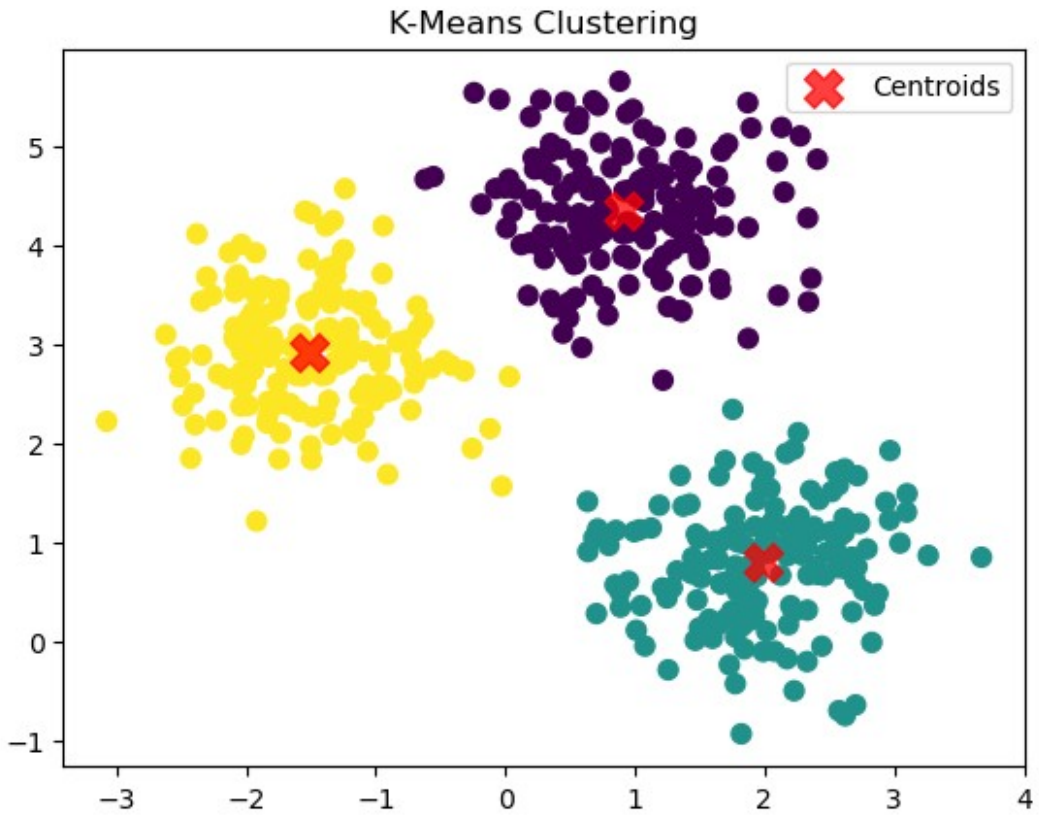
```
import numpy as np
import matplotlib.pyplot as plt
from sklearn.cluster import KMeans
from sklearn.datasets import make_blobs
from scipy.optimize import linear_sum_assignment

# Generate sample data
X, y_true = make_blobs(n_samples=500, centers=3, cluster_std=0.60,
random_state=0)

# Apply KMeans
kmeans = KMeans(n_clusters=3, random_state=0)
kmeans.fit(X)
y_kmeans = kmeans.predict(X)

# Plot the result
plt.scatter(X[:, 0], X[:, 1], c=y_kmeans, s=50, cmap='viridis')
centers = kmeans.cluster_centers_
plt.scatter(centers[:, 0], centers[:, 1], c='red', s=200, alpha=0.75,
marker='X', label='Centroids')
plt.title("K-Means Clustering")
plt.legend()
plt.show()

# Function to calculate accuracy after optimal label mapping
def cluster_accuracy(y_true, y_pred):
    y_true = np.array(y_true)
    y_pred = np.array(y_pred)
    D = max(y_pred.max(), y_true.max()) + 1
    cost_matrix = np.zeros((D, D), dtype=int)
    for i in range(y_pred.size):
        cost_matrix[y_pred[i], y_true[i]] += 1
    row_ind, col_ind = linear_sum_assignment(cost_matrix.max() -
cost_matrix)
    accuracy = cost_matrix[row_ind, col_ind].sum() / y_pred.size
    return accuracy

acc = cluster_accuracy(y_true, y_kmeans)
print(f"KMeans clustering accuracy: {acc:.3f}")
```

```
C:\Users\MGM\anaconda3\Lib\site-packages\sklearn\cluster\
_kmeans.py:1429: UserWarning: KMeans is known to have a memory leak on
Windows with MKL, when there are less chunks than available threads.
You can avoid it by setting the environment variable
OMP_NUM_THREADS=2.
  warnings.warn(
```




```
KMeans clustering accuracy: 1.000
```