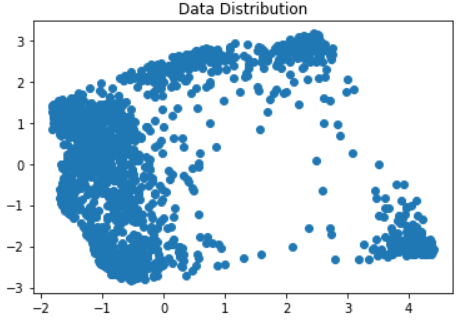
**1. Why this algorithm is suitable for this dataset? [3 marks]**

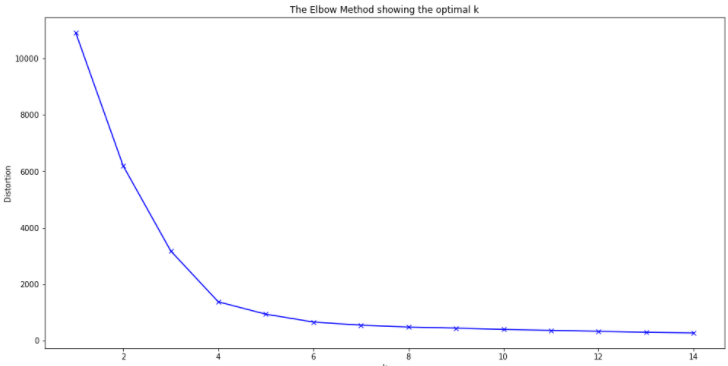
Since the data set is unlabeled, I need an unsupervised learning method. By plotting the data set, I can see that the points of this data set are very dense. So, I used the DBSCAN clustering method, which is a density-based clustering algorithm and it does a good job of finding areas of data with a high density of observations. By observation, there are many noise points around the center of the graph, the DBSCAN algorithm has strong robustness to abnormal points (noise points). Unlike K-means, DBSCAN does not need to set the number of clusters priority. And the graph is clearly not globular, DBSCAN supports non-globular structures.

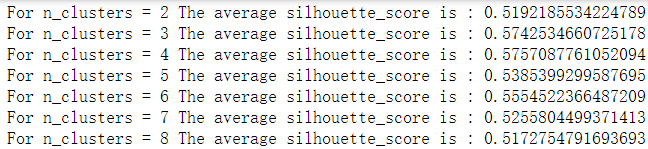
The above advantages make the algorithm suitable for this data set.

**2. How do you determine the number of clusters or other parameters/Python function arguments? [3 marks]**

To determine the number of clusters, I used both elbow method and K means algorithm along with Silhouette distance. A high silhouette score indicates that the objects match their own clusters very well but match their neighboring clusters poorly.

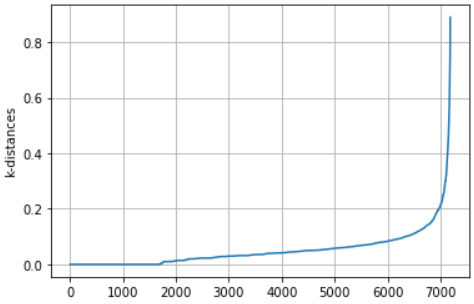
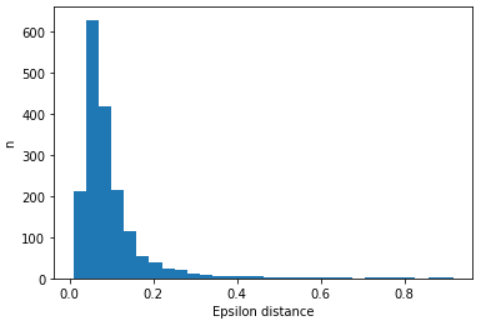
From the result below, it suggests number of clusters to be 4.



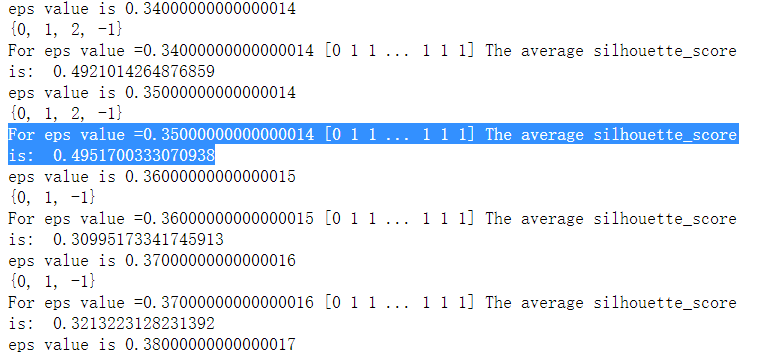


Then to determine the two parameters of DBSCAN:

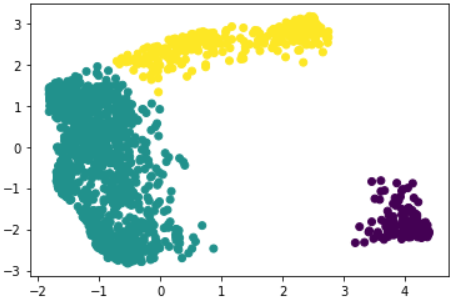
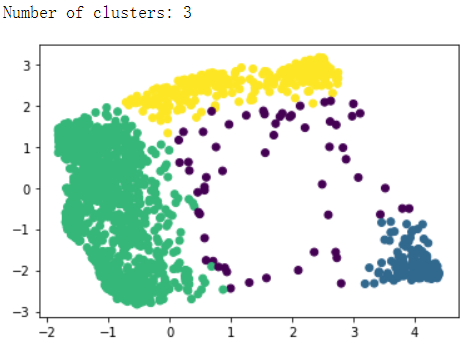
eps: The value of eps can be selected by using the k-distance graph, drawn to the distance k = minPts-1 (here we assume that minPts is 4, which can be changed later), and the order of the nearest neighbors is from largest to smallest. The good value of eps shows an "elbow" in the figure: if eps is selected too small, a large part of the data will not be clustered; however, if the eps value is too high, the clusters will merge and most objects will be in the same cluster. In general, the smaller the eps value, the better. According to experience, only a small part of the points should be within this distance of each other. We plot the elbow method to determine the eps and see it will range from 0.2 to 0.4.

MinPts: As a rule of thumb, the minimum minPts can be derived from the dimension D of the data set, because minPts≥D+1. At least 3 minPts must be selected. However, for noisy data sets, larger values are usually better and will produce more significant clusters. Our data set is quite large and contains a lot of noise, so we can try minPts in the range of 15. By trying different number of minPts with different eps values from 0.2 to 0.4, I got Silhouette Coefficient: 0.495 with eps=0.25 and minPts=7 as our result.



The cluster result look like this:



The dataset has been classified into three cluster with 59 noises in the middle.

If we don’t count in the noise, we can get Silhouette Coefficient as 0.587, which is fairly good clustering result.