



Clustering, Density Estimation, and Anomaly Detection

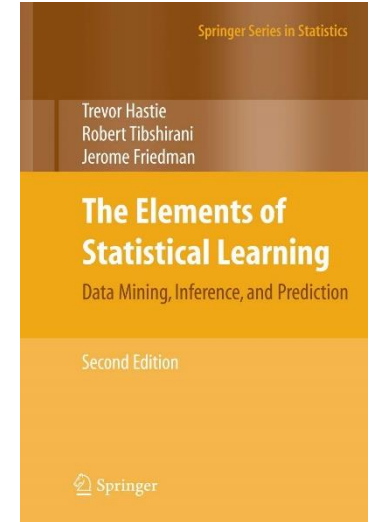
Assoc. Prof. Karl Ezra Pilario, Ph.D.

Process Systems Engineering Laboratory
Department of Chemical Engineering
University of the Philippines Diliman

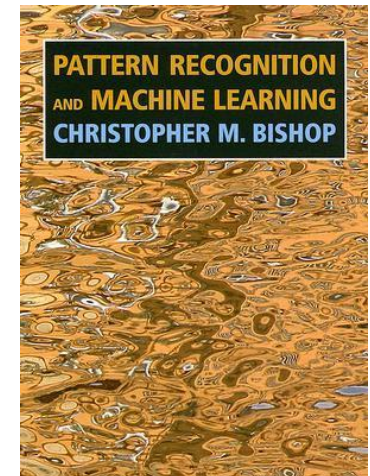
Outline

- Clustering
 - K-means Clustering
 - Hierarchical Clustering
 - Density-based Clustering
 - Gaussian Mixture Model
- Density Estimation
 - Kernel Density Estimation (KDE)
- Anomaly Detection
 - Application of KDE
 - One-class SVM
 - Local Outlier Factor
 - Isolation Forest

Hastie *et al.* (2008)
The Elements of Statistical Learning.
2nd Ed. Springer.

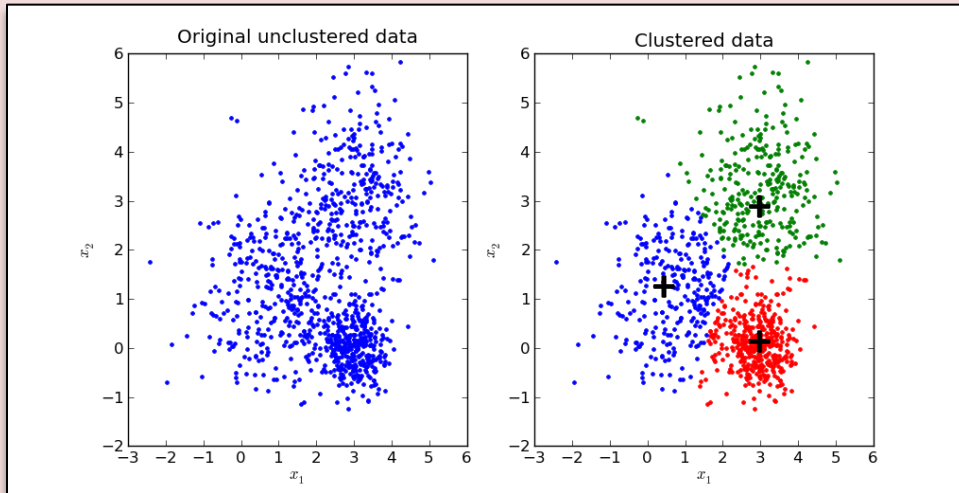


Bishop (2006)
*Pattern Recognition and
Machine Learning.* Springer.



What is Clustering?

The problem of identifying **groups** or **clusters** of data points in a multi-dimensional space.



<https://images.app.goo.gl/HjnuqnNrk74DRddv9>

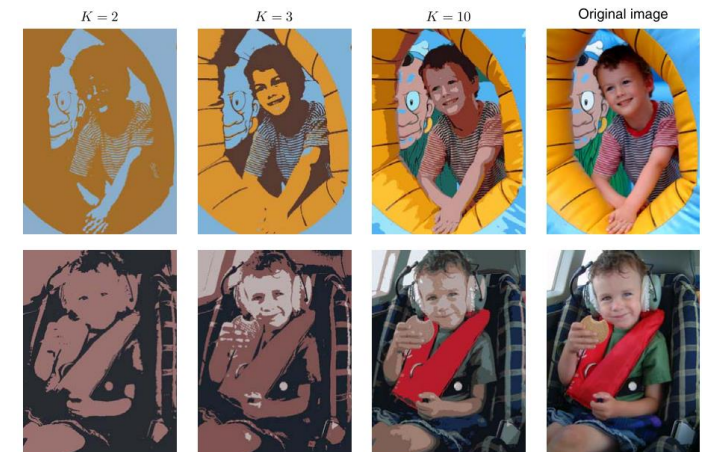
- It is NOT the same as the *classification* problem.
 - In clustering, class labels are unknown. Groupings are established solely from the locations of points.
- **Interpretation:** Samples within the same cluster are more *similar* to each other than those assigned to other clusters.
- The central idea is the notion of “**similarity**”.

Some applications of Clustering:

- **Customer segmentation**
 - Given the attributes of your customers, can they be grouped so we can target our ads to each group of common people?
- **Recommendation systems**
 - It has been reported that Netflix identified 2,000 clusters of subscribers of similar taste by tracking their viewing habits. For example, *Black Mirror* is favored by Clusters 290 and 56.
 - **Source:** <https://www.vulture.com/2018/06/how-netflix-swallowed-tv-industry.html>

- **Image compression**

- An image can be re-colored by clustering “similar” colors as one.
- From left to right, there are 2, 3, and 10, clusters chosen. The last column is the original image.



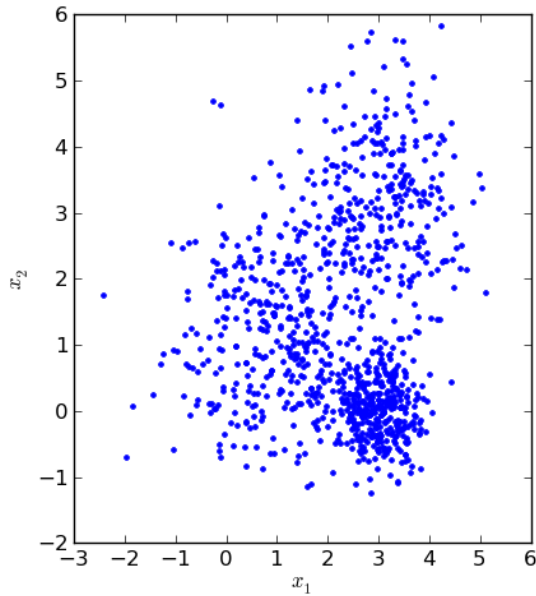
Bishop (2006). Pattern Recognition and Machine Learning. Springer.

Side Note: Clustering \neq Classification

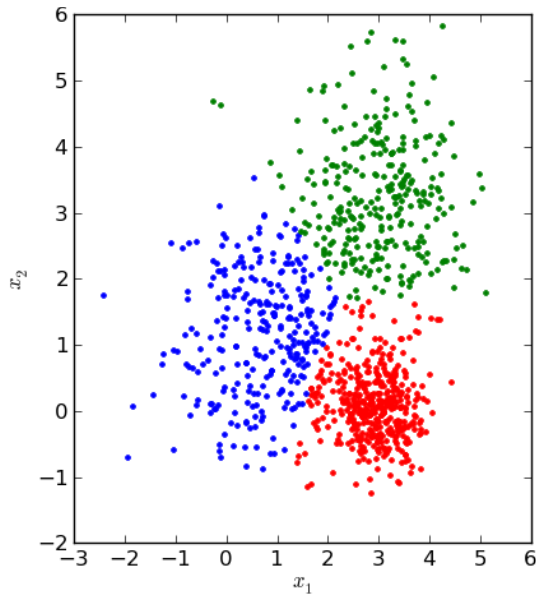
- This is clustering:

- Start with an unlabelled data set.
- After clustering, points are now assigned to clusters.
- There is no external annotator (e.g. human) defining or enforcing any labels.

Before clustering



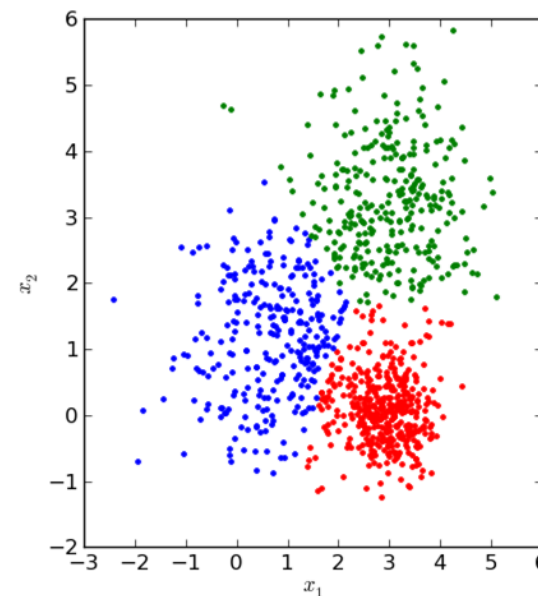
After clustering



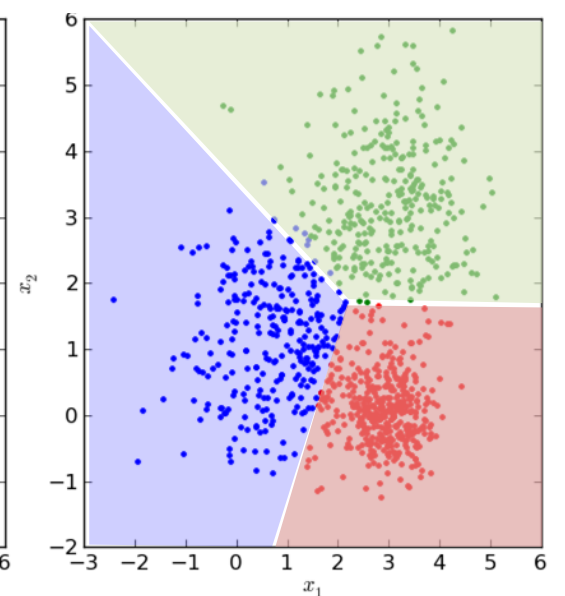
- This is classification:

- Start with a labelled data set.
- Labels are defined / enforced by an annotator (e.g. human).
- After classification, the model tries to predict how the annotator labelled the data.

Before classification

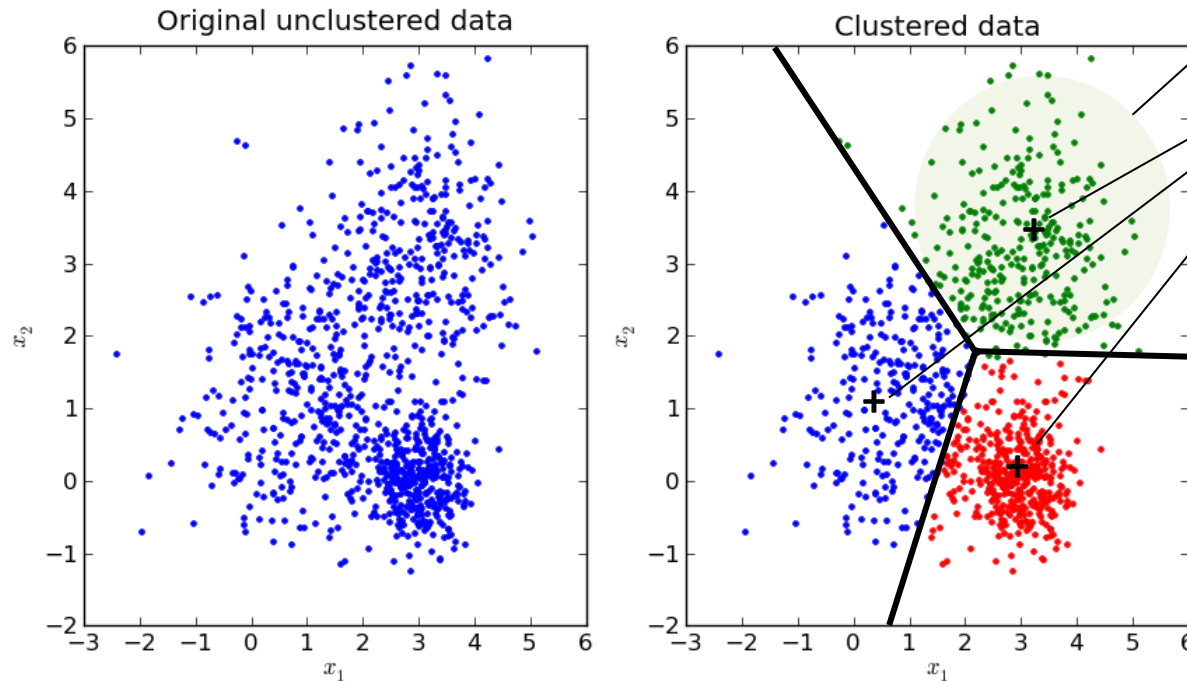


After classification



Side Note: Centroids and Voronoi Diagrams

- Typically, the result of clustering is a set of cluster **centroids**.
- If clusters can be visualized in 2-D, we can use **Voronoi Diagrams** to delineate data points that belong to different clusters.



Cell

A partitioned region of space.

Centroids

Center (mean) of each cluster cell.

Voronoi Diagram

- A **partitioning of space** into cells, based on a given position of centroids.
- Each separating line is the locus of points that are **equidistant** from the 2 nearest centroids.
- **Interpretation:**

All points within a cell are **closer to their assigned centroid** than any other centroid.

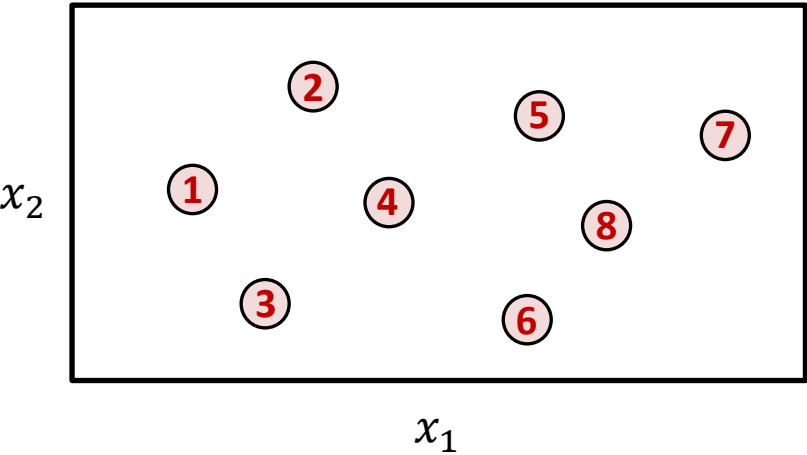
K-means Clustering

Let's begin with some preliminaries.

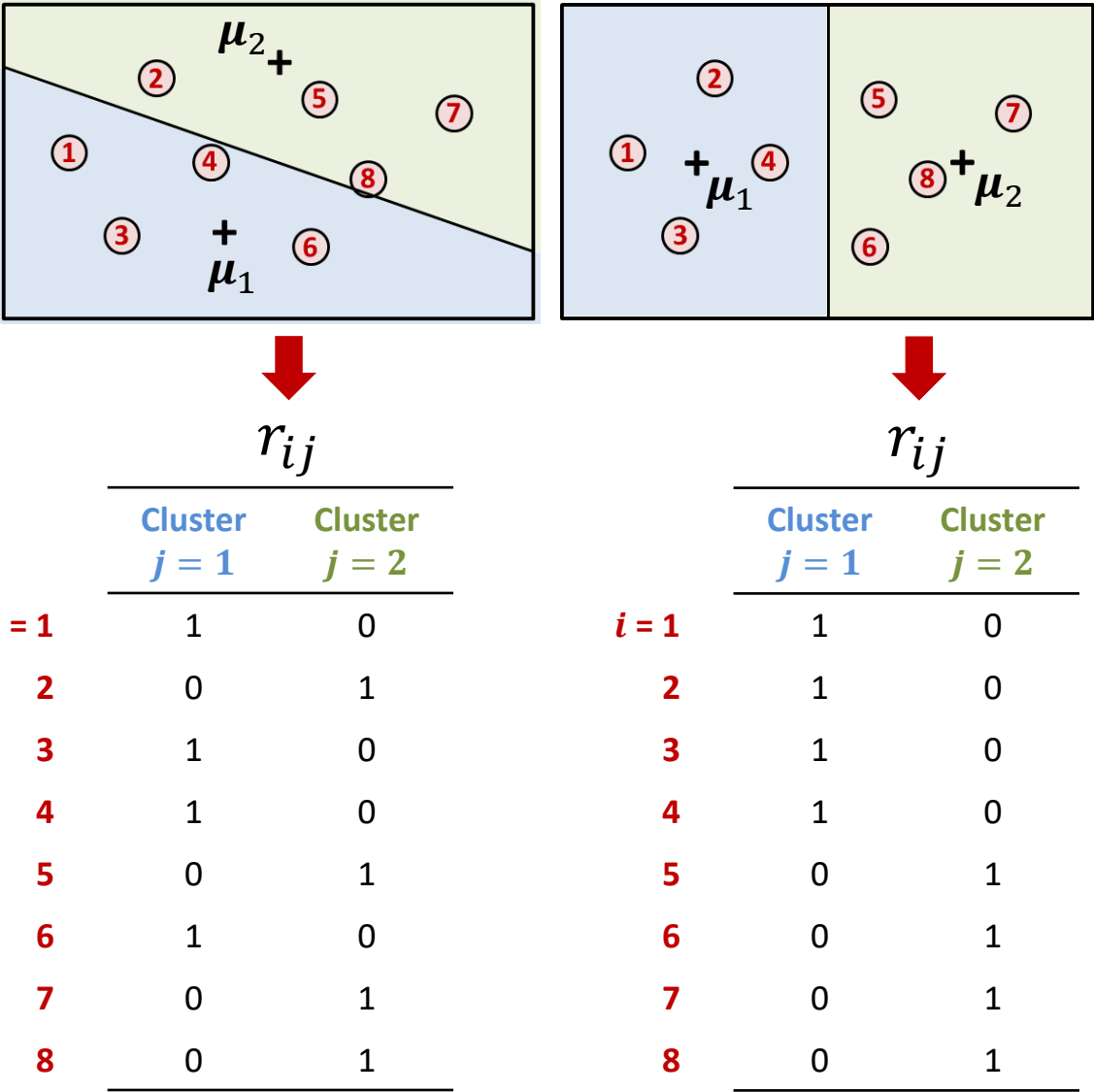
Let: $x_i \in \mathbb{R}^M, i = 1, 2, \dots, N$ be a data set of N samples
 $\mu_j \in \mathbb{R}^M, j = 1, 2, \dots, K$ be the locations of K centroids
 $r_{ij} \in \{0, 1\}, i = 1, 2, \dots, N$ be an indicator
 $j = 1, 2, \dots, K$

$$\begin{cases} r_{ij} = 1 & \text{if sample } i \text{ is closest to centroid } j \\ r_{ij} = 0 & \text{otherwise} \end{cases}$$

Consider the 2-D data set to be clustered below:



An illustration of how the indicator r_{ij} works: ($N = 8, K = 2$)

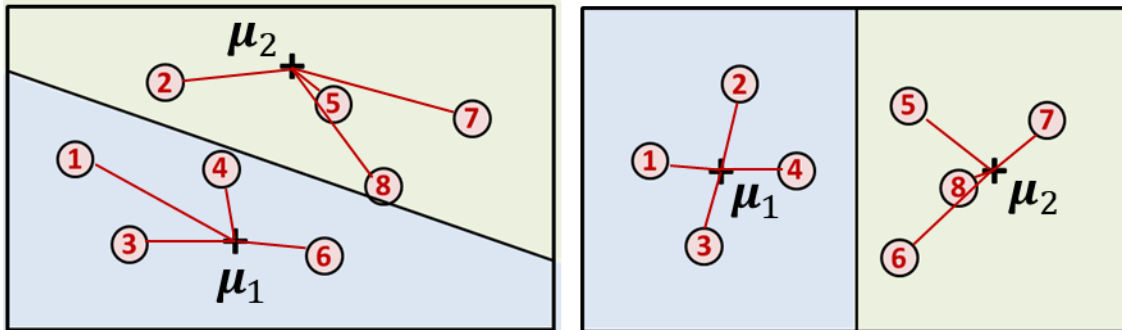


K-means Clustering

Let: $x_i \in \mathbb{R}^M$, $i = 1, 2, \dots, N$ be a data set of N samples
 $\mu_j \in \mathbb{R}^M$, $j = 1, 2, \dots, K$ be the locations of K centroids
 $r_{ij} \in \{0, 1\}$, $i = 1, 2, \dots, N$ be an indicator:
 $j = 1, 2, \dots, K$ $\begin{cases} r_{ij} = 1 & \text{if sample } i \text{ is closest to centroid } j \\ r_{ij} = 0 & \text{otherwise} \end{cases}$

In **K-means clustering**, the goal is to find K centroids, μ_j , that minimize the **inertia** or **distortion measure**, J :

$$J = \sum_i \sum_j r_{ij} \|x_i - \mu_j\|^2$$



Algorithm:

Given: Data Set, $x_i \in \mathbb{R}^M$, $i = 1, 2, \dots, N$

Initialization: Set a tolerance, tol (e.g. 10^{-3})
Set an assumed number of clusters, K
Set an initial guess of K centroids, μ_j

1: Pre-allocate an old solution: $\mu_j^{\text{old}} = \mathbf{0} \in \mathbb{R}^{K \times M}$.

2: **WHILE** $\|\mu_j - \mu_j^{\text{old}}\| > tol$

Expectation 3: Save the old centroids: $\mu_j^{\text{old}} := \mu_j$.

E-step 4: Compute r_{ij} : Assign each x_i to the closest centroid.

M-step 5: Compute the new centroids, μ_j :

Maximization

$$\mu_j = \frac{\sum_i r_{ij} x_i}{\sum_i r_{ij}}$$

Compute the new μ_j as the average position of all data points x_i assigned to cluster j .

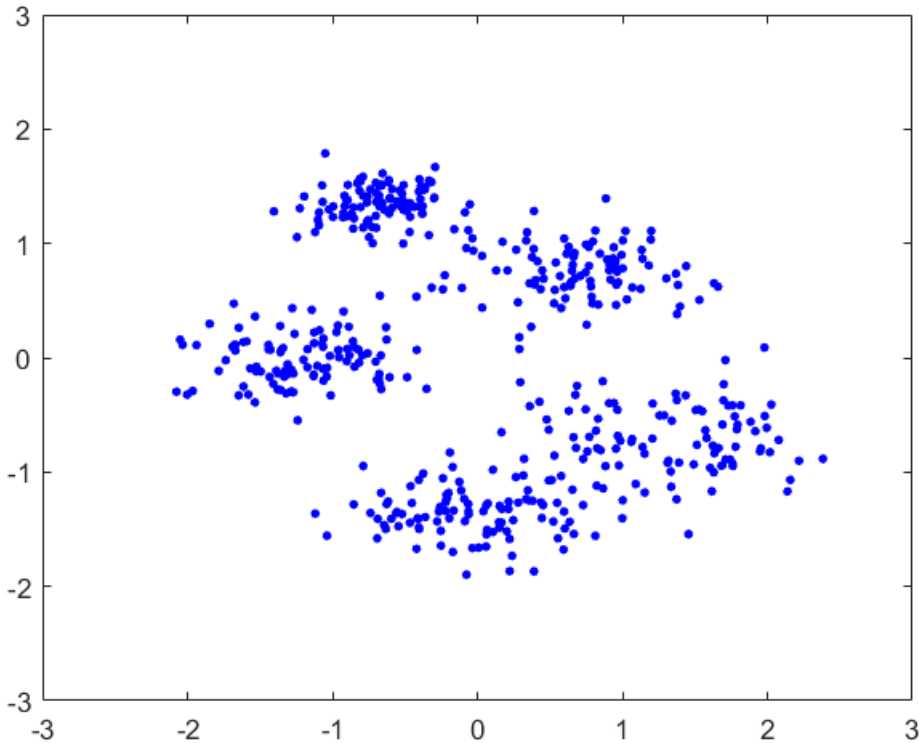
6: **END WHILE**

7: Report the final centroids, μ_j .

K-means Clustering

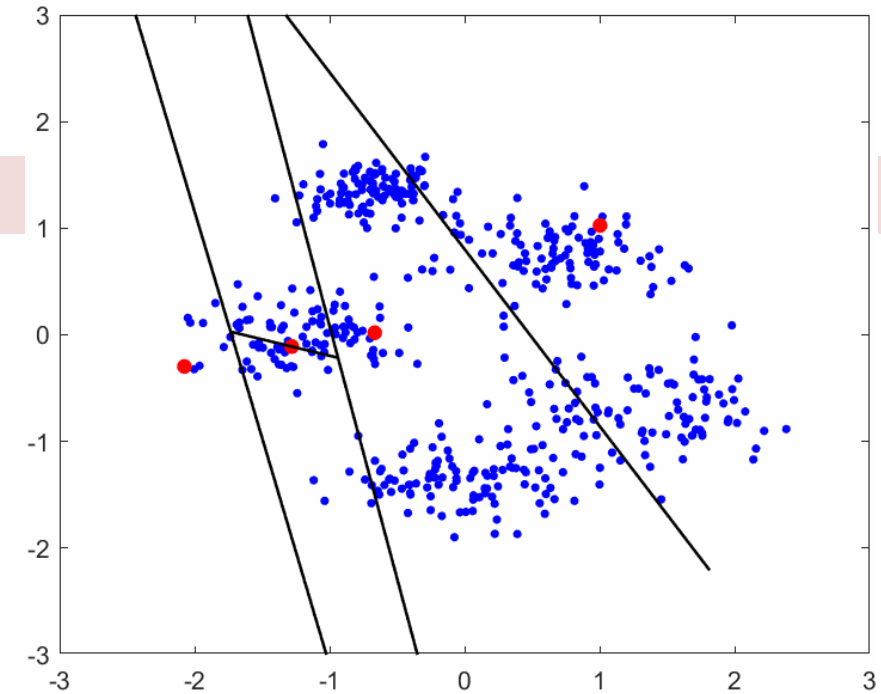
Example 1:

Find $K = 5$ clusters of samples, and their centroids, from the following data set:

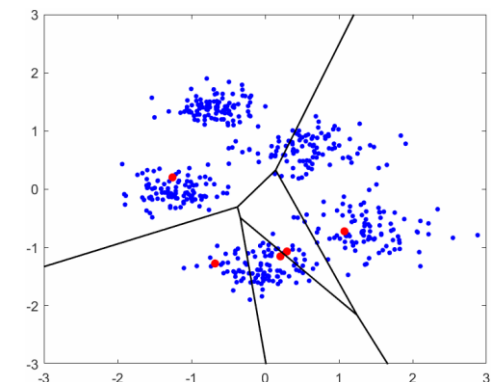
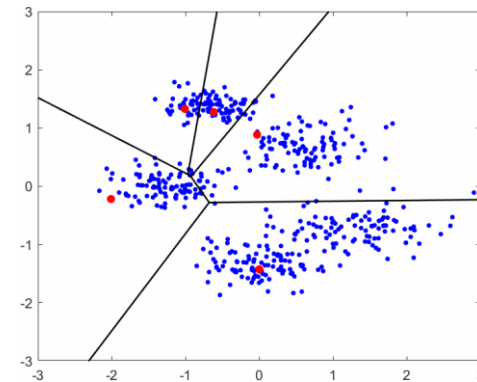


Answer:

```
( 1.3688, -0.7135)
(-0.6328,  1.2988)
(-0.0367, -1.3342)
( 0.7596,  0.7367)
(-1.2234, -0.0198)
```



However, note that different initializations may lead to local minima of J .
Solution: Try to restart K-means many times.



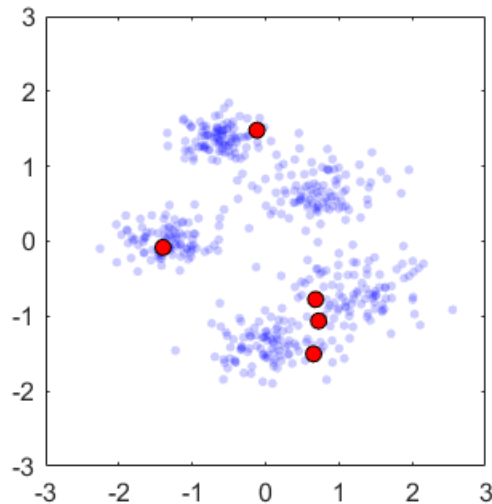
K-means Clustering

There are different ways to **initialize** the centroids in K-means clustering:

Forgy Initialization

Select K existing data points at random, then use them as initial guess of centroids.

sklearn: `cluster.Kmeans(...init='random')`

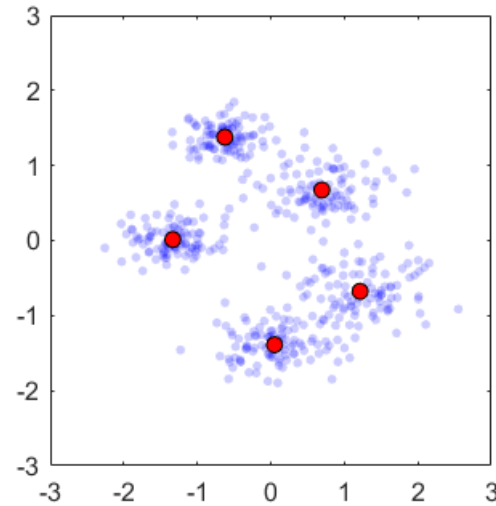


K-means++

Arthur and Vassilvitskii (2006).

Random data points are selected one at a time and as far apart as possible, until you have K number of them.

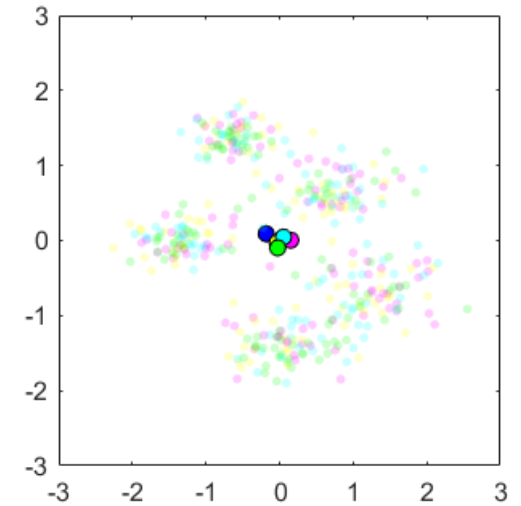
sklearn: `cluster.Kmeans(...init='k-means++')`



Random Partition

Assign each data point to a random cluster, then compute the means of each cluster as the initial guess.

***Not recommended**



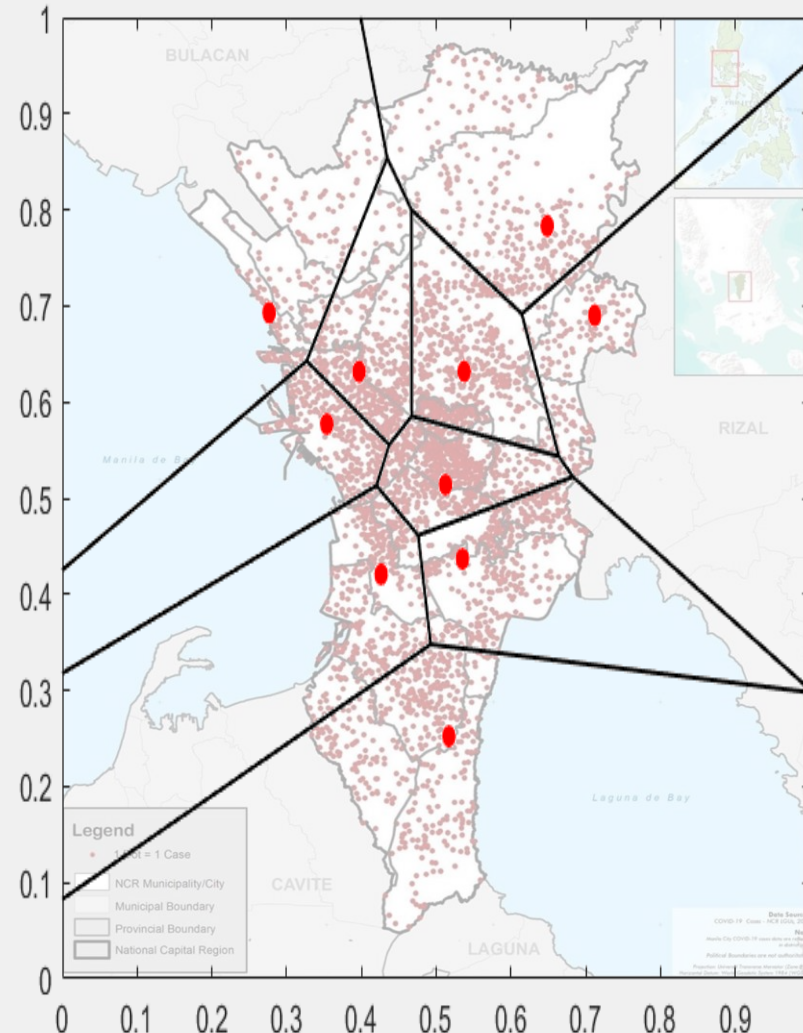
K-means Clustering

Example 2: Covid-19 Cases

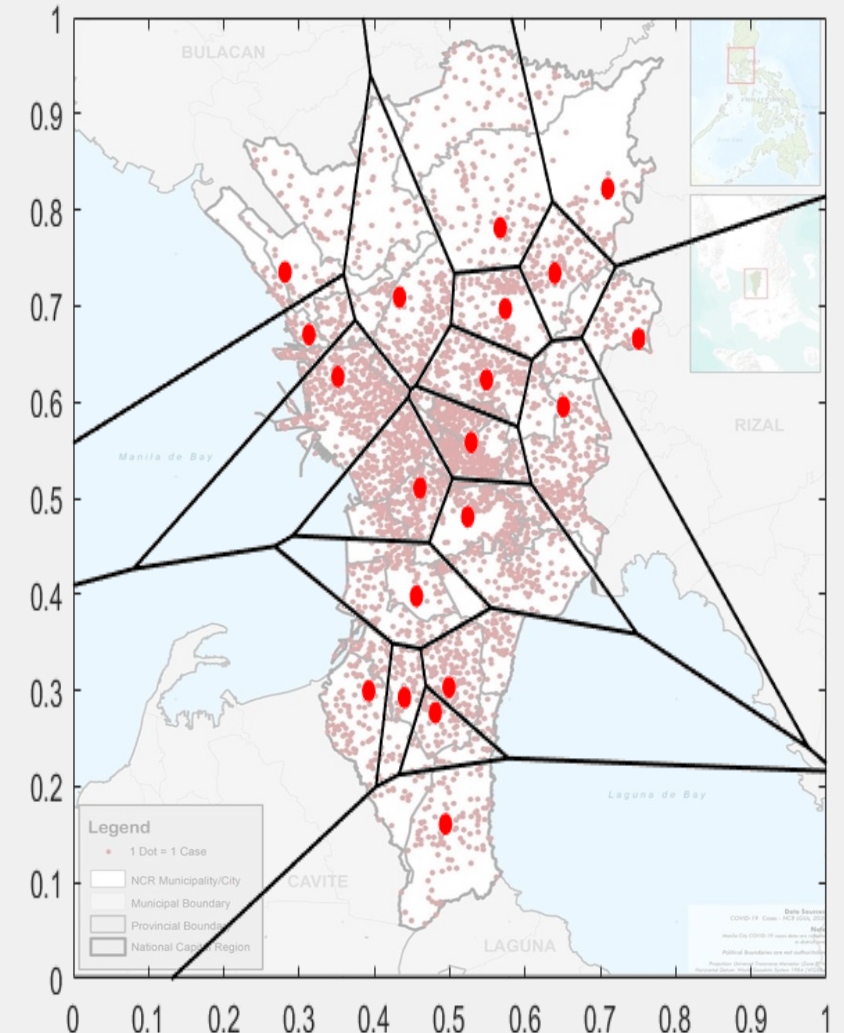
In this application, K-means clustering can be used to find K centroids where testing centers can be located.

Image of Covid cases from:
<https://www.esquiremag.ph/politics/news/covid-19-case-distribution-map-metro-manila-a00293-20200510>

Finding 10 clusters of NCR Covid-19 cases:



Finding 20 clusters of NCR Covid-19 cases:

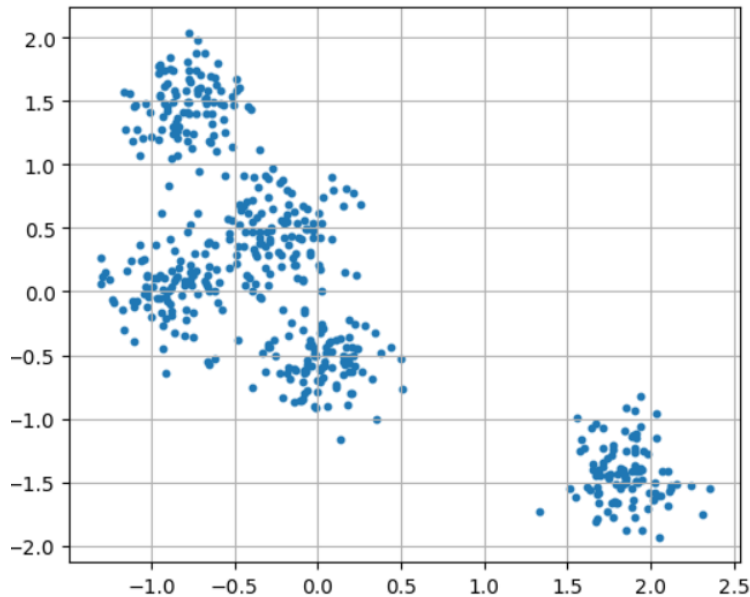


K-means Clustering

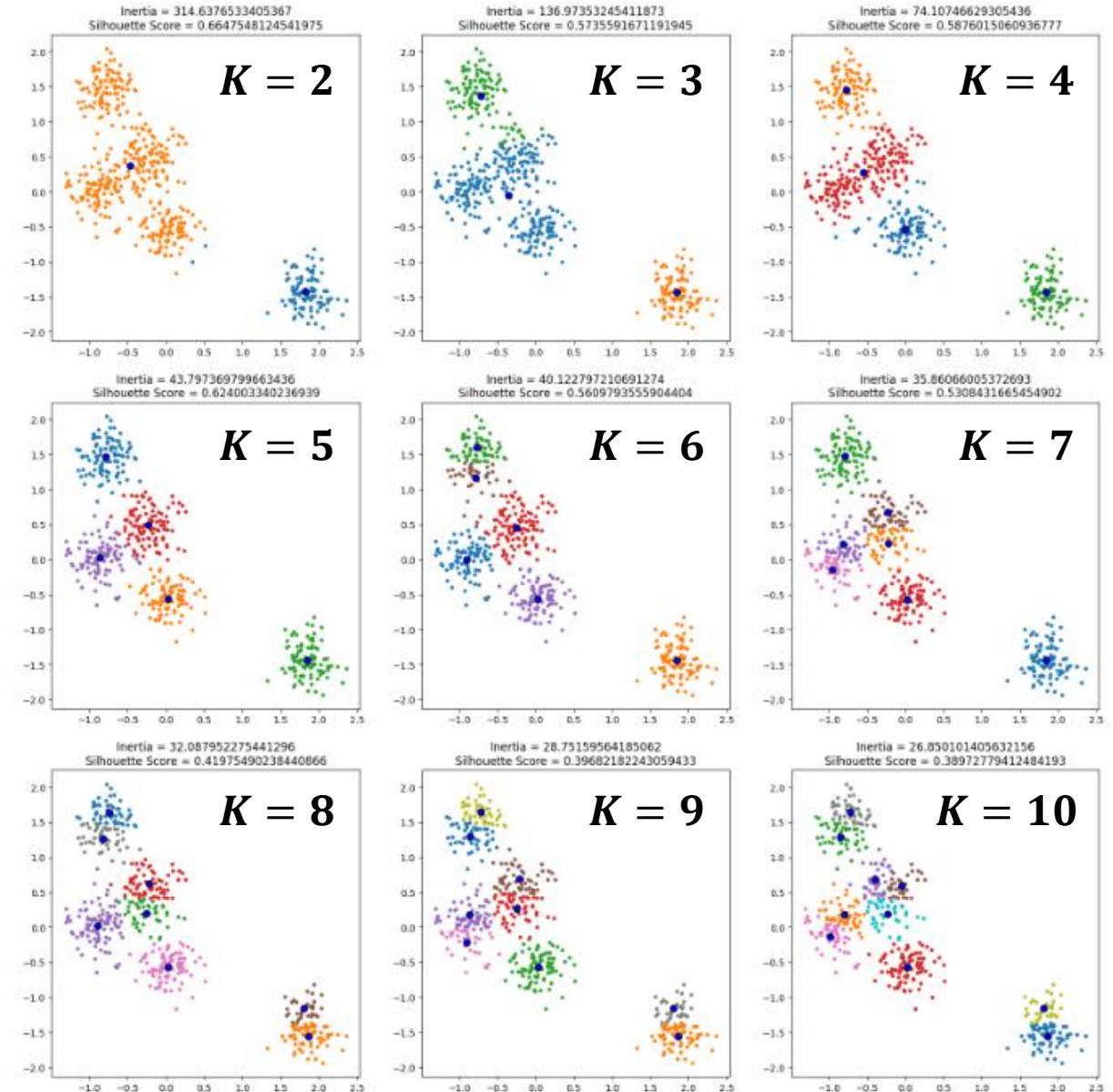
How to determine K in K-means?

- K-means cannot automatically learn the best number of clusters, K , from the data. We have to specify it ourselves prior to running it.
- Sometimes, the number of distinct clusters, K , is obvious (especially if data set is 2D). But this is not always the case.

How many clusters are in this data set?



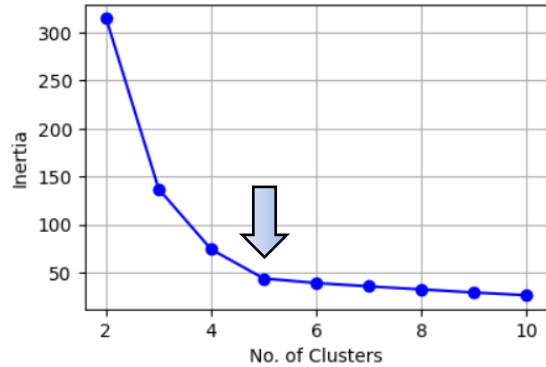
Results for various choices of K :



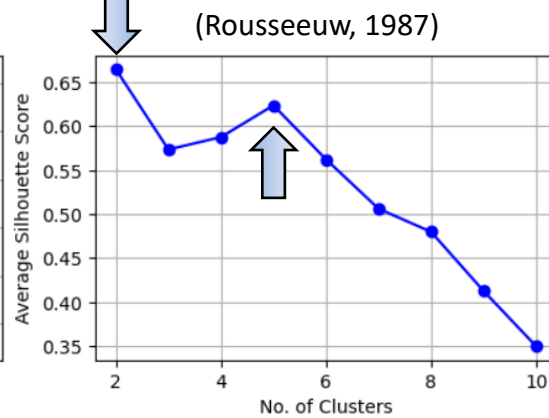
K-means Clustering

Results for various choices of K :

Elbow Method



Silhouette Score



- Calculate the inertia for each K :

$$J = \sum_i^N \sum_j^K r_{ij} \|x_i - \mu_j\|^2$$

- Find the **elbow point** in the inertia plot.

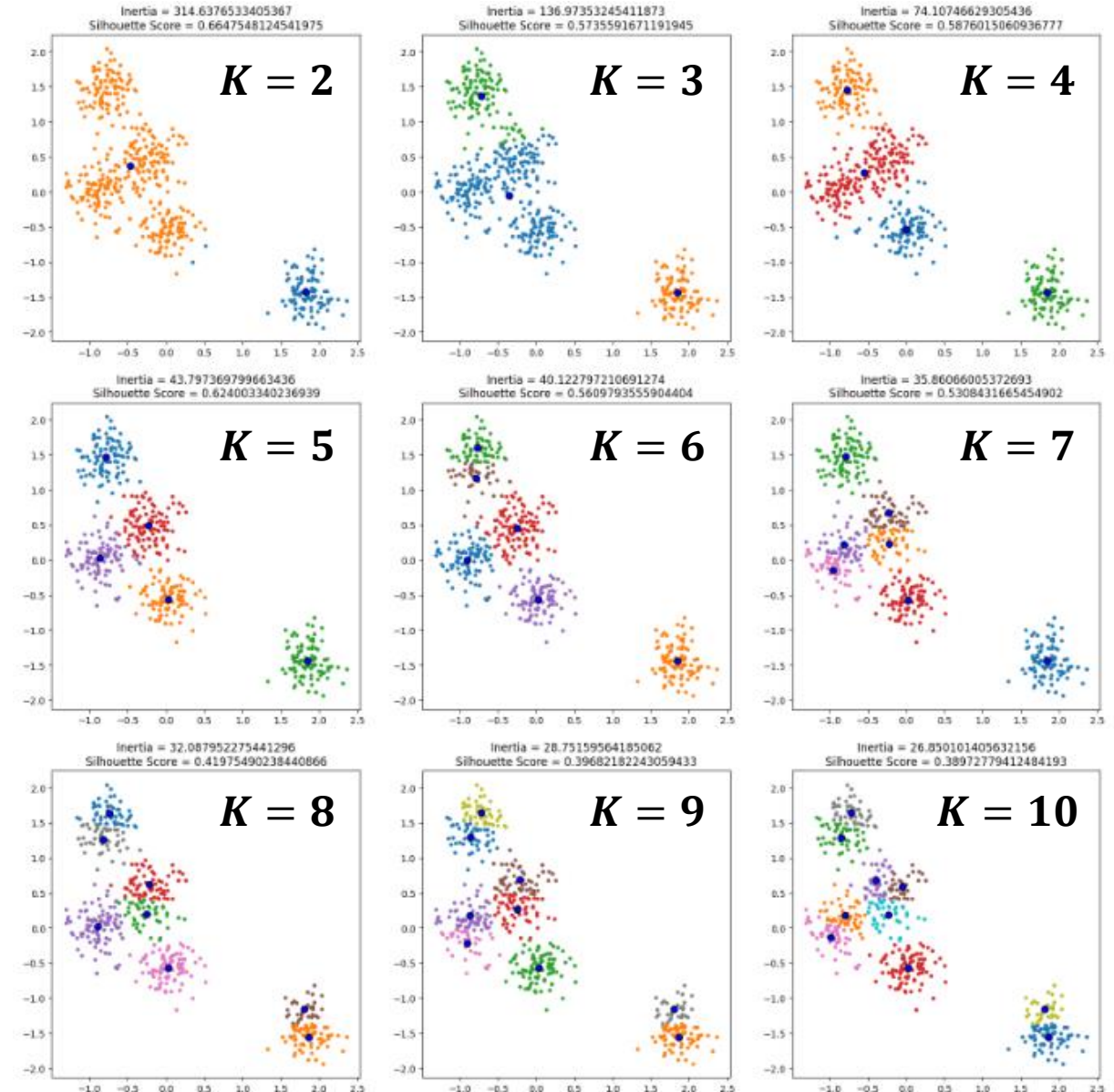
- For each K , calculate the average silhouette score:

$$s(i) = \begin{cases} \frac{b(i) - a(i)}{\max(a(i), b(i))}, & \text{if } |C_i| > 1 \\ 0, & \text{if } |C_i| = 1 \end{cases}$$

$$a(i) = \frac{1}{|C_i| - 1} \sum_{j \in C_i} d(i, j)$$

$$b(i) = \min_{k \neq i} \frac{1}{|C_k|} \sum_{j \in C_k} d(i, j)$$

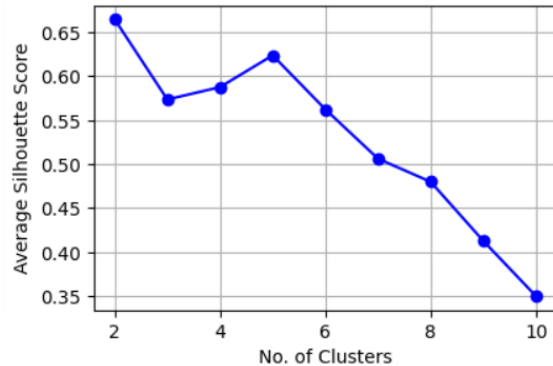
- Find the K with the **maximum** average silhouette score.



K-means Clustering

Silhouette Score

(Rousseeuw, 1987)



$$s(i) = \begin{cases} \frac{b(i) - a(i)}{\max(a(i), b(i))}, & \text{if } |C_i| > 1 \\ 0, & \text{if } |C_i| = 1 \end{cases}$$

$$a(i) = \frac{1}{|C_i| - 1} \sum_{j \in C_i} d(i, j)$$

$$b(i) = \min_{k \neq i} \frac{1}{|C_k|} \sum_{j \in C_k} d(i, j)$$

Notation:

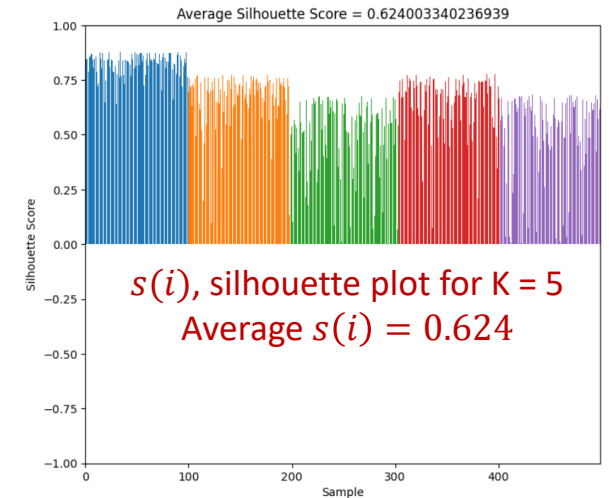
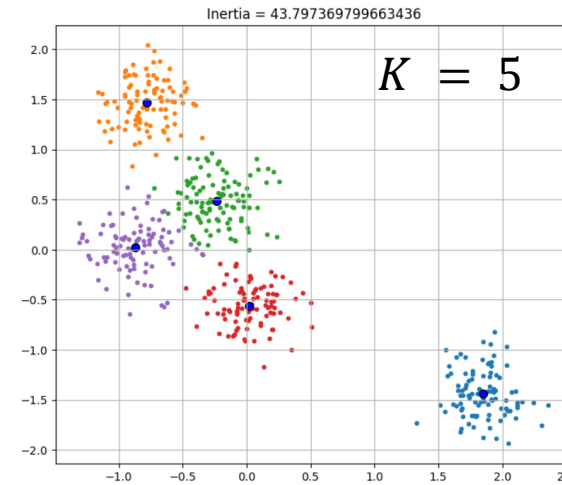
$d(i, j)$ = Euclidean distance of point i to point j .

$a(i)$ = mean distance between point i to all other points in the same cluster (**cohesion**).

$b(i)$ = mean distance between point i to all other points in the *next nearest* cluster (**separation**).

$|C_i|$ = no. of points assigned to cluster i .

$s(i)$ = silhouette score for point i . It ranges from -1 to 1 only.



Other performance metrics for clustering:

Internal Cluster Validity Indices

(Correct labels are *unknown*)

- Calinski-Harabasz Index
- Davies-Bouldin Index
- Dunn Index
- Negentropy Increment
- C-index

External Cluster Validity Indices

(Correct labels are *known*)

- Rand
- Adjusted Rand
- Fowlkes-Mallows
- Jaccard
- Variation of Information

See Arbelaitz, O., Gurrutxaga, I., Muguerza, J., Perez, J. M., & Perona, I. (2013). An extensive comparative study of cluster validity indices. *Pattern Recognition*, 46(1), 243–256.

K-means Clustering

Some **drawbacks** of K-means clustering:

1. The converged solution is **not always** optimal.

$$\min_{\mu} J = \sum_i^N \sum_j^K r_{ij} \|x_i - \mu_j\|^2$$

2. If the initial centroids are poorly chosen, some clusters may end up **empty**.
3. If number of data points is large and high-dimensional, the algorithm can take **a long time** to run.

Some **workarounds**:

- Use a tree data structure to compute distances faster (Moore, 2000).
- Use the triangle inequality instead of Euclidean distance (Elkan, 2003).

Some **extensions** of K-means clustering:

1. Instead of the Euclidean distance, we can use a different **notion of distance**, $\mathcal{V}(x_i, \mu_j)$. This leads to the K-medoids algorithm.

$$\min_{\mu} J = \sum_i^N \sum_j^K r_{ij} \mathcal{V}(x_i, \mu_j)$$

2. Instead of hard clustering (each x_i is assigned to one and only one cluster), we can use **soft clustering** algorithms.

Fuzzy C-means Clustering

Fuzzy clustering by Local Approximation of MEMberships (FLAME)

Rough K-means Clustering

Gaussian Mixture Models

3. So far, we only have a **batch** version of K-means (the entire data set is used at once). Alternatively, we can use **MiniBatch** K-means.

*We can also implement K-means clustering **online**: The clusters are updated as new data points arrive.*

Hierarchical Clustering

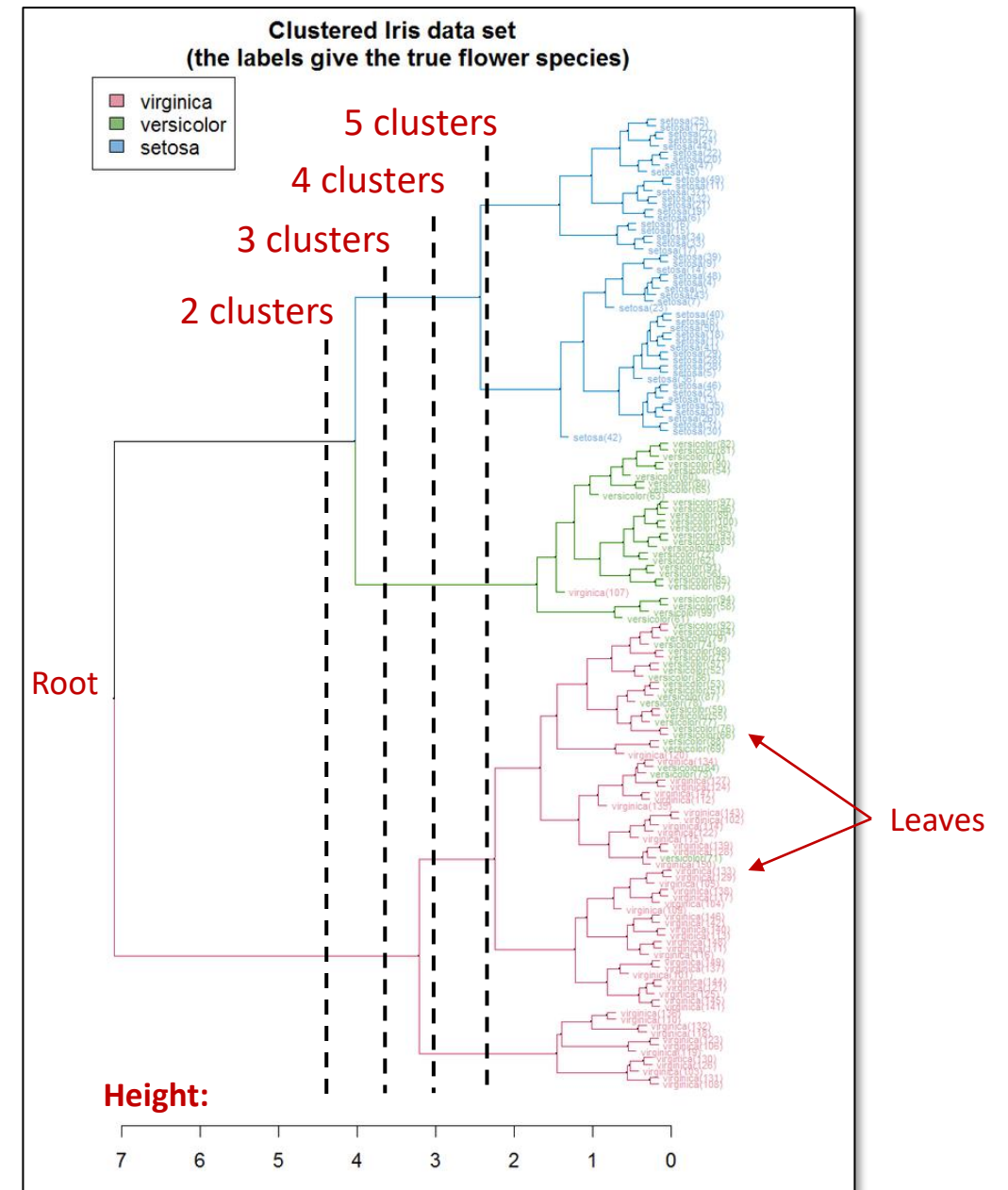
The K-means algorithm only works if we know the number of clusters, K , beforehand.

How can we view all results for any K ?

One solution: Hierarchical Clustering
(Hierarchical Cluster Analysis)

The main result of Hierarchical Clustering is a **dendrogram**.

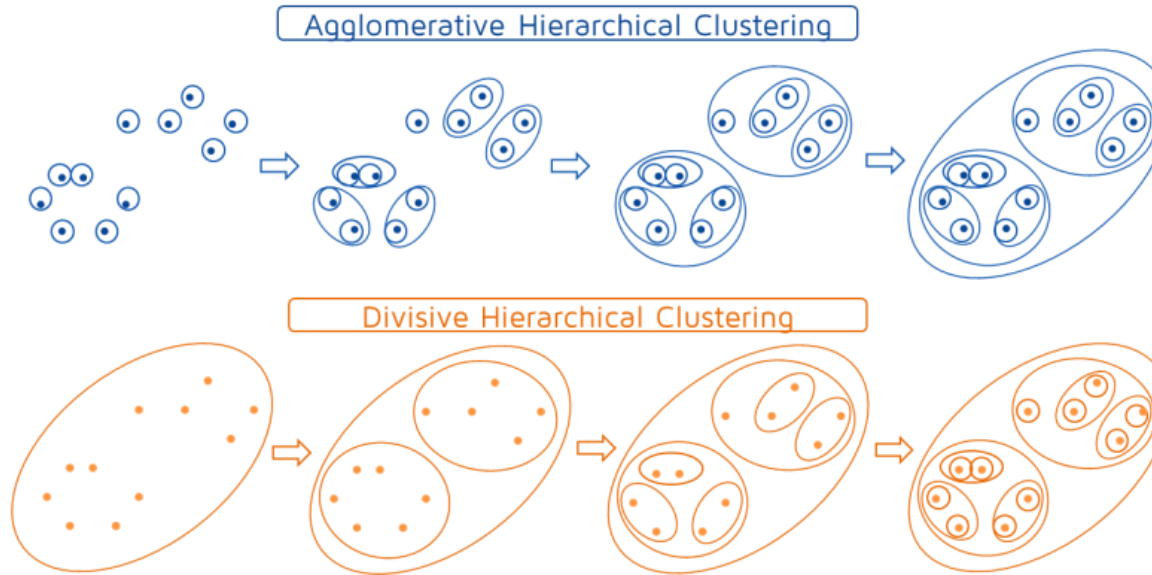
- It is a binary tree.
- The **root** represents all the data as one cluster.
- Each **leaf** represents each data point as is its own cluster.
- Onwards from the root, only one new split occurs at a time. When a split is made, two separate clusters are formed.
- The tree can be cut at any height to give any number of clusters.



https://en.wikipedia.org/wiki/Hierarchical_clustering#/media/File:Iris_dendrogram.png

Hierarchical Clustering

There are two main strategies to build the dendrogram.



<https://quantdare.com/hierarchical-clustering/>

Agglomerative Clustering

- Bottom-up approach
- **Algorithm:**
 1. Treat each data point as a cluster (N clusters)
 2. WHILE (No. of clusters **is not** equal to 1)
 3. Combine the 2 nearest clusters.
 4. END
 5. Report the result as a dendrogram.

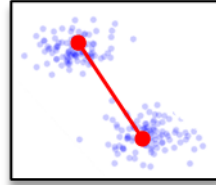
Divisive Clustering (Bisecting K-means)

- Top-down approach
- **Algorithm:**
 1. Treat the entire data set as 1 cluster.
 2. WHILE (No. of clusters **is not** equal to N)
 3. Identify new clusters using K-means.
 4. END
 5. Report the result as a dendrogram.

Hierarchical Clustering

Different ways to **merge** clusters:

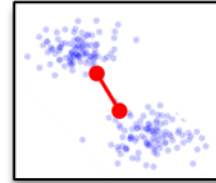
- Merges are determined in a **greedy** manner.
- Two clusters are merged if they are the “closest” pair to each other.
- The term “closest” can be defined in many ways:
 - Centroid-linkage
 - Single-linkage
 - Complete-linkage
 - Average-linkage
 - Ward’s method



Centroid-linkage

- Two clusters are “closest” if their centroids are closest.

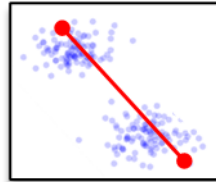
$$D(c_1, c_2) = D\left(\frac{1}{|c_1|} \sum_{x \in c_1} x, \frac{1}{|c_2|} \sum_{x \in c_2} x\right)$$



Single-linkage (SLINK)

- Two clusters are “closest” if the two nearest points between them is closest.
- Favors long chains of clusters.

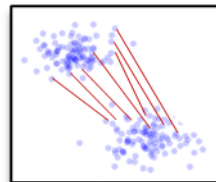
$$D(c_1, c_2) = \min_{x_1 \in c_1, x_2 \in c_2} D(x_1, x_2)$$



Complete-linkage (CLINK)

- Two clusters are “closest” if the two farthest points between them is closest.
- Favors spherical clusters with consistent diameter.

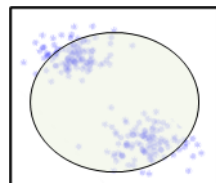
$$D(c_1, c_2) = \max_{x_1 \in c_1, x_2 \in c_2} D(x_1, x_2)$$



Average-linkage

- Two clusters are “closest” if the average distance between all pairs of points is closest.
- Less affected by outliers.

$$D(c_1, c_2) = \frac{1}{|c_1|} \frac{1}{|c_2|} \sum_{x_1 \in c_1} \sum_{x_2 \in c_2} D(x_1, x_2)$$



Ward's Method

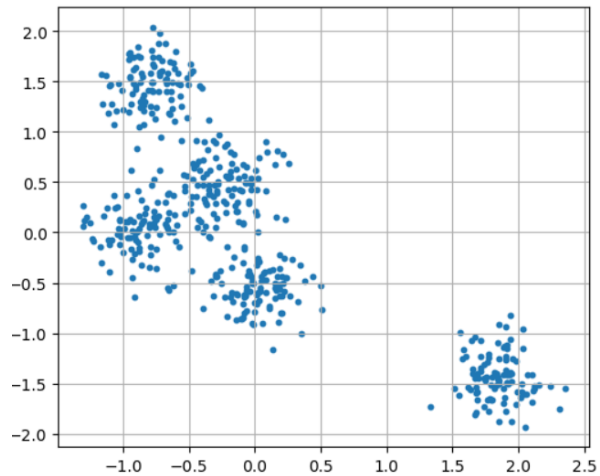
- Two clusters are merged if there is a minimum increase in total within-cluster variance after merging.

$$D(c_1, c_2) = \sum_{x \in c_1 \cup c_2} D(x, \mu_{c_1 \cup c_2})^2$$

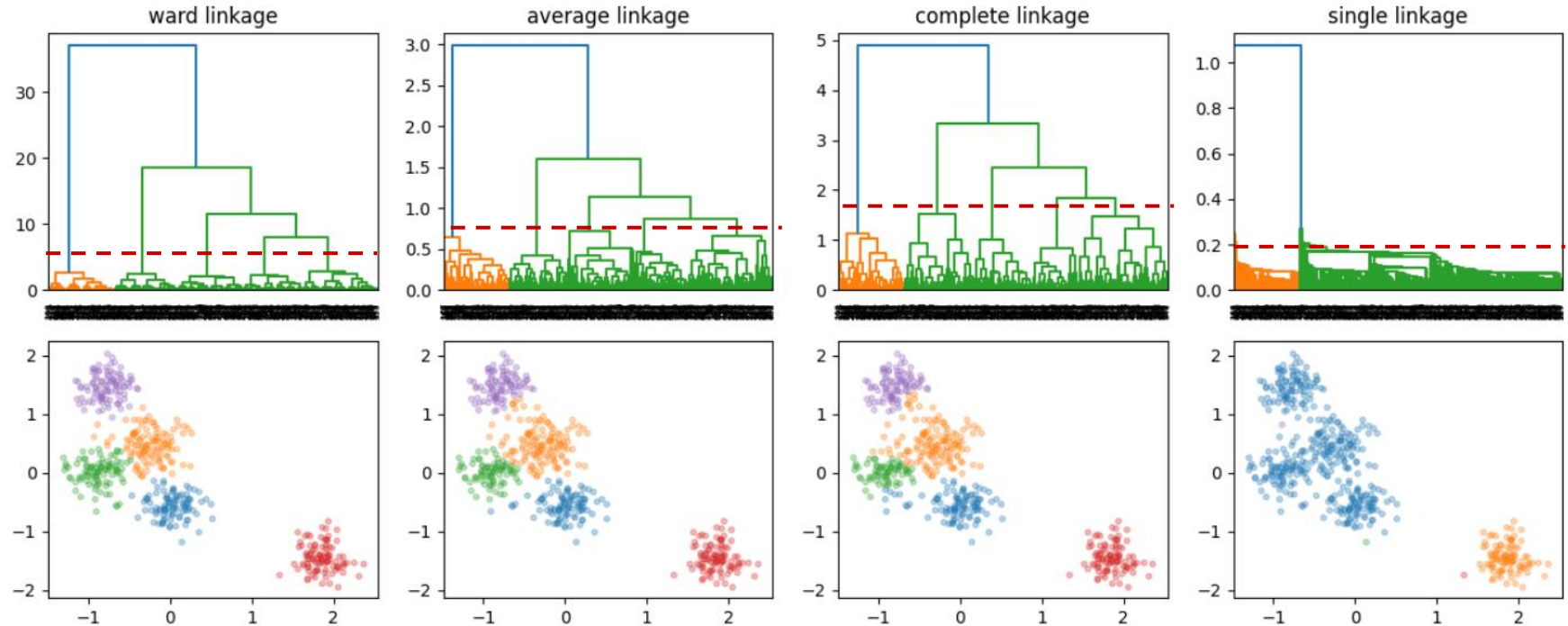
Hierarchical Clustering

In our example, Ward's method shows the most intuitive dendrogram.

Ward's method is the *default* method for Agglomerative Clustering in `sklearn`.



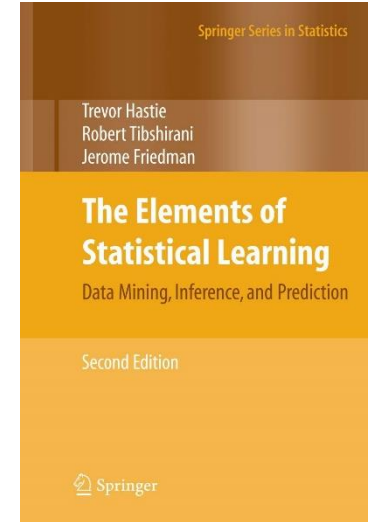
Dendrograms (top) and Cluster Visualizations (bottom)



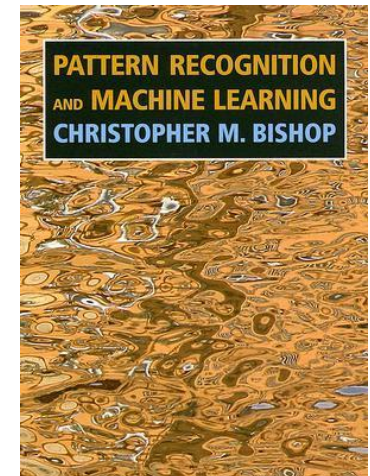
Outline

- Clustering
 - K-means Clustering
 - Hierarchical Clustering
 - Density-based Clustering
 - Gaussian Mixture Model
- Density Estimation
 - Kernel Density Estimation (KDE)
- Anomaly Detection
 - Application of KDE
 - One-class SVM
 - Local Outlier Factor
 - Isolation Forest

Hastie *et al.* (2008)
The Elements of Statistical Learning.
2nd Ed. Springer.



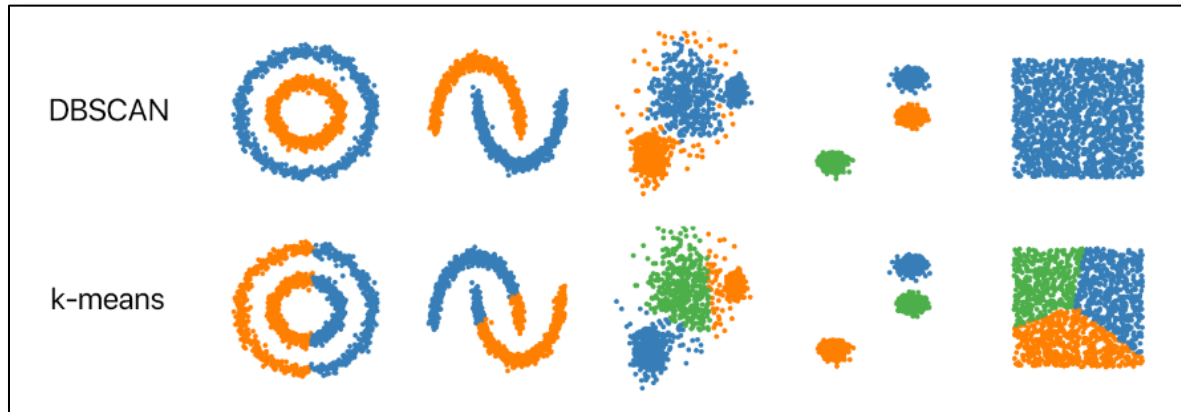
Bishop (2006)
*Pattern Recognition and
Machine Learning.* Springer.



Density-based Clustering

Two famous algorithms for density-based clustering are DBSCAN and OPTICS.

Difference between DBSCAN and K-means Clustering:



<https://www.mygreatlearning.com/blog/dbscan-algorithm/>

K-means Clustering

- Distance-based
- Cluster shapes tend to be **spherical** around the centroids.
- Each point will eventually belong to a cluster
- Sensitive to outliers

DBSCAN

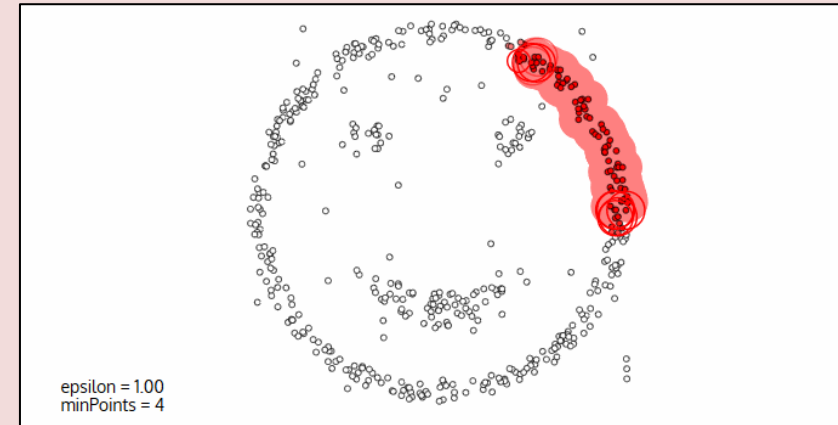
- Density-based
- Clusters can be **any shape** of connected data points.
- Some points will be un-clustered (outliers)
- Robust to outliers

DBSCAN

(Density-based Spatial Clustering of Applications with Noise)

This illustrates DBSCAN on a toy data set:

(<https://www.kdnuggets.com/2020/04/dbscan-clustering-algorithm-machine-learning.htm>)



DBSCAN parameters: minPts and Epsilon (ϵ)

If there are at least 'minPts' points within a radius of ' ϵ ' to a point, then we consider all these points to be part of the same cluster.

Gaussian Mixture Models

- A soft-clustering version of K-means.
- Assumes that each cluster is Gaussian-distributed.
- The entire data set is then clustered as a mixture of K Gaussian distributions. K must be known.

$$p(\mathbf{x}) = \sum_{k=1}^K \pi_k \mathcal{N}(\mathbf{x} | \boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)$$

π_k = weight of k th cluster
 K = total number of clusters
 $\mathcal{N}(\mathbf{x} | \boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)$ = k th Gaussian dist.

Recall: K-means clustering algorithm

Given: Data Set, $\mathbf{x}_i \in \mathbb{R}^M$, $i = 1, 2, \dots, N$

Initialization: Set a tolerance, tol (e.g. 10^{-3})
 Set an assumed number of clusters, K
 Set an initial guess of K centroids, $\boldsymbol{\mu}_j$

- 1: Pre-allocate an old solution: $\boldsymbol{\mu}_j^{\text{old}} = \mathbf{0} \in \mathbb{R}^{K \times M}$.
- 2: **WHILE** $\|\boldsymbol{\mu}_j - \boldsymbol{\mu}_j^{\text{old}}\| > tol$
- 3: Save the old centroids: $\boldsymbol{\mu}_j^{\text{old}} := \boldsymbol{\mu}_j$.
- 4: Compute r_{ij} : Assign each \mathbf{x}_i to the closest centroid.
- 5: Compute the new centroids, $\boldsymbol{\mu}_j$:

$$\boldsymbol{\mu}_j = \frac{\sum_i r_{ij} \mathbf{x}_i}{\sum_i r_{ij}}$$

Compute the new $\boldsymbol{\mu}_j$ as the average position of all data points \mathbf{x}_i assigned to cluster j .

6: **END WHILE**

7: Report the final centroids, $\boldsymbol{\mu}_j$.

The algorithm for GMMs is similar to K-means.

EM = Expectation Maximization
 (Dempster et al., 1997)

- **E-step:** Calculate the responsibilities, r_{ij} , of each \mathbf{x} to each $\boldsymbol{\mu}$.

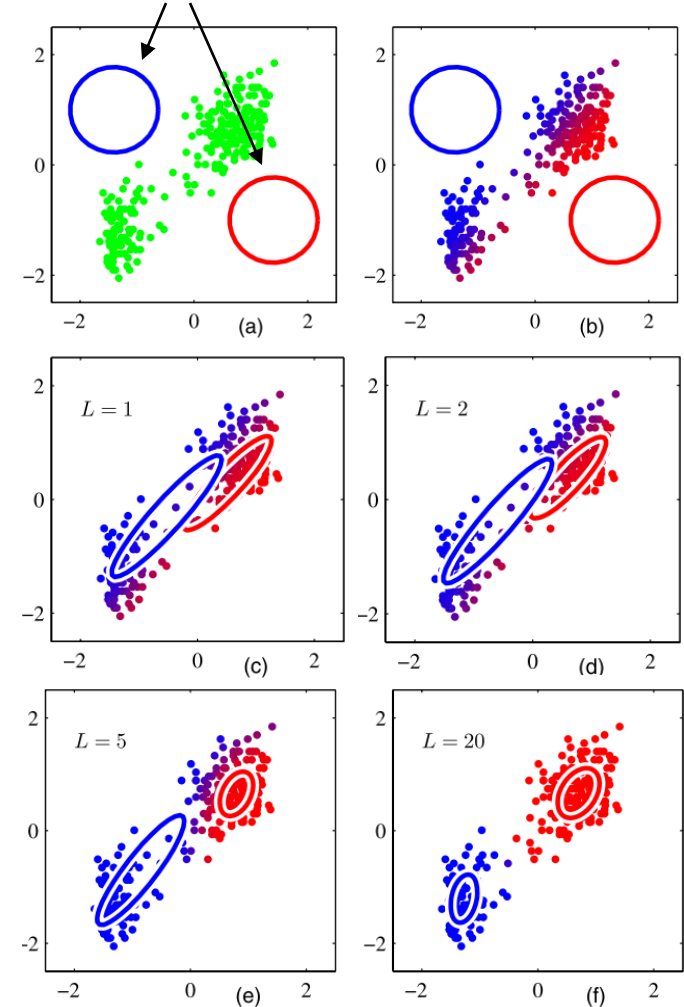
$$r_{ij} = \frac{\pi_j \mathcal{N}(\mathbf{x}_i | \boldsymbol{\mu}_j, \boldsymbol{\Sigma}_j)}{\sum_{k=1}^K \pi_k \mathcal{N}(\mathbf{x}_i | \boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)}$$

- **M-step:** Update new $\boldsymbol{\mu}$ and $\boldsymbol{\Sigma}$.

$$\boldsymbol{\mu}_j = \frac{\sum_i r_{ij} \mathbf{x}_i}{\sum_i r_{ij}} \quad \boldsymbol{\Sigma}_j = \frac{1}{N_j} \sum_{n=1} r_{nj} (\mathbf{x}_n - \boldsymbol{\mu}_j)^2$$

How the EM algorithm works:

2-D Gaussian PDFs
 $\mathcal{N}(\mathbf{x} | \boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k)$



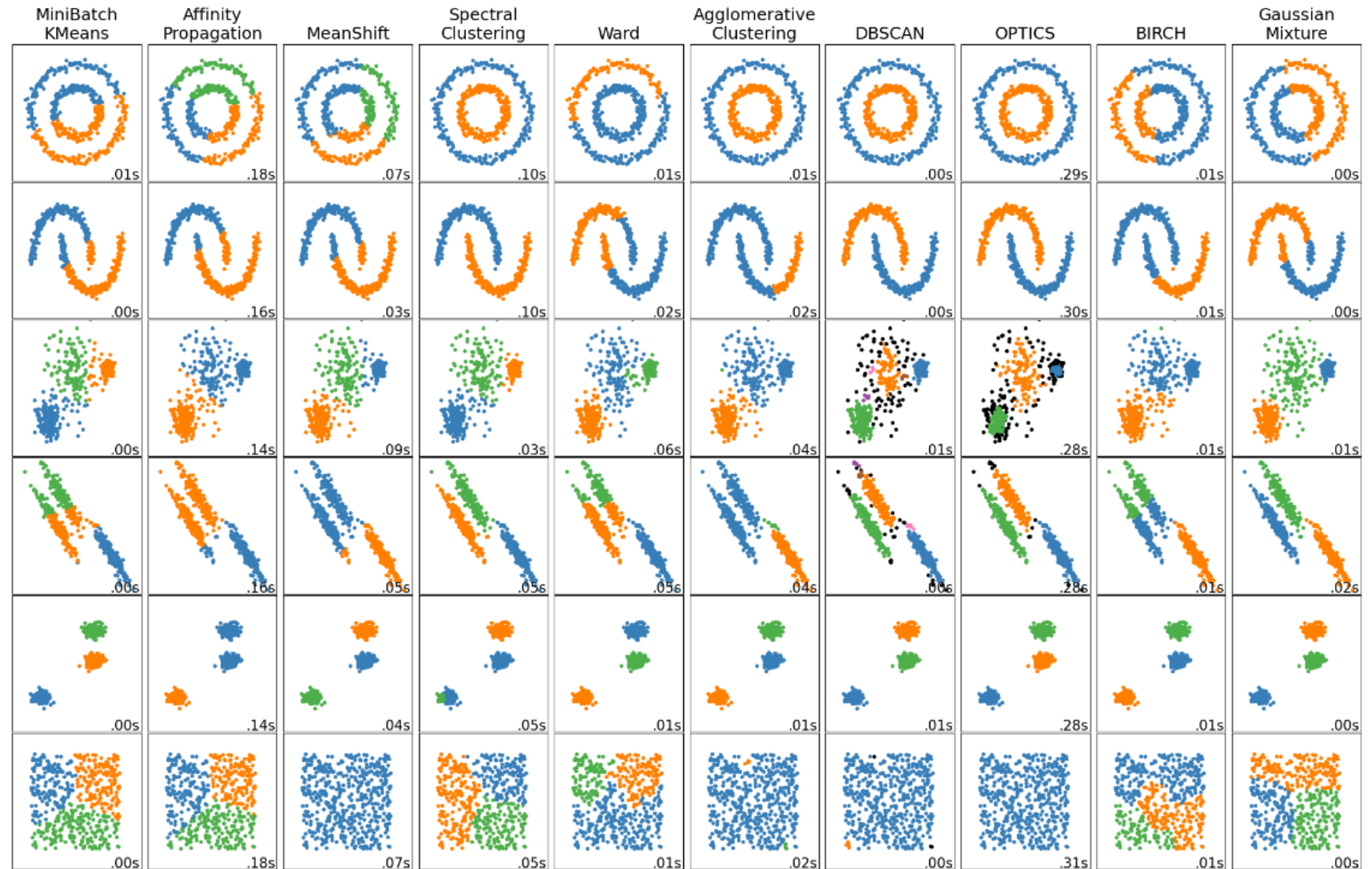
A Comparison of Clustering Methods

So far, we discussed only:

- K-means
- Mini-batch K-means
- Hierarchical Clustering
 - Ward
- DBSCAN
- Gaussian Mixture

In scikit-learn, other clustering algorithms are available.

You can apply a performance metric (e.g. silhouette score) on all these results in order to compare them quantitatively.

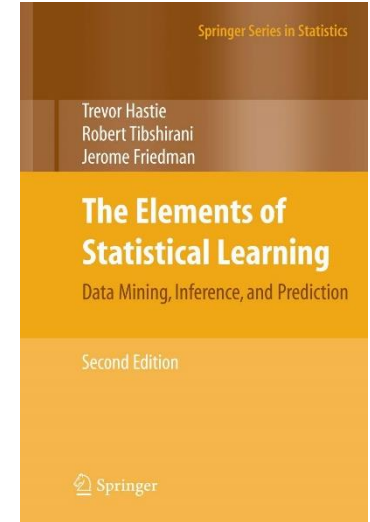


A comparison of the clustering algorithms in scikit-learn

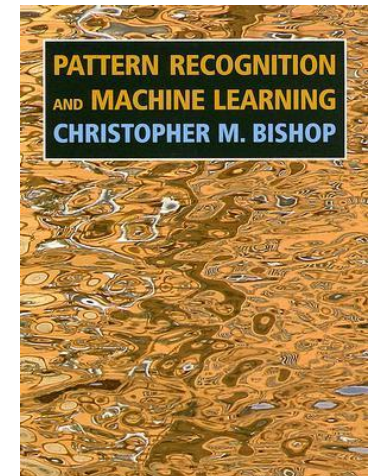
Outline

- Clustering
 - K-means Clustering
 - Hierarchical Clustering
 - Density-based Clustering
 - Gaussian Mixture Model
- **Density Estimation**
 - **Kernel Density Estimation (KDE)**
- **Anomaly Detection**
 - **Application of KDE**
 - **One-class SVM**
 - **Local Outlier Factor**
 - **Isolation Forest**

Hastie *et al.* (2008)
The Elements of Statistical Learning.
2nd Ed. Springer.



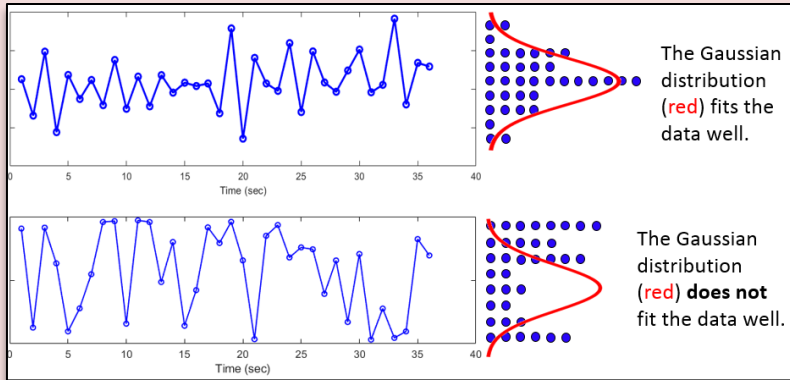
Bishop (2006)
*Pattern Recognition and
Machine Learning.* Springer.



Density Estimation

- Refers to the estimation of an unobservable underlying **probability density function** given only observed data.
- Usually, we only assume that the data is Gaussian-distributed. However, if the data is **multi-modal** or **non-Gaussian**, then we need to perform density estimation.

Gaussian vs. Non-Gaussian Data



Kernel Density Estimation

Estimate the probability, $p(\mathbf{x})$, as a sum of kernel functions $K(\mathbf{x})$:

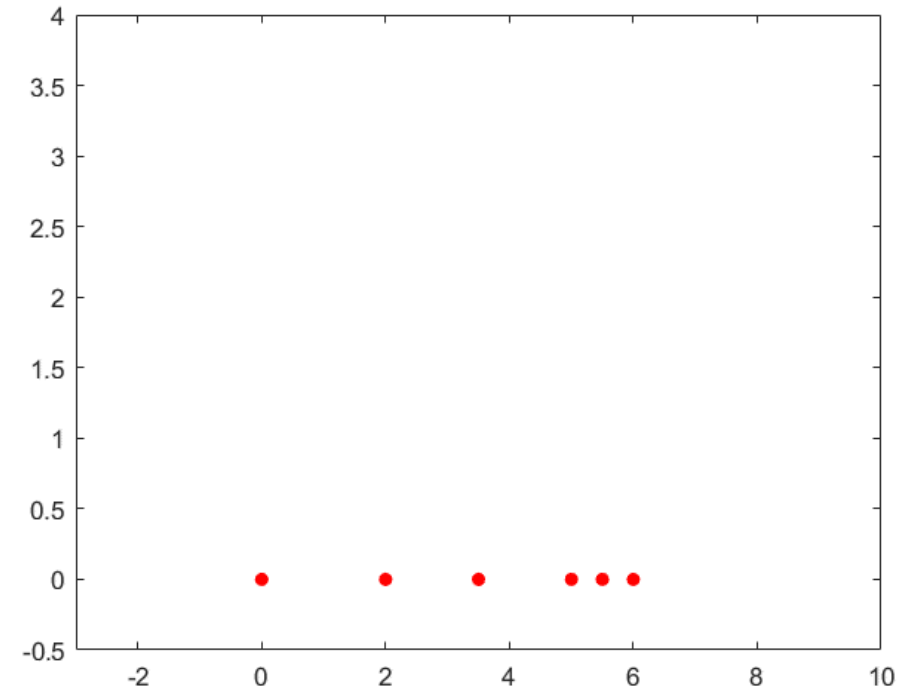
$$p(\mathbf{x}) = \frac{1}{N} \sum_{i=1}^N \frac{1}{h^D} K\left(\frac{\mathbf{x} - \mathbf{x}_i}{h}\right)$$

h = bandwidth
 D = dimensions of the data
 N = no. of data points
 $k(\cdot)$ = kernel function
 \mathbf{x}_i = training data
 \mathbf{x} = any new query data

Example:

Estimate the probability distribution of the following univariate non-Gaussian data:

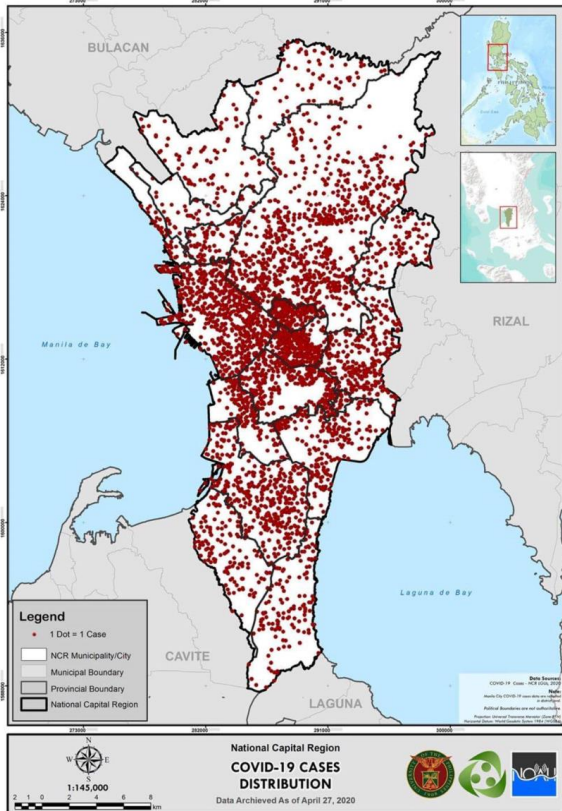
0.0,
2.0,
3.5,
5.0,
5.5,
6.0



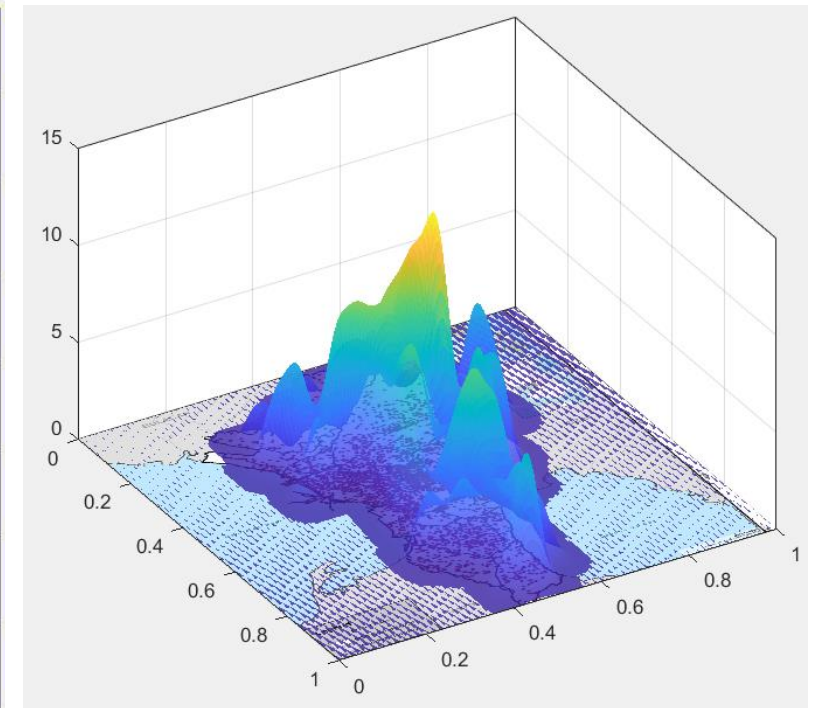
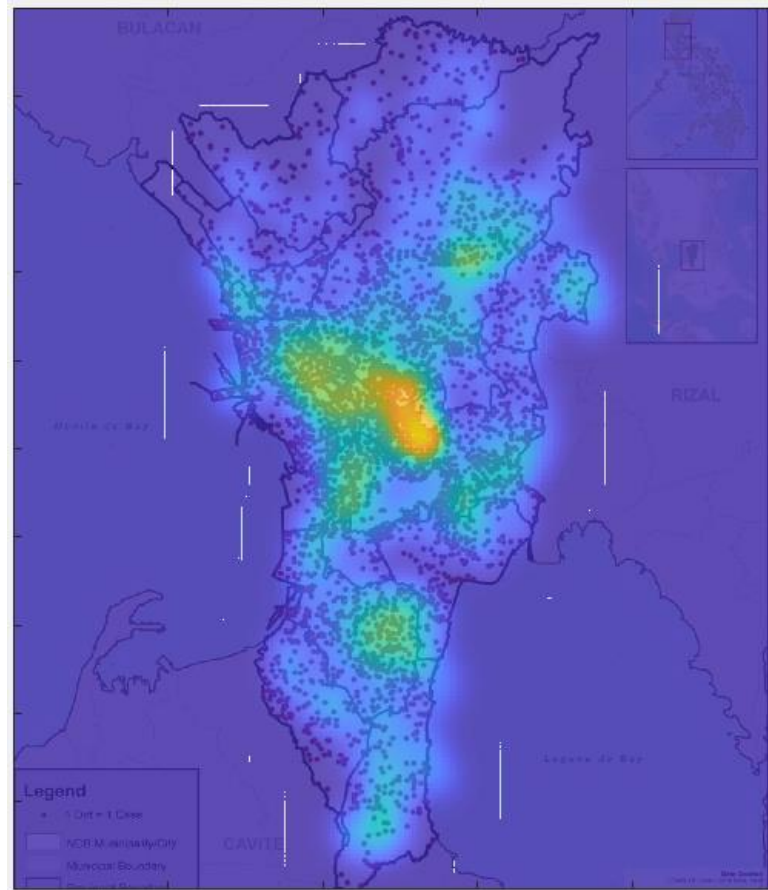
Density Estimation

Example:

Use the bivariate kernel density estimator to find the distribution of Covid cases using a bandwidth of $h = [0.02, 0.02]$ and the Gaussian kernel.



Answer:



The areas with high density of cases will have higher peaks in the estimated distribution.

Anomaly Detection

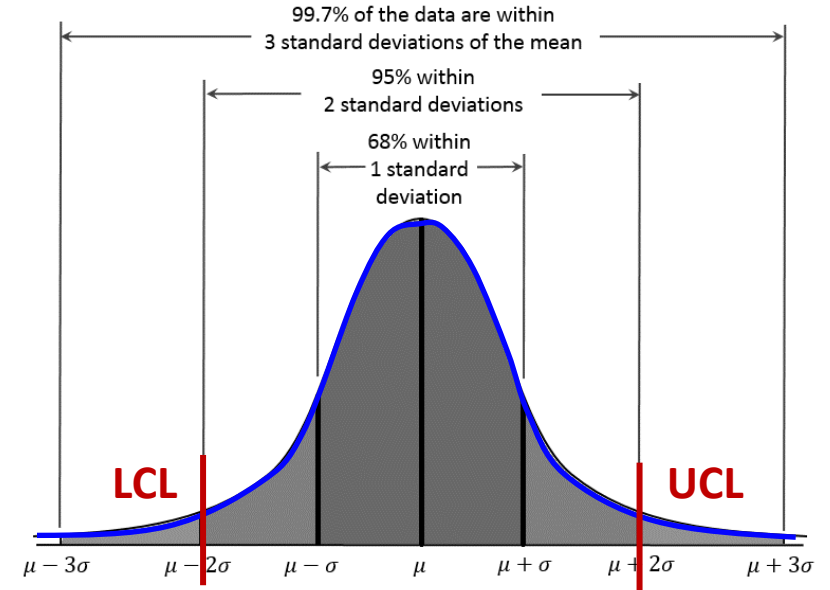
One important application of KDE is *anomaly detection*:
Finding data points that deviate from the expected distribution.

Practical uses of anomaly detection:

- Fault detection in any industrial system
- Defect detection in products
- Fraud detection in financial transactions
- Detecting cyber-attacks, network intrusions
- Detecting fake data (images, texts, etc.)
- Detecting anomalies in medical data
- Outliers and change points detection in time series
- Detecting unusual human behavior*

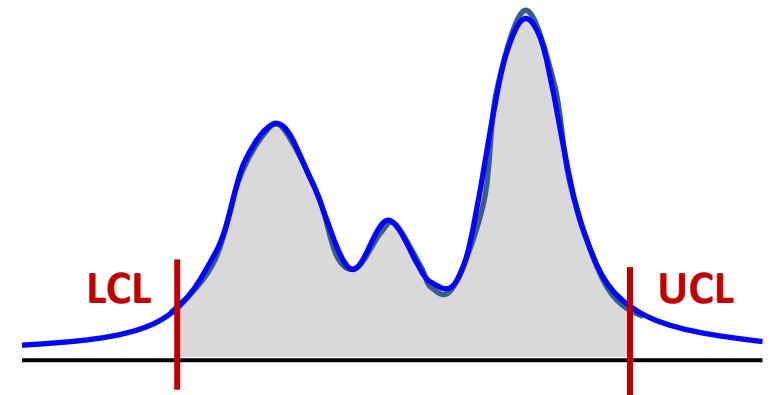
If the data is Gaussian-distributed: (Use Elliptic Envelope or just Quantiles)

- Fit a Gaussian distribution to the data.
Find μ and Σ .
- Establish a confidence level, α , e.g. 95%.
- Calculate the thresholds at which the area under the distribution is a fraction of α .
- Use the thresholds as **upper control limit** and **lower control limit**.



If the data is not Gaussian-distributed: (Use KDE)

- Fit a distribution to the data using **Kernel Density Estimation** (KDE).
- Do the same steps as above.



Other Anomaly Detection Methods

One-Class SVM

Support Vector Data Description (SVDD)

Tax and Duin (2004)

- Minimize the volume of a *hyper-sphere* around the data points.
- All points outside the *hyper-sphere* are outliers.

$$\min_{R, a} R^2 + C \sum_{i=1}^n \epsilon_i$$

$$\|x_i - a\|^2 \leq R^2 + \epsilon_i \quad \forall i = 1, 2, \dots, n$$

$$\epsilon_i \geq 0 \quad \forall i = 1, 2, \dots, n$$

`sklearn.svm.OneClassSVM`

Support Vector Method for Novelty Detection

Scholkopf *et al.* (1999)

- Separate the data points from the origin by maximizing the distance of a *hyper-plane* to the origin.
- All points outside the *hyper-plane* are outliers.

$$\min_{w, \epsilon, \rho} \frac{1}{2} \|w\|^2 + \frac{1}{n} \sum_{i=1}^n \epsilon_i - \rho$$

$$\langle w, \phi(x_i) \rangle \geq \rho - \epsilon_i \quad \forall i = 1, 2, \dots, n$$

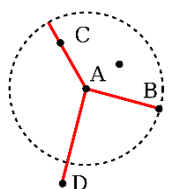
$$\epsilon_i \geq 0 \quad \forall i = 1, 2, \dots, n$$

Local Outlier Factor

Breunig (2000)

- Similar to DBSCAN in that it is density-based.
- Relevant Formula:

- Reachability Distance (RD)**



RD(A,B) and **RD(A,C)** are the same since B and C belong to the k-nearest neighbors of A. Meanwhile, **RD(A,D)** = $\text{dist}(A, D)$ since D is not a neighbor of A.

- Local Reachability Density (LRD)**

$$LRD(A) = \frac{1}{\frac{1}{|N_k(A)|} \sum_{B \in N_k(A)} RD(A, B)}$$

$N_k(A)$ is the set of k-nearest neighbors of A.

- Local Outlier Factor (LOF)**

$$LOF(A) = \frac{\sum_{B \in N_k(A)} LRD(B)}{|N_k(A)| \cdot LRD(A)}$$

Higher LOF, more likely an outlier

“Average LRD of the neighbors of point A divided by A's own LRD”

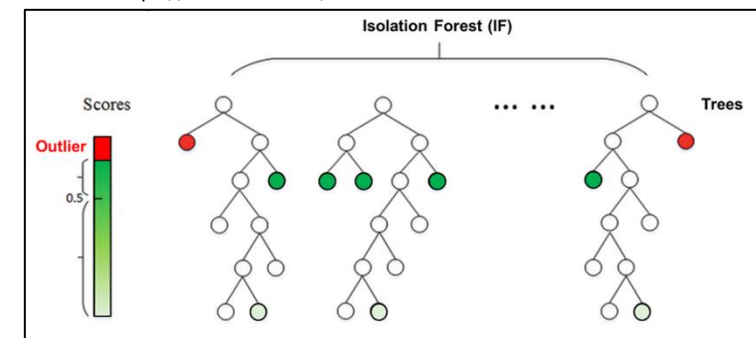
`sklearn.neighbors.LocalOutlierFactor`

Isolation Forest

Liu *et al.* (2008)

- Build a typical forest of Decision Trees.
- The shorter the average paths from the root to a leaf, the more likely the leaf is an outlier.

Source: <https://wiki.datarics.ai/isolation-forest-model>



`sklearn.ensemble.IsolationForest`

Anomaly Score

$$Score(x, n) = 2^{\frac{-E(h(x))}{c(n)}}$$

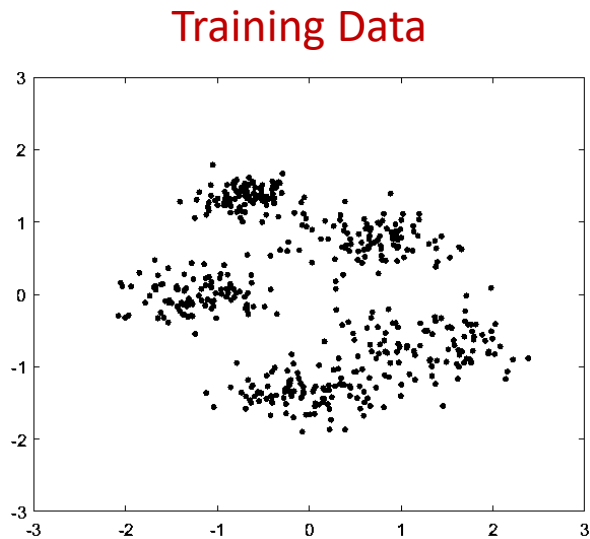
Higher score, more likely an outlier

Path length $h(x)$ is the number of edges traversed from the root to a leaf node x .

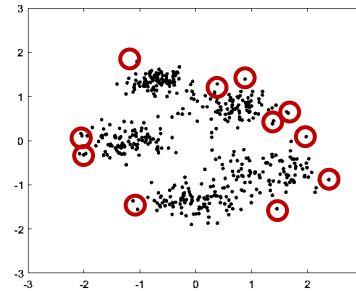
$c(n)$ = average path length of all leaves as if searched via a Binary Search Tree.

Side Note: Outlier Detection vs. Novelty Detection

- Anomalies can be: **outliers** that contaminate the *existing* training data.
- Anomalies can also be: **novel test data** that do not conform to a given training data.

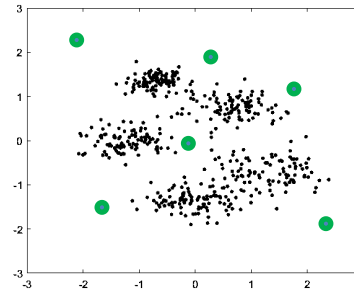


Outlier
Detection
only



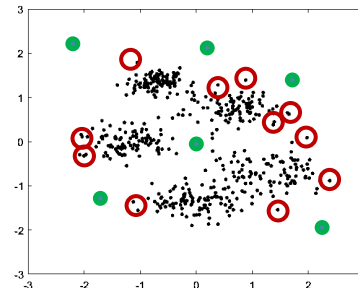
- Only training data are analyzed.
- There are no new test data.
- Which training data points are outliers?

Novelty
Detection
only



- Only test data are analyzed.
- All training data are treated as *normal* or *uncontaminated by anomalies*.
- Which test data points are novel with respect to the training set?

Detecting both
outliers and novel
data points



- Both training data and new test data are analyzed.
- Which training data points are outliers?
Which test data points are novel?

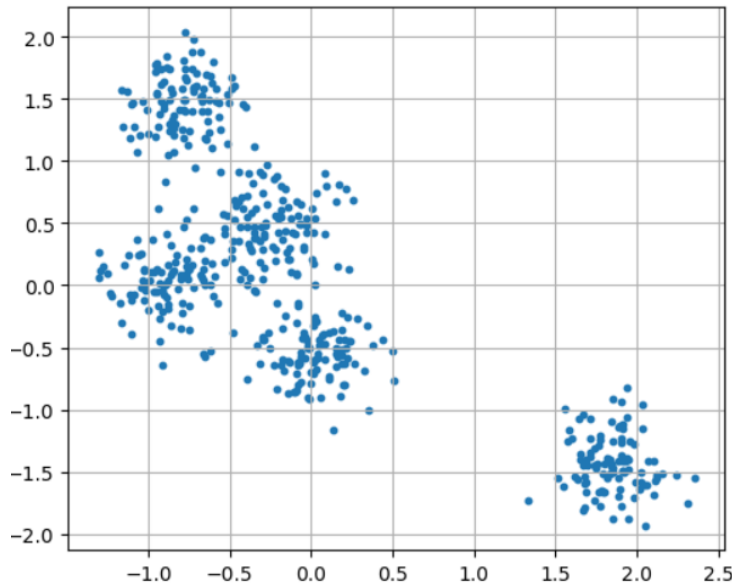
Outlier Detection = unsupervised anomaly detection
Novelty Detection = semi-supervised anomaly detection

Anomaly Detection

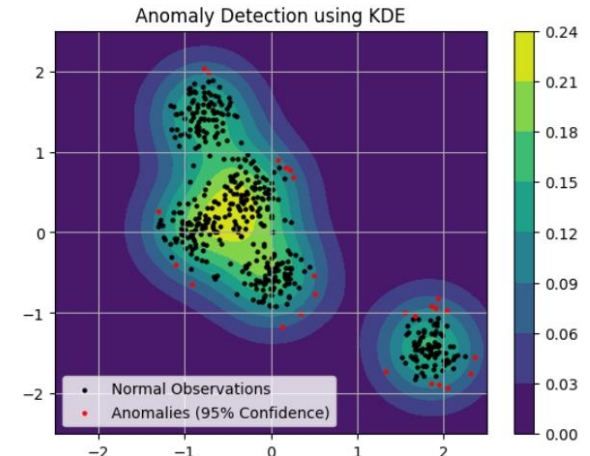
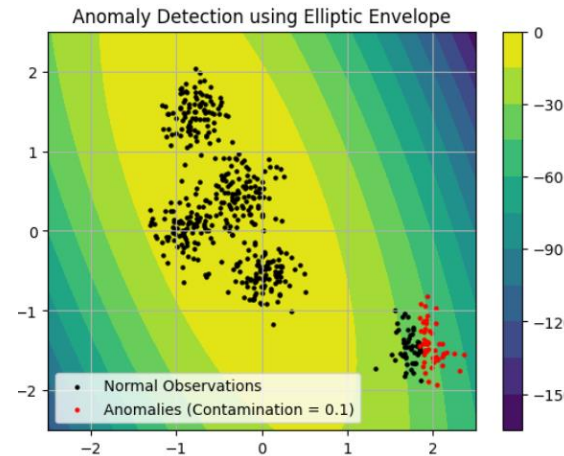
Example:

Identify outliers from this 2D data set using the following methods:

- Elliptic Envelope (contamination = 0.1)
- KDE at 95% confidence
- One-class SVM ($\nu = 0.05, \gamma = 1$)
- Local Outlier Factor (no. of neighbors = 5)
- Isolation Forest (contamination = 0.1)

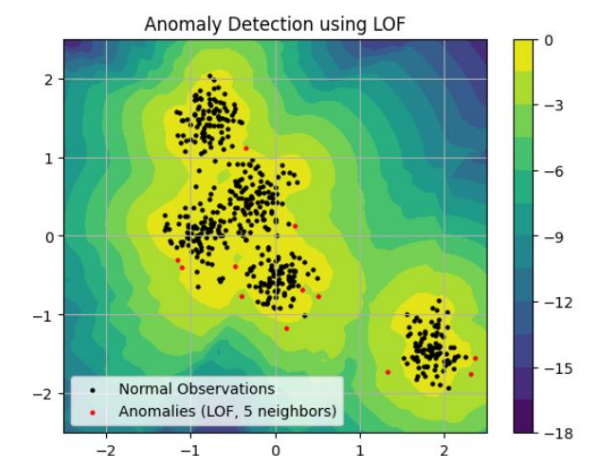
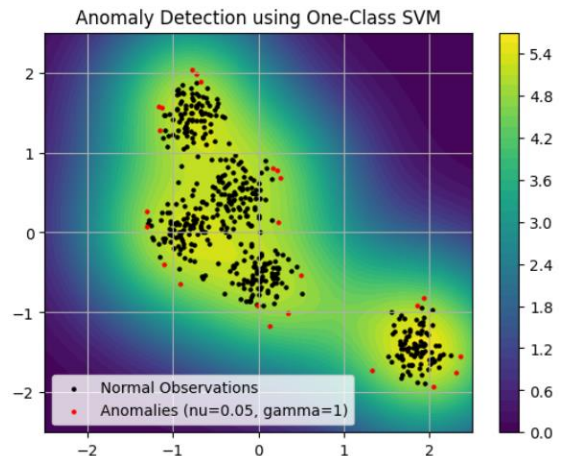


Elliptic Envelope



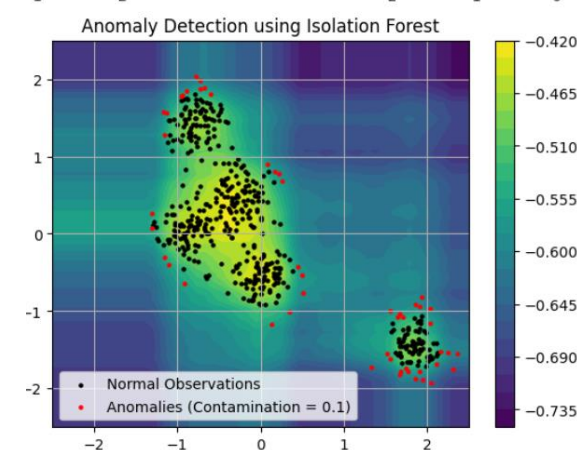
Kernel Density Estimation

One-Class SVM



Local Outlier Factor

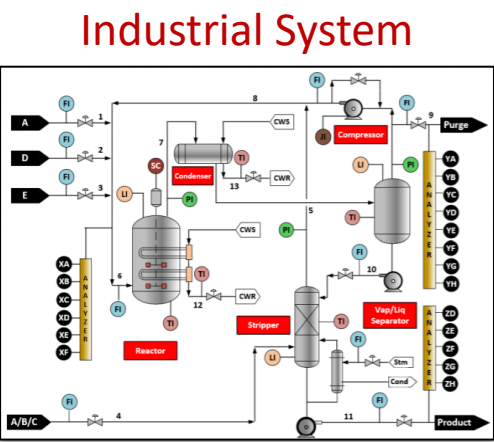
Isolation Forest



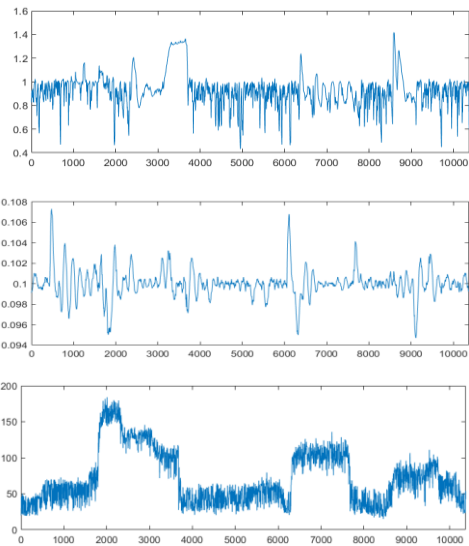
Note: Generating the contour map of anomaly scores is a *novelty* detection task.

Side Note: Anomaly Detection in Multivariate Data

In industry, **statistical control charts** are used to detect outliers / abnormal events.

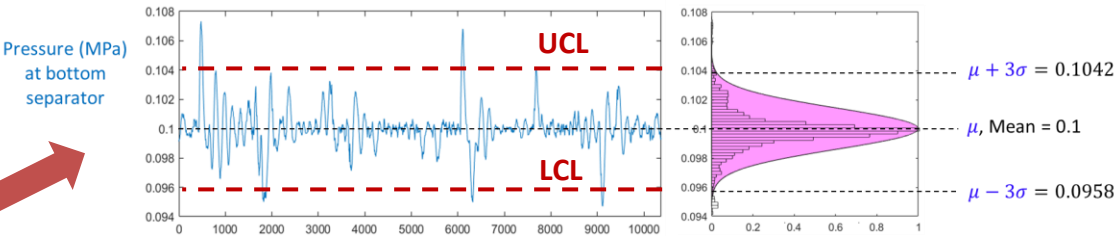


Multivariate Data
(Time Series)

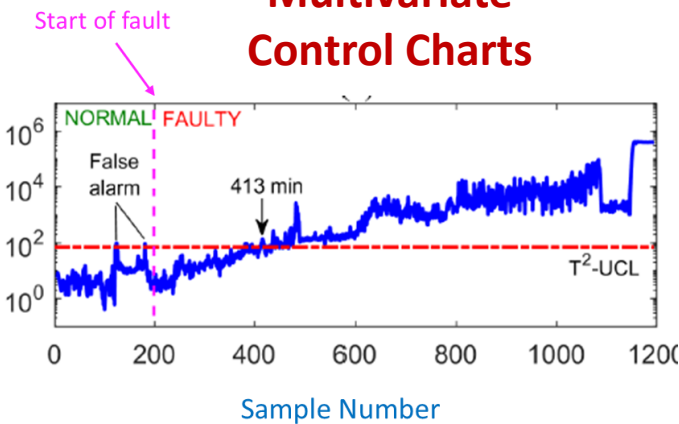


Time
(1 sample per second)

Univariate
Control Charts



Multivariate
Control Charts



Feature Extraction
(e.g. PCA, ICA, Deep Learning)

Health Indicator
(e.g. Hotelling's T^2 statistic)

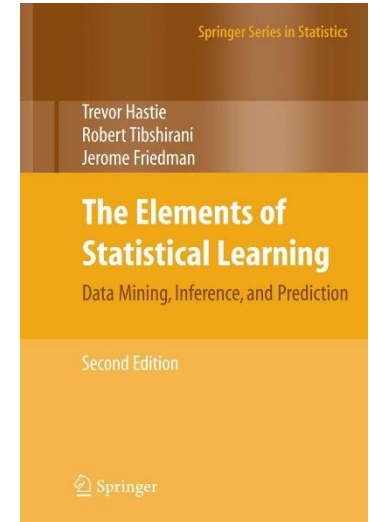
Threshold Setting
(e.g. KDE, LOF, OC-SVM, etc.)

All steps involve only unsupervised learning because abnormal data is scarce in industry.

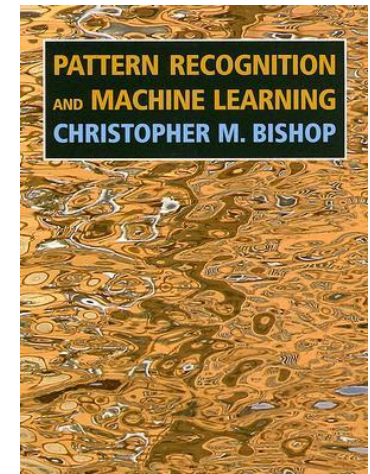
Outline

- Clustering
 - K-means Clustering
 - Hierarchical Clustering
 - Density-based Clustering
 - Gaussian Mixture Model
- Density Estimation
 - Kernel Density Estimation (KDE)
- Anomaly Detection
 - Application of KDE
 - One-class SVM
 - Local Outlier Factor
 - Isolation Forest

Hastie *et al.* (2008)
The Elements of Statistical Learning.
2nd Ed. Springer.



Bishop (2006)
*Pattern Recognition and
Machine Learning.* Springer.



Further Reading

- https://scikit-learn.org/stable/auto_examples/cluster/plot_kmeans_digits.html
- <https://scikit-learn.org/stable/modules/clustering.html>
- Daniel Mullner, “Modern hierarchical, agglomerative clustering algorithms”, [arXiv:1109.2378v1](https://arxiv.org/abs/1109.2378v1).
- Hamerly and Elkan (2002). Alternatives to the K-means algorithm that find better clusterings. <https://dl.acm.org/doi/10.1145/584792.584890>
- Arthur and Vassilvitskii (2006). K-means++: The Advantages of Careful Seeding. <http://ilpubs.stanford.edu:8090/778/1/2006-13.pdf>
- Ward, J. H., Jr. (1963), "Hierarchical Grouping to Optimize an Objective Function", Journal of the American Statistical Association, 58, 236–244.
- Ng, Andrew Y and Jordan, Michael I and Weiss, Yair (2002). "On spectral clustering: analysis and an algorithm" (PDF). Advances in Neural Information Processing Systems. <https://ai.stanford.edu/~ang/papers/nips01-spectral.pdf>
- DBSCAN paper: Ester, M., H. P. Kriegel, J. Sander, and X. Xu, “A Density-Based Algorithm for Discovering Clusters in Large Spatial Databases with Noise”. In: Proceedings of the 2nd International Conference on Knowledge Discovery and Data Mining, Portland, OR, AAAI Press, pp. 226-231. 1996
- Schubert, E., Sander, J., Ester, M., Kriegel, H. P., & Xu, X. (2017). “DBSCAN revisited, revisited: why and how you should (still) use DBSCAN.” ACM Transactions on Database Systems (TODS), 42(3), 19.
- <https://towardsdatascience.com/clustering-algorithm-for-customer-segmentation-e2d79e28cbc3>
- Arbelaitz, O., Gurrutxaga, I., Muguerza, J., Perez, J. M., & Perona, I. (2013). An extensive comparative study of cluster validity indices. *Pattern Recognition*, 46(1), 243–256.
- Ruff, Lukas & Kauffmann, Jacob & Vandermeulen, Robert & Montavon, Gregoire & Samek, Wojciech & Kloft, Marius & Dietterich, Thomas & Müller, Klaus-Robert. (2021). [A Unifying Review of Deep and Shallow Anomaly Detection](#). Proceedings of the IEEE. PP. 1–40. 10.1109/JPROC.2021.3052449.
- Breunig, M. M., Kriegel, H. P., Ng, R. T., & Sander, J. (2000, May). LOF: identifying density-based local outliers. In ACM sigmod record.
- Fei Tony Liu, Kai Ming Ting, and Zhi-Hua Zhou, “Isolation Forest”, IEEE International Conference on Data Mining 2008 (ICDM 08).
- https://scikit-learn.org/stable/auto_examples/miscellaneous/plot_anomaly_comparison.html