

Cuantizare vectorială - K-means.  
Tehnici de dithering.  
Învățare nesupervizată. DBSCAN

# Cuprins

- **Cuantificare scalară vs. cuantificare vectorială**
- **K-Means – ideea generală**
- **Strategii de initializare**
- **Performanțe K-Means**
- **Implementare în Python – exemplu 2D**
- **Biblioteca sklearn**
- **Cuantizare vectorială – compresie**
- **K-Means – clasificarea cifrelor scrise**
- **Limitări**
- **DBSCAN – idee generală**
- **DBSCAN (sklearn)**
- **Tehnici de dithering**

# Cuantificare scalară vs. cuantificare vectorială

Cuantificare – reducerea numărului de valori de reprezentare pentru o mărime (scalară sau vectorială).

Reprezentăm toate valorile dintr-un interval dat  $X$ , cu valorile din mulțimea

$$Q = \{q_1, q_2 \dots q_N\}$$

$N$  puncte de cuantificare – necesită  $\log_2 N$  biți.

Exemplu de cuantificare pentru valorile  $x \in [0 - 255]$

$$Q = \{0, 64, 128, 192\} \rightarrow Q(x) = \left[ \frac{x}{64} \right] * 64$$

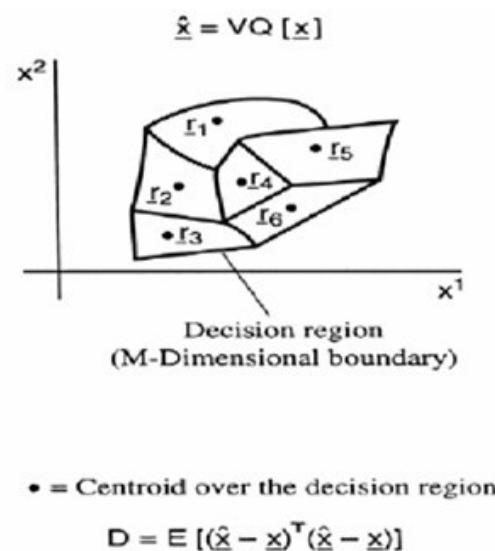
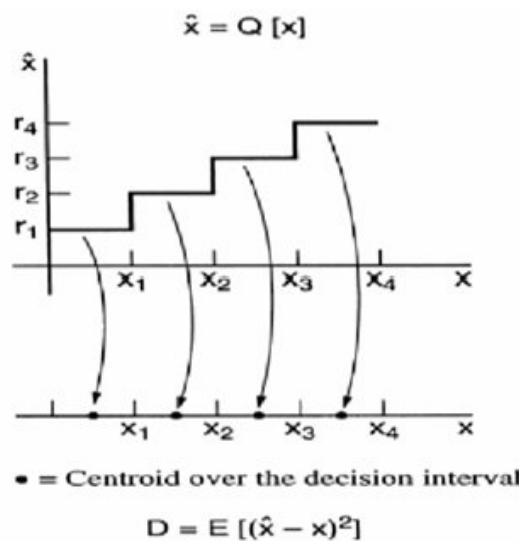
$$Q' = \{32, 96, 160, 224\} \rightarrow Q'(x) = \left[ \frac{x}{64} \right] * 64 + 32$$

Eroarea (zgomot cuantificare):  $e(x) = Q(x) - x$

# Cuantificare scalară vs. cuantificare vectorială

Cuantificare vectorială – reducerea numărului de valori de reprezentare pentru o mărime (scalară sau vectorială).

## Scalar Quantization v.s. Vector Quantization



# Exemplu

2x2 PSNR=32.38 RATA=5.22



4x4 PSNR=27.95 RATA=16.00

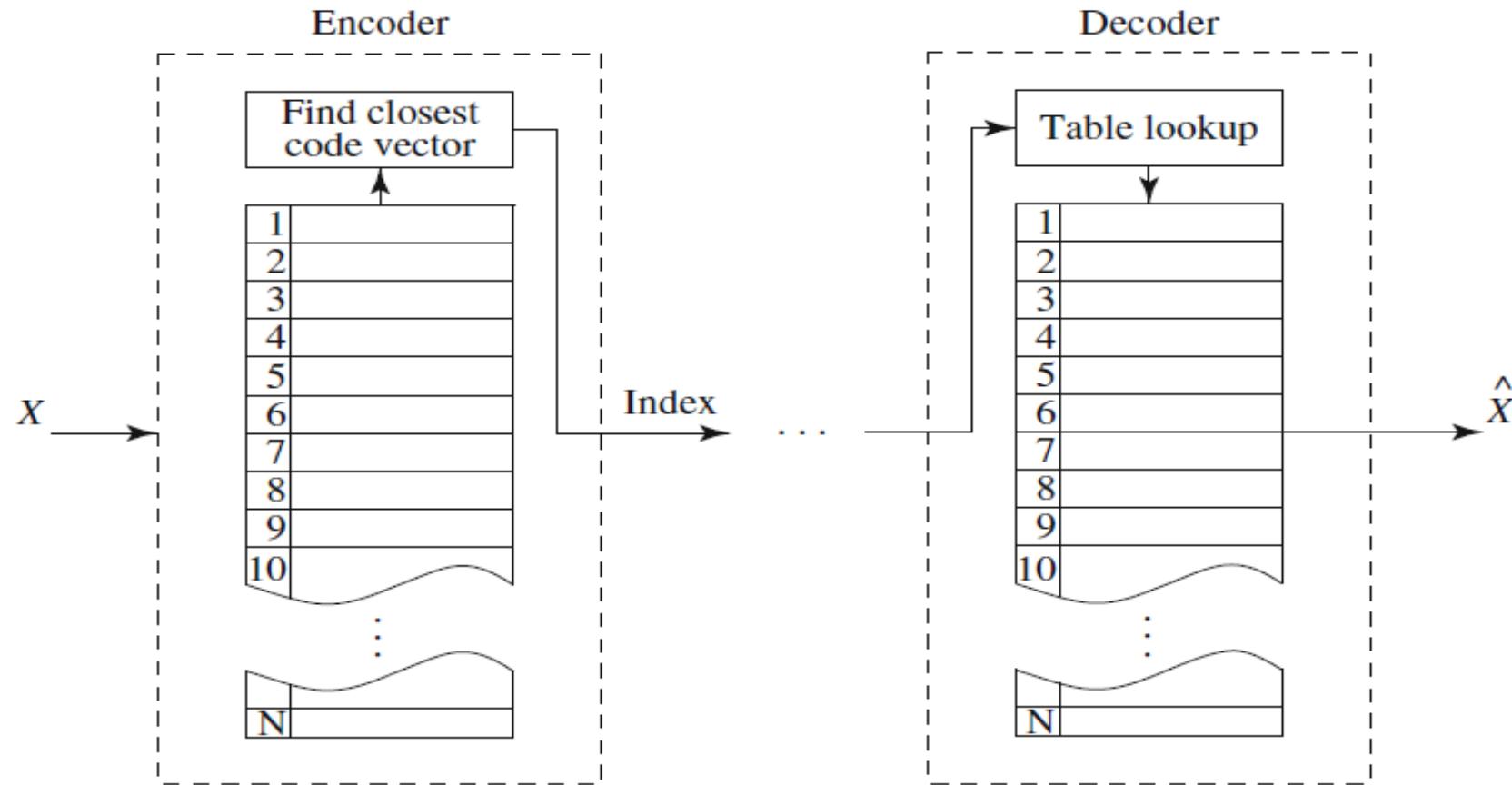


8x8 PSNR=25.36 RATA=13.47



# Cuantificare scalară vs. cuantificare vectorială

Cuantizare vectorială – Cuantizare globală prin vectori reprezentativi.

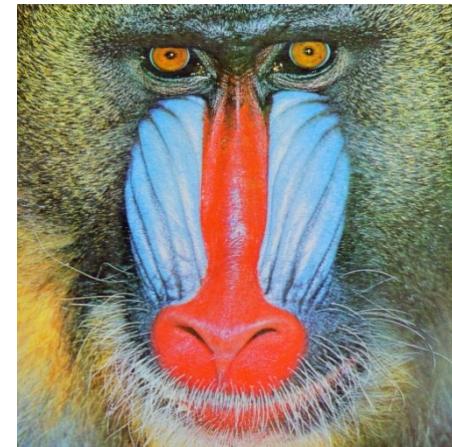


# Cuantificare scalară vs. cuantificare vectorială

VQ	SQ
Simbolurile (ex. pixeli) sunt grupate și procesate împreună.	Fiecare intrare este tratată separat, independent.
Zgomot de cuantizare mai mic.	-
Mai eficient	Mai puțin eficient.
Eroare este afectată atât de forma cât și de mărimea intervalului de cuantificare.	Eroare este afectată doar de mărimea intervalului de cuantificare.
VQ poate fi utilizat în aplicații în care rezoluția redusă este suficientă – compresie de imagine.	În compresia a de imagine, SQ produce artefacte neplăcute.
Pentru orice rată de compresie – erorile sunt mai mici decât SQ.	Pentru orice rată de compresie – erorile sunt mai mari decât VQ.

# Imagini standard

Pentru testarea și compararea algoritmilor de prelucrare de imagini se utilizează “imagini standard”. O listă de astfel de imagini se găsește la adresa [https://www.imageprocessingplace.com/root\\_files\\_V3/image\\_databases.htm](https://www.imageprocessingplace.com/root_files_V3/image_databases.htm)



# Imagini standard

Imagini controversate – **Lena**. Există reviste care recomandă utilizarea altor imagini, sau chiar refuză publicarea de rezultate demonstrează cu ajutorul imaginii Lena. Utilizată încă din 1973, dar nu pentru că este o imagine din Playboy.

*Alexander Sawchuk estimates that it was in June or July of 1973 when he, then an assistant professor of electrical engineering at the USC Signal and Image Processing Institute (SIPI), along with a graduate student and the SIPI lab manager, was hurriedly searching the lab for a good image to scan for a colleague's conference paper. They had tired of their stock of usual test images, dull stuff dating back to television standards work in the early 1960s. They wanted something glossy to ensure good output dynamic range, and they wanted a human face. Just then, somebody happened to walk in with a recent issue of Playboy. The engineers tore away the top third of the centerfold so they could wrap it around the drum of their Muirhead wirephoto scanner, which they had outfitted with analog-to-digital converters (one each for the red, green, and blue channels) and a Hewlett Packard 2100 minicomputer. The Muirhead had a fixed resolution of 100 lines per inch and the engineers anted a 512 × 512 image, so they limited the scan to the top 5.12 inches of the picture, effectively cropping it at the subject's shoulders.*

Hutchison, Jamie (2001). ["Culture, Communication, and an Information Age Madonna"](#) (PDF). IEEE Professional Communication Society Newsletter. **45** (3): 1, 5–7

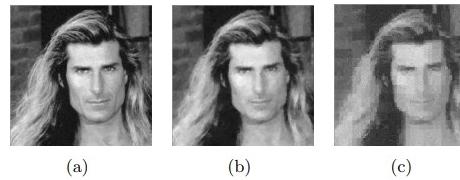
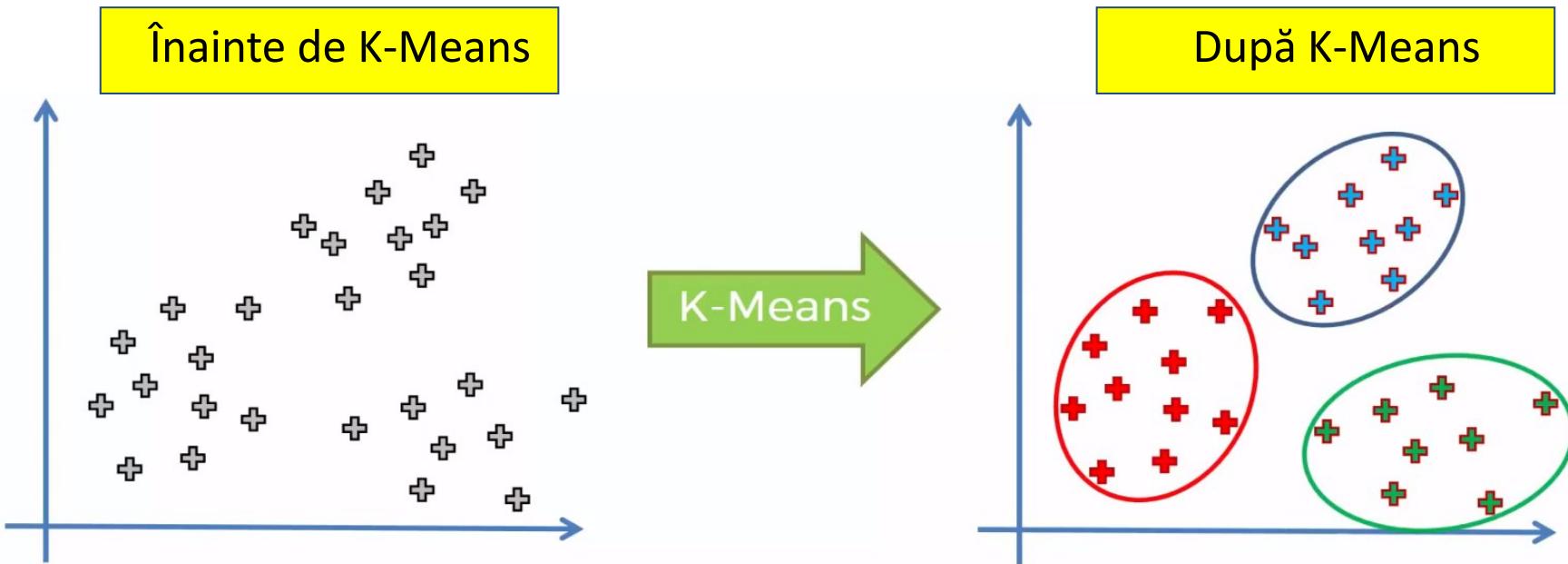


Figure 2: (a) Original 256 × 256 Fabio image corrupted with Gaussian noise and its reconstruction from 20% of its Fourier coefficients using (b) total variation minimization and (c)  $\ell_1$ -minimization of its bivariate Haar coefficients

<https://arxiv.org/pdf/1202.6429.pdf>

# K-Means – ideea generală

Pentru o mulțime de vectori din  $\mathcal{R}^d$  să se găsească ***n* vectori reprezentativi**. Fiecare vector inițial va fi aproximat printr-un vector reprezentativ (**centroid**)  
Punctele approximate prin același vector ⇒ **un cluster**



# K-Means – ideea generală

Intrări : N vectori  $x_i$ . Ieșiri : n vectori repr.  $\mu_j$

$$D = \sum_{j=1}^n D_j = \sum_{j=1}^n \sum_{x_i \in \omega_j} \|x_i - \mu_j\|^2 = \sum_{j=1}^n \sum_{i=1}^N a_{ij} \|x_i - \mu_j\|^2$$
$$a_{ij} = \begin{cases} 1, & \text{daca } x_i \in \omega_j \\ 0, & \text{daca } x_i \notin \omega_j \end{cases}$$

**Minimizarea lui D** : trebuie determinate valorile  $\mu_j$  (centrele claselor) și valorile coeficientilor binari  $a_{ij}$ .

$$\frac{\partial D}{\partial \mu_j} = 2 \sum_{i=1}^N a_{ij} (\mu_j - x_i) = 0 \Rightarrow \mu_j = \frac{\sum_{x \in \omega_j} x}{|\omega_j|}$$

# K-Means – ideea generală

## 1. Inițializare

Se face o partiție aleatoare a setului de vectori și se calculează centroizii claselor inițiale

## 2. Căutare și actualizare centroid

Pentru fiecare vector din mulțimea dată se calculează distanțele sale la centroizii claselor

$$d_k = \|x_i - \mu_k\|^2 \quad k = \overline{1, n}$$

## 3. Actualizează centroidul

- Se mută vectorul în clasa corespunzătoare distanței minime și se actualizează centroidul.

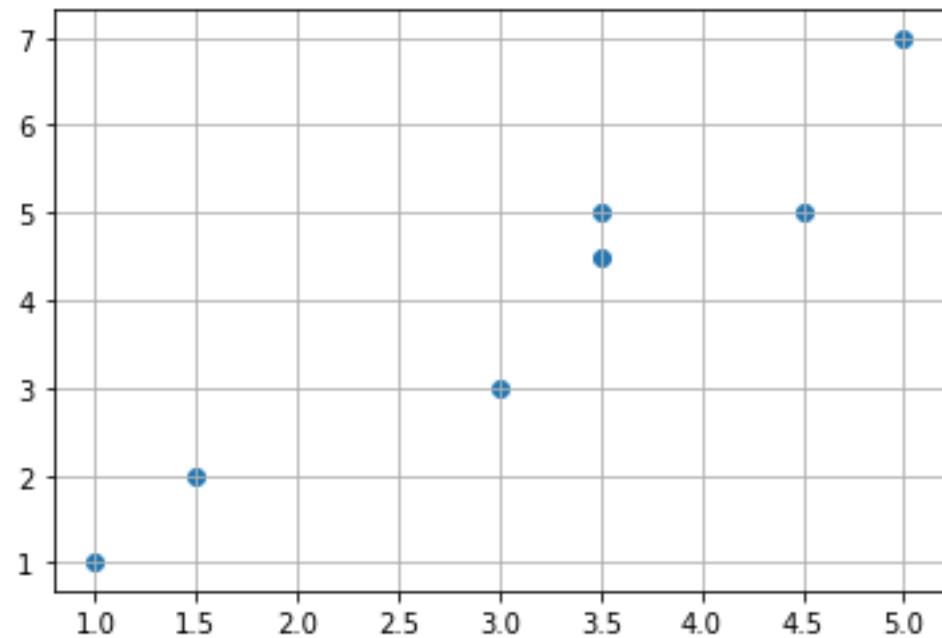
## 4. Repetă (2)-(3) până când până când nu mai apar modificări, adică nu se mai deplasează centroizii, sau după un număr de iterații.

# K-Means – pas cu pas

## Initializare

- 2 centroizi

Vectori	X	Y
1	1.0	1.0
2	1.5	2.0
3	3.0	4.0
4	5.0	7.0
5	3.5	5.0
6	4.5	5.0
7	3.5	4.5

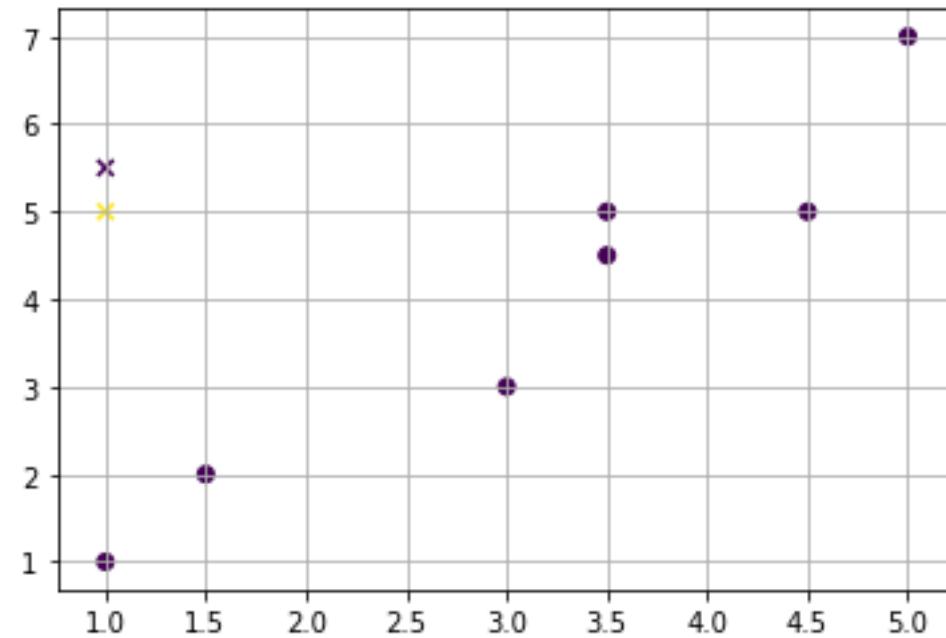


# K-Means – pas cu pas

## 1. Inițializare

- 2 centroizi

Vectori	X	Y
1	1.0	1.0
2	1.5	2.0
3	3.0	4.0
4	5.0	7.0
5	3.5	5.0
6	4.5	5.0
7	3.5	4.5

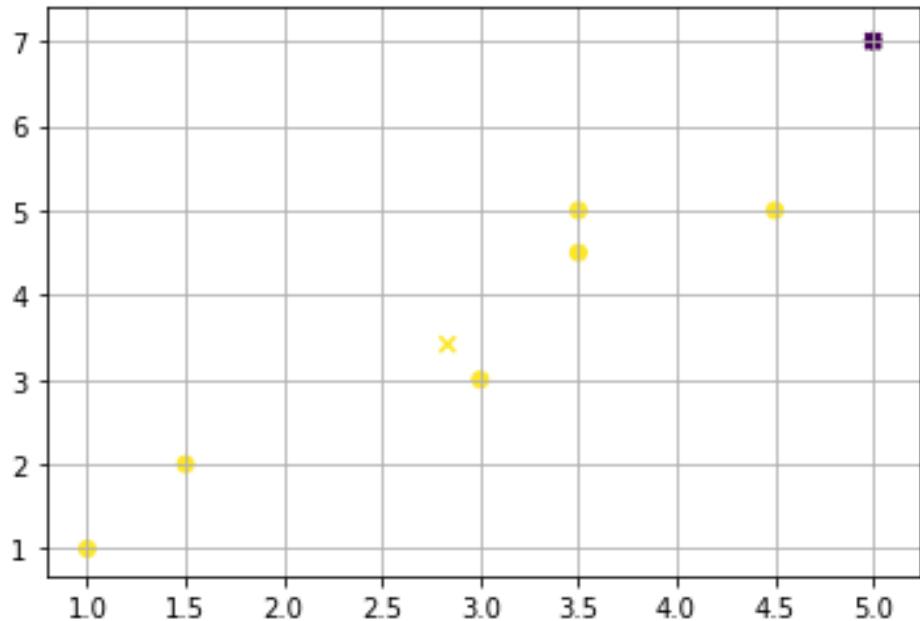


# K-Means – pas cu pas

## 2. Căutare 3. Actualizare centroid

Vectori	X	Y
1	1.0	1.0
2	1.5	2.0
3	3.0	4.0
4	5.0	7.0
5	3.5	5.0
6	4.5	5.0
7	3.5	4.5

	Grup 1		Grup 2	
Pas	Vectori	Centroid	Vectori	Centroid
1	1,2,3,5,6,7	(2.83, 3.41)	4	(5.0, 7.0)

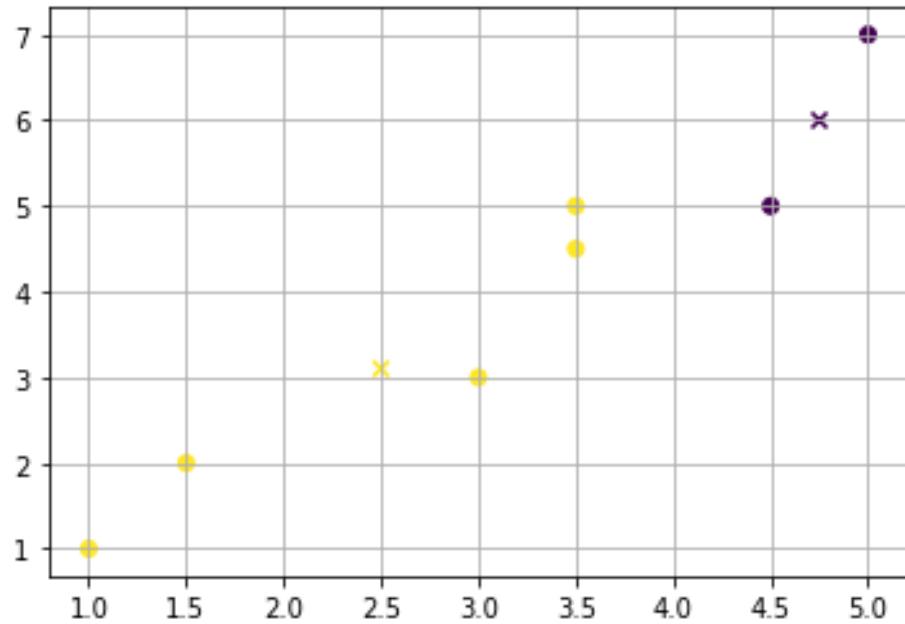


# K-Means – pas cu pas

## 2. Căutare 3. Actualizare centroid

Vectori	X	Y
1	1.0	1.0
2	1.5	2.0
3	3.0	4.0
4	5.0	7.0
5	3.5	5.0
6	4.5	5.0
7	3.5	4.5

	Grup 1		Grup 2	
Pas	Vectori	Centroid	Vectori	Centroid
1	1,2,3,5,6,7	(2.83, 3.41)	4	(5.0, 7.0)
2	1,2,3,5,7	(2.5, 3.1)	4,6	(4.75, 6.0)

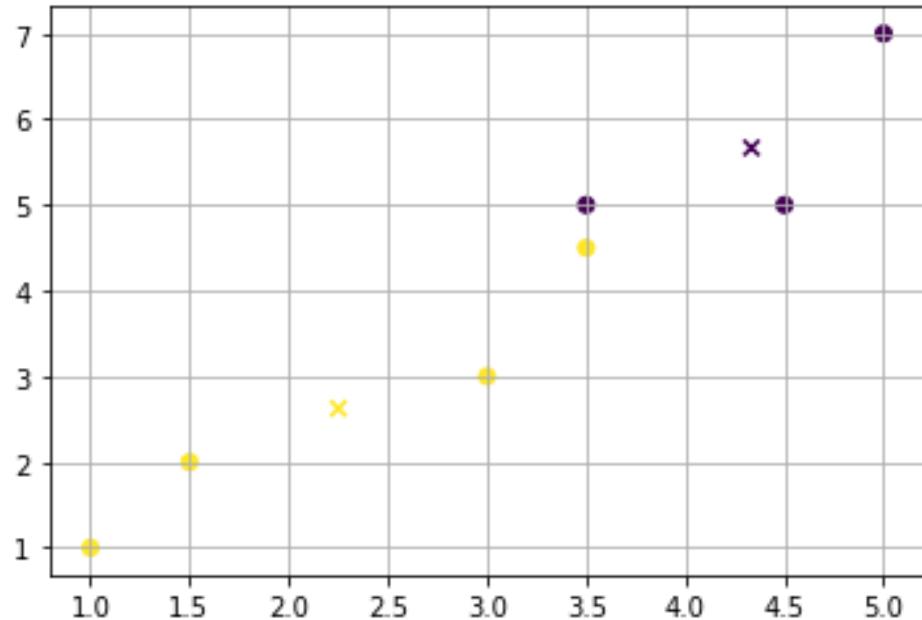


# K-Means – pas cu pas 2,3

## 2. Căutare 3. Actualizare centroid

Vorbitori	A	B
1	1.0	1.0
2	1.5	2.0
3	3.0	4.0
4	5.0	7.0
5	3.5	5.0
6	4.5	5.0
7	3.5	4.5

	Grup 1		Grup 2	
Pas	Vectori	Centroid	Vectori	Centroid
1	1,2,3,5,6,7	(2.83, 3.41)	4	(5.0, 7.0)
2	1,2,3,5,7	(2.5, 3.1)	4,6	(4.75, 6.0)
3	1,2,3,7	(2.25, 2.625)	4,5,6	(4.33, 5.66)

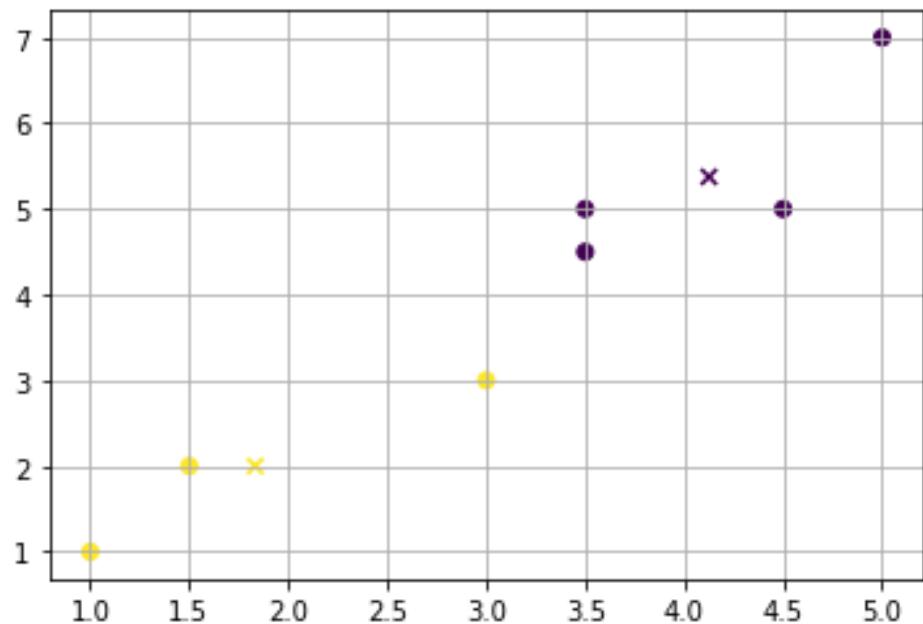


# K-Means – pas cu pas 2,3

## 2. Căutare 3. Actualizare centroid

Vorbitori	A	B
1	1.0	1.0
2	1.5	2.0
3	3.0	4.0
4	5.0	7.0
5	3.5	5.0
6	4.5	5.0
7	3.5	4.5

	Grup 1		Grup 2	
Pas	Vectori	Centroid	Vectori	Centroid
1	1,2,3,5,6,7	(2.83, 3.41)	4	(5.0, 7.0)
2	1,2,3,5,7	(2.5, 3.1)	4,6	(4.75, 6.0)
3	1,2,3,7	(2.25, 2.625)	4,5,6	(4.33, 5.66)
4	1,2,3	(1.83, 2.0)	4,5,6,7	(4.125, 5.375)



# K-Means : Inițializare inspirată



# K-Means : Inițializare neinspirată



# K-Means : Strategii de inițializare

Optim local dependent de inițializarea centroizilor:

- 1) Alegerea aleatoare a centroizilor (ex. dintre puncte)**
- 2) Euristica uniformă : se calculează cele mai îndepărtate puncte A și B : se împarte segmentul AB în  $n$  părți egale.**

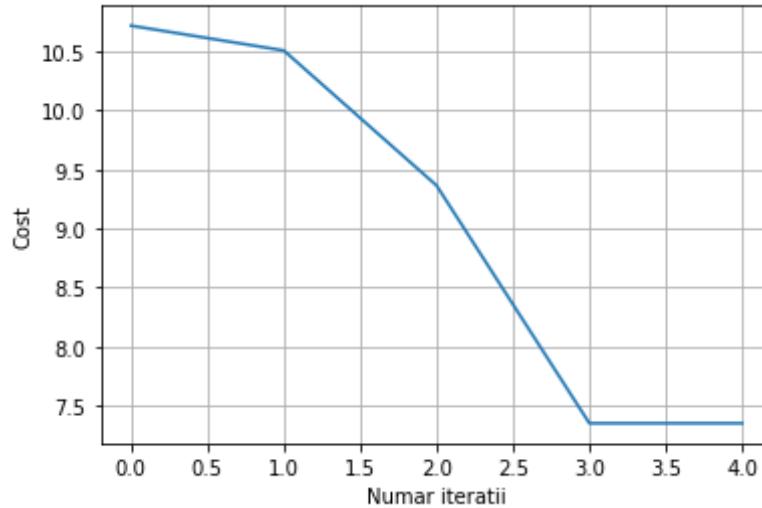
- 3) Euristica “cea mai îndepărtată”:**

- Se alege primul centroid la întâmplare dintre puncte
- Al doilea centroid este cel mai îndepărtat de primul
- ...
- Centroidul  $k$  (cel mai îndepărtat de restul celor aleși)

$$\arg\max\{d(i) = \max\{dist(x_i, \mu_j), j = \overline{1, k-1}\}, i = \overline{1, N}\}$$

# Performanțe K-Means

- Mai multe execuții și alegera celei mai bune. Cum ?



Complexitatea unei iterații :  $\theta(nNk)$

- $N$  numărul de puncte,  $n$  clustere,  $k$  dimensiune vector
- Numărul de iterații  $\propto$  configurație și de inițializare.
- Clustere bine delimitate și inițializare bună  $\Rightarrow$  convergență rapidă (număr mic de iterații)!

Cluster vid la un moment dat  $\Rightarrow$  reducerea nr. de clustere sau alocarea aleatoare a unor puncte în acel cluster ...

# Implementare Python

```
52     def kMeans(X, numCentroids):
53         color=[0.5, 0.8]
54         diff = 1
55         cluster = np.zeros(X.shape[0])
56         centroids = initCentroid(numCentroids)
57         while diff:
58             # for each observation
59             plt.figure()
60             plt.scatter(vec[:,0], vec[:,1],c=cluster)
61             plt.scatter(centroids[:,0],centroids[:,1],marker='x',c=color)
62             plt.grid()
63             plt.show()
64             print(centroids)
65             for i, row in enumerate(X):
66                 mn_dist = float('inf')
67                 # dist of the point from all centroids
68                 for idx, centroid in enumerate(centroids):
69                     d = np.sqrt((centroid[0]-row[0])**2 + (centroid[1]-row[1])**2)
70                     # store closest centroid
71                     if mn_dist > d:
72                         mn_dist = d
73                         cluster[i] = idx
74             new_centroids = computeNewCentroids(X, cluster, numCentroids)
75             # if centroids are same then leave
76             if np.count_nonzero(centroids-new_centroids) == 0:
77                 diff = 0
78             else:
79                 centroids = new_centroids
80             # plt.figure()
81             # plt.scatter(vec[:,0], vec[:,1],c=cluster)
82             # plt.scatter(centroids[:,0],centroids[:,1],marker='o')
83             # plt.show()
84         return centroids, cluster
85 
```

# Biblioteca sklearn

- Extensie pentru scipy
- Algoritmi pentru învățare supervizată și nesupervizată
- open-source
- din 2007

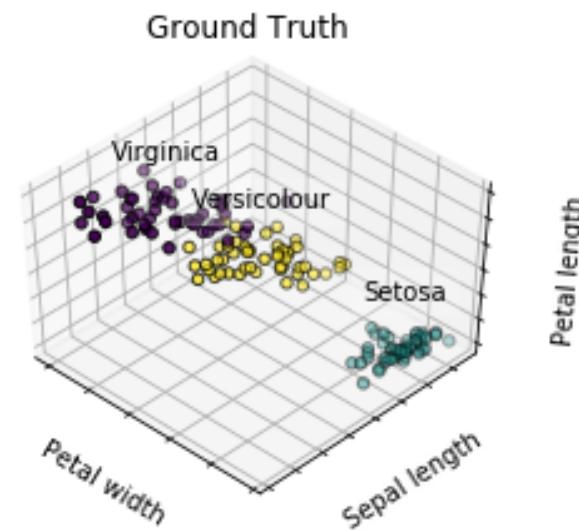
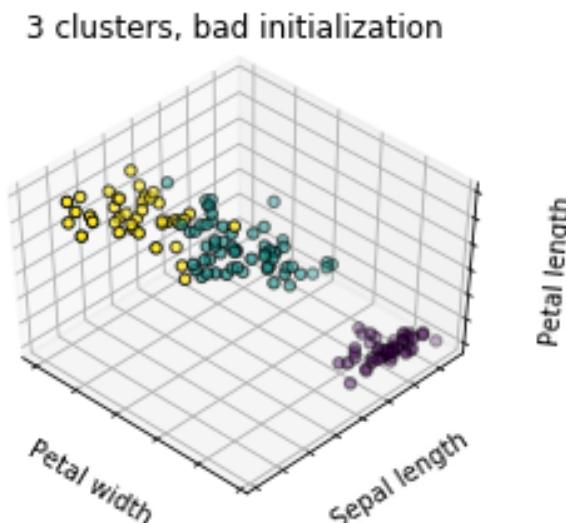
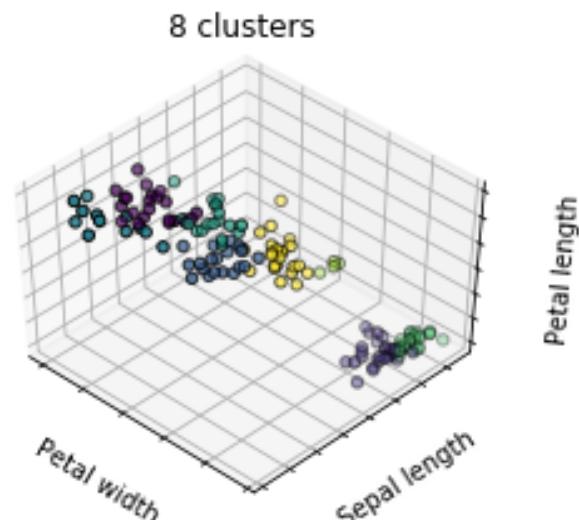
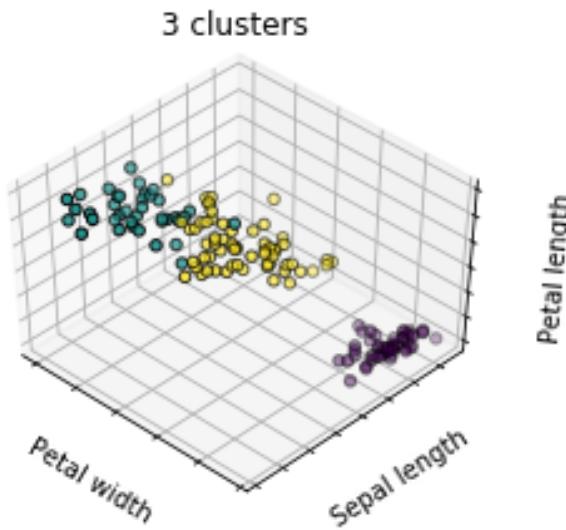


# Biblioteca sklearn - kmeans

```
11  import numpy as np
12  import matplotlib.pyplot as plt
13
14  # Though the following import is not directly being used, it is required
15  # for 3D projection to work
16  from mpl_toolkits.mplot3d import Axes3D
17
18  from sklearn.cluster import KMeans
19  from sklearn import datasets
20
21  np.random.seed(5)
22
23  iris = datasets.load_iris()
24  X = iris.data
25  y = iris.target
26
27  estimators = [
28      ("k_means_iris_8", KMeans(n_clusters=8)),
29      ("k_means_iris_3", KMeans(n_clusters=3)),
30      ("k_means_iris_bad_init", KMeans(n_clusters=3, n_init=1, init="random")),
31  ]
32
33
34  fignum = 1
35  titles = ["8 clusters", "3 clusters", "3 clusters, bad initialization"]
36  for name, est in estimators:
37      fig = plt.figure(fignum, figsize=(4, 3))
38      ax = Axes3D(fig, rect=[0, 0, 0.95, 1], elev=48, azim=134)
39      est.fit(X)
40      labels = est.labels_
```

<https://scikit-learn.org/stable/modules/generated/sklearn.cluster.KMeans.html#sklearn.cluster.KMeans>

# Biblioteca sklearn - kmeans



# Cuantificare Vectorială - Compresie

Vectorii de intrare : blocuri de  $b \times b = 4 \times 4$  pixeli (16 componente) sau 8x8.

Imagini  $m \times n$  în nuanțe de gri și  $N = 2^k$  vectori semnificativi.

Blocurile semnificate sunt date de centroizii K-Means.

$$R = \frac{m * n * 8}{\left(\frac{m}{b} * \frac{n}{b}\right) * k + 2^k * b^2 * 8}$$



Imagine codată cu blocuri 8x8 și o tabelă cu 16 intrări.

# N=64 de blocuri reprezentative

2x2 PSNR=32.38 RATA=5.22



4x4 PSNR=27.95 RATA=16.00



8x8 PSNR=25.36 RATA=13.47

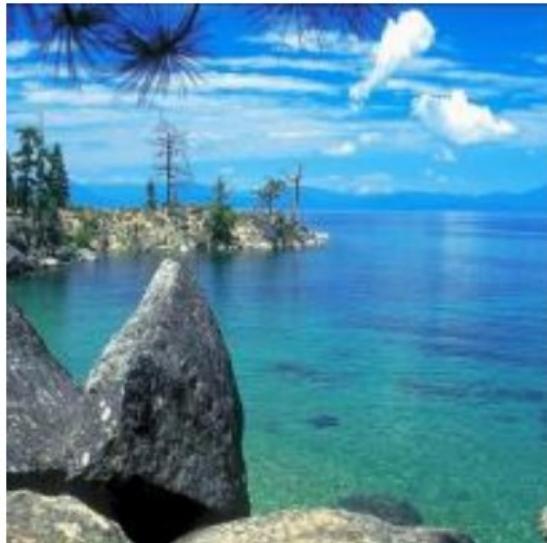


$$R = \frac{m * n * 8}{\left(\frac{m}{b} * \frac{n}{b}\right) * 6 + 2^6 * b^2 * 8}$$

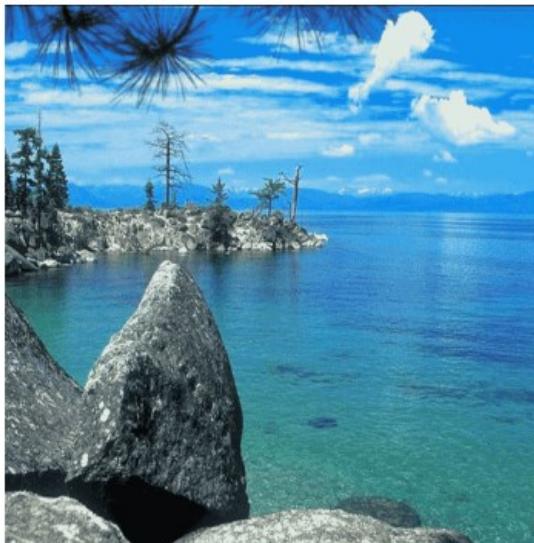
- $m = n = 256$  – dimensiunea imaginii
- blocuri 2x2 : redundanță sporită, clustere mici  $\Rightarrow$  PSNR mare. Nr. mare de sub-blocuri  $\Rightarrow$  rata mică.

# Aplicație : Reducerea culorilor

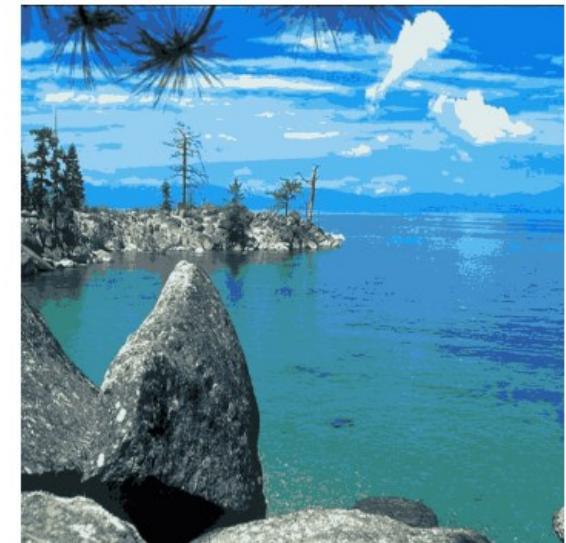
- 1) Pixelii (R,G,B) ai imaginii sunt vectori în  $R^3$ .
- 2) K-Means pentru setul de pixeli și numărul de culori.
- 3) Culorile reprezentative : centroizii clusterelor.
- 4) Construirea noii imagini folosind culorile din paletă.



a) Original picture

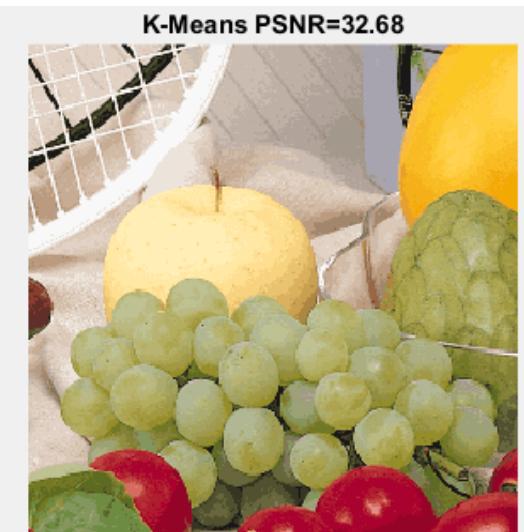
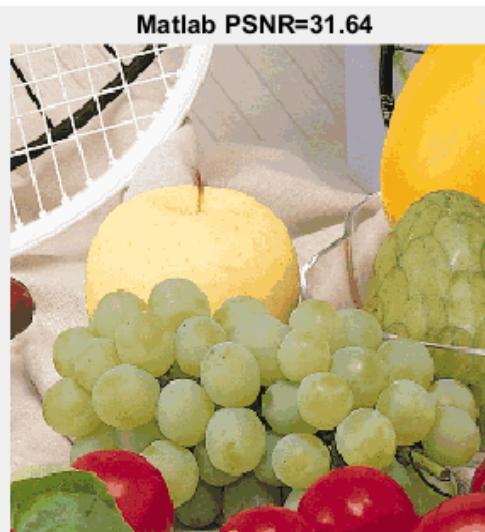
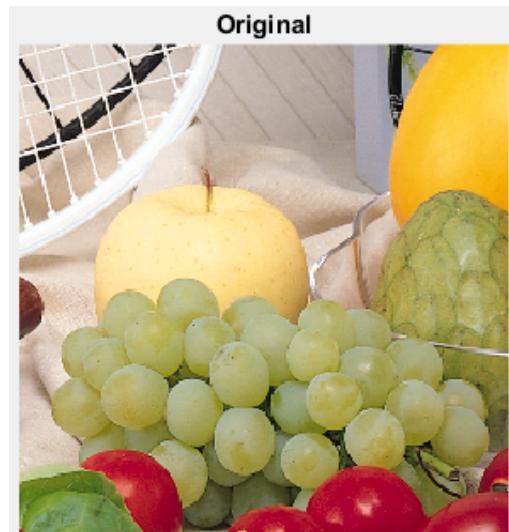
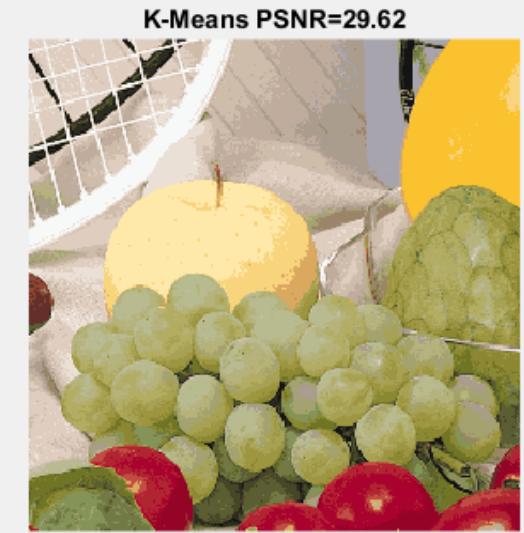
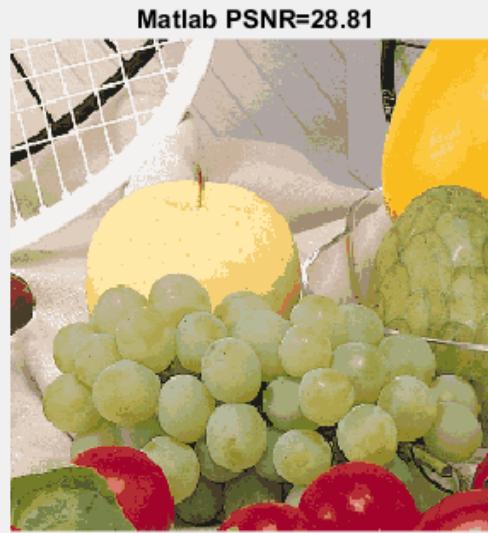


b) K-means with dithering



c) K-means

# K-Means vs. Matlab : 32 și 64 culori



# Metrici pentru evaluarea clusterizării

- **Sunt cunoscute etichetele reale ale claselor rezultate**
  - i. omogenitate
  - ii. index Rand
  - iii. index Rand ajustat
  - iv. Informație mutuală
- **Nu sunt cunoscute etichetele reale**
  - i. silhouette score
  - ii. index Calinski-Harabasz
  - iii. index Davies-Bouldin
  - iv. inertie (pentru K-Means este D – de pe planșele anterioare)

## Omogenitate

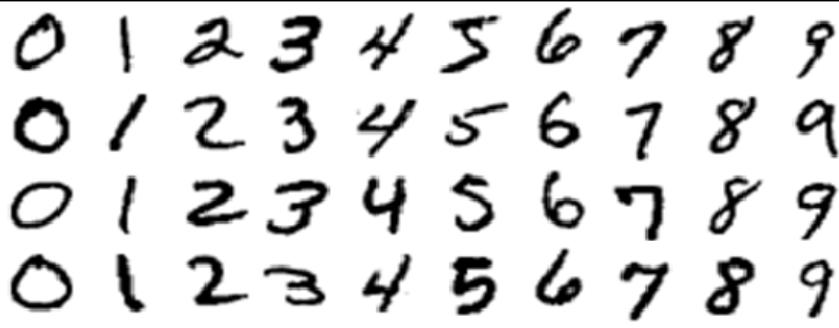
- Clusterele conțin doar puncte de date care sunt membri ai unei singure clase.
- Această măsurătoare este independentă de valorile absolute ale etichetelor: o permutare a valorilor etichetei clasei sau clusterului nu va schimba în niciun fel valoarea scorului.

<https://saltfarmer.github.io/blog/machine%20learning/Evaluation-Metrics-Clustering/>

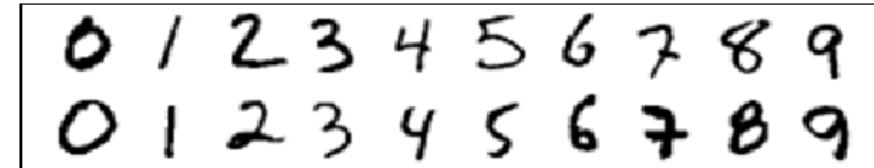
# K-Means – alte aplicații - clasificare cifre

- **Învățare supervizată**
  - input și output cunoscute
  - datele au etichete verificate manual (de către om)
- **Învățare nesupervizată**
  - input cunoscut, output necunoscut
  - datele nu au etichete

[https://colab.research.google.com/github/goodboychan/goodboychan.github.io/blob/main/\\_notebooks/2020-10-26-01-K-Means-Clustering-for-Imagery-Analysis.ipynb#scrollTo=yQGToH9iTQ89](https://colab.research.google.com/github/goodboychan/goodboychan.github.io/blob/main/_notebooks/2020-10-26-01-K-Means-Clustering-for-Imagery-Analysis.ipynb#scrollTo=yQGToH9iTQ89)



(a) – Digit samples from MNIST training set



(b) – Digit samples from MNIST test set

# Metrici pentru clasificare

- Metrici diferite pentru clasificare (ex. acuratețea)

		Clasă detectată	
		DA	NU
Clasa reală	DA	Adevăr pozitiv	Fals negativ
	NU	Fals pozitiv	Adevărat negativ

- Accuracy (acuratețe)

$$Acc = \frac{TP + TN}{TP + FP + FN + TN}$$

- Precision (precizie)

$$P = \frac{TP}{TP + FP}$$

- Recall (sensibilitate)

$$R = \frac{TP}{TP + FN}$$

- F1-score (scor F1)

$$F1 = \frac{2 \cdot R \cdot P}{R + P}$$

# Metrici pentru clasificare

## ➤ Exemplul 1

		Clasă detectată	
Clasa reală		DA	NU
	DA	20 (TP)	4 (FN)
	NU	3 (FP)	22 (TN)

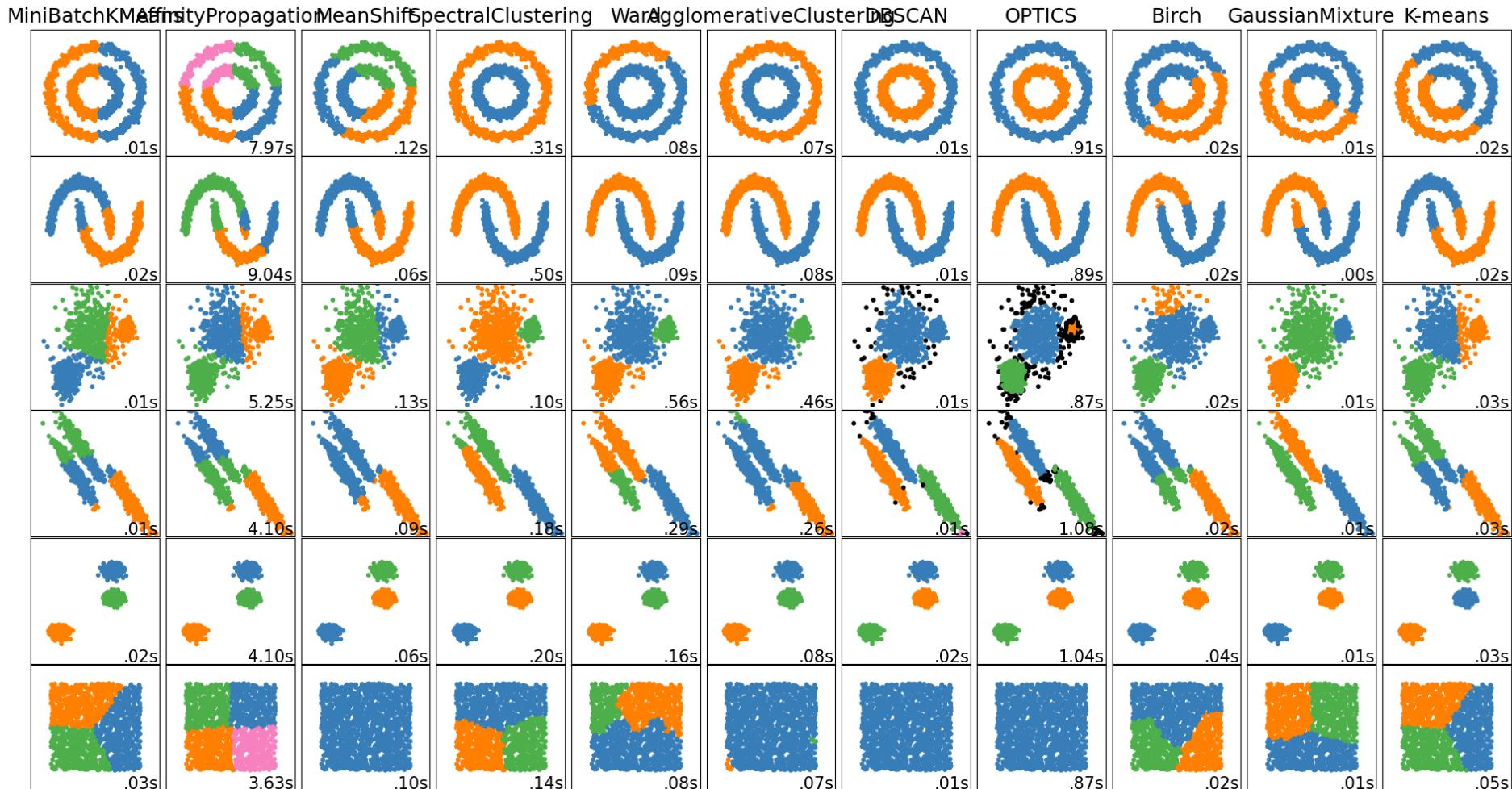
A=0,8571 P=0,8696 R=0,8333 F1=0,8511

## ➤ Exemplul 2

		Clasă detectată	
Clasa reală		DA	NU
	DA	1 (TP)	10 (FN)
	NU	1 (FP)	90 (TN)

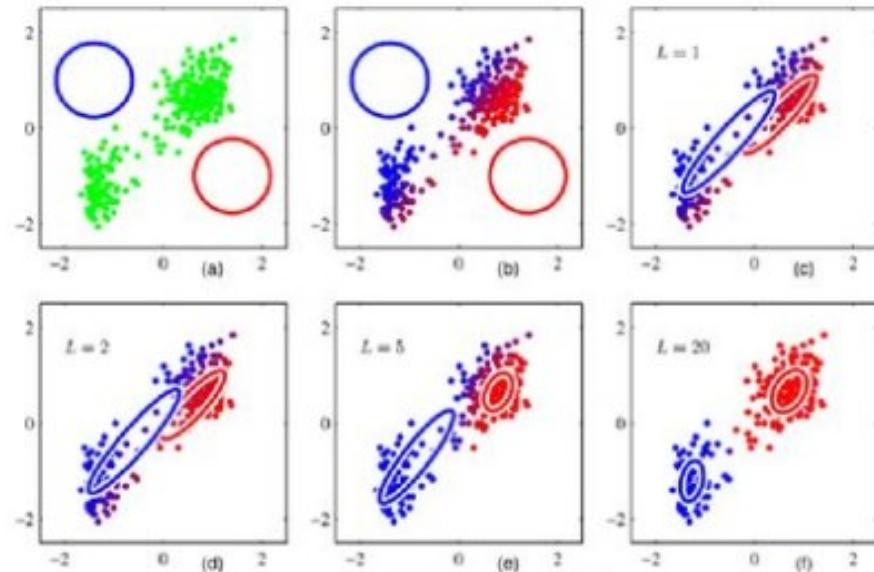
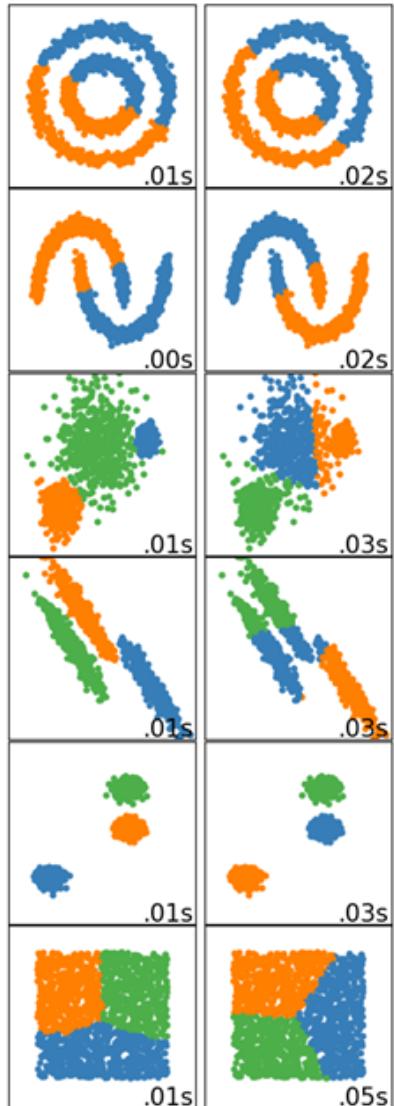
A=0,8922 P=0,500 R=0,0909 F1=0,1538

# Limitări ale K-means



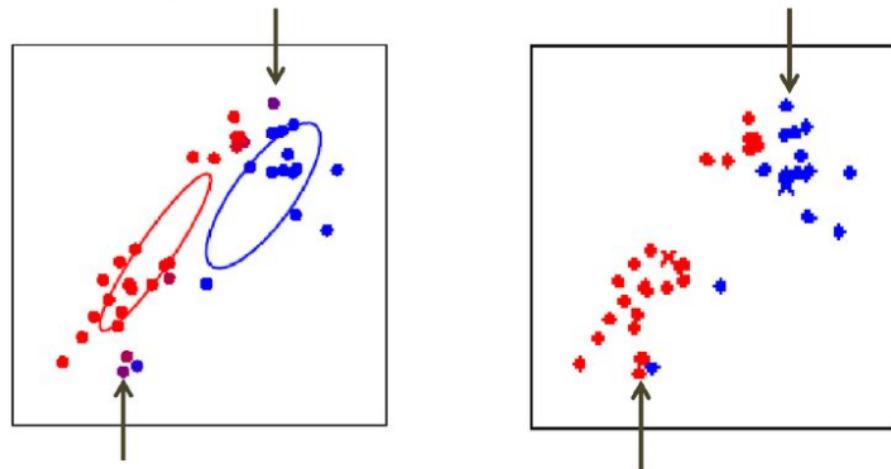
# Limitări ale K-means

GaussianMixture K-means

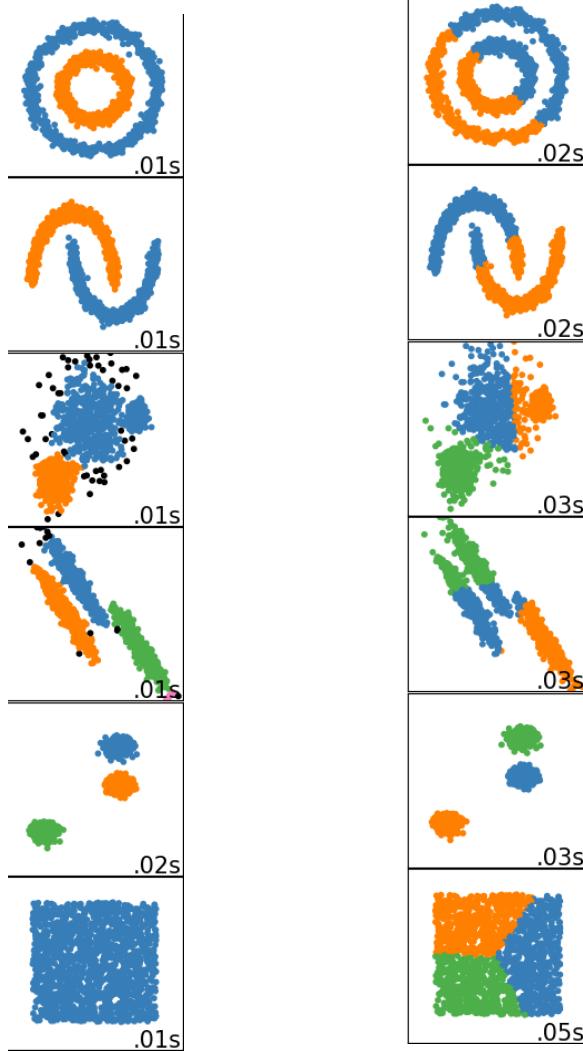


GMM

k-means



# DBSCAN – ideea generală



**Density-based spatial clustering of applications with noise (DBSCAN)**

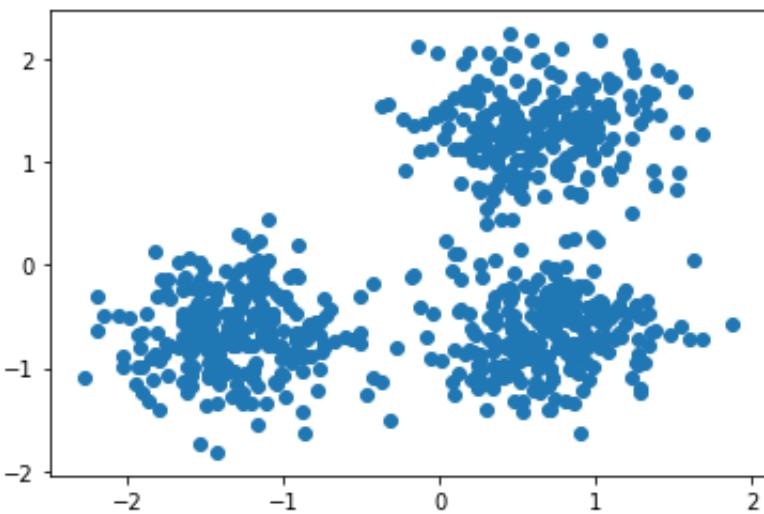
<http://www2.cs.uh.edu/~ceick/7363/Papers/dbscan.pdf>

- Identifică clusterele după densitățile punctelor
- Încearcă să imite modul în care o persoană grupează elemente
- Funcționează bine și în N-D
- Pentru o mulțime de vectori din  $\mathcal{R}^d$  să se găsească grupuri asemănătoare.
- Față de Kmeans și GMM, nu este necesar numărul de centroizi. Dar, intervin alți parametri.
- Descoperă clustere de o formă oarecare ...
- Descoperă puncte izolate care nu aparțin unui cluster

# DBSCAN

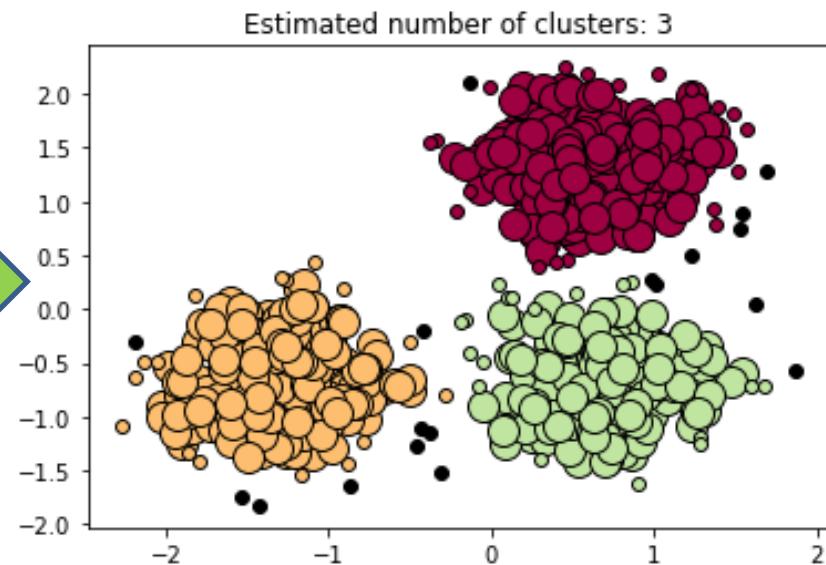
Pentru o mulțime de vectori din  $\mathcal{R}^d$  să se găsească **grupuri asemănătoare pe baza densității punctelor**. Față de K-Means și GMM, nu este necesar numărul de centroizi. Dar, intervin alți parametri.

Înainte de DBSCAN



DBSCAN

După DBSCAN

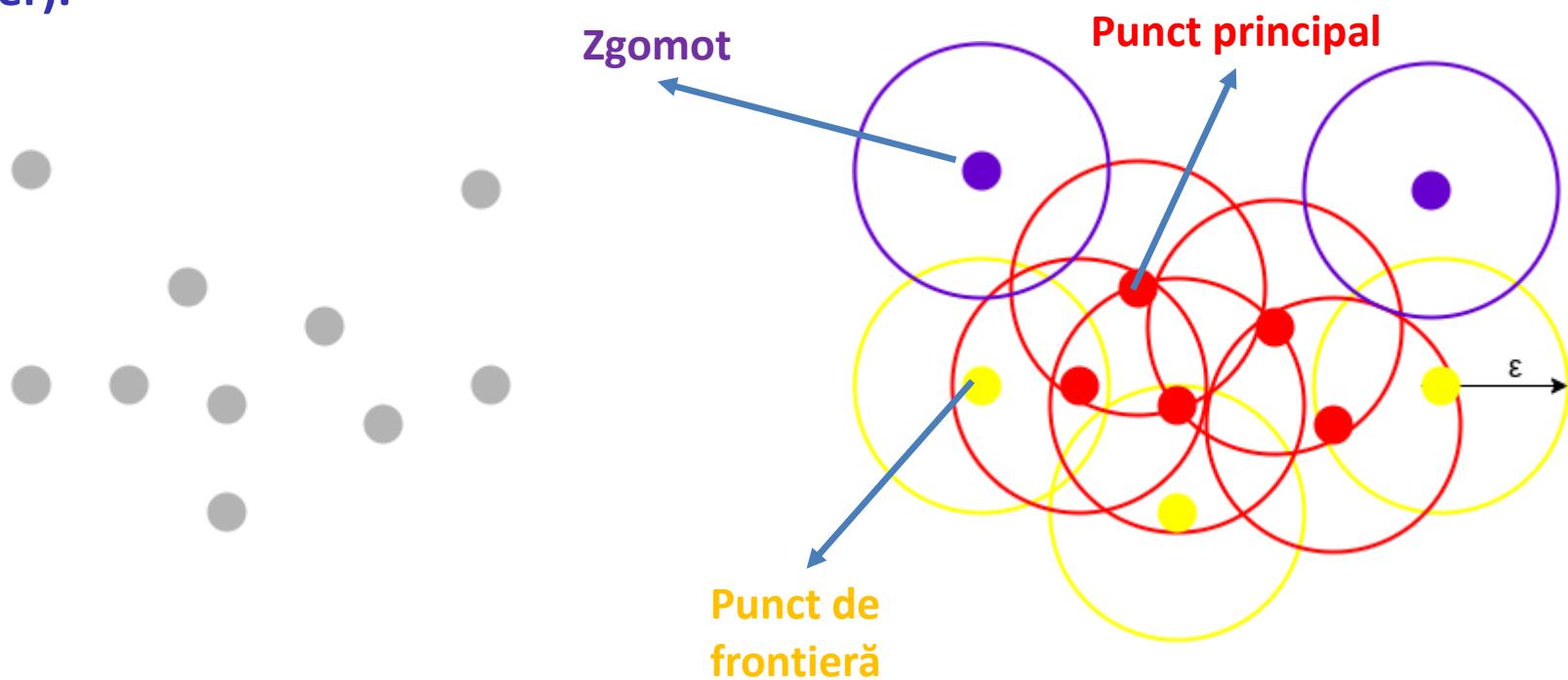


# DBSCAN – definiții

**Core point – puncte principale** care pentru o rază epsilon sunt înconjurate de cel puțin N alte puncte.

**Border point – puncte de frontieră** care pentru o rază epsilon sunt înconjurate mai puțin de N alte puncte și au vecini puncte principale.

**Zgomot** – puncte care pentru rază epsilon nu sunt înconjurate de alte puncte (oulier).

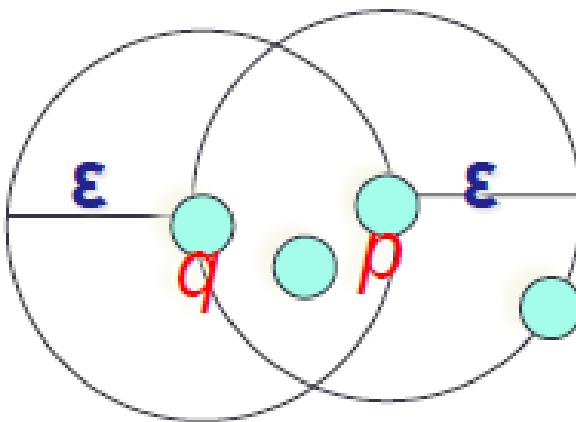


# DBSCAN – parametri

- $\varepsilon$  – distanță maximă între două puncte vecine
- $N_\varepsilon(p)$  - mulțimea punctelor din vecinătatea punctului  $p$

$$N_\varepsilon(p) = \{q \in D \mid d(q, p) \leq \varepsilon\}$$

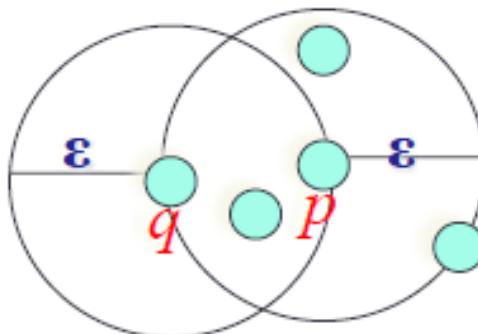
- MinPts : numărul minim de puncte într-o vecinătate



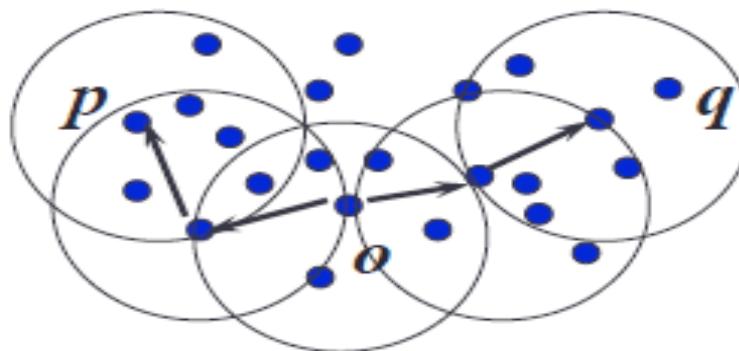
- MinPts=4,  $p$  este un punct de **mare densitate**
- MinPts=4,  $q$  este un punct de **mică densitate**

# DBSCAN – definiții (vecinătate de densitate)

- Puncte **Directly Density-Reachable** :  $q \in N_\varepsilon(p)$



- Puncte **Density-Reachable** :  $\exists p_1 = p \dots p_n = q \quad p_{i+1} \in N_\varepsilon(p_i) \forall i = 1, n - 1$   
(lanț de puncte *Directly Density-Reachable* între punctele  $p$  și  $q$ )



- **Conectivitate bazată pe densitate** : dacă la  $p$  și  $q$  se poate ajunge dintr-un același punct  $o$

# DBSCAN – descrierea unui cluster

- Un cluster este o mulțime de puncte  $C$  care satisface două criterii
- **Maximalitate** :  $\forall p, q$  dacă  $p \in C$  și  $q$  poate fi atins din punctul  $p$  (*Density-Reachable*), atunci  $q \in C$
- **Conecțivitate** :  $\forall p, q \in C$ ,  $p$  și  $q$  sunt conectate

*for each object  $p$  in DataSet*

*if  $p$  is not classified then*

*if  $p$  is a core object*

*$C =$  all points density-reachable from  $p$*

*mark all objects in  $C$  as processed*

*report  $C$  as a cluster*

*else*

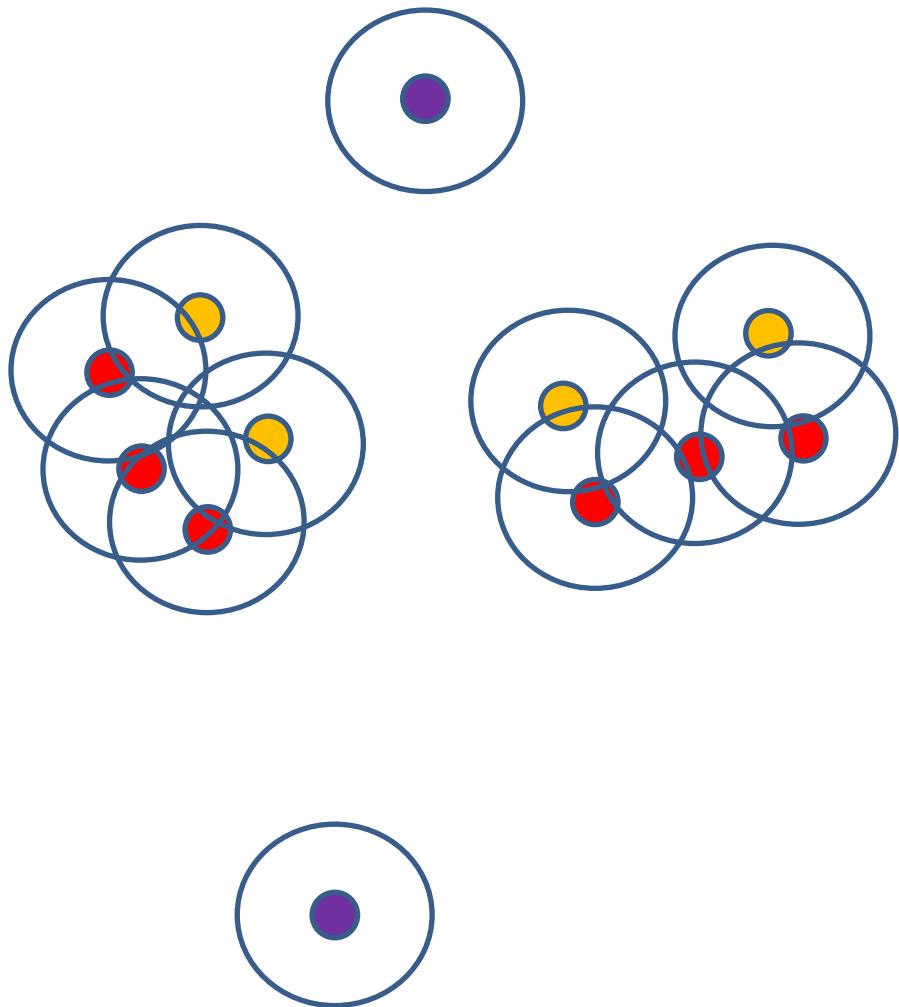
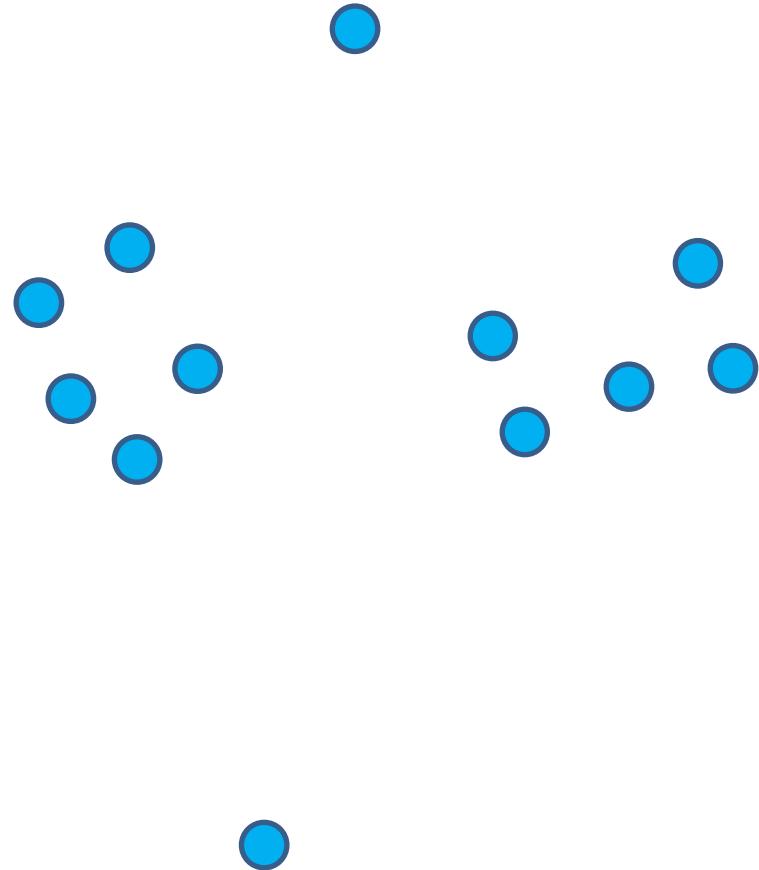
*mark  $p$  as temporarily outlier*

# DBSCAN – pseudocod

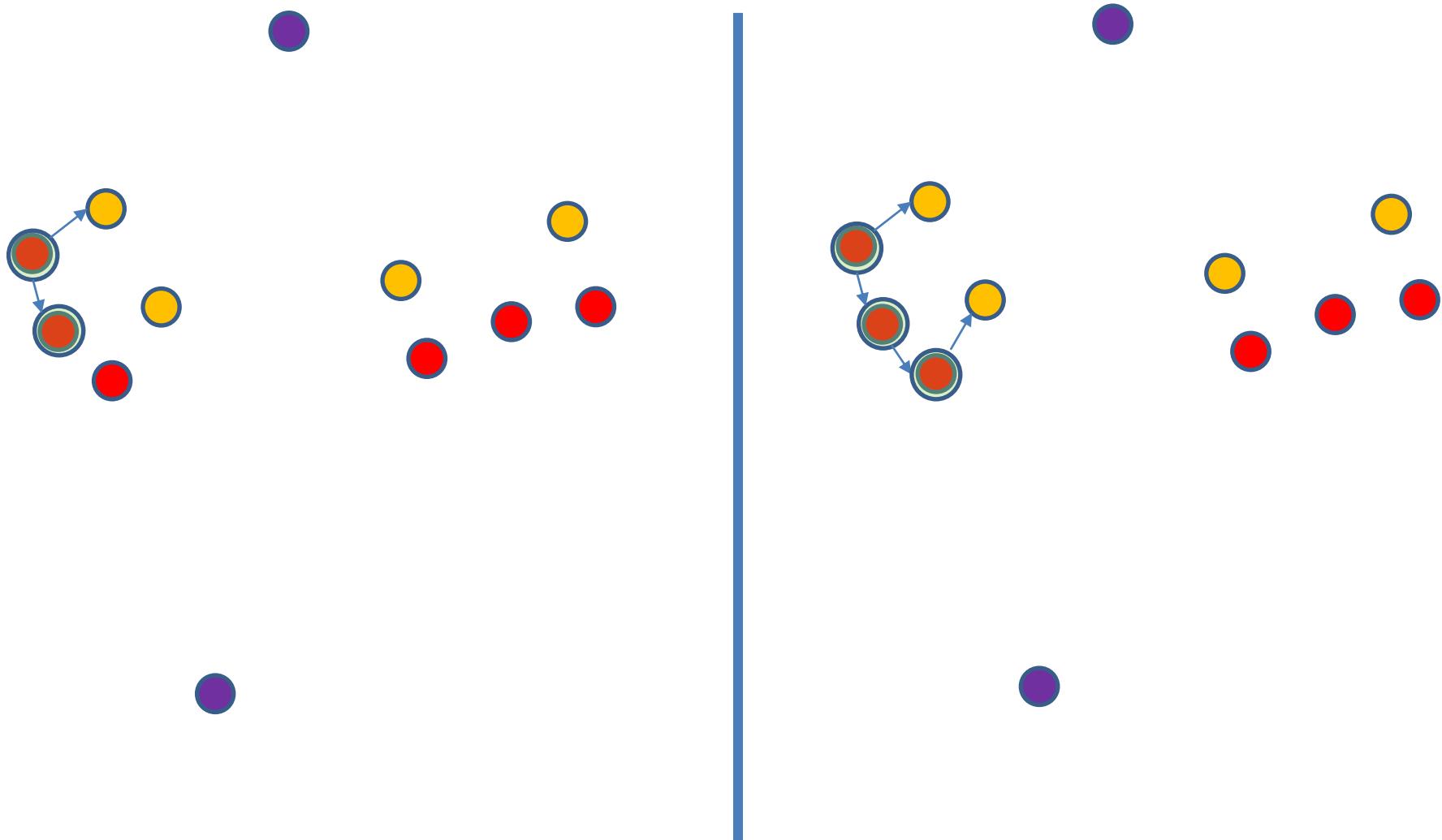
Unesc grupurile de puncte principale, până când nu mai sunt grupuri apropiate. Adaug punctele de frontieră care pot fi grupate cu cele din mijloc. Trec la alt punct de mijloc și formează un nou cluster. Zgomotele rămân izolate.

```
C = 0 // current number of clusters
for each point p in DataSet
    if p is classified continue; // point already processed
    if p is not a core point label(P) = Noise; continue;
    C = C + 1 ; label(p) = C // beginning of a new cluster
    S = set of neighbors not yet classified (queue or stack)
    for each point q from S {
        if label(q) = Noise then label(q) = C ; continue
        label(q) = C // q belongs to C cluster
        if q is a core point
            add his unclassified neighbors to S
    end for
end for
```

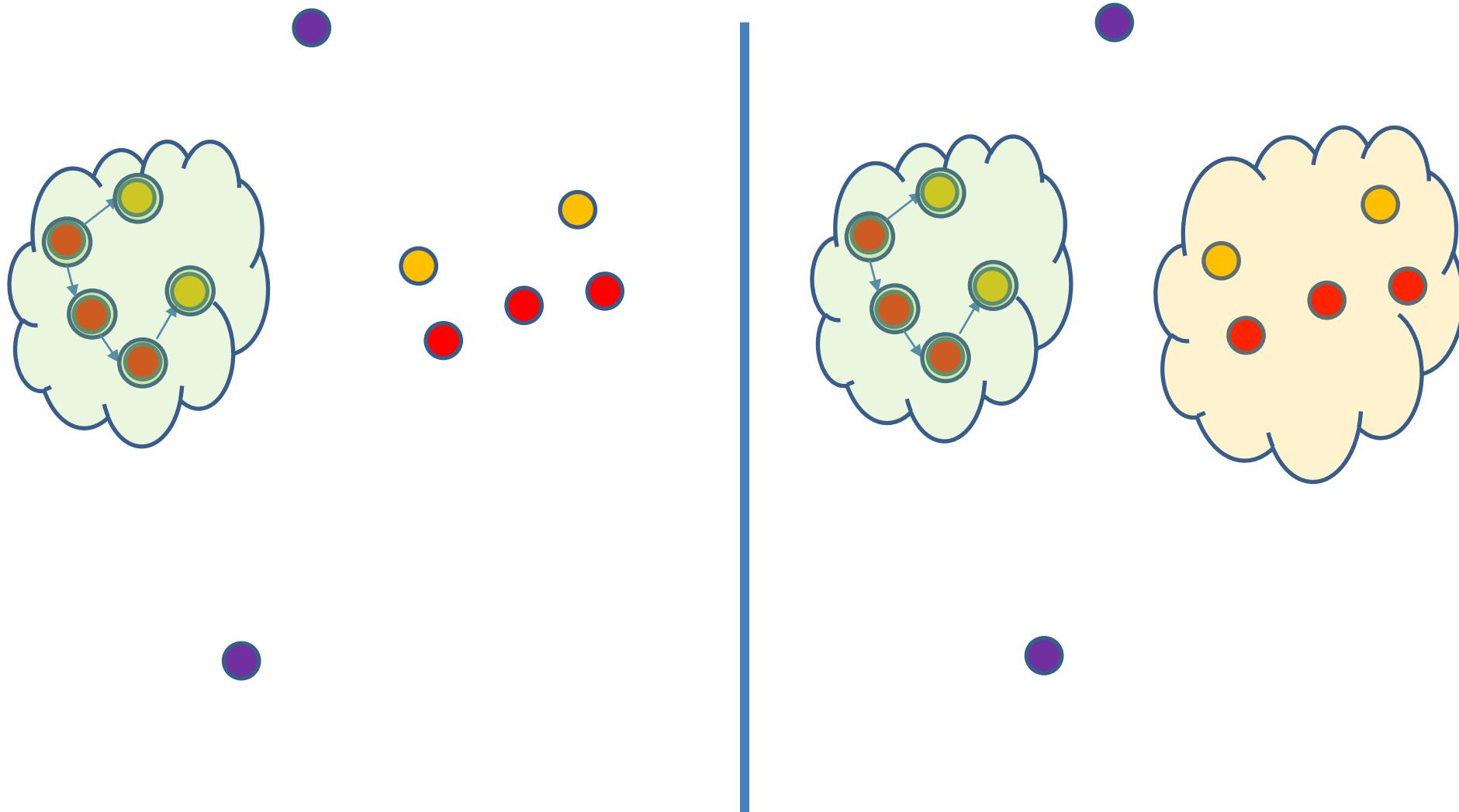
# DBSCAN – pas cu pas



# DBSCAN – pas cu pas



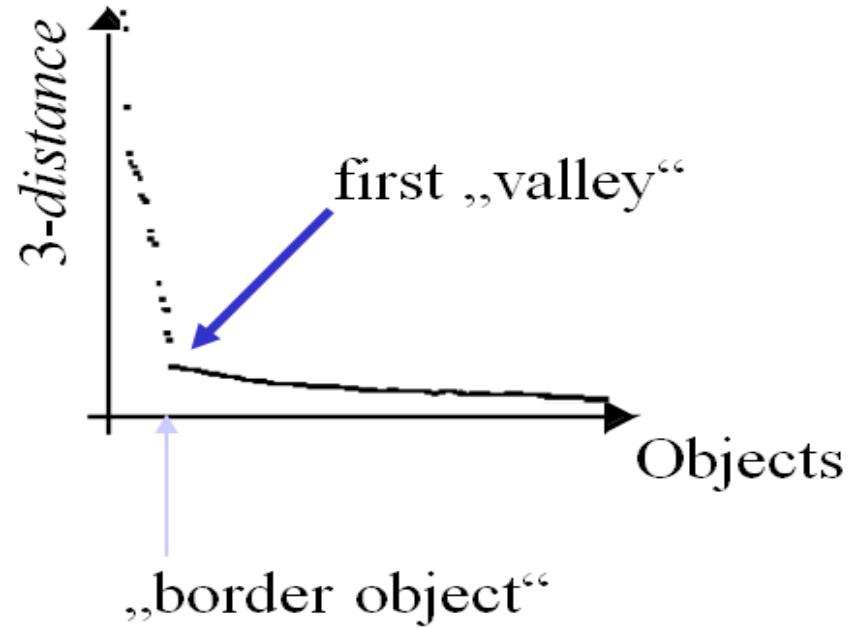
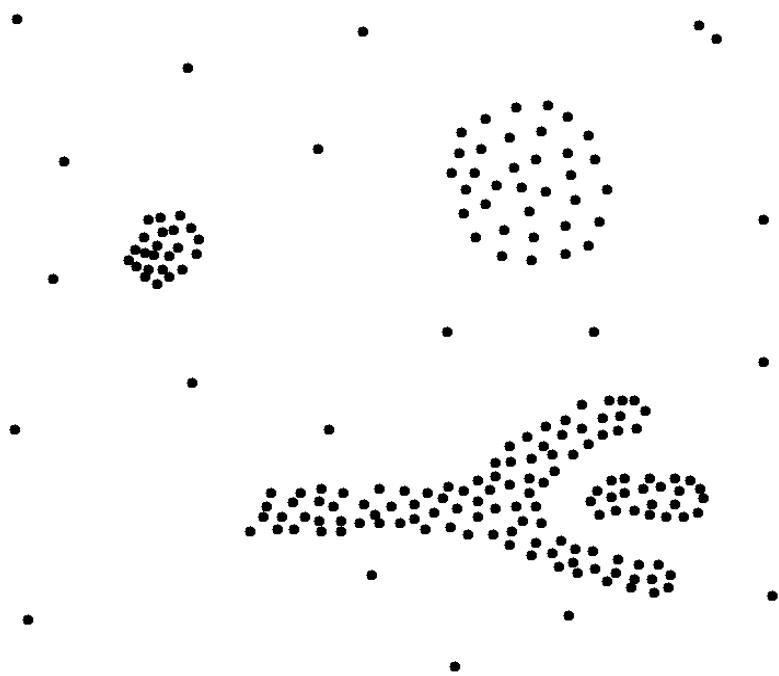
# DBSCAN – pas cu pas



# DBSCAN – alegerea parametrilor

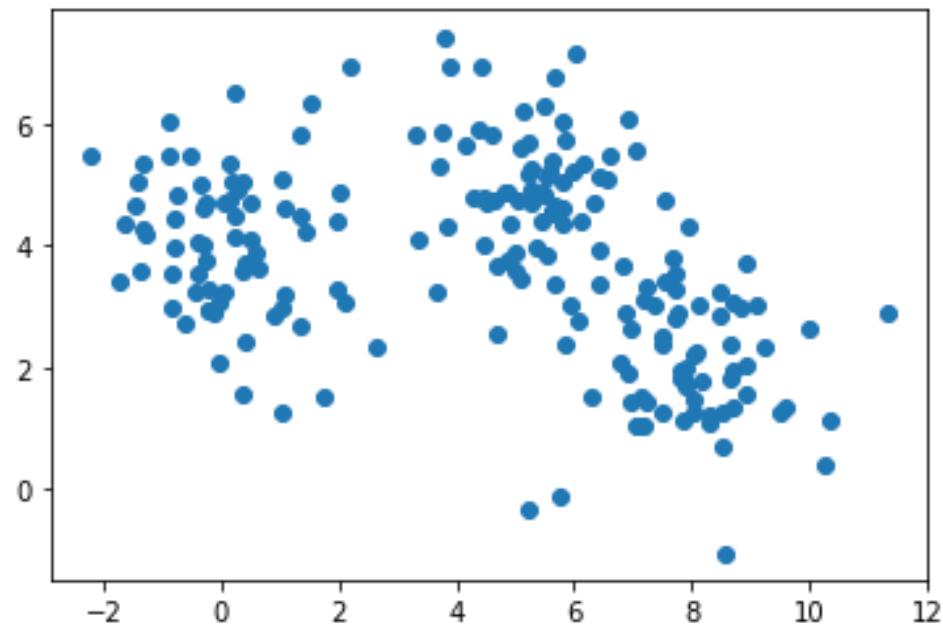
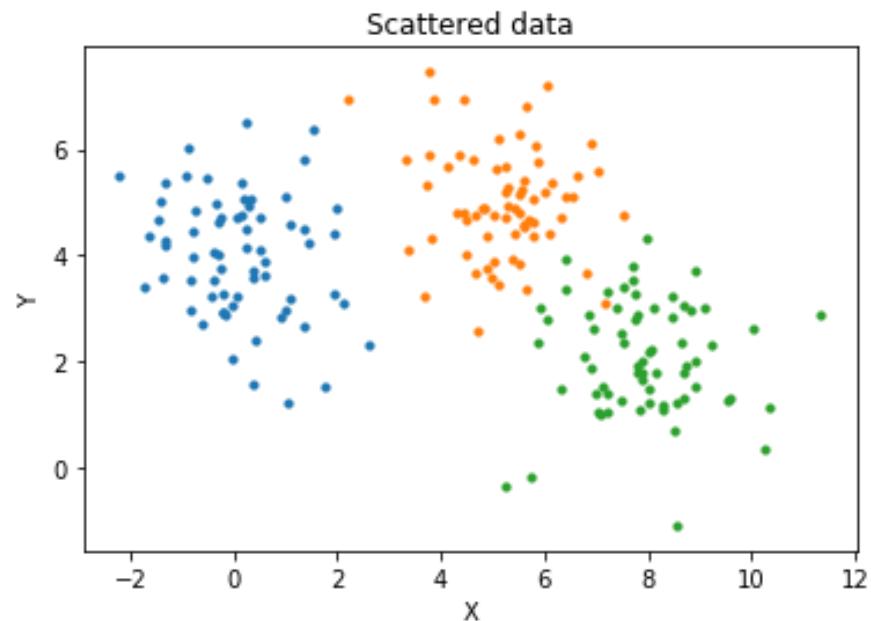
- $\text{MinPts} = 2 * d$  ( $d$  este dimensiunea spațiului) sau  $\text{MinPts} = d + 1$
- $\text{MinPts} = 1$  – nu are sens
- $\text{MinPts} = 2$  – echivalent cu hierarchical clustering
- $\text{MinPts} \geq 3$
  
- Alegerea funcției pentru calculul distanței depinde de datele utilizate.
- Trebuie să fie potrivită datelor. Spre exemplu, pentru date geografice se utilizează distanța între două puncte pe o sferă.
  
- Idee : alegerea ca  $\varepsilon$  a densității celui mai mic cluster
- Cum ? Distanța la cei mai apropiati  $k$ -vecini, unde  $k = \text{MinPts} - 1$   
$$k - dist(x) = f(d(x, x_1), d(x, x_2) \dots d(x, x_k))$$
- Funcția  $f$  poate fi **maximul**, suma, etc.

# DBSCAN – alegerea parametrilor



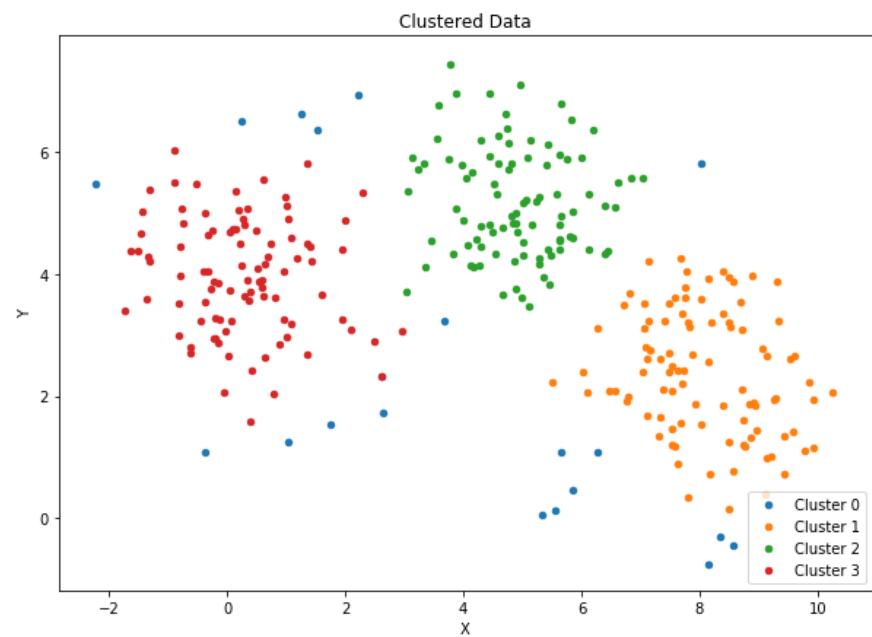
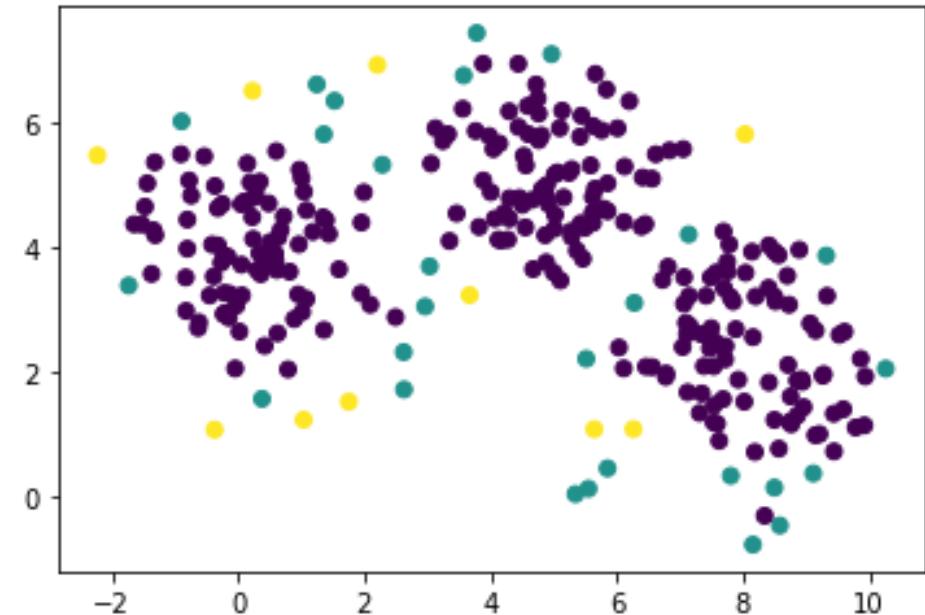
- $k$  – plot : grafic cu toate k-distanțele în ordine desc.
- Alege  $o$  din MinPts-plot primul “cot” (maxim pt.  $|F''(x)|$ ) iar  $\varepsilon = \text{MinPts} - \text{distance}(o)$

# Implementare Python



<https://github.com/Moosa-Ali/DBscan-Clustering-Implementation/blob/main/DBSCAN%20implementation.ipynb>

# Implementare Python



<https://github.com/Moosa-Ali/DBscan-Clustering-Implementation/blob/main/DBSCAN%20implementation.ipynb>

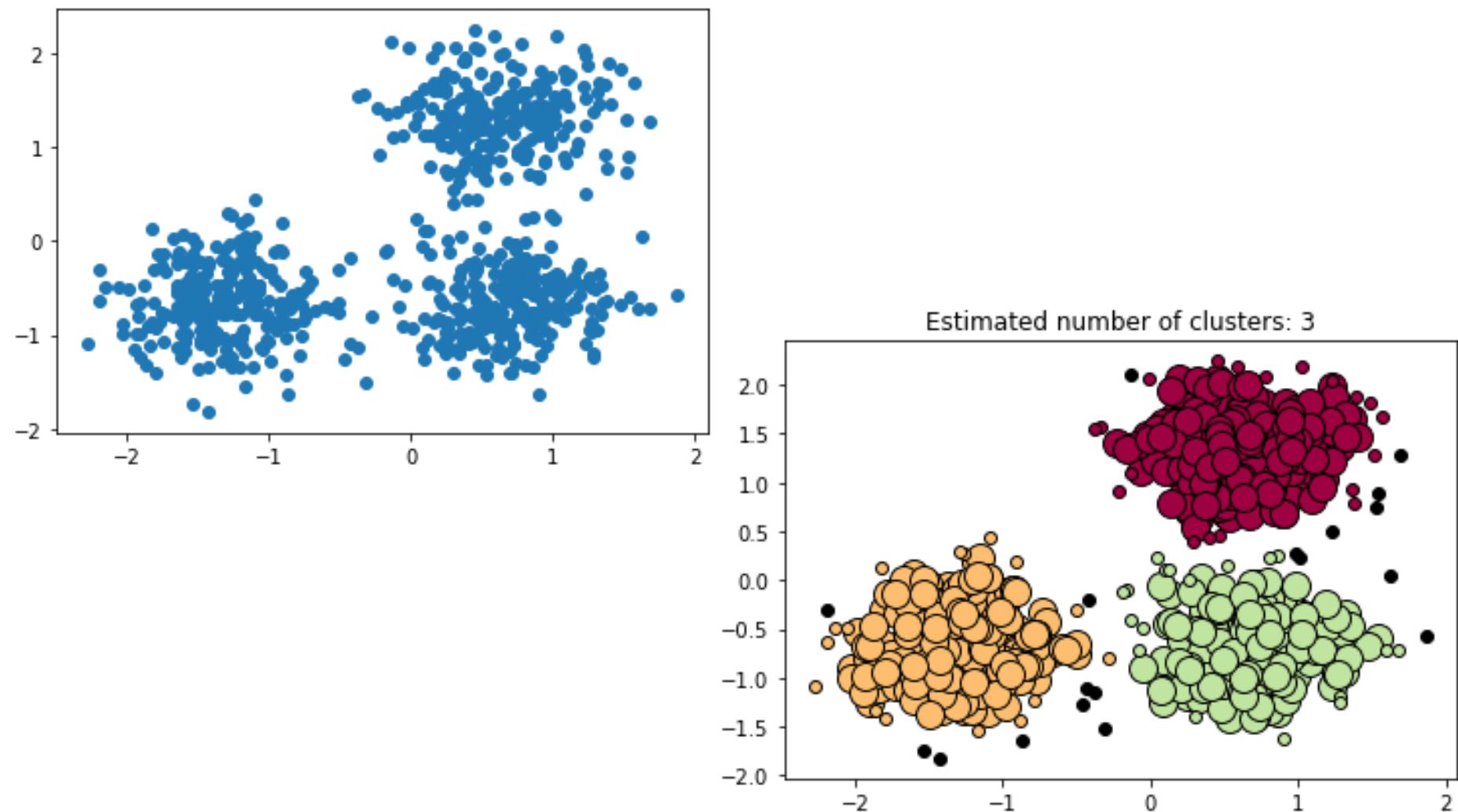
# Biblioteca sklearn - DBSCAN

```
class sklearn.cluster.DBSCAN(  
    eps=0.5,  
    *,  
    min_samples=5,  
    metric='euclidean',  
    metric_params=None,  
    algorithm='auto',  
    leaf_size=30,  
    p=None,  
    n_jobs=None)
```

```
8     import numpy as np  
9  
10    from sklearn.cluster import DBSCAN  
11    from sklearn import metrics  
12    from sklearn.datasets import make_blobs  
13    from sklearn.preprocessing import StandardScaler  
14  
15  
16    # #####  
17    # Generate sample data  
18    centers = [[1, 1], [-1, -1], [1, -1]]  
19    X, labels_true = make_blobs(  
20        n_samples=750, centers=centers, cluster_std=0.4, random_state=0  
21    )  
22  
23    X = StandardScaler().fit_transform(X)  
24  
25    # #####  
26    # Compute DBSCAN  
27    db = DBSCAN(eps=0.3, min_samples=10).fit(X)  
28    core_samples_mask = np.zeros_like(db.labels_, dtype=bool)  
29    core_samples_mask[db.core_sample_indices_] = True  
30    labels = db.labels_  
31  
32    # Number of clusters in labels, ignoring noise if present.  
33    n_clusters_ = len(set(labels)) - (1 if -1 in labels else 0)  
34    n_noise_ = list(labels).count(-1)
```

<https://scikit-learn.org/stable/modules/generated/sklearn.cluster.DBSCAN.html#>

# Biblioteca sklearn - DBSCAN



# Avantaje și dezavantaje

Avantaje	Dezavantaje
Nu trebuie să fie cunoscut numărul de grupuri.	Nu este complet deterministic.
Pot fi identificate grupuri cu forme arbitrară.	Rezultatul depinde de distanța utilizată.
Robust la outlieri.	Nu clusterizează bine date cu densități diferite în spațiul punctelor.
Sunt necesari doar doi parametri – epsilon și min points.	Dacă nu sunt bine înțelese datele, alegerea lui epsilon și min points este dificilă.

# Exemplu

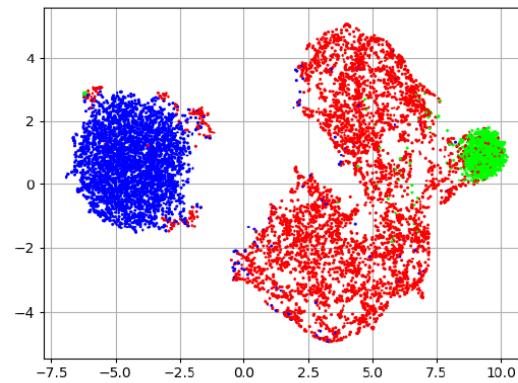
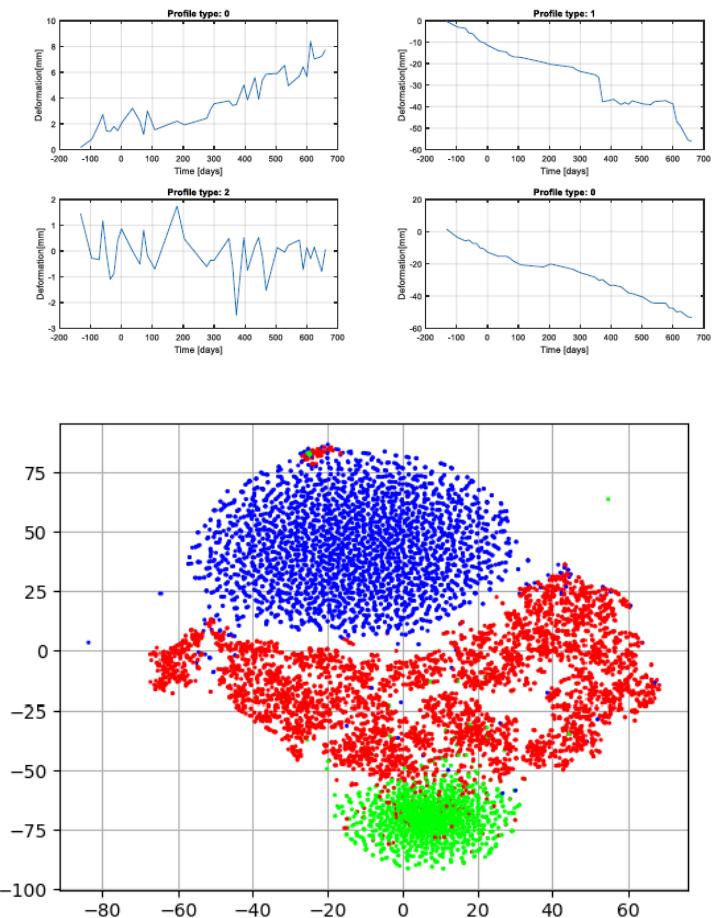


Figure 3 UMAP dimensionality reduction and actual classes (green – piece-wise linear, red – piece-wise linear and step; blue – no significant deformation)

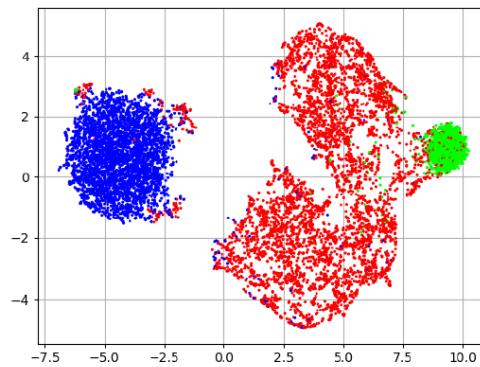


Figure 3 UMAP dimensionality reduction and actual classes (green – piece-wise linear, red – piece-wise linear and step; blue – no significant deformation)

S. -A. Toma, B. Sebacher, D. Teleagă and A. Focşa, "Deformation Profile Analysis Using Uniform Manifold Approximation and Projection," *IGARSS 2020 - 2020 IEEE International Geoscience and Remote Sensing Symposium*, 2020, pp. 4227-4230, doi: 10.1109/IGARSS39084.2020.9323279.

# Dithering

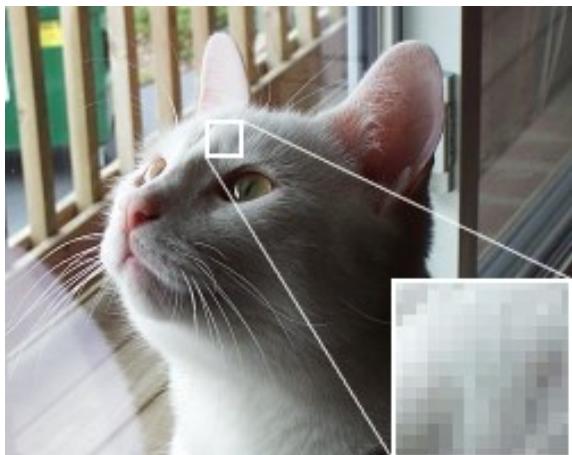
- **Dithering** este un termen folosit pentru a descrie aplicația strategică a zgomotului unei imagini.
- A fost folosit în mod tradițional pentru a îmbunătăți aspectul imaginilor în cazul în care ieșirea este limitată la o anumită gamă de culori. Spre exemplu, dacă vrem să utilizăm 2 “culori” (alb și negru) pentru reprezentarea nuanțelor de gri.



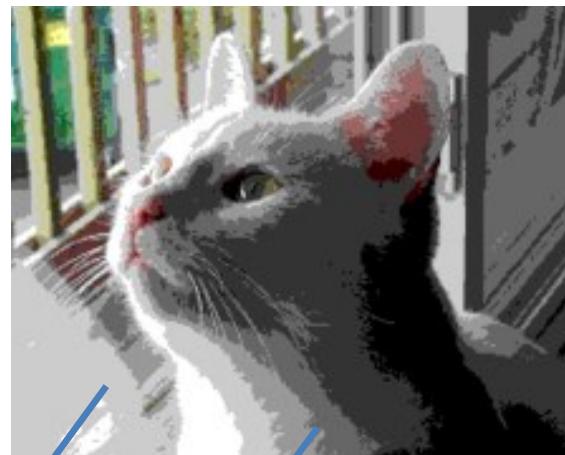
# Dithering

- Când se utilizează cuantizarea pentru reducerea numărului de culori, pot să apară două tipuri de artefacte: benzi de culoare și zone plate.
- Tehnica de *dithering* poate ajuta la corecția acestor artefacte prin împrăștierea erorii de cuantificare.

Original



După cuantificare



După cuantificare  
și dithering

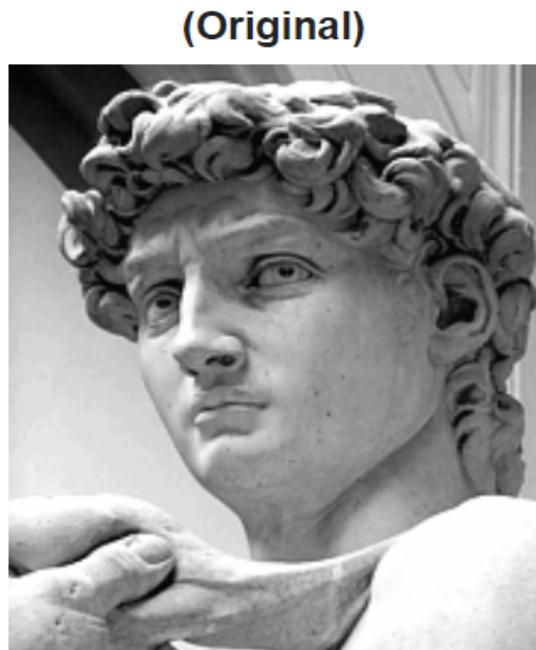


Zonă plată

Benzi de culoare

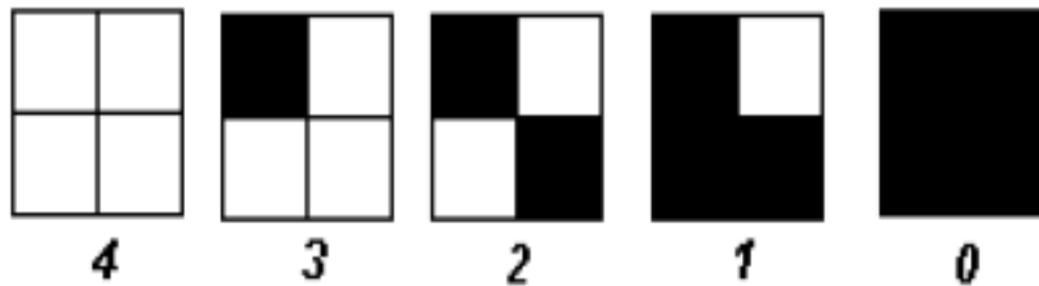
# Dithering - metode

- Average dithering (thresholding) : fiecare pixel este comparat cu o valoare de prag fixă
- Random dithering : fiecare pixel este comparat cu o valoare aleatoare ⇒ imagine foarte zgomotoasă.
- Patterning dithers : folosesc o matrice binară (pattern) pentru reproducerea unei nuanțe de gri.



# Dithering cu pattern

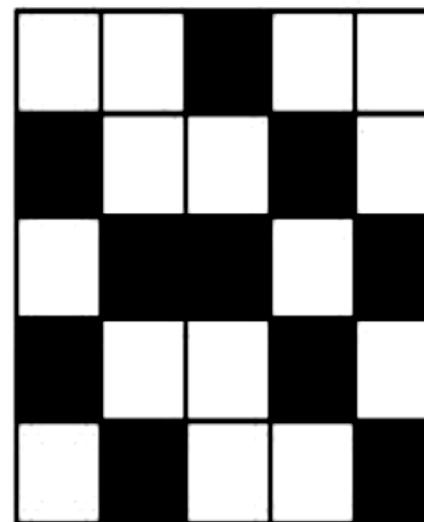
- Dithering utilizat la imprimarea imaginilor în nuanțe de gri.
  - Spre exemplu, pentru imprimarea unei imagini în nuanțe de gri (8 biți) cu rezoluție  $240*180$ , dacă imprimarea se face pe o foaie de  $12.8 * 9.6$  inch, iar rezoluția imprimantei este de  $300 * 300$  DPI . . .
  - Care va fi dimensiunea unui pixel în nr. de puncte ?
  - Puncte :  $(300*12.8)*(300*9.6) = \textcolor{red}{3480*2880 \text{ dots}}$
  - Un pixel :  $(3840/240)*(2880/180) = \textcolor{blue}{16*16=256}$
- 
- **DITHERING CU PATTERN**
    - Conversia rezoluției intensității în rezoluție spațială.
    - Un pixel - pattern monocromatic ( $2 \times 2$ ,  $4 \times 4$ , etc.)
    - O matrice  $N \times N$  poate reprezenta  $N \times N + 1$  intensități.



# Dithering ordonat

- Tip de metoda de dithering cu pattern
- Stocarea unei matrice monocromatice de referință.
- Compararea unui pixel cu valorile matricei de referință : tipărire dacă nivelul de gri  $\leq$  valoare matrice.

0	14	22	5	8
18	9	1	19	13
6	24	16	7	23
21	2	12	20	3
10	15	4	11	17



Matricea standard și reprezentarea nivelului de gri 15

# Dithering ordonat

Reguli pentru generarea matricei de referință:

- 1) Completarea cu valori întregi consecutive.
- 2) Reordonarea acestora astfel încât distanța medie între două valori consecutive să fie maximă.
- 3) La margini să se găsească valorile intermediare.

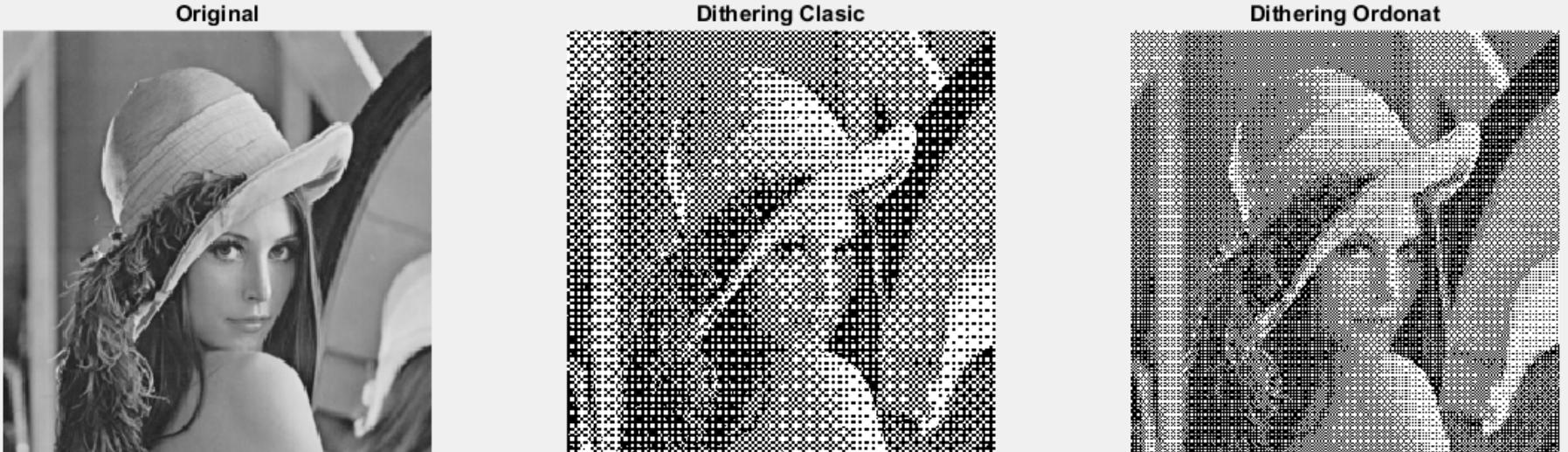
$$\begin{pmatrix} 0 & 1 & 2 & 3 \\ 4 & 5 & 6 & 7 \\ 8 & 9 & 10 & 11 \\ 12 & 13 & 14 & 15 \end{pmatrix} \rightarrow \begin{pmatrix} 0 & 8 & 2 & 10 \\ 12 & 4 & 14 & 6 \\ 3 & 11 & 1 & 9 \\ 15 & 7 & 13 & 5 \end{pmatrix}$$

Matricele cu dimensiuni puteri ale lui 2 pot fi generate recursiv cu relația de mai jos.

$$M_{2n} = \begin{pmatrix} 4 * M_n & 4 * M_n + 2 \\ 4 * M_n + 3 & 4 * M_n + 1 \end{pmatrix}$$

# Dithering ordonat

```
// I (x, y) input, O(x, y) output, D matricea de dithering
for x = 1 to rows // linii
    for y = 1 to columns // rows
        i = (x-1)%n +1; j = (y-1)%n +1;
        if I (x, y) > D(i, j )
            O(x, y) = 1; // pixel alb
        else
            O(x, y) = 0; // pixel negru
    end
end
```



# Dithering prin difuzia erorii

- Exemple: Floyd-Steinberg, Stucki, Burkes, etc.
- Idee : transmiterea erorii (în proporții diverse) către pixelii vecini.

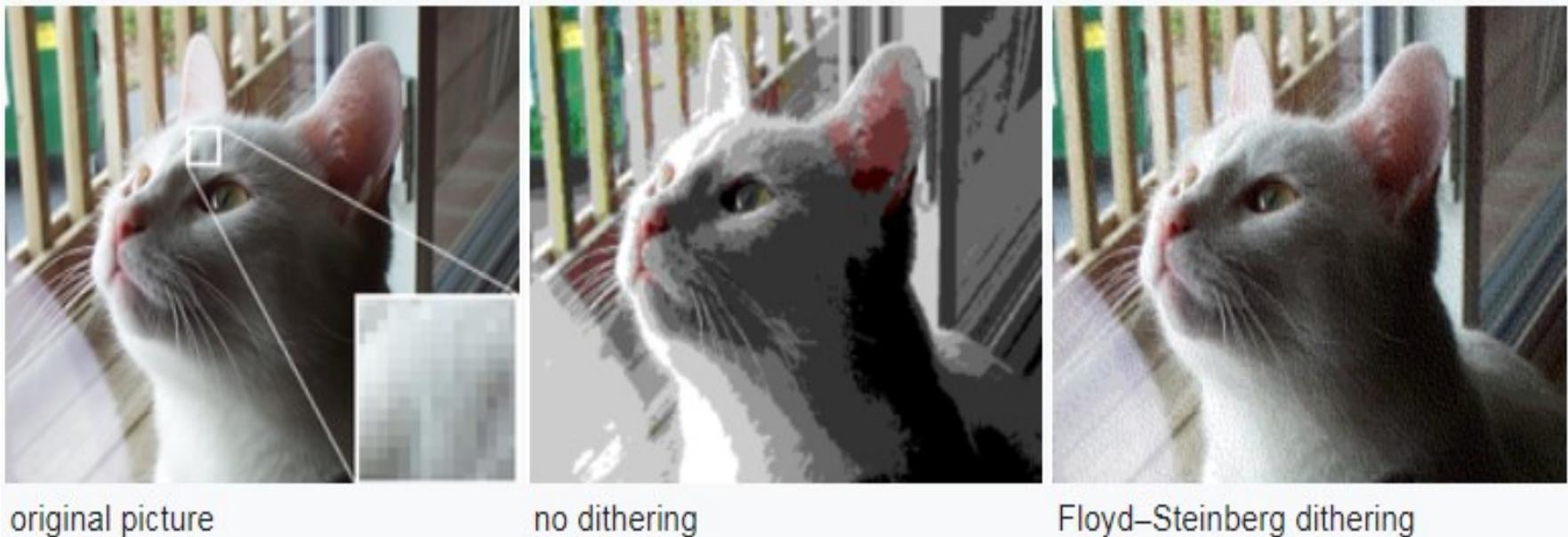
$$\varepsilon = \text{original} - \text{quantized}$$

$$\begin{pmatrix} & & * & \frac{7}{16} \\ & \frac{3}{16} & \frac{5}{16} & \frac{1}{16} \\ \cdots & & & \end{pmatrix}$$

$$I(x+1, y) += \varepsilon * \frac{7}{16}; I(x-1, y+1) += \varepsilon * \frac{3}{16}$$
$$I(x, y+1) += \varepsilon * \frac{5}{16}; I(x+1, y+1) += \varepsilon * \frac{1}{16}$$

# Eroarea acumulată

- Dacă eroarea este pozitivă, cresc pixelii vecini
- Dacă eroarea este negativă, descresc pixelii vecini
- ⇒ probabilitatea ca doi pixeli vecini să fie cuantificați la valori diferite crește ⇒ dispar benzile, difuzia erorii
- ⇒ eroarea globală de cuantificare tinde către zero ...

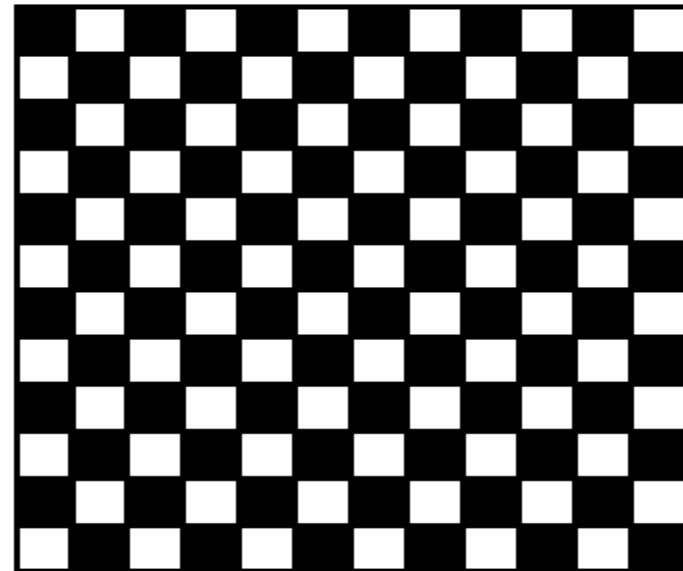
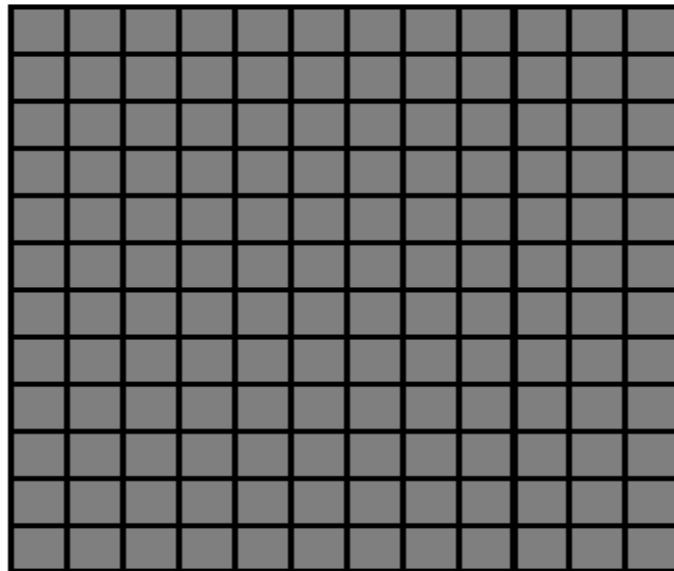


# Algoritmul Floyd-Steinberg - pseudocod

```
for each y from top to bottom do
    for each x from left to right do
        oldpixel = pixel[x][y]
        pixel[x][y] = find_closest_palette_color(oldpixel)
        error = oldpixel - pixel[x][y]
        pixel[x+1][y] += error × 7 / 16
        pixel[x-1][y+1] += error × 3 / 16
        pixel[x][y+1] += error × 5 / 16
        pixel[x+1][y+1] = pixel[x+1][y+1] + error × 1 / 16
    endfor
endfor
```

# Algoritmul Floyd-Steinberg - exemplu

- Un bloc gri constant (0.5) : imaginea in [0-1]
- O tehnica de binarizare ⇒ bloc alb sau negru
- Perceptual ⇒ sa vedem un bloc 50% alb, 50% negru



# Algoritmul Floyd-Steinberg - exemplu

- Binarizare cu prag 0.5;  $[0-0.5] \Rightarrow 0$ ,  $(0.5-1] \Rightarrow 1$

$I_{acc}$			
0.5	0.5	0.5	0.5
0.5	0.5	0.5	0.5
0.5	0.5	0.5	0.5

$I_{out}$			
?	?	?	?
?	?	?	?
?	?	?	?

$I_{acc}$			
X	0.719	0.5	0.5
0.656	0.531	0.5	0.5
0.5	0.5	0.5	0.5

$I_{out}$			
?	?	?	?
?	?	?	?
?	?	?	?

$I_{acc}$			
X	X	0.377	0.5
0.604	0.443	0.482	0.5
0.5	0.5	0.5	0.5

$I_{out}$			
?	?	?	?
?	?	?	?
?	?	?	?

$I_{acc}$			
X	X	X	0.665
0.604	0.514	0.6	0.524
0.5	0.5	0.5	0.5

$I_{out}$			
?	?	?	?
?	?	?	?
?	?	?	?

# Algoritmul Floyd-Steinberg - exemplu

- Binarizare cu prag 0.5;  $[0-0.5] \Rightarrow 0$ ,  $(0.5-1] \Rightarrow 1$

$I_{acc}$			
X	X	X	X
0.604	0.514	0.537	0.419
0.5	0.5	0.5	0.5

$I_{out}$			
?	?	?	?
?	?	?	?

$I_{acc}$			
X	X	X	X
0.604	0.514	0.721	X
0.5	0.5	0.526	0.631

$I_{out}$			
?	?	?	?
?	?	?	?

$I_{acc}$			
X	X	X	X
0.604	0.392	X	X
0.5	0.483	0.439	0.579

$I_{out}$			
?	?	?	?
?	?	?	?

$I_{acc}$			
X	X	X	X
0.775	X	X	X
0.524	0.605	0.512	0.579

$I_{out}$			
?	?	?	?
?	?	?	?

# Algoritmul Floyd-Steinberg - exemplu

- Binarizare cu prag 0.5;  $[0-0.5] \Rightarrow 0$ ,  $(0.5-1] \Rightarrow 1$

$I_{acc}$			
X	X	X	X
X	X	X	X
0.454	0.563	0.512	0.579

$I_{out}$			
?	?	?	?

$I_{acc}$			
X	X	X	X
X	X	X	X
X	0.761	0.512	0.579

$I_{out}$			
?	?	?	?

$I_{acc}$			
X	X	X	X
X	X	X	X
X	X	0.408	0.579

?	?	?	?

$I_{acc}$			
X	X	X	X
X	X	X	X
X	X	X	0.757

?	?	?	?

# Întrebări

