

ANALISIS CLUSTER

Made Satria Wibawa, M.Eng.
2020

Outline

- *Clustering*
- *Disimilaritas*
- *K-Means*
- *K-Medoid*
- *Evaluasi Cluster*
- *Implementasi*

CLUSTERING

Clustering

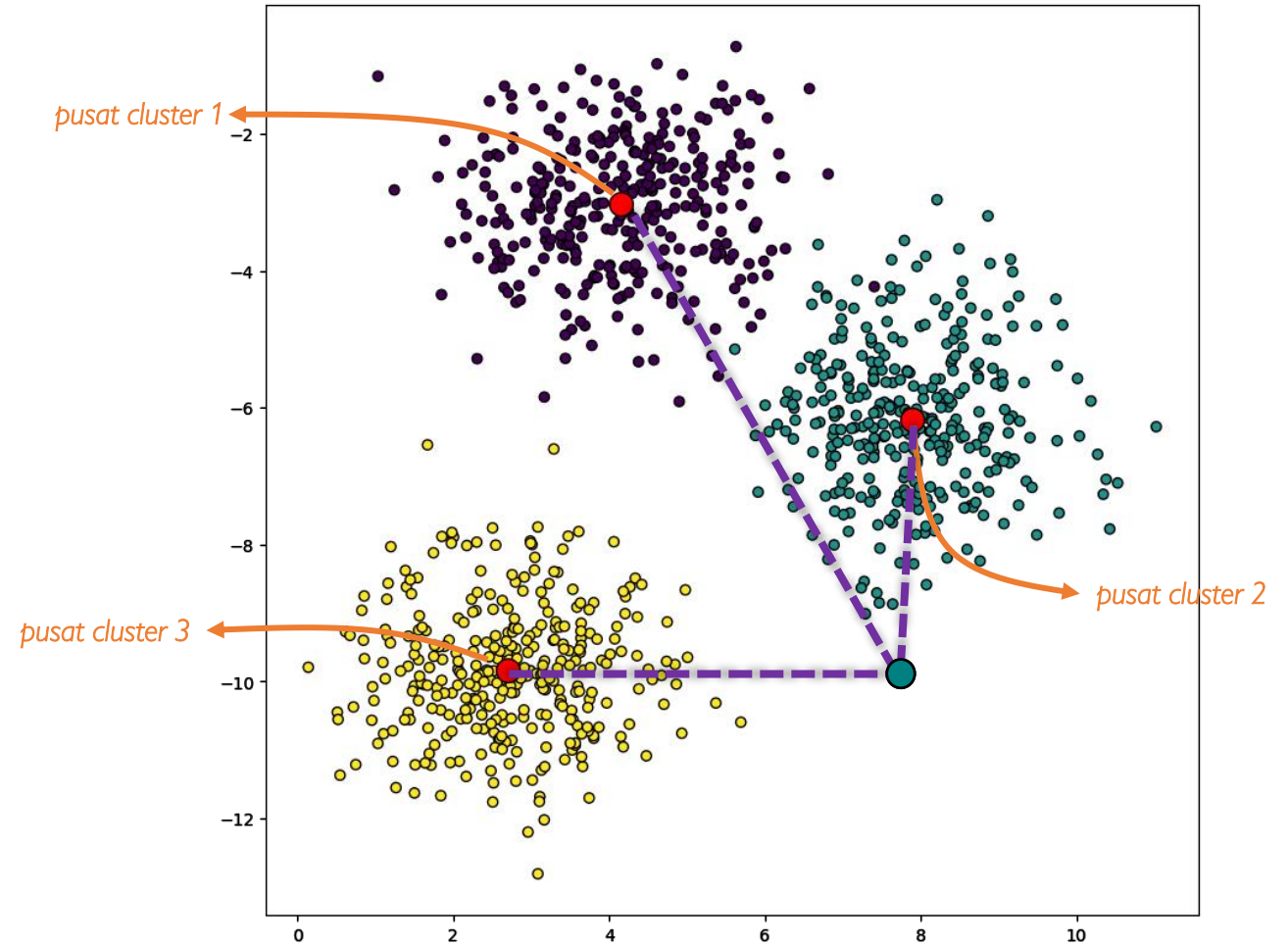
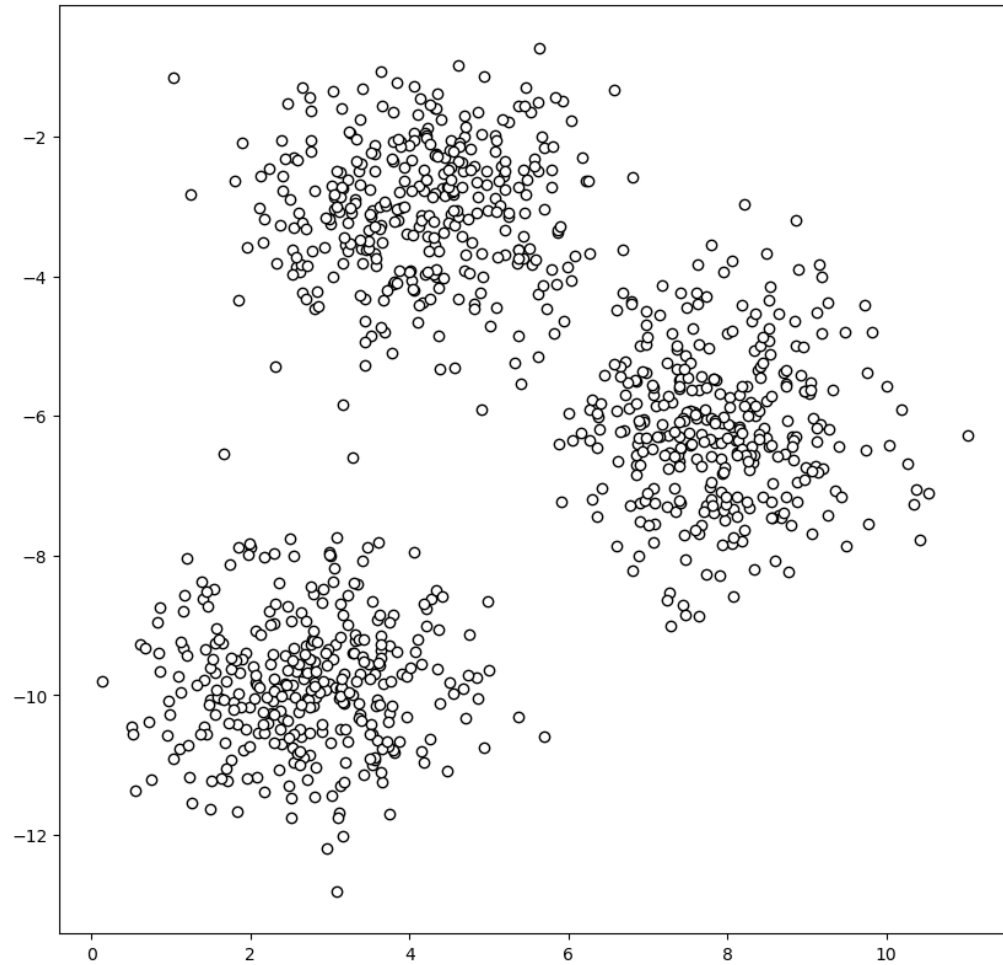
Proses **pengelompokan** kumpulan dari objek data ke dalam beberapa **kelompok (cluster)** tertentu, dimana objek data dalam cluster yang sama memiliki tingkat kesamaan (**similaritas**) yang tinggi, namun sangat jauh berbeda dibandingkan dengan objek data/memiliki **disimilaritas** yang tinggi.

disimilaritas dihitung dari jarak antar nilai pada masing-masing atribut objek

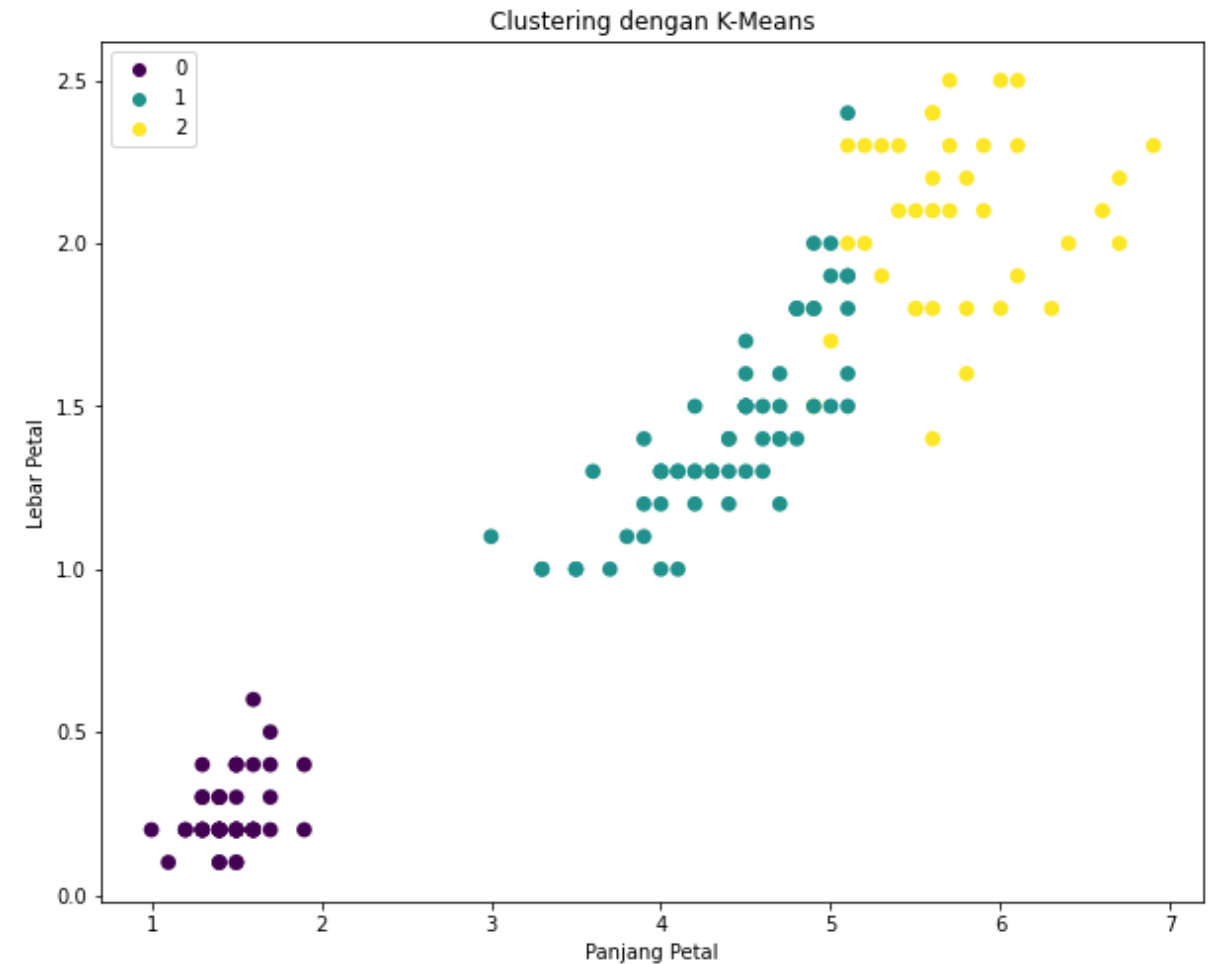
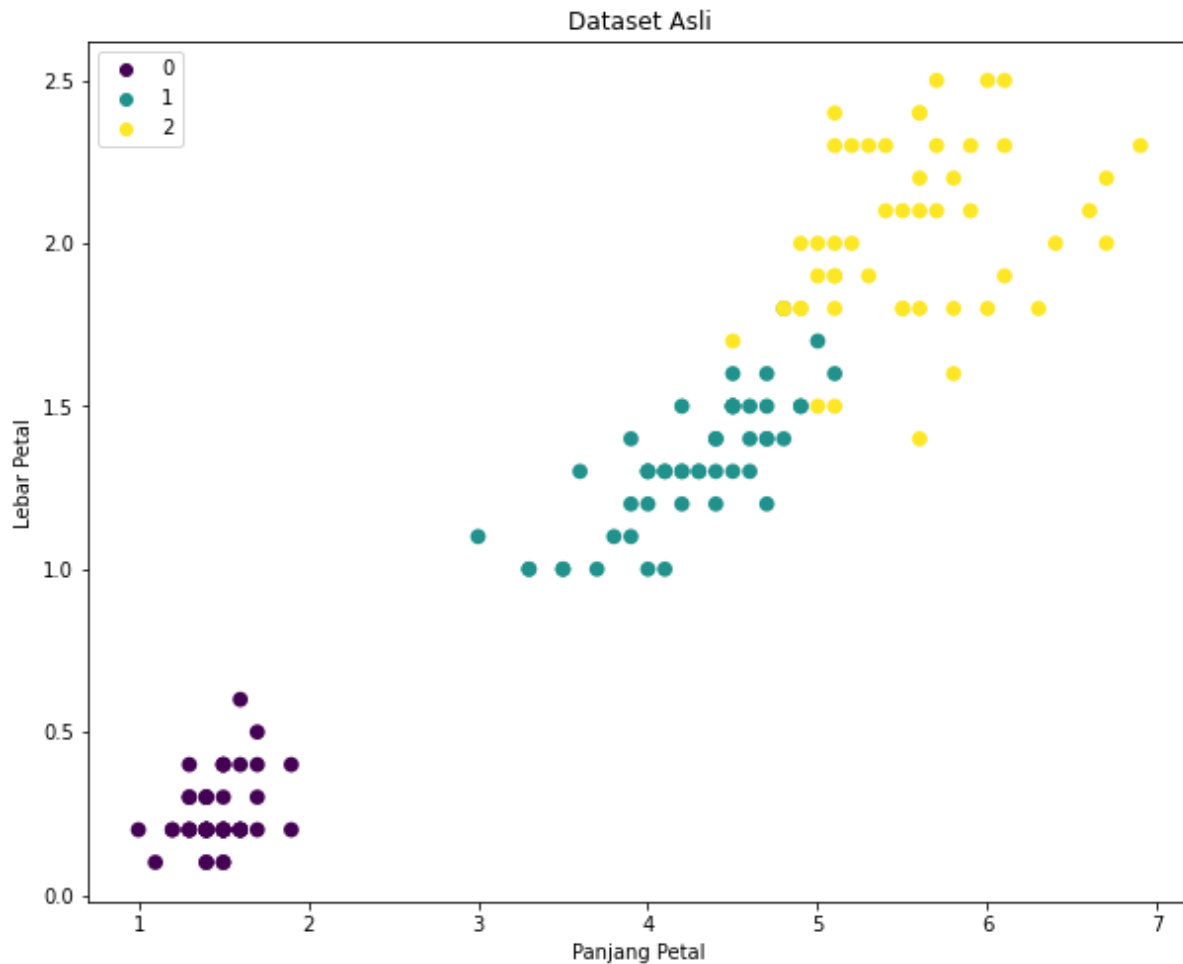
Penerapan :

- *Deteksi hoax*
- *Filter spam*
- *Marketing*
- *Deteksi aktivitas kriminal atau penipuan*
- *Analisis dokumen*

Clustering



K-Means pada Dataset IRIS



Metode Clustering

- Partitioning

Metode ini membentuk data ke dalam partisi, dimana setiap partisi merepresentasikan cluster.

- K-means
- K-medoid
- Fuzzy c-means

- Density-based

Metode ini membentuk cluster dengan mempertimbangkan kerapatan (jumlah data) dalam area terdekat.

- DBSCAN

- Hierarchical

Metode ini membentuk cluster dalam bentuk

dekomposisi hirarki

- BIRCH
- CURE, OPTICS

- Grid-based

Metode ini membentuk cluster ke dalam bentuk struktur jaringan

- Model-based

Metode cluster ini memperkenalkan probability cluster, tidak seperti k-means yang membentuk hard-cluster

- SOM
- EM algorithm

DISIMILARITAS

Disimilaritas

- Untuk banyak permasalahan, kita memerlukan kuantifikasi seberapa dekat antara dua buah objek.
- Disimilaritas adalah ketidakmiripan dari suatu objek. Kebalikannya adalah similaritas.
- Disimilaritas memiliki rentang nilai 0-1.
- 0 adalah sangat mirip, 1 adalah tidak mirip
- Contoh :
 - Segmentasi pasar
 - Pencarian dokumen digital
 - Pengecekan transaksi yang ganjil
- Untuk menyelesaikan permasalahan ini kita memerlukan definisi disimilaritas atau distance
- Perhitungan similaritas akan berbeda untuk setiap tipe atribut

Disimilaritas: Atribut Nominal

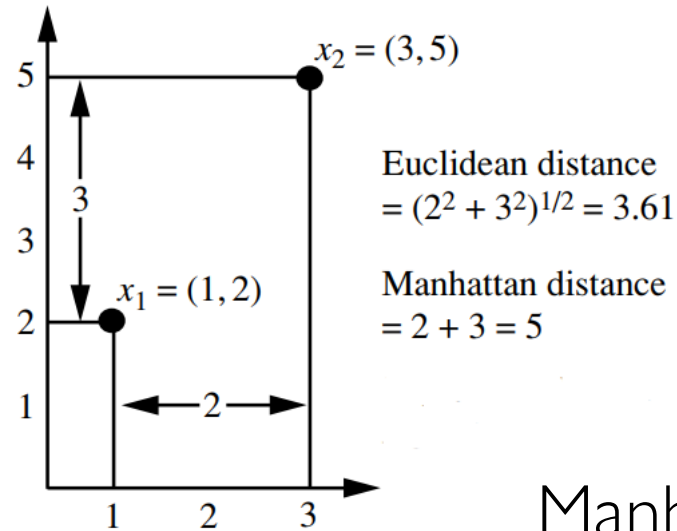
$$d(i, j) = \frac{p - m}{p}$$

- m adalah jumlah atribut yang memiliki nilai yang sama
- $d(i, j)$ adalah distance/jarak antara objek i dan j
- p adalah jumlah keseluruhan atribut

Objek data	domisili	warna rambut	gender
A	Denpasar	Hitam	Pria
B	Gianyar	Hitam	Pria

$$d(A, B) = \frac{p - m}{p} = \frac{3 - 2}{3} = 0.33$$

Disimilaritas: Atribut Numerik



Euclidean distance

$$d_{(a,b)} = \sqrt{\sum_{i=1}^n (a_i - b_i)^2}$$
$$= \sqrt{(a_1 - b_1)^2 + (a_2 - b_2)^2 + \dots + (a_n - b_n)^2}$$

Manhattan distance

$$d_{(a,b)} = \sum_{i=1}^n |a_i - b_i|$$
$$= |a_1 - b_1| + |a_2 - b_2| + \dots + |a_n - b_n|$$

Disimilaritas: Atribut Numerik

Objek data	berat	tinggi	umur
A	75	180	25
B	65	170	23

Euclidean distance

$$\begin{aligned}d_{(A,B)} &= \sqrt{\sum_{i=1}^n (A_i - B_i)^2} \\&= \sqrt{(A_1 - B_1)^2 + (A_2 - B_2)^2 + (A_3 - B_3)^2} \\&= \sqrt{(75 - 65)^2 + (180 - 170)^2 + (25 - 23)^2} \\&= \sqrt{(10)^2 + (10)^2 + (2)^2} \\&= \sqrt{204} = 14.28\end{aligned}$$

Manhattan distance

$$\begin{aligned}d_{(A,B)} &= \sum_{i=1}^n |A_i - B_i| \\&= |A_1 - B_1| + |A_2 - B_2| + |A_3 - B_3| \\&= |75 - 65| + |180 - 170| + |25 - 23| \\&= |10| + |10| + |2| = 22\end{aligned}$$

note : untuk atribut numerik, seharusnya kita melakukan normalisasi terlebih dahulu. namun, hanya sebagai contoh perhitungan, kita tidak akan melakukannya.

K-MEANS

Algoritma K-Means

Algoritma :

K-Means

Input :

k = jumlah cluster

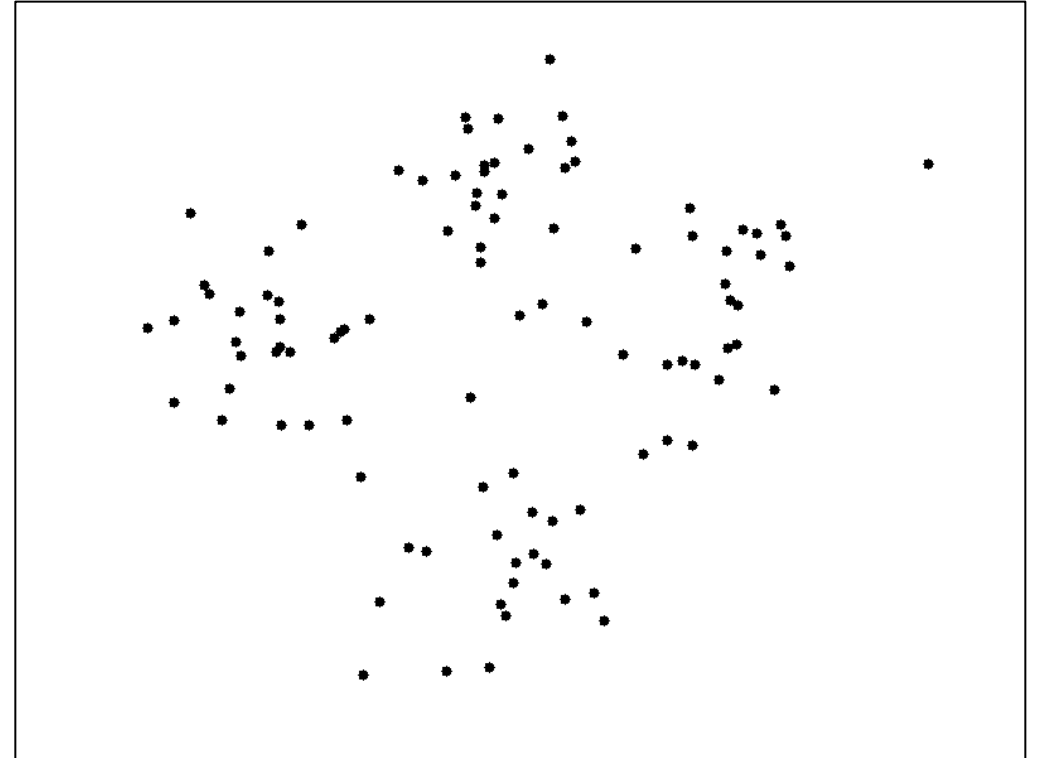
D = dataset yang memiliki n buah objek

Output :

sekumpulan cluster

Metode :

- 1) tentukan centroid sejumlah cluster
- 2) **repeat**
- 3) kelompokkan setiap objek ke centroid berdasarkan jarak terdekat
- 4) tentukan posisi centroid baru berdasarkan nilai rerata atribut setiap objek
- 5) **until** tidak ada perubahan keanggotaan



Contoh: K-Means

Objek Data	x	y
A	1	2
B	2	3
C	4	3
D	5	4



Objek Data	dist-C ₁	dist-C ₂
A	2	3.6
B	3.2	2.2
C	4.2	1
D	5.7	1

k = 2

C₁ = (1,0)

C₂ = (4,4)

Euclidean

Hitung jarak objek dengan centroid :

$$\begin{aligned}
 d_{(A,C_1)} &= \sqrt{(1-1)^2 + (2-0)^2} = 2 \\
 d_{(A,C_2)} &= \sqrt{(1-4)^2 + (2-4)^2} = 3.6 \\
 d_{(B,C_1)} &= \sqrt{(2-1)^2 + (3-0)^2} = 3.2 \\
 d_{(B,C_2)} &= \sqrt{(2-4)^2 + (3-4)^2} = 2.2 \\
 d_{(C,C_1)} &= \sqrt{(4-1)^2 + (3-0)^2} = 4.2 \\
 d_{(C,C_2)} &= \sqrt{(4-4)^2 + (3-4)^2} = 1 \\
 d_{(D,C_1)} &= \sqrt{(5-1)^2 + (4-0)^2} = 5.7 \\
 d_{(D,C_2)} &= \sqrt{(5-4)^2 + (4-4)^2} = 1
 \end{aligned}$$

Centroid baru

C₁ = {A}

x = 1

y = 2

C₂ = {B, C, D}

$$x = \frac{2+4+5}{3} = 3.6$$

$$y = \frac{3+3+4}{3} = 3.3$$

Contoh: K-Means

Objek Data	x	y
A	1	2
B	2	3
C	4	3
D	5	4



Objek Data	dist-C ₁	dist-C ₂
A	0	2.9
B	1.4	1.6
C	3.1	0.5
D	4.5	1.5

k = 2

C₁ = (1,2)

C₂ = (3.6,3.3)

Euclidean

Hitung jarak objek dengan centroid :

$$d_{(A,C_1)} = \sqrt{(1-1)^2 + (2-2)^2} = 0$$

$$d_{(A,C_2)} = \sqrt{(1-3.6)^2 + (2-3.3)^2} = 2.9$$

$$d_{(B,C_1)} = \sqrt{(2-1)^2 + (3-2)^2} = 1.4$$

$$d_{(B,C_2)} = \sqrt{(2-3.6)^2 + (3-3.3)^2} = 1.6$$

$$d_{(C,C_1)} = \sqrt{(4-1)^2 + (3-2)^2} = 3.1$$

$$d_{(C,C_2)} = \sqrt{(4-3.6)^2 + (3-3.3)^2} = 0.5$$

$$d_{(D,C_1)} = \sqrt{(5-1)^2 + (4-2)^2} = 4.5$$

$$d_{(D,C_2)} = \sqrt{(5-3.6)^2 + (4-3.3)^2} = 1.5$$

Centroid baru

C₁ = {A, B}

$$x = \frac{1+2}{2} = 1.5$$

$$y = \frac{2+3}{2} = 2.5$$

C₂ = {C, D}

$$x = \frac{4+5}{2} = 4.5$$

$$y = \frac{3+4}{2} = 3.5$$

Contoh: K-Means

Objek Data	x	y
A	1	2
B	2	3
C	4	3
D	5	4



Objek Data	dist-C ₁	dist-C ₂
A	0.7	3.8
B	0.7	2.5
C	2.5	0.7
D	3.8	0.7

k = 2

C₁ = (1.5, 2.5)

C₂ = (4.5, 3.5)

Euclidean

Hitung jarak objek dengan centroid :

$$d_{(A,C_1)} = \sqrt{(1 - 1.5)^2 + (2 - 2.5)^2} = 0.7$$

$$d_{(A,C_2)} = \sqrt{(1 - 4.5)^2 + (2 - 3.5)^2} = 3.8$$

$$d_{(B,C_1)} = \sqrt{(2 - 1.5)^2 + (3 - 2.5)^2} = 0.7$$

$$d_{(B,C_2)} = \sqrt{(2 - 4.5)^2 + (3 - 3.5)^2} = 2.5$$

$$d_{(C,C_1)} = \sqrt{(4 - 1.5)^2 + (3 - 2.5)^2} = 2.5$$

$$d_{(C,C_2)} = \sqrt{(4 - 4.5)^2 + (3 - 3.5)^2} = 0.7$$

$$d_{(D,C_1)} = \sqrt{(5 - 1.5)^2 + (4 - 2.5)^2} = 3.8$$

$$d_{(D,C_2)} = \sqrt{(5 - 4.5)^2 + (4 - 3.5)^2} = 0.7$$

Tidak ada perubahan keanggotaan

FINISH

K-MEDOIDS

Algoritma K-Medoids

Algoritma :

PAM (Partitioning Around Medoids)

Input :

k = jumlah cluster

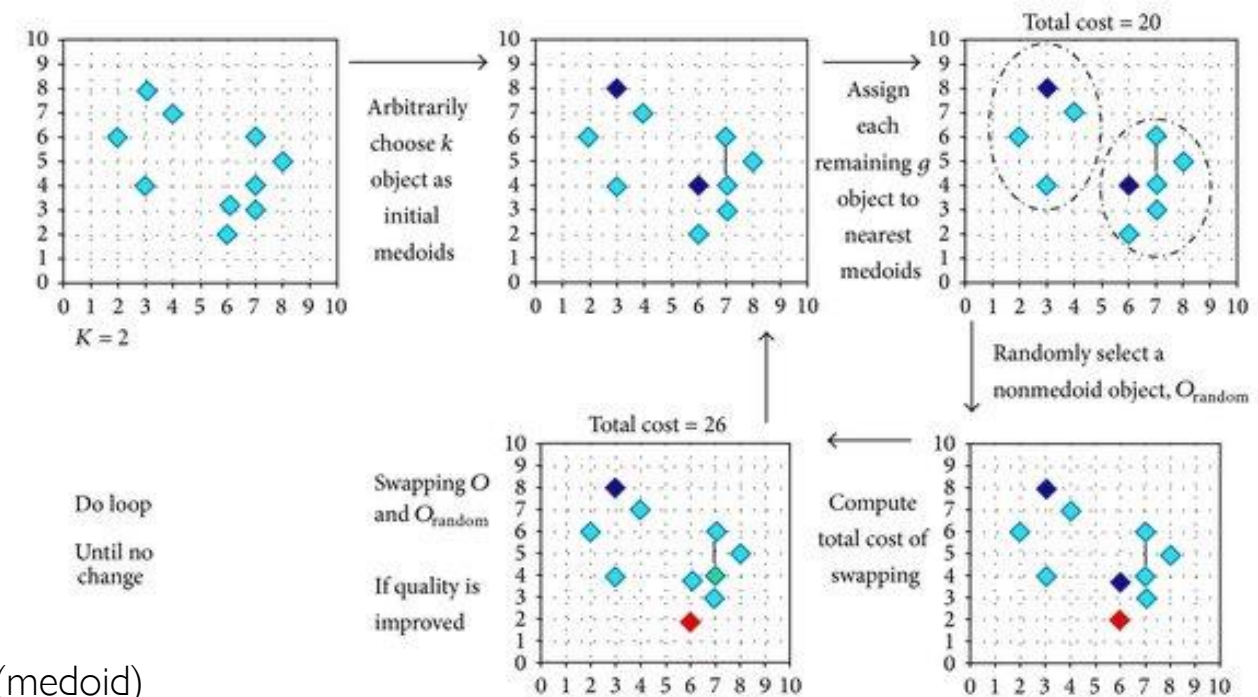
D = dataset yang memiliki n buah objek

Output :

sekumpulan cluster

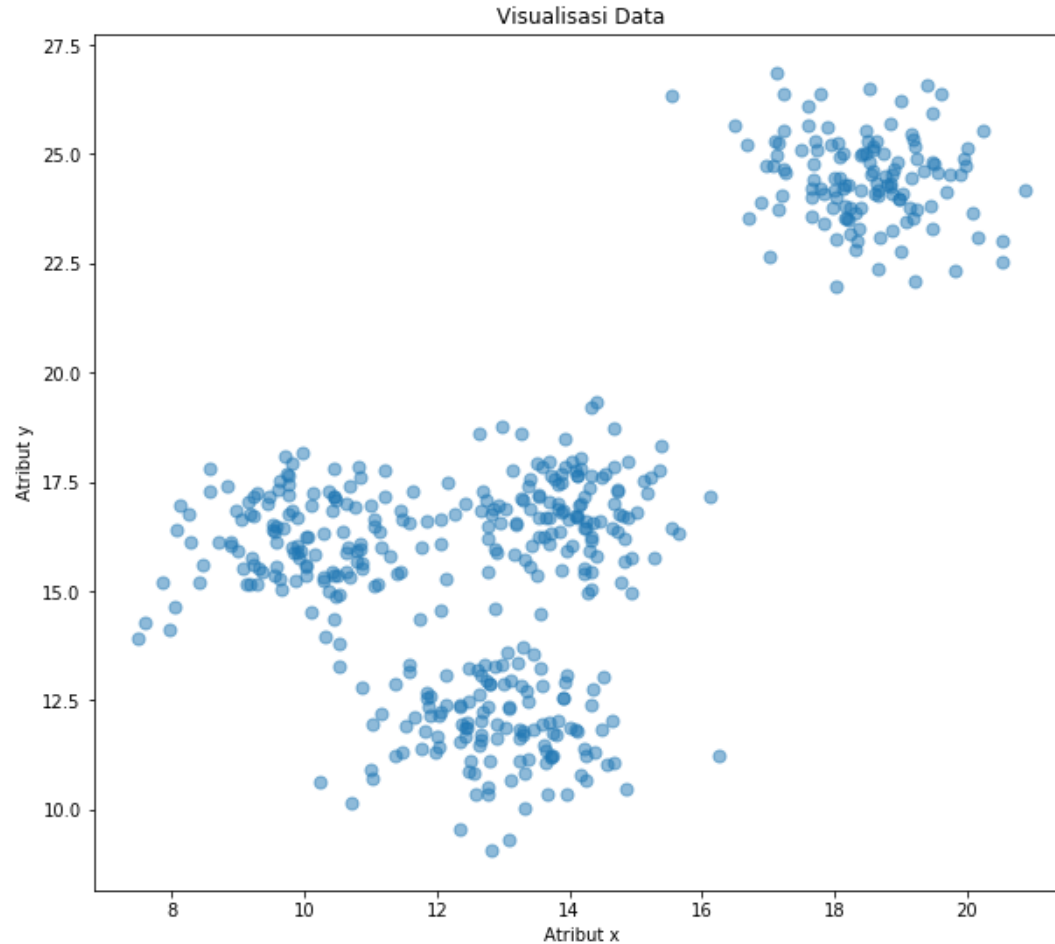
Metode :

- 1) pilih secara acak k objek data dari D sebagai pusat cluster (medoid)
- 2) repeat
- 3) kelompokkan setiap objek ke medoid berdasarkan jarak terdekat
- 4) secara acak pilih objek selain medoid sebagai medoid baru, o_{random}
- 5) hitung cost S , pergantian dari o_j dengan o_{random}
- 6) if $S < 0$ then tukar o_j dengan o_{random}
- 7) until tidak ada perubahan keanggotaan



EVALUASI CLUSTER

Jumlah Cluster Terbaik



*bagaimana cara menilai
bahwa jumlah cluster yang
digunakan adalah yang
terbaik?*

objek dalam satu cluster harus mirip, namun sangat berbeda dengan objek pada cluster yang lain

Metode Silhoutte

Metode Silhoutte

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

hitung jarak anggota dalam satu cluster

$$(2) \quad b(i) = \min_{k \neq i} \frac{1}{|C_k|} \sum_{j \in C_k} d(i, j)$$

hitung jarak anggota dalam cluster yang berbeda

$$(3) \quad s(i) = \frac{b(i) - a(i)}{\max\{a(i), b(i)\}}$$

hitung silhouette score

$$s(i) = \begin{cases} 1 - a(i)/b(i), & \text{if } a(i) < b(i) \\ 0, & \text{if } a(i) = b(i) \\ b(i)/a(i) - 1, & \text{if } a(i) > b(i) \end{cases}$$

$$-1 \leq s(i) \leq 1$$

*semakin tinggi nilai silhouette, maka posisi datum/objek dalam cluster semakin tepat

Contoh Perhitungan Silhoutte Score

Objek Data	x	y
A	1	2
B	2	3
C	4	3
D	5	4

- Hitung nilai silhouette objek A

$$a_A = \frac{1}{2-1} d_{(A,B)} = \frac{1}{1} \left(\left(\sqrt{(1-2)^2 + (2-3)^2} \right) \right) = 1.4$$

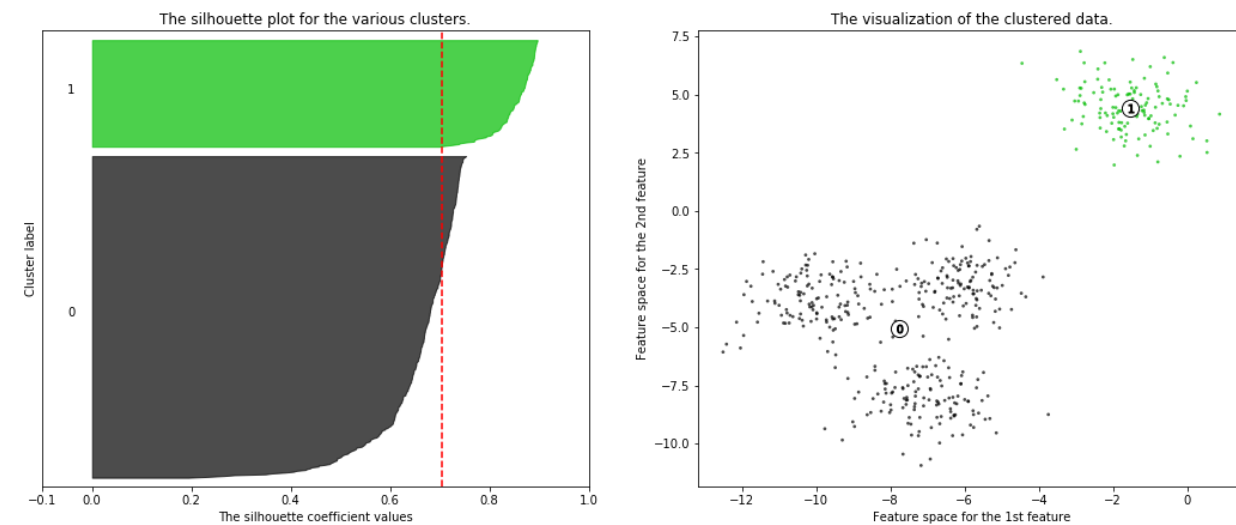
$$b_A = \frac{1}{2} (d_{(A,C)} + d_{(A,D)}) = \frac{1}{2} \left(\left(\sqrt{(1-4)^2 + (2-3)^2} \right) + \left(\sqrt{(1-5)^2 + (2-4)^2} \right) \right) = \frac{1}{2} (3.2 + 4.5) = 3.8$$

$$s_A = \frac{3.8 - 1.4}{\max(3.8, 1.4)} = \frac{2.4}{3.8} = 0.63$$

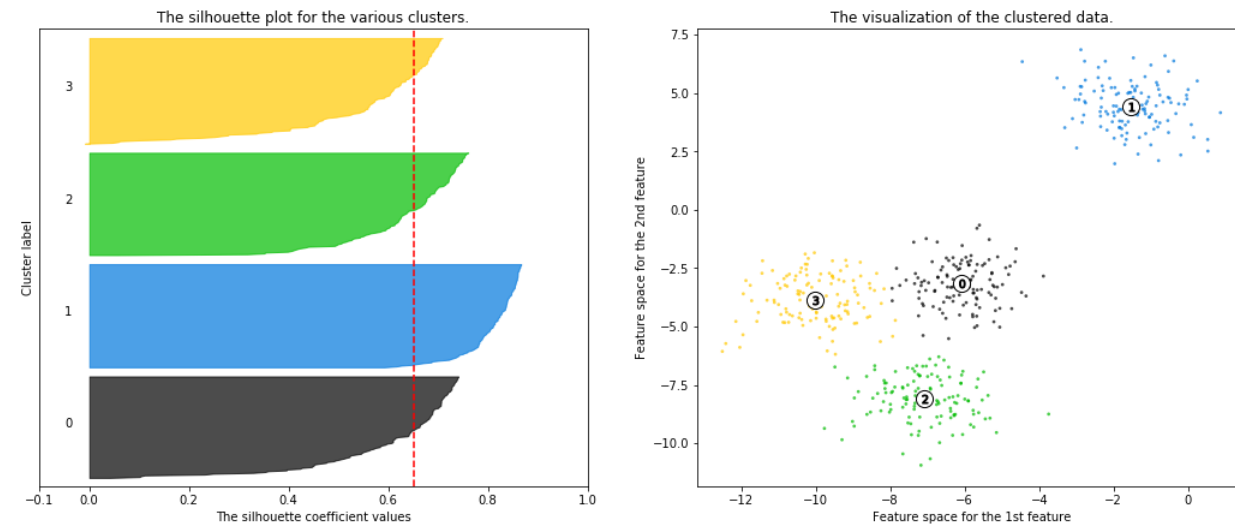
- Hitung nilai silhouette setiap objek, kemudian cari reratanya

*dalam modul scikit-learn metric `sklearn.metrics.silhouette_score`

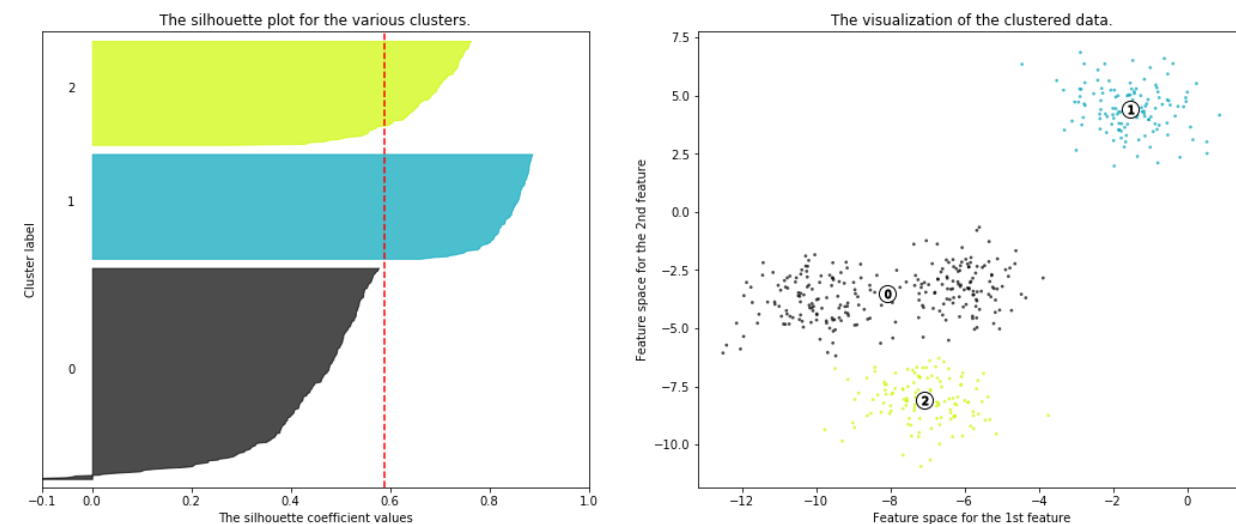
Silhouette analysis for KMeans clustering on sample data with $n_clusters = 2$



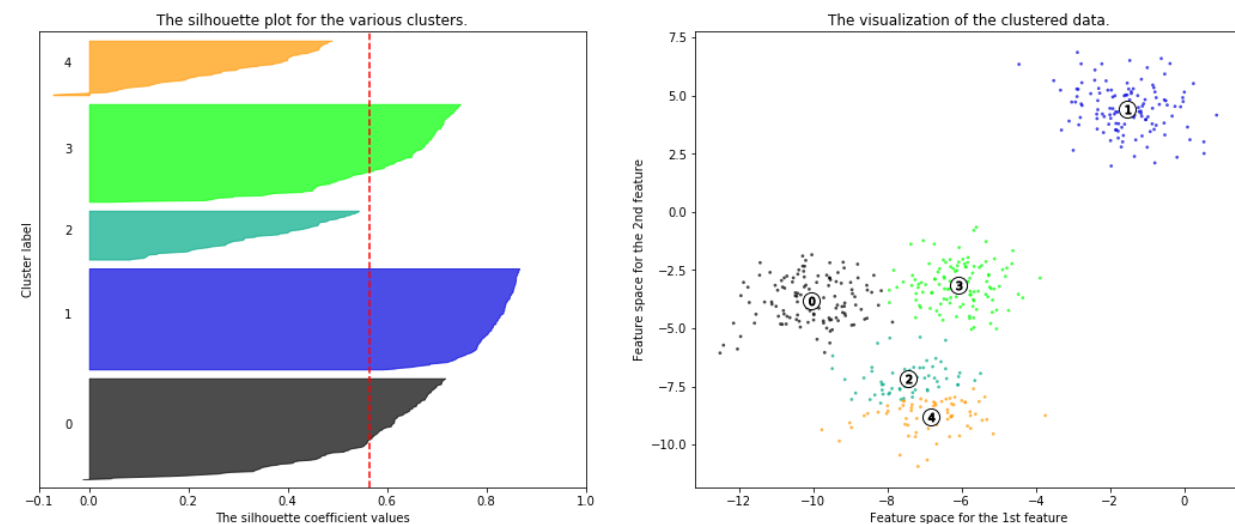
Silhouette analysis for KMeans clustering on sample data with $n_clusters = 4$



Silhouette analysis for KMeans clustering on sample data with $n_clusters = 3$



Silhouette analysis for KMeans clustering on sample data with $n_clusters = 5$



Pertama :
 Nilai rata-rata harus sedekat mungkin dengan 1

Kedua
 Plot masing-masing cluster harus di atas nilai rata-rata sebanyak mungkin.

Ketiga
 Lebar plot harus seseragam mungkin.


IMPLEMENTASI

Implementasi di Python

Untuk clustering, Anda dapat menggunakan library `pyclustering`

github : <https://github.com/annoviko/pyclustering>

pypi : <https://pypi.org/project/pyclustering/>



Search projects

Help Sponsor Log in Register

pyclustering 0.10.0.1

pip install pyclustering

Released: Aug 17, 2020

pyclustering is a python data mining library

Navigation

Project description

Release history

Download files

Project links

Homepage

Bug Tracker

Documentation

Repository

Project description

JOSS 10.21105/joss.01230

PyClustering

`pyclustering` is a Python, C++ data mining library (clustering algorithm, oscillatory networks, neural networks). The library provides Python and C++ implementations (via CCORE library) of each algorithm or model. CCORE library is a part of `pyclustering` and supported for Linux, Windows and MacOS operating systems.

Official repository: <https://github.com/annoviko/pyclustering/>

Documentation: <https://pyclustering.github.io/docs/0.10.0/html/>

Dependencies

Required packages: scipy, matplotlib, numpy, Pillow

README.rst

build passing build passing coverage 93% code documented pypi package 0.9.2 downloads 426k JOSS 10.21105/joss.01230

PyClustering

`pyclustering` is a Python, C++ data mining library (clustering algorithm, oscillatory networks, neural networks). The library provides Python and C++ implementations (via CCORE library) of each algorithm or model. CCORE library is a part of `pyclustering` and supported only for Linux, Windows and MacOS operating systems.

Version: 0.9.dev

License: GNU General Public License

E-Mail: pyclustering@yandex.ru

Documentation: <https://pyclustering.github.io/docs/0.9.2/html/index.html>

Homepage: <https://pyclustering.github.io/>

PyClustering Wiki: <https://github.com/annoviko/pyclustering/wiki>

Similar Projects

[scikit-learn](#): machine learning library for the Python. It features various classification, regression and clustering algorithms.

[ELKI](#): data mining software written in Java. The focus of ELKI is research in algorithms, with an emphasis on unsupervised methods in cluster analysis and outlier detection.

Dependencies

Required packages: scipy, matplotlib, numpy, Pillow

Python version: >= 3.5 (32, 64-bit)

C++ version: >= 14 (32, 64-bit)

Brief Overview of the Library Content

Clustering algorithms and methods (module `pyclustering.cluster`):

Algorithm	Python	C++
Agglomerative	✓	✓
BANG	✓	
BIRCH	✓	
BSAS	✓	✓
CLARANS	✓	
CLIQUE	✓	✓
CURE	✓	✓
DBSCAN	✓	✓
Elbow	✓	✓
EMA	✓	
Fuzzy C-Means	✓	✓
GA (Genetic Algorithm)	✓	✓
G-Means	✓	✓
HSyncNet	✓	✓
K-Means	✓	✓
K-Means++	✓	✓
K-Medians	✓	✓
K-Medoids	✓	✓
MBSAS	✓	✓
OPTICS	✓	✓
ROCK	✓	✓
Silhouette	✓	✓
SOM-SC	✓	✓
SyncNet	✓	✓
Sync-SOM	✓	
TTSAS	✓	✓
X-Means	✓	✓

K-Means

```
In [1]: 1 #import library
        2 from pyclustering.cluster.kmeans import kmeans
        3 from pyclustering.utils.metric import distance_metric, type_metric
```

```
In [2]: 1 #Tentukan data
        2 data = [[1,2],[2,3],[4,3],[5,4]]
        3 print(data)
```

```
[[1, 2], [2, 3], [4, 3], [5, 4]]
```

```
In [3]: 1 #Tentukan centroid awal
        2 start_centroid = [[1,0],[4,4]]
        3 #Tentukan rumus distance
        4 metric = distance_metric(type_metric.EUCLIDEAN)
```

```
In [4]: 1 #buat instance kmeans
        2 kmeans__ = kmeans(data, start_centroid, metric=metric)
        3 #jalankan kmeans
        4 kmeans__.process()
        5 #hitung cluster
        6 cluster_kmeans = kmeans__.get_clusters()
        7 #hitung centroid
        8 final_centroid = kmeans__.get_centers()
```

```
In [5]: 1 print(cluster_kmeans)
        2 print(final_centroid)
```

```
[[0, 1], [2, 3]]
[[1.5, 2.5], [4.5, 3.5]]
```

K-Medoids

```
In [1]: 1 from pyclustering.cluster.kmedoids import kmedoids
        2 from pyclustering.utils.metric import distance_metric, type_metric
```

```
In [2]: 1 #Tentukan data
        2 data = [[1,2],[2,3],[4,3],[5,4]]
        3 print(data)
```

```
[[1, 2], [2, 3], [4, 3], [5, 4]]
```

```
In [3]: 1 #Tentukan medoid awal
        2 start_medoid = [0,3]
        3 #Tentukan rumus distance
        4 metric = distance_metric(type_metric.EUCLIDEAN)
```

```
In [4]: 1 #buat instance kmedoids
        2 kmedoids__ = kmedoids(data, start_medoid, metric=metric)
        3 #jalankan kmeans
        4 kmedoids__.process()
        5 #hitung cluster
        6 cluster_kmedoids = kmedoids__.get_clusters()
        7 #hitung centroid
        8 final_medoids = kmedoids__.get_medoids()
```

```
In [5]: 1 print(cluster_kmedoids)
        2 print(kmedoids__.get_medoids())
```

```
[[0, 1], [3, 2]]
[0, 3]
```

