



BIG DATA & DATA MINING

Lecturer : Kusnawi, S.Kom, M.Eng
Email : Khusnawi@amikom.ac.id
Instagram: @mynameiskhus



✓ BIG DATA & DATA MINING - WEEK 12

Objektif :

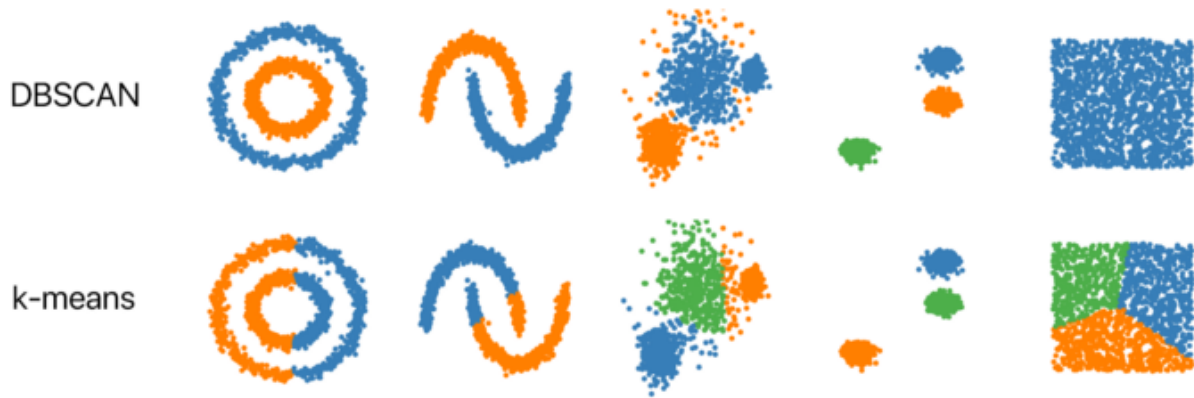
Mahasiswa dapat menerapkan teknik DBSCAN Clustering

✓ DBScan -> Lanjutan Clusering

Tentang DBSCAN

- Density-Based Spatial Clustering Algorithm with Noise (DBSCAN) adalah metode clustering yang berbasis kepadatan (density-based) dari posisi amatan data dengan prinsip mengelompokkan data yang relatif berdekatan. Konsep kepadatan yang dimaksud dalam DBSCAN adalah banyaknya data (minPts) yang berada dalam radius Epsilon (ϵ) dari setiap data. Konsep kepadatan seperti ini menghasilkan tiga macam status dari setiap data, yaitu inti (core), batas (border), dan noise.
- Epsilon merupakan jarak maksimal antara dua data dalam satu cluster yang diizinkan, dan minimum points adalah banyaknya data minimal dalam jarak epsilon agar terbentuk suatu cluster. Metode jarak yang digunakan dalam DBSCAN adalah Euclidian distance.
- DBSCAN lebih cocok untuk data dengan klaster berbentuk tidak beraturan dan mampu menangani outlier secara alami.

Perbedaan dengan KMeans



Langkah Kerja DBSCAN

1. Tentukan nilai minPts dan epsilon (eps) yang akan digunakan.
2. Pilih data awal "p" secara acak.
3. Hitung jarak antara data "p" terhadap semua data menggunakan Euclidian distance.
4. Ambil semua amatan yang density-reachable dengan amatan "p".
5. Jika amatan yang memenuhi nilai epsilon lebih dari jumlah minimal amatan dalam satu gerombol maka amatan "p" dikategorikan sebagai core points dan gerombol terbentuk.
6. Jika amatan "p" adalah border points dan tidak ada amatan yang density-reachable dengan amatan "p", maka lanjutkan pada amatan lainnya.
7. Ulangi langkah 3 sampai 6 hingga semua amatan diproses.

✓ Kasus Mall Customer Segmentation

✓ 1. Import Library

```
import numpy as np
import pandas as pd

import matplotlib.pyplot as plt
import seaborn as sns

from itertools import product
from sklearn.metrics import silhouette_score
from sklearn.cluster import DBSCAN

import warnings
warnings.filterwarnings("ignore")
```

✓ 2. Load Dataset

```
df = pd.read_csv('/content/Mall_Customers.csv')
```

3. EDA

```
df.head()
```

	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

Next steps:

[Generate code with df](#)[New interactive sheet](#)

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
```

```
RangeIndex: 200 entries, 0 to 199
```

```
Data columns (total 5 columns):
```

#	Column	Non-Null Count	Dtype
0	CustomerID	200 non-null	int64
1	Gender	200 non-null	object
2	Age	200 non-null	int64
3	Annual Income (k\$)	200 non-null	int64
4	Spending Score (1-100)	200 non-null	int64

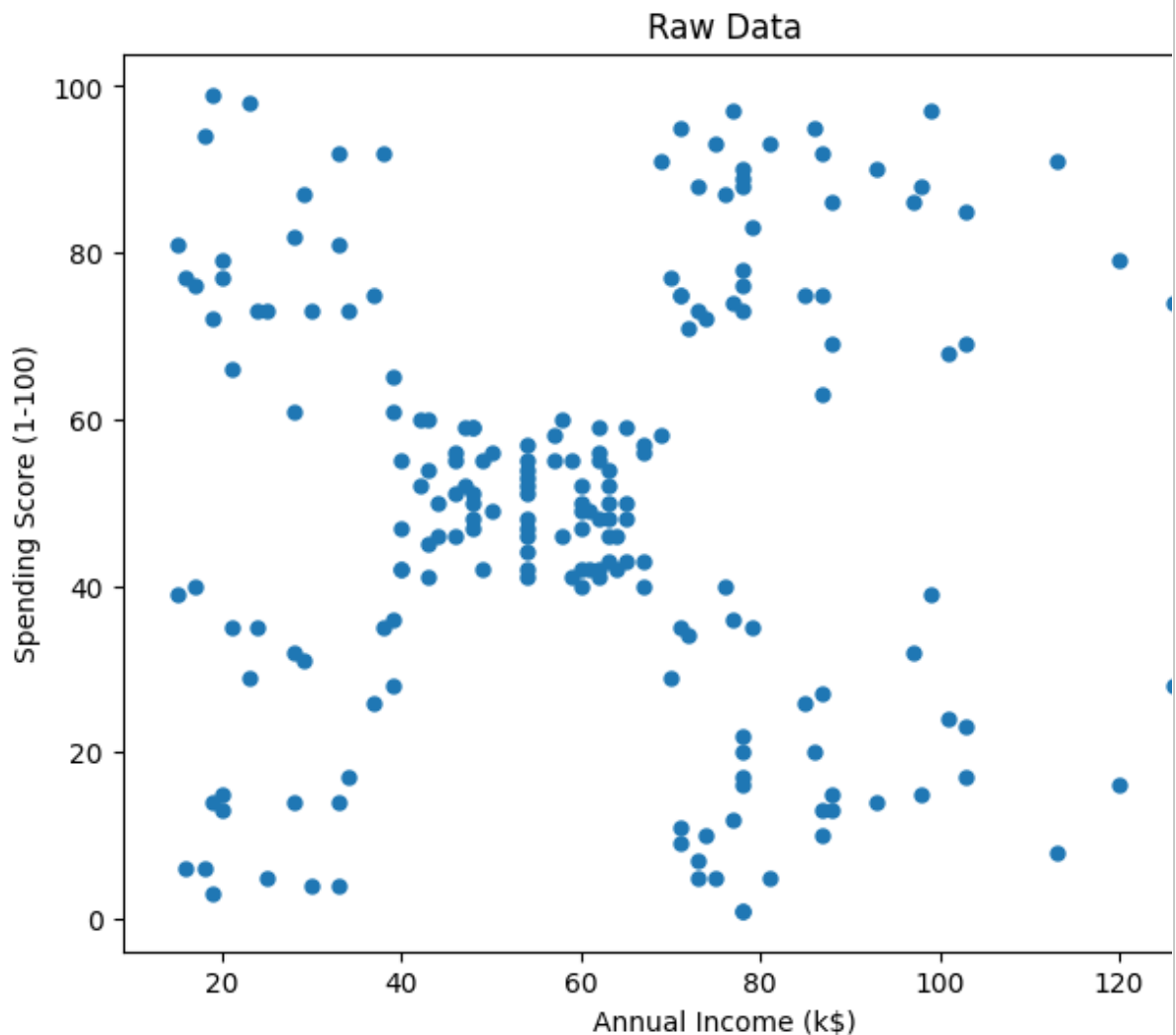
```
dtypes: int64(4), object(1)
```

```
memory usage: 7.9+ KB
```

```
df.describe()
```

	CustomerID	Age	Annual Income (k\$)	Spending Score (1-100)
count	200.000000	200.000000	200.000000	200.000000
mean	100.500000	38.850000	60.560000	50.200000
std	57.879185	13.969007	26.264721	25.823522
min	1.000000	18.000000	15.000000	1.000000
25%	50.750000	28.750000	41.500000	34.750000
50%	100.500000	36.000000	61.500000	50.000000
75%	150.250000	49.000000	78.000000	73.000000
max	200.000000	70.000000	137.000000	99.000000

```
plt.figure(figsize=(8,6))
plt.scatter(df['Annual Income (k$)'], df['Spending Score (1-100)'], s =
plt.title('Raw Data')
plt.xlabel('Annual Income (k$)')
plt.ylabel('Spending Score (1-100)')
plt.show()
```



✓ 4. Select Features

Clustering kali ini akan menggunakan 3 buah fitur yaitu **Age**, **Annual Income**, dan **Spending Score**.

```
X = df[['Annual Income (k$)', 'Spending Score (1-100)']]
```

✓ 5. Determine DBSCAN Hyperparameter

Dalam DBSCAN, terdapat dua hyperparameter utama: (cek lagi Materi di Teori)

- eps (epsilon) - **merupakan jarak maksimum antara dua titik yang akan dianggap tergabung dalam satu cluster**. Nilai yang lebih besar akan

menghasilkan cluster yang lebih besar, sementara nilai yang lebih kecil akan menghasilkan cluster yang lebih kecil.

- `min_samples` (minimum sampel) - **merupakan jumlah sampel minimal yang diperlukan agar suatu titik dapat dianggap sebagai anggota dari suatu cluster**. Nilai yang lebih besar akan menghasilkan cluster yang lebih terbatas, sementara nilai yang lebih kecil akan menghasilkan cluster yang lebih luas.

Salah satu cara untuk menentukan hyperparameter terbaik dengan membuat matriks yang berisi hasil dari beberapa skenario hyperparameter

```
eps_values = np.arange(8,12.75,0.25) #parameter dalam range >=8 & <12.7
min_samples = np.arange(3,10) #parameter dalam range >=3 & <10 dengan k
```

```
DBSCAN_params = list(product(eps_values, min_samples))
DBSCAN_params[:10]
```

```
[(np.float64(8.0), np.int64(3)),
 (np.float64(8.0), np.int64(4)),
 (np.float64(8.0), np.int64(5)),
 (np.float64(8.0), np.int64(6)),
 (np.float64(8.0), np.int64(7)),
 (np.float64(8.0), np.int64(8)),
 (np.float64(8.0), np.int64(9)),
 (np.float64(8.25), np.int64(3)),
 (np.float64(8.25), np.int64(4)),
 (np.float64(8.25), np.int64(5))]
```

X

	Annual Income (k\$)	Spending Score (1-100)	
0	15	39	
1	15	81	
2	16	6	
3	16	77	
4	17	40	
...	...	...	
195	120	79	
196	126	28	
197	126	74	
198	137	18	
199	137	83	

200 rows x 2 columns

Next steps: [Generate code with X](#) [New interactive sheet](#)

```
#melakukan iterasi untuk menghitung silhouette score berdasarkan kombin
#Cek modul lab sebelumnya tentang silhouette score
```

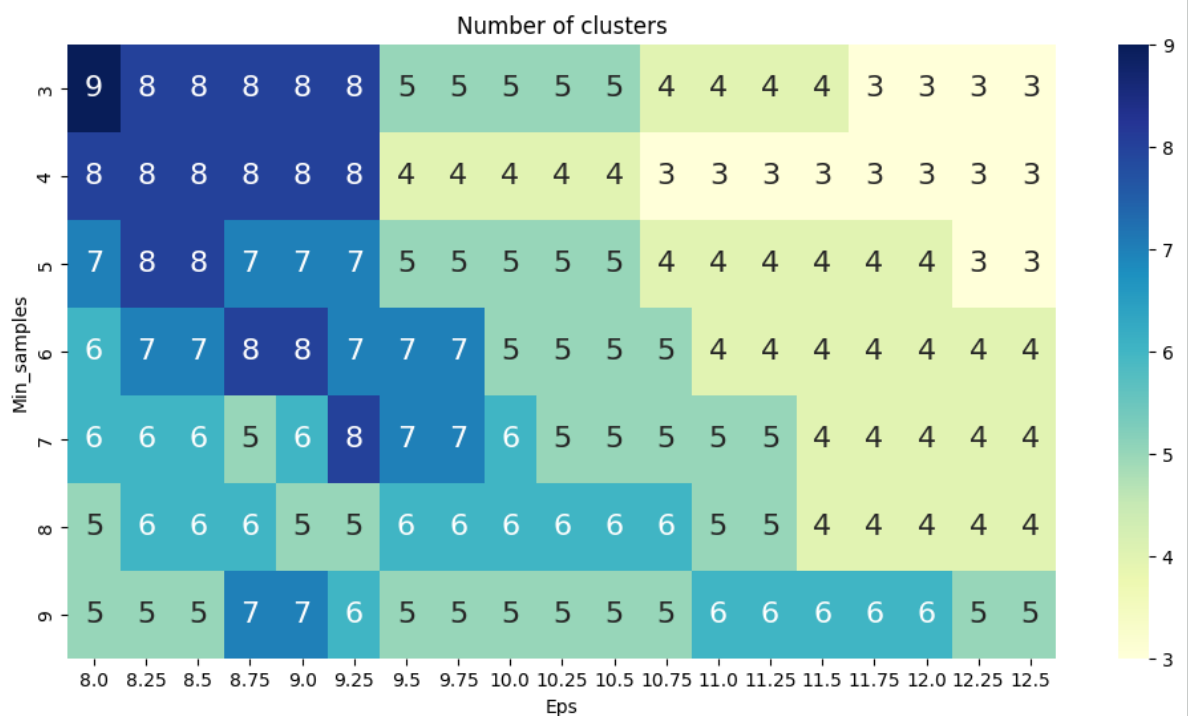
```
no_of_clusters = []
sil_score = []

for p in DBSCAN_params:
    DBS_clustering = DBSCAN(eps=p[0], min_samples=p[1]).fit(X)
    no_of_clusters.append(len(np.unique(DBS_clustering.labels_)))
    sil_score.append(silhouette_score(X, DBS_clustering.labels_))
```

```
tmp = pd.DataFrame.from_records(DBSCAN_params, columns=['Eps', 'Min_sa
tmp['No_of_clusters'] = no_of_clusters
```

```
pivot_1 = pd.pivot_table(tmp, values='No_of_clusters', index='Min_sampl
```

```
fig, ax = plt.subplots(figsize=(12,6))
sns.heatmap(pivot_1, annot=True,annot_kws={"size": 16}, cmap="YlGnBu",
ax.set_title('Number of clusters')
plt.show()
```



- Peningkatan Min_samples (bergerak ke atas pada sumbu Y) juga cenderung mengurangi jumlah cluster karena persyaratan kepadatan menjadi lebih ketat.

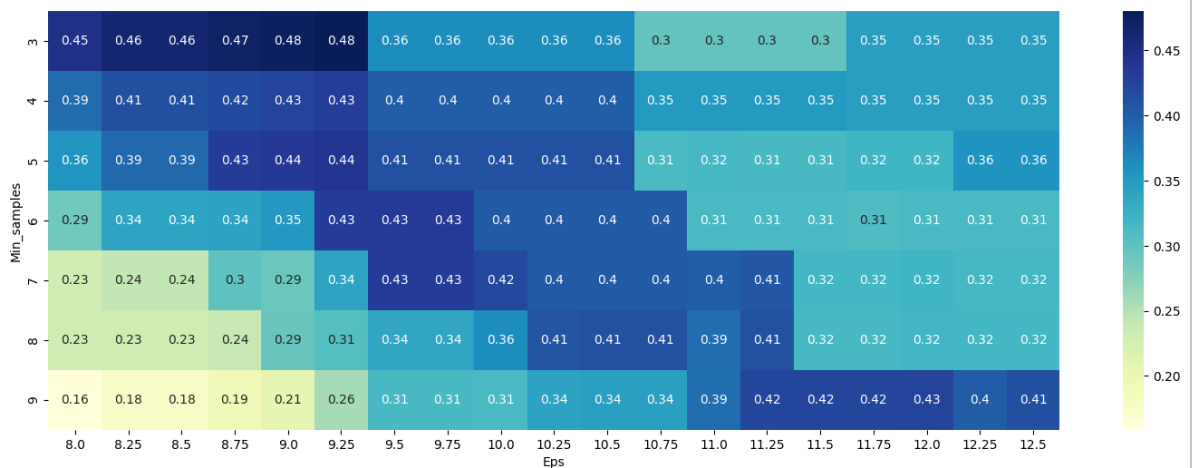
Contohnya, pada Eps=8.0, ketika Min_samples meningkat dari 3 ke 9, jumlah cluster berkurang dari 9 menjadi 5.

- Ketika Min_samples rendah (misalnya 3) dan Eps juga rendah (misalnya 8.0), DBSCAN cenderung menemukan lebih banyak cluster (berkisar 9 cluster).

```
tmp = pd.DataFrame.from_records(DBSCAN_params, columns=['Eps', 'Min_sa
tmp['Sil_score'] = sil_score

pivot_1 = pd.pivot_table(tmp, values='Sil_score', index='Min_samples',

fig, ax = plt.subplots(figsize=(18,6))
sns.heatmap(pivot_1, annot=True, annot_kws={"size": 10}, cmap="YlGnBu",
plt.show())
```



- Berdasarkan plot di atas, Silhouette Score tertinggi adalah 0.414334. Skor ini diperoleh ketika Eps = 12.5 dan Min_samples = 9
- Kombinasi Eps=12.5 dan Min_samples=9 menghasilkan clustering terbaik untuk dataset ini, karena cluster-cluster yang terbentuk paling terpisah dan padat dibandingkan dengan kombinasi hyperparameter lainnya yang diuji.

✓ 6. Create Cluster

```
#sample
DBS_clustering = DBSCAN(eps=12.5, min_samples=9).fit(X)

DBSCAN_clustered = X.copy()
DBSCAN_clustered.loc[:, 'Cluster'] = DBS_clustering.labels_
DBSCAN_clustered.head(10)
```

	Annual Income (k\$)	Spending Score (1-100)	Cluster	
0	15	39	-1	
1	15	81	0	
2	16	6	1	
3	16	77	0	
4	17	40	-1	
5	17	76	0	
6	18	6	1	
7	18	94	-1	
8	19	3	1	
9	19	72	0	

Next steps:

[Generate code with DBSCAN_clustered](#)

[New interactive sheet](#)

- Nilai -1: Ini menunjukkan bahwa titik data tersebut dianggap sebagai noise atau outlier oleh algoritma DBSCAN. Contohnya, pelanggan dengan CustomerID yang awalnya 1 dan 5 (dengan index 0 dan 4) dianggap outlier.

```
#menampilkan distribusi cluster
DBSCAN_clust_sizes = DBSCAN_clustered.groupby('Cluster').size().to_frame
DBSCAN_clust_sizes.columns = ["DBSCAN_size"]
DBSCAN_clust_sizes
```

	DBSCAN_size	
Cluster		
-1	17	
0	112	
1	12	
2	33	
3	26	

Next steps:

[Generate code with DBSCAN_clust_sizes](#)

[New interactive sheet](#)

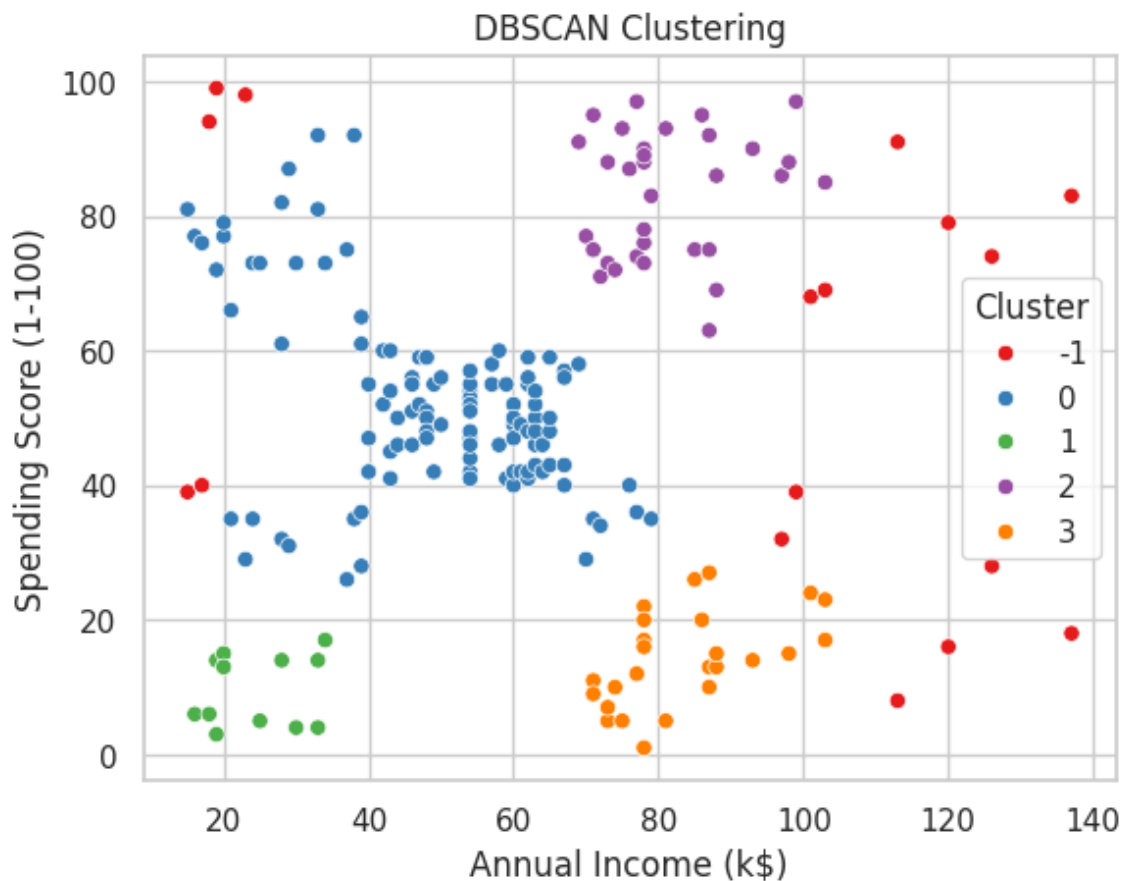
7. Visualisasi Plot

```
print(DBSCAN_clustered[DBSCAN_clustered['Cluster']!=-1])
```

	Annual Income (k\$)	Spending Score (1-100)	Cluster
1	15	81	0
2	16	6	1
3	16	77	0
5	17	76	0
6	18	6	1
..	...	...	...
185	99	97	2
186	101	24	3
188	103	17	3
189	103	85	2
190	103	23	3

[183 rows x 3 columns]

```
sns.set(style="whitegrid")
sns.scatterplot(data=DBSCAN_clustered,
                x="Annual Income (k$)",
                y="Spending Score (1-100)",
                hue="Cluster",
                palette="Set1")
plt.title("DBSCAN Clustering")
plt.show()
```



Penjelasan

- Titik Merah (-1 Cluster): Titik-titik berwarna merah (atau warna yang paling berbeda dari palet 'Set1') adalah outlier atau noise yang tidak termasuk dalam kluster mana pun. Ini adalah kekuatan DBSCAN yang secara eksplisit mengidentifikasi data yang tidak sesuai pola.
- Klaster yang Teridentifikasi (0, 1, 2, 3): Anda akan melihat beberapa kelompok warna yang berbeda, merepresentasikan cluster-cluster yang padat dan terpisah.
- Klaster 0 (Biru): mengelompokkan pelanggan dengan pendapatan rendah tetapi skor pengeluaran tinggi.
- Klaster 1 (Oranye): mengelompokkan pelanggan dengan pendapatan rendah dan skor pengeluaran rendah.
- Klaster 2 (Hijau): mengelompokkan pelanggan dengan pendapatan tinggi dan skor pengeluaran tinggi.
- Klaster 3 (Ungu): mengelompokkan pelanggan dengan pendapatan tinggi dan skor pengeluaran rendah.

TUGAS 🐱

- Lakukan hyperparameter tuning (*eps* dan *min_samples*) dalam pembuatan cluster DBSCAN menggunakan dataset yang sama (Mall_Customers.csv).
- Nilai epsilon diperoleh dari elbow point plot NearestNeighbor, gunakan KneeLocator agar memudahkan mencari elbow point (dapat dilihat pada [DBSCAN Clustering Step - Elbow Point](#)).
- Gunakan nilai $min_samples = 2 \times jumlah\ fitur - 1$ [\[ref\]](#) kemudian amati dan lihat perbedaan hasilnya (jumlah outlier dan jumlah cluster).
- Lakukan beberapa kali percobaan dengan mengubah *n_neighbors* untuk menemukan hasil terbaik (salah satu caranya ditandai dengan jumlah outlier yang sedikit). Berikan kesimpulan mengenai perbedaan hasilnya, dan jelaskan apakah penggunaan *n_neighbors* lebih baik dibandingkan dengan iterasi beberapa kombinasi hyperparameter untuk menentukan hyperparameternya?
- Silahkan baca beberapa artikel berikut sebelum mencoba [Tentang Hyperparameter](#), [DBSCAN Clustering Step](#) atau [Determine Optimal Value for Epsilon](#)

Kerjakan pada notebook/google colab baru. Untuk memudahkan pengerjaan dapat mengikuti step di bawah

```
#import library
```

```
#load dataset
```

```
#select features
```

```
#create elbow using NearestNeighbor
```

```
#locate elbow using KneeLocator
```

```
#create cluster
```

```
#show cluster distribution
```

```
#visualize cluster
```

Link Pengumpulannya :<https://forms.gle/ar1Yoy9rhTLEiPZ16>

Double-click (or enter) to edit

References :

<https://www.kdnuggets.com/2022/08/implementing-dbscan-python.html>

<https://towardsdatascience.com/understanding-dbscan-and-implementation-with-python-5de75a786f9f>

<https://softscients.com/2021/05/06/perbandingan-clustering-kmeans-dengan-dbscan/>

<https://www.kaggle.com/code/datark1/customers-clustering-k-means-dbscan-and-ap>

<https://algotech.netlify.app/blog/dbscan-clustering/>

<https://media.neliti.com/media/publications/97306-ID-perbandingan-metode-k-means-dan-metode-d.pdf>