Task 3: Customer Segmentation Using Clustering

1. Introduction

Customer segmentation is a critical process in marketing and business strategy that involves dividing a customer base into distinct groups based on shared characteristics. This segmentation enables businesses to tailor their marketing efforts, improve customer satisfaction, and optimize resource allocation. In this project, we perform customer segmentation using clustering techniques, leveraging both customer profile data (`Customers.csv`) and transaction data (`Transactions.csv`).

This document provides an in-depth explanation of the approach, process, and algorithms used in the project. It is designed to be research-level, informative, and insightful, ensuring that readers (including interviewers) can understand the logical depth and rationale behind each step. The **full code** is also included for reproducibility and clarity.

2. Objectives

1. **Customer Segmentation**: Group customers into distinct clusters based on their behavior and characteristics.
2. **Feature Engineering**: Derive meaningful features from raw data to enhance clustering performance.
3. **Clustering Evaluation**: Use metrics like the Davies-Bouldin Index to evaluate the quality of clusters.
4. **Visualization**: Visualize clusters to gain insights into customer behavior.
5. **Actionable Insights**: Provide actionable insights for marketing and business strategies.

3. Dataset Description

3.1. Customers.csv

- **CustomerID**: Unique identifier for each customer.
- **CustomerName**: Name of the customer.
- **Region**: Geographic region of the customer.
- **SignupDate**: Date when the customer signed up.

3.2. Transactions.csv

- **TransactionID**: Unique identifier for each transaction.
- **CustomerID**: Identifier linking the transaction to a customer.
- **ProductID**: Identifier for the product purchased.
- **TransactionDate**: Date of the transaction.
- **Quantity**: Number of units purchased.
- **TotalValue**: Total monetary value of the transaction.
- **Price**: Price per unit of the product.

---

4. Methodology

4.1. Data Preprocessing

Before performing clustering, the data must be cleaned and prepared. This involves:

1. **Merging Datasets**: Combining `Customers.csv` and `Transactions.csv` on `CustomerID` to create a unified dataset.
2. **Handling Missing Values**: Removing or imputing missing values to ensure data quality.
3. **Feature Engineering**: Creating new features that capture customer behavior, such as:
    - **TotalSpending**: Total amount spent by each customer.
    - **NumTransactions**: Number of transactions made by each customer.

4.2. Feature Selection

The following features are used for clustering:

- **TotalSpending**: Reflects the monetary value of a customer.

- **NumTransactions**: Reflects the frequency of customer engagement.

These features are chosen because they capture both the **monetary value** and **engagement level** of customers, which are key dimensions in customer segmentation.

4.3. Data Standardization

Clustering algorithms like K-Means are sensitive to the scale of the data. Therefore, the features are standardized using `StandardScaler` to ensure that all features contribute equally to the clustering process.

4.4. Clustering Algorithm: K-Means

K-Means is a centroid-based clustering algorithm that partitions the data into `k` clusters, where each cluster is represented by its centroid. The algorithm works as follows:

1. **Initialization**: Randomly select `k` initial centroids.
2. **Assignment**: Assign each data point to the nearest centroid.
3. **Update**: Recalculate the centroids as the mean of all points in the cluster.
4. **Iteration**: Repeat the assignment and update steps until convergence.

Why K-Means?

- **Simplicity**: Easy to implement and interpret.
- **Scalability**: Efficient for large datasets.
- **Effectiveness**: Works well for datasets with clear separation between clusters.

4.5. Determining the Optimal Number of Clusters

Choosing the right number of clusters (`k`) is critical for meaningful segmentation. Two methods are used:

1. **Elbow Method**: Plots the within-cluster sum of squares (inertia) against the number of clusters. The "elbow" point indicates the optimal number of clusters.
2. **Davies-Bouldin Index**: A metric that evaluates the quality of clustering by measuring the ratio of within-cluster scatter to between-cluster separation. Lower values indicate better clustering.

4.6. Cluster Evaluation

The quality of the clusters is evaluated using the **Davies-Bouldin Index**. This metric is chosen because:

- It is computationally efficient.
- It provides a single score that balances within-cluster compactness and between-cluster separation.

4.7. Visualization

To visualize the clusters, **Principal Component Analysis (PCA)** is used to reduce the dimensionality of the data to 2D. This allows for easy plotting of the clusters on a scatter plot.

4.8. Cluster Analysis

The mean values of the features (`TotalSpending` and `NumTransactions`) are calculated for each cluster to understand the characteristics of the customers in each group.

---

5. Implementation Steps

5.1. Load Data

- Load `Customers.csv` and `Transactions.csv` into Pandas dataframes.

```
import pandas as pd

# Load data
customers_df = pd.read_csv('Customers.csv')
transactions_df = pd.read_csv('Transactions.csv')

# Display the first few rows
print("Customers Data:")
print(customers_df.head())

print("\\nTransactions Data:")
print(transactions_df.head())
```

5.2. Merge Data

- Merge the two datasets on `CustomerID` to create a unified dataset.

```
# Merge data
merged_df = pd.merge(customers_df, transactions_df, on='Cus
tomerID')

# Display the merged data
print("Merged Data:")
print(merged_df.head())
```

5.3. Feature Engineering

- Create `TotalSpending` by summing `TotalValue` for each customer.
- Create `NumTransactions` by counting the number of transactions for each customer.

```
# Total Spending per Customer
total_spending = merged_df.groupby('CustomerID')['TotalValu
e'].sum().reset_index()
total_spending.rename(columns={'TotalValue': 'TotalSpendin
g'}, inplace=True)

# Number of Transactions per Customer
num_transactions = merged_df.groupby('CustomerID')['Transac
tionID'].count().reset_index()
num_transactions.rename(columns={'TransactionID': 'NumTrans
actions'}, inplace=True)

# Merge features with customers dataframe
customers_df = pd.merge(customers_df, total_spending, on='C
ustomerID')
customers_df = pd.merge(customers_df, num_transactions, on
='CustomerID')

# Display the updated customers dataframe
```

```
print("Updated Customers Data:")
print(customers_df.head())
```

5.4. Data Standardization

- Standardize `TotalSpending` and `NumTransactions` using `StandardScaler`.

```
from sklearn.preprocessing import StandardScaler

# Select features for clustering
features = ['TotalSpending', 'NumTransactions']
X = customers_df[features]

# Standardize the features
scaler = StandardScaler()
X_scaled = scaler.fit_transform(X)

# Display the scaled data
print("Scaled Data:")
print(X_scaled[:5])
```

5.5. Determine Optimal Number of Clusters

- Use the Elbow Method and Davies-Bouldin Index to determine the optimal number of clusters.

```
from sklearn.cluster import KMeans
from sklearn.metrics import davies_bouldin_score
import matplotlib.pyplot as plt

# Determine optimal number of clusters
inertia = []
db_scores = []
K = range(2, 11)

for k in K:
    kmeans = KMeans(n_clusters=k, random_state=42)
    kmeans.fit(X_scaled)
```

```
    inertia.append(kmeans.inertia_)
    db_scores.append(davies_bouldin_score(X_scaled, kmeans.
labels_))

# Plot the Elbow Method graph
plt.figure(figsize=(10, 5))
plt.plot(K, inertia, 'bo-', label='Inertia')
plt.xlabel('Number of Clusters')
plt.ylabel('Inertia')
plt.title('Elbow Method For Optimal k')
plt.legend()
plt.show()

# Plot the Davies-Bouldin Index graph
plt.figure(figsize=(10, 5))
plt.plot(K, db_scores, 'go-', label='Davies-Bouldin Index')
plt.xlabel('Number of Clusters')
plt.ylabel('Davies-Bouldin Index')
plt.title('Davies-Bouldin Index For Optimal k')
plt.legend()
plt.show()
```

5.6. Perform Clustering

- Apply K-Means clustering with the optimal number of clusters.

```
# Perform clustering with optimal k
optimal_k = 4  # Example: Choose based on the Elbow Method
and DB Index
kmeans = KMeans(n_clusters=optimal_k, random_state=42)
labels = kmeans.fit_predict(X_scaled)

# Add cluster labels to the customers dataframe
customers_df['Cluster'] = labels

# Display the clustered dataframe
print("Clustered Customers Data:")
print(customers_df.head())
```

5.7. Evaluate Clusters

- Calculate the Davies-Bouldin Index to evaluate the quality of the clusters.

```
# Evaluate clusters
db_index = davies_bouldin_score(X_scaled, labels)
print(f"Davies-Bouldin Index: {db_index}")
```

5.8. Visualize Clusters

- Use PCA to reduce the data to 2D and plot the clusters.

```
from sklearn.decomposition import PCA
import seaborn as sns

# Reduce dimensionality using PCA
pca = PCA(n_components=2)
X_pca = pca.fit_transform(X_scaled)

# Add PCA components to the dataframe
customers_df['PCA1'] = X_pca[:, 0]
customers_df['PCA2'] = X_pca[:, 1]

# Plot the clusters
plt.figure(figsize=(10, 7))
sns.scatterplot(x='PCA1', y='PCA2', hue='Cluster', data=cus
tomers_df, palette='viridis', s=100)
plt.title('Customer Segments')
plt.xlabel('PCA Component 1')
plt.ylabel('PCA Component 2')
plt.legend()
plt.show()
```

5.9. Analyze Clusters

- Calculate the mean values of `TotalSpending` and `NumTransactions` for each cluster.

```
# Analyze clusters
cluster_analysis = customers_df.groupby('Cluster')[features].mean()
print("Cluster Analysis:")
print(cluster_analysis)
```

---

6. Results and Insights

6.1. Cluster Characteristics

- **High-Value Customers**: High `TotalSpending` and high `NumTransactions`.
- **Frequent Customers**: Low `TotalSpending` but high `NumTransactions`.
- **Occasional Customers**: Low `TotalSpending` and low `NumTransactions`.
- **Big Spenders**: High `TotalSpending` but low `NumTransactions`.

6.2. Business Implications

- **Targeted Marketing**: Tailor marketing campaigns based on cluster characteristics.
- **Customer Retention**: Focus on retaining high-value and frequent customers.
- **Upselling Opportunities**: Identify opportunities to upsell to occasional customers.

---

7. Conclusion

This project demonstrates the power of clustering techniques for customer segmentation. By leveraging both profile and transaction data, we were able to group customers into meaningful clusters and derive actionable insights. The use of the Davies-Bouldin Index and visualization techniques ensured that the clusters were both high-quality and interpretable.

---

8. Future Work

1. **Advanced Clustering Algorithms**: Explore other clustering algorithms like DBSCAN or Gaussian Mixture Models.

2. **Additional Features**: Incorporate more features, such as product categories or customer demographics.
3. **Real-Time Segmentation**: Implement real-time clustering for dynamic customer segmentation.
