

PENDAHULUAN

ML (*Machine Learning*) dibagi ke dalam tiga kelompok, yaitu *supervised learning*, *unsupervised learning*, dan *reinforcement learning*. Teknik *clustering* adalah salah satu metode *machine learning* yang termasuk dalam kelompok *unsupervised learning*. Teknik ini melakukan pembelajaran mesin tanpa ada supervisi untuk menyelesaikan suatu masalah. Tidak seperti pembelajaran yang diawasi (*supervised learning*), clustering dianggap sebagai metode pembelajaran tanpa pengawasan karena tidak memiliki kebenaran dasar untuk membandingkan keluaran algoritme pengelompokan dengan label sebenarnya untuk mengevaluasi performanya. Penerapan clustering bertujuan untuk mengeksplorasi dan menyelidiki struktur data dengan mengelompokkan titik data ke dalam subkelompok yang berbeda.

Clustering adalah salah satu teknik analisis dan eksplorasi data yang paling sering digunakan untuk mendapatkan pola suatu struktur data. 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. Dengan kata lain, clustering adalah suatu Teknik untuk kelompok data yang homogen sedemikian rupa sehingga titik data di setiap cluster semirip mungkin menurut ukuran kesamaan seperti jarak berbasis euclidean atau jarak berbasis korelasi. Keputusan tentang ukuran kemiripan dapat ditentukan melalui jumlah centroid masing-masing kelompok dan jaraknya.

Analisis clustering dapat dilakukan berdasarkan fitur yang telah ditentukan untuk menemukan subkelompok berdasarkan fitur atau berdasarkan sampel. Sebagai contoh Ketika terdapat data mahasiswa dalam satu angkatan yang berisi nilai quiz, UTS, UAS, serta jumlah kehadiran dari seluruh mata kuliah. Maka dengan penerapan algoritma *clustering* dapat menganalisa kelompok-kelompok mahasiswa yang memiliki nilai baik, nilai yang biasa saja, baik di semua mata kuliah atau mata kuliah tertentu, serta yang rajin menghadiri kuliah. Hasil dari pengelompokan tersebut dapat menghasilkan analisa sesuai dengan yang diinginkan untuk memberikan suatu dukungan keputusan.

K-Means

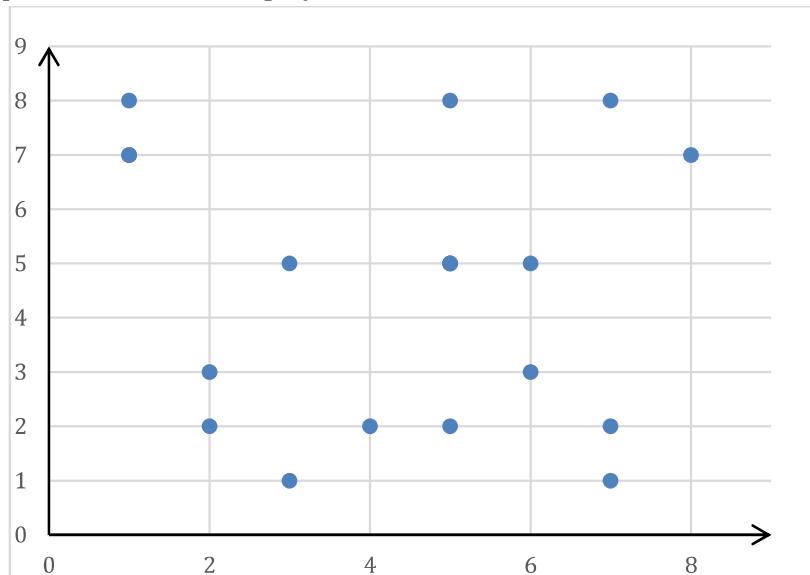
Algoritma Kmeans adalah salah satu algoritma *clustering* yang bersifat iteratif yang mencoba untuk mempartisi dataset menjadi subkelompok non-overlapping berbeda yang ditentukan oleh K (cluster) di mana setiap titik data hanya dimiliki oleh satu kelompok. K-Means mencoba membuat titik data intra-cluster 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 penyelesaian algoritma K-Means

Cara kerja algoritma k-means adalah sebagai berikut:

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

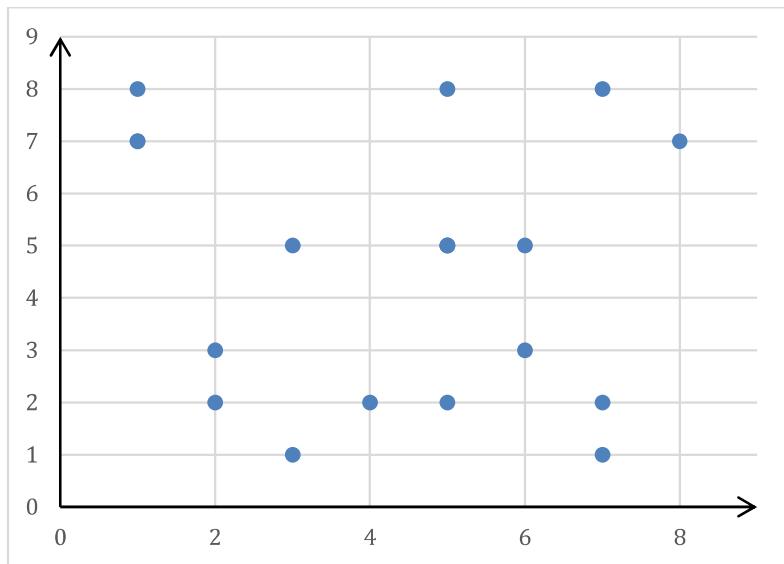
Sebagai contoh terdapat dataset 2-dimensi seperti yang tampak pada Gambar 1 yang ditampilkan dalam grafik, langkah pertama adalah memilih jumlah kluster. Misal kita pilih untuk membaginya ke dalam 2 kluster.



Gambar 1. Grafik klaster dua dimensi

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

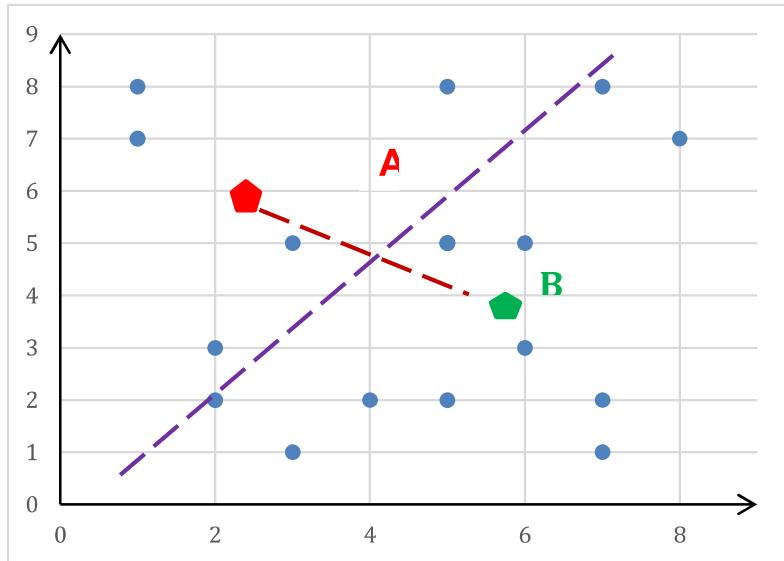
Langkah kedua adalah menentukan titik pusatnya. Dalam satu klaster terdapat satu titik pusat atau yang disebut dengan *centroid*. Penentuan awal posisi titik pusat ini bebas, karena nantinya algoritma K-Means akan merubah posisi tiap titik hingga dicapai solusi paling optimal. Pada Gambar 2, titik merah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



Gambar 2. Klaster dengan dua titik pusat

- 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.

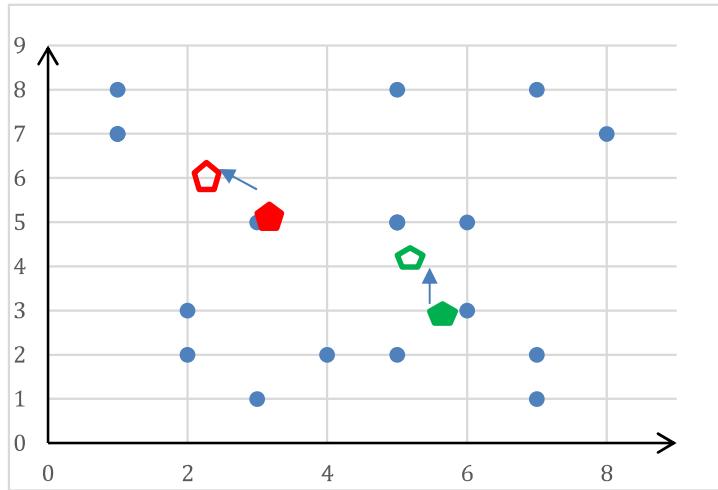
Berdasarkan pengelompokan pada langkah ke dua, maka setiap titik data saat telah tergabung dalam salah satu kluster. Titik data yang diwakili dengan symbol ‘x’ berwarna merah masuk ke kluster 1, dan symbol ‘x’ berwarna biru masuk ke kluster 2, seperti pada Gambar 3.



Gambar 3. Pengelompokan titik data pada dua klaster

- Lakukan kalkulasi, dan tempatkan pusat *centroid* yang baru untuk setiap *cluster*-nya. Langkah ini bisa disebut juga dengan istilah penyempurnaan *centroid*. Langkah ke-empat 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 4.



Gambar 4. Pencarian titik pusat baru

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 metode 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=3, maka ada informasi tambahan 1 kluster. Yang mungkin, akan memberi kita informasi, apa arti dari kluster ketiga ini.

Jika K=10, maka terlalu banyak kluster. Mungkin akan terlalu sulit memilih strategi yang tepat (strategi marketing misalnya) untuk masing-masing klusternya.

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

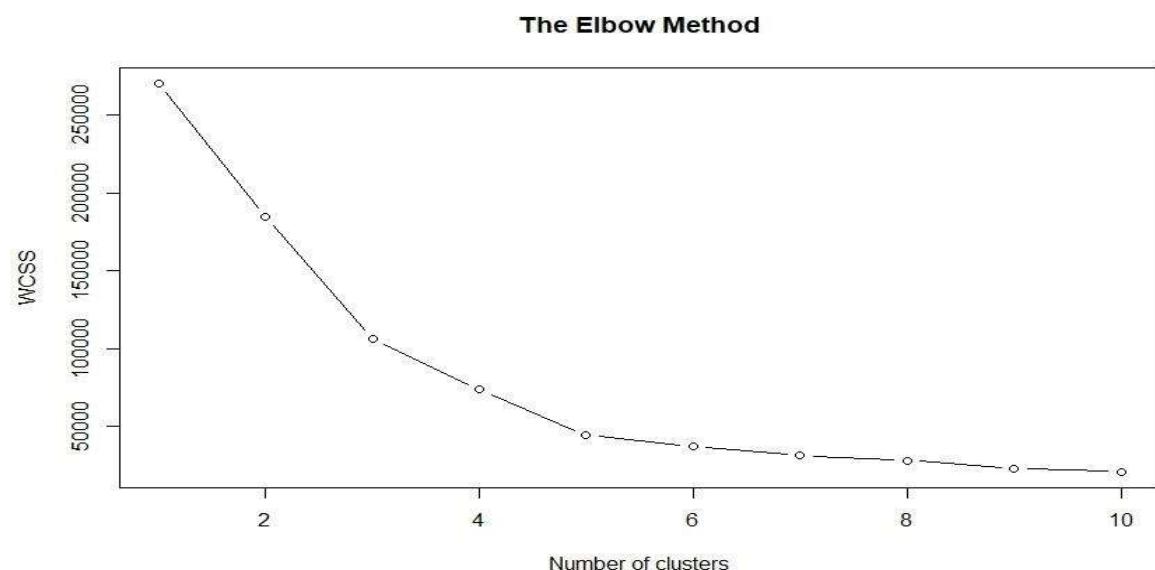
akan membantu kita untuk memilih nilai K yang tepat dengan menggunakan *metric WCSS* (*Within Cluster Sum of Squares*), contoh penghitungan untuk dua klaster:

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

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 Pi 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 kluster 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*.

Contoh perhitungan WCSS dapat dilihat pada grafik berikut:



Gambar 5. Contoh diagram hasil penghitungan menggunakan matriks WCSS

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.

Studi Kasus Metode 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, barang-barang apa saja yang cocok bagi mereka, dll.

Coding dalam 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()

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

plt.show()
```

```

# 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()

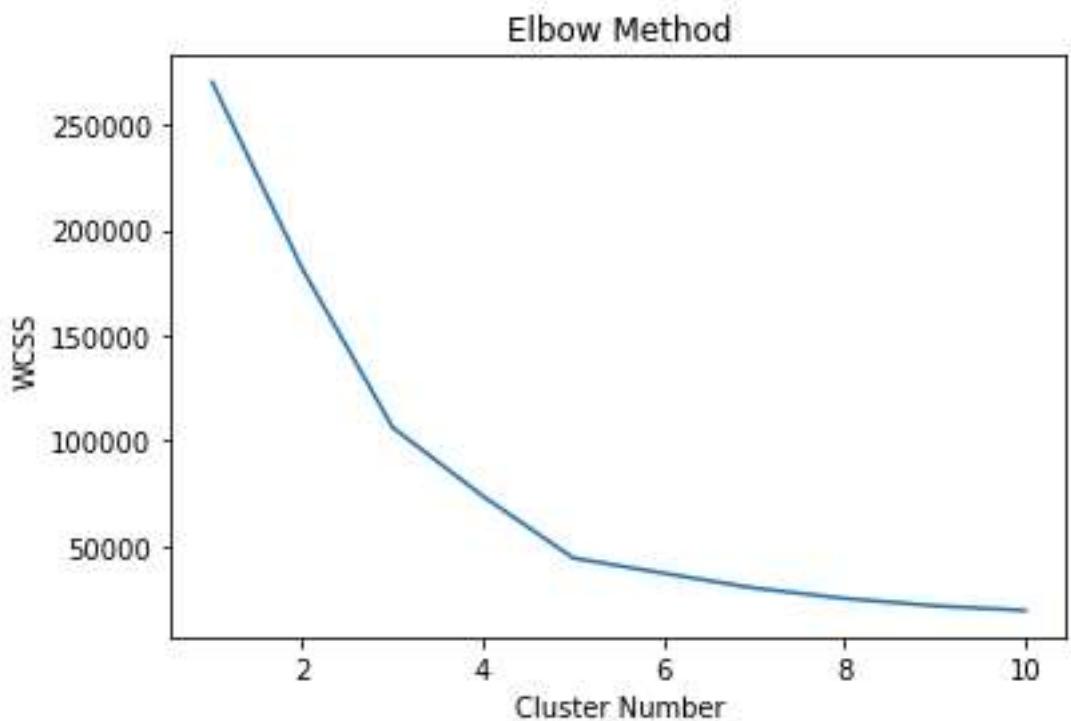
```

Penjelasan dari *coding*:

- Baris 2 sampai 4 mengimpor *library* yang dibutuhkan
- Baris 7, mengimpor dataset dan membacanya
- Baris 8, melakukan *slicing*, dari dataset yang dimiliki.
- Baris 11, mengimpor library K-Means.
- Baris 12, membuat *list* WCSS (mempersiapkan perhitungan WCSS).
- Baris 13 adalah perintah *looping*, pada studi kasus ini dapat diamati bahwa looping yang dilakukan adalah sebanyak 10 kali. Pada coding dapat diamati ditulis dengan range(1,11), hal ini dikarenakan pada python looping terakhir dianggap hasil terakhir sehingga harus melebihkan satu dari jumlah looping yang diinginkan, Misalnya jika ingin iterasi sebanyak 100 maka dapat ditulis dengan (1,100).
- Baris 14 baris ini berfungsi untuk menuliskan objek kmeans pada algoritma K-Means. Selanjutnya perintah pertama adalah KMeans (kapital K dan M), yang merupakan *class* dari library K-Means yang diimpor di line 11, dengan beberapa parameter n_clusters yang merupakan jumlah kluster, diikuti dengan parameter kedua ini yang merupakan pemilihan jumlah K di awal (kali ini kita gunakan K++). Kemudian parameter yang terakhir adalah random_state = 42. Random state ini seperti *seed* pada R, yang jika dipilih 42, maka ketika kita memilih 42 di kesempatan yang berbeda, maka bilangan random yang dihasilkan akan sama.
- Baris 15 merupakan perintah agar objek kmeans di line 14, digunakan untuk mengolah data X yang sudah kita definisikan di line 8.

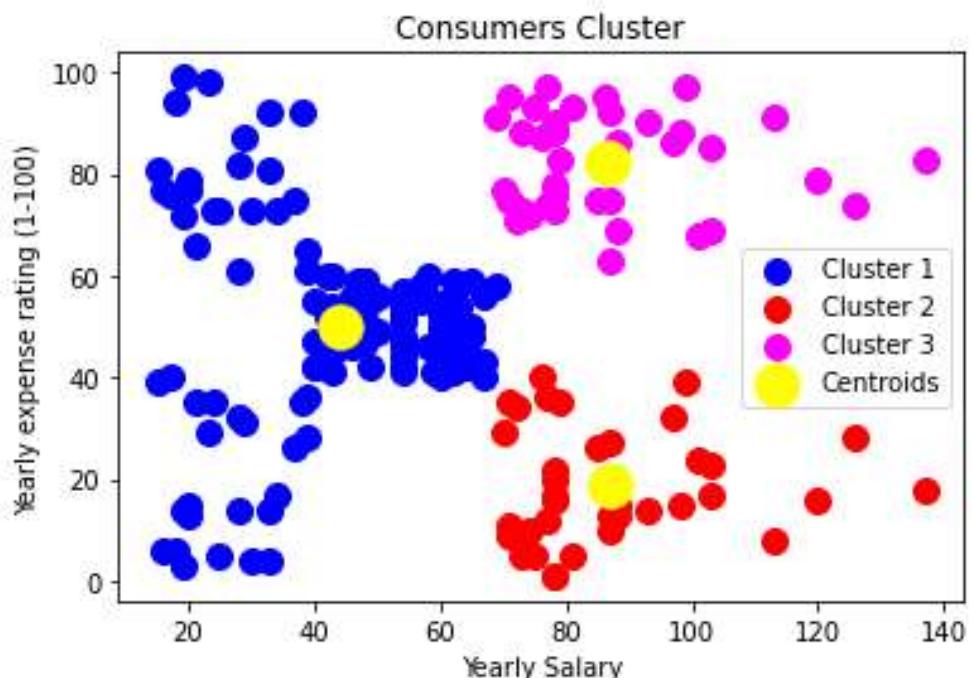
- Baris 16 merupakan perintah untuk menghitung WCSS dengan menuliskan perintah append setelah wcss. Append merupakan *method* di python untuk menambahkan objek. Algoritma wcss dituliskan dengan perintah kmeans.inertia_ (dengan underscore).
- Baris 17 merupakan perintah untuk menampilkan plot. Sumbu x pada plot adalah jumlah kluster dari 1-10, maka ditulis range(1,11). Sumbu y nya adalah skor wcss yang dihitung di line 16.
- Baris 18-20 adalah perintah plot untuk estetika, seperti nama sumbu x, sumbu y dll.
- Baris 21 adalah perintah menampilkan plotnya.
- Baris 25 adalah melakukan prediksi seperti apa pengelompokan klusternya jika kita pilih K=5. Kita siapkan objek y_kmeans (tentu saja pemilihan nama ini bebas) dengan *method* bukan *fit* melainkan *fit_predict* terhadap variabel X yang sudah didefinisikan di line 8.

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

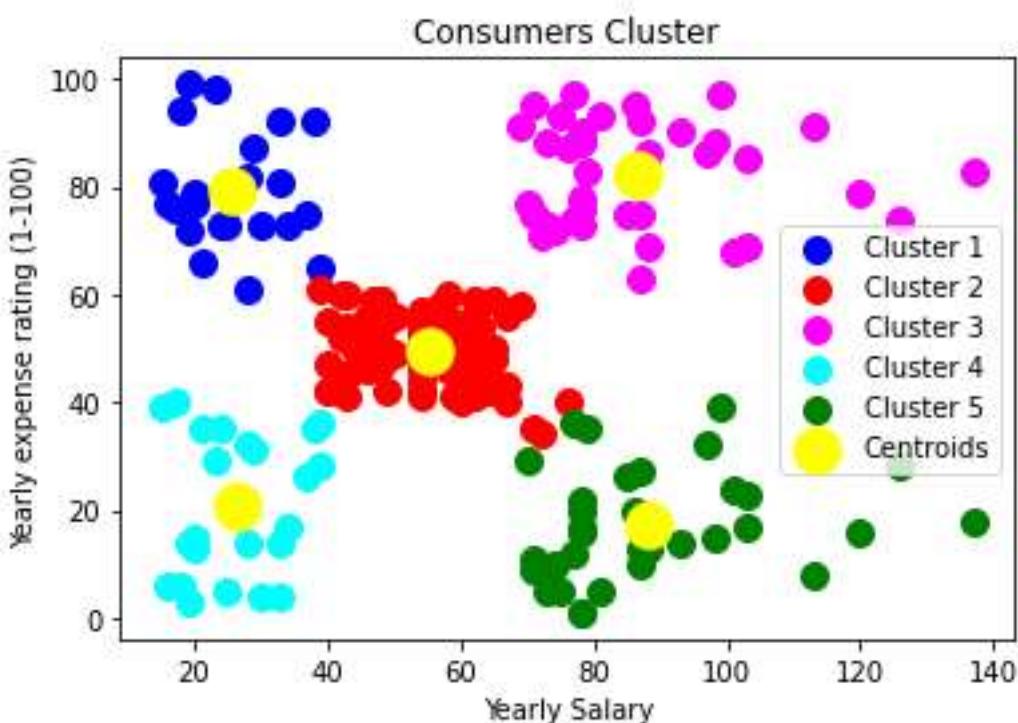


Gambar 6. Hasil penghitungan metode Elbow menggunakan metric WCSS

Pada Gambar 5., 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.



Gambar 7. Hasil clustering menggunakan metode K-Means dengan 3 K

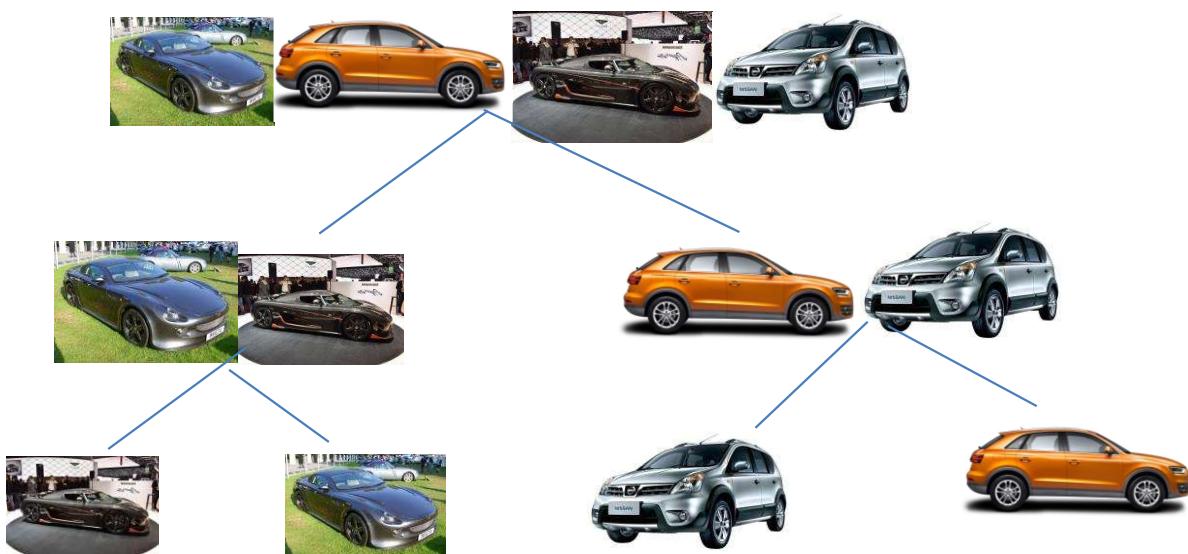


Gambar 8. Hasil clustering menggunakan metode K-Means dengan 5 K

Pada Gambar 6 dan 7 dapat diamati hasil pengelompokan data set yang terbagi dalam 3 dan 5 klaster. Hasil dari pengelompokan tersebut dapat digunakan sebagai dukungan keputusan untuk menentukan startegi 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. Untuk dapat lebih memahami metode hirarkikal clustering, analogi proses pengelompokan hirarkikal dapat diamati pada Gambar 9.



Gambar 9. Pengelompokan data menggunakan metode Hierarchical Clustering

Pada Gambar 9 dapat dijelaskan bahwa pada awal proses clustering terdapat empat mobil yang dapat masukkan 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.

Jenis metode Hierachical Clustering

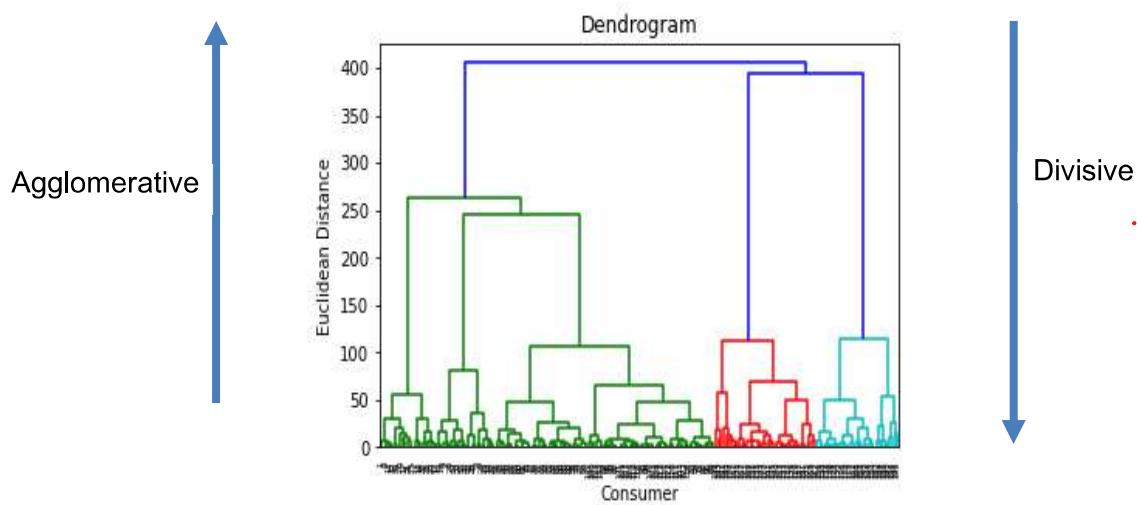
Metode hierarchical clustering dibagi menjadi dua yaitu:

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



Gambar 10. Jenis metode hierarchical clustering

Pada Gambar 10 dapat diamati bahwa metode agglomerative melakukan pengelompokan dari cluster terkecil hingga terbentuk satu cluster besar, sebaliknya metode divisive melakukan clustering dengan cara membagi sebuah cluster besar hingga setiap node di dalam cluster tersebut mewakili satu cluster itu sendiri.

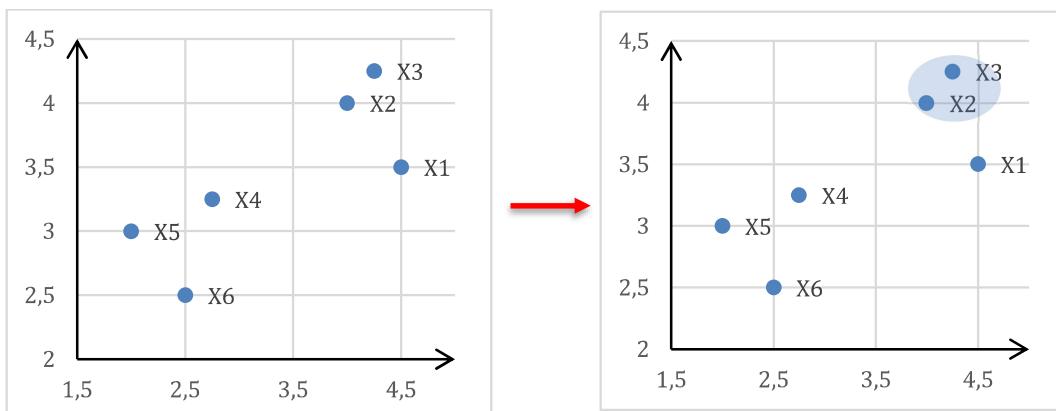
Langkah-langkah metode hierarchical clustering dengan agglomerative:

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.

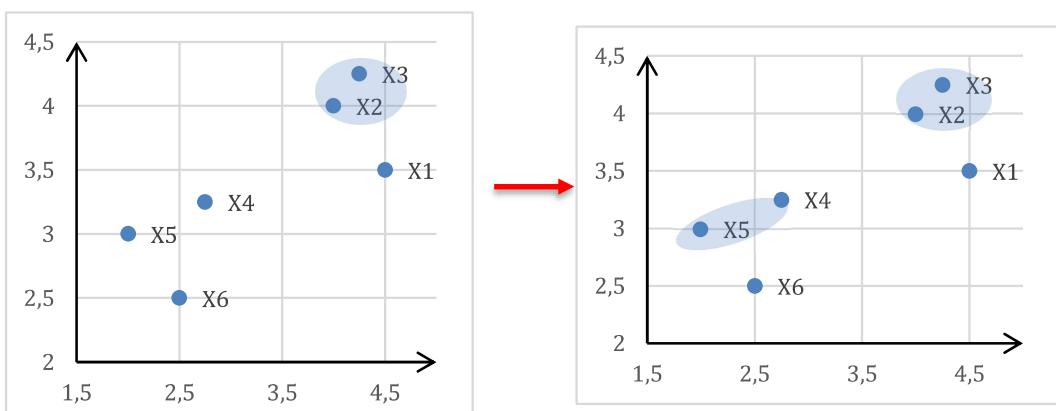
Ilustrasi dari langkah-langkah agglomerative:

1. 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:



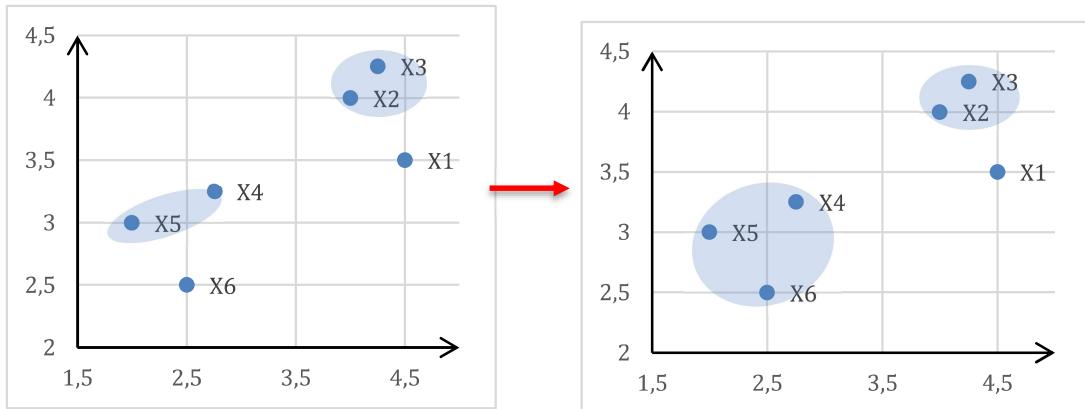
Gambar 11. Menggabungkan dua titik terdekat

2. Langkah kedua adalah proses menggabungkan dua cluster menjadi satu dan dapat masih dilanjutkan jika masih terdapat titik yang terdekat lagi yang memungkinkan untuk digabungkan lagi.



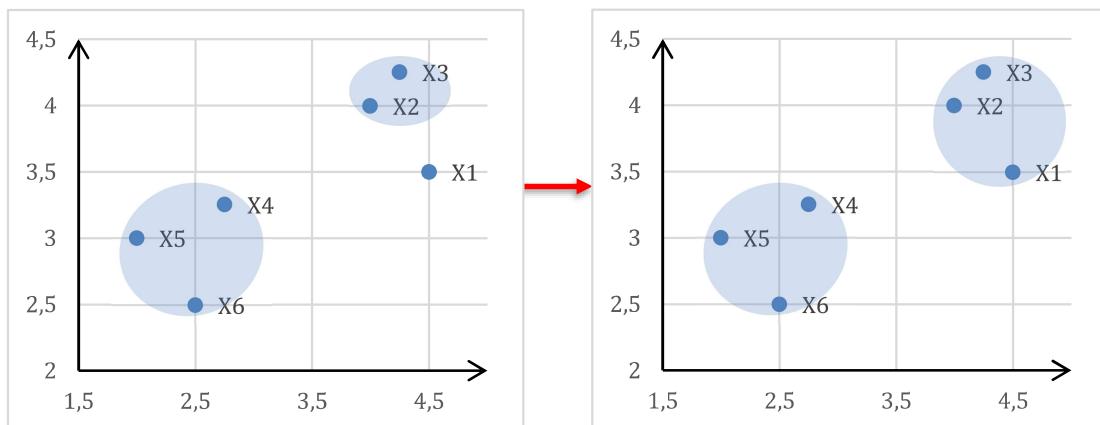
Gambar 12. Mencari titik terdekat berikutnya untuk digabungkan menjadi cluster

3. 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



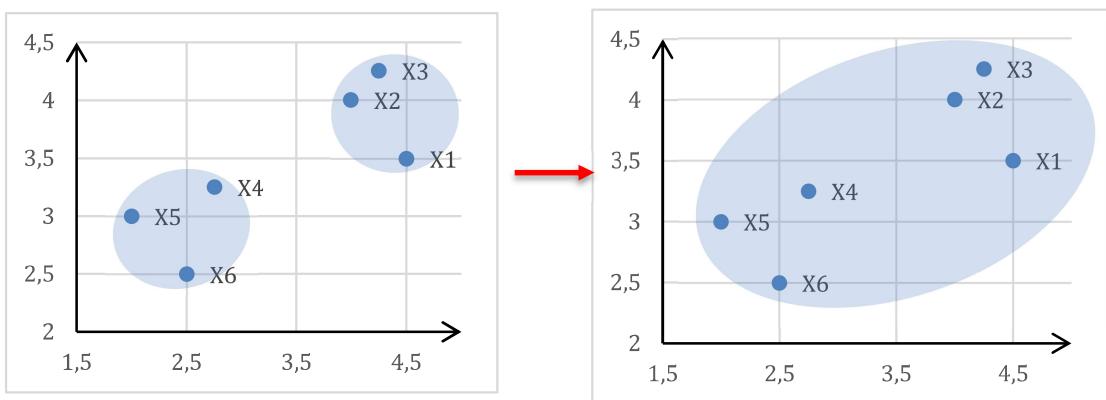
Gambar 13. Menggabungkan X6 ke dalam cluster X4 dan X5

4. 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.



Gambar 14. Menggabungkan X1 ke dalam cluster X2 dan X3

5. Proses akan berakhir ketika semua cluster telah tergabung menjadi satu cluster besar.



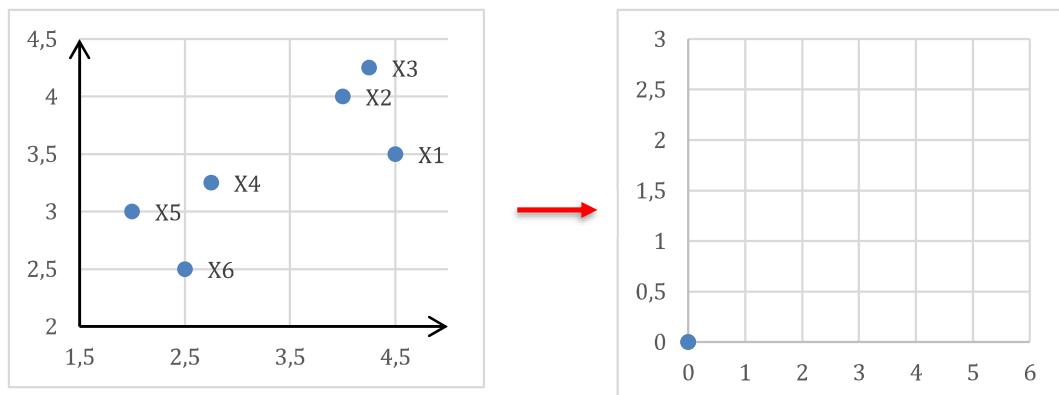
Gambar 15. Menggabungkan seluruh cluster menjadi satu cluster besar

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 euclideananya.

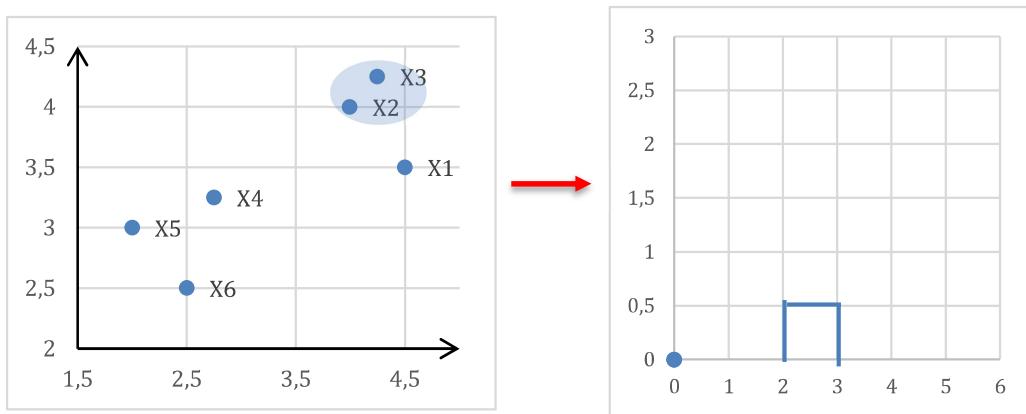
Ilustrasi dari penentuan dendrogram adalah sebagai berikut:

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 dendrogram.



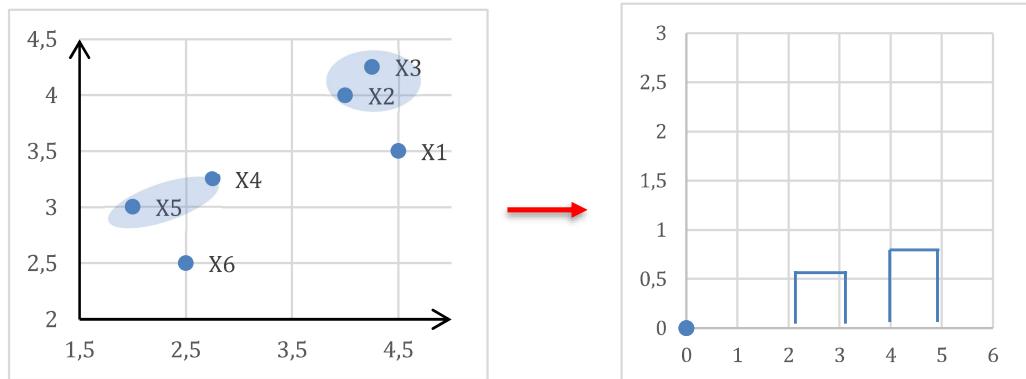
Gambar 16. Transformasi dari grafik dua dimensi ke dendrogram diagram

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



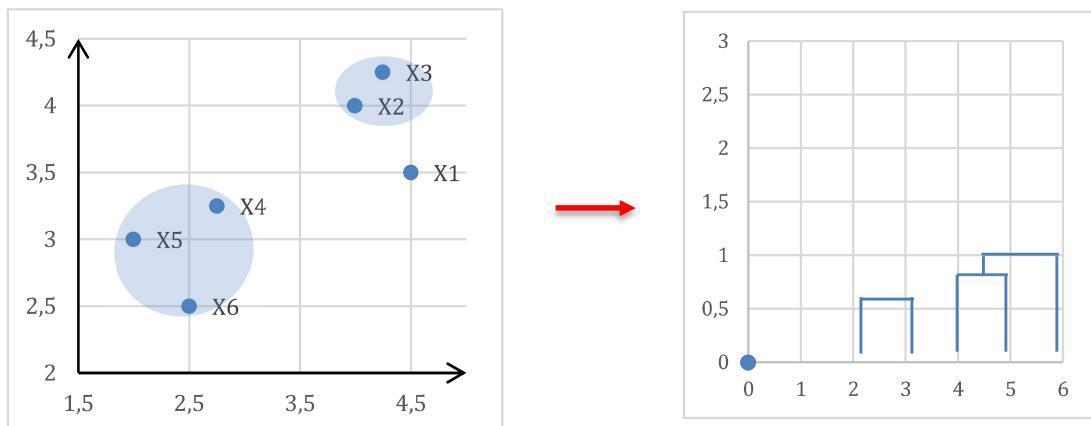
Gambar 17. Transformasi cluster X2X3 ke dalam dendogram diagram

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



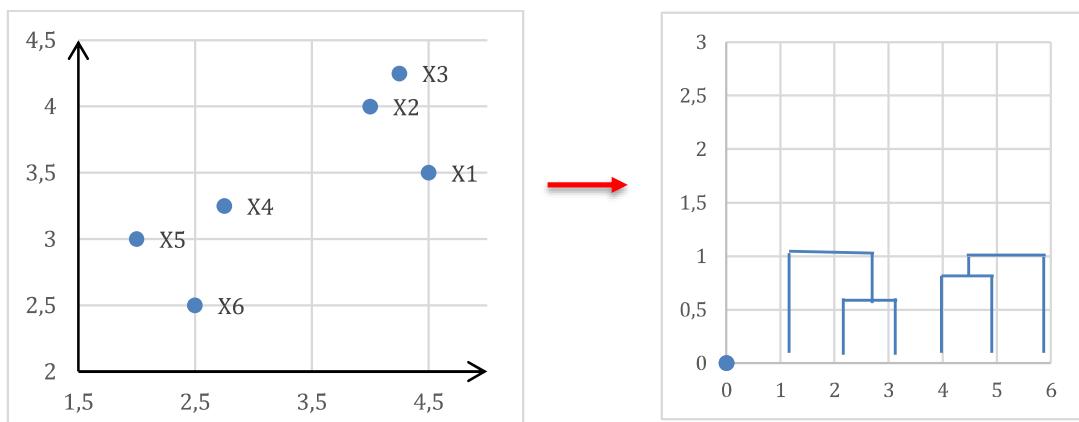
Gambar 18. Dua cluster ditransformasi ke dendogram diagram

4. Proses pada langkah 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 19.



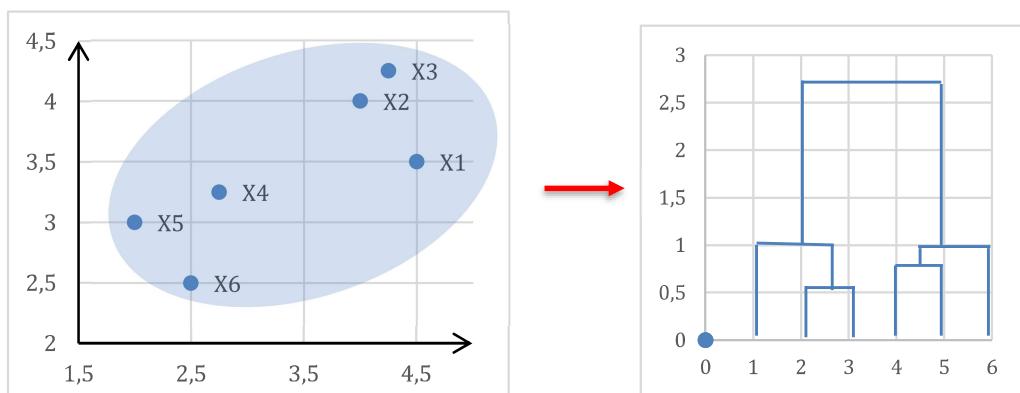
Gambar 19. Menggabungkan dendogram 6 ke dalam dendogram 4-5

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.



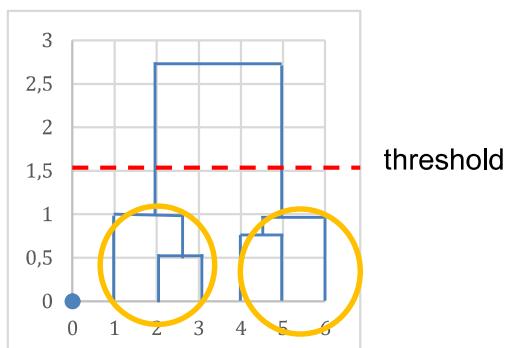
Gambar 20. Menggabungkan X1 ke dalam cluster X2X3

6. Proses akan berhenti setelah semua cluster telah tergabung menjadi satu cluster besar seperti pada Gambar 21.



Gambar 21. Menggabungkan seluruh dendogram menjadi satu dendogram

7. Untuk menentukan berapa jumlah cluster yang paling sesuai pada dataset yang diujikan dapat dianalisa 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 22. Menentukan jumlah cluster

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.

Penyelesaian:

Coding dalam Bahasa pemrograman 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[:, [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()

# 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()

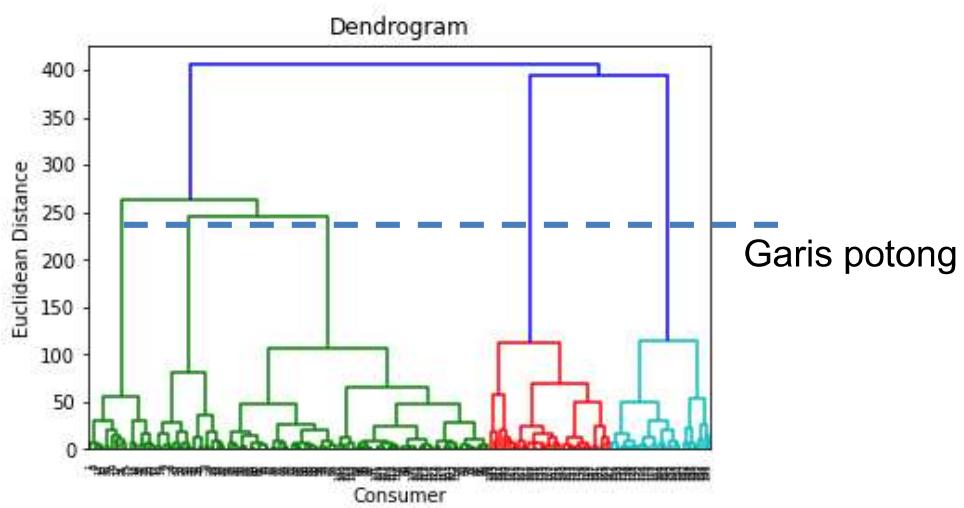
```

Penjelasan coding

- Baris 2 sampai 4 mengimpor *library* yang dibutuhkan.
- Baris 7 mengimpor dataset yang digunakan.
- Baris 8 melakukan slicing yaitu memilih komlom mana yang akan digunakan sebagai data input.
- Baris 11 mengimpor *library* `scipy.cluster.hierarchy` untuk membuat dendrogram diagram.
- Baris 12 mendefinisikan variabel `dendrogram` untuk membuat dendrogram. Paramater yang digunakan adalah `linkage`. Linkage adalah algoritma python yang ada di *hierarchical clustering*.
- Baris 13-16 adalah instruksi untuk menampilkan grafik dapat diamati bahwa grafik di setting dengan warna berbeda untuk membangkitkan dendrogram diagram. Setelah program dijalankan dan diketahui jumlah K yang tepat melalui

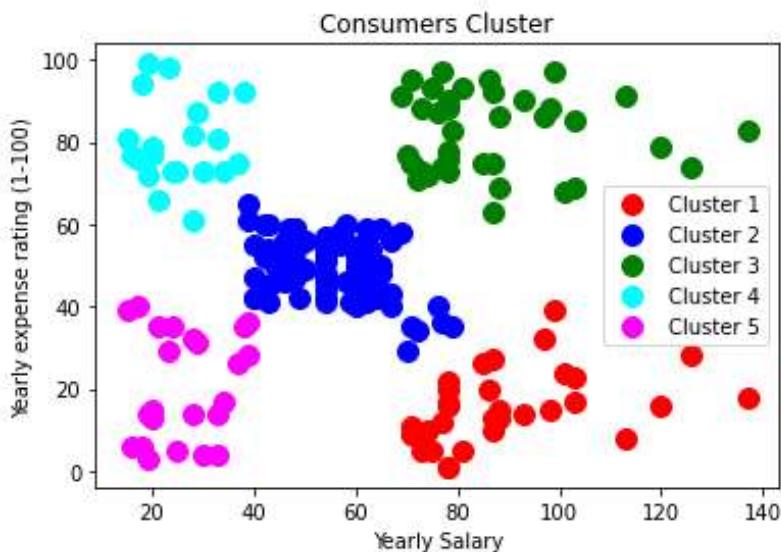
dendogram diagram, langkah selanjutnya adalah melakukan pengolahan Hierachical Clustering sesuai dengan jumlah K yang telah diamati.

- Baris 19 adalah perintah untuk mengimpor *library* AgglomerativeClustering dari sklearn.cluster.
- Baris 20 mendefinisikan variabel untuk membuat model Hierachical Clustering. Untuk parameter pengolahan Hierachical Clustering digunakan N=5 (jumlah K) kemudian pengukuran jarak menggunakan Euclidean Distance dan Metode Ward.
- Baris 24-33 adalah perintah untuk menampilkan hasil clustering. Setiap kelompok didefinisikan dengan warna berbeda agar lebih mudah dianalisa.



Gambar 23. Analisa dendogram diagram untuk menentukan jumlah K

Pada baris coding 13 hingga 16 telah dijelaskan bahwa coding tersebut adalah instruksi untuk memunculkan dendogram diagram. Pada grafik dendogram Gambar 23 dapat diamati bahwa jumlah cluster yang paling sesuai adalah 5 dihitung melalui jumlah garis vertical yang terpotong dari nilai threshold yang telah ditentukan.



Gambar 24. Hasil pengolahan Hierarchical Clustering

Pada gambar Gambar 24 dapat diamati bahwa hasil clustering menggunakan metode hierarchical clustering yang paling sesuai adalah menggunakan K=5. Semua titik dapat tergabung pada cluster dengan baik dan hasil clustering hampir sama hasilnya dengan metode K-Means yang telah diuji coba sebelumnya.

DBSCAN

Pada materi sebelumnya telah dijelaskan teknik clustering berbasis partisi dan hierarki, kedua Teknik ini efisien jika dataset berbentuk normal. Namun ketika data cluster berbentuk arbiter atau ingin mendeteksi cluster out lier, maka DBSCAN merupakan Teknik cluster yang sesuai. **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. Pada Gambar 25 dapat diamati perbedaan hasil clustering menggunakan metode DBSCAN dan K-Means pada data density.

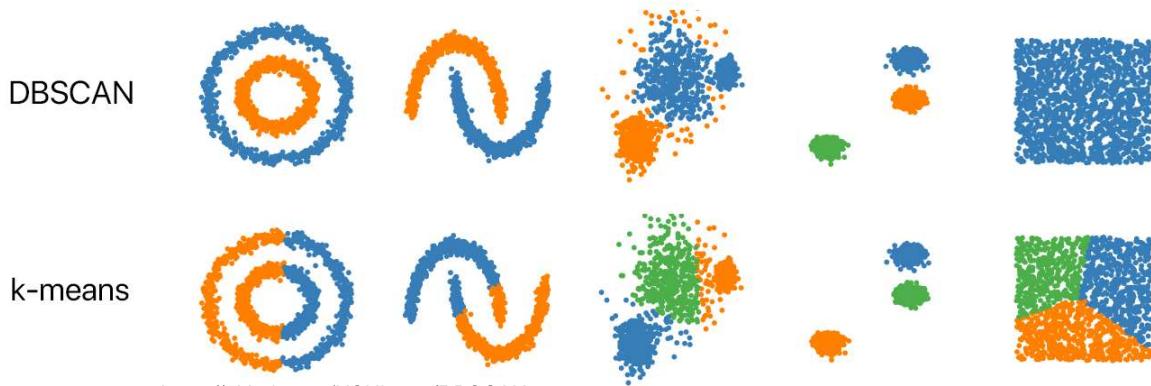


Image source: <https://github.com/NSHipster/DBSCAN>

Gambar 25. Perbedaan clustering menggunakan DBSCAN dan K-Means

Parameter dan Konsep pada metode DBSCAN

Algoritma DBSCAN menggunakan dua parameter yaitu:

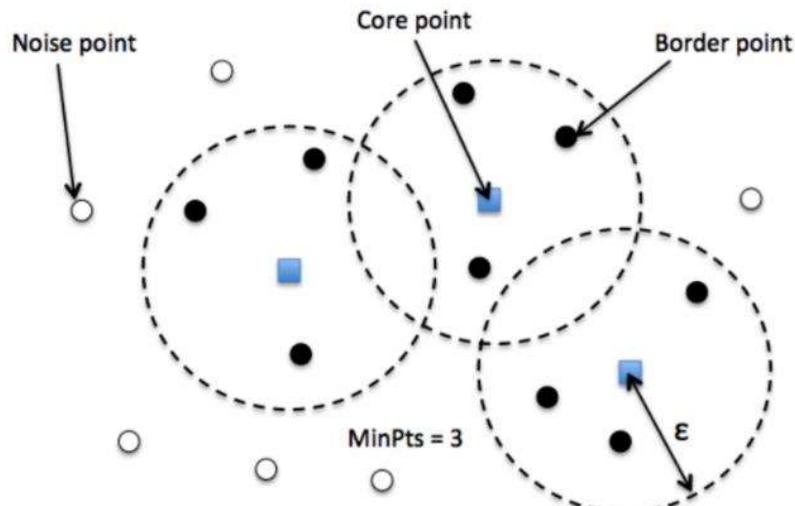
1. **minPts**: Jumlah minimum titik (ambang batas) yang dikelompokkan bersama agar suatu wilayah dianggap density.
2. **eps (ϵ)**: Ukuran jarak yang akan digunakan untuk menemukan titik-titik di sekitar titik mana pun.

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

1. **Reachability** pada konsep ini, digunakan untuk menentukan kepadatan dialukan dengan menetapkan suatu titik yang dapat dijangkau dari yang lain jika terletak dalam jarak tertentu (ϵ) darinya.
2. **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.

Langkah-langkah algoritma DBSCAN:

1. Algoritma dimulai dengan mengambil titik dalam kumpulan data secara random (sampai semua titik telah dikunjungi).
2. Jika setidaknya ada titik 'minPoint' dalam radius " ke titik tersebut, maka dapat dianggap semua titik ini sebagai bagian dari cluster yang sama.
3. Cluster kemudian diperluas dengan mengulangi perhitungan lingkungan secara rekursif untuk setiap titik tetangga
4. Terdapat tiga jenis titik setelah pengelompokan DBSCAN selesai:



Gambar 26. Ilustrasi hasil pengelompokan menggunakan DBSCAN

Pada Gambar 26 dapat dijelaskan bahwa:

1. **Core** adalah titik yang memiliki setidaknya m titik dalam jarak n dari dirinya sendiri.
2. **Border** adalah titik yang memiliki setidaknya satu titik Inti pada jarak n.
3. **Noise** adalah titik yang bukan Core atau Border dan titik tersebut memiliki kurang dari m titik dalam jarak n dari dirinya sendiri.

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.

Penyelesaian:

Coding:

```
#mengimpor library
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_
# jumlah cluster pada 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))

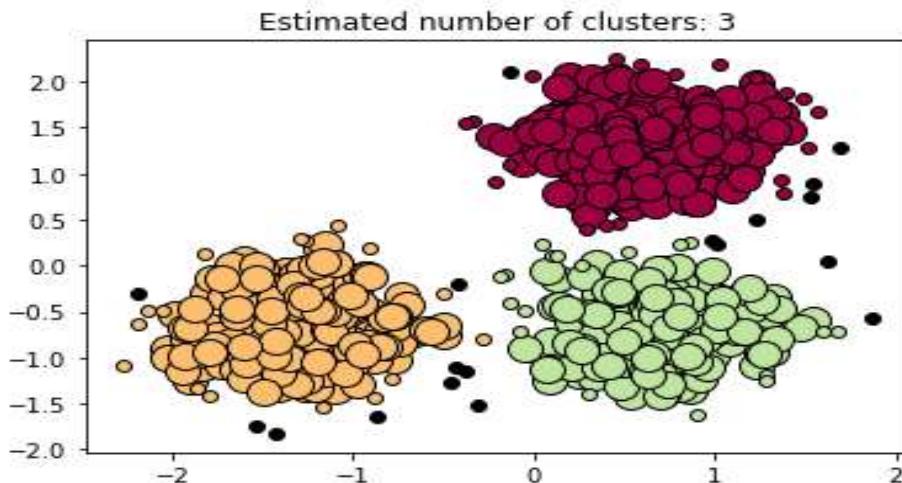
# Hasil Plot
import matplotlib.pyplot as plt

# 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()
```

Hasil running:

```
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
```



Gambar 27. Hasil pengelompokan data spherical menggunakan metode 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.

Penyelesaian:

Coding :

```
#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
```

```

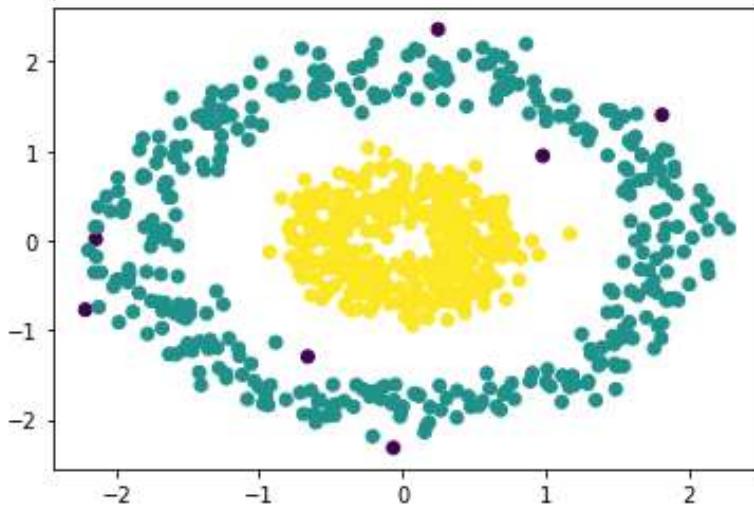
#mengenerate data set
X, y = make_circles(n_samples=750, factor=0.3, noise=0.1)
X = StandardScaler().fit_transform(X)
y_pred = DBSCAN(eps=0.3, min_samples=10).fit_predict(X)
plt.scatter(X[:,0], X[:,1], c=y_pred)

#mencetak hasil pengelompokan
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))

```

Hasil running:

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



Gambar 28. Hasil pengelompokan pada non-spherical data menggunakan DBSCAN

Dapat diamati pada Gambar 27 dan Gambar 28, hasil pengelompokan menggunakan metode DBSCAN dapat memisahkan data dengan baik pada kasus spherical dan non-spherical data. Hasil pengelompokan tersebut akan berbeda jika menggunakan metode

K-Means dikarenakan K-Means memasukkan titik ke dalam clusternya berdasarkan tingkat kedekatannya bukan berdasarkan tingkat density-nya.