Project 2

Student: Kesav Adithya Venkidusamy
Course: DSC680 - Applied Data Science
Instructor: Professor Catherine Williams
Assignment: Project 2

Life Expectancy Prediction

Idea: Everything has an expiration date; humans are no exception either. The term “life expectancy” refers to the number of years a person can expect to live. By definition, life expectancy is based on an estimate of the average age that members of a particular population group will be when they die. We’re in an unprecedented era where humans are living longer with increased access to modern science and healthcare. It’s no secret, though, that life expectancy varies widely across the globe. Life expectancy depends on several factors, the two most important being gender and birth year. Generally, females have a slightly higher life expectancy than males due to biological differences. Other factors that influence life expectancy include:

- Race and ethnicity
- Family medical history
- Risky lifestyles

In this project, I aim to explore the parameters affecting the life span of individuals living in distinct countries and learn how the life span can be estimated with the help of machine learning models. I will also focus on exploring the parameters that greatly impact the life span of an individual.

Dataset The Global Health Observatory (GHO) data repository under World Health Organization (WHO) keeps track of the health status as well as many other related factors of all countries. The datasets are made available to the public for health data analysis.

https://www.kaggle.com/datasets/kumarajarshi/life-expectancy-who

The data-set related to life expectancy, and health factors for 193 countries have been collected from the same WHO data repository website and its corresponding economic data was collected from the United Nation website.

Abstract: Predict the key drivers for the Life Expectancy

Features and Target present in the dataset

1. Country - Country Observed
2. Year - Year Observed
3. Status - Status of the country; Developed or Developing Status
4. Adult Mortality - Adult Mortality Rates on both sexes (probability of dying between 15-60 years/1000 population).
5. Infant deaths - Number of Infant Deaths per 1000 population
6. Alcohol - Alcohol recorded per capita (15+) consumption (in liters of pure alcohol).
7. Percentage expenditure - Expenditure on health as a percentage of Gross Domestic Product per capita(%).
8. Hepatitis B - Hepatitis B (HepB) immunization coverage among 1-year-olds (%)
9. Measles - Number of reported Measles cases per 1000 population
10. BMI - Average Body Mass Index of the entire population
11. Under-five-deaths - Number of under-five deaths per 1000 population
12. Polio - Polio (Pol3) immunization coverage among 1-year-olds (%)
13. Total expenditure - General government expenditure on health as a percentage of total government expenditure (%)
14. Diphtheria - Diphtheria tetanus toxoid and pertussis (DTP3) immunization coverage among 1-year-olds (%)
15. HIV/AIDS - Deaths per 1 000 live births HIV/AIDS (0-4 years)
16. GDP - Gross Domestic Product per capita (in USD)
17. Population - The population of the country
18. thinness 1-19 years - Prevalence of thinness among children and adolescents for Age 10 to 19 (%)
19. thinness 5-9 years - Prevalence of thinness among children for Age 5 to 9(%)
20. Income composition of resources - Human Development Index in terms of income composition of resources (index ranging from 0 to 1)
21. Schooling - Number of years of Schooling(years)

Target:

Life expectancy — Life expectancy in age

Data Exploration

Importing libraries for data processing

In [ ]:
```
#GENERAL
import pandas as pd # data processing, CSV file I/O (e.g. pd.read_csv)
import numpy as np # linear algebra
```

```
import random

#VISUALIZATIONS
import seaborn as sns
import matplotlib.pyplot as plt
import missingno as msno
import plotly.express as px
from plotly.subplots import make_subplots
from scipy.stats import kstest, norm
import pycountry

#FEATURE EGNGG
from sklearn.preprocessing import MinMaxScaler
from sklearn.preprocessing import LabelEncoder
from imblearn.over_sampling import SMOTE
from sklearn.preprocessing import StandardScaler
from sklearn.impute import SimpleImputer
from sklearn.base import BaseEstimator, TransformerMixin

#MODEL
## Importing lib required for modeling
from sklearn.pipeline import Pipeline
from sklearn.model_selection import train_test_split, GridSearchCV
from sklearn.linear_model import LinearRegression
from sklearn.linear_model import Lasso
from sklearn.datasets import make_classification
from sklearn.model_selection import KFold
from mlxtend.feature_selection import SequentialFeatureSelector as sfs
import statsmodels.api as sm
from sklearn.metrics import mean_squared_error, r2_score

#General
import warnings
warnings.filterwarnings('ignore')
pd.options.display.max_columns = None
import plotly.io as pio
pio.renderers.default='notebook+pdf'
from IPython.display import Image
```

Source Data Analysis

In [2]:

```
## Reading input data and create dataframe
raw_df = pd.read_csv('Life_Expectancy_Data.csv')
```

In [3]:
```
## Showing few records from the dataframe using head command
raw_df.head()
```

Out[3]:

	Country	Year	Status	Life expectancy	Adult Mortality	infant deaths	Alcohol	percentage expenditure	Hepatitis B	Measles	BMI	under-five deaths	Polio	Total expenditure	D
0	Afghanistan	2015	Developing	65.0	263.0	62	0.01	71.279624	65.0	1154	19.1	83	6.0	8.16	
1	Afghanistan	2014	Developing	59.9	271.0	64	0.01	73.523582	62.0	492	18.6	86	58.0	8.18	
2	Afghanistan	2013	Developing	59.9	268.0	66	0.01	73.219243	64.0	430	18.1	89	62.0	8.13	
3	Afghanistan	2012	Developing	59.5	272.0	69	0.01	78.184215	67.0	2787	17.6	93	67.0	8.52	
4	Afghanistan	2011	Developing	59.2	275.0	71	0.01	7.097109	68.0	3013	17.2	97	68.0	7.87	

In [4]:
```
## showing the shape of the dataframe
raw_df.shape
```

Out[4]:
```
(2938, 22)
```

The dataset contains 2938 rows and 22 attributes!

In [5]:
```
## Showing the info of the dataframe
raw_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 [6]:

```
## Describe the numerical columns present in the dataframe
raw_df.describe().T
```

Out[6]:

	count	mean	std	min	25%	50%	75%	max
Year	2938.0	2.007519e+03	4.613841e+00	2000.00000	2004.000000	2.008000e+03	2.012000e+03	2.015000e+03
Life expectancy	2928.0	6.922493e+01	9.523867e+00	36.30000	63.100000	7.210000e+01	7.570000e+01	8.900000e+01
Adult Mortality	2928.0	1.647964e+02	1.242921e+02	1.00000	74.000000	1.440000e+02	2.280000e+02	7.230000e+02
infant deaths	2938.0	3.030395e+01	1.179265e+02	0.00000	0.000000	3.000000e+00	2.200000e+01	1.800000e+03
Alcohol	2744.0	4.602861e+00	4.052413e+00	0.01000	0.877500	3.755000e+00	7.702500e+00	1.787000e+01
percentage expenditure	2938.0	7.382513e+02	1.987915e+03	0.00000	4.685343	6.491291e+01	4.415341e+02	1.947991e+04
Hepatitis B	2385.0	8.094046e+01	2.507002e+01	1.00000	77.000000	9.200000e+01	9.700000e+01	9.900000e+01
Measles	2938.0	2.419592e+03	1.146727e+04	0.00000	0.000000	1.700000e+01	3.602500e+02	2.121830e+05
BMI	2904.0	3.832125e+01	2.004403e+01	1.00000	19.300000	4.350000e+01	5.620000e+01	8.730000e+01
under-five deaths	2938.0	4.203574e+01	1.604455e+02	0.00000	0.000000	4.000000e+00	2.800000e+01	2.500000e+03
Polio	2919.0	8.255019e+01	2.342805e+01	3.00000	78.000000	9.300000e+01	9.700000e+01	9.900000e+01
Total expenditure	2712.0	5.938190e+00	2.498320e+00	0.37000	4.260000	5.755000e+00	7.492500e+00	1.760000e+01
Diphtheria	2919.0	8.232408e+01	2.371691e+01	2.00000	78.000000	9.300000e+01	9.700000e+01	9.900000e+01
HIV/AIDS	2938.0	1.742103e+00	5.077785e+00	0.10000	0.100000	1.000000e-01	8.000000e-01	5.060000e+01
GDP	2490.0	7.483158e+03	1.427017e+04	1.68135	463.935626	1.766948e+03	5.910806e+03	1.191727e+05

	count	mean	std	min	25%	50%	75%	max
Population	2286.0	1.275338e+07	6.101210e+07	34.00000	195793.250000	1.386542e+06	7.420359e+06	1.293859e+09
thinness 1-19 years	2904.0	4.839704e+00	4.420195e+00	0.10000	1.600000	3.300000e+00	7.200000e+00	2.770000e+01
thinness 5-9 years	2904.0	4.870317e+00	4.508882e+00	0.10000	1.500000	3.300000e+00	7.200000e+00	2.860000e+01
Income composition of resources	2771.0	6.275511e-01	2.109036e-01	0.00000	0.493000	6.770000e-01	7.790000e-01	9.480000e-01
Schooling	2775.0	1.199279e+01	3.358920e+00	0.00000	10.100000	1.230000e+01	1.430000e+01	2.070000e+01

Observation

- The data has around 15 years of data with Year range from 2000 to 2015
- The Adult Mortality column has a minimum value of 1 and maximum value of 74
- The infant deaths field has a range from 0 to 1800
- The Alchohol consumption ranges between 0.01 to 17.87
- The percentage expenditure has a range of values from 0 to 19479; Total expenditure field has a minimum value of 0.37 and max value of 17.6
- Hepatitis B column has numerical values between 1 and 99. This field has lot of null values.
- Measles field has a range of values between 0 and 212183
- BMI field has range of values from 1 to 87.3. It has few blank values.
- under-five deaths is a continous variable with range from 0 to 2500.
- Polio and Diphtheria are continuous variables denoting the immunization records. The range of values are between 3 and 99 and 2 nd 99 respectively.
- HIV/AIDS field (Deaths per 1 000 live births HIV/AIDS) has range of values from 0.1 to 50.6
- GDP and Population fields are also continuous variables which min value of 1.68135 and 34 respectively and max value of 119172.7 and 1.293859e+09 respectively.

- "thinness 1-19 years" and "thinness 5-9 years" fields denote Prevalence of thinness among children and the values range between 0.10 and 2.770000e+01 and 0.1 and 2.860000e+01 respectively.
- Schooling is also continuous variable with values between 0 and 20.7

One interesting thing in this dataset is that the names of the columns are not written in a nice manner, like in some names there is a space before the name, in some after the name and in some names there is a space in both before and after the name like " BMI ". To tackle this, let's print the names of all the columns and rename space with underscore.

In [7]:
```
## Renaming the fields
raw_df.columns.values.tolist()
```

Out[7]:
```
['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']
```

In [8]:
```
## Renaming the columns with space
raw_df.rename(columns={'Life expectancy ': 'Life_expectancy',
                       'Adult Mortality': 'Adult_Mortality',
                       'infant deaths':'infant_deaths',
                       'percentage expenditure':'percentage_expenditure',
                       'Hepatitis B':'Hepatitis_B','Measles ':'Measles',
                       ' BMI ':'BMI',
                       'under-five deaths ':'under_five_deaths',
```

```
                        'Total expenditure':'Total_expenditure','Diphtheria ':'Diphtheria',
                        ' HIV/AIDS':'HIV/AIDS',
                        ' thinness  1-19 years':'thinness_1_19_years',
                        ' thinness 5-9 years': 'thinness_5_9_years',
                        'Income composition of resources':'Income_composition_of_resources'}, inplace=True)
```

In [9]:
```
## Printing after renaming the fields
raw_df.columns.values.tolist()
```

Out[9]:
```
['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']
```

In [10]:
```
## Analyzing the target variable
raw_df['Life_expectancy'].value_counts()
```

Out[10]:
```
73.0    45
75.0    33
78.0    31
73.6    28
81.0    25
        ..
41.5     1
83.3     1
49.5     1
39.0     1
```

```
55.2     1
Name: Life_expectancy, Length: 362, dtype: int64
```

Observation

The "Life_expectancy" column is our target variable which is continuous in nature having range of values between 36.3 and 89.

In [11]:
```
## Printing the info of the dataframe
raw_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 [12]:
```
## Identifying the numerical columns in the dataset
num_cols = raw_df.select_dtypes(include=[np.float64, np.int64]).columns.tolist()
print(num_cols)
```

```
['Year', 'Life_expectancy', 'Adult_Mortality', 'infant_deaths', 'Alcohol', 'percentage_expenditure', 'Hepatitis_B', 'Meas
les', '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']
```

In [13]:
```
## Identifying categorical variables in the dataset
cat_cols = raw_df.select_dtypes('object').columns.tolist()
print(cat_cols)
```

```
['Country', 'Status']
```

In [14]:
```
## Copying the dataframe to another dataframe
rename_raw_df = raw_df.copy(deep=True)
```

In [15]:
```
## Printing few values from copied dataframe
rename_raw_df.head()
```

Out[15]:

	Country	Year	Status	Life_expectancy	Adult_Mortality	infant_deaths	Alcohol	percentage_expenditure	Hepatitis_B	Measles	BMI
0	Afghanistan	2015	Developing	65.0	263.0	62	0.01	71.279624	65.0	1154	19.1
1	Afghanistan	2014	Developing	59.9	271.0	64	0.01	73.523582	62.0	492	18.6
2	Afghanistan	2013	Developing	59.9	268.0	66	0.01	73.219243	64.0	430	18.1
3	Afghanistan	2012	Developing	59.5	272.0	69	0.01	78.184215	67.0	2787	17.6
4	Afghanistan	2011	Developing	59.2	275.0	71	0.01	7.097109	68.0	3013	17.2

EDA

In [16]:
```
## Analyzing all the categorical variables;
## Printing all the unique values of the categorical variables
## Summary of categorical variables
cat_df=pd.DataFrame(rename_raw_df[cat_cols].melt(var_name='column', value_name='value')
                    .value_counts()).rename(columns={0: 'count'}).sort_values(by=['column', 'count'])
display(rename_raw_df.select_dtypes(include=object).describe())
display(cat_df)
```

	Country	Status
count	2938	2938

	Country	Status
unique	193	2
top	Paraguay	Developing
freq	16	2426

		count
column	**value**	
Country	**Saint Kitts and Nevis**	1
	Marshall Islands	1
	Monaco	1
	Tuvalu	1
	Nauru	1
	...	...
	Germany	16
	Ghana	16
	Japan	16
Status	**Developed**	512
	Developing	2426

195 rows × 1 columns

In [17]:
```
## Printing unique value count for all the variables
print("All columns Unique values count")
for col in rename_raw_df:
    print(col, len(rename_raw_df[col].unique()), sep=': ')
```

```
All columns Unique values count
Country: 193
Year: 16
Status: 2
Life_expectancy: 363
Adult_Mortality: 426
```

```
infant_deaths: 209
Alcohol: 1077
percentage_expenditure: 2328
Hepatitis_B: 88
Measles: 958
BMI: 609
under_five_deaths: 252
Polio: 74
Total_expenditure: 819
Diphtheria: 82
HIV/AIDS: 200
GDP: 2491
Population: 2279
thinness_1_19_years: 201
thinness_5_9_years: 208
Income_composition_of_resources: 626
Schooling: 174
```

Handling Null Values

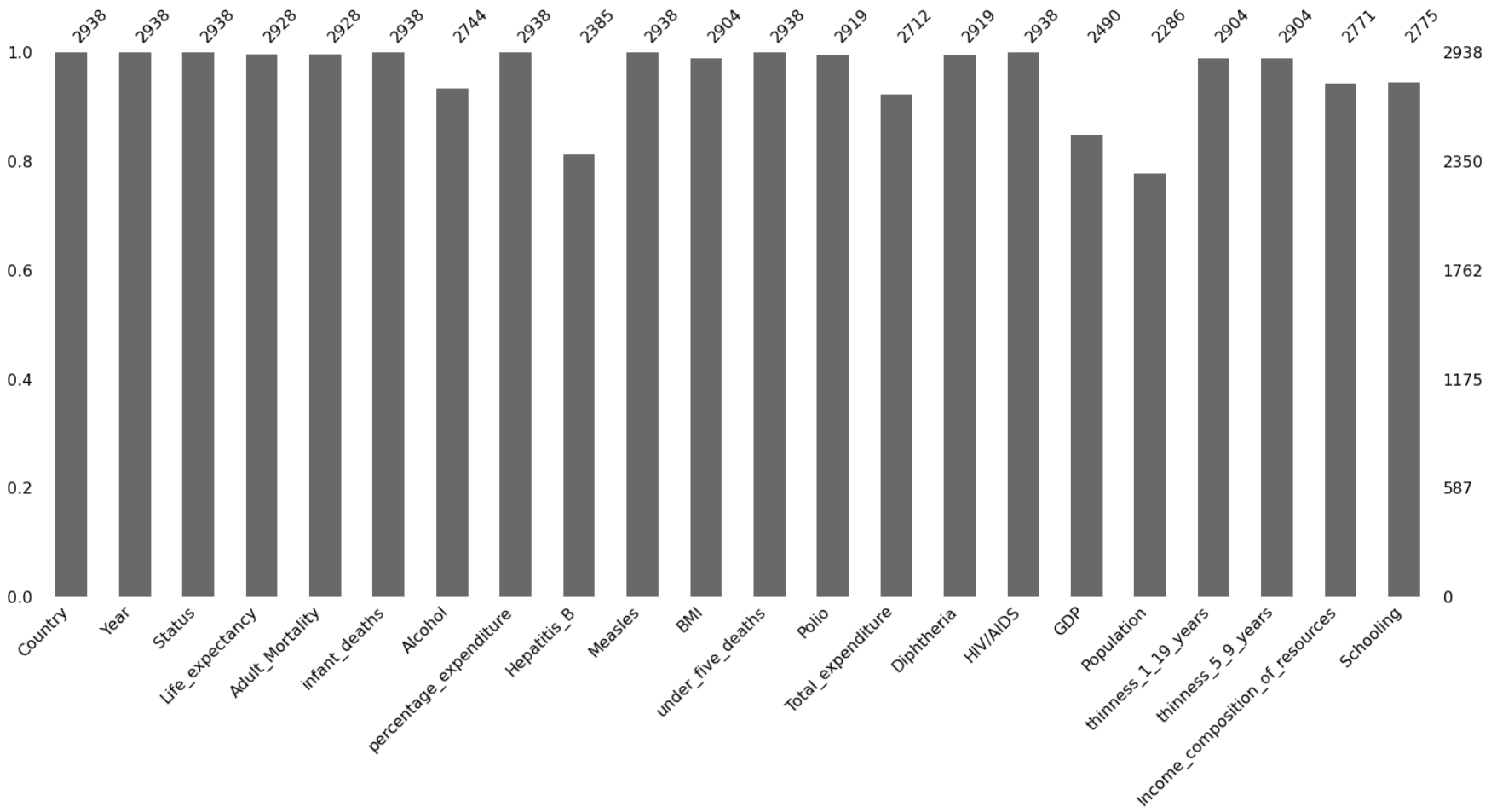
In [18]:
```
## Printing list of null values present in each column
rename_raw_df.isnull().sum()
```

Out[18]:
```
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
```

Missing values can prove to be a major pain as they may affect the run of the algorithms as almost all the algorithms expect the full data but return error if some points are missing. So, cleaning the data is essential!

In [19]:
```
# using missingno library to get a feel of the missing entries
msno.bar(rename_raw_df)
```

Out[19]: <AxesSubplot:>



Observation

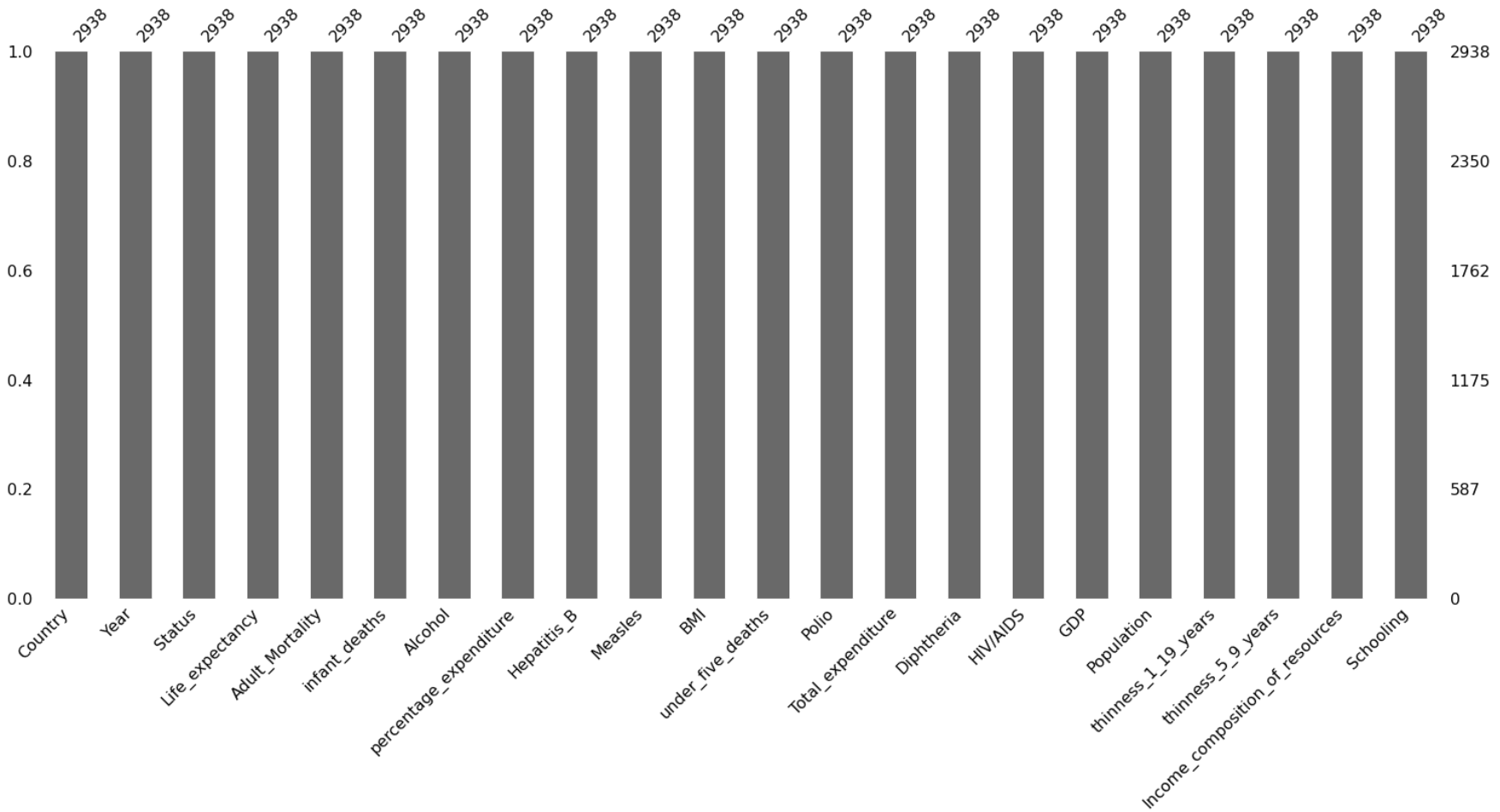
From the above bar chart, I can see that many attributes have missing values, like Hepatitis B has 2385 values, whereas the expected number of values for every attribute is 2938. Now we have to find a way to fill in all these missing values as these may cause problems for

our algorithm. We'll use the impute method of pd.DataFrame.fillna and impute the previous values in all these missing fields, previous values can be a good way to fill in such entries.

In [20]:
```
#### Replace null values with forward fill for all null value fields.
rename_raw_df["Hepatitis_B"].fillna( method ='ffill', inplace = True)
rename_raw_df["Alcohol"].fillna( method ='ffill', inplace = True)
rename_raw_df["Adult_Mortality"].fillna( method ='ffill', inplace = True)
rename_raw_df["Polio"].fillna( method ='ffill', inplace = True)
rename_raw_df["Total_expenditure"].fillna( method ='ffill', inplace = True)
rename_raw_df["GDP"].fillna( method ='ffill', inplace = True)
rename_raw_df["Population"].fillna( method ='ffill', inplace = True)
rename_raw_df["Schooling"].fillna( method ='ffill', inplace = True)
rename_raw_df["Income_composition_of_resources"].fillna( method ='ffill', inplace = True)
rename_raw_df["Life_expectancy"].fillna( method ='ffill', inplace = True)
rename_raw_df["Diphtheria"].fillna( method ='ffill', inplace = True)
rename_raw_df["thinness_5_9_years"].fillna( method ='ffill', inplace = True)
rename_raw_df["thinness_1_19_years"].fillna( method ='ffill', inplace = True)
rename_raw_df["BMI"].fillna( method ='ffill', inplace = True)
```

In [21]:
```
# using missingno library to get a feel of the missing entries after null value replacement
msno.bar(rename_raw_df)
```

Out[21]: <AxesSubplot:>



Duplicate check

In [22]:
```
# Selecting duplicate rows
# occurrence based on all columns
duplicate = rename_raw_df[rename_raw_df.duplicated()]
duplicate
```

Out[22]:

Country	Year	Status	Life_expectancy	Adult_Mortality	infant_deaths	Alcohol	percentage_expenditure	Hepatitis_B	Measles	BMI	under_fiv

Observation

There is no duplicate value present in the dataframe

In [23]:
```
## Creating a final dataframe
life_exp_df = rename_raw_df.copy(deep=True)
```

In [24]:
```
## Printing few records from life_exp_df
life_exp_df.head()
```

Out[24]:

	Country	Year	Status	Life_expectancy	Adult_Mortality	infant_deaths	Alcohol	percentage_expenditure	Hepatitis_B	Measles	BMI
0	Afghanistan	2015	Developing	65.0	263.0	62	0.01	71.279624	65.0	1154	19.1
1	Afghanistan	2014	Developing	59.9	271.0	64	0.01	73.523582	62.0	492	18.6
2	Afghanistan	2013	Developing	59.9	268.0	66	0.01	73.219243	64.0	430	18.1
3	Afghanistan	2012	Developing	59.5	272.0	69	0.01	78.184215	67.0	2787	17.6
4	Afghanistan	2011	Developing	59.2	275.0	71	0.01	7.097109	68.0	3013	17.2

In [25]:
```
## Displaying the Categorical and numerical columns
life_exp_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                  2938 non-null   float64
 4   Adult_Mortality                  2938 non-null   float64
 5   infant_deaths                    2938 non-null   int64
 6   Alcohol                          2938 non-null   float64
 7   percentage_expenditure           2938 non-null   float64
 8   Hepatitis_B                      2938 non-null   float64
 9   Measles                          2938 non-null   int64
 10  BMI                              2938 non-null   float64
 11  under_five_deaths                2938 non-null   int64
 12  Polio                            2938 non-null   float64
 13  Total_expenditure                2938 non-null   float64
 14  Diphtheria                       2938 non-null   float64
 15  HIV/AIDS                         2938 non-null   float64
```

```
 16  GDP                              2938 non-null   float64
 17  Population                       2938 non-null   float64
 18  thinness_1_19_years              2938 non-null   float64
 19  thinness_5_9_years               2938 non-null   float64
 20  Income_composition_of_resources  2938 non-null   float64
 21  Schooling                        2938 non-null   float64
dtypes: float64(16), int64(4), object(2)
memory usage: 505.1+ KB
```

In [26]:

```
## Identifying the numerical columns in the dataset
print("Numberical Columns")
print(num_cols)

print("\nCategorical Columns")
print(cat_cols)
```

```
Numberical Columns
['Year', 'Life_expectancy', 'Adult_Mortality', 'infant_deaths', 'Alcohol', 'percentage_expenditure', 'Hepatitis_B', 'Meas
les', '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']

Categorical Columns
['Country', 'Status']
```

Visualizations

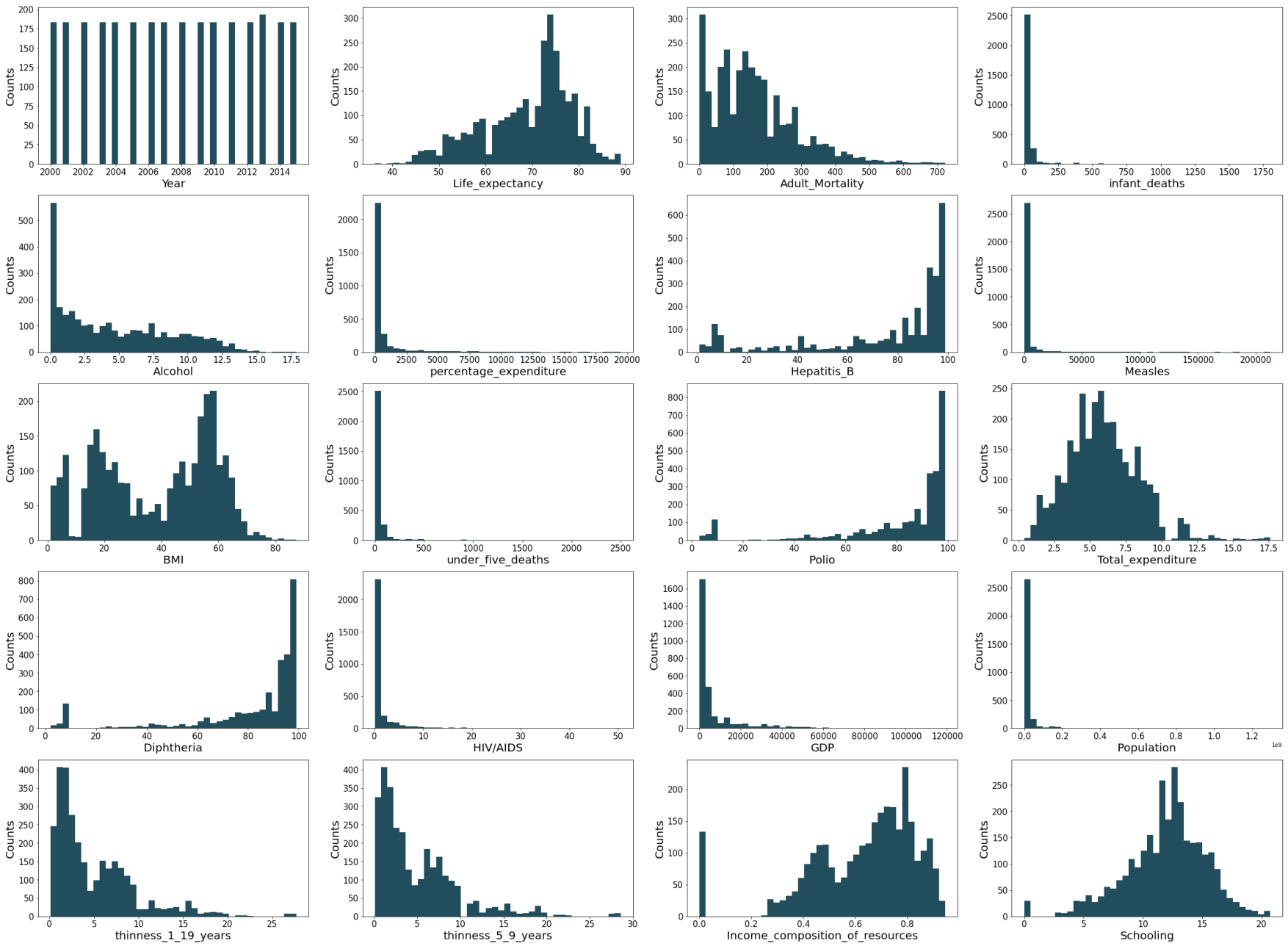
Numerical Variable Exploration

Histogram

In [27]:

```
## Make some histograms of your data
## Specify the features of interest
features = ['Year', 'Life_expectancy', 'Adult_Mortality', 'infant_deaths', 'Alcohol', 'percentage_expenditure', 'Hepatiti
xaxes = features
yaxes = ['Counts', 'Counts', 'Counts', 'Counts', 'Counts', 'Counts','Counts','Counts','Counts','Counts','Counts','Counts'
# set up the figure size
plt.rcParams['figure.figsize'] = (40, 30)
# make subplots
fig, axes = plt.subplots(nrows = 5, ncols = 4)
# draw histograms
axes = axes.ravel()
for idx, ax in enumerate(axes):
    ax.hist(life_exp_df[features[idx]].dropna(), bins=40, color='#214D5C')
    ax.set_xlabel(xaxes[idx], fontsize=20)
```

```
    ax.set_ylabel(yaxes[idx], fontsize=20)
    ax.tick_params(axis='both', labelsize=15)
plt.show()
```



Observation

Right Skewed

From the histogram chart, we see all the below features are rightly skewed as they have a “tail” on the right side of the distribution. The frequency of occurence of values is high at at the beginning and low towards the end.

- Adult Mortality
- Infant_death
- Alcohol
- Percentage Expenditure
- Measles
- Under five deaths
- HIV/AIDS
- GDP
- Population
- thinness_1_19_years
- thinness_5_9_years

Left Skewed

From the histogram chart, we see all the below features are left skewed as they have a “tail” on the left side of the distribution. The frequency of occurence of values is low at the beginning and high at the end .

- Hepatitis B
- Diphtheria
- Income composition of resources
- Polio

Normal Distribution

A normal distribution is an arrangement of a data set in which most values cluster in the middle of the range and the rest taper off symmetrically toward either extreme. Below features are having kind of normalized distribution

- Life Expectancy - The value is high between the age of 70 to 75 all over the world
- Total Expenditure - The value is high around 5
- Schooling - The maximum occurred between the range of 10 to 15

Mulitmode Distribution

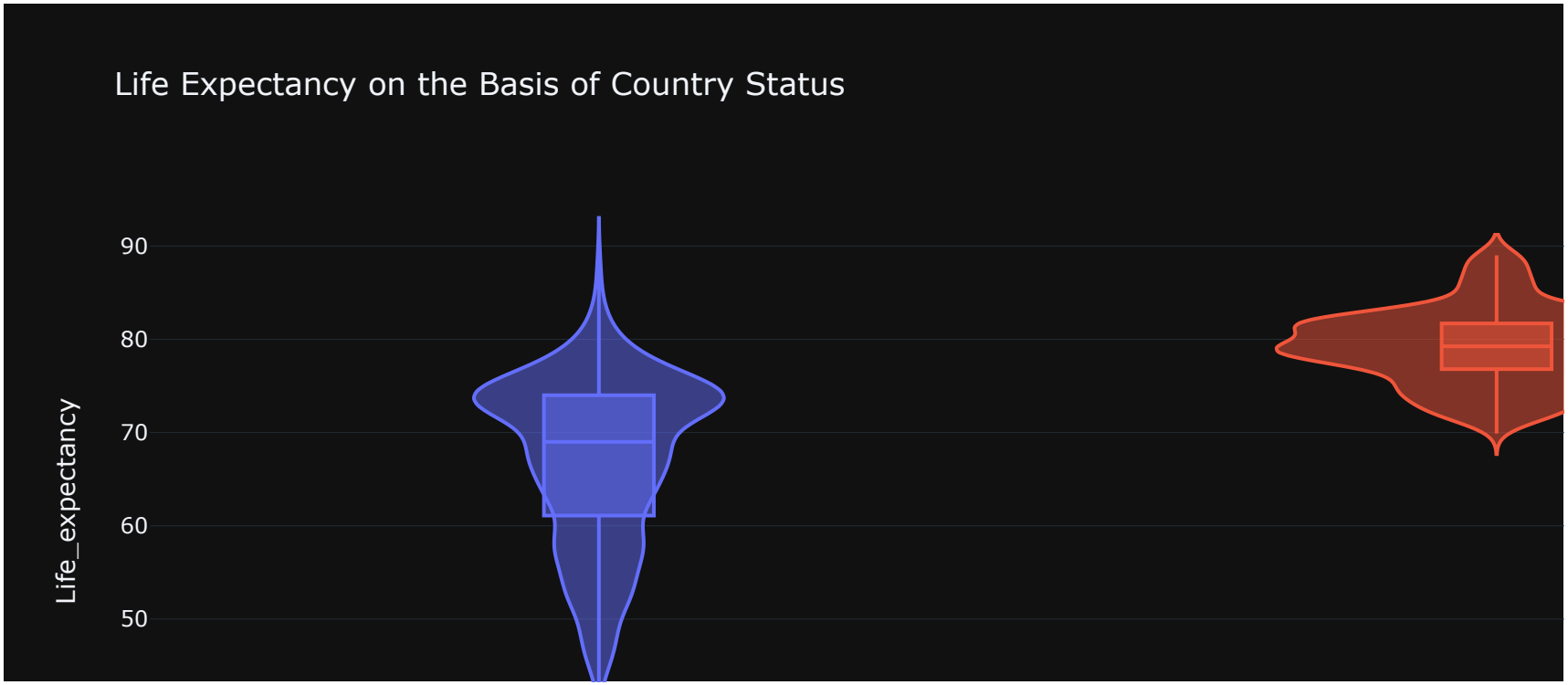
A multimodal distribution is a probability distribution with more than one peak, or “mode.” A bimodal distribution is also multimodal, as there are multiple peaks. The below feature has multimodal distributio

- BMI

Categorical Variable Exploration

Comparing the life expectancy of Developing and Developed Countries using violin Chart

In [28]:
```
# Developing vs Developed Countries
fig = px.violin(life_exp_df, x= 'Status', y= 'Life_expectancy',template='plotly_dark',
                color = 'Status',box = True,title='Life Expectancy on the Basis of Country Status')
fig.show()
```

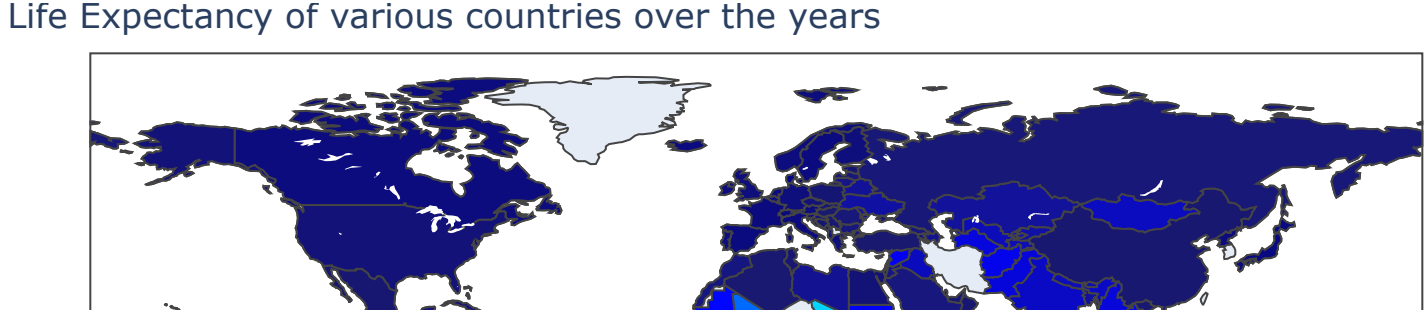


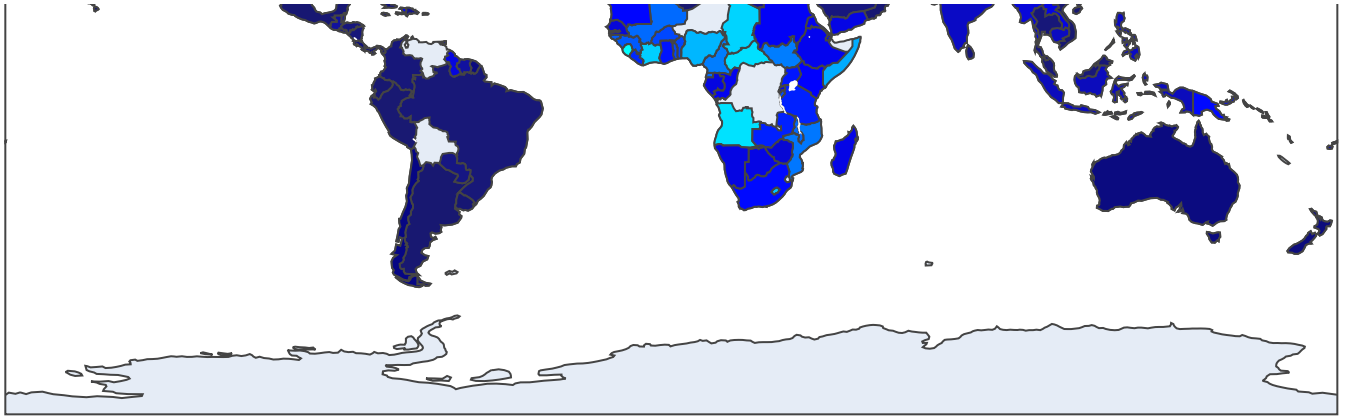
Observation

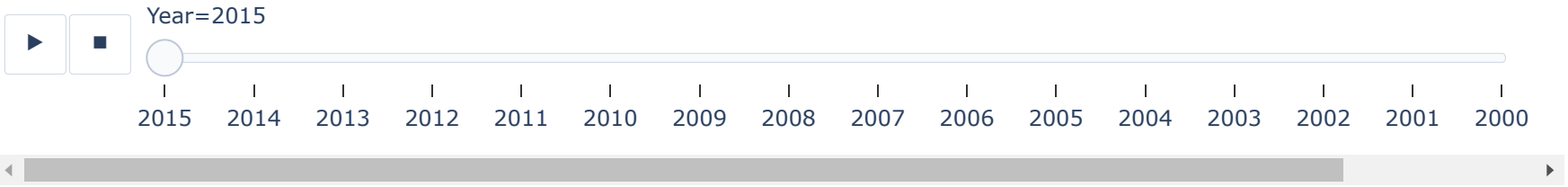
Above the graph we could see Developing countries have low life expectancy and the developed countries have high life expectancy all over the world

Country Wise Life Expectancy over the years using country and line plot

In [29]:
```
list_countries = life_exp_df['Country'].unique().tolist()
d_country_code = {}

for country in list_countries:
    try:
        country_data = pycountry.countries.search_fuzzy(country)
        country_code = country_data[0].alpha_3
        d_country_code.update({country: country_code})
    except:
        d_country_code.update({country: ' '})

for k, v in d_country_code.items():
    life_exp_df.loc[(life_exp_df.Country == k), 'iso_alpha'] = v

fig = px.choropleth(data_frame = life_exp_df,
                    title = "Life Expectancy of various countries over the years",
                    locations= "iso_alpha",
                    color= "Life_expectancy",
                    hover_name= "Country",
                    color_continuous_scale= ['cyan','blue','midnightblue','darkblue'],
                    animation_frame= "Year", width=1000, height=500)
fig.update_layout(margin={"r":0,"t":25,"l":0,"b":0})
fig.show()
```



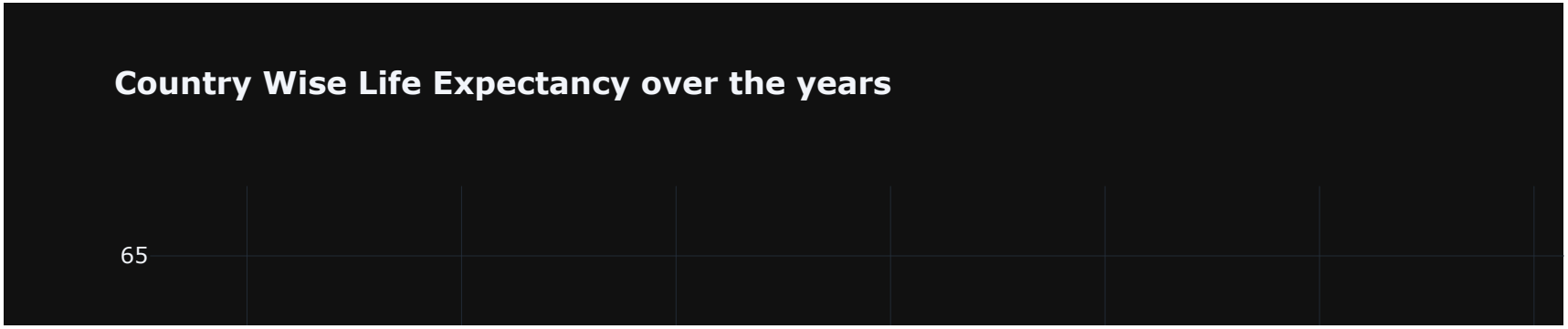


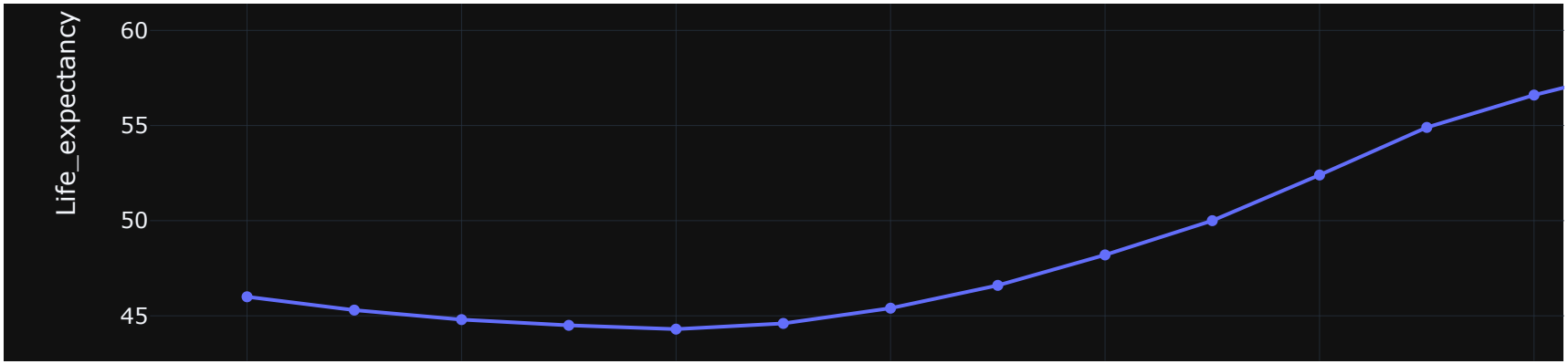


Observation

Looking at the chart, we see the life expectancy is high for developed countries compared to developing countries. In addition, we could also notice that the life expectancy increase over the years across the countries.

In [30]:
```
fig = px.line((life_exp_df.sort_values(by = 'Year')), x = 'Year', y = 'Life_expectancy',
    animation_frame='Country',animation_group='Year',color='Country',
    markers=True,template = 'plotly_dark',title='<b>Country Wise Life Expectancy over the years')
fig.show()
```





Observation

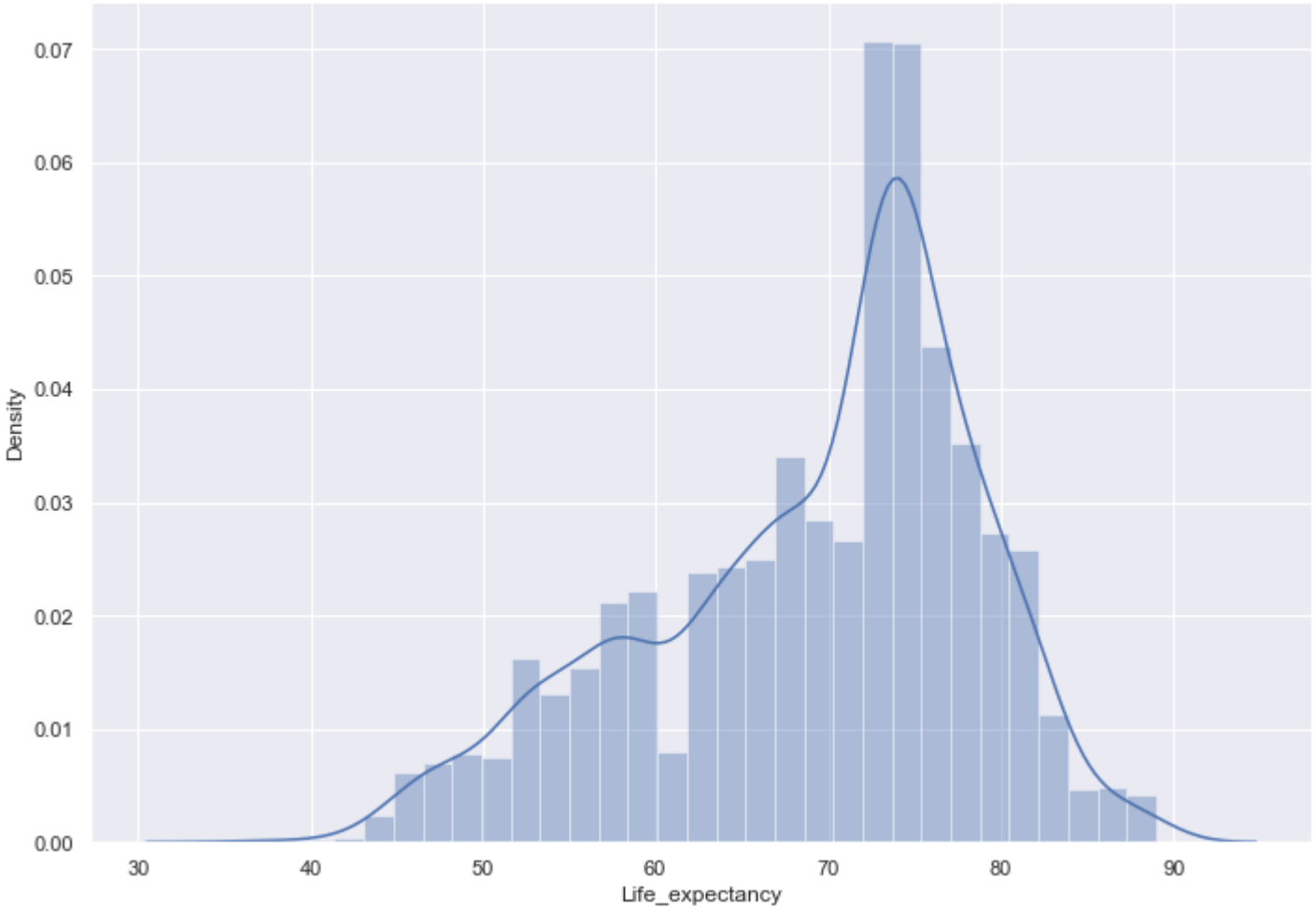
As noticed in the country chart, this chart also shows that life expectancy for the countries increases over the years. This is particulary for developing countries compared to developed countries.

Target Variable Analysis

Distribution Plot

In [31]:
```
## istribution Plot for Life Expectancy
sns.set(rc={'figure.figsize':(11.7,8.27)})
sns.distplot(life_exp_df['Life_expectancy'])
```

Out[31]:
```
<AxesSubplot:xlabel='Life_expectancy', ylabel='Density'>
```



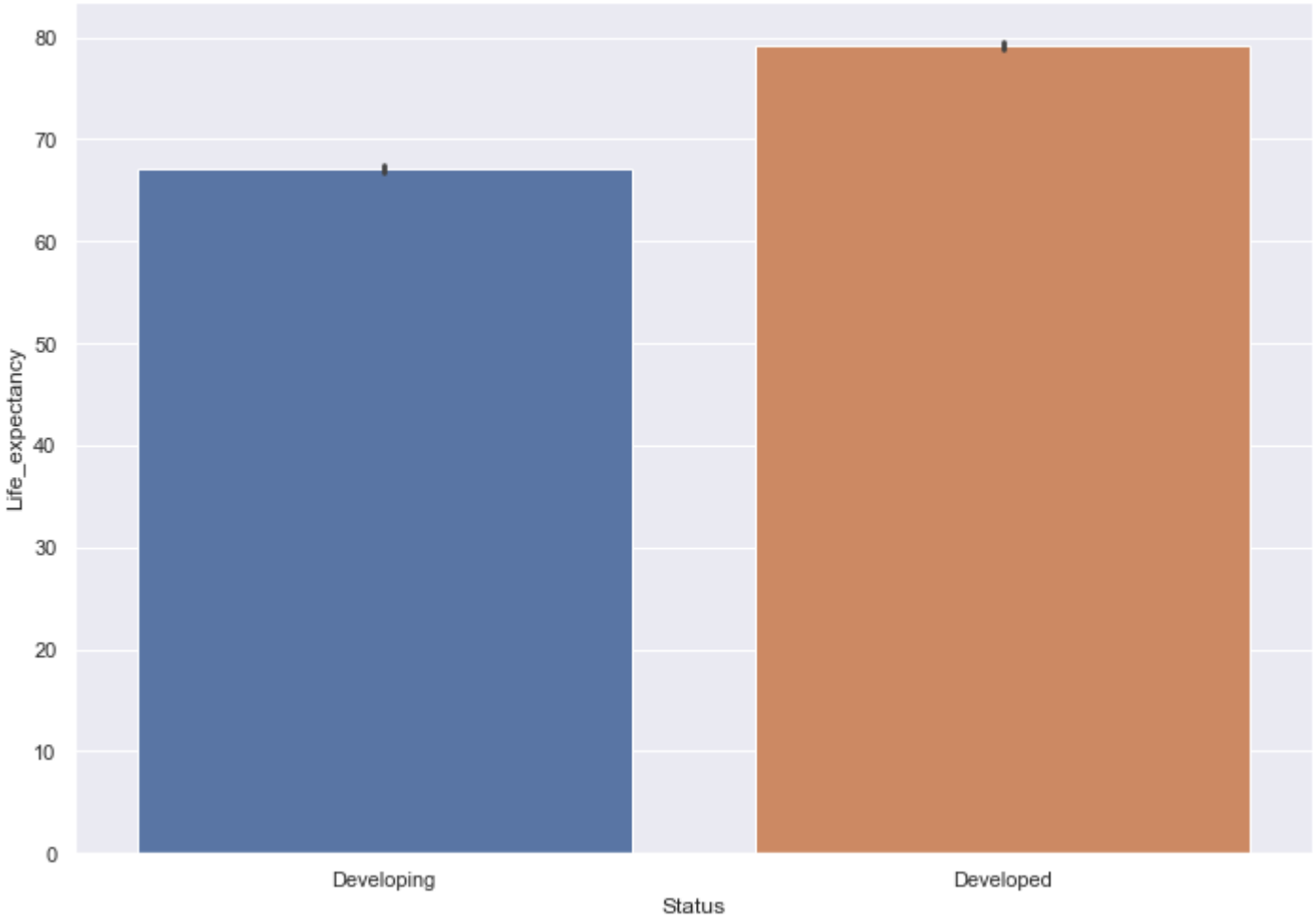
Observation:

Distribution plot shows that life expectancy has high value around 72. This includes all the countries and for all the years.

Bar Chart

In [32]:
```
## Plotting bar graph by status
sns.set(font_scale=2.5)
sns.set(rc={'figure.figsize':(11.7,8.27)})
sns.barplot(life_exp_df["Status"],life_exp_df["Life_expectancy"])
```

Out[32]:
```
<AxesSubplot:xlabel='Status', ylabel='Life_expectancy'>
```



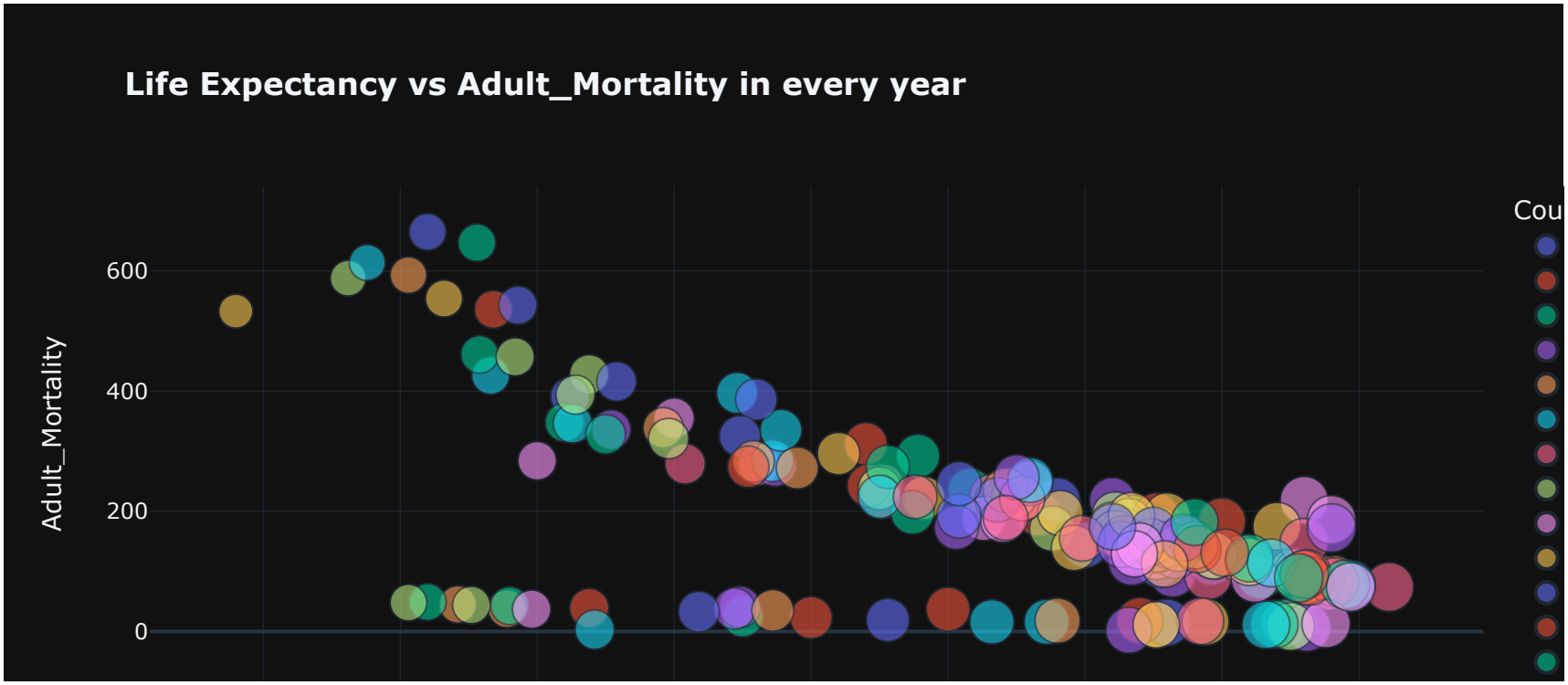
Observation

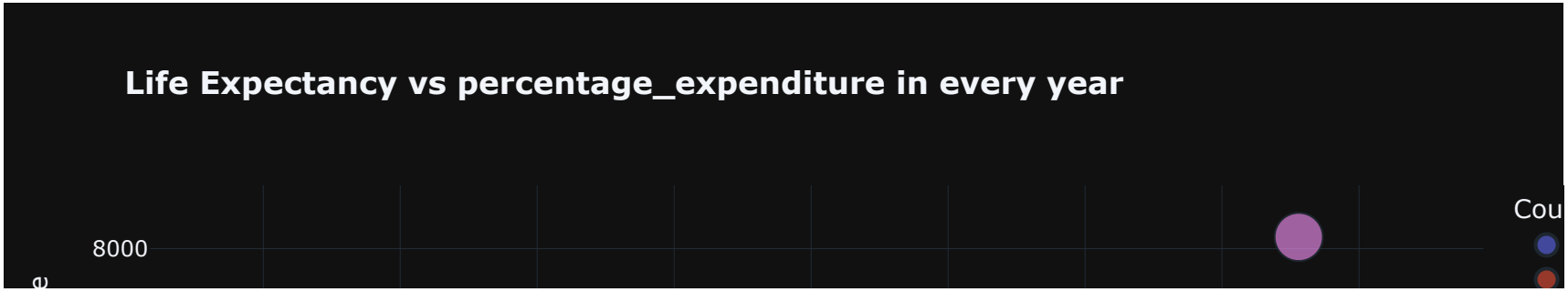
The life expectancy is high for developing countries compared to developed countries.

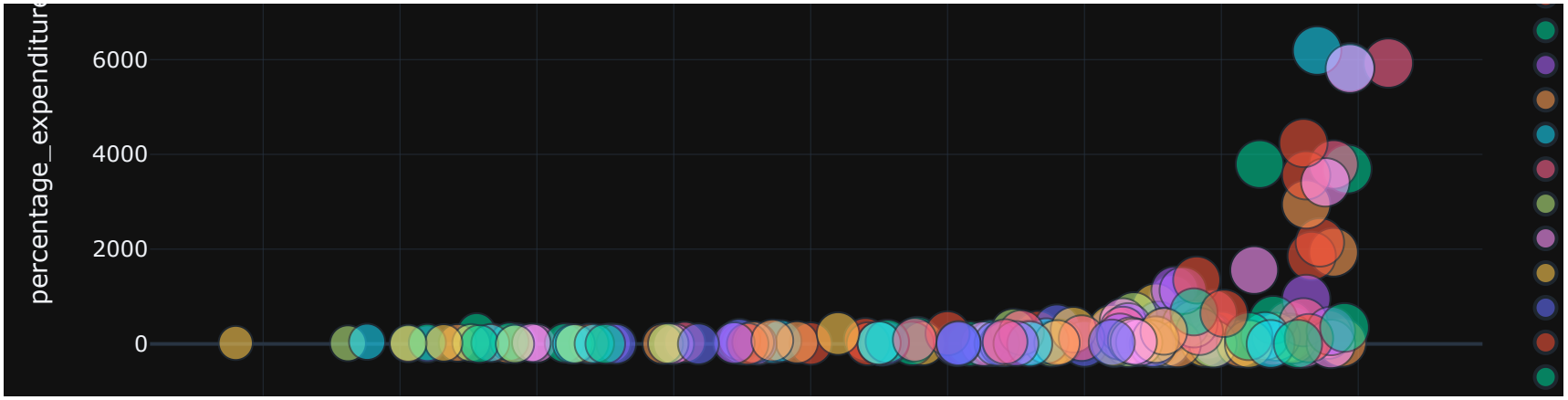
Life Expectancy vs features using Scatter Plot

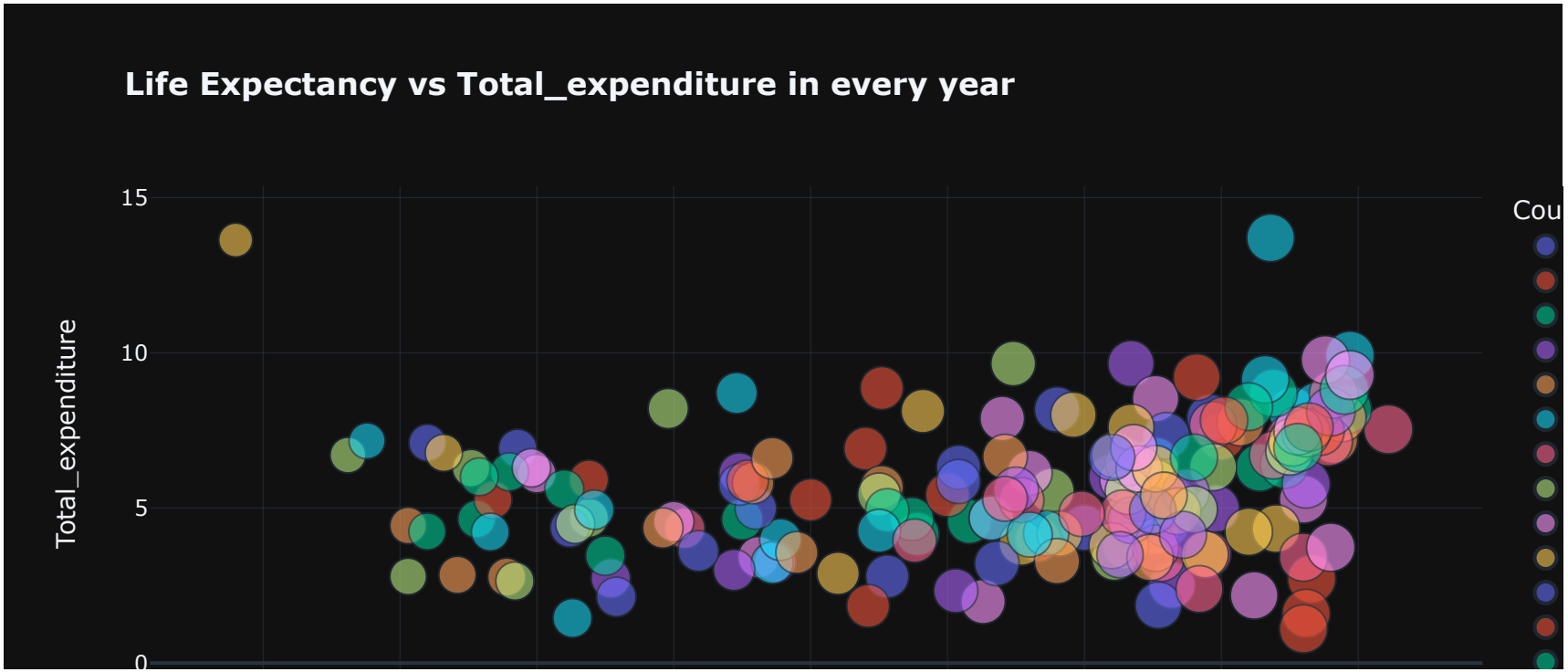
In [33]:
```
features = ['Adult_Mortality', 'percentage_expenditure','Total_expenditure','under_five_deaths','BMI','Schooling']

for feature in features:
    title = '<b> Life Expectancy vs '+ feature + ' in every year'
    fig = px.scatter(life_exp_df.sort_values(by='Year'),y=feature,x='Life_expectancy',animation_frame='Year',
                     animation_group='Country', color='Country',
```

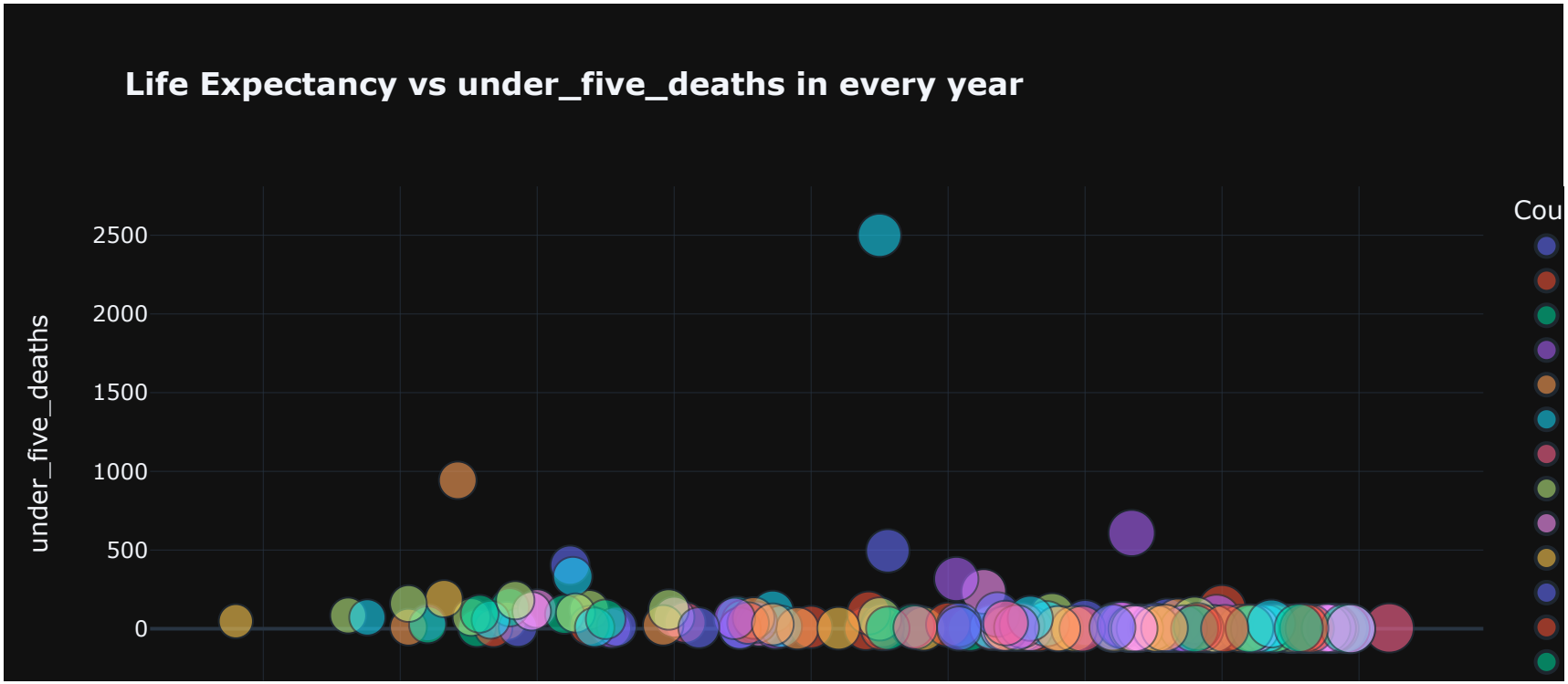
```
                    size='Life_expectancy',template='plotly_dark',opacity=0.6, title=title)
fig.show()
```

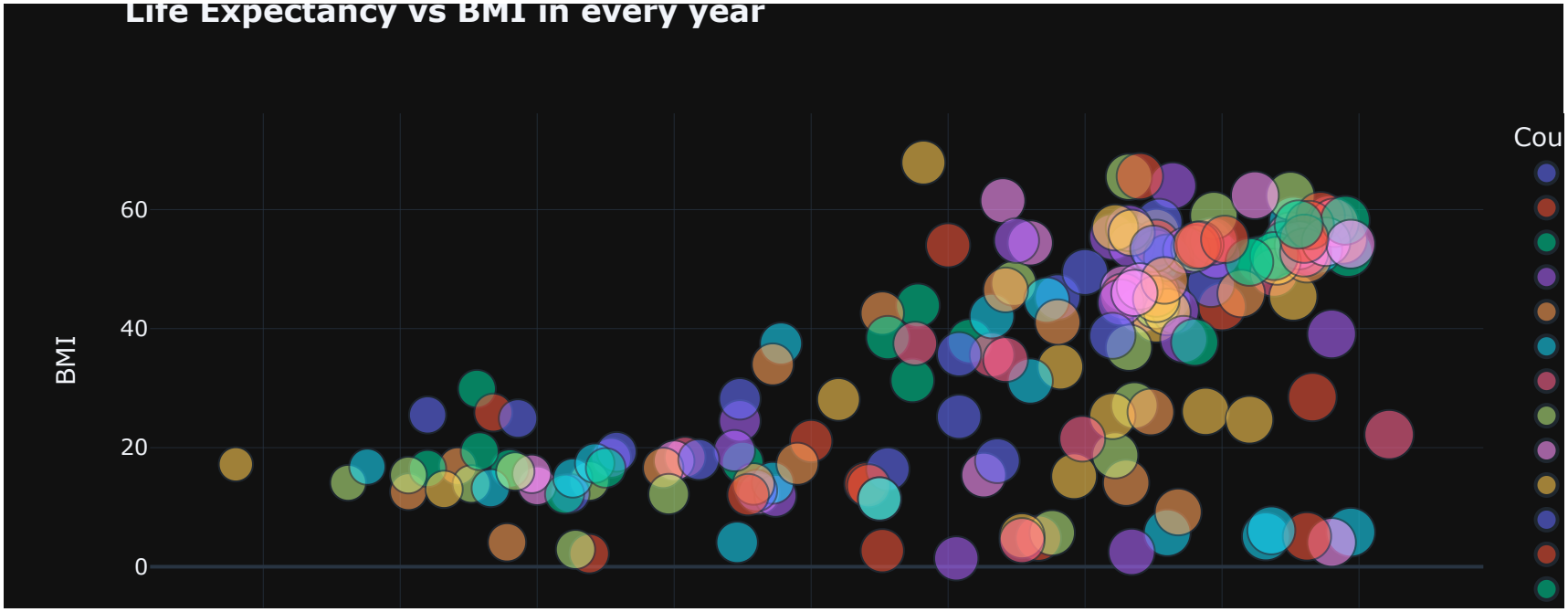


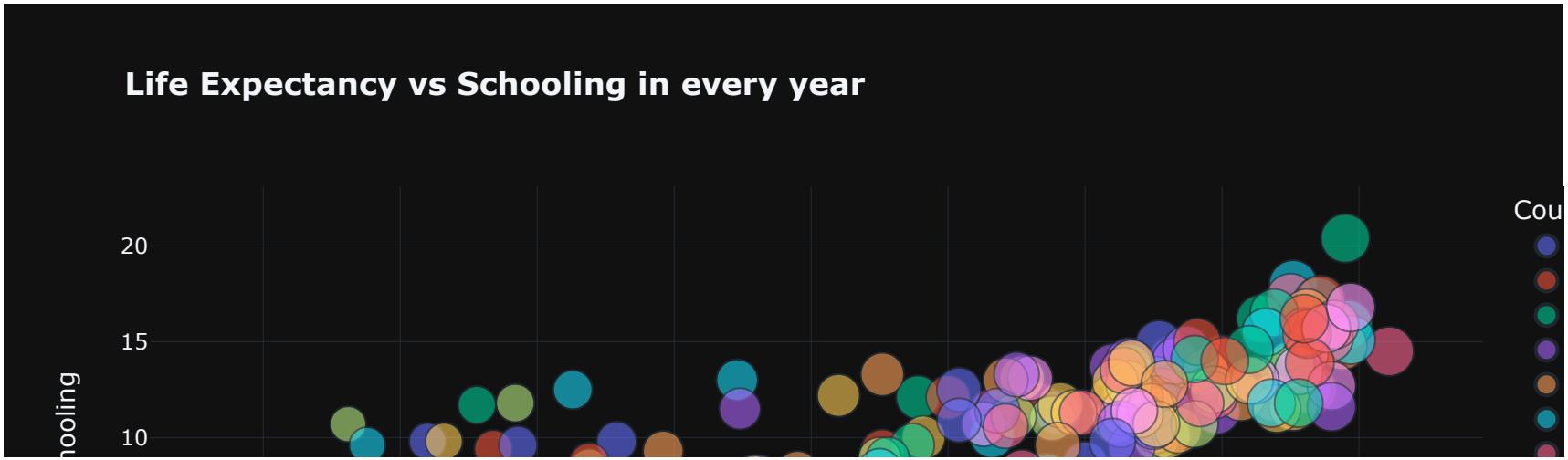


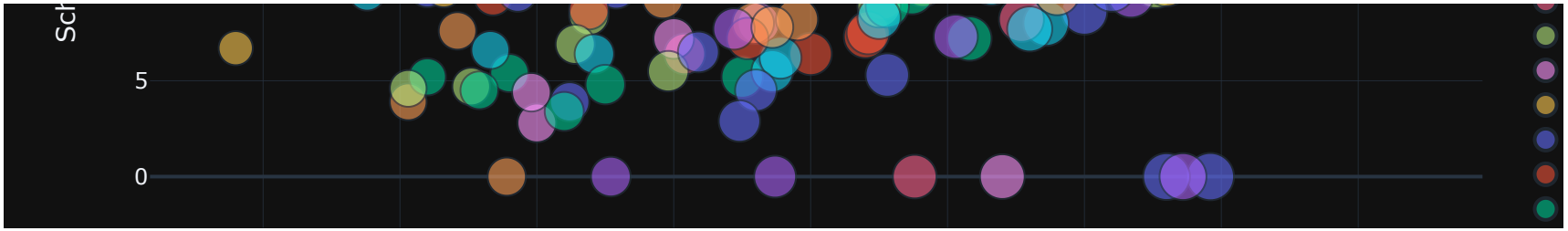

percentage_expenditure
6000
4000
2000
0


Life Expectancy vs Total_expenditure in every year
Total_expenditure
15
10
5
0
Cou


Life Expectancy vs under_five_deaths in every year
under_five_deaths
2500
2000
1500
1000
500
0
Cou


Life Expectancy vs BMI in every year
BMI
60
40
20
0
Cou


Life Expectancy vs Schooling in every year
20
15
10
Cou



Observation

- **Life Expectancy vs Adult Mortality over years:** We see the adult mortality rate is low for developed countries compared to developing countries. Due to the development of medical in developing countries, the adult mortality rate has been increased over the period of time. However, the life expectancy has been increased and adult mortality has been decreased over years across the countries

- **Life Expectancy vs Percentage Expenditure over years:** The medical spending contribution by developing countries is low resulting in low life expectancy compared to the spending by developed countries which results in high life expectancy. I could see the same pattern for all the years.

- **Life Expectancy vs Total Expenditure in every years::** Most of the countries the total expenditure lies below 10 and only limited countries are having expenditure greater than 10. Most of the values for life expectancy lies in the range of 40 and 90.United States is having high total expenditure and life expectancy. The life expectancy has been increased over years for all the countries

- **Life Expectancy vs under_five_deaths in every years::** Most of the countries are having the values less than 500. I also noticed that India is having high under 5 deaths. This might be due to population as India is the 2nd largest population country in the world. However, over the years, under five deaths has been reduced for India.

- **Life Expectance vs BMI in every years:** Over the years, the BMI across the countries has been increased which results in high life expectancy. This is because people are more cautious about their health now a days compared to older years.

- **Life Expectance vs Schooling in every years:** Schooling also plays a major role in improving life expectancy. The schooling gradually increases over years which results in high life expectancy.

Regression Plot

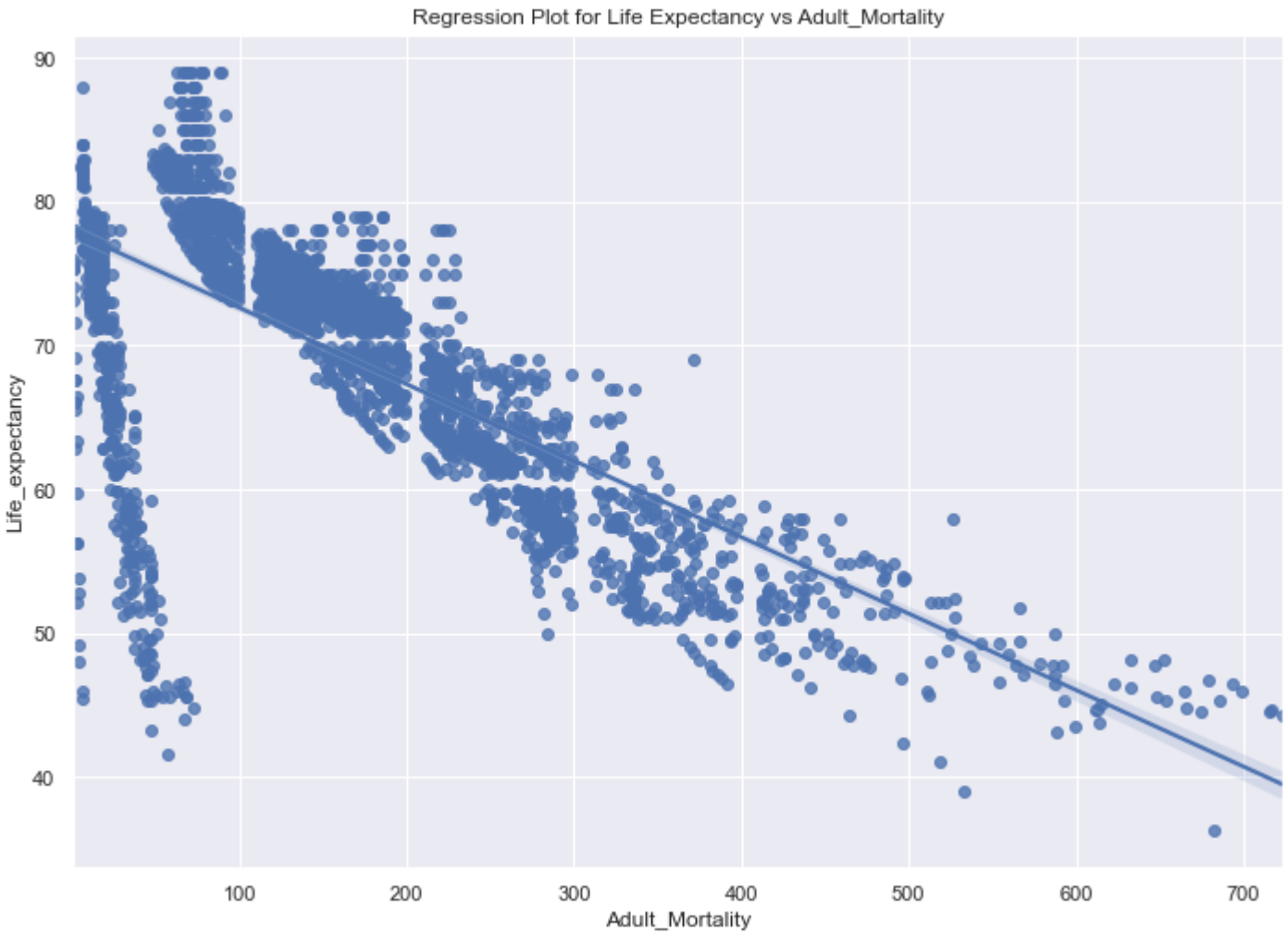
In [34]:
```
num_cols
```

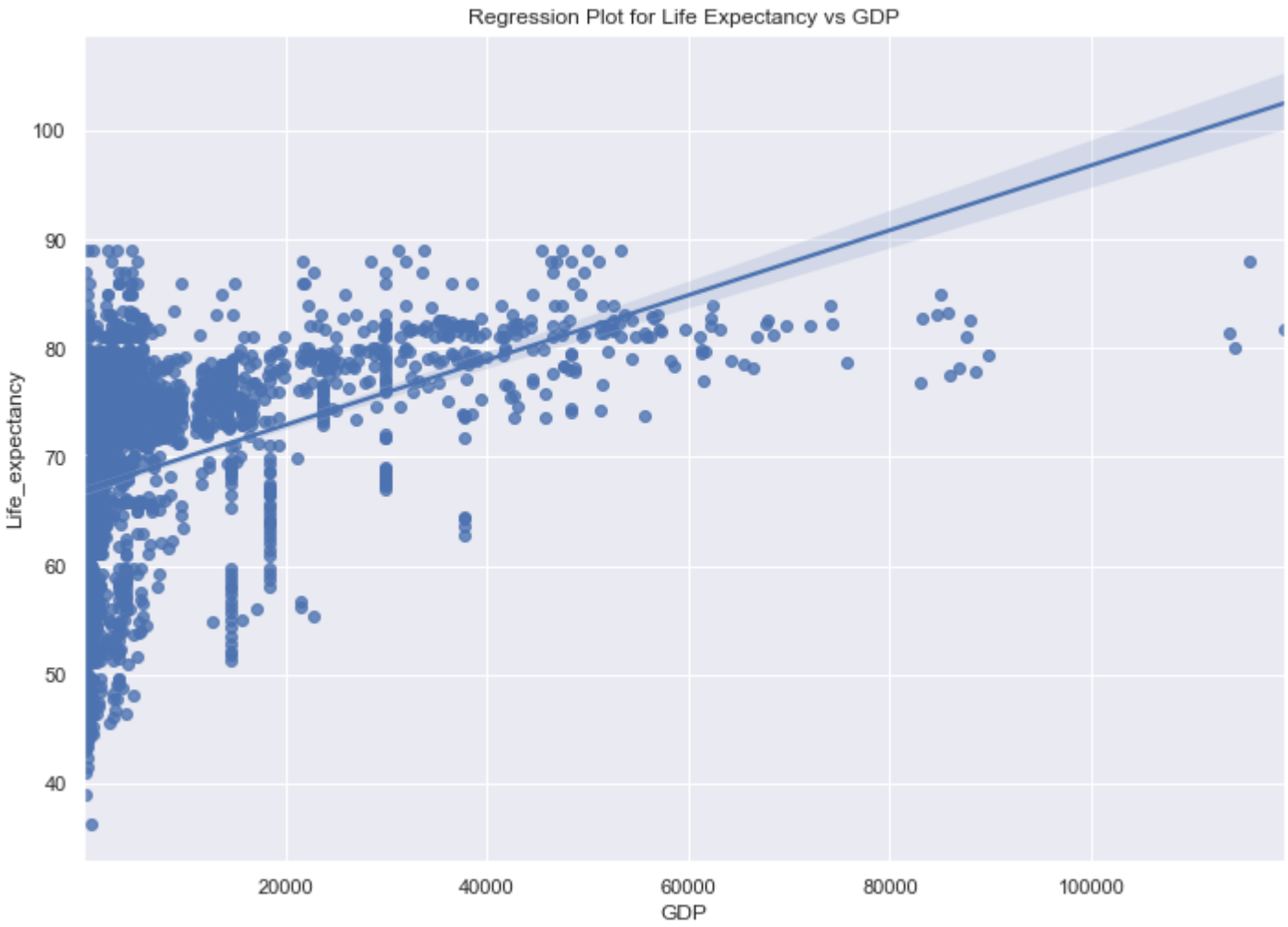
Out[34]:
```
['Year',
 'Life_expectancy',
 'Adult_Mortality',
 'infant_deaths',
 'Alcohol',
 'percentage_expenditure',
 'Hepatitis_B',
 'Measles',
 'BMI',
 'under_five_deaths',
 'Polio',
 'Total_expenditure',
 'Diphtheria',
 'HIV/AIDS',
 'GDP',
 'Population',
 'thinness_1_19_years',
 'thinness_5_9_years',
 'Income_composition_of_resources',
 'Schooling']
```

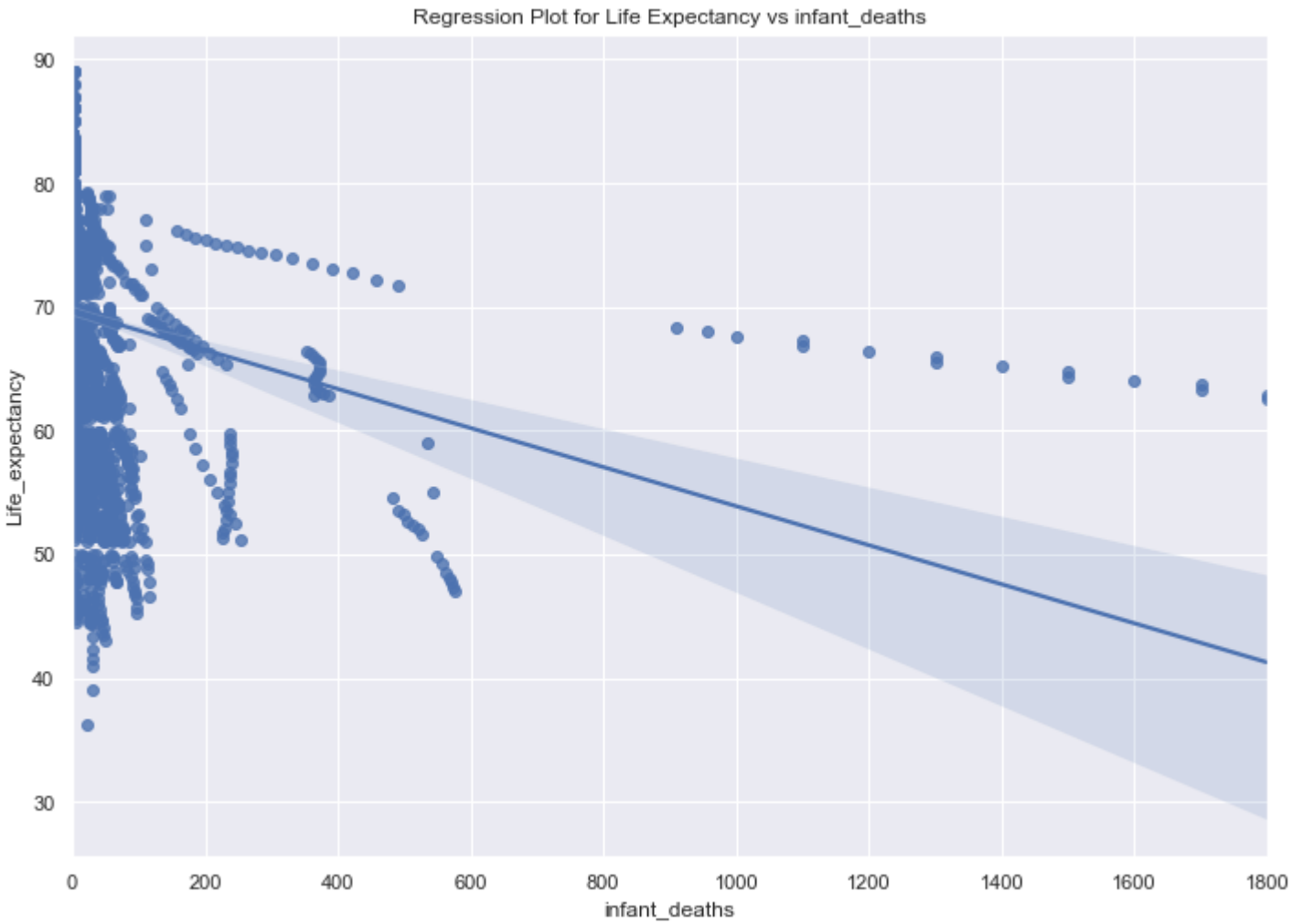
In [36]:
```
## Plotting regression plot

features = ['BMI','Income_composition_of_resources','Adult_Mortality','GDP',"infant_deaths"]
for i, feature in enumerate(features):
    plt.figure(i)
    title = "Regression Plot for Life Expectancy vs " + feature
    sns.regplot(x = feature, y = "Life_expectancy",data = life_exp_df).set(title = title)
```


Regression Plot for Life Expectancy vs BMI
Life_expectancy
BMI
90
80
70
60
50
40
10
20
30
40
50
60
70
80


Regression Plot for Life Expectancy vs Income_composition_of_resources
Life_expectancy
Income_composition_of_resources
90
80
70
60
50
40
0.0
0.2
0.4
0.6
0.8


Regression Plot for Life Expectancy vs Adult_Mortality
Life_expectancy
Adult_Mortality
90
80
70
60
50
40
100
200
300
400
500
600
700


Regression Plot for Life Expectancy vs GDP
Life_expectancy
100
90
80
70
60
50
40
20000
40000
60000
80000
100000
GDP



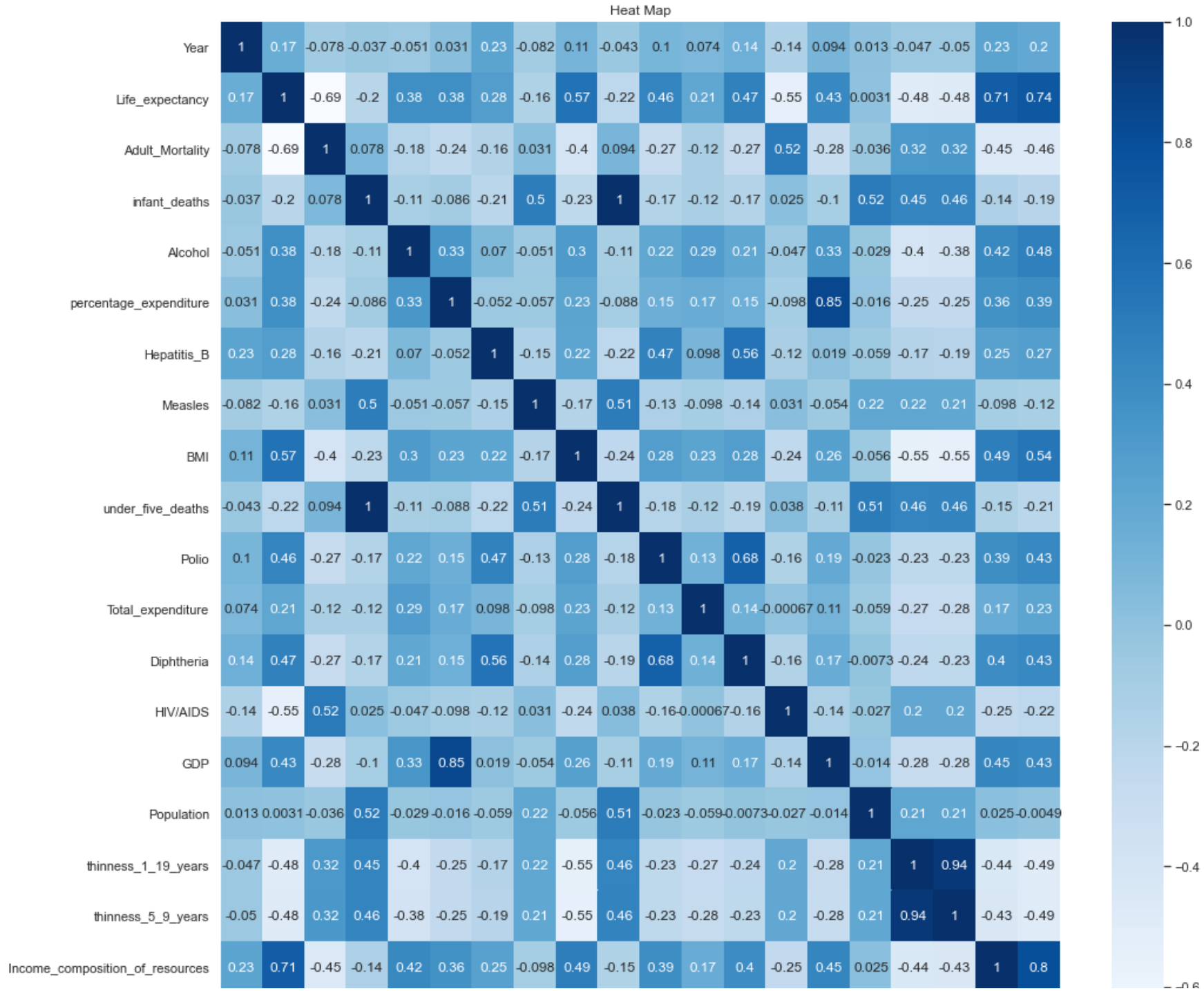
Observation

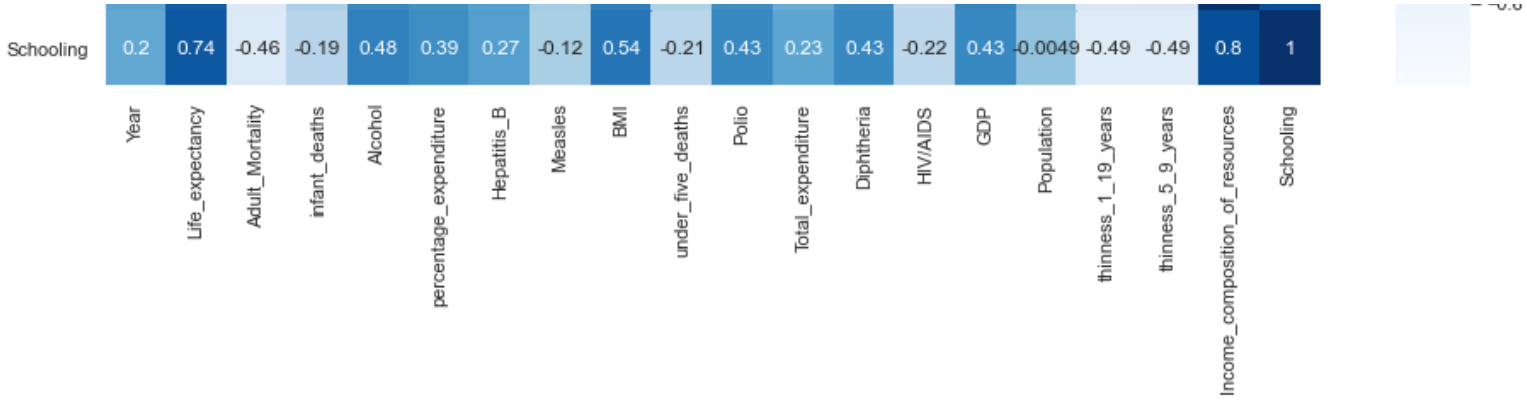
I could see life expectancy increase for increase in BMI, GDP and income composition of resource and decrease for increase in adult mortality and infant deaths. All these scenarios are expected.

Heat Map

In [38]:
```
## Plotting heat map
plt.figure(figsize = (16 ,16))
sns.heatmap(life_exp_df.corr() ,cmap = 'Blues' , cbar = True , annot = True).set_title("Heat Map")
```

Out[38]: Text(0.5, 1.0, 'Heat Map')





Observation

- Confirming our findings in the scatterplot above, Income composition resource and schooling features have strong correlation with our target variable Life Expectancy.
- Under five deaths is having strong correlation with measles; So, most of the deaths occurred due to measles; In addition, I also noticed that it is having positive correlation with population. This confirms my finding of scatter plot where India shows high under five deaths.
- Adult mortality is having positive correlation with HIV/AID; So, most of the deaths occurred due to HIV/AIDS for adult people whose age ranges between 15 and 60.
- GDP is having positive correlation with percentage expenditure; So, the countries who spend high amount for medical results in higher GDP.
- Income composition of resource and schooling are having strong correlation of 0.8; The people who earns more results in sending their children to school;
- Diphtheria and Polio are having positive correlation; This might be due to the fact that children are given with both the vaacines during their earlier age to protect them against infections caused by diphtheria, tetanus (lockjaw), pertussis (whooping cough), and poliovirus

Feature Engineering

Label Encoder

In [39]:
```
## Importing the LabelEncoder library
```

```
from sklearn.preprocessing import LabelEncoder
le = LabelEncoder()
```

In [40]:
```
#### Copying the dataframe to another dataframe
life_df = life_exp_df.copy(deep=True)
life_df = life_df.drop(['iso_alpha'],axis=1)
```

In [41]:
```
## Printing the info
life_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                  2938 non-null   float64
 4   Adult_Mortality                  2938 non-null   float64
 5   infant_deaths                    2938 non-null   int64  
 6   Alcohol                          2938 non-null   float64
 7   percentage_expenditure           2938 non-null   float64
 8   Hepatitis_B                      2938 non-null   float64
 9   Measles                          2938 non-null   int64  
 10  BMI                              2938 non-null   float64
 11  under_five_deaths                2938 non-null   int64  
 12  Polio                            2938 non-null   float64
 13  Total_expenditure                2938 non-null   float64
 14  Diphtheria                       2938 non-null   float64
 15  HIV/AIDS                         2938 non-null   float64
 16  GDP                              2938 non-null   float64
 17  Population                       2938 non-null   float64
 18  thinness_1_19_years              2938 non-null   float64
 19  thinness_5_9_years               2938 non-null   float64
 20  Income_composition_of_resources  2938 non-null   float64
 21  Schooling                        2938 non-null   float64
dtypes: float64(16), int64(4), object(2)
memory usage: 505.1+ KB
```

In [42]:
```
## Choosing categorical columns from the dataframe
cat_cols
```

```
['Country', 'Status']
```

Out[42]:

In [43]:
```
## Converting categorical variables into numerical using label encoder
for col in cat_cols:
    life_df[col] = le.fit_transform(life_df[col])
```

In [44]:
```
## Printing the info of the table
life_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   int32
 1   Year                             2938 non-null   int64
 2   Status                           2938 non-null   int32
 3   Life_expectancy                  2938 non-null   float64
 4   Adult_Mortality                  2938 non-null   float64
 5   infant_deaths                    2938 non-null   int64
 6   Alcohol                          2938 non-null   float64
 7   percentage_expenditure           2938 non-null   float64
 8   Hepatitis_B                      2938 non-null   float64
 9   Measles                          2938 non-null   int64
 10  BMI                              2938 non-null   float64
 11  under_five_deaths                2938 non-null   int64
 12  Polio                            2938 non-null   float64
 13  Total_expenditure                2938 non-null   float64
 14  Diphtheria                       2938 non-null   float64
 15  HIV/AIDS                         2938 non-null   float64
 16  GDP                              2938 non-null   float64
 17  Population                       2938 non-null   float64
 18  thinness_1_19_years              2938 non-null   float64
 19  thinness_5_9_years               2938 non-null   float64
 20  Income_composition_of_resources  2938 non-null   float64
 21  Schooling                        2938 non-null   float64
dtypes: float64(16), int32(2), int64(4)
memory usage: 482.1 KB
```

In [45]:
```
## Printing few records from the dataframe using head
life_df.head()
```

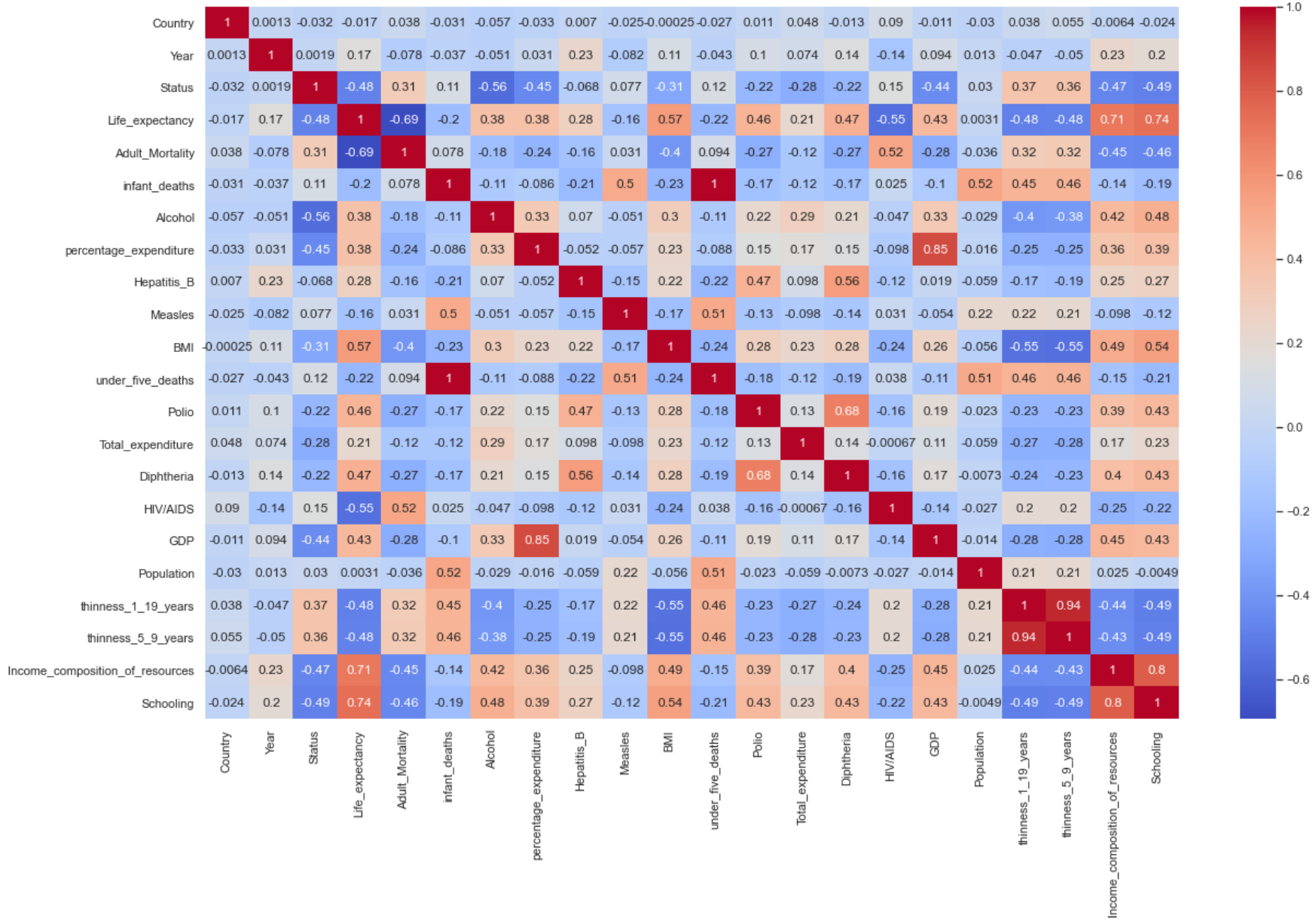
Out[45]:

	Country	Year	Status	Life_expectancy	Adult_Mortality	infant_deaths	Alcohol	percentage_expenditure	Hepatitis_B	Measles	BMI	under_f

	Country	Year	Status	Life_expectancy	Adult_Mortality	infant_deaths	Alcohol	percentage_expenditure	Hepatitis_B	Measles	BMI	under_f
0	0	2015	1	65.0	263.0	62	0.01	71.279624	65.0	1154	19.1	
1	0	2014	1	59.9	271.0	64	0.01	73.523582	62.0	492	18.6	
2	0	2013	1	59.9	268.0	66	0.01	73.219243	64.0	430	18.1	
3	0	2012	1	59.5	272.0	69	0.01	78.184215	67.0	2787	17.6	
4	0	2011	1	59.2	275.0	71	0.01	7.097109	68.0	3013	17.2	

In [46]:
```
## Correlation matrix
corrmat = life_df.corr()
plt.figure(figsize=(20,12))
sns.heatmap(corrmat, annot=True, cmap='coolwarm')
```

Out[46]: <AxesSubplot:>



In [47]:
```
## Printing the Correlation for income field
corrmat['Life_expectancy'].sort_values(ascending = False)
```

Out[47]:
```
Life_expectancy                  1.000000
Schooling                        0.736323
```

```
Income_composition_of_resources    0.705641
BMI                                0.567361
Diphtheria                         0.470947
Polio                              0.456336
GDP                                0.429252
percentage_expenditure             0.381847
Alcohol                            0.380721
Hepatitis_B                        0.284345
Total_expenditure                  0.212256
Year                               0.165457
Population                         0.003051
Country                           -0.017249
Measles                           -0.156446
infant_deaths                     -0.195148
under_five_deaths                 -0.221021
thinness_5_9_years                -0.480438
Status                            -0.481907
thinness_1_19_years               -0.484921
HIV/AIDS                          -0.553770
Adult_Mortality                   -0.694702
Name: Life_expectancy, dtype: float64
```

Observation

1. The target variable "Life Expectancy" is positively correlated with "Schooling", "Income_composition_of_resources", "BMI", "Diphtheria", "Polio", "GDP", "percentage_expenditure", "Alcohol", "Hepatitis_B","Total_expenditure"

2. The target variable is negatively correlated with "Adult Mortality", "HIV/AIDS", "thinness_1_19_years", "Status", "under_five_deaths", "thinness_5_9_years", "infant_deaths"

3. The above correlation is completely make sense as Life Expectancy would be higher if people are educated, took vaccines like Polio, Diphtheria, Hepatits B, Percentage_expenditure spent by the country, Total_expenditure and GDP of the countries. Negative correlation also makes sense for below reasons.

    - HIV/AIDs cause people to die reducing the life expectancy
    - Adult Mortality is high
    - Infant deaths and under_five_deaths are high
    - Thinness_1_19_years and thinness_5_9_years are high

Modeling

Linear Regression without Normalization

Running Linear Regression

In [48]:
```
## Creating source and target
X = life_df.drop(['Life_expectancy'],axis=1)
y = life_df['Life_expectancy']
```

In [49]:
```
## Verifying the shape of X and y
print("The shape of the input: {}".format(X.shape))
print("The shape of the target: {}".format(y.shape))
```

```
The shape of the input: (2938, 21)
The shape of the target: (2938,)
```

In [50]:
```
## Splitting the dataframe in train and test
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.2, random_state = 42)
```

In [51]:
```
## Priting values from X_Train dataframe
X_train.head()
```

Out[51]:

	Country	Year	Status	Adult_Mortality	infant_deaths	Alcohol	percentage_expenditure	Hepatitis_B	Measles	BMI	under_five_deaths	P
456	27	2007	1	126.0	0	5.28	345.463714	96.0	0	25.5	0	
462	27	2001	1	152.0	0	3.81	150.743486	4.0	0	22.1	0	
2172	143	2011	1	143.0	0	10.43	0.000000	99.0	0	44.5	0	
2667	175	2013	1	13.0	3	1.29	594.645310	98.0	16	59.3	3	
381	23	2002	1	95.0	0	0.13	941.703687	99.0	0	28.0	0	

In [52]:
```
## Printing the shape of train and test dataset
print("The shape of training dataset: {}".format(X_train.shape))
print("The shape of test dataset: {}".format(X_test.shape))
```

```
The shape of training dataset: (2350, 21)
The shape of test dataset: (588, 21)
```

The dataset has been successfully splitted into the test and train sets respectively. The division is purely random and each time we run the code, a new training set and testing set will be created. Let's run the resgressor algorithm on this training set and then see it's accuracy

using the testing set!

Without Normalization

Linear Regression

In [53]:
```
## Linear Regression
model = LinearRegression()
model.fit(X_train, y_train)
```

Out[53]:
```
LinearRegression
LinearRegression()
```

In [54]:
```
## Printing Model Coefficients
print(model.coef_)
```

```
[ 3.68941251e-03 -1.19153627e-02 -1.54524383e+00 -1.97576342e-02
  9.73673509e-02  5.28558943e-02  1.89005818e-04 -1.66228776e-03
 -2.88986381e-05  3.80657199e-02 -7.27719326e-02  1.85332399e-02
  3.51793952e-02  3.19527843e-02 -4.73278416e-01  1.81808442e-05
  5.36201661e-10 -8.75784723e-02  3.26826255e-02  6.78062182e+00
  6.80990044e-01]
```

In [55]:
```
# Test the model by predicting on test data
test_pred = model.predict(X_test)
```

In [56]:
```
## Printing predictions
test_pred
```

Out[56]:
```
array([68.18315703, 76.89518468, 74.280269  , 77.35089904, 47.66390998,
       50.52995545, 68.14005762, 71.0563889 , 74.27246486, 55.13253305,
       52.22849446, 55.83448179, 64.77663372, 70.78199156, 71.03683742,
       62.9824121 , 55.9986225 , 80.99609461, 68.56983552, 82.00624411,
       81.7366845 , 84.78887226, 70.63506124, 68.84332359, 70.02389083,
       68.35864859, 61.25291712, 61.31165896, 78.99102843, 71.47367725,
       66.73708237, 75.07663319, 67.53926013, 68.47122441, 80.65107331,
       57.01907039, 67.01946088, 73.16192168, 62.56481729, 76.1848382 ,
       58.50121948, 64.40406096, 75.17653929, 81.56924868, 76.27390454,
       57.6987577 , 79.12514085, 70.69602804, 65.53980413, 67.64117515,
       55.11283953, 62.34650395, 78.14047409, 74.48812476, 73.21882996,
       59.82541811, 76.00042014, 70.44812926, 67.81883917, 69.20341096,
```

```
78.35125255, 56.31378557, 73.92926024, 69.56050139, 53.28129012,
52.82351537, 53.28150337, 79.73369548, 58.58233026, 78.33268251,
79.83694717, 75.08458446, 73.76356576, 60.18101713, 56.16899261,
72.60831234, 64.3584943 , 76.82812926, 57.57267405, 62.69774929,
64.2315095 , 61.62487092, 67.22925642, 67.19396902, 60.09584557,
76.44998325, 75.37324374, 68.28093985, 76.27257521, 76.06418742,
73.25990639, 62.99668287, 72.75981123, 77.26143958, 73.99295611,
68.29489747, 79.55148962, 71.17460545, 66.79233397, 61.91004071,
83.97671199, 55.55093779, 71.00110171, 54.2213336 , 74.95376473,
77.04715327, 70.77371195, 65.27548926, 65.40194434, 64.74001889,
68.71294215, 62.31473092, 50.46963062, 69.99321414, 63.08271449,
76.64853692, 64.01093021, 57.7774725 , 66.5026882 , 64.30346045,
65.98855605, 71.95617033, 80.45373177, 62.63465546, 57.87838101,
65.84776614, 70.76806469, 64.84398496, 73.03145518, 71.72518094,
74.65077961, 80.51335666, 54.9050288 , 68.9020664 , 54.75621756,
69.75838095, 75.24731771, 76.99509481, 75.76497953, 76.23088166,
69.19286895, 66.14405052, 61.98955242, 59.24238781, 50.91568298,
63.97776661, 54.74275943, 74.88519534, 63.69948566, 75.68156808,
79.64121255, 79.51450987, 70.11456283, 72.26937138, 81.14770365,
78.2194586 , 80.35194335, 73.45845266, 80.47613746, 78.62885997,
72.40984321, 78.71538272, 69.99409775, 78.35286116, 68.37443259,
63.68623255, 65.17452502, 80.83164224, 77.05255698, 74.78649833,
81.70996918, 72.0826984 , 66.54651064, 61.3959706 , 71.12273503,
64.0454614 , 69.65360074, 64.06427953, 60.38913968, 72.71271008,
83.38038804, 68.49937833, 82.63562535, 73.89388796, 78.87074728,
35.83775662, 60.34051838, 43.65627741, 72.43837078, 72.98048961,
64.75785747, 68.05513626, 81.21654357, 62.75537209, 76.49227556,
67.56286735, 72.86145102, 67.81158265, 79.29941102, 72.85741404,
73.41924466, 56.80655315, 73.22135506, 71.11905606, 67.54746662,
50.81313331, 75.06163448, 67.99470253, 71.09820172, 82.59054171,
59.77841607, 53.44642127, 72.75849886, 48.47966039, 55.36776816,
57.84201733, 63.17124742, 73.43505547, 61.80415693, 67.73300237,
78.22655712, 74.03339249, 72.60419686, 79.90028199, 77.12988219,
50.50059458, 57.70353305, 74.37186691, 53.97048076, 62.96918882,
69.59433395, 72.55761631, 60.64758214, 73.53827449, 72.90289942,
54.64904604, 77.31423927, 78.08028186, 49.12925518, 72.05704766,
75.88386191, 69.22764893, 73.21567264, 75.29720769, 63.91162901,
52.73567682, 80.85120763, 76.15387972, 76.11842586, 63.47527833,
70.39327904, 75.80569484, 58.72667101, 78.79013934, 69.94495881,
71.77415661, 62.01500234, 79.03370762, 58.72791127, 48.0818151 ,
68.70526549, 71.09529732, 60.36935493, 56.30062453, 67.88783012,
78.95235102, 68.76562265, 81.46377906, 58.90946014, 57.63566207,
55.06756278, 50.90654312, 70.36759795, 71.58842829, 54.8915278 ,
72.70684658, 58.5771381 , 73.72432503, 64.42631256, 63.84875685,
69.29930152, 80.07246292, 59.83190716, 76.24839373, 75.78460097,
64.48328085, 78.41436242, 46.64143888, 61.8736919 , 76.73741673,
66.31628691, 69.56623794, 67.93624751, 73.95103338, 74.87157628,
63.36304003, 78.48928719, 71.52295555, 54.43953947, 72.43739543,
73.92342211, 76.13497959, 70.37615785, 74.48749636, 74.76667348,
```

```
56.94931909, 55.26289203, 50.72373938, 69.23632754, 75.25486193,
72.9342708 , 61.24726756, 75.21824987, 64.34007899, 71.29052093,
66.31053815, 54.70153506, 71.08719456, 55.38524839, 74.00735589,
48.36079384, 78.26493797, 55.64638711, 70.68721584, 69.97524316,
76.6370845 , 76.17717423, 71.12735934, 72.73938452, 81.79795195,
73.13044469, 70.5391574 , 68.3199902 , 69.65974358, 74.51082218,
81.47164566, 72.49339699, 76.43492963, 68.28050524, 70.49129805,
70.24172155, 74.56742301, 71.10270328, 74.54235293, 70.63413048,
50.71090272, 65.60042847, 73.28133306, 73.95519214, 74.67462672,
63.13845805, 69.51441841, 59.01369379, 59.35274486, 72.74775042,
79.80718505, 72.24779774, 54.74349407, 71.04841466, 73.49138709,
63.24666579, 73.02857093, 76.03139626, 68.019068  , 65.91056326,
63.63119006, 60.50935833, 75.26978974, 63.4417706 , 74.85308561,
72.29434105, 55.80895016, 72.78065311, 76.61919794, 74.42318895,
72.53536409, 57.69144456, 70.6547112 , 71.5859003 , 65.61424711,
52.86030864, 81.49964466, 80.72606616, 80.59587298, 62.8230139 ,
55.39613154, 73.50428614, 56.9543151 , 72.86202364, 76.72540676,
49.26167452, 79.21908178, 77.29223871, 80.25435632, 65.13945973,
66.23988261, 79.40744694, 70.82312744, 79.3162279 , 72.22470136,
53.39403995, 76.15116728, 50.68986398, 72.42792656, 65.73363923,
64.17681928, 68.38897532, 73.07447972, 67.82793703, 73.27953218,
72.09146553, 81.64691075, 74.76611093, 75.07149158, 72.61955368,
69.43553182, 60.69847433, 71.38916167, 69.64606947, 65.88643311,
64.40952173, 61.09835735, 70.33715795, 72.86355273, 73.78616298,
71.52791953, 64.87120968, 64.17445076, 73.26299464, 74.6182183 ,
55.1465486 , 75.8725845 , 72.44117942, 66.90755916, 35.19618446,
64.92108913, 57.04468267, 67.86701331, 67.75439025, 73.63266466,
75.19474706, 69.81663624, 71.05947056, 78.42359151, 77.3231296 ,
70.39399211, 76.66280043, 78.99194888, 83.30216122, 58.19992575,
82.23300453, 55.82187431, 70.10579065, 72.70255973, 52.49801934,
76.66146198, 83.74578042, 52.0758612 , 66.99945365, 76.64660028,
73.40967069, 62.08342535, 62.2253972 , 66.86177417, 65.18331493,
57.00082353, 79.2893267 , 71.55564025, 79.89825403, 70.15430219,
68.93652326, 76.53290299, 73.90538768, 71.87507097, 83.16632979,
71.19522928, 67.00628856, 69.77036124, 81.07519701, 73.11600616,
75.55556366, 70.70516814, 70.86650508, 75.99595535, 69.76744827,
82.29316329, 77.40637093, 80.7075494 , 71.58268411, 74.21981689,
74.00573797, 74.18003365, 49.33972779, 70.24383203, 74.7276377 ,
62.98359905, 80.45667697, 72.08938204, 61.36381291, 72.2773492 ,
73.97057755, 57.8646446 , 80.42653541, 70.44391192, 77.02855335,
78.08413104, 68.14241868, 81.99824688, 74.89652912, 69.1961421 ,
65.14881303, 67.26831813, 69.68047169, 73.16980982, 48.42471543,
65.26836194, 62.9975053 , 63.03372632, 65.70313705, 59.3391915 ,
73.57713449, 77.94242001, 72.23283622, 73.10932327, 66.61121715,
63.55447104, 66.43314682, 71.3424798 , 76.32912388, 82.16266842,
57.19273426, 71.53704409, 69.72718771, 78.20129378, 67.08888317,
79.46184034, 69.35242306, 78.38879332, 82.36334282, 82.55371239,
78.92656962, 48.94883025, 64.55607407, 67.63894279, 75.62596757,
64.10001022, 70.6396418 , 61.70752542, 64.66763308, 69.49353629,
```

```
77.30531984, 56.93547348, 78.7127641 , 59.9289644 , 74.72539753,
67.99864654, 78.44637609, 67.92795409, 71.05571371, 78.36536618,
65.30212575, 82.11414553, 85.8024153 , 75.69905437, 51.32675014,
75.77366915, 77.2606161 , 84.89039957, 59.09132128, 72.43946911,
73.89870353, 63.11656452, 78.49501806, 69.01197266, 74.23229624,
65.03557662, 69.26837089, 71.18818185, 57.94246006, 76.5391497 ,
54.40596645, 78.40419877, 81.50746224, 58.61375014, 76.1711045 ,
63.71266264, 70.76338317, 73.93671081])
```

In [57]:
```
print("Root Mean Square Error: {}".format(mean_squared_error(y_test, test_pred)))
print("R2 Score: {}".format(r2_score(y_test, test_pred)*100))
```

```
Root Mean Square Error: 14.263110163857739
R2 Score: 83.54823400752926
```

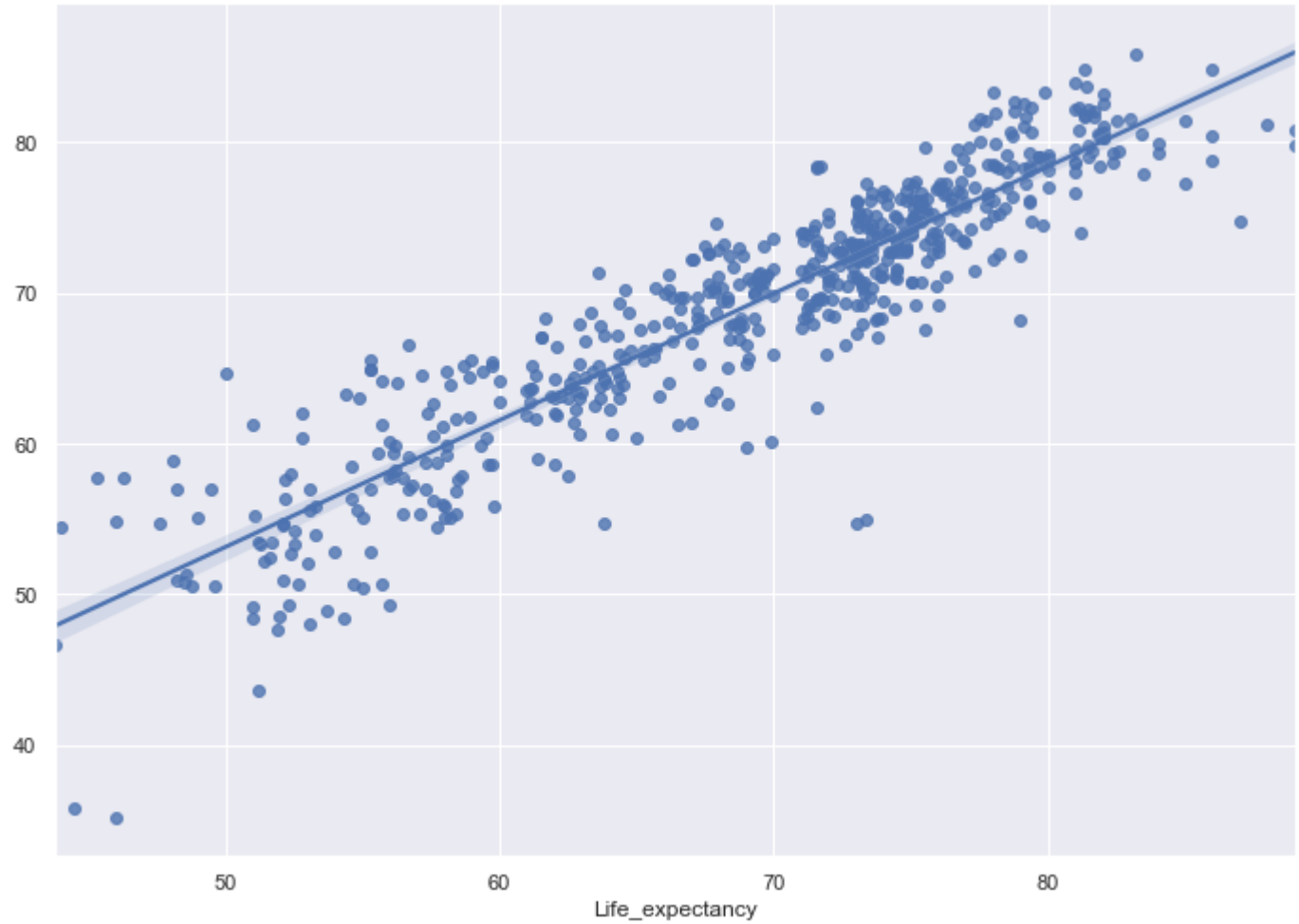
- R-squared value < 0.3 this value is generally considered a None or Very weak effect size,
- R-squared value 0.3 < r < 0.5 this value is generally considered a weak or low effect size,
- R-squared value 0.5 < r < 0.7 this value is generally considered a Moderate effect size,
- R-squared value r > 0.7 this value is generally considered strong effect size,

So, according to our current model, the accuracy that we have got is 83.548%, which is a pretty good score!

Plotting Actual vs Predicted

In [58]:
```
### Regression Plotting the outcome of the model
sns.regplot(x = y_test, y = model.predict(X_test))
```

Out[58]:
```
<AxesSubplot:xlabel='Life_expectancy'>
```



Linear Regression with MinMaxScaler Normalization

In [59]:
```
## Importing the library
from sklearn.preprocessing import MinMaxScaler
scaler = MinMaxScaler()
```

In [60]:
```
## Normalizing the dataframe
life_df_norm = pd.DataFrame(scaler.fit_transform(life_df), columns=life_df.columns)
life_df_norm.head()
```

Out[60]:

	Country	Year	Status	Life_expectancy	Adult_Mortality	infant_deaths	Alcohol	percentage_expenditure	Hepatitis_B	Measles	BMI
0	0.0	1.000000	1.0	0.544592	0.362881	0.034444	0.0	0.003659	0.653061	0.005439	0.209733
1	0.0	0.933333	1.0	0.447818	0.373961	0.035556	0.0	0.003774	0.622449	0.002319	0.203940
2	0.0	0.866667	1.0	0.447818	0.369806	0.036667	0.0	0.003759	0.642857	0.002027	0.198146
3	0.0	0.800000	1.0	0.440228	0.375346	0.038333	0.0	0.004014	0.673469	0.013135	0.192352
4	0.0	0.733333	1.0	0.434535	0.379501	0.039444	0.0	0.000364	0.683673	0.014200	0.187717

In [61]:
```
## Defining source and target
X_norm = life_df_norm.drop(['Life_expectancy'], axis =1 )
y_norm = life_df_norm['Life_expectancy']
```

In [62]:
```
## Printing the shape of source and target
print("The shape of source: {}".format(X_norm.shape))
print("The shape of target: {}".format(y_norm.shape))
```

```
The shape of source: (2938, 21)
The shape of target: (2938,)
```

In [63]:
```
## Splitting the dataframe in train and test
X_norm_train, X_norm_test, y_norm_train, y_norm_test = train_test_split(X_norm, y_norm, test_size = 0.2, random_state = 4
```

In [64]:
```
## Priting values from X_Train dataframe
X_norm_train.head()
```

Out[64]:

	Country	Year	Status	Adult_Mortality	infant_deaths	Alcohol	percentage_expenditure	Hepatitis_B	Measles	BMI	under_five_
456	0.140625	0.466667	1.0	0.173130	0.000000	0.295073	0.017734	0.969388	0.000000	0.283893	
462	0.140625	0.066667	1.0	0.209141	0.000000	0.212766	0.007738	0.030612	0.000000	0.244496	
2172	0.744792	0.733333	1.0	0.196676	0.000000	0.583427	0.000000	1.000000	0.000000	0.504056	
2667	0.911458	0.866667	1.0	0.016620	0.001667	0.071669	0.030526	0.989796	0.000075	0.675550	
381	0.119792	0.133333	1.0	0.130194	0.000000	0.006719	0.048342	1.000000	0.000000	0.312862	

In [65]:
```
## Printing the shape of train and test dataset
print("The shape of training dataset: {}".format(X_norm_train.shape))
print("The shape of test dataset: {}".format(X_norm_test.shape))
```

```
The shape of training dataset: (2350, 21)
The shape of test dataset: (588, 21)
```

MinMaxScaler has been applied on the dataset, and dataset has been successfully splitted into the test and train sets respectively. The division is purely random and each time we run the code, a new training set and testing set will be created. Let's run the resgressor algorithm on this training set and then see it's accuracy using the testing set!

Linear Regression with Normalization

In [66]:
```
## Linear Regression
model_norm = LinearRegression()
model_norm.fit(X_norm_train, y_norm_train)
```

Out[66]:
```
▾ LinearRegression
LinearRegression()
```

In [67]:
```
## Printing Model Coefficients
print(model_norm.coef_)
```

```
[ 1.34415029e-02 -3.39146946e-03 -2.93215148e-02 -2.70683338e-01
  3.32564007e+00  1.79128325e-02  6.98636933e-02 -3.09116130e-03
 -1.16352936e-01  6.23353250e-02 -3.45217897e+00  3.37607407e-02
  1.15017264e-02  5.88125252e-02 -4.53521062e-01  4.11125328e-02
  1.31645064e-02 -4.58665244e-02  1.76746647e-02  1.21973994e-01
  2.67485653e-01]
```

In [68]:
```
# Test the model by predicting on test data
test_norm_pred = model_norm.predict(X_norm_test)
```

In [69]:
```
## Printing predictions
test_norm_pred
```

Out[69]:
```
array([ 0.60499349,  0.77030711,  0.72068822,  0.77895444,  0.21563397,
```

```
 0.27001813,  0.60417567,  0.65951402,  0.72054013,  0.35735357,
 0.30224847,  0.37067328,  0.54035358,  0.65430724,  0.65914303,
 0.50630763,  0.3737879 ,  0.84812324,  0.61233084,  0.86729116,
 0.86217618,  0.92009245,  0.65151919,  0.61752037,  0.63992203,
 0.6083235 ,  0.47348989,  0.47460453,  0.81007644,  0.66743221,
 0.57755375,  0.73579949,  0.59277533,  0.61045967,  0.84157634,
 0.39315124,  0.58291197,  0.6994672 ,  0.49838363,  0.75682805,
 0.42127551,  0.53328389,  0.73769524,  0.85899903,  0.75851811,
 0.40604853,  0.81262127,  0.65267605,  0.55483499,  0.59470921,
 0.35697988,  0.49424106,  0.79393689,  0.72463235,  0.70054706,
 0.44640262,  0.75332866,  0.64797209,  0.59808044,  0.62435315,
 0.79793648,  0.37976823,  0.71402771,  0.63112906,  0.32222562,
 0.31353919,  0.32222967,  0.82416879,  0.42281462,  0.79758411,
 0.82612803,  0.73595037,  0.7108836 ,  0.45315023,  0.37702073,
 0.68896228,  0.53241925,  0.76903471,  0.40365605,  0.50090606,
 0.53000967,  0.48054784,  0.58689291,  0.58622332,  0.45153407,
 0.76185926,  0.74142778,  0.60684895,  0.75849289,  0.75453866,
 0.7013265 ,  0.50657842,  0.69183703,  0.77725692,  0.71523636,
 0.6071138 ,  0.82071138,  0.66175722,  0.57860216,  0.48595903,
 0.90468144,  0.36529294,  0.65846493,  0.34006326,  0.73346802,
 0.77319076,  0.65415013,  0.54981953,  0.55221906,  0.5396588 ,
 0.61504634,  0.49363816,  0.26887345,  0.63933993,  0.5082109 ,
 0.76562689,  0.5258241 ,  0.40754217,  0.57310604,  0.53137496,
 0.56335021,  0.67658767,  0.83783172,  0.49970883,  0.40945695,
 0.56067867,  0.65404297,  0.54163159,  0.69699156,  0.67220457,
 0.72771878,  0.83896312,  0.3530366 ,  0.61863504,  0.35021286,
 0.63488389,  0.73903829,  0.77220294,  0.74886109,  0.75770174,
 0.62415311,  0.56630077,  0.48746779,  0.43533943,  0.27733744,
 0.52519481,  0.34995748,  0.73216689,  0.51991434,  0.74727833,
 0.8224139 ,  0.82000967,  0.64164256,  0.68253077,  0.85100007,
 0.79543565,  0.83590025,  0.70509398,  0.83825688,  0.80320417,
 0.68519627,  0.80484597,  0.63935669,  0.797967  ,  0.60862301,
 0.51966286,  0.5479037 ,  0.8450027 ,  0.7732933 ,  0.73029409,
 0.86166924,  0.67898858,  0.57393758,  0.47620438,  0.66077296,
 0.52647934,  0.63289565,  0.52683642,  0.45709942,  0.69094327,
 0.893366  ,  0.6109939 ,  0.87923388,  0.71335651,  0.80779407,
-0.00877122,  0.45617682,  0.13958781,  0.68573759,  0.69602447,
 0.5399973 ,  0.60256426,  0.85230633,  0.50199947,  0.76266178,
 0.59322329,  0.69376567,  0.59794274,  0.8159281 ,  0.69368907,
 0.70434999,  0.38911866,  0.70059497,  0.66070315,  0.59293106,
 0.27539152,  0.73551489,  0.60141751,  0.66030743,  0.8783784 ,
 0.44551074,  0.32535904,  0.69181212,  0.2311131 ,  0.36181723,
 0.40876693,  0.50989084,  0.70465001,  0.48394985,  0.59645166,
 0.79557034,  0.71600365,  0.68888419,  0.82732983,  0.77476057,
 0.269461  ,  0.40613915,  0.72242632,  0.33530324,  0.50605671,
 0.63177104,  0.68800031,  0.46200346,  0.70660862,  0.69455217,
 0.34817924,  0.77825881,  0.79279472,  0.24343938,  0.67850185,
 0.75111692,  0.62481307,  0.70048715,  0.73998497,  0.52393983,
 0.31187243,  0.84537396,  0.7562406 ,  0.75556785,  0.51565993,
```

```
0.64693129,  0.74963368,  0.42555353,  0.8062645 ,  0.63842427,
0.6731339 ,  0.48795071,  0.81088629,  0.42557706,  0.22356385,
0.61490067,  0.66025232,  0.456724  ,  0.37951849,  0.59938957,
0.80934252,  0.61604597,  0.85699771,  0.42902201,  0.40485127,
0.35612074,  0.27716401,  0.64644398,  0.66960964,  0.35278041,
0.690832  ,  0.42271609,  0.71013899,  0.53370612,  0.52274681,
0.6261727 ,  0.83059702,  0.44652575,  0.75803404,  0.74923341,
0.53478711,  0.79913401,  0.19623224,  0.4852693 ,  0.76731341,
0.56956901,  0.63123791,  0.6003083 ,  0.71444086,  0.73190847,
0.51353017,  0.80055573,  0.66836728,  0.34420378,  0.68571908,
0.71391693,  0.75588197,  0.64660641,  0.72462042,  0.7299179 ,
0.39182769,  0.35982717,  0.27369524,  0.62497775,  0.73918144,
0.69514745,  0.47338269,  0.73848671,  0.53206981,  0.66395675,
0.56945993,  0.34917524,  0.66009857,  0.36214893,  0.7155096 ,
0.22885757,  0.79629863,  0.36710412,  0.65250884,  0.63899892,
0.76540957,  0.75668262,  0.66086071,  0.69144942,  0.86333875,
0.69886992,  0.64969938,  0.60758995,  0.63301221,  0.72506304,
0.85714698,  0.68678173,  0.76157362,  0.60684071,  0.64879123,
0.64405544,  0.72613706,  0.66039285,  0.72566135,  0.65150153,
0.27345166,  0.55598536,  0.70173308,  0.71451977,  0.72817129,
0.50926865,  0.63025462,  0.43099988,  0.43743349,  0.69160817,
0.82556328,  0.6821214 ,  0.34997142,  0.65936271,  0.70571892,
0.51132193,  0.69693683,  0.75391644,  0.60187985,  0.56187027,
0.51861841,  0.45938061,  0.7394647 ,  0.51502411,  0.7315576 ,
0.68300457,  0.37018881,  0.69223251,  0.76507017,  0.72340017,
0.68757807,  0.40590976,  0.65189205,  0.66956168,  0.55624757,
0.31423736,  0.85767827,  0.84299936,  0.8405289 ,  0.503283  ,
0.36235544,  0.70596368,  0.39192249,  0.69377654,  0.76708552,
0.24595208,  0.81440383,  0.77784134,  0.83404851,  0.54723833,
0.56811921,  0.81797812,  0.65508781,  0.81624721,  0.68168314,
0.32436508,  0.75618913,  0.27305245,  0.6855394 ,  0.55851308,
0.5289719 ,  0.60889896,  0.69780796,  0.59825307,  0.7016989 ,
0.67915494,  0.86047269,  0.72990723,  0.73570193,  0.68917559,
0.62875772,  0.46296915,  0.66582849,  0.63275274,  0.56141239,
0.53338751,  0.47055707,  0.64586637,  0.69380555,  0.71131239,
0.66846147,  0.54214819,  0.52892696,  0.7013851 ,  0.72710092,
0.35761952,  0.75090293,  0.68579088,  0.5807886 , -0.02094527,
0.54309467,  0.39363724,  0.59899456,  0.5968575 ,  0.70839971,
0.73804074,  0.6359893 ,  0.6595725 ,  0.79930914,  0.77842751,
0.64694482,  0.76589754,  0.81009391,  0.89188162,  0.41555836,
0.87159401,  0.37043405,  0.6414761 ,  0.69075066,  0.3073628 ,
0.76587214,  0.90029944,  0.2993522 ,  0.58253233,  0.76559014,
0.70416832,  0.48924906,  0.49194302,  0.57991981,  0.54807049,
0.392805  ,  0.81573675,  0.66898748,  0.82729135,  0.64239663,
0.61928887,  0.76343269,  0.71357472,  0.67504879,  0.88930417,
0.66214856,  0.58266202,  0.63511122,  0.84962423,  0.69859594,
0.74488736,  0.65284949,  0.65591091,  0.75324393,  0.63505594,
0.87273555,  0.78000704,  0.842648  ,  0.66950065,  0.71954112,
0.7154789 ,  0.71878622,  0.24743316,  0.64409548,  0.72917719,
```

```
0.50633015,  0.83788761,  0.67911541,  0.47559417,  0.68268215,
0.71481172,  0.40919629,  0.83731566,  0.64789207,  0.77283782,
0.79286776,  0.60422047,  0.86713941,  0.73238196,  0.62421522,
0.54741581,  0.58763412,  0.63340553,  0.69961688,  0.2300705 ,
0.54968429,  0.50659403,  0.50728133,  0.55793429,  0.43717631,
0.70734601,  0.79017875,  0.6818375 ,  0.69846913,  0.57516541,
0.51716264,  0.57178647,  0.66494269,  0.75956592,  0.87025936,
0.39644657,  0.66863461,  0.63429199,  0.79509096,  0.58422928,
0.81901025,  0.6271807 ,  0.79864883,  0.87406723,  0.87767955,
0.80885331,  0.24001575,  0.53616839,  0.59466685,  0.74622329,
0.52751443,  0.65160611,  0.48211623,  0.53828526,  0.62985837,
0.77808956,  0.39156496,  0.80479628,  0.44836745,  0.72913468,
0.60149234,  0.79974148,  0.60015093,  0.65950121,  0.79820429,
0.55032497,  0.86933862,  0.93932477,  0.74761014,  0.28513757,
0.74902598,  0.77724129,  0.92201897,  0.43247289,  0.68575843,
0.71344788,  0.50885322,  0.80066448,  0.62072054,  0.71977792,
0.54526711,  0.62558579,  0.66201484,  0.41067287,  0.76355123,
0.34356673,  0.79894115,  0.85782661,  0.42341082,  0.75656745,
0.52016438,  0.65395414,  0.71416909])
```

In [70]:

```
print("Root Mean Square Error: {}".format(mean_squared_error(y_norm_test, test_norm_pred)))
print("R2 Score: {}".format(r2_score(y_norm_test, test_norm_pred)*100))
```

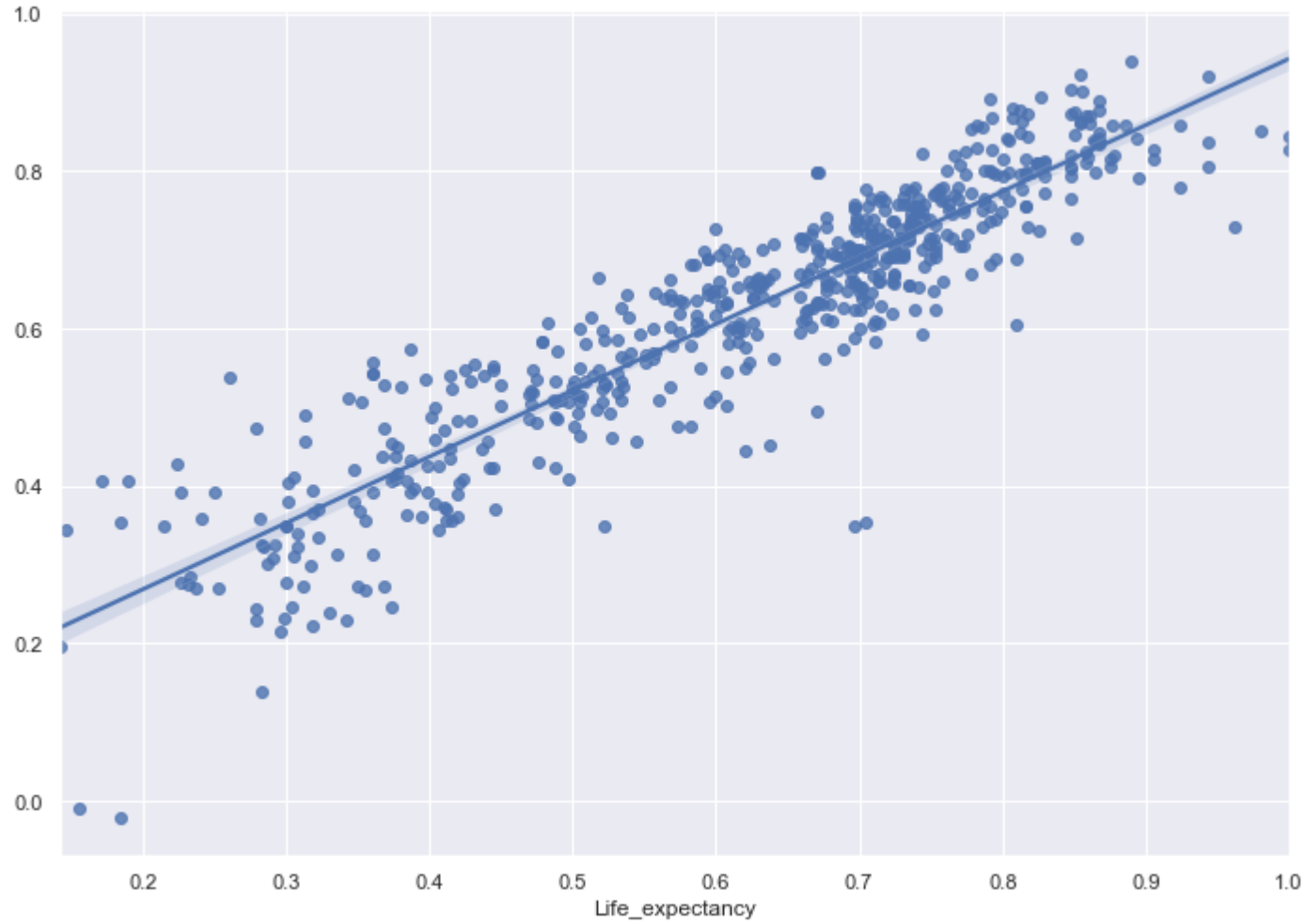
```
Root Mean Square Error: 0.00513562147410703
R2 Score: 83.54823400756966
```

In [71]:

```
### Regression Plotting the outcome of the model
sns.regplot(x = y_norm_test, y = model_norm.predict(X_norm_test))
```

Out[71]:

```
<AxesSubplot:xlabel='Life_expectancy'>
```



Observation:

The Root Mean square error has been reduced from 14.236 to 0.00513 after normalizing the dataset. However, the r2 score turned out to be the same (~83.548) as running linear regression without normalization.

Feature Selection

StandardScalar

In [72]:
```
### Applying StandardScalar to the input dataset
## Apply standard Scalar to the dataset
```

```
sc = StandardScaler()
X_ss_train = pd.DataFrame(sc.fit_transform(X_train), columns = X_train.columns)
X_ss_test = pd.DataFrame(sc.transform(X_test), columns = X_test.columns)
```

In [73]:
```
## Displaying few records from StandardScalar dataset
X_ss_train.head()
```

Out[73]:

	Country	Year	Status	Adult_Mortality	infant_deaths	Alcohol	percentage_expenditure	Hepatitis_B	Measles	BMI	under_fi
0	-1.222471	-0.116557	0.463109	-0.317973	-0.257514	0.170306	-0.192120	0.699278	-0.200110	-0.630633	
1	-1.222471	-1.417792	0.463109	-0.110684	-0.257514	-0.194222	-0.291864	-2.517059	-0.200110	-0.798925	
2	0.841889	0.750932	0.463109	-0.182438	-0.257514	1.447394	-0.369082	0.804159	-0.200110	0.309824	
3	1.411368	1.184677	0.463109	-1.218884	-0.230683	-0.819127	-0.064479	0.769199	-0.198729	1.042390	
4	-1.293656	-1.200920	0.463109	-0.565126	-0.257514	-1.106781	0.113299	0.804159	-0.200110	-0.506888	

Lasso Regression

In [74]:
```
# Create an instance of Lasso Regression implementation
lasso = Lasso(alpha=1.0)
```

In [75]:
```
# Fit the Lasso model
lasso.fit(X_ss_train, y_train)
```

Out[75]:
```
Lasso()
```

In [76]:
```
# Create the model score
lasso.score(X_ss_test, y_test), lasso.score(X_ss_train, y_train)
```

Out[76]:
```
(0.7868452107561621, 0.7742289353976738)
```

In [77]:
```
## Printing coefficient of the lasso
```

```
imp = lasso.coef_
imp
```

Out[77]:
```
array([ 0.        ,  0.        , -0.29081616, -2.40872833, -0.        ,
        0.        ,  0.        ,  0.        , -0.        ,  0.58079753,
       -0.        ,  0.16020648,  0.        ,  0.51746363, -1.85516276,
        0.        , -0.        , -0.        , -0.        ,  1.52967366,
        2.51294607])
```

In [78]:
```
indices = []
for i in range(0, len(imp)):
    if imp[i] > 0:
        indices.append(i)
indices
```

Out[78]:
```
[9, 11, 13, 19, 20]
```

In [79]:
```
## Printing the features corresponding to the index
for i in range(0, len(indices)):
    print(X_ss_train.columns[indices[i]])
```

```
BMI
Polio
Diphtheria
Income_composition_of_resources
Schooling
```

Observation

The best features identified by Lasso Regression are as below. This is same as what is identified by correlation matrix.

- BMI
- Polio
- Diphtheria
- Income_composition_of_resources
- Schooling

OLS Regression (Ordinary Least Square)

In [81]:
```
## Considering the normalized dataset (MinMaxScaler)
print("The shape of traning dataset: {}".format(X_norm_train.shape))
```

```
print("The shape of test dataset: {}".format(X_norm_test.shape))
print("The shape of target traning dataset: {}".format(y_norm_test.shape))
print("The shape of target test dataset: {}".format(y_norm_test.shape))
```

```
The shape of traning dataset: (2350, 21)
The shape of test dataset: (588, 21)
The shape of target traning dataset: (588,)
The shape of target test dataset: (588,)
```

In [82]:

```
## Showing few sample
X_norm_train.head()
```

Out[82]:

	Country	Year	Status	Adult_Mortality	infant_deaths	Alcohol	percentage_expenditure	Hepatitis_B	Measles	BMI	under_five_
456	0.140625	0.466667	1.0	0.173130	0.000000	0.295073	0.017734	0.969388	0.000000	0.283893	
462	0.140625	0.066667	1.0	0.209141	0.000000	0.212766	0.007738	0.030612	0.000000	0.244496	
2172	0.744792	0.733333	1.0	0.196676	0.000000	0.583427	0.000000	1.000000	0.000000	0.504056	
2667	0.911458	0.866667	1.0	0.016620	0.001667	0.071669	0.030526	0.989796	0.000075	0.675550	
381	0.119792	0.133333	1.0	0.130194	0.000000	0.006719	0.048342	1.000000	0.000000	0.312862	

In [83]:

```
# Adds a constant term to the predictor
x_ols = sm.add_constant(X_norm_train)
```

In [84]:

```
## showing few records in x_ols
x_ols.head()
```

Out[84]:

	const	Country	Year	Status	Adult_Mortality	infant_deaths	Alcohol	percentage_expenditure	Hepatitis_B	Measles	BMI	und
456	1.0	0.140625	0.466667	1.0	0.173130	0.000000	0.295073	0.017734	0.969388	0.000000	0.283893	
462	1.0	0.140625	0.066667	1.0	0.209141	0.000000	0.212766	0.007738	0.030612	0.000000	0.244496	
2172	1.0	0.744792	0.733333	1.0	0.196676	0.000000	0.583427	0.000000	1.000000	0.000000	0.504056	
2667	1.0	0.911458	0.866667	1.0	0.016620	0.001667	0.071669	0.030526	0.989796	0.000075	0.675550	
381	1.0	0.119792	0.133333	1.0	0.130194	0.000000	0.006719	0.048342	1.000000	0.000000	0.312862	

In [85]:
```
## Fitting ols regression
result_ols = sm.OLS(y_norm_train, X_norm_train).fit()
```

In [86]:
```
## Printing result summary
print(result_ols.summary())
```

```
                                 OLS Regression Results
=======================================================================================
Dep. Variable:        Life_expectancy   R-squared (uncentered):                   0.981
Model:                            OLS   Adj. R-squared (uncentered):              0.981
Method:                 Least Squares   F-statistic:                              5759.
Date:                Fri, 14 Oct 2022   Prob (F-statistic):                        0.00
Time:                        18:47:09   Log-Likelihood:                          2337.7
No. Observations:                2350   AIC:                                     -4633.
Df Residuals:                    2329   BIC:                                     -4512.
Df Model:                          21
Covariance Type:            nonrobust
=======================================================================================
                                  coef    std err          t      P>|t|      [0.025      0.975]
---------------------------------------------------------------------------------------
Country                          0.0428      0.006      6.756      0.000       0.030       0.055
Year                            -0.0077      0.007     -1.173      0.241      -0.020       0.005
Status                           0.0531      0.006      9.381      0.000       0.042       0.064
Adult_Mortality                 -0.1561      0.013    -11.642      0.000      -0.182      -0.130
infant_deaths                    1.9032      0.372      5.111      0.000       1.173       2.633
Alcohol                          0.0500      0.011      4.615      0.000       0.029       0.071
percentage_expenditure           0.1052      0.036      2.890      0.004       0.034       0.177
Hepatitis_B                      0.0055      0.008      0.685      0.493      -0.010       0.021
Measles                         -0.0681      0.040     -1.704      0.088      -0.146       0.010
BMI                              0.0989      0.010      9.447      0.000       0.078       0.119
under_five_deaths               -1.9999      0.379     -5.281      0.000      -2.743      -1.257
Polio                            0.0701      0.011      6.619      0.000       0.049       0.091
Total_expenditure                0.1053      0.013      7.828      0.000       0.079       0.132
Diphtheria                       0.0880      0.011      7.835      0.000       0.066       0.110
HIV/AIDS                        -0.5097      0.021    -24.062      0.000      -0.551      -0.468
GDP                              0.0402      0.032      1.247      0.212      -0.023       0.103
Population                      -0.0073      0.054     -0.136      0.892      -0.112       0.098
thinness_1_19_years              0.0418      0.034      1.240      0.215      -0.024       0.108
thinness_5_9_years               0.0634      0.034      1.850      0.064      -0.004       0.131
Income_composition_of_resources  0.1809      0.015     12.172      0.000       0.152       0.210
Schooling                        0.4215      0.021     20.476      0.000       0.381       0.462
==============================================================================
Omnibus:                      234.220   Durbin-Watson:                   2.053
Prob(Omnibus):                  0.000   Jarque-Bera (JB):             1092.757
Skew:                           0.366   Prob(JB):                    5.14e-238
```

```
Kurtosis:                       6.259   Cond. No.                         607.
==============================================================================

Notes:
[1] R² is computed without centering (uncentered) since the model does not contain a constant.
[2] Standard Errors assume that the covariance matrix of the errors is correctly specified.
```

Now, if we consider that a good p value would be a one greater than 0.05 or within 5%, then the attributes or features that we can considered as best features are:

- Year
- Hepatitis_B
- GDP
- Population
- thinness_1_19_years
- thinness_5_9_years
- Percentage_expenditure

In [ ]: