MODUL PRAKTIKUM

MATA KULIAH DATA MINING

PERTEMUAN 9

SEMESTER GENAP

TAHUN AJARAN 2024/2025

Disusun oleh:

Dwi Welly Sukma Nirad S.Kom, M.T

Aina Hubby Aziira M.Eng

Ghina Anfasha Nurhadi

Daffa Agustian Saadi

DEPARTEMEN SISTEM INFORMASI

FAKULTAS TEKNOLOGI INFORMASI

UNIVERSITAS ANDALAS

TAHUN 2025

IDENTITAS PRAKTIKUM

IDENTITAS MATA KULIAH

Kode mata kuliah	JSI62122
Nama mata kuliah	Data Mining
CPMK yang dibebankan pada praktikum	CPMK-3, CPMK-4 Mahasiswa mampu memahami teknik klasterisasi dalam data mining (CP-2)
Materi Praktikum Pertemuan 8	Konsep Dasar DBSCAN
	Parameter Epsilon dan MinPoints
	Implementasi DBSCAN
	Analisis hasil clustering

IDENTITAS DOSEN DAN ASISTEN MAHASISWA

Nama Dosen Pengampu	1. Dwi Welly Sukma Nirad S.Kom, M.T 2. Aina Hubby Aziira M.Eng
Nama Asisten Mahasiswa (Kelas A)	1. 2211523034 - Muhammad Fariz 2. 2211521012 - Rizka Kurnia Illahi 3. 2211521010 - Dhiya Gustita Aqila 4. 2211522013 - Benni Putra Chaniago 5. 2211521017 - Ghina Anfasha Nurhadi 6. 2211523022 - Daffa Agustian Saadi 7. 2211521007 - Annisa Nurul Hakim 8. 2211522021 - Rifqi Asverian Putra 9. 2211521009 - Miftahul Khaira

	10. 2211521015- Nurul Afani 11. 2211523028 - M.Faiz Al-Dzikro
Nama Asisten Mahasiswa (Kelas B)	1. 2211523034 - Muhammad Fariz 2. 2211521012 - Rizka Kurnia Illahi 3. 2211521010 - Dhiya Gustita Aqila 4. 2211522013 - Benni Putra Chaniago 5. 2211521017 - Ghina Anfasha Nurhadi 6. 2211523022 - Daffa Agustian Saadi 7. 2211521007 - Annisa Nurul Hakim 8. 2211522021 - Rifqi Asverian Putra 9. 2211521009 - Miftahul Khaira 10. 2211521015- Nurul Afani 11. 2211523028 - M.Faiz Al-Dzikro

DAFTAR ISI

DBSCAN

A. KONSEP DASAR DBSCAN

Density-Based Spatial Clustering of Applications with Noise atau DBSCAN merupakan salah satu algoritma dalam metode *clustering* yang mengelompokkan data berdasarkan kepadatan (*density-based*) dari posisi amatan data dengan prinsip mengelompokkan data yang relatif berdekatan. DBSCAN membentuk *cluster* dengan mengidentifikasi area dengan kepadatan tinggi, yaitu wilayah di mana titik-titik data berdekatan dalam jarak tertentu (*epsilon*, ε) dan memiliki minimal sejumlah titik tertentu (MinPts). Jika banyak titik yang berdekatan dalam satu area, itu dianggap sebagai *cluster*. Namun, jika hanya ada sedikit titik, area itu dianggap sebagai *noise* (*outlier*). Tiga macam titik dalam algoritma ini yaitu :

1. ***Core Points***: Titik-titik yang memiliki setidaknya sejumlah minimum titik tetangga (MinPts) dalam jarak tertentu (ε atau *epsilon*).
2. ***Border Points***: Titik-titik yang berada dalam jarak ε dari titik inti, tetapi tidak memiliki jumlah tetangga yang cukup untuk dikategorikan sebagai titik inti.
3. ***Noise Points***: Titik-titik yang bukan termasuk titik inti maupun titik batas. Titik ini terlalu jauh dari klaster mana pun sehingga tidak dapat dimasukkan ke dalam klaster apa pun.

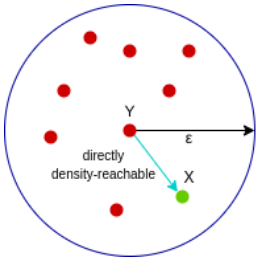
Selain ketiga macam titik di atas, dalam algoritma DBSCAN juga dikenal tiga istilah mengenai keterhubungan dan jangkauan antar titik berdasarkan kepadatan (density), yaitu :

1. ***Directly Density-Reachable*** (Densitas terjangkau langsung)

   Titik **X** disebut ***directly density-reachable*** dari titik **Y** dengan parameter *epsilon* (ε) dan *minPoints* jika:

- **X** berada dalam *neighborhood* (lingkungan) **Y**, artinya jarak antara **X** dan **Y** lebih kecil atau sama dengan *epsilon* (dist(X, Y) <= ε).
- **Y** adalah titik core, yaitu titik yang memiliki setidaknya *minPoints* titik dalam jarak *epsilon* darinya.

Dengan kata lain, **X** dapat dijangkau langsung dari **Y**. Namun, sebaliknya, **Y** belum tentu langsung dijangkau dari **X**.

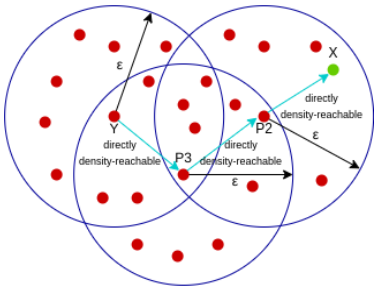


2. ***Density-Reachable*** (Densitas terjangkau)

Titik **X** disebut ***density-reachable*** dari titik **Y** dengan parameter *epsilon* (ε) dan *minPoints* jika ada rangkaian titik **p1, p2, p3, …, pn** yang memenuhi:

- **p1** = **X** dan **pn** = **Y**,
- Di mana setiap titik **pi+1** langsung *density-reachable* dari titik **pi**.

Artinya, **X** bisa dijangkau dari **Y** melalui serangkaian titik yang masing-masing dapat dijangkau langsung (*directly density-reachable*) satu sama lain. Namun, sebaliknya (Y ke X) tidak berlaku.

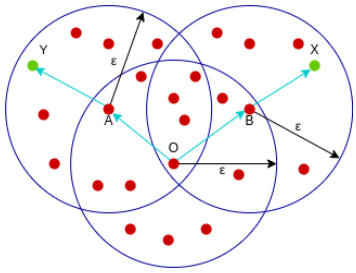


3. ***Density-Connected*** (Densitas terhubung)

   Titik X disebut *density-connected* dengan titik Y dengan parameter epsilon (ε) dan minPoints jika ada titik O yang memenuhi:

   - Baik **X** maupun **Y** adalah ***density-reachable*** dari titik **O** dengan *epsilon* dan *minPoints*.

   Dengan kata lain, **X** dan **Y** saling terhubung melalui titik **O** yang keduanya dapat dijangkau dari **O**.



Kelebihan dari metode *clustering* dengan algoritma DBSCAN, yaitu :

1. Dapat menangani klaster dengan bentuk tidak teratur
2. Tidak memerlukan jumlah klaster sebagai input awal
3. Tahan (*robust*) terhadap *noise*
4. Dapat menangani dataset berukuran besar
5. Fleksibilitas dalam parameter

Kekurangan dari metode *clustering* dengan algoritma DBSCAN, yaitu :

1. Sensitivitas terhadap pemilihan parameter
2. Sulit menangani dataset dengan kepadatan berbeda
3. Tidak efisien pada dimensi tinggi (*High-Dimensional Data*)
4. Kesulitan mengidentifikasi klaster kecil
5. Kesulitan dalam menentukan ε pada data skala besar

B. PARAMETER EPSILON DAN MINPOINTs

Dalam algoritma DBSCAN, dua parameter utama yang sangat penting dalam menentukan struktur *cluster* adalah **ε (epsilon)** dan **MinPoints / MinPts**. Kedua parameter ini berfungsi untuk mengatur kedekatan antar titik data dan kepadatan yang dibutuhkan untuk membentuk cluster yang valid. Pemilihan nilai yang tepat untuk kedua parameter ini akan mempengaruhi hasil clustering, baik dalam hal pembentukan cluster yang tepat maupun pengenalan noise.

1. ε (Epsilon)

Epsilon (ε) adalah parameter yang menentukan jarak maksimum antara dua titik agar dapat dianggap sebagai tetangga atau bagian dari satu cluster. Secara sederhana, **ε** dapat digambarkan sebagai radius yang mengelilingi sebuah titik data. Titik-titik yang berada dalam jarak **ε** dari titik tersebut akan dianggap sebagai *neighbors* atau tetangga.

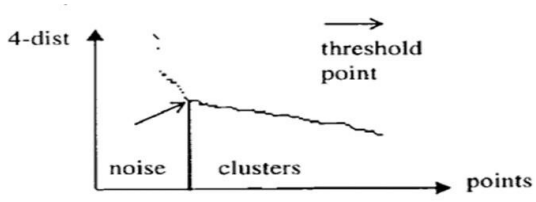
Pemilihan **ε** yang tepat sangat penting dalam DBSCAN. Jika **ε** terlalu kecil, sebagian besar titik akan terklasifikasikan sebagai noise (tidak terhubung ke cluster manapun). Sebaliknya, jika **ε** terlalu besar, maka titik-titik yang sebenarnya terpisah bisa saja tergabung dalam cluster yang sama, sehingga algoritma mungkin gagal membedakan cluster yang berbeda. Oleh karena itu, **ε** harus dipilih dengan hati-hati untuk menyeimbangkan keduanya.

Untuk menentukan nilai epsilon (ε) yang optimal dalam DBSCAN, salah satu metode yang umum digunakan adalah dengan membuat grafik k-*distance*. Grafik ini dibentuk dengan terlebih dahulu menghitung jarak dari setiap titik data ke tetangganya yang ke-k, di mana nilai k biasanya disamakan dengan MinPts yang digunakan dalam algoritma. Misalnya jika MinPts = 4, maka hitunglah jarak ke tetangga ke-4 dari setiap titik. Langkah-langkah pembuatannya adalah sebagai berikut :

a. Hitung jarak setiap titik data ke tetangganya yang ke-k.
b. Urutkan semua jarak tersebut dari yang paling kecil ke terbesar.
c. Buat grafik yang memetakan indeks titik terhadap jarak tetangga ke-k.
d. Amati grafik tersebut dan cari bagian yang memiliki sudut *elbow* atau tekukan tajam. Titik *elbow* ini biasanya menunjukkan perubahan mendalam pada distribusi

jarak, di mana sebelumnya jarak antar titik masih dalam satu *cluster*, dan setelahnya mulai menjauh dari *cluster* (potensi *noise*).

Titik pada grafik tersebut yang membentuk *elbow* dapat dianggap sebagai nilai ε yang ideal, karena merepresentasikan batas antara titik-titik yang masih saling terhubung dalam sebuah cluster dan titik-titik yang dianggap terlalu jauh atau noise.



2. MinPts (Minimum Points)

MinPts adalah parameter yang menetapkan jumlah minimum titik yang dibutuhkan dalam radius **ε** untuk membentuk sebuah *dense region* atau wilayah yang padat. Titik yang memiliki jumlah tetangga (termasuk dirinya sendiri) lebih dari atau sama dengan **MinPts** dianggap sebagai titik **core**, yaitu titik yang berada di pusat sebuah cluster. Titik core ini kemudian menjadi pusat bagi cluster yang terbentuk. Umumnya, nilai **MinPts** ditentukan dengan rumus:

$$\text{MinPts} \geq D + 1$$

di mana **D** adalah jumlah dimensi data. Sebagai contoh, untuk data dua dimensi (**D = 2**), maka **MinPts** minimal adalah 3. Nilai **MinPts** yang lebih besar akan menghasilkan cluster yang lebih padat dan lebih terisolasi.

Parameter ini membantu DBSCAN membedakan antara titik-titik yang padat dan yang terisolasi. Jika sebuah titik tidak memiliki cukup tetangga (kurang dari **MinPts**) dalam radius **ε**, maka titik tersebut akan dianggap sebagai noise atau titik yang tidak terklasifikasi dalam cluster manapun.

Dalam proses pembuatan cluster menggunakan DBSCAN sebuah data akan dikelompokkan dengan tetangganya. Sepasang amatan dikatakan bertetangga apabila jarak antara

dua amatan tersebut kurang dari sama dengan nilai epsilon. Secara sederhana cara kerja DBSCAN adalah sebagai berikut :

1. Tentukan nilai ε (*epsilon*) dan MinPts berdasarkan karakteristik data.
2. Pilih satu titik data secara acak sebagai titik awal (p).
3. Hitung jarak dari titik p ke seluruh titik lainnya menggunakan rumus jarak Euclidean:

$$d(P,C) = \sqrt{\sum_{i=1}^{n}(x_{pi} - x_{ci})^2}$$

   - *d (P, C)* : Jarak antara titik P dan C
   - x_{pi}, x_{ci} : Nilai fitur ke-i dari titik P dan C
   - n : Jumlah dimensi fitur

4. Periksa berapa banyak titik yang berada dalam jarak ε dari titik p.
   - Jika jumlahnya ≥ MinPts → titik p adalah core point, buat *cluster* baru.
   - Tambahkan semua titik yang *density-reachable* dari titik p ke dalam cluster (titik-titik yang bisa dijangkau dari p melalui serangkaian *core point*).
5. Jika titik p hanya memiliki sedikit tetangga (kurang dari MinPts) → cek apakah p merupakan *border point* atau *noise*.
6. Ulangi langkah 2–5 untuk semua titik data yang belum dikunjungi.
7. Setelah semua titik diproses:
   - Titik yang masuk *cluster* akan tergabung sebagai bagian dari *cluster*.
   - Titik yang tidak tergabung dalam *cluster* mana pun dianggap sebagai *noise*.

C. IMPLEMENTASI DBSCAN

Selanjutnya kita akan mengimplementasikan algoritma DBSCAN untuk melakukan clustering pada dataset “Mall_Costumers.csv” DBSCAN CLUSTERING dan mengidentifikasi pelanggan dengan pola kunjungan yang serupa, serta mengevaluasi hasil clustering yang dihasilkan. Langkah langkahnya sebagai berikut:

1. Import libraries yang diperlukan.

[56]:
```
import pandas as pd
from sklearn import metrics
import numpy as np
import matplotlib.pyplot as plt

df = pd.read_csv("Mall_Customers.csv")
df
```

[56]:

	CustomerID	Gender	Age	Annual Income (k$)	Spending Score (1-100)
0	1	Male	19	15	39
1	2	Male	21	15	81
2	3	Female	20	16	6
3	4	Female	23	16	77
4	5	Female	31	17	40
...	...	...	...	...	...
195	196	Female	35	120	79
196	197	Female	45	126	28
197	198	Male	32	126	74
198	199	Male	32	137	18
199	200	Male	30	137	83

200 rows × 5 columns

2. Mendeteksi nilai null pada data.

[57]:
```
df.isnull().sum()
```

[57]:
```
CustomerID              0
Gender                  0
Age                     0
Annual Income (k$)      0
Spending Score (1-100)  0
dtype: int64
```

3. Mendeteksi adanya duplikasi pada data.

```
[58]: df.duplicated().sum()
```

```
[58]: 0
```

4. Melakukan pemilihan fitur untuk clustering dalam hal ini adalah semua baris dan kolom index ke 3 dan 4.

```
[59]: x = df.iloc[:,[3,4]].values
```

5. Melihat dimensi data yang sudah dipilih sebelumnya.

```
[60]: x.shape
```

```
[60]: (200, 2)
```

6. Import library DBSCAN dan membuat objeknya dengan parameter tertentu.

```
[61]: from sklearn.cluster import DBSCAN
      db=DBSCAN(eps=10,min_samples=5,metric='euclidean')
```

7. Menjalankan algoritma DBSCAN terhadap data yang disimpan pada variabel x, mengambil hasil clusteringnya dan menampilkan hasilnya.

```
[62]: model=db.fit(x)
```

```
[63]: label=model.labels_
      label
```

```
[63]: array([ 0,  0,  1,  0,  0,  0,  1, -1,  1,  0,  1, -1,  1,  0,  1,  0,  0,
              0,  0, -1,  0,  0,  1,  0,  1,  0,  0,  0,  0,  0,  1,  0,  1, -1,
              1,  0,  0,  0,  0,  0,  0, -1,  0,  0,  0,  0,  0,  0,  0,  0,  0,
              0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,
              0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,
              0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,  0,
              0,  0,  0,  0,  2,  0,  2,  0,  2,  3,  2,  3,  2,  0,  2,  3,  2,
              3,  2,  3,  2,  3,  2,  0,  2,  3,  2,  0,  2,  3,  2,  3,  2,  3,
              2,  3,  2,  3,  2,  3,  2,  0,  2,  3,  2,  3,  2,  3,  2,  3, -1,
              3,  2,  3,  2,  3,  2,  3,  2,  3,  2,  3,  2,  3,  2, -1,  2,  3,
             -1,  3,  2,  3, -1, -1, -1, -1, -1, -1, -1, -1, -1], dtype=int64)
```

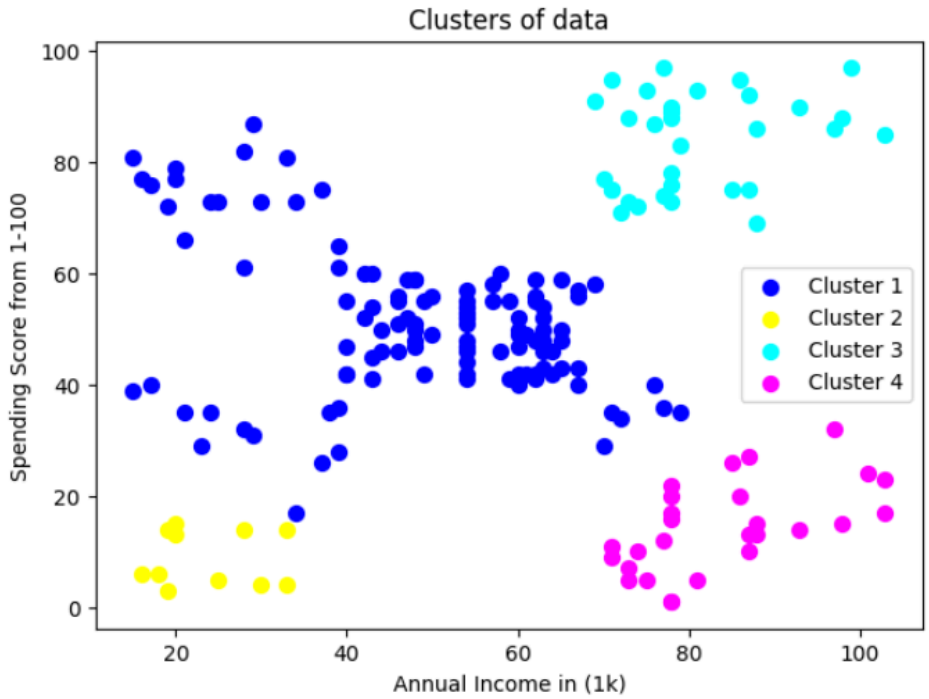
8. Melakukan perhitungan untuk mencari jumlah kluster.

[64]:
```
#identifying the points which makes up our core points
sample_cores=np.zeros_like(label,dtype=bool)

sample_cores[db.core_sample_indices_]=True

#Calculating the number of clusters

n_clusters=len(set(label))- (1 if -1 in label else 0)
print('No of clusters:',n_clusters)
```

```
No of clusters: 4
```

9. Melakukan visualisasi hasil kluster yang terbentuk.

[72]:
```
y_means = db.fit_predict(x)
plt.figure(figsize=(7,5))
plt.scatter(x[y_means == 0, 0], x[y_means == 0, 1], s = 50, c = 'blue',label='Cluster 1')
plt.scatter(x[y_means == 1, 0], x[y_means == 1, 1], s = 50, c = 'yellow',label='Cluster 2')
plt.scatter(x[y_means == 2, 0], x[y_means == 2, 1], s = 50, c = 'cyan',label='Cluster 3')
plt.scatter(x[y_means == 3, 0], x[y_means == 3, 1], s = 50, c = 'magenta',label='Cluster 4')
plt.xlabel('Annual Income in (1k)')
plt.ylabel('Spending Score from 1-100')
plt.title('Clusters of data')
plt.legend()
plt.show()
```



10. Melakukan evaluasi dari hasil clustering yang terbentuk dengan menggunakan *silhouette score.*

```
[71]: #Menampilkan keakuratan dari hasil klaster
      from sklearn.metrics import silhouette_score

      silhouette_avg = silhouette_score(x, y_means)
      print(f'Silhouette Score DBSCAN: {silhouette_avg}')

      Silhouette Score DBSCAN: 0.41249187303464097
```

D. ANALISIS HASIL CLUSTERING

Silhouette Score adalah salah satu metode yang digunakan untuk mengevaluasi kualitas dari hasil clustering. Nilai ini mengukur seberapa baik objek (data poin) dikelompokkan dalam cluster-nya dibandingkan dengan cluster lainnya. Semakin tinggi nilai Silhouette Score, semakin baik pemisahan cluster yang terbentuk.

Silhouette Score memiliki nilai antara -1 dan +1, di mana:

- **+1** menunjukkan bahwa data poin sangat dekat dengan cluster-nya sendiri dan jauh dari cluster lainnya (cluster sangat baik).
- **0** menunjukkan bahwa data poin berada di perbatasan antara dua cluster (tidak jelas ke cluster mana seharusnya).
- **-1** menunjukkan bahwa data poin lebih dekat dengan cluster lain daripada dengan cluster-nya sendiri, yang menandakan adanya kesalahan dalam clustering.

1. Menghitung Silhouette Score

Untuk menghitung Silhouette Score, kita memerlukan dua informasi:

- a(i): Jarak rata-rata antara data poin i dan semua titik lain dalam cluster yang sama (seberapa dekat data tersebut dengan cluster-nya sendiri).
- b(i): Jarak rata-rata antara data poin i dan titik terdekat dari cluster lain (seberapa jauh data tersebut dari cluster lainnya).

Untuk setiap titik i, nilai Silhouette Score dihitung dengan rumus:

$$S(i) = \frac{b(i) - a(i)}{\max(a(i), b(i))}$$

- a(i) lebih kecil berarti titik tersebut lebih dekat dengan cluster-nya sendiri.
- b(i) lebih besar berarti titik tersebut lebih jauh dari cluster lain.

2. Evaluasi Kualitas Cluster dengan Silhouette Score

- Cluster yang Baik: Jika Silhouette Score tinggi (mendekati 1), ini menunjukkan bahwa DBSCAN berhasil membentuk cluster dengan jelas terpisah satu sama lain.
- Cluster yang Buruk: Jika Silhouette Score mendekati 0 atau negatif, itu menunjukkan masalah dalam pembentukan cluster, mungkin karena pemilihan parameter yang kurang tepat (misalnya, nilai *epsilon* atau *minPts* yang tidak sesuai).

REFERENSI

Azwarini, Rahmania. 2024. Metode *Clustering* DBSCAN (*Density-Based Spatial Clustering of Applications with Noise*). Dari https://exsight.id/blog/2024/10/30/clustering-dbscan/. Diakses pada 11 Mei 2025.

David. 2020. DBSCAN Clustering. Dari https://algotech.netlify.app/blog/dbscan-clustering/. Diakses pada 11 Mei 2025.

GeeksforGeeks. 2025. *DBSCAN Clustering in ML | Density based clustering*. Dari https://www.geeksforgeeks.org/dbscan-clustering-in-ml-density-based-clustering/. Diakses pada 11 Mei 2025.

Kumar, Rajesh. 2024. *A Guide to the DBSCAN Clustering Algorithm*. Dari https://www.datacamp.com/tutorial/dbscan-clustering-algorithm. Diakses pada 11 Mei 2025.

Sachinsoni. 2023. Clustering Like a Pro: A Beginner's Guide to DBSCAN. Dari https://medium.com/@sachinsoni600517/clustering-like-a-pro-a-beginners-guide-to-dbscan-6c8274c362c4. Diakses pada 11 Mei 2025.

Sharma, Abhishek. 2025. *How to Master the Popular DBSCAN Clustering Algorithm for Machine Learning?* Dari https://www.analyticsvidhya.com/blog/2020/09/how-dbscan-clustering-works/. Diakses pada 11 Mei 2025.