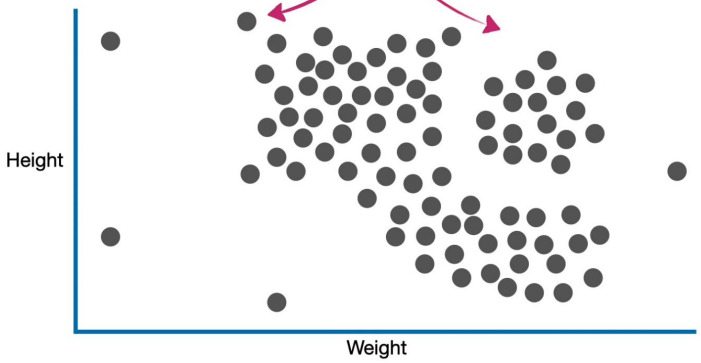
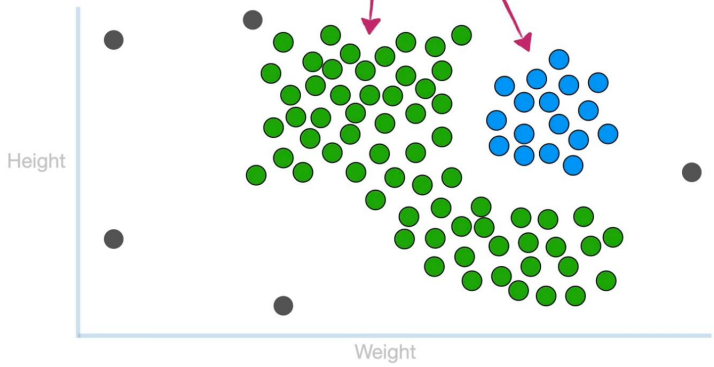
Now, imagine we collected **Weight** and **Height** measurements from a bunch of people…

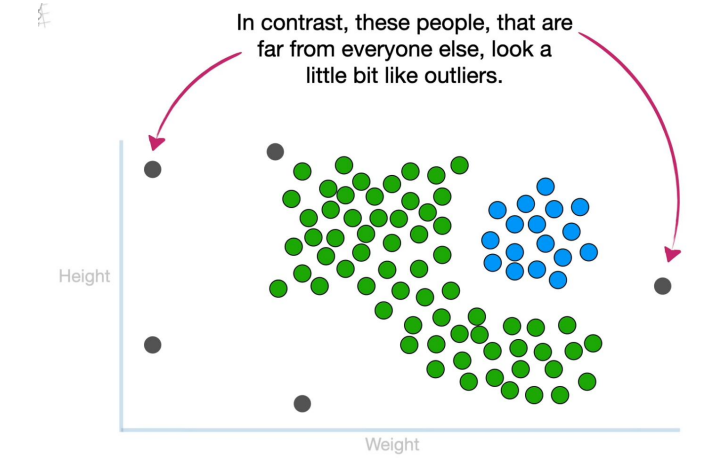
	Weight	Height
Person 1	56	150
Person 2	62	170
Person 3	71	168
...	...	...

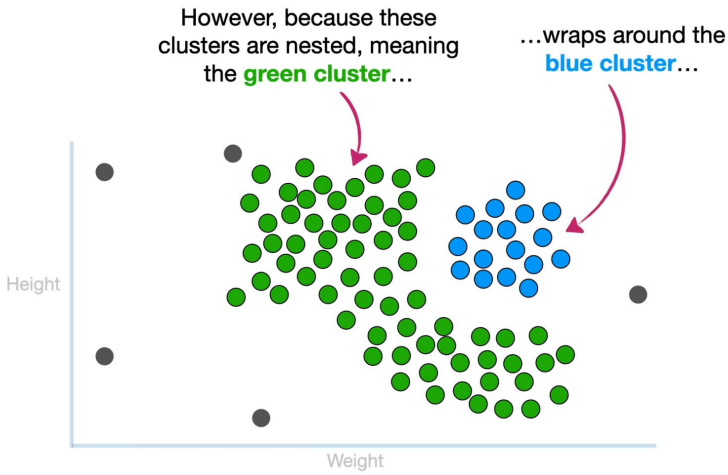
…and we plotted the people on a **2**-dimensional graph like this…

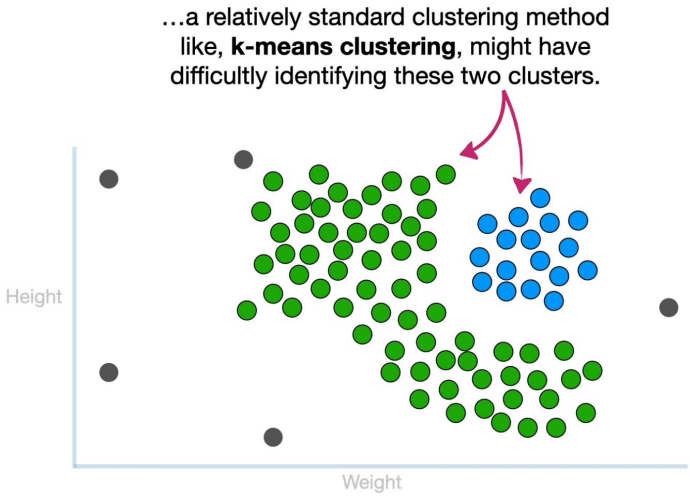


By eye, we can see **2** different clusters…

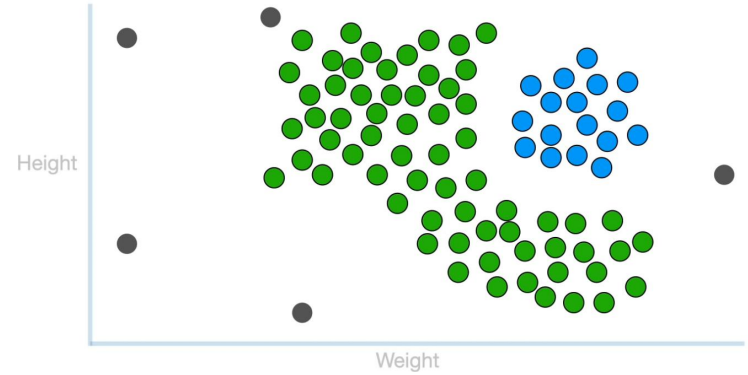



In contrast, these people, that are far from everyone else, look a little bit like outliers.
Height
Weight

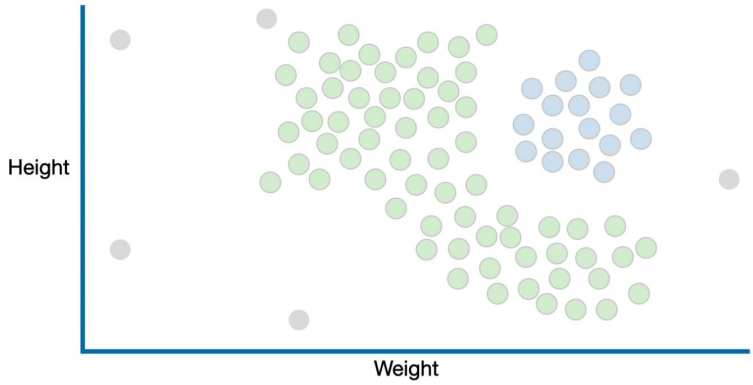

However, because these clusters are nested, meaning the green cluster…
…wraps around the blue cluster…
Height
Weight


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

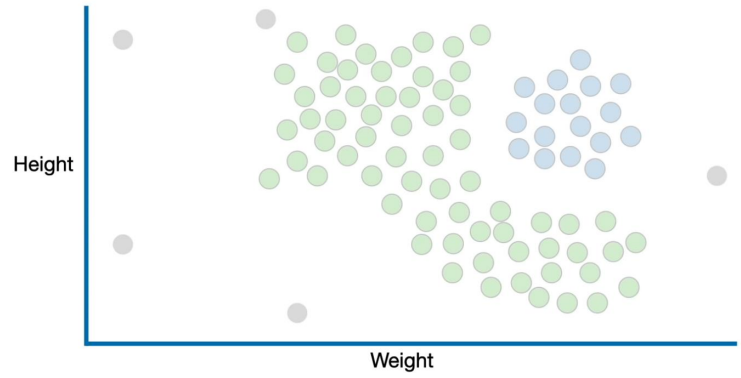
So we need a clustering algorithm that can handle nested clusters.



And if we can't draw and look at a **4** or more-dimensional graph, then we need a way identify nested clusters that we cannot see by eye.

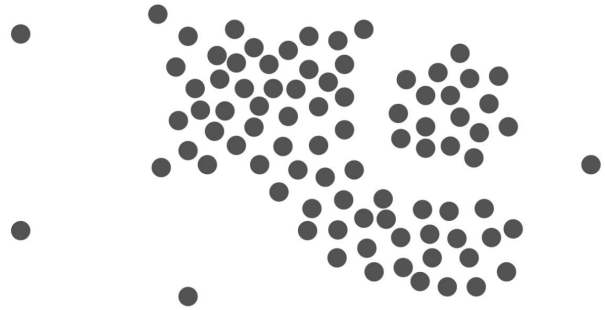


One of these algorithms is called **DBSCAN**, and that's what we will talk about today.

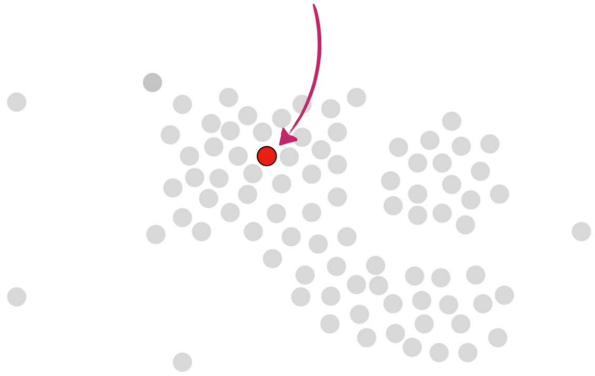


Step 1: Count the Number of Points

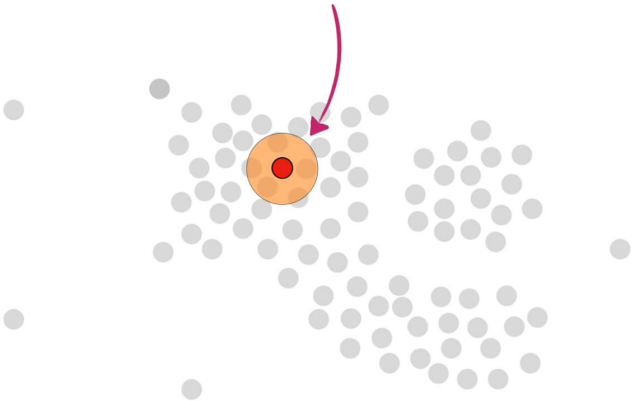
…the first thing we can do is count the number of points close to each point.



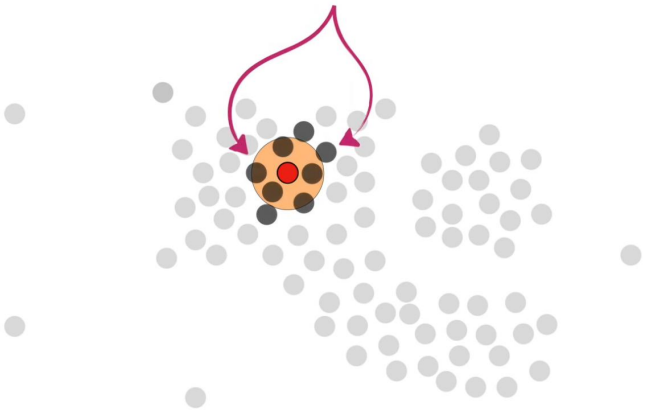
For example, if we start with this **red point**…



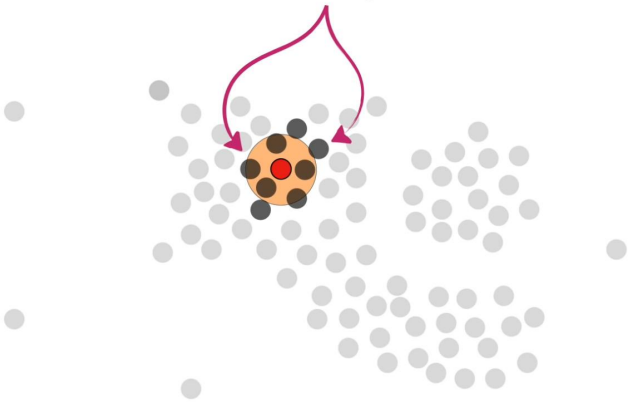
…and we draw an **orange circle** around it…



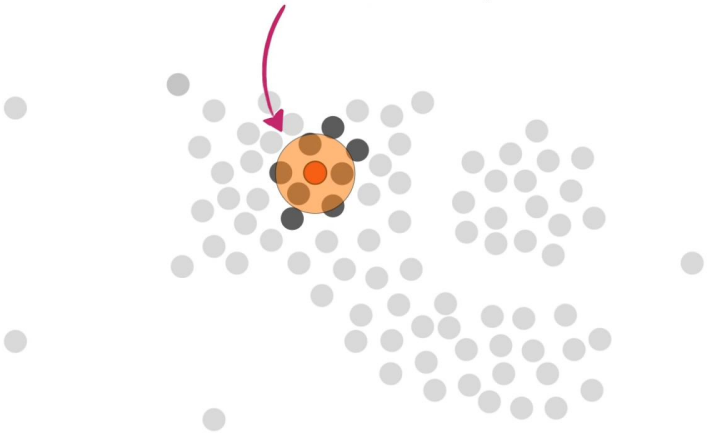
…then we see that the **orange circle** overlaps, at least partially, **8** other points.



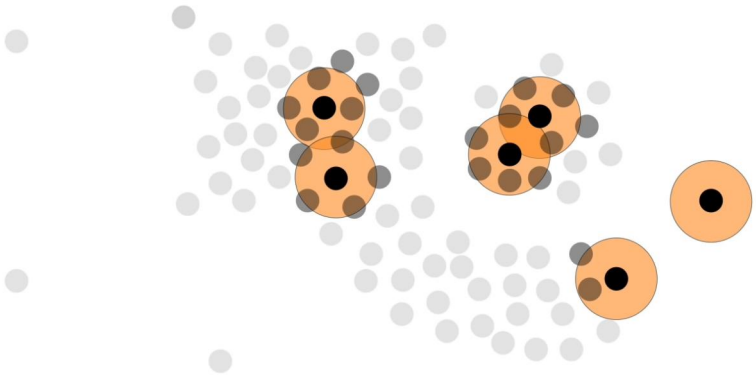
So the **red point** is close to **8** other points.



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

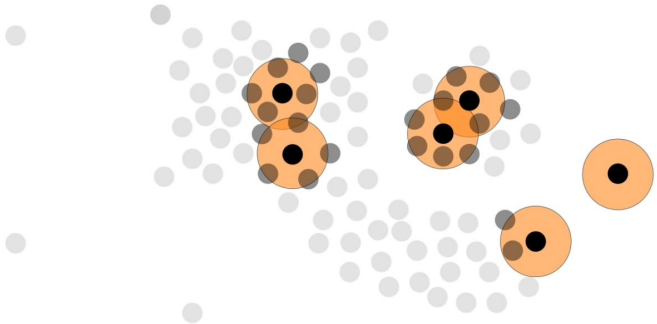


Likewise, for all of the remaining points, we counts the number of close points.

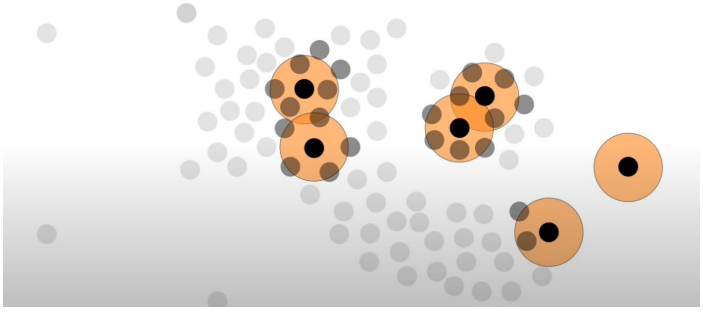


Step 2: Define the Core Point

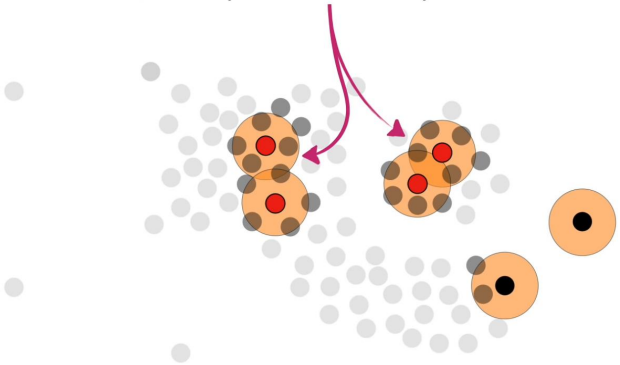
Now, in this example, we will define a **Core Point** to be one that is close to at least **4** other points.



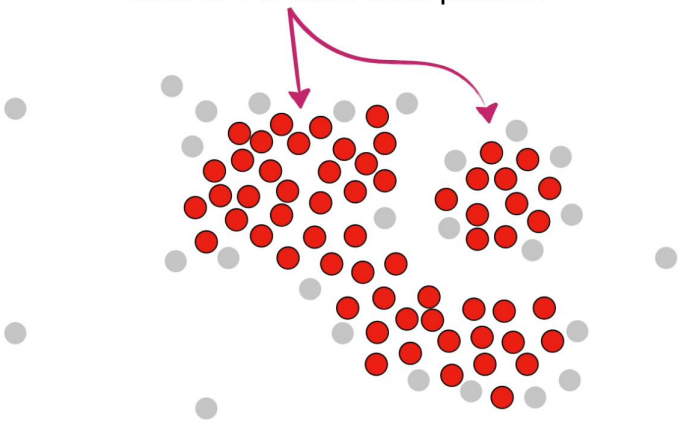
NOTE: The number of close points for a **Core Point** is user defined, so, when using **DBSCAN**, you might need to fiddle with this parameter as well.



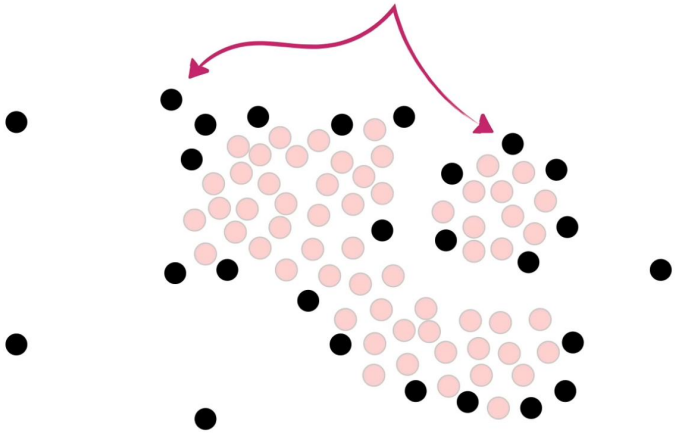
Anyway, these **4** points are some of the **Core Points**, because their **orange circles** overlap at least **4** other points…



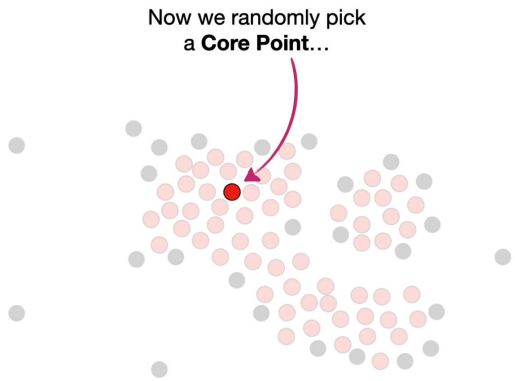
Ultimately, we can call all of these **red points** **Core Points** because they are all close to **4** or more other points…

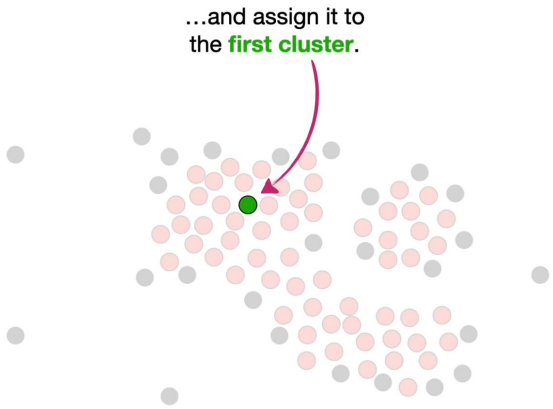


…and the remaining points are **Non-Core**.

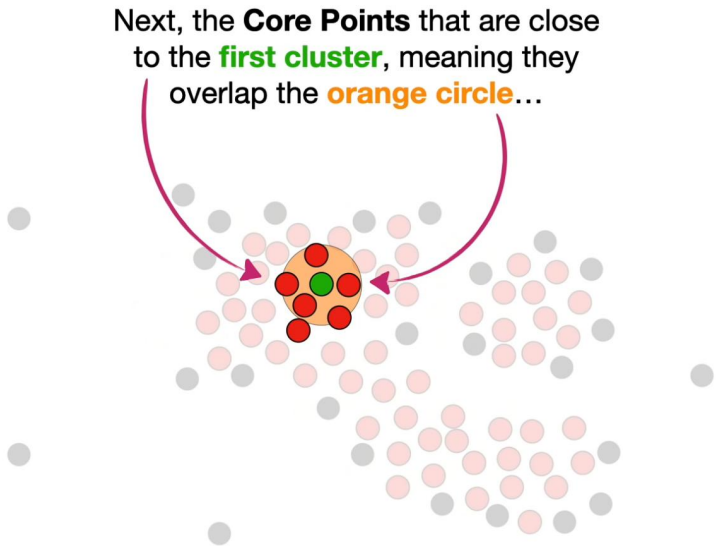


Step 3: Randomly pick a core point as first cluster

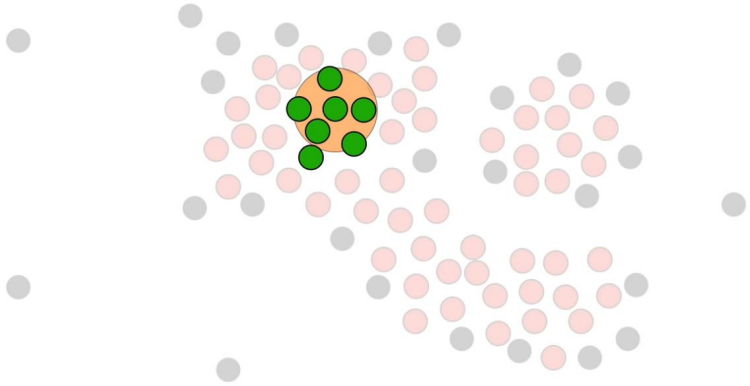




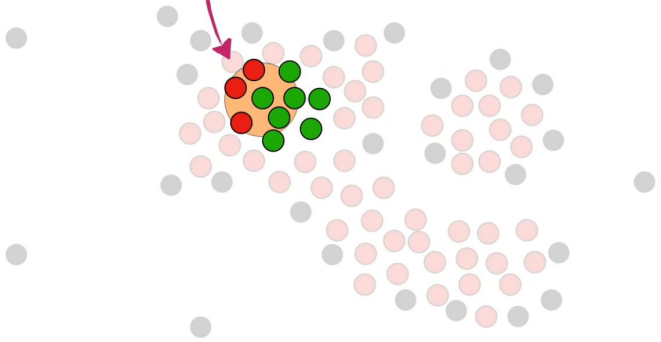
Step 4: Find all the core points that is close to the first cluster, and merge them into the first cluster, keep this finding and merge process until we can't find any core points that can be merged into first cluster anymore



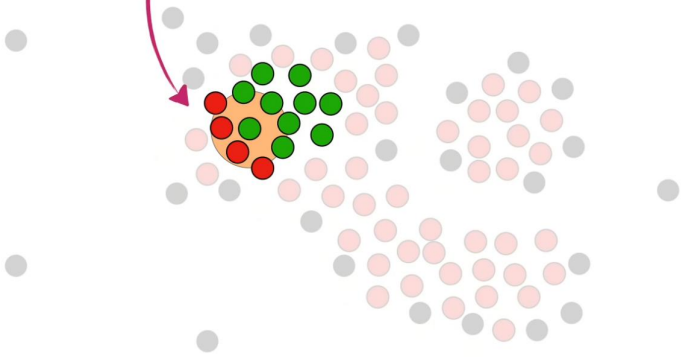
…are all added to the **first cluster**.



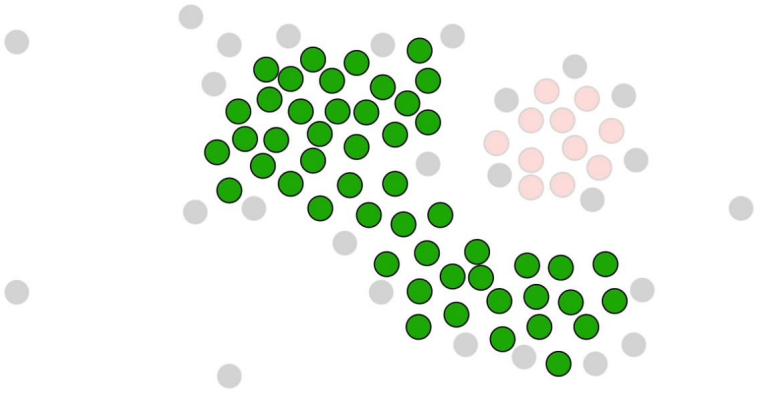
Then the **Core Points** that are close to the growing **first cluster** join it…



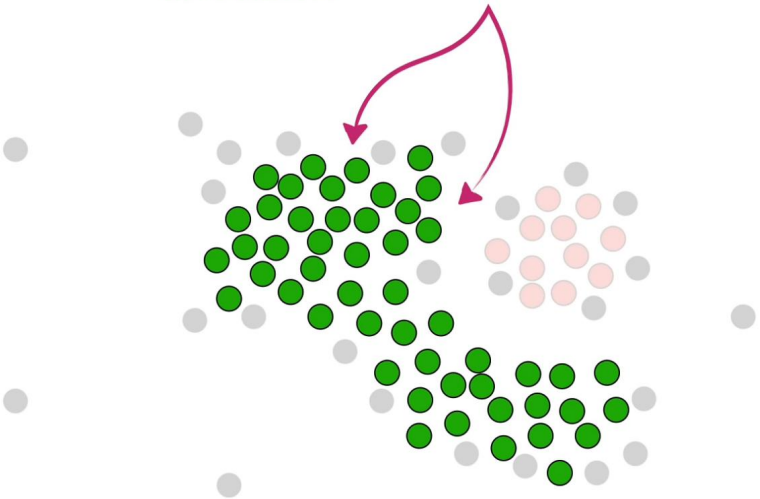
…and extend it to other **Core Points** that are close by.



Ultimately, all of the **Core Points** that are close to the growing **first cluster** are added to it and then used to extend it further.

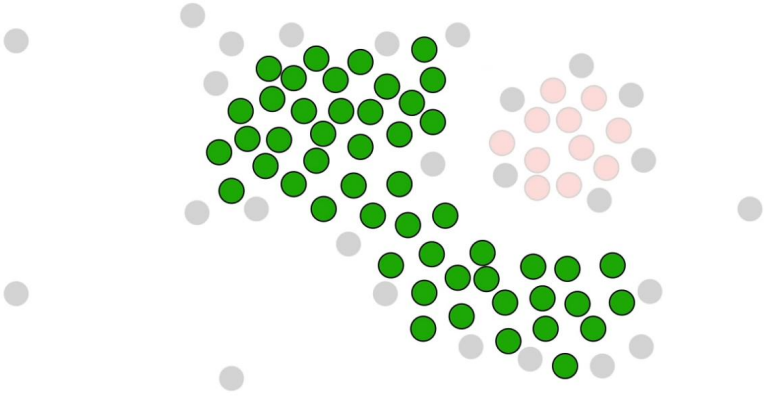


NOTE: At this point, every single point in the **first cluster** is a **Core Point**…

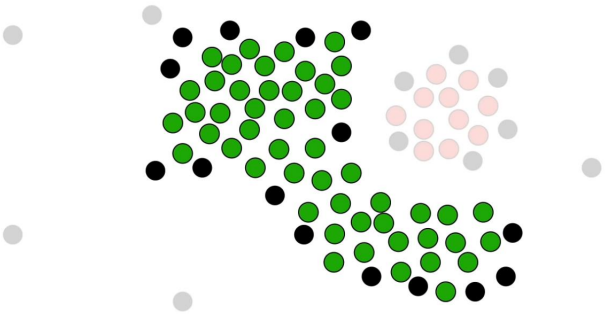


Step 5: Add non-core points into first cluster if they are close to it

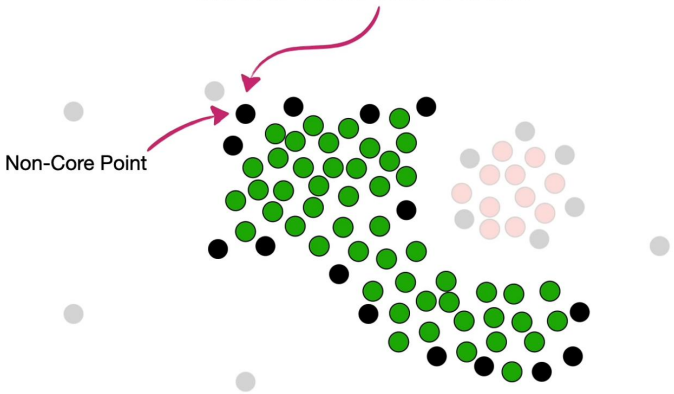
…and because we can no longer add any more **Core Points** to the **first cluster**…

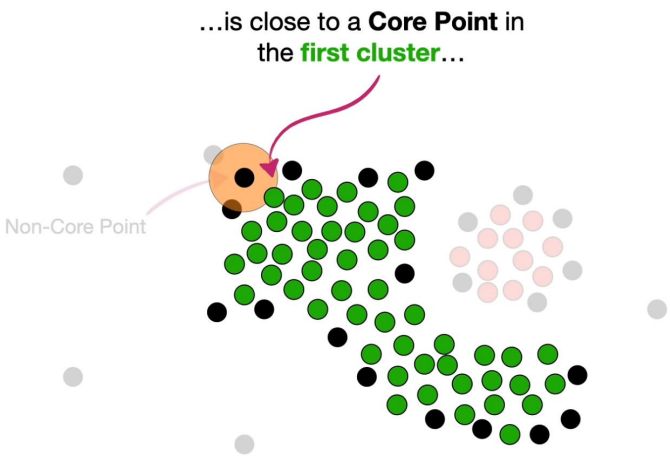


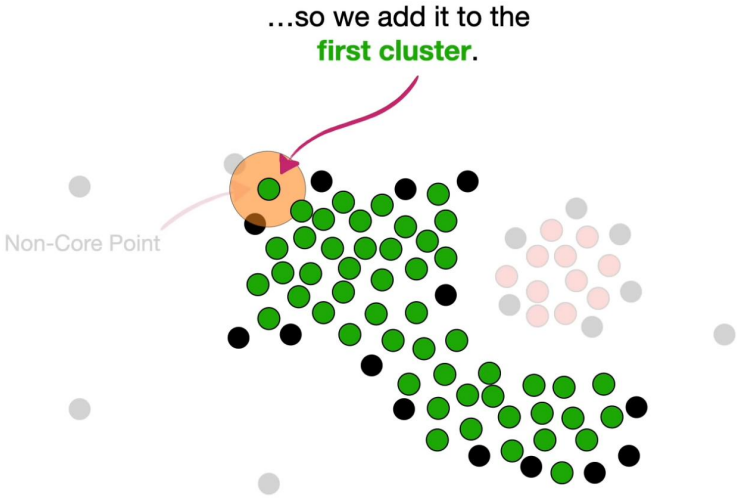
…we add all of the **Non-Core Points** that are close to **Core Points** in the **first cluster**.

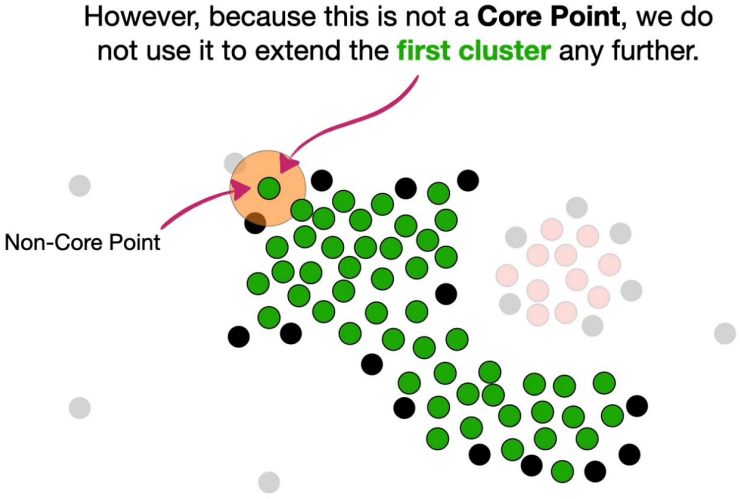


For example, this point. which is a **Non-Core Point**…

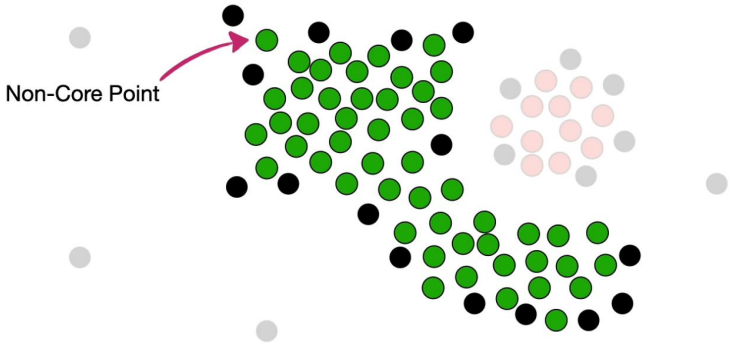



…is close to a **Core Point** in
the **first cluster**…
Non-Core Point

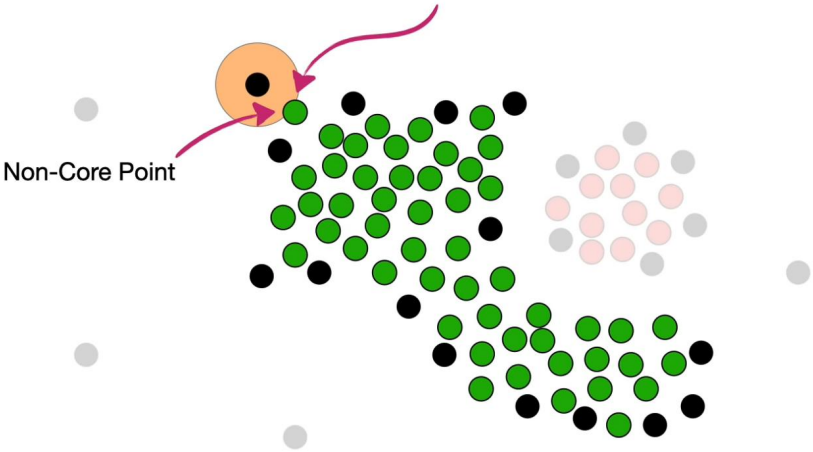

…so we add it to the
first cluster.
Non-Core Point


However, because this is not a **Core Point**, we do
not use it to extend the **first cluster** any further.
Non-Core Point

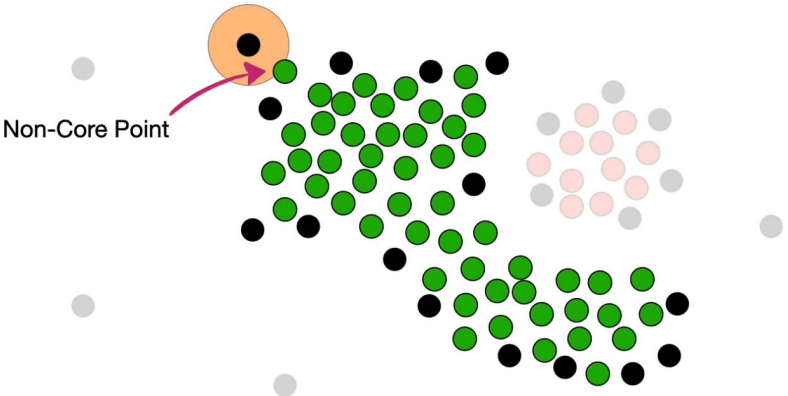
That means that this other **Non-Core Point**…

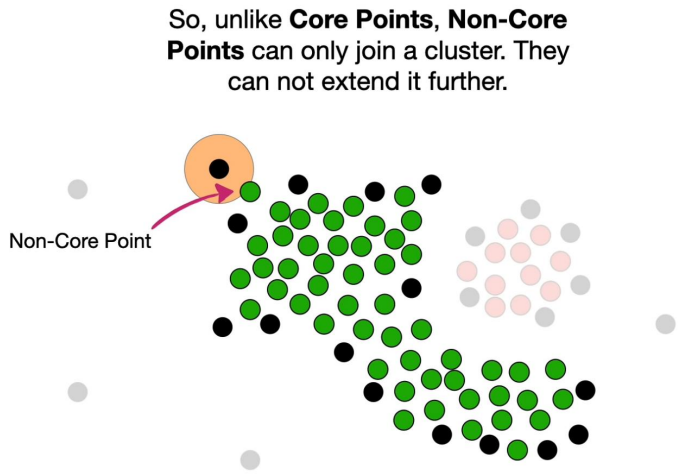


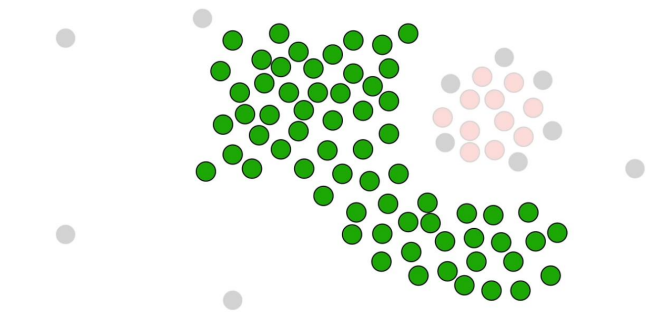
…which is close to the **Non-Core Point** that was just made part of the **first cluster**…



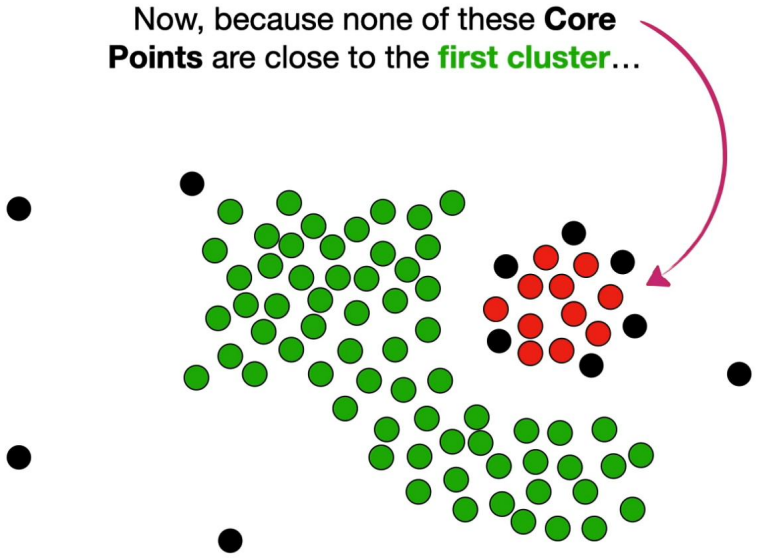
…will not be added to the **first cluster** because it is not close to a **Core Point**.



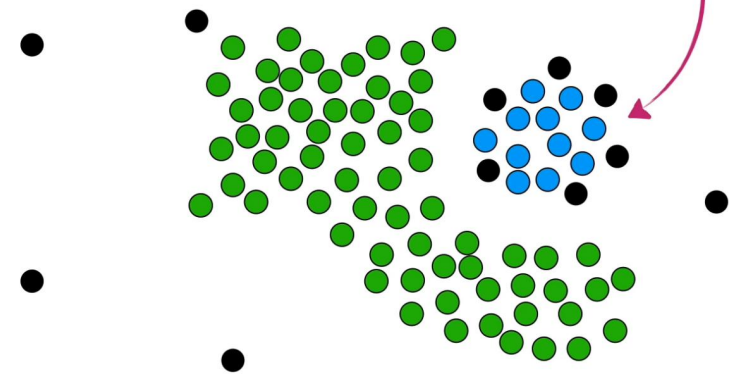




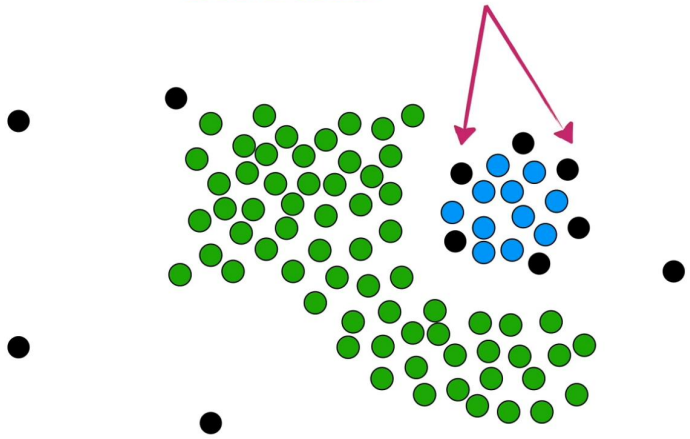
Step 6: Randomly pick a remaining core point as the second cluster and follow above steps to create the second cluster



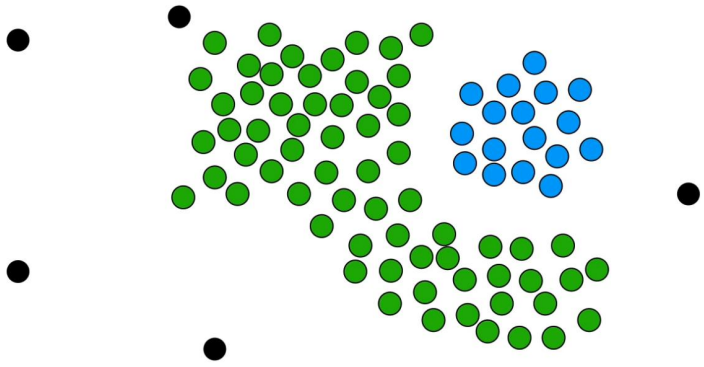
…they form a new, **second cluster** because they are close to each other…



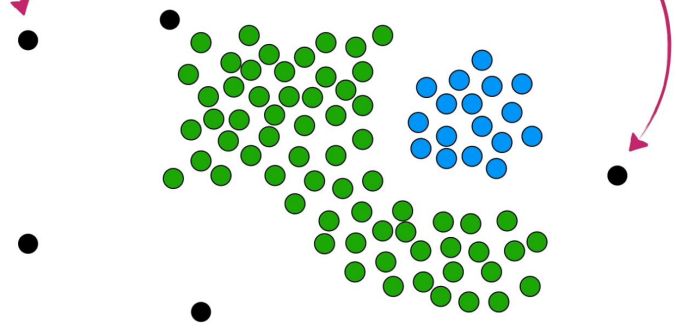
…and the **Non-Core Points** that are close to the **second cluster** are added to it.



Lastly, because all of **Core Points** have been assigned to a cluster, we're done making new clusters…



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



…are *not* added to clusters and called **outliers**…