LECTURER: Nghia Duong-Trung

MACHINE LEARNING

Introduction to Machine Learning	1
Clustering	2
Regression	3
Support Vector Machines	4
Decision Trees	5
Genetic Algorithm	6

UNIT 2

CLUSTERING

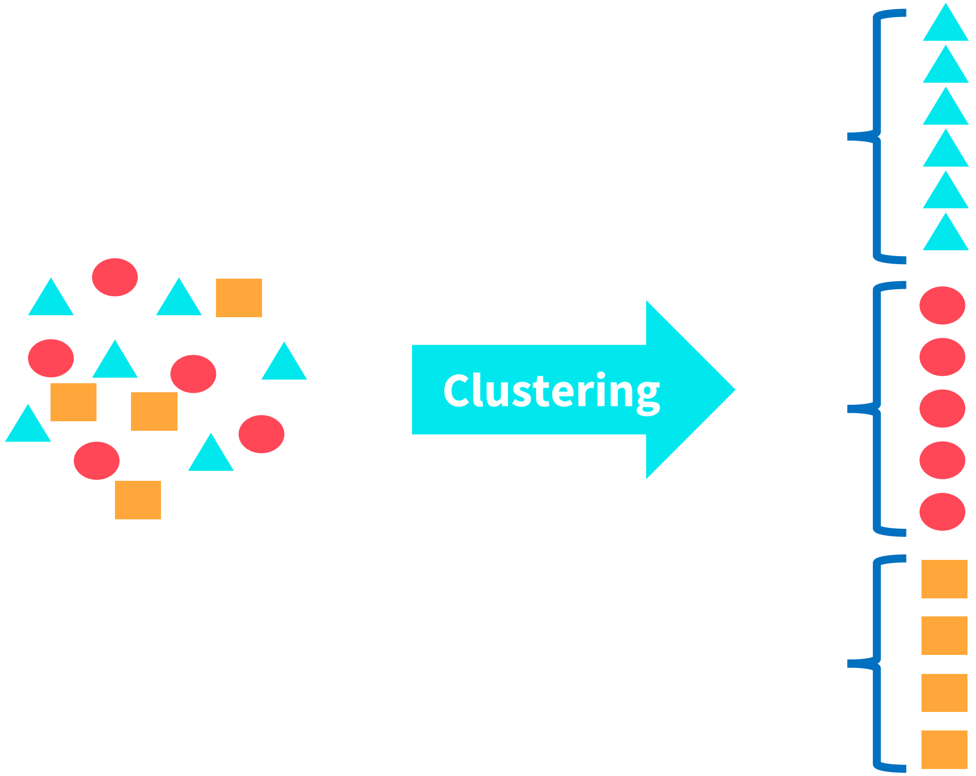
STUDY GOALS

- Know the definitions and terms used for clustering
- Comprehend common applications of clustering analysis
- Understand different methods for clustering analysis
- Analise the advantages and limitations of the clustering methods
- Implement clustering methods in Python

EXPLAIN SIMPLY

1. What is clustering and its application?
2. What are the popular methods of clustering, their advantages and limitations?
3. How to implement the clustering methods in python?

- Clustering is an **unsupervised** learning technique to:
  - reveal meaningful **partitions, hierarchies**
  - find association **rules** of data
  - gather data points into **groups** and extract **useful information**



An example of clustering

Source of the text: Das, Mukherjee, Majumder, & Gupta, 2014.

Clustering applications

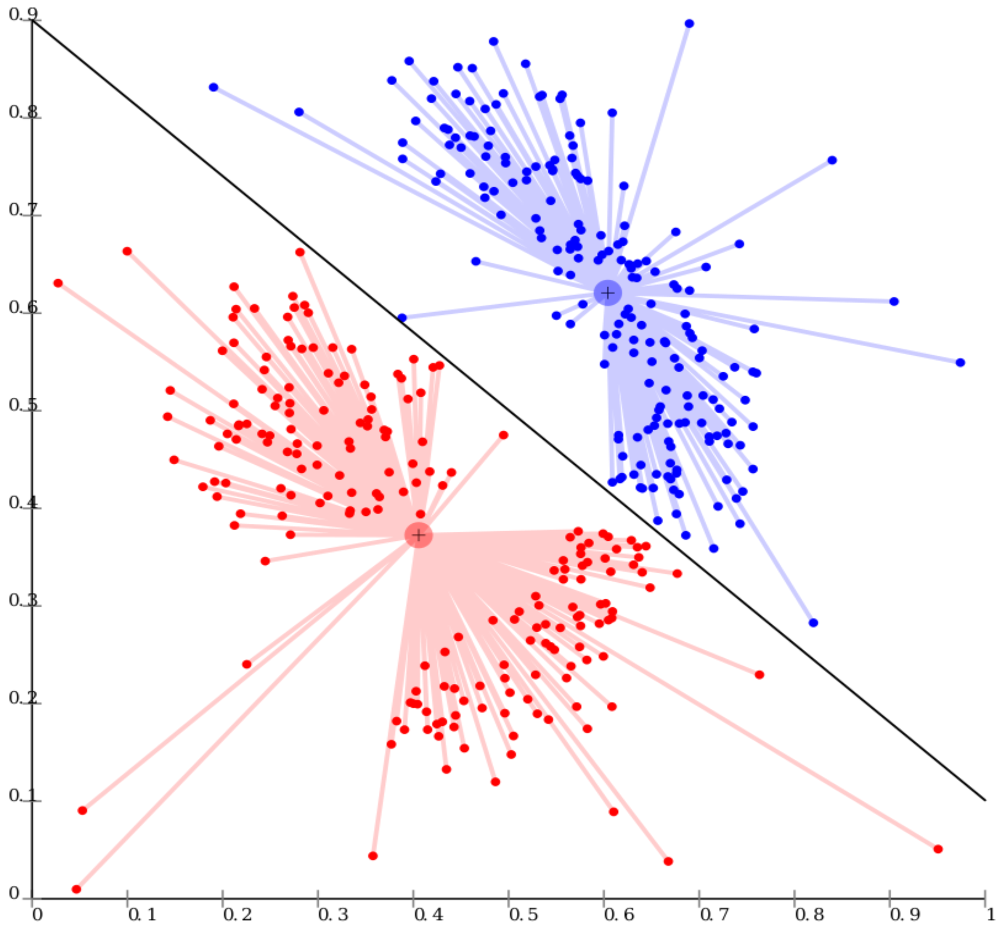
- Pattern recognition
- Robotics
- Spatial data analysis
- Image processing
- Crime analysis
- Bio-informatics
- Medical imaging

Clustering methods

- CENTROID-BASED
- GAUSSIAN MIXTURES MODELS
- HIERARCHICAL
- DENSITY-BASED

Source of the text: Das, Mukherjee, Majumder, & Gupta, 2014.

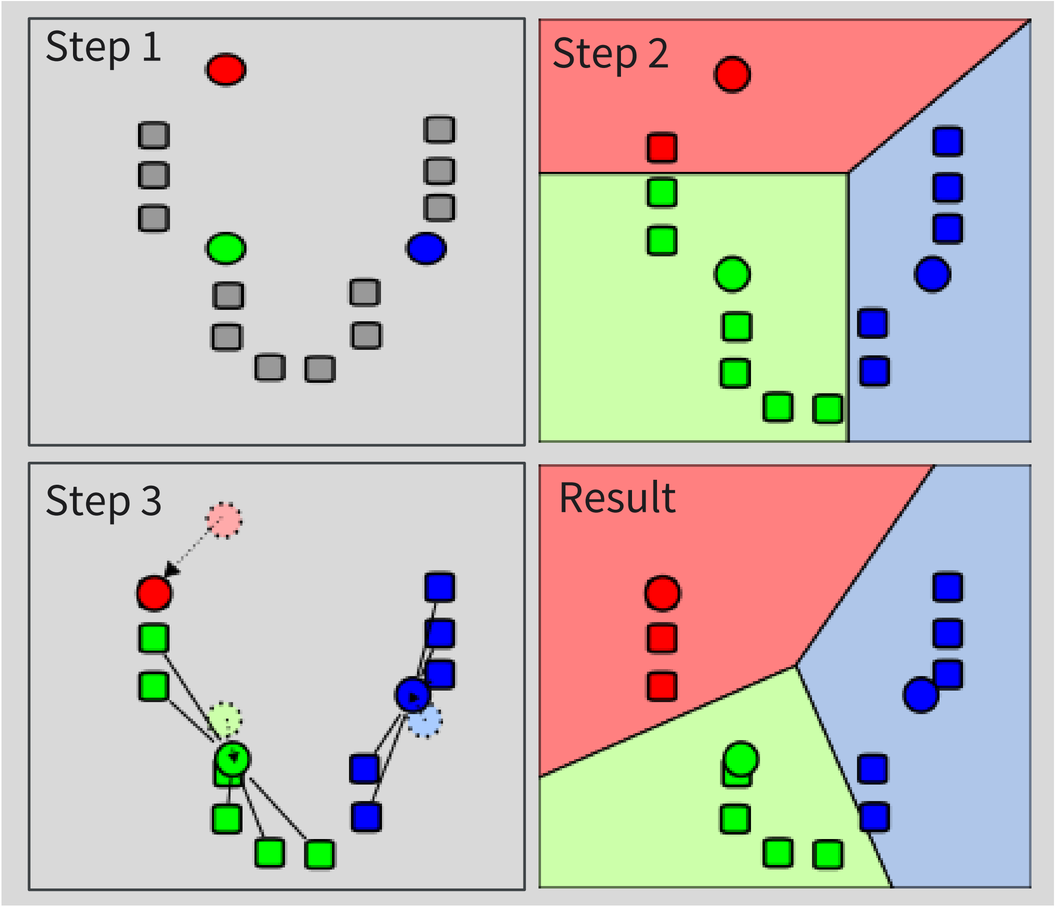
- **Centroid**: the arithmetic average position of all the points
- **Centroid-based clustering** searches for a pre-determined number of clusters
- Each data point is assigned to the cluster achieving the **minimum distance** from the centroid



An example of Centroid-based clustering

Source of the image: File:KMeans-density-data.svg (2020).

- **K-Means Clustering** partitions data points into K clusters
- Three main steps: initialization, assignment, update
- Advantages: **simple** to employ, efficient for large datasets
- Limitation: K must be determined, **initialization-dependent,** problems with high-dimensional data and outliers



Algorithm of K-Means clustering

Source of the text: Data Science Team, 2020; Pragathi, Jayanthi, & Malathi, 2018; Zhu, 2022.
Source of the image: File:K Means Example Step 1.svg (2020).

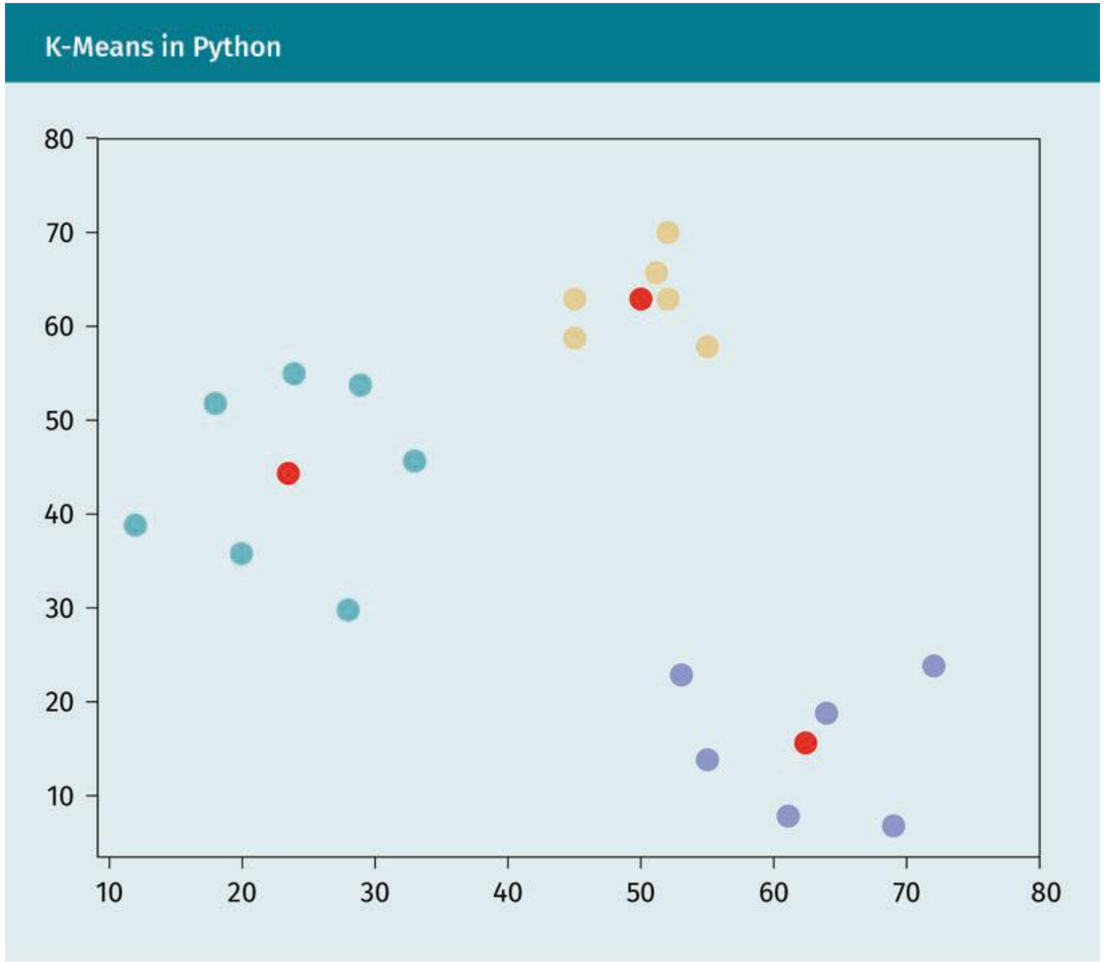
File:K Means Example Step 2.svg (2020).

File:K Means Example Step 3.svg (2020).

File:K Means Example Step 4.svg (2020).

2.1 CENTROID-BASED CLUSTERING

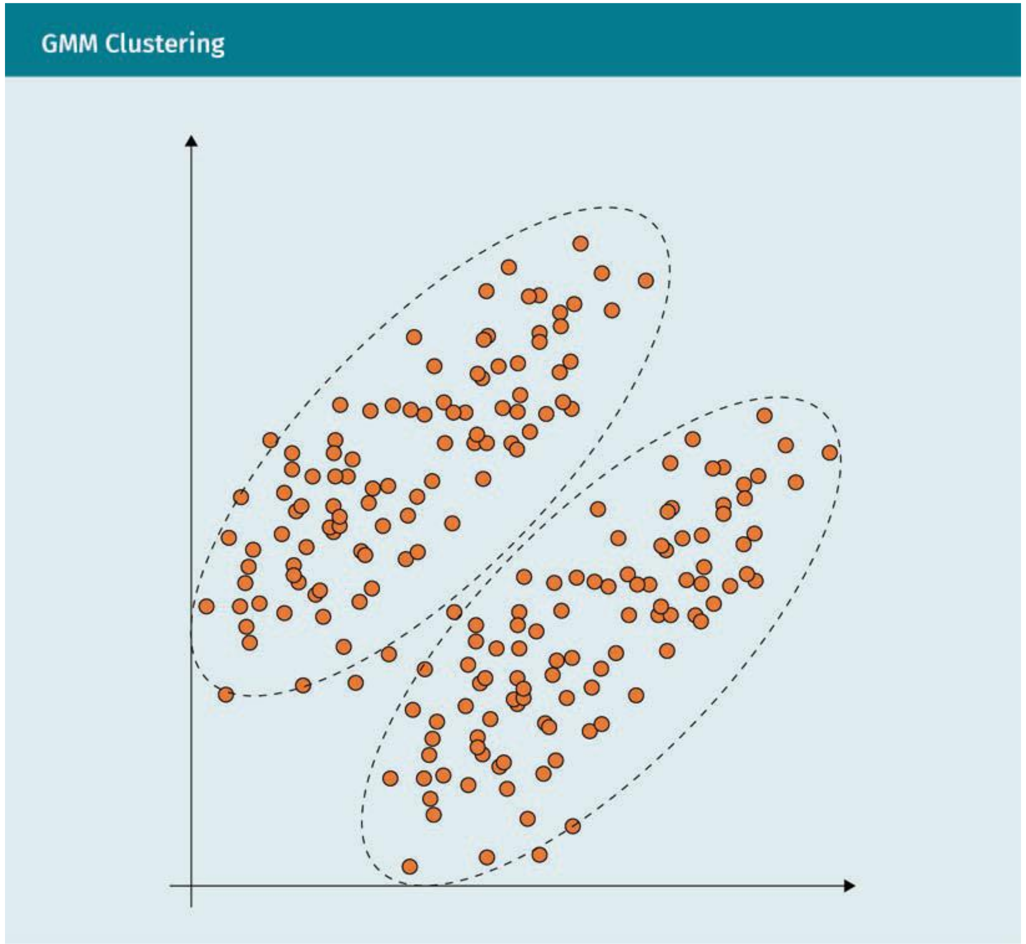
```
#K-Means clustering with Python
>>> import pandas as pd # For reading datasets
>>> import numpy as np # For computations
>>> import matplotlib.pyplot as plt # For visualization
>>> from pandas import DataFrame # For creating data frame
>>> from sklearn.cluster import KMeans
>>> Data={
'x': [12, 20, 28, 18, 29, 33, 24, 45, 45, 52, 51, 52, 55, 53, 55, 61, 64, 69, 72],
'y': [39, 36, 30, 52, 54, 46, 55, 59, 63, 70, 66, 63, 58, 23, 14, 8, 19, 7, 24]
}
>>> df = DataFrame(Data,columns=['x','y'])
# Create and fit the KMeans model
>>> kmeans = KMeans(n_clusters=3).fit(df)
# Find the centroids of the clusters
>>> centroids = kmeans.cluster_centers_
# Get the associated cluster for each data record
>>> kmeans.labels_
# Display the clusters contents and their centroids
>>> plt.scatter(df['x'], df['y'], c= kmeans.labels_.astype(float), s=50,
alpha=0.5)
>>> plt.scatter(centroids[:, 0], centroids[:, 1], c='red', s=50)
>>> plt.show()
```



Scatter Plot of Dataset With Clusters Identified Using K-means Clustering

Source of the text: Data Science Team, 2020; Pragathi, Jayanthi, & Malathi, 2018; Zhu, 2022.
Source of the image: Zöller (2022).

- **GMM**: probabilistic generative models
- Each data point is assigned to the cluster returning the **highest probability**



An example of GMM clustering

Source of the text: Carrasco, 2019.
Source of the image: Zöller (2022).

- The optimum means (μ) and standard deviations (σ) are found by applying the **Expectation maximization** algorithm
- Advantages: **efficient** for datasets with complex data distribution
- Limitation: **slow**, heavy computation, and can get stuck in local-maximum

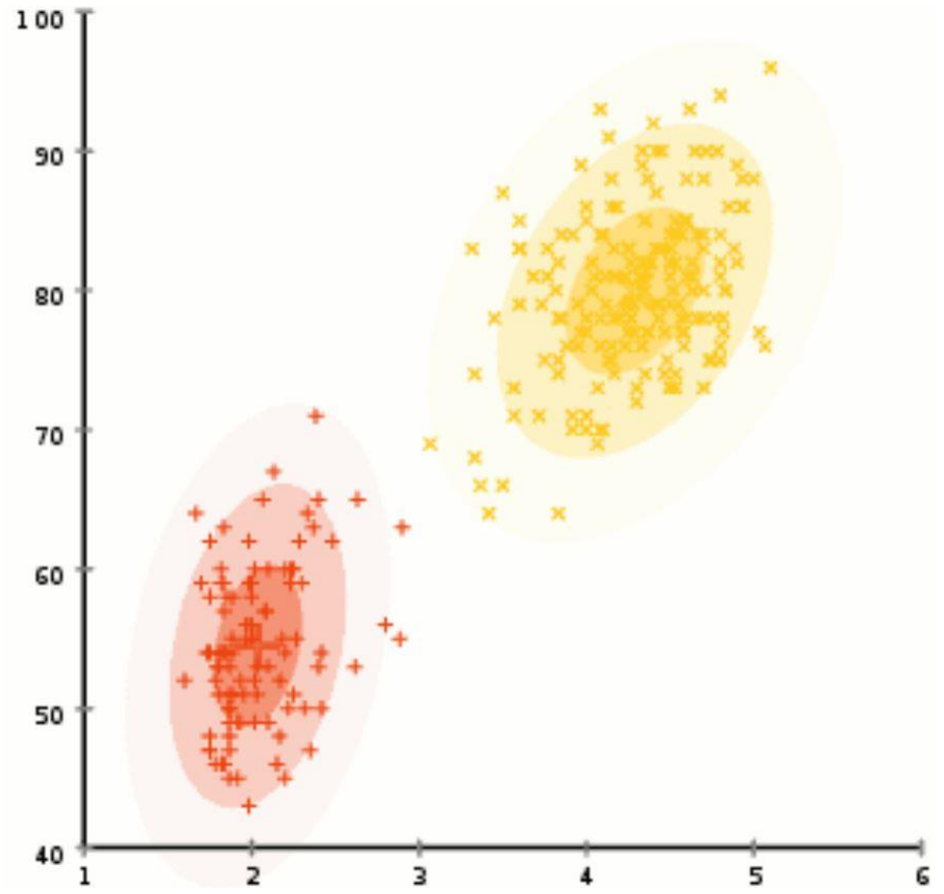
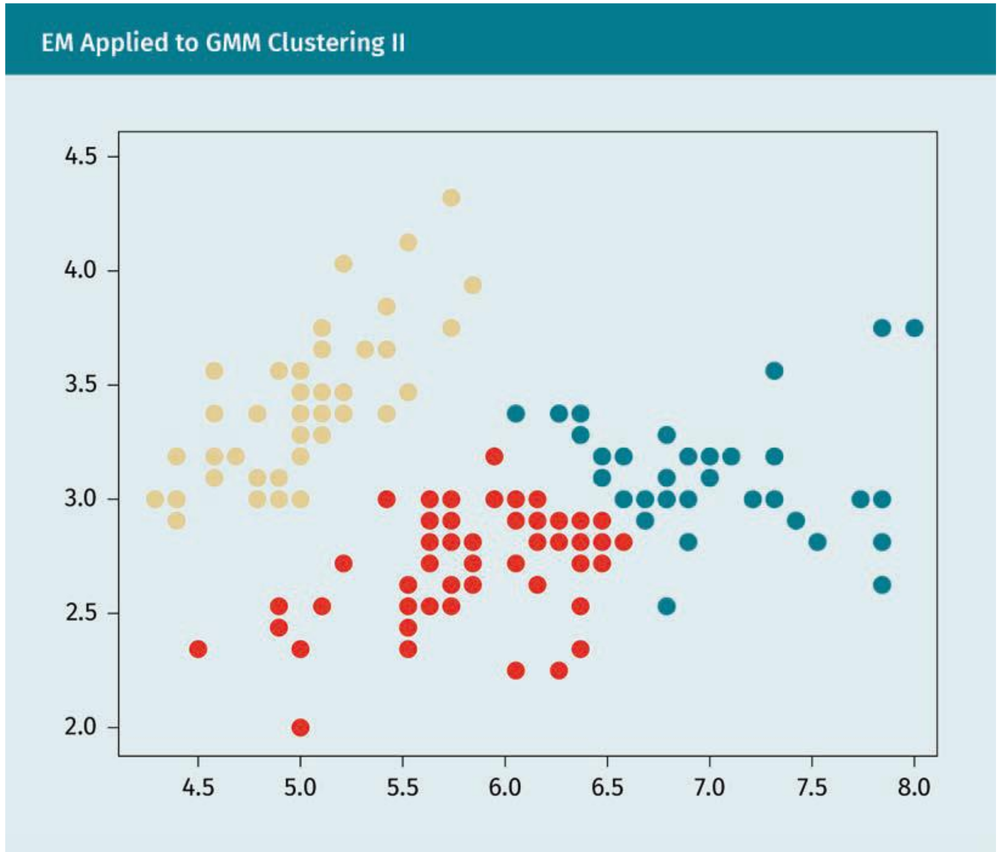


Illustration of expectation-maximization algorithm

Source of the text: Carrasco, 2019.
Source of the image: File:EM Clustering of Old Faithful data.gif (2020).

2.2 GAUSSIAN MIXTURE MODELS CLUSTERING

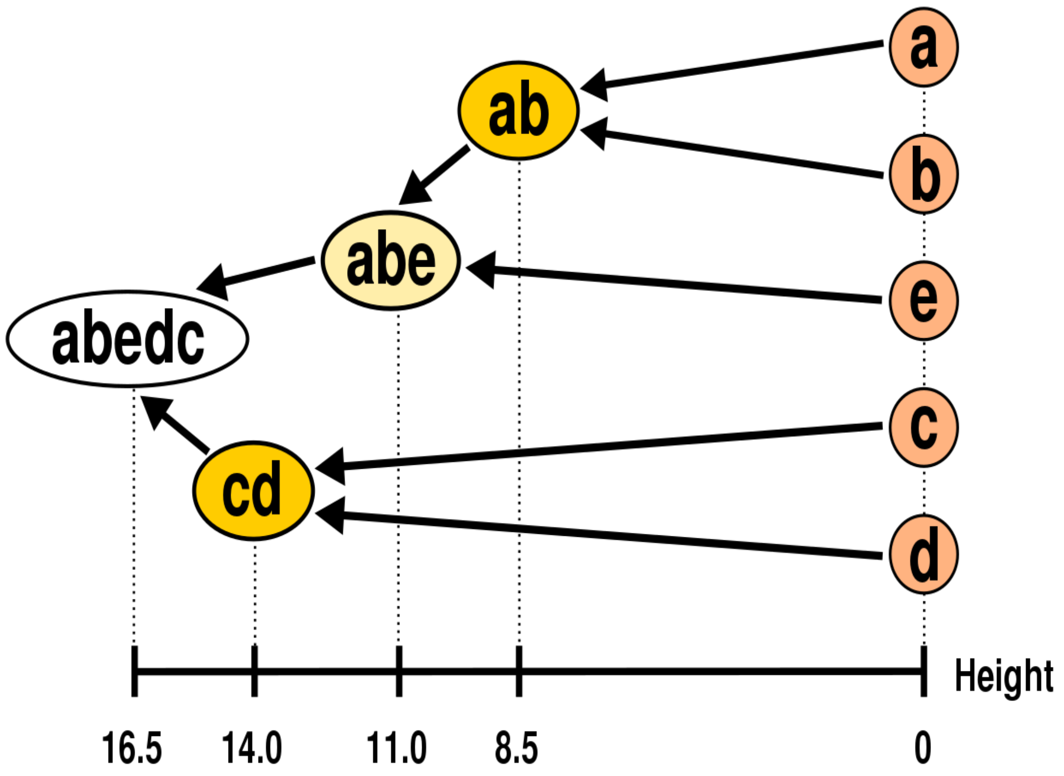
```
# GMM clustering with Python
>>> import numpy as np
>>> import pandas as pd
>>> import matplotlib.pyplot as plt
>>> from pandas import DataFrame
>>> from sklearn import datasets
>>> from sklearn.mixture import GaussianMixture
>>> iris = datasets.load_iris() # load the iris dataset
>>> X = iris.data[:, :2] # select first two columns
>>> d = pd.DataFrame(X) # turn it into a dataframe
>>> plt.scatter(d[0], d[1])
>>> plt.show() # plot the data
>>> gmm = GaussianMixture(n_components = 3)
>>> gmm.fit(d) # fit the data as a mixture of 3 Gaussians
>>> labels = gmm.predict(d) # predict the cluster of each data record
>>> print('Converged:',gmm.converged_) # Check if the model has converged
>>> means = gmm.means_ # get the final “means” for each cluster
>>> covariances = gmm.covariances_ # get the final standard deviations
>>> d['labels']= labels
>>> d0 = d[d['labels']== 0]
>>> d1 = d[d['labels']== 1]
>>> d2 = d[d['labels']== 2]
>>> plt.scatter(d0[0], d0[1], c ='r')
>>> plt.scatter(d1[0], d1[1], c ='yellow')
>>> plt.scatter(d2[0], d2[1], c ='g')
>>> plt.show() # plot the data records in each clusters in different color
```



Scatter Plot of Dataset With Clusters Identified Using GMM Clustering

Source of the image: Zöller (2022).

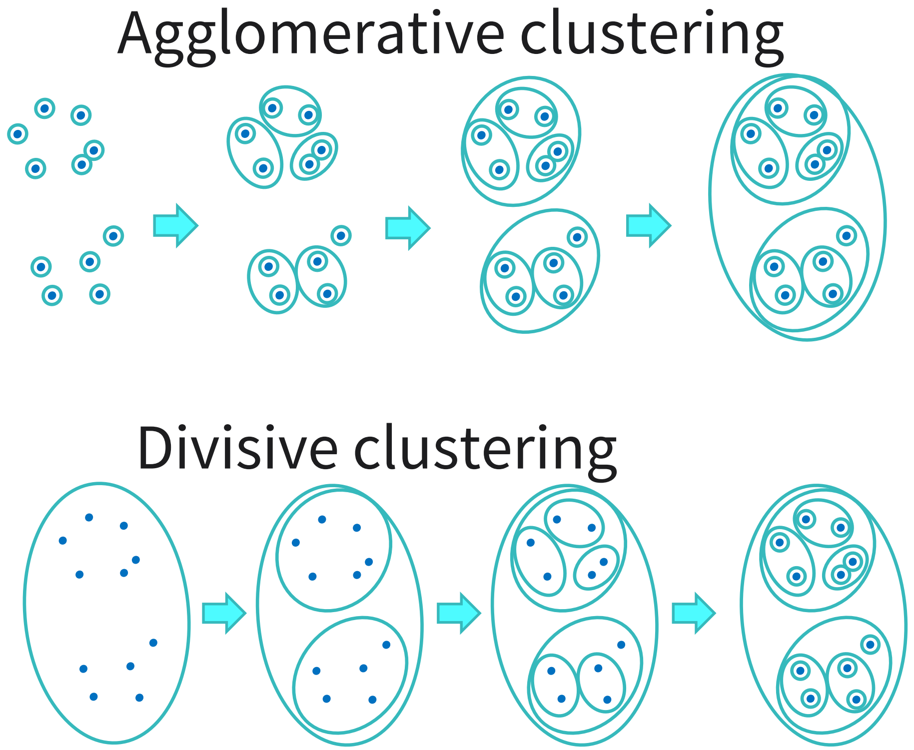
- The clusters are built in a hierarchy as a **tree**
- Three levels of clusters:
  - **Universe**: root
  - **Intermediate**
  - **Single-points**. leaves



Tree-view (dendrogram) of hierarchical clustering

Source of the text: Pai, 2021.
Source of the image: File:UPGMA Dendrogram Hierarchical.svg (2020).

- Two types of clustering:
  - **Agglomerative**: bottom-up
  - **Divisive**: top-down
- Data points are merged/split **based on their distance**
- Advantages: easy to implement, intuitive, no assumption
- Limitation: not suitable for large datasets, sensitive to outliers

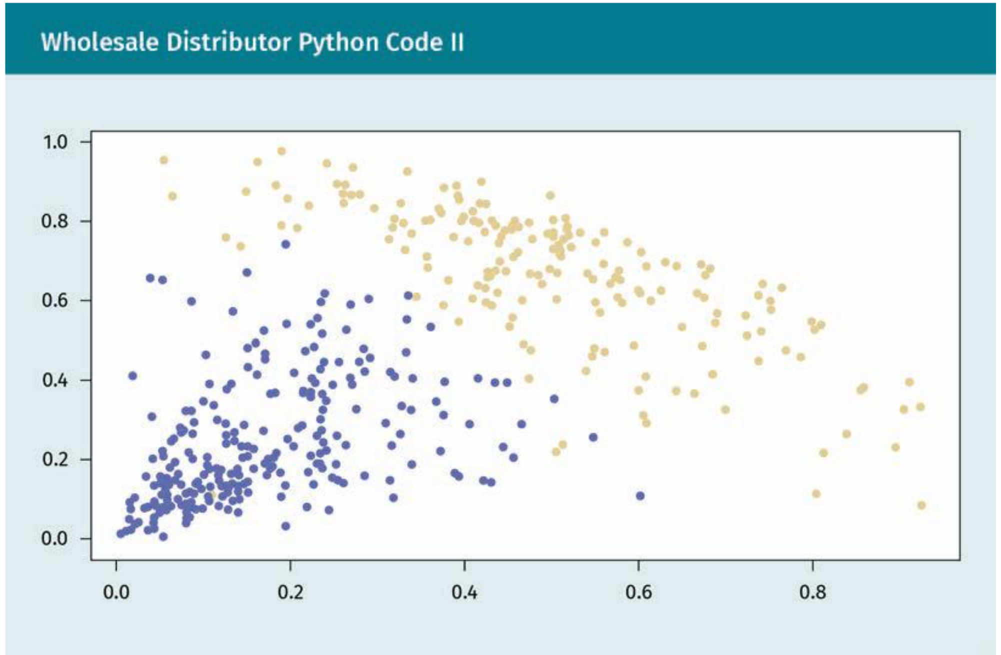


An example of hierarchical clustering

Source of the text: Pai, 2021.

2.3 HIERARCHICAL CLUSTERING

```
# Agglomerative Clustering with Python
>>> import pandas as pd
>>> import numpy as np
>>> import matplotlib.pyplot as plt
>>> data = pd.read_csv('Wholesale customers data.csv’)
# Normalize the dataset to get all the features at the same scale
>>> from sklearn.preprocessing import normalize
>>> data_scaled = normalize(data)
>>> data_scaled = pd.DataFrame(data_scaled, columns=data.columns)
# Draw the dendrogram to find the optimum number of clusters
>>> import scipy.cluster.hierarchy as shc
>>> plt.figure(figsize=(10, 7))
>>> plt.title("Dendrograms")
>>> dend = shc.dendrogram(shc.linkage(data_scaled, method='ward'))
>>> plt.show()
# From the dendrogram, we decide the optimum number of clusters is 2
# apply hierarchical clustering foe two clusters only
>>> from sklearn.cluster import AgglomerativeClustering
>>> cluster = AgglomerativeClustering(n_clusters=2, affinity='euclidean', link
age='ward')
>>> cluster.fit_predict(data_scaled)
# visualize the two clusters
>>> plt.figure(figsize=(10, 7))
>>> plt.scatter(data_scaled['Milk'], data_scaled['Grocery'], c=cluster.labels_)
>>> plt.show()
```

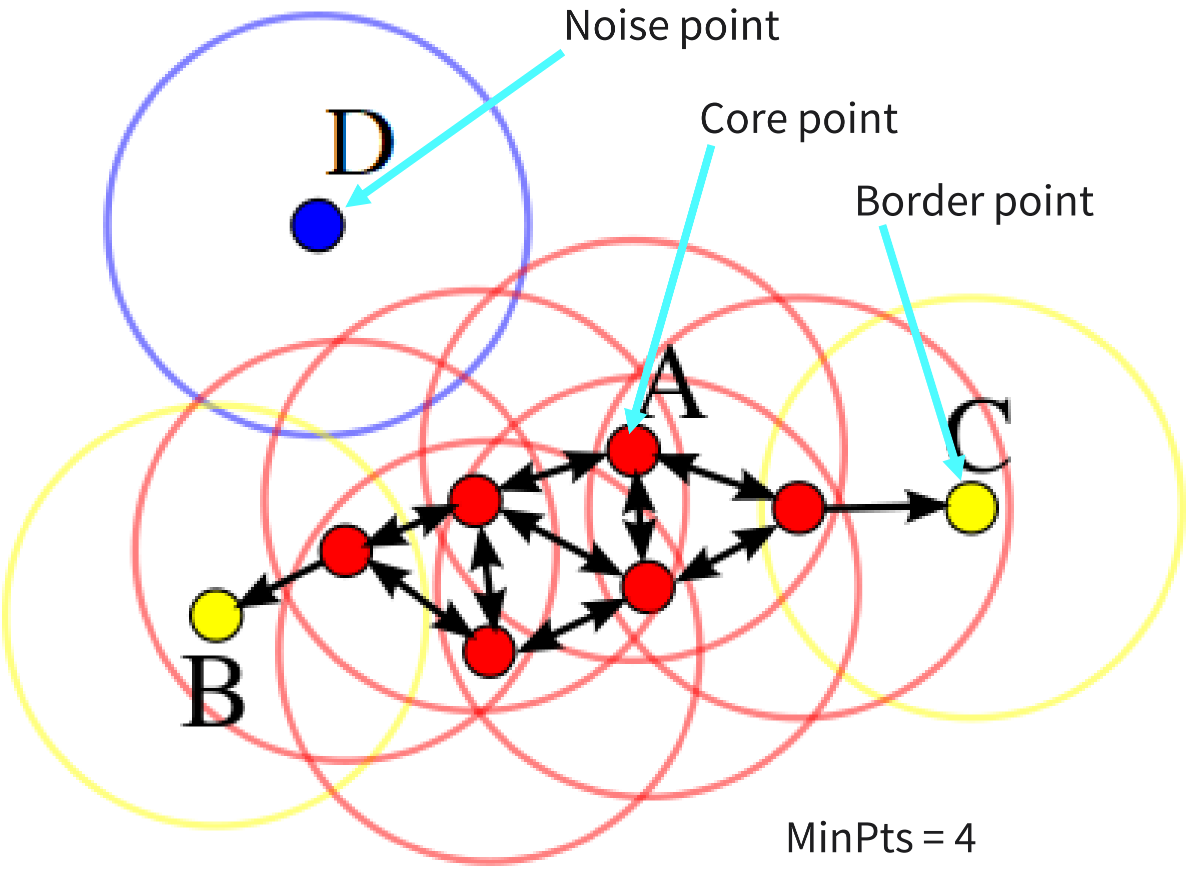


Scatter Plot of Dataset With Clusters Identified Using Agglomerative Clustering

Source of the image: Zöller (2022).

- Clusters are identified by grouping “dense” data points together, which permits:
  - the representation of **arbitrarily shaped clusters,**
  - the learning of **outliers**
- **ε**-Neighborhood of a point **p** in the dataset **d**:

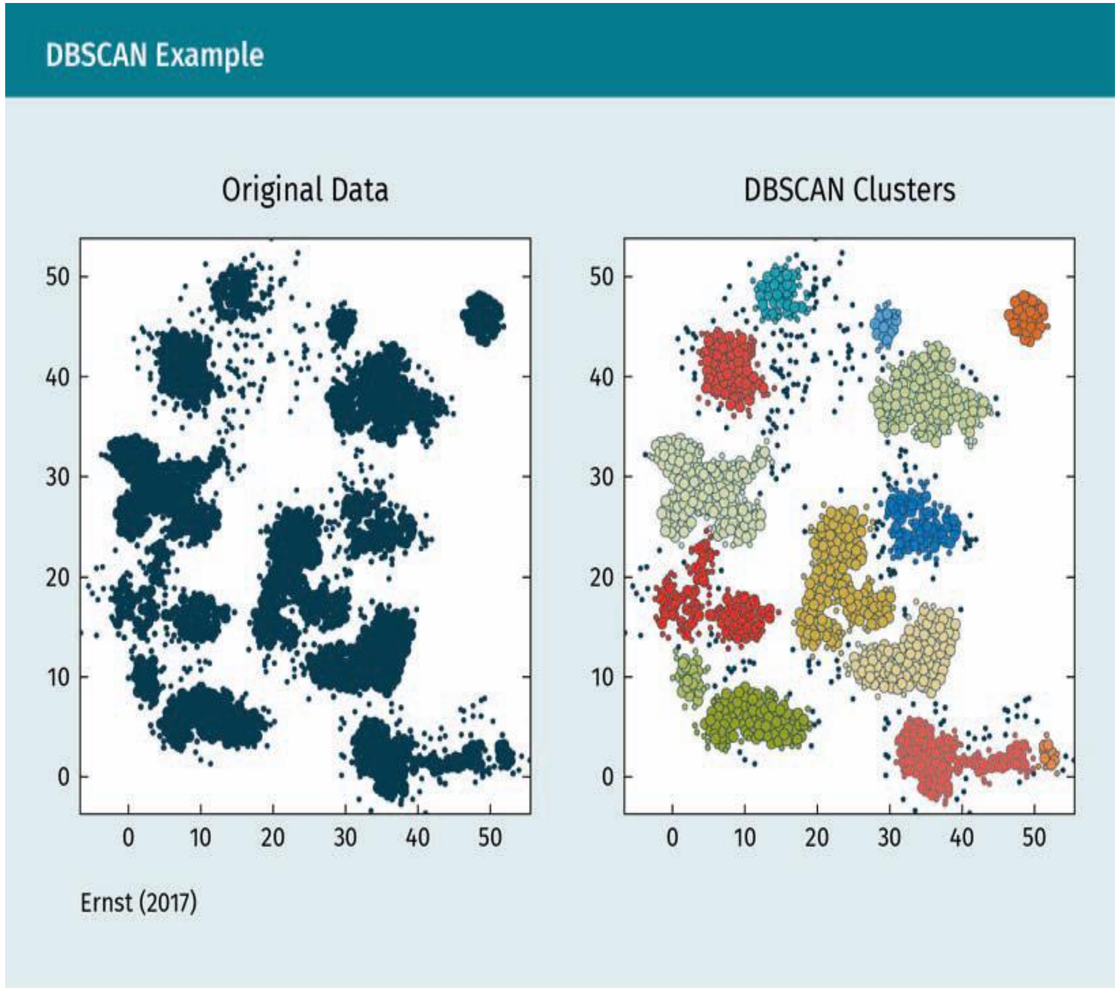
$$N(p) = \{q \in d | dist(p, q) \leq \varepsilon\}$$



Algorithm of Density-based Clustering

Source of the text: Ester, Kriegel, Sander, & Xiaowei, (1996).
Source of the image: File:DBSCAN-Illustration.svg (2020).

- Density-based spatial clustering for data with noise:
  1. Initialize MinPts and ε
  2. Start at a random data point p
  3. Check if p is a core point
  4. Density-connected clustering
  5. Start at new point, repeat 3 & 4
- Advantages: estimation of cluster number is not required
- Limitation: ε and MinPts setting up could be tricky

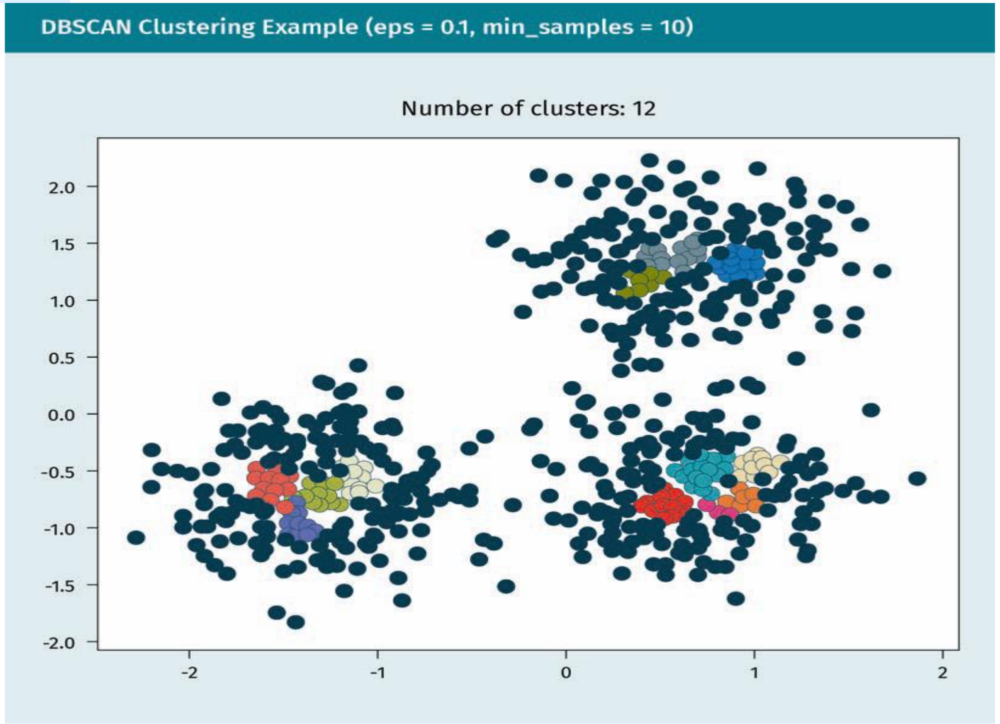


An example of DBSCAN Clustering

Source of the text: Ester, Kriegel, Sander, & Xiaowei, (1996).
Source of the image: Ernst (2017).

2.4 DENSITY-BASED CLUSTERING

```
# DBSCAN clustering with Python
>>> import numpy as np
>>> from sklearn.cluster import DBSCAN
>>> from sklearn import metrics
>>> from sklearn.datasets.samples_generator import make_blobs
>>> from sklearn.preprocessing import StandardScaler
>>> from pylab import *
>>> import matplotlib.pyplot as plt
>>> # Generate sample data
>>> centers = [[1, 1], [-1, -1], [1, -1]]
>>> X, labels_true = make_blobs(n_samples=750, centers=centers,
cluster_std=0.4, random_state=0)
>>> X = StandardScaler().fit_transform(X) # standardize the dataset
>>> xx, yy = zip(*X)
>>> db = DBSCAN(eps=0.3, min_samples=10).fit(X) # Set up parameters
>>> core_samples = db.core_sample_indices_
>>> core_samples_mask = np.zeros_like(db.labels_, dtype=bool)
>>> core_samples_mask[db.core_sample_indices_] = True
>>> n_clusters_ = len(set(labels))-(1 if -1 in labels else 0) # Nr of clusters
>>> labels = db.labels_
>>> outliers = X[labels == -1] # find the outliers
>>> cluster1 = X[labels == 0] # Get the contents of each cluster
>>> cluster2 = X[labels == 1]
>>> cluster3 = X[labels == 2]
>>> unique_labels = set(labels)
>>> colors = ['y', 'b', 'g', 'r']
>>> for k, col in zip(unique_labels, colors): if k == -1: col = 'k'
class_member_mask = (labels == k)
xy = X[class_member_mask & core_samples_mask]
plt.plot(xy[:, 0], xy[:, 1], 'o', markerfacecolor=col,
markeredgecolor='k',markersize=6)
xy = X[class_member_mask & ~core_samples_mask]
plt.plot(xy[:, 0], xy[:, 1], 'o', markerfacecolor=col,
markeredgecolor='k', markersize=6)
>>> plt.title('number of clusters: %d' %n_clusters_)
>>> plt.show()
```



Using DBSCAN Clustering

Source of the image: Zöller (2022).

REVIEW STUDY GOALS

- Know the definitions and terms used for clustering
- Comprehend common applications of clustering analysis
- Understand different methods for clustering analysis
- Analise the advantages and limitations of the clustering methods
- Implement clustering methods in Python

SESSION 2

TRANSFER TASK

Create the dataset using the following commands:

```
>>> from sklearn.datasets import make_blobs
>>> dataset,_ = make_blobs(n_samples=100, random_state=12)
```

Partition the data points into 3 clusters by using:

1. K-Means clustering
2. GMM clustering
3. Agglomerative clustering
4. DBSCAN with $\varepsilon=1$ and MinPts=10

TRANSFER TASK
PRESENTATION OF THE RESULTS

Please present your results.
The results will be discussed in plenary.

1. Which of the following clustering algorithms suffers from the problem of convergence to local optima?

    a) Agglomerative clustering
    b) Only Expectation-Maximization clustering
    c) K- Means clustering and expectation-maximization clustering
    d) DBSCAN

2. For clustering to be applied, what is the minimum number of variables/ features required?

a) 0
b) 1
c) 2
d) 3

3. Is it possible that assignment of observations to clusters does not change between successive iterations in K-Means?

a) Yes
b) No
c) Cannot say
d) It depend on the number of data points

4. What could be the possible reason(s) for producing two different dendrograms using the agglomerative clustering algorithm for the same dataset?

a) Proximity function used
b) Number of data points used
c) Number of variables used
d) All these options are possible

5. K-Means clustering with Euclidean distance measure produces which kind of cluster shapes?

a) Spherical
b) Ellipsoid
c) Arbitrary
d) Cubic

LIST OF SOURCES

Text:

Carrasco, O.C. (2019). Gaussian Mixture Models Explained. https://towardsdatascience.com/gaussian-mixture-models-explained-6986aaf5a95

Das, J., Mukherjee, P., Majumder, S., & Gupta, P. (2014). Clustering-Based recommender system using principles of voting theory. *International Conference on Contemporary Computing and Informatics (IC3I 2014).*

Data Science Team (2020). A Layman's Guide to K Means Clustering. https://datascience.eu/machine-learning/k-means-clustering-in-machine-learning/

Ester, M., Kriegel, H., Sander, J., & Xiaowei, X. (1996). A density-based algorithm for discovering clusters in large spatial databases with noise. *Proceedings of the Second International Conference on Knowledge Discovery and Data Mining,* 226—231. https://www.aaai.org/Papers/KDD/1996/KDD96-037.pdf

Pai, P. (2021). Hierarchical clustering explaned. https://towardsdatascience.com/hierarchical-clustering-explained-e59b13846da8

Pragathi, K., Jayanthi, T., & Malathi V. (2018). Cluster based segmentation using K-Means algorithm. *International Journal of Data Mining Techniques and Applications*, *7*(1), 64—67. http://ijdmta.com/abstract_temp.php?id=V7-I1-P13

Zhu, A. (2022). 7 most asked questions on K-Means Clustering. https://towardsdatascience.com/explain-ml-in-a-simple-way-k-means-clustering-e925d019743b

Image:
Ernst (2017).
Zöller (2022).

File:KMeans-density-data.svg. (2020, October 10). *In Wikimedia Commons, the free media repository*. Retrieved, January 25, 2023, from https://commons.wikimedia.org/w/index.php?title=File:KMeans-density-data.svg&oldid=485735167.

File:EM Clustering of Old Faithful data.gif. (2020, September 23). *In Wikimedia Commons, the free media repository*. Retrieved, January 25, 2023, from https://commons.wikimedia.org/w/index.php?title=File:EM_Clustering_of_Old_Faithful_data.gif&oldid=469170019.

File:UPGMA Dendrogram Hierarchical.svg. (2020, October 31). *In Wikimedia Commons, the free media repository*. Retrieved, January 25, 2023, from https://commons.wikimedia.org/w/index.php?title=File:UPGMA_Dendrogram_Hierarchical.svg&oldid=508481311.

File:DBSCAN-Illustration.svg. (2020, October 7). *In Wikimedia Commons, the free media repository*. Retrieved , January 25, 2023, from https://commons.wikimedia.org/w/index.php?title=File:DBSCAN-Illustration.svg&oldid=483221815.

File:K Means Example Step 1.svg. (2020, September 4). *In Wikimedia Commons, the free media repository*. Retrieved, January 26, 2023, from https://commons.wikimedia.org/w/index.php?title=File:K_Means_Example_Step_1.svg&oldid=448124955.

File:K Means Example Step 2.svg. (2020, September 4). *In Wikimedia Commons, the free media repository*. Retrieved, January 26, 2023, from https://commons.wikimedia.org/w/index.php?title=File:K_Means_Example_Step_2.svg&oldid=448124977.

File:K Means Example Step 3.svg. (2020, September 4). *In Wikimedia Commons, the free media repository*. Retrieved, January 26, 2023, from https://commons.wikimedia.org/w/index.php?title=File:K_Means_Example_Step_3.svg&oldid=448124994.

File:K Means Example Step 4.svg. (2020, September 4). *In Wikimedia Commons, the free media repository*. Retrieved, January 26, 2023, from https://commons.wikimedia.org/w/index.php?title=File:K_Means_Example_Step_4.svg&oldid=448125010.