KMedoid

Target:- The assignment targets to implement K-Means and K-Medoid algorithms to cluster the dataset consists of socio-economic and health factors of countries and determine the overall development of the country

Starting of the program

Imported all important libraries numpy,pandas,matplotlib for plotting final graphs of clusters,KMedoids from sklearn

In [1]:
```
import numpy as np
from matplotlib import pyplot as plt
import pandas as pd
from sklearn.cluster import KMeans
from sklearn_extra.cluster import KMedoids
from sklearn.preprocessing import MinMaxScaler
from sklearn.metrics import silhouette_score
```

Reading csv and scaling

pd.read_csv to read the file given index was set to child_mort as 0th RS minmax scaler was used to scale the data

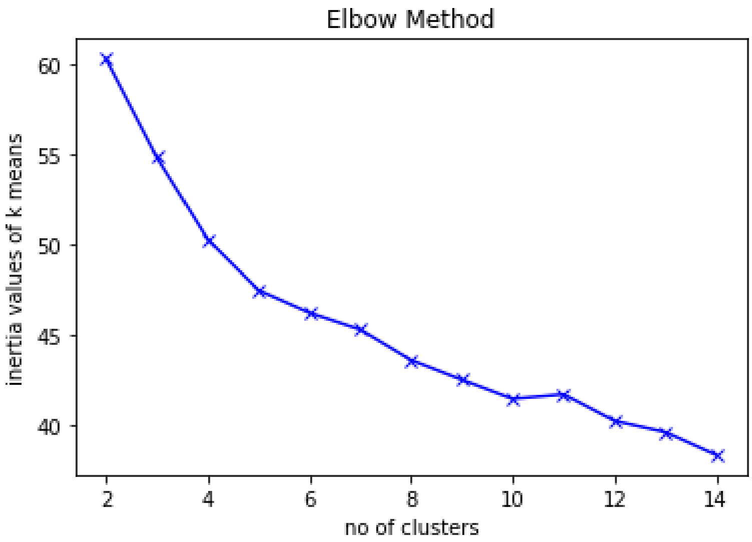
In [2]:
```
df=pd.read_csv('C:\Python\Assignments\MLAssignment\Country_data.csv')
df.set_index("country",
              inplace = True)
RS=MinMaxScaler()
colors=['fuchsia','aqua','gold','brown','gray','black','r','b','g']
def scale_data():
    for x in df.columns[0:9]:
        RS.fit(np.array(df[x]).reshape(-1,1))
        df[x]=RS.transform(np.array(df[x]).reshape(-1,1))
        df[x].round(2)
scale_data()
```

silhouette and elbow scores

plot_scores() will plot silhoette and elbow graphs on two seperate figures

In [3]:
```
def plot_scores():
    SSO={'silhouette_score':[],'elbow':[]}
    for i in range(2,15):
        kmedoid=KMedoids(n_clusters=i)
        a=kmedoid.fit_predict(df[df.columns[0:9]])
        SSO['elbow'].append(kmedoid.inertia_)
        SSO['silhouette_score'].append(silhouette_score(df[df.columns[0:9]]
                                                         ,a))
    plt.title("Elbow Method")
    plt.plot(list(range(2,15)),SSO['elbow'],marker='x',color='b',label='elbow')
    plt.xlabel('no of clusters')
    plt.ylabel('inertia values of k means')
    plt.show()
    plt.title("Silhouette Score")
```

```
    plt.plot(list(range(2,15)),SSO['silhouette_score'],
             marker='x',color='r',label='silhouette_score')
    plt.xlabel('no of clusters')
    plt.ylabel('silhouette scores of k means')
    plt.show()
plot_scores()
```



finding centroids using kmedoids

kmedoids library was used to define clusters and stores them in cluster_list

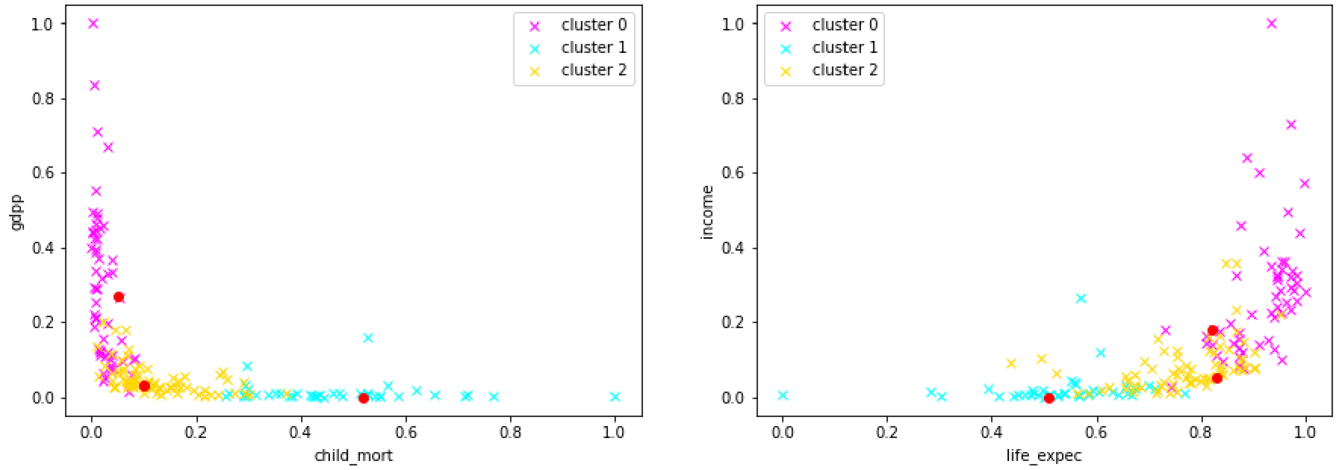
In [4]:
```
def kmedoids_library():
    Kmedoids=KMedoids(n_clusters=3)
    Kmedoids.fit(df[df.columns[0:9]])
    clusters_list=Kmedoids.predict(df[df.columns[0:9]])
    centroid=Kmedoids.cluster_centers_.round(2)

    return clusters_list,centroid
clusters_list,centroid=kmedoids_library()
number_of_clusters=3
```

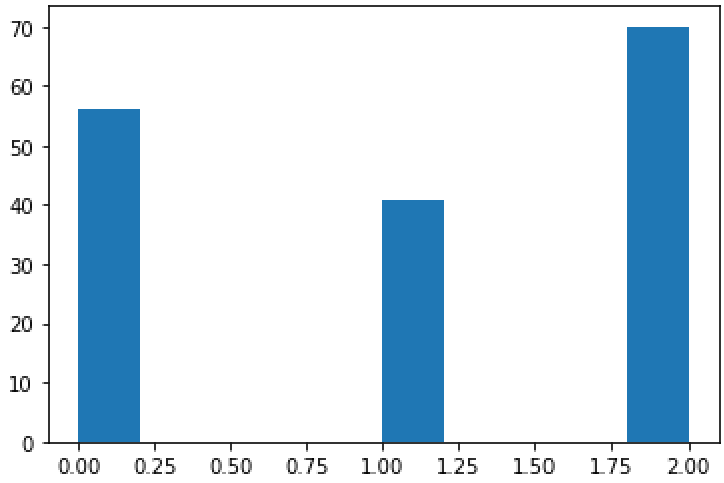
Plotting the final result

both child_mort and gdpp plotted

In [5]:
```
def plot():
    axs=[]
```

```
    df['cluster']=clusters_list
    fig, axs=plt.subplots(1,2,figsize=(15,5))
    cluster=[0,1,2]
    for k in range(number_of_clusters):
        axs[0].plot(df[df['cluster']==cluster[k]]['child_mort'],
                    df[df['cluster']==cluster[k]]['gdpp'],'x',color=colors[k],
                    label=f'cluster {cluster[k]}')
        axs[1].plot(df[df['cluster']==cluster[k]]['life_expec'],
                    df[df['cluster']==cluster[k]]['income'],'x',color=colors[k],
                    label=f'cluster {cluster[k]}')
    axs[0].plot(centroid[:,0],centroid[:,8],'o',color='r')
    axs[1].plot(centroid[:,6],centroid[:,4],'o',color='r')
    axs[0].set_xlabel('child_mort')
    axs[0].set_ylabel('gdpp')
    axs[1].set_xlabel('life_expec')
    axs[1].set_ylabel('income')
    axs[0].legend()
    axs[1].legend()
    plt.show()
    plt.hist(np.sort(df['cluster']))
    plt.show()
plot()
```





Final conlusion

cluster 1 has high child_mort, very low life_expec, income and gdpp with all other coinsiding features i will consider it as underdevelopped

cluster 2 has Normal child_mort, Normal Low life_expec, income and gdpp along with all other coinsiding deatuers i will considered it as developping

cluster 0 has Low child_mort, High life_expec, income and gdpp along with all other coinsiding deatuers i will considered it as developped

In [6]:
```
df['Nation status']=' '
df.loc[df['cluster']==1,'Nation status']='underdevelopped'
df.loc[df['cluster']==2,'Nation status']='developping'
df.loc[df['cluster']==0,'Nation status']='developped'
```

In [7]:
```
df.reset_index(inplace=True)
np.array(df[df['Nation status']=='developping']['country'])
```

Out[7]:
```
array(['Albania', 'Algeria', 'Antigua and Barbuda', 'Armenia',
       'Azerbaijan', 'Bangladesh', 'Belarus', 'Belize', 'Bhutan',
       'Bolivia', 'Botswana', 'Bulgaria', 'Cambodia', 'Cape Verde',
       'China', 'Dominican Republic', 'Ecuador', 'Egypt', 'El Salvador',
       'Estonia', 'Fiji', 'Georgia', 'Grenada', 'Guatemala', 'Guyana',
       'India', 'Indonesia', 'Iran', 'Iraq', 'Jamaica', 'Jordan',
       'Kazakhstan', 'Kiribati', 'Kyrgyz Republic', 'Lao', 'Libya',
       'Macedonia, FYR', 'Malaysia', 'Maldives', 'Malta', 'Mauritius',
       'Micronesia, Fed. Sts.', 'Mongolia', 'Morocco', 'Myanmar',
       'Namibia', 'Nepal', 'Oman', 'Panama', 'Paraguay', 'Peru',
       'Philippines', 'Samoa', 'Saudi Arabia', 'Seychelles',
       'Solomon Islands', 'South Africa', 'Sri Lanka',
       'St. Vincent and the Grenadines', 'Suriname', 'Tajikistan',
       'Thailand', 'Tonga', 'Tunisia', 'Turkmenistan', 'Ukraine',
       'Uzbekistan', 'Vanuatu', 'Venezuela', 'Vietnam'], dtype=object)
```

In [8]:
```
np.array(df[df['Nation status']=='developped']['country'])
```

Out[8]:
```
array(['Argentina', 'Australia', 'Austria', 'Bahamas', 'Bahrain',
       'Barbados', 'Belgium', 'Bosnia and Herzegovina', 'Brazil',
       'Brunei', 'Canada', 'Chile', 'Colombia', 'Costa Rica', 'Croatia',
       'Cyprus', 'Czech Republic', 'Denmark', 'Finland', 'France',
       'Germany', 'Greece', 'Hungary', 'Iceland', 'Ireland', 'Israel',
       'Italy', 'Japan', 'Kuwait', 'Latvia', 'Lebanon', 'Lithuania',
       'Luxembourg', 'Moldova', 'Montenegro', 'Netherlands',
       'New Zealand', 'Norway', 'Poland', 'Portugal', 'Qatar', 'Romania',
       'Russia', 'Serbia', 'Singapore', 'Slovak Republic', 'Slovenia',
       'South Korea', 'Spain', 'Sweden', 'Switzerland', 'Turkey',
       'United Arab Emirates', 'United Kingdom', 'United States',
       'Uruguay'], dtype=object)
```

In [9]:
```
np.array(df[df['Nation status']=='underdevelopped']['country'])
```

Out[9]:
```
array(['Afghanistan', 'Angola', 'Benin', 'Burkina Faso', 'Burundi',
       'Cameroon', 'Central African Republic', 'Chad', 'Comoros',
       'Congo, Dem. Rep.', 'Congo, Rep.', "Cote d'Ivoire",
       'Equatorial Guinea', 'Eritrea', 'Gabon', 'Gambia', 'Ghana',
       'Guinea', 'Guinea-Bissau', 'Haiti', 'Kenya', 'Lesotho', 'Liberia',
       'Madagascar', 'Malawi', 'Mali', 'Mauritania', 'Mozambique',
       'Niger', 'Nigeria', 'Pakistan', 'Rwanda', 'Senegal',
       'Sierra Leone', 'Sudan', 'Tanzania', 'Timor-Leste', 'Togo',
       'Uganda', 'Yemen', 'Zambia'], dtype=object)
```

In [10]:
```
dfcopy=pd.read_csv('C:\Python\Assignments\MLAssignment\country_data_Kmeans.csv')
display(dfcopy)
dfcopy['KMedoids Nation Status']=df['Nation status']
dfcopy.to_csv('C:\Python\Assignments\MLAssignment\country_data_Kmeans_Kmedoids.csv')
```

	Unnamed: 0	country	child_mort	exports	health	imports	income	inflation	life_expec	total_fer	g
0	0	Afghanistan	90.2	10.0	7.58	44.9	1610	9.44	56.2	5.82	
1	1	Albania	16.6	28.0	6.55	48.6	9930	4.49	76.3	1.65	4
2	2	Algeria	27.3	38.4	4.17	31.4	12900	16.10	76.5	2.89	4
3	3	Angola	119.0	62.3	2.85	42.9	5900	22.40	60.1	6.16	3
4	4	Antigua and Barbuda	10.3	45.5	6.03	58.9	19100	1.44	76.8	2.13	12
...	...	...	...	...	...	...	...	...	...	...	
162	162	Vanuatu	29.2	46.6	5.25	52.7	2950	2.62	63.0	3.50	2
163	163	Venezuela	17.1	28.5	4.91	17.6	16500	45.90	75.4	2.47	13
164	164	Vietnam	23.3	72.0	6.84	80.2	4490	12.10	73.1	1.95	1
165	165	Yemen	56.3	30.0	5.18	34.4	4480	23.60	67.5	4.67	1
166	166	Zambia	83.1	37.0	5.89	30.9	3280	14.00	52.0	5.40	1

167 rows × 12 columns

In [11]:

```
df_final=pd.read_csv('C:\Python\Assignments\MLAssignment\country_data_Kmeans_Kmedoids.csv')
matched_list=np.where(df_final['Kmeans Nation status']!=df_final['KMedoids Nation Status'])
del(df_final['Unnamed: 0'])
del(df_final['Unnamed: 0.1'])
df_final.to_csv('C:\Python\Assignments\MLAssignment\country_data_Kmeans_Kmedoids.csv')
```

In [12]:

```
np.where(df_final['Kmeans Nation status']!=df_final['KMedoids Nation Status'])
```

Out[12]:

```
(array([  5,  10,  11,  13,  20,  22,  33,  35,  39,  41,  43,  67,  72,
         81,  84,  85,  86,  90,  98, 102, 104, 108, 121, 124, 125, 130,
        134, 136, 138, 153, 160], dtype=int64),)
```

In [20]:

```
df_final.drop(df_final[df_final.columns[1:10]],axis=1,inplace=True)
df_final.iloc[[  5,  10,  11,  13,  20,  22,  33,  35,  39,  41,  43,  67,  72,
         81,  84,  85,  86,  90,  98, 102, 104, 108, 121, 124, 125, 130,
        134, 136, 138, 153, 160],:]
```

Out[20]:

	country	Kmeans Nation status	KMedoids Nation Status
5	Argentina	developping	developped
10	Bahamas	developping	developped
11	Bahrain	developping	developped
13	Barbados	developping	developped
20	Bosnia and Herzegovina	developping	developped
22	Brazil	developping	developped
33	Chile	developping	developped
35	Colombia	developping	developped
39	Costa Rica	developping	developped
41	Croatia	developping	developped
43	Czech Republic	developping	developped
67	Hungary	developping	developped
72	Iraq	underdevelopped	developping
81	Kiribati	underdevelopped	developping
84	Lao	underdevelopped	developping
85	Latvia	developping	developped
86	Lebanon	developping	developped
90	Lithuania	developping	developped
98	Malta	developped	developping
102	Moldova	developping	developped
104	Montenegro	developping	developped
108	Namibia	underdevelopped	developping
121	Poland	developping	developped
124	Romania	developping	developped
125	Russia	developping	developped
130	Serbia	developping	developped
134	Slovak Republic	developping	developped
136	Solomon Islands	underdevelopped	developping
138	South Korea	developping	developped
153	Turkey	developping	developped
160	Uruguay	developping	developped

In [ ]: