

```

1 import numpy as np
2 from matplotlib import pyplot as plt
3 plt.rcParams['figure.figsize'] = (6, 4)

```

```

1 np.linalg.norm(np.array([8,10]) - np.array([3,10.5]))

```

```

⇒ np.float64(5.024937810560445)

```

```

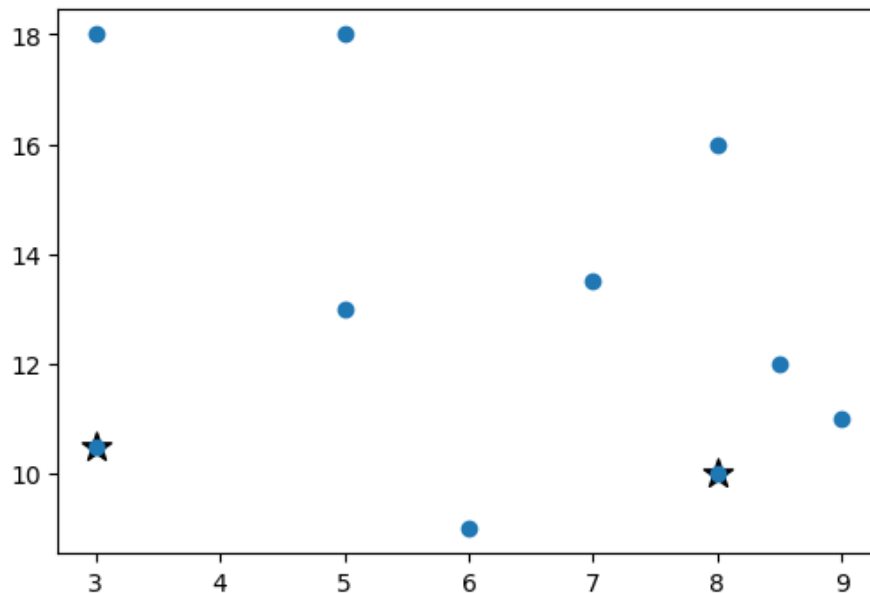
1 data = np.array([[8,10],[3,10.5],[7,13.5],[5,18],[5,13],[6,9],[9,11],[3,18],[8.5,12],[8,16]])
2 #C = np.array([ [8,10],[3,10.5],[7,13.5] ])
3 C = np.array([ [8,10],[3,10.5] ])
4 # Gráfica
5 plt.scatter(C[:,0], C[:,1],marker='*',s=150,c='k') # Centroides
6 plt.scatter(data[:,0],data[:,1]) # Datos

```

```

⇒ <matplotlib.collections.PathCollection at 0x78fd08c49b90>

```



```

1 def dist(a, b):
2     return np.linalg.norm(a[:,np.newaxis] - b, axis=2)

```

```

1 x = np.array([[8,10]])
2 y = np.array([3,10.5])
3 dist(x, y)

```

```

⇒ array([[5.02493781]])

```

```

1 x.shape, x[:,np.newaxis].shape

```

```

⇒ ((1, 2), (1, 1, 2))

```

```

1 # Asignar puntos a cada grupo
2 distancias = dist(data, C)
3 grupos = np.argmin(distancias, axis=1)
4 print(grupos, distancias)

```

```

⇒ [0 1 0 1 1 0 0 1 0 0] [[0.          5.02493781]
 [5.02493781 0.          ]
 [3.64005494 5.          ]
 [8.54400375 7.76208735]
 [4.24264069 3.20156212]
 [2.23606798 3.35410197]
 [1.41421356 6.02079729]
 [9.43398113 7.5          ]
 [2.06155281 5.70087713]
 [6.          7.43303437]]

```

```

1 # Actualizar centroides
2 # list comprehension [ oper(e) for e in list /if cond/ ]
3 for i in range(len(C)):
4     puntos=[ data[j] for j in range(len(data)) if grupos[j]==i ]
5     #print(i,np.array(puntos))
6     C[i] = np.mean(puntos, axis=0)
7 print(C)

```

```

⇒ [[ 7.75          11.91666667]
 [ 4.           14.875        ]]

```

```

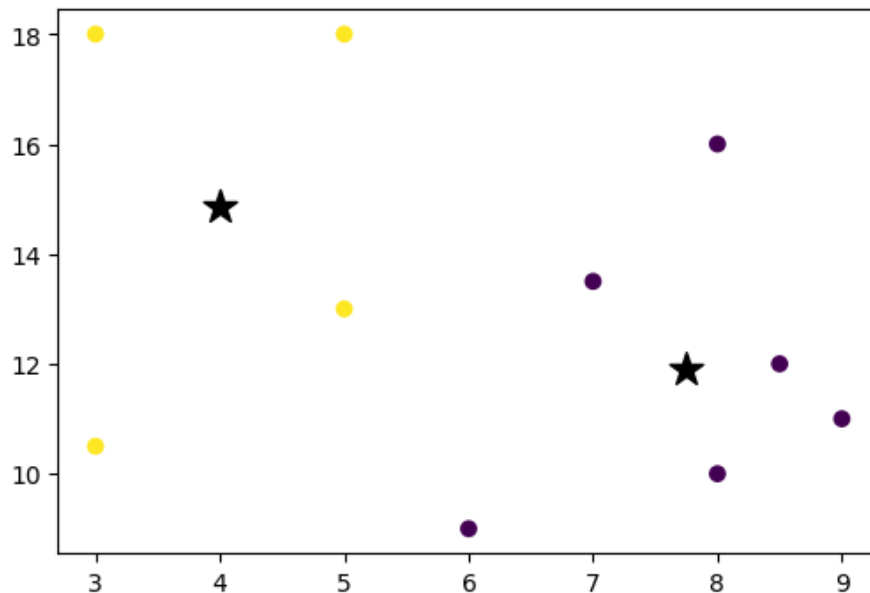
1 # Gráfica
2 plt.scatter(C[:,0], C[:,1], marker='*', s=200, c='k')
3 plt.scatter(data[:,0], data[:,1], c=grupos)

```

```

⇒ <matplotlib.collections.PathCollection at 0x78fd0831c950>

```



1

```

1 ''' scikit-learn '''
2 import numpy as np
3 import pandas as pd
4 from matplotlib import pyplot as plt
5 plt.rcParams['figure.figsize'] = (6,4)
6 plt.style.use('ggplot')

```

```

1 # Conjunto de datos xclara
2 data = pd.read_csv('https://bit.ly/3BxsIDR')
3 print(data.shape)
4 data.tail()

```

→ (3000, 3)

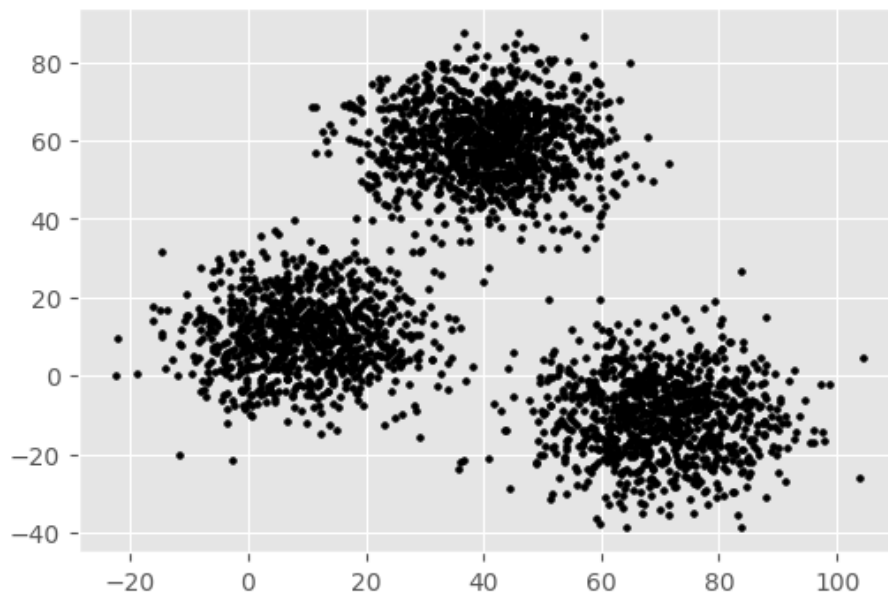
	rownames	V1	V2	
2995	2996	85.65280	-6.461061	
2996	2997	82.77088	-2.373299	
2997	2998	64.46532	-10.501360	
2998	2999	90.72282	-12.255840	
2999	3000	64.87976	-24.877310	

```

1 # Gráfica
2 x1 = data['V1'].values
3 x2 = data.V2.values
4 X = np.array(list(zip(x1,x2)))
5 plt.scatter(x1, x2, c='black', s=7)

```

→ <matplotlib.collections.PathCollection at 0x78fcd9a8eb50>



```

1 from sklearn.cluster import KMeans
2 # Inicializar con número de grupos
3 km = KMeans(n_clusters=3)
4 # Ajuste: ejecutar el algoritmo de aprendizaje -> los centroides de cada clase
5 km = km.fit(X)
6 # Etiquetas de cada clase: obtener la clase a la que pertenece cada punto
7 y = km.predict(X)
8 # Centroides
9 C_ = km.cluster_centers_
10 print(C_)

```

→

```

[[ 69.92418447 -10.11964119]
 [ 40.68362784  59.71589274]
 [  9.4780459   10.686052  ]]

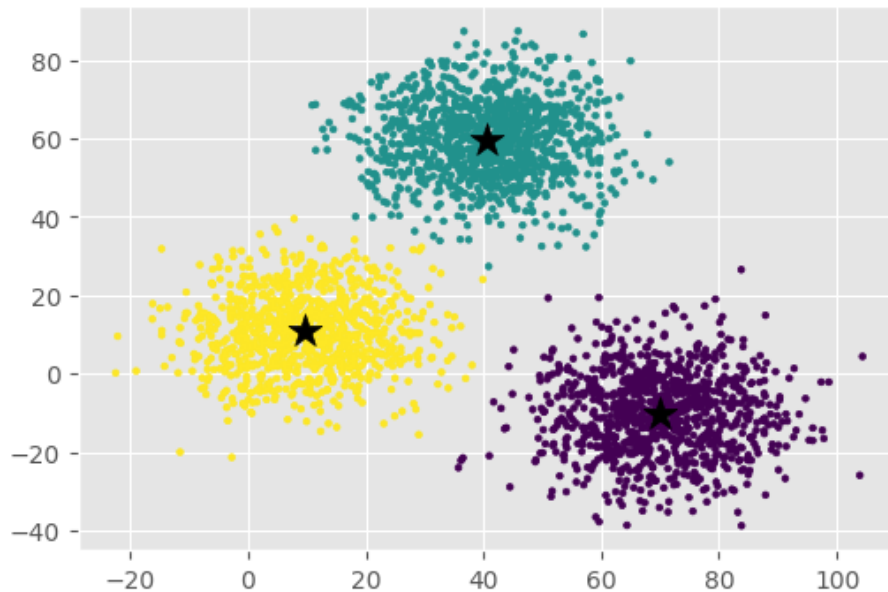
```

```

1 plt.scatter(X[:, 0], X[:, 1], c=y, s=7)
2 plt.scatter(C[:, 0], C[:, 1], marker='*', s=200, c='k')

```

↔ <matplotlib.collections.PathCollection at 0x78fcc1710590>



```

1 # Ejemplo 3D
2 import numpy as np
3 import matplotlib.pyplot as plt
4 from sklearn.cluster import KMeans
5 from sklearn.datasets import make_blobs
6 plt.rcParams['figure.figsize'] = (9, 4)
7 # 4 nubes de puntos de 3D
8 X, y = make_blobs(n_samples=800, n_features=3, centers=4, random_state=42)

```

```

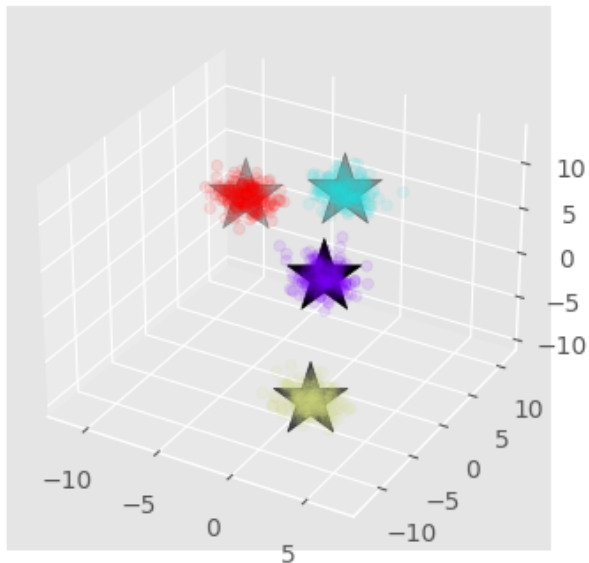
1 # Initializing KMeans
2 kmeans = KMeans(n_clusters=4, random_state=42)
3 # Fitting with inputs
4 kmeans = kmeans.fit(X)
5 # Predicting the clusters
6 y = kmeans.predict(X)
7 # Getting the cluster centers
8 C = kmeans.cluster_centers_

```

```

1 fig, ax = plt.subplots(subplot_kw={"projection": "3d"})
2 ax.scatter(X[:, 0], X[:, 1], X[:, 2], c=y, alpha=0.1, cmap='rainbow')
3 ax.scatter(C[:, 0], C[:, 1], C[:, 2], marker='*', c='k', s=1000)
4 plt.show()

```

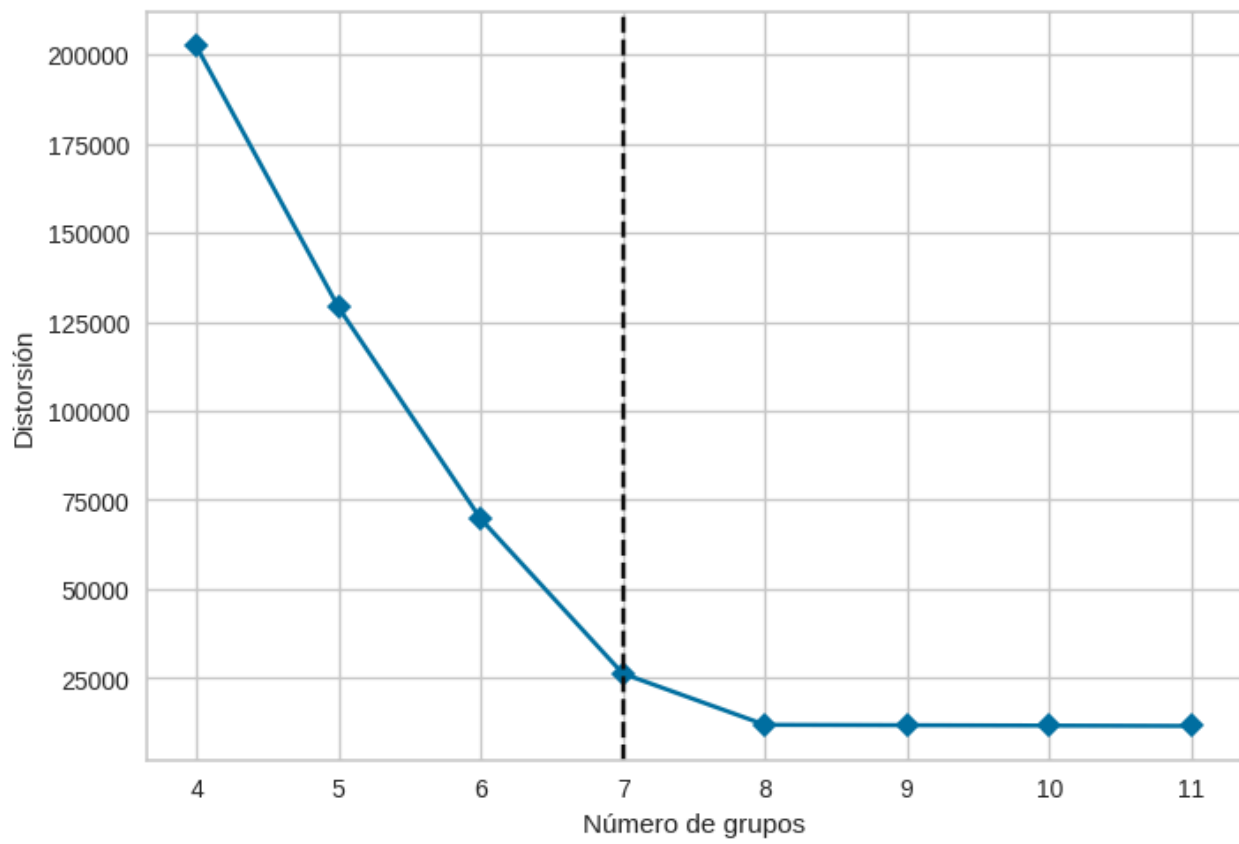


1

```
1 # ¿Cómo saber el valor inicial para K? => Gráfica del codo
2 # Importamos bibliotecas
3 import matplotlib.pyplot as plt
4 import pandas as pd
5 from sklearn import datasets
6 import seaborn as sns; sns.set() # Para el estilo de los gráficos
```

```
1 #!pip install yellowbrick
```

```
1 # Elección de k con la gráfica de codo KElbowVisualizer
2 # https://www.scikit-yb.org/en/latest/api/cluster/elbow.html
3 # Por omisión la métrica es "distortion", que es la suma de las
4 # distancias cuadráticas de cada punto con el centro de su grupo
5 from matplotlib import pyplot as plt
6 from sklearn.cluster import KMeans
7 from sklearn.datasets import make_blobs
8 from yellowbrick.cluster import KElbowVisualizer
9 # Generamos 8 nubes de puntos de dimensión 12
10 X, y = make_blobs(n_samples=1000, n_features=12, centers=8, random_state=42)
11
12 vis = KElbowVisualizer(KMeans(random_state=42), k=(4,12), timings=False)
13 vis.fit(X)
14 plt.xlabel('Número de grupos')
15 plt.ylabel('Distorsión')
16 plt.show()
```



```
1 # KMeans in depth
2 # https://github.com/jakevdp/PythonDataScienceHandbook/blob/master/notebooks/05.11-K-Means.ipynb
```

1

1

1

1

1

1

1

+ Código

+ Texto

1

1

1

1

```

1 # BIBLIOTECA
2 #!pip install kmodes

1 # bibliotecas
2 import pandas as pd
3 import numpy as np
4 import matplotlib.pyplot as plt
5 from kmodes.kmodes import KModes

1 data = np.array(['x', 'y', 'z'],
2                 ['y', 'z', 'x'],
3                 ['z', 'x', 'x'],
4                 ['y', 'z', 'z'],
5                 ['x', 'z', 'y'],
6                 ['z', 'y', 'x'],
7                 ['x', 'x', 'y'],
8                 ['z', 'y', 'x']))
9 # modelo con 2 grupos
10 # n_init Number of time the k-modes algorithm will be run with different
11 #      centroid seeds. The final results will be the best output of
12 #      n_init consecutive runs in terms of cost
13 km = KModes(n_clusters=2,init='random',n_init=5,verbose=False)
14 grupos = km.fit_predict(data)
15 grupos

```

```

⇒ array([0, 0, 1, 0, 0, 1, 0, 1], dtype=uint16)

```

```

1 km.cluster_centroids_

⇒ array(['x', 'z', 'y'],
        ['z', 'y', 'x']), dtype='<U1')

```

```

1

```

```

1 # bibliotecas
2 import pandas as pd
3 import numpy as np
4 from kmodes.kmodes import KModes
5 import matplotlib.pyplot as plt

```

```

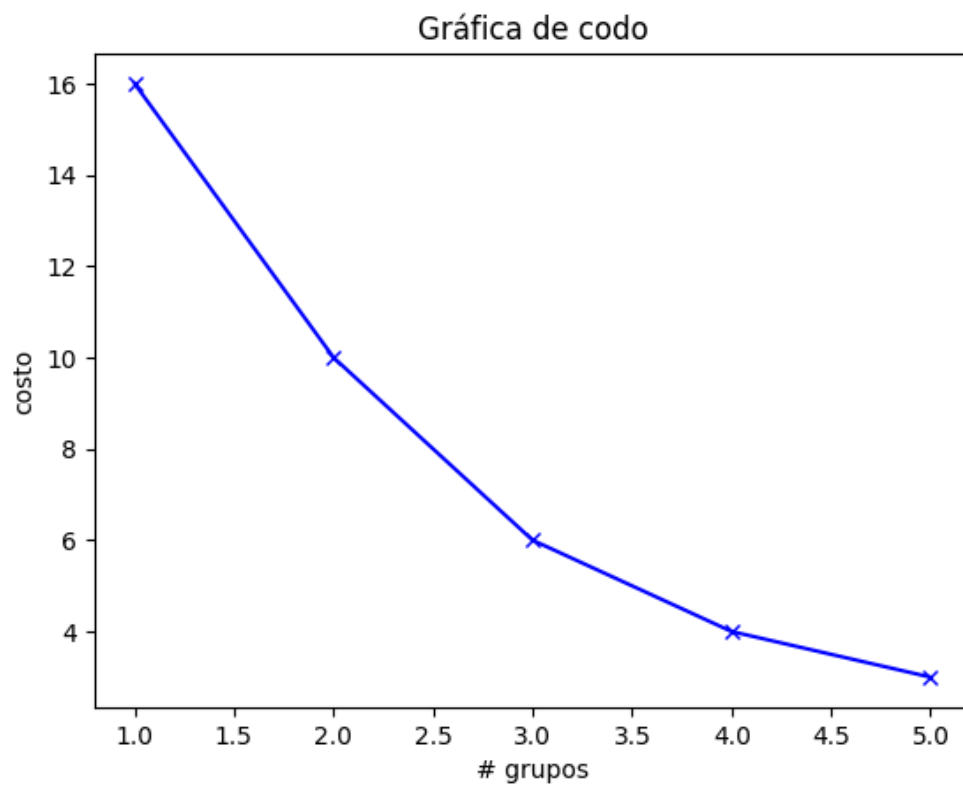
1 # datos
2 col_cabello = np.array(['rubio', 'castaño', 'pelirrojo', 'negro', 'castaño', 'negro', 'pelirrojo'])
3 col_ojos = np.array(['azul', 'gris', 'verde', 'café', 'azul', 'gris', 'azul', 'café'])
4 tipo_cabello = np.array(['lacio', 'chino', 'ondulado', 'ondulado', 'chino', 'chino', 'ondulado'])
5 personas = ['P1', 'P2', 'P3', 'P4', 'P5', 'P6', 'P7', 'P8']
6 data = pd.DataFrame({'person':personas, 'col_cabello':col_cabello, 'col_ojos':col_ojos, 'tipo_cabello':tipo_cabello})
7 data = data.set_index('person')
8 data

```



	col_cabello	col_ojos	tipo_cabello
person			
P1	rubio	azul	lacio
P2	castaño	gris	chino
P3	pelirrojo	verde	ondulado
P4	negro	café	ondulado
P5	castaño	azul	chino
P6	negro	gris	chino
P7	pelirrojo	azul	ondulado
P8	rubio	café	lacio

```
1 # Gráfica del codo
2 cost = []
3 K = range(1,6)
4 for num_clusters in list(K):
5     kmode = KModes(n_clusters=num_clusters,init='random',n_init=5,verbose=False)
6     kmode.fit_predict(data)
7     cost.append(kmode.cost_)
8
9 plt.plot(K, cost, 'bx-')
10 plt.xlabel('# grupos')
11 plt.ylabel('costo')
12 plt.title('Gráfica de codo')
13 plt.show()
```




```
1 # modelo con 3 grupos
2 kmode = KModes(n_clusters=3, init="random", n_init=5, verbose=False)
3 grupos = kmode.fit_predict(data)
4 grupos
```

```
➔ array([0, 2, 1, 0, 2, 2, 1, 0], dtype=uint16)
```

```
1 kmode.cluster_centroids_
```

```
➔ array([[ 'rubio', 'café', 'lacio'],
        [ 'pelirrojo', 'azul', 'ondulado'],
        [ 'castaño', 'gris', 'chino']], dtype='<U9')
```

```
1 data.insert(0, "grupo", grupos)
2 data
```

```
➔
```

	grupo	col_cabello	col_ojos	tipo_cabello
person				
P1	0	rubio	azul	lacio
P2	2	castaño	gris	chino
P3	1	pelirrojo	verde	ondulado
P4	0	negro	café	ondulado
P5	2	castaño	azul	chino
P6	2	negro	gris	chino
P7	1	pelirrojo	azul	ondulado
P8	0	rubio	café	lacio

1

1

1

1

1

1

1

1

1

```

1 # Jerárquico
2 # Ejemplo 1
3 # bibliotecas
4 import pandas as pd
5 import numpy as np
6 import matplotlib.pyplot as plt

1 # datos
2 #https://archive.ics.uci.edu/ml/machine-learning-databases/00292/Wholesale%20customers%20dat
3 url = 'https://bit.ly/2C0HM14'
4 data = pd.read_csv(url)
5 data.head(2)

```



	Channel	Region	Fresh	Milk	Grocery	Frozen	Detergents_Paper	Delicassen
0	2	3	12669	9656	7561	214	2674	1338
1	2	3	7057	9810	9568	1762	3293	1776

```

1 # Preprocesamiento : Escalar los valores de las columnas
2 from sklearn.preprocessing import normalize
3 data_scaled = normalize(data)
4 data_scaled = pd.DataFrame(data_scaled, columns=data.columns)
5 data_scaled.head(2)

```



	Channel	Region	Fresh	Milk	Grocery	Frozen	Detergents_Paper	Delicassen
0	0.000112	0.000168	0.708333	0.539874	0.422741	0.011965	0.149505	0.074809
1	0.000125	0.000188	0.442198	0.614704	0.599540	0.110409	0.206342	0.111286

```

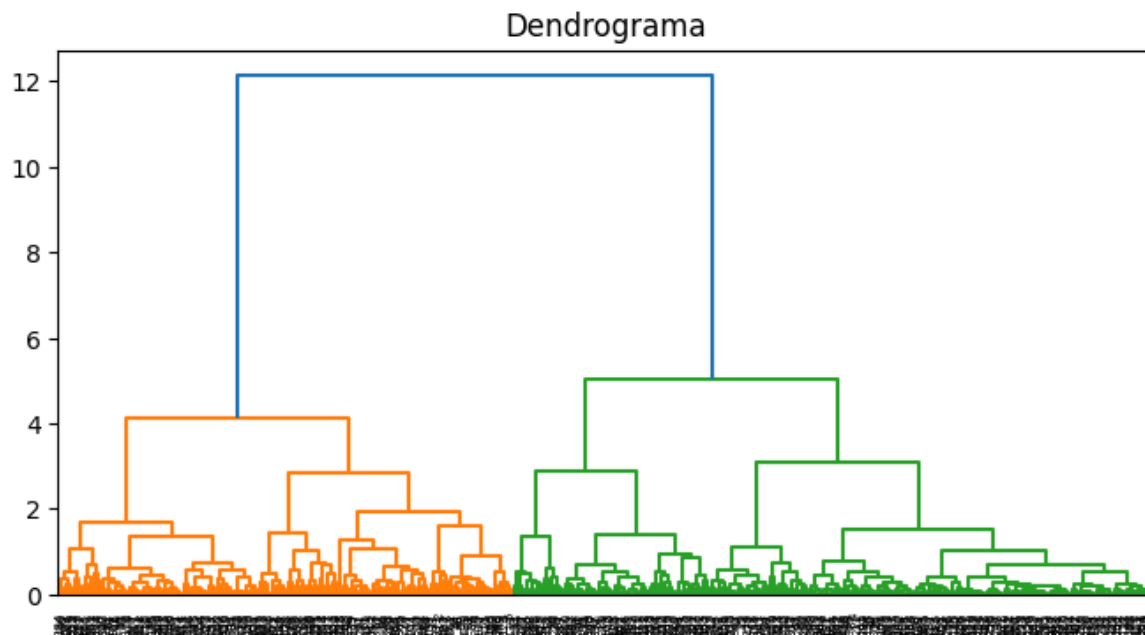
1 #data_scaled.min(), data_scaled.max()

```

```

1 import scipy.cluster.hierarchy as sch
2 # Dendrograma
3 plt.figure(figsize=(8,4))
4 plt.title("Dendrograma")
5 dend = sch.dendrogram(sch.linkage(data_scaled, method='ward'))

```

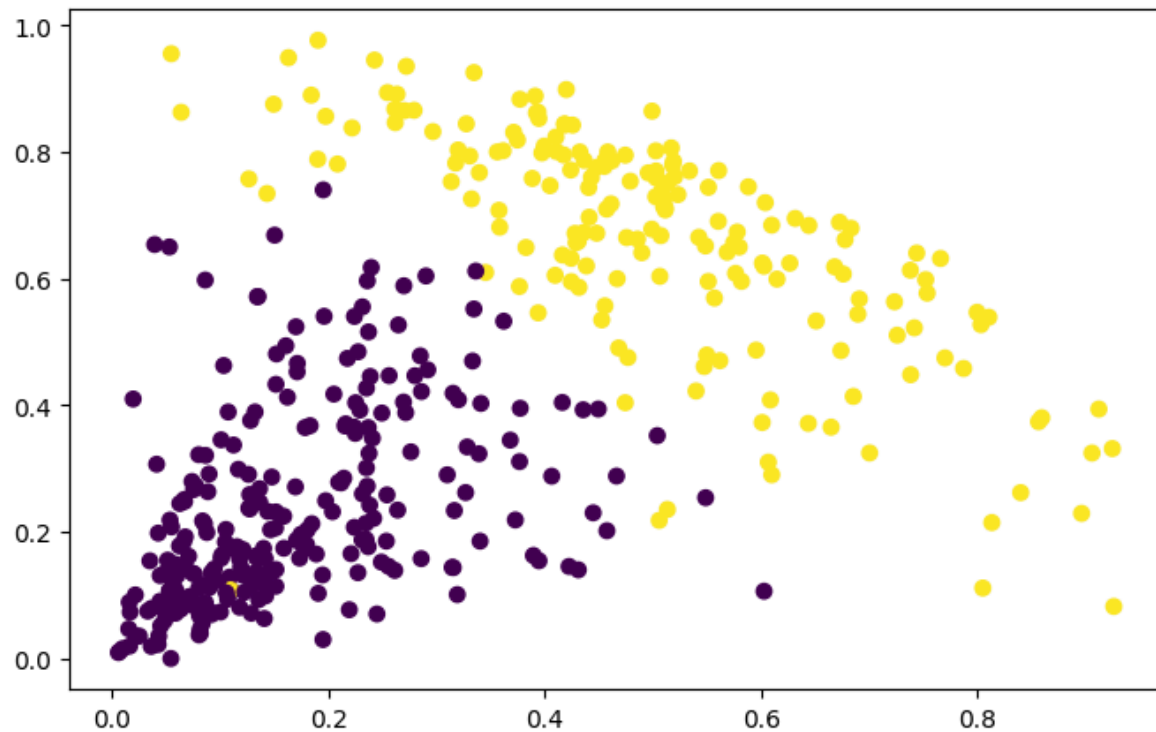


```
1 from sklearn.cluster import AgglomerativeClustering
2 cluster = AgglomerativeClustering(n_clusters=2, metric='euclidean', linkage='ward')
3 cluster.fit_predict(data_scaled)
```

```
array([1, 1, 1, 0, 0, 1, 0, 1, 1, 1, 1, 0, 0, 0, 0, 0, 1, 1, 0, 0, 0, 0,
       0, 1, 0, 0, 0, 0, 1, 0, 0, 1, 0, 0, 1, 1, 0, 1, 1, 0, 0, 0, 1, 1,
       1, 1, 1, 1, 1, 1, 0, 1, 0, 1, 0, 1, 1, 1, 0, 1, 0, 1, 1, 1, 0, 1,
       1, 0, 1, 0, 0, 0, 0, 0, 0, 1, 0, 0, 1, 0, 1, 0, 1, 1, 0, 0, 1, 1, 0,
       0, 0, 0, 0, 1, 0, 1, 1, 1, 0, 0, 0, 1, 1, 1, 0, 0, 0, 1, 1, 1, 1,
       0, 1, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 1, 0, 0, 0,
       0, 0, 0, 0, 1, 1, 0, 1, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 1, 0, 1,
       0, 1, 1, 0, 1, 1, 1, 0, 0, 1, 1, 1, 1, 1, 0, 0, 1, 1, 1, 1, 1, 1,
       0, 0, 0, 1, 0, 0, 1, 1, 1, 0, 0, 1, 1, 1, 0, 0, 0, 1, 0, 0, 0, 1,
       0, 0, 1, 1, 0, 1, 1, 1, 0, 1, 1, 1, 0, 1, 0, 1, 1, 1, 1, 0, 1, 0,
       0, 1, 0, 0, 0, 0, 0, 0, 1, 0, 0, 1, 0, 1, 0, 1, 0, 0, 0, 0, 0, 0,
       0, 0, 1, 1, 0, 0, 0, 0, 0, 1, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 1,
       1, 1, 1, 0, 1, 0, 0, 1, 1, 0, 1, 1, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 1, 0, 1, 1, 0, 0, 0, 0, 1, 1, 0, 1, 1, 1, 1, 1, 1, 0,
       0, 1, 0, 0, 1, 0, 0, 1, 0, 1, 0, 1, 1, 0, 0, 0, 0, 0, 0, 1, 0, 0,
       0, 1, 0, 1, 0, 0, 0, 0, 0, 0, 1, 1, 1, 1, 0, 1, 1, 0, 1, 1, 0, 1,
       1, 1, 0, 1, 0, 1, 1, 1, 0, 0, 1, 0, 0, 1, 0, 0, 0, 0, 0, 0, 1, 0,
       0, 0, 1, 0, 1, 1, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 1, 1, 0, 0, 0,
       1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 1, 1, 0, 0, 1, 1, 1,
       1, 0, 1, 0, 0, 0, 0, 0, 1, 0, 1, 0, 1, 0, 0, 1, 0, 0, 1, 0, 0, 1])
```

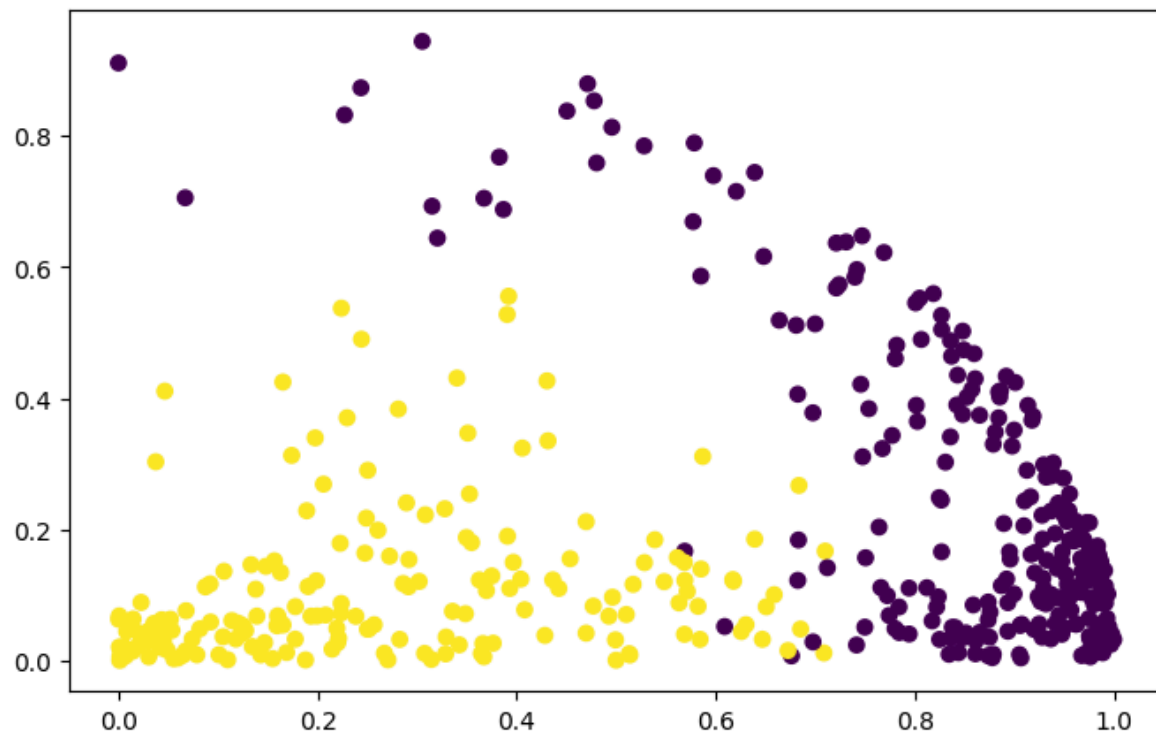
```
1 #Visualización de grupos
2 plt.figure(figsize=(8, 5))
3 plt.scatter(data_scaled.Milk,data_scaled.Grocery,c=cluster.labels_)
```

⇒ <matplotlib.collections.PathCollection at 0x7fd355f82490>



```
1 plt.figure(figsize=(8, 5))  
2 plt.scatter(data_scaled.Fresh, data_scaled.Frozen, c=cluster.labels_)
```

⇒ <matplotlib.collections.PathCollection at 0x7fd355aadc90>



1

```
1 # Ejemplo 2  
2 # Bibliotecas
```

```

3 import numpy as np
4 import matplotlib.pyplot as plt
5 import pandas as pd

1 # Dataset
2 dataset = pd.read_csv('Mall_Customers.csv')
3 dataset.head(3)

```



	CustomerID	Genre	Age	Annual Income (k\$)	Spending Score (1-100)
0	1	Male	19	15	39
1	2	Male	21	15	81
2	3	Female	20	16	6

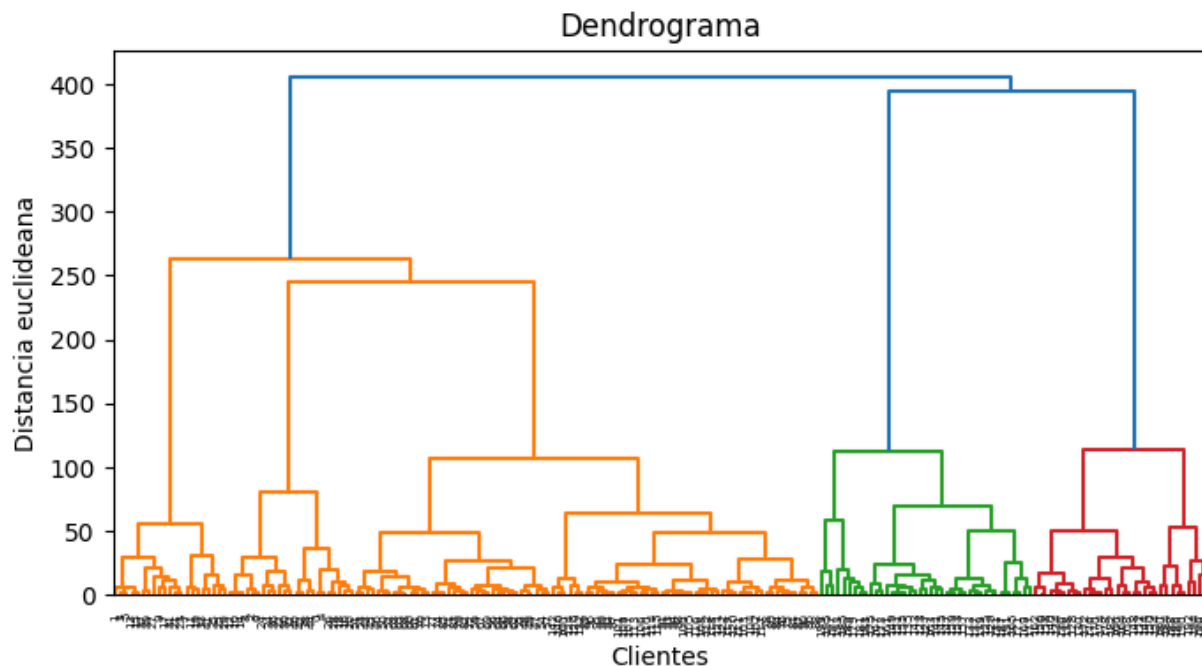
```

1 # Trabajamos solamente con Annual Income (k$)   Spending Score (1-100)
2 X = dataset.iloc[:, [3, 4]].values
3 print(X.shape)
4 # Dendrograma para determinar el número óptimo de grupos
5 import scipy.cluster.hierarchy as sch
6 plt.figure(figsize=(8,4))
7 dend = sch.dendrogram(sch.linkage(X, method='ward'))
8 plt.title('Dendrograma')
9 plt.xlabel('Clientes')
10 plt.ylabel('Distancia euclideana')
11 plt.show()

```



(200, 2)



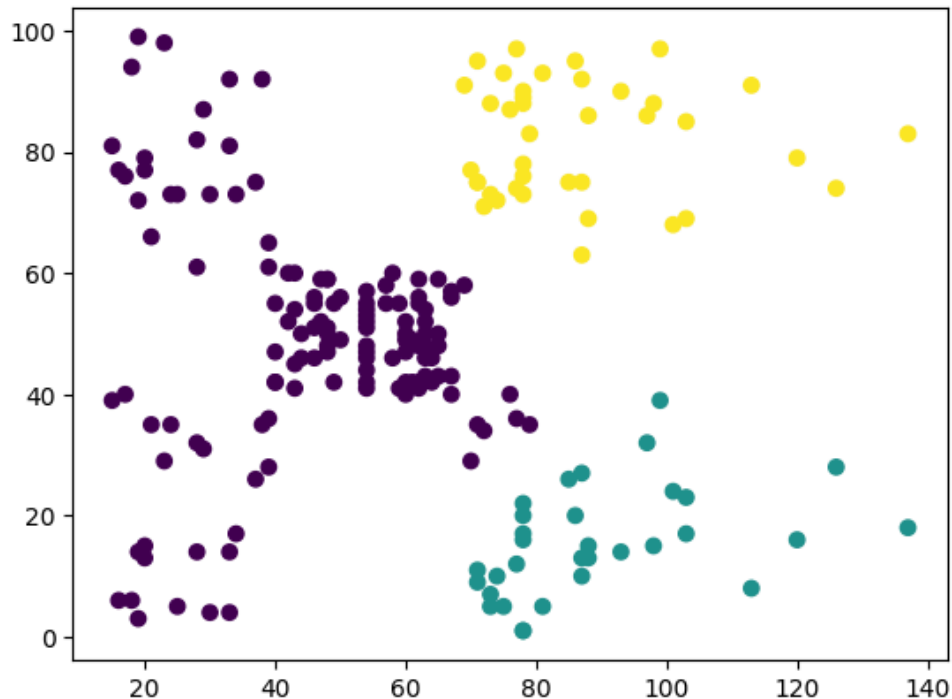
```

1 # Con 3 grupos, siguiendo la recomendación del dendrograma
2 from sklearn.cluster import AgglomerativeClustering
3 hc = AgglomerativeClustering(n_clusters=3,metric='euclidean',linkage='ward')
4 y_hc = hc.fit_predict(X)

```

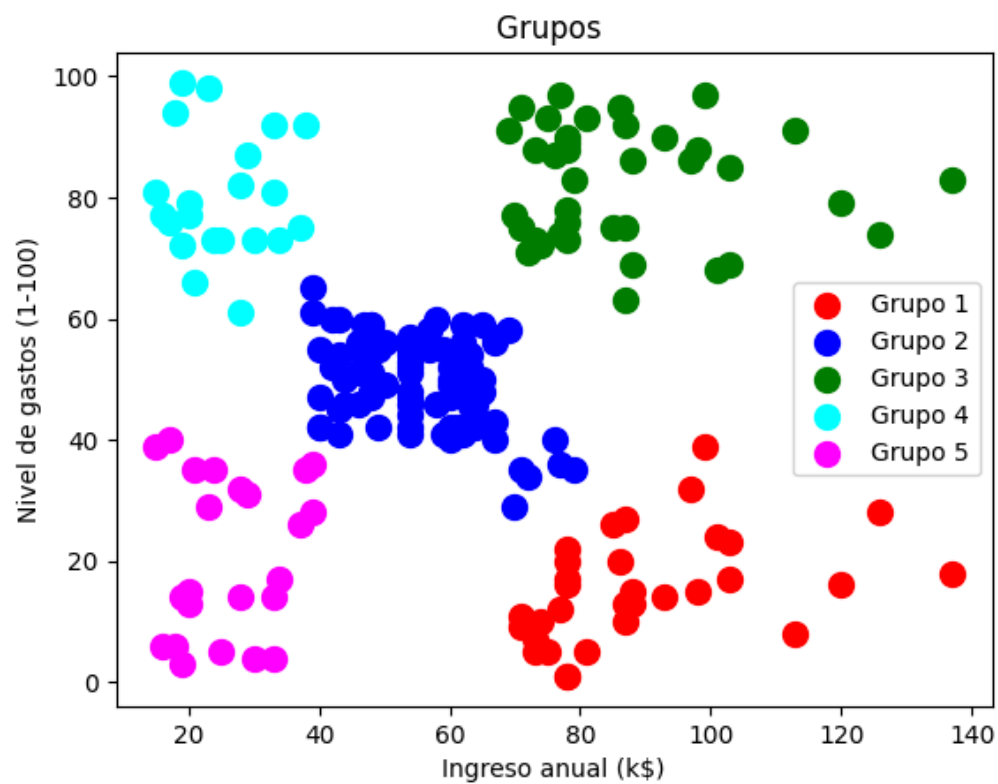
```
1 plt.scatter(X[:,0], X[:,1], c=y_hc)
```

```
⇒ <matplotlib.collections.PathCollection at 0x7fd355ae2310>
```



```
1 # Con 5 grupos, versión del analista humano
2 from sklearn.cluster import AgglomerativeClustering
3 hc = AgglomerativeClustering(n_clusters=5,metric='euclidean',linkage='ward')
4 y_hc = hc.fit_predict(X)
```

```
1 # Graficando los resultados
2 plt.scatter(X[y_hc==0, 0], X[y_hc==0, 1], s=100, c='red', label='Grupo 1')
3 plt.scatter(X[y_hc==1, 0], X[y_hc==1, 1], s=100, c='blue', label='Grupo 2')
4 plt.scatter(X[y_hc==2, 0], X[y_hc==2, 1], s=100, c='green', label='Grupo 3')
5 plt.scatter(X[y_hc==3, 0], X[y_hc==3, 1], s=100, c='cyan', label='Grupo 4')
6 plt.scatter(X[y_hc==4, 0], X[y_hc==4, 1], s=100, c='magenta', label='Grupo 5')
7 plt.title('Grupos ')
8 plt.xlabel('Ingreso anual (k$)')
9 plt.ylabel('Nivel de gastos (1-100)')
10 plt.legend()
11 plt.show()
```



1

1

1

1

1

1

1

1

1

1

1

1

1

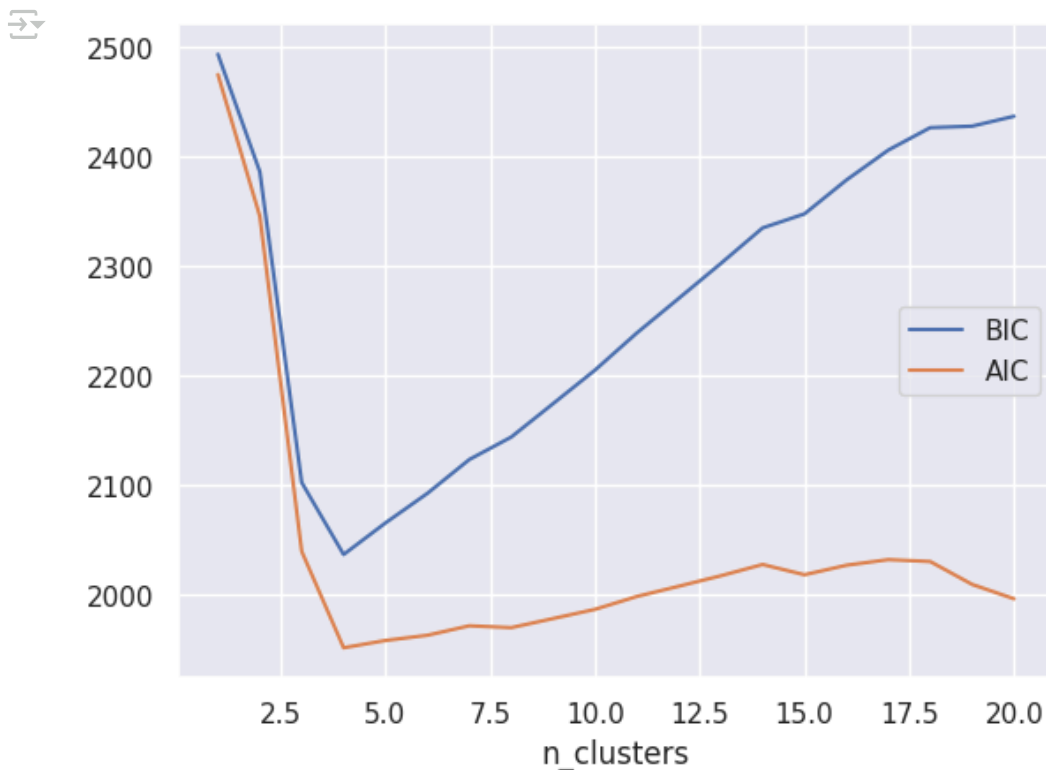
```

1 # Ejemplo 1
2 # Bibliotecas
3 import numpy as np
4 import matplotlib.pyplot as plt
5 import seaborn as sns; sns.set()
6 from matplotlib import pyplot as plt
7 from sklearn.mixture import GaussianMixture
8 from sklearn.datasets import make_blobs

1 X, y = make_blobs(n_samples=300,centers=4,cluster_std=0.6,random_state=0)
2 plt.scatter(X[:,0],X[:,1])

1 # El número óptimo de grupos es el valor que minimiza el
2 # criterio de información de Akaike (AIC) o el criterio Bayesiano (BIC)
3 # https://en.wikipedia.org/wiki/Akaike_information_criterion
4 # https://en.wikipedia.org/wiki/Bayesian_information_criterion
5 n_clusters = np.arange(1, 21)
6 models=[GaussianMixture(n,covariance_type='full',random_state=0).fit(X) for n in n_clusters]
7 plt.plot(n_clusters, [m.bic(X) for m in models], label='BIC')
8 plt.plot(n_clusters, [m.aic(X) for m in models], label='AIC')
9 plt.legend(loc='best')
10 plt.xlabel('n_clusters')
11 plt.show()




```



```

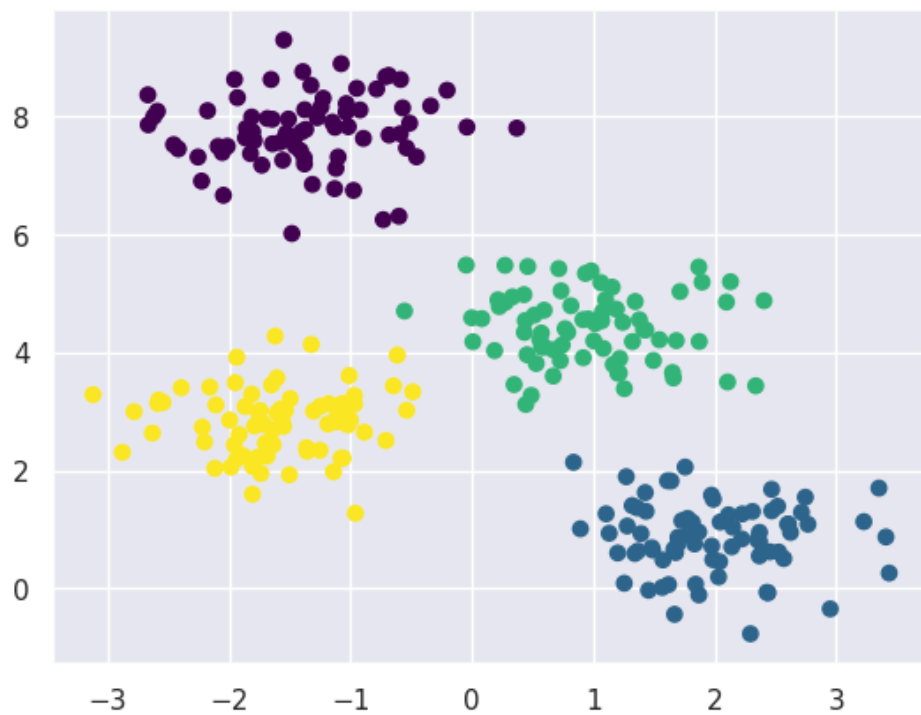
1 gmm = GaussianMixture(n_components=4)
2 gmm.fit(X)

```


GaussianMixture  
 GaussianMixture(n_components=4)


```
1 y_pred = gmm.predict(X)
2 plt.scatter(X[:,0], X[:,1], c=y_pred, cmap='viridis')
```

➞ <matplotlib.collections.PathCollection at 0x7fc1436e4d90>



1

1

1

1

1

1

1

1

1

1

1