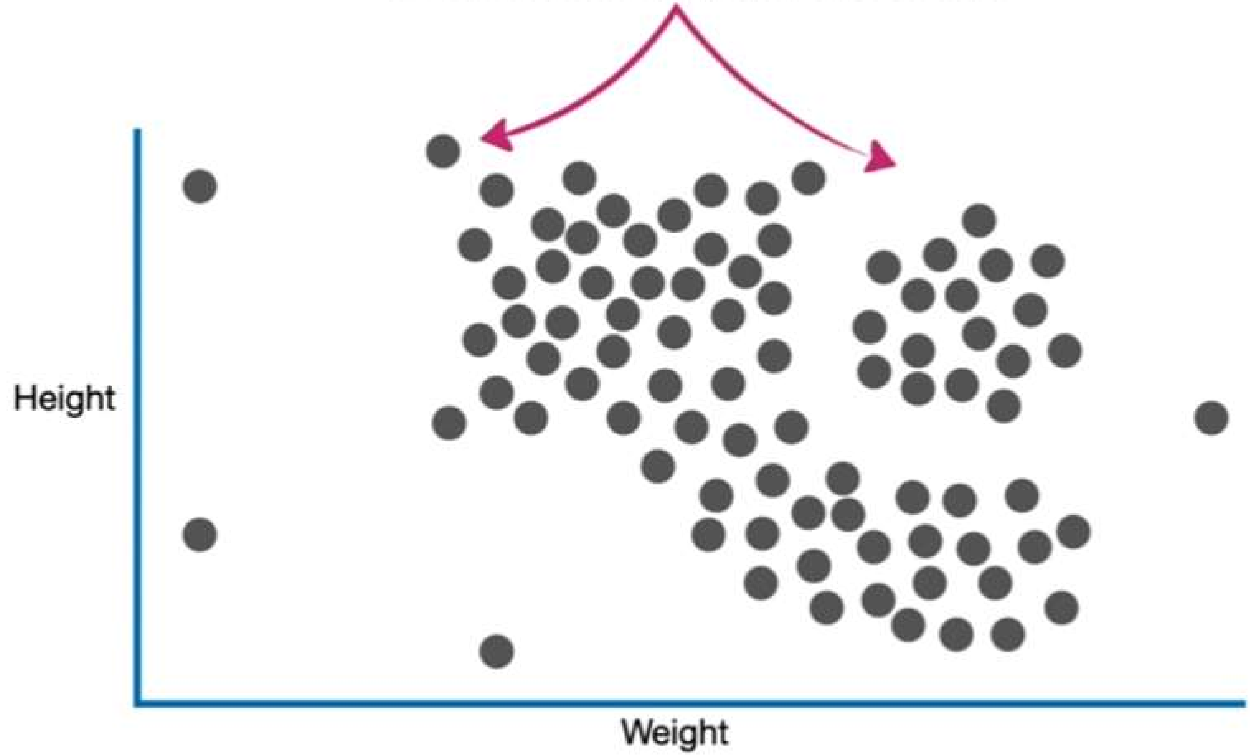
DB SCAN

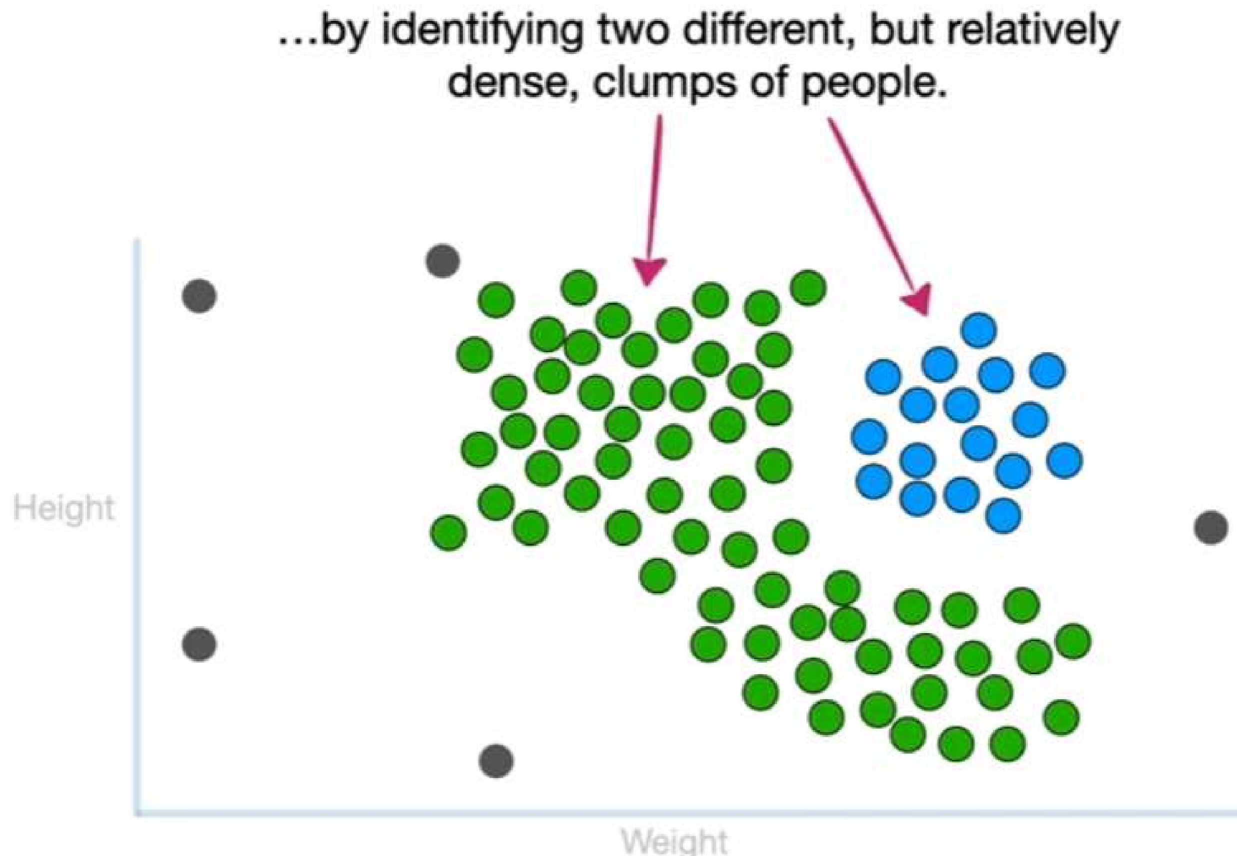
Some slides were adapted/taken from various sources, including Prof. Andrew Ng's Coursera Lectures, Stanford University, Prof. Kilian Q. Weinberger's lectures on Machine Learning, Cornell University, Prof. Sudeshna Sarkar's Lecture on Machine Learning, IIT Kharagpur, Prof. Bing Liu's lecture, University of Illinois at Chicago (UIC), University of Buffalo, CS231n: Convolutional Neural Networks for Visual Recognition lectures, Stanford University, Dr. Luis Serrano, Prof. Alexander Ihler, Dr. Josh Starmer and many more. We thankfully acknowledge them. Students are requested to use this material for their study only and **NOT** to distribute it.

Now, imagine we collected **Weight** and **Height** measurements from a bunch of people…

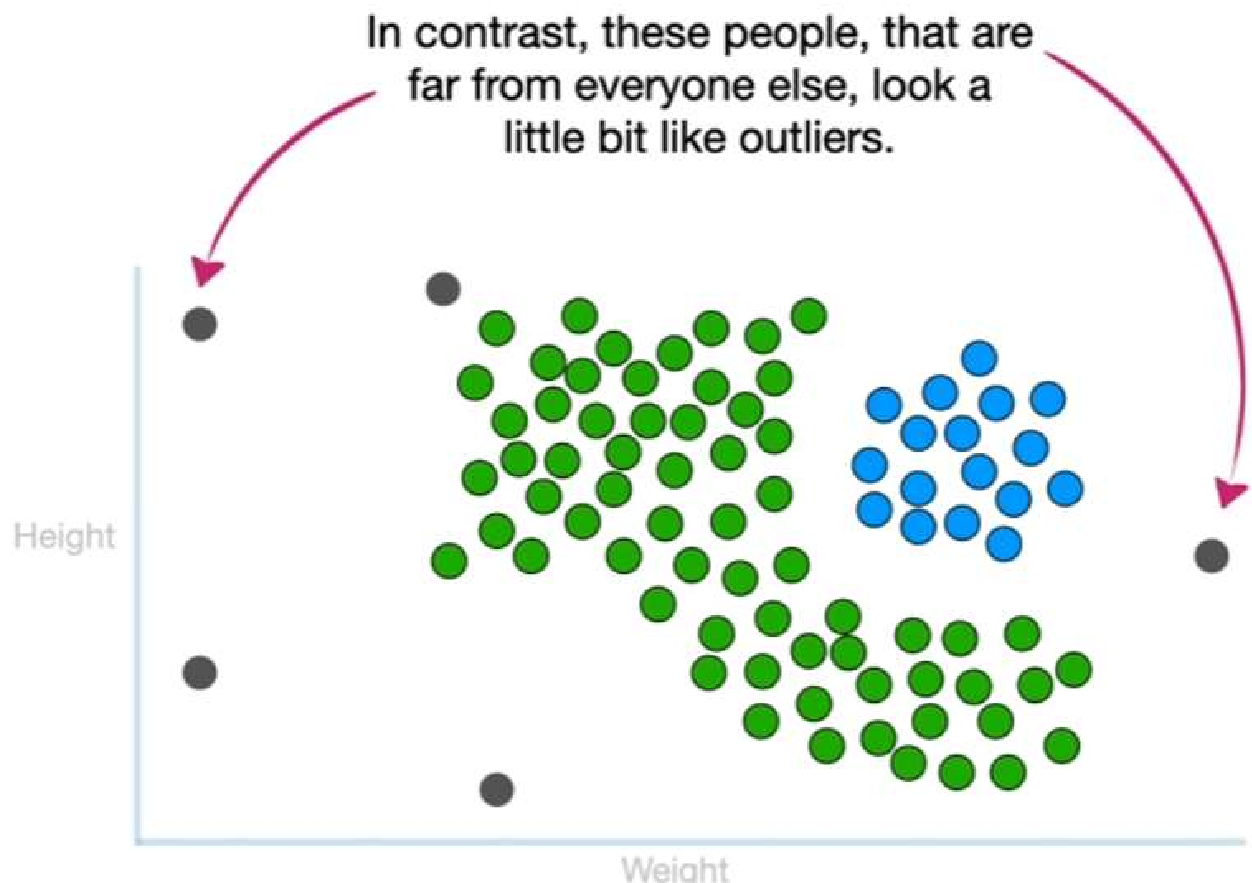
	Weight	Height
Person 1	56	150
Person 2	62	170
Person 3	71	168
…	…	…

…and we plotted the people on a **2**-dimensional graph like this…

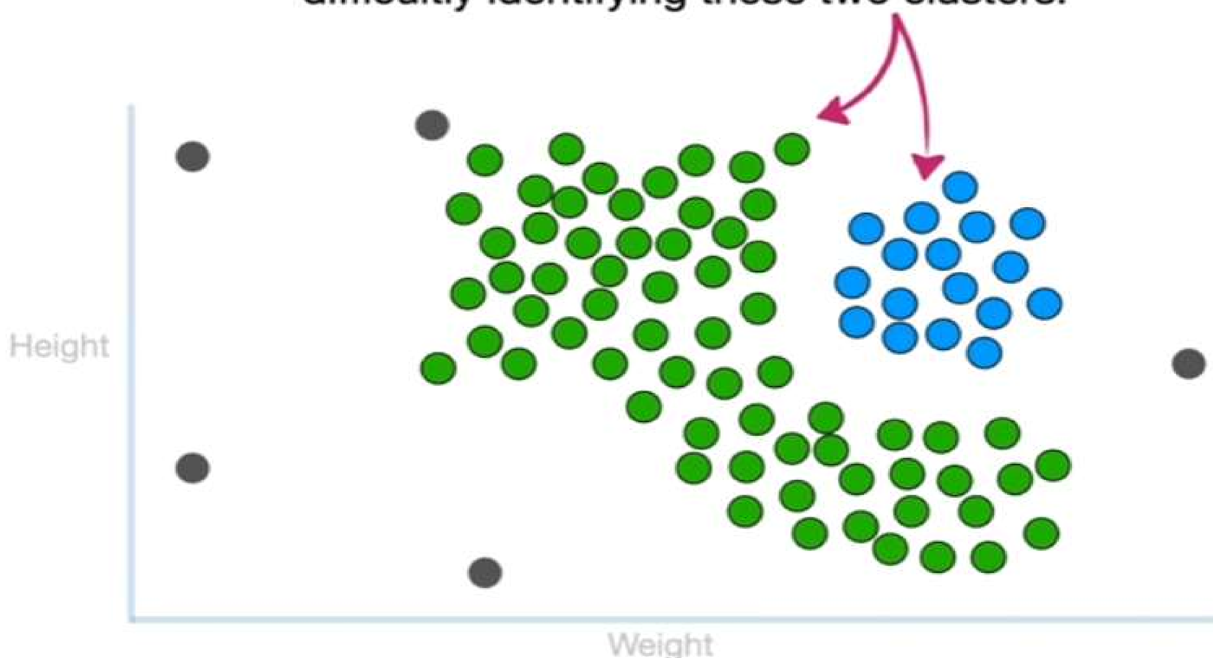




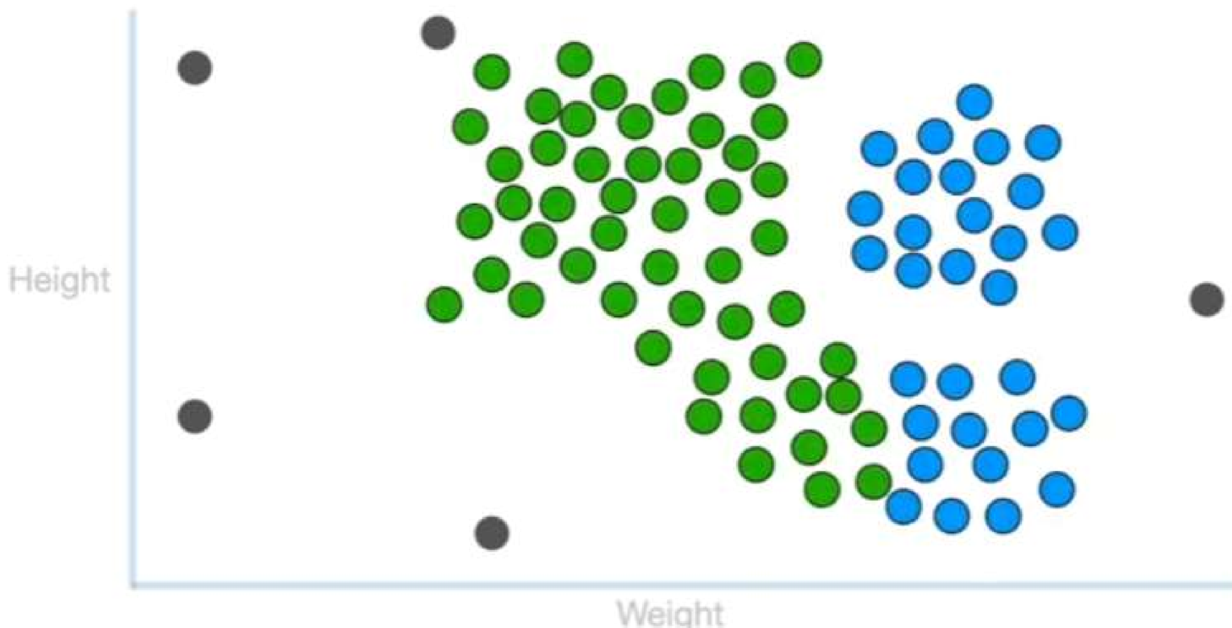
…by identifying two different, but relatively dense, clumps of people.
Height
Weight


In contrast, these people, that are far from everyone else, look a little bit like outliers.
Height
Weight

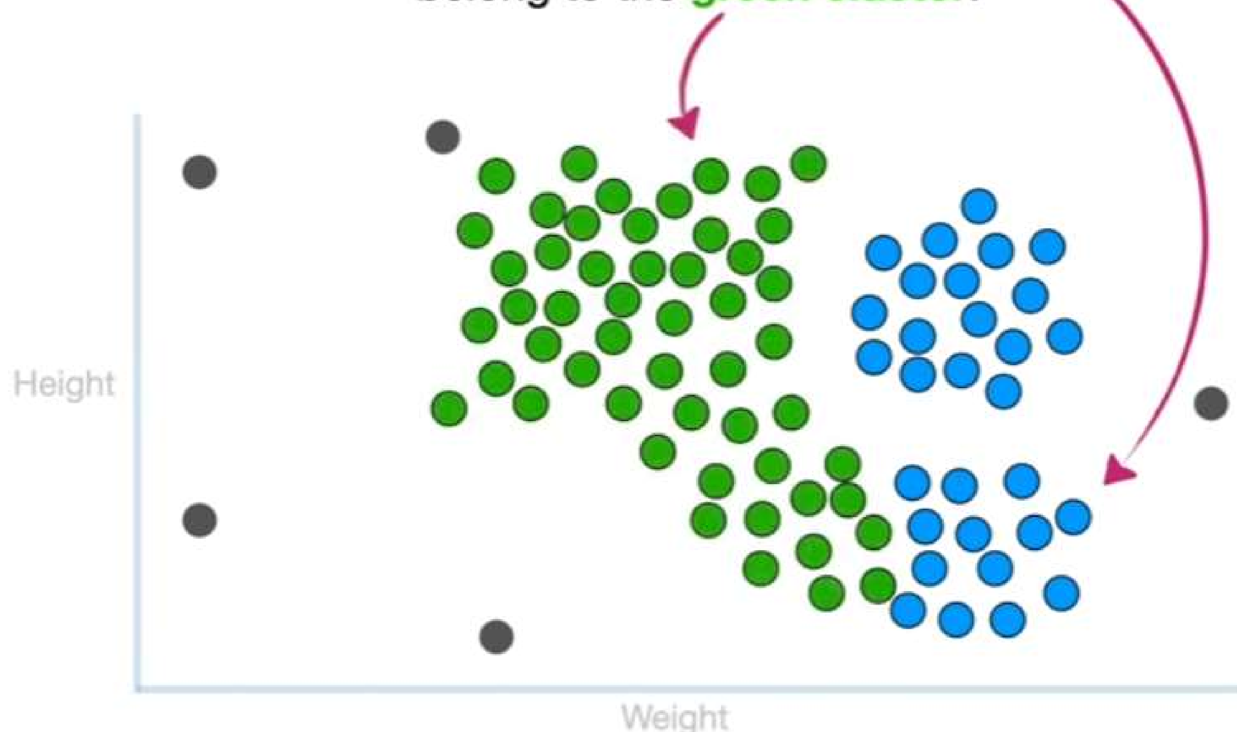
…a relatively standard clustering method like, **k-means clustering**, might have difficultly identifying these two clusters.



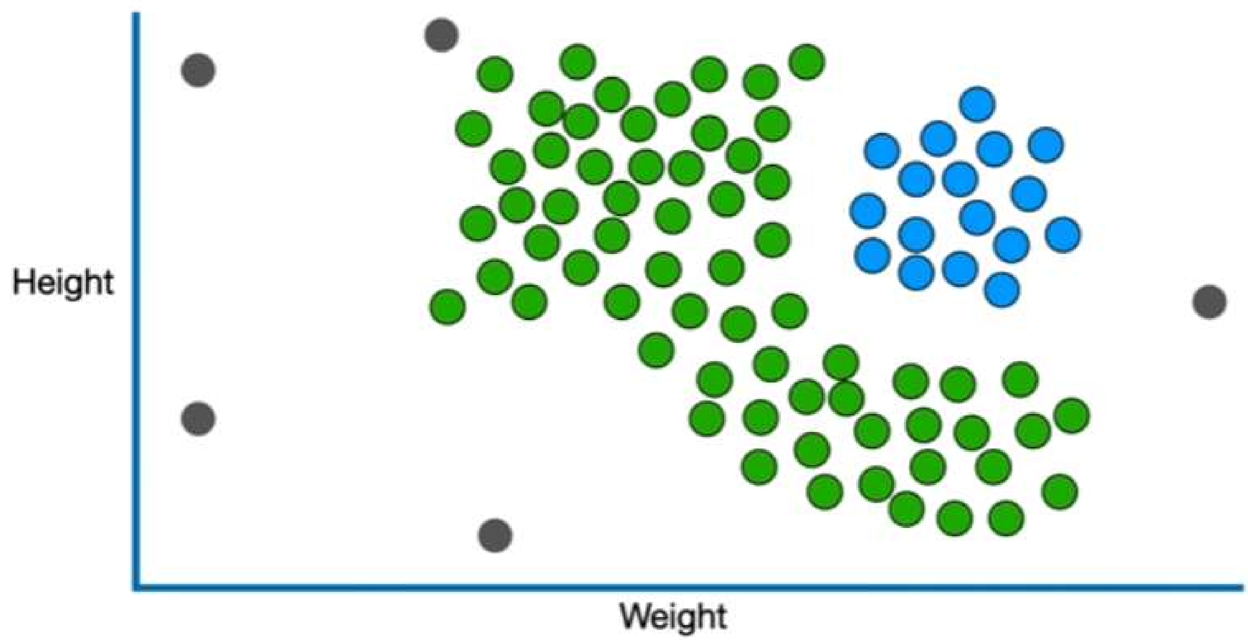
Instead, because of the nesting, a simple clustering method might get something weird like this…



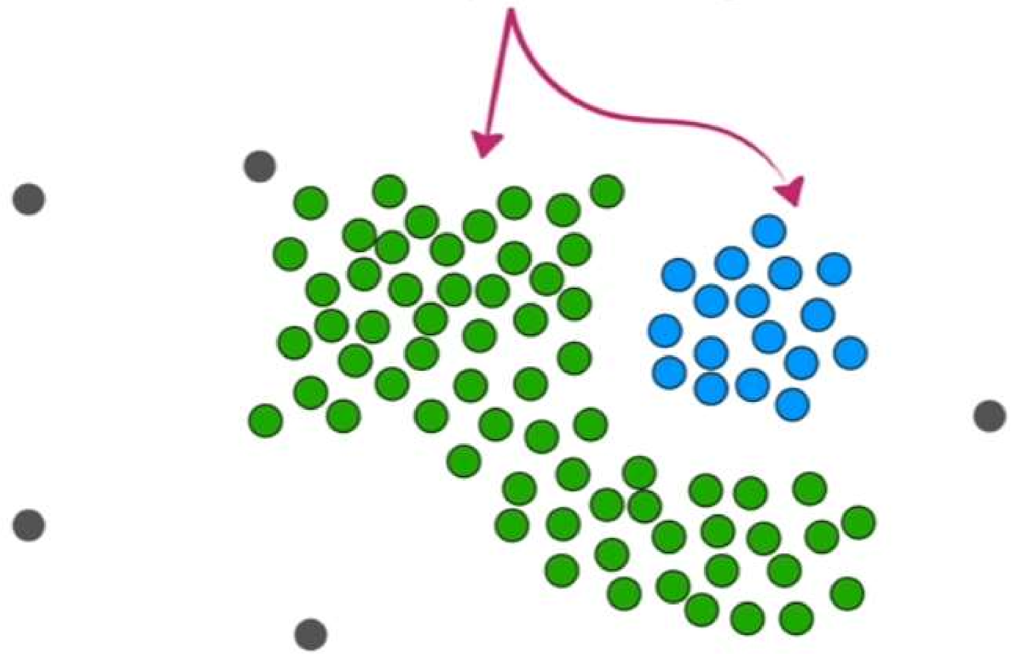
…where these points are assigned to the **blue cluster** even though they look like they belong to the **green cluster**.

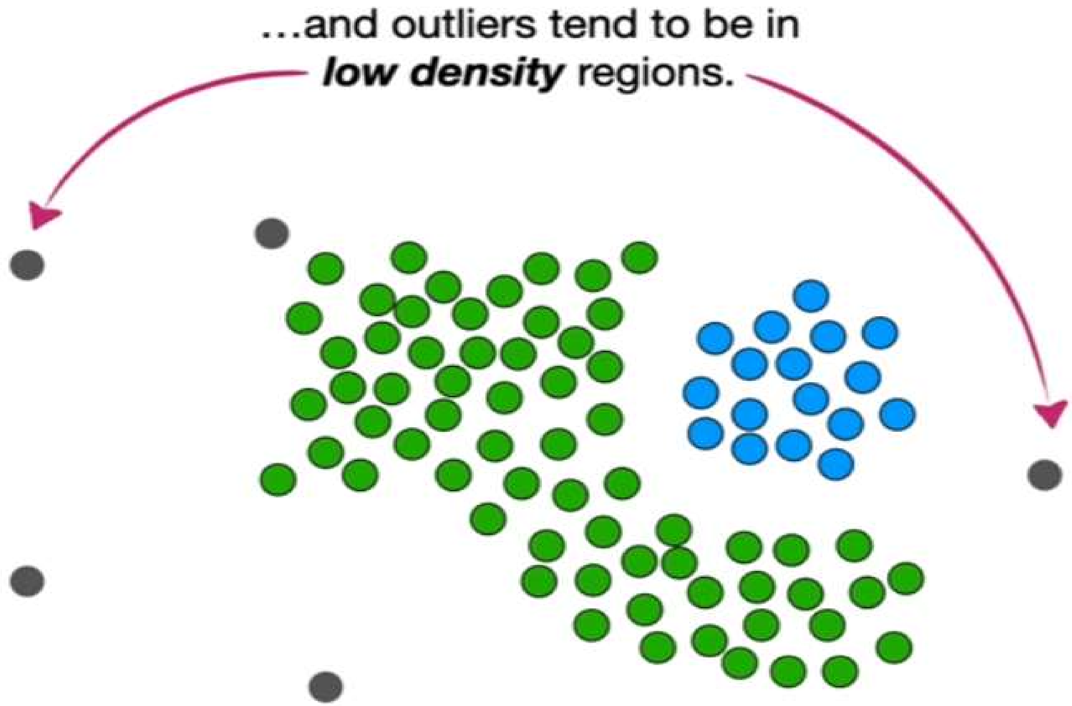


DBSCAN

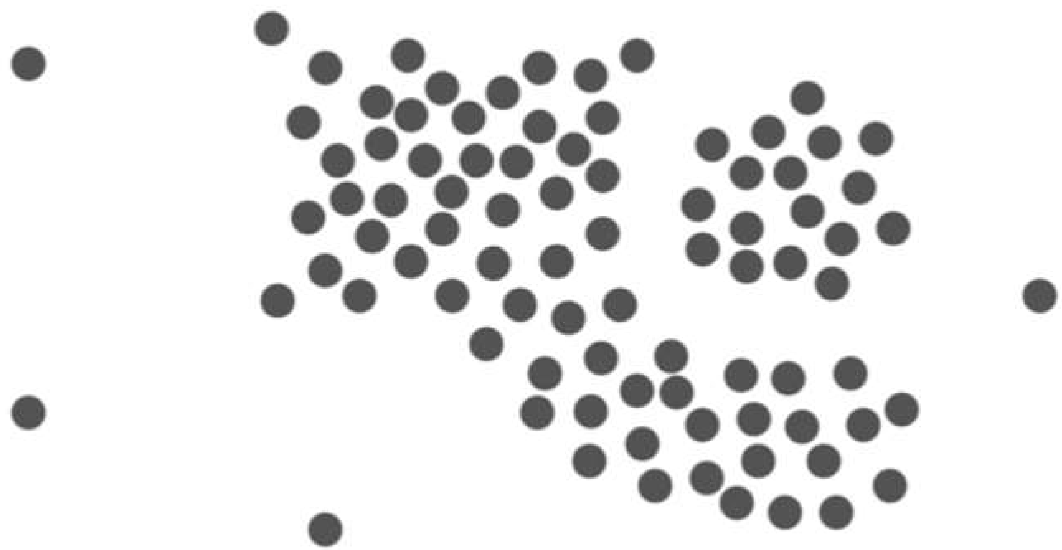


Clusters are in ***high density*** regions…

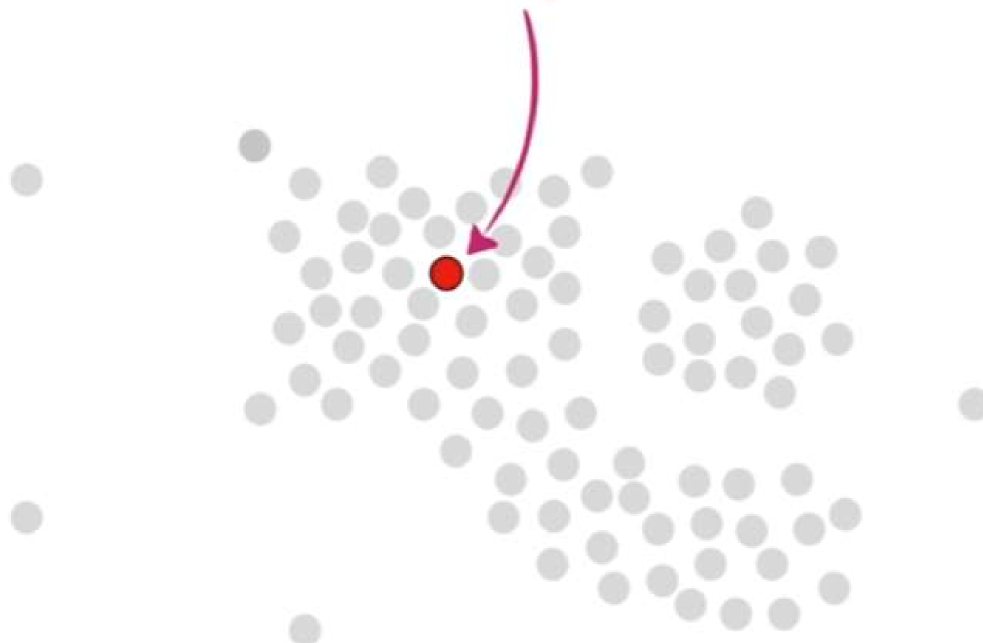



…and outliers tend to be in ***low density*** regions.

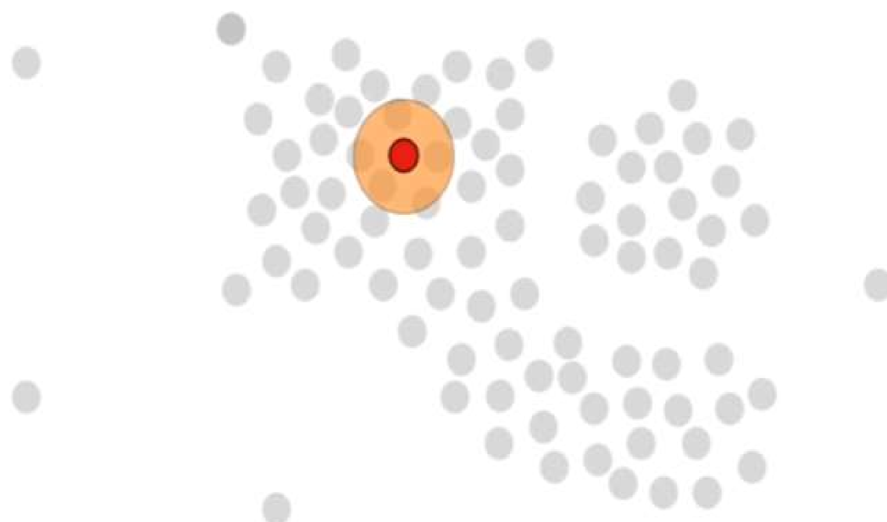
...the first thing we can do is count the number of points close to each point.



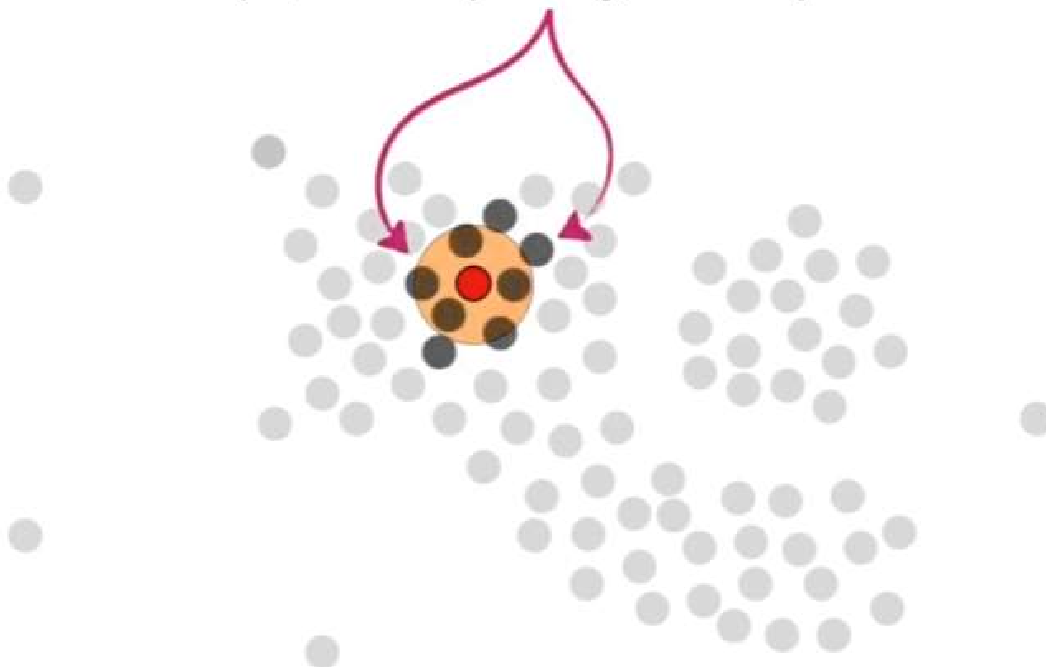
For example, if we start with this **red point**…



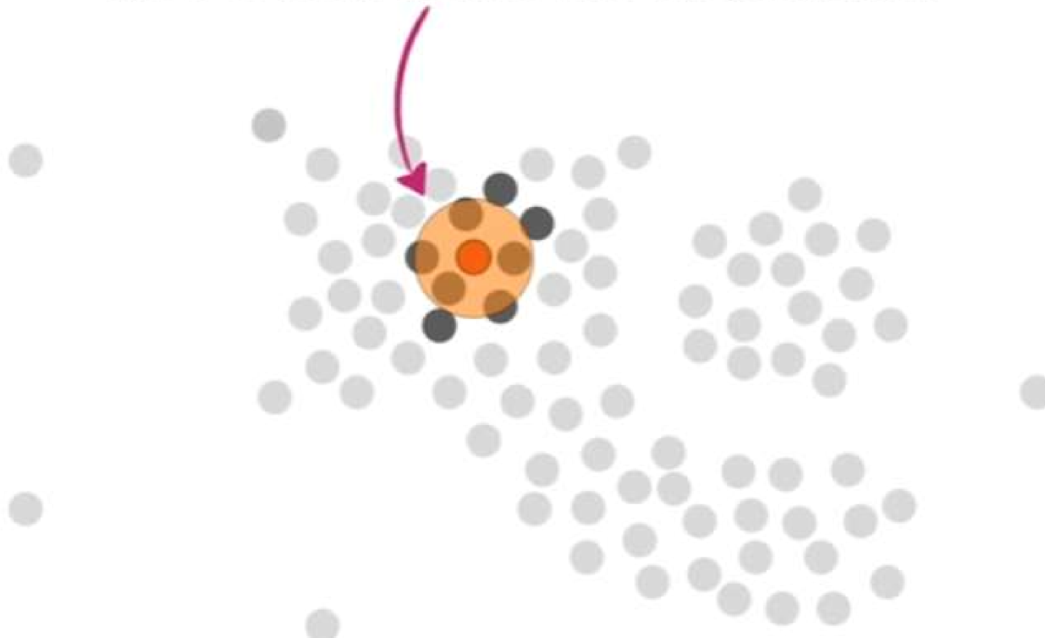
…then we see that the **orange circle** overlaps, at least partially, **8** other points.

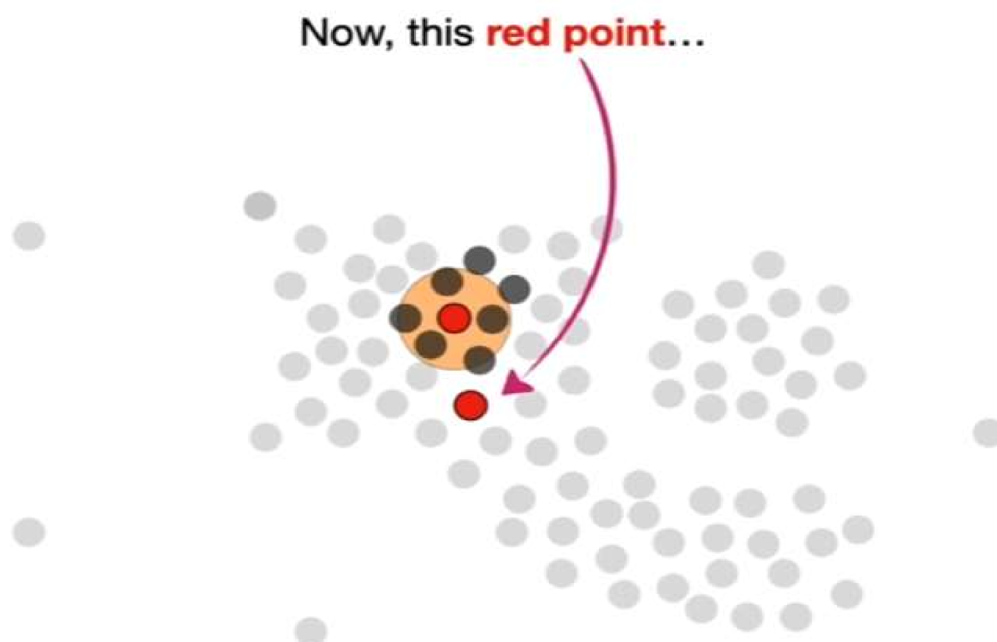


...then we see that the **orange circle** overlaps, at least partially, **8** other points.

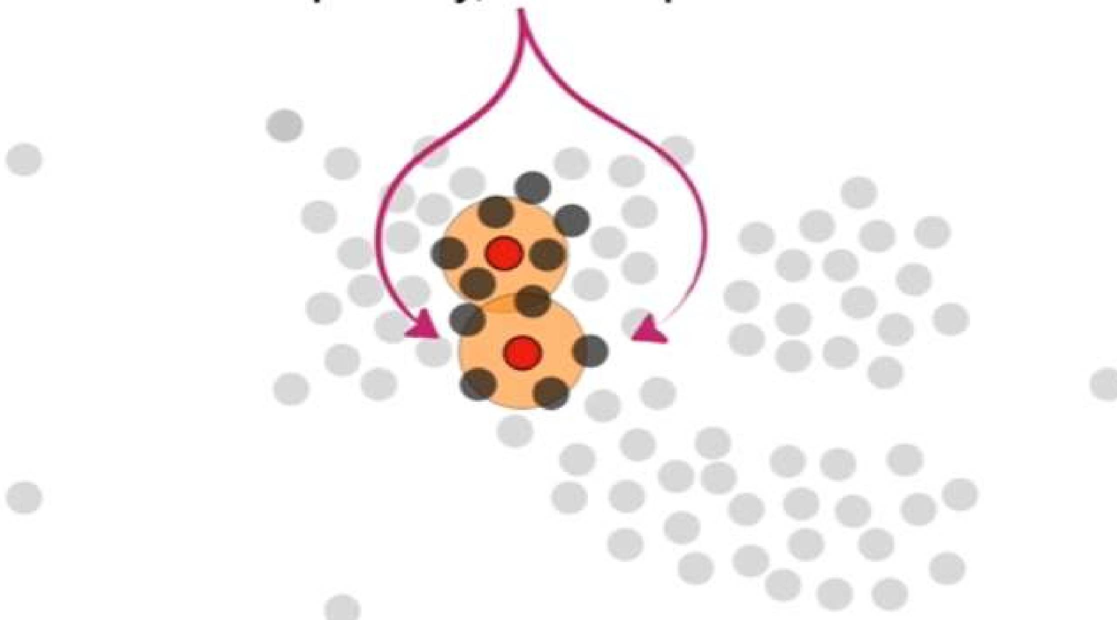


NOTE: The radius of the orange circle is user defined, so when using **DBSCAN**, you may need to fiddle around with this parameter.

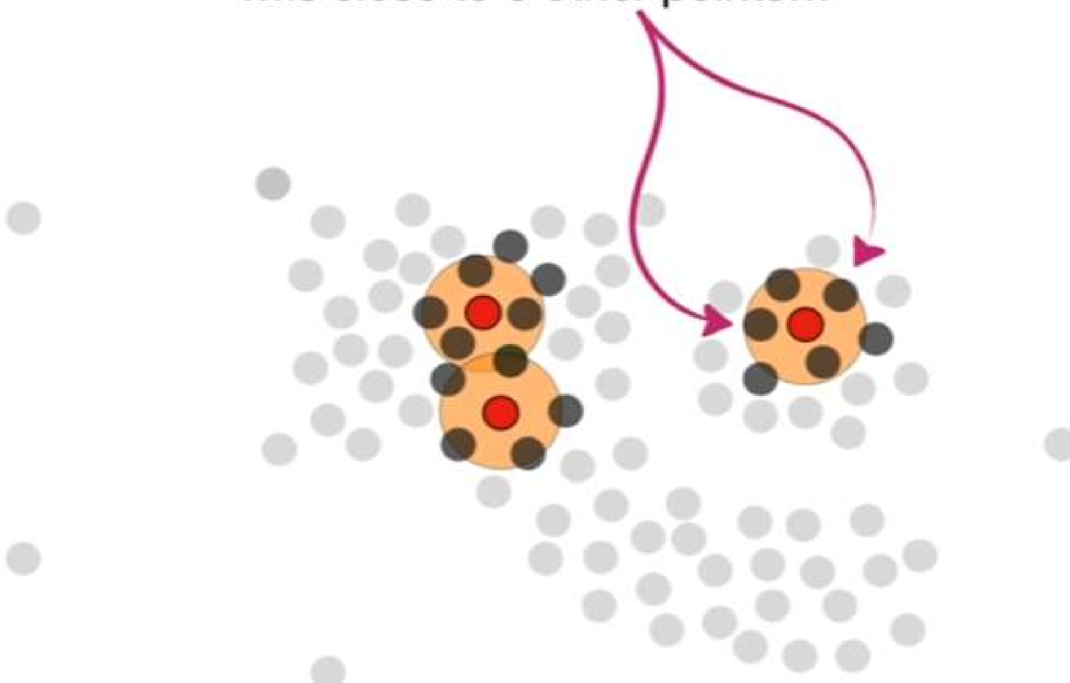


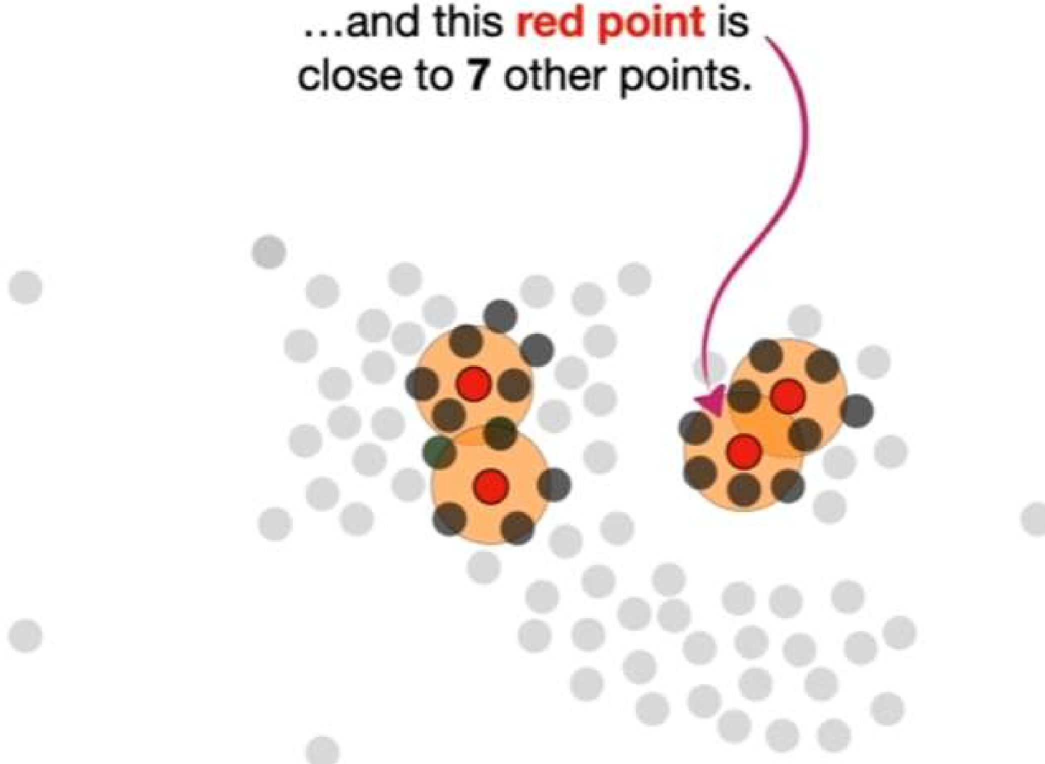

Now, this red point…

…is close to **5** other points because the orange circle overlaps, at least partially, **5** other points.

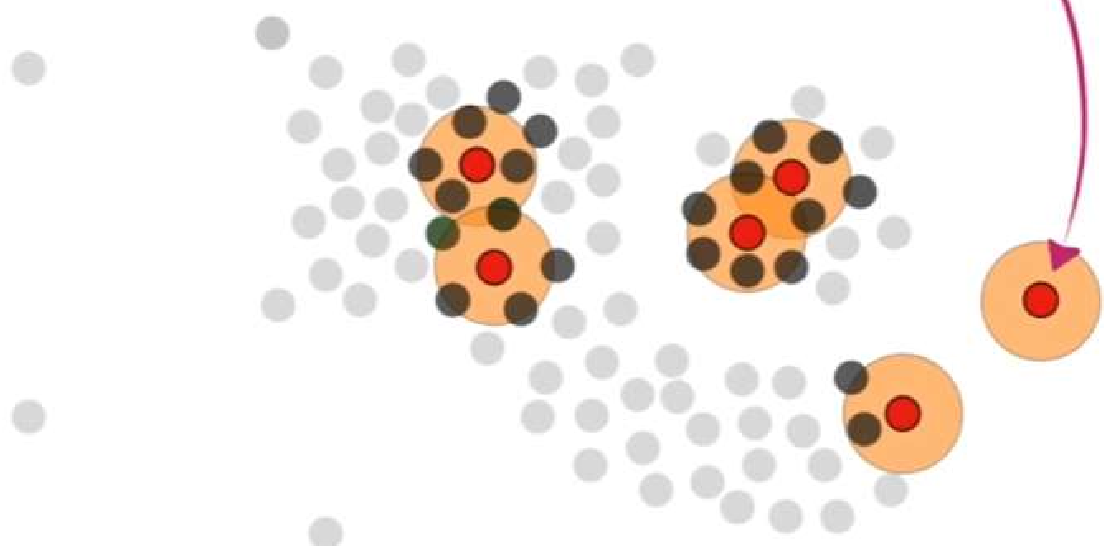


…is close to **6** other points…

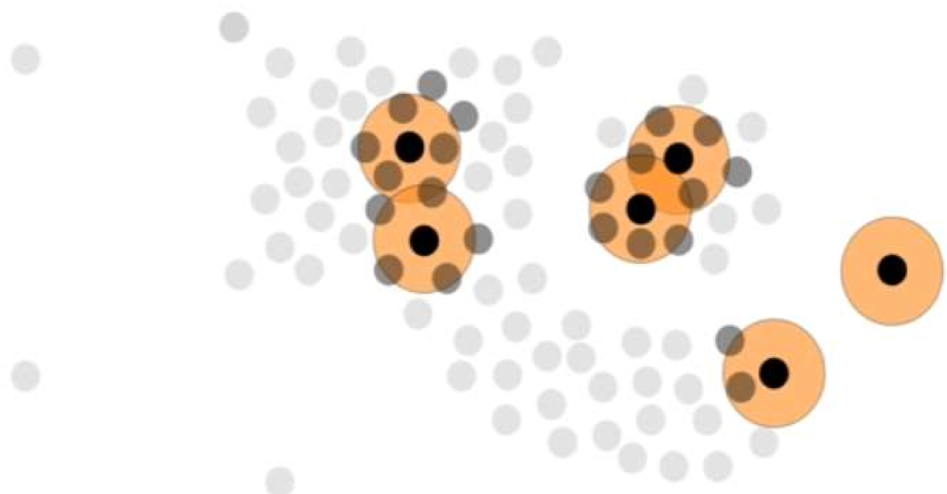



...and this **red point** is
close to **7** other points.

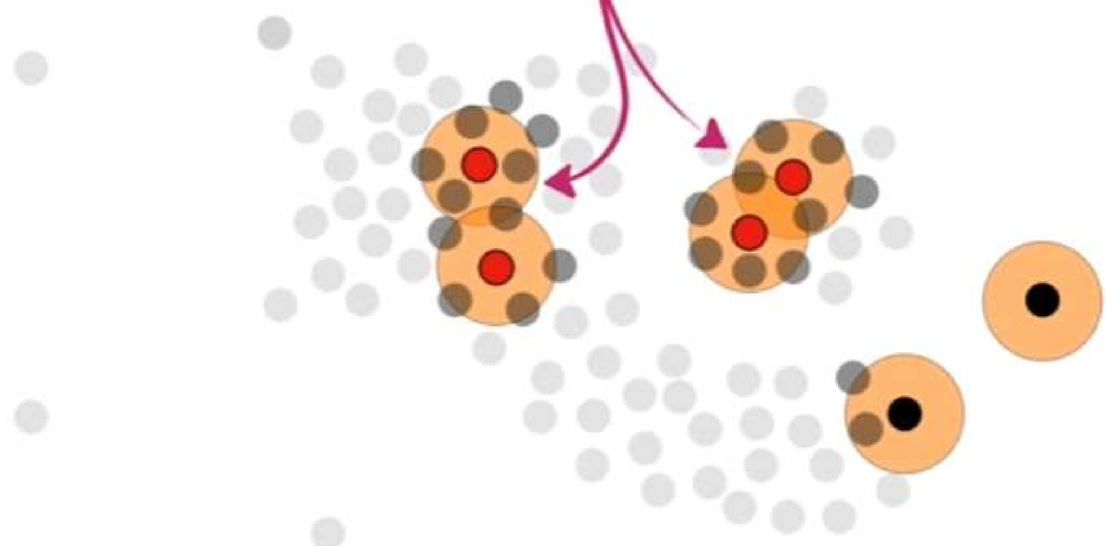
…and this **red point** is not close to any other point because the **orange circle** does not overlap anything else.



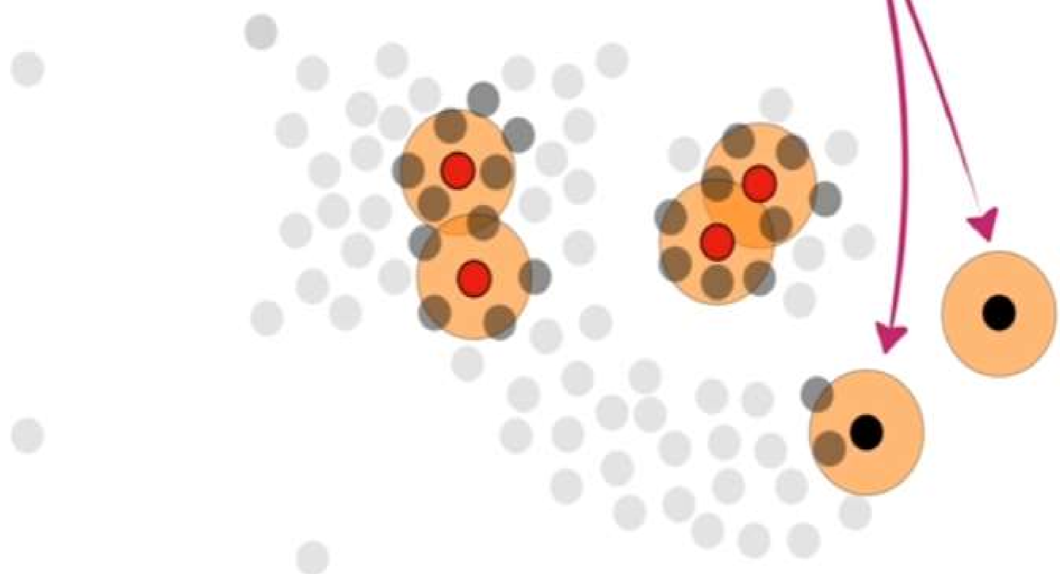
NOTE: The number of close points for a **Core Point** is user defined, so, when using **DBSCAN**, you might need to fiddle with this parameter as well.



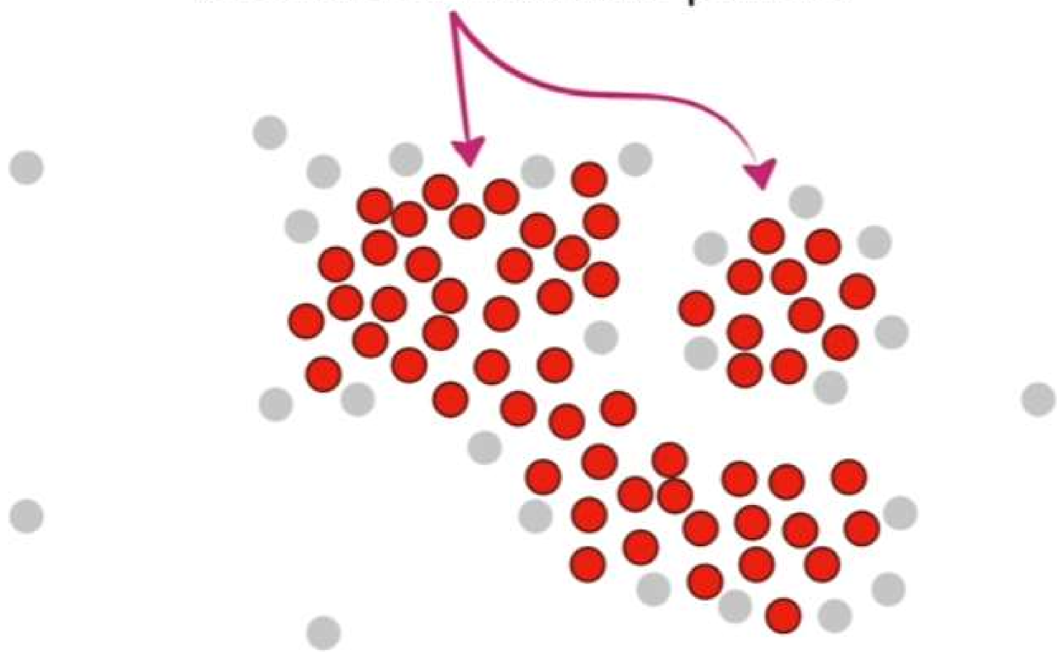
Anyway, these **4** points are some of the **Core Points**, because their **orange circles** overlap at least **4** other points…

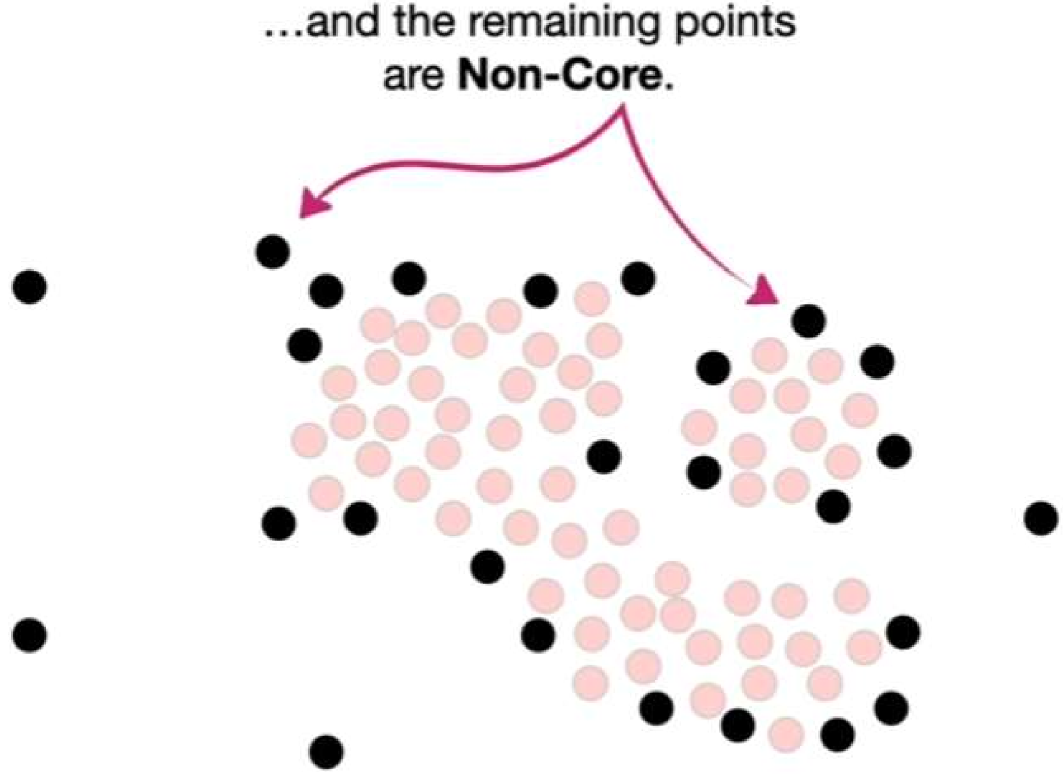


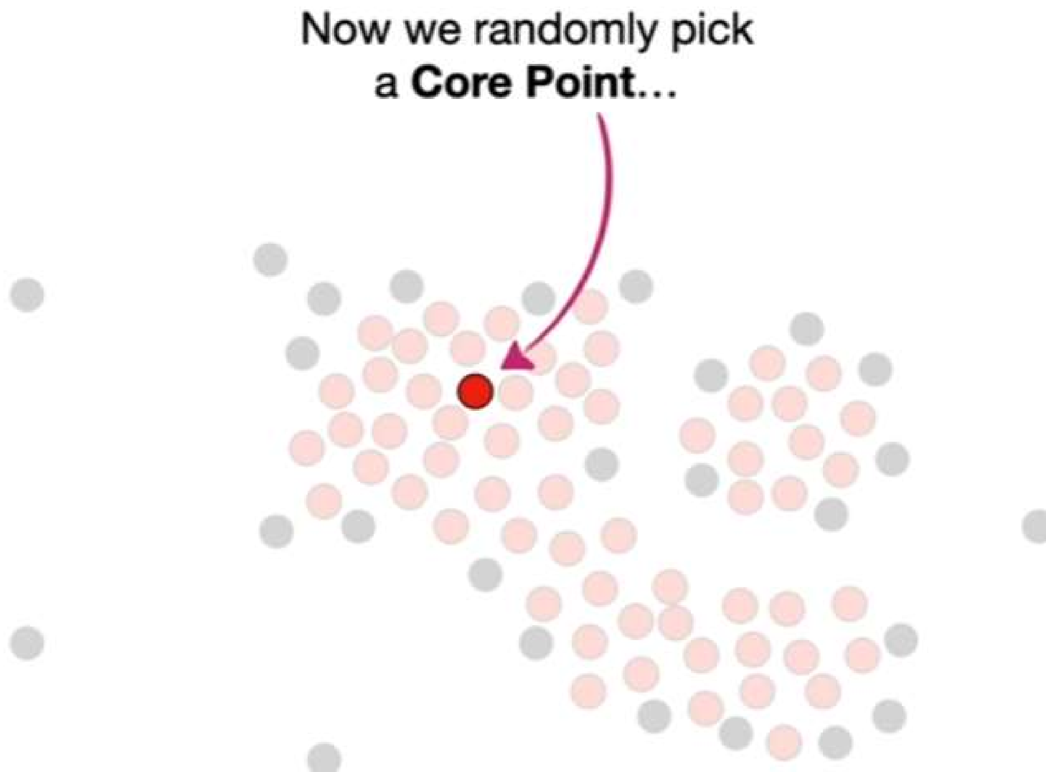
…but neither of these points are **Core Points** because their orange circles do not overlap **4** or more other points.

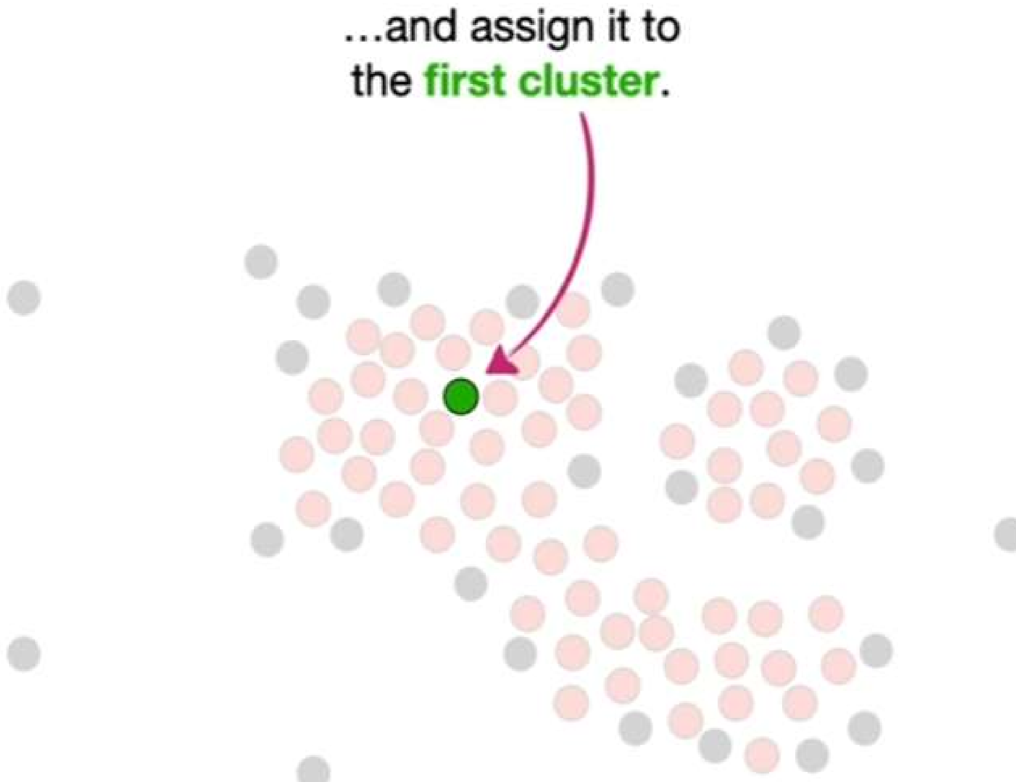


Ultimately, we can call all of these **red points** **Core Points** because they are all close to **4** or more other points…

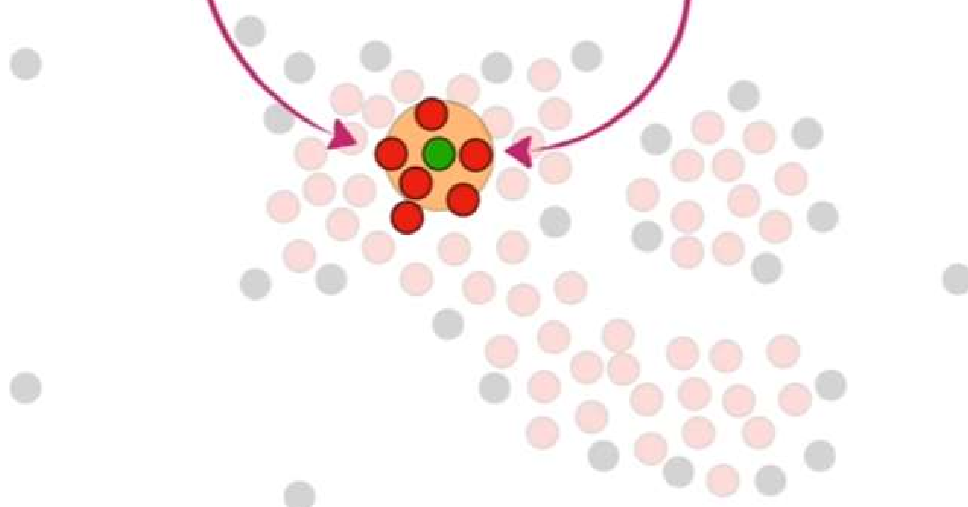



…and the remaining points
are **Non-Core**.

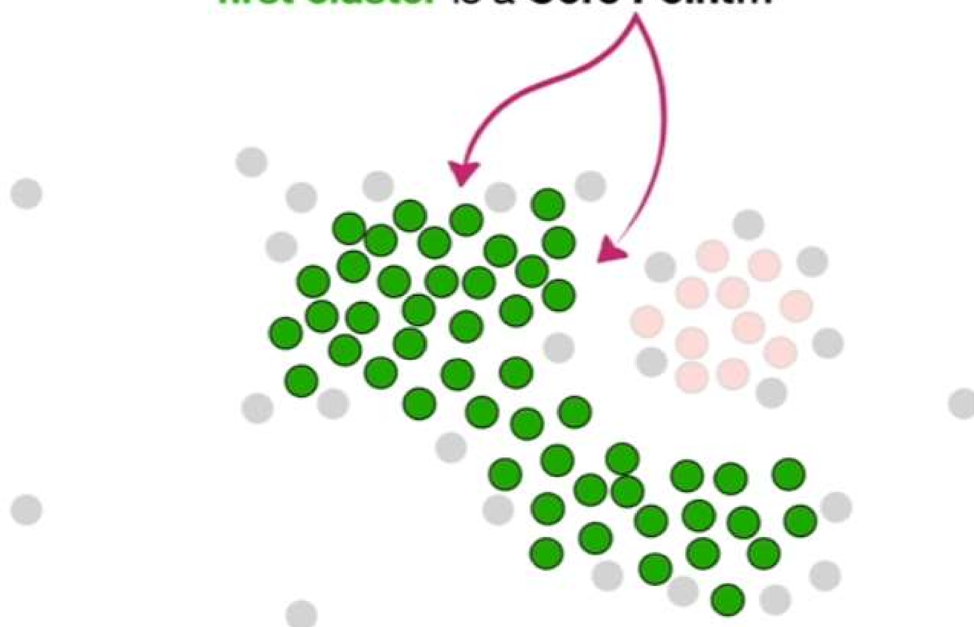

Now we randomly pick
a **Core Point**…


…and assign it to
the first cluster.

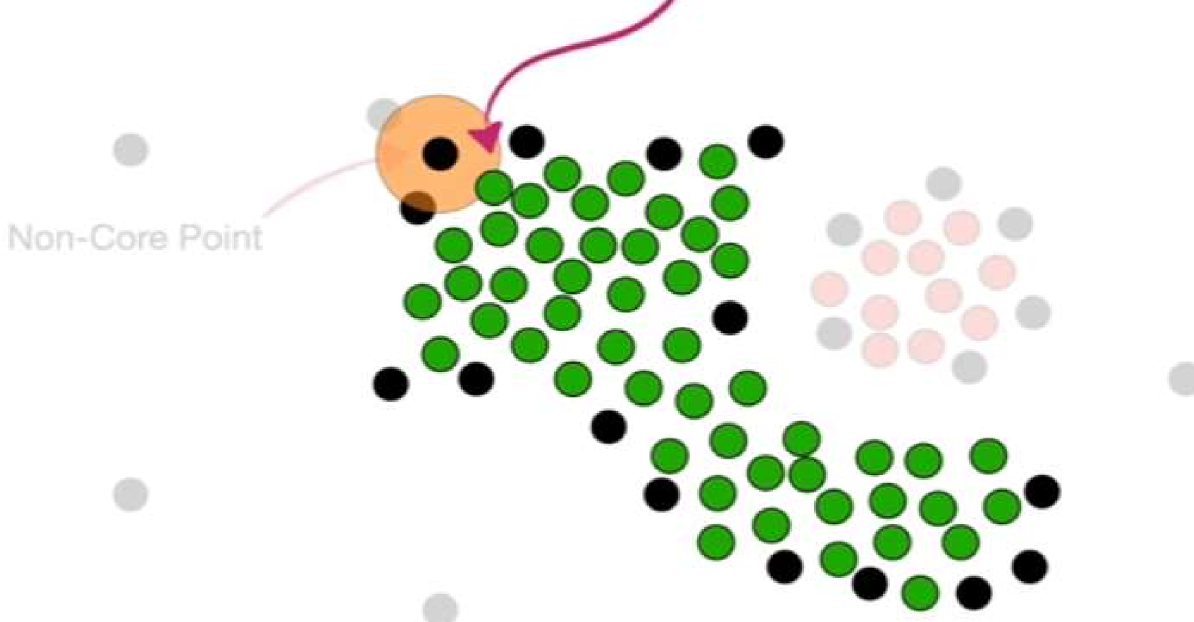
Next, the **Core Points** that are close to the **first cluster**, meaning they overlap the **orange circle**…

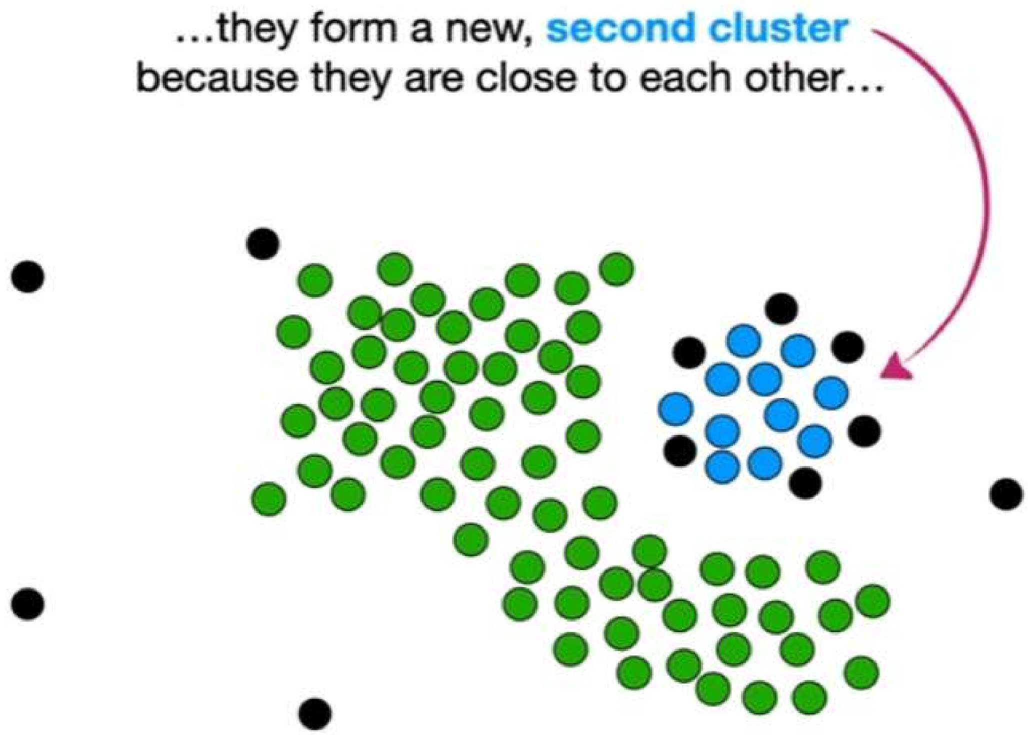


NOTE: At this point, every single point in the **first cluster** is a **Core Point**…

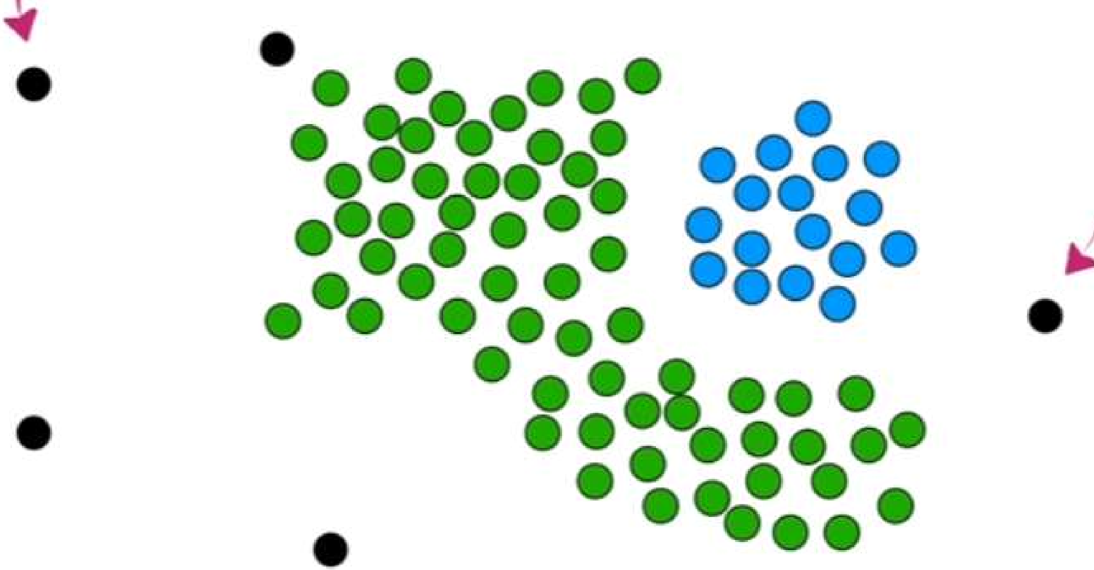


...is close to a **Core Point** in
the **first cluster**...




…they form a new, second cluster
because they are close to each other…

…and any remaining **Non-Core Points** that are not close to **Core Points** in either cluster…



Density based Clustering

- Why Density-Based Clustering methods?
    - Discover clusters of arbitrary shape.
    - Clusters – Dense regions of objects separated by regions of low density
  - DBSCAN – the first density based clustering
  - OPTICS – density based cluster-ordering
  - DENCLUE – a general density-based description of cluster and clustering

DBSCAN:
Density Based Spatial Clustering of Applications with Noise

- Proposed by Ester, Kriegel, Sander, and Xu (KDD96)

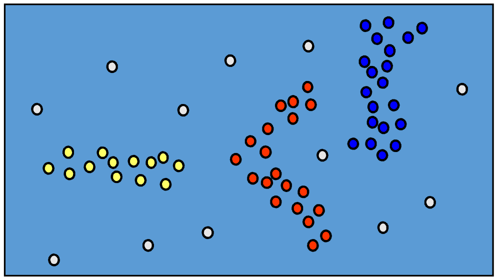
- Relies on a density-based notion of cluster: A cluster is defined as a maximal set of density-connected points.

- Discovers clusters of arbitrary shape in spatial databases with noise

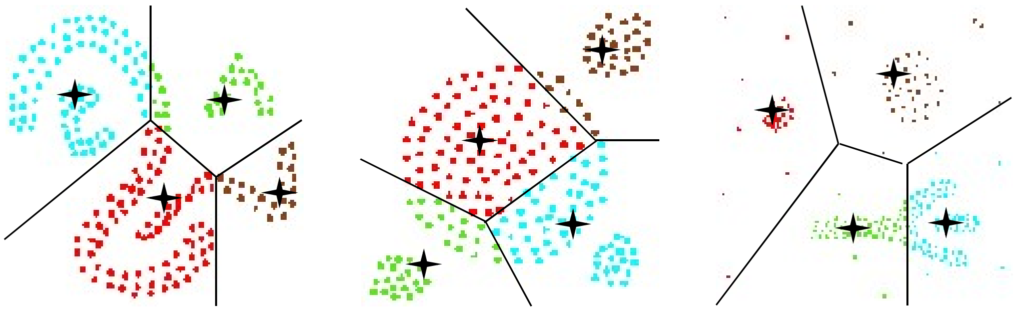
Density based Clustering

- *Basic Idea*:

  Clusters are dense regions in the data space, separated by regions of lower object density



Why Density-Based Clustering?



Results of a *k*-medoid algorithm for *k*=4

Different density-based approaches exist. Here we discuss the ideas underlying the DBSCAN algorithm

Density based Clustering
Basic Concept

- Intuition for the formalization of the basic idea
  - For any point in a cluster, the local point density around that point has to exceed some threshold
  - The set of points from one cluster is spatially connected

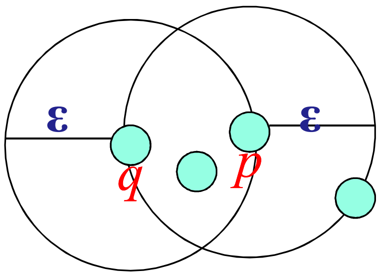
- Local point density at a point p defined by two parameters
  - e – radius for the neighborhood of point p:
    $N_e(p) := \{q \text{ in data set } D \mid dist(p, q) \leq e\}$
  - *MinPts* – minimum number of points in the given neighborhood $N(p)$

ε-Neighborhood

- ε-Neighborhood – Objects within a radius of ε from an object.

$$N_{\varepsilon}(p):\{q \mid d(p,q) \leq \varepsilon\}$$

- "High density" - ε-Neighborhood of an object contains at least *MinPts* of objects.



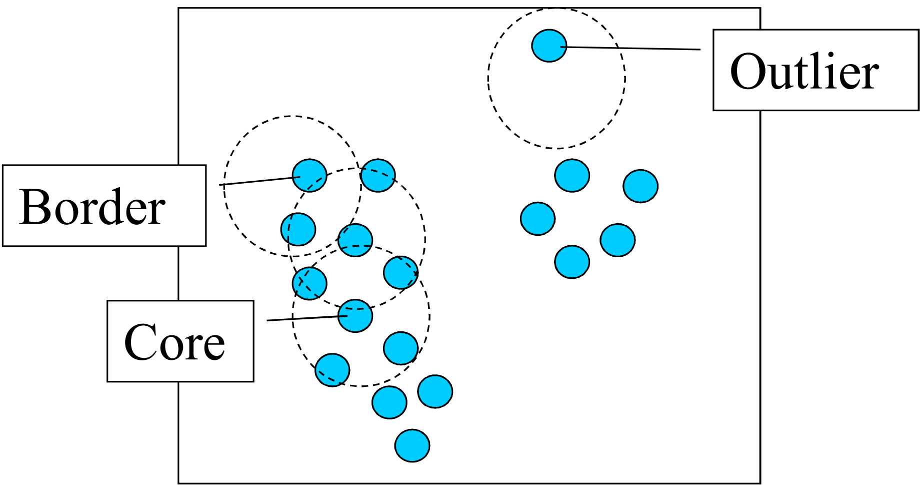
ε-Neighborhood of *p*

ε-Neighborhood of *q*

Density of p is "high" (MinPts = 4)

Density of q is "low" (MinPts = 4)

Core Point, Border Point, Outlier



ε = 1unit, MinPts = 5

Given ε and *MinPts*, categorize the objects into three exclusive groups.

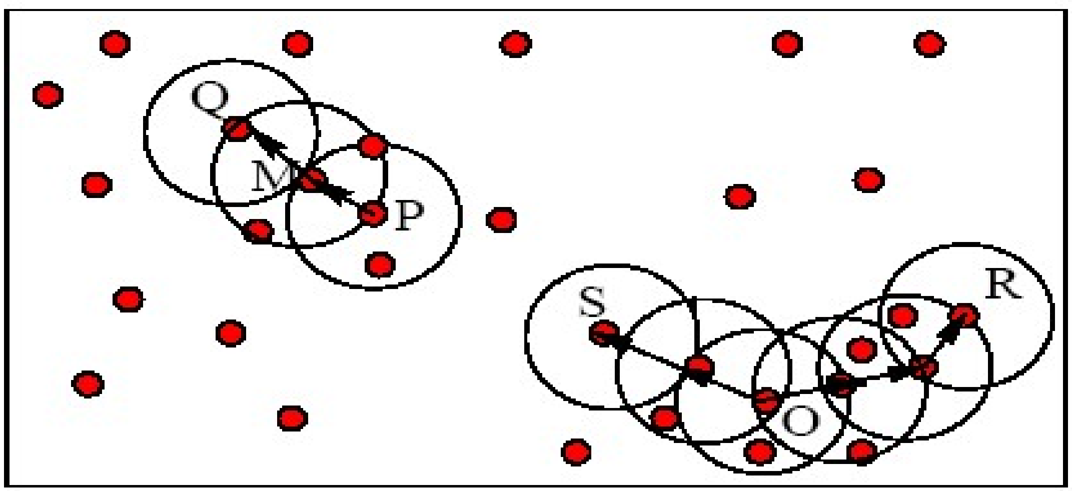
A point is a core point if it has more than a specified number of points (MinPts) within Eps. These are points that are at the interior of a cluster.

A border point has fewer than MinPts within Eps, but is in the neighborhood of a core point.

A noise point is any point that is not a core point nor a border point.

Example:

M, P, O, and R are core objects since each is in an Eps neighborhood containing at least 3 points



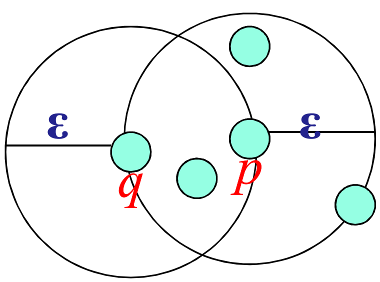
Minpts = 3

Eps=radius of the circles

Density Reachability

Directly density-reachable

An object q is directly density-reachable from object p if p is a core object and q is in p's ε-neighborhood.

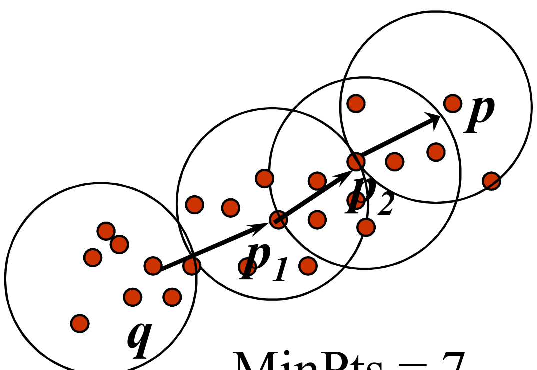


MinPts = 4

- q is directly density-reachable from p
- p is not directly density- reachable from q?
- Density-reachability is asymmetric.

Density Reachability

- Density-Reachable (directly and indirectly):
  - A point p is directly density-reachable from p2;
  - p2 is directly density-reachable from p1;
  - p1 is directly density-reachable from q;
  - p←p2←p1←q form a chain.

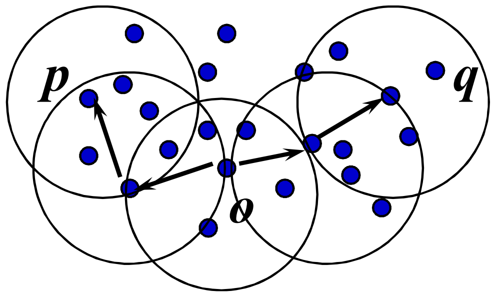


MinPts = 7

- p is (indirectly) density-reachable from q
- q is not density- reachable from p?

Density Reachability

- Density-reachable is not symmetric
  - not good enough to describe clusters
- Density-Connected
  - A pair of points p and q are density-connected if they are commonly density-reachable from a point o.



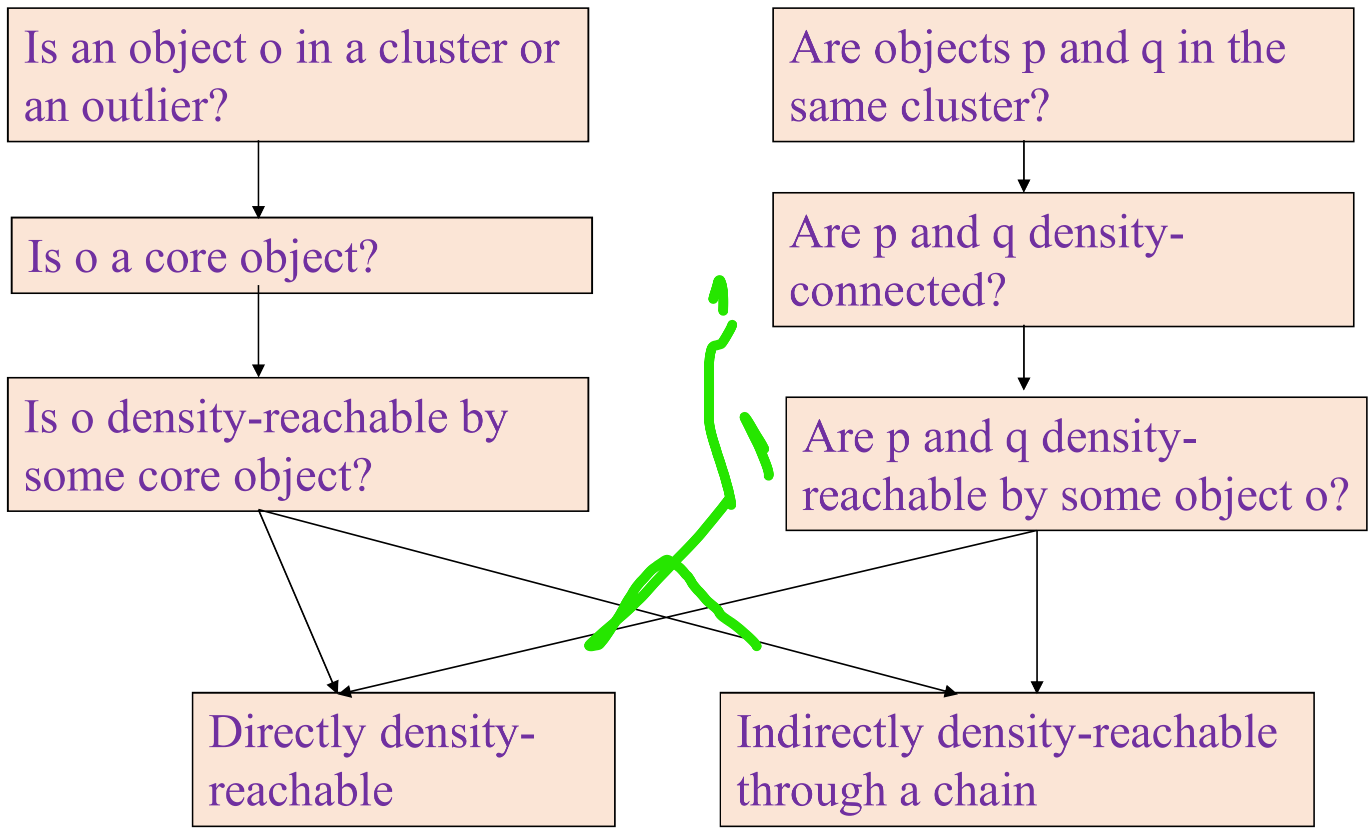
Formal Description of Cluster

- Given a data set D, parameter ε and threshold MinPts.

- A cluster C is a subset of objects satisfying two criteria:

  - *Connected:* $\forall\ p,q \in C$: p and q are density-connected.
  - *Maximal:* $\forall\ p,q$: if $p \in C$ and q is density-reachable from p (where p is the core object), then $q \in C$. (avoid redundancy)

Review of Concepts



DBSCAN Algorithm

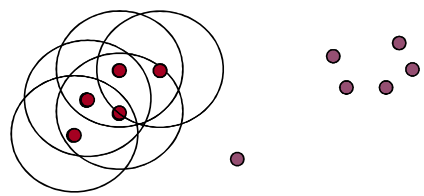
```
Input: The data set D

Parameter: ε, MinPts

For each object p in D
    if p is a core object and not processed then
        C = retrieve all objects density-reachable from p
        mark all objects in C as processed
        report C as a cluster
    else mark p as outlier
    end if

End For
```

DBSCAN: The Algorithm

- Arbitrary select a point *p*
- Retrieve all points density-reachable from *p* wrt *Eps* and *MinPts*.
- If *p* is a core point, a cluster is formed.
- If *p* is a border point, no points are density-reachable from *p* and DBSCAN visits the next point of the database.
- Continue the process until all of the points have been processed.

DBSCAN Algorithm: Example

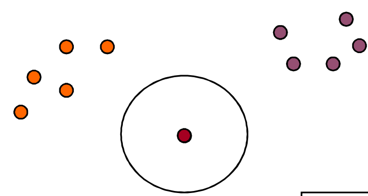
- Parameter
  - $e = 2$ cm
  - $MinPts = 3$



for each *o* Î *D* **do**
    if *o* is not yet classified **then**
        if *o* is a core-object **then**
            collect all objects density-reachable
from *o*
            and assign them to a new cluster.
        else
            assign *o* to NOISE

DBSCAN Algorithm: Example

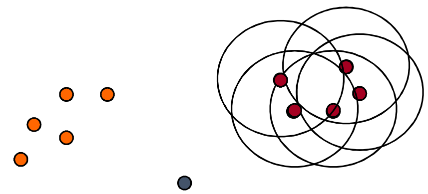
- Parameter
  - ε = 2 cm
  - *MinPts* = 3



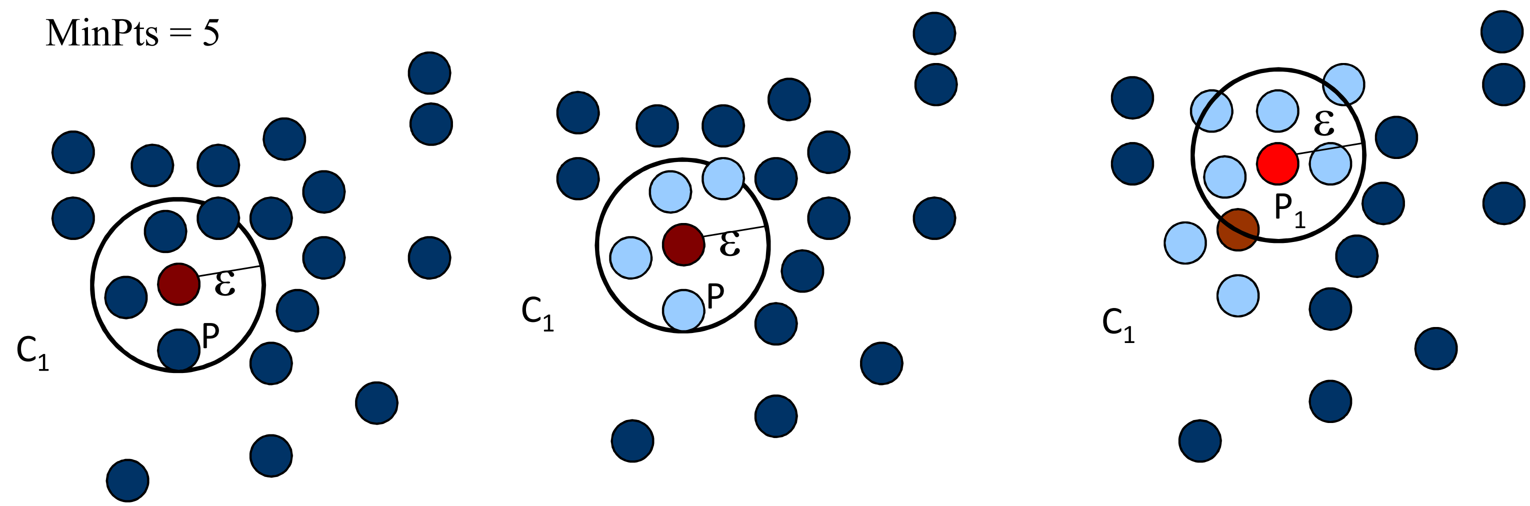
for each *o* Î *D* **do**
    if *o* is not yet classified **then**
        if *o* is a core-object **then**
            collect all objects density-reachable
from *o*
            and assign them to a new cluster.
        else
            assign *o* to NOISE

DBSCAN Algorithm: Example

- Parameter
  - ε = 2 cm
  - *MinPts* = 3

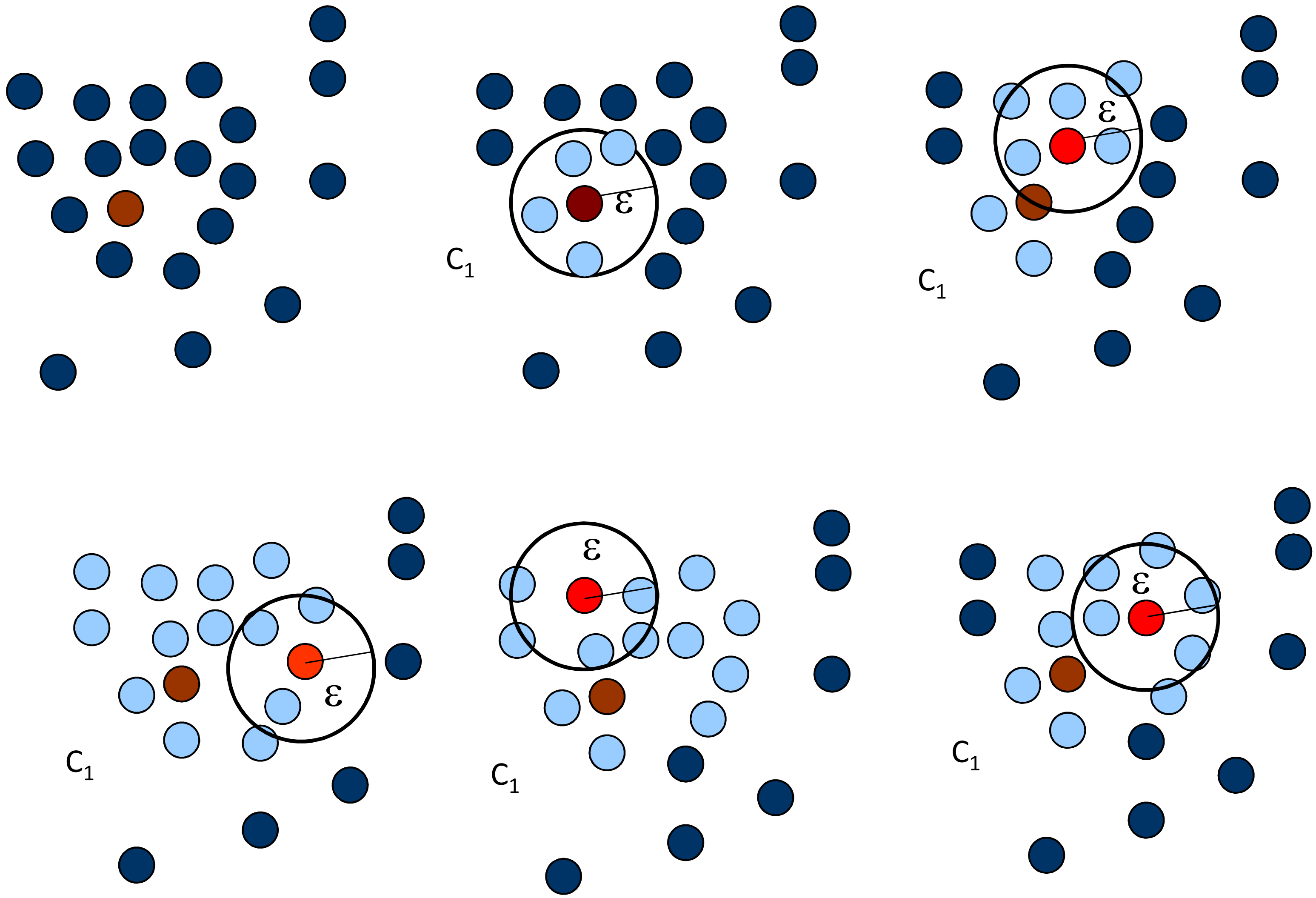


for each *o* Î *D* **do**
    if *o* is not yet classified **then**
        if *o* is a core-object **then**
            collect all objects density-reachable
from *o*
            and assign them to a new cluster.
        else
            assign *o* to NOISE



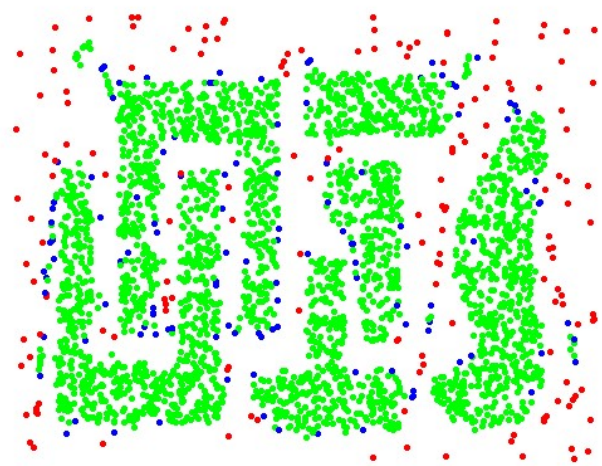
1. Check the ε-neighborhood of p;
2. If p has less than MinPts neighbors then mark p as outlier and continue with the next object
3. Otherwise mark p as processed and put all the neighbors in cluster C

1. Check the unprocessed objects in C
2. If no core object, return C
3. Otherwise, randomly pick up one core object p_1, mark p_1 as processed, and put all unprocessed neighbors of p_1 in cluster C


ε
C_1
ε
C_1
ε
C_1
ε
C_1
ε
C_1

Example

Original Points

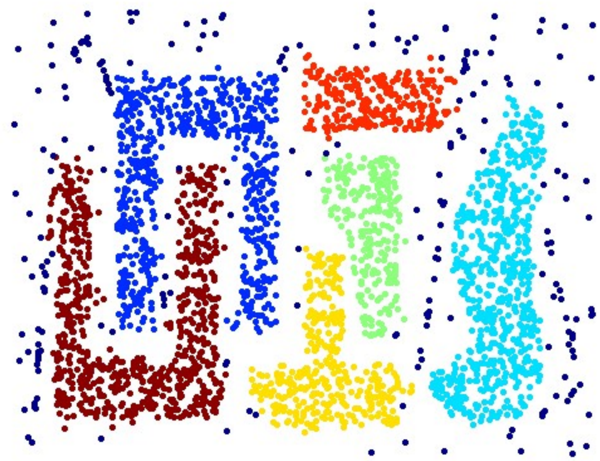


Point types: core, border and outliers

ε = 10, MinPts = 4

When DBSCAN Works Well

Original Points

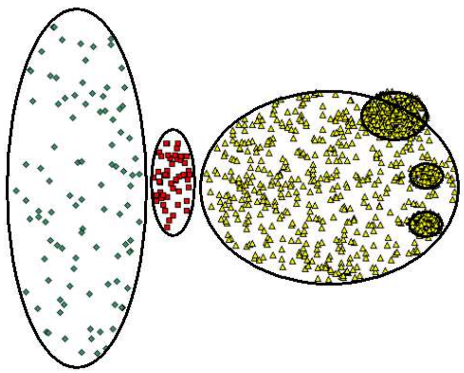


Clusters

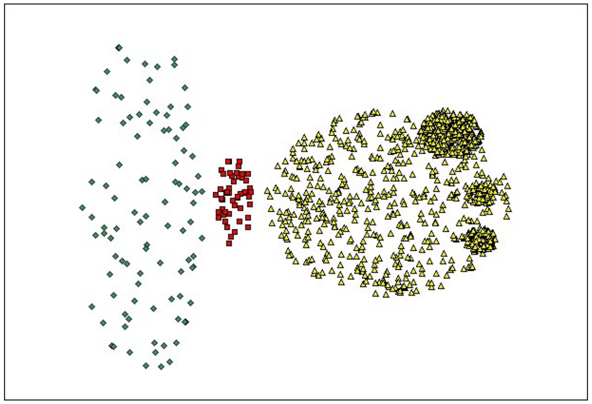
- **Resistant to Noise**
- **Can handle clusters of different shapes and sizes**

Continue…

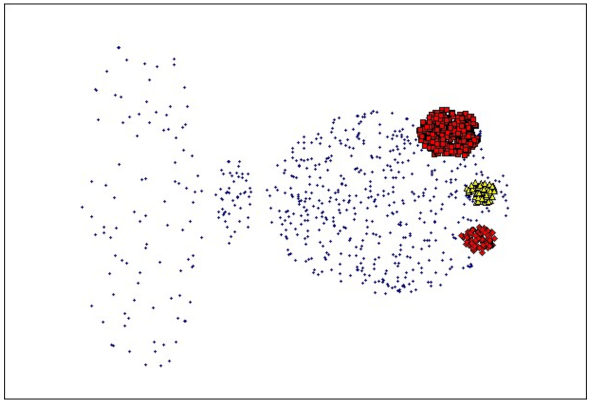
When DBSCAN Does NOT Work Well



Original Points



(MinPts=4, Eps=9.92).

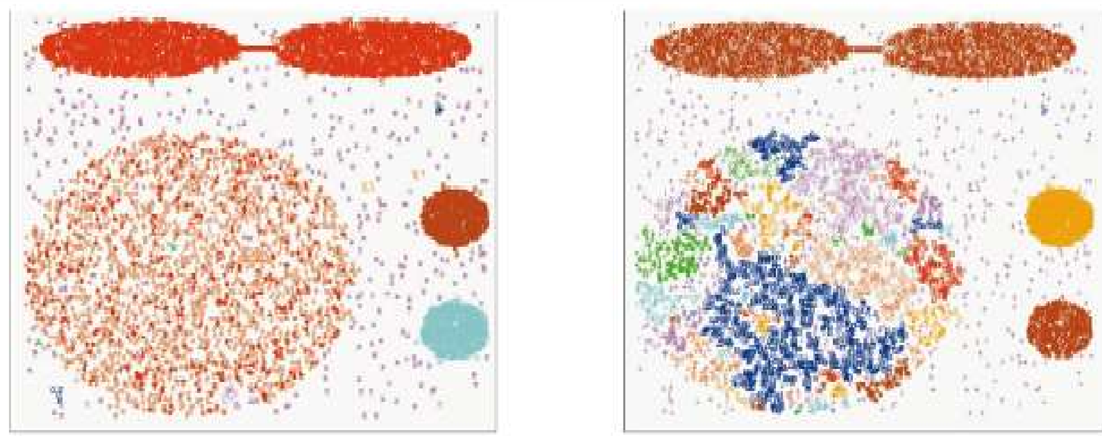


(MinPts=4, Eps=9.75)

- **Cannot handle Varying densities**
- **sensitive to parameters**

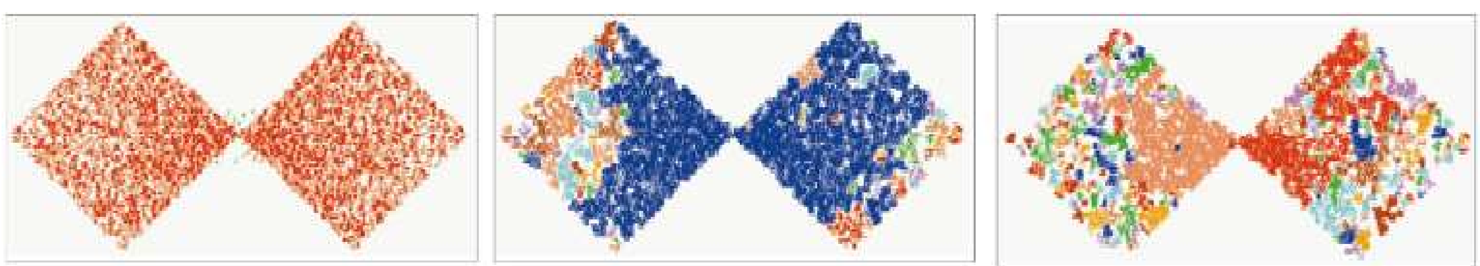
DBSCAN: Sensitive to Parameters

Figure 8. DBScan results for DS1 with MinPts at 4 and Eps at (a) 0.5 and (b) 0.4.



(a) (b)

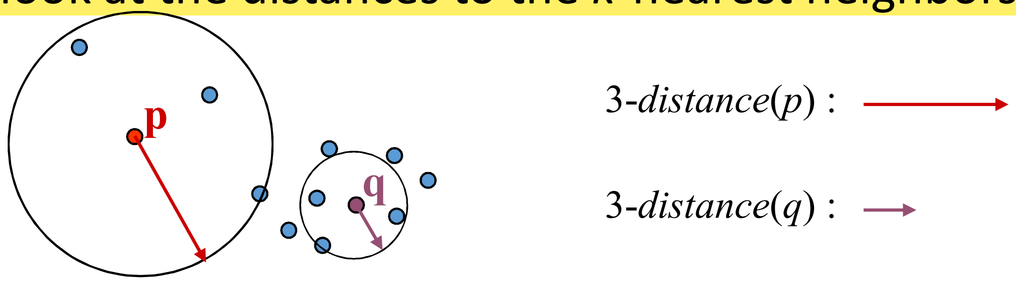
Figure 9. DBScan results for DS2 with MinPts at 4 and Eps at (a) 5.0, (b) 3.5, and (c) 3.0.



(a) (b) (c)

Determining the Parameters ε and *MinPts*

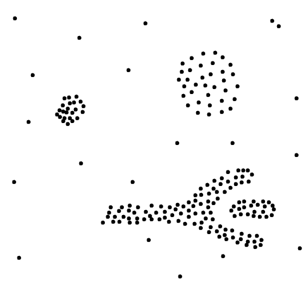
- Cluster: Point density higher than specified by ε and *MinPts*
- Idea: use the point density of the least dense cluster in the data set as parameters – but how to determine this?
- Heuristic: look at the distances to the *k*-nearest neighbors

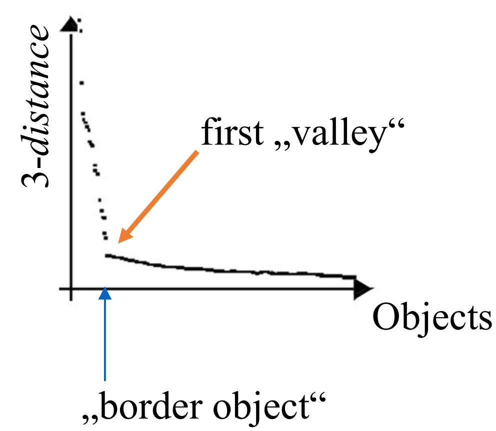


- Function *k-distance*(*p*): distance from *p* to the its *k*-nearest neighbor
- *k-distance plot*: *k*-distances of all objects, sorted in decreasing order

Determining the Parameters ε and *MinPts*

- Example *k*-distance plot

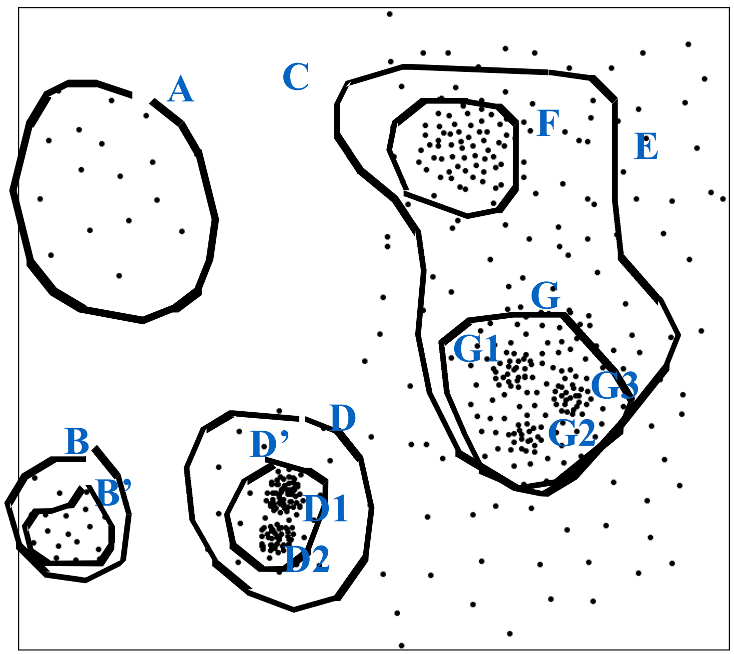


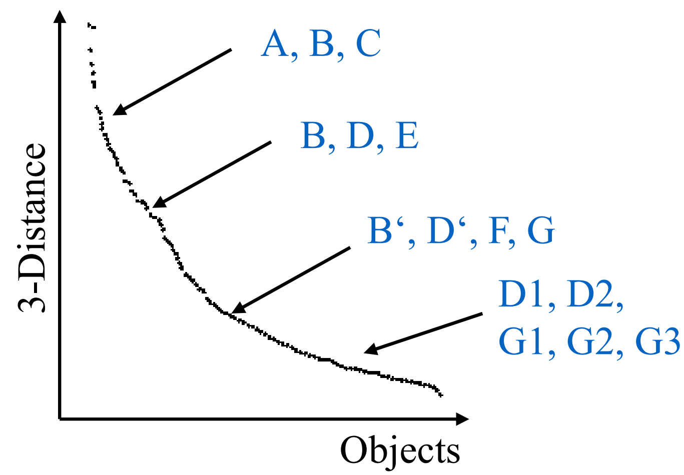


- Heuristic method:
  - Fix a value for *MinPts* (default: $2 \times d - 1$)
  - User selects “border object” *o* from the *MinPts-distance* plot;
    ε is set to *MinPts-distance*(o)

Determining the Parameters ε and *MinPts*

- Problematic example





Density Based Clustering: Discussion

- Advantages
  - Clusters can have arbitrary shape and size
  - Number of clusters is determined automatically
  - Can separate clusters from surrounding noise
  - Can be supported by spatial index structures
- Disadvantages
  - Input parameters may be difficult to determine
  - In some situations very sensitive to input parameter setting