import matplotlib.pyplot as plt

* **Imports the Matplotlib library:** This library is used for creating static, animated, and interactive visualizations in Python. The pyplot module (aliased as plt) provides a MATLAB-like interface for plotting.

from sklearn.datasets import make\_blobs

* **Imports make\_blobs function from sklearn:** This function is used to generate synthetic datasets for clustering. It creates a specified number of clusters in a 2D or higher-dimensional space, with points distributed normally around the cluster centers.

from sklearn.cluster import KMeans

* **Imports KMeans class from sklearn:** This class implements the K-Means clustering algorithm, which partitions the data into k clusters by minimizing the variance within each cluster.

# Generating synthetic data

X, \_ = make\_blobs(n\_samples=300, centers=4, cluster\_std=1.0, random\_state=42)

* **Generates synthetic data using make\_blobs:**
  + n\_samples=300: Specifies that 300 data points will be generated.
  + centers=4: Specifies that the data will have 4 clusters.
  + cluster\_std=1.0: The standard deviation of the clusters, controlling how spread out the points in each cluster will be.
  + random\_state=42: Ensures reproducibility of the results by setting a seed for random number generation.
* The output is a tuple where X contains the coordinates of the data points and the second element (ignored by using \_) contains the true labels (not used here).

# Initialize K-Means with the number of clusters

kmeans = KMeans(n\_clusters=4)

* **Creates an instance of KMeans:** This initializes the KMeans object with n\_clusters=4, indicating that we want to find 4 clusters in our data.

# Fit the K-Means model to the data

kmeans.fit(X)

* **Fits the KMeans model to the data:** This method computes the centroids and assigns each data point in X to the nearest cluster. The KMeans algorithm iteratively updates the centroids until convergence.

# Predict cluster labels

cluster\_labels = kmeans.predict(X)

* **Predicts cluster labels for each point:** After fitting the model, this line assigns each data point in X to one of the clusters based on the learned centroids. The predicted labels are stored in cluster\_labels.

# Visualize the clusters

plt.figure(figsize=(7,5))

* **Creates a new figure for plotting:** This initializes a new figure with a specified size of 7 inches by 5 inches.

plt.scatter(X[:, 0], X[:, 1], c=cluster\_labels, cmap='viridis', edgecolors='k')

* **Creates a scatter plot of the data points:**
  + X[:, 0] and X[:, 1]: Refers to the first and second dimensions (features) of the dataset respectively, creating a 2D plot.
  + c=cluster\_labels: Colors each point according to its assigned cluster label.
  + cmap='viridis': Specifies the colormap to use for coloring the points; 'viridis' is a perceptually uniform color map.
  + edgecolors='k': Sets the edge color of the points to black.

plt.scatter(kmeans.cluster\_centers\_[:, 0], kmeans.cluster\_centers\_[:, 1],

marker='o', s=200, color='red', label='Centroids')

* **Plots the cluster centroids:**
  + kmeans.cluster\_centers\_[:, 0] and kmeans.cluster\_centers\_[:, 1]: Refers to the x and y coordinates of the centroids found by KMeans.
  + marker='o': Specifies the shape of the marker for centroids.
  + s=200: Defines the size of the centroid markers (these will be larger than the other points).
  + color='red': Colors the centroid markers red.
  + label='Centroids': Assigns a label for the centroid points, which can be used in the legend.

plt.title('K-Means Clustering')

* **Sets the title of the plot:** Adds the title to the figure, which will appear at the top of the plot.

plt.xlabel('X')

* **Labels the x-axis:** Adds a label to the horizontal x-axis.

plt.ylabel('Y')

* **Labels the y-axis:** Adds a label to the vertical y-axis.

plt.legend()

* **Displays the legend:** This will show a legend on the plot for the labeled points, which includes the centroid markers.

plt.show()

* **Displays the plot:** This command renders the plot and makes it visible. It will bring up a window (or display in-line if you're using a Jupyter notebook) containing the scatter plot of clustered data points along with their centroids.