1. Jelaskan cara kerja dari algoritma tersebut!
Jawab:
DBSCAN adalah algoritma clustering berbasis kepadatan yang bekerja dengan mengelompokkan data berdasarkan area yang padat dan memisahkan data yang berada di area jarang atau "noise". DBSCAN bekerja dengan mendeteksi cluster yang berbentuk tak beraturan berdasarkan dua parameter utama:

- Epsilon (ε): Jarak maksimum antara dua titik agar keduanya dapat dianggap sebagai bagian dari satu cluster.
- Min Samples: Jumlah minimum titik yang harus ada dalam lingkungan radius ε agar suatu titik dapat dianggap sebagai core point (titik pusat dari cluster).

Langkah-langkahnya adalah sebagai berikut:

- Untuk setiap titik yang belum dikunjungi:
  - Jika titik tersebut memiliki setidaknya Min Samples titik dalam radius ε, maka titik tersebut menjadi core point dan cluster baru dimulai.
  - Tetangga dalam radius ε dari core point tersebut kemudian dijadikan bagian dari cluster yang sama.
- Untuk setiap titik tetangga, jika tetangga tersebut adalah core point, maka proses meluas ke tetangganya, dan seterusnya.
- Proses diulang sampai semua core point dan tetangganya telah diproses, atau semua titik telah dikunjungi.
- Titik yang tidak dapat dimasukkan ke dalam cluster mana pun dianggap sebagai noise.

4. Bandingkan hasil evaluasi model pada nomor 2 dan 3, bagaimana hasil perbandingannya? Jika ada perbedaan, jelaskan alasannya!
Jawab:

Menggunakan Implementasi DBSCAN dari Scratch

```
import dbscan
importlib.reload(dbscan)

from dbscan import DBSCANFromScratch

dbscan_scratch = DBSCANFromScratch(eps=45.3, min_samples=5, metric='euclidean')
dbscan_scratch.fit(X)

scratch_labels = dbscan_scratch.labels
print("Labels (Scratch):", scratch_labels, sep="\n")

# Silhouette Score
score = silhouette_score(X, dbscan_scratch.labels)
print("\nSilhouette Score (Scratch):", score)
```

✓ 0.0s

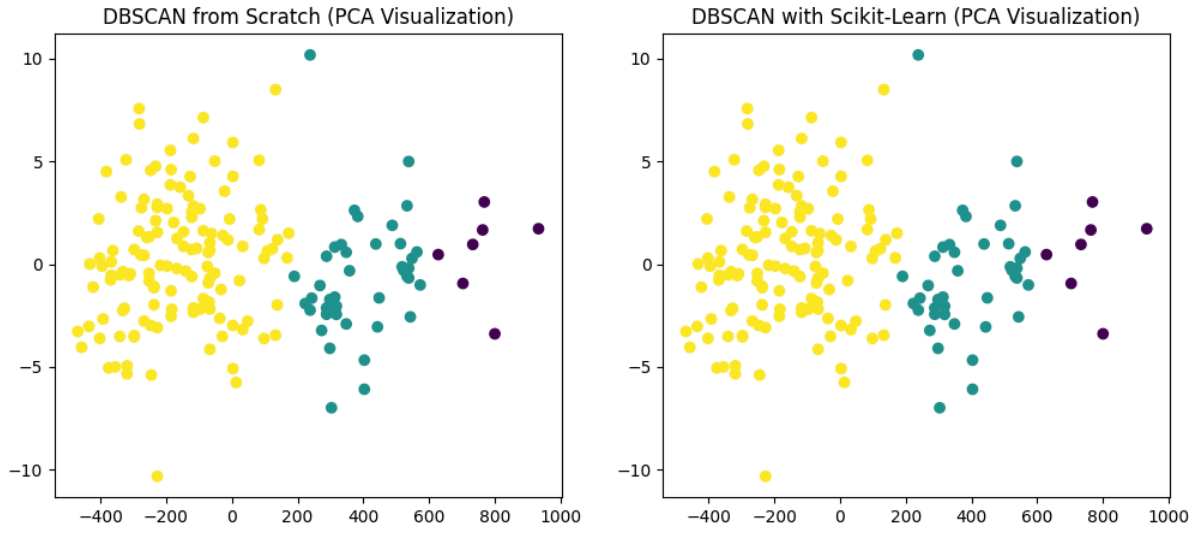
```
Labels (Scratch):
[ 0  0  0 -1  1 -1  0  0  0  0 -1  0  0  0 -1  0  0  0 -1  1  1  1  0  0
  1  1  0  0  1  0  0 -1  0  0  0  1  1  0  0  1  1  0  0  1  1  0  0  0
  0  0  0  0  0 -1  0  0  0  0  0  1  1  1  1  1  1  1  1  1  1  1  1  1
  1  0  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  0
  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1
  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1
  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1
  1  1  1  1  1  1  1  1  1  1]

Silhouette Score (Scratch): 0.6086433275759079
```

Menggunakan Implementasi DBSCAN dari Scikit-Learn

```
dbscan_sklearn = DBSCAN(eps=45.3, min_samples=5, metric='euclidean')
dbscan_sklearn_labels = dbscan_sklearn.fit_predict(X)

sklearn_labels = dbscan_sklearn.labels_
print("Labels (Scikit-Learn):", sklearn_labels, sep="\n")

# Silhouette Score
score = silhouette_score(X, dbscan_sklearn_labels)
print("\nSilhouette Score (Scikit-Learn):", score)
```

✓ 0.0s

```
Labels (Scikit-Learn):
[ 0  0  0 -1  1 -1  0  0  0  0 -1  0  0  0 -1  0  0  0 -1  1  1  1  0  0
  1  1  0  0  1  0  0 -1  0  0  0  1  1  0  0  1  1  0  0  1  1  0  0  0
  0  0  0  0  0 -1  0  0  0  0  0  1  1  1  1  1  1  1  1  1  1  1  1  1
  1  0  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  0
  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1
  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1
  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1  1
  1  1  1  1  1  1  1  1  1  1]

Silhouette Score (Scikit-Learn): 0.6086433275759079
```



Tidak ada perbedaan dalam hasil evaluasi antara implementasi dari scratch dan dari Scikit-Learn.