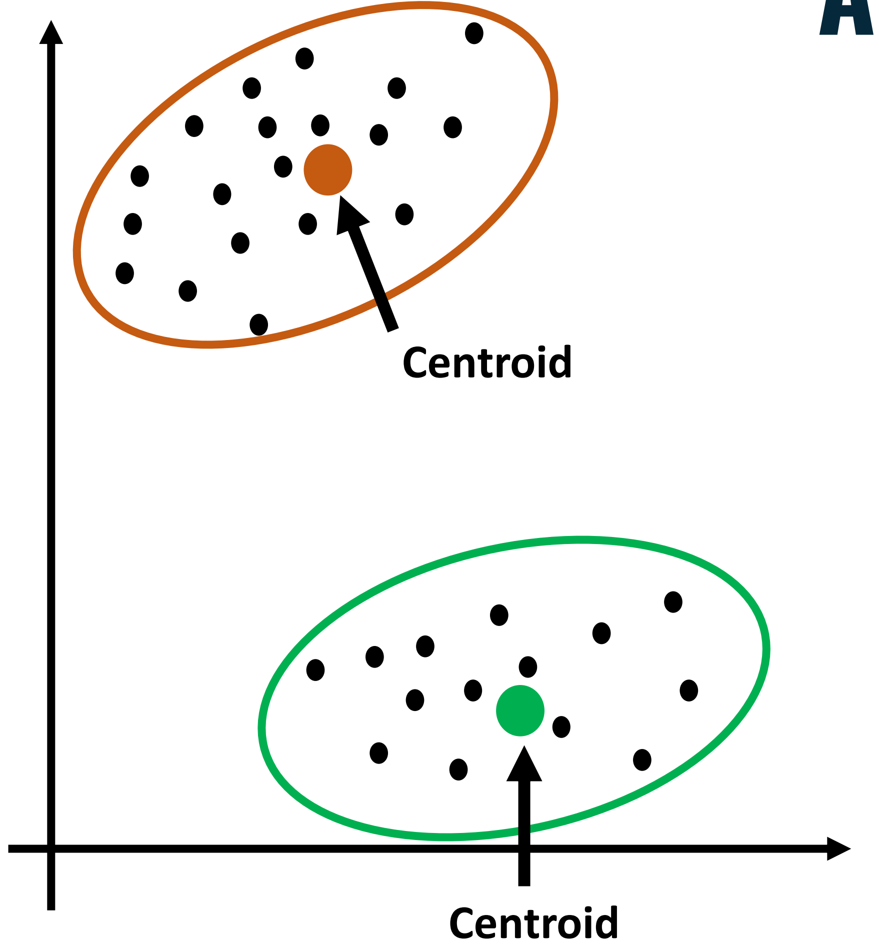
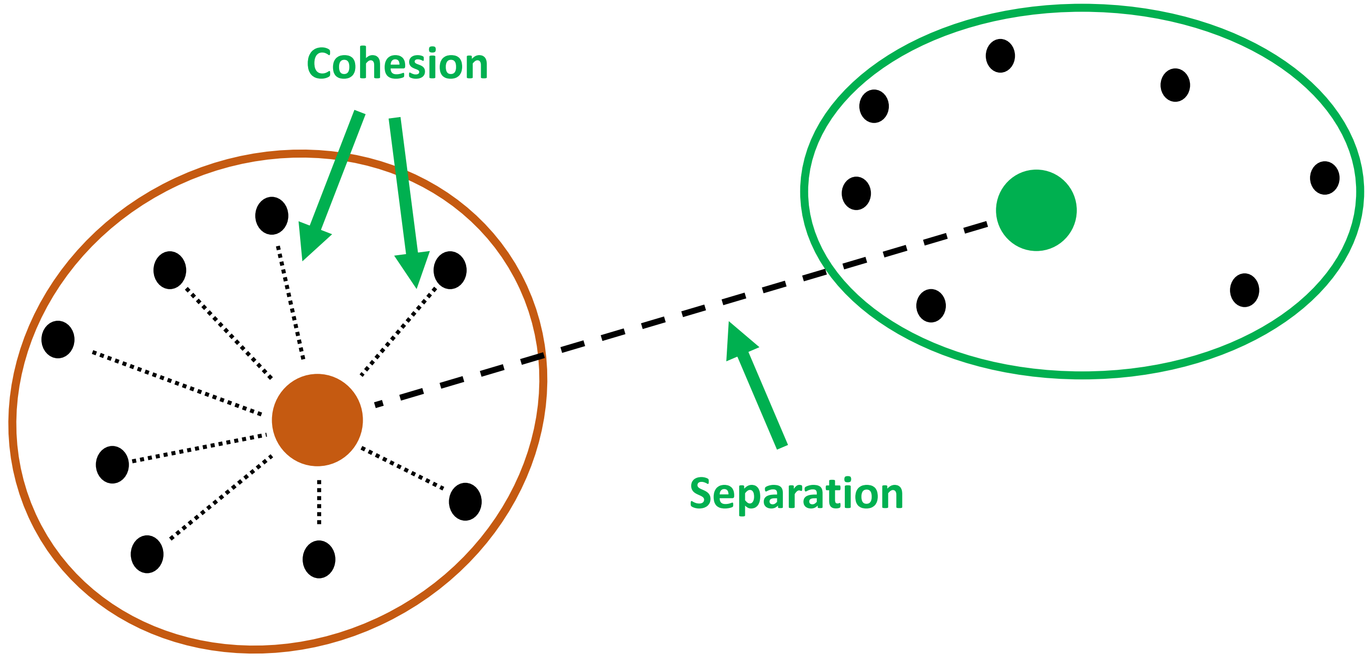
CLUSTER ANALYSIS WITH PYTHON

A CRASH COURSE

Dave ON DATA





About Me

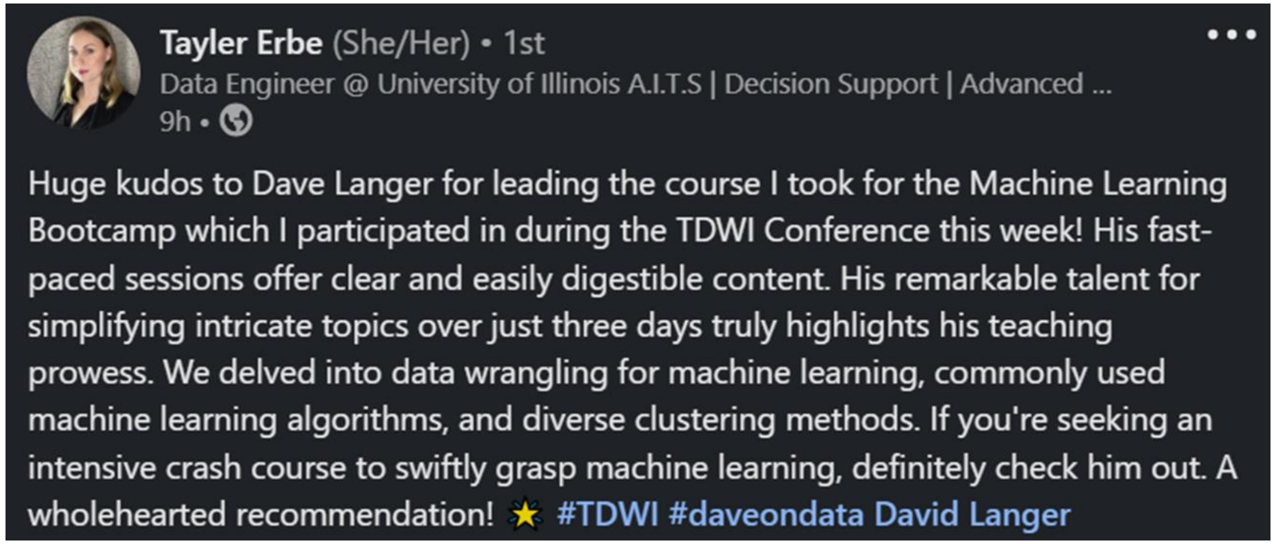
I've been in tech for 26 years and doing hands-on analytics for 12+ years.

I've supported all manner of business functions and advised leaders.

I have successfully trained 1000+ professionals in a live classroom setting.

Trained 1000s more via my online courses and tutorials.

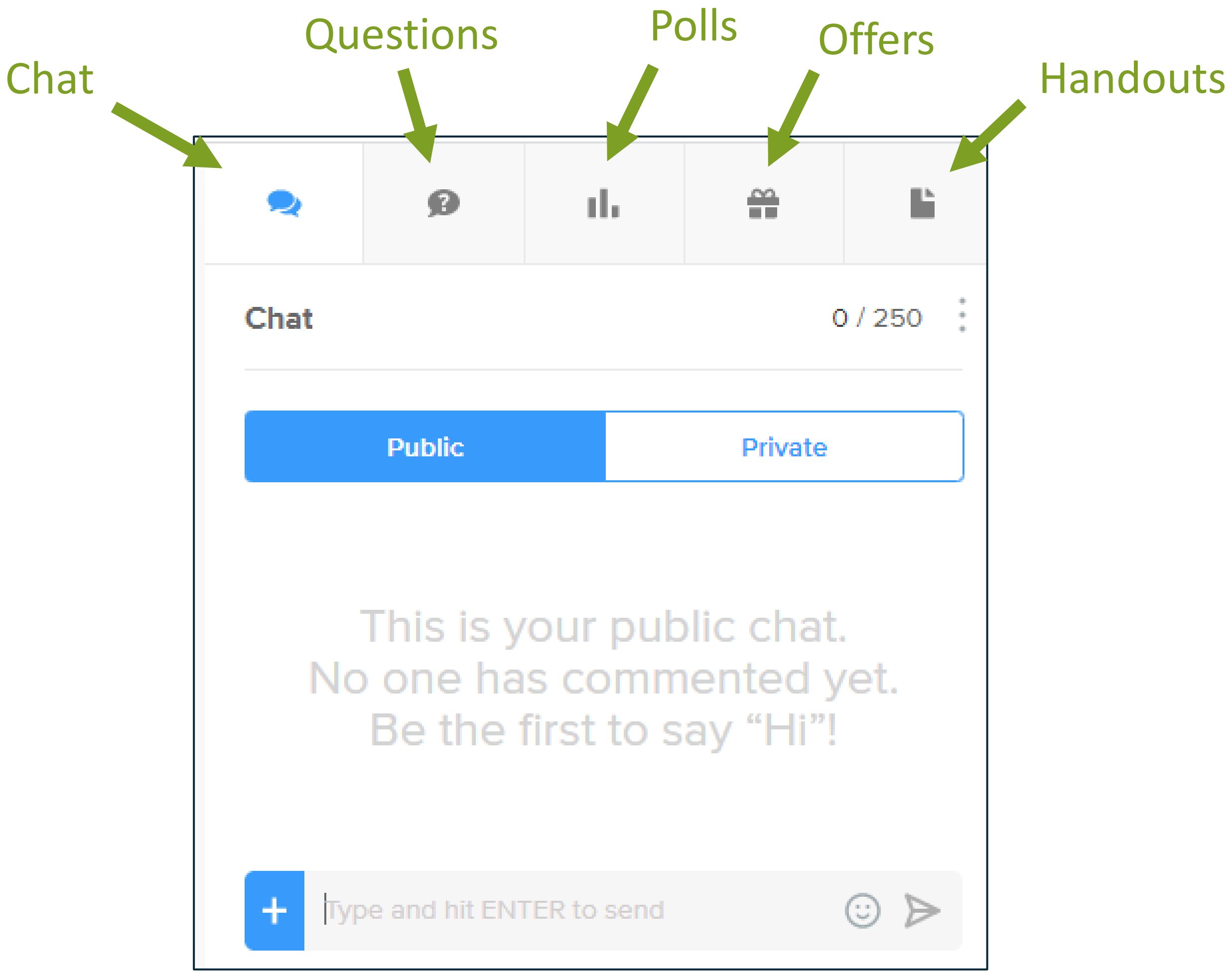
Hands-on analytics consultant and instructor.



Tayler Erbe (She/Her) • 1st
Data Engineer @ University of Illinois A.I.T.S | Decision Support | Advanced ...
9h •

Huge kudos to Dave Langer for leading the course I took for the Machine Learning Bootcamp which I participated in during the TDWI Conference this week! His fast-paced sessions offer clear and easily digestible content. His remarkable talent for simplifying intricate topics over just three days truly highlights his teaching prowess. We delved into data wrangling for machine learning, commonly used machine learning algorithms, and diverse clustering methods. If you're seeking an intensive crash course to swiftly grasp machine learning, definitely check him out. A wholehearted recommendation! ⭐ #TDWI #daveondata David Langer

Housekeeping



This is your public chat.
No one has commented yet.
Be the first to say “Hi”!

Introduction

What is Cluster Analysis?

"**Cluster analysis** groups data objects based only on information found in the data that describes the objects and their relationships."

"The goal is that the objects within a group be similar (or related) to one another and different from (or unrelated to) the object in other groups."

"The greater the similarity (or homogeneity) within a group and the greater the difference between groups, the better or more distinct the clustering." [Pan 2005]

Because cluster analysis has no external information about groups (i.e., **labels**), it belongs to a form of machine learning known as **unsupervised learning**.

Because so much data is unlabeled, cluster analysis is a widely used tool in analytics to discover structure in data and produce new insights.

BTW – The words **groups** and **clusters** are interchangeable in this webinar.

Why Cluster Analysis?

Much effort is put into examining objects in the world and assigning the objects to **classes**.

For example, biologists have created a taxonomy for **classifying** animals – kingdom, phylum, class, etc.

Cluster analysis automates the process of discovering classes directly from data.

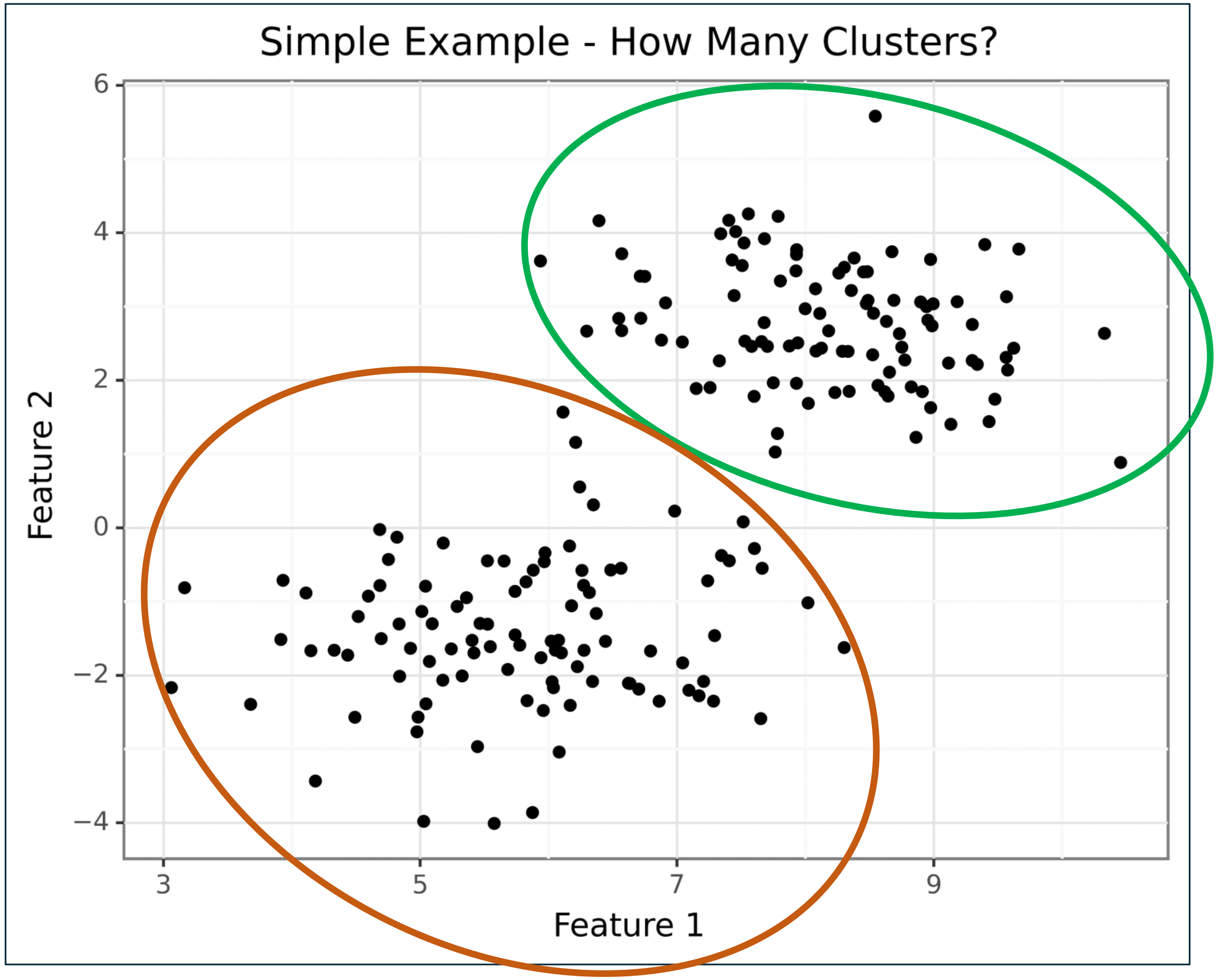
Not surprisingly, this is extremely useful across many scenarios:

- Marketing – Segmenting customers into classes to better understand similarities in backgrounds/behaviors
- IT Operations – Anomaly detection in network operations and security
- Text Analytics – Assigning documents to classes based on the similarity of content
- Supply Chain – Grouping customer deliveries to optimize warehouse locations

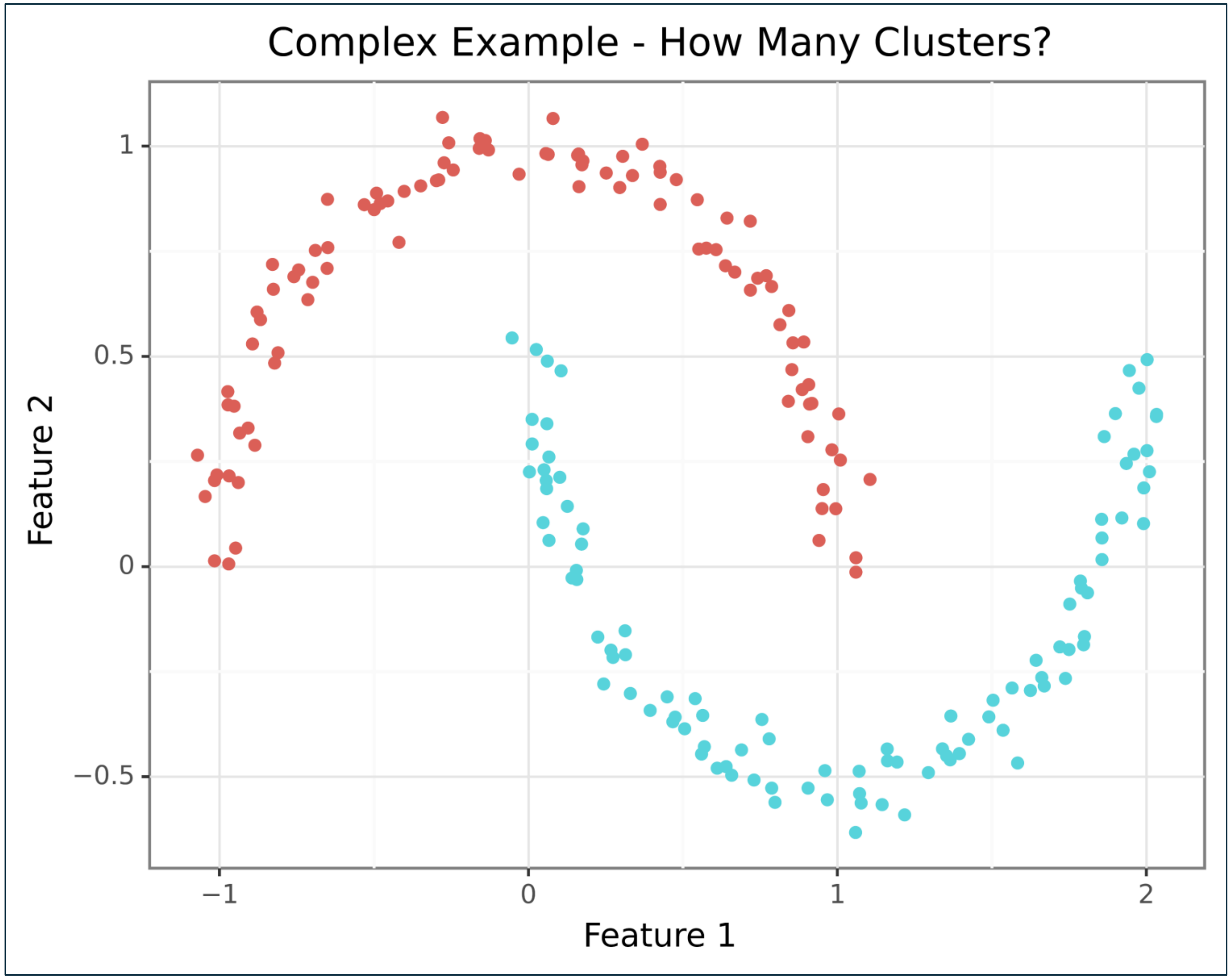
And the list goes on!

BTW – You can think of **group**, **cluster**, and **class** as being interchangeable in this webinar.

The Challenge

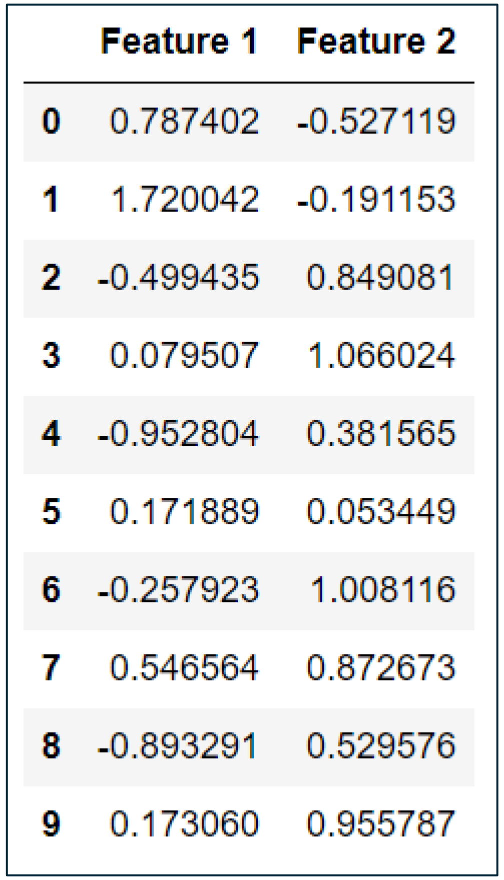


The Challenge



The Challenge

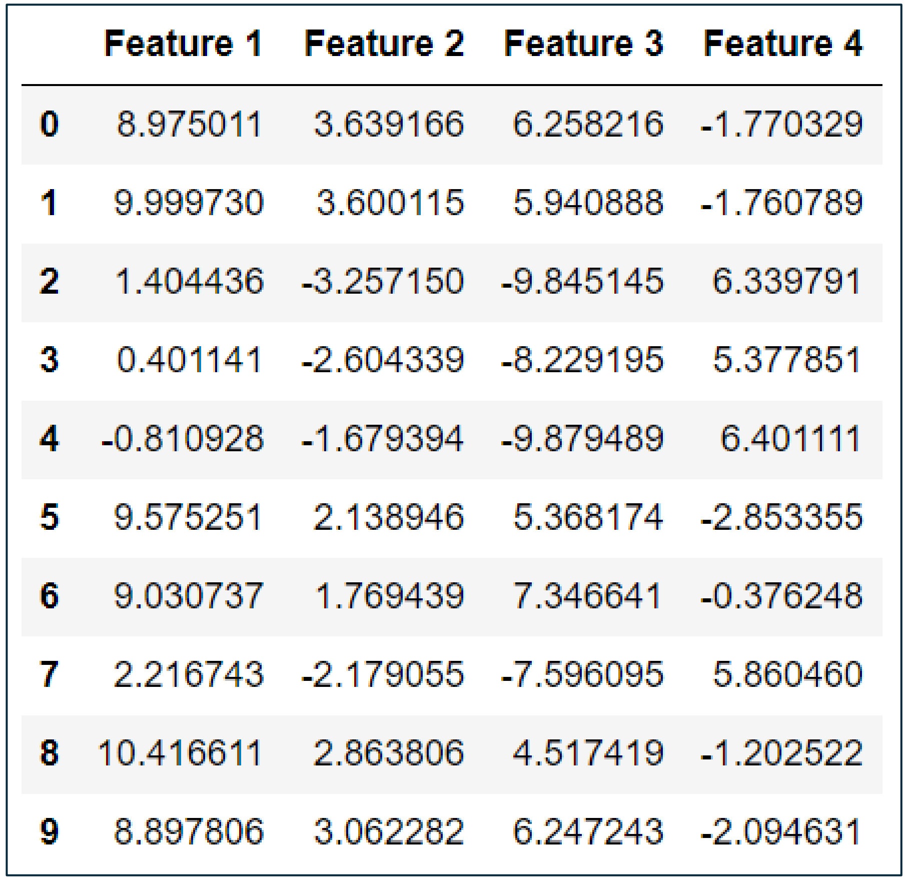
The prior examples were built from simple datasets that look like this:



	Feature 1	Feature 2
0	0.787402	-0.527119
1	1.720042	-0.191153
2	-0.499435	0.849081
3	0.079507	1.066024
4	-0.952804	0.381565
5	0.171889	0.053449
6	-0.257923	1.008116
7	0.546564	0.872673
8	-0.893291	0.529576
9	0.173060	0.955787

Cluster analyses are conducted using 100s, even 1000s of columns!

Clustering is typically done with datasets with many more features:



	Feature 1	Feature 2	Feature 3	Feature 4
0	8.975011	3.639166	6.258216	-1.770329
1	9.999730	3.600115	5.940888	-1.760789
2	1.404436	-3.257150	-9.845145	6.339791
3	0.401141	-2.604339	-8.229195	5.377851
4	-0.810928	-1.679394	-9.879489	6.401111
5	9.575251	2.138946	5.368174	-2.853355
6	9.030737	1.769439	7.346641	-0.376248
7	2.216743	-2.179055	-7.596095	5.860460
8	10.416611	2.863806	4.517419	-1.202522
9	8.897806	3.062282	6.247243	-2.094631

The Data

The Iris Dataset

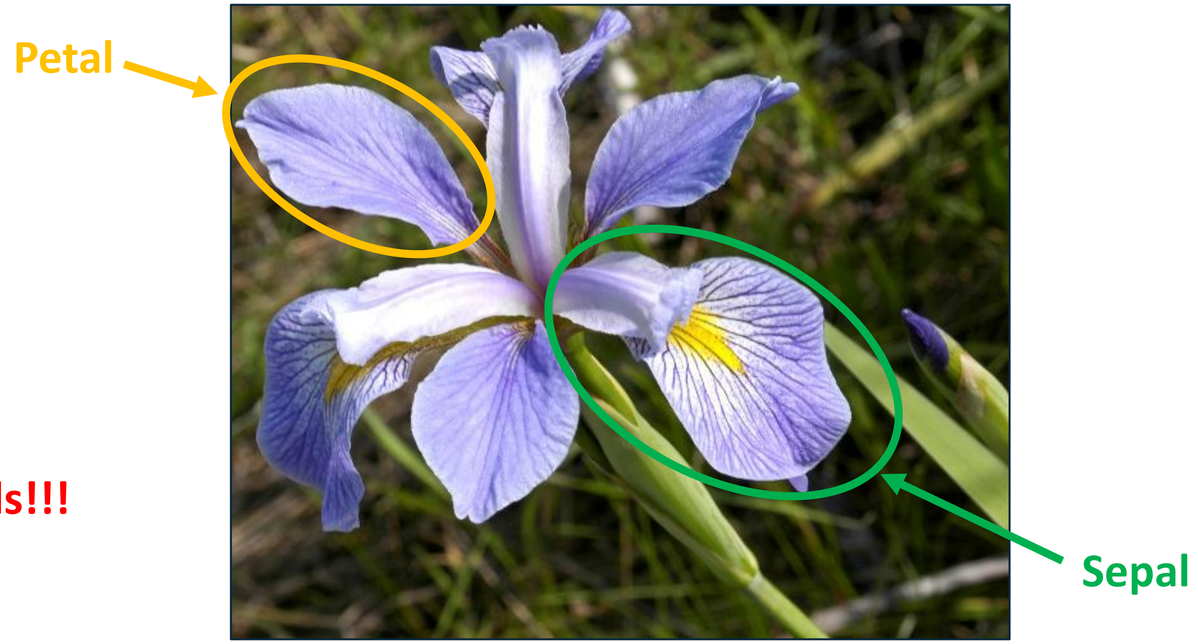
The famous *iris* data set will be used in the slides.

The data set consists of 50 rows for each of three species of iris flower – *setosa*, *versicolor,* and *virginica.*

Each observation consists of the following data:

Variable	Data Type
Sepal Length	Numeric (cm)
Sepal Width	Numeric (cm)
Petal Length	Numeric (cm)
Petal Width	Numeric (cm)
Species	Categorical

Class Labels!!!



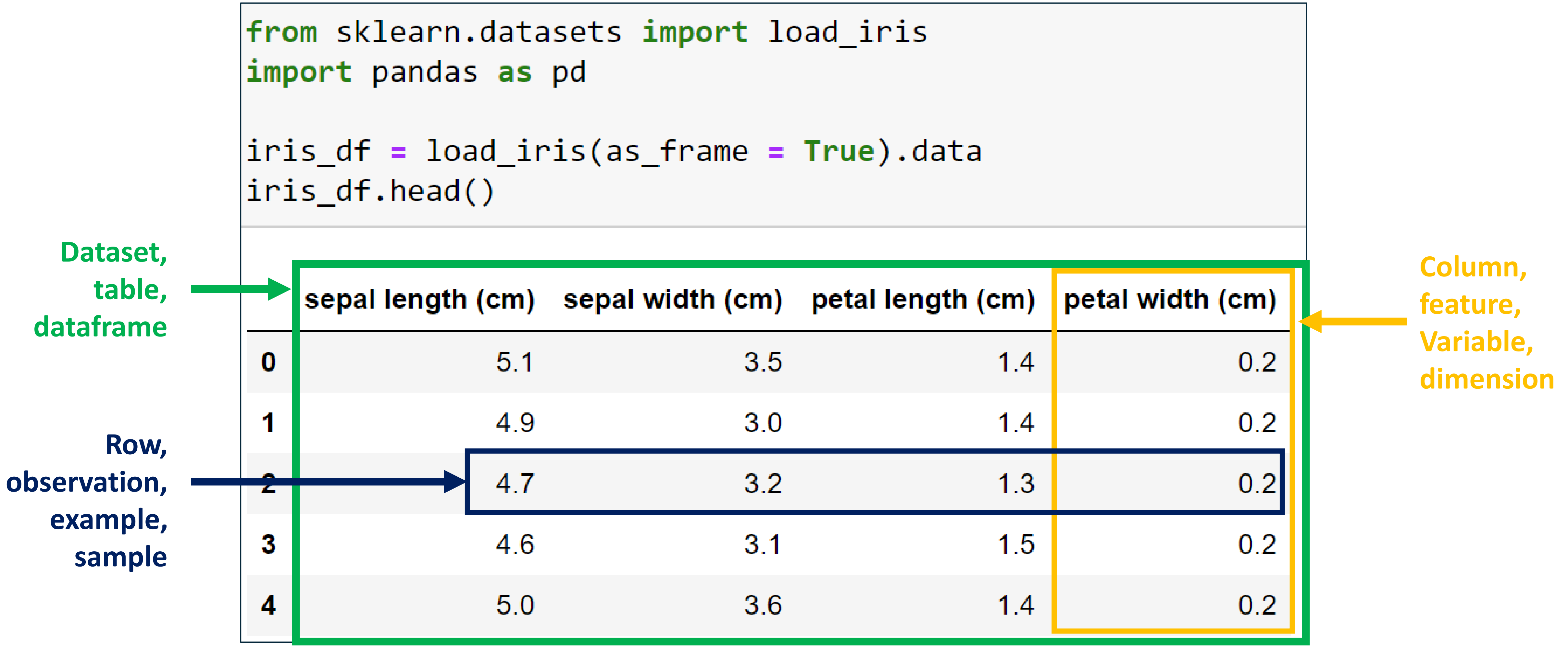
Iris Virginica

Why use a dataset with class labels?

Isn't the point of cluster analysis to find the classes?

The Iris Datast

The *iris* dataset is available via the *scikit-learn* library:



```python
from sklearn.datasets import load_iris
import pandas as pd

iris_df = load_iris(as_frame = True).data
iris_df.head()
```

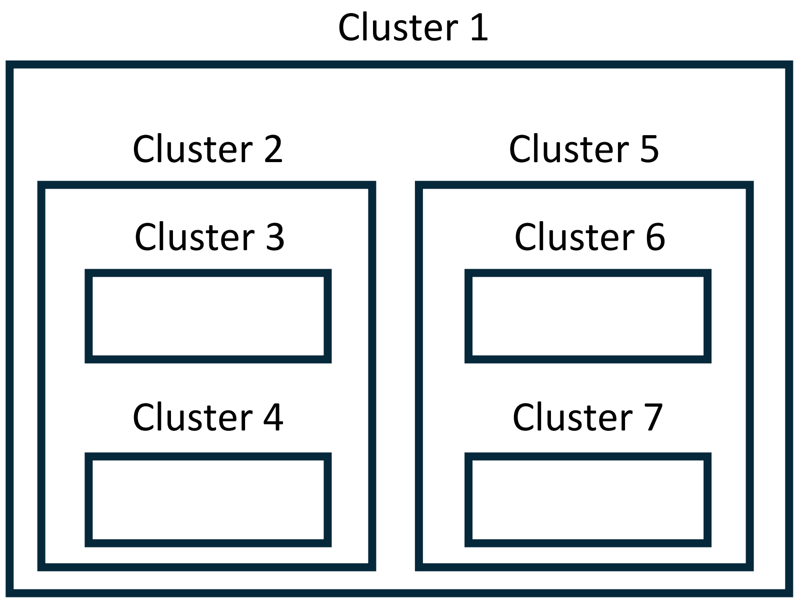
	sepal length (cm)	sepal width (cm)	petal length (cm)	petal width (cm)
0	5.1	3.5	1.4	0.2
1	4.9	3.0	1.4	0.2
2	4.7	3.2	1.3	0.2
3	4.6	3.1	1.5	0.2
4	5.0	3.6	1.4	0.2

Dataset, table, dataframe
Row, observation, example, sample
Column, feature, Variable, dimension

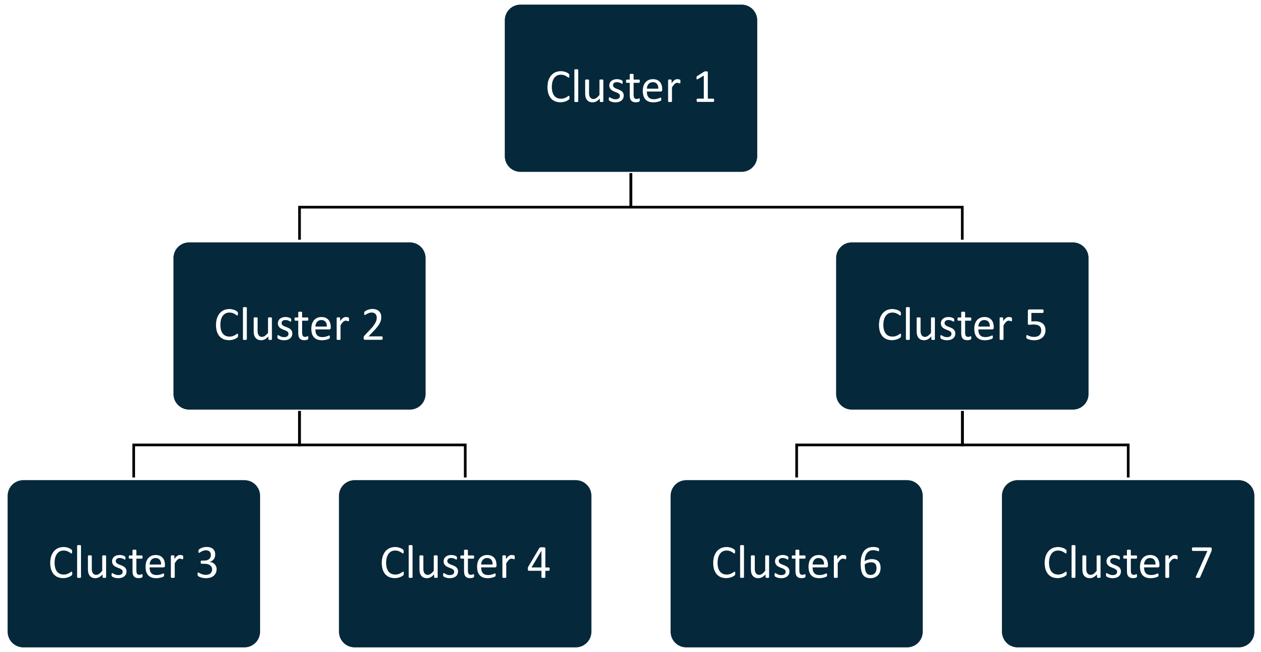
Types of Clusterings and Clusters

Hierarchical

Hierarchical clustering allows for clusters to be nested:



Hierarchical clusters are often visualized using a tree:

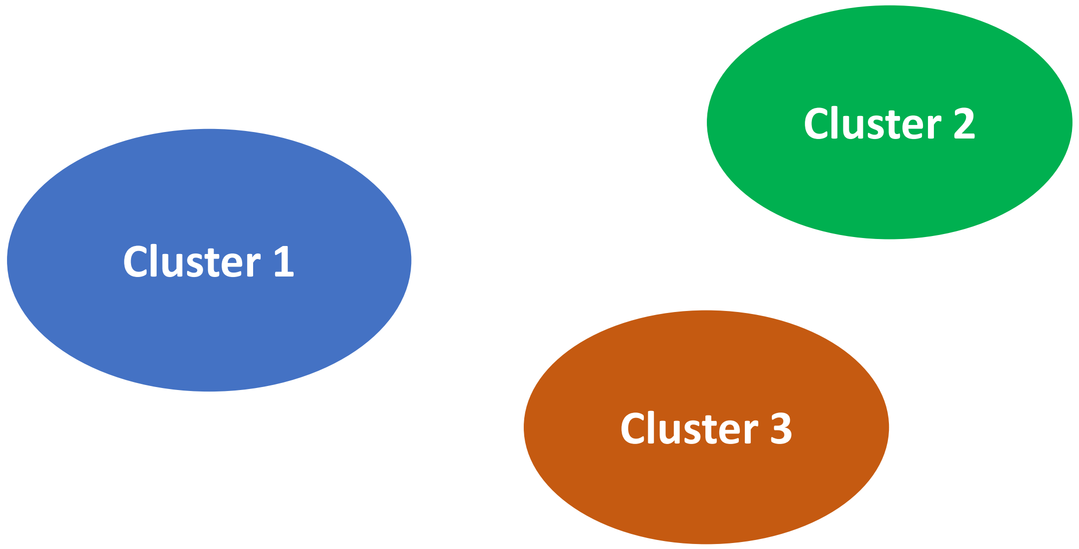


Agglomerative clustering is a bottom-up approach to building the hierarchy.

Divisive clustering is a top-down approach to building the hierarchy.

Partitional

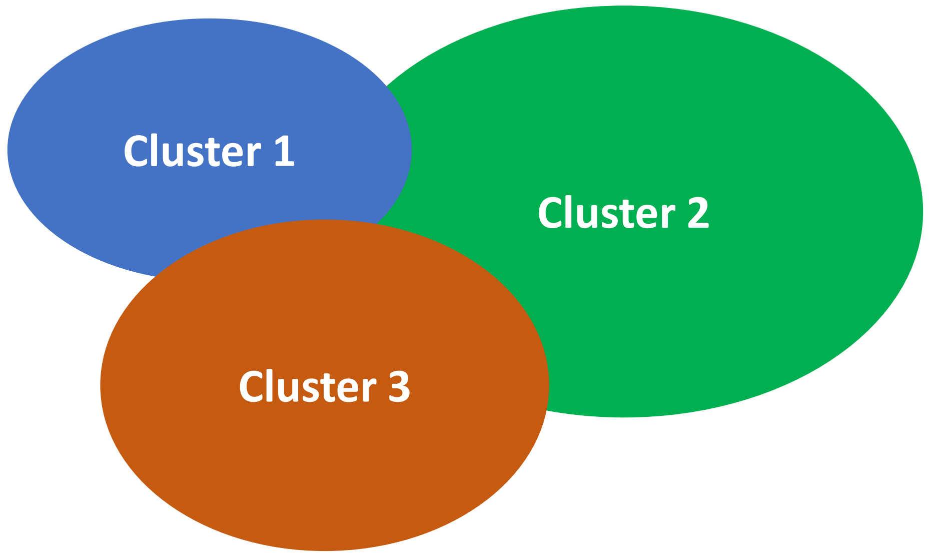
Partitional clustering divides the data into non-overlapping groups (clusters).



The most common applications of cluster analysis use partitional clustering techniques.

Overlapping

Overlapping clustering allows data points to be part of one or more groups (clusters).

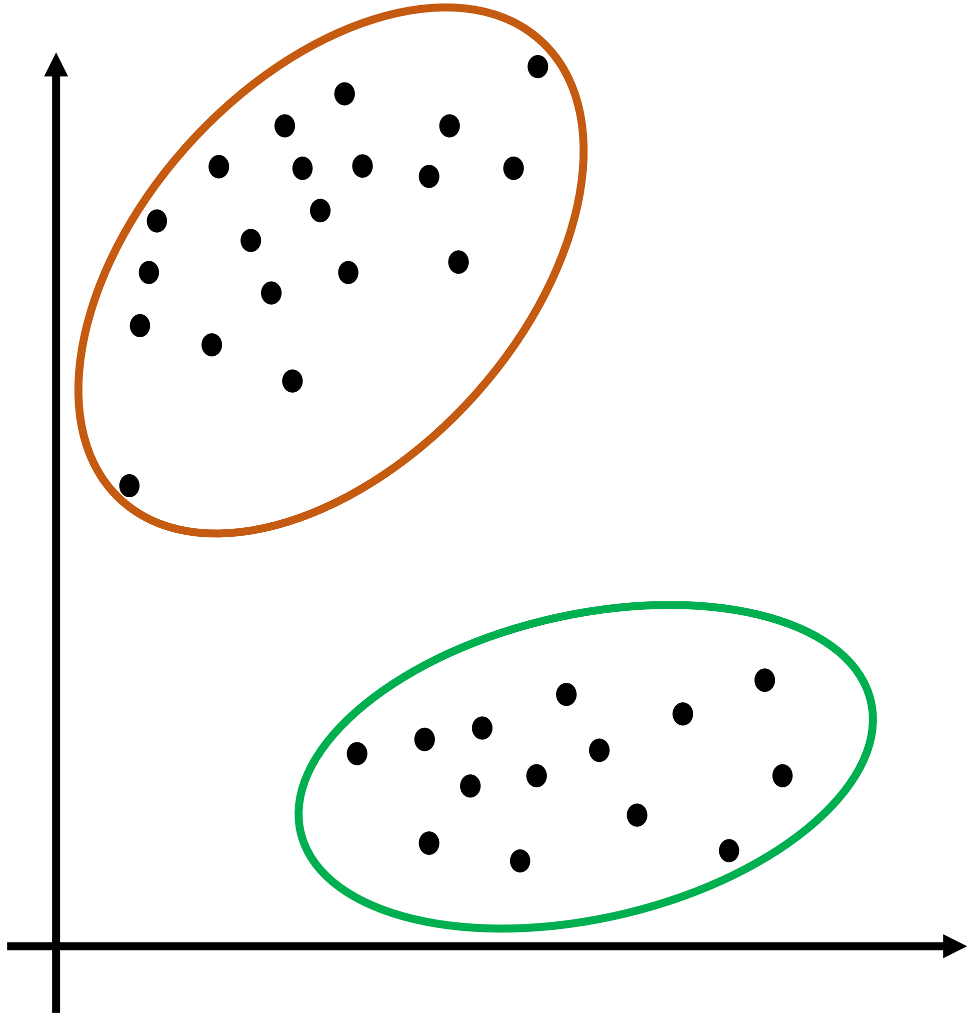


Overlapping clustering allows for many real-world scenarios.

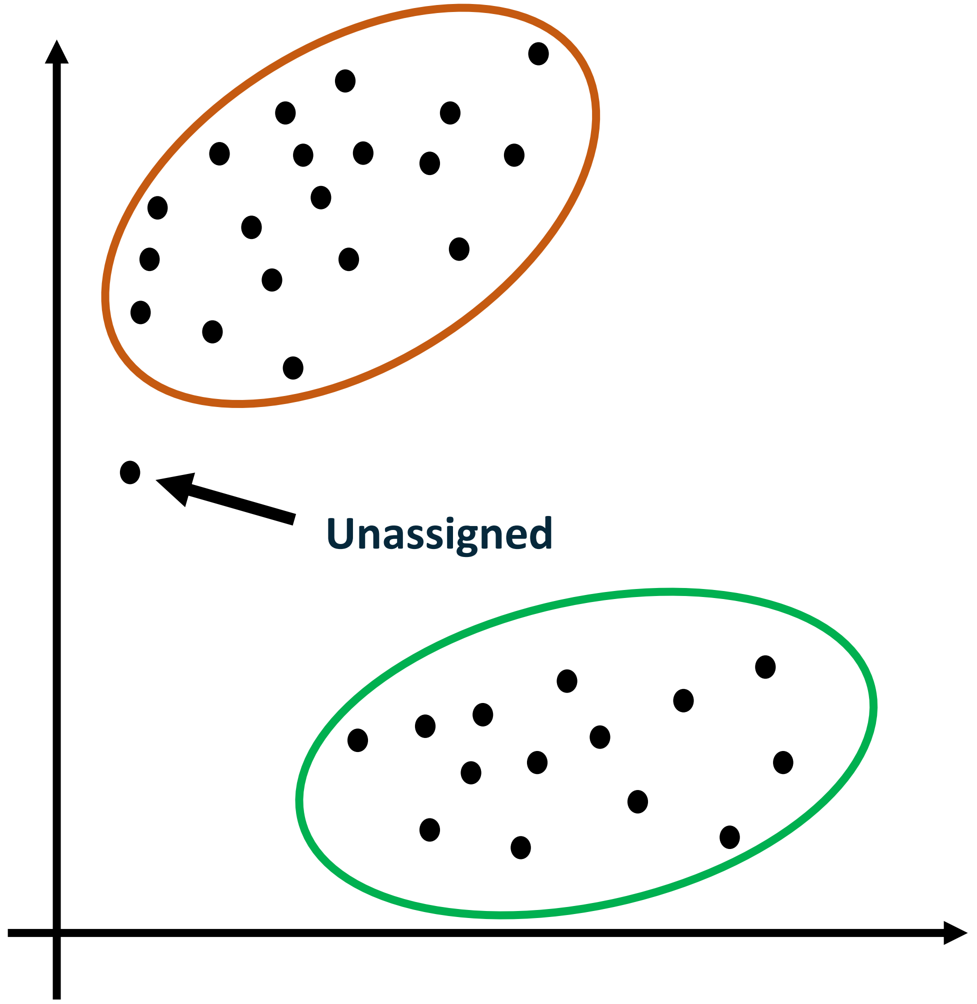
For example, David Langer was once simultaneously a Microsoft employee, shareholder, and customer.

Complete vs. Partial

Complete clustering ensures that every data point is assigned to a cluster:

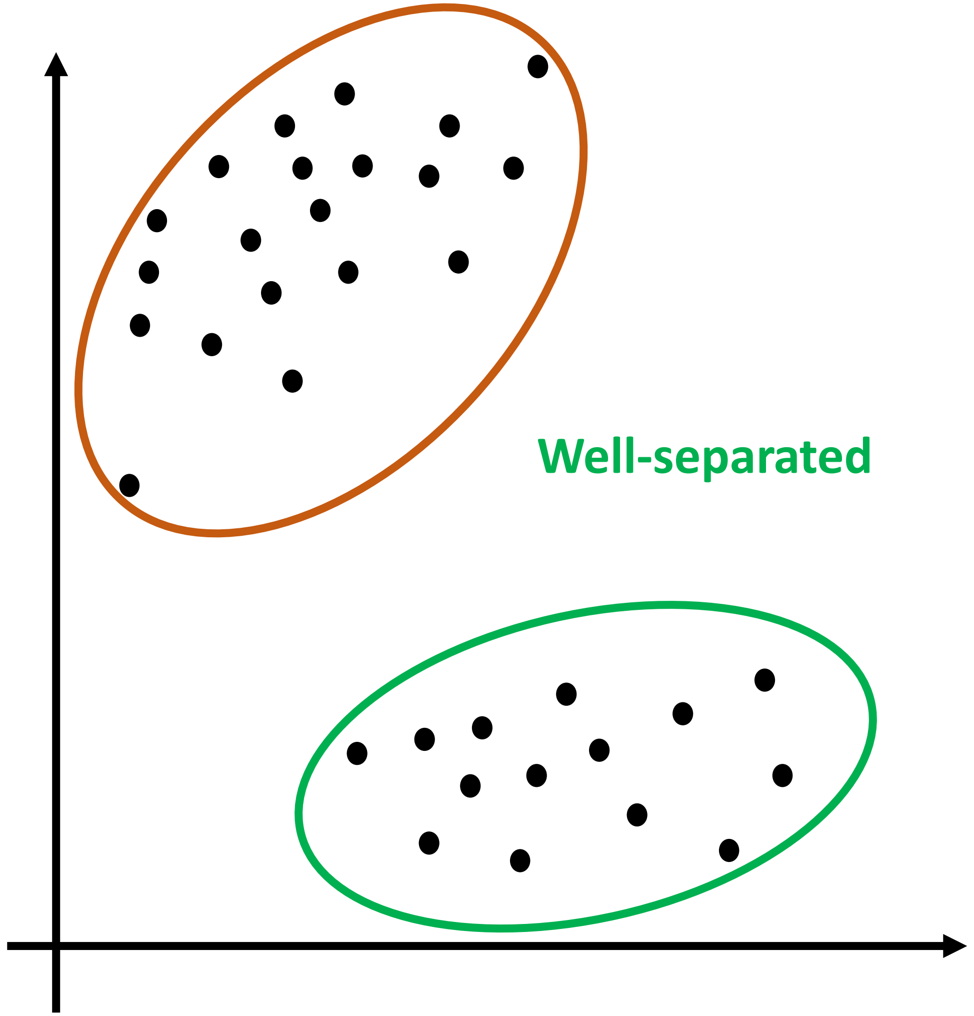


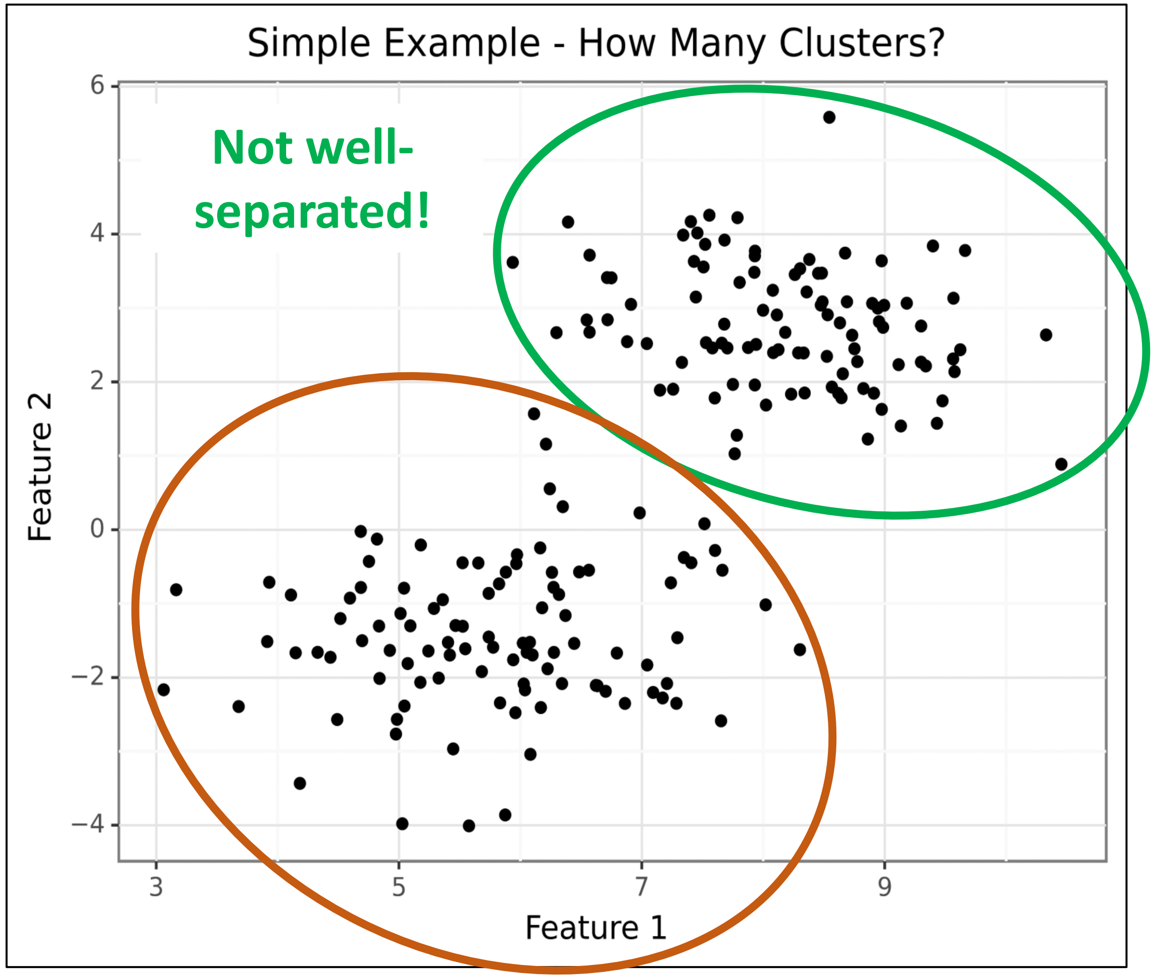
Partial clustering will leave data points unassigned to clusters – typically to improve the clusters:



Well-Separated Clusters

Clusters are **well-separated** when a data point within a cluster is closer to all cluster members than any data points that are members of other clusters.





Prototype Clusters

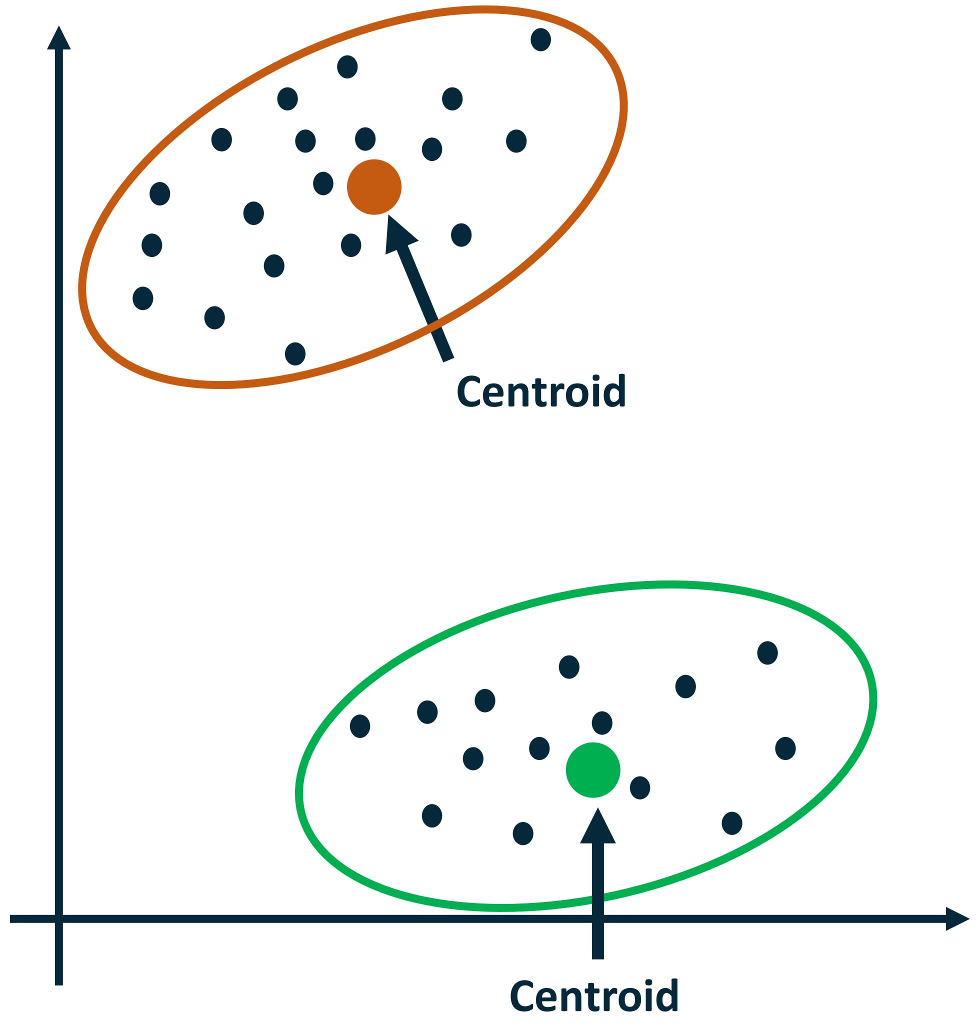
A **prototype cluster** is built around a representative data point that usually does not exist in the original data.

Members of a cluster are closer to the cluster's prototype than any other cluster prototypes.

When clustering numeric data, the prototype is often calculated using the average (mean) of all cluster member data.

In these cases, the prototype is called a **centroid**.

Prototype clusters are commonly used when clusters are spherically shaped.



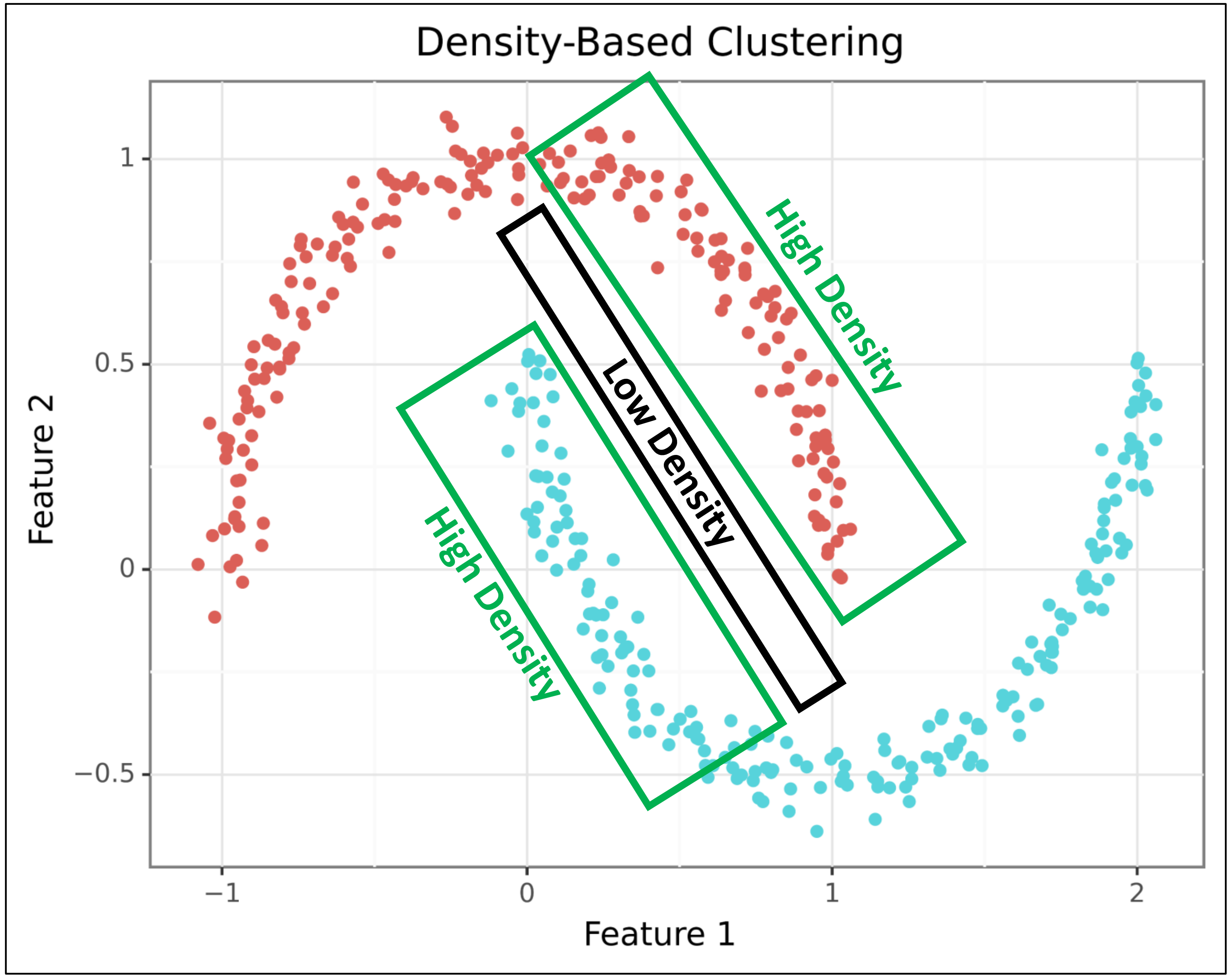
Density-Based Clusters

Density-based clusters are regions dense with data points surrounded by regions with a low density of data points.

Density-based clusters are used when clusters are irregular (e.g., not spherical), intertwined, and/or when outliers are present.

In practice, it is hard to know the shape of clusters due to the use of many features making visualization impossible.

As such, it is common to try multiple clustering techniques and compare results.



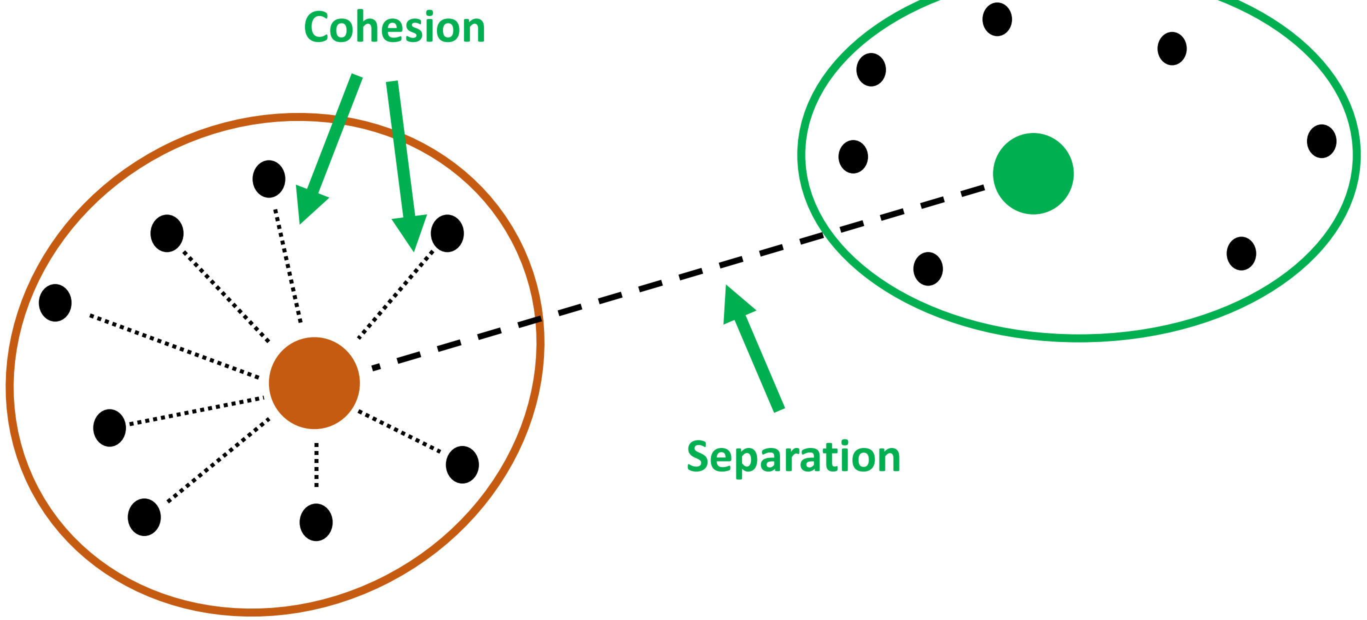
Evaluating a Clustering

The first step in evaluating a clustering is understanding what characterizes one clustering as better than another.

The concepts of cohesion and separation are critical. In terms of k-means:

- **Cohesion** is the sum of proximities of each data point to its assigned centroid
- **Separation** is the proximity between any two centroids

These concepts can be represented graphically.



There exists an important relationship between cohesion and separation.

As cohesion improves (i.e., gets smaller), separation improves (i.e., gets larger).

K-means uses this relationship.

K-Means

Introducing K-Means

The **k-means** algorithm is a prototype-based, complete partitioning clustering technique.

Whew! That was a mouthful.

Let’s break that down:

1. K-means uses centroids based on the average data of cluster members
2. The clusters produced by k-means are spherical
3. Every data point is assigned to a cluster – even outliers
4. Every data point will be assigned to a single cluster

K-means is one of the most popular clustering techniques.

A big part of k-means’ popularity is the algorithm’s simplicity – it is easy to understand how k-means works intuitively.

The K-Means Algorithm

"In mathematics and computer science, an **algorithm** is a finite sequence of rigorous instructions, typically used to solve a class of specific problems or to perform a computation. Algorithms are used as specifications for performing calculations and data processing." - Wikipedia

Here's the **k-means algorithm**:

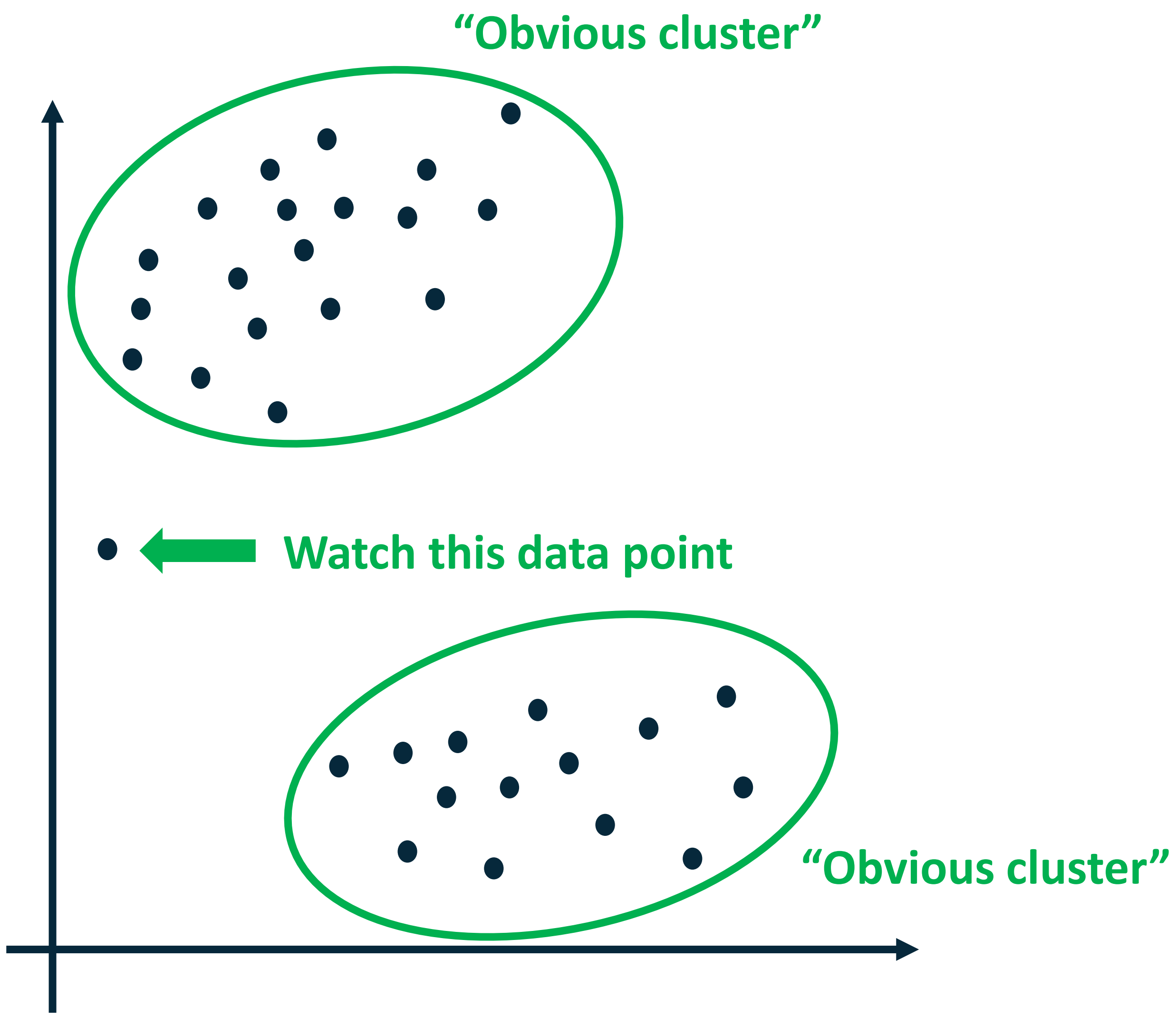
1. Select k points as the initial cluster centroids
2. Form k clusters by assigning each data point to its closest centroid
3. Recompute each centroid as the average of all current cluster members
4. Stop if no centroid changes, otherwise go to 2

Notice that the first step requires you to pick k – the number of clusters to be found.

Turns out that picking the right value for k is a non-trivial problem.

The resources at the end of the webinar will help you in this.

A Contrived Example

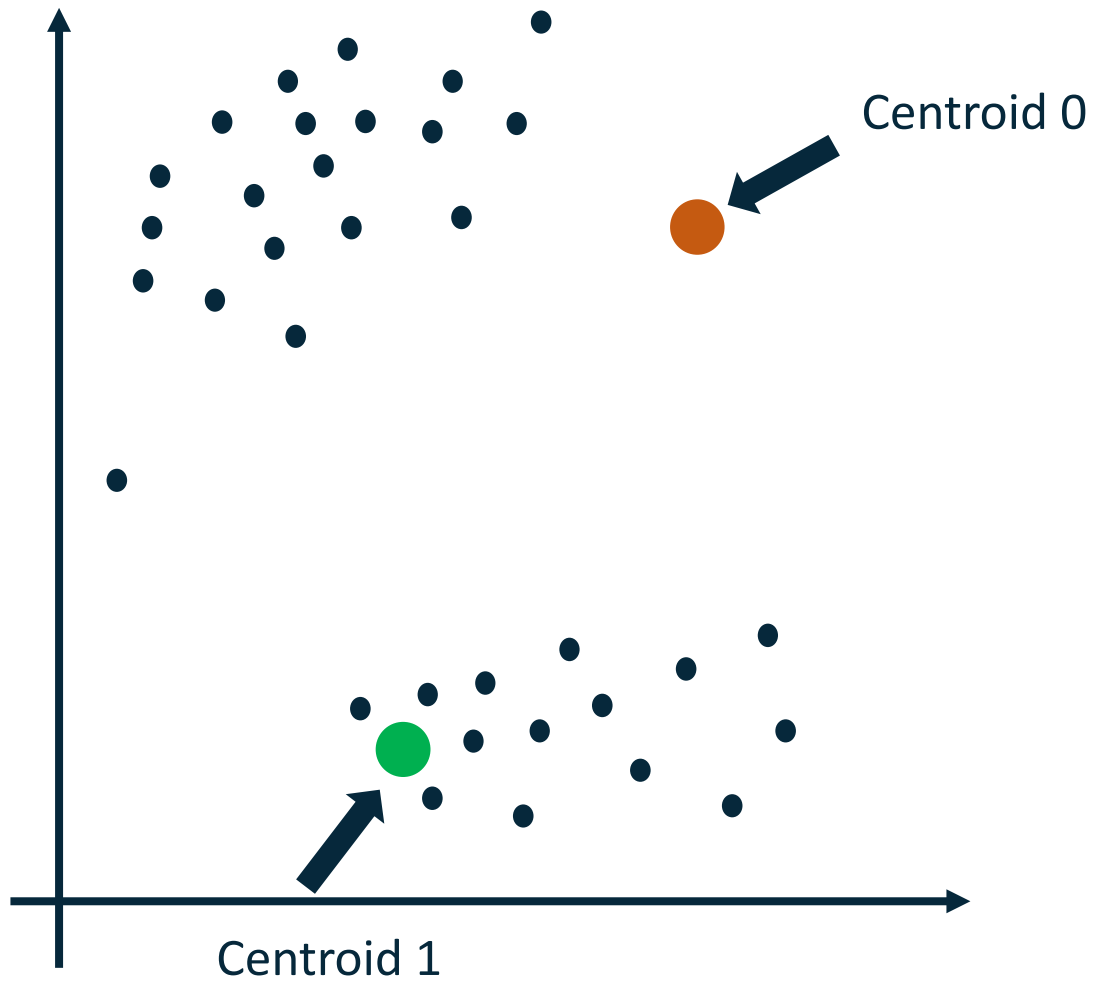


Consider a hypothetical dataset that needs to be clustered. To keep things simple, there are only two numeric features in the dataset.

For this contrived example, we'll assume that we know ahead of time from business subject matter experts that there should be two clusters in the data.

As such, we'll choose k = 2.

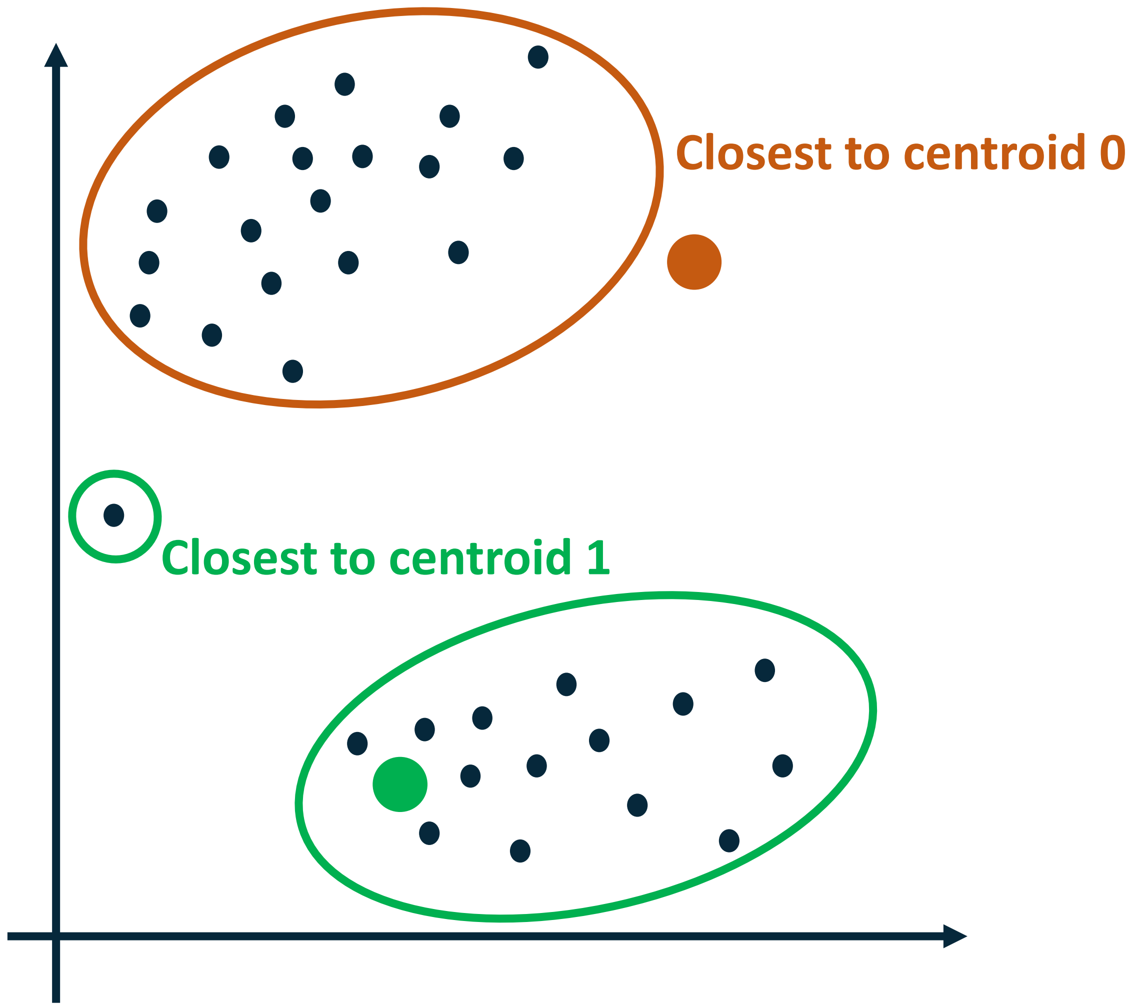
Random Start



K-means needs someplace to start working with the data.

Once the number of clusters is chosen, k-means “throws out” some random cluster centers (i.e., centroids) as a starting point.

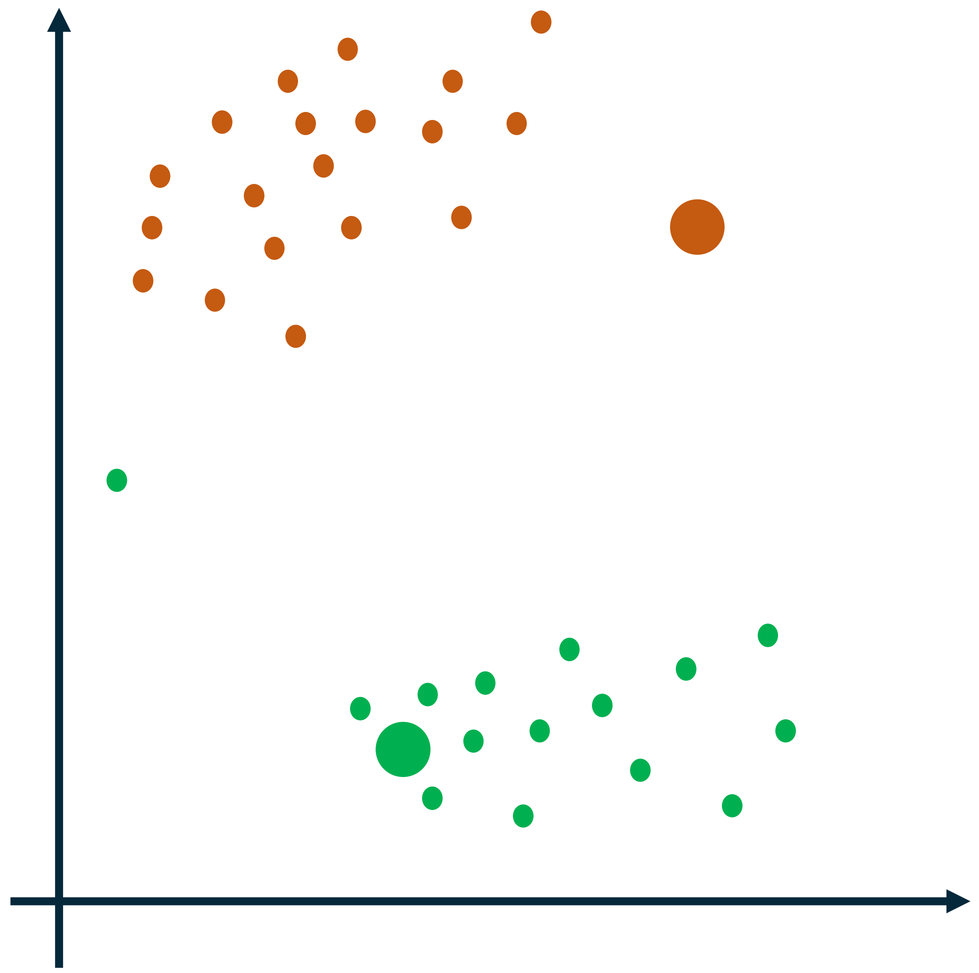
Assign Data to Clusters



K-means then looks at each "point" (e.g., documents) to cluster in turn:

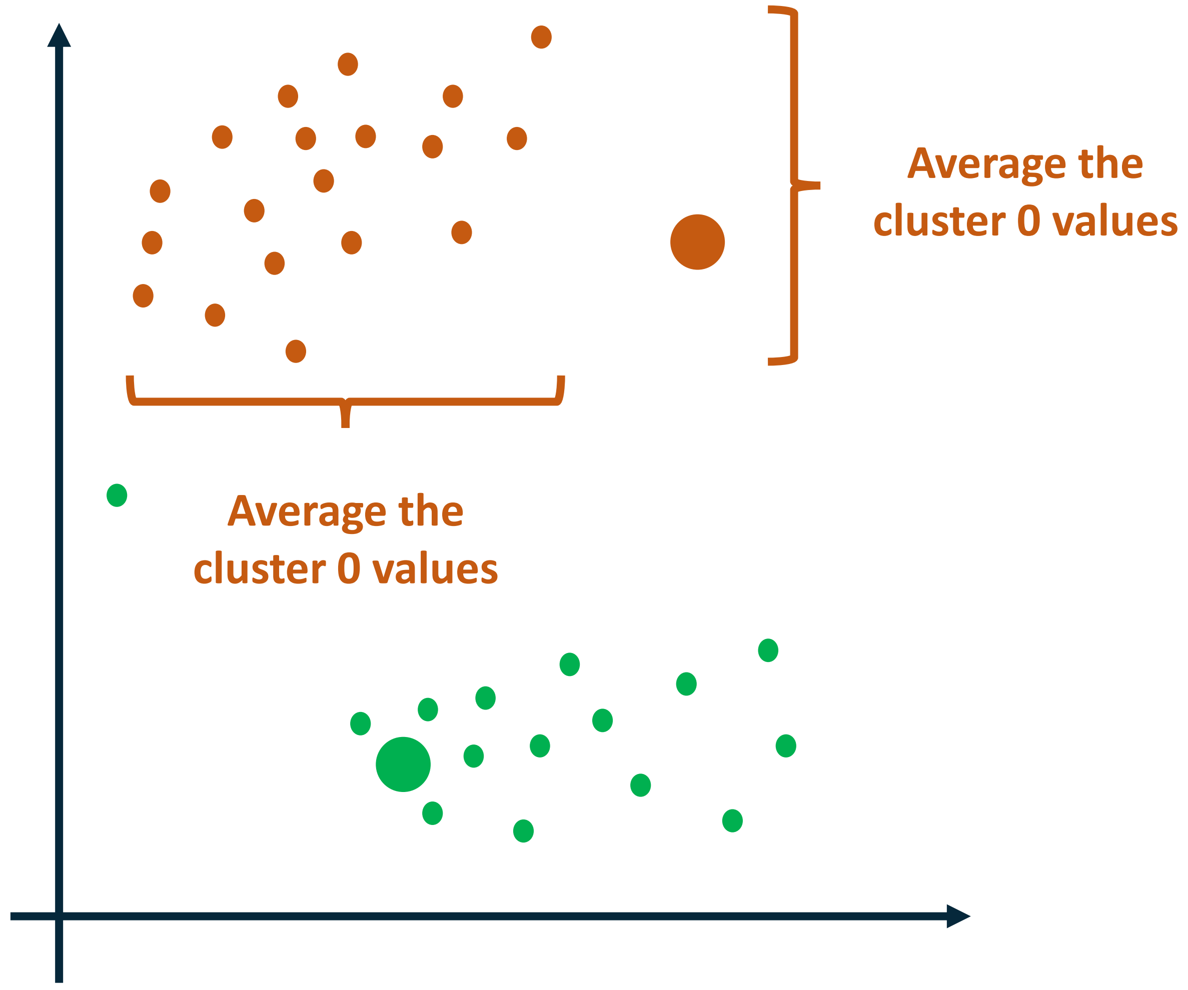
1. What's the distance from the point to centroid 0?
2. What's the distance from the point to centroid 1?

Assign Data to Clusters



Based on proximity, k-means then assigns each data point to a cluster.

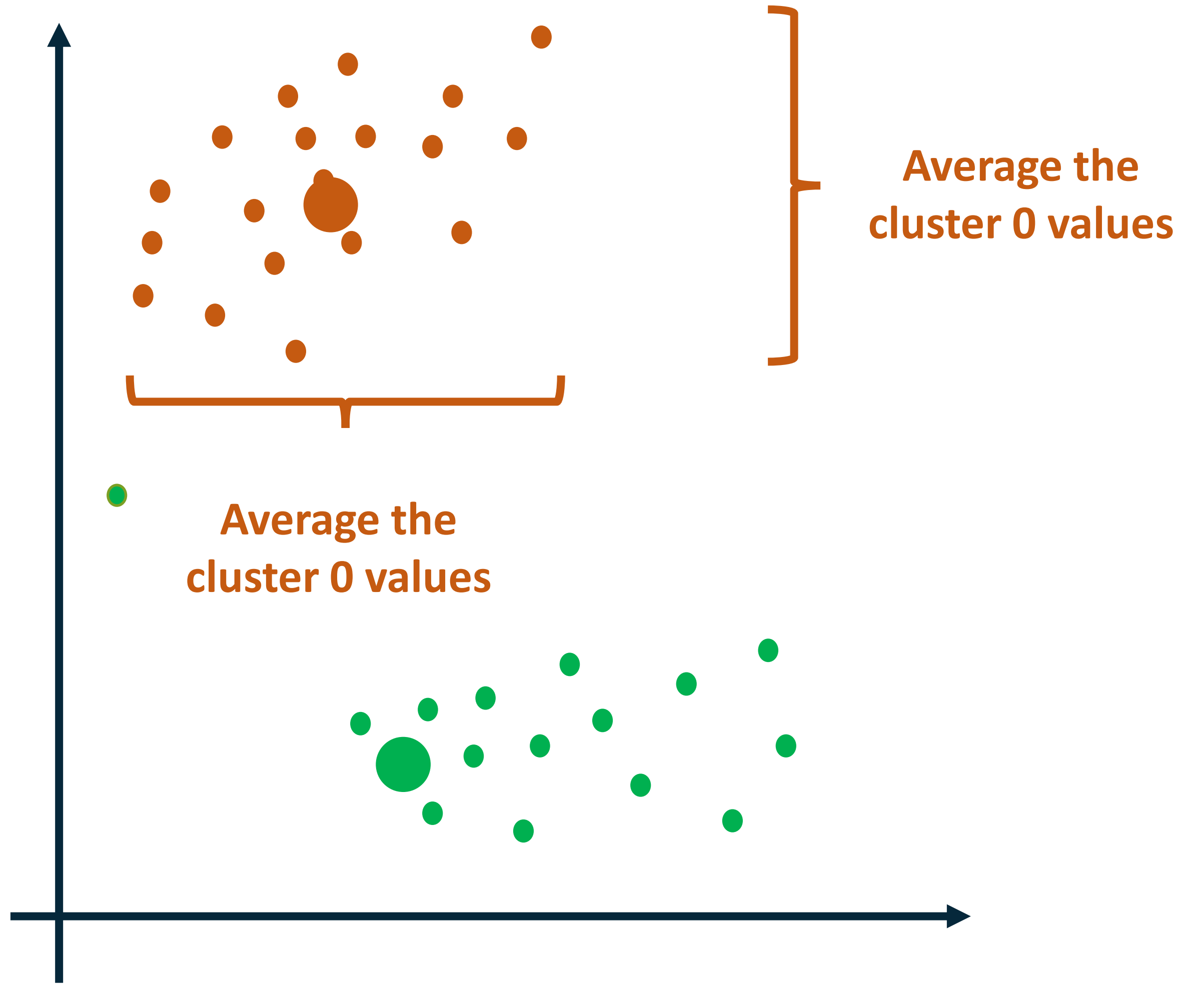
Move Clusters



Moving the centroids is where the name “k-means” comes from.

Centroids are moved based on the average values of the data points within the cluster.

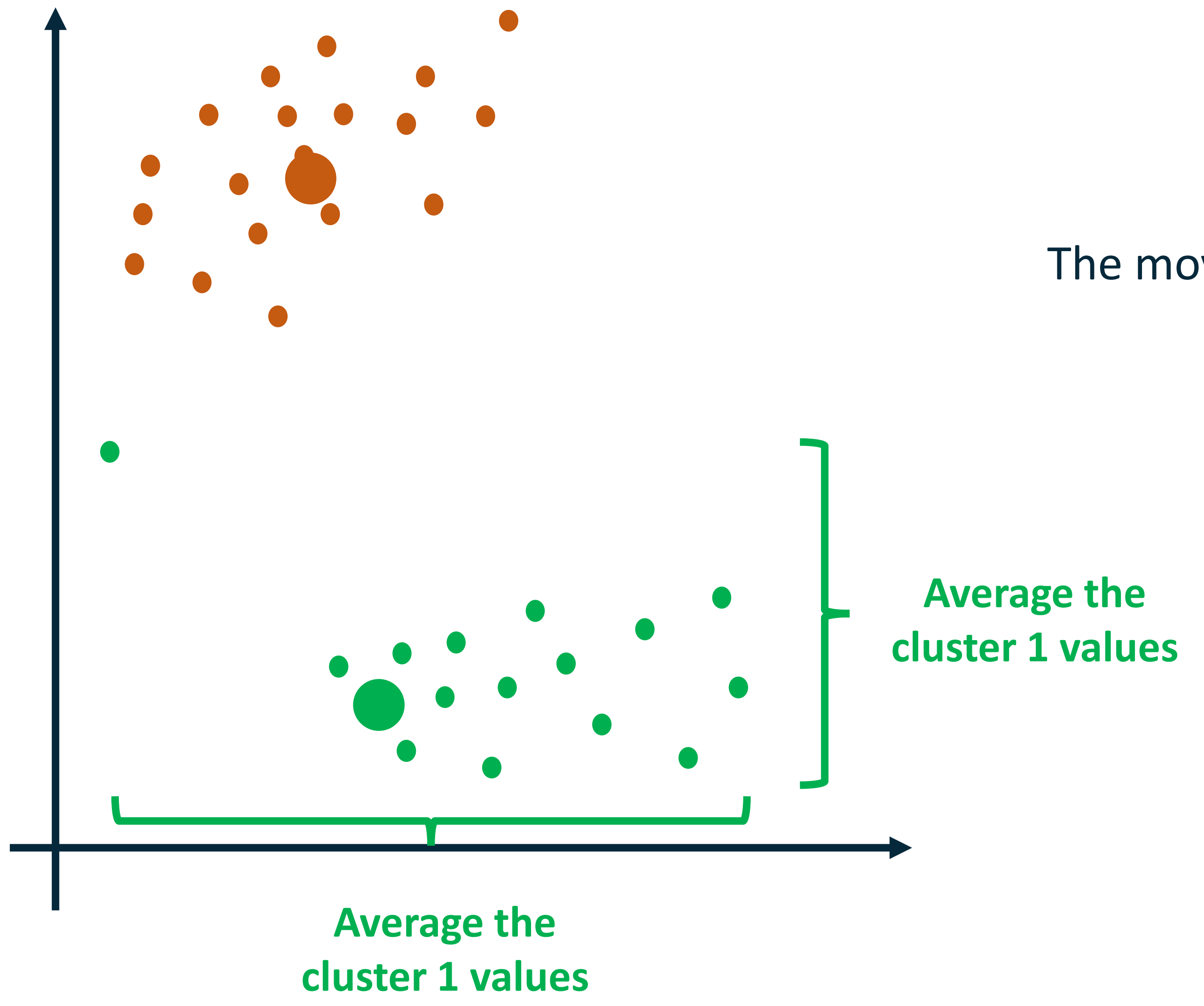
Move Clusters



Moving the centroids is where the name “k-means” comes from.

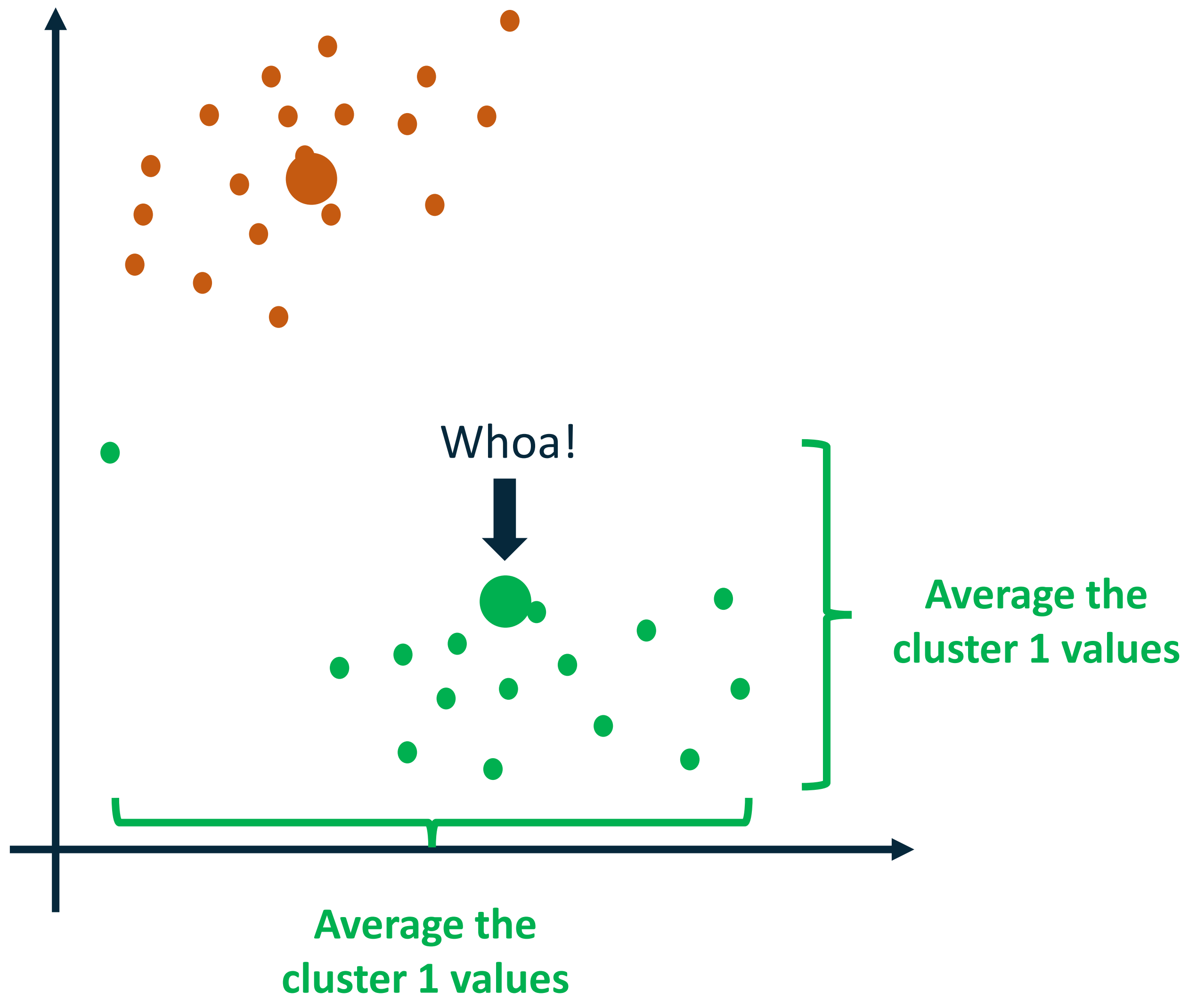
Centroids are moved based on the average values of the data points within the cluster.

Move Clusters

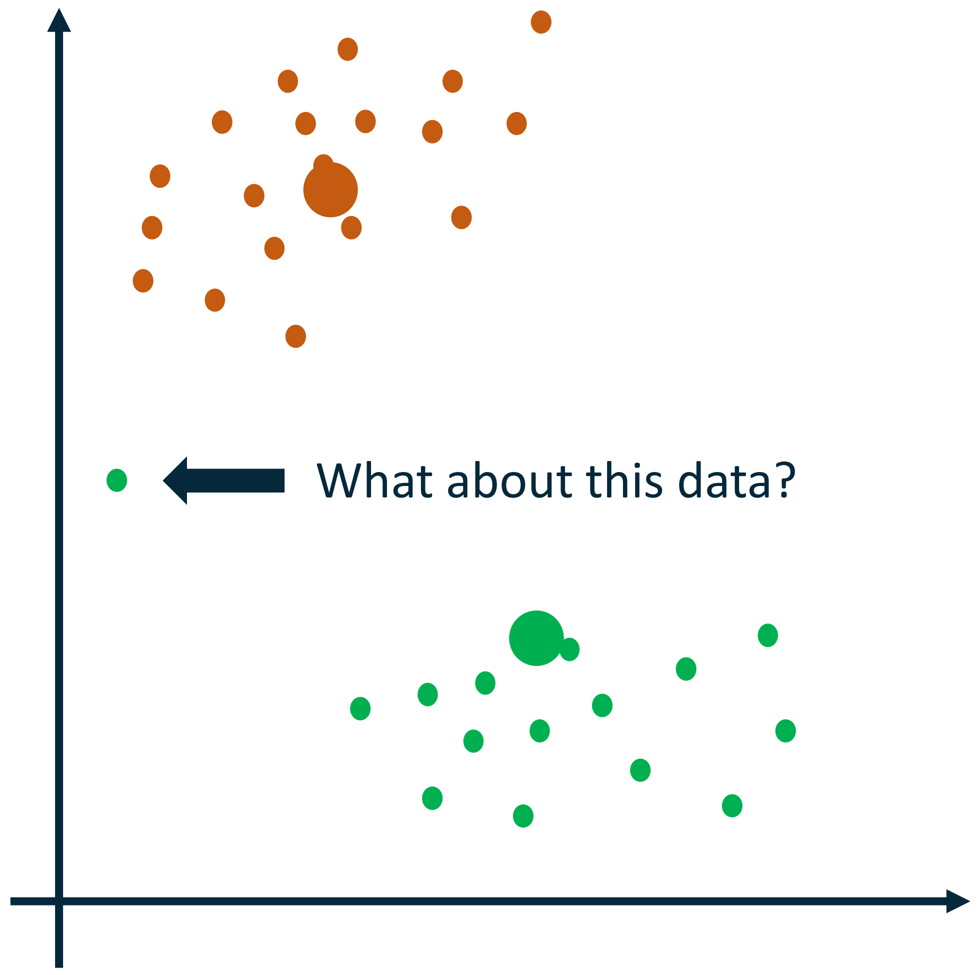


The moving process is repeated for each centroid.

Move Clusters



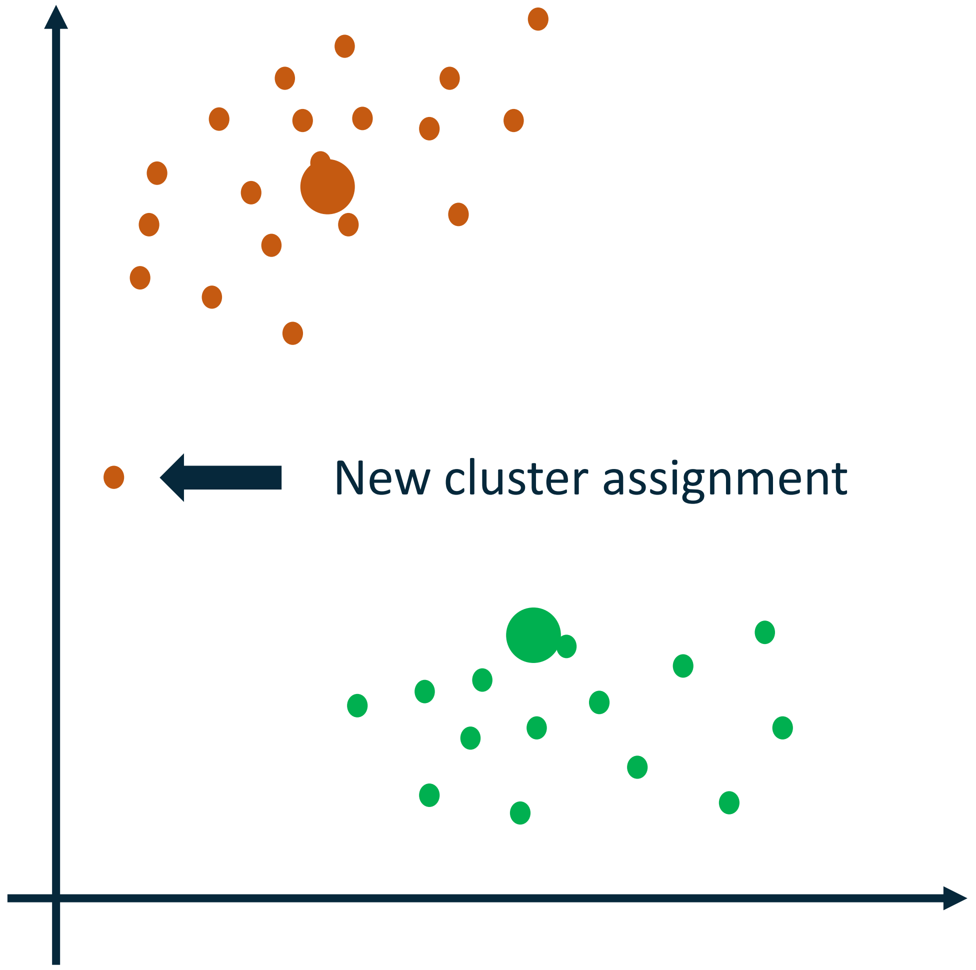
Move Clusters



K-means is an iterative process (i.e., algorithm).

The process is repeated until a stopping condition is reached.

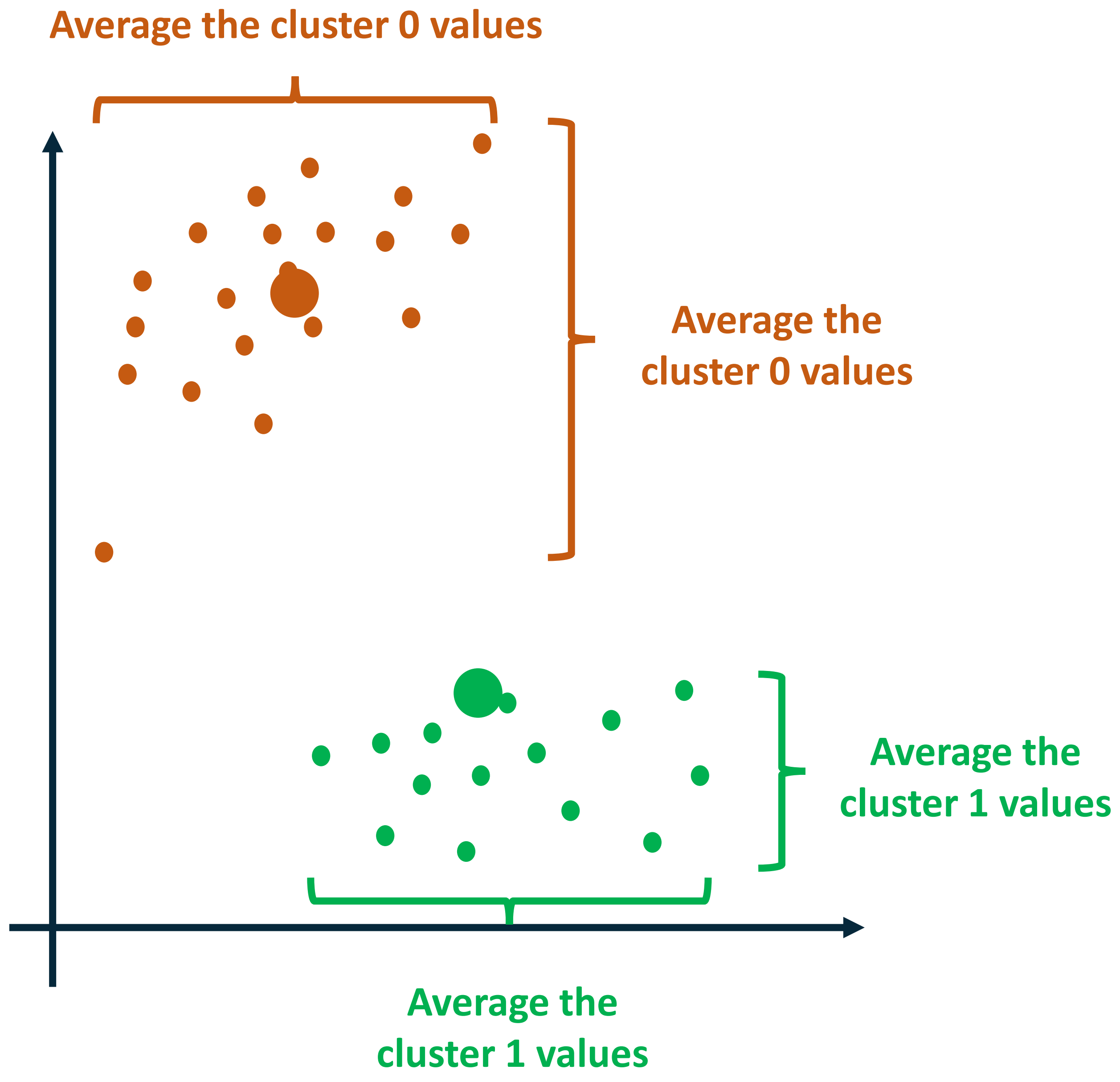
Move Clusters



K-means looks at each “point” (i.e., document) to cluster in turn:

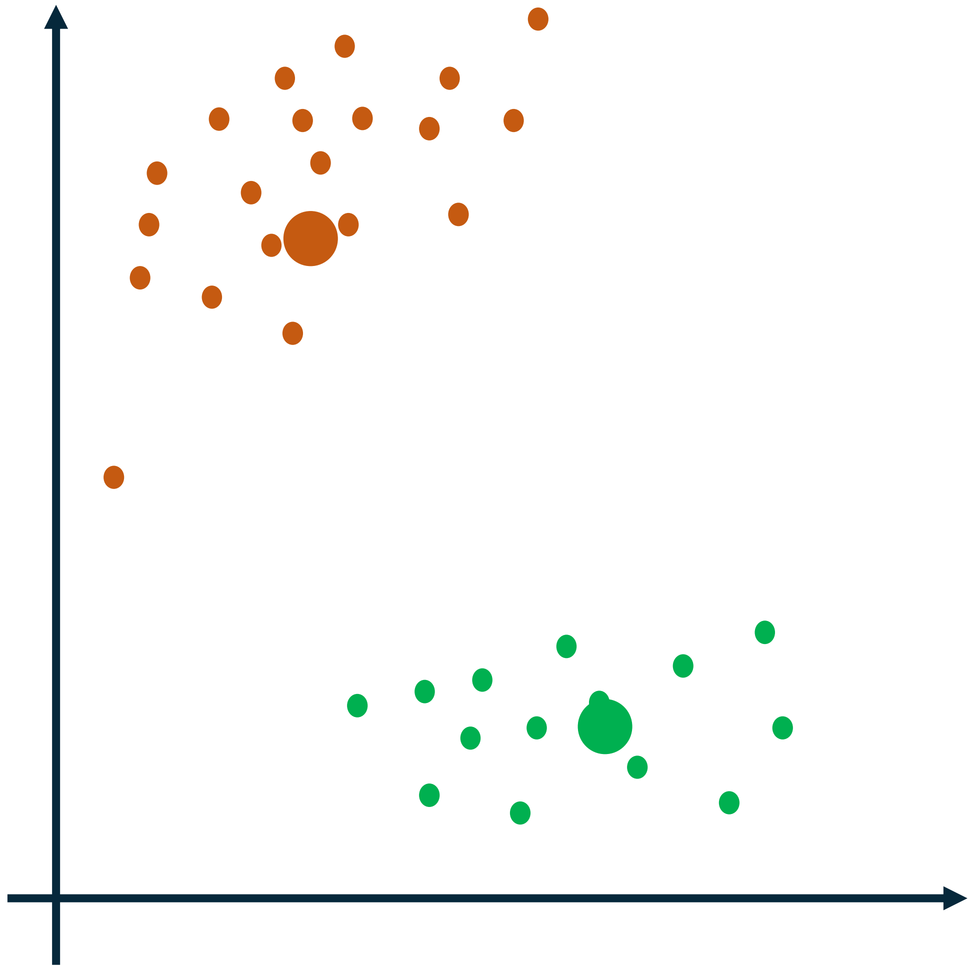
1. What’s the distance from the point to centroid 0?
2. What’s the distance from the point to centroid 1?

Move Clusters


Average the cluster 0 values
Average the cluster 0 values
Average the cluster 1 values
Average the cluster 1 values

The centroids are again moved based on the new point-to-cluster assignments.

Move Clusters & Stopping



The k-means algorithm stops when no data points are assigned to a new cluster.

What is “Close”?

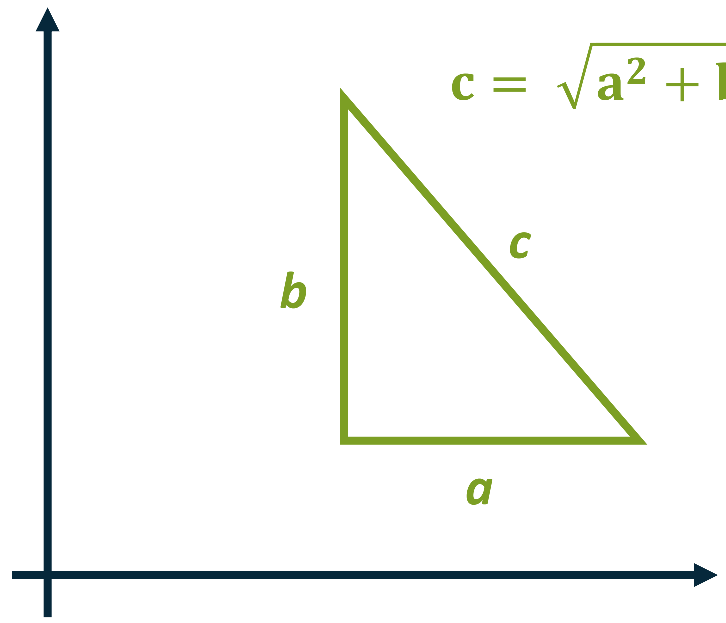
Be default, K-means uses a calculation known as **Euclidean distance**.

While it sounds complicated, it really isn’t.

Set the way back machine to intro geometry. Remember the Pythagorean theorem?

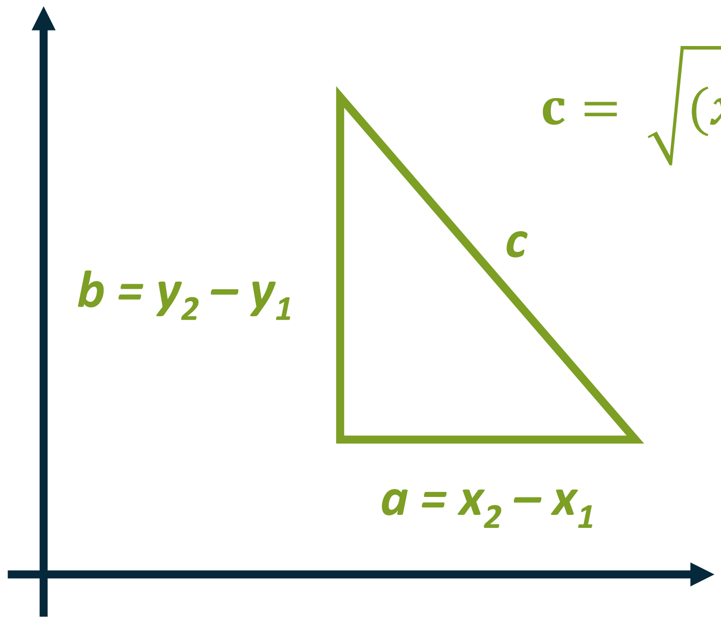
$$a^2 + b^2 = c^2$$

$$c = \sqrt{a^2 + b^2}$$

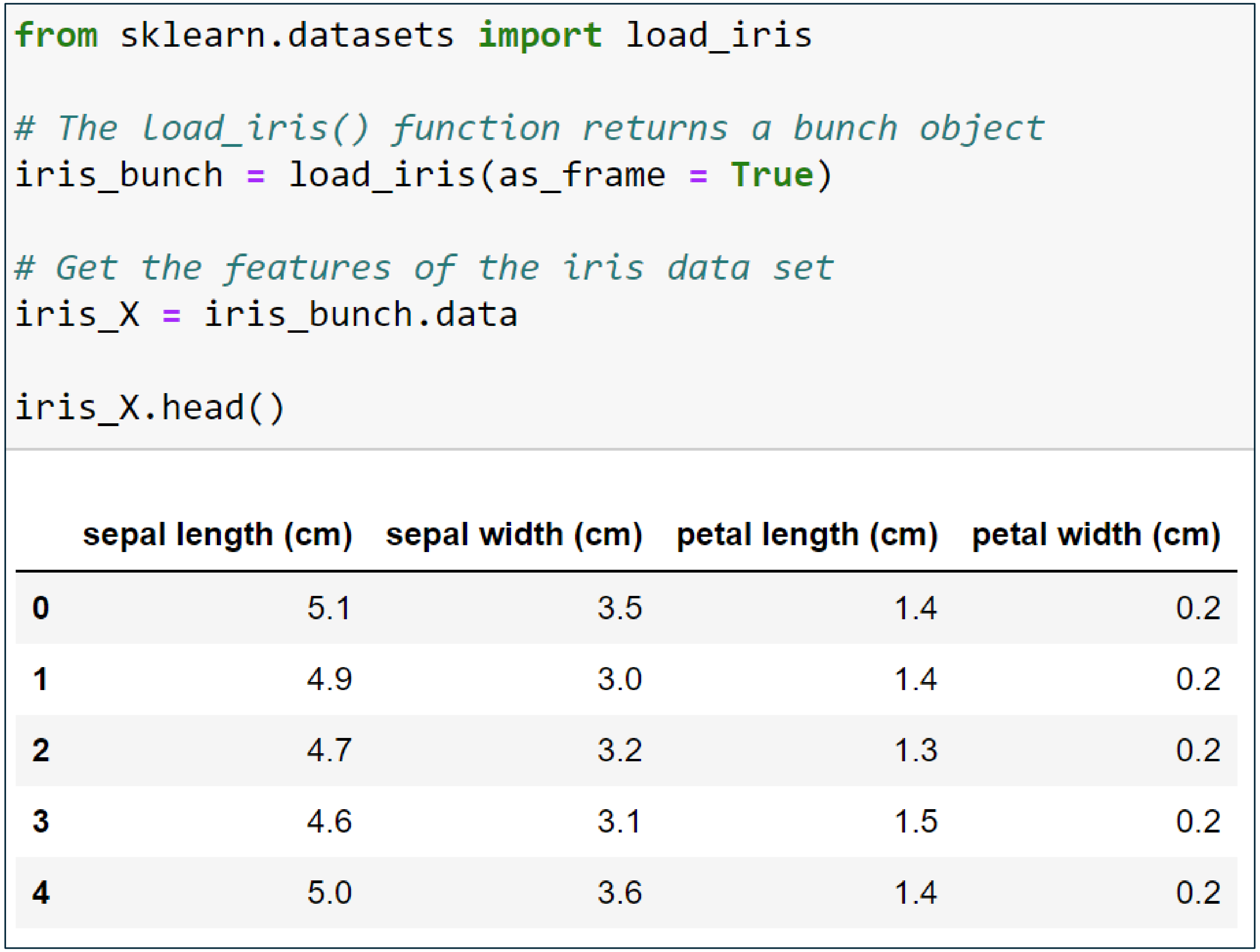


$$a^2 + b^2 = c^2$$

$$c = \sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2}$$



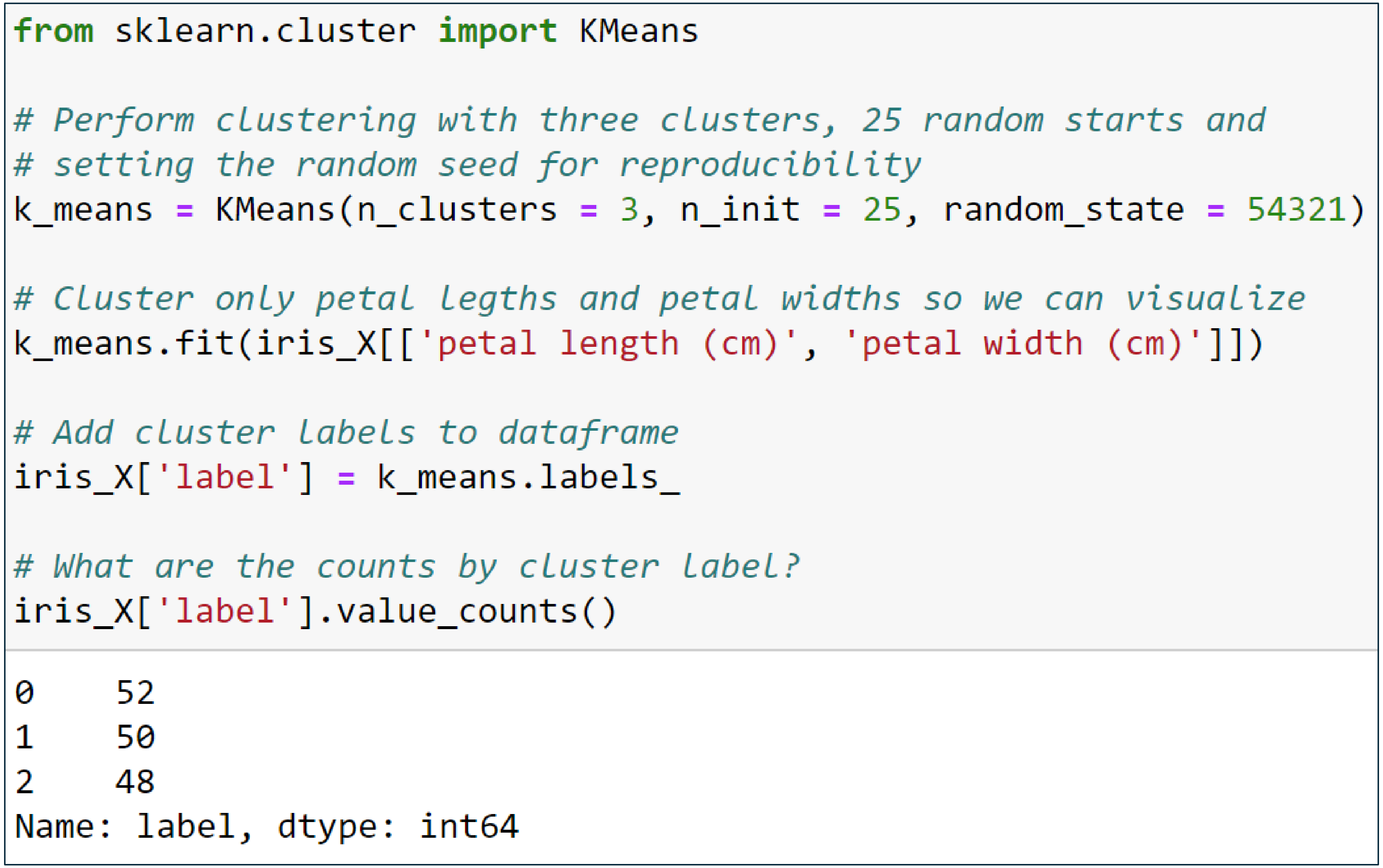
K-Means with Python



```python
from sklearn.datasets import load_iris

# The load_iris() function returns a bunch object
iris_bunch = load_iris(as_frame = True)

# Get the features of the iris data set
iris_X = iris_bunch.data

iris_X.head()
```

	sepal length (cm)	sepal width (cm)	petal length (cm)	petal width (cm)
0	5.1	3.5	1.4	0.2
1	4.9	3.0	1.4	0.2
2	4.7	3.2	1.3	0.2
3	4.6	3.1	1.5	0.2
4	5.0	3.6	1.4	0.2

K-Means with Python



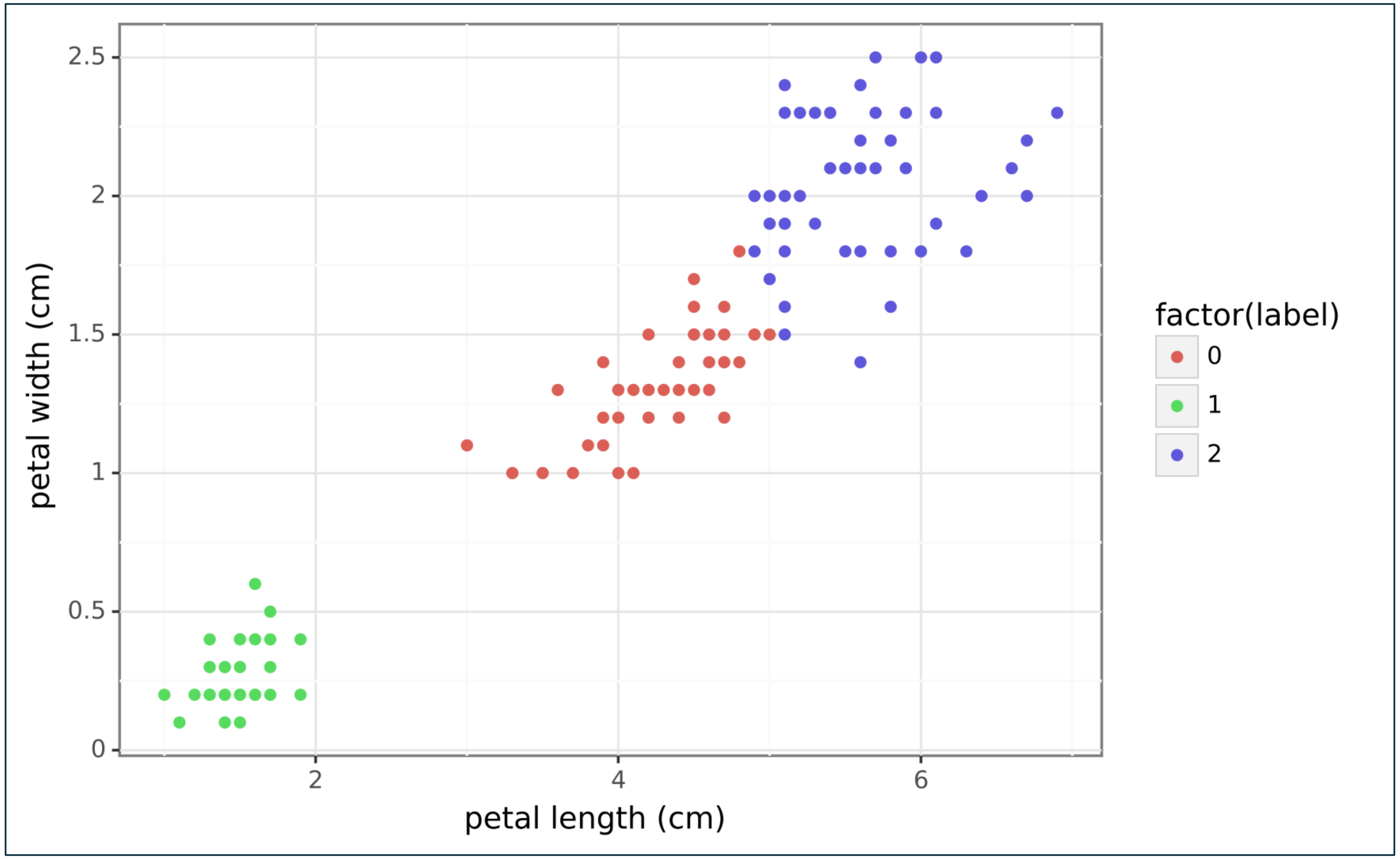
```python
from sklearn.cluster import KMeans

# Perform clustering with three clusters, 25 random starts and
# setting the random seed for reproducibility
k_means = KMeans(n_clusters = 3, n_init = 25, random_state = 54321)

# Cluster only petal legths and petal widths so we can visualize
k_means.fit(iris_X[['petal length (cm)', 'petal width (cm)']])

# Add cluster labels to dataframe
iris_X['label'] = k_means.labels_

# What are the counts by cluster label?
iris_X['label'].value_counts()
```

```
0    52
1    50
2    48
Name: label, dtype: int64
```

Visualizing Clusters

```python
# Create a visualization of cluster assignments
from plotnine import ggplot, geom_point, aes, theme_bw



(ggplot(iris_X, aes(x = 'petal length (cm)', y = 'petal width (cm)')) +
 theme_bw() +
 geom_point(aes(color = 'factor(label)')))
```

Clustering Results

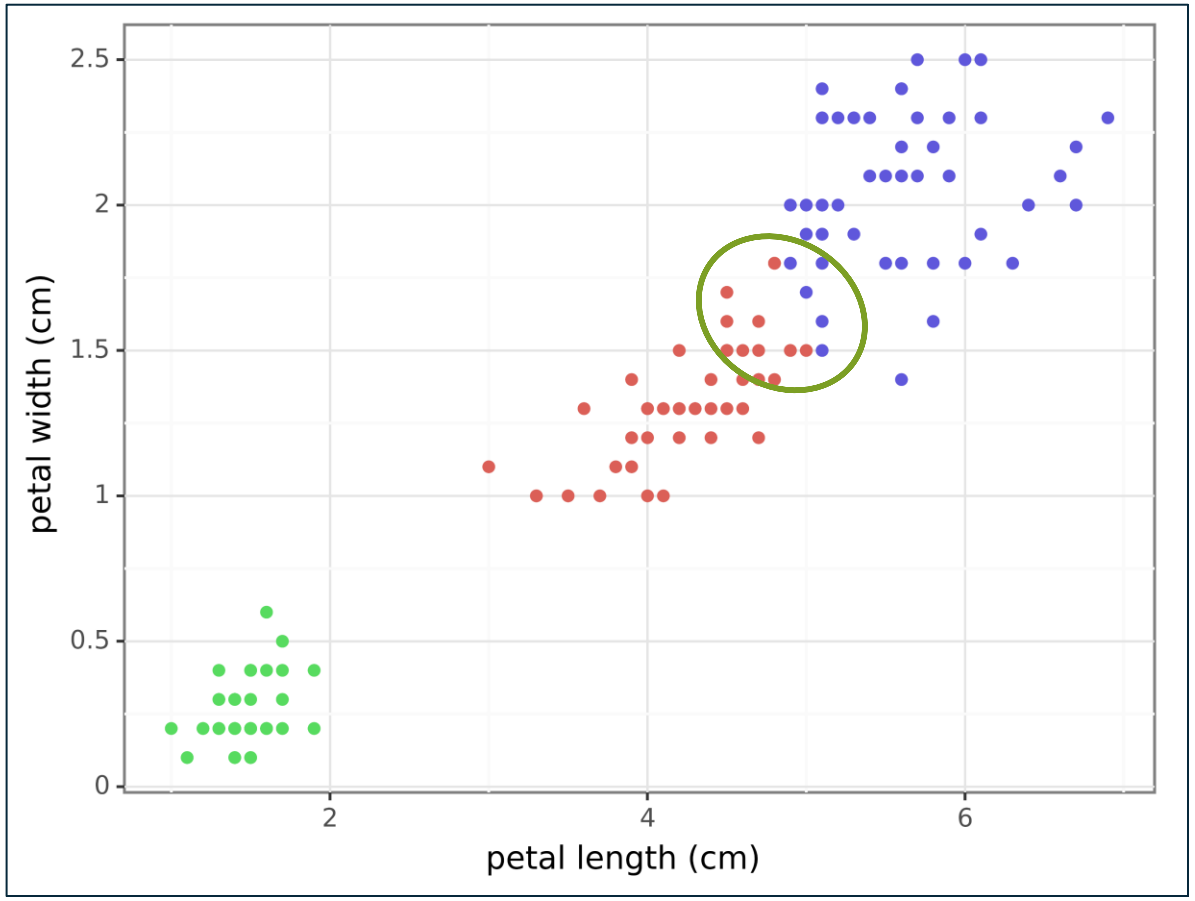


Iris Ground Truth

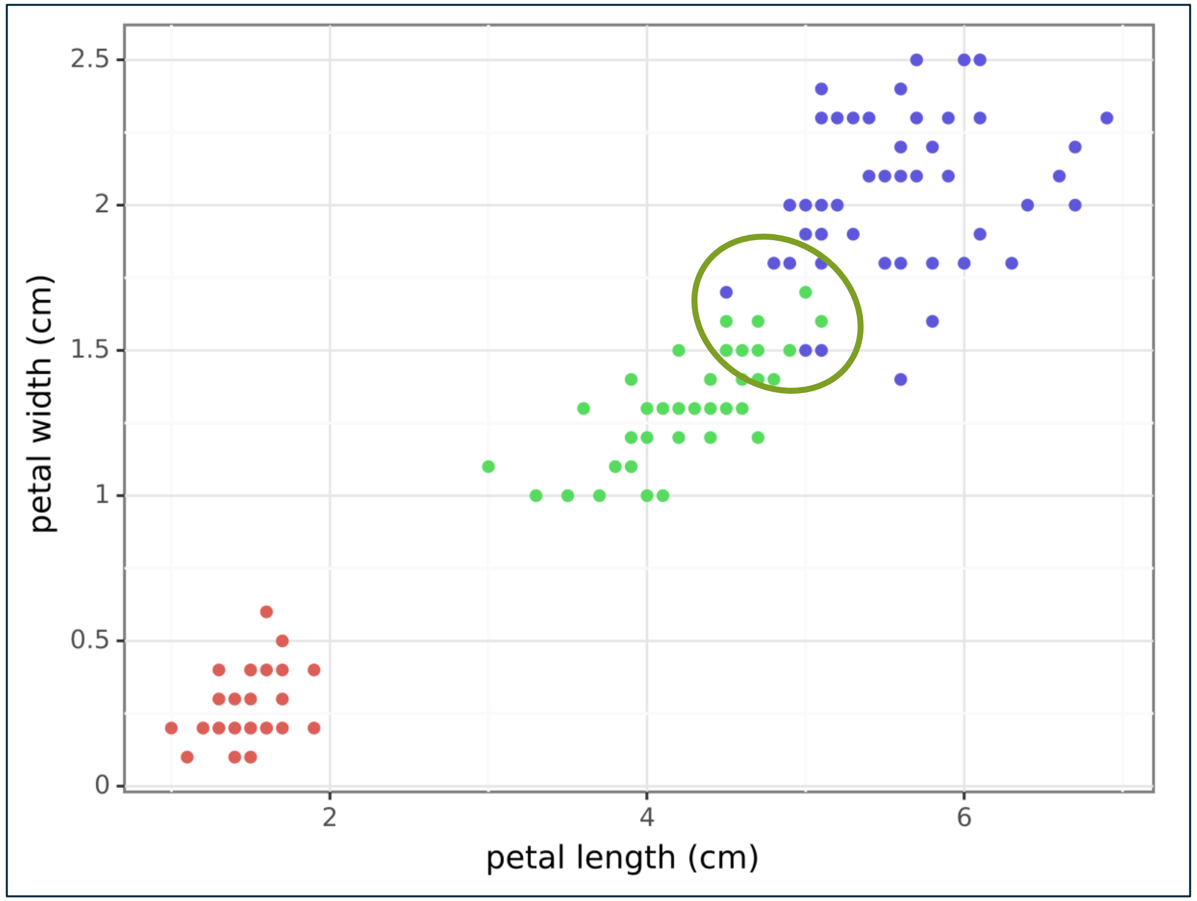
```
# Get the actual labels
iris_Y = iris_bunch.target

(ggplot(iris_X, aes(x = 'petal length (cm)', y = 'petal width (cm)')) +
 theme_bw() +
 geom_point(aes(color = 'factor(iris_Y)')))
```

Evaluating the Clusters

K-means clusters



Actual clusters



K-Means Caveats

K-means requires the number of clusters to be chosen by the data analyst:

- Relying on subject matter expertise to determine the value of k
- The scree plot elbow method
- The average silhouette score

Given the random starting points, k-means is a **non-deterministic** clustering technique:

- Best practice is to run k-means multiple times with the same k and evaluate clusters across runs
- If you need reproducibility, be sure to set the random seed

The Euclidian distance calculation technically only works with numeric data (i.e., non-categorical):

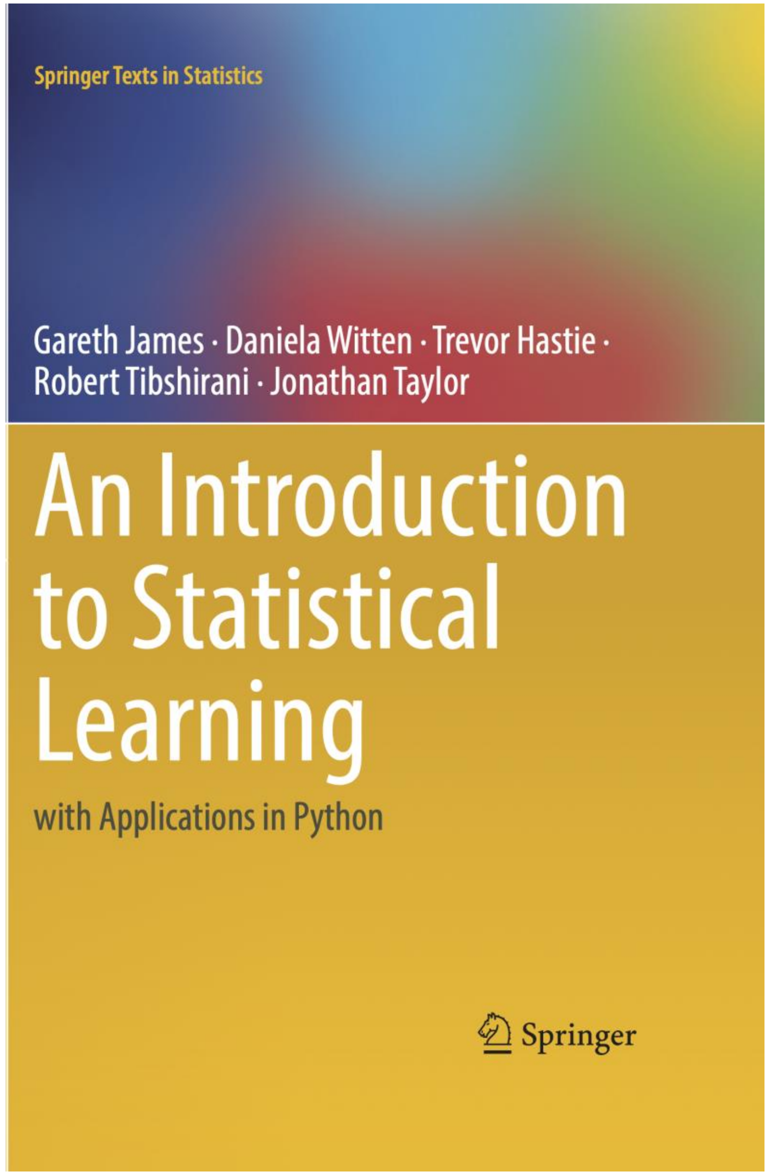
- The resources at the end of the webinar will help you with this

K-means only produces "clouds" of clustered data points:

- If the natural shape of the clusters is irregular, k-means is a less effective method

Wrap-Up

Continue Your Learning



Springer Texts in Statistics

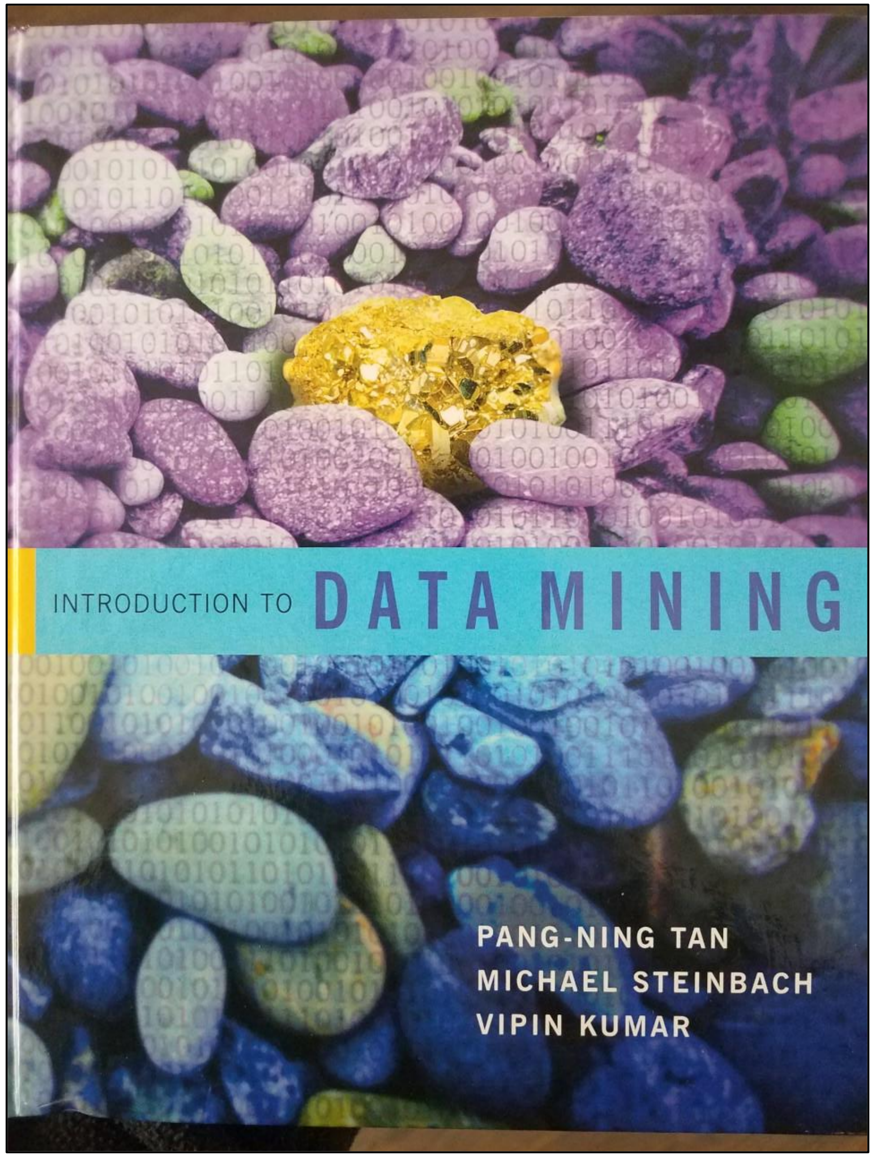
Gareth James · Daniela Witten · Trevor Hastie · Robert Tibshirani · Jonathan Taylor

An Introduction to Statistical Learning

with Applications in Python

Springer

This is the 1st edition. →



INTRODUCTION TO DATA MINING

PANG-NING TAN
MICHAEL STEINBACH
VIPIN KUMAR

Dave ON DATA

THANK YOU!