PREDICTING LIFE EXPECTANCY USING MACHINE LEARNING

By:

Paritosh Purandar

CONTENTS:

1. INTRODUCTION:

1.1 Overview:

Life expectancy is the average number of years a person in a population could expect to live after age x. It is the life table parameter most commonly used to compare the survival experience of populations. The age most often selected to make comparisons is 0.0 (i.e., birth), although, for many substantive and policy analyses, other ages such as 65+ and 85+ are more relevant and may be used (e.g., for determining person-years of Medicare and Social Security benefit entitlement).

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

Life expectancy is perhaps the most important measure of health. Life expectancy increases due to healthcare improvements like the introduction of vaccines, the development of drugs or positive behavior changes like the reduction in smoking or drinking rates.

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.

2. LITERATURE SURVEY

2.1 Existing Solution:

As a result of the evolution of biotechnologies and related technologies such as the development of sophisticated medical equipment, humans are able to enjoy longer life expectancies than previously before. 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:

he project tries to build a model based on the given dataset. The first step was to clean the data, this included detecting and dealing with both missing values and outliers. The variables and dataset were given a general description so that a better understanding of what the variables mean could be gathered. Then both explicit and inexplicit missing values were detected. Inexplicit missing values were values that didn't make sense for a variable given the nature of the data.

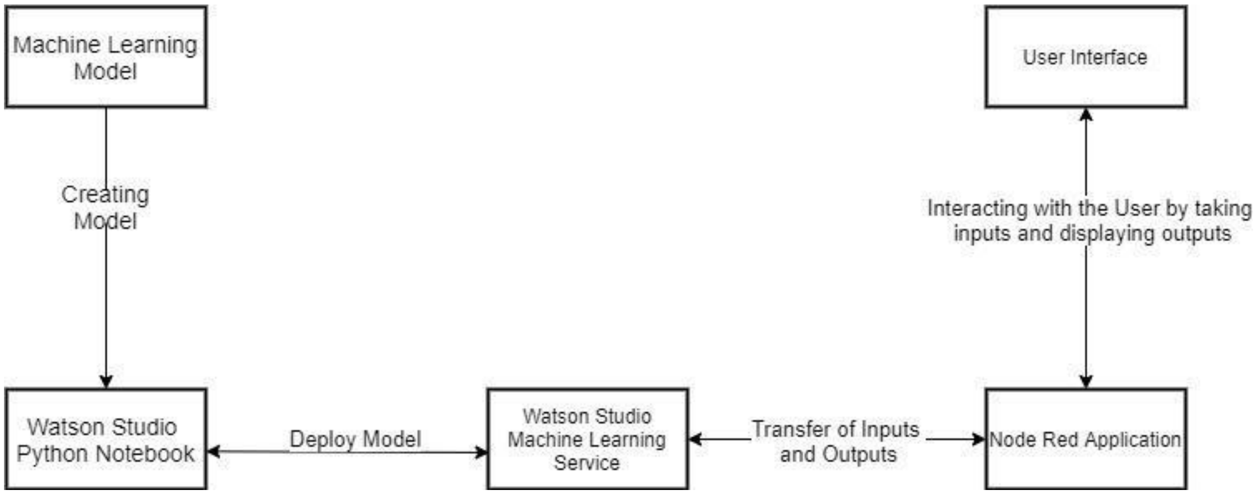
Our first step in this project is DATA PRE PROCESSING , is a crucial step that helps enhance the quality of data to promote the extraction of meaningful insights from the data. It refers to the technique of preparing the raw data to make it suitable for a building and training. We will drop required columns which will not be used in Regression. Analyzing data sets to summarize their main characteristics, often with visual methods is done. We build coefficient matrix and we obtain boxplots to analyze the outliers.

We train our regression models we will need to first split up our data into an A array that contains the features to train on, and a B array with the target variable, here it is

"Life Expectancy column". We split the data into a training set and a testing set. We will train out model on the training set and then use the test set to evaluate the model and the best model is chosen to evaluate the predictions.

3. THEORITICAL ANALYSIS

3.1 Block Diagram:


Machine Learning Model
Creating Model
Watson Studio Python Notebook
Deploy Model
Watson Studio Machine Learning Service
Transfer of Inputs and Outputs
Node Red Application
Interacting with the User by taking inputs and displaying outputs
User Interface

3.2 Hardware/Software Designing:

Project Requirements:

i) Functional Requirements:

To be able to predict the life expectancy accurately using Machine Learning models.

ii)Technical Requirements:

Any working laptop/PC with minimum 2.2Ghz processor and at least 8GB of memory with an Internet connection.

iii) Software Requirements:

a) Python

b) IBM Cloud

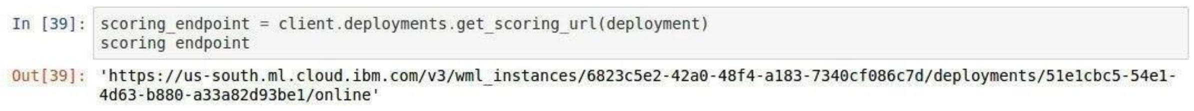
c) IBM Watson

Model Designing (Watson Studio) :

Steps: Open Watson studio => New Project => Create an empty Project => Give project name => Click Create => Add to Project => Notebook

This is the backend of our project. In order to connect our front end and backend we need to generate a scoring point.

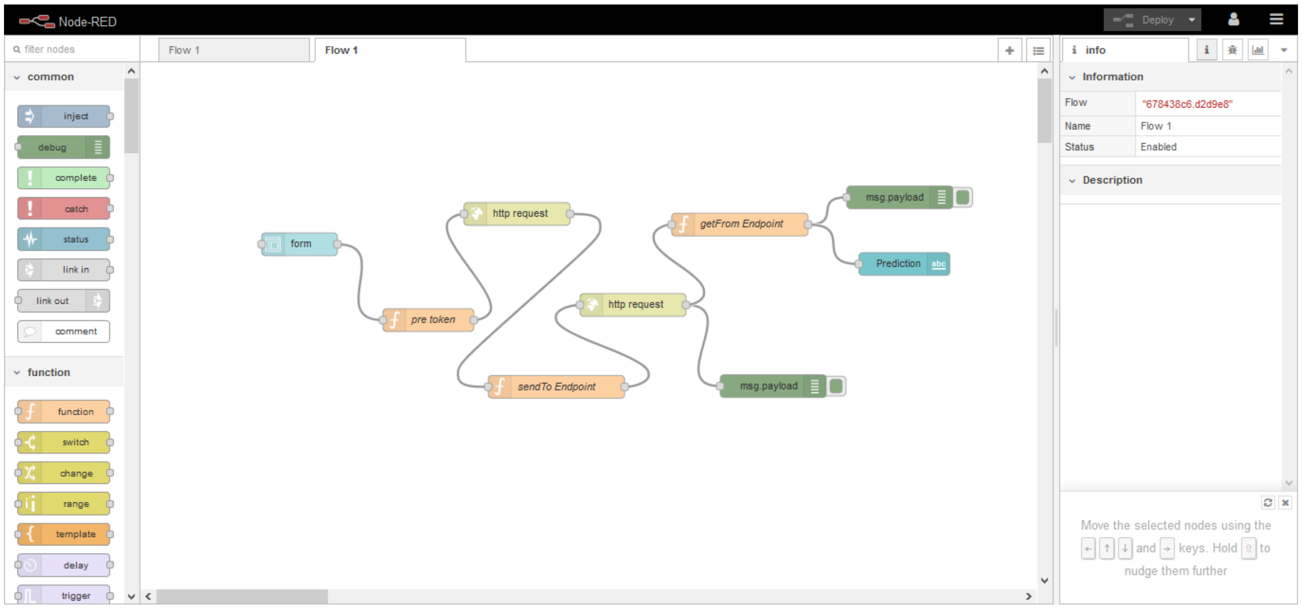
Scoring point Generation:



```
In [39]: scoring_endpoint = client.deployments.get_scoring_url(deployment)
         scoring endpoint

Out[39]: 'https://us-south.ml.cloud.ibm.com/v3/wml_instances/6823c5e2-42a0-48f4-a183-7340cf086c7d/deployments/51e1cbc5-54e1-
         4d63-b880-a33a82d93be1/online'
```

Node red integration with ML model:

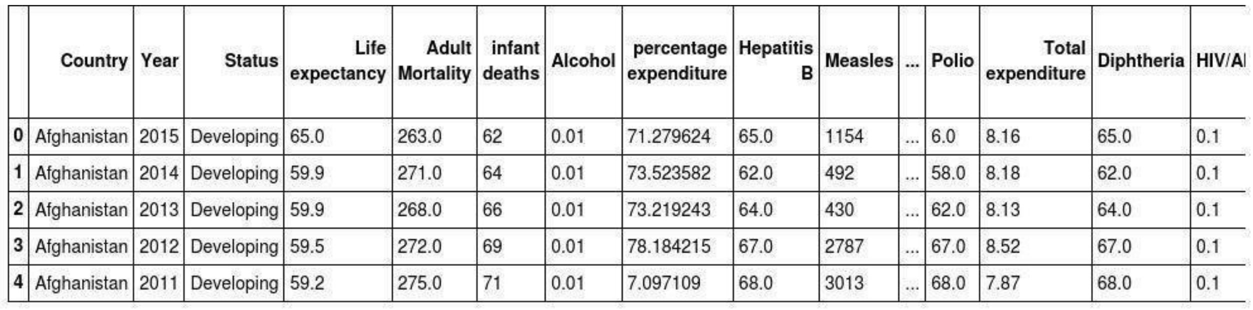


Flow	"678438c6.d2d9e8"
Name	Flow 1
Status	Enabled

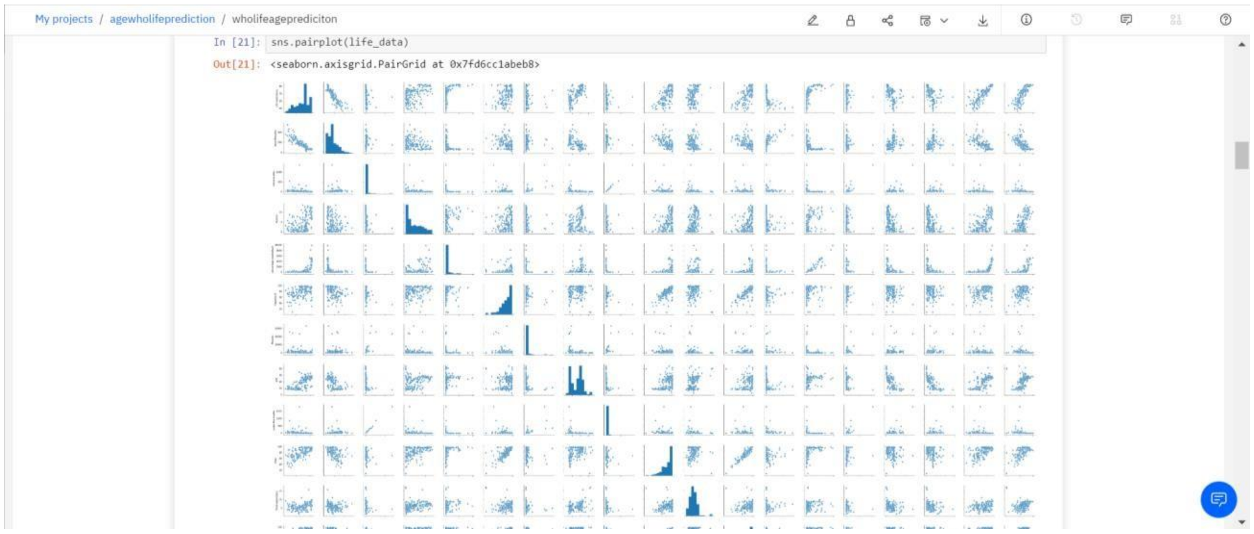
Move the selected nodes using the ← ↑ ↓ and → keys. Hold ⇧ to nudge them further

4. EXPERIMENTAL INVESTIGATIONS

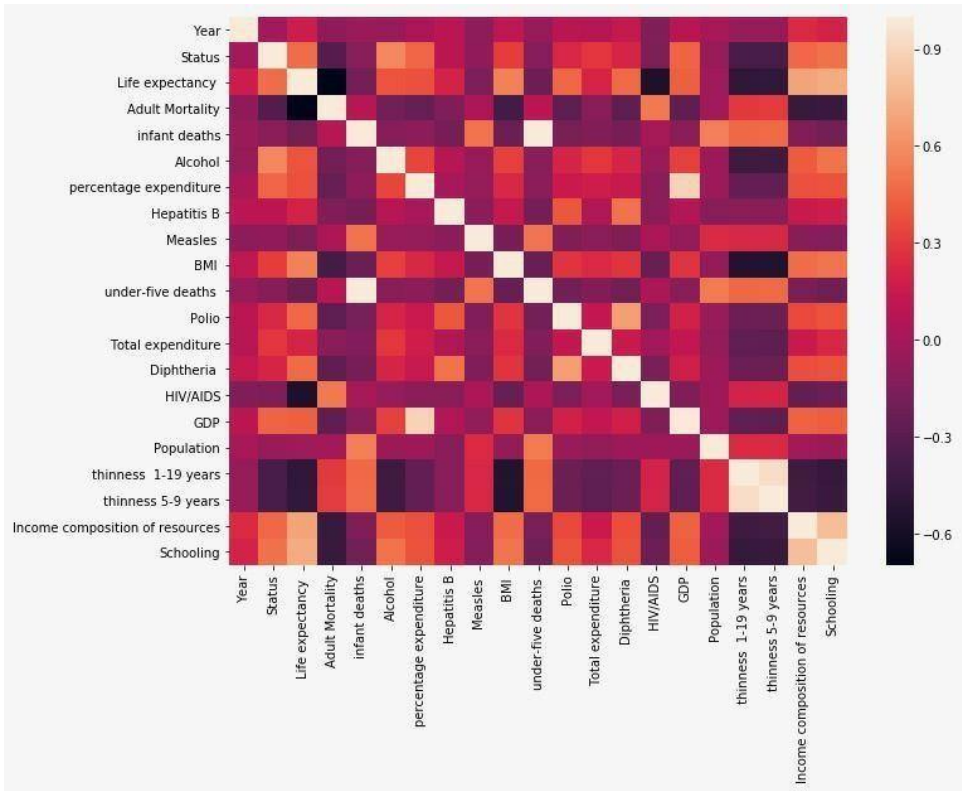
Analyzing every feature in our dataset is very important which helps us to build a model which gives more accurate result.



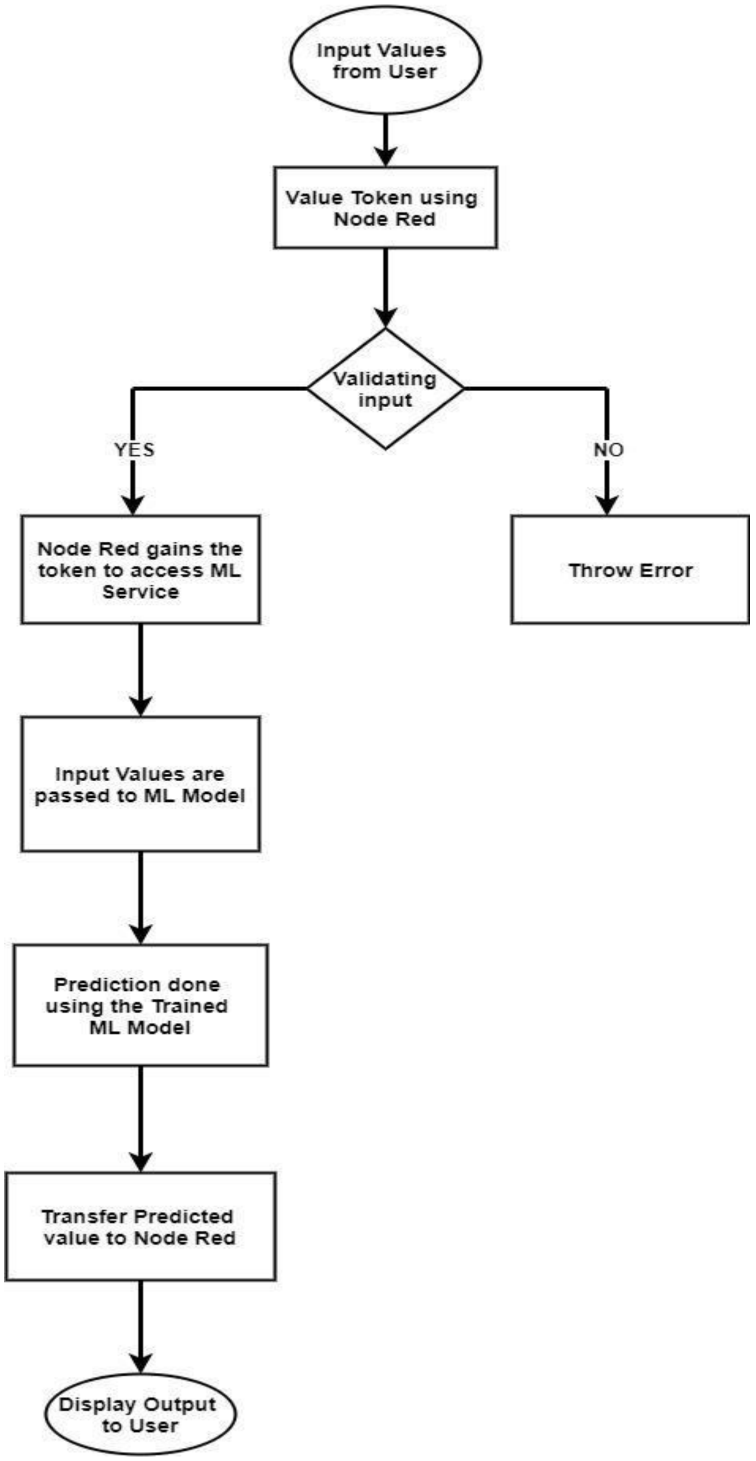
	Country	Year	Status	Life expectancy	Adult Mortality	infant deaths	Alcohol	percentage expenditure	Hepatitis B	Measles	...	Polio	Total expenditure	Diphtheria	HIV/AI
0	Afghanistan	2015	Developing	65.0	263.0	62	0.01	71.279624	65.0	1154	...	6.0	8.16	65.0	0.1
1	Afghanistan	2014	Developing	59.9	271.0	64	0.01	73.523582	62.0	492	...	58.0	8.18	62.0	0.1
2	Afghanistan	2013	Developing	59.9	268.0	66	0.01	73.219243	64.0	430	...	62.0	8.13	64.0	0.1
3	Afghanistan	2012	Developing	59.5	272.0	69	0.01	78.184215	67.0	2787	...	67.0	8.52	67.0	0.1
4	Afghanistan	2011	Developing	59.2	275.0	71	0.01	7.097109	68.0	3013	...	68.0	7.87	68.0	0.1



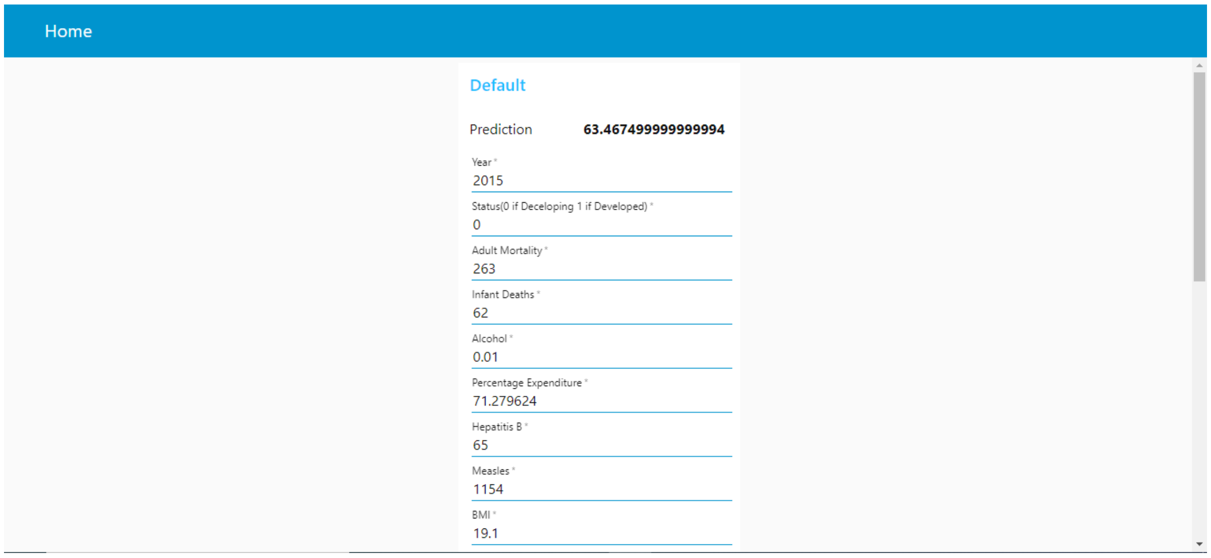
```
In [21]: sns.pairplot(life_data)
Out[21]: <seaborn.axisgrid.PairGrid at 0x7fd6cc1abeb8>
```



5. FLOW CHART


Input Values from User
Value Token using Node Red
Validating input
YES
NO
Node Red gains the token to access ML Service
Throw Error
Input Values are passed to ML Model
Prediction done using the Trained ML Model
Transfer Predicted value to Node Red
Display Output to User

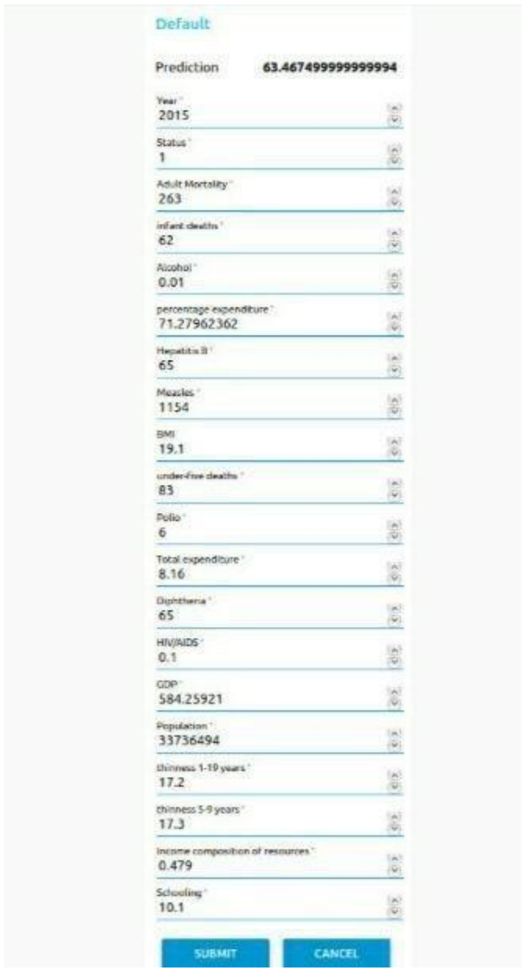
6. RESULT



Default

Prediction 63.467499999999994

Year *	2015
Status(0 if Deceloping 1 if Developed) *	0
Adult Mortality *	263
Infant Deaths *	62
Alcohol *	0.01
Percentage Expenditure *	71.279624
Hepatitis B *	65
Measles *	1154
BMI *	19.1



Prediction	63.467499999999994
Year *	2015
Status *	1
Adult Mortality *	263
infant deaths *	62
Alcohol *	0.01
percentage expenditure *	71.27962362
Hepatitis B *	65
Measles *	1154
BMI	19.1
under-five deaths *	83
Polio *	6
Total expenditure *	8.16
Diphtheria *	65
HIV/AIDS *	0.1
GDP *	584.25921
Population *	33736494
thinness 1-19 years *	17.2
thinness 5-9 years *	17.3
Income composition of resources *	0.479
Schooling *	10.1

7. ADVANTAGES AND DISADVANTAGES

7.1 Advantages:

Life expectancy can be estimated at any age, e.g. life expectancy at 65 years. Gives more weight to deaths at younger ages. Life expectancy has been used nationally to monitor health inequalities.

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.

7.2 Disadvantages:

This model is developed using Machine Learning in which human involvement is very less and might cause some errors and if any error occurs it takes a lot of time for the developer to identify the root cause.

8. APPLICATIONS

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

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

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

One can plan to explore methods for gaining more insight in the nature of the patterns that are detected by neural networks, as well as making the determinants of a certain prediction transparent. 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

- https://developer.ibm.com/tutorials/how-to-create-a-node-red-starter-application/
- https://bookdown.org/caoying4work/watsonstudio-workshop/jn.html#deploy-model-as-web-service
- https://www.ibm.com/watson/products-services

12. APPENDIX

- **Watson Assistant:**
  Watson Assistant is a conversation AI platform that helps you provide customers fast, straightforward and accurate answers to their questions, across any application, device or channel.
- **Watson Studio:**
  Analysts prepare data and build models at scale across any cloud. Build models using images with IBM Watson Visual Recognition and texts with IBM Watson Natural Language Classifier. Deploy and run models through one-click integration with IBM Watson Machine Learning.
- **IBM Cloud Function:**
  IBM Cloud provides a full-stack, public cloud platform with a variety of

offerings in the catalog, including compute, storage, and networking options, end-to-end developer solutions for app development, testing and deployment, security management services, traditional and open-source databases

- **Node-Red:**
  Node-RED is a flow-based development tool for visual programming developed originally by IBM for wiring together hardware devices, APIs and online services as part of the Internet of Things. Node-RED provides a web browser-based flow editor, which can be used to create JavaScript functions.

CODE:

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

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_c5b69c545c424801b6b69f8715e8f26f = ibm_boto3.client(service_name='s3',
    ibm_api_key_id='Otzuw_whbwJeQtrJlzAUlS1jAPNV3m-WHMY1BcZ4F88C',
    ibm_auth_endpoint="https://iam.cloud.ibm.com/oidc/token",
    config=Config(signature_version='oauth'),
    endpoint_url='https://s3-api.us-geo.objectstorage.service.networklayer.com')

body =
client_c5b69c545c424801b6b69f8715e8f26f.get_object(Bucket='lifeexpectancypredict
ion-donotdelete-pr-1vuikf19rmsxi0',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 )

df_data_2 = pd.read_csv(body)
df_data_2.head()

data = df_data_2
data.head(n=5)
data.info()
sns.heatmap(pd.isnull(data));
a=list(data.columns)
b=[]
for i in a:
    c=data[i].isnull().sum()
    b.append(c)
null_df=pd.DataFrame({'Feature name':a,'no. of Nan':b})
null_df

data['Life expectancy ']=data['Life expectancy '].fillna(value=data['Life
expectancy '].mean())

data['Adult Mortality']=data['Adult Mortality'].fillna(value=data['Adult
Mortality'].mean())
corr_data=data.corr()
corr_data

sns.heatmap(corr_data)
sns.scatterplot(x=data['Schooling'],y=data['Alcohol']);
def impute_Alcohol(cols):
    al=cols[0]
    sc=cols[1]
    if pd.isnull(al):
        if sc<=2.5:
            return 4.0
        elif 2.5<sc<=5.0:
            return 1.5
        elif 5.0<sc<=7.5:
            return 2.5
        elif 7.5<sc<=10.0:
            return 3.0
        elif 10.0<sc<=15:
            return 4.0
        elif sc>15:
            return 10.0
    else:
        return al

data['Alcohol']=data[['Alcohol','Schooling']].apply(impute_Alcohol,axis=1)
sns.heatmap(pd.isnull(data))
data['Alcohol']=data['Alcohol'].fillna(value=data['Alcohol'].mean())
# Rechecking the Heatmap.
sns.heatmap(pd.isnull(data))
sns.distplot(data['Alcohol']);
scipy.stats.skew(data['Alcohol'],axis=0)

sns.scatterplot(x=data['Life expectancy '],y=data['Polio']);
def impute_polio(c):
    p=c[0]
    l=c[1]
    if pd.isnull(p):
        if l<=45:
            return 80.0
        elif 45<l<=50:
            return 67.0
        elif 50<l<=60:
            return 87.44
        elif 60<l<=70:
            return 91
        elif 70<l<=80:
            return 94.3
        elif l>80:
            return 95
    else:
        return p

data['Polio']=data[['Polio','Life expectancy ']].apply(impute_polio,axis=1)
a=list(data.columns)
b=[]
for i in a:
    c=data[i].isnull().sum()
    b.append(c)
null_df=pd.DataFrame({'Feature name':a,'no. of Nan':b})
null_df

sns.scatterplot(x=data['Polio'],y=data['Diphtheria ']);
def impute_Diptheria(c):
    d=c[0]
    p=c[1]
    if pd.isnull(d):
        if p<=10:
            return 75.0
        elif 10<p<=40:
            return 37.0
        elif 40<p<=45:
            return 40.0
        elif 45<p<=50:
            return 50.0
        elif 50<p<=60:
            return 55.0
        elif 60<p<=80:
            return 65.0
        elif p>80:
            return 90.0
    else:
        return d
data['Diphtheria ']=data[['Diphtheria ','Polio']].apply(impute_Diptheria,axis=1)
a=list(data.columns)
b=[]
for i in a:
    c=data[i].isnull().sum()
    b.append(c)
null_df=pd.DataFrame({'Feature name':a,'no. of Nan':b})
null_df
sns.scatterplot(x=data['Diphtheria '],y=data['Hepatitis B']);
def impute_HepatatisB(cols):
    hep=cols[0]
    dip=cols[1]
    if pd.isnull(hep):
        if dip<=15:
            return 75.0
        elif 15<dip<=30:
            return 20.0
        elif 30<dip<=45:
            return 38.0
        elif 45<dip<=60:
            return 43.0
        elif 60<dip<=80:
            return 63.0
        elif dip>80:
            return 88.4
    else:
        return hep

data['Hepatitis B']=data[['Hepatitis B','Diphtheria
']].apply(impute_HepatatisB,axis=1)
data[data['Diphtheria ']>80.0]['Hepatitis B'].mean()
a=list(data.columns)
b=[]
for i in a:
    c=data[i].isnull().sum()
    b.append(c)
null_df=pd.DataFrame({'Feature name':a,'no. of Nan':b})
null_df

sns.scatterplot(x=data['Life expectancy '],y=data[' BMI ']);

def impute_BMI(c):
    b=c[0]
    l=c[1]
    if pd.isnull(b):
        if l<=50:
            return 25.0
        elif 50<l<=60:
            return 25.0
        elif 60<l<=70:
            return 32.0
        elif 70<l<=80:
            return 46.8
        elif 80<l<=100:
            return 60.0
    else:
        return b

data[' BMI ']=data[[' BMI ','Life expectancy ']].apply(impute_BMI,axis=1)
a=list(data.columns)
b=[]
for i in a:
    c=data[i].isnull().sum()
    b.append(c)
null_df=pd.DataFrame({'Feature name':a,'no. of Nan':b})
null_df

sns.scatterplot(y=data['Total expenditure'],x=data['Alcohol']);

def impute_Total_exp(c):
    t=c[0]
    a=c[1]
    if pd.isnull(t):
        if a<=2.5:
            return 5.08
        elif 2.5<a<=5.0:
            return 6.0
        elif 5.0<a<=10.0:
            return 6.71
        elif 10.0<a<=12.5:
            return 6.9
        elif a>12.5:
            return 6.68
    else:
        return t

data['Total expenditure']=data[['Total
expenditure','Alcohol']].apply(impute_Total_exp,axis=1)
sns.scatterplot(x=data['percentage expenditure'],y=data['GDP']);
def impute_GDP(c):
    g=c[0]
    p=c[1]
    if pd.isnull(g):
        if p<=1250:
            return 1100.0
        elif 1250<p<=2500:
            return 1800.0
        elif 2500<p<=3750:
            return 2900.0
        elif 3750<p<=7500:
            return 3500.0
        elif 7500<p<=8750:
            return 4500.0
        elif 8750<p<=10000:
            return 5000.0
        elif 10000<p<=11250:
            return 5700.0
        elif 11250<p<=12500:
            return 7000.0
        elif 12500<p<=15000:
            return 8000.0
        elif 15000<p<=17500:
            return 9000.0
        elif p>17500:
            return 8500.0
    else:
        return g

data['GDP']=data[['GDP','percentage expenditure']].apply(impute_GDP,axis=1)
sns.scatterplot(x=data['infant deaths'],y=data['Population']);

def impute_population(c):
    p=c[0]
    i=c[1]
    if pd.isnull(p):
        if i<=100:
            return 0.19*((10)**9)
        elif 100<i<=250:
            return 0.18*((10)**9)
        elif 250<i<=350:
            return 0.02*((10)**9)
        elif 350<i<=900:
            return 0.1*((10)**9)
        elif 900<i<=1100:
            return 0.18*((10)**9)
        elif 1100<i<=1250:
            return 0.05*((10)**9)
        elif 1250<i<=1500:
            return 0.19*((10)**9)
        elif 1500<i<=1750:
            return 0.05*((10)**9)
        elif i>1750:
            return 0.1*((10)**9)
    else:
        return p

data['Population']=data[['Population','infant
deaths']].apply(impute_population,axis=1)
sns.scatterplot(x=data[' BMI '],y=data[' thinness  1-19 years']);

def impute_Thin_1(c):
    t=c[0]
    b=c[1]
    if pd.isnull(t):
        if b<=10:
            return 5.0
        elif 10<b<=20:
            return 10.0
        elif 20<b<=30:
            return 8.0
        elif 30<b<=40:
            return 6.0
        elif 40<b<=50:
            return 3.0
        elif 50<b<=70:
            return 4.0
        elif b>70:
            return 1.0
    else:
        return t

data[' thinness  1-19 years']=data[[' thinness  1-19 years',' BMI
']].apply(impute_Thin_1,axis=1)
sns.scatterplot(x=data[' BMI '],y=data[' thinness 5-9 years']);

def impute_Thin_1(c):
    t=c[0]
    b=c[1]
    if pd.isnull(t):
        if b<=10:
            return 5.0
        elif 10<b<=20:
            return 10.0
        elif 20<b<=30:
            return 8.0
        elif 30<b<=40:
            return 6.0
        elif 40<b<=50:
            return 3.0
        elif 50<b<=70:
            return 4.0
        elif b>70:
            return 1.0
    else:
        return t

data[' thinness 5-9 years']=data[[' thinness 5-9 years',' BMI
']].apply(impute_Thin_1,axis=1)
sns.scatterplot(x=data['Life expectancy '],y=data['Income composition of
resources']);

def impute_Income(c):
    i=c[0]
    l=c[1]
    if pd.isnull(i):
        if l<=40:
            return 0.4
        elif 40<l<=50:
            return 0.42
        elif 50<l<=60:
            return 0.402
        elif 60<l<=70:
            return 0.54
        elif 70<l<=80:
            return 0.71
        elif l>80:
            return 0.88
    else:
        return i

data['Income composition of resources']=data[['Income composition of
resources','Life expectancy ']].apply(impute_Income,axis=1)
sns.scatterplot(x=data['Life expectancy '],y=data['Schooling']);

def impute_schooling(c):
    s=c[0]
    l=c[1]
    if pd.isnull(s):
        if l<= 40:
            return 8.0
        elif 40<l<=44:
            return 7.5
        elif 44<l<50:
            return 8.1
        elif 50<l<=60:
            return 8.2
        elif 60<l<=70:
            return 10.5
        elif 70<l<=80:
            return 13.4
        elif l>80:
            return 16.5
    else:
        return s
```

```
data['Schooling']=data[['Schooling','Life expectancy
']].apply(impute_schooling,axis=1)
data[(data['Life expectancy ']>80) & (data['Life expectancy
']<=90)]['Schooling'].mean()

a=list(data.columns)
b=[]
for i in a:
    c=data[i].isnull().sum()
    b.append(c)
null_df=pd.DataFrame({'Feature name':a,'no. of Nan':b})
null_df

y=data['Life expectancy ']
sns.distplot(y);

X=data.drop('Life expectancy ',axis=1)
X.info()

X['Country']

X['Country'].unique()

X['Country'].nunique()
X['Status'].unique()
Country_dummy=pd.get_dummies(X['Country'])
status_dummy=pd.get_dummies(X['Status'])
X.drop(['Country','Status'],inplace=True,axis=1)
X=pd.concat([X,Country_dummy,status_dummy],axis=1)
X.info()
X.head()
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.30,
random_state=101)
from sklearn.linear_model import LinearRegression
Linear_model= LinearRegression()
Linear_model.fit(X_train,y_train)

predictions1=Linear_model.predict(X_test)
predictions1[0:10]

from sklearn.metrics import mean_squared_error
print(mean_squared_error(y_test,predictions1)**(0.5))
1.915251634676079
from sklearn.metrics import r2_score
r2_score(y_test,predictions1)

from sklearn.linear_model import Ridge
ridge_model=Ridge()
ridge_model.fit(X_train,y_train)

predictions2=ridge_model.predict(X_test)
print(mean_squared_error(y_test,predictions2)**(0.5))

from sklearn.metrics import r2_score
```

```
r2_score(y_test,predictions1)
from sklearn.linear_model import Ridge
ridge_model=Ridge()
ridge_model.fit(X_train,y_train)
predictions2=ridge_model.predict(X_test)
print(mean_squared_error(y_test,predictions2)**(0.5))
2.202246152327118
from sklearn.linear_model import Lasso
lasso_model=Lasso(alpha=0.00000001)



predictions3=lasso_model.predict(X_test)
print(mean_squared_error(y_test,predictions3)**(0.5))



from watson_machine_learning_client import WatsonMachineLearningAPIClient

wml_credentials={

  "apikey": "f9oOZMrSSZ26D1Gx2HyXnU2JfLAwy66-F3Rvaza4sTUk",
  "iam_apikey_description": "Auto-generated for key d20954e7-e157-4e20-b50f-
9ad0296de2b8",
  "iam_apikey_name": "wdp-writer",
  "iam_role_crn": "crn:v1:bluemix:public:iam::::serviceRole:Writer",
  "iam_serviceid_crn": "crn:v1:bluemix:public:iam-
identity::a/c15fdb6e00524eae96f67200c4246156::serviceid:ServiceId-ff1be345-68fe-
44d2-b1e0-55c8599411d2",
  "instance_id": "7b5149df-d618-4d9a-b7c5-97990a077b76",
  "url": https://us-south.ml.cloud.ibm.com

}
client = WatsonMachineLearningAPIClient( wml_credentials )
model_props = {
    client.repository.ModelMetaNames.AUTHOR_NAME: "Paritosh Purandar",
    client.repository.ModelMetaNames.AUTHOR_EMAIL:
"paritosh.purandar2017@vitstudent.ac.in"
    client.repository.ModelMetaNames.NAME: "Predicting_Life_Expectancy"
}

model_artifact =client.repository.store_model(rf, 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="LifeExpectancy")

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