Laporan Case-Based 2
Mata Kuliah Pembelajaran Mesin
K-MEANS CLUSTERING

Muhamad Azmi Rizkifar – 1301218586 – IFX-45-GAB – IKN

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

Program Studi Sarjana Informatika

Fakultas Informatika

Universitas Telkom

Bandung

2022

BAGIAN I
Ikhtisar Data Yang Dipilih

Dataset yang akan dipilih dalam pengimplementasian *K-Means Clustering* adalah dataset *Country Data* yang didapatkan dari *Kaggle*. Tujuan penulis adalah untuk membangun sebuah model *clustering* untuk mengklasifikasikan data negara berdasarkan data label yang nanti akan dibuat berdasarkan data yang telah di analisis. Data yang didapat disajikan dalam bentuk format file csv seperti gambar dibawah:

1	country	child_mort	exports	health	imports	income	inflation	life_expec	total_fer	gdpp
2	Afghanistan	90.2	10	7.58	44.9	1610	9.44	56.2	5.82	553
3	Albania	16.6	28	6.55	48.6	9930	4.49	76.3	1.65	4090
4	Algeria	27.3	38.4	4.17	31.4	12900	16.1	76.5	2.89	4460
5	Angola	119	62.3	2.85	42.9	5900	22.4	60.1	6.16	3530
6	Antigua and Barbuda	10.3	45.5	6.03	58.9	19100	1.44	76.8	2.13	12200
7	Argentina	14.5	18.9	8.1	16	18700	20.9	75.8	2.37	10300

Gambar 1 Dataset Country Data

Dari keseluruhan data yang ada, terdapat total 167 data negara yang terdiri dari 10 kolom dimana terdapat 1 *categorical feature* dan 9 *numerical feature* dengan keterangan sebagai berikut :

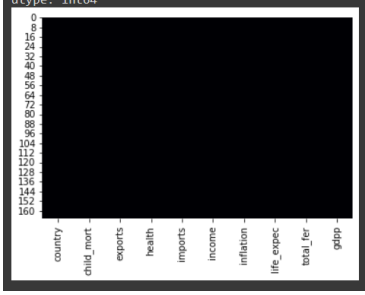
1. **country**: nama negara
2. **child_mort**: anak di bawah usia 5 tahun per 1000 kelahiran hidup
3. **exports**: ekspor barang dan jasa per kapita
4. **health**: total belanja kesehatan per kapita
5. **imports**: impor barang dan jasa per kapita
6. **income**: pendapatan bersih per orang
7. **inflation**: pengukuran tingkat pertumbuhan tahunan Total PDB
8. **life_expec**: rata-rata jumlah tahun seorang anak yang baru lahir akan hidup jika pola kematian saat ini tetap sama
9. **total_fer**: jumlah anak yang akan dilahirkan oleh setiap wanita jika tingkat kesuburan usia saat ini tetap sama
10. **gdpp**: PDB per kapita. Dihitung sebagai Total PDB dibagi dengan total populasi

Sebelum masuk ke tahap *pre*-processing, dilakukan Exploratory Data Analysis untuk meninjau apakah terdapat kecatatan data atau tidak.

```
# check the number of missing values per column
print(country_data.isnull().sum())

# Mencari data null dengan memplot data menggunakan heatmap
sns.heatmap(country_data.isnull(), cmap = 'magma', cbar = False);
```

```
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
```



Dari hasil pengecekan data diatas, tidak ditemukan adanya *missing value* atau *null value* pada data tersebut.

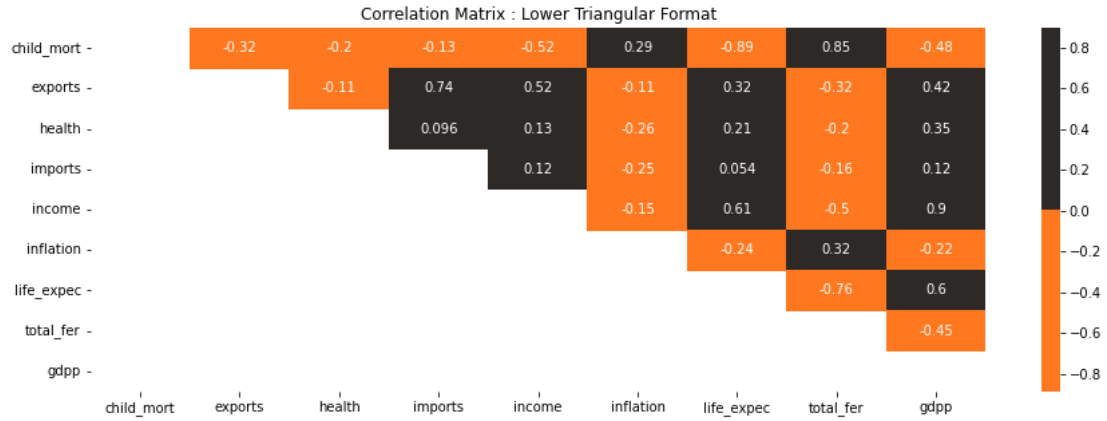
```
[253] # show description per feature
      country_data.describe()
```

	child_mort	exports	health	imports	income	inflation	life_expec	total_fer	gdpp
count	167.00	167.00	167.00	167.00	167.00	167.00	167.00	167.00	167.00
mean	38.27	41.11	6.82	46.89	17144.69	7.78	70.56	2.95	12964.16
std	40.33	27.41	2.75	24.21	19278.07	10.57	8.89	1.51	18328.70
min	2.60	0.11	1.81	0.07	609.00	-4.21	32.10	1.15	231.00
25%	8.25	23.80	4.92	30.20	3355.00	1.81	65.30	1.79	1330.00
50%	19.30	35.00	6.32	43.30	9960.00	5.39	73.10	2.41	4660.00
75%	62.10	51.35	8.60	58.75	22800.00	10.75	76.80	3.88	14050.00
max	208.00	200.00	17.90	174.00	125000.00	104.00	82.80	7.49	105000.00

Dari hasil pengecekan deskripsi data diatas, ditampilkan beberapa nilai jumlah, rata-rata, standar deviasi, dan lain sebagainya.

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 167 entries, 0 to 166
Data columns (total 10 columns):
 #   Column      Non-Null Count  Dtype
---  ------      --------------  -----
 0   country     167 non-null    object
 1   child_mort  167 non-null    float64
 2   exports     167 non-null    float64
 3   health      167 non-null    float64
 4   imports     167 non-null    float64
 5   income      167 non-null    int64
 6   inflation   167 non-null    float64
 7   life_expec  167 non-null    float64
 8   total_fer   167 non-null    float64
 9   gdpp        167 non-null    int64
dtypes: float64(7), int64(2), object(1)
memory usage: 13.2+ KB
```

Terdapat 10 kolom atau fitur dengan 1 *categorical feature* dengan tipe data *object* dan 9 *numerical feature* dengan tipe data *integer* dan *float*. Pada dataset ini, dikarenakan memiliki fitur yang sedikit maka hanya dilakukan pengecekan secara manual seperti tahapan diatas. Berikut ditampilkan gambar *Correlation Matrix*:



Beberapa fitur pada dasarnya berasal dari kategori yang sama dan memiliki keterkaitan terhadap fitur lainnya dari kategori yang berbeda. Hal ini bisa dibuktikan dengan :

1. Kenaikan nilai *export* meningkatkan nilai *import*, *income*, dan *gdpp*.
2. Kenaikan nilai *child_mort* terjadi ketika nilai inflasi dan *total_fer* meningkat.
3. Nilai *life_expec* yang tinggi menampilkan *total_fer* yang rendah dan *gdpp* yang tinggi menyebabkan banyak pengeluaran untuk kesehatan (*health*).

Beberapa fitur berasal dari katageori yang sama karena memiliki keterkaitan dengan fitur lainnya. oleh karena itu, penulis membagi kategori tersebut menjadi 3 dengan keterangan :

1. kesehatan (*health*): *child_mort*, *health*, *life_expec*, *total_fer*
2. perdagangan (*trade*): *import*, *export*
3. keuangan (*finance*): *income*, *inflation*, *gdpp*

BAGIAN II
Pra-Pemrosesan Data

Sebelum dilakukannya pembuatan model dan klasifikasi data, perlu dilakukannya tahap pra-pemrosesan data dengan harapan untuk mendapatkan hasil klasifikasi yang sesuai. Berikut merupakan penjabaran tahapan pra-pemrosesan yang dilakukan oleh penulis :

1. **Penggabungan fitur**

   Dikarenakan beberapa fitur berasal dari katageori yang sama karena memiliki keterkaitan dengan fitur lainnya, dilakukan penggabungan fitur menjadi 3 kategori yang terdiri dari :

   - kesehatan (*health*): *child_mort*, *health*, *life_expec*, *total_fer*
   - perdagangan (*trade*): *import*, *export*
   - keuangan (*finance*): *income*, *inflation*, *gdpp*

```
[260] process_data = pd.DataFrame()
      process_data['health'] = (country_data['child_mort'] / country_data['child_mort'].mean()) + (country_data['health'] / country_data['health'].mea
      process_data['trade'] = (country_data['imports'] / country_data['imports'].mean()) + (country_data['exports'] / country_data['exports'].mean())
      process_data['finance'] = (country_data['income'] / country_data['income'].mean()) + (country_data['inflation'] / country_data['inflation'].mean
```

   Selanjutnya, penulis melakukan penghapusan fitur *country* karena fitur tersebut merupakan *categorical data* yang dimana dalam algoritma *K-Means Clustering* tidak mendukung *categorical features*.

	health	trade	finance
0	6.24	1.20	1.35
1	3.04	1.72	1.47
2	3.39	1.60	3.17
3	6.47	2.43	3.49
4	2.96	2.36	2.24

2. **Scaling Feature**

   Pada tahapan ini, dilakukan scaling fitur karena *machine learning* hanya memperlakukan nilai input sebagai angka sederhana dan tidak memahami satuan nilai fitur. Maka dari itu, perlu dilakukan normalisasi data untuk fitur yang datanya tidak menampilkan distribusi secara normal. Penulis melakukan normalisasi *Min-Max* dengan mengubah sekumpulan data menjadi skalai mulai dari 0 (Min) hingga 1 (Max) dengan bantuan function *MinMaxScaler()*. Berikut terlampir formula perhitungan normalisasi Min-Max dan penerapannya :

$$X_{norm} = \frac{x' - min(x)}{max(x) - min(x)} (new_{max}(x) - new_{min}(x)) + new_{min}(x)$$

```
[261] mms = MinMaxScaler() # Normalization

      process_data['health'] = mms.fit_transform(process_data[['health']])
      process_data['trade'] = mms.fit_transform(process_data[['trade']])
      process_data['finance'] = mms.fit_transform(process_data[['finance']])

      process_data.insert(loc = 0, value = list(country_data['country']), column = 'country')
      process_data.head()
```

	country	health	trade	finance
0	Afghanistan	0.63	0.14	0.08
1	Albania	0.13	0.20	0.09
2	Algeria	0.18	0.19	0.21
3	Angola	0.66	0.28	0.24
4	Antigua and Barbuda	0.12	0.28	0.15

Dari hasil normalisasi data tersebut, data yang saat ini disajikan sudah berada di rentang 0 sampai dengan 1 dan distribusi datanya sudah terlihat normal (tidak ada data yang melebihi range 0 – 1.

BAGIAN III
Penerapan Algoritma

Pada proses penerapan algoritmanya, penulis menggunakan algoritma *K-Means Clustering* untuk melakukan clustering dan mengklasifikasikan data negara berdasarkan data label yang nanti akan dibuat berdasarkan data yang telah di analisis.

Berikut penjabaran implementasi *K-Means Clustering* dalam penyelesaian studi kasus *Country Data* dengan bantuan *Python* :

1. **Menentukan jumlah nilai K**

   Dalam menentukan jumlah nilai K digunakan *Elbow Method* sebagai salah satu metode yang memplot sum of squared error untuk rentang nilai K. Biasanya digunakan analogi lengan karena untuk menentukan nilai K, kita dapat memilih nilai yang menyerupai siku. Dari nilai ini, jumlah nilai kuadrat (inersia) mulai menurun secara linier dan karenanya dianggap sebagai nilai optimal.

```
wcss = []

for i in range(1, 11):
    clustering = KMeans(n_clusters=i, init='k-means++', random_state=42)
    clustering.fit(process_data_model)
    wcss.append(clustering.inertia_)

ks = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10]
sns.lineplot(x = ks, y = wcss);
```

   Dari hasil pembuatan plot diatas, diambil nilai 3 karena membentuk siku bawah yang menunjukkan nilai optimalnya.

2. **Pembangunan model**

   Pada pembangunan algoritmanya, penulis melakukan pembuatan model class K_Means yang memiliki 3 function yang berisi *__init__()* dan *fit()*. Pada function *__init__()* berfungsi sebagai inisialisasi class yang didalamnya menerima input parameter dan

melakukan set atribut di dalam class *K_Means*. Kemudian pada function *fit()* berfungsi sebagai proses pelatihan data yang menentukan lokasi *centroid* dan melakukan pengelompokkan data pada setiap titik dengan menghitung jarak terdekat dari *centroid*.

```
def __init__(self, k=2, tol=0.001, max_iter=300):
  self.k = k
  self.tol = tol
  self.max_iter = max_iter
```

```
def fit(self,data):
  self.centroids = {}

  for i in range(self.k):
    self.centroids[i] = data[i]

  for i in range(self.max_iter):
    self.classifications = {}

    for i in range(self.k):
      self.classifications[i] = []

    for featureset in data:
      distances = [np.linalg.norm(featureset-self.centroids[centroid]) for centroid in self.centroids]
      classification = distances.index(min(distances))
      self.classifications[classification].append(featureset)

    prev_centroids = dict(self.centroids)

    for classification in self.classifications:
      self.centroids[classification] = np.average(self.classifications[classification],axis=0)

    optimized = True

    for c in self.centroids:
      original_centroid = prev_centroids[c]
      current_centroid = self.centroids[c]
      if np.sum((current_centroid-original_centroid)/original_centroid*100.0) > self.tol:
        # print(np.sum((current_centroid-original_centroid)/original_centroid*100.0))
        optimized = False

    if optimized:
      break
```

3. Training Model

Pada proses training modelnya, penulis mendefinisikan paramter x_train sebagai fitur yang akan di latih, y_train sebagai target fitur, epoch sebanyak 50 iterasi, data validation yang digunakan untuk memvalidasi model dan mencegah *overfitting*, dan callbacks untuk menghentikan iterasi apabila nilai akurasi dan akurasi validasinya lebih besar dari 90%.

Pada proses training modelnya, penulis mendefinisikan parameter k dengan nilai 3 sesuai dengan hasil yang didapat dari *Elbow Method*. Kemudian dilakukan fitting model dengan menginputkan data yang berisi 3 fitur yang terdiri dari *health*, *trade*, dan *finance*.

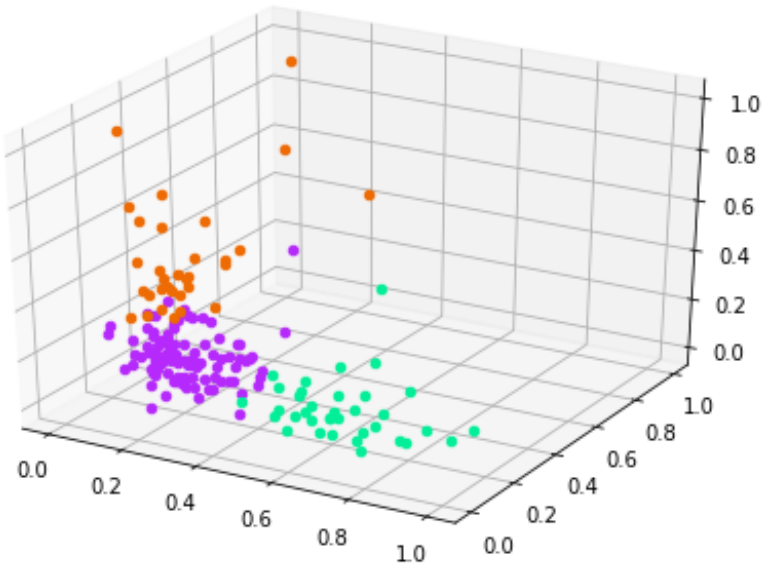
```
[310] # Inisiasi model
      model = K_Means(k=3)
      model.fit(process_data_model)

      70.96525645753786
      63.3610419030481
      80.50264775316064
      26.098108924784377
      24.805246531312598
      3.960054474003082
      14.892885076047737
      12.037677054581014
      2.25952930827778
      11.589575609123688
      1.5793096725679594
      5.190866398791849
      1.0625116954523595
      2.158144136808811
```

Link Google Collab : https://colab.research.google.com/drive/1kYxTvZVYc-M80lAp0dEH-gkiDno5cqsV?usp=sharing

BAGIAN IV
Evaluasi Hasil

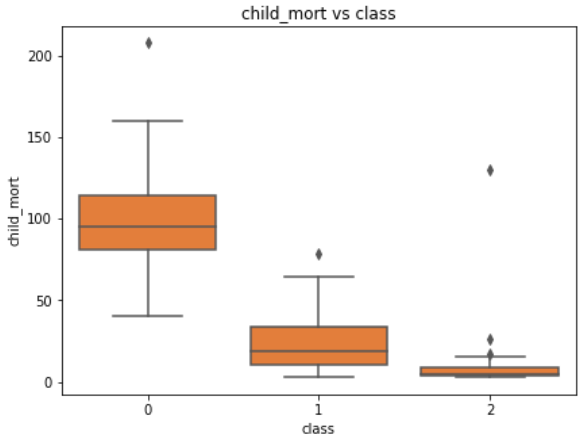
Berikut terlampir hasil clustering country data dengan nilai k = 3 :

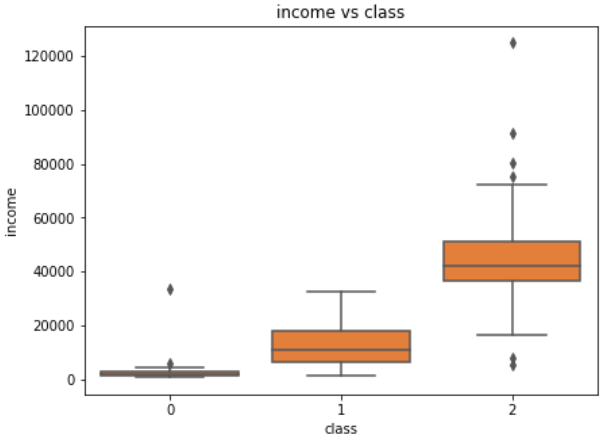


Hasil *centroids* pada proses *clustering* diatas :

```
{0: array([0.63055841, 0.18653481, 0.08930844]),
 1: array([0.18523165, 0.23585981, 0.13945257]),
 2: array([0.16519724, 0.27563471, 0.51548174])}
```

Seperti yang kita ketahui apabila nilai *income* rendah dan *child_mort* (kematian anak) tinggi adalah tanda bahwa negara tersebut terbelakang secara ekonomi, maka digunakan fitur *income* dan *child_mort* untuk menentukan label dari setiap class atau kelompok.

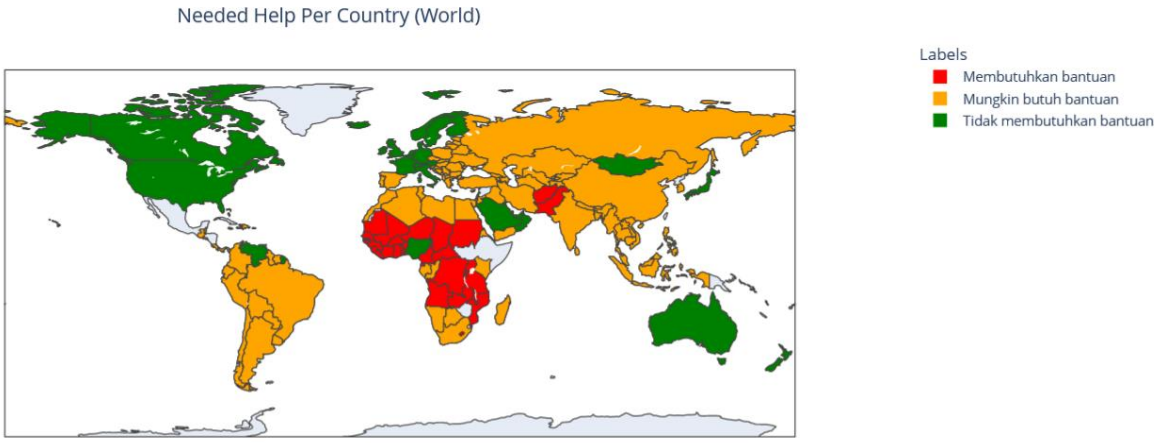




Dari hasil plot diatas, dapat disimpulkan bahwa :

- 0 : Membutuhkan bantuan
- 1 : Mungkin butuh bantuan
- 2 : Tidak membutuhkan bantuan

Selanjutnya dilakukan pembuatan plot peta dunia dengan pelabelan sesuai dengan class dan dataset country data.




```
Jumlah negara yang membutuhkan bantuan => 99 negara
Jumlah negara yang mungkin butuh bantuan => 32 negara
Jumlah negara yang tidak membutuhkan bantuan => 36 negara
```

Dari hasil plot pemetaan data dan class tersebut, terdapat 99 negara yang membutuhkan bantuan, 32 negara yang mungkin butuh bantuan, dan 36 negara yang tidak membutuhkan bantuan.

BAGIAN V
Presentasi Video

Link slide + source code :
https://github.com/azmirizkifar20/Machine-Learning-K-Means-Clustering

Link Video :
https://youtu.be/qxiVU1e1l2o