

# Unsupervised Learning

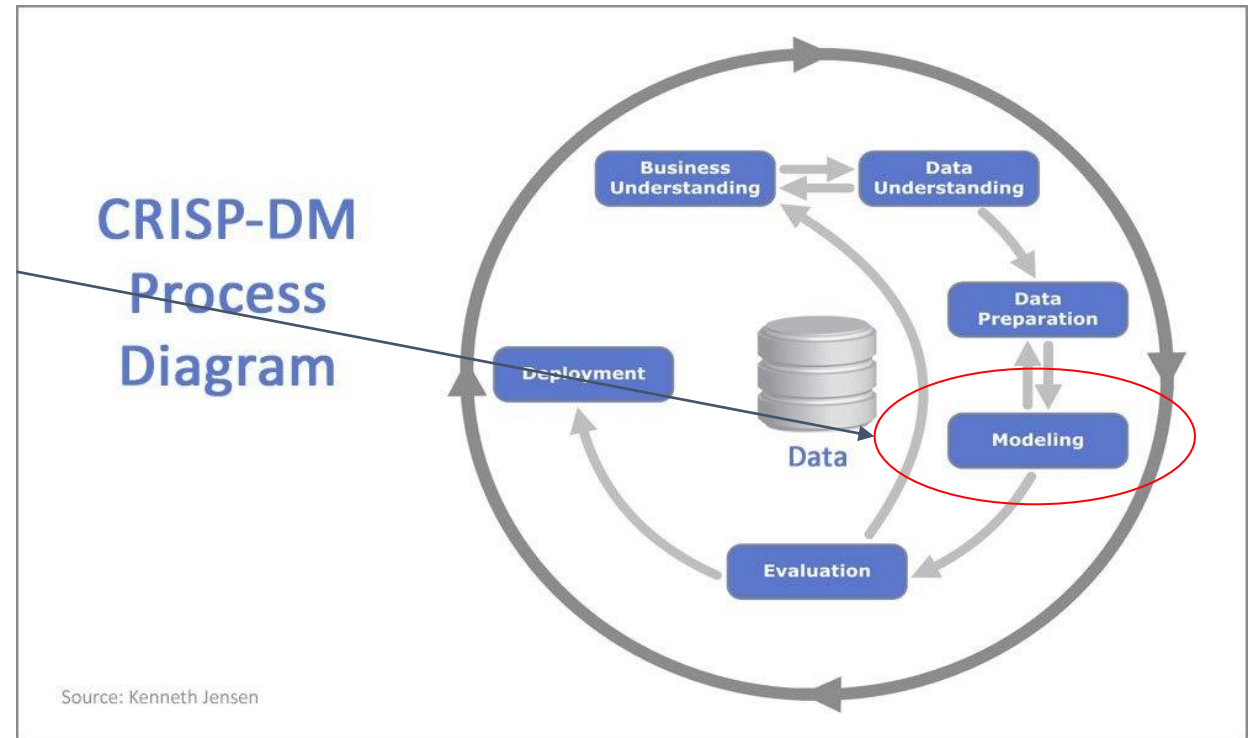
---


Data Science Program

# Outline

What is Unsupervised Learning ?

Unsupervised Transformation  
Clustering Algorithm



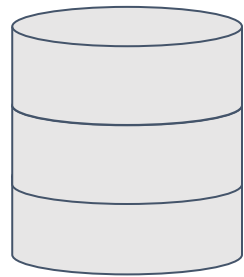


What is Unsupervised  
Learning ?

# What is Unsupervised Machine Learning ?

In Unsupervised Learning, there is no label, we only have features.

The learning algorithm is only exposed to the input data or features then asked to extract knowledge from this data.



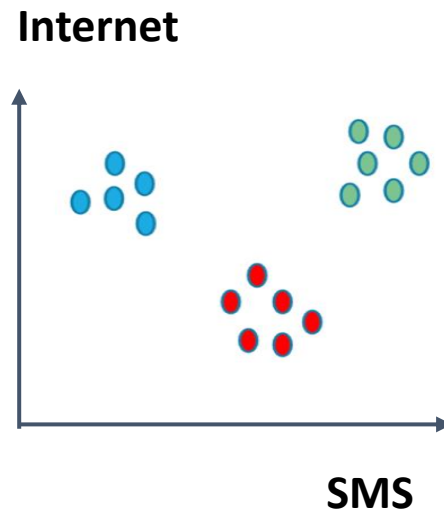
Data



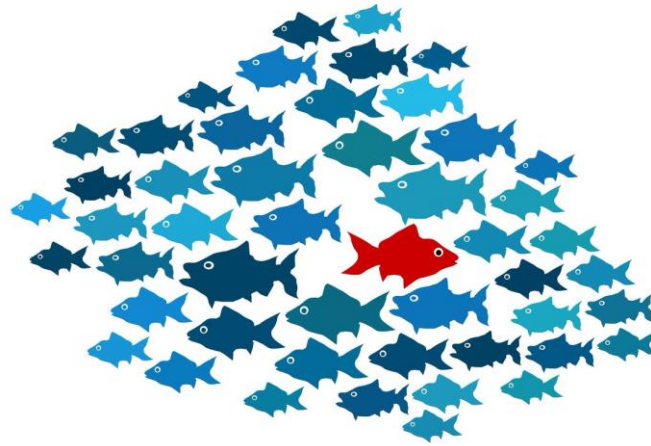
Knowledge

# Unsupervised Learning Application

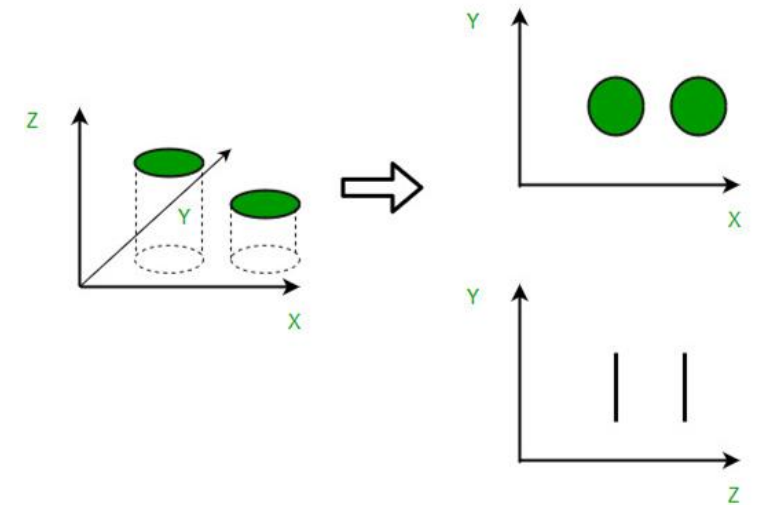
Customer Segmentation



Anomaly Detection



Dimensionality Reduction



# Analogy

Imagine you bring a child to a park and we let them observe. They will eventually find some pattern based on their interest. Like they found that there is a moving object and inanimate objects, there is also so many living things and so on.

# Two Types of Unsupervised Learning

## Unsupervised Transformation

- Dimensional Reduction
- Feature Extraction

## Clustering Algorithm

- Non Hierarchical
- Hierarchical

# Dimensional Reduction



# Dimensional Reduction

Transforming data using like dimensionality reduction can have many motivations:

- Visualization
- Better representation of the data
- Compressing data size

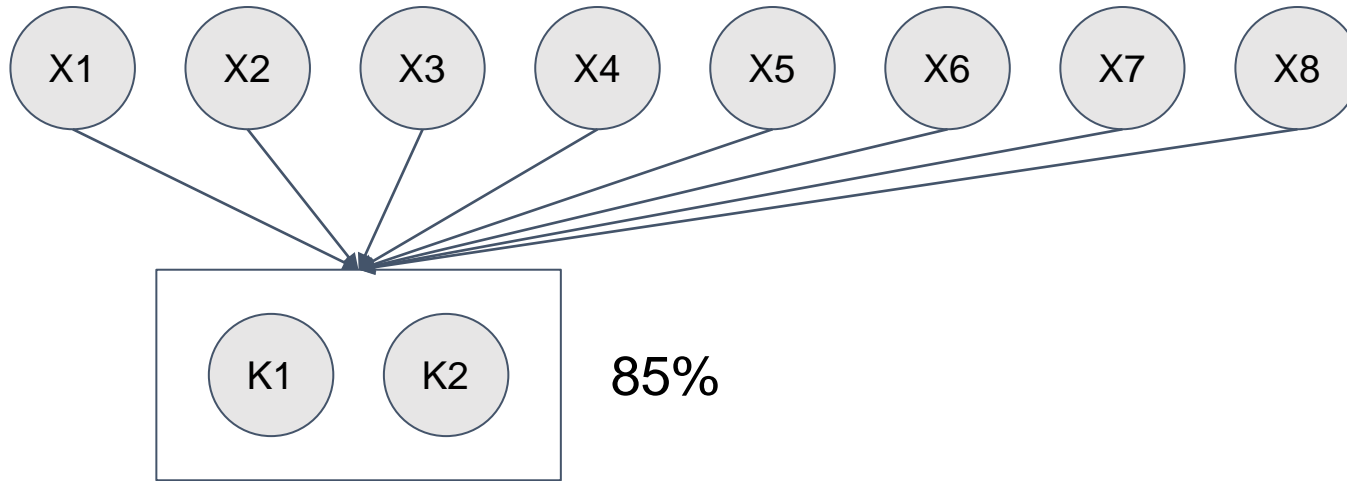
One of the methods you can use is Principal Component Analysis (PCA)

PCA utilizes correlation between features to reduce the dimension

# Analogy

To tell how your couple face in the high school, it is not necessary to mention his sharp nose, smooth skin, beautiful hair and so on. But just say, "My couple in the high school was beautiful." The word "beautiful" can already describe the previous description.

# Principal Component Analysis (PCA)



X1	x2	...	x8
...	...	...	....



K1	K2

What is principal component?

- Combination of your feature
- Features {X1, X2, ..., X8} are transformed into new set of features (K1, K2, ..., K8)
- New features uncorrelated to each other
- In analysis, we only select a subset of new features {K1, K2}
- Depend on how important they are for explaining the data

Pross:

- Reduce the dimension, e.g. 8 to 2
- Easier to visualize
- Easier to explore
- Can handle multicollinearity in linear model
- Faster process in modeling

Cons:

- Some information loss

# Principal Component Formula

feature:  $\{X_1, X_2, \dots, X_k\} \rightarrow$  principal component:  $\{K_1, K_2, \dots, K_k\}$

$$K_1 = a_{11} X_1 + a_{12} X_2 + \dots + a_{1k} X_k$$

$$K_2 = a_{21} X_1 + a_{22} X_2 + \dots + a_{2k} X_k$$

....

$$K_k = a_{k1} X_1 + a_{k2} X_2 + \dots + a_{kk} X_k$$

# Principal Component Example

feature: {X1, X2, X3, X4} → principal component: {K1, K2, K3, K4}

$$K1 = 0.680 X1 - 0.692 X2 + 0.038 X3 + 0.238 X4 \text{ (explained 71.98\%)}$$

$$K2 = 0.145 X1 - 0.060 X2 - 0.882 X3 - 0.445 X4 \text{ (explained 15.29\%)}$$

$$K3 = 0.123 X1 - 0.150 X2 + 0.466 X3 - 0.863 X4 \text{ (explained 8.07\%)}$$

$$K4 = -0.708 X1 - 0.703 X2 - 0.062 X3 - 0.130 X4 \text{ (explained 4.67\%)}$$

# Why PCA?

We already mentioned that, some process will be better:

- data exploration
- data visualization
- modeling

For some cases, some features will be redundant. Often, there is no point on analyzing all the redundant features.

# Issues in PCA

Should we standardize the features?

Optimal number of components we take?

# Illustration: Breast Cancer

## Dataset Description:

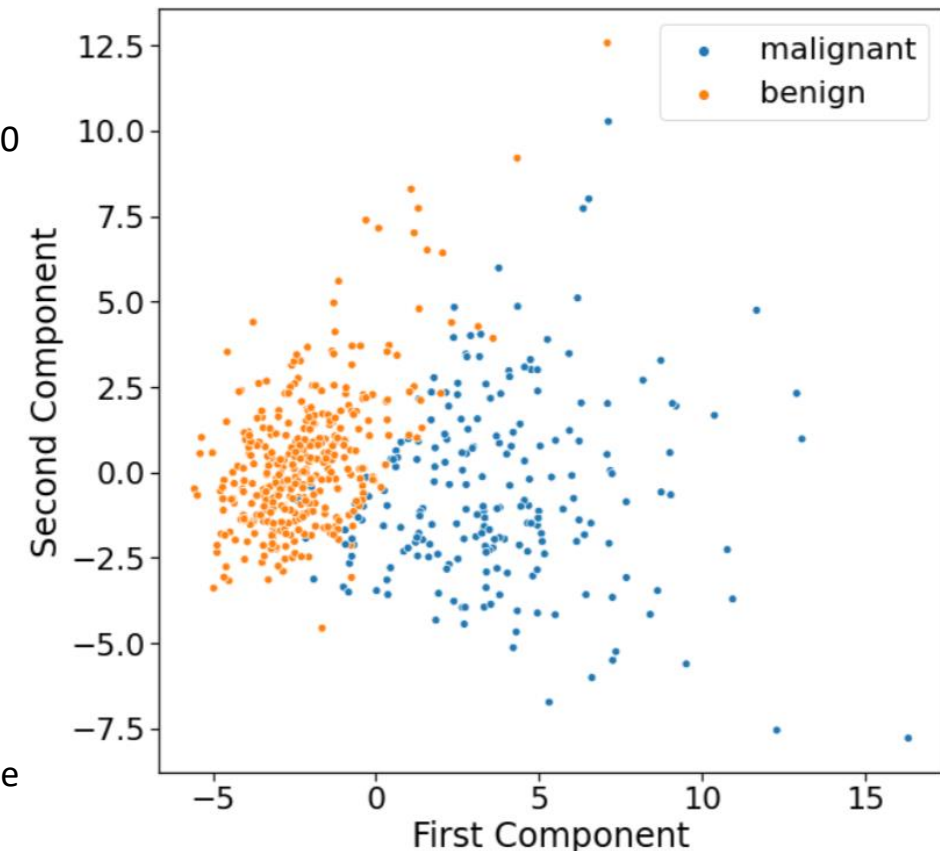
- Dataset: Breast Cancer
- Features: 30 features extracted from images
- You can't see hidden structure in the data easily using data visualization
- If you use scatter plot, this dataset has 30 features, which would result in  $30 * 14 = 420$  scatter plots!

## Task:

- Scale the features using Standard Scaler
- Compute how much information extracted using two components
- Reduce the dimension from 30 to 2 using two components
- Plot the first components and the second components
  - x axis as the first component
  - y axis as the second component
  - hue: Diagnosis

## Result:

- Information extracted from 2 components is 63.24 %
- We reduce the dimension from 30 to 2 and the lost information is 36.76%, but you can see that those two features can separate benign and malignant quite well
- This also means that you can see the hidden structure within the data
- Malignant points are more spread than benign point

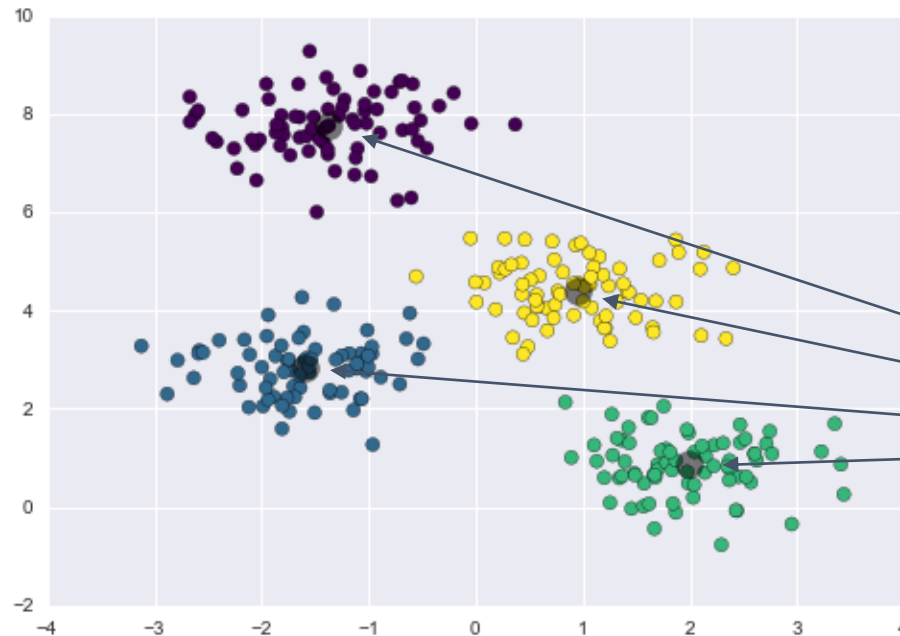




# Clustering

# Clustering

- Clustering is the task of partitioning a dataset into groups
- Data points within a group are similar
- Data points between groups are different

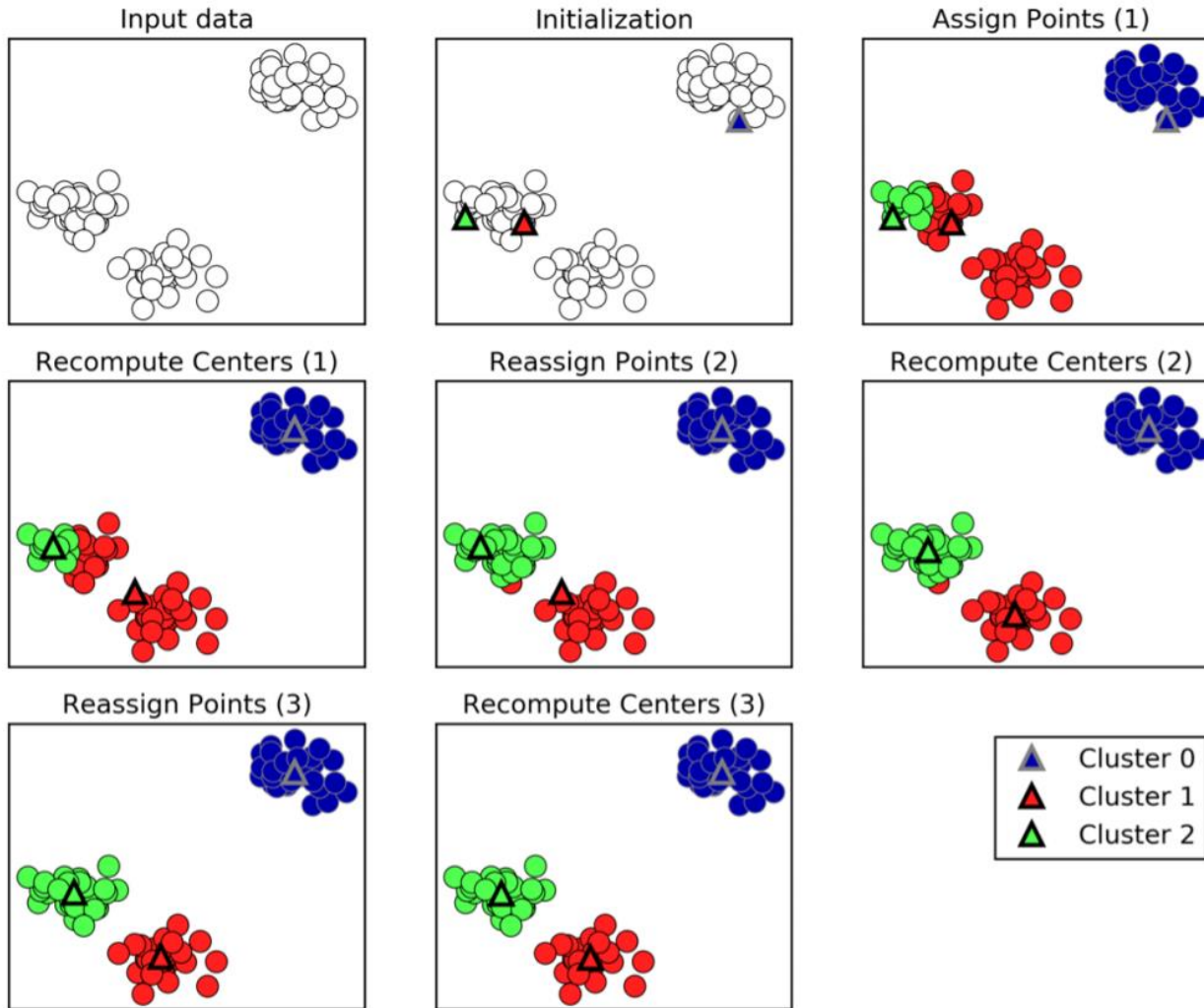


- Clustering is somewhat similar to classification
- But, there is no ground truth
- Each group has no a priori meaning

Centroids

# Clustering: K-Means

# Non Hierarchical: K-Means



Measuring similarity:

- We can measure how close a data point to another using distance concept like Euclidean distance
- Made for clustering with numerical variable

How K-Means work, let's say we want three clusters:

1. Initialization: set cluster positions/centroids randomly
2. Assign Points(1): assign each data point to the closest cluster
3. Recompute Centers(1): update the cluster position based on one mean of the assigned points
4. Repeat step 2 and 3 until the cluster position remained unchanged

# Python Exercise: K-Means

- Generate data using `make_blobs` with `random_state = 42`
- Plot the data using scatter plot
- Fit k-means method (3 clusters)
- Plot the data and clustering result using scatter plot

# Issues in K-means

Should we standardize the features?

Optimal number of cluster?

- Elbow method
- Silhouette method
- Domain knowledge

# Elbow Method

**Key principle:** The criteria that we can use is when the addition of one cluster does not provide a significant change in the level of similarity.

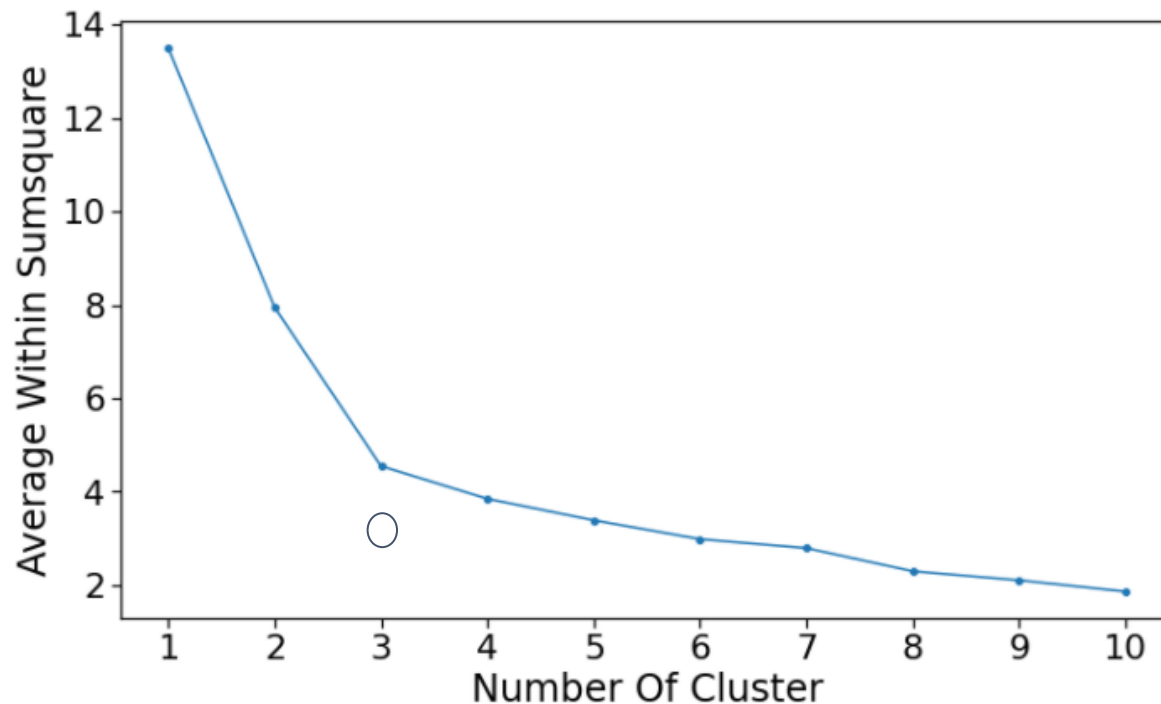


Illustration:

The similarity level of the two clusters to three clusters changed significantly. After that, changes in the similarity level began to be insignificant. Therefore, in this case three clusters is the optimal number of clusters.

This method is vulnerable to subjectivity.

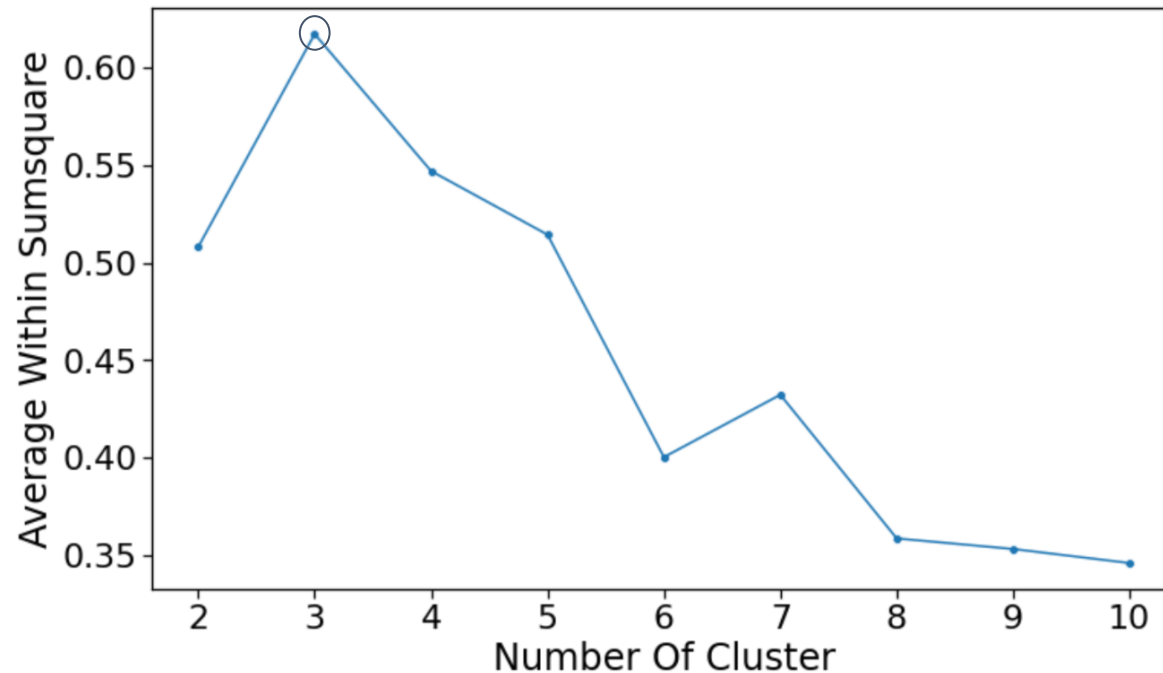
# Similarity Level

- Average Sum Square Error
- F-statistics



# Silhouette Score

**Key principle:** a measure of how close each point in one cluster is to points in the neighboring clusters



Value: from -1 to 1

- 1: clusters are clearly distinguished
- 0: clusters are not clearly distinguished
- -1: clusters assigned the wrong way

$$s(i) = (b - a) / \max(a, b) \rightarrow s(1), s(2), \dots, s(3)$$

- $s(i)$ : silhouette score of  $i$ -th cluster
- $a$ : average **intra** cluster distance
- $b$ : average **inter** cluster distance

# Python Exercise: K-Means 1

Analyze generated data from `make_blobs`

- Set 3 main spots (`random_state = 42`)
- Plot the data using scatter plot
- Determine the optimal number of cluster based on your observation
- Plot the data and clustering result (optimal number of cluster) using scatter plot

# Python Exercise: K-Means 2

Analyze data “ilustrasi k means.csv”

- Plot the data using scatter plot
- Determine the optimal number of cluster
  - Elbow method
  - Silhouette method
- Plot the data and clustering result (optimal number of cluster) using scatter plot

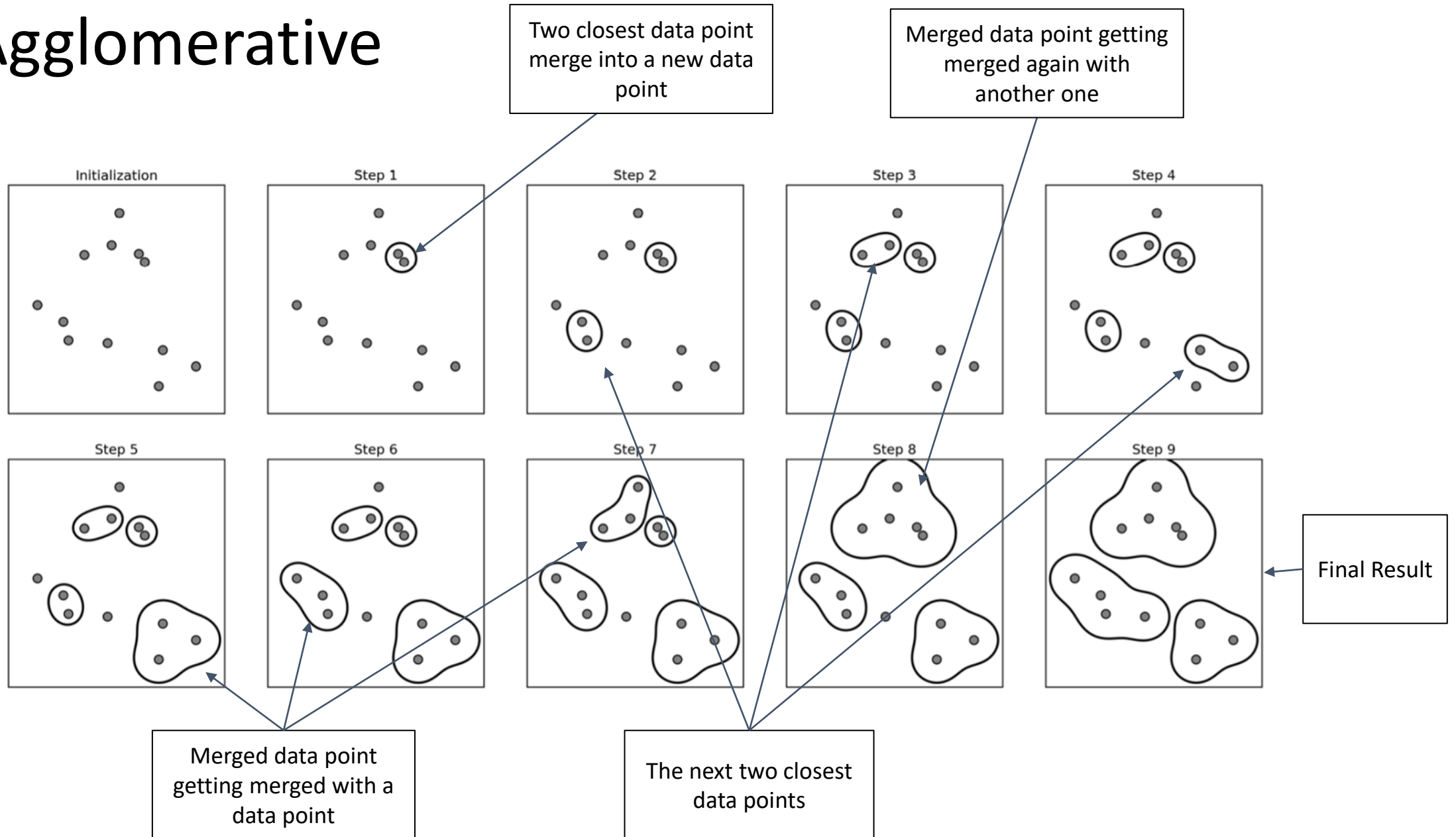


# Clustering: Agglomerative

# Agglomerative

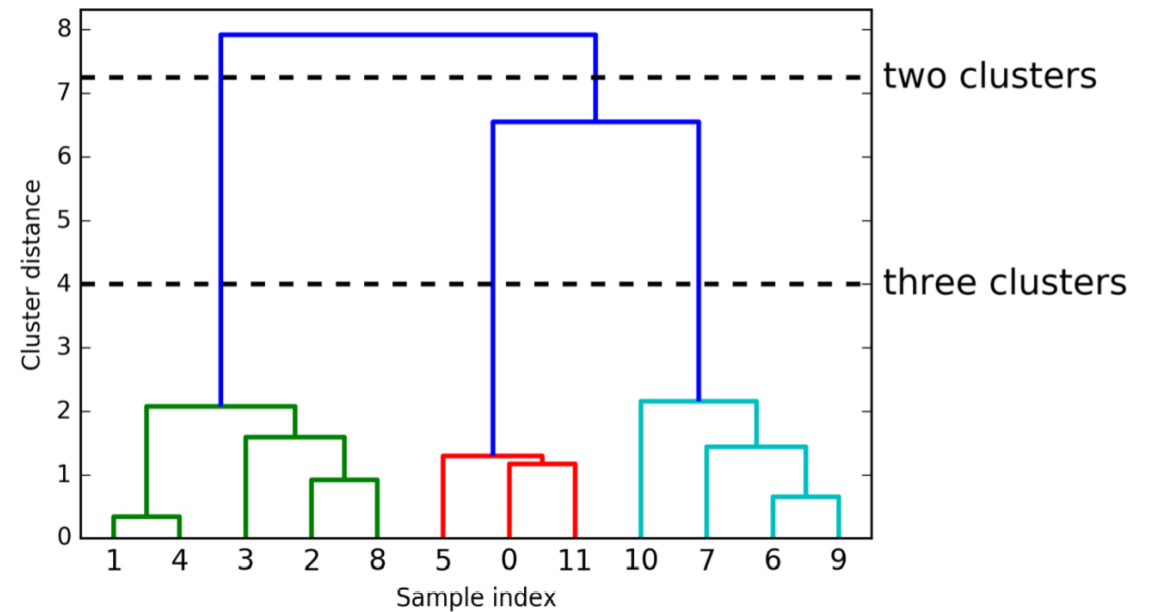
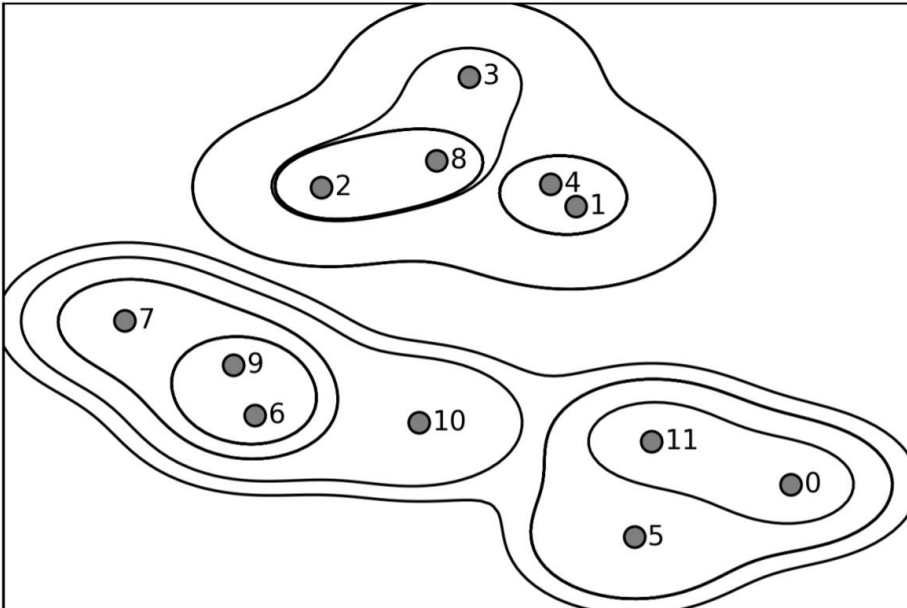
- This algo treats each data points as its own cluster
- Merge the two most similar into one single data point, linkage method:
  - ward
  - complete
  - average
- Repeat until stopping criterion is satisfied (number of cluster)
- Not recommended for large data size
- Made for clustering with numerical variable

# Agglomerative



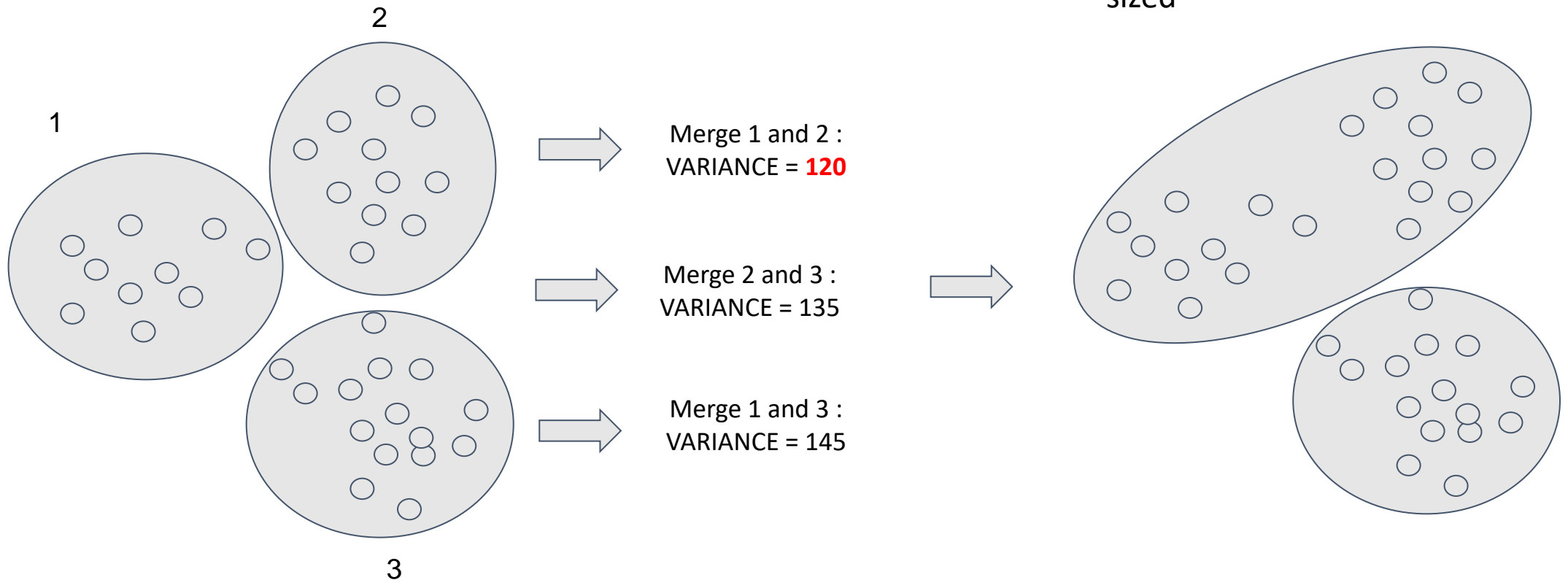
# Dendrograms

- Using dendrogram, you basically summarize the whole process
- Even for more than 2 dimensions



# Linkage: Ward

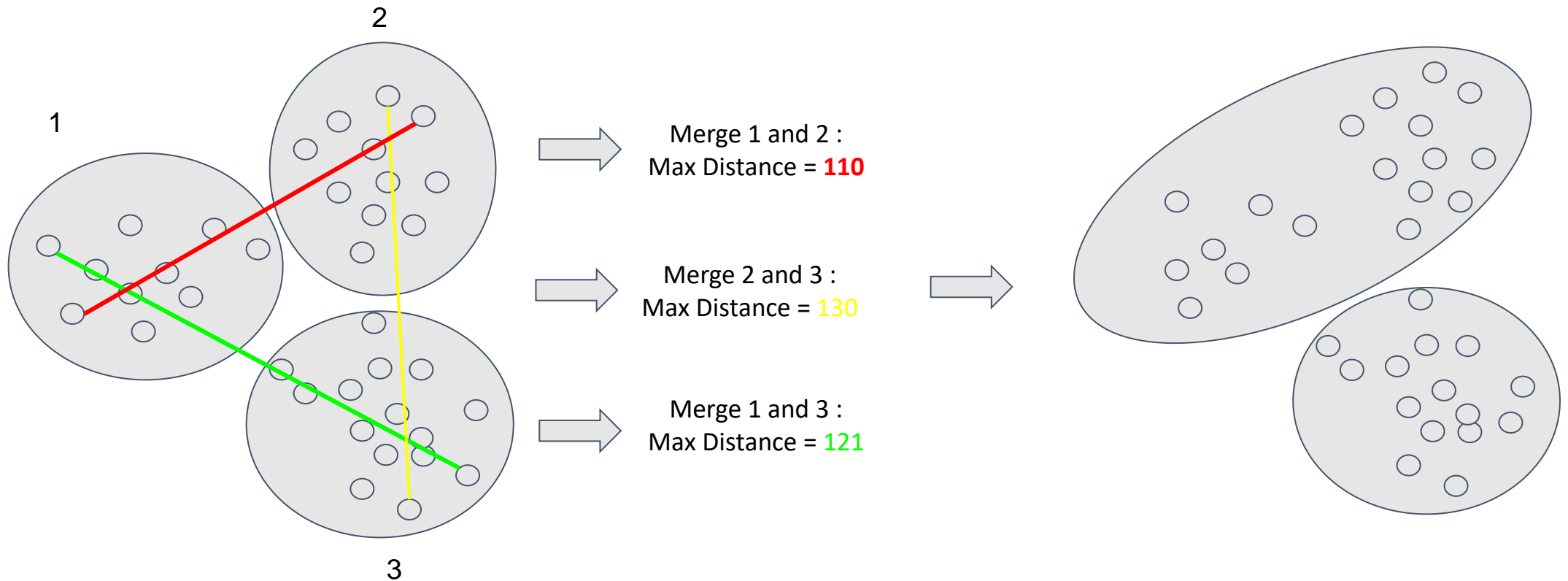
This often leads to clusters that are relatively equally sized



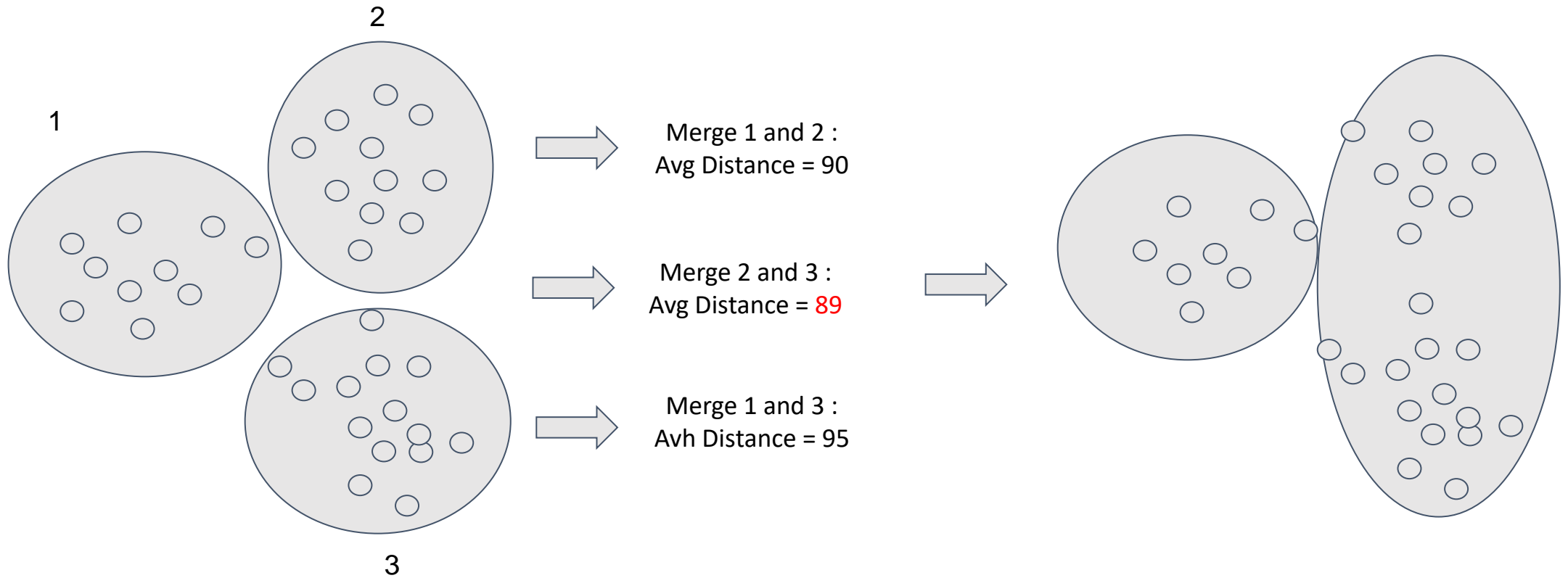


# Linkage: Complete

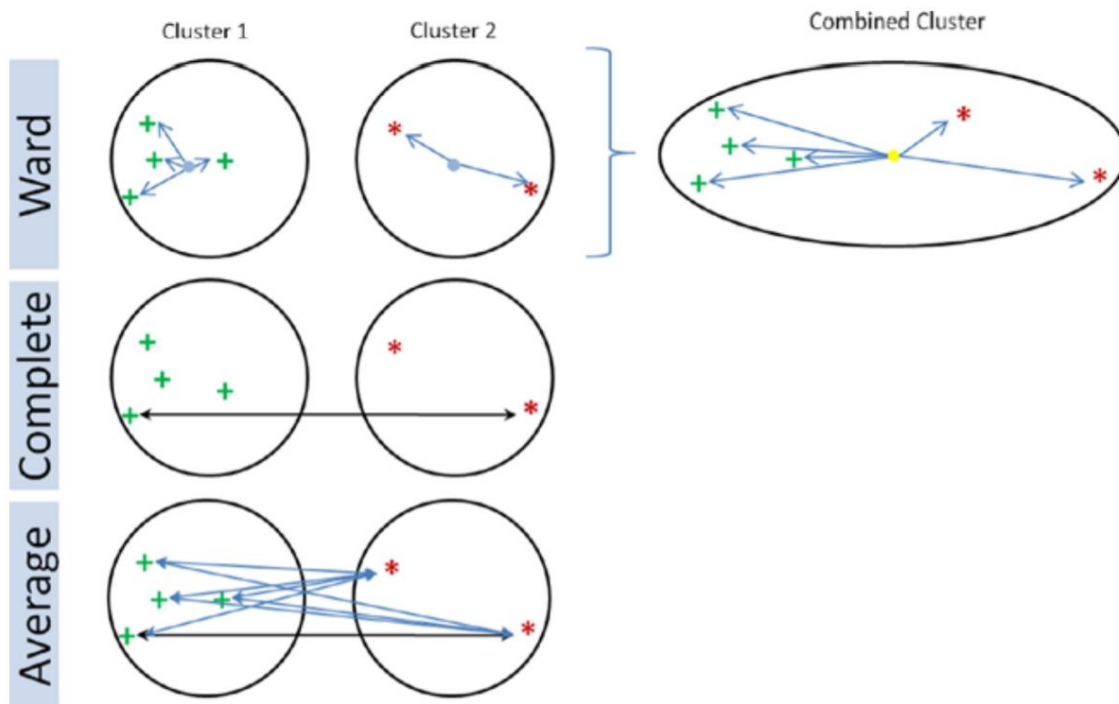
This method is very sensitive to outlier



# Linkage: Average



# Linkage Summary



Ways to represent merged data point:

- Ward: the default choice, ward picks the two clusters to merge such that the variance within all clusters increases the least.
- Complete: complete linkage (also known as maximum linkage) merges the two clusters that have the smallest maximum distance between their points.
- Average: average linkage merges the two clusters that have the smallest average distance between all their points.

# Issues

- Should we standardize the features?
- What distance/dissimilarity measure should be used?
- What type of linkage should we use?
- Where should we cut the dendrogram in order to obtain clusters?

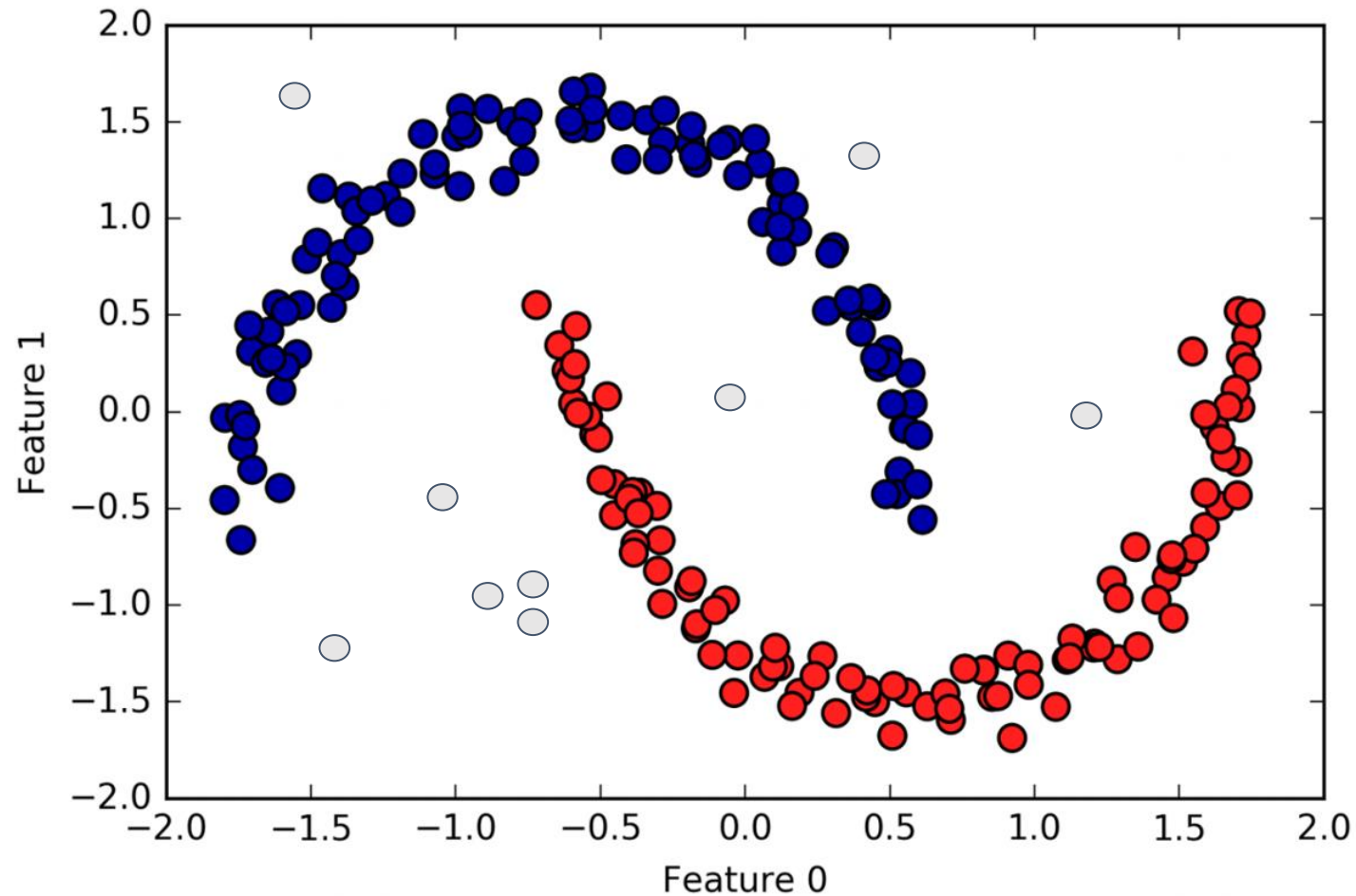
# Python Exercise: Clustering Agglomerative

Analyze data “ilustrasi k means.csv”

- Plot the data using scatter plot
- Make dendrograms
  - ward linkage
  - average linkage
  - complete linkage
- Compare the result:
  - is it different?
  - what about another data?
- Plot the data and clustering result (optimal number of cluster from either ward, average, complete) using scatter plot

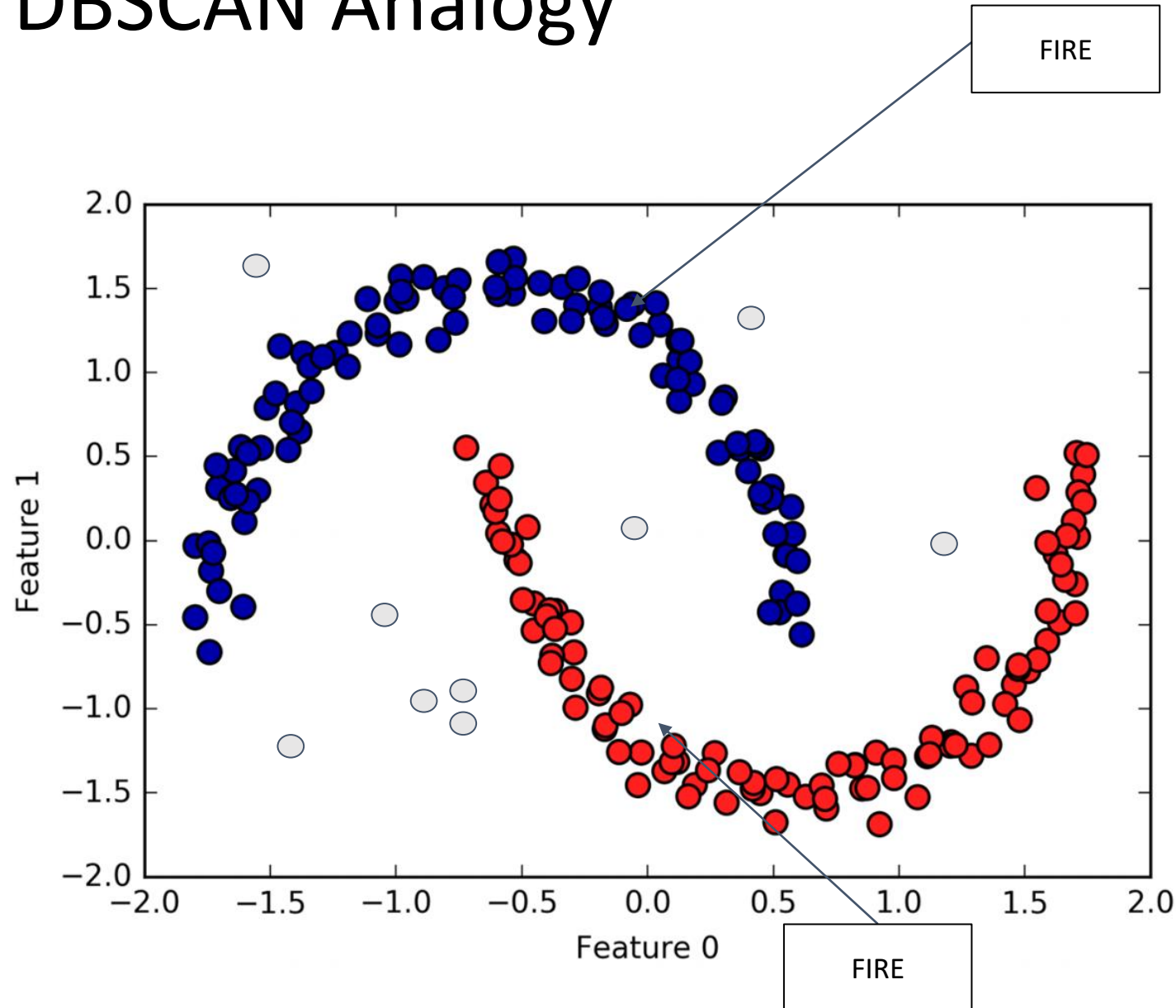
# Clustering: DBSCAN

# DBSCAN



- Main idea: identifying points that are in “crowded” where many data points are close together and separate cluster by regions that relatively empty or rare
- Can capture complex shapes
- Able to identify points that are not part of any cluster

# DBSCAN Analogy

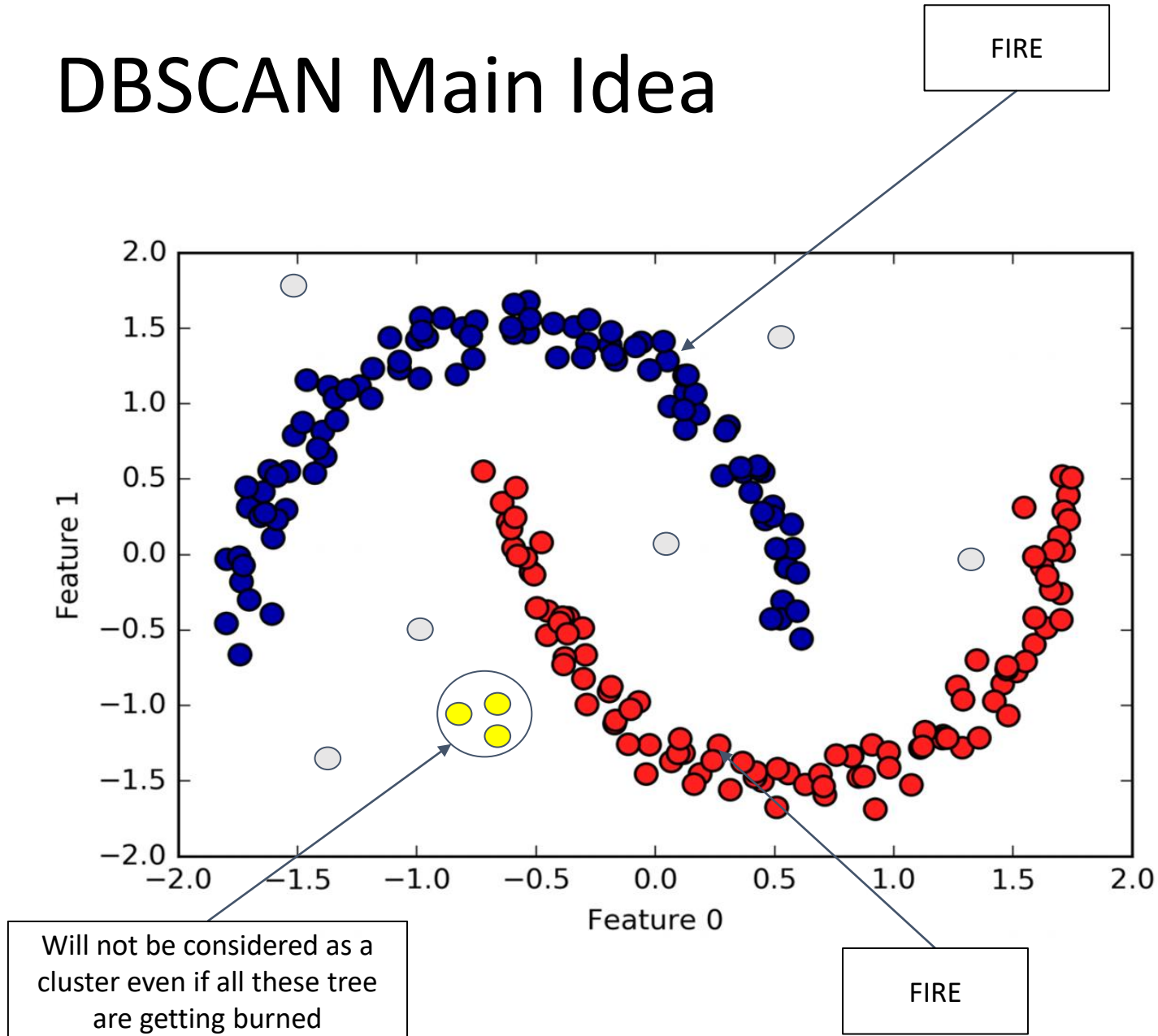


It can be seen as forest fire:

- Fire starts somewhere in the blue region
- Then burn another tree that close enough until all tree in the blue region burned
- Other fire ignited somewhere in the red region
- Then fire spread until all tree in the red region burned
- Some trees are safe because too far



# DBSCAN Main Idea



Identified points:

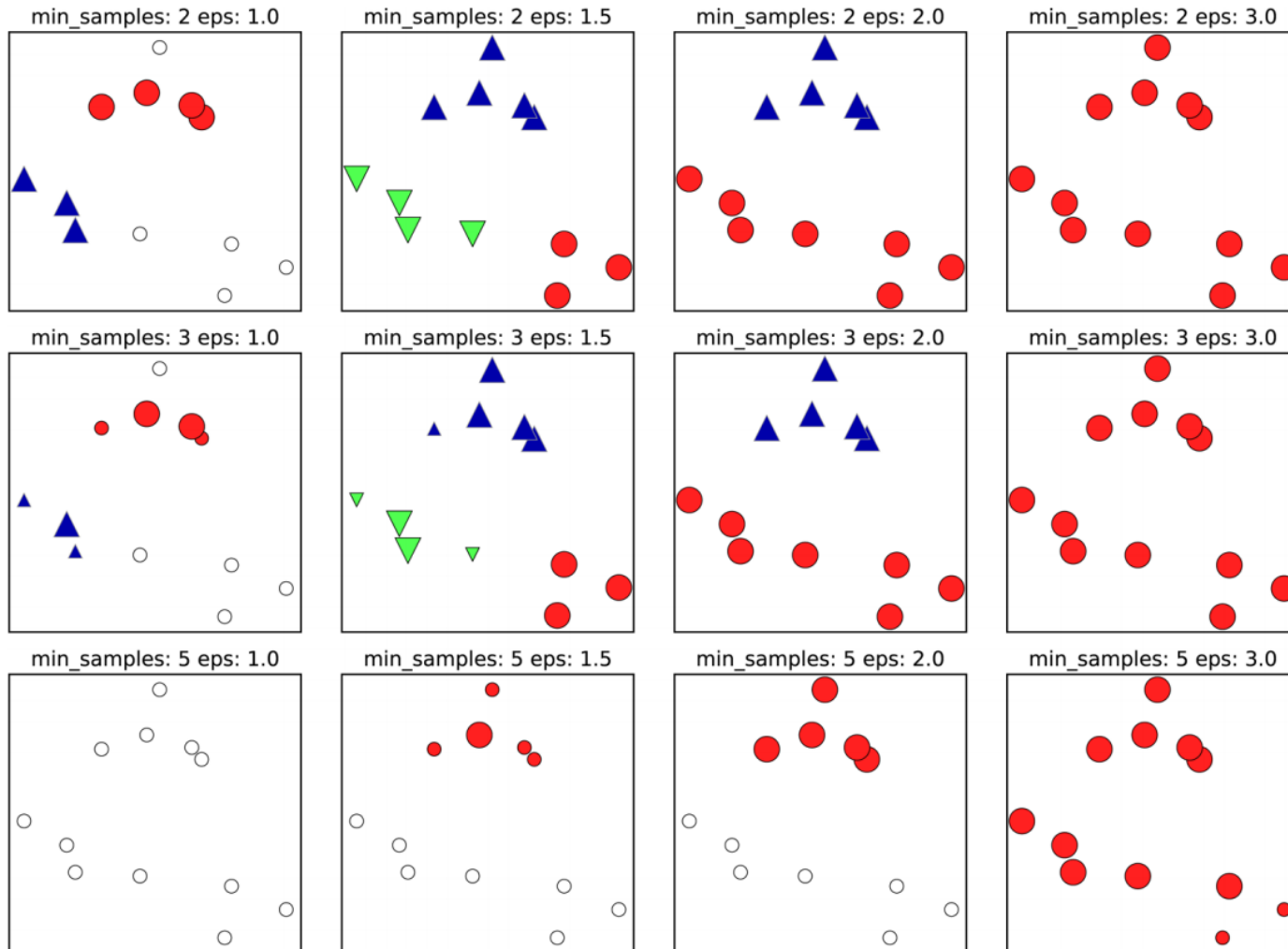
- core samples (red): cluster 1
- core samples (blue): cluster 2
- boundary points (edge): the last trees that getting burned (either in cluster 1 or cluster 2)
- noise (grey and yellow): not part of any cluster

DBSCAN doesn't require to set the number of cluster

DBSCAN needs two hyperparameters:

- min samples: minimum number of tree burned to consider as a region/min cluster size (e.g. yellow region)
- eps: range of fire

# DBSCAN Hyperparameters



Left to right: increasing epsilon

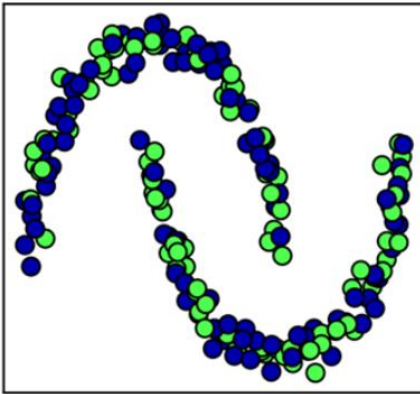
- larger eps
- more data points assigned into clusters
- bigger cluster size, fewer number of cluster and more noise

Up down: increasing minimum samples

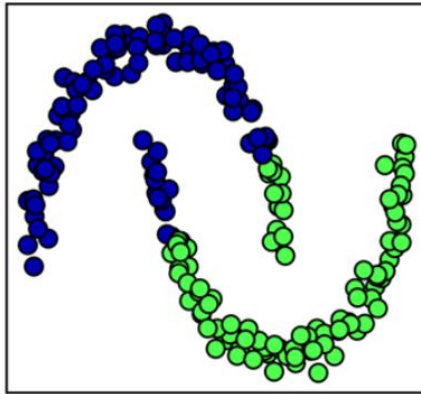
- larger minimum samples
- fewer core samples
- more noise

# Why DBSCAN ?

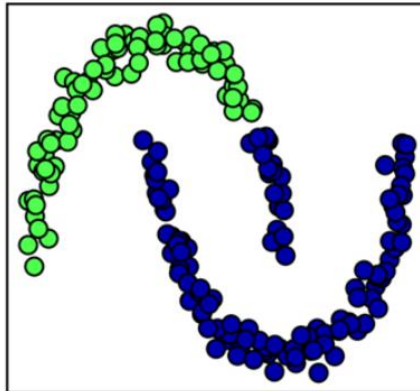
Random Assignment



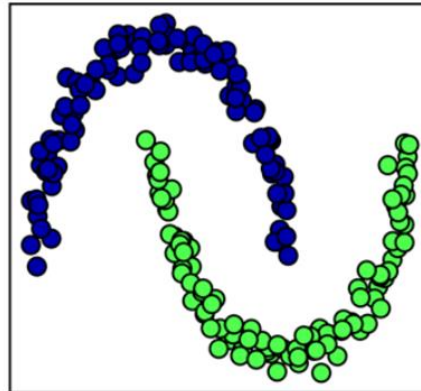
K-Means



Agglomerative



DBSCAN



- Can capture complex shapes
- You can compare the result with k-means and agglomerative clustering

# Issues

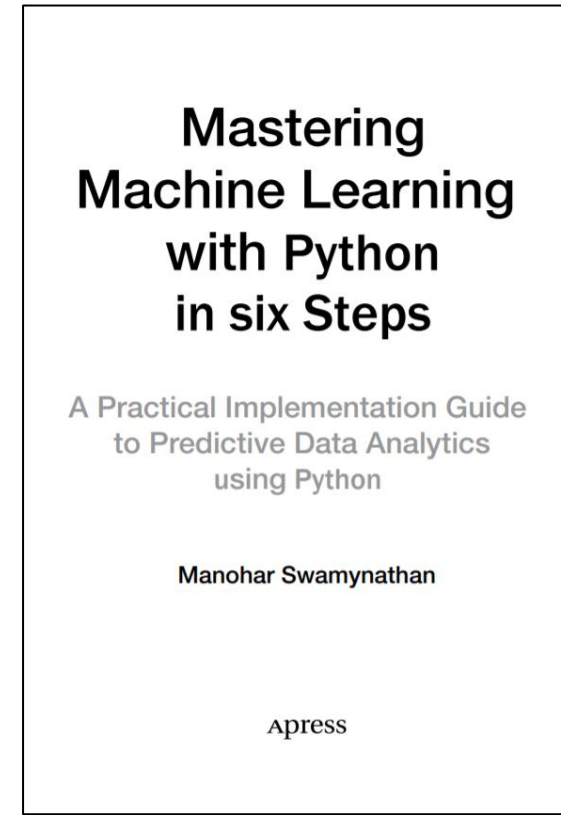
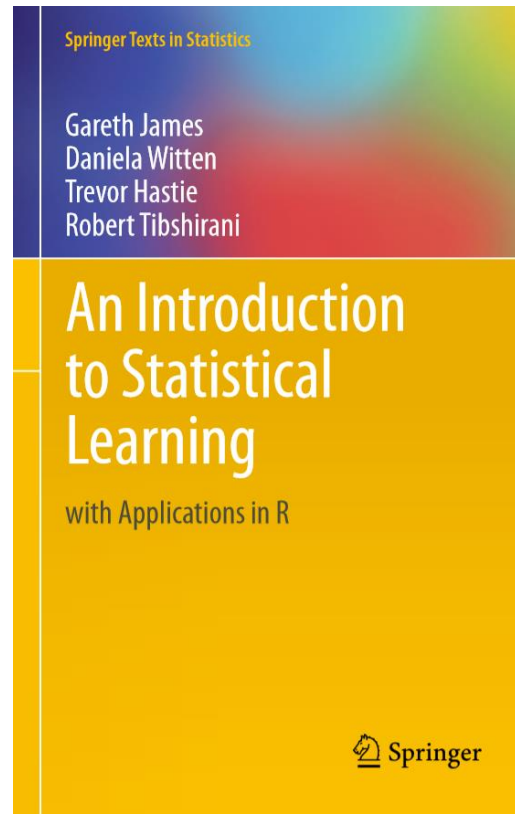
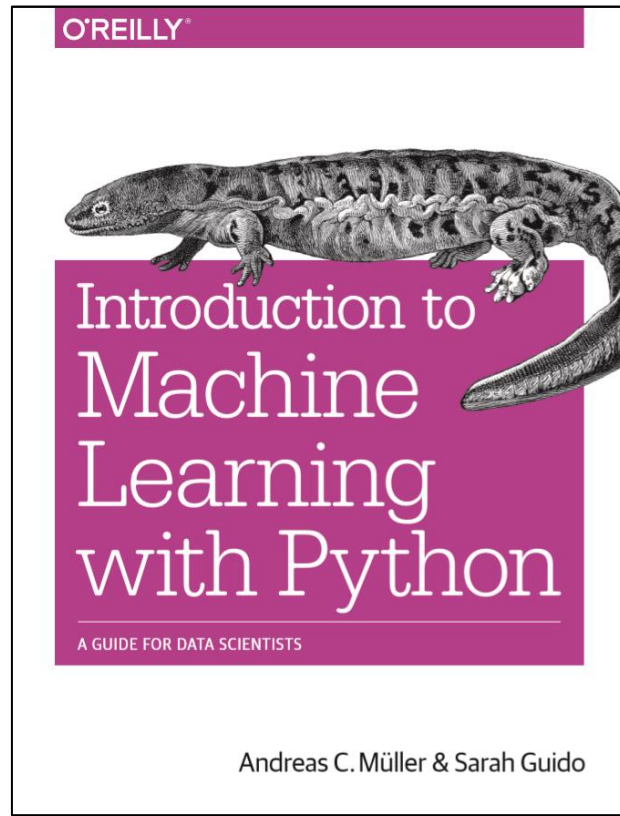
- Should we standardize the features?
- Determine the optimal number of min samples and eps

# Python Exercise: DBScan

Analyze data “ilustrasi k means.csv”

- Plot the data using scatter plot
- Make initial clustering
- Determine the optimal hyperparameter
  - minimum sample per cluster
  - epsilon
- Plot the data and clustering result (optimal hyperparameter) using scatter plot

# References



# References

[https://en.wikipedia.org/wiki/Elbow method \(clustering\)](https://en.wikipedia.org/wiki/Elbow_method_(clustering))

<https://www.geeksforgeeks.org/elbow-method-for-optimal-value-of-k-in-kmeans/>

<https://skillplus.web.id/elbow-method/>

[https://scikit-learn.org/stable/auto\\_examples/cluster/plot\\_kmeans\\_silhouette\\_analysis.html](https://scikit-learn.org/stable/auto_examples/cluster/plot_kmeans_silhouette_analysis.html)