**Project Report**

**Predicting Life Expectancy Using Machine Learning**

**Name:** Urvi Shukla

**Email** : [richashukla0920@gmail.com](mailto:richashukla0920@gmail.com)

**Project ID:** SPS\_PRO\_215

**Internship Title :** Predicting Life Expectancy using Machine Learning - SB49438

**Internship Under :** Smartbridge Educational Services PVT LTD.

**GitHub:** <https://github.com/SmartPracticeschool/llSPS-INT-2614-Predicting-Life-Expectancy-using-Machine-Learning>

**Notebook Link:**

<https://eu-gb.dataplatform.cloud.ibm.com/analytics/notebooks/v2/2c8865ee-da1f-442e-bdcf-05a31f88ae7f/view?projectid=b94ee4ad-8ac5-4888-ac0a-cf04b1c1b2fb&context=wdp>

**Node Red Link:**

<https://node-red-urvi.eu-gb.mybluemix.net/red/#flow/a52ce7ee.171718>

|  |  |
| --- | --- |
| 1 | Introduction |
|  | 1.1 Overview |
|  | 1.2 purpose |
| 2 | Literature Survey |
|  | 2.1 Existing Problem |
|  | 2.2 Proposed Solution |
| 3 | Theoretical Background |
|  | 3.1 Block diagram |
|  | 3.2 Hardware/Software Design |
| 4 | Experimental Investigation |
| 5 | Flowchart |
| 6 | Result |
| 7 | Advantages/Disadvantages |
| 8 | Applications |
| 9 | Solution |
| 10 | Future Scope |
| 11 | Bibliography |
|  | Appendix |
|  | 1. Source code |

**Predicting Life Expectancy Using Machine Learning**

**1) Introduction:**

**1.1) Overview:**

Life Expectancy refers to the amount of years a human is expected to be alive based upon the statistical calculations. Life Expectancy depends on various factors like health status, age,geographical area,Education , Mental illness , Sex difference, Ecomonic background, etc.

The end result will be a web page where it will predict the life expectancy based upon various inputs provided by the user by the technique of regression algorithm.

The dataset used for the prediction contains data from year 2000 to 2015 for 193 countries. It contains more than 2500 entries and around 22 columns with various features such as Population, Alcohol Consumption, Infant Mortality Rate etc., which aids the prediction of the model.

**1.2) Purpose:**

Life expectancy is a statistical measure of on average the human will live based upon the factor such as health status, age,geographical area,Education , Mental illness , Sex difference, Ecomonic background, etc.

Life expectancy of humans in a particular region defines the lifestyle of that region . If the condition of the region and economic stability is good enough which clearly shows that the life range of the people in particular region will be high.

By predicting life expectancy and having good prognostication can help in making valuable decisions like the course of treatment and helps to anticipate the procurement of health care services and facilities.

**2) Literature Survey:**

**2.1) Existing Problem:**

The Regression Machine learning project predicts the life span of the user based upon the leverages historical data provided by the users. Life expectancy is a statistical measure of on average the human will live based upon the factor such as health status, age,geographical area,Education , Mental illness , Sex difference, Ecomonic background, etc.Life expectancy of humans in a particular region defines the lifestyle of that region . If the condition of the region and economic stability is good enough which clearly shows that the life range of the people in particular region will be high.

By predicting life expectancy and having good prognostication can help in making valuable decisions like the course of treatment and helps to anticipate the procurement of health care services and facilities.Predicting 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.

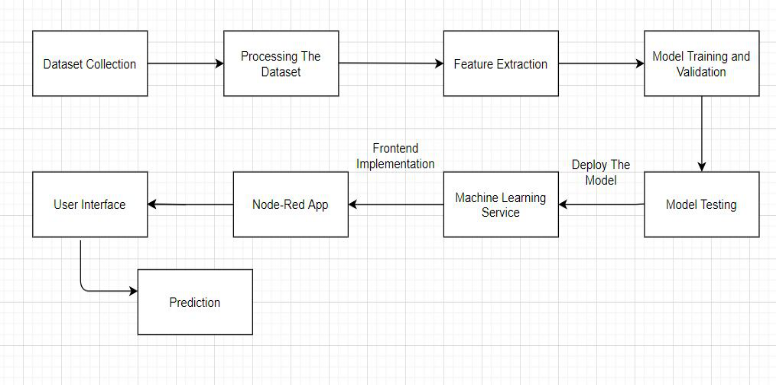
2.2) Proposed Solution:

The end result will be a web page where it will predict the life expectancy based upon various inputs provided by the user by the technique of regression algorithm. It depend on factors such as Status of the country, Adult Mortality Rate, Infant deaths, Alcohol, Hepatitis B, Measles, BMI, Polio, Total Expenditure, Diphtheria, HIV/AIDS, GDP of a country, Population, Income Composition of Resources, Schooling status of the country.

To access the trained model, we will use Node-Red App from IBM Cloud.

**3) Theoretical Analysis:**

**3.1) Block Diagram:**



**3.2) Hardware/Software designing:**

1. Collecting the Dataset

2. Creating Necessary IBM Cloud Service

3. Creating and Configuring Watson Studio

4. Create Machine Learning Service

5. Adding Jupyter Notebook

6. Build ML model and create Scoring Endpoint for Node-Red Integration

7. Build Node-Red Flow and integrate ML services and deploy.

**4) Experimental Investigations:**

This Project aims to predict Life Expectancy of a human in any Country. Whole Project is based on the dataset accuracy. Thus, the data set has been taken from WHO, which was provided publicly.

The 21 factors which are taken into account for predicting the life expectancy of a country are as follows:

**1. Country**

**2. Year**

**3. Status**: Developed or Developing status of the country.

**4. Adult mortality**: Adult Mortality Rates of both sexes (probability of dying between 15 and 60 years per 1000 population).

**5. Infant deaths**: Number of Infant Deaths per 1000 population.

**6. Alcohol**: Alcohol, recorded per capita (15+) consumption.

**7. Percentage Expenditure**: Expenditure on health as a percentage of Gross Domestic Product per capita (%).

**8. Hepatitis B**: Immunization coverage among 1-year-olds (%).

**9. Measles**: Number of reported cases per 1000 population.

**10. BMI**: Average Body Mass Index of entire population.

**11. Under-five deaths**: Number of under-five deaths per 1000 population.

**12. Polio**: Immunization coverage among 1-year-olds (%).

**13. Total expenditure**: General government expenditure on health as a percentage of total government expenditure (%).

**14. Diphtheria**: Diphtheria tetanus toxoid and pertussis (DTP3) immunization coverage among 1-year olds (%).

**15. HIV/AIDS**: Deaths per 1 000 live births HIV/AIDS (0-4 years).

**16. GDP**: Gross Domestic Product per capita (in USD).

**17. Population**: Population of the country.

**18. Thinness 10-19 years**: Prevalence of thinness among children and adolescents for Age 10 to 19 (%).

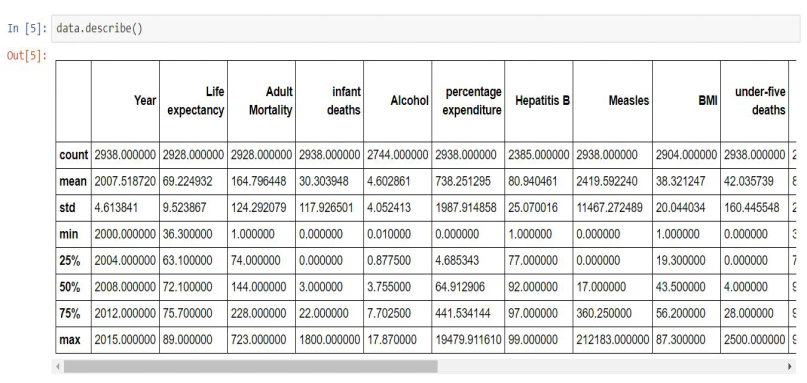
**19. Thinness 5-9 years**: Prevalence of thinness among children for Age 5 to 9 (%).

**20. Income composition of resources**: Human Development Index in terms of income composition of resources (index ranging from 0 to 1).

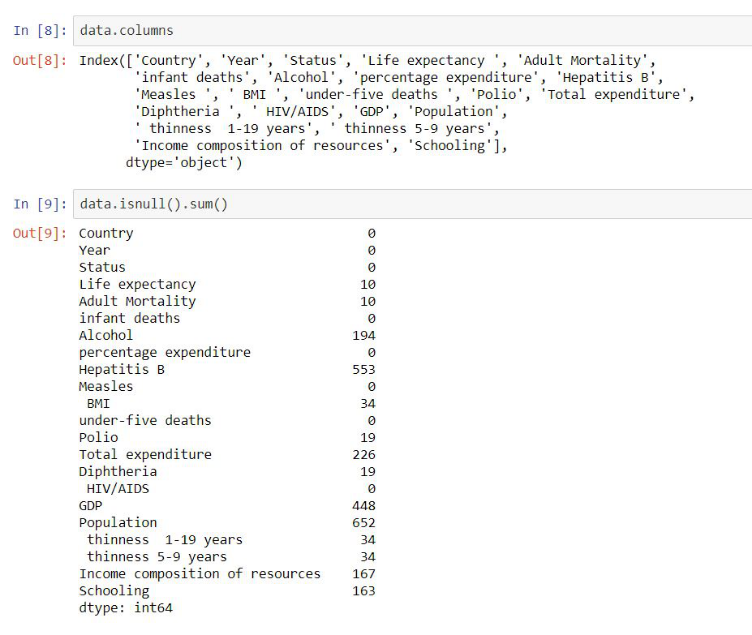
**21. Schooling**: Number of years of schooling

Algorithm Used:- Random Forest Regression

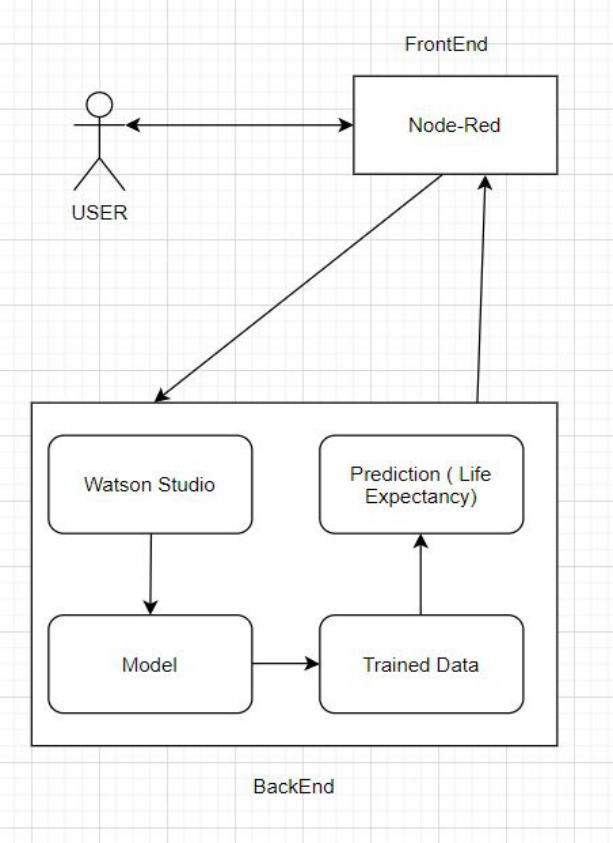
**Analysing the features:**



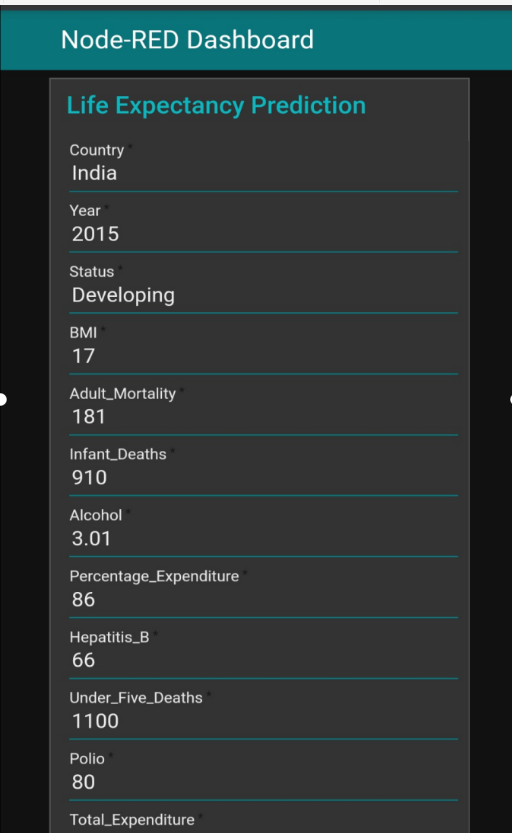


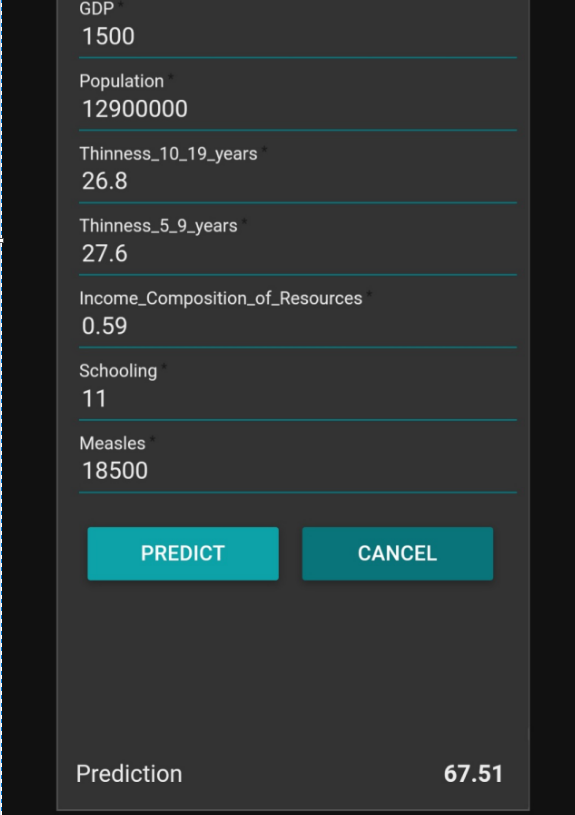


**5) Flowchart:**



**6)Result:**





**7) Advantages/Disadvantages:**

Advantages:

* Easy to build and deploy
* Model efficiency is high and get improves over time because we have deployed it in machine learning
* Space compatible
* Easy for user to interact with the model via UI

Disadvantages:

* Computation cost and training the model will increase the cost if dataset is too large in size
* Node-Red doesn’t give much flexibility to design own templates, although it’s a great service.
* Required Internet connection
* Wrong prediction if the user provides wrong information as it is user dependent.

**8) Applications:**

* This will help in suggesting a country which area should be given importance in order to efficiently improve the life expectancy of its population.
* To help the government prepare life insurance policies for people. This will benefit the people.
* To analyse all the factors and plan out measures to increase the life expectancy of the country

**9) Conclusion:**

In this Project, we developed a Machine Learning Model to predict Life Expectancy of humans in a country. Predicting Life Expectancy can lead to the development of the country. It can widely impact Health Sectors, Public Sectors and Economic Sectors by improving the resources, funds and services provided to people.

**10) Future Scope:**

1) Other factors such as sentiment analysis and mental health can be added to predict life expectancy.

2)Happiness index is also one such feature which can be proved vital determining life expectancy.

**11) Bibliography:**

* Project Planning and Kick-off:

<https://www.youtube.com/watch?v=LOCkV-mENq8&feature=youtu.be>

<https://www.allbusinesstemplates.com/download/?filecode=2KBA4&lang=en&iuid=9f9faa69-9fab-40ee-8457-ea0e5df8c8de>

* Node-Red starter tutorial:

<https://developer.ibm.com/tutorials/how-to-create-a-node-red-starter-application/>

### **Introductory workshop for Watson Studio Cloud:** [**https://bookdown.org/caoying4work/watsonstudio-workshop/jn.html**](https://bookdown.org/caoying4work/watsonstudio-workshop/jn.html)

* AutoAI References:

<https://developer.ibm.com/tutorials/watson-studio-auto-ai/>

<https://www.youtube.com/watch?v=IDKCmC1fCiU>

* Dataset : From Kaggle

<https://www.kaggle.com/kumarajarshi/life-expectancy-who>

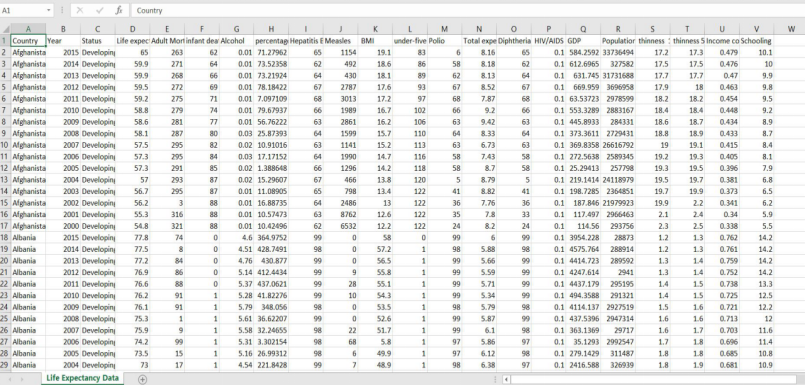
* Creating and Importing dataset in Jupyter Notebook:

<https://www.youtube.com/watch?v=Jtej3Y6uUng>

**Appendix**

1. **Source Code:**
2. **Data set**

Link: <https://www.kaggle.com/kumarajarshi/life-expectancy-who>



**1) Life Expectancy Notebook Code**:

# **Analysing 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

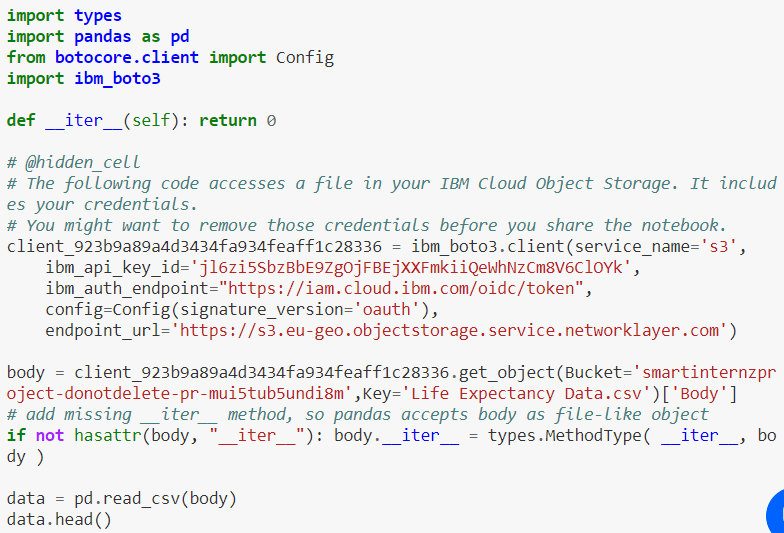
from sklearn.model\_selection import cross\_val\_score

from sklearn.linear\_model import LinearRegression

from sklearn.tree import DecisionTreeRegressor

from sklearn.metrics import r2\_score,mean\_squared\_error

**Reading the dataset in IBM Watson Studio**

****

data.head()

data.shape

data.describe()

data.info()

data.size

data.columns

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

data.isna().sum()

**Corelation matrix**

corrMatrix = data.corr()

corrMatrix.style.background\_gradient(cmap='plasma', low=.5, high=0).highlight\_null('red')

**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))

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

**Exploratory Data Analysis (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 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')

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')

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()

le\_country = data.groupby('Country')['winsorized\_Life\_Expectancy'].mean().sort\_values(ascending=True)

le\_country.plot(kind='bar', figsize=(50,15), fontsize=25)

plt.title("Life\_Expectancy w.r.t Country",fontsize=40)

plt.xlabel("Country",fontsize=35)

plt.ylabel("Avg Life\_Expectancy",fontsize=35)

plt.show()

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.title("Life\_Expectancy w.r.t Year")

plt.show()

cor\_matrix=data.corr()

print(cor\_matrix['winsorized\_Life\_Expectancy'].sort\_values(ascending=False))

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)

]

)

# **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**





**3)Node Red Flow:**





