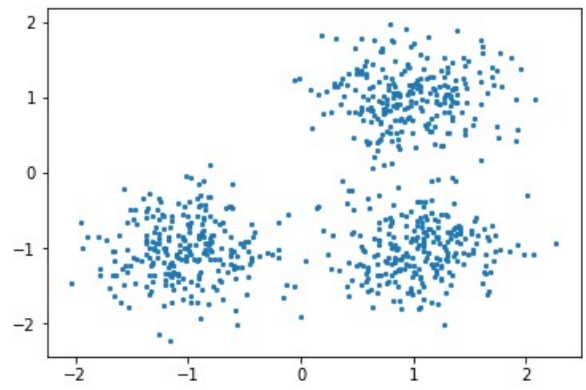
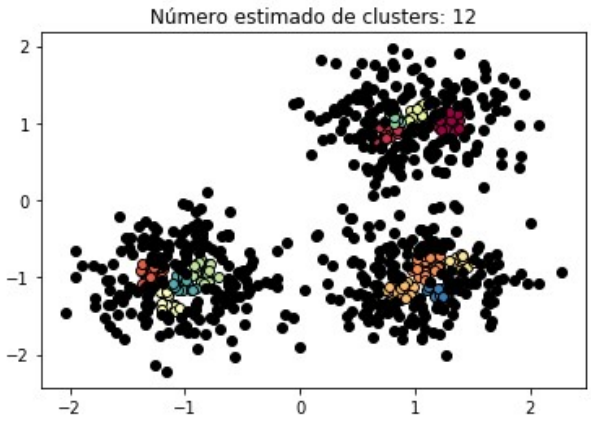
ML_CLUSTERING_DBSCAN_01	Segmentación en clusters utilizando DBScan	ML

En esta práctica se utiliza el algoritmo de clustering DBScan (basado en criterios de densidad). Para ello, elegidos unos centroides iniciales, se genera de forma aleatoria un conjunto de datos y se aplica el algoritmo DBScan, visualizando los resultados que se obtienen con diferentes valores del radio mínimo para considerar un cluster.

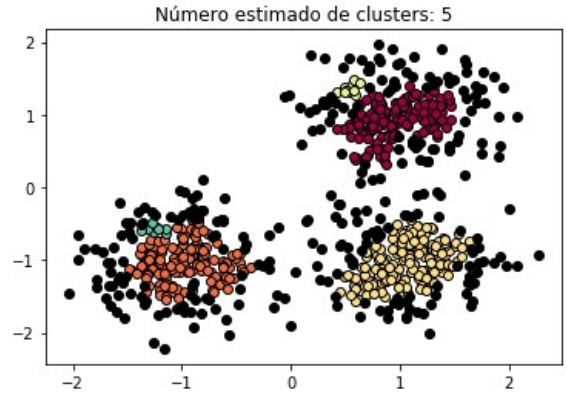
Además se evalúan diferentes métricas disponibles (homogeneidad, completeness,…)



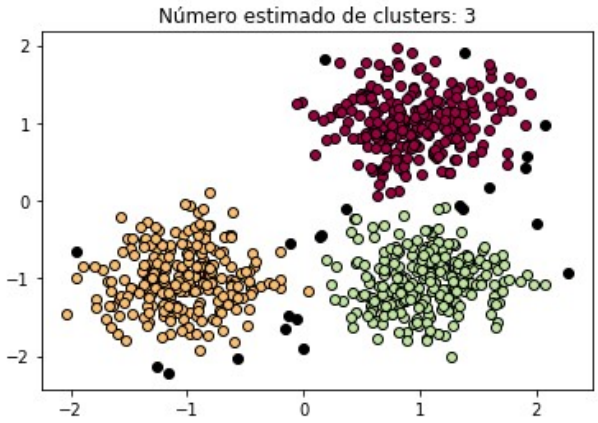
Conjunto inicial de datos



Clustering con radio = 0.1



Clustering con radio = 0.14



Clustering con radio = 0.3

SOLUCIÓN

Definir las librerías a utilizar

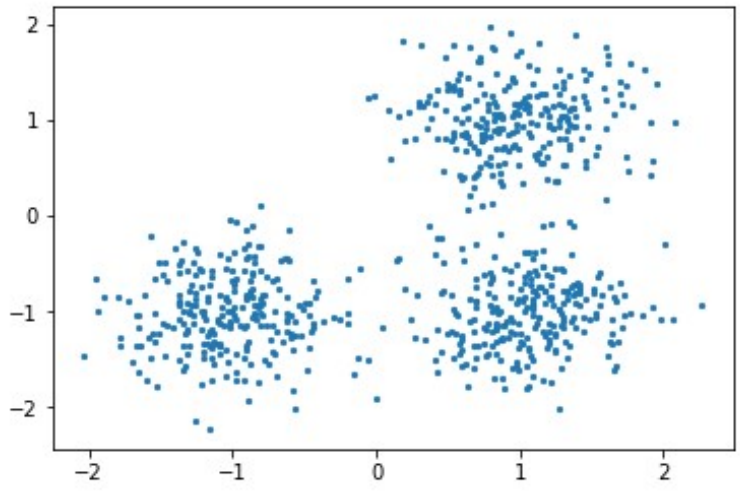
```
# Importar librerías
import numpy as np
import matplotlib.pyplot as plt
from sklearn import metrics
from sklearn.datasets.samples_generator import make_blobs
from sklearn.cluster import DBSCAN
```

Definir los centroides iniciales

```
# Definir centroides iniciales
centroides = [[1, 1], [-1, -1], [1, -1]]
```

Generar el conjunto de datos X y visualizarlo

```
# Generar un conjunto de datos X utilizando 'make_blobs'
X, labels_true = make_blobs(n_samples=750, centers=centroides, cluster_std=0.4,
                            random_state=0)

# Visualizar los datos generados de forma aleatoria
plt.scatter(X[:,0],X[:,1],s=5)
plt.show()
```



Definir el radio y ejecutar el algoritmo DBScan

```
# Ejecutar DBSCAN  (eps => radio, min_samples => mínimo número de puntos para considerar un
cluster). Ajustar a los datos generados X
radio=0.1
print("Algoritmo DBScan con radio (eps) = ",radio)

clustering = DBSCAN(eps=radio, min_samples=10).fit(X)

core_samples_mask = np.zeros_like(clustering.labels_, dtype=bool)
# core_sample_indices_   (índices de los núcleos)
core_samples_mask[clustering.core_sample_indices_] = True
```

```
labels = clustering.labels_
# Una etiqueta con valor -1 se corresponde con ruido. No se clasifica en ningún cluster
```

Visualizar el número de clusters y las métricas

```
# Número de clusters en las etiquetas. Ignorando el ruido (label=-1)
n_clusters_ = len(set(labels)) - (1 if -1 in labels else 0)

# Visualizar diferentes métricas
print('Número estimado de clusters: %d' % n_clusters_)

print("Homogeneidad: %0.3f" % metrics.homogeneity_score(labels_true, labels))
print("Completeness: %0.3f" % metrics.completeness_score(labels_true, labels))
print("V-measure: %0.3f" % metrics.v_measure_score(labels_true, labels))
print("Adjusted Rand Index: %0.3f"
      % metrics.adjusted_rand_score(labels_true, labels))
print("Adjusted Mutual Information: %0.3f"
      % metrics.adjusted_mutual_info_score(labels_true, labels))
print("Silhouette Coefficient: %0.3f"
      % metrics.silhouette_score(X, labels))
```

Algoritmo DBScan con radio (eps) = 0.1

Número estimado de clusters: 12

Homogeneidad: 0.281

Completeness: 0.241

V-measure: 0.260

Adjusted Rand Index: 0.017

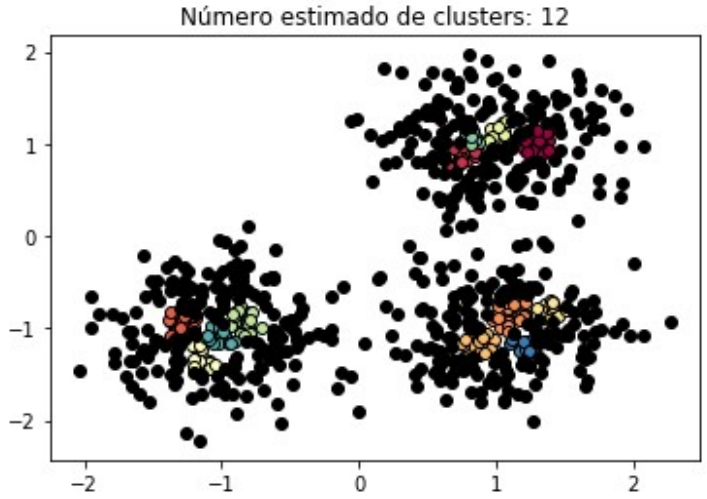
Adjusted Mutual Information: 0.231

Silhouette Coefficient: -0.401

Visualizar el gráfico con los resultados

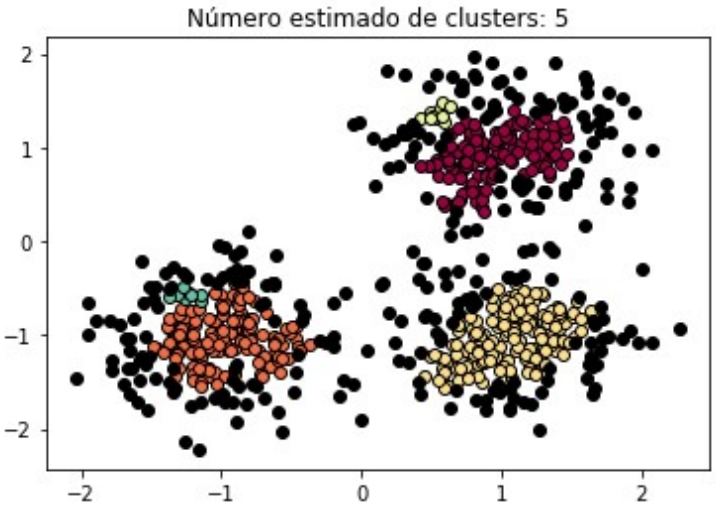
```
# Visualizar resultados
unique_labels = set(labels)

# Definir los colores. [x,x,x,x]
colors = [plt.cm.Spectral(each)
          for each in np.linspace(0, 1, len(unique_labels))]

for k, col in zip(unique_labels, colors):
    if k == -1:
        # Color negro utilizado para representar el ruido. Puntos fuera de un cluster
        col = [0, 0, 0, 1]

    class_member_mask = (labels == k)

    xy = X[class_member_mask & core_samples_mask]
    plt.plot(xy[:, 0], xy[:, 1], 'o', markerfacecolor=tuple(col),
             markeredgecolor='k', markersize=6)

    xy = X[class_member_mask & ~core_samples_mask]
```

```
    plt.plot(xy[:, 0], xy[:, 1], 'o', markerfacecolor=tuple(col),
             markeredgecolor='k', markersize=6)
plt.title('Número estimado de clusters: %d' % n_clusters_)
plt.show()
```



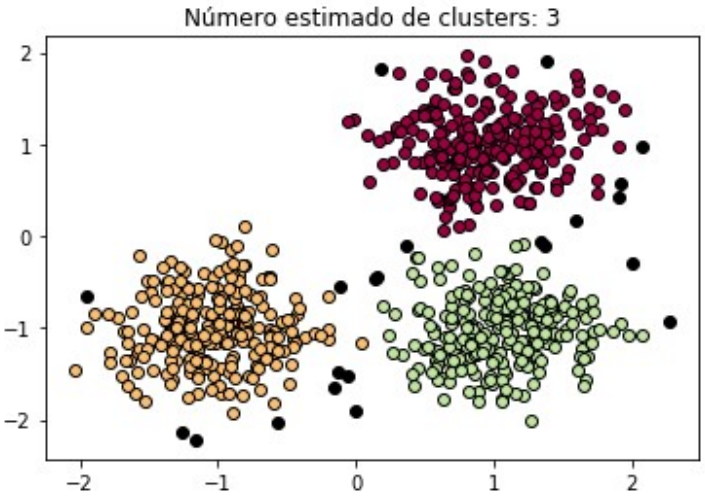
Utilizando un radio = 0.14 los resultados son:

Algoritmo DBScan con radio (eps) = 0.14
Número estimado de clusters: 5
Homogeneidad: 0.622
Completeness: 0.474
V-measure: 0.538
Adjusted Rand Index: 0.376
Adjusted Mutual Information: 0.472
Silhouette Coefficient: 0.024



Utilizando un radio = 0.3 los resultados son:

Algoritmo DBScan con radio (eps) = 0.3
Número estimado de clusters: 3
Homogeneidad: 0.947
Completeness: 0.868
V-measure: 0.906
Adjusted Rand Index: 0.943
Adjusted Mutual Information: 0.867
Silhouette Coefficient: 0.621


Número estimado de clusters: 3
2
1
0
−1
−2
−2
−1
0
1
2