DBScan

1. **Generate sample data**
   - First, I use make_blobs of scikit-learn to generate X, which is an array of n_samples points 2D, all of which have x-axis and y-axis are random numbers between -2 and 2

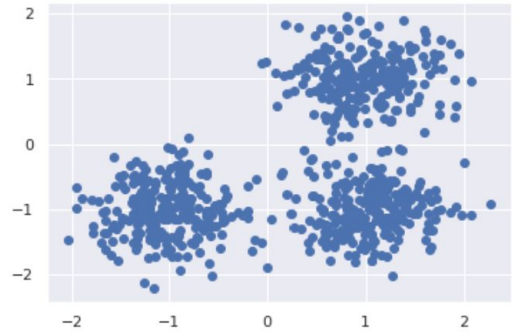
```
[ ]  centers = [[1, 1], [-1, -1], [1, -1]]
     X, labels_true = make_blobs(n_samples=750, centers=centers, cluster_std=0.4,
                                 random_state=0)
     X
```

```
array([[ 0.84022039,  1.14802236],
       [-1.15474834, -1.2041171 ],
       [ 0.67863613,  0.72418009],
       ...,
       [ 0.26798858, -1.27833405],
       [-0.88628813, -0.30293249],
       [ 0.60046048, -1.29605472]])
```

   - To make it easier to observe, I plot all of those points on a 2D coordinate plane and get the result like this:

```
[ ]  plt.scatter(X[:,0], X[:,1])
```

```
<matplotlib.collections.PathCollection at 0x7fe42ea3e898>
```



2. **Find out about DBScan algorithm for clustering**
   - We have to notice two parameters:
     + **eps**: Two points are considered neighbors if the distance between the two points is below the threshold epsilon.
     + **min_samples**: The minimum number of neighbors a given point should have in order to be classified as a core point.
   - The DBSCan algorithm works by computing the distance between every point and all other points. We then place the points into one of three categories.
     + Core point: A point with at least min_samples points whose distance with respect to the point is below the threshold defined by epsilon.
     + Border point: A point that isn't in close proximity to at least min_samples points but is close enough to one or more core point. Border points are included in the cluster of the closest core point.
     + Noise point: Points that aren't close enough to core points to be considered border points. Noise points are ignored. That is to say, they aren't part of any cluster.
3. **Find the optimal value for eps**

- I calculate the distance from each point to its closest neighbour using the NearestNeighbors. The point itself is included in n_neighbors. The kneighbors method returns two arrays, one which contains the distance to the closest n_neighbors points and the other which contains the index for each of those points.

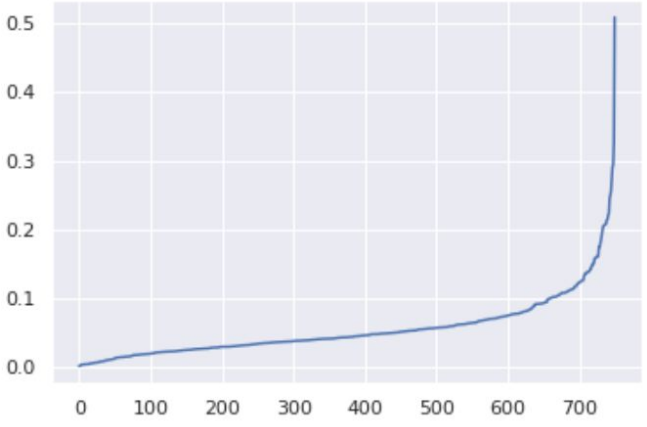
```
[ ]  neigh = NearestNeighbors(n_neighbors=2)
     nbrs = neigh.fit(X)
     distances, indices = nbrs.kneighbors(X)
     distances
```

```
array([[0.        , 0.05142838],
       [0.        , 0.02659756],
       [0.        , 0.05183912],
       ...,
       [0.        , 0.07794371],
       [0.        , 0.0495315 ],
       [0.        , 0.05001771]])
```

- Next, I sort and plot results.

```
[ ]  distances = np.sort(distances, axis=0)
     distances = distances[:,1]
     plt.plot(distances)
```

```
[<matplotlib.lines.Line2D at 0x7fe42e992a58>]
```



- The optimal value for eps will be found at the point of maximum curvature (which is approximately 0.13)

4. **Train my model, selecting 0.13 for eps and setting min_samples to 6.**
   - This is result of estimating number of clusters, noise points, ...

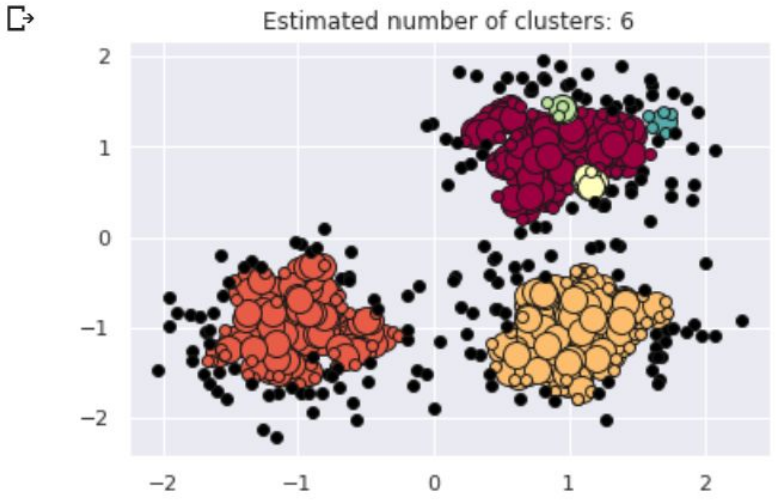
```
db = DBSCAN(eps=0.13, min_samples=6).fit(X)
core_samples_mask = np.zeros_like(db.labels_, dtype=bool)
core_samples_mask[db.core_sample_indices_] = True
labels = db.labels_

# Number of clusters in labels, ignoring noise if present.
n_clusters_ = len(set(labels)) - (1 if -1 in labels else 0)
n_noise_ = list(labels).count(-1)

print('Estimated number of clusters: %d' % n_clusters_)
print('Estimated number of noise points: %d' % n_noise_)
print("Homogeneity: %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))
```

```
Estimated number of clusters: 6
Estimated number of noise points: 178
Homogeneity: 0.763
Completeness: 0.563
V-measure: 0.648
Adjusted Rand Index: 0.589
Adjusted Mutual Information: 0.645
Silhouette Coefficient: 0.179
```

- Plot the result:



Reference:

[1]: Code in my Google Colab: https://colab.research.google.com/drive/1GAXAr0nhVldBUDrNVpvtgjMtWX0-xk0i#scrollTo=qSi3znMeIXAf

[2]: https://towardsdatascience.com/machine-learning-clustering-dbscan-determine-the-optimal-value-for-epsilon-eps-python-example-3100091cfbc