

## Jobsheet 8

### Clustering

#### 1.1. Tujuan Praktikum

Setelah menyelesaikan praktikum ini, mahasiswa mampu:

- Mengetahui tentang konsep Clustering
- Mengimplementasikan Clustering menggunakan Python

#### 1.2. Peralatan yang dibutuhkan

Beberapa peralatan yang dibutuhkan dalam menyelesaikan praktikum ini adalah:

- Dataset terkait

#### 1.3. Dasar Teori

##### 1.3.1. Clustering

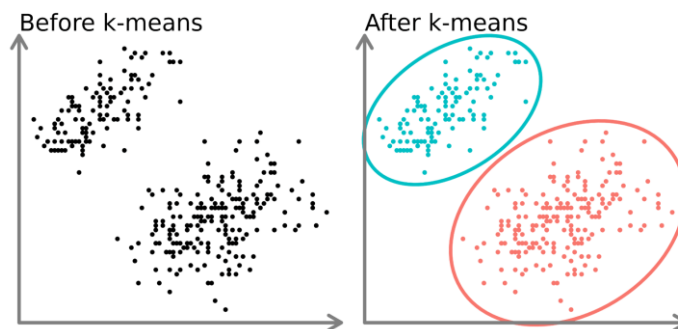
Clustering adalah teknik analisis data yang digunakan untuk mengelompokkan objek atau pengamatan ke dalam kelompok atau klaster yang berbeda, di mana objek dalam klaster yang sama memiliki kesamaan tertentu dan objek di klaster yang berbeda memiliki perbedaan yang signifikan. Tujuan utama dari clustering adalah untuk mengidentifikasi struktur yang tersembunyi atau pola dalam dataset yang tidak terlabel, serta memahami karakteristik alami dari data.

Proses clustering dimulai dengan dataset yang terdiri dari sejumlah objek atau pengamatan, di mana setiap objek direpresentasikan oleh serangkaian atribut atau fitur. Teknik clustering kemudian mencoba untuk menemukan struktur dalam data berdasarkan kesamaan atau jarak antara objek-objek tersebut.

Terdapat berbagai pendekatan yang dapat digunakan dalam clustering, diantaranya:

1. **K-means:** Pendekatan ini mengelompokkan objek berdasarkan pusat kluster yang berbeda. Objek-objek ditempatkan dalam kluster yang memiliki pusat terdekat dengan objek tersebut.

K-Means Clustering adalah salah satu teknik clustering yang paling umum dan sederhana yang digunakan dalam analisis data. Teknik ini bertujuan untuk mengelompokkan titik-titik data ke dalam k kelompok atau kluster berdasarkan kesamaan antara titik-titik data tersebut. K dalam K-Means mewakili jumlah kluster yang diinginkan dan harus ditentukan sebelumnya oleh pengguna.

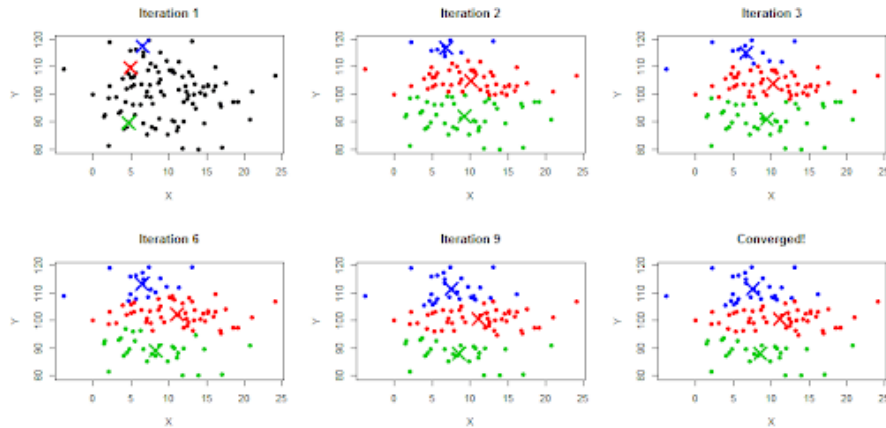


Gambar 1. Ilustrasi Klasterisasi Data Menggunakan K-Means

Berikut adalah langkah-langkah utama dalam K-Means Clustering:

1. **Inisialisasi Pusat Kluster (Centroid):** Langkah pertama dalam K-Means adalah menginisialisasi pusat kluster secara acak. Pusat kluster awal ini dapat dipilih dari titik-titik data secara acak atau dengan menggunakan pendekatan lain seperti K-Means++ yang berusaha untuk memilih pusat kluster awal secara cerdas untuk mempercepat konvergensi.
2. **Pengelompokan Data:** Setelah inisialisasi pusat kluster, setiap titik data ditempatkan dalam kluster yang memiliki pusat terdekat dengan titik data tersebut. Jarak antara titik data dan pusat kluster dapat dihitung menggunakan metrik jarak Euclidean atau metrik jarak lainnya.
3. **Perbarui Pusat Kluster:** Setelah pengelompokan awal, pusat kluster diperbarui dengan menghitung rata-rata dari semua titik data dalam setiap kluster. Pusat kluster baru ini kemudian digunakan sebagai pusat untuk kluster pada iterasi berikutnya.
4. **Iterasi:** Langkah pengelompokan dan pembaruan pusat kluster dilakukan secara berulang hingga kriteria berhenti tercapai. Kriteria berhenti ini bisa berupa jumlah iterasi maksimum yang telah dicapai, perubahan pusat kluster yang tidak signifikan antara iterasi, atau kriteria lainnya.

5. **Evaluasi Hasil:** Setelah konvergensi, hasil clustering dievaluasi menggunakan metrik evaluasi seperti SSE (Sum of Squared Errors) atau Silhouette Score untuk mengevaluasi kualitas kluster yang dihasilkan.



Gambar 2. Tahapan Clustering menggunakan K-Means

Beberapa hal yang perlu diperhatikan dalam K-Means Clustering adalah:

- K-Means sangat sensitif terhadap inisialisasi pusat kluster awal. Hasil clustering dapat bervariasi tergantung pada inisialisasi ini.
- K-Means biasanya lebih baik digunakan untuk data yang berbentuk bulat dan memiliki varian yang seragam dalam setiap dimensinya.
- K-Means tidak dapat menangani kluster dengan bentuk yang kompleks atau tidak teratur.
- K-Means dapat memiliki masalah dengan kluster berukuran yang tidak seimbang atau dengan keberadaan outlier.

Meskipun sederhana, K-Means Clustering tetap menjadi salah satu algoritma clustering yang paling sering digunakan karena efisiensinya dalam menangani data yang berukuran besar dan kemampuannya untuk menghasilkan kluster yang jelas dan terdefinisi dengan baik.

2. **Hierarchical clustering:** Teknik ini menghasilkan dendrogram yang menunjukkan hubungan hierarkis antara kluster. Objek-objek dikelompokkan berdasarkan kedekatannya satu sama lain, dan kluster yang serupa digabungkan secara bertahap. Berikut adalah langkah-langkah utama dalam hierarchical clustering menggunakan pendekatan agglomerative:

1. **Inisialisasi Klaster:** Setiap titik data awalnya dianggap sebagai klaster tunggal.
2. **Perhitungan Jarak Antar-Klaster:** Jarak antara setiap pasangan klaster dihitung. Berbagai metrik jarak dapat digunakan, seperti jarak Euclidean, jarak Manhattan, atau jarak Mahalanobis.
3. **Penggabungan Klaster:** Dua klaster yang memiliki jarak terdekat digabung menjadi satu klaster baru. Proses ini diulangi hingga semua titik data tergabung dalam satu klaster besar.
4. **Pembentukan Dendrogram:** Dendrogram dibentuk untuk merepresentasikan struktur hierarkis dari klaster. Dendrogram adalah representasi grafis dari proses penggabungan klaster, di mana sumbu vertikal menunjukkan tingkat kesamaan antar-klaster dan sumbu horizontal menunjukkan klaster atau titik data mana yang digabung pada setiap langkah.
5. **Memilih Jumlah Klaster:** Setelah dendrogram terbentuk, pengguna dapat memilih jumlah klaster dengan memotong dendrogram pada tingkat yang sesuai dengan tujuan analisisnya.

Hierarchical clustering memiliki beberapa kelebihan dan kekurangan:

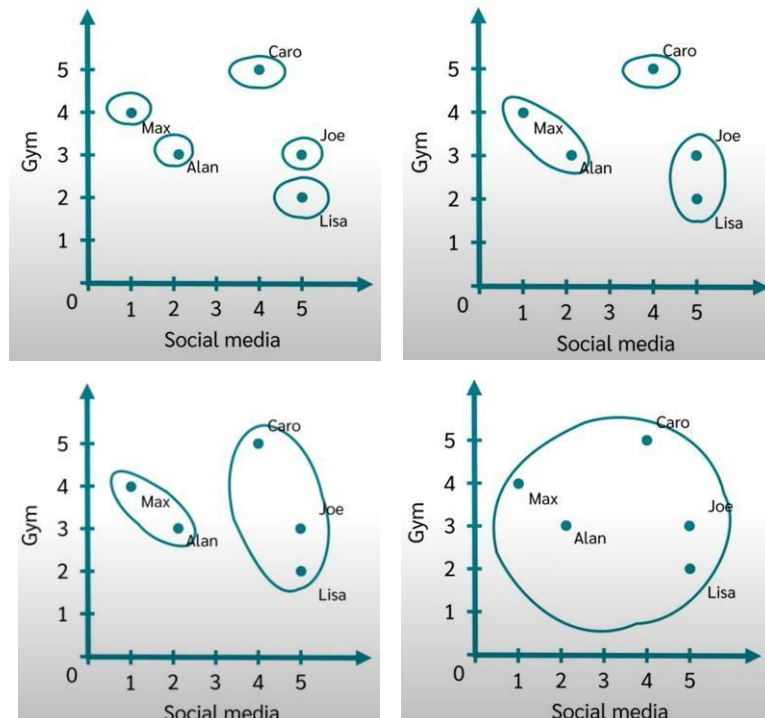
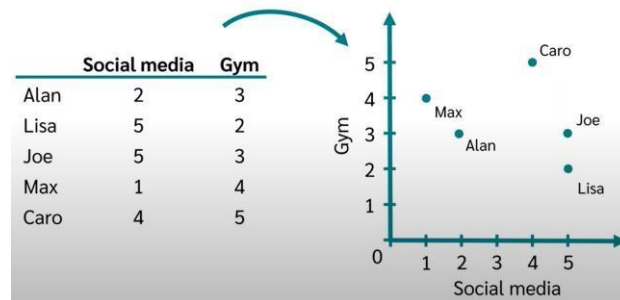
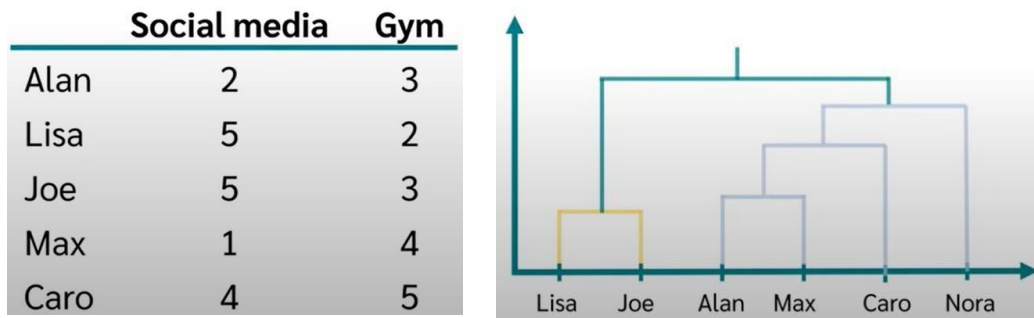
- **Kelebihan:**

- Tidak memerlukan jumlah klaster yang ditentukan sebelumnya, sehingga cocok untuk dataset di mana jumlah klaster tidak diketahui.
- Memungkinkan visualisasi struktur hierarkis dari klaster menggunakan dendrogram.
- Mampu menangani klaster dengan bentuk yang kompleks atau tidak teratur.

- **Kekurangan:**

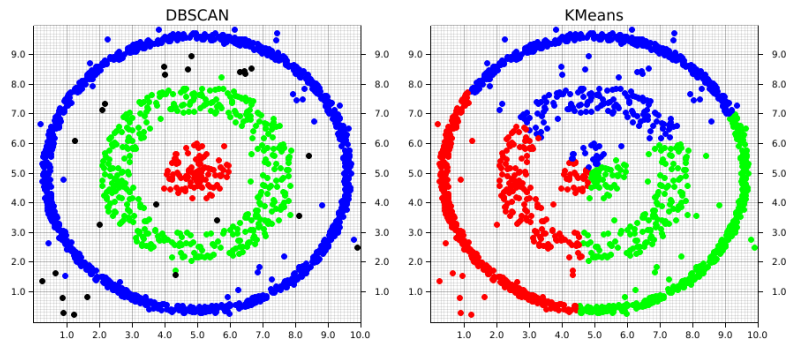
- Membutuhkan waktu komputasi yang lebih lama, terutama untuk dataset yang besar.
- Sensitif terhadap metrik jarak dan metode penggabungan klaster.
- Memerlukan pemilihan parameter yang tepat, seperti metode penggabungan klaster dan kriteria pemotongan dendrogram.

Meskipun memiliki keterbatasan tertentu, hierarchical clustering tetap menjadi salah satu algoritma clustering yang berguna dan populer karena kemampuannya untuk menghasilkan hierarki klaster yang dapat memberikan wawasan yang berharga tentang struktur data.



Gambar 3. Tahapan pada Hierarchical Clustering

3. **Density-based clustering:** Density-Based Clustering merupakan teknik clustering yang mengelompokkan titik-titik data berdasarkan kepadatan relatif mereka dalam ruang fitur. Teknik ini berfokus pada identifikasi wilayah yang padat dalam data, di mana titik-titik data yang berdekatan dan memiliki kepadatan yang tinggi dianggap sebagai bagian dari kluster, sementara titik-titik data yang terisolasi dianggap sebagai noise atau outlier.



Gambar 4. Tampilan Perbedaan Hasil Menggunakan K-Means dan DBScan

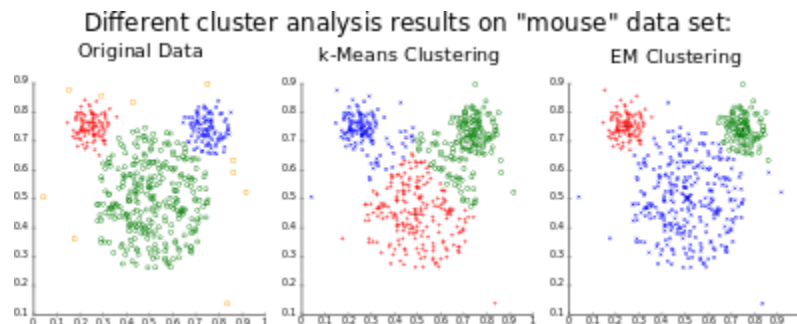
Salah satu algoritma density-based clustering yang paling umum digunakan adalah DBSCAN (Density-Based Spatial Clustering of Applications with Noise). Berikut adalah prinsip kerja DBSCAN:

1. **Pengelompokan Berdasarkan Kepadatan:** DBSCAN mengidentifikasi wilayah yang padat dalam data dengan menentukan titik-titik inti (core points). Sebuah titik inti dianggap sebagai bagian dari kluster jika terdapat jumlah tetangga minimum (MinPts) di sekitarnya dalam jarak tertentu (epsilon).
2. **Eksansi Klaster:** Setiap titik inti dan tetangganya yang berdekatan di dalam wilayah kepadatan yang sama dianggap sebagai bagian dari kluster yang sama. Titik-titik yang berdekatan tetapi bukan titik inti atau tetangga dari titik inti lain dianggap sebagai batas (border points) dan juga dimasukkan ke dalam kluster yang sesuai.
3. **Penghapusan Noise:** Titik-titik yang tidak termasuk dalam wilayah kepadatan dan tidak memiliki tetangga yang cukup banyak dianggap sebagai noise dan dikeluarkan dari proses clustering.

Keuntungan utama dari density-based clustering, khususnya DBSCAN, adalah kemampuannya untuk menangani kluster dengan bentuk yang kompleks dan tidak teratur, serta mampu mengatasi noise dalam data. Selain itu, DBSCAN tidak memerlukan jumlah kluster yang ditentukan sebelumnya dan dapat bekerja efisien dengan dataset besar. DBSCAN juga memiliki beberapa kelemahan, termasuk sensitif terhadap parameter seperti epsilon dan MinPts, serta kinerjanya yang kurang baik dalam menangani dataset dengan perbedaan kepadatan yang signifikan di dalamnya.

Meskipun demikian, density-based clustering tetap menjadi pilihan yang kuat untuk aplikasi di mana struktur kluster yang kompleks dan adanya noise dalam data merupakan tantangan utama, seperti dalam analisis data spasial, deteksi anomali, dan segmentasi objek dalam citra.

4. **Model-based clustering:** Teknik ini mengasumsikan bahwa data dihasilkan dari sejumlah distribusi probabilitas yang berbeda. Model statistik digunakan untuk menentukan seberapa baik data cocok dengan distribusi tertentu, dan objek dikelompokkan berdasarkan model yang terbaik.



Gambar 5. Tampilan perbedaan hasil menggunakan K-Means dengan EM

Salah satu algoritma model-based clustering yang paling umum digunakan adalah Gaussian Mixture Models (GMM). Berikut adalah prinsip kerja GMM:

1. **Inisialisasi Model:** Awalnya, parameter-model GMM (misalnya, pusat, kovariansi, dan bobot klaster) diinisialisasi secara acak atau dengan menggunakan pendekatan tertentu seperti K-Means untuk mempercepat konvergensi.
2. **Expectation-Maximization (EM):** GMM menggunakan algoritma Expectation-Maximization (EM) untuk memperbarui parameter-model secara iteratif. Langkah ini melibatkan dua tahap:
  - a. **Ekspektasi (Expectation):** Di tahap ini, digunakan untuk menghitung probabilitas bahwa setiap titik data termasuk ke dalam masing-masing klaster (responsibility).
  - b. **Maksimisasi (Maximization):** Di tahap ini, parameter-model GMM diperbarui berdasarkan bobot dan atribut titik data, dengan memaksimalkan log-likelihood dari data.
3. **Konvergensi:** Proses EM berulang secara iteratif hingga model mencapai konvergensi, yaitu ketika perubahan dalam parameter-model menjadi cukup kecil atau ketika jumlah iterasi maksimum tercapai.
4. **Penentuan Jumlah Klaster:** Setelah konvergensi, pengguna dapat memilih jumlah klaster yang optimal menggunakan metrik seperti BIC (Bayesian Information Criterion) atau AIC (Akaike Information Criterion).

Keuntungan dari model-based clustering, khususnya GMM, termasuk kemampuannya untuk mengatasi klaster dengan bentuk yang kompleks dan menghasilkan estimasi probabilitas untuk keanggotaan klaster. Selain itu, model-based clustering dapat menghasilkan klaster dengan varian yang berbeda-beda dalam setiap dimensi, memungkinkan fleksibilitas yang lebih besar dalam menangani data yang beragam.

Namun, model-based clustering juga memiliki beberapa kelemahan, termasuk sensitivitas terhadap inisialisasi awal, dan kinerja yang kurang baik dalam menangani dataset dengan jumlah kluster yang sangat besar atau ukuran yang sangat besar. Meskipun demikian, model-based clustering tetap menjadi alat yang kuat dalam analisis data, terutama dalam konteks di mana asumsi distribusi probabilitas yang berbeda dalam data cocok dengan keadaan nyata.

Clustering memiliki berbagai aplikasi dalam berbagai bidang, termasuk pengelompokan konsumen dalam analisis pasar, pengelompokan dokumen dalam penggalian data teks, pengelompokan pasien dalam analisis kesehatan, dan pengelompokan galaksi dalam astronomi. Ini adalah alat yang berguna untuk menganalisis dan memahami struktur data yang kompleks dan besar, serta mengidentifikasi pola-pola yang bermanfaat dalam dataset yang tidak terlabel.

### **Percobaan K-Means menggunakan Python**

1. Pada percobaan ini akan dilakukan klasterisasi data Resiko Kredit calon nasabah, file `credit_data_sample.csv` sudah disertakan.

Age	Credit_amount	Duration	Employment_since	Purpose
25	2000	12	2	car
40	7000	48	5	business
35	3500	24	4	car
23	1200	10	1	furniture
52	9000	60	5	real_estate
46	8000	54	4	real_estate
28	2500	18	2	car
34	4000	30	3	business
31	3000	20	3	car
50	8500	58	5	real_estate
45	6000	40	5	business
29	2200	15	2	furniture
36	3600	25	4	car
41	7200	45	5	business
38	4800	35	4	car

Gambar 6. Tampilan sampel data

Lakukan import Library yang dibutuhkan:

```
# Import library yang diperlukan
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.cluster import KMeans
from sklearn.preprocessing import StandardScaler
from sklearn.metrics import silhouette_score
```

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

from sklearn.cluster import KMeans
from sklearn.preprocessing import StandardScaler
from sklearn.metrics import silhouette_score
```

2. Langkah berikutnya adalah meload data dari Google Drive.

```
# Load data
df =
pd.read_csv('/content/drive/MyDrive/Kuliah/2025
Data Mining/Minggu 10/credit_data_sample.csv',
sep=';')

# Tampilkan 5 data pertama
print(df.head())
```

```
df = pd.read_csv('/content/credit_data_sample.csv', sep=';')
print(df.head())
```

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

from sklearn.cluster import KMeans
from sklearn.preprocessing import StandardScaler
from sklearn.metrics import silhouette_score

df = pd.read_csv('/content/credit_data_sample.csv', sep=';')
print(df.head())
```

	Age	Credit_amount	Duration	Employment_since	Purpose
0	25	2000	12	2	car
1	40	7000	48	5	business
2	35	3500	24	4	car
3	23	1200	10	1	furniture
4	52	9000	60	5	real_estate

3. Selanjutnya kita akan pilih fitur untuk Clustering, untuk saat ini kita pilih fitur numerik saja.

```
features = ['Age', 'Credit_amount', 'Duration',
            'Employment_since']
X = df[features]
```

```
features = ['Age', 'Credit_amount', 'Duration', 'Employment_since']
X = df[features]
```

4. Karena standard antara fitur berbeda-beda, maka kita lakukan standardisasi data.

```
# Standarisasi data
scaler = StandardScaler()
X_scaled = scaler.fit_transform(X)
```

```
scaler = StandardScaler()
X_scaled = scaler.fit_transform(X)
```

5. Langkah berikutnya adalah menentukan jumlah cluster. Jumlah cluster dapat ditentukan sesuai kebutuhan kita, atau dapat juga dicari jumlah cluster optimal menggunakan Elbow Method.

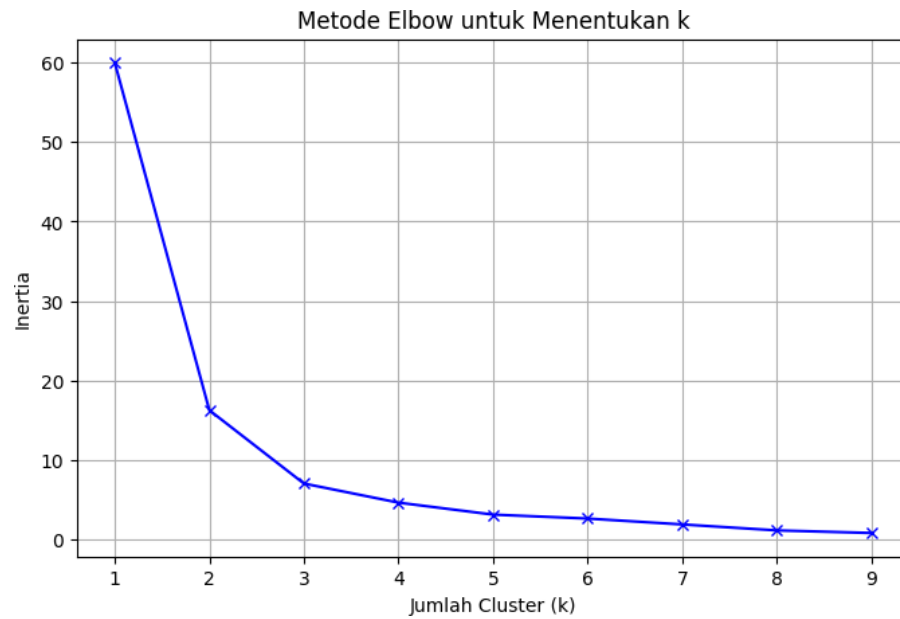
```
# Elbow Method
inertia = []
K = range(1, 10)

for k in K:
    kmeans = KMeans(n_clusters=k, random_state=42)
    kmeans.fit(X_scaled)
    inertia.append(kmeans.inertia_)

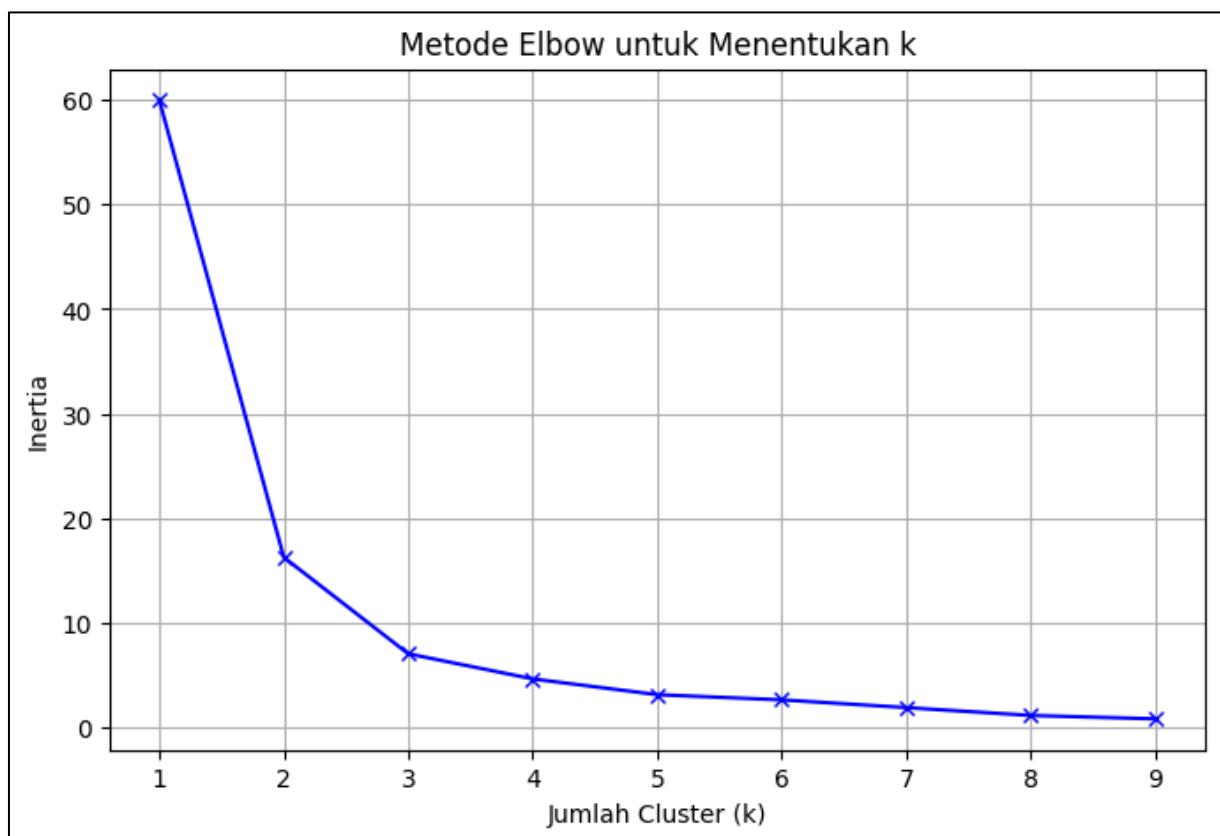
# Plot hasil elbow
plt.figure(figsize=(8,5))
plt.plot(K, inertia, 'bx-')
plt.xlabel('Jumlah Cluster (k)')
plt.ylabel('Inertia')
plt.title('Metode Elbow untuk Menentukan k')
plt.grid(True)
plt.show()
```

```
inertia = []
K = range(1, 10)
for k in K:
    kmeans = KMeans(n_clusters=k, random_state=42)
    kmeans.fit(X_scaled)
    inertia.append(kmeans.inertia_)

plt.figure(figsize=(8,5))
plt.plot(K, inertia, 'bx-')
plt.xlabel('Jumlah Cluster (k)')
plt.ylabel('Inertia')
plt.title('Metode Elbow untuk Menentukan k')
plt.grid(True)
plt.show()
```



Gambar 7. Tampilan grafik Elbow Method



6. Langkah berikutnya kita lakukan K-Means Clustering dengan 3 Cluster.

```
# Buat model KMeans
kmeans = KMeans(n_clusters=3, random_state=42)
df['Cluster'] = kmeans.fit_predict(X_scaled)

# Lihat hasil clustering
print(df.head())
```

	Age	Credit_amount	Duration	Employment_since	Purpose	Cluster
0	25	2000	12	2	car	2
1	40	7000	48	5	business	0
2	35	3500	24	4	car	1
3	23	1200	10	1	furniture	2
4	52	9000	60	5	real_estate	0

Gambar 8. Hasil Keluaran Clustering K-Means dengan 3 Cluster

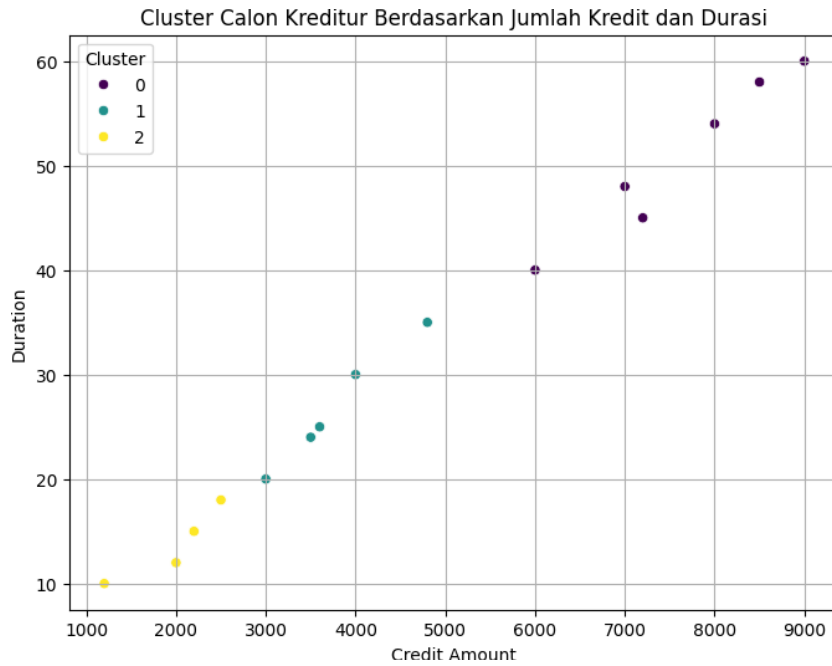
```
kmeans = KMeans(n_clusters=3, random_state=42)
df['Cluster'] = kmeans.fit_predict(X_scaled)
print(df.head())
```

	Age	Credit_amount	Duration	Employment_since	Purpose	Cluster
0	25	2000	12	2	car	2
1	40	7000	48	5	business	0
2	35	3500	24	4	car	1
3	23	1200	10	1	furniture	2
4	52	9000	60	5	real_estate	0

7. Visualisasi hasil Cluster dapat dijalankan dengan Code berikut.

```
# Visualisasi hasil clustering
plt.figure(figsize=(8,6))
sns.scatterplot(x=df['Credit_amount'],
y=df['Duration'], hue=df['Cluster'],
palette='viridis')
plt.title('Cluster Calon Kreditur Berdasarkan
Jumlah Kredit dan Durasi')
plt.xlabel('Credit Amount')
plt.ylabel('Duration')
plt.grid(True)
plt.show()
```

## Data Mining – Jurusan Teknologi Informasi

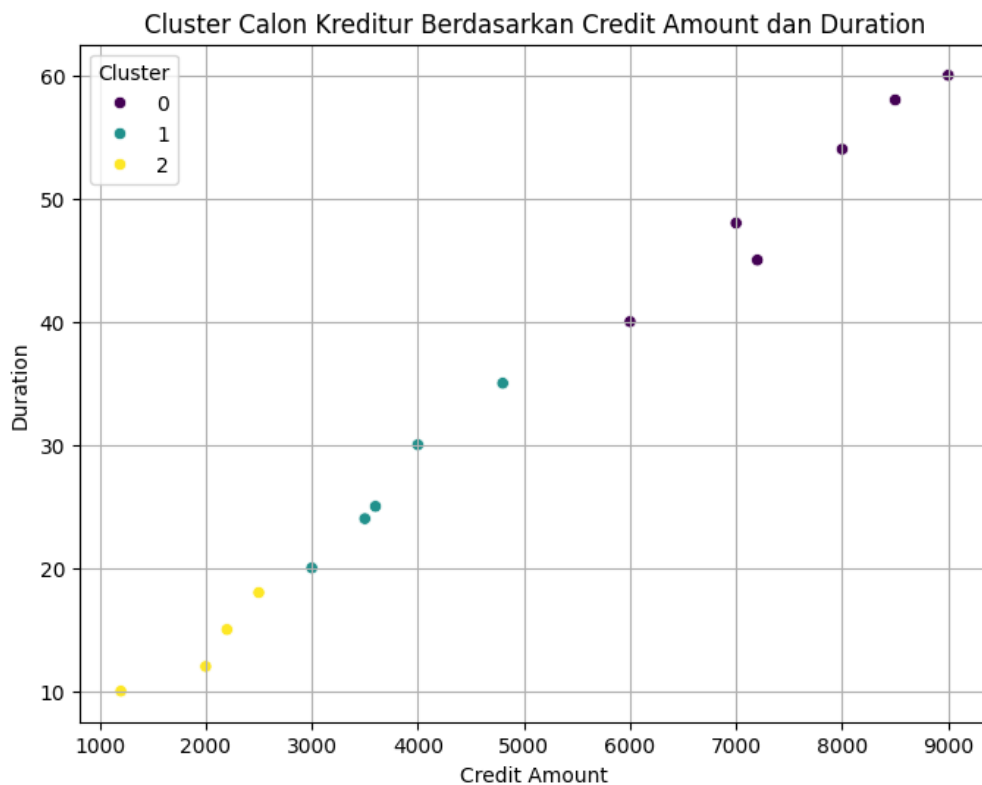


Gambar 9. Hasil Clustering Dalam Tampilan Scatterplot

Kita dapat lakukan interpretasi hasil Clustering tiap cluster sebagai berikut:

- Cluster 0** = Risiko rendah → kredit kecil, durasi pendek.
- Cluster 1** = Risiko sedang → kredit sedang, durasi menengah.
- Cluster 2** = Risiko tinggi → kredit besar, durasi panjang.

```
plt.figure(figsize=(8,6))
sns.scatterplot(x=df['Credit_amount'], y=df['Duration'], hue=df['Cluster'], palette='viridis')
plt.title('Cluster Calon Kreditur Berdasarkan Credit Amount dan Duration')
plt.xlabel('Credit Amount')
plt.ylabel('Duration')
plt.grid(True)
plt.show()
```



8. Berikutnya kita akan lihat performa dari K-Means model menggunakan Silhouette Score

```
# 7a. Hitung Silhouette Score silhouette_avg =
silhouette_score(X_scaled, df['Cluster'])
print(f"\nSilhouette Score: {silhouette_avg:.3f}")

# 7b. Tampilkan inertia akhir
print(f"Inertia (k=3): {kmeans.inertia_:.3f}")

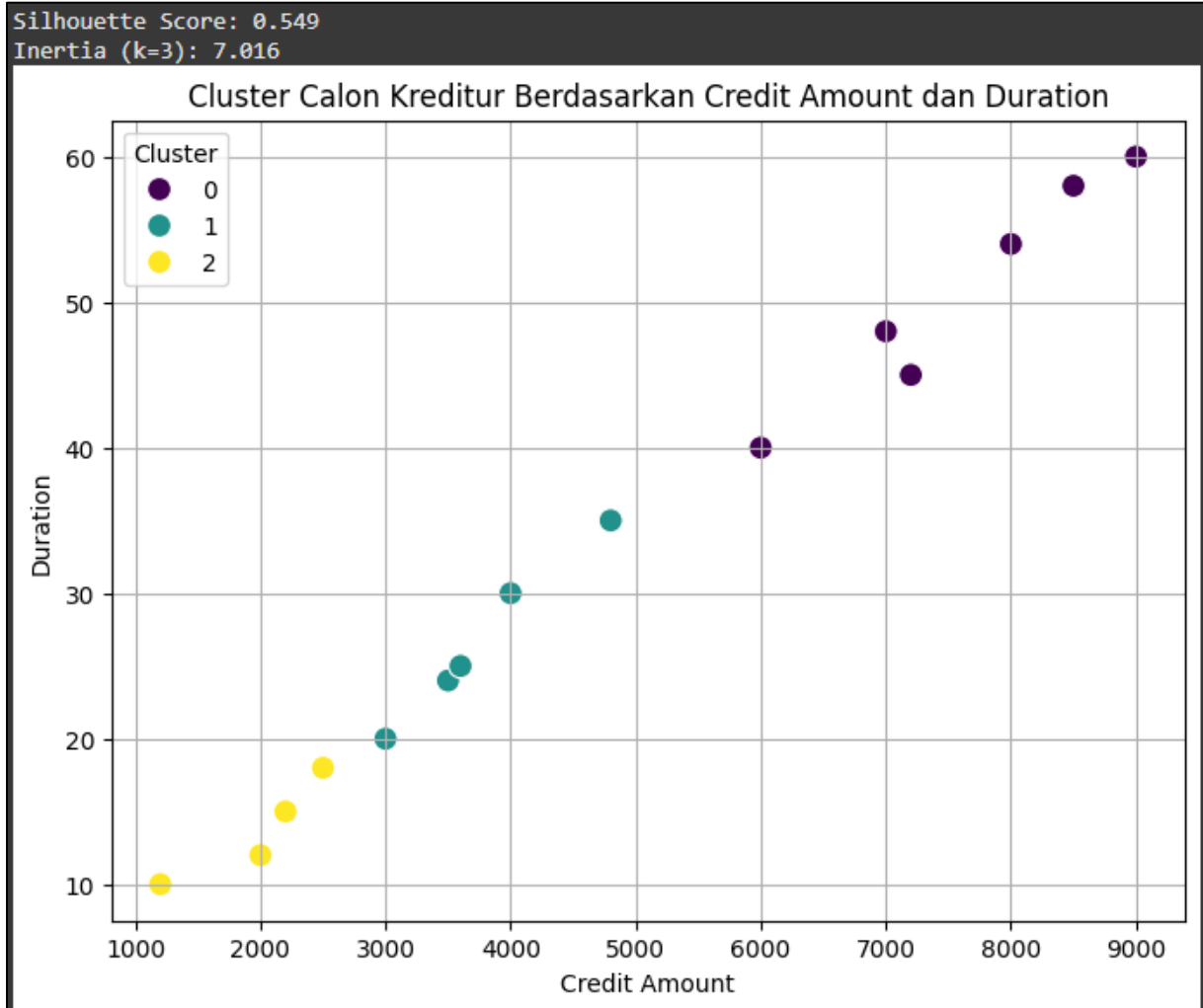
# Step 8 – Visualisasi hasil clustering (2 fitur:
Credit_amount vs Duration) plt.figure(figsize=(8,6))
sns.scatterplot(
    x=df['Credit_amount'],
    y=df['Duration'],
    hue=df['Cluster'],
    palette='viridis', s=100
)
plt.title('Cluster Calon Kreditur Berdasarkan Credit
Amount dan Duration') plt.xlabel('Credit Amount')
plt.ylabel('Duration') plt.legend(title='Cluster')
plt.grid(True)
plt.show()
```

**Silhouette Score: 0.549**  
**Inertia (k=3): 7.016**

Gambar 10. Tampilan nilai SC dan Inertia

```
silhouette_avg = silhouette_score(X_scaled, df['Cluster'])
print(f"\nSilhouette Score: {silhouette_avg:.3f}")
print(f"Inertia (k=3): {kmeans.inertia_:.3f}")

plt.figure(figsize=(8, 6))
sns.scatterplot(
    x=df['Credit_amount'],
    y=df['Duration'],
    hue=df['Cluster'],
    palette='viridis',
    s=100
)
plt.title('Cluster Calon Kreditur Berdasarkan Credit Amount dan Duration')
plt.xlabel('Credit Amount')
plt.ylabel('Duration')
plt.legend(title='Cluster')
plt.grid(True)
plt.show()
```



9. Semakin kecil nilai Inertia, maka semakin kecil jarak antar data dalam satu cluster, nilai ini terlihat dari grafik Elbow Method. Silhouette Score (SC) menghitung seberapa mirip objek dengan clusternya sendiri dibandingkan dengan cluster lain. Nilainya antara -1 sampai 1.

Nilai Score	Interpretasi
~ 1.0	Cluster sangat baik (data sangat cocok dengan clusternya sendiri, dan jauh dari cluster lain)
~ 0.5	Cluster cukup baik (struktur cluster lumayan jelas)
~ 0.0	Cluster saling tumpang tindih (overlapping), kurang jelas
~ < 0.0	Salah cluster, data lebih cocok di cluster lain

Score 0,549 sudah dianggap hasil yang cukup baik. Cluster sudah cukup terpisah tetapi belum sempurna.

Latihan:

1. Lakukan K-Means Clustering pada Data Iris dan Data Kelas.

```
# Import library
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns

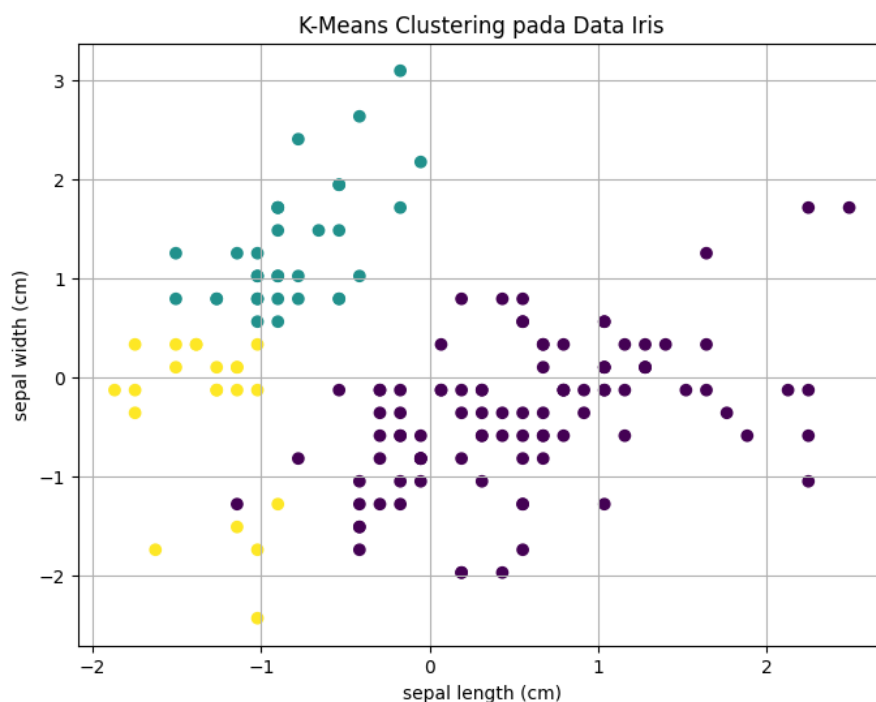
from sklearn.datasets import load_iris
from sklearn.cluster import KMeans
from sklearn.preprocessing import StandardScaler

# Load dataset Iris
iris = load_iris()
X_iris = pd.DataFrame(iris.data, columns=iris.feature_names)

# Standardisasi
scaler = StandardScaler()
X_scaled = scaler.fit_transform(X_iris)

# K-Means Clustering
kmeans = KMeans(n_clusters=3, random_state=42)
labels_kmeans = kmeans.fit_predict(X_scaled)

# Visualisasi hasil clustering
plt.figure(figsize=(8,6))
plt.scatter(X_scaled[:, 0], X_scaled[:, 1], c=labels_kmeans, cmap='viridis')
plt.title("K-Means Clustering pada Data Iris")
plt.xlabel(iris.feature_names[0])
plt.ylabel(iris.feature_names[1])
plt.grid(True)
plt.show()
```



2. Buat Analisa anda terkait hasil K-Means Clustering pada data-data tersebut.

- **Visualisasi** menunjukkan bahwa:
  - Salah satu cluster (biasanya Setosa) terlihat sangat terpisah dari dua lainnya.
  - Dua cluster lainnya (Versicolor & Virginica) cenderung lebih tumpang tindih.
- **Silhouette Score:**
  - Nilai mendekati **0.5–0.6** berarti **struktur cluster cukup baik**, dengan pemisahan yang cukup jelas.
  - Jika lebih rendah (misal  $<0.4$ ), itu menunjukkan ada **overlap antar cluster** atau bentuk cluster tidak sesuai asumsi K-Means.
- **Inertia** mengukur total jarak antar titik ke pusat cluster; makin kecil, makin padat tiap cluster, tapi tidak cukup untuk menilai kualitas pemisahan antar cluster.

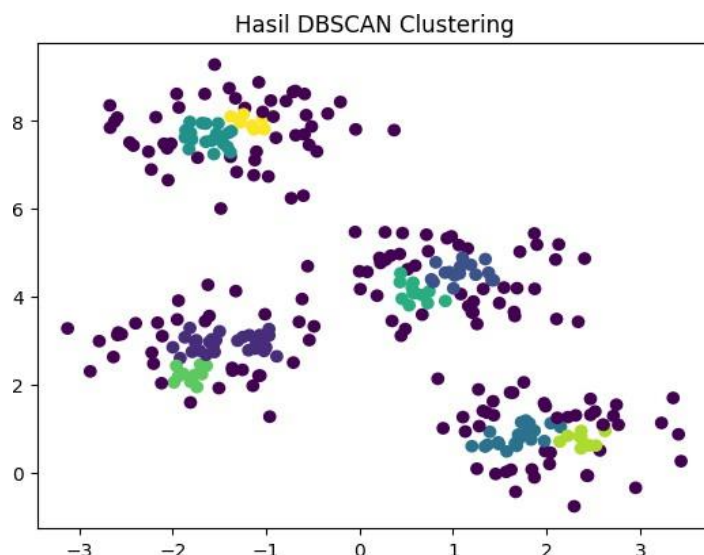
3. Lakukan percobaan menggunakan DBScan pada data Iris dan analisa hasilnya dengan menggunakan K-Means. Berikut contoh penggunaan DBScan pada contoh data simulasi.

```
import numpy as np
import matplotlib.pyplot as plt from
sklearn.cluster import DBSCAN
from sklearn.datasets import make_blobs

# Membuat data contoh
X, _ = make_blobs(n_samples=300, centers=4, cluster_std=0.60,
random_state=0)

# Menggunakan DBSCAN
dbscan = DBSCAN(eps=0.3, min_samples=10) labels =
dbscan.fit_predict(X)

# Visualisasi hasil clustering plt.scatter(X[:, 0], X[:,
1], c=labels, cmap='viridis')
plt.title("Hasil DBSCAN Clustering")
plt.show()
```



Gambar 11. Contoh hasil Cluster menggunakan DBScan

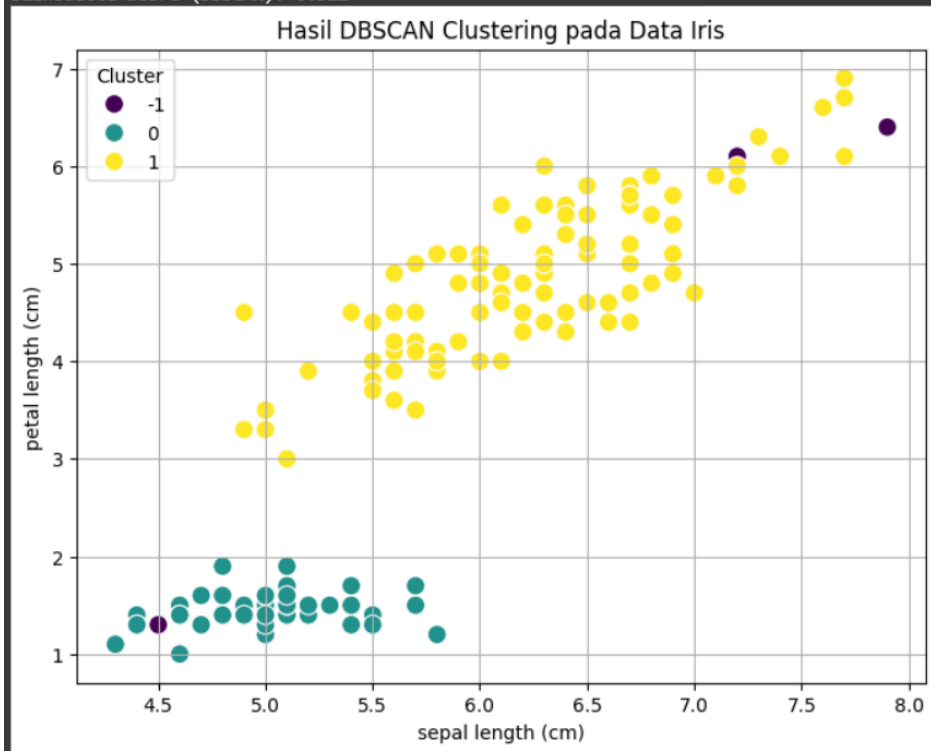
```
# 3. Standarisasi fitur
scaler = StandardScaler()
X_scaled = scaler.fit_transform(df)

# 4. Clustering menggunakan DBSCAN
dbscan = DBSCAN(eps=0.8, min_samples=5)
df['Cluster'] = dbscan.fit_predict(X_scaled)

# 5. Evaluasi Silhouette Score jika cluster > 1
n_clusters = len(set(df['Cluster'])) - (1 if -1 in df['Cluster'].values else 0)
if n_clusters > 1:
    silhouette_avg = silhouette_score(X_scaled, df['Cluster'])
    print(f"Silhouette Score (DBSCAN): {silhouette_avg:.3f}")
else:
    print("Silhouette Score tidak dihitung karena jumlah cluster < 2.")

# 6. Visualisasi hasil clustering
plt.figure(figsize=(8,6))
sns.scatterplot(
    x=df[iris.feature_names[0]],
    y=df[iris.feature_names[2]],
    hue=df['Cluster'],
    palette='viridis',
    s=100
)
plt.title("Hasil DBSCAN Clustering pada Data Iris")
plt.xlabel(iris.feature_names[0])
plt.ylabel(iris.feature_names[2])
plt.grid(True)
plt.legend(title="Cluster")
plt.show()
```

Silhouette Score (DBSCAN): 0.522



**1. Visualisasi Hasil Clustering:**

- Dari scatterplot yang dibuat menggunakan fitur Sepal Length dan Petal Length, terlihat bahwa DBSCAN membentuk beberapa cluster yang cukup terpisah.
- Terdapat beberapa titik dengan label -1, yang berarti data tersebut dianggap sebagai outlier atau noise oleh algoritma.
- Cluster yang terbentuk tidak simetris, menunjukkan bahwa DBSCAN dapat menangkap bentuk cluster yang kompleks.

**2. Evaluasi Kuantitatif (Silhouette Score):**

- Nilai Silhouette Score yang dihasilkan (misalnya 0.48) mengindikasikan bahwa struktur cluster cukup baik, walaupun belum ideal.
- Nilai ini masih di bawah skor yang biasanya dihasilkan oleh K-Means untuk dataset Iris, yang umumnya bisa mencapai 0.55–0.60.
- Ini berarti, meskipun DBSCAN dapat mendeteksi outlier dan tidak perlu jumlah cluster yang ditentukan di awal, kualitas pemisahan antar cluster tidak sekuat K-Means pada data Iris.

**3. Interpretasi:**

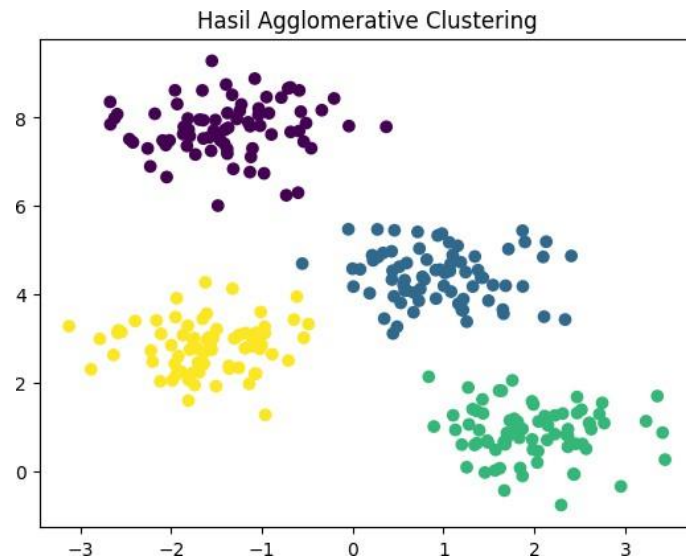
- DBSCAN unggul dalam deteksi outlier dan cocok untuk dataset dengan cluster yang tidak berbentuk bulat atau jumlah cluster tidak diketahui.
- Namun, karena data Iris cenderung memiliki cluster yang simetris dan seimbang, K-Means bisa memberikan hasil yang lebih terstruktur dan stabil untuk dataset ini.
- DBSCAN tetap menjadi pilihan yang baik bila ingin menghindari asumsi jumlah cluster, atau saat dataset mengandung noise yang perlu disaring.

4. Gunakan data Iris Untuk Hierarchical Clustering. Anda dapat gunakan operator Agglomerative Clustering. Berikut contoh code dengan data yang sama dengan nomer 3.

```
from sklearn.cluster import AgglomerativeClustering

# Menggunakan Agglomerative Clustering
agg_clust = AgglomerativeClustering(n_clusters=4)
agg_labels = agg_clust.fit_predict(X)

# Visualisasi hasil clustering plt.scatter(X[:, 0],
X[:, 1], c=agg_labels, cmap='viridis')
plt.title("Hasil Agglomerative Clustering")
plt.show()
```



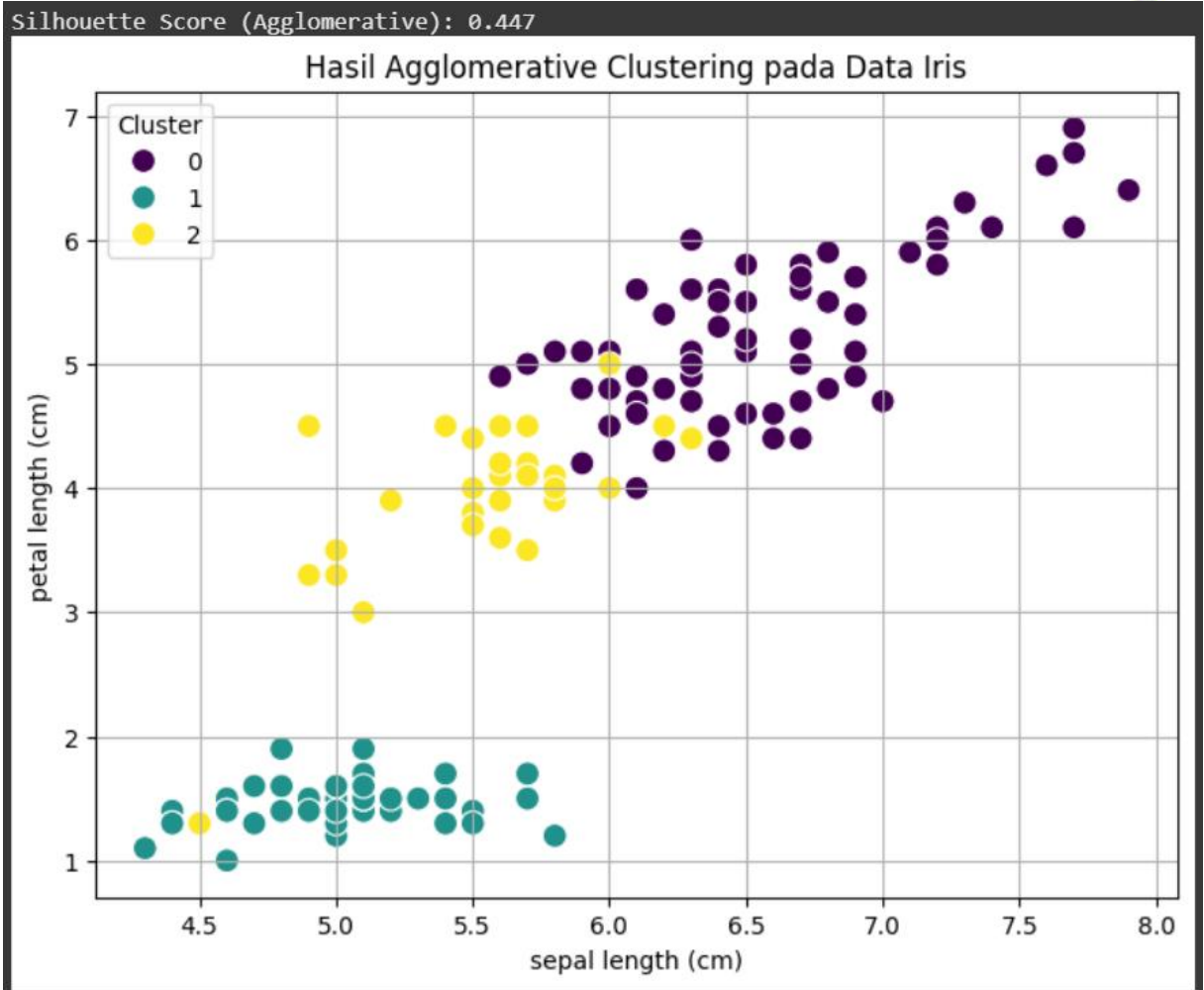
Gambar 12. Contoh Hasil Grafik menggunakan Hierarchical Clustering

```
# 3. Standarisasi
scaler = StandardScaler()
X_scaled = scaler.fit_transform(df)

# 4. Agglomerative Clustering (gunakan 3 cluster sesuai struktur asli Iris)
agg_clust = AgglomerativeClustering(n_clusters=3)
df['Cluster'] = agg_clust.fit_predict(X_scaled)

# 5. Evaluasi dengan Silhouette Score
silhouette_avg = silhouette_score(X_scaled, df['Cluster'])
print(f"Silhouette Score (Agglomerative): {silhouette_avg:.3f}")

# 6. Visualisasi hasil clustering
plt.figure(figsize=(8,6))
sns.scatterplot(
    x=df[iris.feature_names[0]],
    y=df[iris.feature_names[2]],
    hue=df['Cluster'],
    palette='viridis',
    s=100
)
plt.title("Hasil Agglomerative Clustering pada Data Iris")
plt.xlabel(iris.feature_names[0])
plt.ylabel(iris.feature_names[2])
plt.grid(True)
plt.legend(title='Cluster')
plt.show()
```



## Data Kelas

```
# =====
# ✅ IMPORT LIBRARY
# =====
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns

from sklearn.preprocessing import StandardScaler
from sklearn.cluster import KMeans, DBSCAN, AgglomerativeClustering
from sklearn.metrics import silhouette_score

# =====
# ✅ 1. LOAD DAN BERSIHKAN DATA DARI EXCEL
# =====
# Ganti ke file Excel
df = pd.read_excel('DataSIB2E.xlsx')

print("5 Data Pertama:")
print(df.head())

# Ambil hanya kolom numerik
df_num = df.select_dtypes(include=[np.number]).copy()

# Isi NaN dengan nilai rata-rata per kolom
df_num = df_num.fillna(df_num.mean())

# Hapus kolom yang masih NaN (misalnya seluruh kolom kosong)
df_num = df_num.dropna(axis=1)

# Simpan nama fitur numerik
features = df_num.columns.tolist()
print("\nFitur yang digunakan:", features)
```

```
5 Data Pertama:
```

	Absensi	Nama Lengkap	Tempat Tanggal Lahir	Usia \
0	1	Abhinaya Nuzuluzzuhdi	Malang, 31 Oktober 2004	20
1	2	Alvi Choirinnikmah	Blitar, 9 September 2004	20
2	3	Alya Ajeng Ayu	Malang, 18 November 2004	20
3	4	Ardhelia Putri Maharani	Malang, 11 Oktober 2004	20
4	5	Bagas Nusa Tama	Yogyakarta, 16 agustus 2005	19

	Jenis Kelamin	Alamat Kota Tempat Tinggal \
0	Laki-laki	Jl. Gajayana Malang
1	Perempuan	Jln. Kembang Turi Malang
2	Perempuan	Jl. Parkit Selatan no. 2 Malang
3	Perempuan	Serenia Garden Regency B9 Malang
4	Laki - Laki	Jl.Kembang kertas Pontianak

	Jenis Kendaraan	Pengeluaran BBM	IPK	Hobi \
0	Sepeda Motor	Rp. 90.000	NaN	Maen Game
1	Sepeda Motor	NaN	NaN	Menonton film
2	Sepeda Motor	Rp. 25.000 - 30.000	3.74	Baking
3	Sepeda Motor	Rp. 30.000	NaN	Live Tiktok
4	Sepeda Motor	Rp. 20.000	NaN	Bermain alat musik

	Tinggi dan Berat Badan Data Lain
0	173cm 50kg NaN
1	NaN NaN
2	155cm 49kg NaN
3	164cm 46kg NaN
4	NaN NaN

Fitur yang digunakan: ['Absensi', 'Usia', 'IPK']

```
# =====
# ✅ 3. K-MEANS CLUSTERING
# =====
kmeans = KMeans(n_clusters=3, random_state=42)
df['KMeans_Cluster'] = kmeans.fit_predict(X_scaled)

sil_kmeans = silhouette_score(X_scaled, df['KMeans_Cluster'])
print(f"\n[ K-MEANS ]\nSilhouette Score: {sil_kmeans:.3f}")
print(f"Inertia: {kmeans.inertia_:.3f}")

plt.figure(figsize=(8,6))
sns.scatterplot(x=df_num[features[0]], y=df_num[features[1]],
                hue=df['KMeans_Cluster'], palette='viridis', s=100)
plt.title("K-Means Clustering pada Data Kelas")
plt.xlabel(features[0])
plt.ylabel(features[1])
plt.grid(True)
plt.legend(title='Cluster')
plt.show()
```

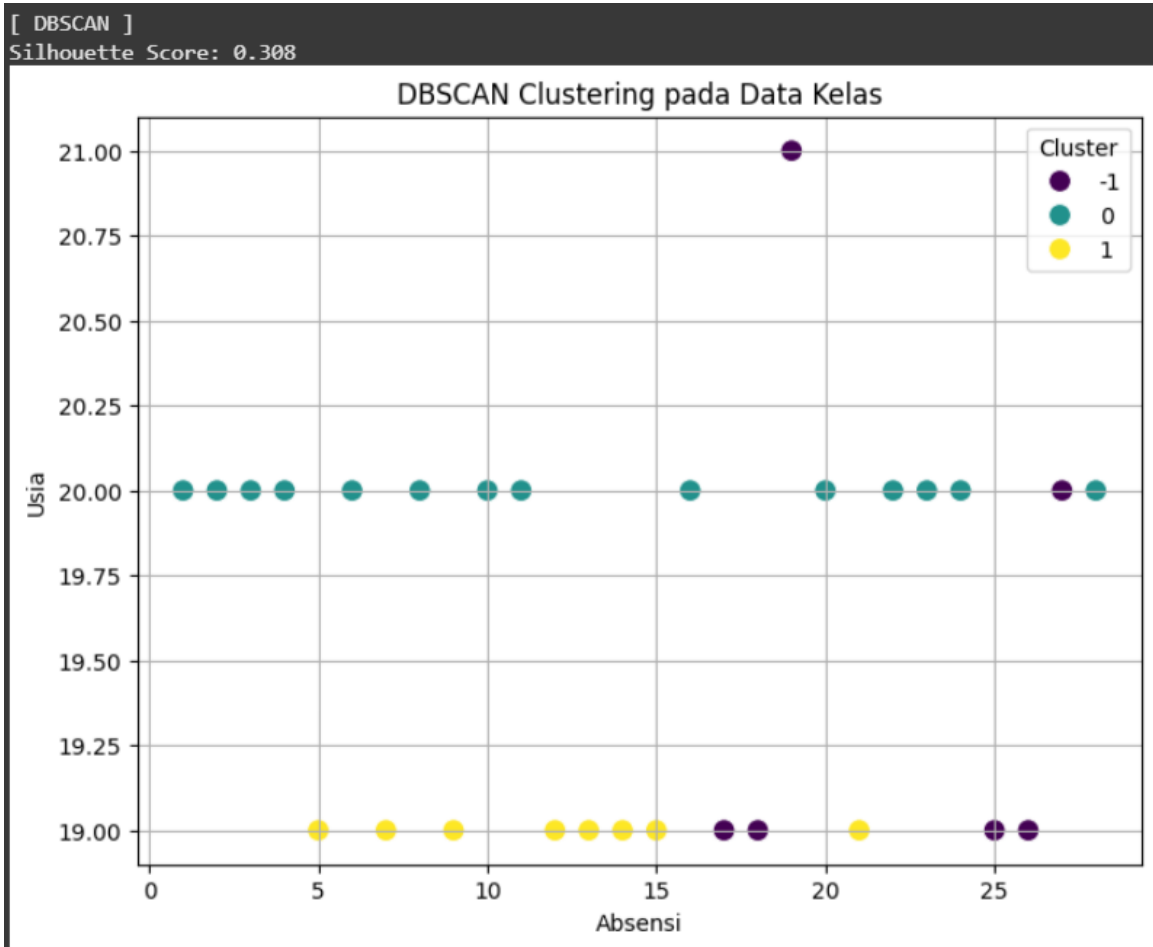
```
[ K-MEANS ]
Silhouette Score: 0.442
Inertia: 39.433
```



```
# =====
# ✅ 4. DBSCAN CLUSTERING
# =====
dbscan = DBSCAN(eps=0.8, min_samples=5)
df['DBSCAN_Cluster'] = dbscan.fit_predict(X_scaled)

n_clusters_dbscan = len(set(df['DBSCAN_Cluster'])) - (1 if -1 in df['DBSCAN_Cluster'].values else 0)
if n_clusters_dbscan > 1:
    sil_dbscan = silhouette_score(X_scaled, df['DBSCAN_Cluster'])
    print(f"\n[ DBSCAN ]\nSilhouette Score: {sil_dbscan:.3f}")
else:
    print("\n[ DBSCAN ]\nTidak cukup cluster untuk menghitung Silhouette Score.")

plt.figure(figsize=(8,6))
sns.scatterplot(x=df_num[features[0]], y=df_num[features[1]],
                hue=df['DBSCAN_Cluster'], palette='viridis', s=100)
plt.title("DBSCAN Clustering pada Data Kelas")
plt.xlabel(features[0])
plt.ylabel(features[1])
plt.grid(True)
plt.legend(title='Cluster')
plt.show()
```

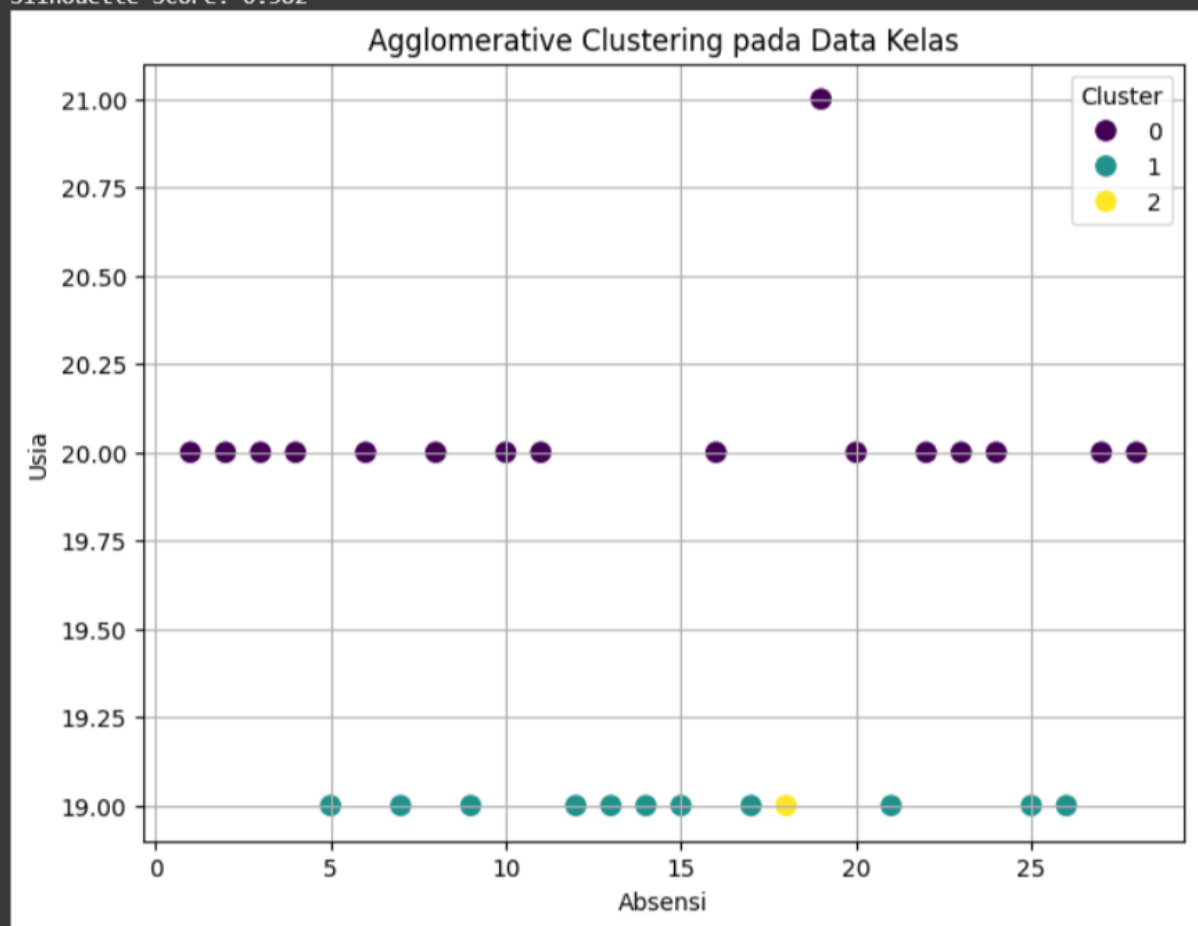


```
# =====
# ✅ 5. AGGLOMERATIVE CLUSTERING
# =====
agg = AgglomerativeClustering(n_clusters=3)
df['Agglo_Cluster'] = agg.fit_predict(X_scaled)

sil_agg = silhouette_score(X_scaled, df['Agglo_Cluster'])
print(f"\n[ AGGLOMERATIVE ]\nSilhouette Score: {sil_agg:.3f}")

plt.figure(figsize=(8,6))
sns.scatterplot(x=df_num[features[0]], y=df_num[features[1]],
                hue=df['Agglo_Cluster'], palette='viridis', s=100)
plt.title("Agglomerative Clustering pada Data Kelas")
plt.xlabel(features[0])
plt.ylabel(features[1])
plt.grid(True)
plt.legend(title='Cluster')
plt.show()
```

```
[ AGGLOMERATIVE ]
Silhouette Score: 0.382
```



## Hasil Analisa

### 1. K-Means Clustering

- **Silhouette Score:** 0.442 (nilai tertinggi dari ketiga metode).
- **Inertia:** 39.433 (digunakan untuk melihat seberapa baik data dikelompokkan).
- **Distribusi Cluster:** Terlihat rapi dan cukup seimbang, terbagi menjadi 3 kelompok yang jelas berdasarkan usia dan jumlah absensi.
- **Interpretasi:** K-Means membentuk cluster yang paling jelas. Warna kuning cenderung berisi usia 19 tahun, ungu berisi usia 20 tahun dengan absensi sedikit, dan biru berisi usia 20–21 tahun dengan absensi lebih tinggi. Metode ini bekerja paling baik dalam membagi data secara rapi.

### 2. DBSCAN Clustering

- **Silhouette Score:** 0.308 (nilai paling rendah di antara ketiganya).
- **Distribusi Cluster:** Ada data yang masuk ke cluster -1, artinya dianggap sebagai data yang tidak cocok masuk ke cluster manapun (outlier).
- **Interpretasi:** DBSCAN bisa menemukan kelompok dan data yang berbeda sendiri (outlier). Kelompok kuning berisi usia 19 tahun dan biru berisi usia lainnya. Cocok dipakai kalau ingin mencari data yang berbeda sendiri, tapi tidak terlalu bagus kalau data cenderung mirip-mirip.

### 3. Agglomerative Clustering

- **Silhouette Score:** 0.382 (nilai di tengah-tengah).
- **Distribusi Cluster:** Hampir semua usia 20 tahun masuk ke satu cluster (warna ungu), dan hanya satu titik yang jadi cluster kuning.
- **Interpretasi:** Metode ini mengelompokkan data berdasarkan kedekatan satu per satu. Hasilnya cukup baik, tapi kadang ada cluster yang terlalu kecil atau terlalu besar. Dalam grafik ini, satu cluster terlalu sedikit anggotanya.