Assignment No. 5

Implement K-Means clustering/ hierarchical clustering on sales_data_sample.csv dataset. Determine the number of clusters using the elbow method.

Step 1: Download the Dataset

1. **Dataset link:** Sample Sales Data
2. Go to the Kaggle dataset page.
3. Download the sales_data_sample.csv file.
4. Save the dataset in the directory where you will run the Jupyter notebook.

Step 2: Open Jupyter Notebook

1. **Open Jupyter Notebook:**
   - Launch **Jupyter Notebook**.
   - Navigate to the directory where you saved the sales_data_sample.csv file.
   - Create a new Python notebook.

Step 3: Import Necessary Libraries

```
import pandas as pd

import numpy as np

import matplotlib.pyplot as plt

from sklearn.cluster import KMeans
```

- **Explanation**: These libraries are used for data manipulation (pandas), numerical operations (numpy), data visualization (matplotlib), and implementing K-Means clustering (sklearn).

Step 4: Load and Inspect the Dataset

```
# Load the dataset

df = pd.read_csv("sales_data_sample.csv", encoding='latin')

df
```

- **Explanation**: The dataset is loaded into a pandas DataFrame called df. The encoding='latin' argument is specified to handle any special characters.

```
# Check the data types of the columns
df.dtypes
```

- **Explanation**: This line prints the data types of each column, which helps to understand the structure of the dataset.

Step 5: Select Features for Clustering

```
# Select columns at index positions 3 and 4 from df and convert them to a NumPy array
X = df.iloc[:, [3, 4]].values
```

- **Explanation**: Here, we are selecting the columns at index positions 3 and 4 from the DataFrame for clustering. These columns are likely to represent numerical features relevant for clustering (e.g., Total Price, Quantity, etc.). The selected features are converted into a NumPy array X.

Step 6: Calculate WCSS for K-Means Clustering

Using the Elbow Method

The Elbow method helps determine the optimal number of clusters (K) by plotting the Within-Cluster Sum of Squares (WCSS) against the number of clusters.

```
# Initialize a list to store WCSS values
wcss = []   # within-cluster sum of squares


# Calculate WCSS for KMeans clustering from K=1 to K=10
for i in range(1, 11):
    # Initialize the KMeans model with 'i' clusters
    kmeans = KMeans(n_clusters=i, init="k-means++", random_state=42)
    # Fit the model to the data
    kmeans.fit(X)
    # Append the inertia (WCSS) value to the list
    wcss.append(kmeans.inertia_)
```

- **Explanation**:
    - The loop iterates through possible cluster counts from 1 to 10.
    - For each i, a KMeans model is initialized and fitted to the data.

- The inertia_ attribute provides the WCSS for that number of clusters, which is added to the wcss list.

Step 7: Plot the Elbow Method

```
# Plot the Elbow method

ks = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10]

plt.plot(ks, wcss, 'bx-')

plt.title("Elbow Method")

plt.xlabel("K value")

plt.ylabel("WCSS")

plt.show()
```

- **Explanation**:
    - This code generates a plot of the WCSS values against the number of clusters (K).
    - The plot helps visually determine the optimal K value where the WCSS starts to decrease at a slower rate (the "elbow" point).

Step 8: Dataset Summary Statistics

```
# Display summary statistics of the dataset

df.describe()
```

- **Explanation**: This line provides summary statistics of the numerical columns in the dataset, such as count, mean, standard deviation, minimum, and maximum values. This can help understand the distribution of the data.