Clustering Negara Menggunakan Unsupervised Learning untuk International HELP

Tugas Case-based 2

Mata Kuliah Pembelajaran Mesin: Unsupervised Learning

Khalilullah Al Faath – 1301204376
Kode Dosen Pengampu: DDR
Program Studi S-1 Informatika, Fakultas Informatika
Universitas Telkom, Jl. Telekomunikasi, Terusan Buahbatu, Sukapura
E -mail (gmail): khalilullahalfaath@student.telkomuniversity.ac.id

***Abstrak*—salah satu algoritma dalam pembelajaran mesin adalah unsupervised learning, yaitu algoritma yang mengidentifikasi pola dari data set yang mana data pointnya belum diklasifikasi atau dilabelkan. Dataset yang digunakan di sini adalah dataset yang diuat oleh HELP Internasional yang mengumpulkan data-data dari negara yang kemudian akan dikelompokkan, apakah membutuhkan bantuin atau tidak. Di sini penulis menggunakan algoritma k-means dalam pengerjaannya.**

Keywords—unsupervised, learning, k-means

I. PENDAHULUAN

A. Definisi unsupervised learning

Pada algoritma unsupervised learning, data tidak memiliki label secara eksplisit dan model mampu belajar dari data dengan menemukan pola yang implisit. Sangat berbeda dengan supervised learning, unsupervised learning merupakan jenis learning yang hanya mempunyai variabel input tapi tidak mempunyai variabel output yang berhubungan. Tujuan dari Machine Learning ini adalah untuk memodelkan struktur data dan menyimpulkan fungsi yang mendeskripsikan data tersebut.

Unsupervised learning adalah salah satu tipe algoritma machine learning yang digunakan untuk menarik kesimpulan dari dataset. Metode ini hanya akan mempelajari suatu data berdasarkan kedekatannya saja atau yang biasa disebut dengan clustering. Metode unsupervised learning yang paling umum adalah analisis cluster, yang digunakan pada analisa data untuk mencari pola-pola tersembunyi atau pengelompokan dalam data (Miftah Rezkia, 2020).

B. Definisi k-means

Algoritma k-means adalah salah satu algoritma unsupervised learning yang mana pengelompokan data dilakukan berdasarkan kedekatannya dengan suatu centroid yang kemudian data dikelompokkna sebanyak k-cluster untuk setiap centroid (Delyani Nursyafitri, 2022).

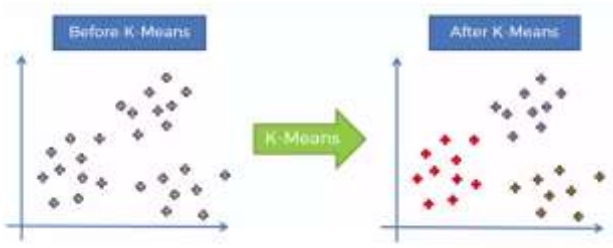


Fig 1. Contoh penerapan k-means

1) Proses klasterisasi dengan k-means

Untuk melakukan klasterisasi, dilakukan beberapa tahapan berikut:

- Penentuan nilai k atau cluster yang akan dibuat.
- Inisialisasi nilai centroid[1] secara random.
- Menetapkan jarak setiap data point ke centroid terdekat.
- Menghitung ulang nilai centroid dari cluster yang baru terbentuk dengan nilai rata-rata setiap data point yang memiliki jarak terdekat dari setiap centroid.
- Melakukan optimasi agar kriteria terpenuhi dengan mengulang step 3 dan 4 (Delyani Nursyafitri, 2022).

2) Kelebihan dan kekuranganklasterisasi dengan k-means

Berikut ini adalah kelebihan yang dimiliki oleh K-Means Clustering:

- Terbilang cukup mudah untuk dipahami dan diimplementasikan.
- Proses pembelajaran membutuhkan waktu yang relatif cepat.
- Sangat umum digunakan sebagai teknik clustering

Selain kelebihan, k-Means Clustering tentunya juga memiliki kekurangan. Beberapa kekurangannya adalah:

[1] Centroid merupakan nilai pusat (center) dari sebuah cluster. Misalkan kita mengatur k = 3, maka akan terbentuk centroid C1, C2, dan C3 secara random.

- Perlu inisialisasi nilai k menggunakan metode lain untuk mendapatkan nilai k yang optimal.
- Apabila hasil nilai random untuk centroid kurang baik, maka hasil clustering yang didapatkan menjadi tidak optima
- Cukup sulit jika digunakan untuk mencari jarak dari data yang berdimensi banyak (Delyani Nursyafitri, 2022).

3) PCA (Principal Component Analysis)

PCA pada dasarnya merupakan dasar dari analisis data multivariat yang menerapkan metode proyeksi. Teknik analisis ini biasanya digunakan untuk meringkas tabel data multivariat dalam skala besar hingga bisa dijadikan kumpulan variabel yang lebih kecil atau indeks ringkasan. Dari situ, kemudian variabel dianalisis untuk mengetahui tren tertentu, klaster variabel, hingga outlier.

II. INFORMASI TERKAIT DATASET

A. Judul Dataset

Dataset yang digunakan berjudul "Clustering the Countries by using Unsupervised Learning for HELP International".

B. Tujuan dari dataset

Dataset ini bertujuan untuk meng-kategorisasi negara berdasarkan factor sosio-ekonomi dan Kesehatan untuk menentukan bagaimana kemajuan suatu negara secara umum[2].

C. File dataset

Dataset ini diambil dari Kaggle dengan judul "*Unsupervised Learning on Country Data*". Terdapat dua file, yaitu country-data.csv dan data-dictionary.csv.

Dataset ini terdiri dari 167 baris yang terdiri dari negara-negara dan 10 kolom yang berisi aspek-aspek sosio-ekonomi dan Kesehatan suatu negara.

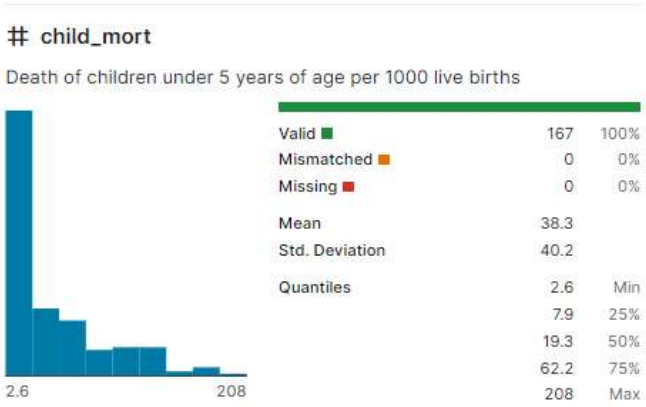
III. IKHTISAR KUMPULAN DATA YANG DIPILIH

A. Penjelasan Tiap Kolom

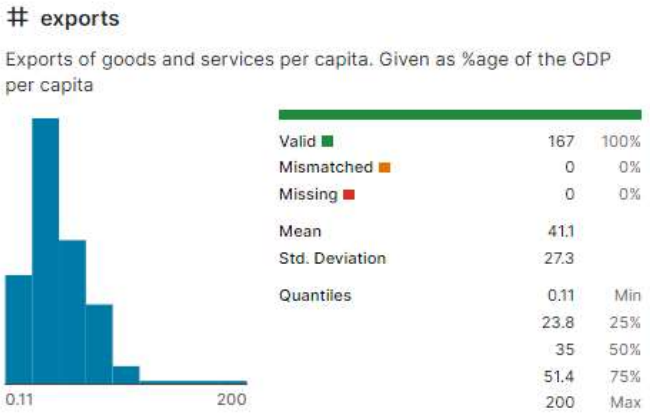
Informasi terkait dataset di file data-dictionary.csv

country: Nama negara

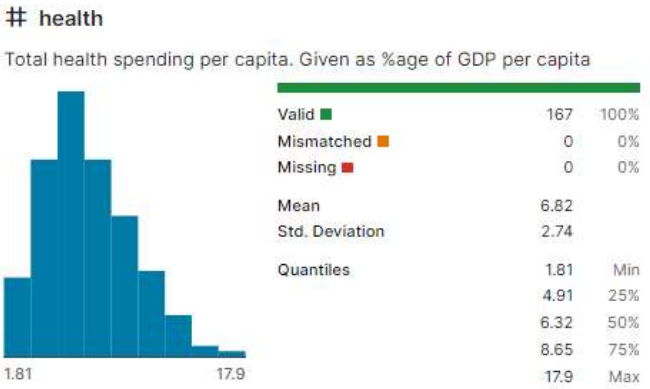
child_mort: Kematian anak di bawah umur 5 tahun setiap 1000 kelahiran



exports: Jumlah eksport barang dan servis percapita. Diberikan sebagai persentase dari GDP perkapita



health: Jumlah pengeluaran untuk biaya kesehatan perkapita. Diberikan sebagai persentase GDP perkapita

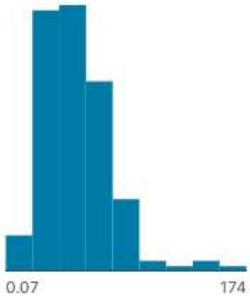


imports: Jumlah import barang dan servis perkapita. Diberikan sebagai persentase GDP perkapita

[2] https://www.kaggle.com/datasets/rohan0301/unsupervised-learning-on-country-data

 # imports

Imports of goods and services per capita. Given as %age of the GDP per capita

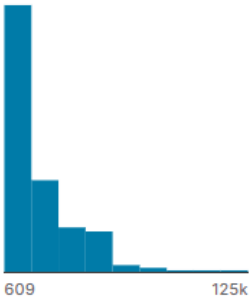


Valid	167	100%
Mismatched	0	0%
Missing	0	0%
Mean	46.9	
Std. Deviation	24.1	
Quantiles	0.07	Min
	30	25%
	43.3	50%
	58.9	75%
	174	Max

Income: Pendapatan bersih perorang

 # income

Net income per person

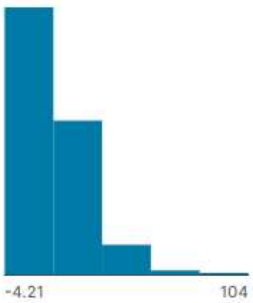


Valid	167	100%
Mismatched	0	0%
Missing	0	0%
Mean	17.1k	
Std. Deviation	19.2k	
Quantiles	609	Min
	3340	25%
	9960	50%
	22.9k	75%
	125k	Max

Inflation: Pengukuran dari pertambahan jumlah GDP pertahun

 # inflation

The measurement of the annual growth rate of the Total GDP

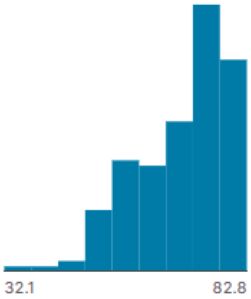


Valid	167	100%
Mismatched	0	0%
Missing	0	0%
Mean	7.78	
Std. Deviation	10.5	
Quantiles	-4.21	Min
	1.77	25%
	5.39	50%
	10.9	75%
	104	Max

life_expec: Rata-rata jumlah tahun dari anak yang belum lahir akan hidup jika pola mortalitas tetap sama

 # life_expec

The average number of years a new born child would live if the current mortality patterns are to remain the same

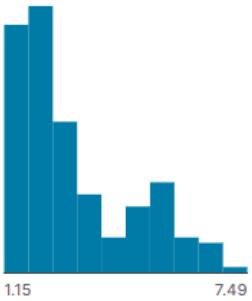


Valid	167	100%
Mismatched	0	0%
Missing	0	0%
Mean	70.6	
Std. Deviation	8.87	
Quantiles	32.1	Min
	65.3	25%
	73.1	50%
	76.8	75%
	82.8	Max

total_fer: Banyak anak yang akan lahir untuk setiap wanita jika age-fertility sekarang masih sama

 # total_fer

The number of children that would be born to each woman if the current age-fertility rates remain the same.

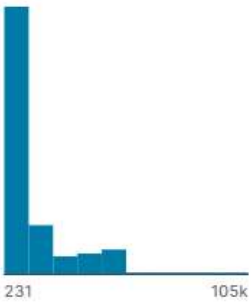


Valid	167	100%
Mismatched	0	0%
Missing	0	0%
Mean	2.95	
Std. Deviation	1.51	
Quantiles	1.15	Min
	1.79	25%
	2.41	50%
	3.91	75%
	7.49	Max

gdpp: GDP perkapita. Dihitung dari total GDP dan dibagi dengan banyaknya populasi

 # gdpp

The GDP per capita. Calculated as the Total GDP divided by the total population.



Valid	167	100%
Mismatched	0	0%
Missing	0	0%
Mean	13k	
Std. Deviation	18.3k	
Quantiles	231	Min
	1310	25%
	4660	50%
	14.6k	75%
	105k	Max

B. Jumlah baris dan kolom

Jumlah baris dan kolom dari dataset dapat dicek sebagai berikut:

Fig 1. Dimensi terkait dataset

C. Tiga data pertama

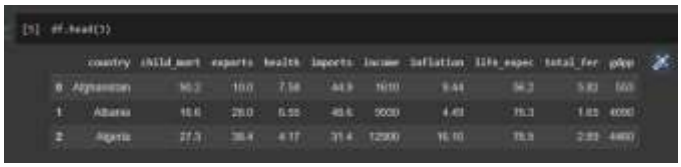


Fig 2. Tiga data pertama dari dataset

D. Tiga data terakhir

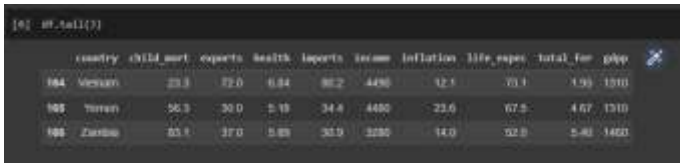


Fig 3. Tiga data terakhir dari dataset

E. Tiga data sampel

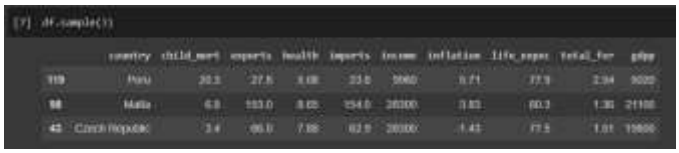


Fig 4. Tiga data sampel dari dataset

F. Informasi terkait dataset

```
[8]  df.info()

<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
```

Fig 5. Info tipe data dari dataset

Dapat dilihat bahwa dataset di atas memiliki kolom sebanyak 10 dengan baris sebanyak 167 dan tidak ada datanull. Tipe data yang ada, yaitu object (string) untuk data country, integer untuk data income dan gdpp, dan selainnya adalah float (floating number integer) atau data kontinu.

G. Informasi terkait dataset secara deskriptif

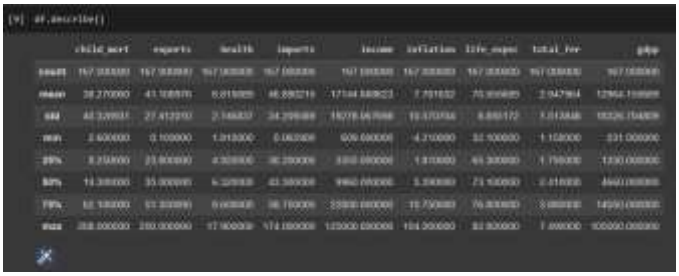


Fig 6. Informasi dataset secara deskriptif

Dapat dilihat bahwa untuk rata-rata data terdistribusi ke kiri (condong ke arah kiri). Kecuali untuk data life_expec yang condong ke kanan.

Data dengan variansi data yang lebar atau jauh dari rata-rata antara lain:

- Child_mort
- Gdpp
- Income
- Total_fer

Data yang memiliki data negative hanyalah inflation.

H. Kualitas Data

Dapat dilihat dari keterangan di atas bahwa dataset di atas memiliki kualitas yang cukup bagus. Walaupun tidak terdistribusi secara normal. Untuk selengkapnya bisa dilihat dari beberapa diagram berikut.

Fig 7. Pemisahan data kolom ‘country’ karena merupakan string. Sheingga tidak bisa diolah

I. Pemilihan Preprocessing yang dilakukan

1) Mengecek duplikasi data

Data yang merupakan data duplikasi harus dihapus karena bisa saja nanti akan merusak model algoritma yang telah dibangun

2) Mengecek missing values

Missing values adalah data yang tidak memiliki nilai. Sehingga harus dilakukan handling

3) Mengecek outlier

Outlier adalah data pencilan. Sebuah pencilan adalah titik data yang terpaut jauh dari titik data lainnya.

Karena k-means adalah algoritma yang sensitive pada pencilan maka perlu dilakukan penghandlingan jika perlu.

4) Scaling data dengan min-max scaling

Untuk memudahkan komputasi, dilakukan min-max scaling untuk mentransformasi data sehingga data hanya terdiri atas [0,1]

Rumus min-max scaling adalah sebagai berikut:

$$x' = \frac{x - \min(x)}{\max(x) - \min(x)}$$

J. Feature engineering atau feature selection yang dilakukan

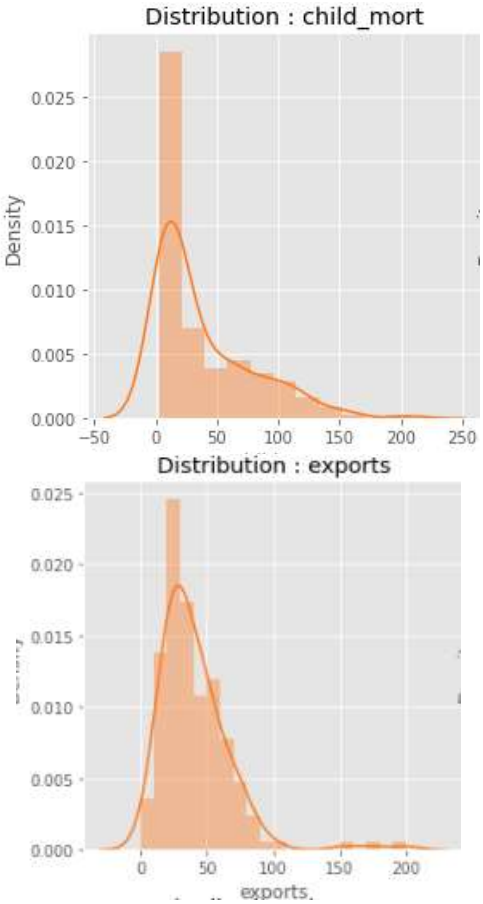
- *PCA*
- Pembagian secara literal berdasarkan kelompok
- Feature selection berdasarkan korelasi

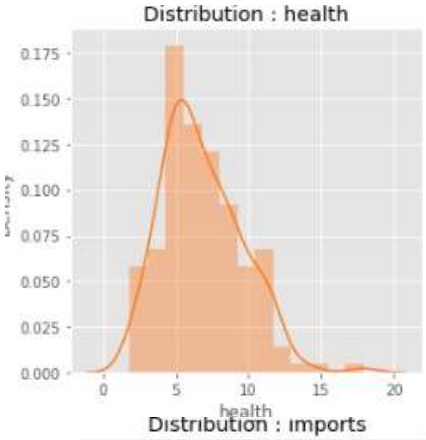
K. Plot

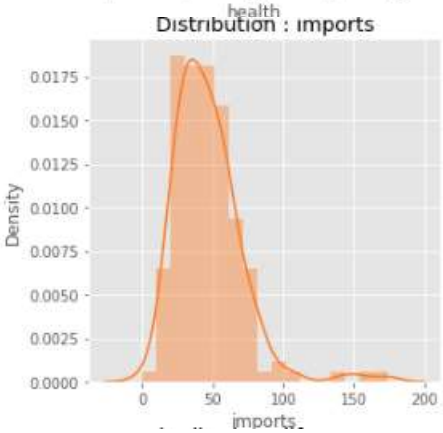
1) plot density (kedalaman data)

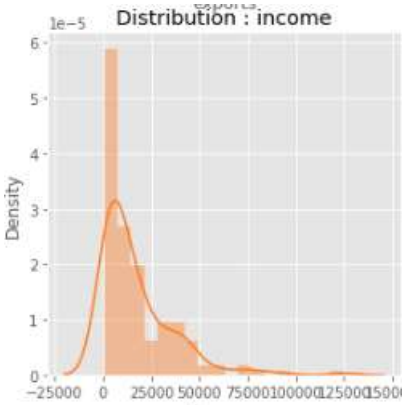
```
colors = ['#FF781F','#2D2926']
fig, ax = plt.subplots(nrows = 3,ncols = 3,figsize = (15,15))
for i in range(len(col_num)):
    plt.subplot(3,3,i+1)
    sns.distplot(df[col_num[i]],color = colors[0])
    title = 'Distribution : ' + col_num[i]
    plt.title(title)
plt.show()
```

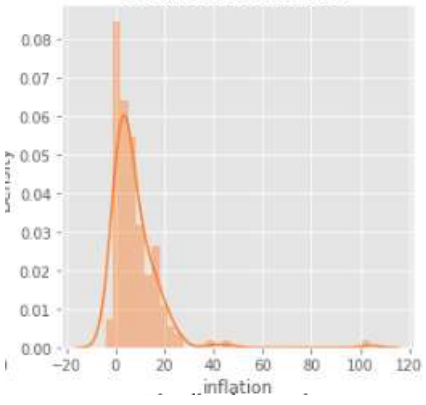
Fig 8. Kode untuk menghitung plot density

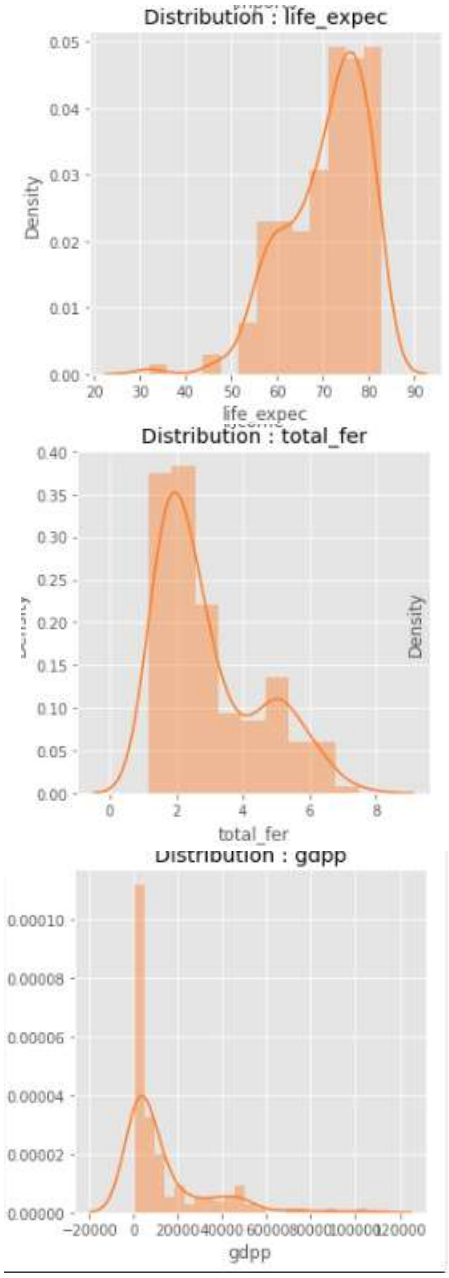












Dari diagram di atas dapat kita lihat bagaimana distribusi dari data yang ada. Kalau datanya berkumpul di arah kiri berarti datanya condong ke arah kiri dan apabila datanya berkumpul di arah kanan maka datanya condong ke arah kanan. Dengan hasil sebagai berikut:

- Condong ke-kiri: Semua fitur kecuali life_expec
- Condong ke-kanan: life expec

2) Plot korelasi

```
df.corr()
```

	child_mort	exports	health	imports	income	inflation	life_expec	total_fer	gdpp
child_mort	1.000000	-0.318093	-0.200402	-0.127211	-0.524315	0.288276	-0.886676	0.848478	-0.483032
exports	-0.318093	1.000000	-0.114408	0.737381	0.516784	-0.107294	0.316343	-0.320011	0.418725
health	-0.200402	-0.114408	1.000000	0.095717	0.129579	-0.255376	0.210032	-0.190674	0.345966
imports	-0.127211	0.737381	0.095717	1.000000	0.122406	-0.246994	0.054391	-0.159048	0.115498
income	-0.524315	0.516784	0.129579	0.122406	1.000000	-0.147756	0.611962	-0.501840	0.895571
inflation	0.288276	-0.107294	-0.255376	-0.246994	-0.147756	1.000000	-0.239708	0.316921	-0.221631
life_expec	-0.886676	0.316343	0.210032	0.054391	0.611962	-0.239708	1.000000	-0.760875	0.600089
total_fer	0.848478	-0.320011	-0.190674	-0.159048	-0.501840	0.316921	-0.760875	1.000000	-0.454910
gdpp	-0.483032	0.418725	0.345966	0.115498	0.895571	-0.221631	0.600089	-0.454910	1.000000

Fig 9. Korelasi dalam bentuk tabel

```
fig, ax = plt.subplots(figsize=(10, 8))

mask = np.triu(np.ones_like(df.corr(), dtype=bool))

heatmap = sns.heatmap(df.corr(), mask=mask, vmin=-1, vmax=1, annot=True)

plt.suptitle("Correlation Plot", ha='left', x=0.155, y=1.04, fontsize=18, fontweight='bold')
plt.title("Korelasi antara beberapa fitur", loc='left', fontsize=12)
plt.tight_layout()
plt.show()
```

Fig 10. Code untuk korelasi plot

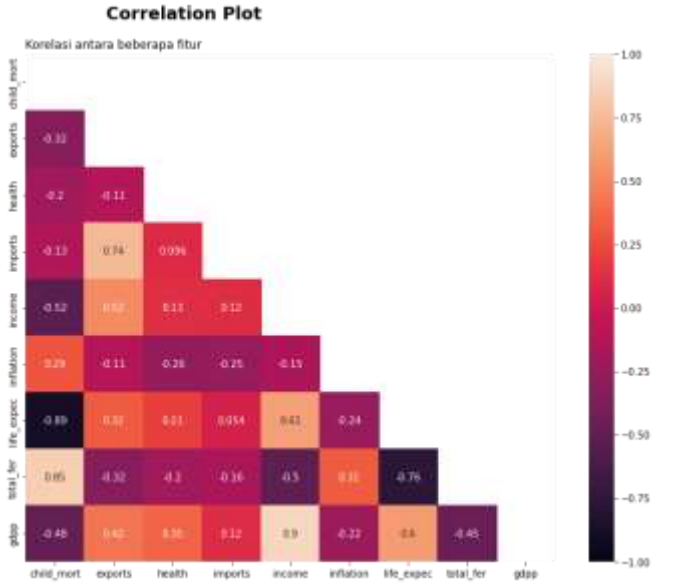


Fig 11. Plot korelasi

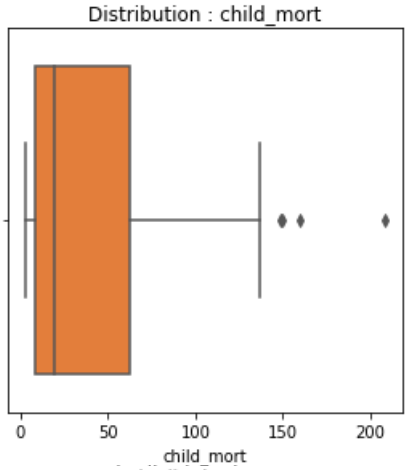
```
corr = df.corr()
plt.figure(figsize=(12,10))
sns.heatmap(corr[(corr>=0.5) | (corr <= -0.5)],vmax=1.0, vmin=-1.0,linewidths=0.1,annot=True,
            annot_kws={"size":14},square=True)
```

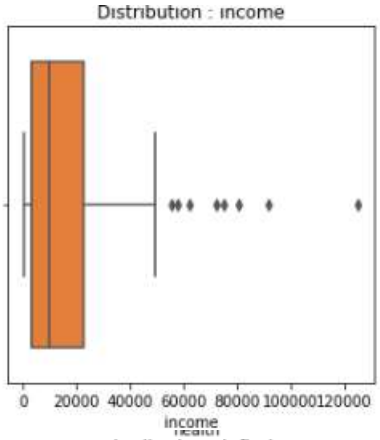
Fig 12. Kode plot korelasi yang hanya menampilkan yang >= 0.5 dan <= 0.5

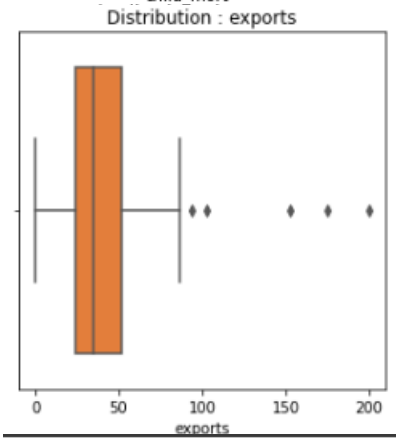
Fig 13. Plot korelasi yang kurang dari sama dengan -0.5 dan lebih dari sama dengan 0.5

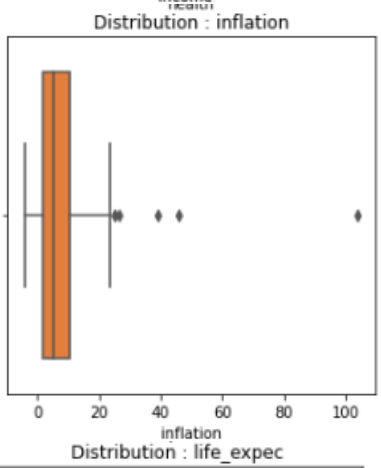
Dapat dilihat dari korelasi plot di atas bahwa yang bis akita pilih untuk fitur selection adalah life_expec, total_fer, dan gdpp. Karena memiliki korelasi yang baik untuk semua fitur.

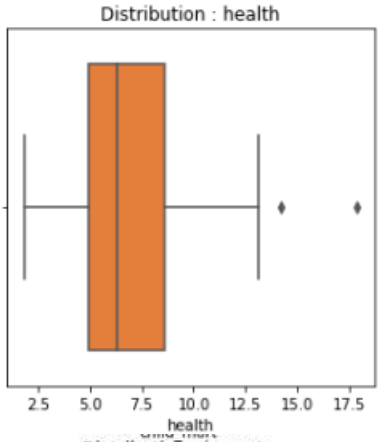
L. Boxplot

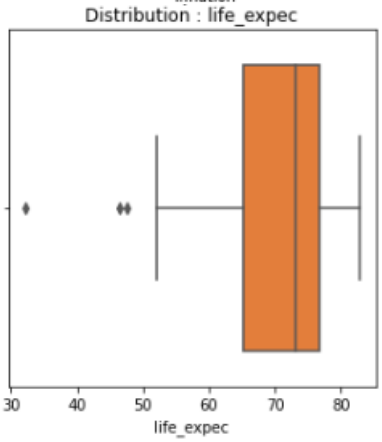

Distribution : child_mort
0
50
100
150
200
child_mort

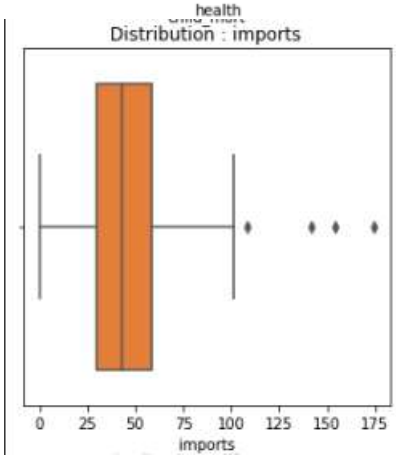

Distribution : income
0
20000
40000
60000
80000
100000
120000
income

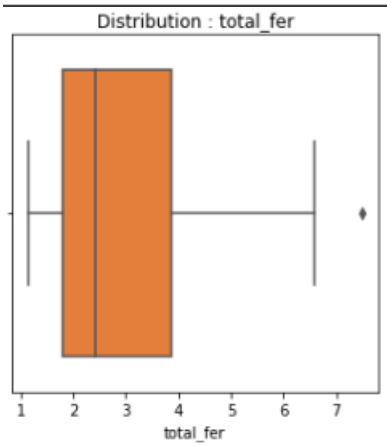

Distribution : exports
0
50
100
150
200
exports

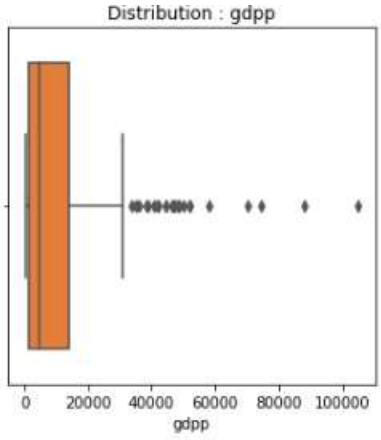

Distribution : inflation
0
20
40
60
80
100
inflation


Distribution : health
2.5
5.0
7.5
10.0
12.5
15.0
17.5
health


Distribution : life_expec
30
40
50
60
70
80
life_expec


Distribution : imports
0
25
50
75
100
125
150
175
imports


Distribution : total_fer
1
2
3
4
5
6
7
total_fer



Dapat dilihat dengan lebih jelas distribusi dari data yang semuanya kecuali fitur life_expec condong ke kiri daripada ke kanan. Selain itu, semua data memiliki outlier. Fitur dengan outlier terbanyak ada di gdpp, income, dan inflation.

IV. RINGKASAN PRA-PEMROSESAN DATA YANG DIUSULKAN

A. Mengecek duplikasi data

```
[11] bool_series = df.duplicated()
     print(bool_series)

0      False
1      False
2      False
3      False
4      False
       ...
162    False
163    False
164    False
165    False
166    False
Length: 167, dtype: bool

[12] print(type(bool_series))
     bool_series.value_counts()

<class 'pandas.core.series.Series'>
False    167
dtype: int64
```

Tidak ada duplikasi pada record data

Didapatkan bahwa tidak ada duplikasi data untuk setiap baris yang ada.

B. Mengecek missing values

```
[13] df.isna().sum()

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
```

Karena tidak ada missing values, maka tidak perlu dilakukan handling

Fig 14. Code untuk mengecek missing values

```
sns.heatmap(df.isnull(),cbar=False)
plt.title("Heatmap missing value")
plt.show()
```

Fig 15. Kode untuk mengecek heatmap dari missing values

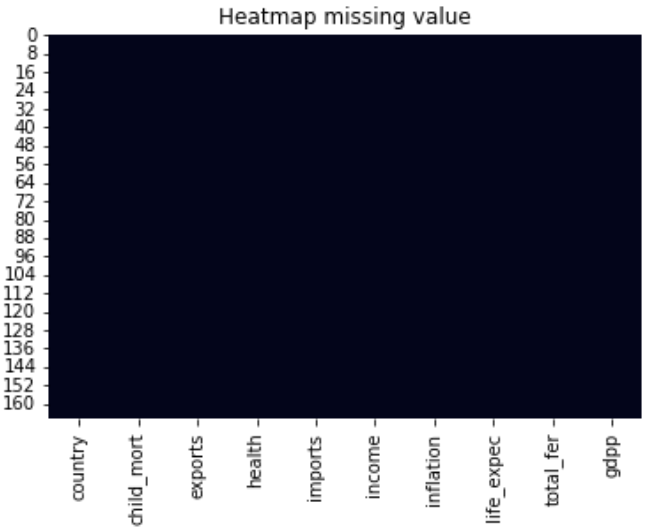


Fig 16. Heatmap missing values

Karena missing values tidak ditemukan maka tidak perlu dilakukan handling missing values.

C. Menghitung Outlier

```
def hitungOutliers(df):
    q1 = df.quantile(0.25)
    q3 = df.quantile(0.75)
    IQR = q3 - q1
    outliers = df[((df<(q1-1.5*IQR)) | (df>(q3+1.5*IQR)))]
    return outliers

for i in col_num:
  outliers = hitungOutliers(df[i])
  percentage = len(outliers)/len(df[i])*100
  print(i)
  print("number of outliers: " + str(len(outliers)))
  print("max outlier value: " + str(outliers.max()))
  print("min outlier value: " + str(outliers.min()))
  print("Outliers percentage: "+ str(float(f'{percentage:.2f}'))+"%")
  print()
```

Fig 17. Kode untuk menghitung missing values dan iterasi untuk setiap kolom

```
child_mort
number of outliers: 4
max outlier value: 208.0
min outlier value: 149.0
Outliers percentage: 2.4%

exports
number of outliers: 5
max outlier value: 200.0
min outlier value: 93.8
Outliers percentage: 2.99%

health
number of outliers: 2
max outlier value: 17.9
min outlier value: 14.2
Outliers percentage: 1.2%
```

```
imports
number of outliers: 4
max outlier value: 174.0
min outlier value: 108.0
Outliers percentage: 2.4%

income
number of outliers: 8
max outlier value: 125000
min outlier value: 55500
Outliers percentage: 4.79%

inflation
number of outliers: 5
max outlier value: 104.0
min outlier value: 24.9
Outliers percentage: 2.99%
```

```
life_expec
number of outliers: 3
max outlier value: 47.5
min outlier value: 32.1
Outliers percentage: 1.8%

total_fer
number of outliers: 1
max outlier value: 7.49
min outlier value: 7.49
Outliers percentage: 0.6%
```

```
gdpp
number of outliers: 25
max outlier value: 105000
min outlier value: 33700
Outliers percentage: 14.97%
```

Dapat dilihat bahwa:

- Outlier di bawah 1%: total_fer
- Outlier di atas 1% di bawah 5%: child_mort, exports, health, imports, income, inflation, life_expect
- Outlier di atas 10%: gdpp

Karena hanya kolom gdpp yang memiliki outlier di atas 10% dan ini wajar karena gdpp antara negara terbelakang dan negara maju itu sangat jauh. Biasanya dilakukan handling dengan mepetkan outlier, tetapi penulis rasa tidak perlu.

D. Feature Selection

1) Pengelompokan kolom berdasarkan kategori

```
df_literal = pd.DataFrame()
df_literal['Country'] = df['country']
df_literal['Health'] = ((df['health']/df['health'].mean())
+(df['child_mort']/df['child_mort'].mean())
+(df['life_expec']/df['life_expec'].mean())+
(df['total_fer']/df['total_fer'].mean()))

df_literal['Economics'] = ((df['exports']/df['exports'].mean())
+(df['imports']/df['imports'].mean()))

df_literal['Moneter'] = ((df['income']/df['income'].mean())
+(df['gdpp']/df['gdpp'].mean())
+(df['inflation']/df['inflation'].mean()))
```

	Country	Health	Economics	Moneter
0	Afghanistan	6.239852	1.200812	1.349645
1	Albania	3.035901	1.717580	1.471658
2	Algeria	3.389763	1.603752	3.165367
3	Angola	6.469020	2.430387	3.494919
4	Antigua and Barbuda	2.964898	2.362940	2.240150
...	...	...	...	...
162	Vanuatu	3.613452	2.257474	0.737840
163	Venezuela	3.073747	1.068624	7.902084
164	Vietnam	3.309933	3.461820	1.917840
165	Yemen	4.771970	1.463396	3.395056
166	Zambia	5.604372	1.559033	2.102993

167 rows × 4 columns

Dapat dilihat bahwa data bisa dibagi setidaknya dengan pengelompokan literal sebagai berikut:

* Health : Health, child_mort, life_expec, total_fer
* economics: exports, imports
* moneter: income, gdpp, inflation

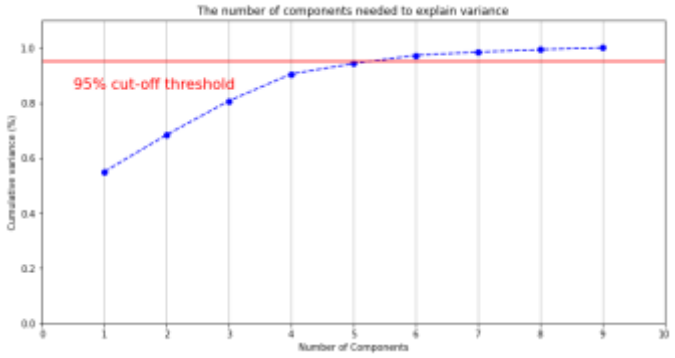
2) Reduksi dimensi dengan PCA

Sebelum mereduksi dimensi dengan PCA, perlu dilakukan untuk menentukan banyaknya komponen PCA terbaik. Hal ini bisa dilakukan dnegan menghitung variansi dari PCA untuk tiap fitur.

```
[34] pca = PCA()
     pca_features = pca.fit_transform(x)

     print('Shape before PCA: ', x.shape)
     print('Shape after PCA: ', pca_features.shape)

     _,n = pca_features.shape

     columnsPCA = []
     for i in range(1,n+1):
       colName = "PC"+str(i)
       columnsPCA.append(colName)
     print(columnsPCA)

     pca_df = pd.DataFrame(
         data=pca_features,
         columns=columnsPCA)

     Shape before PCA:  (167, 9)
     Shape after PCA:  (167, 9)
     ['PC1', 'PC2', 'PC3', 'PC4', 'PC5', 'PC6', 'PC7', 'PC8', 'PC9']

[35] pca.explained_variance_

     array([0.14180615, 0.03450913, 0.03171502, 0.02513534, 0.00974048,
            0.00776992, 0.00386022, 0.00228894, 0.00178941])
```

```
plt.rcParams['figure.figsize'] = (12,6)

fig, ax = plt.subplots()
xi = np.arange(1, n+1, step=1)
y = np.cumsum(pca.explained_variance_ratio_)

plt.ylim(0.0,1.1)
plt.plot(xi, y, marker='o', linestyle='--', color='b')

plt.xlabel('Number of Components')
plt.xticks(np.arange(0, 11, step=1)) #change from 0-based array index to 1-based human-readable label
plt.ylabel('Cumulative variance (%)')
plt.title('The number of components needed to explain variance')

plt.axhline(y=0.95, color='r', linestyle='-')
plt.text(0.5, 0.85, '95% cut-off threshold', color = 'red', fontsize=16)

ax.grid(axis='x')
plt.show()
```

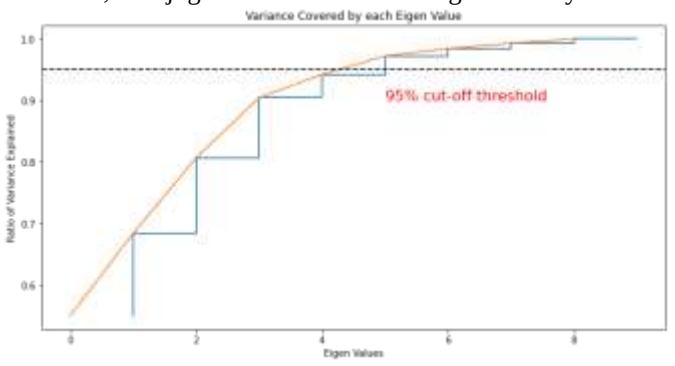
Setelah itu kita memanggil diagram untuk melihat bagaiman variansi untuk tiap komponen



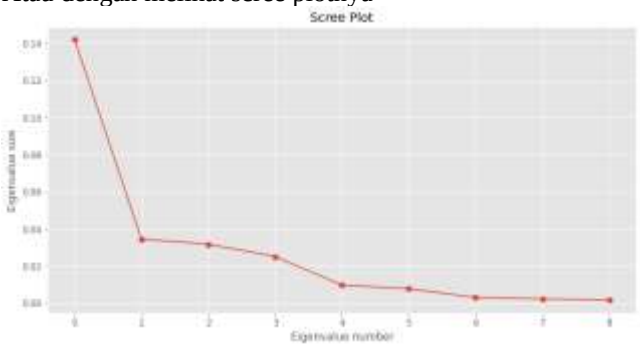
Dapat dilihat bahwa minimal 6 komponen yang dibutuhkan untuk mempertahankan 95% variansi.

```
plt.step(list(range(1,n+1)), np.cumsum(pca.explained_variance_ratio_))
plt.plot(np.cumsum(pca.explained_variance_ratio_))
plt.xlabel('Eigen Values')
plt.ylabel('Ratio of Variance Explained')
plt.title('Variance Covered by each Eigen Value')

plt.axhline(y=0.95, color='black', linestyle='dashed')
plt.text(5, 0.90, '95% cut-off threshold', color = 'red', fontsize=16)

plt.show()
```

Selain itu, bisa juga dilihat berdasarkan eigen valuenya.



Atau dengan melihat scree plotnya



Semua plot di atas menyatakan bahwa minimal 6 komponen yang kita perlukan untuk PCA nantinya.

```
pca = PCA(n_components=6)
pca_features = pca.fit_transform(x)

print('Shape before PCA: ', x.shape)
print('Shape after PCA: ', pca_features.shape)

_,n = pca_features.shape

columnsPCA = []
for i in range(1,n+1):
  colName = "PC"+str(i)
  columnsPCA.append(colName)
print(columnsPCA)

pca_df = pd.DataFrame(
    data=pca_features,
    columns=columnsPCA)
```

```
Shape before PCA:  (167, 9)
Shape after PCA:  (167, 6)
['PC1', 'PC2', 'PC3', 'PC4', 'PC5', 'PC6']
```

Penerapan 6 komponen PCA

Hasilnya:

pca_df

	PC1	PC2	PC3	PC4	PC5	PC6
0	-0.599078	0.095490	0.157554	-0.024333	-0.045618	-0.046532
1	0.158474	-0.212092	-0.064189	-0.061247	0.014191	-0.010246
2	0.003686	-0.135867	-0.134182	0.133574	-0.091150	0.025988
3	-0.650235	0.275975	-0.142672	0.156018	-0.081997	0.032170
4	0.200711	-0.064662	-0.100715	-0.037902	-0.035799	-0.055817
...	...	...	...	...	...	...
162	-0.160078	-0.029625	-0.121910	-0.066099	-0.009043	-0.063646
163	0.061133	-0.171339	-0.058586	0.247460	-0.093260	0.291515
164	0.115512	-0.032034	-0.195243	-0.231993	-0.035734	0.089237
165	-0.332968	-0.019824	-0.029989	0.105416	-0.141550	0.063254
166	-0.573897	0.108788	0.032552	0.044839	0.024088	0.013233

167 rows × 6 columns

E. Feature Selection dengan Korelasi dan treshold 0.75

Reduksi dimensi dapat juga dilihat dari korelasinya dengan kode sebagai berikut:

```
def select_feature(df, threshold):
    corr = set()
    cor_matrix = df.corr()
    for i in range(len(cor_matrix.columns)):
        for j in range(i):
            if abs(cor_matrix.iloc[i, j]) > threshold:
                colname = cor_matrix.columns[i]
                corr.add(colname)
    return corr
```

```
corr_features = select_feature(df, 0.75)
corr_features, len(corr_features)
```

```
({'gdpp', 'life_expec', 'total_fer'}, 3)
```

```
df_selected = df[corr_features]
```

```
df_selected.insert(0, "country", df['country'])
```

Hasilnya:

df_selected

	country	life_expec	total_fer	gdpp
0	Afghanistan	56.2	5.82	553
1	Albania	76.3	1.65	4090
2	Algeria	76.5	2.89	4460
3	Angola	60.1	6.16	3530
4	Antigua and Barbuda	76.8	2.13	12200
...	...	...	...	...
162	Vanuatu	63.0	3.50	2970
163	Venezuela	75.4	2.47	13500
164	Vietnam	73.1	1.95	1310
165	Yemen	67.5	4.67	1310
166	Zambia	52.0	5.40	1460

167 rows × 4 columns

F. Scaling

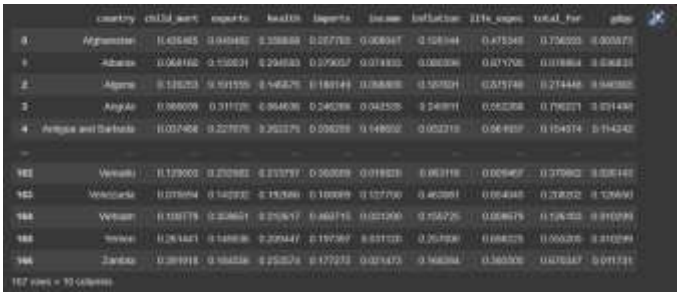
Kodenya:

```
#normalisasi
def minMaxScaling(df) :
    return (df.iloc[:,1:] - df.iloc[:,1:].min()) / (df.iloc[:,1:].max() - df.iloc[:,1:].min())
```

1) Scaling dataset awal

```
#normalisasi
def minMaxScaling(df) :
    return (df.iloc[:,1:] - df.iloc[:,1:].min()) / (df.iloc[:,1:].max() - df.iloc[:,1:].min())

df_normalized = minMaxScaling(df)

df_normalized.insert(0, "country", df['country'])
df_normalized
```

Hasil



2) Pembagian Literal berdasarkan kategori di duna nyata

Hasil scaling:

	country	Health	Economics	Moneter
0	Afghanistan	0.625740	0.139614	0.079820
1	Albania	0.127451	0.199901	0.088756
2	Algeria	0.182485	0.186622	0.212808
3	Angola	0.661381	0.283058	0.236946
4	Antigua and Barbuda	0.116409	0.275189	0.145043
...	...	...	...	...
162	Vanuatu	0.217274	0.262886	0.035009
163	Venezuela	0.133337	0.124193	0.559740
164	Vietnam	0.170070	0.403386	0.121436
165	Yemen	0.397451	0.170248	0.229632
166	Zambia	0.526909	0.181405	0.134997

167 rows × 4 columns

3) Scaling hasil dari feature selection

Hasilnya:

```
df_selected_normalized.insert(0, "country", df['country'])
df_selected_normalized
```

	country	life_expec	total_fer	gdpp
0	Afghanistan	0.475345	0.736593	0.003073
1	Albania	0.871795	0.078864	0.036833
2	Algeria	0.875740	0.274448	0.040365
3	Angola	0.552268	0.790221	0.031488
4	Antigua and Barbuda	0.881657	0.154574	0.114242
...	...	...	...	...
162	Vanuatu	0.609467	0.370662	0.026143
163	Venezuela	0.854043	0.208202	0.126650
164	Vietnam	0.808679	0.126183	0.010299
165	Yemen	0.698225	0.555205	0.010299
166	Zambia	0.392505	0.670347	0.011731

167 rows × 4 columns

V. PENERAPAN ALGORITMA K-MEANS

Untuk algoritma di bawah, terinspirasi dari video berikut: https://www.youtube.com/watch?v=lX-3nGHDhQg&t=581s dan beberapa artikel di internet.

1) Perhitungan centroid awal secara random

Perhitungan centroid awal secara random

```
[50] def random_centroid(k, n, df):
        df_copy = df.copy()
        if 'country' in df:
          df_copy.pop('country')
        # k adalah banyaknya klustar
        # n adalah banyaknya fitur
        return pd.DataFrame(np.random.rand(n,k),index = [i for i in df_copy])
```

Di sini untuk setiap fitur akan dihitung centroidnya sebanyak banyak cluster. Nilai tersebut teridir atas nilai acak dari [0,1]

Contoh hasilnya adalah sebagai berikut.

```
print(random_centroid(3,9,df_normalized))
```

	0	1	2
child_mort	0.861499	0.272030	0.387913
exports	0.003187	0.510625	0.695391
health	0.925311	0.575251	0.856366
imports	0.333263	0.113942	0.094918
income	0.066344	0.986976	0.930331
inflation	0.946466	0.333323	0.523187
life_expec	0.029486	0.326047	0.503277
total_fer	0.319384	0.981915	0.269904
gdpp	0.572526	0.333178	0.302319

2) Perhitungan jarak dan pengelompokan

```
def set_class(data, data_centroids):
  distances = data_centroids.apply(lambda x: np.sqrt(((data - x)**2).sum(axis=1)))
  return distances.idxmin(axis=1) # mencari index dari minimum value tiap baris
```

Di sini dilakukan perhitungan jarak degan Euclidean. Setelah itu dijumlahkan untuk mendapatkan jarak dari setiap datapoint ke ke setiap centroid. Setelah itu untuk setiap datapoint akan dihiutng yang mana yang memiliki jarak terpendek untuk setiap centroid. Jarak terpendek itulah yang menajdi class sementara dari datapoint tersebut.

3) Mengupdate nilai centroid

```
def set_new_centroids(df, labels):
  # mencari ln dari tiap baris kemudian mencari rata-ratanya dan lalu diekspoenkan
  return df.groupby(labels).apply(lambda x : np.exp(np.log(x).mean())).T
```

Untuk mendapatkan centroid yang baru, maka perlu dicari untuk setiap data point yang memiliki class yang sama dengan suatu centroid, kemudian dicari rata-ratanya untuk setiap feature.

```
def plot(df, label, centroid, iterasi):
    pca = PCA(n_components=2)
    df_2d = pca.fit_transform(df)
    centroid_2d = pca.fit_transform(centroid.T)
    plt.title(f'Iterasi ke {iterasi}')

    plt.scatter(x = df_2d[:,0], y = df_2d[:,1], c = label)
    plt.scatter(x = centroid_2d[:,0], y = centroid_2d[:,1])
    plt.show()
```

Setelah itu terdapat scatter plot yang akan menampikan titik-titik dan pembagian kategori untuk setiap datapoint.

Main program dari k-means

```
def k_means1(df):
  df_asli = df.copy()
  df_asli_tanpa_country = df.copy()
  if 'country' in df_asli_tanpa_country.columns:
    df_asli_tanpa_country.pop('country')
    _, n = df_asli_tanpa_country.shape

  max_iterations = 100
  k = 3
  centroids = random_centroid(k,n,df_asli)
  old_centroids = pd.DataFrame()

  iteration = 1
  while iteration < max_iterations and not centroids.equals(old_centroids):
    old_centroids = centroids

    labels = set_class(df_asli_tanpa_country, centroids)
    centroids = set_new_centroids(df_asli_tanpa_country, labels)
    plot(df_asli_tanpa_country, labels, centroids, iteration)
    iteration += 1
  return labels
```

Di sini ditentukan bahwa banyak k adalah 3 dan max iterasi adalah 100. Setelah itu memanggil fungsi-fungsi yang ada di atas secara berurutan sesuai proses dari k-means algorithm.

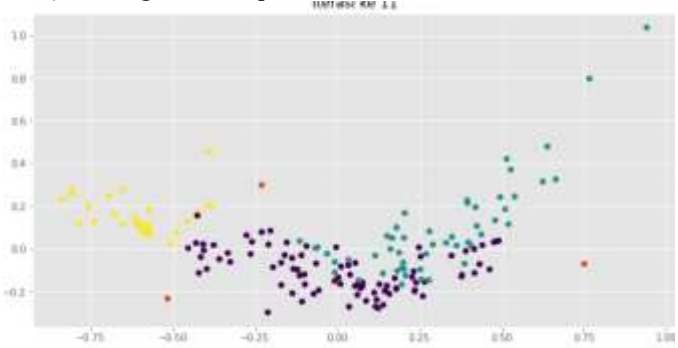
VI. HASIL

Untuk plot negara didapatkan dari: https://www.kaggle.com/code/tanmay111999/clustering-pca-k-means-dbscan-hierarchical

Kode plot negara

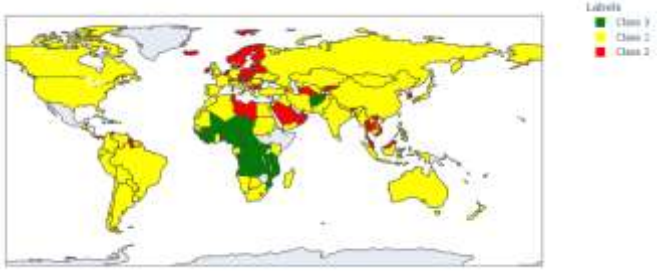
```
def plot_negara(df_plot,labels):
  df_plot['Class'] = labels

  df_plot['Class'].loc[df_plot['Class'] == 0] = 'Class 1'
  df_plot['Class'].loc[df_plot['Class'] == 1] = 'Class 2'
  df_plot['Class'].loc[df_plot['Class'] == 2] = 'Class 3'

  fig = px.choropleth(df_plot[['country','Class']],
                      locationmode = 'country names',
                      locations = 'country',
                      title = 'Pentagon',
                      color = df_plot['Class'],
                      color_discrete_map = {'Class 1':'Red',
                                            'Class 2':'Yellow',
                                            'Class 3': 'Green'})
  fig.update_geos(fitbounds = "locations", visible = True)
  fig.update_layout(legend_title_text = 'Labels',legend_title_side = 'top',title_pad_l = 200,title_y = 0.93)
  fig.show(engine = 'kaleido')
```

1) *Scaling biasa tanpa reduksi dimensi*

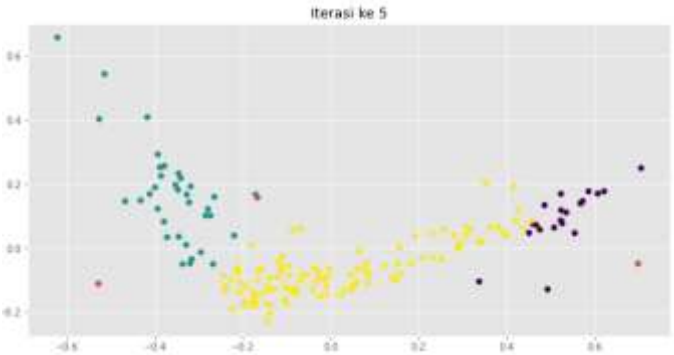


```
labels_normalized.value_counts()
```

```
0    87
1    52
2    28
dtype: int64
```



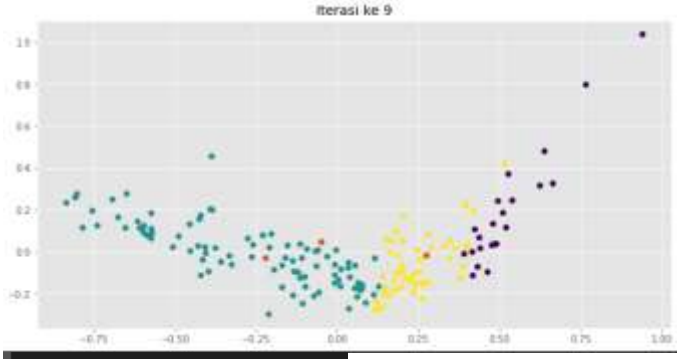
2) *Hasil dari feature selection dengan korelasi dan treshold 0.75*




```
labels_feature_selected.value_counts()
```

```
2    112
1     35
0     20
dtype: int64
```

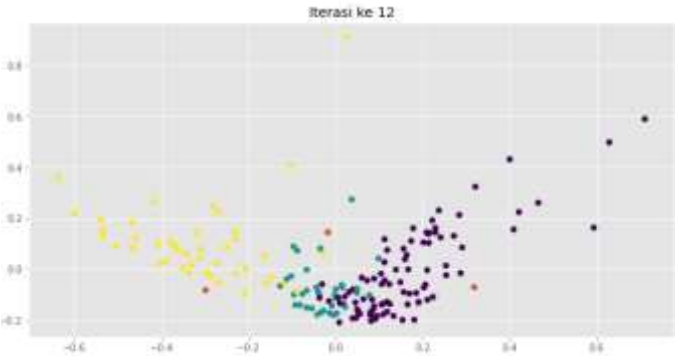
3) *Hasil dari df pca*




```
labels_pca.value_counts()
```

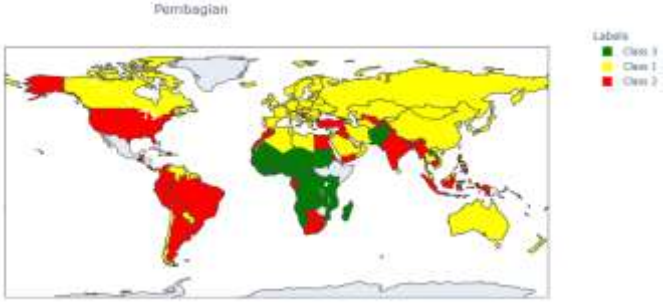
```
1    92
2    53
0    22
dtype: int64
```

4) *Hasil dari pembagian literal dunia nyata*




```
labels_literal.value_counts()
```

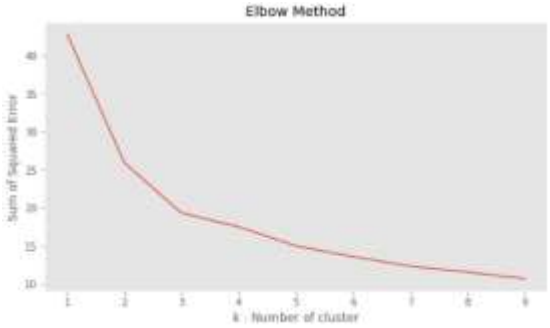
```
0    89
2    43
1    35
dtype: int64
```

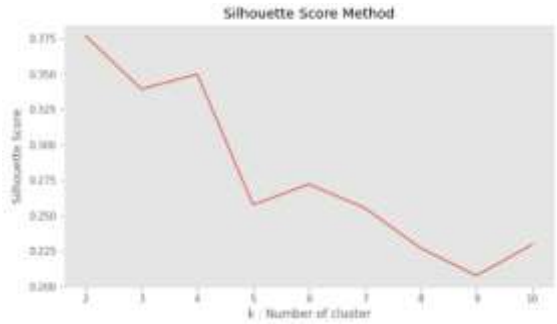


VII. EVALUASI

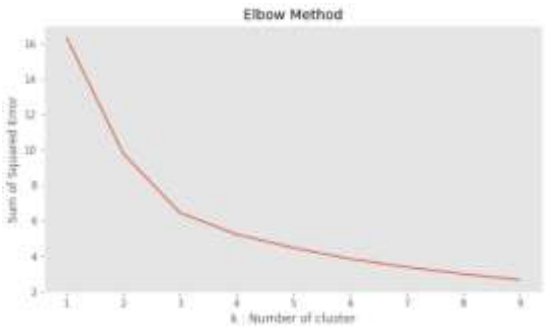
```
from sklearn.cluster import KMeans
from sklearn.metrics import silhouette_score

def evaluate(df):
  df_test = df.copy()
  sse = {};sil = [];kmax = 10
  fig = plt.subplots(nrows = 1, ncols = 2, figsize = (10,5))

  if 'Class' in df_test.columns:
    df_test.pop('Class')
  if 'country' in df_test.columns:
    df_test.pop('country')

  # Elbow Method :
  plt.subplot(1,2,1)
  for k in range(1, 10):
      kmeans = KMeans(n_clusters=k, max_iter=1000).fit(df_test)
      sse[k] = kmeans.inertia_ # Inertia: Sum of distances of samples to their closest cluster center
  sns.lineplot(x = list(sse.keys()), y = list(sse.values()));
  plt.title('Elbow Method')
  plt.xlabel("k : Number of cluster")
  plt.ylabel("Sum of Squared Error")
  plt.grid()

  # Silhouette Score Method
  plt.subplot(1,2,2)
  for k in range(2, kmax + 1):
      kmeans = KMeans(n_clusters = k).fit(df_test)
      labels = kmeans.labels_
      sil.append(silhouette_score(df_test, labels, metric = 'euclidean'))
  sns.lineplot(x = range(2,kmax + 1), y = sil);
  plt.title('Silhouette Score Method')
  plt.xlabel("k : Number of cluster")
  plt.ylabel("Silhouette Score")
  plt.grid()
```

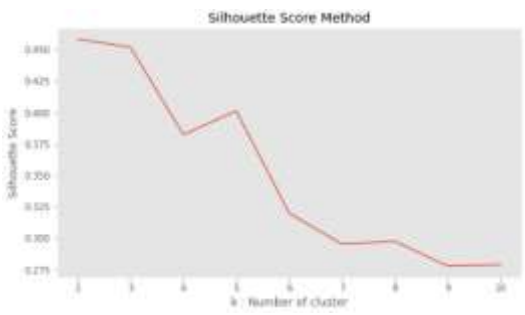
Untuk normalisasi biasa



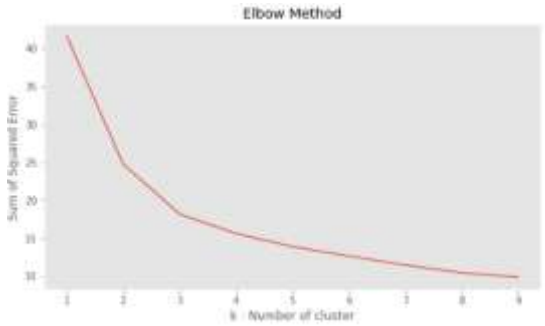


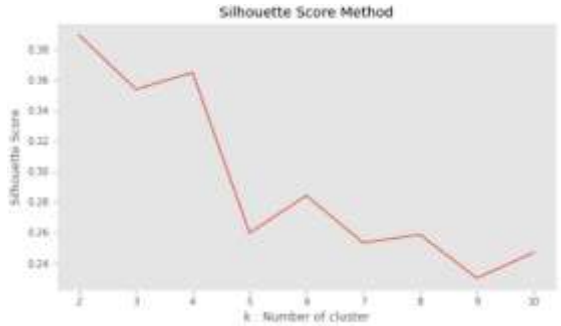
Untuk reduksi berdasarkan kelompok di dunia nyata



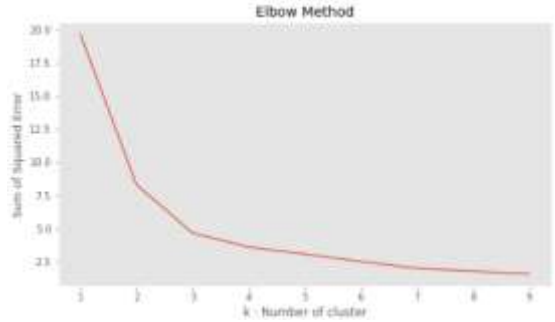


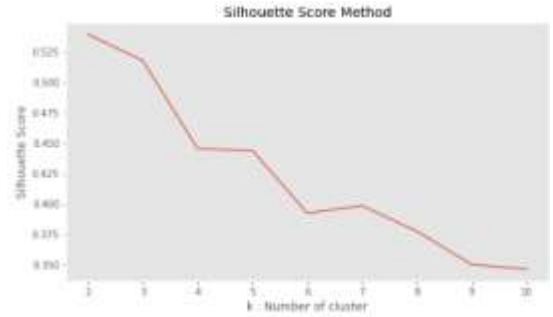
Untuk pca





Untuk reduksi berdasarkan korelasi





Dapat dilihat bahwa untuk setiap feature selection atau engineering yang dilakukan semua menghasilkan bahwa k = 3 adalah yang terbaik. Sehingga pengecekan tidak perlu diulangi kembali

VIDEO LINK AT YOUTUBE

https://youtu.be/3QEIDAaY9TE

LINK COLAB

https://colab.research.google.com/drive/1DJ3JAHpmWFva9g2v-z4BMsnGqvtbbL7a?usp=sharing

LINK GITHUB

https://github.com/khalilullahalfaath/case_based_2_ML

KESIMPULAN

Dengan Algoritma k-means clustering kita sudah dapat menentukan negara mana saja yang memerlukan bantuin, mungkin membutuhkan bantuin, dan tidak membutuhkan bantuin. Selain karena mudah diimplementasikan k-means juga cocok untuk data yang sedikit dan cepat dalam komputasinya.

REFERENCES

[1] Miftah Rezkia. (2020, December 14). Mengenal Lebih Dalam Algoritma Unsupervised Learning. Dqlab. Retrieved December 4, 2022, from https://www.dqlab.id/mengenal-leboh-dalam-algoritma-unsupervised-learning

[2] https://www.youtube.com/watch?v=lX-3nGHDhQg&t=581s

[3] https://www.kaggle.com/code/tanmay111999/clustering-pca-k-means-dbscan-hierarchical

[4] Pandas documentation

[5] Numpy documentation

[6] Sklearn documentation

[7] Matlplotlib dan seaborn documtation

PERNYATAAN

Dengan ini saya menyatakan bahwa laporan yang saya tulis ini adalah tulisan saya sendiri, bukan saduran, atau terjemahan dari makalah orang lain, dan bukan plagiasi.

Bandung, 12 Desember 2022
Khalilullah Al Faath (1301204376)