MACHINE LEARNING

MEMBANGUN MODEL -3

(Clustering - **Library K-Means, Hierarchical Clustering, DBSCAN**)

Magister Teknik Informatika

Dr. Chairani, S.Kom., M.Eng

IIB DARMAJAYA, 2023/2024

Course Definition

Topik : Membangun Model (Clustering)

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

Leaming Objective

Pada materi kali ini:

A. Melakukan pemilahan data untuk digunakan sebagai input feature

B. Mempelajari algortima clustering K-Means, Hierachical Clustering, DBSCAN

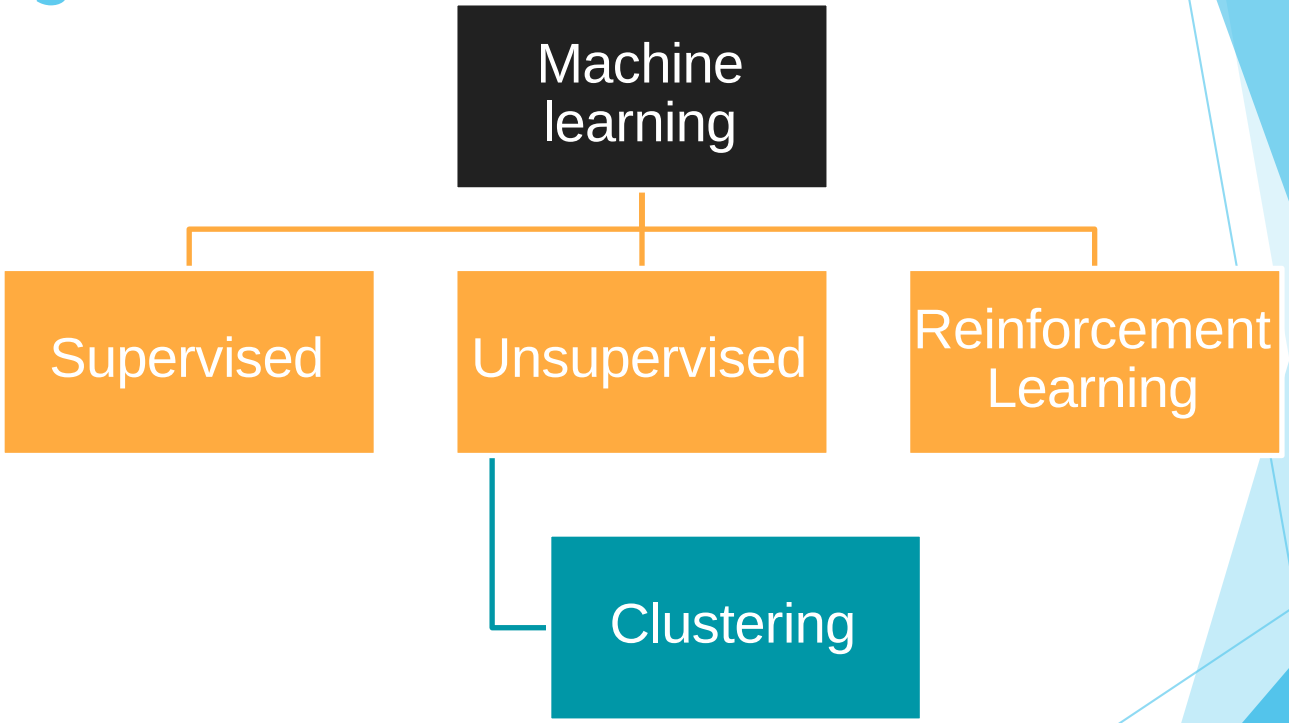
C. Mempelajari optimasi K-Means dengan metode Elbow untuk menentukan jumlah K yang tepat

D. Mempelajari optimasi Hierachical 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


Machine learning
Supervised
Unsupervised
Reinforcement Learning
Clustering

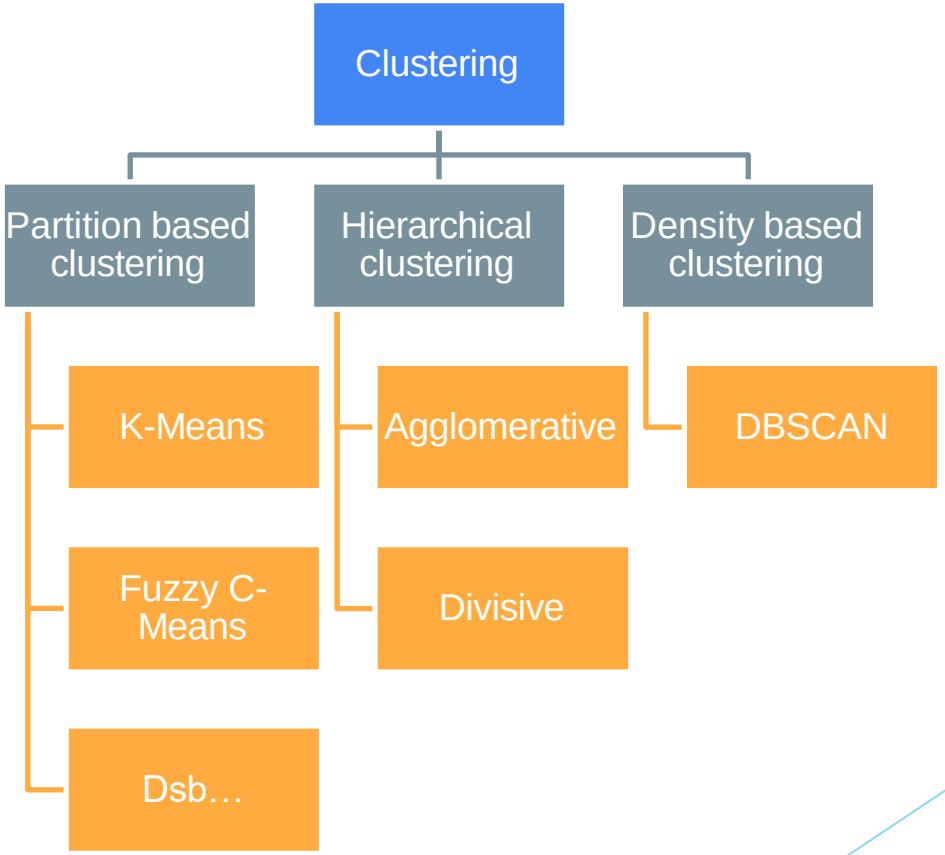
Clustering

Teknik clustering melakukan pembelajaran mesin tanpa ada supervisi untuk menyelesaikan suatu masalah

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
Partition based clustering
Hierarchical clustering
Density based clustering
K-Means
Fuzzy C-Means
Dsb…
Agglomerative
Divisive
DBSCAN

Clustering

Method	General Characteristics
Partitioning Methods	• Find mutually exclusive clusters of spherical shape • Distance based • May use mean to represent cluster • Effective for small to medium data sets
Hierachical Methods	✓ Clustering is a hierarchical decomposition ✓ Cannot correct erroneous merges or split ✓ May incorporate other techniques like micro-clustering or object “linkages”
Density Based methods	• 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

- The clustering of data into groups requires some methods for computing the distance or the similarity between each pair of observations. The result of this computation is known as a dissimilarity or distance matrix.

- The classical methods for distance measures:

Euclidean distance:

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

Manhattan distance:

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

Where, x and y are two vectors of length n

Clustering Distance Measures

Other dissimilarity measures exist such as correlation-based distances, which is widely used for gene expression data analyses.

Pearson correlation distance:

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

Spearman correlation distance:

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

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

K-Means

- Algoritma 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 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

$$w(C_k) = \sum_{x_i \epsilon C_k} (x_i - \mu_k)^2$$

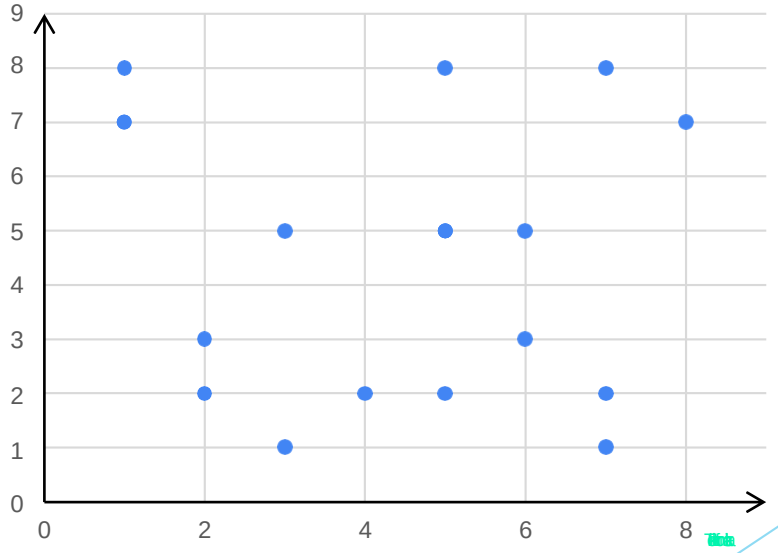
where:

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

Langkah-langkah metode K-Means

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

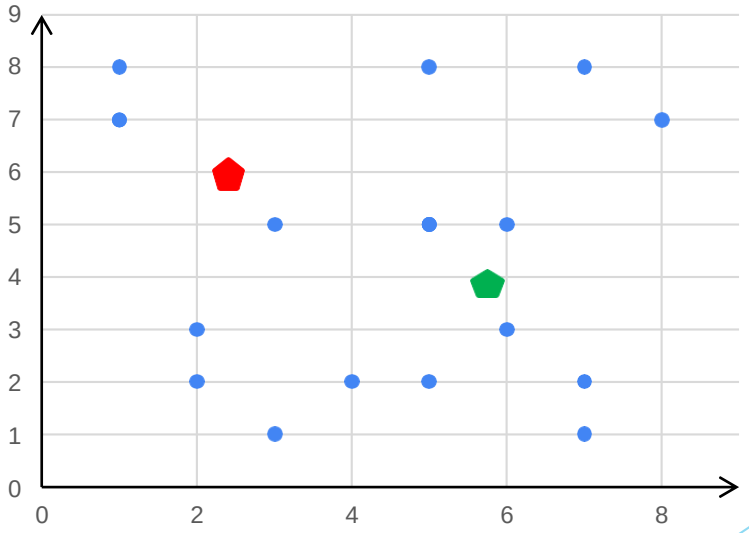
Sebagai contoh terdapat data 2 dimensi seperti yang ditampilkan dalam grafik, langkah pertama adalah memilih jumlah kluster. Misal kita pilih untuk membaginya ke dalam 2 kluster.



Langkah-langkah metode K-Means

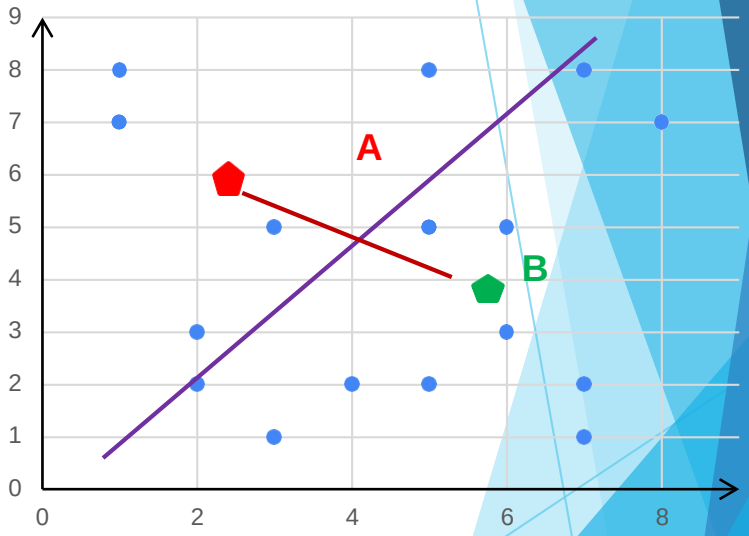
2. Memilih titik secara random sebanyak K buah, di mana titik ini akan menjadi pusat (*centroid*) dari masing-masing kelompok (*clusters*).

Langkah kedua adalah menentukan titik pusatnya. Dalam satu 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



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.

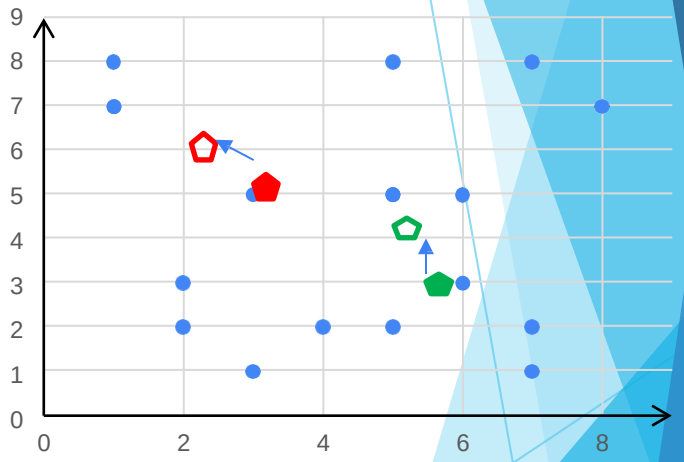


Berdasarkan pengelompokan pada langkah ke dua, maka setiap titik data saat telah tergabung dalam salah satu kluster. Titik data yang diwakili dengan symbol ‘x’ berwarna merah masuk ke kluster 1, dan symbol ‘x’ berwarna biru masuk ke kluster 2, seperti pada berikut:

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 maisng-masing klaster.

Langkah ke-empat adalah melakukan penghitungan sesuai algoritma K-Means, yaitu mencari posisi titik pusat yang paling sesuai untuk sestiap 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



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 pengelompokan 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 klaster:

$$wcss = \underset{P_i \text{ in cluster } 1}{\&} 7arak(P_i, C_1)^2 + \underset{P_i \text{ in cluster } 2}{\&} 7arak(P_i, C_2)^2 + \underset{P_i \text{ in cluster } 3}{\&} 7arak(P_i, C_3)^2$$

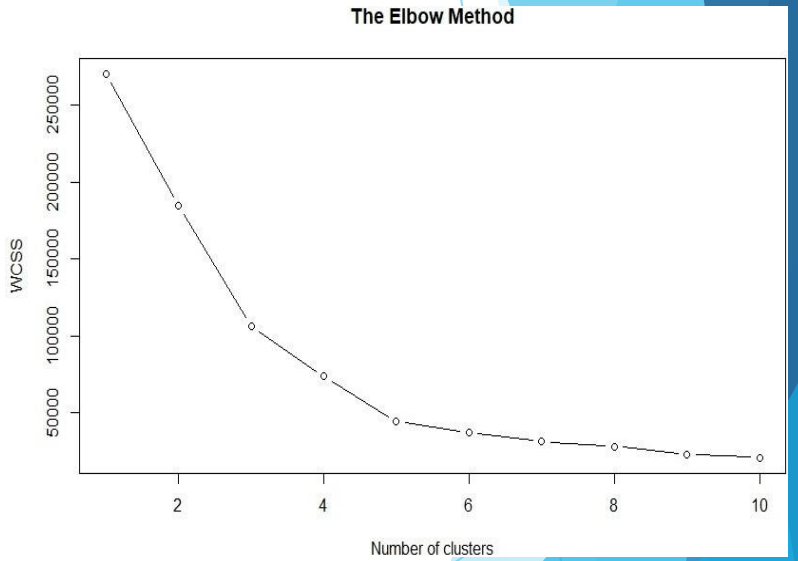
Optimasi K-Means

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 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 kudrat 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 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:



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 (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[:, [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()
```

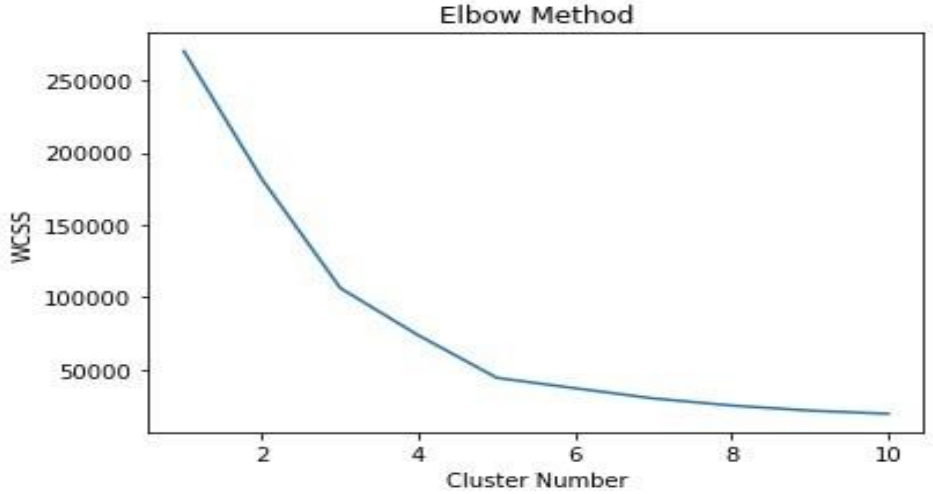
Studi Kasus K-Means

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], 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

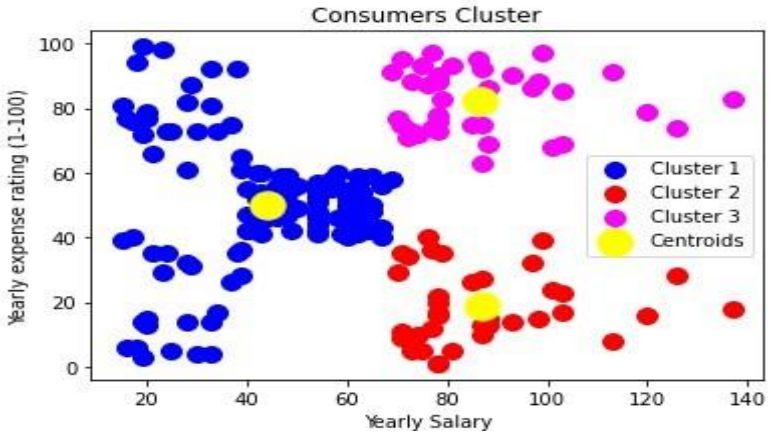
Hasil pengelompokan data menggunakan metode K-Means yang dioptimasi dengan metrics WCSS:

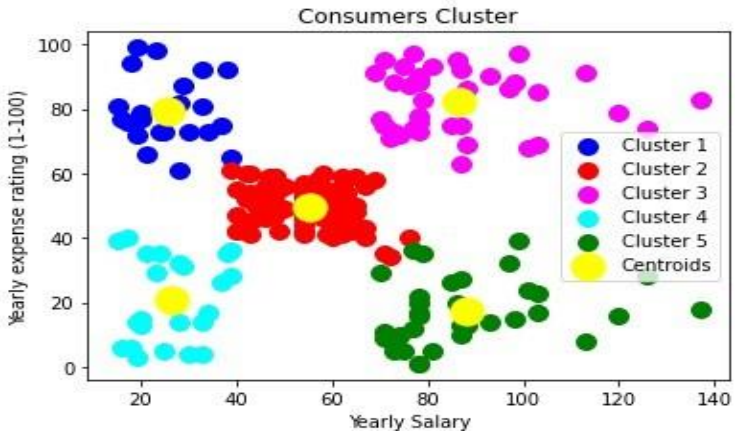


Pada Gambar di atas, dapat diamati bahwa garis grafik yang berbentuk siku terdapat pada klaster 3 dan 5, maka dapat disimpulkan bahwa pembagian klaster sebanyak 3 atau 5 adalah jumlah yang paling optimum

Studi Kasus K-Means

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



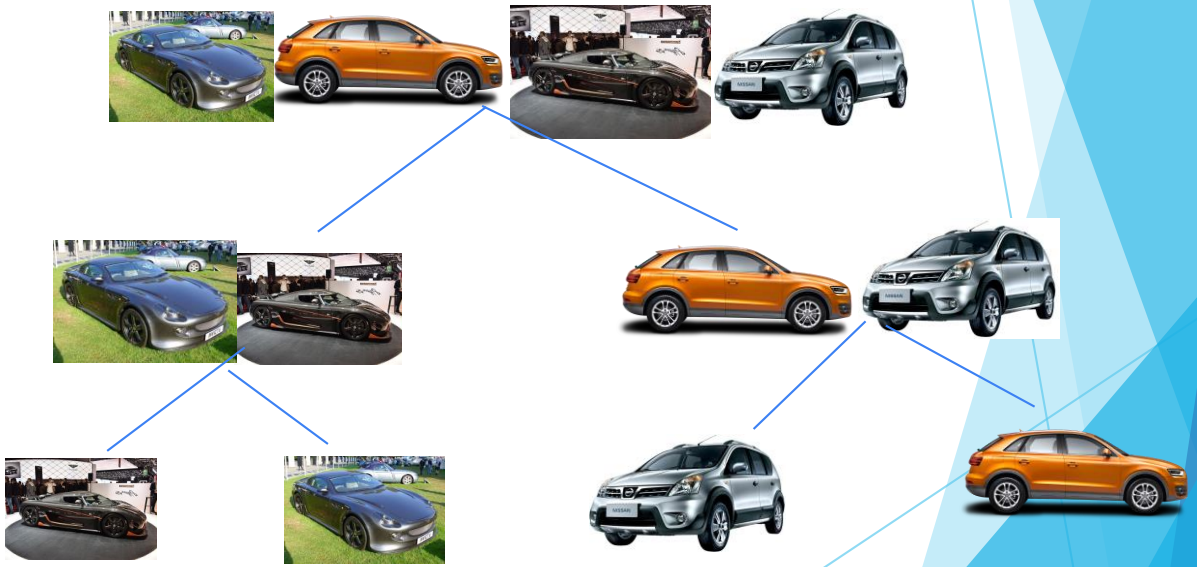


Pada dua gambar di atas dapat diamati hasil pengelompokan data set yang terbagi dalam 3 dan 5 klaster. Hasil dari pengelompokan tersebut dapat digunakan sebagai dukungan keputusan untuk menentukan startegi yang dituju oleh pemilik toko terhadap pelanggannya

Hierachical 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.

Pertama, 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.



Types of Hierachical Clustering

Metode hierarchical clustering dibagi menjadi dua yaitu:

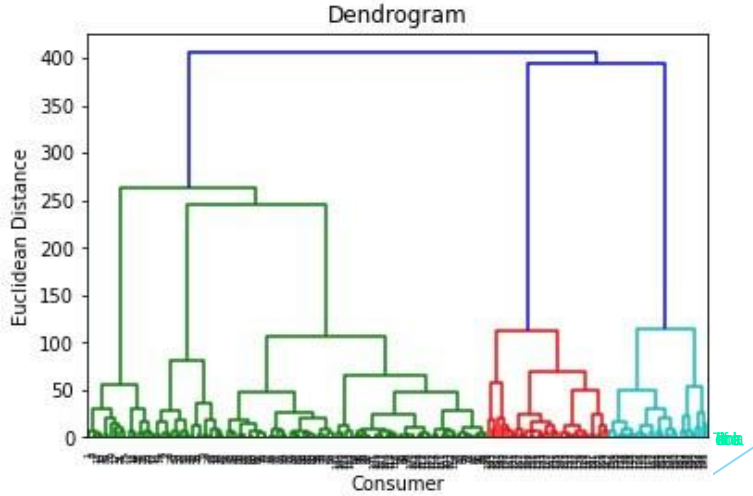
1. Divisive

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

2. Aglomerative

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

Aglomerative



Divisive

Hierachical Clustering

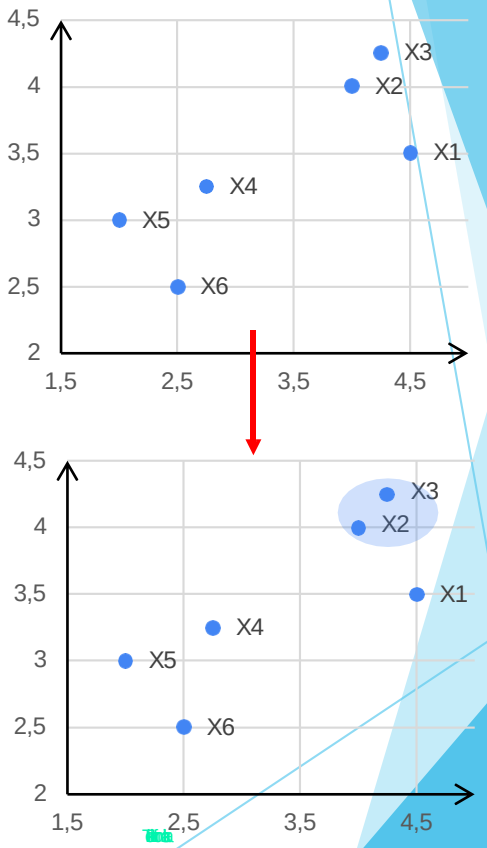
Langkah-langkah metode hierarchical clustering dengan agglomerative:

1. Buat setiap data poin dalam dataset menjadi sebuah cluster, sehi8ngga 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.

Hierachical Clustering (HC)

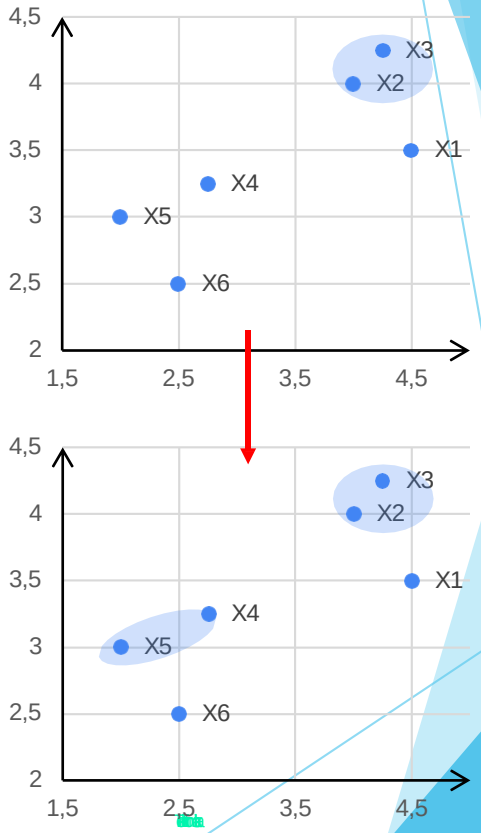
Ilustrasi dari langkah-langkah aglomerative:

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:



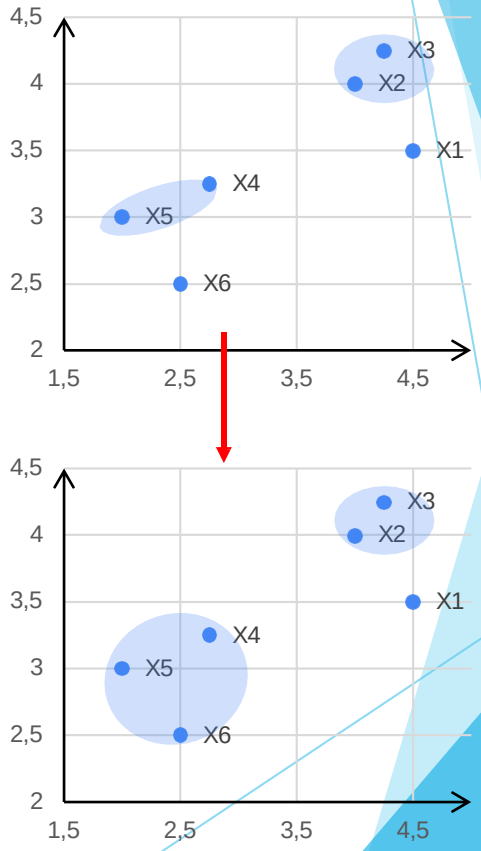
Hierachical Clustering

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



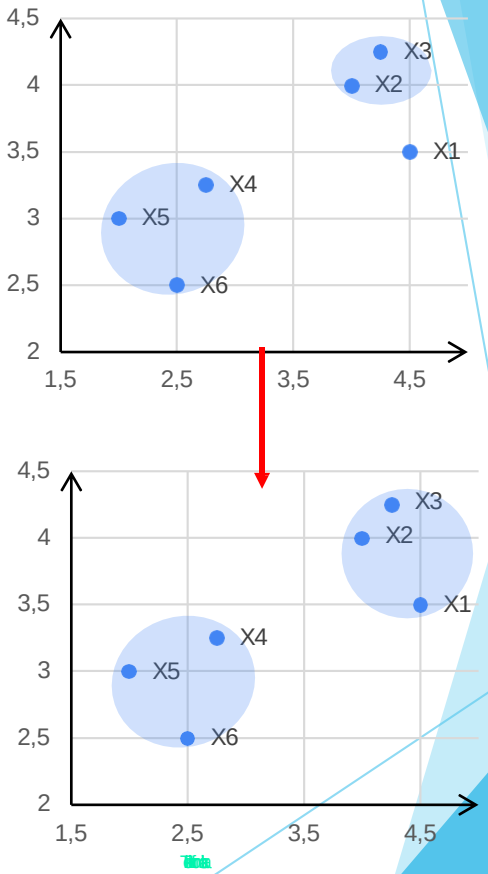
Hierachical Clustering

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.



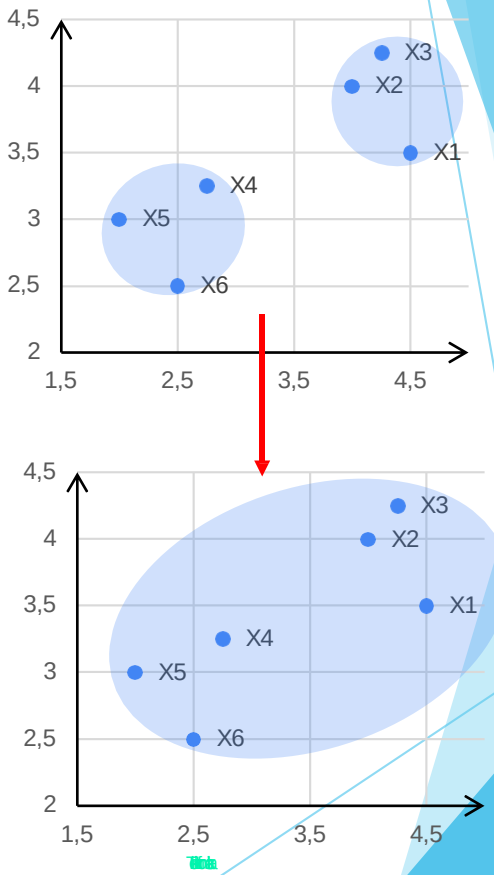
Hierachical Clustering

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.



Hierachical Clustering

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



Optimasi Hierachical Clustering

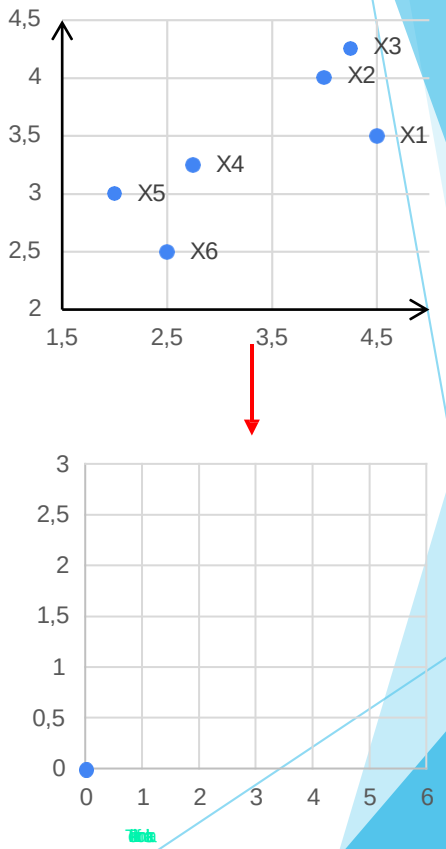
Pada clustering K-Means yang kita pelajari sebelumnya, untuk mengetahui jumlah cluster yang tepat kita dapat menggunakan metode **Elbow**, pada hierarchical clustering kita dapat menggunakan **Dendogram**.

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 euclideannya.

Optimasi Hierachical Clustering

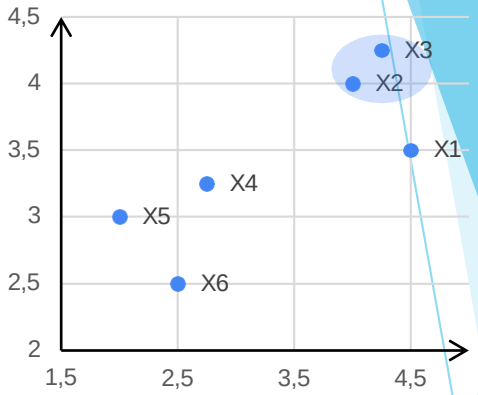
Ilustrasi dari penentuan dendogram adalah sebagai berikut:

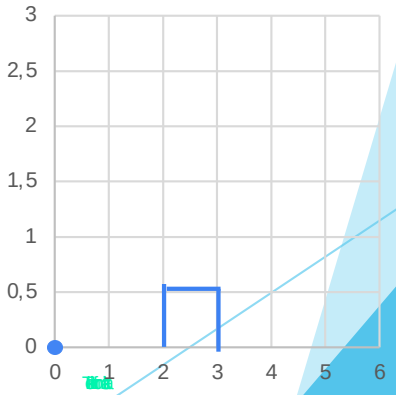
1. Sama seperti pada contoh hierarchical clustering sebelumnya, terdapat enam titik dalam satu diagram. Grafik yang atas adalah grafik awal dan yang bawah adalah grafik dendogram.



Optimasi Hierachical Clustering

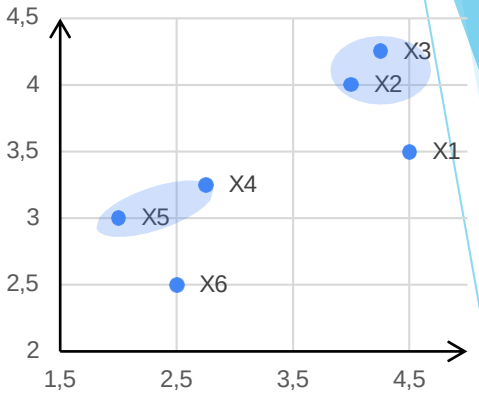
2. Sama seperti ilustrasi pada hierarchical clustering, langkah pertama adalah menentukan dua titik terdekat kemudian menerjemahkannya ke dalam diagram dendogram

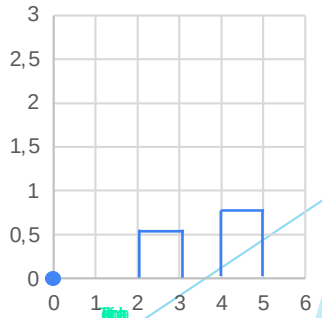




Optimasi Hierachical Clustering

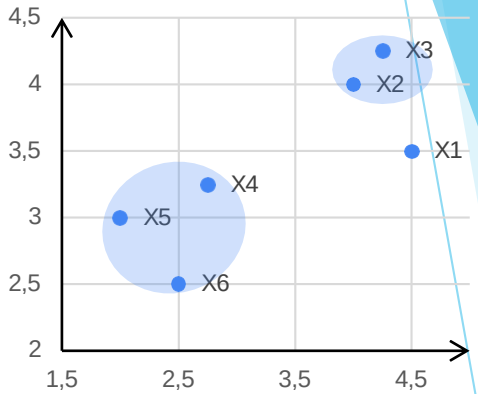
3. Mencari lagi dua cluster yang berdekatan untuk digabungkan menjadi satu cluster lagi. Tinggi diagram Dendogram berbeda-beda sesuai dengan hasil penghitungan Euclidean Distancenya.

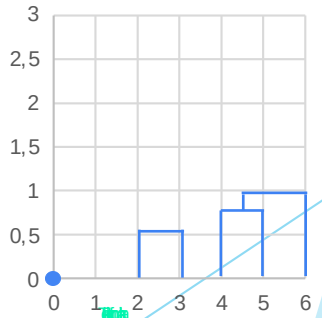




Optimasi Hierachical Clustering

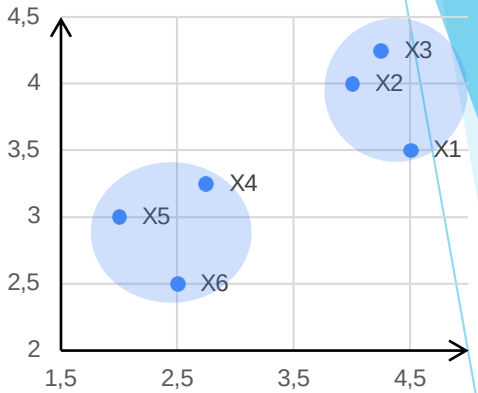
4. Proses yang ketiga diulang lagi dengan mencari dua cluster yang terdekat. Jika dua cluster yang digabungkan sebelumnya adalah cluster antara dua titik maka cara menerjemahkan dalam dendogram dapat diamati pada gambar disamping,

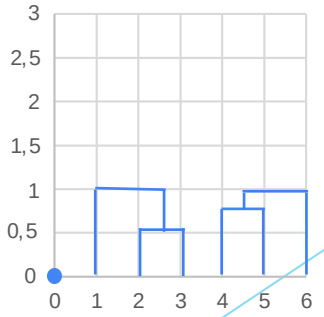




Optimasi Hierachical Clustering

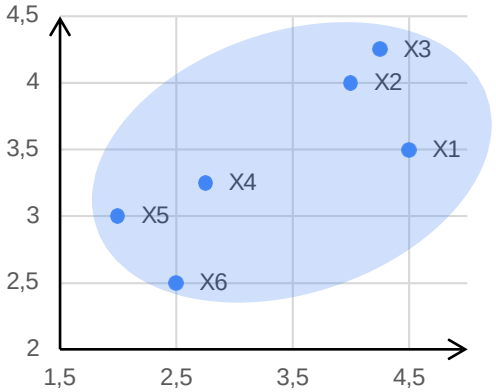
5. Mengulang proses dengan menggabungkan cluster yang sudah ada, Pada gambar disamping tampak bahwa dua cluster terakhir merupakan gabungan dari cluster (dua titik yang menjadi satu cluster) pada proses awal.

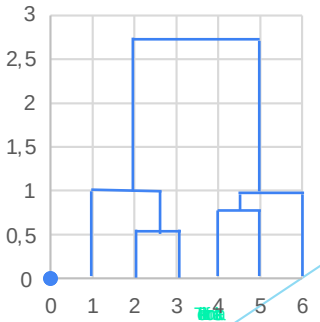




Optimasi Hierachical Clustering

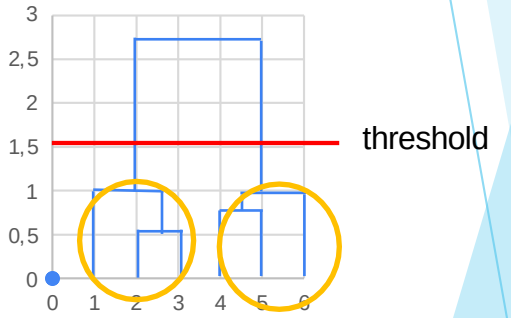
6. Proses akan berhenti setelah semua cluster telah tergabung menjadi satu cluster besar seperti pada gambar di samping:





Optimasi Hierachical Clustering

7. Untuk menentukan berapa jumlah cluster yang paling sesuai pada dats set 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 di samping:



Studi Kasus Hierachical 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.

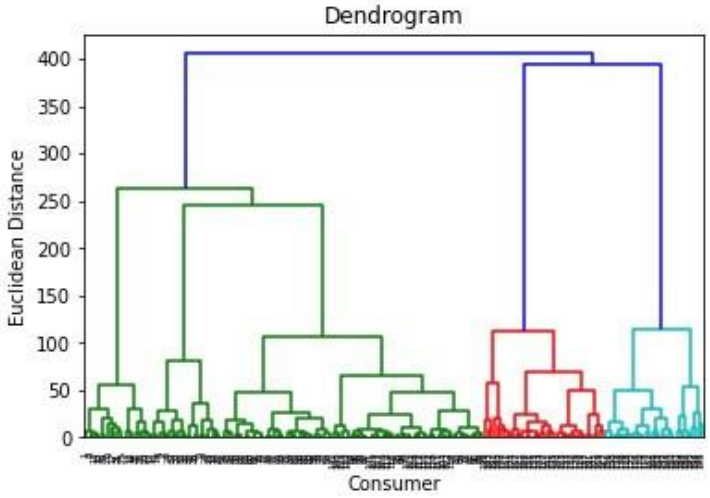
Studi Kasus Hierachical Clustering

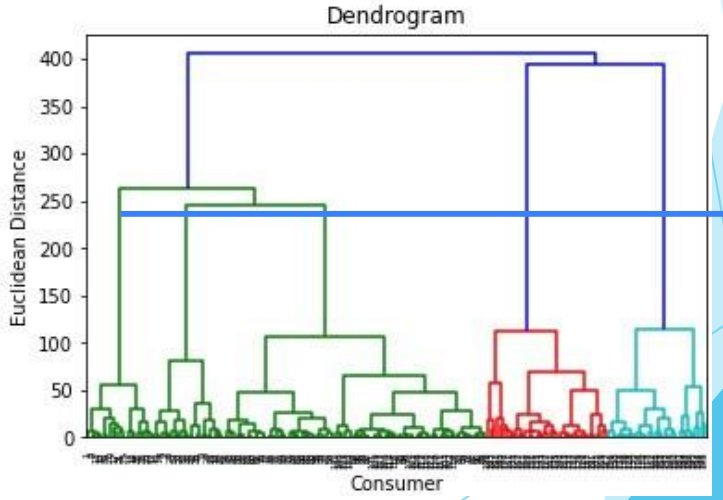
Coding dalam Bahasa pemrograman Python (1):

```
# 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 Hierachical Clustering

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





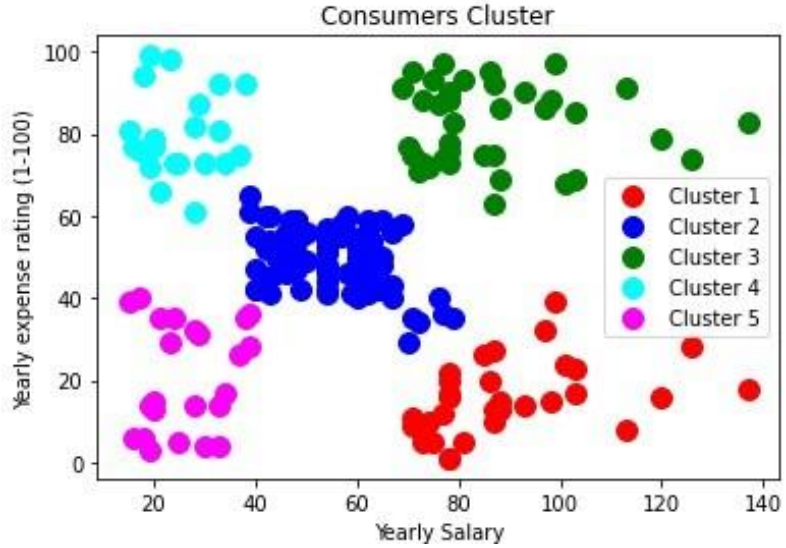
Garis potong

Studi Kasus Hierachical 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 Hierachical 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 out lier, maka DBSCAN merupaka Teknik cluster yang sesuai. Untuk lebih jelasnya dapat diamati pada gambar berikut:

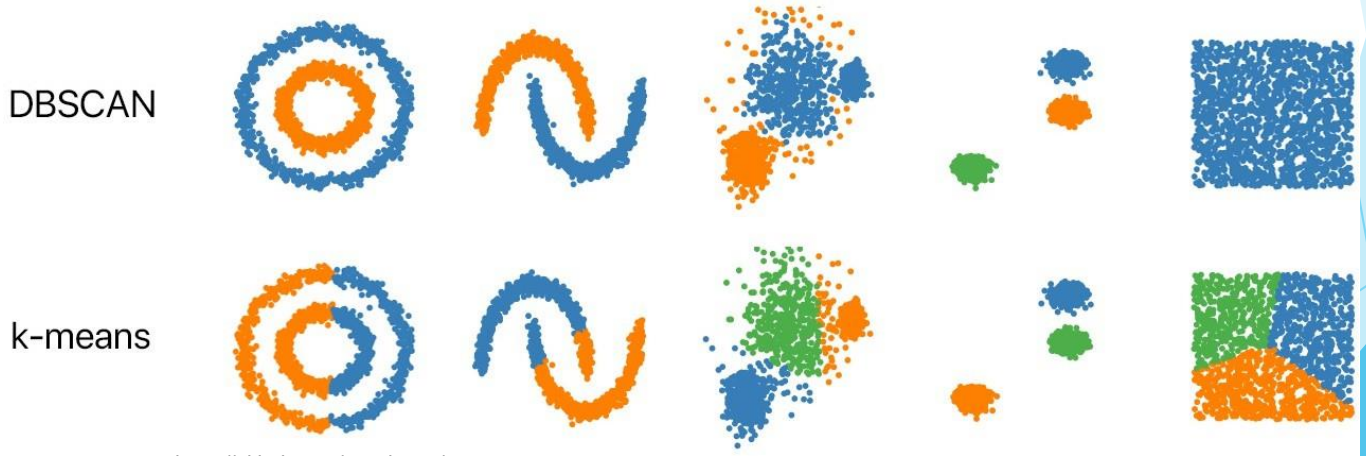


Image source: https://github.com/NSHipster/DBSCAN

DBSCAN

Density-**B**ased **S**patial **C**lustering of **A**pplications with **N**oise (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.

DBSCAN

Algoritma DBSCAN menggunakan dua parameter yaitu:

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

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

DBSCAN

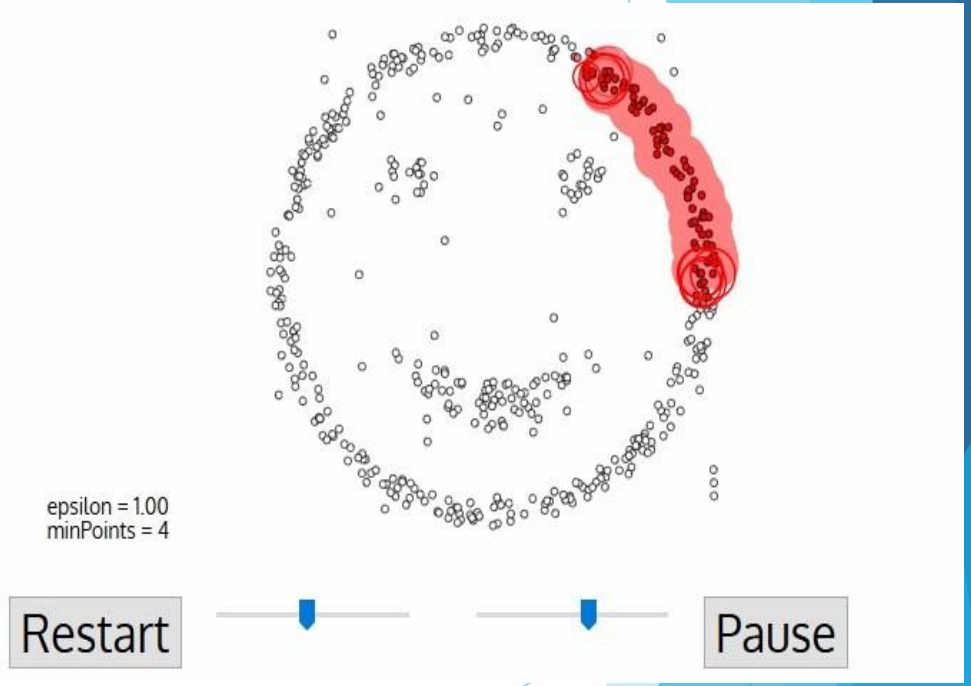
Reachability pada konsep ini, untuk menentukan kepadatan dialukan dengan menetapkan suatu titik yang dapat dijangkau dari yang lain jika terletak dalam jarak tertentu (eps) darinya.

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->r->s->t->q, di mana x->y berarti x berada di sekitar (neighborhood) y.

DBSCAN

Langkah-langkah algortima DBSCAN:

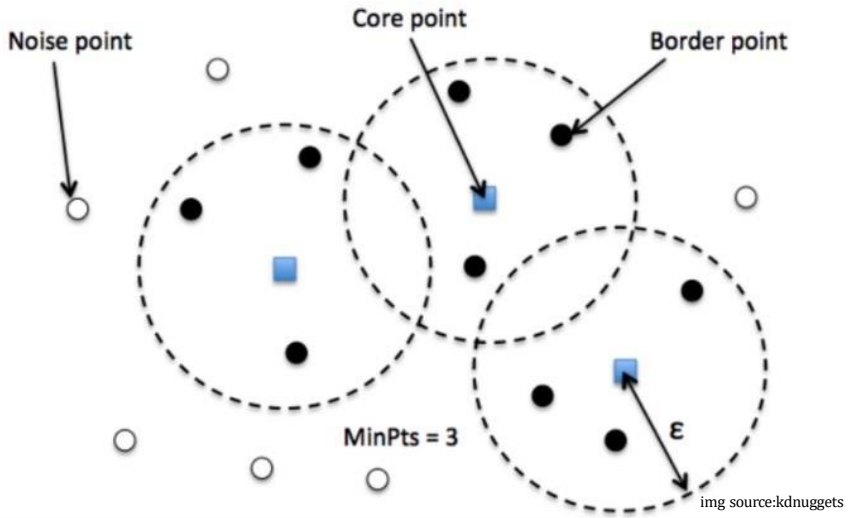
1. Algoritma dimulai dengan mengambil titik dalam kumpulan data secara random (sampai semua titik telah dikunjungi).
2. Jika setidaknya ada titik 'minPoint' dalam radius " ke titik tersebut, maka dapat dianggap semua titik ini sebagai bagian dari cluster yang sama.
3. Cluster kemudian diperluas dengan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



Img source: digitalvidya

DBSCAN

Terdapat tiga jenis titik setelah pengelompokan DBSCAN selesai:



img source:kdnuggets

Core adalah titik yang memiliki setidaknya m titik dalam jarak n dari dirinya sendiri.
Border adalah titik yang memiliki setidaknya satu titik Inti pada jarak n.
Noise adalah titik yang bukan Core atau Border. Dan ia 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.

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

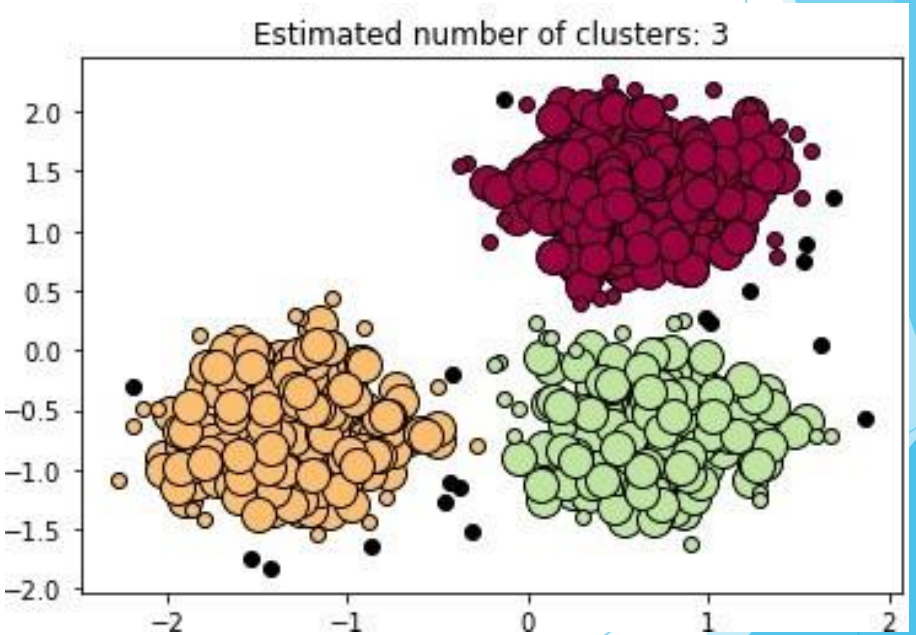
Kasus 1. Coding 3

```
# 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

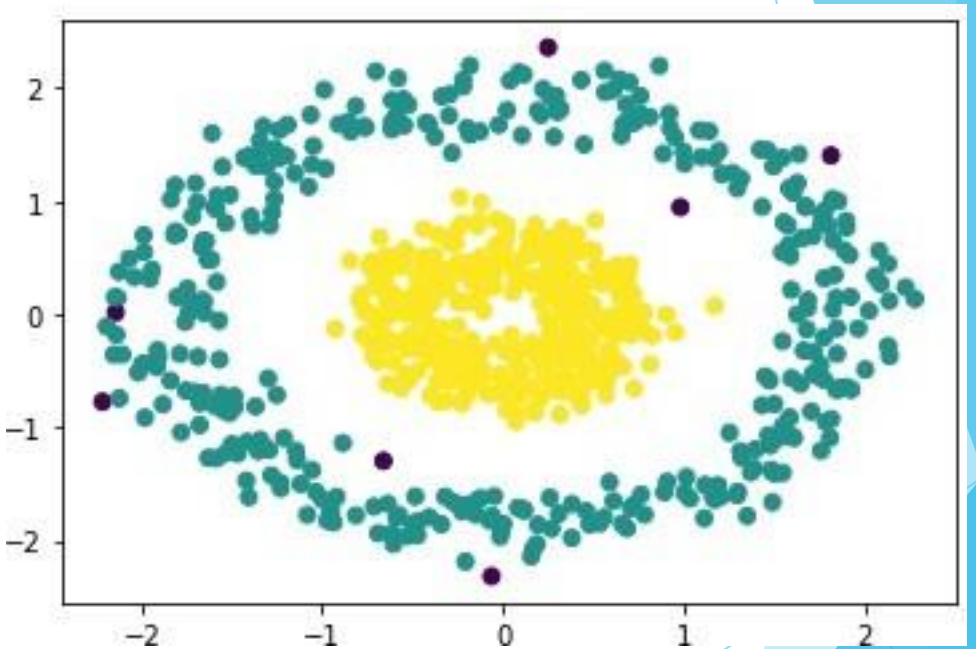
Kasus 2. Coding 1

```
#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

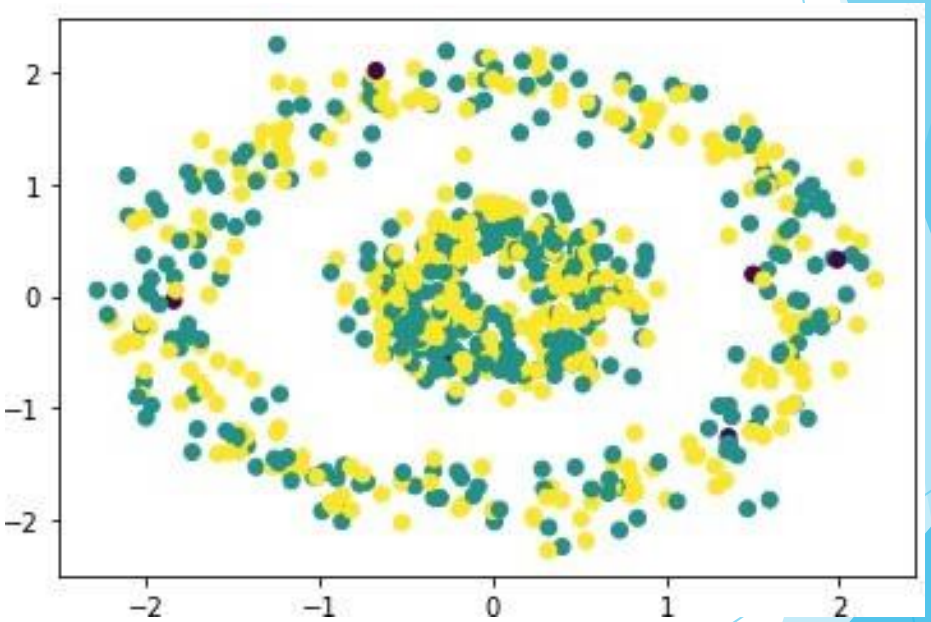
```
Number of clusters: 2
Homogeneity: 1.0000000000000007
Completeness: 0.9372565941867823
V-measure: 0.917
Adjusted Rand Index: 0.952
Adjusted Mutual Information: 0.916
Silhouette Coefficient: -0.061
```



Studi Kasus DBSCAN

Kasus 2. Hasil bila menggunakan metode clustering K-Means

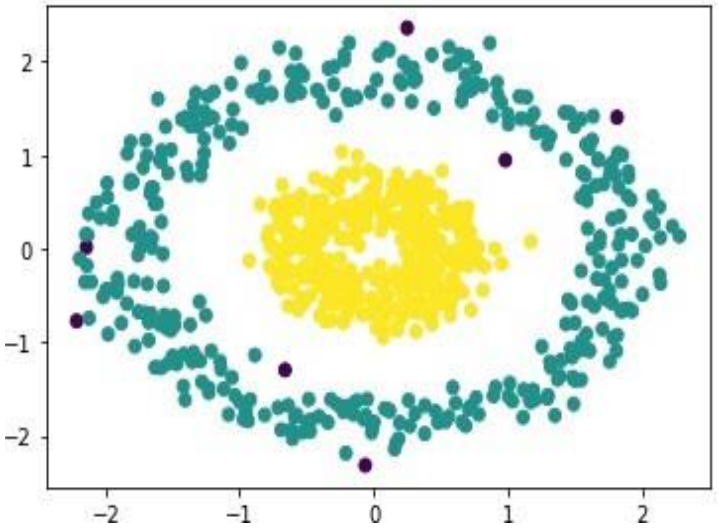
```
Number of clusters: 2
Homogeneity: 0.0016646133996537423
Completeness: 0.0015601698855971465
V-measure: 0.917
Adjusted Rand Index: 0.952
Adjusted Mutual Information: 0.916
Silhouette Coefficient: -0.057
```



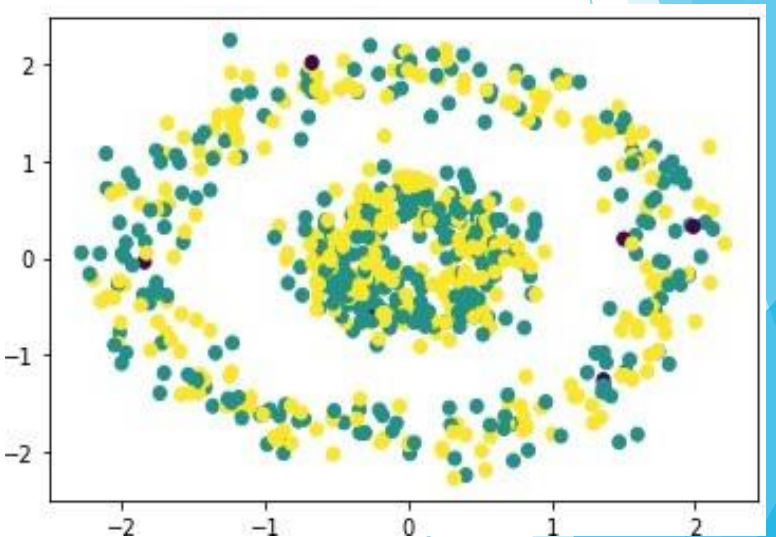
Studi Kasus DBSCAN

Kasus 2. Hasil bila menggunakan metode clustering K-Means

Dapat diamati bahwa DBSCAN menghasilkan hasil cluster yang sesuai



DBSCAN



K-Means

Referensi

- Ethem Alpaydin, “Introduction to Machine Learning, Fourth Edition”, MIT Press, 2020
- Zhi-Hua Zhou, Shaowu Liu, “Machine Learning”, Springer,2021
- Charu C. Aggarwal, Chandan K. Reddy,” Data Clustering“, Chapman and Hall/CRC,2013
- Anil K. Kain, Richard C. Dubes, “Algorithm for Clustering Data”, Michigan State Univesrity, Prentice Hall Advance Reference, 2018 (free download at http://www.cse.msu.edu/~jain/Clustering_Jain_Dubes.pdf

Tools / Lab Online

1. Spyder
2. Jupiter Notebook

TERIMA KASIH