PROJECT REPORT

PREDICTING LIFE EXPECTANCY USING MACHINE LEARNING

SUBMITTED BY : Anjana Anil

Govt.Model Engineering College

1.Introduction

1.1 Overview

Life expectancy is a statistical measure of the average time a human being is expected to live, Life expectancy depends on various factors: Regional variations, Economic Circumstances, Sex Differences, Mental Illnesses, Physical Illnesses, Education, Year of their birth and other demographic factors. This problem statement provides a way to predict the 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.

1.2 Purpose

The project tries to create a model based on data provided by the World Health Organization (WHO) to evaluate the life expectancy for different countries in years. The data offers a timeframe from 2000 to 2015. The data originates from here: https://www.kaggle.com/kumarajarshi/life-expectancy-who/data The output algorithms have been used to test if they can maintain their accuracy in predicting the life expectancy

2. Literary survey

2.1 Existing Problem

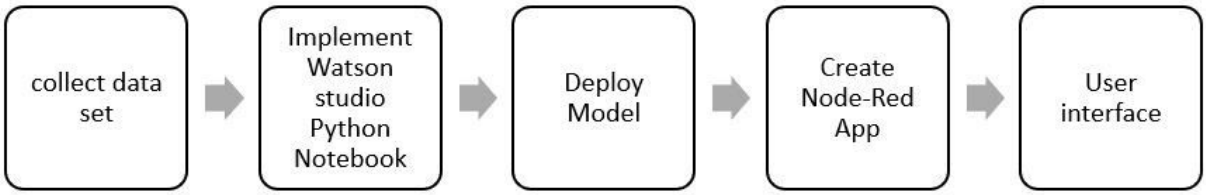
Although there have been lot of studies undertaken in the past on factors affecting life expectancy considering demographic variables, income composition and mortality rates. It was found that affect of immunization and human development index was not taken into account in the past.This gives motivation to resolve both the factors stated previously by formulating a regression model based on considering data from a period of 2000 to 2015 for all the countries. Important immunization like Hepatitis B, Polio and Diphtheria will also be considered. In a nutshell, this study will focus on immunization factors, mortality factors, economic factors, social factors and other health related factors as well. This will help in suggesting which area should be given importance in order to efficiently improve the life expectancy of its population.

2.2 Proposed solution

In order to make regression models we use a lot of libraries and tools like Linear Regression and train test split from sklearn,Pandas, Numpy, Matplotlib, etc. in Python.
I made this research based on Life Expectancy data set which is published by The Global Health Observatory (GHO) data repository under World Health Organization (WHO) and predict the life expectancy of a person based on the different features in the dataset.

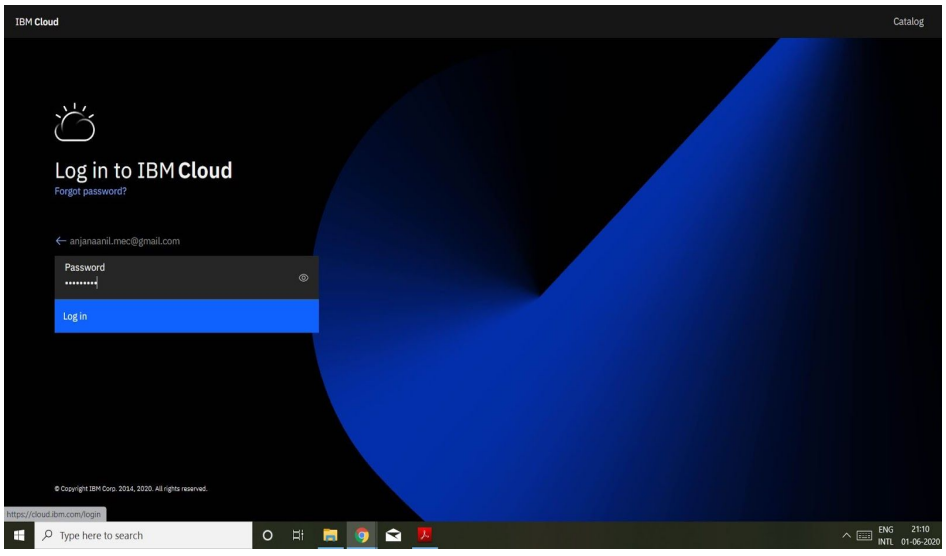
3.Theoretical Analysis

3.1 Block diagram



3.2 Hardware/software designing

Create IBM Account:
Go to [IBM Platform] to create an IBM account.

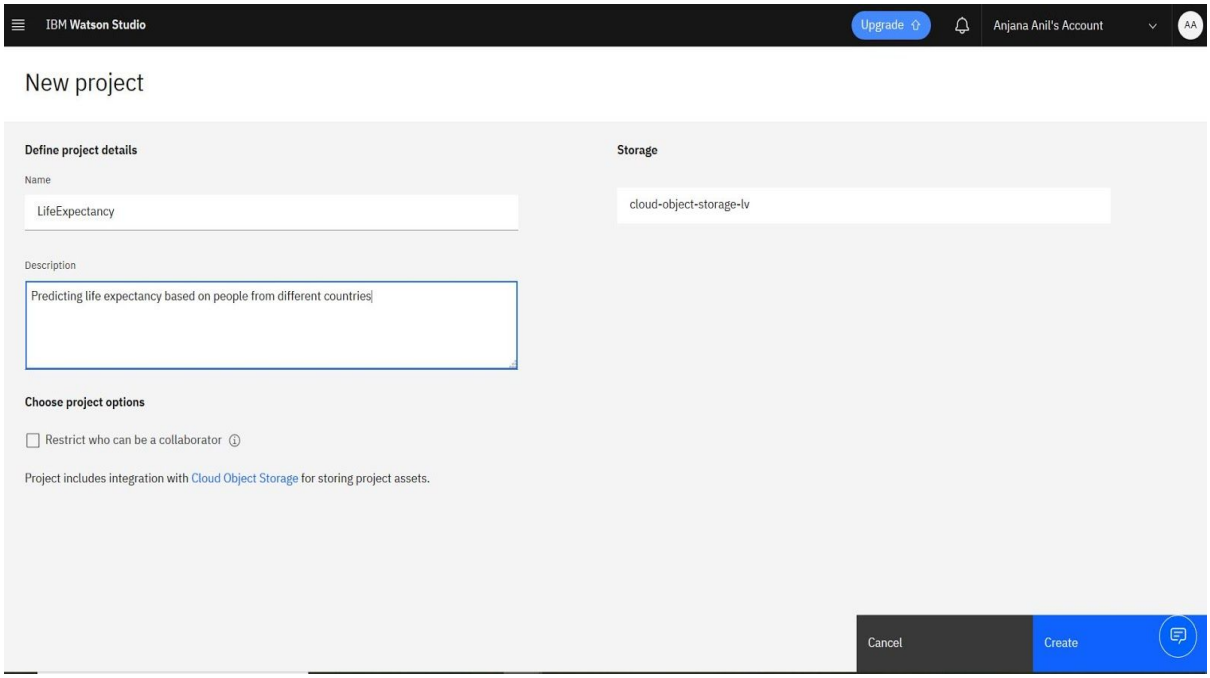


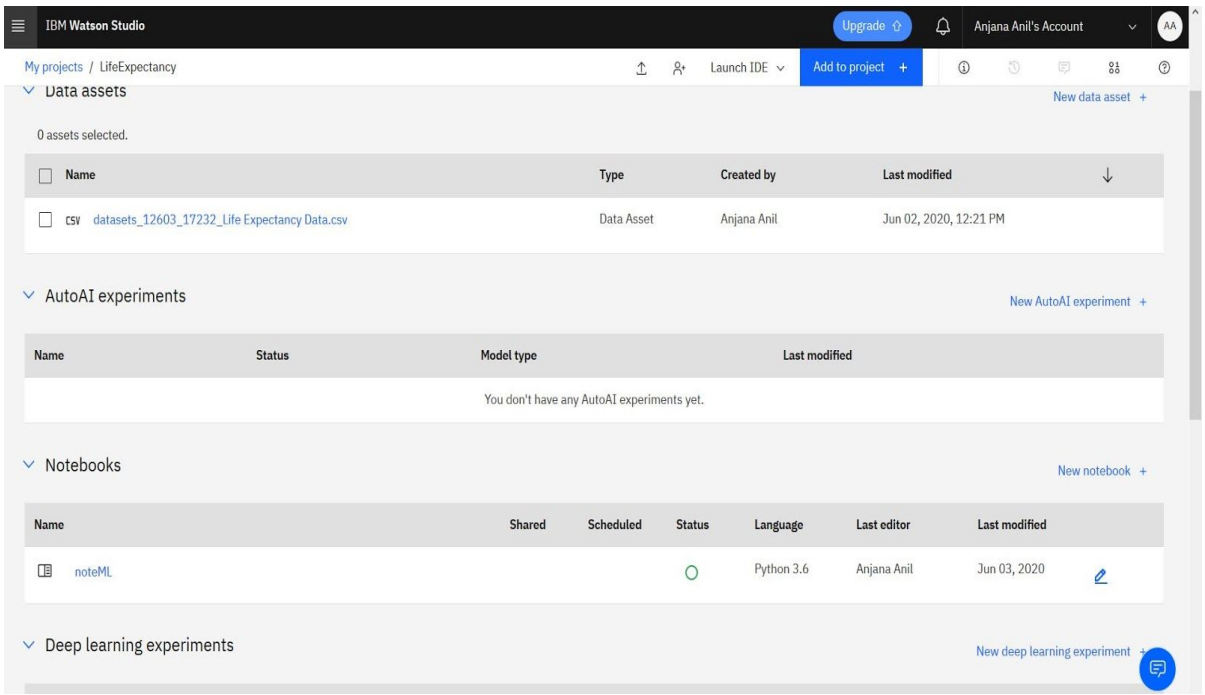
To create Watson Studio Service

1.Go catalog a Search Watson Studio in search bar

2.Click on Get started -> Create project and add cloud object storage

3.Inside the project Tab -> Assets -> Add Data Set and Notebook





To create Node-Red Application

1.From Dashboard select catalog.

2.Select Software

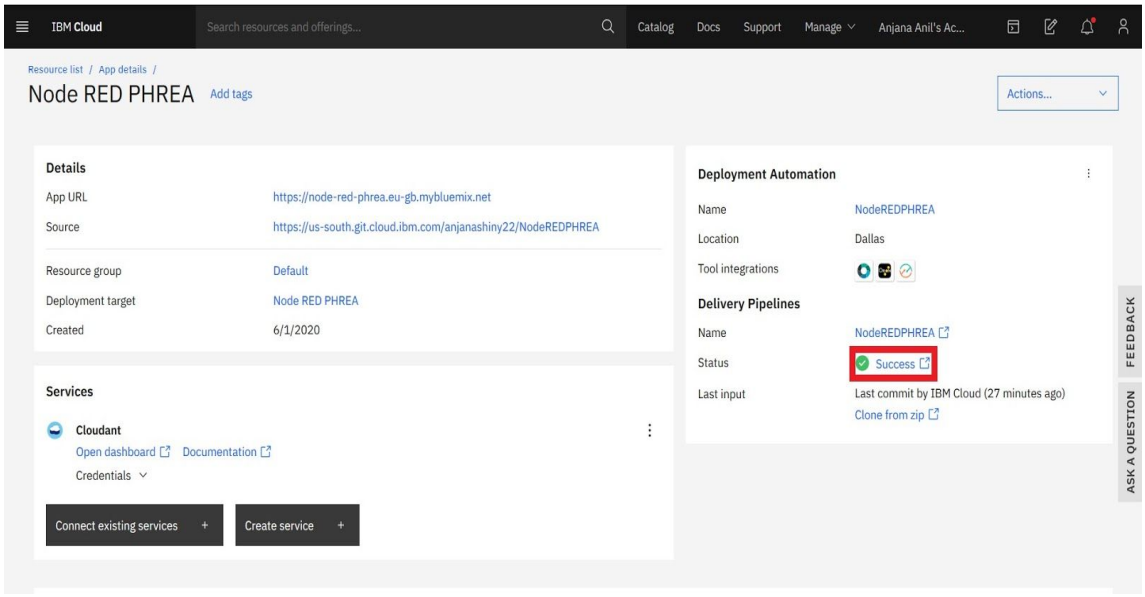
3.Select Web and Application

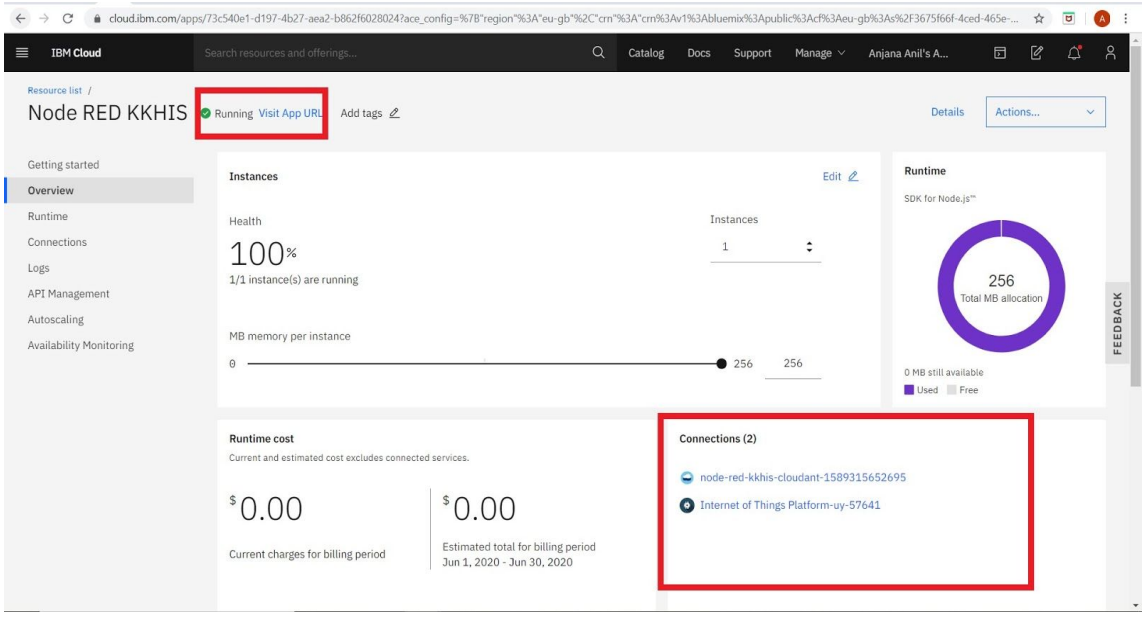
4.Select Node-Red Application

5.Keep all the values to default and continue the App development.

6.Once you select on click Create app and then proceed to click on Deploy App.

7.After deployment wait for the status turn into Success.





8.App URL will appear after successful deployment . Click on the App URL

· Add connections to the associated services clicking Connections à Add Connections.

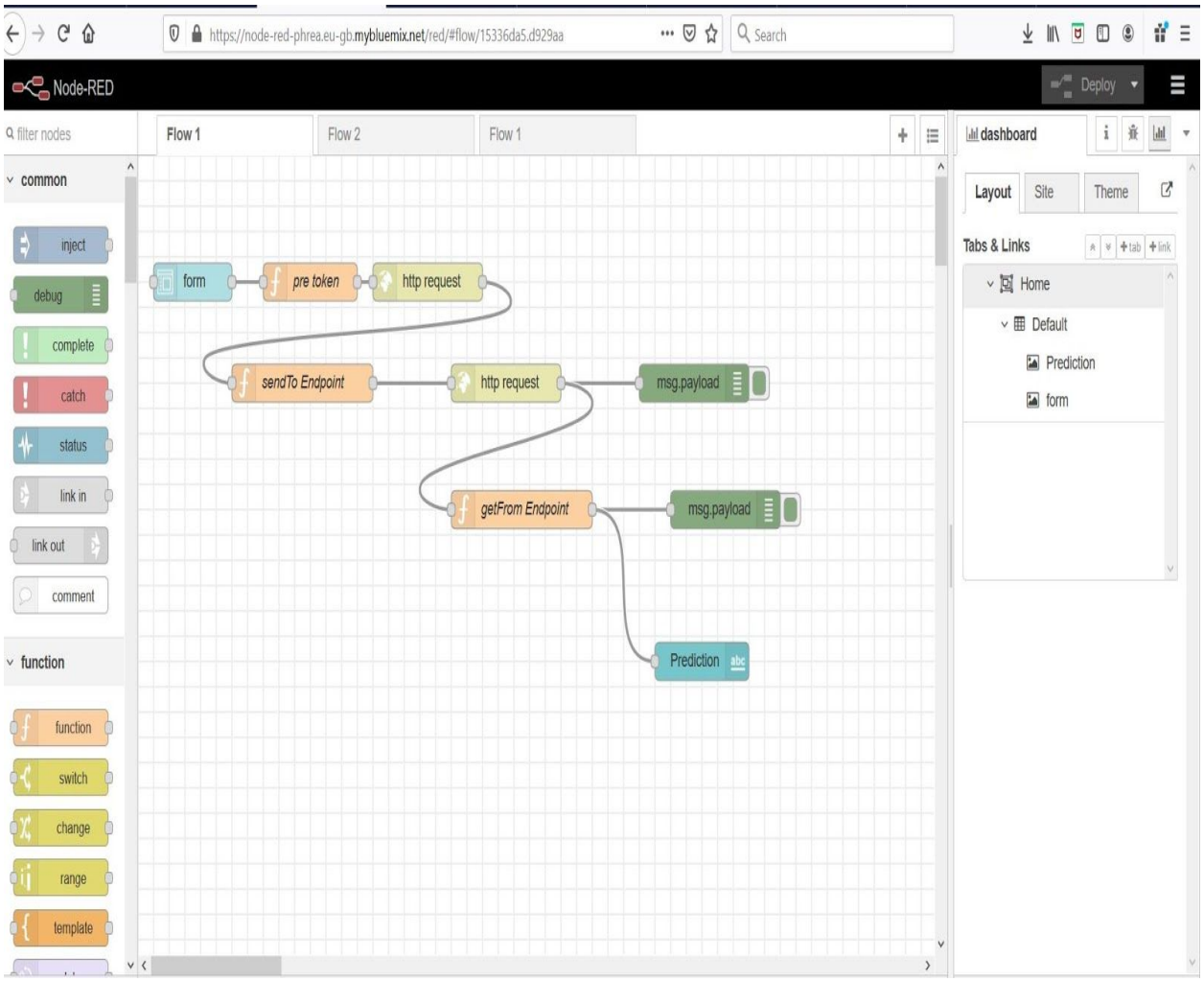
· Click on Visit App URL in the Overview page.

· Setup the Node RED editor.

9.Continue clicking on Next and click on Finish at the end to launch the Node RED editor.

10.Click on Node-RED flow editor.

11.Flow for the App is created by taking the necessary nodes.



12.Install “node-red-dashboard” from Manage Palette.

The link to Flow: https://node-red-phrea.eu-gb.mybluemix.net/red/#flow/15336da5.d929aa

4.Experimental Investigations

The data set is analyzed to get more information about the features.

```
In [3]: LifeExpectancyData.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 2938 entries, 0 to 2937
Data columns (total 22 columns):
Country                          2938 non-null object
Year                             2938 non-null int64
Status                           2938 non-null object
Life expectancy                  2928 non-null float64
Adult Mortality                  2928 non-null float64
infant deaths                    2938 non-null int64
Alcohol                          2744 non-null float64
percentage expenditure           2938 non-null float64
Hepatitis B                      2385 non-null float64
Measles                          2938 non-null int64
 BMI                             2904 non-null float64
under-five deaths                2938 non-null int64
Polio                            2919 non-null float64
Total expenditure                2712 non-null float64
Diphtheria                       2919 non-null float64
 HIV/AIDS                        2938 non-null float64
GDP                              2490 non-null float64
Population                       2286 non-null float64
 thinness  1-19 years            2904 non-null float64
 thinness 5-9 years              2904 non-null float64
Income composition of resources  2771 non-null float64
Schooling                        2775 non-null float64
dtypes: float64(16), int64(4), object(2)
memory usage: 505.0+ KB
```

The data contains 2 object values ‘Country’ and ‘Status’ which is converted to an int value.

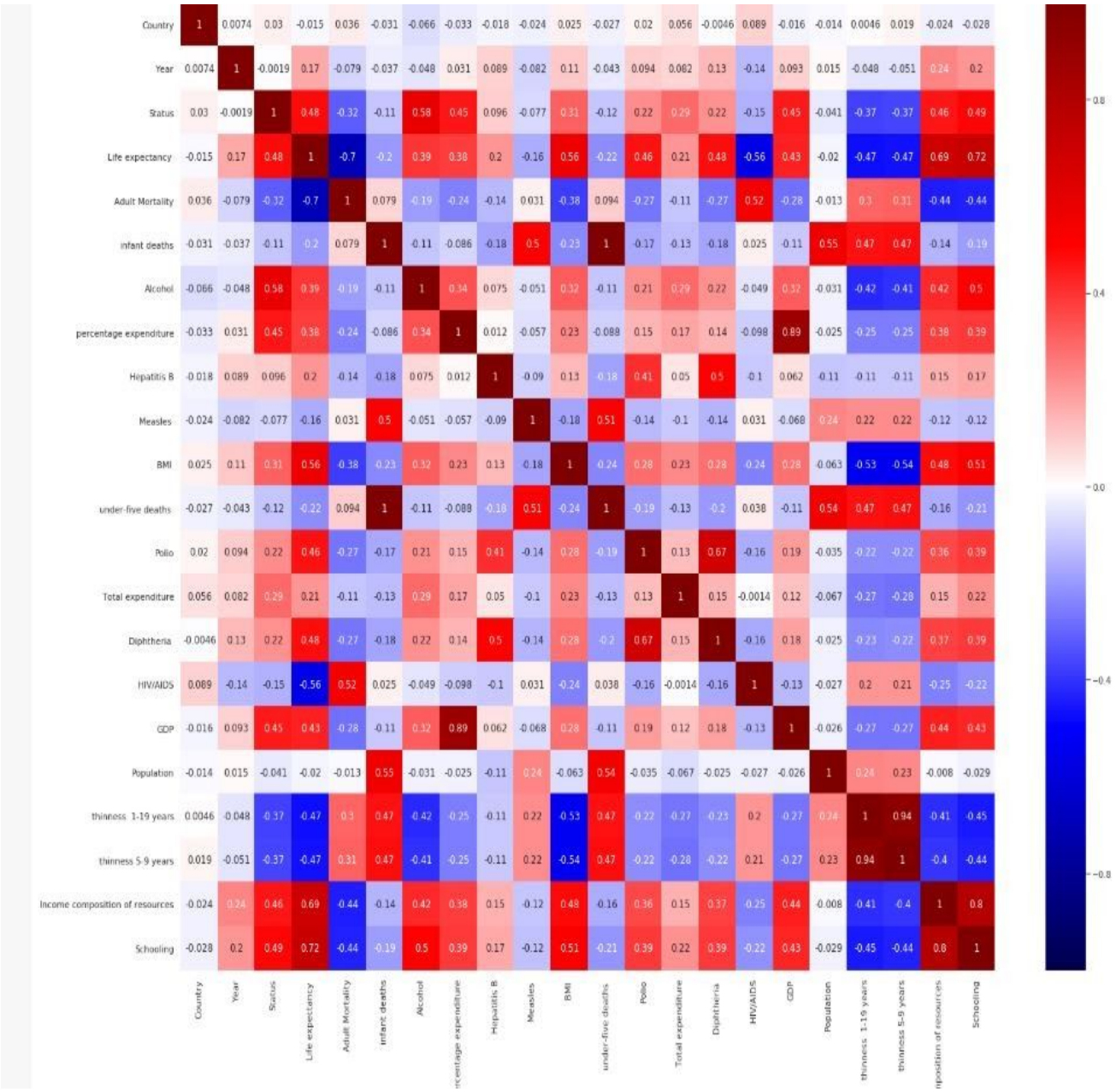
```
In [6]: LifeExpectancyData.describe()
```

Out[6]:

	Country	Year	Status	Life expectancy	Adult Mortality	infant deaths	Alcohol	percentage expenditure	Hepatitis B	Measles	...
count	2938.000000	2938.000000	2938.000000	2928.000000	2928.000000	2938.000000	2744.000000	2938.000000	2385.000000	2938.000000	...
mean	92.328455	2007.518720	1.174268	69.224932	164.796448	30.303948	4.602861	738.251295	80.940461	2419.592240	...
std	53.044716	4.613841	0.379405	9.523867	124.292079	117.926501	4.052413	1987.914858	25.070016	11467.272489	...
min	1.000000	2000.000000	1.000000	36.300000	1.000000	0.000000	0.010000	0.000000	1.000000	0.000000	...
25%	46.000000	2004.000000	1.000000	63.100000	74.000000	0.000000	0.877500	4.685343	77.000000	0.000000	...
50%	92.000000	2008.000000	1.000000	72.100000	144.000000	3.000000	3.755000	64.912906	92.000000	17.000000	...
75%	138.000000	2012.000000	1.000000	75.700000	228.000000	22.000000	7.702500	441.534144	97.000000	360.250000	...
max	193.000000	2015.000000	2.000000	89.000000	723.000000	1800.000000	17.870000	19479.911610	99.000000	212183.000000	...

8 rows × 22 columns

Null values are replaced by the mean of the data present in each row and the data analysed to get the features that contribute more to predicting the data more accurately with help of heatmap which shows the correlation of features .The legend tells that the red shades show higher and positive correlation, while the blue shades 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.

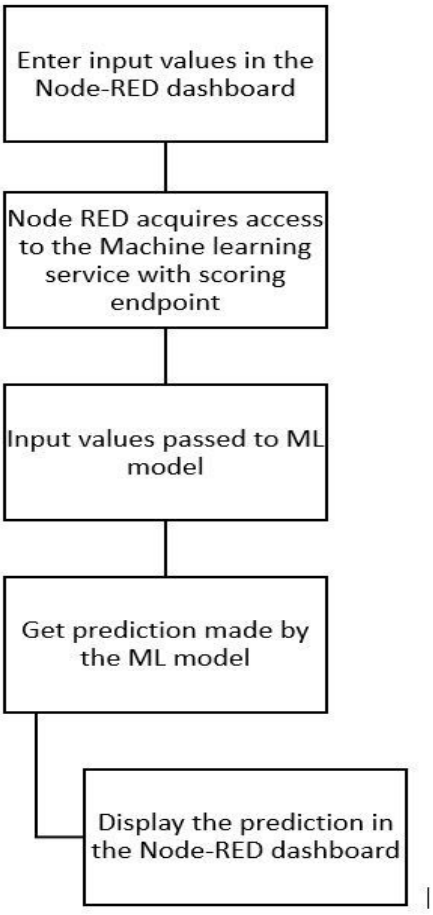


Selected features for prediction:

```
print(coefficients)
```

	0	0
0	Adult Mortality	-0.022826
1	Income composition of resources	6.688339
2	BMI	0.061542
3	GDP	0.000058
4	Schooling	0.895610
5	HIV/AIDS	-0.471609

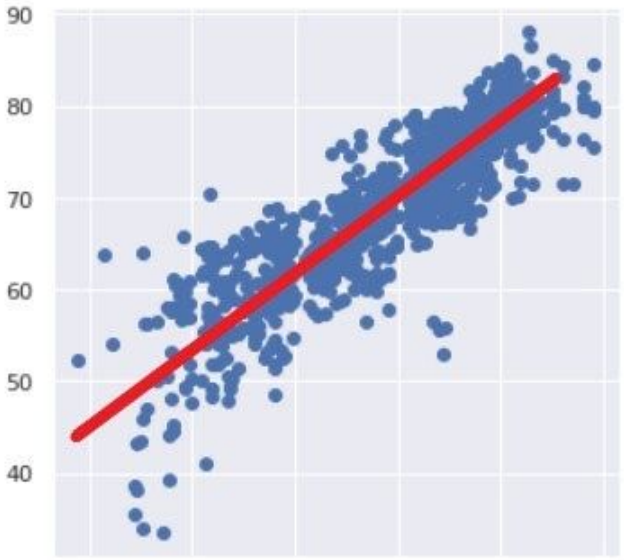
5.Flow chart



6.Result

```
In [23]: plt.scatter(y_test,predictions)
```

```
Out[23]: <matplotlib.collections.PathCollection at 0x7feae1f79438>
```



Accuracy of result:

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

```
In [22]: from sklearn.metrics import r2_score
         print(r2_score(y_test,predictions )*100)
```

```
80.6517868291436
```

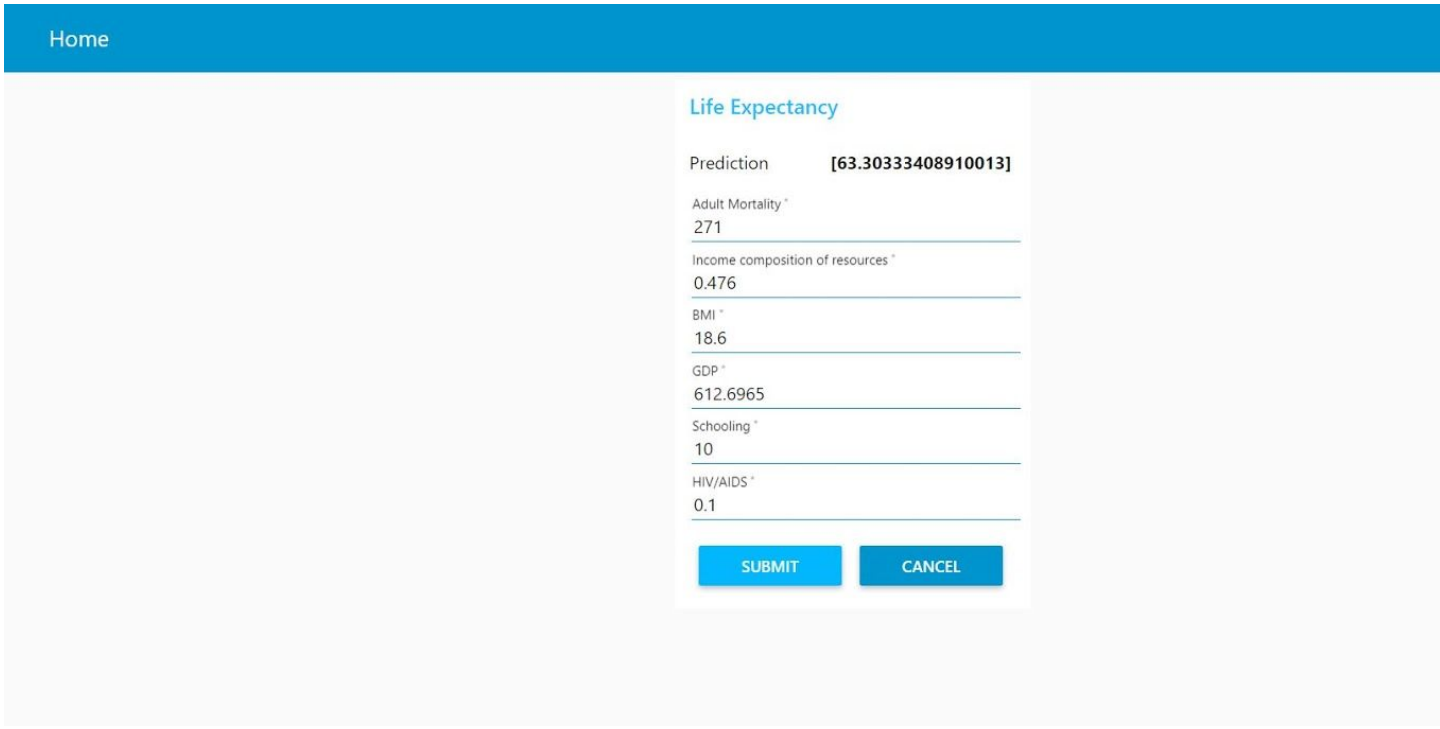
With this method we can get accuracy of 80% for predicting the life expectancy.

The Node-RED is then integrated with the machine learning services created in the IBM cloud.Scoring endpoint URL to send payload data to a model or function deployment for analysis (for example, to classify the data, or make a prediction from the data).

```
In [38]: scoring_endpoint = client.deployments.get_scoring_url(deployment)

In [39]: scoring_endpoint

Out[39]: 'https://eu-gb.ml.cloud.ibm.com/v3/wml_instances/f02389af-bdad-4f92-9d76-2d1c27
```



Link to Dashboard :
https://node-red-phrea.eu-gb.mybluemix.net/ui/#!/0?socketid=D5cxusRTqHAu-7X1AAAK

7.Advantages and Disadvantages

- Machine Learning can review large volumes of data and discover specific trends and patterns that would not be apparent to humans and thereby increasing the accuracy in prediction.
- We can create a user interface easily with help of Node-RED and give the input to the model and predicts the Life expectancy.

- Countries or people may get to know of features they should focus on to improve the life expectancy of a person.
- Machine Learning requires massive data sets to train on, and these should be inclusive/unbiased, and of good quality. There can also be times where they must wait for new data to be generated.
- ML needs time to let the algorithms learn and develop enough to fulfill their purpose with a considerable amount of accuracy and relevancy. It also needs massive resources to function. This can mean additional requirements of computer power.

8.Applications

Life expectancy is one of the most important factors in end-of-life decision making. It helps to determine the features that will help in course of treatment and helps to anticipate the procurement of health care services and facilities.People will be able to predict their life expectancy and will help them to take necessary step to improve their health.

9.Conclusion

From the project the main features that contributed to increasing the accuracy were Adult Mortality,Income composition of resources,BMI,GDP,Schooling, HIV/AIDS.We could reduce 22 features to the best possible fit of 6 features.From the analysis we could conclude that high values for above the resources will help in improving the life expectancy of the a person.Therefore people,governments and countries can use the application to predict the life expectancy of the people and can adopt good incentives and could improve resources that significantly contribute to the higher life expectancy

10.Future Scope

This application can be used to suggest good health practices and life style to the users based on their daily activities and provide suggestions for exercises for improving their health.Pharmaceutical companies can check which diseases impact more people and therefore impact life expectancy and based on this manufacture medicine.

11.Bibliography

1. https://www.kaggle.com/kumarajarshi/life-expectancy-who
2. https://bookdown.org/caoying4work/watsonstudio-workshop/jn.html#deploy-model-as-web-service
3. https://developer.ibm.com/technologies/machine-learning/series/learning-path-machine-learning-for-developers/

12.Appendix

Source code

In [1]:

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

In [2]:

```

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

def __iter__(self): return 0

client = ibm_boto3.client(service_name='s3',
  ibm_api_key_id='klOxRjUmxKUKAo37LJSl7l2LWfPjRe6TGR82qD6Dkt9-',
  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.get_object(Bucket='lifeexpectancy-donotdelete-pr-mun0aus0pjxqew',Key='datasets_12603_17232_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 )


LifeExpectancyData = pd.read_csv(body)
LifeExpectancyData.head()
```

Out[2]:In [3]:

```
LifeExpectancyData.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 2938 entries, 0 to 2937
Data columns (total 22 columns):
Country                          2938 non-null object
Year                             2938 non-null int64
Status                           2938 non-null object
Life expectancy                  2928 non-null float64
Adult Mortality                  2928 non-null float64
infant deaths                    2938 non-null int64
Alcohol                          2744 non-null float64
percentage expenditure           2938 non-null float64
Hepatitis B                      2385 non-null float64
Measles                          2938 non-null int64
 BMI                             2904 non-null float64
under-five deaths                2938 non-null int64
Polio                            2919 non-null float64
Total expenditure                2712 non-null float64
Diphtheria                       2919 non-null float64
 HIV/AIDS                        2938 non-null float64
GDP                              2490 non-null float64
Population                       2286 non-null float64
 thinness  1-19 years            2904 non-null float64
 thinness 5-9 years              2904 non-null float64
Income composition of resources    2771 non-null float64
Schooling                        2775 non-null float64
dtypes: float64(16), int64(4), object(2)
memory usage: 505.0+ KB
```

In [4]:

```
LifeExpectancyData['Country'] = LifeExpectancyData['Country'].replace(['Afghanistan' , 'Albania' ,
'Algeria' , 'Angola' , 'Antigua and Barbuda' , 'Argentina' , 'Armenia' , 'Australia' , 'Austria' ,
'Azerbaijan' , 'Bahamas' , 'Bahrain' , 'Bangladesh' , 'Barbados' , 'Belarus' , 'Belgium' , 'Belize' , 'Benin' ,
'Bhutan' , 'Bolivia (Plurinational State of)' , 'Bosnia and Herzegovina' , 'Botswana' , 'Brazil' , 'Brunei
Darussalam' , 'Bulgaria' , 'Burkina Faso' , 'Burundi' , "Côte d'Ivoire" , 'Cabo Verde' , 'Cambodia' ,
'Cameroon' , 'Canada' , 'Central African Republic' , 'Chad' , 'Chile' , 'China' , 'Colombia' , 'Comoros' ,
```

```
'Congo' , 'Costa Rica' , 'Croatia' , 'Cuba' , 'Cyprus' , 'Czechia' , "Democratic People's Republic of
Korea" , 'Democratic Republic of the Congo' , 'Denmark' , 'Djibouti' , 'Dominican Republic' , 'Ecuador'
, 'Egypt' , 'El Salvador' , 'Equatorial Guinea' , 'Eritrea' , 'Estonia' , 'Ethiopia' , 'Fiji' , 'Finland' , 'France' ,
'Gabon' , 'Gambia' , 'Georgia' , 'Germany' , 'Ghana' , 'Greece' , 'Grenada' , 'Guatemala' , 'Guinea' ,
'Guinea-Bissau' , 'Guyana' , 'Haiti' , 'Honduras' , 'Hungary' , 'Iceland' , 'India' , 'Indonesia' , 'Iran
(Islamic Republic of)' , 'Iraq' , 'Ireland' , 'Israel' , 'Italy' , 'Jamaica' , 'Japan' , 'Jordan' , 'Kazakhstan' ,
'Kenya' , 'Kiribati' , 'Kuwait' , 'Kyrgyzstan' , "Lao People's Democratic Republic" , 'Latvia' , 'Lebanon' ,
'Lesotho' , 'Liberia' , 'Libya' , 'Lithuania' , 'Luxembourg' , 'Madagascar' , 'Malawi' , 'Malaysia' ,
'Maldives' , 'Mali' , 'Malta' , 'Mauritania' , 'Mauritius' , 'Mexico' , 'Micronesia (Federated States of)' ,
'Mongolia' , 'Montenegro' , 'Morocco' , 'Mozambique' , 'Myanmar' , 'Namibia' , 'Nepal' ,
'Netherlands' , 'New Zealand' , 'Nicaragua' , 'Niger' , 'Nigeria' , 'Norway' , 'Oman' , 'Pakistan' ,
'Panama' , 'Papua New Guinea' , 'Paraguay' , 'Peru' , 'Philippines' , 'Poland' , 'Portugal' , 'Qatar' ,
'Republic of Korea' , 'Republic of Moldova' , 'Romania' , 'Russian Federation' , 'Rwanda' , 'Saint Lucia'
, 'Saint Vincent and the Grenadines' , 'Samoa' , 'Sao Tome and Principe' , 'Saudi Arabia' , 'Senegal' ,
'Serbia' , 'Seychelles' , 'Sierra Leone' , 'Singapore' , 'Slovakia' , 'Slovenia' , 'Solomon Islands' ,
'Somalia' , 'South Africa' , 'South Sudan' , 'Spain' , 'Sri Lanka' , 'Sudan' , 'Suriname' , 'Swaziland' ,
'Sweden' , 'Switzerland' , 'Syrian Arab Republic' , 'Tajikistan' , 'Thailand' , 'The former Yugoslav
republic of Macedonia' , 'Timor-Leste' , 'Togo' , 'Tonga' , 'Trinidad and Tobago' , 'Tunisia' , 'Turkey' ,
'Turkmenistan' , 'Uganda' , 'Ukraine' , 'United Arab Emirates' , 'United Kingdom of Great Britain and
Northern Ireland' , 'United Republic of Tanzania' , 'United States of America' , 'Uruguay' ,
'Uzbekistan' , 'Vanuatu' , 'Venezuela (Bolivarian Republic of)' , 'Viet Nam' , 'Yemen' , 'Zambia' ,
'Zimbabwe' , 'Cook Islands' , 'Dominica' , 'Marshall Islands' , 'Monaco' , 'Nauru' , 'Niue' , 'Palau' ,
'Saint Kitts and Nevis' , 'San Marino' , 'Tuvalu'], [1 ,2 ,3 ,4 ,5 ,6 ,7 ,8 ,9 ,10 ,11 ,12 ,13 ,14 ,15 ,16 ,17
,18 ,19 ,20 ,21 ,22 ,23 ,24 ,25 ,26 ,27 ,28 ,29 ,30 ,31 ,32 ,33 ,34 ,35 ,36 ,37 ,38 ,39 ,40 ,41 ,42 ,43 ,44
,45 ,46 ,47 ,48 ,49 ,50 ,51 ,52 ,53 ,54 ,55 ,56 ,57 ,58 ,59 ,60 ,61 ,62 ,63 ,64 ,65 ,66 ,67 ,68 ,69 ,70 ,71
,72 ,73 ,74 ,75 ,76 ,77 ,78 ,79 ,80 ,81 ,82 ,83 ,84 ,85 ,86 ,87 ,88 ,89 ,90 ,91 ,92 ,93 ,94 ,95 ,96 ,97 ,98
,99 ,100 ,101 ,102 ,103 ,104 ,105 ,106 ,107 ,108 ,109 ,110 ,111 ,112 ,113 ,114 ,115 ,116 ,117 ,118
,119 ,120 ,121 ,122 ,123 ,124 ,125 ,126 ,127 ,128 ,129 ,130 ,131 ,132 ,133 ,134 ,135 ,136 ,137 ,138
,139 ,140 ,141 ,142 ,143 ,144 ,145 ,146 ,147 ,148 ,149 ,150 ,151 ,152 ,153 ,154 ,155 ,156 ,157 ,158
,159 ,160 ,161 ,162 ,163 ,164 ,165 ,166 ,167 ,168 ,169 ,170 ,171 ,172 ,173 ,174 ,175 ,176 ,177 ,178
,179 ,180 ,181 ,182 ,183 ,184 ,185 ,186 ,187 ,188 ,189 ,190 ,191 ,192 ,193])
```

In [4]:

```
LifeExpectancyData['Status'] = LifeExpectancyData['Status'].replace(['Developing', 'Developed'],[1,
2])
```

In [5]:

```
LifeExpectancyData.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 2938 entries, 0 to 2937
Data columns (total 22 columns):
Country                          2938 non-null int64
Year                             2938 non-null int64
Status                           2938 non-null int64
Life expectancy                  2928 non-null float64
Adult Mortality                  2928 non-null float64
infant deaths                    2938 non-null int64
Alcohol                          2744 non-null float64
percentage expenditure           2938 non-null float64
Hepatitis B                      2385 non-null float64
Measles                          2938 non-null int64
 BMI                             2904 non-null float64
under-five deaths                2938 non-null int64
Polio                            2919 non-null float64
Total expenditure                2712 non-null float64
Diphtheria                       2919 non-null float64
 HIV/AIDS                        2938 non-null float64
GDP                              2490 non-null float64
Population                       2286 non-null float64
 thinness  1-19 years            2904 non-null float64
 thinness 5-9 years              2904 non-null float64
Income composition of resources    2771 non-null float64
Schooling                        2775 non-null float64
dtypes: float64(16), int64(6)
memory usage: 505.0 KB
```

In [6]:

In [5]:

```
LifeExpectancyData = LifeExpectancyData.fillna(LifeExpectancyData.mean())
```

In [6]:

```
LifeExpectancyData.columns
```

Out[6]:

```
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 [1832]:

In [1833]:

```
LifeExpectancyData.columns
```

Out[1833]:

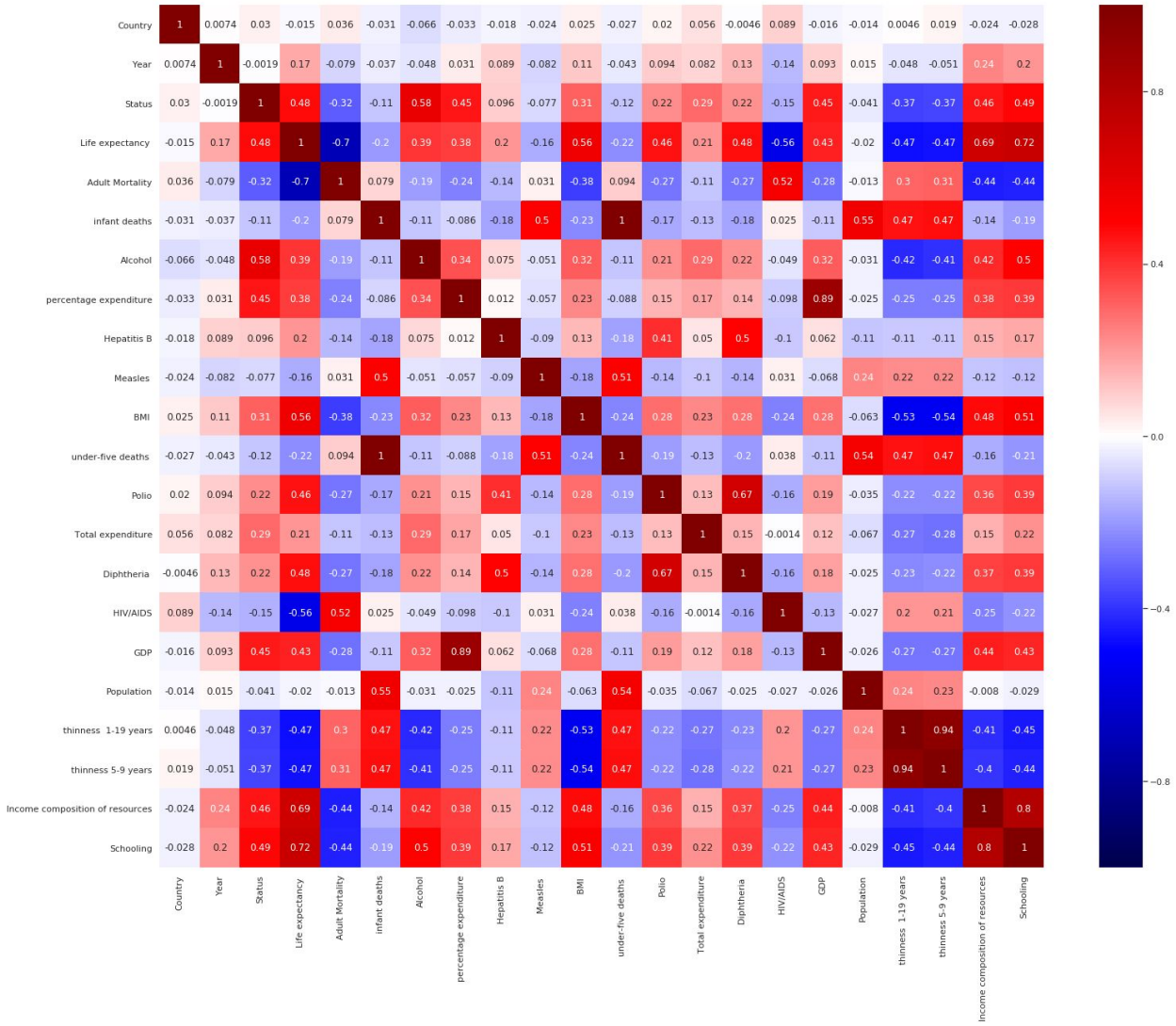
```
Index(['Country', 'Year', 'Status', 'Life expectancy ', 'Adult Mortality',
       'infant deaths', 'Alcohol', 'percentage expenditure', 'Hepatitis B',
       'Measles ', ' BMI ', 'under-five deaths ', 'Polio', 'Total expenditure',
       'Diphtheria ', ' HIV/AIDS', 'GDP', 'Population',
       ' thinness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

In [1834]:

```
sns.set(rc={'figure.figsize':(25,20)})
sns.heatmap(LifeExpectancyData.corr(),
            cmap='seismic',annot=True,vmin=-1,vmax=1)
```

Out[1834]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f9fec8179e8>
```



In [9]:

```
LifeExpectancyData.columns
```

Out[9]:

```
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 [10]:

```
X = LifeExpectancyData[['Adult Mortality','Income composition of resources',' BMI
','GDP','Schooling',' HIV/AIDS']]
```

```
y = LifeExpectancyData[['Life expectancy ']]
```

In [11]:

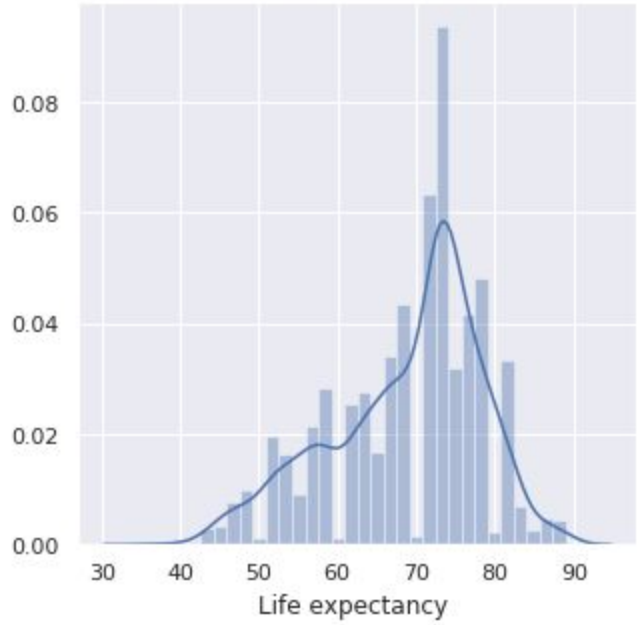
```
LifeExpectancyData = LifeExpectancyData.fillna(0)
LifeExpectancyData['Life expectancy ']= LifeExpectancyData['Life expectancy '].astype(int)
pd.options.mode.chained_assignment = None
```

In [12]:

```
sns.set(rc={'figure.figsize':(5,5)})
sns.distplot(LifeExpectancyData['Life expectancy '])
```

Out[12]:

```
<matplotlib.axes._subplots.AxesSubplot at 0x7fc16355b908>
```



In [13]:

```
from sklearn.model_selection import train_test_split
```

In [14]:

```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.4)
```

In [15]:

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

In [16]:

```
lm = LinearRegression()
```

In [17]:

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

Out[17]:

```
LinearRegression(copy_X=True, fit_intercept=True, n_jobs=None,
```

normalize=False)

In [18]:

```
print(lm.intercept_)
```

```
[55.78888274]
```

In [19]:

```
coefficients = pd.concat([pd.DataFrame(X.columns),pd.DataFrame(np.transpose(lm.coef_))], axis = 1)
```

In [20]:

```
print(coefficients)
```

```
                          0         0
0                Adult Mortality -0.021385
1  Income composition of resources  7.293769
2                            BMI   0.060752
3                            GDP   0.000061
4                      Schooling   0.872127
5                       HIV/AIDS -0.504165
```

In [21]:

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

In [23]:

```
from sklearn.metrics import r2_score
print(r2_score(y_test,predictions )*100)
```

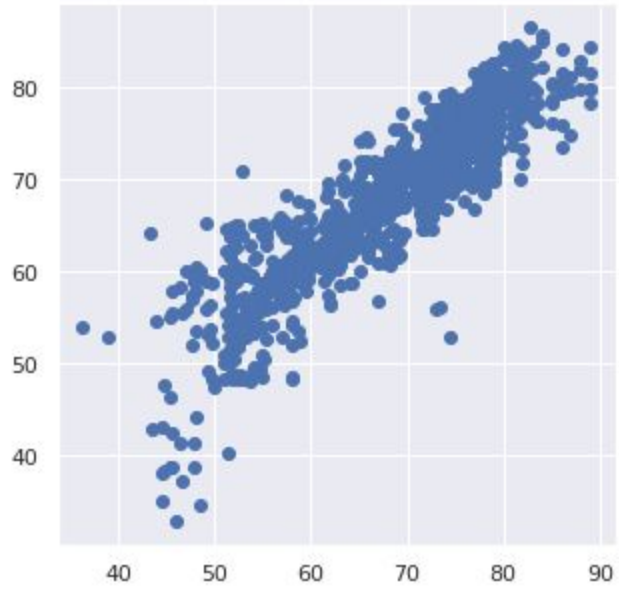
```
80.6517868291436
```

In [24]:

```
plt.scatter(y_test,predictions)
```

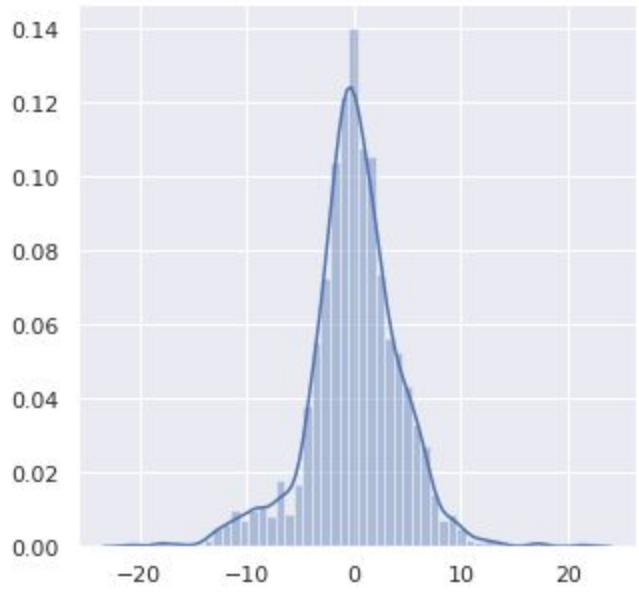
Out[24]:

```
<matplotlib.collections.PathCollection at 0x7fc1631065f8>
```



In [25]:

```
sns.distplot((y_test-predictions),bins=50);
```



In [26]:

```
from sklearn import metrics
```

In [27]:

```
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: 3.0857457093052725

MSE: 18.32084474015159
RMSE: 4.280285590956705

In [28]:

```
!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: urllib3 in
/opt/conda/envs/Python36/lib/python3.6/site-packages (from watson-machine-learning-client)
(1.24.1)
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: pandas in
/opt/conda/envs/Python36/lib/python3.6/site-packages (from watson-machine-learning-client)
(0.24.1)
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: lomond in
/opt/conda/envs/Python36/lib/python3.6/site-packages (from watson-machine-learning-client)
(0.3.3)
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: ibm-cos-sdk in
/opt/conda/envs/Python36/lib/python3.6/site-packages (from watson-machine-learning-client)
(2.4.3)
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: pytz>=2011k in
/opt/conda/envs/Python36/lib/python3.6/site-packages (from
pandas->watson-machine-learning-client) (2018.9)
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: 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: 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: 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: 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: 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: 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)
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)
```

In [30]:

```
from watson_machine_learning_client import WatsonMachineLearningAPIClient
```

In [31]:

```
wml_credentials={
 "apikey": "fsyeRmnwWYtQTBV6YSJ4AO33DyfY8o4CshPrq5yrnodQ",
 "instance_id": "f02389af-bdad-4f92-9d76-2d1c270e6f0b",
 "url": "https://eu-gb.ml.cloud.ibm.com",
}
```

In [32]:

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

In [33]:

```
model_props = {client.repository.ModelMetaNames.AUTHOR_NAME: "Anjana",
```

```
        client.repository.ModelMetaNames.AUTHOR_EMAIL: "anjanaanil.mec@gmail.com",
        client.repository.ModelMetaNames.NAME: "LifeExpectancyData"}
```

In [34]:

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

In [35]:

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

In [36]:

```
published_model_uid
```

Out[36]:

```
'a00b6c98-4820-42e8-a23f-04820365d10e'
```

In [37]:

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

```
#########################################################################################

Synchronous deployment creation for uid: 'a00b6c98-4820-42e8-a23f-04820365d10e' started

#########################################################################################


INITIALIZING
DEPLOY_SUCCESS


------------------------------------------------------------------------------------------
Successfully finished deployment creation,
deployment_uid='a696111d-6ca5-4a27-b2af-f56282649cd1'
------------------------------------------------------------------------------------------
```

In [38]:

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

In [39]:

```
scoring_endpoint
```

Out[39]:

'https://eu-gb.ml.cloud.ibm.com/v3/wml_instances/f02389af-bdad-4f92-9d76-2d1c270e6f0b/deployments/a696111d-6ca5-4a27-b2af-f56282649cd1/online'

In [ ]: