Laporan

Case Based 2 Machine Learning

Kode Dosen : COK

Oleh :

Muhammad Ghiyaats Daffa (IF4411/1301204068)

Saya mengerjakan tugas ini dengan cara yang tidak melanggar aturan perkuliahan dan kode etik akademisi.

A. Pemilihan Data

Data yang saya dapat merupakan data tentang berbagai rekapan berbagai negara. Rekapan tersebut berisi data child_mort, exports, health, imports, income, inflation, life_expec, total_fer, dan gdpp.

Hal yang akan saya lakukan pertama kali adalah mengecek kualitas data yang saya dapatkan.

- Pertama, saya mengimport library yang diperlukan untuk tugas ini

```
import pandas as pd
import random as random
from sklearn import preprocessing
import matplotlib.pyplot as plt
from sklearn.decomposition import PCA
random.seed(1301204068)
```

Disini saya mengatur seed untuk random menggunakan NIM saya agar tiap kali run program, hasilnya tidak berubah ubah saat hyper parameter yang digunakan sama

- Selanjutnya, saya load dataset yang akan digunakan

```
Import Data

data = pd.read_csv("https://raw.githubusercontent.com/mughidaf/CaseBased2/main/Country-data.csv")
data
```

- Berikut adalah bentuk datanya

index	country	child_mort	exports	health	imports	income	inflation	life_expec	total_fer	gdpp
0	Afghanistan	90.2	10.0	7.58	44.9	1610	9.44	56.2	5.82	553
1	Albania	16.6	28.0	6.55	48.6	9930	4.49	76.3	1.65	4090
2	Algeria	27.3	38.4	4.17	31.4	12900	16.1	76.5	2.89	4460
3	Angola	119.0	62.3	2.85	42.9	5900	22.4	60.1	6.16	3530
4	Antigua and Barbuda	10.3	45.5	6.03	58.9	19100	1.44	76.8	2.13	12200
5	Argentina	14.5	18.9	8.1	16.0	18700	20.9	75.8	2.37	10300
6	Armenia	18.1	20.8	4.4	45.3	6700	7.77	73.3	1.69	3220
7	Australia	4.8	19.8	8.73	20.9	41400	1.16	82.0	1.93	51900
8	Austria	4.3	51.3	11.0	47.8	43200	0.873	80.5	1.44	46900
9	Azerbaijan	39.2	54.3	5.88	20.7	16000	13.8	69.1	1.92	5840
10	Bahamas	13.8	35.0	7.89	43.7	22900	-0.393	73.8	1.86	28000
11	Bahrain	8.6	69.5	4.97	50.9	41100	7.44	76.0	2.16	20700
12	Bangladesh	49.4	16.0	3.52	21.8	2440	7.14	70.4	2.33	758
13	Barbados	14.2	39.5	7.97	48.7	15300	0.321	76.7	1.78	16000
14	Belarus	5.5	51.4	5.61	64.5	16200	15.1	70.4	1.49	6030
15	Belgium	4.5	76.4	10.7	74.7	41100	1.88	80.0	1.86	44400
16	Belize	18.8	58.2	5.2	57.5	7880	1.14	71.4	2.71	4340
17	Benin	111.0	23.8	4.1	37.2	1820	0.885	61.8	5.36	758
18	Bhutan	42.7	42.5	5.2	70.7	6420	5.99	72.1	2.38	2180
19	Bolivia	46.6	41.2	4.84	34.3	5410	8.78	71.6	3.2	1980
20	Bosnia and Herzegovina	6.9	29.7	11.1	51.3	9720	1.4	76.8	1.31	4610
21	Botswana	52.5	43.6	8.3	51.3	13300	8.92	57.1	2.88	6350
22	Brazil	19.8	10.7	9.01	11.8	14500	8.41	74.2	1.8	11200
23	Brunei	10.5	67.4	2.84	28.0	80600	16.7	77.1	1.84	35300
24	Bulgaria	10.8	50.2	6.87	53.0	15300	1.11	73.9	1.57	6840
25	Burkina Faso	116.0	19.2	6.74	29.6	1430	6.81	57.9	5.87	575

- Setelah itu, saya memisahkan kolom yang berisi string nama negara

```
Nama_negara = data.iloc[:,:1]
Nama_negara
```

	country
0	Afghanistan
1	Albania
2	Algeria
3	Angola
4	Antigua and Barbuda
...	...
162	Vanuatu
163	Venezuela
164	Vietnam
165	Yemen
166	Zambia

167 rows × 1 columns

- Lalu menghapus kolom nama negara dari data

```
data = data.iloc[:,1:]
data
```

	child_mort	exports	health	imports	income	inflation	life_expec	total_fer	gdpp
0	90.2	10.0	7.58	44.9	1610	9.44	56.2	5.82	553
1	16.6	28.0	6.55	48.6	9930	4.49	76.3	1.65	4090
2	27.3	38.4	4.17	31.4	12900	16.10	76.5	2.89	4460
3	119.0	62.3	2.85	42.9	5900	22.40	60.1	6.16	3530
4	10.3	45.5	6.03	58.9	19100	1.44	76.8	2.13	12200
...	...	...	...	...	...	...	...	...	...
162	29.2	46.6	5.25	52.7	2950	2.62	63.0	3.50	2970
163	17.1	28.5	4.91	17.6	16500	45.90	75.4	2.47	13500
164	23.3	72.0	6.84	80.2	4490	12.10	73.1	1.95	1310
165	56.3	30.0	5.18	34.4	4480	23.60	67.5	4.67	1310
166	83.1	37.0	5.89	30.9	3280	14.00	52.0	5.40	1460

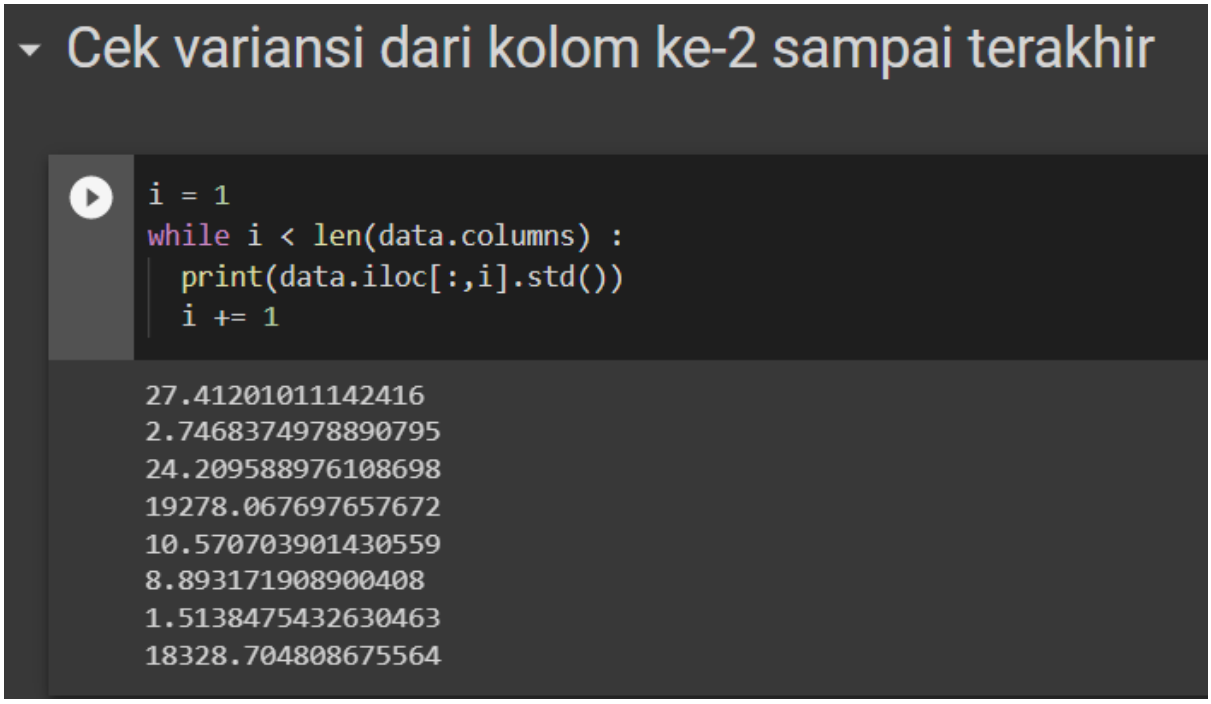
- Kemudian saya cek apakah ada missing value atau tidak

```
[ ] data.info()

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 167 entries, 0 to 166
Data columns (total 9 columns):
 #   Column      Non-Null Count  Dtype  
---  ------      --------------  -----  
 0   child_mort  167 non-null    float64
 1   exports     167 non-null    float64
 2   health      167 non-null    float64
 3   imports     167 non-null    float64
 4   income      167 non-null    int64  
 5   inflation   167 non-null    float64
 6   life_expec  167 non-null    float64
 7   total_fer   167 non-null    float64
 8   gdpp        167 non-null    int64  
dtypes: float64(7), int64(2)
memory usage: 11.9 KB
```

Karena tidak terdapat missing value, maka saya memakai semua kolom data

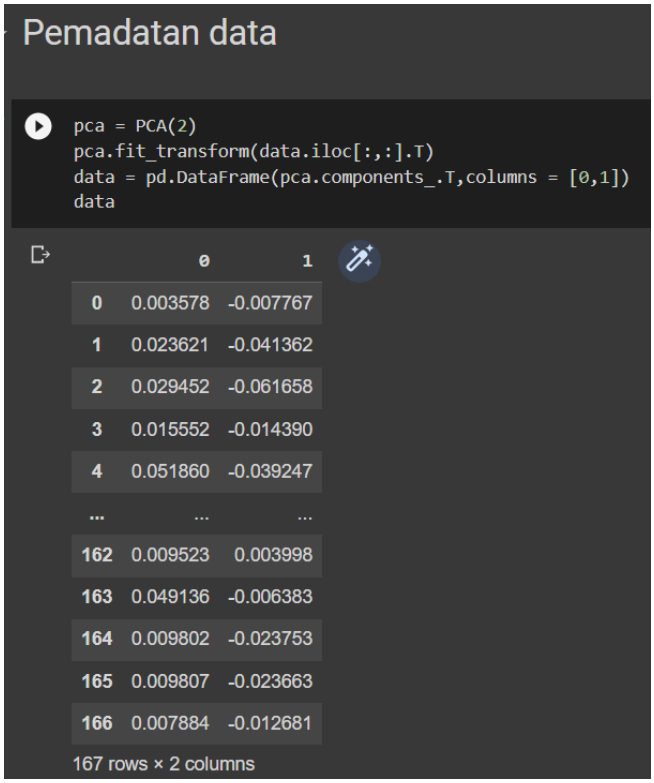
B. Ringkasan Pra Pemrosesan Data

- Yang pertama kali saya lakukan pada pre processing adalah mengecek variansi tiap kolom agar mengetahui apakah ada data yang mayoritas sama atau tidak



Dapat dilihat semua variansi lebih dari 1, maka semua kolom data memiliki kualitas yang bagus untuk dipakai

- Karna saya akan melakukan clustering, dan nantinya untuk evaluasi hasil saya akan menggunakan scatterplot, maka data yang saya miliki ini saya ringkas atau saya padatkan menjadi 2 kolom menggunakan PCA(Principal component analysis).



	0	1
0	0.003578	-0.007767
1	0.023621	-0.041362
2	0.029452	-0.061658
3	0.015552	-0.014390
4	0.051860	-0.039247
...	...	...
162	0.009523	0.003998
163	0.049136	-0.006383
164	0.009802	-0.023753
165	0.009807	-0.023663
166	0.007884	-0.012681

167 rows × 2 columns

- Setelah saya ringkas menjadi 2 kolom, saya melakukan scaling agar data tersebut berada dalam interval 0 sampai 1 saja

Normalisasi data

```
Scaler = preprocessing.MinMaxScaler(feature_range=(0,1))
data = Scaler.fit_transform(data)
data = pd.DataFrame(data)
data
```

	0	1
0	0.006598	0.508619
1	0.068366	0.458003
2	0.086337	0.427424
3	0.043500	0.498641
4	0.155396	0.461190
...	...	...
162	0.024921	0.526346
163	0.147001	0.510705
164	0.025781	0.484534
165	0.025794	0.484669
166	0.019869	0.501216

167 rows × 2 columns

- Setelah itu, untuk memudahkan saya dalam mengolah data tersebut, saya merubah bentuk data dari DataFrame menjadi bentuk array

```
arrData = []
for i in range(len(data)) :
  temp = []
  for j in range(0,len(data.columns)) :
    temp.append(data[j][i])
  arrData.append(temp)
```

C. Menerapkan Algoritma

Pada tugas case based 2 ini, saya melakukan clustering menggunakan algoritma DBSCAN. Karena algoritma DBSCAN ini kebal terhadap outlier, maka saya tidak melakukan sesuatu ke data outlier. Pada bagian ini, penggunaan library dilarang, oleh karena itu, saya membuat algoritma DBSCAN sendiri tanpa bantuan library.

- Yang pertama saya lakukan adalah membuat function untuk mencari jarak antar data menggunakan manhattan distance.

Membuat fungsi jarak menggunakan manhattan

```
def jarak(a,b) :
  hasil = 0
  for i in range(len(a)) :
    hasil += abs(a[i]-b[i])
  return hasil
```

- Lalu saya menentukan minPts dan juga Eps yang akan dipakai, dimana bagian ini lah yang nantinya akan saya gunakan beberapa kombinasi yang berbeda untuk melihat hasil mana yang terbaik

Menentukan minPts dan Eps

```
minPts = 10
Eps = 0.3
```

- Setelah menentukan minPts dan Eps, saya mencari data mana saja yang dapat dijadikan core point dan memasukkan data tersebut ke array core point

Menentukan core point

```
corePoint = []
for i in range(len(arrData)) :
  pts = 0
  for j in range(len(arrData)) :
    if i != j :
      if jarak(arrData[i],arrData[j]) <= Eps :
        pts += 1
  if pts >= minPts :
    corePoint.append(arrData[i])
print(len(corePoint))
```

```
157
```

- Karena pada array data yang saya gunakan belum ada tempat untuk menambahkan cluster, maka saya memberikan tempat terlebih dahulu untuk memasukkan cluster pada data tersebut yang saya mulai dengan 0 yang artinya data tersebut belum atau tidak masuk ke cluter manapun

Menambah field pada array yang nantinya akan diisi untuk menentukan termasuk kelas yang mana data tersebut

```
[14] for i in range(len(arrData)) :
       arrData[i] = [arrData[i],0] #diisi dengan 0 sebagai tanda bahwa data tersebut belum masuk ke kelas manapun
     print(arrData[0])
```

```
[[0.006597956727736502, 0.5086189989809378], 0]
```

- Setelah itu, saya memulai clustering data dengan cara menghitung jarak masing masing data ke core point yang sudah ditentukan sebelumnya

Memulai clustering data

```
Cluster = 1
while len(corePoint) > 0 :
  indexCore = random.randint(0,len(corePoint)-1)
  core = corePoint[indexCore]
  corePoint.remove(core)
  i = 0
  tempCores = []
  while i < len(arrData) :
    if arrData[i][1] == 0 :
      if jarak(arrData[i][0],core) <= Eps :
        arrData[i][1] = Cluster
        if arrData[i][0] in corePoint :
          corePoint.remove(arrData[i][0]) #menghapus corePoint yang sudah memiliki cluster
          tempCores.append(arrData[i][0])
    i += 1
  while len(tempCores) > 0 :
    core = tempCores[0]
    tempCores.remove(core)
    while i < len(arrData) :
      if arrData[i][1] == 0 :
        if jarak(arrData[i][0],core) <= Eps :
          arrData[i][1] = Cluster
          if arrData[i][0] in corePoint :
            corePoint.remove(arrData[i][0]) #menghapus corePoint yang sudah memiliki cluster
            tempCores.append(arrData[i][0])
      i += 1
  Cluster += 1
```

- Lalu, setelah saya selesai melakukan clustering, saya mengembalikan bentuk data menjadi DataFrame kembali

Mengembalikan data agar menjadi dataframe lagi

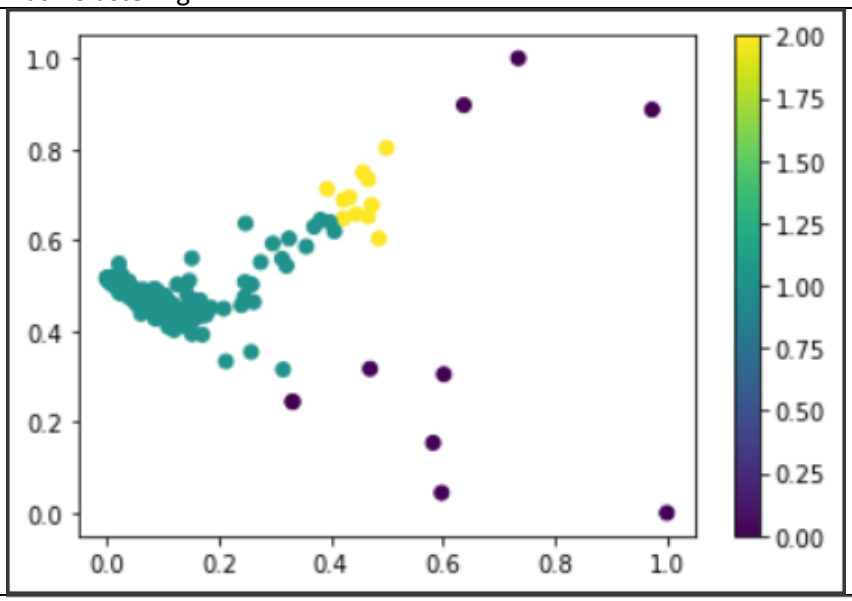
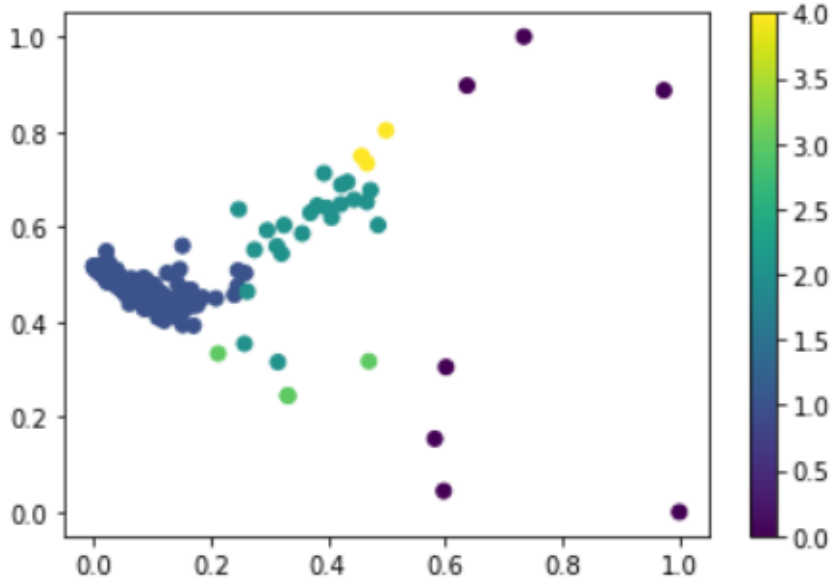
```
temp = []
for i in range(len(arrData)) :
  temp2 = []
  for j in range(len(arrData[i][0])) :
    temp2.append(arrData[i][0][j])
  temp2.append(arrData[i][1])
  temp.append(temp2)

for k in range(len(Nama_negara)) :
    temp[k].insert(0,Nama_negara['country'][k])
df = pd.DataFrame(temp,columns = ['country' ,0,1,'class'])
df
```

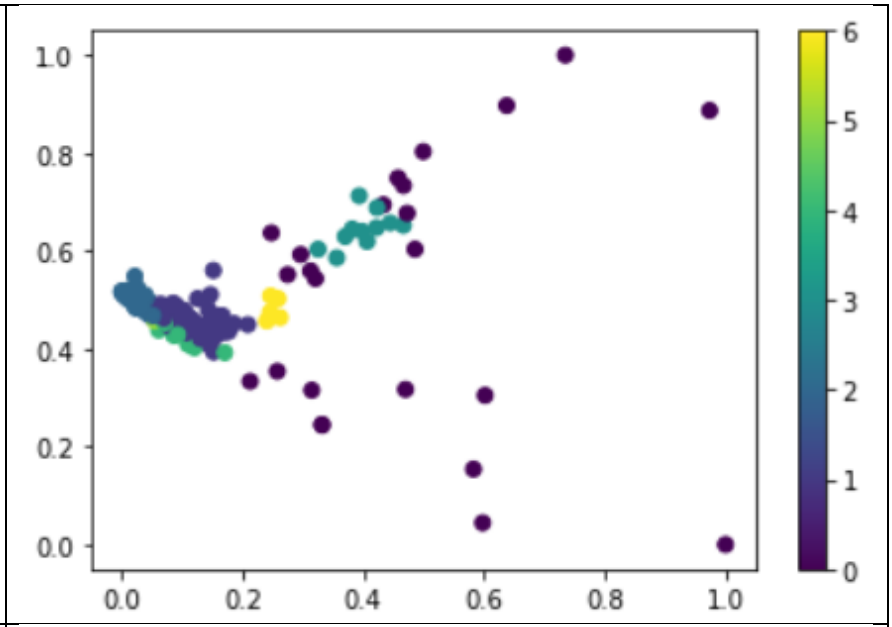
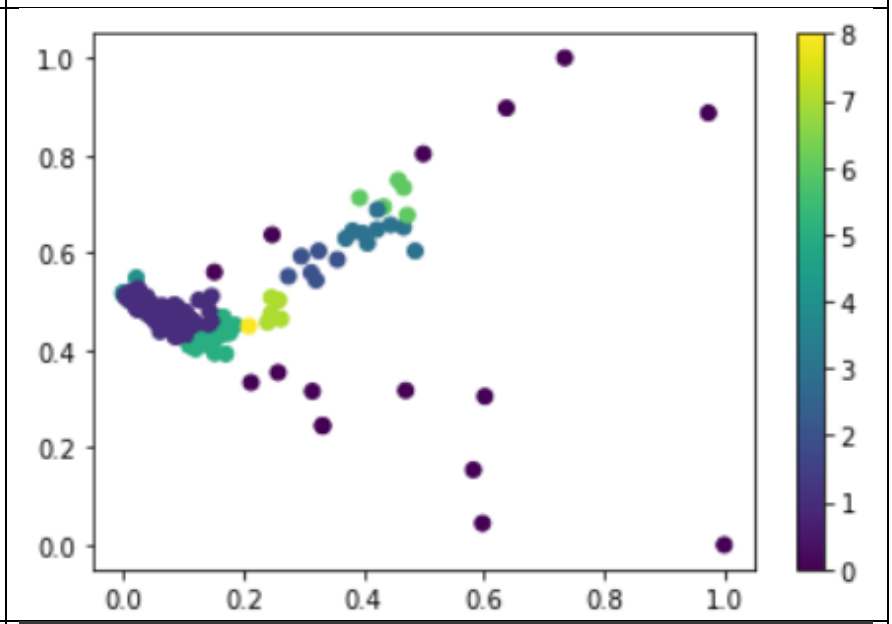
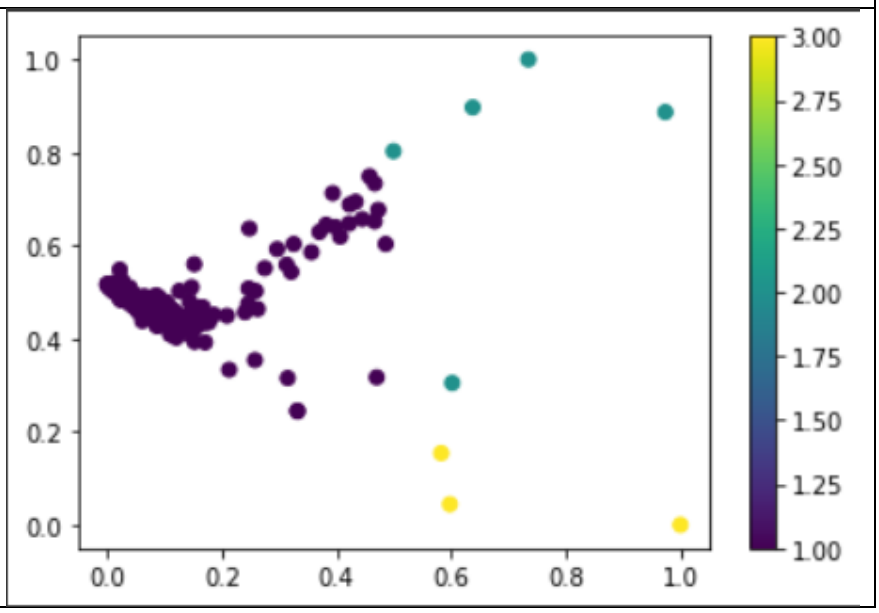
	country	0	1	class
0	Afghanistan	0.006598	0.508619	1
1	Albania	0.068366	0.458003	1
2	Algeria	0.086337	0.427424	1
3	Angola	0.043500	0.498641	1
4	Antigua and Barbuda	0.155396	0.461190	1
...	...	...	...	...
162	Vanuatu	0.024921	0.526346	1
163	Venezuela	0.147001	0.510705	1

- Link keseluruhan code :https://colab.research.google.com/drive/1Pduwn9YHBOTWrgmszVm-GbBlnyivLUHB?usp=sharing

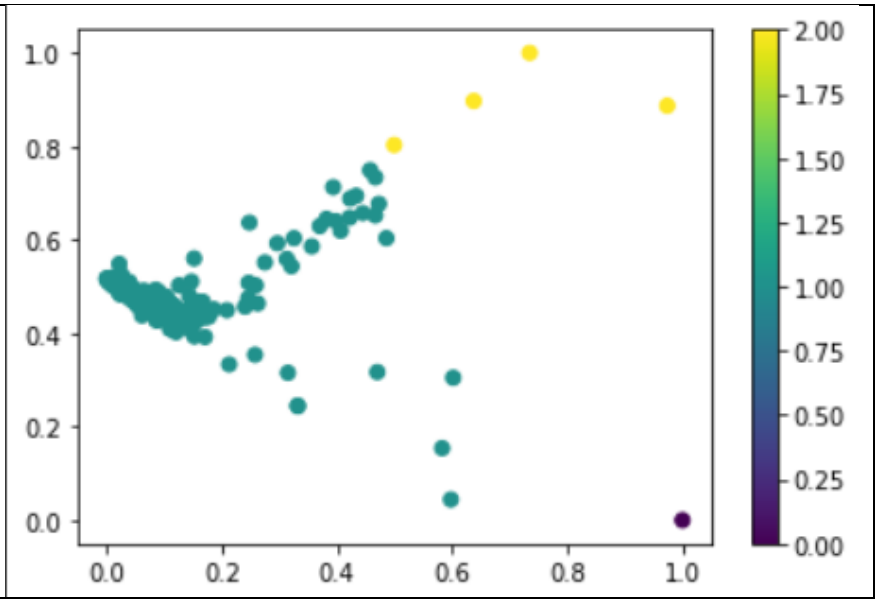
D. Evaluasi Hasil

Berikut adalah hasil clustering yang telah saya buat.

minPts dan Eps	Hasil Clustering
`minPts = 10` `Eps = 0.3`	
`minPts = 5` `Eps = 0.3`	

minPts = 10 Eps = 0.1	
minPts = 5 Eps = 0.1	
minPts = 10 Eps = 0.7	

```
minPts = 5
Eps = 0.7
```



Menurut pengamatan saya, hasil clustering terbaik didapat saat minPts = 10 dan Eps = 0.3. Dapat dilihat bahwa data yang berdekatan memiliki cluster atau warna yang sama dan terbagi secara rata atau sesuai tempatnya, tidak ada yang menumpuk dan data yang jaraknya jauh atau outlier termasuk cluster 0 yang sebenarnya berarti data tersebut tidak memiliki cluster.

E. Lampiran

Link Collab:

https://colab.research.google.com/drive/1Pduwn9YHBOTWrgmszVm-GbBlnyivLUHB?usp=sharing

Link Video :

https://youtu.be/GmTkeO_FA4Y

Link Slide :
https://docs.google.com/presentation/d/14MkGTYdTheZ7sRE4WHvB0HgwNjp_HYZd/edit?usp=sharing&ouid=117848929309785745185&rtpof=true&sd=true