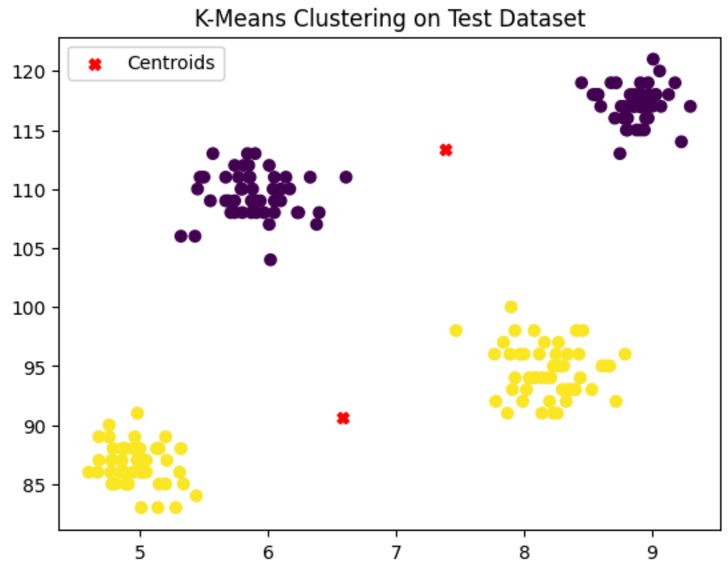
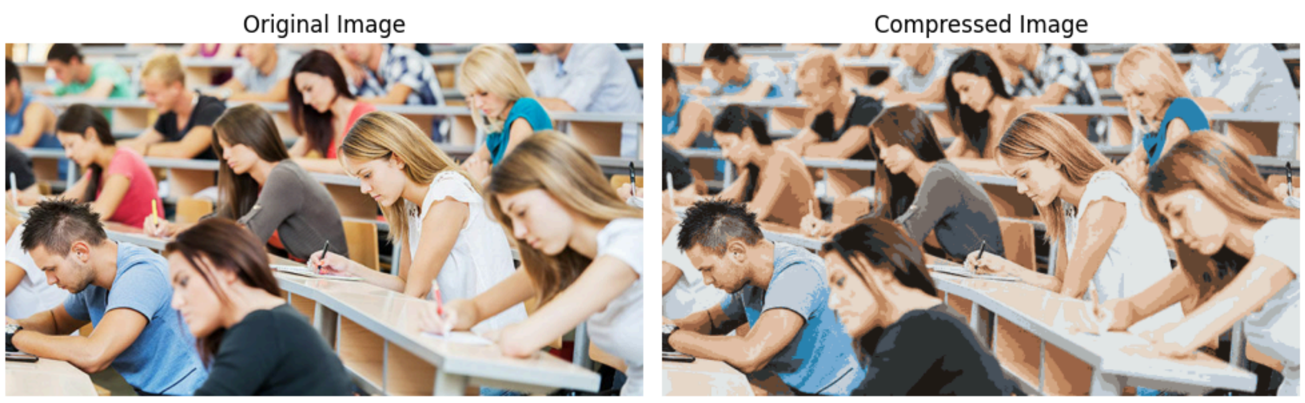
```
import numpy as np
import matplotlib.pyplot as plt
from sklearn.cluster import KMeans

# Test Dataset
data = pd.read_csv('/content/student_clustering.csv')
data = np.array(data)
# 1. Random Initialization of Centroids
def initialize_random_centroids(data, k):
    indices = np.random.choice(len(data), k, replace=False)
    return data[indices]

# Example: Initialize 2 Centroids
k = 2
initial_centroids = initialize_random_centroids(data, k)
print("Initial Centroids:", initial_centroids)

# 2. Finding Closest Centroids
def find_closest_centroids(data, centroids):
    distances = np.linalg.norm(data[:, np.newaxis] - centroids, axis=2)
    return np.argmin(distances, axis=1)

# Example: Assign Data Points to Closest Centroids
labels = find_closest_centroids(data, initial_centroids)
print("Labels:", labels)

# 3. Computing Centroid Means
def compute_centroid_means(data, labels, k):
    return np.array([data[labels == i].mean(axis=0) for i in range(k)])

# Example: Compute New Centroids
new_centroids = compute_centroid_means(data, labels, k)
print("Updated Centroids:", new_centroids)

# 4. Iterative K-means Algorithm
def run_k_means(data, initial_centroids, max_iters):
    centroids = initial_centroids
    for i in range(max_iters):
        labels = find_closest_centroids(data, centroids)
        centroids = compute_centroid_means(data, labels, len(centroids))
    return centroids, labels

# Run K-means on Test Data
final_centroids, final_labels = run_k_means(data, initial_centroids, max_iters=10)
print("Final Centroids:", final_centroids)
print("Final Labels:", final_labels)

# 5. Equivalent Code Using Scikit-Learn
kmeans = KMeans(n_clusters=2, random_state=42)
kmeans.fit(data)

# Compare Results
print("Scikit-Learn Centroids:", kmeans.cluster_centers_)
print("Scikit-Learn Labels:", kmeans.labels_)

# Visualization
plt.scatter(data[:, 0], data[:, 1], c=kmeans.labels_, cmap='viridis', marker='o')
plt.scatter(kmeans.cluster_centers_[:, 0], kmeans.cluster_centers_[:, 1], c='red', marker='X', label='Centroids')
plt.title("K-Means Clustering on Test Dataset")
plt.legend()
plt.show()
```

```
Initial Centroids: [[  6.61 111.  ]
 [  5.15  85.  ]]
Labels: [1 0 1 1 0 0 0 0 0 1 1 0 1 1 0 1 0 1 0 0 0 1 1 1 1 0 1 0 1 0 0 0 0 0 1 1 0
 0 1 0 1 1 1 1 0 0 1 0 0 0 1 1 0 1 0 0 0 0 0 0 0 1 1 0 1 0 1 1 0 1 0 0 1 1
 0 0 0 0 1 1 1 0 0 1 0 1 0 0 0 0 0 0 1 1 1 1 0 1 1 0 0 1 1 0 1 1 1 1 0 0 1
 0 0 0 1 0 1 0 0 1 1 0 1 0 1 1 0 0 1 1 1 0 1 1 0 0 0 1 0 1 1 1 0 1 1 0 1 0
 0 1 0 1 0 0 1 1 0 0 0 1 1 0 0 1 0 1 0 1 1 0 0 0 0 1 1 0 0 0 1 0 0 0 0 0 1
 1 1 1 0 0 1 1 1 1 1 1 0 0 0 0]
Updated Centroids: [[  7.42037736 112.49056604]
 [  6.4906383   90.15957447]]
Final Centroids: [[  7.3831 113.34  ]
 [  6.5837  90.65  ]]
Final Labels: [1 0 1 1 0 0 1 0 0 1 1 0 1 1 0 1 0 1 0 0 1 1 1 1 1 1 1 0 1 0 0 0 0 0 1 1 0
 0 1 0 1 1 1 1 0 0 1 0 0 0 1 1 0 1 0 0 0 0 0 0 0 1 1 0 1 0 1 1 0 1 0 0 1 1
 0 0 0 0 1 1 1 0 0 1 0 1 0 0 0 0 0 0 1 1 1 1 0 1 1 0 0 1 1 0 1 1 1 1 0 0 1
 0 0 0 1 0 1 0 0 1 1 0 1 0 1 1 1 0 1 1 1 0 1 1 0 0 0 1 1 1 1 1 0 1 1 0 1 0
 0 1 0 1 0 0 1 1 0 0 0 1 1 0 0 1 0 1 0 1 1 0 0 0 0 1 1 1 0 0 1 0 0 0 0 0 1
 1 1 1 0 0 1 1 1 1 1 1 0 0 0 0]
Scikit-Learn Centroids: [[  7.3831 113.34  ]
 [  6.5837  90.65  ]]
Scikit-Learn Labels: [1 0 1 1 0 0 1 0 0 1 1 0 1 1 0 1 0 1 0 0 1 1 1 1 1 1 1 0 1 0 0 0 0 0 1 1 0
 0 1 0 1 1 1 1 0 0 1 0 0 0 1 1 0 1 0 0 0 0 0 0 0 1 1 0 1 0 1 1 0 1 0 0 1 1
 0 0 0 0 1 1 1 0 0 1 0 1 0 0 0 0 0 0 1 1 1 1 0 1 1 0 0 1 1 0 1 1 1 1 0 0 1
 0 0 0 1 0 1 0 0 1 1 0 1 0 1 1 1 0 1 1 1 0 1 1 0 0 0 1 1 1 1 1 0 1 1 0 1 0
 0 1 0 1 0 0 1 1 0 0 0 1 1 0 0 1 0 1 0 1 1 0 0 0 0 1 1 1 0 0 1 0 0 0 0 0 1
 1 1 1 0 0 1 1 1 1 1 1 0 0 0 0]
```



```
import numpy as np
from sklearn.cluster import KMeans
import matplotlib.pyplot as plt
from PIL import Image

# Load image
image_path = "/content/clg.jpg"  # Update with your image path
image = Image.open(image_path)
image = np.array(image)

# Reshape image data to a 2D array: (num_pixels, 3)
rows, cols, channels = image.shape
image_2d = image.reshape(rows * cols, channels)

# Run K-Means
kmeans = KMeans(n_clusters=16, random_state=0)
kmeans.fit(image_2d)

# Get centroids and labels
centroids = kmeans.cluster_centers_
labels = kmeans.labels_

# Compress image
compressed_image = centroids[labels].reshape(rows, cols, channels).astype(np.uint8)

# Display original and compressed images
plt.figure(figsize=(10, 5))
plt.subplot(1, 2, 1)
plt.imshow(image)
plt.title("Original Image")
plt.axis("off")

plt.subplot(1, 2, 2)
plt.imshow(compressed_image)
plt.title("Compressed Image")
plt.axis("off")

plt.tight_layout()
plt.show()
```



Start coding or generate with AI.