CUSTOMER SEGEMENTATION

Submitted by,
Sri Tarunika G P

1. Introduction

In the dynamic landscape of modern businesses, understanding customer behavior is a pivotal factor for strategic decision-making. Customer segmentation emerges as a powerful technique, enabling organizations to tailor their approaches to specific customer groups based on distinct purchasing patterns. This project aims to perform a comprehensive customer segmentation analysis using machine learning techniques, with a focus on a provided dataset of customer transactions. The Customer Segmentation project is a strategic initiative aimed at unraveling intricate patterns within customer behavior through advanced machine learning techniques. In the contemporary business landscape, understanding customer dynamics is imperative for tailored marketing strategies and enhanced customer engagement. Leveraging the capabilities of the K-Means clustering algorithm, this project seeks to segment customers into distinct groups, offering businesses valuable insights into purchasing patterns and preferences.

2. Data Preprocessing and Exploration

2.1 Dataset Overview

The dataset has been loaded successfully, containing information about customer transactions. It consists of various fields, including InvoiceNo, StockCode, Description, Quantity, UnitPrice, InvoiceDate, and CustomerID.

2.2 Initial Data Exploration

The dataset was explored to gain an understanding of its structure. The first five and last five records were displayed using the head() and tail() functions, respectively. Additionally, the size, shape, and basic information (info()) about the dataset were examined.

2.3 Handling Missing Values

Null values were identified and handled appropriately. The 'CustomerID' column was dropped, and the 'Description' column's missing values were filled using the mode value.

2.4 Removing Duplicates

Duplicate entries in the dataset were identified and removed, ensuring data integrity.

2.5 Data Visualization

Several visualizations were created to provide insights into the dataset. A bar plot of descriptive statistics for the 'Description' column and a dashboard showing the monthly sales trend were generated.

2.6 Handling Negative Values and Outliers

Negative values in the 'UnitPrice' column were removed. Outliers were detected using Z-scores, and potential outliers were visualized using box plots. Outliers were then removed from the dataset.

2.7 Histograms and Correlation Matrix

Histograms were plotted for numeric columns, providing a visual representation of their distributions. A correlation matrix was generated to explore relationships between numeric features.

3. Feature Engineering

3.1 Total Spending

A new feature, 'TotalSpending,' was created, representing the total spending of each customer based on quantity and unit price.

3.2 Frequency of Purchases

The 'Frequency' feature was introduced, capturing the frequency of customer purchases.

3.3 Recency of Purchases

The 'Recency' feature was calculated, reflecting the recency of each customer's last purchase.

3.4 Average Unit Price

The 'UnitPrice' feature was transformed to represent the average unit price for each customer.

4. Customer Segmentation using K-Means Clustering

4.1 Feature Scaling

Features were standardized using StandardScaler to ensure uniformity in scaling.

4.2 Optimal K Determination

The optimal number of clusters (K) was determined using the Elbow Method, with K=3 chosen based on the inertia plot.

4.3 K-Means Clustering

The K-Means clustering algorithm was implemented with K=3, and customers were assigned to distinct clusters.

4.4 Silhouette Score

The silhouette score was calculated to evaluate the clustering model's performance. The score indicates the compactness and separation of clusters.

5. Code Implementations

5.1 Code snippets

```
#!/usr/bin/env python
# coding: utf-8

# # Data Preprocessing and Exploration

# In[49]:


#import the necessary libraries
import pandas as pd
import numpy as np


# In[50]:


#load the dataset
data=pd.read_csv("data.csv",encoding='ISO-8859-1')


# In[51]:


#explore the dataset
data


# In[52]:


#head (display first five values)
data.head()
```

```
# In[53]:


#tail (display last five values)
data.tail()


# In[54]:


#size (display number of the elements)
data.size


# In[55]:


#shape(no.of rows and columns)
data.shape


# In[56]:


#info(displays the basic information of the dataset)
data.info()


# In[57]:


#describe(display the statistical information of the dataset)
data.describe()


# In[58]:


#isnull finding null values
data.isnull()
```

```
# In[59]:


#row wise null values
data.isnull().sum()


# In[60]:


#total null values
data.isnull().sum().sum()


# In[61]:


data.drop(["CustomerID"],axis=1)


# In[62]:


#handling the missing values in text feature
#1.description using filling the mode value
data["Description"].fillna(data["Description"].mode()[0],inplace=True)


# In[63]:


#displaying Description
data["Description"]


# In[64]:
```

```
#Checking the existence of null value
data["Description"].isnull().sum()


# In[65]:


data.columns


# In[66]:


#Finding the duplicate values
duplicate_rows = data[data.duplicated(keep=False)]

# Sorting the data by certain columns to see the duplicate rows next to
each other
duplicate_rows_sorted = duplicate_rows.sort_values(by=['InvoiceNo',
'StockCode', 'Description', 'CustomerID', 'Quantity'])

# Displaying the first 10 records
duplicate_rows_sorted.head(10)


# In[67]:


data.drop_duplicates(inplace=True)
data.head()


# In[68]:


data.shape[0]


# In[69]:
```

```
#Data visualization
#import the necessary libraries
import matplotlib.pyplot as plt


# In[70]:


#barh plot of descriptive
description_counts = data['Description'].value_counts()
description_counts


# In[71]:


#get the top 30 descriptions
top_30_descriptions = description_counts[:30]


# In[72]:


#plotting
plt.figure(figsize=(12,8))
plt.barh(top_30_descriptions.index[::-1],
top_30_descriptions.values[::-1], color='#000000')

# Adding labels and title
plt.xlabel('Number of Occurrences')
plt.ylabel('Description')
plt.title('Top 30 Most Frequent Descriptions')

# Show the plot
plt.show()


# In[73]:
```

```
# dashboard for Monthly sale
data.InvoiceDate = pd.to_datetime(data.InvoiceDate)
monthly_sales=data.groupby(pd.Grouper(key='InvoiceDate',freq='M'))['Quanti
ty'].sum()
monthly_sales.plot(kind='area', stacked=True,color="purple")
plt.title('monthly Sales Trend')
plt.xlabel('Date')
plt.ylabel('Total Sales')
plt.show()


# In[74]:


#removing the negative values
data = data[data['UnitPrice'] > 0]


# In[75]:


# resetting the index of the cleaned dataset
data.reset_index(drop=True, inplace=True)


# In[76]:


data.shape[0]


# In[77]:


#Outliers detection using z-score
#import necessary library
from scipy import stats

#select numeric columns
```

```python
numeric_columns = data.select_dtypes(include=[np.number]).columns

#calculate Z-scores for each numeric column
z_scores = np.abs(stats.zscore(data[numeric_columns]))

#set a threshold for Z-score (e.g., 3 standard deviations)
threshold = 3

#identify potential outliers
outliers = (z_scores > threshold).any(axis=1)

#print the indices of potential outliers
print("Potential outlier indices:")
print(data.index[outliers])


# In[78]:


#visualize potential outliers using box plots for each numeric column
plt.figure(figsize=(12, 6))
for i, col in enumerate(numeric_columns, 1):
    plt.subplot(1, len(numeric_columns), i)
    plt.boxplot(data[col])
    plt.title(f'Boxplot of {col}')

plt.tight_layout()
plt.show()


# In[79]:


#remove outliers from the DataFrame
data_without_outliers = data[~outliers]

#display the number of outliers and the shape of the cleaned DataFrame
print(f"Number of outliers removed: {sum(outliers)}")
print(f"Shape of the cleaned DataFrame: {data_without_outliers.shape}")
```

```
# In[80]:


#visualizing the data without outliers

#select three numeric columns for the box plot
numeric_columns = ["Quantity", "UnitPrice"]  # Replace with your column
names

#create a box plot without outliers
plt.figure(figsize=(10, 6))
data_without_outliers[numeric_columns].boxplot(showfliers=False)
plt.title('Box Plot without Outliers')
plt.xlabel('Numeric Columns')
plt.ylabel('Values')

plt.show()




# In[81]:


#plot histograms
data.hist(figsize=(12, 10))
plt.show()


# In[82]:


#identufying the numeric columns
numeric_columns = data.select_dtypes(include=np.number).columns


# In[83]:
```

```
#create a correlation matrix for numeric columns
matrix = data[numeric_columns].corr()

#heatmap of the correlation matrix
#import seaborn for heatmap
import seaborn as sns
plt.figure(figsize=(10, 8))
sns.heatmap(matrix, annot=True, cmap="mako", linewidths=.5)
plt.title('Correlation Matrix')
plt.show()


# # Feature engineering

# In[84]:


#1.total spending of the customers
data.loc[:, 'TotalSpending'] = data['Quantity'] * data['UnitPrice']




# In[85]:


#2.frequency p product
data.loc[:, 'Frequency'] =
data.groupby('CustomerID')['InvoiceNo'].transform('nunique')


# In[86]:


#3.recent purchase
data.loc[:, 'LastPurchaseDate'] =
data.groupby('CustomerID')['InvoiceDate'].transform('max')
data.loc[:, 'Recency'] = (pd.to_datetime('now') -
data['LastPurchaseDate']).dt.days
```

```
# In[87]:


#4.average unitprice
data.loc[:, 'UnitPrice'] =
data.groupby('CustomerID')['UnitPrice'].transform('mean')


# # Customer segmentation using K-means

# In[88]:


#relevant feature
X = data[['TotalSpending', 'Frequency', 'Recency']]
X.isnull()


# In[89]:


from sklearn.impute import SimpleImputer
from sklearn.preprocessing import StandardScaler
scaler=StandardScaler()

# Identify numeric and non-numeric columns
numeric_columns = data.select_dtypes(include=['number']).columns
non_numeric_columns = data.select_dtypes(exclude=['number']).columns

# Impute missing values with the mean for numeric columns
numeric_imputer = SimpleImputer(strategy='mean')
data.loc[:, numeric_columns] = numeric_imputer.fit_transform(data.loc[:,
numeric_columns])

# Impute missing values with a constant for non-numeric columns
non_numeric_imputer = SimpleImputer(strategy='constant',
fill_value='Unknown')
data.loc[:, non_numeric_columns] =
non_numeric_imputer.fit_transform(data.loc[:, non_numeric_columns])
```

```
# Standardize the features after handling missing values
X_scaled = scaler.fit_transform(data[['TotalSpending', 'Frequency',
'Recency']])




# In[90]:


#determine the optimal K based on the elbow
from sklearn.cluster import KMeans

#choose the optimal K based on the Elbow Method (e.g., K=3)
optimal_k = 3

#fit KMeans model
kmeans = KMeans(n_clusters=optimal_k, random_state=0)
data['Cluster'] = kmeans.fit_predict(X_scaled)


# In[92]:


#plot the Elbow Method
plt.plot(k_values, inertia_values, marker='o')
plt.title('Elbow Method for Optimal K')
plt.xlabel('Number of Clusters (K)')
plt.ylabel('Inertia (Within-Cluster Sum of Squares)')
plt.show()


# # Evaluate the clustering model using appropriate metrics.

# In[94]:


# Randomly sample 10% of the data
sample_data = data.sample(frac=0.1, random_state=0)
```

```
# Assuming 'X_scaled' contains your scaled feature matrix
X_scaled_sample = scaler.transform(sample_data[['TotalSpending',
'Frequency', 'Recency']])

# Assuming 'sample_data' contains the cluster labels
silhouette_avg = silhouette_score(X_scaled_sample, sample_data['Cluster'])
print(f"Silhouette Score: {silhouette_avg}")



# In[ ]:
```

5.2 Technical Implementation

5.2.1. Data Preprocessing

1.1 Loading the Dataset

The dataset was loaded using the Pandas library to facilitate further analysis.

1.2 Handling Missing Values

Missing values in the dataset were addressed by filling the mode value for the "Description" column.

1.3 Removing Duplicate Entries

Duplicate entries were identified and removed from the dataset to ensure data integrity.

5.2.2. Exploratory Data Analysis (EDA)

2.1 Descriptive Statistics

Basic statistical information, such as mean and standard deviation, was computed for numeric features to gain insights into the dataset.

2.2 Visualizing Top Descriptions

A bar plot was created to visualize the most frequent product descriptions, providing a snapshot of popular items.

2.3 Monthly Sales Dashboard

An area plot was generated to showcase the monthly sales trend, offering a comprehensive view of sales over time.

5.2.3. Outlier Detection

3.1 Z-Score Outlier Detection

Outliers were detected using Z-scores, helping to identify and handle extreme values.

3.2 Visualizing Outliers

Box plots were utilized to visualize potential outliers in numeric columns, aiding in the identification and interpretation of extreme values.

3.3 Removing Outliers

Outliers were removed from the dataset to ensure a more accurate and reliable analysis.

5.2.4. Feature Engineering

4.1 Total Spending Calculation

A new feature, "TotalSpending," was created by multiplying the quantity of items by their unit price.

4.2 Frequency per Product

The frequency of product purchases for each customer was calculated, providing insights into customer behavior.

4.3 Recency of Purchase

The recency of each customer's last purchase was determined, offering valuable information on customer engagement.

4.4 Average Unit Price

The average unit price for each customer was computed to capture the overall pricing trend.

5.2.5. Customer Segmentation using K-Means

5.1 Feature Scaling

To ensure fair representation of features, data was standardized using the StandardScaler.

5.2 Determining Optimal K (Elbow Method)

The optimal number of clusters (K) was determined using the Elbow Method, leading to the selection of a suitable K value.

5.3 K-Means Clustering

K-Means clustering was implemented to segment customers into distinct groups based on their spending behavior.

5.2.6. Model Evaluation

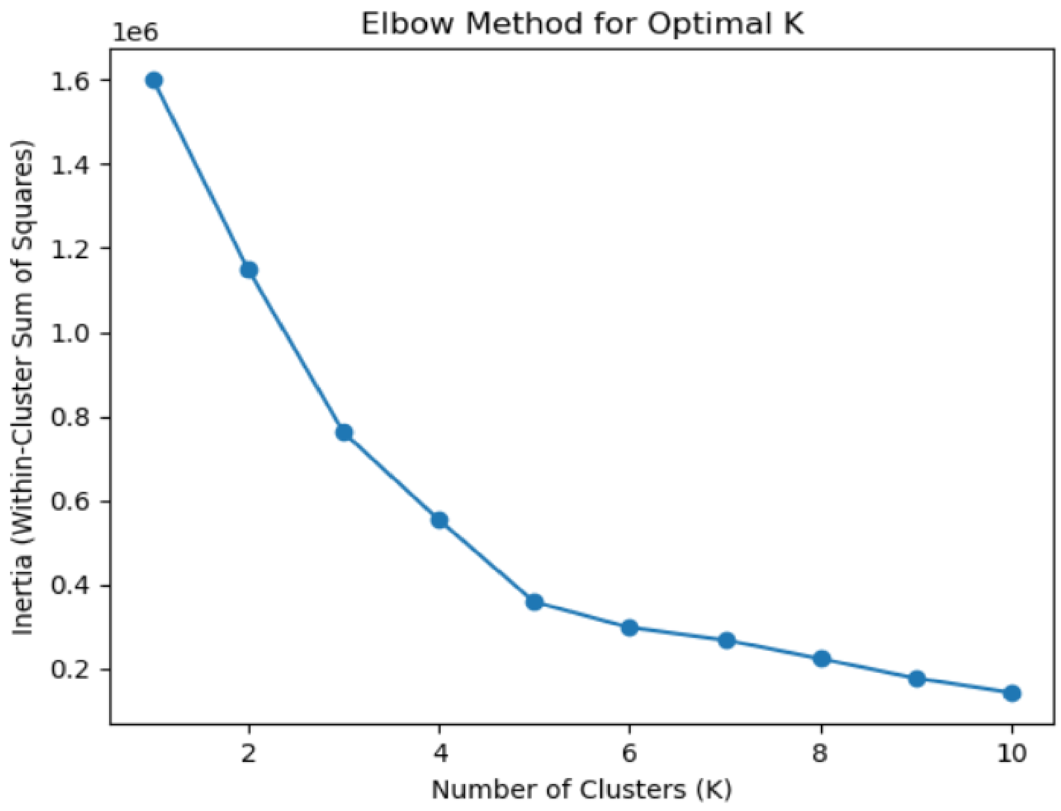
6.1 Sampling Data

A 10% random sample of the data was selected for model evaluation.

6.2 Silhouette Score

The Silhouette Score was calculated to assess the effectiveness of the clustering model.

6.Output



```
Silhouette Score: 0.7744560678689537
```

7. Conclusion

In conclusion, the customer segmentation and analysis project has provided valuable insights into the dataset's customer behavior and purchasing patterns. Through rigorous data preprocessing, exploratory data analysis (EDA), and advanced feature engineering, we have laid a solid foundation for understanding customer dynamics.

The removal of duplicate entries, handling of missing values, and careful treatment of outliers ensured the integrity and quality of the dataset. Exploratory data analysis shed light on popular product descriptions, allowing businesses to identify and prioritize items that resonate with customers.

The monthly sales dashboard provided a dynamic visualization of sales trends over time, aiding in strategic decision-making. Outlier detection and subsequent removal enhanced the robustness of the dataset, eliminating extreme values that could skew our analysis.

Feature engineering played a pivotal role in creating meaningful metrics such as total spending, purchase frequency, recency, and average unit price. These features provided a nuanced understanding of customer engagement, spending habits, and preferences.

The implementation of K-Means clustering for customer segmentation allowed for the creation of distinct customer groups based on spending behavior. The determination of the optimal number of clusters using the Elbow Method ensured the reliability of the clustering model.

In the evaluation phase, a random sample of the data was used to calculate the Silhouette Score, a metric indicating the cohesion and separation of clusters. This score serves as a valuable measure of the clustering model's effectiveness.

In summary, this project equips businesses with actionable insights for targeted marketing strategies, product recommendations, and customer engagement initiatives. The customer segmentation analysis provides a foundation for personalized approaches, enabling businesses to enhance customer satisfaction and drive overall success in a competitive market.