Machine Learning Applications

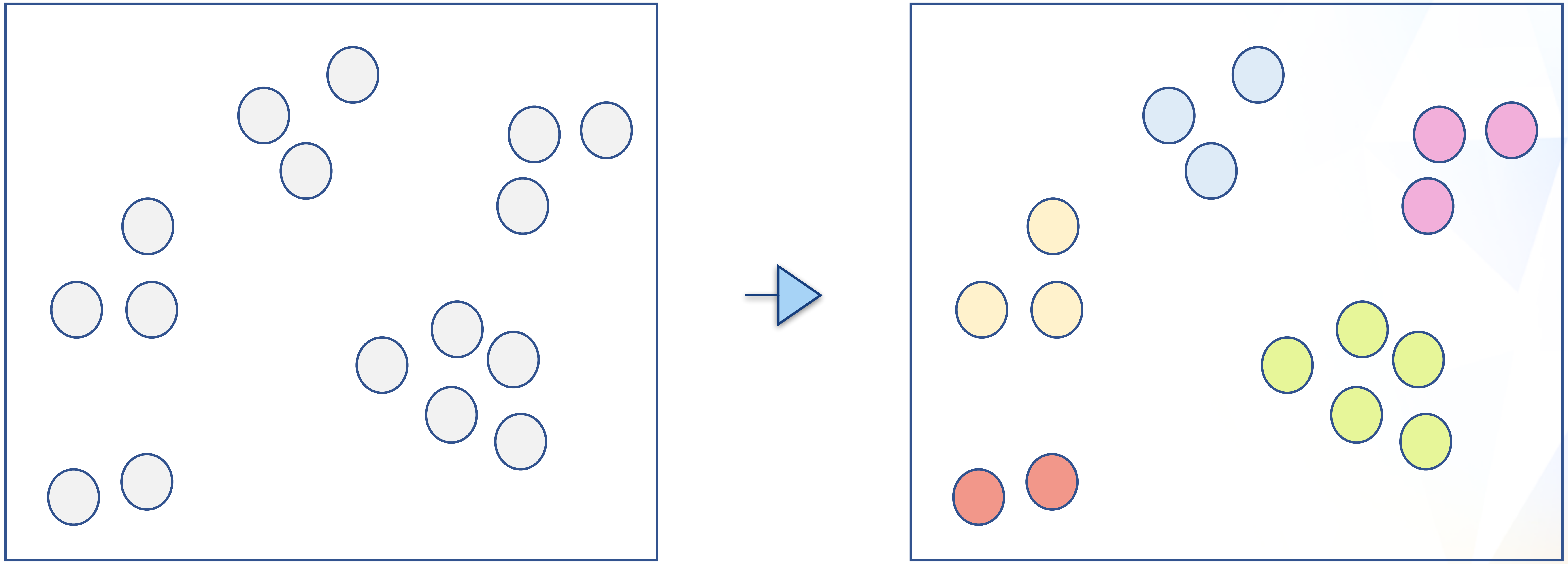
Clustering

Clustering Objective

- The goal of clustering is to discover hidden structure in our data
- Clustering, therefore, is a form of data mining. We group similar data together to deduce additional meaning
- For example, Amazon might cluster its customers based on their annual spending and buying patterns to direct promotions to them

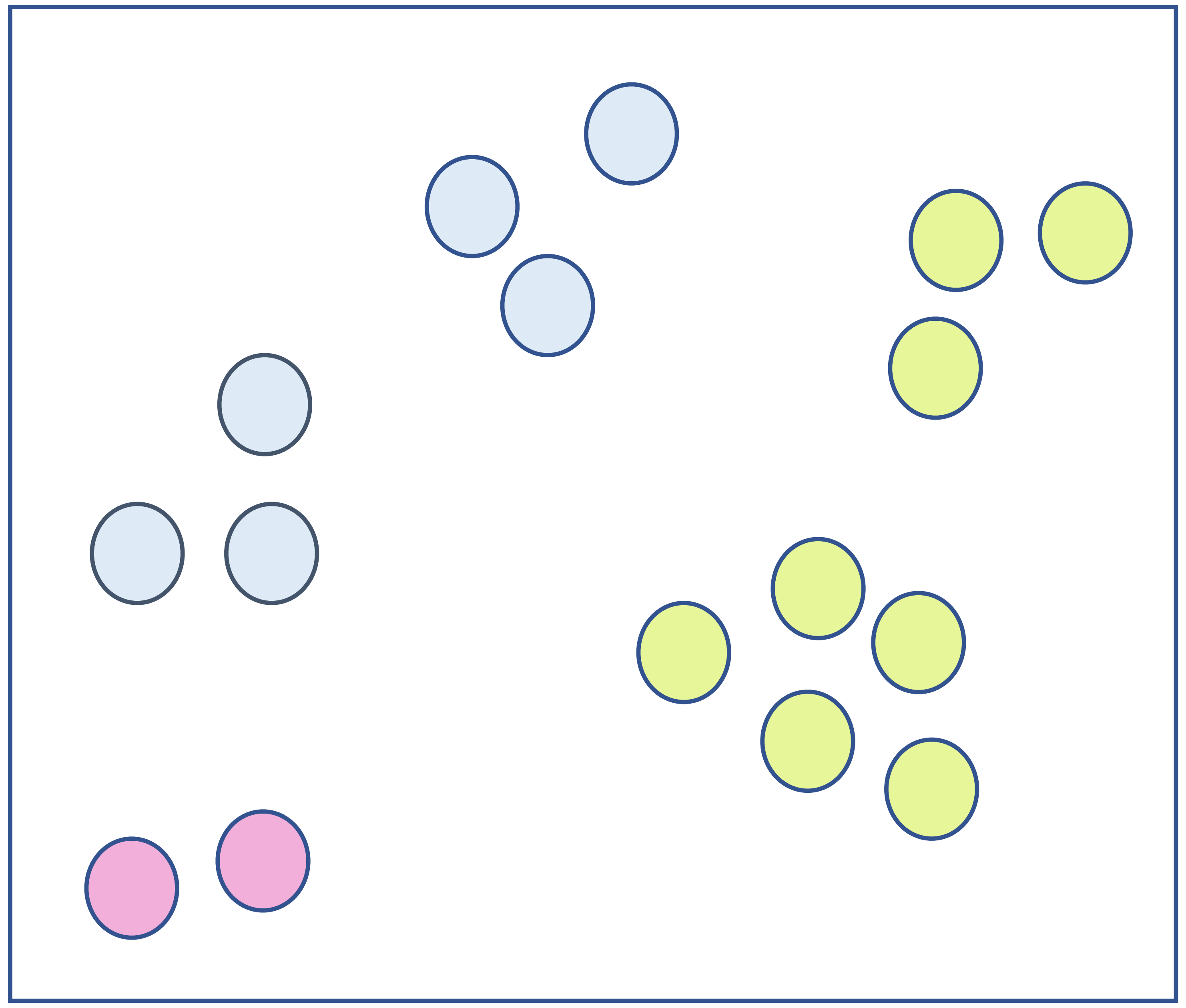
Clustering

- Find possible groupings in our data
- Group similar data points to the same cluster

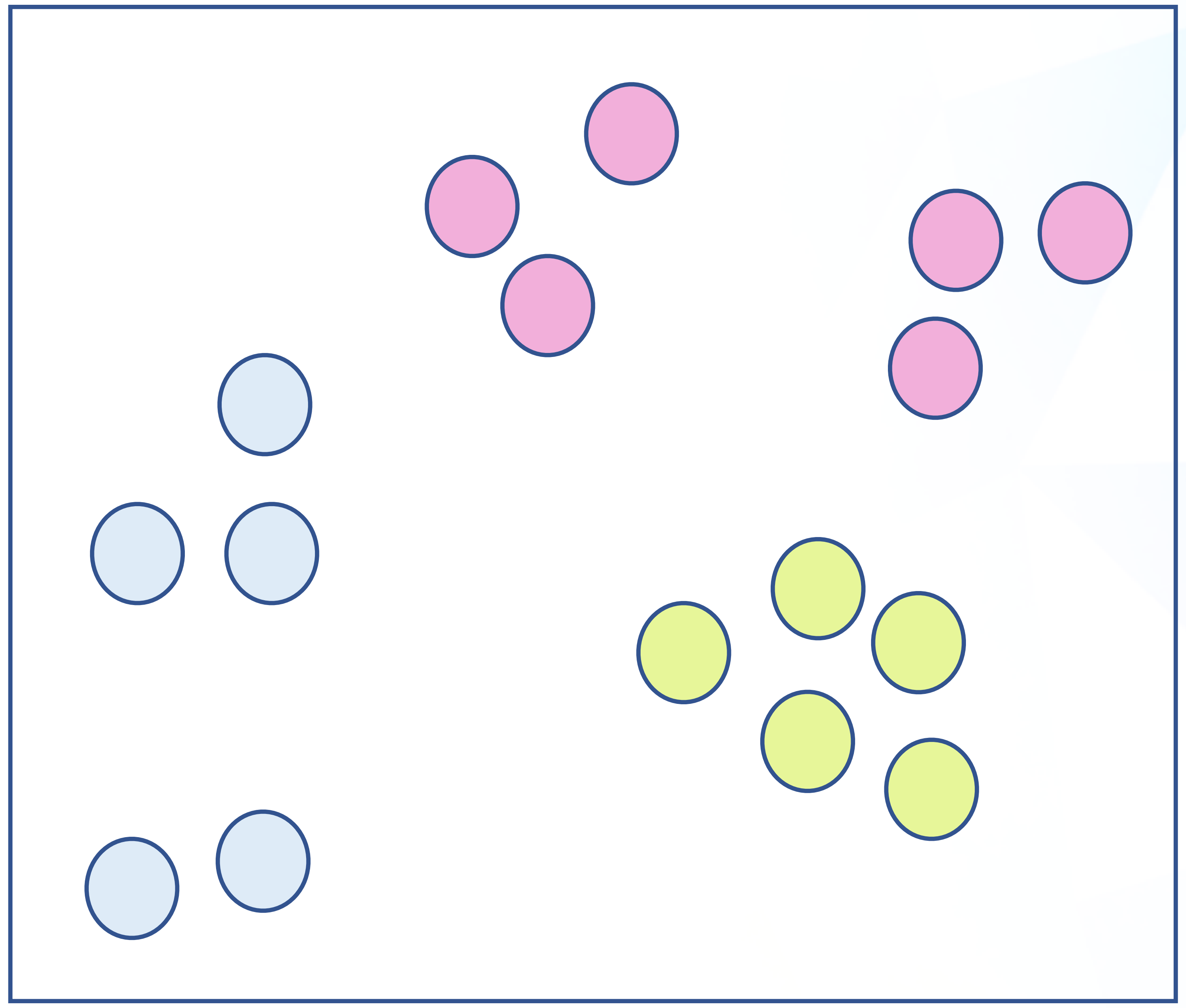


Clustering

- Different clustering algorithms can yield different results



Algorithm 1

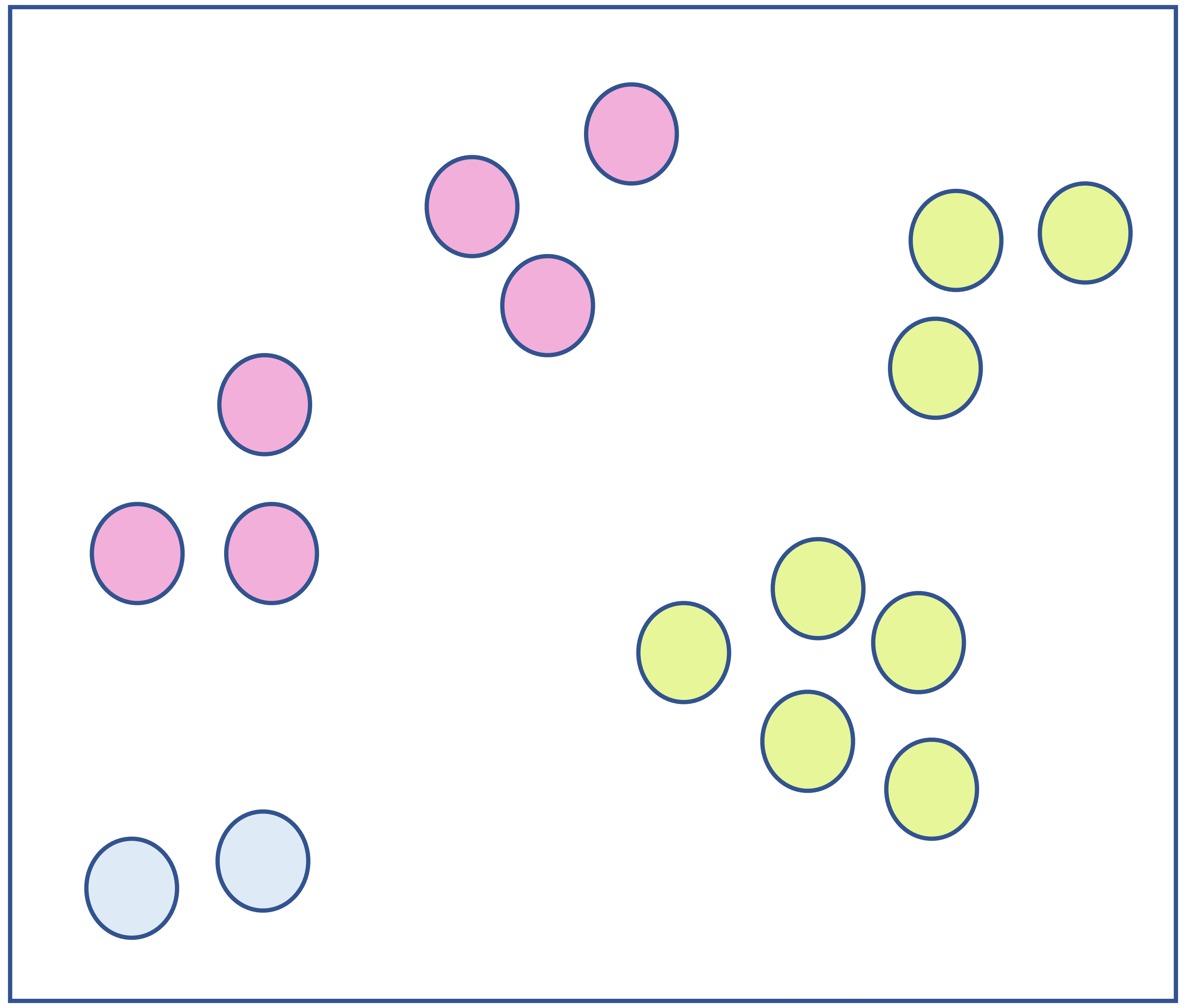


Algorithm 2

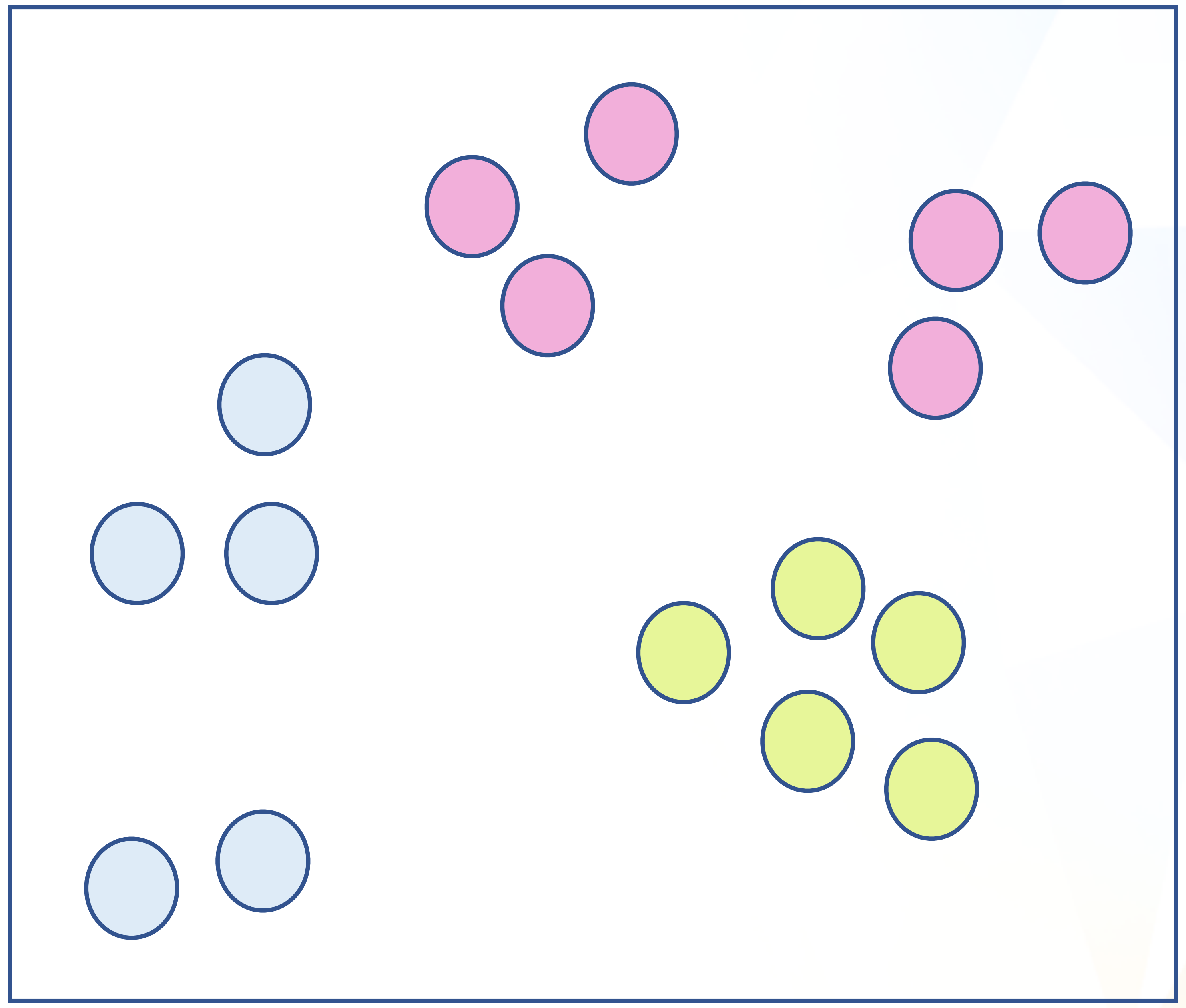
Clustering

The same algorithm can produce different outcomes

- Initialization methods
- Seed values for random number generator
- Distance formulas



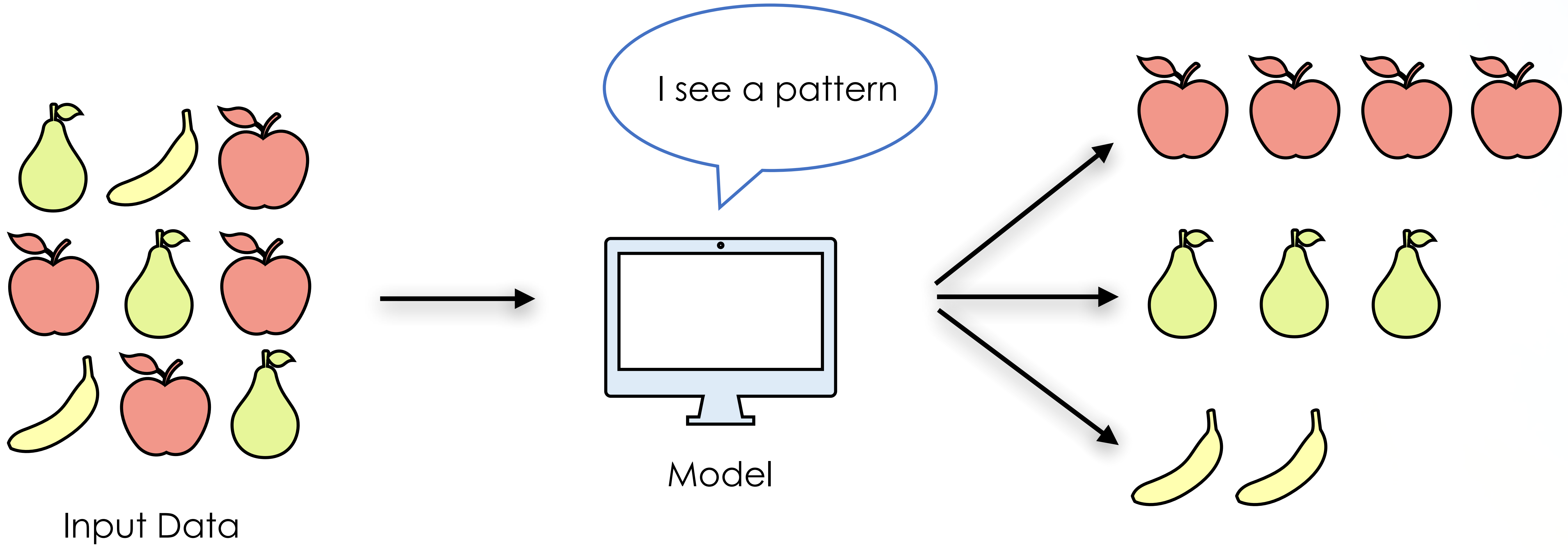
Algorithm 1 (Init Method A)



Algorithm 1 (Init Method B)

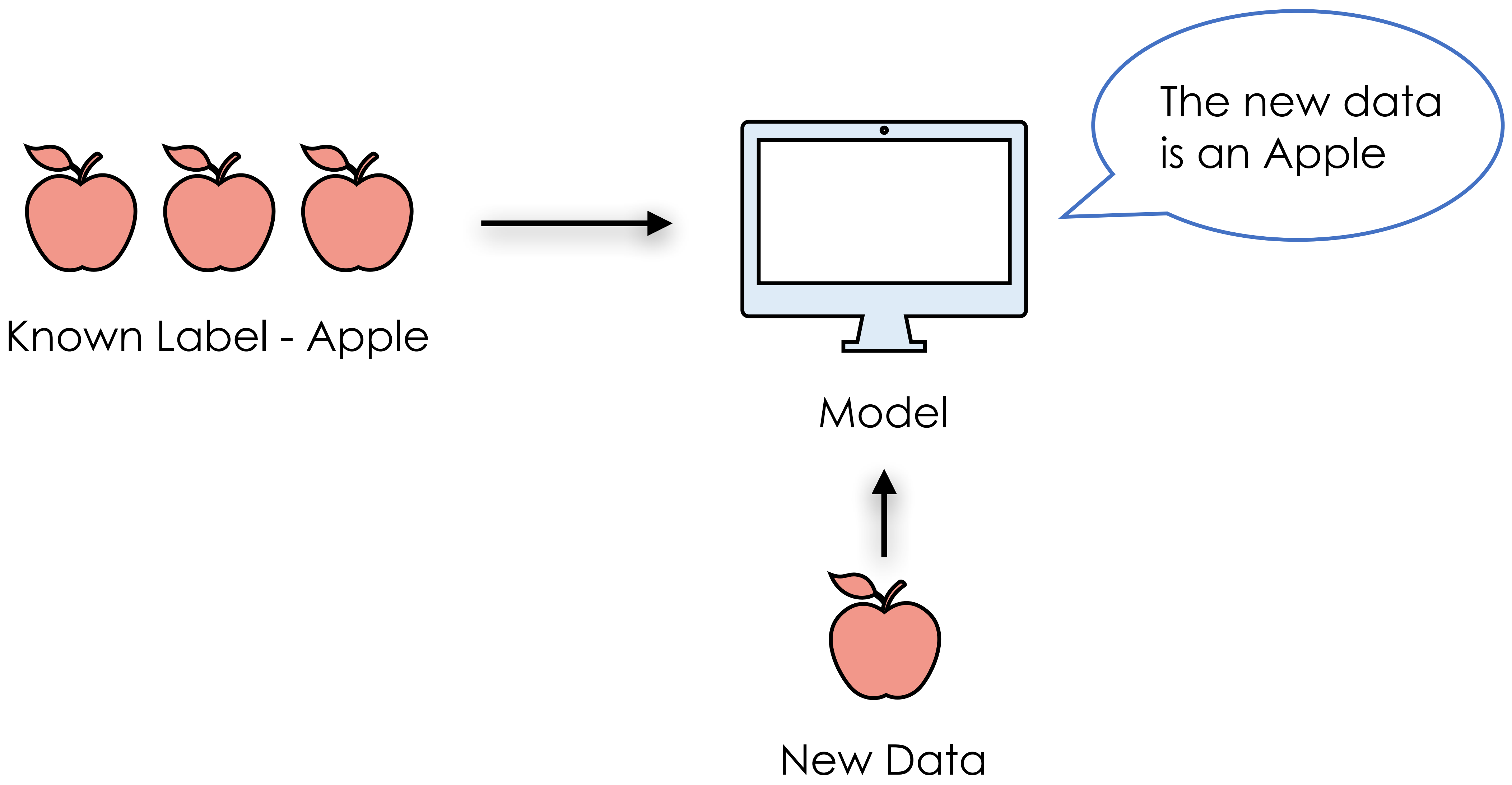
Unsupervised Learning

- Clustering belongs to the class of learning algorithms called Unsupervised Learning
- Unsupervised Learning works by looking for existing patterns in our data set



Supervised Learning

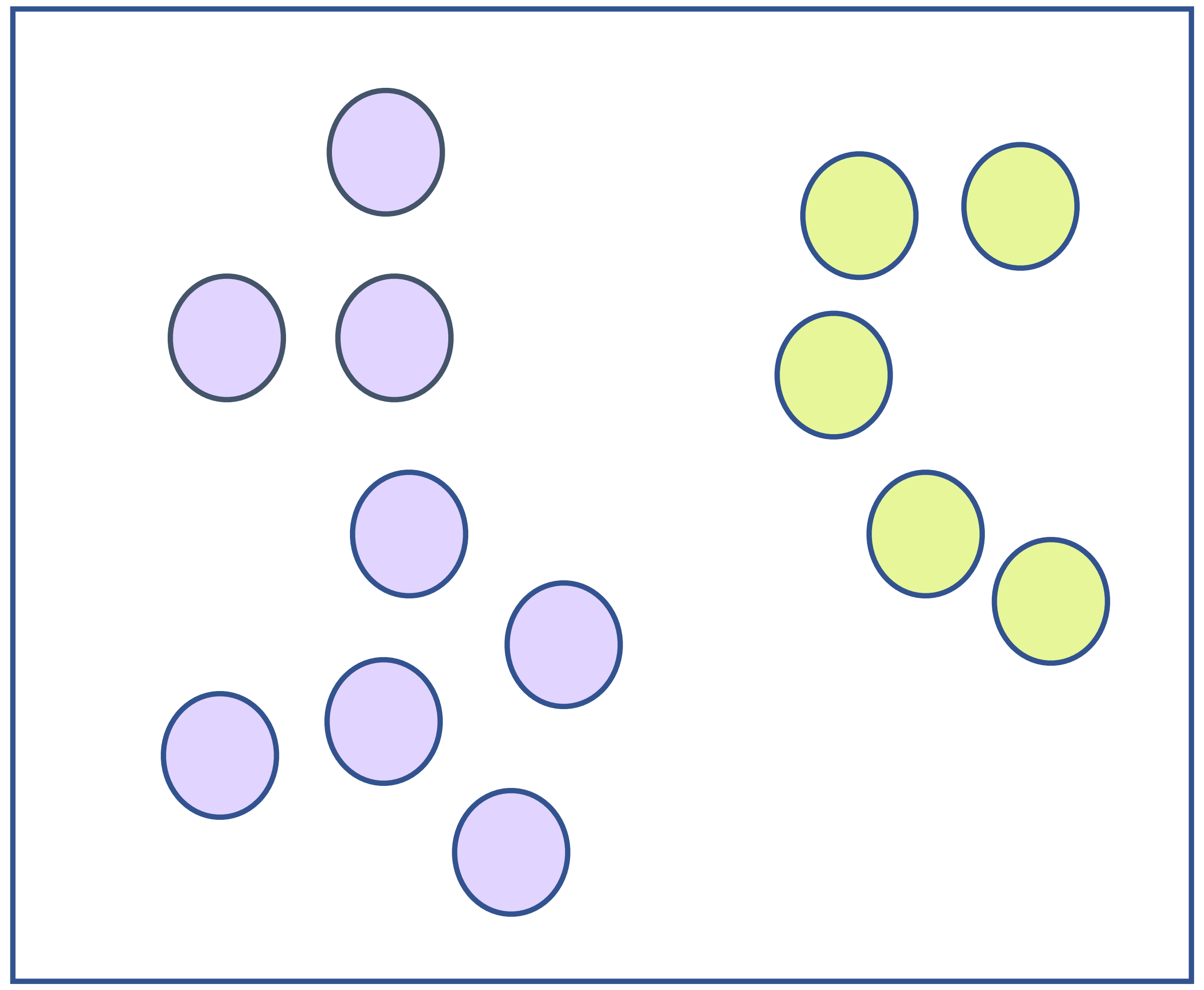
- In contrast, Supervised Learning uses data with known labels
- Here, the Model is trained to identify an Apple (a known label)



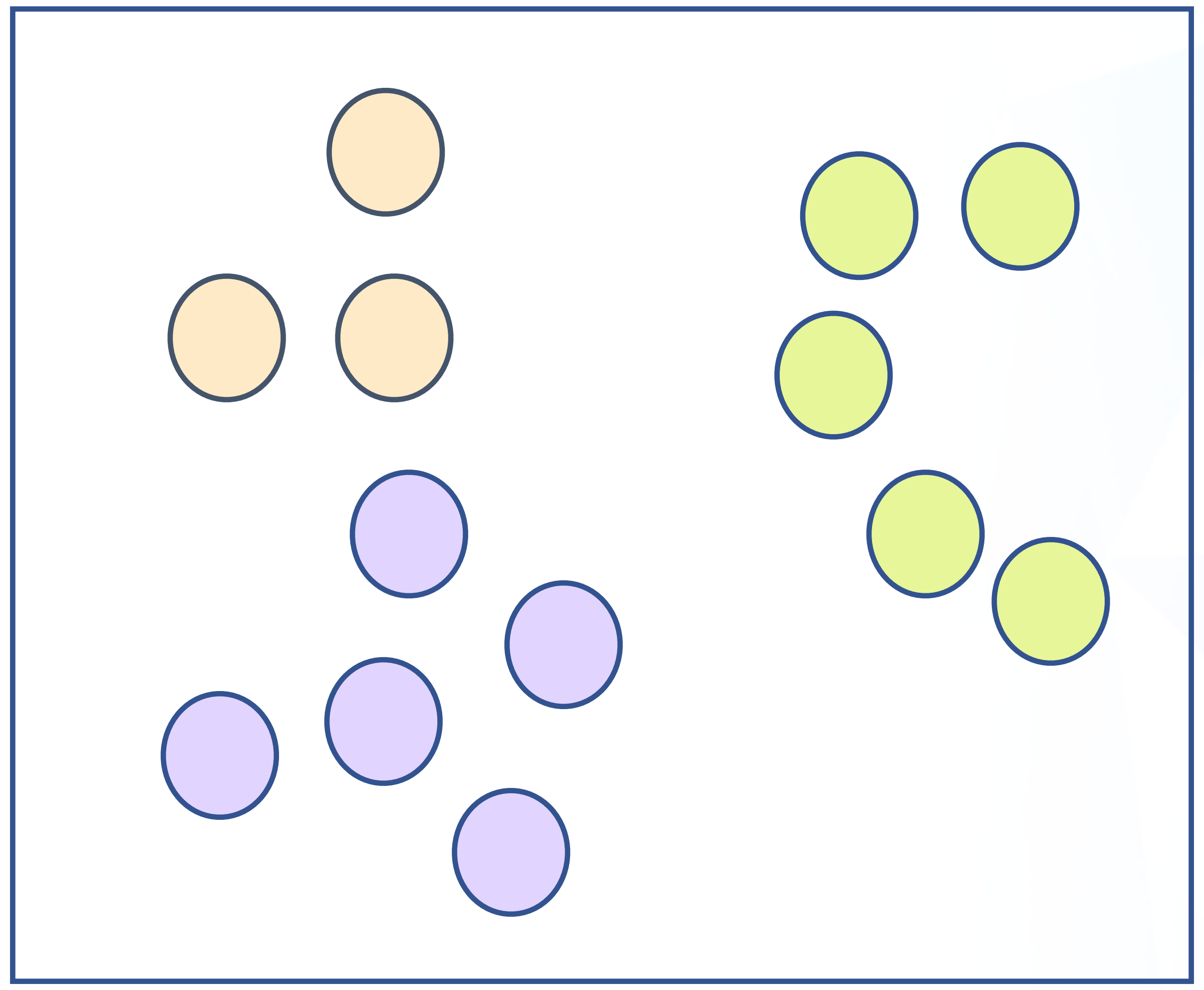
K-Means

Hyperparameter

A Hyper-parameter is a value that is fed to a learning model before computation



K = 2

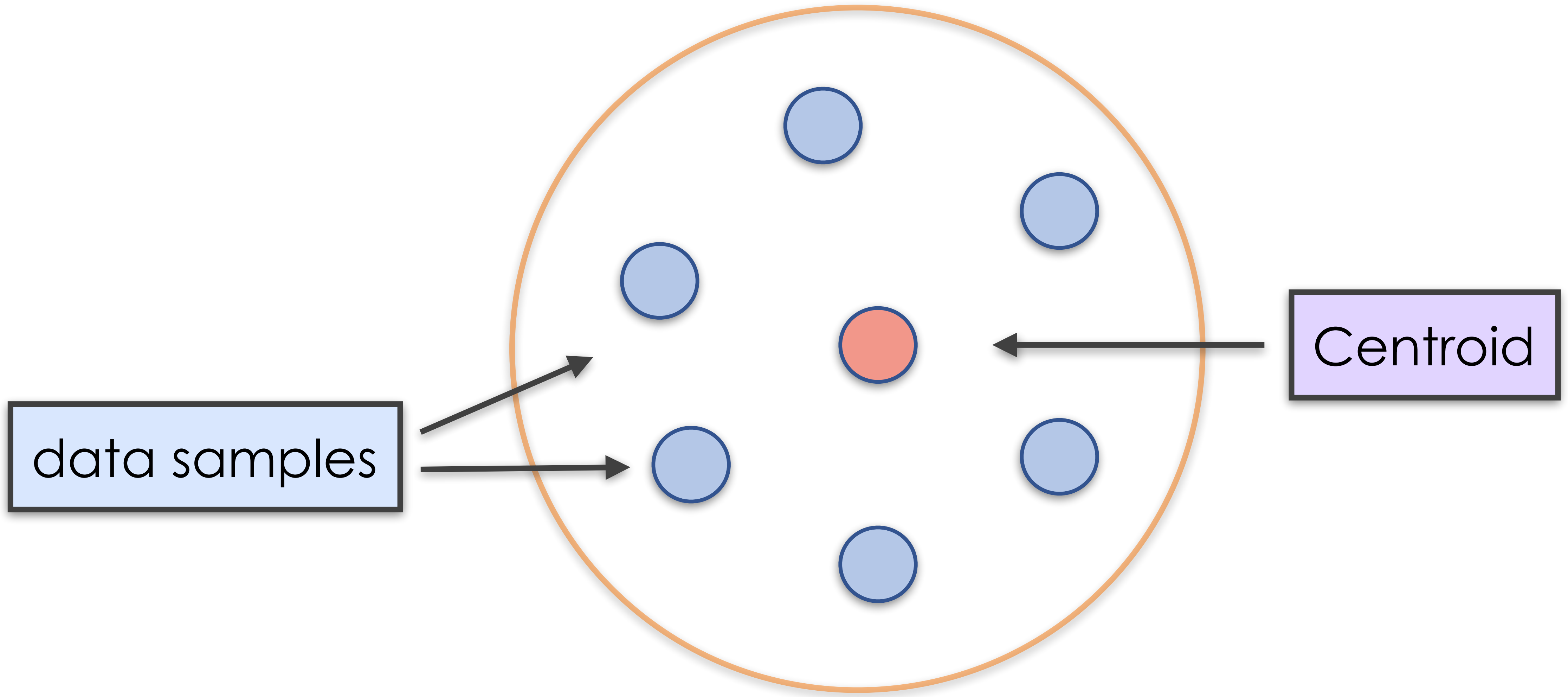


K = 3

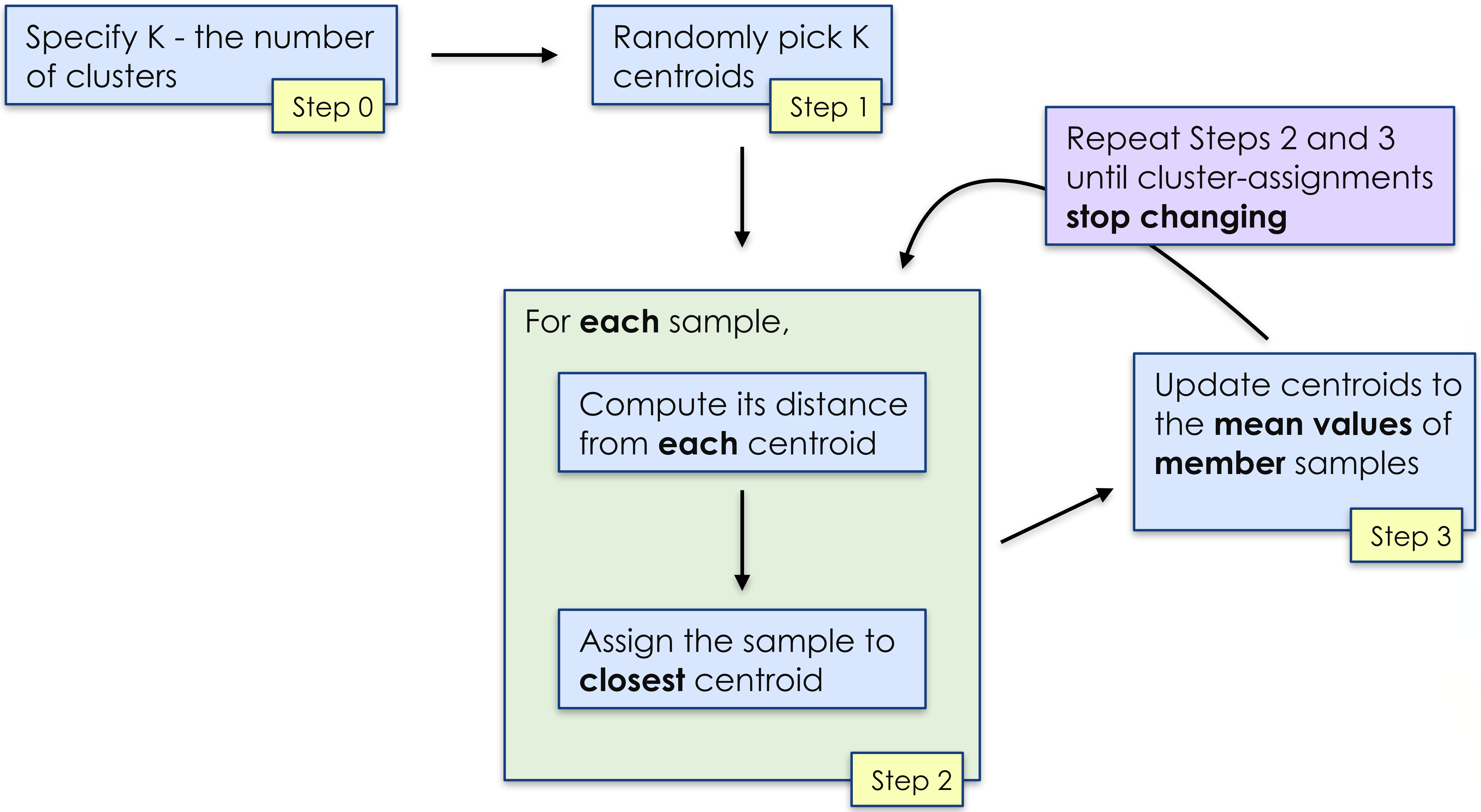
K is a hyper-parameter as it specifies the number of clusters that our data should be grouped into

Centroid

A Centroid is the central location within a cluster



K-Means Clustering Algorithm



Performing K-means Clustering

Using hyper-parameter as K = 2 (number of clusters), we first randomly choose 2 centroids from our data points

	X	Y
Centroid 1	1	1
Centroid 2	2	2

Calculate distance from each sample to the centroids with Euclidean distance formula:

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

Feature 1 (X)	Feature 2 (Y)	Distance to Centroid 1	Distance to Centroid 2	Cluster
1	1	0	1.414213562	1
2	2	1.414213562	0	2
3	3	2.828427125	1.414213562	2
4	4	4.242640687	2.828427125	2
5	5	5.656854249	4.242640687	2

Performing K-means Clustering

Re-compute new positions for our centroids based on member positions

Feature 1 (X)	Feature 2 (Y)	Cluster
1	1	1
2	2	2
3	3	2
4	4	2
5	5	2

	X	Y
Centroid 1	(1)/1 = **1**	(1)/1 = **1**
Centroid 2	(2 + 3 + 4 + 5) / 4 = **3.5**	(2 + 3 + 4 + 5) / 4 = **3.5**

Performing K-means Clustering

Calculate the new centroids by taking the average from the members of the cluster

	X	Y
Centroid 1	1	1
Centroid 2	3.5	3.5

Centroid 2 has an updated position

Calculate distance from each sample to the centroids with Euclidean distance formula:

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

Feature 1 (X)	Feature 2 (Y)	Distance to Centroid 1	Distance to Centroid 2	Cluster
1	1	0	3.535533906	1
2	2	1.414213562	2.121320344	1
3	3	2.828427125	0.707106781	2
4	4	4.242640687	0.707106781	2
5	5	5.656854249	2.121320344	2

Performing K-means Clustering

Re-compute new positions for our centroids based on member positions

Feature 1 (X)	Feature 2 (Y)	Cluster
1	1	1
2	2	1
3	3	2
4	4	2
5	5	2

	X	Y
Centroid 1	(1 + 2) / 2 = **1.5**	(1 + 2) / 2 = **1.5**
Centroid 2	(3 + 4 + 5) / 3 = **4**	(3 + 4 + 5) / 3 = **4**

Performing K-means Clustering

Calculate the new centroids by taking the average from the members of the cluster

	X	Y
Centroid 1	1.5	1.5
Centroid 2	4	4

Both Centroid 1 and Centroid 2 have updated positions

Calculate distance from each sample to the centroids with Euclidean distance formula:

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

Feature 1 (X)	Feature 2 (Y)	Distance to Centroid 1	Distance to Centroid 2	Cluster
1	1	0.707106781	4.242640687	1
2	2	0.707106781	2.828427125	1
3	3	2.121320344	1.414213562	2
4	4	3.535533906	0	2
5	5	4.949747468	1.414213562	2

Performing K-means Clustering

As the cluster assignment remains the same, we stop the iteration with the final centroids and clustering results

Iteration n-1

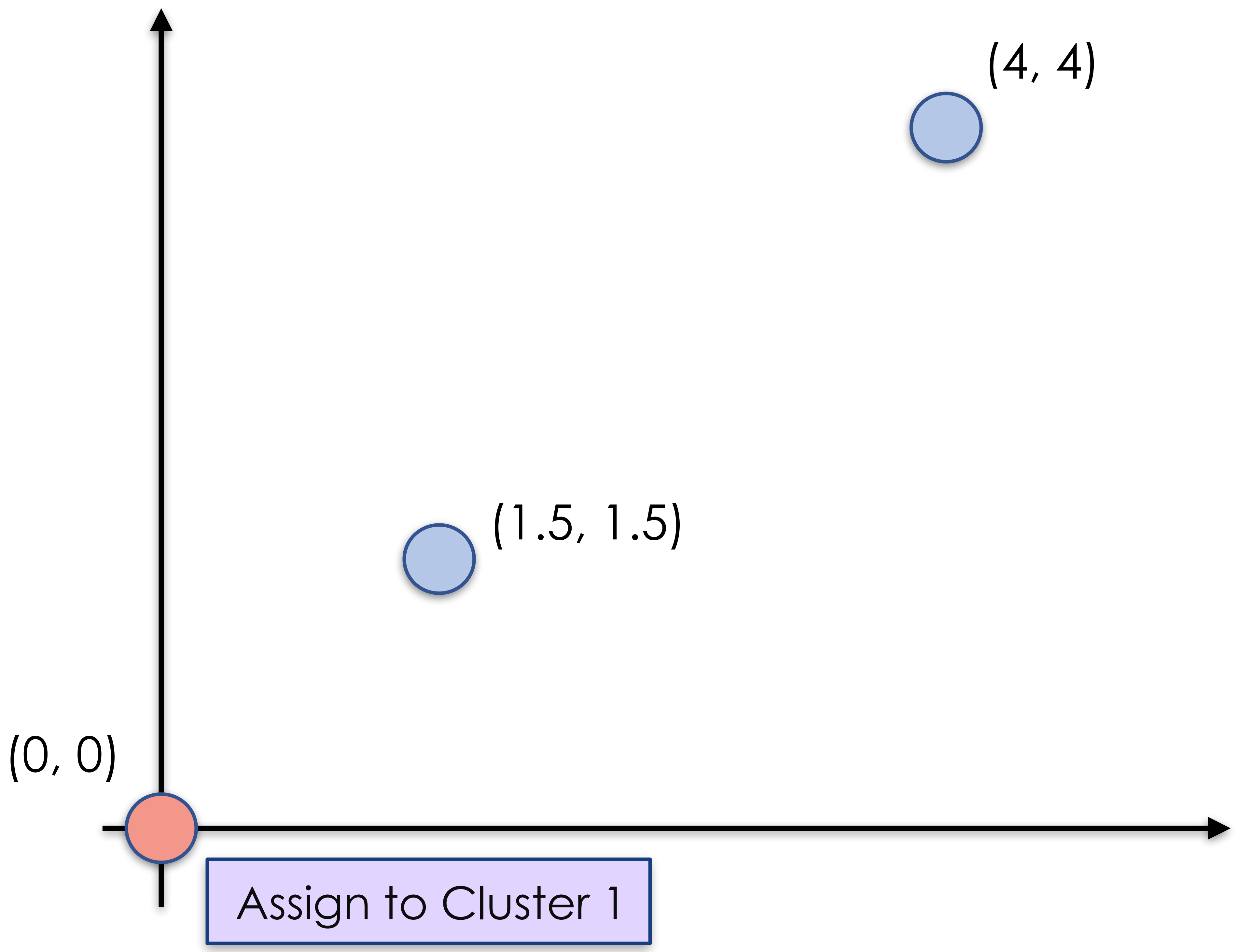
Feature 1 (X)	Feature 2 (Y)	Distance to Centroid 1	Distance to Centroid 2	Cluster
1	1	0	3.535533906	1
2	2	1.414213562	2.121320344	1
3	3	2.828427125	0.707106781	2
4	4	4.242640687	0.707106781	2
5	5	5.656854249	2.121320344	2

No cluster changes between iterations

Iteration n

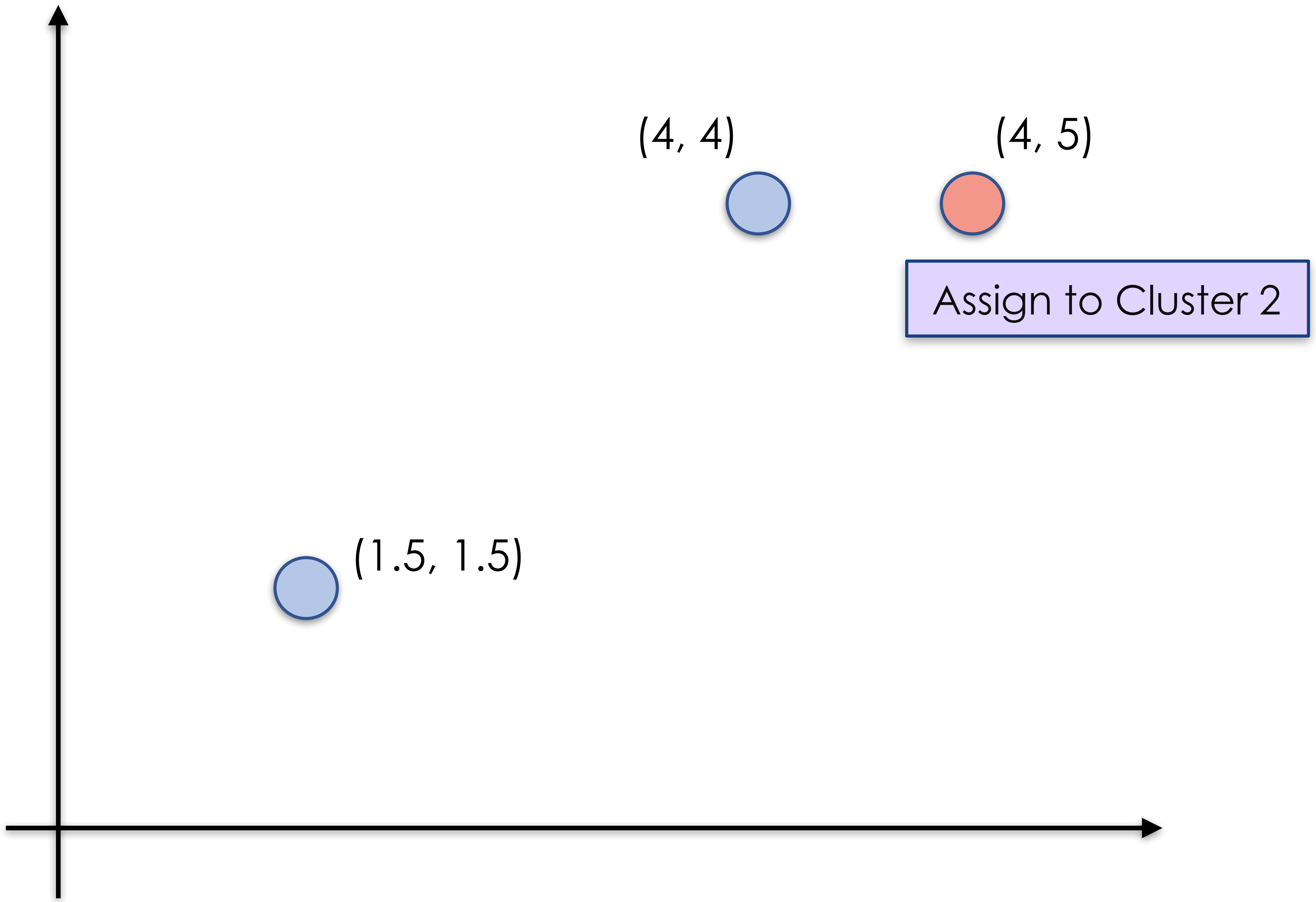
Feature 1 (X)	Feature 2 (Y)	Distance to Centroid 1	Distance to Centroid 2	Cluster
1	1	0.707106781	4.242640687	1
2	2	0.707106781	2.828427125	1
3	3	2.121320344	1.414213562	2
4	4	3.535533906	0	2
5	5	4.949747468	1.414213562	2

Assign new sample to cluster



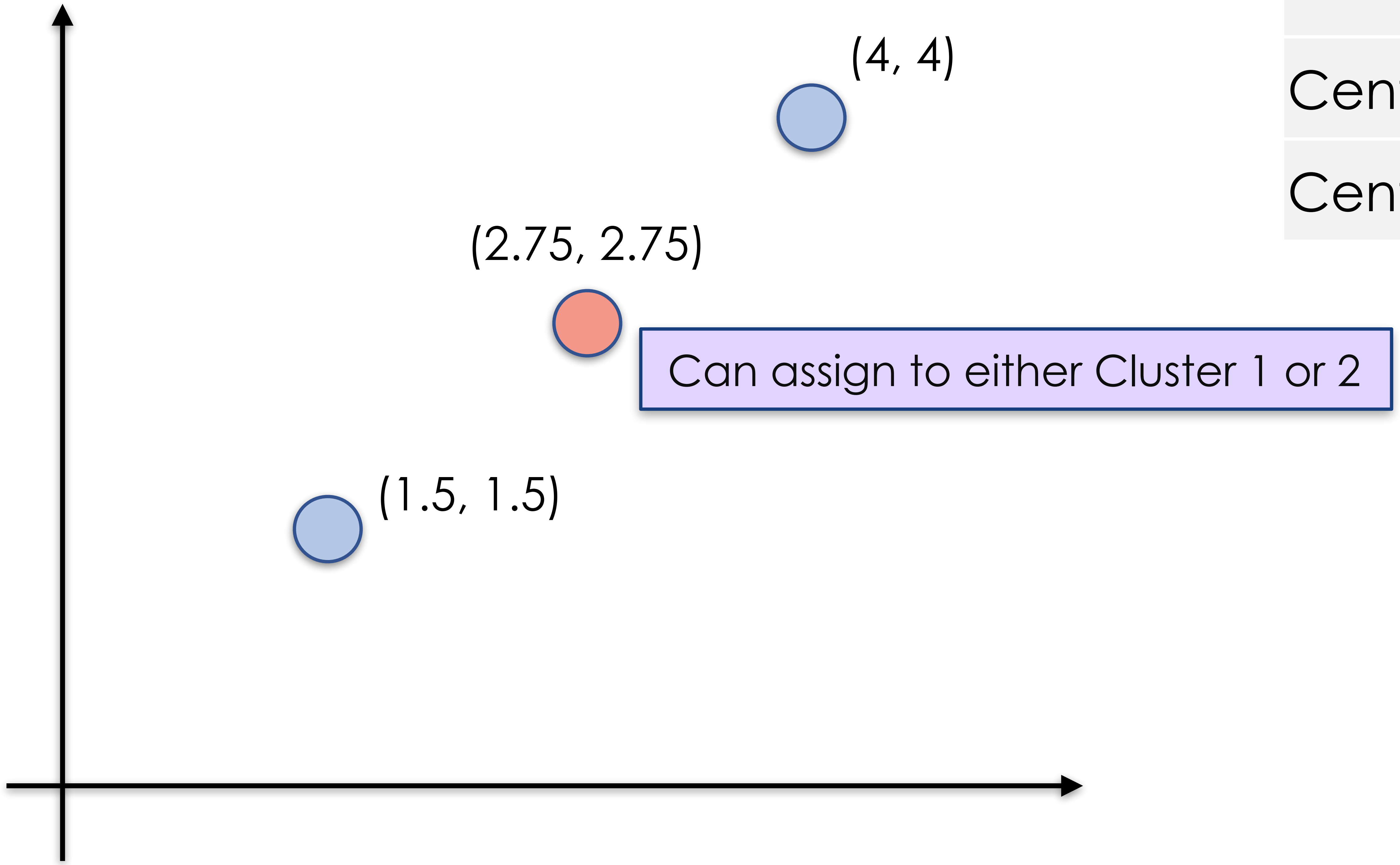
	X	Y
Centroid 1	1.5	1.5
Centroid 2	4	4

Assign new sample to cluster



	X	Y
Centroid 1	1.5	1.5
Centroid 2	4	4

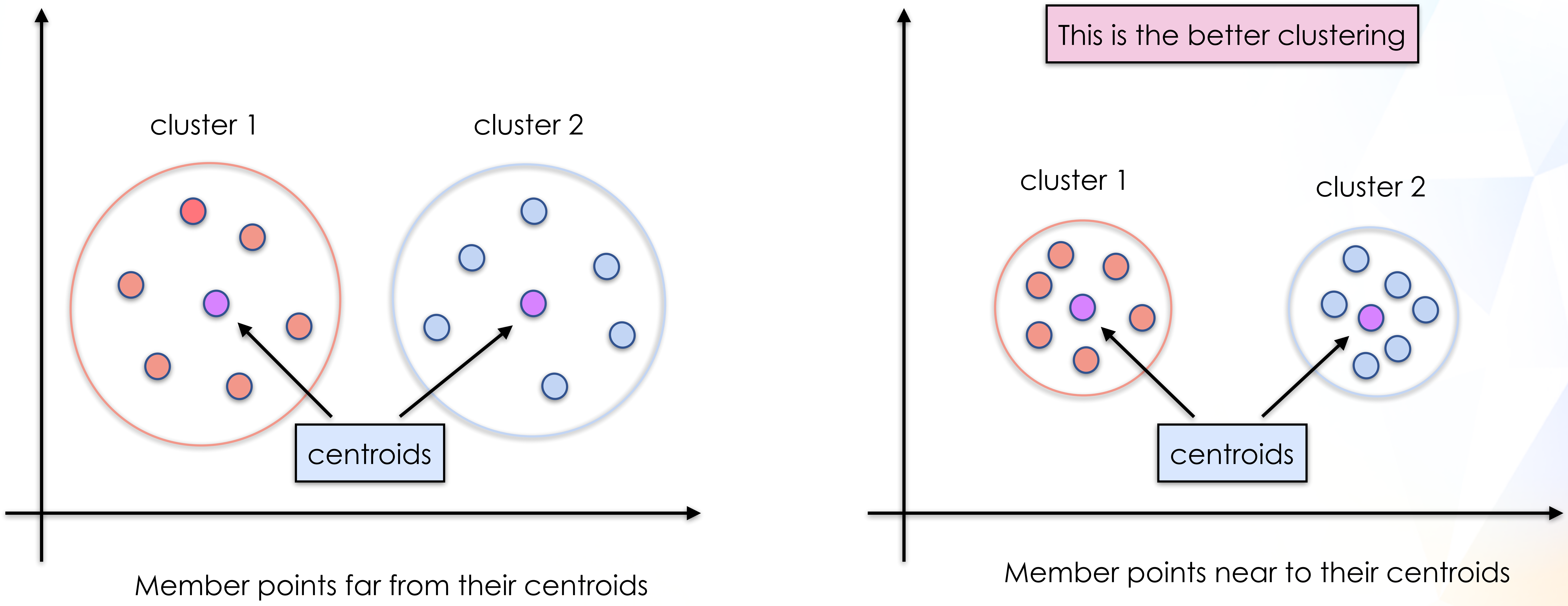
Assign new sample to cluster



	X	Y
Centroid 1	1.5	1.5
Centroid 2	4	4

Goodness-of-Fit

- Compute how close our data points are with respect to the centroid of their assigned clusters



WCSS

Within Cluster Sum of Squares (WCSS) measures the goodness-of-fit of a centroid-based clustering

no. of clusters

no. of data points in a cluster

one of the centroids

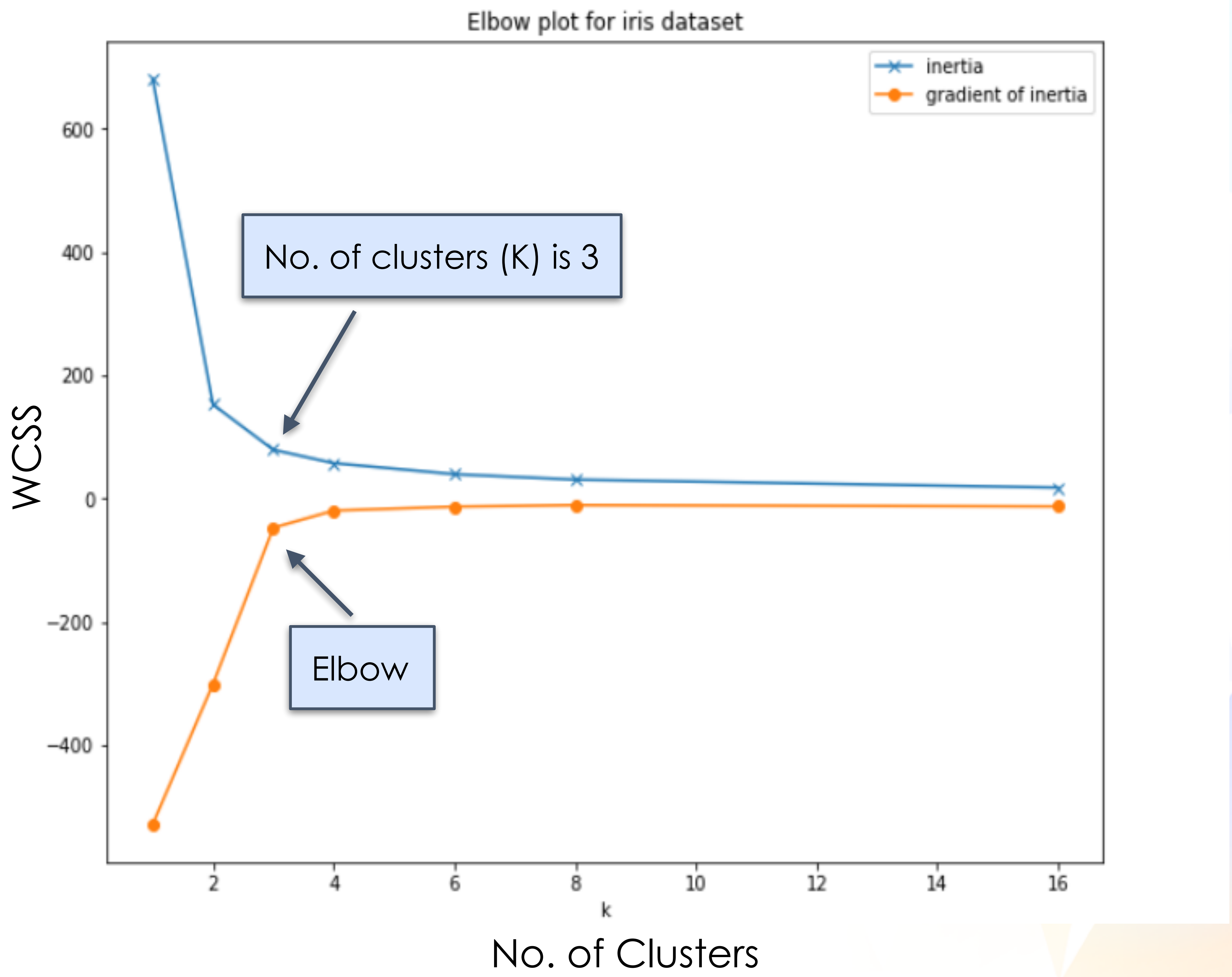
$$WCSS = \sum_{j=1}^{m} \sum_{i=1}^{n} (x_i - c_j)^2$$

one of the data points

WCSS is defined as the sum of squared distance between each member of a cluster and its centroid

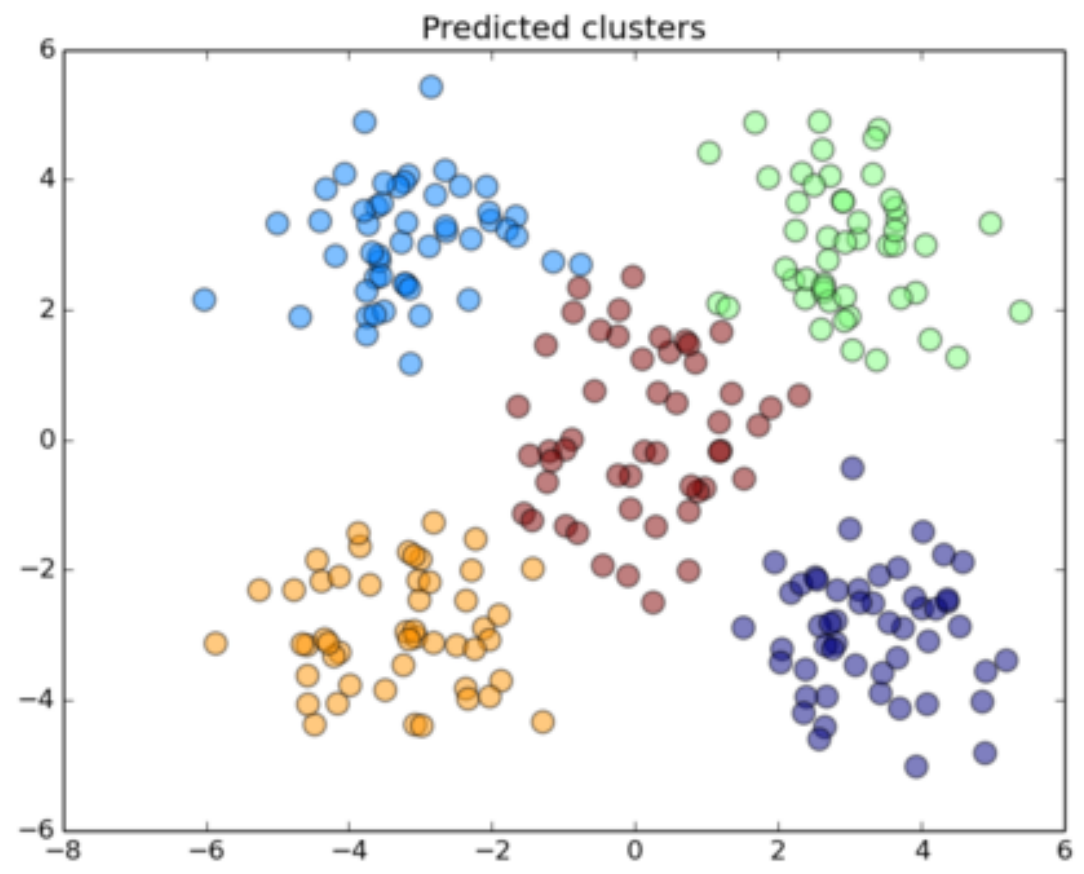
Selecting K for K-Means

The Elbow is when the gradient stops dropping drastically

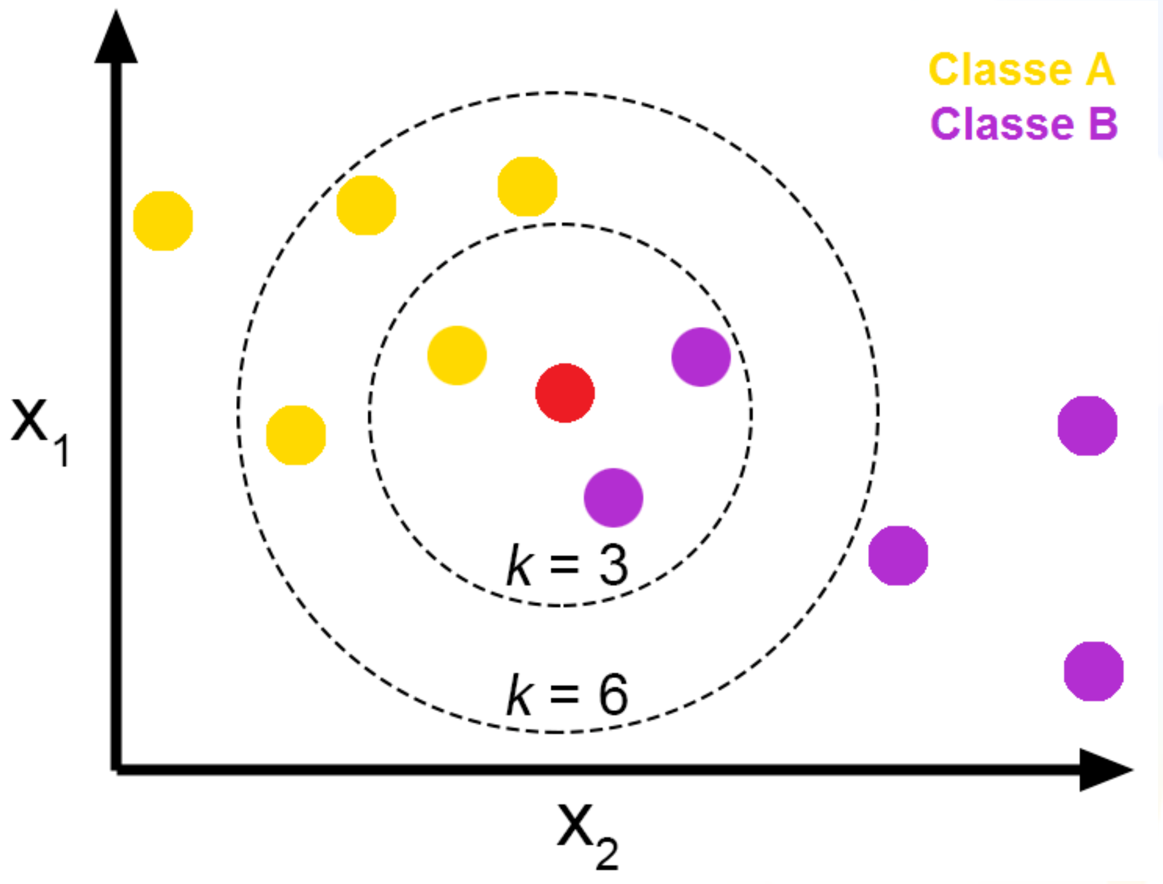


K-Means vs KNN

- The **K** in **K**-Means refers to the number of clusters that our data points can be grouped into



- The **K** in **K**NN refers to the number of neighbors to consider when classifying a new data point



K-Means using sklearn

- Let’s perform a K-Means on our Iris dataset

	A	B	C	D	E	F
1	Id	SepalLength(	SepalWidthC	PetalLength(	PetalWidthC	Species
2	1	5.1	3.5	1.4	0.2	Iris-setosa
3	2	4.9	3	1.4	0.2	Iris-setosa
4	3	4.7	3.2	1.3	0.2	Iris-setosa
5	4	4.6	3.1	1.5	0.2	Iris-setosa
6	5	5	3.6	1.4	0.2	Iris-setosa
7	6	5.4	3.9	1.7	0.4	Iris-setosa
8	7	4.6	3.4	1.4	0.3	Iris-setosa
9	8	5	3.4	1.5	0.2	Iris-setosa
10	9	4.4	2.9	1.4	0.2	Iris-setosa
11	10	4.9	3.1	1.5	0.1	Iris-setosa
12	11	5.4	3.7	1.5	0.2	Iris-setosa
13	12	4.8	3.4	1.6	0.2	Iris-setosa
14	13	4.8	3	1.4	0.1	Iris-setosa
15	14	4.3	3	1.1	0.1	Iris-setosa
16	15	5.8	4	1.2	0.2	Iris-setosa
17	16	5.7	4.4	1.5	0.4	Iris-setosa
18	17	5.4	3.9	1.3	0.4	Iris-setosa
19	18	5.1	3.5	1.4	0.3	Iris-setosa
20	19	5.7	3.8	1.7	0.3	Iris-setosa
21	20	5.1	3.8	1.5	0.3	Iris-setosa
22	21	5.4	3.4	1.7	0.2	Iris-setosa
23	22	5.1	3.7	1.5	0.4	Iris-setosa
24	23	4.6	3.6	1	0.2	Iris-setosa
25	24	5.1	3.3	1.7	0.5	Iris-setosa
26	25	4.8	3.4	1.9	0.2	Iris-setosa
27	26	5	3	1.6	0.2	Iris-setosa
28	27	5	3.4	1.6	0.4	Iris-setosa
29	28	5.2	3.5	1.5	0.2	Iris-setosa

K-Means using sklearn

- Exclude data from the first and last column
- Use ".values" to convert to NumPy array

```python
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd
from sklearn.cluster import KMeans

data = pd.read_csv('iris.csv')
X = data.iloc[:, 1:-1].values
```

5.1	3.5	1.4	0.2
4.9	3	1.4	0.2
4.7	3.2	1.3	0.2
4.6	3.1	1.5	0.2
5	3.6	1.4	0.2
5.4	3.9	1.7	0.4
4.6	3.4	1.4	0.3
5	3.4	1.5	0.2
4.4	2.9	1.4	0.2
4.9	3.1	1.5	0.1
5.4	3.7	1.5	0.2
4.8	3.4	1.6	0.2
4.8	3	1.4	0.1
4.3	3	1.1	0.1
5.8	4	1.2	0.2
5.7	4.4	1.5	0.4
5.4	3.9	1.3	0.4
5.1	3.5	1.4	0.3
5.7	3.8	1.7	0.3
5.1	3.8	1.5	0.3
5.4	3.4	1.7	0.2
5.1	3.7	1.5	0.4
4.6	3.6	1	0.2
5.1	3.3	1.7	0.5
4.8	3.4	1.9	0.2
5	3	1.6	0.2
5	3.4	1.6	0.4

K-Means using sklearn

- Specify the required no. of clusters
- K-Means assigns each sample (a row of data) to a cluster

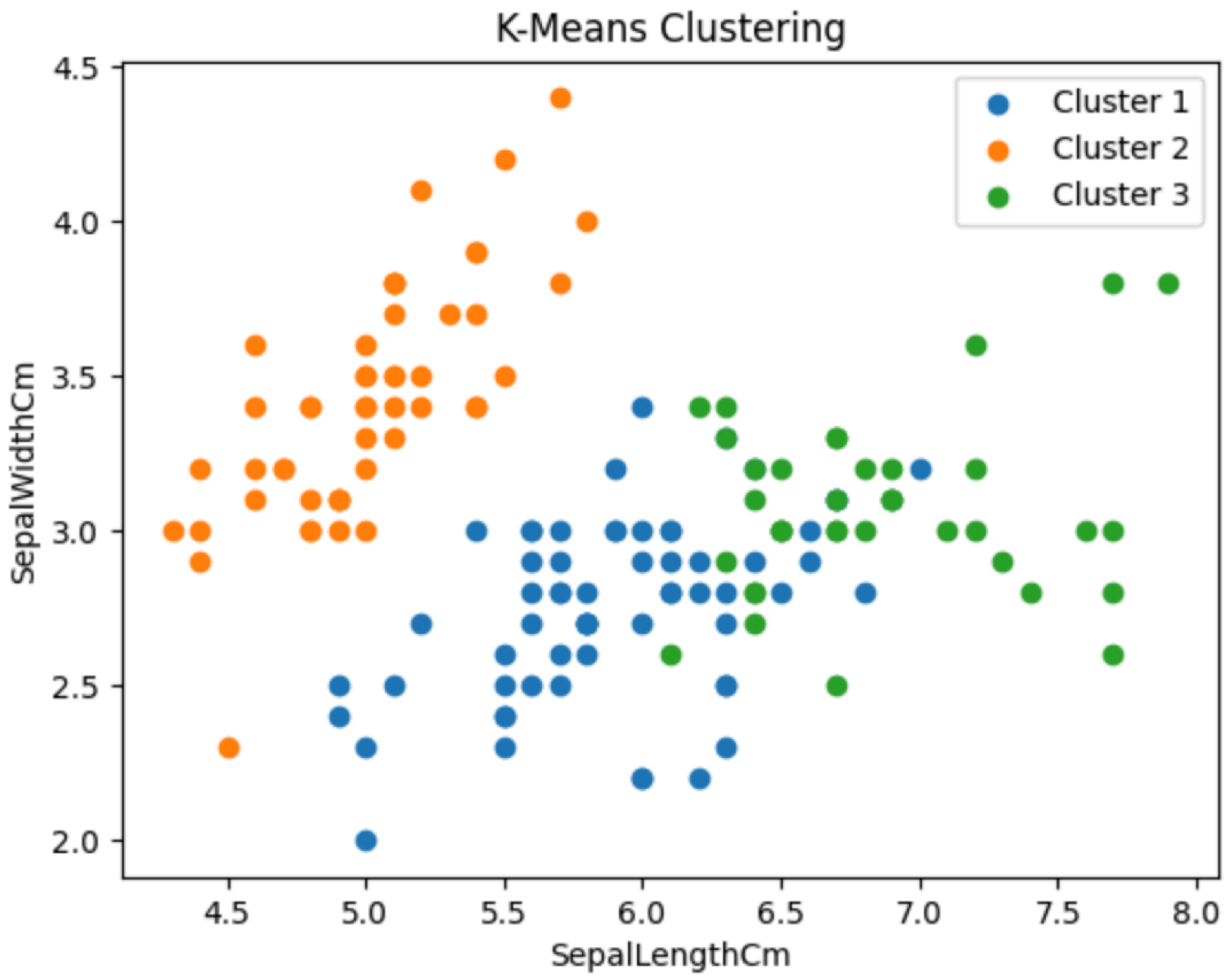
```python
# apply kmeans to the dataset
kmeans = KMeans(n_clusters=3)
kmeans.fit(X)
row_cluster_map = kmeans.predict(X)
print(row_cluster_map)
```

```
[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 0 0 2 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
 0 0 0 2 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 2 0 2 2 2 2 0 2 2 2 2
 2 2 0 0 2 2 2 2 0 2 0 2 0 2 2 0 0 2 2 2 2 2 0 2 2 2 2 0 2 2 2 0 2 2 2 0 2
 2 0]
```

K-Means using sklearn

- For each cluster, retrieve samples that have been assigned to that cluster
- Plot each sample based on its first 2 features

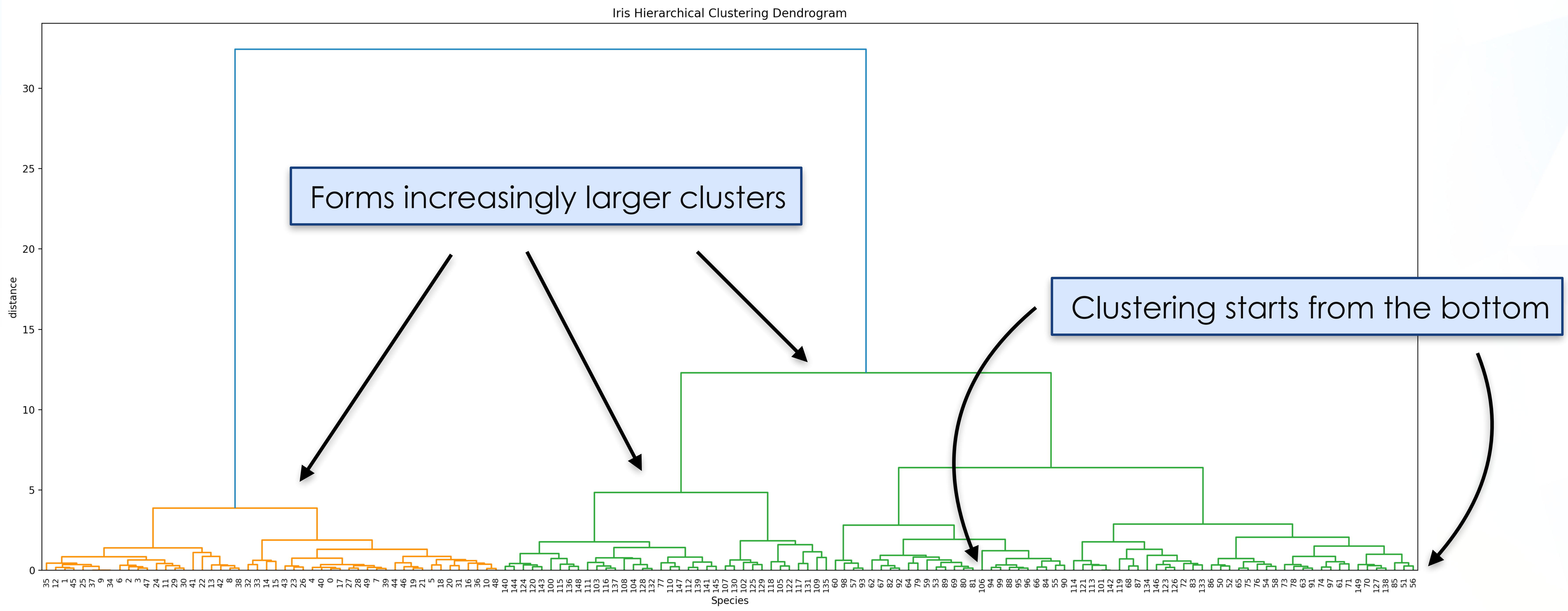
```
# plot clustered data
for i in np.unique(row_cluster_map):
    plt.scatter(x=X[row_cluster_map == i, 0],
                y=X[row_cluster_map == i, 1],
                label='Cluster ' + str(i + 1))

plt.title('K-Means Clustering')
plt.xlabel(data.columns[1])
plt.ylabel(data.columns[2])
plt.legend()
plt.show()
```



Hierarchical Clustering

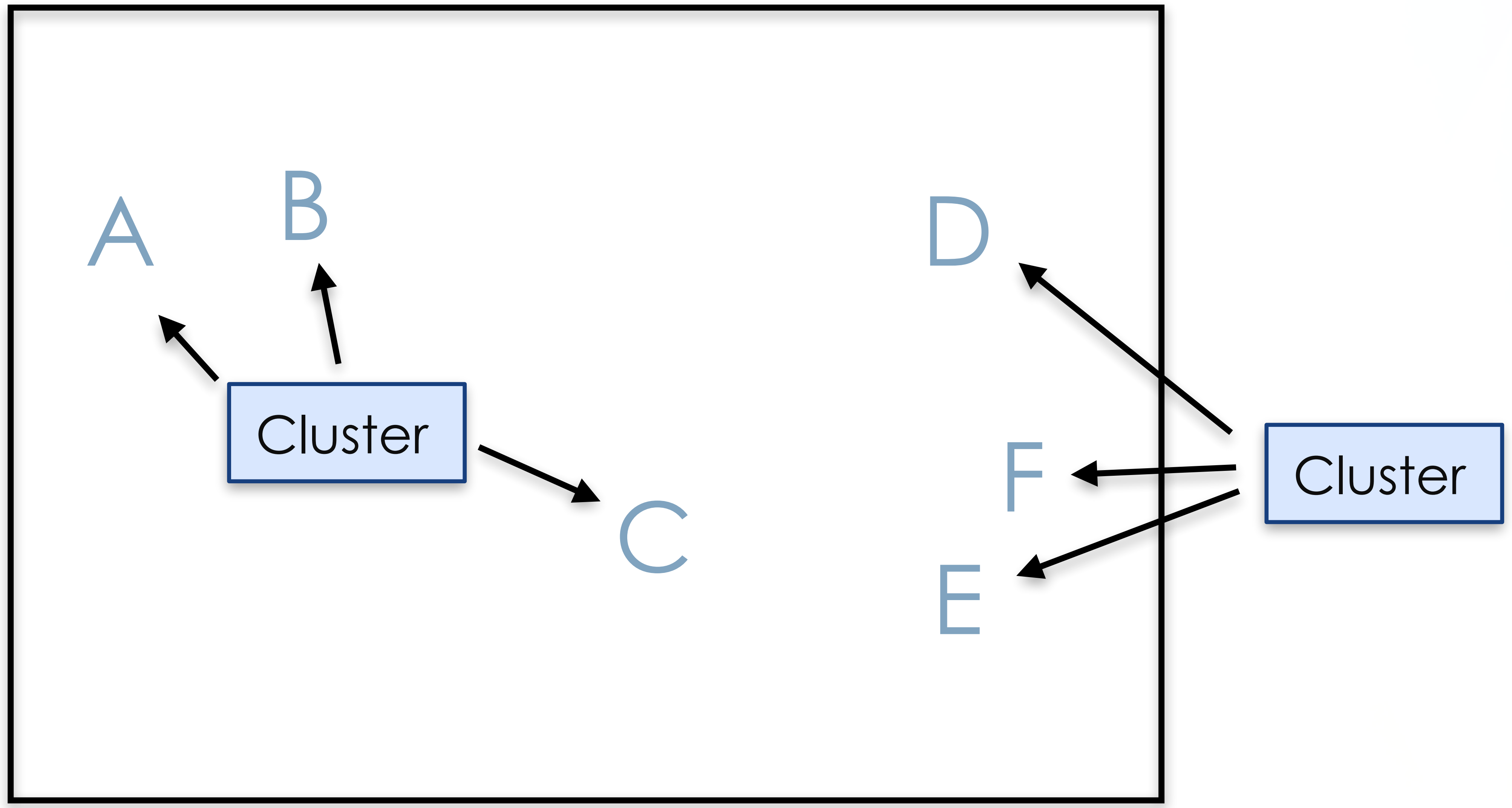
Hierarchical Clustering

Hierarchical Clustering is a bottom-up or agglomerative clustering technique



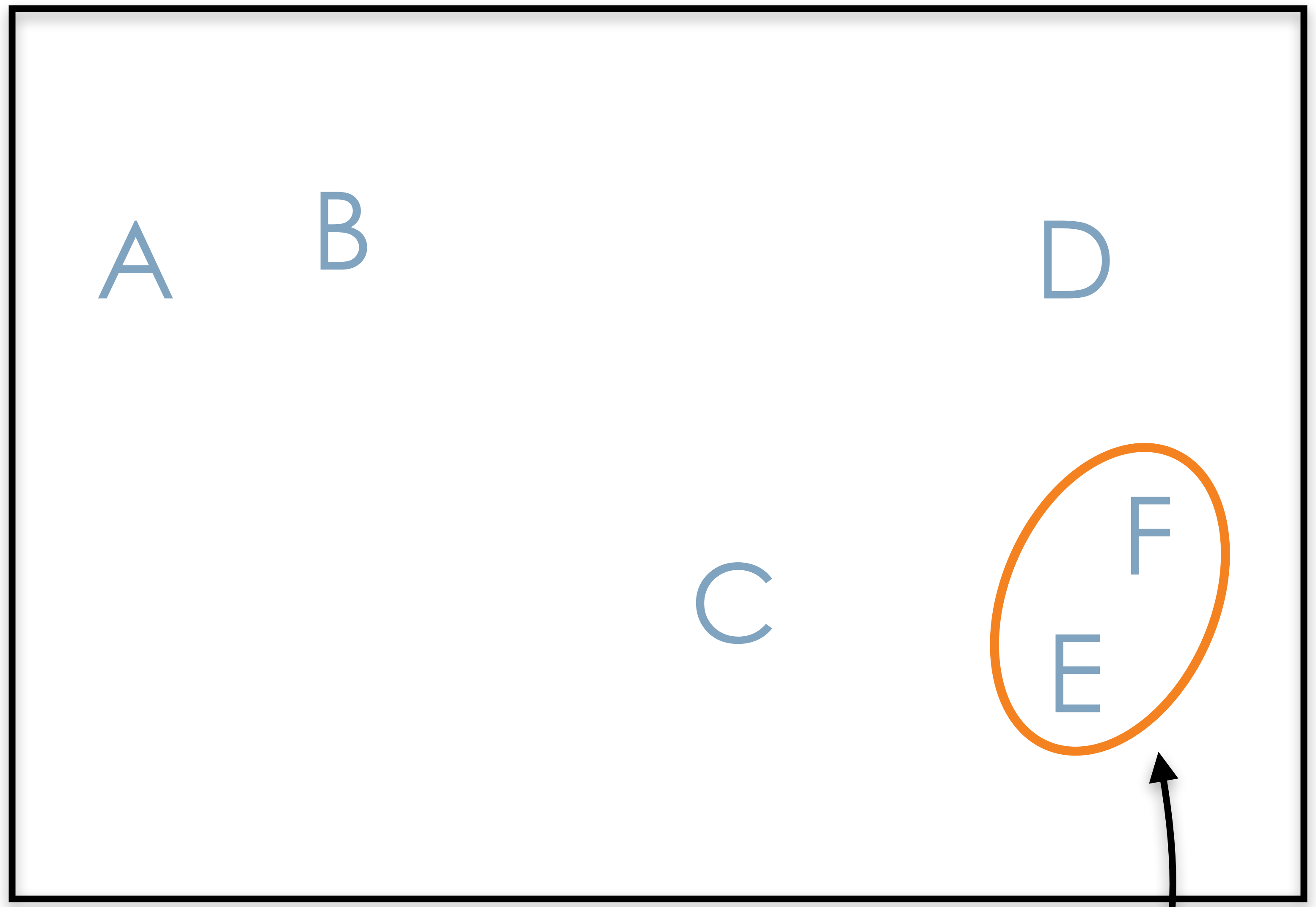
Hierarchical Clustering

Each point starts off as its own cluster



Hierarchical Clustering

Merge two clusters that are closest to each other

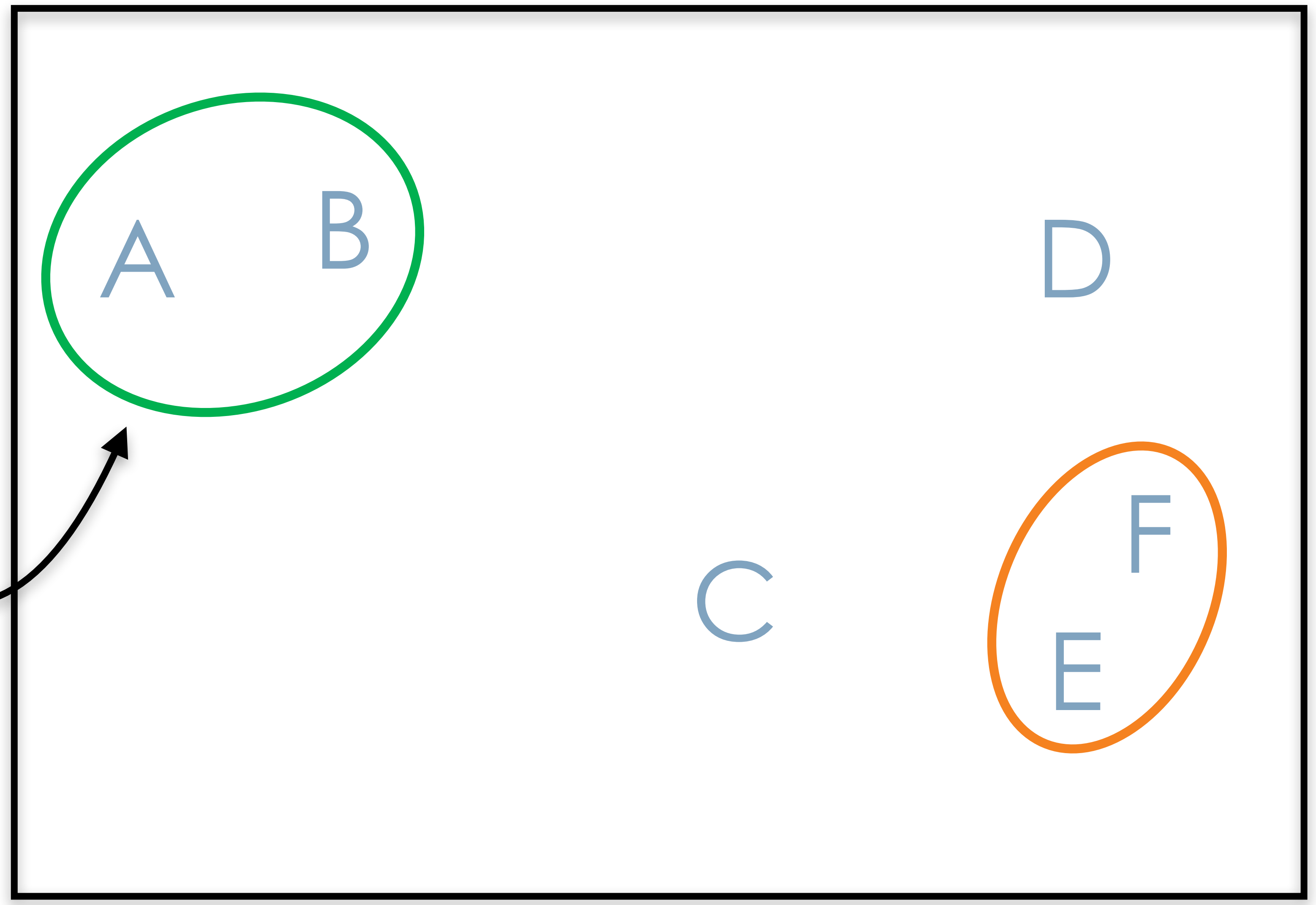


{E} and {F} merge to form a new cluster {E, F}

Hierarchical Clustering

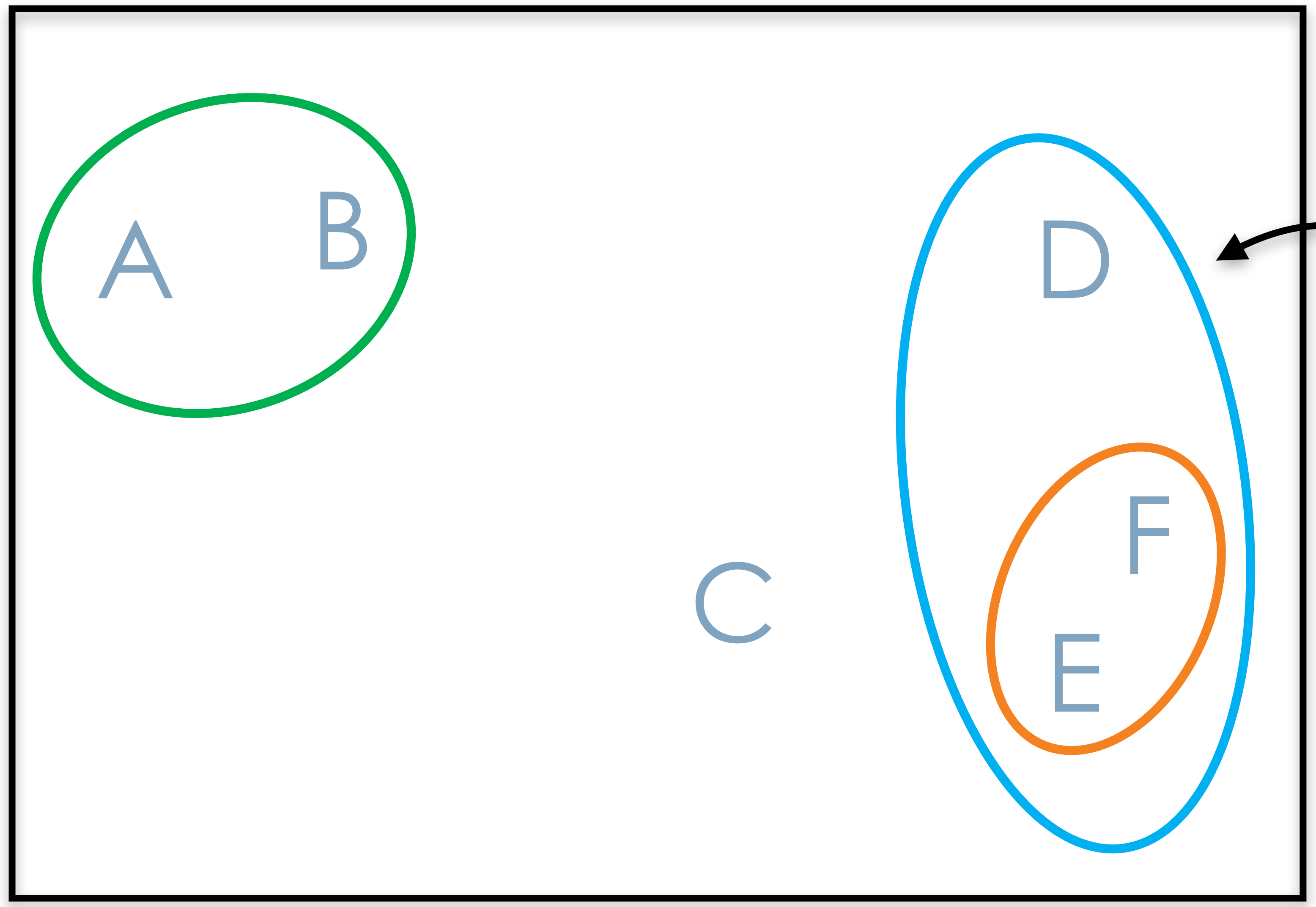
Merge two clusters that are closest to each other

{A} and {B} merge to form
a new cluster {A, B}



Hierarchical Clustering

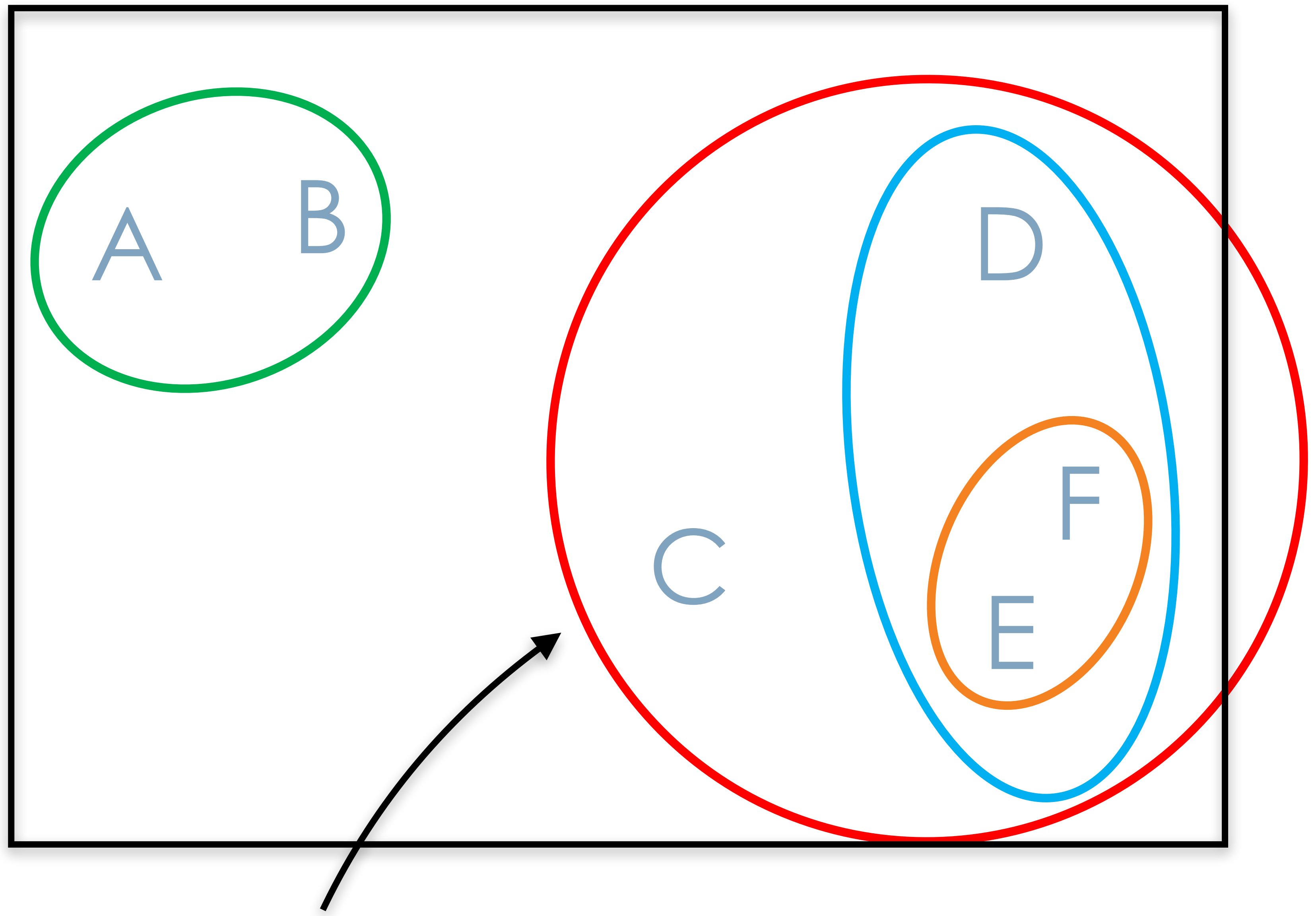
Merge two clusters that are closest to each other



{D} and {E, F} merge to form a new cluster {D, E, F}

Hierarchical Clustering

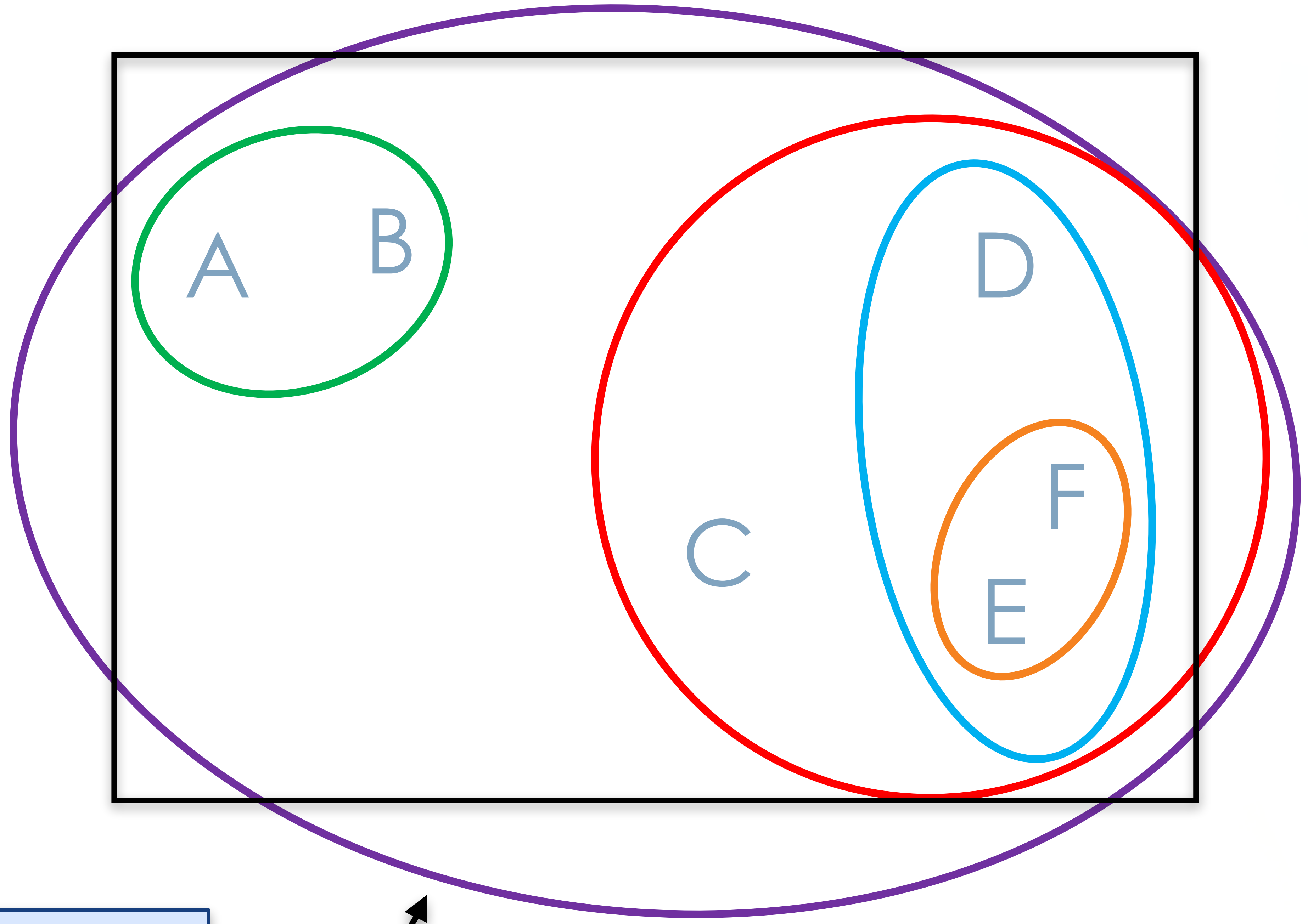
Merge two clusters that are closest to each other



{C} and {D, E, F} merge to form a new cluster {C, D, E, F}

Hierarchical Clustering

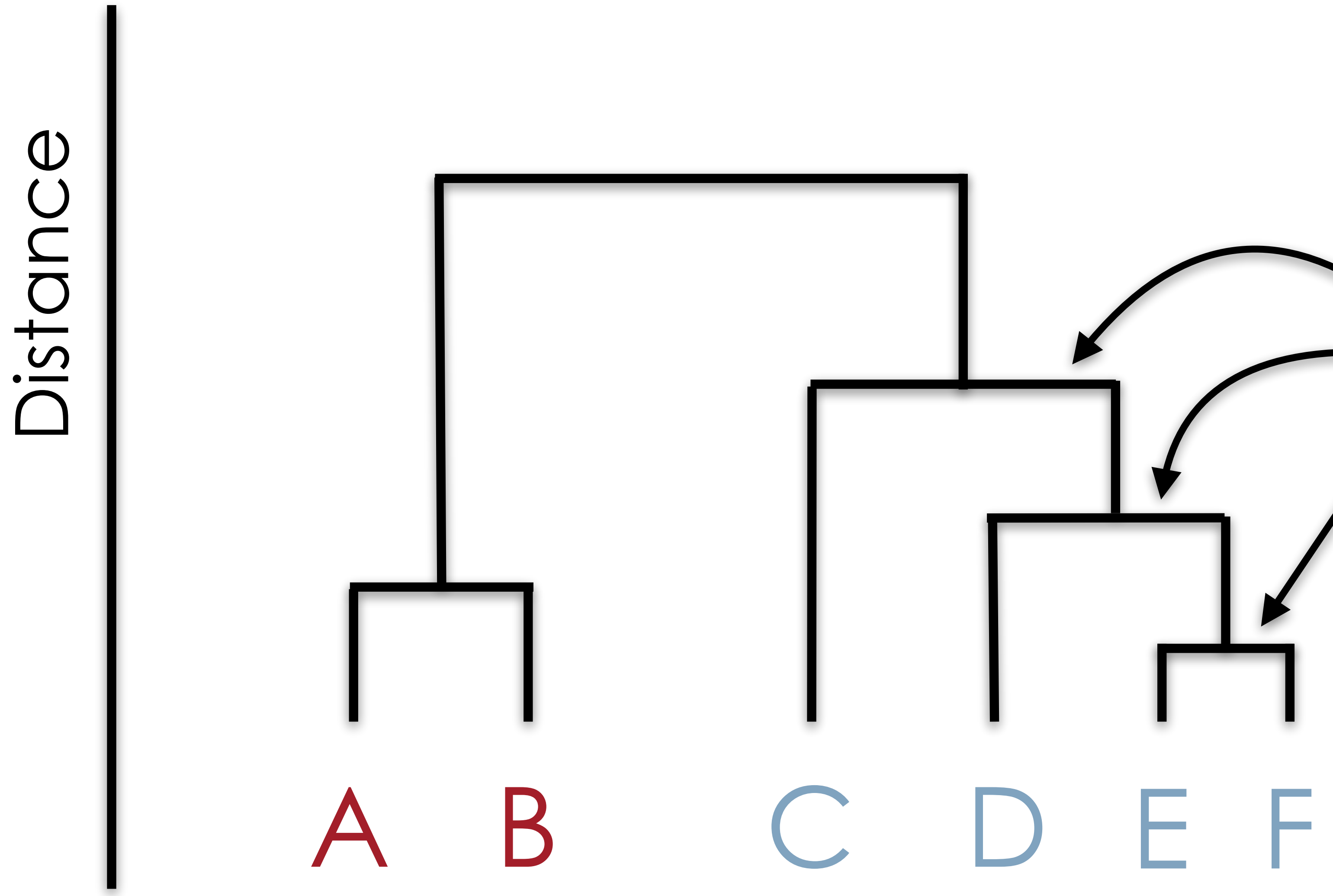
Merge the remaining two clusters



{A, B} and {C, D, E, F} merge to form a new cluster {A, B, C, D, E, F}

Dendrogram

A Dendrogram is a visual representation of the outcome of a Hierarchical Clustering

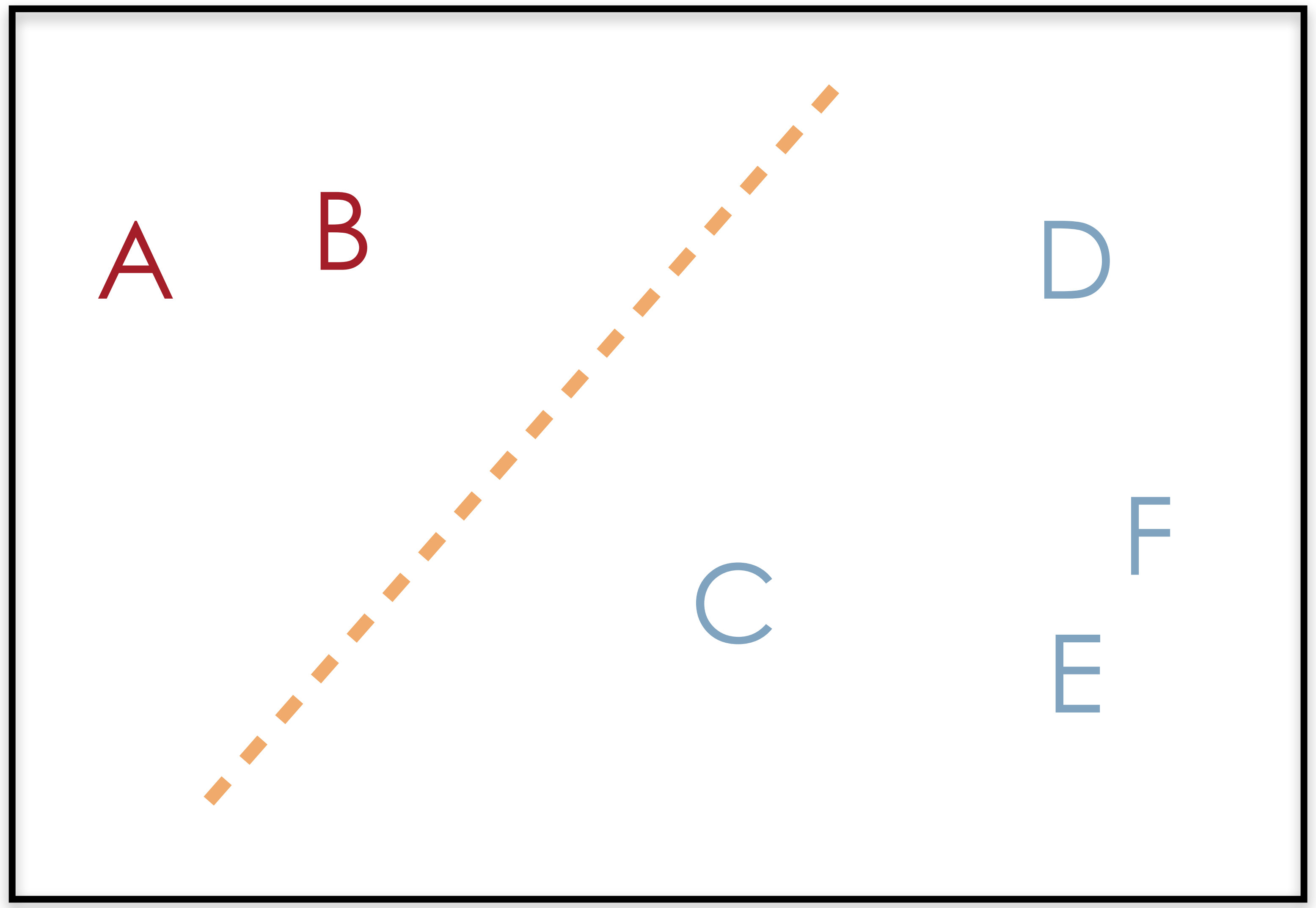


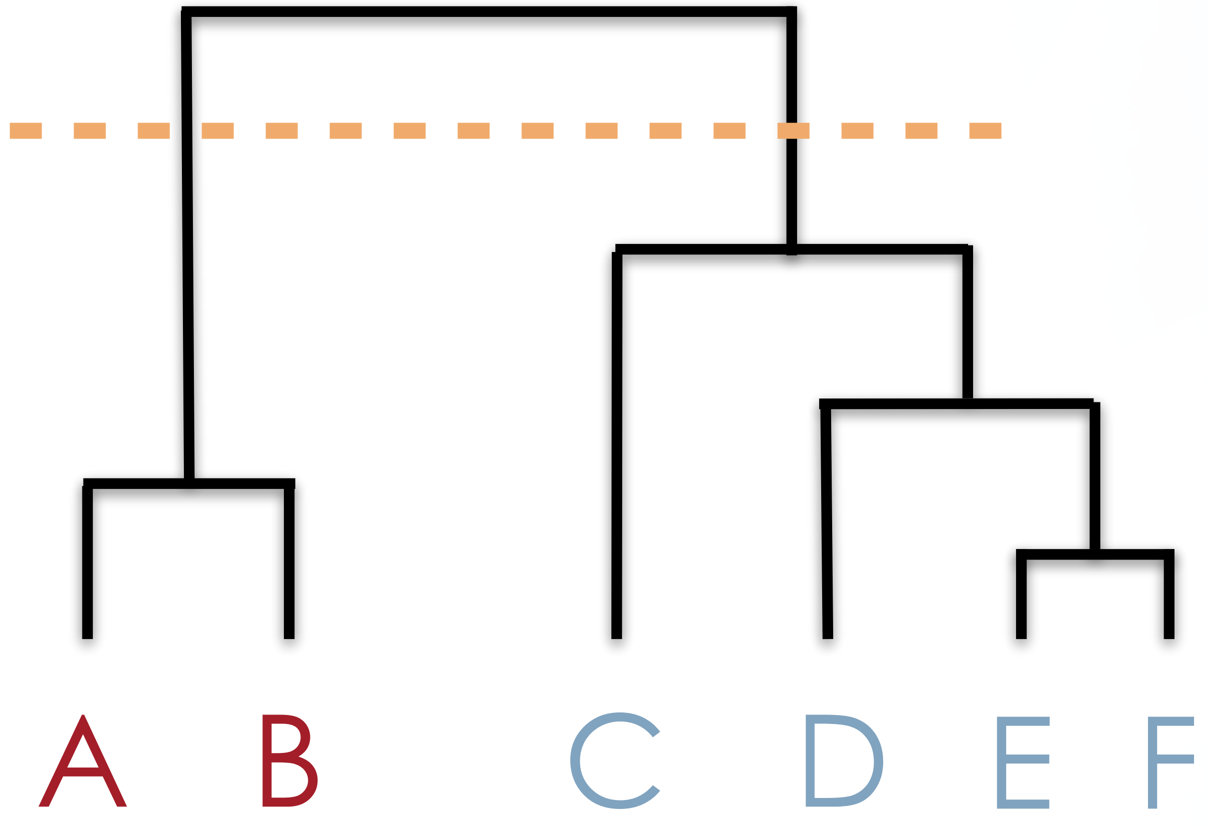
The heights of the pairing lines indicates the distance and the order which the clusters were merged

Dendrogram

Intersecting Line

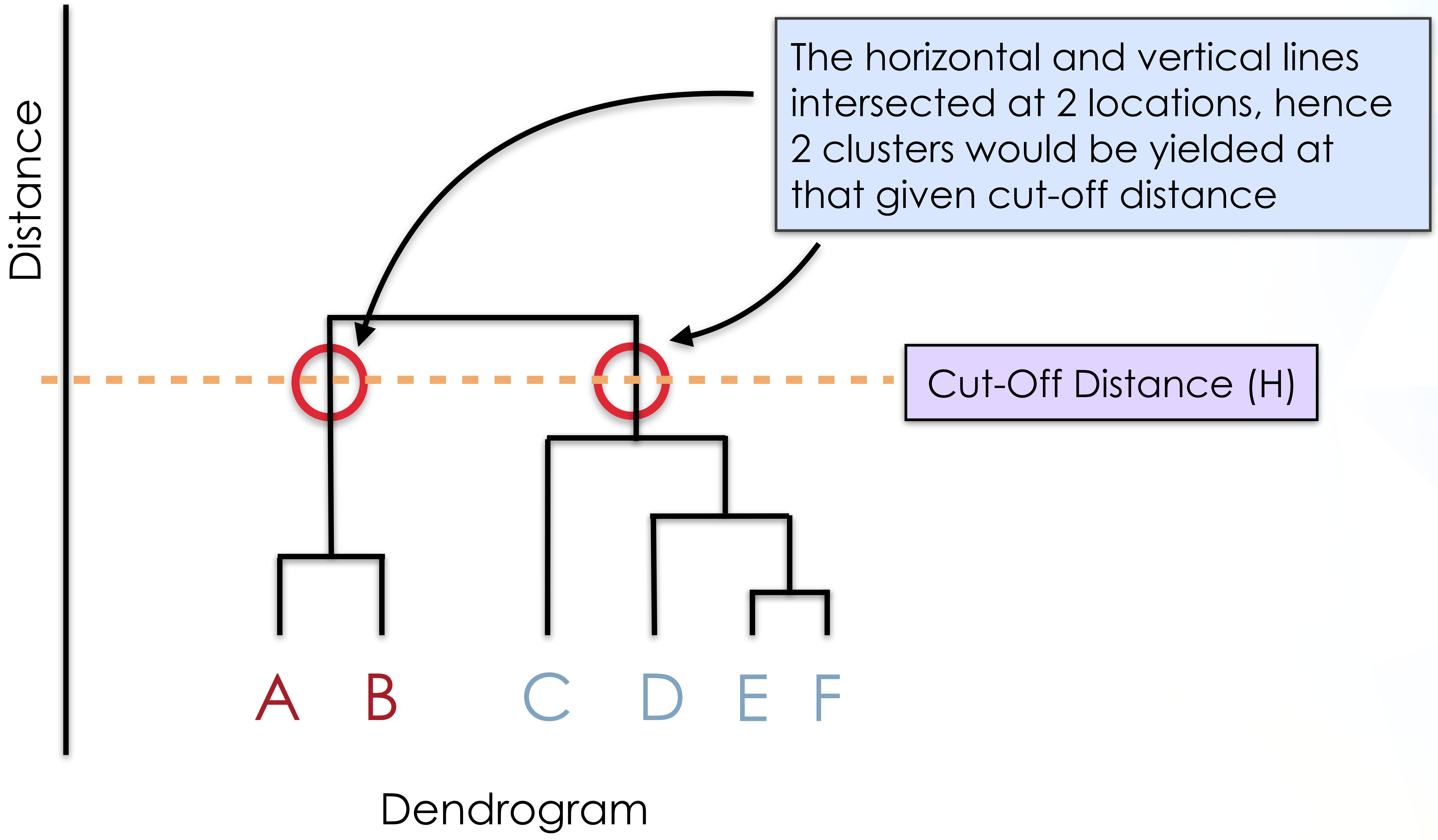
The horizontal line through a dendrogram tells us how many clusters we could get at a given cut-off merging distance





Dendrogram

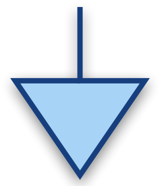
Number of Intersections



Hierarchical Clustering using sklearn

- Assign each sample to a cluster using Hierarchical Clustering

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

data = pd.read_csv('iris.csv')
X = data.iloc[:, 1:-1].values

# hierarchical clustering of iris data
model = AgglomerativeClustering(linkage="ward", n_clusters=3)
row_cluster_map = model.fit_predict(X)
print(row_cluster_map)
```

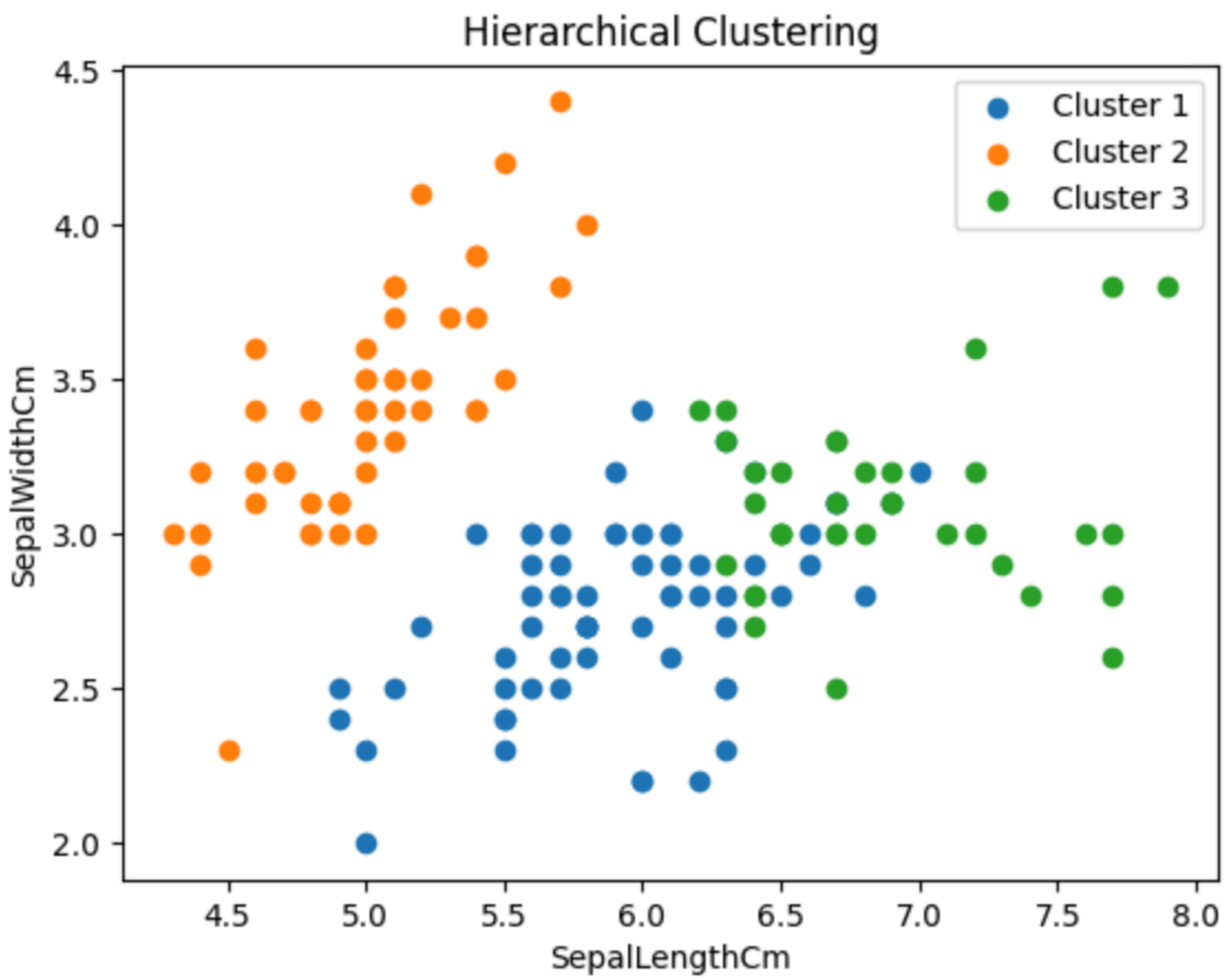


```
[1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
 1 1 1 1 1 1 1 1 1 1 1 1 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
 0 0 0 2 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 2 0 2 2 2 2 0 2 2 2 2
 2 2 0 0 2 2 2 2 0 2 0 2 0 2 2 0 0 2 2 2 2 2 0 0 2 2 2 0 2 2 2 0 2 2 2 0 2
 2 0]
```

Hierarchical Clustering using sklearn

- Plotting samples with respect to their assigned cluster

```
# plot 2d graphs
for i in np.unique(row_cluster_map):
    plt.scatter(x=X[row_cluster_map == i, 0],
                y=X[row_cluster_map == i, 1],
                label='Cluster ' + str(i + 1))

plt.title('Hierarchical Clustering')
plt.xlabel(data.columns[1])
plt.ylabel(data.columns[2])
plt.legend()
plt.show()
```

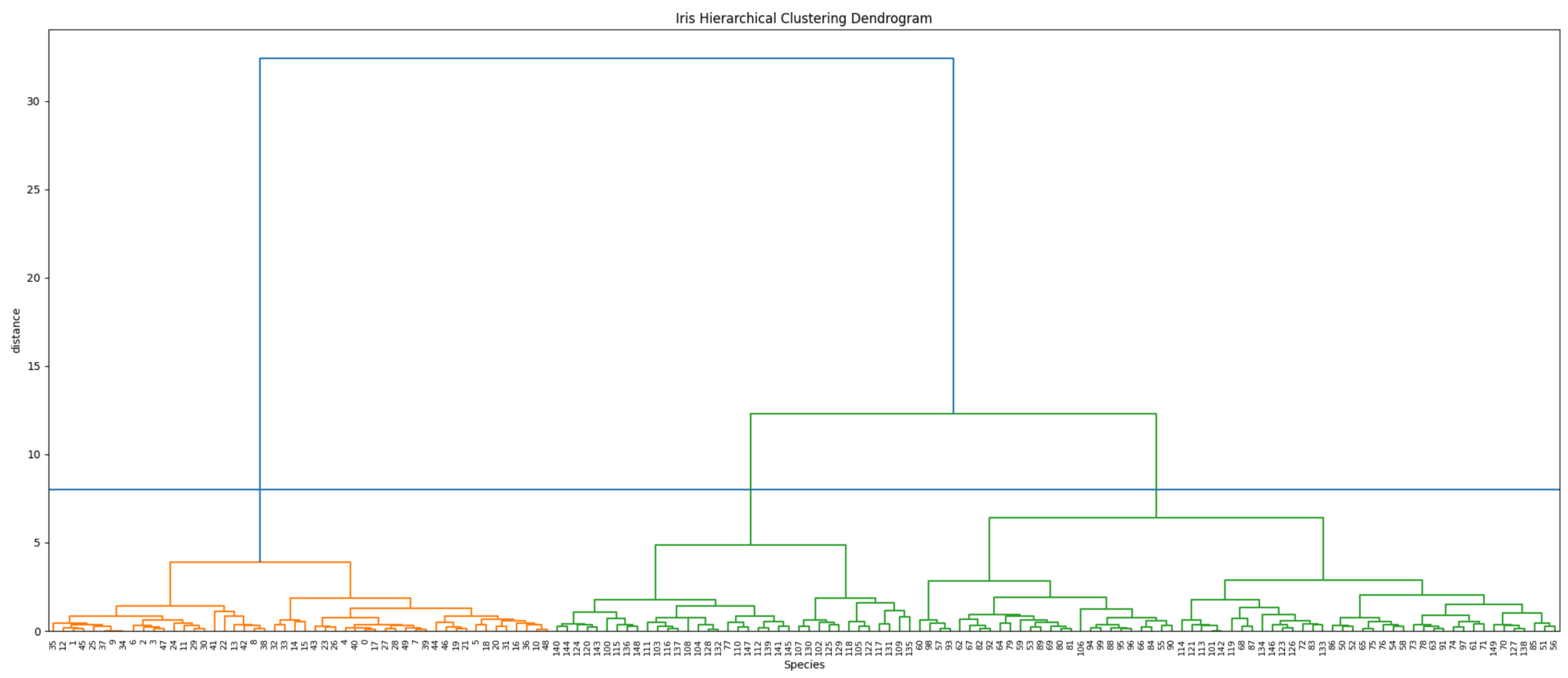


Hierarchical Clustering using sklearn

- Generating a Dendrogram from the Iris samples

```
# draw a dendrogram
plt.figure(figsize=(25, 10))
plt.title('Iris Hierarchical Clustering Dendrogram')
plt.xlabel('Species')
plt.ylabel('distance')

dendrogram(
    linkage(X, 'ward'),  # generate the linkage matrix
    leaf_font_size=8     # font size for the x axis labels
)

plt.axhline(y=8)
plt.show()
```

Algorithm to compute distance between samples

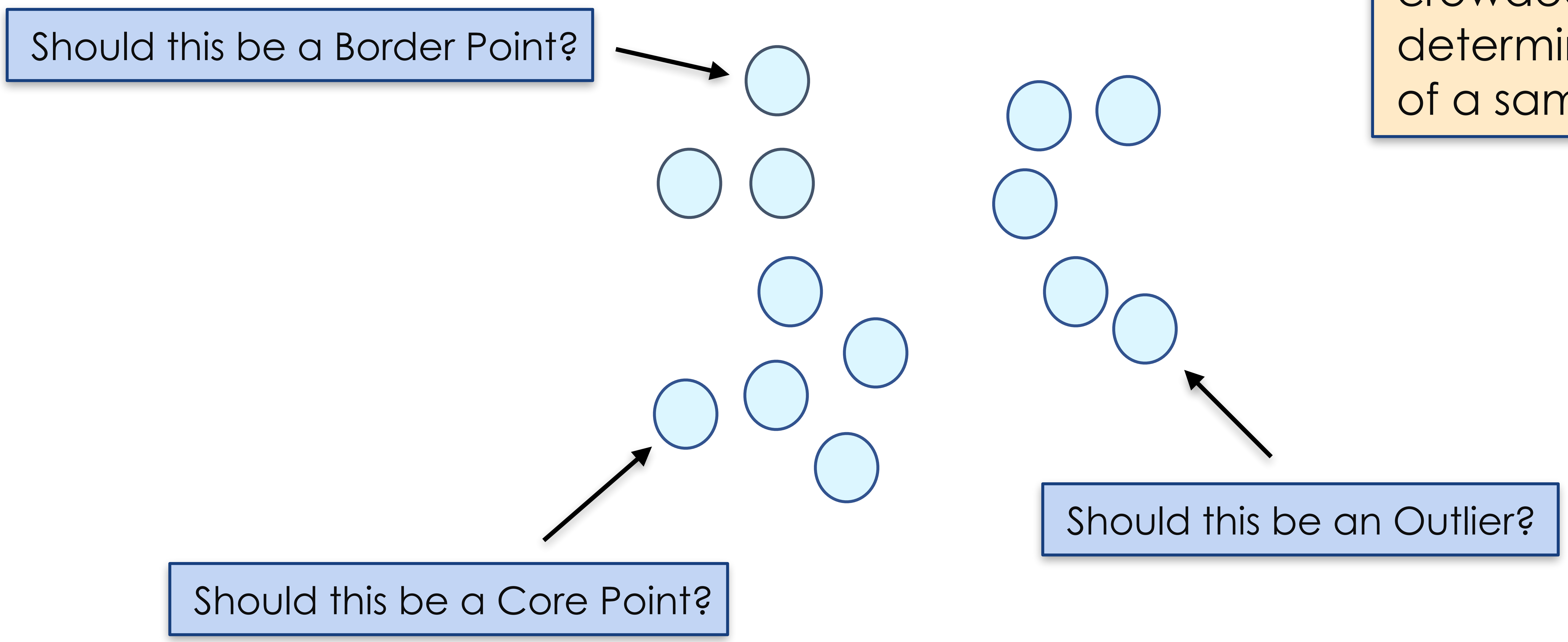


DBSCAN

DBSCAN

Density **B**ased **S**patial **C**lustering of **A**pplications with **N**oise

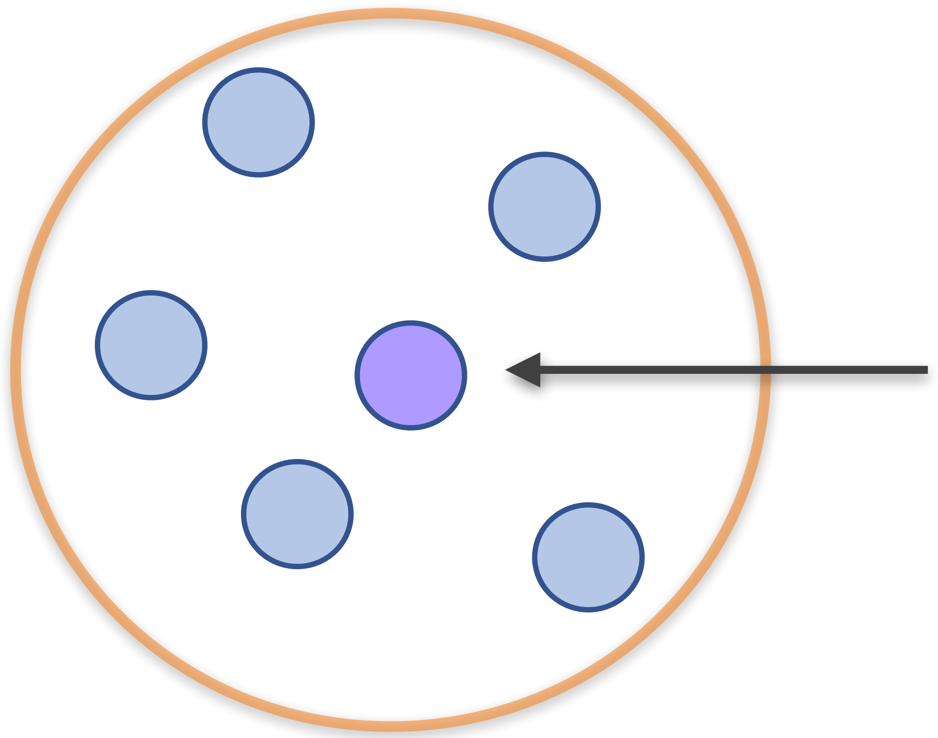
The algorithm uses the crowdedness of a region to determine the classification of a sample



Minimum Points

The minimum number of points (N) to form a dense region

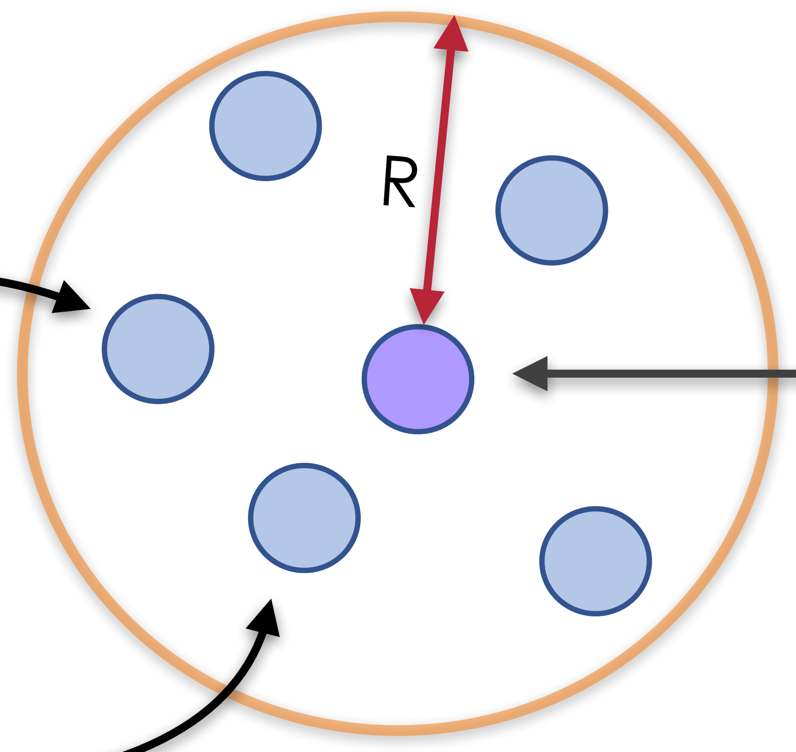
If N = 6, then the circle is a dense region



Point under consideration

Radius

The radius (R) of a dense region with respect to the point under consideration

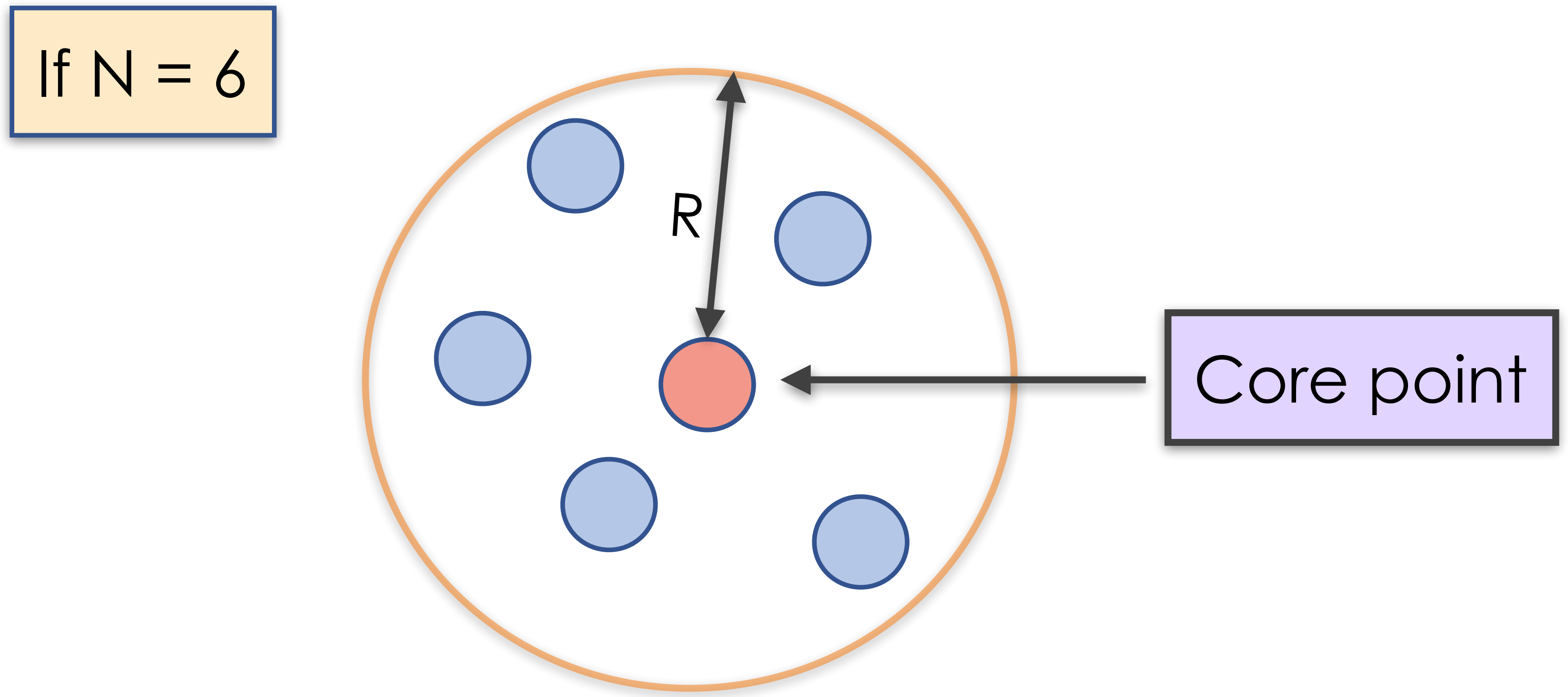


Point under consideration

Points within the circle of radius R are counted; including the point under consideration

Core Point

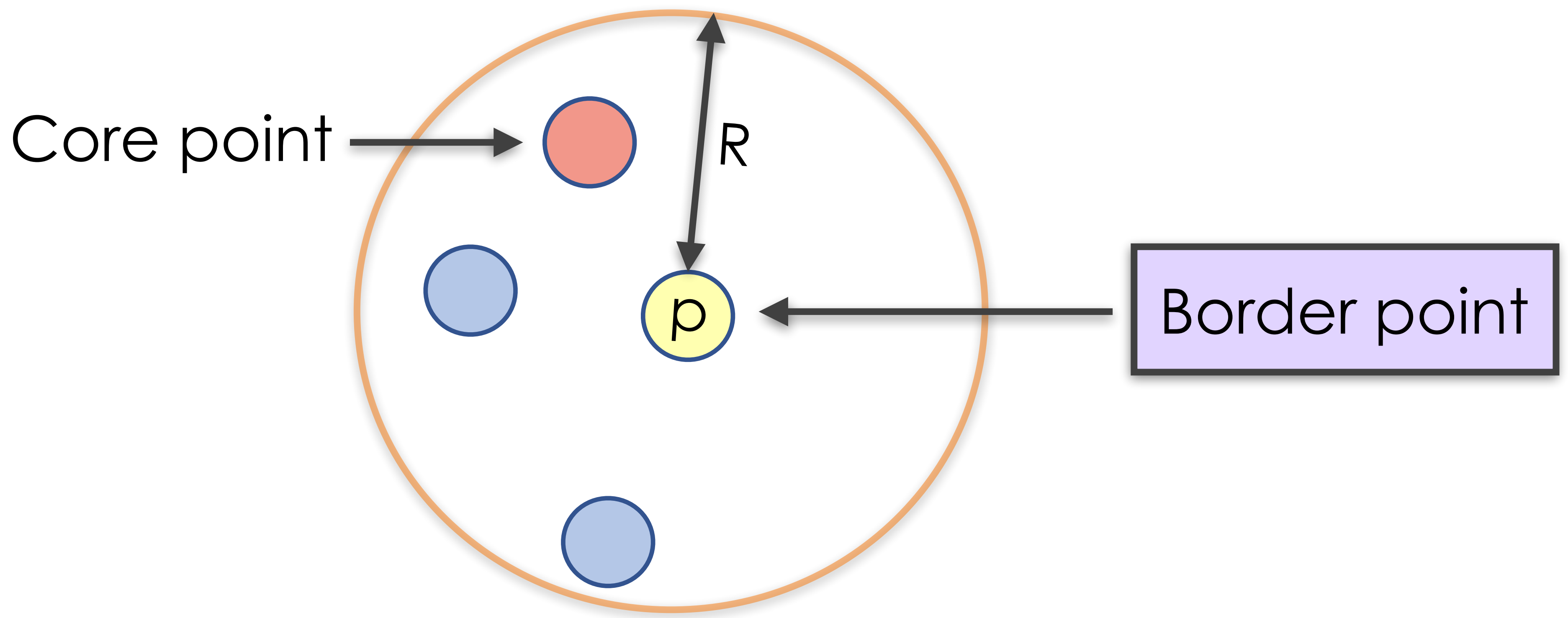
If there are at least N data points within a distance of R to a given data point, including the point itself, then that data point is classified as a Core point



Border Point

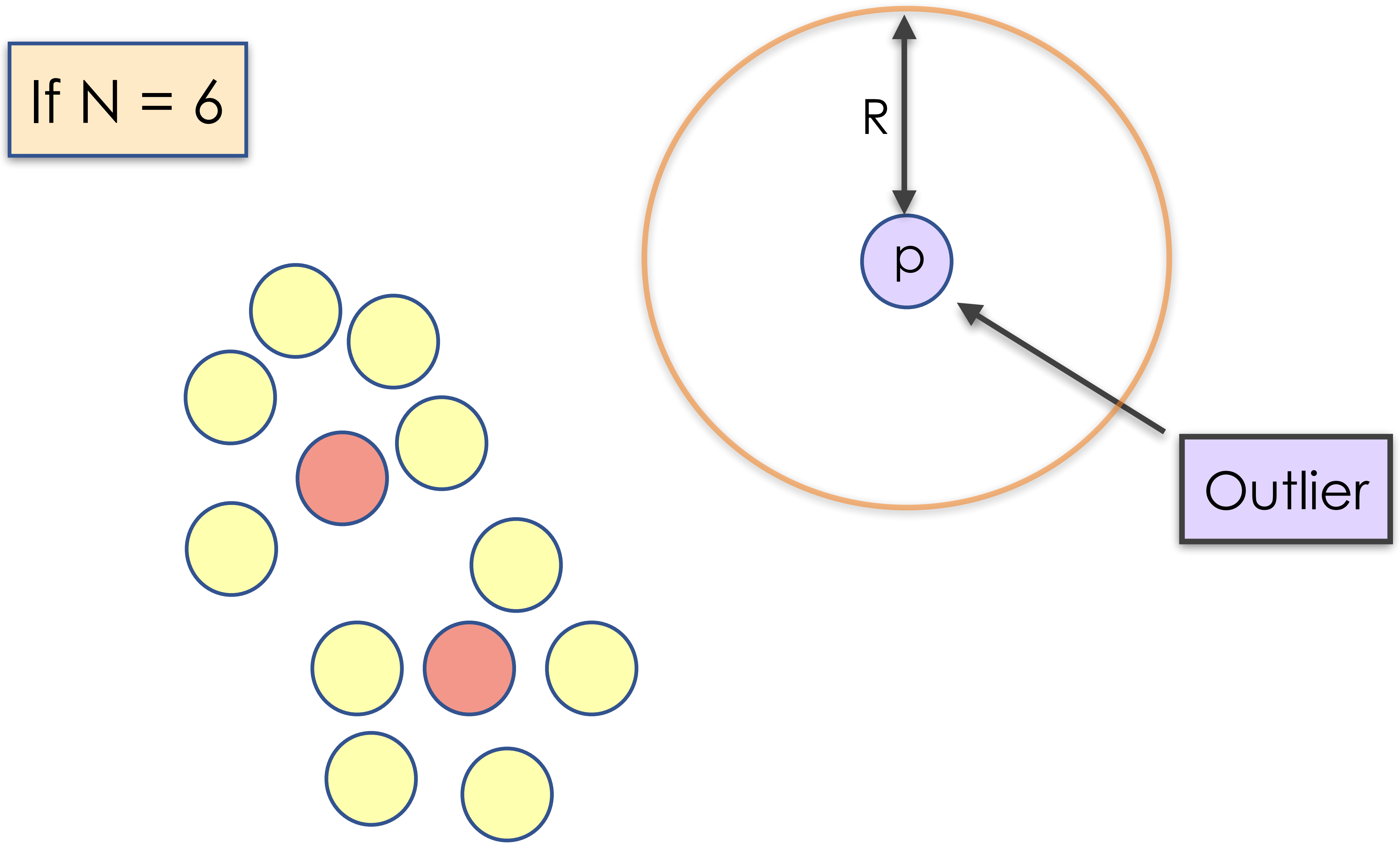
If point p has fewer than N data points within a distance of R to it but within a distance of R to a Core point, then point p is a Border point

If N = 6



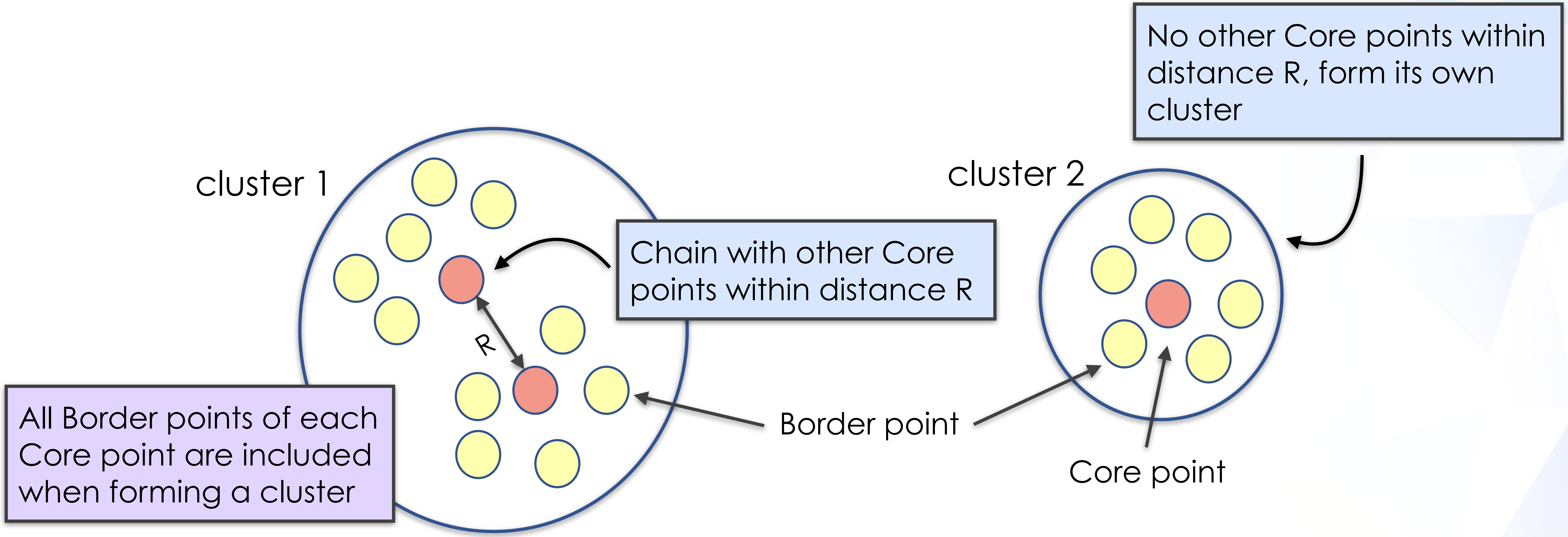
Outlier

If point p is neither a Core point or a Border point, then it is an Outlier



Forming a Cluster

To form a cluster, follow the chain of Core points



DBSCAN using sklearn

- Specify the radius R (eps) and number of neighbors N
- Outliers are assigned to a cluster value of -1

```python
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from sklearn.cluster import DBSCAN

data = pd.read_csv('iris.csv')
X = data.iloc[:, 1:-1].values

dbscan = DBSCAN(eps=0.6, min_samples=5)
row_cluster_map = dbscan.fit_predict(X)
print(row_cluster_map)
```

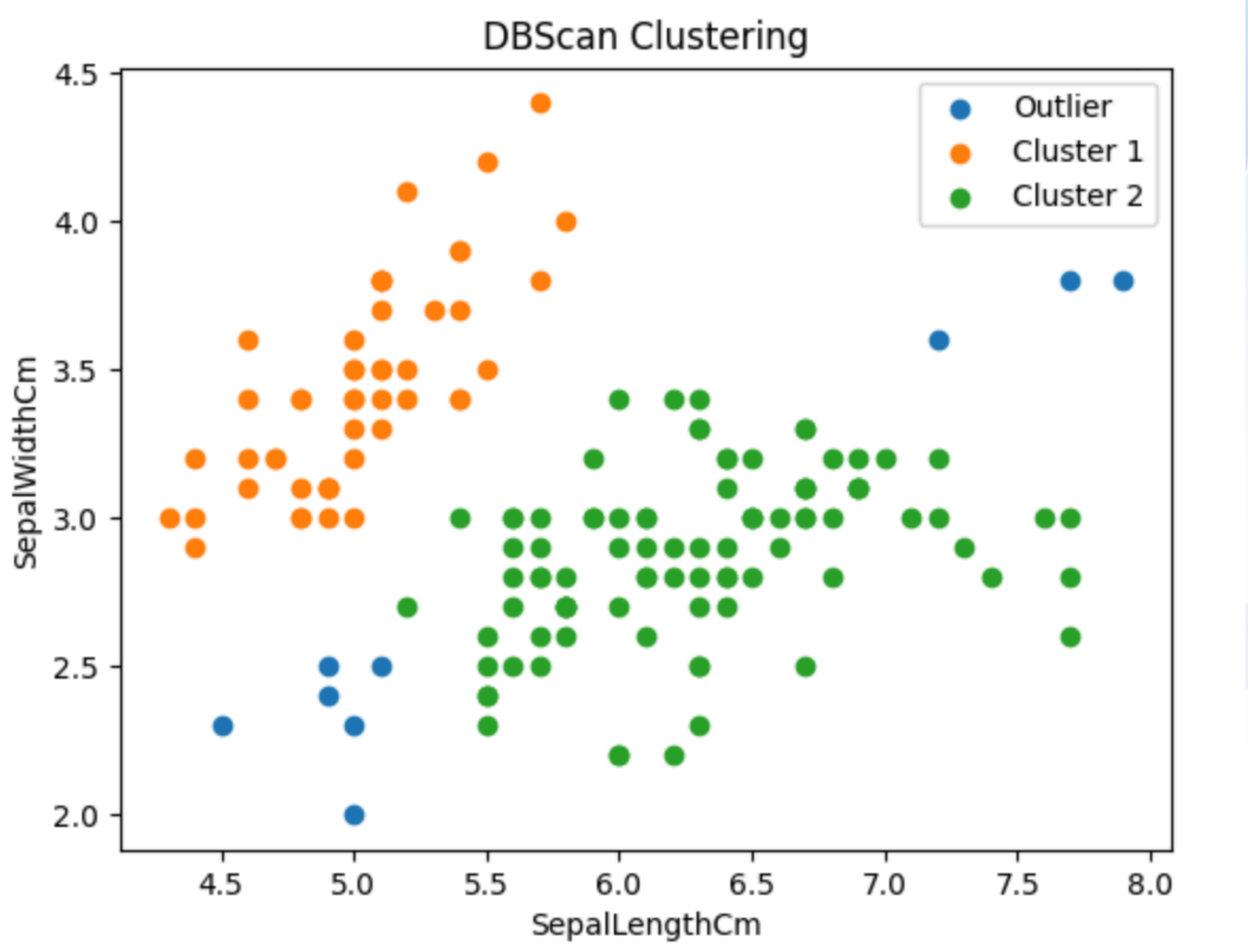
→

```
[ 0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0
  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0 -1  0  0  0  0  0  0
  0  0  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  1  1  1 -1  1  1  1  1  1  1  1  1  1  1  1  1
  1  1  1  1  1  1]
```

DBSCAN using sklearn

- Plotting the samples based on their assigned clusters
- The blue-colored samples are Outliers

```
for i in np.unique(row_cluster_map):
    label = 'Outlier' if i == -1 else 'Cluster ' + str(i + 1)
    plt.scatter(x=X[row_cluster_map == i, 0],
                y=X[row_cluster_map == i, 1],
                label=label)

plt.title('DBScan Clustering')
plt.xlabel(data.columns[1])
plt.ylabel(data.columns[2])
plt.legend()
plt.show()
```



THE END