**Useful** clusters, on the other hand, serve as an intermediate step in a [data pipeline](https://en.wikipedia.org/wiki/Pipeline_(computing)). For example, businesses use clustering for customer segmentation. The clustering results segment customers into groups with similar purchase histories, which businesses can then use to create targeted advertising campaigns.

Selecting an appropriate clustering algorithm for your dataset is often difficult due to the number of choices available. Some important factors that affect this decision include the characteristics of the clusters, the features of the dataset, the number of outliers, and the number of data objects.

You’ll explore how these factors help determine which approach is most appropriate by looking at three popular categories of clustering algorithms:

1. Partitional clustering
2. Hierarchical clustering
3. Density-based clustering

**Partitional Clustering**

**Partitional clustering** divides data objects into nonoverlapping groups. In other words, no object can be a member of more than one cluster, and every cluster must have at least one object.

These techniques require the user to specify the number of clusters, indicated by the [variable](https://realpython.com/python-variables/) *k*. Many partitional clustering algorithms work through an iterative process to assign subsets of data points into *k* clusters. Two examples of partitional clustering algorithms are *k*-means and *k*-medoids.

These algorithms are both **nondeterministic**, meaning they could produce different results from two separate runs even if the runs were based on the same input.

Partitional clustering methods have several **strengths**:

* They work well when clusters have a **spherical shape**.
* They’re **scalable** with respect to algorithm complexity.

They also have several **weaknesses**:

* They’re not well suited for clusters with **complex shapes** and different sizes.
* They break down when used with clusters of different **densities**.

**Density-Based Clustering**

**Density-based clustering** determines cluster assignments based on the density of data points in a region. Clusters are assigned where there are high densities of data points separated by low-density regions.

Unlike the other clustering categories, this approach doesn’t require the user to specify the number of clusters. Instead, there is a distance-based parameter that acts as a tunable threshold. This threshold determines how close points must be to be considered a cluster member.

Examples of density-based clustering algorithms include Density-Based Spatial Clustering of Applications with Noise, or [**DBSCAN**](https://scikit-learn.org/stable/modules/generated/sklearn.cluster.DBSCAN.html), and Ordering Points To Identify the Clustering Structure, or [**OPTICS**](https://scikit-learn.org/stable/modules/generated/sklearn.cluster.OPTICS.html#sklearn.cluster.OPTICS).

The **strengths** of density-based clustering methods include the following:

* They excel at identifying clusters of **nonspherical shapes**.
* They’re resistant to **outliers**.

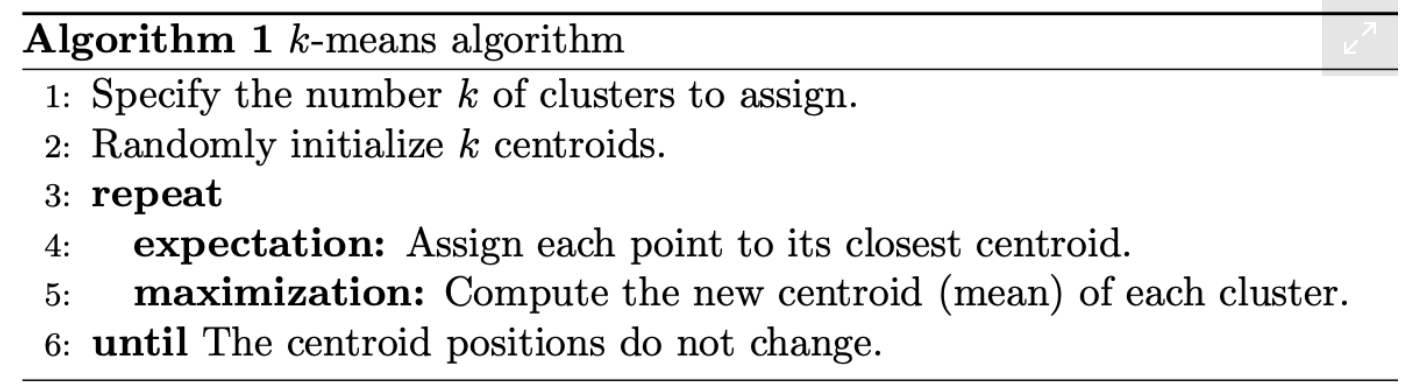
The **weaknesses** of density-based clustering methods include the following:

* They aren’t well suited for [clustering in **high-dimensional spaces**](https://en.wikipedia.org/wiki/Clustering_high-dimensional_data).
* They have trouble identifying clusters of **varying densities**.

### Understanding the K-Means Algorithm

Conventional k-means requires only a few steps. The first step is to randomly select k centroids, where k is equal to the number of clusters you choose. **Centroids** are data points representing the center of a cluster.

The main element of the algorithm works by a two-step process called **expectation-maximization**. The **expectation** step assigns each data point to its nearest centroid. Then, the **maximization** step computes the mean of all the points for each cluster and sets the new centroid. Here’s what the conventional version of the k-means algorithm looks like:



After choosing a number of clusters and the initial centroids, the expectation-maximization step is repeated until the centroid positions reach convergence and are unchanged.

The random initialization step causes the k-means algorithm to be **nondeterministic**, meaning that cluster assignments will vary if you run the same algorithm twice on the same dataset. Researchers commonly run several initializations of the entire k-means algorithm and choose the cluster assignments from the initialization with the lowest SSE.