Predicting Life Expectancy using Machine Learning

Project ID: SPS_PRO_215

By:
Divyanshi Agarwal
SBID: SB20200044380
Email: divyanshiagarwal23@gmail.com

TABLE OF CONTENTS

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 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 purpose of this project is to predict life expectancy of a person with highest possible frequency. Life Expectancy affects the economic growth, Population growth, Personal growth, growth in health sector, insurance sector. So, predicting life expectancy and taking actions accordingly before hand helps in ensuring the development of the country.

2 LITERATURE SURVEY

2.1 Existing problem

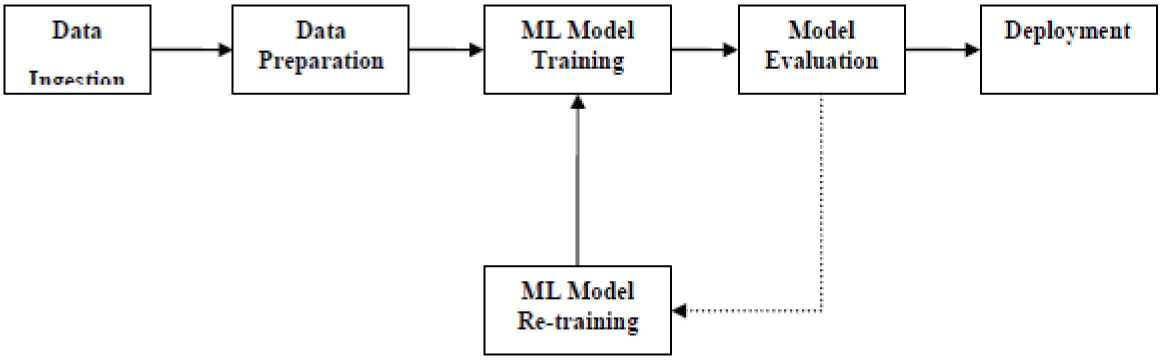
Predicting a human's life expectancy has been a long-term question to humankind. Many calculations and research have been done to create an equation despite it being impractical to simplify these variables into one equation. Currently there are various smart devices and applications such as smartphone apps and wearable devices that provide wellness and fitness tracking. Some apps provide health related data such as sleep monitoring, heart rate measuring, and calorie expenditure collected and processed by the devices and servers in the cloud. However no existing works provide the Personalized Life expectancy

2.2 Proposed solution

A WebApp with integrated machine learning model with high accuracy that could predict the life expectancy of a coutry based on various factors like BMI, GDP, Alcohol intake, Year, HIV/AIDS, etc. in real time.

3 THEORITICAL ANALYSIS

3.1 Block diagram



3.2 Hardware / Software designing

- **Functional Requirements**

  The project provides a way to predict average life expectancy of people living in a country taking into account 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 which are already given.

- **Technical Requirements**

  ➤ Python(3.8)

  ➤ IBM Cloud

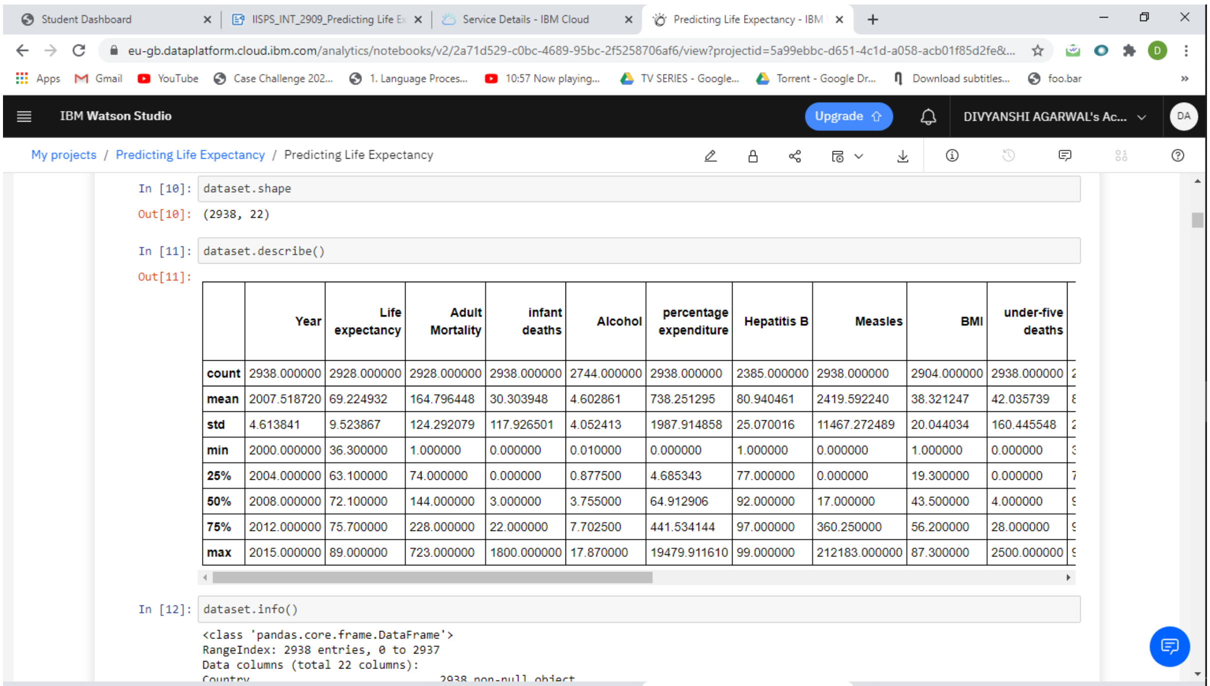
  ➤ IBM Watson

- **Software Requirements**

  No specific software requirements as work would be done using IBM Cloud and documentation would be maintained using ZOHO writer.

4 EXPERIMENTAL INVESTIGATIONS

Analysing and visualising the data:



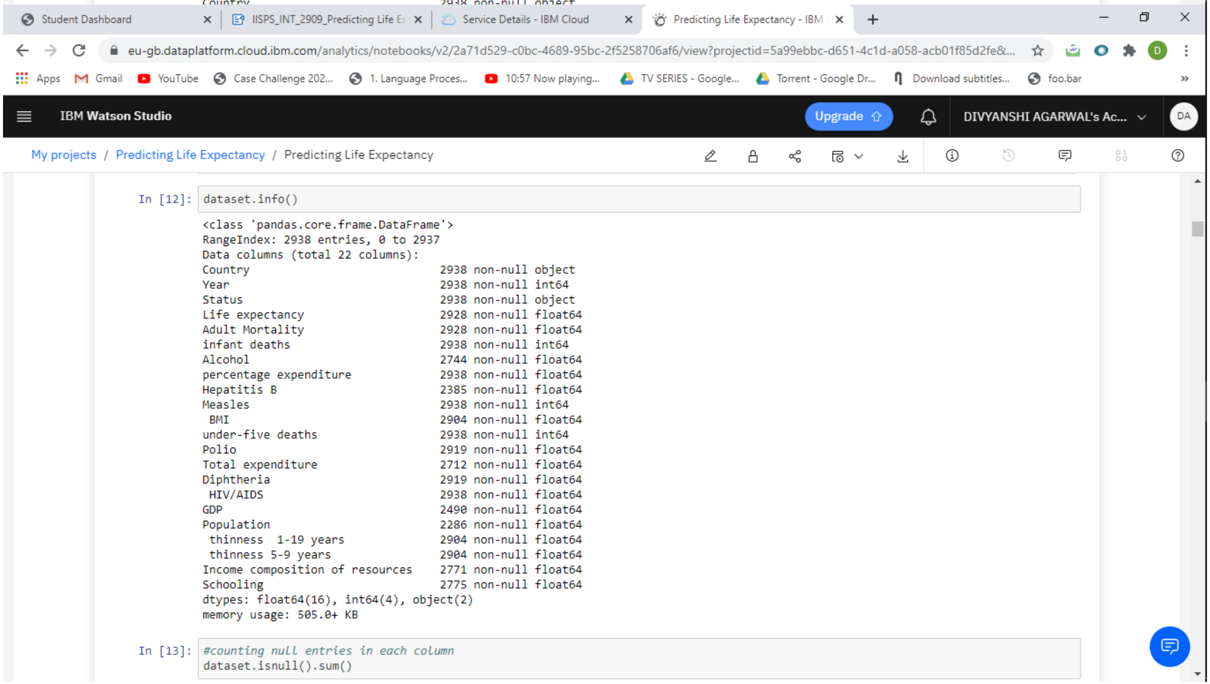

```
In [10]: dataset.shape
Out[10]: (2938, 22)

In [11]: dataset.describe()
Out[11]:
```

	Year	Life expectancy	Adult Mortality	infant deaths	Alcohol	percentage expenditure	Hepatitis B	Measles	BMI	under-five deaths
count	2938.000000	2928.000000	2928.000000	2938.000000	2744.000000	2938.000000	2385.000000	2938.000000	2904.000000	2938.000000
mean	2007.518720	69.224932	164.796448	30.303948	4.602861	738.251295	80.940461	2419.592240	38.321247	42.035739
std	4.613841	9.523867	124.292079	117.926501	4.052413	1987.914858	25.070016	11467.272489	20.044034	160.445548
min	2000.000000	36.300000	1.000000	0.000000	0.010000	0.000000	1.000000	0.000000	1.000000	0.000000
25%	2004.000000	63.100000	74.000000	0.000000	0.877500	4.685343	77.000000	0.000000	19.300000	0.000000
50%	2008.000000	72.100000	144.000000	3.000000	3.755000	64.912906	92.000000	17.000000	43.500000	4.000000
75%	2012.000000	75.700000	228.000000	22.000000	7.702500	441.534144	97.000000	360.250000	56.200000	28.000000
max	2015.000000	89.000000	723.000000	1800.000000	17.870000	19479.911610	99.000000	212183.000000	87.300000	2500.000000

```
In [12]: dataset.info()

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 2938 entries, 0 to 2937
Data columns (total 22 columns):
```



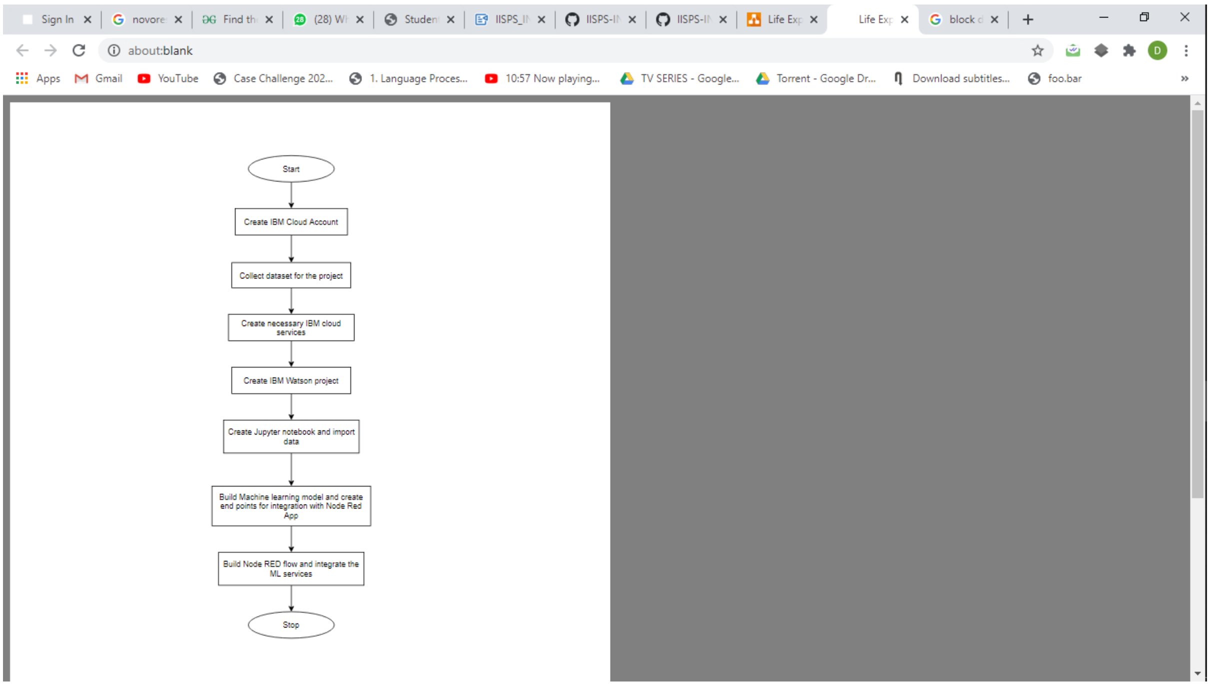

```
In [12]: dataset.info()

<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 [13]: #counting null entries in each column
dataset.isnull().sum()
```

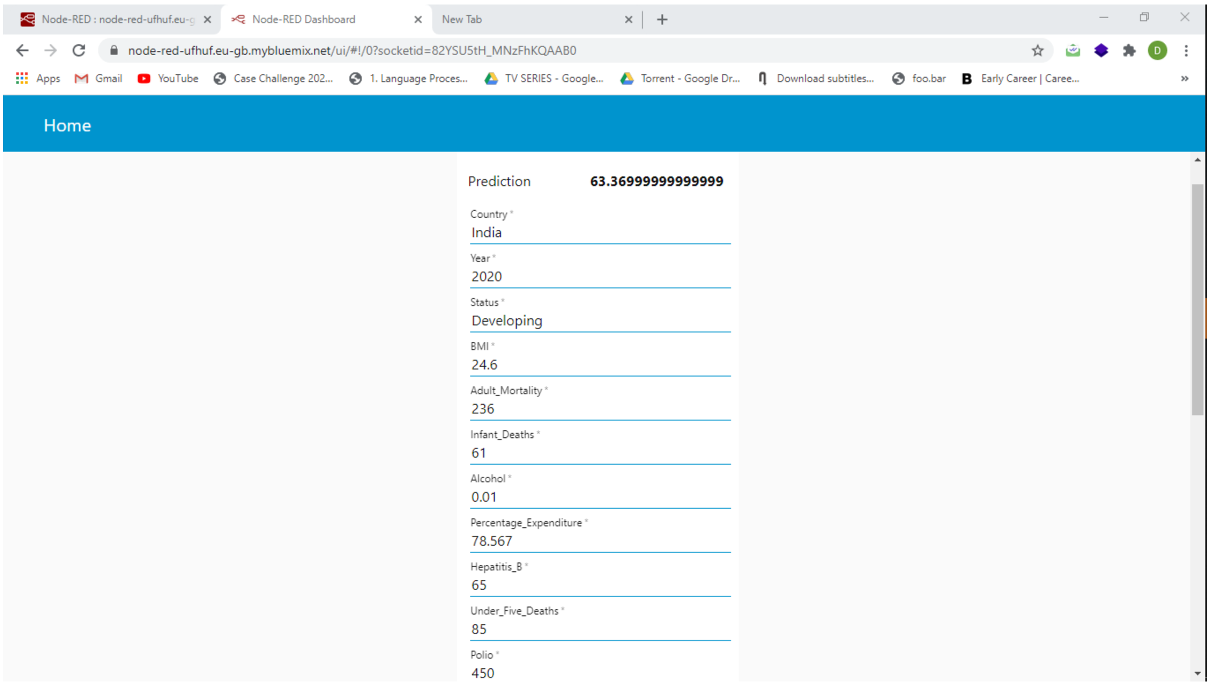
5 FLOWCHART

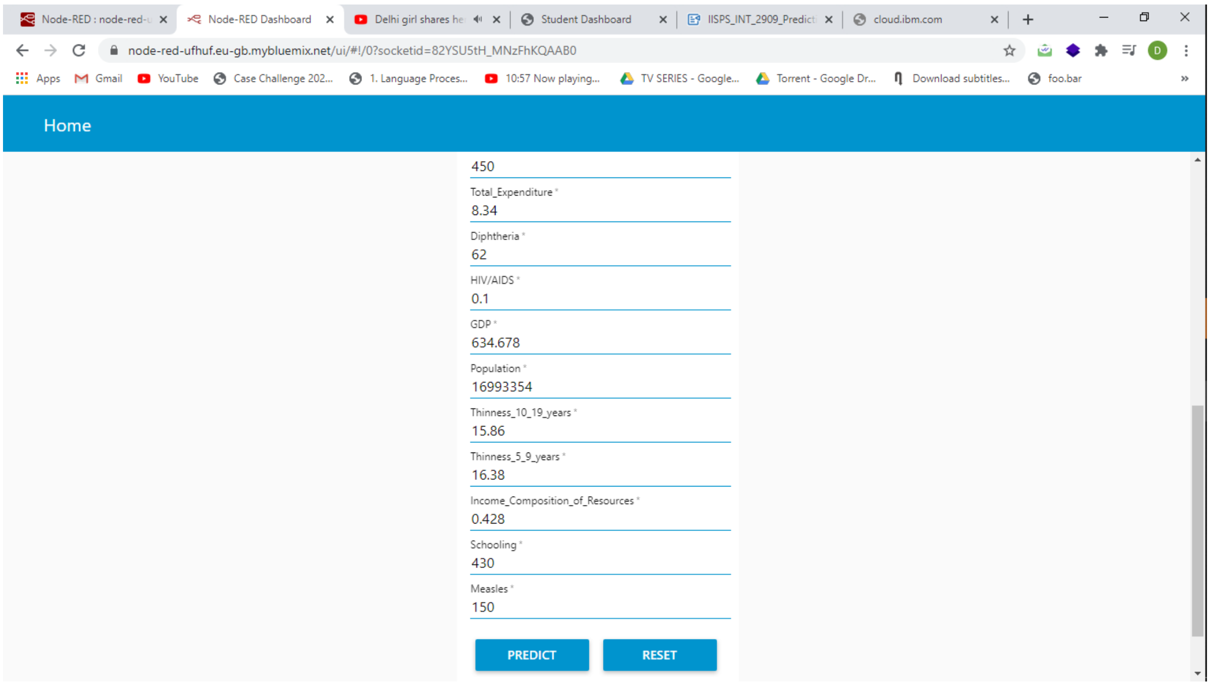
The following figure depicts the workflow that was adopted while the building of this project.



6 RESULT

The project is able to predict the average life expectancy in a country after taking into account various factors with the maximum accuracy.





7 ADVANTAGES & DISADVANTAGES

Advantages

- Can be accessed from anywhere as it is a WebApp.
- Easy to use.
- User freindly UI.

- Gives Real time response.

Disadvantages

The results should not be interpreted as definitive as the actual life expectancy of an individual may vary depending on his lifestyle. The results are based on statistical regression.

8 APPLICATIONS

The proposed project could be applied to the following sectors:

- **Personal Healthcare:** People can use the developed WebApp to predict their life expectancy by inputting values for different parameters. This would make people more aware of their general health and motivate them to adopt a healthy lifestyle.
- **Mediacl Sector:** Health care sector can use the WebApp to analuse the life expectancy of people and the various factors that affect it nd hence can fund and provide better services to those with greater needs**.**
- **Insurance Sector:** Life insurance companies can use the WebApp to analyse life expectancy of people and hence provide them personalised services.

9 CONCLUSION

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 12 Advance Care Planning. This breakthrough can widely impact health sectors and economic sectors by improving the resources, funds and services provided to the common people. It can also increase the ease of access to the individuals.

10 FUTURE SCOPE

Future Scope of the Model can be:

1. **Attractive UI-** It is a simple webpage only asking inputs and predict output. In future I have decided to make it more user friendly by providing some useful information about the country in the webpage itself so that user does not need to do any kind of prior research for the values.

2. **Feature Reduction-** It requires much more data about 21 columns to be known prior for predicting life expectancy which can be again difficult for a normal user to gather such data so we can do some kind of feature reduction or replacement of some features as individuals or groups to make it more user friendly.

11 BIBILOGRAPHY

- https://cloud.ibm.com/docs/overview?topic=overview-whatis-platform
- https://developer.ibm.com/tutorials/how-to-create-a-node-red-starter-application/
- https://nodered.org/
- https://github.com/watson-developer-cloud/node-red-labs
- https://www.youtube.com/embed/r7E1TJ1HtM0
- https://www.kaggle.com/kumarajarshi/life-expectancy-who
- https://www.youtube.com/watch?v=DBRGlAHdj48&list=PLzpeuWUENMK2PYtasCaKK4bZjaYzhW23
- https://www.youtube.com/watch?v=CUi8GezG1I&list=PLzpeuWUENMK2PYtasCaKK4bZjaYzhW23L&index=2
- https://www.youtube.com/watch?v=Jtej3Y6uUng
- https://machinelearningmastery.com/columntransformer-for-numerical-and-categorical-data/

APPENDIX

Source Code:

Jupyter Notebook Code:

```
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 Pipeline,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



import types
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_d6f92bb2b7844480bc18e54d9ef34687 = ibm_boto3.client(service_name='s3',
    ibm_api_key_id=API key comes here,
    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_d6f92bb2b7844480bc18e54d9ef34687.get_object(Bucket='predictinglifeexpectancy-donotdelete-pr-bong1t0kv4wc9d',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 )

dataset = pd.read_csv(body)
dataset.head()

dataset.shape

dataset.describe()

dataset.info()

#counting null entries in each column
dataset.isnull().sum()

country_list = dataset.Country.unique()
len(country_list)

country_list = dataset.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 the country of that record using interpolate()
for country in country_list:
    dataset.loc[dataset['Country'] == country,fill_list] = dataset.loc[dataset['Country'] ==
country,fill_list].interpolate()
dataset.dropna(inplace=True)

dataset.shape  #(1987, 22) size reduced

#Corelation matrix

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

#Renaming columns to remove the trailing spaces

dataset.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
#ignoring non-numeric features

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 boxplot

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()

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))

#Removing Outliers

from scipy.stats.mstats import winsorize

winsorized_Life_Expectancy = winsorize(dataset['Life expectancy'],(0.01,0))
winsorized_Adult_Mortality = winsorize(dataset['Adult Mortality'],(0,0.03))
winsorized_Infant_Deaths = winsorize(dataset['infant deaths'],(0,0.10))
winsorized_Alcohol = winsorize(dataset['Alcohol'],(0,0.01))
winsorized_Percentage_Exp = winsorize(dataset['percentage expenditure'],(0,0.12))
winsorized_HepatitisB = winsorize(dataset['Hepatitis B'],(0.11,0))
winsorized_Measles = winsorize(dataset['Measles'],(0,0.19))
winsorized_Under_Five_Deaths = winsorize(dataset['under-five deaths'],(0,0.12))
winsorized_Polio = winsorize(dataset['Polio'],(0.09,0))
winsorized_Tot_Exp = winsorize(dataset['Total expenditure'],(0,0.01))
winsorized_Diphtheria = winsorize(dataset['Diphtheria'],(0.10,0))
winsorized_HIV = winsorize(dataset['HIV/AIDS'],(0,0.16))
winsorized_GDP = winsorize(dataset['GDP'],(0,0.13))
winsorized_Population = winsorize(dataset['Population'],(0,0.14))
winsorized_thinness_10_19_years = winsorize(dataset['thinness 10-19 years'],(0,0.04))
winsorized_thinness_5_9_years = winsorize(dataset['thinness 5-9 years'],(0,0.04))
winsorized_Income_Comp_Of_Resources = winsorize(dataset['Income composition of
resources'],(0.05,0))
winsorized_Schooling = winsorize(dataset['Schooling'],(0.02,0.01))


winsorized_list =
[winsorized_Life_Expectancy,winsorized_Adult_Mortality,winsorized_Alcohol,winsorized_Measles,wins
orized_Infant_Deaths,
        winsorized_Percentage_Exp,winsorized_HepatitisB,winsorized_Under_Five_Deaths,winsorized_P
olio,winsorized_Tot_Exp,winsorized_Diphtheria,
```

```
        winsorized_HIV,winsorized_GDP,winsorized_Population,winsorized_thinness_10_19_years,winso
rized_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 without outliers to the dataset

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

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

Expolartory Data Analysis

```
dataset.columns

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

disease_cols=dataset[['Life expectancy','Alcohol','Hepatitis
B','Measles','BMI','Polio','Diphtheria','HIV/AIDS','Adult Mortality',
            'infant deaths','under-five deaths','thinness 10-19 years','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')
```

```
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','wins
orized_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(dataset[col[i]])
    plt.title(col[i])

plt.show()

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

plt.figure(figsize=(6,6))
plt.bar(dataset.groupby('Status')['Status'].count().index,dataset.groupby('Status')['winsorized_Life_Expecta
ncy'].mean())
plt.ylabel("Avg Life_Expectancy")
plt.title("Life_Expectancy w.r.t Status")
plt.show()

country_data =
dataset.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.bar(dataset.groupby('Year')['Year'].count().index,dataset.groupby('Year')['winsorized_Life_Expectancy'
].mean())
plt.xlabel("Year",fontsize=12)
plt.ylabel("Avg Life_Expectancy",fontsize=12)
plt.show()

cor_matrix=dataset.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','w
insorized_thinness_5_9_years']
cormat=dataset[attributes].corr()
plt.figure(figsize=(15,15))
sns.heatmap(cormat, square=True, annot=True, linewidths=.5)
plt.show()

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

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

dataset.columns
```

Creating new dataframe with refined data

```
new_dataset=pd.DataFrame(data=dataset,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_dataset.shape

new_dataset.head()

#Renaming the columns

new_dataset.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_dataset.head()

new_dataset.columns

#dividing dataset into features and label

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

X.head()

Y.head()

#Splitting into training and testing data set with 80% fro training and 20% for testing

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

#Creating 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']

categorical_transformer = Pipeline(steps=[
    ('onehot', OneHotEncoder(handle_unknown='ignore')),
])

numeric_transformer = Pipeline(steps=[
    ('imputer', SimpleImputer(strategy='median'))

])

preprocessor = ColumnTransformer(
    transformers=[
        ('cat', categorical_transformer, categorical_features),
        ('num', numeric_transformer, numeric_features)
    ]
)

#Finding best algo

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
```

Random Forest Regressor is the best algo for us

RandomForest Regressor

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

RFRegressor.fit(X_train,Y_train)

predict= RFRegressor.predict(X_test)

r2_score(predict, Y_test)

# @hidden_cell

wml_credentials={
#Credentials go here }

from watson_machine_learning_client import WatsonMachineLearningAPIClient

client = WatsonMachineLearningAPIClient( wml_credentials )
```

```
model_props = {client.repository.ModelMetaNames.AUTHOR_NAME: "divyanshi agarwal",
              client.repository.ModelMetaNames.AUTHOR_EMAIL: "dagarwal1_be17@thapar.edu",
              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
```

Node RED Flow Code:

[
    {
        "id":"3ede1df2.1f27f2",
        "type":"tab",
        "label":"Life Expectancy",
        "disabled":false,
        "info":""
    },

    {
        "id":"b9540a3.318b5f8",
        "type":"ui_form",
        "z":"3ede1df2.1f27f2",
        "name":"",
        "label":"",
        "group":"aec46c50.97f1c",
        "order":0,
        "width":0,
        "height":0,
        "options":[
            {"label":"Country",
            "value":"Country",
            "type":"text",

"required":true,

"rows":null},

{"label":"Year",

"value":"Year",

"type":"number",

"required":true,

"rows":null},

{"label":"Status",

"value":"Status",

"type":"text",

"required":true,

"rows":null},

{"label":"BMI",

"value":"BMI",

"type":"number",

"required":true,

"rows":null},

{"label":"Adult_Mortality",

"value":"Adult_Mortality",

"type":"number",

"required":true,

"rows":null},

{"label":"Infant_Deaths",

"value":"Infant_Deaths",

"type":"number",

"required":true,

"rows":null},

{"label":"Alcohol",

"value":"Alcohol",

"type":"number",

"required":true,

"rows":null},

{"label":"Percentage_Expenditure",

"value":"Percentage_Expenditure",

"type":"number",

"required":true,

"rows":null},

{"label":"Hepatitis_B",

"value":"Hepatitis_B",

"type":"number",

"required":true,

"rows":null},

{"label":"Under_Five_Deaths",

"value":"Under_Five_Deaths",

"type":"number",

"required":true,

"rows":null},

{"label":"Polio",

"value":"Polio",

"type":"number",

"required":true,
"rows":null},

{"label":"Total_Expenditure",
"value":"Total_Expenditure",
"type":"number",
"required":true,
"rows":null},

{"label":"Diphtheria",
"value":"Diphtheria",
"type":"number",
"required":true,
"rows":null},

{"label":"HIV/AIDS",
"value":"HIVAIDS",
"type":"number",
"required":true,
"rows":null},

{"label":"GDP",
"value":"GDP",
"type":"number",
"required":true,
"rows":null},

{"label":"Population",
"value":"Population",
"type":"number",

"required":true,

"rows":null},

{"label":"Thinness_10_19_years",

"value":"Thinness_10_19_years",

"type":"number",

"required":true,

"rows":null},

{"label":"Thinness_5_9_years",

"value":"Thinness_5_9_years",

"type":"number",

"required":true,

"rows":null},

{"label":"Income_Composition_of_Resources",

"value":"Income_Composition_of_Resources",

"type":"number",

"required":true,

"rows":null},

{"label":"Schooling",

"value":"Schooling",

"type":"number",

"required":true,

"rows":null},

{"label":"Measles",

"value":"Measles",

"type":"number",

"required":true,

"rows":null}],

"formValue":{"Country":"","Year":"","Status":"","BMI":"","Adult_Mortality":"",

"Infant_Deaths":"","Alcohol":"","Percentage_Expenditure":"","Hepatitis_B":"",

"Under_Five_Deaths":"","Polio":"","Total_Expenditure":"","Diphtheria":"",

"HIVAIDS":"","GDP":"","Population":"","Thinness_10_19_years":"",

"Thinness_5_9_years":"","Income_Composition_of_Resources":"","Schooling":"",

"Measles":""},

"payload":"",

"submit":"Predict",

"cancel":"Reset",

"topic":"",

"x":90,

"y":80,

"wires":[["d5913867.baf458"]]

},

{"id":"d5913867.baf458",

"type":"function",

"z":"3ede1df2.1f27f2",

"name":"PreToken",

"func":"//make user give values as global variables\nglobal.set(\"Country\",msg.payload.Country);\nglobal.set(\"Year\",msg.payload.Year);\nglobal.set(\"Status\",msg.payload.Status);\nglobal.set(\"BMI\",msg.payload.BMI);\nglobal.set(\"Adult_Mortality\",msg.payload.Adult_Mortality);\nglobal.set(\"Infant_Deaths\",msg.payload.Infant_Deaths);\nglobal.set(\"Alcohol\",msg.payload.Alcohol);\nglobal.set(\"Percentage_Expenditure\",msg.payload.Percentage_Expenditure);\nglobal.set(\"Hepatitis_B\",msg.payload.Hepatitis_B);\nglobal.set(\"Under_Five_Deaths\",msg.payload.Under_Five_Deaths);\nglobal.set(\"Polio\",msg.payload.Polio);\nglobal.set(\"Total_Expenditure\",msg.payload.Total_Expenditure);\nglobal.set(\"Diphtheria\",msg.payload.Diphtheria);\nglobal.set(\"HIVAIDS

\",msg.payload.HIVAIDS);\nglobal.set(\"GDP\",msg.payload.GDP);\nglobal.set(\"Population \",msg.payload.Population);\nglobal.set(\"Thinness_10_19_years\",msg.payload.Thinness_10 _19_years);\nglobal.set(\"Thinness_5_9_years\",msg.payload.Thinness_5_9_years);\nglobal. set(\"Income_Composition_of_Resources\",msg.payload.Income_Composition_of_Resource s);\nglobal.set(\"Schooling\",msg.payload.Schooling);\nglobal.set(\"Measles\",msg.payload. Measles);\n\n//For receiving the token\nvar apikey=\"API key comes here\";\nmsg.headers={\"content-type\":\"application/x-www-form-urlencoded\"};\nmsg.payload={\"grant_type\":\"urn:ibm:params:oauth:grant-type:apikey\",\"apikey\":apikey};\nreturn msg;",

"outputs":1,

"noerr":0,

"x":280,

"y":80,

"wires":[["50518351.96a96c"]]},

{"id":"50518351.96a96c",

"type":"http request",

"z":"3ede1df2.1f27f2",

"name":"",

"method":"POST",

"ret":"obj",

"paytoqs":false,

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

"tls":"",

"persist":false,

"proxy":"",

"authType":"",

"x":490,

"y":80,

"wires":[["dc746923.89cd68"]]},

{"id":"dc746923.89cd68",

"type":"function",

"z":"3ede1df2.1f27f2",

"name":"sendToEndPoint",

"func":"var token=msg.payload.access_token;\nvar instance_id=\"End point id\";\nmsg.headers={'Content-Type': 'application/json',\"Authorization\":\"Bearer \"+token,\"ML-Instance-ID\":instance_id};\n\n//\nvar Country = global.get('Country');\nvar Year = global.get('Year');\nvar Status = global.get('Status');\nvar BMI = global.get('BMI');\nvar Adult_Mortality = global.get('Adult_Mortality');\nvar Infant_Deaths = global.get('Infant_Deaths');\nvar Alcohol = global.get('Alcohol');\nvar Percentage_Expenditure = global.get('Percentage_Expenditure');\nvar Hepatitis_B = global.get('Hepatitis_B');\nvar Under_Five_Deaths = global.get('Under_Five_Deaths');\nvar Polio = global.get('Polio');\nvar Total_Expenditure = global.get('Total_Expenditure');\nvar Diphtheria = global.get('Diphtheria');\nvar HIVAIDS = global.get('HIV/AIDS');\nvar GDP = global.get('GDP');\nvar Population = global.get('Population');\nvar Thinness_10_19_years = global.get('Thinness_10_19_years');\nvar Thinness_5_9_years = global.get('Thinness_5_9_years');\nvar Income_Composition_of_Resources = global.get('Income_Composition_of_Resources');\nvar Schooling = global.get('Schooling');\nvar Measles = global.get('Measles');\n\n//send user values to service endpoints\nmsg.payload={\n   \"fields\":['Country', 'Year', 'Status', 'BMI', 'Adult_Mortality', 'Infant_Deaths',\n   'Alcohol', 'Percentage_Expenditure', 'Hepatitis_B', 'Under_Five_Deaths',\n   'Polio', 'Total_Expenditure', 'Diphtheria', 'HIV/AIDS', 'GDP',\n 'Population', 'Thinness_10_19_years', 'Thinness_5_9_years',\n 'Income_Composition_of_Resources', 'Schooling', 'Measles'],\n   \"values\":[[Country, Year, Status, BMI, Adult_Mortality, Infant_Deaths,\n   Alcohol, Percentage_Expenditure, Hepatitis_B, Under_Five_Deaths,\n   Polio, Total_Expenditure, Diphtheria, HIVAIDS, GDP,\n   Population, Thinness_10_19_years, Thinness_5_9_years,\n Income_Composition_of_Resources, Schooling, Measles]]\n}\nreturn msg;",

"outputs":1,

"noerr":0,

"x":720,

"y":80,

"wires":[["9fbd786c.f4c448"]]},

{"id":"9fbd786c.f4c448",

"type":"http request",

"z":"3ede1df2.1f27f2",

"name":"",

"method":"POST",

"ret":"obj",

"paytoqs":false,

"url":"deployment URL",

"tls":"",

"persist":false,

"proxy":"",

"authType":"basic",

"x":130,

"y":260,

"wires":[["6fc5d8fd.69e108","9f653359.a46e3"]]},

{"id":"6fc5d8fd.69e108",

"type":"function",

"z":"3ede1df2.1f27f2",

"name":"getFromEndPoint",

"func":"//actually getting our predicted values\nmsg.payload=msg.payload.values[0][0];\nreturn msg;",

"outputs":1,

"noerr":0,

"x":450,

"y":260,

"wires":[["a8ffb2b0.8a0b3","e19bb7c1.805d58"]]},

{"id":"a8ffb2b0.8a0b3",

"type":"ui_text",

"z":"3ede1df2.1f27f2",

"group":"aec46c50.97f1c",

"order":1,

"width":0,

"height":0,

"name":"",

"label":"Prediction",

"format":"{{msg.payload}}",

"layout":"row-spread",

"x":740,

"y":260,

"wires":[]},

{"id":"9f653359.a46e3",

"type":"debug",

"z":"3ede1df2.1f27f2",

"name":"",

"active":true,

"tosidebar":true,

"console":false,

"tostatus":false,

"complete":"payload",

"targetType":"msg",

"x":170,

"y":340,

"wires":[]},

{"id":"e19bb7c1.805d58",

"type":"debug",

"z":"3ede1df2.1f27f2",

"name":"",

"active":true,

"tosidebar":true,

"console":false,

"tostatus":false,

"complete":"false",

"x":490,

"y":340,

"wires":[]},

{"id":"aec46c50.97f1c",

"type":"ui_group",

"z":"",

"name":"Life Expectancy",

"tab":"be3db9b1.9af108",

"order":1,

"disp":true,

"width":"6",

"collapse":false},

{"id":"be3db9b1.9af108",

"type":"ui_tab",

"z":"",

"name":"Home",

"icon":"dashboard",

"disabled":false,

"hidden":false}

]