DBSCAN Notebook

June 29, 2018

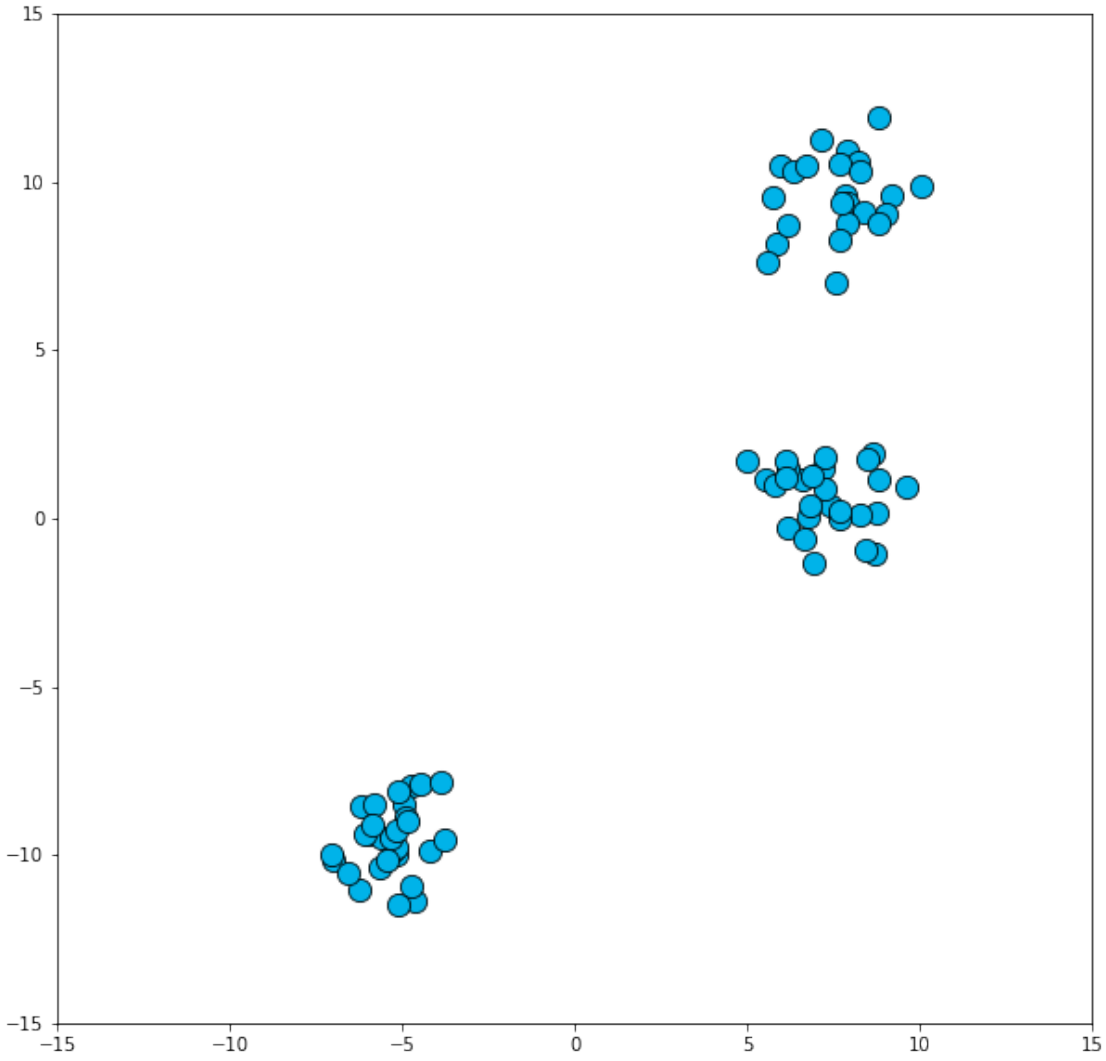
1 DBSCAN Lab

In this notebook, we will use DBSCAN to cluster a couple of datasests. We will examine how changing its parameters (epsilon and min_samples) changes the resulting cluster structure.

```
In [1]: import pandas as pd
        dataset_1 = pd.read_csv('blobs.csv')[:80].values
```

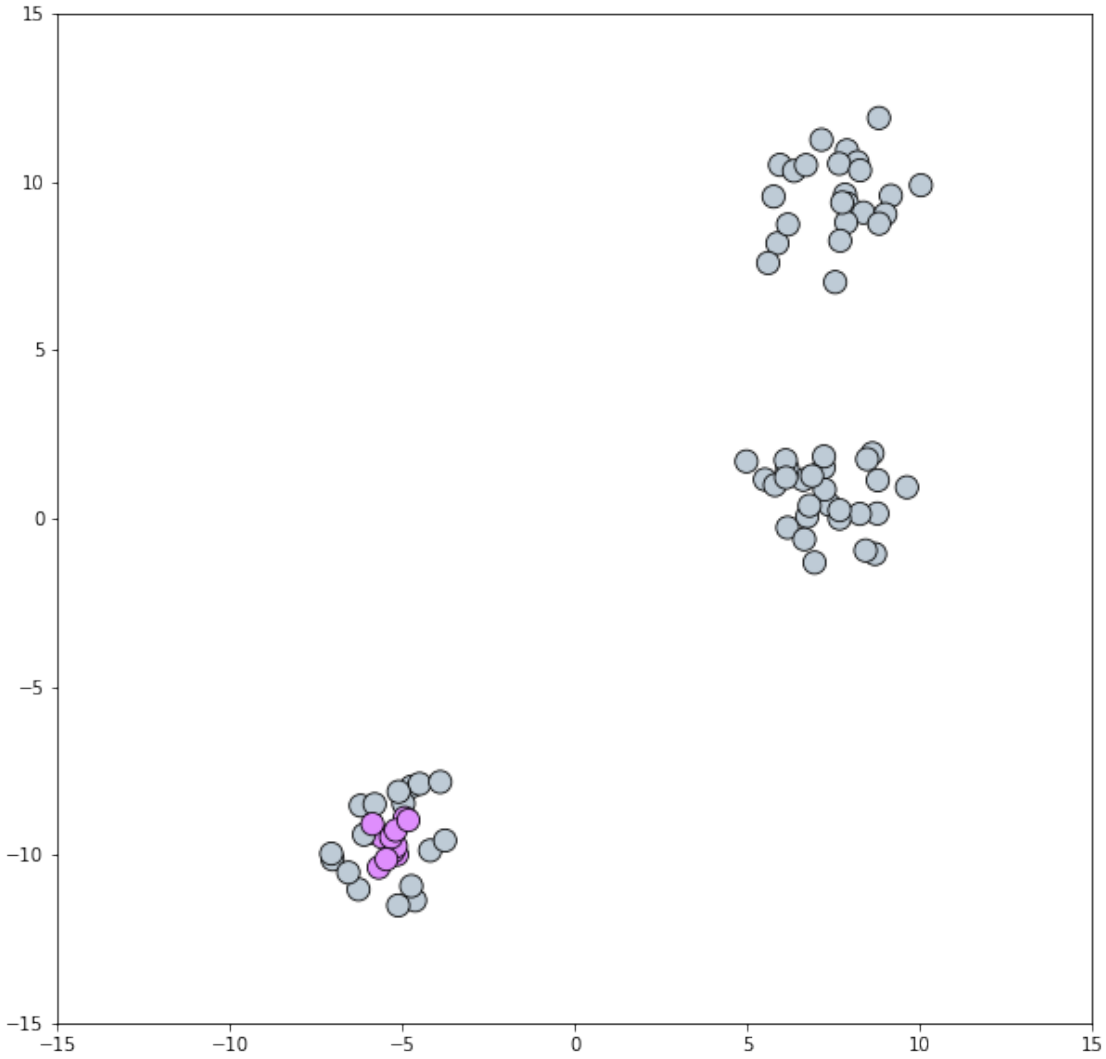
This our first dataset. It looks like this:

```
In [2]: %matplotlib inline

        import dbscan_lab_helper as helper

        helper.plot_dataset(dataset_1)
```



Let's cluster it using DBSCAN's default settings and see what happens. We are hoping for it to be able to assign each of the three "blobs" into its own cluster. Can it do that out of the box?

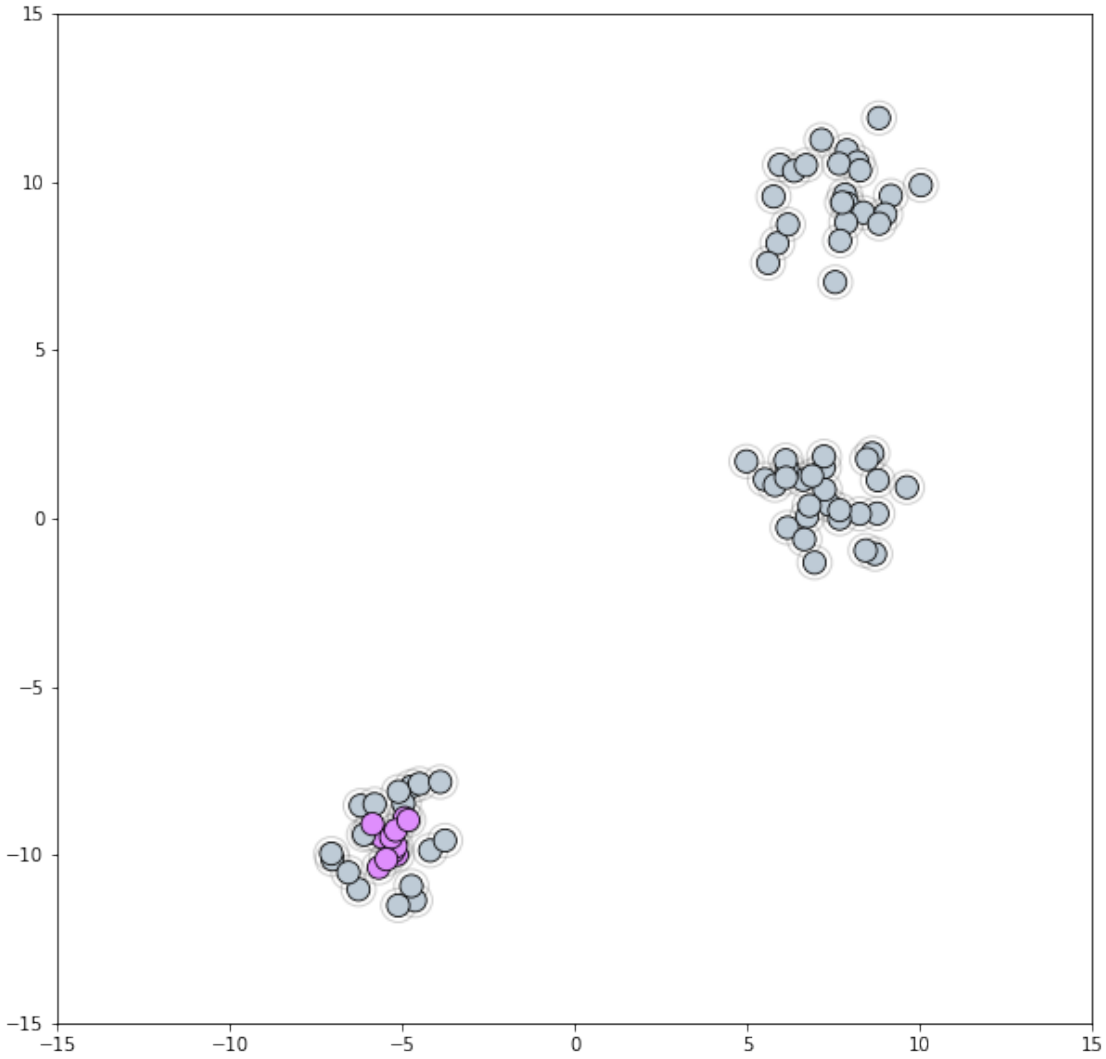
```
In [3]: #TODO: Import sklearn's cluster module
        from sklearn import cluster

        #TODO: create an instance of DBSCAN
        dbscan = cluster.DBSCAN()
        #TODO: use DBSCAN's fit_predict to return clustering labels for dataset_1
        clustering_labels_1 = dbscan.fit_predict(dataset_1)

In [4]: # Plot clustering
        helper.plot_clustered_dataset(dataset_1, clustering_labels_1)
```



Does that look okay? Was it able to group the dataset into the three clusters we were hoping for?

As you see, we will have to make some tweaks. Let's start by looking at Epsilon, the radius of each point's neighborhood. The default value in sklearn is 0.5.

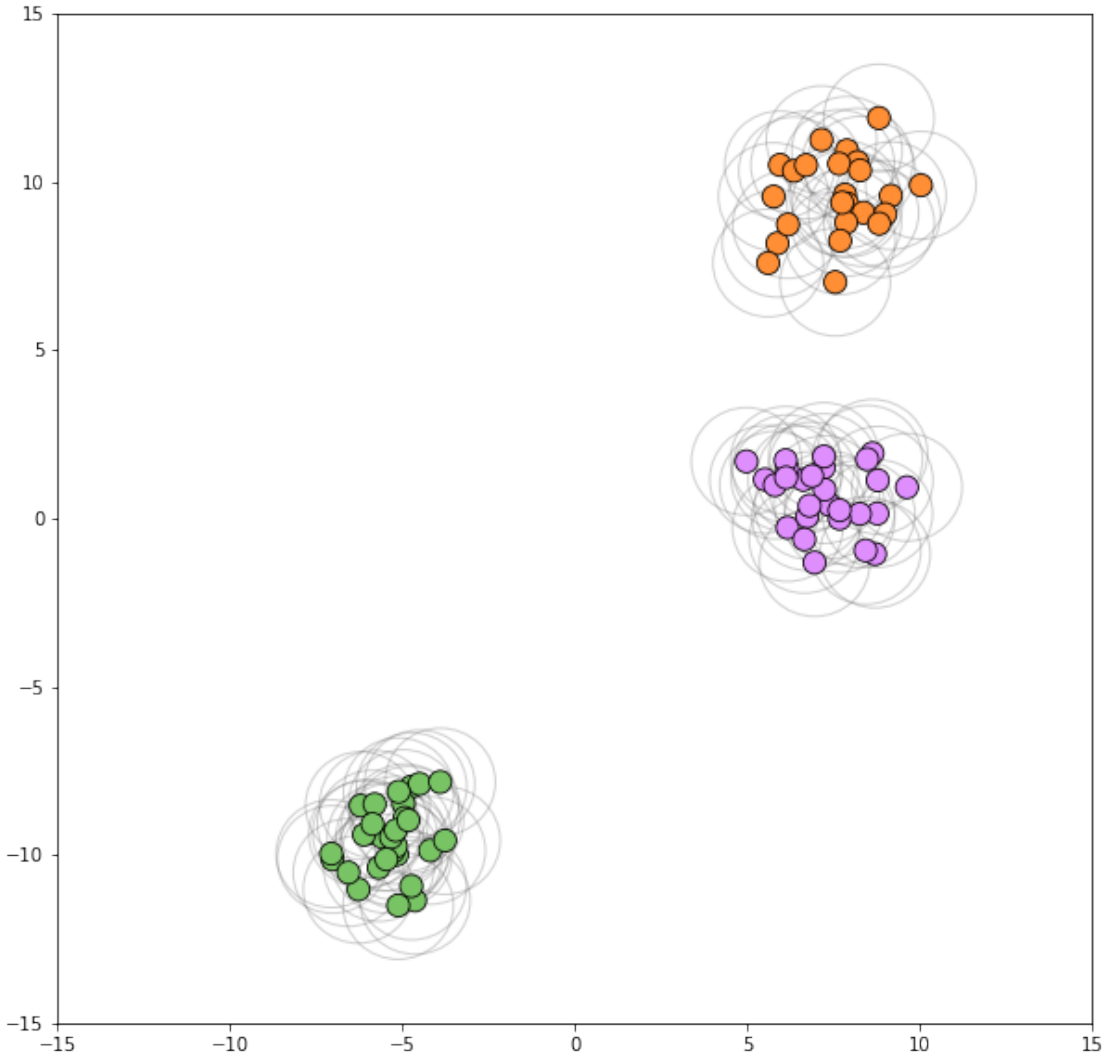
```
In [5]: # Plot clustering with neighborhoods
        helper.plot_clustered_dataset(dataset_1, clustering_labels_1, neighborhood=True)
```



From the graph, we can see that an Epsilon value of 0.5 is too small for this dataset. We need to increase it so the points in a blob overlap each others' neighborhoods, but not to the degree where a single cluster would span two blobs.

Quiz: Change the value of Epsilon so that each blob is its own cluster (without any noise points). The graph shows the points in the datasets as well as the neighborhood of each point:

```
In [9]: # TODO: increase the value of epsilon to allow DBSCAN to find three clusters in the data
        epsilon= 1.6

        # Cluster
        dbscan = cluster.DBSCAN(eps=epsilon)
        clustering_labels_2 = dbscan.fit_predict(dataset_1)

        # Plot
        helper.plot_clustered_dataset(dataset_1, clustering_labels_2, neighborhood=True, epsilon
```

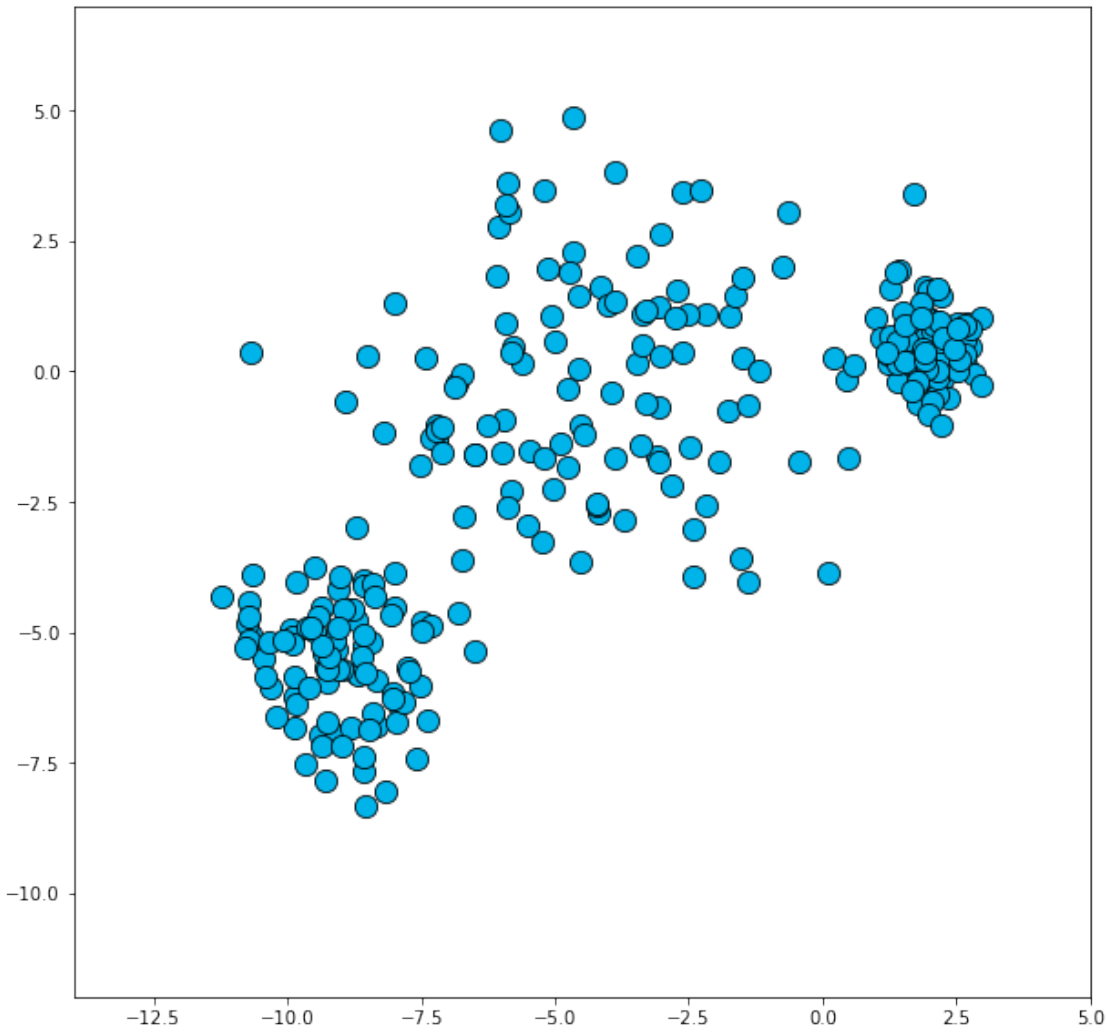


Were you able to do it? As you change the values, you can see that the points cluster into larger clusters and the number of noise points keeps on decreasing. Then at Epsilon values above 1.6 we get the clustering we're after. But once we increase it to above 5, we start to see two blobs joining together into one cluster. So the right Epsilon would be in the range between those values in this scenario.

1.1 Dataset 2

Let's now look at a dataset that's a little more tricky

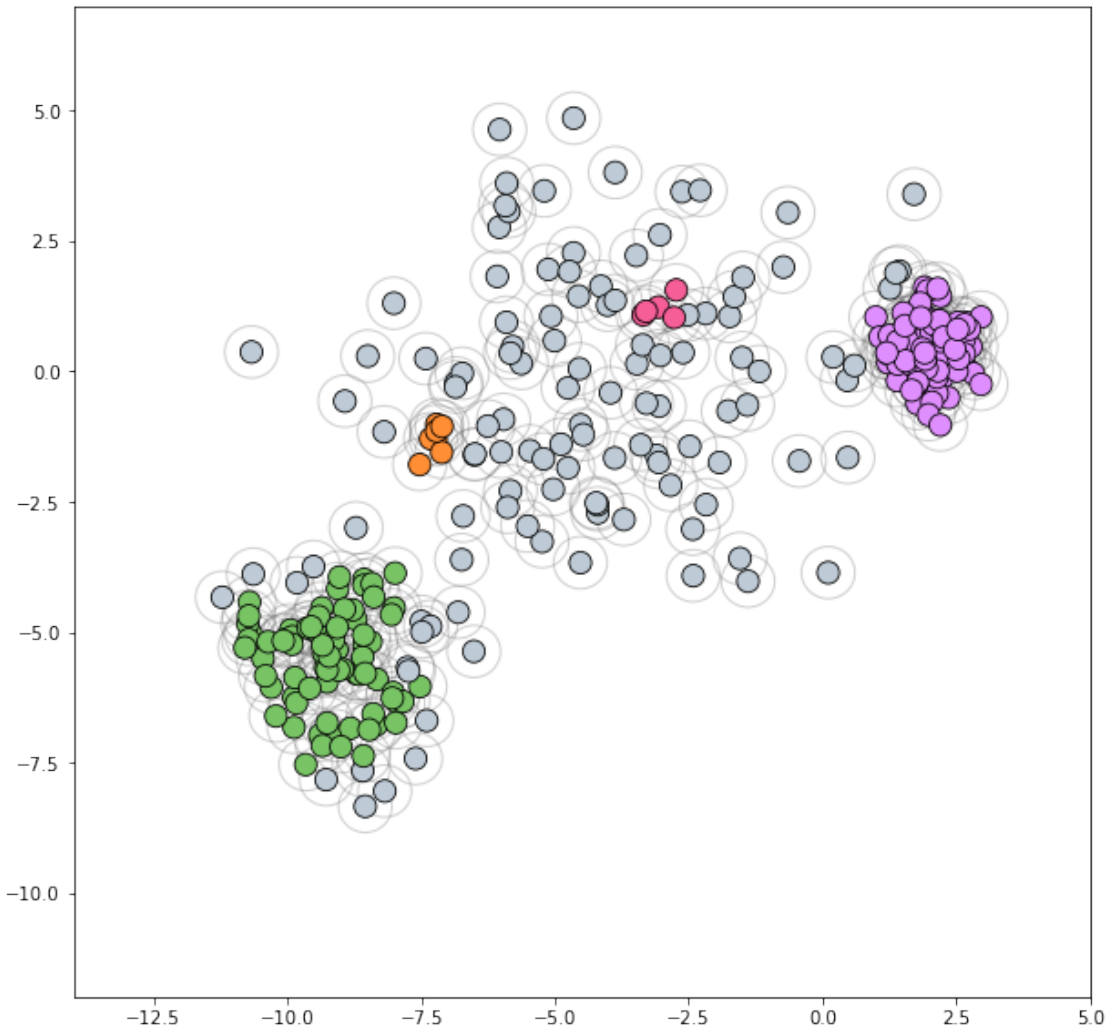
```
In [10]: dataset_2 = pd.read_csv('varied.csv')[:300].values

In [11]: # Plot
         helper.plot_dataset(dataset_2, xlim=(-14, 5), ylim=(-12, 7))
```



What happens if we run DBSCAN with the default parameter values?

```
In [12]: # Cluster with DBSCAN
         # TODO: Create a new isntance of DBSCAN
         dbscan = cluster.DBSCAN()
         # TODO: use DBSCAN's fit_predict to return clustering labels for dataset_2
         clustering_labels_3 = dbscan.fit_predict(dataset_2)
```

```
In [13]: # Plot
         helper.plot_clustered_dataset(dataset_2,
                                       clustering_labels_3,
                                       xlim=(-14, 5),
                                       ylim=(-12, 7),
                                       neighborhood=True,
                                       epsilon=0.5)
```

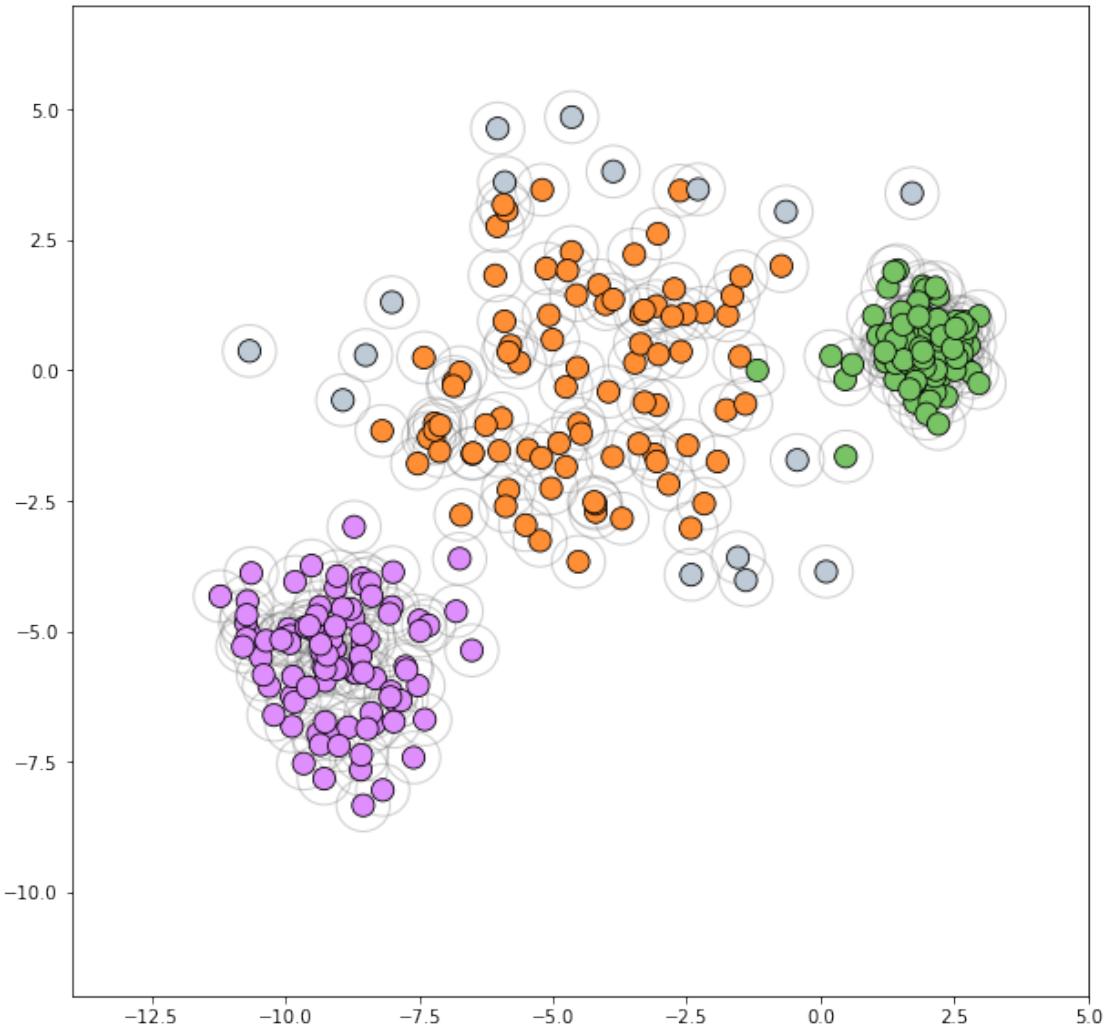


This clustering could make sense in some scenarios, but it seems rather arbitrary. Looking at the dataset, we can imagine at least two scenarios for what we'd want to do: * **Scenario** 1: Break the dataset up into three clusters: the blob on the left, the blob on the right, and the central area (even though it's less dense than the blobs on either side). * **Scenario 2**: Break the dataset up into two clusters: the blob on the left, and the blob on the right. Marking all the points in the center as noise.

What values for the DBSCAN parameters would allow us to satisfy each of those senarios? Try a number of parameters to see if you can find a clustering that makes more sense.

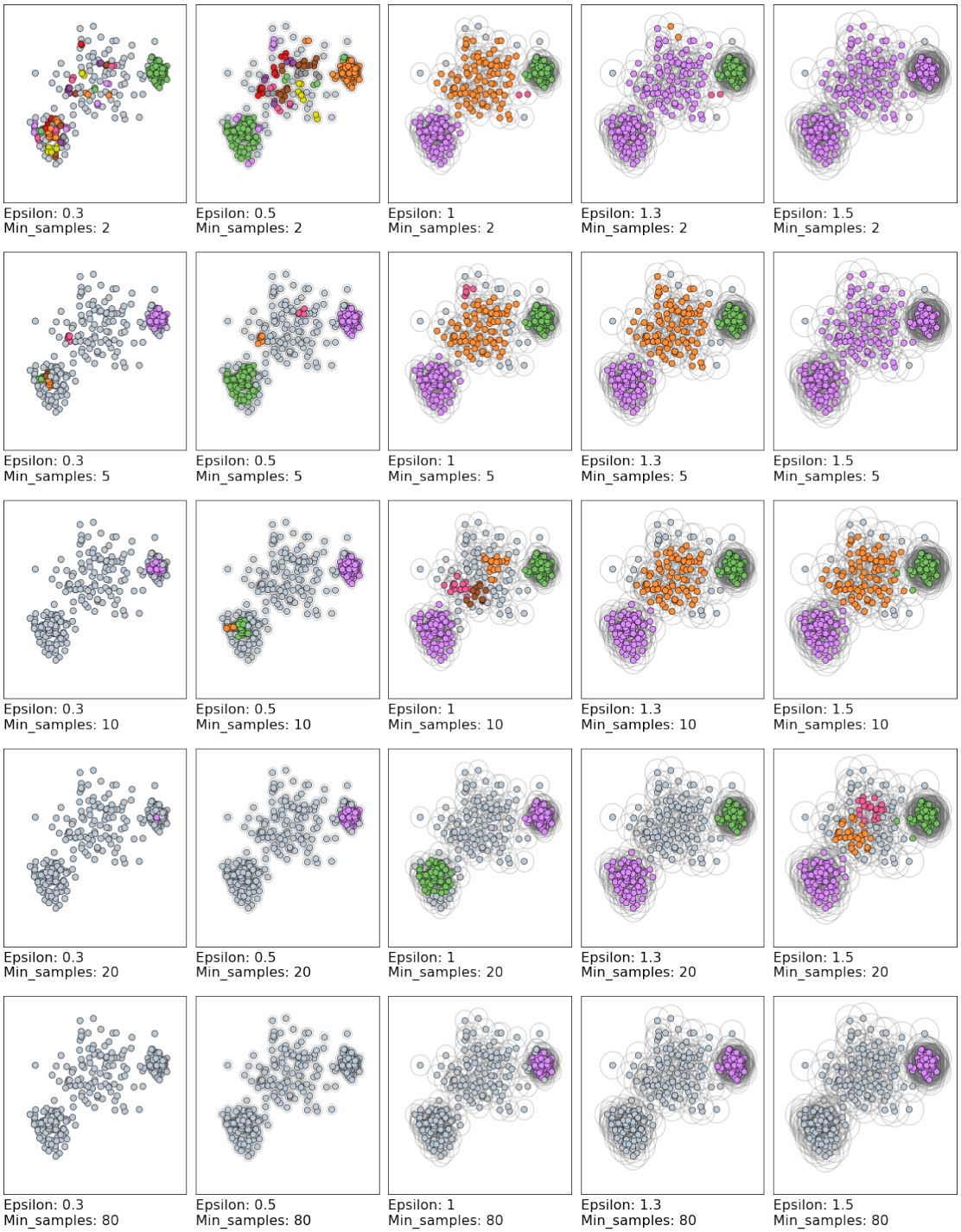
```
In [24]: # TODO: Experiment with different values for eps and min_samples to find a suitable clu
         eps= 1.5
         min_samples= 11

         # Cluster with DBSCAN
         dbscan = cluster.DBSCAN(eps=eps, min_samples=min_samples)
         clustering_labels_4 = dbscan.fit_predict(dataset_2)
```

```
# Plot
helper.plot_clustered_dataset(dataset_2,
                              clustering_labels_4,
                              xlim=(-14, 5),
                              ylim=(-12, 7),
                              neighborhood=True,
                              epsilon=0.5)
```



The following grid plots the DBSCAN clustering results of a range of parameter values. Epsilon varies horizontally, while vertically each row shows a different value of min_samples.

```
In [25]: eps_values = [0.3, 0.5, 1, 1.3, 1.5]
         min_samples_values = [2, 5, 10, 20, 80]

         helper.plot_dbscan_grid(dataset_2, eps_values, min_samples_values)
```



1.2 Heuristics for experimenting with DBSCAN's parameters

Looking at this grid, we can guess at some general heuristics for tweaking the parameters of DBSCAN:

	Epsilon too low	Epsilon too high
min_samples too low	Many small clusters. More than anticipated for the dataset. **Action**: increase min_samples and epsilon	Most points belong to one cluster**Action**: decrease epsilon and increase min_samples
min_samples too high	Most/all data points are labeled as noise**Action**: increase epsilon and decrease min_sample	Except for extremely dense regions, most/all data points are labeled as noise. (Or all points are labeled as noise). **Action**: decrease min_samples and epsilon.

1.2.1 Quiz

- Which values do you believe best satisfy scenario 1?
- Which values do you believe best satisfy scenario 2?