# AN INDUSTRIAL PROJECT REPORT

# SUMMER INTERNSHIP
# AT
# SMARTBRIDGE

# ON
# PREDICTING LIFE EXPECTANCY USING MACHINE LEARNING

# BY


## Ibra Nafis
## Vellore Institute of Technology

# 1. INTRODUCTION

## 1.1 Overview

The term “life expectancy” refers to the number of years a person can expect to live. By definition, life expectancy is based on an estimate of the average age that members of a particular population group will be when they die.

Understanding potential trajectories in health and drivers of health is crucial to guiding long-term investments and policy implementation. Past work on forecasting has provided an incomplete landscape of future health scenarios, highlighting a need for a more robust modelling platform from which policy options and potential health trajectories can be assessed. This study provides a novel approach to modelling life expectancy, all-cause mortality and cause of death forecasts —and alternative future scenarios—for 250 causes of death from 2016 to 2040 in 195 countries and territories.

## 1.2 Purpose

Prognostication of life expectancy is difficult for humans. Our research shows that machine learning and natural language processing techniques offer a feasible and promising approach to predicting life expectancy. The research has potential for real-life applications, such as supporting timely recognition of the right moment to start Advance Care Planning.

The purpose of this paper is to estimate the life expectancy of the world population so that the government has a benchmark in determining policies to further improve the health and health
of the people in their respective countries. The estimation stated in this paper will use the Machine Learning method like Multiple Linear Regression. The data used in this paper is the number of
world population. Data sources come from the Global Health Observatory(GHO) under World Health Organization(WHO). The results of this study are expected to be a reference for the governments of each country to pay more attention to the level of health and welfare of its population so that the life expectancy of the population will be higher.

# 2. LITERATURE SURVEY

## 2.1 Existing Problem

Health forecasts and alternative future scenarios can serve as vital inputs into long-term planning and investments in health, particularly in terms of framing different choices, their potential effects, and the relative certainty associated with each option. Past work to generate health-focused forecasts includes that from the UN Population Division, and the Austrian Wittgenstein Center, which produces life expectancy forecasts with different scenarios to the end of the 21st century. Longer-range forecasts have also been developed to assess the potential effects of climate change on mortality. Furthermore, various national agencies produce country-level mortality forecasts, and forecasts for individual causes of death have been produced periodically as well. Comprehensive forecasts of cause-specific and all-cause mortality were developed as part of the Global Burden of Disease Study 1990 (GBD 1990); those methods were then applied for the period from 2002–30. The primary purpose of these past modelling efforts was to generate reference or baseline forecasts of what was likely to occur on the basis of past trends; however, few—if any—offered insights into a range of future scenarios while accounting for independent drivers of potential health changes.

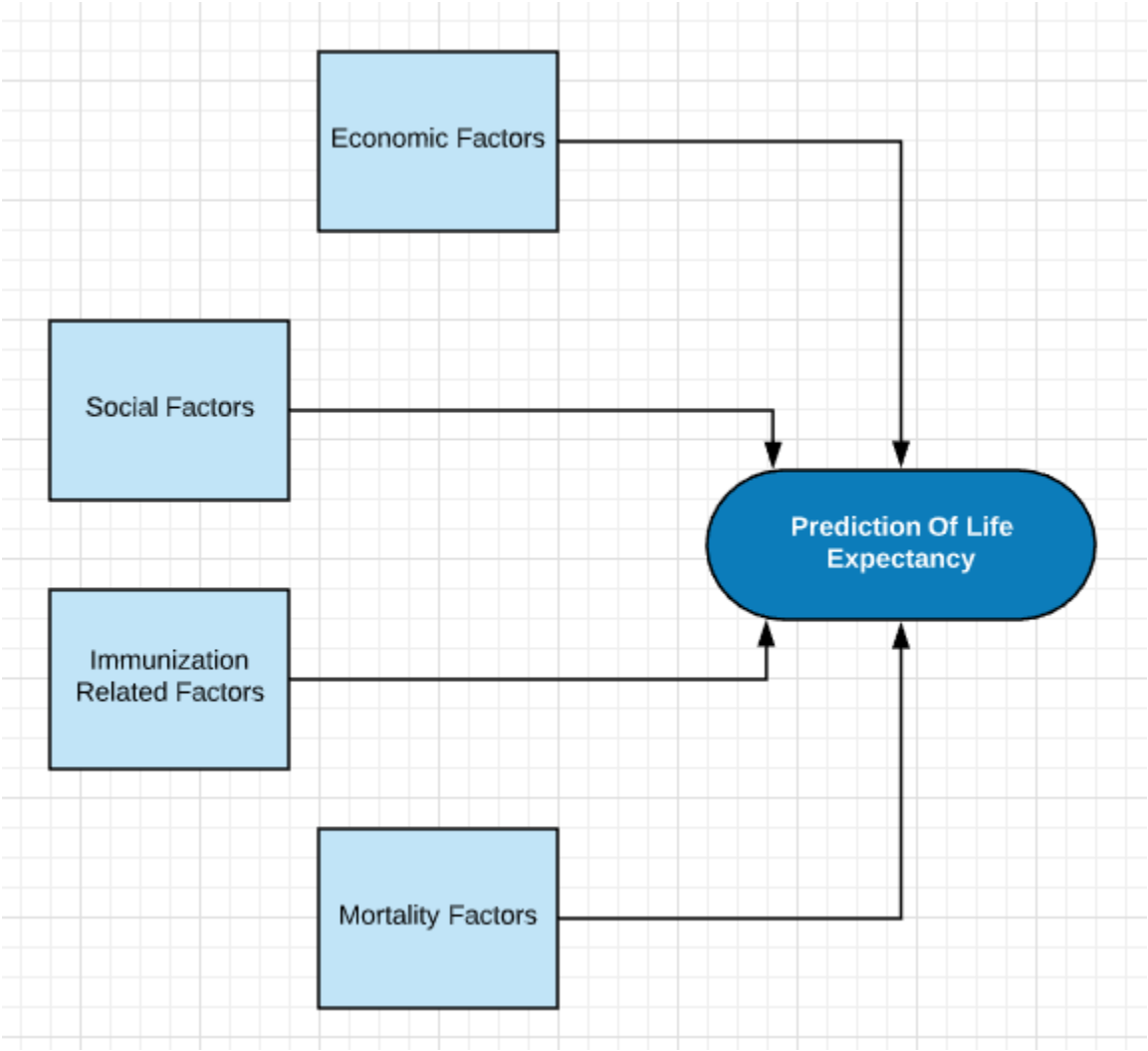
## 2.2 Proposed Solution

With this study, I have provided a completely novel approach to forecasting all-cause and cause-specific mortality and scenario construction. Our modelling framework was designed to leverage the data on risk outcome relationships in the Global Burden of Diseases, Injuries, and Risk Factors Study 2016 such that the relationship between risk factors (eg, smoking) and specific disease outcomes (eg, lung cancer) were consistent with relevant cohort studies and randomised controlled trials. Because this forecasting framework is grounded in 79 independent drivers of health change, we could leverage these models to generate a full suite of alternative scenarios beyond reference forecasts.

Here, I provide a cross-sectional model of life expectancy, using a comprehensive worldwide sample, which analyses the impact of country level variables on average life expectancy. The model variants suggest robustly that proxies for technology, education, disposable income and healthcare all have a significant and positive effect on country variation in average life expectancy, at all income levels.

# 3. THEORETICAL ANALYSIS

## 3.1 Block Diagram



## 3.2 Hardware/Software designing

IBM Cloud, Watson Machine Learning is the Development Environment I have used for my project.

Watson is well-suited for a wide variety of applications and IBM is working closely with partners to address more and more of them. Potential applications can be loosely grouped into the following categories:

- **Diagnosis and action** - Assistance for knowledge workers dealing with a single case for a single client to pinpoint a condition from among many possibilities and make resultant decisions
- **Contact center support** - Personalized self-service experience for clients by dynamically developing personal profiles from unstructured data
- **Research and discovery** - Identification of rare studies and information sources while building a case for original research
- **Process optimization** - Identification of areas for improvement in business processes by analyzing unstructured data that documents and describes process steps and output
- **Fraud and risk management** - Identification of early signs of fraud and management of risk in order to lower overall liability and costs of doing business.

I created Machine Learning service in IBM Cloud and used Watson Machine Learning to create a new Jupyter notebook called Predicting Life Expectancy to train the model. After the deployment of the model, a Node-RED starter application was made in order to predict the life expectancy.

# 4. EXPERIMENTAL INVESTIGATIONS

After importing the data in the Jupyter notebook, I analyzed the dataset.

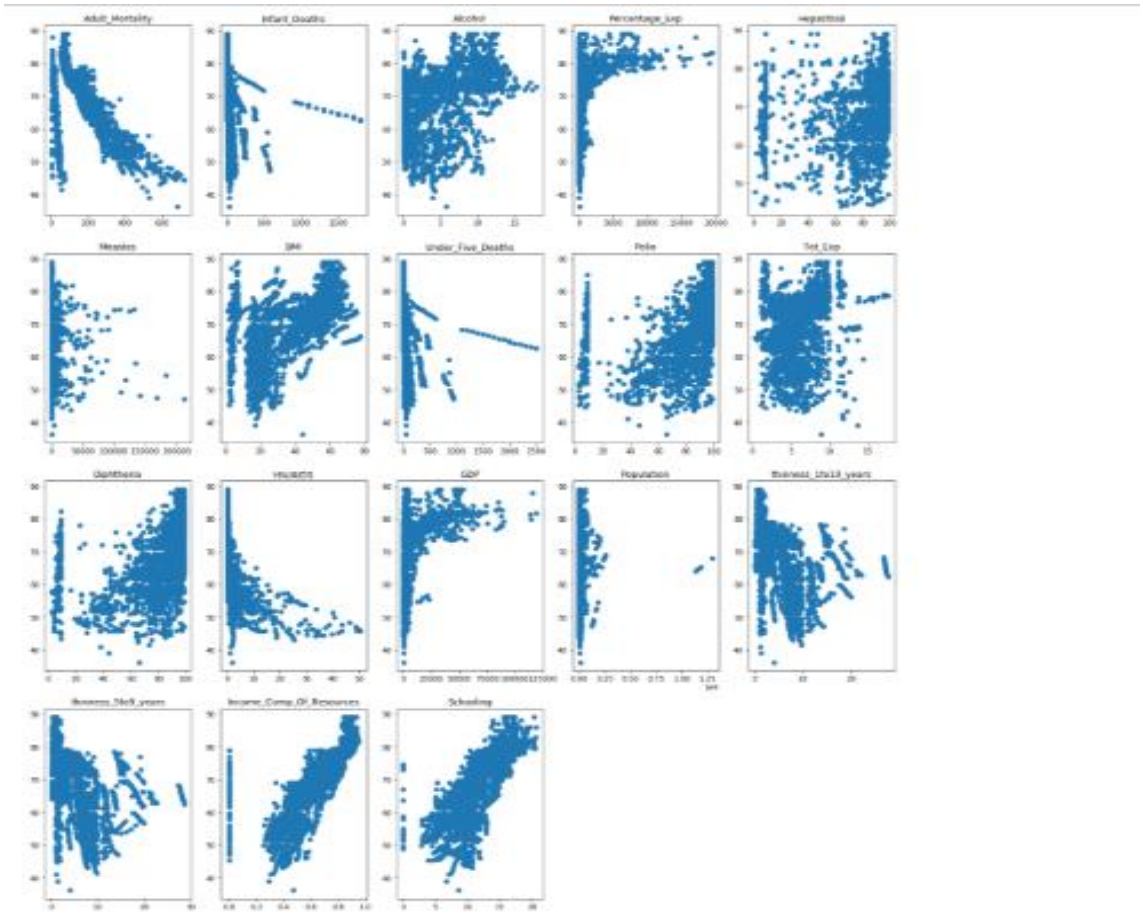
Variable Descriptions

Format: variable (type) - description

- country (Nominal) - the country in which the indicators are from (i.e. United States of America or Congo)
- year (Ordinal) - the calendar year the indicators are from (ranging from 2000 to 2015)
- status (Nominal) - whether a country is considered to be 'Developing' or 'Developed' by WHO standards
- life_expectancy (Ratio) - the life expectancy of people in years for a particular country and year
- adult_mortality (Ratio) - the adult mortality rate per 1000 population (i.e. number of people dying between 15 and 60 years per 1000 population); if the rate is 263 then that means 263 people will die out of 1000 between the ages of 15 and 60; another way to think of this is that the chance an individual will die between 15 and 60 is 26.3%
- infant_deaths (Ratio) - number of infant deaths per 1000 population; similar to above, but for infants
- alcohol (Ratio) - a country's alcohol consumption rate measured as liters of pure alcohol consumption per capita
- percentage_expenditure (Ratio) - expenditure on health as a percentage of Gross Domestic Product (gdp)
- hepatitis_b (Ratio) - number of 1 year olds with Hepatitis B immunization over all 1 year olds in population
- measles (Ratio) - number of reported Measles cases per 1000 population
- bmi (Interval/Ordinal) - average Body Mass Index (BMI) of a country's total population
- under-five_deaths (Ratio) - number of people under the age of five deaths per 1000 population
- polio (Ratio) - number of 1 year olds with Polio immunization over the number of all 1 year olds in population
- total_expenditure (Ratio) - government expenditure on health as a percentage of total government expenditure

- diphtheria (Ratio) - Diphtheria tetanus toxoid and pertussis (DTP3) immunization rate of 1 year olds
- hiv/aids (Ratio) - deaths per 1000 live births caused by HIV/AIDS for people under 5; number of people under 5 who die due to HIV/AIDS per 1000 births
- gdp (Ratio) - Gross Domestic Product per capita
- population (Ratio) - population of a country
- thinness_1-19_years (Ratio) - rate of thinness among people aged *10-19* (Note: variable should be renamed to *thinness_10-19_years* to more accurately represent the variable)
- thinness_5-9_years (Ratio) - rate of thinness among people aged 5-9
- income_composition_of_resources (Ratio) - Human Development Index in terms of income composition of resources (index ranging from 0 to 1)
- schooling (Ratio) - average number of years of schooling of a population

In Exploratory Data Analysis (EDA), I created simple plots with the help of seaborn library to analyze and interpret the data.



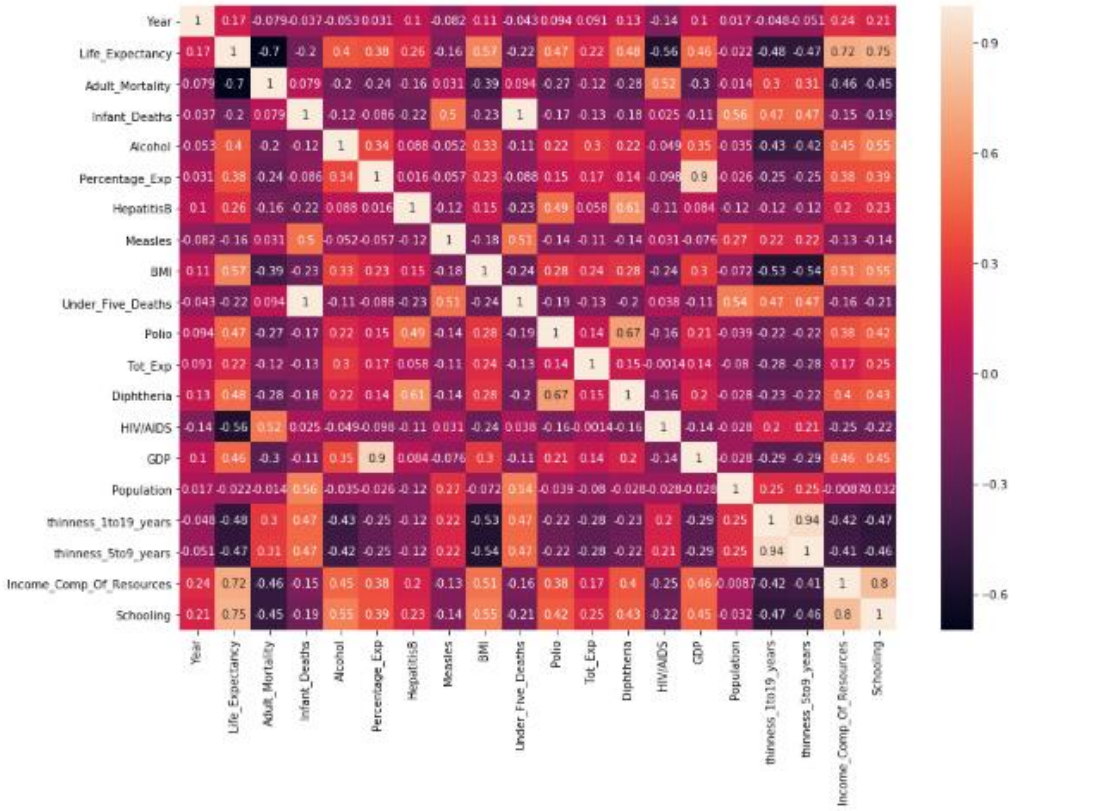
There seem to be a positive correlation between The Percentage of Healthcare Expenditure, Schooling, GDP and BMI and Life Expectancy, while there is a negative one between Adult

Mortality, AIDS and Life Expectancy, there does not seem to have any correlation between Alcohol, under 5 years – old deaths and Life Expectancy.

A heatmap further showed the correlation between different columns.

```
In [85]: plt.figure(figsize = (14, 10))
         sns.heatmap(dataset.corr(), annot = True)
         #sns.heatmap(dataset.corr())
```

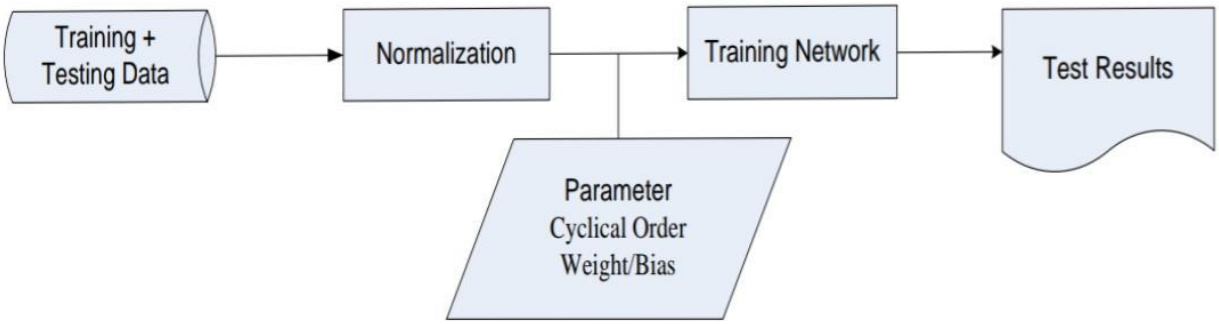
```
Out[85]: <matplotlib.axes._subplots.AxesSubplot at 0x7fe4e00fdb70>
```




The legend tells that the warmer colors show higher and positive correlation, while the colder low or negative.

There is a very high correlation between thinness of 5-9-year-old and that of 1-19-year-old. Also, between population and infant deaths, under 5 deaths, another is between schooling and income composition of resources. On the other hand, Life expectancy and Adult Mortality are very highly negatively correlated.

## 5. FLOWCHART

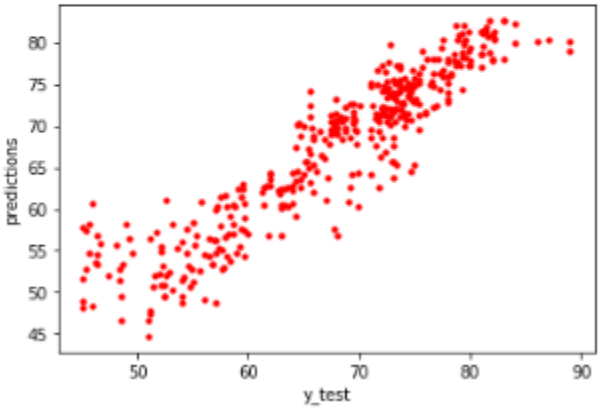



## 6. RESULT

After training the model with multiple linear regression, I plotted my graph as follows:

```
#plt.scatter(x, y, s=10)
plt.scatter(yvalid, predictions, s=10, color='r')
    #plt.scatter(Xvalid, yvalid, s=10)
plt.xlabel('y_test')
plt.ylabel('predictions')

    # predicted values
    #plt.plot(X_valid, y_predicted, color='r')
plt.show()
```



The evaluation metrics that are used are Mean Absolute Error, Mean Squared Error, Root Mean Squared Error and Accuracy.

Mean Absolute Error (MSE) of a model refers to the mean of the absolute values of each prediction error on all instances of the test data-set. It's the average over the test sample of the absolute differences between prediction and actual observation where all individual differences have equal weight.

Mean Squared Error (MSE): MSE is the average of the squared error that is used as the loss function for least squares regression: It is the sum, over all the data points, of the square of the difference between the predicted and actual target variables, divided by the number of data points. MSE is calculated by taking the average of the square of the difference between the original and predicted values of the data.

Root mean squared error (RMSE): RMSE is a quadratic scoring rule that also measures the average magnitude of the error. It's the square root of the average of squared differences between prediction and actual observation.

Classification Accuracy is what we usually mean, when we use the term accuracy. It is the ratio of number of correct predictions to the total number of input samples.

```
print('MAE:', round(metrics.mean_absolute_error(y_test, predictions),2))
print('MSE:', round(metrics.mean_squared_error(y_test, predictions),2))
print('RMSE:', round(np.sqrt(metrics.mean_squared_error(y_test, predictions)),2))
print("R2 Score (Accuracy):", round(metrics.r2_score(y_test, predictions)*100,2,),"%")
```

```
MAE: 2.76
MSE: 13.91
RMSE: 3.73
R2 Score (Accuracy): 86.04 %
```

My model has an accuracy of 86.04% for prediction of life expectancy.

After deployment of the model on IBM Cloud, I built a NODE-RED flow to integrate the Machine Learning Services.

Node-RED is a flow-based development tool for visual programming developed originally by IBM for wiring together hardware devices, APIs and online services as part of the Internet of Things. Node-RED provides a web browser-based flow editor, which can be used to create JavaScript functions. Elements of applications can be saved or shared for re-use. The runtime is built on Node.js. The flows created in Node-RED are stored using JSON.

With its browser-based editor you can simply wire together hardware devices, APIs and online services to create your application.

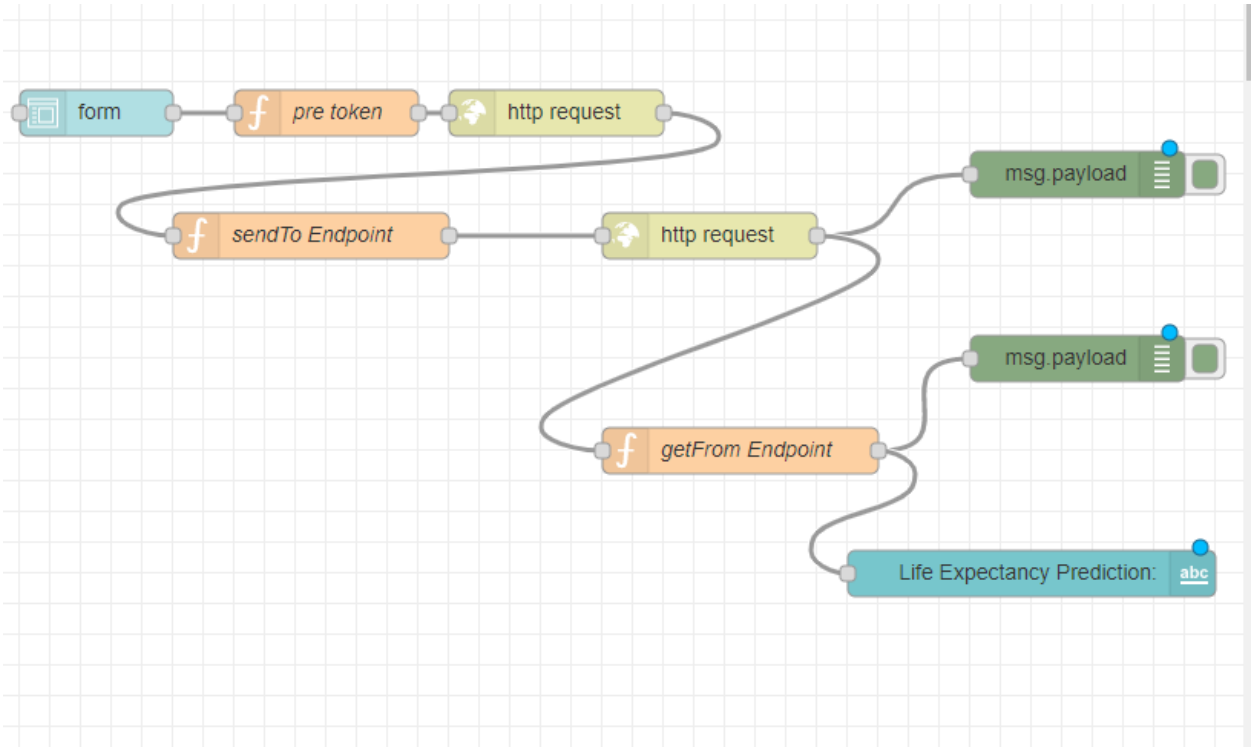
The Node-RED runtime is lightweight and built on top of Node.js. It takes full advantage of Node.js' event-driven, non-blocking I/O model. There is also the added benefit of tapping into the most used programming language — JavaScript.
A Node-RED node consists of three main files:

package.json: standard file used by Node.js modules, but with an added node-red section

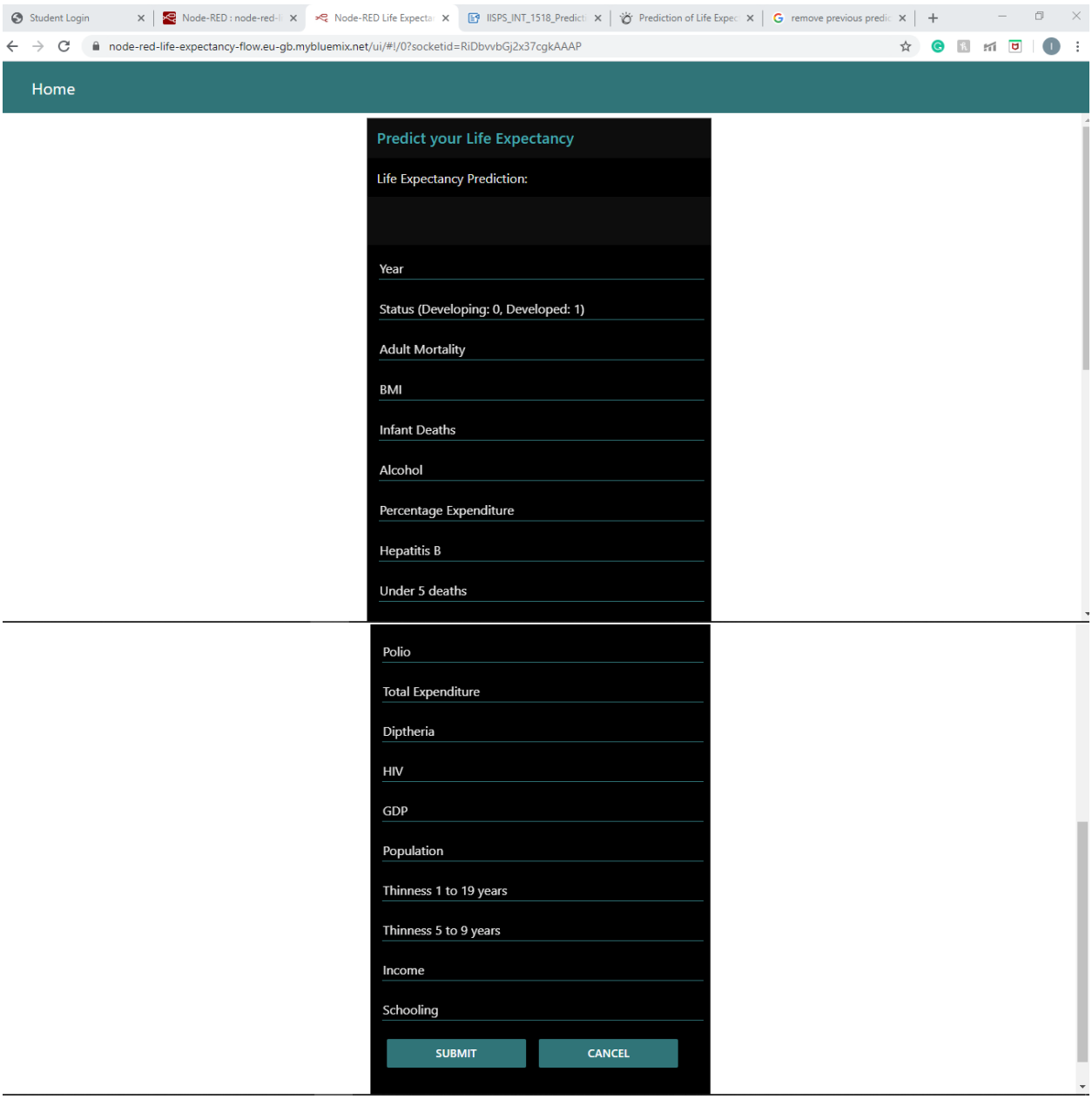
JavaScript file that defines the node's behavior

HTML file that defines the node's properties, edit dialog and help text

Flow:




The web UI/ output is as follows:


Student Login
Node-RED : node-red-l
Node-RED Life Expecta
IISPS_INT_1518_Predict
Prediction of Life Expec
remove previous predic
node-red-life-expectancy-flow.eu-gb.mybluemix.net/ui/#!/0?socketid=RiDbvvbGj2x37cgkAAAP
Home
Predict your Life Expectancy
Life Expectancy Prediction:
Year
Status (Developing: 0, Developed: 1)
Adult Mortality
BMI
Infant Deaths
Alcohol
Percentage Expenditure
Hepatitis B
Under 5 deaths
Polio
Total Expenditure
Diptheria
HIV
GDP
Population
Thinness 1 to 19 years
Thinness 5 to 9 years
Income
Schooling
SUBMIT
CANCEL

# 7. ADVANTAGES AND DISADVANTAGES

Predicting the lifespan of people would greatly alter our lives. Life expectancy predictions have the potential to be beneficial to individuals, health service providers and governments. For instance, they would make people more aware of their general health, and its improvement or deterioration over time. This may motivate them to make healthier lifestyle choices.

They could also be used by insurance companies to provide individualised services, such as how some car insurance companies use black-box technology to reduce premiums for more cautious drivers.

Governments may be able to use predictions to more efficiently allocate limited resources, such as social welfare assistance and health care funding, to individuals and areas of greater need.

On one hand, it may have benefits for policy making, and help optimise an individual's health, or the services they receive. But the potential misuse of this information by the government or private sector poses major risks to our rights and privacy.

People may become distressed if their life expectancy is unexpectedly low, or at the thought of having one at all. This raises concerns about how such predictions could impact those who experience or are at risk of mental health problems.

Having people's detailed health data could also let insurance companies more accurately profile applicants, leading to discrimination against groups or individuals.

Also, pharmaceutical companies could coordinate targeted medical campaigns based on people's life expectancy. And governments could choose to tax individuals differently, or restrict services for certain people.

Dramatic changes to medicine, technology, food habits, can and do happen to populations over the course of a life time.
It really doesn't tell you much unless you start looking at dozens of variables individually.
The main disadvantage is that no one can predict the future. No one knows when someone will die, who will get cancer or not, who will recover and who won't. Statistics work in generalities. Humans, however, do not.

# 8. APPLICATIONS

Life expectancy predictions have the potential to be beneficial to individuals, health service providers and governments.
For instance, they would make people more aware of their general health, and its improvement or deterioration over time. This may motivate them to make healthier lifestyle choices.

They could also be used by insurance companies to provide individualised services, such as how some car insurance companies use black-box technology to reduce premiums for more cautious drivers.

Governments may be able to use predictions to more efficiently allocate limited resources, such as social welfare assistance and health care funding, to individuals and areas of greater need.

# 9. CONCLUSION

In a sample involving a large range of countries we find that life expectancy is affected by many variables, some of which suggest that life expectancy can continue to rise in both developed and developing countries. While there may be an upper limit to life expectancy, in the absence of fundamental breakthroughs in anti-aging research, the empirical analysis here suggests that we are not at that limit yet.
More importantly, from an immediate policy perspective, are the indications that fairly low-cost policies (e.g. enhanced provision of water, medical drugs, or AIDS education/care) can lead to dramatic improvements in life expectancy in developing countries. While doubtless the case that increased income levels would result in more or less automatic improvements in provision of water and medical care, we show here that, holding income constant, great improvements are possible with the expenditure of what would globally be regarded as trivial amounts of resources. This is important, since the solution to the general development problem has been an intractable goal.

# 10. FUTURE SCOPE

Government policies affecting water quality and health care have many benefits, such as morbidity and workdays lost, benefits not limited to the life expectancy benefits focussed on here. However, the life expectancy benefits of affecting those variables should be added to other benefits of education, improved access to clean water and drugs, and AIDS prevention. Doing so offers the potential to result in better allocation of costly scarce resources in countries at various stages of development.
Future research could build a forecasting model incorporating mortality trends along with the impacts of socioeconomic variables investigated here, as well as others that are likely to become available as data measurement and accessibility improves over time.

# 11. BIBLIOGRAPHY

## REFERENCES


https://www.thelancet.com/journals/lancet/article/PIIS0140-6736(18)31694-5/fulltext

https://bmcmedinformdecismak.biomedcentral.com/articles/10.1186/s12911-019-0775-2

https://iopscience.iop.org/article/10.1088/1742-6596/1255/1/012017/pdf

https://www.sciencedirect.com/science/article/abs/pii/S1431761304700919

https://www.ncbi.nlm.nih.gov/pmc/articles/PMC6223892/

https://ourworldindata.org/life-expectancy

# APPENDIX

## A. Source Code

PredictionOfLifeExpectancy.ipynb

```
import numpy as np
import matplotlib.pyplot as plt
import pandas as pdimport seaborn as sns
%matplotlib inline
```

In [2]:

```
import typesimport pandas as pd
from botocore.client import Configimport ibm_boto3
```

In [7]:

```
body =
client_d958e4555edf4dfd9c3d98016077ec93.get_object(Bucket='internshipproject-
donotdelete-pr-p35kai233wqxlk',Key='Life Expectancy Data.csv')['Body']# add
missing __iter__ method, so pandas accepts body as file-like objectif not
hasattr(body, "__iter__"): body.__iter__ = types.MethodType( __iter__, body
)dataset = pd.read_csv(body)

dataset.head()
```

Out[7]:

| | Country | Year | Status | Lifeexpectancy | AdultMortality | infantdeaths | Alcohol | percentageexpenditure | HepatitisB | Measles | ... | Polio | Totalexpenditure | Diphtheria | HIV/AIDS | GDP | Population | thinness1-19years | thinness5-9years | Incomecompositionofresources | Schooling |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|

| | | | | | | | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **0** | Afghanistan | 2015 | Developing | 65.0 | 2630 | 62 | 0.01 | 71.2796 24 | 65.0 | 1154 | ... | 6.0 | 8.16 | 65.0 | 0.1 | 584.2592 10 | 3373649 4.0 | 17.2 | 17.3 | 0.479 | 10.1 |
| **1** | Afghanistan | 2014 | Developing | 59.9 | 2710 | 64 | 0.01 | 73.5235 82 | 62.0 | 492 | ... | 58.0 | 8.18 | 62.0 | 0.1 | 612.6965 14 | 3275 82.0 | 17.5 | 17.5 | 0.476 | 10.0 |
| **2** | Afghanistan | 2013 | Developing | 59.9 | 2680 | 66 | 0.01 | 73.2192 43 | 64.0 | 430 | ... | 62.0 | 8.13 | 64.0 | 0.1 | 631.7449 76 | 3173 1688 8.0 | 17.7 | 17.7 | 0.470 | 9.9 |
| **3** | Afghanistan | 2012 | Developing | 59.5 | 2720 | 69 | 0.01 | 78.1842 15 | 67.0 | 2787 | ... | 67.0 | 8.52 | 67.0 | 0.1 | 669.9590 000 | 3696 9586 .0 | 17.9 | 18.0 | 0.463 | 9.8 |
| **4** | Afghanistan | 2011 | Developing | 59.2 | 2750 | 71 | 0.01 | 7.0971 09 | 68.0 | 3013 | ... | 68.0 | 7.87 | 68.0 | 0.1 | 63.53 7231 | 2978 599 .0 | 18.2 | 18.2 | 0.454 | 9.5 |

5 rows × 22 columns

In [8]:

```
dataset.columns
```

Out[8]:

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

In [9]:

```
dataset.info()
```

```
<class 'pandas.core.frame.DataFrame'>RangeIndex: 2938 entries, 0 to 2937Data
columns (total 22 columns):Country                           2938 non-null
objectYear                                2938 non-null int64Status
2938 non-null objectLife expectancy                   2928 non-null float64Adult
Mortality                    2928 non-null float64infant deaths
2938 non-null int64Alcohol                            2744 non-null
float64percentage expenditure            2938 non-null float64Hepatitis B
2385 non-null float64Measles                             2938 non-null int64 BMI
2904 non-null float64under-five deaths                   2938 non-null int64Polio
2919 non-null float64Total expenditure                   2712 non-null
float64Diphtheria                         2919 non-null float64 HIV/AIDS
2938 non-null float64GDP                                 2490 non-null
float64Population                         2286 non-null float64 thinness  1-19
years              2904 non-null float64 thinness 5-9 years               2904
non-null float64Income composition of resources    2771 non-null float64Schooling
2775 non-null float64dtypes: float64(16), int64(4), object(2)memory usage: 505.0+
KB
```

In [10]:

```
dataset.describe()
```

Out[10]:

| | Year | Life expectancy | Adult Mortality | infant deaths | Alcohol | percentage expenditure | Hepatitis B | Measles | BMI | under-five deaths | Polio | Total expenditure | Diphtheria | HIV/AIDS | GDP | Population | thinness 1-19 years | thinness 5-9 years | Income composition of | Schooling |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|

| | | | | | | | | | | | | | | | | | | | | resources | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **count** | 2938.000000 | 2928.000000 | 2928.000000 | 2938.000000 | 2744.000000 | 2938.000000 | 2385.000000 | 2938.000000 | 2904.000000 | 2938.000000 | 2919.000000 | 2712.000000 | 2919.000000 | 2938.000000 | 2490.000000 | 2.286000e+03 | 2904.000000 | 2904.000000 | 2771.000000 | 2775.000000 |
| **mean** | 2007.518720 | 69.224932 | 164.7964648 | 30.303948 | 4.60286161 | 738.2512951 | 80.94046 1 | 2419.592240 | 38.321247 | 42.035739 | 82.55018 8 | 5.938189 | 82.324084 | 1.74210 3 | 7483.158469 | 1.27538e+07 | 4.83397 04 | 4.87031 7 | 0.62751 | 11.992793 |
| **std** | 4.61384 1 | 9.52387 | 124.2920 79 | 117.9265 01 | 4.05241 3 | 1987.914858 | 25.070 016 | 11467.272489 | 20.044034 | 160.4455 48 | 23.428046 | 2.49832 | 23.716912 | 5.07778 5 | 14270.1693 42 | 6.10121 10e+07 | 4.42 0195 | 4.50888 2 | 0.21 0904 | 3.358920 |
| **min** | 2000.000000 | 36.300000 | 1.000000 | 0.000000 | 0.010000 | 0.000000 | 1.000000 | 0.000000 | 1.000000 | 0.000000 | 3.000000 | 0.37000 0 | 2.000000 | 0.10000 0 | 1.681350 | 3.40000000e+01 | 0.10000 0 | 0.10000 0 | 0.00000 0 | 0.000000 |
| **25%** | 2004.000000 | 63.100000 | 74.000000 | 0.000000 | 0.875000 | 4.685343 | 77.000000 | 0.000000 | 19.300000 | 0.000000 | 78.000000 | 4.26000 0 | 78.000000 | 0.10000 0 | 463.935626 | 1.95793 2e+05 | 1.60000 0 | 1.50000 0 | 0.49300 0 | 10.100000 |
| **50%** | 2008.000000 | 72.100000 | 144.000000 | 3.000000 | 3.755000 | 64.912906 | 92.000000 | 17.000000 | 43.500000 | 4.000000 | 93.000000 | 5.75500 | 93.000000 | 0.10000 0 | 1766.947595 | 1.38654 2e+06 | 3.30000 0 | 3.30000 0 | 0.67700 0 | 12.300000 |
| **75%** | 2012.000000 | 75.700000 | 228.000000 | 22.000000 | 7.702500 | 441.534144 | 97.000000 | 360.250000 | 56.200000 | 28.000000 | 97.000000 | 7.49250 | 97.000000 | 0.80000 0 | 5910.806335 | 7.42 0359e+06 | 7.20000 0 | 7.20000 0 | 0.77900 0 | 14.300000 |

| | | | | | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **max** | 2015.000000 | 89.000000 | 723.000000 | 1800.000000 | 17.870000 | 19479.911610 | 99.000000 | 2121830.000000 | 87.300000 | 25000.000000 | 99.000000 | 17.600000 | 99.000000 | 50.600000 | 1191727.741800 | 1.298859e+09 | 27.700000 | 28.600000 | 0.948000 | 20.700000 |

In [11]:

```
dataset.rename(columns={" BMI ":"BMI","Life expectancy ":"Life_Expectancy","Adult Mortality":"Adult_Mortality",                    "infant deaths":"Infant_Deaths","percentage expenditure":"Percentage_Exp","Hepatitis B":"HepatitisB",                  "Measles ":"Measles"," BMI ":"BMI","under-five deaths ":"Under_Five_Deaths","Diphtheria ":"Diphtheria",                    " HIV/AIDS":"HIV/AIDS"," thinness  1-19 years":"thinness_1to19_years"," thinness 5-9 years":"thinness_5to9_years","Income composition of resources":"Income_Comp_Of_Resources",                  "Total expenditure":"Tot_Exp"},inplace=True)
```

In [12]:

```
dataset.columns
```

Out[12]:

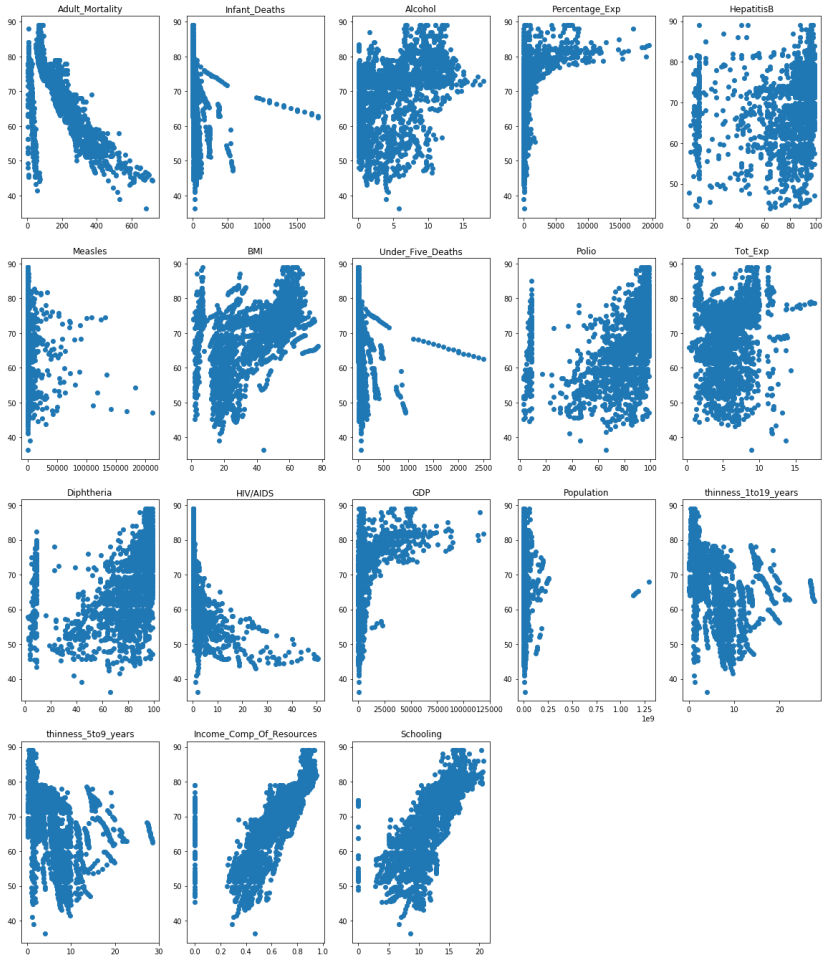
```
Index(['Country', 'Year', 'Status', 'Life_Expectancy', 'Adult_Mortality',
'Infant_Deaths', 'Alcohol', 'Percentage_Exp', 'HepatitisB', 'Measles',
'BMI', 'Under_Five_Deaths', 'Polio', 'Tot_Exp', 'Diphtheria',       'HIV/AIDS',
'GDP', 'Population', 'thinness_1to19_years',       'thinness_5to9_years',
'Income_Comp_Of_Resources', 'Schooling'],      dtype='object')
```

# EDA

## Creating simple plots to check out the data

In [13]:

```
col_dict =
{'Adult_Mortality':1,'Infant_Deaths':2,'Alcohol':3,'Percentage_Exp':4,'HepatitisB':5,'Measles':6,'BMI':7,'Under_Five_Deaths':8,'Polio':9,'Tot_Exp':10,'Diphtheria':11,'HIV/AIDS':12,'GDP':13,'Population':14,'thinness_1to19_years':15,'thinness_5to9_years':16,'Income_Comp_Of_Resources':17,'Schooling':18}# Detect outliers in each variable using box plots.plt.figure(figsize=(20,30))for variable,i in col_dict.items():                     plt.subplot(5,5,i)
plt.scatter(dataset[variable], dataset['Life_Expectancy'])
plt.title(variable)plt.show()
```

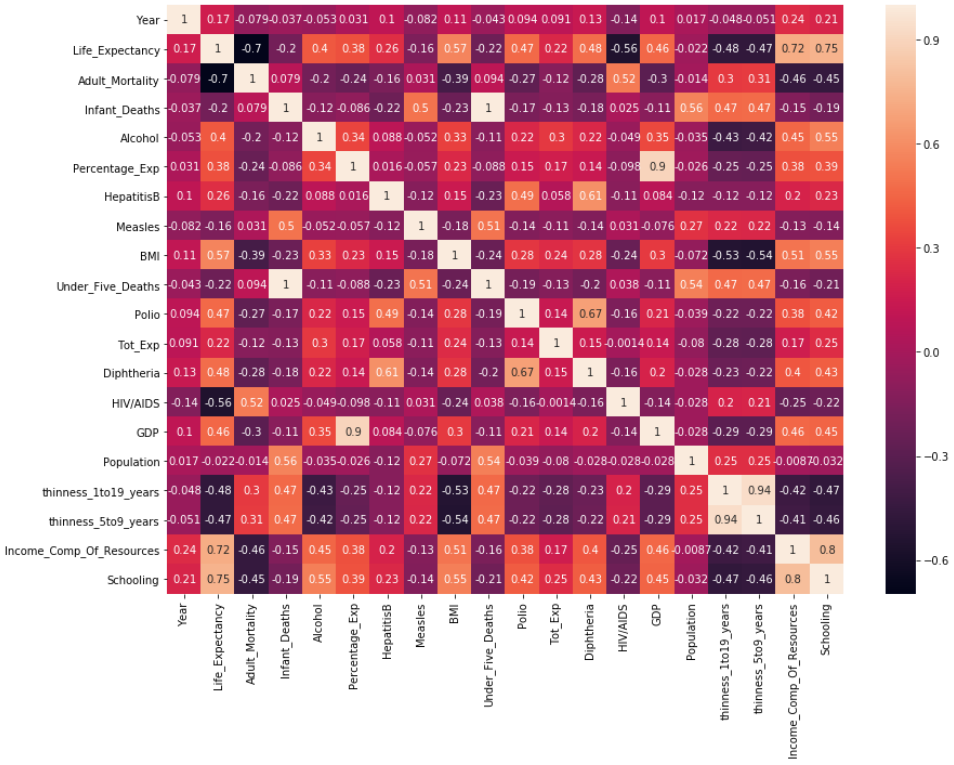



In [14]:

```
plt.figure(figsize = (14, 10))sns.heatmap(dataset.corr(), annot =
True)#sns.heatmap(dataset.corr())
```

Out[14]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f3c65095e10>
```




# Finding out null Values

In [15]:

```
dataset.isnull().sum()
```

Out[15]:

```
Country                          0Year                            0Status
0Life_Expectancy                 10Adult_Mortality                10Infant_Deaths
0Alcohol                         194Percentage_Exp                 0HepatitisB
553Measles                           0BMI
34Under_Five_Deaths              0Polio                           19Tot_Exp
226Diphtheria                    19HIV/AIDS                         0GDP
448Population                    652thinness_1to19_years
34thinness_5to9_years            34Income_Comp_Of_Resources       167Schooling
163dtype: int64
```

In [16]:

```
dataset.columns
```

Out[16]:

```
Index(['Country', 'Year', 'Status', 'Life_Expectancy', 'Adult_Mortality',
'Infant_Deaths', 'Alcohol', 'Percentage_Exp', 'HepatitisB', 'Measles',
'BMI', 'Under_Five_Deaths', 'Polio', 'Tot_Exp', 'Diphtheria',       'HIV/AIDS',
```

```
'GDP', 'Population', 'thinness_1to19_years',       'thinness_5to9_years',
'Income_Comp_Of_Resources', 'Schooling'],      dtype='object')
```

In [17]:

```
country_list = dataset.Country.unique()fill_list =
['Life_Expectancy','Adult_Mortality','Alcohol','HepatitisB','BMI','Polio','Tot_Ex
p','Diphtheria','GDP','Population','thinness_1to19_years','thinness_5to9_years','
Income_Comp_Of_Resources','Schooling']##Interpolationfor country in country_list:
dataset.loc[dataset['Country'] == country,fill_list] =
dataset.loc[dataset['Country'] == country,fill_list].interpolate()  ##Drop cols
with missing valuesdataset.dropna(inplace=True)
```

In [18]:

```
dataset.isnull().sum()
```

Out[18]:

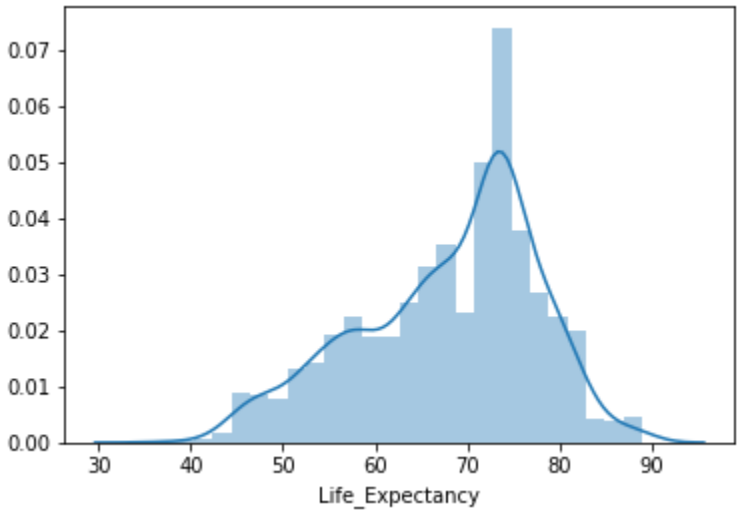
```
Country                          0Year                            0Status
0Life_Expectancy                  0Adult_Mortality                  0Infant_Deaths
0Alcohol                          0Percentage_Exp                   0HepatitisB
0Measles                          0BMI                              0Under_Five_Deaths
0Polio                            0Tot_Exp                          0Diphtheria
0HIV/AIDS                         0GDP                              0Population
0thinness_1to19_years             0thinness_5to9_years
0Income_Comp_Of_Resources         0Schooling                        0dtype: int64
```

In [19]:

```
sns.distplot(dataset['Life_Expectancy'])
```

Out[19]:

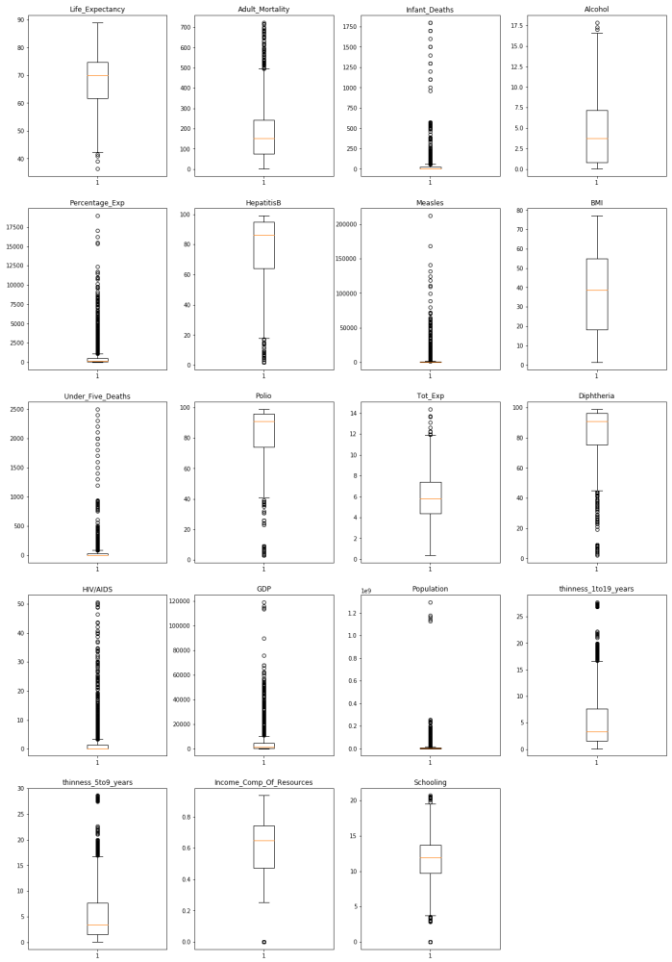
```
<matplotlib.axes._subplots.AxesSubplot at 0x7f3c63518518>
```



# Finding out the outliers for each variable.

In [20]:

```
# Create a dictionary of columns.col_dict =
{'Life_Expectancy':1,'Adult_Mortality':2,'Infant_Deaths':3,'Alcohol':4,'Percentag
e_Exp':5,'HepatitisB':6,'Measles':7,'BMI':8,'Under_Five_Deaths':9,'Polio':10,'Tot
_Exp':11,'Diphtheria':12,'HIV/AIDS':13,'GDP':14,'Population':15,'thinness_1to19_y
ears':16,'thinness_5to9_years':17,'Income_Comp_Of_Resources':18,'Schooling':19}#
Detect outliers in each variable using box plots.plt.figure(figsize=(20,30))for
variable,i in col_dict.items():                    plt.subplot(5,4,i)
plt.boxplot(dataset[variable],whis=1.5)
plt.title(variable)plt.show()
```




In [21]:

```
for variable in col_dict.keys():    q75, q25 = np.percentile(dataset[variable],
[75 ,25])    iqr = q75 - q25    min_val = q25 - (iqr*1.5)    max_val = q75 +
(iqr*1.5)    print("Number of outliers and percentage of it in {} : {} and
{}".format(variable,
len((np.where((dataset[variable] > max_val) |
```

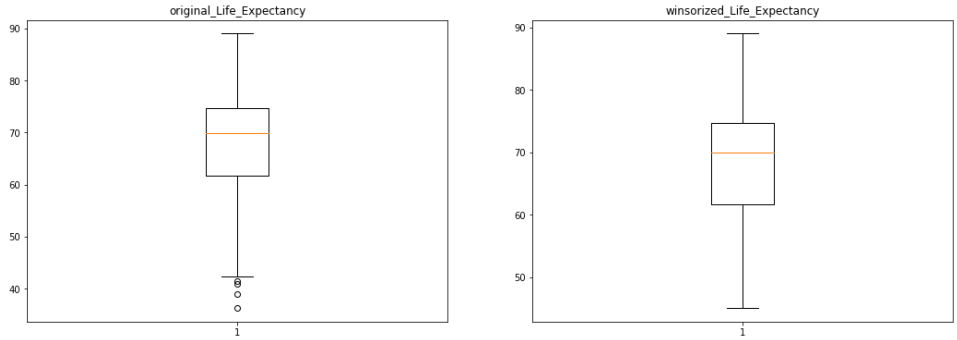
```
(dataset[variable] < min_val))[0])),len((np.where((dataset[variable] > max_val) |
(dataset[variable] < min_val))[0]))*100/1987))
```

```
Number of outliers and percentage of it in Life_Expectancy : 4 and
0.20130850528434827Number of outliers and percentage of it in Adult_Mortality :
58 and 2.9189733266230498Number of outliers and percentage of it in Infant_Deaths
: 198 and 9.96477101157524Number of outliers and percentage of it in Alcohol : 3
and 0.1509813789632612Number of outliers and percentage of it in Percentage_Exp :
232 and 11.675893306492199Number of outliers and percentage of it in HepatitisB :
216 and 10.870659285354806Number of outliers and percentage of it in Measles :
361 and 18.16809260191243Number of outliers and percentage of it in BMI : 0 and
0.0Number of outliers and percentage of it in Under_Five_Deaths : 227 and
11.424257674886764Number of outliers and percentage of it in Polio : 159 and
8.002013085052843Number of outliers and percentage of it in Tot_Exp : 13 and
0.6542526421741318Number of outliers and percentage of it in Diphtheria : 195 and
9.813789632611979Number of outliers and percentage of it in HIV/AIDS : 309 and
15.551082033215904Number of outliers and percentage of it in GDP : 244 and
12.279818822345245Number of outliers and percentage of it in Population : 260 and
13.085052843482638Number of outliers and percentage of it in thinness_1to19_years
: 70 and 3.5228988424760947Number of outliers and percentage of it in
thinness_5to9_years : 75 and 3.77453447408153Number of outliers and percentage of
it in Income_Comp_Of_Resources : 91 and 4.579768495218923Number of outliers and
percentage of it in Schooling : 32 and 1.6104680422747861
```

## Winsorization of Outliers

In [22]:

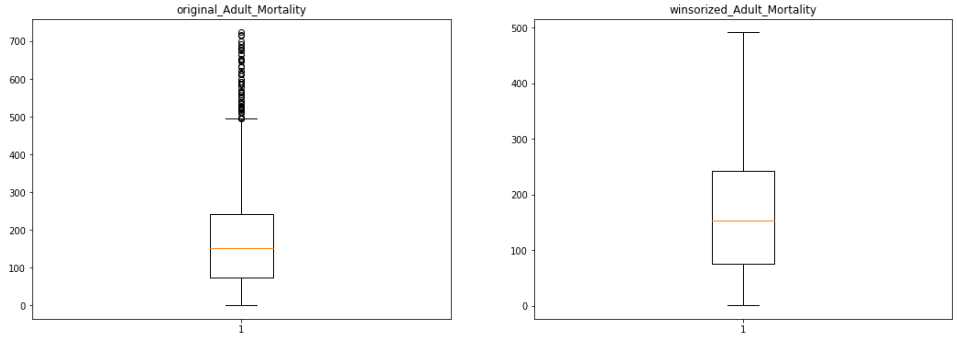
```
from scipy.stats.mstats import
winsorizeplt.figure(figsize=(18,6))plt.subplot(1,2,1)original_Life_Expectancy =
dataset['Life_Expectancy']plt.boxplot(original_Life_Expectancy)plt.title("origina
l_Life_Expectancy")plt.subplot(1,2,2)winsorized_Life_Expectancy =
winsorize(dataset['Life_Expectancy'],(0.01,0))plt.boxplot(winsorized_Life_Expecta
ncy)plt.title("winsorized_Life_Expectancy")plt.show()
```




In [23]:

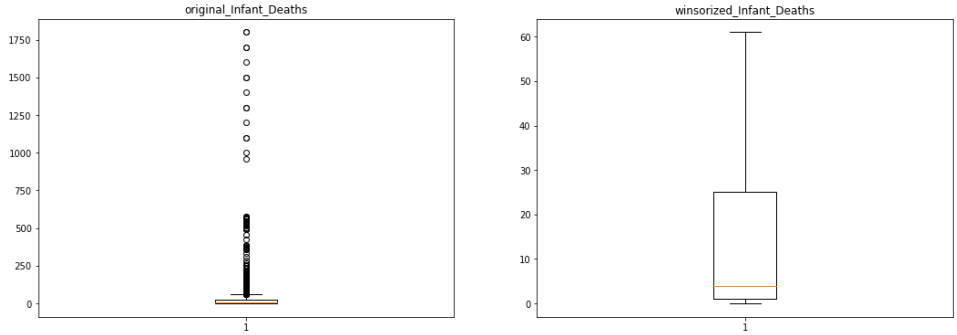
```
# Winsorize Adult_Mortalityfrom scipy.stats.mstats import
winsorizeplt.figure(figsize=(18,6))plt.subplot(1,2,1)original_Adult_Mortality =
dataset['Adult_Mortality']plt.boxplot(original_Adult_Mortality)plt.title("origina
```

```
l_Adult_Mortality")plt.subplot(1,2,2)winsorized_Adult_Mortality =
winsorize(dataset['Adult_Mortality'],(0,0.03))plt.boxplot(winsorized_Adult_Mortal
ity)plt.title("winsorized_Adult_Mortality")plt.show()
```



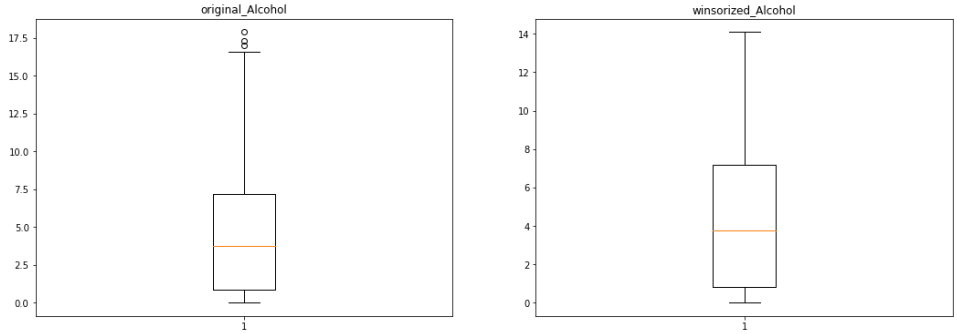

In [24]:

```
# Winsorize Infant_Deathsfrom scipy.stats.mstats import
winsorizeplt.figure(figsize=(18,6))plt.subplot(1,2,1)original_Infant_Deaths =
dataset['Infant_Deaths']plt.boxplot(original_Infant_Deaths)plt.title("original_In
fant_Deaths")plt.subplot(1,2,2)winsorized_Infant_Deaths =
winsorize(dataset['Infant_Deaths'],(0,0.10))plt.boxplot(winsorized_Infant_Deaths)
plt.title("winsorized_Infant_Deaths")plt.show()
```



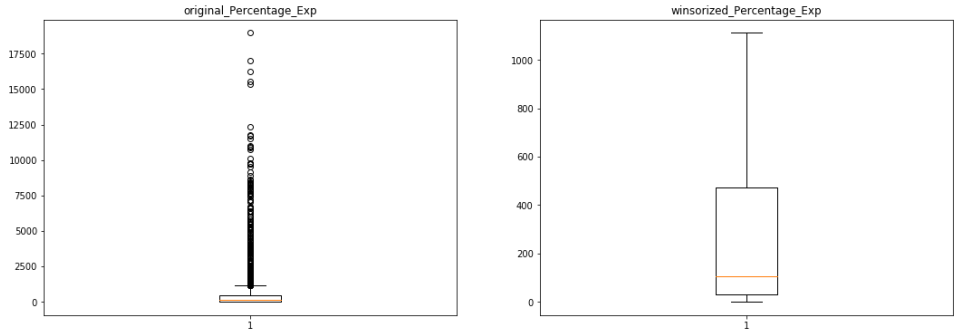

In [25]:

```
# Winsorize Alcoholfrom scipy.stats.mstats import
winsorizeplt.figure(figsize=(18,6))plt.subplot(1,2,1)original_Alcohol =
dataset['Alcohol']plt.boxplot(original_Alcohol)plt.title("original_Alcohol")plt.s
ubplot(1,2,2)winsorized_Alcohol =
winsorize(dataset['Alcohol'],(0,0.01))plt.boxplot(winsorized_Alcohol)plt.title("w
insorized_Alcohol")plt.show()
```



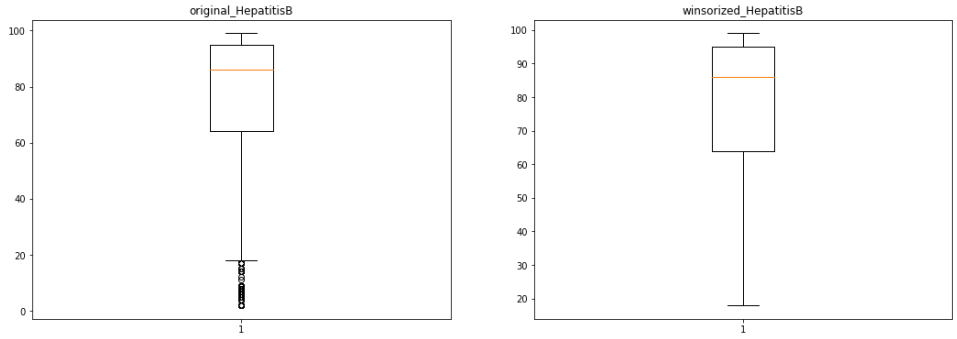

In [26]:

```
# Winsorize Percentage_Expfrom scipy.stats.mstats import
winsorizeplt.figure(figsize=(18,6))plt.subplot(1,2,1)original_Percentage_Exp =
dataset['Percentage_Exp']plt.boxplot(original_Percentage_Exp)plt.title("original_
Percentage_Exp")plt.subplot(1,2,2)winsorized_Percentage_Exp =
winsorize(dataset['Percentage_Exp'],(0,0.12))plt.boxplot(winsorized_Percentage_Ex
p)plt.title("winsorized_Percentage_Exp")plt.show()
```



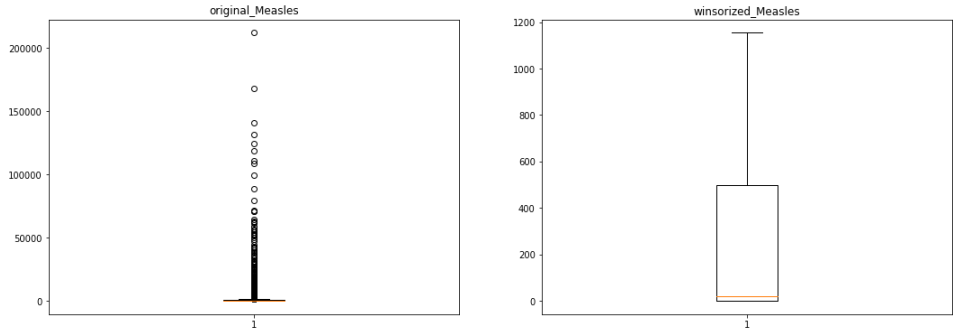

In [27]:

```
# Winsorize HepatitisBfrom scipy.stats.mstats import
winsorizeplt.figure(figsize=(18,6))plt.subplot(1,2,1)original_HepatitisB =
dataset['HepatitisB']plt.boxplot(original_HepatitisB)plt.title("original_Hepatiti
sB")plt.subplot(1,2,2)winsorized_HepatitisB =
winsorize(dataset['HepatitisB'],(0.11,0))plt.boxplot(winsorized_HepatitisB)plt.ti
tle("winsorized_HepatitisB")plt.show()
```




In [28]:

```
# Winsorize Measlesfrom scipy.stats.mstats import
winsorizeplt.figure(figsize=(18,6))plt.subplot(1,2,1)original_Measles =
dataset['Measles']plt.boxplot(original_Measles)plt.title("original_Measles")plt.s
ubplot(1,2,2)winsorized_Measles =
winsorize(dataset['Measles'],(0,0.19))plt.boxplot(winsorized_Measles)plt.title("w
insorized_Measles")plt.show()
```

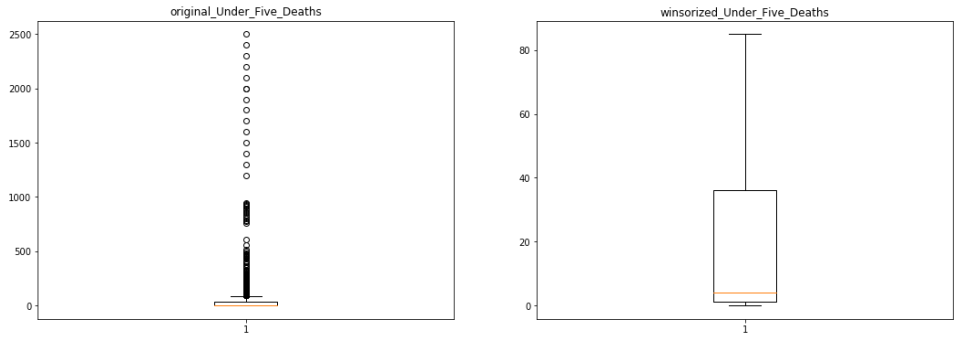



In [29]:

```
#Winsorization changes 19% of the data, which may not give better results. Hence
drop this column.dataset = dataset.drop('Measles',axis=1)
```

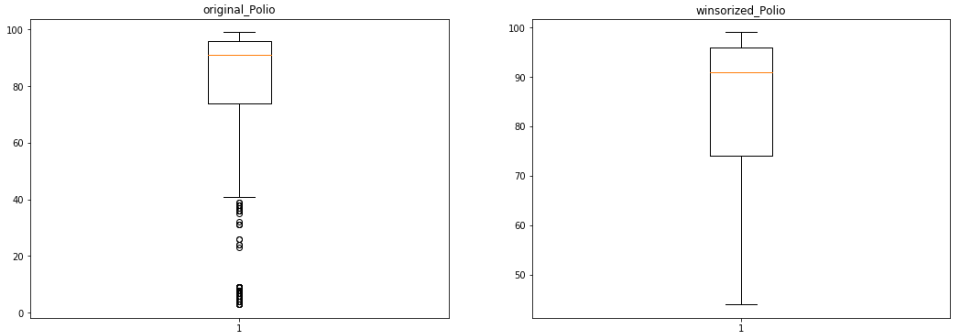
In [30]:

```
# Winsorize Under_Five_Deathsfrom scipy.stats.mstats import
winsorizeplt.figure(figsize=(18,6))plt.subplot(1,2,1)original_Under_Five_Deaths =
dataset['Under_Five_Deaths']plt.boxplot(original_Under_Five_Deaths)plt.title("ori
ginal_Under_Five_Deaths")plt.subplot(1,2,2)winsorized_Under_Five_Deaths =
winsorize(dataset['Under_Five_Deaths'],(0,0.12))plt.boxplot(winsorized_Under_Five
_Deaths)plt.title("winsorized_Under_Five_Deaths")plt.show()
```



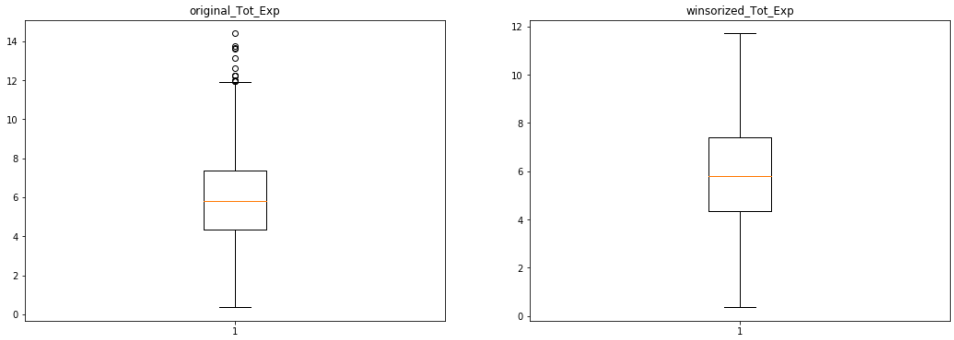

In [31]:

```
# Winsorize Poliofrom scipy.stats.mstats import
winsorizeplt.figure(figsize=(18,6))plt.subplot(1,2,1)original_Polio =
dataset['Polio']plt.boxplot(original_Polio)plt.title("original_Polio")plt.subplot
(1,2,2)winsorized_Polio =
winsorize(dataset['Polio'],(0.09,0))plt.boxplot(winsorized_Polio)plt.title("winso
rized_Polio")plt.show()
```



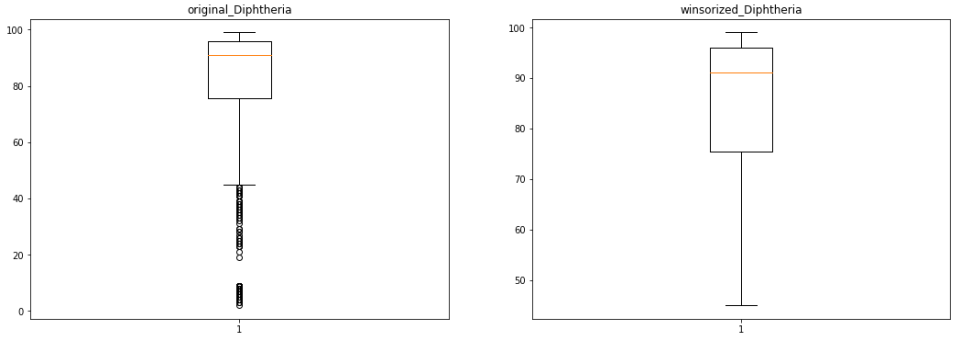

In [32]:

```
# Winsorize Tot_Expfrom scipy.stats.mstats import
winsorizeplt.figure(figsize=(18,6))plt.subplot(1,2,1)original_Tot_Exp =
dataset['Tot_Exp']plt.boxplot(original_Tot_Exp)plt.title("original_Tot_Exp")plt.s
ubplot(1,2,2)winsorized_Tot_Exp =
winsorize(dataset['Tot_Exp'],(0,0.01))plt.boxplot(winsorized_Tot_Exp)plt.title("w
insorized_Tot_Exp")plt.show()
```



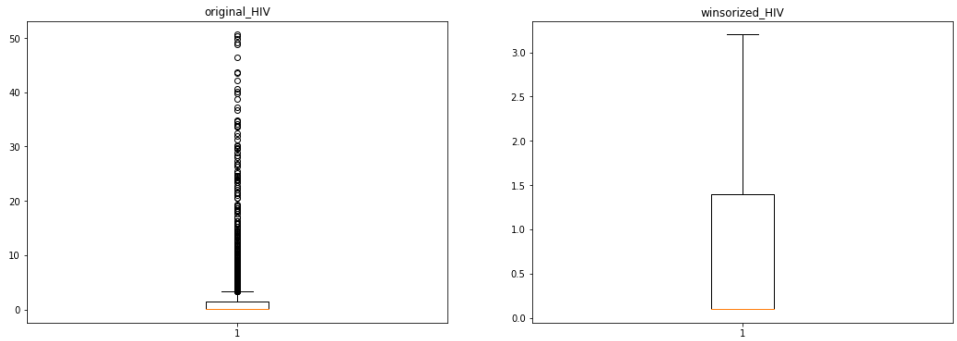

In [33]:

```
# Winsorize Diphtheriafrom scipy.stats.mstats import
winsorizeplt.figure(figsize=(18,6))plt.subplot(1,2,1)original_Diphtheria =
dataset['Diphtheria']plt.boxplot(original_Diphtheria)plt.title("original_Diphther
ia")plt.subplot(1,2,2)winsorized_Diphtheria =
winsorize(dataset['Diphtheria'],(0.10,0))plt.boxplot(winsorized_Diphtheria)plt.ti
tle("winsorized_Diphtheria")plt.show()
```



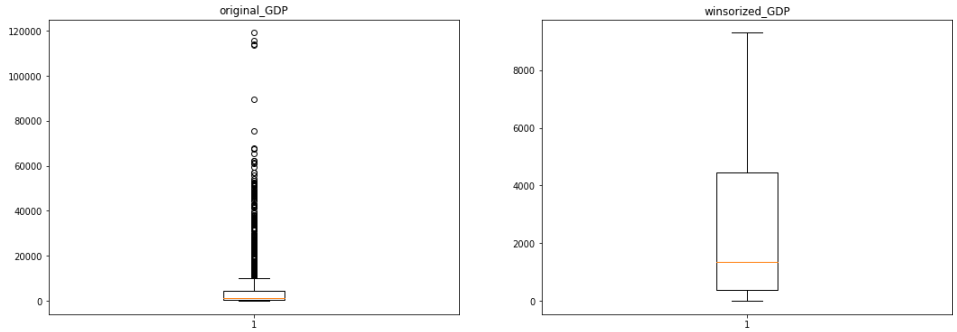
In [34]:

```
# Winsorize HIV/AIDSfrom scipy.stats.mstats import
winsorizeplt.figure(figsize=(18,6))plt.subplot(1,2,1)original_HIV =
dataset['HIV/AIDS']plt.boxplot(original_HIV)plt.title("original_HIV")plt.subplot(
1,2,2)winsorized_HIV =
winsorize(dataset['HIV/AIDS'],(0,0.16))plt.boxplot(winsorized_HIV)plt.title("wins
orized_HIV")plt.show()
```



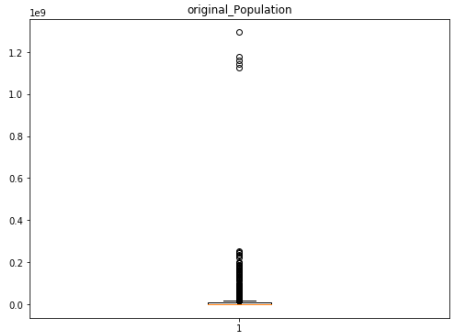

In [35]:

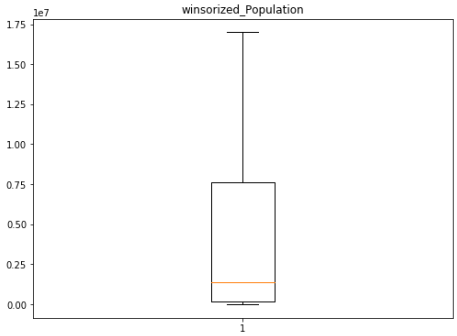
```
# Winsorize GDPfrom scipy.stats.mstats import
winsorizeplt.figure(figsize=(18,6))plt.subplot(1,2,1)original_GDP =
dataset['GDP']plt.boxplot(original_GDP)plt.title("original_GDP")plt.subplot(1,2,2
)winsorized_GDP =
winsorize(dataset['GDP'],(0,0.13))plt.boxplot(winsorized_GDP)plt.title("winsorize
d_GDP")plt.show()
```




In [36]:

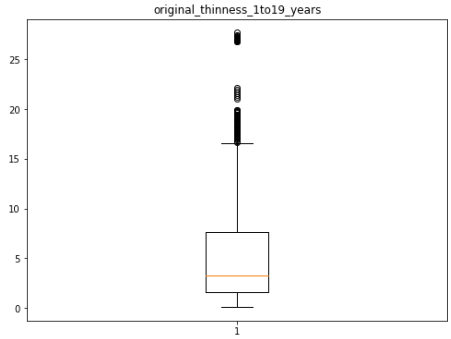
```
# Winsorize Populationfrom scipy.stats.mstats import
winsorizeplt.figure(figsize=(18,6))plt.subplot(1,2,1)original_Population =
dataset['Population']plt.boxplot(original_Population)plt.title("original_Populati
on")plt.subplot(1,2,2)winsorized_Population =
winsorize(dataset['Population'],(0,0.14))plt.boxplot(winsorized_Population)plt.ti
tle("winsorized_Population")plt.show()
```

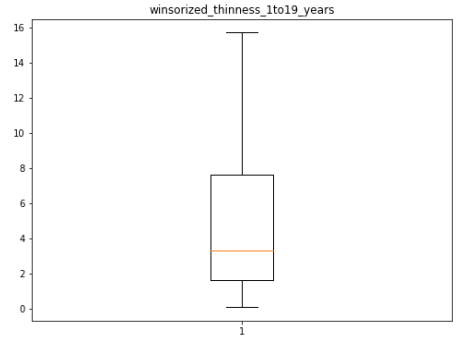



In [37]:

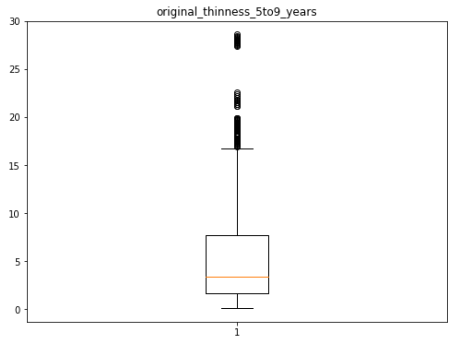
```
# Winsorize thinness_1to19_yearsfrom scipy.stats.mstats import
winsorizeplt.figure(figsize=(18,6))plt.subplot(1,2,1)original_thinness_1to19_year
s =
dataset['thinness_1to19_years']plt.boxplot(original_thinness_1to19_years)plt.titl
e("original_thinness_1to19_years")plt.subplot(1,2,2)winsorized_thinness_1to19_yea
rs =
winsorize(dataset['thinness_1to19_years'],(0,0.04))plt.boxplot(winsorized_thinnes
s_1to19_years)plt.title("winsorized_thinness_1to19_years")plt.show()
```

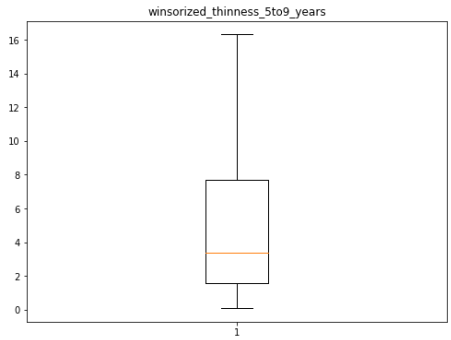



In [38]:

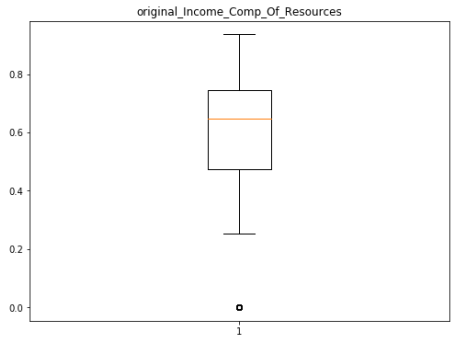
```
# Winsorize thinness_1to19_yearsfrom scipy.stats.mstats import
winsorizeplt.figure(figsize=(18,6))plt.subplot(1,2,1)original_thinness_5to9_years
=
dataset['thinness_5to9_years']plt.boxplot(original_thinness_5to9_years)plt.title(
"original_thinness_5to9_years")plt.subplot(1,2,2)winsorized_thinness_5to9_years =
winsorize(dataset['thinness_5to9_years'],(0,0.04))plt.boxplot(winsorized_thinness
_5to9_years)plt.title("winsorized_thinness_5to9_years")plt.show()
```

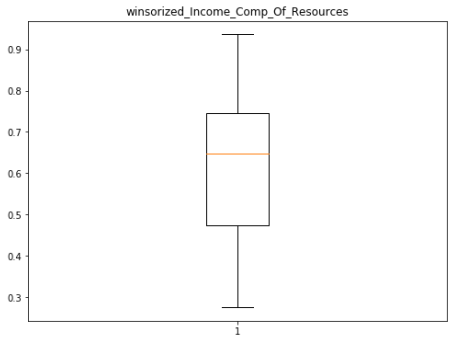



In [39]:

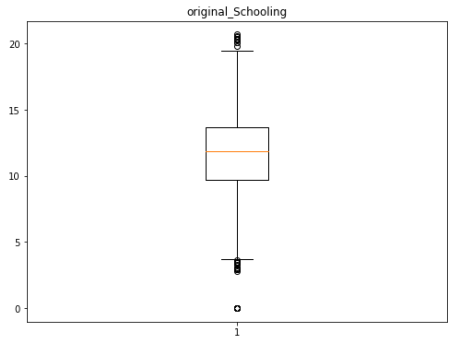
```
#Winsorize Income_Comp_Of_Resourcesfrom scipy.stats.mstats import
winsorizeplt.figure(figsize=(18,6))plt.subplot(1,2,1)original_Income_Comp_Of_Reso
urces =
dataset['Income_Comp_Of_Resources']plt.boxplot(original_Income_Comp_Of_Resources)
plt.title("original_Income_Comp_Of_Resources")plt.subplot(1,2,2)winsorized_Income
_Comp_Of_Resources =
winsorize(dataset['Income_Comp_Of_Resources'],(0.05,0))plt.boxplot(winsorized_Inc
ome_Comp_Of_Resources)plt.title("winsorized_Income_Comp_Of_Resources")plt.show()
```

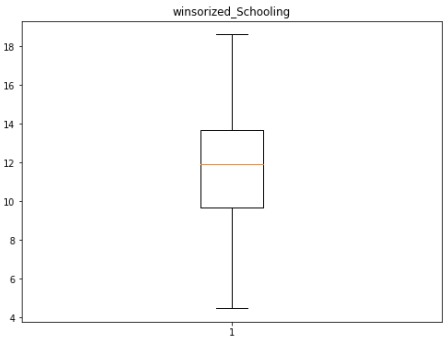



In [40]:

```
# Winsorize Schoolingfrom scipy.stats.mstats import
winsorizeplt.figure(figsize=(18,6))plt.subplot(1,2,1)original_Schooling =
dataset['Schooling']plt.boxplot(original_Schooling)plt.title("original_Schooling"
)plt.subplot(1,2,2)winsorized_Schooling =
winsorize(dataset['Schooling'],(0.02,0.01))plt.boxplot(winsorized_Schooling)plt.t
itle("winsorized_Schooling")plt.show()
```




## Adding winsorized variables to the data frame.

In [41]:

```
win_list =
[winsorized_Life_Expectancy,winsorized_Adult_Mortality,winsorized_Infant_Deaths,w
insorized_Alcohol,
winsorized_Percentage_Exp,winsorized_HepatitisB,winsorized_Under_Five_Deaths,wins
orized_Polio,winsorized_Tot_Exp,winsorized_Diphtheria,winsorized_HIV,winsorized_G
DP,winsorized_Population,winsorized_thinness_1to19_years,winsorized_thinness_5to9
_years,winsorized_Income_Comp_Of_Resources,winsorized_Schooling]for variable in
win_list:    q75, q25 = np.percentile(variable, [75 ,25])    iqr = q75 - q25
min_val = q25 - (iqr*1.5)    max_val = q75 + (iqr*1.5)        print("Number of
outliers after winsorization : {}".format(len(np.where((variable > max_val) |
(variable < min_val))[0])))
```

```
Number of outliers after winsorization : 0Number of outliers after winsorization
: 0Number of outliers after winsorization : 0Number of outliers after
winsorization : 0Number of outliers after winsorization : 0Number of outliers
after winsorization : 0Number of outliers after winsorization : 0Number of
outliers after winsorization : 0Number of outliers after winsorization : 0Number
of outliers after winsorization : 0Number of outliers after winsorization :
0Number of outliers after winsorization : 0Number of outliers after winsorization
: 0Number of outliers after winsorization : 0Number of outliers after
winsorization : 0Number of outliers after winsorization : 0Number of outliers
after winsorization : 0
```

In [42]:

```
dataset['winsorized_Life_Expectancy'] =
winsorized_Life_Expectancydataset['winsorized_Adult_Mortality'] =
winsorized_Adult_Mortalitydataset['winsorized_Infant_Deaths'] =
winsorized_Infant_Deathsdataset['winsorized_Alcohol'] =
winsorized_Alcoholdataset['winsorized_Percentage_Exp'] =
winsorized_Percentage_Expdataset['winsorized_HepatitisB'] =
winsorized_HepatitisBdataset['winsorized_Under_Five_Deaths'] =
winsorized_Under_Five_Deathsdataset['winsorized_Polio'] =
winsorized_Poliodataset['winsorized_Tot_Exp'] =
winsorized_Tot_Expdataset['winsorized_Diphtheria'] =
winsorized_Diphtheriadataset['winsorized_HIV'] =
```

```
winsorized_HIVdataset['winsorized_GDP'] =
winsorized_GDPdataset['winsorized_Population'] =
winsorized_Populationdataset['winsorized_thinness_1to19_years'] =
winsorized_thinness_1to19_yearsdataset['winsorized_thinness_5to9_years'] =
winsorized_thinness_5to9_yearsdataset['winsorized_Income_Comp_Of_Resources'] =
winsorized_Income_Comp_Of_Resourcesdataset['winsorized_Schooling'] =
winsorized_Schooling
```

# Dividing the dataset into features and target

In [43]:

```
X = dataset[['Year', 'Status','winsorized_Adult_Mortality',       'BMI',
'winsorized_Infant_Deaths', 'winsorized_Alcohol', 'winsorized_Percentage_Exp',
'winsorized_HepatitisB',       'winsorized_Under_Five_Deaths',
'winsorized_Polio', 'winsorized_Tot_Exp', 'winsorized_Diphtheria',
'winsorized_HIV', 'winsorized_GDP', 'winsorized_Population',
'winsorized_thinness_1to19_years',       'winsorized_thinness_5to9_years',
'winsorized_Income_Comp_Of_Resources', 'winsorized_Schooling']]y =
dataset[['winsorized_Life_Expectancy']]
```

# Train Test Split

We will train out model on the training set and then use the test set to evaluate the model.

In [44]:

```
from sklearn.model_selection import train_test_splitX_train, X_test, y_train,
y_test=train_test_split(X,y,train_size=0.8, test_size=0.2, random_state=1)
```

# Dealing with Categorical variables using Label Encoding

In [45]:

```
##LABEL ENCODINGprint("\n\nUnique values in Status column in training data:",
X_train['Status'].unique())print("\nUnique values in Status column in validation
data:", X_test['Status'].unique())# All categorical columnsobject_cols = [col for
col in X_train.columns if X_train[col].dtype == "object"]# Columns that can be
safely label encodedgood_label_cols = [col for col in object_cols if
set(X_train[col]) == set(X_test[col])]        # Problematic columns that will be
dropped from the datasetbad_label_cols = list(set(object_cols)-
set(good_label_cols))        print('Categorical columns that will be label
encoded:', good_label_cols)print('\nCategorical columns that will be dropped from
the dataset:', bad_label_cols)
```

```
Unique values in Status column in training data: ['Developing' 'Developed']Unique
values in Status column in validation data: ['Developing' 'Developed']Categorical
columns that will be label encoded: ['Status']Categorical columns that will be
dropped from the dataset: []
```

In [46]:

```
from sklearn.preprocessing import LabelEncoder# Drop categorical columns that
will not be encodedlabel_X_train = X_train.drop(bad_label_cols,
axis=1)label_X_test = X_test.drop(bad_label_cols, axis=1)# Apply label encoder
label_encoder=LabelEncoder()for col in set(good_label_cols):
label_X_train[col] = label_encoder.fit_transform(X_train[col])
label_X_test[col] = label_encoder.transform(X_test[col])
X_train=label_X_trainX_test=label_X_test
```

# Creating and Training the Model

In [47]:

```
from sklearn.linear_model import LinearRegression
```

In [48]:

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

Out[48]:

```
LinearRegression(copy_X=True, fit_intercept=True, n_jobs=None,
normalize=False)
```

# Model Evaluation

In [49]:

```
preds = lm.predict(X_test) # print the interceptprint(lm.intercept_)
```

```
[-59.6279908]
```

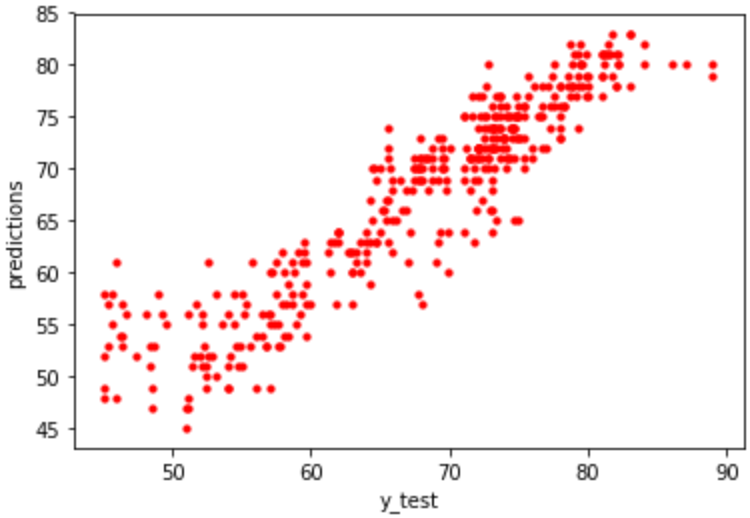
# Predictions

In [50]:

```
predictions = np.round(lm.predict(X_test))
```

In [51]:

```
#plt.scatter(x, y, s=10) plt.scatter(y_test, predictions, s=10, color='r')
#plt.scatter(Xvalid, yvalid, s=10)
plt.xlabel('y_test')plt.ylabel('predictions')        # predicted values
#plt.plot(X_valid, y_predicted, color='r')plt.show()
```




In [52]:

```
print('R_square score on the training: %.2f' % lm.score(X_train, y_train))
```

```
R_square score on the training: 0.87
```

## Regression Evaluation Metrics

In [53]:

```
from sklearn import metrics
```

In [54]:

```
print('MAE:', round(metrics.mean_absolute_error(y_test,
predictions),2))print('MSE:', round(metrics.mean_squared_error(y_test,
predictions),2))print('RMSE:', round(np.sqrt(metrics.mean_squared_error(y_test,
predictions)),2))print("R2 Score (Accuracy):", round(metrics.r2_score(y_test,
predictions)*100,2,), "%")
```

```
MAE: 2.76MSE: 13.91RMSE: 3.73R2 Score (Accuracy): 86.04 %
```

## Deploying the model

In [55]:

```
!pip install watson-machine-learning-client
```

```
Requirement already satisfied: watson-machine-learning-client in
/opt/conda/envs/Python36/lib/python3.6/site-packages (1.0.376)Requirement already
satisfied: ibm-cos-sdk in /opt/conda/envs/Python36/lib/python3.6/site-packages
(from watson-machine-learning-client) (2.4.3)Requirement already satisfied:
pandas in /opt/conda/envs/Python36/lib/python3.6/site-packages (from watson-
machine-learning-client) (0.24.1)Requirement already satisfied: certifi in
/opt/conda/envs/Python36/lib/python3.6/site-packages (from watson-machine-
learning-client) (2020.4.5.1)Requirement already satisfied: requests in
```

```
/opt/conda/envs/Python36/lib/python3.6/site-packages (from watson-machine-
learning-client) (2.21.0)Requirement already satisfied: tabulate in
/opt/conda/envs/Python36/lib/python3.6/site-packages (from watson-machine-
learning-client) (0.8.2)Requirement already satisfied: tqdm in
/opt/conda/envs/Python36/lib/python3.6/site-packages (from watson-machine-
learning-client) (4.31.1)Requirement already satisfied: urllib3 in
/opt/conda/envs/Python36/lib/python3.6/site-packages (from watson-machine-
learning-client) (1.24.1)Requirement already satisfied: lomond in
/opt/conda/envs/Python36/lib/python3.6/site-packages (from watson-machine-
learning-client) (0.3.3)Requirement already satisfied: ibm-cos-sdk-
core==2.*,>=2.0.0 in /opt/conda/envs/Python36/lib/python3.6/site-packages (from
ibm-cos-sdk->watson-machine-learning-client) (2.4.3)Requirement already
satisfied: ibm-cos-sdk-s3transfer==2.*,>=2.0.0 in
/opt/conda/envs/Python36/lib/python3.6/site-packages (from ibm-cos-sdk->watson-
machine-learning-client) (2.4.3)Requirement already satisfied: numpy>=1.12.0 in
/opt/conda/envs/Python36/lib/python3.6/site-packages (from pandas->watson-
machine-learning-client) (1.15.4)Requirement already satisfied: python-
dateutil>=2.5.0 in /opt/conda/envs/Python36/lib/python3.6/site-packages (from
pandas->watson-machine-learning-client) (2.7.5)Requirement already satisfied:
pytz>=2011k in /opt/conda/envs/Python36/lib/python3.6/site-packages (from pandas-
>watson-machine-learning-client) (2018.9)Requirement already satisfied:
chardet<3.1.0,>=3.0.2 in /opt/conda/envs/Python36/lib/python3.6/site-packages
(from requests->watson-machine-learning-client) (3.0.4)Requirement already
satisfied: idna<2.9,>=2.5 in /opt/conda/envs/Python36/lib/python3.6/site-packages
(from requests->watson-machine-learning-client) (2.8)Requirement already
satisfied: six>=1.10.0 in /opt/conda/envs/Python36/lib/python3.6/site-packages
(from lomond->watson-machine-learning-client) (1.12.0)Requirement already
satisfied: docutils>=0.10 in /opt/conda/envs/Python36/lib/python3.6/site-packages
(from ibm-cos-sdk-core==2.*,>=2.0.0->ibm-cos-sdk->watson-machine-learning-client)
(0.14)Requirement already satisfied: jmespath<1.0.0,>=0.7.1 in
/opt/conda/envs/Python36/lib/python3.6/site-packages (from ibm-cos-sdk-
core==2.*,>=2.0.0->ibm-cos-sdk->watson-machine-learning-client) (0.9.3)
```

In [56]:

```
from watson_machine_learning_client import WatsonMachineLearningAPIClient
```

```
2020-06-08 14:06:17,154 - watson_machine_learning_client.metanames - WARNING -
'AUTHOR_EMAIL' meta prop is deprecated. It will be ignored.
```

In [58]:

```
client = WatsonMachineLearningAPIClient( wml_credentials )
```

In [59]:

```
model_props = {client.repository.ModelMetaNames.AUTHOR_NAME: "Ibra",
client.repository.ModelMetaNames.AUTHOR_EMAIL: "ibranafis@gmail.com",
client.repository.ModelMetaNames.NAME: "LifeExpectancy"}
```

In [60]:

```
model_artifact =client.repository.store_model(lm, meta_props=model_props)
```

In [61]:

```
published_model_uid = client.repository.get_model_uid(model_artifact)
```

In [62]:

```
published_model_uid
```

Out[62]:

```
'd49eaa0d-8718-4b39-9d48-0fedb4c05942'
```

In [63]:

```
deployment = client.deployments.create(published_model_uid,
name="LifeExpectancy")
```

```
#######################################################################################
######Synchronous deployment creation for uid: 'd49eaa0d-8718-4b39-9d48-
0fedb4c05942'
started#######################################################################################
#############INITIALIZINGDEPLOY_SUCCESS----------------------------------------
-----------------------------------------------------Successfully finished
deployment creation, deployment_uid='c4e12022-0500-4886-9a43-7af2b6f2efd9'-------
-------------------------------------------------------------------------------
--------
```

In [ ]:

```
scoring_endpoint = client.deployments.get_scoring_url(deployment)
```

In [ ]:

```
scoring_endpoint
```

In [ ]:

## flow.json:

```
[
  {
    "id": "a2508b5e.6863c8",
```

```
    "type": "tab",
    "label": "Flow 1",
    "disabled": false,
    "info": ""
  },
  {
    "id": "9e67a9fd.f58988",
    "type": "function",
    "z": "a2508b5e.6863c8",
    "name": "pre token",
    "func": "//make user given values as global
variables\nglobal.set(\"a\",msg.payload.a);\nglobal.set(\"b\",msg.payload.b);\nglobal.set(\"c\",
msg.payload.c);\nglobal.set(\"d\",msg.payload.d);\nglobal.set(\"e\",msg.payload.e);\nglobal.se
t(\"f\",msg.payload.f);\nglobal.set(\"g\",msg.payload.g);\nglobal.set(\"h\",msg.payload.h);\ngl
obal.set(\"i\",msg.payload.i);\nglobal.set(\"j\",msg.payload.j);\nglobal.set(\"k\",msg.payload.k)
;\nglobal.set(\"l\",msg.payload.l);\nglobal.set(\"m\",msg.payload.m);\nglobal.set(\"n\",msg.pa
yload.n);\nglobal.set(\"o\",msg.payload.o);\nglobal.set(\"p\",msg.payload.p);\nglobal.set(\"q\"
,msg.payload.q);\nglobal.set(\"r\",msg.payload.r);\nglobal.set(\"s\",msg.payload.s);\nglobal.set
(\"t\",msg.payload.t);\n\n//following are required to receive a token\nvar
apikey=\"f_REvsTCHubzSJoBr1DlLlyhKMyuTaHtB5XPNFanVqs3\";\nmsg.headers={\"content-
type\":\"application/x-www-form-
urlencoded\"};\nmsg.payload={\"grant_type\":\"urn:ibm:params:oauth:grant-
type:apikey\",\"apikey\":apikey};\nreturn msg;\n",
    "outputs": 1,
    "noerr": 0,
    "x": 220,
    "y": 100,
    "wires": [
      [
        "46e81f82.1627a"
      ]
```

```
    ]
  },
  {
    "id": "731ff45f.9a132c",
    "type": "http request",
    "z": "a2508b5e.6863c8",
    "name": "",
    "method": "POST",
    "ret": "obj",
    "paytoqs": false,
    "url": "https://us-south.ml.cloud.ibm.com/v3/wml_instances/2982a739-5cdd-41a5-8598-0df7b94edf07/deployments/c4e12022-0500-4886-9a43-7af2b6f2efd9/online",
    "tls": "",
    "persist": false,
    "proxy": "",
    "authType": "",
    "x": 470,
    "y": 180,
    "wires": [
      [
        "c9a872bb.ddc21",
        "729d39cf.55c948"
      ]
    ]
  },
  {
    "id": "e6e6b417.565768",
```

```
    "type": "debug",
    "z": "a2508b5e.6863c8",
    "name": "",
    "active": true,
    "tosidebar": true,
    "console": false,
    "tostatus": false,
    "complete": "payload",
    "targetType": "msg",
    "x": 750,
    "y": 280,
    "wires": []
},
{
    "id": "729d39cf.55c948",
    "type": "function",
    "z": "a2508b5e.6863c8",
    "name": "getFrom Endpoint",
    "func": "msg.payload=msg.payload.values[0][0];\nreturn msg;",
    "outputs": 1,
    "noerr": 0,
    "x": 490,
    "y": 320,
    "wires": [
        [
            "e6e6b417.565768",
            "a6d7775f.fcb3c8"
```

```
        ]
      ]
    },
    {
      "id": "c9a872bb.ddc21",
      "type": "debug",
      "z": "a2508b5e.6863c8",
      "name": "",
      "active": true,
      "tosidebar": true,
      "console": false,
      "tostatus": false,
      "complete": "payload",
      "targetType": "msg",
      "x": 710,
      "y": 180,
      "wires": []
    },
    {
      "id": "4feea9ea.678fb8",
      "type": "function",
      "z": "a2508b5e.6863c8",
      "name": "sendTo Endpoint",
      "func": "//get token and make headers\nvar token=msg.payload.access_token;\nvar instance_id=\"2982a739-5cdd-41a5-8598-0df7b94edf07\"\nmsg.headers={'Content-Type': 'application/json',\"Authorization\":\"Bearer \"+token,\"ML-Instance-ID\":instance_id}\n\n//get variables that are set earlier\nvar a = global.get(\"a\");\nvar b = global.get(\"b\");\nvar c = global.get(\"c\");\nvar d = global.get(\"d\");\nvar e = global.get(\"e\");\nvar f = global.get(\"f\");\nvar g = global.get(\"g\");\nvar h =
```

global.get(\"h\");\nvar i = global.get(\"i\");\nvar j = global.get(\"j\");\nvar k = global.get(\"k\");\nvar l = global.get(\"l\");\nvar m = global.get(\"m\");\nvar n = global.get(\"n\");\nvar o = global.get(\"o\");\nvar p = global.get(\"p\");\nvar q = global.get(\"q\");\nvar r = global.get(\"r\");\nvar s = global.get(\"s\");\nvar t = global.get(\"t\");\nvar u = global.get(\"u\");\n\n//send the user values to service endpoint\nmsg.payload = \n{\"fields\":['Year', 'Status','winsorized_Adult_Mortality',\n 'BMI', 'winsorized_Infant_Deaths', 'winsorized_Alcohol', 'winsorized_Percentage_Exp', 'winsorized_HepatitisB',\n 'winsorized_Under_Five_Deaths', 'winsorized_Polio', 'winsorized_Tot_Exp', 'winsorized_Diphtheria',\n 'winsorized_HIV', 'winsorized_GDP', 'winsorized_Population', 'winsorized_thinness_1to19_years',\n 'winsorized_thinness_5to9_years', 'winsorized_Income_Comp_Of_Resources', 'winsorized_Schooling'],\n\"values\":[[a,b,c,d,e,f,g,h,i,j,k,l,m,n,o,p,q,r,s]]};\n\nreturn msg;\n",

    "outputs": 1,

    "noerr": 0,

    "x": 210,

    "y": 180,

    "wires": [

      [

        "731ff45f.9a132c"

      ]

    ]

  },

  {

    "id": "46e81f82.1627a",

    "type": "http request",

    "z": "a2508b5e.6863c8",

    "name": "",

    "method": "POST",

    "ret": "obj",

    "paytoqs": false,

    "url": "https://iam.cloud.ibm.com/identity/token",

```
    "tls": "",
    "persist": false,
    "proxy": "",
    "authType": "basic",
    "x": 370,
    "y": 100,
    "wires": [
      [
        "4feea9ea.678fb8"
      ]
    ]
  },
  {
    "id": "a6d7775f.fcb3c8",
    "type": "ui_text",
    "z": "a2508b5e.6863c8",
    "group": "fdbd9753.5a2e18",
    "order": 1,
    "width": 0,
    "height": 0,
    "name": "",
    "label": "Life Expectancy Prediction:",
    "format": "{{msg.payload}}",
    "layout": "row-spread",
    "x": 780,
    "y": 400,
    "wires": []
```

```
},
{
    "id": "c2feea5f.17cd68",
    "type": "ui_form",
    "z": "a2508b5e.6863c8",
    "name": "",
    "label": "",
    "group": "fdbd9753.5a2e18",
    "order": 3,
    "width": 0,
    "height": 0,
    "options": [
        {
            "label": "Year",
            "value": "a",
            "type": "number",
            "required": true,
            "rows": null
        },
        {
            "label": "Status (Developing: 0, Developed: 1)",
            "value": "b",
            "type": "number",
            "required": true,
            "rows": null
        },
        {
```

```
    "label": "Adult Mortality",
    "value": "c",
    "type": "number",
    "required": true,
    "rows": null
},
{
    "label": "BMI",
    "value": "d",
    "type": "number",
    "required": true,
    "rows": null
},
{
    "label": "Infant Deaths",
    "value": "e",
    "type": "number",
    "required": true,
    "rows": null
},
{
    "label": "Alcohol",
    "value": "f",
    "type": "number",
    "required": true,
    "rows": null
},
```

```
{
    "label": "Percentage Expenditure",
    "value": "g",
    "type": "number",
    "required": true,
    "rows": null
},
{
    "label": "Hepatitis B",
    "value": "h",
    "type": "number",
    "required": true,
    "rows": null
},
{
    "label": "Under 5 deaths",
    "value": "i",
    "type": "number",
    "required": true,
    "rows": null
},
{
    "label": "Polio",
    "value": "j",
    "type": "number",
    "required": true,
    "rows": null
```

```
},
{
    "label": "Total Expenditure",
    "value": "k",
    "type": "number",
    "required": true,
    "rows": null
},
{
    "label": "Diptheria",
    "value": "l",
    "type": "number",
    "required": true,
    "rows": null
},
{
    "label": "HIV",
    "value": "m",
    "type": "number",
    "required": true,
    "rows": null
},
{
    "label": "GDP",
    "value": "n",
    "type": "number",
    "required": true,
```

```
    "rows": null
},
{
    "label": "Population",
    "value": "o",
    "type": "number",
    "required": true,
    "rows": null
},
{
    "label": "Thinness 1 to 19 years",
    "value": "p",
    "type": "number",
    "required": true,
    "rows": null
},
{
    "label": "Thinness 5 to 9 years",
    "value": "q",
    "type": "number",
    "required": true,
    "rows": null
},
{
    "label": "Income",
    "value": "r",
    "type": "number",
```

```
      "required": true,
      "rows": null
    },
    {
      "label": "Schooling\t",
      "value": "s",
      "type": "number",
      "required": true,
      "rows": null
    }
  ],
  "formValue": {
    "a": "",
    "b": "",
    "c": "",
    "d": "",
    "e": "",
    "f": "",
    "g": "",
    "h": "",
    "i": "",
    "j": "",
    "k": "",
    "l": "",
    "m": "",
    "n": "",
    "o": "",
```

```
      "p": "",
      "q": "",
      "r": "",
      "s": ""
    },
    "payload": "",
    "submit": "submit",
    "cancel": "cancel",
    "topic": "",
    "x": 70,
    "y": 100,
    "wires": [
      [
        "9e67a9fd.f58988"
      ]
    ]
  },
  {
    "id": "fdbd9753.5a2e18",
    "type": "ui_group",
    "z": "",
    "name": "Predict your Life Expectancy",
    "tab": "98049835.130968",
    "order": 1,
    "disp": true,
    "width": 8,
    "collapse": false
```

```
},
{
    "id": "98049835.130968",
    "type": "ui_tab",
    "z": "",
    "name": "Home",
    "icon": "dashboard",
    "disabled": false,
    "hidden": false
}
        ]
```