# Customer Segmentation Using K-means Clustering

## Objective

The objective of this task is to apply K-means clustering to segment customer data and analyze patterns without predefined labels. We will use the Mall Customer Segmentation dataset from Kaggle, which includes information like customer ID, gender, age, annual income, and spending score. These features will be used to segment customers based on shopping behavior.

## Dataset

- **Dataset URL:** https://www.kaggle.com/datasets/vjchoudhary7/customer-segmentation-tutorial-in-python
- **Features:**
    - CustomerID
    - Gender
    - Age
    - Annual Income (k$)
    - Spending Score (1-100)

## Activities

### 1. Data Exploration

### 1.1 Load the dataset and conduct basic exploratory data analysis to understand the features.

```
import pandas as pd


# Load the dataset

file_path = '/mnt/data/Mall_Customers.csv'

df = pd.read_csv(file_path)


# Display the first few rows

print("First few rows of the dataset:")
print(df.head())


# Basic statistics of numerical features

print("\nBasic statistics:")

print(df.describe())


# Check for missing values

print("\nMissing values:")

print(df.isnull().sum())
```

**Output:**

- The first few rows of the dataset provide an initial glimpse into the data structure.
- Basic statistics include count, mean, standard deviation, min, max, and percentiles for the numerical features.
- Checking for missing values ensures that there are no gaps in the data that could affect the analysis.

**2. Clustering Implementation**

**2.1 Select relevant features for clustering.**

We will use Age, Annual Income (k$), and Spending Score (1-100) as the features for clustering.

python

Copy code

```
# Select relevant features for clustering

features = df[['Age', 'Annual Income (k$)', 'Spending Score (1-100)']]
```

**2.2 Standardize the features.**

Standardizing the features ensures that each feature contributes equally to the distance calculations used in the K-means algorithm.

python

Copy code

```
from sklearn.preprocessing import StandardScaler


# Standardize the features
scaler = StandardScaler()
scaled_features = scaler.fit_transform(features)
```

**2.3 Use the elbow method to determine the optimal number of clusters.**

The elbow method helps to find the optimal number of clusters by plotting the within-cluster sum of squares (WCSS) against the number of clusters.

python

Copy code

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


# Use the elbow method to determine the optimal number of clusters
wcss = []  # Within-cluster sum of squares


for i in range(1, 11):
    kmeans = KMeans(n_clusters=i, init='k-means++', max_iter=300, n_init=10, random_state=42)
    kmeans.fit(scaled_features)
    wcss.append(kmeans.inertia_)


# Plot the elbow method results
plt.figure(figsize=(10, 5))
plt.plot(range(1, 11), wcss, marker='o')
plt.title('Elbow Method')
plt.xlabel('Number of clusters')
plt.ylabel('WCSS')
plt.grid(True)
plt.show()
```

**Output:**

- The elbow plot shows the WCSS for each number of clusters. The "elbow" point indicates the optimal number of clusters. In this case, the optimal number of clusters is determined to be 4.

**2.4 Implement K-means clustering with the optimal number of clusters.**

python

Copy code

```
# Implement K-means clustering with 4 clusters
kmeans = KMeans(n_clusters=4, init='k-means++', max_iter=300, n_init=10, random_state=42)
clusters = kmeans.fit_predict(scaled_features)

# Add the cluster labels to the original dataframe
df['Cluster'] = clusters
```

**3. Analysis of Clusters**

**3.1 Analyze and profile the characteristics of each customer cluster.**

python

Copy code

```
# Analyze the characteristics of each cluster
cluster_profile = df.groupby('Cluster').mean(numeric_only=True)
print("\nCluster profiling:")
print(cluster_profile)
```

**Output:**

- The cluster profile provides the mean values of the numerical features for each cluster. This helps in understanding the characteristics of each cluster.

**3.2 Visualize the clusters using scatter plots.**

python

Copy code

```
# Visualize the clusters
plt.figure(figsize=(15, 7))

# Plot Age vs. Annual Income
plt.subplot(1, 3, 1)
plt.scatter(df['Age'], df['Annual Income (k$)'], c=df['Cluster'], cmap='viridis')
plt.title('Clusters of customers (Age vs. Annual Income)')
plt.xlabel('Age')
plt.ylabel('Annual Income (k$)')

# Plot Age vs. Spending Score
plt.subplot(1, 3, 2)
plt.scatter(df['Age'], df['Spending Score (1-100)'], c=df['Cluster'], cmap='viridis')
plt.title('Clusters of customers (Age vs. Spending Score)')
plt.xlabel('Age')
plt.ylabel('Spending Score (1-100)')

# Plot Annual Income vs. Spending Score
plt.subplot(1, 3, 3)
plt.scatter(df['Annual Income (k$)'], df['Spending Score (1-100)'], c=df['Cluster'], cmap='viridis')
plt.title('Clusters of customers (Annual Income vs. Spending Score)')
plt.xlabel('Annual Income (k$)')
plt.ylabel('Spending Score (1-100)')

plt.tight_layout()
plt.show()
```

**Output:**

- Scatter plots visualize the clusters based on different pairs of features. These plots help in understanding the distribution and separation of clusters.

**Conclusion**

This documentation outlines the process of applying K-means clustering to segment customer data using the Mall Customer Segmentation dataset. The steps include:

1. **Data Exploration:**
    - Loading the dataset
    - Conducting exploratory data analysis
2. **Clustering Implementation:**
    - Selecting relevant features
    - Standardizing the features
    - Determining the optimal number of clusters using the elbow method
    - Implementing K-means clustering
3. **Analysis of Clusters:**
    - Profiling the characteristics of each cluster
    - Visualizing the clusters using scatter plots

By following these steps, we can effectively segment customers based on their shopping behavior and gain insights into different customer groups. This information can be used for targeted marketing strategies and improving customer experiences.