Preprocessing and Visualization

- 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 [33]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
```

Read Dataset

In [34]:
```
df = pd.read_csv('Life Expectancy Data.csv')
```

In [35]:
```
df.head()
```

Out[35]:

	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	Afghanistan	2013	Developing	59.9	268.0	66	0.01	73.219243	64
3	Afghanistan	2012	Developing	59.5	272.0	69	0.01	78.184215	67
4	Afghanistan	2011	Developing	59.2	275.0	71	0.01	7.097109	68

5 rows × 22 columns

In [36]:
```
df.tail()
```

Out[36]:

	Country	Year	Status	Life expectancy	Adult Mortality	infant deaths	Alcohol	percentage expenditure	Hepa
2933	Zimbabwe	2004	Developing	44.3	723.0	27	4.36	0.0	
2934	Zimbabwe	2003	Developing	44.5	715.0	26	4.06	0.0	
2935	Zimbabwe	2002	Developing	44.8	73.0	25	4.43	0.0	
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	

5 rows × 22 columns

Sanity Check of Data

In [37]:
```
df.shape
```

Out[37]:
```
(2938, 22)
```

In [38]:
```
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 [39]:
```
# display the missing values count
df.isnull().sum()
```

Out[39]:
```
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 [40]:
```
# display the missing values percentage
round(df.isnull().sum() / df.shape[0]*100,2)
```

Out[40]:
```
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 [102]:
```
print(df.shape)
print(df.shape[0])
print(df.shape[0]*100)
```

```
(2938, 22)
2938
293800
```

In [41]:
```
# check the duplicate value
df.duplicated().sum()
```

Out[41]:
```
0
```

In [42]:
```
# identify the garbage value
for i in df.select_dtypes(include='object').columns:
    print(df[i].value_counts())
    print('*'*30)
```

```
Country
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: count, Length: 193, dtype: int64
******************************
Status
Developing    2426
Developed      512
Name: count, dtype: int64
******************************
```

In [107]:

```
# it filter columns

df.select_dtypes(include='object')
```

Out[107]:

	Country	Status
0	Afghanistan	Developing
1	Afghanistan	Developing
2	Afghanistan	Developing
3	Afghanistan	Developing
4	Afghanistan	Developing
...	...	...
2933	Zimbabwe	Developing
2934	Zimbabwe	Developing
2935	Zimbabwe	Developing
2936	Zimbabwe	Developing
2937	Zimbabwe	Developing

2938 rows × 2 columns

In [108]:

```
# it filter columns

df.select_dtypes(include='object').columns
```

Out[108]:

```
Index(['Country', 'Status'], dtype='object')
```

In [109]:

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

Out[109]:

```
Country
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: count, Length: 193, dtype: int64
```

In [43]:

```
# describe numerical features and T means moving hrizontly
df.describe().T
```

Out[43]:

	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 [115]:
```
df.describe()
```

Out[115]:

	Year	Life expectancy	Adult Mortality	infant deaths	Alcohol	percentage expenditure	Hepa
count	2938.000000	2938.000000	2938.000000	2938.000000	2938.000000	2938.000000	2938.(
mean	2007.518720	69.234802	162.024154	13.635126	4.602861	284.045797	84.6
std	4.613841	9.479612	115.483835	19.108928	3.916288	389.455566	12.8
min	2000.000000	44.600000	1.000000	0.000000	0.010000	0.000000	58.:
25%	2004.000000	63.200000	74.000000	0.000000	1.092500	4.685343	80.9
50%	2008.000000	72.000000	144.000000	3.000000	4.160000	64.912906	87.(
75%	2012.000000	75.600000	227.000000	22.000000	7.390000	441.534144	96.(
max	2015.000000	89.000000	456.500000	55.000000	17.870000	1096.807347	99.(

In [44]:
```
# describing categorical features
df.describe(include="object").T
```

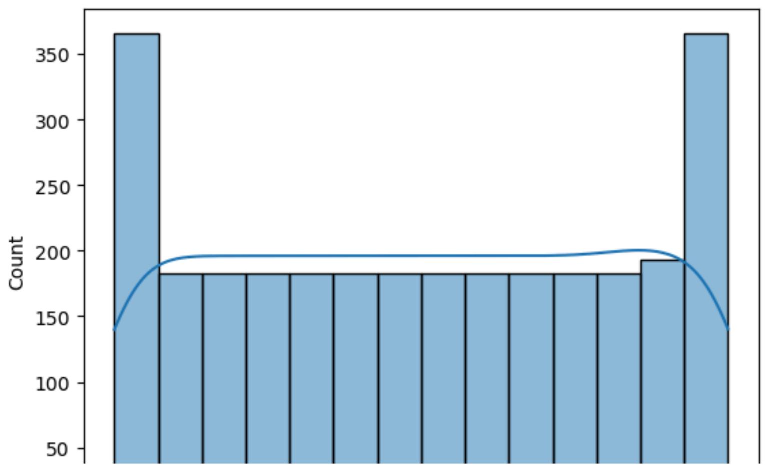
Out[44]:

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

Exploratory Data Analysis

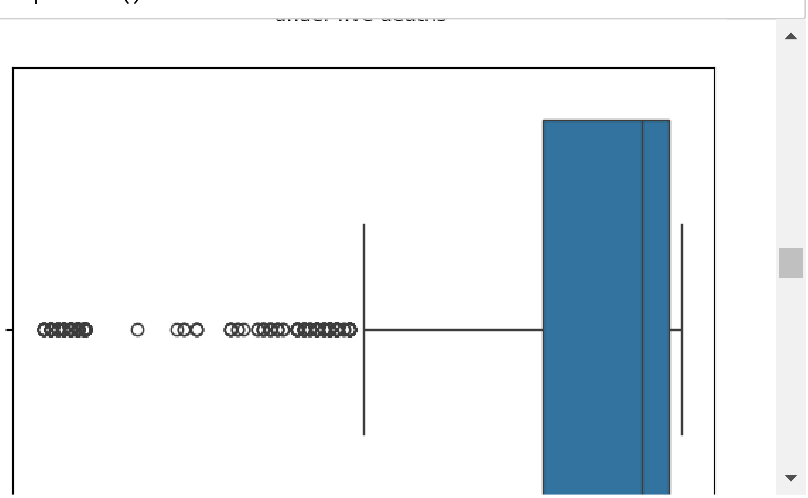
- **check data distribution**

In [45]:
```
import warnings

warnings.filterwarnings("ignore")

for i in df.select_dtypes(include='number').columns:
    sns.histplot(data=df, x=i, kde=True)
    plt.show()
```



- **Identify Outlier**

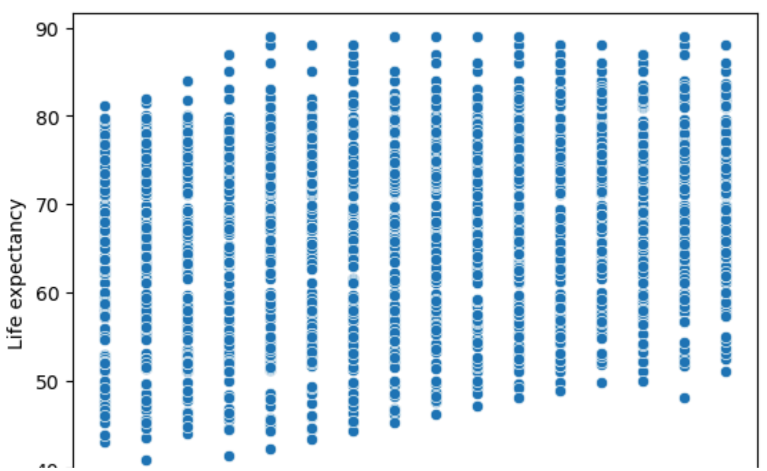
```
In [46]: for i in df.select_dtypes(include='number').columns:
             sns.boxplot(data=df, x=i, vert=False)
             plt.show()
```



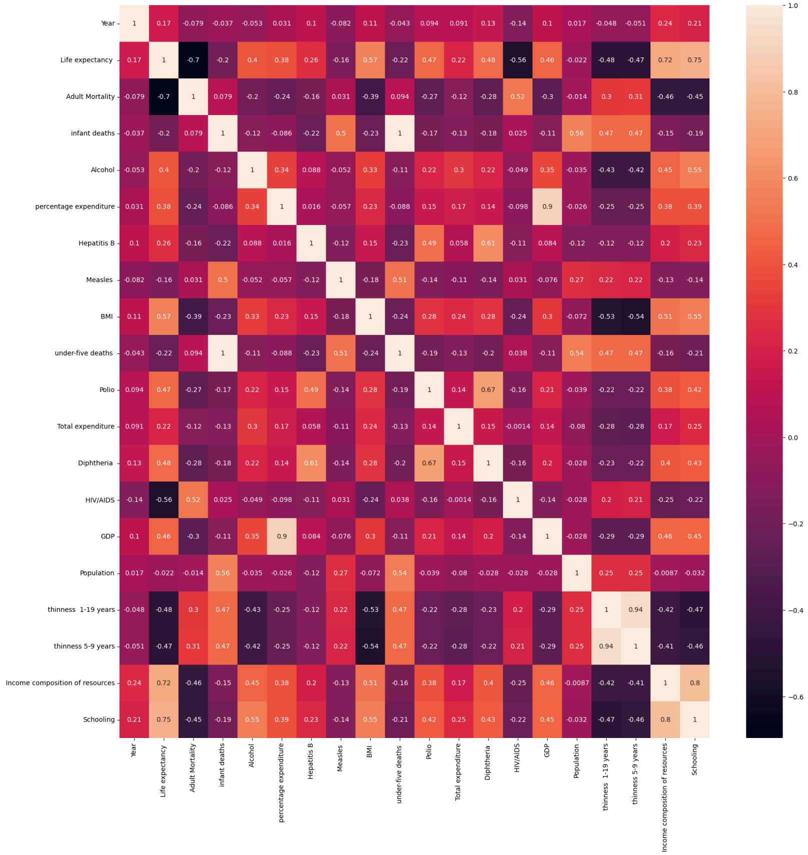
- **Relation between Feature Matrix and Target Vector**

```
In [47]: x = df.select_dtypes(include="number").columns
         x = list(x)
         target = 'Life expectancy '
         x.remove(target) # removing target vector
```

```
In [48]: for i in x:
             sns.scatterplot(data=df,x=i,y=target)
             plt.show()
```



In [49]:
```
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 [50]:
```
for i in [' BMI ','Polio','Income composition of resources']:
    df[i].fillna(df[i].median(), inplace=True)
```

In [51]:
```
df.isna().sum()
```

Out[51]:
```
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 [52]:
```
# using KNNImputer
from sklearn.impute import KNNImputer

imputer = KNNImputer()
```

In [53]:
```
for i in df.select_dtypes(include='number').columns:
    df[i] = imputer.fit_transform(df[[i]])
```

In [54]:
```
df.isna().sum()
```

Out[54]:
```
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 [55]:
```
imputer.n_neighbors
```

Out[55]:
```
5
```

- **Outlier Treatment**

In [56]:
```
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 [57]:
```
wisker(df['GDP'])
```

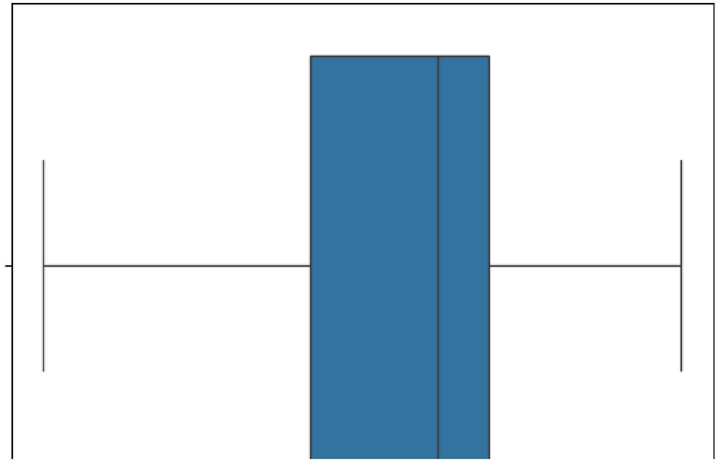
Out[57]:
```
(-9773.52021495771, 17837.165679596183)
```

In [58]:
```
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
```

Out[58]:
```
['Life expectancy ',
 'Adult Mortality',
 'infant deaths',
 'percentage expenditure',
 'Hepatitis B',
 'Measles ',
 'under-five deaths ',
 'Polio',
 'Total expenditure',
 'Diphtheria ',
 ' HIV/AIDS',
 'GDP',
 'Population',
 ' thinness  1-19 years',
 ' thinness 5-9 years',
 'Income composition of resources',
 'Schooling']
```

In [59]:
```
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]))
    # df[df[i] > uw][i] = uw
```

In [60]:
```
for i in df_outlier_cols:
    sns.boxplot(data=df,x=i)
    plt.show()
```



remove duplicate

In [61]:

```
print(df.shape)
df.drop_duplicates(inplace=True)
print(df.shape)
```

```
(2938, 22)
(2938, 22)
```

Encodding

In [62]:

```
pd.get_dummies(data=df,columns=["Country","Status"],drop_first=True)
```

Out[62]:

	Year	Life expectancy	Adult Mortality	infant deaths	Alcohol	percentage expenditure	Hepatitis B	Measles	BMI
0	2015.0	65.0	263.0	55.0	0.01	71.279624	65.000000	900.625	19.1
1	2014.0	59.9	271.0	55.0	0.01	73.523582	62.000000	492.000	18.6
2	2013.0	59.9	268.0	55.0	0.01	73.219243	64.000000	430.000	18.1
3	2012.0	59.5	272.0	55.0	0.01	78.184215	67.000000	900.625	17.6
4	2011.0	59.2	275.0	55.0	0.01	7.097109	68.000000	900.625	17.2
...	...	...	...	...	...	...	...	...	...
2933	2004.0	44.6	456.5	27.0	4.36	0.000000	68.000000	31.000	27.1
2934	2003.0	44.6	456.5	26.0	4.06	0.000000	58.351153	900.625	26.7
2935	2002.0	44.8	73.0	25.0	4.43	0.000000	73.000000	304.000	26.3
2936	2001.0	45.3	456.5	25.0	1.72	0.000000	76.000000	529.000	25.9
2937	2000.0	46.0	456.5	24.0	1.68	0.000000	79.000000	900.625	25.5

2938 rows × 213 columns

In [63]:

```
pd.get_dummies(data=df,columns=["Country","Status"],drop_first=True,dtype=i
```

Out[63]:

	Year	Life expectancy	Adult Mortality	infant deaths	Alcohol	percentage expenditure	Hepatitis B	Measles	BMI
0	2015.0	65.0	263.0	55.0	0.01	71.279624	65.000000	900.625	19.1
1	2014.0	59.9	271.0	55.0	0.01	73.523582	62.000000	492.000	18.6
2	2013.0	59.9	268.0	55.0	0.01	73.219243	64.000000	430.000	18.1
3	2012.0	59.5	272.0	55.0	0.01	78.184215	67.000000	900.625	17.6
4	2011.0	59.2	275.0	55.0	0.01	7.097109	68.000000	900.625	17.2
...	...	...	...	...	...	...	...	...	...
2933	2004.0	44.6	456.5	27.0	4.36	0.000000	68.000000	31.000	27.1
2934	2003.0	44.6	456.5	26.0	4.06	0.000000	58.351153	900.625	26.7
2935	2002.0	44.8	73.0	25.0	4.43	0.000000	73.000000	304.000	26.3
2936	2001.0	45.3	456.5	25.0	1.72	0.000000	76.000000	529.000	25.9
2937	2000.0	46.0	456.5	24.0	1.68	0.000000	79.000000	900.625	25.5

2938 rows × 213 columns

In [64]:

```
mydata = pd.get_dummies(data=df,columns=["Country","Status"],drop_first=Tru
mydata
```

Out[64]:

	Year	Life expectancy	Adult Mortality	infant deaths	Alcohol	percentage expenditure	Hepatitis B	Measles	BMI
0	2015.0	65.0	263.0	55.0	0.01	71.279624	65.000000	900.625	19.1
1	2014.0	59.9	271.0	55.0	0.01	73.523582	62.000000	492.000	18.6
2	2013.0	59.9	268.0	55.0	0.01	73.219243	64.000000	430.000	18.1
3	2012.0	59.5	272.0	55.0	0.01	78.184215	67.000000	900.625	17.6
4	2011.0	59.2	275.0	55.0	0.01	7.097109	68.000000	900.625	17.2
...	...	...	...	...	...	...	...	...	...
2933	2004.0	44.6	456.5	27.0	4.36	0.000000	68.000000	31.000	27.1
2934	2003.0	44.6	456.5	26.0	4.06	0.000000	58.351153	900.625	26.7
2935	2002.0	44.8	73.0	25.0	4.43	0.000000	73.000000	304.000	26.3
2936	2001.0	45.3	456.5	25.0	1.72	0.000000	76.000000	529.000	25.9
2937	2000.0	46.0	456.5	24.0	1.68	0.000000	79.000000	900.625	25.5

2938 rows × 213 columns

In [82]:

```
X = mydata.drop('Life expectancy ',axis=1)

Y = mydata['Life expectancy ']
```

In [83]:

```
X
```

Out[83]:

	Year	Adult Mortality	infant deaths	Alcohol	percentage expenditure	Hepatitis B	Measles	BMI	under-five deaths	Poli
0	2015.0	263.0	55.0	0.01	71.279624	65.000000	900.625	19.1	70.0	49.
1	2014.0	271.0	55.0	0.01	73.523582	62.000000	492.000	18.6	70.0	58.
2	2013.0	268.0	55.0	0.01	73.219243	64.000000	430.000	18.1	70.0	62.
3	2012.0	272.0	55.0	0.01	78.184215	67.000000	900.625	17.6	70.0	67.
4	2011.0	275.0	55.0	0.01	7.097109	68.000000	900.625	17.2	70.0	68.
...	...	...	...	...	...	...	...	...	...	.
2933	2004.0	456.5	27.0	4.36	0.000000	68.000000	31.000	27.1	42.0	67.
2934	2003.0	456.5	26.0	4.06	0.000000	58.351153	900.625	26.7	41.0	49.
2935	2002.0	73.0	25.0	4.43	0.000000	73.000000	304.000	26.3	40.0	73.
2936	2001.0	456.5	25.0	1.72	0.000000	76.000000	529.000	25.9	39.0	76.
2937	2000.0	456.5	24.0	1.68	0.000000	79.000000	900.625	25.5	39.0	78.

2938 rows × 212 columns

In [84]:

```
Y
```

Out[84]:

```
0       65.0
1       59.9
2       59.9
3       59.5
4       59.2
        ...
2933    44.6
2934    44.6
2935    44.8
2936    45.3
2937    46.0
Name: Life expectancy , Length: 2938, dtype: float64
```

Normalization

- Scalling using Standerdization

In [110]:

```
from sklearn.preprocessing import StandardScaler
scaler = StandardScaler()
scaler
```

Out[110]:

```
StandardScaler()
```

In a Jupyter environment, please rerun this cell to show the HTML representation or trust the notebook.
On GitHub, the HTML representation is unable to render, please try loading this page with nbviewer.org.

In [86]:

```
scaler.fit_transform(X)
```

Out[86]:

```
array([[ 1.6217623 ,  0.87452096,  2.16505686, ..., -0.07399798,
        -0.07399798,  0.45939851],
       [ 1.40498625,  0.94380652,  2.16505686, ..., -0.07399798,
        -0.07399798,  0.45939851],
       [ 1.1882102 ,  0.91782444,  2.16505686, ..., -0.07399798,
        -0.07399798,  0.45939851],
       ...,
       [-1.19632639, -0.77101101,  0.59484283, ..., -0.07399798,
        13.51388175,  0.45939851],
       [-1.41310244,  2.55036537,  0.59484283, ..., -0.07399798,
        13.51388175,  0.45939851],
       [-1.62987849,  2.55036537,  0.54250236, ..., -0.07399798,
        13.51388175,  0.45939851]])
```

In [87]:

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

Out[87]:

	Year	Adult Mortality	infant deaths	Alcohol	percentage expenditure	Hepatitis B	Measles	BMI
0	1.621762	0.874521	2.165057	-1.172958	-0.546410	-1.534064	1.886225	-0.967349
1	1.404986	0.943807	2.165057	-1.172958	-0.540647	-1.768413	0.730456	-0.992434
2	1.188210	0.917824	2.165057	-1.172958	-0.541429	-1.612181	0.555093	-1.017519
3	0.971434	0.952467	2.165057	-1.172958	-0.528678	-1.377832	1.886225	-1.042605
4	0.754658	0.978449	2.165057	-1.172958	-0.711239	-1.299715	1.886225	-1.062673
...	...	...	...	...	...	...	...	...
2933	-0.762774	2.550365	0.699524	-0.062024	-0.729465	-1.299715	-0.573453	-0.565984
2934	-0.979550	2.550365	0.647183	-0.138640	-0.729465	-2.053448	1.886225	-0.586052
2935	-1.196326	-0.771011	0.594843	-0.044146	-0.729465	-0.909134	0.198710	-0.606120
2936	-1.413102	2.550365	0.594843	-0.736246	-0.729465	-0.674785	0.835108	-0.626188
2937	-1.629878	2.550365	0.542502	-0.746462	-0.729465	-0.440436	1.886225	-0.646257

2938 rows × 212 columns

- Scaling using Normalization

In [89]:

```
# in normalization convert data by default  into 0 to 1

from  sklearn.preprocessing import MinMaxScaler
scaler = MinMaxScaler(feature_range=(-1,1))   # here we have defind a accer
X = pd.DataFrame(scaler.fit_transform(X),columns=X.columns)
X
```

Out[89]:

	Year	Adult Mortality	infant deaths	Alcohol	percentage expenditure	Hepatitis B	Measles	BMI
0	1.000000	0.150384	1.000000	-1.000000	-0.870023	-0.672864	1.000000	-0.580533
1	0.866667	0.185510	1.000000	-1.000000	-0.865932	-0.820470	0.092575	-0.592121
2	0.733333	0.172338	1.000000	-1.000000	-0.866487	-0.722066	-0.045108	-0.603708
3	0.600000	0.189901	1.000000	-1.000000	-0.857433	-0.574460	1.000000	-0.615295
4	0.466667	0.203074	1.000000	-1.000000	-0.987059	-0.525259	1.000000	-0.624565
...	...	...	...	...	...	...	...	...
2933	-0.466667	1.000000	-0.018182	-0.512878	-1.000000	-0.525259	-0.931159	-0.395133
2934	-0.600000	1.000000	-0.054545	-0.546473	-1.000000	-1.000000	1.000000	-0.404403
2935	-0.733333	-0.683864	-0.090909	-0.505039	-1.000000	-0.279249	-0.324913	-0.413673
2936	-0.866667	1.000000	-0.090909	-0.808511	-1.000000	-0.131643	0.174740	-0.422943
2937	-1.000000	1.000000	-0.127273	-0.812990	-1.000000	0.015962	1.000000	-0.432213

2938 rows × 212 columns

In [90]:

```

```

In [98]:

```
from sklearn.model_selection import train_test_split
x_train,x_test,y_train,y_test = train_test_split(X,Y,test_size=0.3,random_s


# if use random_state then will not change accuracy of ml model
```

In [97]:

```
len(x_train),len(x_test),len(y_train),len(y_test)
```

Out[97]:

```
(2056, 882, 2056, 882)
```

In [ ]:

```

```

In [ ]:

```

```

In [ ]:

```

```