

# EPISEN – ING3. SI

## Machine Learning



**Abdallah EL HIDALI**

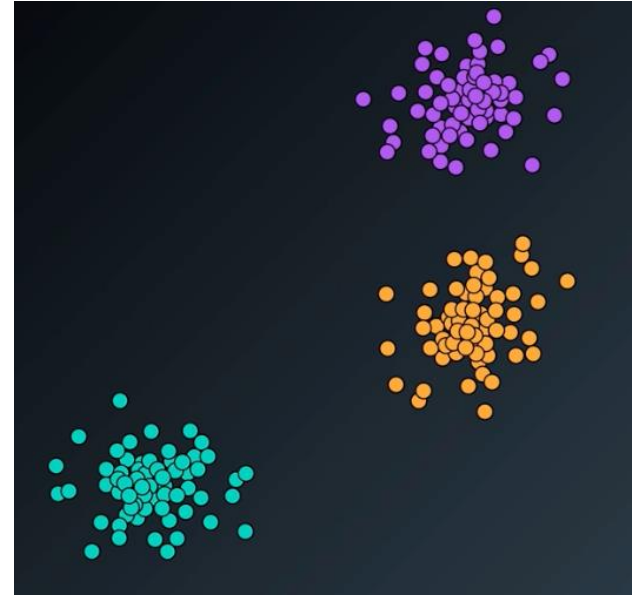
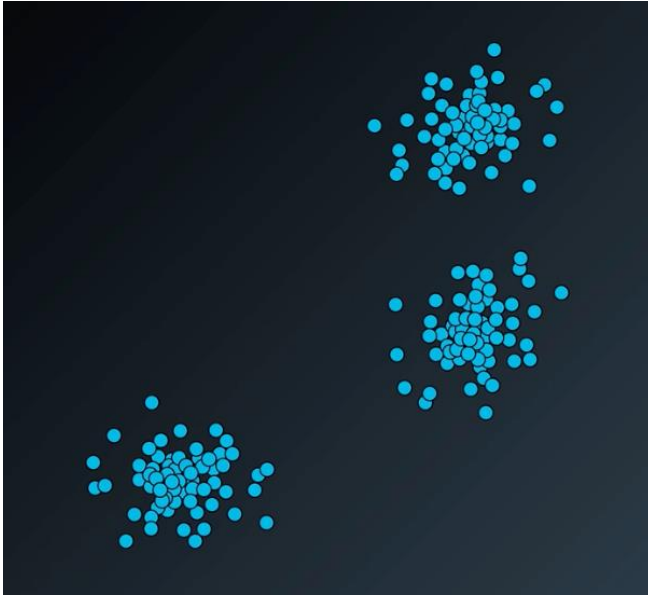
Tech Lead Sita For Aircraft  
abdallah.el-hidali@sit.aero

**EPISEN**

2024/2025

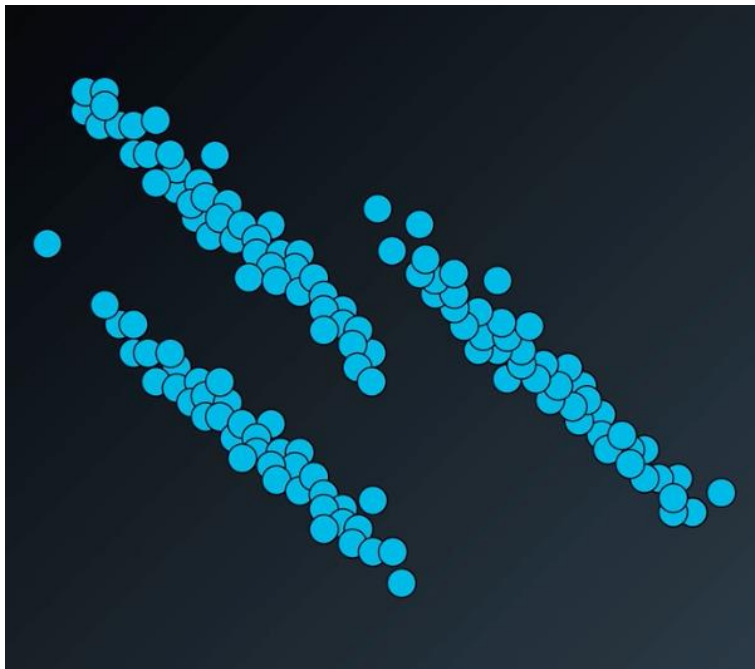
# **XI. Clustering hiérarchique et clustering basé sur la densité**

# Les limites du K-means



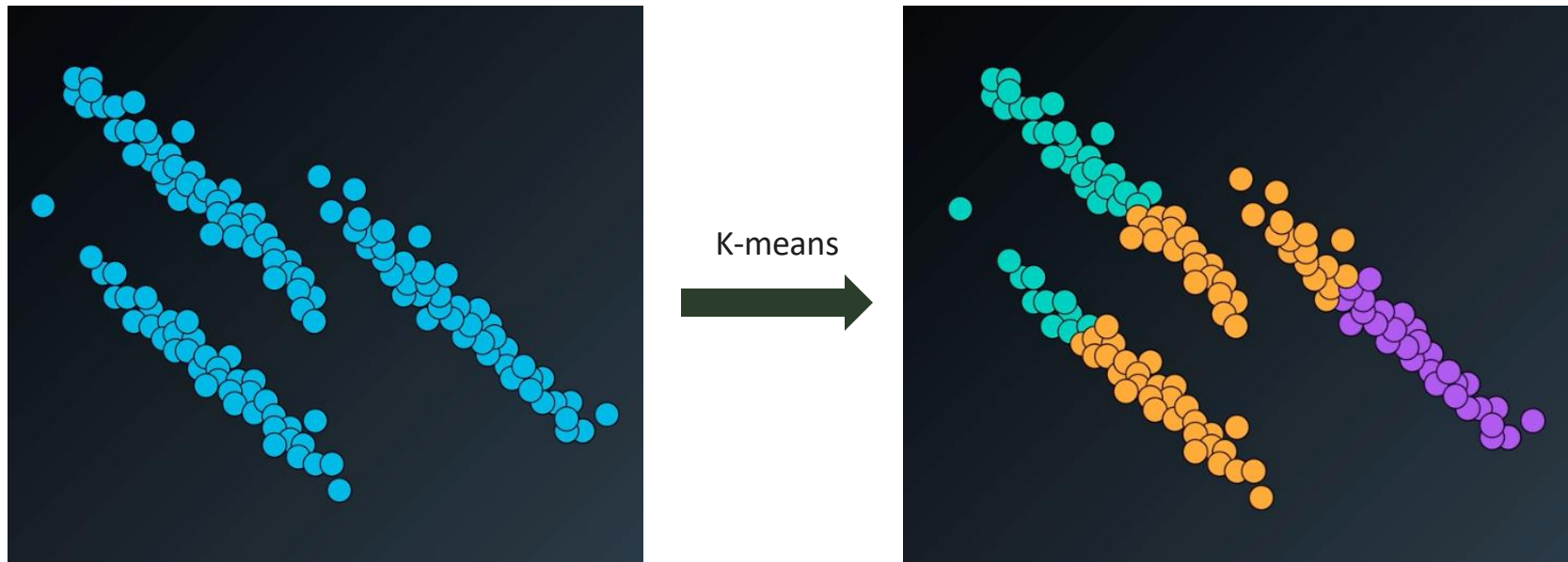
K-means peut facilement trouver les 3 clusters dans cet exemple

# Les limites du K-means



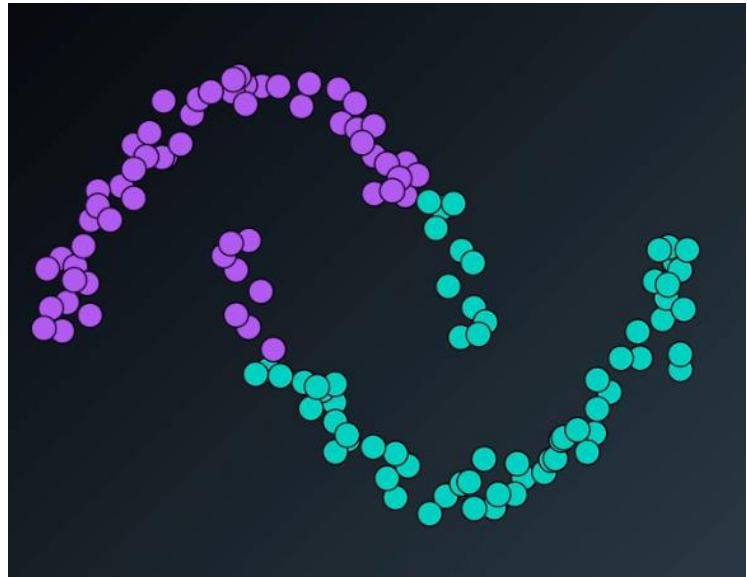
Mais qu'en est-il de cet exemple ? Est-ce que K-means pourrait retrouver les 3 clusters ?

# Les limites du K-means



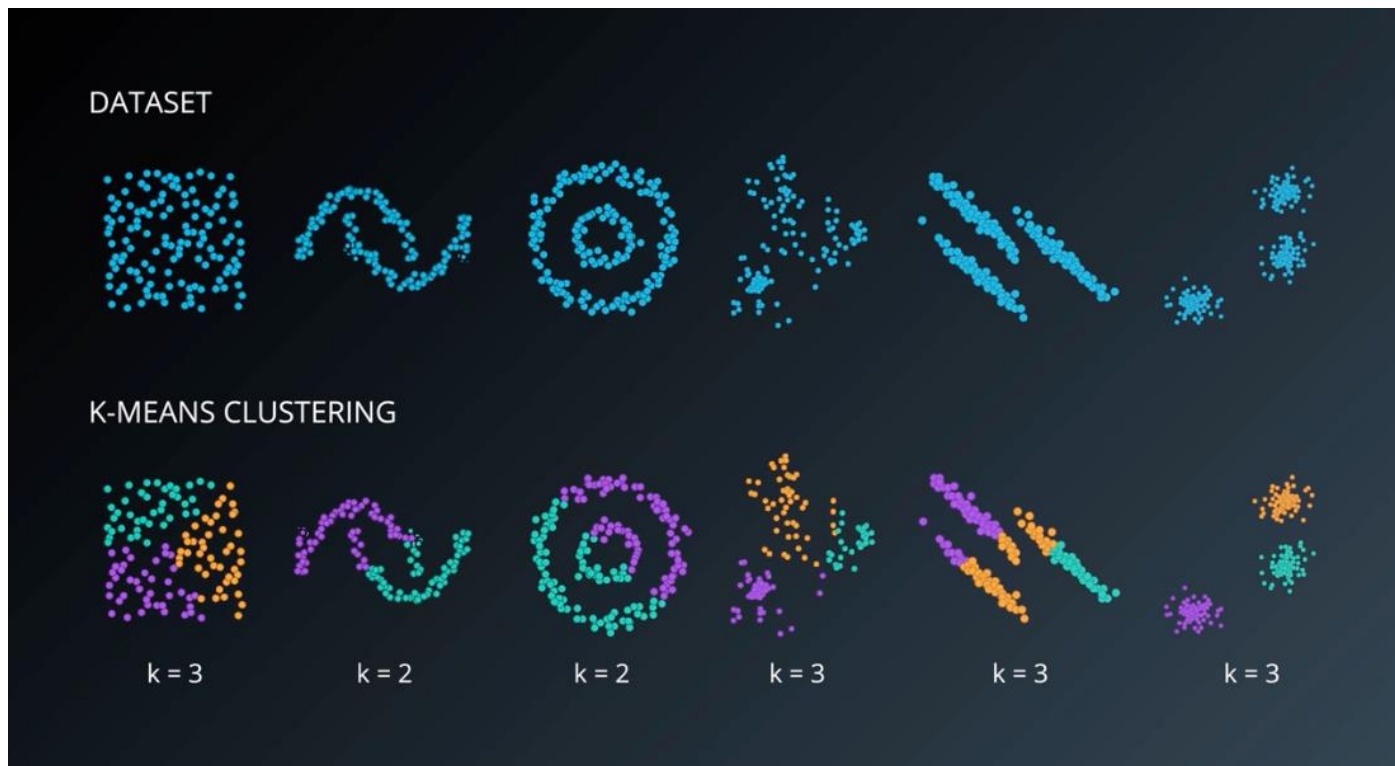
K-means n'arrive pas à identifier les 3 clusters de cet exemple car il se base sur la distance comme moyen de les séparer.

# Les limites du K-means



Pour cet exemple également, K-means n'arrive pas à bien séparer les deux clusters.

# Les limites du K-means



Nous allons essayer les algorithmes de clustering sur différents jeux de données pour comparer leurs performances.

# Aperçu des autres méthodes de clustering

HIERARCHICAL CLUSTERING



DENSITY CLUSTERING

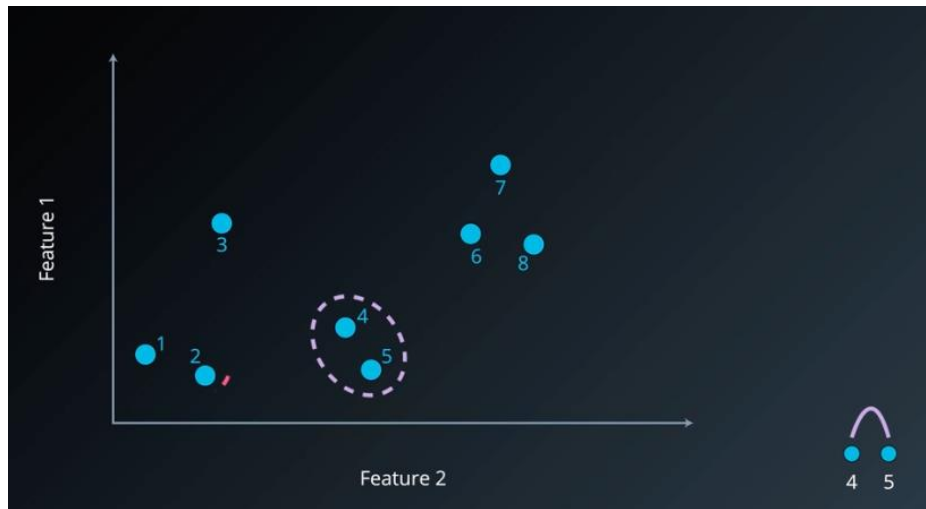
DBSCAN





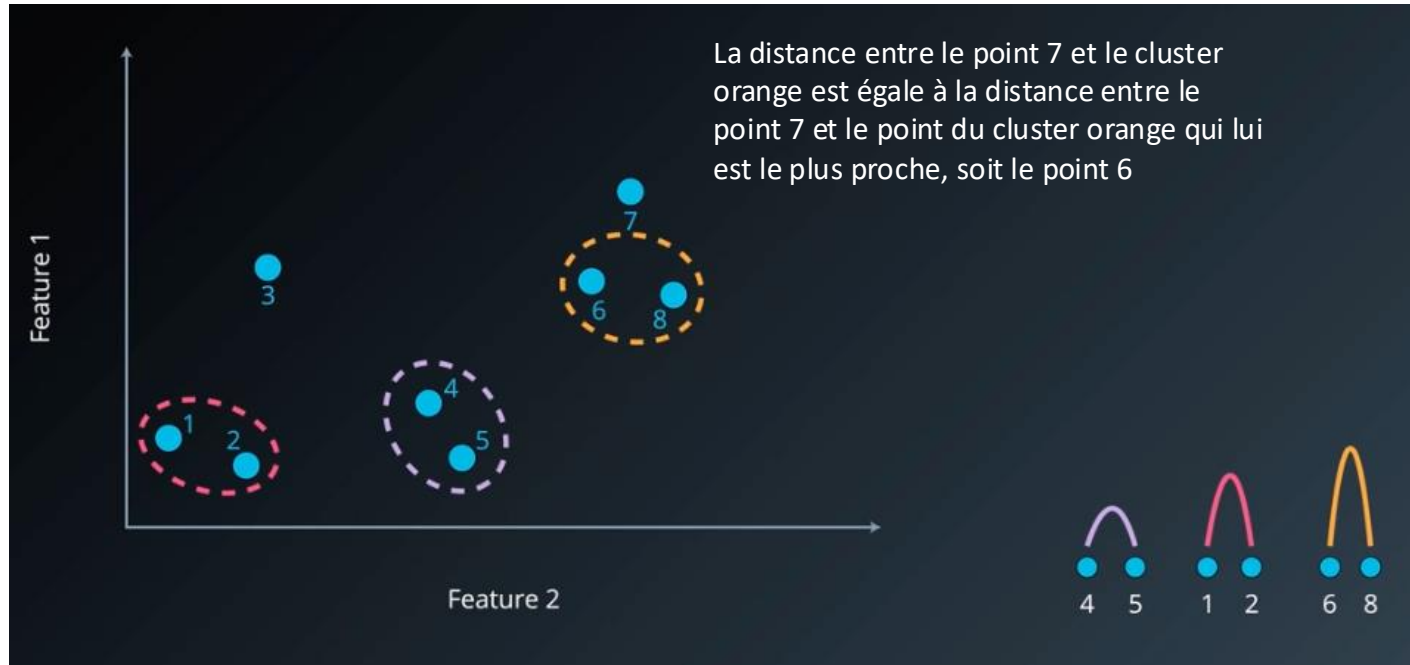
# Clustering hiérarchique

On considère chaque point  
comme un cluster

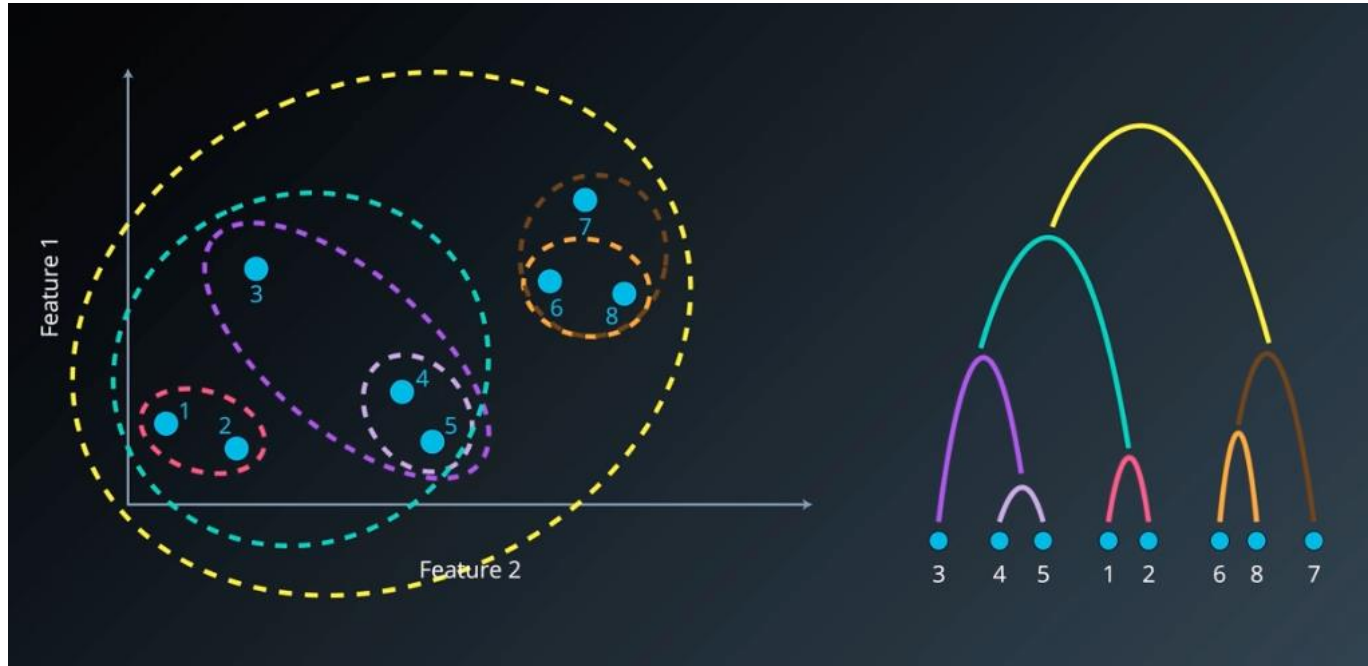


On calcule la distance entre chaque point et le  
reste des points puis on regroupe les 2 points  
avec la distance la plus faible dans le même  
cluster

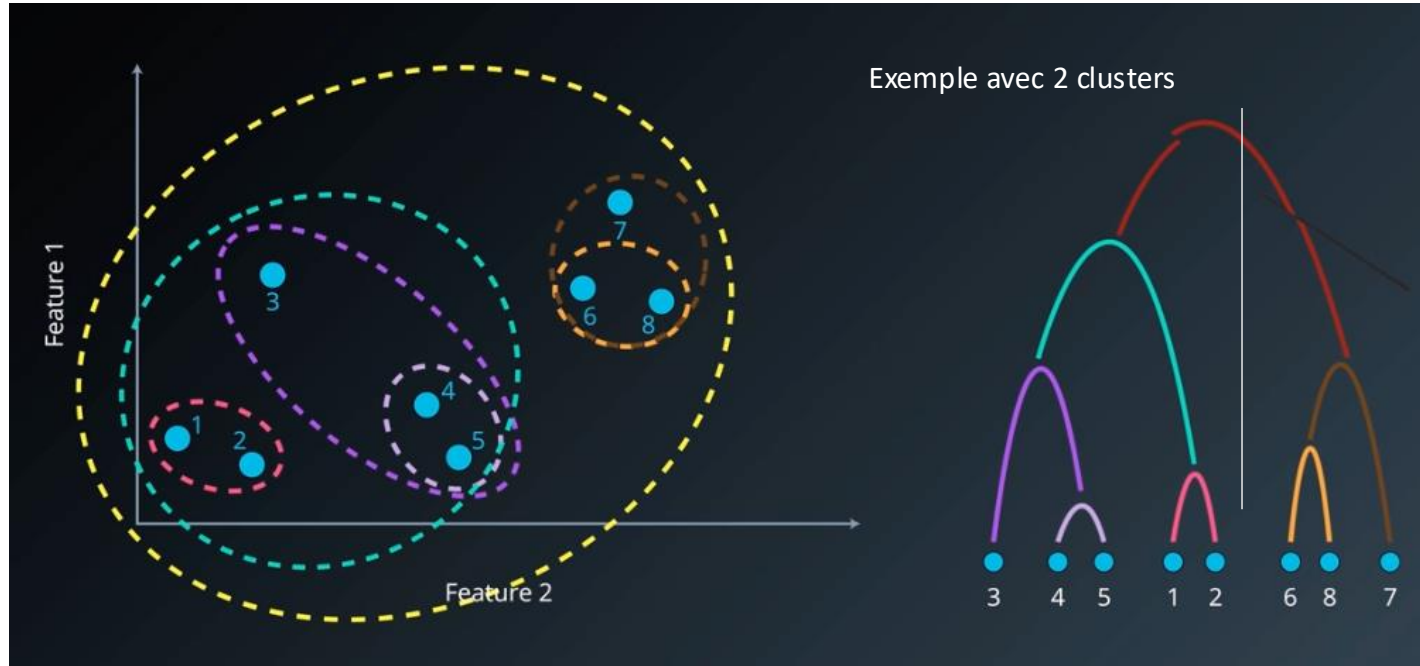
# Clustering hiérarchique



# Clustering hiérarchique



# Clustering hiérarchique



# Comparaison des méthodes de clustering

## K-MEANS CLUSTERING



k = 3



k = 2



k = 2



k = 3

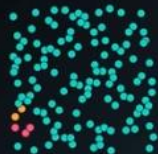


k = 3



k = 3

## SINGLE LINK HIERARCHICAL CLUSTERING



# Clustering hiérarchique sur sklearn

## HIERARCHICAL CLUSTERING | IMPLEMENTATION

```
from sklearn import datasets, cluster

# Load dataset
X = datasets.load_iris().data[:10]

# Specify the parameters for the clustering. 'ward' linkage
# is default. Can also use 'complete' or 'average'.
clust = cluster.AgglomerativeClustering(n_clusters=3, linkage='ward')

labels = clust.fit_predict(X)

# 'labels' now contains an array representing which cluster
# each point belongs to:
# [1 0 0 0 1 2 0 1 0 0]
```

# Clustering hiérarchique

## HIERARCHICAL CLUSTERING | IMPLEMENTATION

```
from scipy.cluster.hierarchy import dendrogram, ward, single
from sklearn import datasets
import matplotlib.pyplot as plt
```

```
# Load dataset
X = datasets.load_iris().data[:10]
```

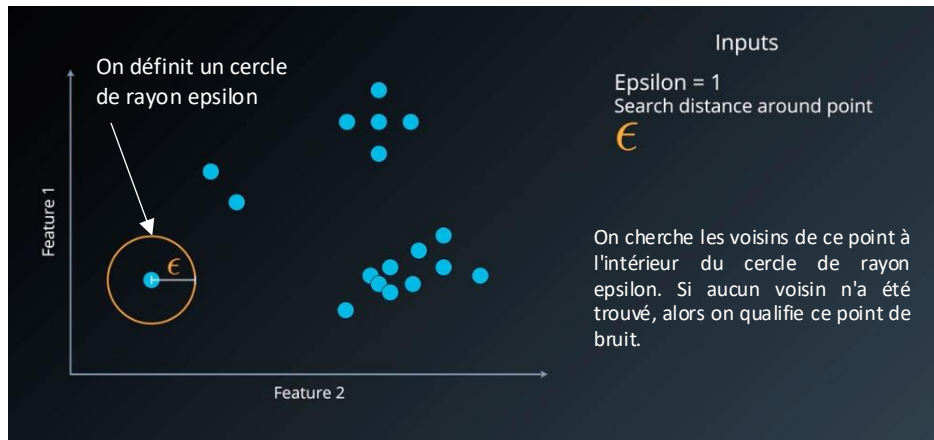
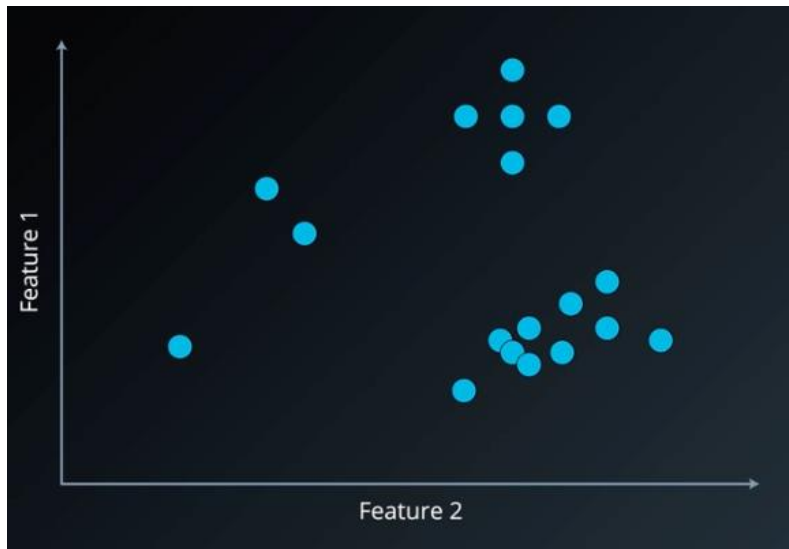
```
# Perform clustering
linkage_matrix = ward(X)
```

```
# Plot dendrogram
dendrogram(linkage_matrix)
```

```
plt.show()
```



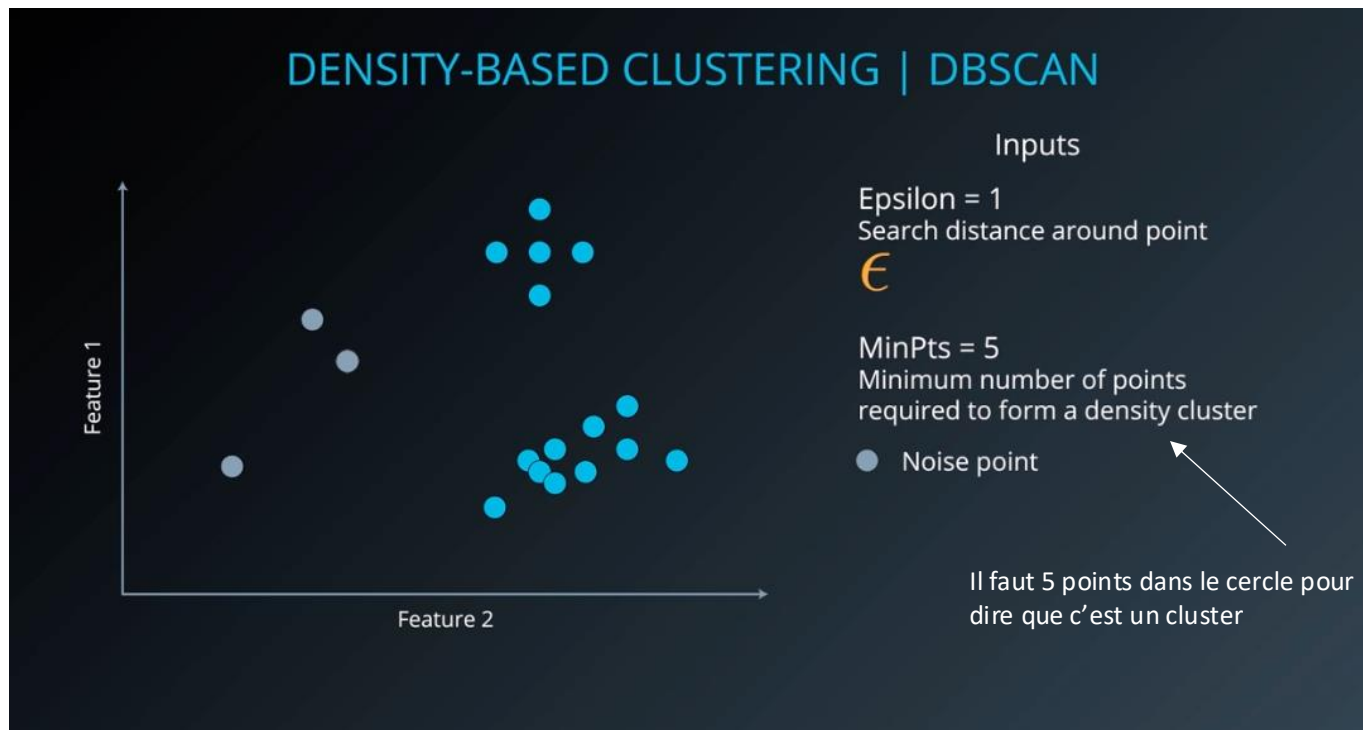
# DBSCAN: Density-Based Spatial Clustering of Applications with Noise



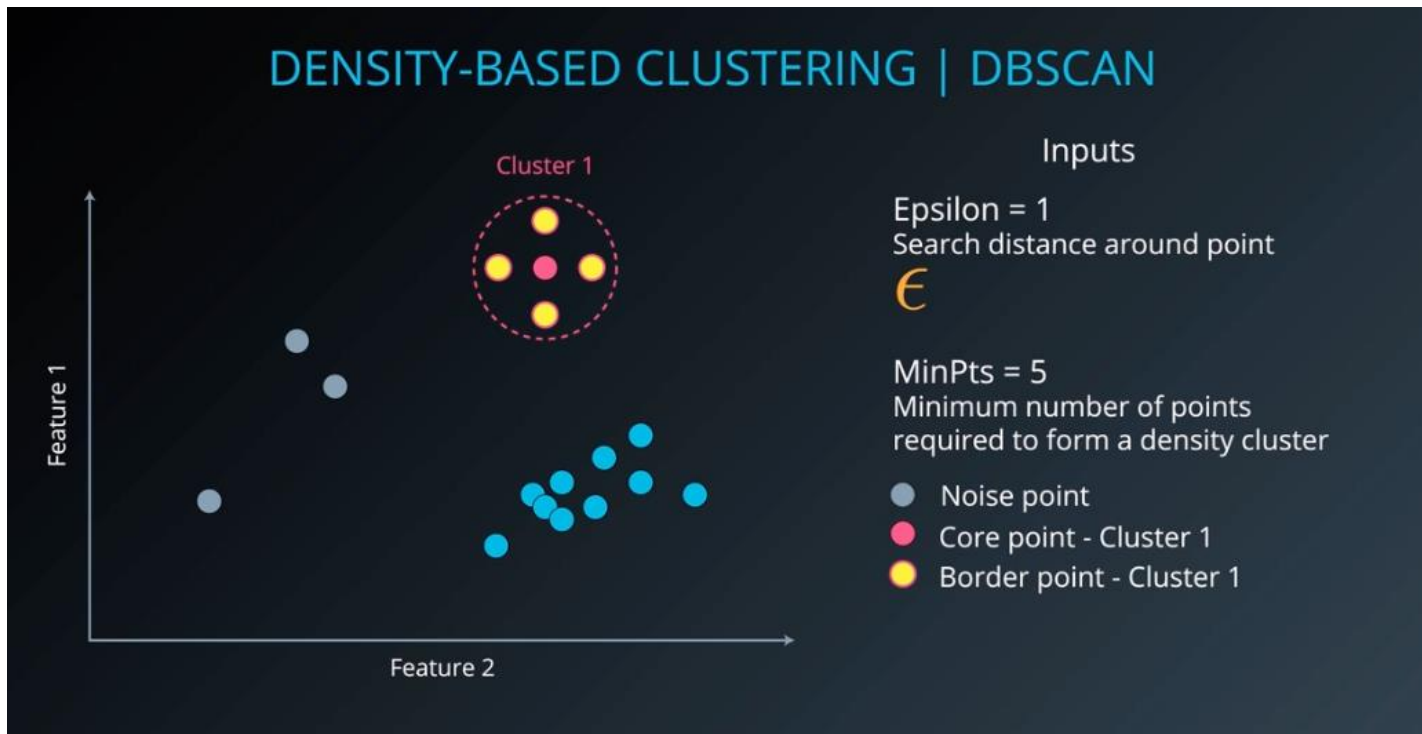
<https://www.naftaliharris.com/blog/visualizing-dbscan-clustering/>



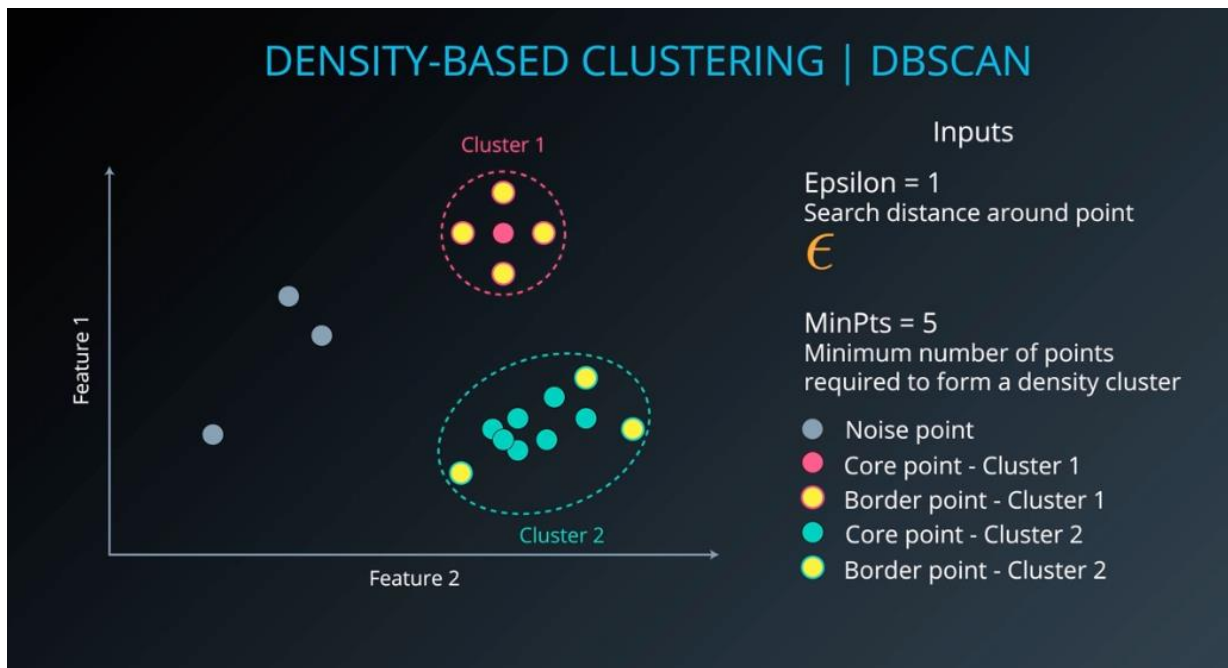
# DBSCAN: Density-Based Spatial Clustering of Applications with Noise



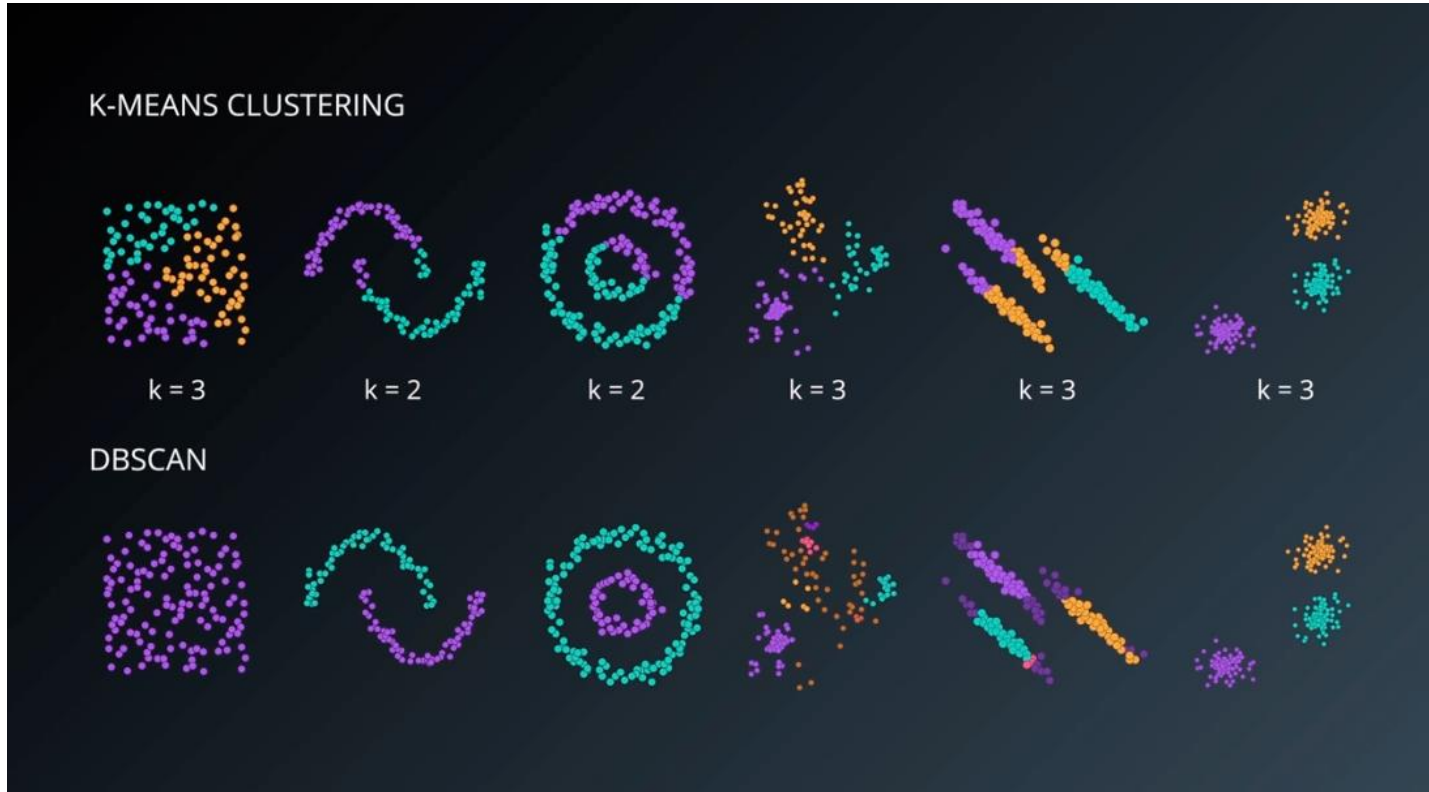
# DBSCAN: Density-Based Spatial Clustering of Applications with Noise



# DBSCAN: Density-Based Spatial Clustering of Applications with Noise



# Comparaison des méthodes de clustering



# DBSCAN sur sklearn

## DENSITY-BASED CLUSTERING | DBSCAN IMPLEMENTATION

```
from sklearn import datasets, cluster

# Load dataset
X = datasets.load_iris().data

# Specify the parameters for the clustering. These are the defaults.
db = cluster.DBSCAN(eps=0.5, min_samples=5)
db.fit(X)

'''
'db.labels_' now contains an array representing which cluster
each point belongs to. Samples labeled '-1' are noise.
array([ 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])
'''
```

# DBSCAN: exercice

[https://github.com/elhidali/EPISEN-2024/tree/main/exercice\\_session\\_5/dbscan](https://github.com/elhidali/EPISEN-2024/tree/main/exercice_session_5/dbscan)