

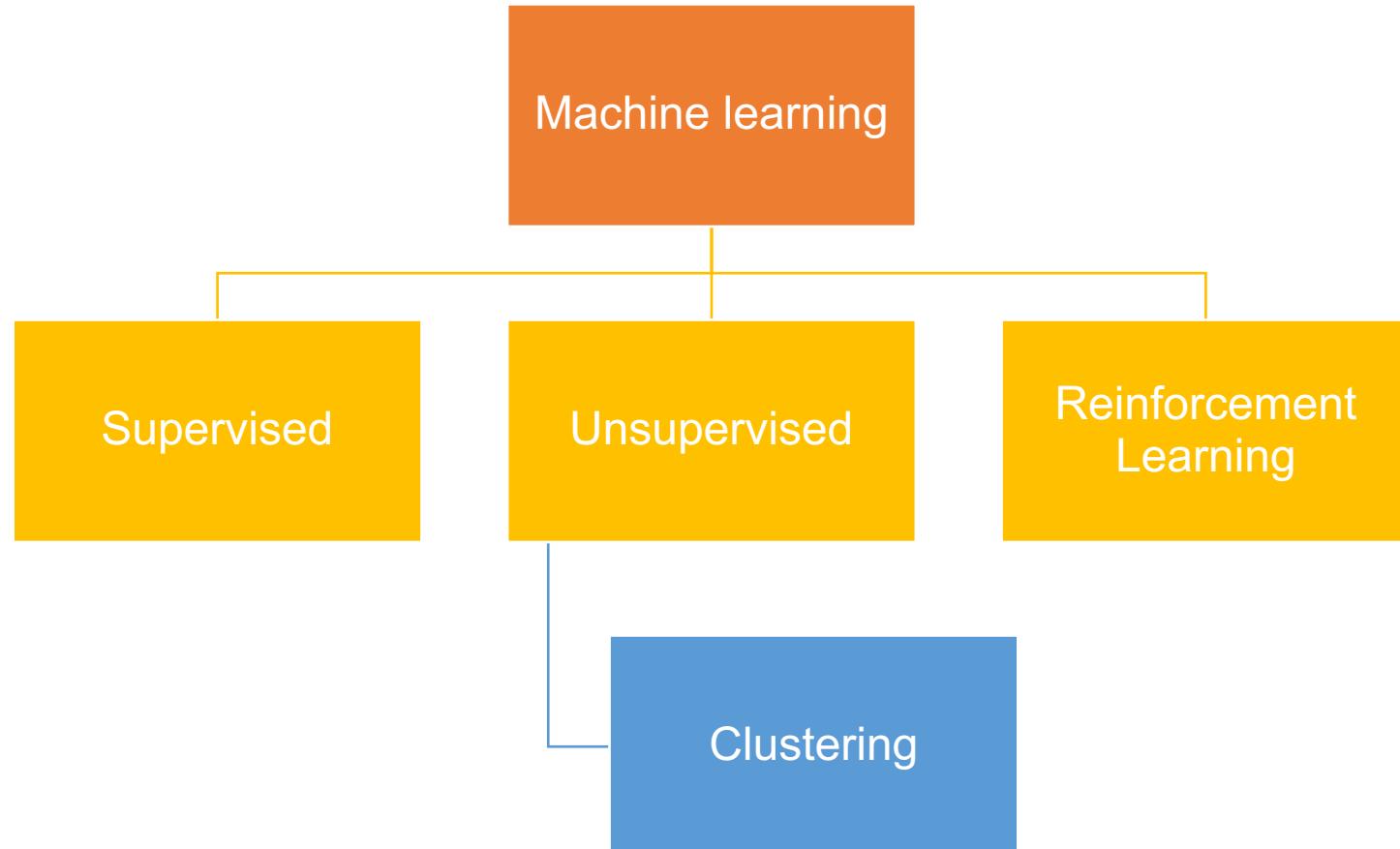
Unsupervised Learning (Kmeans & DBSCAN)

Artificial Intelligence dan Big Data | AAK2KAB3 | Kur. 2024 | 2024/2025

Unsupervised Learning

Unsupervised Learning adalah teknik machine learning di mana model belajar dari data **tanpa label**. Salah satu tugas utama dalam unsupervised learning adalah **clustering**, yaitu mengelompokkan data berdasarkan kesamaan karakteristiknya.

Clustering

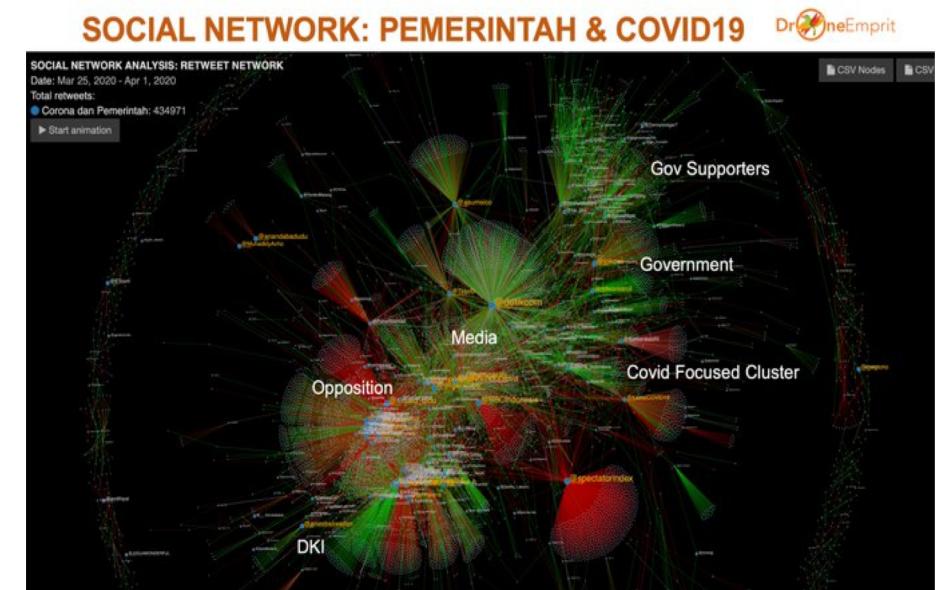
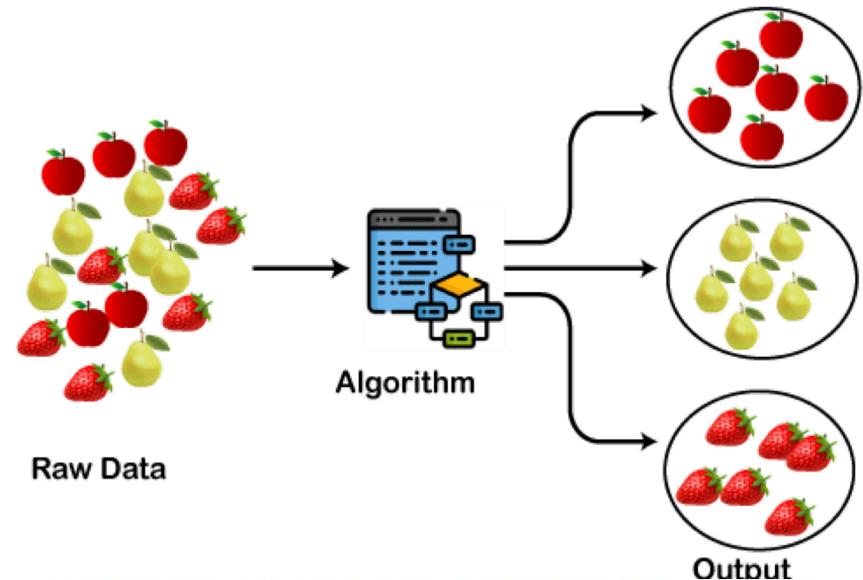


Clustering

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



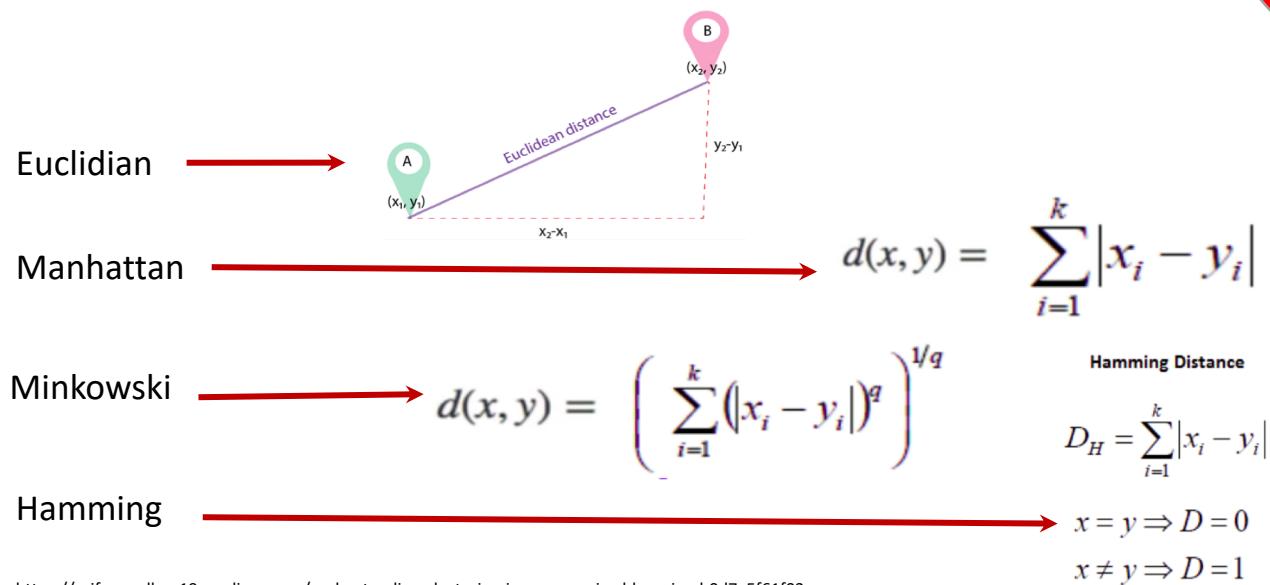
Kapan digunakannya Clustering

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 outlier. Pemahaman anomali data membantu optimasi tools koleksi data dan hasil lebih akurat

Clustering

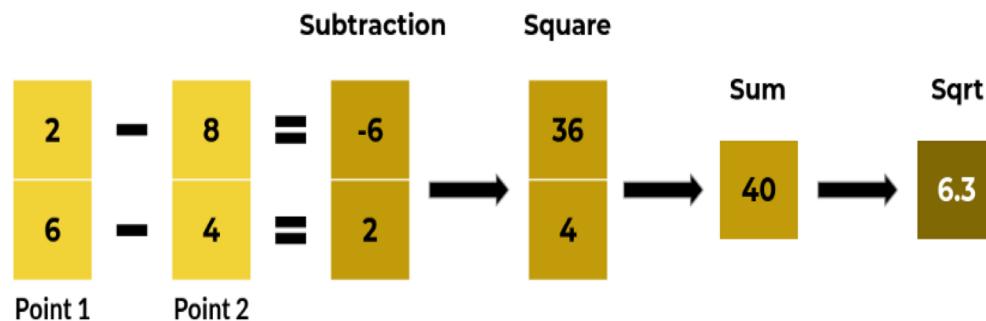
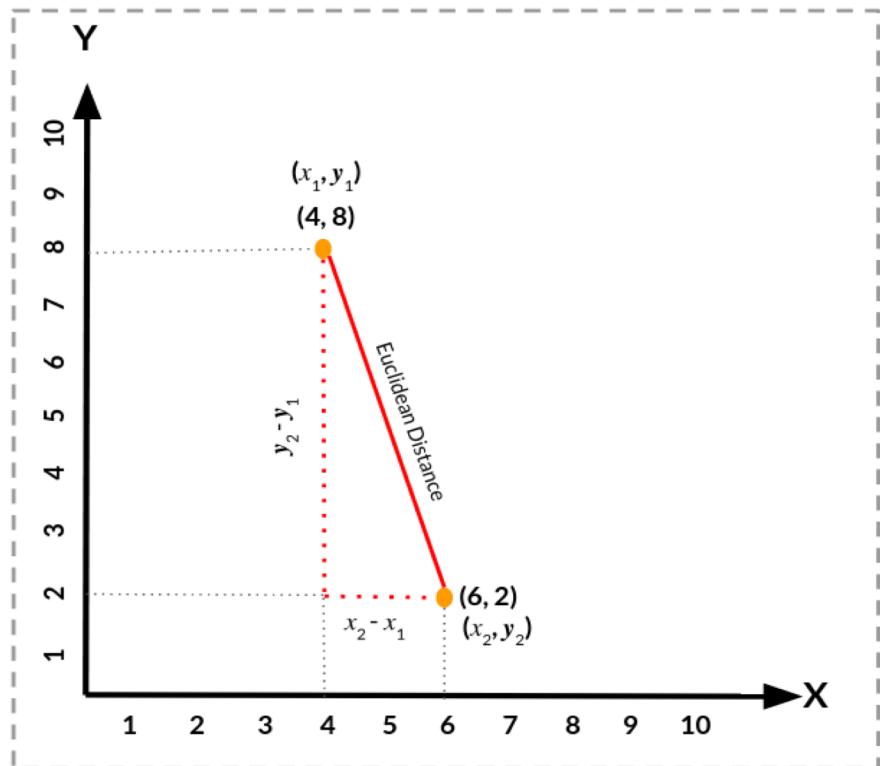
Proses Clustering



<https://ariffromadhan19.medium.com/understanding-clustering-in-unsupervised-learning-b0d7a5f61f03>

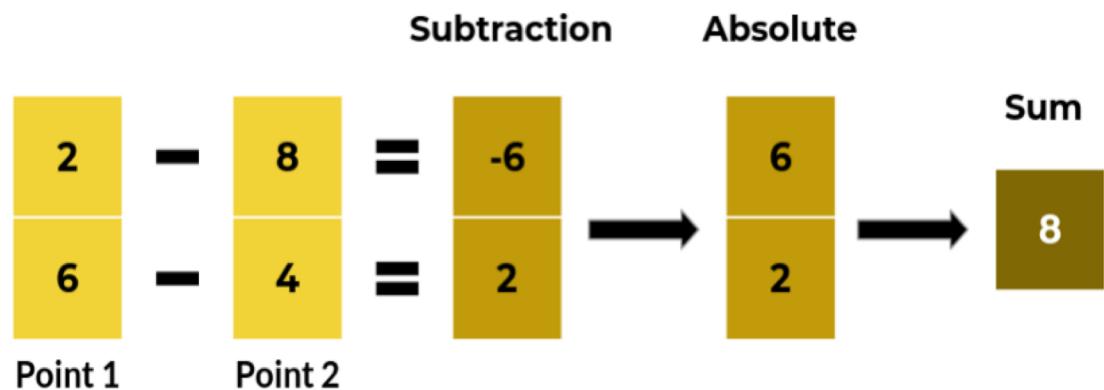
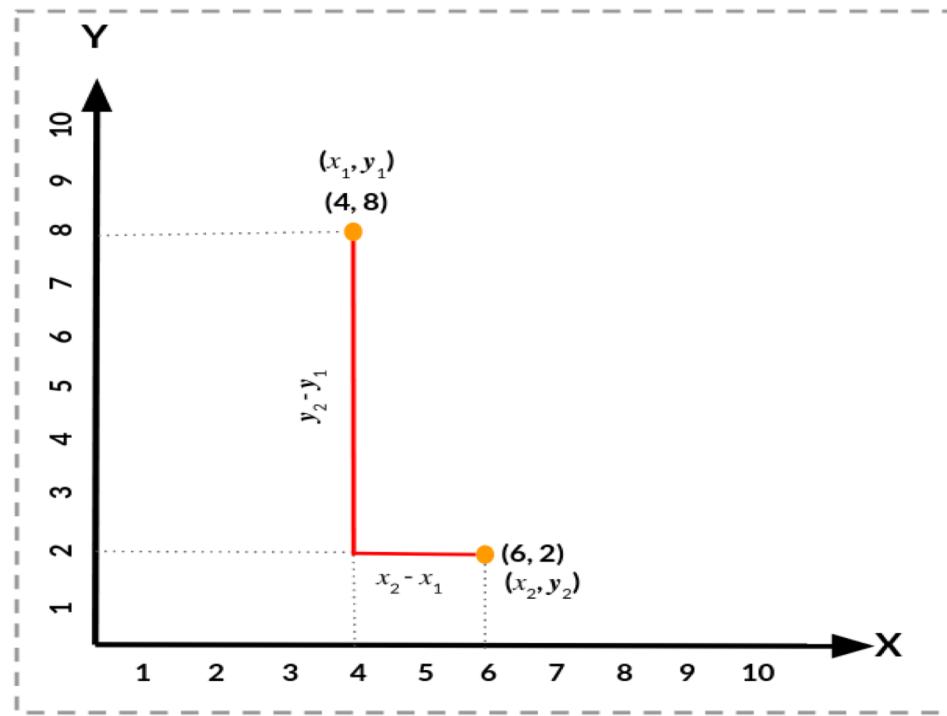
Clustering

Contoh Euclidian Distance

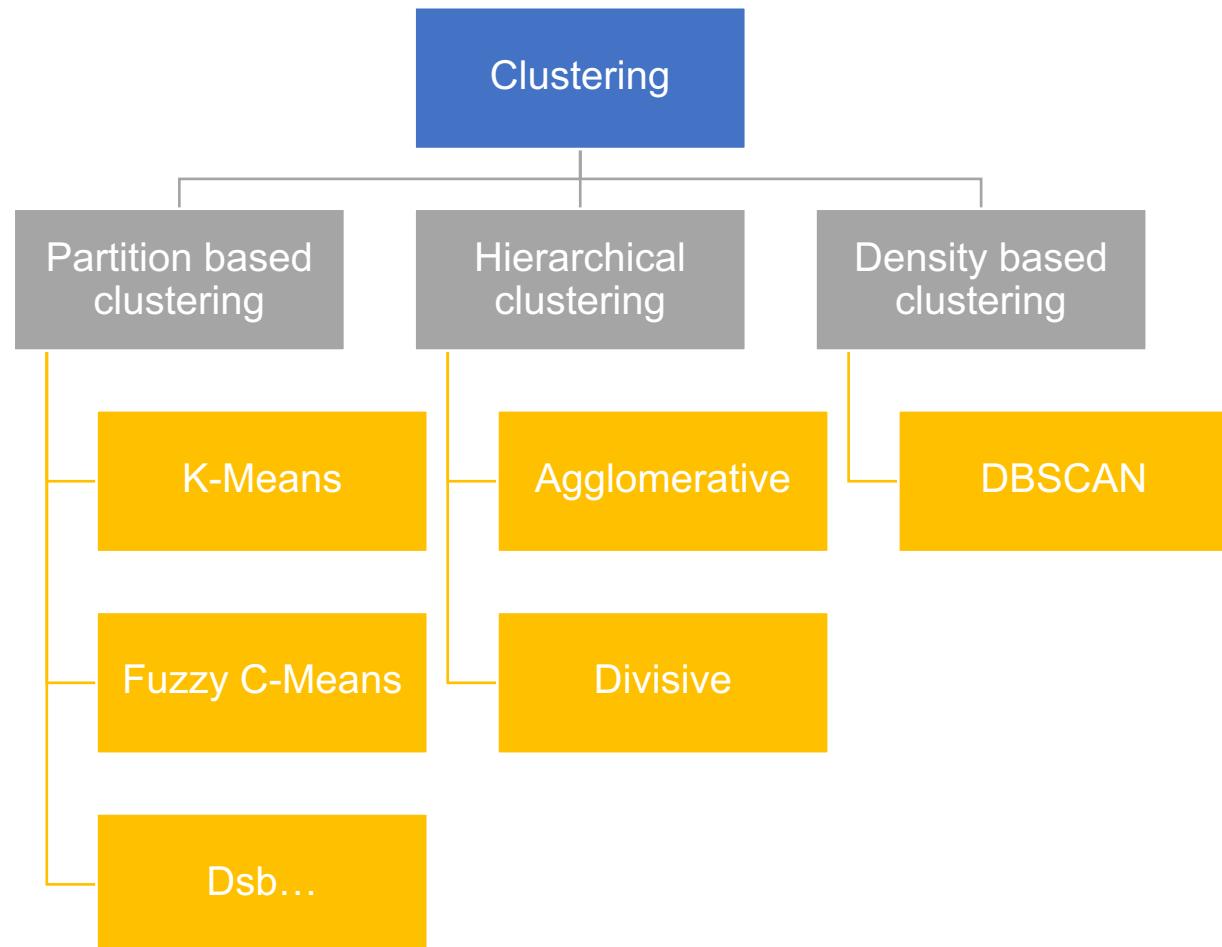


Clustering

Contoh Manhattan Distance



Clustering



K-Means adalah algoritma clustering berbasis **centroid** yang bekerja dengan membagi dataset menjadi **K kelompok** berdasarkan kedekatan titik data dengan pusat cluster (**centroid**).

Cara Kerja K-Means:

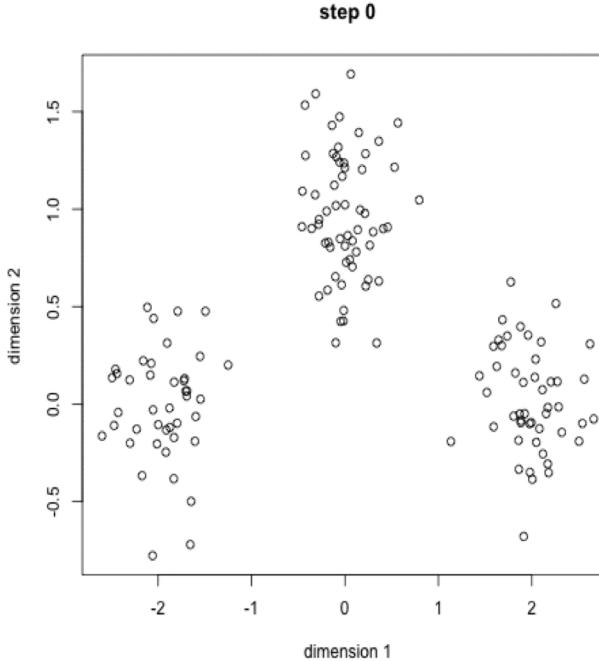
1. Tentukan jumlah cluster K
2. Pilih K titik secara acak sebagai **centroid awal**.
3. Hitung jarak setiap data ke centroid, lalu **tetapkan ke cluster terdekat**.
4. Perbarui **centroid** dengan menghitung rata-rata titik dalam masing-masing cluster.
5. Ulangi langkah 3-4 hingga **centroid tidak berubah** atau mencapai iterasi maksimum.

K-Means Clustering

$$W(\mathcal{C}_k) = \sum_{x_i \in \mathcal{C}_k} (x_i - \mu_k)^2$$

where:

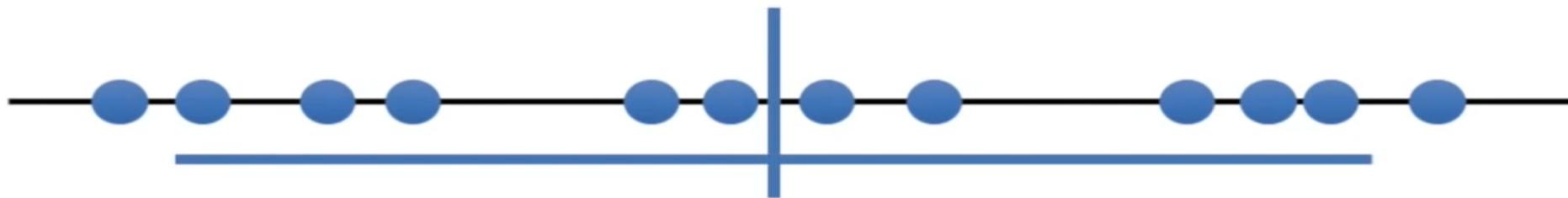
- x_i is a data point belonging to the cluster \mathcal{C}_k
- μ_k is the mean value of the points assigned to the



K-Means Clustering

Ilustrasi K-Means

Mulai dengan K=1



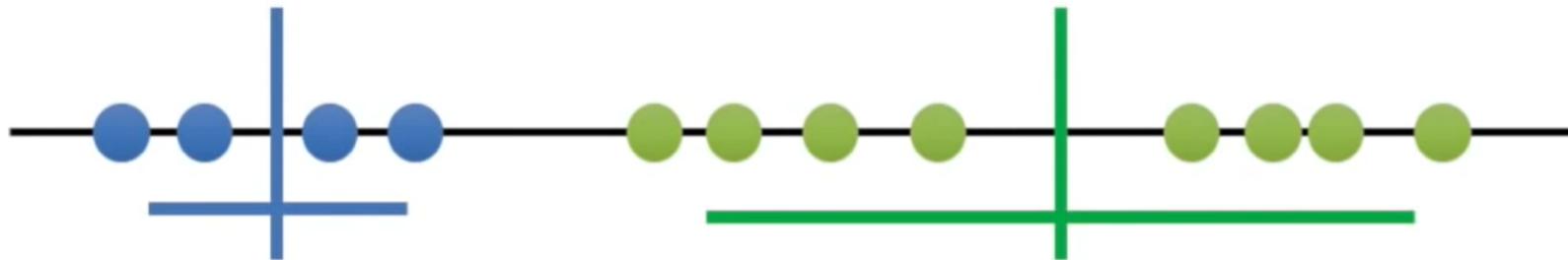
K=1 adalah skenario kasus terburuk.

Kita dapat kuantifikasi masalah skenario buruk ini dengan variasi total

K-Means Clustering

Ilustrasi K-Means

Sekarang dgn $K = 2$



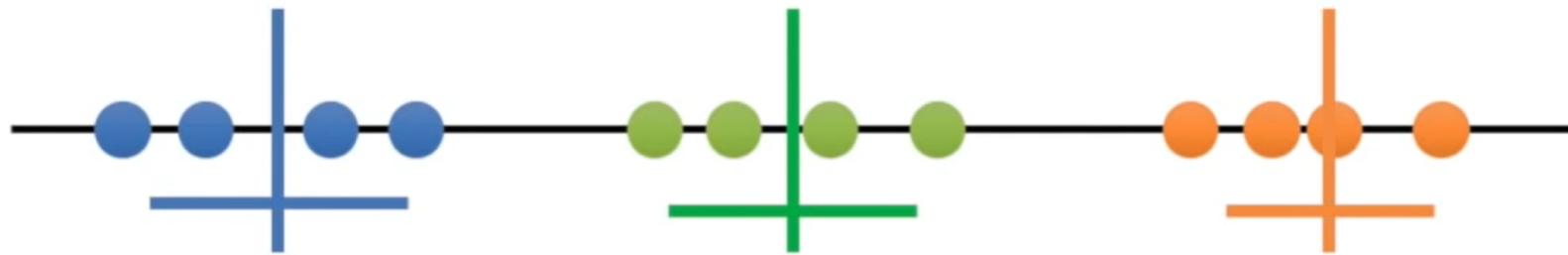
$K = 2$ lebih baik, dan kita dapat kuantifikasi nilai K yg lebih baik dengan membandingkan variasi total dalam 2 kluster terhadap $k = 1$



K-Means Clustering

Ilustrasi K-Means

Sekarang coba dgn $K = 3$



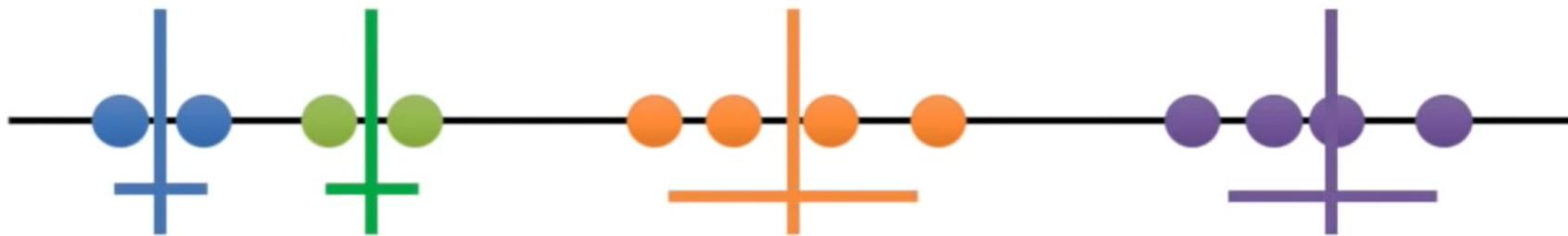
$K = 3$ jadi lebih baik! dan kita dapat kuantifikasi nilai K yg lebih baik dengan membandingkan variasi total dalam 3 cluster terhadap $k = 2$



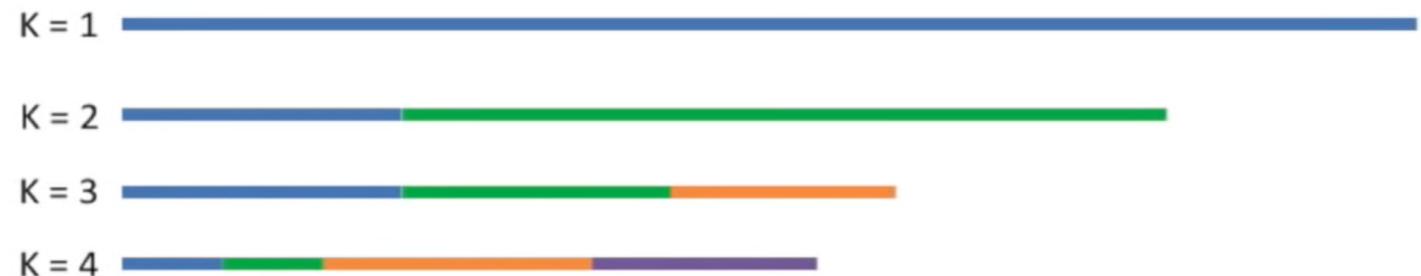
K-Means Clustering

Ilustrasi K-Means

Sekarang coba dgn $K = 4$

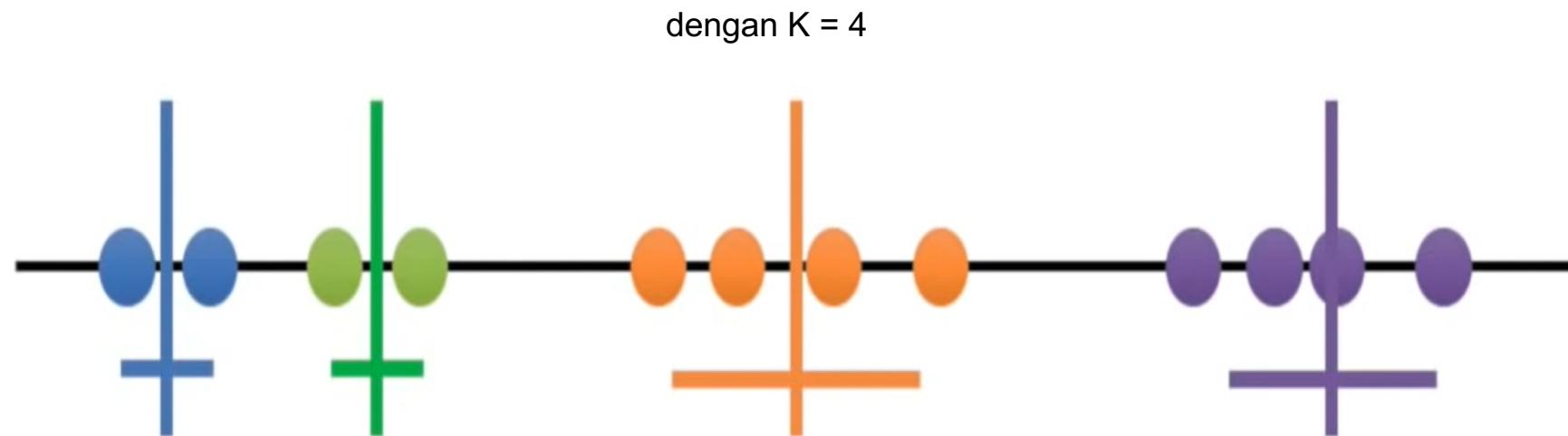


Variasi total dalam setiap cluster adalah menjadi semakin kurang dgn $K = 3$



K-Means Clustering

Ilustrasi K-Means



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

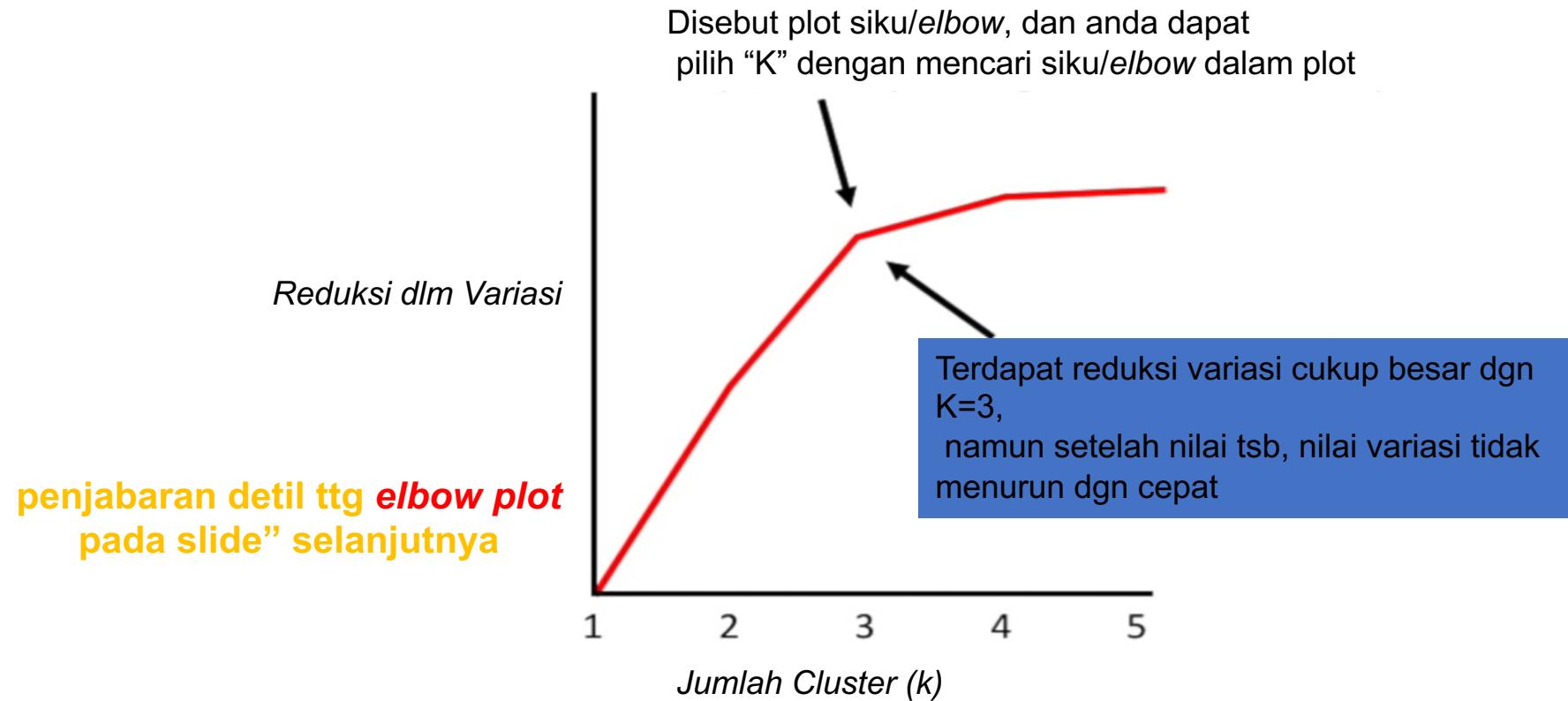
Setiap kita tambahkan cluster baru, variasi total dalam setiap cluster semakin kecil dari sebelumnya.

Dan jika hanya ada 1 titik per cluster, maka variasi = 0

Bagaimanapun, jika kita plot reduksi dalam variansi per nilai utk $K ..$

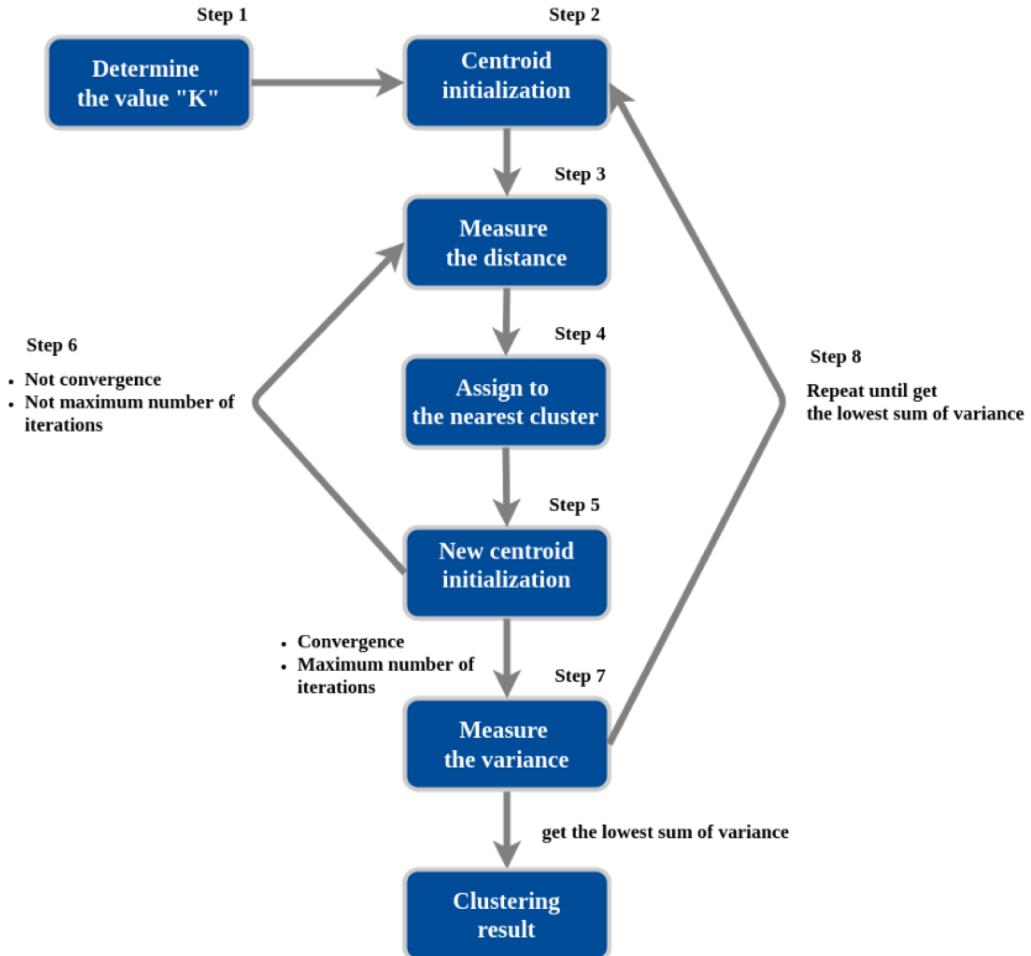
K-Means Clustering

Ilustrasi K-Means



K-Means Clustering

Alur K-Means



K-Means Clustering

Mengimpor library

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

Mengimpor dataset

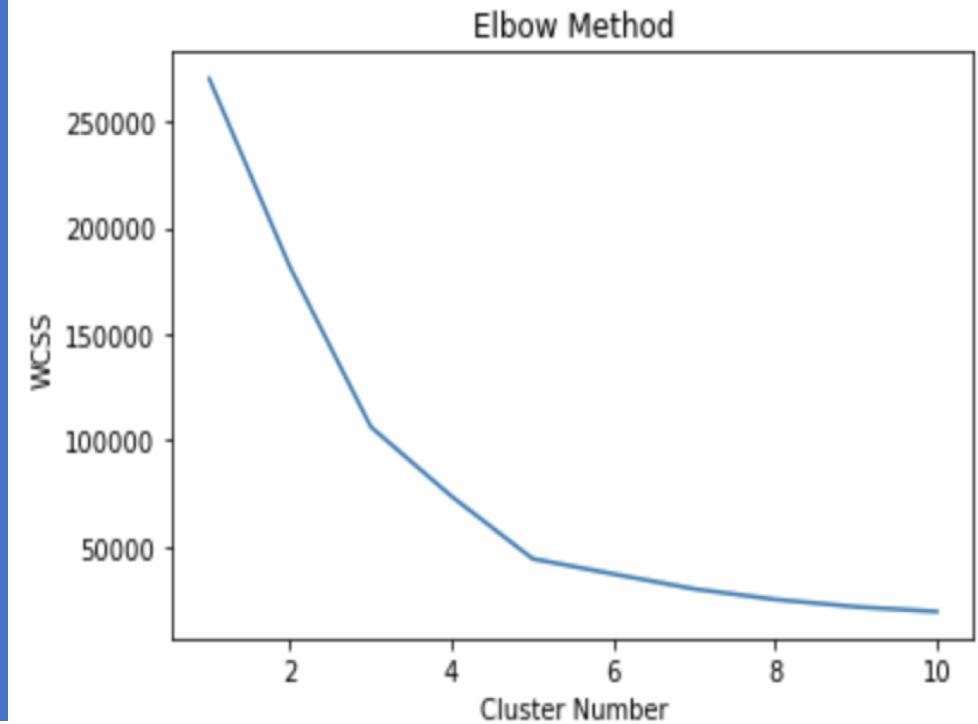
```
dataset = pd.read_csv('Pengunjung_mall.csv')
X = dataset.iloc[:, [3, 4]].values
```

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

K-Means Clustering

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()
```



K-Means Clustering

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)
```

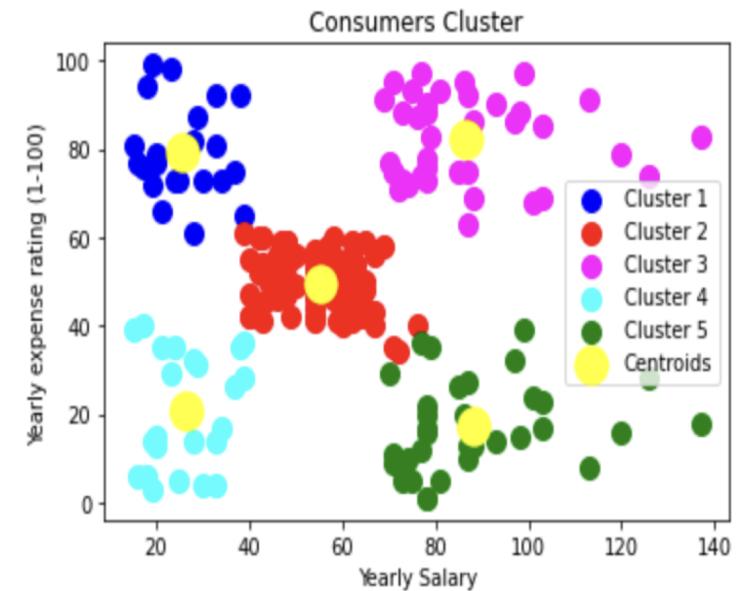
y_kmeans

```
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, 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)
```

K-Means Clustering

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()
```



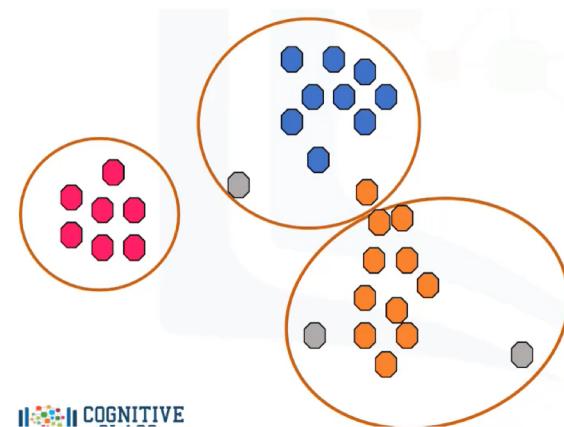
Density-Based Spatial Clustering of Applications with Noise (DBSCAN) adalah algoritma clustering berbasis **density** yang mengelompokkan titik-titik yang memiliki kepadatan tinggi dan menandai titik dengan kepadatan rendah sebagai **outlier**. Noise dalam DBSCAN adalah poin-poin dengan kepadatan/densitas objek yang sedikit. Memiliki prinsip yakni suatu poin dapat dikelompokan/kategori apabila poin tsb berada dekat dengan poin-poin dalam kategori tsb

Cara Kerja DBSCAN

1. Pilih titik data secara acak.
2. Jika titik memiliki **cukup banyak tetangga** ($\geq \text{MinPts}$ dalam radius ϵ), maka titik menjadi **core point** dan membentuk cluster.
3. Semua tetangga dalam radius ϵ ditambahkan ke dalam cluster.
4. Jika titik tidak memiliki cukup tetangga, titik dianggap **outlier (noise)**.
5. Proses berulang hingga semua titik telah dikunjungi.

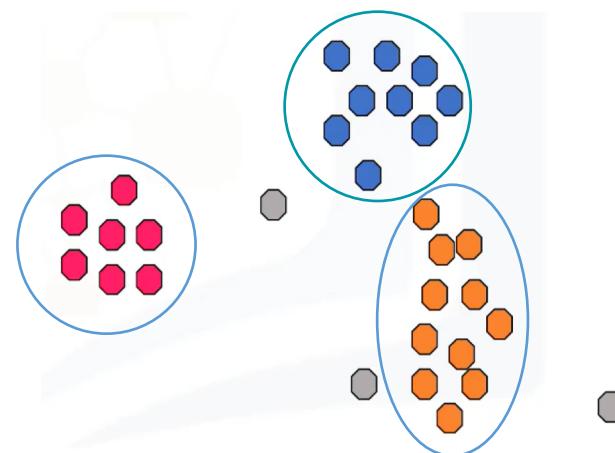
K-Means

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



Algoritma DBSCAN menggunakan dua parameter yaitu:

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

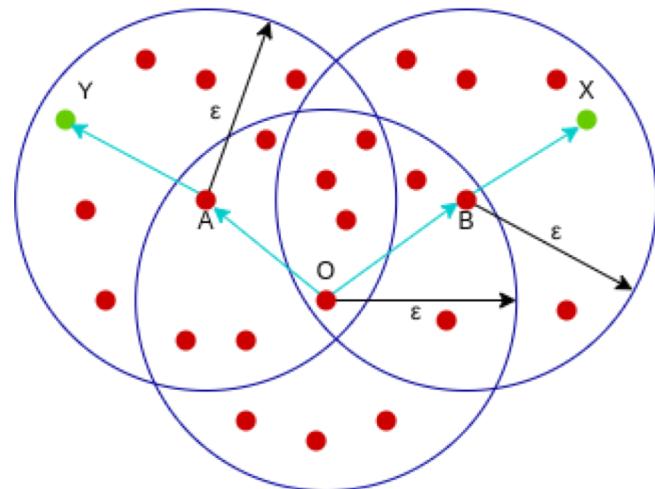
eps (ϵ) atau Radius: Ukuran jarak radius yang akan digunakan untuk menemukan titik-titik di sekitar titik mana pun.

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

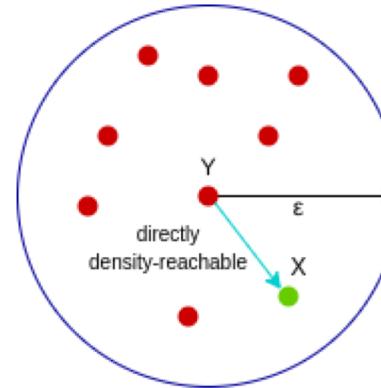
Reachability pada konsep ini, untuk menentukan kepadatan dialukan dengan menetapkan suatu titik yang dapat dijangkau dari yang lain jika terletak dalam jarak tertentu (eps) darinya.

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 dan 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 epsilon/R dan minPoints jika ada titik O sedemikian rupa sehingga X dan Y dapat dijangkau kerapatan dari O w.r.t ke epsilon dan minPoints.

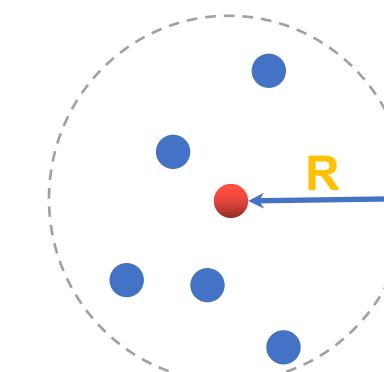
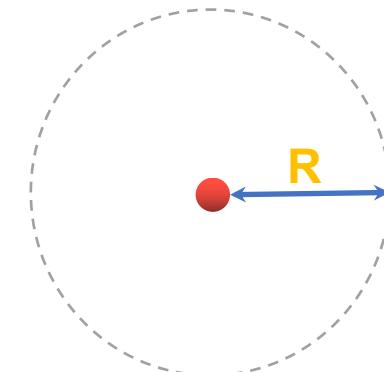
Parameter DBSCAN

Radius (jari-jari utk area terdekat)

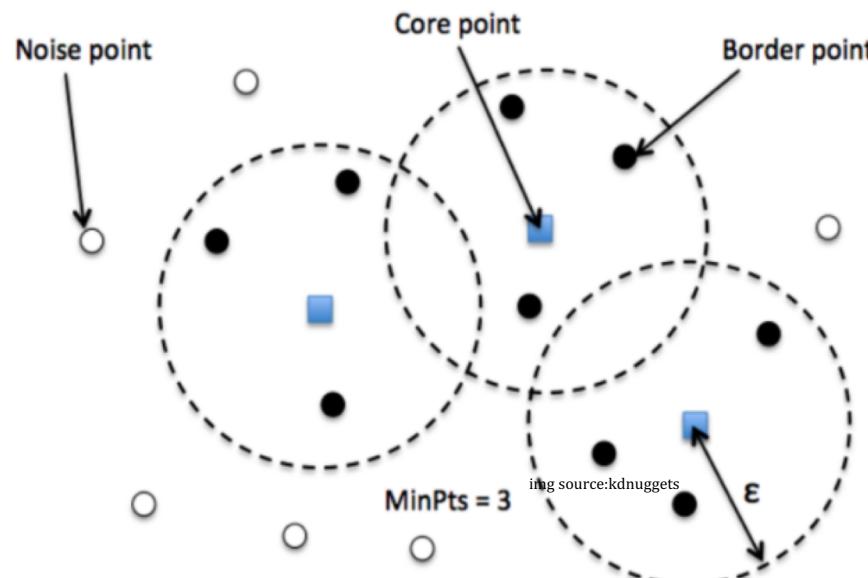
Jika suatu poin masih dalam jangkauan radius (R) dan memiliki sejumlah poin lain dalam area tsb, maka area ini dikategorikan sebagai area (dengan densitas) padat.

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



Terdapat tiga jenis titik setelah pengelompokan DBSCAN selesai:



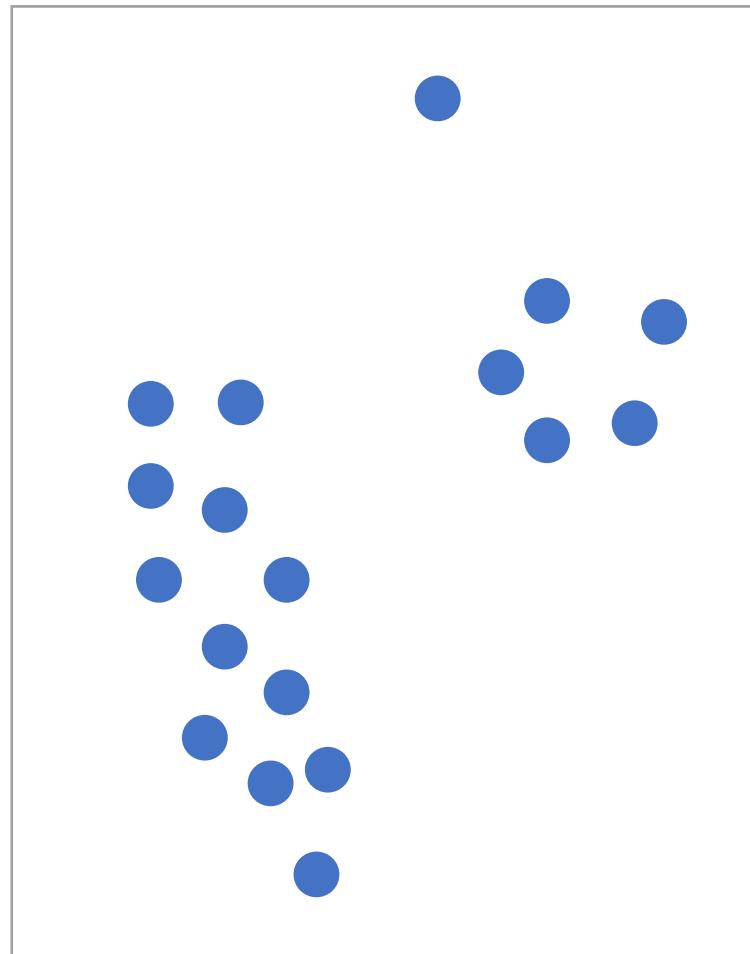
Core adalah titik yang memiliki setidaknya m titik dalam jarak (radius/epsilon) n dari dirinya sendiri.

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

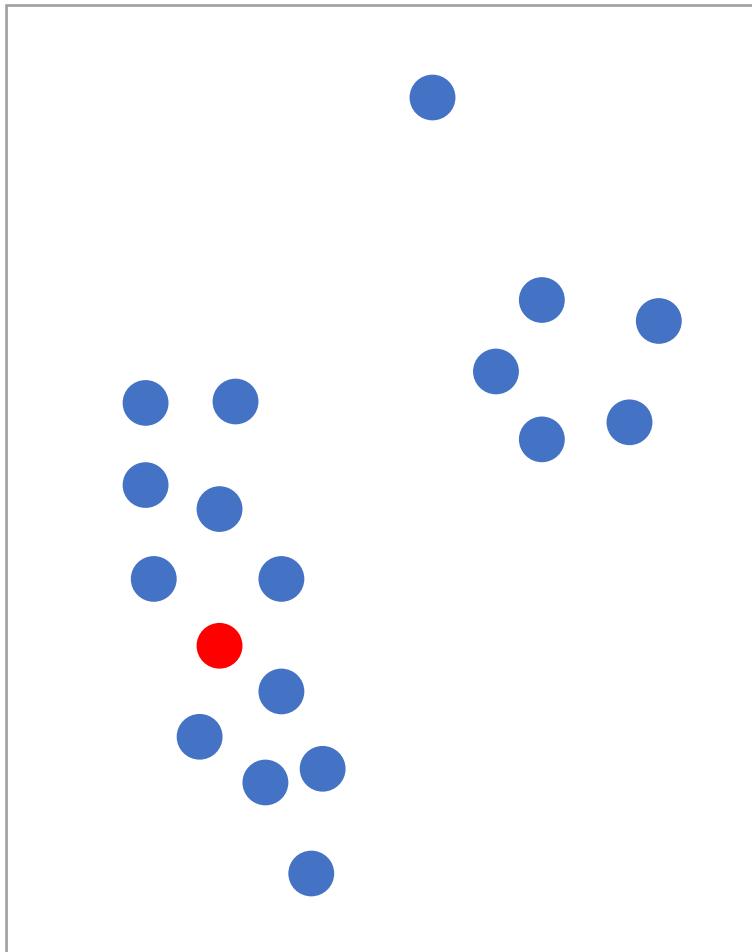
Ditentukan parameter: $R/\text{epsilon} = 2$; dan $\text{MinPoints}/M = 5$



Ilustrasi DBSCAN

parameter:

R/ϵ = 2; dan M = 5

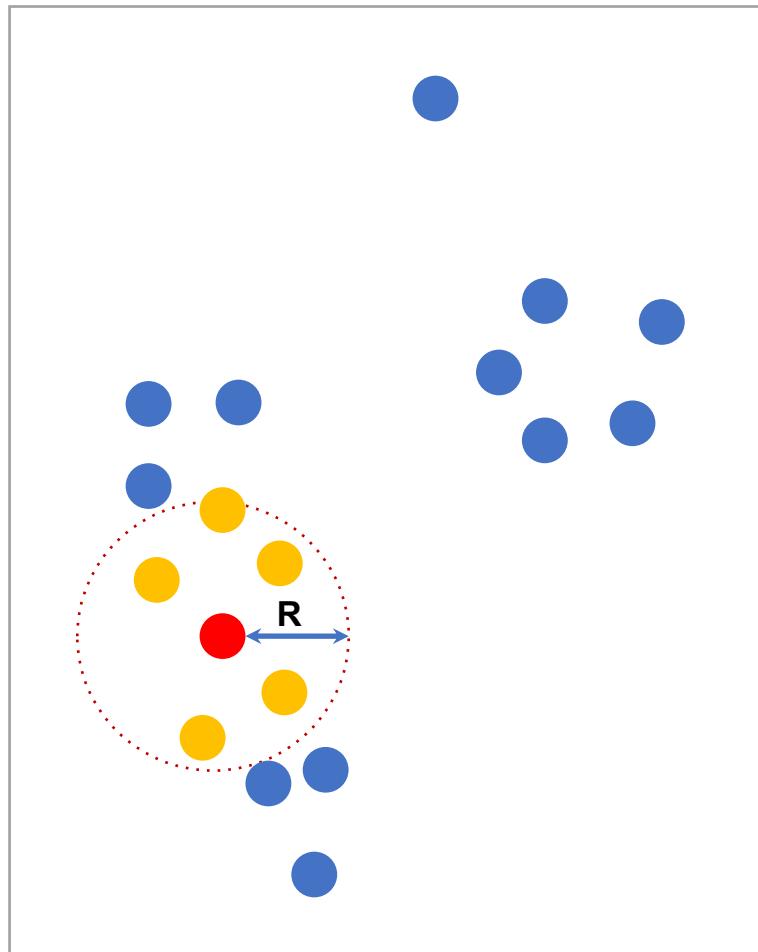


Langkah 1:
Pilih satu poin secara acak

Ilustrasi DBSCAN

parameter:

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



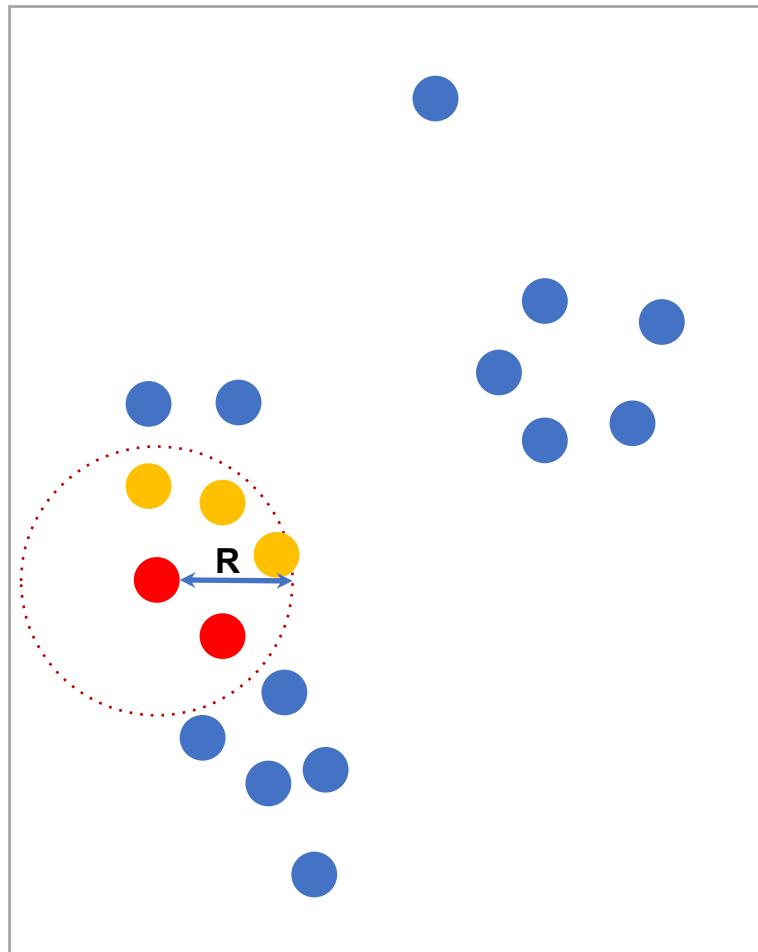
Langkah 2: Kategorikan poin

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

Ilustrasi DBSCAN

parameter:

R/ϵ = 2; dan M = 5



Langkah 3:

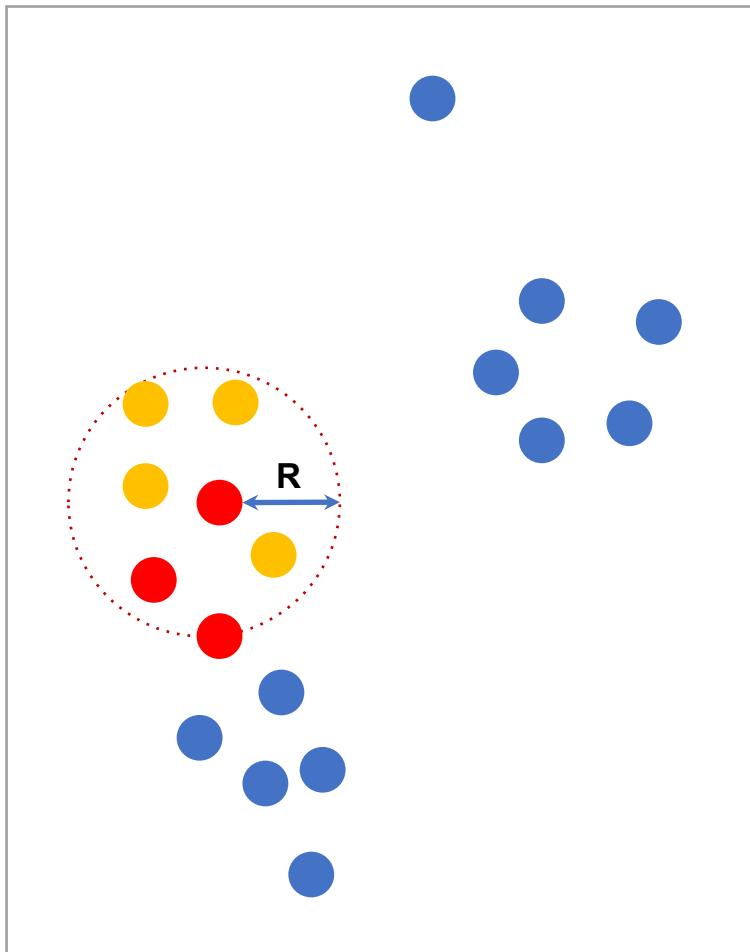
Ulangi langkah 1 dan 2 hingga semua poin telah dikategorikan

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 3:

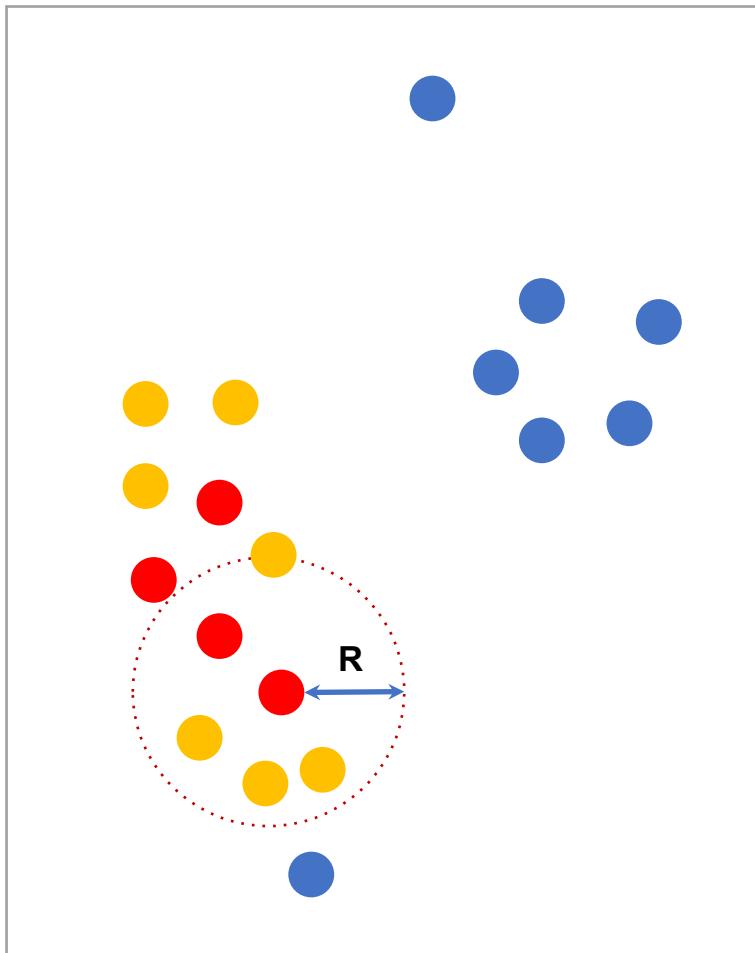
Ulangi langkah 1 dan 2 hingga semua poin telah dikategorikan

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

Ilustrasi DBSCAN

parameter:

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



Langkah 3:

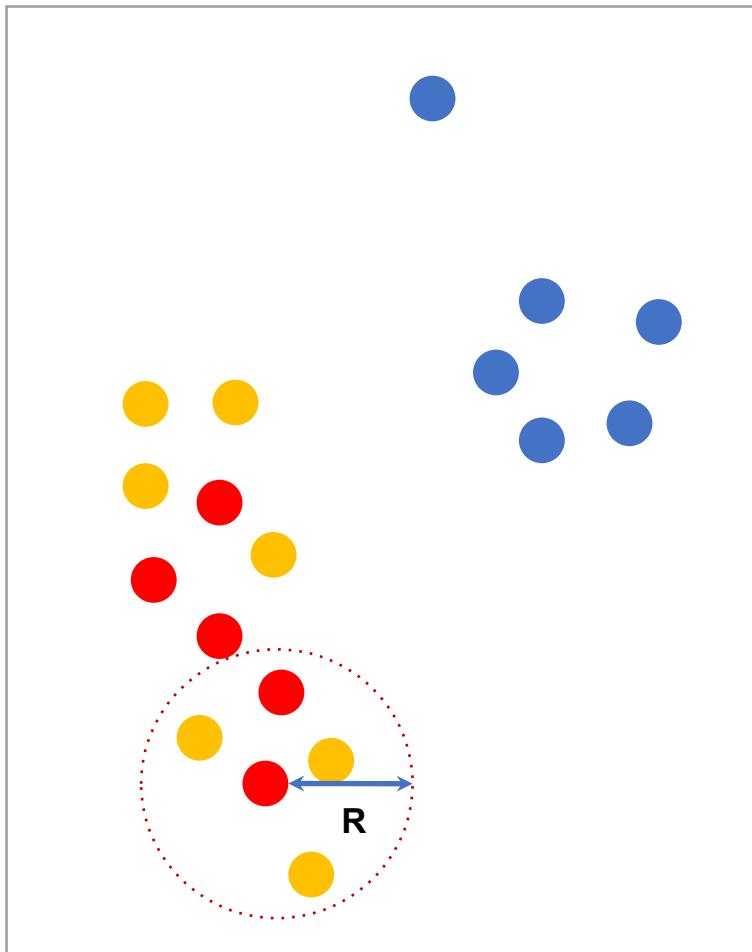
Ulangi langkah 1 dan 2 hingga semua poin telah dikategorikan

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

Ilustrasi DBSCAN

parameter:

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



Langkah 3:

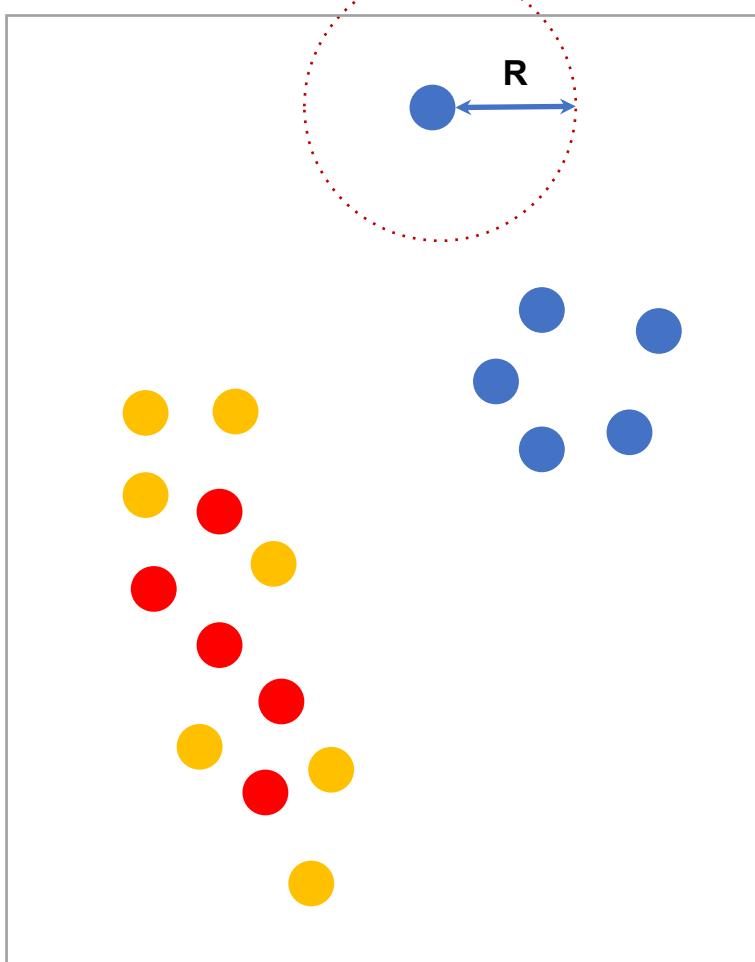
Ulangi langkah 1 dan 2 hingga semua poin telah dikategorikan

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

Ilustrasi DBSCAN

parameter:

R/epsilon = 2; dan M = 5



Langkah 3:

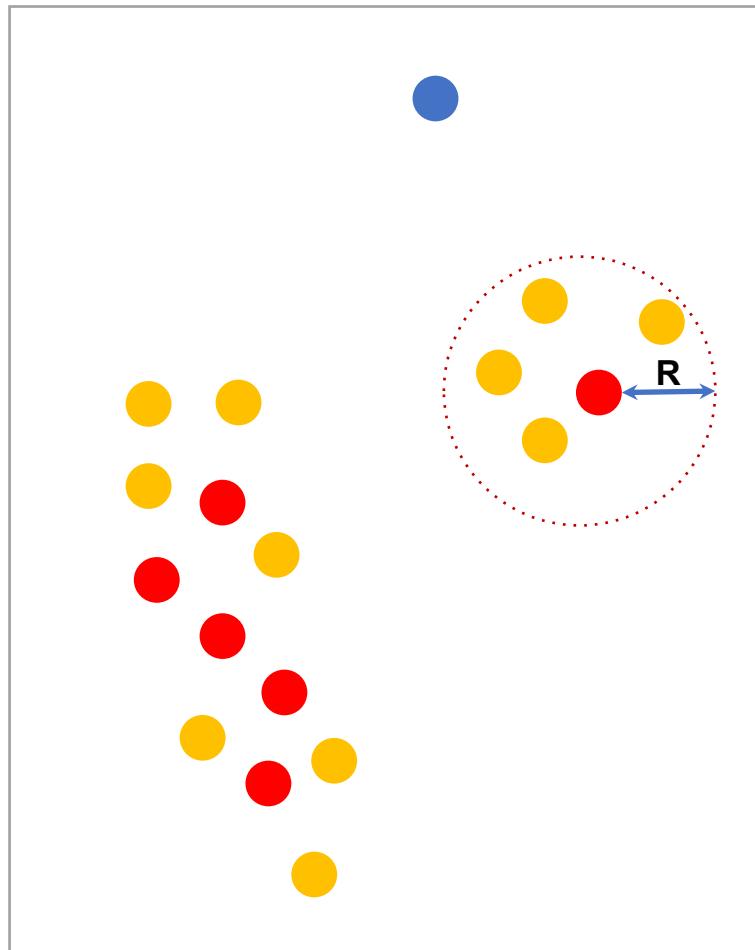
Ulangi langkah 1 dan 2 hingga semua poin telah dikategorikan

- *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:

$R/\epsilon = 2$; dan $M = 5$



Langkah 3:

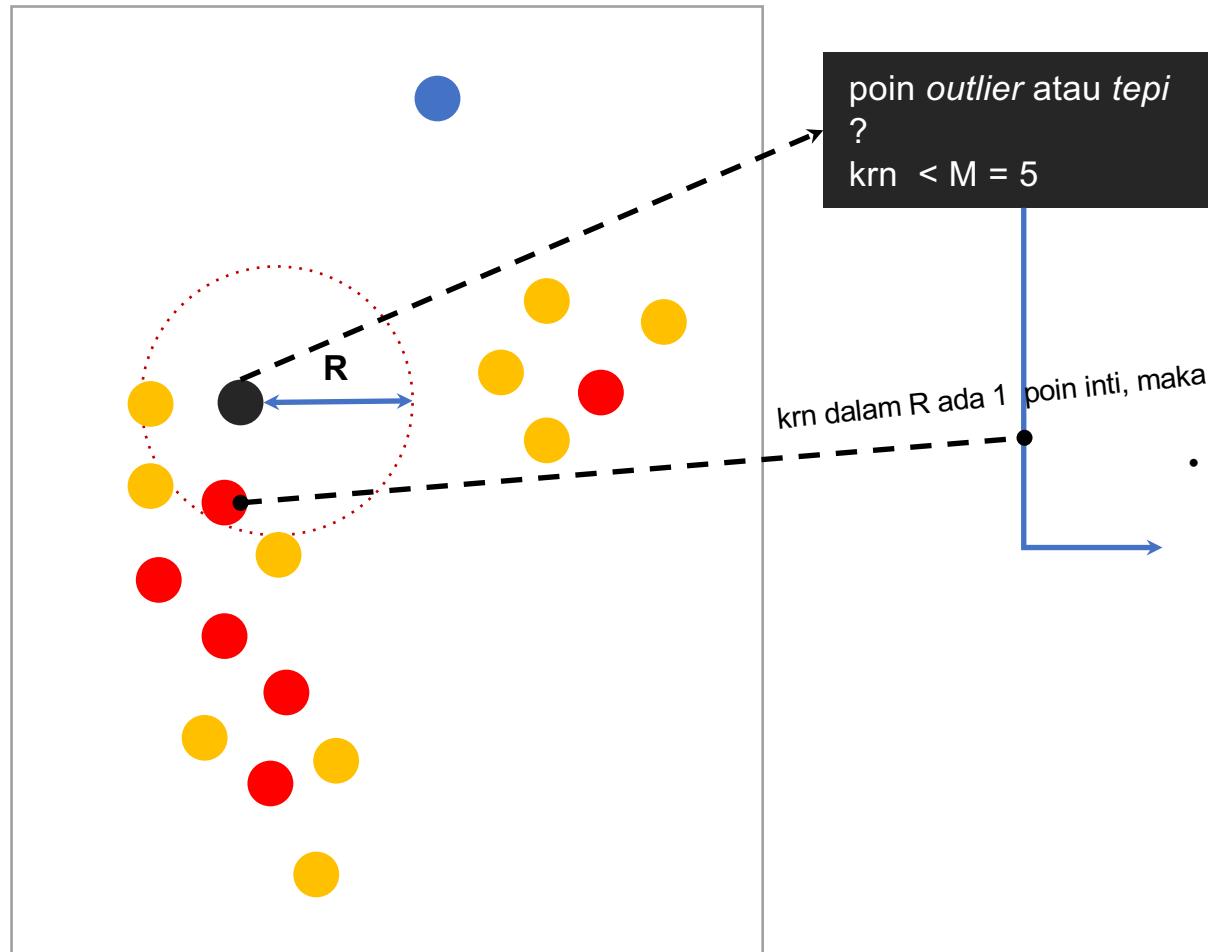
Ulangi langkah 1 dan 2 hingga semua poin telah dikategorikan

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

Ilustrasi DBSCAN

parameter:

R/epsilon = 2; dan M = 5



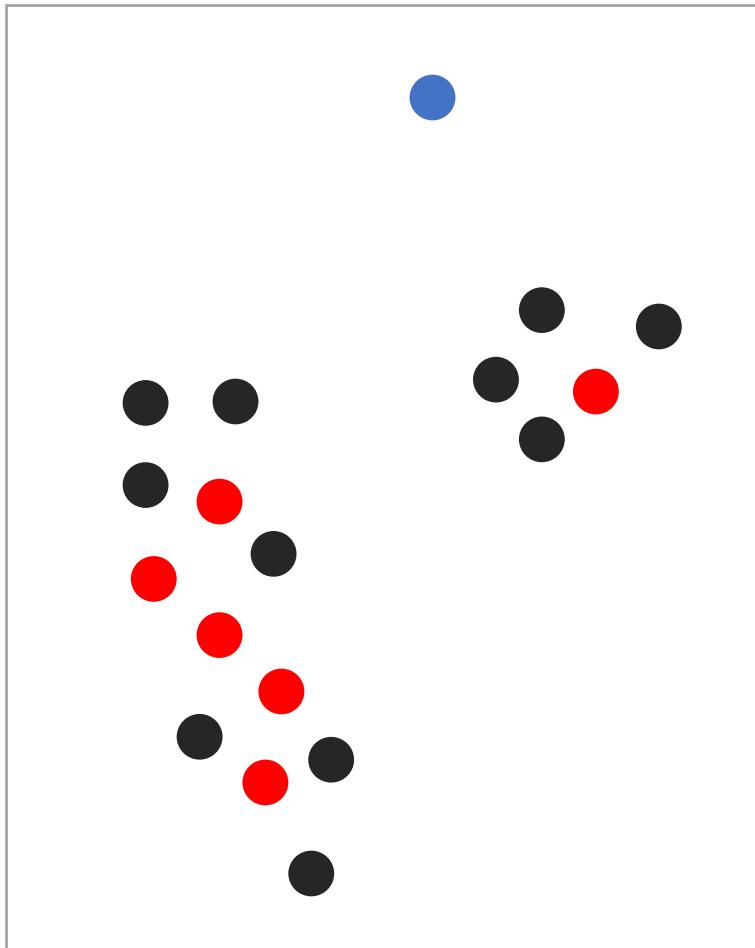
Langkah 3:
Ulangi langkah 1 dan 2
hingga semua poin telah
dikategorikan

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 3:

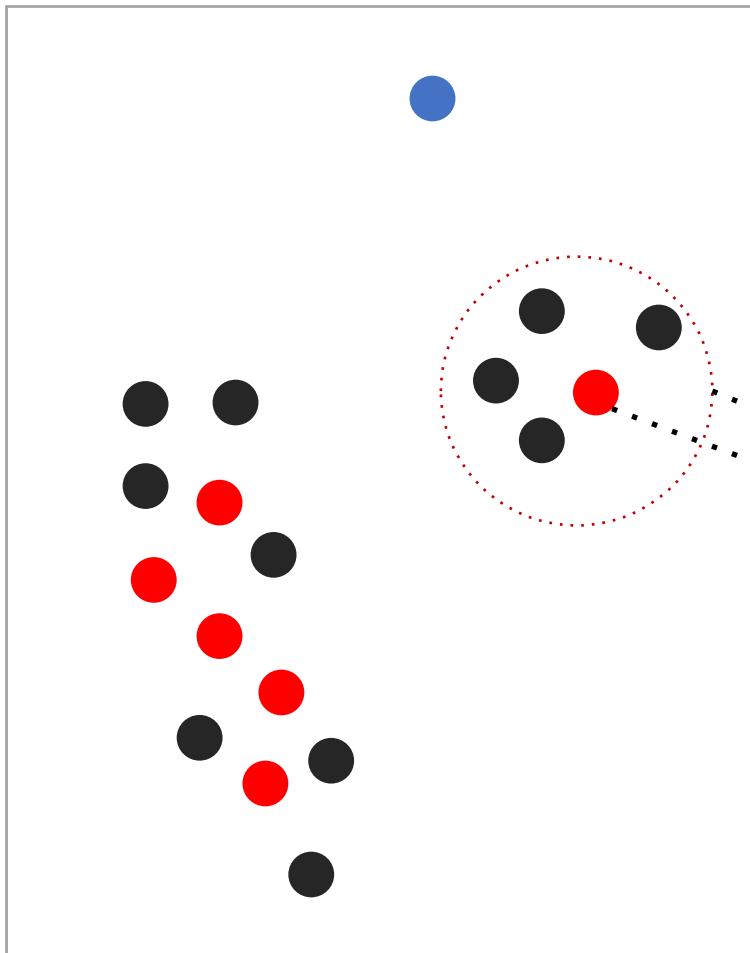
Ulangi langkah 1 dan 2 hingga semua poin telah dikategorikan

- *Seluruh poin telah dikategorikan tanpa terlewat satupun*

Ilustrasi DBSCAN

parameter:

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



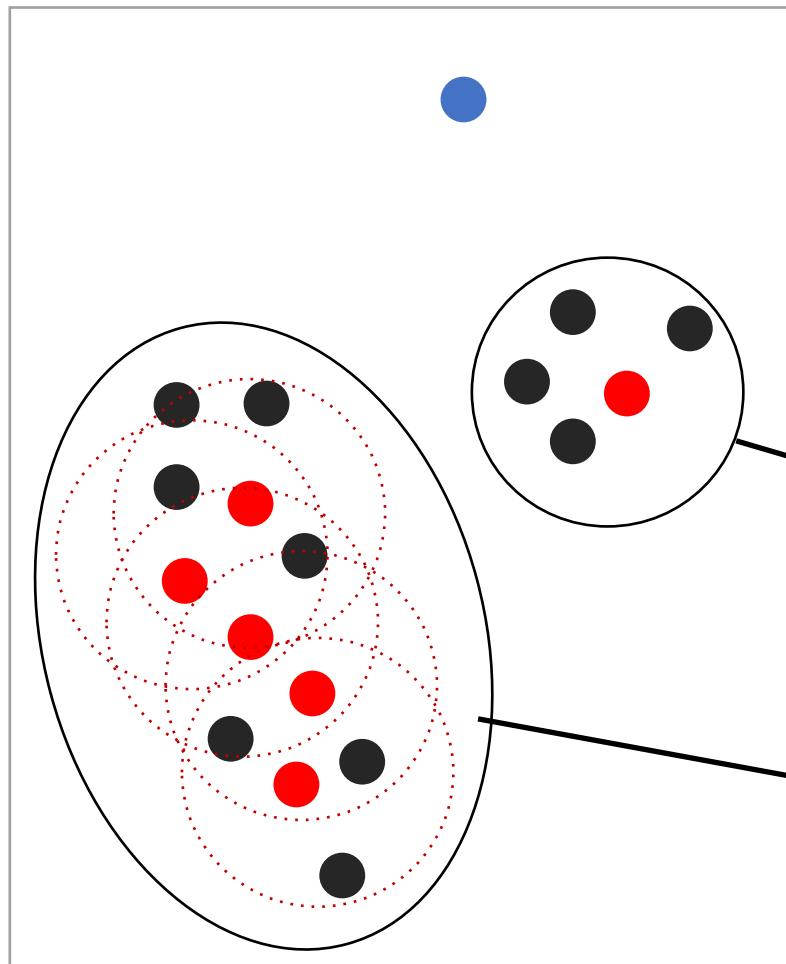
Langkah 4: Menentukan jumlah cluster

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/epsilon = 2; dan M = 5



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” datar
- 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)