| | | | |
|---|---|---|---|
| NIM | : 2311523007 | Tanggal | : Senin, 19 Mei 2025 |
| Nama | : Vania Zhafira Zahra | Asisten | : Miftahul Khaira Ghina Anfasha |
| Mata Kuliah | : Praktikum Data Mining | | |
| Modul | : 09 | | |
| Kelas | : A | | |

# Intruksi “DBSCAN”

Algoritma DBSCAN adalah sebuah algoritma clustering yang dikembangkan berdasarkan tingkat kerapatan data (density-based). Dimana algoritma ini menumbuhkan daerah yang memiliki kerapatan tinggi menjadi cluster-cluster, dan menemukan cluster-cluster tersebut pada bentuk bebas dalam sebuah ruang database dengan memanfaatkan noise.

Noise dalam metode ini digunakan untuk mewakili daerah yang kurang padat yang digunakan untuk memisahkan antara cluster satu dengan cluster lainnya, pada objek dalam ruang data. Dalam menentukan cluster, DBSCAN menggunakan sebuah kumpulan maksimal dari kerapatan titik-titik terhubung dengan menggunakan parameter Eps dan MinPts. Parameter Eps digunakan untuk menentukan radius (jarak maksimal) titik-titik anggota cluster dari pusat cluster. Dan parameter MinPts digunakan untuk memberikan batasan jumlah titik-titik yang menjadi anggota cluster dalam radius Eps tersebut.

1. Kelebihan algoritma DBSCAN diantaranya:

    a) Dapat memperoleh cluster dengan akurat dari bentuk data yang tidak struktur jika dibandingkan dengan partitional clustering.
    b) Mampu menangani outliner/noise.
    c) Algoritma DBSCAN sangat baik untuk data dengan jumlah yang sangat besar.

2. Kekurangan algoritma DBSCAN diantaranya:

    a) Pada dataset berdimensi tinggi hasilnya kurang maksimal.
    b) Memiliki permasalahan pada saat identifikasi cluster dari kepadatan yang bervariasi.
    c) Tidak dapat membentuk fungsi kepadatan yang sesungguhnya, tetapi lebih kearah poin – poin kepadatan yang saling berhubungan dan membentuk graf

## INSTRUKSI

1. Import Library yang akan digunakan seperti pandas, seabron, matplotlib dll

```
# Langkah 1 : Import Library
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
from sklearn.preprocessing import StandardScaler
from sklearn.cluster import DBSCAN
from sklearn.metrics import silhouette_score
from sklearn.decomposition import PCA
import numpy as np
import matplotlib.cm as cm
```

2. Import dataset Country-data.csv dan data-dictionary.csv sebagai keterangan dari feature dataset Country-data.csv.

```
# Langkah 2 : Import Dataset Country-data.csv dan data-dictionary.csv
country_df = pd.read_csv('Country-data.csv')
dictionary_df = pd.read_csv('data-dictionary.csv')

# Tampilkan beberapa baris awal dari masing-masing dataset
print("Data Country:")
print(country_df.head())

print("\nData Dictionary:")
print(dictionary_df.head(10))
```

Pada hasil run ini, menampilkan bawah data country merupakan dataset yang akan di proses, dan untuk data dictionary merupakan penjelasan dari fitur-fitur yang ada pada data country

3. Pembersihan Data

```
# Langkah 3: Pembersihan data (Deteksi Nilai null)
print("\nCek nilai null:")
print(country_df.isnull().sum())
```

```
Cek nilai null:
country       0
child_mort    0
exports       0
health        0
imports       0
income        0
inflation     0
life_expec    0
total_fer     0
gdpp          0
dtype: int64
```

Pada langkah 3 ini memeriksa apakah data country memiliki nilai yang null atau kosong, hasil dari penampilan ini bahwa data nya tidak ada yang null

```
# Langkah 4: Pembersihan data (Duplikasi data)
print("\nCek duplikat:")
print(country_df.duplicated().sum())
```

```
Cek duplikat:
0
```

Pada langkah 4 ini, menganalisis apakah ada data yang terduplikasi, hasil dari pengecheckan bahwa tidak ada yang duplikasi.

[103]:

```
# Langkah 5 : Pembersihan data (Menghapus data yang null atau duplikat)
country_df = country_df.dropna()
country_df = country_df.drop_duplicates()
```

Melakukan penghapusan data yang null atau duplikat dengan menggunakan dropna(nilai null) dan drop_duplicates(data yang duplikasi)

4. Drop kolom ‘country’, simpan dalam variabel baru.

[104]:

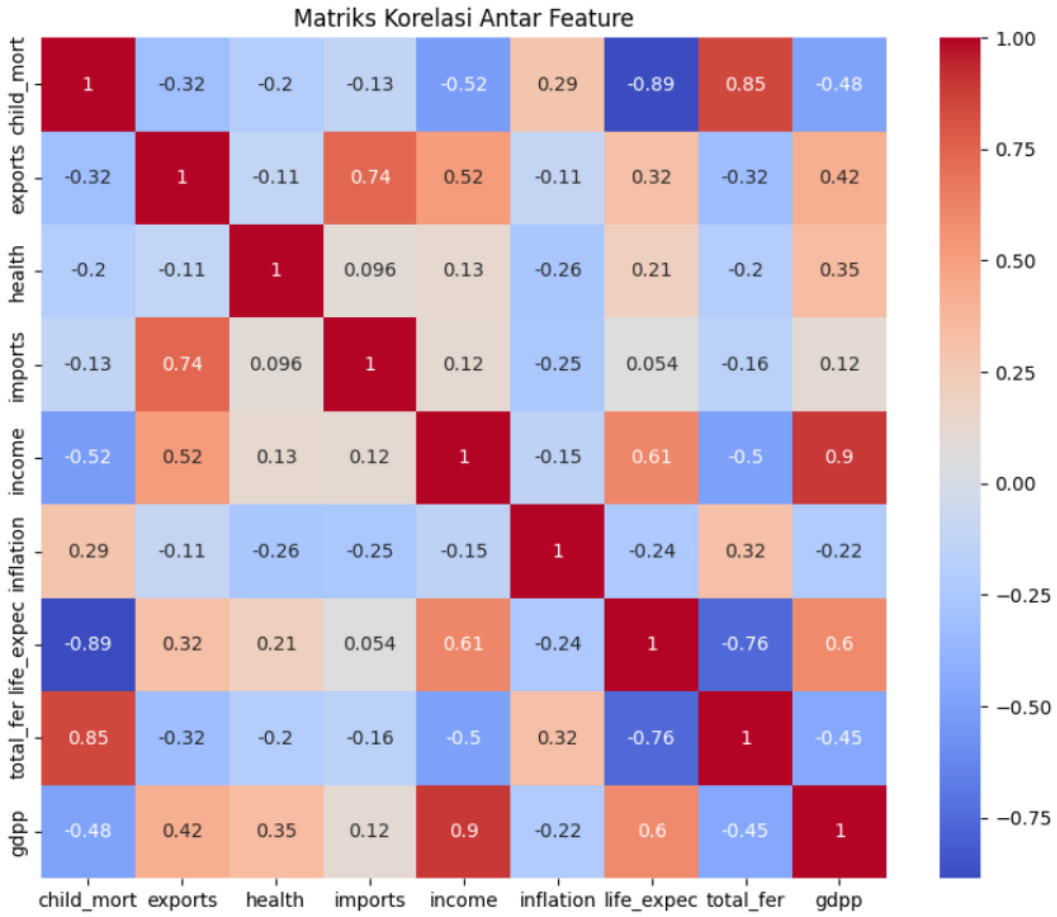
```
# Langkah 6 : Drop kolom 'country' dan simpan ke variabel baru
country_names = country_df['country']  # Simpan nama negara jika perlu nanti
data_clean = country_df.drop(columns=['country'])
```

Pada drop kolom ini, data di simpan dengan variable baru yaitu data_clean

5. Tampilkan visualisasi hubungan setiap feature dengan matriks korelasi

[105]:

```
# Langkah 7 : Visualisasi matriks korelasi
plt.figure(figsize=(10, 8))
sns.heatmap(data_clean.corr(), annot=True, cmap='coolwarm')
plt.title('Matriks Korelasi Antar Feature')
plt.show()
```




6. Lakukan pemilihan Fitur dan Scalling data untuk standarisasi nilai pada data

[106]:

```
# Langkah 8 : Pemilihan fitur yang relevan dan Scaling data (Standardisasi)
x = data_clean.iloc[:,[4,8]].values

scaler = StandardScaler()
data_scaled = scaler.fit_transform(x)

x.shape
```

[106]:

```
(167, 2)
```

Fitur yang diambil income dan gdpp, karena keduanya mencerminkan kesejahteraan ekonomi secara jelas, memiliki korelasi yang relevan untuk pengelompokan, mudah diinterpretasi, dan memungkinkan DBSCAN mendeteksi cluster serta outlier secara efektif tanpa memerlukan jumlah cluster di awal. Lalu setelah melakukan pemilihan fitur maka akan melakukan scaled data, lalu mengecheck jumlah kolom baris data, dan data yang di scaled akan disimpan menjadi data_scaled.

7. Melakukan analisis eps yang terbaik untuk dataset nya

[107]:

```
# Langkah 9 : MElakukan analisis eps terbaik untuk DBSCAN
best_eps = 0
best_score = -1
best_labels = None

for eps in np.arange(0.1, 5.0, 0.1):
    model = DBSCAN(eps=eps, min_samples=4)
    labels = model.fit_predict(data_scaled)
    n_clusters = len(set(labels)) - (1 if -1 in labels else 0)

    if n_clusters > 1:
        score = silhouette_score(data_scaled, labels)
        if score > best_score:
            best_score = score
            best_eps = eps
            best_labels = labels

print(f"Best eps: {best_eps}")
print(f"Best silhouette score: {best_score:.3f}")
```

```
Best eps: 0.2
Best silhouette score: 0.490
```

Dengan melakukan perhitungan best eps dan best silhoutte yang akan didapat, maka hasil yang didapat yaitu eps 0.2 dan sillhoutte nya 0.490, jika sillhoute yang terakhir berbeda dengan yang di analisis ini karena minimum sample nya yang di analsis dan digunakan berbeda.

8. Membuat model DBSCAN

[108]:

```
# Langkah 10 : Membuat model DBSCAN
dbscan_model = DBSCAN(eps=0.2, min_samples=3, metric='euclidean')
dbscan_model.fit(data_scaled)
```

[108]:

```
DBSCAN
DBSCAN(eps=0.2, min_samples=3)
```

Membuat mode DBSCAN dengan menggunakan eps, min sample. Dengam mengambil data data_scaled untuk pembuatan model nya.

9. Tampilkan label cluster dan simpan dalam variabel baru

```
# Langkah 11  : Tampilkan label cluster
model = dbscan_model.fit(data_scaled)
label  = model.labels_
label
```

[109]:

```
array([ 0,  0,  0,  0,  0,  0,  0,  1,  1,  0, -1, -1,  0,  0,  0,  1,  0,
        0,  0,  0,  0,  0,  0, -1,  0,  0,  0,  0,  0,  1,  0,  0,  0,  0,
        0,  0,  0,  0,  0,  0,  0,  0,  2,  3, -1,  0,  0,  0,  0, -1,  0,
        0,  0,  1,  1,  0,  0,  0,  1,  0,  3,  0,  0,  0,  0,  0,  0,  0,
        1,  0,  0,  0,  0,  1,  2,  1,  0, -1,  0,  0,  0,  0, -1,  0,  0,
        0,  0,  0,  0, -1,  0, -1,  0,  0,  0,  0,  0,  0,  3,  0,  0,  0,
        0,  0,  0,  0,  0,  0,  0,  0,  1,  2,  0,  0, -1, -1,  0,  0,  0,
        0,  0,  0,  3, -1,  0,  0,  0,  0, -1,  0,  0,  0,  0, -1,  0,  3,
        0,  0,  3,  2,  0,  0,  0,  0,  1, -1,  0,  0,  0,  0,  0,  0,  0,
        0,  0,  0,  0, -1,  1,  1,  0,  0,  0,  0,  0,  0,  0])
```

Model yang dibuat akan di buat dalam bentuk label cluster, berbentuk array.

10. Perhitungan untuk jumlah cluster

[110]:

```
# Langkah 12 : Perhitungan untuk jumlah cluster
sample_cores=np.zeros_like(label,dtype=bool)

sample_cores[dbscan_model.core_sample_indices_]=True

n_clusters=len(set(label))- (1 if -1 in label else 0)
print('No of clusters:',n_clusters)
```

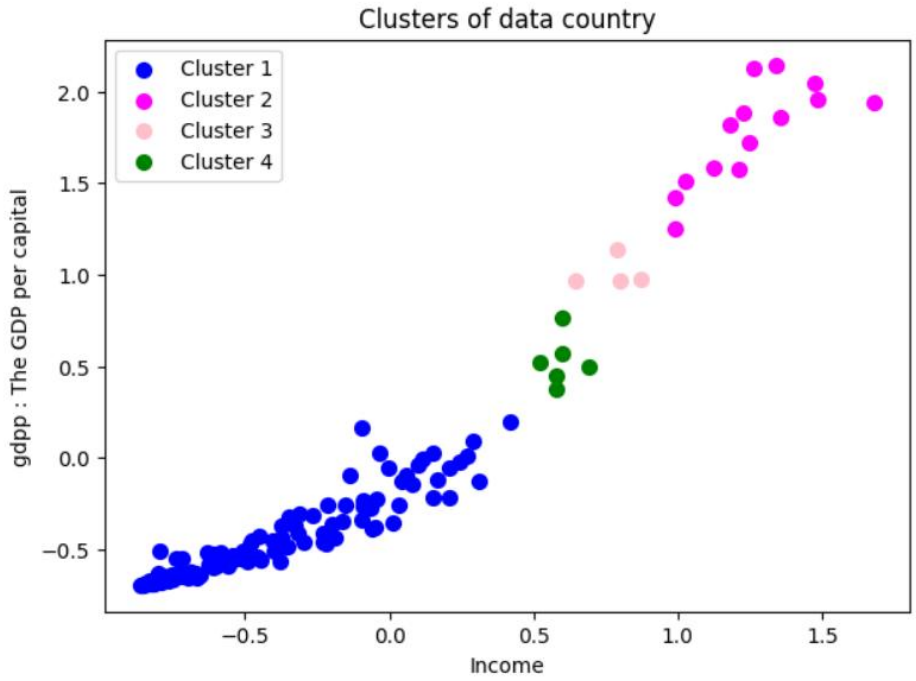
```
No of clusters: 4
```

Menghitung jumlah cluster yang didapat dari pembuatan model DBSCAN, hasil nya terdapat 4 cluster yang terbentuk

11. Visualisasikan cluster yang terbentuk

```
# Langkah 13 : Visualisasikan cluster dengan plt

y_means = dbscan_model.fit_predict(data_scaled)
plt.figure(figsize=(7,5))
plt.scatter(data_scaled[y_means == 0, 0], data_scaled[y_means == 0, 1], s = 50, c = 'blue', label='Cluster 1')
plt.scatter(data_scaled[y_means == 1, 0], data_scaled[y_means == 1, 1], s = 50, c = 'magenta', label='Cluster 2')
plt.scatter(data_scaled[y_means == 2, 0], data_scaled[y_means == 2, 1], s = 50, c = 'pink', label='Cluster 3')
plt.scatter(data_scaled[y_means == 3, 0], data_scaled[y_means == 3, 1], s = 50, c = 'green', label='Cluster 4')
plt.xlabel('Income')
plt.ylabel('gdpp : The GDP per capital')
plt.title('Clusters of data country')
plt.legend()
plt.show()
```




12. Tampilkan jumlah data tiap clusternya

```
# Langkah 14: Tampilkan jumlah tiap cluster
unique, counts = np.unique(label, return_counts=True)
cluster_count = dict(zip(unique, counts))
print("\nJumlah data per cluster:")
for cluster_id, count in cluster_count.items():
    print(f"Cluster {cluster_id}: {count} data")
```

```
Jumlah data per cluster:
Cluster -1: 16 data
Cluster 0: 127 data
Cluster 1: 14 data
Cluster 2: 4 data
Cluster 3: 6 data
```

Pada hasil ini menampilkan ada cluster -1 itu itu merupakan cluster yang outlier, namun pada visualisasi nya tidak dimasukkan, yang dimasukkan dalam hanya cluster 1-4

13. Evaluasi dengan silhoutte score

```python
# Langkah 15: Evaluasi dengan silhouette score (hanya jika cluster > 1)
if n_clusters > 1:
    score = silhouette_score(data_scaled, label)
    print(f"\nSilhouette Score: {score:.4f}")
else:
    print("\nTidak dapat menghitung Silhouette Score karena hanya satu cluster yang terbentuk
```

```
Silhouette Score: 0.5051
```

Karena cluster yang terbentuk lebih dari 1 maka dapat menghitung silhoutte score nya, dan hasil nya jika lebih dari 0 maka cluster nya bagus.

**KESIMPULAN**