# Life Expectancy Prediction

## About Dataset

### Context

Although there have been lot of studies undertaken in the past on factors affecting life expectancy considering demographic variables, income composition and mortality rates. It was found that affect of immunization and human development index was not taken into account in the past. Also, some of the past research was done considering multiple linear regression based on data set of one year for all the countries. Hence, this gives motivation to resolve both the factors stated previously by formulating a regression model based on mixed effects model and multiple linear regression while considering data from a period of 2000 to 2015 for all the countries. Important immunization like Hepatitis B, Polio and Diphtheria will also be considered. In a nutshell, this study will focus on immunization factors, mortality factors, economic factors, social factors and other health related factors as well. Since the observations this dataset are based on different countries, it will be easier for a country to determine the predicting factor which is contributing to lower value of life expectancy. This will help in suggesting a country which area should be given importance in order to efficiently improve the life expectancy of its population.

### Content

The project relies on accuracy of data. 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 for all countries The data-sets are made available to public for the purpose of health data analysis. The data-set related to life expectancy, health factors for 193 countries has been collected from the same WHO data repository website and its corresponding economic data was collected from United Nation website. Among all categories of health-related factors only those critical factors were chosen which are more representative. It has been observed that in the past 15 years , there has been a huge development in health sector resulting in improvement of human mortality rates especially in the developing nations in comparison to the past 30 years. Therefore, in this project we have considered data from year 2000-2015 for 193 countries for further analysis. The individual data files have been merged together into a single data-set. On initial visual inspection of the data showed some missing values. As the data-sets were from WHO, we found no evident errors. Missing data was handled in R software by using Missmap command. The result indicated that most of the missing data was for population, Hepatitis B and GDP. The missing data were from less known countries like Vanuatu, Tonga, Togo, Cabo Verde etc. Finding all data for these countries was difficult and hence, it was decided that we exclude these countries from the final model data-set. The final merged file(final dataset) consists of 22 Columns and 2938 rows which meant 20 predicting variables. All predicting variables was then divided into several broad categories:Immunization related factors, Mortality factors, Economical factors and Social factors.

### Acknowledgements

The data was collected from WHO and United Nations website with the help of Deeksha Russell and Duan Wang.

## 🎯 Project Goals

- We will be exploring the data, cleaning, analyizng, and perfroming regression analysis on our data.
- Our aim is to build a robust model with high R-squared score and low mean square error for predicting life expectency depending our selected features while addressing the model's assumptions.

# EDA

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

from sklearn.linear_model import LinearRegression
from sklearn.preprocessing import PolynomialFeatures
from sklearn.model_selection import train_test_split
from sklearn.metrics import mean_squared_error, r2_score

from sklearn.experimental import enable_iterative_imputer
from sklearn.impute import IterativeImputer
from sklearn.ensemble import RandomForestRegressor

import scipy.stats as stats

import warnings
warnings.filterwarnings("ignore")
```

In [2]:
```
df = pd.read_csv('Life Expectancy Data.csv')
pd.set_option('display.max_columns', None)
```

In [3]:
```
df.sample(5)
```

Out[3]:

| | 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 | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **1198** | India | 2003 | Developing | 63.7 | 216.0 | 1700 | 1.19 | 19.480868 | NaN | 47147 | 12.6 | 2200 | 57.0 | 4.30 | 61.0 | 0.3 | 541 |
| **2339** | Slovakia | 2005 | Developed | 74.0 | 141.0 | 0 | 10.81 | 0.000000 | 99.0 | 0 | 53.1 | 0 | 99.0 | 7.40 | 99.0 | 0.1 | |
| **1361** | Kazakhstan | 2000 | Developing | 63.9 | 292.0 | 9 | 6.00 | 112.541157 | 99.0 | 245 | 43.9 | 10 | 96.0 | 4.16 | 97.0 | 0.1 | 1229 |
| **1846** | New Zealand | 2014 | Developed | 81.5 | 67.0 | 0 | 9.07 | 1040.278436 | 93.0 | 280 | 66.9 | 0 | 93.0 | 11.30 | 93.0 | 0.1 | 4453 |
| **736** | Democratic Republic of the Congo | 2000 | Developing | 51.3 | 346.0 | 226 | 1.98 | 0.000000 | NaN | 8282 | 14.9 | 332 | 42.0 | 1.45 | 4.0 | 2.5 | |

In [4]:
```
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
```

Seems that there is some inconssistency in column naming which we will address.

In [5]:
```
column_rename_mapping = {
    'Life expectancy ': 'life_expectancy',
    'Measles ': 'measles',
    ' BMI ': 'bmi',
    'under-five deaths ': 'under_five_deaths',
    'Diphtheria ': 'diphtheria',
    'thinness__1-19_years': 'thinness_10_19_years'
}

df.rename(columns=column_rename_mapping, inplace=True)
```

```
df.rename(columns=lambda x: x.strip().lower().replace(' ', '_'), inplace=True)
df.rename(columns=column_rename_mapping, inplace=True)
df['Status'] = np.where(df['status'] == 'Developing', 0, 1)
df.columns
```

Out[5]:
```
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_10_19_years',
       'thinness_5-9_years', 'income_composition_of_resources', 'schooling',
       'Status'],
      dtype='object')
```

In [6]:
```
df.duplicated().sum()
```

Out[6]:
```
0
```

In [7]:
```
df.isna().sum().to_frame()
```

Out[7]:

| | 0 |
|---|---|
| **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_10_19_years** | 34 |
| **thinness_5-9_years** | 34 |
| **income_composition_of_resources** | 167 |
| **schooling** | 163 |

| | 0 |
|---|---|
| **Status** | 0 |

In [8]:
```
df.describe().T
```

Out[8]:

| | 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 |
| **population** | 2286.0 | 1.275338e+07 | 6.101210e+07 | 34.00000 | 195793.250000 | 1.386542e+06 | 7.420359e+06 | 1.293859e+09 |
| **thinness_10_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 |
| **Status** | 2938.0 | 1.742682e-01 | 3.794045e-01 | 0.00000 | 0.000000 | 0.000000e+00 | 0.000000e+00 | 1.000000e+00 |

There seems to be a large number of nulls in our dataset with some having large standard deviation, so instead of imputing the nulls with an average metric, we will use an advanced technique like `IterativeImputer` from sickit learn and use `RandomForestRegrressor` for estimating/predicting the null values

In [9]:
```
columns_with_missing_values = ['alcohol', 'bmi', 'schooling', 'income_composition_of_resources',
                               'gdp', 'thinness_10_19_years', 'thinness_5-9_years', 'polio',
                               'diphtheria', 'hepatitis_b', 'population', 'total_expenditure','life_expectancy','adult_mortality']

imputer = IterativeImputer(estimator=RandomForestRegressor(n_jobs=-1, random_state=42))

df_imputed = df.copy()

df_imputed[columns_with_missing_values] = imputer.fit_transform(df_imputed[columns_with_missing_values])

df_imputed.isnull().sum().to_frame()
```

Out[9]:

| | 0 |
|---|---|
| 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_10_19_years | 0 |
| thinness_5-9_years | 0 |
| income_composition_of_resources | 0 |
| schooling | 0 |

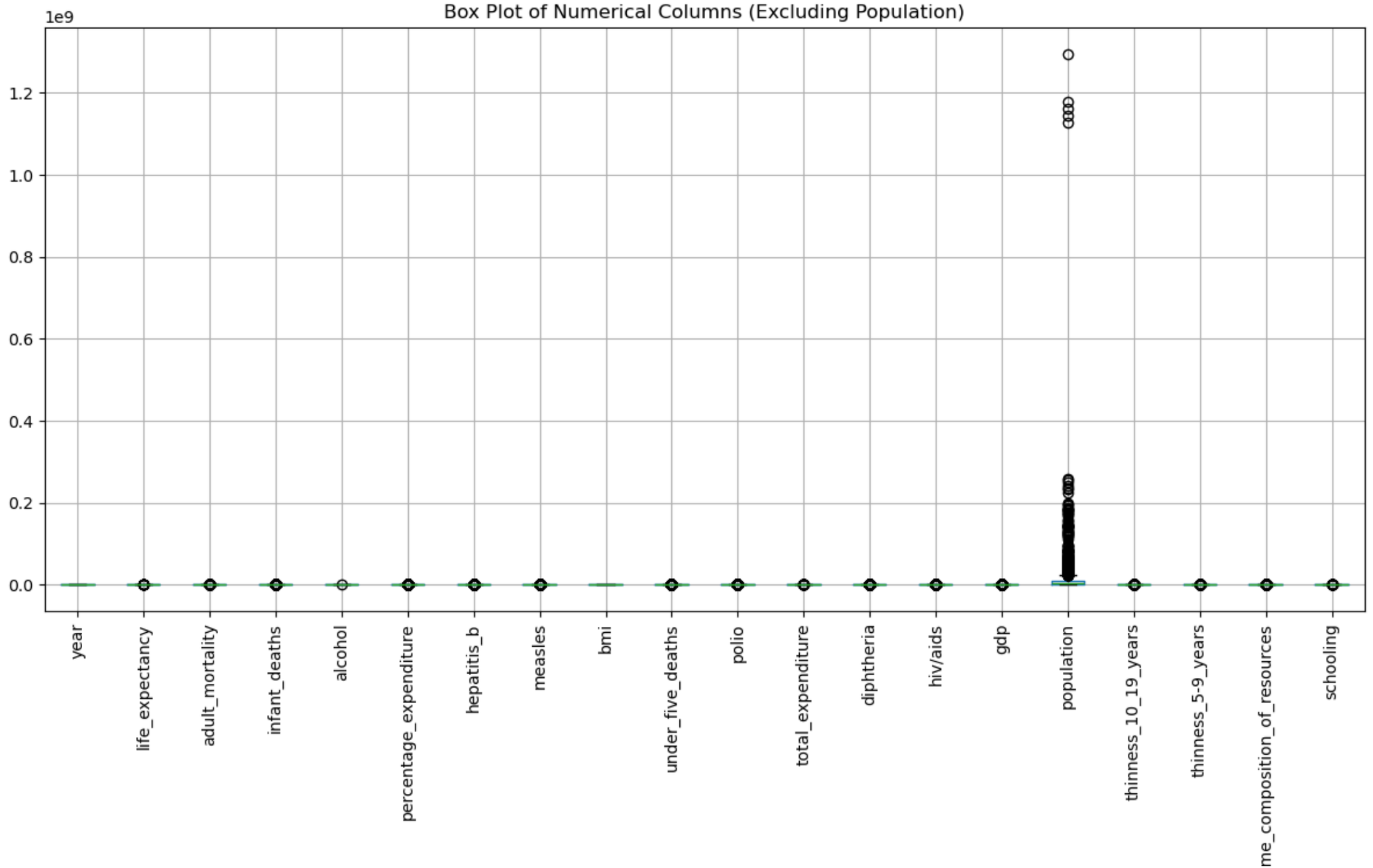
| | 0 |
|---|---|
| **Status** | 0 |

Next, We see some columns have extreme value that are very far from the 75th percentile which indicates the presence of outliers

In [10]:
```
# Checking outlier percentage based on iqr
def calculate_outlier_percentage(column):
    q1 = column.quantile(0.25)
    q3 = column.quantile(0.75)
    iqr = q3 - q1
    lower_bound = q1 - 1.5 * iqr
    upper_bound = q3 + 1.5 * iqr
    outliers = (column < lower_bound) | (column > upper_bound)
    return outliers.mean() * 100

outlier_percentages = df_imputed.select_dtypes(include=['int64', 'float64']).apply(calculate_outlier_percentage)

# Print outlier percentages
print("Outlier Percentages in Each Column:")
print(outlier_percentages)
```

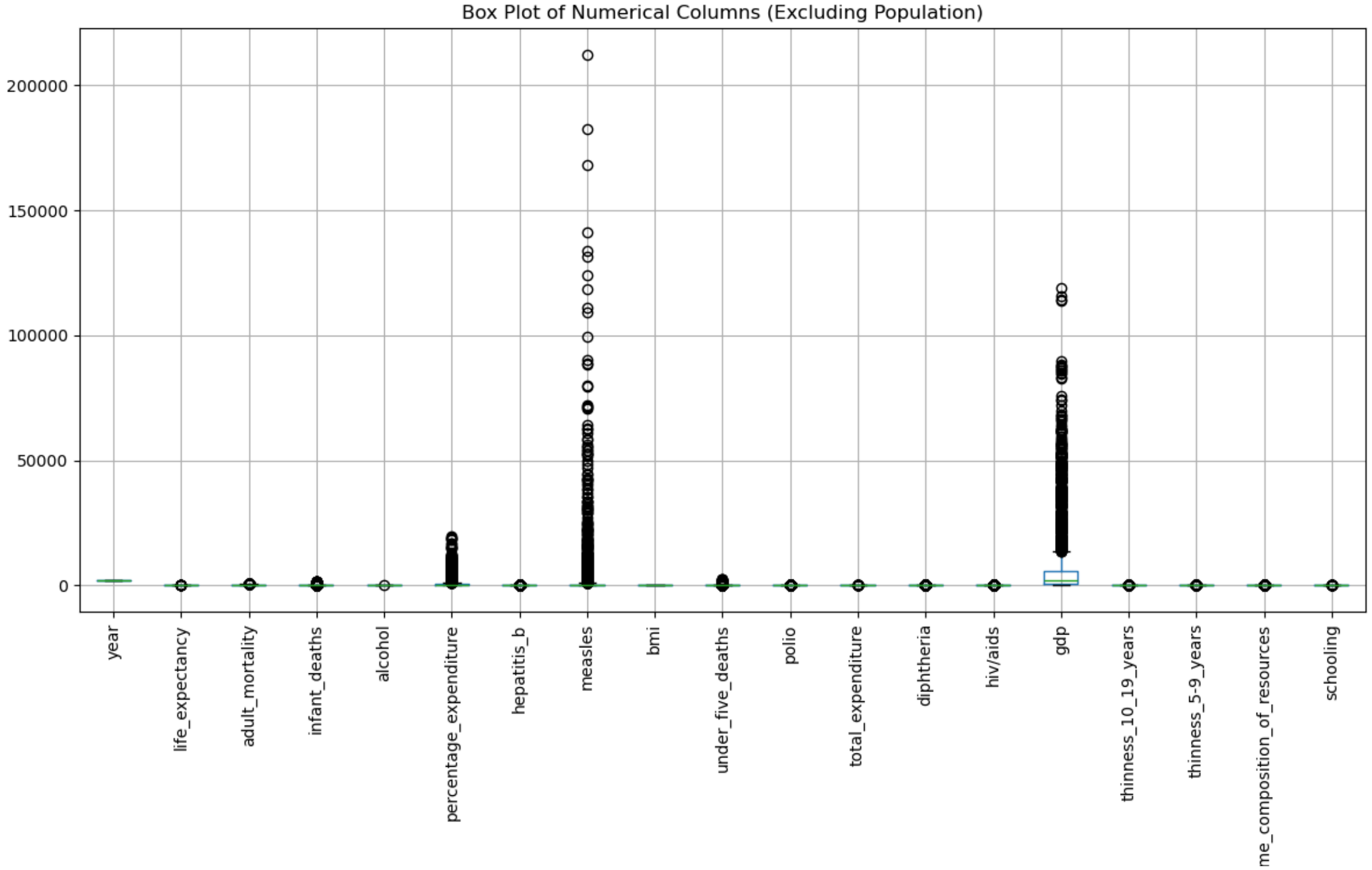
```
Outlier Percentages in Each Column:
year                              0.000000
life_expectancy                   0.408441
adult_mortality                   2.927161
infant_deaths                    10.721579
alcohol                           0.034037
percentage_expenditure           13.240300
hepatitis_b                       6.603131
measles                          18.447924
bmi                               0.000000
under_five_deaths                13.410483
polio                             9.087815
total_expenditure                 1.259360
diphtheria                       10.687543
hiv/aids                         18.447924
gdp                              13.921035
population                       10.993873
thinness_10_19_years              3.403676
thinness_5-9_years                3.369639
income_composition_of_resources   4.492852
schooling                         1.361470
dtype: float64
```

In [11]:
```
numerical_columns = df_imputed.select_dtypes(include=['int64', 'float64'])

plt.figure(figsize=(12, 8))
numerical_columns.boxplot()
plt.title('Box Plot of Numerical Columns (Excluding Population)')
plt.xticks(rotation=90)
plt.tight_layout()
plt.show()
```


Box Plot of Numerical Columns (Excluding Population)
1e9
1.2
1.0
0.8
0.6
0.4
0.2
0.0
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_10_19_years
thinness_5-9_years
me_composition_of_resources
schooling

inco

In [12]:
```
numerical_columns = df_imputed.select_dtypes(include=['int64', 'float64']).drop(columns=['population'])

plt.figure(figsize=(12, 8))
numerical_columns.boxplot()
plt.title('Box Plot of Numerical Columns (Excluding Population)')
plt.xticks(rotation=90)
plt.tight_layout()
plt.show()
```


Box Plot of Numerical Columns (Excluding Population)
200000
150000
100000
50000
0
year
life_expectancy
adult_mortality
infant_deaths
alcohol
percentage_expenditure
hepatitis_b
measles
bmi
under_five_deaths
polio
total_expenditure
diphtheria
hiv/aids
gdp
thinness_10_19_years
thinness_5-9_years
me_composition_of_resources
schooling

incol

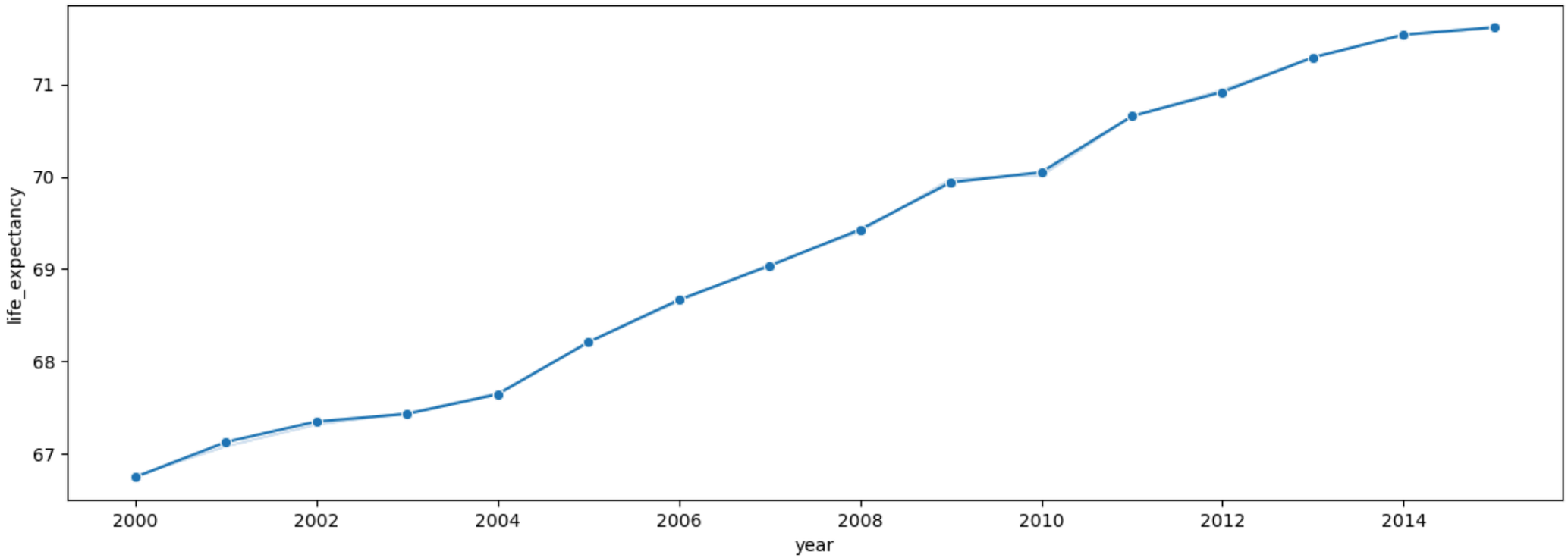
Due to the large variation in our data and the existance of some extreme outliers, while some of them may be true outliers, they would impact our models performance and removing these outliers would cause a huge loss of data so we will use `winsorization` to reduce the extreme outliers to achieve more interpretable analysis and better results in our model.

- While this doesn't completly eleminate outliers it reduces their impact. there may be still columns with extreme values, but we will adress this columns while building our model and see how they affect our model.

In [13]:
```
numerical_columns = df_imputed.select_dtypes(include=['int64', 'float64']).columns

# Function to winsorize outliers
def winsorize_outliers(df, columns, lower_percentile=0.05, upper_percentile=0.95):
    # Calculate lower and upper bounds for winsorization
    lower_bound = df[columns].quantile(lower_percentile)
    upper_bound = df[columns].quantile(upper_percentile)

    # Winsorize outliers
    df_winsorized = df.copy()
    for col in columns:
        df_winsorized[col] = np.where(df_winsorized[col] < lower_bound[col], lower_bound[col], df_winsorized[col])
        df_winsorized[col] = np.where(df_winsorized[col] > upper_bound[col], upper_bound[col], df_winsorized[col])

    return df_winsorized


df_imputed_win = winsorize_outliers(df_imputed, numerical_columns)
```

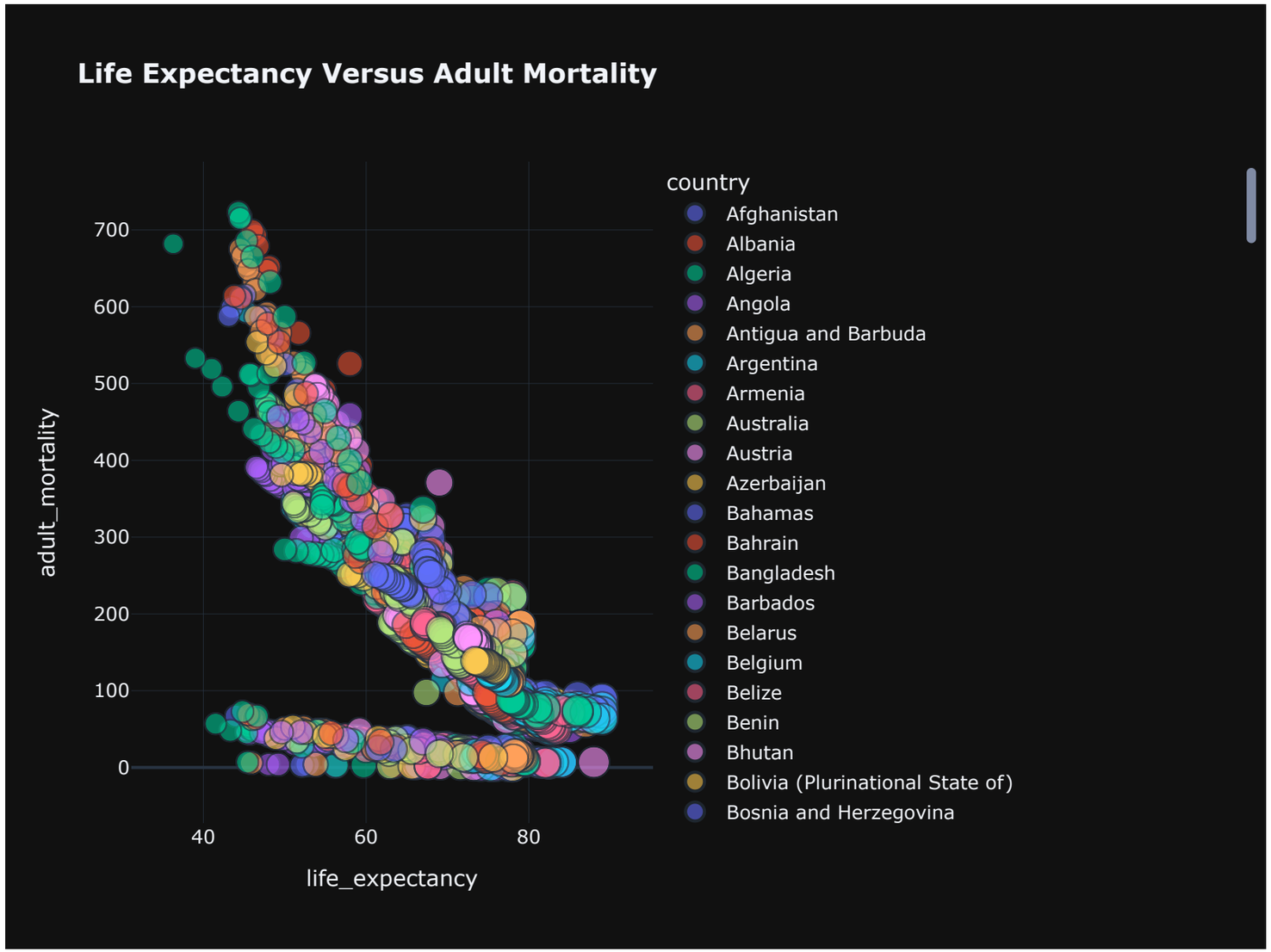
# Analysis

In [15]:
```
plt.figure(figsize=(15,5))

sns.lineplot(x = 'year', y = 'life_expectancy', data = df_imputed, marker = 'o' ,errorbar=('ci', False))
plt.show()
```




An analysis reveals a significant increase in life expectancy over the years, suggesting a corresponding decline in mortality rates.

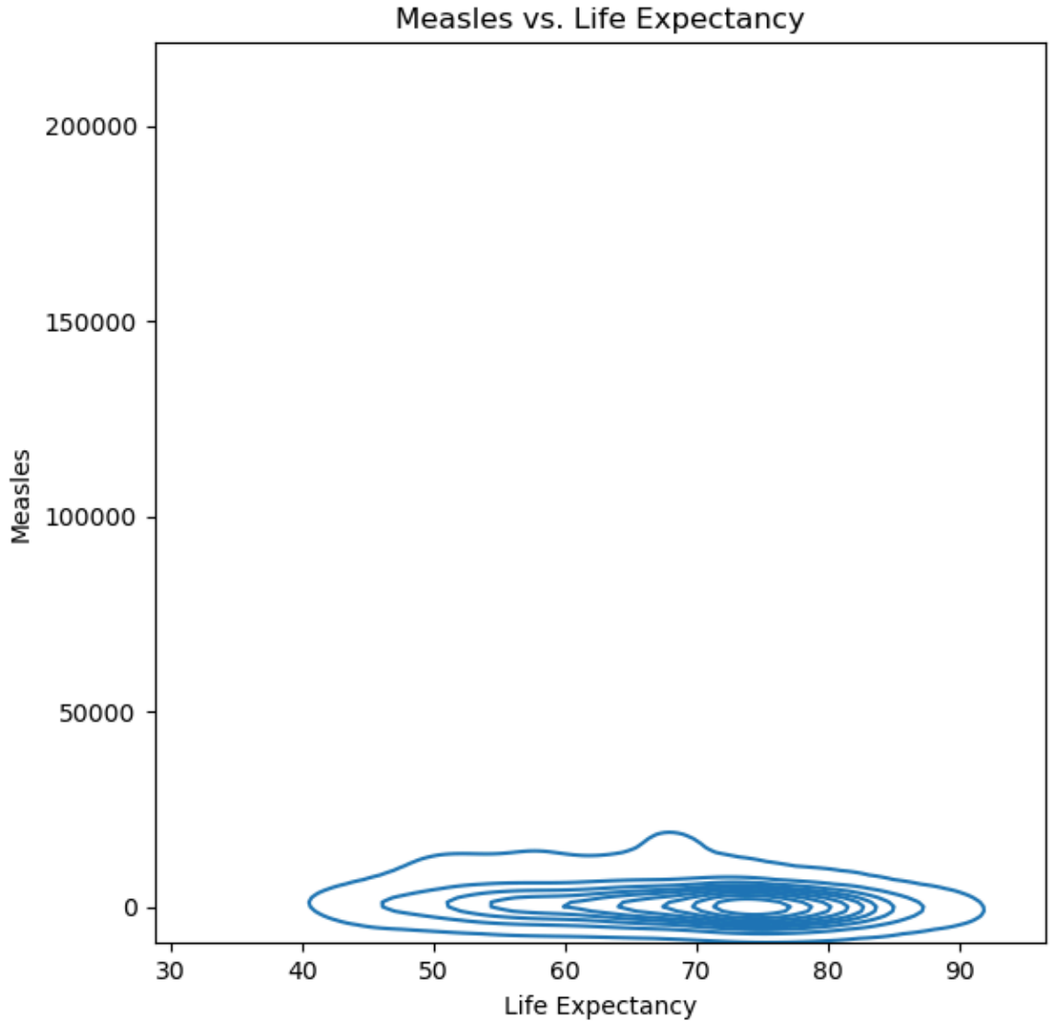
In [16]:
```
import plotly.express as px
px.scatter(df_imputed,y = 'adult_mortality',x='life_expectancy',
           color='country',size='life_expectancy',
           template='plotly_dark',opacity=0.6,
           height= 600, width = 800,
           title='<b> Life Expectancy Versus Adult Mortality')
```

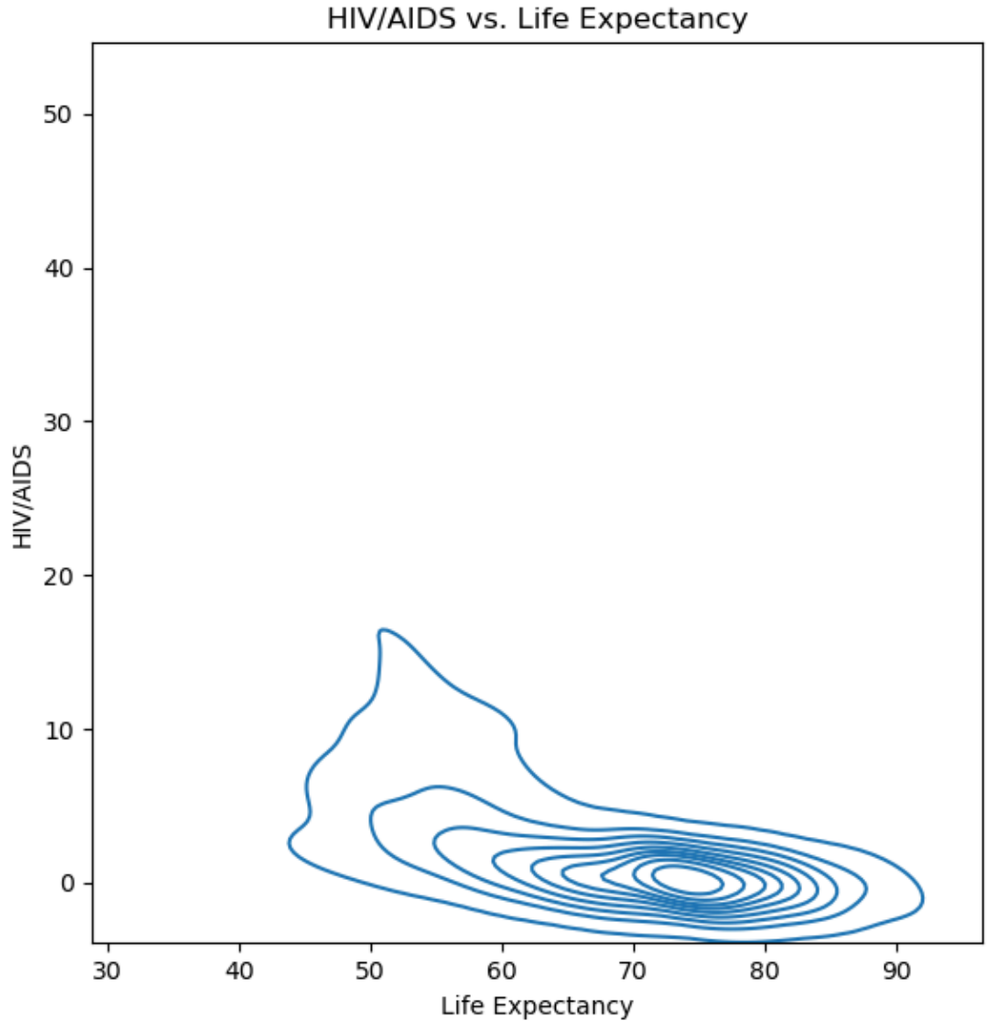


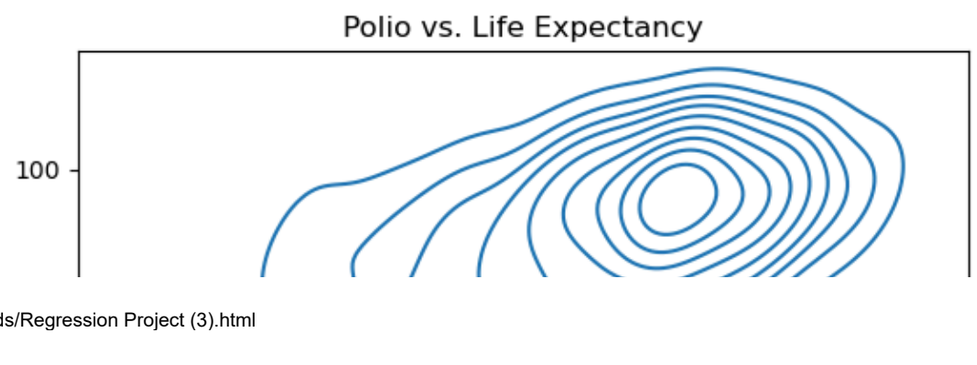

Zimbabwe stands out with the highest adult mortality rate among the listed countries, reporting 723 deaths per 1000 people. Additionally, Zimbabwe also records a lower life expectancy compared to all other countries, although it's not the lowest among them.

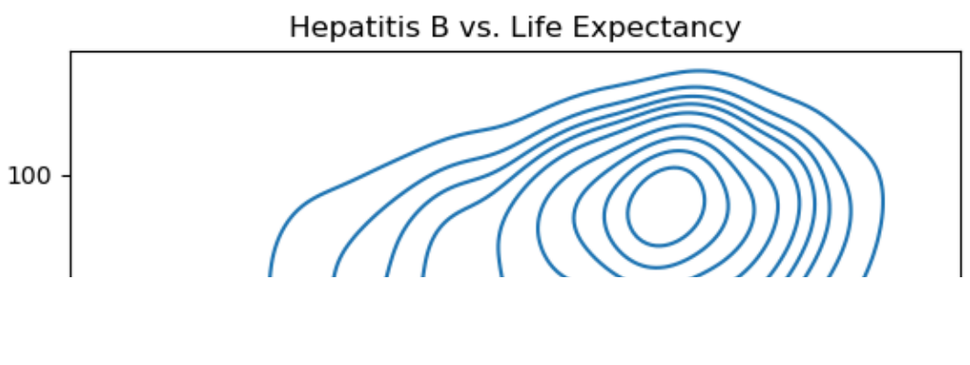
In [17]:
```
import seaborn as sns
import matplotlib.pyplot as plt
```

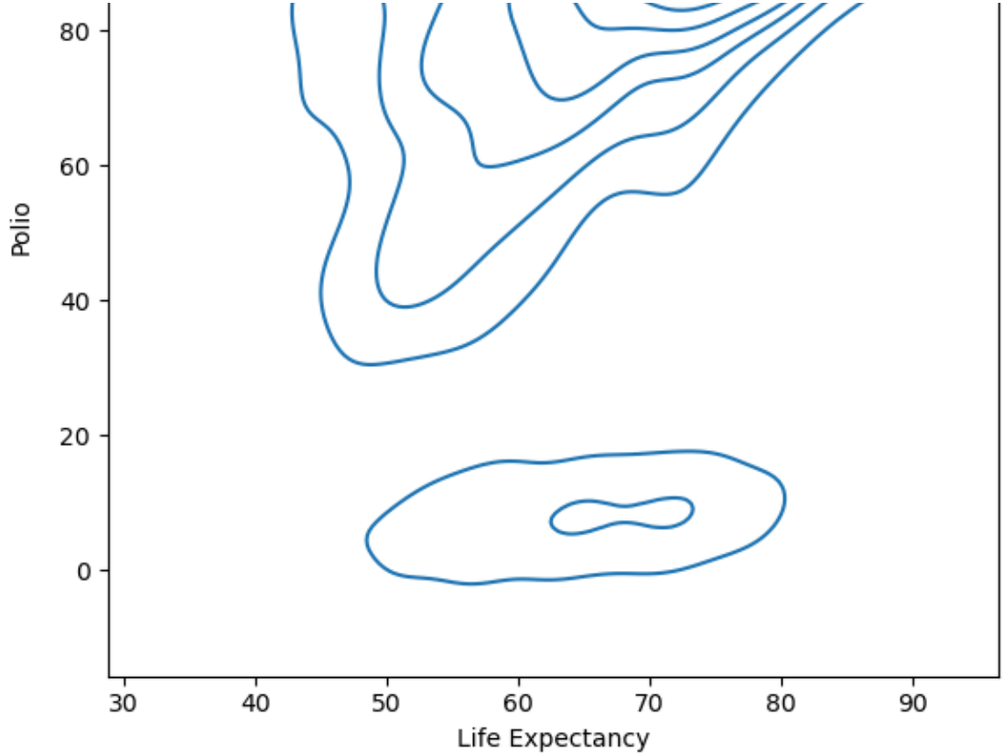
```
fig, axs = plt.subplots(2, 2, figsize=(12, 12))

Target = ['life_expectancy']

sns.kdeplot(data=df_imputed, x=Target[0], y='measles', ax=axs[0][0])
axs[0][0].set_title('Measles vs. Life Expectancy')
axs[0][0].set_xlabel('Life Expectancy')
axs[0][0].set_ylabel('Measles')

sns.kdeplot(data=df_imputed, x=Target[0], y='hiv/aids', ax=axs[0][1])
axs[0][1].set_title('HIV/AIDS vs. Life Expectancy')
axs[0][1].set_xlabel('Life Expectancy')
axs[0][1].set_ylabel('HIV/AIDS')

sns.kdeplot(data=df_imputed, x=Target[0], y='polio', ax=axs[1][0])
axs[1][0].set_title('Polio vs. Life Expectancy')
axs[1][0].set_xlabel('Life Expectancy')
axs[1][0].set_ylabel('Polio')

sns.kdeplot(data=df_imputed, x=Target[0], y='hepatitis_b', ax=axs[1][1])
axs[1][1].set_title('Hepatitis B vs. Life Expectancy')
axs[1][1].set_xlabel('Life Expectancy')
axs[1][1].set_ylabel('Hepatitis B')

plt.tight_layout()
plt.show()
```

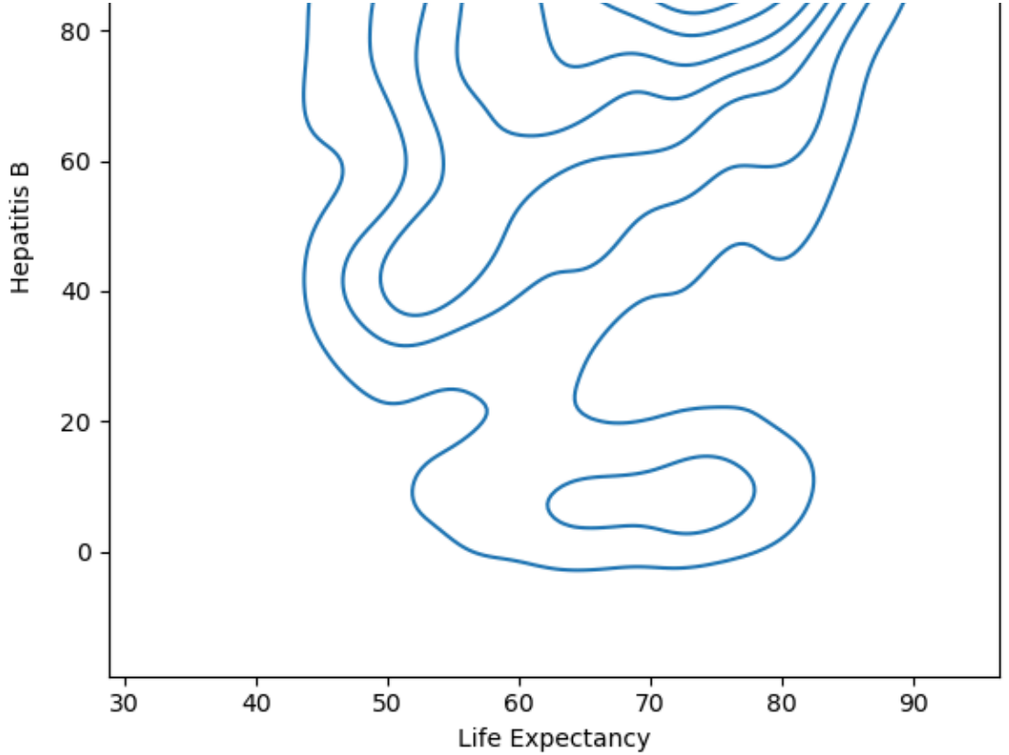

Measles vs. Life Expectancy
Measles
200000
150000
100000
50000
0
30
40
50
60
70
80
90
Life Expectancy


HIV/AIDS vs. Life Expectancy
HIV/AIDS
50
40
30
20
10
0
30
40
50
60
70
80
90
Life Expectancy


Polio vs. Life Expectancy
100

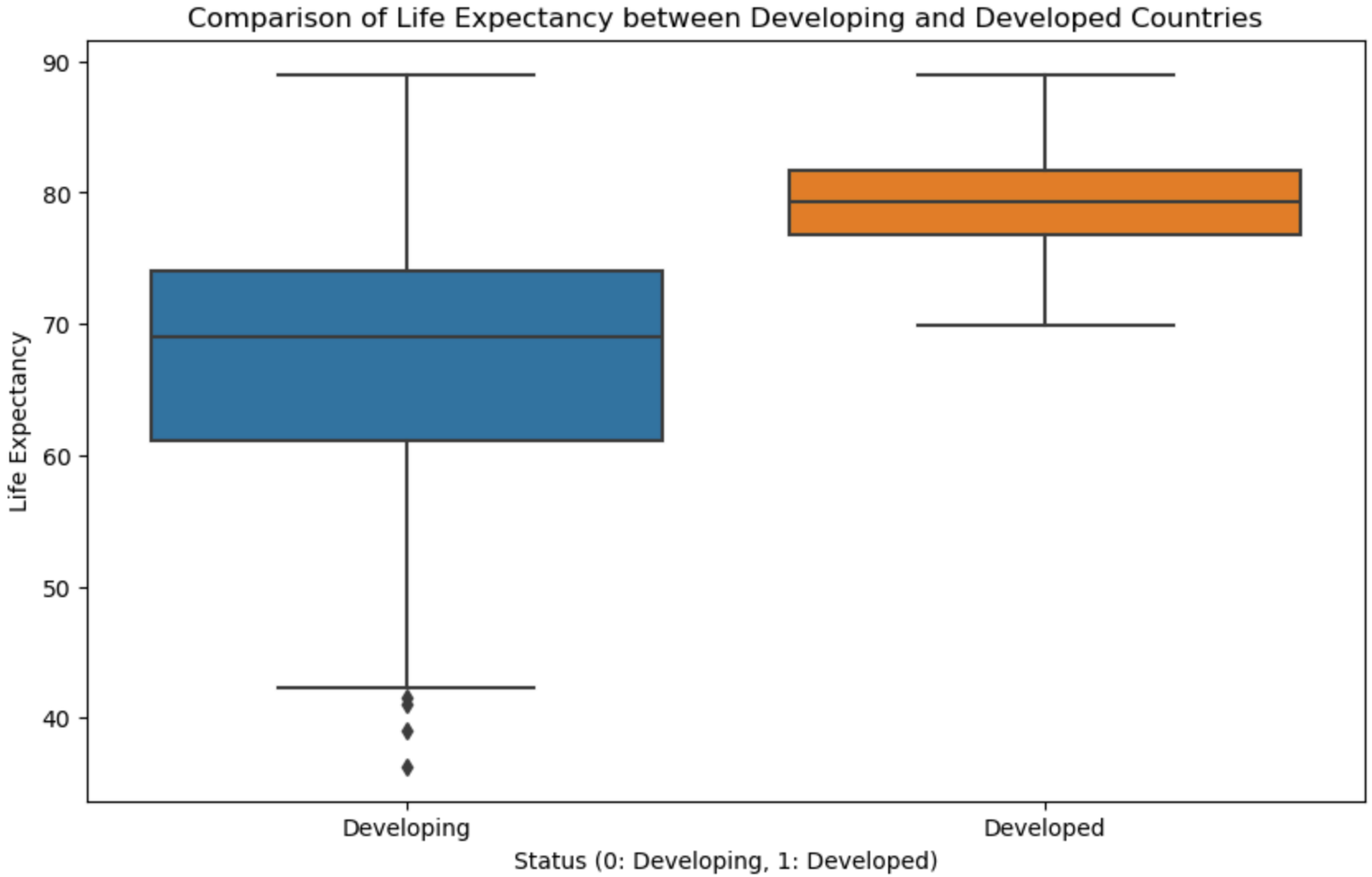

Hepatitis B vs. Life Expectancy
100




Based on the analysis of diseases, it has been observed that measles has a relatively minor impact on life expectancy. Conversely, HIV/AIDS demonstrates a strong negative correlation with life expectancy, indicating that as HIV/AIDS prevalence increases, life expectancy tends to decrease. Additionally, there is a weak correlation observed between polio, hepatitis B, and life expectancy.

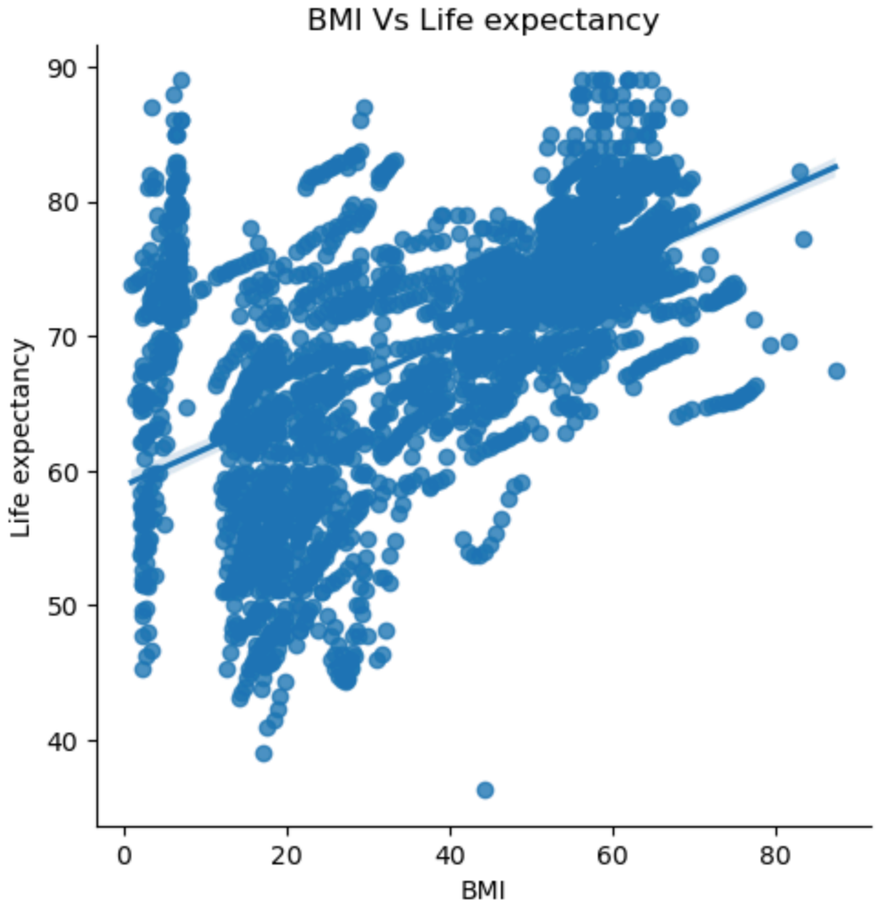
In [18]:
```
plt.figure(figsize=(10, 6))
sns.boxplot(data=df_imputed, x='status', y='life_expectancy')
plt.title('Comparison of Life Expectancy between Developing and Developed Countries')
plt.xlabel('Status (0: Developing, 1: Developed)')
plt.ylabel('Life Expectancy')
plt.show()
```




It's clear from the analysis that in developed countries, where people enjoy better living conditions, life expectancy tends to be higher compared to developing nations.

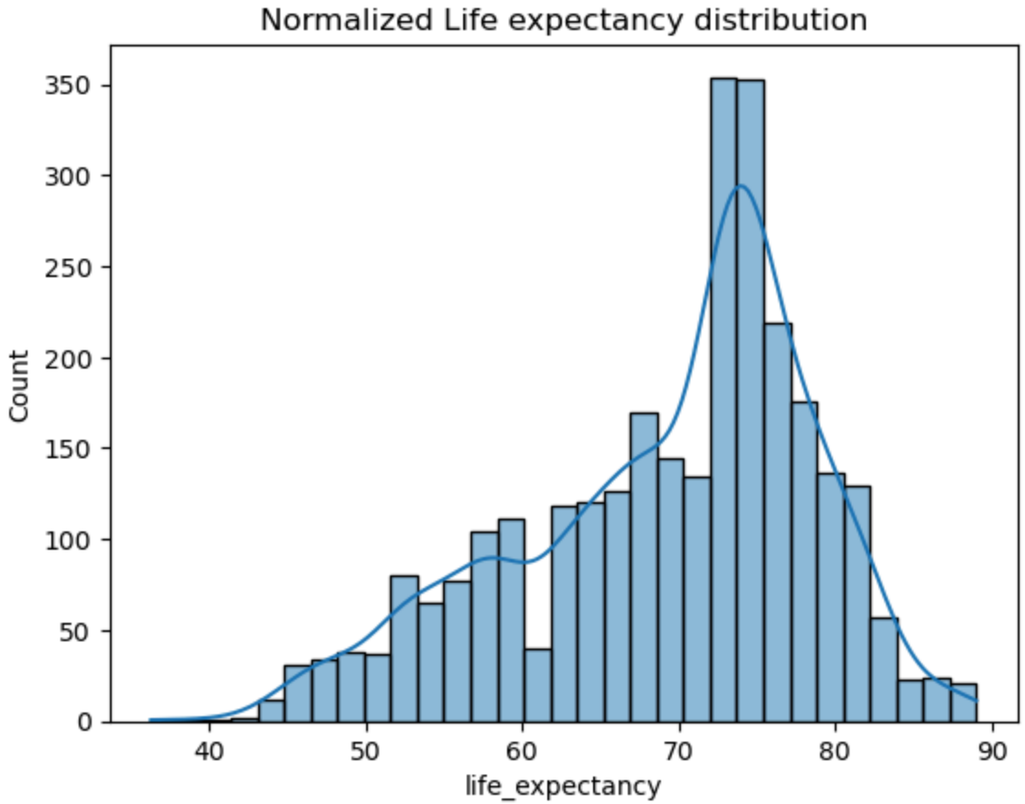
In [19]:
```
sns.lmplot(x = 'bmi', y = 'life_expectancy', data = df_imputed )
plt.xlabel("BMI")
plt.ylabel("Life expectancy")
plt.title("BMI Vs Life expectancy")
```

Out[19]:
```
Text(0.5, 1.0, 'BMI Vs Life expectancy')
```




Analyzed data reveals a robust and strong positive correlation between BMI and life expectancy.

In [20]:
```
sns.histplot(data=df_imputed, x="life_expectancy", kde=True, bins=31)
plt.title('Normalized Life expectancy distribution')
plt.show()
```




It almost have a normal distribution with negative skew.

In [ ]:

## Kolmogorov-Smirnov test for distributional differences

### Significance level 0.05

In [21]:
```
import pandas as pd
from scipy.stats import ks_2samp

# Kolmogorov-Smirnov test
```

```python
def perform_ks_test(original_data, winsorized_data):
    # Perform KS test
    ks_statistic, p_value = ks_2samp(original_data, winsorized_data)

    return ks_statistic, p_value

# Perform KS test for each numerical column
ks_results = {}
for col in numerical_columns:
    ks_statistic, p_value = perform_ks_test(df_imputed[col], df_imputed_win[col])
    reject_null = p_value < 0.05
    ks_results[col] = {'KS Statistic': ks_statistic, 'P-value': p_value, 'Reject Null Hypothesis': reject_null}

ks_results_df = pd.DataFrame.from_dict(ks_results, orient='index')

ks_results_df
```

Out[21]:

| | KS Statistic | P-value | Reject Null Hypothesis |
|---|---|---|---|
| **year** | 0.000000 | 1.000000 | False |
| **life_expectancy** | 0.049353 | 0.001557 | True |
| **adult_mortality** | 0.050034 | 0.001276 | True |
| **infant_deaths** | 0.050034 | 0.001276 | True |
| **alcohol** | 0.050034 | 0.001276 | True |
| **percentage_expenditure** | 0.050034 | 0.001276 | True |
| **hepatitis_b** | 0.031995 | 0.098819 | False |
| **measles** | 0.050034 | 0.001276 | True |
| **bmi** | 0.049694 | 0.001410 | True |
| **under_five_deaths** | 0.049694 | 0.001410 | True |
| **polio** | 0.034377 | 0.062097 | False |
| **total_expenditure** | 0.050034 | 0.001276 | True |
| **diphtheria** | 0.035058 | 0.054044 | False |
| **hiv/aids** | 0.050034 | 0.001276 | True |
| **gdp** | 0.050034 | 0.001276 | True |
| **population** | 0.050034 | 0.001276 | True |
| **thinness_10_19_years** | 0.049013 | 0.001718 | True |
| **thinness_5-9_years** | 0.049353 | 0.001557 | True |
| **income_composition_of_resources** | 0.050034 | 0.001276 | True |
| **schooling** | 0.048673 | 0.001895 | True |

We notice here that most of our data distributions are different from the original data so we will base our analysis on the original imputed data and use the winsorized data for modeling

In [22]:
```
df_imputed.head()
```

Out[22]:

| | country | year | status | life_expectancy | adult_mortality | infant_deaths | alcohol | percentage_expenditure | hepatitis_b | measles | bmi | under_five_deaths | polio | tota |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **0** | Afghanistan | 2015 | Developing | 65.0 | 263.0 | 62 | 0.01 | 71.279624 | 65.0 | 1154 | 19.1 | 83 | 6.0 | |
| **1** | Afghanistan | 2014 | Developing | 59.9 | 271.0 | 64 | 0.01 | 73.523582 | 62.0 | 492 | 18.6 | 86 | 58.0 | |
| **2** | Afghanistan | 2013 | Developing | 59.9 | 268.0 | 66 | 0.01 | 73.219243 | 64.0 | 430 | 18.1 | 89 | 62.0 | |
| **3** | Afghanistan | 2012 | Developing | 59.5 | 272.0 | 69 | 0.01 | 78.184215 | 67.0 | 2787 | 17.6 | 93 | 67.0 | |
| **4** | Afghanistan | 2011 | Developing | 59.2 | 275.0 | 71 | 0.01 | 7.097109 | 68.0 | 3013 | 17.2 | 97 | 68.0 | |

In [ ]:
```
```

# Model Building

We will be using different types of multiregression and performing feature selection to achieve a strong robust model while addressing model assumptions.

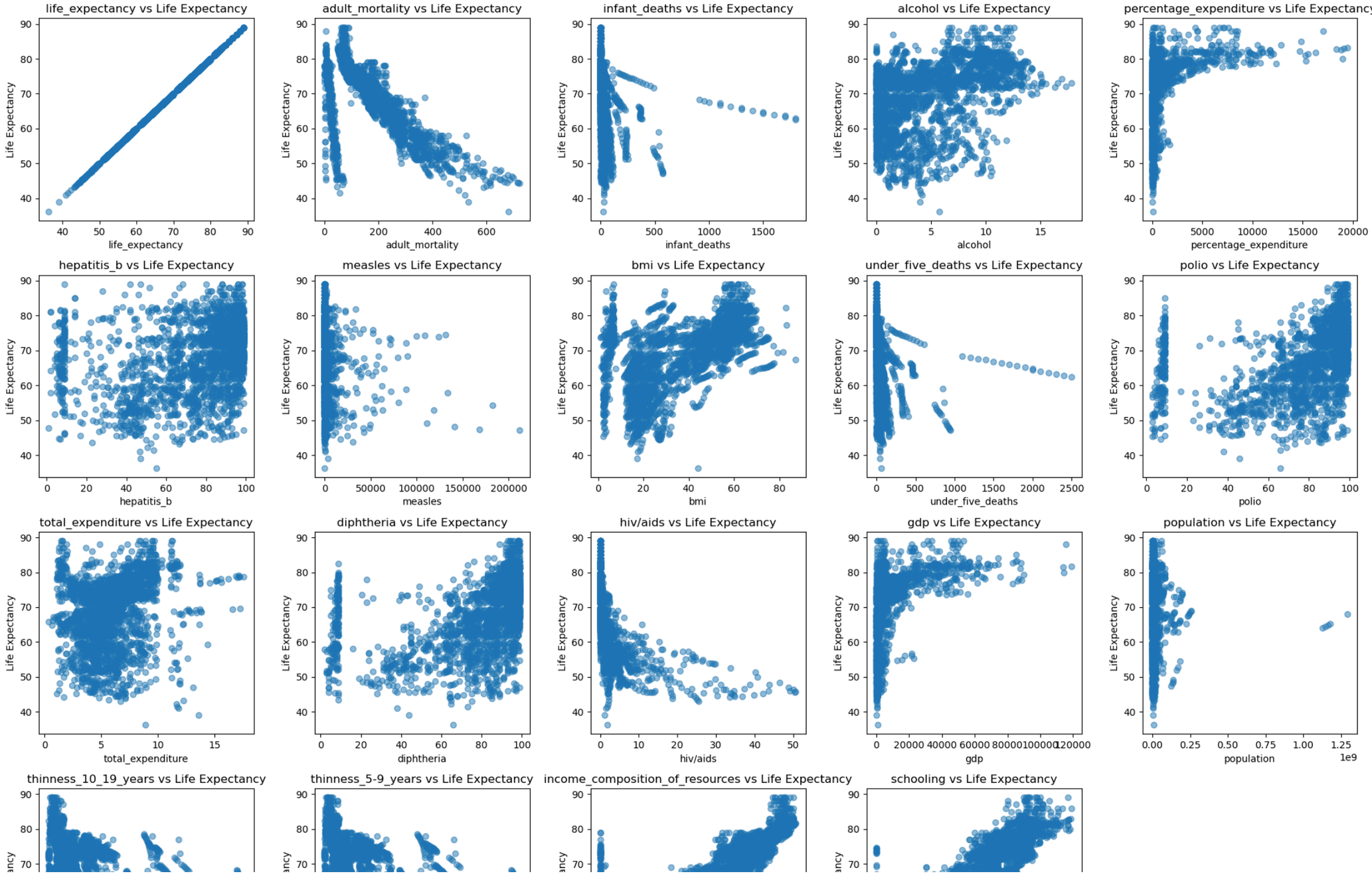
## Model Assumption: Independance

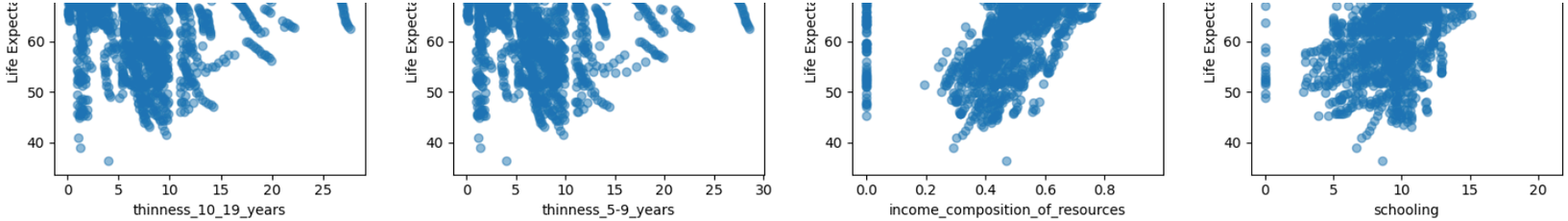
The independent observation assumption states that each observation in the dataset is independent. Since data has some observations (but not all) over different years some observations are not independant. The assumption is partially violated.

## Model Assumption: Linearity

In [23]:
```
numerical_cols = df_imputed.select_dtypes(include=['float64', 'int64']).columns.tolist()
numerical_cols.remove('year')

plt.figure(figsize=(20, 15))
for i, column in enumerate(numerical_cols):
    plt.subplot(4, 5, i+1)
    plt.scatter(df_imputed[column], df_imputed['life_expectancy'], alpha=0.5)
```

```
    plt.xlabel(column)
    plt.ylabel('Life Expectancy')
    plt.title(f'{column} vs Life Expectancy')

plt.tight_layout()
plt.show()
```

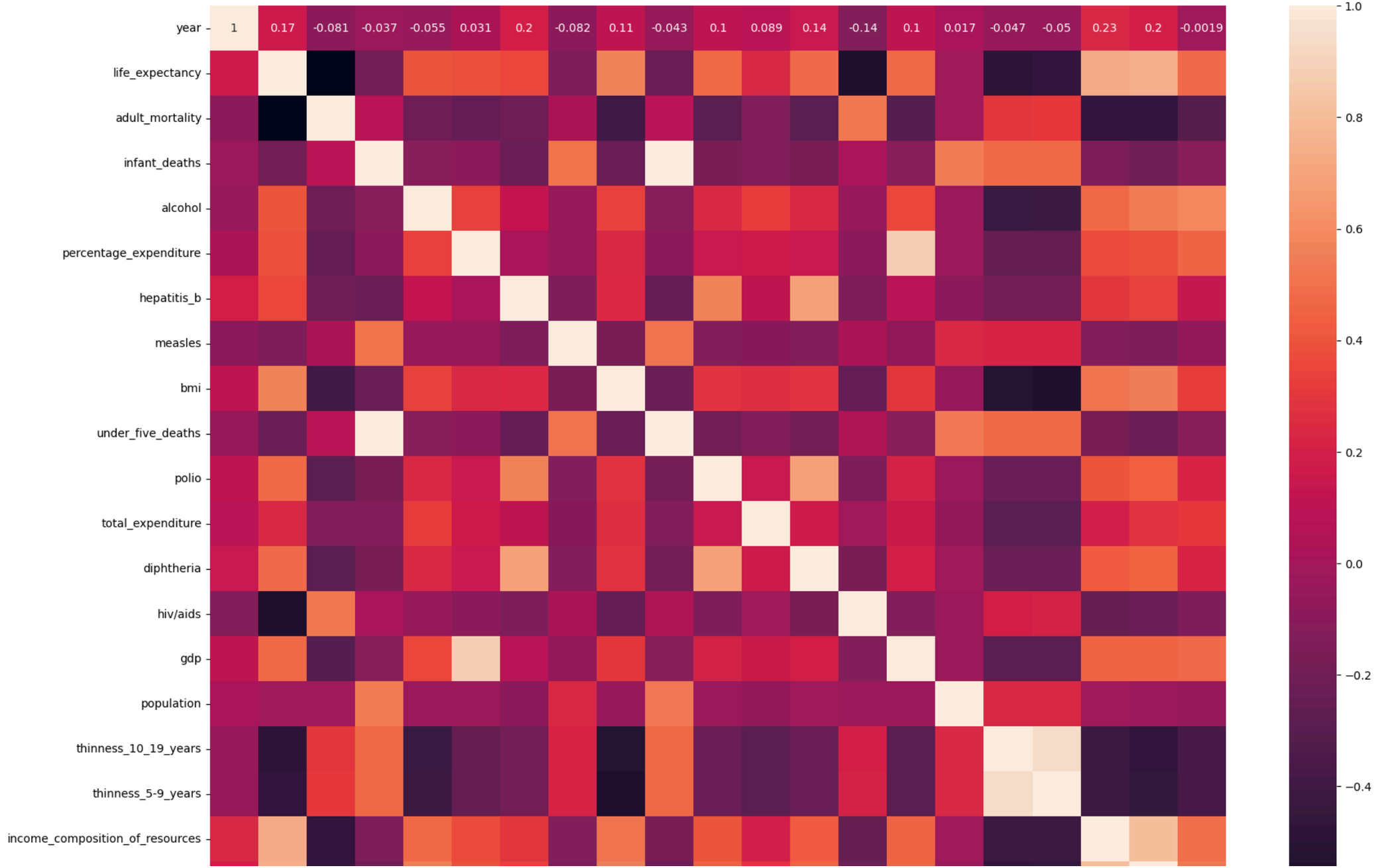

life_expectancy vs Life Expectancy
adult_mortality vs Life Expectancy
infant_deaths vs Life Expectancy
alcohol vs Life Expectancy
percentage_expenditure vs Life Expectancy
hepatitis_b vs Life Expectancy
measles vs Life Expectancy
bmi vs Life Expectancy
under_five_deaths vs Life Expectancy
polio vs Life Expectancy
total_expenditure vs Life Expectancy
diphtheria vs Life Expectancy
hiv/aids vs Life Expectancy
gdp vs Life Expectancy
population vs Life Expectancy
thinness_10_19_years vs Life Expectancy
thinness_5-9_years vs Life Expectancy
income_composition_of_resources vs Life Expectancy
schooling vs Life Expectancy
Life Expectancy
life_expectancy
adult_mortality
infant_deaths
alcohol
percentage_expenditure
hepatitis_b
measles
bmi
under_five_deaths
polio
total_expenditure
diphtheria
hiv/aids
gdp
population
1e9

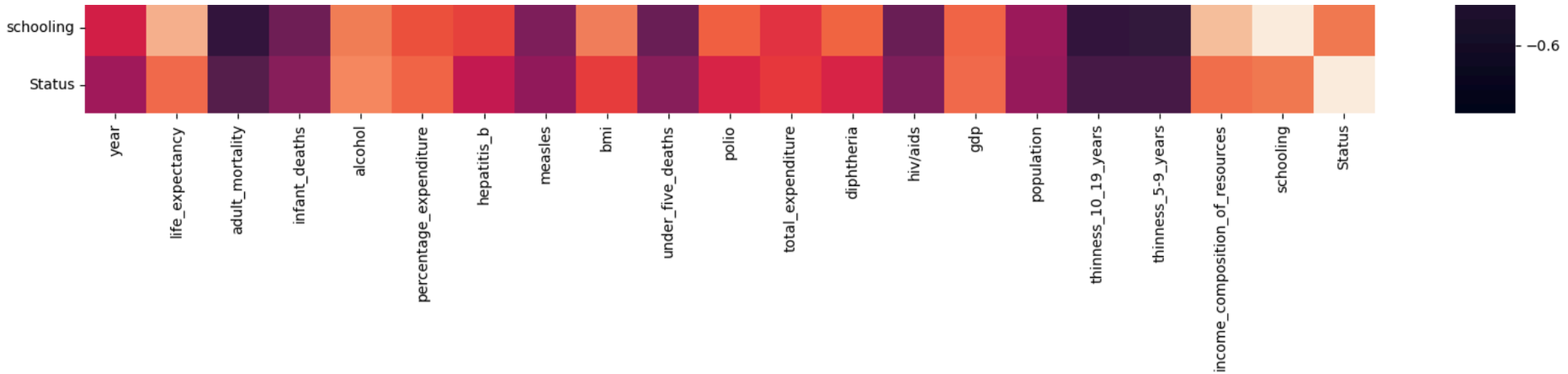



There is a clear linear relation between life expectany and other columns. So, linearity assumption is met.

## Model Assumption: No Multicollinearity

In [24]:
```
corr = df_imputed.drop(columns=['status', 'country']).corr()
plt.figure(figsize=(20, 15))
sns.heatmap(corr, annot=True)
plt.show()
```


year 1 0.17 -0.081 -0.037 -0.055 0.031 0.2 -0.082 0.11 -0.043 0.1 0.089 0.14 -0.14 0.1 0.017 -0.047 -0.05 0.23 0.2 -0.0019
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_10_19_years
thinness_5-9_years
income_composition_of_resources
1.0
0.8
0.6
0.4
0.2
0.0
−0.2
−0.4




Features such as `under_five_death` and `infant_deaths`, `gdp` and `percentage_expenditure`, `schooling` and `income_composition_of_resources`, `thinness_10_19_years` and `thinness_5-9_years`, and `polio` and `diphtheria` are collerated and therefore will not be used together.

Next, we will go ahead and start building our model by first running an algorithm iteratively to determine the features that results in the best score

## Multilinear Regression

In [25]:
```
# X = df_imputed.drop(columns=['life_expectancy', 'country', 'status', 'percentage_expenditure','income_composition_of_resources','under_five_deaths','thinne
# y = df_imputed['life_expectancy']

# # Initialize variables to store the best features and scores
# best_features = list(X.columns)
# best_score = -np.inf
# best_adj_r_squared = -np.inf

# # Iterate over features and remove one feature at a time
# for i in range(len(X.columns)):
#     # Exclude one feature
#     X_temp = X.drop(columns=[X.columns[i]])

#     # Split the data into training and testing sets
#     X_train, X_test, y_train, y_test = train_test_split(X_temp, y, test_size=0.2, random_state=42)

#     # Initialize and fit the Linear Regression model
```

```
#     model = LinearRegression()
#     model.fit(X_train, y_train)

#     # Predict the target variable on the testing set
#     y_pred = model.predict(X_test)

#     # Calculate R-squared
#     r_squared = r2_score(y_test, y_pred)

#     # Calculate adjusted R-squared
#     n = len(y_test)
#     p = X_test.shape[1]
#     adj_r_squared = 1 - (1 - r_squared) * (n - 1) / (n - p - 1)

#     # Check if current model is better than the best model so far
#     if adj_r_squared > best_adj_r_squared:
#         best_adj_r_squared = adj_r_squared
#         best_features = list(X_temp.columns)

# # Print the best features and scores
# print("Best Features:", best_features)
# print("Best Adjusted R-squared:", best_adj_r_squared)
```

In [26]:

```
X = df_imputed_win.drop(columns=['life_expectancy','country','status' ,'population','measles','gdp','under_five_deaths','polio'])
y = df_imputed_win['life_expectancy']
X_train_linear, X_test_linear, y_train_linear, y_test_linear = train_test_split(X, y, test_size=0.2, random_state=42)

model = LinearRegression()
model.fit(X_train_linear, y_train_linear)

y_pred_linear = model.predict(X_test_linear)

mse = mean_squared_error(y_test_linear, y_pred_linear)

n = len(y_test_linear)
p = X_test_linear.shape[1]
r_squared = r2_score(y_test_linear, y_pred_linear)
adjusted_r_squared = 1 - (1 - r_squared) * (n - 1) / (n - p - 1)

print("Mean Squared Error:", mse)
```

```
print("R-squared:", r_squared)
print("Adjusted R-squared:", adjusted_r_squared)
```

```
Mean Squared Error: 9.182071355569587
R-squared: 0.8833824134566242
Adjusted R-squared: 0.8803242599633538
```

## Polynomial Regression

In [27]:

```
# # Define the predictor variables (features) and target variable
# X = df_imputed.drop(columns=['life_expectancy','country','status' ,'population','measles','gdp'])  # Exclude the target column
# y = df_imputed['life_expectancy']

# # Create polynomial features of degree 2
# poly = PolynomialFeatures(degree=2, include_bias=False)
# X_poly = poly.fit_transform(X)

# # Initialize lists to store results
# best_features = list(X.columns)
# best_score = -np.inf

# # Iterate over features and remove one feature at a time
# for i in range(len(X_poly[0])):
#     # Exclude one feature
#     X_temp = np.delete(X_poly, i, axis=1)

#     # Split the data into training and testing sets
#     X_train, X_test, y_train, y_test = train_test_split(X_temp, y, test_size=0.2, random_state=42)

#     # Initialize and fit the Linear Regression model
#     model = LinearRegression()
#     model.fit(X_train, y_train)

#     # Predict the target variable on the testing set
#     y_pred = model.predict(X_test)

#     # Calculate R-squared
#     r_squared = r2_score(y_test, y_pred)

#     # Calculate adjusted R-squared
#     n = len(y_test)
```

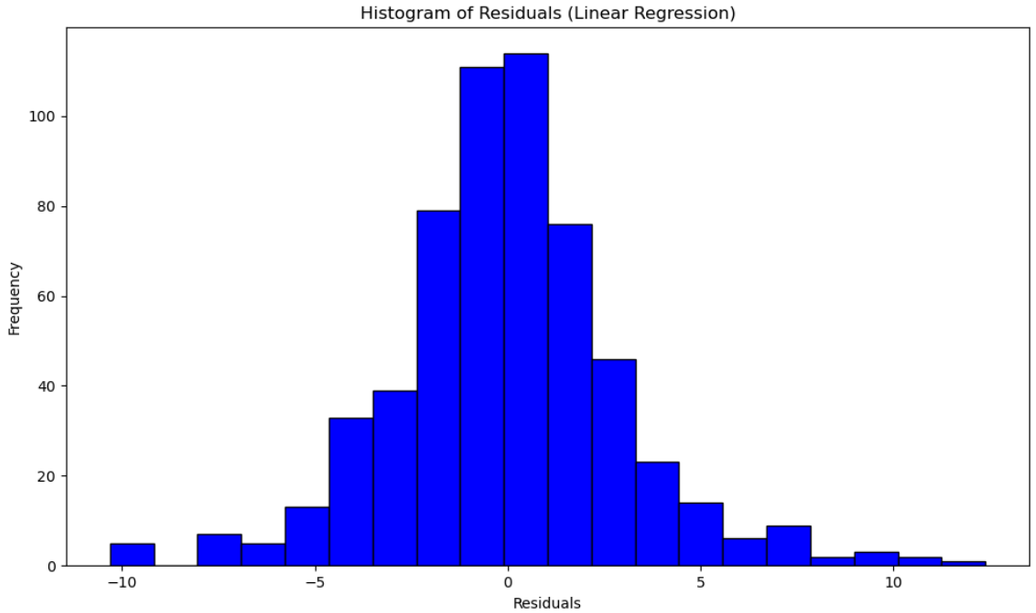
```
#     p = X_test.shape[1]
#     adjusted_r_squared = 1 - (1 - r_squared) * (n - 1) / (n - p - 1)

#     # Check if current model is better than the best model so far
#     if adjusted_r_squared > best_score:
#         best_score = adjusted_r_squared
#         best_features = [f for j, f in enumerate(X.columns) if j != i]

# # Print the best features
# print("Best Features:", best_features)
# print("Best Adjusted R-squared:", best_score)
```

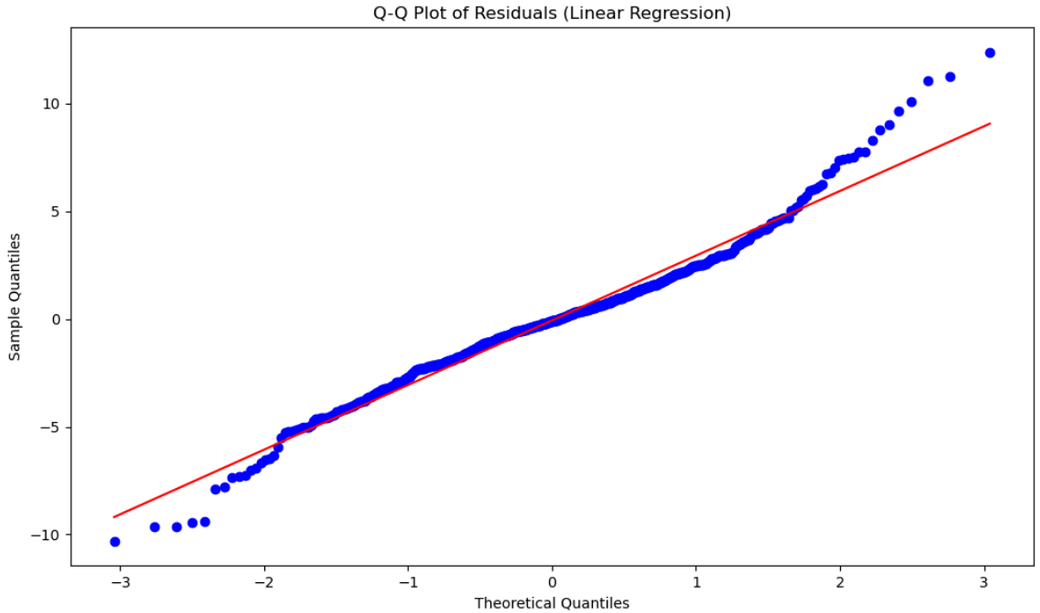
In [28]:
```
X = df_imputed_win.drop(columns=['life_expectancy','country','status' ,'population','measles','gdp','under_five_deaths','polio'])  # Exclude the target colum
y = df_imputed_win['life_expectancy']

poly = PolynomialFeatures(degree=2, include_bias=False)
X_poly = poly.fit_transform(X)

X_train_poly, X_test_poly, y_train_poly, y_test_poly = train_test_split(X_poly, y, test_size=0.2, random_state=42)

model = LinearRegression()
model.fit(X_train_poly, y_train_poly)

y_pred_poly = model.predict(X_test_poly)

mse = mean_squared_error(y_test_poly, y_pred_poly)

n = len(y_test_poly)
p = X_test_poly.shape[1]
r_squared = r2_score(y_test_poly, y_pred_poly)
adjusted_r_squared = 1 - (1 - r_squared) * (n - 1) / (n - p - 1)

print("Mean Squared Error:", mse)
print("R-squared:", r_squared)
print("Adjusted R-squared:", adjusted_r_squared)
```

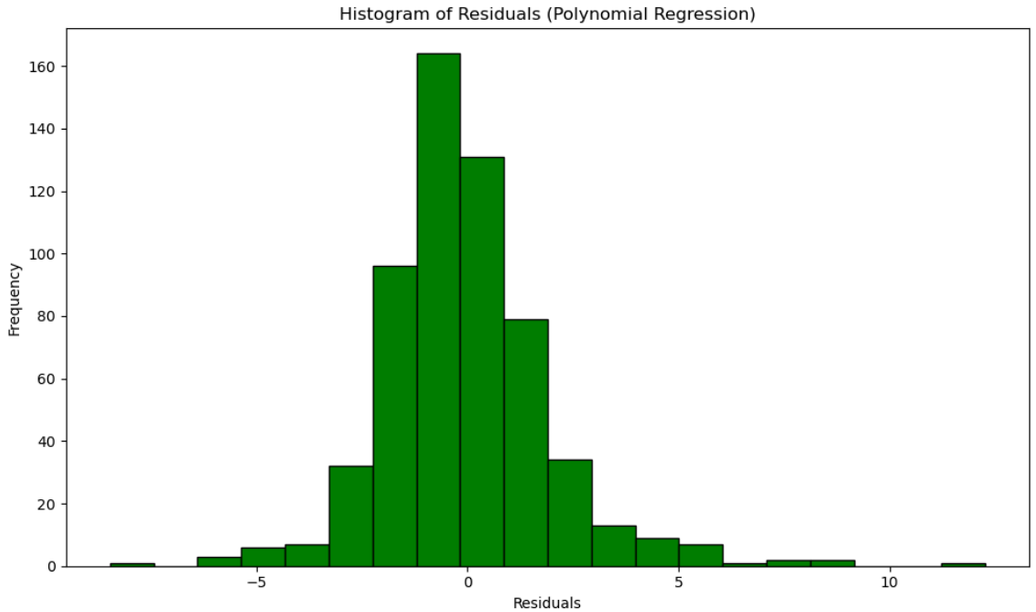
```
Mean Squared Error: 4.0723337672493205
R-squared: 0.9482790192816738
Adjusted R-squared: 0.9328313812352711
```

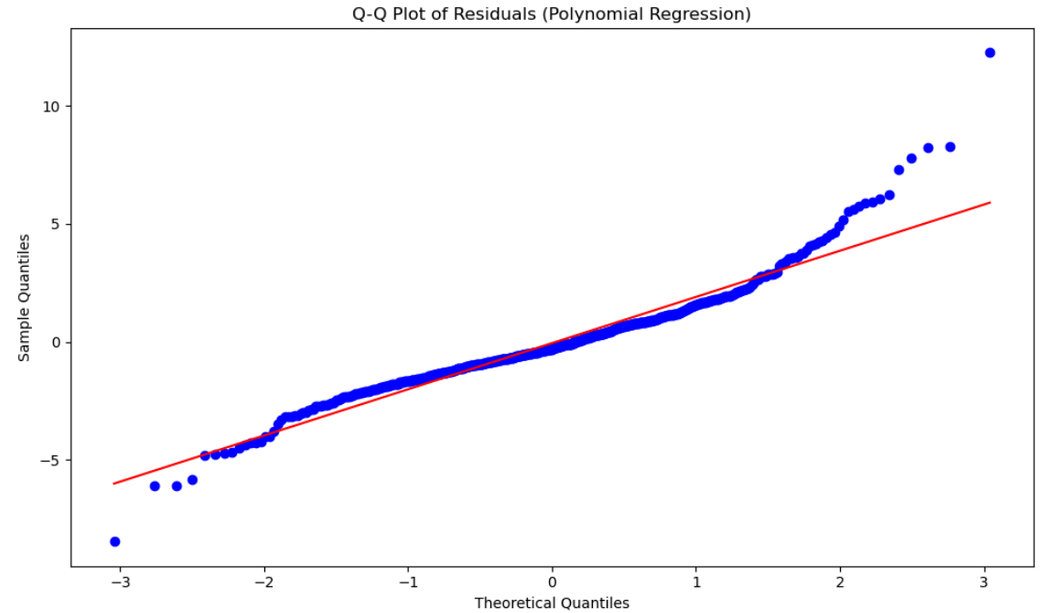
## Model Assumption: Normality

In [29]:
```
residuals_linear = y_test_linear - y_pred_linear

residuals_poly = y_test_poly - y_pred_poly

plt.figure(figsize=(20, 12))

# Linear Regression Plots
plt.subplot(2, 2, 1)
plt.hist(residuals_linear, bins=20, color='blue', edgecolor='black')
plt.title('Histogram of Residuals (Linear Regression)')
plt.xlabel('Residuals')
plt.ylabel('Frequency')

plt.subplot(2, 2, 2)
stats.probplot(residuals_linear, dist="norm", plot=plt)
plt.title('Q-Q Plot of Residuals (Linear Regression)')
plt.xlabel('Theoretical Quantiles')
plt.ylabel('Sample Quantiles')

# Polynomial Regression Plots
plt.subplot(2, 2, 3)
plt.hist(residuals_poly, bins=20, color='green', edgecolor='black')
plt.title('Histogram of Residuals (Polynomial Regression)')
plt.xlabel('Residuals')
plt.ylabel('Frequency')

plt.subplot(2, 2, 4)
stats.probplot(residuals_poly, dist="norm", plot=plt)
plt.title('Q-Q Plot of Residuals (Polynomial Regression)')
plt.xlabel('Theoretical Quantiles')
plt.ylabel('Sample Quantiles')

plt.tight_layout()
plt.show()
```


Histogram of Residuals (Linear Regression)
Frequency
Residuals
0
20
40
60
80
100
−10
−5
0
5
10


Q-Q Plot of Residuals (Linear Regression)
Sample Quantiles
Theoretical Quantiles
−10
−5
0
5
10
−3
−2
−1
0
1
2
3

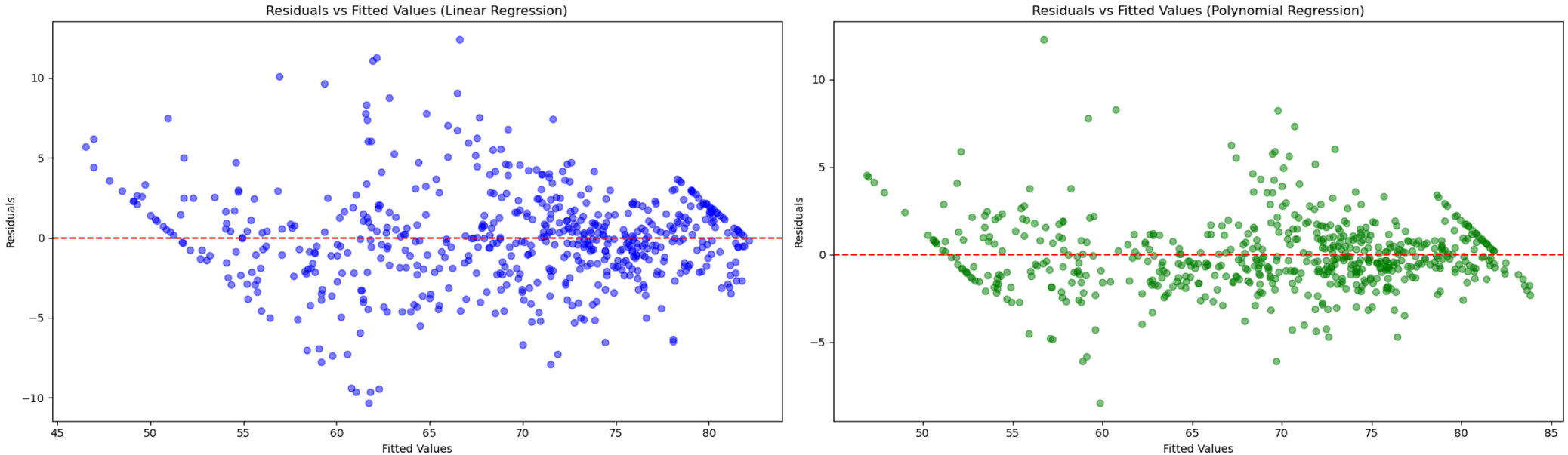

Histogram of Residuals (Polynomial Regression)
Frequency
Residuals
0
20
40
60
80
100
120
140
160
−5
0
5
10


Q-Q Plot of Residuals (Polynomial Regression)
Sample Quantiles
Theoretical Quantiles
−5
0
5
10
−3
−2
−1
0
1
2
3

We can see that for both the linear regression and polynomial regression the residuals are approximately following linear distributions also supported by their qq plots. Which in this case normality assumption is met.

## Model assumption: Homoscedasticity

In [30]:
```
# Linear regression
plt.figure(figsize=(20, 6))
plt.subplot(1, 2, 1)
plt.scatter(y_pred_linear, residuals_linear, color='blue', alpha=0.5)
plt.axhline(0, color='red', linestyle='--')
plt.title('Residuals vs Fitted Values (Linear Regression)')
plt.xlabel('Fitted Values')
plt.ylabel('Residuals')

# Polynomial regression
plt.subplot(1, 2, 2)
plt.scatter(y_pred_poly, residuals_poly, color='green', alpha=0.5)
plt.axhline(0, color='red', linestyle='--')
plt.title('Residuals vs Fitted Values (Polynomial Regression)')
plt.xlabel('Fitted Values')
plt.ylabel('Residuals')

plt.tight_layout()
plt.show()
```




We can see that in each model the variance of residuals is random and constant with no obvious patterns indicating that Homoscedasticity is met.

# Lazy Models

Now, let's suppose we took a more traditional approach in cleaning, not decreasing the effect f outliers, and not doing feature selction

In [31]:
```
df = pd.read_csv('Life Expectancy Data.csv')
df.drop(['Status', 'Country'], axis=1, inplace=True)
df_filled = df.copy()
df_filled = df.fillna(df.mean())

X = df_imputed_win.drop(columns=['life_expectancy', 'country', 'status'])
y = df_imputed_win['life_expectancy']

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

model = LinearRegression()
model.fit(X_train, y_train)
```

```
y_pred = model.predict(X_test)

mse = mean_squared_error(y_test, y_pred)
r_squared = r2_score(y_test, y_pred)
adjusted_r_squared = 1 - (1 - r_squared) * (len(y_test) - 1) / (len(y_test) - X_test.shape[1] - 1)

print("Mean Squared Error:", mse)
print("R-squared:", r_squared)
print("Adjusted R-squared:", adjusted_r_squared)
```

```
Mean Squared Error: 8.710349290712415
R-squared: 0.889373554953203
Adjusted R-squared: 0.8854713875794182
```

In [32]:
```
X = df_imputed_win.drop(columns=['life_expectancy', 'country', 'status'])
y = df_imputed_win['life_expectancy']

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

poly = PolynomialFeatures(degree=2, include_bias=False)
X_train_poly = poly.fit_transform(X_train)
X_test_poly = poly.transform(X_test)

model = LinearRegression()
model.fit(X_train_poly, y_train)

y_pred = model.predict(X_test_poly)

mse = mean_squared_error(y_test, y_pred)
r_squared = r2_score(y_test, y_pred)
adjusted_r_squared = 1 - (1 - r_squared) * (len(y_test) - 1) / (len(y_test) - X_test_poly.shape[1] - 1)

print("Mean Squared Error:", mse)
print("R-squared:", r_squared)
print("Adjusted R-squared:", adjusted_r_squared)
```

```
Mean Squared Error: 6.93942100464484
R-squared: 0.9118653625928079
Adjusted R-squared: 0.8550839435349531
```

# Lazy Models

### Linear Regression Results:

Mean Squared Error: 15.242936417691073

R-squared: 0.8240562394233331

Adjusted R-squared: 0.8178501103024629

### Polynomial Regression Results:

Mean Squared Error: 9.055163867039644

R-squared: 0.8954794837590591

Adjusted R-squared: 0.8281413360408059

## Chosen Models

### Linear Regression Results:

Mean Squared Error: 8.7920597474489

R-squared: 0.8883357851634719

Adjusted R-squared: 0.8848033495447416

### Polynomial Regression Results:

Mean Squared Error: 4.072333757853935

R-squared: 0.9482790194010007

Adjusted R-squared: 0.9328313813902376

- We can see that our "more advanced" techniques of handling missing data and outliers and feature selection caused our models performance to increase significantly compared to the lazy models in all aspects, as polynomial regression adjusted R-squared increased my more than 10% while the error decreased by a big amount.

- We can see that polynomial regression model outperformes the linear regression for this dataset.
- Both of our models have met the assumptions while partially violiting the independance assumption
- Polynomial regression explain 94.8% of the values using R squared while 93.3% using adjusted R squared
- High performance score on our test data indicates the success of our project goals in building an accurate stronge model
- As our performance is high there is no need to try to regualize the model using lasso or ridge regression, and polynomial regression is sufficient for our final model

# Conclusion

In conclusion, our study aimed to address gaps in previous research on life expectancy by developing a comprehensive regression model using data from 2000 to 2015 for 193 countries. By incorporating critical factors such as immunization, mortality, economy, and social determinants, we sought to provide insights into the complex interplay of factors influencing life expectancy.

Our analysis involved meticulous preprocessing of data, including handling missing values and merging datasets from multiple sources. Through exploratory data analysis (EDA), we gained a deeper understanding of the distributions and relationships among key variables.

Utilizing regression modeling techniques, we identified significant predictors of life expectancy while ensuring adherence to key assumptions. Feature selection and rigorous model evaluation enabled us to build robust models capable of accurately predicting life expectancy.

By leveraging accurate and reliable data from sources like the World Health Organization (WHO) and the United Nations, our study contributes to the growing body of knowledge on global public health. The insights gained can inform policymakers and healthcare practitioners in prioritizing interventions to improve life expectancy and overall population health outcomes.

We extend our gratitude to the WHO, the United Nations, and our collaborators for their invaluable contributions to this research endeavor.

Through collaborative efforts and continued research, we can work towards addressing health disparities and promoting equitable access to healthcare, ultimately striving for healthier and more prosperous communities worldwide.

Trend of Increasing Life Expectancy: There's a noticeable trend of increasing life expectancy over the years, suggesting a corresponding decline in mortality rates globally. Zimbabwe's Unique Situation: Zimbabwe stands out with the highest adult mortality rate among the listed countries, indicating significant health challenges. Despite this, its life expectancy is not the lowest, suggesting other factors at play. Impact of Diseases on Life Expectancy: Measles has a relatively minor impact on life expectancy compared to HIV/AIDS, which shows a strong negative correlation with life expectancy. Polio and hepatitis B show a weaker correlation. This indicates the significant impact of HIV/AIDS prevalence on life expectancy. Disparity Between Developed and Developing Countries: Developed countries generally exhibit higher life expectancy due to better living conditions compared to developing nations. Positive Correlation Between BMI and Life Expectancy: There's a robust and strong positive correlation between Body Mass Index

(BMI) and life expectancy, suggesting that higher BMI is associated with longer life expectancy. Distribution Characteristics: The distribution of the data appears to be almost normal but with a negative skew, indicating that while the majority of countries may exhibit increasing life expectancy, there are outliers with lower life expectancies affecting the overall distribution.

In [ ]: