LAPORAN TUGAS PRATIKUM MANDIRI

Judul Tugas: Clustering Data GPS Menggunakan Algoritma DBSCAN untuk Analisis Kepadatan Titik

Nama Mahasiswa	: Al Hijir
Program studi	: Teknik Informatika, STT Terpadu Nurul Fikri, Depok
E-mail	: 0110224222@student.nurulfikri.ac.id

ABSTRAK

Pada praktikum ini dilakukan analisis clustering menggunakan algoritma DBSCAN terhadap data koordinat GPS. DBSCAN merupakan metode *density-based clustering* yang mampu mengelompokkan titik berdasarkan tingkat kepadatannya tanpa perlu menentukan jumlah cluster di awal. Dataset yang digunakan berisi titik-titik GPS berupa latitude dan longitude. Data terlebih dahulu dikonversi ke radian dan diproses menggunakan DBSCAN dengan *metric* haversine agar perhitungan jarak sesuai dengan permukaan bumi. Selain itu, dilakukan perhitungan estimasi luas area tiap cluster menggunakan *bounding box* untuk memperoleh nilai *density* (jumlah titik per km²). Hasil analisis berupa pembentukan cluster, visualisasi sebaran titik, serta ringkasan tiap cluster meliputi jumlah titik, luas area, dan tingkat kepadatan. Praktikum ini membuktikan bahwa DBSCAN efektif digunakan pada data spasial seperti GPS untuk mengidentifikasi wilayah padat, noise, dan pola penyebaran titik.

1. Mengakses File di Google Drive

```
from google.colab import drive
drive.mount('/content/gdrive')
```

```
Drive already mounted at /content/gdrive; to atte
```

Penjelasan:

Digunakan untuk menghubungkan Google Drive agar file dataset dapat diakses langsung di

lingkungan Google Colab.

2. Menentukan Path Folder

```
path = "/content/gdrive/MyDrive/praktikmML/praktikum11"
```

Penjelasan:

Variabel path digunakan untuk menentukan lokasi folder kerja tempat dataset disimpan.

3. Import Library yang Dibutuhkan

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from sklearn.cluster import DBSCAN
from math import radians, cos
```

Penjelasan fungsi library:

Library	Kegunaan
pandas	Membaca dataset CSV dan mengolah data
numpy	Mengubah data ke array numerik
matplotlib	Membuat visualisasi grafik
sklearn.cluster.DBSCAN	Algoritma clustering DBSCAN
math.radians, cos	Dipakai untuk menghitung area bounding box

4. Load Dataset GPS

```
import pandas as pd

df = pd.read_csv('/content/gdrive/MyDrive/PraktikumML/praktikum11/data/gps_dataset.csv')
df.head()
```

	id	lat	lon	source
0	0	-6.195429	106.802189	cluster_1
1	1	-6.215600	106.805788	cluster_1
2	2	-6.188743	106.848593	cluster_1
3	3	-6.185892	106.804345	cluster_1
4	4	-6.229266	106.829243	cluster_1

Langkah berikutnya: New interactive sheet

Penjelasan:

- Membaca dataset gps_dataset.csv yang berisi kolom **latitude (lat)** dan **longitude (lon)**.
- df.head() menampilkan 5 baris pertama data.

5. Mengambil Kolom GPS dan Mengubah ke Radian

```
coords = df[["lat", "lon"]].to_numpy()
coords_rad = np.radians(coords)
```

Penjelasan:

- DBSCAN dengan metric *haversine* membutuhkan data dalam **radian**, bukan derajat.
- Karena itu latitude dan longitude diubah menjadi radian.

6. Menentukan Parameter DBSCAN

```
# Parameter DBSCAN
earth_radius_km = 6371.0088
eps_km = 0.8              # Jarak radius tetangga 0.8 km
eps_rad = eps_km / earth_radius_km
min_samples = 8           # Minimum point dalam radius
```

Penjelasan parameter:

- eps_km: radius pencarian tetangga dalam kilometer (0.8 km).
- eps_rad: DBSCAN dengan *haversine* membutuhkan eps dalam **radian**, jadi dikonversi.
- min_samples: jumlah minimal titik dalam radius eps untuk membentuk cluster.

7. Menjalankan Algoritma DBSCAN

```
# Clustering DBSCAN
db = DBSCAN(eps=eps_rad, min_samples=min_samples, metric="haversine
labels = db.fit_predict(coords_rad)

df["cluster"] = labels
df.head()
```

	id	lat	lon	source	cluster
0	0	-6.195429	106.802189	cluster_1	0
1	1	-6.215600	106.805788	cluster_1	0
2	2	-6.188743	106.848593	cluster_1	-1
3	3	-6.185892	106.804345	cluster_1	0
4	4	-6.229266	106.829243	cluster_1	-1

Langkah berikutnya: New interactive sheet

Penjelasan:

- DBSCAN mengelompokkan titik berdasarkan **kepadatan** (density-based).
- metric="haversine" digunakan untuk menghitung jarak permukaan bumi.
- labels berisi nomor cluster untuk setiap titik.
    - -1 berarti **noise** atau titik yang tidak masuk cluster mana pun.
- Kolom baru **cluster** ditambahkan ke dataframe.

8. Fungsi Untuk Menghitung Area Bounding Box

```
def approx_bbox_area_km2(lat_list, lon_list):
    if len(lat_list) == 0:
        return 0
    lat_min, lat_max = min(lat_list), max(lat_list)
    lon_min, lon_max = min(lon_list), max(lon_list)

    mean_lat = (lat_min + lat_max) / 2
    km_lat = 111.32
    km_lon = 111.32 * cos(radians(mean_lat))

    height = (lat_max - lat_min) * km_lat
    width = (lon_max - lon_min) * km_lon

    area = abs(height * width)
    return area if area > 0 else 0.01
```

Loading...

Penjelasan:
Fungsi ini menghitung **perkiraan luas area cluster (km²)** menggunakan bounding box:

- Mengambil nilai minimum dan maksimum latitude & longitude.
- Mengubah perbedaan derajat menjadi kilometer.
- Menghitung luas persegi panjang (height × width).
- Jika area terlalu kecil → diberi nilai minimal 0.01 km² agar tidak membagi nol.

9. Menghitung Ringkasan Cluster (Jumlah Titik, Luas, Density)

```
clusters = []

for label in sorted(df["cluster"].unique()):
    sub = df[df["cluster"] == label]
    area = approx_bbox_area_km2(sub["lat"], sub["lon"])
    density = len(sub) / area
    clusters.append({
        "cluster": label,
        "jumlah_titik": len(sub),
        "area_km2(bbox)": round(area, 4),
        "density_titik/km2": round(density, 4)
    })

summary = pd.DataFrame(clusters)
summary
```

	cluster	jumlah_titik	area_km2(bbox)	density_titik/km2
0	-1	188	160324.9308	0.0012
1	0	134	30.3813	4.4106
2	1	93	24.2224	3.8394
3	2	85	27.2714	3.1168

Langkah berikutnya: New interactive sheet

Penjelasan:

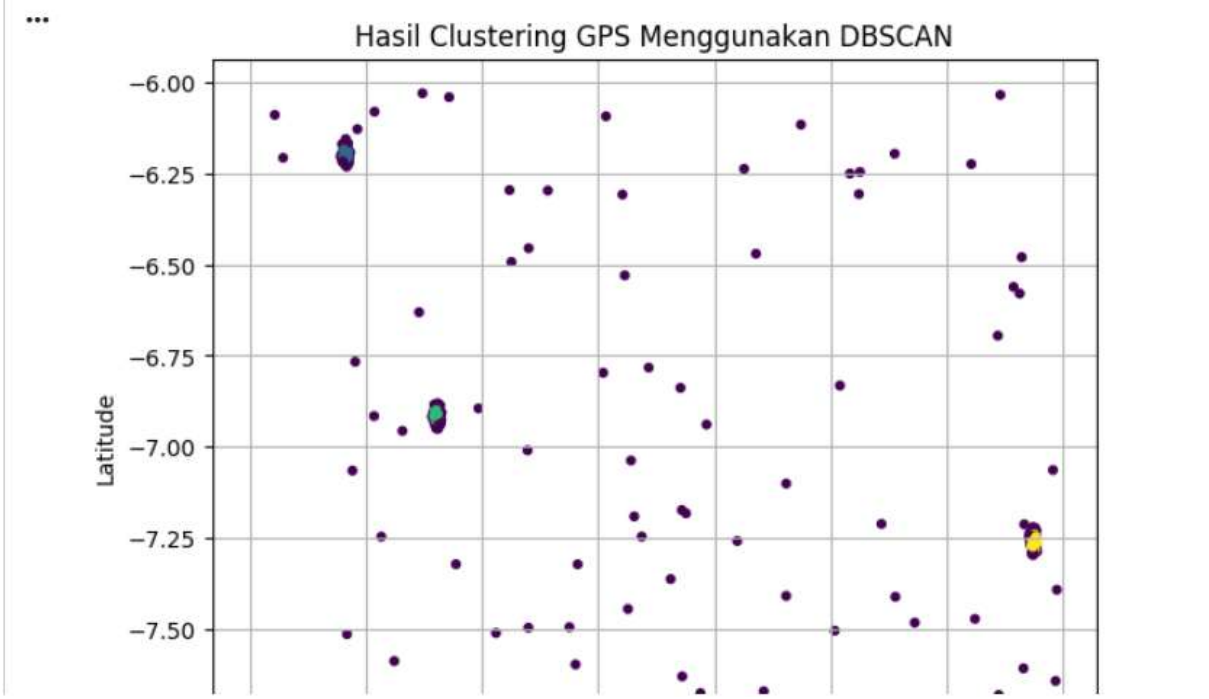
Untuk tiap cluster:

- Mengambil datanya (sub).
- Menghitung luas area.

- Menghitung **density = jumlah titik / area cluster**.
- Menyimpan semuanya ke dataframe summary.

10. Visualisasi Hasil Clustering

```
plt.figure(figsize=(7, 6))
plt.scatter(df["lon"], df["lat"], c=df["cluster"], s=12)
plt.title("Hasil Clustering GPS Menggunakan DBSCAN")
plt.xlabel("Longitude")
plt.ylabel("Latitude")
plt.grid(True)
plt.show()
```



Penjelasan:

- Membuat scatter plot GPS.
- Warna berdasarkan nomor cluster.
- Titik dengan cluster = -1 akan terlihat berbeda → itu adalah **noise**.

11. Menyimpan Hasil ke File

```
df.to_csv("hasil_dbscan_gps.csv", index=False)
summary.to_csv("ringkasan_cluster.csv", index=False)
print("File hasil disimpan:")
print("- hasil_dbscan_gps.csv")
print("- ringkasan_cluster.csv")
```

```
File hasil disimpan:
- hasil_dbscan_gps.csv
- ringkasan_cluster.csv
```

Penjelasan:

- Menyimpan hasil clustering lengkap ke file CSV.
- summary disimpan sebagai ringkasan cluster (jumlah titik, luas area, density).
- Dua file akan muncul di output Colab.