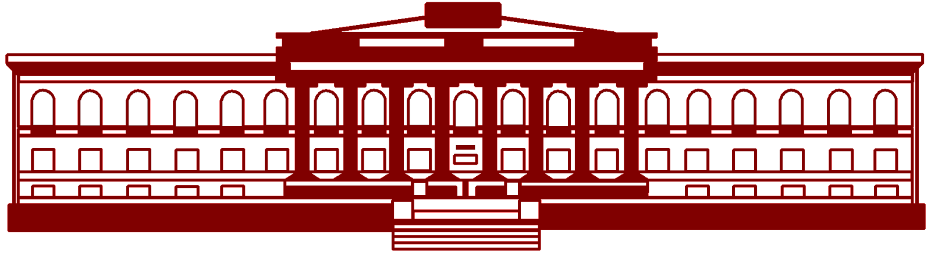
**КИЇВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ**

**імені ТАРАСА ШЕВЧЕНКА**

****

**ФАКУЛЬТЕТ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ**

**Кафедра прикладних інформаційних систем**

**Звіт до лабораторної роботи №4**

**з курсу**

**«Інтелектуальний аналіз даних»**

*Студента 3 курсу*

*групи ПП-32*

*спеціальності 122 «Комп'ютерні науки»*

*ОП «Прикладне програмування»*

Чалого Єгора Олеговича

*Викладач:*

*Білий Р.О.*

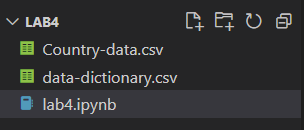
**Київ – 2023**

**Тема:** Кластеризація. Метод k-mean, пошук оптимальних параметрів.

**Мета роботи:** Отримання практичних навичок з виконання кластеризації даних, використовуючи метод k-mean та інші з пакету sklearn.

**Хід роботи:**

1. Після вивчення наданих прикладів завантажив файли з даними у папку проекту:



2. Почав із необхідної перевірки даних, очищення даних та EDA, зробив однофакторний аналіз, двофакторний аналіз. Візуалізував закономірності:

import pandas as pd

import matplotlib.pyplot as plt

import seaborn as sns

from sklearn.cluster import KMeans

import os

NUM\_THREADS = "1"

os.environ["OMP\_NUM\_THREADS"] = NUM\_THREADS

os.environ["OPENBLAS\_NUM\_THREADS"] = NUM\_THREADS

os.environ["MKL\_NUM\_THREADS"] = NUM\_THREADS

os.environ["VECLIB\_MAXIMUM\_THREADS"] = NUM\_THREADS

os.environ["NUMEXPR\_NUM\_THREADS"] = NUM\_THREADS

%matplotlib inline

import matplotlib.pyplot as plt

import seaborn as sns; sns.set()  # for plot styling

import numpy as np

import warnings; warnings.simplefilter('ignore')

data = pd.read\_csv('Country-data.csv')

print(f"Перевірка загальної інформації про дані:\n")

print(data.info())

print("")

print(f"Перевірка на пропущені значення:\n")

print(data.isnull().sum())

# очищення даних

numeric\_data = data.select\_dtypes(include=['float64', 'int64'])

# однофакторний аналіз

print(f"Однофакторний аналіз:\n")

for column in data.columns[1:]:

    plt.figure(figsize=(8, 6))

    sns.histplot(data[column], bins=20, kde=True)

    plt.title(f'{column} distribution')

    plt.xlabel(column)

    plt.show()

# двофакторний аналіз

print(f"Двофакторний аналіз:\n")

plt.figure(figsize=(8, 6))

sns.scatterplot(x='gdpp', y='child\_mort', data=data)

plt.title('GDP per Capita vs Child Mortality Rate')

plt.xlabel('GDP per Capita')

plt.ylabel('Child Mortality Rate')

plt.show()

# Кореляційна матриця

print(f"Кореляційна матриця:\n")

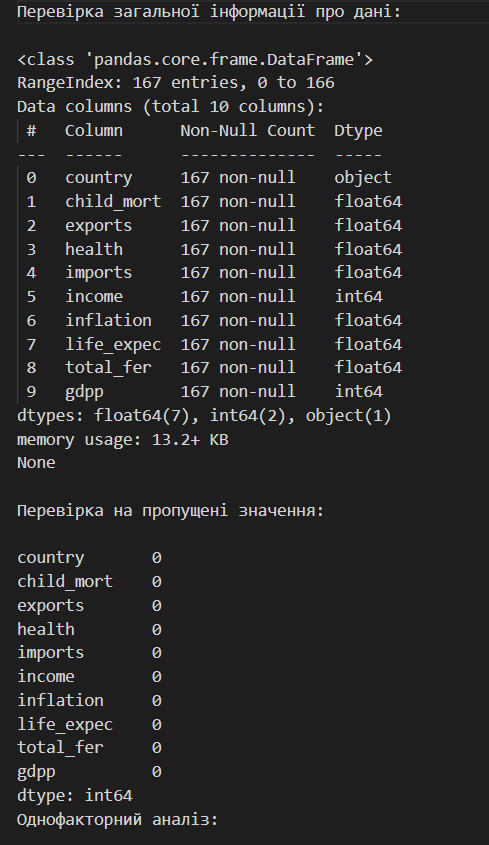
plt.figure(figsize=(10, 8))

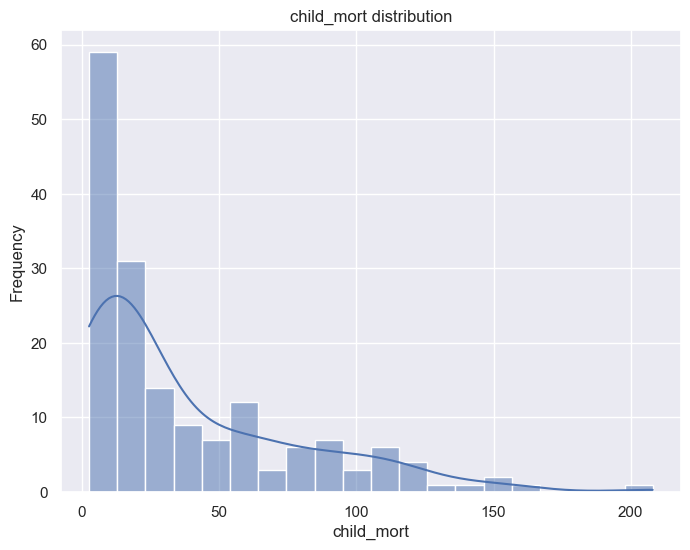
sns.heatmap(numeric\_data.corr(), annot=True, cmap='coolwarm', fmt='.2f')

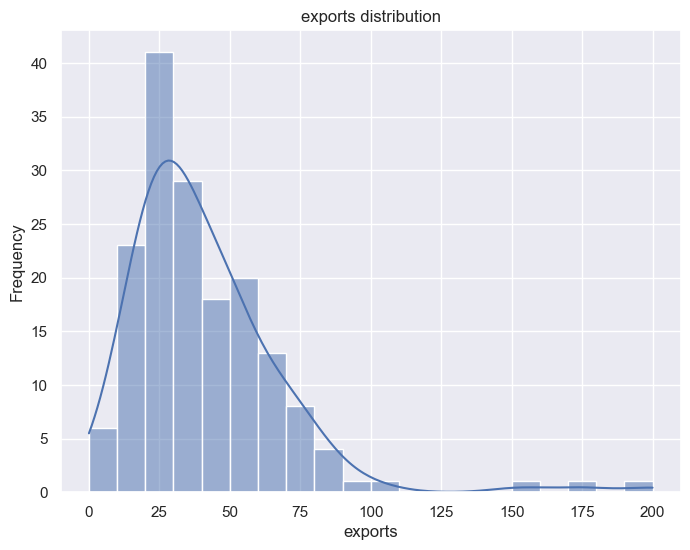
plt.title('Correlation Matrix')

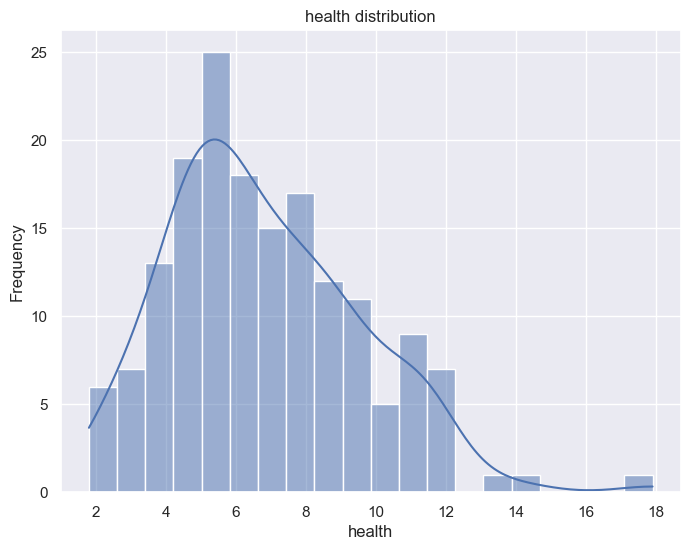
plt.show()

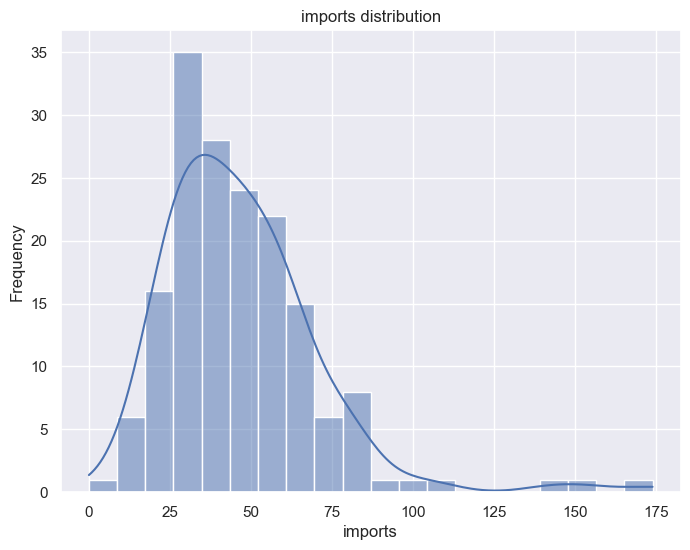
Результат:

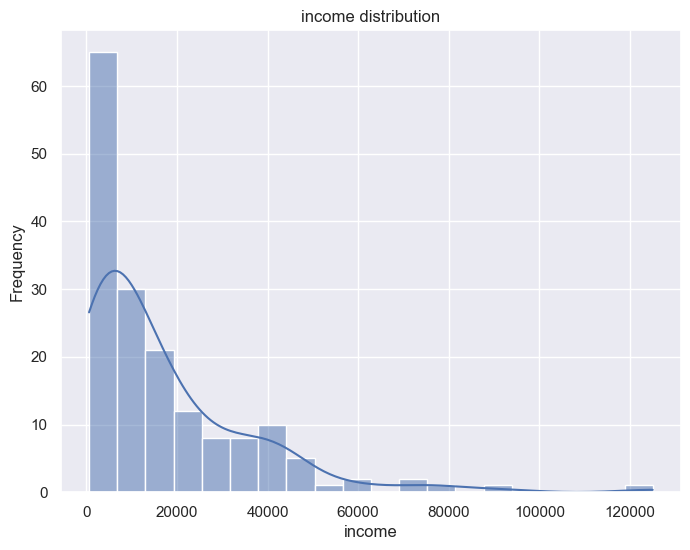


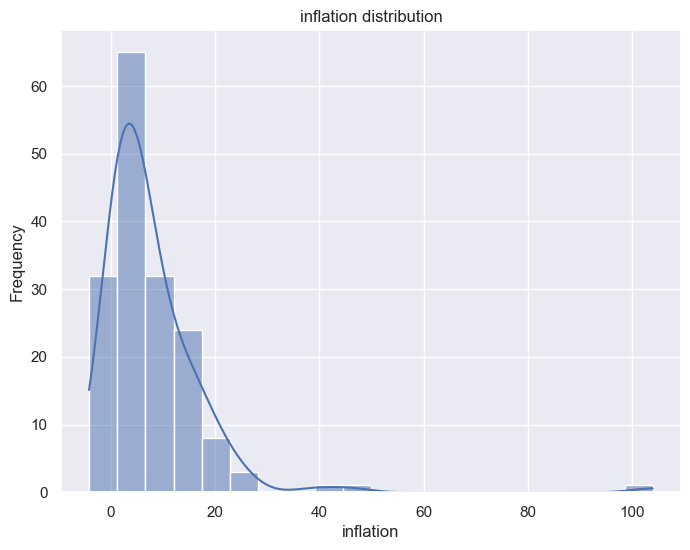


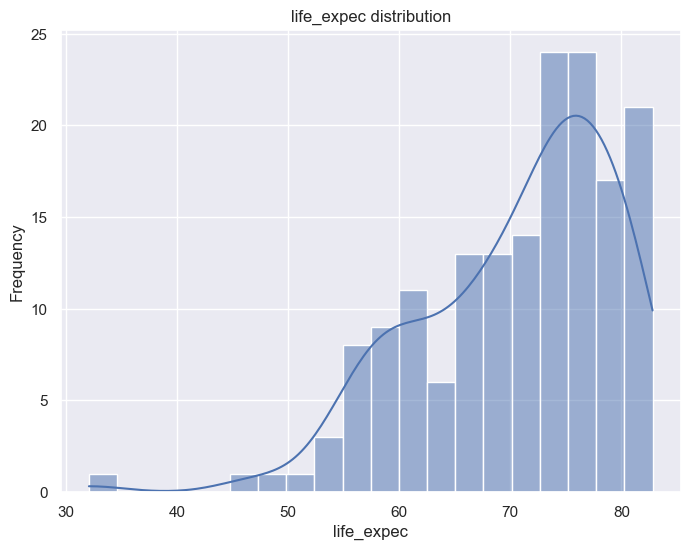


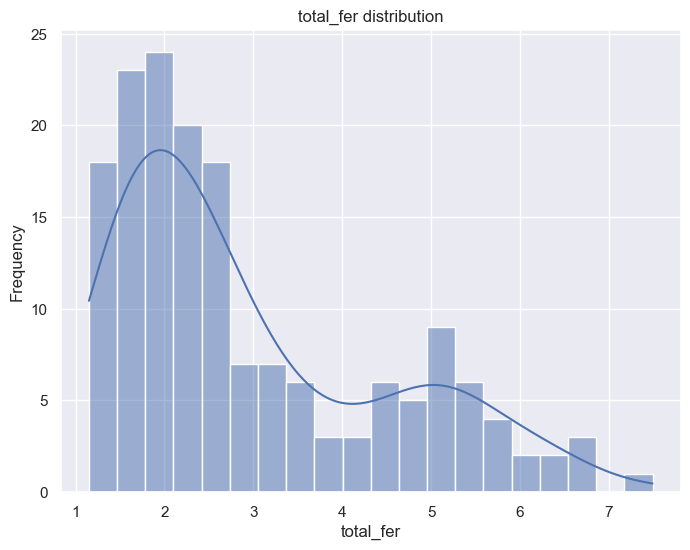


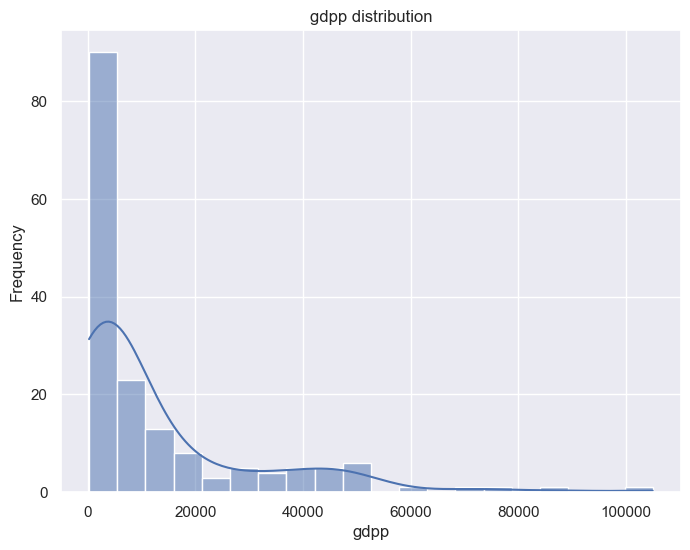


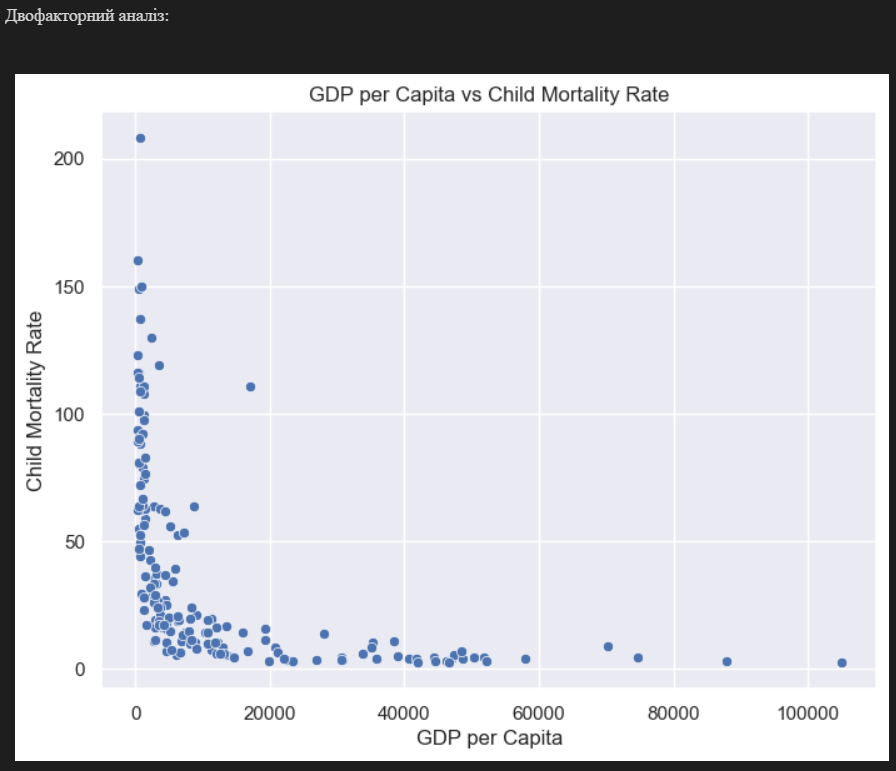




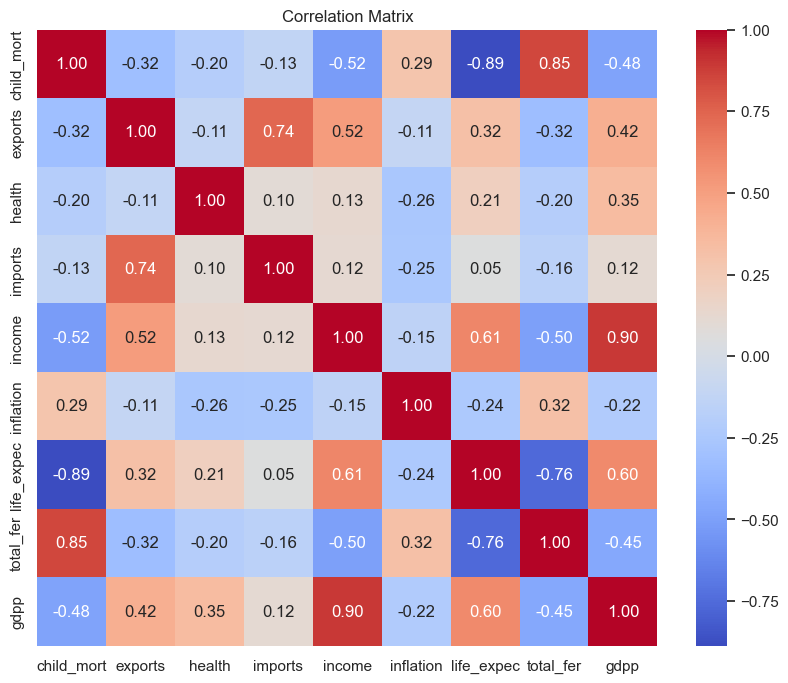












3. Виконав аналіз викидів для набору даних і вивів результати:

# аналіз викидів

import seaborn as sns

import matplotlib.pyplot as plt

variables = ['gdpp', 'child\_mort', 'income']

plt.figure(figsize=(10, 6))

for i, variable in enumerate(variables, 1):

    plt.subplot(1, 3, i)

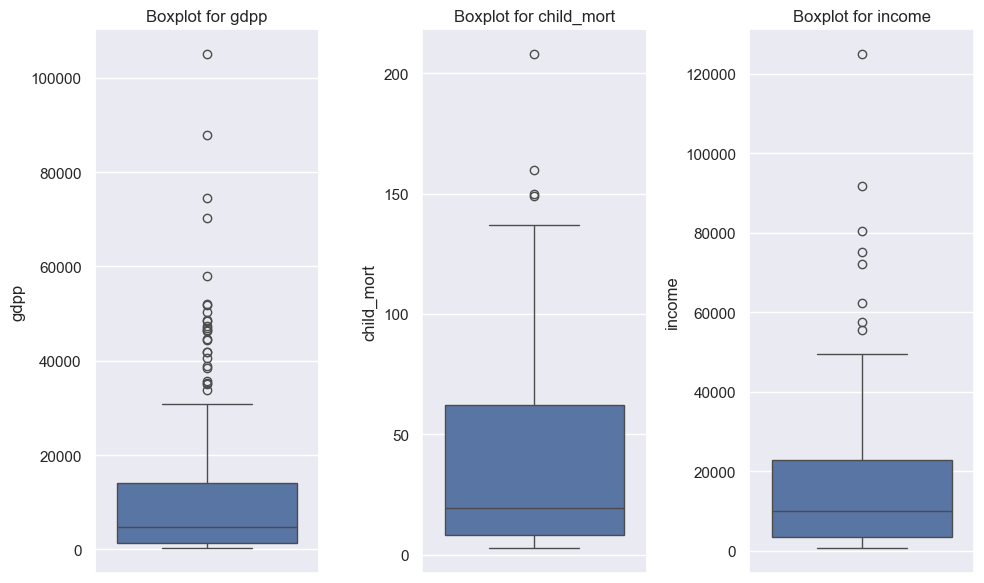
    sns.boxplot(y=data[variable])

    plt.title(f'Boxplot for {variable}')

    plt.tight\_layout()

plt.show()

Результат:



4. Провів кластеризацію, використовуючи k-mean методологію з прикладів та документації:

import pandas as pd

import matplotlib.pyplot as plt

from sklearn.cluster import KMeans

pd.set\_option('display.max\_rows', None)

data = pd.read\_csv('Country-data.csv')

features = ['gdpp', 'child\_mort', 'income']

optimal\_k = 3

kmeans = KMeans(n\_clusters=optimal\_k, random\_state=42)

data['Cluster'] = kmeans.fit\_predict(data[features])

for cluster\_index in range(optimal\_k):

    cluster\_countries = data[data['Cluster'] == cluster\_index]['country']

    print(f"Країни у кластері {cluster\_index}:")

    print(cluster\_countries)

# візуалізація кластерів

for i, col1 in enumerate(features):

    for j, col2 in enumerate(features):

        if i < j:

            plt.figure(figsize=(8, 6))

            for cluster in range(optimal\_k):

                cluster\_data = data[data['Cluster'] == cluster]

                plt.scatter(cluster\_data[col1], cluster\_data[col2], label=f'Cluster {cluster}')

            plt.xlabel(col1)

            plt.ylabel(col2)

            plt.title(f'{col1} vs {col2}')

            plt.legend()

            plt.show()

# аналіз кластерів

cluster\_means = data.groupby('Cluster')[features].mean()

print("Середні значення для кожного кластера:")

print(cluster\_means)

# кластери, що потребують допомоги

# розглядаємо кластери з низьким GDPP(ввп на душу населення),високим child\_mort(смертністю дітей) та низьким доходом як ты, що потребую допомоги:

needy\_clusters = cluster\_means[(cluster\_means['gdpp'] < cluster\_means['gdpp'].median()) &

                               (cluster\_means['child\_mort'] > cluster\_means['child\_mort'].median()) &

                               (cluster\_means['income'] < cluster\_means['income'].median())]

print("\nКластери, які гостро потребують допомоги:")

print(needy\_clusters)

needy\_cluster\_indices = data[data['Cluster'].isin(needy\_clusters.index)]

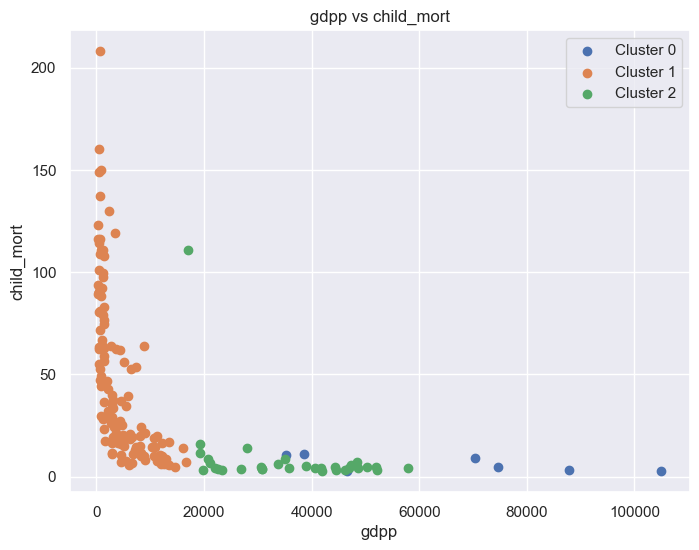
needy\_countries = needy\_cluster\_indices['country']

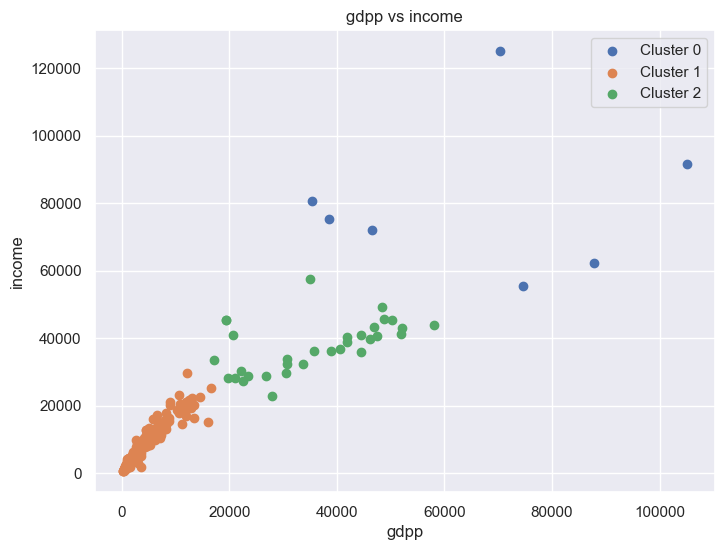
print("\nКраїни, які гостро потребують допомоги:")

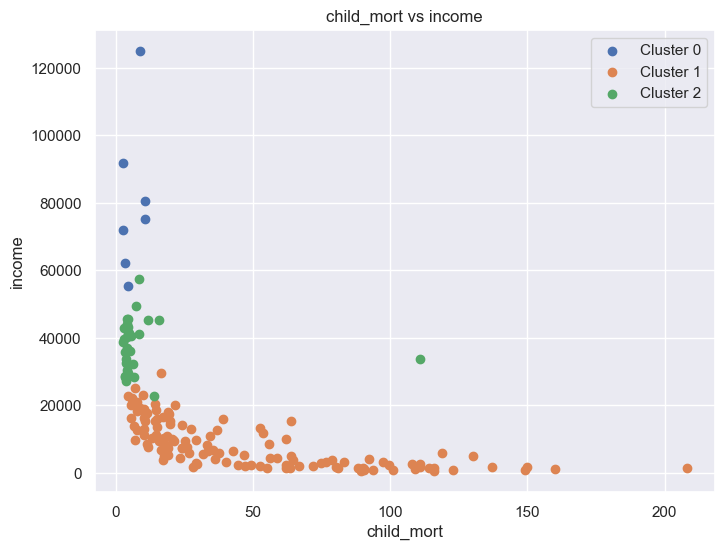
print(needy\_countries)

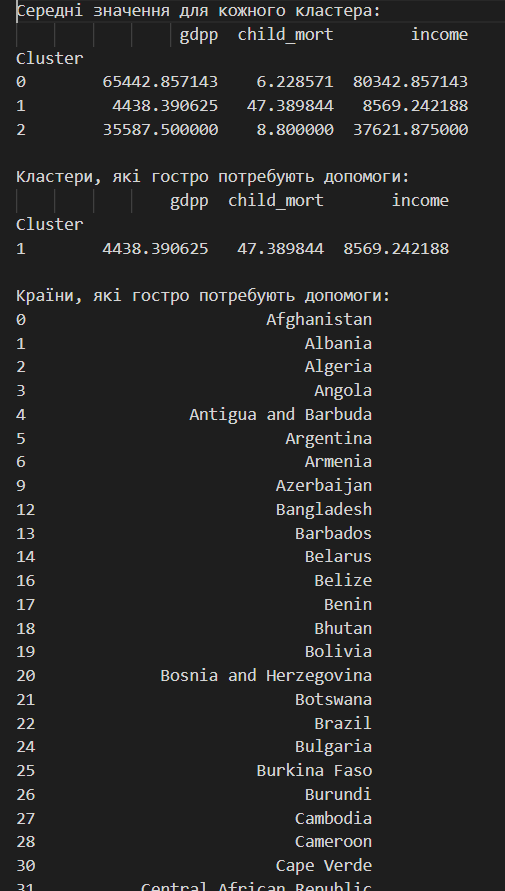
Результат:











5. Оптимізував k, використовуючи kneed:

import pandas as pd

import numpy as np

from sklearn.preprocessing import StandardScaler

from kneed import KneeLocator

from sklearn.cluster import KMeans

import matplotlib.pyplot as plt

data = pd.read\_csv('Country-data.csv')

features = data.drop('country', axis=1)

scaler = StandardScaler()

scaled\_features = scaler.fit\_transform(features)

# визначення оптимального k за допомогою kneed

sse = []

for k in range(1, 11):

    kmeans = KMeans(n\_clusters=k, random\_state=42)

    kmeans.fit(scaled\_features)

    sse.append(kmeans.inertia\_)

kl = KneeLocator(range(1, 11), sse, curve="convex", direction="decreasing")

optimal\_k = kl.elbow

plt.figure(figsize=(8, 6))

plt.plot(range(1, 11), sse, marker='o')

plt.xlabel('Number of clusters (k)')

plt.ylabel('Sum of squared distances')

plt.title('Elbow Method for Optimal k')

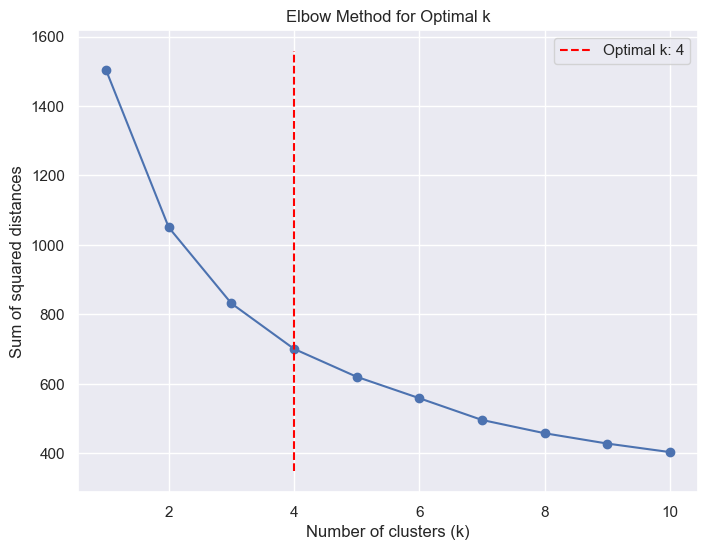
plt.vlines(optimal\_k, plt.ylim()[0], plt.ylim()[1], linestyles='dashed', colors='red', label=f'Optimal k: {optimal\_k}')

plt.legend()

plt.show()

print(f'Optimal k is: {optimal\_k}')

Результат:





6. Візуалізував результати кластеризації з оптимальним значенням k(4):

import pandas as pd

import matplotlib.pyplot as plt

from sklearn.cluster import KMeans

pd.set\_option('display.max\_rows', None)

data = pd.read\_csv('Country-data.csv')

features = ['gdpp', 'child\_mort', 'income']

optimal\_k = 4

kmeans = KMeans(n\_clusters=optimal\_k, random\_state=42)

data['Cluster'] = kmeans.fit\_predict(data[features])

# візуалізація кластерів

for i, col1 in enumerate(features):

    for j, col2 in enumerate(features):

        if i < j:

            plt.figure(figsize=(8, 6))

            for cluster in range(optimal\_k):

                cluster\_data = data[data['Cluster'] == cluster]

                plt.scatter(cluster\_data[col1], cluster\_data[col2], label=f'Cluster {cluster}')

            plt.xlabel(col1)

            plt.ylabel(col2)

            plt.title(f'{col1} vs {col2}')

            plt.legend()

            plt.show()

# аналіз кластерів

cluster\_means = data.groupby('Cluster')[features].mean()

print("Середні значення для кожного кластера:")

print(cluster\_means)

# розглядаємо кластери з низьким GDPP, високим child\_mort та низьким income як ті, що потребують допомоги:

needy\_clusters = cluster\_means[(cluster\_means['gdpp'] < cluster\_means['gdpp'].median()) &

                               (cluster\_means['child\_mort'] > cluster\_means['child\_mort'].median()) &

                               (cluster\_means['income'] < cluster\_means['income'].median())]

print("\nКластери, які гостро потребують допомоги:")

print(needy\_clusters)

# Вивід країн, які гостро потребують допомоги для кожного кластера

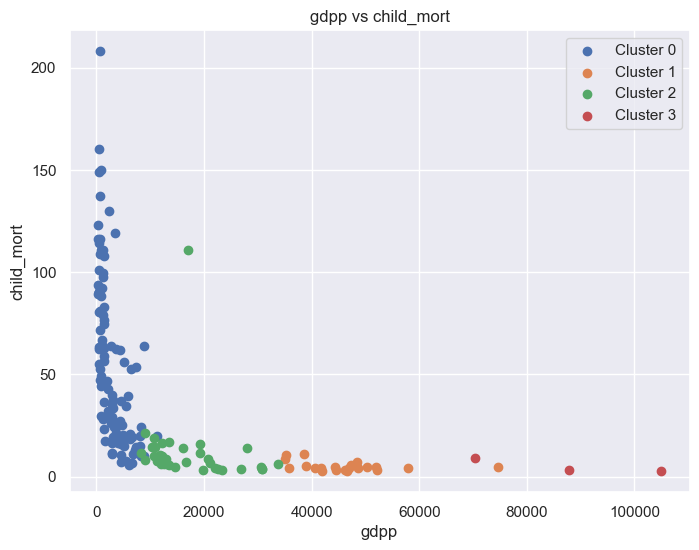
for cluster\_index in needy\_clusters.index:

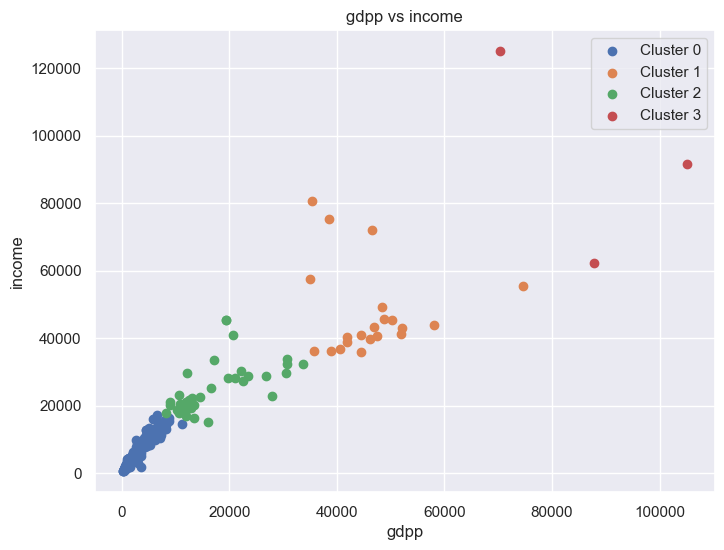
    cluster\_countries = data[data['Cluster'] == cluster\_index]['country']

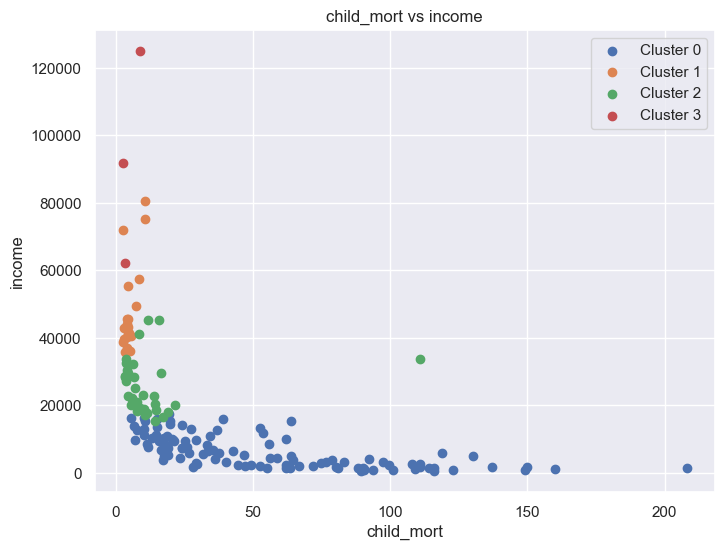
    print(f"\nКраїни у кластері {cluster\_index}, які гостро потребують допомоги:")

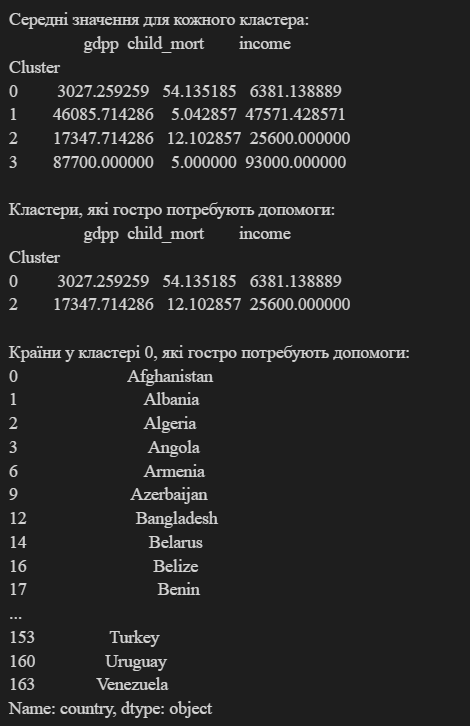
    print(cluster\_countries)

Результат:









7. Зробив summary (надав 10 країн, які найбільше потребують допомоги в результаті аналізу):

import pandas as pd

from sklearn.cluster import KMeans

data = pd.read\_csv('Country-data.csv')

features = ['gdpp', 'child\_mort', 'income']

optimal\_k = 4

kmeans = KMeans(n\_clusters=optimal\_k, random\_state=42)

data['Cluster'] = kmeans.fit\_predict(data[features])

cluster\_means = data.groupby('Cluster')[features].mean()

needy\_clusters = cluster\_means[(cluster\_means['gdpp'] < cluster\_means['gdpp'].median()) &

                               (cluster\_means['child\_mort'] > cluster\_means['child\_mort'].median()) &

                               (cluster\_means['income'] < cluster\_means['income'].median())]

# обираємо країни, які гостро потребують допомоги для кожного кластера

needy\_countries = []

for cluster\_index in needy\_clusters.index:

    cluster\_countries = data[data['Cluster'] == cluster\_index]['country']

    needy\_countries.extend(cluster\_countries)

# 10 країн, які найбільше потребують допомоги

top\_needy\_countries = data[data['country'].isin(needy\_countries)]

top\_needy\_countries = top\_needy\_countries.sort\_values(by=features, ascending=[True, False, True]).head(10)

print("Топ 10 країн, які найбільше потребують допомоги:")

print(top\_needy\_countries[['country', 'gdpp', 'child\_mort', 'income']])

Результат:



**Висновок:**

В ході виконання лабораторної роботи отримав практичні навички з виконання кластеризації даних, використовуючи метод k-mean та інші з пакету sklearn.