Introducción

Muchas veces. al trabajar con grandes conjuntos de datos requerimos segmentarlos en ciertos grupos con los que podamos trabajar de forma separada para que el análisis sea más sencillo. K-medias es un algoritmo de clasificación no supervisada (clusterización) que agrupa objetos en k grupos basándose en sus características. El agrupamiento se realiza minimizando la suma de distancias entre cada objeto y el centroide de su grupo o cluster. Se suele usar la distancia cuadrática.

El algoritmo consta de tres pasos:

1. **Inicialización:** una vez escogido el número de grupos, k, se establecen k centroides en el espacio de los datos, por ejemplo, escogiéndolos aleatoriamente.
2. **Asignación objetos a los centroides:** cada objeto de los datos es asignado a su centroide más cercano.
3. **Actualización centroides:** se actualiza la posición del centroide de cada grupo tomando como nuevo centroide la posición del promedio de los objetos pertenecientes a dicho grupo.

Se repiten los pasos 2 y 3 hasta que los centroides no se mueven, o se mueven por debajo de una distancia umbral en cada paso.

Existen tres criterios para terminar el algoritmo:

1. **Los centroides dejan de cambiar**. Es decir, después de múltiples iteraciones, los centroides de cada cluster no cambian. Por lo que se asume que el algoritmo ha convergido.
2. **Los puntos dejan de cambiar de cluster**. Parecido al anterior, pero esta vez no nos fijamos en los centroides, si no en los puntos que pertenecen a cada clúster. Cuando se observa que no hay un intercambio de clusters se asume que el modelo está entrenado.
3. **Límite de iteraciones**. Podemos fijar un número máximo de iteraciones que queremos que nuestro algoritmo ejecute antes de pararlo. Cuando llega a ese número, se asume que el modelo no va a mejorar drásticamente y se para el entrenamiento.

El algoritmo siempre termina, ya que no necesariamente encuentra la configuración óptima, la que se corresponde con el mínimo de la función objetivo. Hallar un mínimo de la función, a pesar de que no se trate del mínimo absoluto, garantiza un agrupamiento en el que los grupos son poco dispersos y se encuentran separados entre sí. El algoritmo es significativamente sensible a los centroides que se seleccionan inicialmente de manera aleatoria. Este efecto se puede reducir realizando varias corridas del método.

Mediante esta práctica se realizará la implementación del algoritmo k-medias para 40 patrones y grupos de 2 a 7 con el objetivo de identificar la forma en que trabaja para diferentes casos.

Código

```
import numpy as np
import matplotlib.pyplot as plt
from statistics import mean

semilla = int(input("Ingrese el valor de la semilla: "))
size = int(input("Ingrese el numero de patrones: ")) #Número de patrones
k = int(input("Ingrese el numero de grupos: ")) #Número de grupos/centroides
np.random.seed(semilla)
patrones=np.random.rand(size,2) #Permite generar los patrones
centroides=np.random.rand(k,2) #Permite generar los centroides
distancias=np.zeros((size,k)) #Sirve para almacenar las distancias
euclidianas de los patrones con los centroides

colores=['blue', 'orange', 'purple', 'g', 'r', 'violet', 'c', 'k']

"""Función para calcular la distancia euclidiana entre dos puntos a
partir de sus coordenadas"""
def distancia(x1,x2,y1,y2):
  return pow((x2-x1)**2+(y2-y1)**2,0.5)

"""Graficar centroides"""
for i in range(0,k):
  plt.scatter(centroides[i,0], centroides[i,1],marker='*',
color=colores[i], alpha=0.8, s=49)

"""Graficar patrones"""
for i in range(0,size):
  plt.scatter(patrones[i,0], patrones[i,1],marker='o', color=colores[7],
alpha=0.8, s=49)

plt.show()

"""Ciclo que permite realizar el algoritmo de k-medias a partir del
número de grupos y patrones definidos anteriormente. El ciclo termina
cuando los centroides ya no cambian de posición"""
for n in range(0,100):
  """Graficar centroides"""
  for i in range(0,k):
    plt.scatter(centroides[i,0], centroides[i,1],marker='*',
color=colores[i], alpha=0.8, s=49)

  """Graficar patrones"""
  for i in range(0,size):
    plt.scatter(patrones[i,0], patrones[i,1],marker='o',
color=colores[7], alpha=0.8, s=49)
```

```
  centroides_old = centroides.copy() #Crea una copia de las posiciones de
los centroides antes de cambiarlos de posición
  """Mediante el siguiente ciclo se obtienen las distancias de los
patrones con los centroides y se almacenan en un array
  en donde cada columna corresponde a la distancia de cada centroide"""
  for i in range(0,size):
    if(k == 2): #Distancias con 2 centroides
      distancias[i] = distancia(patrones[i,0], centroides[0,0],
patrones[i,1], centroides[0,1]), distancia(patrones[i,0],
centroides[1,0], patrones[i,1], centroides[1,1])
    elif(k == 3): #Distancias con 3 centroides
      distancias[i] = distancia(patrones[i,0], centroides[0,0],
patrones[i,1], centroides[0,1]), distancia(patrones[i,0],
centroides[1,0], patrones[i,1], centroides[1,1]),
distancia(patrones[i,0], centroides[2,0], patrones[i,1], centroides[2,1])
    elif(k == 4): #Distancias con 4 centroides
      distancias[i] = distancia(patrones[i,0], centroides[0,0],
patrones[i,1], centroides[0,1]), distancia(patrones[i,0],
centroides[1,0], patrones[i,1], centroides[1,1]),
distancia(patrones[i,0], centroides[2,0], patrones[i,1],
centroides[2,1]), distancia(patrones[i,0], centroides[3,0],
patrones[i,1], centroides[3,1])
    elif(k == 5): #Distancias con 5 centroides
      distancias[i] = distancia(patrones[i,0], centroides[0,0],
patrones[i,1], centroides[0,1]), distancia(patrones[i,0],
centroides[1,0], patrones[i,1], centroides[1,1]),
distancia(patrones[i,0], centroides[2,0], patrones[i,1],
centroides[2,1]), distancia(patrones[i,0], centroides[3,0],
patrones[i,1], centroides[3,1]), distancia(patrones[i,0],
centroides[4,0], patrones[i,1], centroides[4,1])
    elif(k == 6): #Distancias con 6 centroides
      distancias[i] = distancia(patrones[i,0], centroides[0,0],
patrones[i,1], centroides[0,1]), distancia(patrones[i,0],
centroides[1,0], patrones[i,1], centroides[1,1]),
distancia(patrones[i,0], centroides[2,0], patrones[i,1],
centroides[2,1]), distancia(patrones[i,0], centroides[3,0],
patrones[i,1], centroides[3,1]), distancia(patrones[i,0],
centroides[4,0], patrones[i,1], centroides[4,1]),
distancia(patrones[i,0], centroides[5,0], patrones[i,1], centroides[5,1])
    elif(k == 7): #Distancias con 7 centroides
      distancias[i] = distancia(patrones[i,0], centroides[0,0],
patrones[i,1], centroides[0,1]), distancia(patrones[i,0],
centroides[1,0], patrones[i,1], centroides[1,1]),
distancia(patrones[i,0], centroides[2,0], patrones[i,1],
centroides[2,1]), distancia(patrones[i,0], centroides[3,0],
patrones[i,1], centroides[3,1]), distancia(patrones[i,0],
centroides[4,0], patrones[i,1], centroides[4,1]),
distancia(patrones[i,0], centroides[5,0], patrones[i,1],
```

```
centroides[5,1]), distancia(patrones[i,0], centroides[6,0],
patrones[i,1], centroides[6,1])

  col = np.zeros(size) #Array para identificar la columna en donde está
el centroide más cercano y con ello poder agrupar el patrón
  """Las siguientes listas permiten almacenar las coordenadas X e Y de
cada patrón perteneciente a un centroide"""
  c1_x = list()
  c1_y = list()
  c2_x = list()
  c2_y = list()
  c3_x = list()
  c3_y = list()
  c4_x = list()
  c4_y = list()
  c5_x = list()
  c5_y = list()
  c6_x = list()
  c6_y = list()
  c7_x = list()
  c7_y = list()
  for i in range(0,size):
    col = np.where(distancias[i] == np.amin(distancias[i]))[0] #Permite
detectar la columna con la distancia más cercana de cierto patrón para
poder asociarlo a un centroide
    if(col == 0): #Se obtienen los patrones pertenecientes al centroide 1
      plt.scatter(patrones[i,0], patrones[i,1],marker='o',
color=colores[0],alpha=0.8, s=49)
      c1_x.append(patrones[i,0])
      c1_y.append(patrones[i,1])
    elif(col == 1): #Se obtienen los patrones pertenecientes al centroide
2
      plt.scatter(patrones[i,0], patrones[i,1],marker='o',
color=colores[1],alpha=0.8, s=49)
      c2_x.append(patrones[i,0])
      c2_y.append(patrones[i,1])
    elif(col == 2): #Se obtienen los patrones pertenecientes al centroide
3
      plt.scatter(patrones[i,0], patrones[i,1],marker='o',
color=colores[2],alpha=0.8, s=49)
      c3_x.append(patrones[i,0])
      c3_y.append(patrones[i,1])
    elif(col == 3): #Se obtienen los patrones pertenecientes al centroide
4
      plt.scatter(patrones[i,0], patrones[i,1],marker='o',
color=colores[3],alpha=0.8, s=49)
      c4_x.append(patrones[i,0])
      c4_y.append(patrones[i,1])
```

```
    elif(col == 4): #Se obtienen los patrones pertenecientes al centroide
5
      plt.scatter(patrones[i,0], patrones[i,1],marker='o',
color=colores[4],alpha=0.8, s=49)
      c5_x.append(patrones[i,0])
      c5_y.append(patrones[i,1])
    elif(col == 5): #Se obtienen los patrones pertenecientes al centroide
6
      plt.scatter(patrones[i,0], patrones[i,1],marker='o',
color=colores[5],alpha=0.8, s=49)
      c6_x.append(patrones[i,0])
      c6_y.append(patrones[i,1])
    elif(col == 6): #Se obtienen los patrones pertenecientes al centroide
7
      plt.scatter(patrones[i,0], patrones[i,1],marker='o',
color=colores[6],alpha=0.8, s=49)
      c7_x.append(patrones[i,0])
      c7_y.append(patrones[i,1])

  """Mediante los siguientes condicionales se obtienen las nuevas
coordenadas para cada centroide de acuerdo con su agrupación
  con los patrones, para ello se obtienen los promedios de las
coordenadas X e Y de todos los patrones pertenecientes a cierto
  centroide"""
  if(k==2):
    centroides[0,0] = mean(c1_x)
    centroides[0,1] = mean(c1_y)
    centroides[1,0] = mean(c2_x)
    centroides[1,1] = mean(c2_y)
  elif(k==3):
    centroides[0,0] = mean(c1_x)
    centroides[0,1] = mean(c1_y)
    centroides[1,0] = mean(c2_x)
    centroides[1,1] = mean(c2_y)
    centroides[2,0] = mean(c3_x)
    centroides[2,1] = mean(c3_y)
  elif(k==4):
    centroides[0,0] = mean(c1_x)
    centroides[0,1] = mean(c1_y)
    centroides[1,0] = mean(c2_x)
    centroides[1,1] = mean(c2_y)
    centroides[2,0] = mean(c3_x)
    centroides[2,1] = mean(c3_y)
    centroides[3,0] = mean(c4_x)
    centroides[3,1] = mean(c4_y)
  elif(k==5):
    centroides[0,0] = mean(c1_x)
    centroides[0,1] = mean(c1_y)
    centroides[1,0] = mean(c2_x)
```

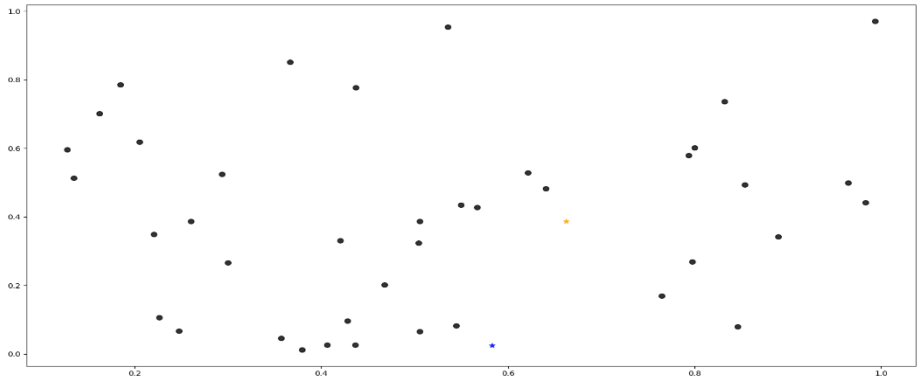
```
    centroides[1,1] = mean(c2_y)
    centroides[2,0] = mean(c3_x)
    centroides[2,1] = mean(c3_y)
    centroides[3,0] = mean(c4_x)
    centroides[3,1] = mean(c4_y)
    centroides[4,0] = mean(c5_x)
    centroides[4,1] = mean(c5_y)
  elif(k==6):
    centroides[0,0] = mean(c1_x)
    centroides[0,1] = mean(c1_y)
    centroides[1,0] = mean(c2_x)
    centroides[1,1] = mean(c2_y)
    centroides[2,0] = mean(c3_x)
    centroides[2,1] = mean(c3_y)
    centroides[3,0] = mean(c4_x)
    centroides[3,1] = mean(c4_y)
    centroides[4,0] = mean(c5_x)
    centroides[4,1] = mean(c5_y)
    centroides[5,0] = mean(c6_x)
    centroides[5,1] = mean(c6_y)
  elif(k==7):
    centroides[0,0] = mean(c1_x)
    centroides[0,1] = mean(c1_y)
    centroides[1,0] = mean(c2_x)
    centroides[1,1] = mean(c2_y)
    centroides[2,0] = mean(c3_x)
    centroides[2,1] = mean(c3_y)
    centroides[3,0] = mean(c4_x)
    centroides[3,1] = mean(c4_y)
    centroides[4,0] = mean(c5_x)
    centroides[4,1] = mean(c5_y)
    centroides[5,0] = mean(c6_x)
    centroides[5,1] = mean(c6_y)
    centroides[6,0] = mean(c7_x)
    centroides[6,1] = mean(c7_y)

  plt.show()
  plt.clf()

  """Mediante el siguiente condicional se comprueba si los centroides
siguen cambiando de posición"""

  if(np.array_equal(centroides_old, centroides)): #Se compara la posición
anterior con la actual de los centroides
    break;
```

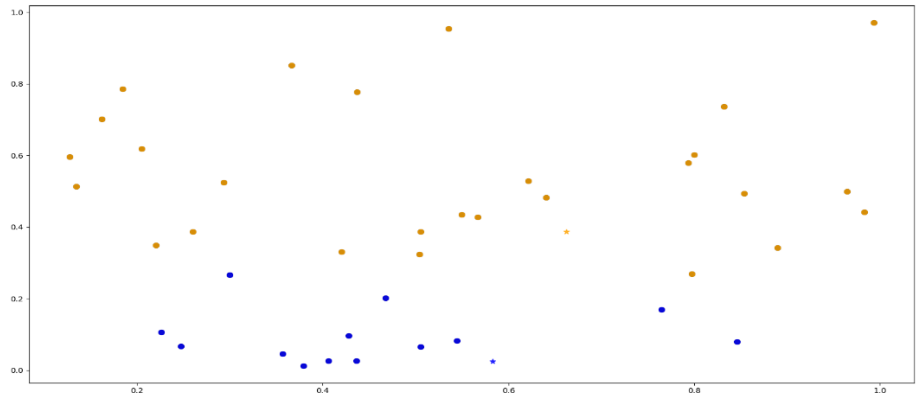
Desarrollo

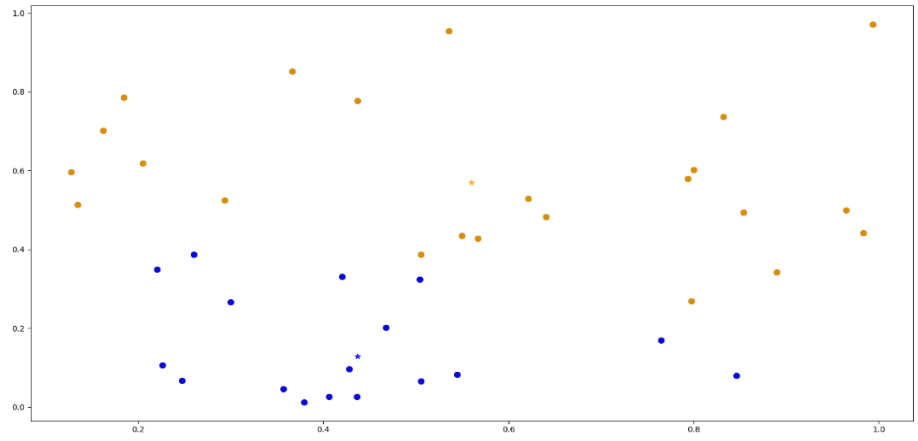
A continuación, se presentan los resultados del algoritmo empleando de 2 a 7 grupos con 40 patrones y diferentes semillas. Primero se muestra el estado inicial del gráfico con los patrones sin agrupar, después se muestran los diferentes estados del gráfico hasta llegar al final del algoritmo que ocurre cuando los centroides dejan de cambiar de posición.

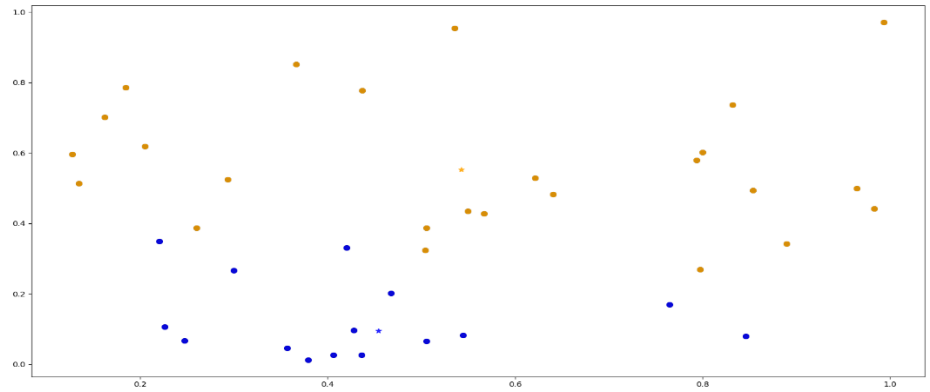
Para k = 2

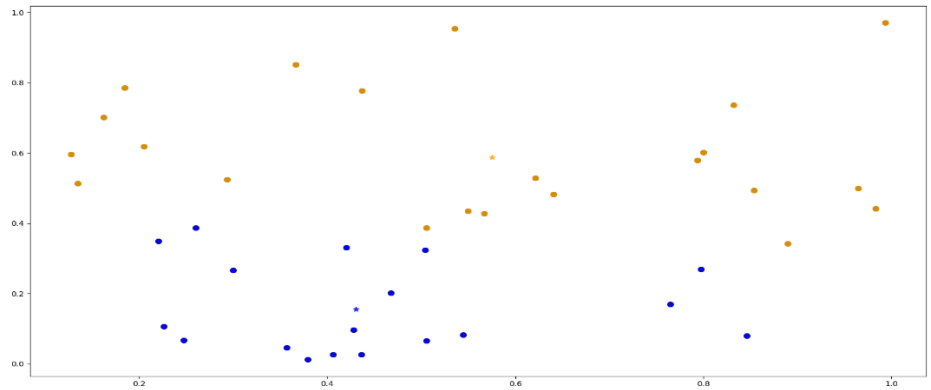
```
Ingrese el valor de la semilla: 2
Ingrese el numero de patrones: 40
Ingrese el numero de grupos: 2
```

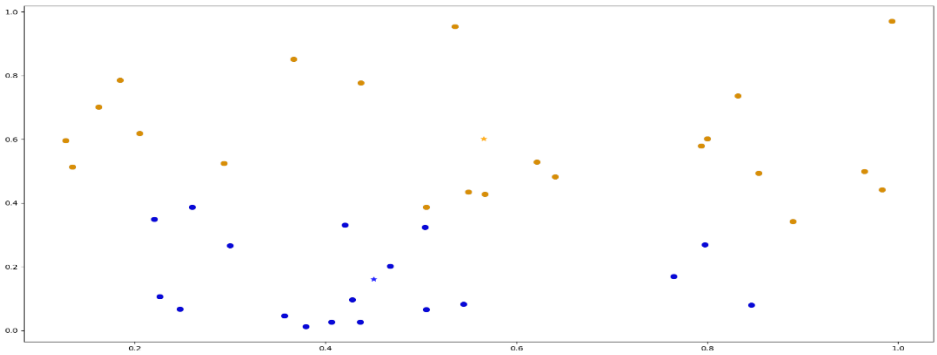






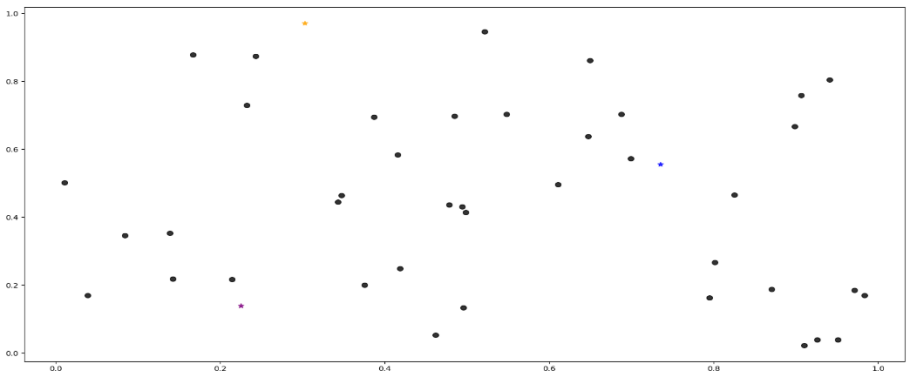


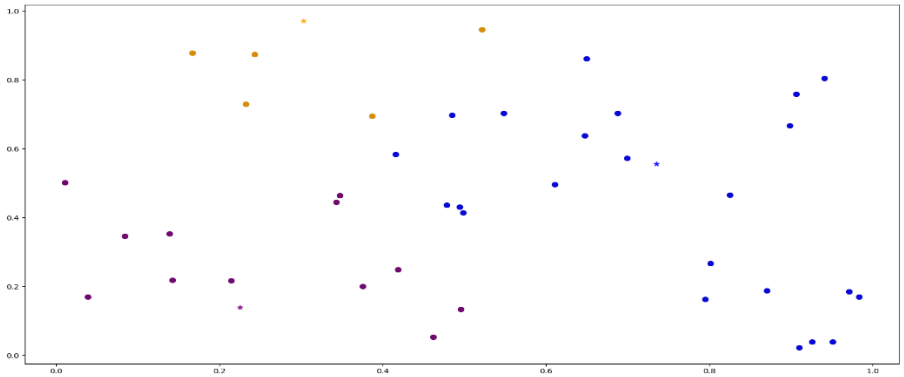
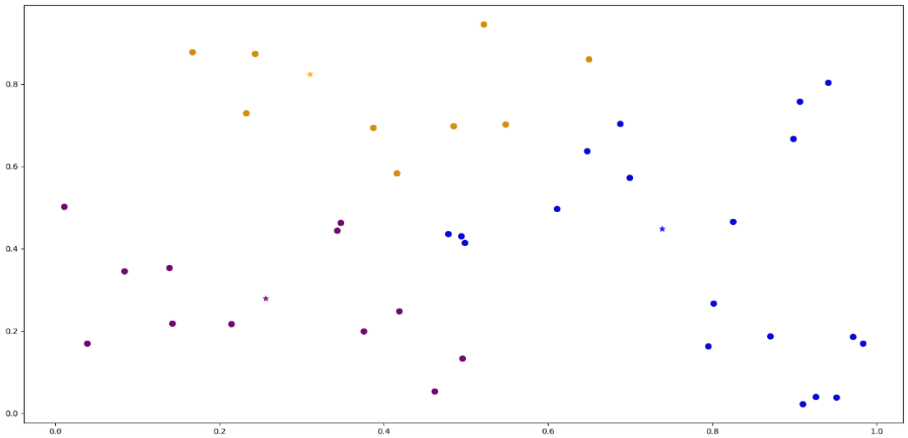
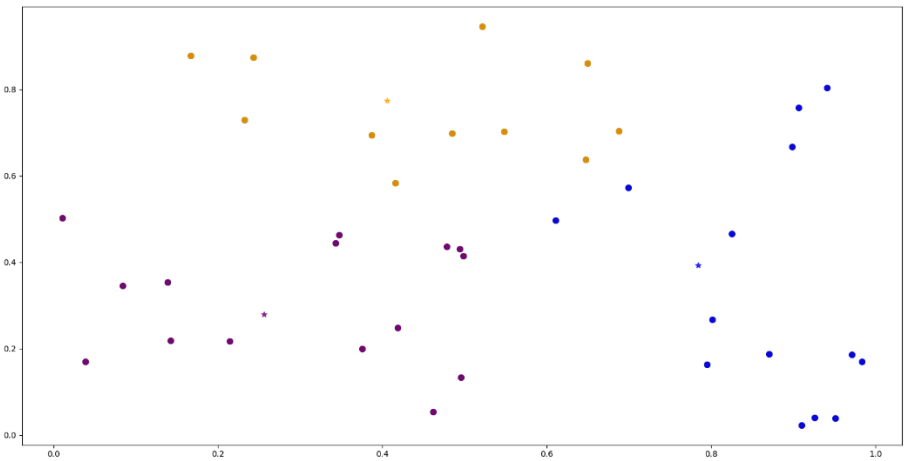
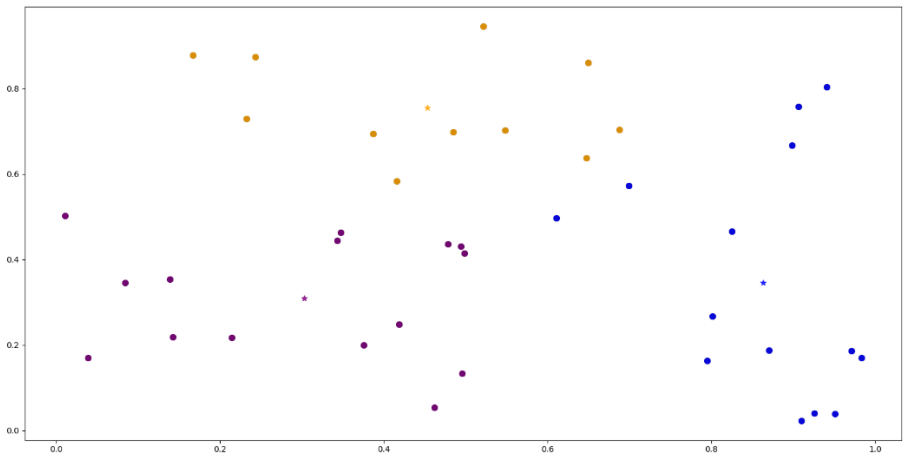




Para k = 3

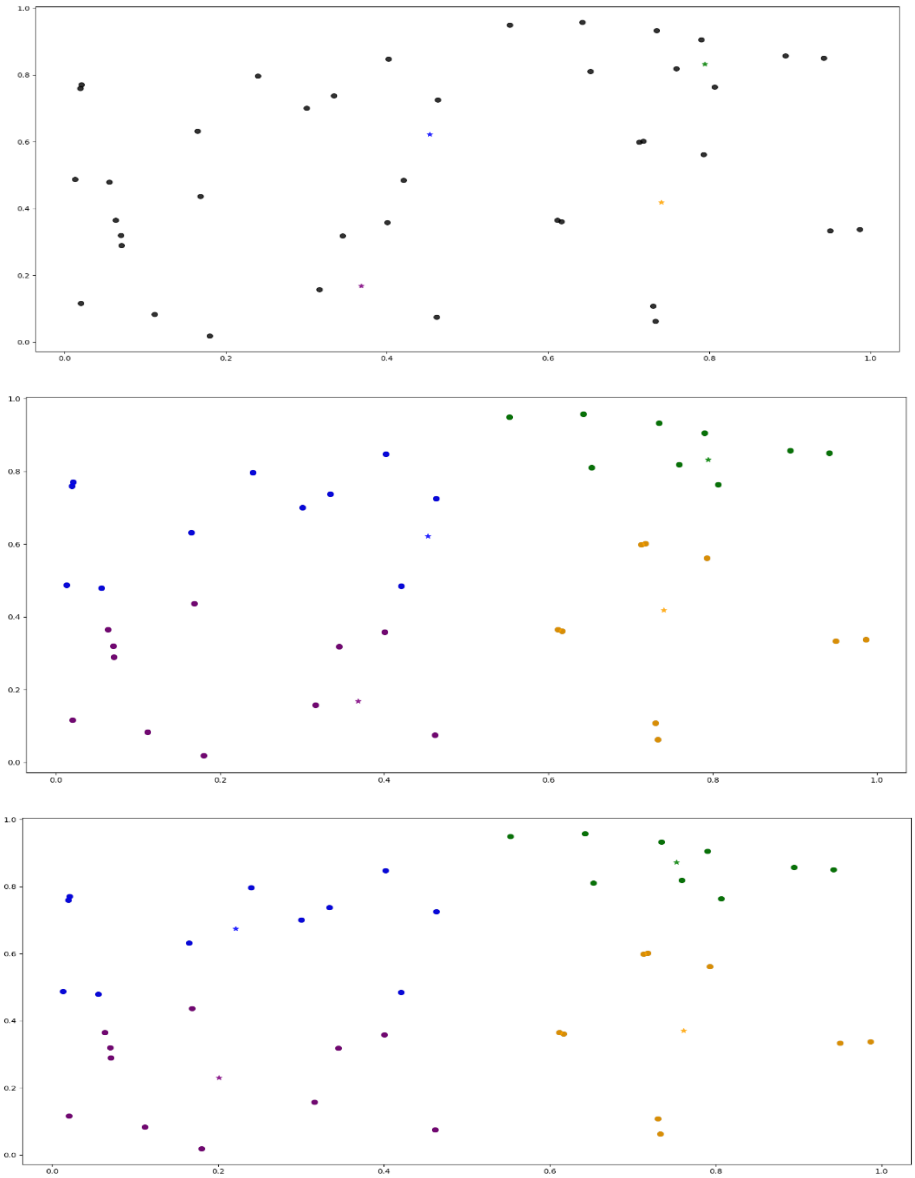
```
Ingrese el valor de la semilla: 9
Ingrese el numero de patrones: 40
Ingrese el numero de grupos: 3
```



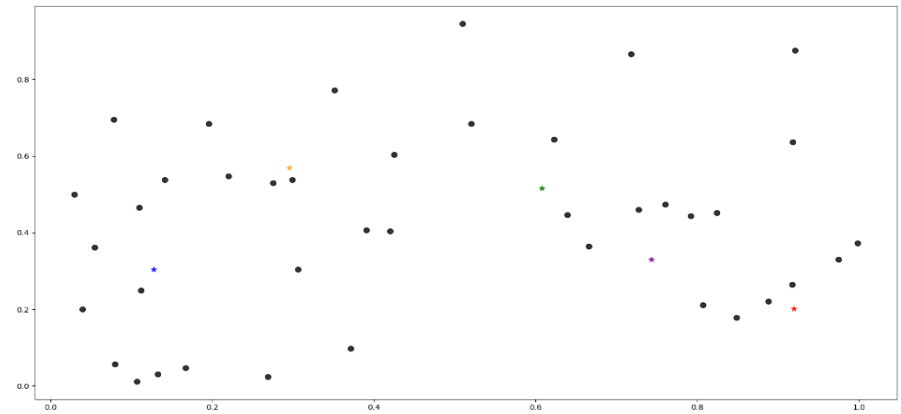
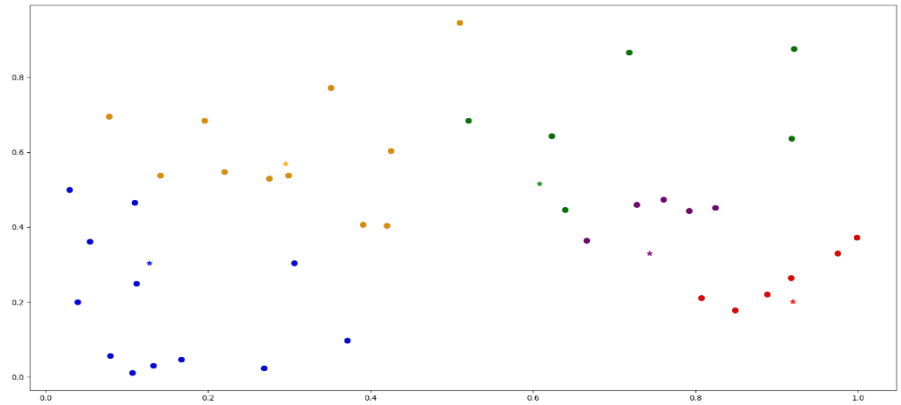
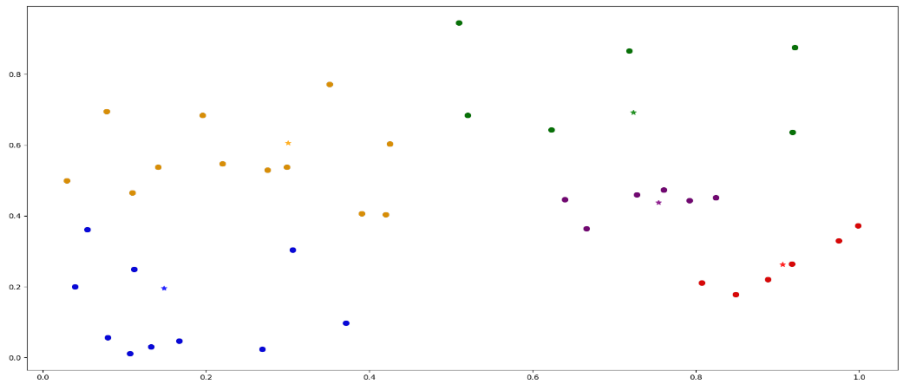
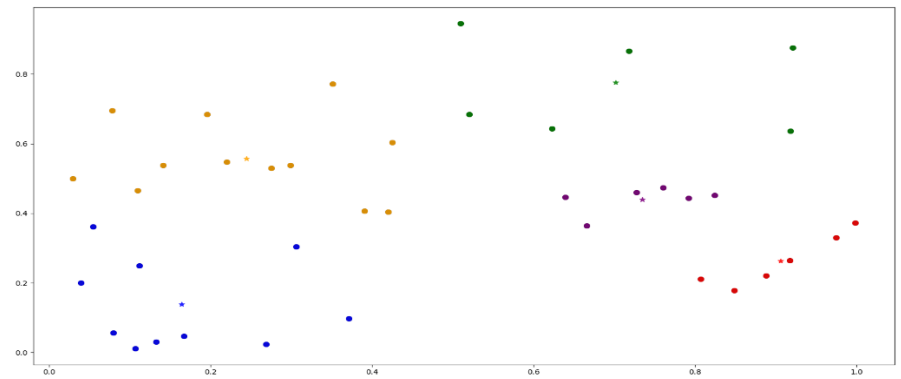
Para k = 4

```
Ingrese el valor de la semilla: 11
Ingrese el numero de patrones: 40
Ingrese el numero de grupos: 4
```



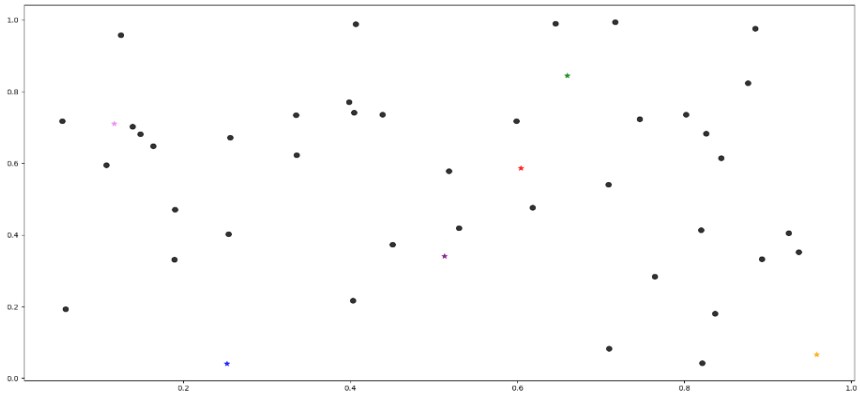
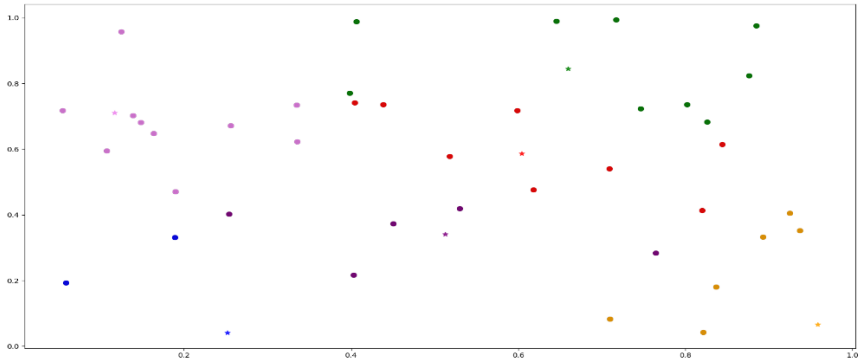
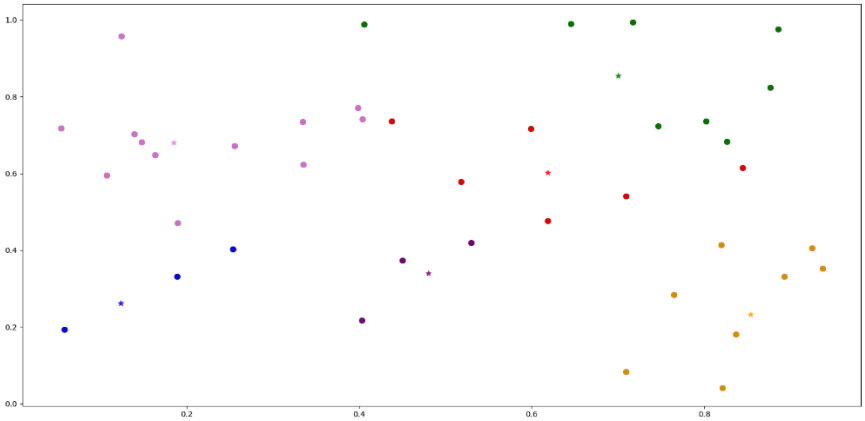
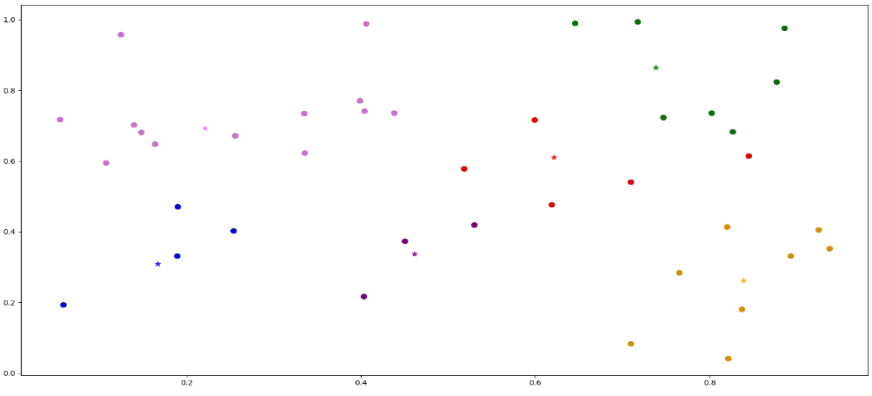
Para k = 5

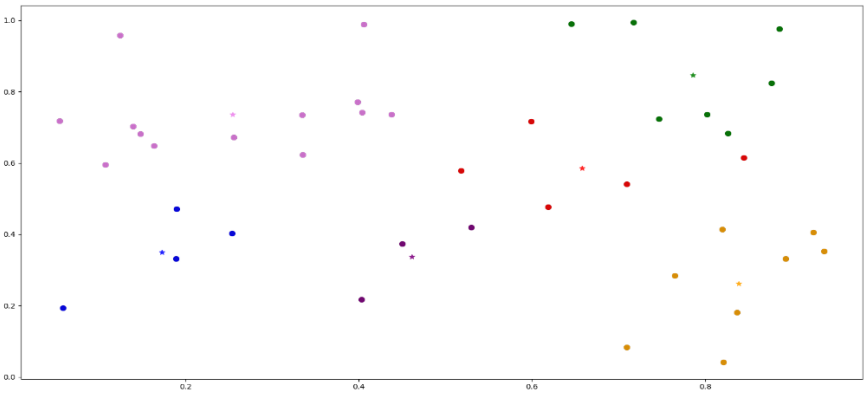
```
Ingrese el valor de la semilla: 15
Ingrese el numero de patrones: 40
Ingrese el numero de grupos: 5
```

Para k = 6

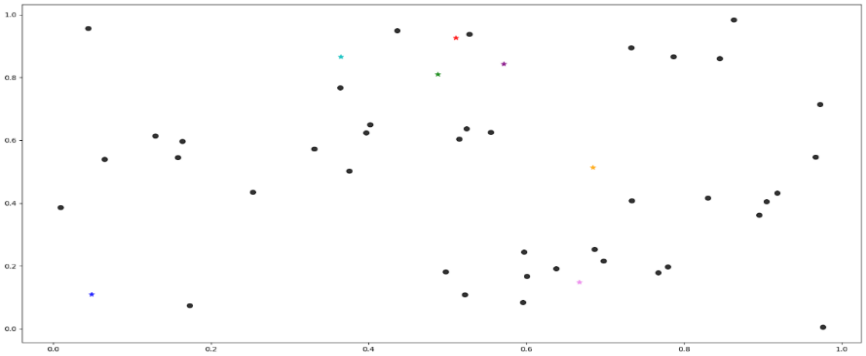
```
Ingrese el valor de la semilla: 6
Ingrese el numero de patrones: 40
Ingrese el numero de grupos: 6
```

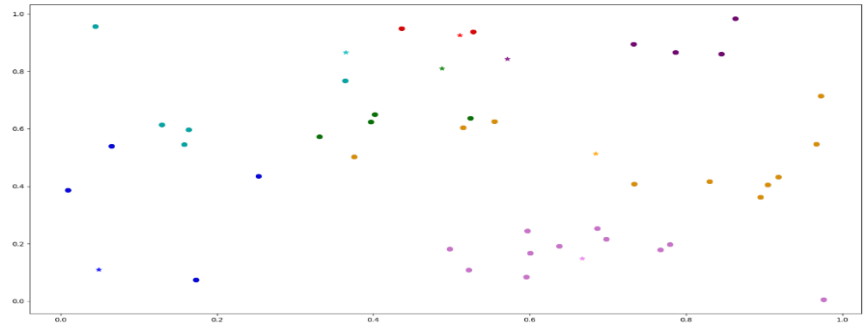





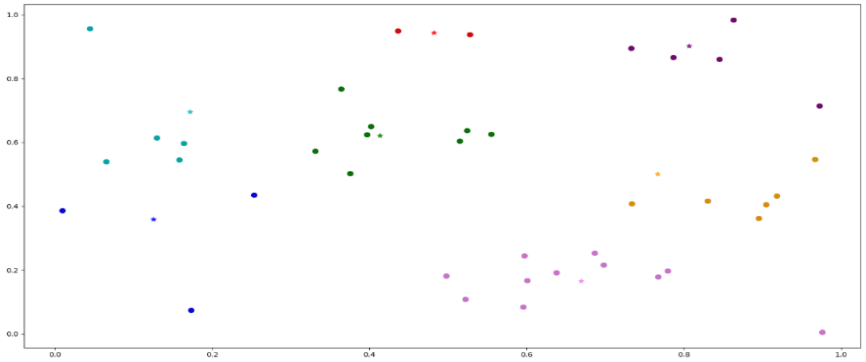


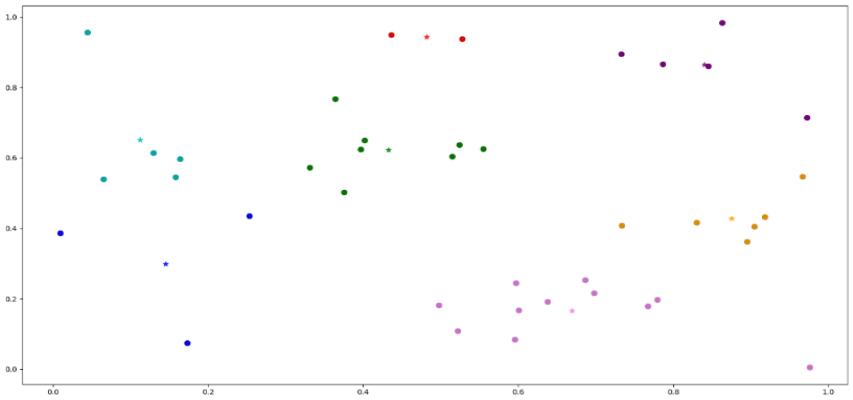
Para k = 7

```
Ingrese el valor de la semilla: 4
Ingrese el numero de patrones: 40
Ingrese el numero de 4: 7
```









Conclusiones

Mediante esta práctica pudimos comprender el funcionamiento del algoritmo k-medias el cual es muy útil cuando queremos segmentar datos dispersos en un cierto número de grupos. Consideramos que este es un algoritmo bastante fácil de entender e implementar, pues no requiere el empleo de muchos pasos u operaciones. Lo que más nos costó fue definir el criterio para finalizar el algoritmo, pues al principio pensamos que siempre iban a estar cambiando los centroides, por lo que consideramos definir cierto número de iteraciones para finalizarlo, pero al implementarlo nos dimos cuenta de que siempre llegaba un cierto numero de iteraciones en el que los centroides dejaban de cambiar, por lo que sólo había que detectar esa iteración para finalizar el algoritmo. Además, notamos que por lo general entre menor sea el número de grupos, le toma más tiempo al algoritmo agrupar a todos los patrones debido a los constantes movimientos de los centroides. Sin duda conocer este algoritmo es de mucha utilidad para el análisis de datos.

Referencias

- L. Ramirez, (2023). “Algoritmo k-means: ¿Qué es y cómo funciona?”. [Internet]. Disponible en: https://www.iebschool.com/blog/algoritmo-k-means-que-es-y-como-funciona-big-data/
- IBM, (2022). “Análisis de clústeres de K-medias”. [Internet]. Disponible en: https://www.ibm.com/docs/es/spss-statistics/saas?topic=features-k-means-cluster-analysis
- F. Sanz, (s.f.) “Algoritmo K-Means Clustering – aplicaciones y desventajas”. [Internet]. Disponible en: https://www.themachinelearners.com/k-means/