Department of Computer Engineering

**Academic Term: July-November 2023**

Rubrics for Lab Experiments

|  |  |  |
| --- | --- | --- |
| **Class** | **: B*.E. Computer*** | **Subject Name :*BDA*** |
| **Semester** | **: VII** | **Subject Code :** |

|  |  |
| --- | --- |
| **Practical No:** | 9 |
| **Title:** | Write a program to implement k-Means algorithm using any programming language. |
| **Date of Performance:** | 26/09/2023 |
| **Roll No:** | 9427 |
| **Name of the Student:** | Atharva Prashant Pawar |

**Evaluation:**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Performance Indicator** | **Below average** | **Average** | **Good** | **Excellent** | **Marks** |
| **On time Submission (2)** | Not submitted(0) | Submitted after deadline  (1) | Early or on time submission(2) | --- |  |
| **Test cases and** | Incorrect | The expected | The expected | Expected output is |  |
| **output** | output (1) | output is | output is Verified | obtained for all test |
| **(4)** |  | verified only a | for all test cases | cases. Presentable and |
|  |  | for few test | but is | easy to follow (4) |
|  |  | cases (2) | not presentable (3) |  |
| **Coding** | The code is not | The code is | The code is | - |  |
| **efficiency (2)** | structured at all | structured but | structured |  |
|  | (0) | not efficient (1) | and |  |
|  |  |  | efficient. (2) |  |
| **Knowledge(2)** | Basic concepts not clear  (0) | Understood the basic concepts (1) | Could explain the concept with  suitable example (1.5) | Could relate the theory with real world  application(2) |  |
| **Total** |  | | | | |

**Signature of the Teacher :**

# Importing libraries

import pandas as pd import numpy as np

import matplotlib.pyplot as plt

%matplotlib inline

import seaborn as sns

pd.options.display.float\_format = '{:.2f}'.format

from sklearn.cluster import KMeans

from sklearn.metrics import silhouette\_score from mpl\_toolkits.mplot3d import Axes3D

import plotly.express as px # !pip install -U kaleido

import kaleido

# Loading the dataset

dict\_df = pd.read\_csv("data-dictionary.csv") data = pd.read\_csv("Country-data.csv")

dict\_df

**Column Name Description**

1. country Name of the country 
2. child\_mort Death of children under 5 years of age per 100...
3. exports Exports of goods and services per capita. Give...
4. health Total health spending per capita. Given as %ag...
5. imports Imports of goods and services per capita. Give...
6. Income Net income per person
7. Inflation The measurement of the annual growth rate of t...
8. life\_expec The average number of years a new born child w...
9. total\_fer The number of children that would be born to e...
10. gdpp The GDP per capita. Calculated as the Total GD...

data.shape

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| data.head()  **country** | **child\_mort** | **exports** | **health** | **imports** | **income** | **inflation** | **life\_expec** | **total\_fer** | **gdpp** |
| **0** Afghanistan | 90.20 | 10.00 | 7.58 | 44.90 | 1610 | 9.44 | 56.20 | 5.82 | 553 |
| **1** Albania | 16.60 | 28.00 | 6.55 | 48.60 | 9930 | 4.49 | 76.30 | 1.65 | 4090 |
| **2** Algeria | 27.30 | 38.40 | 4.17 | 31.40 | 12900 | 16.10 | 76.50 | 2.89 | 4460 |
| **3** Angola | 119.00 | 62.30 | 2.85 | 42.90 | 5900 | 22.40 | 60.10 | 6.16 | 3530 |
| **4** Antigua and Barbuda | 10.30 | 45.50 | 6.03 | 58.90 | 19100 | 1.44 | 76.80 | 2.13 | 12200 |
| EDA |  |  |  |  |  |  |  |  |  |

(167, 10)

data.info()

<class 'pandas.core.frame.DataFrame'> RangeIndex: 167 entries, 0 to 166

Data columns (total 10 columns):

# Column Non-Null Count Dtype

1. country 167 non-null object
2. child\_mort 167 non-null float64
3. exports 167 non-null float64
4. health 167 non-null float64
5. 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** | **inflation** | **life\_expec** | **total\_fer** | **gdpp** |
| **count** | 167.00 | 167.00 | 167.00 | 167.00 | 167.00 | 167.00 | 167.00 | 167.00 | 167.00 |
| **mean** | 38.27 | 41.11 | 6.82 | 46.89 | 17144.69 | 7.78 | 70.56 | 2.95 | 12964.16 |
| **std** | 40.33 | 27.41 | 2.75 | 24.21 | 19278.07 | 10.57 | 8.89 | 1.51 | 18328.70 |
| **min** | 2.60 | 0.11 | 1.81 | 0.07 | 609.00 | -4.21 | 32.10 | 1.15 | 231.00 |
| **25%** | 8.25 | 23.80 | 4.92 | 30.20 | 3355.00 | 1.81 | 65.30 | 1.79 | 1330.00 |
| **50%** | 19.30 | 35.00 | 6.32 | 43.30 | 9960.00 | 5.39 | 73.10 | 2.41 | 4660.00 |
| **75%** | 62.10 | 51.35 | 8.60 | 58.75 | 22800.00 | 10.75 | 76.80 | 3.88 | 14050.00 |
| **max** | 208.00 | 200.00 | 17.90 | 174.00 | 125000.00 | 104.00 | 82.80 | 7.49 | 105000.00 |

data.isnull().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

print("No of countries present: ",data['country'].nunique()) No of countries present: 167

col = list(data.columns) col.remove('country')

categorical\_features = ['country'] numerical\_features = [\*col]

print('Categorical Features :',\*categorical\_features) print('Numerical Features :',\*numerical\_features)

Categorical Features : country

Numerical Features : child\_mort exports health imports income inflation life\_expec total\_fer gdpp

fig, ax = plt.subplots(nrows = 3,ncols = 3,figsize = (15,15)) colors = ['#FF781F','#2D2926']

for i in range(len(numerical\_features)): plt.subplot(3,3,i+1)

sns.histplot(data[numerical\_features[i]],color = colors[0]) title = 'Distribution : ' + numerical\_features[i]

plt.title(title) plt.show()

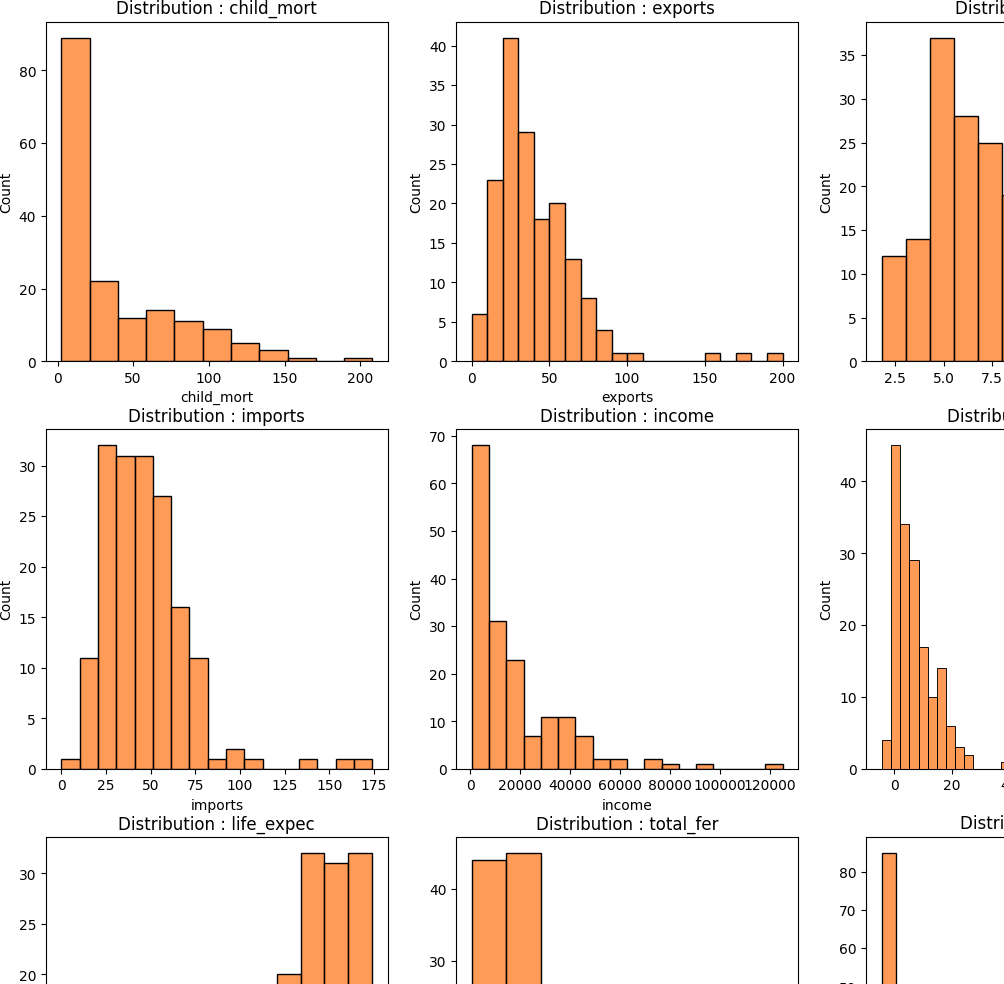
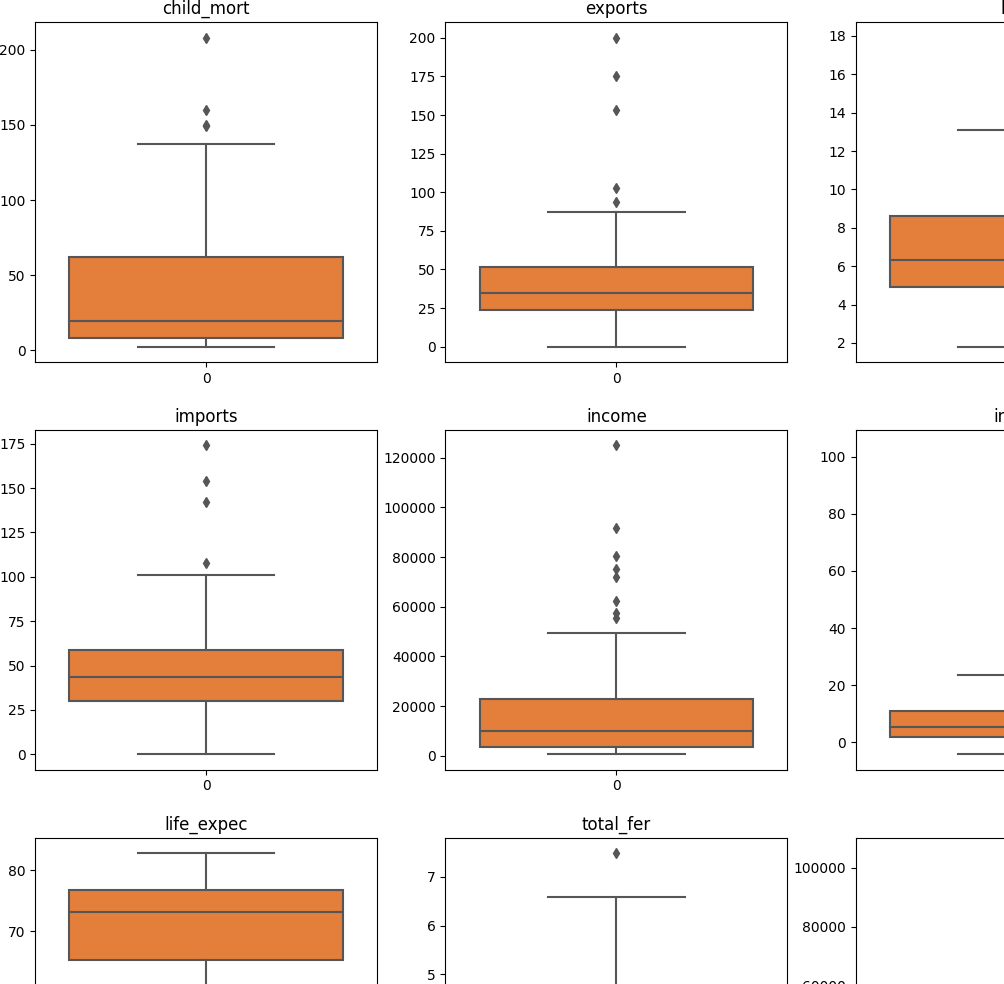


fig = plt.subplots(nrows = 3,ncols = 3,figsize = (15,15)) for i in range(len(numerical\_features)):

plt.subplot(3,3,i+1)

ax = sns.boxplot(data=data[numerical\_features[i]],color = colors[0]) plt.title(numerical\_features[i])

plt.show()

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

sns.heatmap(data.corr(), annot=True) plt.title("Heatmap")

plt.show()

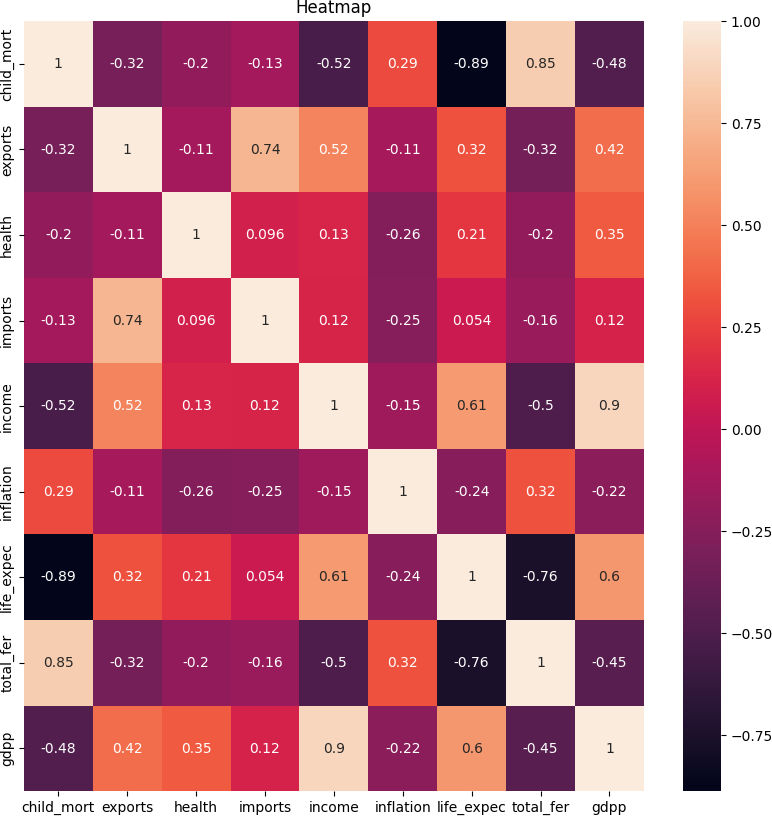
<ipython-input-41-938d23771fea>:2: FutureWarning:

The default value of numeric\_only in DataFrame.corr is deprecated. In a future version, it will default to False. Select only valid

|  |  |  |
| --- | --- | --- |
| **Health** | **Trade** | **Finance** |
| **0** 6.24 | 1.20 | 1.35 |
| **1** 3.04 | 1.72 | 1.47 |
| **2** 3.39 | 1.60 | 3.17 |
| **3** 6.47 | 2.43 | 3.49 |
| **4** 2.96 | 2.36 | 2.24 |

|  |  |  |  |
| --- | --- | --- | --- |
| **Country** | **Health** | **Trade** | **Finance** |
| **0** Afghanistan | 0.63 | 0.14 | 0.08 |
| **1** Albania | 0.13 | 0.20 | 0.09 |
| **2** Algeria | 0.18 | 0.19 | 0.21 |
| **3** Angola | 0.66 | 0.28 | 0.24 |
| **4** Antigua and Barbuda | 0.12 | 0.28 | 0.15 |

df2 = data.copy(deep = True)



df1 = pd.DataFrame()

df1['Health'] = (data['child\_mort'] / data['child\_mort'].mean()) + (data['health'] / data['health'].mean()) + (data['life\_expec'] / data[ df1['Trade'] = (data['imports'] / data['imports'].mean()) + (data['exports'] / data['exports'].mean())

df1['Finance'] = (data['income'] / data['income'].mean()) + (data['inflation'] / data['inflation'].mean()) + (data['gdpp'] / data['gdpp'] df1.head()

from sklearn.preprocessing import MinMaxScaler,StandardScaler

mms = MinMaxScaler() ss = StandardScaler()

df1['Health'] = mms.fit\_transform(df1[['Health']]) df1['Trade'] = mms.fit\_transform(df1[['Trade']])

df1['Finance'] = mms.fit\_transform(df1[['Finance']])

df1.insert(loc = 0, value = list(data['country']), column = 'Country') df1.head()

col = list(data.columns)

col.remove('health'); col.remove('country')

df2['health'] = ss.fit\_transform(df2[['health']]) # Standardization for i in col:

df2[i] = mms.fit\_transform(df2[[i]]) # Normalization df2.drop(columns = 'country',inplace = True)

df2.head()

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **child\_mort** | **exports** | **health** | **imports** | **income** | **inflation** | **life\_expec** | **total\_fer** | **gdpp** |
| **0** 0.43 | 0.05 | 0.28 | 0.26 | 0.01 | 0.13 | 0.48 | 0.74 | 0.00 |
| **1** 0.07 | 0.14 | -0.10 | 0.28 | 0.07 | 0.08 | 0.87 | 0.08 | 0.04 |
| **2** 0.12 | 0.19 | -0.97 | 0.18 | 0.10 | 0.19 | 0.88 | 0.27 | 0.04 |
| **3** 0.57 | 0.31 | -1.45 | 0.25 | 0.04 | 0.25 | 0.55 | 0.79 | 0.03 |
| **4** 0.04 | 0.23 | -0.29 | 0.34 | 0.15 | 0.05 | 0.88 | 0.15 | 0.11 |

from sklearn.decomposition import PCA pca = PCA()

pca\_df2 = pd.DataFrame(pca.fit\_transform(df2)) pca.explained\_variance\_

array([1.01740511, 0.13090418, 0.03450018, 0.02679822, 0.00979752,

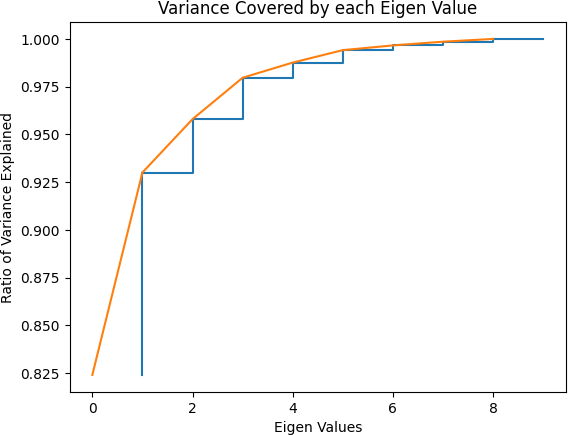
0.00803398, 0.00307055, 0.00239976, 0.00179388])

plt.step(list(range(1,10)), 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.show()



pca\_df2 = pca\_df2.drop(columns = [3,4,5,6,7,8]) pca\_df2.head()

|  |  |  |
| --- | --- | --- |
| **0** | **1** | **2** |
| **0** 0.22 | 0.64 | 0.09 |
| **1** -0.08 | -0.17 | -0.21 |
| **2** -0.96 | -0.13 | -0.13 |
| **3** -1.51 | 0.47 | 0.28 |
| **4** -0.26 | -0.24 | -0.06 |

m1 = df1.drop(columns = ['Country']).values # Feature Combination : Health - Trade - Finance m2 = pca\_df2.values # PCA Data

sse = {};sil = [];kmax = 10

fig = plt.subplots(nrows = 1, ncols = 1, figsize = (20,5))

# Elbow Method :

plt.subplot(1,2,1)

for k in range(1, 10):

kmeans = KMeans(n\_clusters=k, max\_iter=1000).fit(m1)

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()

<ipython-input-49-4566498423fd>:5: MatplotlibDeprecationWarning:

Auto-removal of overlapping axes is deprecated since 3.6 and will be removed two minor releases later; explicitly call ax.remove() a

/usr/local/lib/python3.10/dist-packages/sklearn/cluster/\_kmeans.py:870: FutureWarning:

The default value of `n\_init` will change from 10 to 'auto' in 1.4. Set the value of `n\_init` explicitly to suppress the warning

/usr/local/lib/python3.10/dist-packages/sklearn/cluster/\_kmeans.py:870: FutureWarning:

The default value of `n\_init` will change from 10 to 'auto' in 1.4. Set the value of `n\_init` explicitly to suppress the warning

/usr/local/lib/python3.10/dist-packages/sklearn/cluster/\_kmeans.py:870: FutureWarning:

The default value of `n\_init` will change from 10 to 'auto' in 1.4. Set the value of `n\_init` explicitly to suppress the warning

/usr/local/lib/python3.10/dist-packages/sklearn/cluster/\_kmeans.py:870: FutureWarning:

The default value of `n\_init` will change from 10 to 'auto' in 1.4. Set the value of `n\_init` explicitly to suppress the warning

/usr/local/lib/python3.10/dist-packages/sklearn/cluster/\_kmeans.py:870: FutureWarning:

The default value of `n\_init` will change from 10 to 'auto' in 1.4. Set the value of `n\_init` explicitly to suppress the warning

/usr/local/lib/python3.10/dist-packages/sklearn/cluster/\_kmeans.py:870: FutureWarning:

The default value of `n\_init` will change from 10 to 'auto' in 1.4. Set the value of `n\_init` explicitly to suppress the warning

/usr/local/lib/python3.10/dist-packages/sklearn/cluster/\_kmeans.py:870: FutureWarning:

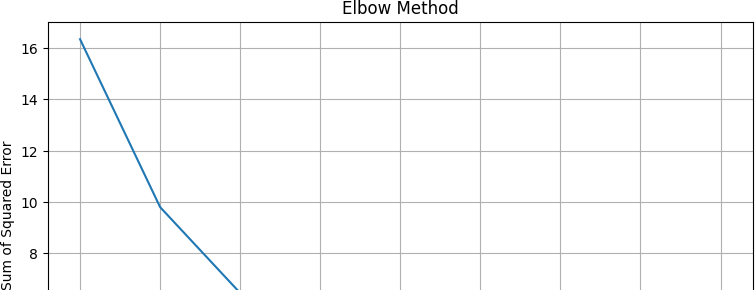
The default value of `n\_init` will change from 10 to 'auto' in 1.4. Set the value of `n\_init` explicitly to suppress the warning

/usr/local/lib/python3.10/dist-packages/sklearn/cluster/\_kmeans.py:870: FutureWarning:

The default value of `n\_init` will change from 10 to 'auto' in 1.4. Set the value of `n\_init` explicitly to suppress the warning

/usr/local/lib/python3.10/dist-packages/sklearn/cluster/\_kmeans.py:870: FutureWarning:

The default value of `n\_init` will change from 10 to 'auto' in 1.4. Set the value of `n\_init` explicitly to suppress the warning



model = KMeans(n\_clusters = 3,max\_iter = 1000) model.fit(m1)

cluster = model.cluster\_centers\_ centroids = np.array(cluster)

labels = model.labels\_

data['Class'] = labels; df1['Class'] = labels

fig = plt.figure() ax = Axes3D(fig)

x = np.array(df1['Health']) y = np.array(df1['Trade']) z = np.array(df1['Finance'])

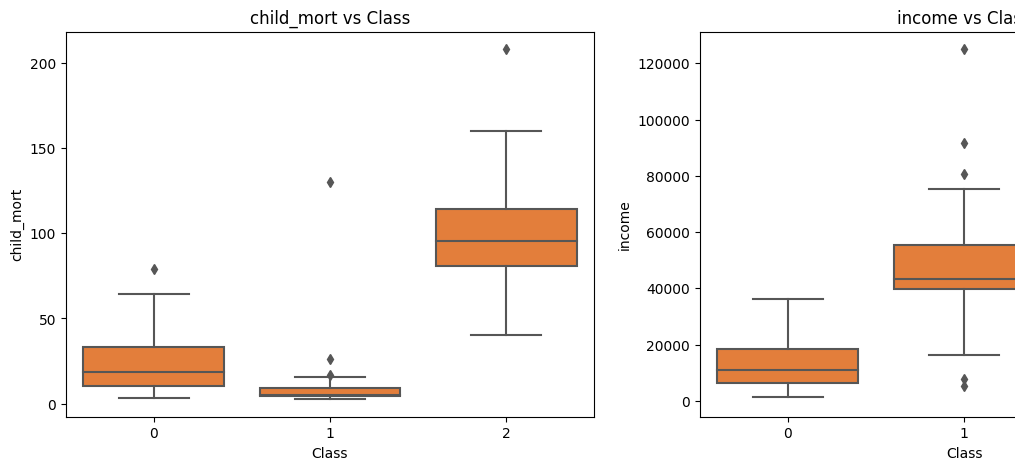
ax.scatter(centroids[:,0],centroids[:,1],centroids[:,2],marker="X", color = 'b') ax.scatter(x,y,z,c = y)

plt.title('Health vs Trade vs Finance') ax.set\_xlabel('Health')

ax.set\_ylabel('Trade') ax.set\_zlabel('Finance') plt.show()

/usr/local/lib/python3.10/dist-packages/sklearn/cluster/\_kmeans.py:870: FutureWarning:

The default value of `n\_init` will change from 10 to 'auto' in 1.4. Set the value of `n\_init` explicitly to suppress the warning



fig, ax = plt.subplots(nrows = 1, ncols = 2, figsize = (15,5)) plt.subplot(1,2,1)

sns.boxplot(x = 'Class', y = 'child\_mort', data = data, color = '#FF781F'); plt.title('child\_mort vs Class')

plt.subplot(1,2,2)

sns.boxplot(x = 'Class', y = 'income', data = data, color = '#FF781F'); plt.title('income vs Class')

plt.show()

## From the above plot we can conclude :

0 : No Help Needed 1 : Help Needed

## 2 : Might Need Help

df1['Class'].loc[df1['Class'] == 0] = 'No Help Needed' df1['Class'].loc[df1['Class'] == 1] = 'Help Needed'

df1['Class'].loc[df1['Class'] == 2] = 'Might Need Help'

fig = px.choropleth(df1[['Country','Class']],

locationmode = 'country names', locations = 'Country',

title = 'Needed Help Per Country (World)', color = df1['Class'],

color\_discrete\_map = {'Help Needed':'Red',

'No Help Needed':'Green', 'Might Need Help':'Yellow'}

)

fig.update\_geos(fitbounds = "locations", visible = True)

fig.update\_layout(legend\_title\_text = 'Labels',legend\_title\_side = 'top',title\_pad\_l = 260,title\_y = 0.86) fig.show(engine = 'kaleido')

<ipython-input-55-a313759b4d1f>:1: SettingWithCopyWarning:

A value is trying to be set on a copy of a slice from a DataFrame

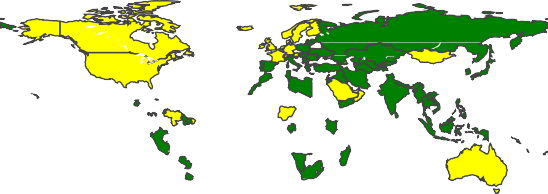
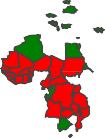
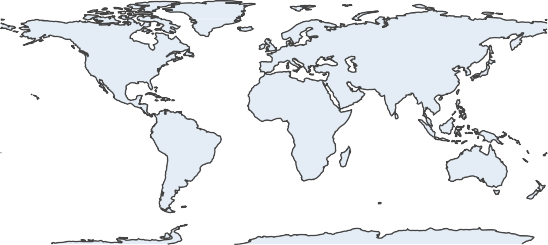
See the caveats in the documentation: [https://pandas.pydata.org/pandas-docs/stable/u](https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy)s

Needed Help Per Country (World)

Labels

 Help Needed

No Help Needed Might Need Help



Postlab:

