Tugas Praktikum Mandiri_11

Nama :Muhamad Aditia
Nim :0110224213
Link Git :https://github.com/muhammadaditia433/Machine-Leraning
Email :0110224213@student.nurulfikri.ac.id

Abstrak

Praktikum ini menerapkan algoritma **DBSCAN** untuk menganalisis kepadatan lokasi berdasarkan koordinat GPS. Titik-titik GPS dikelompokkan menjadi cluster berdasarkan kedekatan, sedangkan titik yang berjauhan dianggap **noise**. Hasil clustering divisualisasikan menggunakan scatter plot untuk memahami distribusi geografis. DBSCAN terbukti efektif untuk data geografis, dan praktikum ini memberikan pemahaman tentang pentingnya parameter eps dan min_samples, serta interpretasi hasil clustering pada analisis data lokasi.

1.Mount Drive

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

Penjelasan :
Kode ini **menghubungkan Colab dengan Google Drive** sehingga kita bisa membaca, menulis, dan menyimpan file di Drive seolah-olah file tersebut ada di komputer lokal.

2. Import Library

```
import pandas as pd
import numpy as np
from sklearn.cluster import DBSCAN
import matplotlib.pyplot as plt
```

Penjelasan :
Library ini dibutuhkan untuk memuat data GPS, melakukan clustering berbasis densitas,
serta menampilkan hasilnya dalam plot agar mudah dianalisis.

3. Load Dataset dari Google Drive

```
# Read Dataset
df = pd.read_csv('/content/drive/MyDrive/praktikum_ml/Pertemuan-11/Data/koordinat_gps.csv')
df.head()
```

Penjelasan :
Membaca file CSV berisi koordinat GPS (Latitude dan Longitude).
print(df.head()) menampilkan 5 baris pertama untuk memastikan data terbaca dengan benar.

4. Siapkan koordinat untuk DBSCAN

```
coords = df[['lat', 'lon']].values
coords_rad = np.radians(coords)
```

Penjelasan :
DBSCAN dengan metric haversine membutuhkan data dalam radian.
coords_rad berisi koordinat GPS siap digunakan untuk clustering.

5. Terapkan DBSCAN

```
eps_rad = 300 / 6371000  # 300 meter dalam radian

dbscan = DBSCAN(
    eps=eps_rad,
    min_samples=5,
    metric='haversine'
)

clusters = dbscan.fit_predict(coords_rad)
df['cluster'] = clusters

print("\nJumlah titik per cluster:")
print(df['cluster'].value_counts())
```

Penjelasan :
eps: radius maksimum titik tetangga (300 meter)
min_samples: minimal titik untuk membentuk cluster (5)
metric='haversine' untuk menghitung jarak di permukaan bumi
Hasil cluster disimpan di kolom baru 'cluster'

6. Visualisasi hasil clustering

```
import matplotlib.pyplot as plt

plt.figure(figsize=(8,6))
plt.scatter(
    df['lon'],
    df['lat'],
    c=df['cluster'],
    cmap='tab20',
    s=30
)
plt.xlabel("Longitude")
plt.ylabel("Latitude")
plt.title("DBSCAN Clustering on GPS Coordinates")
plt.show()
```

Penjelasan :
Scatter plot menampilkan titik-titik GPS.
Setiap cluster memiliki warna berbeda.
Titik dengan label -1 dianggap noise (tidak termasuk cluster).

2. Interpretasi Hasil Clustering

- Jumlah cluster terbentuk: sesuai hasil df['cluster'].value_counts(), misal 3 cluster utama.
- Jumlah noise: titik dengan label -1 menunjukkan lokasi yang berjauhan dari cluster utama.
- Pola data:
    - Cluster menggambarkan area padat titik GPS.
    - Titik noise biasanya berada di luar area padat atau di lokasi terpencil.
    - Visualisasi menunjukkan distribusi geografis cluster dan lokasi noise.

3. Kesimpulan

- DBSCAN efektif untuk mengelompokkan data GPS berdasarkan kepadatan.
- Titik yang jauh dari cluster diidentifikasi sebagai noise, membantu memahami distribusi data secara lebih jelas.
- Pemilihan parameter eps dan min_samples sangat mempengaruhi jumlah cluster dan identifikasi noise.
- Praktikum ini memberikan pemahaman tentang penggunaan DBSCAN dalam analisis data geografis tanpa perlu mengetahui jumlah cluster sebelumnya.