COMPUTER SCIENCE - FMIPA - UNILA

DBSCAN CLUSTERING

Dr. rer. nat. Akmal Junaidi, M.Sc
Rahman Taufik, S.Pd., M.Kom

DISCUSSION

1. Density-Based
2. DBSCAN
3. Algoritma DBSCAN dengan Studi Kasus

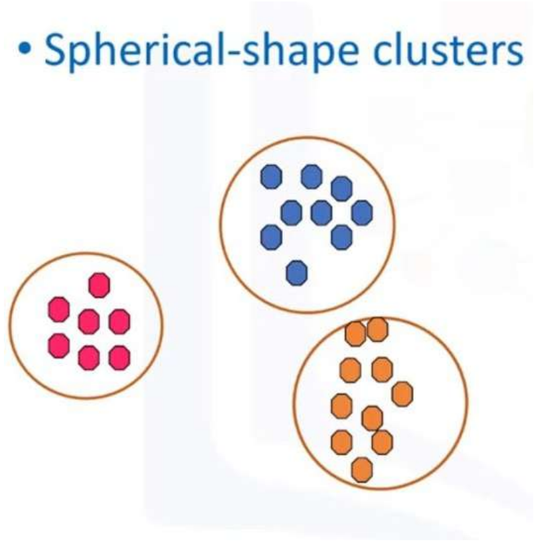
MATERI SEBELUMNYA

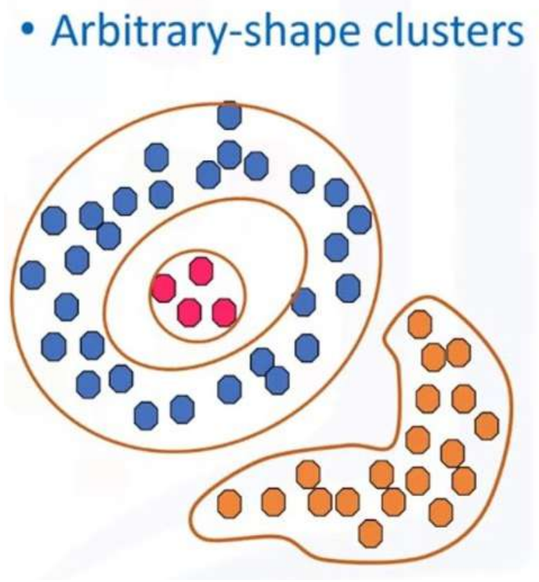
Unsupervised Learning

- Partition-Based
  - K-means Clustering
- Hierarchical-Based
  - Agglomerative Clustering
  - Divisive Clustering

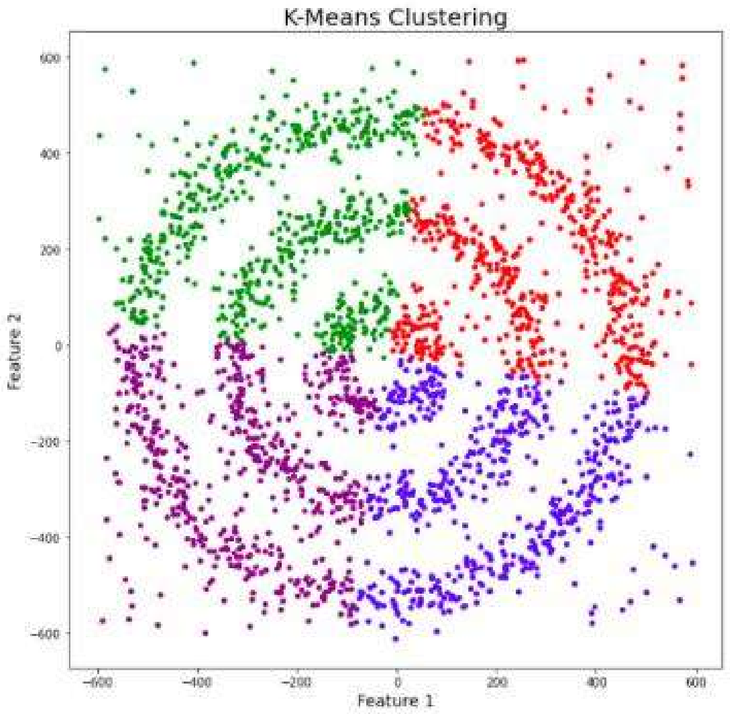
DENSITY-BASED

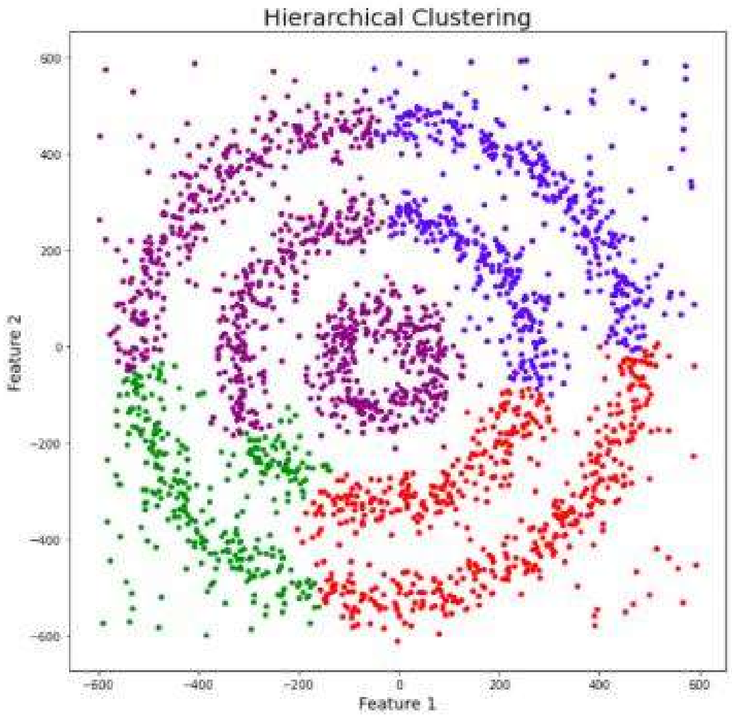
- K-means dan HC (Hierarchical Clustering) cocok untuk data dengan pola yang jelas, tetapi sulit untuk data dengan pola yang sembarang (arbitraty-shape)
- Pendekatan yang dapat digunakan untuk menemukan cluster dari arbitraty-shape adalah density-based
- Density-based clustering terbentuk berdasarkan kepadatan

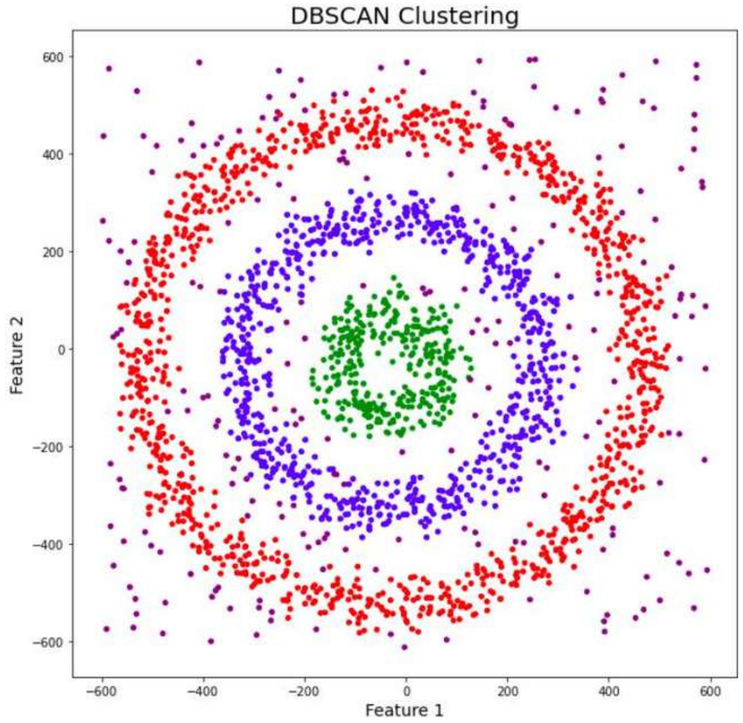




K-MEANS VS HC VS DENSITY-BASED







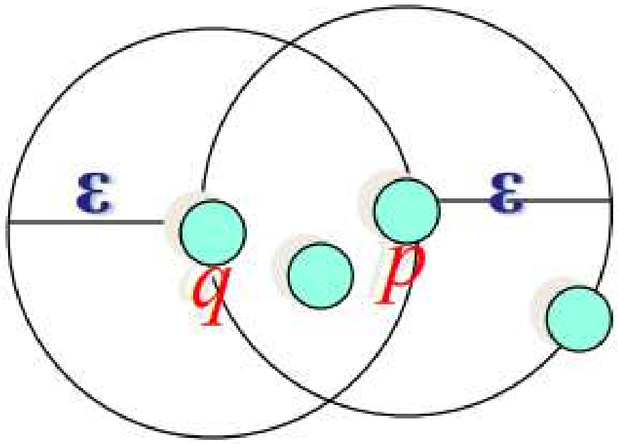
DENSITY-BASED

- Ide dasar:
  - Cluster merupakan daerah padat di ruang data, dipisahkan oleh daerah dengan kepadatan yang lebih rendah
  - Cluster terbentuk berdasarkan kepadatan data, sekumpulan titik-titik yang saling berdekatan, dan terdapat perbedaan kepadatan dengan daerah sekitarnya
- Salah satu teknik clustering dengan pendekatan Density-Based adalah DBSCAN (Density-Based Spatial Clustering of Applications with Noise)

DBSCAN

DBSCAN bekerja dengan menetapkan dua parameter:

- ϵ (epsilon): jari-jari lingkaran yang dibuat di sekitar titik data, jika jarak antara dua titik lebih rendah atau sama dengan ϵ maka dianggap sebagai tetangga (ϵ-neighborhood)
- minPoints: jumlah minimum titik data dalam radius ϵ-neighborhood

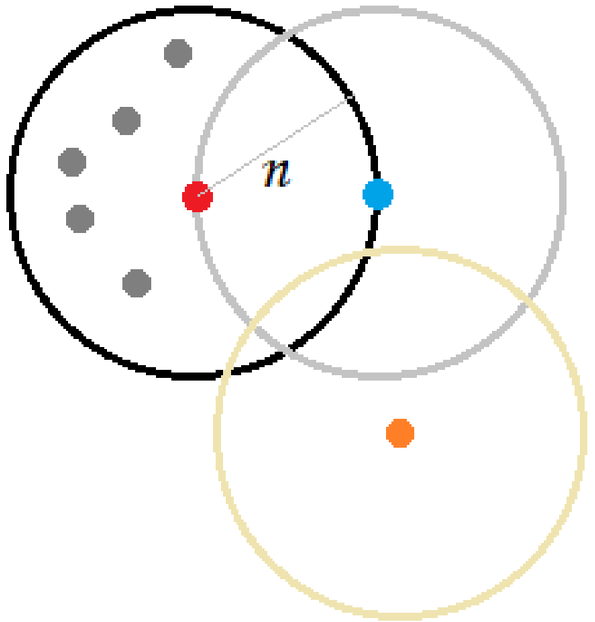


  - ϵ-neighborhood dari p
  - ϵ-neighborhood dari q
  - jika MinPts = 4, maka
    - kepadatan p tinggi
    - kepadatan q rendah

DBSCAN

Terdapat tiga kategori yang harus ditentukan untuk mendapatkan cluster di DBSCAN, yaitu:

- Core points: titik data yang memenuhi syarat minPts dalam radius ϵ
- Border points: titik data yang memiliki jumlah tetangga kurang dari minPts, tetapi berada di perbatasan radius core points
- Noise/Outlier points: titik data yang tidak memiliki tetangga yang memenuhi syarat apapun



Core Point
Border Point
Noise Point
n = Neighbourhood
m = 4

DBSCAN

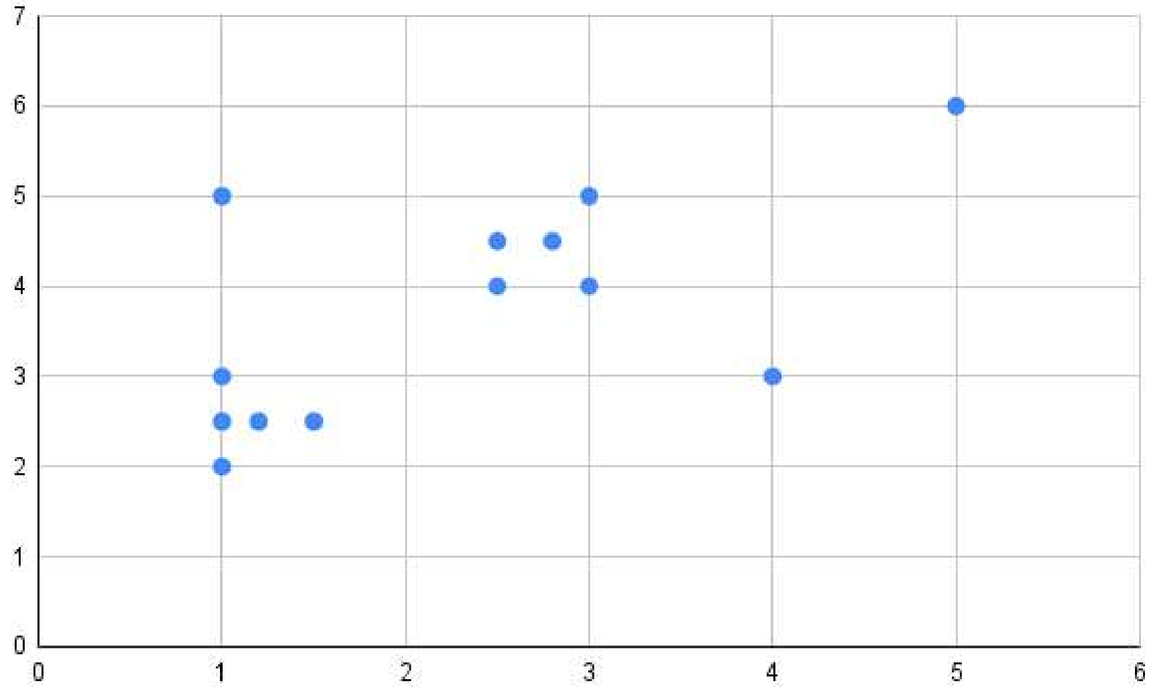
```
for each o ∈ D do
    if o is not yet classified then
        if o is a core-object then
            collect all objects density-reachable from o
            and assign them to a new cluster.
        else
            assign o to NOISE
```

Algoritma

1. Pilih nilai untuk ϵ (epsilon), MinPts, dan titik data acak sebagai titik awal
2. Dari titik awal hitung jaraknya dari setiap titik data lainnya
3. Temukan semua titik tetangga dari x yang berada di dalam radius ϵ (jaraknya dari x lebih kecil atau sama dengan eps)
4. Tandai x sebagai yang suda dikunjungi dan jika jumlah titik tetangga di sekitar x lebih besar atau sama dengan MinPts, perlakukan x sebagai core-point, jika belum ditetapkan ke cluster mana pun, buat sebagai cluster baru
5. Jika jumlah titik tetangga disekitar x kurang dari MinPts dan terdapat core-point sebagai tetangganya, perlakukan sebagai border-point
6. Masukan semua titik-titik yang terhubung oleh kepadatan sebagai satu kluster
7. Ulangi langkah-langkah di atas untuk setiap titik yang belum dikunjungi dalam kumpulan data dan temukan semua core-point, border-point dan outlier-point

DBSCAN - STUDI KASUS

Data



X	Y
1	2
3	4
2.5	4
1.5	2.5
3	5
2.8	4.5
2.5	4.5
1.2	2.5
1	3
1	5
1	2.5
5	6
4	3

eps = 0.6
MinPts = 4
titik awal = (1.2)

X	Y	Jarak dari (1.2)
1	2	0
3	4	2.8
2.5	4	2.5
1.5	2.5	0.7
3	5	3.6
2.8	4.5	3.08
2.5	4.5	2.9
1.2	2.5	**0.53**
1	3	1
1	5	3
1	2.5	**0.5**
5	6	5.6
4	3	3.1

untuk menghitung jarak bisa menggunakan rumus euclidean distance

pada titik (1, 2), yang memenuhi eps <= 0.6 adalah titik (1.2, 2.5) dan (1, 2.5)

maka, tetangga dari titik (1, 2) adalah (1.2, 2.5) dan (1, 2.5)

lakukan perhitungan jarak untuk setiap titik data, maka akan didapatkan tetangganya

tetangga: titik data yang nilai jaraknya <= nilai epsilon yang sudah ditentukan (yaitu 0.6)

titik data	tetangga
(1, 2)	(1.2, 2.5), (1, 2.5)
(3, 4)	(2.5, 4), (2.8, 4.5)
(2.5, 4)	(3, 4), (2.8, 4.5), (2.5, 4.5)
(1.5, 2.5)	(1.2, 2.5), (1, 2.5)
(3, 5)	(2.8, 4.5)
(2.8, 4.5)	(3, 4), (2.5, 4), (3, 5), (2.5, 4.5)
(2.5, 4.5)	(2.5, 4), (2.8, 4.5)
(1.2, 2.5)	(1, 2), (1.5, 2.5), (1, 3), (1, 2.5)
(1, 3)	(1.2, 2.5), (1, 2.5)
(1, 5)	-
(1, 2.5)	(1, 2), (1.5, 2.5), (1.2, 2.5), (1, 3)
(5, 6)	-
(4, 3)	-

titik data	tetangga	kategori
(1, 2)	(1.2, 2.5), (1, 2.5)	border point
(3, 4)	(2.5, 4), (2.8, 4.5)	border point
(2.5, 4)	(3, 4), (2.8, 4.5), (2.5, 4.5)	border point
(1.5, 2.5)	(1.2, 2.5), (1, 2.5)	border point
(3, 5)	(2.8, 4.5)	border point
(2.8, 4.5)	**(3, 4), (2.5, 4), (3, 5), (2.5, 4.5)**	**core point**
(2.5, 4.5)	(2.5, 4), (2.8, 4.5)	border point
(1.2, 2.5)	**(1, 2), (1.5, 2.5), (1, 3), (1, 2.5)**	**core point**
(1, 3)	(1.2, 2.5), (1, 2.5)	border point
(1, 5)	-	outlier point
(1, 2.5)	**(1, 2), (1.5, 2.5), (1.2, 2.5), (1, 3)**	**core point**
(5, 6)	-	outlier point
(4, 3)	-	outlier point

pemetaan kategori titik/point ini didasarkan pada jumlah tetangga dan nilai MinPts

- core point: jumlah tetangga >= MinPts
- border point: jumlah tetangga < MinPts
- outlier point: tidak memiliki tetangga

titik data	tetangga	kategori
(1, 2)	(1.2, 2.5), (1, 2.5)	border point
(3, 4)	(2.5, 4), (2.8, 4.5)	border point
(2.5, 4)	(3, 4), (2.8, 4.5), (2.5, 4.5)	border point
(1.5, 2.5)	(1.2, 2.5), (1, 2.5)	border point
(3, 5)	(2.8, 4.5)	border point
(2.8, 4.5)	**(3, 4), (2.5, 4), (3, 5), (2.5, 4.5)**	**core point**
(2.5, 4.5)	(2.5, 4), (2.8, 4.5)	border point
(1.2, 2.5)	**(1, 2), (1.5, 2.5), (1, 3), (1, 2.5)**	**core point**
(1, 3)	(1.2, 2.5), (1, 2.5)	border point
(1, 5)	-	outlier point
(1, 2.5)	**(1, 2), (1.5, 2.5), (1.2, 2.5), (1, 3)**	**core point**
(5, 6)	-	outlier point
(4, 3)	-	outlier point

cluster baru didasarkan keberadaan core point

<-- cluster 1

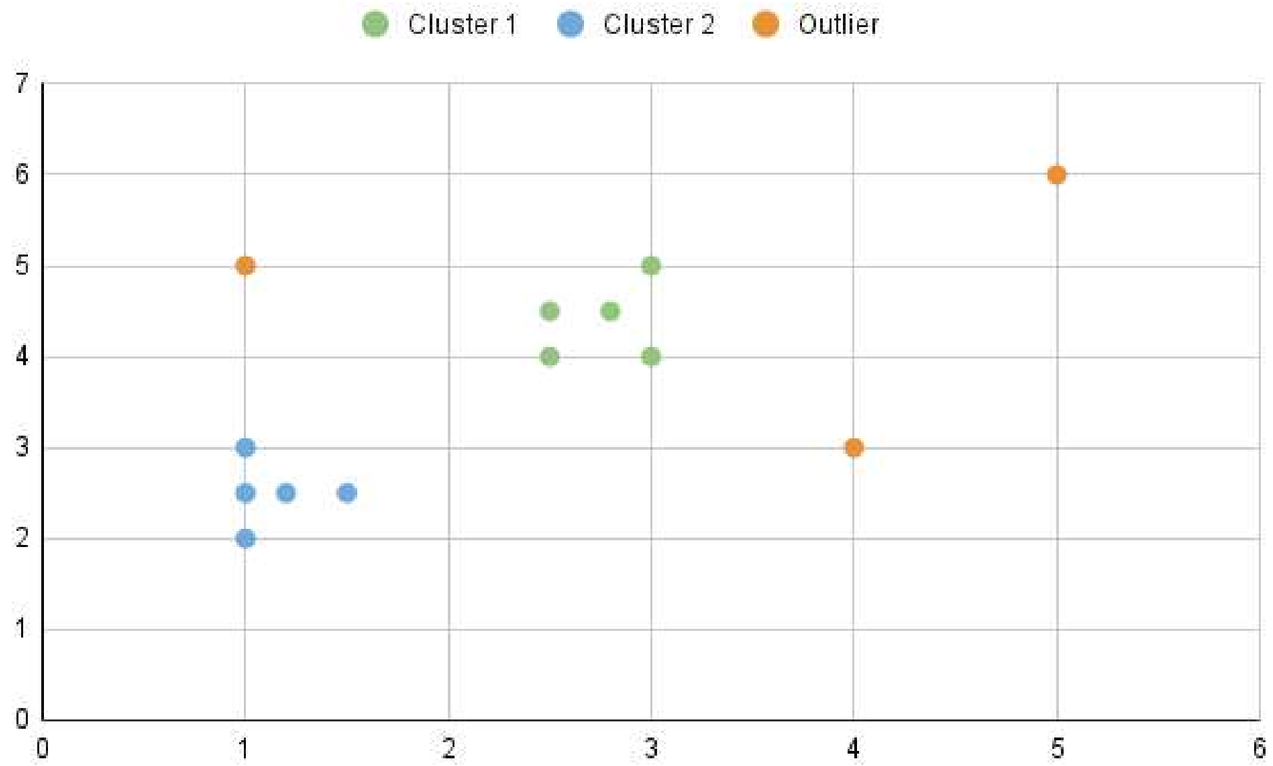
<-- cluster 2

<-- cluster 2

core point (1.2, 2.5) dan (1, 2.5) memiliki tetangga yang sama yaitu (1, 2), maka kedua titik tersebut menjadi cluster yang sama

X	Y	Cluster
1	2	Cluster 2
3	4	Cluster 1
2.5	4	Cluster 1
1.5	2.5	Cluster 2
3	5	Cluster 1
2.8	4.5	Cluster 1
2.5	4.5	Cluster 1
1.2	2.5	Cluster 2
1	3	Cluster 2
1	5	Outlier
1	2.5	Cluster 2
5	6	Outlier
4	3	Outlier

Hasil akhir clusterisasi berdasarkan algoritma DBSCAN



TERIMA KASIH

Referensi

- http://pajarito.materials.cmu.edu/Data_Analytics-lectures/27737-Clustering-L11-24Mar21.pdf
- https://cse.buffalo.edu/~jing/cse601/fa12/materials/clustering_density.pdf
- https://towardsdatascience.com/dbscan-make-density-based-clusters-by-hand-2689dc335120