Importing the Libraries

In [226…]
```
import pandas as pd
from matplotlib import pyplot as plt
import numpy as np
import math
import seaborn as sns

import statsmodels.api as sm
import statsmodels.graphics.tsaplots as tsap
from statsmodels.compat import lzip
from statsmodels.stats.diagnostic import het_white

import pylab
import scipy.stats as stats

import sklearn
import sklearn.impute
from sklearn import impute
from sklearn.impute import SimpleImputer
from sklearn import linear_model
from sklearn import preprocessing

from sklearn.metrics import mean_squared_error
from sklearn.metrics import explained_variance_score
from sklearn import ensemble
from sklearn.model_selection import cross_val_score

from linearmodels import PanelOLS
from linearmodels import RandomEffects
```

Import the Dataset

In [227…]
```
lifedata = pd.read_csv(r"C:\Users\adspa\OneDrive\Desktop\Life Expectancy Data_HV22 - L
```

In [228…]
```
lifedata
```

Out[228]:

	Country	Year	Status	Life expectancy	Adult Mortality	infant deaths	Alcohol	percentage expenditure	Hepatitis B
0	Afghanistan	2015	Developing	65.0	263.0	62	0.01	71.279624	65.0
1	Afghanistan	2014	Developing	59.9	271.0	64	0.01	73.523582	62.0
2	Afghanistan	2013	Developing	59.9	268.0	66	0.01	73.219243	64.0
3	Afghanistan	2012	Developing	59.5	272.0	69	0.01	78.184215	67.0
4	Afghanistan	2011	Developing	59.2	275.0	71	0.01	7.097109	68.0
...	...	...	...	...	...	...	...	...	...
2933	Zimbabwe	2004	Developing	44.3	723.0	27	4.36	0.000000	68.0
2934	Zimbabwe	2003	Developing	44.5	715.0	26	4.06	0.000000	7.0
2935	Zimbabwe	2002	Developing	44.8	73.0	25	4.43	0.000000	73.0
2936	Zimbabwe	2001	Developing	45.3	686.0	25	1.72	0.000000	76.0
2937	Zimbabwe	2000	Developing	46.0	665.0	24	1.68	0.000000	79.0

2938 rows × 22 columns

About Dataset:

This dataset columns indicate:

Country- Country

Year- Year

Status- Developed or Developing status

Life Expectancy- Age(years)

Adult Mortality- Adult Mortality Rates of both sexes(probability of dying between 15&60 years per 1000 population)

Infant Deaths- Number of Infant Deaths per 1000 population

Alcohol- Alcohol, recorded per capita (15+) consumption (in litres of pure alcohol)

Percent Expenditure- Expenditure on health as a percentage of Gross Domestic Product per capita(%)

Hep B- Hepatitis B (HepB) immunization coverage among 1-year-olds(%)

Measles- number of reported measles cases per 1000 population

BMI- Average Body Mass Index of entire population

U-5 Deaths- Number of under-five deaths per 1000 population

Polio- Polio(Pol3) immunization coverage among 1-year-olds(%)

Total Expenditure- General government expenditure on health as a percentage of total government expenditure(%)

Diphtheria- Diphtheria tetanus toxoid and pertussis (DTP3) immunization coverage among 1-year-olds(%)

HIV/AIDS- Deaths per 1000 live births HIV/AIDS(0-4 years)

GDP- Gross Domestic Product per capita(in USD)

Population- Population Thinness (1-19)- Prevalence of thinness among children and adolescents for Age 10 to 19(%)

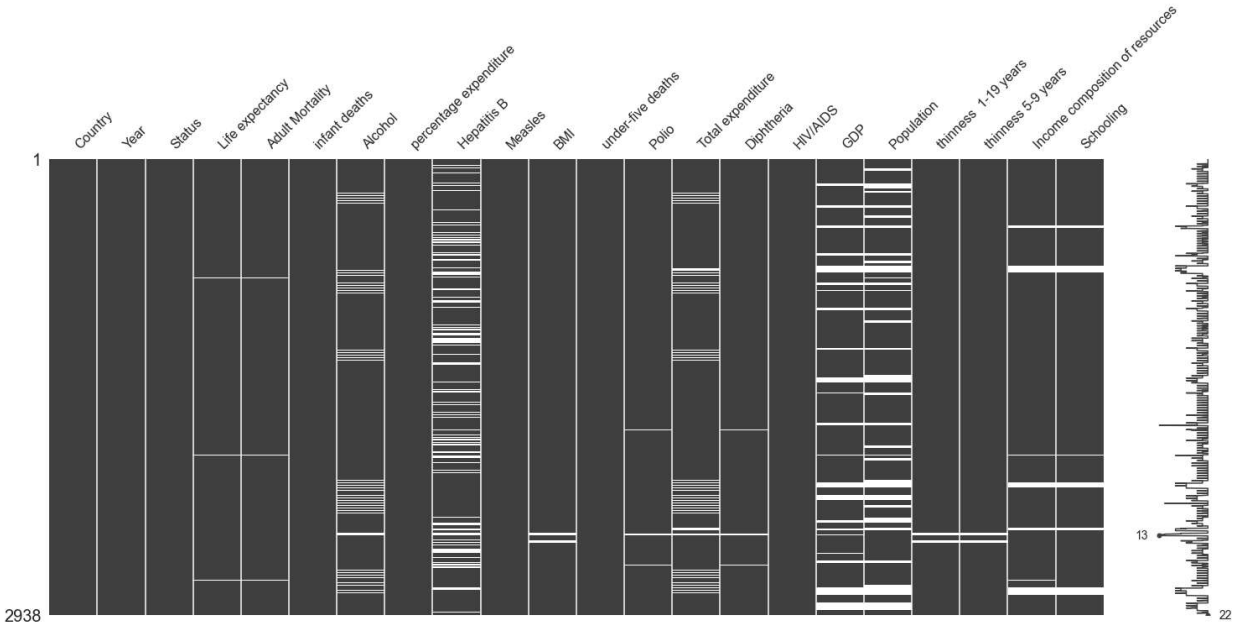
Thinness- (5-9)- Prevalence of thinness among children for Age 5 to 9(%)

Income Composition- Human Development Index in terms of income composition of resources(0-1)

Schooling- Number of years of Schooling

In [229…
```
import missingno as msno
print(msno.matrix(lifedata))
```

```
AxesSubplot(0.125,0.125;0.698618x0.755)
```



Data Summarization

In [230…
```
lifedata.columns
```

Out[230]:
```
Index(['Country', 'Year', 'Status', 'Life expectancy', 'Adult Mortality',
       'infant deaths', 'Alcohol', 'percentage expenditure', 'Hepatitis B',
       'Measles', 'BMI', 'under-five deaths', 'Polio', 'Total expenditure',
       'Diphtheria', 'HIV/AIDS', 'GDP', 'Population', 'thinness  1-19 years',
       'thinness 5-9 years', 'Income composition of resources', 'Schooling'],
      dtype='object')
```

In [231…
```
lifedata.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 [232…
```
lifedata.describe()
```

Out[232]:

	Year	Life expectancy	Adult Mortality	infant deaths	Alcohol	percentage expenditure	Hepatitis B
count	2938.000000	2928.000000	2928.000000	2938.000000	2744.000000	2938.000000	2385.000000
mean	2007.518720	69.224932	164.796448	30.303948	4.602861	738.251295	80.940461
std	4.613841	9.523867	124.292079	117.926501	4.052413	1987.914858	25.070016
min	2000.000000	36.300000	1.000000	0.000000	0.010000	0.000000	1.000000
25%	2004.000000	63.100000	74.000000	0.000000	0.877500	4.685343	77.000000
50%	2008.000000	72.100000	144.000000	3.000000	3.755000	64.912906	92.000000
75%	2012.000000	75.700000	228.000000	22.000000	7.702500	441.534144	97.000000
max	2015.000000	89.000000	723.000000	1800.000000	17.870000	19479.911610	99.000000

In [233…
```
lifedata.shape
```

Out[233]:
```
(2938, 22)
```

In [234…
```
lifedata.isnull().sum()
```

Out[234]:
```
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
```

Data Pre-Processing

Missing Value Imputation

In [235…
```
from sklearn.impute import SimpleImputer
imputer = SimpleImputer(missing_values=np.nan, strategy='mean')
imputer.fit(x[:,3:])
x[:,3:] = imputer.transform(x[:,3:])
```

In [236…
```
# Filling the null values with the 'mean' of the respective columns
nndata = lifedata.fillna(lifedata.mean())
nndata.isnull().sum()
```

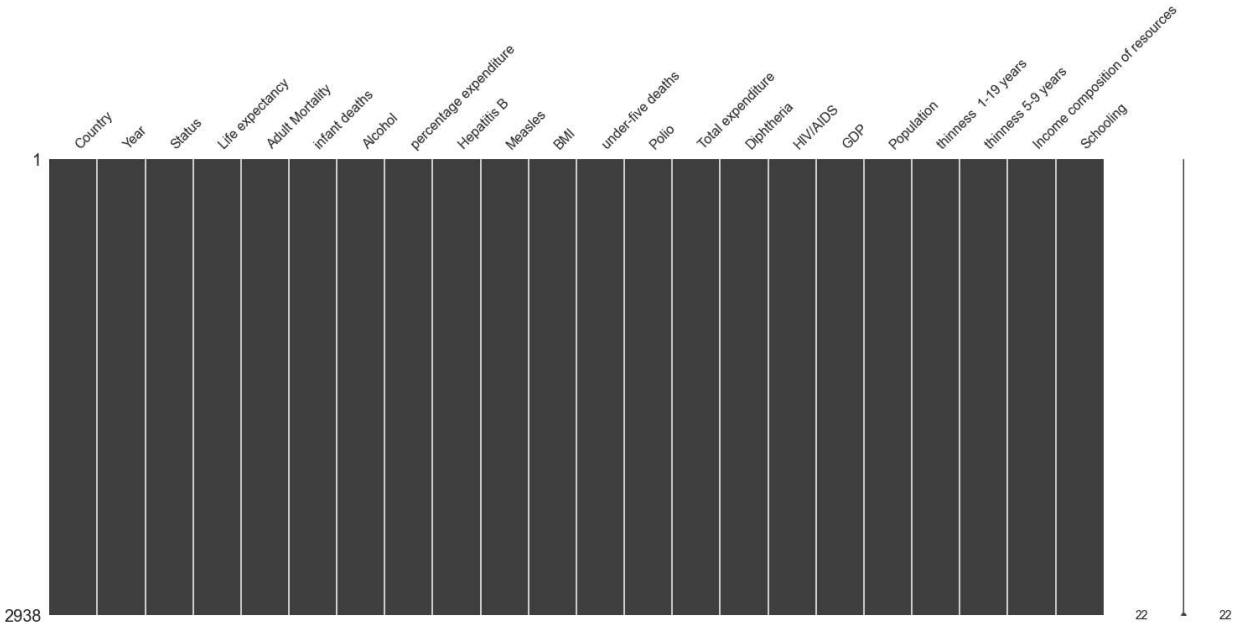
```
C:\Users\adspa\AppData\Local\Temp\ipykernel_34392\1317151566.py:2: FutureWarning: Dro
pping of nuisance columns in DataFrame reductions (with 'numeric_only=None') is depre
cated; in a future version this will raise TypeError.  Select only valid columns befo
re calling the reduction.
  nndata = lifedata.fillna(lifedata.mean())
```

Out[236]:
```
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
```

Plot Showing that all the missing values have been filled successfully

In [237…
```
print(msno.matrix(nndata))
```

```
AxesSubplot(0.125,0.125;0.698618x0.755)
```

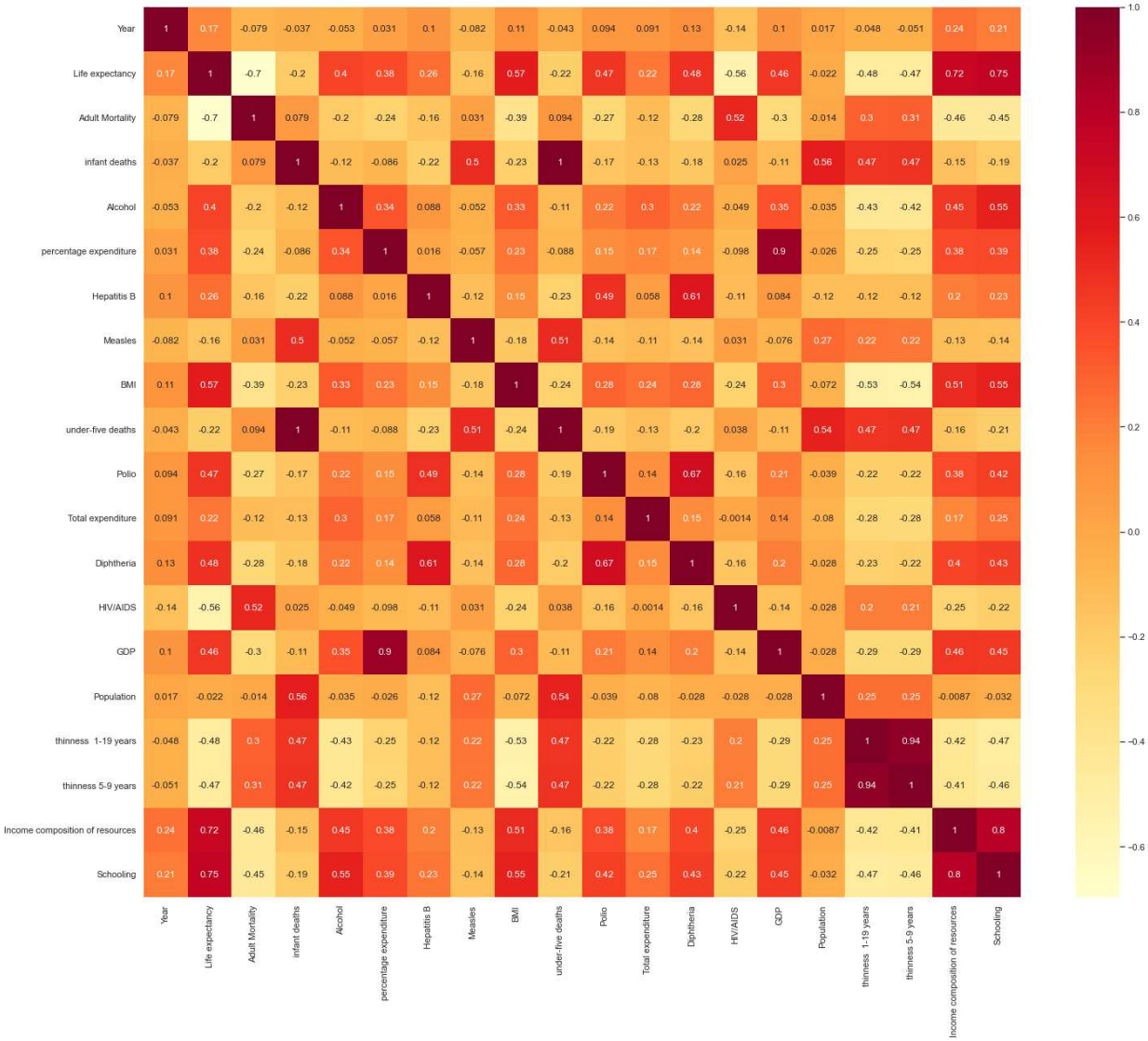


Removing Multicollinearity

Checking with Correlation Matrix and Heatmap

In [238…
```
# visualizing correlation between columns
import matplotlib.pyplot as plt
```

```
import seaborn as sns

# Get correlation of all the features of the dataset
corr_matrix = lifedata.corr()
top_corr_features = corr_matrix.index

# Plotting the heatmap
plt.figure(figsize=(24,20))
g = sns.heatmap(data=lifedata[top_corr_features].corr(), annot=True, cmap='YlOrRd')
```



Checking Multicollinearity with Variance Inflation Factor (VIFs)

In [239…

```
from statsmodels.stats.outliers_influence import variance_inflation_factor

X = nndata[['Adult Mortality',
       'infant deaths', 'Alcohol', 'percentage expenditure', 'Hepatitis B',
       'Measles', 'BMI', 'under-five deaths', 'Polio', 'Total expenditure',
       'Diphtheria', 'HIV/AIDS', 'GDP', 'Population', 'thinness  1-19 years',
       'thinness 5-9 years', 'Income composition of resources', 'Schooling']]

vif_data = pd.DataFrame()
```

```
vif_data["Features"]=X.columns

vif_data["VIF"]=[variance_inflation_factor(X.values, i) for i in range(len(X.columns))]
print(vif_data)
```

```
                          Features         VIF
0                  Adult Mortality    3.851726
1                    infant deaths  182.925170
2                          Alcohol    3.716947
3          percentage expenditure    5.682942
4                      Hepatitis B   17.983301
5                          Measles    1.427810
6                              BMI    7.786855
7                under-five deaths  182.229297
8                            Polio   25.610090
9                Total expenditure    7.586151
10                      Diphtheria   28.930883
11                        HIV/AIDS    1.577186
12                             GDP    6.849052
13                      Population    1.567142
14           thinness  1-19 years   19.183925
15             thinness 5-9 years   19.190739
16  Income composition of resources   30.747148
17                       Schooling   43.869733
```

In [240…

```
cdata = nndata.drop(['infant deaths', 'Hepatitis B', 'Diphtheria', 'Polio', 'thinness
                     'Income composition of resources', 'Schooling'], axis = 1)

print(f'After preaparing the data our cleaned data is :\n')
cdata
```

```
After preaparing the data our cleaned data is :
```

Out[240]:

	Country	Year	Status	Life expectancy	Adult Mortality	Alcohol	percentage expenditure	Measles	BMI	un de
0	Afghanistan	2015	Developing	65.0	263.0	0.01	71.279624	1154	19.1	
1	Afghanistan	2014	Developing	59.9	271.0	0.01	73.523582	492	18.6	
2	Afghanistan	2013	Developing	59.9	268.0	0.01	73.219243	430	18.1	
3	Afghanistan	2012	Developing	59.5	272.0	0.01	78.184215	2787	17.6	
4	Afghanistan	2011	Developing	59.2	275.0	0.01	7.097109	3013	17.2	
...	...	...	...	...	...	...	...	...	...	
2933	Zimbabwe	2004	Developing	44.3	723.0	4.36	0.000000	31	27.1	
2934	Zimbabwe	2003	Developing	44.5	715.0	4.06	0.000000	998	26.7	
2935	Zimbabwe	2002	Developing	44.8	73.0	4.43	0.000000	304	26.3	
2936	Zimbabwe	2001	Developing	45.3	686.0	1.72	0.000000	529	25.9	
2937	Zimbabwe	2000	Developing	46.0	665.0	1.68	0.000000	1483	25.5	

2938 rows × 15 columns

In [241…
```
cdata.columns
```

Out[241]:
```
Index(['Country', 'Year', 'Status', 'Life expectancy', 'Adult Mortality',
       'Alcohol', 'percentage expenditure', 'Measles', 'BMI',
       'under-five deaths', 'Total expenditure', 'HIV/AIDS', 'GDP',
       'Population', 'thinness  1-19 years'],
      dtype='object')
```

In [242…
```
Y = cdata[['Adult Mortality',
       'Alcohol', 'percentage expenditure', 'Measles', 'BMI',
       'under-five deaths', 'Total expenditure', 'HIV/AIDS',
       'GDP', 'Population', 'thinness  1-19 years']]

vif_data = pd.DataFrame()
vif_data["Features"]=Y.columns

vif_data["VIF"]=[variance_inflation_factor(Y.values, i) for i in range(len(Y.columns))
print(vif_data)
```

```
                  Features       VIF
0          Adult Mortality  3.573356
1                  Alcohol  3.147744
2   percentage expenditure  5.549326
3                  Measles  1.409439
4                      BMI  4.849081
5        under-five deaths  2.233452
6        Total expenditure  6.232489
7                  HIV/AIDS  1.546154
8                      GDP  6.463295
9               Population  1.500639
10    thinness  1-19 years  2.891294
```

In [243…
```
#Assigning x as the 'independent variables' & y  as the 'target variable'
x = lifedata.iloc[:, :-1].values
y = lifedata.iloc[:, -1].values
```

In [244…
```
print(x)
```

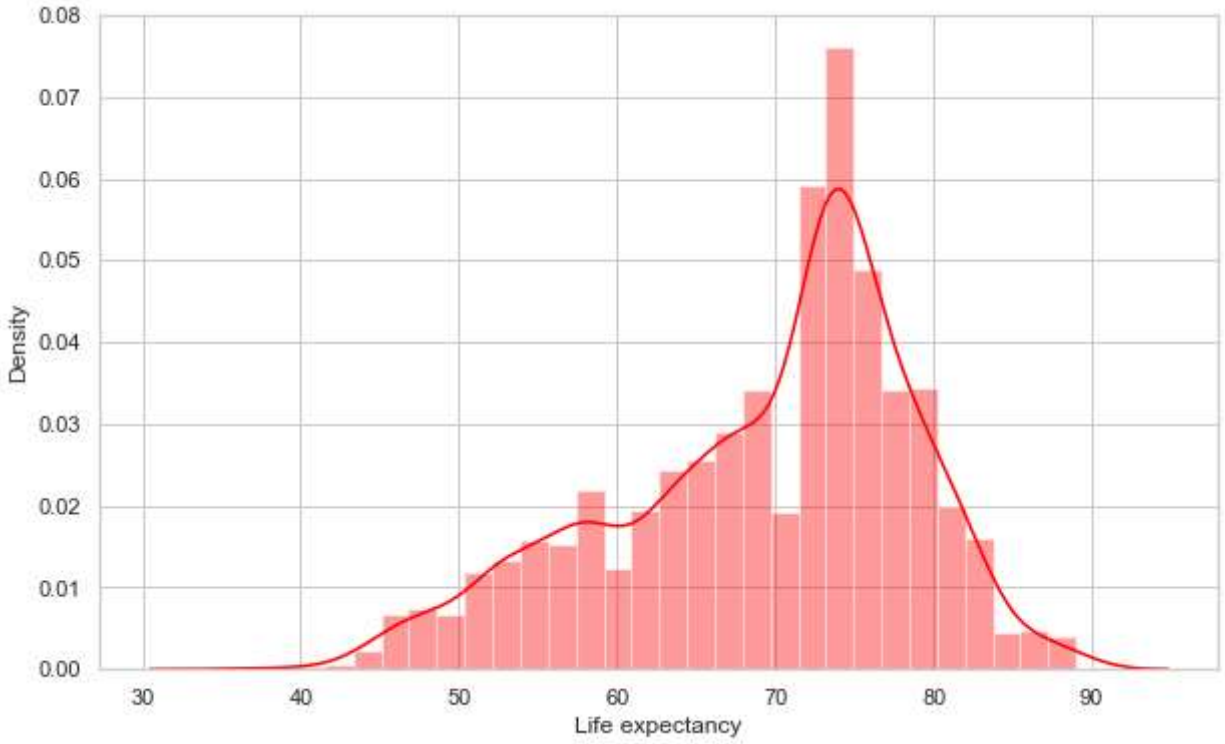
```
[['Afghanistan' 2015 'Developing' ... 17.2 17.3 0.479]
 ['Afghanistan' 2014 'Developing' ... 17.5 17.5 0.476]
 ['Afghanistan' 2013 'Developing' ... 17.7 17.7 0.47]
 ...
 ['Zimbabwe' 2002 'Developing' ... 1.2 1.3 0.427]
 ['Zimbabwe' 2001 'Developing' ... 1.6 1.7 0.427]
 ['Zimbabwe' 2000 'Developing' ... 11.0 11.2 0.434]]
```

In [245…
```
print(y)
```

```
[10.1 10.   9.9 ... 10.   9.8  9.8]
```

In [246…
```
# Before proceeding with feature elimination lets have a look at the distribution of c

sns.set(style='whitegrid')
f,ax = plt.subplots(1,1,figsize = (10,6))
ax = sns.distplot(lifedata['Life expectancy'], kde = True, color = 'red')
```

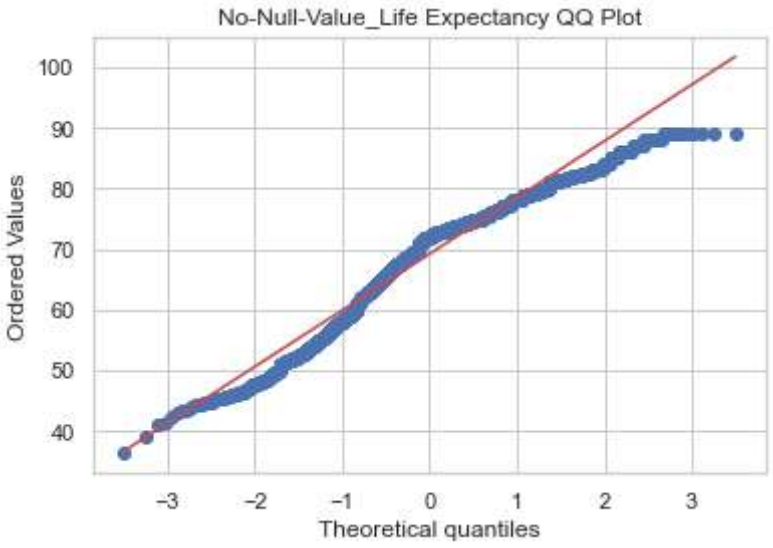
```
C:\Users\adspa\anaconda3\lib\site-packages\seaborn\distributions.py:2619: FutureWarni
ng: `distplot` is a deprecated function and will be removed in a future version. Plea
se adapt your code to use either `displot` (a figure-level function with similar flex
ibility) or `histplot` (an axes-level function for histograms).
  warnings.warn(msg, FutureWarning)
```



Exploratory Data Analysis

In [247…
```
stats.probplot(nndata['Life expectancy'], dist="norm", plot=plt)
plt.title('No-Null-Value_Life Expectancy QQ Plot')
print(stats.shapiro(nndata['Life expectancy']))
```

```
ShapiroResult(statistic=0.9563312530517578, pvalue=8.011480975249593e-29)
```



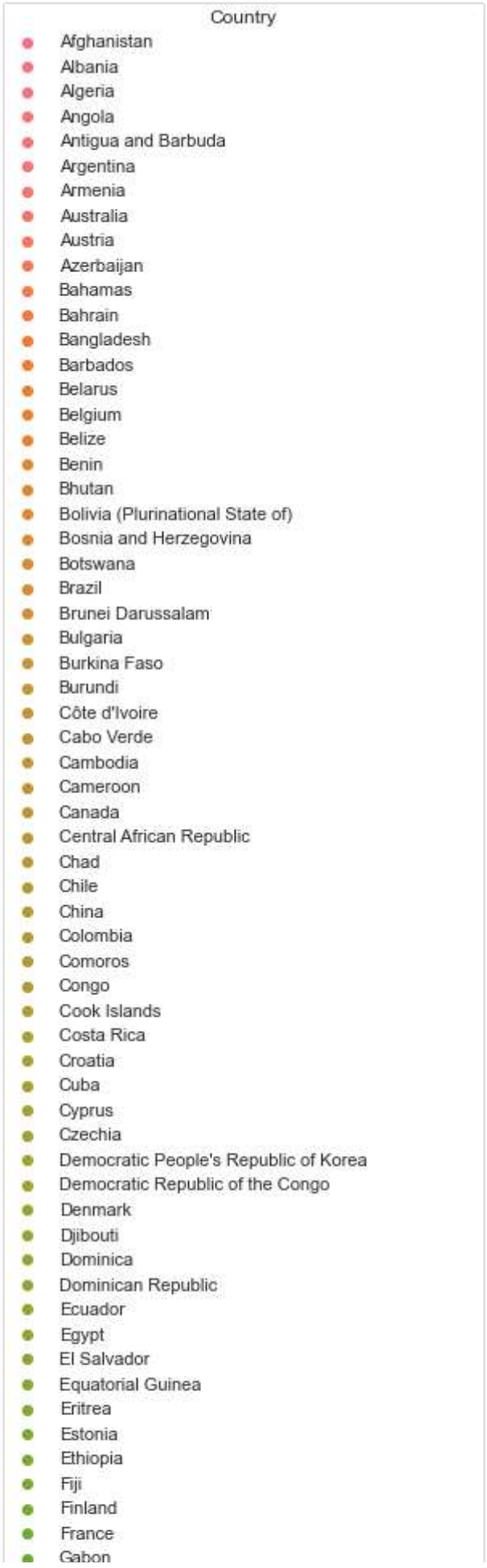
In [248…
```
cdata.columns
```

Out[248]:
```
Index(['Country', 'Year', 'Status', 'Life expectancy', 'Adult Mortality',
       'Alcohol', 'percentage expenditure', 'Measles', 'BMI',
       'under-five deaths', 'Total expenditure', 'HIV/AIDS', 'GDP',
       'Population', 'thinness  1-19 years'],
      dtype='object')
```

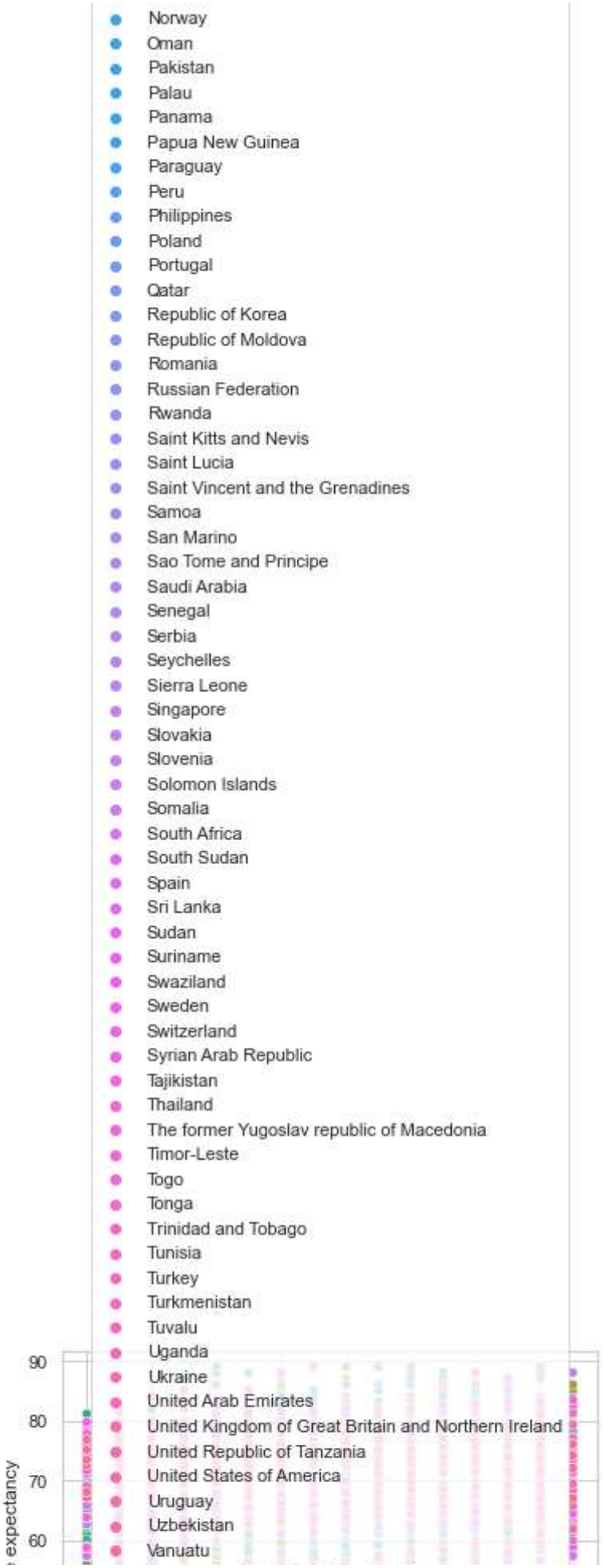
In [249…
```
from scipy import stats

for i in cdata.drop(['Life expectancy', 'Country', 'Status'], axis=1):

    print(f'\nTtest_results of independent column *{i}* with the target variable "Life
```

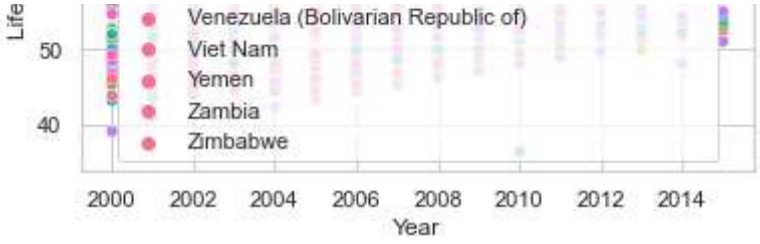
```
Ttest_results of independent column *Year* with the target variable "Life expectancy"
:
 Ttest_indResult(statistic=9941.512304565262, pvalue=0.0)

Ttest_results of independent column *Adult Mortality* with the target variable "Life
expectancy" :
 Ttest_indResult(statistic=41.62749791080229, pvalue=0.0)

Ttest_results of independent column *Alcohol* with the target variable "Life expectan
cy" :
 Ttest_indResult(statistic=-340.64534335039104, pvalue=0.0)

Ttest_results of independent column *percentage expenditure* with the target variable
"Life expectancy" :
 Ttest_indResult(statistic=18.241745019827178, pvalue=2.322142448020643e-72)

Ttest_results of independent column *Measles* with the target variable "Life expectan
cy" :
 Ttest_indResult(statistic=11.109674931831677, pvalue=2.155797900296202e-28)

Ttest_results of independent column *BMI* with the target variable "Life expectancy"
:
 Ttest_indResult(statistic=-75.86570720187584, pvalue=0.0)

Ttest_results of independent column *under-five deaths* with the target variable "Lif
e expectancy" :
 Ttest_indResult(statistic=-9.169240634590636, pvalue=6.462905630057323e-20)

Ttest_results of independent column *Total expenditure* with the target variable "Lif
e expectancy" :
 Ttest_indResult(statistic=-349.82370490968066, pvalue=0.0)

Ttest_results of independent column *HIV/AIDS* with the target variable "Life expecta
ncy" :
 Ttest_indResult(statistic=-339.3556926402777, pvalue=0.0)

Ttest_results of independent column *GDP* with the target variable "Life expectancy"
:
 Ttest_indResult(statistic=30.590379763834957, pvalue=7.883589079896424e-191)

Ttest_results of independent column *Population* with the target variable "Life expec
tancy" :
 Ttest_indResult(statistic=12.845221420127023, pvalue=2.893075763217165e-37)

Ttest_results of independent column *thinness  1-19 years* with the target variable
"Life expectancy" :
 Ttest_indResult(statistic=-333.1920604204805, pvalue=0.0)
```

In [250…
```
sns.scatterplot(x=cdata['Year'],  y=cdata['Life expectancy'], hue=cdata['Country'])


plt.show()
```


Country
Afghanistan
Albania
Algeria
Angola
Antigua and Barbuda
Argentina
Armenia
Australia
Austria
Azerbaijan
Bahamas
Bahrain
Bangladesh
Barbados
Belarus
Belgium
Belize
Benin
Bhutan
Bolivia (Plurinational State of)
Bosnia and Herzegovina
Botswana
Brazil
Brunei Darussalam
Bulgaria
Burkina Faso
Burundi
Côte d'Ivoire
Cabo Verde
Cambodia
Cameroon
Canada
Central African Republic
Chad
Chile
China
Colombia
Comoros
Congo
Cook Islands
Costa Rica
Croatia
Cuba
Cyprus
Czechia
Democratic People's Republic of Korea
Democratic Republic of the Congo
Denmark
Djibouti
Dominica
Dominican Republic
Ecuador
Egypt
El Salvador
Equatorial Guinea
Eritrea
Estonia
Ethiopia
Fiji
Finland
France
Gabon

- Gambia
- Georgia
- Germany
- Ghana
- Greece
- Grenada
- Guatemala
- Guinea
- Guinea-Bissau
- Guyana
- Haiti
- Honduras
- Hungary
- Iceland
- India
- Indonesia
- Iran (Islamic Republic of)
- Iraq
- Ireland
- Israel
- Italy
- Jamaica
- Japan
- Jordan
- Kazakhstan
- Kenya
- Kiribati
- Kuwait
- Kyrgyzstan
- Lao People's Democratic Republic
- Latvia
- Lebanon
- Lesotho
- Liberia
- Libya
- Lithuania
- Luxembourg
- Madagascar
- Malawi
- Malaysia
- Maldives
- Mali
- Malta
- Marshall Islands
- Mauritania
- Mauritius
- Mexico
- Micronesia (Federated States of)
- Monaco
- Mongolia
- Montenegro
- Morocco
- Mozambique
- Myanmar
- Namibia
- Nauru
- Nepal
- Netherlands
- New Zealand
- Nicaragua
- Niger
- Nigeria
- Niue


Norway
Oman
Pakistan
Palau
Panama
Papua New Guinea
Paraguay
Peru
Philippines
Poland
Portugal
Qatar
Republic of Korea
Republic of Moldova
Romania
Russian Federation
Rwanda
Saint Kitts and Nevis
Saint Lucia
Saint Vincent and the Grenadines
Samoa
San Marino
Sao Tome and Principe
Saudi Arabia
Senegal
Serbia
Seychelles
Sierra Leone
Singapore
Slovakia
Slovenia
Solomon Islands
Somalia
South Africa
South Sudan
Spain
Sri Lanka
Sudan
Suriname
Swaziland
Sweden
Switzerland
Syrian Arab Republic
Tajikistan
Thailand
The former Yugoslav republic of Macedonia
Timor-Leste
Togo
Tonga
Trinidad and Tobago
Tunisia
Turkey
Turkmenistan
Tuvalu
Uganda
Ukraine
United Arab Emirates
United Kingdom of Great Britain and Northern Ireland
United Republic of Tanzania
United States of America
Uruguay
Uzbekistan
Vanuatu
90
80
70
60
expectancy



Training and Testing Splitting

In [251...]
```
cdata.columns
```

Out[251]:
```
Index(['Country', 'Year', 'Status', 'Life expectancy', 'Adult Mortality',
       'Alcohol', 'percentage expenditure', 'Measles', 'BMI',
       'under-five deaths', 'Total expenditure', 'HIV/AIDS', 'GDP',
       'Population', 'thinness  1-19 years'],
      dtype='object')
```

In [252...]
```
cdata.describe()
```

Out[252]:

	Year	Life expectancy	Adult Mortality	Alcohol	percentage expenditure	Measles	BMI
count	2938.000000	2938.000000	2938.000000	2938.000000	2938.000000	2938.000000	2938.000000
mean	2007.518720	69.224932	164.796448	4.602861	738.251295	2419.592240	38.321247
std	4.613841	9.507640	124.080302	3.916288	1987.914858	11467.272489	19.927677
min	2000.000000	36.300000	1.000000	0.010000	0.000000	0.000000	1.000000
25%	2004.000000	63.200000	74.000000	1.092500	4.685343	0.000000	19.400000
50%	2008.000000	72.000000	144.000000	4.160000	64.912906	17.000000	43.000000
75%	2012.000000	75.600000	227.000000	7.390000	441.534144	360.250000	56.100000
max	2015.000000	89.000000	723.000000	17.870000	19479.911610	212183.000000	87.300000

In [253...]
```
cdata.shape
```

Out[253]:
```
(2938, 15)
```

In [254...]
```
Y1=cdata['Life expectancy']
X1=cdata.drop('Life expectancy',axis=1)
```

In [255...]
```
Country_dummy=pd.get_dummies(X1['Country'])
Status_dummy=pd.get_dummies(X1['Status'])
X1.drop(['Country','Status'],inplace=True,axis=1)
X2=pd.concat([X1,Country_dummy,Status_dummy],axis=1)
```

In [256...]
```
from sklearn.model_selection import train_test_split

x_train, x_test, y_train, y_test = train_test_split(X2, Y1, test_size=0.3)
```

```
print(f"Training set: {x_train.shape} and Test set: {x_test.shape}")
```

```
Training set: (2056, 207) and Test set: (882, 207)
```

In [257…
```
x_test
```

Out[257]:

	Year	Adult Mortality	Alcohol	percentage expenditure	Measles	BMI	under-five deaths	Total expenditure	HIV/AIDS	
2020	2002	144.0	4.030000	40.537779	0	46.7	20	4.94000	0.4	259
2234	2014	88.0	0.090000	2017.643131	154	67.3	9	4.68000	0.1	24575
2760	2001	14.0	1.670000	243.753913	30	55.0	1	2.48000	0.1	3161
398	2001	16.0	10.720000	25.062629	8	57.5	1	7.23000	0.1	1764
785	2000	176.0	6.580000	44.792478	253	43.1	9	5.90000	2.5	282
...	...	...	...	...	...	...	...	...	...	
1975	2015	275.0	4.602861	0.000000	38	48.6	12	5.93819	0.6	7483
433	2014	47.0	0.010000	0.000000	50	27.4	80	5.72000	2.0	7483
1277	2004	69.0	2.230000	1895.342792	116	6.1	1	7.35000	0.1	19888
852	2013	266.0	0.010000	0.000000	45	17.5	8	3.10000	0.5	7483
1501	2004	329.0	4.240000	17.858708	4	21.5	17	8.77000	3.2	149

882 rows × 207 columns

In [258…
```
x_train
```

Out[258]:

	Year	Adult Mortality	Alcohol	percentage expenditure	Measles	BMI	under-five deaths	Total expenditure	HIV/AIDS	
929	2000	15.0	8.59	397.753369	0	55.5	0	7.22	0.1	24253.2
460	2003	144.0	4.16	209.086500	0	23.2	0	5.00	0.9	1768.9
974	2003	297.0	2.47	0.000000	119	19.7	6	4.22	2.7	7483.1
2160	2006	328.0	6.88	78.470202	494	15.9	34	1.20	6.2	342.3
2917	2004	578.0	2.46	8.369852	35	18.0	59	7.33	17.6	53.2
...	...	...	...	...	...	...	...	...	...	
1770	2009	4.0	1.18	39.752169	60	2.1	101	5.43	11.3	463.8
748	2004	98.0	11.27	6948.839868	0	54.4	0	9.67	0.1	46511.6
2771	2006	82.0	11.61	0.000000	764	61.3	4	8.36	0.1	7483.1
1487	2002	622.0	2.95	3.534574	0	25.9	7	6.91	32.5	47.8
1299	2014	133.0	3.83	427.305453	0	53.5	1	5.36	0.5	4855.7

2056 rows × 207 columns

In [259…
```
y_test
```

Out[259]:
```
2020    72.6
2234    74.4
2760    74.5
398     71.6
785     72.0
        ...
1975    62.9
433     52.8
1277    81.0
852     64.0
1501    54.0
Name: Life expectancy, Length: 882, dtype: float64
```

In [260…
```
y_train
```

Out[260]:
```
929     77.5
460     71.1
974     57.0
2160    57.6
2917    47.9
        ...
1770    53.8
748     77.7
2771    79.3
1487    46.4
1299    75.8
Name: Life expectancy, Length: 2056, dtype: float64
```

Linear Regression

In [261…
```
from sklearn.linear_model import LinearRegression

linear_regressor = LinearRegression()
linear_regressor.fit(x_train, y_train)
```

Out[261]:
```
LinearRegression()
```

In [262…
```
y_pred_lr = linear_regressor.predict(x_test)
```

In [263…
```
from sklearn.metrics import mean_absolute_error as mae, mean_squared_error as mse
from sklearn.metrics import r2_score

print("***---- Linear Regression - Model Evaluation ----***")
print(f"Mean Absolute Error (MAE): {mae(y_test, y_pred_lr)}")
print(f"Mean Squared Error (MSE): {mse(y_test, y_pred_lr)}")
print(f"Root Mean Squared Error (RMSE): {np.sqrt(mse(y_test, y_pred_lr))}")
print("R2 Score : %.2f" %r2_score(y_test,y_pred_lr))
```

```
***---- Linear Regression - Model Evaluation ----***
Mean Absolute Error (MAE): 1.23620250799355
Mean Squared Error (MSE): 3.650517281144313
Root Mean Squared Error (RMSE): 1.9106326913209437
R2 Score : 0.96
```

Decision Tree

In [264…
```
from sklearn.tree import DecisionTreeRegressor

decision_regressor = DecisionTreeRegressor(min_samples_leaf=.000001)
decision_regressor.fit(x_train,y_train)
```

Out[264]:
```
DecisionTreeRegressor(min_samples_leaf=1e-06)
```

In [265…
```
y_pred_dt = decision_regressor.predict(x_test)
```

In [266…
```
print("***---- Decision Tree Regression - Model Evaluation ----***")
print(f"Mean Absolute Error (MAE): {mae(y_test, y_pred_dt)}")
print(f"Mean Squared Error (MSE): {mse(y_test, y_pred_dt)}")
print(f"Root Mean Squared Error(RMSE): {np.sqrt(mse(y_test, y_pred_dt))}")
print("R2 score : %.2f" % r2_score(y_test,y_pred_dt))
```

```
***---- Decision Tree Regression - Model Evaluation ----***
Mean Absolute Error (MAE): 1.6169220010408532
Mean Squared Error (MSE): 7.188887394226741
Root Mean Squared Error(RMSE): 2.681210061563014
R2 score : 0.92
```

Random Forest

In [267…
```
from sklearn.ensemble import RandomForestRegressor
```

```
random_regressor = RandomForestRegressor(n_estimators = 1000, random_state = 42)
random_regressor.fit(x_train, y_train)
```

Out[267]:
```
RandomForestRegressor(n_estimators=1000, random_state=42)
```

In [268…
```
y_pred_rf = random_regressor.predict(x_test)
```

In [269…
```
# Random Forest Regression - Model Evaluation

print("***---- Random Forest Regression - Model Evaluation ----***")
print(f"Mean Absolute Error (MAE): {mae(y_test, y_pred_rf)}")
print(f"Mean Squared Error (MSE): {mse(y_test, y_pred_rf)}")
print(f"Root Mean Squared Error (RMSE): {np.sqrt(mse(y_test, y_pred_rf))}")
print(f"R2 score : %.2f" % r2_score(y_test,y_pred_rf))
```

```
***---- Random Forest Regression - Model Evaluation ----***
Mean Absolute Error (MAE): 1.152860568457802
Mean Squared Error (MSE): 3.414329713104831
Root Mean Squared Error (RMSE): 1.8477904949167887
R2 score : 0.96
```

Panel Data Regression

1. Pooled OLS Model: (does not take the time factor into account, and treats the data like a normal cross-sectional data)¶

In [270…
```
pooled_x = cdata.drop(['Life expectancy', 'Country', 'Status'], axis=1)
```

In [271…
```
pooled_y = cdata['Life expectancy']
```

In [272…
```
pooled_x = sm.add_constant(pooled_x)

pooled_olsr_model = sm.OLS(endog=pooled_y, exog=pooled_x)

pooled_olsr_model_results = pooled_olsr_model.fit()

print(pooled_olsr_model_results.summary())
```

OLS Regression Results

Dep. Variable:	Life expectancy	R-squared:	0.709
Model:	OLS	Adj. R-squared:	0.708
Method:	Least Squares	F-statistic:	593.8
Date:	Sun, 27 Nov 2022	Prob (F-statistic):	0.00
Time:	10:23:38	Log-Likelihood:	-8971.8
No. Observations:	2938	AIC:	1.797e+04
Df Residuals:	2925	BIC:	1.805e+04
Df Model:	12		
Covariance Type:	nonrobust		

	coef	std err	t	P>|t|	[0.025	0.975]
const	-200.2370	42.582	-4.702	0.000	-283.730	-116.744
Year	0.1340	0.021	6.315	0.000	0.092	0.176
Adult Mortality	-0.0294	0.001	-30.427	0.000	-0.031	-0.027
Alcohol	0.3890	0.029	13.590	0.000	0.333	0.445
percentage expenditure	5.883e-05	0.000	0.552	0.581	-0.000	0.000
Measles	-3.365e-05	9.69e-06	-3.474	0.001	-5.26e-05	-1.47e-05
BMI	0.0938	0.006	15.652	0.000	0.082	0.106
under-five deaths	-0.0042	0.001	-4.946	0.000	-0.006	-0.003
Total expenditure	0.0928	0.043	2.167	0.030	0.009	0.177
HIV/AIDS	-0.4767	0.022	-21.363	0.000	-0.521	-0.433
GDP	0.0001	1.63e-05	6.289	0.000	7.03e-05	0.000
Population	9.016e-09	2.1e-09	4.290	0.000	4.9e-09	1.31e-08
thinness 1-19 years	-0.1148	0.030	-3.812	0.000	-0.174	-0.056

Omnibus:	266.636	Durbin-Watson:	0.716
Prob(Omnibus):	0.000	Jarque-Bera (JB):	541.154
Skew:	-0.590	Prob(JB):	3.09e-118
Kurtosis:	4.740	Cond. No.	2.48e+10

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.
[2] The condition number is large, 2.48e+10. This might indicate that there are strong multicollinearity or other numerical problems.