- Import neccessary library
- Read Dataset
- Sanity Check of Data
- Exploratory Data Analysis
- Missing Value Treatment
- Outlier Treatment
- Duplicates & garbage value treatment
- Normalization
- Encoding of data

Import neccessary library

In [1]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
```

Read Dataset

In [2]:
```
df = pd.read_csv('Life Expectancy Data.csv')
```

In [3]:
```
df.head(2)
```

Out[3]:

	Country	Year	Status	Life expectancy	Adult Mortality	infant deaths	Alcohol	percentage expenditure	Hepatit
0	Afghanistan	2015	Developing	65.0	263.0	62	0.01	71.279624	65
1	Afghanistan	2014	Developing	59.9	271.0	64	0.01	73.523582	62

2 rows × 22 columns

In [4]:
```
df.tail(2)
```

Out[4]:

	Country	Year	Status	Life expectancy	Adult Mortality	infant deaths	Alcohol	percentage expenditure	Hepa
2936	Zimbabwe	2001	Developing	45.3	686.0	25	1.72	0.0	
2937	Zimbabwe	2000	Developing	46.0	665.0	24	1.68	0.0	

2 rows × 22 columns

Sanity Check of Data

In [5]:
```
df.shape
```

Out[5]:
```
(2938, 22)
```

In [6]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 2938 entries, 0 to 2937
Data columns (total 22 columns):
 #   Column                           Non-Null Count  Dtype  
---  ------                           --------------  -----  
 0   Country                          2938 non-null   object 
 1   Year                             2938 non-null   int64  
 2   Status                           2938 non-null   object 
 3   Life expectancy                  2928 non-null   float64
 4   Adult Mortality                  2928 non-null   float64
 5   infant deaths                    2938 non-null   int64  
 6   Alcohol                          2744 non-null   float64
 7   percentage expenditure           2938 non-null   float64
 8   Hepatitis B                      2385 non-null   float64
 9   Measles                          2938 non-null   int64  
 10   BMI                             2904 non-null   float64
 11  under-five deaths                2938 non-null   int64  
 12  Polio                            2919 non-null   float64
 13  Total expenditure                2712 non-null   float64
 14  Diphtheria                       2919 non-null   float64
 15   HIV/AIDS                        2938 non-null   float64
 16  GDP                              2490 non-null   float64
 17  Population                       2286 non-null   float64
 18   thinness  1-19 years            2904 non-null   float64
 19   thinness 5-9 years              2904 non-null   float64
 20  Income composition of resources  2771 non-null   float64
 21  Schooling                        2775 non-null   float64
dtypes: float64(16), int64(4), object(2)
memory usage: 505.1+ KB
```

In [7]:
```
# display the missing values count
df.isnull().sum()
```

Out[7]:
```
Country                              0
Year                                 0
Status                               0
Life expectancy                     10
Adult Mortality                     10
infant deaths                        0
Alcohol                            194
percentage expenditure               0
Hepatitis B                        553
Measles                              0
 BMI                                34
under-five deaths                    0
Polio                               19
Total expenditure                  226
Diphtheria                          19
 HIV/AIDS                            0
GDP                                448
Population                         652
 thinness  1-19 years               34
 thinness 5-9 years                 34
Income composition of resources    167
Schooling                          163
dtype: int64
```

In [8]:
```
# display the missing values percentage
round(df.isnull().sum() / df.shape[0]*100,2)
```

Out[8]:
```
Country                             0.00
Year                                0.00
Status                              0.00
Life expectancy                     0.34
Adult Mortality                     0.34
infant deaths                       0.00
Alcohol                             6.60
percentage expenditure              0.00
Hepatitis B                        18.82
Measles                             0.00
 BMI                                1.16
under-five deaths                   0.00
Polio                               0.65
Total expenditure                   7.69
Diphtheria                          0.65
 HIV/AIDS                           0.00
GDP                                15.25
Population                         22.19
 thinness  1-19 years               1.16
 thinness 5-9 years                 1.16
Income composition of resources     5.68
Schooling                           5.55
dtype: float64
```

In [9]:
```
# check the duplicate value
df.duplicated().sum()
```

Out[9]:
```
0
```

In [10]:
```
# identify the garbage value
for i in df.select_dtypes(include='object').columns:
    print(df[i].value_counts())
    print('*'*30)
```

```
Afghanistan              16
Peru                     16
Nicaragua                16
Niger                    16
Nigeria                  16
                         ..
Niue                      1
San Marino                1
Nauru                     1
Saint Kitts and Nevis     1
Dominica                  1
Name: Country, Length: 193, dtype: int64
******************************
Developing    2426
Developed      512
Name: Status, dtype: int64
******************************
```

In [11]:
```python
# describe numerical features
df.describe().T
```

Out[11]:

	count	mean	std	min	25%	50%
Year	2938.0	2.007519e+03	4.613841e+00	2000.00000	2004.000000	2.008000e+03
Life expectancy	2928.0	6.922493e+01	9.523867e+00	36.30000	63.100000	7.210000e+01
Adult Mortality	2928.0	1.647964e+02	1.242921e+02	1.00000	74.000000	1.440000e+02
infant deaths	2938.0	3.030395e+01	1.179265e+02	0.00000	0.000000	3.000000e+00
Alcohol	2744.0	4.602861e+00	4.052413e+00	0.01000	0.877500	3.755000e+00
percentage expenditure	2938.0	7.382513e+02	1.987915e+03	0.00000	4.685343	6.491291e+01
Hepatitis B	2385.0	8.094046e+01	2.507002e+01	1.00000	77.000000	9.200000e+01
Measles	2938.0	2.419592e+03	1.146727e+04	0.00000	0.000000	1.700000e+01
BMI	2904.0	3.832125e+01	2.004403e+01	1.00000	19.300000	4.350000e+01
under-five deaths	2938.0	4.203574e+01	1.604455e+02	0.00000	0.000000	4.000000e+00
Polio	2919.0	8.255019e+01	2.342805e+01	3.00000	78.000000	9.300000e+01
Total expenditure	2712.0	5.938190e+00	2.498320e+00	0.37000	4.260000	5.755000e+00
Diphtheria	2919.0	8.232408e+01	2.371691e+01	2.00000	78.000000	9.300000e+01
HIV/AIDS	2938.0	1.742103e+00	5.077785e+00	0.10000	0.100000	1.000000e-01
GDP	2490.0	7.483158e+03	1.427017e+04	1.68135	463.935626	1.766948e+03
Population	2286.0	1.275338e+07	6.101210e+07	34.00000	195793.250000	1.386542e+06
thinness 1-19 years	2904.0	4.839704e+00	4.420195e+00	0.10000	1.600000	3.300000e+00
thinness 5-9 years	2904.0	4.870317e+00	4.508882e+00	0.10000	1.500000	3.300000e+00
Income composition of resources	2771.0	6.275511e-01	2.109036e-01	0.00000	0.493000	6.770000e-01
Schooling	2775.0	1.199279e+01	3.358920e+00	0.00000	10.100000	1.230000e+01

In [12]:
```python
# describing categorical features
df.describe(include="object").T
```

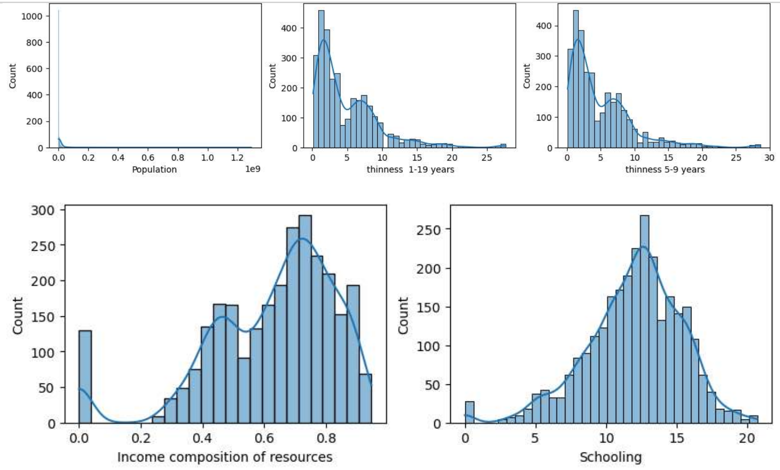
Out[12]:

	count	unique	top	freq
Country	2938	193	Afghanistan	16
Status	2938	2	Developing	2426

Exploratory Data Analysis

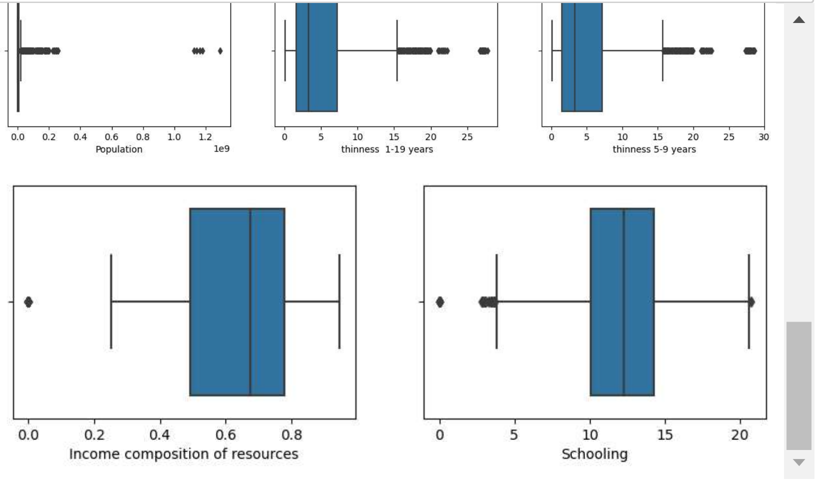
- **check data distribution**

In [13]:
```python
import warnings

warnings.filterwarnings("ignore")

f = 1
plt.figure(figsize=(10,4))

for i in df.select_dtypes(include='number').columns:
    plt.subplot(1,3,f)
    sns.histplot(data=df, x=i, kde=True)
    if f<3:
        f += 1
    else:
        f = 1
        plt.show()
        plt.figure(figsize=(15,3))
```



- **Identify Outlier**

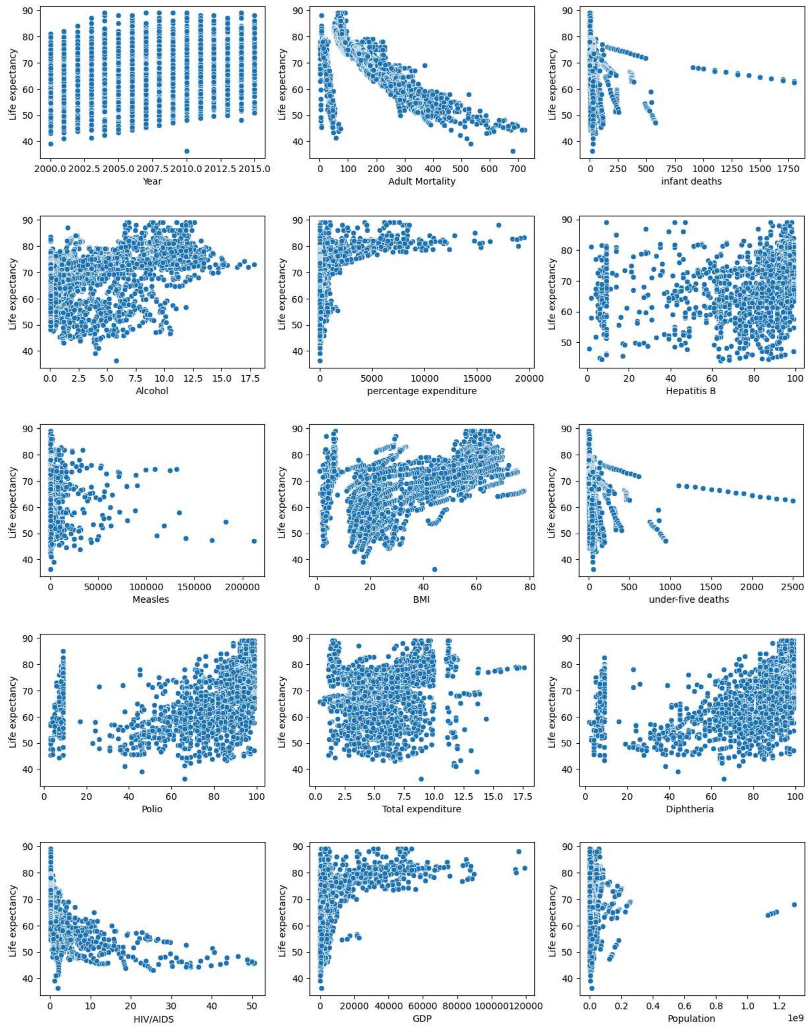
```
In [14]: f = 1
         plt.figure(figsize=(15,3))
         for i in df.select_dtypes(include='number').columns:
             plt.subplot(1,3,f)
             sns.boxplot(data=df, x=i)
             if f<3:
                 f += 1
             else:
                 f = 1
                 plt.show()
                 plt.figure(figsize=(15,3))
```

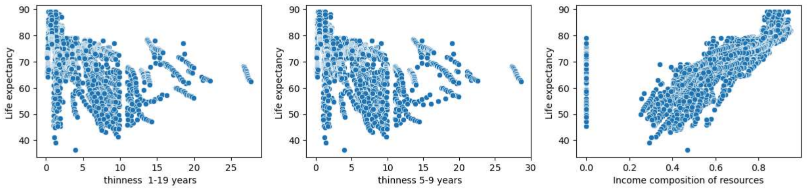


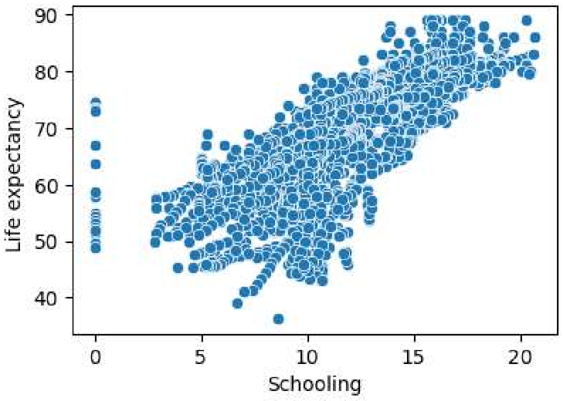
- **Relation between Feature Matrix and Target Vector**

```
In [15]: x = df.select_dtypes(include="number").columns
         x = list(x)
         target = 'Life expectancy '
         x.remove(target) # removing target vector
```

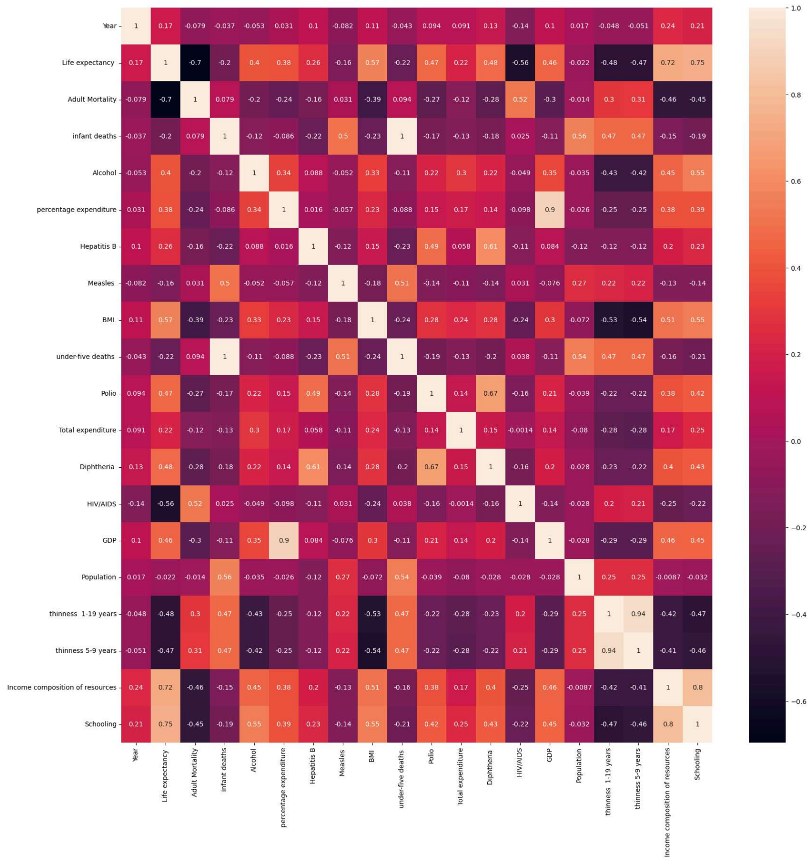
```
In [16]: f = 1
         plt.figure(figsize=(15,3))
         for i in x:
             plt.subplot(1,3,f)
             sns.scatterplot(data=df,x=i,y=target)
             if f<3:
                 f += 1
             else:
                 f = 1
                 plt.show()
                 plt.figure(figsize=(15,3))
```







In [17]:
```
corr_matrix = df.select_dtypes(include='number').corr()
plt.figure(figsize=(20,20))
sns.heatmap(corr_matrix, annot=True)
plt.show()
```



- ## Missing Value Treatment

  - Traditional Method - (Mean, Mode, Median)
  - New Method - KNNImputer

In [18]:
```
for i in [' BMI ','Polio','Income composition of resources']:
    df[i].fillna(df[i].median(), inplace=True)
```

In [19]:
```
df.isna().sum()
```

Out[19]:
```
Country                              0
Year                                 0
Status                               0
Life expectancy                     10
Adult Mortality                     10
infant deaths                        0
Alcohol                            194
percentage expenditure               0
Hepatitis B                        553
Measles                              0
 BMI                                 0
under-five deaths                    0
Polio                                0
Total expenditure                  226
Diphtheria                          19
 HIV/AIDS                            0
GDP                                448
Population                         652
 thinness  1-19 years               34
 thinness 5-9 years                 34
Income composition of resources      0
Schooling                          163
dtype: int64
```

In [20]:
```
# using KNNImputer
from sklearn.impute import KNNImputer

imputer = KNNImputer()
```

In [21]:
```
for i in df.select_dtypes(include='number').columns:
    df[i] = imputer.fit_transform(df[[i]])
```

In [22]:
```
df.isna().sum()
```

Out[22]:
```
Country                            0
Year                               0
Status                             0
Life expectancy                    0
Adult Mortality                    0
infant deaths                      0
Alcohol                            0
percentage expenditure             0
Hepatitis B                        0
Measles                            0
 BMI                               0
under-five deaths                  0
Polio                              0
Total expenditure                  0
Diphtheria                         0
 HIV/AIDS                          0
GDP                                0
Population                         0
 thinness  1-19 years              0
 thinness 5-9 years                0
Income composition of resources    0
Schooling                          0
dtype: int64
```

In [23]:
```
imputer.n_neighbors
```

Out[23]:
```
5
```

- **Outlier Treatment**

In [24]:
```
def wisker(col):
    q1,q3 = np.percentile(col,[25,75])
    iqr = q3 - q1
    hf = q3 + 1.5 * iqr
    lf = q1 - 1.5 * iqr
    return lf,hf
```

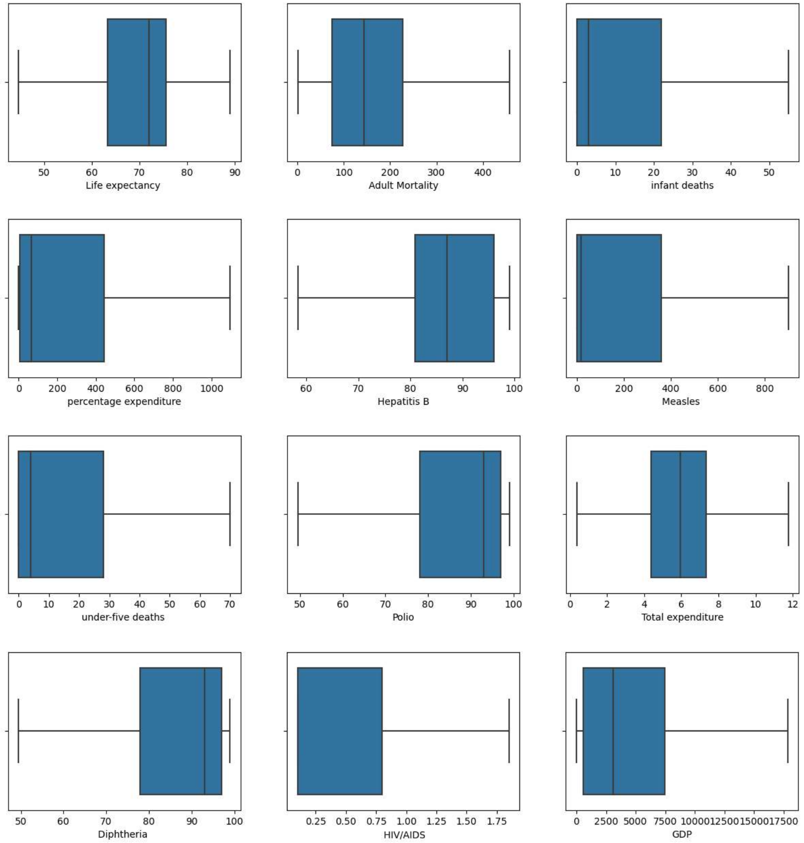
In [25]:
```
wisker(df['GDP'])
```

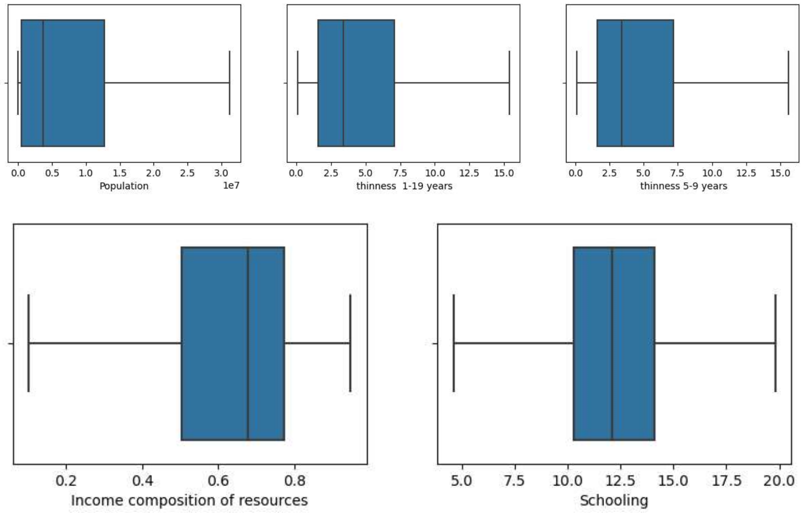
Out[25]:
```
(-9773.52021495771, 17837.165679596183)
```

In [26]:
```
df_outlier_cols = list(df.select_dtypes(include='number').columns)
df_outlier_cols.remove('Year')
df_outlier_cols.remove(' BMI ')
df_outlier_cols.remove('Alcohol')
df_outlier_cols[:2]
```

Out[26]:
```
['Life expectancy ', 'Adult Mortality']
```

In [28]:
```
# removing outlier
for i in df_outlier_cols:
    lw, uw = wisker(df[i])
    df[i] = np.where(df[i]<lw,lw,np.where(df[i]>uw,uw,df[i]))

# boxplot after rmoving outlier

f = 1
plt.figure(figsize=(15,3))
for i in df_outlier_cols:
    plt.subplot(1,3,f)
    sns.boxplot(data=df,x=i)
    if f<3:
        f += 1
    else:
        f = 1
        plt.show()
        plt.figure(figsize=(15,3))
```





- **Duplicates & garbage value treatment**

In [29]:
```
print(df.shape)
df.drop_duplicates(inplace=True)
df.shape
```

```
(2938, 22)
```

Out[29]:
```
(2938, 22)
```

- ## Encoding of data

In [30]:
```
mydata = pd.get_dummies(data=df,columns=['Country','Status'], drop_first=Tr
print(mydata.shape)
mydata.head()
```

```
(2938, 213)
```

Out[30]:

	Year	Life expectancy	Adult Mortality	infant deaths	Alcohol	percentage expenditure	Hepatitis B	Measles	BMI	unc f dea
0	2015.0	65.0	263.0	55.0	0.01	71.279624	65.0	900.625	19.1	7
1	2014.0	59.9	271.0	55.0	0.01	73.523582	62.0	492.000	18.6	7
2	2013.0	59.9	268.0	55.0	0.01	73.219243	64.0	430.000	18.1	7
3	2012.0	59.5	272.0	55.0	0.01	78.184215	67.0	900.625	17.6	7
4	2011.0	59.2	275.0	55.0	0.01	7.097109	68.0	900.625	17.2	7

5 rows × 213 columns

- ## Normalization

In [31]:
```
X = mydata.drop(['Life expectancy '],axis=1)
y = mydata['Life expectancy ']
```

In [32]:
```
# perform Standardization using StandardScaler

from sklearn.preprocessing import StandardScaler

scaler = StandardScaler()

X_scaled = pd.DataFrame(scaler.fit_transform(X), columns=X.columns)
```

In [33]:
```
X_scaled.head(2)
```

Out[33]:

	Year	Adult Mortality	infant deaths	Alcohol	percentage expenditure	Hepatitis B	Measles	BMI	un de
0	1.621762	0.874521	2.165057	-1.172958	-0.546410	-1.534064	1.886225	-0.967349	2.06
1	1.404986	0.943807	2.165057	-1.172958	-0.540647	-1.768413	0.730456	-0.992434	2.06

2 rows × 212 columns

In [34]:
```
# or perform Normalization using MinMaxScaler

from sklearn.preprocessing import MinMaxScaler

scaler = MinMaxScaler(feature_range=(-1,1)) # default range is 0 to 1

X_scaled_normal = pd.DataFrame(scaler.fit_transform(X), columns=X.columns)
```

In [35]:
```
X_scaled_normal.head(4)
```

Out[35]:

	Year	Adult Mortality	infant deaths	Alcohol	percentage expenditure	Hepatitis B	Measles	BMI	under-five deaths
0	1.000000	0.150384	1.0	-1.0	-0.870023	-0.672864	1.000000	-0.580533	1.0
1	0.866667	0.185510	1.0	-1.0	-0.865932	-0.820470	0.092575	-0.592121	1.0
2	0.733333	0.172338	1.0	-1.0	-0.866487	-0.722066	-0.045108	-0.603708	1.0
3	0.600000	0.189901	1.0	-1.0	-0.857433	-0.574460	1.000000	-0.615295	1.0

4 rows × 212 columns

- ## Split dataset into Train and Test

In [36]:
```
from sklearn.model_selection import train_test_split

X_train, X_test, y_train, y_test = train_test_split(X_scaled_normal, y, ran
```

In [37]:
```
X_test.shape, X_train.shape
```

Out[37]:
```
((882, 212), (2056, 212))
```

In [ ]: