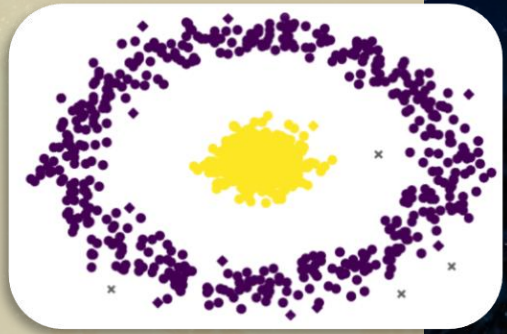
DBSCAN

DBSCAN

Density based spatial clustering of applications with noise
(Agrupamiento espacial de aplicaciones con ruido basado en densidad)

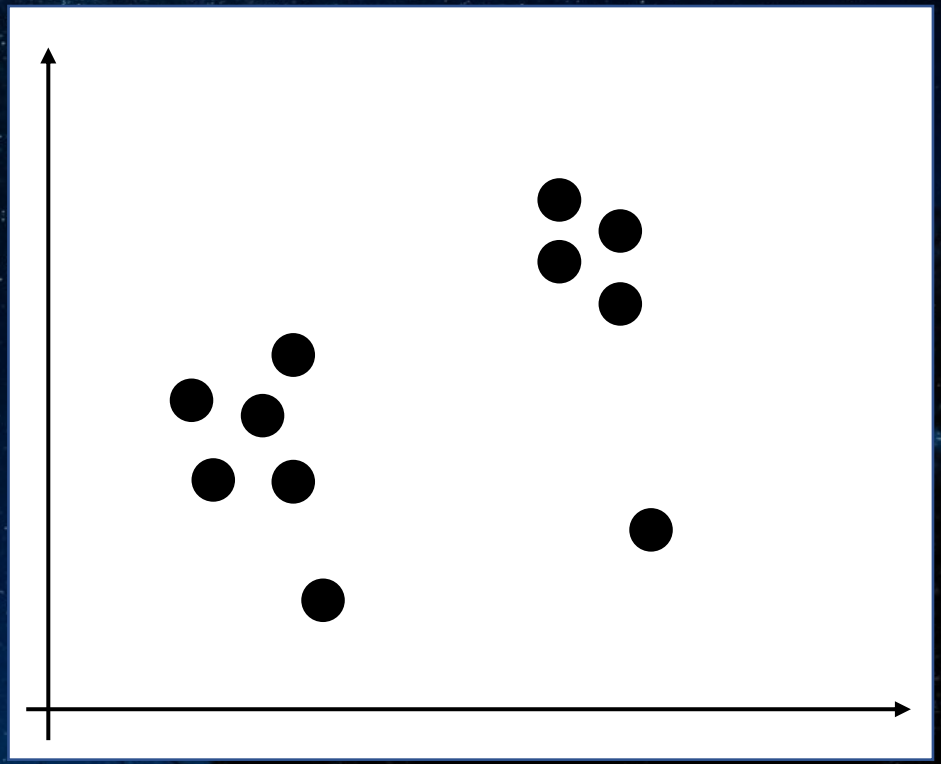


- No necesita la definición de cantidad de clusters.
- Se adapta mejor a formas no convexas.
- Identifica outliers.
- Reconoce regiones densas (con muchos datos) y las distingue de las poco densas.
- Se adecúa muy bien al agrupamiento intuitivo humano.

DBSCAN

Algoritmo

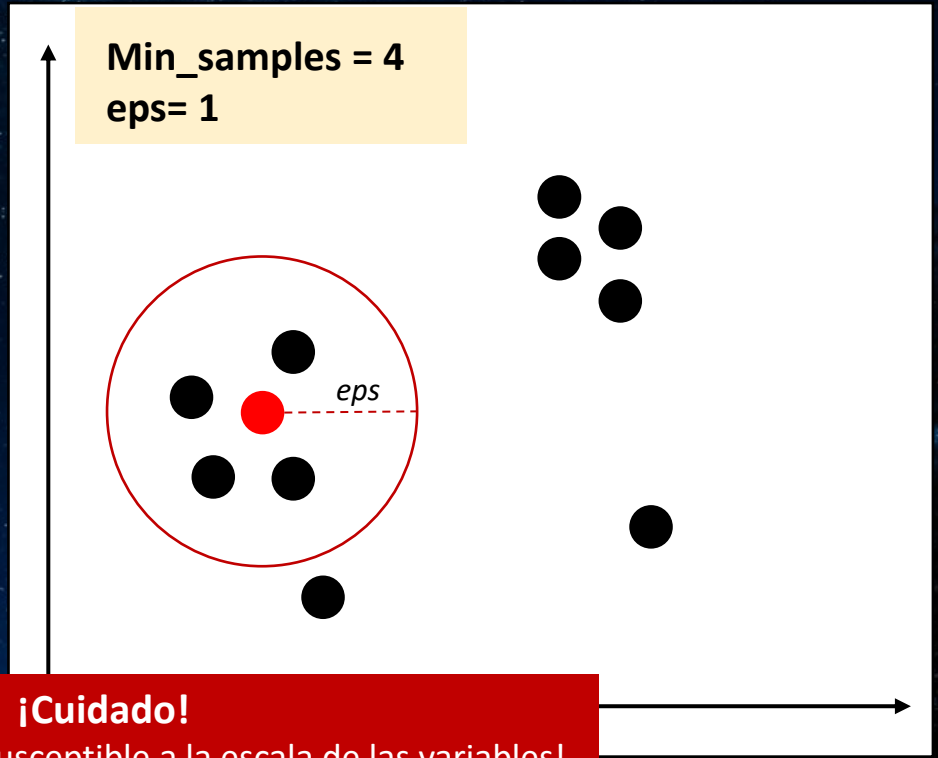
- **Epsilon**: distancia para considerar dos datos vecinos.
- **Min_samples**: mínima cantidad de datos necesarios para considerarse un *core sample*.



DBSCAN

Algoritmo

- Comienza con un dato aleatorio y analiza los vecinos que haya con un radio *eps*.
- Si num_vecinos ≥ min_samples entonces el dato es considerado un *core sample*. Si no existe el cluster, se crea.

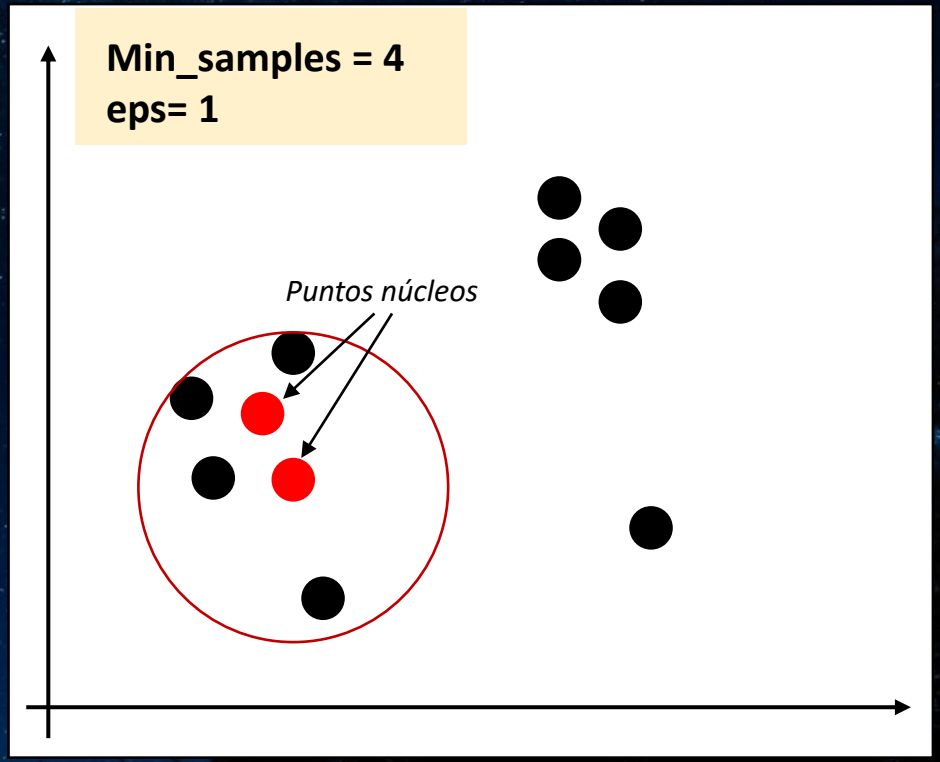


¡Cuidado!
Al medir distancia, es susceptible a la escala de las variables!

DBSCAN

Algoritmo

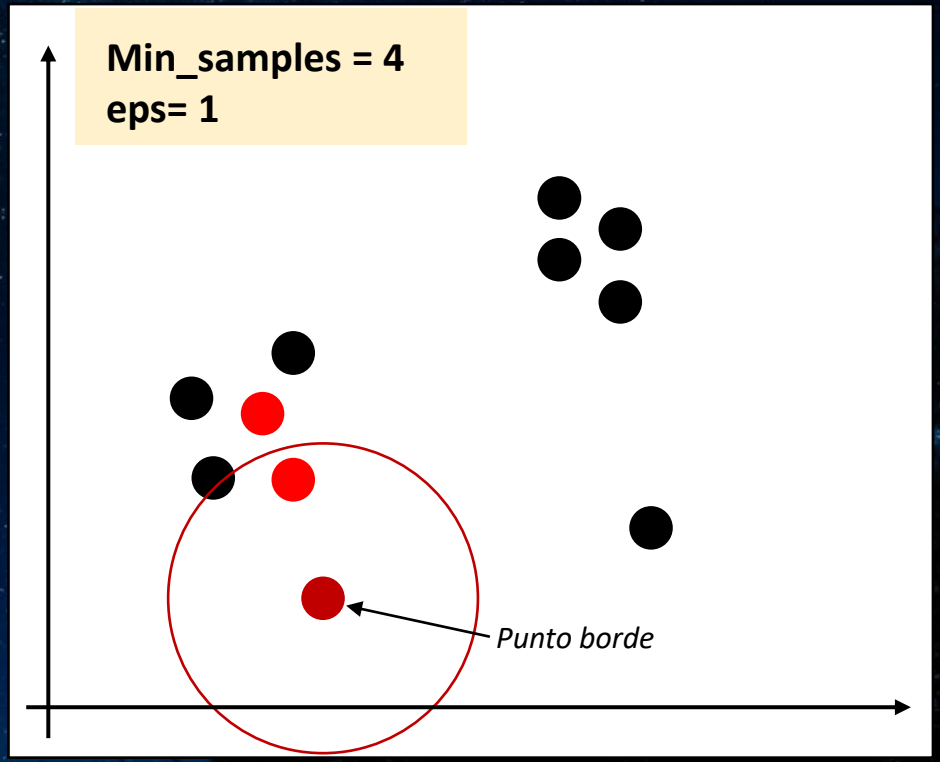
- Continúa con un vecino del cluster generado, aplicando el mismo procedimiento.



DBSCAN

Algoritmo

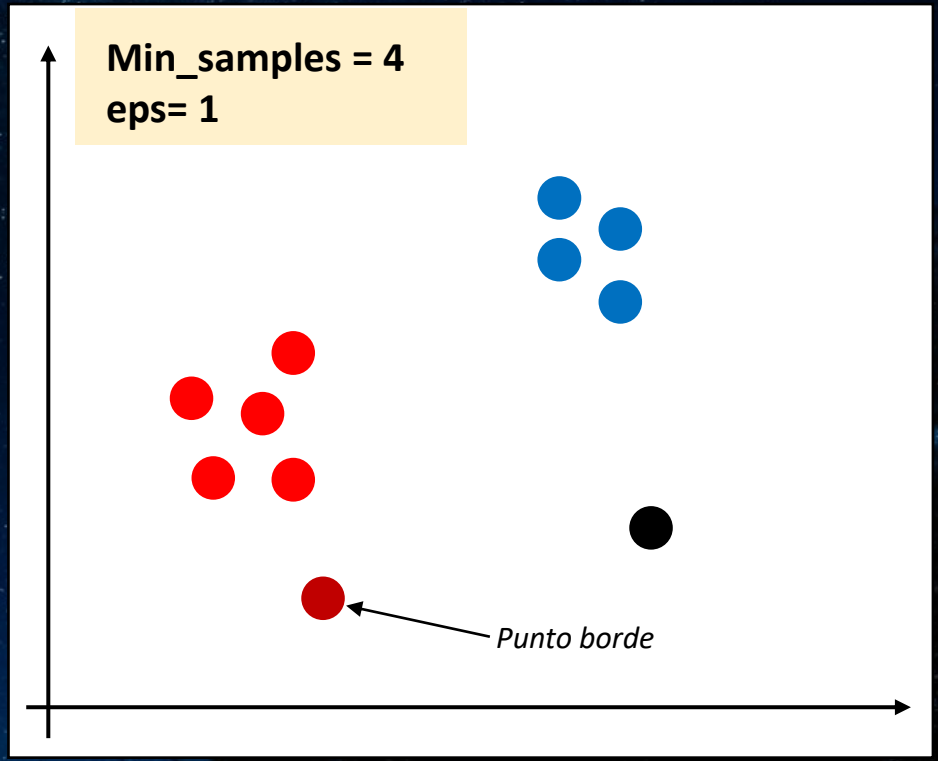
- Si un vecino de un *core sample* no tiene suficientes vecinos para ser otro *core sample* del cluster, entonces es considerado un *border point*.



DBSCAN

Algoritmo

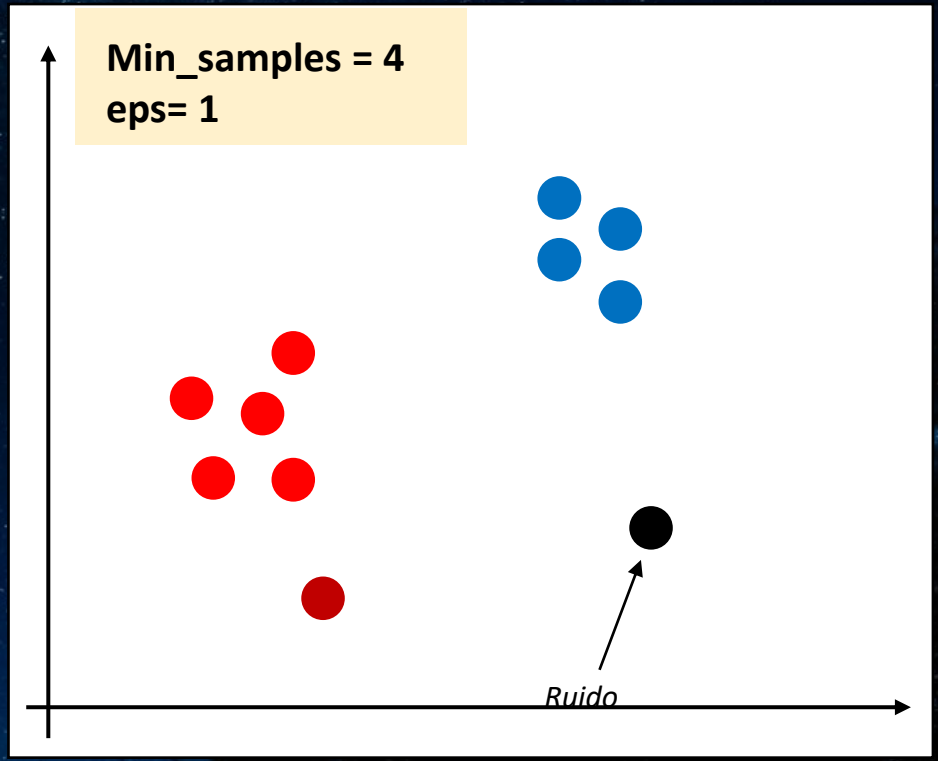
- Continúa con el resto de vecinos hasta que se terminan.
- Luego, comienza con un nuevo punto aleatorio y realiza el mismo procedimiento.



DBSCAN

Algoritmo

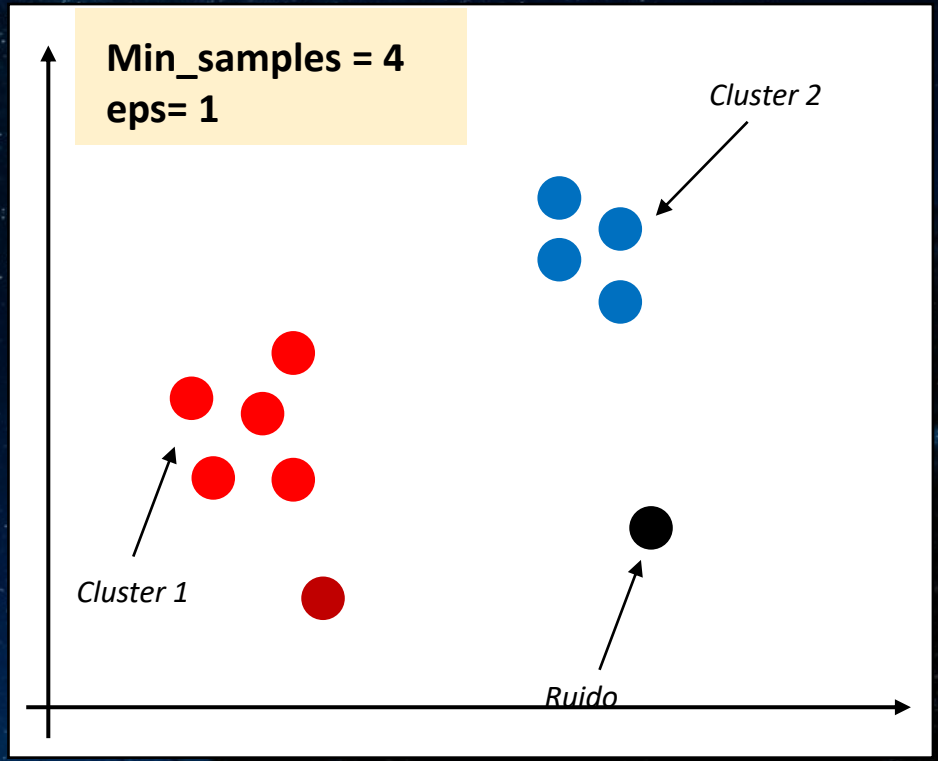
- Si algún dato no tiene el mínimo de vecinos dentro del radio *eps* y no es vecino de ningún *core sample* entonces es considerado ruido (outlier).



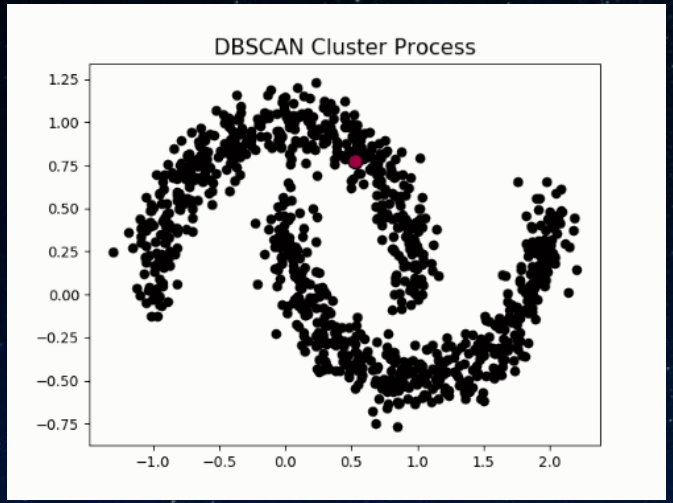
DBSCAN

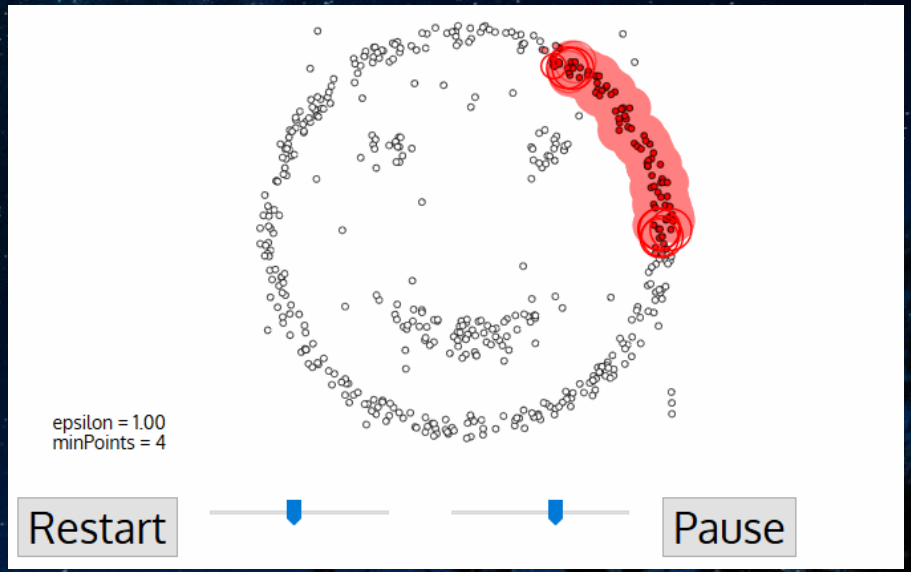
Algoritmo

- Al finalizar, el algoritmo detecta una cierta cantidad de clusters con sus puntos núcleo y puntos bordes y datos outliers que no pertenecen a ningún cluster.



DBSCAN - ejemplos





Visualización:

https://www.naftaliharris.com/blog/visualizing-dbscan-clustering/

DBSCAN en sklearn

```
from sklearn.cluster import DBSCAN

db = DBSCAN(eps= 0.5, min_samples=5).fit(data)
labels = db.labels_

noise_maks= labels==-1
X_noise= data[noise_maks]

core_samples_mask = np.zeros_like(db.labels_, dtype=bool)
core_samples_mask[db.core_sample_indices_] = True
X_coresamples= data[~noise_maks & core_samples_mask]
```

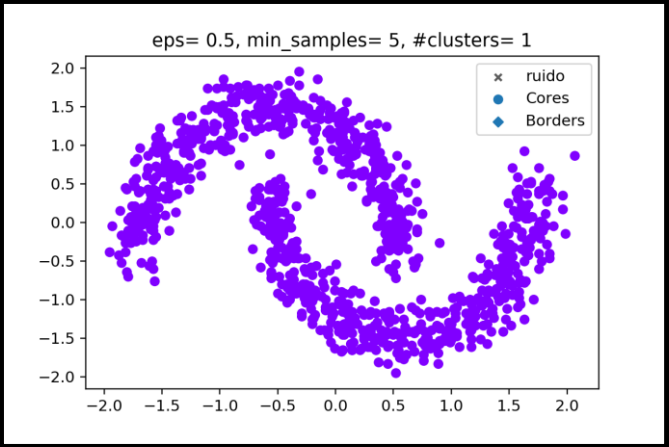
Parámetros por defecto

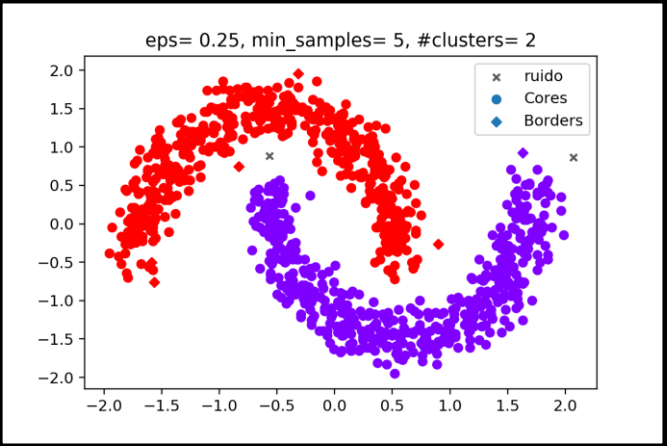
Etiquetas de cada cluster. Sklearn pone -1 para datos con ruido

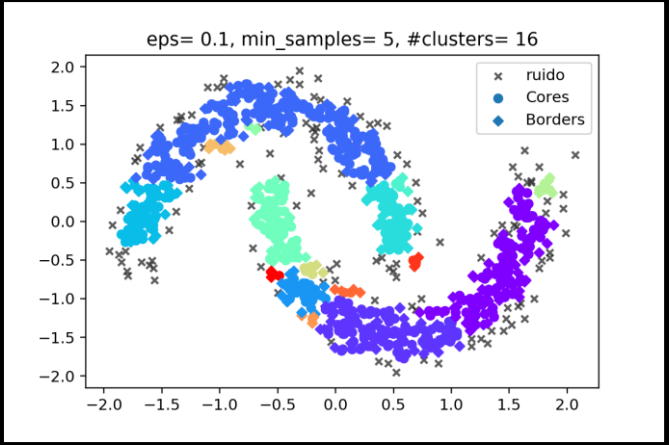
Filtramos los datos detectados como outliers.

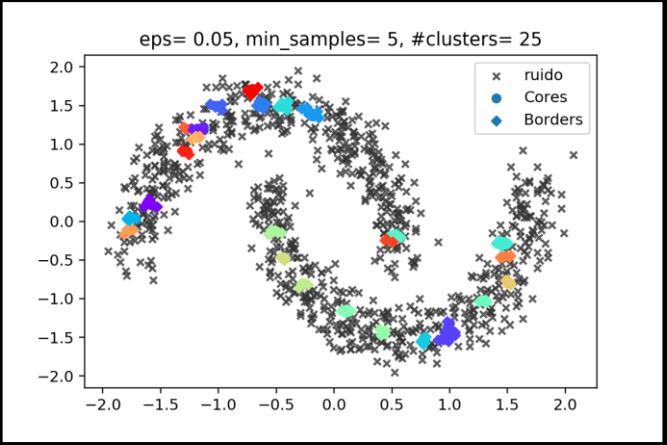
Filtramos los datos detectados como *core samples*.

DBSCAN – hiper-parámetros









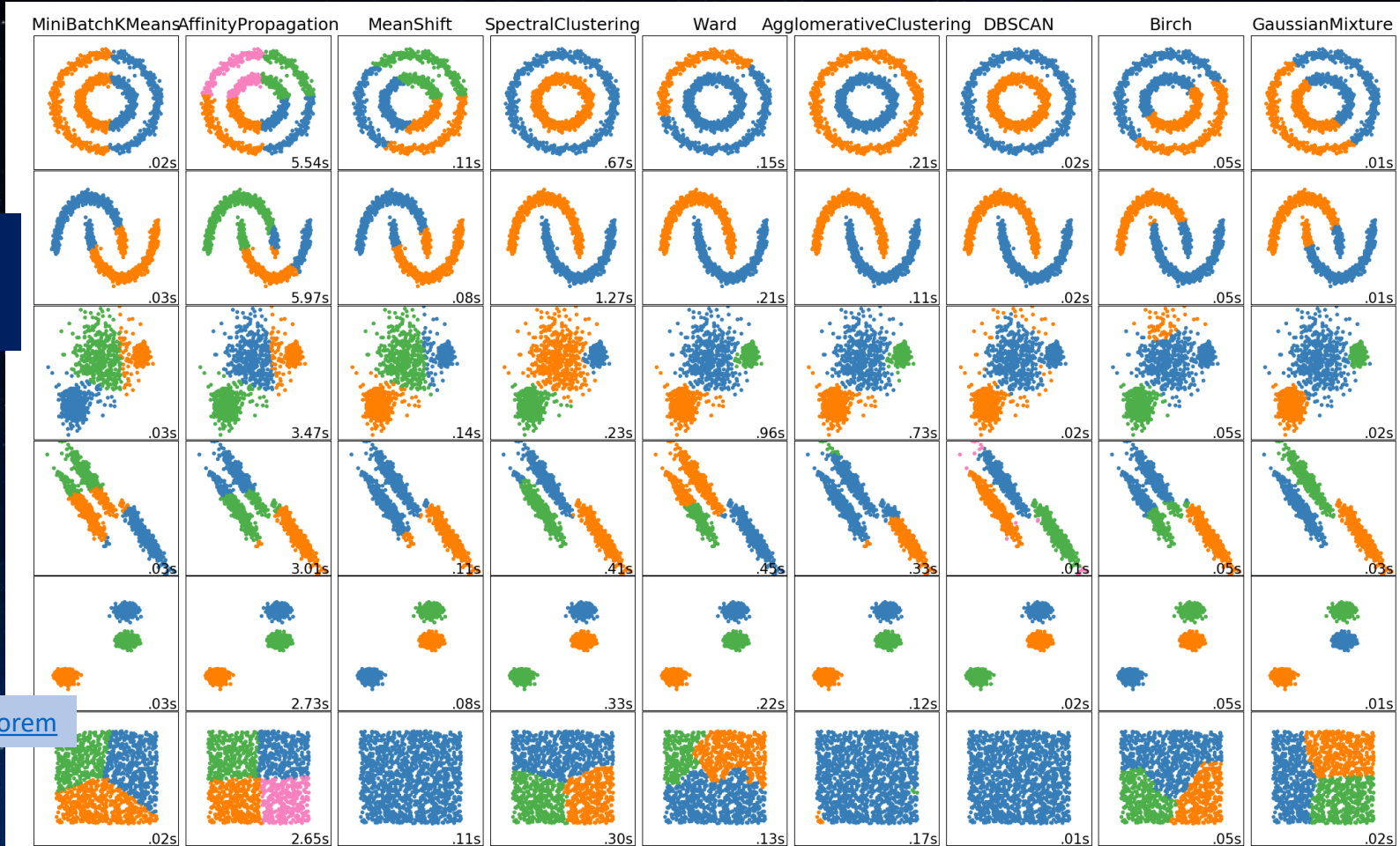
DBSCAN - desventajas

Desventajas:

- Configurar *eps* y *minsamples* no es sencillo. Generalmente requiere experticia del dominio.
- No se adapta bien a clusters de densidad variable, ya que todos los clusters comparten los hiperparámetros *eps* y *minsamples*. HDBSCAN soluciona parcialmente estos problemas.
- Es muy sensible a los hiperparámetros. Pequeños cambios generan cambios drásticos.
- No es muy bueno con datos de alta dimensionalidad.
- La interpretación del índice Silhouette no resulta del todo adecuado, ya que presupone clusters gaussianos.

Diferentes técnicas de agrupamiento

Copiado de la documentación de SKLearn



[No_free_lunch_theorem]