# Trained K-means Model

This code snippet demonstrates how to train a K-means clustering model with the optimal number of clusters (in this case, 4) on a dataset called “processed data”.

Train the k-means model with the optimal number of clusters

optimal\_k = 4  # Assuming 4 is the optimal number of clusters

kmeans = KMeans(n\_clusters=optimal\_k, random\_state=42)

k-means.fit(processed data)

labels = k-means.labels\_

print("K-means model trained.")

1. **Set Optimal Number of Clusters**:

* optimal k = 4: This line sets the number of clusters (k) to 4, based on the analysis using the Elbow Method and Silhouette Score.

1. **Initialize KMeans Model**:

* k-means = K-Means(n\_clusters=optimal k, random\_state=42): This line initializes the KMeans clustering model from the sklearn.cluster module with n clusters=4. The random\_state=42 ensures reproducibility of the results by fixing the random seed.

1. **Fit the Model**:

* kmeans.fit(processed\_data): This line trains the K-Means model on the dataset processed\_data. The fit method computes the cluster centres and assigns each point in the dataset to the nearest cluster centre.

1. **Get Cluster Labels**:

* labels = k-means.labels\_: After fitting the model, this line retrieves the cluster labels for each point in the dataset. K-means.labels\_ is an array where each element represents the cluster index assigned to the corresponding point in processed\_data.

### **Explanation in Terms of the Trained K-means Model**

* **Initialization**:
  + The KMeans model is initialized to create 4 clusters.
  + The random\_state parameter ensures that the results are consistent across different runs of the code.
* **Training**:
  + The fit method is called on the processed\_data to train the model. This involves:
    - **Randomly initializing cluster centres** (or using another initialization method like k-means++).
    - **Iteratively refining cluster centres** by alternating between assigning points to the nearest cluster centre and updating the cluster centres based on the assigned points until convergence.
* **Output**:
  + The labels array contains the cluster index (ranging from 0 to 3) for each data point, indicating which cluster each point belongs to.

### **Summary**

This code trains a KMeans clustering model with 4 clusters on the given dataset, assigns each point to one of these clusters, and stores the cluster assignments in the labels array. This is useful for understanding the structure of the data and for tasks like segmentation, pattern recognition, and data summarization.