1. Cara kerja algoritma DBSCAN dari scratch pada repository ini

a. Inisialisasi parameter

Algoritma DBSCAN dimulai dengan menginisialisasi beberapa parameter penting yaitu

- eps (epsilon)
  Sebagai yang menentukan radius lingkungan sekitar suatu titik data di mana titik lain dianggap sebagai tetangga
- min_samples
  Sebagai yang menetapkan jumlah minimum titik data yang diperlukan untuk membentuk sebuah cluster
- Metric
  Sebagai yang menentukan cara perhitungan jarak (misalnya, Euclidean, Manhattan, atau Minkowski).
- Parameter tambahan p
  Digunakan khusus untuk perhitungan jarak Minkowski, memberikan fleksibilitas lebih dalam memilih metrik jarak.

b. Menghitung jarak antar titik data

Untuk setiap pasangan titik data, algoritma menghitung jarak di antara mereka menggunakan metrik jarak yang ditentukan. Misalnya, jika metrik jaraknya adalah 'euclidean', algoritma menggunakan norma Euclidean untuk menghitung jarak. Jika metrik jaraknya adalah 'manhattan', algoritma menghitung jarak absolut total, dan untuk 'minkowski', jarak dihitung menggunakan rumus Minkowski dengan parameter p yang dapat disesuaikan.

c. Pencarian tetangga (*Region Query*)

Algoritma kemudian menjalankan fungsi region_query untuk setiap titik data. Fungsi ini mencari semua titik yang berada dalam radius eps dari titik saat ini (titik inti). Semua titik yang ditemukan dalam radius ini dianggap sebagai tetangga dan disimpan dalam daftar tetangga. Titik dengan jumlah tetangga yang lebih sedikit dari min_samples tidak dianggap sebagai titik inti dan bisa menjadi noise.

d. Ekspansi cluster (*Expand Cluster*)

Jika suatu titik inti memiliki setidaknya min_samples tetangga, maka titik tersebut memulai sebuah cluster baru. Algoritma kemudian memperluas cluster ini dengan memeriksa tetangga-tetangganya. Untuk setiap tetangga yang merupakan titik inti, tetangga lebih lanjut ditemukan dan ditambahkan ke cluster. Proses ini berulang hingga tidak ada lagi titik baru yang bisa ditambahkan ke cluster saat ini.

e. Penentuan noise dan pemberian label cluster

Titik-titik data yang tidak memiliki cukup tetangga (kurang dari min_samples) dalam radius eps diberi label sebagai noise dengan nilai -1. Sementara itu, titik-titik lain yang berhasil dimasukkan ke dalam cluster diberi label sesuai dengan cluster mereka masing-masing.

f. Output hasil clustering
   Setelah semua titik data diproses, algoritma menghasilkan label cluster untuk setiap titik data. Label ini menunjukkan ke mana setiap titik diklasifikasikan, baik ke dalam cluster tertentu atau sebagai noise. Hasil akhir berupa array yang menyimpan label cluster untuk setiap titik data, di mana noise ditandai dengan -1.

4. Perbandingan hasil evaluasi model

Berdasarkan hasil evaluasi, kedua model DBSCAN, baik yang diimplementasikan dari scratch (DBSCANScratch) maupun yang menggunakan pustaka scikit-learn (DBSCAN_Sklearn), menunjukkan hasil yang sangat mirip. Kedua model menghasilkan Silhouette Score yang sama, yaitu -0.1434, menunjukkan bahwa kualitas clustering yang dihasilkan oleh kedua model serupa. Visualisasi clustering menunjukkan bahwa kedua model mengidentifikasi noise (ditandai dengan warna berbeda) dan sebagian besar data termasuk dalam satu cluster besar.

Kemiripan hasil ini mengindikasikan bahwa implementasi DBSCANScratch sudah cukup akurat dan sesuai dengan metode DBSCAN yang diimplementasikan dalam pustaka scikit-learn. Namun, nilai Silhouette Score yang negatif menunjukkan bahwa data mungkin tidak cocok untuk clustering menggunakan DBSCAN dengan parameter saat ini, atau parameter eps dan min_samples tidak optimal. Hal ini mengarah ke interpretasi bahwa data mungkin terlalu tersebar atau jarak antar cluster tidak cukup signifikan, sehingga algoritma kesulitan memisahkan titik-titik data menjadi cluster yang berbeda.