Step 1: Import Libraries

First, we need to import the libraries we will use. We use pandas to handle datasets, matplotlib to plot data, and scikit-learn (sklearn) to perform K-Means clustering.

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

Step 2: Load the Dataset

We load the dataset using pandas. In this case, we will use a simple CSV file containing points with X and Y coordinates.

```
data = pd.read_csv('kmeans_dataset.csv')
print(data)
```

Step 3: Select Features for Clustering

We only need the X and Y values for clustering, not the labels like A1 or B1.

```
X = data[['X','Y']]
```

Step 4: Apply K-Means Clustering

We create a KMeans object from sklearn, define the number of clusters (k), and then fit the model to our data. Here, we try with k=2 or k=3.

```
kmeans = KMeans(n_clusters=2)
kmeans.fit(X)
print(kmeans.labels_)
```

Step 5: Add Cluster Labels to the Dataset

After fitting, we get labels for each data point showing which cluster it belongs to.

```
data['Cluster'] = kmeans.labels_
print(data)
```

Step 6: Visualize the Clusters

We can plot the data points and color them according to their cluster. We can also plot the centroids given by KMeans.

```
plt.scatter(X['X'], X['Y'], c=kmeans.labels_, cmap='rainbow')
plt.scatter(kmeans.cluster_centers_[:,0], kmeans.cluster_centers_[:,1], color='black', marker='X
plt.xlabel('X')
plt.ylabel('Y')
plt.title('K-Means Clustering')
plt.show()
```

Final Note:

This process shows how K-Means clustering groups points into clusters based on their distances. You can change the number of clusters (k) to see how results differ.