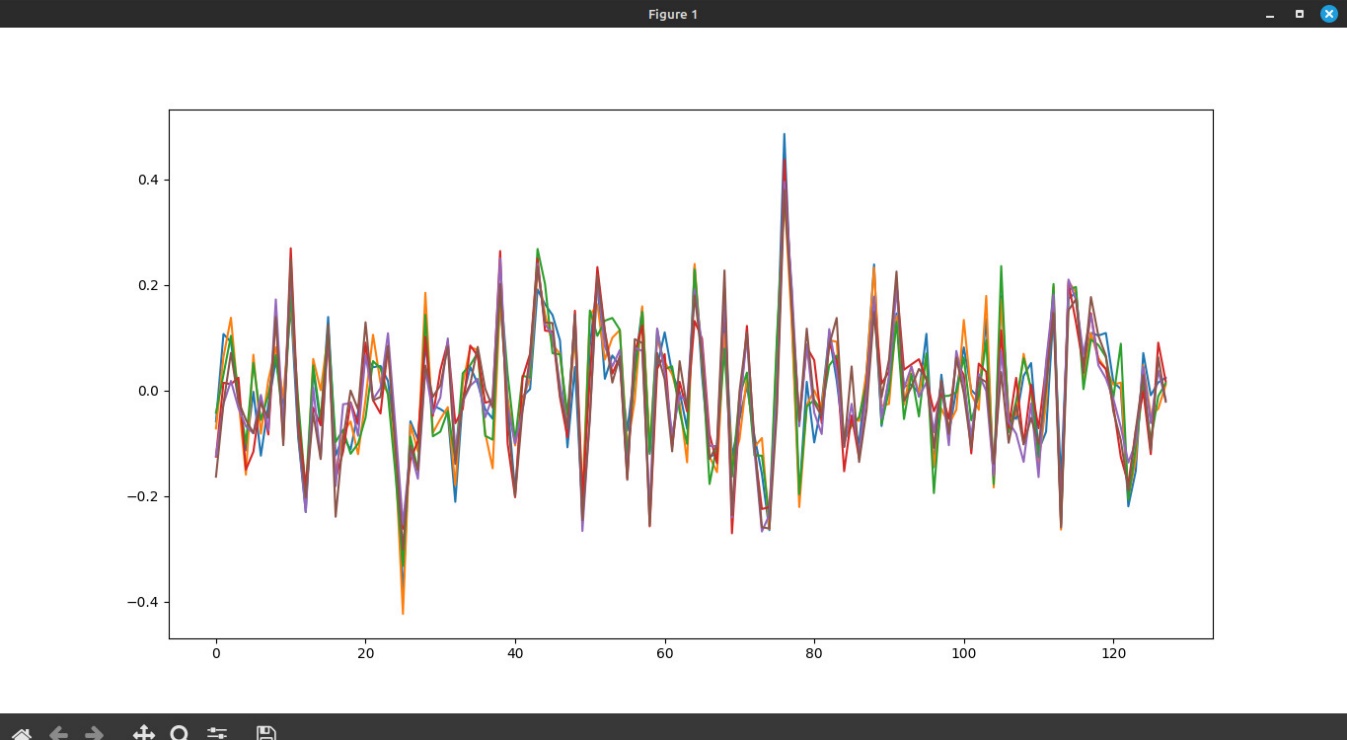
|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28  29  30  31  32  33  34  35  36  37  38  39  40  41  42  43  44  45  46  47  48  49  50  51  52  53  54  55  56  57  58  59  60  61  62  63  64  65  66  67  68  69  70  71  72  73  74  75  76  77  78  79  80  81  82  83  84  85  86  87  88  89  90 | #Librerias  **import** **face\_recognition**  **import** **cv2**  **import** **numpy** **as** **np**  **from** **matplotlib** **import** pyplot **as** plt  # Load image:  img5 = face\_recognition.load\_image\_file("5.jpg")  img6 = face\_recognition.load\_image\_file("6.jpg")  img7 = face\_recognition.load\_image\_file("7.jpg")  img8 = face\_recognition.load\_image\_file("8.jpg")  img9 = face\_recognition.load\_image\_file("9.jpg")  img10 = face\_recognition.load\_image\_file("10.jpg")  # Calculate the encodings for every face of the image:  encodings5 = face\_recognition.face\_encodings(img5)  encodings6 = face\_recognition.face\_encodings(img6)  encodings7 = face\_recognition.face\_encodings(img7)  encodings8 = face\_recognition.face\_encodings(img8)  encodings9 = face\_recognition.face\_encodings(img9)  encodings10 = face\_recognition.face\_encodings(img10)  # Show the first encoding:  **print**(len(encodings5))  **print**(encodings5[**0**])  **print**('')  **print**(len(encodings6))  **print**(encodings6[**0**])  **print**('')  **print**(len(encodings7))  **print**(encodings7[**0**])  **print**('')  **print**(len(encodings8))  **print**(encodings8[**0**])  **print**('')  **print**(len(encodings9))  **print**(encodings9[**0**])  **print**('')  **print**(len(encodings10))  **print**(encodings10[**0**])  **print**('')  plt.plot(encodings5[**0**])  plt.plot(encodings6[**0**])  plt.plot(encodings7[**0**])  plt.plot(encodings8[**0**])  plt.plot(encodings9[**0**])  plt.plot(encodings10[**0**])  plt.show()  # Create data (encodings of the images):  data\_A = np.vstack((encodings5[**0**], encodings6[**0**], encodings7[**0**]))  data\_B = np.vstack((encodings8[**0**], encodings9[**0**], encodings10[**0**]))  # Convertir datos a tipo float32  data\_A = np.float32(data\_A)  data\_B = np.float32(data\_B)  # Reorganizar los datos para tener solo 2 dimensiones  data\_A = data\_A.reshape(-**1**, **1**)  data\_B = data\_B.reshape(-**1**, **1**)  # Ejecutar K-means clustering para grupo A  criteria\_A = (cv2.TERM\_CRITERIA\_EPS + cv2.TERM\_CRITERIA\_MAX\_ITER, **20**, **1.0**)  ret\_A, label\_A, center\_A = cv2.kmeans(data\_A, **2**, None, criteria\_A, **10**, cv2.KMEANS\_RANDOM\_CENTERS)  # Ejecutar K-means clustering para grupo B  criteria\_B = (cv2.TERM\_CRITERIA\_EPS + cv2.TERM\_CRITERIA\_MAX\_ITER, **20**, **1.0**)  ret\_B, label\_B, center\_B = cv2.kmeans(data\_B, **2**, None, criteria\_B, **10**, cv2.KMEANS\_RANDOM\_CENTERS)  # Obtener los índices de cada grupo  indices\_A = np.where(label\_A.ravel() == **0**)[**0**]  indices\_B = np.where(label\_B.ravel() == **0**)[**0**]  # Obtener los puntos correspondientes a cada grupo  A1 = data\_A[indices\_A]  A2 = data\_A[np.where(label\_A.ravel() == **1**)[**0**]]  B1 = data\_B[indices\_B]  B2 = data\_B[np.where(label\_B.ravel() == **1**)[**0**]]  # Plot the results  plt.plot(A1[:, **0**], 'b-', linewidth=**1**, label='Group A1')  plt.plot(A2[:, **0**], 'b-', linewidth=**1**, label='Group A2')  plt.plot(B1[:, **0**], 'g-', linewidth=**1**, label='Group B1')  plt.plot(B2[:, **0**], 'g-', linewidth=**1**, label='Group B2')  plt.scatter(center\_A[:, **0**], center\_A[:, **0**], s=**100**, c='r', label='Centers A')  plt.scatter(center\_B[:, **0**], center\_B[:, **0**], s=**100**, c='m', label='Centers B')  plt.legend()  plt.show() |



“Encodings” de las imágenes, una grafica que junta todo el “encodings” de todas las imágenes juntas

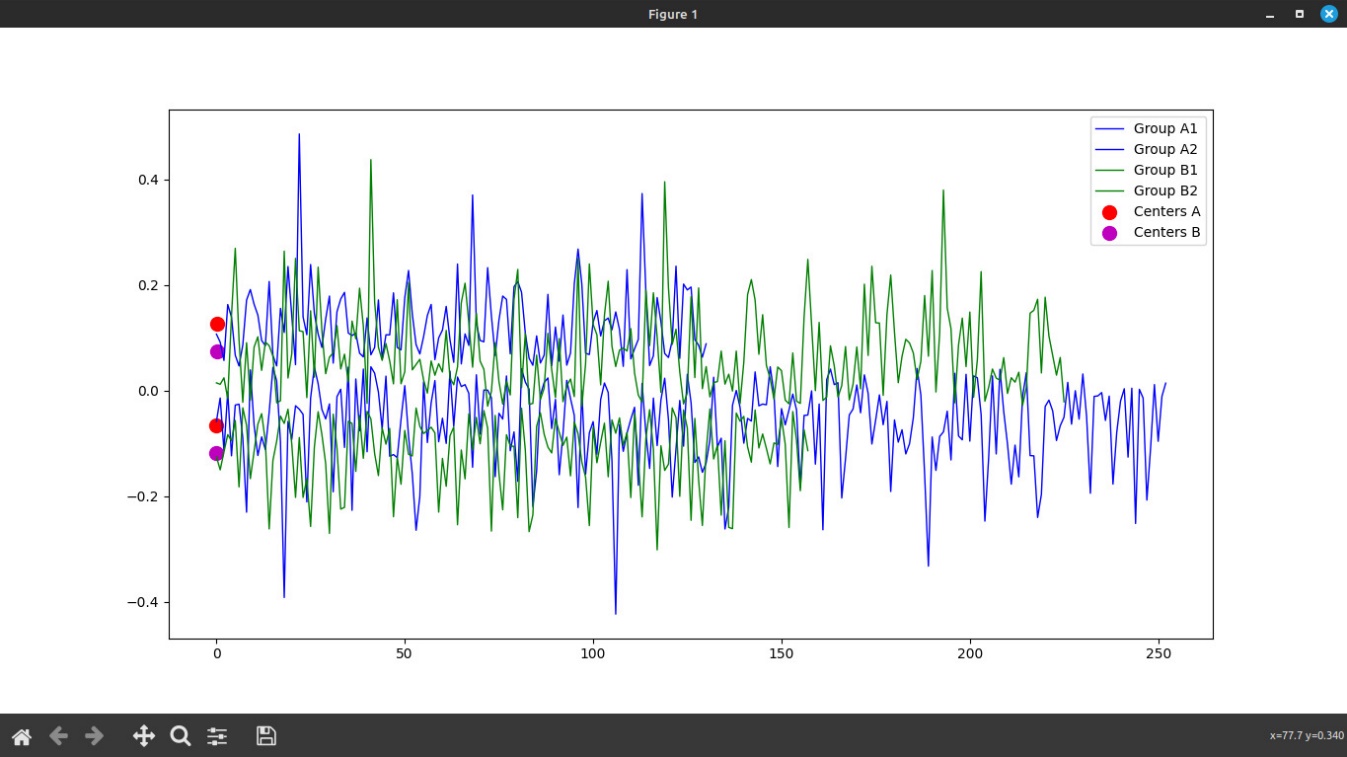


Imagen usando el algoritmo K-means para separar en closters cada grupo, el grupo A1 Y A2 son las fotos del profesor, los grupos B1 y B2 son mis fotos, le intente dar un color a cada grupo para que hubieran líneas, pero como están muy juntos los puntos no se logran apreciar al completo y se ve con poca legibilidad.

Lo primero que hacemos es cargar las imágenes para poder sacar su “encoding”, después creamos dos matrices data\_A y data\_B, que contienen los encodings de las imágenes que deseas clasificar en los grupos A y B, respectivamente.

Se utiliza np.float32() para convertir los datos data\_A y data\_B al tipo float32. Luego, se utiliza reshape() para reorganizar los datos de manera que tengan solo 2 dimensiones.

Luego, aplicamos el algoritmo de clustering K-means separadamente para cada grupo. Creamos dos conjuntos de criterios diferentes, criteria\_A y criteria\_B, y utilizamos estos criterios para ejecutar cv2.kmeans en data\_A y data\_B, respectivamente.

Después de obtener las etiquetas y los centros para cada grupo, separamos los datos en subconjuntos A1, A2, B1 y B2 según las etiquetas.

Los valores A1, A2, B1 y B2 representan los subconjuntos de puntos obtenidos después de aplicar el algoritmo de K-means y separar los datos en los grupos A y B.

Finalmente, trazamos los puntos de los grupos A y B, junto con los centros de cada grupo, en un gráfico para visualizar los resultados. Asegúrate de ajustar los colores y el estilo de trazado según tus preferencias.