```python
import numpy as np
from matplotlib import pyplot as plt
plt.rcParams['figure.figsize'] = (12, 6)
plt.style.use('ggplot')
np.linalg.norm(np.array([8,10]) - np.array([3,10.5]))
```

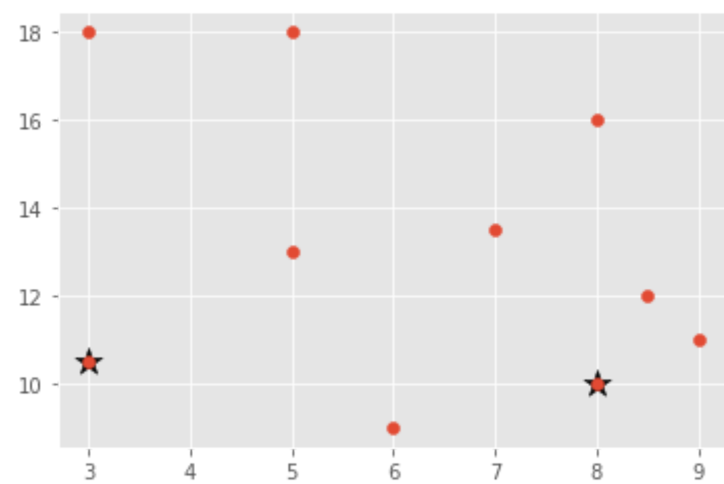
```
5.024937810560445
```

```python
np.sqrt((8-3)**2+(10-10.5)**2)
```

```
5.024937810560445
```

```python
data = np.array([[8,10],[3,10.5],[7,13.5],[5,18],[5,13],[6,9],[9,11],[3,18],[8.5,
C = np.array([[8,10],[3,10.5]])

# Gráfica
fig = plt.figure()
plt.scatter(C[0][0], C[0][1], marker='*', s=200, c='#050505')
plt.scatter(C[1][0], C[1][1], marker='*', s=200, c='#050505')
plt.scatter(data[:, 0], data[:, 1])
```

```
<matplotlib.collections.PathCollection at 0x7f2f39789dd0>
```




```python
distances = []
clusters = np.zeros(len(data))

def dist(a, b, ax=1):
    return np.linalg.norm(a - b, axis=ax)

for i in range(len(data)):
    distance = dist(data[i], C)
    distances.append(distance)
    cluster = np.argmin(distance)
    clusters[i] = cluster
```

```
print(clusters)
distances
```

```
[0. 1. 0. 1. 1. 0. 0. 1. 0. 0.]
[array([0.        , 5.02493781]),
 array([5.02493781, 0.        ]),
 array([3.64005494, 5.        ]),
 array([8.54400375, 7.76208735]),
 array([4.24264069, 3.20156212]),
 array([2.23606798, 3.35410197]),
 array([1.41421356, 6.02079729]),
 array([9.43398113, 7.5       ]),
 array([2.06155281, 5.70087713]),
 array([6.        , 7.43303437])]
```

```
for i in range(len(C)):
  points = [data[j] for j in range(len(data)) if clusters[j] == i]
  C[i] = np.mean(points, axis=0)

print(C)
```

```
[[ 7.75       11.91666667]
 [ 4.         14.875     ]]
```

```
# Gráfica
fig = plt.figure()
plt.scatter(C[0][0], C[0][1], marker='*', s=200, c='#050505')
plt.scatter(C[1][0], C[1][1], marker='*', s=200, c='#050505')
plt.scatter(data[:, 0], data[:, 1])
```

```
<matplotlib.collections.PathCollection at 0x7f2f39707a10>
```


```
'''
==========================================================
scikit-learn
```

```
==============================================================
'''
import numpy as np
import pandas as pd
from matplotlib import pyplot as plt
plt.rcParams['figure.figsize'] = (12, 6)
plt.style.use('ggplot')
```

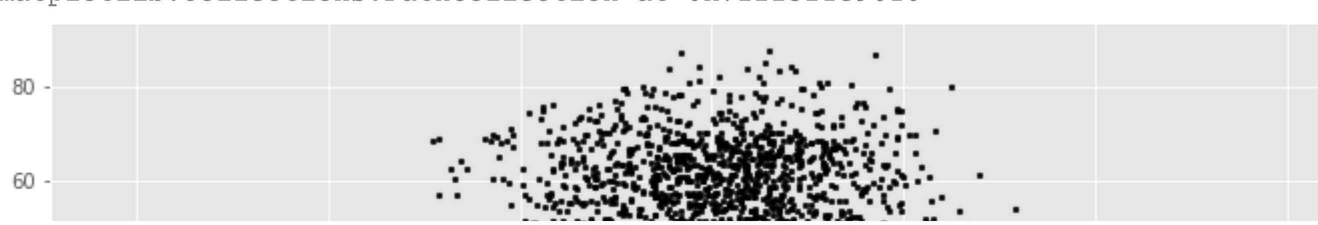
```
# Conjunto de datos xclara
data = pd.read_csv('http://dicyg.fi-c.unam.mx:8080/lalo/pypcd/presentaciones/xcla
print(data.shape)
data.head()
```

```
(3000, 2)
```

	V1	V2
0	2.072345	-3.241693
1	17.936710	15.784810
2	1.083576	7.319176
3	11.120670	14.406780
4	23.711550	2.557729

```
# Gráfica
f1 = data['V1'].values
f2 = data['V2'].values
X = np.array(list(zip(f1, f2)))
plt.scatter(f1, f2, c='black', s=7)
```

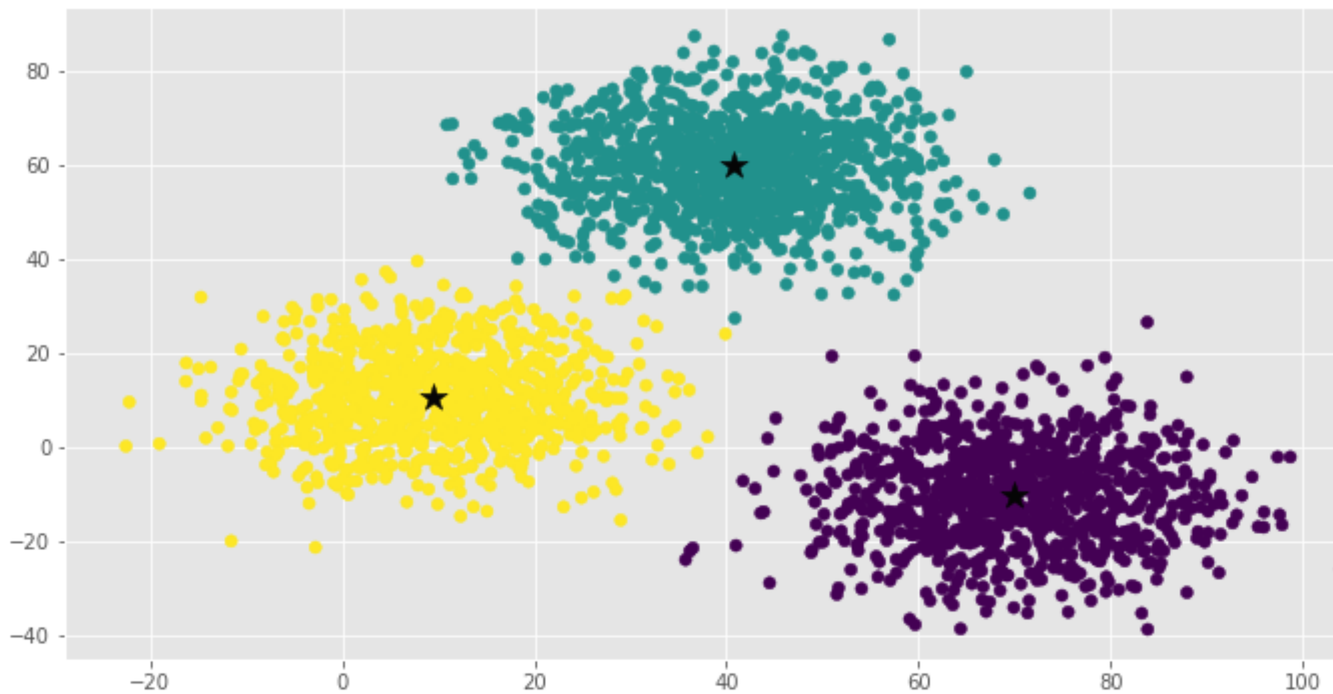
```
<matplotlib.collections.PathCollection at 0x7f2f321e9610>
```



```python

from sklearn.cluster import KMeans

# Número de grupos
kmeans = KMeans(n_clusters=3)
# Ajuste
kmeans = kmeans.fit(X)
# Etiquetas de cada clase
y = kmeans.predict(X)
# Centroides
C_skl = kmeans.cluster_centers_
```

```python
fig, ax = plt.subplots()
ax.scatter(X[:, 0], X[:, 1], c=y)
ax.scatter(C_skl[:, 0], C_skl[:, 1], marker='*', s=200, c='#050505')
```

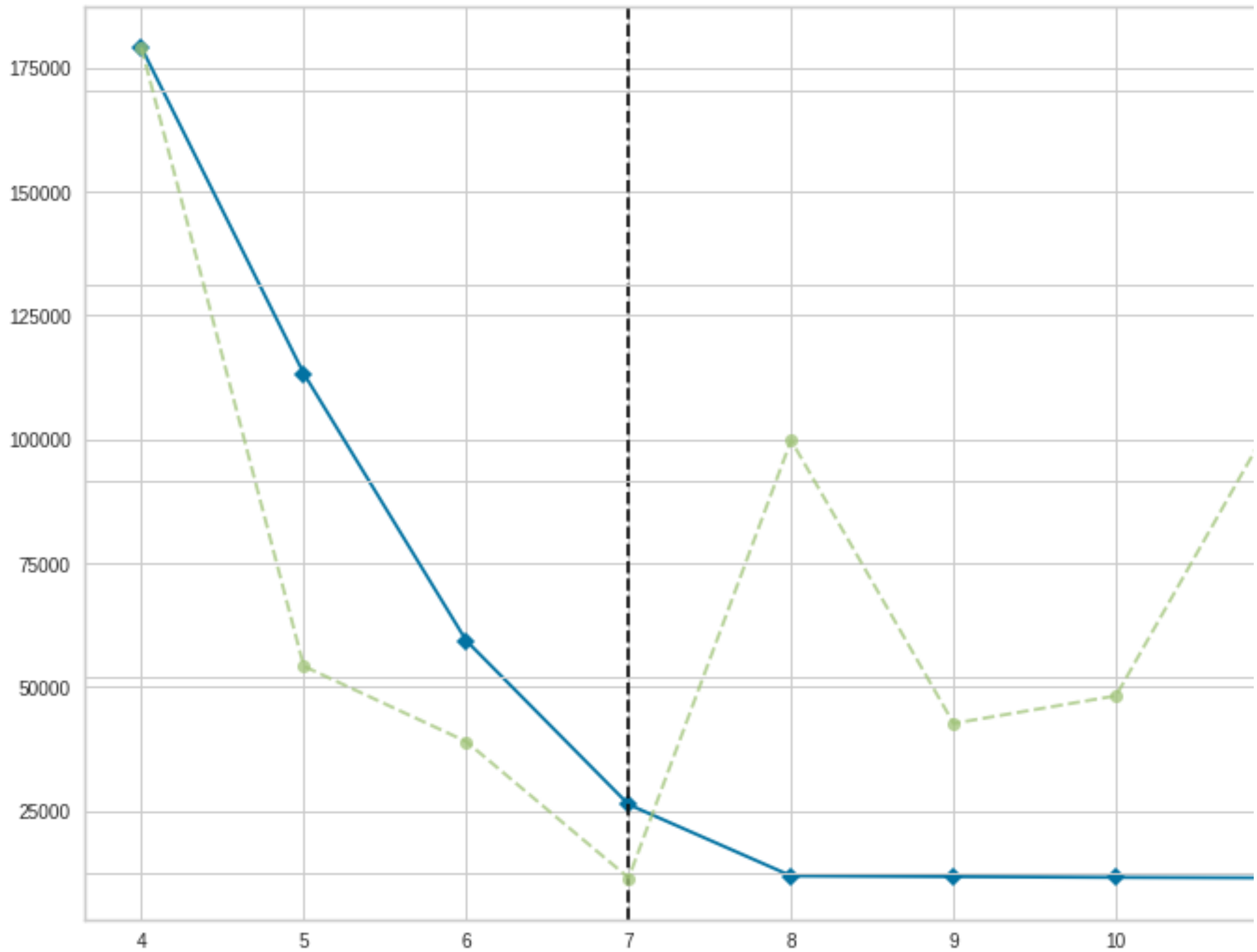
```
<matplotlib.collections.PathCollection at 0x7f2f19e231d0>
```




```python
# ¿Cómo saber el valor inicial para K?
```

```python
# Elección de k con la gráfica de codo KElbowVisualizer
# https://www.scikit-yb.org/en/latest/api/cluster/elbow.html
# By default, the scoring parameter metric is set to distortion,
# which computes the sum of squared distances from each point to its assigned cen
from sklearn.cluster import KMeans
from sklearn.datasets import make_blobs
from yellowbrick.cluster import KElbowVisualizer

X, y = make_blobs(n_samples=1000, n_features=12, centers=8, random_state=42)
```

```
visualizer = KElbowVisualizer(KMeans(), k=(4,12), timings=True)
visualizer.fit(X)
plt.xlabel('Número de grupos')
plt.ylabel('Distorsión')
plt.show()
```




```
# KMeans in depth
# https://github.com/jakevdp/PythonDataScienceHandbook/blob/master/notebooks/05.1
```

```
# bibliotecas
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
```

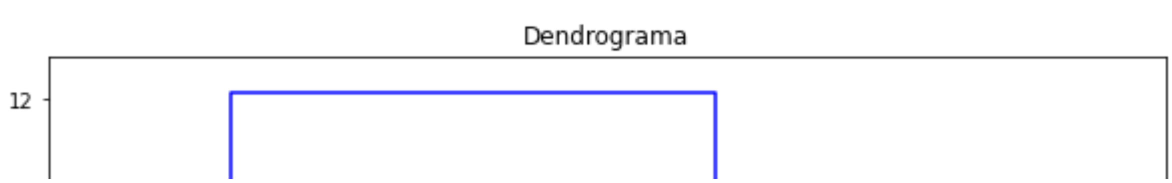
```
# datos
#https://archive.ics.uci.edu/ml/machine-learning-databases/00292/Wholesale%20cust
url = 'https://bit.ly/2COHM14'
data = pd.read_csv(url)
data.head()

```

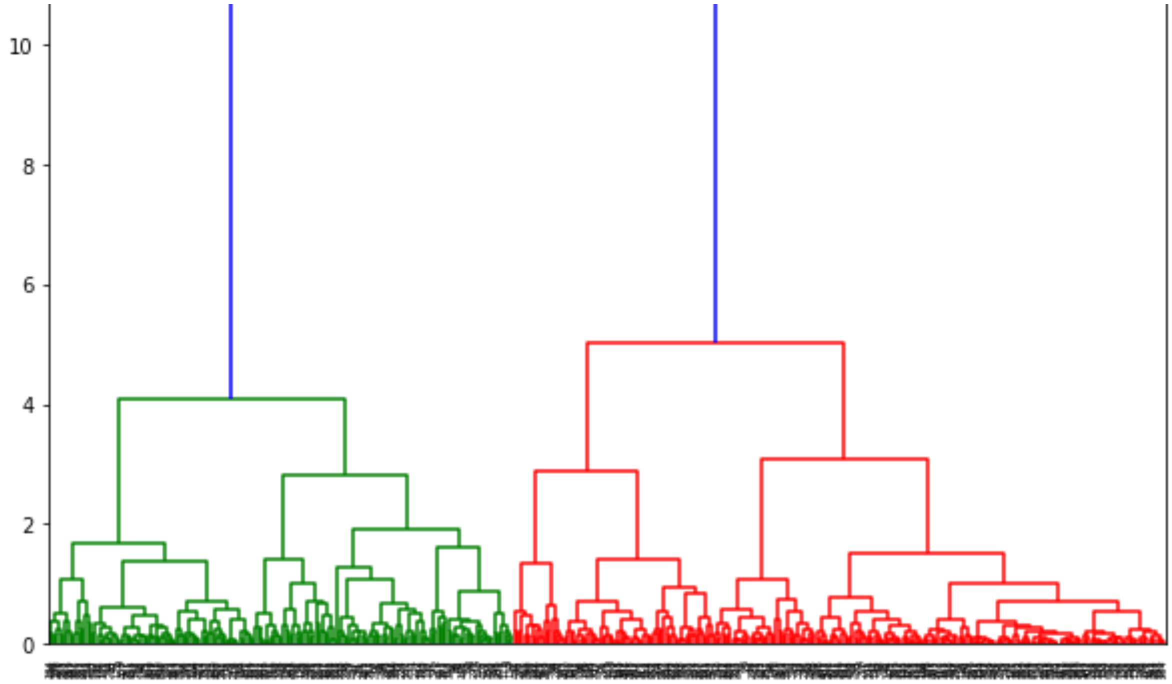
	Channel	Region	Fresh	Milk	Grocery	Frozen	Detergents_Paper	Delicassen
0	2	3	12669	9656	7561	214	2674	1338
1	2	3	7057	9810	9568	1762	3293	1776
2	2	3	6353	8808	7684	2405	3516	7844
3	1	3	13265	1196	4221	6404	507	1788
4	2	3	22615	5410	7198	3915	1777	5185

```
from sklearn.preprocessing import normalize
data_scaled = normalize(data)
data_scaled = pd.DataFrame(data_scaled, columns=data.columns)
data_scaled.head()
```

	Channel	Region	Fresh	Milk	Grocery	Frozen	Detergents_Paper	Delicas
0	0.000112	0.000168	0.708333	0.539874	0.422741	0.011965	0.149505	0.07
1	0.000125	0.000188	0.442198	0.614704	0.599540	0.110409	0.206342	0.11
2	0.000125	0.000187	0.396552	0.549792	0.479632	0.150119	0.219467	0.48
3	0.000065	0.000194	0.856837	0.077254	0.272650	0.413659	0.032749	0.11
4	0.000079	0.000119	0.895416	0.214203	0.284997	0.155010	0.070358	0.20

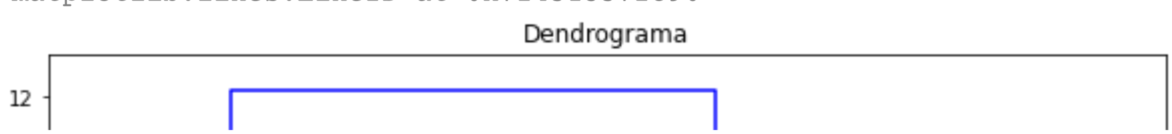
```
import scipy.cluster.hierarchy as shc
plt.figure(figsize=(10, 7))
plt.title("Dendrograma")
dend = shc.dendrogram(shc.linkage(data_scaled, method='ward'))
```






```
plt.figure(figsize=(10, 7))
plt.title("Dendrograma")
dend = shc.dendrogram(shc.linkage(data_scaled, method='ward'))
plt.axhline(y=6, color='r', linestyle='--')
```

```
<matplotlib.lines.Line2D at 0x7f4513572c90>
```




```
from sklearn.cluster import AgglomerativeClustering
cluster = AgglomerativeClustering(n_clusters=2, affinity='euclidean',
```

```
                                linkage='ward')
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, 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, 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, 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, 1, 0, 0, 0, 0, 1, 1, 0, 0, 0,
       1, 0, 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, 0, 1, 0, 1])
```

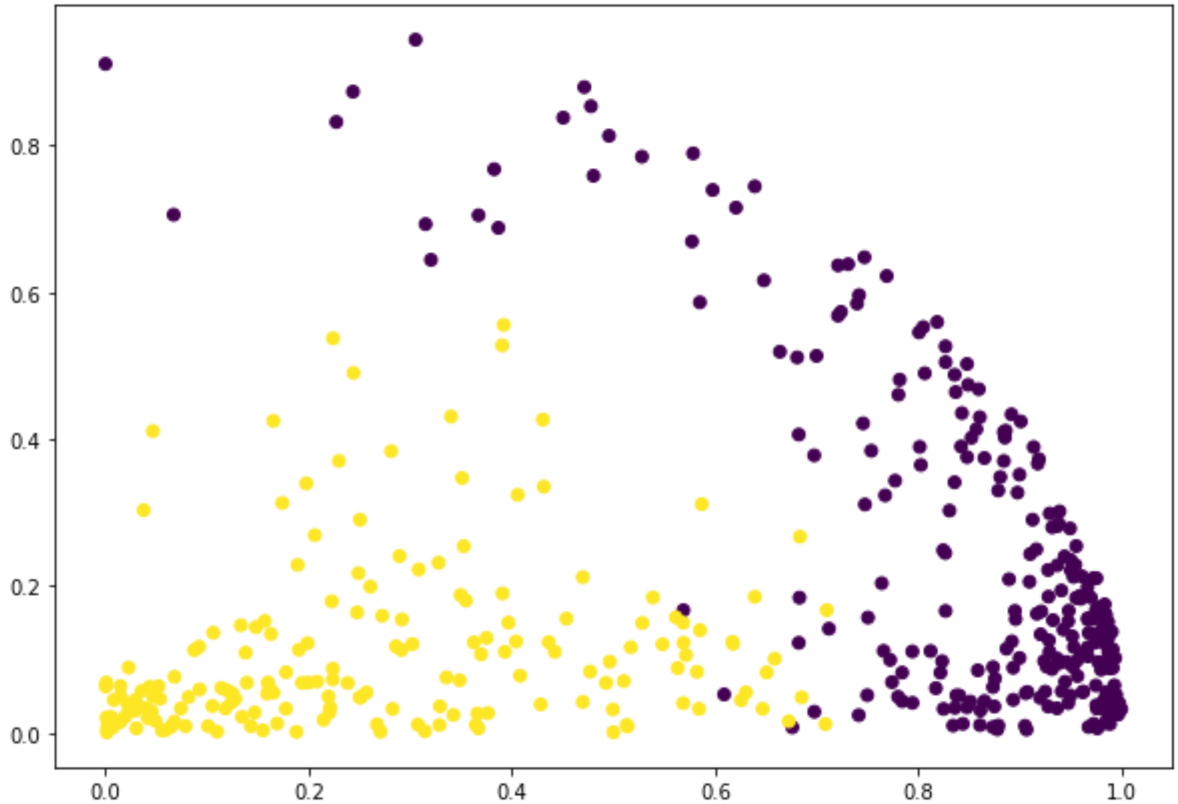
```
plt.figure(figsize=(10, 7))
plt.scatter(data_scaled['Milk'], data_scaled['Grocery'], c=cluster.labels_)
```

```
<matplotlib.collections.PathCollection at 0x7f451ba13550>
```


```
plt.figure(figsize=(10, 7))
```

```
plt.scatter(data_scaled['Fresh'], data_scaled['Frozen'], c=cluster.labels_)
```

```
<matplotlib.collections.PathCollection at 0x7f450bfe0410>
```

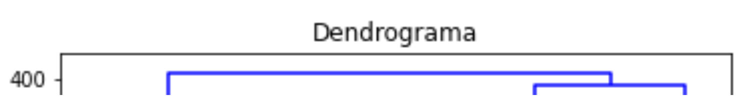


```python
# Bibliotecas
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd
```

```python
# Dataset
dataset = pd.read_csv('http://dicyg.fi-c.unam.mx:8080/lalo/pypcd/presentaciones/M
X = dataset.iloc[:, [3, 4]].values
dataset.head()
```

	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
3	4	Female	23	16	77
4	5	Female	31	17	40

```python
# Dendrograma para determinar el número óptimo de grupos
import scipy.cluster.hierarchy as sch
dendrograma = sch.dendrogram(sch.linkage(X, method = 'ward'))
plt.title('Dendrograma')
plt.xlabel('Clientes')
plt.ylabel('Distancia euclideana')
plt.show()
```

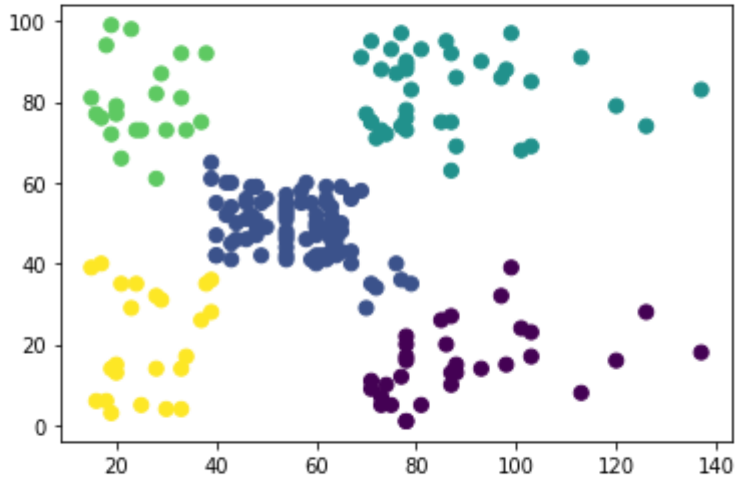



```python
from sklearn.cluster import AgglomerativeClustering
hc = AgglomerativeClustering(n_clusters = 5, affinity = 'euclidean',
                             linkage = 'ward')
y_hc = hc.fit_predict(X)
```

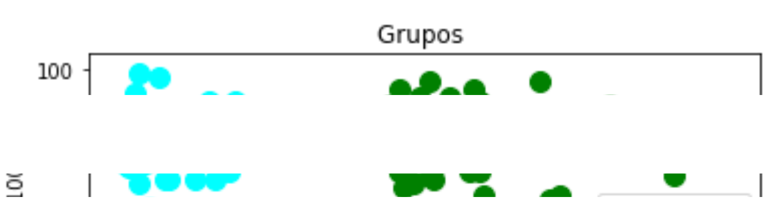
```
y_hc = hc.fit_predict(X)
```

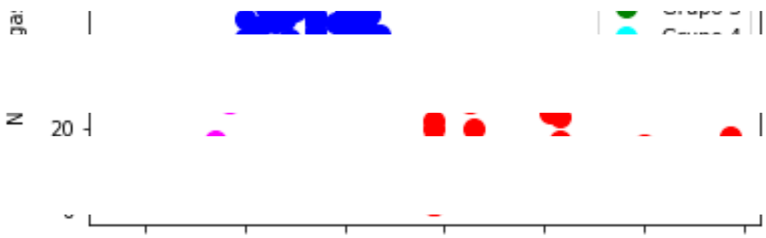
```
plt.scatter(X[:,0],X[:,1],s=50,c=y_hc)
```

<matplotlib.collections.PathCollection at 0x7fbb7d03dd50>




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





1

1

1

1

1

1

1

1

1

1

1

1

1

1

1

1

1

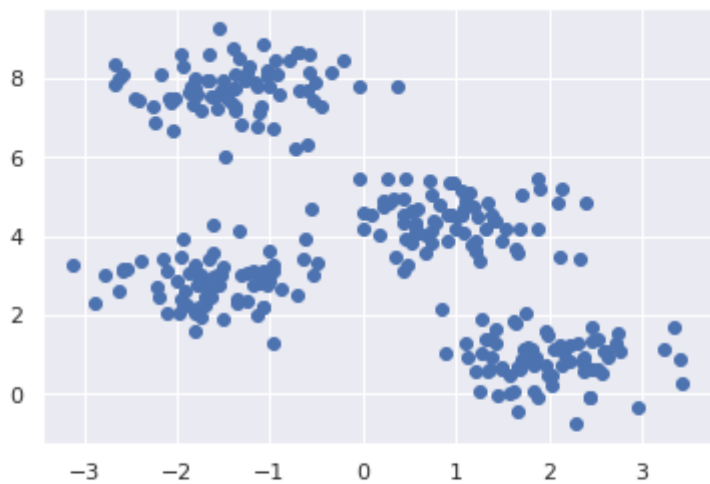
1

1

```python
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
sns.set()
from sklearn.mixture import GaussianMixture
from sklearn.datasets import make_blobs
```

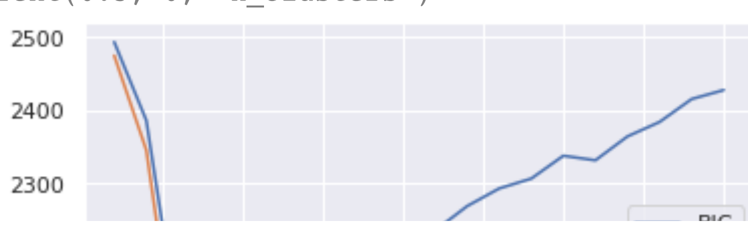
```python
X, y = make_blobs(n_samples=300, centers=4, cluster_std=0.6, random_state=0)
plt.scatter(X[:,0],X[:,1])
```

```
<matplotlib.collections.PathCollection at 0x7faf2360d890>
```




```python
# The optimal number of clusters (K) is the value that minimizes the
# Akaike information criterion (AIC) or the Bayesian information criterion (BIC)
# https://en.wikipedia.org/wiki/Akaike_information_criterion
# https://en.wikipedia.org/wiki/Bayesian_information_criterion
n_clusters = np.arange(1, 21)
models = [GaussianMixture(n, covariance_type='full',
                          random_state=0).fit(X) for n in n_clusters]
plt.plot(n_clusters, [m.bic(X) for m in models], label='BIC')
plt.plot(n_clusters, [m.aic(X) for m in models], label='AIC')
plt.legend(loc='best')
plt.xlabel('n_clusters')
```

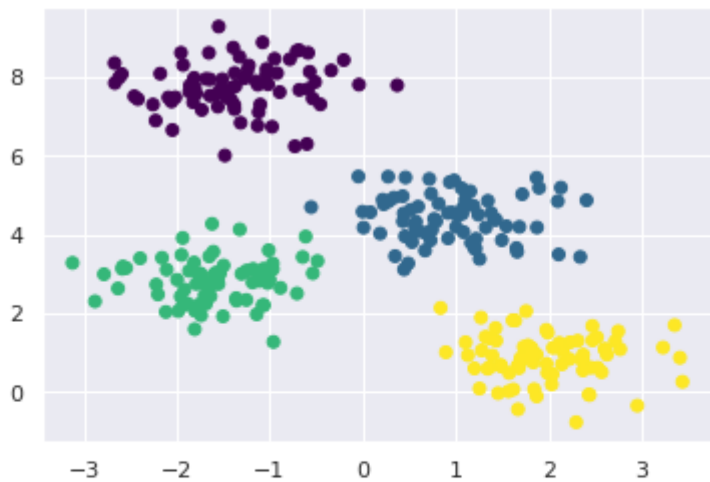
```
Text(0.5, 0, 'n_clusters')
```




```
gmm = GaussianMixture(n_components=4)
gmm.fit(X)
```

```
GaussianMixture(n_components=4)
```

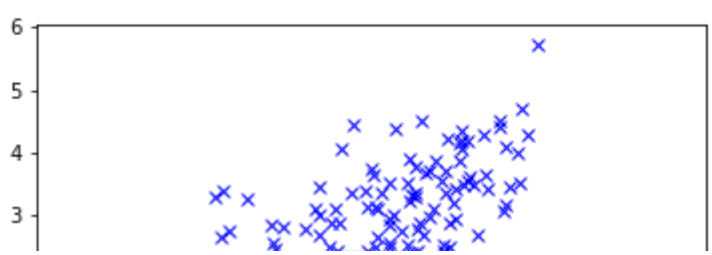
```
labels = gmm.predict(X)
plt.scatter(X[:, 0], X[:, 1], c=labels, cmap='viridis');
```

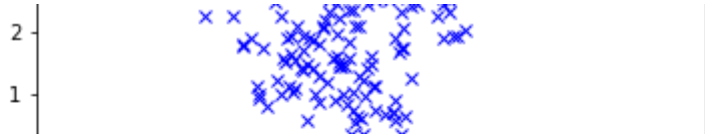



```
import numpy as np
import matplotlib.pyplot as plt
from sklearn.mixture import GaussianMixture
```

```
# https://bit.ly/3Bic3iu
X_train = np.load('data.npy')
```

```
plt.plot(X_train[:,0], X_train[:,1], 'bx')
plt.axis('equal')
plt.show()
```

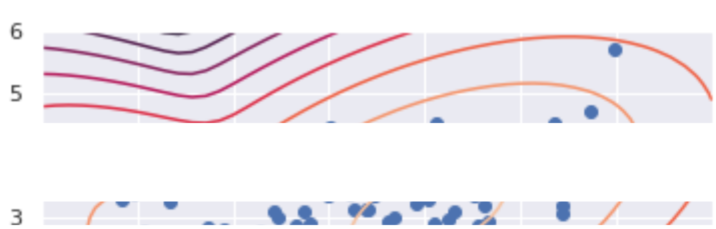





```
gmm = GaussianMixture(n_components=2)
gmm.fit(X_train)

print("Medias: \n", gmm.means_)
print('\n')
print("Covarianzas: \n",gmm.covariances_)

```

```
Medias: 
 [[3.04641134 3.10654272]
 [1.60718016 1.35251723]]


Covarianzas: 
 [[[ 0.83656079  0.37865596]
  [ 0.37865596  0.72727426]]

 [[ 0.74995307 -0.5010097 ]
  [-0.5010097   0.74377694]]]
```

```
X, Y = np.meshgrid(np.linspace(-1, 6), np.linspace(-1,6))
XX = np.array([X.ravel(), Y.ravel()]).T
Z = gmm.score_samples(XX)
Z = Z.reshape((50,50))

plt.contour(X, Y, Z)
plt.scatter(X_train[:, 0], X_train[:, 1])
plt.show()

```




1
1
0
1
1
1
1
1
1
1
1
1
1
1
1
1
1
1
1
1
1