Idea: Customer Segmentation Analysis

Project Description

The aim of this data analytics project is to perform customer segmentation analysis for an e- commerce company. By analyzing customer behavior and purchase patterns, the goal is to group customers into distinct segments. This segmentation can inform targeted marketing strategies, improve customer satisfaction, and enhance overall business strategies.

```
#Importing necessary libraries
import numpy as np
import pandas as pd
import warnings
warnings.filterwarnings("ignore")
import seaborn as sns
import matplotlib.pyplot as plt
from scipy.stats import pointbiserialr
from google.colab import drive
drive.mount('/content/drive')
```

```
Mounted at /content/drive
```

```
from sklearn.preprocessing import StandardScaler
from sklearn.cluster import KMeans
```

```
# Access the CSV file with the  path STEP1 LOAD DATA
path="/content/drive/MyDrive/OASIS/ifood_df.csv"
df = pd.read_csv(path, na_values=["NA", "NaN", "", "?","Not Available"])
```

Data Cleaning and exploration

```
#taking look at Data
df.head()
```

	Income	Kidhome	Teenhome	Recency	MntWines	MntFruits	MntMeatProducts	MntFishProc
0	58138.0	0	0	58	635	88	546	
1	46344.0	1	1	38	11	1	6	
2	71613.0	0	0	26	426	49	127	
3	26646.0	1	0	26	11	4	20	
4	58293.0	1	0	94	173	43	118	

5 rows × 39 columns

```
df.columns
```

```
Index(['Income', 'Kidhome', 'Teenhome', 'Recency', 'MntWines', 'MntFruits',
       'MntMeatProducts', 'MntFishProducts', 'MntSweetProducts',
       'MntGoldProds', 'NumDealsPurchases', 'NumWebPurchases',
       'NumCatalogPurchases', 'NumStorePurchases', 'NumWebVisitsMonth',
       'AcceptedCmp3', 'AcceptedCmp4', 'AcceptedCmp5', 'AcceptedCmp1',
       'AcceptedCmp2', 'Complain', 'Z_CostContact', 'Z_Revenue', 'Response',
       'Age', 'Customer_Days', 'marital_Divorced', 'marital_Married',
       'marital_Single', 'marital_Together', 'marital_Widow',
       'education_2n Cycle', 'education_Basic', 'education_Graduation',
       'education_Master', 'education_PhD', 'MntTotal', 'MntRegularProds',
       'AcceptedCmpOverall'],
      dtype='object')
```

```
# Handle missing values (e.g., impute or drop rows/columns based on analysis)
(df.isnull().sum())
```

```
Income                 0
Kidhome                0
```

```
Teenhome                0
Recency                 0
MntWines                0
MntFruits               0
MntMeatProducts         0
MntFishProducts         0
MntSweetProducts        0
MntGoldProds            0
NumDealsPurchases       0
NumWebPurchases         0
NumCatalogPurchases     0
NumStorePurchases       0
NumWebVisitsMonth       0
AcceptedCmp3            0
AcceptedCmp4            0
AcceptedCmp5            0
AcceptedCmp1            0
AcceptedCmp2            0
Complain                0
Z_CostContact           0
Z_Revenue               0
Response                0
Age                     0
Customer_Days           0
marital_Divorced        0
marital_Married         0
marital_Single          0
marital_Together        0
marital_Widow           0
education_2n Cycle      0
education_Basic         0
education_Graduation    0
education_Master        0
education_PhD           0
MntTotal                0
MntRegularProds         0
AcceptedCmpOverall      0
dtype: int64
```

```
#Checking column types
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 2205 entries, 0 to 2204
Data columns (total 39 columns):
 #   Column                Non-Null Count  Dtype  
---  ------                --------------  -----  
 0   Income                2205 non-null   float64
 1   Kidhome               2205 non-null   int64  
 2   Teenhome              2205 non-null   int64  
 3   Recency               2205 non-null   int64  
 4   MntWines              2205 non-null   int64  
 5   MntFruits             2205 non-null   int64  
 6   MntMeatProducts       2205 non-null   int64  
 7   MntFishProducts       2205 non-null   int64  
 8   MntSweetProducts      2205 non-null   int64  
 9   MntGoldProds          2205 non-null   int64  
 10  NumDealsPurchases     2205 non-null   int64  
 11  NumWebPurchases       2205 non-null   int64  
 12  NumCatalogPurchases   2205 non-null   int64  
 13  NumStorePurchases     2205 non-null   int64  
 14  NumWebVisitsMonth     2205 non-null   int64  
 15  AcceptedCmp3          2205 non-null   int64  
 16  AcceptedCmp4          2205 non-null   int64  
 17  AcceptedCmp5          2205 non-null   int64  
 18  AcceptedCmp1          2205 non-null   int64  
 19  AcceptedCmp2          2205 non-null   int64  
 20  Complain              2205 non-null   int64  
 21  Z_CostContact         2205 non-null   int64  
 22  Z_Revenue             2205 non-null   int64  
 23  Response              2205 non-null   int64  
 24  Age                   2205 non-null   int64  
 25  Customer_Days         2205 non-null   int64  
 26  marital_Divorced      2205 non-null   int64  
 27  marital_Married       2205 non-null   int64  
 28  marital_Single        2205 non-null   int64  
 29  marital_Together      2205 non-null   int64  
 30  marital_Widow         2205 non-null   int64  
 31  education_2n Cycle    2205 non-null   int64  
 32  education_Basic       2205 non-null   int64  
 33  education_Graduation  2205 non-null   int64  
 34  education_Master      2205 non-null   int64  
 35  education_PhD         2205 non-null   int64  
```

```
 36  MntTotal              2205 non-null   int64
 37  MntRegularProds       2205 non-null   int64
 38  AcceptedCmpOverall    2205 non-null   int64
dtypes: float64(1), int64(38)
memory usage: 672.0 KB
```

```
df.nunique()
```

```
Income                 1963
Kidhome                   3
Teenhome                  3
Recency                 100
MntWines                775
MntFruits               158
MntMeatProducts         551
MntFishProducts         182
MntSweetProducts        176
MntGoldProds            212
NumDealsPurchases        15
NumWebPurchases          15
NumCatalogPurchases      13
NumStorePurchases        14
NumWebVisitsMonth        16
AcceptedCmp3              2
AcceptedCmp4              2
AcceptedCmp5              2
AcceptedCmp1              2
AcceptedCmp2              2
Complain                  2
Z_CostContact             1
Z_Revenue                 1
Response                  2
Age                      56
Customer_Days           662
marital_Divorced          2
marital_Married           2
marital_Single            2
marital_Together          2
marital_Widow             2
education_2n Cycle        2
education_Basic           2
education_Graduation      2
education_Master          2
education_PhD             2
MntTotal                897
MntRegularProds         974
AcceptedCmpOverall        5
dtype: int64
```

```
df.drop(['Z_CostContact', 'Z_Revenue'], axis=1, inplace=True)
```

```
# Drop rows with missing values
df.dropna(inplace=True)
```

```
print("Duplicates:", df.duplicated().sum())
```

```
Duplicates: 184
```

```
# Remove duplicates if any
df.drop_duplicates(inplace=True)
```

```
# Display data types of each column
print(df.dtypes)
```

```
Income                 float64
Kidhome                  int64
Teenhome                 int64
Recency                  int64
MntWines                 int64
MntFruits                int64
MntMeatProducts          int64
MntFishProducts          int64
MntSweetProducts         int64
MntGoldProds             int64
NumDealsPurchases        int64
NumWebPurchases          int64
```

```
NumCatalogPurchases       int64
NumStorePurchases         int64
NumWebVisitsMonth         int64
AcceptedCmp3              int64
AcceptedCmp4              int64
AcceptedCmp5              int64
AcceptedCmp1              int64
AcceptedCmp2              int64
Complain                  int64
Response                  int64
Age                       int64
Customer_Days             int64
marital_Divorced          int64
marital_Married           int64
marital_Single            int64
marital_Together          int64
marital_Widow             int64
education_2n Cycle        int64
education_Basic           int64
education_Graduation      int64
education_Master          int64
education_PhD             int64
MntTotal                  int64
MntRegularProds           int64
AcceptedCmpOverall        int64
dtype: object
```

```
# Check data types of columns
non_numeric_columns = df.select_dtypes(exclude=[np.number]).columns.tolist()
print("Non-numeric columns:", non_numeric_columns)
```

```
Non-numeric columns: []
```

Descriptive Statistics

```
# Separate numerical
numerical_vars = df.select_dtypes(include=['int64', 'float64']).columns

# Generate descriptive statistics for numerical columns
numerical_stats = df[numerical_vars].describe()

# Display the results
print(numerical_stats)
```

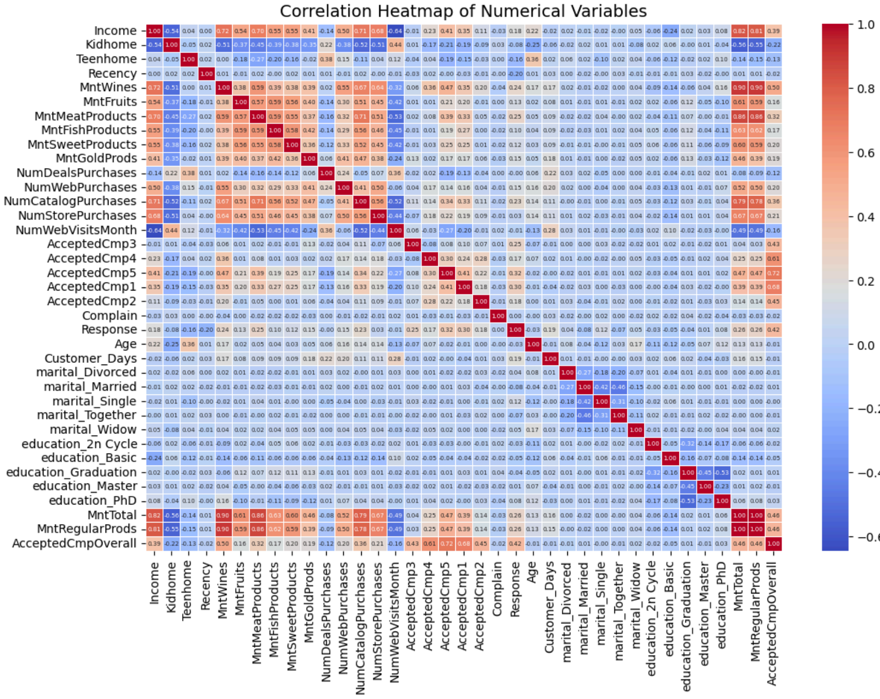
```
              Income      Kidhome     Teenhome      Recency     MntWines  \
count    2021.000000  2021.000000  2021.000000  2021.000000  2021.000000
mean    51687.258783     0.443345     0.509649    48.880752   306.492331
std     20713.046401     0.536196     0.546393    28.950917   337.603877
min      1730.000000     0.000000     0.000000     0.000000     0.000000
25%     35416.000000     0.000000     0.000000    24.000000    24.000000
50%     51412.000000     0.000000     0.000000    49.000000   178.000000
75%     68274.000000     1.000000     1.000000    74.000000   507.000000
max    113734.000000     2.000000     2.000000    99.000000  1493.000000

         MntFruits  MntMeatProducts  MntFishProducts  MntSweetProducts  \
count  2021.000000      2021.000000      2021.000000       2021.000000
mean     26.364671       166.059871        37.603662         27.268679
std      39.776518       219.869126        54.892196         41.575454
min       0.000000         0.000000         0.000000          0.000000
25%       2.000000        16.000000         3.000000          1.000000
50%       8.000000        68.000000        12.000000          8.000000
75%      33.000000       230.000000        50.000000         34.000000
max     199.000000      1725.000000       259.000000        262.000000

       MntGoldProds  ...  marital_Together  marital_Widow  education_2n Cycle  \
count   2021.000000  ...       2021.000000    2021.000000         2021.000000
mean      43.921821  ...          0.251856       0.034636            0.090549
std       51.678211  ...          0.434186       0.182902            0.287038
min        0.000000  ...          0.000000       0.000000            0.000000
25%        9.000000  ...          0.000000       0.000000            0.000000
50%       25.000000  ...          0.000000       0.000000            0.000000
75%       56.000000  ...          1.000000       0.000000            0.000000
max      321.000000  ...          1.000000       1.000000            1.000000

       education_Basic  education_Graduation  education_Master  education_PhD  \
count      2021.000000           2021.000000       2021.000000    2021.000000
mean          0.024245              0.502227          0.165760       0.217219
std           0.153848              0.500119          0.371957       0.412455
```

```
min          0.000000              0.000000          0.000000       0.000000
25%          0.000000              0.000000          0.000000       0.000000
50%          0.000000              1.000000          0.000000       0.000000
75%          0.000000              1.000000          0.000000       0.000000
max          1.000000              1.000000          1.000000       1.000000

          MntTotal  MntRegularProds  AcceptedCmpOverall
count  2021.000000      2021.000000         2021.000000
mean    563.789213       519.867392            0.302326
std     576.775749       554.797857            0.680812
min       4.000000      -283.000000            0.000000
25%      55.000000        42.000000            0.000000
50%     343.000000       288.000000            0.000000
75%     964.000000       883.000000            0.000000
max    2491.000000      2458.000000            4.000000

[8 rows x 37 columns]
```

```
# Check if there are categorical columns
categorical_vars = df.select_dtypes(include=['category']).columns

if not categorical_vars.empty:
    # Generate descriptive statistics for categorical columns
    categorical_stats = df[categorical_vars].describe()
    print(categorical_stats)
else:
    print("No categorical columns found in the DataFrame.")
```

```
No categorical columns found in the DataFrame.
```

```
# Compute the correlation matrix
correlation_matrix = df[numerical_vars].corr()

# Create a heatmap
plt.figure(figsize=(12, 8))
sns.heatmap(correlation_matrix, annot=True, cmap='coolwarm', fmt=".2f", linewidths=0.5, annot_kws={"size":5 })
plt.title('Correlation Heatmap of Numerical Variables', fontsize=14)
plt.xticks(fontsize=10)
plt.yticks(fontsize=10)
plt.show()
```




```
df.describe()
```

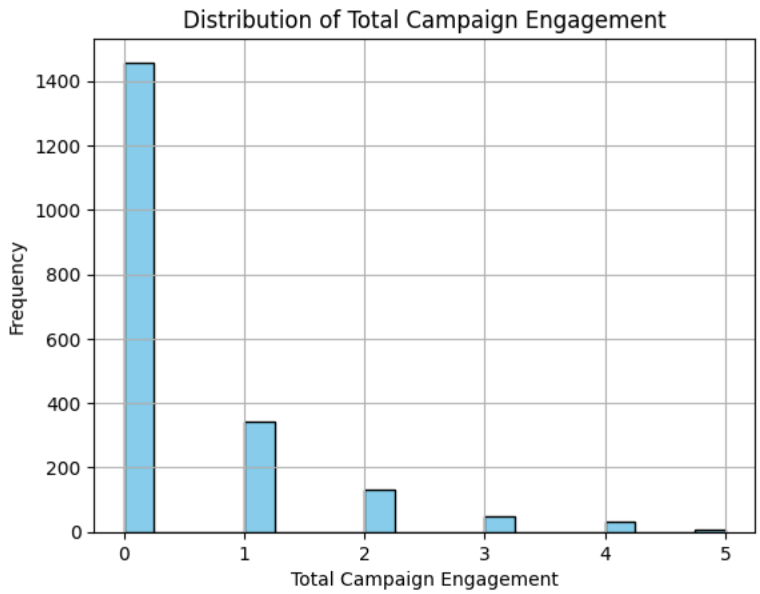
	Income	Kidhome	Teenhome	Recency	MntWines	MntFruits	MntI
count	2021.000000	2021.000000	2021.000000	2021.000000	2021.000000	2021.000000	
mean	51687.258783	0.443345	0.509649	48.880752	306.492331	26.364671	
std	20713.046401	0.536196	0.546393	28.950917	337.603877	39.776518	
min	1730.000000	0.000000	0.000000	0.000000	0.000000	0.000000	
25%	35416.000000	0.000000	0.000000	24.000000	24.000000	2.000000	
50%	51412.000000	0.000000	0.000000	49.000000	178.000000	8.000000	
75%	68274.000000	1.000000	1.000000	74.000000	507.000000	33.000000	
max	113734.000000	2.000000	2.000000	99.000000	1493.000000	199.000000	

8 rows × 37 columns

New feature

```
# Create a new feature representing total campaign engagement
df['TotalCampaignEngagement'] = df[['AcceptedCmp1', 'AcceptedCmp2', 'AcceptedCmp3', 'AcceptedCmp4', 'AcceptedCmp5', 'Response']].sum(axis=1)
```

```
# Plot histogram of TotalCampaignEngagement
plt.hist(df['TotalCampaignEngagement'], bins=20, color='skyblue', edgecolor='black')
plt.xlabel('Total Campaign Engagement')
plt.ylabel('Frequency')
plt.title('Distribution of Total Campaign Engagement')
plt.grid(True)
plt.show()
```



K-Means Clustring

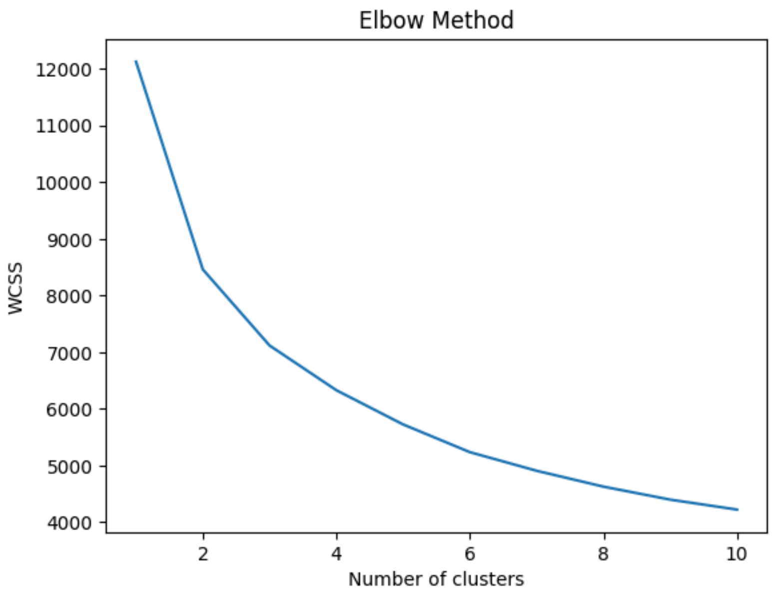
1. Standardising data
2. Principal Component Analysis (PCA)
3. Elbow method
4. Silhouette score analysis

```
# Select only a few columns for K-means clustering
cols_for_clustering = ['Income', 'Recency', 'MntWines', 'NumWebPurchases', 'Age', 'TotalCampaignEngagement']

# Create a new DataFrame with selected features
X = df[cols_for_clustering]
```

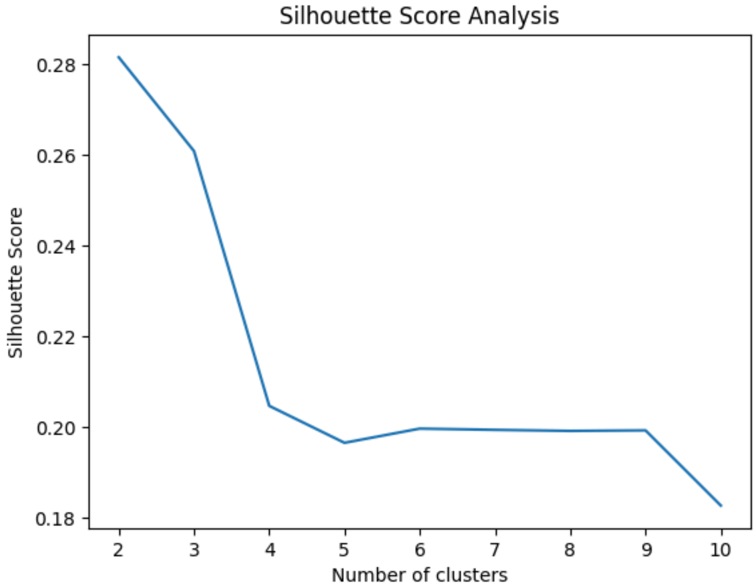
```
from sklearn.preprocessing import StandardScaler

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

```
from sklearn.decomposition import PCA

# Apply PCA
pca = PCA(n_components=2)  # Specify the number of components to keep
X_pca = pca.fit_transform(X_scaled)
```

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

# Determine optimal number of clusters using elbow method
wcss = []
for i in range(1, 11):
    kmeans = KMeans(n_clusters=i, init='k-means++', max_iter=300, n_init=10, random_state=0)
    kmeans.fit(X_scaled)
    wcss.append(kmeans.inertia_)

# Plotting the elbow method graph
plt.plot(range(1, 11), wcss)
plt.title('Elbow Method')
plt.xlabel('Number of clusters')
plt.ylabel('WCSS')
plt.show()
```



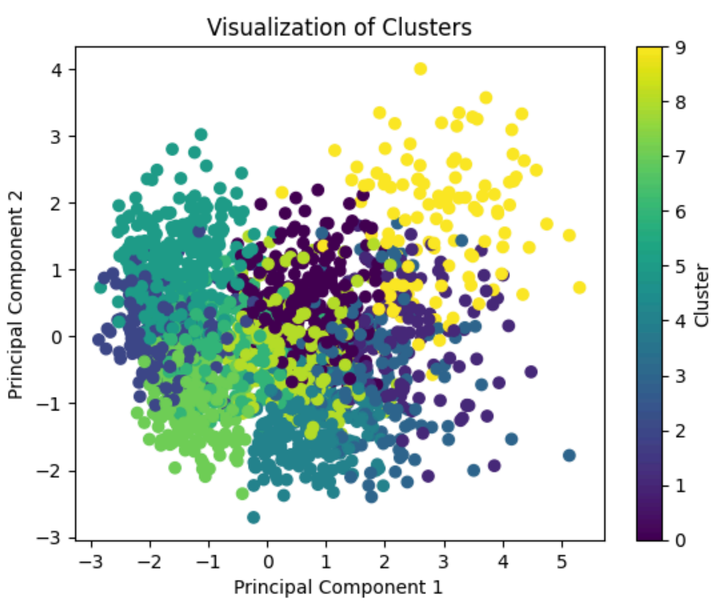

```
from sklearn.metrics import silhouette_score

# Calculate silhouette score for different number of clusters
silhouette_scores = []
for i in range(2, 11):
    kmeans = KMeans(n_clusters=i, init='k-means++', max_iter=300, n_init=10, random_state=0)
    cluster_labels = kmeans.fit_predict(X_scaled)
    silhouette_scores.append(silhouette_score(X_scaled, cluster_labels))

# Plotting silhouette scores
plt.plot(range(2, 11), silhouette_scores)
plt.title('Silhouette Score Analysis')
plt.xlabel('Number of clusters')
plt.ylabel('Silhouette Score')
plt.show()
```



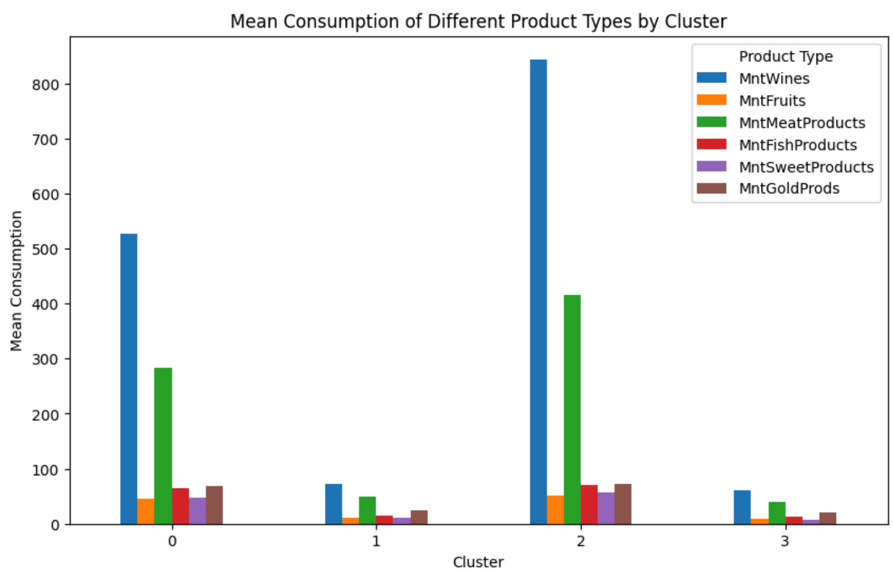

```
# Elbow Method
optimal_clusters_elbow = wcss.index(min(wcss)) + 1
print("Optimal number of clusters using elbow method:", optimal_clusters_elbow)

# Silhouette Score Analysis
optimal_clusters_silhouette = silhouette_scores.index(max(silhouette_scores)) + 2
print("Optimal number of clusters using silhouette score analysis:", optimal_clusters_silhouette)

# Select the highest number of clusters
highest_number_of_clusters = max(optimal_clusters_elbow, optimal_clusters_silhouette)
print("Highest number of clusters:", highest_number_of_clusters)
```

```
Optimal number of clusters using elbow method: 10
Optimal number of clusters using silhouette score analysis: 2
Highest number of clusters: 10
```

```
plt.scatter(X_pca[:, 0], X_pca[:, 1], c=kmeans.labels_, cmap='viridis')
plt.xlabel('Principal Component 1')
plt.ylabel('Principal Component 2')
plt.title('Visualization of Clusters')
plt.colorbar(label='Cluster')
plt.show()
```



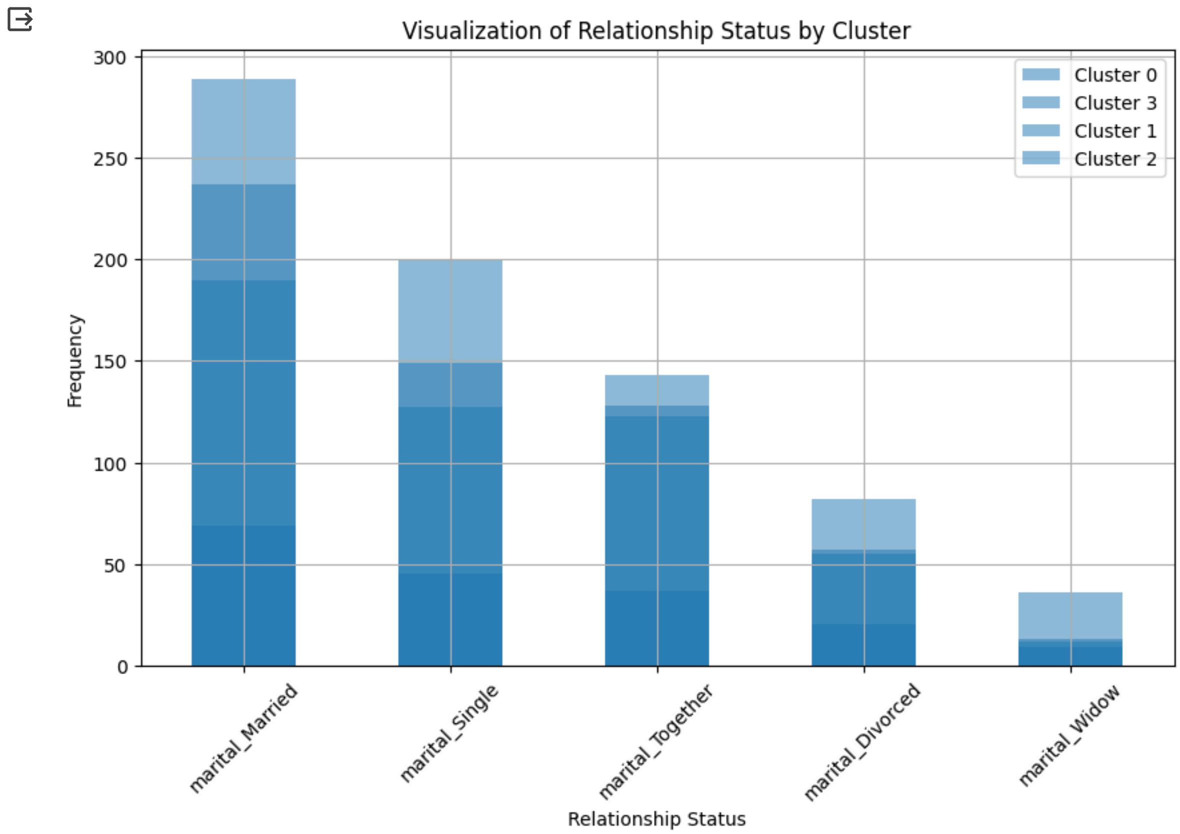
```
# Fit KMeans to the scaled data
kmeans = KMeans(n_clusters=4, init='k-means++', max_iter=300, n_init=10, random_state=0)
kmeans.fit(X_scaled)

# Add cluster labels to the DataFrame
df['Cluster'] = kmeans.labels_

# Group data by cluster and calculate mean consumption of different product types
mean_consumption_by_cluster = df.groupby('Cluster')[['MntWines', 'MntFruits', 'MntMeatProducts', 'MntFishProducts', 'MntSweetProducts', 'Mnt

# Plot mean consumption of different product types by cluster
mean_consumption_by_cluster.plot(kind='bar', figsize=(10, 6))
plt.title('Mean Consumption of Different Product Types by Cluster')
plt.xlabel('Cluster')
plt.ylabel('Mean Consumption')
plt.xticks(rotation=0)
plt.legend(title='Product Type')
plt.show()
```




```
# Visualize cluster sizes
cluster_sizes = df['Cluster'].value_counts()
cluster_sizes.plot(kind='bar', color='skyblue', edgecolor='black')
plt.title('Visualization of Cluster Sizes')
plt.xlabel('Cluster')
plt.ylabel('Number of Customers')
plt.xticks(rotation=0)
plt.show()
```


```
# Visualize income by cluster
plt.figure(figsize=(10, 6))
for cluster_label in df['Cluster'].unique():
    cluster_data = df[df['Cluster'] == cluster_label]
    plt.hist(cluster_data['Income'], bins=20, alpha=0.5, label=f'Cluster {cluster_label}')
plt.title('Visualization of Income by Cluster')
plt.xlabel('Income')
plt.ylabel('Frequency')
plt.legend()
plt.grid(True)
plt.show()
```

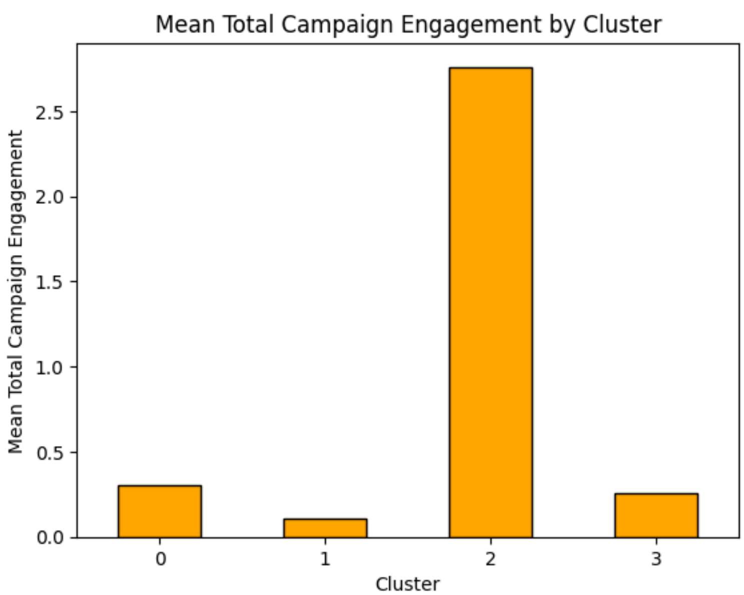
```
# Visualize relationship status by cluster
plt.figure(figsize=(10, 6))
for cluster_label in df['Cluster'].unique():
    cluster_data = df[df['Cluster'] == cluster_label]
    cluster_data['marital_status'] = cluster_data[['marital_Divorced', 'marital_Married', 'marital_Single', 'marital_Together', 'marital_Wic
    cluster_data['marital_status'].value_counts().plot(kind='bar', alpha=0.5, label=f'Cluster {cluster_label}')
plt.title('Visualization of Relationship Status by Cluster')
plt.xlabel('Relationship Status')
plt.ylabel('Frequency')
plt.legend()
plt.xticks(rotation=45)
plt.grid(True)
plt.show()
```



Visualizations to illustrate customer segments based on the clustering results:

```
plt.scatter(X_pca[:, 0], X_pca[:, 1], c=kmeans.labels_, cmap='viridis')
plt.xlabel('Principal Component 1')
plt.ylabel('Principal Component 2')
plt.title('Customer Segments (PCA)')
plt.colorbar(label='Cluster')
plt.show()
```


```
mean_campaign_engagement = df.groupby('Cluster')['TotalCampaignEngagement'].mean()
mean_campaign_engagement.plot(kind='bar', color='orange', edgecolor='black')
plt.title('Mean Total Campaign Engagement by Cluster')
plt.xlabel('Cluster')
plt.ylabel('Mean Total Campaign Engagement')
plt.xticks(rotation=0)
plt.show()
```



Insights and Recommendations: Analyzed characteristics of each segment and insights.

```
# Analyze characteristics of each segment
segment_characteristics = df.groupby('Cluster').agg({
    'Income': 'mean',
    'Age': 'mean',
    'TotalCampaignEngagement': 'mean',
    'MntWines': 'mean',
    'MntFruits': 'mean',
    'MntMeatProducts': 'mean',
    'MntFishProducts': 'mean',
    'MntSweetProducts': 'mean',
    'MntGoldProds': 'mean',
    'NumWebPurchases': 'mean',
    'NumCatalogPurchases': 'mean',
    'NumStorePurchases': 'mean',
    'NumWebVisitsMonth': 'mean',
    'Response': 'mean'
})

# Provide insights
for cluster_label, characteristics in segment_characteristics.iterrows():
    print(f"Cluster {cluster_label} Insights:")
    print(f"Average Income: ${characteristics['Income']:.2f}")
    print(f"Average Age: {characteristics['Age']:.2f} years")
    print(f"Average Total Campaign Engagement: {characteristics['TotalCampaignEngagement']:.2f}")
    print(f"Average Wine Purchases: {characteristics['MntWines']:.2f}")
    print(f"Average Fruit Purchases: {characteristics['MntFruits']:.2f}")
    print(f"Average Meat Product Purchases: {characteristics['MntMeatProducts']:.2f}")
    print(f"Average Fish Product Purchases: {characteristics['MntFishProducts']:.2f}")
    print(f"Average Sweet Product Purchases: {characteristics['MntSweetProducts']:.2f}")
    print(f"Average Gold Product Purchases: {characteristics['MntGoldProds']:.2f}")
    print(f"Average Number of Web Purchases: {characteristics['NumWebPurchases']:.2f}")
    print(f"Average Number of Catalog Purchases: {characteristics['NumCatalogPurchases']:.2f}")
    print(f"Average Number of Store Purchases: {characteristics['NumStorePurchases']:.2f}")
    print(f"Average Number of Web Visits per Month: {characteristics['NumWebVisitsMonth']:.2f}")
    print(f"Average Response Rate: {characteristics['Response']:.2%}")
    print()
```