César Isaí García Cornejo
cesar.cornejo@cimat.mx
Ciencia de Datos

Tarea 4

Semestre 3

CIMAT

Ejercicio 1
Retomemos el conjunto de datos Iris que podemos llamar gracias a la función *load_iris* del paquete *sklearn.datasets*. En estudios de datos anteriores vimos que las dos variables *petal length* y *petal width* son muy correlacionadas con las categorías de las plantas.

a) Usando los datos bidimensionales correspondientes a *petal length* y *petal width* de cada planta, observar el resultado del clustering *k-medias* de los datos con 2,3 y 4 clases. Se usará la función *KMeans* de *sklearn.cluster*.

b) Escribir una función *Riesgo*(X,y,C) que toma los datos X, las categorías correspondientes y los centroides C y que responde el valor de la función G (definida en el curso).

c) Graficar el resultado de *Riesgo* para k que varia entre 2 y 10.

d) Para elegir el mejor k una técnica famosa el la *Elbow Method*. Hacer una búsqueda de referencias que explican la técnica. Implementar la técnica para la grafica del inciso anterior.

Solución:

a) Usaremos los datos de Iris correspondientes a lo largo y ancho del pétalo. Dichos datos con la clasificación predeterminada de la base de datos (a priori) se observan en el siguiente gráfico

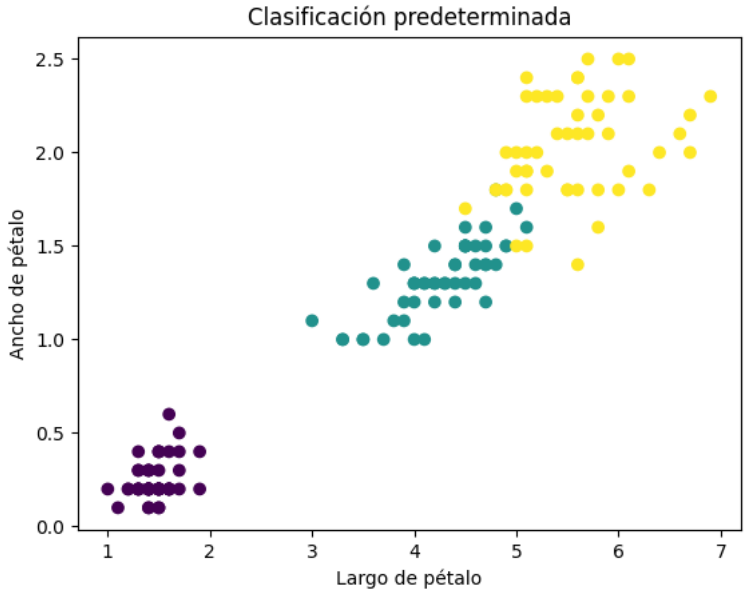

Figura 1: Largo y ancho de los pétalos en la base de datos Iris.

Entonces, ocultaremos la etiqueta de las clases para proceder a Implementar el método de k-means con la paquetería requerida. Ampliamente hablando, la implementación fundamental está en el código

```
kmeans = KMeans(n_clusters=n)
kmeans.fit(X)
y_kmeans = kmeans.fit_predict(X)
```

donde n es el número de cluster fijado por el experimentador.

Así, dicha clusterización para 2,3 y 4 clases se muestan en las graficamente en las siguientes figuras.

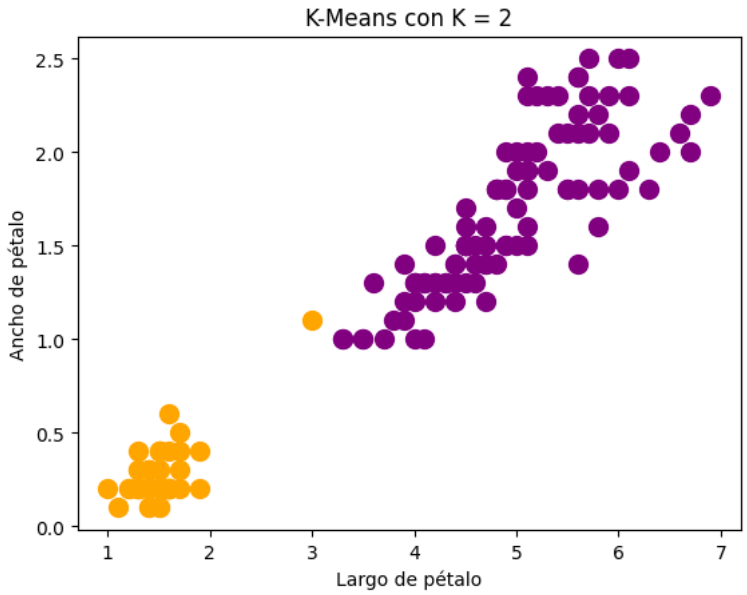

Figura 2: Método de k-means para 2 clusters.

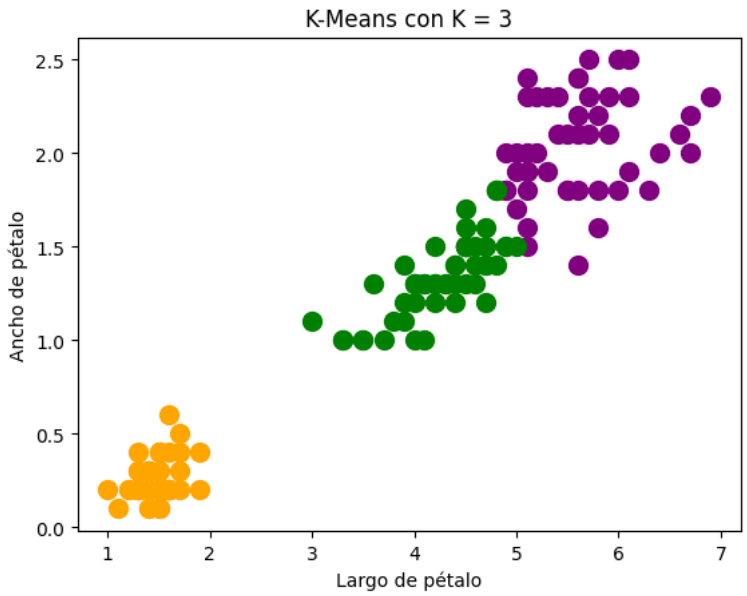

Figura 3: Método de k-means para 3 clusters.

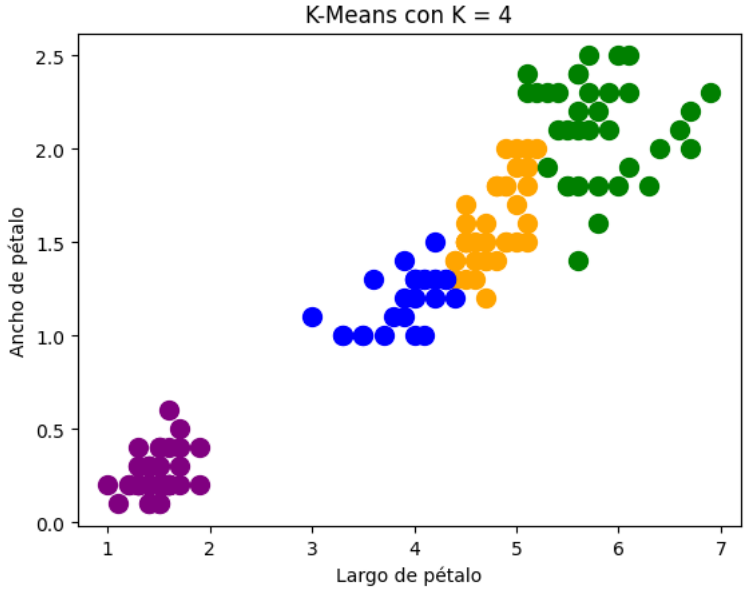

Figura 4: Método de k-means para 4 clusters.

El código completo está anexo en la actividad correspondiente.

b) La función de riesgo implementada se basa en usar DataFrames de pandas para que una vez hecha la clusterización podamos filtrar aquellos datos que pertenecen a la misma clase y poder calcular su centroide. Luego, calculamos el cuadrado de la distancia y sumamos obteniendo el riesgo para dicha clase. Hacemos lo mismo para cada clase y sumamos.

La implementación es

```
def riesgo(X,k):
    # Base de datos
    df = pd.DataFrame(X, columns=['Largo', 'Ancho'])

    kmeans = KMeans(n_clusters=k)
    kmeans.fit(X)
    y_kmeans = kmeans.fit_predict(X)
    df['Cluster'] = y_kmeans

    riesgo = 0
    for i in range(n):

        #Filtrar por clase i
        d0 = df[df['Cluster'] == i]

        #Calculo de la media
        x = d0['Largo']-d0['Largo'].mean()
        y = d0['Ancho']-d0['Ancho'].mean()

        # plt.plot(d0['Largo'].mean(), d0['Ancho'].mean(), '*k') #plot
        dist = (x**2 + y**2).sum()
        riesgo += dist
        # print(dist)

    return riesgo
```

c) Hacemos un ciclo para k, los valores entre 2 y 10, con el fin de que podamos calcular el riesgo con el método de k-means para cada valor de k dentro del ciclo. Dicho resultado nos dá la siguiente grafica

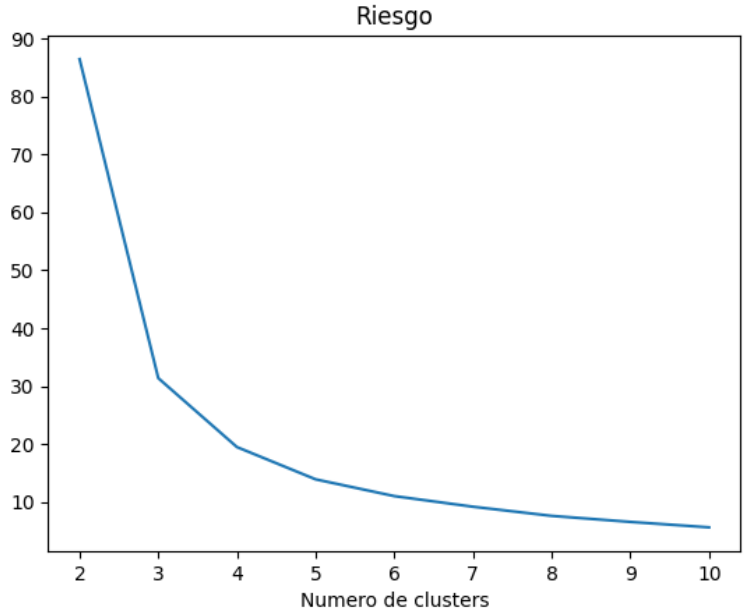

Figura 5: Riesgo dado por el método de K-means con k clusters.

d) Por último la interpretación dada a la gráfica anterior con el método del codo, observamos donde el riesgo cambia de forma que aparenta un ángulo de 90 grados, que como podemos observar este valor se da en k = 3. Es decir, el valor óptimo de cluster es 3, que se corresponde con el etiquetado correcto.