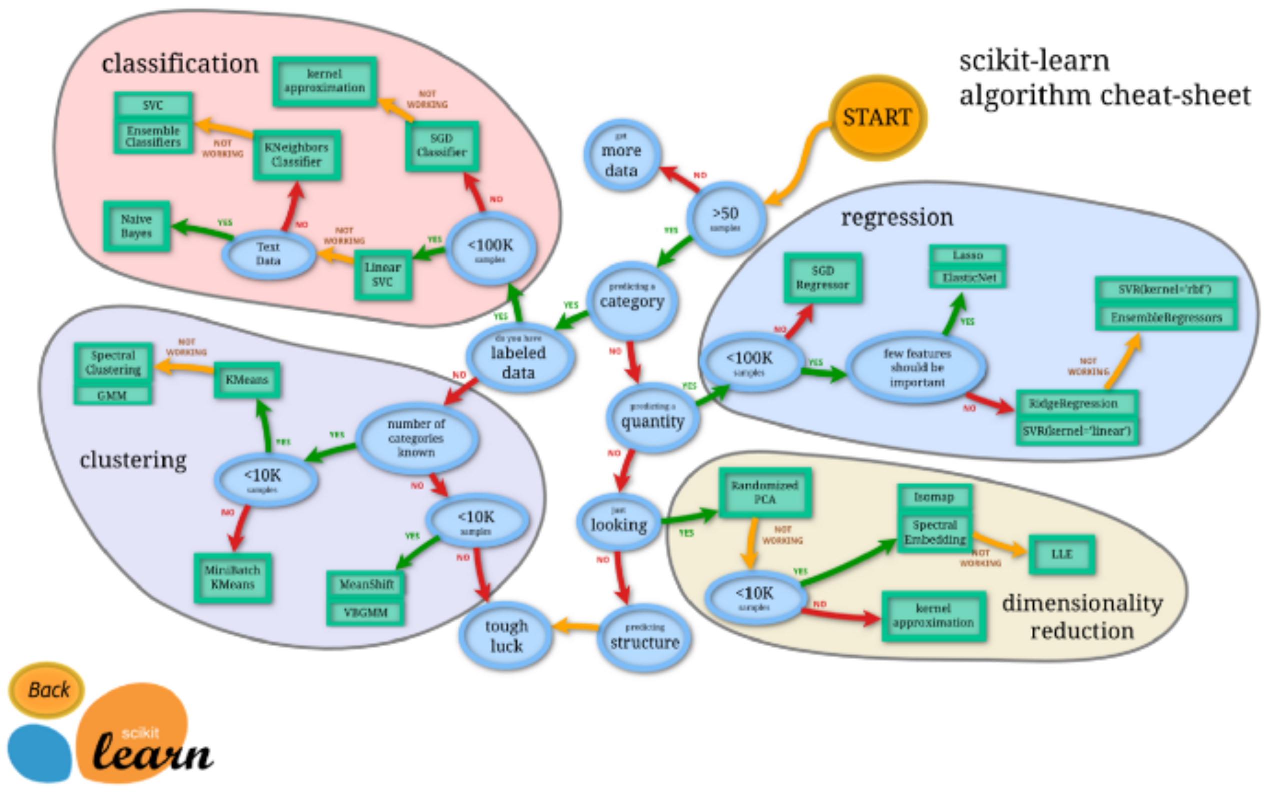
Get started Open in app

Follow 586K Followers

You have **1** free member-only story left this month. Sign up for Medium and get an extra one



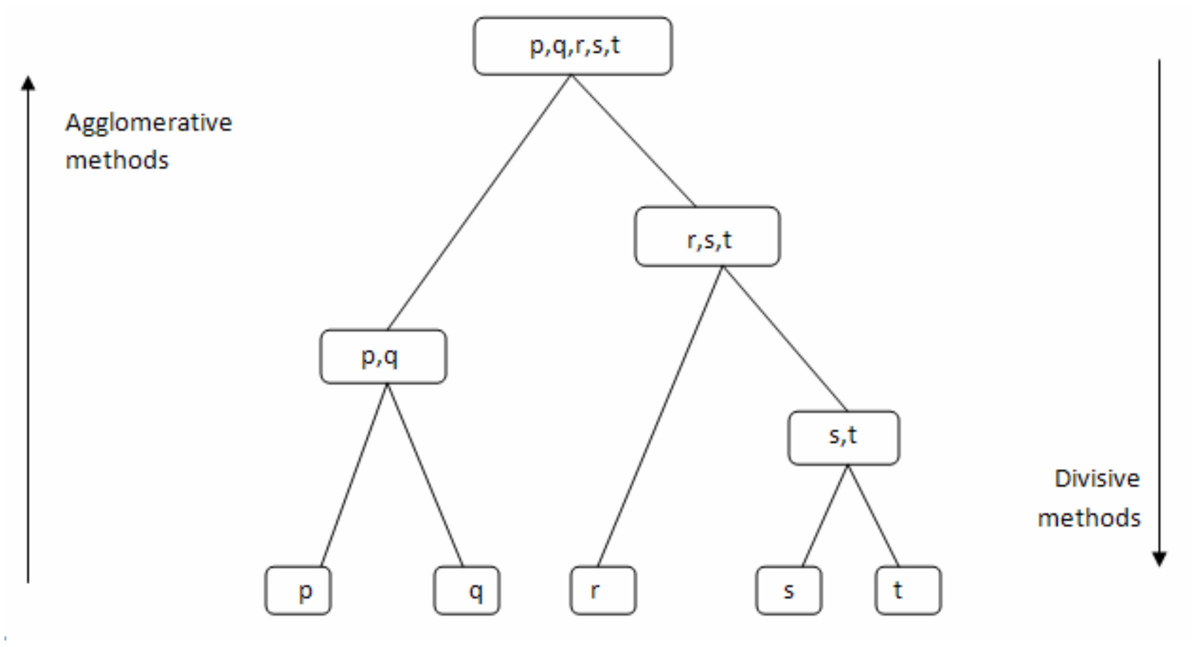
Hierarchical Agglomerative Clustering Algorithm **Example In Python**

Cory Maklin Dec 31, 2018 · 5 min read ★

Hierarchical clustering algorithms group similar objects into groups called **clusters**. There are two types of hierarchical clustering algorithms:

- Agglomerative — Bottom up approach. Start with many small clusters and merge them together to create bigger clusters.

- Divisive — Top down approach. Start with a single cluster than break it up into smaller clusters.



Some pros and cons of Hierarchical Clustering

Pros

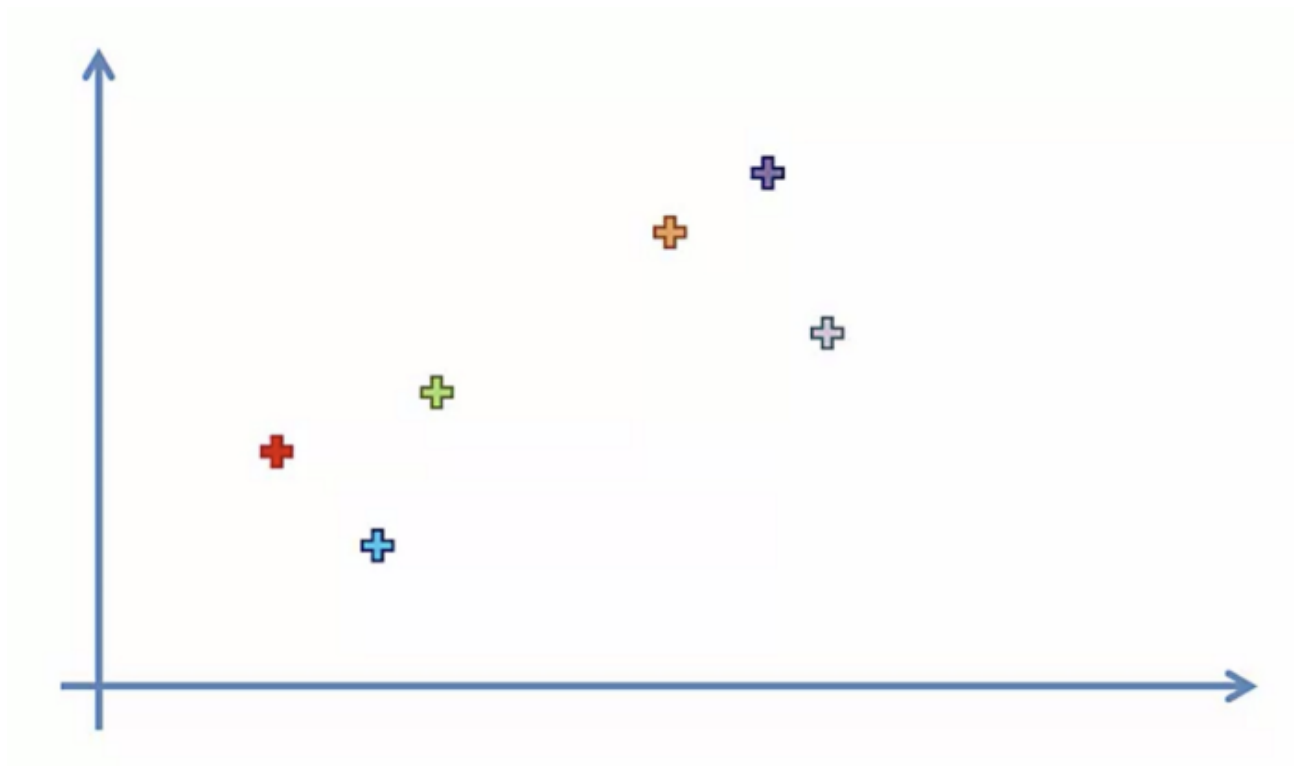
- No assumption of a particular number of clusters (i.e. k-means)
- May correspond to meaningful taxonomies

Cons

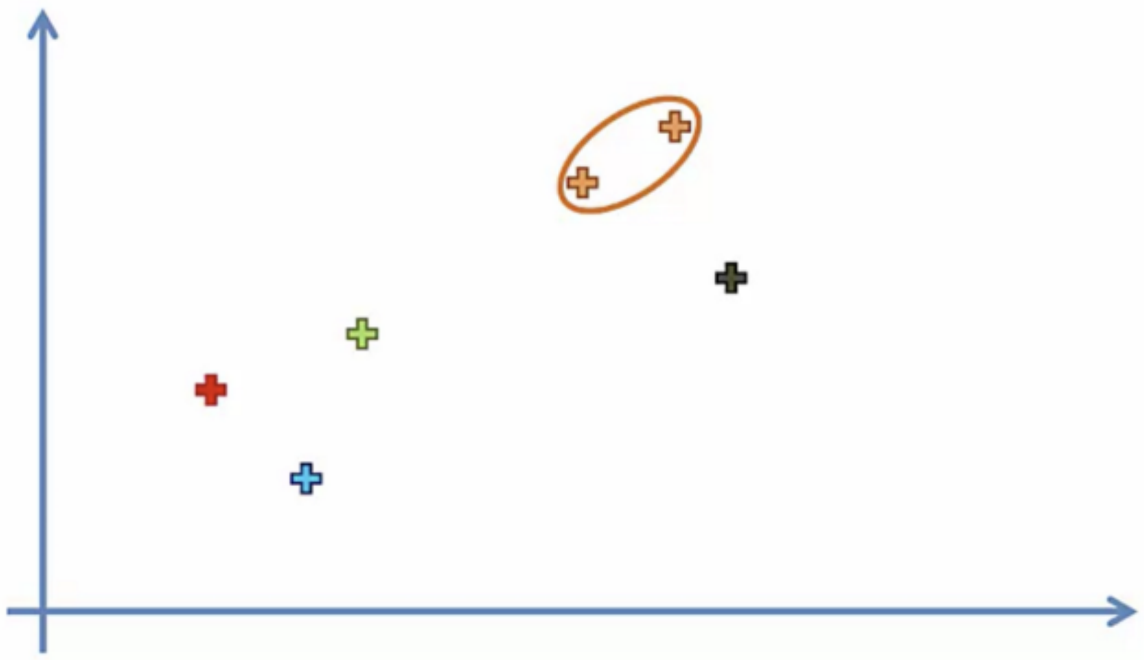
- Once a decision is made to combine two clusters, it can't be undone
- Too slow for large data sets, O(n2 log(n))

How it works

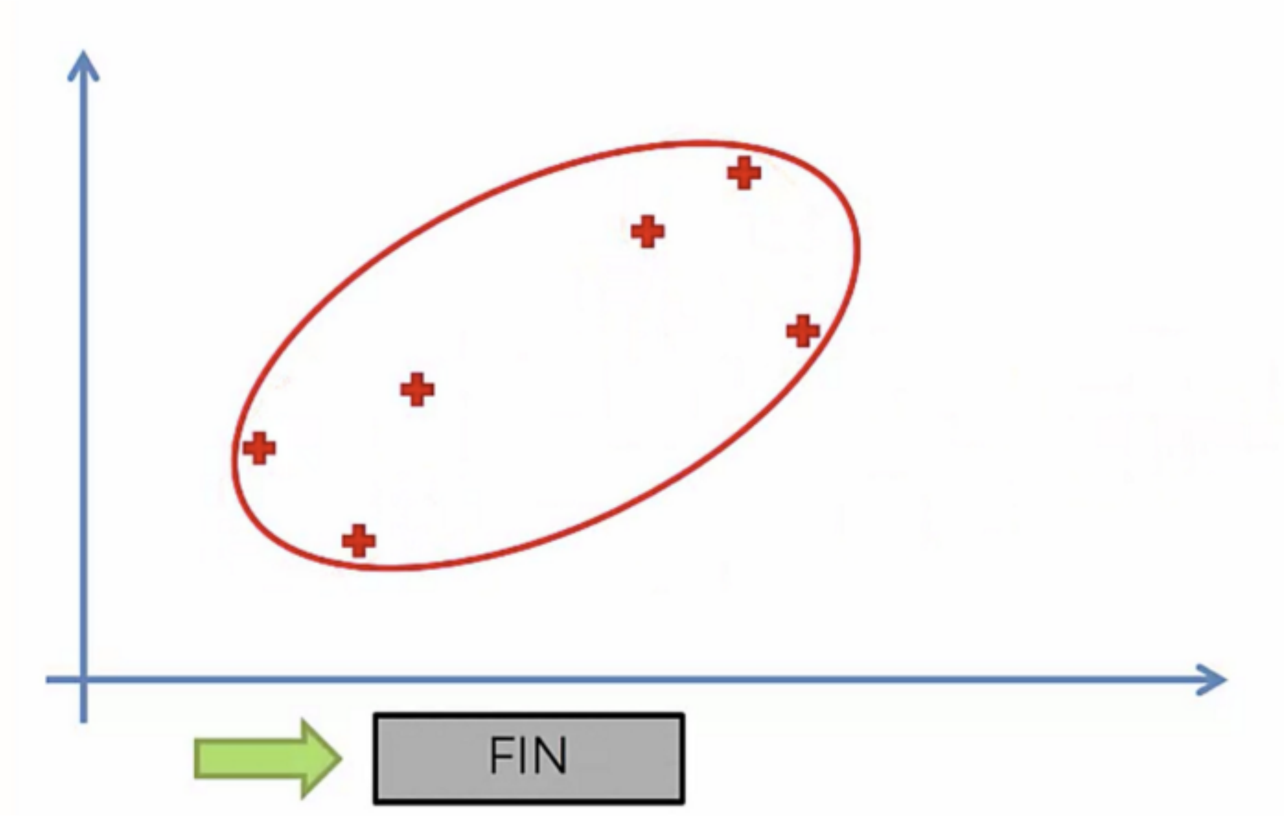
1. Make each data point a cluster



2. Take the two closest clusters and make them one cluster



3. Repeat step 2 until there is only one cluster

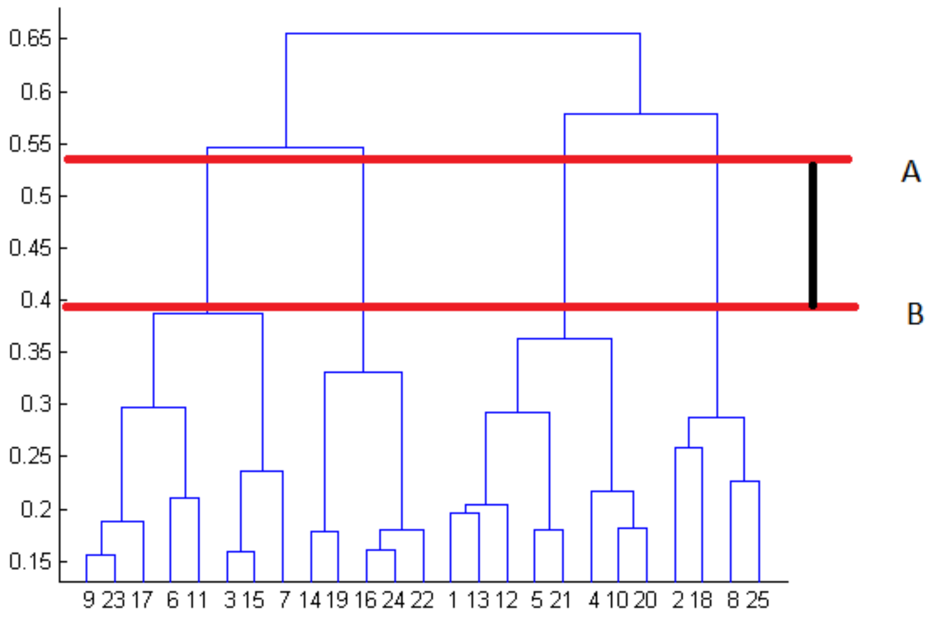


Dendrograms

We can use a dendrogram to visualize the history of groupings and figure out the optimal number of clusters.

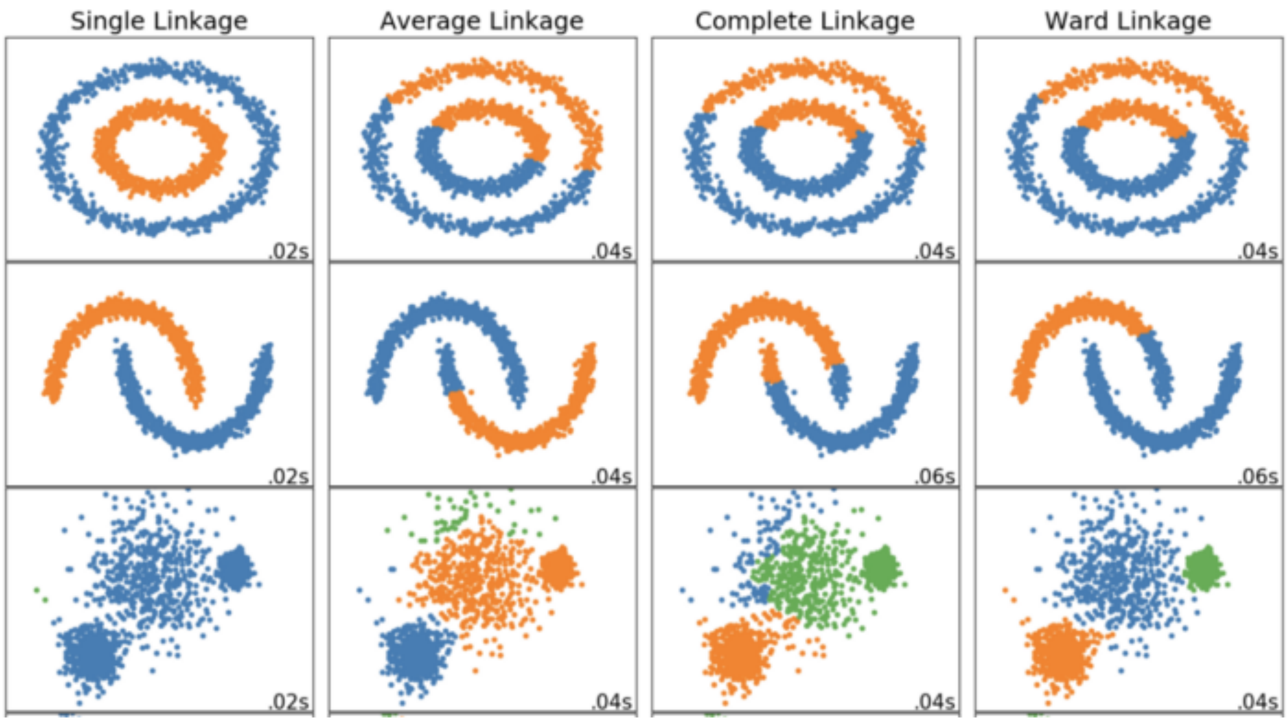
1. Determine the largest vertical distance that doesn't intersect any of the other clusters
2. Draw a horizontal line at both extremities
3. The optimal number of clusters is equal to the number of vertical lines going through the horizontal line

For eg., in the below case, best choice for no. of clusters will be ***4***.

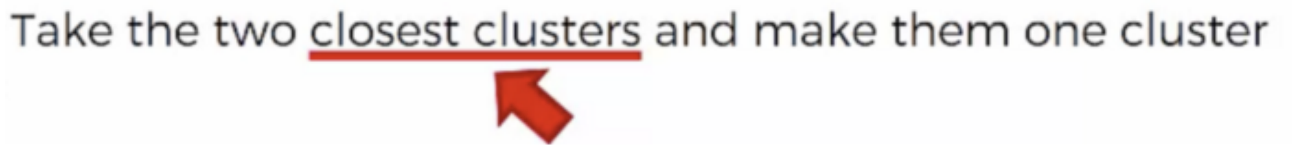


Linkage Criteria

Similar to gradient descent, you can tweak certain parameters to get drastically different results.

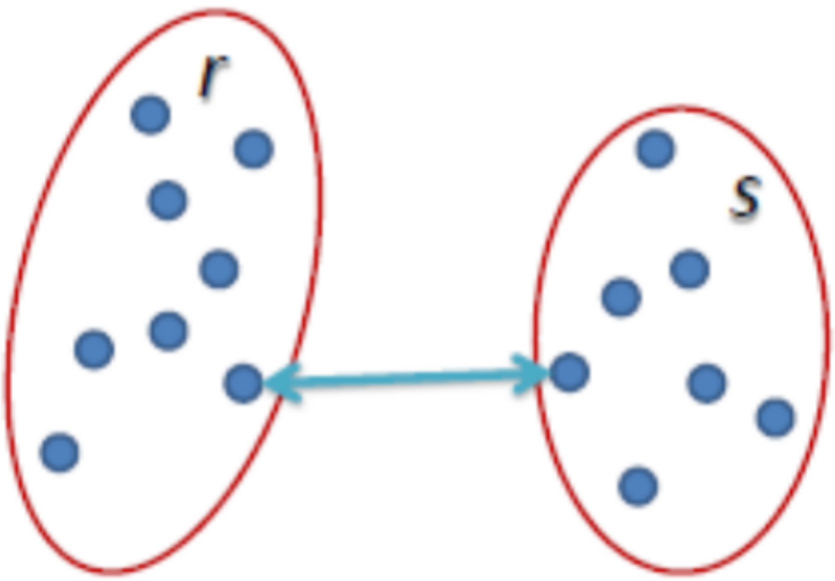


The linkage criteria refers to how the distance between clusters is calculated.


Take the two closest clusters and make them one cluster

Single Linkage

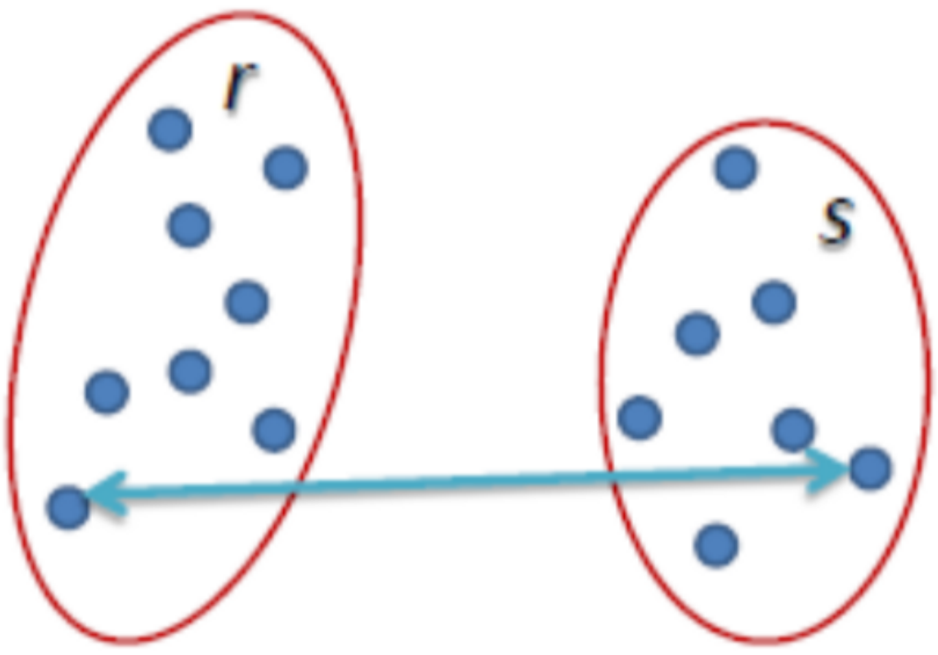
The distance between two clusters is the shortest distance between two points in each cluster



$$L(r,s) = \min(D(x_{ri}, x_{sj}))$$

Complete Linkage

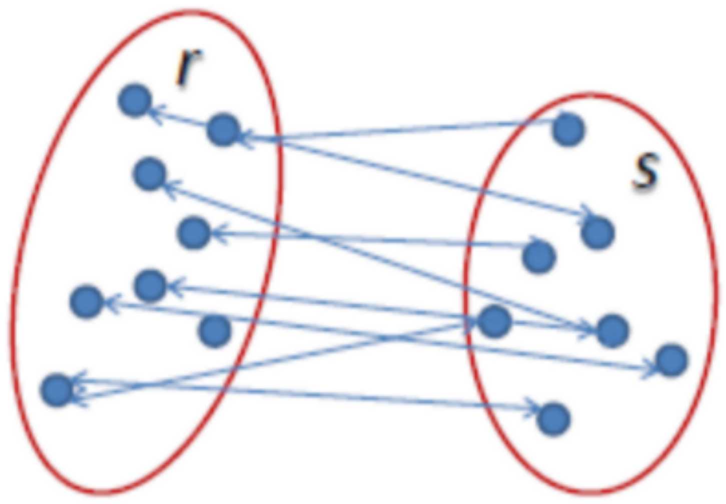
The distance between two clusters is the longest distance between two points in each cluster



$$L(r,s) = \max(D(x_{ri}, x_{sj}))$$

Average Linkage

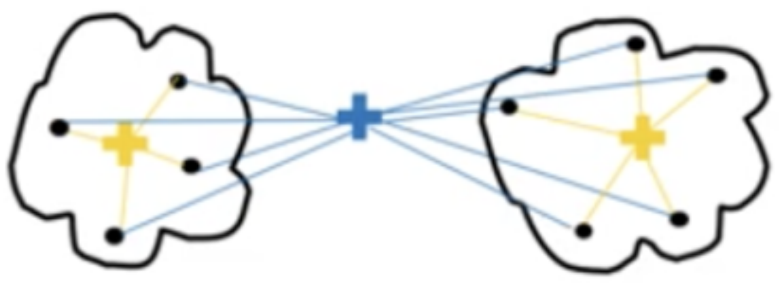
The distance between clusters is the average distance between each point in one cluster to every point in other cluster



$$L(r,s)=\frac{1}{n_r n_s}\sum_{i=1}^{n_r}\sum_{j=1}^{n_s}D(x_{ri},x_{sj})$$

Ward Linkage

The distance between clusters is the sum of squared differences within all clusters

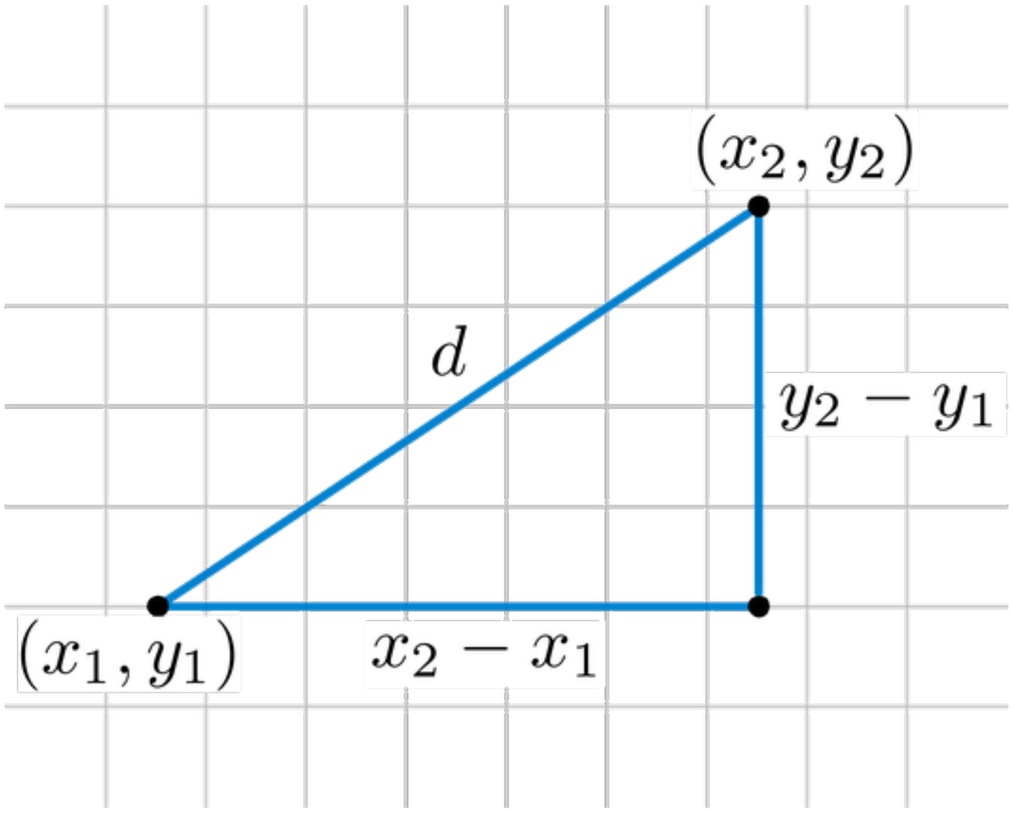


Distance Metric

The method you use to calculate the distance between data points will affect the end result.

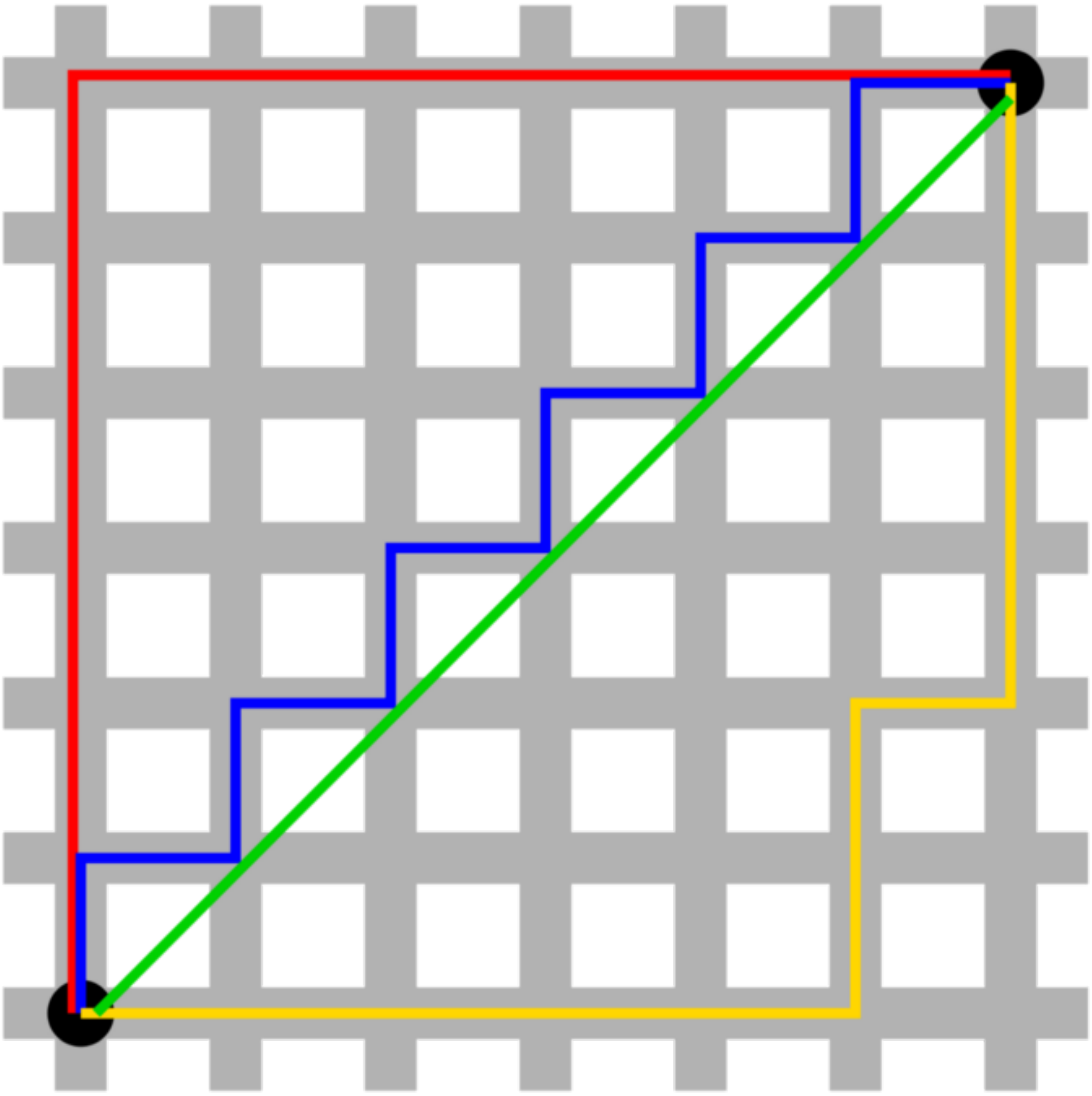
Euclidean Distance

The shortest distance between two points. For example, if x=(a,b) and y=(c,d), the Euclidean distance between x and y is $\sqrt{(a-c)^2+(b-d)^2}$



Manhattan Distance

Imagine you were in the downtown center of a big city and you wanted to get from point A to point B. You wouldn't be able to cut across buildings, rather you'd have to make your way by walking along the various streets. For example, if x=(a,b) and y=(c,d), the Manhattan distance between x and y is |a−c|+|b−d|

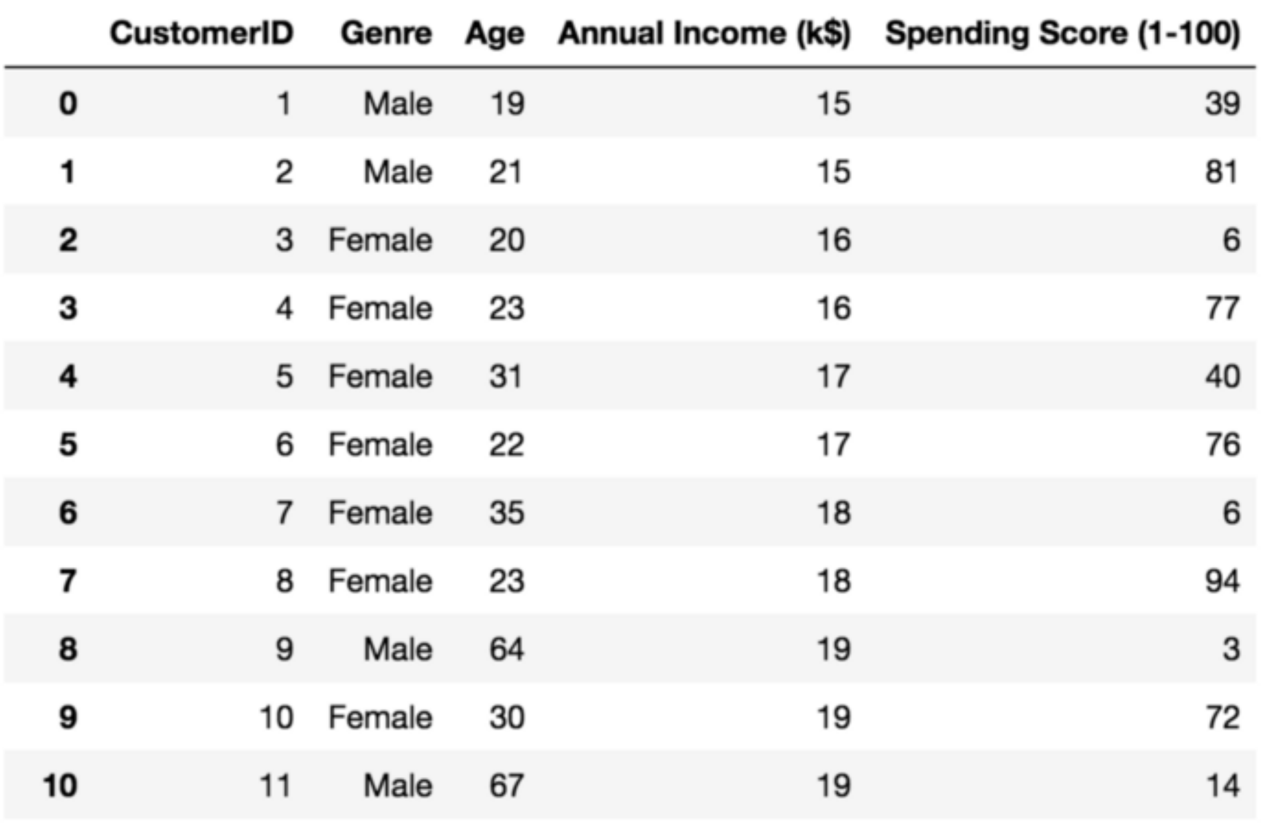


Example in python

Let’s take a look at a concrete example of how we could go about labelling data using hierarchical agglomerative clustering.

```
import pandas as pd
import numpy as np
from matplotlib import pyplot as plt
from sklearn.cluster import AgglomerativeClustering
import scipy.cluster.hierarchy as sch
```

In this tutorial, we use the csv file containing a list of customers with their gender, age, annual income and spending score.



	CustomerID	Genre	Age	Annual Income (k$)	Spending Score (1-100)
0	1	Male	19	15	39
1	2	Male	21	15	81
2	3	Female	20	16	6
3	4	Female	23	16	77
4	5	Female	31	17	40
5	6	Female	22	17	76
6	7	Female	35	18	6
7	8	Female	23	18	94
8	9	Male	64	19	3
9	10	Female	30	19	72
10	11	Male	67	19	14

If you want to follow along, you can get the dataset from the *superdatascience* website.

Machine Learning A-Z™: Download Practice Datasets - SuperDataScience - Big Data | Analytics Careers...

Greetings Welcome to the data repository for the Machine Learning course by Kirill Eremenko and Hadelin de Ponteves...

www.superdatascience.com

Sign up for The Variable

By Towards Data Science

Every Thursday, the Variable delivers the very best of Towards Data Science: from hands-on tutorials and cutting-edge research to original features you don't want to miss. Take a look.

Get this newsletter

You'll need to sign in or create an account to receive this newsletter.

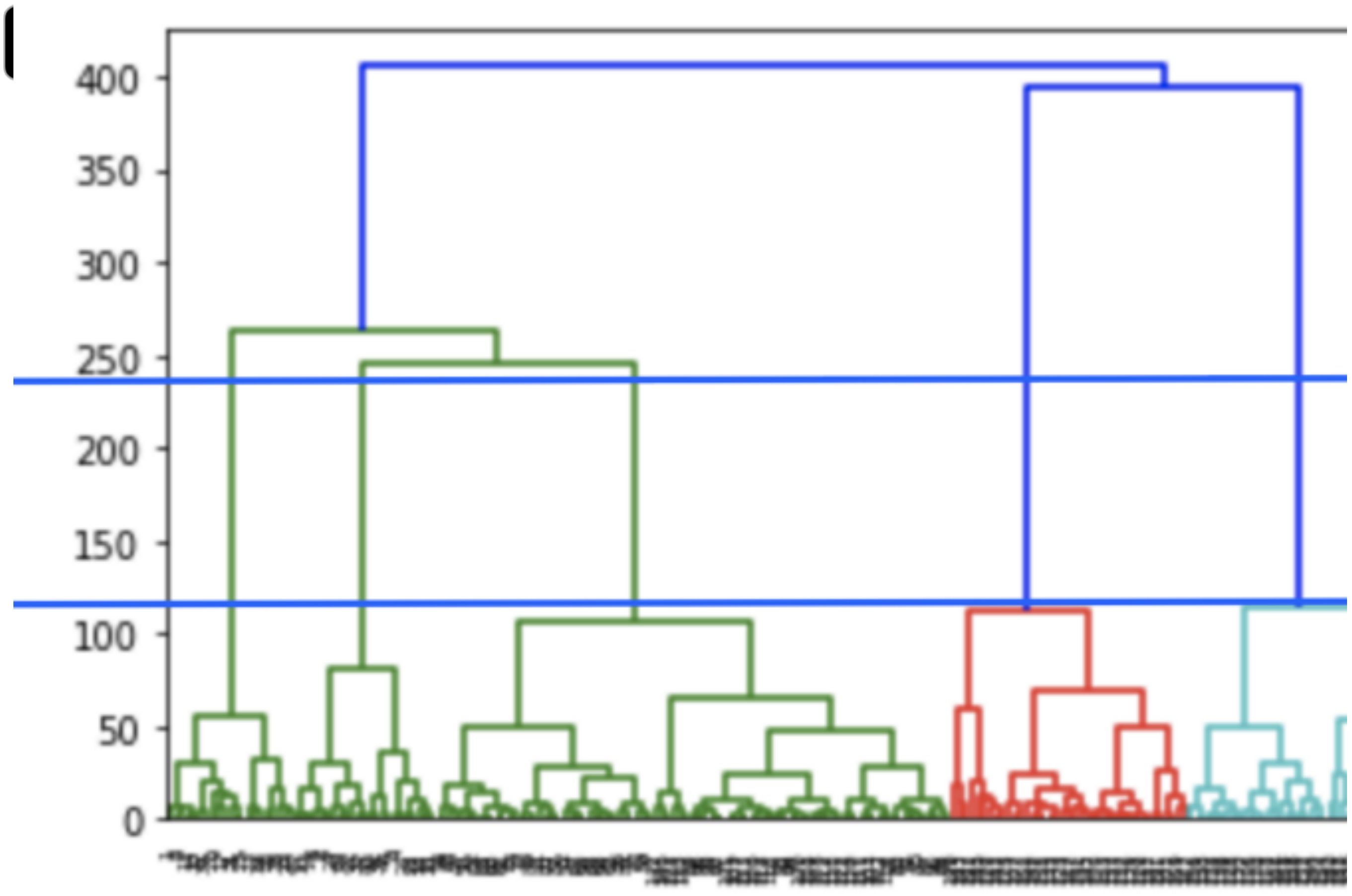
Looking at the dendrogram, the highest vertical distance that doesn't intersect with any clusters is the middle green one. Given that 5 vertical lines cross the threshold, the optimal number of clusters is 5.

Machine Learning Towards Data Science Data Science Programming Artificial Intelligence

```
dendrogram = sch.dendrogram(sch.linkage(X, method='ward'))
```

About Help Legal

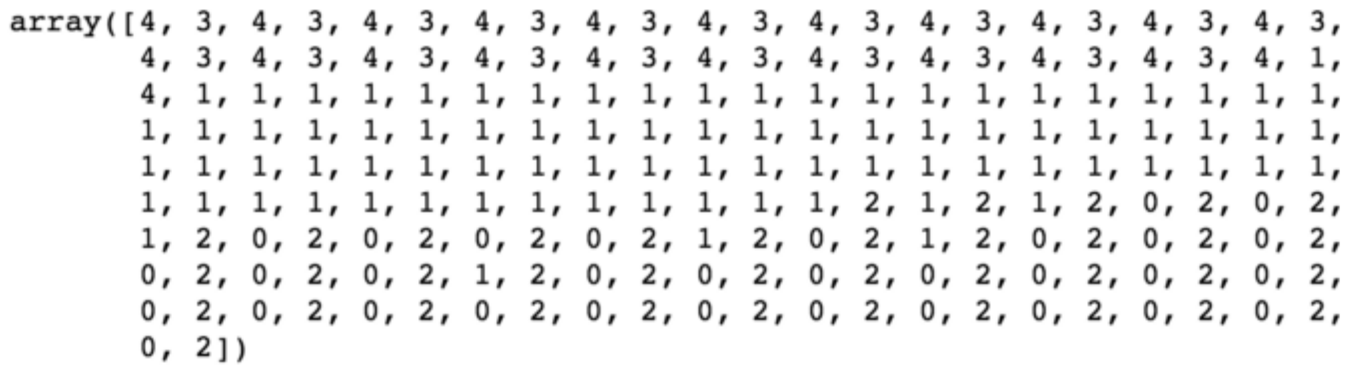
Get the Medium app



We create an instance of `AgglomerativeClustering` using the euclidean distance as the measu
of distance between points and ward linkage to calculate the proximity of clusters.

```
model = AgglomerativeClustering(n_clusters=5, affinity='euclidean',
linkage='ward')
model.fit(X)
labels = model.labels_
```

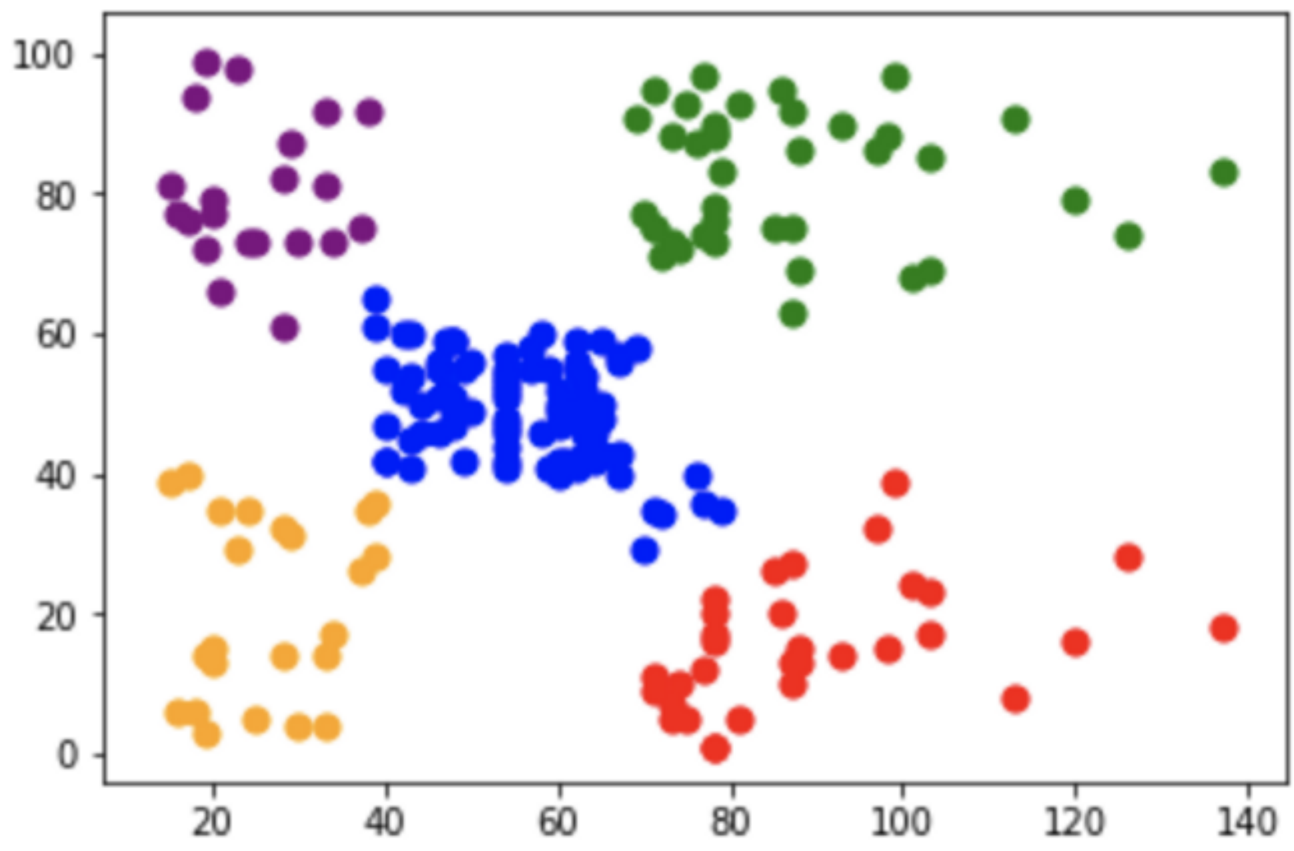
The `labels_` property returns an array of integers where the values correspond to the distinc
categories.



```
array([4, 3, 4, 3, 4, 3, 4, 3, 4, 3, 4, 3, 4, 3, 4, 3, 4, 3, 4, 3, 4, 3,
       4, 3, 4, 3, 4, 3, 4, 3, 4, 3, 4, 3, 4, 3, 4, 3, 4, 3, 4, 3, 4, 1,
       4, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
       1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
       1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
       1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 2, 1, 2, 1, 2, 0, 2, 0, 2,
       1, 2, 0, 2, 0, 2, 0, 2, 0, 2, 1, 2, 0, 2, 1, 2, 0, 2, 0, 2, 0, 2,
       0, 2, 0, 2, 0, 2, 1, 2, 0, 2, 0, 2, 0, 2, 0, 2, 0, 2, 0, 2, 0, 2,
       0, 2, 0, 2, 0, 2, 0, 2, 0, 2, 0, 2, 0, 2, 0, 2, 0, 2, 0, 2, 0, 2,
       0, 2])
```

We can use a shorthand notation to display all the samples belonging to a category as a speci
color.

```
plt.scatter(X[labels==0, 0], X[labels==0, 1], s=50, marker='o',
color='red')
plt.scatter(X[labels==1, 0], X[labels==1, 1], s=50, marker='o',
color='blue')
plt.scatter(X[labels==2, 0], X[labels==2, 1], s=50, marker='o',
color='green')
plt.scatter(X[labels==3, 0], X[labels==3, 1], s=50, marker='o',
color='purple')
plt.scatter(X[labels==4, 0], X[labels==4, 1], s=50, marker='o',
color='orange')
plt.show()
```


100
80
60
40
20
0
20
40
60
80
100
120
140