

DATA SCIENCE and BUSINESS INTELLIGENT

Author: Egi Safitri

Meeting 17



1. CUSTOMER SEGMENTS

Classification

Learning Objective

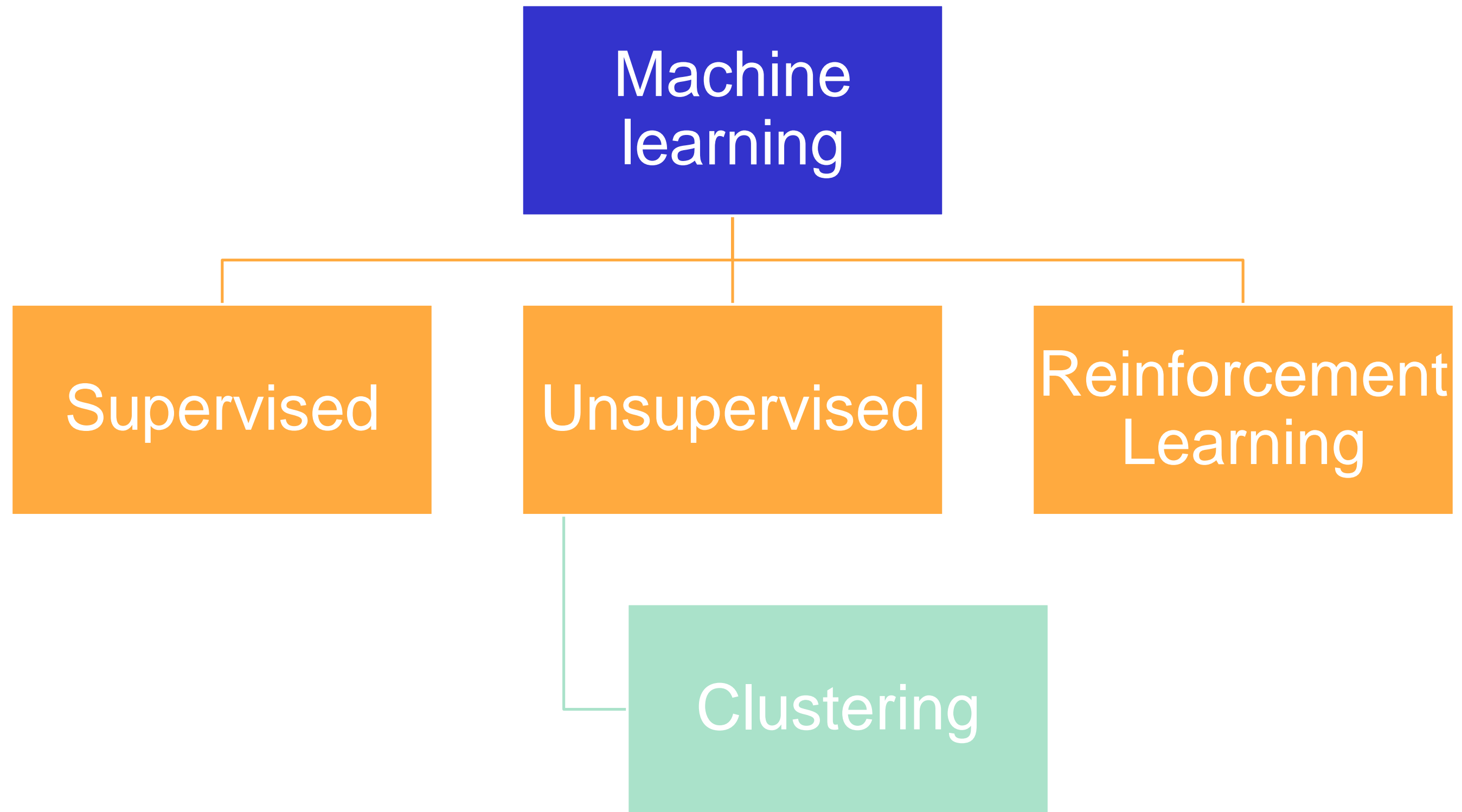
Peserta mempelajari pengertian, cara menyiapkan, dan cara implementasi model regresi dengan algoritma:

- A. Melakukan pemilahan 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 Tools menggunakan Rapid Miner

Clustering



Clustering



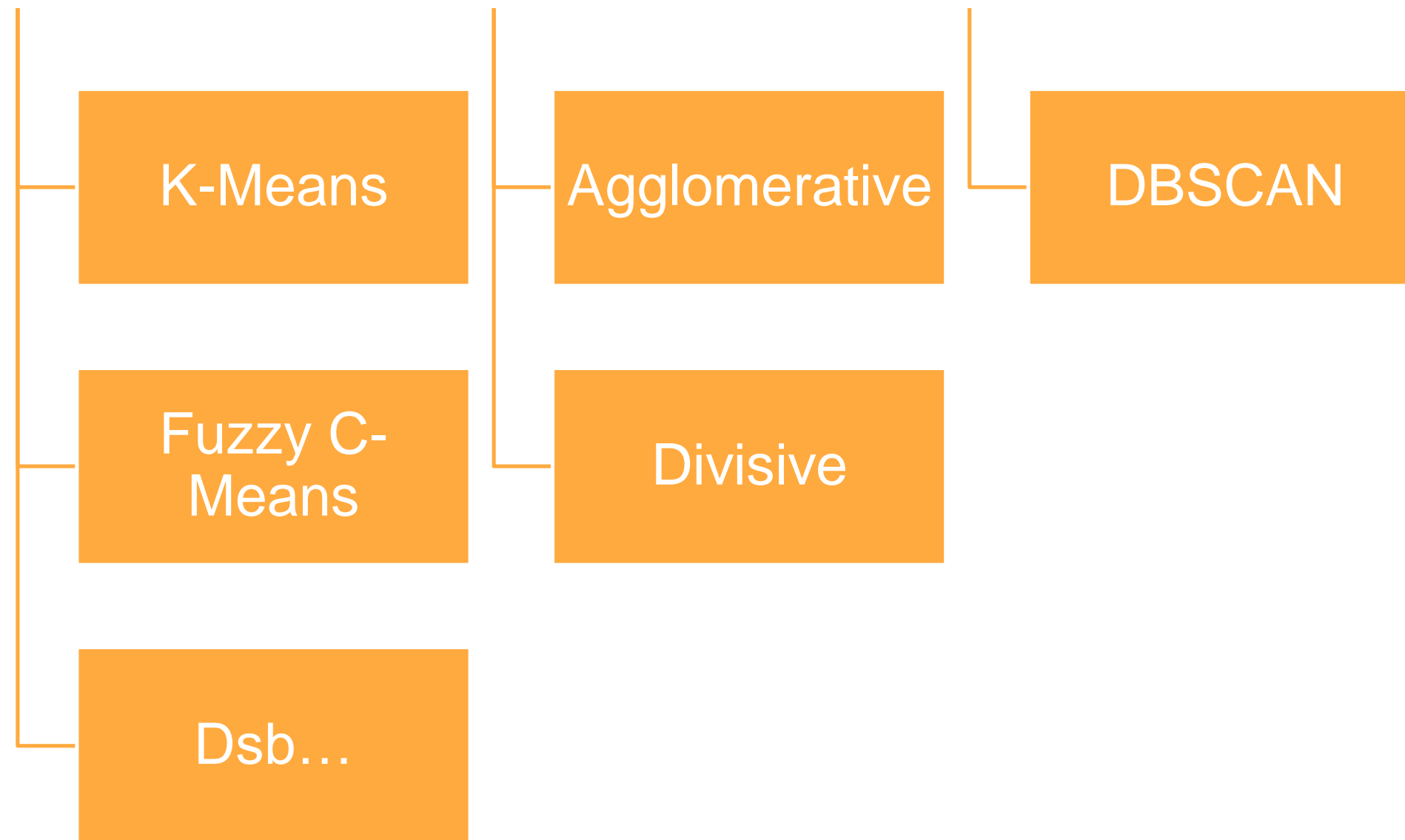
Clustering

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.

Clustering

Clustering



K-Means

Algoritma Kmeans adalah 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 mencoba membuat titik data intra-cluster semirip mungkin sambil dengan titik data yang lain pada satu cluster.

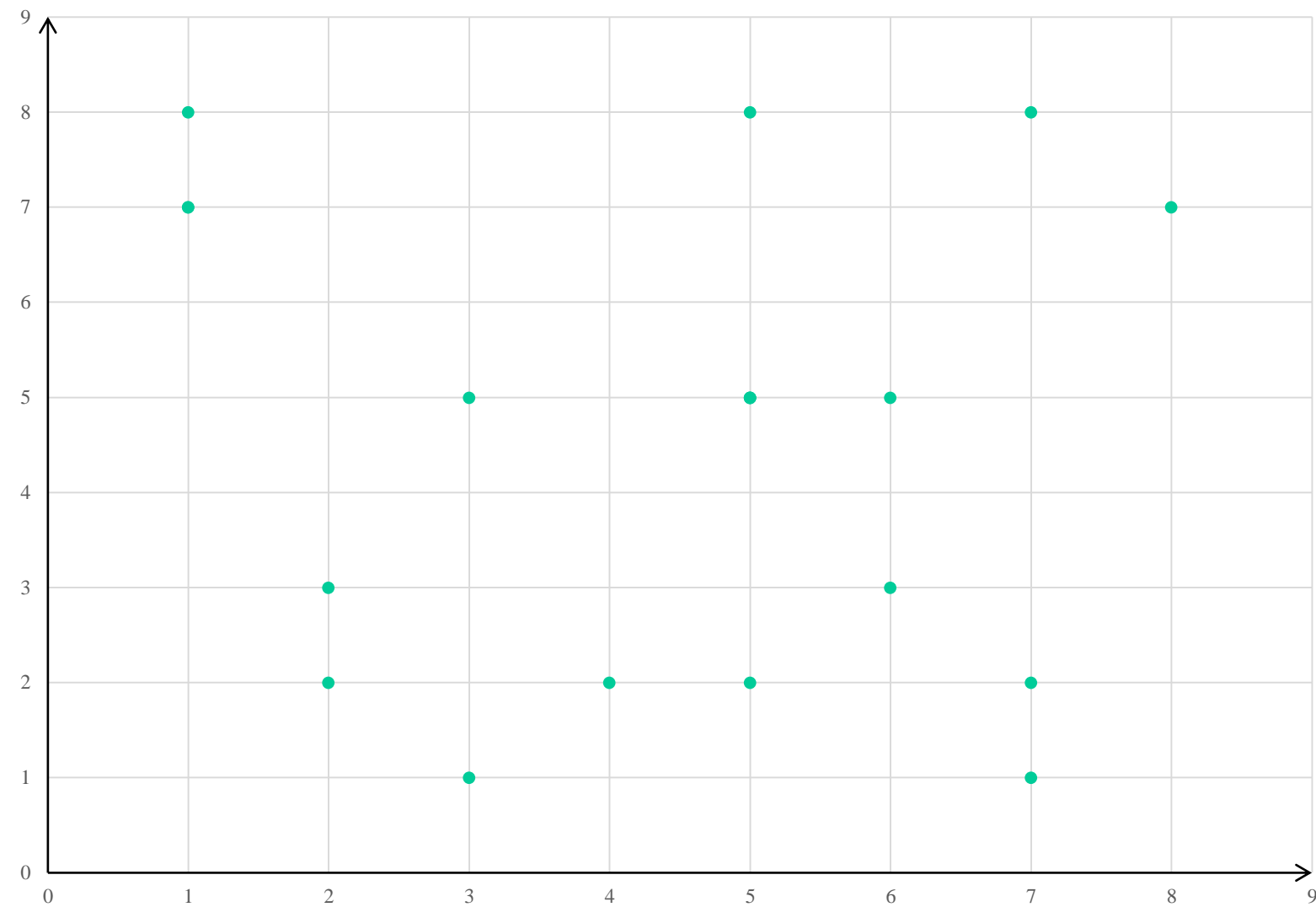
K-Means menetapkan poin data ke cluster sedemikian rupa sehingga jumlah jarak kuadrat antara titik data dan pusat massa cluster (rata-rata aritmatika dari semua titik data yang termasuk dalam cluster itu) minimal.

Semakin sedikit variasi yang kita miliki dalam cluster, semakin homogen (serupa) titik data dalam cluster yang sama.

Langkah-langkah metode K-Means

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

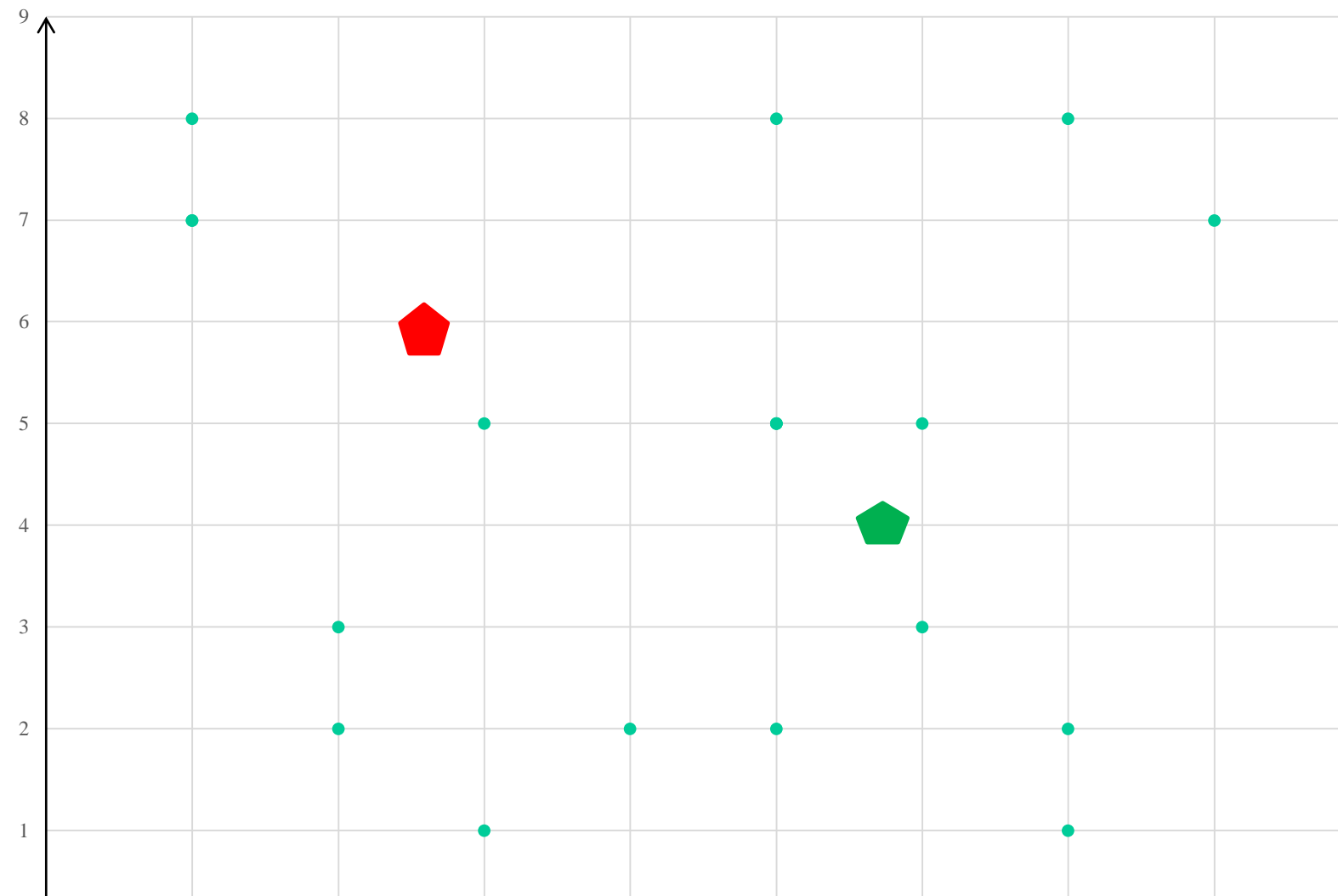
Sebagai contoh terdapat data 2 dimensi seperti yang ditampilkan dalam grafik, langkah pertama adalah memilih jumlah kluster. Misal kita pilih untuk membaginya ke dalam 2 kluster.



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*).

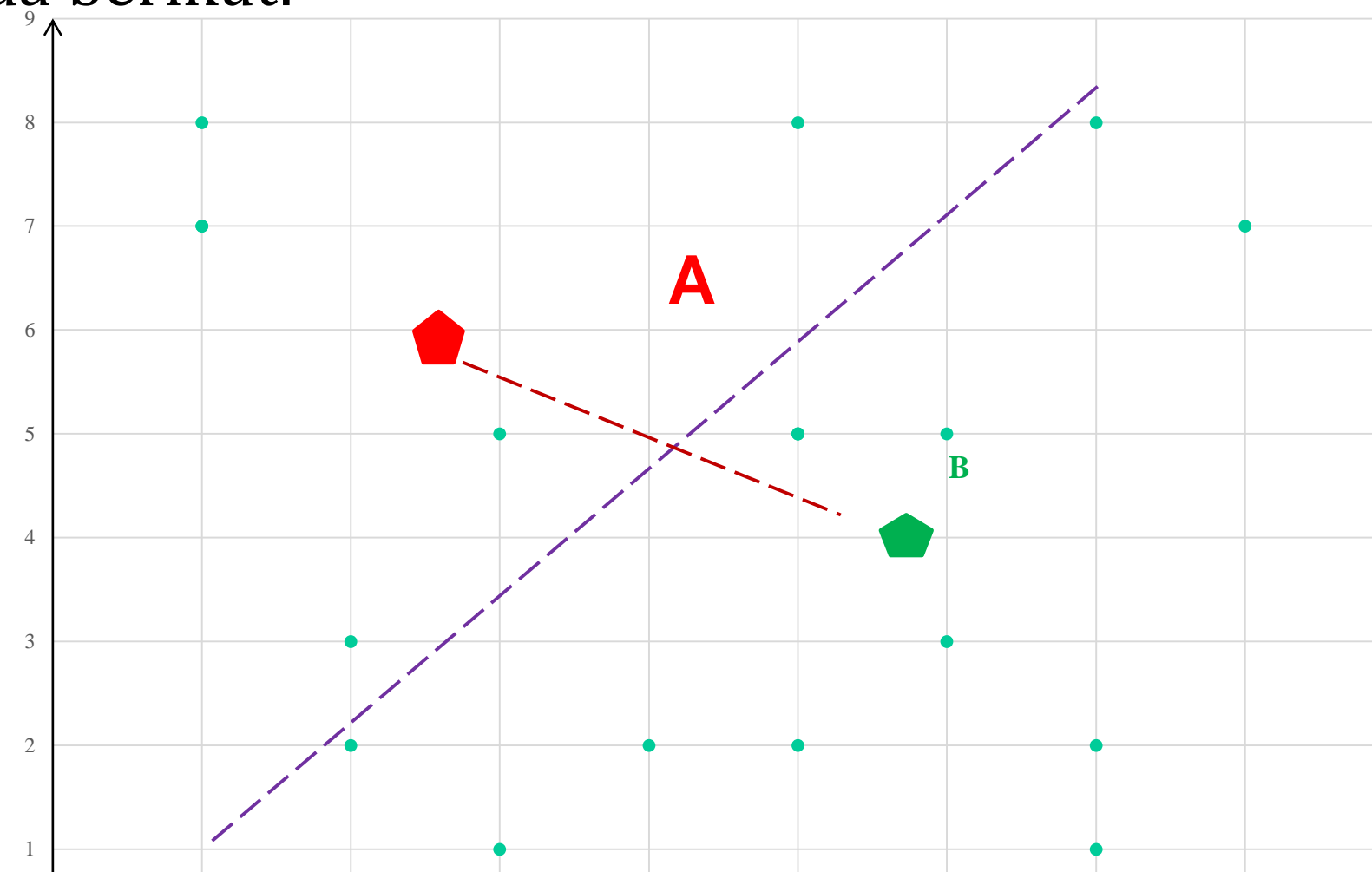
Langkah kedua adalah menentukan titik pusatnya. Dalam satu kluster terdapat satu titik pusat atau yang disebut dengan *centroid*. Penentuan awal posisi titik pusat ini bebas, karena nantinya algoritma K-Means akan merubah posisi tiap titik hingga dicapai solusi paling optimal. Pada Gambar 2, titik merah 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

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.

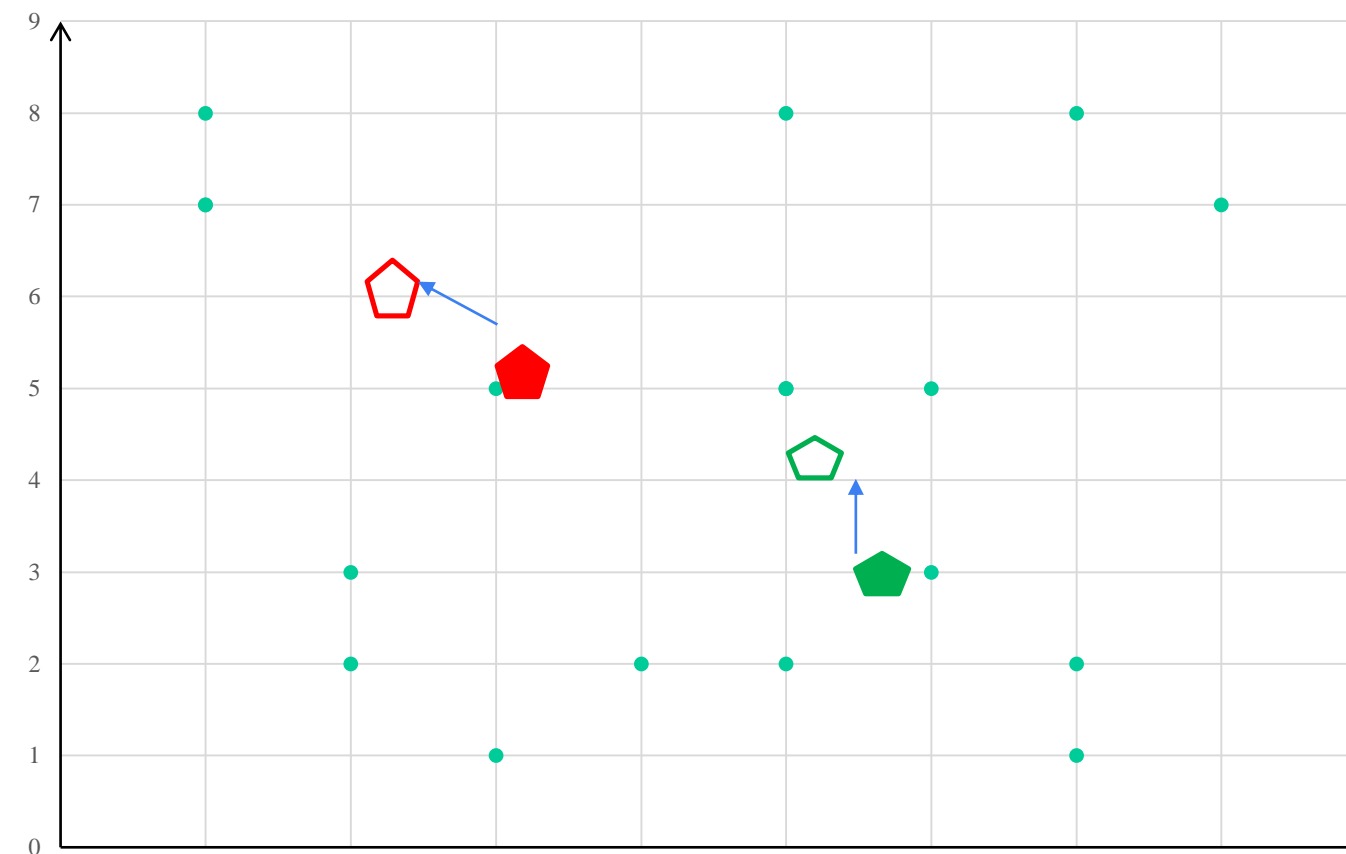
Berdasarkan pengelompokan pada langkah ke dua, maka setiap titik data saat telah tergabung dalam salah satu kluster. Titik data yang diwakili dengan symbol 'x' berwarna merah masuk ke kluster 1, dan symbol 'x' berwarna biru masuk ke kluster 2, seperti pada berikut:



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 kluster.

Langkah ke-empat adalah melakukan penghitungan sesuai algoritma K-Means, yaitu mencari posisi titik pusat yang paling sesuai untuk setiap klusternya berdasarkan penghitungan jarak terdekat. Penghitungan jarak masing-masing titik data ke pusat kluster dapat menggunakan metode Euclidean distance, ilustrasi penghitungan jarak dapat dilihat pada Gambar 4



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 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 pengemlompokan akan menghasilkan analisa 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 analisa 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 kluster:

$$\begin{aligned} & \text{WCSS} \\ &= \sum_{P_i \text{ in cluster 1}} \text{jarak } (P_i, C_1)^2 + \sum_{P_i \text{ in cluster 2}} \text{jarak } (P_i, C_2)^2 \end{aligned}$$

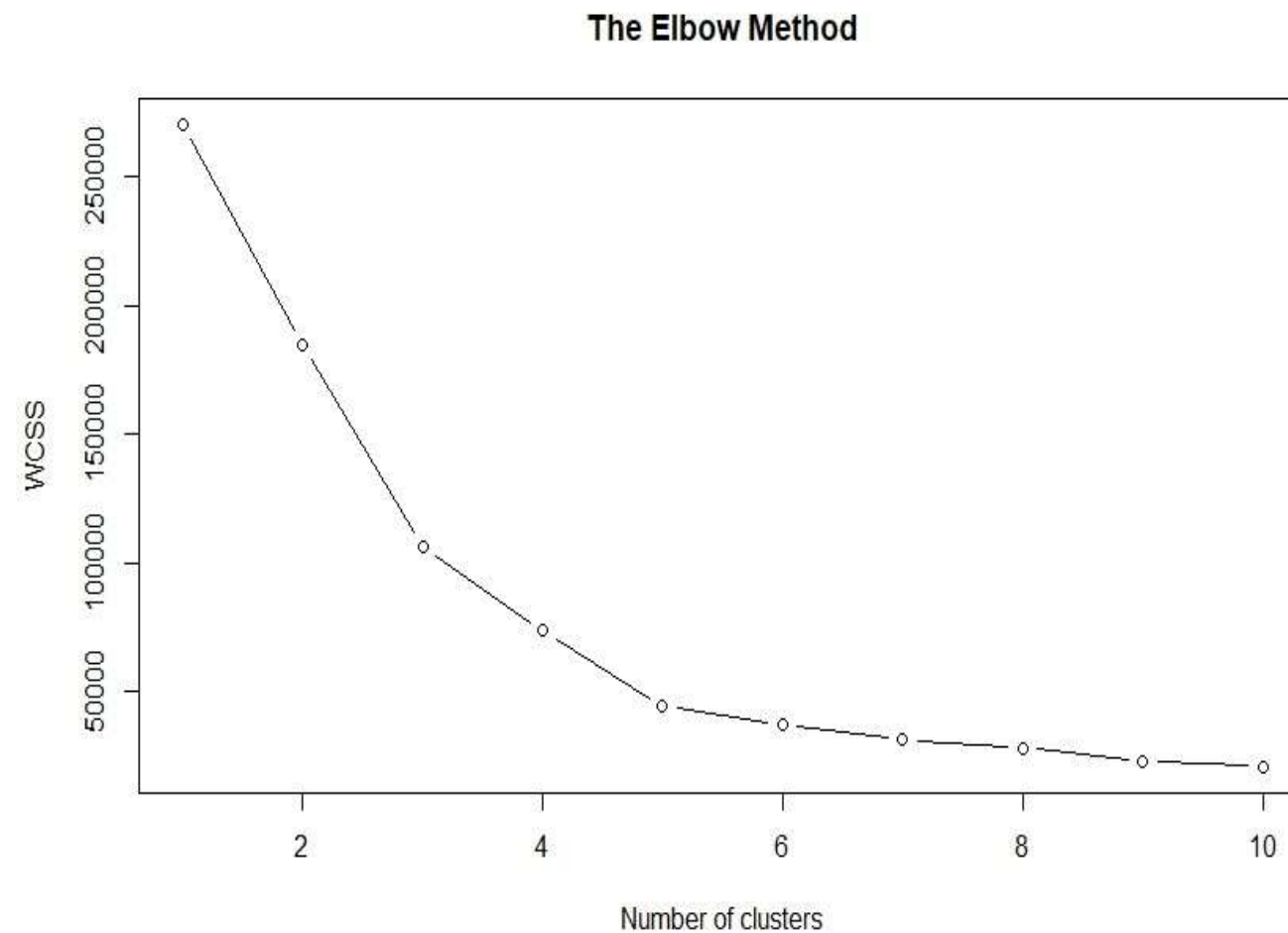
Optimasi K-Means

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 klaster dua, dan seterusnya sesuai jumlah klaster 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 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 klaster yang tepat untuk grafik di atas adalah 5. Contoh hasil perhitungan WCSS dapat dilihat pada grafik berikut:



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

Contoh coding menggunakan
Bahasa python:

```
# 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[:, [4, 5]].values

# 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

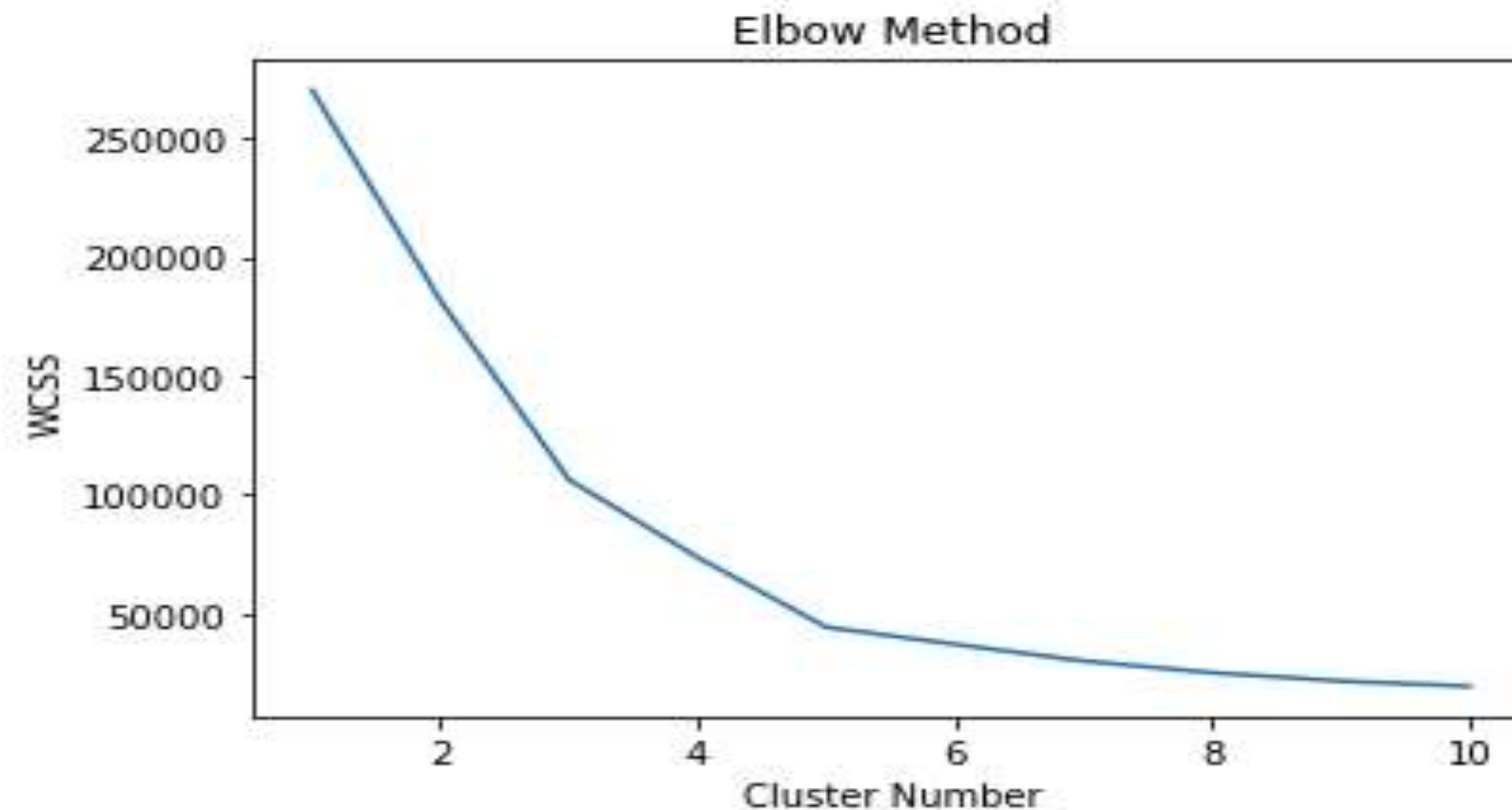
Contoh coding
menggunakan Bahasa
python 2:

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

# 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, 0], kmeans.cluster_centers_[0, 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

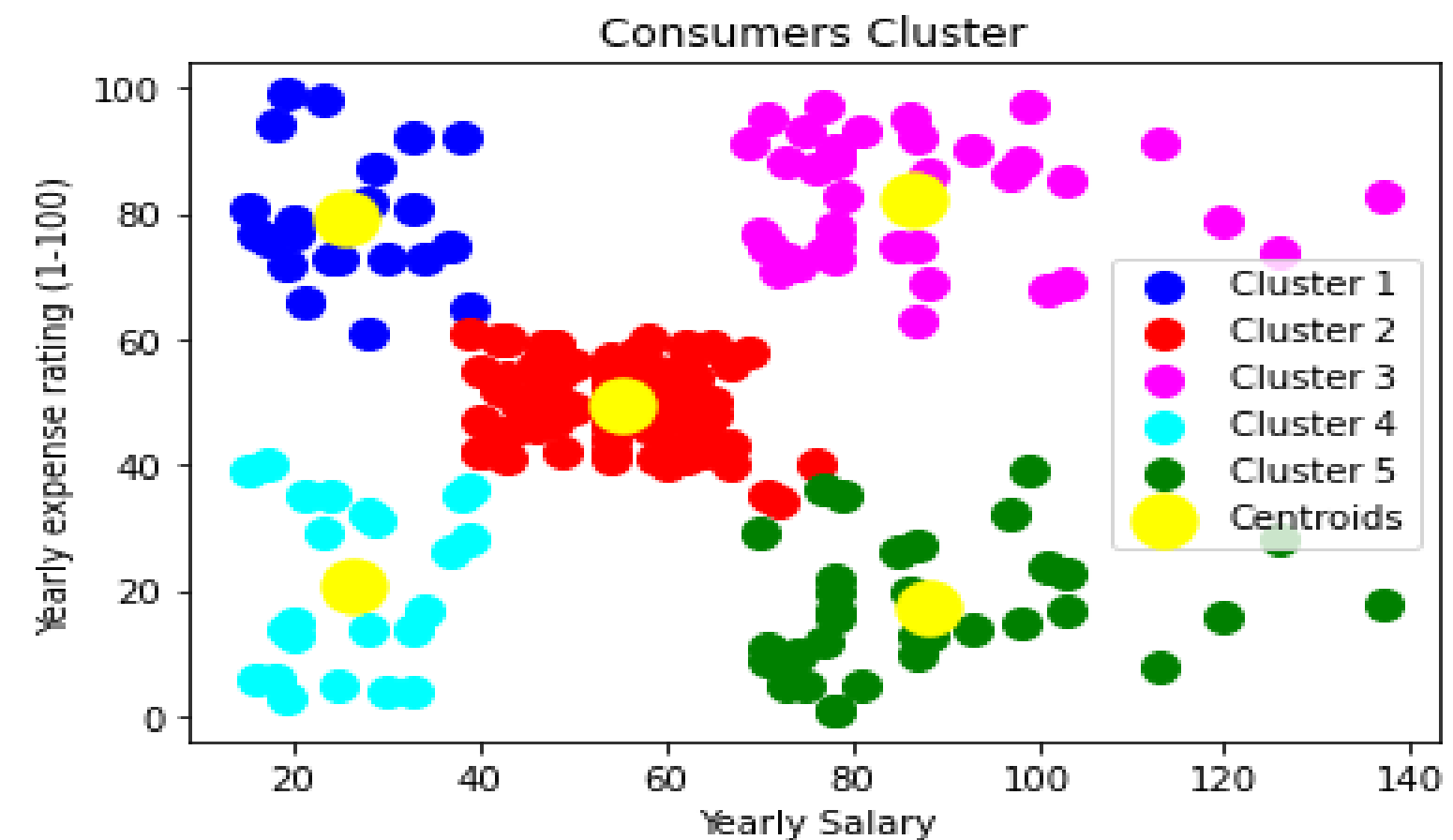
Hasil pengelompokan data menggunakan metode K-Means yang dioptimasi dengan metrics WCSS:



Pada Gambar di atas, dapat diamati bahwa garis grafik yang berbentuk siku terdapat pada kluster 3 dan 5, maka dapat disimpulkan bahwa pembagian kluster 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:

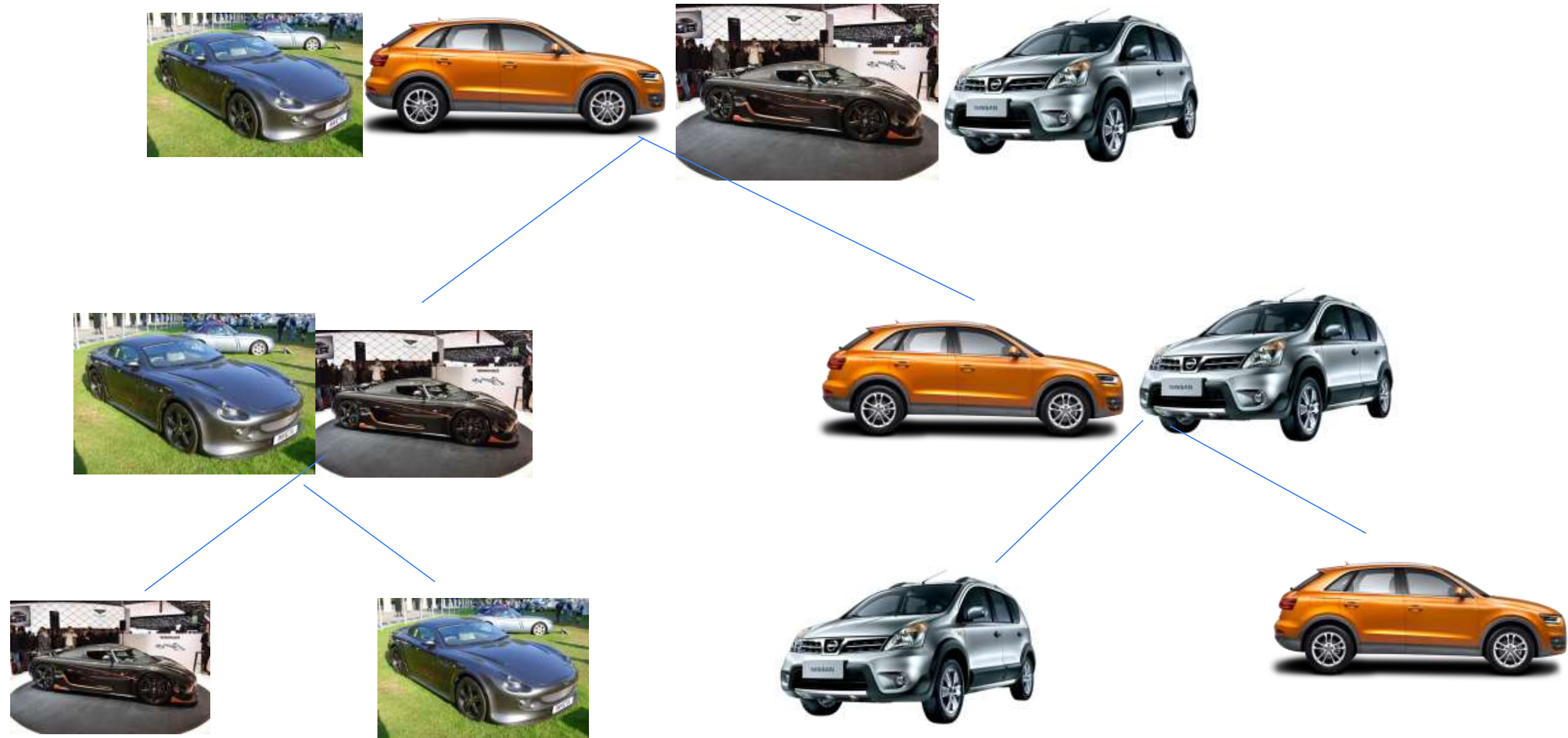


Pada dua gambar di atas dapat diamati hasil pengelompokan data set yang terbagi dalam 3 dan 5 klaster. Hasil dari pengelompokan tersebut dapat digunakan sebagai dukungan keputusan untuk menentukan **strategi** yang dituju oleh pemilik toko terhadap pelanggannya

Hierarchical CLustering

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.

Pertama, terdapat empat mobil yang dapat dimasukkan ke dalam dua kelompok jenis mobil: sedan dan SUV. Selanjutnya, HC akan menggabungkan sedan dan SUV. Untuk langkah terakhir, yaitu mengelompokkan semuanya ke dalam satu cluster dan selesai ketika kita hanya memiliki satu cluster.



Types of Hierarchical Clustering

Metode hierarchical clustering dibagi menjadi dua yaitu:

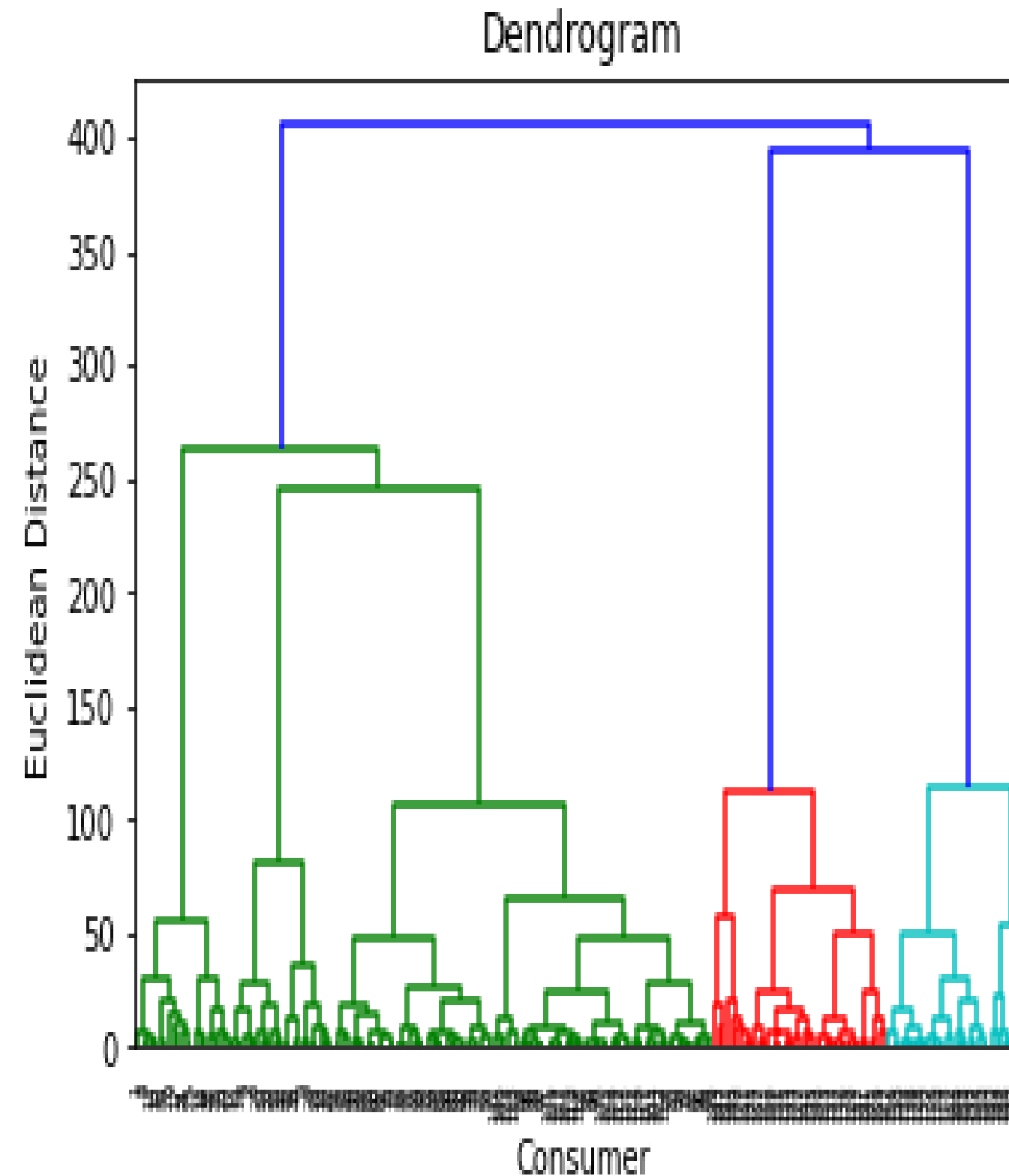
1. Divisive

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

2. Agglomerative

Pengelompokan aglomeratif dikenal sebagai pendekatan bottom-up, yaitu pengelompokan dimulai dari cluster kecil menuju satu cluster besar.

Agglomerative



Divisive



Hierarchical Clustering

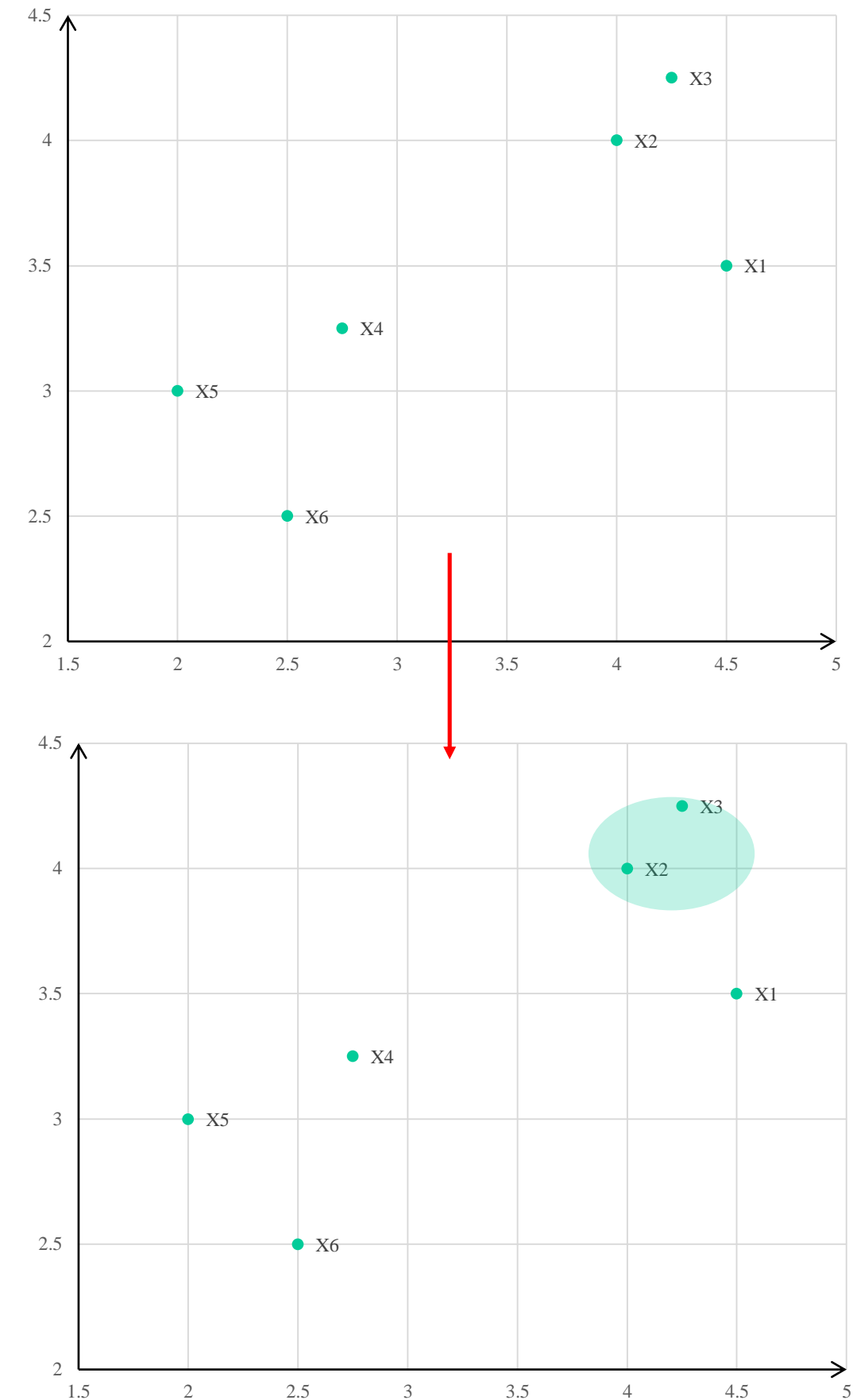
Langkah-langkah metode hierarchical clustering dengan agglomerative:

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

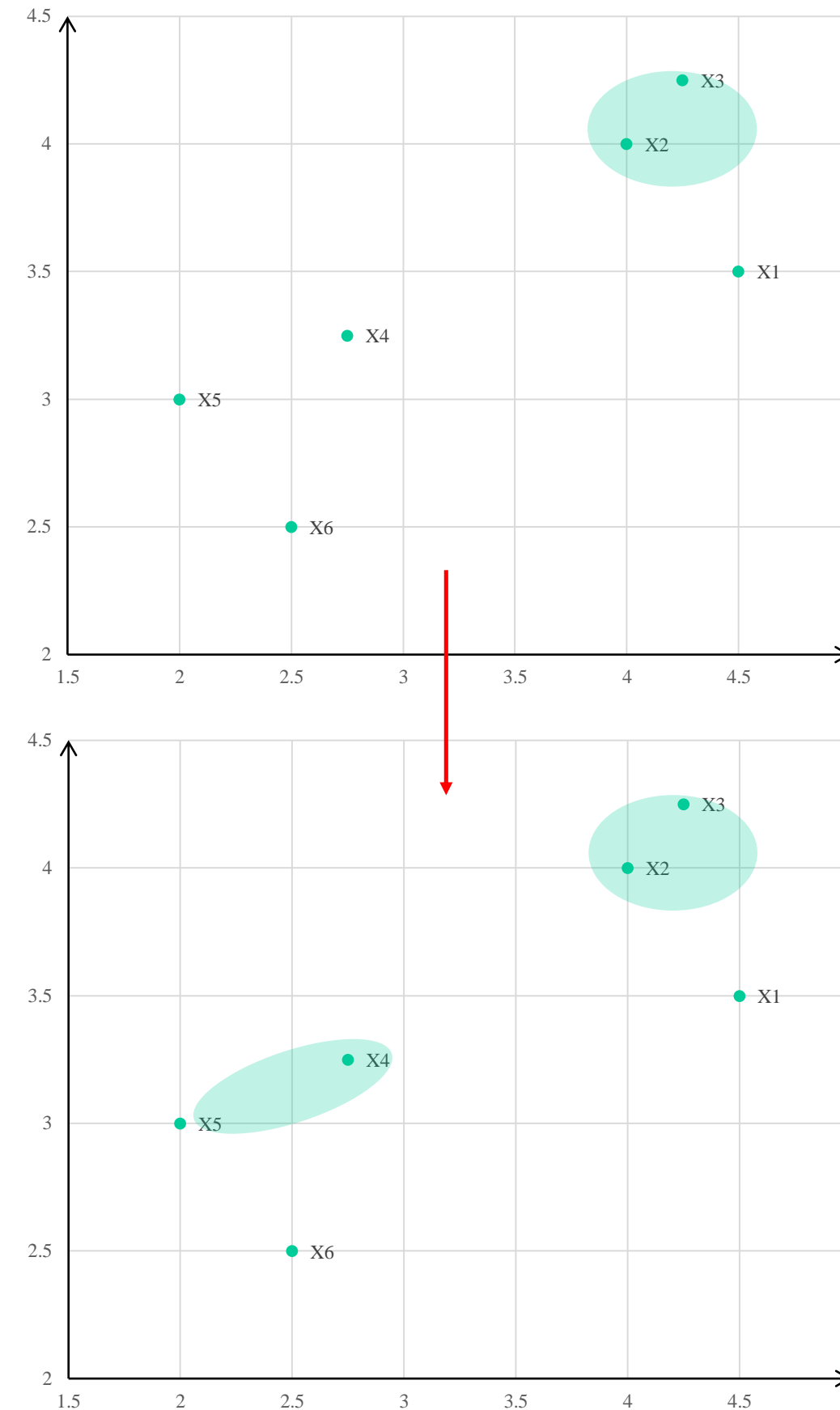
Ilustrasi dari langkah-langkah agglomerative:

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



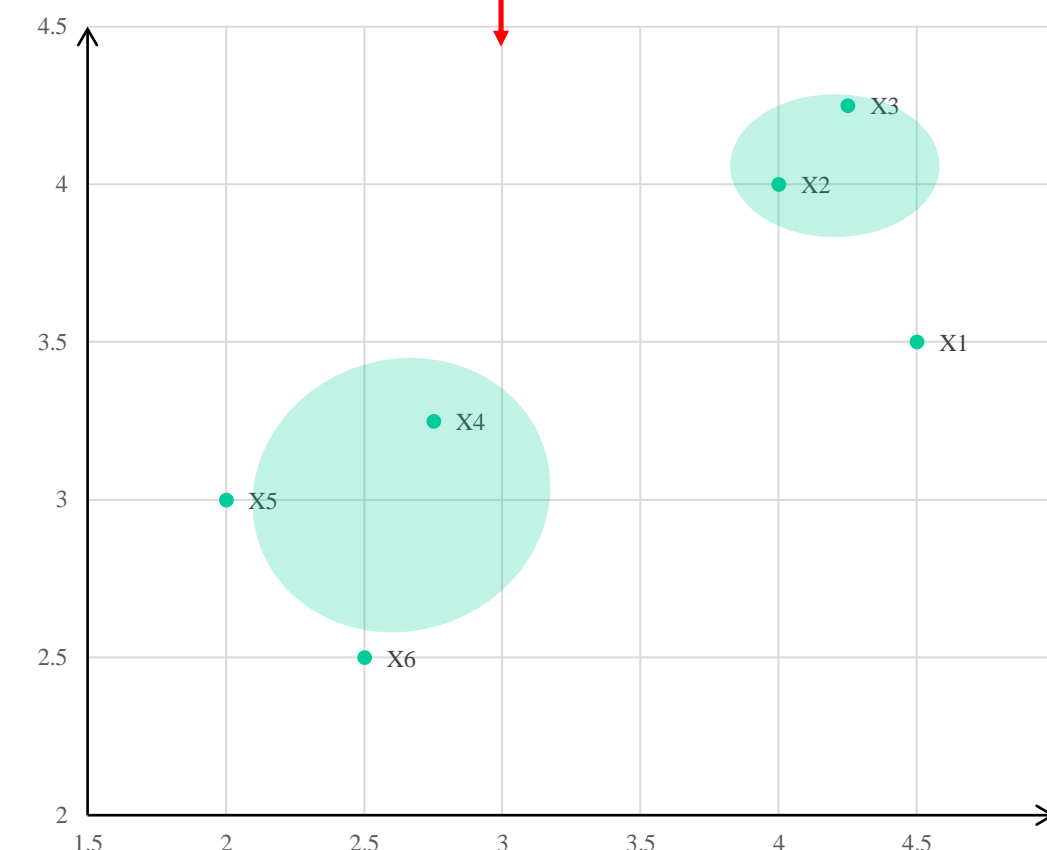
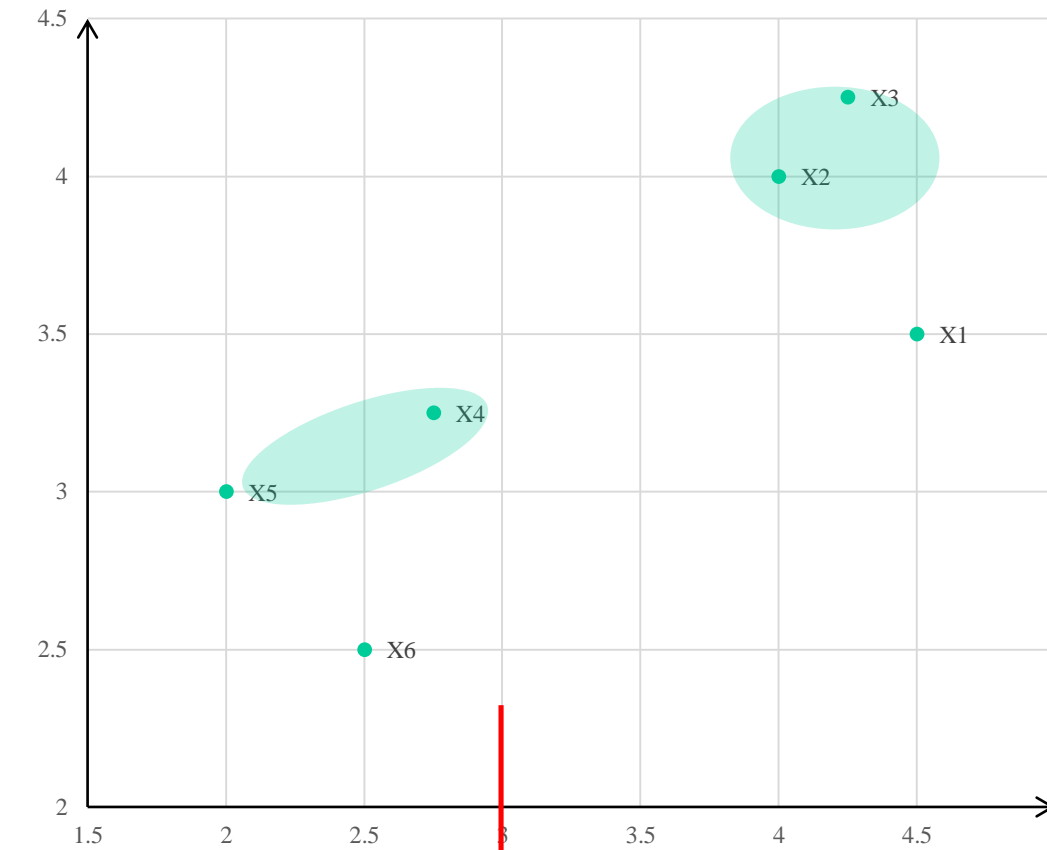
Hierarchical Clustering

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



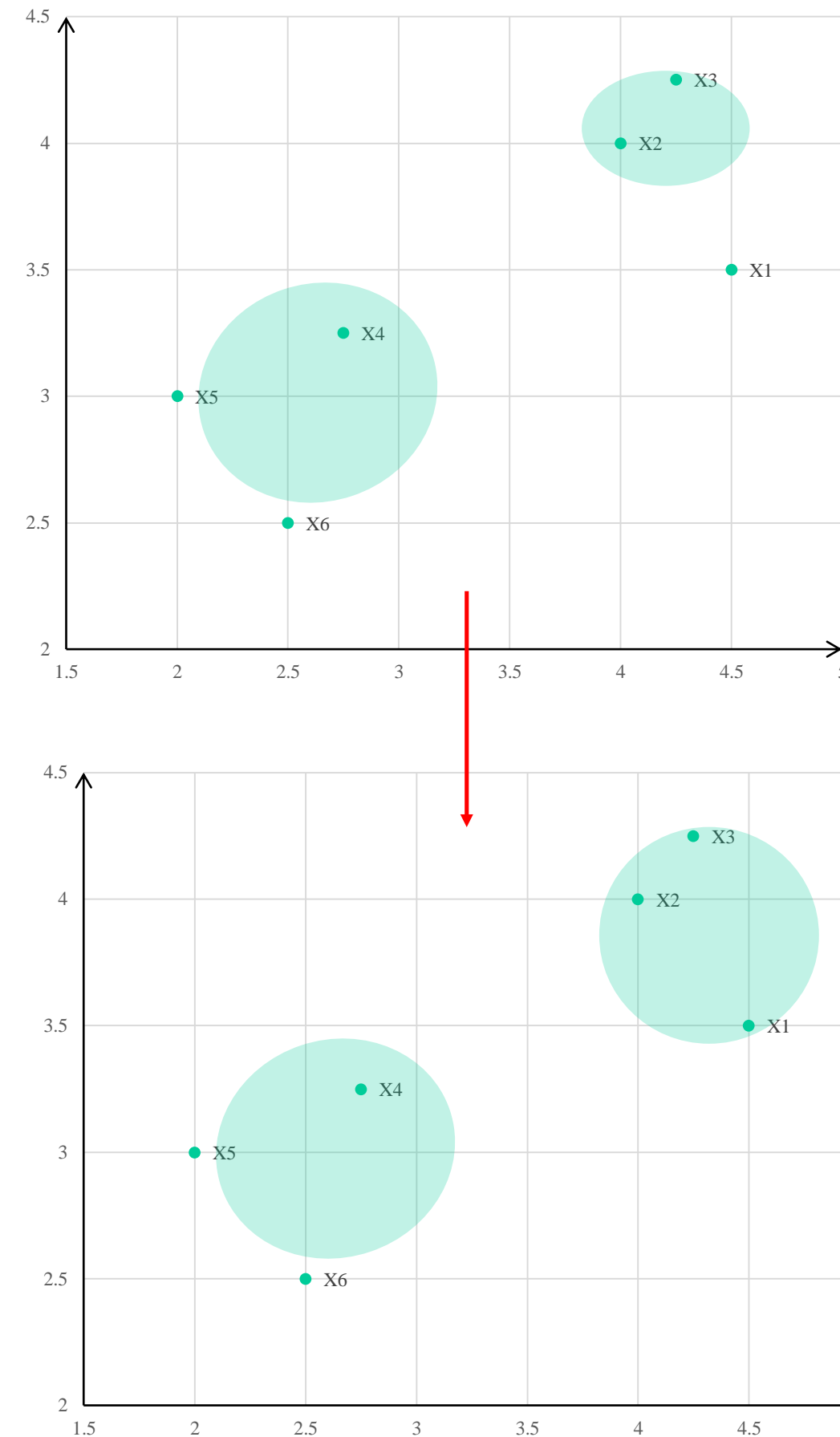
Hierarchical Clustering

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.



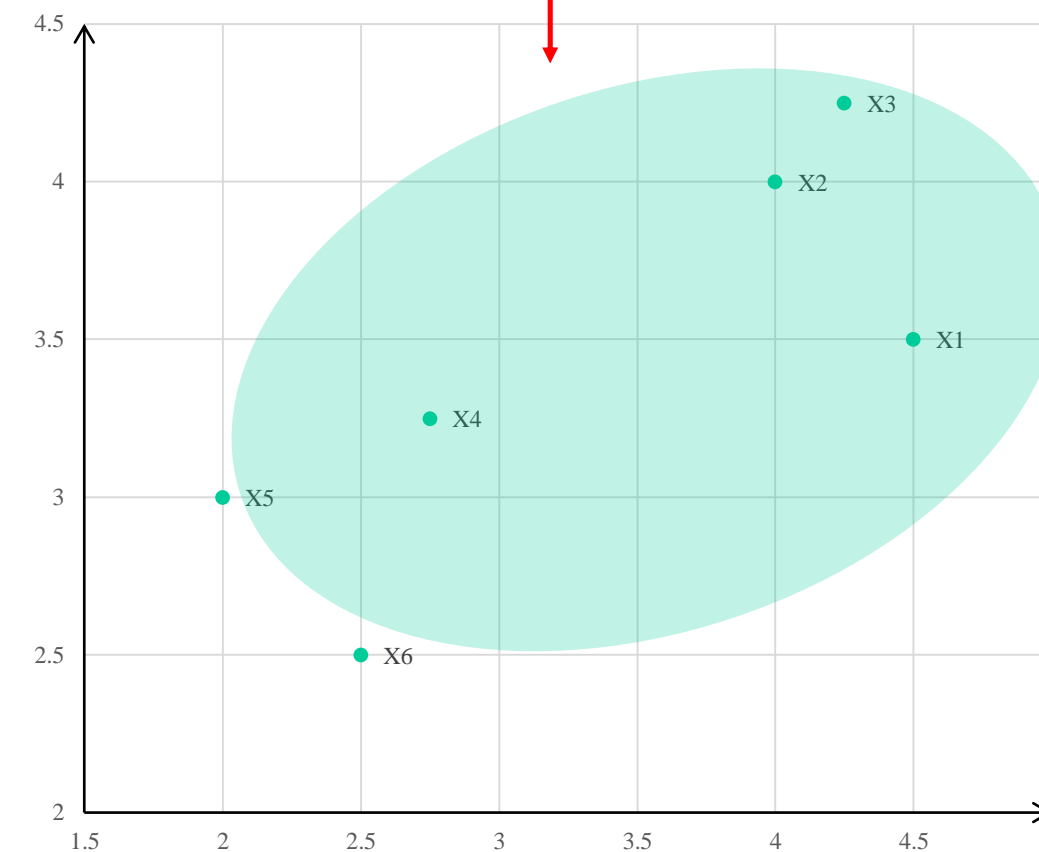
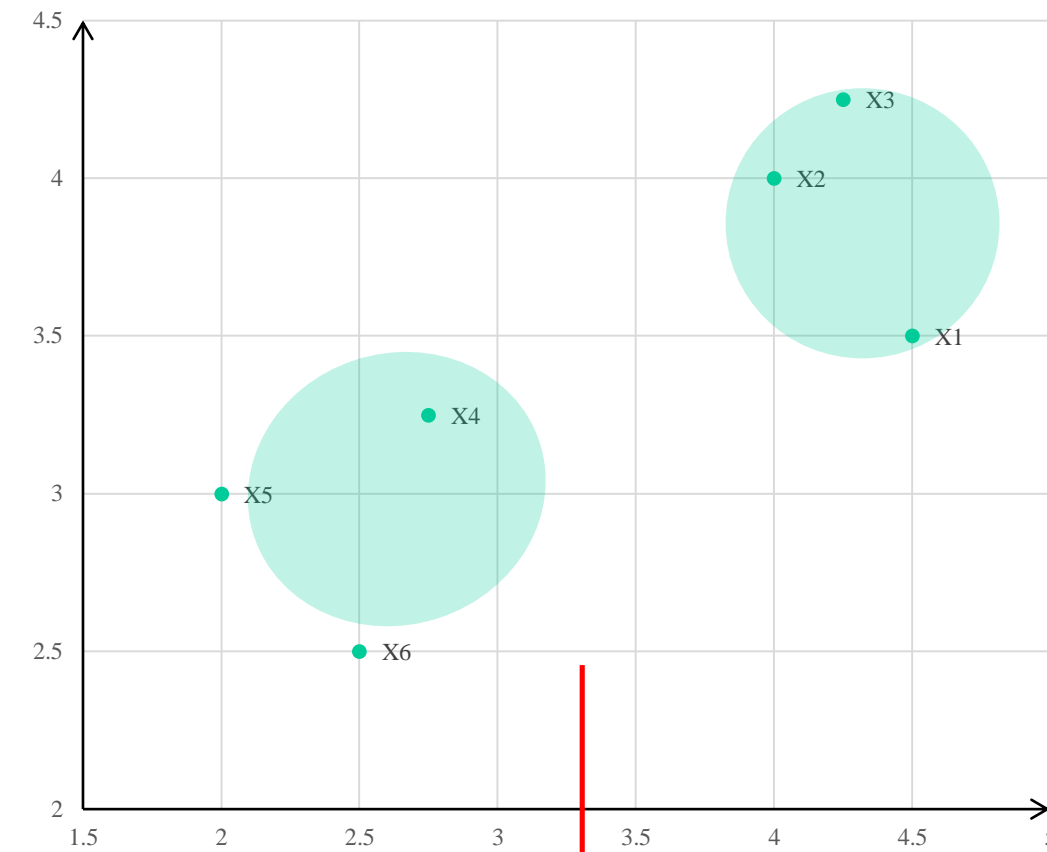
Hierarchical Clustering

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 bergabung menjadi satu cluster besar.



Optimasi Hierarchical Clustering

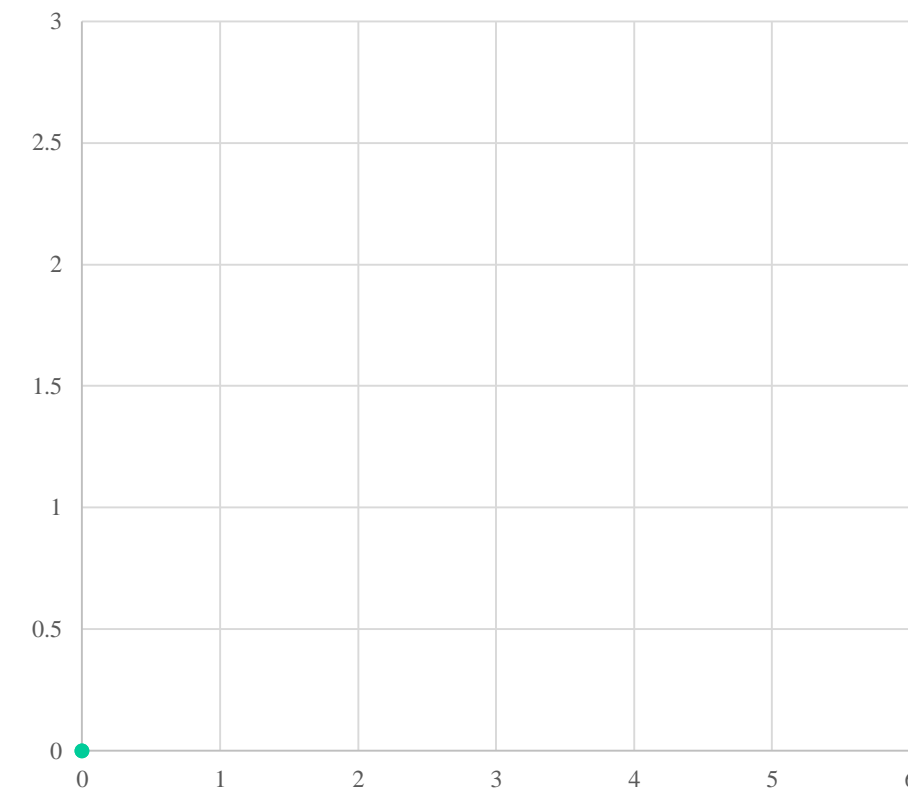
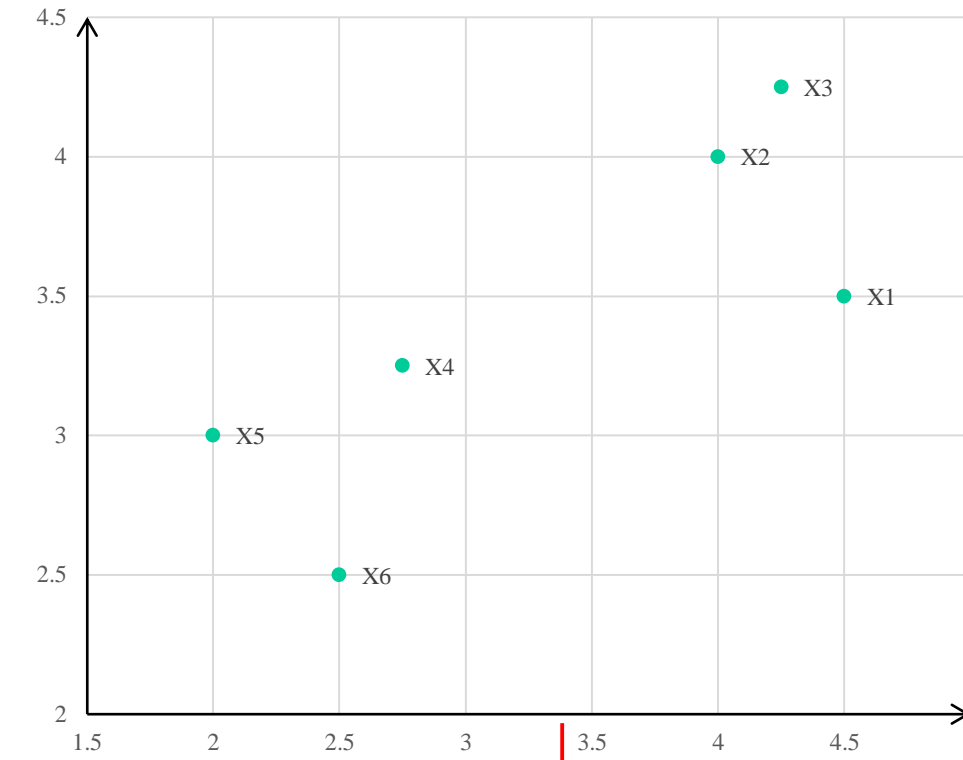
Pada clustering K-Means yang kita pelajari sebelumnya, untuk mengetahui jumlah cluster yang tepat kita dapat menggunakan metode Elbow, pada hierarchical clustering kita dapat menggunakan Dendrogram.

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 euclidean-nya.

Optimasi Hierarchical Clustering

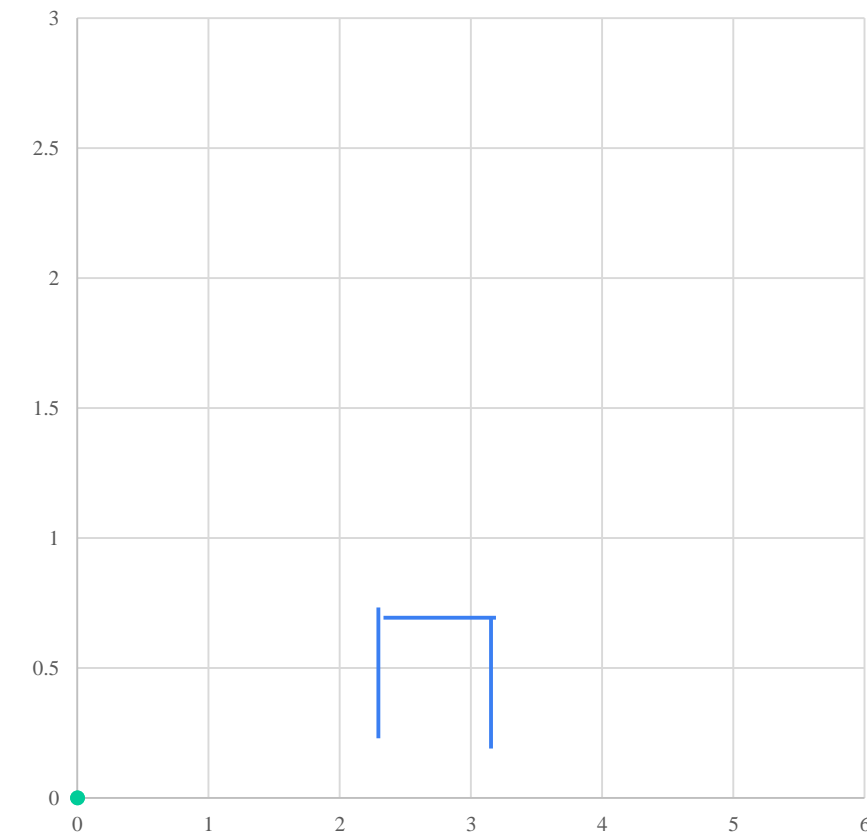
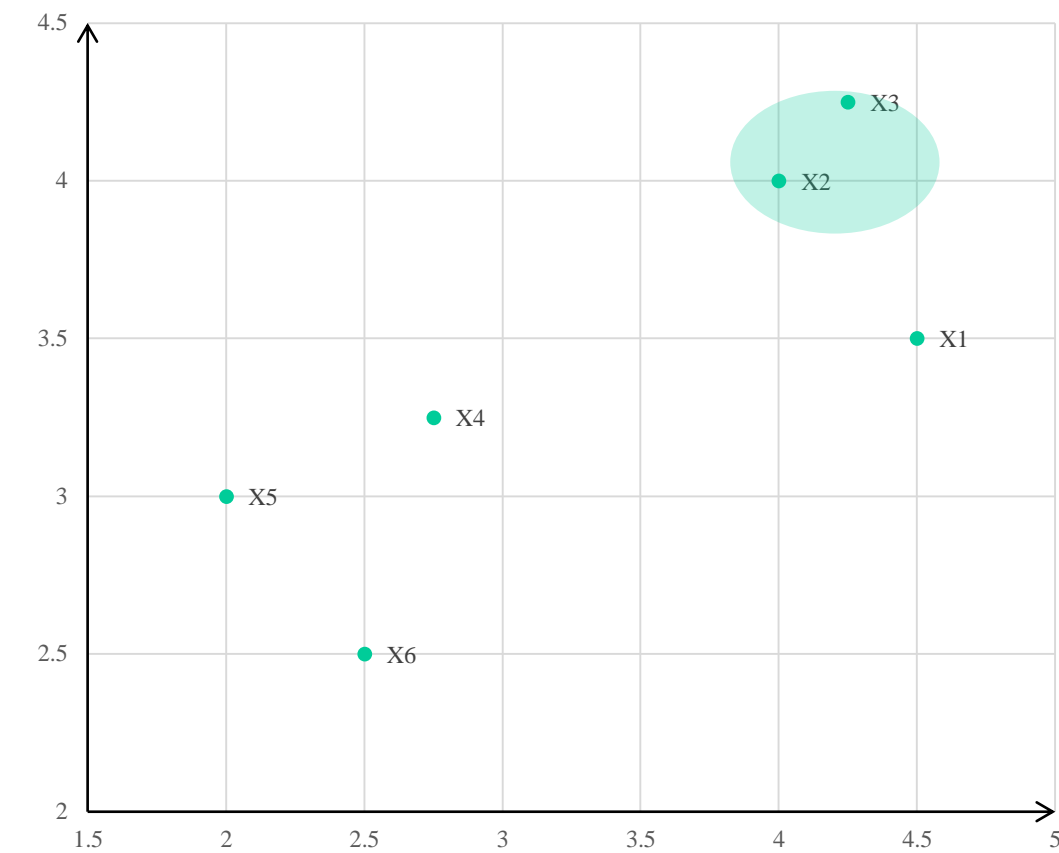
Ilustrasi dari penentuan dendogram adalah sebagai berikut:

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.



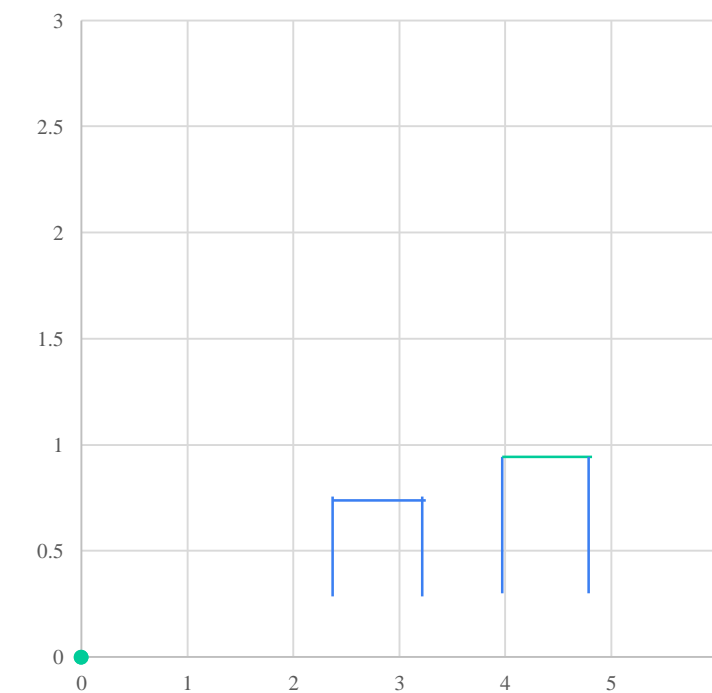
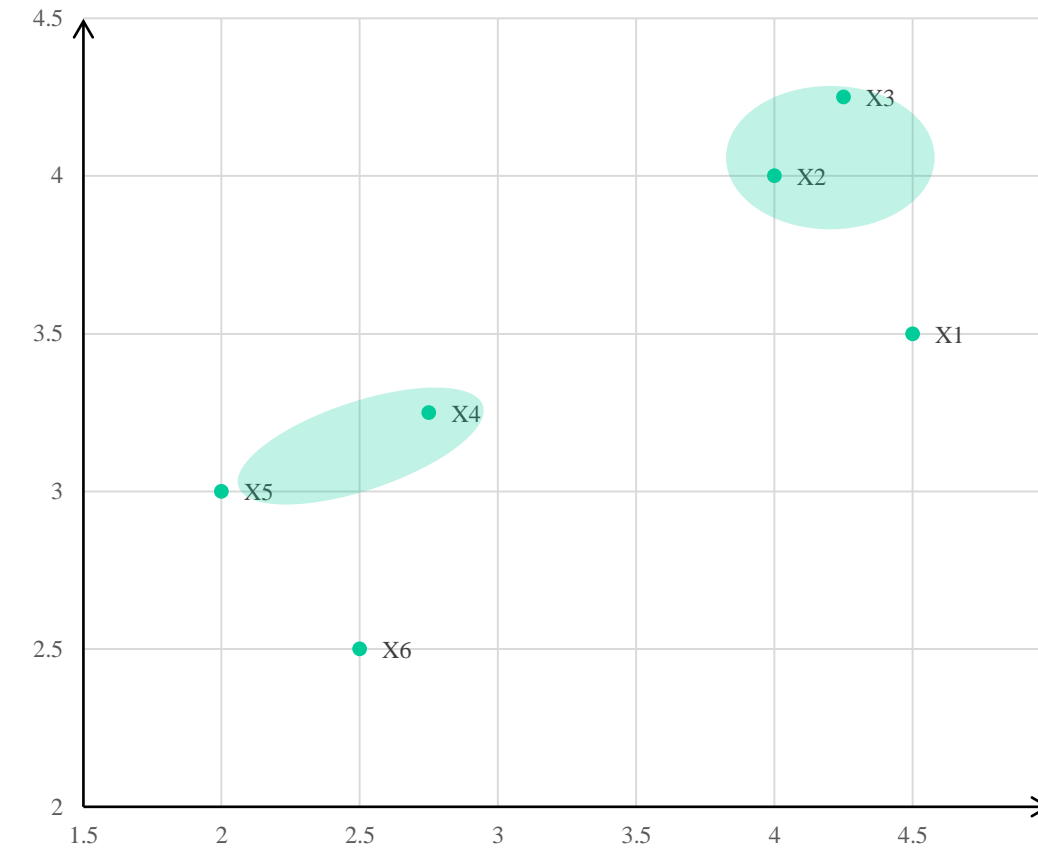
Optimasi Hierarchical Clustering

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



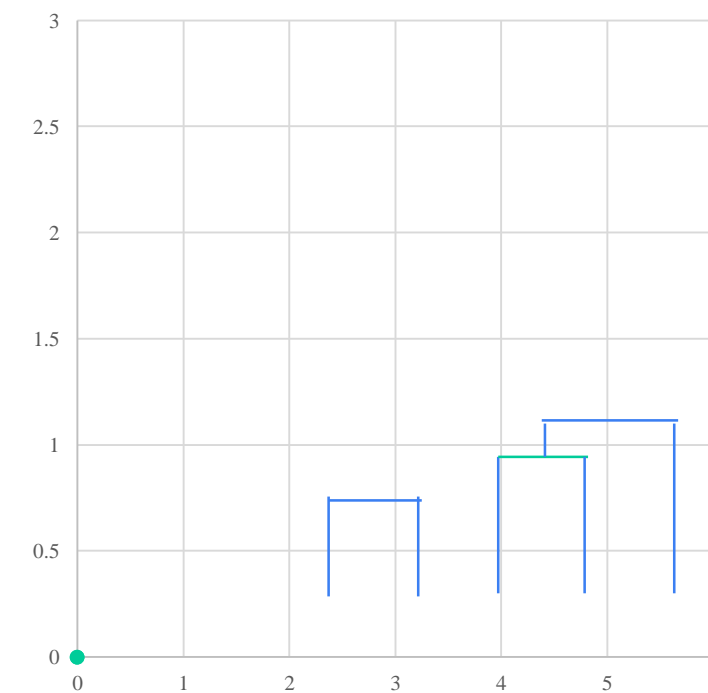
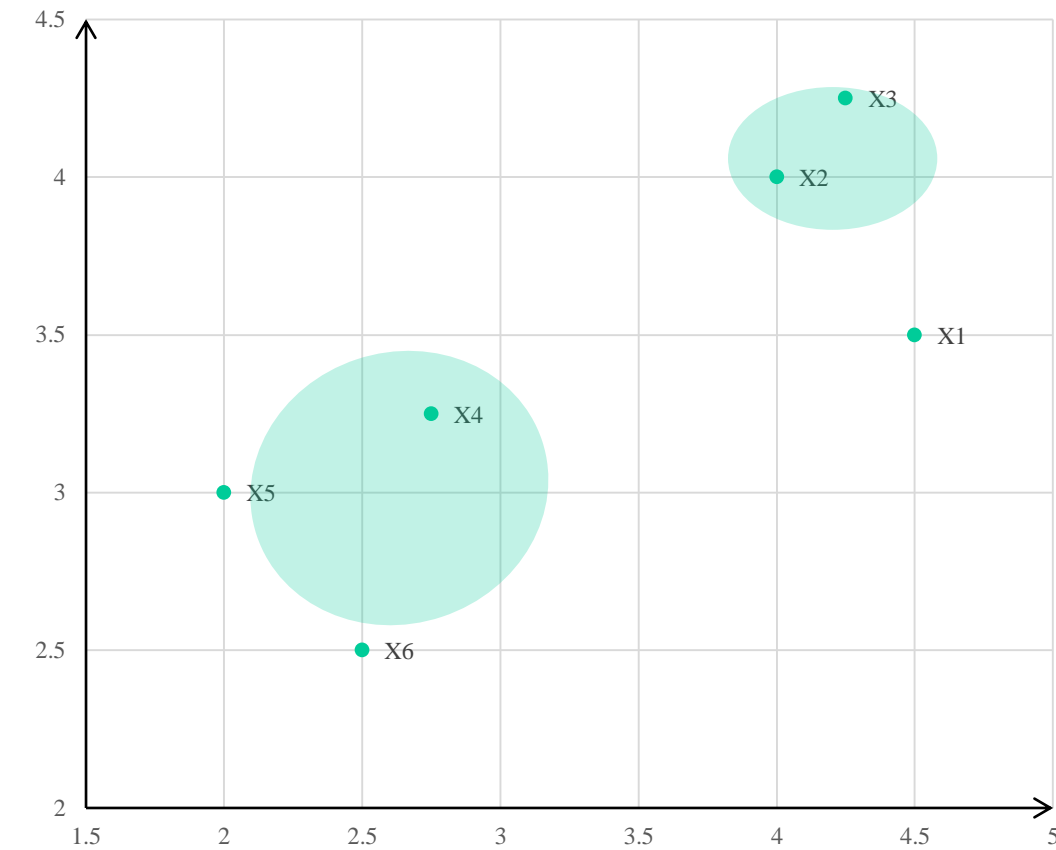
Optimasi Hierarchical Clustering

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



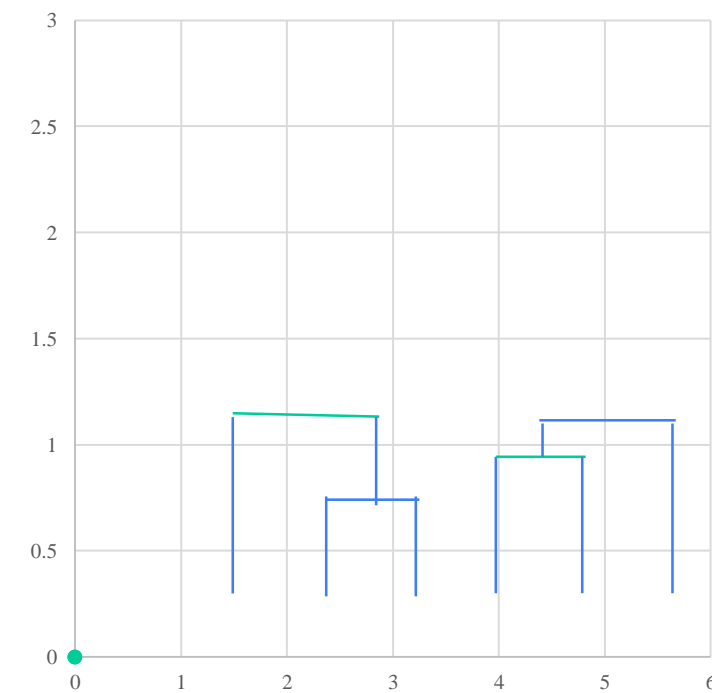
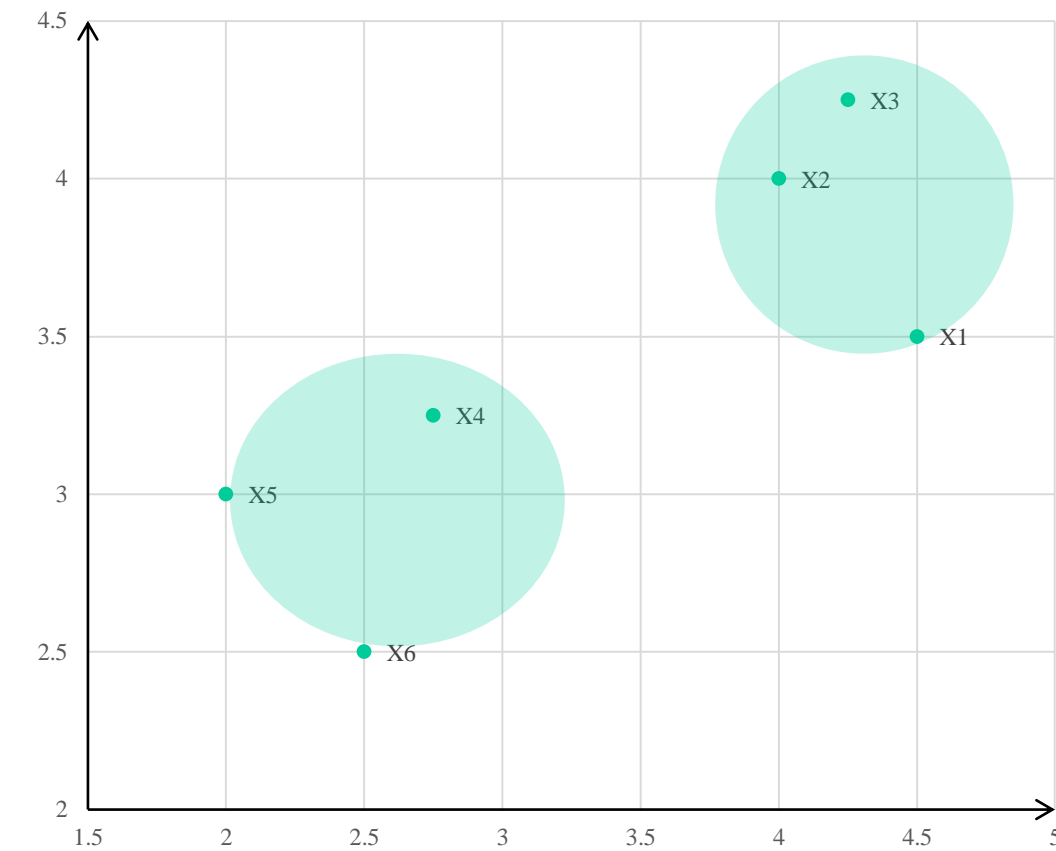
Optimasi Hierarchical Clustering

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,



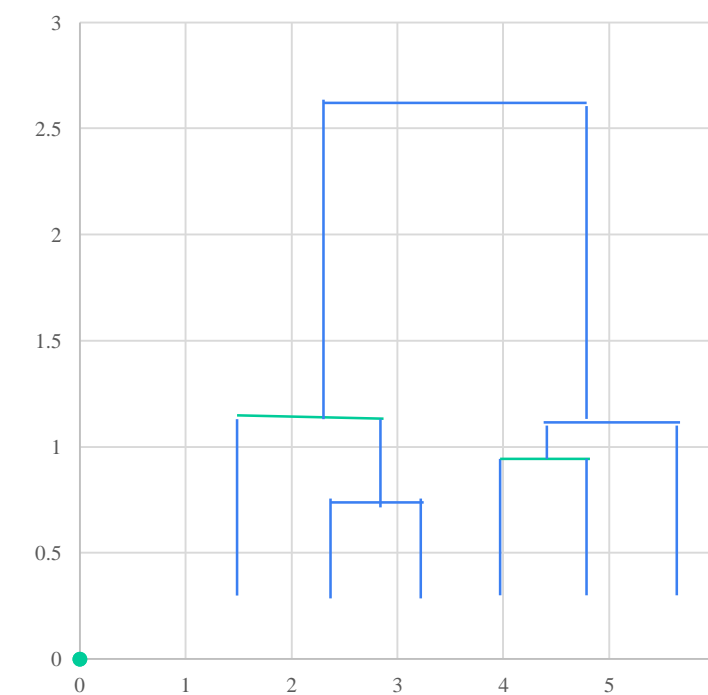
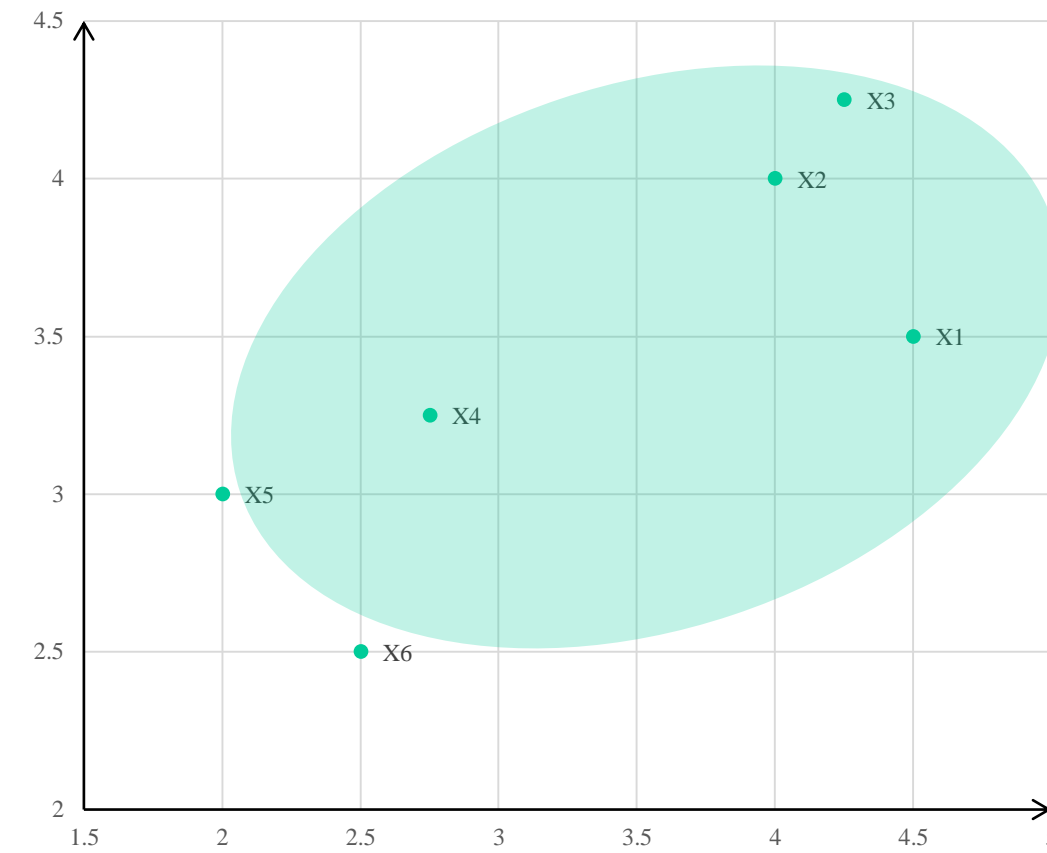
Optimasi Hierarchical Clustering

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.



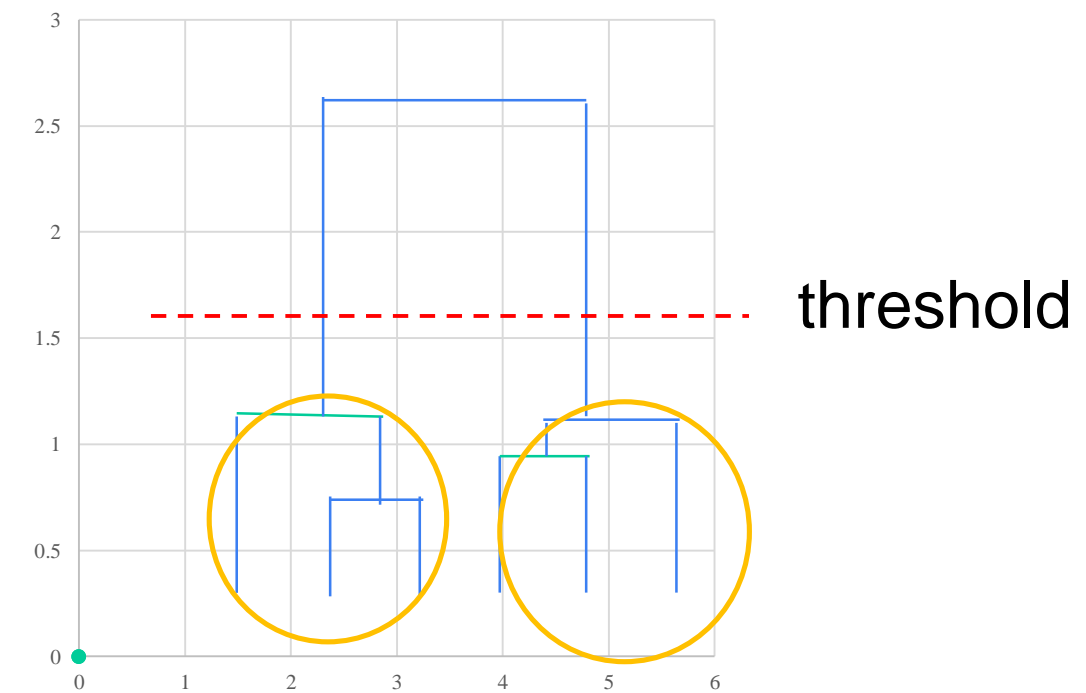
Optimasi Hierarchical Clustering

6. Proses akan berhenti setelah semua cluster telah bergabung 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:



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

Coding dalam Bahasa pemrograman Python 1:

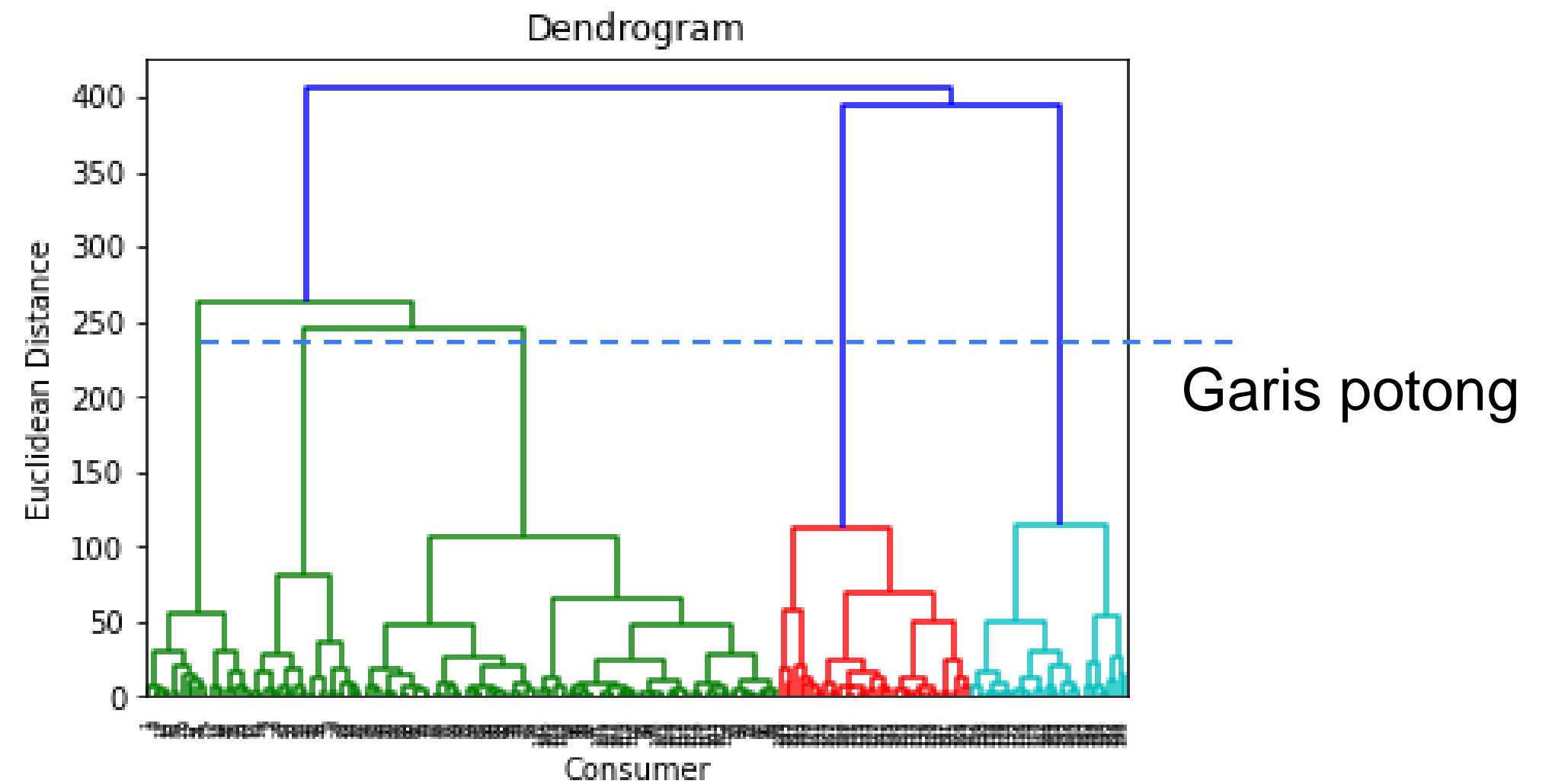
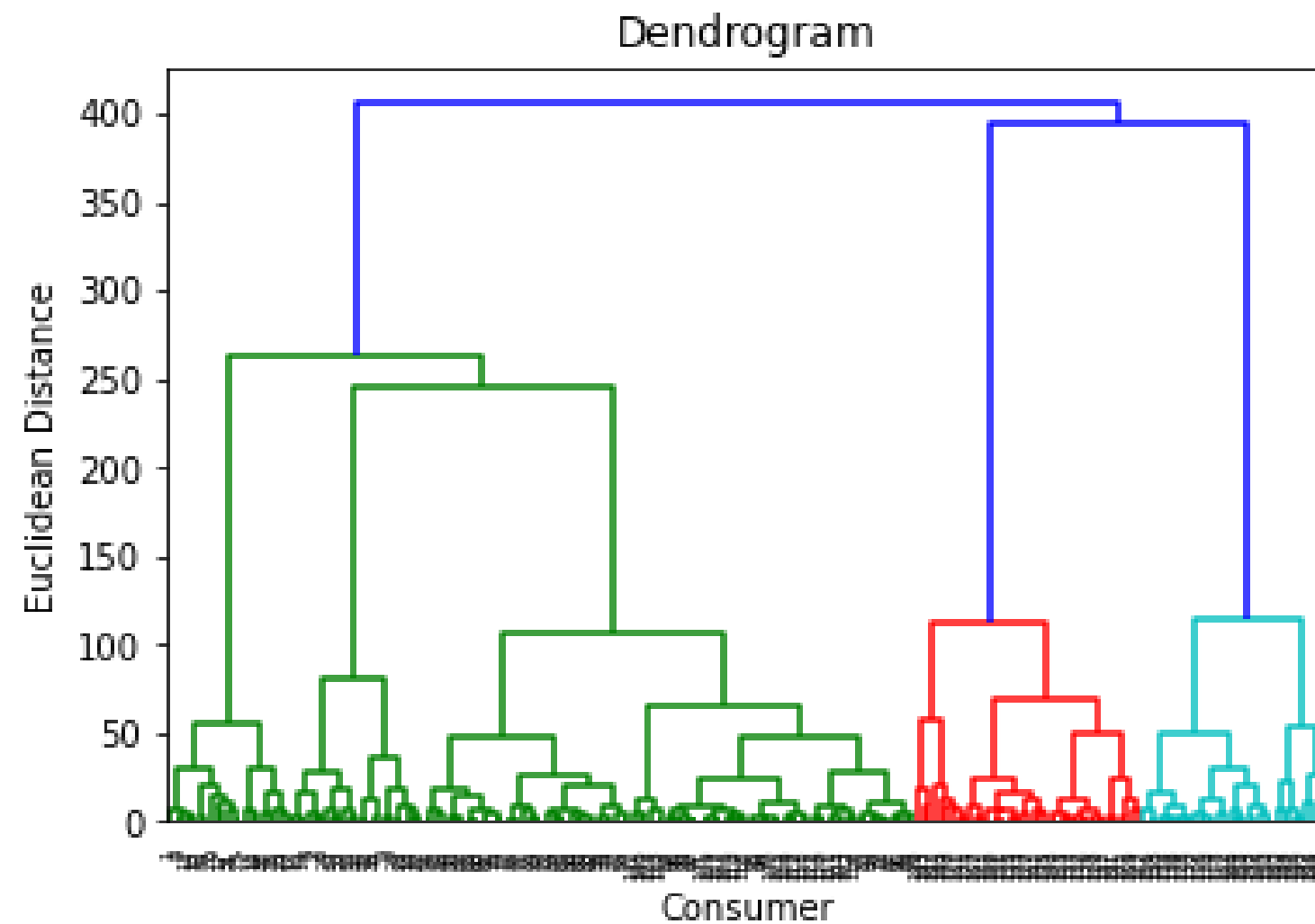
```
# Mengimpor library
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd

# 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 vertikal yang terpotong.



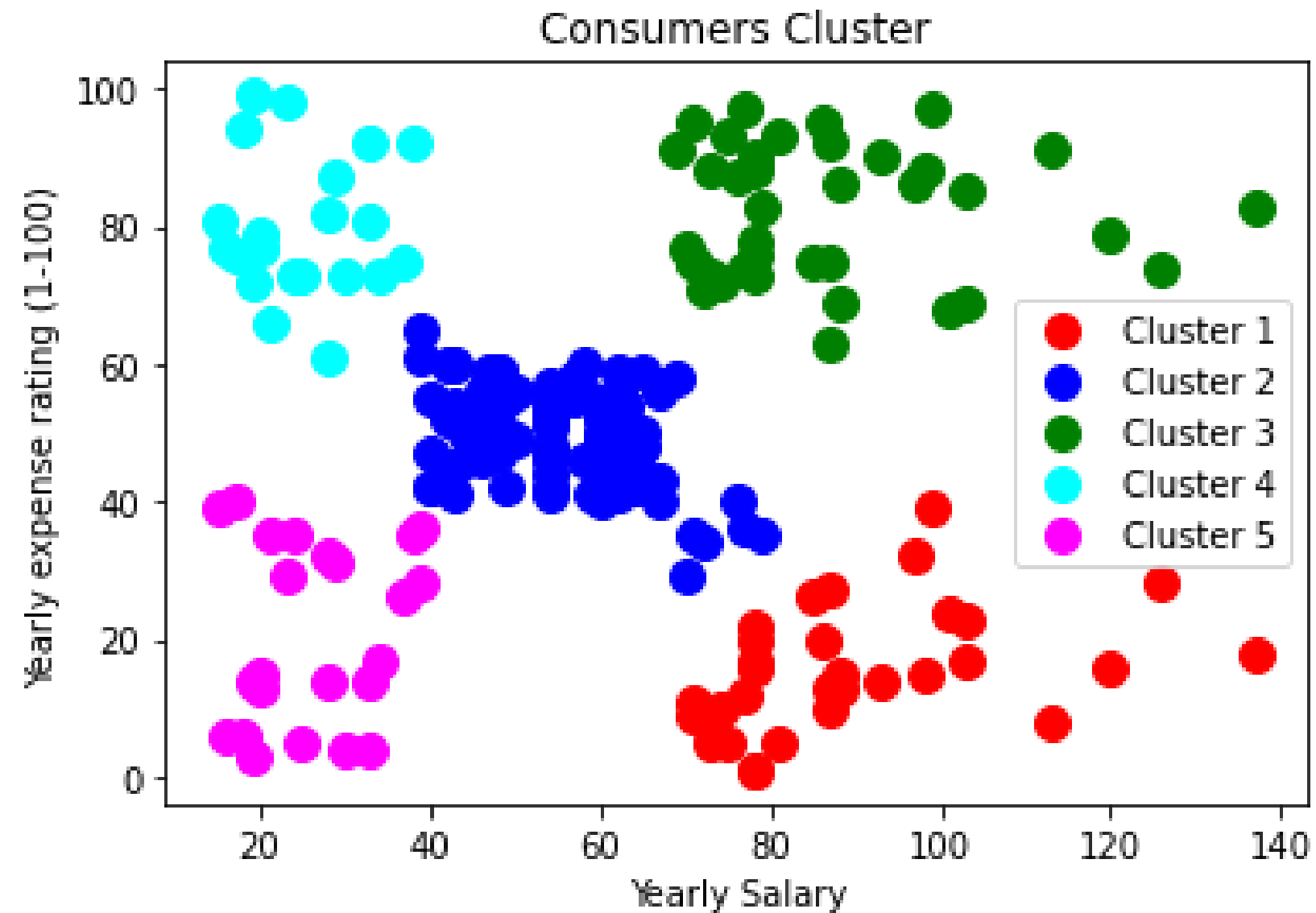
Studi Kasus Hierarchical Clustering

Coding dalam Bahasa pemrograman Python 2:

```
# 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

Pada slide sebelumnya telah dijelaskan Teknik clustering berbasis partisi dan hirarki, kedua Teknik ini efisien jika dataset berbentuk normal. Namun ketika data cluster berbentuk arbiter atau ingin mendeteksi cluster outlier, maka DBSCAN merupakan Teknik cluster yang sesuai. Untuk lebih jelasnya dapat diamati pada gambar berikut:

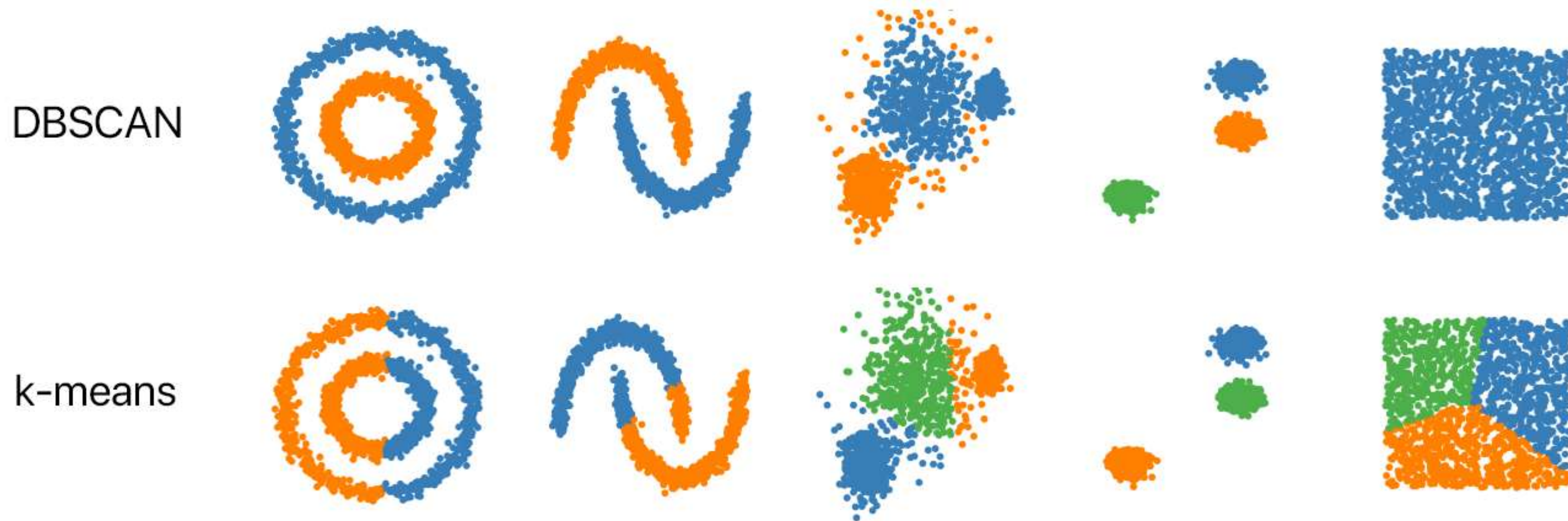


Image source: <https://github.com/NSHipster/DBSCAN>

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.

Referensi

- Ethem Alpaydin, "Introduction to Machine Learning, Fourth Edition", MIT Press, 2020
- Zhi-Hua Zhou, Shaowu Liu, "Machine Learning", Springer, 2023
- Charu C. Aggarwal, Chandan K. Reddy, "Data Clustering", Chapman and Hall/CRC, 2013
- Anil K. Jain, Richard C. Dubes, "Algorithm for Clustering Data", Michigan State University, Prentice Hall Advance Reference, 2018 (free download at http://www.cse.msu.edu/~jain/Clustering_Jain_Dubes.pdf)

