KELOMPOK 1 UTS MACHINE LEARNING 1 KELAS DS-01-01

- RAHMAT SIGIT HIDAYAT (1206210005)
- HALIM ARIF CAHYONO
- ASTIKHATUL MUFAIDAH
- ANNAS THASYA HAAFIDZAH S

Link Repository Github : MachineLearning1_UTS_Kelompok1

Introduction

Unsupervised learning atau pembelajaran tak terarah adalah teknik pembelajaran mesin di mana pengguna tidak perlu mengawasi modelnya. Sebaliknya, pembelajaran tak terarah memungkinkan model bekerja sendiri untuk menemukan pola dan informasi yang sebelumnya tidak terdeteksi, terutama yang berkaitan dengan data yang tidak berlabel.

Ada beberapa jenis unsupervised learning yang umum digunakan dalam machine learning. Dua jenis utama adalah clustering (pengelompokan) dan dimensionality reduction (pengurangan dimensi). Berikut adalah penjelasan singkat tentang keduanya:

1. Exclusive (Partitioning) Dalam metode clustering yang satu ini, data dikelompokkan sedemikian rupa sehingga satu data hanya dapat dimiliki oleh satu kluster saja.

Contoh: K-means

1. Agglomerative Dalam teknik clustering ini, setiap data adalah sebuah kluster. Penggabungan berulang antara dua kluster terdekat mengurangi jumlah kluster.

Contoh: Hierarchical cluster

1. Overlapping Dalam teknik ini, himpunan fuzzy digunakan untuk mengelompokkan data. Setiap titik dapat dimiliki oleh dua kluster atau lebih dengan derajat keanggotaan yang berbeda.

Di sini, data akan dikaitkan dengan nilai keanggotaan yang sesuai.

Contoh: Fuzzy C-Means

1. Probabilistic Teknik ini menggunakan distribusi probabilitas untuk membuat cluster

Contoh: Keyword di bawah ini

“sepatu pria.” “sepatu wanita.” “sarung tangan wanita.” “sarung tangan pria.” dapat dikelompokkan menjadi dua kategori “sepatu” dan “sarung tangan” atau “pria” dan “wanita.”

1. Association Aturan asosiasi memungkinkan kamu untuk membuat asosiasi di antara objek data di dalam database besar. Teknik pembelajaran tak terarah ini adalah tentang menemukan hubungan yang menarik antara variabel dalam database besar. Misalnya, orang yang membeli rumah baru kemungkinan besar akan membeli perabotan baru.

Contoh lain:

Subkelompok pasien kanker yang dikelompokkan berdasarkan pengukuran ekspresi gen mereka Kelompok pembelanja berdasarkan riwayat penelusuran dan pembelian mereka Kelompok film berdasarkan peringkat yang diberikan oleh penonton film

PCA

PCA adalah teknik statistik yang bertujuan untuk mengurangi dimensi dataset sambil mempertahankan struktur dan informasi esensialnya. Dengan memproyeksikan data dimensi tinggi ke subruang dimensi rendah, PCA membantu mengungkap fitur dan [pola paling signifikan yang ada dalam data. Principal Component Analysis (PCA) memiliki beberapa kegunaan penting dalam analisis data dan machine learning, terutama dalam konteks pengurangan dimensi. Berikut adalah beberapa kegunaan utama PCA:

1) Pengurangan Dimensi: Efisiensi Representasi Data: PCA membantu mengidentifikasi arah-arah utama variasi dalam data. Dengan memilih komponen utama yang paling penting, PCA memungkinkan kita merepresentasikan data dalam dimensi yang lebih rendah tanpa kehilangan sebagian besar informasi.

2) Visualisasi Data: Visualisasi Ruang Dimensi Tinggi: PCA memungkinkan visualisasi data dalam ruang dimensi yang lebih rendah (biasanya dua atau tiga dimensi) sehingga kita dapat memahami struktur dan pola data dengan lebih baik. Ini terutama berguna ketika kita memiliki data dengan banyak fitur yang sulit untuk divisualisasikan.

3) Reduksi Noise: Pengurangan Efek Noise: Komponen utama dalam PCA biasanya lebih terkait dengan variasi yang signifikan dalam data, sementara komponen yang kurang penting dapat dianggap sebagai noise. Dengan demikian, PCA dapat membantu mengurangi efek noise dalam data.

4) Praproses Data: Praproses Sebagai Langkah Awal: PCA sering digunakan sebagai langkah praproses sebelum menerapkan algoritma machine learning lainnya. Dengan mengurangi dimensi data, PCA dapat membantu meningkatkan kinerja algoritma dan mengurangi risiko overfitting.

5) Identifikasi Fitur Penting: Identifikasi Fitur Penting: PCA membantu mengidentifikasi fitur-fitur yang memberikan kontribusi signifikan terhadap variasi dalam data. Ini dapat berguna dalam pemilihan fitur, di mana kita dapat fokus pada fitur-fitur yang paling informatif.

6) Analisis Hubungan Antar Variabel: Analisis Korelasi Antar Variabel: PCA dapat membantu mengidentifikasi korelasi dan hubungan antar variabel dalam data. Ini dapat memberikan wawasan tentang bagaimana variabel-variabel tersebut berinteraksi satu sama lain.

7) Ekstraksi Fitur: Ekstraksi Fitur: PCA dapat digunakan untuk menghasilkan fitur-fitur baru yang merupakan kombinasi linear dari fitur-fitur asli. Fitur-fitur ini dapat digunakan sebagai input untuk model machine learning, dan mereka dapat mencerminkan aspek-aspek penting dari data.

Langkah-langkah utama dalam PCA adalah sebagai berikut:

1. Data Preprocessing: ini dimulai dengan standardisasi atau normalisasi dataset untuk memastikan bahwa setiap fitur berkontribusi secara proporsional terhadap analisis. Langkah ini melibatkan pemusatan rata-rata dan penskalaan data agar memiliki rata-rata nol dan varian satuan.

2. Perhitungan Matriks Kovarian: Langkah selanjutnya melibatkan komputasi matriks kovarians, yang mengukur hubungan antara pasangan variabel dalam kumpulan data. Matriks kovarian memberikan wawasan tentang kekuatan dan arah hubungan linier antar variabel.

3. Dekomposisi Nilai Eigen: Pada langkah ini, matriks kovarian didekomposisi menjadi vektor eigen dan nilai eigennya. Vektor eigen mewakili arah di mana data paling bervariasi, sedangkan nilai eigen menghitung jumlah varians yang dijelaskan oleh masing-masing vektor eigen.

4. Pemilihan Komponen Utama: Vektor eigen dengan nilai eigen tertinggi, dikenal sebagai komponen utama, menangkap variasi paling signifikan dalam data. Komponen utama ini membentuk sistem koordinat baru yang dapat secara efektif merepresentasikan data asli dalam ruang berdimensi lebih rendah.

5. Pengurangan Dimensi: Akhirnya, kumpulan data asli diproyeksikan ke komponen utama yang dipilih, sehingga mengurangi dimensinya. Jumlah komponen utama yang dipertahankan menentukan dimensi dari kumpulan data yang diubah.

- Unsupervise Learning : https://greatnusa.com/artikel/unsupervised-learning-adalah/
- PCA : https://lp2m.uma.ac.id/2023/06/14/principal-component-analysis-pca-definisi-dan-penjelasannya/

Klasterisasi/Clustering

Klasterisasi adalah metode pengelompokan data yang bertujuan untuk mengelompokkan data yang serupa ke dalam satu kelompok dan data yang berbeda ke dalam kelompok yang berbeda pula[1]. Klasterisasi berguna untuk memahami struktur data, mengidentifikasi pola, dan mempermudah analisis data[3]. Jenis-jenis klasterisasi antara lain:

Hierarchical Clustering : Metode klasterisasi yang mengelompokkan data secara bertahap dari kelompok yang lebih kecil ke kelompok yang lebih besar[3].

- Partitioning Clustering : Metode klasterisasi yang membagi data ke dalam kelompok-kelompok yang saling tidak tumpang tindih[3].
- Density-Based Clustering : Metode klasterisasi yang mengelompokkan data berdasarkan kepadatan data[3].

K-means Clustering, merupakan satu algoritma klasterisasi yang paling populer[2]. Algoritma ini bertujuan untuk membagi data ke dalam kelompok-kelompok yang saling tidak tumpang tindih, di mana setiap kelompok memiliki pusat kelompok (centroid) yang merupakan rata-rata dari data di dalam kelompok tersebut[1]. Langkah-langkah dalam K-means Clustering adalah sebagai berikut[5]:

- Tentukan jumlah kelompok (K) yang diinginkan.
- Pilih K titik acak sebagai pusat kelompok awal.
- Hitung jarak antara setiap data dengan setiap pusat kelompok.
- Masukkan setiap data ke dalam kelompok dengan pusat kelompok terdekat.
- Hitung ulang pusat kelompok untuk setiap kelompok.

- Ulangi langkah 3-5 hingga pusat kelompok tidak berubah atau iterasi telah mencapai batas tertentu.

K-means Clustering cocok digunakan untuk mengelompokkan data yang tidak memiliki label atau informasi kelompok sebelumnya[1]. Algoritma ini telah digunakan dalam berbagai bidang, seperti segmentasi pasar, pengolahan citra, dan astronomi[2]. Namun, K-means Clustering sensitif terhadap data yang berbeda-beda dan dapat memberikan hasil yang berbeda-beda jika data diubah sedikit saja[6]. Oleh karena itu, perlu dilakukan evaluasi hasil klasterisasi secara cermat.

Sumber:

- [1] https://statisticsbyjim.com/basics/k-means-clustering/
- [2] https://en.wikipedia.org/wiki/K-means_clustering
- [3] https://www.simplilearn.com/tutorials/machine-learning-tutorial/k-means-clustering-algorithm
- [4] https://www.edureka.co/blog/k-means-clustering/
- [5]https://towardsdatascience.com/understanding-k-means-clustering-in-machine-learning6a6e67336aa1
- [6] https://neptune.ai/blog/k-means-clustering

Task

1. Gunakan metode-metode unsupervised learning untuk mengolah dataset yang disediakan
2. Lakukan analisis sekereatif mungkin terkait dataset dan model.
3. Berikan bentuk-bentuk visual terkait data dan model yang dihasilkan.

Data Preparation

Import Library

```
import pandas as pd
import numpy as np
import seaborn as sns

import matplotlib.pyplot as plt
from matplotlib.colors import ListedColormap

from sklearn.cluster import KMeans
from sklearn.decomposition import PCA
from sklearn.preprocessing import LabelEncoder
from sklearn.preprocessing import MinMaxScaler
from sklearn.preprocessing import StandardScaler
from sklearn import metrics
from sklearn.metrics import confusion_matrix, classification_report
```

```
from sklearn.metrics import accuracy_score
from sklearn.metrics import ConfusionMatrixDisplay

from sklearn.neighbors import KNeighborsClassifier
from sklearn_som.som import SOM
from sklearn.metrics import silhouette_score
from sklearn import datasets

from collections import Counter
from minisom import MiniSom
import pickle
import time

%matplotlib inline
sns.set(color_codes=True)
import warnings
warnings.filterwarnings("ignore")
```

Read Dataset

```
data = pd.read_csv('1. Country-data.csv')
data.head()
```

```
               country  child_mort  exports  health  imports  income

0          Afghanistan        90.2     10.0    7.58     44.9
1610  \
1              Albania        16.6     28.0    6.55     48.6    9930

2              Algeria        27.3     38.4    4.17     31.4   12900

3               Angola       119.0     62.3    2.85     42.9    5900

4  Antigua and Barbuda        10.3     45.5    6.03     58.9   19100


   inflation  life_expec  total_fer   gdpp
0       9.44        56.2       5.82    553
1       4.49        76.3       1.65   4090
2      16.10        76.5       2.89   4460
3      22.40        60.1       6.16   3530
4       1.44        76.8       2.13  12200
```

Metadata

Dataset terdiri dari beberapa atribut mengenai kualitas/penilaian suatu negara. Dengan detail atribut sebagai berikut

- Country: Nama negara yang bersangkutan.

- Child Mortality: Tingkat kematian anak-anak di bawah usia lima tahun per 1.000 kelahiran hidup. Ini dapat mencerminkan kesehatan masyarakat dan akses ke layanan medis.
- Exports: Nilai total ekspor barang dan jasa dari negara tersebut, biasanya diukur dalam dolar.
- Health: Mungkin ini adalah indikator kesehatan masyarakat secara umum, tetapi tanpa satuan atau konteks lebih lanjut, sulit untuk memberikan interpretasi yang pasti.
- Imports: Nilai total impor barang dan jasa ke negara tersebut, juga diukur dalam dolar.
- Income: Tingkat pendapatan per kapita atau pendapatan nasional bruto per penduduk.
- Inflation: Tingkat inflasi, yaitu persentase perubahan harga rata-rata untuk sekelompok barang dan jasa dalam suatu periode waktu tertentu.
- Life Expectancy: Rata-rata usia penduduk pada saat diperkirakan akan meninggal. Ini dapat memberikan gambaran tentang kualitas hidup dan sistem kesehatan.
- Total Fertility Rate: Rata-rata jumlah anak yang akan lahir dari seorang wanita selama hidupnya. Ini dapat menunjukkan kebijakan keluarga dan demografi penduduk.
- GDP: Produk Domestik Bruto, yaitu nilai total semua barang dan jasa yang dihasilkan oleh suatu negara dalam suatu periode waktu tertentu.

Data Preprocessing

Descriptive Statistics

```
data.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

data.describe()

       child_mort     exports      health     imports         income

count  167.000000  167.000000  167.000000  167.000000
```

```
167.000000  \
mean    38.270060   41.108976    6.815689   46.890215   17144.688623

std     40.328931   27.412010    2.746837   24.209589   19278.067698

min      2.600000    0.109000    1.810000    0.065900     609.000000

25%      8.250000   23.800000    4.920000   30.200000    3355.000000

50%     19.300000   35.000000    6.320000   43.300000    9960.000000

75%     62.100000   51.350000    8.600000   58.750000   22800.000000

max    208.000000  200.000000   17.900000  174.000000  125000.000000


        inflation  life_expec   total_fer           gdpp
count  167.000000  167.000000  167.000000     167.000000
mean     7.781832   70.555689    2.947964   12964.155689
std     10.570704    8.893172    1.513848   18328.704809
min     -4.210000   32.100000    1.150000     231.000000
25%      1.810000   65.300000    1.795000    1330.000000
50%      5.390000   73.100000    2.410000    4660.000000
75%     10.750000   76.800000    3.880000   14050.000000
max    104.000000   82.800000    7.490000  105000.000000
```

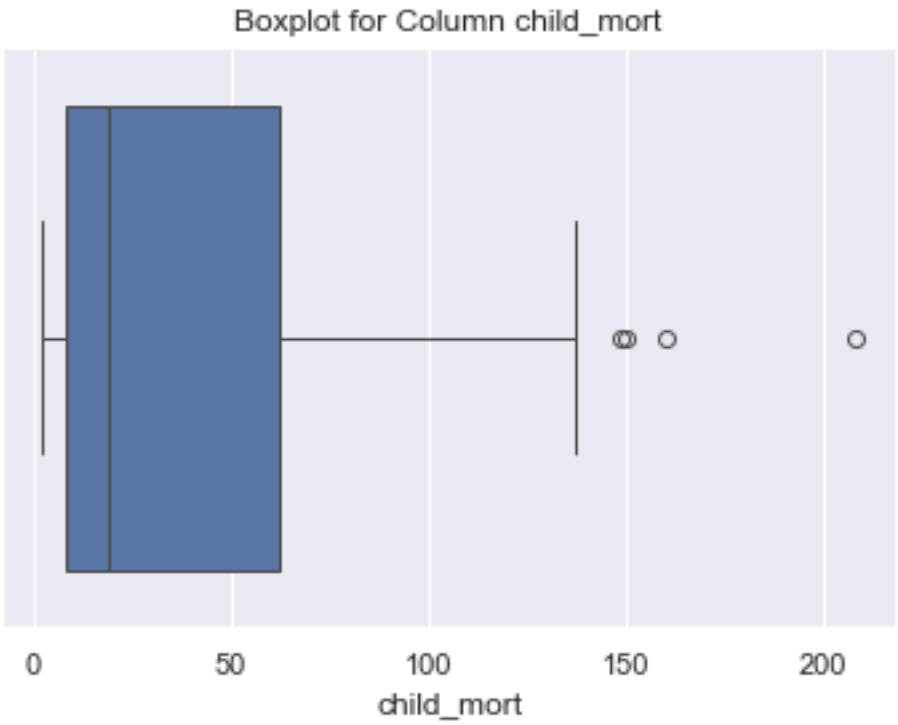
Missing Value

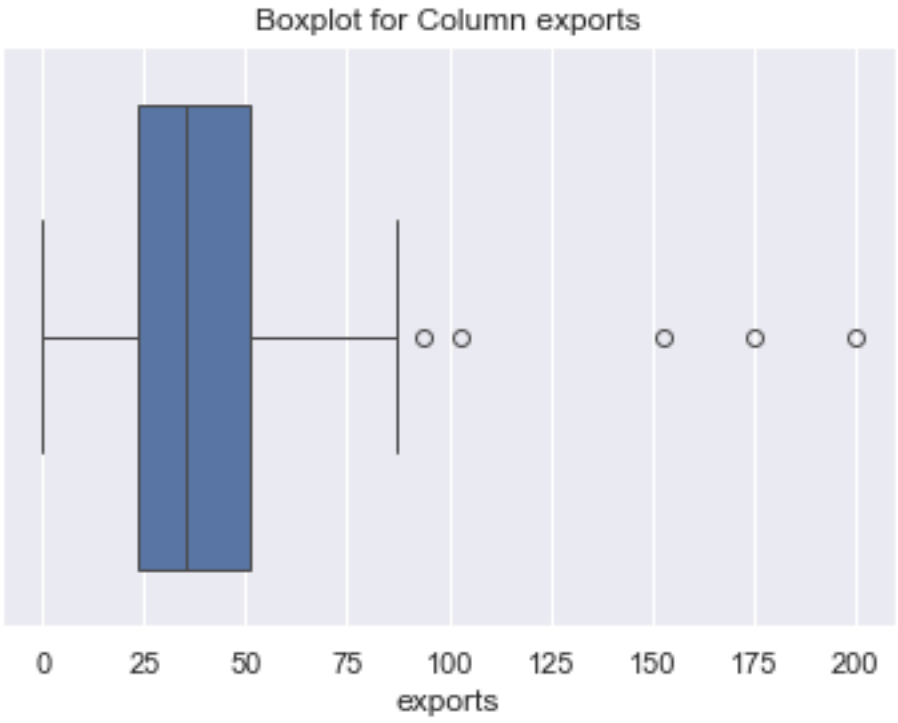
```
data.duplicated().sum() #Cek Duplikasi

0

data.isnull().sum() #Cek Missing Value

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
```

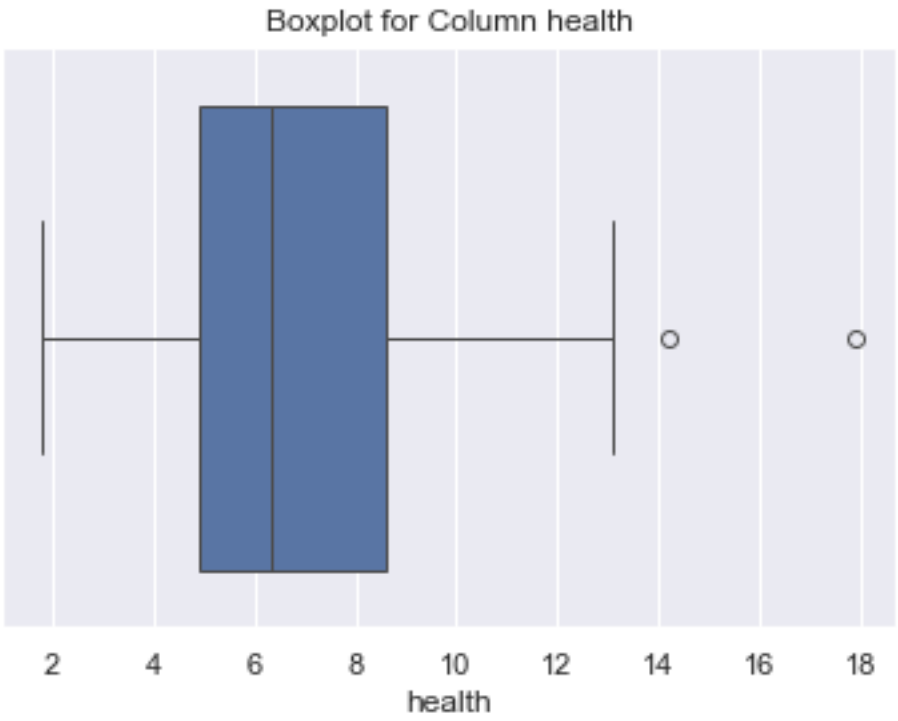
Data Distribution

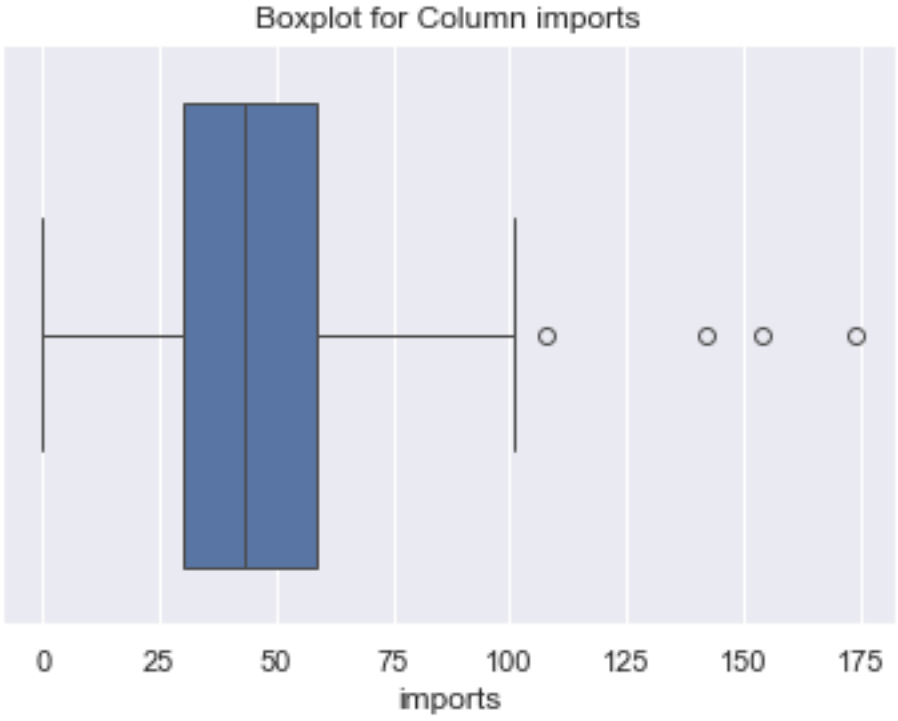
```
data1 = data.iloc[:, 1:]
```

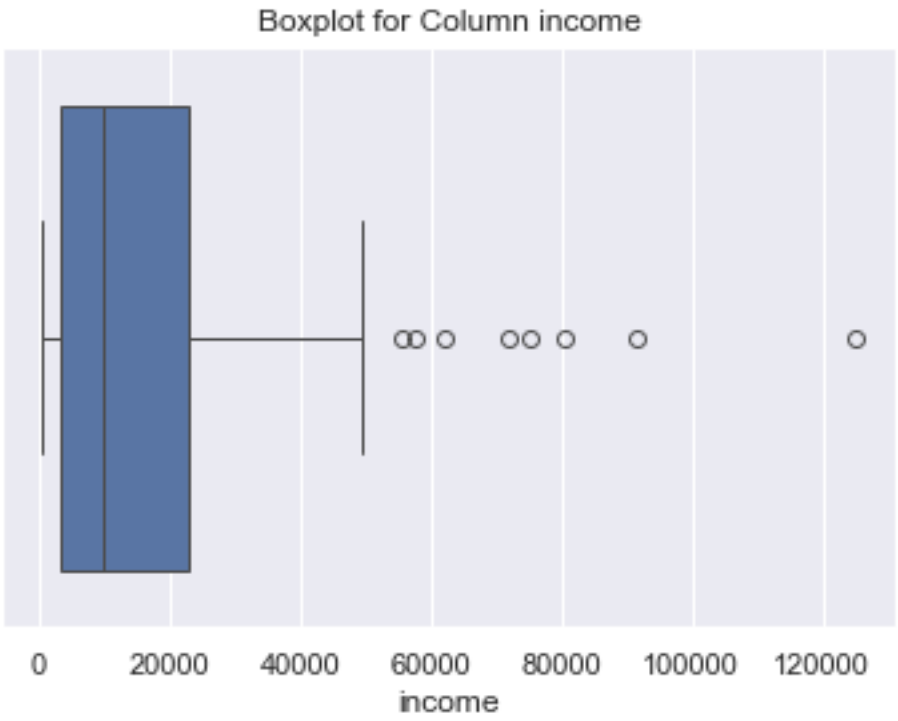
```
for column in data1.columns:
    sns.boxplot(x=data1[column])
    plt.title(f'Boxplot for Column {column}')
    plt.show()
```

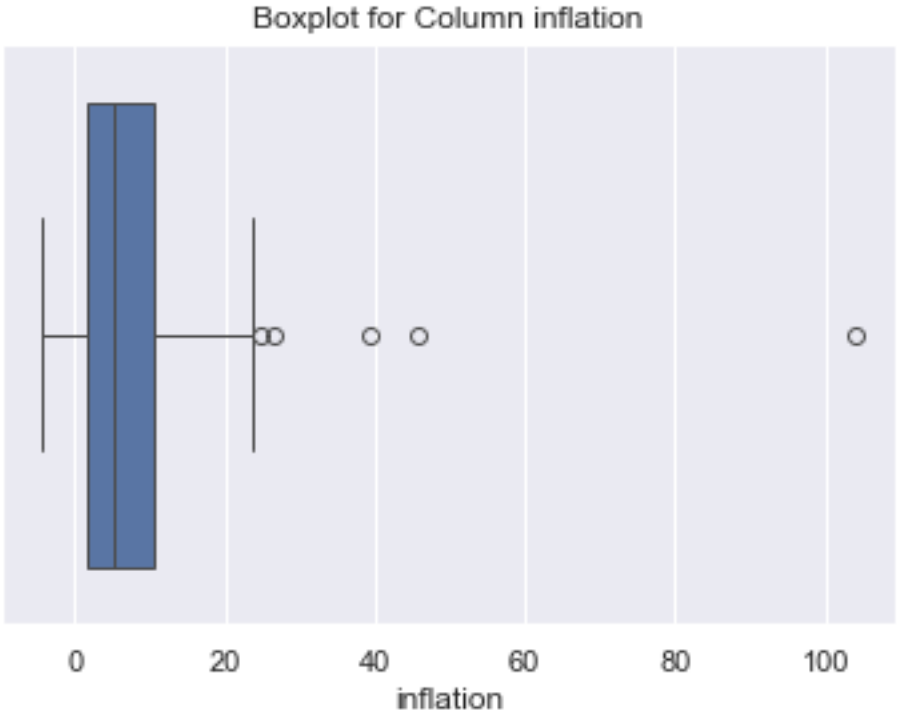


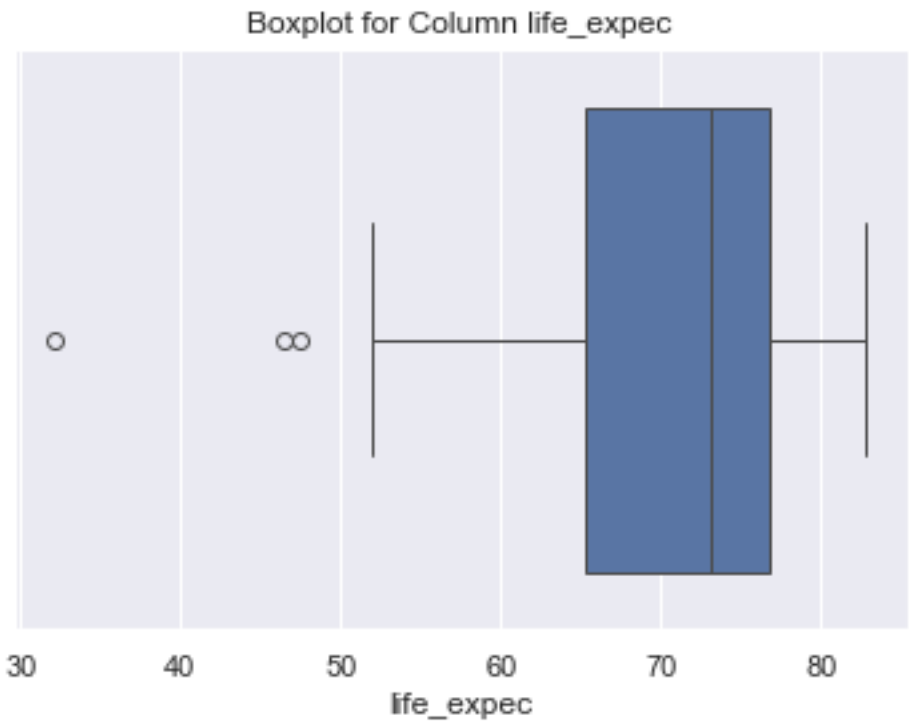


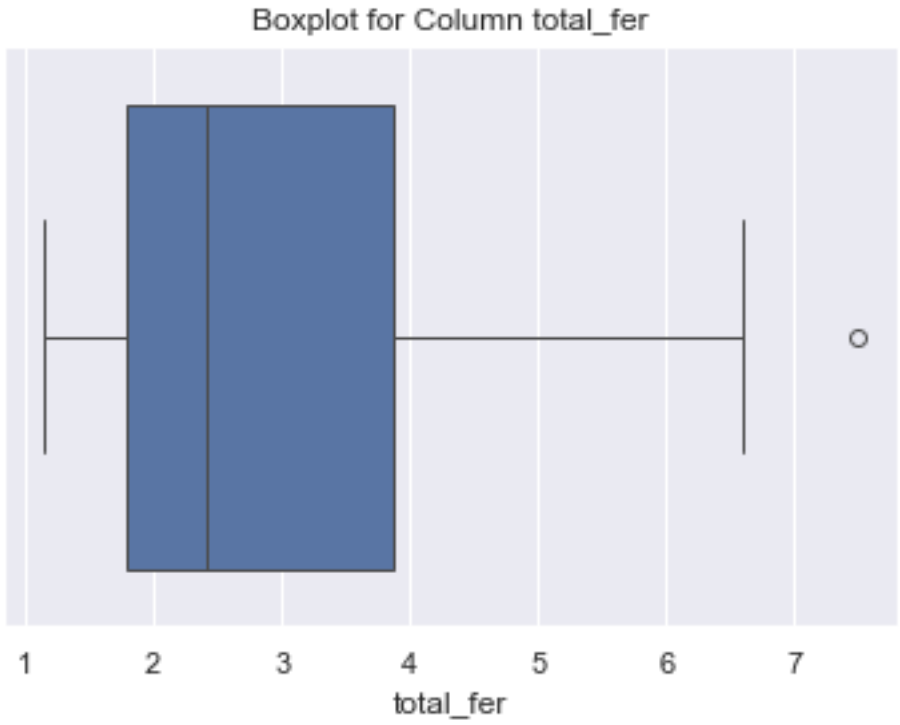

Boxplot for Column health
2
4
6
8
10
12
14
16
18
health

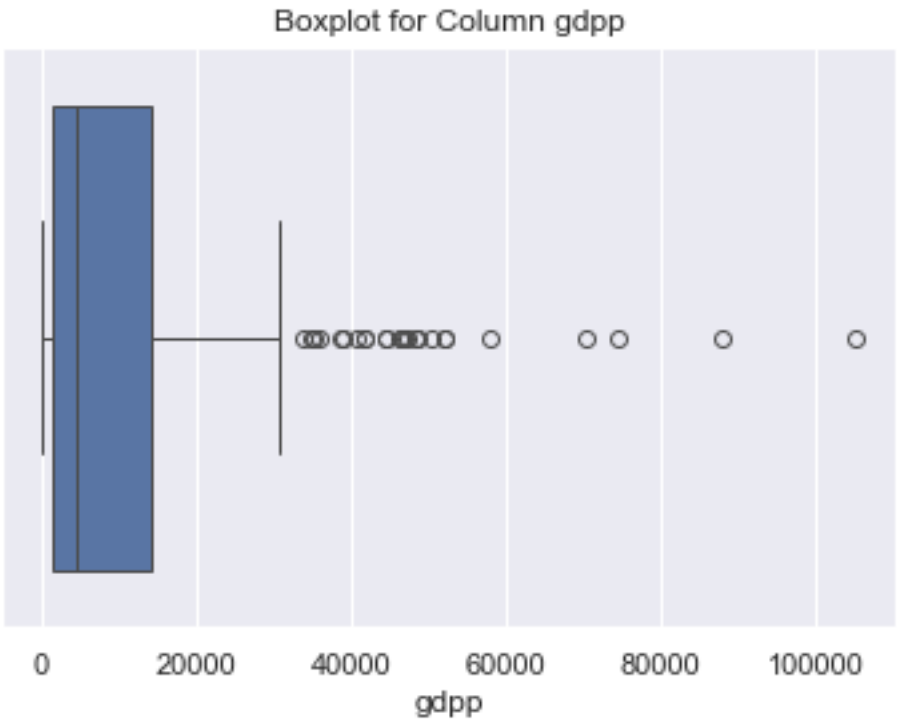

Boxplot for Column imports
0
25
50
75
100
125
150
175
imports


Boxplot for Column income
0
20000
40000
60000
80000
100000
120000
income


Boxplot for Column inflation
0
20
40
60
80
100
inflation


Boxplot for Column life_expec
30
40
50
60
70
80
life_expec


Boxplot for Column total_fer
1
2
3
4
5
6
7
total_fer



Normalisasi/Standarisasi

```
scaler = MinMaxScaler() #Karena distribusi data tidak normal/ada
outlier
data_scaled = pd.DataFrame(scaler.fit_transform(data1),
columns=data1.columns)
data_scaled.head()
```

```
   child_mort   exports    health   imports    income  inflation
life_expec
0    0.426485  0.049482  0.358608  0.257765  0.008047   0.126144
0.475345  \
1    0.068160  0.139531  0.294593  0.279037  0.074933   0.080399
0.871795
2    0.120253  0.191559  0.146675  0.180149  0.098809   0.187691
0.875740
3    0.566699  0.311125  0.064636  0.246266  0.042535   0.245911
0.552268
4    0.037488  0.227079  0.262275  0.338255  0.148652   0.052213
0.881657

   total_fer      gdpp
0   0.736593  0.003073
1   0.078864  0.036833
2   0.274448  0.040365
3   0.790221  0.031488
4   0.154574  0.114242
```

Modelling

PCA

```
X = data_scaled
mean_vec = np.mean(X, axis=0)
cov_mat = (X-mean_vec).T.dot((X-mean_vec))/(X.shape[0]-1)
eig_vals, eig_vecs = np.linalg.eig(cov_mat)
eigen_tpl = [(np.abs(eig_vals[i]), eig_vecs[:, i]) for i in
range(len(eig_vals))]
# Mengurutkan list berdasarkan nilai eigen tertinggi hingga terendah
eigen_tpl.sort(key=lambda x: x[0], reverse=True)
# Membuat tuple berpasangan
eigen_tuples = list(zip(eig_vals, eig_vecs.T))
tot = sum(eig_vals)
var_exp = [(i/tot)*100 for i in sorted(eig_vals, reverse=True)]
sum(var_exp[:5])
```

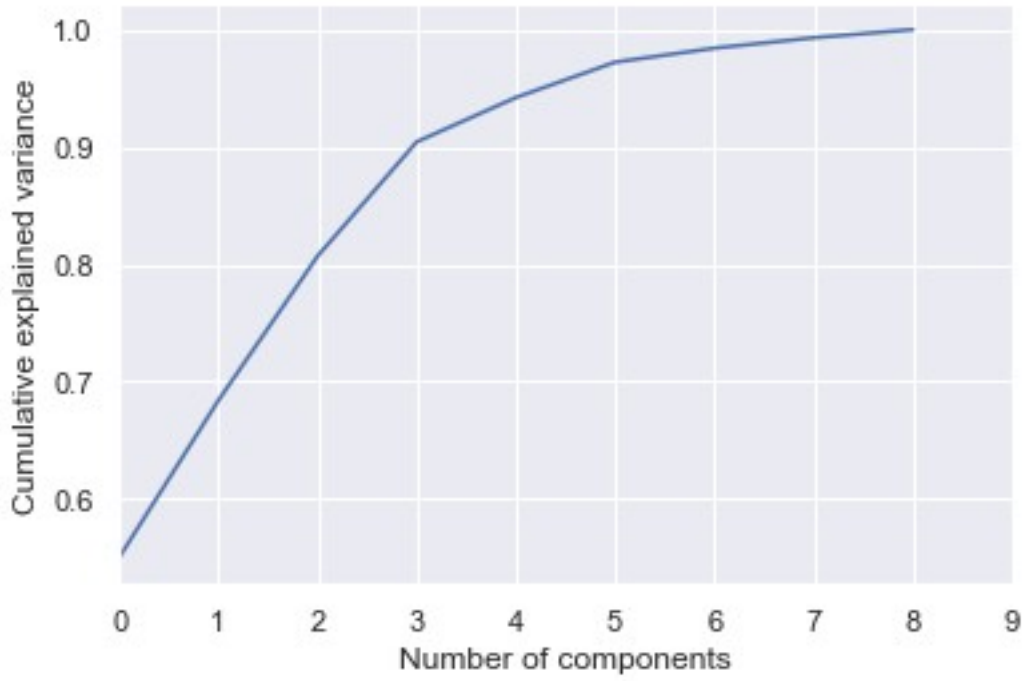
```
94.21407334594117
```

```
var_exp[:5] #Variance Explained
```

```
[55.00122655774046,
 13.384783693757754,
 12.301052915212333,
 9.749046549423326,
 3.7779636298072834]
```

```
pca = PCA().fit(data_scaled)
plt.plot(np.cumsum(pca.explained_variance_ratio_))
plt.xlim(0,9,1)
plt.xlabel('Number of components')
plt.ylabel('Cumulative explained variance')
#xticks 0 = pc ke 1
```

```
Text(0, 0.5, 'Cumulative explained variance')
```




```
sklearn_pca = PCA(n_components=5)
Y = sklearn_pca.fit_transform(data_scaled)
data_pca = pd.DataFrame(Y)
data_pca.columns = ['PC1', 'PC2', 'PC3', 'PC4', 'PC5']
data_pca.head()
```

	PC1	PC2	PC3	PC4	PC5
0	-0.599078	0.095490	0.157554	-0.024333	-0.045618
1	0.158474	-0.212092	-0.064189	-0.061247	0.014191
2	0.003686	-0.135867	-0.134182	0.133574	-0.091150
3	-0.650235	0.275975	-0.142672	0.156018	-0.081997
4	0.200711	-0.064662	-0.100715	-0.037902	-0.035799

Alternative PCA

Kategori 1: Faktor-faktor yang mempengaruhi kesehatan penduduk

- child_mort (angka kematian anak)
- health (pengeluaran kesehatan)
- life_expec (usia harapan hidup)
- total_fer (tingkat kesuburan total)

Kategori 2: Faktor-faktor ekonomi

- exports (ekspor)
- imports (impor)
- income (pendapatan)
- inflation (inflasi)

- gdpp (produk domestik bruto)

Kategori 1

```
X_cat1 = data_scaled[['child_mort', 'health', 'life_expec',
'total_fer']]
mean_vec = np.mean(X_cat1, axis=0)
cov_mat = (X_cat1-mean_vec).T.dot((X_cat1-mean_vec))/(X_cat1.shape[0]-
1)
eig_vals, eig_vecs = np.linalg.eig(cov_mat)
eigen_tpl = [(np.abs(eig_vals[i]), eig_vecs[:, i]) for i in
range(len(eig_vals))]
# Mengurutkan list berdasarkan nilai eigen tertinggi hingga terendah
eigen_tpl.sort(key=lambda x: x[0], reverse=True)
# Membuat tuple berpasangan
eigen_tuples = list(zip(eig_vals, eig_vecs.T))
tot = sum(eig_vals)
var_exp = [(i/tot)*100 for i in sorted(eig_vals, reverse=True)]
print(var_exp[:2],'\nvariance explained',sum(var_exp[:2]))
```

```
[73.38281126947672, 17.62934274946547]
variance explained 91.01215401894218
```

```
sklearn_pca = PCA(n_components=2)
Y = sklearn_pca.fit_transform(X_cat1)
data_alt_pca1 = pd.DataFrame(Y)
data_alt_pca1.columns = ['PC1', 'PC2']
data_alt_pca1.head()
```

```
        PC1       PC2
0  0.570686  0.130444
1 -0.246844 -0.052718
2 -0.067636 -0.172977
3  0.690146 -0.145954
4 -0.213201 -0.079150
```

Kategori 2

```
X_cat2 = data_scaled[['exports','imports', 'income', 'inflation',
'gdpp']]
mean_vec = np.mean(X_cat2, axis=0)
cov_mat = (X_cat2-mean_vec).T.dot((X_cat2-mean_vec))/(X_cat2.shape[0]-
1)
eig_vals, eig_vecs = np.linalg.eig(cov_mat)
eigen_tpl = [(np.abs(eig_vals[i]), eig_vecs[:, i]) for i in
range(len(eig_vals))]
# Mengurutkan list berdasarkan nilai eigen tertinggi hingga terendah
eigen_tpl.sort(key=lambda x: x[0], reverse=True)
# Membuat tuple berpasangan
eigen_tuples = list(zip(eig_vals, eig_vecs.T))
tot = sum(eig_vals)
```

```
var_exp = [(i/tot)*100 for i in sorted(eig_vals, reverse=True)]
print(var_exp[:2],'\nvariance explained',sum(var_exp[:2]))
```

```
[58.6423944317119, 26.480637466812755]
variance explained 85.12303189852466
```

```
sklearn_pca = PCA(n_components=2)
Y = sklearn_pca.fit_transform(X_cat2)
data_alt_pca2 = pd.DataFrame(Y)
data_alt_pca2.columns = ['PC1', 'PC2']
data_alt_pca2.head()
```

```
        PC1       PC2
0 -0.217559 -0.015057
1 -0.110550  0.020820
2 -0.107147 -0.041511
3 -0.089075  0.081291
4  0.035098  0.065220
```

Combined Category

```
X_combo = pd.concat([data_alt_pca1, data_alt_pca2], axis=1)
mean_vec = np.mean(X_combo, axis=0)
cov_mat = (X_combo-mean_vec).T.dot((X_combo-
mean_vec))/(X_combo.shape[0]-1)
eig_vals, eig_vecs = np.linalg.eig(cov_mat)
eigen_tpl = [(np.abs(eig_vals[i]), eig_vecs[:, i]) for i in
range(len(eig_vals))]
# Mengurutkan list berdasarkan nilai eigen tertinggi hingga terendah
eigen_tpl.sort(key=lambda x: x[0], reverse=True)
# Membuat tuple berpasangan
eigen_tuples = list(zip(eig_vals, eig_vecs.T))
tot = sum(eig_vals)
var_exp = [(i/tot)*100 for i in sorted(eig_vals, reverse=True)]
print(var_exp[:2],'\nvariance explained',sum(var_exp[:2]))
```

```
[61.91124123014254, 14.871311546405428]
variance explained 76.78255277654796
```

```
sklearn_pca = PCA(n_components=2)
Y = sklearn_pca.fit_transform(X_combo)
data_alt_pcac = pd.DataFrame(Y)
data_alt_pcac.columns = ['PC1', 'PC2']
data_alt_pcac.head()
```

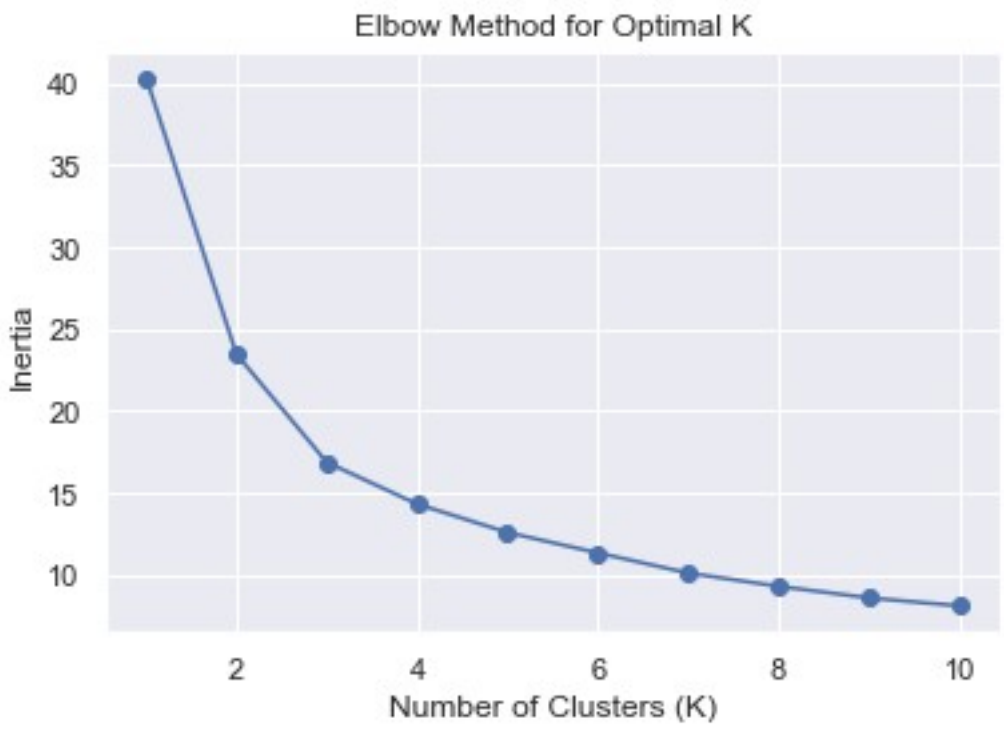
```
        PC1       PC2
0 -0.600371  0.092621
1  0.156871 -0.202674
2  0.004947 -0.151342
3 -0.646634  0.253723
4  0.198324 -0.056324
```

K-Means Clustering

PCA

ELBOW METHOD FOR N CLUSTER DECISION

```
# Calculate inertia for different numbers of clusters
inertia = []
for n_clusters in range(1, 11):
    kmeans = KMeans(n_clusters=n_clusters, random_state=42)
    kmeans.fit(data_pca)
    inertia.append(kmeans.inertia_)

# Plot the Elbow Method graph
plt.plot(range(1, 11), inertia, marker='o')
plt.title('Elbow Method for Optimal K')
plt.xlabel('Number of Clusters (K)')
plt.ylabel('Inertia')
plt.show()
```



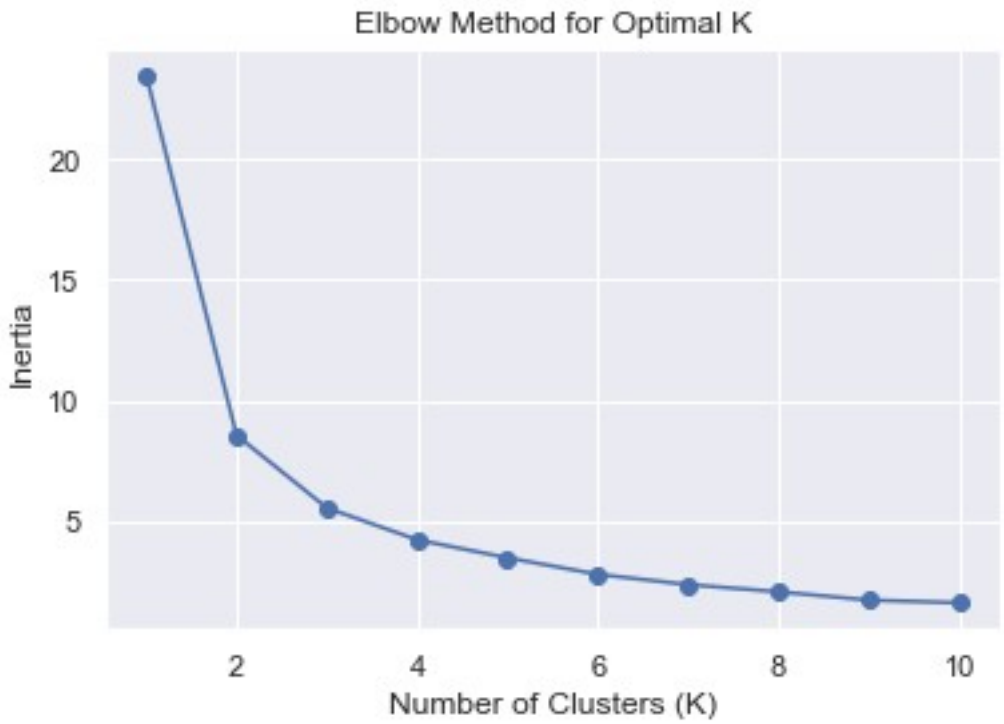

```
kmeans = KMeans(n_clusters=3)
data_pca['Cluster'] = kmeans.fit_predict(data_pca)
data_pca.head()
```

	PC1	PC2	PC3	PC4	PC5	Cluster
0	-0.599078	0.095490	0.157554	-0.024333	-0.045618	0

```
1  0.158474 -0.212092 -0.064189 -0.061247  0.014191         1
2  0.003686 -0.135867 -0.134182  0.133574 -0.091150         1
3 -0.650235  0.275975 -0.142672  0.156018 -0.081997         0
4  0.200711 -0.064662 -0.100715 -0.037902 -0.035799         1
```

Alternative Category1

```
# Calculate inertia for different numbers of clusters
inertia = []
for n_clusters in range(1, 11):
    kmeans = KMeans(n_clusters=n_clusters, random_state=42)
    kmeans.fit(data_alt_pca1)
    inertia.append(kmeans.inertia_)

# Plot the Elbow Method graph
plt.plot(range(1, 11), inertia, marker='o')
plt.title('Elbow Method for Optimal K')
plt.xlabel('Number of Clusters (K)')
plt.ylabel('Inertia')
plt.show()
```



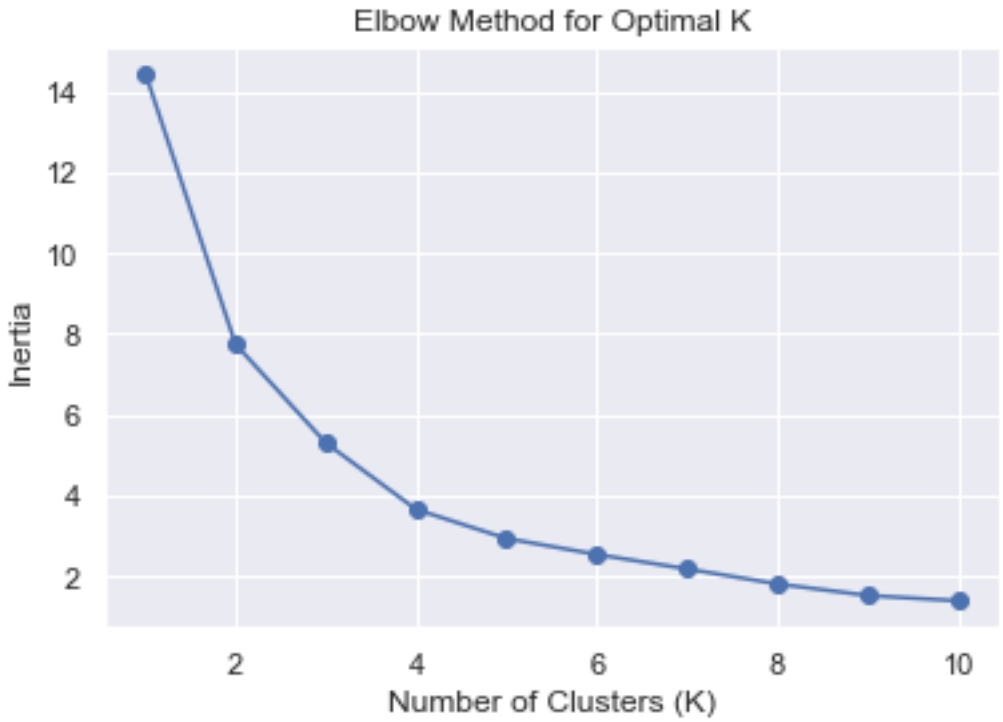

```
kmeans = KMeans(n_clusters=2)
data_alt_pca1['Cluster'] = kmeans.fit_predict(data_alt_pca1)
data_alt_pca1.head()
```

```
        PC1       PC2  Cluster
0  0.570686  0.130444        0
```

```
1 -0.246844 -0.052718        1
2 -0.067636 -0.172977        1
3  0.690146 -0.145954        0
4 -0.213201 -0.079150        1
```

Alternative Category2

```
# Calculate inertia for different numbers of clusters
inertia = []
for n_clusters in range(1, 11):
    kmeans = KMeans(n_clusters=n_clusters, random_state=42)
    kmeans.fit(data_alt_pca2)
    inertia.append(kmeans.inertia_)

# Plot the Elbow Method graph
plt.plot(range(1, 11), inertia, marker='o')
plt.title('Elbow Method for Optimal K')
plt.xlabel('Number of Clusters (K)')
plt.ylabel('Inertia')
plt.show()
```



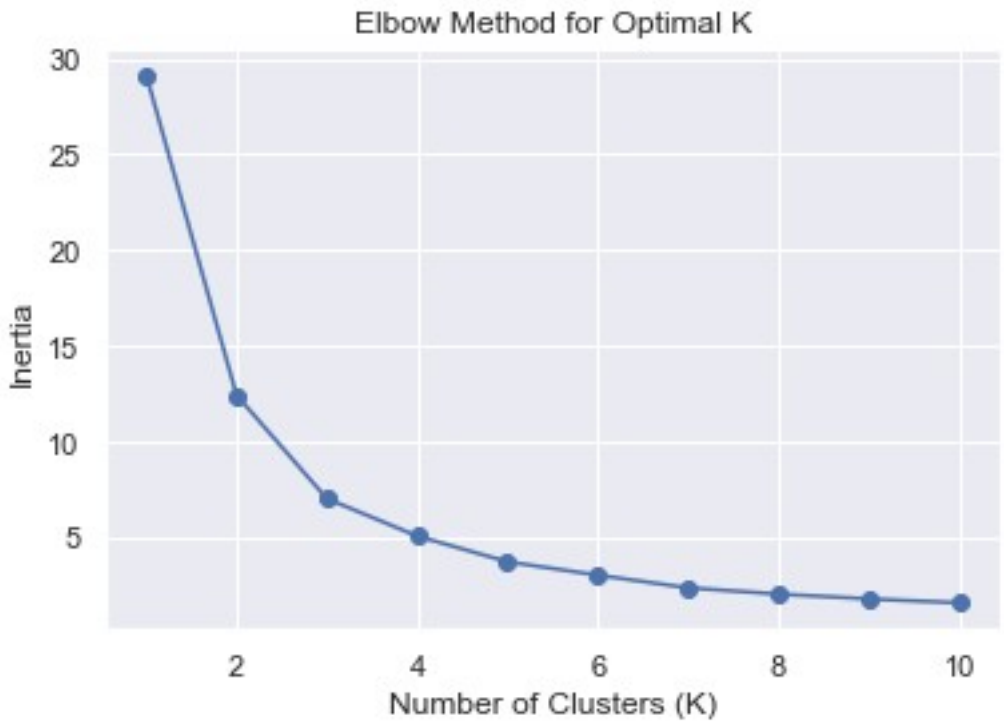

```
kmeans = KMeans(n_clusters=3)
data_alt_pca2['Cluster'] = kmeans.fit_predict(data_alt_pca2)
data_alt_pca2.head()
```

```
        PC1       PC2  Cluster
0 -0.217559 -0.015057        0
```

```
1 -0.110550  0.020820        0
2 -0.107147 -0.041511        0
3 -0.089075  0.081291        0
4  0.035098  0.065220        0
```

Alternative Combined Category

```
# Calculate inertia for different numbers of clusters
inertia = []
for n_clusters in range(1, 11):
    kmeans = KMeans(n_clusters=n_clusters, random_state=42)
    kmeans.fit(data_alt_pcac)
    inertia.append(kmeans.inertia_)

# Plot the Elbow Method graph
plt.plot(range(1, 11), inertia, marker='o')
plt.title('Elbow Method for Optimal K')
plt.xlabel('Number of Clusters (K)')
plt.ylabel('Inertia')
plt.show()
```



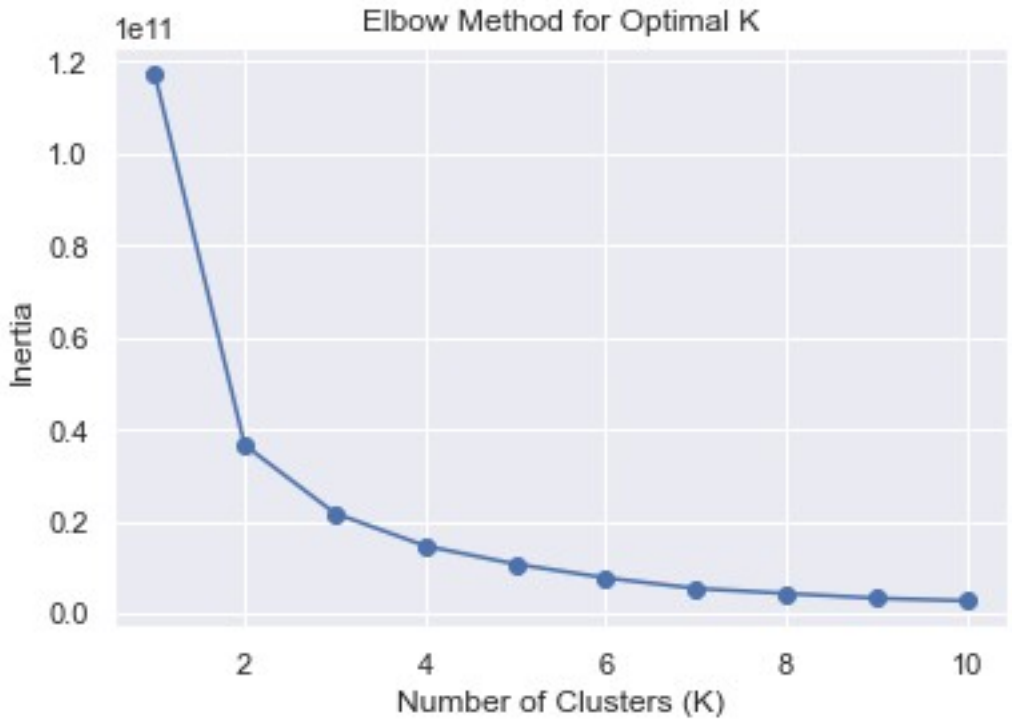

```
kmeans = KMeans(n_clusters=4)
data_alt_pcac['Cluster'] = kmeans.fit_predict(data_alt_pcac)
data_alt_pcac.head()
```

```
        PC1       PC2  Cluster
0 -0.600371  0.092621        0
```

```
1  0.156871 -0.202674         3
2  0.004947 -0.151342         3
3 -0.646634  0.253723         0
4  0.198324 -0.056324         1
```

Alternative Without PCA

```
# Calculate inertia for different numbers of clusters
inertia = []
for n_clusters in range(1, 11):
    kmeans = KMeans(n_clusters=n_clusters, random_state=42)
    kmeans.fit(data.iloc[:,1:])
    inertia.append(kmeans.inertia_)

# Plot the Elbow Method graph
plt.plot(range(1, 11), inertia, marker='o')
plt.title('Elbow Method for Optimal K')
plt.xlabel('Number of Clusters (K)')
plt.ylabel('Inertia')
plt.show()
```




```
data_nopca = data.iloc[:,1:]
kmeans = KMeans(n_clusters=2)
data_nopca['Cluster'] = kmeans.fit_predict(data_nopca)
data_nopca.head()
```

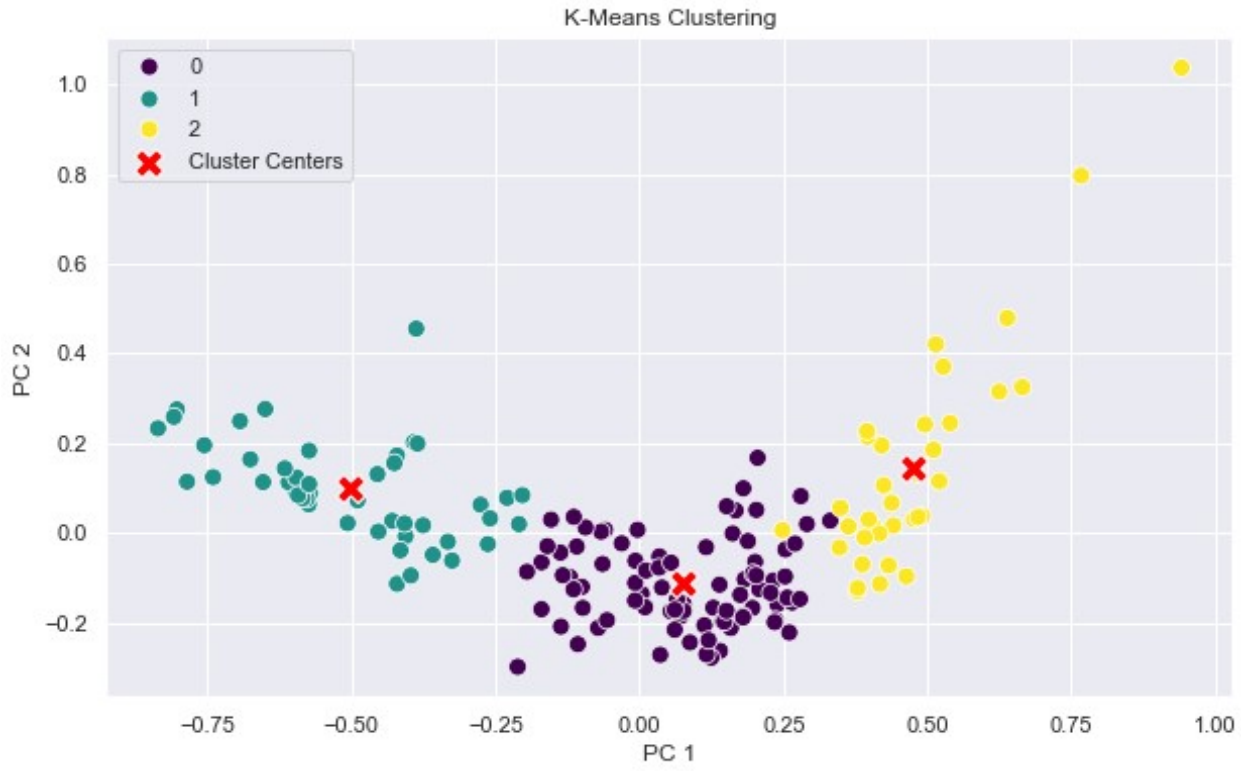
```
   child_mort  exports  health  imports  income  inflation  life_expec

0        90.2     10.0    7.58     44.9    1610       9.44        56.2
\
1        16.6     28.0    6.55     48.6    9930       4.49        76.3

2        27.3     38.4    4.17     31.4   12900      16.10        76.5

3       119.0     62.3    2.85     42.9    5900      22.40        60.1

4        10.3     45.5    6.03     58.9   19100       1.44        76.8


   total_fer   gdpp  Cluster
0       5.82    553        0
1       1.65   4090        0
2       2.89   4460        0
3       6.16   3530        0
4       2.13  12200        0
```

Visualization

PCA >>> K-Means

Normal PCA

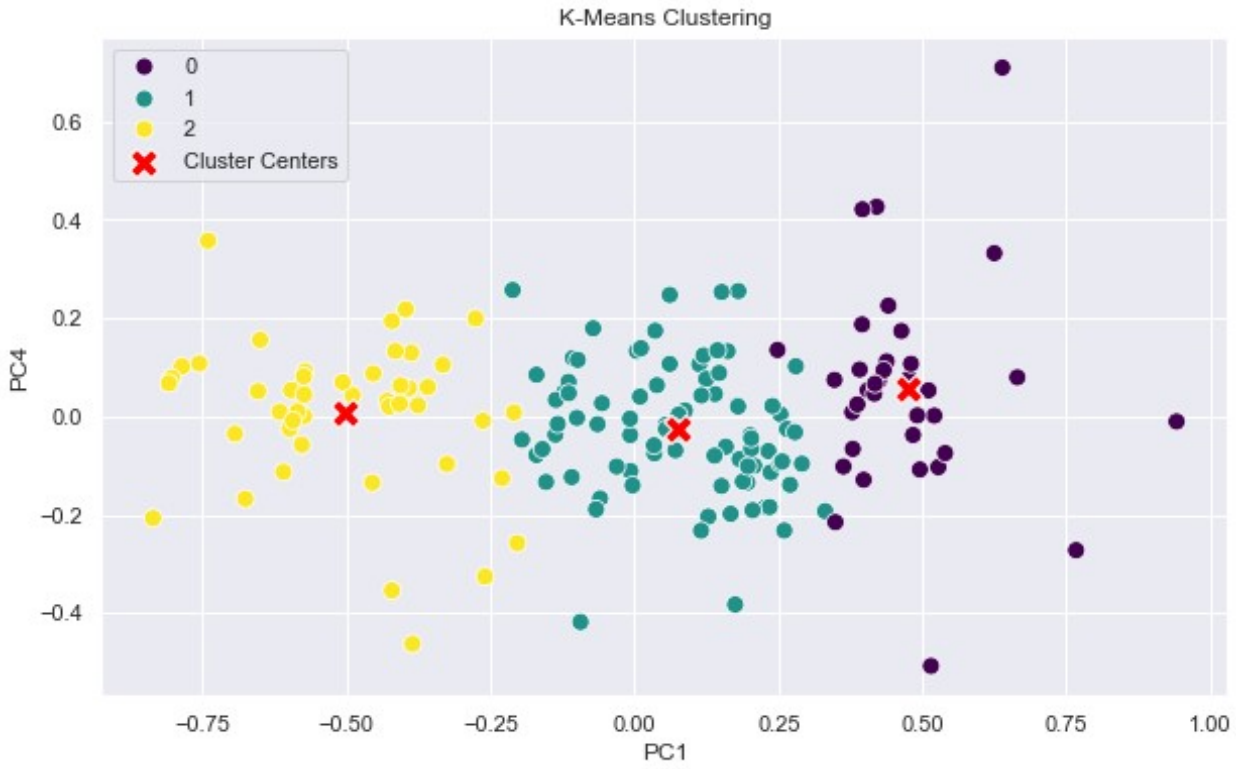
```
kmeans = KMeans(n_clusters=3)
data_pca['Cluster'] = kmeans.fit_predict(data_pca)
plt.figure(figsize=(10, 6))

# Plot original data
sns.scatterplot(x='PC1', y='PC2', data=data_pca, hue='Cluster',
palette='viridis', s=80)

# Plot cluster centers
sns.scatterplot(x=kmeans.cluster_centers_[:, 0],
y=kmeans.cluster_centers_[:, 1],
                marker='X', color='red', s=200, label='Cluster
Centers')

plt.title('K-Means Clustering')
plt.xlabel('PC 1')
plt.ylabel('PC 2')
plt.legend()
plt.show()
```



Function

```
def linear_search(arr, target):
    for i, val in enumerate(arr):
        if val == target:
            return i
    return -1
def kmeans_clustering(data, x, y, n):

    kmeans = KMeans(n_clusters=n)
    data['Cluster'] = kmeans.fit_predict(data)

    plt.figure(figsize=(10, 6))
    sns.scatterplot(x=x, y=y, data=data, hue='Cluster',
palette='viridis', s=80)
    sns.scatterplot(x=kmeans.cluster_centers_[:,
linear_search(data.columns, x)],
                    y=kmeans.cluster_centers_[:,
linear_search(data.columns, y)],
                    marker='X', color='red', s=200, label='Cluster
Centers')

    plt.title('K-Means Clustering')
    plt.xlabel(x)
    plt.ylabel(y)
    plt.legend()
    plt.show()
```

```
kmeans_clustering(data_pca,'PC1','PC4',3)
```

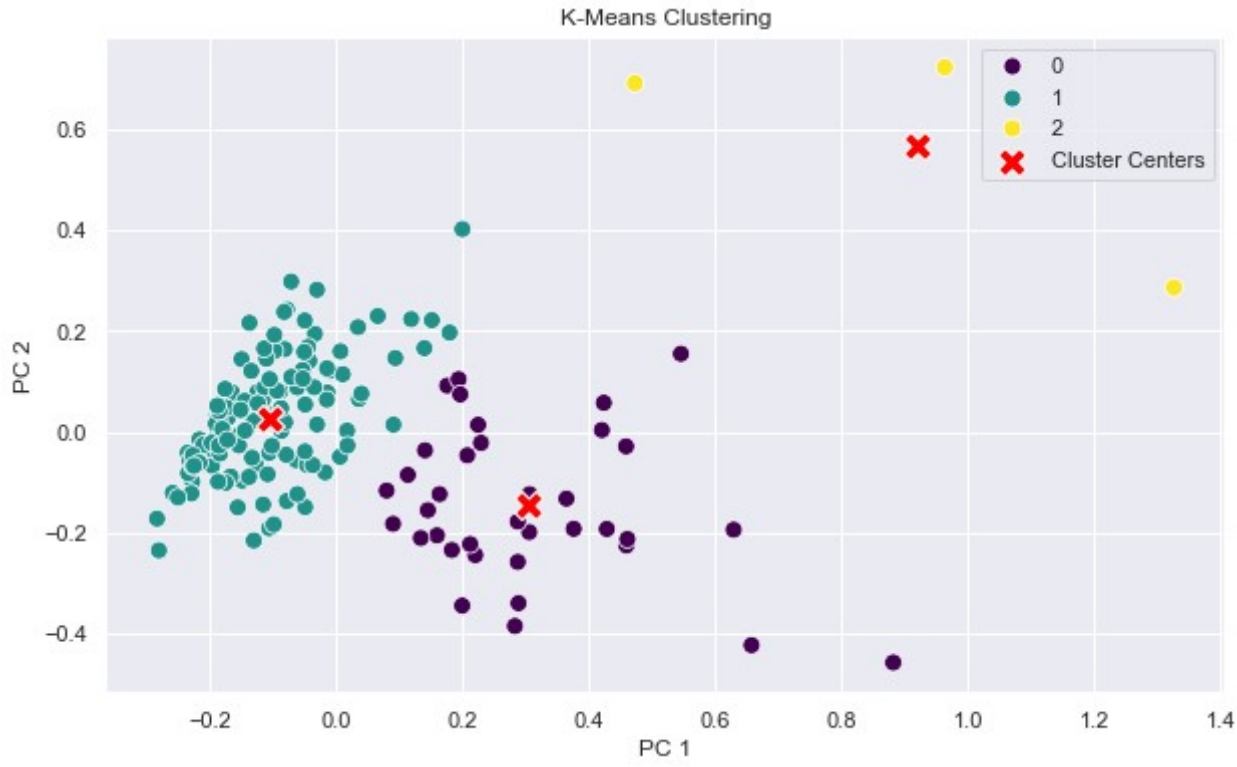


Alternative Category1

```
kmeans = KMeans(n_clusters=2)
data_alt_pca1['Cluster'] = kmeans.fit_predict(data_alt_pca1)
plt.figure(figsize=(10, 6))

# Plot original data
sns.scatterplot(x='PC1', y='PC2', data=data_alt_pca1, hue='Cluster',
palette='viridis', s=80)

# Plot cluster centers
sns.scatterplot(x=kmeans.cluster_centers_[:, 0],
y=kmeans.cluster_centers_[:, 1],
                marker='X', color='red', s=200, label='Cluster
Centers')

plt.title('K-Means Clustering')
plt.xlabel('PC 1')
plt.ylabel('PC 2')
plt.legend()
plt.show()
```

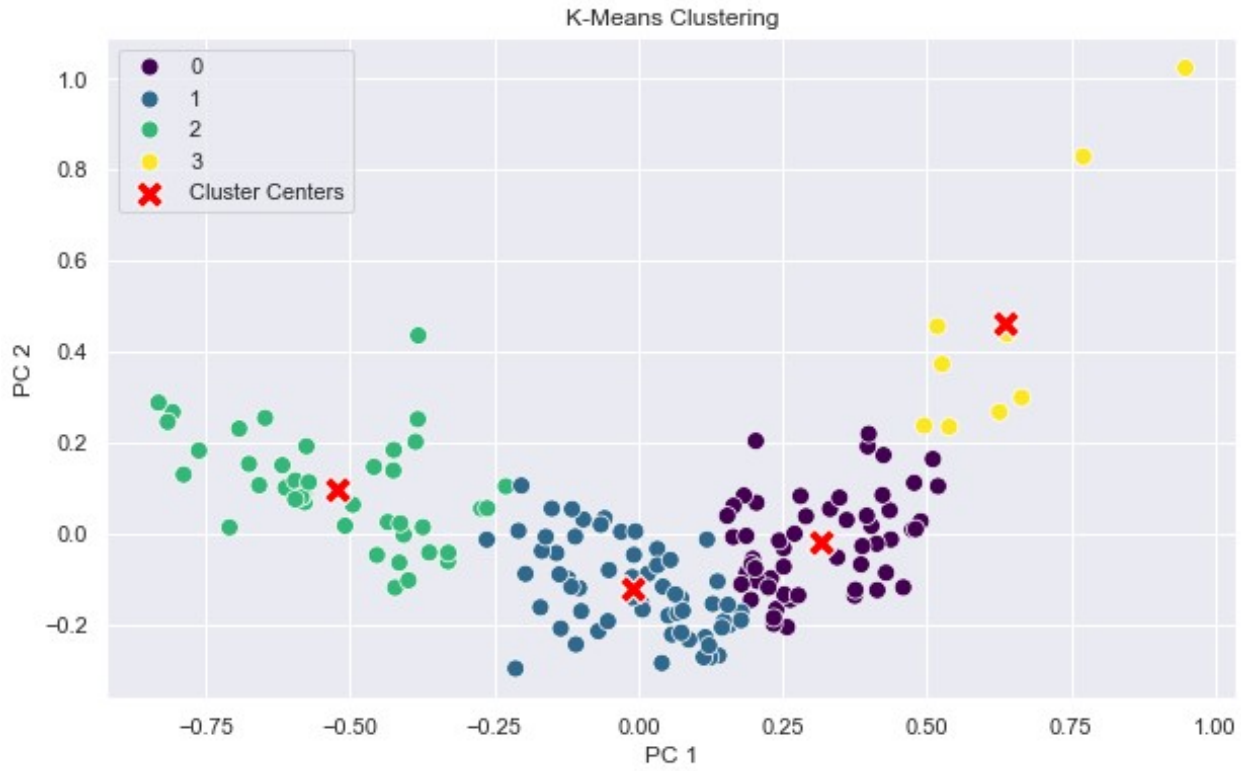
Alternative Category2

```
kmeans = KMeans(n_clusters=3)
data_alt_pca2['Cluster'] = kmeans.fit_predict(data_alt_pca2)
plt.figure(figsize=(10, 6))

# Plot original data
sns.scatterplot(x='PC1', y='PC2', data=data_alt_pca2, hue='Cluster',
palette='viridis', s=80)

# Plot cluster centers
sns.scatterplot(x=kmeans.cluster_centers_[:, 0],
y=kmeans.cluster_centers_[:, 1],
                marker='X', color='red', s=200, label='Cluster
Centers')

plt.title('K-Means Clustering')
plt.xlabel('PC 1')
plt.ylabel('PC 2')
plt.legend()
plt.show()
```



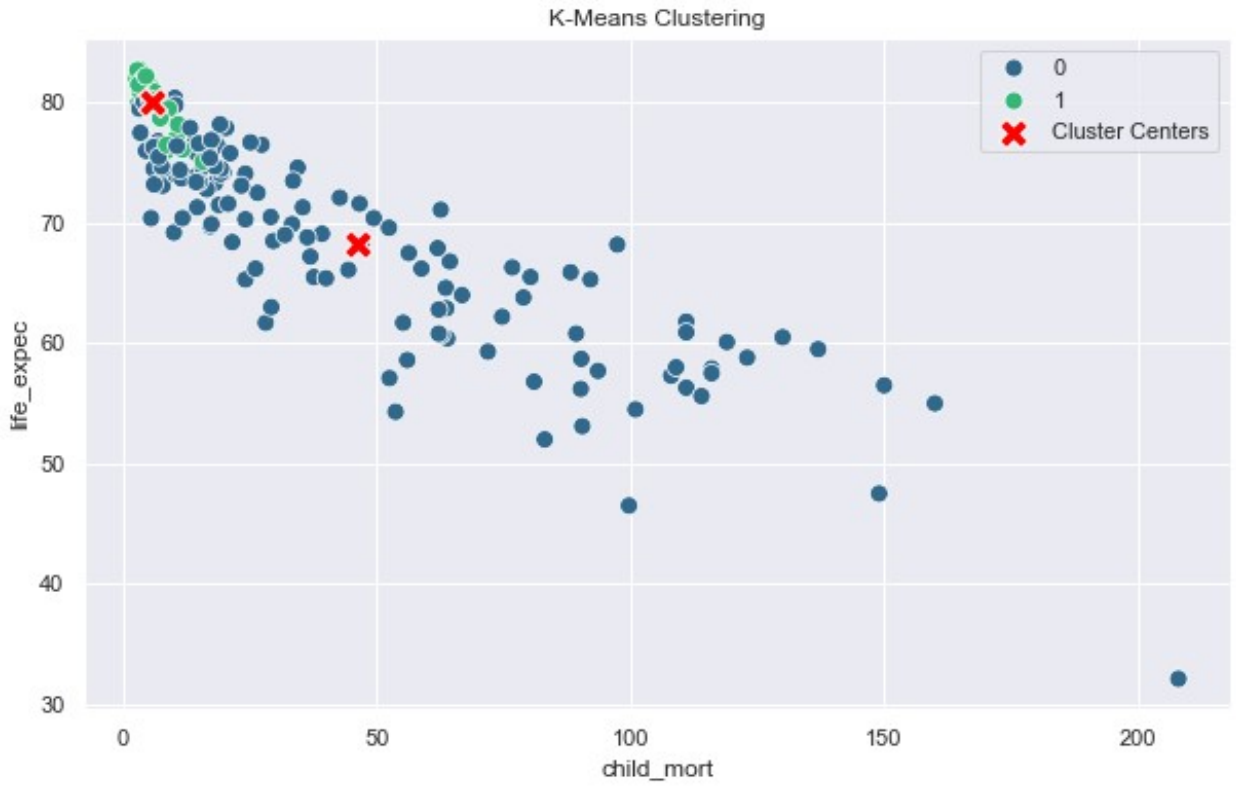
Alternative Combined Category

```
kmeans = KMeans(n_clusters=4)
data_alt_pcac['Cluster'] = kmeans.fit_predict(data_alt_pcac)
plt.figure(figsize=(10, 6))

# Plot original data
sns.scatterplot(x='PC1', y='PC2', data=data_alt_pcac, hue='Cluster',
palette='viridis', s=80)

# Plot cluster centers
sns.scatterplot(x=kmeans.cluster_centers_[:, 0],
y=kmeans.cluster_centers_[:, 1],
                marker='X', color='red', s=200, label='Cluster
Centers')

plt.title('K-Means Clustering')
plt.xlabel('PC 1')
plt.ylabel('PC 2')
plt.legend()
plt.show()
```



Without PCA >>> K-means

```
kmeans_clustering(data_nopca,'child_mort','life_expec',2)
```



Result

```
result = pd.concat([data_pca.Cluster, data_alt_pca1.Cluster,
data_alt_pca2.Cluster, data_alt_pcac.Cluster], axis=1)
result.columns = ['Cluster_PCA', 'Cluster_Cat1', 'Cluster_Cat2',
'Cluster_Combine']
result.replace({0: 1, 1: 2,
            2: 3, 3: 4}, inplace=True)
result = pd.concat([data,result], axis=1).set_index(pd.Index(range(1,
len(data) + 1)))
result.head()
```

	country	child_mort	exports	health	imports	income
1	Afghanistan	90.2	10.0	7.58	44.9	1610 \
2	Albania	16.6	28.0	6.55	48.6	9930
3	Algeria	27.3	38.4	4.17	31.4	12900
4	Angola	119.0	62.3	2.85	42.9	5900
5	Antigua and Barbuda	10.3	45.5	6.03	58.9	19100

```
   inflation  life_expec  total_fer   gdpp  Cluster_PCA  Cluster_Cat1
1       9.44        56.2       5.82    553            3             2
\
2       4.49        76.3       1.65   4090            2             1

3      16.10        76.5       2.89   4460            2             1

4      22.40        60.1       6.16   3530            3             2

5       1.44        76.8       2.13  12200            2             1


   Cluster_Cat2  Cluster_Combine
1             2                3
2             2                2
3             2                2
4             2                3
5             2                1
```

Conclusion

Dataset yang kami olah berisikian kualitas/penilaian negara-negara di seluruh dunia. Sebelum menerapkan metode unsupervised learning, kami perlu mengecek kondisi data terlebih dahulu. Ditemukan bahwa dataset tersebut sudah bersih dari nilai yang tidak seragam maupun missing value. Setelahnya, kami perlu melakukan normalisasi/standarisasi kepada data (tujuannya adalah menyeragamkan variansi data). Namun sebelumnya perlu dilakukan cek distribusi data untuk menentukan metode standarisasi apa yang harus digunakan. Karena semua kolom memiliki outlier, serta sebagian besar tidak berdistribusi normal. Maka kami menggunakan standarisasi dengan min-max.

Setelah data terstandarisasi, berikutnya kami akan menerapkan sejumlah metode unsupervised learning dalam beberapa skema. Pertama, mereduksi dimensi menggunakan PCA terhadap data asli dan data yang dibagi menjadi 2 kategori (Kategori 1, Kesehatan dan Kategori 2, Ekonomi). Data yang sudah terpisah dari kategori nantinya akan digabung dan direduksi lagi dimensinya (Pada tahap tersebut tingkat variansi yang bisa dijelaskan jelas mengalami penurunan beberapa persen). Kedua, melakukan pemodelan klasterisasi dengan K-means terhadap semua data hasil dari reduksi dimensi, dan data asli. Terakhir, membuat visualisasi dari hasil model tersebut.

Semua metode unsupervised learning dilakukan dengan bantuan modul sklearn yang sudah ada, namun terdapat beberapa poin yang perlu dilakukan terpisah (tetapi tetap dengan memanfaatkan modul sklearn). Seperti pada saat PCA, untuk menentukan jumlah komponen baru (reduksi komponennya) perlu dilihat jumlah variansi yang dapat dijelaskan (secara kumulatif) untuk mengetahui sejauh mana kita ingin menjaga keaslian data. Begitu juga pada saat klasterisasi dengan K-means, perlu dilihat nilai inersia dari model. Dengan menggunakan elbow method, didapatkan nilai K yang cocok untuk klasterisasi.

Karena klastering tidak memberikan luaran label yang spesifik (hanya label klaster tanpa keterangan lebih lanjut, karena bukan klasifikasi), maka hasil dari klastering secara umum adalah sebagai berikut;

- Klaster yang besar cenderung mewakili kelompok data yang lebih umum, sedangkan klaster yang kecil cenderung mewakili kelompok data yang lebih spesifik
- Jarak antar klaster menunjukkan tingkat kemiripan antar klaster. Klaster yang jaraknya dekat cenderung lebih mirip, sedangkan klaster yang jaraknya jauh cenderung lebih berbeda.

Perlu diingat bahwa interpretasi hasil visualisasi klasterisasi tanpa diketahui konteks labelnya hanya bersifat sementara. Interpretasi yang lebih akurat dapat dilakukan setelah konteks label diketahui. Sayangnya dari data yang disediakan tidak ada konteks lebih lanjut mengenai labelnya.