K-Mean

The K-Means algorithm is a simple algorithm capable of clustering this unlabeled dataset very quickly and efficiently, often in just a few iterations. It is partitional clustering approach and each cluster is associated with a centroid (center point). Each point is assigned to the cluster with the closest centroid.

a) Perform standard data cleaning operations such as data cleaning (handling missing values)

[20]

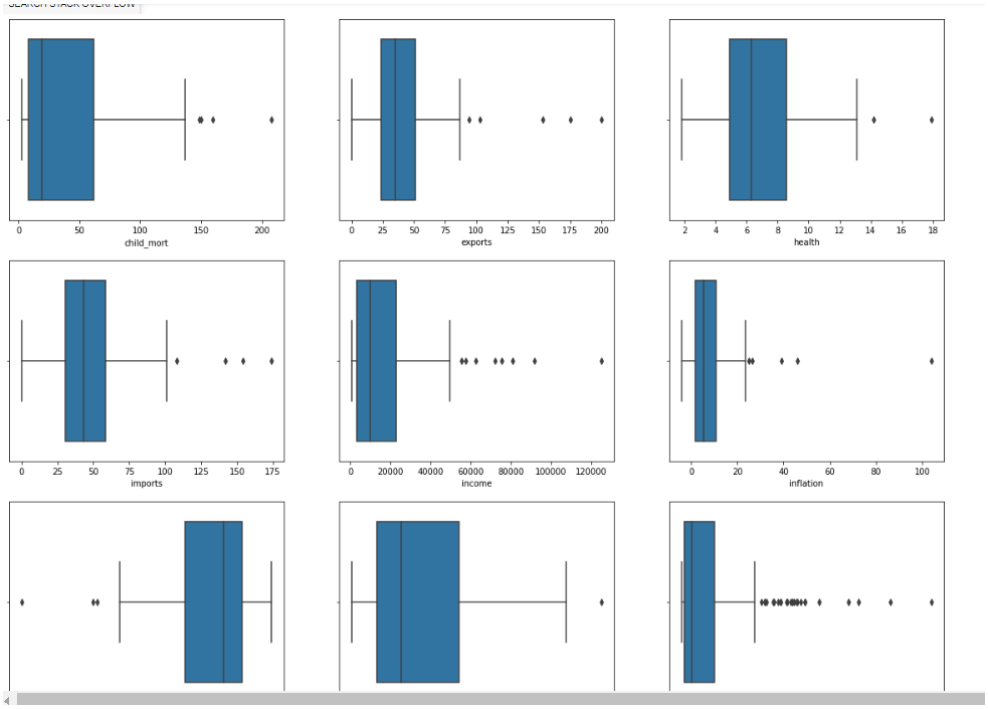
	country	child_mort	exports	health	imports	income	inflation	life_expec	total_fer	gdpp
0	Afghanistan	90.2	10.0	7.58	44.9	1610	9.440	56.2	5.82	553
1	Albania	16.6	28.0	6.55	48.6	9930	4.490	76.3	1.65	4090
2	Algeria	27.3	38.4	4.17	31.4	12900	16.100	76.5	2.89	4460
3	Angola	119.0	62.3	2.85	42.9	5900	22.400	60.1	6.16	3530
4	Antigua and Barbuda	10.3	45.5	6.03	58.9	19100	1.440	76.8	2.13	12200
5	Argentina	14.5	18.9	8.10	16.0	18700	20.900	75.8	2.37	10300
6	Armenia	18.1	20.8	4.40	45.3	6700	7.770	73.3	1.69	3220
7	Australia	4.8	19.8	8.73	20.9	41400	1.160	82.0	1.93	51900
8	Austria	4.3	51.3	11.00	47.8	43200	0.873	80.5	1.44	46900
9	Azerbaijan	39.2	54.3	5.88	20.7	16000	13.800	69.1	1.92	5840

```
df.isnull().sum()
```

```
child_mort        0
exports           0
health            0
imports           0
income            0
inflation         0
life_expec        0
total_fer         0
gdpp              0
Cluster           0
cluster_kmeans    0
K_means_labels    0
dtype: int64
```

By performing data cleaning there is no any missing values.

b) data scaling (handling the outliers)



During data scaling I show the outliers but I could not remove those outlier because if I am do then many developed country are removed from analysis.

Silhouette score

```
from sklearn.metrics import silhouette_score
range_n_clusters = [3]

for num_clusters in range_n_clusters:

    # intialise kmeans
    kmeans = KMeans(n_clusters=num_clusters, max_iter=50)
    kmeans.fit(X)

    cluster_labels = kmeans.labels_

    # silhouette score
    silhouette_avg = silhouette_score(X, cluster_labels)
    print("For n_clusters={0}, the silhouette score is {1}".format(num_clusters, silhouette_avg))
```

```
For n_clusters=3, the silhouette score is 0.28329575683463126
```

By calculating Silhouette score for 3 cluster = 0.2832

Classify the countries according to the following categories:
■ Developed Country
■ Developing Country
■ Under-Developing Country

```
[30] df['Cluster']=y_hc
```

```
df['Cluster'].value_counts()
```

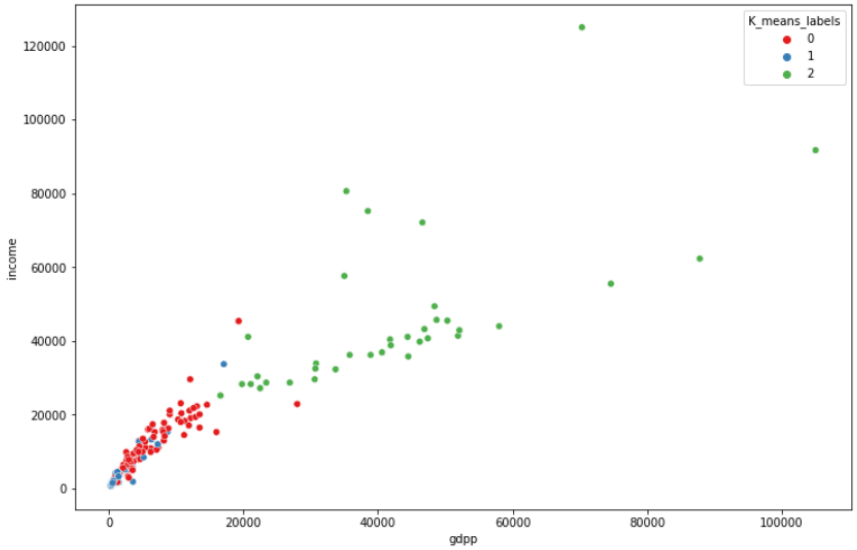
```
1    84
0    47
2    36
Name: Cluster, dtype: int64
```

By applying Elbow method, I found number of clusters is 3 and on the basis of this
Cluster 0 = under developed country and numbers are 47
Cluster 1 = developing country and numbers are 84
Cluster 2= developed country and numbers are 36

Visualization of clusters

```
plt.figure(figsize=[12,8])
sns.scatterplot(df.gdpp, df.income,hue = df.K_means_labels, palette='Set1')
plt.show()
```

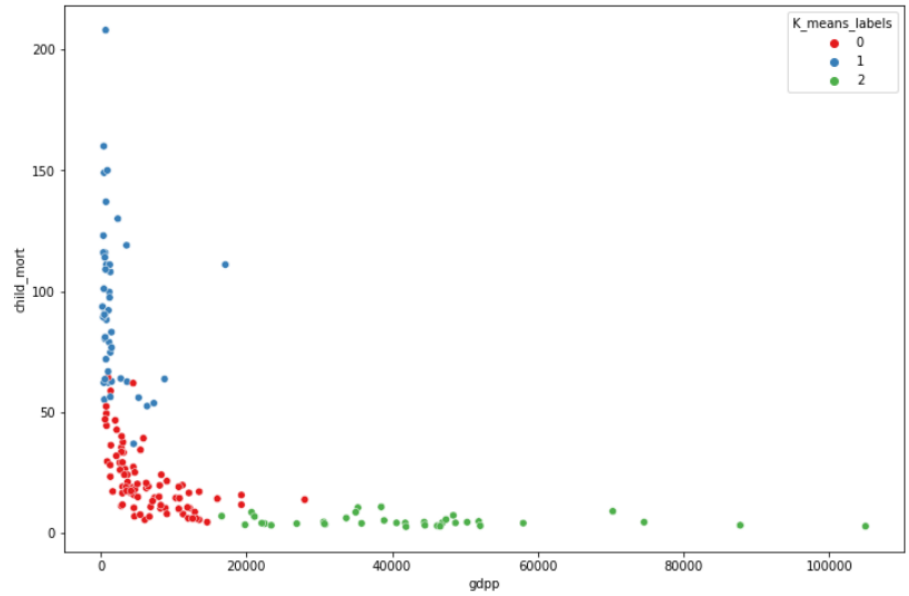
```
/usr/local/lib/python3.7/dist-packages/seaborn/_decorators.py:43: FutureWarning: Pass the following variables as keyword args: x,
  FutureWarning
```



This cluster show INCOME vs GDP and this shows as income increases GDP also increases.

```
plt.figure(figsize=[12,8])
sns.scatterplot(df.gdpp, df.child_mort,hue = df.K_means_labels, palette='Set1')
plt.show()
```

```
/usr/local/lib/python3.7/dist-packages/seaborn/_decorators.py:43: FutureWarning: Pass the following variables as keywor
  FutureWarning
```



K-medoid

This method is very sensitive to outliers than k-means and each cluster is represented by the most central object in the cluster.

Silhouette score

```
from sklearn.metrics import silhouette_score
range_n_clusters = [3]

for num_clusters in range_n_clusters:

    # intialise kmeans
    kmedoids = KMedoids(n_clusters=num_clusters, max_iter=50)
    kmedoids.fit(X)

    cluster_labels = kmedoids.labels_

    # silhouette score
    silhouette_avg = silhouette_score(X, cluster_labels)
    print("For n_clusters={0}, the silhouette score is {1}".format(num_clusters, silhouette_avg))
```

```
For n_clusters=3, the silhouette score is 0.1562250700966545
```

By calculating Silhouette score for 3 cluster = 0.15622

Classify the countries according to the following categories:
■ Developed Country
■ Developing Country
■ Under-Developing Country

```
df['Cluster'].value_counts()
```

```
0    75
1    47
2    45
Name: Cluster, dtype: int64
```

By applying Elbow method, I found number of clusters is 3 and on the basis of this
Cluster 0 = developed country and numbers are 75
Cluster 1 = under developing country and numbers are 47
Cluster 2= developing country and numbers are 45

Visualization of clusters