PROJECT REPORT

Name: Chinkit Manchanda

Email: chinkitm51@gmail.com

Title: Predicting Life Expectancy

Category: Machine Learning Internship

College: HMR Institute of Technology and Management

Webpage Link:

https://node-red-nnamd.eu-gb.mybluemix.net/ui/#!/0?socketid=609McA5eQw8hEBJRAAAp

1. **INTRODUCTION**

1.1 Overview

Life expectancy is a statistical measure of the average time a human being is expected to live, the project relies on accuracy of data. The Global Health Observatory (GHO) data repository under World Health Organization (WHO) keeps track of the health status as well as many other related factors for all countries. The data-set related to life expectancy, health factors for 193 countries has been collected from the same WHO data repository website and its corresponding economic data was collected from United Nation website. Among all categories of health-related factors only those critical factors were chosen which are more representative. It has been observed that in the past 15 years, there has been a huge development in health sector resulting in improvement of human mortality rates especially in the developing nations in comparison to the past 30 years. Therefore, in this project we have considered data from year 2000-2015 for 193 countries for further analysis. The individual data files have been merged together into a single data-set. The final merged file (final dataset) consists of 22 Columns and 2938 rows which meant 20 predicting variables. All predicting variables was then divided into several broad categories: Immunization related factors, Mortality factors, Economical factors and Social 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:

➤ **Functional Requirements:**
Predicting the life expectancy rate of a country.

➤ **Technical Requirements:**
Python, Machine Learning IBM Cloud, IBM Watson.

➤ **Hardware Requirements:**
Processor: i3 7th gen or higher
Speed: 2GHz or more
Hard disk space: 10 GB or more
Ram Memory: 4 GB or more

➤ **Software Requirements:**
Watson Studio, Node-Red

Project Deliverables:

- Project Documentation
- Machine Learning Prediction Model
- Node red flow diagram

Project Team: Chinkit Manchanda

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, Adult Mortality, education, 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 Problem

Although there have been lot of studies undertaken in the past on factors affecting life expectancy considering demographic variables, income composition and mortality rates. It was found that effect of immunization and human development index was not taken into account in the past. Also, some of the past research was done considering multiple linear regression based on data set of one year for all the countries. Hence, this gives motivation to resolve both the factors stated previously by formulating a regression model based on mixed effects model and multiple linear regression while considering data from a period of 2000 to 2015 for all the countries. Important immunization like Hepatitis B, Polio and Diphtheria will also be considered. In a nutshell, this study will focus on immunization factors, mortality factors, economic factors, social factors and other health related factors as well. Since the observations this dataset are based on different countries, it will be easier for a country to determine the predicting factor which is contributing to lower value of life expectancy. This will help in suggesting a country which area should be given importance in order to efficiently improve the life expectancy of its population.

2.2. Proposed Solution

Steps:

a) Create IBM cloud services

b) Configure Watson Studio

c) Create Machine Learning Notebook

d) Save and Deploy Model in Notebook

e) Create Node-Red Flow to connect all services together

f) 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

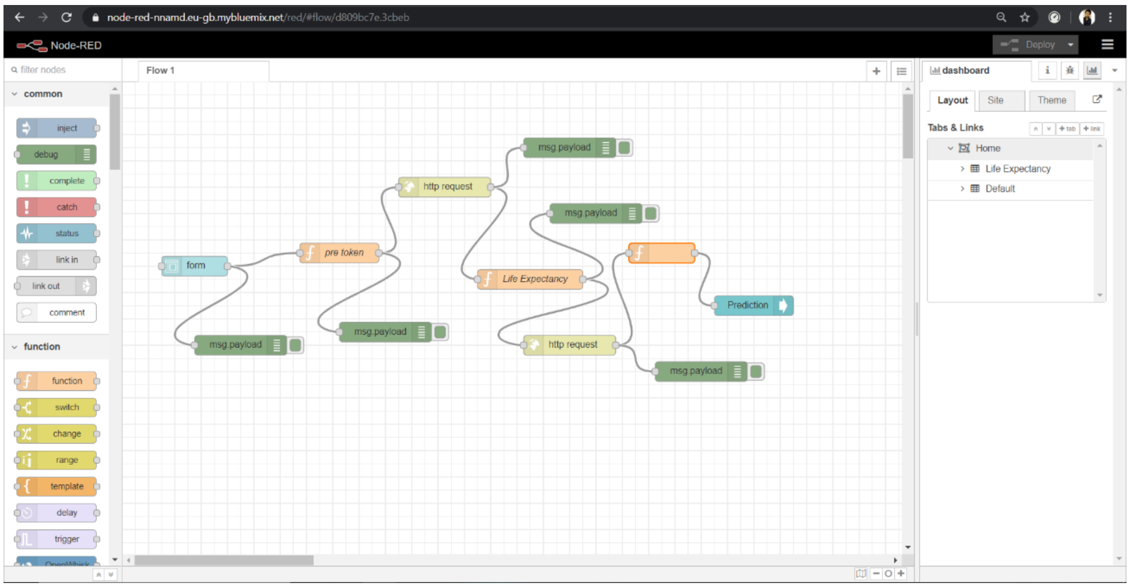
After creating all services, go to resource list and launch Watson studio then get started with Watson studio. Then create an empty project and add machine learning resource as associated services in settings. Create a token as editor type.
Then create empty Jupyter notebook into Assets and add dataset.

After that go to notebook and write your code to build model and get the scoring endpoint URL. Steps for notebook:

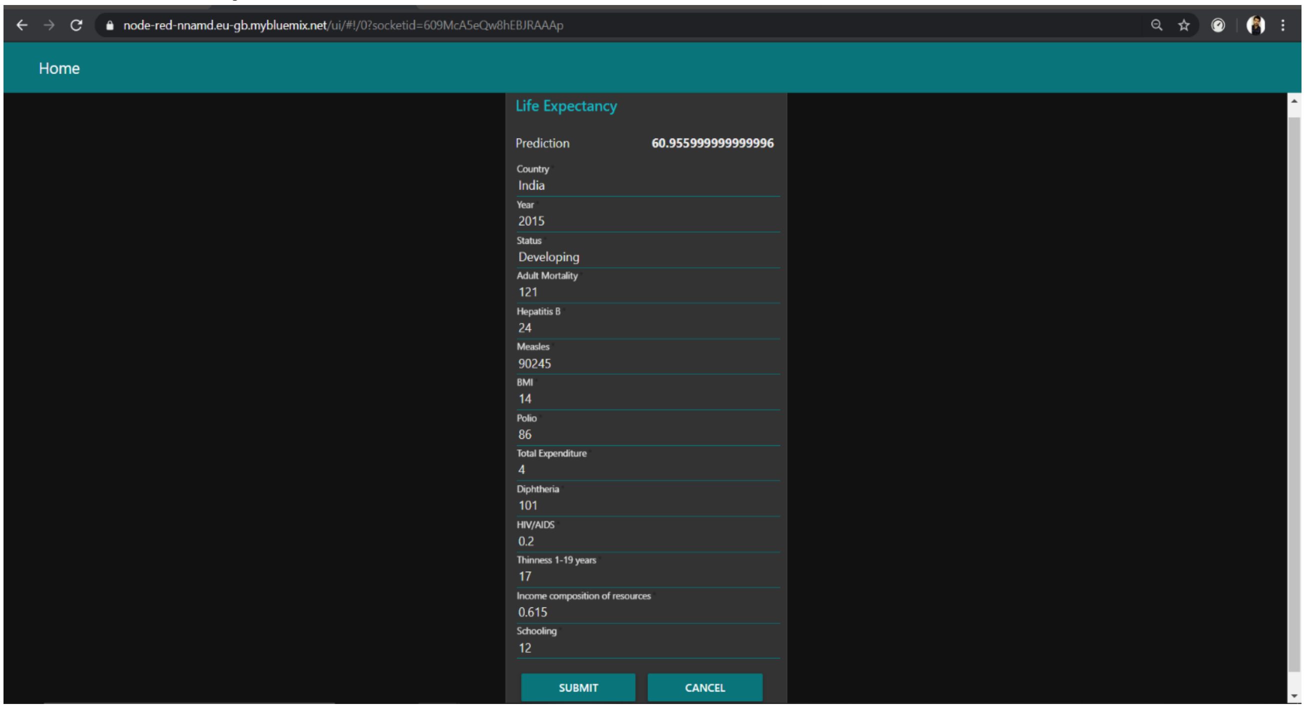
- Install Watson_machine_learning_client
- Import necessary libraries
- Import Dataset
- Descriptive Analysis of Data
    - Removing unusual species in column names using rename function.
    - Replacing nan values if any with their mean values.
    - Plotting a heatmap to check if dimensional reduction can be performed
- Calculating P-value to know the impact of features on the target value and remove the features with p-value more than 0.05.
- Train and Test
    - The dataset was split 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 variables.
    - 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 use LabelEncoder to convert categoric variables to numeric.
    - Or we can standardize the numeric and categoric columns using pipelining.
    - At first independent pipelines for both the parts were designed then they were joined using column transform.
    - After that a regressor pipeline was designed using the regression technique.
    - We apply different regression techniques including MultiLinear Regression, Decision Tree Regression, Random Forest Regression on our dataset.
    - So I have used Random Forest Regressor technique of sklearn.essemble as my regression algorithm as it gives best accuracy.
    - Then train and test split were performed and 80% of dataset were trained data and 20% were test data
    - Then dataset was fitted and predicted.
    - Then error and accuracy were estimated and the mean squared error is 3.4 whereas the R2_score or accuracy is 96.07%.
- Model Building and Deployment
    - At first the machine learning service credentials was stored in a variable and passed into WatsonMachineLearningAPIClient.
    - Then the model was built 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 pallete.
- 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 up to .net/ and paste at a new tab by UI at the end of the URL like this



node-red-nnamd.eu-gb.mybluemix.net/ui/#!/0?socketid=609McA5eQw8hEBJRAAAp

Home

Life Expectancy

Prediction 60.955999999999996

Country
India

Year
2015

Status
Developing

Adult Mortality
121

Hepatitis B
24

Measles
90245

BMI
14

Polio
86

Total Expenditure
4

Diphtheria
101

HIV/AIDS
0.2

Thinness 1-19 years
17

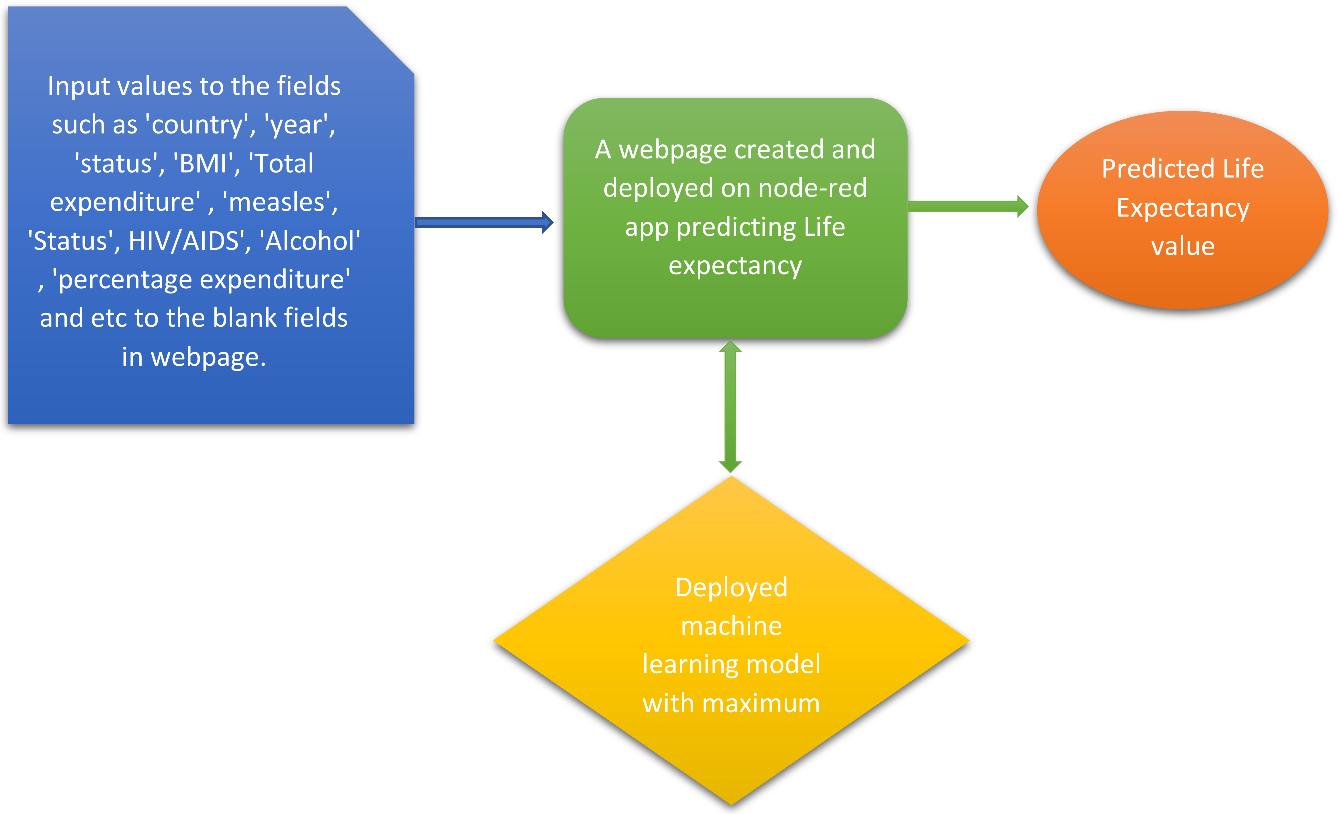
Income composition of resources
0.615

Schooling
12

SUBMIT CANCEL

3. THEORETICAL ANALYSIS

3.1. BLOCK DIAGRAM

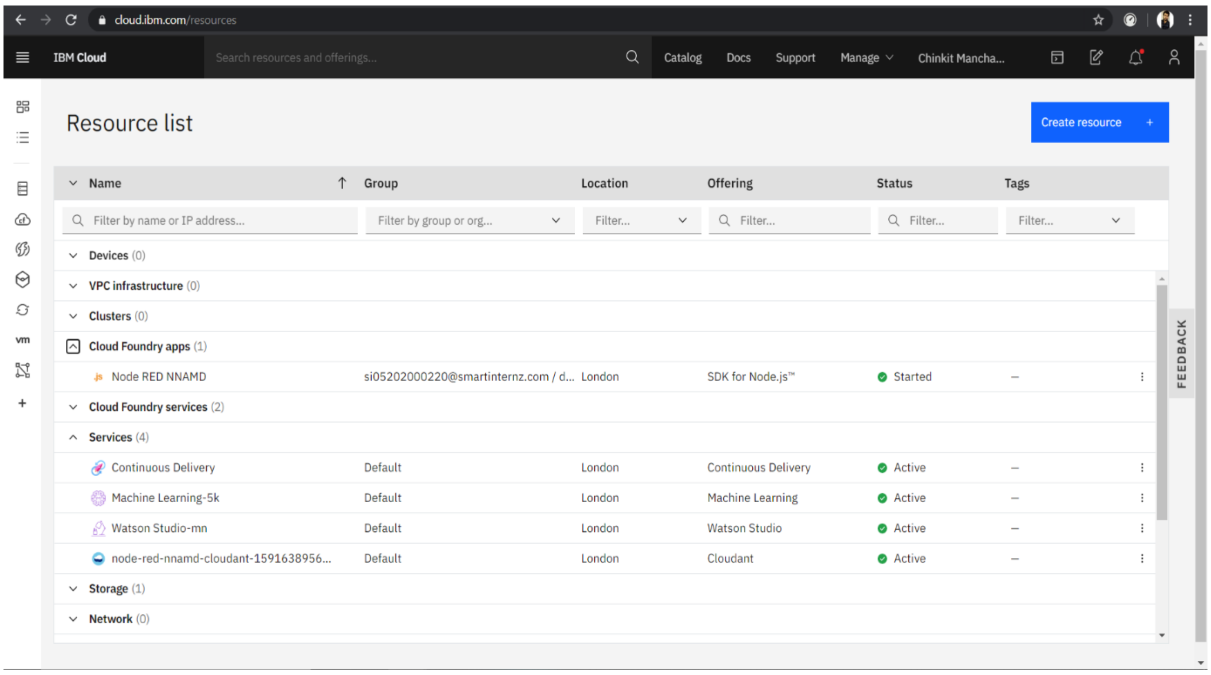

Input values to the fields such as 'country', 'year', 'status', 'BMI', 'Total expenditure' , 'measles', 'Status', HIV/AIDS', 'Alcohol' , 'percentage expenditure' and etc to the blank fields in webpage.
A webpage created and deployed on node-red app predicting Life expectancy
Predicted Life Expectancy value
Deployed machine learning model with maximum

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



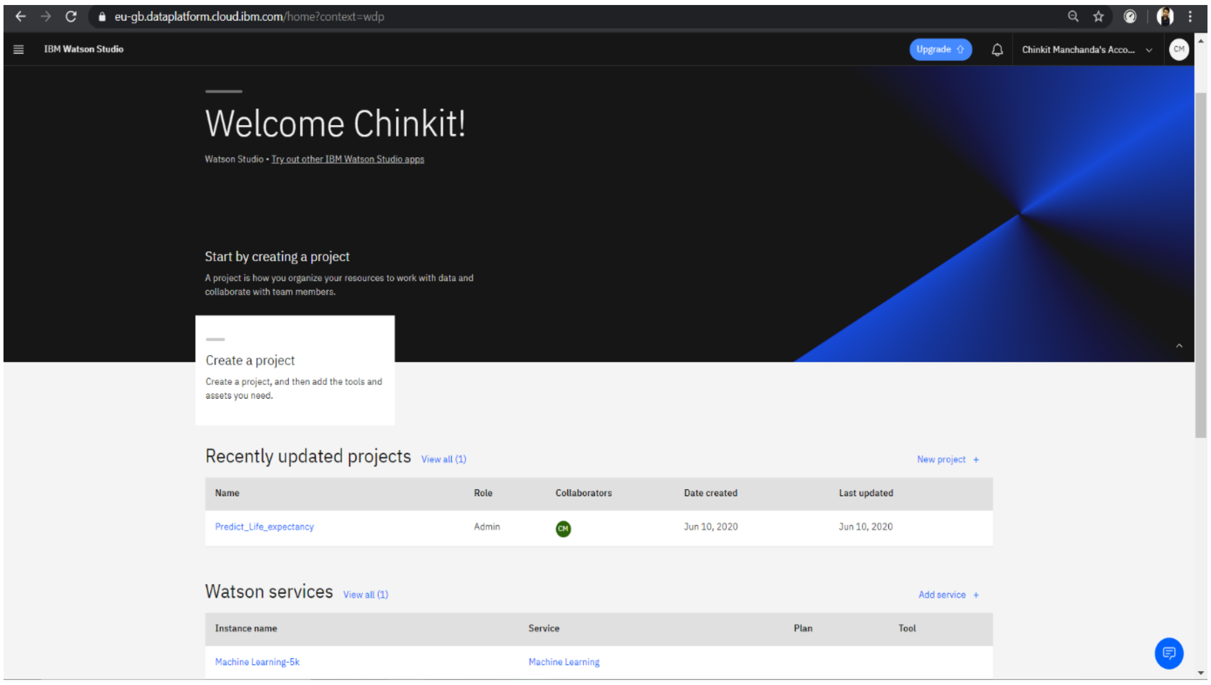
cloud.ibm.com/resources

IBM Cloud

Resource list

Name	Group	Location	Offering	Status	Tags
Devices (0)					
VPC infrastructure (0)					
Clusters (0)					
Cloud Foundry apps (1)					
Node RED NNAMD	si05202000220@smartinternz.com / d...	London	SDK for Node.js™	Started	–
Cloud Foundry services (2)					
Services (4)					
Continuous Delivery	Default	London	Continuous Delivery	Active	–
Machine Learning-5k	Default	London	Machine Learning	Active	–
Watson Studio-mn	Default	London	Watson Studio	Active	–
node-red-nnamd-cloudant-1591638956...	Default	London	Cloudant	Active	–
Storage (1)					
Network (0)					

B) IBM Watson Studio



eu-gb.dataplatform.cloud.ibm.com/home?context=wdp

IBM Watson Studio

Welcome Chinkit!

Watson Studio • Try out other IBM Watson Studio apps

Start by creating a project

A project is how you organize your resources to work with data and collaborate with team members.

Create a project

Create a project, and then add the tools and assets you need.

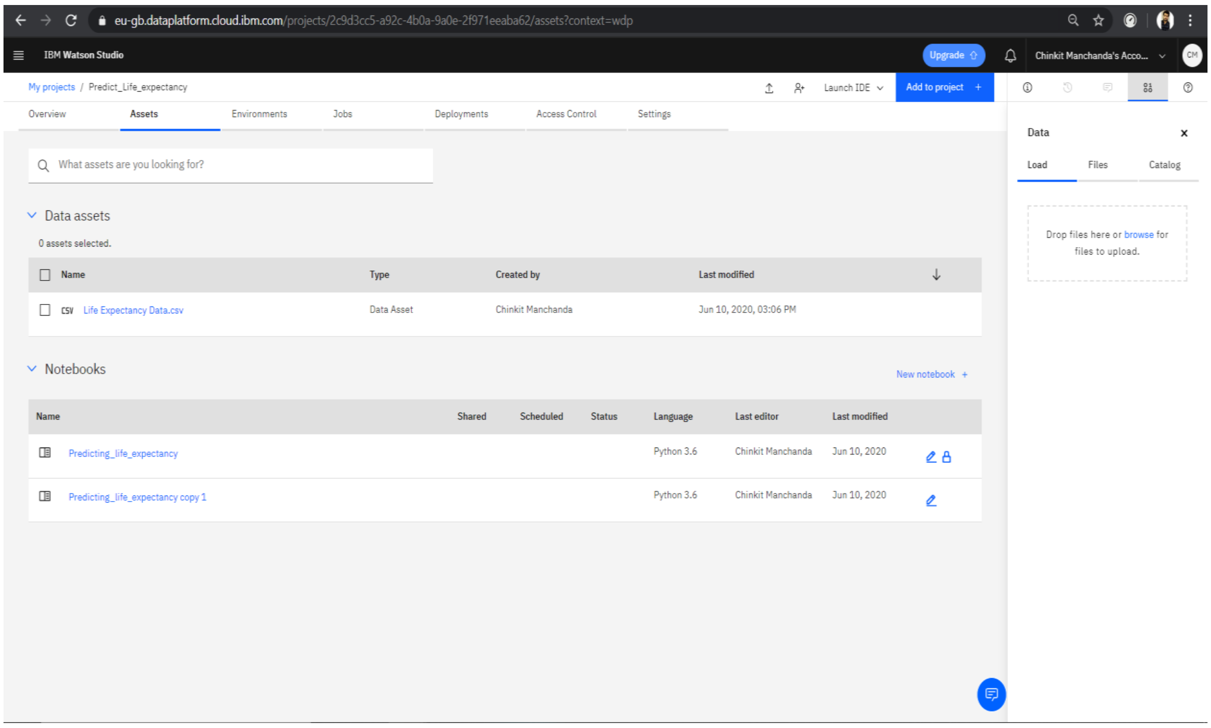
Recently updated projects View all (1)

Name	Role	Collaborators	Date created	Last updated
Predict_Life_expectancy	Admin	CM	Jun 10, 2020	Jun 10, 2020

Watson services View all (1)

Instance name	Service	Plan	Tool
Machine Learning-5k	Machine Learning		

C) IBM Cloud Project Details



eu-gb.dataplatform.cloud.ibm.com/projects/2c9d3cc5-a92c-4b0a-9a0e-2f971eeaba62/assets?context=wdp

IBM Watson Studio

My projects / Predict_Life_expectancy

Overview | Assets | Environments | Jobs | Deployments | Access Control | Settings

What assets are you looking for?

Data assets

0 assets selected.

Name	Type	Created by	Last modified
CSV Life Expectancy Data.csv	Data Asset	Chinkit Manchanda	Jun 10, 2020, 03:06 PM

Notebooks

New notebook +

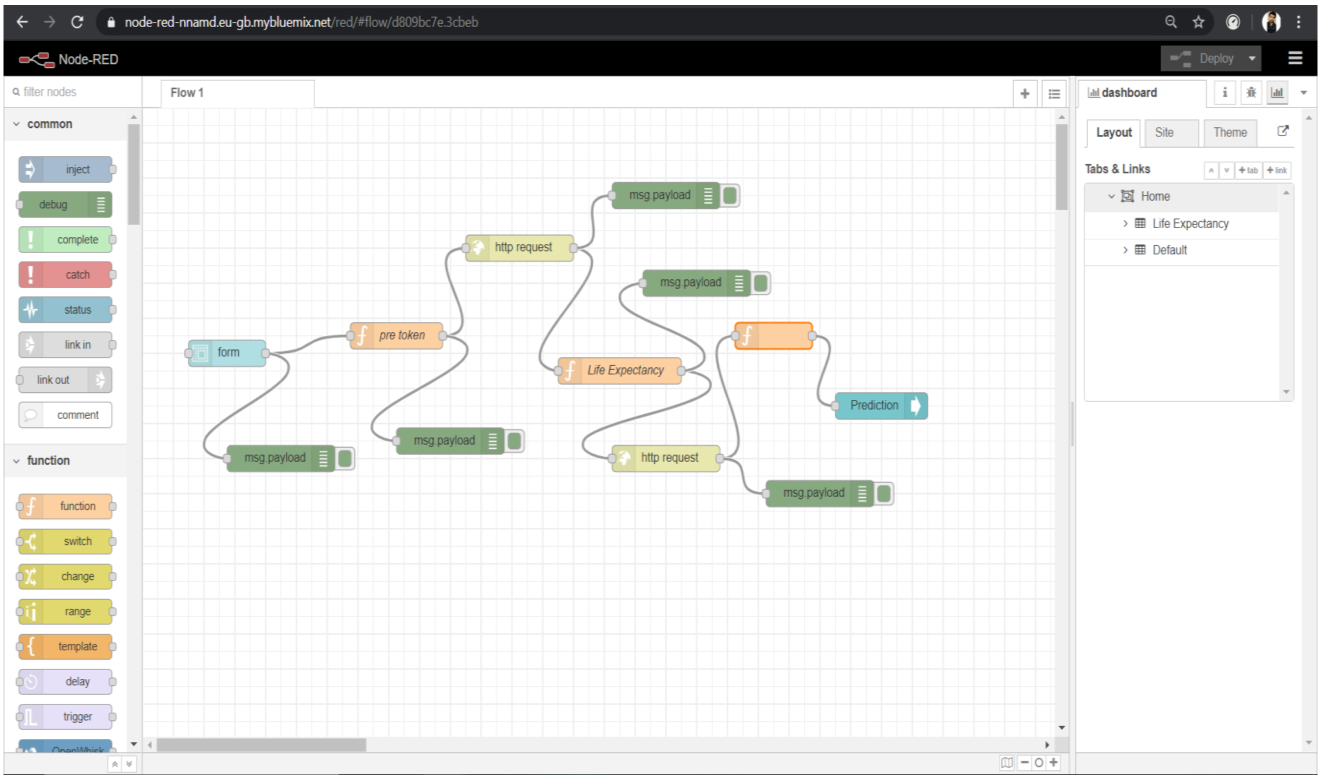
Name	Shared	Scheduled	Status	Language	Last editor	Last modified
Predicting_life_expectancy				Python 3.6	Chinkit Manchanda	Jun 10, 2020
Predicting_life_expectancy copy 1				Python 3.6	Chinkit Manchanda	Jun 10, 2020

Data

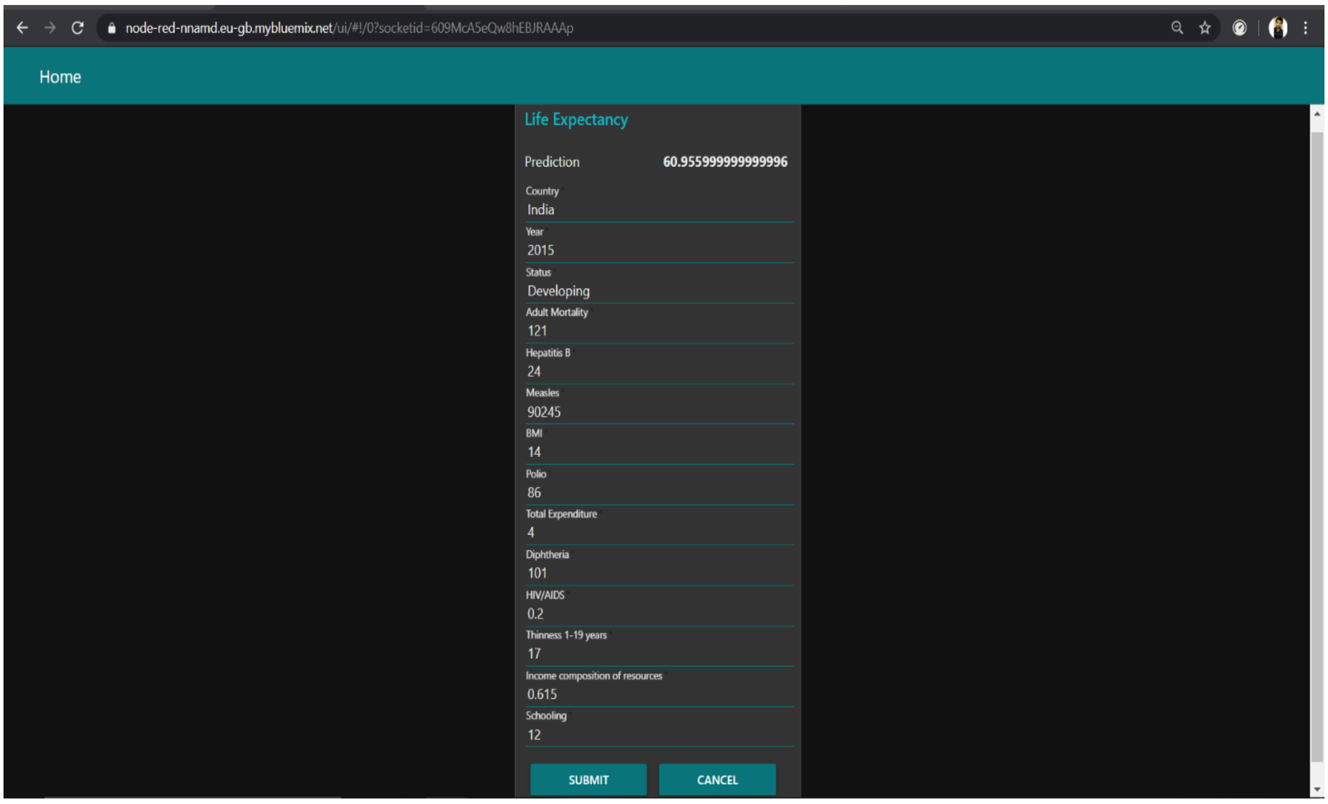
Load | Files | Catalog

Drop files here or browse for files to upload.

D)Node-Red Flow



E) Life Expectancy Prediction UI


node-red-nnamd.eu-gb.mybluemix.net/ui/#!/0?socketid=609McA5eQw8hEBJRAAAp

Home

Life Expectancy

Prediction 60.955999999999996

Country
India

Year
2015

Status
Developing

Adult Mortality
121

Hepatitis B
24

Measles
90245

BMI
14

Polio
86

Total Expenditure
4

Diphtheria
101

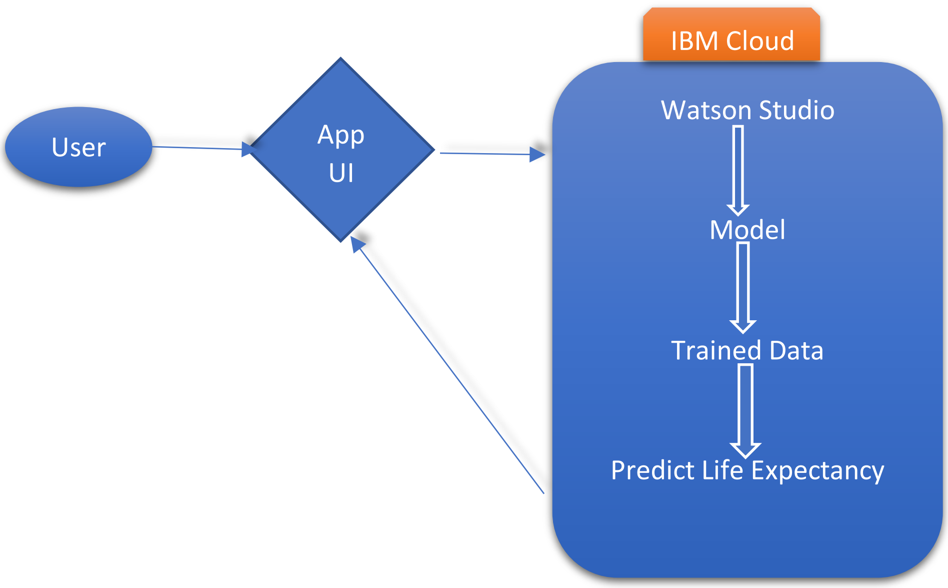
HIV/AIDS
0.2

Thinness 1-19 years
17

Income composition of resources
0.615

Schooling
12

5. FLOWCHART



a) The user input all the required values in the app

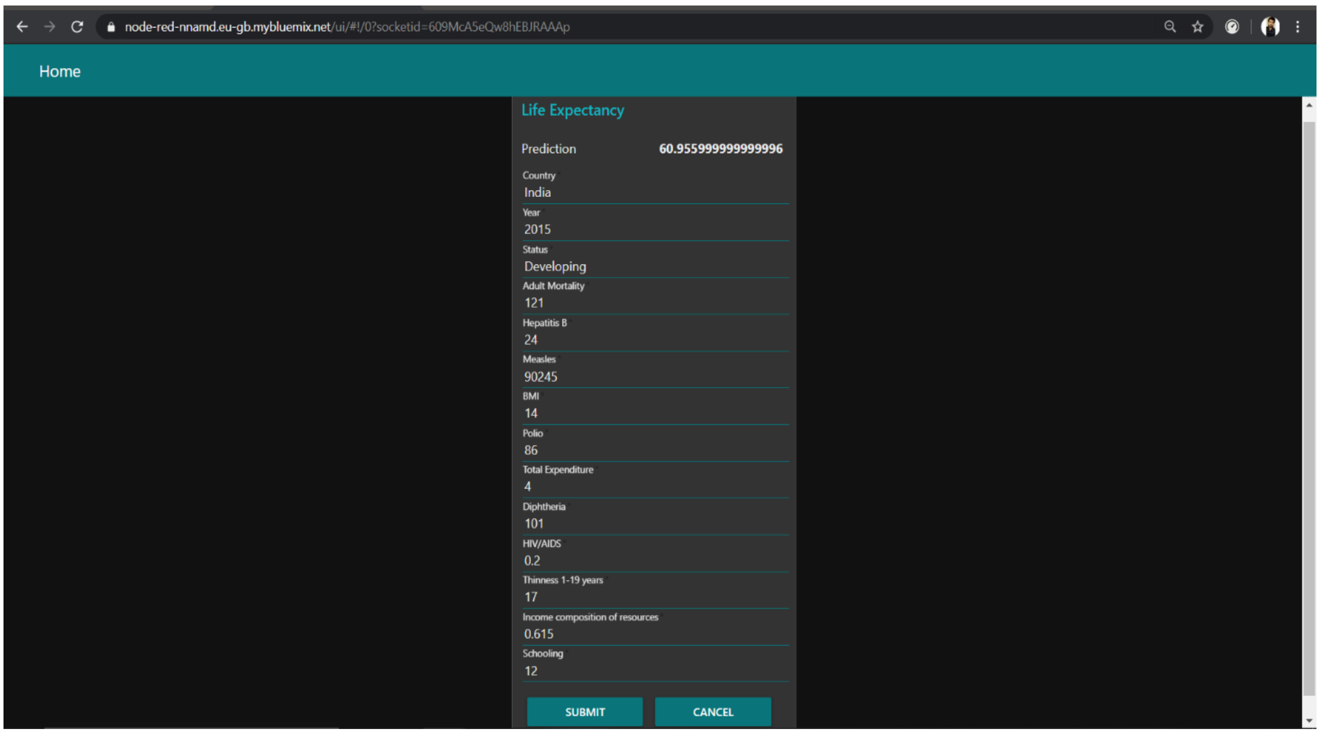
b) The data then entered into Watson and the scoring_endpoint URL matches with the deployed model.

c) Then it enters into trained data and predict the life expectancy value

d) The value predicted is prompted in the app screen.

6. RESULT

This is the Life Expectancy UI



node-red-nnamd.eu-gb.mybluemix.net/ui/#!/0?socketid=609McA5eQw8hEBJRAAAp

Home

Life Expectancy

Prediction	60.955999999999996
Country	India
Year	2015
Status	Developing
Adult Mortality	121
Hepatitis B	24
Measles	90245
BMI	14
Polio	86
Total Expenditure	4
Diphtheria	101
HIV/AIDS	0.2
Thinness 1-19 years	17
Income composition of resources	0.615
Schooling	12

7. ADVANTAGES AND DISADVANTAGES

ADVANTAGES:

- Health Inequalities: Life expectancy has been used nationally to monitor health inequalities of a country.
- 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.
- 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:

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

8. APPLICATION

- It can be used to monitor health inequalities of a country.
- It can be used to develop statistics for country development process.
- It can be used to analyse the factors for high life expectancy.
- 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:

- ❖ 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's 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.

- ❖ 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.

- ❖ Integrating with services such as speech recognition

11. BIBLIOGRAPHY

- ➢ 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://bookdown.org/caoying4work/watsonstudio-workshop/jn.html
- ➢ https://www.kaggle.com/kumarajarshi/life-expectancy-who
- ➢ https://www.youtube.com/watch?v=DBRGlAHdj48&list=PLzpeuWUENMK2PYtasCaKK4bZjaYzhW23L
- ➢ 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 pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
from sklearn.preprocessing import StandardScaler
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression
from sklearn.metrics import mean_squared_error, mean_absolute_error, r2_score
from sklearn.ensemble import RandomForestRegressor


# # Dataset

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_87f918dae33a4b2b834f7d6bedb24fa6 =
ibm_boto3.client(service_name='s3',
    ibm_api_key_id='Be6Btl_5Etl1TcaS6wV_68nfrZot5V6GwtGo5zMQHNop',
    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_87f918dae33a4b2b834f7d6bedb24fa6.get_object(Bucket='predictlifeexpect
ancy-donotdelete-pr-42scae56tsebnf',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 )

# If you are reading an Excel file into a pandas DataFrame, replace `read_csv` by
`read_excel` in the next statement.
```

```
df_data_0 = pd.read_csv(body)
df_data_0.head()
df = df_data_0
print (df.shape)

df.head()


# # Descriptive Analysis of the Data
df.rename(columns = lambda x: x.strip().replace(' ', '_').lower(), inplace=True)
df.rename(columns = {'thinness__1-19_years':'thinness_10-19_years'},
inplace=True)

df.columns

plt.figure(figsize=(25,10))
df.groupby('country')['life_expectancy'].mean().head(40).plot.bar()

df.describe()

df.isnull().sum()

#list of all columns containing NA values
na_cols =['life_expectancy', 'adult_mortality', 'alcohol', 'hepatitis_b',
       'bmi', 'polio', 'total_expenditure','diphtheria', 'gdp', 'population',
        'thinness_10-19_years', 'thinness_5-9_years',
        'income_composition_of_resources', 'schooling']

#filling NA cells with overall mean
for col in na_cols:
    df[col].fillna(df[col].mean(), inplace=True)

# Getting percentage of empty data in columns
df.isnull().sum()/len(df)*100

print(df.dtypes)

train_data=df.drop(['life_expectancy'],axis=1)
train_labels = df['life_expectancy']
train_data.shape,train_labels.shape

corr = train_data.corr()
plt.figure(figsize=(20,12))
sns.heatmap(corr,square=True,annot=True)
```

```
col_corr = []
for i in range(len(corr.columns)):
        for j in range(i):
            if (corr.iloc[i, j] >= 0.9) and (corr.columns[j] not in col_corr):
                colname = corr.columns[i] # getting the name of column
                if colname not in col_corr:
                    col_corr.append(colname)
                if colname in train_data.columns:
                    del train_data[colname]

col_corr #these columns are removed

train_data.shape

# Plotting Developed v Developing
plt.figure(figsize=(8,5))

sns.barplot(y='status', x='life_expectancy', data=df, orient='h',
            palette = ['xkcd:cranberry','xkcd:neon green'], errcolor='grey');
plt.title('Life Expectancy by Country Status', fontsize=20)
plt.xlabel('Life Expectancy')
plt.ylabel('');


# # Calculation of p-value
X = train_data.iloc[:,3:].values
y = train_labels.values

import statsmodels.api as sm

mod = sm.OLS(y,X)

fii = mod.fit()

p_values = fii.summary2().tables[1]['P>|t|']

print(p_values)

columns = train_data.columns
reduce = []
for i in range(len(p_values)):
    if(p_values[i]>0.05):
        print(columns[i+3],p_values[i])
        reduce.append(columns[i+3])
```

```
#drop columns with higher p_values
train_data=train_data.drop(reduce,axis=1)

columns = train_data.columns
print(columns)

train_data = train_data.drop(['country'],axis=1)

# # MultiLinear Regression

X = train_data.iloc[:,:].values
y = train_labels.values

from sklearn.preprocessing import LabelEncoder
# labelencoder_X = LabelEncoder()
# X[:,0] = labelencoder_X.fit_transform(X[:,0])
labelencoder_X_2 = LabelEncoder()
X[:,1] = labelencoder_X_2.fit_transform(X[:,1])

X.shape

from sklearn.model_selection import train_test_split
X_train,X_test,y_train,y_test = train_test_split(X,y,test_size=0.2)

from sklearn.linear_model import LinearRegression
reg = LinearRegression()
reg.fit(X_train, y_train)

y_pred = reg.predict(X_test)
y_pred[:10]

#mse
mse=mean_squared_error(y_test, y_pred)
#r2
r2=r2_score(y_test,y_pred)
#mae
mae=mean_absolute_error(y_test,y_pred)
#rmse
rmse=np.sqrt(mae)
#To retrieve the intercept
intercept=reg.intercept_
#For retrieving the slope
coef = reg.coef_
{'mse':[mse],'r2':[r2],'mae':[mae],'rmse':[rmse],'intercept':[intercept],'coef':coef[0]}
```

```
#accuracy
error = []
for i in range(len(y_test)):
    a=abs(y_test[i]-y_pred[i])
    error.append(a)
acc=100-(sum(error)/len(y_test))

round(acc,2)


# # Decision Tree
from sklearn.tree import DecisionTreeRegressor
regres = DecisionTreeRegressor()
regres.fit(X_train, y_train)

y_pred = reg.predict(X_test)

#mse
mse=mean_squared_error(y_test, y_pred)
#r2
r2=r2_score(y_test,y_pred)
#mae
mae=mean_absolute_error(y_test,y_pred)
#rmse
rmse=np.sqrt(mae)
#To retrieve the intercept
intercept=reg.intercept_
#For retrieving the slope
coef = reg.coef_
{'mse':[mse],'r2':[r2],'mae':[mae],'rmse':[rmse],'intercept':[intercept],'coef':coef[0]}

error = []
for i in range(len(y_test)):
    a=abs(y_test[i]-y_pred[i])
    error.append(a)
acc=100-(sum(error)/len(y_test))

round(acc,2)


# # Random Forest Regression
reger = RandomForestRegressor(n_estimators= 50, random_state = 0)
reger.fit(X_train,y_train)
```

```
y_pred = reger.predict(X_test)

#mse
mse=mean_squared_error(y_test, y_pred)
#r2
r2=r2_score(y_test,y_pred)
#mae
mae=mean_absolute_error(y_test,y_pred)
#rmse
rmse=np.sqrt(mae)
{'mse':[mse],'r2':[r2],'mae':[mae],'rmse':[rmse]}

#accuracy
error = []
for i in range(len(y_test)):
    a=abs(y_test[i]-y_pred[i])
    error.append(a)
acc=100-(sum(error)/len(y_test))

acc

X = pd.DataFrame(X)
X.columns = [ 'year', 'status', 'adult_mortality', 'hepatitis_b',
       'measles', 'bmi', 'polio', 'total_expenditure', 'diphtheria',
       'hiv/aids', 'thinness_10-19_years', 'income_composition_of_resources',
       'schooling']

# Get Feature Importance from the classifier
feature_importance = reger.feature_importances_
print (reger.feature_importances_)
feat_importances = pd.Series(reger.feature_importances_, index=X.columns)
feat_importances = feat_importances.nlargest(19)
feat_importances.plot(kind='barh' , figsize=(10,10))

# # Save and Deploy Model
from watson_machine_learning_client import WatsonMachineLearningAPIClient
wml_credentials = {
  "apikey": "XXXXXX",
  "iam_apikey_description": "XXXXX",
  "iam_apikey_name": "Service credentials-1",
  "iam_role_crn": "XXXXX",
  "iam_serviceid_crn": "XXXXX",
  "instance_id": "XXXXX",
  "url": "XXXXX"
}

client = WatsonMachineLearningAPIClient(wml_credentials)
print(client.version)

client.repository.list_models()
client.deployments.list()

meta_props={client.repository.ModelMetaNames.NAME: "Random Forest to
predict life expectancy"}
published_model = client.repository.store_model(model=reger,
meta_props={client.repository.ModelMetaNames.NAME: "Random Forest to
predict life expectancy"})

# new list of models
client.repository.list_models()

# get UID of our just stored model
model_uid = client.repository.get_model_uid(published_model)
print("Model id: {}".format(model_uid))

# create deployment
created_deployment = client.deployments.create(model_uid,
name="life_expectancy_model_deploy")

# new list of deployments
client.deployments.list()

# get UID of our new deployment
deployment_uid = client.deployments.get_uid(created_deployment)
print("Deployment id: {}".format(deployment_uid))
print(created_deployment)

# get scoring end point
scoring_endpoint = client.deployments.get_scoring_url(created_deployment)
print(scoring_endpoint)

# use our WML client to score our model

# add some test data
scoring_payload = {'fields': [ 'year', 'status', 'adult_mortality', 'hepatitis_b',
    'measles', 'bmi', 'polio', 'total_expenditure', 'diphtheria',
    'hiv/aids', 'thinness_10-19_years', 'income_composition_of_resources',
    'schooling'],'values': [[2010,0,121,94,17,59.8,95,6.55,94,0.1,1,0.802,16.8]]}
```

```
# score the model
predictions = client.deployments.score(scoring_endpoint, scoring_payload)
print(predictions)
```