

**Project Report**

*on*

**Predicting Life Expectancy  
using**

**Machine Learning**

*by*

**Joshni Princia Saldanha**

**(princias987@gmail.com)**

# **Predicting Life Expectancy Using Machine Learning**

<b>1. Introduction.....</b>	<b>4</b>
1.1 Overview .....	4
1.2 Purpose.....	5
<b>2. Literature Survey .....</b>	<b>6</b>
2.1 Existing Problem.....	6
2.2 Proposed Solution.....	6
<b>3. Theoretical Analysis .....</b>	<b>7</b>
3.1 Block Diagram.....	7
3.2 Hardware / Software Designing.....	8
<b>4. Experimental Investigations.....</b>	<b>9</b>
<b>5. Flowchart .....</b>	<b>16</b>
<b>6. Result .....</b>	<b>18</b>
<b>7. Advantages &amp; Disadvantages.....</b>	<b>20</b>
<b>8. Applications.....</b>	<b>21</b>
<b>9. Conclusion .....</b>	<b>22</b>
<b>10. Future Scope.....</b>	<b>23</b>
<b>11. Bibliography .....</b>	<b>24</b>
<b>Appendix .....</b>	<b>25</b>
A. Source Code.....	25

# 1. Introduction

Since ancient times, there are a lot of change in the behaviours and cultures of people in different places. According to their way of living, the health care and life expectancy of people varies among each other. These differences are may be based on various factors such as Regional variations, Economic Circumstances, Sex Differences, Mental Illnesses, Physical Illnesses, Education, Year of their birth and other demographic factors.

## 1.1. Overview

Life expectancy is a statistical measure of the average time a human being is expected to live. A typical Regression Machine Learning project leverages historical data to predict insights into the future. This problem statement is aimed at predicting Life Expectancy rate of a country given various features. This problem statement provides a way to predict average life expectancy of people living in a country when various factors such as year, GDP, education, alcohol intake of people in the country, expenditure on healthcare system and some specific disease related deaths that happened in the country are given in a dataset.

In order to predict life expectancy rate of a given country, we will be using Machine Learning algorithms to draw inferences from the given dataset and give an output. For better usability by the customer, we are also going to be creating a UI for the user to interact with using Node-Red.

## 1.2. Purpose

The purpose of this project is that the people from various places can easily predict their life expectancy by providing the inputs asked by the model. This software can be used by all people in the world because the training part of this model contains inputs and predictions of more number of countries.

### Economic growth:

Predicting life expectancy would play a vital role in judging the growth and development of the economy.

Across countries, high life expectancy is associated with high income per capita. Increase in life expectancy also leads to an increase in the “manpower” of a country. The knowledge asset of a country increases with the number of individuals in a country.

### Population Growth:

Helps the government bodies take appropriate measures to control the population growth and also direct the utilization of the increase in human resources and skillset acquired by people over many years.

### Personal growth:

This project would also help an individual assess his/her lifestyle choices and alter them accordingly to lead a longer and healthier life. It would make them more aware of their general health and its improvement or deterioration over time.

### Growth in Health Sector:

Based on the factors used to calculate life expectancy of an individual and the outcome, health care will be able to fund and provide better services to those with greater need.

### Insurance Companies:

Insurance sector will be able to provide individualized services to people based on the life expectancy outcomes and factors.

## **2. Literature Survey**

There are so many organizations that are making research in the prediction of life expectancy. Many research papers dealing with the creation of this model under many algorithms such as Machine Learning, Deep learning and programming languages such as Python and Java script.

### **2.1. Existing Problem**

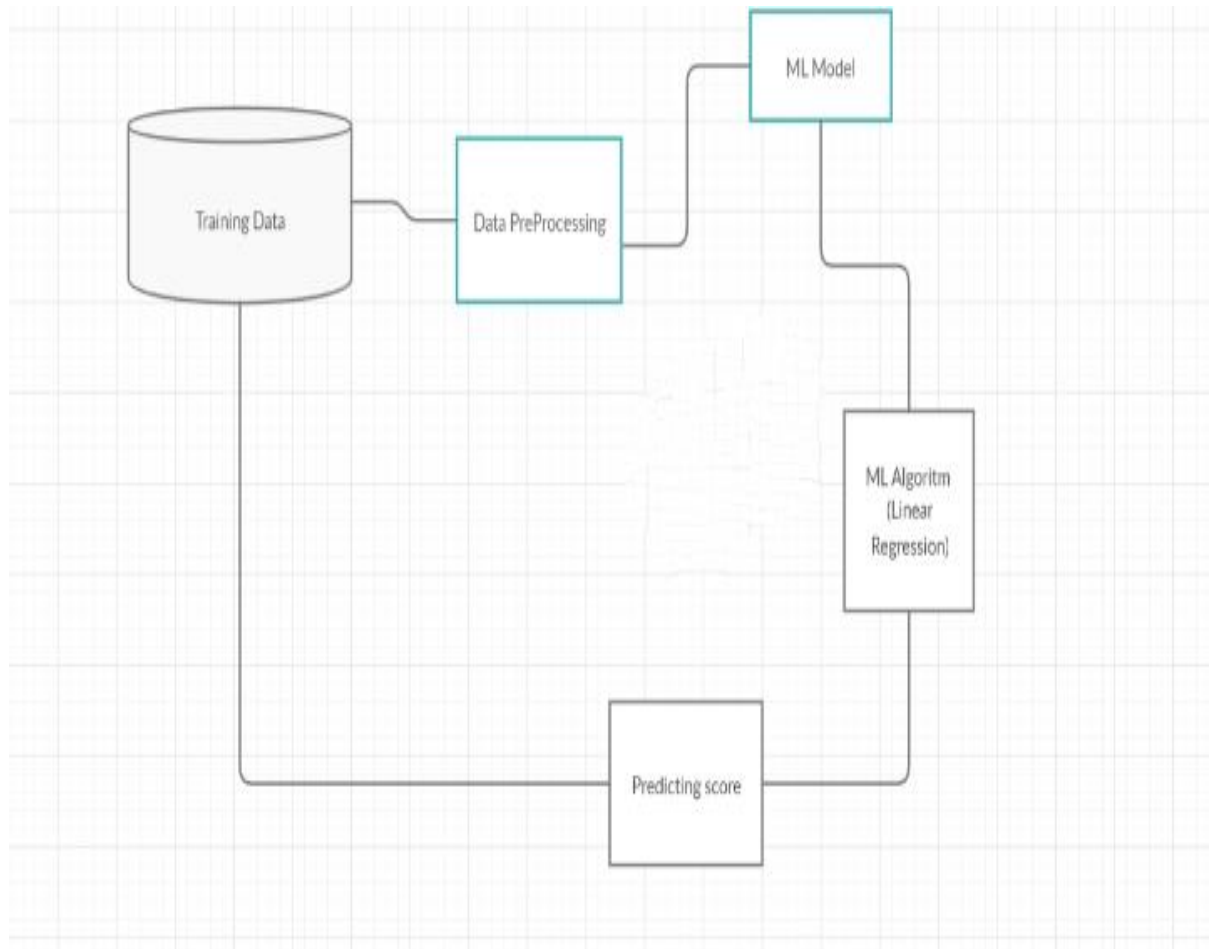
The World Health Organization (WHO) began producing annual life tables for all Member States in 1999. These life tables are a basic input to all WHO estimates of global, regional and country-level patterns and trends in all-cause and cause-specific mortality. After the publication of life tables for years to 2009 in the 2011 edition of World Health Statistics, WHO has shifted to a two year cycle for the updating of life tables for all Member States. Even still the model is not really updated in every fields. WHO applies standard methods to the analysis of Member State data to ensure comparability of estimates across countries. This will inevitably result in differences for some Member States with official estimates for quantities such as life expectancy, where a variety of different projection methods and other methods are used.

### **2.2. Proposed Solution**

So many people were expecting to use a model of life expectancy prediction. In order to that, many institutions and companies are leading their team to build that model. In my project, I have proposed a solution to predict the life expectancy using machine learning. Machine Learning is the process of training the computer to think and decide solutions like human. The reason why I have chosen this architecture was only with the help of Machine Learning, deep understanding of the data and an ability to create a model can be done. Design a Regression model to predict life expectancy ratio of a given country based on some features provided such as year, GDP (gross domestic product), education, alcohol intake of people in the country, expenditure on healthcare system and some specific disease related deaths that happened in the country.

## 3. Theoretical Analysis

### 3.1. Block Diagram



## 3.2. Hardware / Software Designing

1. PROJECT PLANNING AND KICKOFF:
  - a. Understanding the project description and analyze the data and attributes in the given dataset.
  - b. Creating Github account
  - c. Installing Slack and create account with the mail id
  - d. Learning to use Zoho writer.
2. EXPLORE IBM CLOUD PLATFORM:
  - a. Creating IBM cloud account with the mail id
  - b. Creating IBM academic initiative account with the mail id
  - c. Create a Node-Red starter application.
3. EXPLORE IBM WATSON SERVICES:
  - a. Exploring IBM Watson use cases.
  - b. Learning about IBM Watson Machine Learning.
4. INTRODUCTION TO WATSON STUDIO:
  - a. Learning to build own Machine Learning model using IBM Watson.
  - b. Automate the Machine Learning Model
5. PREDICTING LIFE EXPECTANCY WITH PYTHON:
  - a. Collecting Data set from [www.kaggle.com](https://www.kaggle.com)
  - b. Creating IBM Watson services
  - c. Create a jupyter notebook and import data from Object storage.
6. PREDICTING LIFE EXPECTANCY WITHOUT PYTHON:
  - a. Created Node-Red model and integrated with Machine Learning model.



## 4. Experimental Investigation

### Life Expectancy Dataset:

The dataset used is a life expectancy dataset released by the World Health Organization.

The data set has the following features:

The data is saved as a csv file as LifeExpectancy.csv and it is read and stored in the life data variable. The Year column is dropped as it will not be used in the analysis. The first 5 rows are shown below. The data contains 21 columns and 2938 rows with the header row. The table contains data about:

- Countries
- Status
- Life Expectancy
- Adult Mortality
- 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

### Preprocessing and cleaning the datasets:

- Before the data can be imported using the machine learning libraries and can be trained, the data needs to be cleaned and pre-processed.
- All the null values in the data set need to be either set to 0, deleted or set equal to the mean value.
- In the cleaning process, I have set the null values as 0 for the ease of calculation and maintaining the accuracy of the model.

### Importing the required libraries:

```
In [4]: import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
%matplotlib inline
```

### Loading the packages:

The following packages have been imported NumPy, Pandas, Matplotlib, SciPy, and Seaborn. Sklearn is the most widely used package for the machine learning process. The following subpackages have been used:

1. train\_test\_split
2. linear\_model
3. model selection
4. metrics
5. tree
6. ensemble
7. pre-processing

### Importing the dataset:

The required dataset in the csv file is imported as the panda data frame.

```
life.head()
```

	year	status	life	adultmortality	infantdeaths	alcohol	percentageexpenditure	hepatitisb	measles	bmi	...	polio	total
0	2015	0	65.0	263	62	0.01	71.279624	65	1154	19.1	...	6	8.16
1	2014	0	59.9	271	64	0.01	73.523582	62	492	18.6	...	58	8.16
2	2013	0	59.9	268	66	0.01	73.219243	64	430	18.1	...	62	8.13
3	2012	0	59.5	272	69	0.01	78.184215	67	2787	17.6	...	67	8.52
4	2011	0	59.2	275	71	0.01	7.097109	68	3013	17.2	...	68	7.87

5 rows x 14 columns

## Displaying the structure of the dataset:

```
In [7]: life.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 2938 entries, 0 to 2937
Data columns (total 21 columns):
year                2938 non-null int64
status              2938 non-null int64
life                2938 non-null float64
adultmortality       2938 non-null int64
infantdeaths         2938 non-null int64
alcohol              2938 non-null float64
percentageexpenditure 2938 non-null float64
hepatitisb           2938 non-null int64
measles              2938 non-null int64
bmi                  2938 non-null float64
underfivedeaths      2938 non-null int64
polio                2938 non-null int64
totalexpenditure     2938 non-null float64
dip                  2938 non-null int64
hiv/aids              2938 non-null float64
gdp                  2938 non-null float64
population            2938 non-null float64
thinness119           2938 non-null float64
thinness59           2938 non-null float64
incomecomp           2938 non-null float64
schooling             2938 non-null float64
dtypes: float64(12), int64(9)
memory usage: 482.1 KB
```

## Displaying the columns of the dataset:

```
In [9]: life.columns
```

```
Out[9]: Index(['year', 'status', 'life', 'adultmortality', 'infantdeaths', 'alcohol',
               'percentageexpenditure', 'hepatitisb', 'measles', 'bmi',
               'underfivedeaths', 'polio', 'totalexpenditure', 'dip', 'hiv/aids',
               'gdp', 'population', 'thinness119', 'thinness59', 'incomecomp',
               'schooling'],
              dtype='object')
```

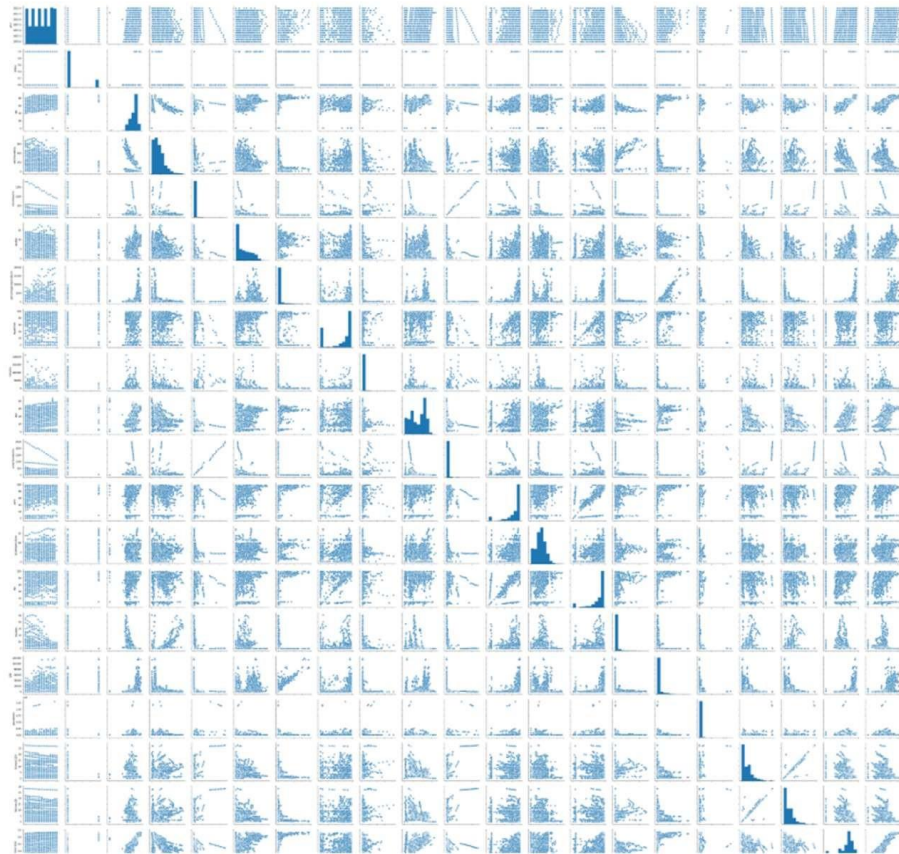
## EDA:

### EDA

Let's create some simple plots to check out the data!

```
In [10]: sns.pairplot(life)
```

```
Out[10]: <seaborn.axisgrid.PairGrid at 0x7f7b7813b588>
```

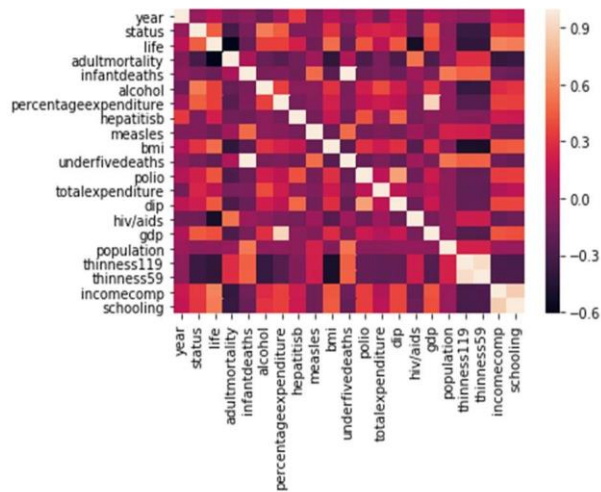


Now we will plot the correlation matrix visualizing it with a heatmap. The legend tells that the warmer colors show higher and positive correlation, while the colder low or negative.

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.

```
In [12]: sns.heatmap(life.corr())
```

```
Out[12]: <matplotlib.axes._subplots.AxesSubplot at 0x7f7b67908ef0>
```



## Training the regression model:

### Training a Linear Regression Model

Let's now begin to train our regression model! We will need to first split up our data into an X array that contains the features to train on, and a y array with the target variable, in this case the Price column. We will toss out the Address column because it only has text info that the linear regression model can't use.

#### X and y arrays

```
In [13]: X = life[['year', 'status', 'adultmortality', 'infantdeaths', 'alcohol', 'percentageexpenditure', 'hepatitisb', 'measles', 'bmi', 'underfivedeaths', 'polio', 'totalexpenditure', 'dip', 'hiv/aids', 'gdp', 'population', 'thinness119', 'thinness59', 'incomecomp', 'schooling']]
y = life['life']
```

#### Train Test Split

Now let's split the data into a training set and a testing set. We will train our model on the training set and then use the test set to evaluate the model.

```
In [14]: from sklearn.model_selection import train_test_split
```

```
In [15]: X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.4, random_state=101)
```

#### Creating and Training the Model

```
In [16]: from sklearn.linear_model import LinearRegression
```

```
In [17]: lm = LinearRegression()
```

```
In [18]: lm.fit(X_train, y_train)
```

```
Out[18]: LinearRegression(copy_X=True, fit_intercept=True, n_jobs=None,
normalize=False)
```

#### Model Evaluation

Let's evaluate the model by checking out its coefficients and how we can interpret them.

```
In [19]: # print the intercept
print(lm.intercept_)

54.32620483709512
```

## Predictions from our model:

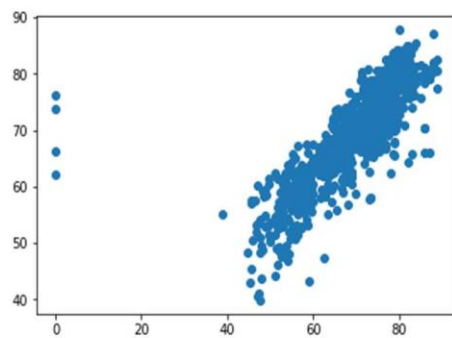
### Predictions from our Model

Let's grab predictions off our test set and see how well it did!

```
In [21]: predictions = lm.predict(X_test)
```

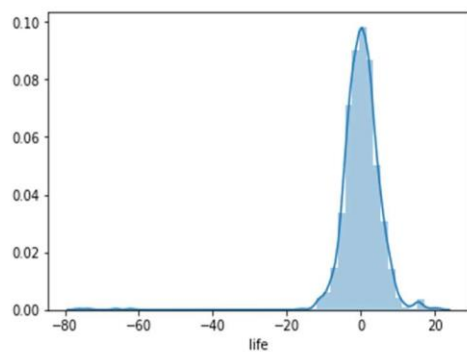
```
In [22]: plt.scatter(y_test, predictions)
```

```
Out[22]: <matplotlib.collections.PathCollection at 0x7f7b60110550>
```



### Residual Histogram

```
In [23]: sns.distplot((y_test-predictions),bins=50);
```



## Linear regression with polynomial functions:

### Regression Evaluation Metrics

Here are three common evaluation metrics for regression problems:

**Mean Absolute Error (MAE)** is the mean of the absolute value of the errors:

$$\frac{1}{n} \sum_{i=1}^n |y_i - \hat{y}_i|$$

**Mean Squared Error (MSE)** is the mean of the squared errors:

$$\frac{1}{n} \sum_{i=1}^n (y_i - \hat{y}_i)^2$$

**Root Mean Squared Error (RMSE)** is the square root of the mean of the squared errors:

$$\sqrt{\frac{1}{n} \sum_{i=1}^n (y_i - \hat{y}_i)^2}$$

Comparing these metrics:

- **MAE** is the easiest to understand, because it's the average error.
- **MSE** is more popular than MAE, because MSE "punishes" larger errors, which tends to be useful in the real world.
- **RMSE** is even more popular than MSE, because RMSE is interpretable in the "y" units.

All of these are **loss functions**, because we want to minimize them.

```
In [24]: from sklearn import metrics
```

```
In [26]: print('MAE:', metrics.mean_absolute_error(y_test, predictions))
print('MSE:', metrics.mean_squared_error(y_test, predictions))
print('RMSE:', np.sqrt(metrics.mean_squared_error(y_test, predictions)))
```

```
MAE: 3.5394253196842085
MSE: 35.63069322049429
RMSE: 5.969145099634812
```

Mean Absolute Error: 3.53942531968

Root Mean Square Error: 5.9691450996

After applying the regression algorithm:

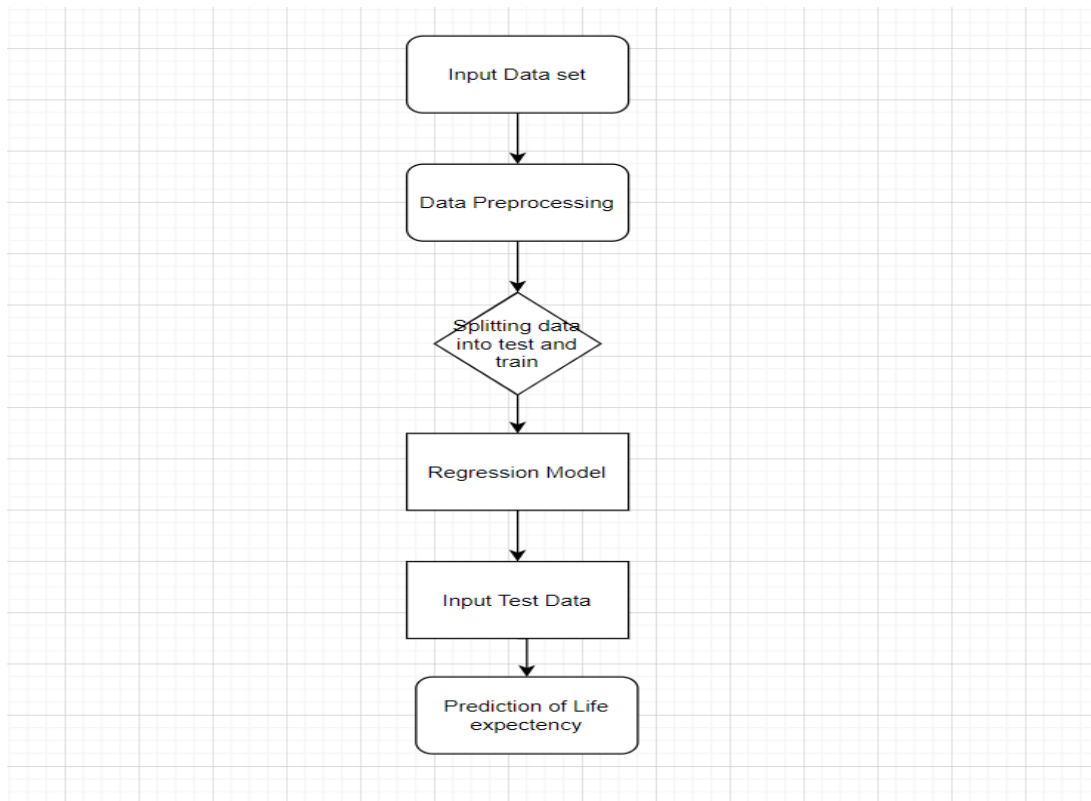
1. MAE of 1.83
2. RMSE of 6.05

Three models have been created. The Algorithms have been used to test if they can provide good prediction with fewer errors while predicting the life expectancy for new data. The Model Algorithms used are:

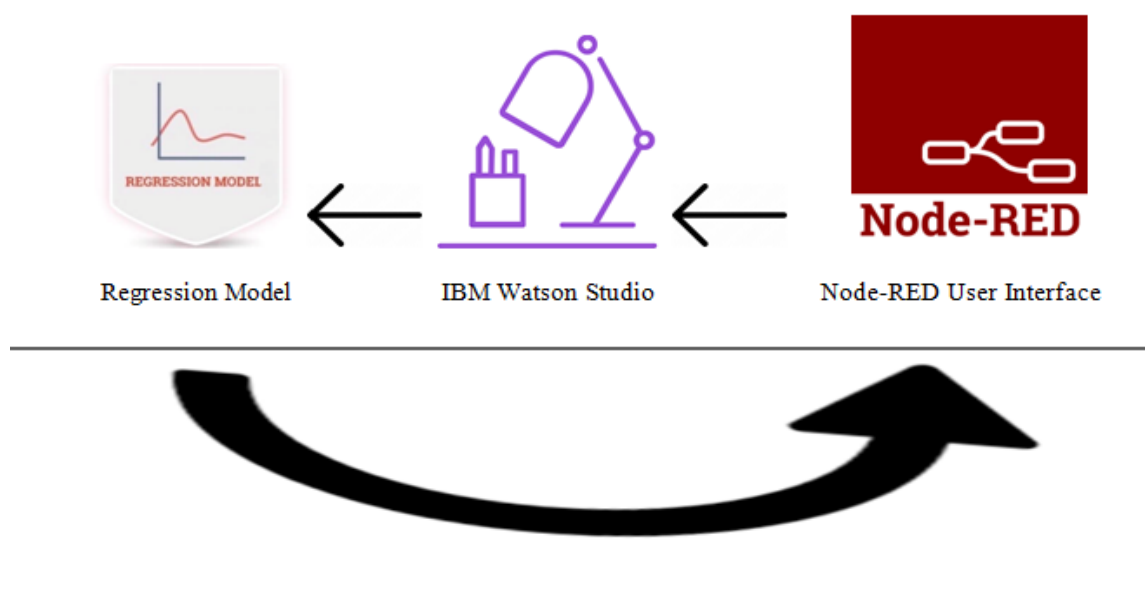
- Decision Tree Regression
- Linear Regression
- Random Forest Regression

On Comparing Both the Models, we came to this conclusion that Random Forest Model is giving us less error and best Prediction score in compare to Linear Regression Model and Decision Tree Regression.

## 5. Flowchart



### UI USING THE NODE RED





To integrate the ML model with the UI, we would be using the Node Red functionality provided by the IBM Watson Studio.

To design the UI, we need to import the flow of the UI.

Once, we have setup the flow, we need to integrate the ML model with it. To integrate the ML Model with it we need to access the endpoint URL of our ML Model.

Components of the flow are:

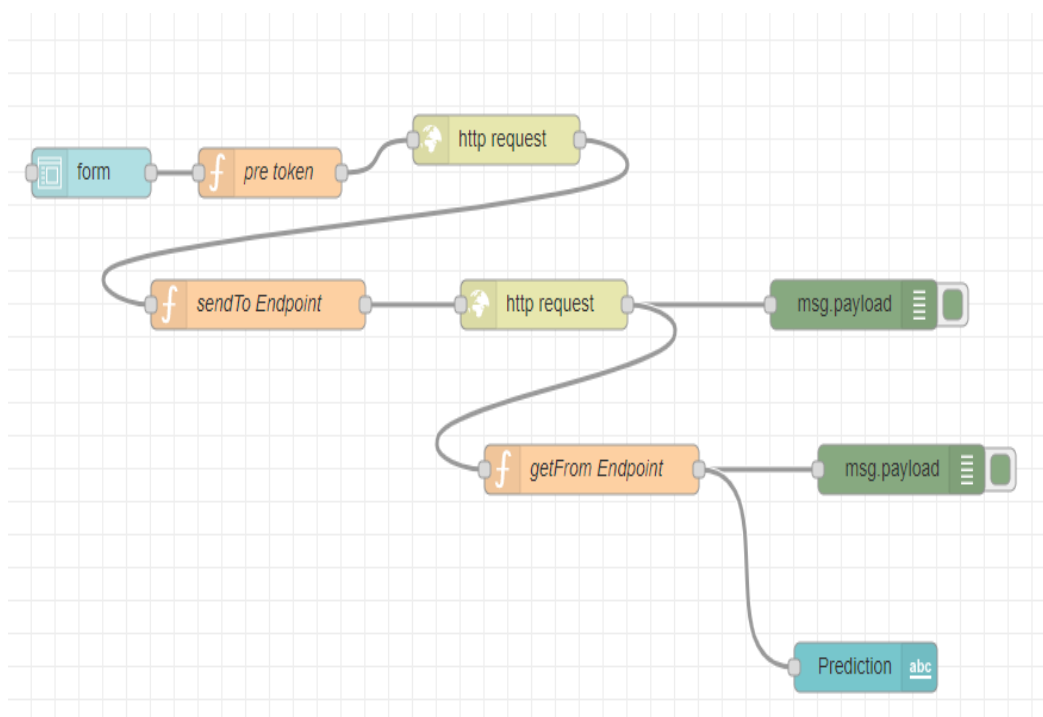
Form: The form contains all the elements of the UI. All the labels are associated with a variable.

Http requests: To setup the flow, we need two http requests.

The first http request requires a token to connect to the machine learning service of the Watson studio.

The second http request helps us in integrating the model using the endpoint URL.

Once the flow has been setup, we deploy the model.



## 6. Result

Web based UI was developed by integrating all the services using NODE-RED.

URL for UI Dashboard: <https://node-red-uyrym.eu-gb.mybluemix.net/ui>

URL for Notebook: [https://github.com/SmartPracticeschool/IISPS-INT-2027-Predicting-Life-Expectancy-using-Machine-Learning/blob/master/Life\\_Expectancy\\_Prediction\\_Notebook.ipynb](https://github.com/SmartPracticeschool/IISPS-INT-2027-Predicting-Life-Expectancy-using-Machine-Learning/blob/master/Life_Expectancy_Prediction_Notebook.ipynb)

While giving the inputs for the country Afghanistan in the year 2017, the life expectancy value 57.540 has been predicted.

**Life Expectancy Prediction**

Prediction **57.540000000000006**

Country \*  
Afghanistan

Year \*  
2017

Status \*  
Developing

BMI \*  
15.2

Adult\_Mortality \*  
295

Infant\_Deaths \*  
82

Alcohol \*  
0.02

Percentage\_Expenditure \*  
10.01015500

Alcohol \*

0.02

Percentage\_Expenditure \*

10.91015598

Hepatitis\_B \*

63

Under\_Five\_Deaths \*

113

Polio \*

63

Total\_Expenditure \*

6.73

Diphtheria \*

63

HIV/AIDS \*

0.1

GDP \*

369.835796

Population \*

26616792

Thinness\_10\_19\_years \*

19

Thinness\_5\_9\_years \*

19.1

Income\_Composition\_of\_Resources \*

0.415

Schooling \*

8.4

Measles \*

1141

PREDICT

CANCEL

## 7. Advantages & Disadvantages

### Advantages:

One of the biggest advantages of embedding machine learning algorithms is their ability to improve over time. Machine learning technology typically improves efficiency and accuracy thanks to the ever-increasing amounts of data that are processed.

The application learns the patterns and trends hidden within the data without human intervention which makes predicting much simpler and easier. The more data is fed to the algorithm, the higher the accuracy of the algorithm is. It is also the key component in technologies for automation.

Using Node-Red also simplifies the effort put into creating the front-end. The programmer doesn't need extensive knowledge on HTML and JavaScript. It also makes the integration between Machine learning model and the UI much easier.

### Disadvantages:

Using machine learning interface comes with its own problems. Since the whole point of it is minimize human involvement, it also makes error detection and fixing much more problematic. It takes a lot of time to identify the root cause for the problem.

Machine learning can also be very time-consuming. When the size of the data fed to the machine learning is very large, the computational cost and the time taken to train the model on the data increases drastically. This can increase the cost of resources required to implement the application on a large scale.

At the same time, Node-Red does not give many features to customize our UI.

## 8. Applications

- Personalized Life Expectancy: Individuals can predict their own life expectancy by inputting values in the corresponding fields. This could help make people more aware of their general health, and its improvement or deterioration over time. This may motivate them to make healthier lifestyle choices.
- Government: It could help the government bodies take appropriate measures to control the population growth and also direct the utilization of the increase in human resources and skillset acquired by people over many years. Across countries, high life expectancy is associated with high income per capita. Increase in life expectancy also leads to an increase in the “manpower” of a country. The knowledge asset of a country increases with the number of individuals in a country.
- Health Sector: Based on the factors used to calculate life expectancy of an individual and the outcome, health care will be able to fund and provide better services to those with greater need.
- Insurance Companies: Insurance sector will be able to provide individualized services to people based on the life expectancy outcomes and factors.

## 9. Conclusion

- The end product is a webpage created and deployed on node-red app of IBM cloud. The backend of webpage is a linear regression model created and deployed on Watson Studio using machine learning service.
- This model can be used to predict the life expectancy of people in different places.
- This model contains various factors such as 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.
- With the help of all these input values, the model will predict the life expectancy of such people.
- The accuracy level of prediction in my model is more than 95%.
- From the help of this model, the life expectancies of more than 190 countries can be detected.

## 10. Future Scope

For future use, one can integrate the life expectancy prediction with providing suggestions and medications to the individual using the application. This will help predict as well as increase the individual's life expectancy.

The scalability and flexibility of the application can also be improved with advancement in technology and availability of new and improved resources. Also, with the growth in Artificial Neural networks and Deep learning, one can integrate that with our existing application. With the help of Convolutional Neural networks and Computer vision, we can also try to take into account the physical health and appearance of a person.

Mental health can also be taken into account while predicting life expectancy with the help of sentiment analysis systems as well.

## 11. Bibliography

1. Node-RED Starter Application :  
<https://developer.ibm.com/tutorials/how-to-create-a-node-red-starter-application/>
2. Watson Studio Cloud :  
<https://bookdown.org/caoying4work/watsonstudio-workshop/jn.html>
3. Dataset Reference :  
<https://www.kaggle.com/kumarajarshi/life-expectancy-who>
4. IBM Cloud Services :  
<https://www.youtube.com/watch?v=DBRGIAHdj48&list=PLzpeuWUENMK2PYtasCaKK4bZjaYzhW23L>
5. Import the Dataset into Jupyter Notebook :  
<https://www.youtube.com/watch?v=Jtej3Y6uUng>



# Appendix

## A. Source Code

### Node-RED Flow code

```
[{"id":"719c1fee.e5b94","type":"tab","label":"Flow
2","disabled":false,"info":"","id":"e3e4b719.968578","type":"ui_form","z":"719c1fee.e5b94","name":"","label":"","group":"d01d05bb.7b14d8","order":2,"width":0,"height":0,"options":[{"label":"Country","value":"a","type":"text","required":true,"rows":null},{"label":"Year","value":"b","type":"number","required":true,"rows":null},{"label":"Status","value":"c","type":"text","required":true,"rows":null},{"label":"BMI","value":"d","type":"number","required":true,"rows":null},{"label":"Adult_Mortality","value":"e","type":"number","required":true,"rows":null},{"label":"Infant_Deaths","value":"f","type":"number","required":true,"rows":null},{"label":"Alcohol","value":"g","type":"number","required":true,"rows":null},{"label":"Percentage_Expenditure","value":"h","type":"number","required":true,"rows":null},{"label":"Hepatitis_B","value":"i","type":"number","required":true,"rows":null},{"label":"Under_Five_Deaths","value":"j","type":"number","required":true,"rows":null},{"label":"Polio","value":"k","type":"number","required":true,"rows":null},{"label":"Total_Expenditure","value":"l","type":"number","required":true,"rows":null},{"label":"Diphtheria","value":"m","type":"number","required":true,"rows":null},{"label":"HIV/AIDS","value":"n","type":"number","required":true,"rows":null},{"label":"GDP","value":"o","type":"number","required":true,"rows":null},{"label":"Population","value":"p","type":"number","required":true,"rows":null},{"label":"Thinness_10_19_years","value":"q","type":"number","required":true,"rows":null},{"label":"Thinness_5_9_years","value":"r","type":"number","required":true,"rows":null},{"label":"Income_Composition_of_Resources","value":"s","type":"number","required":true,"rows":null},{"label":"Schooling","value":"t","type":"number","required":true,"rows":null},{"label":"Measles","value":"u","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":"","t":"","u":""},"payload":"","submit":"Predict","cancel":"cancel","topic":"","x":70,"y":100,"wires":[["c8f4a328.e0202"]]},{"id":"c8f4a328.e0202","type":"function","z":"719c1fee.e5b94","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.set(\\\"f\\\",msg.payload.f);\\nglobal.set(\\\"g\\\",msg.payload.g);\\nglobal.set(\\\"h\\\",msg.payload.h);\\nglobal.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.payload.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.p
```

```

payload.t);\nglobal.set(\"u\",msg.payload.u);\n\n//following are required to receive a
token\n\nvar apikey=\"SfC-
TWpXlzz7_c1P74LM1fRYcD4IbCPedggiYCqYA5ss\";\n\nmsg.headers={\"content-
type\":\"application/x-www-form-
urlencoded\"};\n\nmsg.payload={\"grant_type\":\"urn:ibm:params:oauth:grant-
type:apikey\", \"apikey\":apikey};\n\nreturn
msg;\n\n,\"outputs\":1,\"noerr\":0,\"x\":220,\"y\":100,\"wires\":[[\"e164f901.a85778\"]]],{\"id\":\"
1db0486e.22e378\",\"type\":\"http
request\",\"z\":\"719c1fee.e5b94\",\"name\":\"\",\"method\":\"POST\",\"ret\":\"obj\",\"paytoqs\":fal
se,\"url\":\"https://eu-gb.ml.cloud.ibm.com/v3/wml_instances/b819851b-fdeb-48da-
8c0e-655136f2b699/deployments/cebf8c6d-ea54-4fbe-8167-
66b30e2bbaef/online\",\"tls\":\"\",\"persist\":false,\"proxy\":\"\",\"authType\":\"basic\",\"x\":450,\"
y\":180,\"wires\":[[\"ffd8333c.db4c5\",\"1be6a2f8.efed9d\"]]],{\"id\":\"7a53335e.b5123c\",\"ty
pe\":\"debug\",\"z\":\"719c1fee.e5b94\",\"name\":\"\",\"active\":true,\"tosidebar\":true,\"console
\":false,\"tostatus\":false,\"complete\":\"false\",\"x\":750,\"y\":280,\"wires\":[]},{\"id\":\"1be6a2f8
.efed9d\",\"type\":\"function\",\"z\":\"719c1fee.e5b94\",\"name\":\"getFrom
Endpoint\",\"func\":\"msg.payload=msg.payload.values[0][0];\n\nreturn
msg;\",\"outputs\":1,\"noerr\":0,\"x\":490,\"y\":280,\"wires\":[[\"7a53335e.b5123c\",\"3c53f487
.61d98c\"]]],{\"id\":\"ffd8333c.db4c5\",\"type\":\"debug\",\"z\":\"719c1fee.e5b94\",\"name\":\"\",
\"active\":true,\"tosidebar\":true,\"console\":false,\"tostatus\":false,\"complete\":\"payload\", \"t
argetType\":\"msg\", \"x\":710,\"y\":180,\"wires\":[]},{\"id\":\"9e7824f4.ef15f8\",\"type\":\"functio
n\", \"z\":\"719c1fee.e5b94\", \"name\":\"sendTo Endpoint\", \"func\":\"//get token and make
headers\n\nvar token=msg.payload.access_token;\n\nvar instance_id=\"b819851b-fdeb-
48da-8c0e-655136f2b699\"\n\nmsg.headers={\"Content-Type':
'application/json', \"Authorization\":\"Bearer \"+token, \"ML-Instance-
ID\":instance_id}\n\n\n//get variables that are set earlier\n\nvar a =
global.get(\"a\");\n\nvar b = global.get(\"b\");\n\nvar c = global.get(\"c\");\n\nvar d =
global.get(\"d\");\n\nvar e = global.get(\"e\");\n\nvar f = global.get(\"f\");\n\nvar g =
global.get(\"g\");\n\nvar h = global.get(\"h\");\n\nvar i = global.get(\"i\");\n\nvar j =
global.get(\"j\");\n\nvar k = global.get(\"k\");\n\nvar l = global.get(\"l\");\n\nvar m =
global.get(\"m\");\n\nvar n = global.get(\"n\");\n\nvar o = global.get(\"o\");\n\nvar p =
global.get(\"p\");\n\nvar q = global.get(\"q\");\n\nvar r = global.get(\"r\");\n\nvar s =
global.get(\"s\");\n\nvar t = global.get(\"t\");\n\nvar u = global.get(\"u\");\n\n\n//send the
user values to service endpoint\n\nmsg.payload = \n\n{\n\n\"fields\":{\n\n\"Country\", \"Year\",
\"Status\", \"BMI\", \"Adult_Mortality\", \"Infant_Deaths\", \"Alcohol\",
\"Percentage_Expenditure\", \"Hepatitis_B\", \"Under_Five_Deaths\", \"Polio\",
\"Total_Expenditure\", \"Diphtheria\", \"HIV/AIDS\", \"GDP\", \"Population\",
\"Thinness_10_19_years\", \"Thinness_5_9_years\", \n\n
\"Income_Composition_of_Resources\", \"Schooling\",
\"Measles\"},\n\n\"values\":[[a,b,c,d,e,f,g,h,i,j,k,l,m,n,o,p,q,r,s,t,u]]};\n\n\nreturn
msg;\n\n,\"outputs\":1,\"noerr\":0,\"x\":210,\"y\":180,\"wires\":[[\"1db0486e.22e378\"]]],{\"id\":
\"e164f901.a85778\",\"type\":\"http
request\",\"z\":\"719c1fee.e5b94\",\"name\":\"\",\"method\":\"POST\",\"ret\":\"obj\",\"paytoqs\":fal

```

```
se,"url":"https://iam.cloud.ibm.com/identity/token","tls":"","persist":false,"proxy":
","authType":"basic","x":410,"y":80,"wires":[["9e7824f4.ef15f8"]]],{"id":"3c53f487.6
1d98c","type":"ui_text","z":"719c1fee.e5b94","group":"d01d05bb.7b14d8","order":
1,"width":0,"height":0,"name":"","label":"Prediction","format":"{{msg.payload}}","la
yout":"row-
spread","x":720,"y":400,"wires":[[]],{"id":"d01d05bb.7b14d8","type":"ui_group","z":
","name":"Life Expectancy
Prediction","tab":"bf44b03d.ccb08","order":1,"disp":true,"width":"6","collapse":fals
e},{id":"bf44b03d.ccb08","type":"ui_tab","z":"","name":"Home
Page","icon":"dashboard","disabled":false,"hidden":false}]
```

## NOTEBOOK

### Analyzing the dataset

#### Importing required libraries

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.preprocessing import OneHotEncoder
from sklearn.model_selection import train_test_split
from sklearn.neural_network import MLPClassifier
from sklearn.neighbors import KNeighborsClassifier
from sklearn.svm import SVC
from sklearn.preprocessing import StandardScaler
from sklearn.compose import ColumnTransformer, make_column_transformer
from sklearn.pipeline import make_pipeline
from sklearn.impute import SimpleImputer
from sklearn.gaussian_process import GaussianProcessClassifier
from sklearn.gaussian_process.kernels import RBF
from sklearn.tree import DecisionTreeClassifier
from sklearn.ensemble import RandomForestClassifier, AdaBoostClassifier
from sklearn.naive_bayes import GaussianNB
from sklearn.preprocessing import LabelEncoder
from sklearn.metrics import accuracy_score
from collections import OrderedDict
from sklearn.svm import SVR
from sklearn.ensemble import RandomForestRegressor
from sklearn.model_selection import ShuffleSplit
from sklearn.model_selection import cross_val_score
from sklearn.linear_model import LinearRegression
from sklearn.tree import DecisionTreeRegressor
from sklearn.metrics import r2_score, mean_squared_error
```

## Reading the dataset

```
import pandas as pd
from botocore.client import Config
import ibm_boto3

def __iter__(self): return 0

# @hidden_cell
# The following code accesses a file in your IBM Cloud Object Storage.
# It includes your credentials.
# You might want to remove those credentials before you share the
# notebook.
client_f6b95832a0b74b43a30591d2690780ec =
ibm_boto3.client(service_name='s3',
                 ibm_api_key_id='SBRRWsbqKBIgYomid_cQf2nCV8J_6XQhLmSdeRcxFLJ',
                 ibm_auth_endpoint="https://iam.cloud.ibm.com/oidc/token",
                 config=Config(signature_version='oauth'),
                 endpoint_url='https://s3.eu-
geo.objectstorage.service.networklayer.com')

body =
client_f6b95832a0b74b43a30591d2690780ec.get_object(Bucket='lifeexpectan
cy01-donotdelete-pr-zao7bdogflparf',Key='Life Expectancy
Data.csv')['Body']
# add missing __iter__ method, so pandas accepts body as file-like
object
if not hasattr(body, "__iter__"): body.__iter__ = types.MethodType(
__iter__, body )

# If you are reading an Excel file into a pandas DataFrame, replace
`read_csv` by `read_excel` in the next statement.
data = pd.read_csv(body)
data.head()
```

```
data.head()
```

```
data.shape # (2938, 22)
```

```
data.describe()
```

```
data.info()
```

```
data.isnull().sum()
```

## Handling missing value

```
country_list = data.Country.unique()
len(country_list)
```

```
country_list = data.Country.unique()
fill_list = ['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']

```

## Filling missing value according to country column using interpolate()

```

for country in country_list:
    data.loc[data['Country'] == country, fill_list] =
data.loc[data['Country'] == country, fill_list].interpolate()
data.dropna(inplace=True)

```

```
data.shape # (1987, 22) size reduced
```

```
data.isna().sum()
```

## Correlation matrix

```

corrMatrix = data.corr()
corrMatrix.style.background_gradient(cmap='plasma', low=.5,
high=0).highlight_null('red')

```

Thinness 1-19 years must be thinness 10-19 years.

## Renaming the columns as it contains trailing spaces

```

data.rename(columns={" BMI ": "BMI", 'Life expectancy ': 'Life
expectancy',
                    "under-five deaths ": "under-five deaths", "Measles
": "Measles", "Diphtheria ": "Diphtheria",
                    ' HIV/AIDS': "HIV/AIDS",
                    " thinness 1-19 years": "thinness 10-19 years", "
thinness 5-9 years": "thinness 5-9 years"}, inplace=True)

```

## Removing outliers

Taking numeric features (country, year, status columns are excluded)

```

col_dict = {'Life expectancy':1 , 'Adult Mortality':2 ,
            'Alcohol':3 , 'percentage expenditure': 4, 'Hepatitis B': 5,
            'Measles' : 6, 'BMI': 7, 'under-five deaths' : 8, 'Polio' : 9,
            'Total expenditure' :10,
            'Diphtheria':11, 'HIV/AIDS':12, 'GDP':13, 'Population' :14,
            'thinness 10-19 years' :15, 'thinness 5-9 years' :16,
            'Income composition of resources' : 17, 'Schooling' :18, 'infant
deaths':19}

```

## Showing outliers using box plot

```
import matplotlib.pyplot as plt
plt.figure(figsize=(20,30))

for variable,i in col_dict.items():
    plt.subplot(5,4,i)
    plt.boxplot(data[variable],whis=1.5)
    plt.title(variable)

plt.show()
```

## BMI has no outliers

```
import numpy as np

for variable in col_dict.keys():
    q75, q25 = np.percentile(data[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((data[variable] > max_val) | (data[variable] < min_val))[0])),
len((np.where((data[variable] > max_val) | (data[variable] < min_val))[0]))*100/1987))
```

## 18 columns having outliers

```
from scipy.stats.mstats import winsorize
winsorized_Life_Expectancy = winsorize(data['Life expectancy'],(0.01,0))
winsorized_Adult_Mortality = winsorize(data['Adult Mortality'],(0,0.03))
winsorized_Infant_Deaths = winsorize(data['infant deaths'],(0,0.10))
winsorized_Alcohol = winsorize(data['Alcohol'],(0,0.01))
winsorized_Percentage_Exp = winsorize(data['percentage expenditure'],(0,0.12))
winsorized_HepatitisB = winsorize(data['Hepatitis B'],(0.11,0))
winsorized_Measles = winsorize(data['Measles'],(0,0.19))
winsorized_Under_Five_Deaths = winsorize(data['under-five deaths'],(0,0.12))
winsorized_Polio = winsorize(data['Polio'],(0.09,0))
winsorized_Tot_Exp = winsorize(data['Total expenditure'],(0,0.01))
winsorized_Diphtheria = winsorize(data['Diphtheria'],(0.10,0))
winsorized_HIV = winsorize(data['HIV/AIDS'],(0,0.16))
winsorized_GDP = winsorize(data['GDP'],(0,0.13))
winsorized_Population = winsorize(data['Population'],(0,0.14))
winsorized_thinness_10_19_years = winsorize(data['thinness 10-19 years'],(0,0.04))
winsorized_thinness_5_9_years = winsorize(data['thinness 5-9 years'],(0,0.04))
```

```

winsorized_Income_Comp_Of_Resources = winsorize(data['Income
composition of resources'], (0.05,0))
winsorized_Schooling = winsorize(data['Schooling'], (0.02,0.01))

winsorized_list =
[winsorized_Life_Expectancy,winsorized_Adult_Mortality,winsorized_Alcohol,
winsorized_Measles,winsorized_Infant_Deaths,

winsorized_Percentage_Exp,winsorized_HepatitisB,winsorized_Under_Five_Deaths,
winsorized_Polio,winsorized_Tot_Exp,winsorized_Diphtheria,

winsorized_HIV,winsorized_GDP,winsorized_Population,winsorized_thinness_10_19_years,
winsorized_thinness_5_9_years,
winsorized_Income_Comp_Of_Resources,winsorized_Schooling]

for variable in winsorized_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 in : {}".format(len(np.where((variable > max_val) | (variable < min_val))[0])))

```

### Adding 18 new columns having no outliers to the dataframe

```

data['winsorized_Life_Expectancy'] = winsorized_Life_Expectancy
data['winsorized_Adult_Mortality'] = winsorized_Adult_Mortality
data['winsorized_Infant_Deaths'] = winsorized_Infant_Deaths
data['winsorized_Alcohol'] = winsorized_Alcohol
data['winsorized_Percentage_Exp'] = winsorized_Percentage_Exp
data['winsorized_HepatitisB'] = winsorized_HepatitisB
data['winsorized_Under_Five_Deaths'] = winsorized_Under_Five_Deaths
data['winsorized_Polio'] = winsorized_Polio
data['winsorized_Tot_Exp'] = winsorized_Tot_Exp
data['winsorized_Diphtheria'] = winsorized_Diphtheria
data['winsorized_HIV'] = winsorized_HIV
data['winsorized_GDP'] = winsorized_GDP
data['winsorized_Population'] = winsorized_Population
data['winsorized_thinness_10_19_years'] = winsorized_thinness_10_19_years
data['winsorized_thinness_5_9_years'] = winsorized_thinness_5_9_years
data['winsorized_Income_Comp_Of_Resources'] = winsorized_Income_Comp_Of_Resources
data['winsorized_Schooling'] = winsorized_Schooling
data['winsorized_Measles'] = winsorized_Measles

```

```
data.shape #More 18 columns are added
```

## EDA

```
data.columns
```

```
sns.distplot(data['Life expectancy'],kde=True)
```

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

```
disease_cols.corr()
```

```
sns.pairplot(disease_cols,diag_kind='kde')
```

**Hence all the features are significant to predict the target variable**

```
col = ['Life expectancy', 'winsorized_Life_Expectancy', 'Adult Mortality', 'winsorized_Adult_Mortality', 'infant deaths', 'winsorized_Infant_Deaths', 'Alcohol', 'winsorized_Alcohol', 'percentage expenditure', 'winsorized_Percentage_Exp', 'Hepatitis B', 'winsorized_HepatitisB', 'under-five deaths', 'winsorized_Under_Five_Deaths', 'Polio', 'winsorized_Polio', 'Total expenditure', 'winsorized_Tot_Exp', 'Diphtheria', 'winsorized_Diphtheria', 'HIV/AIDS', 'winsorized_HIV', 'GDP', 'winsorized_GDP', 'Population', 'winsorized_Population', 'thinness 10-19 years', 'winsorized_thinness_10_19_years', 'thinness 5-9 years', 'winsorized_thinness_5_9_years', 'Income composition of resources', 'winsorized_Income_Comp_Of_Resources', 'Schooling', 'winsorized_Schooling', 'Measles', 'winsorized_Measles', 'GDP', 'winsorized_GDP']

plt.figure(figsize=(15,75))

for i in range(len(col)):
    plt.subplot(19,2,i+1)
    plt.hist(data[col[i]])
    plt.title(col[i])

plt.show()
```

```
data.describe(include= 'O') #include specifies the list of datatype to be included .here is Object
```

```
plt.figure(figsize=(6,6))
plt.bar(data.groupby('Status')['Status'].count().index, data.groupby('Status')['winsorized_Life_Expectancy'].mean())
plt.ylabel("Avg Life_Expectancy")
```



```
plt.title("Life_Expectancy w.r.t Status")
plt.show()
```

```
country_data =
data.groupby('Country')['winsorized_Life_Expectancy'].mean().sort_values(ascending=True)
country_data.plot(kind='bar', figsize=(50,15), fontsize=30, color='g')
plt.title("Life_Expectancy w.r.t Country", fontsize=30)
plt.xlabel("Country", fontsize=30)
plt.ylabel("Avg Life_Expectancy")
plt.show()
plt.figure(figsize=(7,5))
plt.figure(figsize=(7,5))
plt.bar(data.groupby('Year')['Year'].count().index, data.groupby('Year')['winsorized_Life_Expectancy'].mean())
plt.xlabel("Year", fontsize=12)
plt.ylabel("Avg Life_Expectancy", fontsize=12)
plt.show()
```

```
cor_matrix=data.corr()
print(cor_matrix['winsorized_Life_Expectancy'].sort_values(ascending=False))
```

```
import seaborn as sns
from pandas.plotting import scatter_matrix
attributes=
['winsorized_Life_Expectancy', 'winsorized_Income_Comp_Of_Resources', 'winsorized_Schooling',
'winsorized_Diphtheria', 'winsorized_Polio', 'winsorized_Adult_Mortality',
'winsorized_Alcohol', 'winsorized_Measles', 'winsorized_Infant_Deaths',

'winsorized_Percentage_Exp', 'winsorized_HepatitisB', 'winsorized_Under_Five_Deaths', 'winsorized_Tot_Exp',

'winsorized_HIV', 'winsorized_GDP', 'winsorized_Population', 'winsorized_thinness_10_19_years', 'winsorized_thinness_5_9_years']
cormat=data[attributes].corr()
plt.figure(figsize=(15,15))
sns.heatmap(cormat, square=True, annot=True, linewidths=.5)
plt.show()
```

```
round(data[['Status', 'winsorized_Life_Expectancy']].groupby(['Status']).mean(), 2)
```

Since 'status' is a categorical feature, we have to find the correlation with Life expectancy.

```
import scipy.stats as stats
stats.ttest_ind(data.loc[data['Status']=='Developed', 'winsorized_Life_Expectancy'], data.loc[data['Status']=='Developing', 'winsorized_Life_Expectancy'])
```

```
data.columns
```

Now our data has no null values and no outliers.

## Creating a new dataframe with refined data

```
new_data=pd.DataFrame(data=data,columns=['Country', 'Year', 'Status',
    'BMI', 'winsorized_Adult_Mortality',
    '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_10_19_years',
    'winsorized_thinness_5_9_years',
    'winsorized_Income_Comp_Of_Resources', 'winsorized_Schooling',
    'winsorized_Measles',
    'winsorized_Life_Expectancy'])
```

```
new_data.shape
```

```
new_data.head()
```

```
new_data.rename(columns={
    'winsorized_Adult_Mortality':'Adult_Mortality',
    'winsorized_Infant_Deaths': 'Infant_Deaths',
    'winsorized_Alcohol':'Alcohol',
    'winsorized_Percentage_Exp':'Percentage_Expenditure',
    'winsorized_HepatitisB':'Hepatitis_B',
    'winsorized_Under_Five_Deaths':'Under_Five_Deaths',
    'winsorized_Polio':'Polio',
    'winsorized_Tot_Exp':'Total_Expenditure',
    'winsorized_Diphtheria':'Diphtheria',
    'winsorized_HIV':'HIV/AIDS',
    'winsorized_GDP':'GDP',
    'winsorized_Population':'Population',
    'winsorized_thinness_10_19_years':'Thinness_10_19_years',
    'winsorized_thinness_5_9_years':'Thinness_5_9_years',
    'winsorized_Income_Comp_Of_Resources':'Income_Composition_of_Resources',
    'winsorized_Schooling':'Schooling',
    'winsorized_Measles':'Measles',
    'winsorized_Life_Expectancy':'Life_Expectancy' },inplace=True)
```

```
new_data.head()
```

```
new_data.columns
```

## Separating the input features and label

```
X = new_data.drop('Life_Expectancy', axis=1)
Y = pd.DataFrame(data=new_data,columns=['Life_Expectancy'])
```

```
X.head()
```

```
Y.head()
```

## Splitting the data into train set and test set

```
X_train, X_test, Y_train, Y_test = train_test_split(X, Y, test_size = 0.2, random_state = 42)
```

## Creating a pipeline

```
numeric_features = ['Year', 'BMI',  
                    'Adult_Mortality', 'Infant_Deaths', 'Alcohol',  
                    'Percentage_Expenditure',  
                    'Hepatitis_B', 'Under_Five_Deaths', 'Polio',  
                    'Total_Expenditure',  
                    'Diphtheria', 'HIV/AIDS', 'GDP', 'Population',  
                    'Thinness_10_19_years',  
                    'Thinness_5_9_years', 'Income_Composition_of_Resources',  
                    'Schooling',  
                    'Measles']  
categorical_features = ['Country', 'Status']
```

```
from sklearn.pipeline import Pipeline  
from sklearn.preprocessing import OneHotEncoder  
  
categorical_transformer = Pipeline(steps=[  
    ('onehot', OneHotEncoder(handle_unknown='ignore')),  
])
```

```
from sklearn.impute import SimpleImputer  
from sklearn.preprocessing import StandardScaler  
  
numeric_transformer = Pipeline(steps=[  
    ('imputer', SimpleImputer(strategy='median'))  
])
```

```
from sklearn.compose import ColumnTransformer  
  
preprocessor = ColumnTransformer(  
    transformers=[  
        ('cat', categorical_transformer, categorical_features),  
        ('num', numeric_transformer, numeric_features)  
    ]  
)
```

## Finding best algorithm

```
models = OrderedDict([
    ( "Linear Regression", Pipeline([
        ('preprocessor',
preprocessor),
        ('LRegressor',
LinearRegression())]) ),
    ( "Decision Tree Regressor", Pipeline([
        ('preprocessor',
preprocessor),
        ('DTRegressor',
DecisionTreeRegressor())]) ),
    ( "Random Forest Regressor", Pipeline([
        ('preprocessor',
preprocessor),
        ('RFRegressor',
RandomForestRegressor())]) ),
])
```

```
scores = {}
for (name, model) in models.items():
    model.fit(X_train,Y_train)
    scores[name] =r2_score(model.predict(X_test), Y_test)

scores = OrderedDict(sorted(scores.items()))
scores
```

**Hence Random forest regression is the most suitable algorithm for this dataset.**

## Random forest regression

```
RFRegressor = Pipeline([
    ('preprocessor', preprocessor),
    ('RFRegressor', RandomForestRegressor())
])
```

```
RFRegressor.fit(X_train,Y_train)
```

```
predict= RFRegressor.predict(X_test)
```

```
r2_score(predict, Y_test)
```

## Deploying model

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

```
from watson_machine_learning_client import  
WatsonMachineLearningAPIClient
```

```
wml_credentials={  
    "apikey": "SfC-TWpXIzz7_c1P74LM1fRYcD4IbCPedggjYCqYA5ss",  
    "instance_id": "b819851b-fdeb-48da-8c0e-655136f2b699",  
    "url": "https://eu-gb.ml.cloud.ibm.com"  
}
```

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

```
model_props = {client.repository.ModelMetaNames.AUTHOR_NAME: "Joshni",  
               client.repository.ModelMetaNames.AUTHOR_EMAIL:  
               "princias987@gmail.com",  
               client.repository.ModelMetaNames.NAME:  
               "Life_expectancy"}
```

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

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

```
deployment = client.deployments.create(published_model_uid,  
name="life_expectancy")  
scoring_endpoint = client.deployments.get_scoring_url(deployment)  
scoring_endpoint
```