K-Means Clustering Code Explanation

This document explains how the K-Means clustering algorithm works in the provided Python code. The code generates sample data, applies the K-Means algorithm, and visualizes the resulting clusters and their centroids.

1. Importing Libraries

The following libraries are imported at the beginning of the code:

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

KMeans is the clustering algorithm used to group data.
make_blobs is used to generate artificial data that naturally forms clusters.
matplotlib.pyplot is used to create scatter plots for visualization.

2. Creating Example Data

The following line generates 300 sample data points that form three clusters:

```
X, _ = make_blobs(n_samples=300, centers=3, random_state=42)
```

In this line:

- n_samples=300 means the dataset contains 300 points.
- centers=3 means the data will be distributed into three natural groups.
- random_state=42 ensures the data is the same every time the code is run.

The variable X contains all the coordinates of the generated points. The second value (_) contains the true labels, but these are not used because clustering is an unsupervised method.

3. Running the K-Means Algorithm

The K-Means model is created and then trained using the following code:

```
kmeans = KMeans(n_clusters=3)
kmeans.fit(X)
```

Here:

- n_clusters=3 tells the algorithm to divide the data into three clusters.
- fit(X) runs the clustering process on the dataset.

When the algorithm runs, K-Means follows these steps:

1. It selects three initial centroid positions.

2. It assigns every data point to the nearest centroid.
3. It recalculates each centroid as the average (mean) of all points assigned to it.
4. It repeats the assignment and recalculation steps until the centroids stop moving.

This iterative process is why the algorithm is called K-Means: “K” refers to the number of clusters, and “Means” refers to using the mean value to compute the cluster centers.

4. Extracting Clustering Results

After fitting the model, two important values are extracted:

```
labels = kmeans.labels_
centroids = kmeans.cluster_centers_
```

labels contains the cluster number for each data point. For example, it may assign points to cluster 0, 1, or 2 depending on where they belong.

centroids contains the final coordinates of the three cluster centers. Each centroid represents the center of a group of points and is computed using the mean value of all points in that cluster.

5. Visualizing the Results

The final section of the code creates a scatter plot to display the clusters and centroid locations:

```
plt.scatter(X[:, 0], X[:, 1], c=labels)
plt.scatter(centroids[:, 0], centroids[:, 1], s=200, c="red", marker="X")
plt.title("Clustering K-Means")
plt.show()
```

The first scatter plot displays all data points. Each point is colored according to the cluster label assigned by the algorithm.

The second scatter plot displays the cluster centroids. They are drawn as large red X markers so that their positions are easy to identify.

The result is a visual representation of:

- how the data was grouped into clusters,
- and where the algorithm placed the center of each cluster.