Tugas 11: Analisis Klasterisasi Data Bencana Global Menggunakan Algoritma K-Means dan DBSCAN Berbasis Koordinat GPS

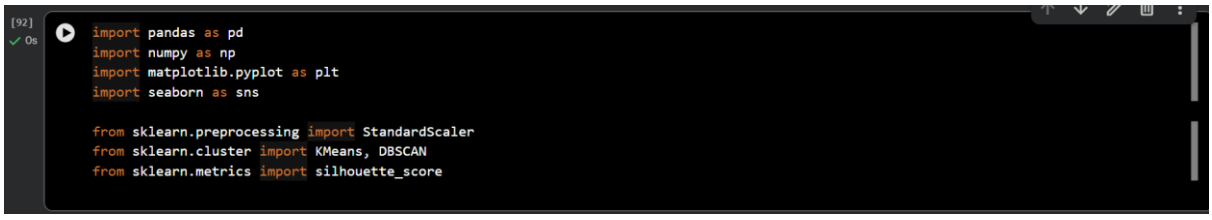
Sulthan Nabil Al Hakim - 0110224234

[1] Teknik Informatika, STT Terpadu Nurul Fikri, Depok

*E-mail: sultannabilalhakim@gmail.com

Abstract. Penelitian ini bertujuan untuk menganalisis pola persebaran bencana global menggunakan metode unsupervised learning berbasis data geospasial. Dataset yang digunakan adalah Global Disaster Response 2018–2024, yang berisi data lokasi bencana dalam bentuk koordinat GPS (latitude dan longitude). Dua algoritma klasterisasi, yaitu K-Means dan DBSCAN, diterapkan untuk mengidentifikasi pola densitas dan pembentukan klaster berdasarkan kedekatan geografis.

1. Import Library

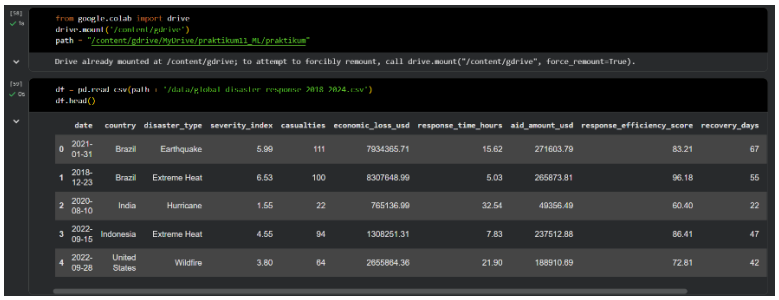


Cell ini memuat (import) semua pustaka (library) yang dibutuhkan.

- pandas: Untuk manipulasi dan analisis data (membuat tabel/DataFrame).
- numpy: Untuk operasi matematika dan array numerik.
- matplotlib & seaborn: Untuk membuat visualisasi data (grafik/plot).
- sklearn.preprocessing.StandardScaler: Untuk melakukan standarisasi data (scaling) agar rentang nilai data seragam.
- sklearn.cluster: Mengimpor algoritma clustering (KMeans, DBSCAN).

- ❖ sklearn.metrics.silhouette_score: Untuk mengevaluasi seberapa bagus cluster yang terbentuk.

2. Menghubungkan ke Google Drive



```
from google.colab import drive
drive.mount('/content/gdrive')
path = "/content/gdrive/MyDrive/praktikum11_ML/praktikum"
```

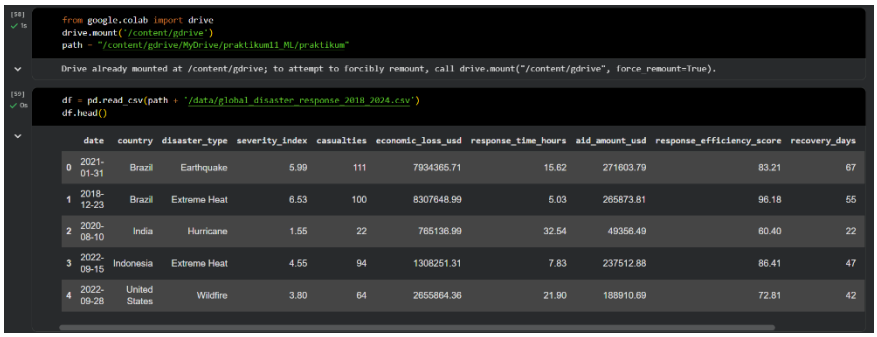
```
Drive already mounted at /content/gdrive; to attempt to forcibly remount, call drive.mount("/content/gdrive", force_remount=True).
```

```
df = pd.read_csv(path + '/data/global_disaster_response_2018_2024.csv')
df.head()
```

	date	country	disaster_type	severity_index	casualties	economic_loss_usd	response_time_hours	aid_amount_usd	response_efficiency_score	recovery_days
0	2021-01-31	Brazil	Earthquake	5.99	111	7934365.71	15.62	271603.79	83.21	67
1	2018-12-23	Brazil	Extreme Heat	6.53	100	8307648.99	5.03	265873.81	96.18	55
2	2020-08-10	India	Hurricane	1.55	22	765136.99	32.54	49356.49	60.40	22
3	2022-09-15	Indonesia	Extreme Heat	4.55	94	1308251.31	7.83	237512.88	86.41	47
4	2022-09-28	United States	Wildfire	3.80	64	2655864.36	21.90	188910.69	72.81	42

Kode ini menghubungkan Google Colab dengan Google Drive Anda. Ini diperlukan agar notebook bisa membaca file dataset (csv) yang tersimpan di folder Drive Anda (/praktikum11_ML/praktikum).

3. Membaca Dataset



```
from google.colab import drive
drive.mount('/content/gdrive')
path = "/content/gdrive/MyDrive/praktikum11_ML/praktikum"
```

```
Drive already mounted at /content/gdrive; to attempt to forcibly remount, call drive.mount("/content/gdrive", force_remount=True).
```

```
df = pd.read_csv(path + '/data/global_disaster_response_2018_2024.csv')
df.head()
```

	date	country	disaster_type	severity_index	casualties	economic_loss_usd	response_time_hours	aid_amount_usd	response_efficiency_score	recovery_days
0	2021-01-31	Brazil	Earthquake	5.99	111	7934365.71	15.62	271603.79	83.21	67
1	2018-12-23	Brazil	Extreme Heat	6.53	100	8307648.99	5.03	265873.81	96.18	55
2	2020-08-10	India	Hurricane	1.55	22	765136.99	32.54	49356.49	60.40	22
3	2022-09-15	Indonesia	Extreme Heat	4.55	94	1308251.31	7.83	237512.88	86.41	47
4	2022-09-28	United States	Wildfire	3.80	64	2655864.36	21.90	188910.69	72.81	42

- ❖ pd.read_csv(...): Membaca file CSV data bencana global ke dalam variabel df (DataFrame).
- ❖ df.head(): Menampilkan 5 baris pertama data untuk melihat gambaran awal isinya (seperti kolom date, country, disaster_type, severity_index, lokasi latitude/longitude, dll).

4. Cek Informasi Data

```
[60]
✓ 0s
df.info()

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 50000 entries, 0 to 49999
Data columns (total 12 columns):
 #   Column                    Non-Null Count  Dtype
---  ------                    --------------  -----
 0   date                      50000 non-null  object
 1   country                   50000 non-null  object
 2   disaster_type             50000 non-null  object
 3   severity_index            50000 non-null  float64
 4   casualties                50000 non-null  int64
 5   economic_loss_usd         50000 non-null  float64
 6   response_time_hours       50000 non-null  float64
 7   aid_amount_usd            50000 non-null  float64
 8   response_efficiency_score 50000 non-null  float64
 9   recovery_days             50000 non-null  int64
 10  latitude                  50000 non-null  float64
 11  longitude                 50000 non-null  float64
dtypes: float64(7), int64(2), object(3)
memory usage: 4.6+ MB
```

Penjelasan: Memberikan ringkasan teknis tentang dataset, termasuk:

- ❖ Jumlah baris (50,000 entries).
- ❖ Nama kolom dan tipe datanya (float, int, object/text).
- ❖ Jumlah data non-null (untuk melihat apakah ada data kosong).

5. Cek Nilai Kosong (Missing Values)

```
[61]
✓ 0s
df.isnull().sum()
```

	0
date	0
country	0
disaster_type	0
severity_index	0
casualties	0
economic_loss_usd	0
response_time_hours	0
aid_amount_usd	0
response_efficiency_score	0
recovery_days	0
latitude	0
longitude	0

dtype: int64

Menghitung jumlah nilai NaN (kosong) di setiap kolom.

- ❖ Hasil: Semua kolom bernilai 0, artinya data ini bersih dan tidak ada yang kosong.

6. Cek Data Duplikat

```
[62]
✓ 0s
df.duplicated().sum()

np.int64(0)
```

Mengecek apakah ada baris data yang sama persis (duplikat).

- ❖ Hasil: 0, artinya tidak ada data ganda.

7. Statistik Deskriptif

```
[63] ✓ 0s
df.describe()
```

	severity_index	casualties	economic_loss_usd	response_time_hours	aid_amount_usd	response_efficiency_score	recovery_days	lat
count	50000.000000	50000.000000	5.000000e+04	50000.000000	5.000000e+04	50000.000000	50000.000000	50000.0
mean	5.015769	100.591140	5.068593e+06	12.183027	2.500003e+05	87.574025	49.682560	0.2
std	1.942843	65.052064	3.268541e+06	9.259081	1.432275e+05	10.188961	20.098944	34.7
min	1.000000	0.000000	5.273900e+02	1.000000	1.660000e+01	29.750000	2.000000	-59.9
25%	3.660000	51.000000	2.585513e+06	6.270000	1.429663e+05	83.060000	36.000000	-29.8
50%	4.990000	91.000000	4.548351e+06	10.510000	2.305365e+05	89.180000	49.000000	0.2
75%	6.340000	138.000000	6.950615e+06	15.450000	3.352259e+05	94.700000	63.000000	30.4
max	10.000000	524.000000	2.445624e+07	63.100000	1.126465e+06	100.000000	112.000000	59.9

Menampilkan statistik dasar untuk kolom numerik (rata-rata/mean, standar deviasi/std, nilai minimum/min, nilai maksimum/max, kuartil). Ini membantu memahami persebaran data, misal rata-rata severity_index adalah sekitar 5.01.

8. Cek Jenis Bencana

```
[64] ✓ 0s
df["disaster_type"].unique()

array(['Earthquake', 'Extreme Heat', 'Hurricane', 'Wildfire', 'Flood',
       'Storm Surge', 'Drought', 'Tornado', 'Landslide',
       'Volcanic Eruption'], dtype=object)
```

Menampilkan daftar unik jenis-jenis bencana yang ada di kolom disaster_type (contoh: Earthquake, Flood, Wildfire, dll).

9. Hitung Jumlah per Jenis Bencana

```
[65] ✓ 0s
df["disaster_type"].value_counts()
```

disaster_type	count
Landslide	5130
Earthquake	5068
Flood	5039
Hurricane	5002
Extreme Heat	5001
Storm Surge	4988
Volcanic Eruption	4983
Wildfire	4954
Tornado	4939
Drought	4896

dtype: int64

Menghitung berapa kali setiap jenis bencana muncul dalam dataset.

- ❖ Hasil: Landslide (Longsor) adalah yang terbanyak (5130 data), diikuti oleh Earthquake (Gempa), dst.

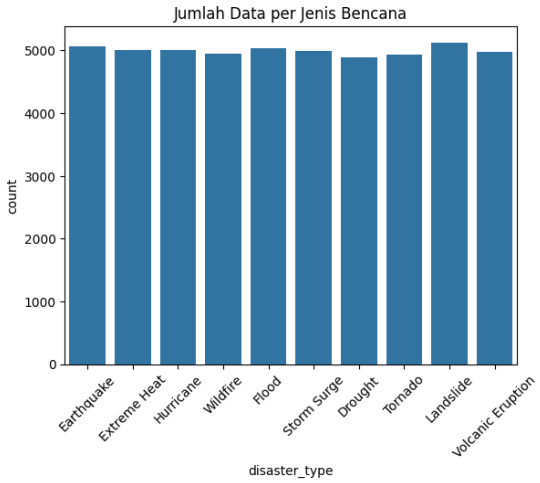
10. Encoding Data Kategori

```
[66]
✓
    from sklearn.preprocessing import LabelEncoder

    le = LabelEncoder()
    df["disaster_type_encoded"] = le.fit_transform(df["disaster_type"])
```

Mengubah kolom teks disaster_type menjadi angka (numeric) agar bisa diproses oleh komputer jika diperlukan nanti. Contoh: 'Earthquake' mungkin menjadi angka 1, 'Flood' menjadi angka 2. Hasilnya disimpan di kolom baru disaster_type_encoded.

11. Visualisasi Jumlah Bencana

```
[67]
✓ 0s
    sns.countplot(x="disaster_type", data=df)
    plt.xticks(rotation=45)
    plt.title("Jumlah Data per Jenis Bencana")
    plt.show()
```



Membuat grafik batang (bar chart) yang memvisualisasikan jumlah kejadian setiap jenis bencana. rotation=45 memiringkan label sumbu-x agar tulisan jenis bencana tidak bertumpuk.

12 & 13. Inspeksi Data Lokasi

```
[68]
✓ 0s
X[10:16]

array([[ 0.4617696 , -0.46805501],
       [ 0.5808456 ,  0.69658113],
       [ 0.11121608, -0.88284787],
       [ 1.68334146,  0.48292829],
       [ 1.17697359,  0.95294623],
       [-0.34785512, -1.50017333]])
```

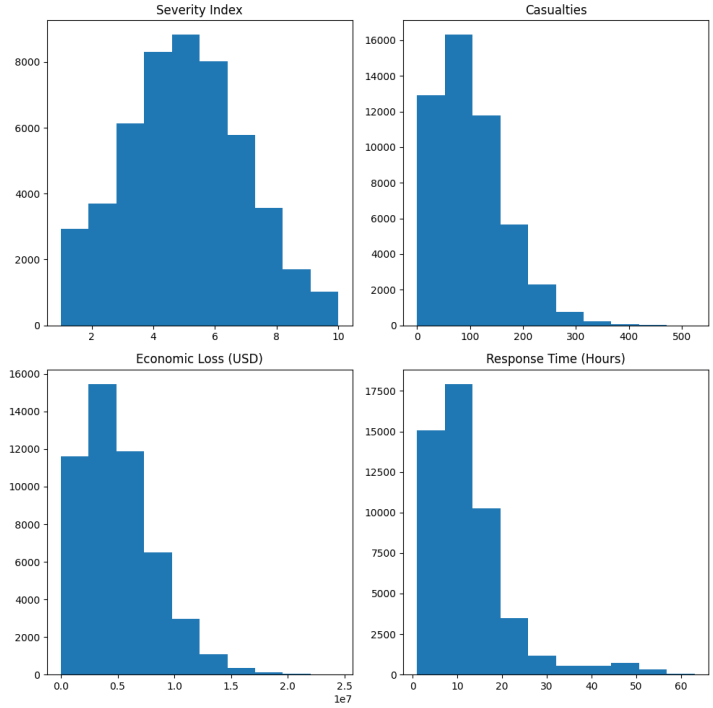
```
[85]
✓ 0s
X.head()
```

	latitude	longitude
0	-30.613	-122.557
1	10.859	-159.194
2	0.643	-160.978
3	-33.547	30.350
4	-19.170	-117.137

Menampilkan sebagian data dari kolom latitude dan longitude (koordinat lokasi). Ini adalah fitur utama yang akan digunakan untuk clustering geografis nanti.

14. Histogram Distribusi Data

```
[70]
✓ 0s
fig, axes = plt.subplots(2, 2, figsize=(10, 10))

# Histogram 1: Severity Index
axes[0, 0].set_title("Severity Index")
axes[0, 0].hist(df['severity_index'], bins=10)

# Histogram 2: Casualties
axes[0, 1].set_title("Casualties")
axes[0, 1].hist(df['casualties'], bins=10)

# Histogram 3: Economic Loss (USD)
axes[1, 0].set_title("Economic Loss (USD)")
axes[1, 0].hist(df['economic_loss_usd'], bins=10)

# Histogram 4: Response Time (Hours)
axes[1, 1].set_title("Response Time (Hours)")
axes[1, 1].hist(df['response_time_hours'], bins=10)

plt.tight_layout()
plt.show()
```

- 

Membuat 4 histogram sekaligus untuk melihat sebaran data pada kolom:

1. Severity Index: Indeks keparahan.
2. Casualties: Jumlah korban.
3. Economic Loss: Kerugian ekonomi.
4. Response Time: Waktu tanggap.

15, 16, 17. Persiapan Data (Scaling) untuk Clustering

- [71] ✓ 0s

```
from sklearn.preprocessing import StandardScaler

scaler = StandardScaler()

# Pilih fitur untuk clustering
X = df[['latitude', 'longitude']]

# Scaling
X = scaler.fit_transform(X)

X[:5]
```

```
array([[-0.88736519, -1.24667469],
       [ 0.30575387, -1.61908055],
       [ 0.01184703, -1.63721446],
       [-0.97177422,  0.3075864 ],
       [-0.55815846, -1.19158176]])
```

- [72] ✓ 0s

```
# Standarisasi untuk GPS clustering
scaler = StandardScaler()

x = df[['latitude', 'longitude']]
x = scaler.fit_transform(x)
x
```

```
array([[-0.88736519, -1.24667469],
       [ 0.30575387, -1.61908055],
       [ 0.01184703, -1.63721446],
       ...,
       [ 0.43113033, -0.28406291],
       [-1.27255691,  0.01863311],
       [ 0.36976549, -0.82474617]])
```

- [73] ✓

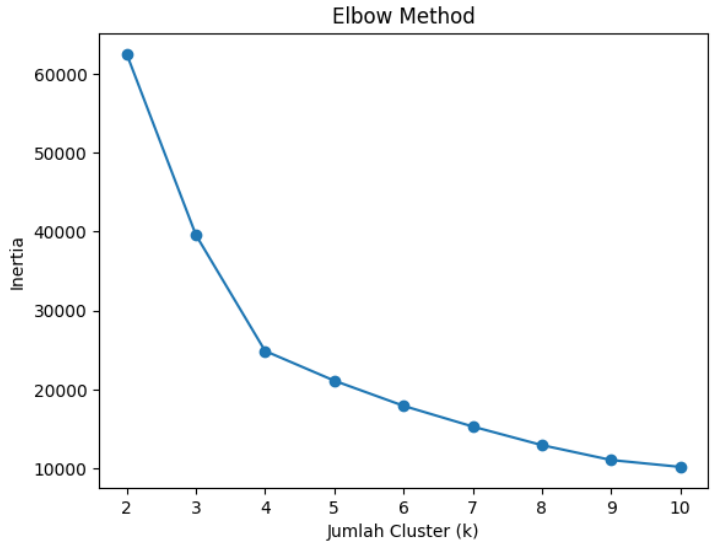
```
X = df[['latitude', 'longitude']].copy()

# handle missing values
X = X.dropna()

# scaling
scaler = StandardScaler()
X_scaled = scaler.fit_transform(X)
```

Penjelasan:

- Kita mengambil kolom latitude dan longitude sebagai fitur untuk clustering (kita ingin mengelompokkan bencana berdasarkan lokasi).
- StandardScaler: Mengubah nilai koordinat agar memiliki skala yang sama (mean=0, std=1). Ini sangat penting untuk K-Means karena algoritma ini menghitung jarak; jika skalanya berbeda jauh, hasil cluster bisa bias.

18. Mencari Jumlah Cluster Optimal (Elbow Method)

```
[74]
✓ 0s
    inertia = []
    K = range(2, 11)

    for k in K:
        kmeans = KMeans(n_clusters=k, random_state=42)
        kmeans.fit(X_scaled)
        inertia.append(kmeans.inertia_)

    plt.plot(K, inertia, marker='o')
    plt.xlabel('Jumlah Cluster (k)')
    plt.ylabel('Inertia')
    plt.title('Elbow Method')
    plt.show()
```



- Penjelasan:

- Melakukan looping untuk mencoba jumlah cluster (k) dari 2 sampai 10.
- Menghitung inertia (jumlah kuadrat jarak titik ke pusat clusternya) untuk setiap k.
- Membuat grafik garis. Titik di mana penurunan inersia mulai melambat (membentuk siku/elbow) adalah jumlah cluster yang optimal. Dari grafik, sepertinya k=4 dipilih.

19, 20, 21. Penerapan K-Means Clustering

```
[75] ✓ 0s
kmeans = KMeans(n_clusters=4, random_state=42)
clusters = kmeans.fit_predict(X_scaled)

df['Cluster_KMeans'] = clusters
df.head()
```

	date	country	disaster_type	severity_index	casualties	economic_loss_usd	response_time_hours	aid_amount_usd	response_efficiency_score
	2021-01-31	Brazil	Earthquake	5.99	111	7934365.71	15.62	271603.79	83.21
	2018-12-23	Brazil	Extreme Heat	6.53	100	8307648.99	5.03	265873.81	96.18
	2020-08-10	India	Hurricane	1.55	22	765136.99	32.54	49356.49	60.40
	2022-09-15	Indonesia	Extreme Heat	4.55	94	1308251.31	7.83	237512.88	86.41
	2022-09-28	United States	Wildfire	3.80	64	2655864.36	21.90	188910.69	72.81

```
[76] ✓ 0s
df["Cluster"] = kmeans.fit_predict(X_scaled)
print("Cluster ditemukan:", df["Cluster"].unique())

Cluster ditemukan: [2 1 0 3]
```

```
[77] ✓ 0s
# Latih model (menentukan centroid + proses clustering)
kmeans.fit(X)

KMeans
KMeans(n_clusters=4, random_state=42)
```

- Penjelasan:
  - ❖ Membuat model K-Means dengan 4 cluster (n_clusters=4).
  - ❖ fit_predict: Melatih model pada data lokasi yang sudah di-scaling dan langsung memprediksi setiap data masuk ke cluster mana (0, 1, 2, atau 3).
  - ❖ Hasil prediksi disimpan ke kolom baru Cluster_KMeans (dan juga Cluster di cell 111).

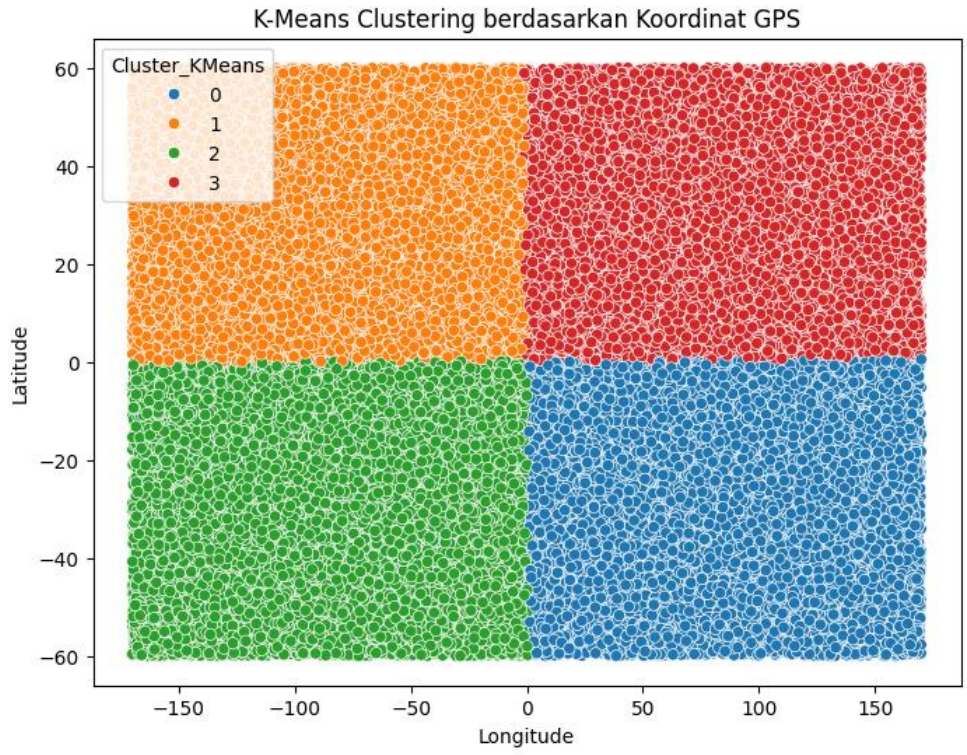
22. Evaluasi Model (Silhouette Score)

```
[78] ✓ 26s
from sklearn.metrics import silhouette_score

score = silhouette_score(X_scaled, clusters)
print("Silhouette Score:", score)

Silhouette Score: 0.40847166786670236
```

- Penjelasan: Menghitung skor Silhouette. Skor ini berkisar antara -1 hingga 1.
  - o Nilai yang mendekati 1 berarti cluster terpisah dengan sangat baik dan padat.
  - o Nilai sekitar 0 berarti cluster tumpang tindih.
  - o Hasil: 0.408 menunjukkan clustering yang cukup baik (cluster terdefinisi dengan lumayan jelas).

23. Visualisasi Hasil Clustering (Scatter Plot)

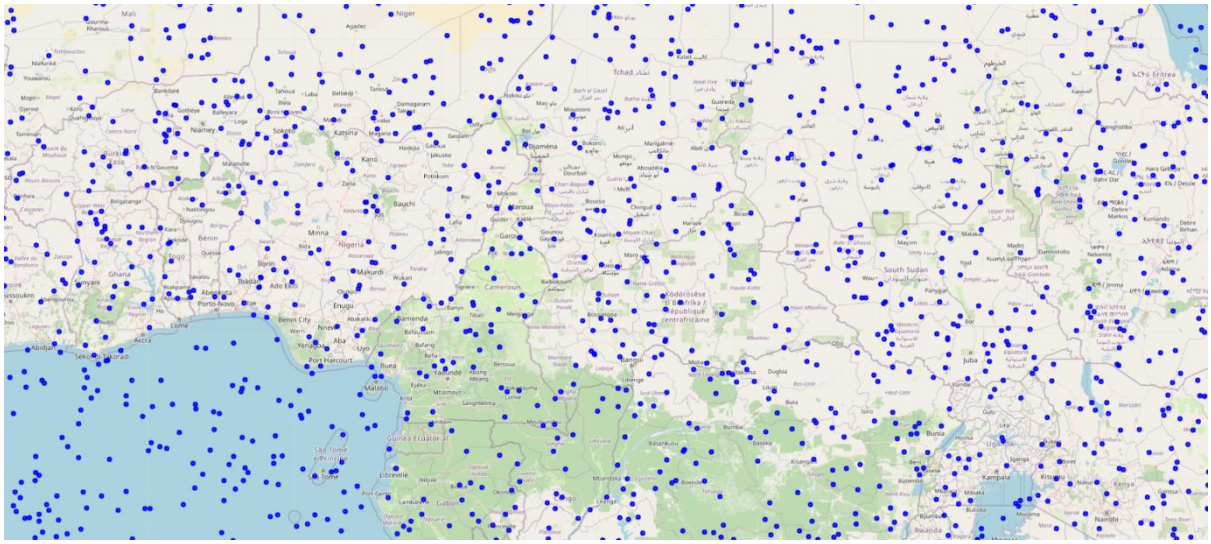
```
[80]
✓ 3s
plt.figure(figsize=(8, 6))
sns.scatterplot(x=df['longitude'], y=df['latitude'], hue=df['Cluster_KMeans'], palette='tab10')
plt.title('K-Means Clustering berdasarkan Koordinat GPS')
plt.xlabel('Longitude')
plt.ylabel('Latitude')
plt.show()
```



- Penjelasan:
  - ❖ Membuat plot titik (scatter plot) dengan sumbu X longitude dan Y latitude.
  - ❖ hue='Cluster_KMeans': Mewarnai titik-titik berdasarkan hasil cluster (kelompok) yang telah dibuat oleh K-Means.
  - ❖ Ini menunjukkan peta persebaran bencana yang sudah dikelompokkan berdasarkan wilayah geografisnya.

24. Visualisasi Peta Interaktif (Folium)

```
[91]
✓ 1m
    !pip install folium

    import folium

    # Titik tengah dunia
    m = folium.Map(location=[0,0], zoom_start=2)

    for _, row in df.iterrows():
        folium.CircleMarker(
            location=[row['latitude'], row['longitude']],
            radius=3,
            color='blue',
            fill=True,
            fill_opacity=0.6
        ).add_to(m)

    m
```



- Penjelasan:
  - ❖ Menggunakan library folium untuk membuat peta dunia interaktif.
  - ❖ Melakukan looping pada setiap baris data (df.iterrows()) dan menambahkan titik (CircleMarker) di peta sesuai koordinat latitude dan longitude bencana tersebut.
  - ❖ Peta ini memungkinkan Anda untuk zoom in/out dan melihat lokasi bencana secara detail di atas peta asli.

16. Kesimpulan Hasil Implementasi Algoritma

Metode: Algoritma K-Means diterapkan pada fitur lokasi (latitude & longitude).
Hasil Optimal: Terbentuk 4 Cluster (berdasarkan *Elbow Method*).
Kualitas Model: Silhouette Score 0.408, menunjukkan pengelompokan yang cukup baik dan terdefinisi, meskipun wajar jika ada kedekatan antar wilayah.
Implikasi: Algoritma berhasil membagi peta bencana global menjadi 4 zona operasional utama. Ini sangat berguna untuk menentukan lokasi strategis pusat logistik/bantuan di setiap wilayah.
Singkatnya: K-Means efektif digunakan untuk segmentasi wilayah bencana secara otomatis guna mendukung manajemen logistik.

LINK GITHUB : https://github.com/Mrhankim/praktikum11_ML.git

Referensi:

Munir, S., Seminar, K. B., Sudradjat, Sukoco, H., & Buono, A. (2022). The Use of Random Forest Regression for Estimating Leaf Nitrogen Content of Oil Palm Based on Sentinel 1-A Imagery. Information, 14(1), 10. https://doi.org/10.3390/info14010010

Seminar, K. B., Imantho, H., Sudradjat, Yahya, S., Munir, S., Kaliana, I., Mei Haryadi, F., Noor Baroroh, A., Supriyanto, Handoyo, G. C., Kurnia Wijayanto, A., Ijang Wahyudin, C., Liyantono, Budiman, R., Bakir Pasaman, A., Rusiawan, D., & Sulastri. (2024). PreciPalm: An Intelligent System for Calculating Macronutrient Status and Fertilizer Recommendations for Oil Palm on Mineral Soils Based on a Precision Agriculture Approach. Scientific World Journal, 2024(1). https://doi.org/10.1155/2024/1788726