

PROJECT REPORT

Name: Mohammad Faisal Quraishi

Email: faisalahmad7868@gmail.com

Title: **Predicting Life Expectancy of A Country**

Category: **Machine Learning**

WebPage Link:

<https://node-red-mjktp.eu-gb.mybluemix.net/ui/#!/0?socketid=zEAUMI51kYBVPLXTAAAV>

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. It is very important to predict average life expectancy of a country to analyse further requirements to increase its rate of growth or stabilise the rate of growth in that country. So this is a typical Regression Machine Learning project that leverages historical data to predict insights into the future.

The end product will be a webpage where you need to give all the required inputs and then submit it . Afterwards it will predict the life expectancy value based on your regression technique.

Project Requirements: Python, IBM Cloud, IBM Watson

Functional Requirements: IBM cloud

Technical Requirements: ML, WATSON Studio, Python, Node-Red

Software Requirements: Watson Studio, Node-Red

Project Deliverables: Smartinternz Internship

Project Team: Mohammad Faisal

Project Duration: 23.5 Days

1.2. Purpose

The purpose of the project is to design a model for predicting Life Expectancy rate of a country given various features 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. Proposed Problem

The typical regression model that can predict average life expectancy of the country based on some user inputted values such as GDP, BMI, HIV/AIDS, Year, Alcohol intake and etc.

2.2. Proposed Solution

Steps:

- a) Create IBM cloud services
- b) Configure Watson Studio
- c) Create Node-Red Flow to connect all services together
- d) Deploy and run Node-Red app

2.2.1. Create IBM cloud Services

- Watson Studio
- Machine Learning resource
- Node-Red

2.2.2. Configure Watson Studio

After creating all services, Go to resource list and launch watson studio then get started with watson studio. Then create an empty project and add machine learning resource as associated services in settings. Create a token as editor type.

Then add dataset and empty jupyter notebook into Assets.

After that go to notebook and write your code to build model and get the scoring endpoint url.

Steps for notebook:

- Install `Watson_machine_learning_client`
- Import necessary libraries
- Import dataset
- Data Preprocessing
 - Removing unusual species in column names using `rename` function.
 - Replacing nan values if any with their mean values.
- Exploratory Data Analysis
 - Plotting a heatmap to check if dimensional reduction can be performed
 - Plotting a pairplot for analysing pairwise relationship among features.
- Train and Test
 - The dataset was splitted into two parts i.e Input and Output. As Life Expectancy needs to be predicted so it is to be treated as output and all other columns are treated as Input

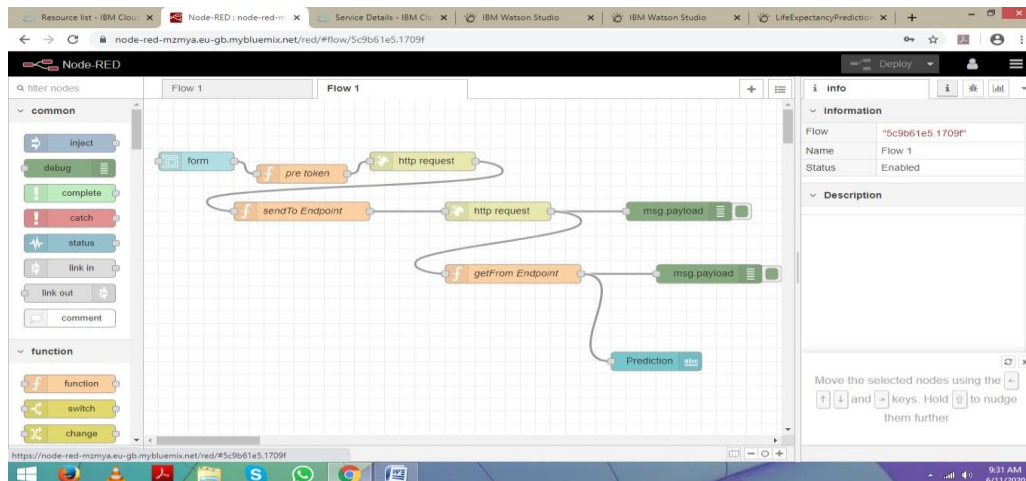
- Afterwards as we need regression technique to build our model so each and every column needs to be numeric . So then we check for numeric and categoric columns
 - Then we standardize the numeric and categoric columns using pipelining.
 - At first independent pipelines for both the parts were designed then they were joined using columntransform
 - After that a regressor pipeline was designed using the regression technique.
 - So I have used ExtraTreesRegressor technique of sklearn.ensemble as my regression algorithm because it best fits my dataset.
 - Then train and test split was performed and 80% of dataset were trained data and 20% were test data.
 - Then dataset was fitted and predicted.
 - Then error and accuracy was estimated and the mean squared error is 2.4 whereas the R2_score or accuracy is 97.07%.
- **Model Building and Deployment**
 - At first the machine learning service credentials was stored in a variable and passed into WatsonMachineLearningAPIClient.

```
wml_credentials={
  "apikey": "bolp9z-fIcxOwWd2OVgW_ME3LRehhTcjoN_4KWWMGYIx3",
  "instance_id": "6ee18429-4d62-454e-b51e-fc9d4b9cec1e",
  "url": "https://eu-gb.ml.cloud.ibm.com"
}
client = WatsonMachineLearningAPIClient(wml_credentials)
```

- Then the model was build and stored in model_artifact.
- Then the model was deployed and scoring_endpoint url was generated.

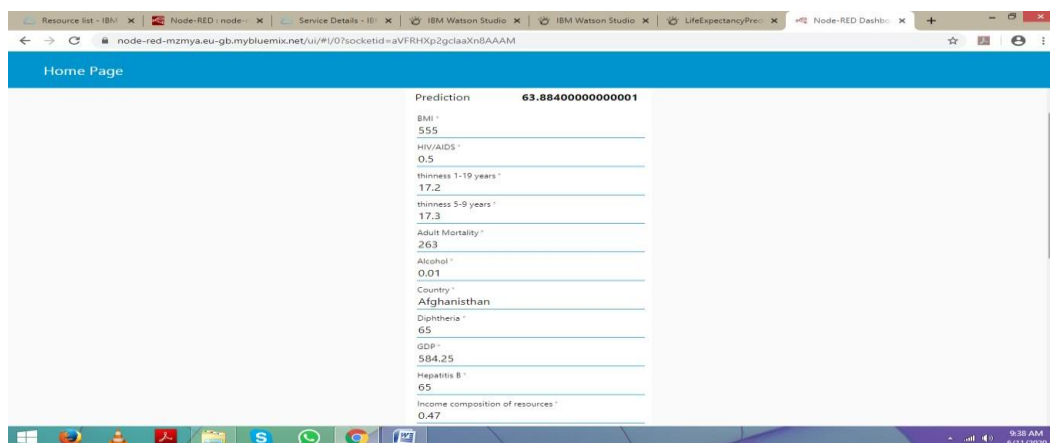
2.2.3. Create Node-Red Flow to connect all services together

- Go to Node-Red Editor from resource list.
- Install node-red Dashboard from manage pallette.
- Now create the flow with the help of following node.
 - Inject
 - Debug
 - Function
 - Ui_Form
 - Ui_Text



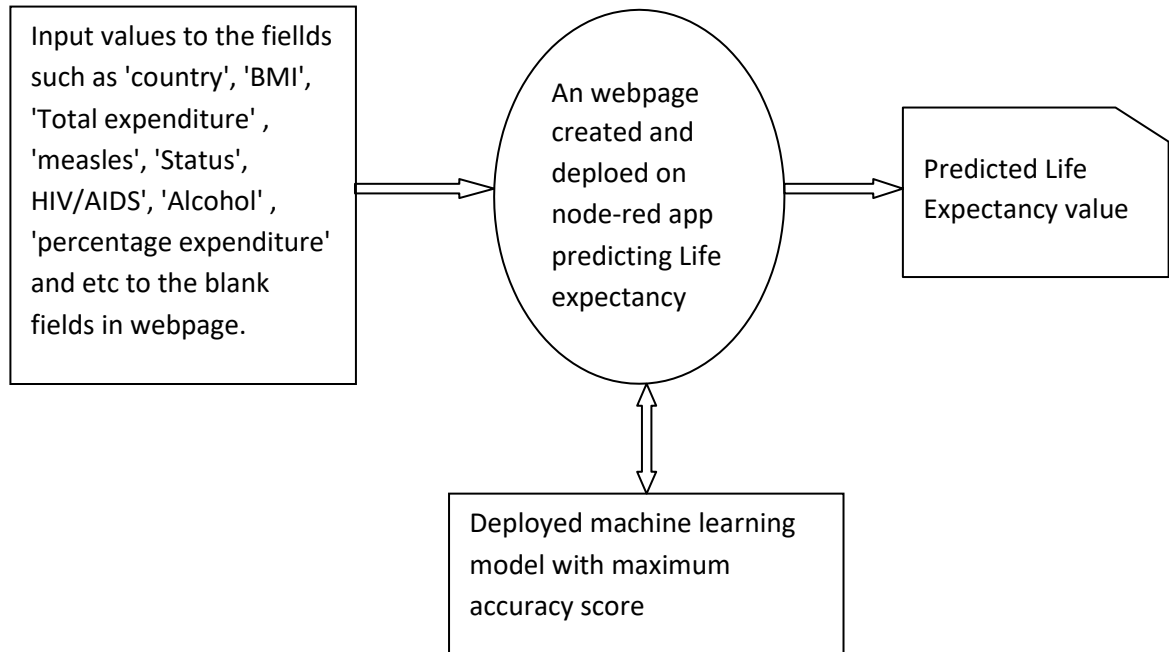
- Deploy and run Node Red app.

Deploy the Node Red flow. Then copy the link url upto .net/ and paste at a new tab by ui at the end of the url like this



3. THEORETICAL ANALYSIS

3.1. BLOCK DIAGRAM

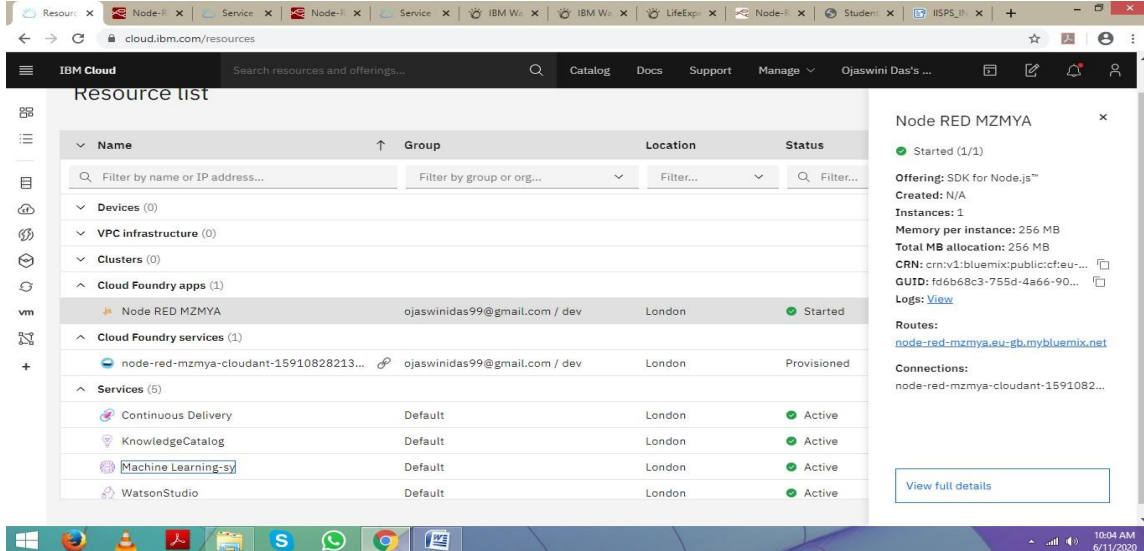


3.2. HARDWARE / SOFTWARE DESIGNING

- **Project Requirements:** Python, IBM Cloud, IBM Watson
- **Functional Requirements:** IBM cloud
- **Technical Requirements:** ML, WATSON Studio, Python, Node-Red
- **Software Requirements:** Watson Studio, Node-Red

4. EXPERIMENTAL INVESTIGATIONS

A) IBM Cloud Resource List



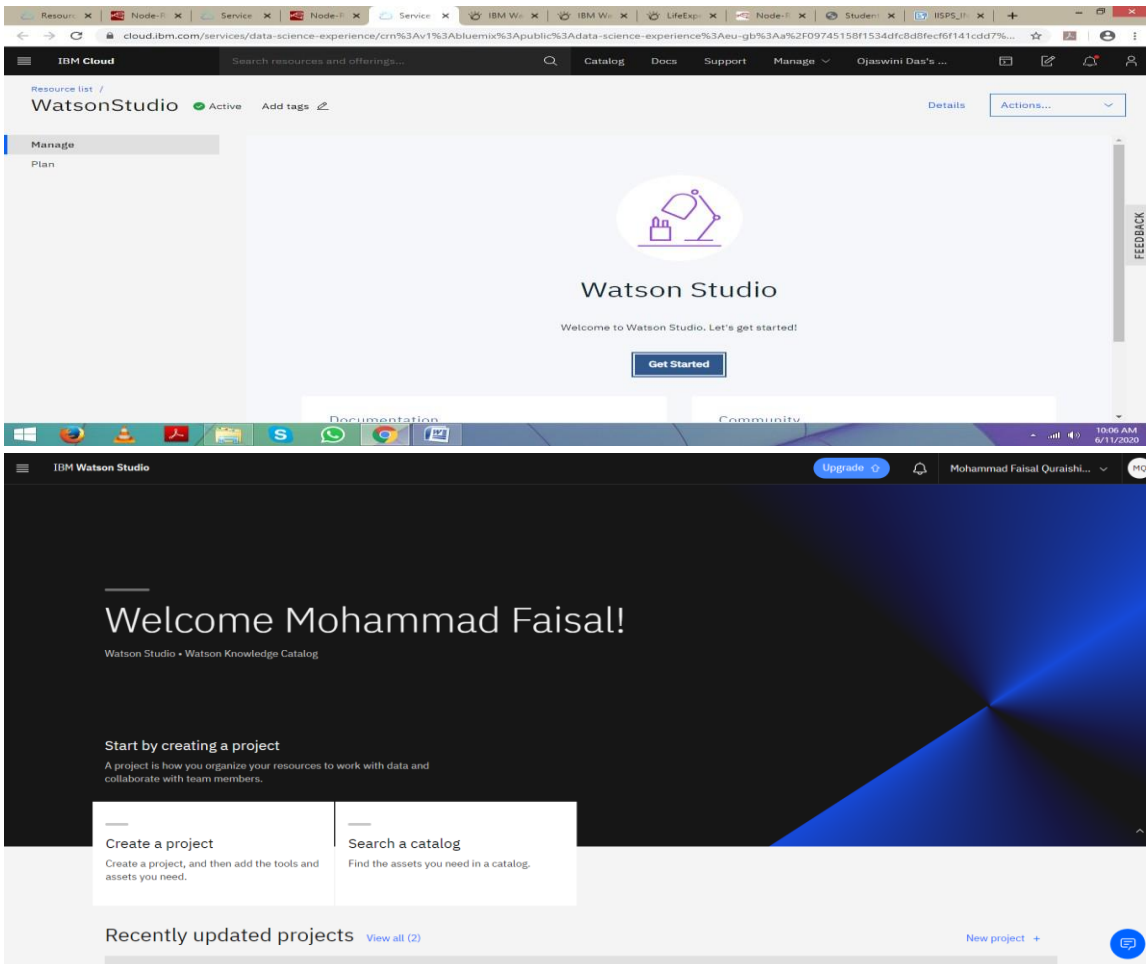
The screenshot shows the IBM Cloud Resource List page. The table lists resources with columns for Name, Group, Location, and Status. The 'Node RED MZMYA' resource is highlighted, and its details are shown in a side panel.

| Name | Group | Location | Status |
|--|-------------------------------|----------|-------------|
| Node RED MZMYA | ojaswinidas99@gmail.com / dev | London | Started |
| node-red-mzmya-cloudant-15910828213... | ojaswinidas99@gmail.com / dev | London | Provisioned |
| Continuous Delivery | Default | London | Active |
| KnowledgeCatalog | Default | London | Active |
| Machine Learning-sy | Default | London | Active |
| WatsonStudio | Default | London | Active |

Node RED MZMYA Details:

- Started (1/1)
- Offering: SDK for Node.js™
- Created: N/A
- Instances: 1
- Memory per instance: 256 MB
- Total MB allocation: 256 MB
- CRN: crn:v1:bluemix:public:cf:eu-...
- GUID: fd6b68c3-755d-4a66-90...
- Logs: [View](#)
- Routes: [node-red-mzmya.eu-gb.mybluemix.net](#)
- Connections: node-red-mzmya-cloudant-1591082...

B) IBM Watson Studio

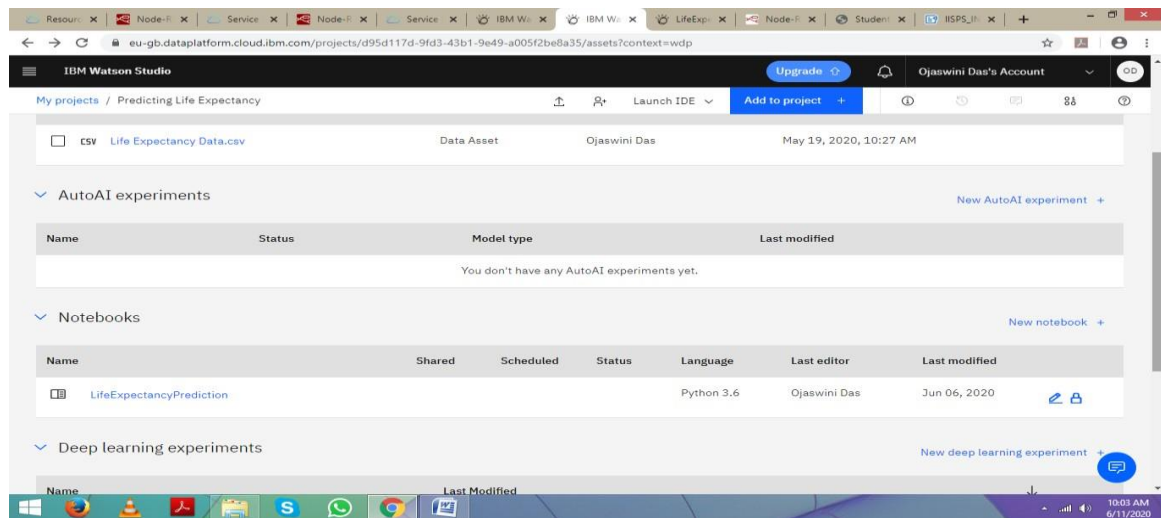


The screenshot shows the IBM Watson Studio interface. The main heading is 'Welcome Mohammad Faisal!'. Below it, there are two main options: 'Create a project' and 'Search a catalog'. The 'Create a project' option includes a sub-heading 'Start by creating a project' and a description: 'A project is how you organize your resources to work with data and collaborate with team members.' The 'Search a catalog' option includes a sub-heading 'Search a catalog' and a description: 'Find the assets you need in a catalog.'

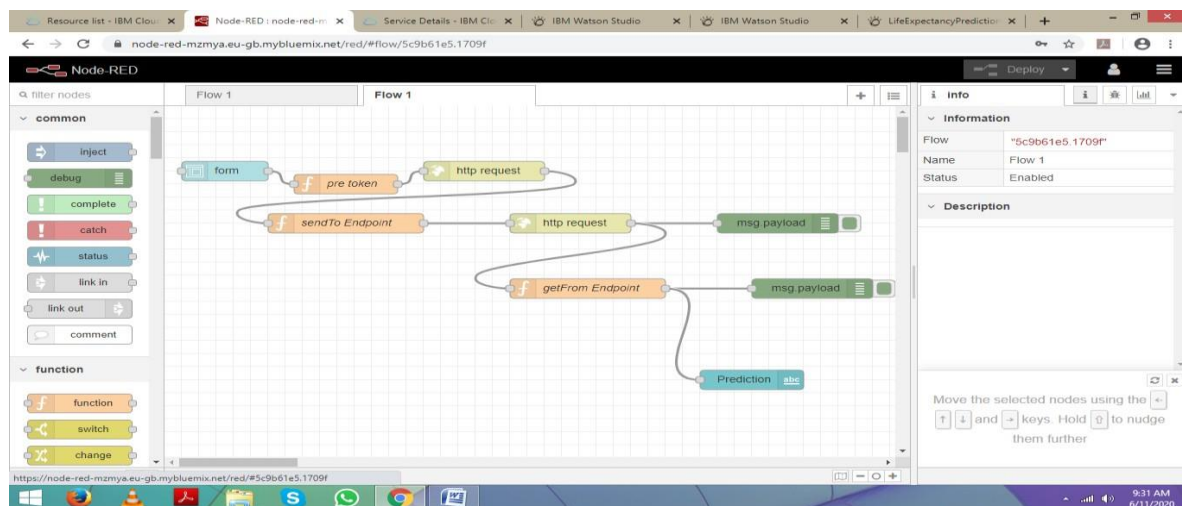
Recent projects table:

| Name | Role | Collaborator | Date created | Last updated |
|------|------|--------------|--------------|--------------|
| | | | | |

C) IBM Cloud Project Details



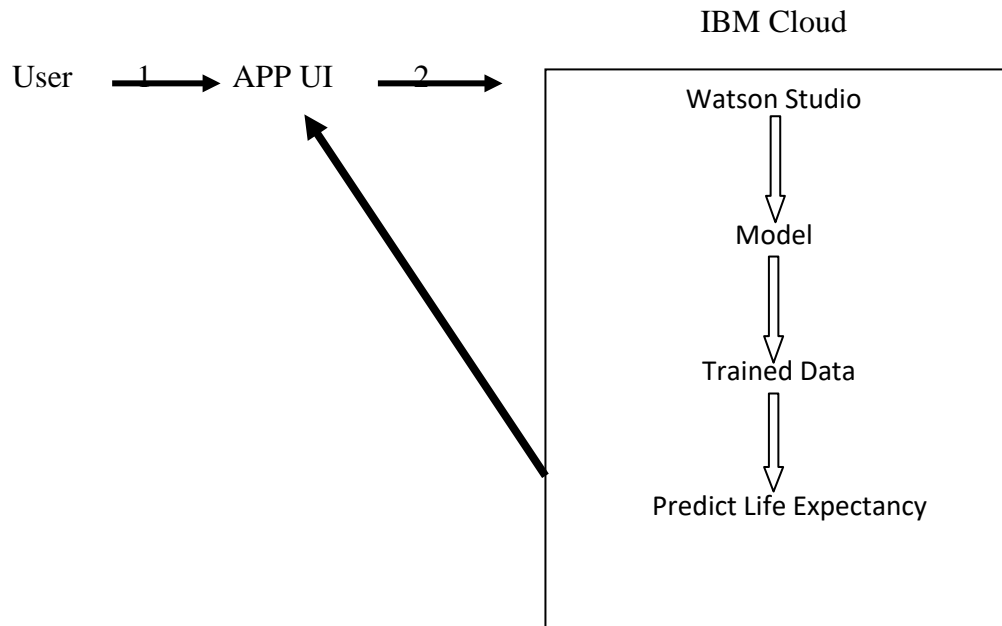
D) Node-Red Flow



E) Life Expectancy Prediction UI



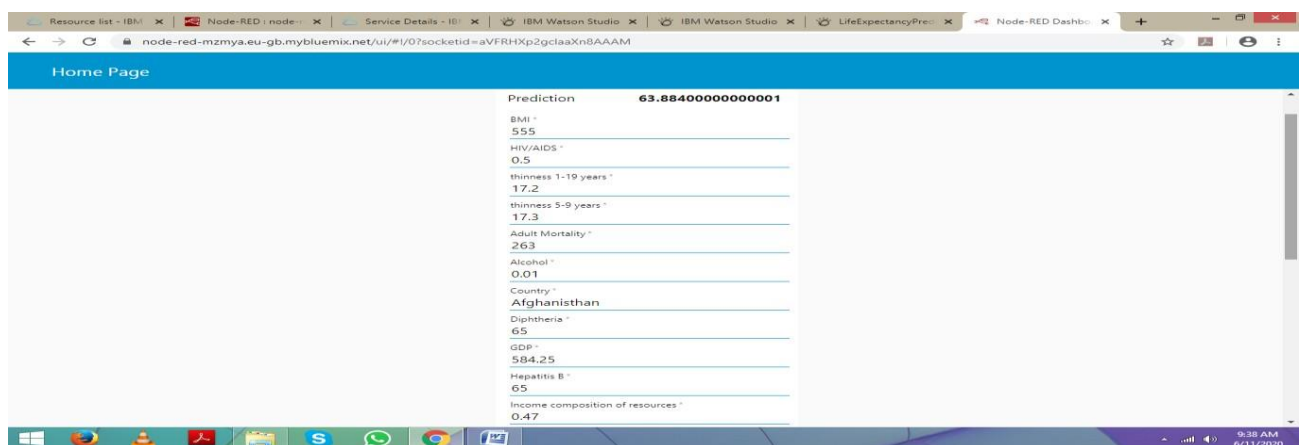
5. FLOWCHART



- The user input all the required values in the app
- The data then entered into watson and the scoring_endpoint url matches with the deployed model.
- Then it enters into trained data and predict the life expetancy value
- The value predicted is prompted in the app screen.

6. RESULT

This is the Life Expectancy UI.



7. ADVANTAGES AND DISADVANTAGES

ADVANTAGES:

- a) Health Inequalities: Life expectancy has been used nationally to monitor health inequalities of a country.
- b) Reduced Costs: This is a simple webpage and can be accessed by any citizen of a country to calculate life expectancy of their country and doesnot required any kind of payment neither for designing nor for using.
- c) User Friendly Interface: This interface requires no background knowledge of how to use it. It's a simple interface and only ask for required values and predict the output.

DISADVANTAGES:

- a) Wrong Prediction: As it depends completely on user, so if user provides some wrong values then it will predict wrong value.
- b) Average Prediction: The model predicts average or approximate value with 97.07% accuracy but not accurate value.

8. APPLICATION

- a) It can be used to monitor health inequalities of a country.
- b) It can be used to develop statistics for country development process.
- c) It can be used to analyse the factors for high life expectancy.
- d) It is user friendly and can be used by anyone.

9. CONCLUSION

This user interface will be useful for the user to predict life expectancy value of their own country or any other country based on some required details such as GDP, BMI, Year, Alcohol Intake, Total expenditure and etc.

10. FUTURE SCOPE

Future Scope of the Model can be:

a) 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 I have decided to do some kind of feature reduction or replacement of some features as individuals or groups to make it more user friendly.

b) 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.

c) Integrating with services such as speech recognition

11. BIBLIOGRAPHY

- <https://cloud.ibm.com/docs/overview?topic=overview-what-is-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://bookdown.org/caoying4work/watsonstudio-workshop/jn.html>
- <https://www.kaggle.com/kumarajarshi/life-expectancy-who>
- <https://www.youtube.com/watch?v=Jtej3Y6uUng>
- <https://bookdown.org/caoying4work/watsonstudio-workshop/jn.html#deploy-model-as-web-service>
- <https://machinelearningmastery.com/columntransformer-for-numerical-and-categorical-data/>

APPENDIX: Source Code

1) Notebook

```
#import libraires
```

```
import pandas as pd
```

```
import numpy as np
```

```
import os
```

```

import matplotlib.pyplot as plt
import seaborn as sns
import warnings
from sklearn.pipeline import Pipeline
from sklearn.preprocessing import OneHotEncoder
from sklearn.impute import SimpleImputer
from sklearn.preprocessing import StandardScaler
from sklearn.compose import ColumnTransformer
from sklearn.model_selection import train_test_split
from sklearn.ensemble import ExtraTreesRegressor
from sklearn.metrics import mean_squared_error, r2_score
from watson_machine_learning_client import WatsonMachineLearningAPIClient

```

#import dataset

#Data Preprocessing

```

df=df.rename(columns={'old name:new name'})
df=df.fillna(df.mean())

```

#Exploratory data Analysis

```

df_kor=df.corr()
plt.figure(figsize=(10,10))
#heatmap
sns.heatmap(df_kor,vmin=-1,vmax=1,annot=True,linewidth=0.1)
#pairplot
sns.pairplot(df)

```

#Train&Test

```

Y=df['Life expectancy']
X=df[df.columns.difference(['Life expectancy'])]
categorical_features = ['Country', 'Status']
categorical_feature_mask = X.dtypes==object
categorical_features = X.columns[categorical_feature_mask].tolist()
categorical_transformer = Pipeline(steps=[
    ('onehot', OneHotEncoder(handle_unknown='ignore')),
])
numeric_features = ['Year','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']
numeric_feature_mask = X.dtypes!=object
numeric_features = X.columns[numeric_feature_mask].tolist()
numeric_transformer = Pipeline(steps=[
    ('imputer', SimpleImputer(strategy='median')),
    ('scaler', StandardScaler()),
])
preprocessor = ColumnTransformer(
    transformers=[
        ('num', numeric_transformer, numeric_features),
        ('cat', categorical_transformer, categorical_features)
    ]
)
ExtraTreeRegressor = Pipeline([
    ('preprocessor', preprocessor),

```

```

    ('ExtraTreeRegressor', ExtraTreesRegressor(n_estimators=100, random_state=0))
])
reg = ExtraTreeRegressor.fit(X_train, Y_train)
test_pred=reg.predict(X_test)

```

#Model Building and Deployment

```

wml_credentials={
    "apikey": "*****",
    "instance_id": "*****",
    "url": "*****"
}
client = WatsonMachineLearningAPIClient(wml_credentials)

model_props = {client.repository.ModelMetaNames.AUTHOR_NAME: "****",
                client.repository.ModelMetaNames.AUTHOR_EMAIL: "*****",
                client.repository.ModelMetaNames.NAME: "LifeExpectancy"}
model_artifact=client.repository.store_model(ExtraTreeRegressor, meta_props=model_props)
model_uid = client.repository.get_model_uid(model_artifact)
create_deployment = client.deployments.create(model_uid, name="LifeExpectancyPrediction")
scoring_endpoint = client.deployments.get_scoring_url(create_deployment)
print(scoring_endpoint)

```

2) Flow.json

```
[{"id": " "}]
```