Life Expectancy (WHO) Statistical Analysis on factors influencing Life Expectancy Author: Vrushali Sudha

The Scope of this study is to enable countries to determine the predicting factors which are contributing to lower value of life expectancy by taking into account the immunization factors, mortality factors, economic factors, social factors and other health related factors as well. This will help in suggesting which area should be given importance in order to efficiently improve the life expectancy of its population.

Introduction About the Dataset

The dataset is related to life expectancy. Health factors for 193 countries have been collected from the same WHO data repository website and the 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. The dataset consists of 22 Columns and 2938 rows. The predictor variables are divided into several broad categories: Immunization related factors, Mortality factors, Economical factors and Social factors.

Response variable: Life Expectancy

The predictor variables include:

Country: Name of the county- (Categorical) Year: Year the information was retrieved (Quantitative) Status: Developing or developed status (Categorical) Life Expectancy: Life Expectancy in Age (Quantitative) Adult Mortality: Adult Mortality Rates of both sexes (probability of dying between 15 and 60 years per 1000 population) (Quantitative) Infant Deaths: Number of Infant Deaths per 1000 population (Quantitative) Alcohol: Alcohol, recorded per capita (15+) consumption (in litres of pure alcohol) (Quantitative) Percentage Expenditure: Expenditure on health as a percentage of Gross Domestic Product per capita(%) (Quantitative)) Hepatitis B: Hepatitis B (HepB) immunization coverage among 1-year-olds (%)- (Quantitative) Measles: Number of reported cases per 1000 population (Quantitative) BMI: Average Body Mass Index of entire population (Quantitative) Under-Five Deaths: Number of under-five deaths per 1000 population (Quantitative) Polio: Polio (Pol3) immunization coverage among 1-year-olds (%) (Quantitative)) Total Expenditure:General government expenditure on health as a percentage of total government expenditure (%) (Quantitative)) Diphtheria: Diphtheria tetanus toxoid and pertussis (DTP3) immunization coverage among 1-year-olds (%) (Quantitative)) HIV/AIDS: Deaths per 1000 live births HIV/AIDS (0-4 years) (Quantitative)) GDP: Gross Domestic Product per capita (in USD) (Quantitative) Population: Population of the country (Quantitative) Thinness 1-19 Years: Prevalence of thinness among children and adolescents for Age 10 to 19 (%) (Quantitative) Thinness 5-9 Years: Prevalence of thinness among children for Age 5 to 9(%) (Quantitative) Income Composition of Resources: Human Development Index in terms of income composition of resources (index ranging from 0 to 1) (Quantitative) Schooling: Number of years of Schooling(years) (Quantitative)

In [1]:
```
 # Importing standard packages
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from matplotlib import style
style.use('ggplot')
import seaborn as sns
%matplotlib inline
import matplotlib.pyplot as plt
import statsmodels.api as sm
import scipy.stats as stats
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
from sklearn.metrics import r2_score, mean_absolute_error, mean_squared_error
from sklearn.feature_selection import RFE
from sklearn.preprocessing import RobustScaler
from sklearn.dummy import DummyRegressor
from sklearn.neighbors import LocalOutlierFactor
from sklearn.preprocessing import OneHotEncoder
from sklearn.linear_model import Ridge
from sklearn.model_selection import train_test_split
#from yellowbrick.regressor import ResidualsPlot
from sklearn.preprocessing import FunctionTransformer
sns.set_theme(style="whitegrid")
pd.options.display.float_format = '{:.2f}'.format
```

In [2]:
```
# Importing the Kings County House Data
df_data = pd.read_csv('C:/Users/vrush/Documents/Life ExpectancyKaggleWHO.csv')
df_data.head()
```

Out[2]:

	Country	Year	Status	Life expectancy	Adult Mortality	infant deaths	Alcohol	percentage expenditure	Hepatitis B	Measles	...	Polio	Total expenditure	Diphtheria	HIV/AIDS	G
0	Afghanistan	2015	Developing	65.00	263.00	62	0.01	71.28	65.00	1154	...	6.00	8.16	65.00	0.10	58
1	Afghanistan	2014	Developing	59.90	271.00	64	0.01	73.52	62.00	492	...	58.00	8.18	62.00	0.10	61
2	Afghanistan	2013	Developing	59.90	268.00	66	0.01	73.22	64.00	430	...	62.00	8.13	64.00	0.10	63
3	Afghanistan	2012	Developing	59.50	272.00	69	0.01	78.18	67.00	2787	...	67.00	8.52	67.00	0.10	66
4	Afghanistan	2011	Developing	59.20	275.00	71	0.01	7.10	68.00	3013	...	68.00	7.87	68.00	0.10	6

5 rows × 22 columns

In [3]:
```
df_data.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 [4]:
```
df_data.describe().T
```

Out[4]:

	count	mean	std	min	25%	50%	75%	max
Year	2938.00	2007.52	4.61	2000.00	2004.00	2008.00	2012.00	2015.00
Life expectancy	2928.00	69.22	9.52	36.30	63.10	72.10	75.70	89.00
Adult Mortality	2928.00	164.80	124.29	1.00	74.00	144.00	228.00	723.00
infant deaths	2938.00	30.30	117.93	0.00	0.00	3.00	22.00	1800.00
Alcohol	2744.00	4.60	4.05	0.01	0.88	3.75	7.70	17.87
percentage expenditure	2938.00	738.25	1987.91	0.00	4.69	64.91	441.53	19479.91
Hepatitis B	2385.00	80.94	25.07	1.00	77.00	92.00	97.00	99.00
Measles	2938.00	2419.59	11467.27	0.00	0.00	17.00	360.25	212183.00
BMI	2904.00	38.32	20.04	1.00	19.30	43.50	56.20	87.30
under-five deaths	2938.00	42.04	160.45	0.00	0.00	4.00	28.00	2500.00
Polio	2919.00	82.55	23.43	3.00	78.00	93.00	97.00	99.00
Total expenditure	2712.00	5.94	2.50	0.37	4.26	5.75	7.49	17.60
Diphtheria	2919.00	82.32	23.72	2.00	78.00	93.00	97.00	99.00
HIV/AIDS	2938.00	1.74	5.08	0.10	0.10	0.10	0.80	50.60
GDP	2490.00	7483.16	14270.17	1.68	463.94	1766.95	5910.81	119172.74
Population	2286.00	12753375.12	61012096.51	34.00	195793.25	1386542.00	7420359.00	1293859294.00
thinness 1-19 years	2904.00	4.84	4.42	0.10	1.60	3.30	7.20	27.70
thinness 5-9 years	2904.00	4.87	4.51	0.10	1.50	3.30	7.20	28.60
Income composition of resources	2771.00	0.63	0.21	0.00	0.49	0.68	0.78	0.95
Schooling	2775.00	11.99	3.36	0.00	10.10	12.30	14.30	20.70

In [5]:
```
# First, I will drop "Country" and "Year" columns.
df_data = df_data.drop(['Country','Year'], axis=1)
df_data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 2938 entries, 0 to 2937
Data columns (total 20 columns):
 #   Column                           Non-Null Count  Dtype  
---  ------                           --------------  -----  
 0   Status                           2938 non-null   object 
 1   Life expectancy                  2928 non-null   float64
 2   Adult Mortality                  2928 non-null   float64
 3   infant deaths                    2938 non-null   int64  
 4   Alcohol                          2744 non-null   float64
 5   percentage expenditure           2938 non-null   float64
 6   Hepatitis B                      2385 non-null   float64
 7   Measles                          2938 non-null   int64  
 8    BMI                             2904 non-null   float64
 9   under-five deaths                2938 non-null   int64  
 10  Polio                            2919 non-null   float64
 11  Total expenditure                2712 non-null   float64
 12  Diphtheria                       2919 non-null   float64
 13   HIV/AIDS                        2938 non-null   float64
 14  GDP                              2490 non-null   float64
 15  Population                       2286 non-null   float64
 16   thinness  1-19 years            2904 non-null   float64
 17   thinness 5-9 years              2904 non-null   float64
 18  Income composition of resources  2771 non-null   float64
 19  Schooling                        2775 non-null   float64
dtypes: float64(16), int64(3), object(1)
memory usage: 459.2+ KB
```

In [6]:
```
df_data.isna().sum()
```

Out[6]:
```
Status                             0
Life expectancy                   10
Adult Mortality                   10
infant deaths                      0
Alcohol                          194
percentage expenditure             0
Hepatitis B                      553
Measles                            0
 BMI                              34
under-five deaths                  0
Polio                             19
Total expenditure                226
Diphtheria                        19
 HIV/AIDS                          0
GDP                              448
Population                       652
 thinness  1-19 years             34
 thinness 5-9 years               34
Income composition of resources  167
Schooling                        163
dtype: int64
```

In [7]:
```
df_data.isna().sum().sum()
```

Out[7]:
```
2563
```

In [8]:
```
new_df = df_data.dropna(axis=0)
```

In [9]:
```
new_df.isna().sum()
```

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

In [10]:
```
new_df.isna().sum().sum()
```

Out[10]:
```
0
```

In [11]:
```
new_df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 1649 entries, 0 to 2937
Data columns (total 20 columns):
 #   Column                           Non-Null Count  Dtype  
---  ------                           --------------  -----  
 0   Status                           1649 non-null   object 
 1   Life expectancy                  1649 non-null   float64
 2   Adult Mortality                  1649 non-null   float64
 3   infant deaths                    1649 non-null   int64  
 4   Alcohol                          1649 non-null   float64
 5   percentage expenditure           1649 non-null   float64
 6   Hepatitis B                      1649 non-null   float64
 7   Measles                          1649 non-null   int64  
 8    BMI                             1649 non-null   float64
 9   under-five deaths                1649 non-null   int64  
 10  Polio                            1649 non-null   float64
 11  Total expenditure                1649 non-null   float64
 12  Diphtheria                       1649 non-null   float64
 13   HIV/AIDS                        1649 non-null   float64
 14  GDP                              1649 non-null   float64
 15  Population                       1649 non-null   float64
 16   thinness  1-19 years            1649 non-null   float64
 17   thinness 5-9 years              1649 non-null   float64
 18  Income composition of resources  1649 non-null   float64
 19  Schooling                        1649 non-null   float64
dtypes: float64(16), int64(3), object(1)
memory usage: 270.5+ KB
```

In [12]:
```
new_df.head()
```

Out[12]:

	Status	Life expectancy	Adult Mortality	infant deaths	Alcohol	percentage expenditure	Hepatitis B	Measles	BMI	under-five deaths	Polio	Total expenditure	Diphtheria	HIV/AIDS	GDP	
0	Developing	65.00	263.00	62	0.01	71.28	65.00	1154	19.10	83	6.00	8.16	65.00	0.10	584.26	3
1	Developing	59.90	271.00	64	0.01	73.52	62.00	492	18.60	86	58.00	8.18	62.00	0.10	612.70	
2	Developing	59.90	268.00	66	0.01	73.22	64.00	430	18.10	89	62.00	8.13	64.00	0.10	631.74	3
3	Developing	59.50	272.00	69	0.01	78.18	67.00	2787	17.60	93	67.00	8.52	67.00	0.10	669.96	
4	Developing	59.20	275.00	71	0.01	7.10	68.00	3013	17.20	97	68.00	7.87	68.00	0.10	63.54	

In [13]:
```
new_df.describe().T
```

Out[13]:

	count	mean	std	min	25%	50%	75%	max
Life expectancy	1649.00	69.30	8.80	44.00	64.40	71.70	75.00	89.00
Adult Mortality	1649.00	168.22	125.31	1.00	77.00	148.00	227.00	723.00
infant deaths	1649.00	32.55	120.85	0.00	1.00	3.00	22.00	1600.00
Alcohol	1649.00	4.53	4.03	0.01	0.81	3.79	7.34	17.87
percentage expenditure	1649.00	698.97	1759.23	0.00	37.44	145.10	509.39	18961.35
Hepatitis B	1649.00	79.22	25.60	2.00	74.00	89.00	96.00	99.00
Measles	1649.00	2224.49	10085.80	0.00	0.00	15.00	373.00	131441.00
BMI	1649.00	38.13	19.75	2.00	19.50	43.70	55.80	77.10
under-five deaths	1649.00	44.22	162.90	0.00	1.00	4.00	29.00	2100.00
Polio	1649.00	83.56	22.45	3.00	81.00	93.00	97.00	99.00
Total expenditure	1649.00	5.96	2.30	0.74	4.41	5.84	7.47	14.39
Diphtheria	1649.00	84.16	21.58	2.00	82.00	92.00	97.00	99.00
HIV/AIDS	1649.00	1.98	6.03	0.10	0.10	0.10	0.70	50.60
GDP	1649.00	5566.03	11475.90	1.68	462.15	1592.57	4718.51	119172.74
Population	1649.00	14653625.89	70460393.40	34.00	191897.00	1419631.00	7658972.00	1293859294.00
thinness 1-19 years	1649.00	4.85	4.60	0.10	1.60	3.00	7.10	27.20
thinness 5-9 years	1649.00	4.91	4.65	0.10	1.70	3.20	7.10	28.20
Income composition of resources	1649.00	0.63	0.18	0.00	0.51	0.67	0.75	0.94
Schooling	1649.00	12.12	2.80	4.20	10.30	12.30	14.00	20.70

In [14]:
```
new_df['Status'].value_counts()
```

Out[14]:
```
Developing    1407
Developed      242
Name: Status, dtype: int64
```

Multicollinearity of Features Before diving into the modelling, I will check the multicollinearity between my numerical values to decide which ones shouldn't be used.

In [15]:
```
df_num = new_df.drop(['Status','Life expectancy '], axis=1,)
df_num.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 1649 entries, 0 to 2937
Data columns (total 18 columns):
 #   Column                           Non-Null Count  Dtype  
---  ------                           --------------  -----  
 0   Adult Mortality                  1649 non-null   float64
 1   infant deaths                    1649 non-null   int64  
 2   Alcohol                          1649 non-null   float64
 3   percentage expenditure           1649 non-null   float64
 4   Hepatitis B                      1649 non-null   float64
 5   Measles                          1649 non-null   int64  
 6    BMI                             1649 non-null   float64
 7   under-five deaths                1649 non-null   int64  
 8   Polio                            1649 non-null   float64
 9   Total expenditure                1649 non-null   float64
 10  Diphtheria                       1649 non-null   float64
 11   HIV/AIDS                        1649 non-null   float64
 12  GDP                              1649 non-null   float64
 13  Population                       1649 non-null   float64
 14   thinness  1-19 years            1649 non-null   float64
 15   thinness 5-9 years              1649 non-null   float64
 16  Income composition of resources  1649 non-null   float64
 17  Schooling                        1649 non-null   float64
dtypes: float64(15), int64(3)
memory usage: 244.8 KB
```

In [16]:
```
column_names = new_df.columns.values.tolist()
```

In [17]:
```
column_names
```

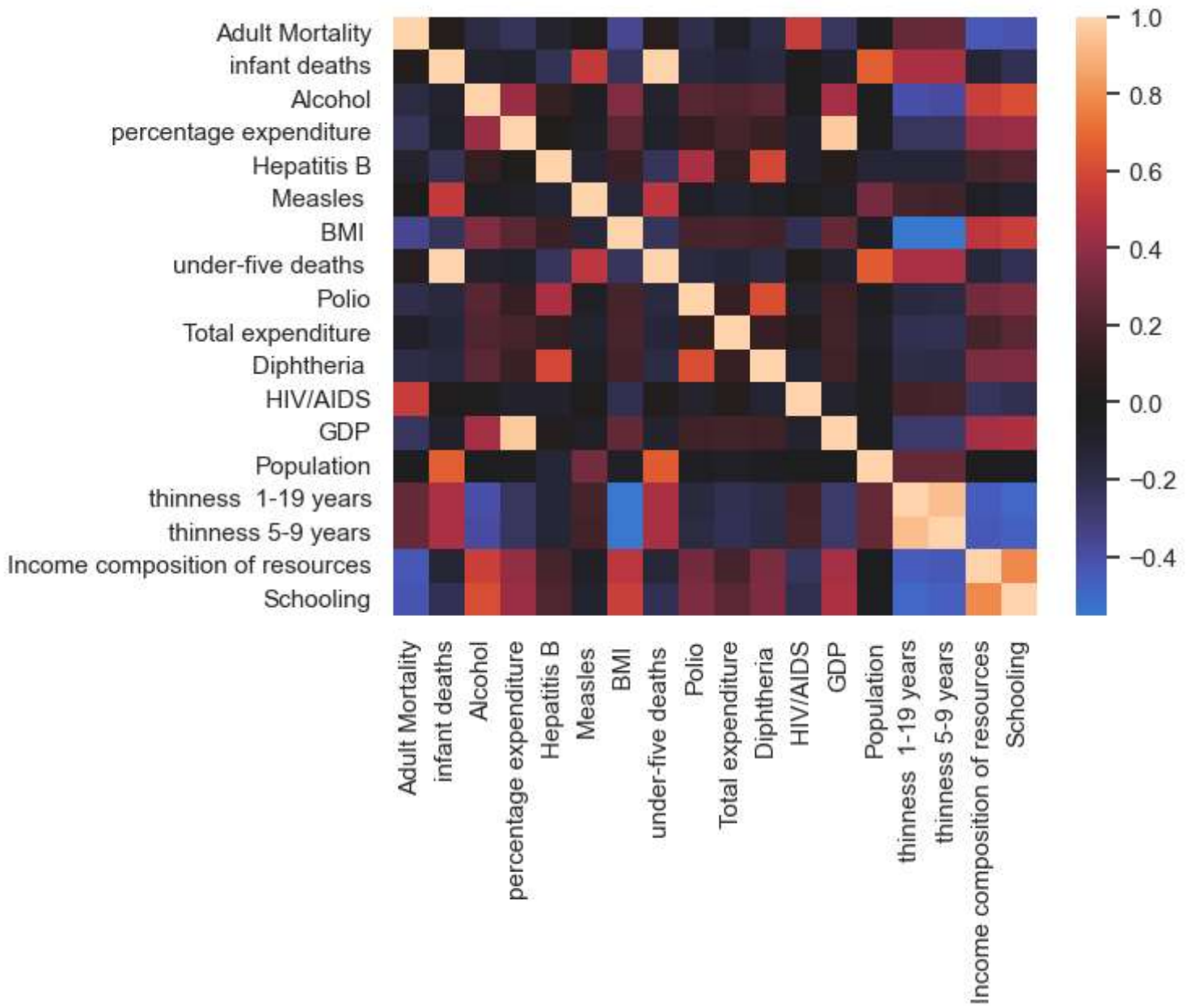
Out[17]:
```
['Status',
 'Life expectancy ',
 'Adult Mortality',
 'infant deaths',
 'Alcohol',
 'percentage expenditure',
 'Hepatitis B',
 'Measles ',
 ' BMI ',
 'under-five deaths ',
 'Polio',
 'Total expenditure',
 'Diphtheria ',
 ' HIV/AIDS',
 'GDP',
 'Population',
 ' thinness  1-19 years',
 ' thinness 5-9 years',
 'Income composition of resources',
 'Schooling']
```

In [18]:
```
df_num.head()
```

Out[18]:

	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	thi
0	263.00	62	0.01	71.28	65.00	1154	19.10	83	6.00	8.16	65.00	0.10	584.26	33736494.00	17.20	
1	271.00	64	0.01	73.52	62.00	492	18.60	86	58.00	8.18	62.00	0.10	612.70	327582.00	17.50	
2	268.00	66	0.01	73.22	64.00	430	18.10	89	62.00	8.13	64.00	0.10	631.74	31731688.00	17.70	
3	272.00	69	0.01	78.18	67.00	2787	17.60	93	67.00	8.52	67.00	0.10	669.96	3696958.00	17.90	
4	275.00	71	0.01	7.10	68.00	3013	17.20	97	68.00	7.87	68.00	0.10	63.54	2978599.00	18.20	

In [19]:
```
# Creating a heatmap of the independent variables
sns.heatmap(df_num.corr(), center=0);
```



In [20]:
```
# Creating a table that will show the highly correlated features
# Correlation greater than 0.75)
df = df_num.corr().abs().stack().reset_index()
df.columns = ['feature1', 'feature2', 'corr']
df[(df['corr']>.75) & (df['corr'] <1)]
```

Out[20]:

	feature1	feature2	corr
25	infant deaths	under-five deaths	1.00
66	percentage expenditure	GDP	0.96
127	under-five deaths	infant deaths	1.00
219	GDP	percentage expenditure	0.96
267	thinness 1-19 years	thinness 5-9 years	0.93
284	thinness 5-9 years	thinness 1-19 years	0.93
305	Income composition of resources	Schooling	0.78
322	Schooling	Income composition of resources	0.78

In [21]:
```
# I will create a new dataframe without the features I'm excluding.
# I'm not simply dropping the features because I may need to use them in the future.
data_final = new_df.drop(['infant deaths','percentage expenditure',' thinness 5-9 years','Income composition of resources',]
# Displaying the new dataframe
data_final.head()
```

Out[21]:

	Status	Life expectancy	Adult Mortality	Alcohol	Hepatitis B	Measles	BMI	under-five deaths	Polio	Total expenditure	Diphtheria	HIV/AIDS	GDP	Population	thinness 1-19 years
0	Developing	65.00	263.00	0.01	65.00	1154	19.10	83	6.00	8.16	65.00	0.10	584.26	33736494.00	17.20
1	Developing	59.90	271.00	0.01	62.00	492	18.60	86	58.00	8.18	62.00	0.10	612.70	327582.00	17.50
2	Developing	59.90	268.00	0.01	64.00	430	18.10	89	62.00	8.13	64.00	0.10	631.74	31731688.00	17.70
3	Developing	59.50	272.00	0.01	67.00	2787	17.60	93	67.00	8.52	67.00	0.10	669.96	3696958.00	17.90
4	Developing	59.20	275.00	0.01	68.00	3013	17.20	97	68.00	7.87	68.00	0.10	63.54	2978599.00	18.20

In [22]:
```
data_final.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 1649 entries, 0 to 2937
Data columns (total 16 columns):
 #   Column                 Non-Null Count  Dtype  
---  ------                 --------------  -----  
 0   Status                 1649 non-null   object 
 1   Life expectancy        1649 non-null   float64
 2   Adult Mortality        1649 non-null   float64
 3   Alcohol                1649 non-null   float64
 4   Hepatitis B            1649 non-null   float64
 5   Measles                1649 non-null   int64  
 6    BMI                   1649 non-null   float64
 7   under-five deaths      1649 non-null   int64  
 8   Polio                  1649 non-null   float64
 9   Total expenditure      1649 non-null   float64
 10  Diphtheria             1649 non-null   float64
 11   HIV/AIDS              1649 non-null   float64
 12  GDP                    1649 non-null   float64
 13  Population             1649 non-null   float64
 14   thinness  1-19 years  1649 non-null   float64
 15  Schooling              1649 non-null   float64
dtypes: float64(13), int64(2), object(1)
memory usage: 219.0+ KB
```

In [23]:
```
column_names = data_final.columns.values.tolist()
column_names
```

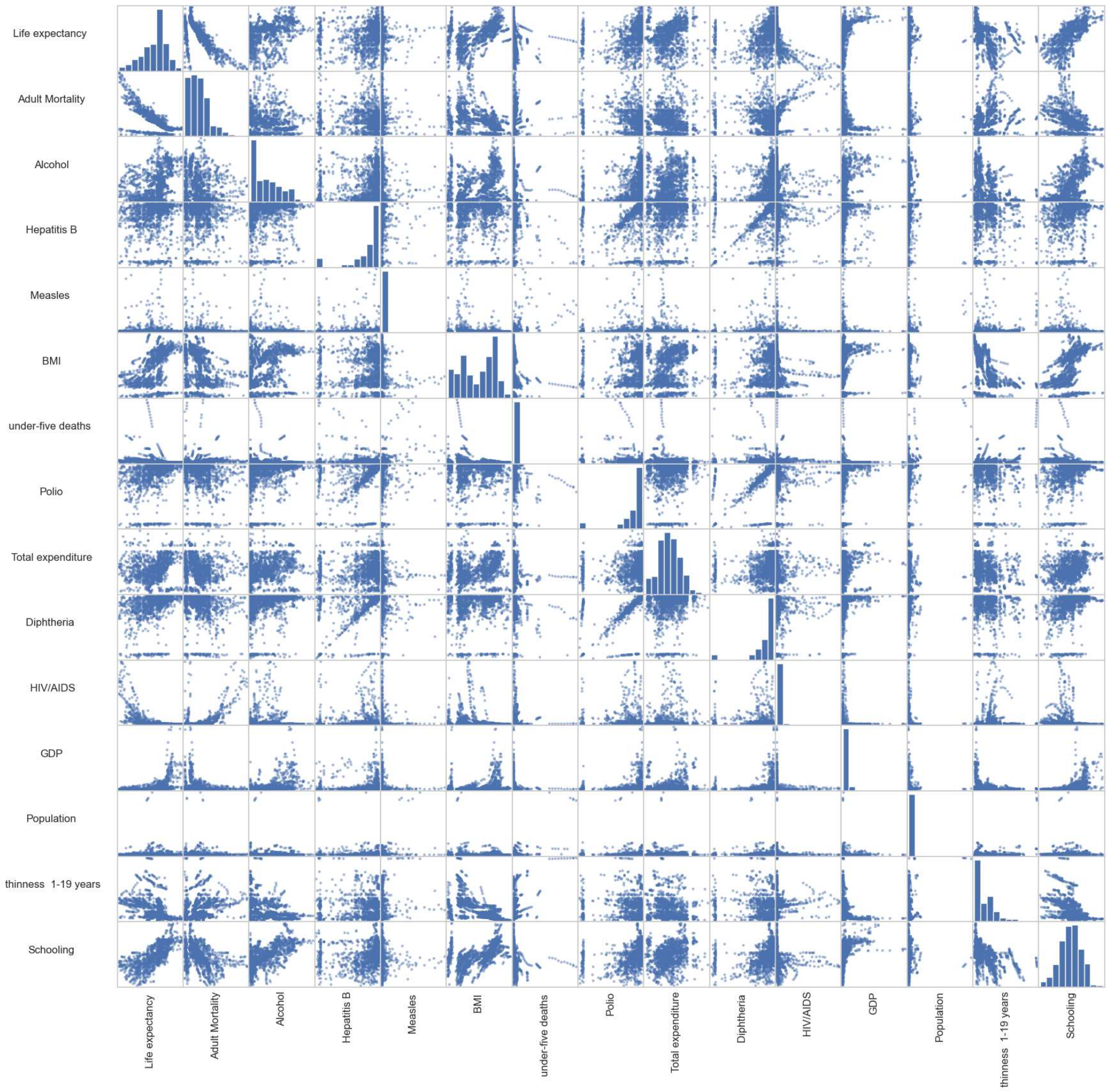
Out[23]:
```
['Status',
 'Life expectancy ',
 'Adult Mortality',
 'Alcohol',
 'Hepatitis B',
 'Measles ',
 ' BMI ',
 'under-five deaths ',
 'Polio',
 'Total expenditure',
 'Diphtheria ',
 ' HIV/AIDS',
 'GDP',
 'Population',
 ' thinness  1-19 years',
 'Schooling']
```

In [24]:
```
data_final['Measles '] = data_final['Measles '].astype('float')
```

In [25]:
```
data_final.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 1649 entries, 0 to 2937
Data columns (total 16 columns):
 #   Column                 Non-Null Count  Dtype  
---  ------                 --------------  -----  
 0   Status                 1649 non-null   object 
 1   Life expectancy        1649 non-null   float64
 2   Adult Mortality        1649 non-null   float64
 3   Alcohol                1649 non-null   float64
 4   Hepatitis B            1649 non-null   float64
 5   Measles                1649 non-null   float64
 6    BMI                   1649 non-null   float64
 7   under-five deaths      1649 non-null   int64  
 8   Polio                  1649 non-null   float64
 9   Total expenditure      1649 non-null   float64
 10  Diphtheria             1649 non-null   float64
 11   HIV/AIDS              1649 non-null   float64
 12  GDP                    1649 non-null   float64
 13  Population             1649 non-null   float64
 14   thinness  1-19 years  1649 non-null   float64
 15  Schooling              1649 non-null   float64
dtypes: float64(14), int64(1), object(1)
memory usage: 219.0+ KB
```

In [26]:
```
sm = pd.plotting.scatter_matrix(data_final, figsize=[20, 20]);

# Rotates the text
[s.xaxis.label.set_rotation(90) for s in sm.reshape(-1)]
[s.yaxis.label.set_rotation(0) for s in sm.reshape(-1)]

#May need to offset label when rotating to prevent overlap of figure
[s.get_yaxis().set_label_coords(-1,0.5) for s in sm.reshape(-1)]

#Hide all ticks
[s.set_xticks(()) for s in sm.reshape(-1)]
[s.set_yticks(()) for s in sm.reshape(-1)]

plt.show()
```



From a first glance, I can see that none of the columns have a strong linear relationship with price. Because I'm running a MLR model, I will transform the features iteratively

In [27]:
```
data_final['Adult Mortality'].value_counts()
```

Out[27]:
```
14.00     18
12.00     18
22.00     15
144.00    15
127.00    15
          ..
432.00     1
356.00     1
317.00     1
292.00     1
665.00     1
Name: Adult Mortality, Length: 369, dtype: int64
```

In [28]:
```
data_final['Alcohol'].value_counts()
```

Out[28]:
```
0.01    183
0.49      9
0.55      8
0.28      7
1.29      7
        ...
2.72      1
3.74      1
4.70      1
7.17      1
4.06      1
Name: Alcohol, Length: 833, dtype: int64
```

In [29]:
```
data_final['Hepatitis B' ].value_counts()
```

Out[29]:
```
98.00    137
95.00    115
96.00    111
99.00    109
97.00     92
         ...
38.00      1
26.00      1
23.00      1
15.00      1
21.00      1
Name: Hepatitis B, Length: 83, dtype: int64
```

In [30]:
```
data_final['Measles ' ].value_counts()
```

Out[30]:
```
0.00       554
1.00        73
2.00        39
3.00        28
6.00        18
           ...
6024.00      1
4657.00      1
779.00       1
778.00       1
1483.00      1
Name: Measles , Length: 603, dtype: int64
```

In [31]:
```
data_final[' BMI ' ].value_counts()
```

Out[31]:
```
55.70    10
58.50    10
57.00    10
57.20    10
47.90     9
         ..
36.30     1
33.40     1
29.80     1
27.00     1
3.30      1
Name:  BMI , Length: 538, dtype: int64
```

In [32]:
```
data_final['under-five deaths '].value_counts()
```

Out[32]:
```
0      353
1      230
2      115
3       93
4       90
      ...
256      1
243      1
221      1
211      1
164      1
Name: under-five deaths , Length: 199, dtype: int64
```

In [33]:
```
data_final['Polio'].value_counts()
```

Out[33]:
```
99.00    186
98.00    144
96.00    128
97.00    114
95.00    110
        ...
3.00       1
32.00      1
43.00      1
53.00      1
26.00      1
Name: Polio, Length: 68, dtype: int64
```

In [34]:
```
data_final['Total expenditure'].value_counts()
```

Out[34]:
```
4.60    12
5.30     8
6.70     8
5.90     8
5.92     8
        ..
2.81     1
2.37     1
2.53     1
2.27     1
6.16     1
Name: Total expenditure, Length: 669, dtype: int64
```

In [35]:
```
data_final['Diphtheria '].value_counts()
```

Out[35]:
```
99.00    180
98.00    146
95.00    121
96.00    117
97.00    110
        ...
51.00      1
2.00       1
19.00      1
41.00      1
52.00      1
Name: Diphtheria , Length: 66, dtype: int64
```

In [36]:
```
data_final[' HIV/AIDS'].value_counts()
```

Out[36]:
```
0.10     964
0.30      82
0.20      66
0.40      51
0.50      33
         ...
8.30       1
33.80      1
34.80      1
34.10      1
43.50      1
Name:  HIV/AIDS, Length: 167, dtype: int64
```

In [37]:
```
data_final['GDP'].value_counts()
```

Out[37]:
```
584.26     1
1464.50    1
43.65      1
116.27     1
143.68     1
          ..
194.27     1
763.66     1
8754.11    1
1716.23    1
547.36     1
Name: GDP, Length: 1649, dtype: int64
```

In [38]:
```
data_final['Population'].value_counts()
```

Out[38]:
```
718239.00      2
1141.00        2
33736494.00    1
5666581.00     1
526796.00      1
              ..
1536411.00     1
1586754.00     1
16421.00       1
169711.00      1
12222251.00    1
Name: Population, Length: 1647, dtype: int64
```

In [39]:
```
data_final[' thinness  1-19 years'].value_counts()
```

Out[39]:
```
2.00     42
1.90     41
2.20     40
1.00     39
2.10     38
         ..
17.30     1
4.40      1
14.80     1
11.80     1
14.70     1
Name:  thinness  1-19 years, Length: 179, dtype: int64
```

In [40]:
```
data_final['Schooling'].value_counts()
```

Out[40]:
```
12.90    44
12.50    35
12.80    33
12.30    31
10.70    31
         ..
19.00     1
20.70     1
20.60     1
20.50     1
17.50     1
Name: Schooling, Length: 147, dtype: int64
```

In [41]:
```
# Creating dummies for "Status", as its a categorical feature.
# Dropping one of the dummy variables for each categorical feature to avoid "dummy variable trap"
Status_dummies = pd.get_dummies(data_final['Status'], prefix='Stat', drop_first=True)
```

In [42]:
```
df1_final = data_final.drop(['Status'], axis = 1)
# Adding the dummy columns
df1_final = pd.concat([df1_final, Status_dummies], axis=1)


df1_final.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 1649 entries, 0 to 2937
Data columns (total 16 columns):
 #   Column                 Non-Null Count  Dtype
---  ------                 --------------  -----
 0   Life expectancy        1649 non-null   float64
 1   Adult Mortality        1649 non-null   float64
 2   Alcohol                1649 non-null   float64
 3   Hepatitis B            1649 non-null   float64
 4   Measles                1649 non-null   float64
 5    BMI                   1649 non-null   float64
 6   under-five deaths      1649 non-null   int64
 7   Polio                  1649 non-null   float64
 8   Total expenditure      1649 non-null   float64
 9   Diphtheria             1649 non-null   float64
 10   HIV/AIDS              1649 non-null   float64
 11  GDP                    1649 non-null   float64
 12  Population             1649 non-null   float64
 13   thinness  1-19 years  1649 non-null   float64
 14  Schooling              1649 non-null   float64
 15  Stat_Developing        1649 non-null   uint8
dtypes: float64(14), int64(1), uint8(1)
memory usage: 207.7 KB
```

In [43]:
```
column_names = df1_final.columns.values.tolist()
column_names
```

Out[43]:
```
['Life expectancy ',
 'Adult Mortality',
 'Alcohol',
 'Hepatitis B',
 'Measles ',
 ' BMI ',
 'under-five deaths ',
 'Polio',
 'Total expenditure',
 'Diphtheria ',
 ' HIV/AIDS',
 'GDP',
 'Population',
 ' thinness  1-19 years',
 'Schooling',
 'Stat_Developing']
```

Model Creation This is the step where we fit the data with a multilinear regression model. It is an iterative approach that will tune models to get the highest performance possible. While creating the model iterations, we'll check the primary assumptions for the multilinear regression - linearity, normality and homoscedasticity. We'll also use a "train and test split" to validate each model iteration

Iteration 1: Model 1 I will create the first model without running any transformations. So far, I removed the predictors that have high multicollinearity and created dummy variables for the categorical predictors.

In [44]:
```
# First, I'm splitting the data into train and test groups
# X is my independent variables aka features
X = df1_final.drop('Life expectancy ', axis=1)
# Y is my dependent variable which is "price" in this model
y = df1_final['Life expectancy ']
# Splitting the data
# Using a random state for reproducible output
# I picked "25" as a random state
# We have 21,597 entries in this dataframe and I need to decide what the split ratio will be.
# With less testing data, the performance of the model will have greater variance.
# With less training data, my parameters estimates will have greater variance.
# Because the size of my sample (n=21,597) is not too large, I'll follow the industry best practice.
# I'm using 80/20 split here (instead of 75/25 default split)
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.2, random_state=25)
```

In [45]:
```
regressor = LinearRegression()
regressor.fit(X_train, y_train)
```

Out[45]:
```
LinearRegression()
```

In [46]:
```
# Fitting a linear regression model and calculate MSE for test and train
linreg = LinearRegression()
linreg.fit(X_train, y_train)
y_hat_train = linreg.predict(X_train)
y_hat_test = linreg.predict(X_test)
# Calculating the R2 and MSE
train_r2 = r2_score(y_train, y_hat_train)
test_r2 = r2_score(y_test, y_hat_test)
train_mse = mean_squared_error(y_train, y_hat_train)
test_mse = mean_squared_error(y_test, y_hat_test)
print('Training Scores:', 'R2', train_r2, '&', 'Mean Absolute Error', train_mse)
print('Testing Scores:', 'R2', test_r2, '&', 'Mean Absolute Error', test_mse)
```

```
Training Scores: R2 0.8150932037574432 & Mean Absolute Error 14.505405485037937
Testing Scores: R2 0.7955814388942134 & Mean Absolute Error 14.900817533935173
```

2.69% DIFFERNCE BETWEEN TRAIN AND TEST MSE

Comments: The first model explains 81.3% of the variance in the Life expectancy (dependent variable Y) by the independent variables (X) whereas he test group explains 79.95% of the variance. The difference in R2s only -1.95% (Test vs Train). Mean Absolute Erros (MSEs) are also producing close outputs. Moreover DIFFERNCE BETWEEN TRAIN AND TEST MSE is only 2.69%.

In [47]:
```
# I'll run the Ordinary Least Squares (OLS) Regression to evaluate the model.
#X_train X_test y_train y_test
import statsmodels.api as sm
from statsmodels.formula.api import ols
X_train_with_intercept = sm.add_constant(X_train)
model_1 = sm.OLS(y_train,X_train_with_intercept).fit()
model_1.summary()
```

Out[47]:

OLS Regression Results

Dep. Variable:	Life expectancy	**R-squared:**	0.815
Model:	OLS	**Adj. R-squared:**	0.813
Method:	Least Squares	**F-statistic:**	382.9
Date:	Sun, 26 Mar 2023	**Prob (F-statistic):**	0.00
Time:	10:19:45	**Log-Likelihood:**	-3635.4
No. Observations:	1319	**AIC:**	7303.
Df Residuals:	1303	**BIC:**	7386.
Df Model:	15		
Covariance Type:	nonrobust		

	coef	std err	t	P>|t|	[0.025	0.975]
const	55.4091	0.986	56.218	0.000	53.476	57.343
Adult Mortality	-0.0192	0.001	-17.042	0.000	-0.021	-0.017
Alcohol	-0.0928	0.038	-2.416	0.016	-0.168	-0.017
Hepatitis B	-0.0113	0.005	-2.118	0.034	-0.022	-0.001
Measles	3.093e-05	1.4e-05	2.212	0.027	3.5e-06	5.84e-05
BMI	0.0431	0.007	6.119	0.000	0.029	0.057
under-five deaths	-0.0020	0.001	-1.888	0.059	-0.004	7.98e-05
Polio	0.0128	0.006	2.049	0.041	0.001	0.025
Total expenditure	0.0514	0.048	1.076	0.282	-0.042	0.145
Diphtheria	0.0277	0.007	3.862	0.000	0.014	0.042
HIV/AIDS	-0.4544	0.022	-20.594	0.000	-0.498	-0.411
GDP	7.366e-05	1.1e-05	6.692	0.000	5.21e-05	9.53e-05
Population	2.179e-09	2.01e-09	1.084	0.279	-1.76e-09	6.12e-09
thinness 1-19 years	-0.0726	0.032	-2.234	0.026	-0.136	-0.009
Schooling	1.2345	0.061	20.268	0.000	1.115	1.354
Stat_Developing	-1.2341	0.398	-3.102	0.002	-2.015	-0.454

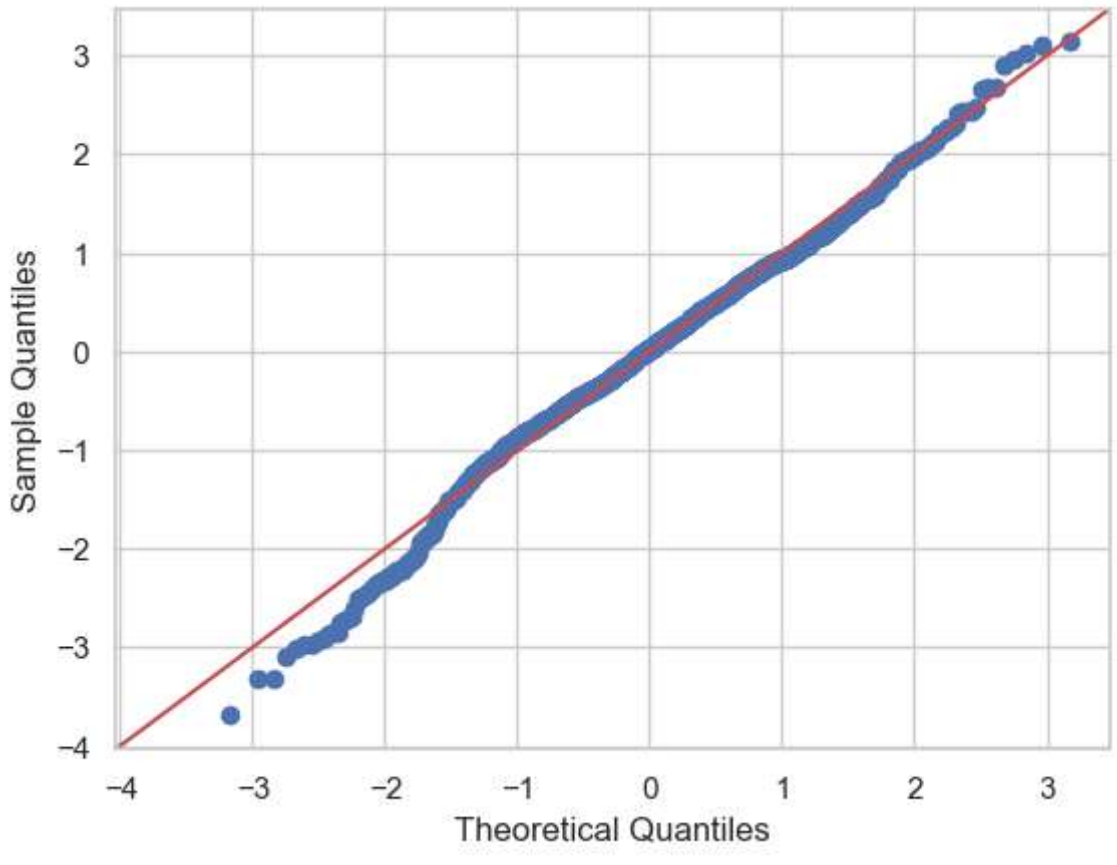
Omnibus:	27.661	**Durbin-Watson:**	1.962
Prob(Omnibus):	0.000	**Jarque-Bera (JB):**	35.347
Skew:	-0.257	**Prob(JB):**	2.11e-08
Kurtosis:	3.616	**Cond. No.**	6.89e+08

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.
[2] The condition number is large, 6.89e+08. This might indicate that there are
strong multicollinearity or other numerical problems.

Comments:

A high Jarque-Bera of 35.34 result shows that the errors are not normally distributed. Finally, the condition number is large, 6.89e+08. It means that the model's predictions are highly sensitive to small changes in the input features, and the model may not be very stable or reliable. Based on the above model i will have to eliminate a FEW features that will not be useful in the predictive analysis as they are not statistically significant in in explaining the variance or implications in Life Expectancy of the model moving forward . This will be done based on the P VALUES which are greater than 0.05(significance level). The statisctical significance of these predictor variables with respect to Life Expectancy can also be seen in the confidence interval of these variables(as it has has zero in them) so we sill be dropping-: -'under-five deaths ','Total expenditure','Population' in ITERATION2

In [48]:
```
# Let's look into the normality assumptions with visuals.
# Drawing a Q-Q Plot the check how the errors are distributed.
resid1 = model_1.resid
fig = sm.graphics.qqplot(resid1, dist=stats.norm, line='45', fit=True)
```



As expected, the residuals are not normally dostributed as indicated by the Graph

In [49]:
```
from statsmodels.compat import lzip
import statsmodels.stats.api as sms

#perform Bresuch-Pagan test
names = ['Lagrange multiplier statistic', 'p-value',
        'f-value', 'f p-value']
test = sms.het_breuschpagan(model_1.resid, model_1.model.exog)

lzip(names, test)
```

Out[49]:
```
[('Lagrange multiplier statistic', 141.73613678469613),
 ('p-value', 1.0454576103199362e-22),
 ('f-value', 10.458272043678251),
 ('f p-value', 4.0673914928471036e-24)]
```

I have used the Bresuch-Pagan test for determining heteroscedasticity. If the p-value is below a certain threshold (common choices are 0.01, 0.05, and 0.10) then there is sufficient evidence to say that heteroscedasticity is present. So, as the p-value is definetely below 0.05, we can discern hetroscedasticity in error residuals.

START PROCESSING BEFORE ITERATION 2 - Log trasformations

Dropping the columns that have p>0.05

In [50]:
```
df2 = df1_final.drop(['under-five deaths ','Total expenditure','Population',], axis=1)
```

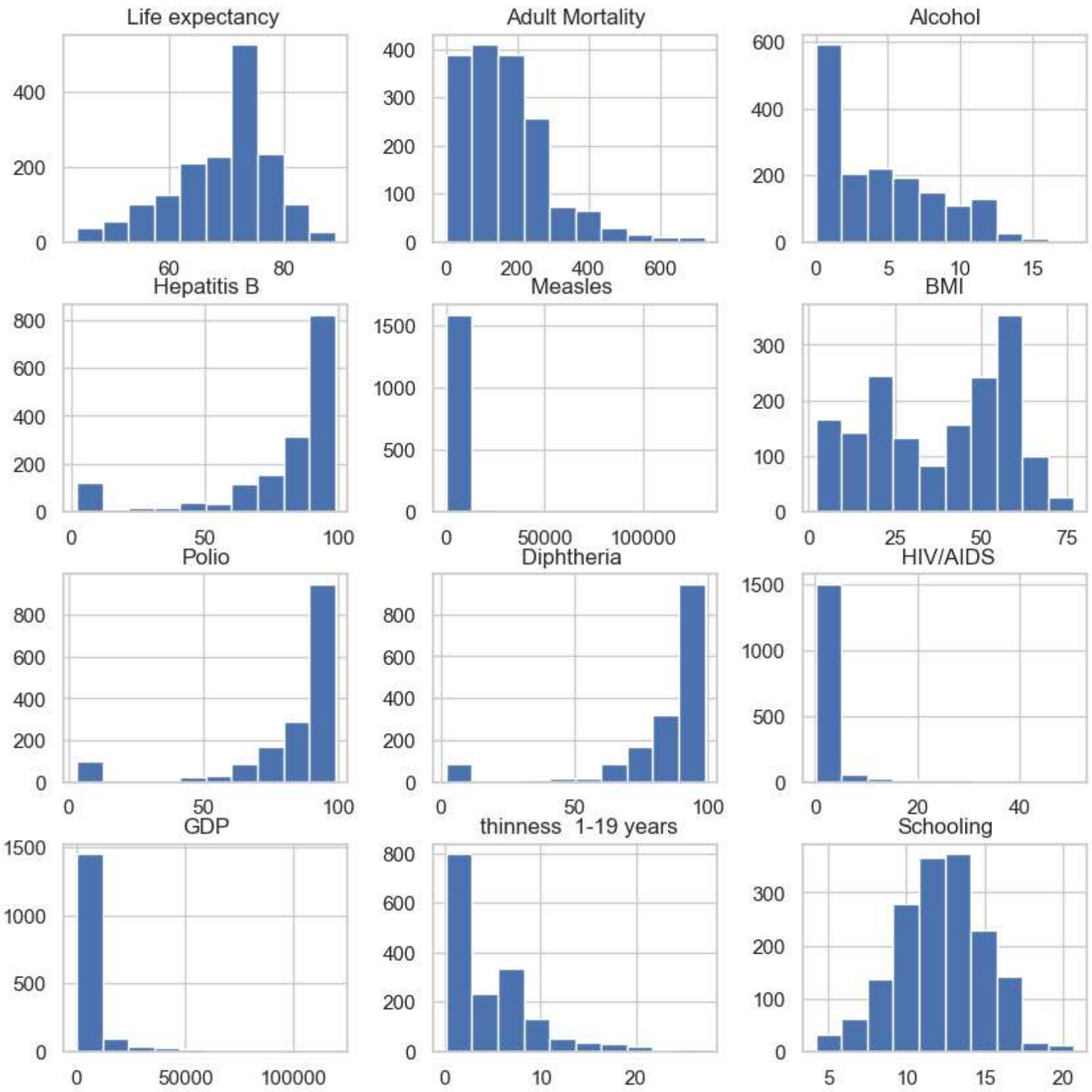
In [51]:
```
df2.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 1649 entries, 0 to 2937
Data columns (total 13 columns):
 #   Column                 Non-Null Count  Dtype
---  ------                 --------------  -----
 0   Life expectancy        1649 non-null   float64
 1   Adult Mortality        1649 non-null   float64
 2   Alcohol                1649 non-null   float64
 3   Hepatitis B            1649 non-null   float64
 4   Measles                1649 non-null   float64
 5    BMI                   1649 non-null   float64
 6   Polio                  1649 non-null   float64
 7   Diphtheria             1649 non-null   float64
 8    HIV/AIDS              1649 non-null   float64
 9   GDP                    1649 non-null   float64
 10   thinness  1-19 years  1649 non-null   float64
 11  Schooling              1649 non-null   float64
 12  Stat_Developing        1649 non-null   uint8
dtypes: float64(12), uint8(1)
memory usage: 169.1 KB
```

In [52]:
```
column_names = df2.columns.values.tolist()
column_names
```

Out[52]:
```
['Life expectancy ',
 'Adult Mortality',
 'Alcohol',
 'Hepatitis B',
 'Measles ',
 ' BMI ',
 'Polio',
 'Diphtheria ',
 ' HIV/AIDS',
 'GDP',
 ' thinness  1-19 years',
 'Schooling',
 'Stat_Developing']
```

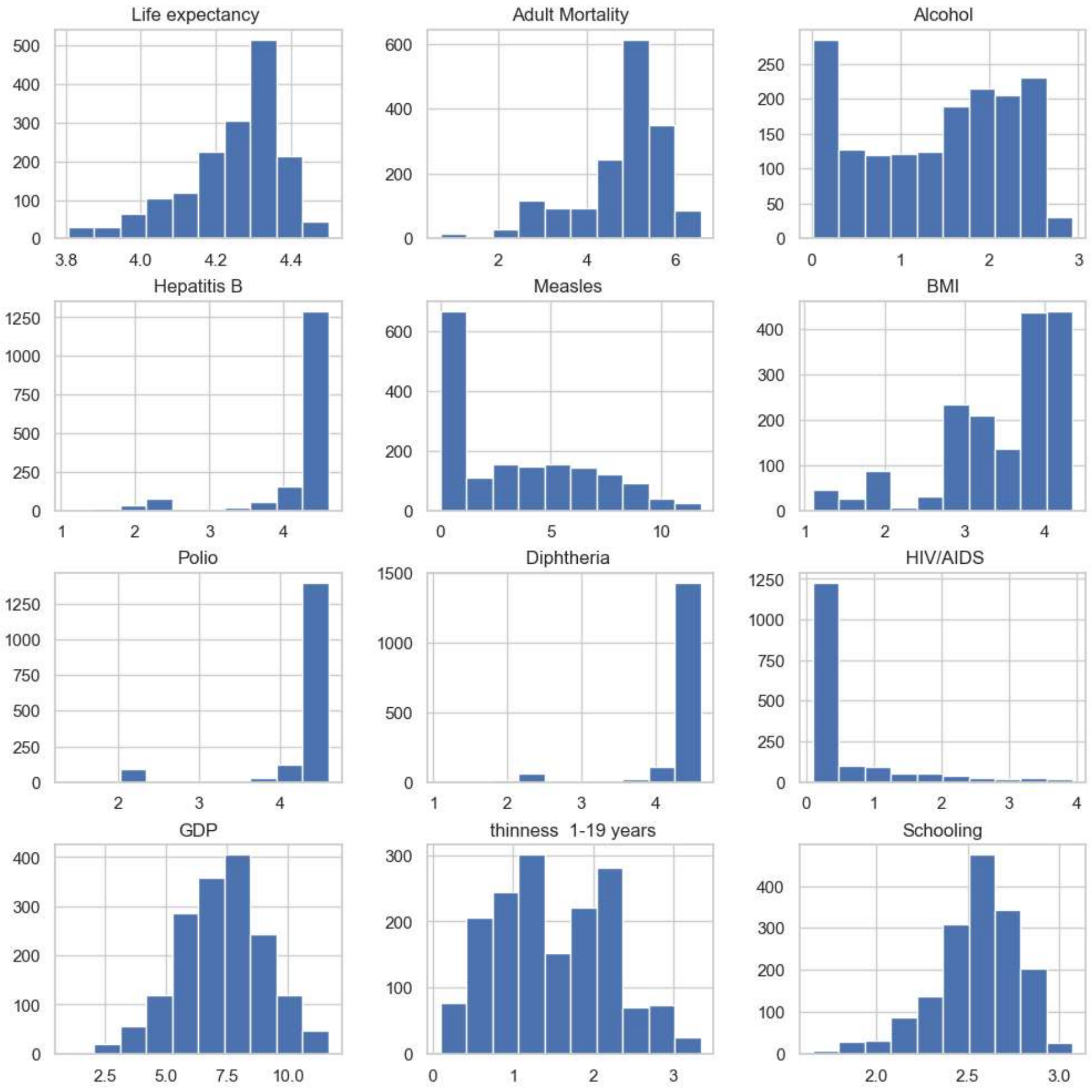
In [53]:
```
# Seperating the continuous variables
df2_continuous = df2.drop(['Stat_Developing'], axis=1)
# Let's check the skewness with histograms
df2_continuous.hist(figsize= (10, 10))
plt.show()
```



In [54]:
```
df2_continuous.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 1649 entries, 0 to 2937
Data columns (total 12 columns):
 #   Column                 Non-Null Count  Dtype  
---  ------                 --------------  -----  
 0   Life expectancy        1649 non-null   float64
 1   Adult Mortality        1649 non-null   float64
 2   Alcohol                1649 non-null   float64
 3   Hepatitis B            1649 non-null   float64
 4   Measles                1649 non-null   float64
 5    BMI                   1649 non-null   float64
 6   Polio                  1649 non-null   float64
 7   Diphtheria             1649 non-null   float64
 8    HIV/AIDS              1649 non-null   float64
 9   GDP                    1649 non-null   float64
 10   thinness  1-19 years  1649 non-null   float64
 11  Schooling              1649 non-null   float64
dtypes: float64(12)
memory usage: 167.5 KB
```

In [55]:
```
# Taking logs of continuous dataset
df2_continuous_log = np.log1p(df2_continuous)
# Replace infinite updated data with nan
df2_continuous_log.replace([np.inf, -np.inf], np.nan, inplace=True)
# Drop rows with NaN
df2_continuous_log.dropna(inplace=True)
# Plotting histograms
df2_continuous_log.hist(figsize= (12, 12))
plt.show()
```



In [56]:
```
df2_continuous_log.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 1649 entries, 0 to 2937
Data columns (total 12 columns):
 #   Column                 Non-Null Count  Dtype  
---  ------                 --------------  -----  
 0   Life expectancy        1649 non-null   float64
 1   Adult Mortality        1649 non-null   float64
 2   Alcohol                1649 non-null   float64
 3   Hepatitis B            1649 non-null   float64
 4   Measles                1649 non-null   float64
 5    BMI                   1649 non-null   float64
 6   Polio                  1649 non-null   float64
 7   Diphtheria             1649 non-null   float64
 8    HIV/AIDS              1649 non-null   float64
 9   GDP                    1649 non-null   float64
 10   thinness  1-19 years  1649 non-null   float64
 11  Schooling              1649 non-null   float64
dtypes: float64(12)
memory usage: 167.5 KB
```

In [57]:
```
log_transformer = FunctionTransformer(np.log, validate=True)

log_names = [f'{column}_log' for column in df2_continuous.columns]
df2_continuous_log.columns = log_names
```

In [58]:
```
df2_continuous_log.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 1649 entries, 0 to 2937
Data columns (total 12 columns):
 #   Column                     Non-Null Count  Dtype  
---  ------                     --------------  -----  
 0   Life expectancy _log       1649 non-null   float64
 1   Adult Mortality_log        1649 non-null   float64
 2   Alcohol_log                1649 non-null   float64
 3   Hepatitis B_log            1649 non-null   float64
 4   Measles _log               1649 non-null   float64
 5    BMI _log                  1649 non-null   float64
 6   Polio_log                  1649 non-null   float64
 7   Diphtheria _log            1649 non-null   float64
 8    HIV/AIDS_log              1649 non-null   float64
 9   GDP_log                    1649 non-null   float64
 10   thinness  1-19 years_log  1649 non-null   float64
 11  Schooling_log              1649 non-null   float64
dtypes: float64(12)
memory usage: 167.5 KB
```

In [59]:
```
df2_categorical = df2['Stat_Developing']
```

In [60]:
```
df2_processed = pd.concat([df2_continuous_log, df2_categorical], axis=1)
```

In [61]:
```
df2_processed.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 1649 entries, 0 to 2937
Data columns (total 13 columns):
 #   Column                     Non-Null Count  Dtype  
---  ------                     --------------  -----  
 0   Life expectancy _log       1649 non-null   float64
 1   Adult Mortality_log        1649 non-null   float64
 2   Alcohol_log                1649 non-null   float64
 3   Hepatitis B_log            1649 non-null   float64
 4   Measles _log               1649 non-null   float64
 5    BMI _log                  1649 non-null   float64
 6   Polio_log                  1649 non-null   float64
 7   Diphtheria _log            1649 non-null   float64
 8    HIV/AIDS_log              1649 non-null   float64
 9   GDP_log                    1649 non-null   float64
 10   thinness  1-19 years_log  1649 non-null   float64
 11  Schooling_log              1649 non-null   float64
 12  Stat_Developing            1649 non-null   uint8  
dtypes: float64(12), uint8(1)
memory usage: 169.1 KB
```

In [62]:
```
df2_processed.head()
```

Out[62]:

	Life expectancy _log	Adult Mortality_log	Alcohol_log	Hepatitis B_log	Measles _log	BMI _log	Polio_log	Diphtheria _log	HIV/AIDS_log	GDP_log	thinness 1-19 years_log	Schooling_log	Stat_Develop
0	4.19	5.58	0.01	4.19	7.05	3.00	1.95	4.19	0.10	6.37	2.90	2.41	
1	4.11	5.61	0.01	4.14	6.20	2.98	4.08	4.14	0.10	6.42	2.92	2.40	
2	4.11	5.59	0.01	4.17	6.07	2.95	4.14	4.17	0.10	6.45	2.93	2.39	
3	4.10	5.61	0.01	4.22	7.93	2.92	4.22	4.22	0.10	6.51	2.94	2.38	
4	4.10	5.62	0.01	4.23	8.01	2.90	4.23	4.23	0.10	4.17	2.95	2.35	

In [63]:
```
column_names = df2_processed.columns.values.tolist()
column_names
```

Out[63]:
```
['Life expectancy _log',
 'Adult Mortality_log',
 'Alcohol_log',
 'Hepatitis B_log',
 'Measles _log',
 ' BMI _log',
 'Polio_log',
 'Diphtheria _log',
 ' HIV/AIDS_log',
 'GDP_log',
 ' thinness  1-19 years_log',
 'Schooling_log',
 'Stat_Developing']
```

In [64]:
```
# X is my independent variables aka features
X_2 = df2_processed.drop('Life expectancy _log', axis=1)
# Y is my dependent variable which is "price" in this model
Y_2 =df2_processed['Life expectancy _log']
# Splitting the data
X_2_train, X_2_test, y_2_train, y_2_test = train_test_split(X_2, Y_2, test_size = 0.2, random_state=12)
```

In [65]:
```
regressor = LinearRegression()
regressor.fit(X_2_train, y_2_train)
```

Out[65]:
```
LinearRegression()
```

In [66]:
```
#Fitting a linear regression model and calculate MSE for test and train
linreg = LinearRegression()
linreg.fit(X_2_train, y_2_train)
y_2_hat_train = linreg.predict(X_2_train)
y_2_hat_test = linreg.predict(X_2_test)
# Calculating the R2 and MSE
train_r2 = r2_score(y_2_train, y_2_hat_train)
test_r2 = r2_score(y_2_test, y_2_hat_test)
train_mse = mean_squared_error(y_2_train, y_2_hat_train)
test_mse = mean_squared_error(y_2_test, y_2_hat_test)
print('Training Scores:', 'R2', train_r2, '&', 'Mean Absolute Error', train_mse)
print('Testing Scores:', 'R2', test_r2, '&', 'Mean Absolute Error', test_mse)
```

```
Training Scores: R2 0.8381965351117399 & Mean Absolute Error 0.00286480054008969
Testing Scores: R2 0.8161521162592456 & Mean Absolute Error 0.003195749569142791
```

Comments: The second model (after Log Transformation) explains 83.8% of the variance in the Life expectancy (dependent variable Y) by the independent variables (X) whereas he test group explains 81.61% of the variance. The difference in R2s only -2.19% (Test vs Train). Mean Absolute Erros (MSEs) are also producing close diffrence being that is difference between Train and Test MSE is 10.92%.

In [67]:
```
# I'll run the Ordinary Least Squares (OLS) Regression to evaluate the model.
#X_train X_test y_train y_test
import statsmodels.api as sm
from statsmodels.formula.api import ols
X_2_train_with_intercept = sm.add_constant(X_2_train)
model_2 = sm.OLS(y_2_train,X_2_train_with_intercept).fit()
model_2.summary()
```

Out[67]:

OLS Regression Results

Dep. Variable:	Life expectancy _log	**R-squared:**	0.838
Model:	OLS	**Adj. R-squared:**	0.837
Method:	Least Squares	**F-statistic:**	563.8
Date:	Sun, 26 Mar 2023	**Prob (F-statistic):**	0.00
Time:	10:19:49	**Log-Likelihood:**	1990.0
No. Observations:	1319	**AIC:**	-3954.
Df Residuals:	1306	**BIC:**	-3887.
Df Model:	12		
Covariance Type:	nonrobust		

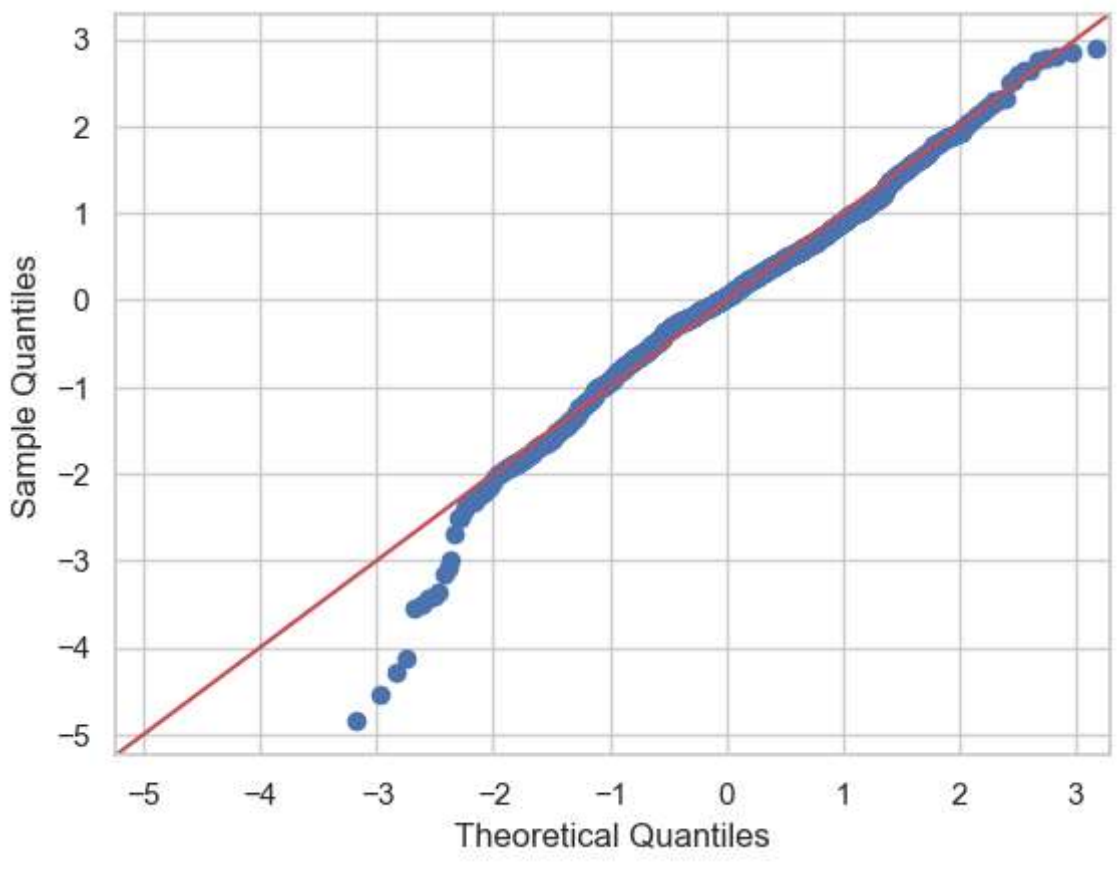
	coef	std err	t	P>|t|	[0.025	0.975]
const	3.8658	0.029	132.638	0.000	3.809	3.923
Adult Mortality_log	-0.0114	0.002	-7.367	0.000	-0.014	-0.008
Alcohol_log	0.0080	0.002	3.477	0.001	0.003	0.013
Hepatitis B_log	-0.0030	0.003	-1.146	0.252	-0.008	0.002
Measles _log	-0.0019	0.000	-3.938	0.000	-0.003	-0.001
BMI _log	0.0032	0.002	1.426	0.154	-0.001	0.008
Polio_log	0.0014	0.003	0.482	0.630	-0.004	0.007
Diphtheria _log	0.0115	0.003	3.400	0.001	0.005	0.018
HIV/AIDS_log	-0.0965	0.002	-42.681	0.000	-0.101	-0.092
GDP_log	0.0068	0.001	6.407	0.000	0.005	0.009
thinness 1-19 years_log	-0.0115	0.003	-4.095	0.000	-0.017	-0.006
Schooling_log	0.1593	0.010	16.115	0.000	0.140	0.179
Stat_Developing	-0.0180	0.005	-3.514	0.000	-0.028	-0.008

Omnibus:	80.842	**Durbin-Watson:**	2.015
Prob(Omnibus):	0.000	**Jarque-Bera (JB):**	147.262
Skew:	-0.439	**Prob(JB):**	1.05e-32
Kurtosis:	4.382	**Cond. No.**	264.

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.

Comments: Our Adj. R-squared has increased from 81.3% to 83.7%.Condition Number has reduced exponentialy from (6.89e+08) to 264

In [68]:
```
# Drawing a Q-Q Plot the check how the errors are distributed.
resid2 = model_2.resid
fig = sm.graphics.qqplot(resid2, dist=stats.norm, line='45', fit=True)
```



The residuals are not normally dostributed as indicated by the Graph even after Log transformations

In [69]:
```
from statsmodels.compat import lzip
import statsmodels.stats.api as sms

#perform Bresuch-Pagan test
names = ['Lagrange multiplier statistic', 'p-value',
        'f-value', 'f p-value']
test = sms.het_breuschpagan(model_2.resid, model_2.model.exog)

lzip(names, test)
```

Out[69]:
```
[('Lagrange multiplier statistic', 53.573296860096015),
 ('p-value', 3.2563247000970366e-07),
 ('f-value', 4.607584509219764),
 ('f p-value', 2.3573228707716821e-07)]
```

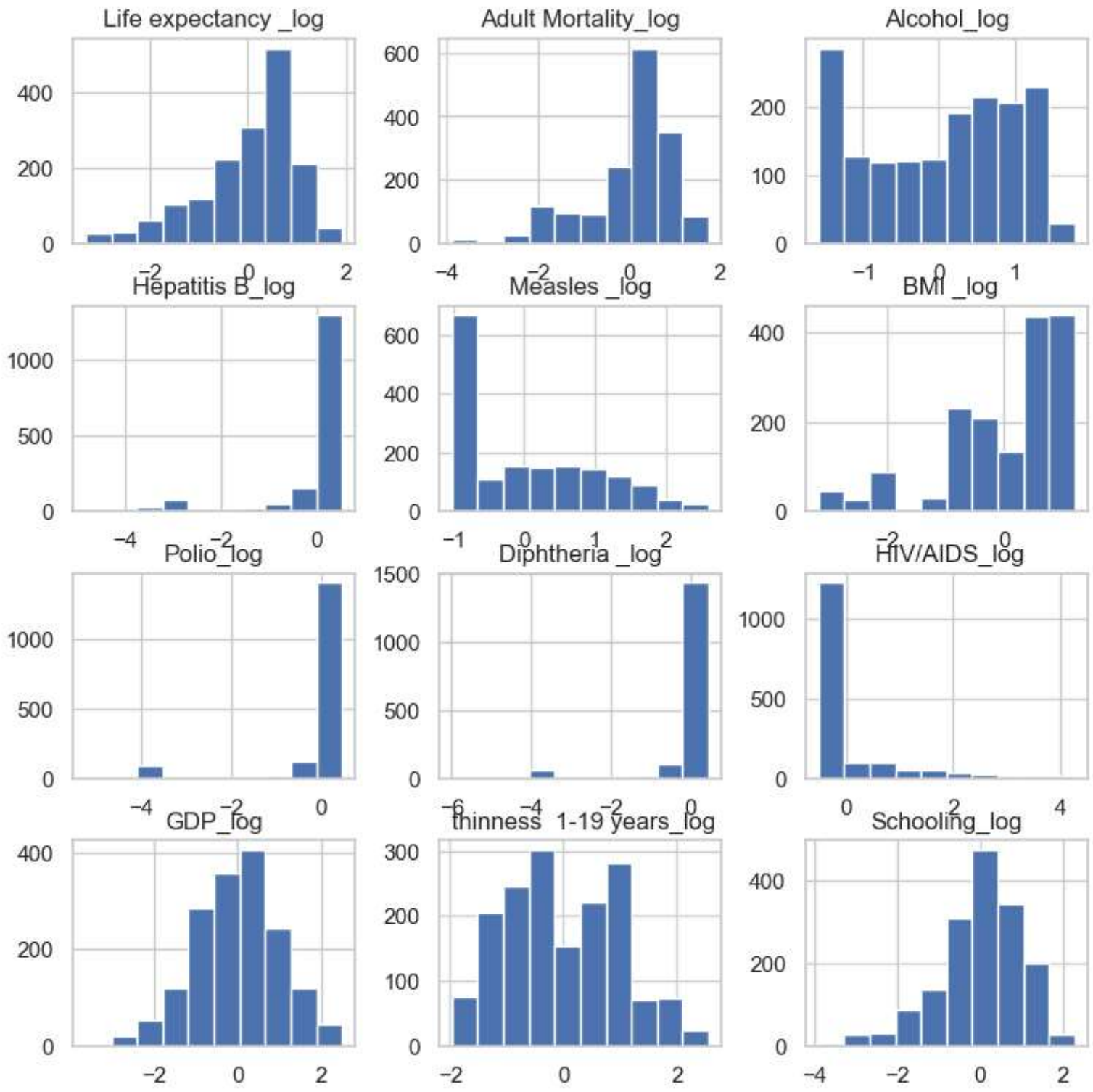
As the p-value of the Bresuch-Pagan test is less than significance level (i.e. α = .05) then we reject the null hypothesis and conclude that heteroscedasticity is still present in the regression model and we discern hetroscedasticity in error residuals.

Start of iteration 3 - Feature Scaling and Normalisation

In [70]:
```
df2_continuous_log.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 1649 entries, 0 to 2937
Data columns (total 12 columns):
 #   Column                      Non-Null Count  Dtype  
---  ------                      --------------  -----  
 0   Life expectancy _log        1649 non-null   float64
 1   Adult Mortality_log         1649 non-null   float64
 2   Alcohol_log                 1649 non-null   float64
 3   Hepatitis B_log             1649 non-null   float64
 4   Measles _log                1649 non-null   float64
 5    BMI _log                   1649 non-null   float64
 6   Polio_log                   1649 non-null   float64
 7   Diphtheria _log             1649 non-null   float64
 8    HIV/AIDS_log               1649 non-null   float64
 9   GDP_log                     1649 non-null   float64
 10   thinness  1-19 years_log   1649 non-null   float64
 11  Schooling_log               1649 non-null   float64
dtypes: float64(12)
memory usage: 167.5 KB
```

In [71]:
```
def normalize(feature):
    return (feature - feature.mean()) / feature.std()

df3_log_norm = df2_continuous_log.apply(normalize)
df3_log_norm.hist(figsize  = [9, 9]);
```



In [72]:
```
lognorm_names = [f'{column}_norm' for column in df2_continuous_log.columns]
df3_log_norm.columns = lognorm_names
```

In [73]:
```
df3_log_norm.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 1649 entries, 0 to 2937
Data columns (total 12 columns):
 #   Column                          Non-Null Count  Dtype  
---  ------                          --------------  -----  
 0   Life expectancy _log_norm       1649 non-null   float64
 1   Adult Mortality_log_norm        1649 non-null   float64
 2   Alcohol_log_norm                1649 non-null   float64
 3   Hepatitis B_log_norm            1649 non-null   float64
 4   Measles _log_norm               1649 non-null   float64
 5    BMI _log_norm                  1649 non-null   float64
 6   Polio_log_norm                  1649 non-null   float64
 7   Diphtheria _log_norm            1649 non-null   float64
 8    HIV/AIDS_log_norm              1649 non-null   float64
 9   GDP_log_norm                    1649 non-null   float64
 10   thinness  1-19 years_log_norm  1649 non-null   float64
 11  Schooling_log_norm              1649 non-null   float64
dtypes: float64(12)
memory usage: 167.5 KB
```

In [74]:
```
df3_processed = pd.concat([df3_log_norm, df2_categorical], axis=1)
```

In [75]:
```
df3_processed.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 1649 entries, 0 to 2937
Data columns (total 13 columns):
 #   Column                           Non-Null Count  Dtype  
---  ------                           --------------  -----  
 0   Life expectancy _log_norm        1649 non-null   float64
 1   Adult Mortality_log_norm         1649 non-null   float64
 2   Alcohol_log_norm                 1649 non-null   float64
 3   Hepatitis B_log_norm             1649 non-null   float64
 4   Measles _log_norm                1649 non-null   float64
 5    BMI _log_norm                   1649 non-null   float64
 6   Polio_log_norm                   1649 non-null   float64
 7   Diphtheria _log_norm             1649 non-null   float64
 8    HIV/AIDS_log_norm               1649 non-null   float64
 9   GDP_log_norm                     1649 non-null   float64
 10   thinness  1-19 years_log_norm   1649 non-null   float64
 11  Schooling_log_norm               1649 non-null   float64
 12  Stat_Developing                  1649 non-null   uint8  
dtypes: float64(12), uint8(1)
memory usage: 169.1 KB
```

In [76]:
```
column_names = df3_processed.columns.values.tolist()
column_names
```

Out[76]:
```
['Life expectancy _log_norm',
 'Adult Mortality_log_norm',
 'Alcohol_log_norm',
 'Hepatitis B_log_norm',
 'Measles _log_norm',
 ' BMI _log_norm',
 'Polio_log_norm',
 'Diphtheria _log_norm',
 ' HIV/AIDS_log_norm',
 'GDP_log_norm',
 ' thinness  1-19 years_log_norm',
 'Schooling_log_norm',
 'Stat_Developing']
```

In [77]:
```
# X is my independent variables aka features
X_3 = df3_processed.drop('Life expectancy _log_norm', axis=1)
# Y is my dependent variable which is "price" in this model
Y_3 = df3_processed['Life expectancy _log_norm']
# Splitting the data
X_3_train, X_3_test, y_3_train, y_3_test = train_test_split(X_3, Y_3, test_size = 0.2, random_state=12)
```

In [78]:
```
regressor = LinearRegression()
regressor.fit(X_3_train, y_3_train)
```

Out[78]:
```
LinearRegression()
```

In [79]:
```
#Fitting a linear regression model and calculate MSE for test and train
linreg = LinearRegression()
linreg.fit(X_3_train, y_3_train)
y_3_hat_train = linreg.predict(X_3_train)
y_3_hat_test = linreg.predict(X_3_test)
# Calculating the R2 and MSE
train_r3 = r2_score(y_3_train, y_3_hat_train)
test_r3 = r2_score(y_3_test, y_3_hat_test)
train_mse = mean_squared_error(y_3_train, y_3_hat_train)
test_mse = mean_squared_error(y_3_test, y_3_hat_test)
print('Training Scores:', 'R2', train_r3, '&', 'Mean Absolute Error', train_mse)
print('Testing Scores:', 'R2', test_r3, '&', 'Mean Absolute Error', test_mse)
```

```
Training Scores: R2 0.83819653511174 & Mean Absolute Error 0.16225893160448604
Testing Scores: R2 0.8161521162592454 & Mean Absolute Error 0.1810034951851734
```

Comments: The Third model (after Feature Scaling) explains 83.8% of the variance in the Life expectancy (dependent variable Y) by the independent variables (X) whereas he test group explains 81.61% of the variance. The difference in R2s only -2.19% (Test vs Train). Mean Absolute Erros (MSEs) are also producing close diffrence being that is difference between Train and Test MSE is 10.92%. All same as Iteration 2

In [80]:
```
# I'll run the Ordinary Least Squares (OLS) Regression to evaluate the model.
#X_train X_test y_train y_test
import statsmodels.api as sm
from statsmodels.formula.api import ols
X_3_train_with_intercept = sm.add_constant(X_3_train)
model_3 = sm.OLS(y_3_train,X_3_train_with_intercept).fit()
model_3.summary()
```

Out[80]:

OLS Regression Results

Dep. Variable:	Life expectancy _log_norm	**R-squared:**	0.838
Model:	OLS	**Adj. R-squared:**	0.837
Method:	Least Squares	**F-statistic:**	563.8
Date:	Sun, 26 Mar 2023	**Prob (F-statistic):**	0.00
Time:	10:19:52	**Log-Likelihood:**	-672.24
No. Observations:	1319	**AIC:**	1370.
Df Residuals:	1306	**BIC:**	1438.
Df Model:	12		
Covariance Type:	nonrobust		

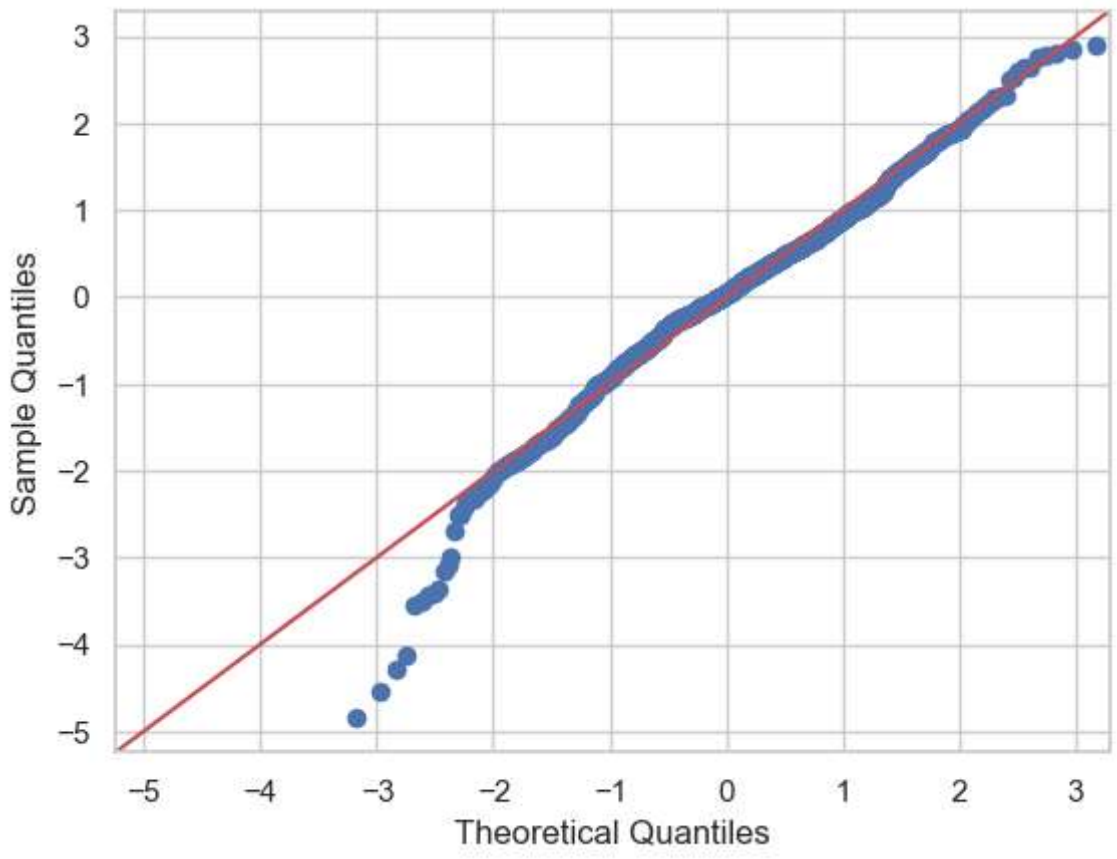
	coef	std err	t	P>|t|	[0.025	0.975]
const	0.1180	0.035	3.409	0.001	0.050	0.186
Adult Mortality_log_norm	-0.0899	0.012	-7.367	0.000	-0.114	-0.066
Alcohol_log_norm	0.0521	0.015	3.477	0.001	0.023	0.081
Hepatitis B_log_norm	-0.0147	0.013	-1.146	0.252	-0.040	0.010
Measles _log_norm	-0.0472	0.012	-3.938	0.000	-0.071	-0.024
BMI _log_norm	0.0182	0.013	1.426	0.154	-0.007	0.043
Polio_log_norm	0.0060	0.013	0.482	0.630	-0.019	0.031
Diphtheria _log_norm	0.0473	0.014	3.400	0.001	0.020	0.075
HIV/AIDS_log_norm	-0.5813	0.014	-42.681	0.000	-0.608	-0.555
GDP_log_norm	0.0898	0.014	6.407	0.000	0.062	0.117
thinness 1-19 years_log_norm	-0.0622	0.015	-4.095	0.000	-0.092	-0.032
Schooling_log_norm	0.2768	0.017	16.115	0.000	0.243	0.311
Stat_Developing	-0.1353	0.039	-3.514	0.000	-0.211	-0.060

Omnibus:	80.842	**Durbin-Watson:**	2.015
Prob(Omnibus):	0.000	**Jarque-Bera (JB):**	147.262
Skew:	-0.439	**Prob(JB):**	1.05e-32
Kurtosis:	4.382	**Cond. No.**	8.69

Notes:
[1] Standard Errors assume that the covariance matrix of the errors is correctly specified.

Comments: Neither the Adj. R-squared nor the Jarque-Bera result had changed. On the other hand, standardization helped with the condition number - which is has further plummeted frpm 264 to 8.69

In [81]:
```
# Drawing a Q-Q Plot the check how the errors are distributed.
resid3 = model_3.resid
fig = sm.graphics.qqplot(resid3, dist=stats.norm, line='45', fit=True)
```



In [82]:
```
from statsmodels.compat import lzip
import statsmodels.stats.api as sms

#perform Bresuch-Pagan test
names = ['Lagrange multiplier statistic', 'p-value',
        'f-value', 'f p-value']
test = sms.het_breuschpagan(model_3.resid, model_3.model.exog)

lzip(names, test)
```

Out[82]:
```
[('Lagrange multiplier statistic', 53.57329686009031),
 ('p-value', 3.256324700104676e-07),
 ('f-value', 4.607584509219251),
 ('f p-value', 2.3573228707774105e-07)]
```

Again after Iteration_3, the p-value of the test is less than significance level (i.e. α = .05) then so we have to reject the null hypothesis and conclude that heteroscedasticity is still present in the regression model as we don't observe any change in the residuals behaviour.

K-FOLD CROSS VALIDATION - Iteration3 Model_3

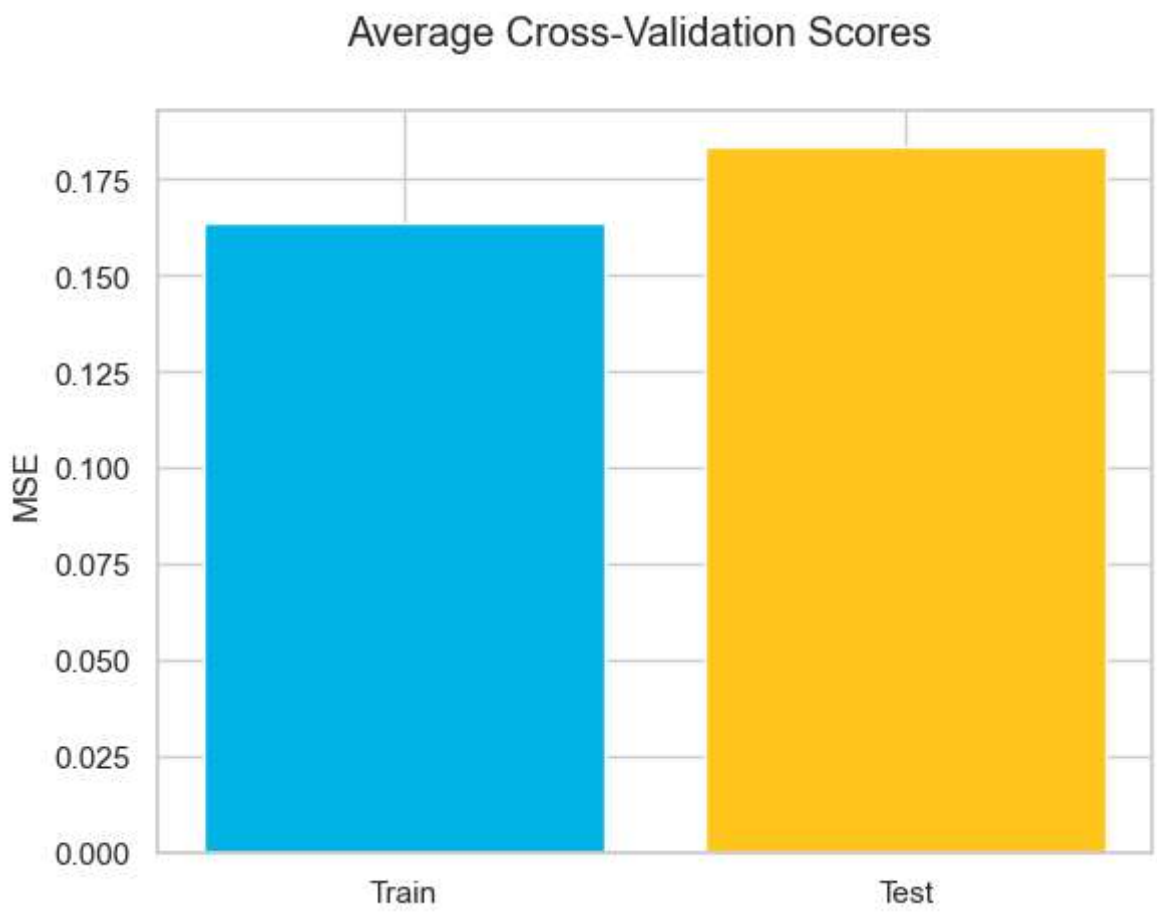
In [83]:
```
from sklearn.model_selection import cross_val_score
from sklearn.metrics import make_scorer

cross_val_score(linreg, X_3,Y_3, scoring=make_scorer(mean_squared_error))
```

Out[83]:
```
array([0.18274131, 0.21130661, 0.20153494, 0.13824113, 0.18469856])
```

In [84]:
```
from sklearn.model_selection import cross_validate
cross_validate(linreg, X_3, Y_3, return_train_score=True)
```

Out[84]:
```
{'fit_time': array([0.01018691, 0.        , 0.00495362, 0.00199485, 0.00498343]),
 'score_time': array([0.00363803, 0.        , 0.00099325, 0.00199556, 0.00199485]),
 'test_score': array([0.83607408, 0.72448438, 0.82405843, 0.79268052, 0.8468868 ]),
 'train_score': array([0.83068453, 0.84713734, 0.83647479, 0.83942968, 0.82522183])}
```

In [85]:
```
cross_val_results = cross_validate(linreg, X_3, Y_3, scoring="neg_mean_squared_error", return_train_score=True)
# Negative signs in front to convert back to MSE from -MSE
train_avg = -cross_val_results["train_score"].mean()
test_avg = -cross_val_results["test_score"].mean()
labels = ["Train", "Test"]
colors = ["#00B3E6", "#FFC51B"]
fig, ax = plt.subplots()
ax.bar(labels, [train_avg, test_avg], color=colors)
ax.set_ylabel("MSE")
fig.suptitle("Average Cross-Validation Scores");
```



COMPARING RESULTS OF TRAIN TEST SPLIT AND CROSS VALIDATION

In [86]:
```
# Find MSE scores for a 5-fold cross-validation
cv_5_results = -cross_val_score(linreg, X_3, Y_3, cv=5, scoring="neg_mean_squared_error")
cv_5_results
```

Out[86]:
```
array([0.18274131, 0.21130661, 0.20153494, 0.13824113, 0.18469856])
```

In [87]:
```
# Get the average MSE score
cv_5_results.mean()
```

Out[87]:
```
0.18370451080988734
```

The Test MSE of the third model is 0.1810034951851734. Rounding to five decimal plcaes the third model Test MSE becomes 0.18100 and the K Fold Cross Validation Test MSE becomes 0.18370. The Difference is 0.0027. Also None of the Cross validation Test MSE 5 splits have a better MSE than the train-test split, meaning that the average is not being skewed by just 1 or 2 values that are significantly worse than the train-test split result Taking the average as well as the spread into account, I would be more inclined to "trust" the cross-validated (less optimistic) score and assume that the performance on unseen data would be closer to 0.18370 than 0.18100

Following code snippet will generate sorted dataframe with feature name and it's p-value. Hence, you will see most relevant features on the top (p-values will be sorted in ascending order)

In [88]:
```
d = {}
for i in X_3_train_with_intercept.columns.tolist():
    d[f'{i}'] = model_3.pvalues[i]

df_pvalue= pd.DataFrame(d.items(), columns=['Var_name', 'p-Value']).sort_values(by = 'p-Value').reset_index(drop=True)
```

In [89]:
```
df_pvalue
```

Out[89]:

	Var_name	p-Value
0	HIV/AIDS_log_norm	0.00
1	Schooling_log_norm	0.00
2	Adult Mortality_log_norm	0.00
3	GDP_log_norm	0.00
4	thinness 1-19 years_log_norm	0.00
5	Measles _log_norm	0.00
6	Stat_Developing	0.00
7	Alcohol_log_norm	0.00
8	const	0.00
9	Diphtheria _log_norm	0.00
10	BMI _log_norm	0.15
11	Hepatitis B_log_norm	0.25
12	Polio_log_norm	0.63

Final Review and conclusion:

The 3 most important determining features that determine the Life expectancy are (as per the three Iterations):-

1) HIV/AIDS : Deaths per 1000 live births due to HIV/AIDS (0-4 years) 2) Schooling: Number of years of Schooling(years) 3) Adult Mortality: Adult Mortality Rates of both sexes (probability of dying between 15 and 60 years per 1000 population)

Life expectancy: Findings

There is a negative correlation between life expectancy and the following fields:

1) HIV/AIDS : Deaths per 1000 live births due to HIV/AIDS (0-4 years) 2) Adult Mortality: Adult Mortality Rates of both sexes (probability of dying between 15 and 60 years per 1000 population) 3) Thinness 1-19 Years: Prevalence of thinness among children and adolescents for Age 10 to 19 (%) 4) Measles: Number of reported cases per 1000 population This suggests that if more adults die, more infants die from hiv/aids and if more of the population is thin(from poor nutrition) life expectancy is expected to go lower.

There is a positive correlation between life expectancy and the following fields:

1) Schooling: Number of years of Schooling(years) 2) GDP: Gross Domestic Product per capita (in USD) 3) Diphtheria: Diphtheria tetanus toxoid and pertussis (DTP3) immunization coverage among 1-year-olds This suggets that if a country's GDP is high, it is expected that a larger percentage is directed towards the health sector. Going to school ensures that a population is fed and hence improved nutrition, health and therefore Life expectancy.