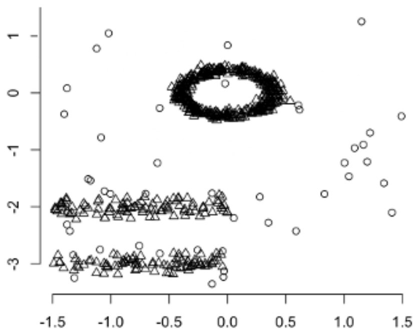
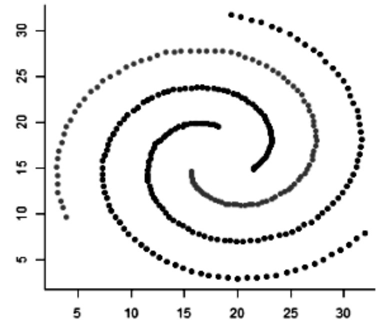
Case Study 1:

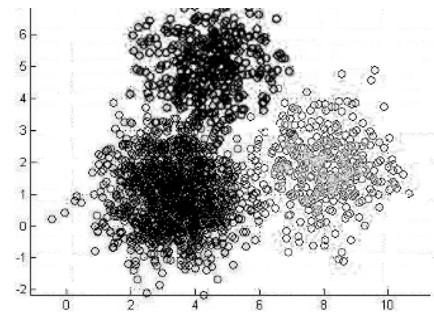
1) Given the following datasets:



(A)



(B)



(C)

I. If we want to apply clustering techniques on *each* dataset, would it be better to apply *k-means* or *DBSCAN*? And explain why?

Answer:

- (A) DBSCAN is better.
- (B) DBSCAN is better.
- (C) K-means or DBSCAN.
- Because DBSCAN doesn't assume a cluster shape, while Kmeans is suitable for spherical clusters.

II. In figure (A), you can observe some noise in the dataset.

- Which *step(s)* in the typical Data Science process will help to identify and fix this noise?
- Briefly explain each step.
- Clearly indicate the order of the *step(s)* as part of your answer.

Answer:

- Data preparation / data exploration
- Order: Data preparation / data exploration
- There should be detailed explanation for each step

2) Suppose you have a data set that includes two *categorical* and three *numerical* columns. (If you don't know the name, you can sketch an example picture.)

i) Name two kinds of graphs that can be used to visualise *categorical* data

Answer: Barplot, pie chart

ii) Propose a simple analysis to explore the relationship between a *categorical* and a *numerical* column.

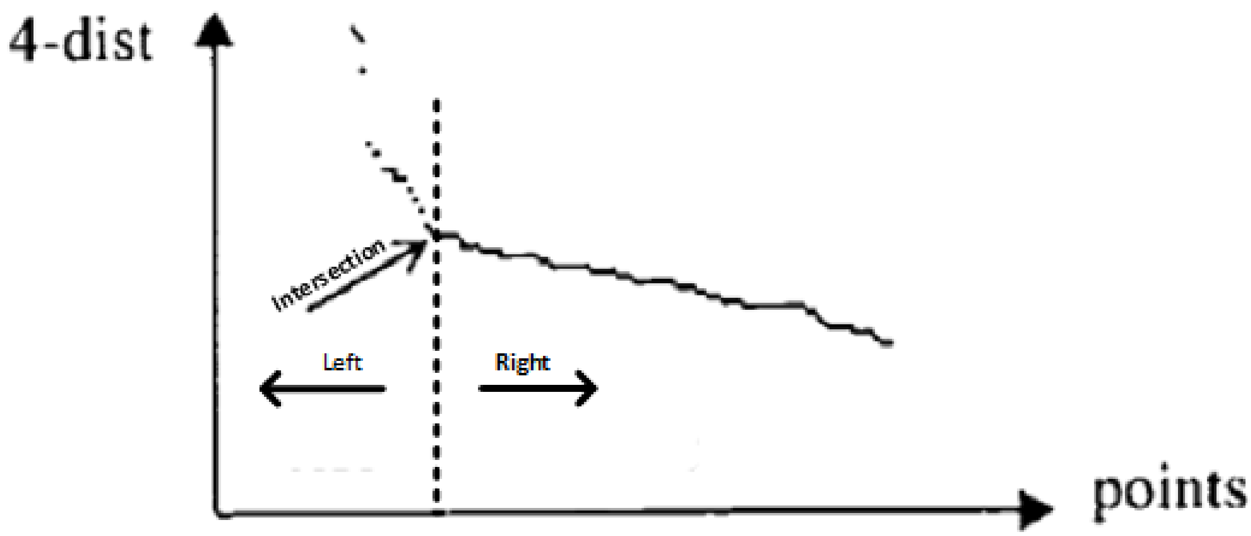
Answer: Boxplot by category

iii) Propose a simple analysis to explore the relationship between two *numerical* columns.

Answer: Scatterplot

Case Study 2:

The following figure shows the k *distance graph* for a DBSCAN algorithm where $minPts$ is equal to 4:



Please answer

1. What is the meaning of the *intersection* point in the graph. How can we use it in the DBSCAN algorithm?

   Answer:

   - The threshold point represents the value of the Eps parameter. Eps is the maximum distance required to consider a neighbour point.

   - Good values of Eps are where this plot shows a strong bend.

2. What is the meaning of the points to the *Left* of the dotted line in this graph?

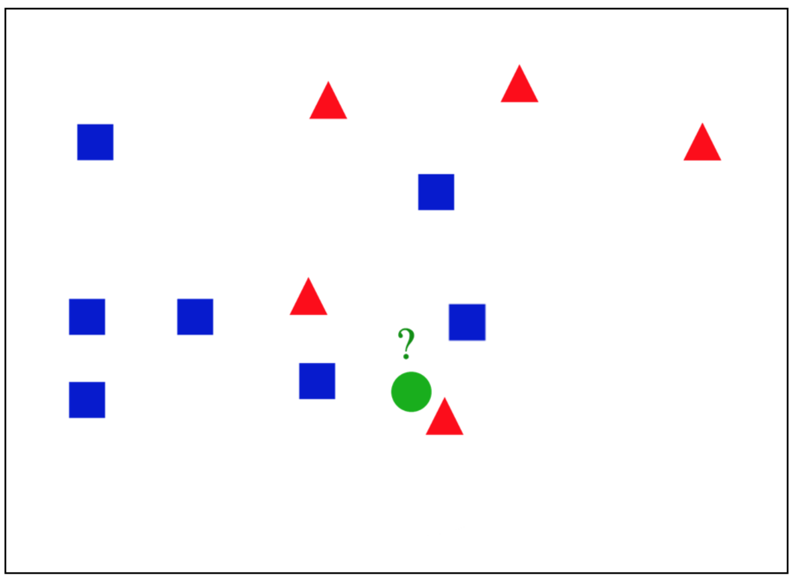
Answer: Noise. If Eps is chosen less than the noise value, a large part of the data will not be clustered and will be considered noise

3. What is the meaning of the points to the *Right* of the dotted line in this graph?

   Answer: Clusters. If Eps is chosen greater than the noise value, the model will detect clusters. However, a too high value of Eps will merge and the majority of objects will be in the same cluster.

Case Study 3:

Apply *k-nearest neighbours* (*sklearn.neighbors.KNeighborsClassifier*) classifier on the following given data:



There are two classes: *squares* and *triangles*. For the given test sample (the *circle* dot , shown with a question mark), answer the following question:

1. When we set *k = 3* and *weights = uniform*, which class the test sample should be classified to? And Why?

   Answer:
   - Square
   - because it's the majority vote for k=3, two squares to one triangle
2. When we set *k = 2* and *weights = distance,* which class the test sample should be classified to? And Why?

   Answer:

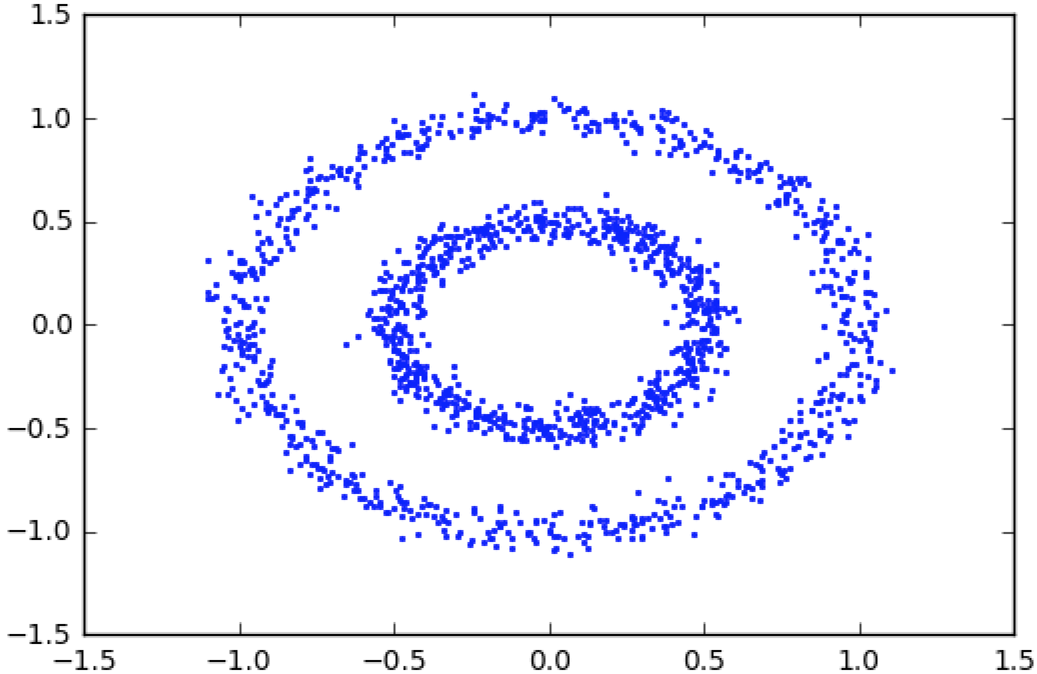
- Triangle
- Because each vote is multiplied by 1/distance. So, the red triangle is closer

Practical exercise 1: Data Retrieving

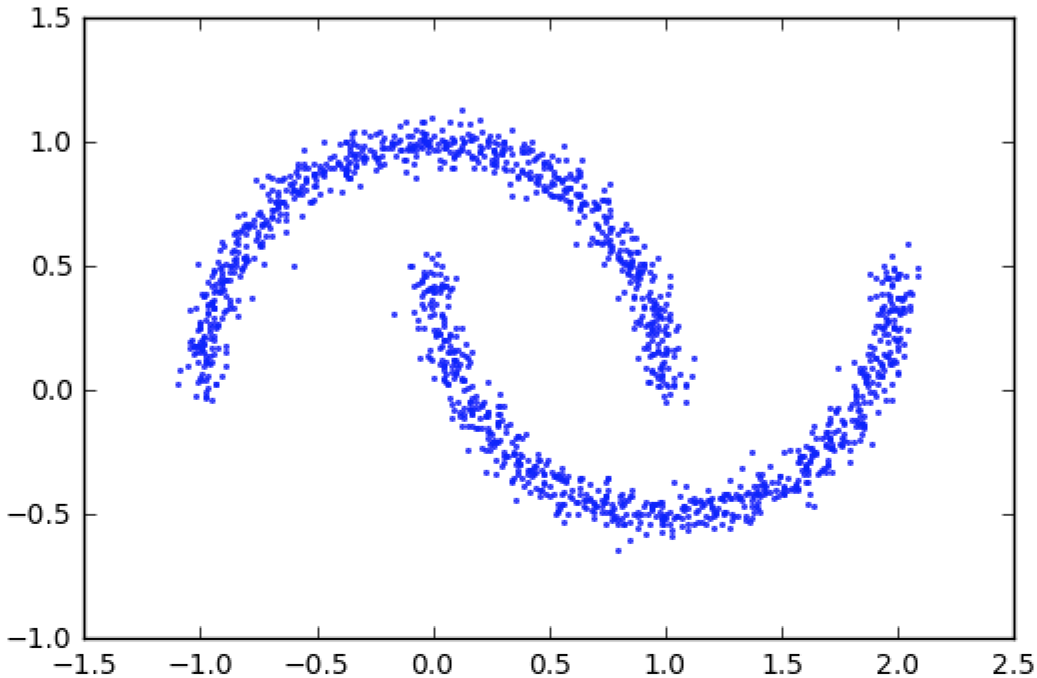
This week, let's explore the DBSCAN on several datasets with different shapes of clusters.

First, let's load the data sets.

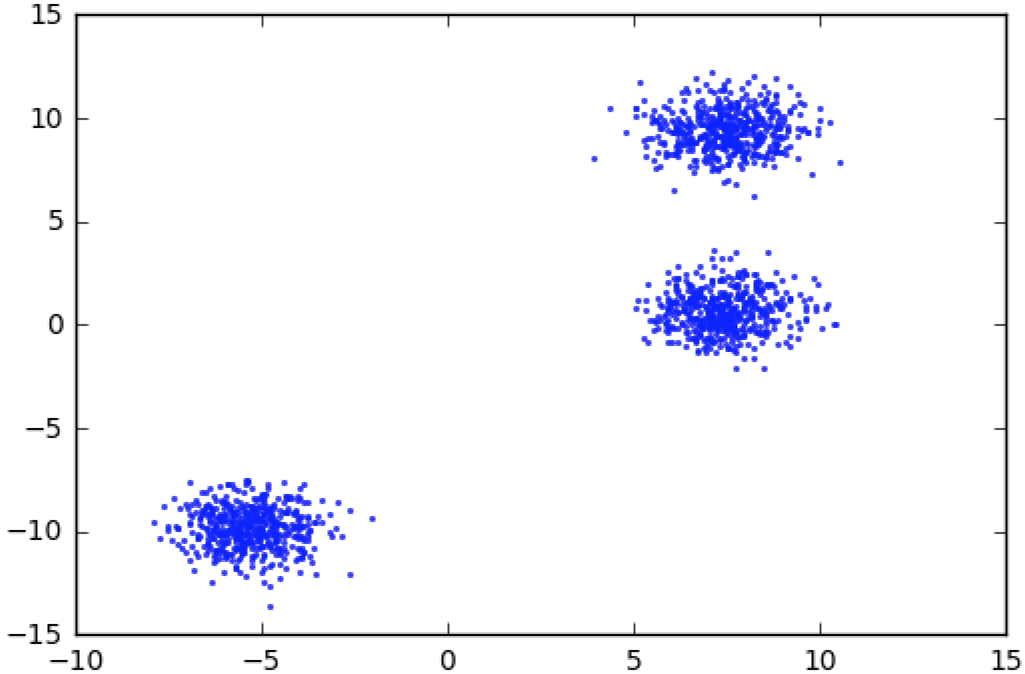
```
In [1]: import numpy as np
In [2]: import matplotlib.pyplot as plt
In [3]: from sklearn import cluster, datasets

In [4]: n_samples = 1500
In [5]: circles = datasets.make_circles(n_samples=n_samples, factor=.5,
                                        noise=.05)
In [6]: moons = datasets.make_moons(n_samples=n_samples, noise=.05)
In [7]: blobs = datasets.make_blobs(n_samples=n_samples, random_state=8)

In [8]: circles
Out [8]: (array([[-0.21009776,  0.46181291],
        [ 0.67629075, -0.68385711],
        [ 0.87536459,  0.5651477 ],
        ...,
        [-0.0331735 , -0.46811284],
        [-0.96187452,  0.27081349],
        [-0.21301649, -0.51512252]]), array([1, 0, 0, ..., 1, 0, 1]))

In [9]: plt.scatter(circles[0][:, 0], circles[0][:, 1], alpha = 0.8, s= 5.0,
lw= 0)
In [10]: plt.show()
Out [10]:
```




```
In [11]: moons
Out [11]: (array([[ 0.04454316,  0.0845715 ],
        [ 1.58566568, -0.26181606],
        [ 0.04062815,  0.50069292],
        ...,
        [ 0.73273691,  0.51027295],
        [ 0.57784829, -0.34139774],
        [-0.95526833, -0.03978616]]), array([1, 1, 1, ..., 0, 1, 0]))

In [12]: plt.scatter(moons[0][:, 0], moons[0][:, 1], alpha = 0.8, s= 5.0, lw=
0)
In [13]: plt.show()
Out [13]:
```




```
In [14]: blobs
Out [14]: (array([[  5.86749807,   8.17715188],
        [  5.61369982,   9.93295527],
        [  7.22508428,  10.44886194],
        ...,
        [  7.73674097,  10.82855388],
        [ -4.61701094,  -9.64855983],
        [ -3.48640175,  -9.25766922]]), array([0, 0, 0, ..., 0, 2, 2]))
In [15]: plt.scatter(blobs[0][:, 0], blobs[0][:, 1], alpha = 0.8, s= 5.0, lw=
0)
In [16]: plt.show()
Out [16]:
```



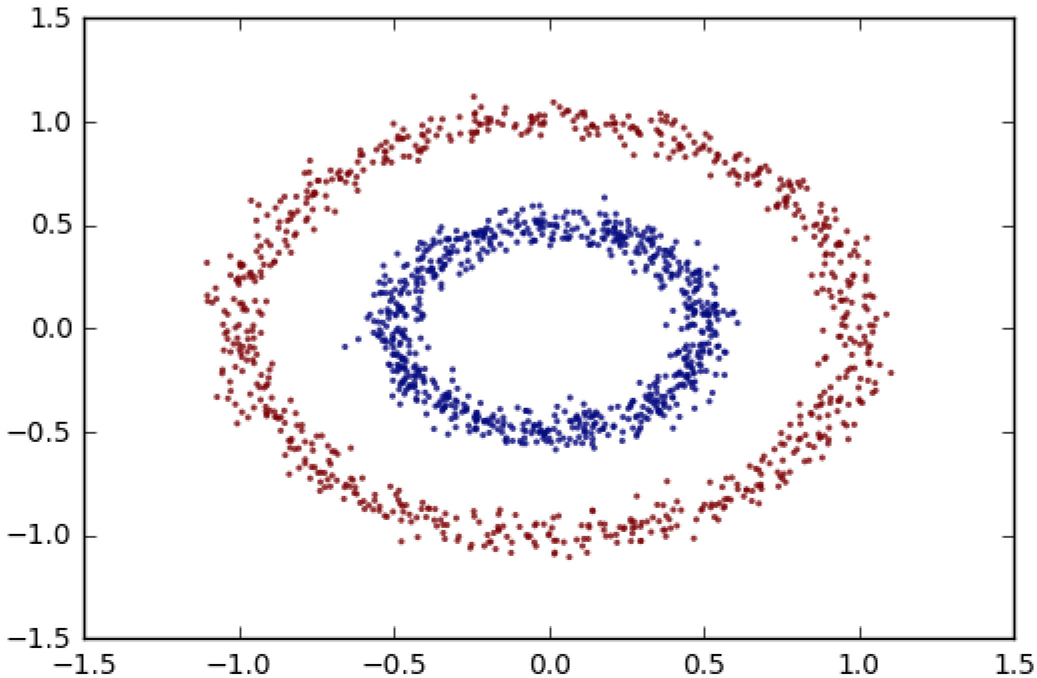
Practical exercise 2: *DBSCAN* Clustering

Next, we would like to build the *DBSCAN* model for each dataset.

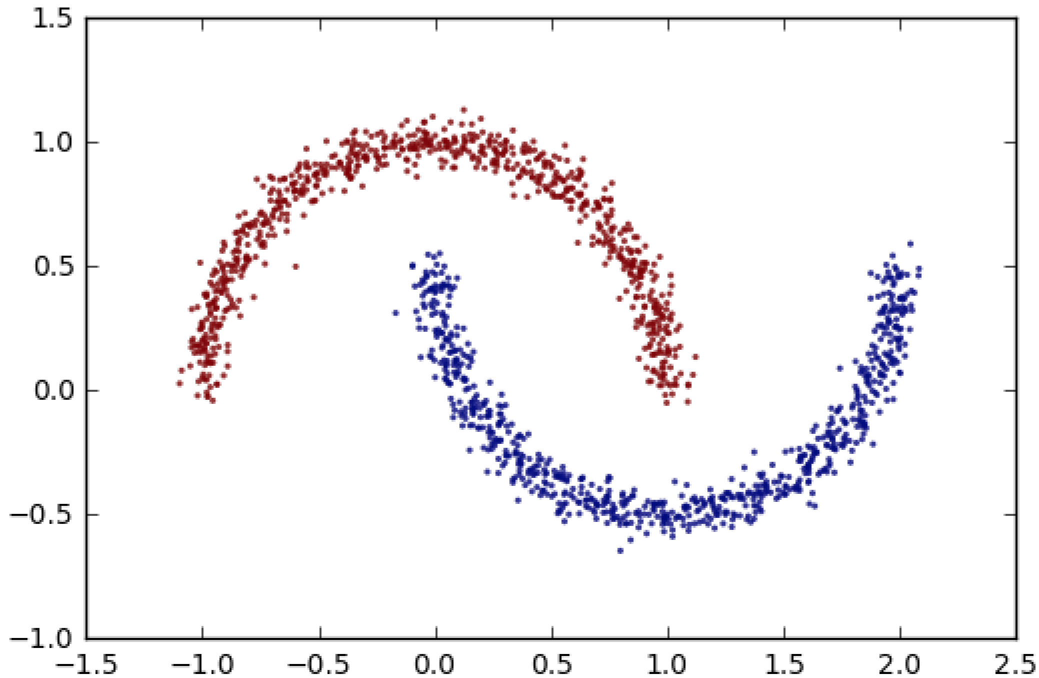
```
In [17]: dbs_1 = cluster.DBSCAN(eps=.2)
In [18]: dbs_fit = dbs_1.fit(circles[0])
```

Please think about why we use *circles[0]* here? Hint: please look at Out [8].

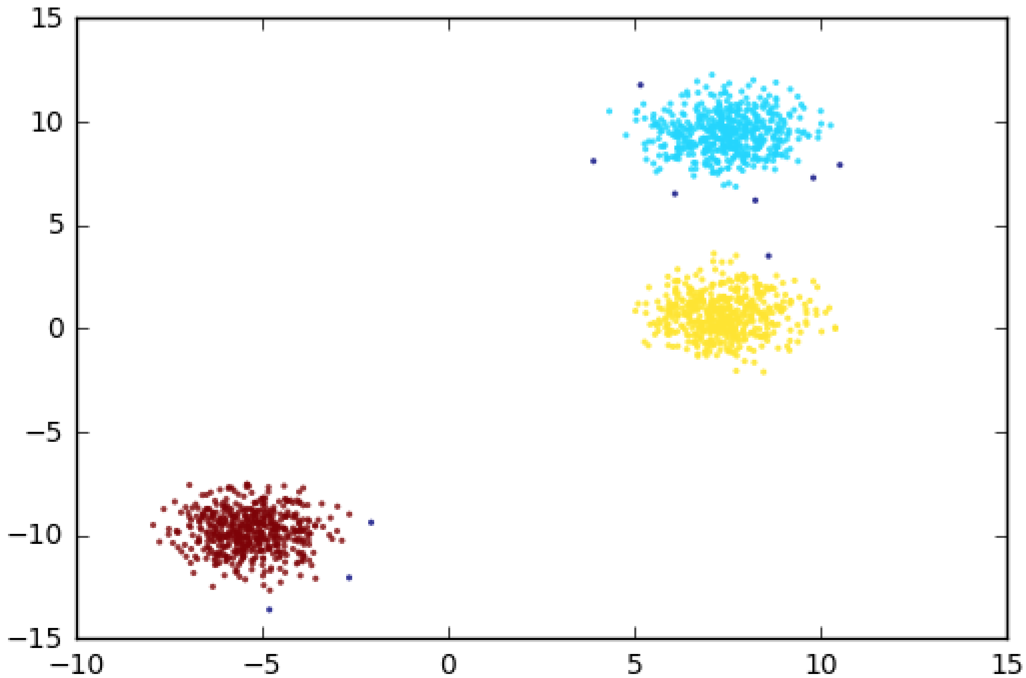
```
In [19]: labels_1 = dbs_fit.labels_
In [20]: plt.scatter(circles[0][:, 0], circles[0][:, 1], c=labels_1, alpha =
0.8, s= 5.0, lw= 0)
In [21]: plt.show()
Out [21]:
```




```
In [22]: dbs_2 = cluster.DBSCAN(eps=.2)
In [23]: dbs_fit = dbs_2.fit(moons[0])
In [24]: labels_2 = dbs_fit.labels_
In [25]: plt.scatter(moons[0][:, 0], moons[0][:, 1], c=labels_2, alpha = 0.8,
s= 5.0, lw= 0)
In [26]: plt.show()
Out [26]:
```



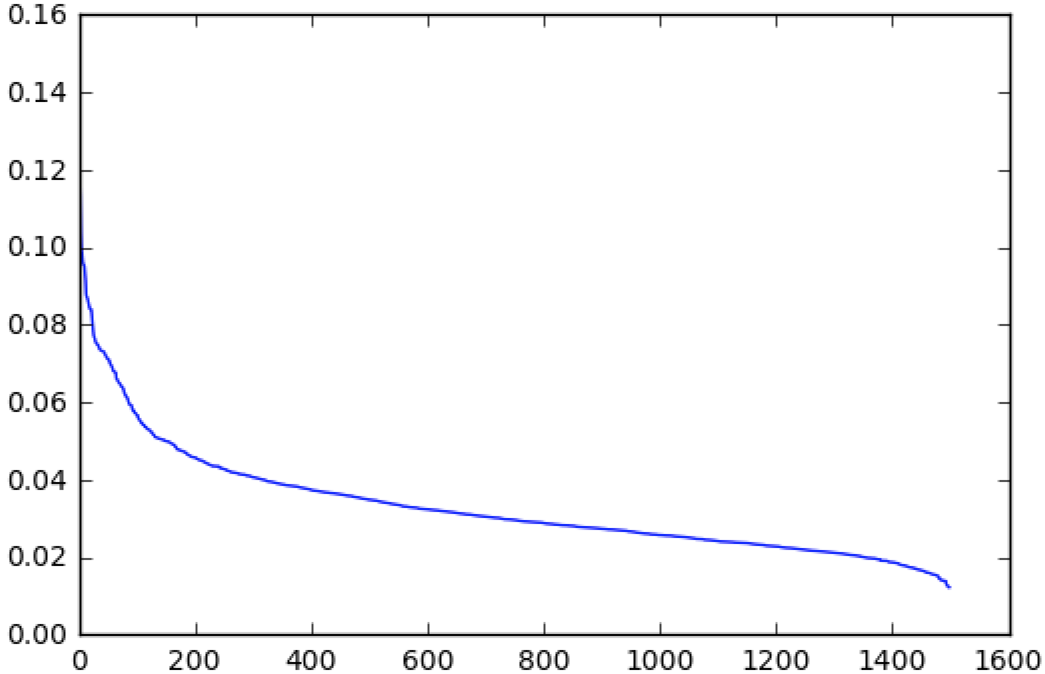

```
In [27]: dbs_3 = cluster.DBSCAN(eps=.8)
In [28]: dbs_fit = dbs_3.fit(blobs[0])
In [29]: labels_3 = dbs_fit.labels_
In [30]: plt.scatter(blobs[0][:, 0], blobs[0][:, 1], c=labels_3, alpha = 0.8,
s= 5.0, lw= 0)
In [31]: plt.show()
Out [31]:
```



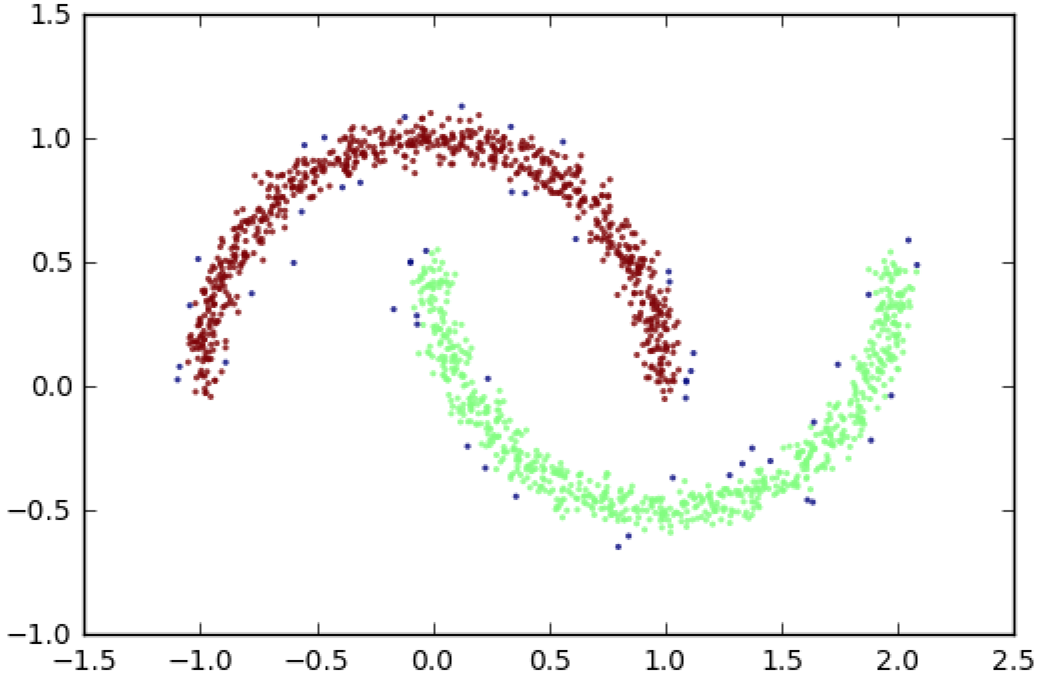
Practical exercise 3: k distance graph

We explore the k-distance graph for the moons dataset:

```
In [32]: from sklearn.neighbors import NearestNeighbors
In [33]: nbrs = NearestNeighbors().fit(moons[0])
In [34]: distances, indices = nbrs.kneighbors(moons[0], 20)
In [35]: kDis = distances[:, 4]
In [36]: kDis.sort()
In [37]: kDis = kDis[::-1]
In [38]: plt.plot(range(0,len(kDis)), kDis)
In [39]: plt.show()
Out [39]:
```




```
In [40]: dbs_2 = cluster.DBSCAN(eps=.05)
In [41]: dbs_fit = dbs_2.fit(moons[0])
In [42]: labels_2 = dbs_fit.labels_
In [43]: plt.scatter(moons[0][:, 0], moons[0][:, 1], c=labels_2, alpha = 0.8,
s= 5.0, lw= 0)
In [44]: plt.show()
Out [44]:
```



Please compare `Out [44]` with `Out [26]`, which was generated using the parameter $Eps = 0.2$.

- What are the differences?
- Why could this happen?

Extra Exercise:

Please go to Canvas -> Modules -> Week 8 -> week8-Lectorial.zip, then unzip the zip file and review Week8.ipynb. If you have any questions, please ask your Lab demonstrator.