['CustomerID'].isnull() | df['Description'].isnull()].head()
```

| | OderlID | ProductID | Description | Quantity | InvoiceDate | Price | CustomerID | Country |
|---|---|---|---|---|---|---|---|---|
| **622** | 536414 | 22139 | NaN | 56 | 2010-12-01 11:52:00 | 0.00 | NaN | United Kingdom |
| **1443** | 536544 | 21773 | DECORATIVE ROSE BATHROOM BOTTLE | 1 | 2010-12-01 14:32:00 | 2.51 | NaN | United Kingdom |
| **1444** | 536544 | 21774 | DECORATIVE CATS BATHROOM BOTTLE | 2 | 2010-12-01 14:32:00 | 2.51 | NaN | United Kingdom |
| **1445** | 536544 | 21786 | POLKADOT RAIN HAT | 4 | 2010-12-01 14:32:00 | 0.85 | NaN | United Kingdom |
| **1446** | 536544 | 21787 | RAIN PONCHO RETROSPOT | 2 | 2010-12-01 14:32:00 | 1.66 | NaN | United Kingdom |

```
[ ] # Data Wrangling: làm sạch
    # Dealing with Missing values
    # Removing NaN's in Customer ID and 'Description' columns
    print("Shape of data before removing NaN's CustomerID",df.shape)
    df = df.dropna(subset=['CustomerID', 'Description'])

    Shape of data before removing NaN's CustomerID (541909, 8)

[ ] # Verifying the removal of missing vvalues : kiểm tra xem các giá trị bị thiếu đã được loại bỏ thành công khỏi tập dữ liệu hay không
    print("Missing values in each column after cleaning customerID :\n",df.isnull().sum().sum())

    Missing values in each column after cleaning customerID :
     0
```

**- Bước 2: Handling Duplicates (xử lý giá trị trùng lặp)**

```
# 3.2: Handling Duplicates: xử lý các dòng trùng lặp trong tập dữ liệu
# Finding duplicate rows (keeping all instances): Tìm các dòng trùng lặp và giữ tất cả.
# Cho biết mỗi dòng có phải là một bản sao hay không.
# Nếu giá trị của dòng đó đã xuất hiện trước đó, nó được coi là một bản sao và được giữ lại
duplicate_rows = df[df.duplicated(keep=False)]

# Sorting the data by certain columns to see the duplicate rows next to each other
# Sắp xếp dữ liệu theo các cột nhất định để nhìn thấy các dòng trùng lặp liền kề nhau
# Dữ liệu được sắp xếp theo các cột bên dưới
duplicate_rows_sorted = duplicate_rows.sort_values(by=['OderlID', 'ProductID', 'Description', 'CustomerID', 'Quantity'])

# Displaying the first 10 records
# Hiển thị 10 dòng đầu tiên của các dòng trùng lặp đã được sắp xếp.
duplicate_rows_sorted.head(10)
```

| | OderlID | ProductID | Description | Quantity | InvoiceDate | Price | CustomerID | Country |
|---|---|---|---|---|---|---|---|---|
| **494** | 536409 | 21866 | UNION JACK FLAG LUGGAGE TAG | 1 | 2010-12-01 11:45:00 | 1.25 | 17908.0 | United Kingdom |
| **517** | 536409 | 21866 | UNION JACK FLAG LUGGAGE TAG | 1 | 2010-12-01 11:45:00 | 1.25 | 17908.0 | United Kingdom |
| **485** | 536409 | 22111 | SCOTTIE DOG HOT WATER BOTTLE | 1 | 2010-12-01 11:45:00 | 4.95 | 17908.0 | United Kingdom |
| **539** | 536409 | 22111 | SCOTTIE DOG HOT WATER BOTTLE | 1 | 2010-12-01 11:45:00 | 4.95 | 17908.0 | United Kingdom |
| **489** | 536409 | 22866 | HAND WARMER SCOTTY DOG DESIGN | 1 | 2010-12-01 11:45:00 | 2.10 | 17908.0 | United Kingdom |
| **527** | 536409 | 22866 | HAND WARMER SCOTTY DOG DESIGN | 1 | 2010-12-01 11:45:00 | 2.10 | 17908.0 | United Kingdom |
| **521** | 536409 | 22900 | SET 2 TEA TOWELS I LOVE LONDON | 1 | 2010-12-01 11:45:00 | 2.95 | 17908.0 | United Kingdom |
| **537** | 536409 | 22900 | SET 2 TEA TOWELS I LOVE LONDON | 1 | 2010-12-01 11:45:00 | 2.95 | 17908.0 | United Kingdom |
| **578** | 536412 | 21448 | 12 DAISY PEGS IN WOOD BOX | 1 | 2010-12-01 11:49:00 | 1.65 | 17920.0 | United Kingdom |
| **598** | 536412 | 21448 | 12 DAISY PEGS IN WOOD BOX | 1 | 2010-12-01 11:49:00 | 1.65 | 17920.0 | United Kingdom |

**Kết luận**: Giữ các hàng trùng lặp này có thể gây ra nhiễu và không chính xác trong việc phân cụm. Do đó sẽ loại bỏ các hàng trùng lặp giúp tạo ra tập dữ liệu sạch hơn, từ đó giúp xây dựng các nhóm khách hàng chính xác hơn dựa trên hành vi mua hàng của họ. Cũng như xác định đúng các sản phẩm được mua nhiều nhất.

```
# Displaying the number of duplicate rows : Hiển thị số hàng trùng lặp
print("Number of duplicates before cleaning:",df.duplicated().sum())
# Removing duplicate rows: xóa số hàng trùng lặp
df.drop_duplicates(inplace=True)
print("Number of duplicates after cleaning:",df.duplicated().sum())
```

```
Number of duplicates before cleaning: 5225
Number of duplicates after cleaning: 0
```

```
# Getting the number of rows in the dataframe: Lấy số hàng trong bảng dữ liệu
df.shape[0]
```

```
401604
```

**- Bước 3: Treating Cancelled Transactions (xử lý các giao dịch bị hủy)**

```
# 3.3: Treating Cancelled Transactions: Xử lý các giao dịch bị hủy
# Filter out the rows with OderlID starting with "C" and create a new column indicating the transaction status
# "C" được sử dụng để biểu thị các giao dịch hủy bỏ trong dữ liệu.
# Lọc ra các dòng và đánh dấu cho các giao dịch hủy, tạo một cột mới để chỉ ra trạng thái của giao dịch.
df['Transaction_Status'] = np.where(df['OderlID'].astype(str).str.startswith('C'), 'Cancelled', 'Completed')

# Analyze the characteristics of these rows (considering the new column)
# Phân tích các đặc điểm của các giao dịch bị hủy
# Cho biết liệu trạng thái của giao dịch có phải là 'Cancelled' hay không.
cancelled_transactions = df[df['Transaction_Status'] == 'Cancelled']

# Cột 'CustomerID' được loại bỏ ra khỏi bảng tóm tắt vì thông tin này không liên quan đến phân tích.
cancelled_transactions.describe().drop('CustomerID', axis=1)
```

| | Quantity | Price |
|---|---|---|
| **count** | 8872.000000 | 8872.000000 |
| **mean** | -30.774910 | 18.899512 |
| **std** | 1172.249902 | 445.190864 |
| **min** | -80995.000000 | 0.010000 |
| **25%** | -6.000000 | 1.450000 |
| **50%** | -2.000000 | 2.950000 |
| **75%** | -1.000000 | 4.950000 |
| **max** | -1.000000 | 38970.000000 |

**Từ những kết quả trên, ta thấy dữ liệu giao dịch bị hủy:**

- Đa số các số lượng trong các giao dịch bị hủy đều là số âm, cho thấy rằng đó thực sự là các đơn hàng đã bị hủy.
- Cột Price có một phạm vi giá khá rộng, nghĩa là có nhiều loại sản phẩm khác nhau, từ giá thấp đến cao, đã được hủy trong các giao dịch này. Điều này cho

thấy sự đa dạng trong các mặt hàng bị hủy, có thể là do nhiều nguyên nhân khác nhau như sự không hài lòng của khách hàng, lỗi trong quá trình đặt hàng hoặc giao hàng, hoặc các vấn đề khác liên quan đến sản phẩm.

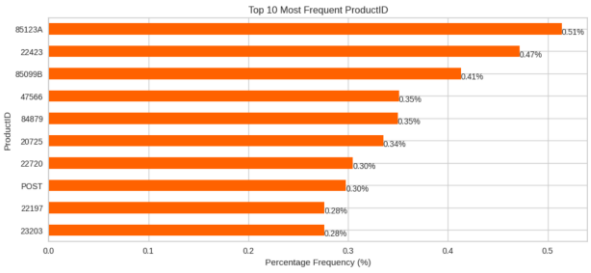
**Chiến lược/giải pháp cho việc xử lý các giao dịch bị hủy:** Giữ lại những giao dịch này trong tập dữ liệu, đồng thời đánh dấu chúng một cách rõ ràng để dễ dàng phân tích thêm. Điều này sẽ giúp tăng cường quá trình phân cụm bằng cách tích hợp các mẫu và xu hướng được quan sát trong dữ liệu hủy, có thể đại diện cho một số hành vi hoặc sở thích của khách hàng từ đó tìm ra xu hướng mua hàng để cải thiện năng suất bán hang và cho phép hệ thống đề xuất có thể ngăn không đề xuất các sản phẩm có khả năng cao bị hủy, từ đó cải thiện chất lượng các đề xuất.

### - Bước 4: Chỉnh sửa các bất thường của ProductID

```
# 3.4: Correcting ProductID Anomalies: Chỉnh sửa các bất thường của ProductID
# Finding the number of unique product id: Tìm số lượng product id duy nhất
unique_product_id = df['ProductID'].nunique()

# Printing the number of unique product id: in ra
print(f"The number of unique product id in the dataset is: {unique_product_id}")
```

```
The number of unique product id in the dataset is: 3684
```

```
# Finding the top 10 most frequent product id: Tìm 10 mã sản phẩm phổ biến nhất
top_10_product_id = df['ProductID'].value_counts(normalize=True).head(10) * 100

# Plotting the top 10 most frequent product id: Vẽ biểu đồ cho 10 mã sản phẩm phổ biến nhất
plt.figure(figsize=(12, 5))
top_10_product_id.plot(kind='barh', color='#ff6200')

# Adding the percentage frequency on the bars: Thêm phần trăm trên bars chart
for index, value in enumerate(top_10_product_id):
    plt.text(value, index+0.25, f'{value:.2f}%', fontsize=10)

plt.title('Top 10 Most Frequent ProductID')
plt.xlabel('Percentage Frequency (%)')
plt.ylabel('ProductID')
plt.gca().invert_yaxis()
plt.show()
```




Để hiểu rõ hơn về cách các mã sản phẩm(ProductID) được tạo ra, sẽ kiểm tra số lần xuất hiện của các ký tự trong các mã sản phẩm. Điều này giúp phát hiện ra các mục không bình thường hoặc không phù hợp với quy tắc thông thường của mã sản phẩm.

```
# Finding the number of numeric characters in each unique product id:  tìm số lượng ký tự trong mỗi mã sản phẩm duy nhất trong tập dữ liệu
unique_product_id = df['ProductID'].unique()
numeric_char_counts_in_unique_id = pd.Series(unique_product_id).apply(lambda x: sum(c.isdigit() for c in str(x))).value_counts()

# Printing the value counts for unique product id: In
print("Value counts of numeric character frequencies in unique product id:")
print("-"*70)
print(numeric_char_counts_in_unique_id)
```

```
Value counts of numeric character frequencies in unique product id:
----------------------------------------------------------------------
5    3676
0       7
1       1
dtype: int64
```

**Các kết quả cho thấy:** Đa số các mã sản phẩm (3676 trong số 3684) theo chuẩn có đúng 5 chữ số. Điều này có vẻ như là định dạng tiêu chuẩn được sử dụng trong tập dữ liệu. Tuy nhiên, có một số sự khác biệt:

- 7 mã sản phẩm không (0) có chữ số nào

- 1 mã chỉ có một (1) chữ số

Những mã này không tuân thủ định dạng tiêu chuẩn và cần được điều tra kỹ lưỡng hơn để xác định tính chất của chúng.

**Số lượng ký tự "số" trong mỗi mã sản phẩm:**

```
# Finding and printing the product_id with 0 and 1 numeric characters: Tìm và in
anomalous_product_id = [id for id in unique_product_id if sum(c.isdigit() for c in str(id)) in (0, 1)]

# Printing each product id on a new line: in
print("Anomalous product_id:")
print("-"*22)
for id in anomalous_product_id:
    print(id)
```

```
Anomalous product_id:
----------------------
POST
D
C2
M
BANK CHARGES
PADS
DOT
CRUK
```

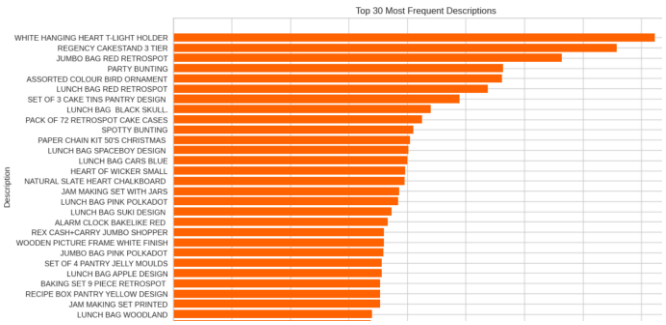
```
# Calculating the percentage of records with these product id:  tính phần trăm mã sản phẩm bất thường, tức là
percentage_anomalous = (df['ProductID'].isin(anomalous_product_id).sum() / len(df)) * 100

# Printing the percentage: In ra kết quả nè
print(f"The percentage of records with anomalous product id in the dataset is: {percentage_anomalous:.2f}%")
```

```
The percentage of records with anomalous product id in the dataset is: 0.48%
```

**Đưa ra kết luận**: Dựa trên phân tích, thấy rằng chỉ có một tỷ lệ rất nhỏ các mã, khoảng 0,48%, có các mã sản phẩm bất thường, không tuân thủ định dạng tiêu chuẩn như phần lớn dữ liệu. Ngoài ra, những mã này chỉ chiếm một phần nhỏ trong số tất cả các mã sản phẩm duy nhất (chỉ có 8 trong tổng số 3684). Có vẻ những mã này đại diện cho các giao dịch không phải sản phẩm như "BANK CHARGES", "POST" (có thể là phí bưu điện), v.v. Vì chúng không đại diện cho các sản phẩm thực tế và chỉ là một phần nhỏ của tập dữ liệu. Vì vậy, việc loại bỏ các giao dịch không phải là sản phẩm thực sự từ tập dữ liệu sẽ giúp tập trung vào các giao dịch chính xác và ý nghĩa hơn, từ đó tạo ra các cụm khách hàng và hệ thống đề xuất sản phẩm chính xác hơn.

**- Bước 5: Làm sạch cột Description**

```
# 3.5: Cleaning Description Column
# Calculate the occurrence of each unique description and sort them
# Tính toán số lần xuất hiện của mỗi mô tả sản phẩm duy nhất trong c
description_counts = df['Description'].value_counts()

# Get the top 30 descriptions: lấy top 30 mô tả
top_30_descriptions = description_counts[:30]

# Plotting: vẽ biểu đồ để hiển thị số lần xuất hiện của mỗi mô tả
plt.figure(figsize=(12,8))
plt.barh(top_30_descriptions.index[::-1], top_30_descriptions.values

# Adding labels and title: thêm các nhãn và tiêu đề cho biểu đồ
plt.xlabel('Number of Occurrences')
plt.ylabel('Description')
plt.title('Top 30 Most Frequent Descriptions')

# Show the plot
plt.show()
```




Qua đó ta thấy với các bất đồng và bất thường đã gặp trong tập dữ liệu, việc kiểm tra xem có mô tả được nhập bằng chữ thường hoặc kết hợp các kiểu chữ là cần thiết.

```
# Find unique descriptions containing lowercase characters: tìm các mô tả trong cột "Description" mà chứa ít
lowercase_descriptions = df['Description'].unique()
lowercase_descriptions = [desc for desc in lowercase_descriptions if any(char.islower() for char in desc)]

# Print the unique descriptions containing lowercase characters: in ra
print("The unique descriptions containing lowercase characters are:")
print("-"*60)
for desc in lowercase_descriptions:
    print(desc)
```

```
The unique descriptions containing lowercase characters are:
------------------------------------------------------------
BAG 500g SWIRLY MARBLES
POLYESTER FILLER PAD 45x45cm
POLYESTER FILLER PAD 45x30cm
POLYESTER FILLER PAD 40x40cm
FRENCH BLUE METAL DOOR SIGN No
BAG 250g SWIRLY MARBLES
BAG 125g SWIRLY MARBLES
3 TRADITIONAl BISCUIT CUTTERS  SET
NUMBER TILE COTTAGE GARDEN No
FOLK ART GREETING CARD,pack/12
ESSENTIAL BALM 3.5g TIN IN ENVELOPE
POLYESTER FILLER PAD 65CMx65CM
NUMBER TILE VINTAGE FONT No
POLYESTER FILLER PAD 30CMx30CM
POLYESTER FILLER PAD 60x40cm
FLOWERS HANDBAG blue and orange
Next Day Carriage
THE KING GIFT BAG 25x24x12cm
High Resolution Image
```

Dựa trên kết quả trên, dường như các mô tả chứa các ký tự viết thường như "Next Day Carriage" và "High Resolution Image," không phải là mô tả sản phẩm. Do đó sẽ loại bỏ và chuẩn hóa.

```
# Remove rows with service-related information in the description: xóa
df = df[~df['Description'].isin(service_related_descriptions)]

# Standardize the text to uppercase to maintain uniformity across the ddataset
df['Description'] = df['Description'].str.upper()
```

### - Bước 6: Xử lý các giá trị bằng 0

```
# 3.6 : Treating Zero Prices
df['Price'].describe()
```

```
count    399606.000000
mean          2.904957
std           4.448796
min           0.000000
25%           1.250000
50%           1.950000
75%           3.750000
max         649.500000
Name: Price, dtype: float64
```

```
df[df['Price']==0].describe()[['Quantity']]
```

| | Quantity |
|---|---|
| count | 33.000000 |
| mean | 420.515152 |
| std | 2176.713608 |
| min | 1.000000 |
| 25% | 2.000000 |
| 50% | 11.000000 |
| 75% | 36.000000 |
| max | 12540.000000 |

```
# Removing records with a
df = df[df['Price'] > 0]
```

Vì muốn hiểu hành vi của khách hàng thông qua K-means clustering. Tuy nhiên, loại bỏ các điểm ngoại lệ sớm có thể làm mất thông tin quan trọng. Vì vậy,sẽ xử lý các điểm ngoại lệ sau khi tạo tập dữ liệu tập trung vào khách hàng.

## 2.3 Xây dựng các đặc trưng

### - Bước 1: RFM features

```
# 4: Feature Engineering
# 4.1 | RFM Features
# 4.1.1 | Recency (R)
# Convert InvoiceDate to datetime type: chuyển đổi cột "InvoiceDate" thành kiểu dữ liệu datetime
df['InvoiceDate'] = pd.to_datetime(df['InvoiceDate'])

# Convert InvoiceDate to datetime and extract only the date: Trích xuất ngày và gán vào cột mới InvoiceDay
df['InvoiceDay'] = df['InvoiceDate'].dt.date

# Find the most recent purchase date for each customer: tìm ngày mua gần nhất
# Nhóm dữ liệu theo CustomerID và chọn ngày lớn nhất cho mỗi nhóm
customer_data = df.groupby('CustomerID')['InvoiceDay'].max().reset_index()

# Find the most recent date in the entire dataset : tìm ngày gần nhất trong toàn bộ tập dữ liệu
most_recent_date = df['InvoiceDay'].max()

# Convert InvoiceDay to datetime type before subtraction : Chuyển đổi cột 'InvoiceDay' sang kiểu dữ liệu d
customer_data['InvoiceDay'] = pd.to_datetime(customer_data['InvoiceDay'])
most_recent_date = pd.to_datetime(most_recent_date)

# Calculate the number of days since the last purchase for each customer: Tính số ngày kể từ lần mua hàng
customer_data['Days_Since_Last_Purchase'] = (most_recent_date - customer_data['InvoiceDay']).dt.days

# Remove the InvoiceDay column: xóa cột InvoiceDay khỏi DataFrame customer_data
customer_data.drop(columns=['InvoiceDay'], inplace=True)
```

```
# Hiển thị năm hàng đầu của DataFrame customer_data
customer_data.head()
```

| | CustomerID | Days_Since_Last_Purchase |
|---|---|---|
| 0 | 12346.0 | 325 |
| 1 | 12347.0 | 2 |
| 2 | 12348.0 | 75 |
| 3 | 12349.0 | 18 |
| 4 | 12350.0 | 310 |

```
# 4.1.2 | Frequency (F)
# Calculate the total number of transactions made by each customer: Tính tổng số giao dịch của
total_transactions = df.groupby('CustomerID')['OderlID'].nunique().reset_index()
total_transactions.rename(columns={'OderlID': 'Total_Transactions'}, inplace=True)

# Calculate the total number of products purchased by each customer: Tính tổng số sản phẩm mà mỗ
total_products_purchased = df.groupby('CustomerID')['Quantity'].sum().reset_index()
total_products_purchased.rename(columns={'Quantity': 'Total_Products_Purchased'}, inplace=True)

# Merge the new features into the customer_data dataframe: Gộp các thuộc tính mới vào bảng dữ l
customer_data = pd.merge(customer_data, total_transactions, on='CustomerID')
customer_data = pd.merge(customer_data, total_products_purchased, on='CustomerID')

# Display the first few rows of the customer_data dataframe: Hiển thị một số dòng đầu tiên của
customer_data.head()
```

| | CustomerID | Days_Since_Last_Purchase | Total_Transactions | Total_Products_Purchased |
|---|---|---|---|---|
| 0 | 12346.0 | 325 | 2 | 0 |
| 1 | 12347.0 | 2 | 7 | 2458 |
| 2 | 12348.0 | 75 | 4 | 2332 |
| 3 | 12349.0 | 18 | 1 | 630 |
| 4 | 12350.0 | 310 | 1 | 196 |

```
# 4.1.3 | Monetary (M)
# Calculate the total spend by each customer: tính toán tổng chi phí mà mỗi khách hàng đã chi tiêu
df['Total_Spend'] = df['Price'] * df['Quantity']
total_spend = df.groupby('CustomerID')['Total_Spend'].sum().reset_index()

# Calculate the average transaction value for each customer: tính giá trị giao dịch trung bình cho mỗi khách hàng
average_transaction_value = total_spend.merge(total_transactions, on='CustomerID')
average_transaction_value['Average_Transaction_Value'] = average_transaction_value['Total_Spend'] / average_transaction_value['Total_Transactions']

# Merge the new features into the customer_data dataframe: gộp các đặc điểm mới vào bảng dữ liệu customer_data
customer_data = pd.merge(customer_data, total_spend, on='CustomerID')
customer_data = pd.merge(customer_data, average_transaction_value[['CustomerID', 'Average_Transaction_Value']], on='CustomerID')

# Display the first few rows of the customer_data dataframe: Hiển thị một số dòng đầu tiên của bảng dữ liệu customer_data
customer_data.head()
```

| | CustomerID | Days_Since_Last_Purchase | Total_Transactions | Total_Products_Purchased | Total_Spend | Average_Transaction_Value |
|---|---|---|---|---|---|---|
| 0 | 12346.0 | 325 | 2 | 0 | 0.00 | 0.000000 |
| 1 | 12347.0 | 2 | 7 | 2458 | 4310.00 | 615.714286 |
| 2 | 12348.0 | 75 | 4 | 2332 | 1437.24 | 359.310000 |
| 3 | 12349.0 | 18 | 1 | 630 | 1457.55 | 1457.550000 |
| 4 | 12350.0 | 310 | 1 | 196 | 294.40 | 294.400000 |

## - Bước 2: Đo lường mức độ đa dạng của các sản phẩm

```
# 4.2 | Product Diversity: đo lường mức độ đa dạng của các sản phẩm
# Calculate the number of unique products purchased by each customer: tính số lượng sản phẩm unique
unique_products_purchased = df.groupby('CustomerID')['ProductID'].nunique().reset_index()
unique_products_purchased.rename(columns={'ProductID': 'Unique_Products_Purchased'}, inplace=True)

# Merge the new feature into the customer_data dataframe:
# kết hợp DataFrame unique_products_purchased (chứa số lượng sản phẩm độc đáo mỗi khách hàng đã mua
# Sau khi thực hiện, customer_data sẽ chứa thông tin về số lượng sản phẩm unique mà mỗi khách hàng
customer_data = pd.merge(customer_data, unique_products_purchased, on='CustomerID')

# Display the first few rows of the customer_data dataframe: hiển thị một số dòng đầu tiên của Data
customer_data.head()
```

| | CustomerID | Days_Since_Last_Purchase | Total_Transactions | Total_Products_Purchased | Total_Spend | Average_Transaction_Value | Unique_Products_Purchased |
|---|---|---|---|---|---|---|---|
| 0 | 12346.0 | 325 | 2 | 0 | 0.00 | 0.000000 | 1 |
| 1 | 12347.0 | 2 | 7 | 2458 | 4310.00 | 615.714286 | 103 |
| 2 | 12348.0 | 75 | 4 | 2332 | 1437.24 | 359.310000 | 21 |
| 3 | 12349.0 | 18 | 1 | 630 | 1457.55 | 1457.550000 | 72 |
| 4 | 12350.0 | 310 | 1 | 196 | 294.40 | 294.400000 | 16 |

## - Bước 3: Các đặc điểm hành vi

Trong bước này, sẽ tìm hiểu thói quen mua sắm của khách hàng bằng cách xem xét các thông tin sau:

- Trung bình số ngày giữa các lần mua(Average Days Between Purchases): Đo lường thời gian trung bình mà một khách hàng chờ đợi trước khi mua tiếp lần tiếp theo.
- Ngày mua sắm yêu thích(Favorite Shopping Day): Xác định ngày trong tuần mà khách hàng thường mua sắm nhiều nhất.
- Giờ mua sắm yêu thích(Favorite Shopping Hour): Xác định thời điểm trong ngày mà khách hàng thường mua sắm nhiều nhất.

```
#  4.3 | Behavioral Features
# Extract day of week and hour from InvoiceDate:  trích xuất ngày trong tuần và giờ từ cột InvoiceDate,
df['Day_Of_Week'] = df['InvoiceDate'].dt.dayofweek
df['Hour'] = df['InvoiceDate'].dt.hour

# Calculate the average number of days between consecutive purchases: tính trung bình số ngày giữa các lần mua hàng liên tiếp
days_between_purchases = df.groupby('CustomerID')['InvoiceDay'].apply(lambda x: (x.diff().dropna()).apply(lambda y: y.days))
average_days_between_purchases = days_between_purchases.groupby('CustomerID').mean().reset_index()
average_days_between_purchases.rename(columns={'InvoiceDay': 'Average_Days_Between_Purchases'}, inplace=True)

# Find the favorite shopping day of the week: tìm ngày mua sắm ưa thích của tuần
favorite_shopping_day = df.groupby(['CustomerID', 'Day_Of_Week']).size().reset_index(name='Count')
favorite_shopping_day = favorite_shopping_day.loc[favorite_shopping_day.groupby('CustomerID')['Count'].idxmax()][['CustomerID', 'Day_Of_Week']]
# Find the favorite shopping hour of the day: tìm giờ mua sắm ưa thích trong ngày
favorite_shopping_hour = df.groupby(['CustomerID', 'Hour']).size().reset_index(name='Count')
favorite_shopping_hour = favorite_shopping_hour.loc[favorite_shopping_hour.groupby('CustomerID')['Count'].idxmax()][['CustomerID', 'Hour']]

# Merge the new features into the customer_data dataframe: gộp các đặc trưng mới vào dataframe customer_data
# Sau khi thực hiện, dataframe customer_data sẽ chứa tất cả các đặc trưng mới, bao gồm cả giờ mua sắm ưa thích của ngày cho mỗi khách hàng.
customer_data = pd.merge(customer_data, average_days_between_purchases, on='CustomerID')
customer_data = pd.merge(customer_data, favorite_shopping_day, on='CustomerID')
customer_data = pd.merge(customer_data, favorite_shopping_hour, on='CustomerID')

# Display the first few rows of the customer_data dataframe: hiển thị một số dòng đầu tiên của DataFrame customer_data, mặc định là 5 dòng
customer_data.head()
```

**Kết quả:**

| | CustomerID | Days_Since_Last_Purchase | Total_Transactions | Total_Products_Purchased | Total_Spend | Average_Transaction_Value | Unique_Products_Purchased | Average_Days_Between_Purchases | Day_Of_Week | Hour |
|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 12346.0 | 325 | 2 | 0 | 0.00 | 0.000000 | 1 | 0.000000 | 1 | 10 |
| 1 | 12347.0 | 2 | 7 | 2458 | 4310.00 | 615.714286 | 103 | 2.016575 | 1 | 14 |
| 2 | 12348.0 | 75 | 4 | 2332 | 1437.24 | 359.310000 | 21 | 10.884615 | 3 | 19 |
| 3 | 12349.0 | 18 | 1 | 630 | 1457.55 | 1457.550000 | 72 | 0.000000 | 0 | 9 |
| 4 | 12350.0 | 310 | 1 | 196 | 294.40 | 294.400000 | 16 | 0.000000 | 2 | 16 |

**- Bước 4: Các đặc trưng địa lý**

Trong bước Geographic Features, chúng ta sẽ thêm một đặc trưng địa lý vào dữ liệu, cho biết quốc gia mà mỗi khách hàng đến từ. Điều này giúp hiểu sâu hơn về các mẫu mua sắm và sở thích mua hàng ở các khu vực khác nhau.

- Đặc trưng Country(Quốc gia): Đặc trưng này xác định quốc gia mà mỗi khách hàng đang ở.

```
# 4.4 | Geographic Features
df['Country'].value_counts(normalize=True).head()
```

```
United Kingdom    0.890971
Germany           0.022722
France            0.020402
EIRE              0.018440
Spain             0.006162
Name: Country, dtype: float64
```

```
# Group by CustomerID and Country to get the number of transactions per country for each customer
customer_country = df.groupby(['CustomerID', 'Country']).size().reset_index(name='Number_of_Transactions')

# Get the country with the maximum number of transactions for each customer (in case a customer has transactions from multiple countries)
# lấy số giao dịch cho mỗi quốc gia cho mỗi khách hàng, ta cần nhóm dữ liệu theo cả hai cột 'CustomerID' và 'Country'
# Sau đó đếm số lần xuất hiện của mỗi quốc gia trong mỗi nhóm
customer_main_country = customer_country.sort_values('Number_of_Transactions', ascending=False).drop_duplicates('CustomerID')

# Create a binary column indicating whether the customer is from the UK or not
# Lấy thông tin về quốc gia mà mỗi khách hàng đã thực hiện số lượng giao dịch lớn nhất
# Đây là quốc gia mà khách hàng thường thực hiện nhiều giao dịch nhất trong tất cả các giao dịch của mình.
customer_main_country['Is_UK'] = customer_main_country['Country'].apply(lambda x: 1 if x == 'United Kingdom' else 0)

# Merge this data with our customer_data dataframe: gộp dữ liệu này với DataFrame customer_data
customer_data = pd.merge(customer_data, customer_main_country[['CustomerID', 'Is_UK']], on='CustomerID', how='left')

# Display the first few rows of the customer_data dataframe: hiển thị một số dòng đầu tiên của DataFrame customer_data, mặc định là 5 dòng
customer_data.head()
```

Sau khi đã tạo ra một bộ dữ liệu customer_data chứa tất cả các thuộc tính và đặc điểm mới thì bây giờ bộ dữ liệu đã sẵn sàng tiếp tục đến các bước tiếp theo gồm: xử lý các giá trị ngoại lai, và chuẩn bị dữ liệu của chúng ta cho quá trình phân cụm sau đó sẽ xây dựng hệ thống gợi ý.

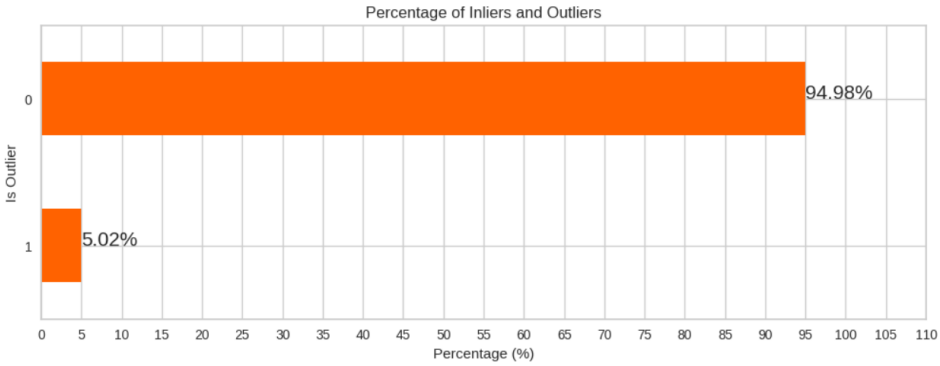
**2.4 Xử lý các giá trị ngoại lai**

**- Bước 1:** Sử dụng thuật toán Isolation Forest(Rừng cách ly) để phát hiện các điểm ngoại lai và xử lý để dữ liệu trở nên đồng nhất và phù hợp hơn cho việc phân tích tiếp theo.

```
# 5 | Outlier Detection and Treatment
# Initializing the IsolationForest model with a contamination parameter of 0.05
model = IsolationForest(contamination=0.05, random_state=0)

# Fitting the model on our dataset (converting DataFrame to NumPy to avoid warning)
customer_data['Outlier_Scores'] = model.fit_predict(customer_data.iloc[:, 1:].to_numpy())

# Creating a new column to identify outliers (1 for inliers and -1 for outliers)
customer_data['Is_Outlier'] = [1 if x == -1 else 0 for x in customer_data['Outlier_Scores']]

# Display the first few rows of the customer_data dataframe
customer_data.head()
```

**- Bước 2:** Sau khi sử dụng thuật toán Isolation Forest, đã xác định được các điểm ngoại lệ và đánh dấu chúng trong một cột mới có tên là Is_Outlier. Tiếp theo vẽ biểu đồ để

hiển thị phân phối của các điểm số ngoại lệ và số lượng điểm bình thường và ngoại lệ được mô hình phát hiện.

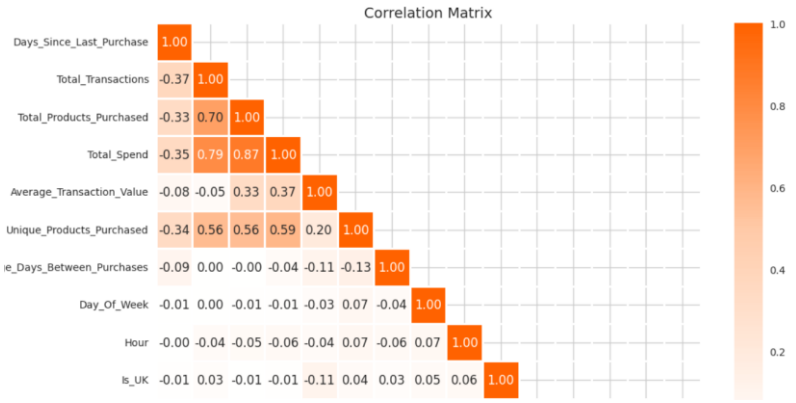



Từ biểu đồ, có khoảng 5% khách hàng là các điểm ngoại lệ, tỷ lệ này hợp lý cho việc phân đoạn khách hàng. Chiến lược tiếp theo là loại bỏ các điểm ngoại lệ để không ảnh hưởng đến quá trình phân cụm và giúp cải thiện chất lượng của quá trình phân đoạn khách hàng.

**- Bước 3:** Chuẩn hóa đặc trưng và áp dụng các thuật toán phân cụm để xác định các nhóm khách hàng.

### 2.5 Đánh giá mối quan hệ giữa các biến trong tập dữ liệu

Trước khi phân cụm K-means, cần kiểm tra tương quan giữa các đặc trưng. Đa tuyến tính có thể làm suy giảm chất lượng của cụm. Sau đó sử dụng PCA để giảm chiều dữ liệu và cải thiện hiệu suất của quá trình phân cụm.




Biểu đồ heatmap cho thấy một số cặp biến có tương quan cao như: Monthly_Spending_Mean và Average_Transaction_Value, Total_Spend và Total_Products_Purchased, Total_Transactions và Total_Spend, Cancellation_Rate và Cancellation_Frequency, Total_Transactions và Total_Products_Purchased. Trước khi

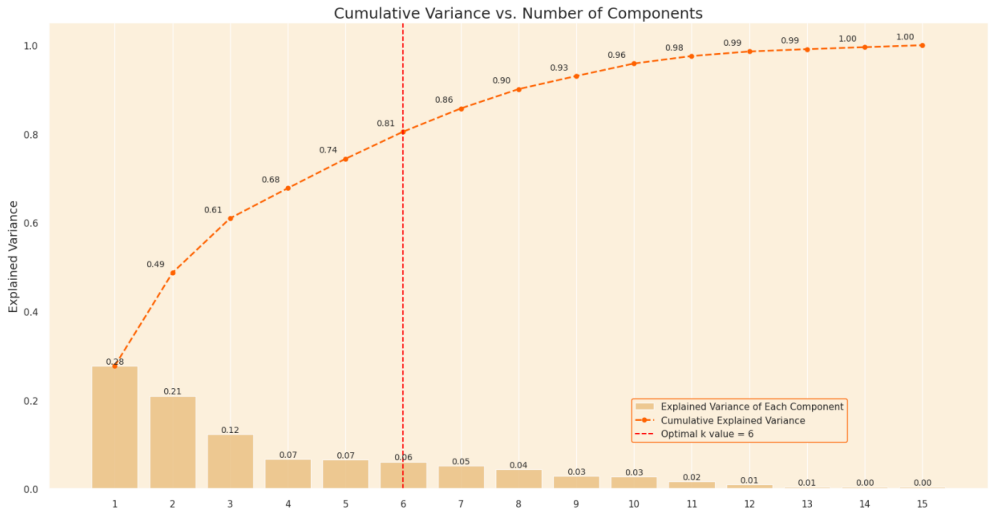
tiếp tục sẽ xử lý đa tuyến tính thông qua PCA có thể giúp tạo ra các cụm ổn định hơn trong quá trình phân cụm bằng KMeans.

### 2.6 Chuẩn hoá dữ liệu về cùng phạm vi (same range)

Trước khi phân cụm và giảm chiều dữ liệu, việc scale dữ liệu là rất quan trọng, đặc biệt đối với K-means và PCA. Việc scale giúp cân bằng ảnh hưởng và phát hiện mẫu thực sự trong dữ liệu bằng cách chuyển đổi các đặc trưng sao cho giá trị trong khoảng từ 0 đến 1. Tuy nhiên, không phải tất cả các đặc trưng đều cần được scale.

### 2.7 Giảm chiều dữ liệu

Giảm chiều dữ liệu giúp cải thiện phân cụm, giảm nhiễu, tăng tính hiển thị và hiệu suất tính toán bằng phương pháp PCA.

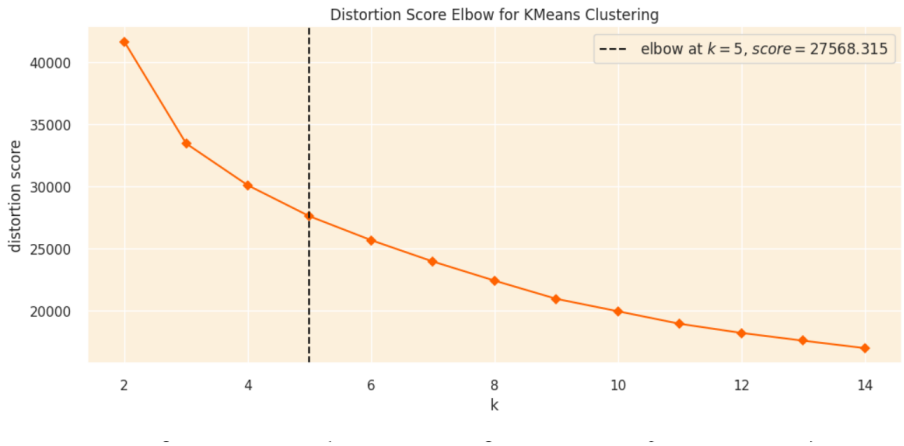



Chọn giữ lại 6 thành phần đầu tiên để giữ lại thông tin quan trọng trong khi giảm chiều dữ liệu.

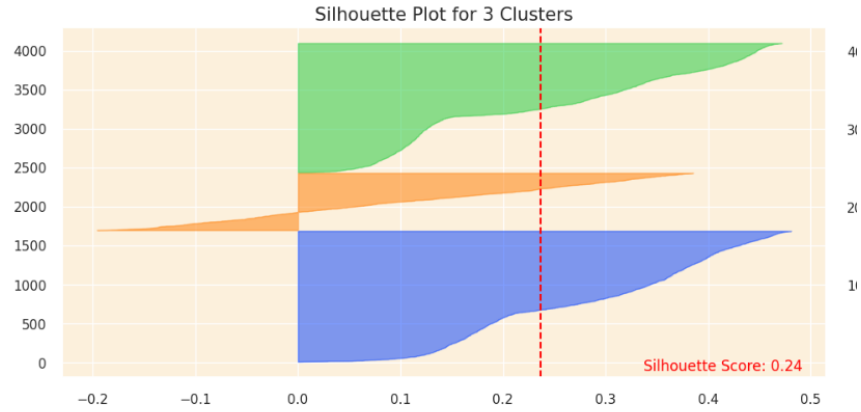
| CustomerID | PC1 | PC2 | PC3 | PC4 | PC5 | PC6 |
|---|---|---|---|---|---|---|
| **12346.0** | -2.186469 | -1.705370 | -1.576745 | 1.008187 | -0.411803 | -1.658012 |
| **12347.0** | 3.290264 | -1.387375 | 1.923310 | -0.930990 | -0.010591 | 0.873150 |
| **12348.0** | 0.584684 | 0.585019 | 0.664727 | -0.655411 | -0.470280 | 2.306657 |
| **12349.0** | 1.791116 | -2.695652 | 5.850040 | 0.853418 | 0.677111 | -1.520098 |
| **12350.0** | -1.997139 | -0.542639 | 0.578781 | 0.183682 | -1.484838 | 0.062672 |

### 2.8 K-means

- **Bước 1**: Xác định số lượng cụm tối ưu bằng phương pháp khuỷu tay (Elbow Method) hoặc phương pháp Silhouette.




Number of




Giá trị k tối ưu cho thuật toán KMeans được tìm thấy tại điểm khuỷu tay là k =5 nhưng áp dụng phương pháp Silhouette thì thấy k=3 là tốt nhất

**- Bước 2: Phân Cụm - K-means**

Ứng dụng thuật toán phân cụm K-means để phân đoạn khách hàng thành các nhóm khác nhau dựa trên hành vi mua hàng và các đặc điểm khác, sử dụng số lượng cụm tối ưu đã xác định trước đó. Để đảm bảo việc gán nhãn nhất quán, nhóm đã thực hiện việc hoán đổi nhãn dựa trên tần suất mẫu trong mỗi cụm.

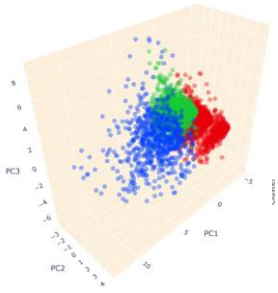
```
# Apply KMeans clustering using the optimal k
kmeans = KMeans(n_clusters=3, init='k-means++', n_init=10, max_iter=100, random_state=0)
kmeans.fit(customer_data_pca)

# Get the frequency of each cluster
cluster_frequencies = Counter(kmeans.labels_)

# Create a mapping from old labels to new labels based on frequency
label_mapping = {label: new_label for new_label, (label, _) in
                 enumerate(cluster_frequencies.most_common())}

# Reverse the mapping to assign labels as per your criteria
label_mapping = {v: k for k, v in {2: 1, 1: 0, 0: 2}.items()}

# Apply the mapping to get the new labels
new_labels = np.array([label_mapping[label] for label in kmeans.labels_])

# Append the new cluster labels back to the original dataset
customer_data_cleaned['cluster'] = new_labels

# Append the new cluster labels to the PCA version of the dataset
customer_data_pca['cluster'] = new_labels
```

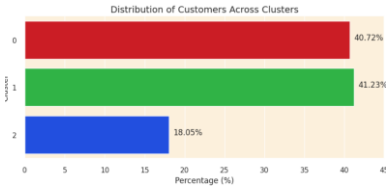
```
[56] # Display the 10 rows of the original dataframe
     customer_data_cleaned.head()
```

## 3. Kết quả thực nghiệm:

3.1. Đánh giá từng cụm

Sau khi xác định số lượng cụm tối ưu (k= 3), sẽ sử dụng ba phương pháp để đánh giá chất lượng của các cụm: 3D Visualization of Top PCs, Cluster Distribution Visualization, Evaluation Metrics




| Metric | Value |
|---|---|
| Number of Observations | 4067 |
| Silhouette Score | 0.2362713702277992 |
| Calinski Harabasz Score | 1257.179496292158 |
| Davies Bouldin Score | 1.368405537299602 |

- Cụm 0 (Biểu đồ Đỏ): Những Người Mua Hàng Đa Dạng với Sở Thích Mua Sắm vào Cuối Tuần
    - Khách hàng trong nhóm này thường chi tiêu ít, với số giao dịch và sản phẩm mua ít hơn.
    - Họ có xu hướng mua sắm vào cuối tuần, được biểu hiện qua giá trị Day_of_Week rất cao.
    - Xu hướng chi tiêu của họ khá ổn định nhưng ở mức thấp, và họ có độ biến động chi tiêu hàng tháng thấp (Monthly_Spending_Std thấp).
    - Những khách hàng này ít khi hủy giao dịch, thể hiện mức độ hủy giao dịch và tỷ lệ hủy thấp.
    - Giá trị giao dịch trung bình ở mức thấp, cho thấy khi họ mua sắm, họ có xu hướng chi tiêu ít mỗi giao dịch.
- Cụm 1 (Biểu đồ Xanh Lá Cây): Người Mua Hàng Lớn Nhưng Hiếm Khi Mua và Có Xu Hướng Chi Tiêu Cao
    - Khách hàng trong nhóm này có mức chi tiêu trung bình, nhưng giao dịch của họ không thường xuyên, được thể hiện qua giá trị Days_Since_Last_Purchase và Average_Days_Between_Purchases cao.
    - Họ có xu hướng chi tiêu rất cao, thể hiện rằng chi tiêu của họ đã tăng lên theo thời gian.

- Những khách hàng này thích mua sắm vào cuối ngày, được thể hiện qua giá trị Hour cao, và chủ yếu sống ở Vương Quốc Anh.
- Họ có xu hướng hủy một số giao dịch, với mức độ hủy và tỷ lệ hủy trung bình.
- Giá trị giao dịch trung bình của họ là khá cao, có nghĩa là khi họ mua sắm, họ thường thực hiện các giao dịch lớn.

- Cụm 2 (Biểu đồ Xanh Dương): Những Người Mua Hàng Tổng Chi Tiêu Cao với Tỉ Lệ Hủy Cao
  - Khách hàng trong nhóm này là người mua có mức chi tiêu cao với tổng chi tiêu rất cao và họ mua một loạt các sản phẩm độc đáo.
  - Họ thực hiện các giao dịch thường xuyên, nhưng cũng có mức độ hủy giao dịch và tỷ lệ hủy cao.
  - Những khách hàng này có khoảng thời gian trung bình giữa các giao dịch rất thấp, và họ thường mua sắm vào sáng sớm (giá trị Hour thấp).
  - Chi tiêu hàng tháng của họ có biến động cao, cho thấy mô hình chi tiêu của họ có thể ít dự đoán hơn so với các nhóm khác.
  - Mặc dù chi tiêu của họ cao, họ thể hiện mức độ chi tiêu thấp, ngụ ý rằng mức chi tiêu cao của họ có thể đang giảm theo thời gian.

### 3.2. Hệ thống gợi ý

Nhóm sẽ tạo ra một hệ thống gợi ý sản phẩm dựa trên cách mua hàng của từng nhóm khách hàng. Nó sẽ đề xuất ba sản phẩm phổ biến nhất mà khách hàng chưa mua, dựa trên cụm của họ. Đối với nhóm ngoại lệ, chúng ta có thể đề xuất các sản phẩm ngẫu nhiên để bắt đầu tương tác.

| CustomerID | Rec1_ProductID | Rec1_Description | Rec2_ProductID | Rec2_Description | Rec3_ProductID | Rec3_Description |
|---|---|---|---|---|---|---|
| **15746.0** | 84879 | ASSORTED COLOUR BIRD ORNAMENT | 15036 | ASSORTED COLOURS SILK FAN | 85123A | WHITE HANGING HEART T-LIGHT HOLDER |
| **15728.0** | 84077 | WORLD WAR 2 GLIDERS ASSTD DESIGNS | 84879 | ASSORTED COLOUR BIRD ORNAMENT | 15036 | ASSORTED COLOURS SILK FAN |
| **17459.0** | 18007 | ESSENTIAL BALM 3.5G TIN IN ENVELOPE | 84879 | ASSORTED COLOUR BIRD ORNAMENT | 17003 | BROCADE RING PURSE |
| **17415.0** | 18007 | ESSENTIAL BALM 3.5G TIN IN ENVELOPE | 84879 | ASSORTED COLOUR BIRD ORNAMENT | 17003 | BROCADE RING PURSE |
| **15339.0** | 18007 | ESSENTIAL BALM 3.5G TIN IN ENVELOPE | 84879 | ASSORTED COLOUR BIRD ORNAMENT | 17003 | BROCADE RING PURSE |
| **14335.0** | 84077 | WORLD WAR 2 GLIDERS ASSTD DESIGNS | 84879 | ASSORTED COLOUR BIRD ORNAMENT | 15036 | ASSORTED COLOURS SILK FAN |
| **15367.0** | 22616 | PACK OF 12 LONDON TISSUES | 84879 | ASSORTED COLOUR BIRD ORNAMENT | 16014 | SMALL CHINESE STYLE SCISSOR |
| **17604.0** | 84077 | WORLD WAR 2 GLIDERS ASSTD DESIGNS | 84879 | ASSORTED COLOUR BIRD ORNAMENT | 15036 | ASSORTED COLOURS SILK FAN |
| **17828.0** | 22616 | PACK OF 12 LONDON TISSUES | 84077 | WORLD WAR 2 GLIDERS ASSTD DESIGNS | 85099B | JUMBO BAG RED RETROSPOT |
| **13229.0** | 84077 | WORLD WAR 2 GLIDERS ASSTD DESIGNS | 84879 | ASSORTED COLOUR BIRD ORNAMENT | 15036 | ASSORTED COLOURS SILK FAN |

# CHƯƠNG IV. KẾT LUẬN

Trong cuộc cạnh tranh không ngừng của thị trường thương mại điện tử, việc hiểu rõ và tương tác hiệu quả với khách hàng đã trở thành yếu tố quyết định giữa sự thành công và thất bại của một doanh nghiệp. Trong dự án này, nhóm đã chứng minh sức mạnh của khoa học dữ liệu và phân tích thông tin trong việc biến dữ liệu thành những thông tin quý giá, từ đó tạo ra giá trị kinh doanh đích thực.

Qua việc phân loại khách hàng thành các nhóm đặc trưng và xây dựng hệ thống gợi ý sản phẩm, nhóm đã tạo ra một cơ sở vững chắc cho các chiến lược tiếp thị cá nhân hóa và tăng trải nghiệm mua sắm. Hệ thống gợi ý sản phẩm không chỉ là một công cụ tiếp thị thông thường, mà còn là một trải nghiệm mua sắm tùy chỉnh và độc đáo cho từng khách hàng.

Hy vọng rằng, doanh nghiệp sẽ nhận ra sức mạnh của dữ liệu và công nghệ trong việc xây dựng mối quan hệ chặt chẽ hơn với khách hàng và đạt được sự thành công trong thị trường ngày càng cạnh tranh của thương mại điện tử.

# TÀI LIỆU THAM KHẢO

[1] Mì AI. “Thử làm model phân khúc khách hàng bằng RFM và Kmean”. Youtube, July 26th, 2021, https://www.youtube.com/watch?v=I3Oo__t_mks

[2] Prateek. “K Means Clustering Algorithm”. Keytodatascience, On March 20, 2022, https://keytodatascience.com/k-means-clustering-algorithm/

[3] Đỗ Minh Đức. “Phân tích RFM là gì? Phân loại khách hàng theo mô hình RFM”. Bizfly, November 9th, 2023, https://bizfly.vn/techblog/phan-tich-rfm-la-gi-phan-loai-khach-hang-theo-mo-hinh-rfm.html/

[4] sellshock1911.”creating-customer-segments: Unsupervised Learning for market segmentation”. GitHub: https://github.com/shellshock1911/Creating-Customer-Segment.