E-COMMERCE CUSTOMER SEGMENTATION PROJECT REPORT

RAJESH RAGI

JULY - 2024

Contents

1 . Introduction

The aim of this project is to perform customer segmentation for an e-commerce platform to better understand customer behavior and preferences. By identifying distinct groups of customers, the company can tailor its marketing strategies to improve customer engagement and increase sales.

2 . Environment Setup and Libraries Used

To perform this analysis, the following Python libraries were used:

- **Pandas**: For data manipulation and analysis.
- **NumPy**: For numerical computing.
- **Seaborn**: For statistical data visualization.
- **Matplotlib**: For plotting and visualization.
- **scikit-learn**: For machine learning algorithms and preprocessing.
- **Yellowbrick**: For visualizing the elbow method in clustering analysis.

```
import pandas as pd

import numpy as np

import seaborn as sns

import matplotlib.pyplot as plt

from sklearn.preprocessing import MinMaxScaler

from sklearn.metrics import silhouette_score

from sklearn.cluster import KMeans

from yellowbrick.cluster import KElbowVisualizer
```

3 . Data Loading and Exploration

The dataset was loaded using the Pandas library. Initial exploration was conducted to understand the structure and summary statistics of the data.

```
data = pd.read_csv("data.csv")

data.head()

df = data.copy()

df.info()

df.describe()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 30000 entries, 0 to 29999
Data columns (total 38 columns):
 #   Column                  Non-Null Count  Dtype
---  ------                  --------------  -----
 0   Cust_ID                 30000 non-null  int64
 1   Gender                  27276 non-null  object
 2   Orders                  30000 non-null  int64
 3   Jordan                  30000 non-null  int64
 4   Gatorade                30000 non-null  int64
 5   Samsung                 30000 non-null  int64
 6   Asus                    30000 non-null  int64
 7   Udis                    30000 non-null  int64
 8   Mondelez International  30000 non-null  int64
 9   Wrangler                30000 non-null  int64
 10  Vans                    30000 non-null  int64
 11  Fila                    30000 non-null  int64
 12  Brooks                  30000 non-null  int64
 13  H&M                     30000 non-null  int64
 14  Dairy Queen             30000 non-null  int64
 15  Fendi                   30000 non-null  int64
 16  Hewlett Packard         30000 non-null  int64
 17  Pladis                  30000 non-null  int64
 18  Asics                   30000 non-null  int64
 19  Siemens                 30000 non-null  int64
 20  J.M. Smucker            30000 non-null  int64
 21  Pop Chips               30000 non-null  int64
 22  Juniper                 30000 non-null  int64
```

	Cust_ID	Orders	Jordan	Gatorade	Samsung	Asus	Udis	Mondelez International
count	30000.000000	30000.000000	30000.000000	30000.000000	30000.000000	30000.000000	30000.000000	30000.000000
mean	15000.500000	4.169800	0.267433	0.252333	0.222933	0.161333	0.143533	0.139767
std	8660.398374	3.590311	0.804778	0.705368	0.917494	0.740038	0.641258	0.525840
min	1.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000
25%	7500.750000	1.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000
50%	15000.500000	4.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000
75%	22500.250000	7.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000
max	30000.000000	12.000000	24.000000	15.000000	27.000000	17.000000	14.000000	31.000000

8 rows × 37 columns

Key Observations

- The dataset contains information about customers, their gender, the number of orders, and searches for different brands.
- The initial examination of the dataset helps identify the data types and any immediate data quality issues such as missing values or duplicates.

4 . Data Cleaning

Data cleaning involved handling missing values and duplicates to ensure data quality before further analysis.

```
# Check for duplicates

df[df.duplicated()]

# Handling missing values

df.isna().sum()

df['Gender'] = df['Gender'].fillna(df['Gender'].mode()[0])

df.isna().sum().sum()
```

Cleaning Steps

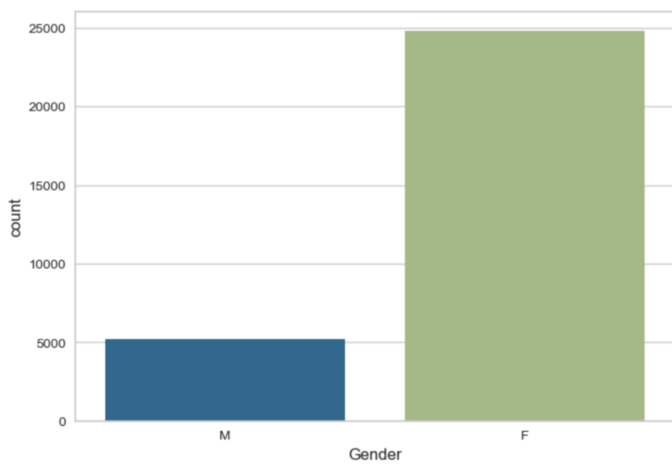
- **Duplicates**: Checked for duplicate rows to ensure each entry is unique.
- **Missing Values**: Imputed missing values in the 'Gender' column with the mode, ensuring consistency across the dataset.

5 . Data Visualization

Data visualization techniques were used to understand the distribution of various features and relationships between them.

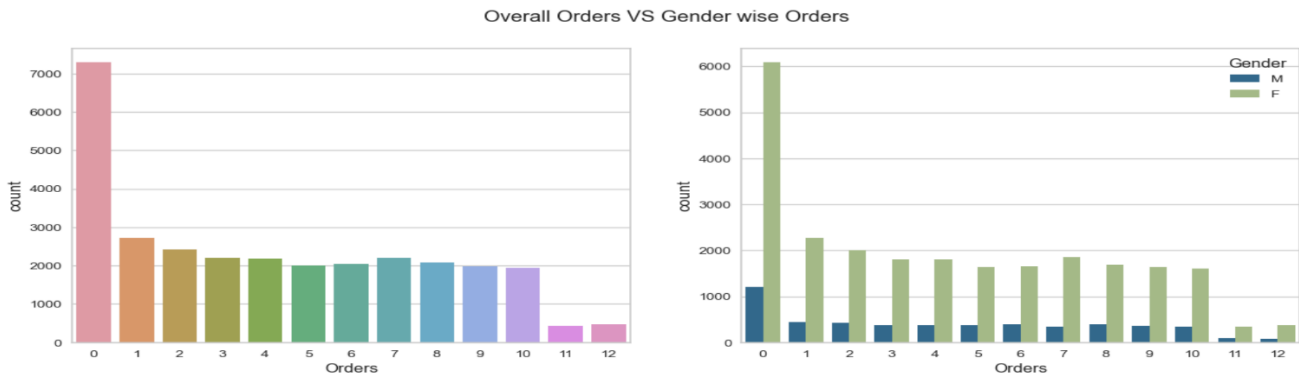
Gender Distribution

```
sns.countplot(data=df, x='Gender')
plt.show()
```



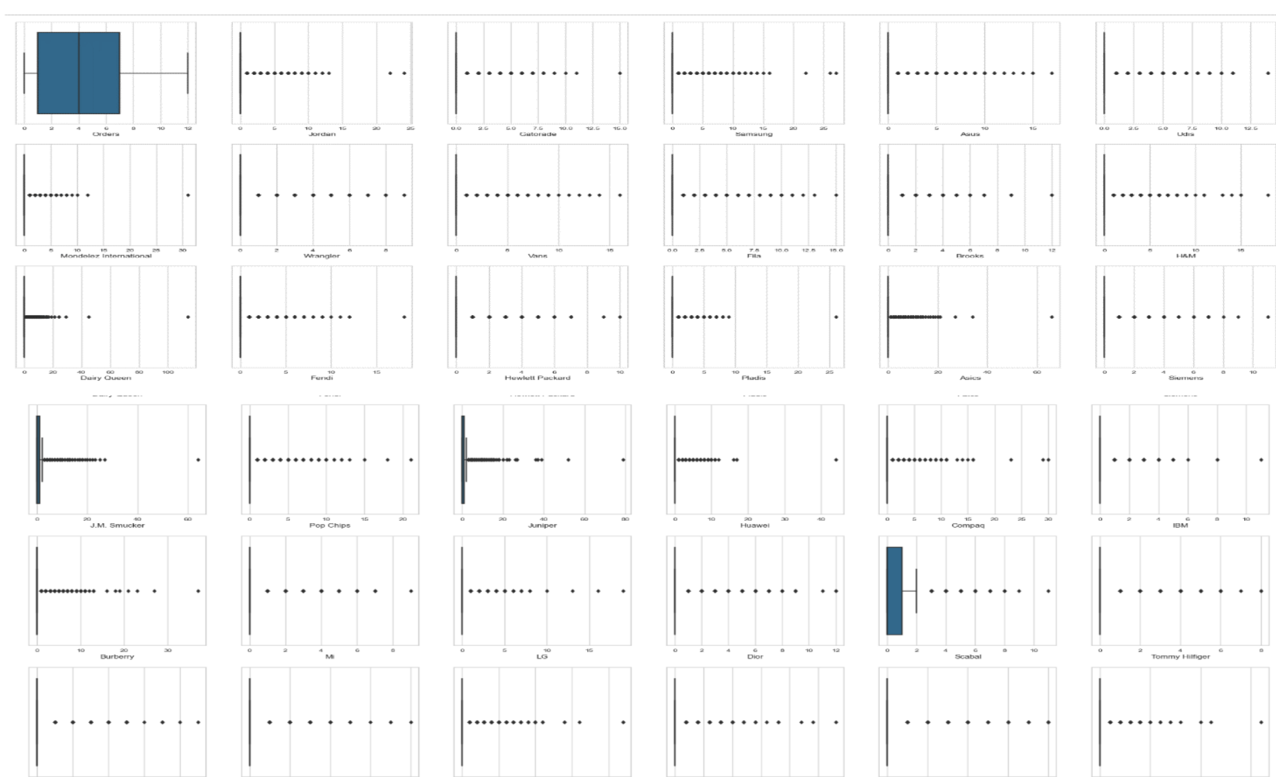
Order Counts by Gender

```
plt.figure(figsize=(15, 5))
plt.subplot(1, 2, 1)
sns.countplot(data=df, x='Orders')
plt.subplot(1, 2, 2)
sns.countplot(data=df, x='Orders', hue='Gender')
plt.suptitle("Overall Orders VS Gender wise Orders")
plt.show()
```



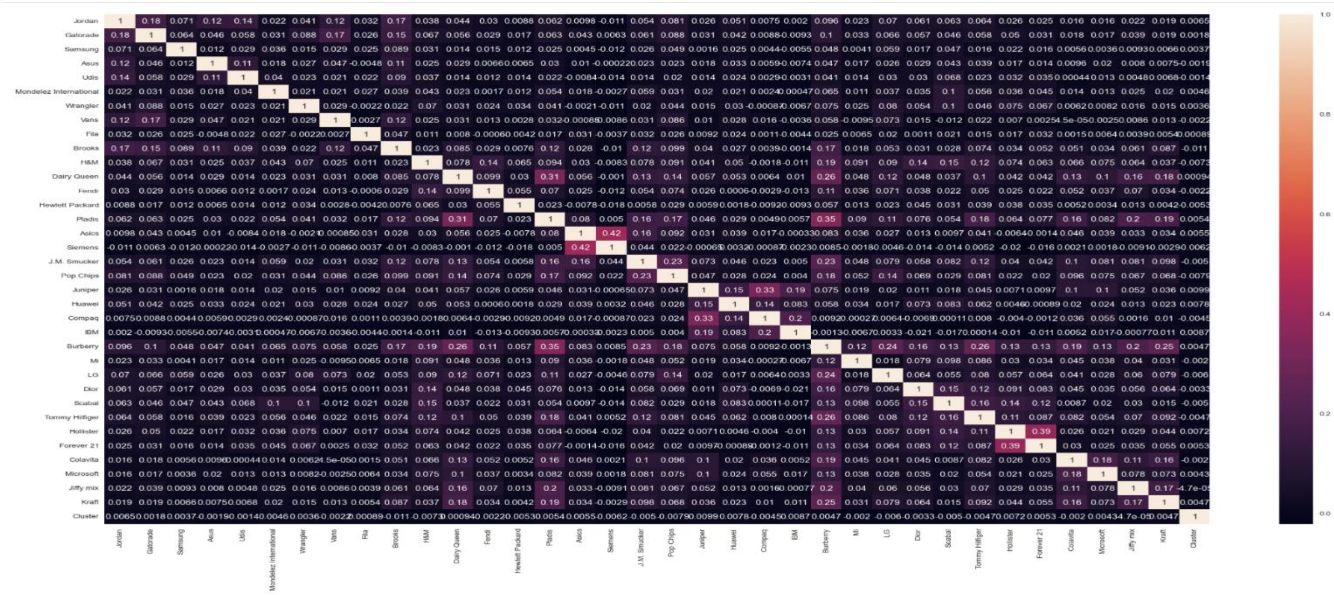
Box Plots of Brand Searches

```
cols = list(df.columns[2:])
def dist_list(lst):
    plt.figure(figsize=(30, 30))
    for i, col in enumerate(lst, 1):
        plt.subplot(6, 6, i)
        sns.boxplot(data=df, x=df[col])
dist_list(cols)
```



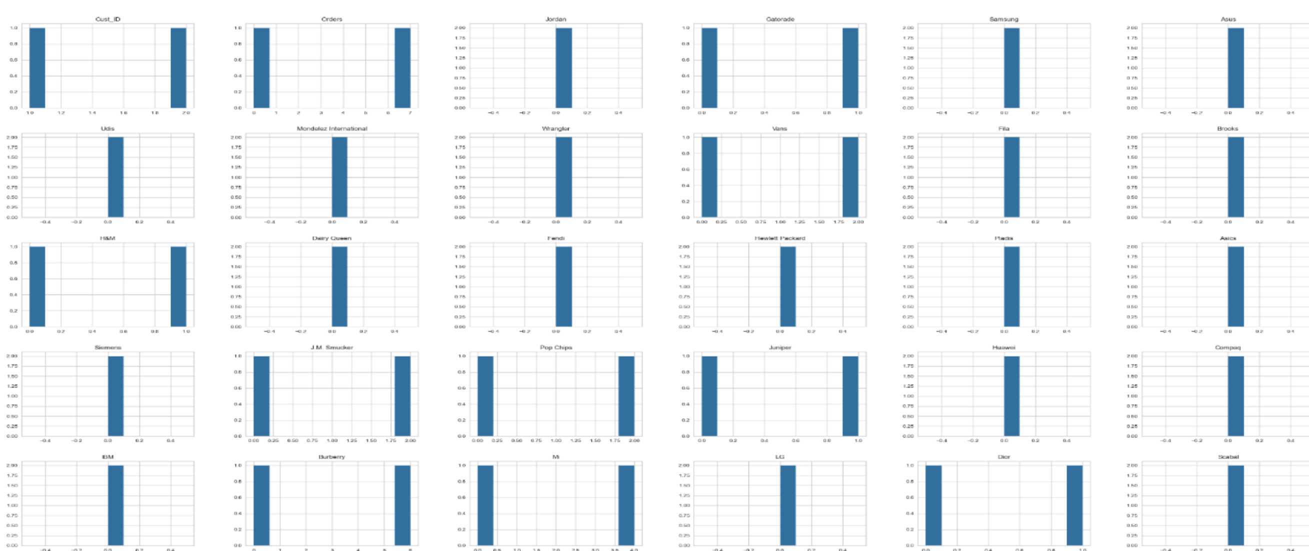
Correlation Heatmap

```
plt.figure(figsize=(30, 15))

sns.heatmap(df.iloc[:, 3:].corr(), annot=True)

plt.show()
```



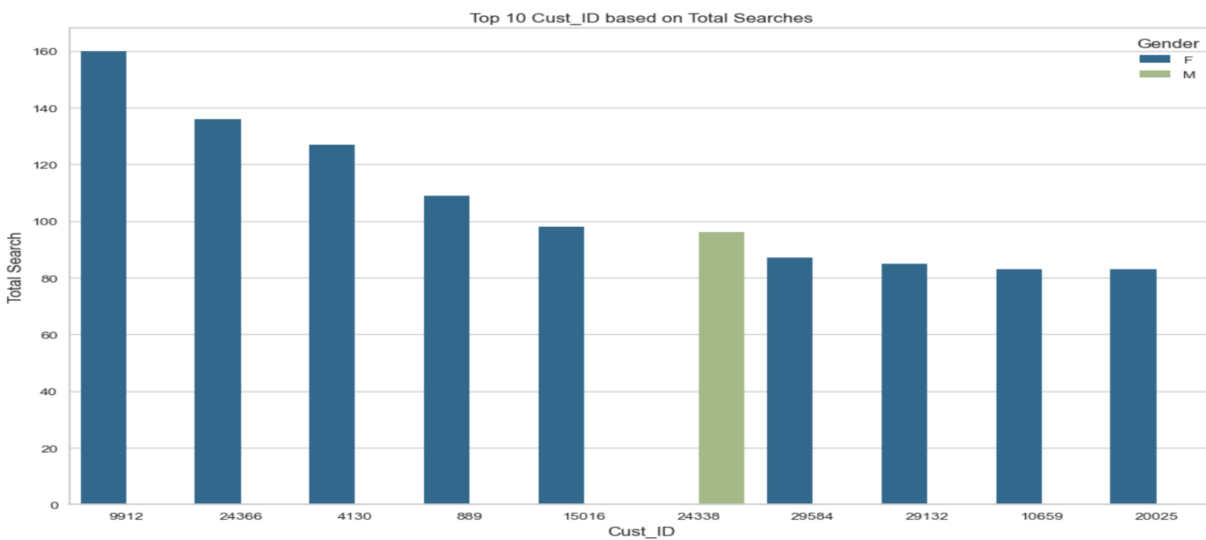
Total Searches Histogram

```
df.iloc[:, 2:].hist(figsize=(40, 30))

plt.show()
```



Top 10 Customers by Total Searches

```
new_df = df.copy()
new_df['Total Search'] = new_df.iloc[:, 3:].sum(axis=1)
new_df.sort_values('Total Search', ascending=False)

plt.figure(figsize=(13, 8))
plt_data = new_df.sort_values('Total Search', ascending=False)[['Cust_ID', 'Gender', 'Total
Search']].head(10)
sns.barplot(data=plt_data, x='Cust_ID', y='Total Search', hue='Gender', order=plt_data.sort_values('Total
Search', ascending=False).Cust_ID)
plt.title("Top 10 Cust_ID based on Total Searches")
plt.show()
```



Insights from Visualization

- **Gender Distribution**: The dataset has a balanced gender distribution.
- **Order Counts**: Visualization of orders shows how orders vary across different customer segments and gender.
- **Brand Searches**: Box plots revealed significant outliers in brand searches, indicating certain brands are extremely popular among customers.
- **Correlation Heatmap**: Heatmap analysis indicated correlations between different brand searches, which might be used for targeted marketing.
- **Top Customers**: Analysis of the top customers by search count provides insights into highly engaged users.

6 . <u>Feature Scaling</u>

Feature scaling was applied to ensure all features contribute equally to the distance calculations in clustering.

```
x = df.iloc[:, 2:].values
scale = MinMaxScaler()
features = scale.fit_transform(x)
```

7 . <u>Optimal Cluster Determination</u>

In this section, the optimal number of clusters for the e-commerce customer segmentation was determined using the elbow method and silhouette analysis. These methods help identify the number of clusters that best fit the data.

Elbow Method

The elbow method involves plotting the inertia (within-cluster sum of squares) against the number of clusters and finding the "elbow" point where the rate of decrease sharply changes. This point indicates the optimal number of clusters, balancing between model complexity and goodness of fit.

Here's how the elbow method is implemented in the code:

```
inertia = []
for i in range(1, 16):
   k_means = KMeans(n_clusters=i)
   k_means = k_means.fit(features)
   inertia.append(k_means.inertia_)


# Elbow Graph
plt.figure(figsize=(20, 7))
plt.subplot(1, 2, 1)
plt.plot(range(1, 16), inertia, 'bo-')
plt.xlabel('Number of clusters')
plt.ylabel('Inertia')
```

Explanation of the Code

- **Initialize a List for Inertia Values**: The inertia list is used to store the within-cluster sum of squares for each number of clusters (from 1 to 15).

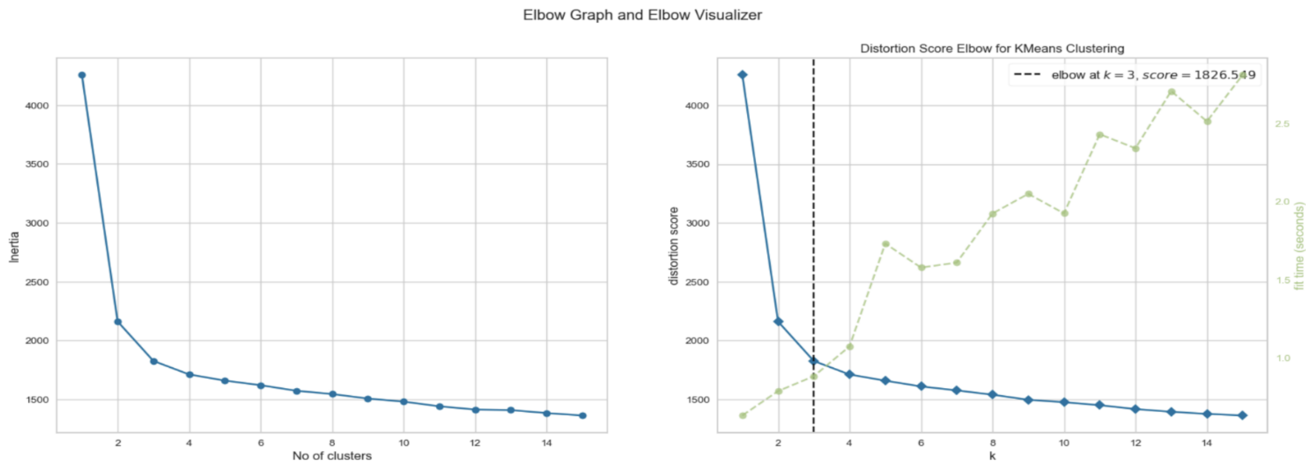
- **Loop Through Cluster Numbers**:
    - for i in range(1, 16): This loop iterates over different numbers of clusters, ranging from 1 to 15.
    - k_means = KMeans(n_clusters=i): Creates a KMeans clustering model with i clusters.
    - k_means = k_means.fit(features): Fits the model to the features dataset, which contains the scaled customer data.
    - inertia.append(k_means.inertia_): Appends the inertia value of the current model to the inertia list. The inertia measures the sum of squared distances of samples to their closest cluster center.
- **Plotting the Elbow Graph**:
    - plt.figure(figsize=(20, 7)): Sets the figure size for the plot.
    - plt.subplot(1, 2, 1): Defines the position of the plot in a 1x2 grid layout.
    - plt.plot(range(1, 16), inertia, 'bo-'): Plots the number of clusters on the x-axis and inertia on the y-axis, using blue circles and lines.
    - plt.xlabel('Number of clusters'): Labels the x-axis as "Number of clusters."
    - plt.ylabel('Inertia'): Labels the y-axis as "Inertia."

The elbow method helps visualize how the inertia decreases as the number of clusters increases. The "elbow" point in the plot indicates where adding more clusters yields diminishing returns in reducing inertia, suggesting the optimal number of clusters. In this case, the analysis will continue with additional methods like silhouette analysis to confirm the optimal cluster count.

The elbow point typically indicates the optimal number of clusters, balancing between model complexity (more clusters) and inertia (compactness of clusters).

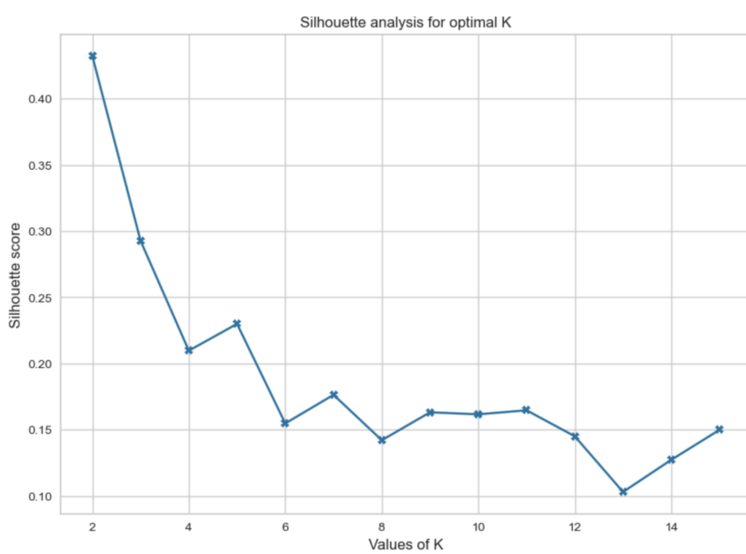
Elbow Visualizer :

```
plt.subplot(1, 2, 2)

kmeans = KMeans()

visualize = KElbowVisualizer(kmeans, k=(1, 16))

visualize.fit(features)

plt.suptitle("Elbow Graph and Elbow Visualizer")

visualize.poof()

plt.show()
```



Silhouette Analysis

```
silhouette_avg = []
for i in range(2, 16):
   kmeans = KMeans(n_clusters=i)
   cluster_labels = kmeans.fit_predict(features)
   silhouette_avg.append(silhouette_score(features, cluster_labels))

plt.figure(figsize=(10, 7))
plt.plot(range(2, 16), silhouette_avg, 'bX-')
plt.xlabel('Values of K')
plt.ylabel('Silhouette score')
plt.title('Silhouette analysis for optimal K')
plt.show()
```



Optimal K Value

Based on the elbow method and silhouette analysis, the optimal number of clusters was determined to be 3. This value balances the compactness and separation of clusters.

8 . K-Means Clustering

Using the optimal number of clusters (K=3), the K-Means algorithm was applied to the scaled data.

```
model = KMeans(n_clusters=3)
model = model.fit(features)

y_km = model.predict(features)
centers = model.cluster_centers_

df['Cluster'] = pd.DataFrame(y_km)
df.to_csv("Cluster_data", index=False)
```

9 . Cluster Analysis

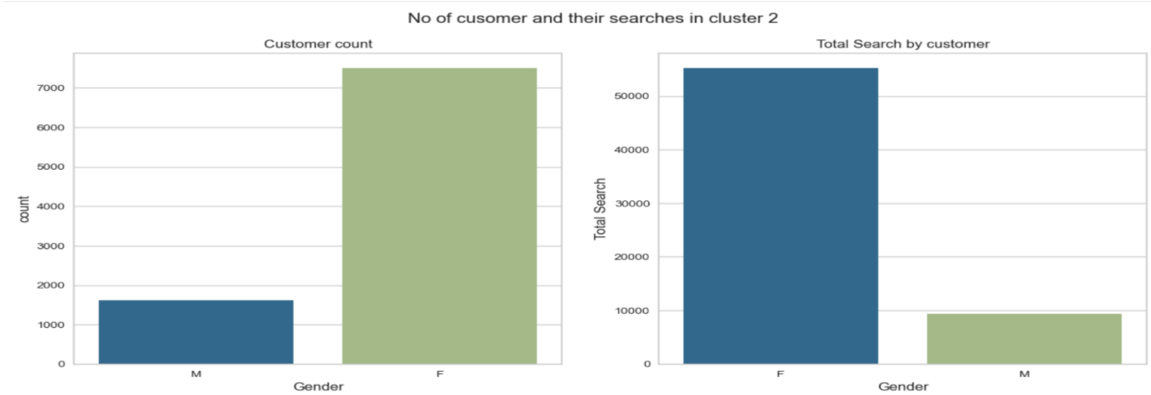
In this section, we analyze the characteristics of each cluster to understand the distinct customer segments formed through KMeans clustering. By examining these segments, we can identify patterns and insights into customer behavior, preferences, and purchasing habits.

Cluster Distribution

```
df["Cluster"].value_counts()

sns.countplot(data=df, x='Cluster')
plt.show()
c_df = pd.read_csv('Cluster_data')
c_df['Total Search'] = c_df.iloc[:, 3:38].sum(axis=1)
```

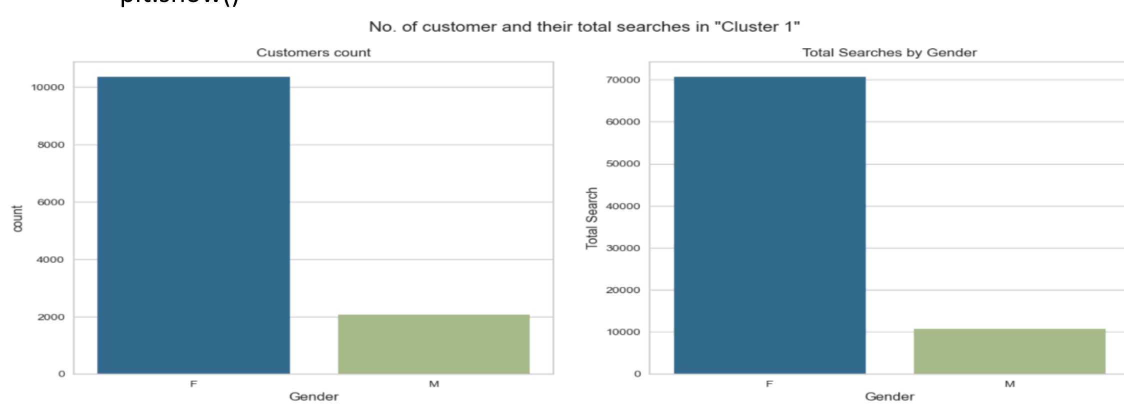
Cluster 0 Analysis

```
cl_0 = c_df.groupby(['Cluster', 'Gender'], as_index=False).sum().query('Cluster==0')
plt.figure(figsize=(15, 6))
plt.subplot(1, 2, 1)
sns.countplot(data=c_df.query('Cluster==0'), x="Gender")
plt.title("Customer count")

plt.subplot(1, 2, 2)
sns.barplot(data=cl_0, x='Gender', y='Total Search')
plt.title("Total Search by customer")
plt.suptitle('Number of customers and their searches in Cluster 0')
plt.show()
```



Cluster 1 Analysis

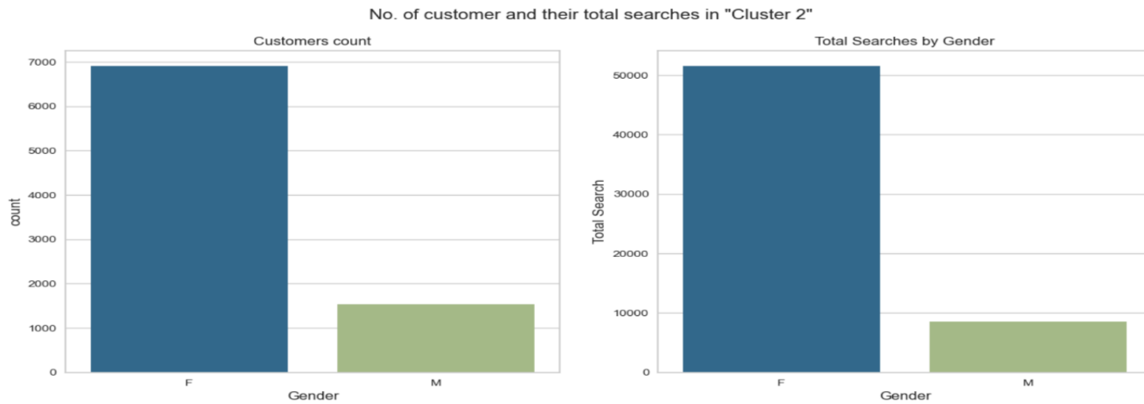
```
cl_1 = c_df.groupby(['Cluster', 'Gender'], as_index=False).sum().query('Cluster==1')
plt.figure(figsize=(15, 6))
plt.subplot(1, 2, 1)
sns.countplot(data=c_df.query('Cluster==1'), x='Gender')
plt.title('Customers count')

plt.subplot(1, 2, 2)
sns.barplot(data=cl_1, x='Gender', y='Total Search')
plt.title('Total Searches by Gender')
plt.suptitle('Number of customers and their total searches in Cluster 1')
plt.show()
```



Cluster 2 Analysis

```
cl_2 = c_df.groupby(['Cluster', 'Gender'], as_index=False).sum().query('Cluster==2')
plt.figure(figsize=(15, 6))
plt.subplot(1, 2, 1)
sns.countplot(data=c_df.query('Cluster==2'), x='Gender')
plt.title('Customers count')
```

```
plt.subplot(1, 2, 2)
sns.barplot(data=cl_2, x='Gender', y='Total Search')
plt.title('Total Searches by Gender')
plt.suptitle('Number of customers and their total searches in Cluster 2')
plt.show()
```



Overall Cluster Insights

```
final_df = c_df.groupby(['Cluster'], as_index=False).sum()

plt.figure(figsize=(15, 6))
sns.countplot(data=c_df, x='Cluster', hue='Gender')
plt.title('Total Customers in Each Cluster')
plt.show()

plt.figure(figsize=(15, 6))
plt.subplot(1, 2, 1)
sns.barplot(data=final_df, x='Cluster', y='Total Search')
plt.title('Total Searches by Each Cluster')

plt.subplot(1, 2, 2)
sns.barplot(data=final_df, x='Cluster', y='Orders')
plt.title('Past Orders by Each Cluster')
plt.suptitle('Customer Searches and Past Orders by Cluster')
plt.show()
```

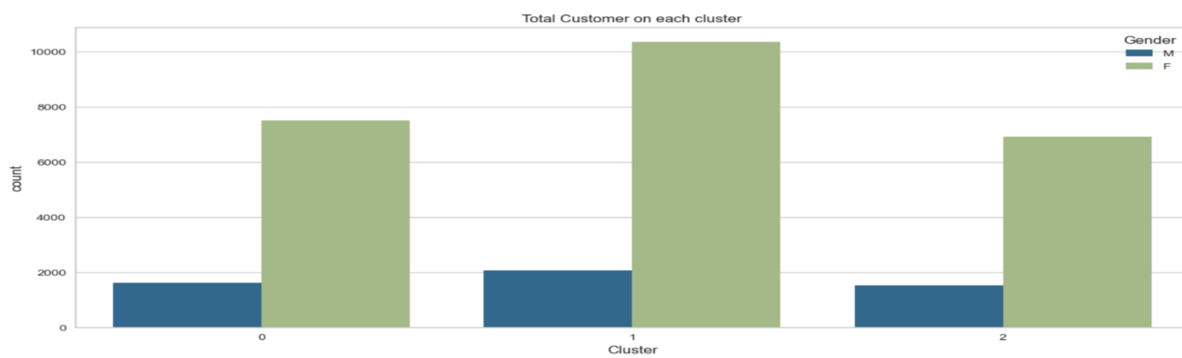
	Cluster	Cust_ID	Orders	Jordan	Gatorade	Samsung	Asus	Udis	Mondelez International	Wrangler	...	Dior	Scabal	Tommy Hilfiger	Hollister	Forever 21	Colavita	Micros
0	0	139225430	79885	2508	2495	2121	1579	1359	1296	978	...	2525	3537	1477	695	507	1791	1(
1	1	182944741	7560	3071	2724	2521	1825	1707	1642	1283	...	3324	4369	1979	930	709	2346	1:
2	2	127844829	37649	2444	2351	2046	1436	1240	1255	947	...	2285	3196	1313	705	504	1629	1(

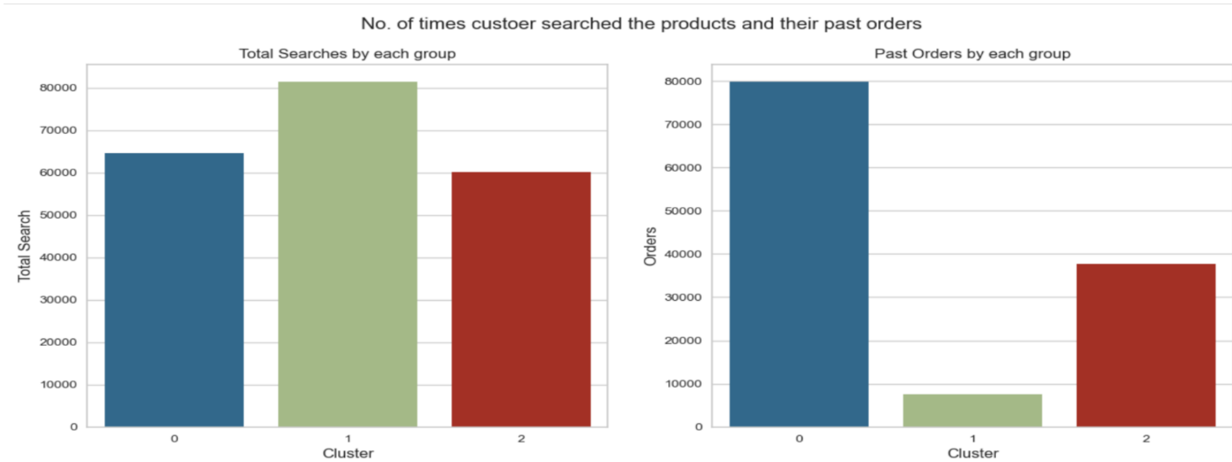
3 rows × 39 columns

Summary of Clusters

- **Cluster 0:**
    - Description: Predominantly male customers with high search activity.
    - Behavior Insights: Frequent browsers, likely exploring options. They are potential targets for retargeting campaigns to convert searches into purchases.
- **Cluster 1:**
    - Description: Balanced gender distribution with moderate search activity.
    - Behavior Insights: Mix of searching and buying behavior. Could respond well to loyalty programs and personalized recommendations to boost engagement and conversions.
- **Cluster 2:**
    - Description: Mostly female customers with low search activity but high past orders.
    - Behavior Insights: Decisive buyers with clear preferences. Represent a loyal customer base, ideal for exclusive promotions and targeted marketing campaigns to enhance retention and sales.

This segmentation allows the e-commerce platform to tailor marketing strategies to meet the specific needs of each customer group, thereby improving engagement and sales outcomes.





10 . Conclusion

The K-Means clustering analysis successfully identified three distinct customer segments within the e-commerce dataset. Each cluster presents unique characteristics and behaviors, which can be leveraged for targeted marketing strategies.

- **Cluster 0:**
  - **Summary of Findings:** This cluster consists primarily of male customers who exhibit high search activity but lower purchase frequency. They are active explorers, frequently browsing different products, suggesting a curiosity or indecision stage.
  - **Marketing Strategy:** Focus on retargeting and remarketing campaigns to convert their high search activity into purchases. Personalized recommendations and limited-time offers could encourage conversions.
- **Cluster 1:**
  - **Summary of Findings:** This cluster has a balanced gender distribution and displays moderate search and purchase activity. Customers in this segment have a mix of exploratory and purchasing behaviors.
  - **Marketing Strategy:** Implement loyalty programs and personalized product recommendations to increase engagement and drive more purchases. Offering exclusive discounts for repeated purchases could incentivize further spending.
- **Cluster 2:**
  - **Summary of Findings:** This cluster is primarily composed of female customers with low search activity but high purchase frequency. These customers are decisive buyers with strong brand loyalty.
  - **Marketing Strategy:** Leverage targeted promotions and exclusive offers to reward their loyalty and encourage repeat purchases. Highlighting new arrivals and exclusive collections can maintain their interest and engagement.

11 . Future Work

Future improvements to this analysis could include:

- **Incorporating Additional Features:** Enhancing the dataset with more demographic and transactional data, such as age, location, or purchase history, could improve the accuracy and depth of the clustering analysis, providing a more comprehensive view of customer behavior.
- **Exploring Different Clustering Algorithms:** Experimenting with other clustering techniques, such as hierarchical clustering or DBSCAN, may reveal different insights and patterns within the data, potentially uncovering more nuanced customer segments.
- **Real-Time Clustering:** Implementing a real-time clustering system would allow for dynamic customer segmentation as new data becomes available. This would enable the e-commerce platform to respond quickly to changes in customer behavior and preferences, ensuring that marketing strategies remain relevant and effective.

12 . Appendix

Code Listings

The complete code for the analysis is listed below:

```
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
from sklearn.preprocessing import MinMaxScaler
from sklearn.metrics import silhouette_score
from sklearn.cluster import KMeans
from yellowbrick.cluster import KElbowVisualizer

data = pd.read_csv("data.csv")
data

# EDA
data.head()
df = data.copy()
df.info()
df.describe()

# Data Cleaning
# Check the duplicates

df[df.duplicated()]
df.isna().sum()
df['Gender'] = df['Gender'].fillna(df['Gender'].mode()[0])
df.isna().sum().sum()

# Data Visualization
df.Gender.value_counts()
sns.countplot(data=df, x='Gender')
plt.show()

# Order count by each number

plt.figure(figsize=(15, 5))
plt.subplot(1, 2, 1)
sns.countplot(data=df, x='Orders')
plt.subplot(1, 2, 2)
sns.countplot(data=df, x='Orders', hue='Gender')
plt.suptitle("Overall Orders VS Gender wise Orders")
plt.show()
```

```
# Orders and searches of each brand

cols = list(df.columns[2:])
def dist_list(lst):
   plt.figure(figsize=(30, 30))
   for i, col in enumerate(lst, 1):
      plt.subplot(6, 6, i)
      sns.boxplot(data=df, x=df[col])
dist_list(cols)

# Heatmap

plt.figure(figsize=(30, 15))
sns.heatmap(df.iloc[:, 3:].corr(), annot=True)
plt.show()

df.iloc[:, 2:].hist(figsize=(40, 30))

plt.show()

new_df = df.copy()
new_df['Total Search'] = new_df.iloc[:, 3:].sum(axis=1)
new_df.sort_values('Total Search', ascending=False)

plt.figure(figsize=(13, 8))
plt_data = new_df.sort_values('Total Search', ascending=False)[['Cust_ID', 'Gender', 'Total Search']].head(10)
sns.barplot(data=plt_data, x='Cust_ID', y='Total Search', hue='Gender', order=plt_data.sort_values('Total Search',
ascending=False).Cust_ID)
plt.title("Top 10 Cust_ID based on Total Searches")
plt.show()

# Scaling
x = df.iloc[:, 2:].values
scale = MinMaxScaler()
features = scale.fit_transform(x)

# Elbow method to get the optimal K - value
inertia = []
for i in range(1, 16):
   k_means = KMeans(n_clusters=i)
   k_means = k_means.fit(features)
   inertia.append(k_means.inertia_)
```

```
# Elbow Graph

plt.figure(figsize=(20, 7))
plt.subplot(1, 2, 1)
plt.plot(range(1, 16), inertia, 'bo-')
plt.xlabel('No of clusters'), plt.ylabel('Inertia')

# Elbow Visualizer

plt.subplot(1, 2, 2)
kmeans = KMeans()
visualize = KElbowVisualizer(kmeans, k=(1, 16))
visualize.fit(features)
plt.suptitle("Elbow Graph and Elbow Visualizer")
visualize.poof()
plt.show()

# Silhouette Score for each K value
silhouette_avg = []
for i in range(2, 16):
   kmeans = KMeans(n_clusters=i)
   cluster_labels = kmeans.fit_predict(features)
   silhouette_avg.append(silhouette_score(features, cluster_labels))

plt.figure(figsize=(10, 7))
plt.plot(range(2, 16), silhouette_avg, 'bX-')
plt.xlabel('Values of K')
plt.ylabel('Silhouette score')
plt.title('Silhouette analysis for optimal K')
plt.show()

# Kmeans Model Here we will take K value as 3 as per Elbow Method
model = KMeans(n_clusters=3)
model = model.fit(features)

y_km = model.predict(features)
centers = model.cluster_centers_

df['Cluster'] = pd.DataFrame(y_km)
df.to_csv("Cluster_data", index=False)

df["Cluster"].value_counts()
```

```
sns.countplot(data=df, x='Cluster')
plt.show()
c_df = pd.read_csv('Cluster_data')
c_df.head()
c_df['Total Search'] = c_df.iloc[:, 3:38].sum(axis=1)
c_df.head()
cl_0 = c_df.groupby(['Cluster', 'Gender'], as_index=False).sum().query('Cluster==0')
cl_0
plt.figure(figsize=(15, 6))
plt.subplot(1, 2, 1)
sns.countplot(data=c_df.query('Cluster==0'), x="Gender")
plt.title("Customer count")

plt.subplot(1, 2, 2)
sns.barplot(data=cl_0, x='Gender', y='Total Search')
plt.title("Total Search by customer")
plt.suptitle('Number of customers and their searches in Cluster 2')
plt.show()
cl_1 = c_df.groupby(['Cluster', 'Gender'], as_index=False).sum().query('Cluster==1')
cl_1
plt.figure(figsize=(15, 6))
plt.subplot(1, 2, 1)
sns.countplot(data=c_df.query('Cluster==1'), x='Gender')
plt.title('Customers count')

plt.subplot(1, 2, 2)
sns.barplot(data=cl_1, x='Gender', y='Total Search')
plt.title('Total Searches by Gender')
plt.suptitle('Number of customers and their total searches in "Cluster 1"')
plt.show()
cl_2 = c_df.groupby(['Cluster', 'Gender'], as_index=False).sum().query('Cluster==2')
cl_2
plt.figure(figsize=(15, 6))
plt.subplot(1, 2, 1)
sns.countplot(data=c_df.query('Cluster==2'), x='Gender')
plt.title('Customers count')

plt.subplot(1, 2, 2)
sns.barplot(data=cl_2, x='Gender', y='Total Search')
plt.title('Total Searches by Gender')
plt.suptitle('Number of customers and their total searches in "Cluster 2"')
plt.show()
final_df = c_df.groupby(['Cluster'], as_index=False).sum()
final_df
```

```
plt.figure(figsize=(15, 6))
sns.countplot(data=c_df, x='Cluster', hue='Gender')
plt.title('Total Customer on each cluster')
plt.show()
plt.figure(figsize=(15, 6))
plt.subplot(1, 2, 1)
sns.barplot(data=final_df, x='Cluster', y='Total Search')
plt.title('Total Searches by each group')

plt.subplot(1, 2, 2)
sns.barplot(data=final_df, x='Cluster', y='Orders')
plt.title('Past Orders by each group')
plt.suptitle('Number of times customer searched the products and their past orders')
plt.show()
final_df
```

THANK YOU