Reading: Clustering beyond K-means

As you now know, K-means is a powerful and straightforward way to group data based on its proximity to other data. However, as with all models, K-means has its limitations. This reading will review the strengths of K-means and also demonstrate some of its weaknesses. Additionally, you'll learn about two additional clustering methodologies to explore as alternatives:

- DBSCAN
- Agglomerative clustering

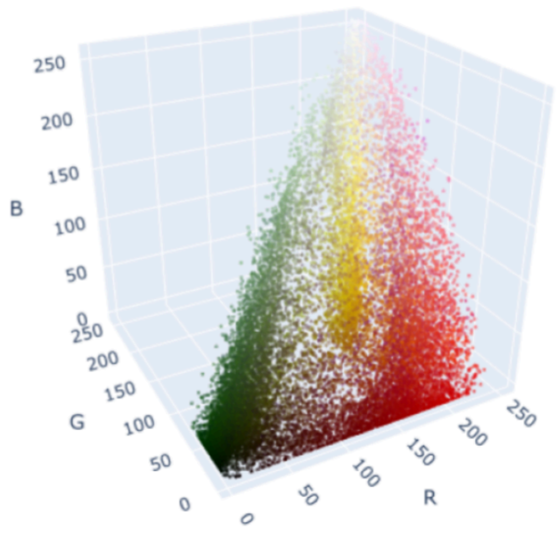
The purpose is not to make you an expert in clustering, but rather to give you a map of the terrain that you can use to guide you if you'd like to learn more about this branch of unsupervised learning.

Why use other algorithms?

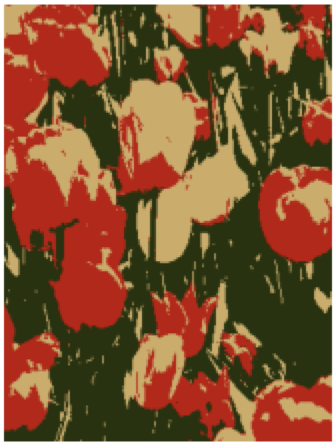
In the K-means demonstration video, you saw how the K-means algorithm clustered the pixels of a photograph of some tulips. Recall that the color of every pixel is represented by a vector with three values, each with a range of [0, 255]. These three values correspond to the amounts of red, green, and blue (RGB) that are combined to make the color of each pixel. The RGB values can be plotted in a three-dimensional space in order to visualize how the colors in the photograph are related to each other.

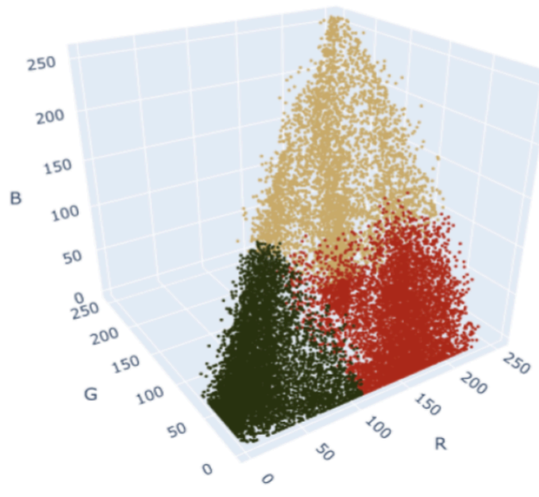
In this way, you could go from this:

To this:

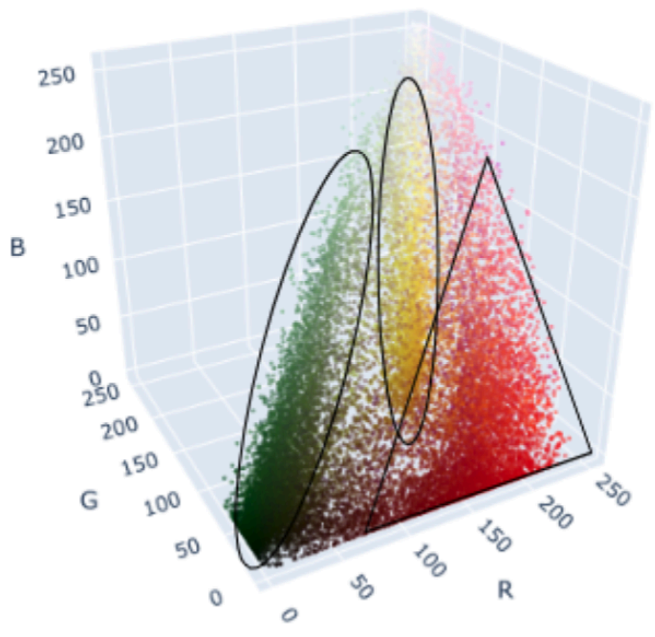


From this graph, it is clear that all of the pixels fall within a space that resembles a lopsided pyramid. In this same demonstration, the pixels were grouped into three clusters using K-means. When the RGB values for each pixel were replaced with those of its nearest centroid, the results were this:





Perhaps you were thinking that you would have clustered this data differently. After all, there do appear to be denser areas of points along the vertices of the pyramid:

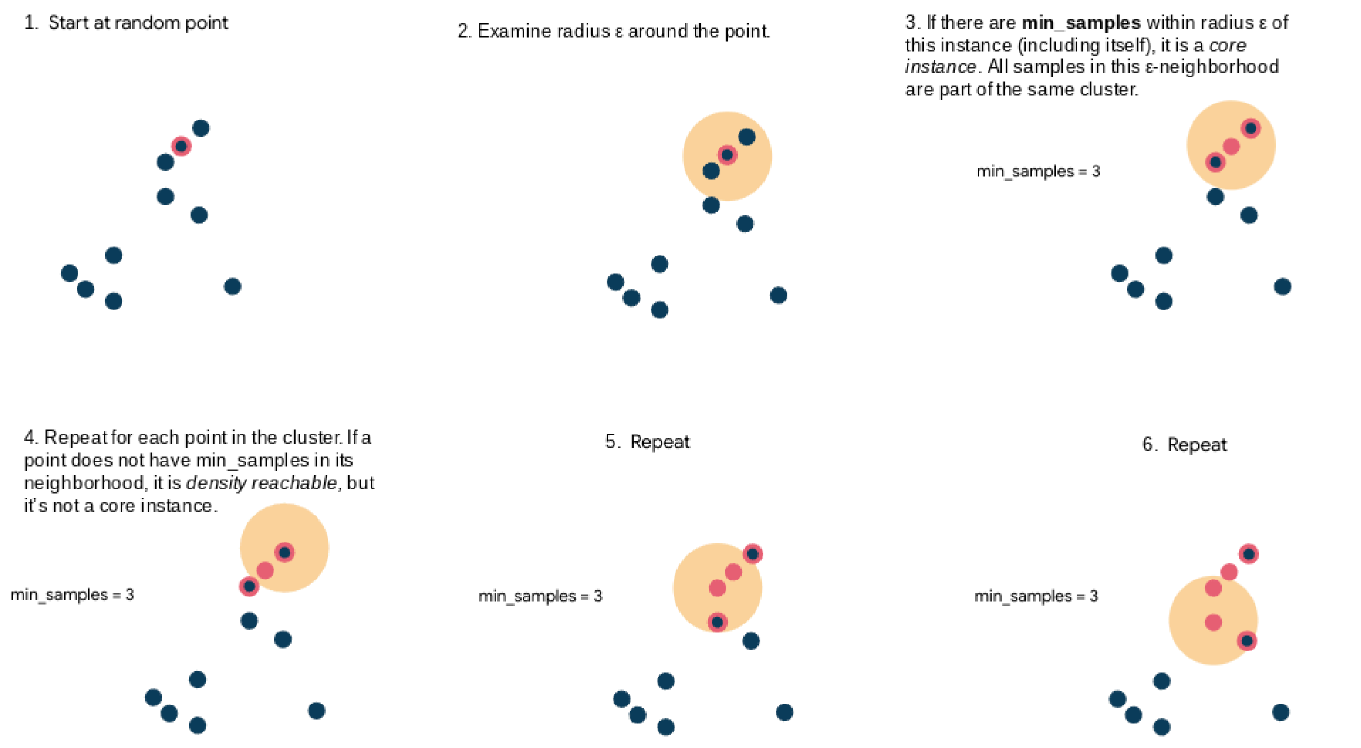


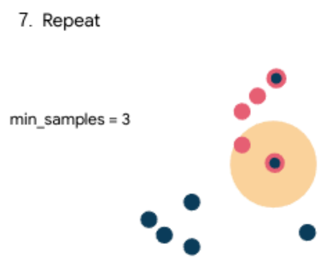
So why were the K-means clusters so boxy? Well, this is because K-means works by minimizing intercluster variance. In other words, it aims to minimize the distance between points and their centroids. This means that K-means works best when the clusters are round. If you aren't satisfied with the way K-means is clustering your data, don't worry, there are many other clustering methods available to choose from.

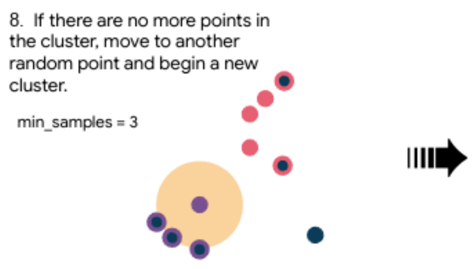
DBSCAN

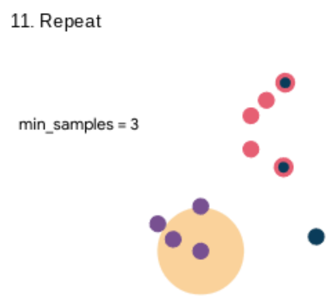
DBSCAN stands for density-based spatial clustering of applications with noise. Instead of trying to minimize variance between points in each cluster, DBSCAN searches your data space for continuous regions of high density. Here's how it works.

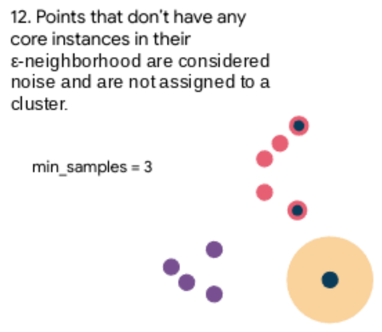
Note: You can also find text alternative versions of these conversations in the DBSCAN addendum transcript.


1. Start at random point
2. Examine radius ε around the point.
3. If there are **min_samples** within radius ε of this instance (including itself), it is a *core instance*. All samples in this ε-neighborhood are part of the same cluster.
min_samples = 3
4. Repeat for each point in the cluster. If a point does not have min_samples in its neighborhood, it is *density reachable*, but it's not a core instance.
min_samples = 3
5. Repeat
min_samples = 3
6. Repeat
min_samples = 3




8. If there are no more points in the cluster, move to another random point and begin a new cluster.
min_samples = 3




12. Points that don't have any core instances in their ε-neighborhood are considered noise and are not assigned to a cluster.
min_samples = 3

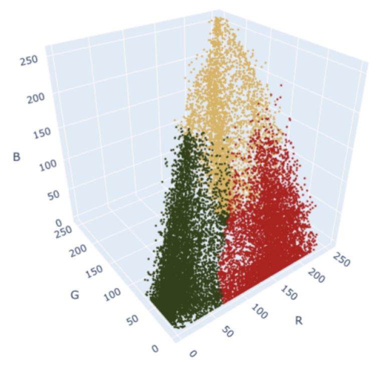
Hyperparameters

The most important hyperparameters for DBSCAN in scikit-learn are:

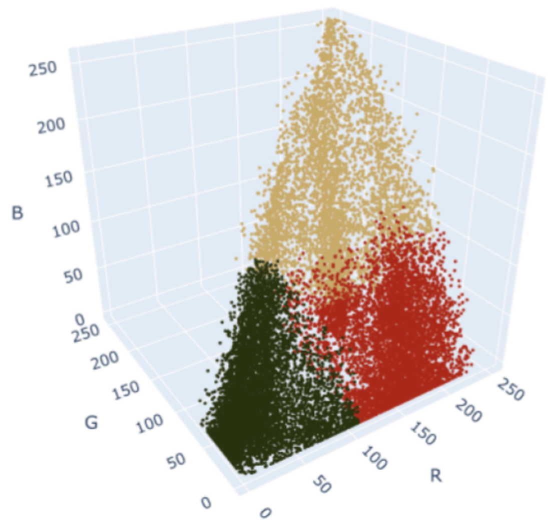
- **eps:** Epsilon (ε) - The radius of your search area from any given point
- **min_samples:** The number of samples in an ε-neighborhood for a point to be considered a core point (including itself)

Since DBSCAN is designed to find clusters based on density, the shape of the cluster isn't as important as it is for K-means. Here's what the tulips data looks like when clustered using DBSCAN compared to K-means:

DBSCAN



K-means

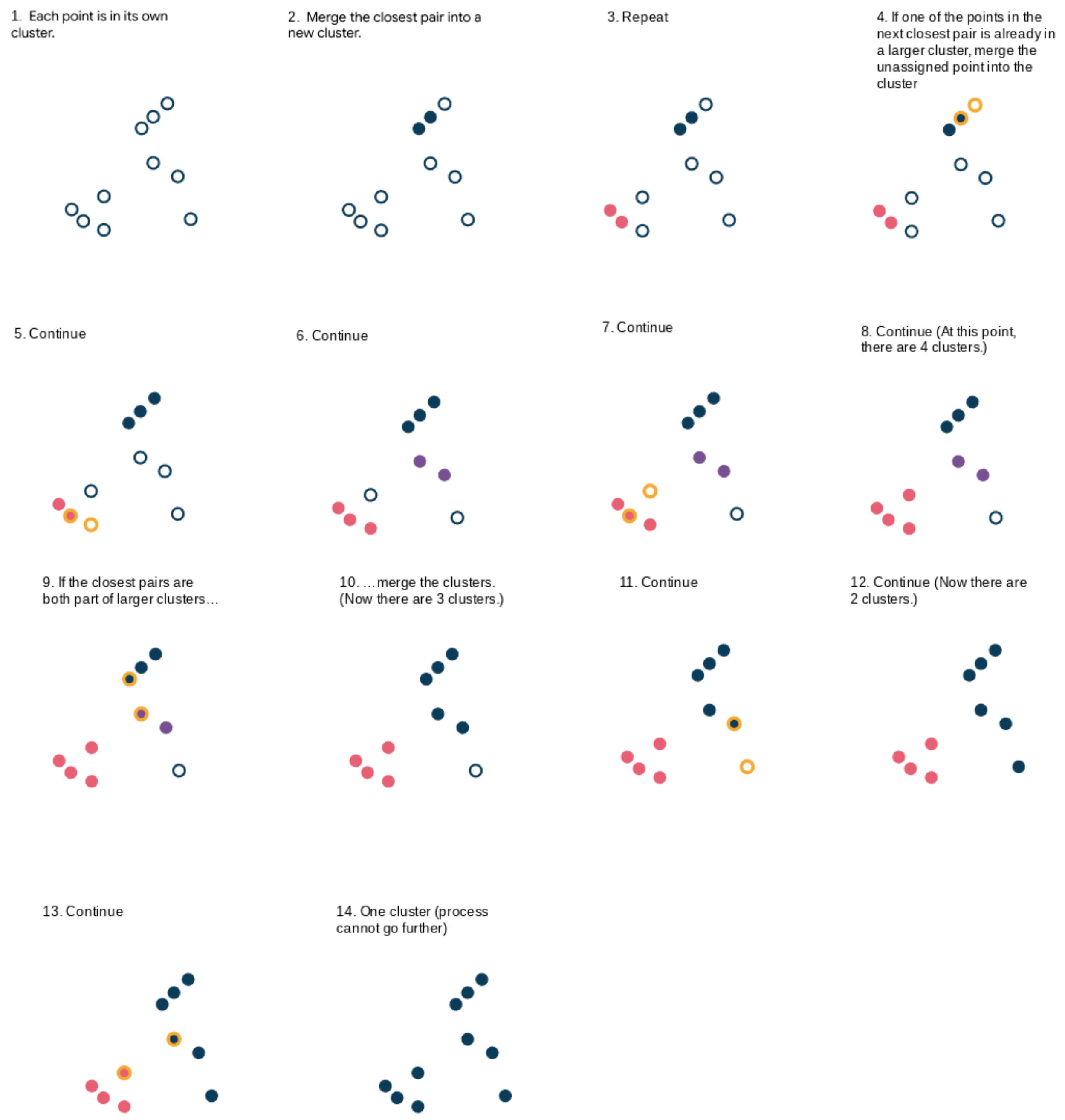


Notice that the clusters aren't as block-like as they were for K-means, and they correspond with the vertices of the pyramid a lot more closely than K-means. Also, of course, different clustering arrangements result in different colors for each of the three centroids.

Agglomerative clustering

Another way to cluster data is by using a technique called **agglomerative clustering.** Agglomerative clustering works by first assigning every point to its own cluster, then progressively combining clusters based on intercluster distance. Here's how it works.

Note: You can also find text alternative versions of these conversations in the Agglomerative clustering transcript.

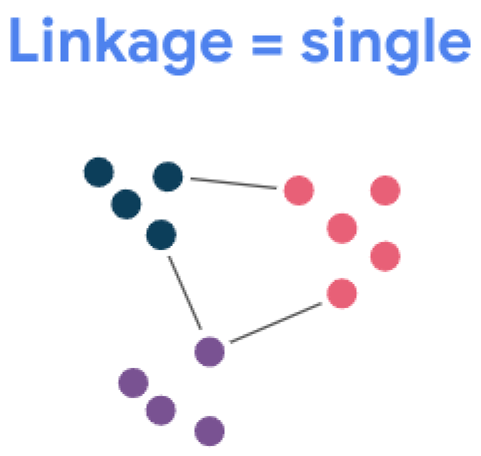

1. Each point is in its own cluster.
2. Merge the closest pair into a new cluster.
3. Repeat
4. If one of the points in the next closest pair is already in a larger cluster, merge the unassigned point into the cluster
5. Continue
6. Continue
7. Continue
8. Continue (At this point, there are 4 clusters.)
9. If the closest pairs are both part of larger clusters…
10. …merge the clusters. (Now there are 3 clusters.)
11. Continue
12. Continue (Now there are 2 clusters.)
13. Continue
14. One cluster (process cannot go further)

Agglomerative clustering requires that you specify a desired number of clusters or a distance threshold, which is the linkage distance (explained further in the next section) above which clusters will not be merged. If you do not specify a desired number of clusters, then the distance threshold is an important parameter, because without it the model would converge into a single cluster every time.

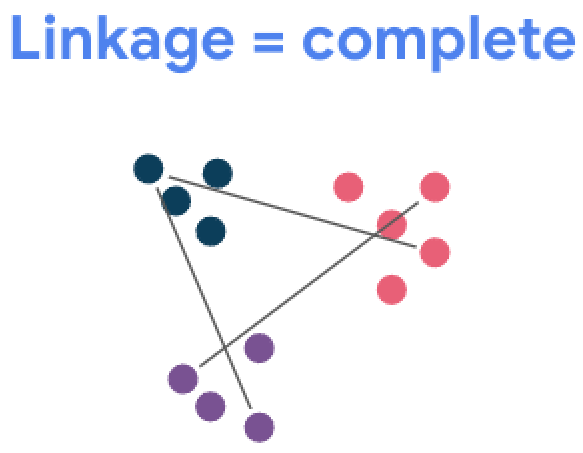
Linkage

There are different ways to measure the distances that determine whether or not to merge the clusters. This is known as the **linkage**. Here are some of the most common.

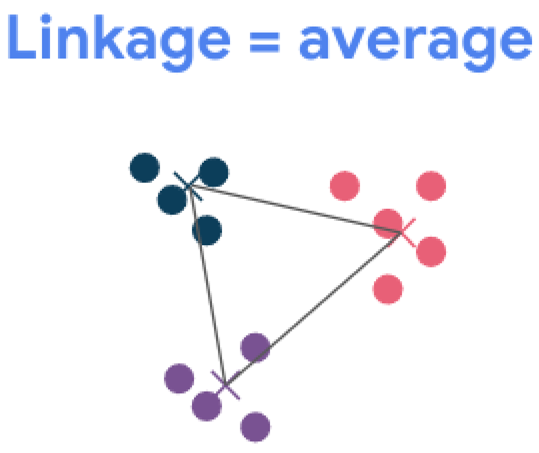
- **Single:** The minimum pairwise distance between clusters



- **Complete:** The maximum pairwise distance between clusters



- **Average:** The distance between each cluster's centroid and other clusters' centroids.



- **Ward:** This is not a distance measurement. Instead, it merges the two clusters whose merging will result in the lowest inertia.

Note that these linkage strategies aren't specific just to agglomerative clustering. You'll encounter them in many other clustering methodologies if you choose to continue learning about them, and even beyond clustering in other areas of data science.

When does it stop?

The agglomerative clustering algorithm will stop when one of the following conditions is met:

1. You reach a specified number of clusters.
2. You reach an intercluster distance threshold (clusters that are separated by more than this distance are too far from each other and will not be merged).

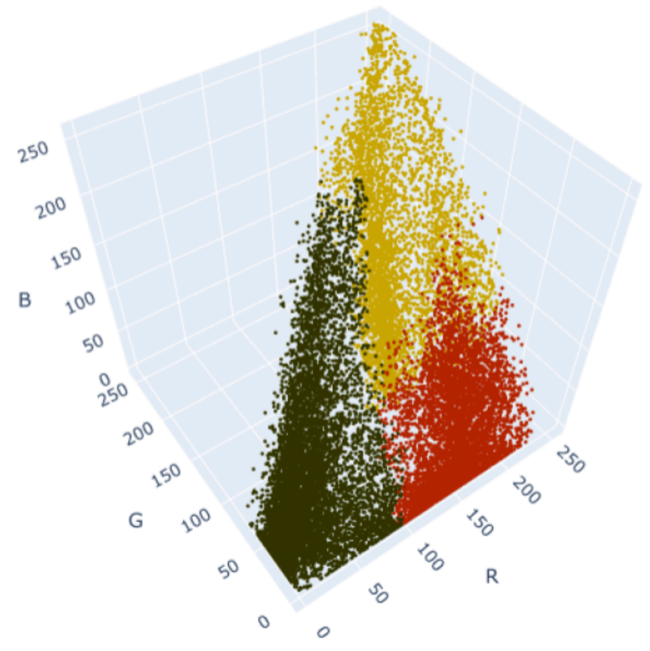
Hyperparameters

There are numerous hyperparameters available for agglomerative clustering in scikit-learn. These are some of the most important ones:

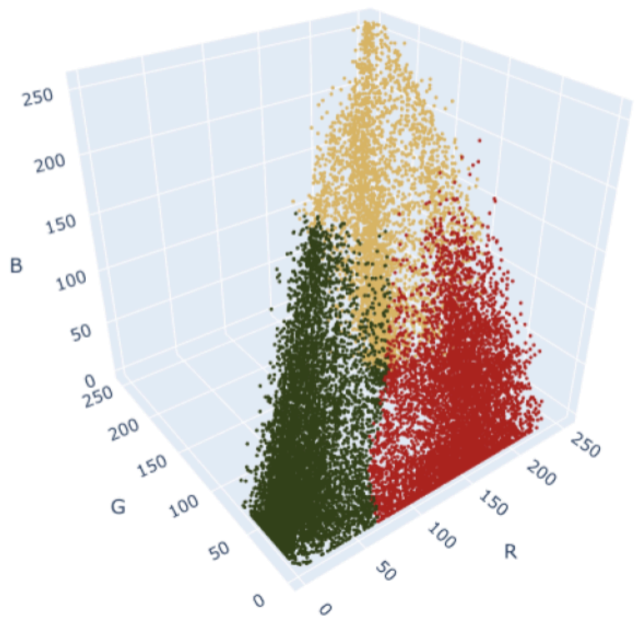
- **n_clusters:** The number of clusters you want in your final model
- **linkage:** The linkage method to use to determine which clusters to merge (as described above)
- **affinity:** The metric used to calculate the distance between clusters. Default = euclidean distance.
- **distance_threshold:** The distance above which clusters will not be merged (as described above)

Agglomerative clustering can be very effective. It scales reasonably well, and it can detect clusters of various shapes. Compare how agglomerative clustering performs on the tulip data compared to DBSCAN and K-means:

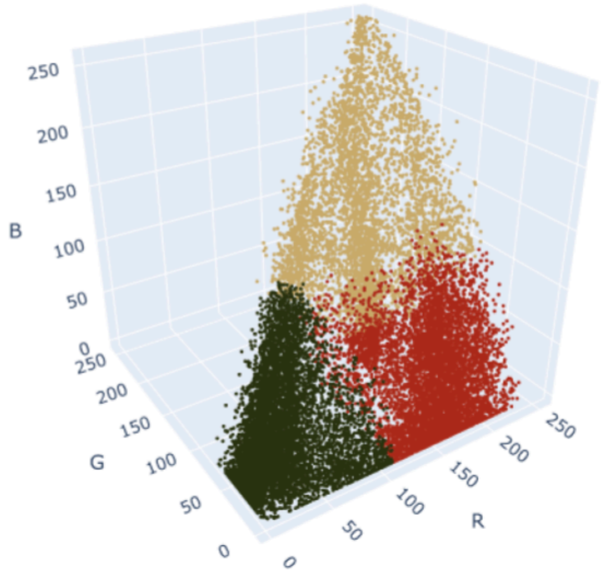
Agglomerative clustering:



DBSCAN



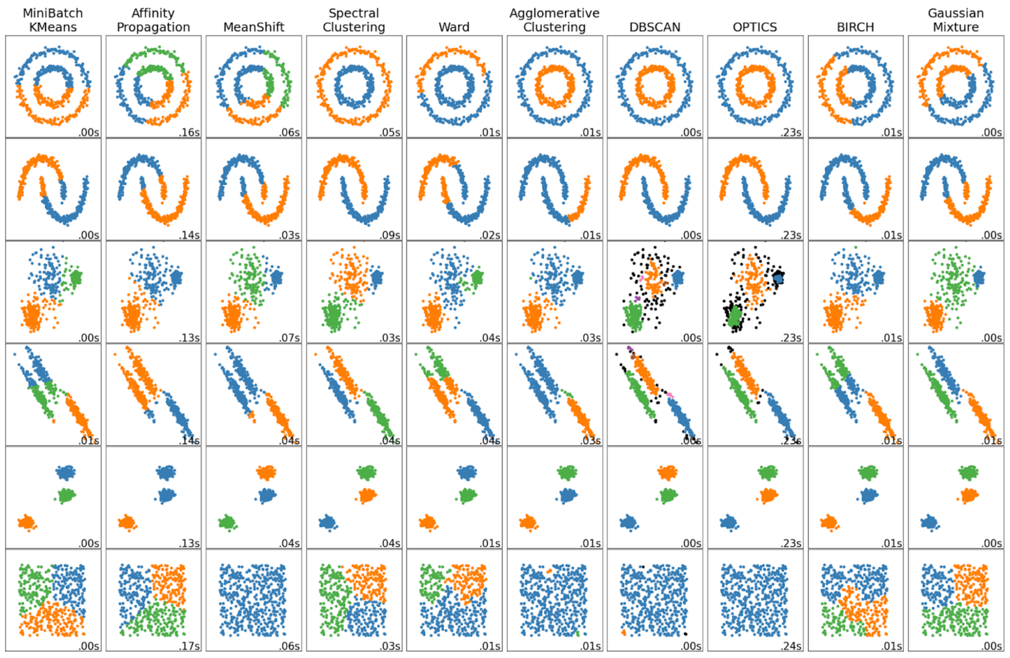
K-means



Agglomerative clustering gives even more definition to the data clustered along the vertices of the pyramid, which most closely represents the clusters that appear to the eye.

Other clustering algorithms

There are many other ways to cluster data than what is covered here. Scikit-learn's documentation provides a helpful reference that illustrates some of the strengths and weaknesses of each methodology by running them on a series of toy datasets.



As always, feel free to explore some of these algorithms on your own!

Key takeaways

K-means is a simple yet powerful clustering algorithm, but it's not the only one. Depending on the nature of the problem you're trying to solve, other algorithms might outperform it. DBSCAN and agglomerative clustering are two methodologies that are accessible to novices and effective, and they can cluster your data even when the clusters themselves have unusual shapes.

Resources for more information

You can find more information about scikit-learn's implementations of DBSCAN and agglomerative clustering in the documentation:

- DBSCAN
- Agglomerative clustering
