



Direktorat Jenderal Pendidikan Tinggi, Riset, dan, Teknologi
Kementerian Pendidikan, Kebudayaan, Riset, dan Teknologi
Republik Indonesia



MICROCREDENTIAL: ASSOCIATE DATA SCIENTIST

01 November – 10 Desember 2021

Pertemuan ke-15

Membangun Model 6 (Clustering)



[ditjen.dikti](https://www.facebook.com/ditjen.dikti)



@ditjendik
ti



[ditjen.dikti](https://www.instagram.com/ditjen.dikti/)



Ditjen
Diktristek



<https://dikti.kemdikbud.go.id/>



Profil Pengajar: Dzikri Rahadian Fudholi, S.Kom., M.Comp.



Jabatan Akademik: Tenaga Pengajar

Latar Belakang Pendidikan:

- S1: Ilmu Komputer – Universitas Gadjah Mada
- S2: Computing – The Australian National University.

Riwayat/Pengalaman Pekerjaan:

- Dosen
- Data Science
- Product Manager
- Data Engineer

Contak Pengajar:

Ponsel:

081393131133

Email:

dzikri.r.f@ugm.ac.id



Deskripsi Topik

KODE UNIT : **J.62DMI00.013.1**

JUDUL UNIT : **Membangun Model**

DESKRIPSI UNIT: Unit kompetensi ini berhubungan dengan pengetahuan, keterampilan, dan sikap kerja yang dibutuhkan dalam membangun model.

| ELEMEN KOMPETENSI | KRITERIA UNJUK KERJA |
|--------------------------------|--|
| 1. Menyiapkan parameter model | 1.1 Parameter-parameter yang sesuai dengan model diidentifikasi. 1.2 Nilai toleransi parameter evaluasi pengujian ditetapkan sesuai dengan tujuan teknis. |
| 2. Menggunakan tools pemodelan | 2.1 Tools untuk membuat model diidentifikasi sesuai dengan tujuan teknis <i>data science</i> . 2.2 Algoritma untuk teknik pemodelan yang ditentukan dibangun menggunakan <i>tools</i> yang dipilih. 2.3 Algoritma pemodelan dieksekusi sesuai dengan skenario pengujian dan <i>tools</i> untuk membuat model yang telah ditetapkan. 2.4 Parameter model algoritma dioptimasi untuk menghasilkan nilai parameter evaluasi yang sesuai dengan skenario pengujian. |

BATASAN VARIABEL

1. Konteks variabel
 - 1.1 Termasuk di dalam skenario pengujian adalah komposisi *data training* dan *data testing*, cara pemilihan *data training* dan *data testing* seperti *percentage splitting*, *random selection*, atau *cross validation*.
 - 1.2 Yang dimaksud dengan parameter model di antaranya arsitektur model, banyaknya *layer* atau simpul, *learning rate* untuk *neural network*, nilai *k* untuk *k-means*, nilai *pruning* untuk *decision tree*.
 - 1.3 Nilai parameter evaluasi adalah nilai ambang batas (*threshold*) yang bisa diterima.
 - 1.4 Yang dimaksud dengan *tools* pemodelan di antaranya perangkat lunak *data science* di antaranya: *rapid miner*, *weka*, atau *development* untuk bahasa pemrograman tertentu seperti *python* atau R.



Course Definition

Topik : Membangun Model (UK J.62DMloo.013.1 - Membangun Model)

Deskripsi topik : J.62DMloo.013.1 - Membangun Model (Clustering)

- a) Menyiapkan parameter model
- b) Menggunakan tools pemodelan
- c) Menjelaskan algoritma dan menggunakan Library K-Means, Hierarchical Clustering, DBSCAN.

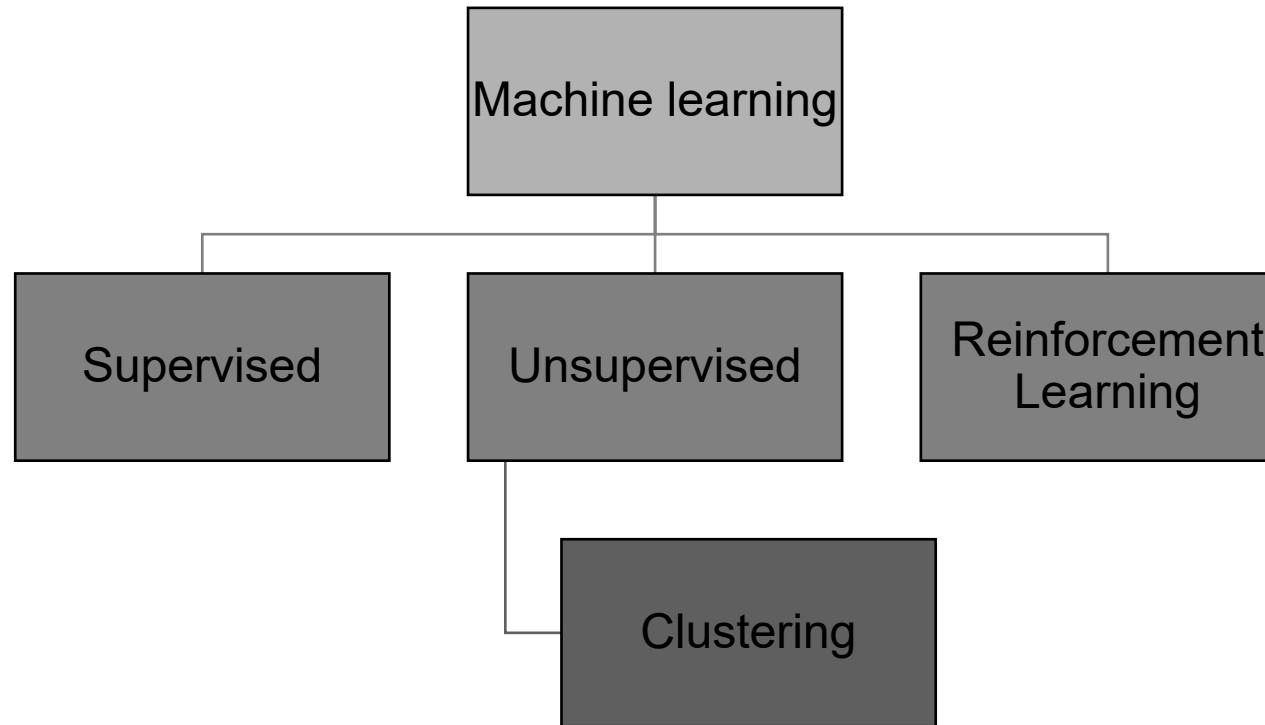
Learning Objective

In this course you will:

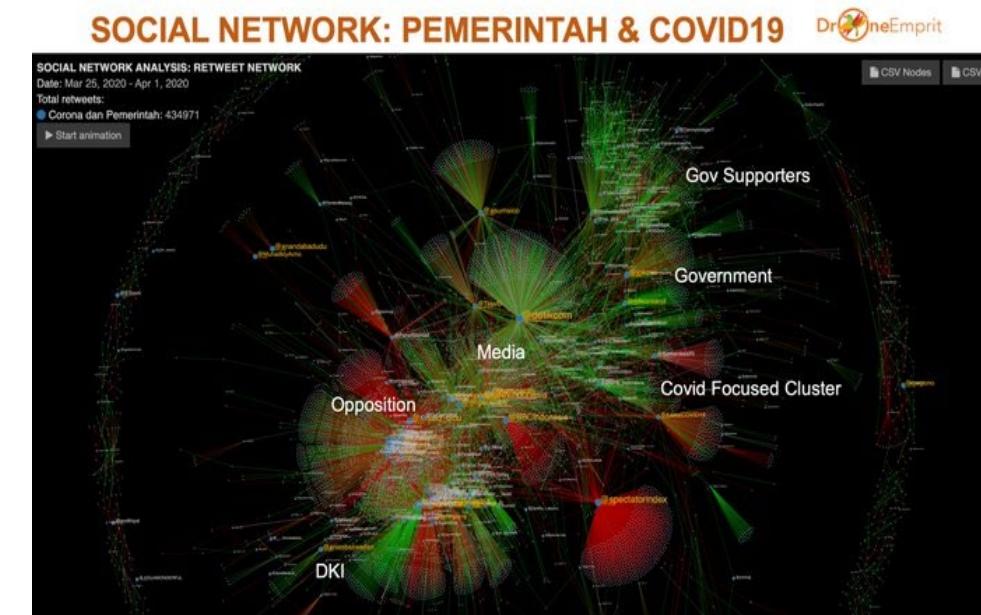
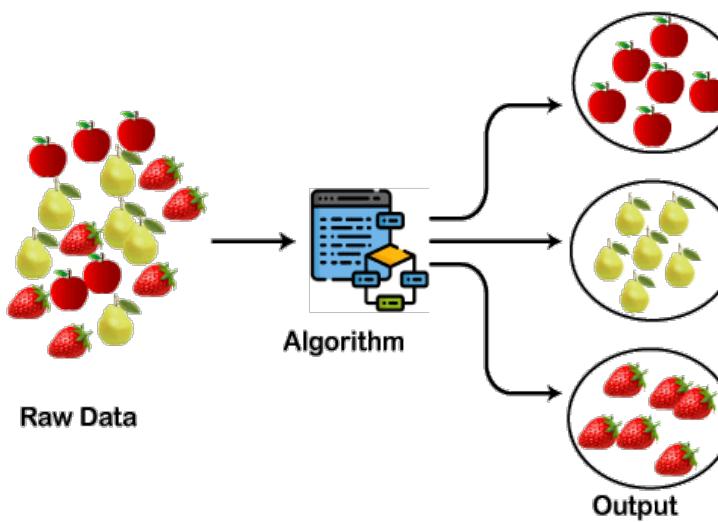
- A. Melakukan pemilihan data untuk digunakan sebagai input feature
 - B. Mempelajari algoritma clustering K-Means, Hierarchical Clustering, DBSCAN
 - C. Mempelajari optimasi K-Means dengan metode Elbow untuk menentukan jumlah K yang tepat
 - D. Mempelajari optimasi Hierarchical Clustering dengan Dendogram Diagram untuk menentukan jumlah K yang tepat
 - E. Mempelajari optimasi clustering untuk density data menggunakan metode DBSCAN
 - F. Praktek coding menggunakan python



Clustering



Clustering





Clustering

Teknik clustering melakukan pembelajaran mesin tanpa ada supervisi untuk menyelesaikan suatu masalah (identifikasi dan mengelompokkan titik data serupa dalam dataset yang lebih besar tanpa memperhatikan hasil/luaran tertentu)

Clustering juga dapat dianalogikan sebagai tugas mengidentifikasi subkelompok dalam data sedemikian rupa sehingga titik data dalam subkelompok yang sama (cluster) sangat mirip sedangkan titik data dalam cluster yang berbeda sangat berbeda

Keputusan tentang ukuran kemiripan dapat ditentukan melalui jumlah centroid masing-masing kelompok dan jaraknya.



Aplikasi Clustering

Beberapa penerapan clustering yang populer untuk:

- A. Recommendation engines
- B. Market segmentation
- C. Social network analysis
- D. Search result grouping
- E. Medical imaging
- F. Image segmentation
- G. Anomaly detection, dll



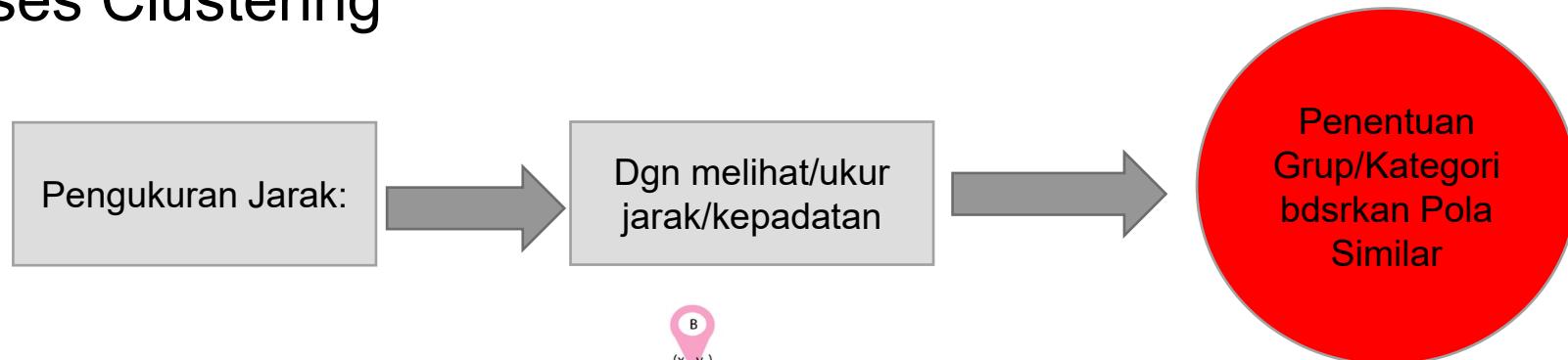


Kapan digunakannya Clustering

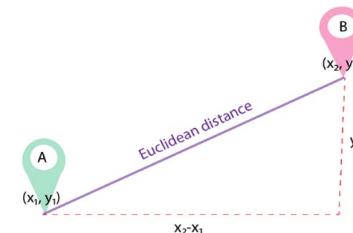
Clustering digunakan pada saat:

- dataset unstructured berukuran sangat besar: cluster memberikan jawaban cepat ttg data, dgn perorganisasian secara otomatis
- jumlah kelas pembagian dataset tidak diketahui: cluster adalah langkah pertama yang baik utk persiapan data, karena mulai menjawab pertanyaan kunci tentang kumpulan dataset
- proses pelabelan/anotasi manual dilakukan memerlukan banyak sumber daya: clustering dapat memotong waktu anotasi dan klasifikasi krn tidak terlalu memperhatikan luaran, namun lebih fokus pada pengkategorisasian.
- mencari anomali data: kebanyakan algoritma clustering, sensitif thdp data oulier. Pemahaman anomali data membantu optimasi tools koleksi data dan hasil lebih akurat

Proses Clustering



Euclidian



Manhattan

$$d(x, y) = \sum_{i=1}^k |x_i - y_i|$$

Minkowski

$$d(x, y) = \left(\sum_{i=1}^k (|x_i - y_i|)^q \right)^{1/q}$$

Hamming Distance

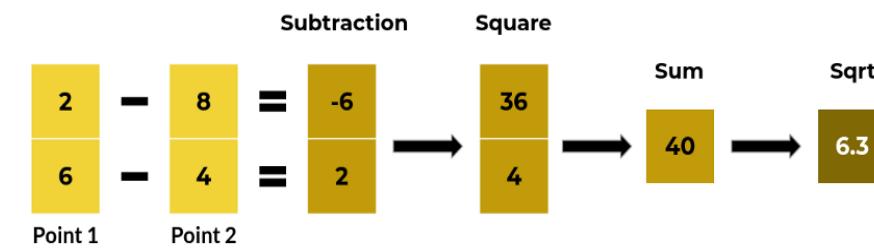
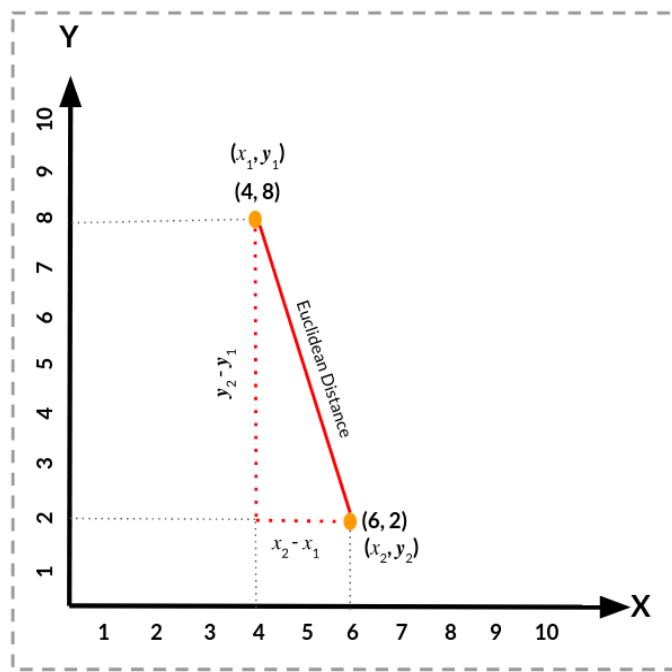
$$D_H = \sum_{i=1}^k |x_i - y_i|$$

Hamming

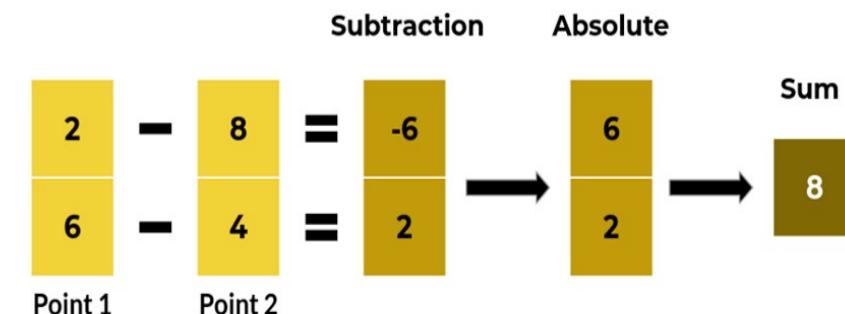
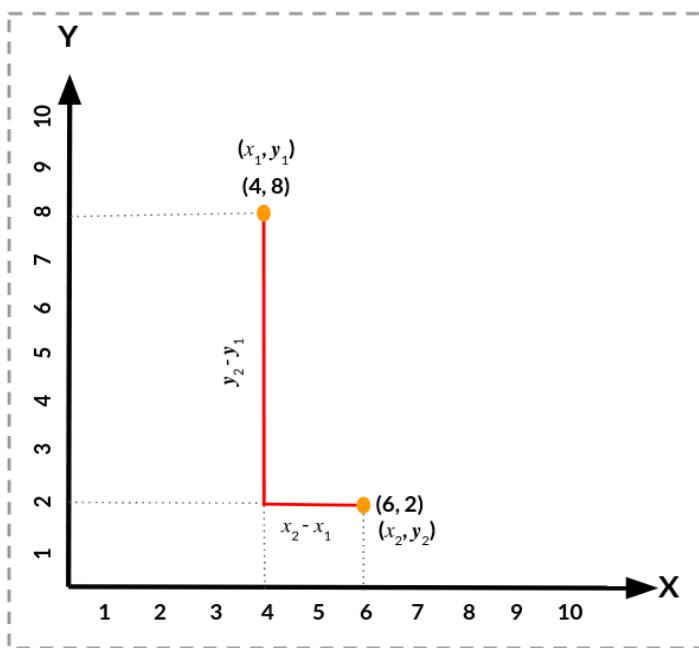
$$x = y \Rightarrow D = 0$$

$$x \neq y \Rightarrow D = 1$$

Contoh Euclidian Distance



Contoh Manhattan Distance



Jenis Clustering

Secara umum jenis Clustering dibagi 2 subgrup:

Lebih Umum dan Lebih
Mudah dilakukan

- Cocok utk aplikasi hirarki
yang browsable (mis. kita ingin
membagi sepatu menjadi 2 cluster:
sepatu olahraga dan sepatu kerja)

A

Hard Clustering

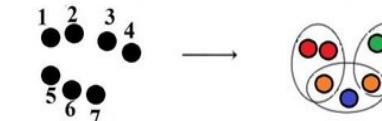
- Every node may belong to only one cluster



B

Soft Clustering

- Every node may belong to several clusters with a fractional degree of membership in each



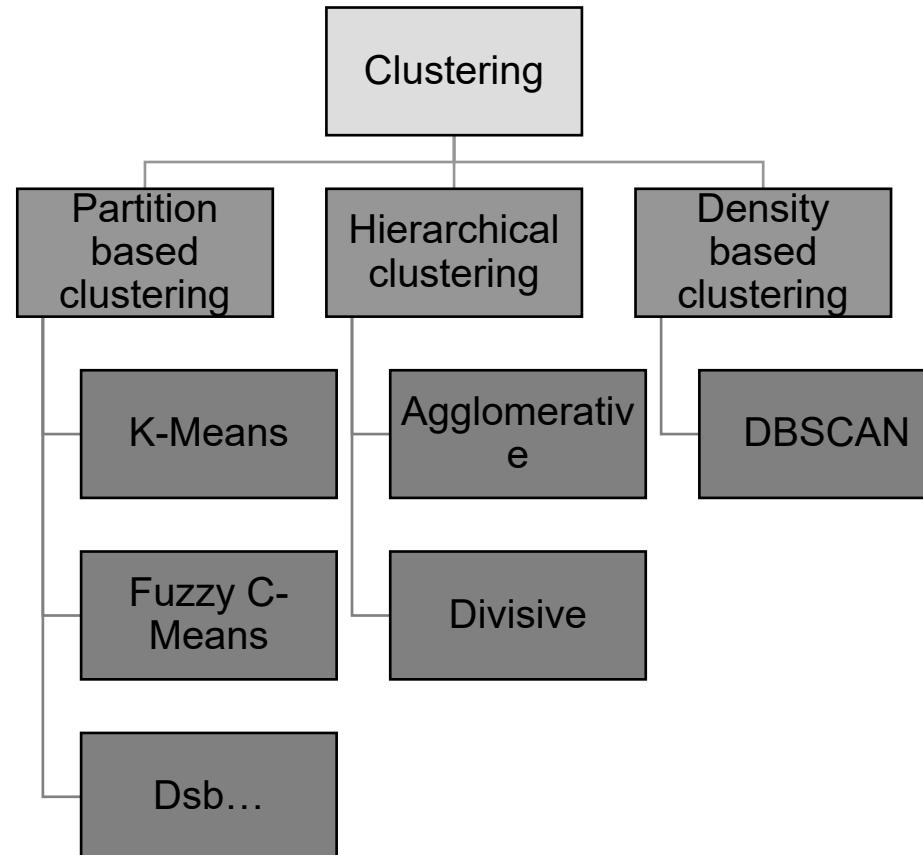
Community Affiliation

| | Nodes | 1 | 2 | 3 | 4 | 5 | 6 | 7 |
|-----------|-------|---|---|---|---|---|---|---|
| Cluster 1 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | |
| Cluster 2 | 0 | 0 | 1 | 1 | 0 | 0 | 0 | |
| Cluster 3 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | |

| | Nodes | 1 | 2 | 3 | 4 | 5 | 6 | 7 |
|-----------|-------|---|---|---|---|---|---|---|
| Cluster 1 | 1 | 1 | 0 | 0 | 1 | 0 | 0 | |
| Cluster 2 | 0 | 0 | 1 | 1 | 0 | 0 | 1 | |
| Cluster 3 | 0 | 0 | 0 | 0 | 1 | 1 | 1 | |



Clustering by Algorithm





Clustering

| Method | General Characteristics |
|-----------------------|--|
| Partitioning Methods | <ul style="list-style-type: none">• Find mutually exclusive clusters of spherical shape• Distance based• May use mean to represent cluster• Effective for small to medium data sets |
| Hierarchical Methods | <ul style="list-style-type: none">✓ Clustering is a hierarchical decomposition✓ Cannot correct erroneous merges or split✓ May incorporate other techniques like micro-clustering or object “linkages” |
| Density Based methods | <ul style="list-style-type: none">• Can find arbitrarily shaped clusters• Clusters are dense regions of objects in space that are separated by low-density regions• Cluster density: each point must have a minimum number of points within its “neighborhood”• May filter outliers |



Clustering Distance Measures

- Clustering data ke dalam kelompok memerlukan beberapa metode untuk menghitung jarak atau kesamaan antara setiap pasangan pengamatan. Hasil dari perhitungan ini dikenal sebagai *dissimilarity* atau *distance matrix*.
- Methods klasik utk *distance measures* :

Euclidean distance:

$$d_{euc}(x, y) = \sqrt{\sum_{i=1}^n (x_i - y_i)^2}$$

Manhattan distance:

$$d_{man}(x, y) = \sum_{i=1}^n |(x_i - y_i)|$$

dimana, x dan y adalah dua vectors dengan panjang n



Clustering Distance Measures

Pearson correlation distance:

$$d_{cor}(x, y) = 1 - \frac{\sum_{i=1}^n (x_i - \bar{x})(y_i - \bar{y})}{\sqrt{\sum_{i=1}^n (x_i - \bar{x})^2} \sqrt{\sum_{i=1}^n (y_i - \bar{y})^2}}$$

Pengukuran *dissimilarity* lainnya: correlation-based distances, yang banyak digunakan untuk analisis data ekspresi gen.

Spearman correlation distance:

$$d_{spear}(x, y) = 1 - \frac{\sum_{i=1}^n (x'_i - \bar{x}')(y'_i - \bar{y}')}{\sqrt{\sum_{i=1}^n (x'_i - \bar{x}')^2} \sqrt{\sum_{i=1}^n (y'_i - \bar{y}')^2}}$$

Dimana x'_i adalah rank (x_i) dan y'_i adalah rank (y_i)



K-Means

- Algoritma *K-means*: salah satu algoritma *clustering* yang bersifat **iteratif** yang mencoba untuk **mempartisi dataset** menjadi subkelompok non-overlapping berbeda yang ditentukan oleh K (cluster) **di mana setiap titik data hanya dimiliki oleh satu kelompok.**
- K-Means menetapkan poin data ke cluster sedemikian rupa sehingga **jumlah jarak kuadrat antara titik data dan pusat massa cluster (rata-rata aritmatika dari semua titik data yang termasuk dalam cluster itu)** minimal.
- Semakin **sedikit variasi** yang kita miliki dalam cluster, **semakin homogen** (serupa) titik data dalam cluster yang sama.

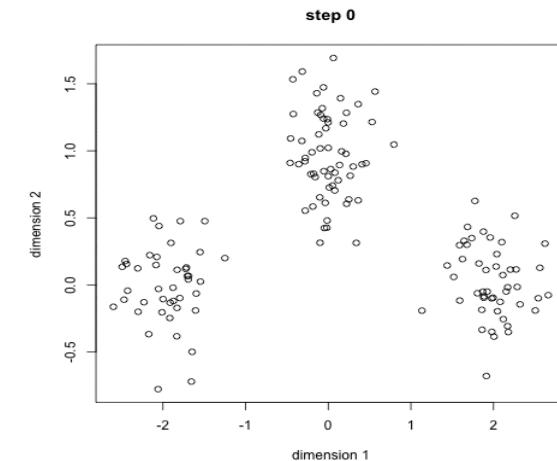


K-Means

$$W(C_k) = \sum_{x_i \in C_k} (x_i - \mu_k)^2$$

where:

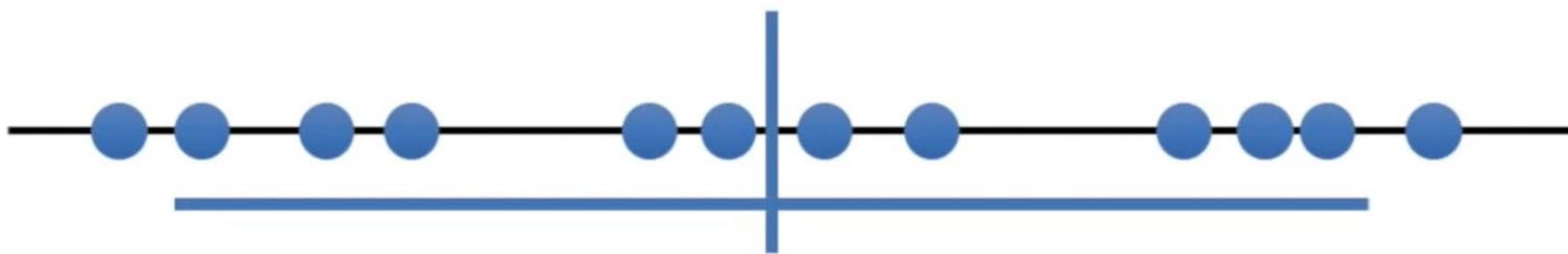
- x_i is a data point belonging to the cluster C_k
- μ_k is the mean value of the points assigned to the





Ilustrasi K-Means

Mulai dengan K=1



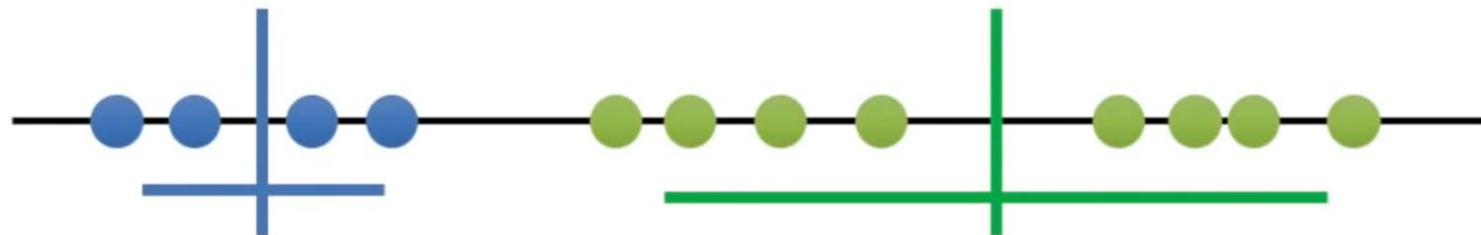
K=1 adalah skenario kasus terburuk.

Kita dapat kuantifikasi masalah skenario buruk ini dengan variasi total



Ilustrasi K-Means

Sekarang dgn $K = 2$



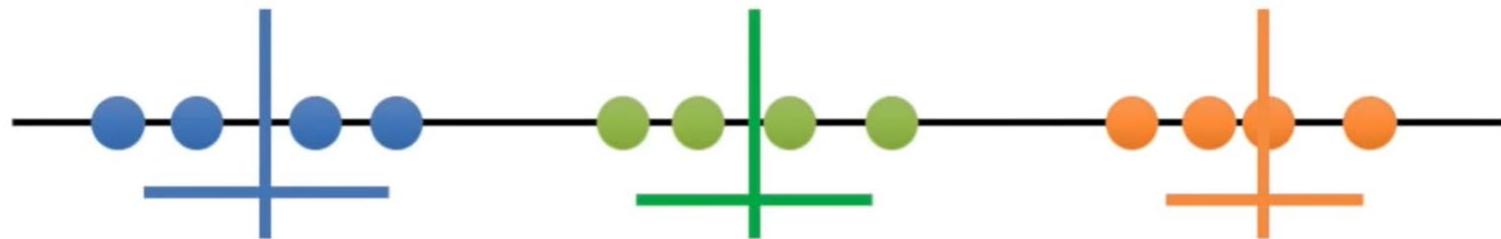
$K = 2$ lebih baik, dan kita dapat kuantifikasi nilai K yg lebih baik dengan membandingkan variasi total dalam 2 kluster terhadap $k = 1$





Ilustrasi K-Means

Sekarang coba dgn $K = 3$



$K = 3$ jadi lebih baik! dan kita dapat kuantifikasi nilai K yg lebih baik dengan membandingkan variasi total dalam 3 cluster terhadap $k = 2$

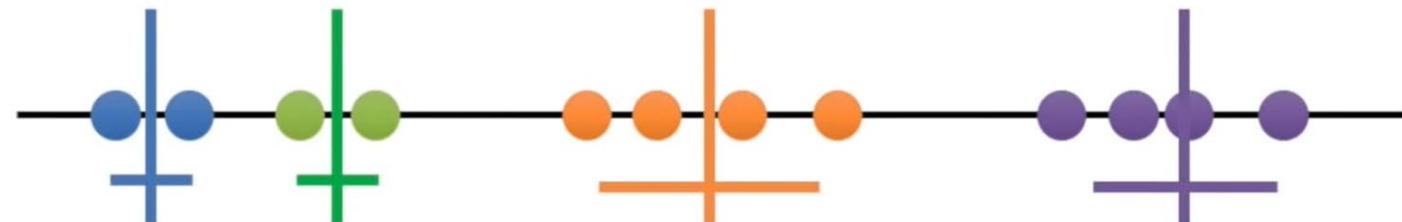
$K = 1$ ——————

$K = 2$ ——————

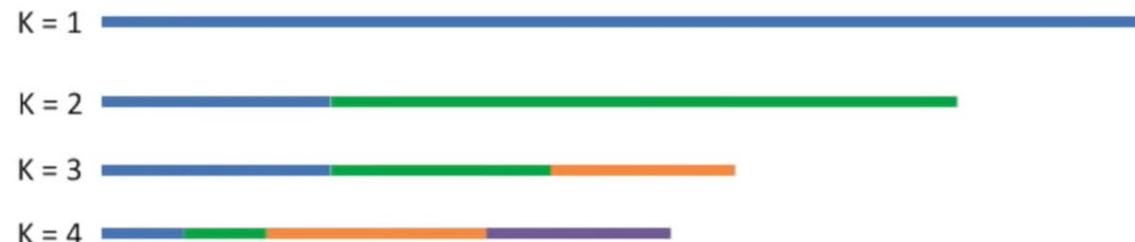
$K = 3$ ——————

Ilustrasi K-Means

Sekarang coba dgn $K = 4$

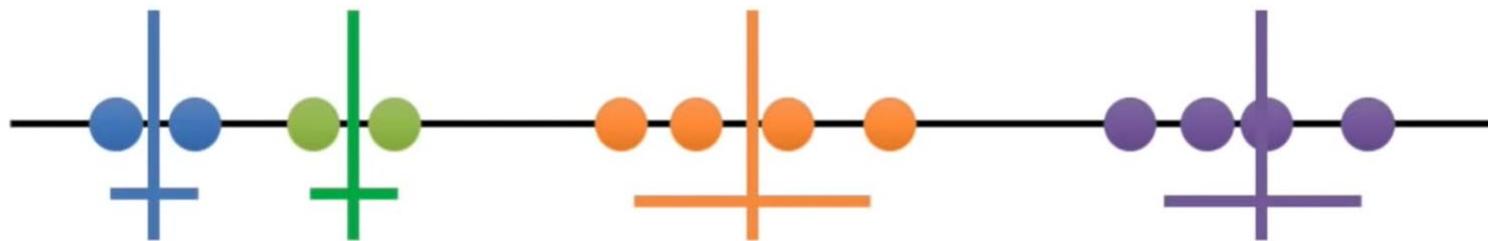


Variasi total dalam setiap cluster adalah menjadi semakin kurang dgn $K = 3$



Ilustrasi K-Means

dgn $K = 4$

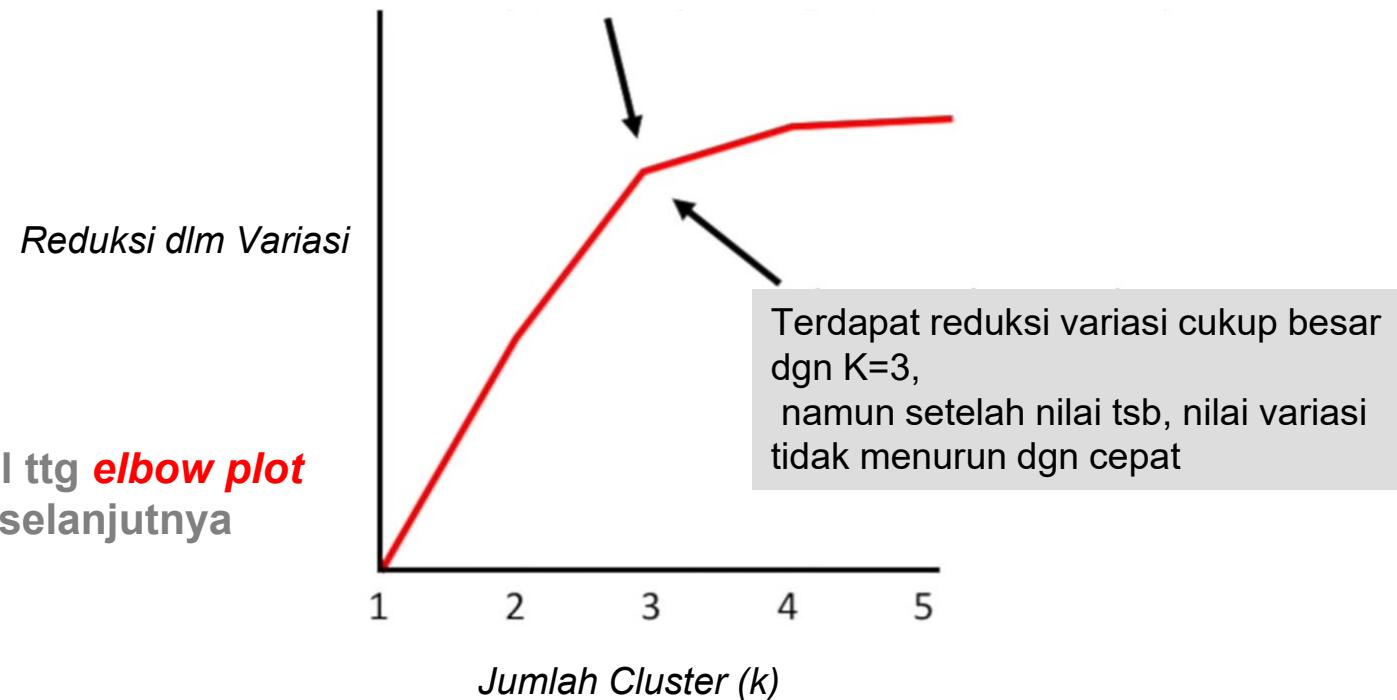


Variasi total dalam setiap cluster adalah menjadi semakin kecil dgn $K = 3$
Setiap kita tambahkan cluster baru, variasi total dalam setiap cluster semakin kecil
dari sebelumnya. Dan jika hanya ada 1 titik per cluster, maka variasi = 0
Bagaimanapun, jika kita plot reduksi dalam variansi per nilai utk $K ..$



Ilustrasi K-Means

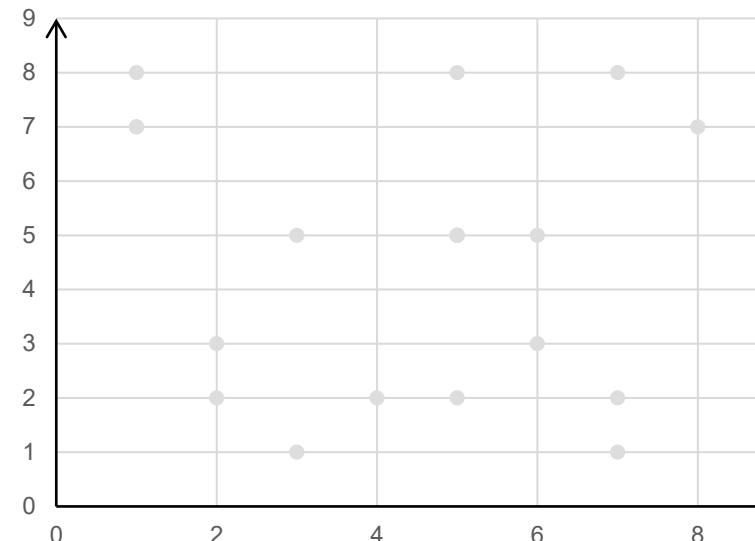
Disebut plot siku/*elbow*, dan anda dapat pilih “K” dengan mencari siku/*elbow* dalam plot



Langkah-langkah metode K-Means

1. Memilih jumlah *cluster* awal (K) yang ingin dibuat

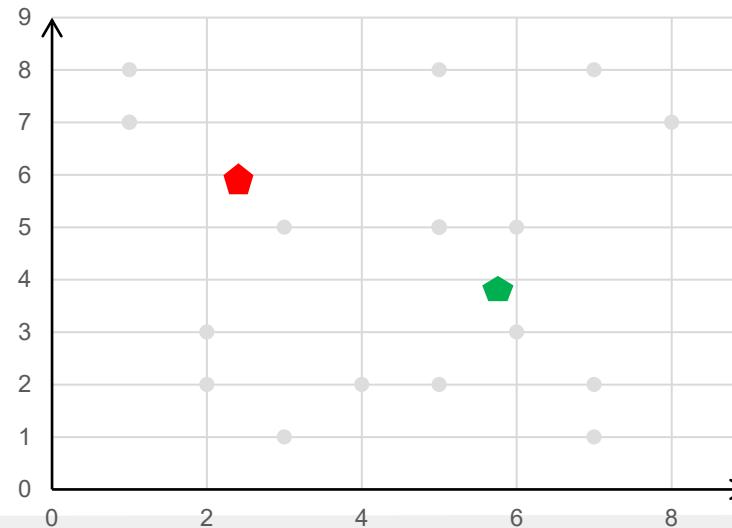
Sebagai contoh terdapat data 2 dimensi seperti yang ditampilkan dalam grafik, langkah pertama adalah memilih jumlah (K) kluster. Misal kita pilih untuk membaginya ke dalam $K=2$ kluster.



Langkah-langkah metode K-Means

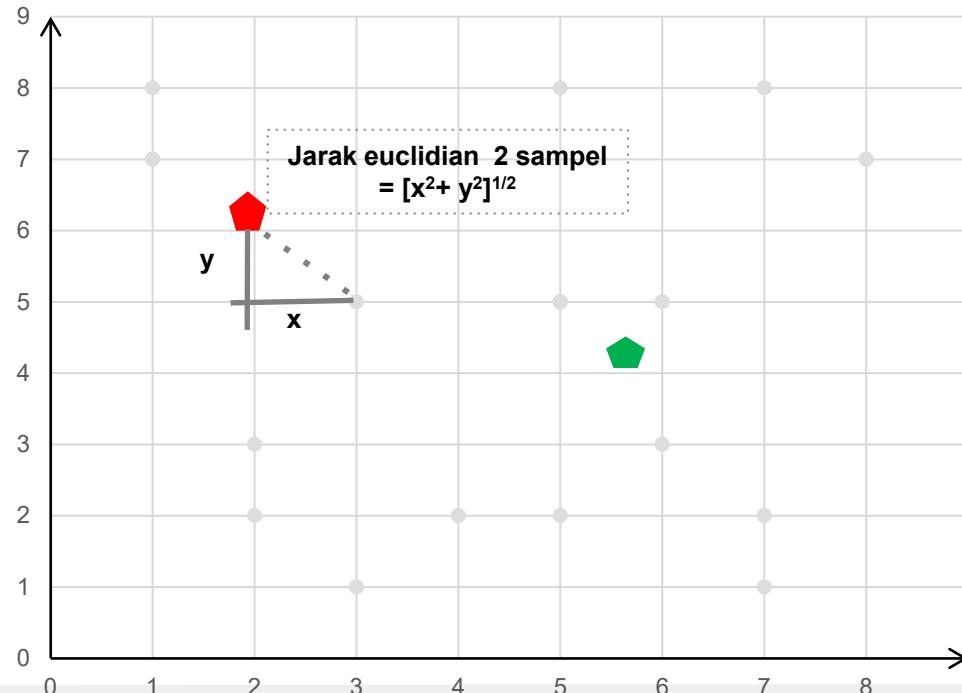
2. Memilih titik secara random sebanyak K buah, di mana titik ini akan menjadi pusat (*centroid*) dari masing-masing kelompok (*clusters*).

Langkah kedua adalah menentukan titik pusatnya. Dalam satu klaster terdapat satu titik pusat atau yang disebut dengan *centroid*. Penentuan awal posisi titik pusat ini bebas, karena nantinya algoritma *K-Means* akan merubah posisi tiap titik hingga dicapai solusi paling optimal. Pada Gambar, titik merah mewakili pusat dari kluster 1, dan biru untuk kluster 2. Dengan demikian, maka masing-masing data point akan memilih titik pusat (*centroid*) yang paling dekat. Jika sudah dipilih maka data point tersebut akan menjadi bagian dari klusternya





Langkah-langkah metode K-Means

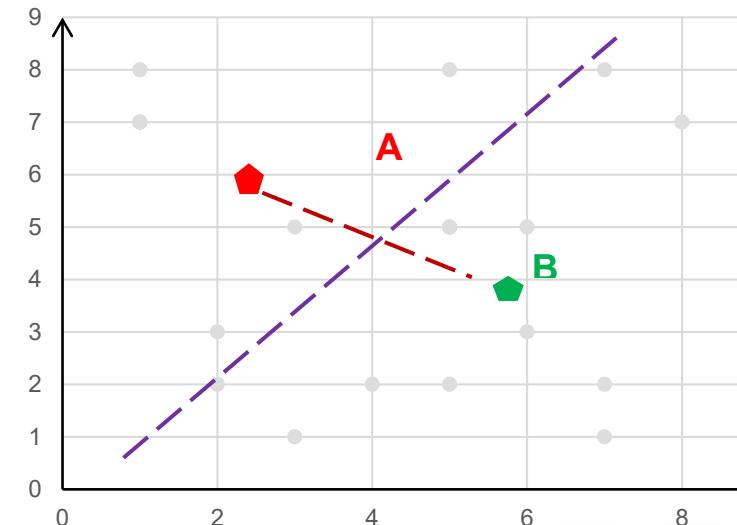


Jarak euclidian 3 sampel
 $= [x^2 + y^2 + z^2]^{1/2}$

Langkah-langkah metode K-Means

3. Dari dataset yang kita miliki, buat dataset yang terdekat dengan titik *centroid* sebagai bagian dari *cluster* tersebut. Sehingga secara total akan terbentuk *clusters* sebanyak K buah.

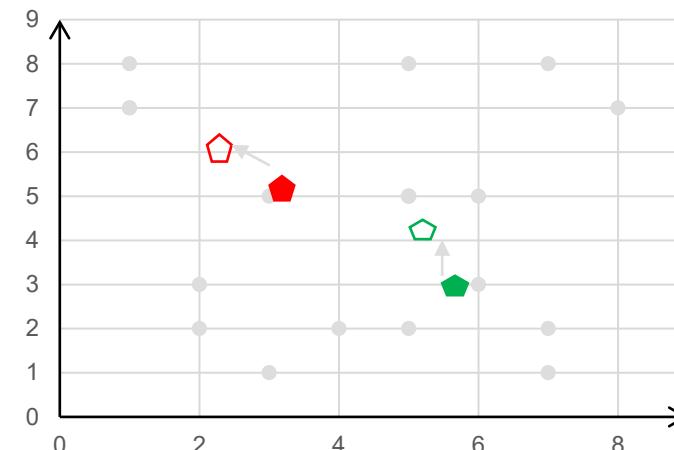
Berdasarkan pengelompokan pada langkah ke dua, maka setiap titik data saat telah tergabung dalam salah satu kluster. Titik data yang diwakili dengan symbol ‘x’ berwarna merah masuk ke kluster 1, dan symbol ‘x’ berwarna biru masuk ke kluster 2, seperti pada berikut:



Langkah-langkah metode K-Means

4. Lakukan kalkulasi, dan tempatkan pusat *centroid* yang baru untuk setiap *cluster*-nya. Langkah ini dilakukan untuk menemukan centroid yang paling tepat untuk masing-masing klaster.

Langkah keempat adalah melakukan penghitungan sesuai algoritma *K-Means*, yaitu mencari posisi titik pusat yang paling sesuai untuk setiap klasternya berdasarkan penghitungan jarak terdekat. Penghitungan jarak masing-masing titik data ke pusat klaster dapat menggunakan metode Euclidean distance, ilustrasi penghitungan jarak dapat dilihat pada Gambar





Langkah-langkah metode K-Means

5. Dari dataset yang kita miliki ambil titik *centroid* terdekat, sehingga dataset tadi menjadi bagian dari *cluster* tersebut. Jika masih ada data yang berubah kelompok (pindah *cluster*), kembali ke langkah 4. Jika tidak, maka *cluster* yang terbentuk (dianggap) sudah baik.

Langkah terakhir dari algoritma K-Means adalah melakukan pengecekan pada titik pusat yang telah ditentukan sebelumnya. Pilih titik pusat terdekat, dan masuk ke dalam kluster tersebut. Jika masih ada perpindahan kluster, kembali ke langkah 4. Algoritma K-Means akan terus mencari titik pusatnya, sampai pembagian datasetnya optimum dan posisi titik pusat tidak berubah lagi



Optimasi K-Means

Dari pembahasan di atas, dapat dianalisa bahwa salah satu faktor krusial baik tidaknya metode ini adalah saat menentukan jumlah klusternya (nilai K). Karena hasil pengelompokan akan menghasilkan analisis yang berbeda untuk jumlah klaster yang berbeda juga.

Jika terlalu sedikit K (misal 2), maka pembagian kluster menjadi cepat, namun mungkin ada informasi tersembunyi yang tidak terungkap.

Jika $K=8$, maka terlalu banyak kluster. Mungkin akan terlalu sulit untuk membuat analisis atau memilih dukungan keputusan dari hasil cluster.



Optimasi K-Means

Untuk mengatasi ini, maka dapat ditambahkan fungsi optimasi yang akan memilih jumlah awal kluster secara tepat. kita gunakan di sesi latihan dan sebuah metode *elbow* yang akan membantu kita untuk memilih nilai K yang tepat dengan menggunakan *metric WCSS (Within Cluster Sum of Squares)*, contoh penghitungan untuk tiga klaster:

wcss

$$= \sum_{P_i \text{ in cluster 1}} \text{jarak}(P_i' C_1)^2 + \sum_{P_i \text{ in cluster 2}} \text{jarak}(P_i' C_2)^2 +$$



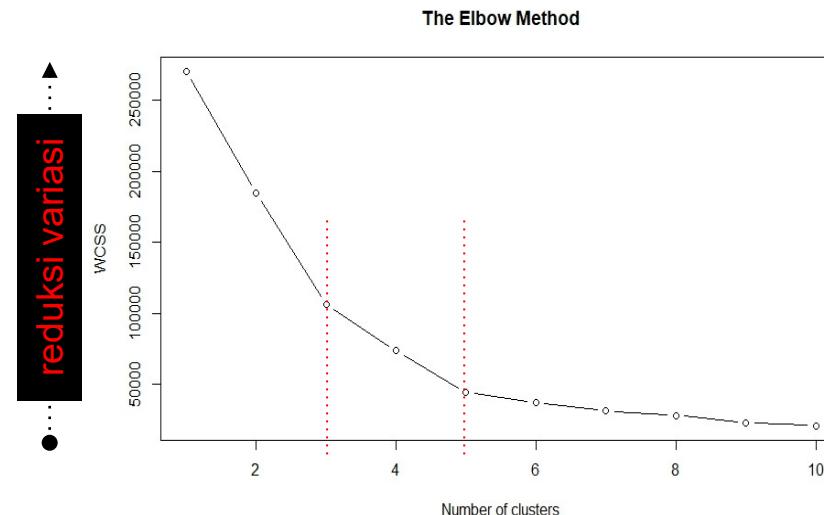
Optimasi K-Means

Pada metric di atas dapat diamati bahwa WCSS sebagai variabel dependennya. Kemudian ada simbol Sigma (seperti E), yang menyatakan jumlah kuadrat dari jarak tiap titik P_i yang ada pada kluster 1. *Sum of Squares* (jumlah kuadrat) adalah menjumlahkan hasil kuadrat dari masing-masing jarak. Selanjutnya hasil penjumlahan kluster 1 ditambah dengan hasil kuadrat jarak untuk tiap data poin terhadap titik pusat kluster dua, dan seterusnya sesuai jumlah kluster yang kita inginkan.

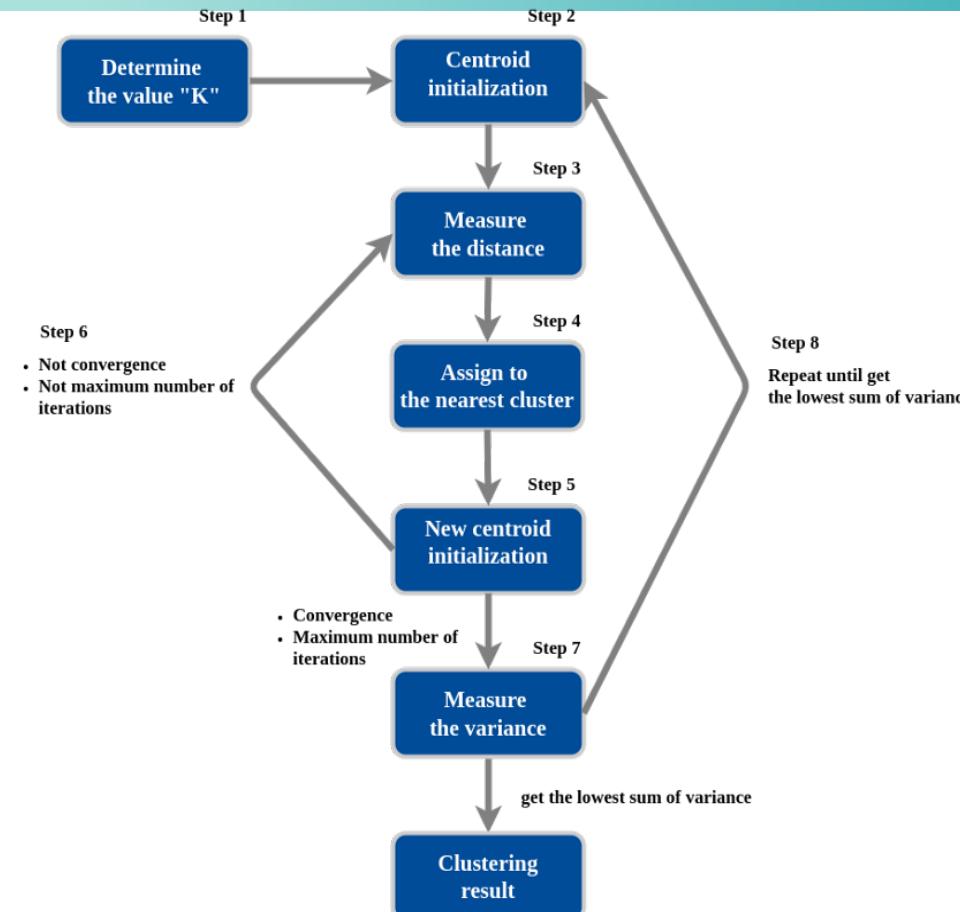
Untuk mengetahui jumlah klaster yang paling baik untuk studi kasus yang diuji coba adalah dengan cara melihat perbandingan WCSS untuk 2 kluster, 3 kluster, 4 dan seterusnya. Yang kita pilih adalah ketika perubahan nilai WCSS nya sangat signifikan, seperti sebuah siku (elbow). Oleh karena itu cara pemilihan ini disebut dengan **elbow plot method**.

Optimasi K-Means

Grafik perhitungan WCSS untuk sebuah contoh dataset. Semakin kecil skor WCSS, semakin baik. Sumbu x adalah jumlah kluster, sumbu y adalah skor WCSS. Bisa dilihat bahwa saat K=1, nilai WCSS sangat tinggi. Kemudian menurun terus sampai K=5 terlihat membentuk seperti sebuah siku. Mulai K=6 sampai K=10 penurunan skor WCSS sudah tidak signifikan. Dengan demikian, dapat diketahui bahwa jumlah kluster yang tepat untuk grafik di atas adalah 5. Contoh hasil perhitungan WCSS dapat dilihat pada grafik berikut:



Alur KMeans





Studi Kasus K-Means

Pada studi kasus ini, kita akan mengaplikasikan metode K-Means ini untuk sebuah permasalahan nyata. Misalnya, seorang *data scientist* diminta untuk menganalisis data pelanggan toko. Data tersebut adalah data *member* atau pelanggan dimana hasil yang diinginkan sebuah analisa kecenderungan pembelian dari suatu kelompok pelanggan sehingga dapat memperkuat hubungan mereka terhadap konsumen. Misal untuk penguatan marketing, strategi penawaran yang tepat, dan sebagainya.



Studi Kasus **K**Means

Hands-On

Mengimpor library

```
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd
```

Mengimpor dataset

```
dataset = pd.read_csv('Pengunjung_mall.csv')
X = dataset.iloc[:, [3, 4]].values
```

| IDPelanggan | Kelamin | Usia | Pendapatan (juta Rp) | Rating_pengeluaran (1-100) |
|-------------|---------|-----------|----------------------|----------------------------|
| 0 | 1 | Laki | 19 | 15 |
| 1 | 2 | Laki | 21 | 15 |
| 2 | 3 | Perempuan | 20 | 16 |
| 3 | 4 | Perempuan | 23 | 16 |
| 4 | 5 | Perempuan | 31 | 17 |
| ... | ... | ... | ... | ... |
| 195 | 196 | Perempuan | 35 | 120 |
| 196 | 197 | Perempuan | 45 | 126 |
| 197 | 198 | Laki | 32 | 126 |
| 198 | 199 | Laki | 32 | 137 |
| 199 | 200 | Laki | 30 | 137 |

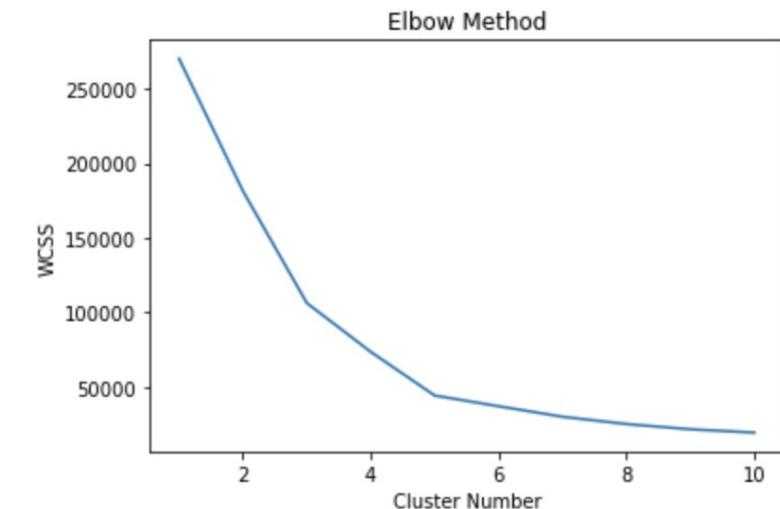


Studi Kasus K-Means

Hands-On

Optimasi K-Means dengan metode elbow untuk menentukan jumlah klaster yang tepat

```
from sklearn.cluster import KMeans
wcss = []
for i in range(1, 11):
    kmeans = KMeans(n_clusters = i, init =
'k-means++', random_state = 42)
    kmeans.fit(X)
    wcss.append(kmeans.inertia_)
plt.plot(range(1, 11), wcss)
plt.title('Elbow Method')
plt.xlabel('Cluster Number')
plt.ylabel('WCSS')
plt.show()
```





Studi Kasus K-Means

Hands-On

Proses K-Means Clustering (5 Cluster)

```
kmeans = KMeans(n_clusters = 5, init = 'k-means++',  
random_state = 42)  
y_kmeans = kmeans.fit_predict(X)
```

kmeans

```
KMeans(n_clusters=5, random_state=42)
```

y_kmeans

```
array([3, 0, 3, 0, 3, 0, 3, 0, 3, 0, 3, 0, 3, 0, 3, 0, 3, 0, 3, 0, 3, 0,  
      3, 0, 3, 0, 3, 0, 3, 0, 3, 0, 3, 0, 3, 0, 3, 0, 3, 0, 3, 0, 3, 0, 3, 0,  
      3, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,  
      1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,  
      1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,  
      1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,  
      1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,  
      1, 2, 4, 2, 4, 2, 4, 2, 4, 2, 1, 2, 4, 2, 4, 2, 4, 2, 4, 2, 4, 2, 4, 2, 4, 2,  
      4, 2, 4, 2, 4, 2, 4, 2, 4, 2, 4, 2, 4, 2, 4, 2, 4, 2, 4, 2, 4, 2, 4, 2, 4, 2,  
      4, 2, 4, 2, 4, 2, 4, 2, 4, 2, 4, 2, 4, 2, 4, 2, 4, 2, 4, 2, 4, 2, 4, 2, 4, 2,  
      4, 2], dtype=int32)
```

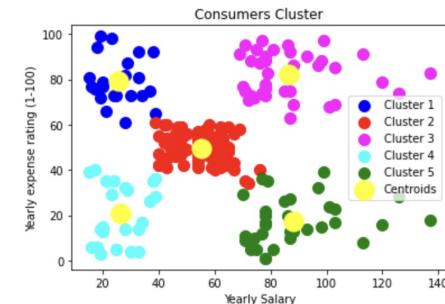


Studi Kasus **K**Means

Hands-On

Visualisasi hasil clusters

```
plt.scatter(X[y_kmeans == 0, 0], X[y_kmeans == 0, 1], s = 100, c = 'blue', label =  
'Cluster 1')  
plt.scatter(X[y_kmeans == 1, 0], X[y_kmeans == 1, 1], s = 100, c = 'red', label =  
'Cluster 2')  
plt.scatter(X[y_kmeans == 2, 0], X[y_kmeans == 2, 1], s = 100, c = 'magenta', label =  
'Cluster 3')  
plt.scatter(X[y_kmeans == 3, 0], X[y_kmeans == 3, 1], s = 100, c = 'cyan', label =  
'Cluster 4')  
plt.scatter(X[y_kmeans == 4, 0], X[y_kmeans == 4, 1], s = 100, c = 'green', label =  
'Cluster 5')  
plt.scatter(kmeans.cluster_centers_[:, 0], kmeans.cluster_centers_[:, 1], s = 300, c  
= 'yellow', label = 'Centroids')  
plt.title('Consumers Cluster')  
plt.xlabel('Yearly Salary')  
plt.ylabel('Yearly expense rating (1-100)')  
plt.legend()
```





Studi Kasus K-Means

Hands-On

Proses K-Means Clustering (3 Cluster)

```
kmeans = KMeans(n_clusters = 3, init = 'k-means++',  
random_state = 42)  
y_kmeans = kmeans.fit_predict(X)
```

kmeans

```
KMeans(n_clusters=3, random_state=42)
```

y_kmeans

```
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, 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, 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, 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, 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, 2, 1, 2, 1, 2, 1, 2, 1, 2, 1, 2, 1, 2, 1, 2, 1, 2, 1, 2, 1, 2,  
1, 2, 1, 2, 1, 2, 1, 2, 1, 2, 1, 2, 1, 2, 1, 2, 1, 2, 1, 2, 1, 2,  
1, 2, 1, 2, 1, 2, 1, 2, 1, 2, 1, 2, 1, 2, 1, 2, 1, 2, 1, 2, 1, 2,  
1, 2], dtype=int32)
```

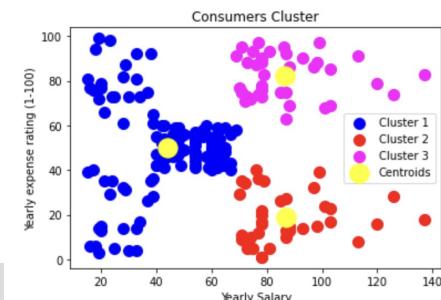


Studi Kasus KMeans

Hands-On

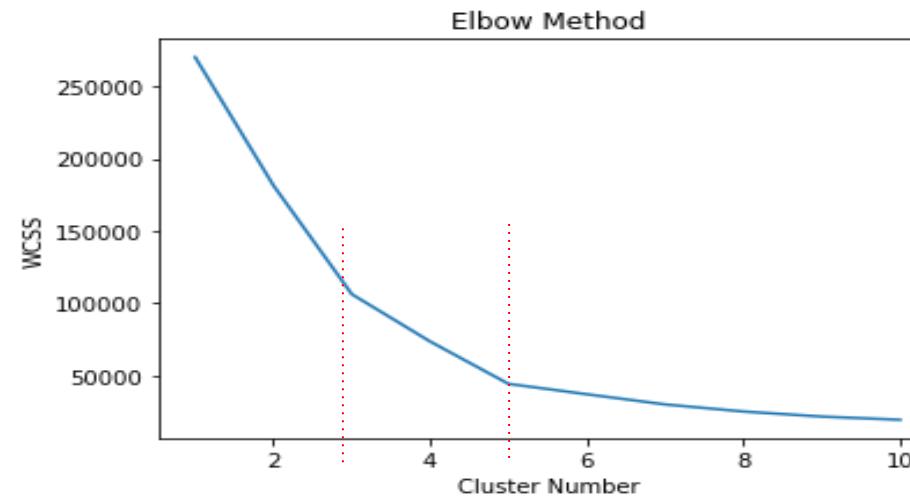
Visualisasi hasil clusters

```
plt.scatter(X[y_kmeans == 0, 0], X[y_kmeans == 0, 1], s = 100, c = 'blue', label =  
'Cluster 1')  
plt.scatter(X[y_kmeans == 1, 0], X[y_kmeans == 1, 1], s = 100, c = 'red', label =  
'Cluster 2')  
plt.scatter(X[y_kmeans == 2, 0], X[y_kmeans == 2, 1], s = 100, c = 'magenta', label =  
'Cluster 3')  
plt.scatter(kmeans.cluster_centers_[:, 0], kmeans.cluster_centers_[:, 1], s = 300, c  
= 'yellow', label = 'Centroids')  
plt.title('Consumers Cluster')  
plt.xlabel('Yearly Salary')  
plt.ylabel('Yearly expense rating (1-100)')  
plt.legend()
```



Studi Kasus K-Means

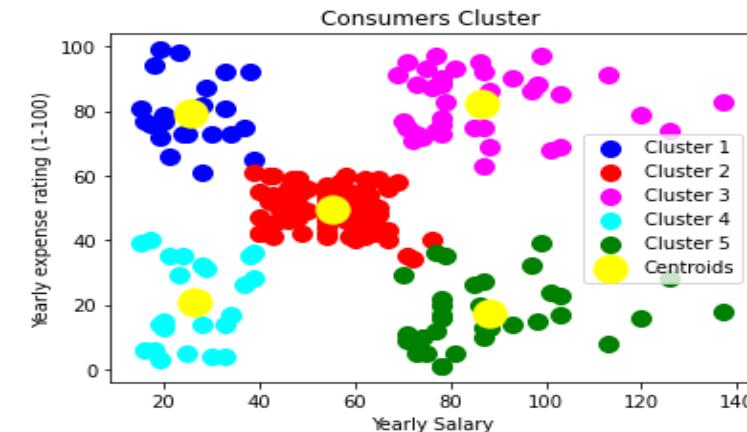
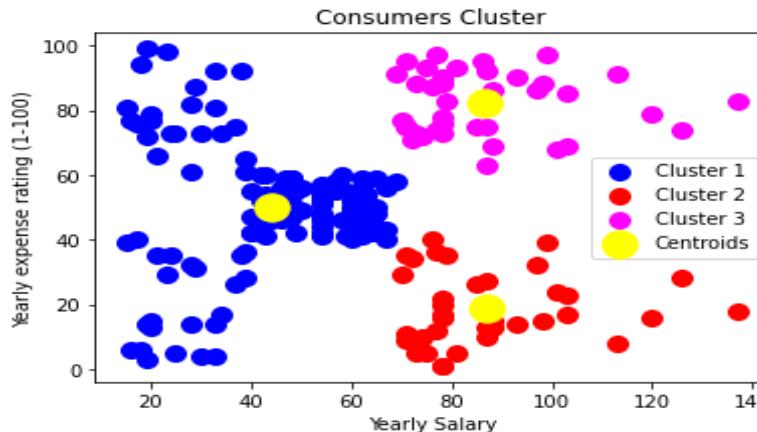
Hasil pengelompokan data menggunakan metode K-Means yang dioptimasi dengan metrics WCSS:



Pada Gambar di atas, dapat diamati bahwa garis grafik yang berbentuk siku terdapat pada klaster 3 dan 5, maka dapat disimpulkan bahwa pembagian klaster sebanyak 3 atau 5 adalah jumlah yang paling **optimum**

Studi Kasus K-Means

Hasil pengelompokan data menggunakan metode K-Means yang dioptimasi dengan metrics WCSS:

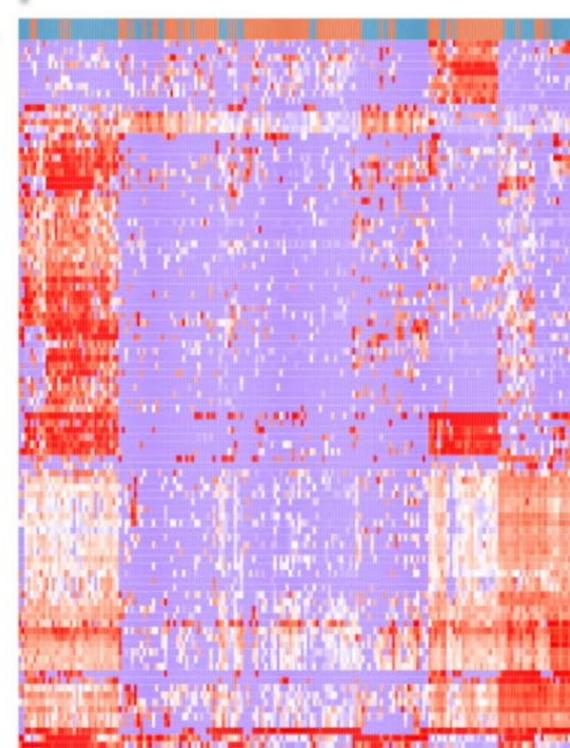


Pada dua gambar di atas dapat diamati hasil pengelompokan data set yang terbagi dalam 3 dan 5 klaster. Hasil dari pengelompokan tersebut dapat digunakan sebagai dukungan keputusan untuk menentukan strategi yang dituju oleh pemilik toko terhadap pelanggannya

Hierarchical Clustering (HC): sering diasosiasikan dengan *Heatmaps*

baris
merepresentasikan
pengukuran untuk
gen berbeda

kolom merepresentasikan
sampel berbeda

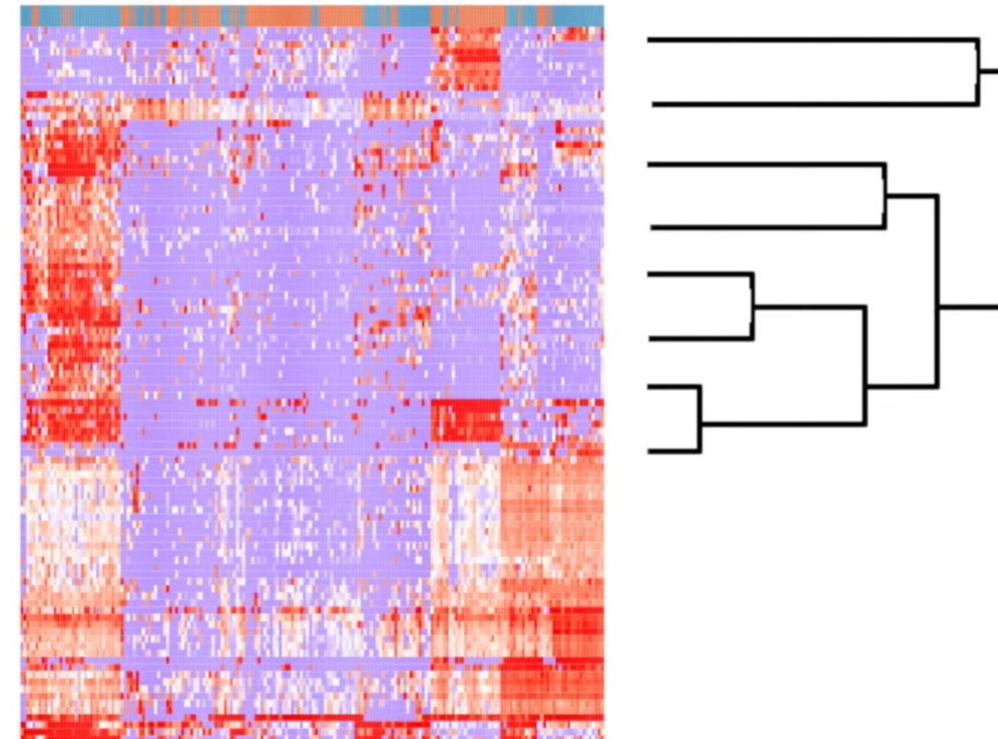


HC mengurutkan baris dan/atau kolom berdasarkan similaritas.

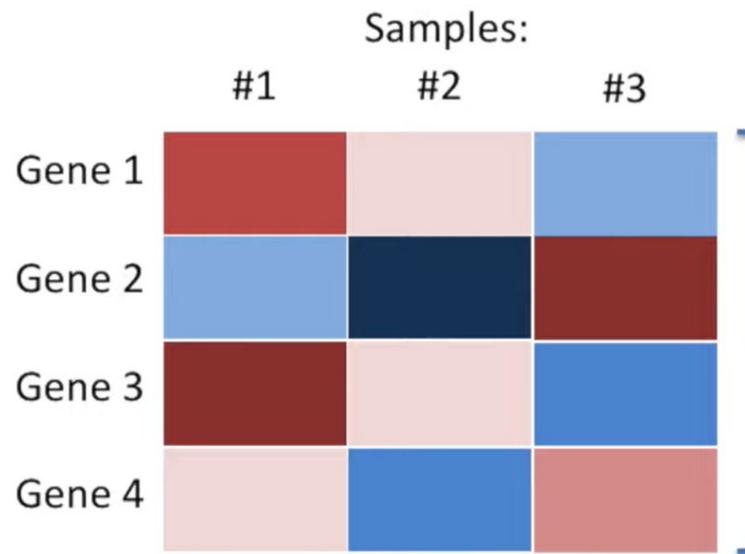
Hal ini mempermudah utk melihat korelasi dalam data



Hierarchical Clustering (HC):
sering diasosiasikan dengan
Heatmaps, dan heatmaps
biasanya berpasangan dengan
dendogram



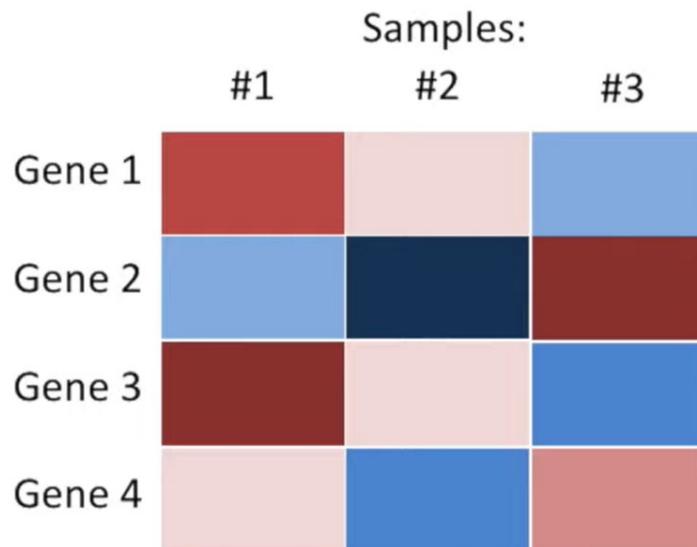
Ilustrasi (HC):



utk ilustrasi ini, kita hanya akan men-cluster-kan (mengurut ulang/reorder) baris (gen = gene)

Scr konsep:
1) Cari gen yg paling similar dgn gene#1

Ilustrasi (HC):



gen yg ada dalam gene #1 dan #2 berbeda (dari warna di tiap sampelnya)

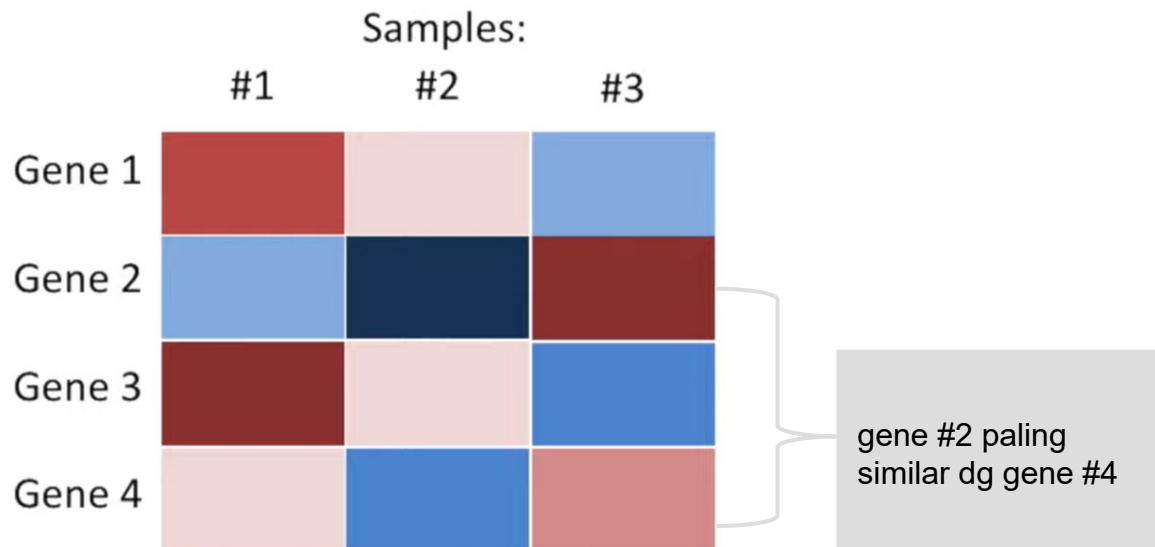
gene #1 dan #3 masing sampelnya similar

gene #1 dan #4 masing sampelnya similar

Scr konsep:
1) Cari gen yg paling similar dgn gene#1

gene #1 dan #3 lebih similar vs gene #1 dan #4

Ilustrasi (HC):



Scr konsep:

- 1) Cari gen yg paling similar dgn gene#1
- 2) Cari gen yg paling similar dengan gene#2**
- 3) lakukan pencarian yg sama utk gene#3
- 4) lakukan pencarian yg sama utk gene#3

Ilustrasi (HC):

Samples:

| | #1 | #2 | #3 |
|--------|------------|-----------|----------|
| Gene 1 | Red | Pink | Blue |
| Gene 2 | Light Blue | Dark Blue | Dark Red |
| Gene 3 | Dark Brown | Pink | Blue |
| Gene 4 | Pink | Blue | Red |

gene #1 dan #3
adalah kombinasi
yang paling similar vs
kombinasi lainnya

Samples:

#1 #2 #3

(gene#1 dan #3) Cluster #1

Gene 2

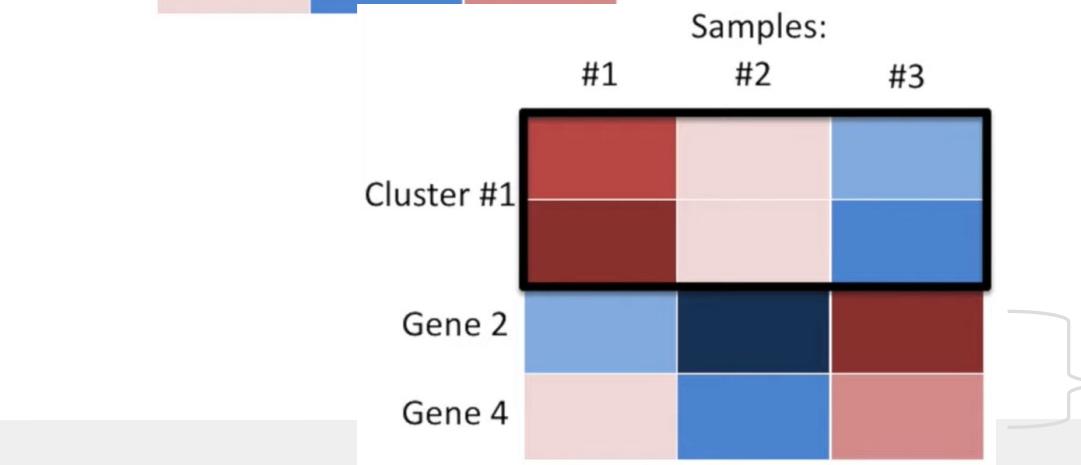
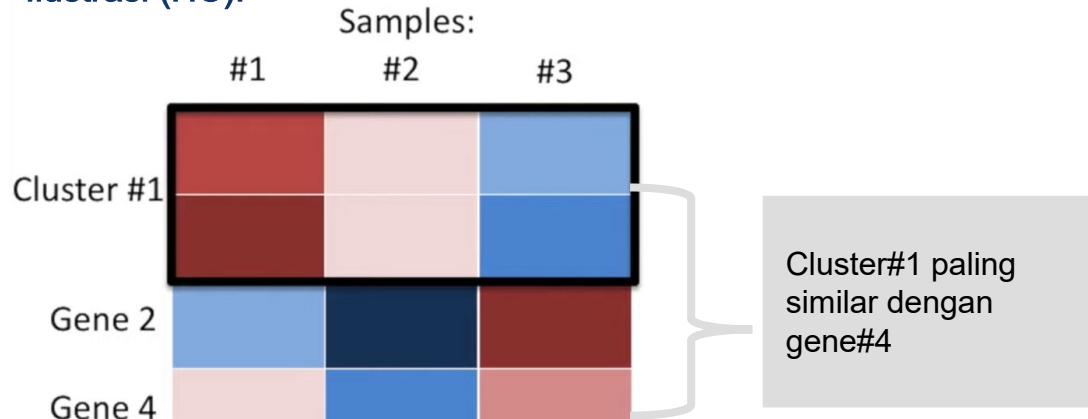
Gene 4



Scr konsep:

- 1) Cari gen yg paling similar dgn gene#1
- 2) Cari gen yg paling similar dengan gene#2
- 3) lakukan pencarian yg sama utk gene#3
- 4) lakukan pencarian yg sama utk gene#3
- 5) dengan kombinasi yg berbeda,
cari dua gene yang paling similar.
merge/gabung keduanya menjadi
satu cluster
- 6) kembali ke langkah 1), tapi
perlakukan clustuer#1 sbg satu gene

Ilustrasi (HC):

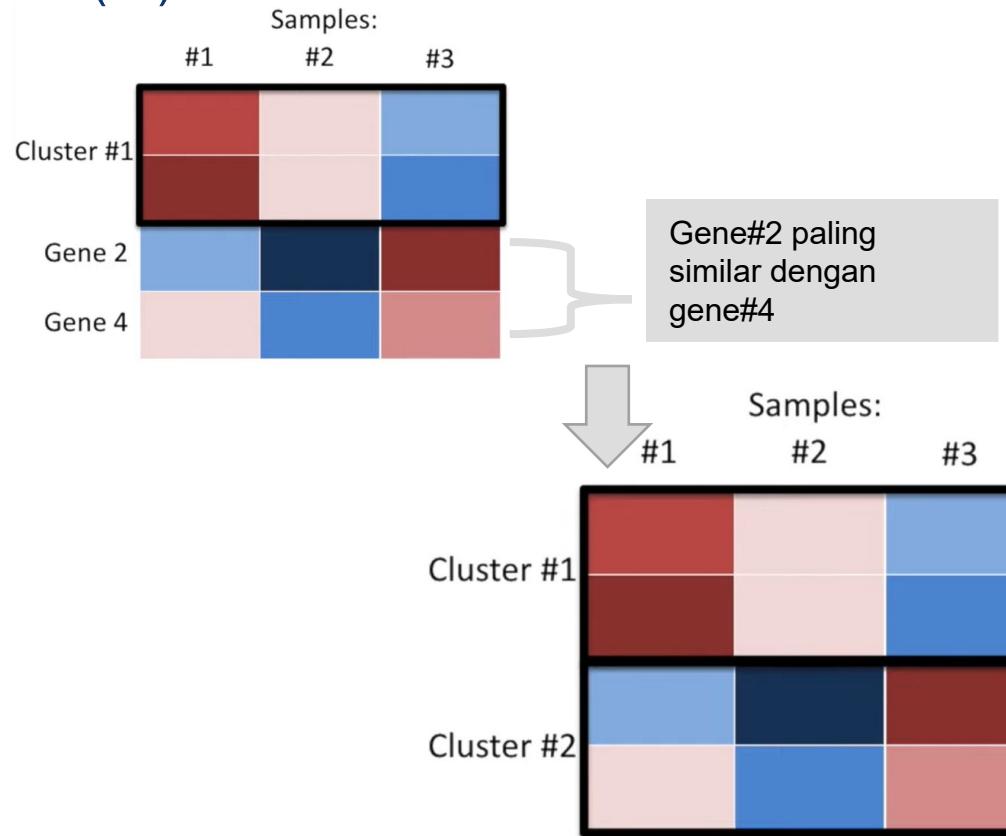


Scr konsep:

- 1) Cari gen yg paling similar dgn Cluster#1
- 2) Cari gen yg paling similar dgn gene#2 (dan lakukan yg sama dengan gene#4)

Gene#2 paling similar dgn gene#4, namun perhatikan bhw yg sdg kita bandingkan adlh gene#2 dgn Cluster#1

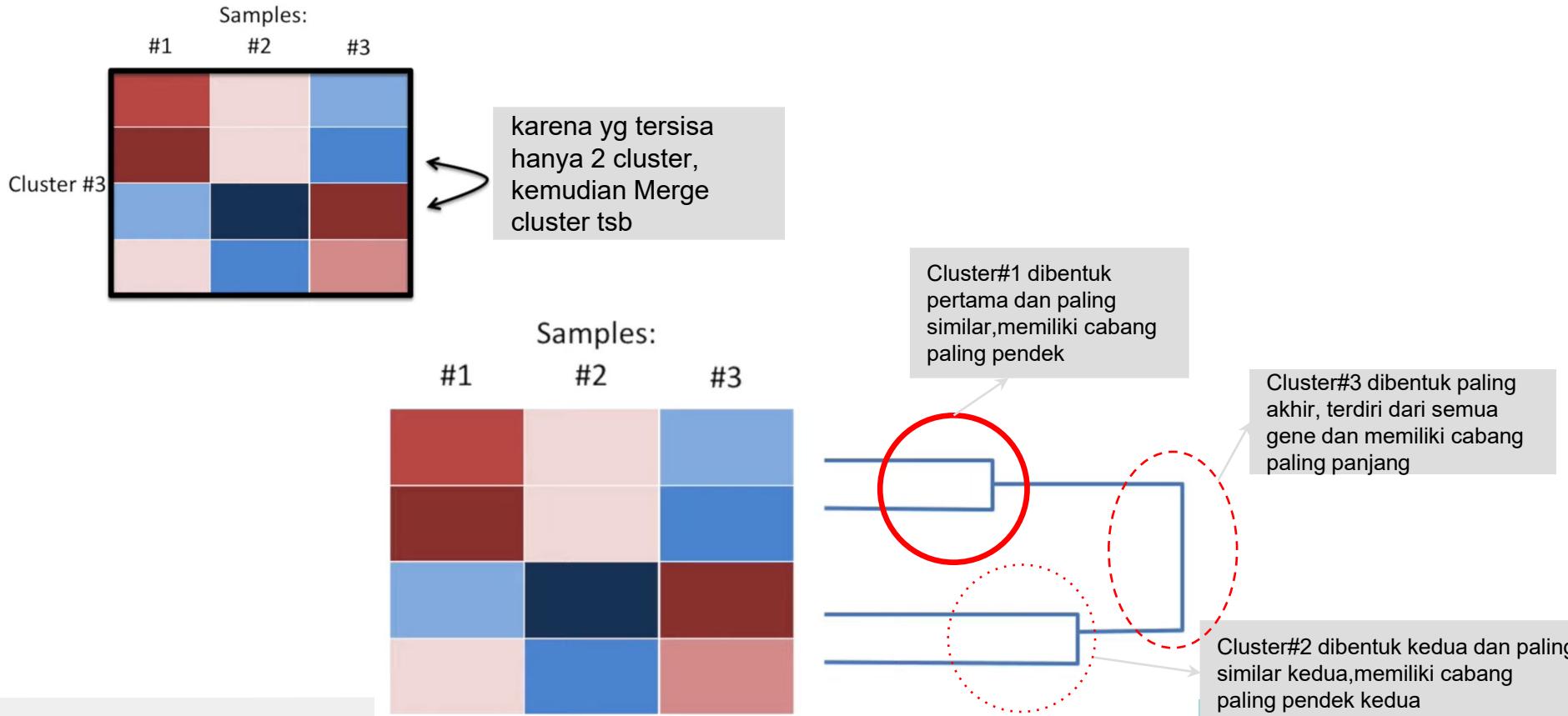
Ilustrasi (HC):



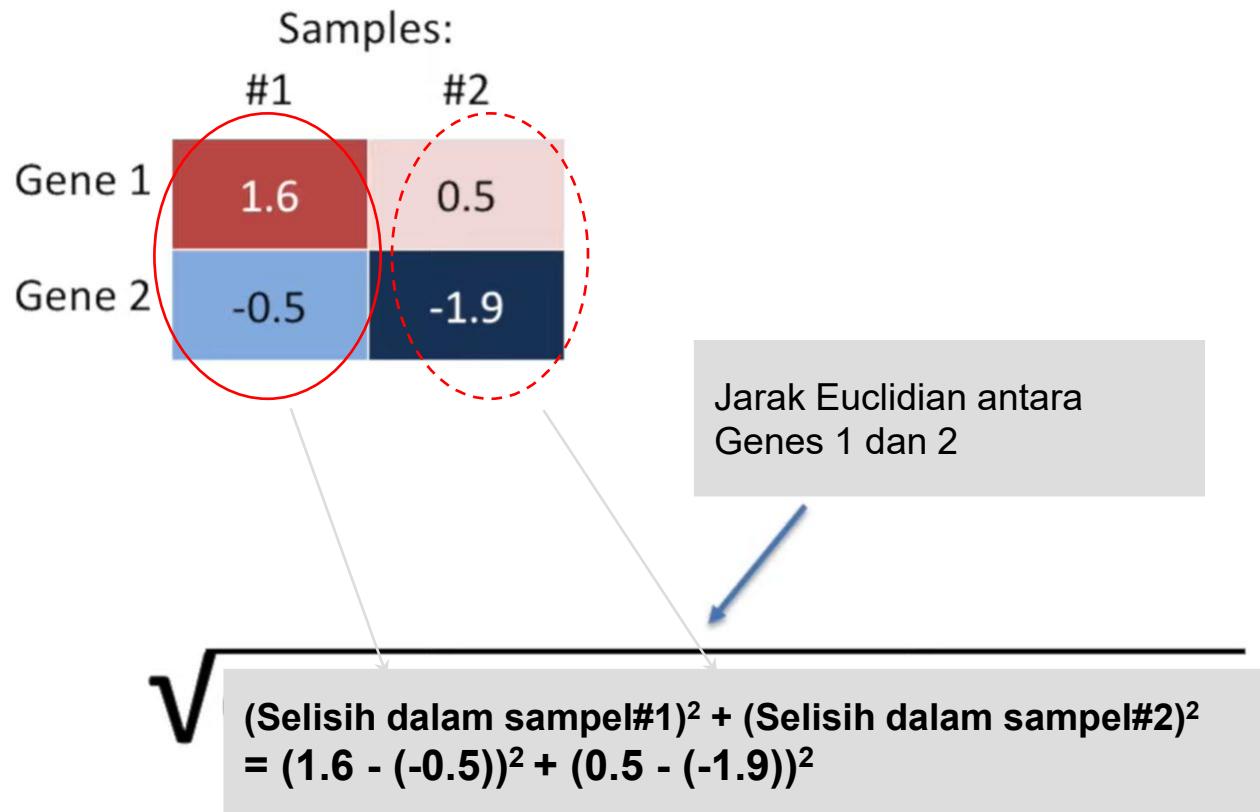
Scr konsep:

- 1) Cari gen yg paling similar dgn Cluster#1
- 2) Cari gen yg paling similar dgn gene#2 (dan lakukan yg sama dengan gene#4)
- 3) Dgn kombinasi yg berbeda, cari dua gene yg paling similar.
Merge/gabung menjadi satu cluster
- 4) Kembali ke langkah 1)

Ilustrasi (HC):



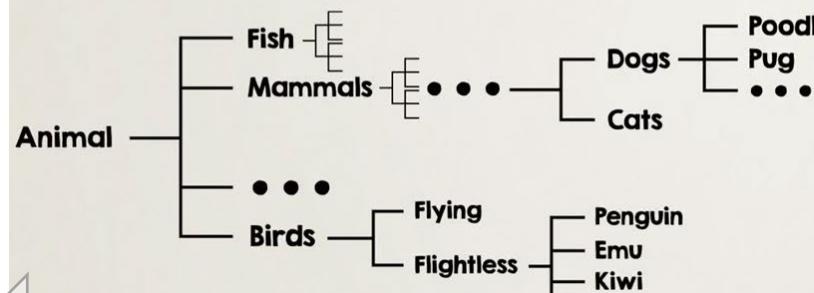
Ilustrasi (HC):



Hierarchical Clustering (HC)

Pengelompokan hierarki adalah Teknik clustering dengan memisahkan data ke dalam kelompok berdasarkan beberapa ukuran kesamaan, menemukan cara untuk mengukur bagaimana mereka sama dan berbeda, dan selanjutnya mempersempit data.

AGGLOMERATIVE (BOTTOM-UP)



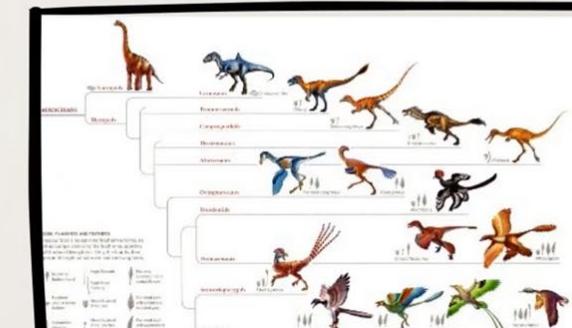
induksi: dari banyak cluster ke sedikit cluster

<https://www.youtube.com/watch?v=ijUMKMC4f9I>

Divisive (pembagian)

Pengelompokan divisif dikenal sebagai pendekatan top-down, yaitu mengambil cluster besar dan mulai membagiannya menjadi dua, tiga, empat, atau lebih cluster

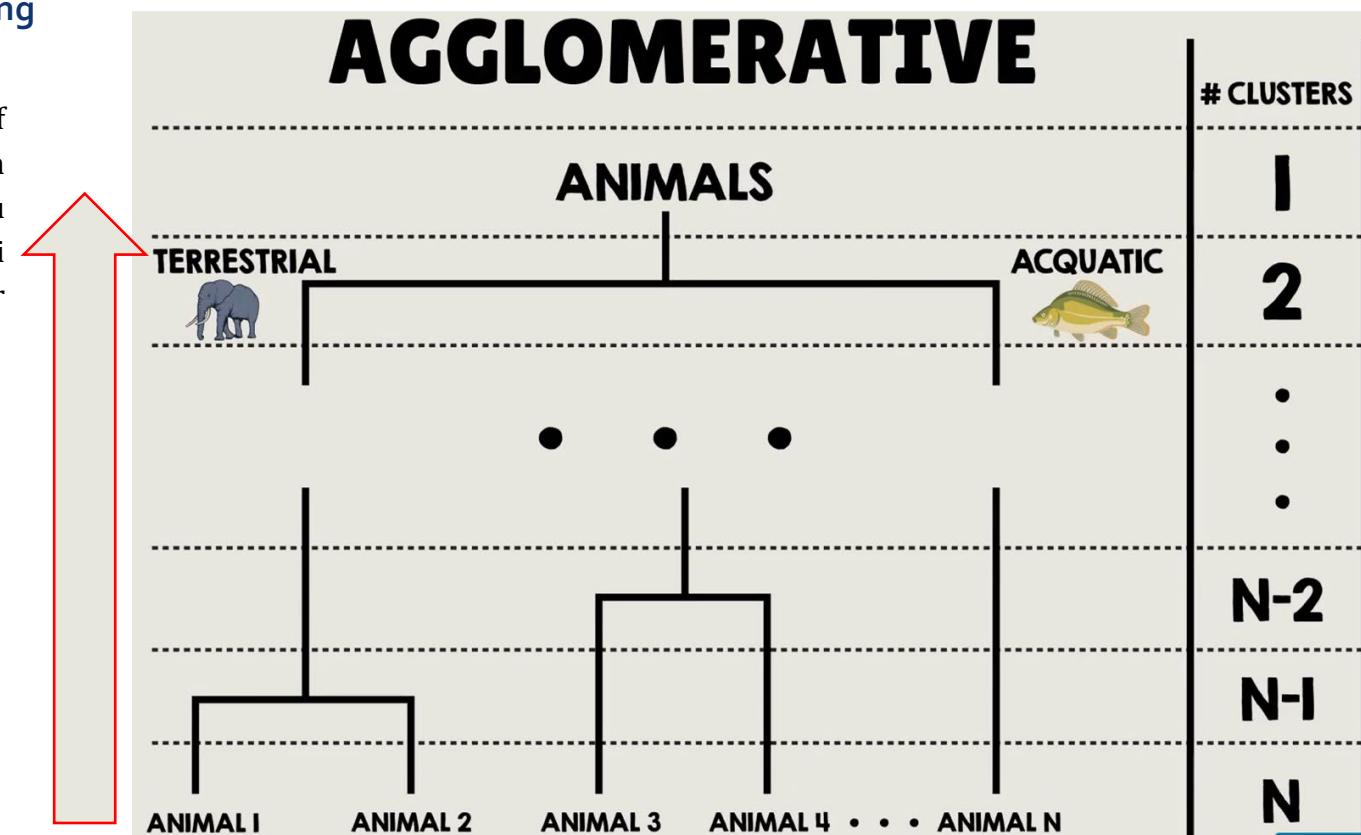
DIVISIVE (TOP-DOWN)



deduksi: dari satu cluster ke banyak cluster

Types of Hierarchical Clustering

Agglomerative (penggabungan)
Pengelompokan agglomeratif dikenal sebagai pendekatan bottom-up, yaitu pengelompokan dimulai dari cluster kecil menuju satu cluster besar.





Hierarchical Clustering

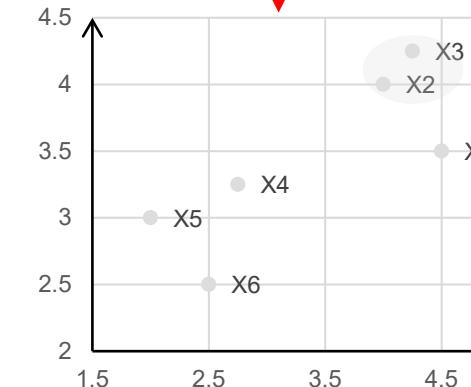
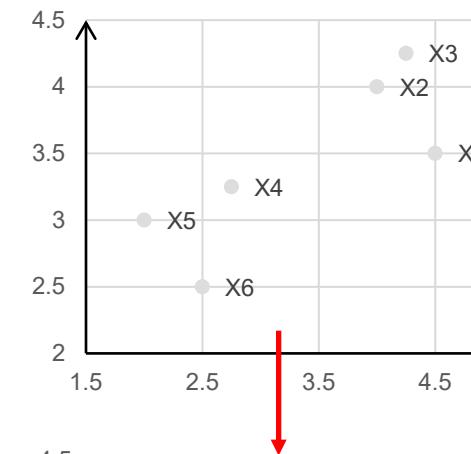
Langkah-langkah metode hierarchical clustering dengan **agglomerative**:

1. Buat setiap data poin dalam dataset menjadi sebuah cluster, sehingga untuk N data kita memiliki N cluster. Misalnya jika jumlah row data adalah 500 maka akan terdapat 500 cluster.
2. Cari dua poin/2 cluster yang saling berdekatan untuk digabung menjadi satu cluster sehingga jumlah cluster menjadi lebih kecil.
3. Cari 2 cluster lagi yang berdekatan dengan yang lain (termasuk dengan kluster yang baru saja dibuat di langkah 2 jika memang cluster tersebut memiliki jarak terdekat dengan kluster lain), dan jadikan dua cluster terdekat ini menjadi 1 kluster. Dengan demikian, sekarang kita memiliki N-2 kluster.
4. Langkah ketiga akan diulang terus hingga mendapatkan satu buah cluster besar.

Hierarchical Clustering

Ilustrasi dari langkah-langkah **agglomerative**:

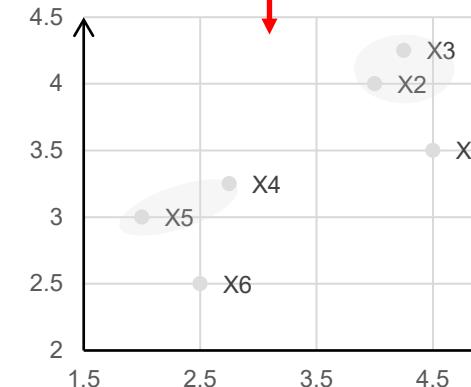
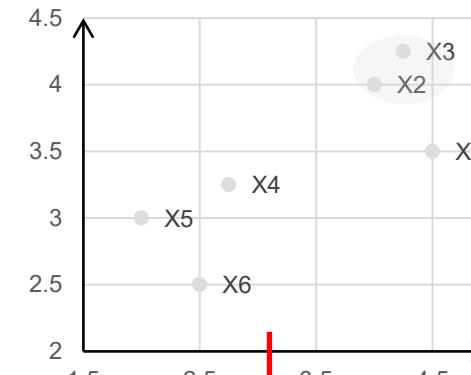
Misalnya terdapat enam data poin. Pada langkah pertama sudah jelas bahwa N data = N cluster. Kita akan mendefinisikan jarak antara 2 kluster sebagai jarak euclidean terdekatnya. Ilustrasinya sebagai berikut:





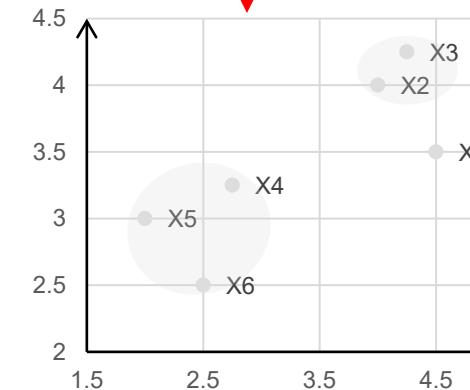
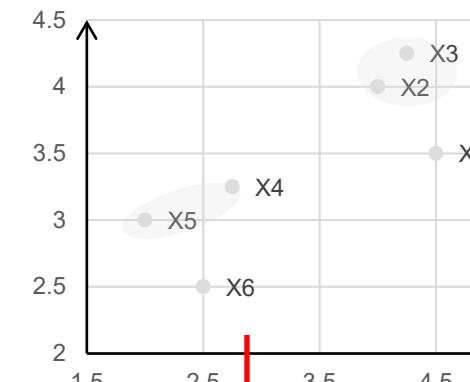
Hierarchical Clustering

Proses menggabungkan dua cluster menjadi dapat masih dilanjutkan jika masih terdapat titik yang terdekat lagi yang memungkinkan untuk digabungkan



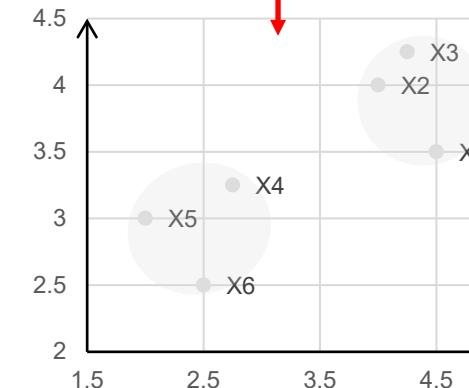
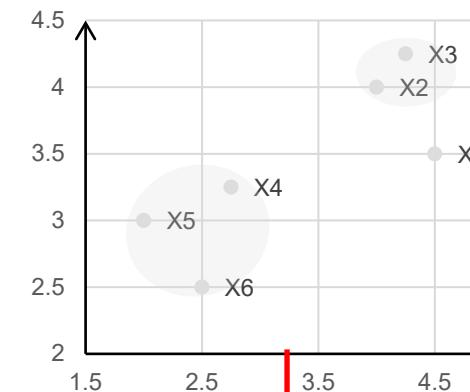
Hierarchical Clustering

Selanjutnya, seperti langkah sebelumnya, kita cari dua cluster terdekat lagi untuk digabungkan. Cluster yang digabungkan boleh cluster yang berupa satu titik pada langkah pertama atau cluster yang merupakan gabungan dari dua titik/cluster.



Hierarchical Clustering

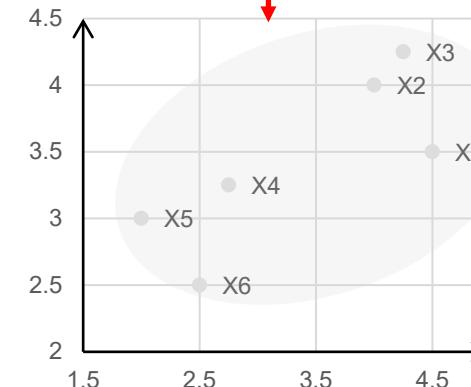
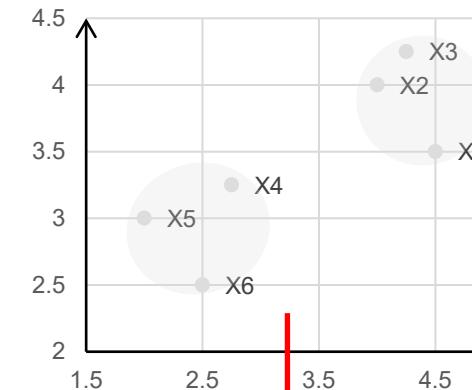
Menggabungkan lagi dua cluster yang terdekat menjadi satu cluster. Jumlah titik dalam cluster tidak harus ditentukan sama, dapat saja cluster yang satu jumlah titiknya lebih banyak dibandingkan cluster yang lain.





Hierarchical Clustering

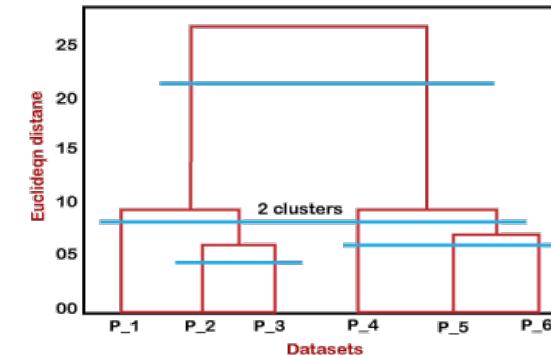
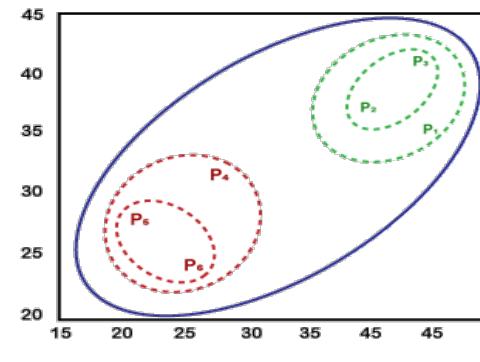
Proses akan berakhir Ketika semua cluster telah tergabung menjadi satu cluster besar.



Optimasi Hierarchical Clustering

Pada clustering K-Means yang kita pelajari sebelumnya, untuk mengetahui jumlah cluster yang tepat kita dapat menggunakan metode Elbow, pada hierarchical clustering kita dapat menggunakan Dendogram.

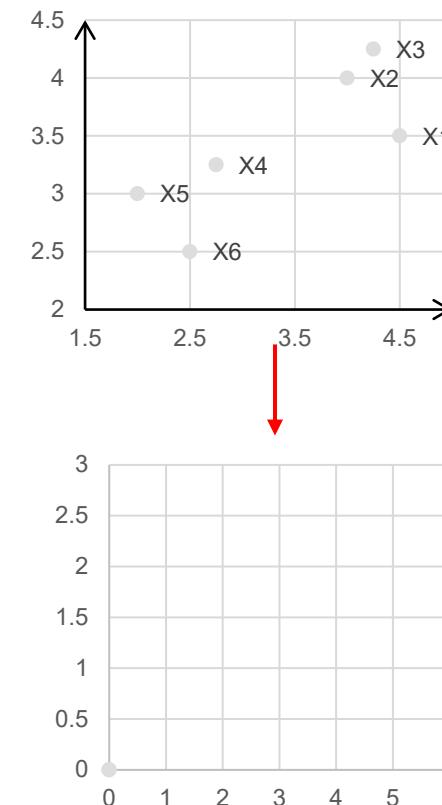
Dendrogram adalah sebuah grafik (diagram) yang menunjukkan proses penggabungan kluster. Di sumbu x dari sebuah dendrogram kita memiliki data kluster, sementara di sumbu y adalah jarak euclideananya.



Optimasi Hierarchical Clustering

Ilustrasi dari penentuan dendogram adalah sebagai berikut:

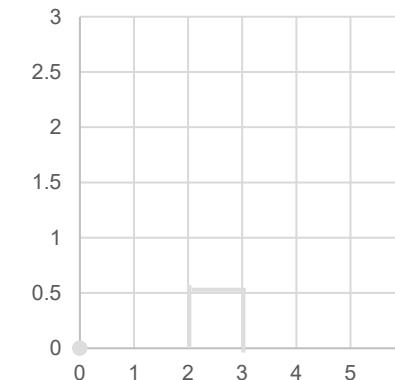
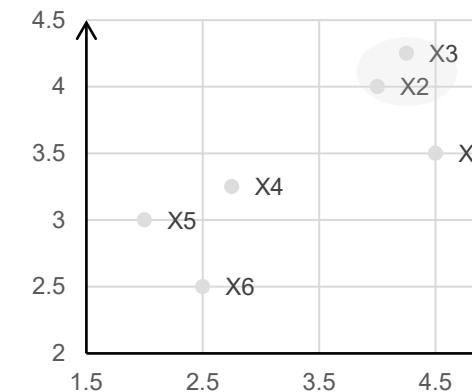
1. Sama seperti pada contoh hierarchical clustering sebelumnya, terdapat enam titik dalam satu diagram. Grafik yang atas adalah grafik awal dan yang bawah adalah grafik dendogram.





Optimasi Hierarchical Clustering

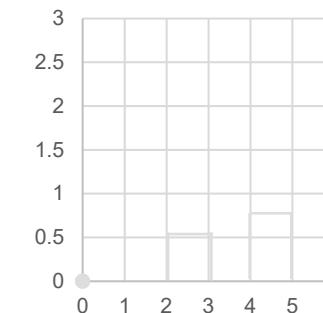
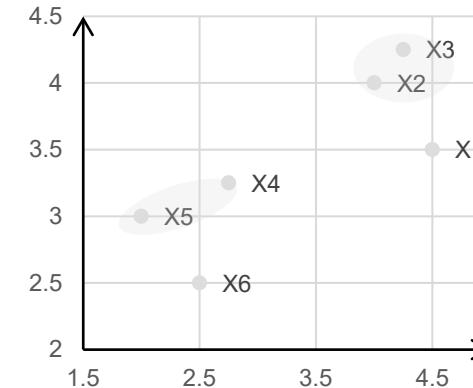
2. Sama seperti ilustrasi pada hierarchical clustering, langkah pertama adalah menentukan dua titik terdekat kemudian menerjemahkannya ke dalam diagram dendrogram





Optimasi Hierarchical Clustering

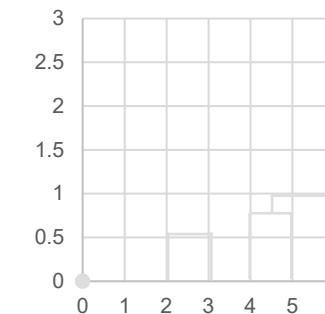
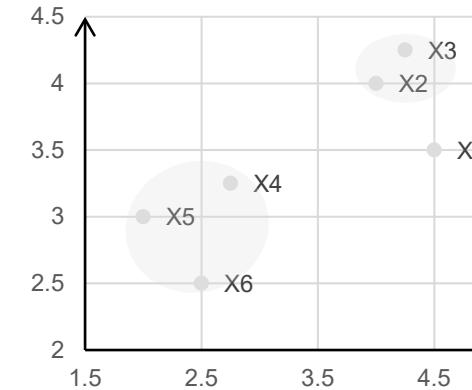
3. Mencari lagi dua cluster yang berdekatan untuk digabungkan menjadi satu cluster lagi. Tinggi diagram Dendogram berbeda-beda sesuai dengan hasil penghitungan Euclidean Distance-nya.





Optimasi Hierarchical Clustering

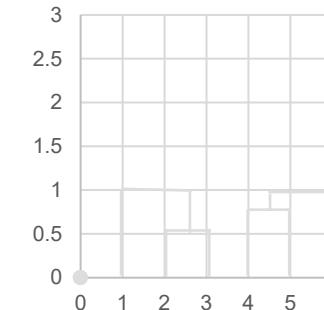
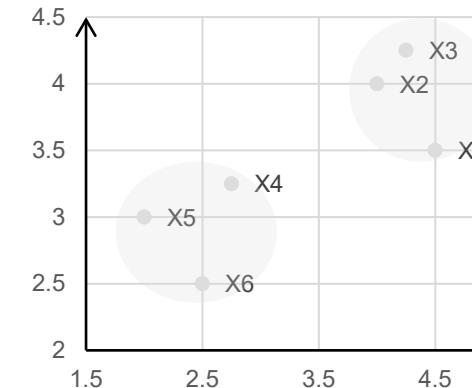
4. Proses yang ketiga diulang lagi dengan mencari dua cluster yang terdekat. Jika dua cluster yang digabungkan sebelumnya adalah cluster antara dua titik maka cara menerjemahkan dalam dendogram dapat diamati pada gambar disamping,





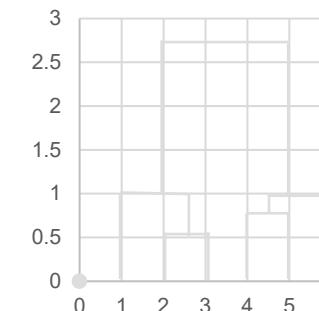
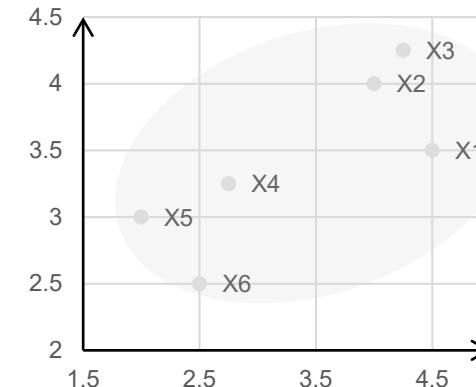
Optimasi Hierarchical Clustering

5. Mengulang proses dengan menggabungkan cluster yang sudah ada, Pada gambar disamping tampak bahwa dua cluster terakhir merupakan gabungan dari cluster (dua titik yang menjadi satu cluster) pada proses awal.



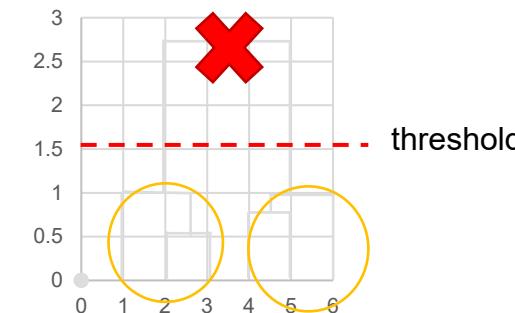
Optimasi Hierarchical Clustering

6. Proses akan berhenti setelah semua cluster telah tergabung menjadi satu cluster besar seperti pada gambar di samping:



Optimasi Hierarchical Clustering

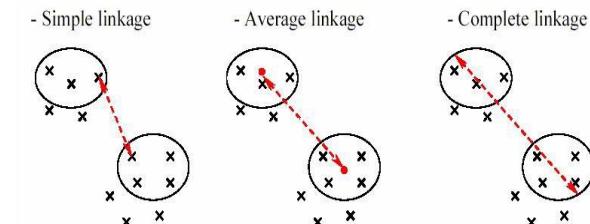
7. Untuk menentukan berapa jumlah cluster yang paling sesuai pada data set yang diujikan dapat dianalisis melalui dendogram. Yaitu dengan menentukan garis grafik dendogram yang paling panjang yang tidak terkena potongan atau bisa juga dengan menentukan nilai threshold, seperti pada gambar di samping:



Kesimpulan Hierarchical Clustering

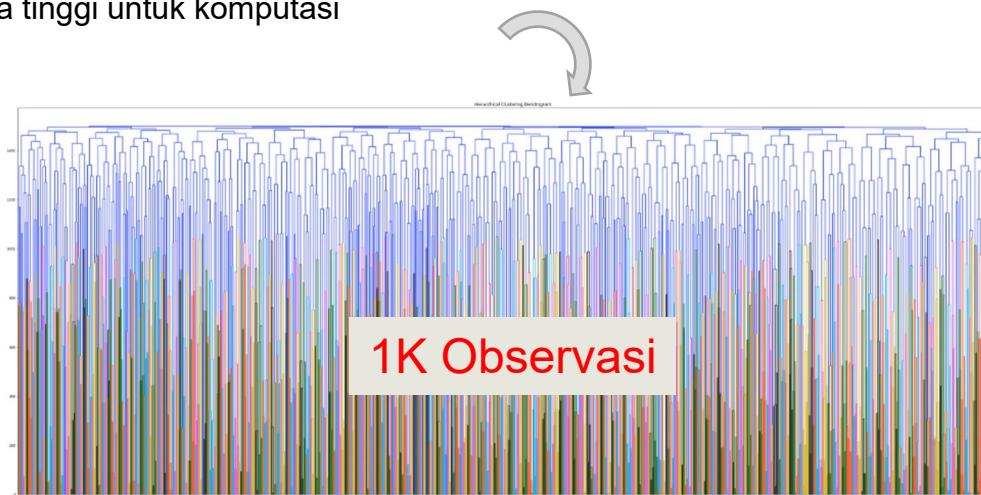
Pro:

- HC menunjukkan semua kemungkinan hubungan(linkages) antar cluster
- Mempermudah pemahaman data lebih baik
- Tidak membutuhkan penentuan jumlah cluster (dibanding K-Means)
- Banyak metode yang mengimplementasikan HC (misal metode Ward)



Kontra:

- Skalabilitas untuk data berukuran besar
- Membutuhkan biaya tinggi untuk komputasi





Studi Kasus Hierarchical Clustering

Pada studi kasus ini, kita akan mengaplikasikan metode Hierarchical Clustering ini untuk sebuah permasalahan nyata. Misalnya, seorang *data scientist* diminta untuk menganalisis data pelanggan toko. Data tersebut adalah data *member* atau pelanggan dimana hasil yang diinginkan sebuah analisa kecenderungan pembelian dari suatu kelompok pelanggan sehingga dapat memperkuat hubungan mereka terhadap konsumen. Misal untuk penguatan marketing, strategi penawaran yang tepat, barang-barang apa saja yang cocok bagi mereka, dll.



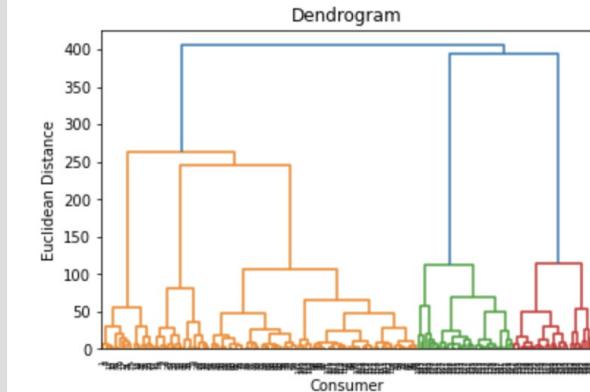
Studi Kasus Hierarchical Clustering

Hands-on

```
# Mengimpor library
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd

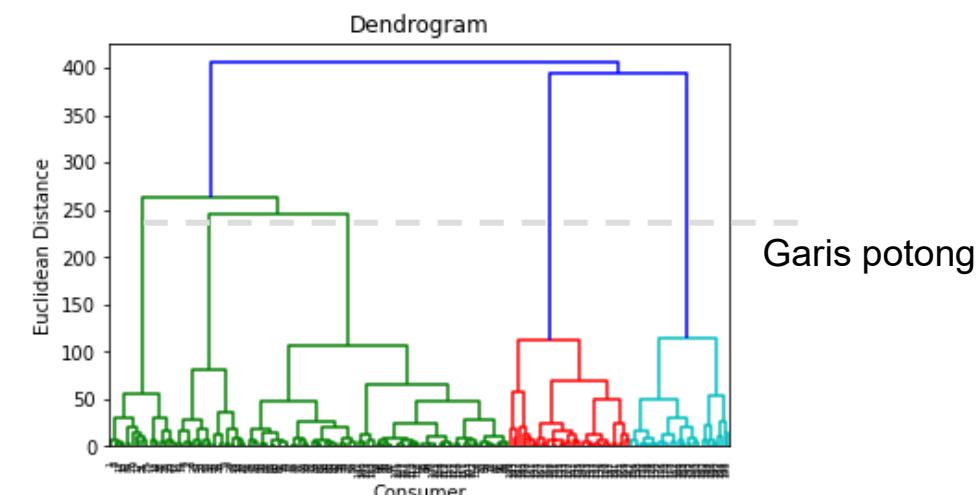
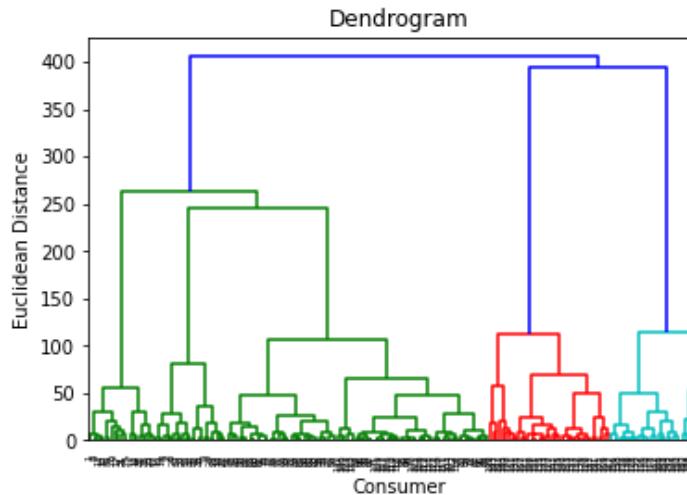
# Mengimpor dataset
dataset = pd.read_csv('Customer.csv')
X = dataset.iloc[:, [3, 4]].values

# Menggunakan dendrogram untuk menentukan angka cluster yang tepat
import scipy.cluster.hierarchy as sch
dendrogram = sch.dendrogram(sch.linkage(X, method = 'ward'))
plt.title('Dendrogram')
plt.xlabel('Consumer')
plt.ylabel('Euclidean Distance')
plt.show()
```



Studi Kasus Hierarchical Clustering

- Line 13-16 adalah perintah untuk memunculkan grafik dendrogram. Pada grafik dendrogram di bawah ini dapat diamati bahwa jumlah cluster yang paling sesuai adalah 5 dari jumlah garis vertical yang terpotong.



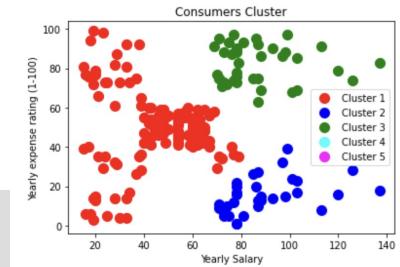


Studi Kasus Hierarchical Clustering

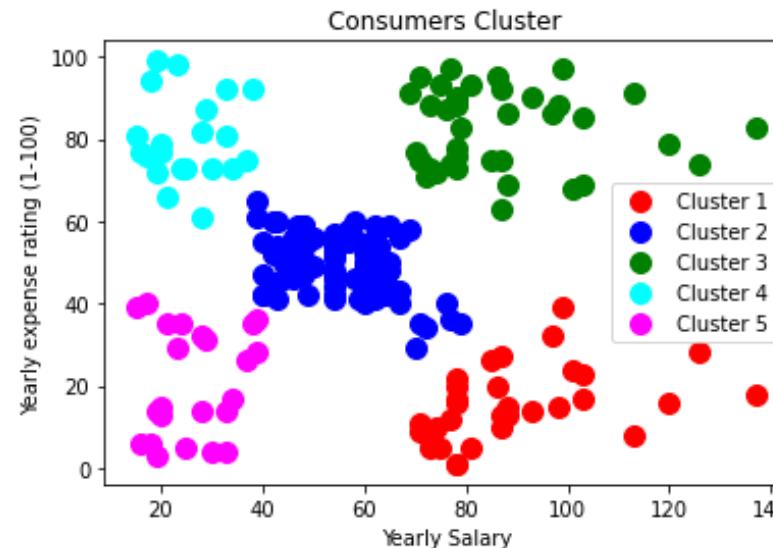
Hands-on

```
# Menjalankan Hierarchical Clustering ke dataset
from sklearn.cluster import AgglomerativeClustering
hc = AgglomerativeClustering(n_clusters = 3, affinity = 'euclidean', linkage = 'ward')
y_hc = hc.fit_predict(X)

# Visualisasi hasil clusters
plt.scatter(X[y_hc == 0, 0], X[y_hc == 0, 1], s = 100, c = 'red', label = 'Cluster 1')
plt.scatter(X[y_hc == 1, 0], X[y_hc == 1, 1], s = 100, c = 'blue', label = 'Cluster 2')
plt.scatter(X[y_hc == 2, 0], X[y_hc == 2, 1], s = 100, c = 'green', label = 'Cluster 3')
plt.scatter(X[y_hc == 3, 0], X[y_hc == 3, 1], s = 100, c = 'cyan', label = 'Cluster 4')
plt.scatter(X[y_hc == 4, 0], X[y_hc == 4, 1], s = 100, c = 'magenta', label = 'Cluster 5')
plt.title('Consumers Cluster')
plt.xlabel('Yearly Salary')
plt.ylabel('Yearly expense rating (1-100)')
plt.legend()
plt.show()
```



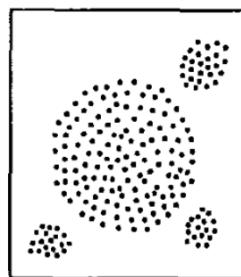
Studi Kasus Hierarchical Clustering



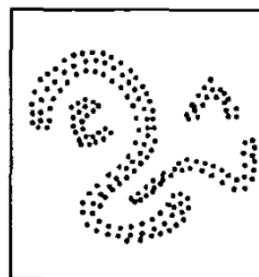
- Pada gambar di atas dapat diamati bahwa hasil clustering menggunakan metode hierarchical clustering dengan $K=5$. Semua titik dapat tergabung pada cluster dengan baik dan hasil clustering hampir sama hasilnya dengan metode K-Means

DBSCAN

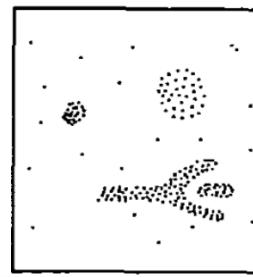
Cluster pada dasarnya adalah daerah padat dalam ruang data yg dibedakan daerahnya dari tingkat kepadatan. Kedua teknik (basis partisi dan hirarki) sebelumnya akan efisien jika dataset berbentuk normal (compact & well separated cluster: convex atau spherical-shape). Namun ketika data cluster berbentuk arbiter atau ingin mendeteksi cluster penculan (outlier), maka DBSCAN merupakan Teknik cluster yang sesuai. Untuk lebih jelasnya dapat diamati pada gambar berikut:



database 1

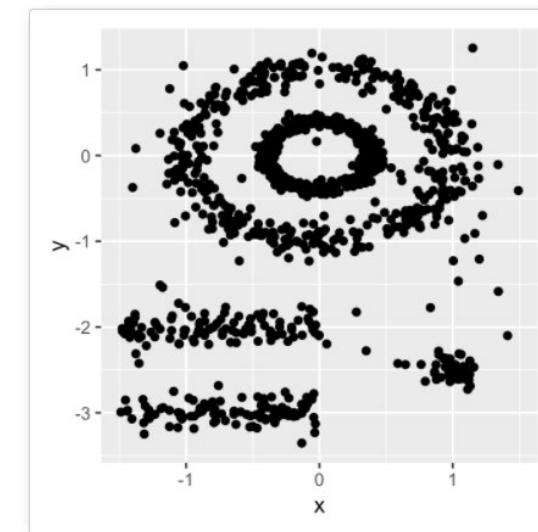


database 2



database 3

Ester et al.
1996



cara identifikasi
cluster ?

2 cluster oval
2 cluster linier
1 cluster compact

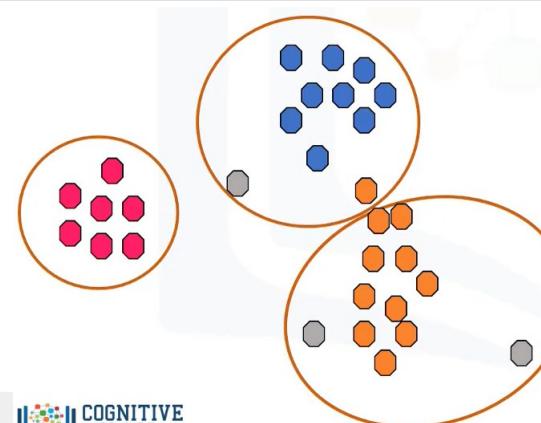
Image source:
<https://www.datanovia.com/en/lessons/dbscan-density-based-clustering-essentials/>



DBSCAN vs K-Means

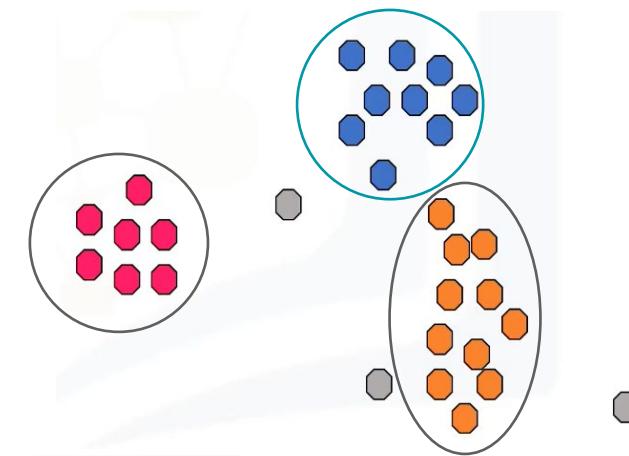
K-Means

1. Cenderung mengelompokan semua poin ke dalam beberapa kelompok (cluster) yg telah ditentukan sebelumnya, meskipun terdapat beberapa poin yang seharusnya/sebaiknya tidak berada dalam kelompok manapun yang ada/ditentukan
2. Dengan kata lain, K-Means “**memaksakan**” proses clusterisasi utk bbrp data point yg tidak/kurang layak dimasukan dalam suatu cluster



DBSCAN

1. Mengelompokan poin-poin terpilih yang selayaknya masuk dalam cluster yang terbentuk (tidak ditentukan sebelumnya)
2. Memisahkan poin-poin yang tidak terpilih ke dalam kategori outlier/pencilan





DBSCAN

Density-Based Spatial Clustering of Applications with Noise (DBSCAN) adalah:

- algoritma dasar untuk pengelompokan berbasis density. Algoritma ini dapat menemukan cluster dengan berbagai bentuk dan ukuran dari sejumlah besar data, yang mengandung **noise** dan **outlier**.
- Noise dalam DBSCAN adalah poin-poin dengan kepadatan/densitas objek yang sedikit
- Memiliki prinsip yakni suatu poin dapat dikelompokan/kategori apabila poin tsb berada dekat dengan poin-poin dalam kategori tsb



DBSCAN

Algoritma DBSCAN menggunakan dua parameter yaitu:

minPts: Jumlah minimum titik (ambang batas) yang dikelompokkan bersama agar suatu wilayah dianggap density.

eps (ϵ) atau Radius: Ukuran jarak radius yang akan digunakan untuk menemukan titik-titik di sekitar titik mana pun.

Kedua parameter ini dapat diterapkan dengan baik dengan menggunakan dua konsep yaitu Density Reachability dan Density Connectivity



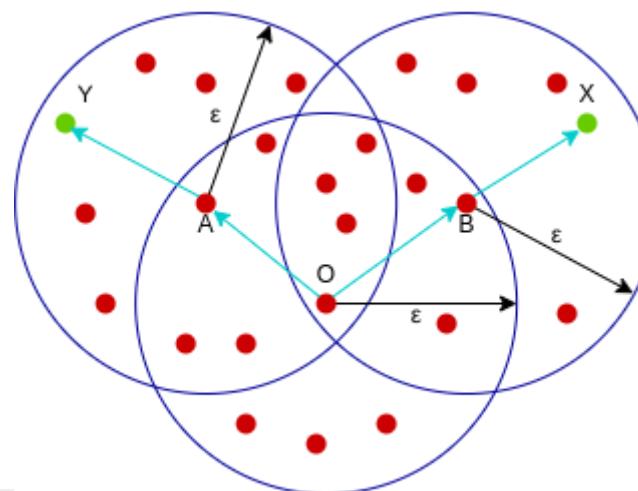
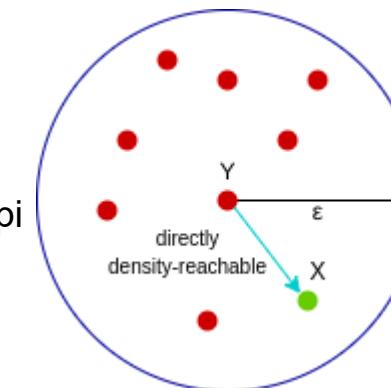
DBSCAN

Reachability pada konsep ini, untuk menentukan kepadatan dialukan dengan menetapkan suatu titik yang dapat dijangkau dari yang lain jika terletak dalam jarak tertentu (eps) darinya.

Connectivity, konsep ini melakukan pendekatan chaining berbasis transitivitas untuk menentukan apakah titik terletak di cluster tertentu. Misalnya, titik p dan q dapat dihubungkan jika $p \rightarrow r \rightarrow s \rightarrow t \rightarrow q$, di mana $x \rightarrow y$ berarti x berada di sekitar (neighborhood) y.

DBSCAN: Reachability and Connectivity

X secara langsung dapat dijangkau kepadatan (directly density-reachable) dari Y, tetapi sebaliknya tidak valid.



titik X terhubung kerapatan (density-connected) dari titik Y w.r.t ϵ /R dan minPoints jika ada titik O sedemikian rupa sehingga X dan Y dapat dijangkau kerapatan dari O w.r.t ke ϵ dan minPoints.



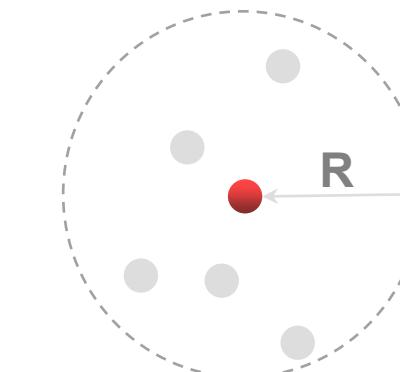
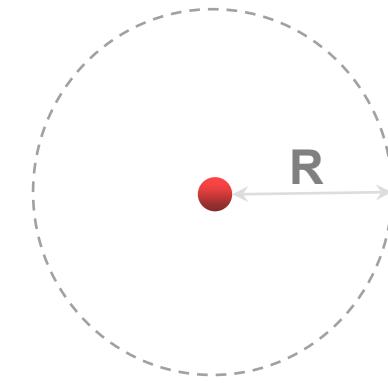
Parameter DBSCAN

Radius (jari-jari utk area terdekat)

Jika suatu poin masih dalam jangkauan radius (R) dan memiliki sejumlah poin lain dalam area tsb, maka area ini dikategorikan sebagai area (dengan densitas) padat.

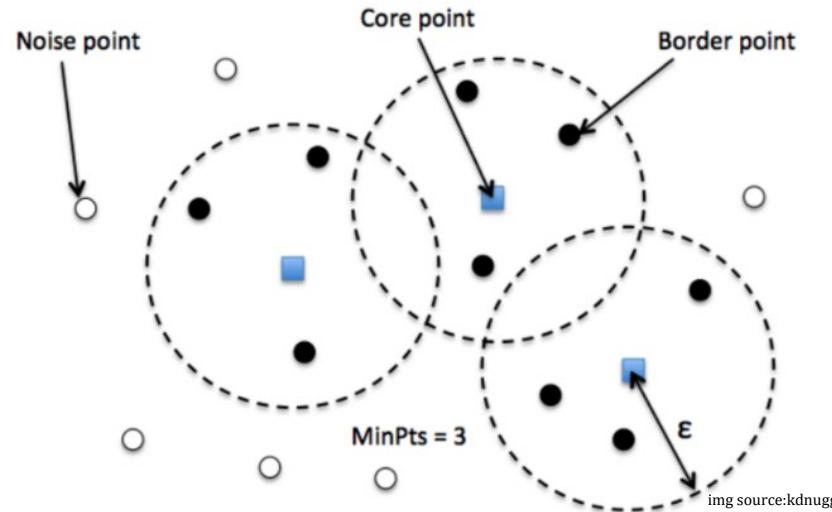
MinPoints/M (jumlah poin minimum utk membentuk suatu area)

Parameter M merupakan jumlah poin minimum di dalam suatu area yang dibutuhkan utk kategori sebagai satu area padat



DBSCAN

Terdapat tiga jenis titik setelah pengelompokan DBSCAN selesai:



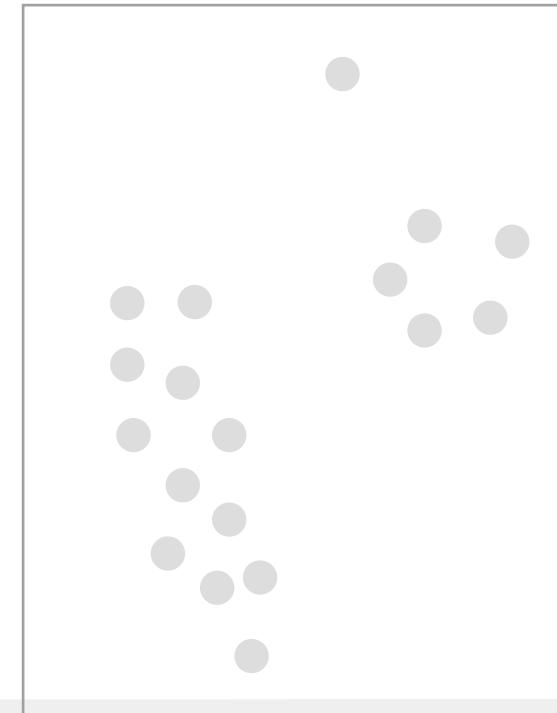
Core adalah titik yang memiliki setidaknya m titik dalam jarak (radius/epsilon) n dari dirinya sendiri.

Border adalah titik yang memiliki setidaknya satu titik Inti pada jarak n.

Noise/Outlier adalah titik yang bukan Core atau Border. Dan ia memiliki kurang dari m titik dalam jarak n dari dirinya sendiri.

Ilustrasi DBSCAN

Ditentukan parameter: $R/\epsilon = 2$; dan $\text{MinPoints}/M = 5$

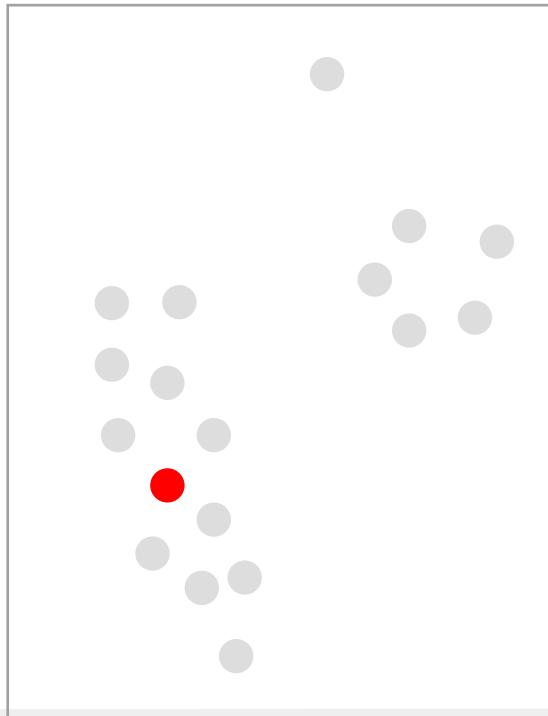




Ilustrasi DBSCAN

parameter:

R/epsilon = 2; dan M = 5



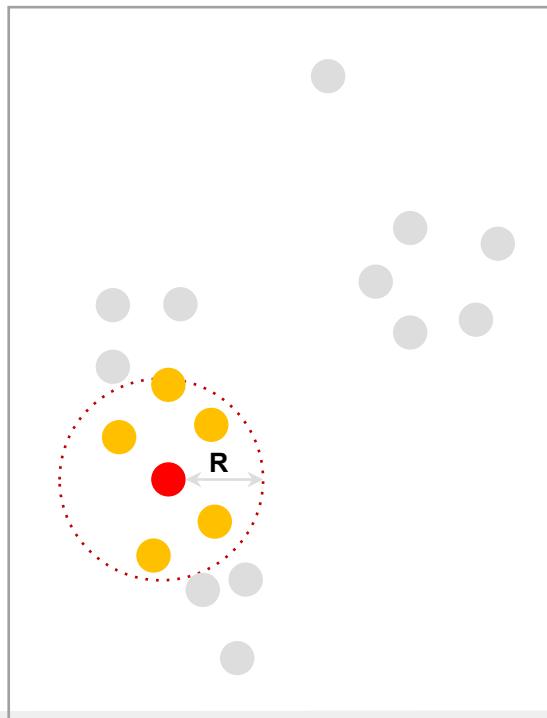
Langkah 1:
Pilih satu poin secara acak



Ilustrasi DBSCAN

parameter:

R/ϵ psilon = 2; dan $M = 5$



Langkah 1:

Pilih satu poin secara acak

Langkah 2:

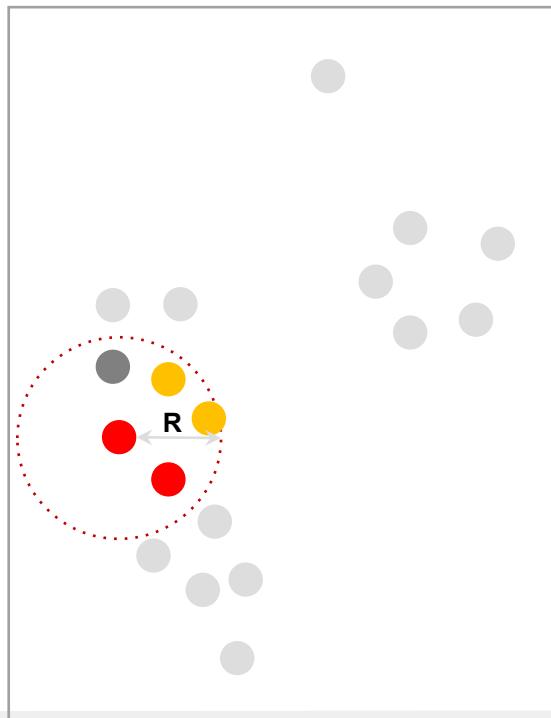
Kategorikan poin

Poin pertama merupakan poin inti (core) krn dalam radius R sebesar 2 unit, terdapat 6 poin (termasuk poin pertama)

Ilustrasi DBSCAN

parameter:

$R/\text{epsilon} = 2$; dan $M = 5$



Langkah 1:

Pilih satu poin secara acak

Langkah 2:

Kategorikan poin

Langkah 3:

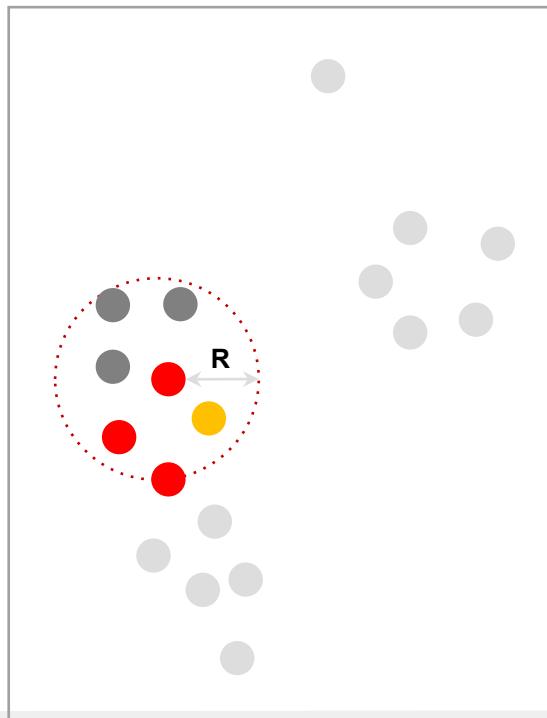
Ulangi langkah 1 dan 2 hingga semua poin telah dikategorikan

Poin kedua merupakan poin inti (core) krn dalam radius R sebesar 2 unit, terdapat 5 poin data

Ilustrasi DBSCAN

parameter:

$R/\text{epsilon} = 2$; dan $M = 5$



Langkah 1:

Pilih satu poin secara acak

Langkah 2:

Kategorikan poin

Langkah 3:

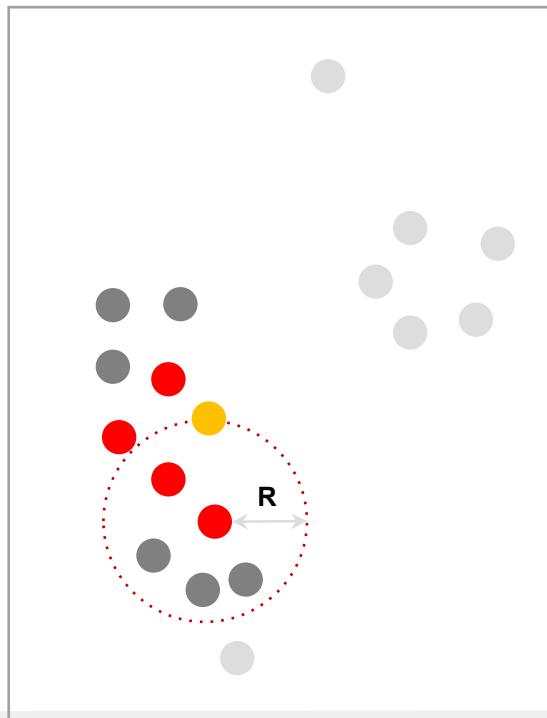
Ulangi langkah 1 dan 2
hingga semua poin telah
dikategorikan

Poin ketiga merupakan poin inti (core) krn dalam radius R sebesar 2 unit, terdapat 7 poin data ($> M=5$)

Ilustrasi DBSCAN

parameter:

R/ϵ = 2; dan $M = 5$



Langkah 1:

Pilih satu poin secara acak

Langkah 2:

Kategorikan poin

Langkah 3:

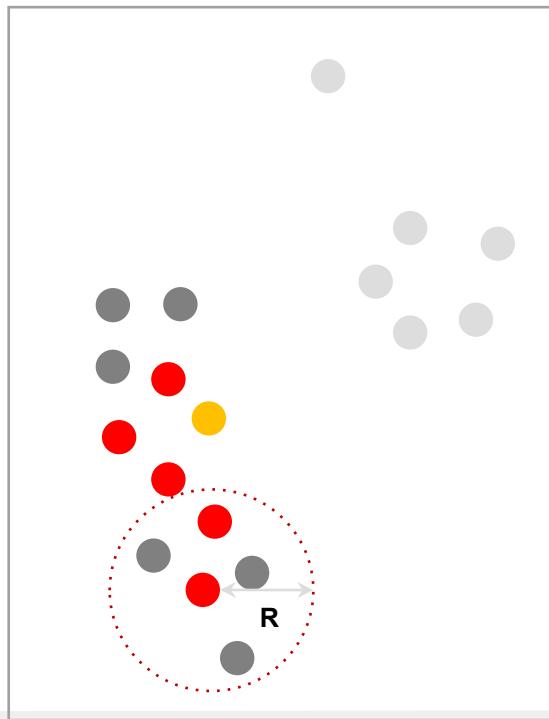
Ulangi langkah 1 dan 2
hingga semua poin telah
dikategorikan

Poin berikutnya merupakan poin inti (core) krn dalam radius R sebesar 2 unit, terdapat 6 poin data ($> M = 5$)

Ilustrasi DBSCAN

parameter:

R/ϵ = 2; dan $M = 5$



Langkah 1:

Pilih satu poin secara acak

Langkah 2:

Kategorikan poin

Langkah 3:

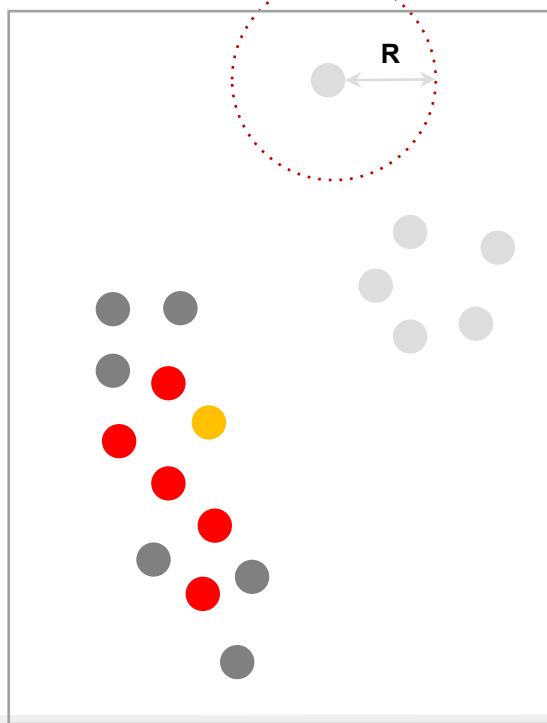
Ulangi langkah 1 dan 2 hingga semua poin telah dikategorikan

Poin berikutnya merupakan poin inti (core) krn dalam radius R sebesar 2 unit, terdapat 5 poin data

Ilustrasi DBSCAN

parameter:

$R/\text{epsilon} = 2$; dan $M = 5$



Langkah 1:

Pilih satu poin secara acak

Langkah 2:

Kategorikan poin

Langkah 3:

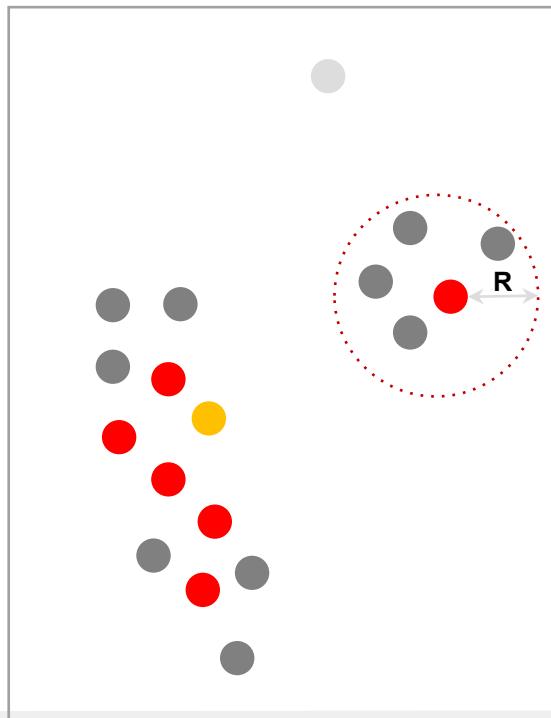
Ulangi langkah 1 dan 2 hingga semua poin telah dikategorikan

- *Poin berikutnya merupakan poin outlier/noise(pencilan) krn dalam radius R sebesar 2 unit, hanya terdiri poin itu sendiri < minimum poin (M)*
- *Poin tidak dapat menjangkau radius poin inti yang lain (tidak memenuhi syarat poin tepi)*

Ilustrasi DBSCAN

parameter:

R/ϵ = 2; dan $M = 5$



Langkah 1:

Pilih satu poin secara acak

Langkah 2:

Kategorikan poin

Langkah 3:

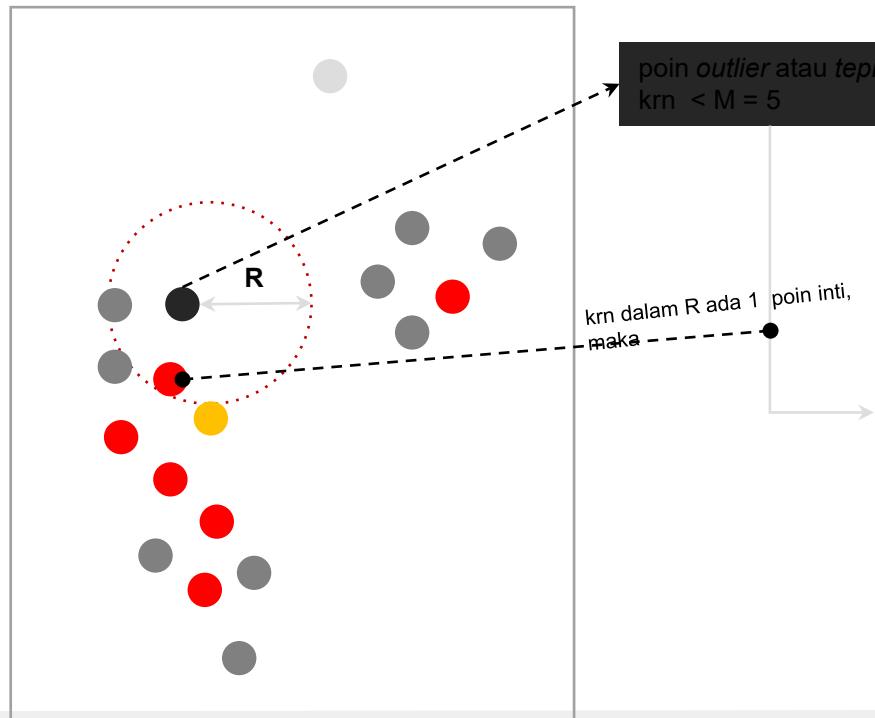
Ulangi langkah 1 dan 2 hingga semua poin telah dikategorikan

- *Poin berikutnya merupakan poin inti krn dalam radius $R=2$ terdiri dari 5 poin*

Ilustrasi DBSCAN

parameter:

$R/\text{epsilon} = 2$; dan $M = 5$



Langkah 1:

Pilih satu poin secara acak

Langkah 2:

Kategorikan poin

Langkah 3:

Ulangi langkah 1 dan 2 hingga semua poin telah dikategorikan

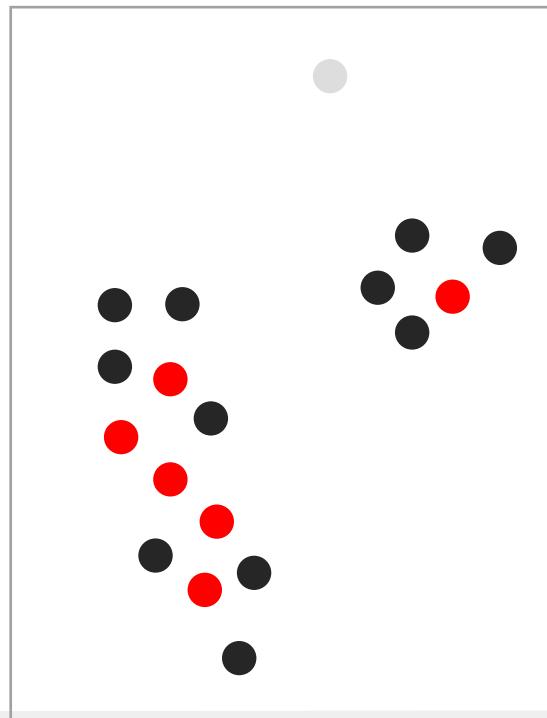
- *Poin berikutnya merupakan poin tepi krn dalam radius $R=2$ ada 1 poin inti. Artinya poin tersebut berdekatan dengan poin inti*



Ilustrasi DBSCAN

parameter:

R/epsilon = 2; dan M = 5



Langkah 1:

Pilih satu poin secara acak

Langkah 2:

Kategorikan poin

Langkah 3:

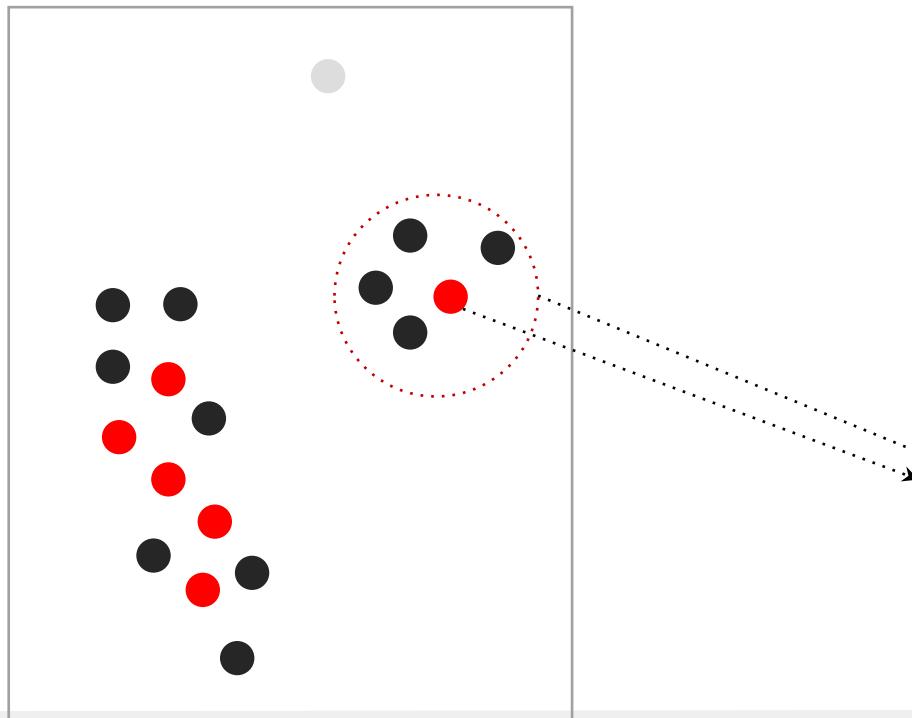
Ulangi langkah 1 dan 2 hingga semua poin telah dikategorikan

- *Seluruh poin telah dikategorikan tanpa terlewat satupun*

Ilustrasi DBSCAN

parameter:

$R/\text{epsilon} = 2$; dan $M = 5$



Langkah 1:

Pilih satu poin secara acak

Langkah 2:

Kategorikan poin

Langkah 3:

Ulangi langkah 1 dan 2 hingga semua poin telah dikategorikan

Langkah 4:

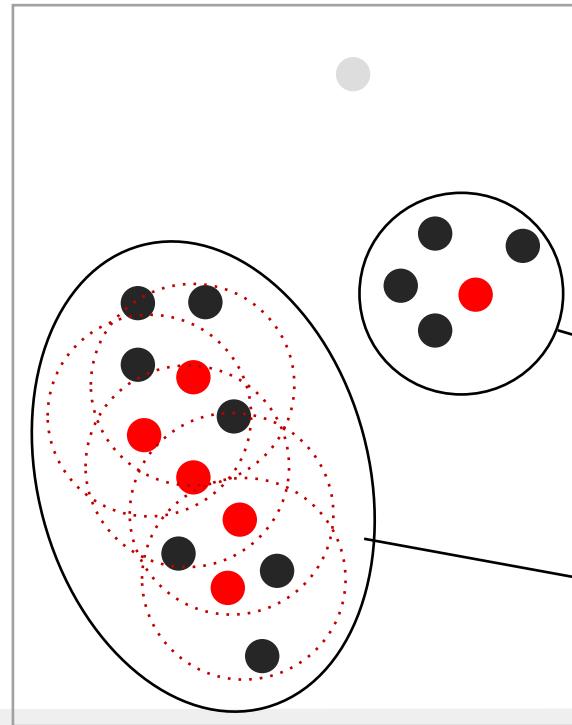
Menentukan jumlah cluster

Utk menentukan jumlah cluster, kita gunakan alat bantu lingkaran sebesar radius $R/\text{epsilon}$, dengan titik pusat yaitu poin inti/core

Ilustrasi DBSCAN

parameter:

$R/\text{epsilon} = 2$; dan $M = 5$



Langkah 1:

Pilih satu poin secara acak

Langkah 2:

Kategorikan poin

Langkah 3:

Ulangi langkah 1 dan 2 hingga semua poin telah dikategorikan

Langkah 4:

Menentukan jumlah cluster

Cluster 1: krn di dalam jangkauan radius poin hanya memiliki satu poin inti

Cluster 2: krn di dalam jangkauan radius poin hanya memiliki lebih dari satu poin inti yang berkesinambungan (beririsan/singgungan)



Kesimpulan DBSCAN

- Memiliki kemampuan untuk mengkategorikan dataset dengan bentuk sembarang (arbiter), dan tidak selalu areanya berbentuk lingkaran yang sejenis
- Memiliki kemampuan sekaligus deteksi noise/outlier
- Memiliki kecerdasan lebih baik vs K-Means
- Tidak bergantung pada bentuk awal “cluster” data
- Pemilihan parameter R/Epsilon dan MinPoint masih debatable (best practice setiap individu dan dataset yang ada akan berbeda (rumus yg terkait ukuran dataset (dimensi), trial-error, KNN)



Cheat Sheet Clustering

| Type | K Means | Hierarchical | PCA |
|---------------------|--|--|---|
| Overview of process | Clustering Nonhierarchical method that finds mutually exclusive spherical clusters based on distance. | Clustering Repeatedly links pairs of clusters until every data object is included. | Dimension Reduction Transforms high dimensional data into low dimensional data using orthogonal transformations. |
| Disadvantages | <ul style="list-style-type: none">Requires known "k" clusters; Can be difficult to choose.Initial cluster choices and order of data strongly affect results.Can be difficult to reproduce results due to random initial "centroid" choice. | <ul style="list-style-type: none">Initial seeds, data order have strong effect.Merges cannot be undone.No statistical / theoretical foundation for results.Sensitive to outliers. | <ul style="list-style-type: none">Principal components (a linear combination of the original features) can be challenging to interpret and read, compared to the original features.Data must be standardized beforehand. |
| Advantages | <ul style="list-style-type: none">Easy to implement.Fast (for small k).Clusters can be recalibrated. | <ul style="list-style-type: none">Easy to implement.Dendrogram makes for easy visualization of "k".Results easily reproducible. | <ul style="list-style-type: none">Good visualization tool.Reduces irrelevant or redundant features.Reduces overfitting (by reducing features). |
| Works well for | Big data; Hyper spherical (e.g. 3D sphere). | Small to medium data. Performance and execution time increase dramatically for large data sets. | Extracting important features (components) from big data. |



Studi Kasus DBSCAN

Kasus 1.

Penerapan DBSCAN pada cluster spherical data.

1. Generate data set pelatihan sebanyak 750 titik data pelatihan bola dengan label yang sesuai
2. Melakukan normalisasi fitur pada proses pelatihan data,
3. menggunakan DBSCAN dari library sklearn.



Studi Kasus DBSCAN

Kasus 1. Coding 1

```
import numpy as np
from sklearn.cluster import DBSCAN
from sklearn import metrics
from sklearn.datasets import make_blobs
from sklearn.preprocessing import StandardScaler

# 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)

# Menghitung DBSCAN
db = DBSCAN(eps=0.3, min_samples=10).fit(X)
core_samples_mask = np.zeros_like(db.labels_, dtype=bool)
core_samples_mask[db.core_sample_indices_] = True
labels = db.labels_
```



Studi Kasus DBSCAN

Kasus 1. Coding 2

```
# Number of clusters in labels, ignoring noise if present.  
n_clusters_ = len(set(labels)) - (1 if -1 in labels else 0)  
n_noise_ = list(labels).count(-1)  
  
print('Estimated number of clusters: %d' % n_clusters_)  
print('Estimated number of noise points: %d' % n_noise_)  
print("Homogeneity: %0.3f" % metrics.homogeneity_score(labels_true, labels))  
print("Completeness: %0.3f" % metrics.completeness_score(labels_true, labels))  
print("V-measure: %0.3f" % metrics.v_measure_score(labels_true, labels))  
print("Adjusted Rand Index: %0.3f"  
      % metrics.adjusted_rand_score(labels_true, labels))  
print("Adjusted Mutual Information: %0.3f"  
      % metrics.adjusted_mutual_info_score(labels_true, labels))  
print("Silhouette Coefficient: %0.3f"  
      % metrics.silhouette_score(X, labels))
```



Studi Kasus DBSCAN

Kasus 1. Coding 3

```
# Plot result
import matplotlib.pyplot as plt

# Black removed and is used for noise instead.
unique_labels = set(labels)
colors = [plt.cm.Spectral(each)
          for each in np.linspace(0, 1, len(unique_labels))]
for k, col in zip(unique_labels, colors):
    if k == -1:
        # Black used for noise.
        col = [0, 0, 0, 1]

    class_member_mask = (labels == k)

    xy = X[class_member_mask & core_samples_mask]
    plt.plot(xy[:, 0], xy[:, 1], 'o', markerfacecolor=tuple(col),
              markeredgecolor='k', markersize=14)

    xy = X[class_member_mask & ~core_samples_mask]
    plt.plot(xy[:, 0], xy[:, 1], 'o', markerfacecolor=tuple(col),
              markeredgecolor='k', markersize=6)

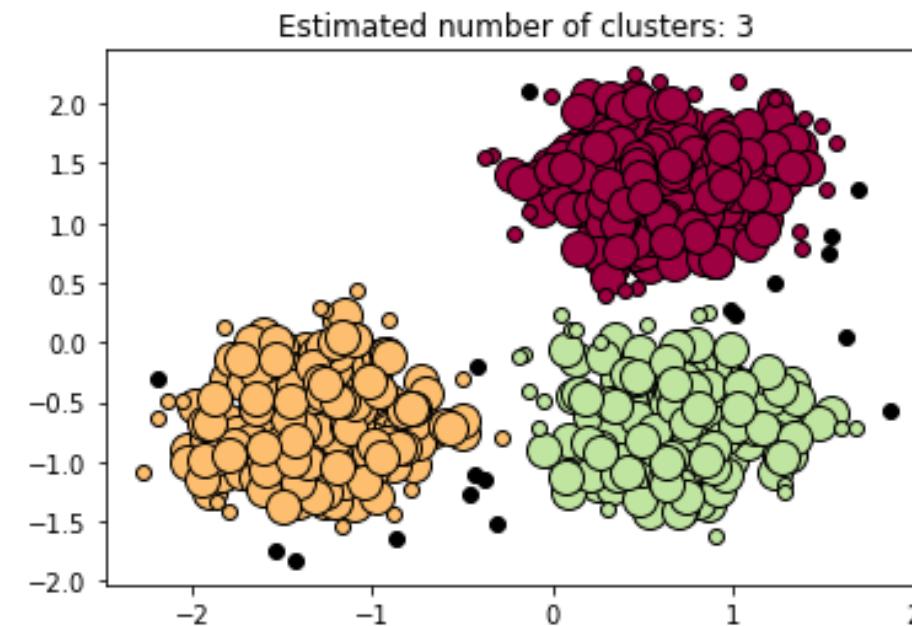
plt.title('Estimated number of clusters: %d' % n_clusters_)
plt.show()
```



Studi Kasus DBSCAN

Kasus 1. Hasil

Estimated number of clusters: 3
Estimated number of noise points: 18
Homogeneity: 0.953
Completeness: 0.883
V-measure: 0.917
Adjusted Rand Index: 0.952
Adjusted Mutual Information: 0.916
Silhouette Coefficient: 0.626





Studi Kasus DBSCAN

Kasus 2.

Penerapan DBSCAN pada cluster non-spherical data.

1. Generate data set pelatihan sebanyak 750 titik data pelatihan bola dengan label yang sesuai
2. Melakukan normalisasi fitur pada proses pelatihan data,
3. menggunakan DBSCAN dari library sklearn.



Studi Kasus DBSCAN

Kasus 2. Coding 1

```
#Penerapan DBSCAN pada cluster non-spherical data.
import numpy as np
import matplotlib.pyplot as plt
from sklearn import metrics
from sklearn.datasets import make_circles
from sklearn.preprocessing import StandardScaler
from sklearn.cluster import DBSCAN
X, y = make_circles(n_samples=750, factor=0.3, noise=0.1)
X = StandardScaler().fit_transform(X)
y_pred = DBSCAN(eps=0.3, min_samples=10).fit_predict(X)

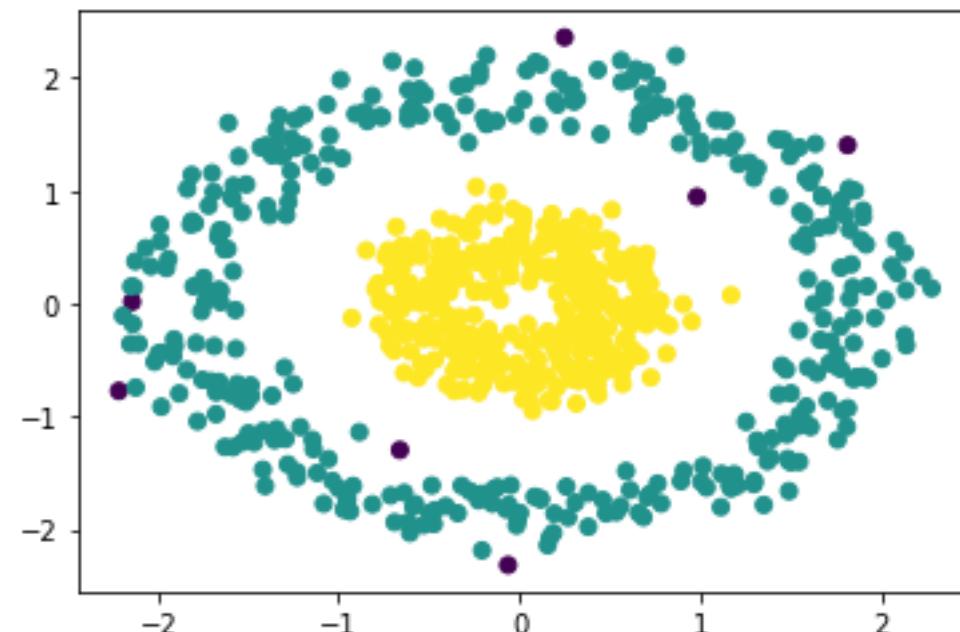
plt.scatter(X[:,0], X[:,1], c=y_pred)
print('Number of clusters: {}'.format(len(set(y_pred[np.where(y_pred != -1)]))))
print('Homogeneity: {}'.format(metrics.homogeneity_score(y, y_pred)))
print('Completeness: {}'.format(metrics.completeness_score(y, y_pred)))
print("V-measure: %0.3f" % metrics.v_measure_score(labels_true, labels))
print("Adjusted Rand Index: %0.3f"
      % metrics.adjusted_rand_score(labels_true, labels))
print("Adjusted Mutual Information: %0.3f"
      % metrics.adjusted_mutual_info_score(labels_true, labels))
print("Silhouette Coefficient: %0.3f"
      % metrics.silhouette_score(X, labels))
```



Studi Kasus DBSCAN

Kasus 2. Hasil

Number of clusters: 2
Homogeneity: 1.0000000000000007
Completeness: 0.9372565941867823
V-measure: 0.917
Adjusted Rand Index: 0.952
Adjusted Mutual Information: 0.916
Silhouette Coefficient: -0.061



Studi Kasus DBSCAN

Kasus 2. Hasil bila menggunakan metode clustering K-Means

Number of clusters: 2

Homogeneity: 0.0016646133996537423

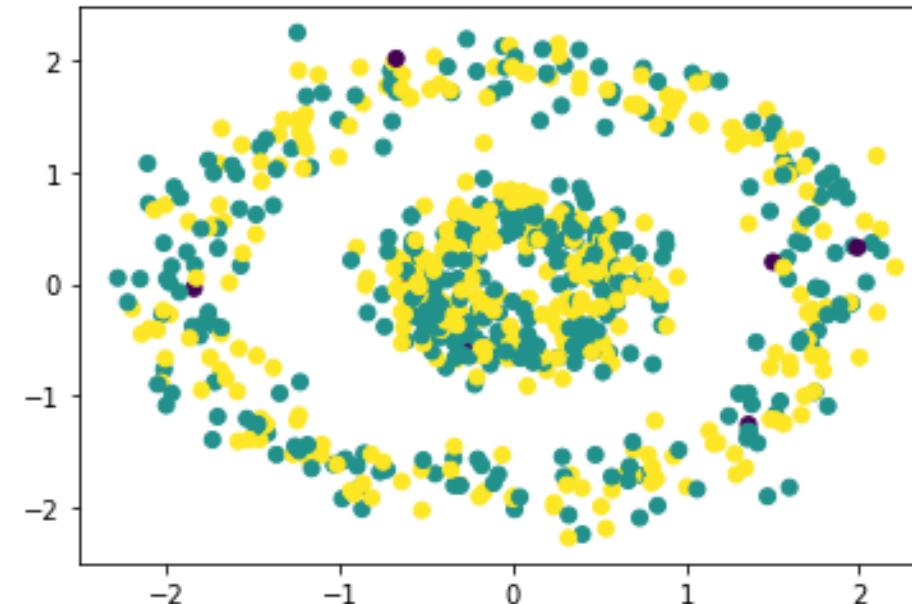
Completeness: 0.0015601698855971465

V-measure: 0.917

Adjusted Rand Index: 0.952

Adjusted Mutual Information: 0.916

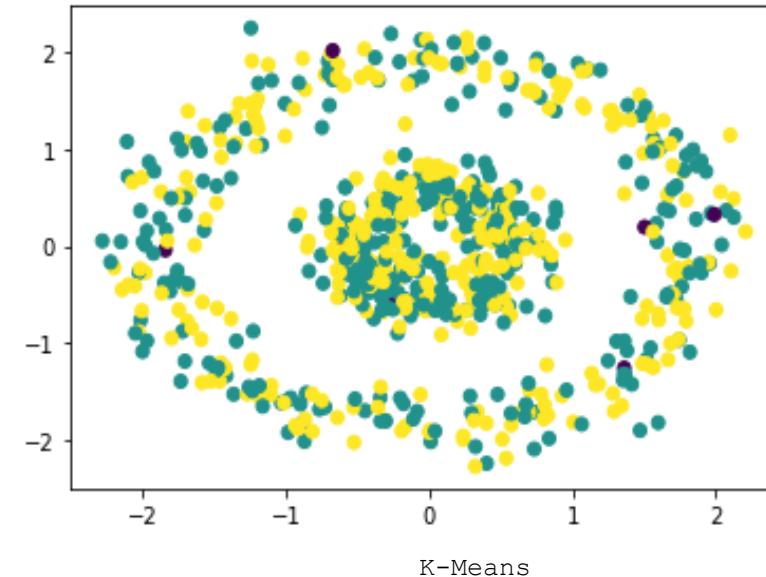
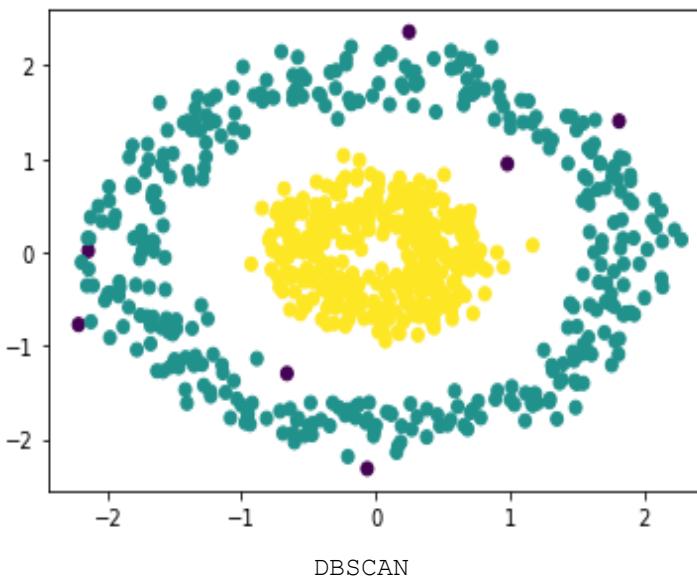
Silhouette Coefficient: -0.057



Studi Kasus DBSCAN

Kasus 2. Hasil bila menggunakan metode clustering K-Means

Dapat diamati bahwa DBSCAN menghasilkan hasil cluster yang sesuai





Referensi

- Ethem Alpaydin, "Introduction to Machine Learning, Fourth Edition", MIT Press, 2020
- ZhiHua ZhouShaowu Liu, "Machine Learning", Springer,2021
- Charu C. AggarwalChandan K. Reddy," Data Clustering", Chapman and Hall/CRC,2013
- Anil K. Jain, Richard C. Dubes, "Algorithm for Clustering Data", Michigan State University, Prentice Hall Advanced Reference, 2018
(free download at http://www.cse.msu.edu/~jain/Clustering_Jain_Dubes.pdf)



Pembuat:

Dr. Berlian Al Kindhi (Institut Teknologi Sepuluh Nopember)

Email: berlian@its.ac.id



Tools / Lab Online

1. Spyder
2. Jupyter Notebook



Quiz / Tugas

Quiz dapat diakses melalui <https://spadadikti.id/>



Terima kasih