SMARTBRIDGE
Let's Bridge the Gap

PROJECT REPORT

PREDICTING LIFE EXPECTANCY

Name: Pradhuman Gupta

E-Mail: pradhumangupta099@gmail.com

Category: Machine Learning

College: HMR Institute Of Techynology And Management

Webpage Link:

https://node-red-qrqvq.eu-gb.mybluemix.net/ui/#!/6?socketid=ycf9xjGZr1hLtTjdAAAy

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 analyze further requirements to increase its rate of growth or stabilize 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: Pradhuman Gupta
Project Duration: 1 Month

1.2) PURPOSE

The result of this life expectancy should not be interpreted as definitive. Actual longevity is based on many factors, not all of which are captured here. This will ask about your **illness** such as **HIV/AIDS** and **POLIO**, **Age**, **Region,** or **Country** you belong to, consumes **Alcohol** or **Not**, **Education,** and **Income composition.** The results are based on **Statistical Regression**. This will predict Your Age when you will Die.

2.) LITERATURE SURVEY

2.1) EXISTING PROBLEM

A typical Regression Machine Learning project leverages historical data to predict insights into the future. This problem statement is aimed at predicting Life Expectancy rate of a country given various features.

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.

2.2) PROPOSED SOLUTION

STEPS:

a) Create IBM cloud services

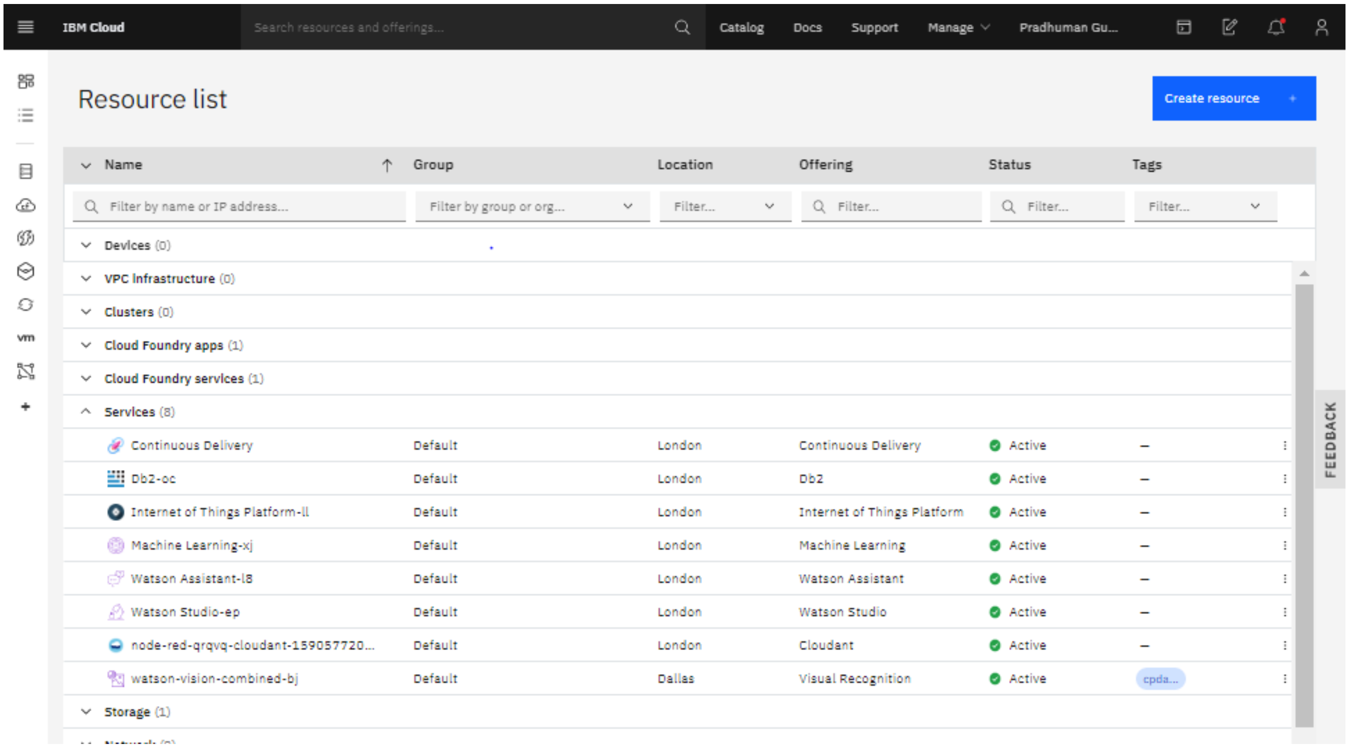
b) Configure Watson Studio

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
- Node-RED



Resource list

Name	Group	Location	Offering	Status	Tags
Devices (0)					
VPC infrastructure (0)					
Clusters (0)					
Cloud Foundry apps (1)					
Cloud Foundry services (1)					
Services (8)					
Continuous Delivery	Default	London	Continuous Delivery	Active	–
Db2-oc	Default	London	Db2	Active	–
Internet of Things Platform-ll	Default	London	Internet of Things Platform	Active	–
Machine Learning-xj	Default	London	Machine Learning	Active	–
Watson Assistant-l8	Default	London	Watson Assistant	Active	–
Watson Studio-ep	Default	London	Watson Studio	Active	–
node-red-qrqvq-cloudant-159057720...	Default	London	Cloudant	Active	–
watson-vision-combined-bj	Default	Dallas	Visual Recognition	Active	cpda...
Storage (1)					

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.

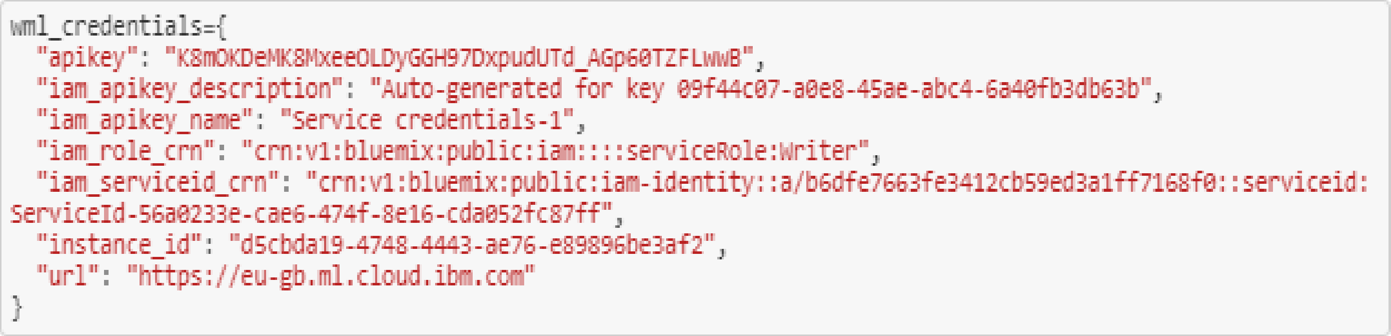
Then add dataset and empty jupyter notebook into Assets.
After that go to notebook and write your code to build model and get the scoring endpoint URL.

STEPS FOR NOTEBOOK:

- Install Watson_Machine_Learning_Client
- Import necessary Libraries
- Import DataSet.
- Data Processing
  - Removing unusual species in column names using rename function.
  - Replacing NAN values with their Mean values.
- Exploratory Data Analysis
  - Plotting **Pair plot** for analysing pairwise relationship among features.

  - Ploting a **HEATMAP** to check nif Dimensional Reduction can be Performed.

- 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 train and test split was peformed and 80% of dataset were trained data and 20% were test data.
- Linear Regression
  - In statistics, **linear regression** is a **linear** approach to modeling the relationship between a scalar response (or dependent variable) and one or more explanatory variables (or independent variables). The case of one explanatory variable is called simple **linear regression**.
- Model Building and Deployment
  - At first the machine learning service credentials was stored in a variable and passed into WatsonMachineLearningAPIClient .



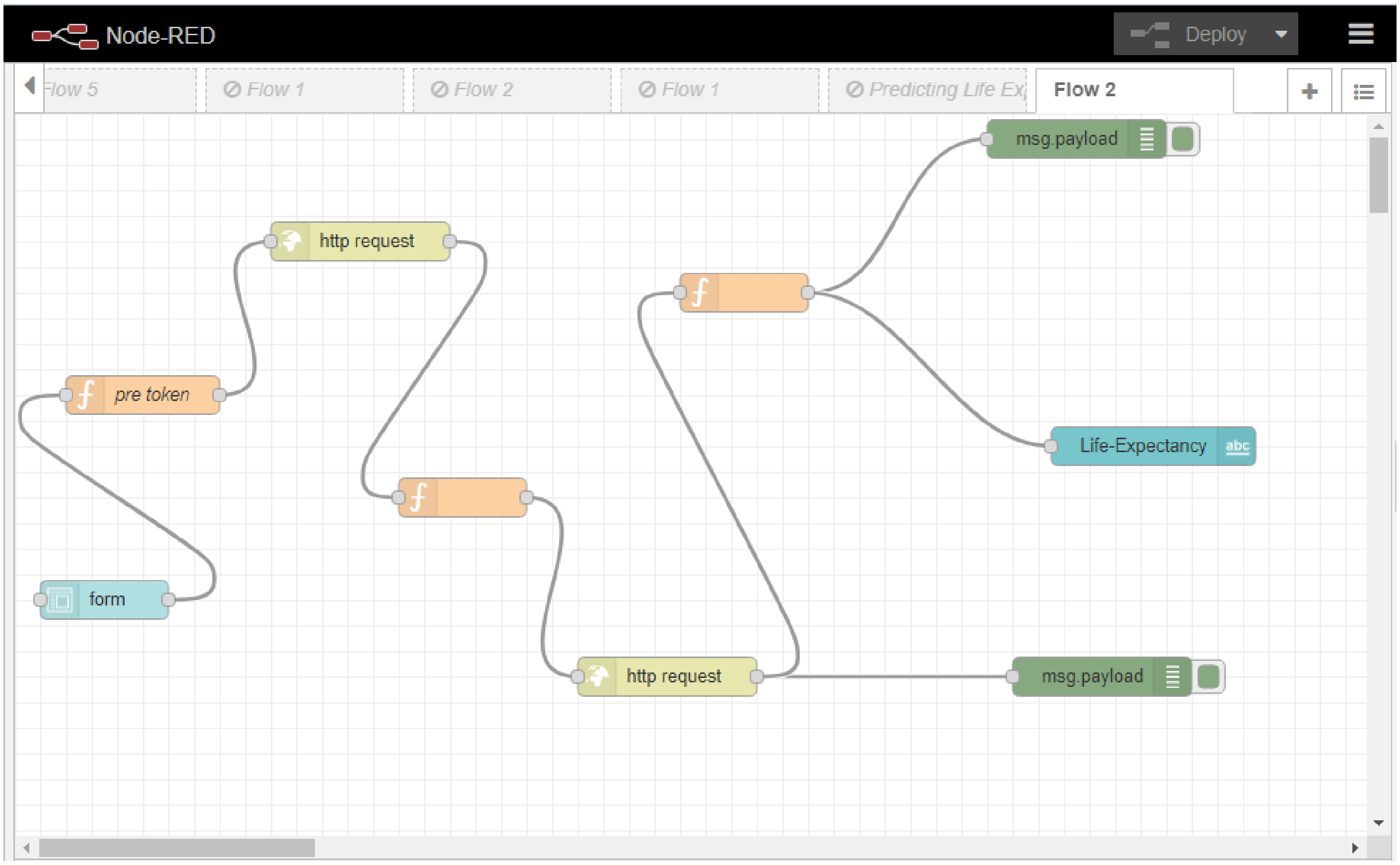
```
wml_credentials={
  "apikey": "K8mOKDeMK8MxeeOLDyGGH97DxpudUTd_AGp60TZFLwwB",
  "iam_apikey_description": "Auto-generated for key 09f44c07-a0e8-45ae-abc4-6a40fb3db63b",
  "iam_apikey_name": "Service credentials-1",
  "iam_role_crn": "crn:v1:bluemix:public:iam::::serviceRole:Writer",
  "iam_serviceid_crn": "crn:v1:bluemix:public:iam-identity::a/b6dfe7663fe3412cb59ed3a1ff7168f0::serviceid:
ServiceId-56a0233e-cae6-474f-8e16-cda052fc87ff",
  "instance_id": "d5cbda19-4748-4443-ae76-e89896be3af2",
  "url": "https://eu-gb.ml.cloud.ibm.com"
}
```

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

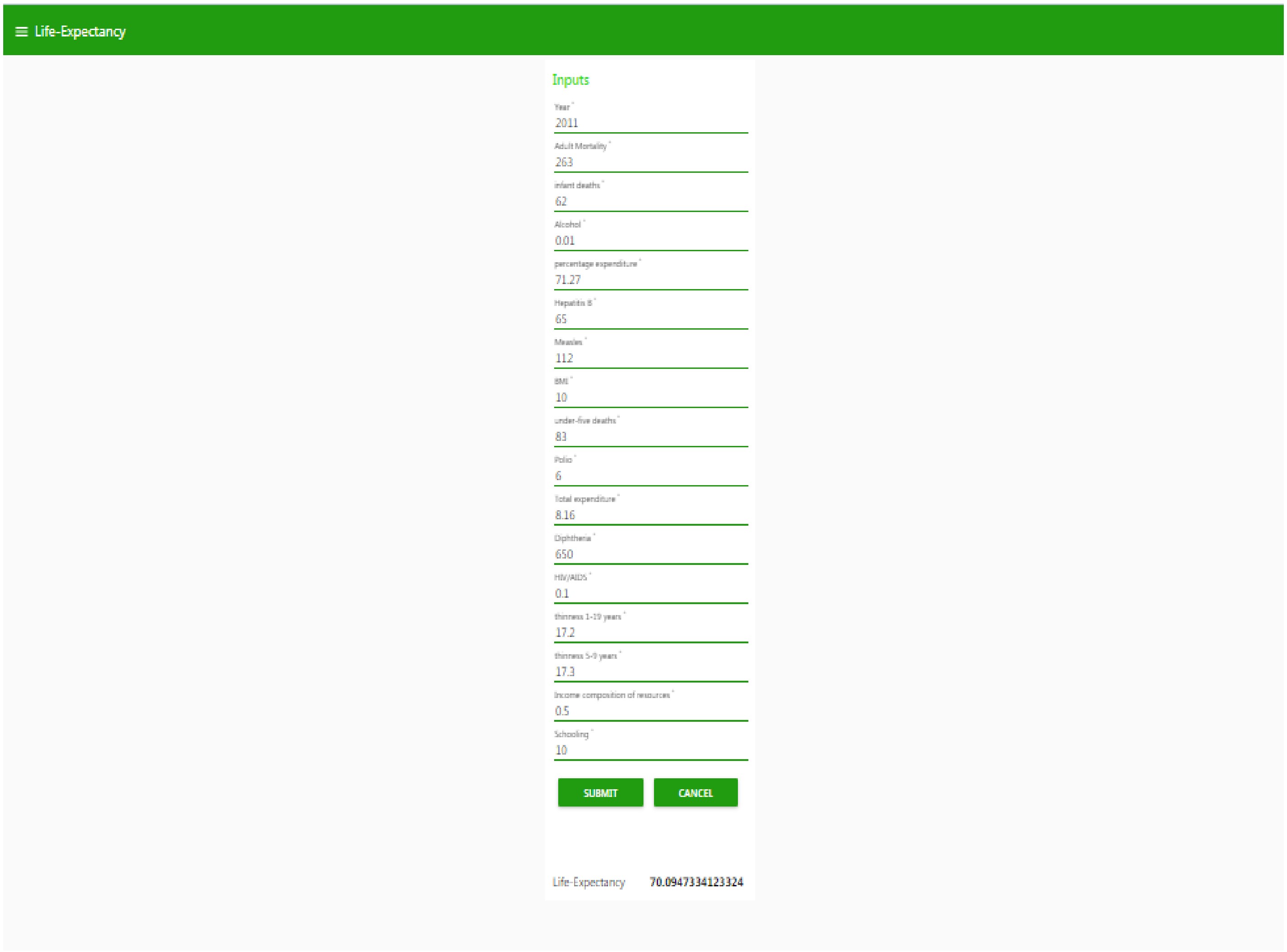
2.2.3) <u>Create Node-RED Flow to Connect all Services together</u>

- Go to Node-RED Editor from Resource List.
- Install Node-RED Dashboard from Manage Pallete.
- Now Create the Flow With the Help of Following Flow:
  - Inject

- UI_Form
- Function
- Http_Request
- Debug
- 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.



Inputs

Year *
2011

Adult Mortality *
263

infant deaths *
62

Alcohol *
0.01

percentage expenditure *
71.27

Hepatitis B *
65

Measles *
112

BMI *
10

under-five deaths *
83

Polio *
6

Total expenditure *
8.16

Diphtheria *
650

HIV/AIDS *
0.1

thinness 1-19 years *
17.2

thinness 5-9 years *
17.3

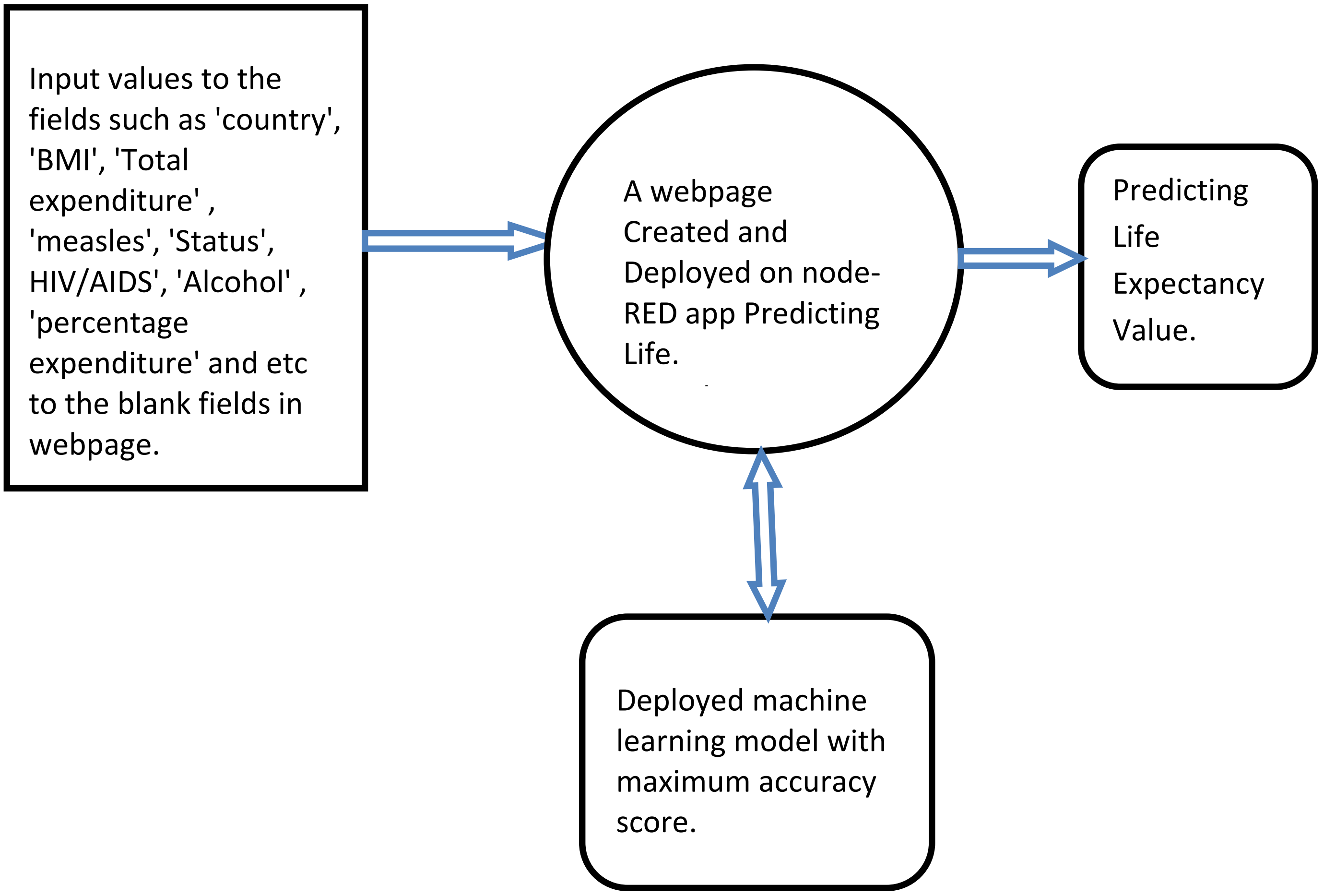
Income composition of resources *
0.5

Schooling *
10

Life-Expectancy 70.0947334123324

3.) THEORETICAL ANALYSIS

3.1) BLOCK DIAGRAM



Input values to the fields such as 'country', '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.

Predicting Life Expectancy Value.

Deployed machine learning model with maximum accuracy score.

3.2) HARDWARE/SOFTWARE DESIGNING

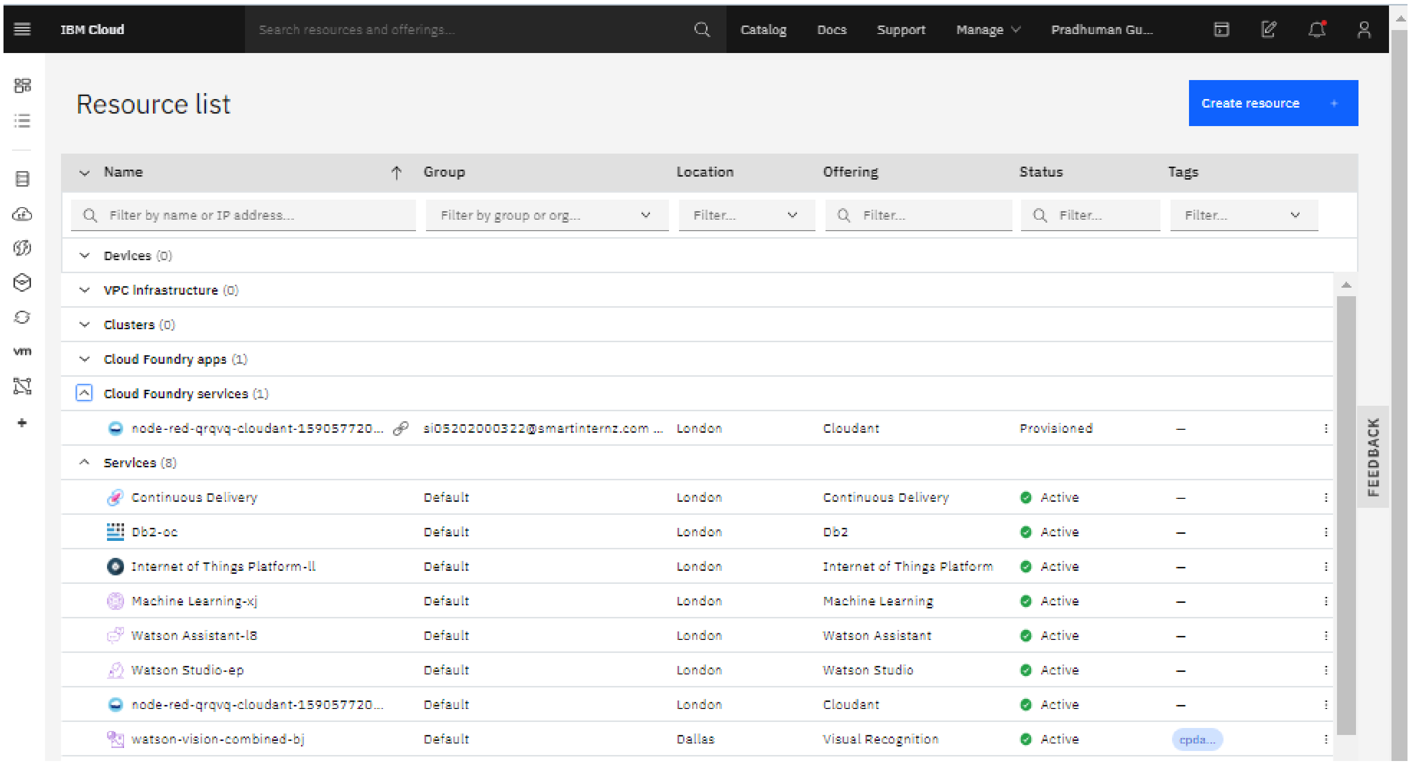
- **PROJECT REQUIREMENTS**
    - Python
    - IBM Cloud
    - IBM Watson
- **FUNCTIONAL REQUIREMENTS**
    - IBM Cloud
- **TECHNICAL REQUIREMENTS**

- ➢ Python
- ➢ IBM Watson
- ➢ IBM Cloud
- ➢ Machine Learning
- ➢ AutoAI

- **SOFTWARE REQUIREMENTS**
  - ➢ IDLE (Python 3.8)
  - ➢ Jupyter Notebook
  - ➢ IBM Cloud
  - ➢ IBM Watson

4.) EXPERIMENTAL INVESTIGATIONS

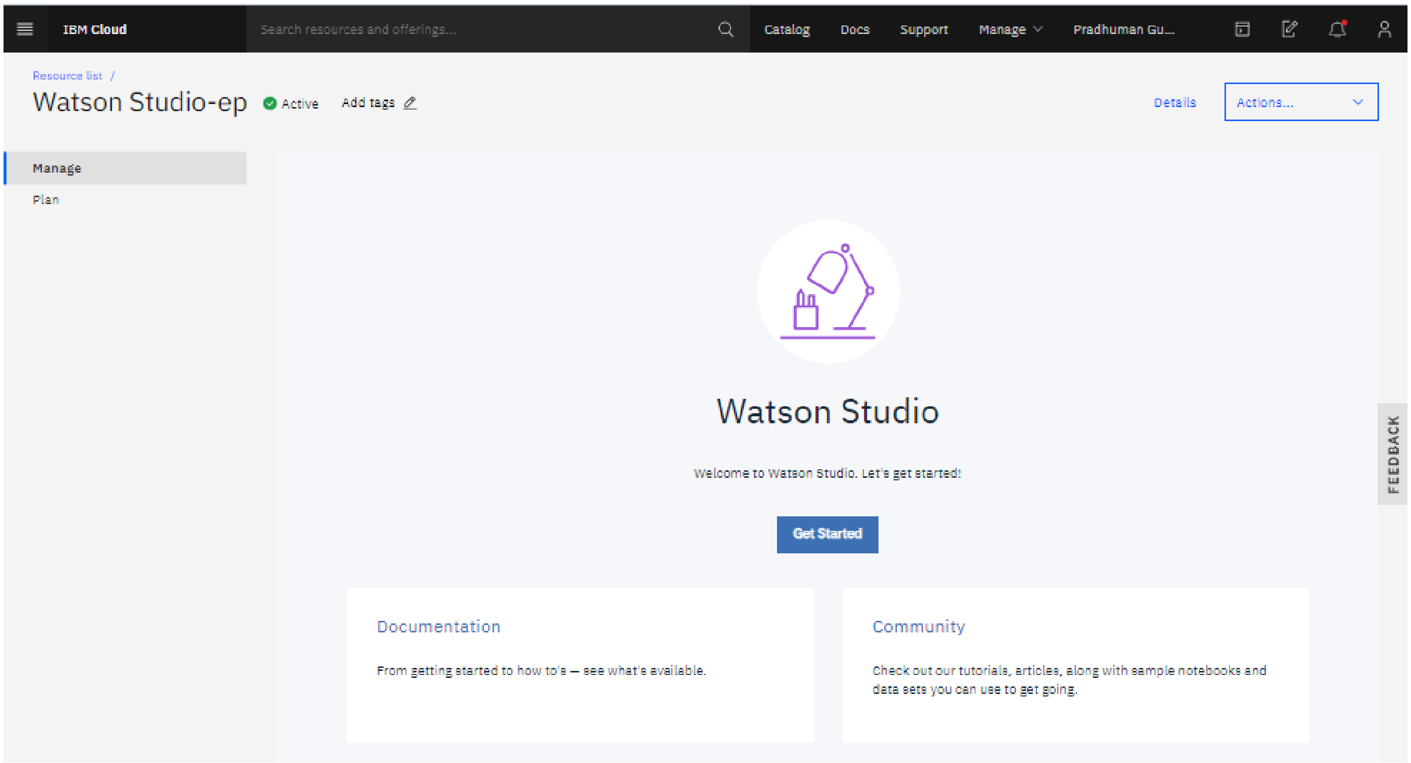
IBM CLOUD RESOURCE LIST



Resource list

Name	Group	Location	Offering	Status	Tags
Filter by name or IP address...	Filter by group or org...	Filter...	Filter...	Filter...	Filter...
Devices (0)					
VPC Infrastructure (0)					
Clusters (0)					
Cloud Foundry apps (1)					
Cloud Foundry services (1)					
node-red-qrqvq-cloudant-159057720...	si05202000322@smartinternz.com ...	London	Cloudant	Provisioned	–
Services (8)					
Continuous Delivery	Default	London	Continuous Delivery	Active	–
Db2-oc	Default	London	Db2	Active	–
Internet of Things Platform-ll	Default	London	Internet of Things Platform	Active	–
Machine Learning-xj	Default	London	Machine Learning	Active	–
Watson Assistant-l8	Default	London	Watson Assistant	Active	–
Watson Studio-ep	Default	London	Watson Studio	Active	–
node-red-qrqvq-cloudant-159057720...	Default	London	Cloudant	Active	–
watson-vision-combined-bj	Default	Dallas	Visual Recognition	Active	cpda...

IBM WATSON STUDIO



Watson Studio-ep ● Active Add tags

Watson Studio

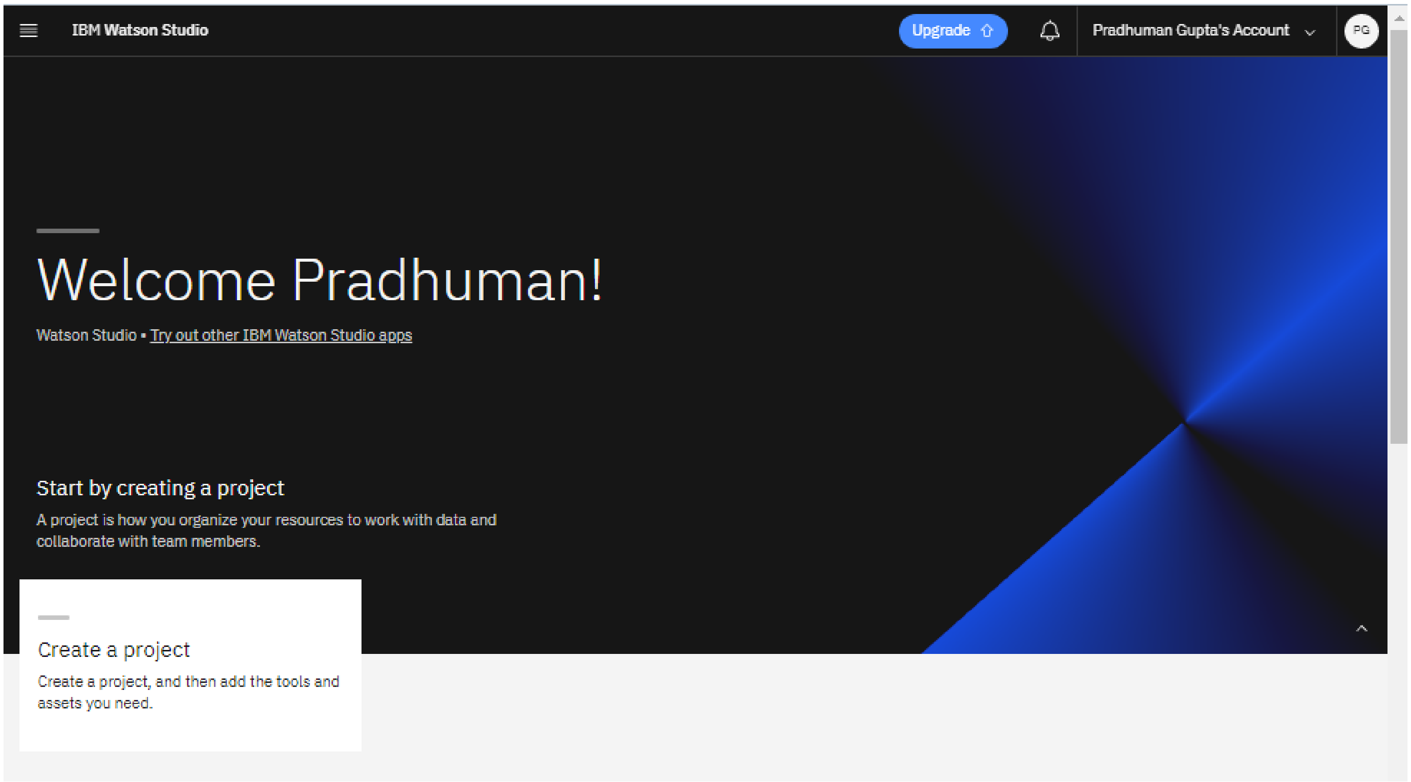
Welcome to Watson Studio. Let's get started!

Documentation

From getting started to how to's — see what's available.

Community

Check out our tutorials, articles, along with sample notebooks and data sets you can use to get going.



Welcome Pradhuman!

Watson Studio • Try out other IBM Watson Studio apps

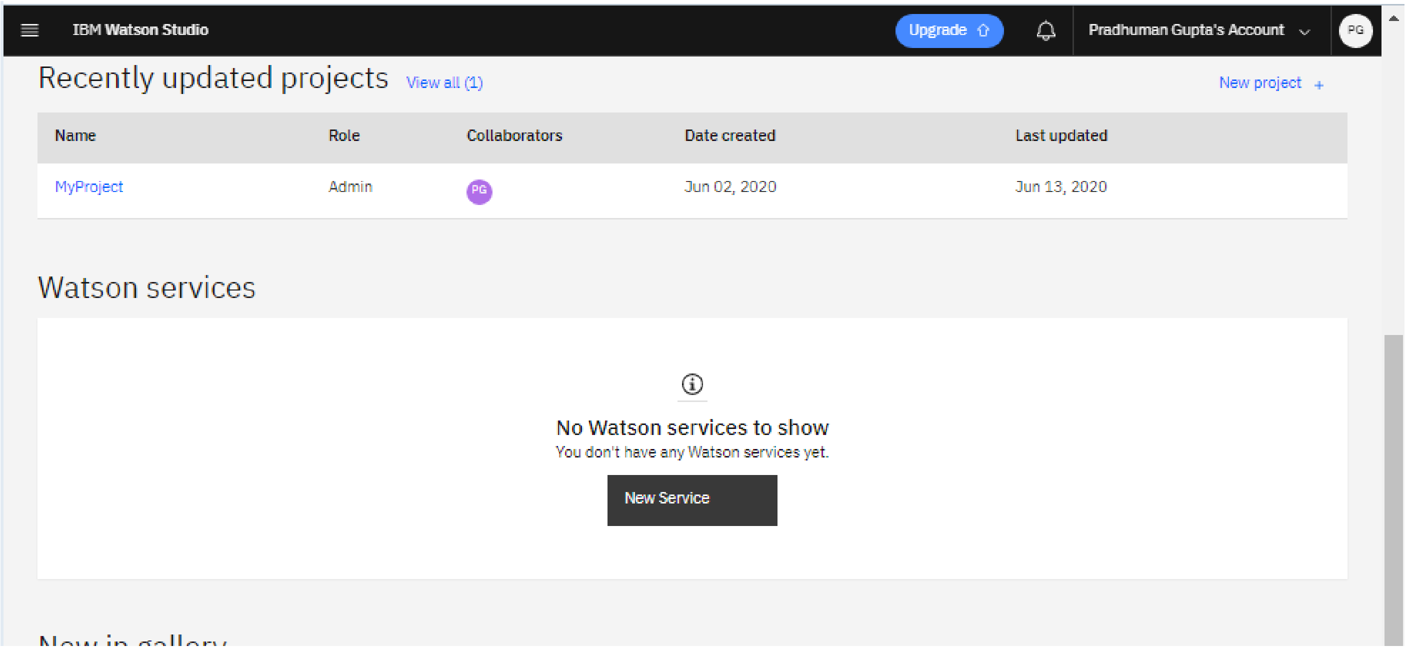
Start by creating a project

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.

IBM CLOUD PROJECT DETAILS



IBM Watson Studio Upgrade Pradhuman Gupta's Account PG

Recently updated projects View all (1) New project +

Name	Role	Collaborators	Date created	Last updated
MyProject	Admin	PG	Jun 02, 2020	Jun 13, 2020

Watson services

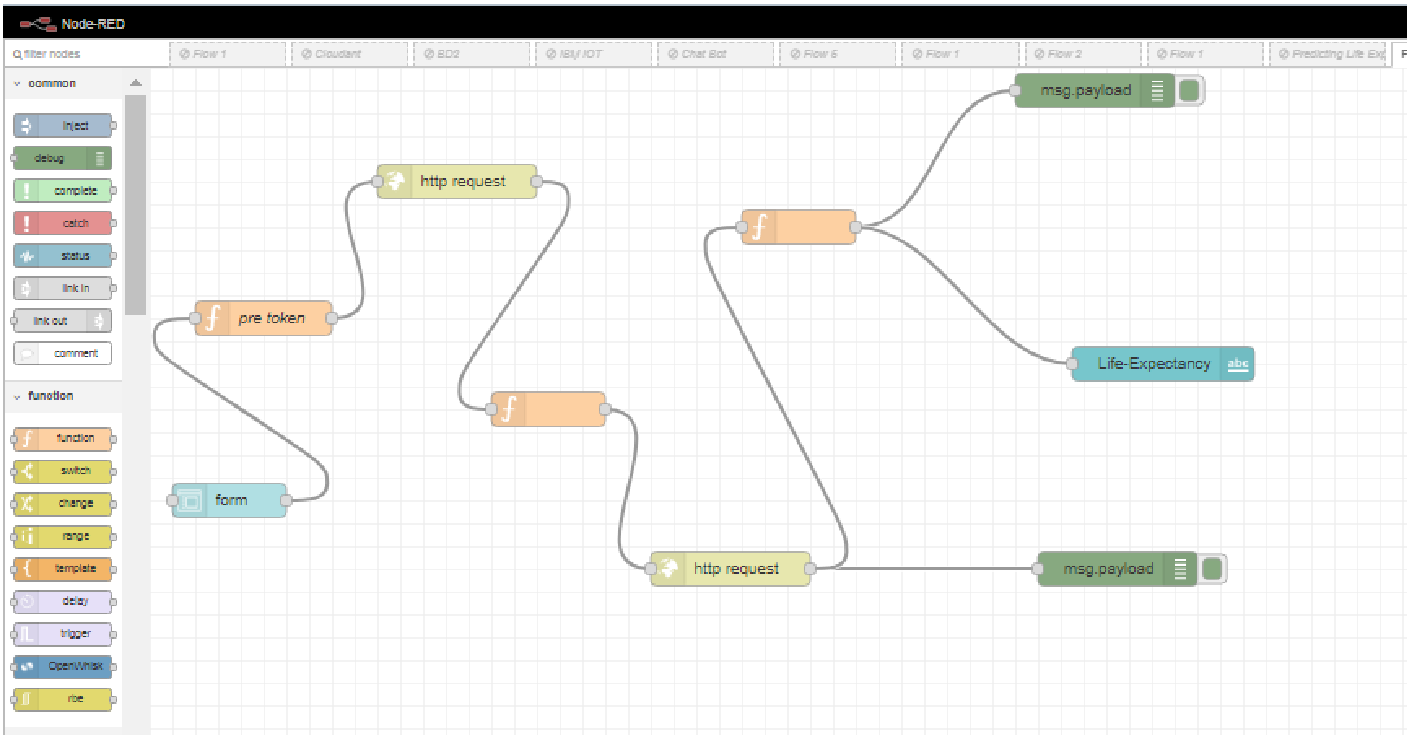
No Watson services to show

You don't have any Watson services yet.

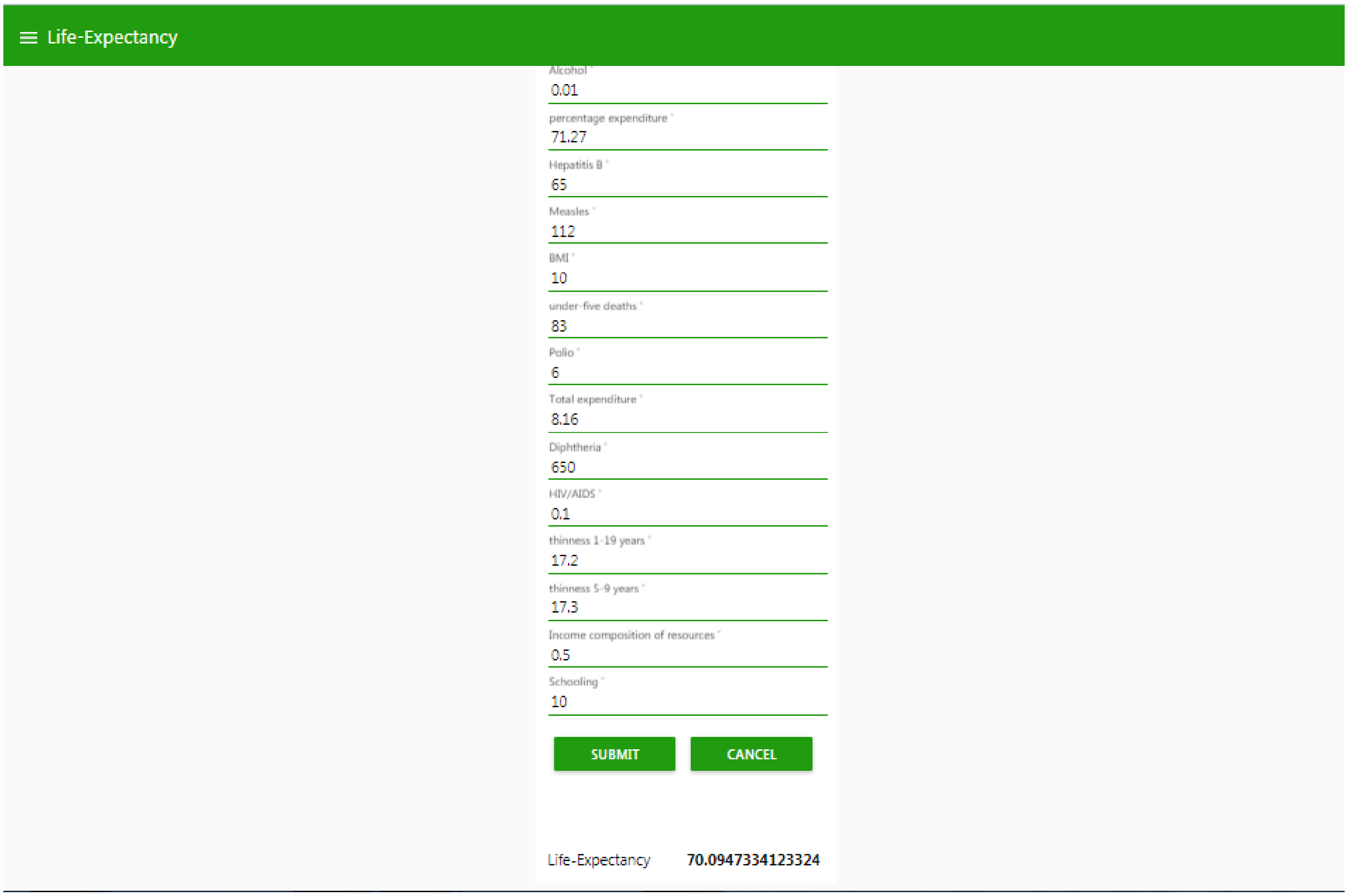
New Service

New in gallery

Node-RED FLOW



LIFE EXPECTANCY PREDICTION UI



Alcohol *
0.01

percentage expenditure *
71.27

Hepatitis B *
65

Measles *
112

BMI *
10

under-five deaths *
83

Polio *
6

Total expenditure *
8.16

Diphtheria *
650

HIV/AIDS *
0.1

thinness 1-19 years *
17.2

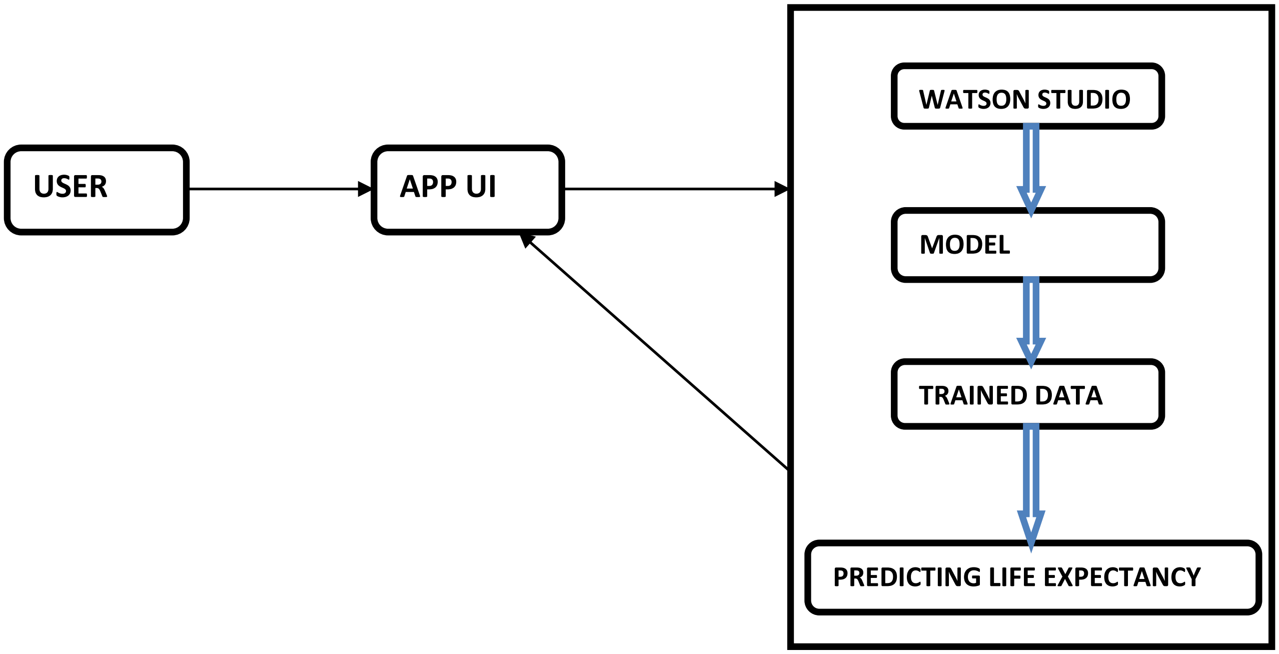
thinness 5-9 years *
17.3

Income composition of resources *
0.5

Schooling *
10

Life-Expectancy 70.0947334123324

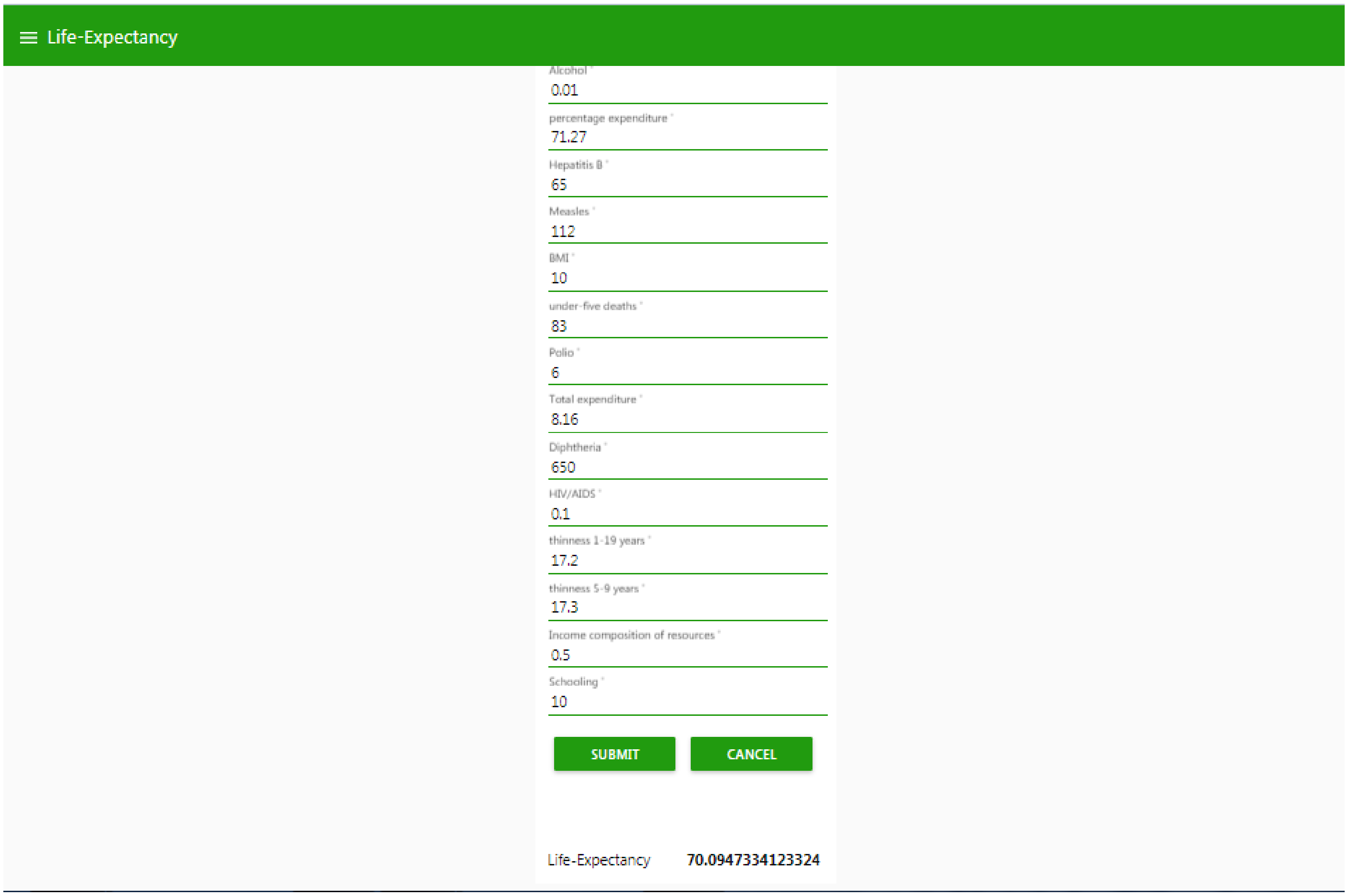
5.) FLOW CHART



- The **USER** input all the Required Values in the App.
- The **Data** will Enter into **Watson** and the **Scoring_Endpoint URL** matches with the **Deployed Model**.
- Then it Enters into the **Trained Data** and Predict **The Life Expectancy** Value.
- The Value predicted is opted into the **App** Screen.

6.) RESULT

This is the **LIFE EXPECTANCY UI**.



Alcohol *
0.01

percentage expenditure *
71.27

Hepatitis B *
65

Measles *
112

BMI *
10

under-five deaths *
83

Polio *
6

Total expenditure *
8.16

Diphtheria *
650

HIV/AIDS *
0.1

thinness 1-19 years *
17.2

thinness 5-9 years *
17.3

Income composition of resources *
0.5

Schooling *
10

Life-Expectancy **70.0947334123324**

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.) APPLICATIONS

- 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 datas 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://www.kaggle.com/kumarajarshi/life-expectancy-who
- https://www.youtube.com/watch?v=DBRGlAHdj48&list=PLzpeuWUENMK2PYtasCaKK4bZjaYzhW23L
- https://www.youtube.com/watch?v=Jtej3Y6uUng
- https://bookdown.org/caoying4work/watsonstudio-workshop/jn.html#deploy-model-as-web-service
- https://bookdown.org/caoying4work/watsonstudio-workshop/auto.html#add-asset-as-auto-ai
- https://www.youtube.com/watch?v=LOCkV-mENq8&feature=youtu.be
- https://developer.ibm.com/tutorials/how-to-create-a-node-red-starter-application/

APPENDIX

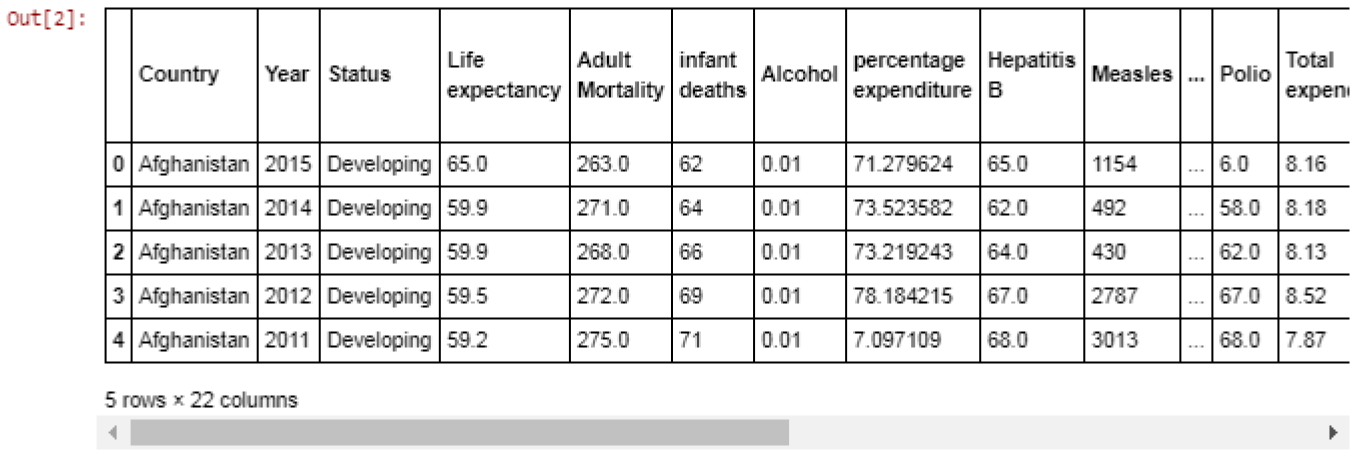
SOURCE CODE:

#Importing Libraries

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

#Importing 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_2c4b212ecf374fa692c761ecd7f72bbb =
ibm_boto3.client(service_name='s3',
    ibm_api_key_id='N2YTIh93Yfvlw61IhV1f-i_lmclF9FQpPBQCQ9xuwYPY',
    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_2c4b212ecf374fa692c761ecd7f72bbb.get_object(Bucket='myproject-
donotdelete-pr-hzgogmf9qpe5kg',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 = pd.read_csv(body)
df.head()
```



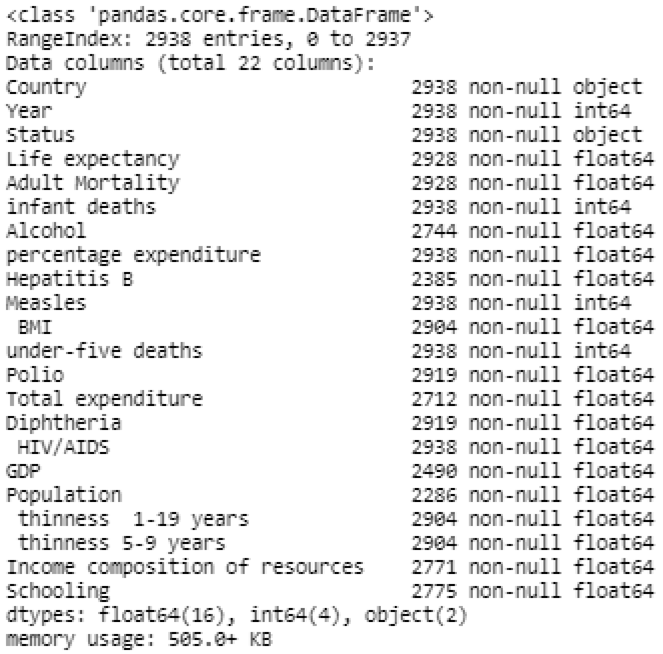
Out[2]:

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

5 rows × 22 columns

# Extracting Information From DataSet Using DataFrame

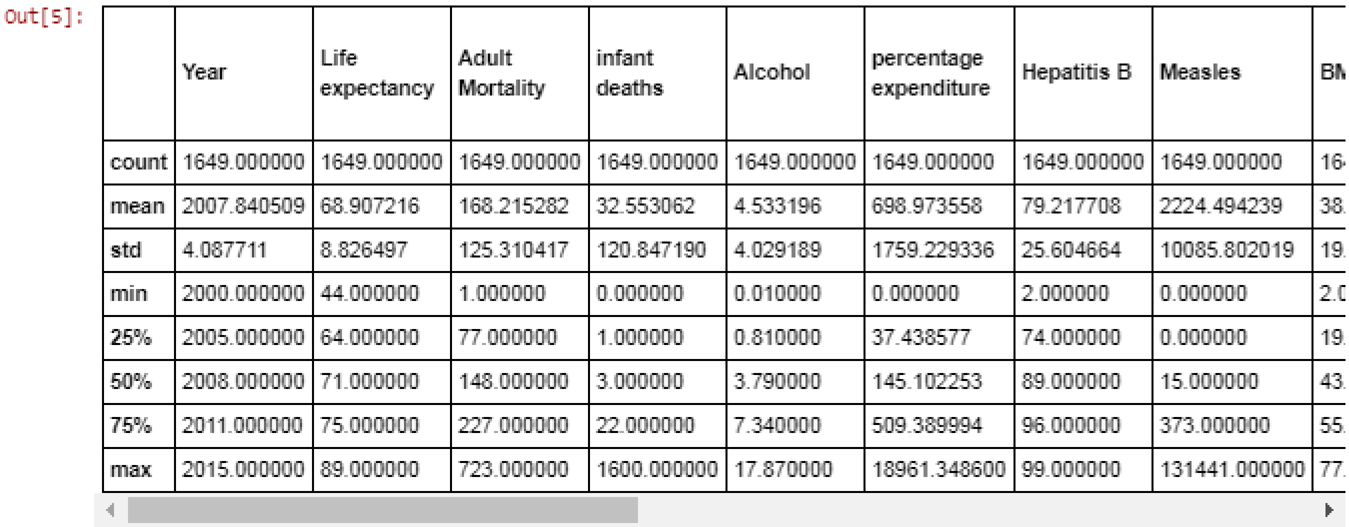
```
df.info()
```



```
<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
```

```
df.dropna(inplace=True) df['Life expectancy ']=df['Life expectancy
'].astype(int,copy=True)
```

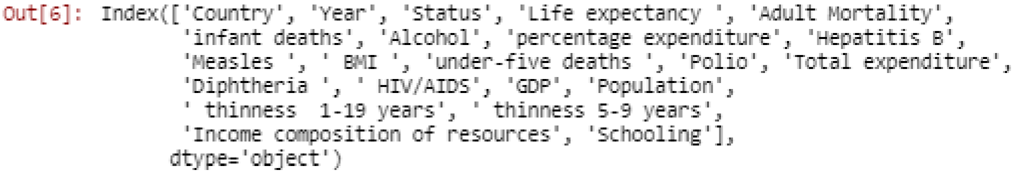
```
df.describe()
```



Out[5]:

	Year	Life expectancy	Adult Mortality	infant deaths	Alcohol	percentage expenditure	Hepatitis B	Measles	BM
count	1649.000000	1649.000000	1649.000000	1649.000000	1649.000000	1649.000000	1649.000000	1649.000000	16
mean	2007.840509	68.907216	168.215282	32.553062	4.533196	698.973558	79.217708	2224.494239	38.
std	4.087711	8.826497	125.310417	120.847190	4.029189	1759.229336	25.604664	10085.802019	19.
min	2000.000000	44.000000	1.000000	0.000000	0.010000	0.000000	2.000000	0.000000	2.0
25%	2005.000000	64.000000	77.000000	1.000000	0.810000	37.438577	74.000000	0.000000	19.
50%	2008.000000	71.000000	148.000000	3.000000	3.790000	145.102253	89.000000	15.000000	43.
75%	2011.000000	75.000000	227.000000	22.000000	7.340000	509.389994	96.000000	373.000000	55.
max	2015.000000	89.000000	723.000000	1600.000000	17.870000	18961.348600	99.000000	131441.000000	77.

```
df.columns
```

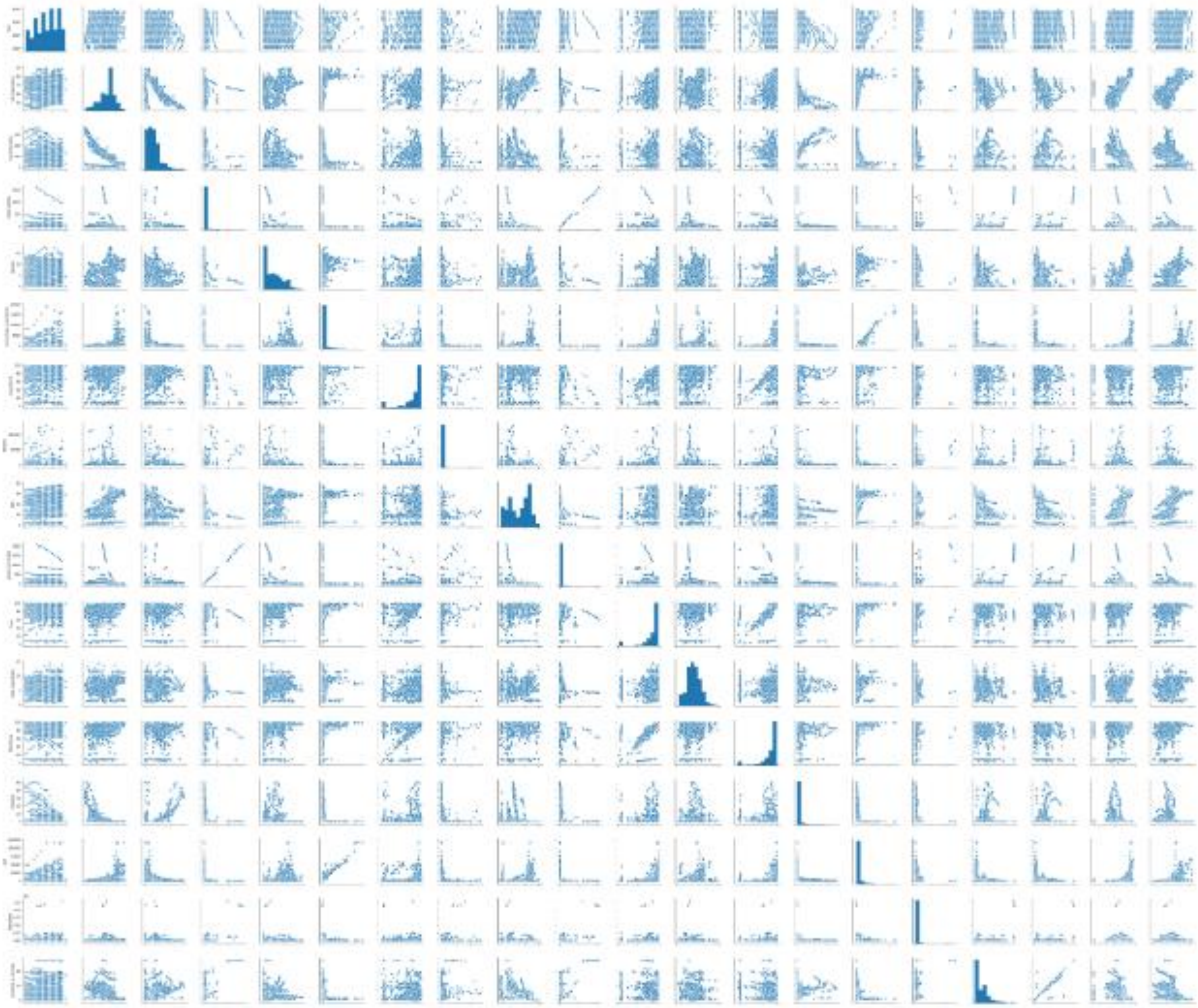


```
Out[6]: Index(['Country', 'Year', 'Status', 'Life expectancy ', 'Adult Mortality',
               'infant deaths', 'Alcohol', 'percentage expenditure', 'Hepatitis B',
               'Measles ', ' BMI ', 'under-five deaths ', 'Polio', 'Total expenditure',
               'Diphtheria ', ' HIV/AIDS', 'GDP', 'Population',
               ' thinness  1-19 years', ' thinness 5-9 years',
               'Income composition of resources', 'Schooling'],
              dtype='object')
```

#DATA VISUALIZING USING SEABORN

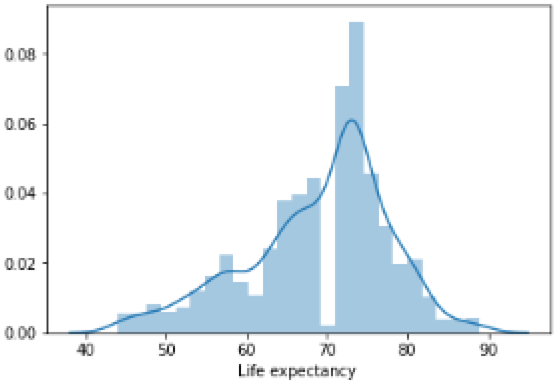
```
sns.pairplot(df)
```

```
Out[7]: <seaborn.axisgrid.PairGrid at 0x7f228cc9c128>
```



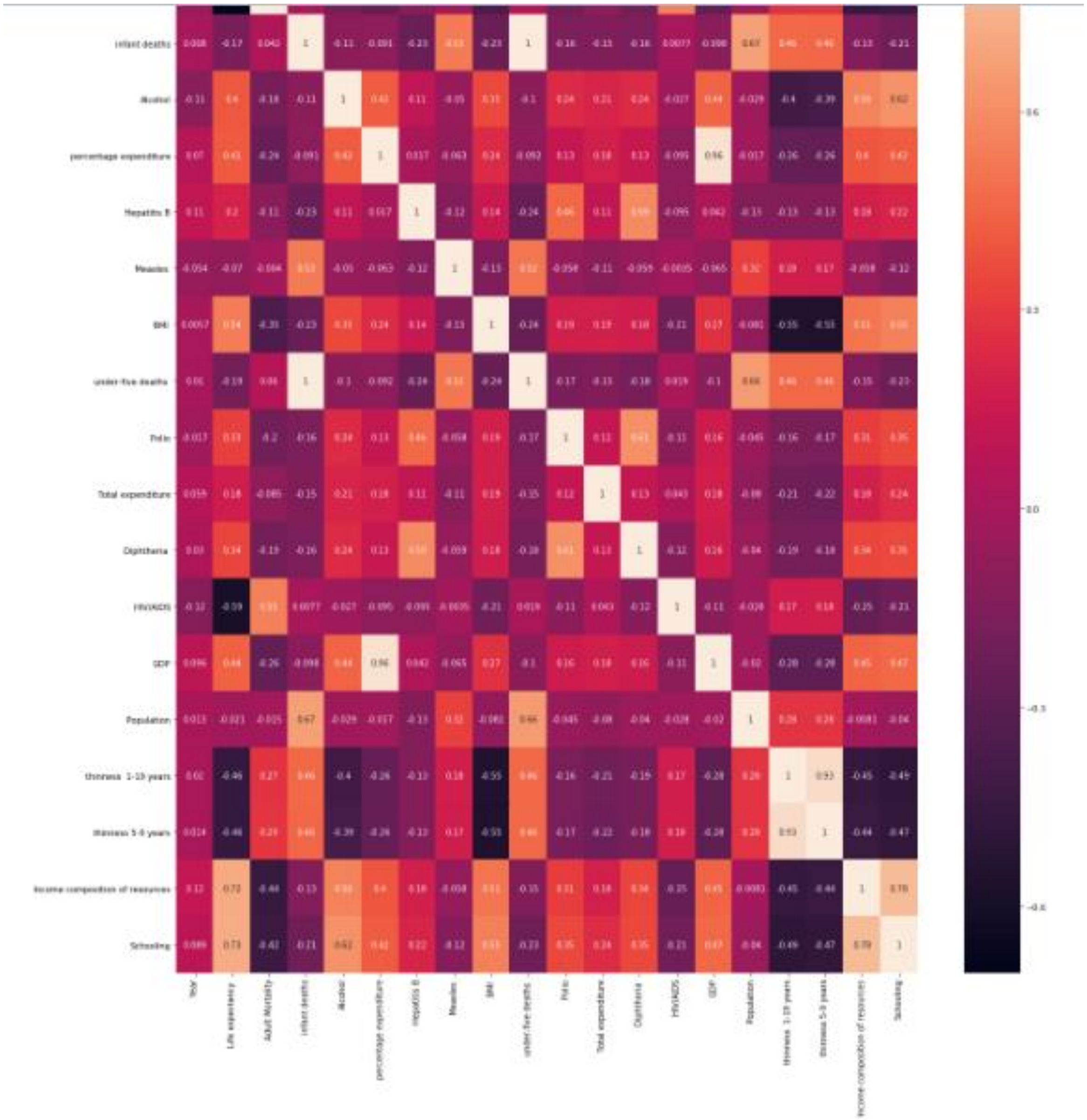
```
sns.distplot(df['Life expectancy '])
```

```
Out[8]: <matplotlib.axes._subplots.AxesSubplot at 0x7f227dfc87b8>
```



```
plt.figure(figsize=(20,25))
```

```
sns.heatmap(df.corr(),annot=True)
```



```
X=df[['Year', 'Adult Mortality', 'infant deaths', 'Alcohol', 'percentage
expenditure', 'Hepatitis B', 'Measles ', ' BMI ', 'under-five deaths ',
'Polio', 'Total expenditure', 'Diphtheria ', ' HIV/AIDS',' thinness 1-19
years', ' thinness 5-9 years', 'Income composition of resources',
'Schooling']]
```

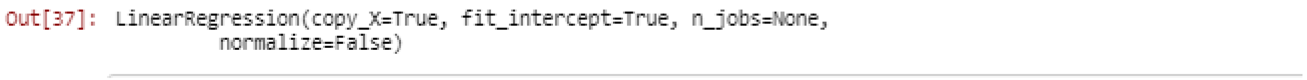
```
y=df['Life expectancy ']
```

#TRAINING AND TESTING DATA

```
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.4)
```

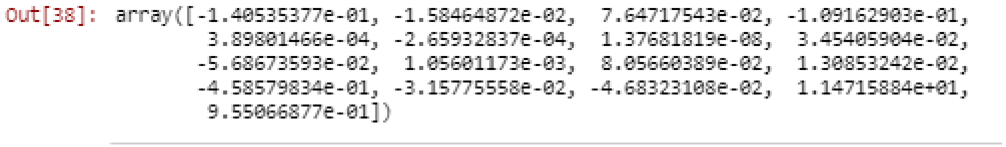
#Linear Regression

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



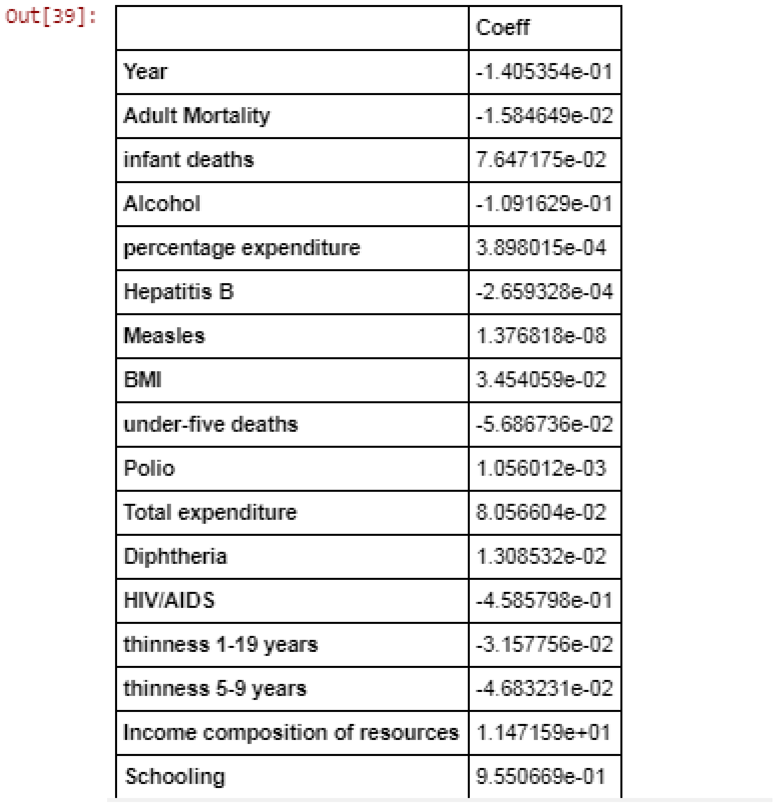
```
Out[37]: LinearRegression(copy_X=True, fit_intercept=True, n_jobs=None,
                  normalize=False)
```

```
linr.coef_
```



```
Out[38]: array([-1.40535377e-01, -1.58464872e-02,  7.64717543e-02, -1.09162903e-01,
                 3.89801466e-04, -2.65932837e-04,  1.37681819e-08,  3.45405904e-02,
                -5.68673593e-02,  1.05601173e-03,  8.05660389e-02,  1.30853242e-02,
                -4.58579834e-01, -3.15775558e-02, -4.68323108e-02,  1.14715884e+01,
                 9.55066877e-01])
```

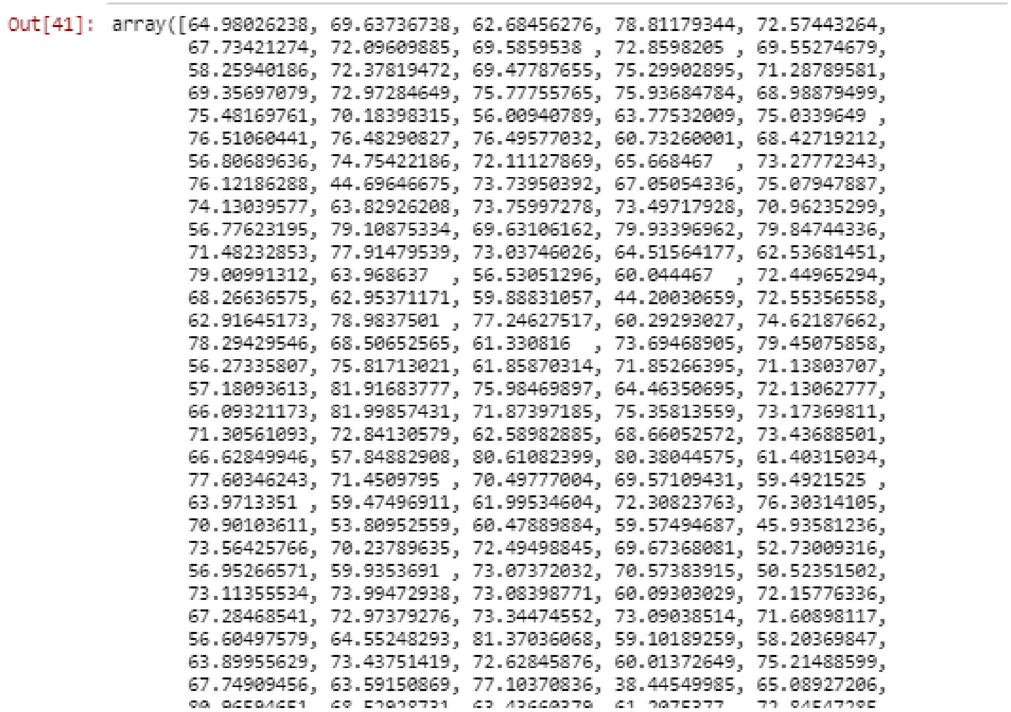
```
pd.DataFrame(linr.coef_,X.columns,columns=['Coeff'])
```



Out[39]:

	Coeff
Year	-1.405354e-01
Adult Mortality	-1.584649e-02
infant deaths	7.647175e-02
Alcohol	-1.091629e-01
percentage expenditure	3.898015e-04
Hepatitis B	-2.659328e-04
Measles	1.376818e-08
BMI	3.454059e-02
under-five deaths	-5.686736e-02
Polio	1.056012e-03
Total expenditure	8.056604e-02
Diphtheria	1.308532e-02
HIV/AIDS	-4.585798e-01
thinness 1-19 years	-3.157756e-02
thinness 5-9 years	-4.683231e-02
Income composition of resources	1.147159e+01
Schooling	9.550669e-01

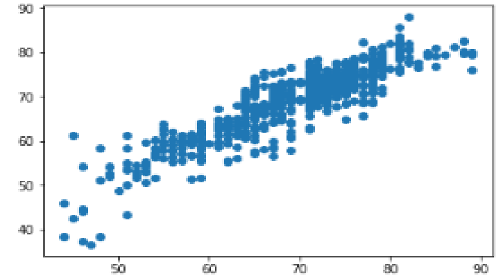
```
predict=linr.predict(X_test)
predict
```



```
Out[41]: array([64.98026238, 69.63736738, 62.68456276, 78.81179344, 72.57443264,
                67.73421274, 72.09609885, 69.5859538 , 72.8598205 , 69.55274679,
                58.25940186, 72.37819472, 69.47787655, 75.29902895, 71.28789581,
                69.35697079, 72.97284649, 75.77755765, 75.93684784, 68.98879499,
                75.48169761, 70.18398315, 56.00940789, 63.77532009, 75.0339649 ,
                76.51060441, 76.48290827, 76.49577032, 60.73260001, 68.42719212,
                56.80689636, 74.75422186, 72.11127869, 65.668467  , 73.27772343,
                76.12186288, 44.69646675, 73.73950392, 67.05054336, 75.07947887,
                74.13039577, 63.82926208, 73.75997278, 73.49717928, 70.96235299,
                56.77623195, 79.10875334, 69.63106162, 79.93396962, 79.84744336,
                71.48232853, 77.91479539, 73.03746026, 64.51564177, 62.53681451,
                79.00991312, 63.968637  , 56.53051296, 60.044467  , 72.44965294,
                68.26636575, 62.95371171, 59.88831057, 44.20030659, 72.55356558,
                62.91645173, 78.9837501 , 77.24627517, 60.29293027, 74.62187662,
                78.29429546, 68.50652565, 61.330816  , 73.69468905, 79.45075858,
                56.27335807, 75.81713021, 61.85870314, 71.85266395, 71.13803707,
                57.18093613, 81.91683777, 75.98469897, 64.46350695, 72.13062777,
                66.09321173, 81.99857431, 71.87397185, 75.35813559, 73.17369811,
                71.30561093, 72.84130579, 62.58982885, 68.66052572, 73.43688501,
                66.62849946, 57.84882908, 80.61082399, 80.38044575, 61.40315034,
                77.60346243, 71.4509795 , 70.49777004, 69.57109431, 59.4921525 ,
                63.9713351 , 59.47496911, 61.99534604, 72.30823763, 76.30314105,
                70.90103611, 53.80952559, 60.47889884, 59.57494687, 45.93581236,
                73.56425766, 70.23789635, 72.49498845, 69.67368081, 52.73009316,
                56.95266571, 59.9353691 , 73.07372032, 70.57383915, 50.52351502,
                73.11355534, 73.99472938, 73.08398771, 60.09303029, 72.15776336,
                67.28468541, 72.97379276, 73.34474552, 73.09038514, 71.60898117,
                56.60497579, 64.55248293, 81.37036068, 59.10189259, 58.20369847,
                63.89955629, 73.43751419, 72.62845876, 60.01372649, 75.21488599,
                67.74909456, 63.59150869, 77.10370836, 38.44549985, 65.08927206,
```

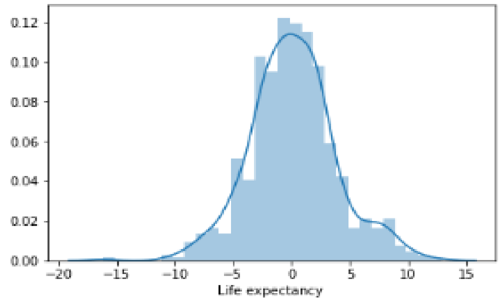
```
plt.scatter(y_test,predict)
```

```
Out[42]: <matplotlib.collections.PathCollection at 0x7f223c308a90>
```



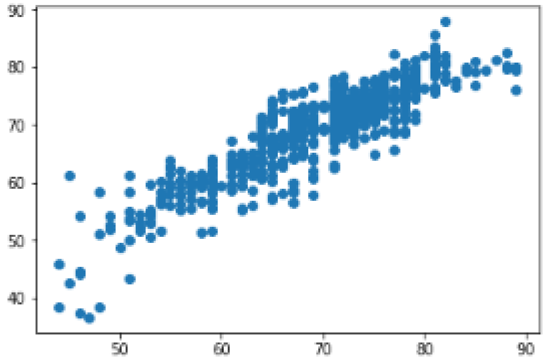
```
sns.distplot(y_test-predict)
```

```
Out[43]: <matplotlib.axes._subplots.AxesSubplot at 0x7f223c258400>
```



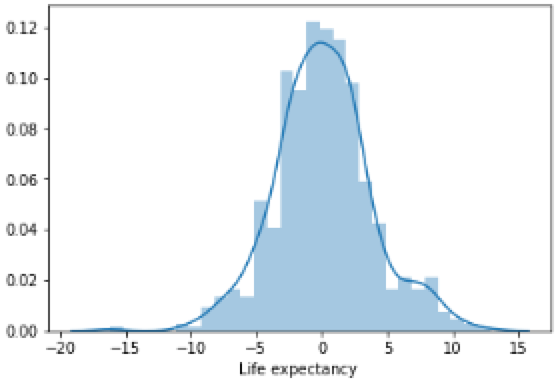
```
plt.scatter(y_test,predict)
```

```
Out[44]: <matplotlib.collections.PathCollection at 0x7f223c1a7b38>
```



```
sns.distplot(y_test-predict)
```

```
Out[45]: <matplotlib.axes._subplots.AxesSubplot at 0x7f223c133dd8>
```



#CREATING ENDPOINT

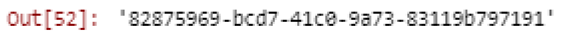
```
!pip install watson-machine-learning-client

wml_credentials={
  "apikey": "K8mOKDeMK8MxeeOLDyGGH97DxpudUTd_AGp60TZFLwwB",
  "iam_apikey_description": "Auto-generated for key 09f44c07-a0e8-45ae-abc4-6a40fb3db63b",
  "iam_apikey_name": "Service credentials-1",
  "iam_role_crn": "crn:v1:bluemix:public:iam::::serviceRole:Writer",
  "iam_serviceid_crn": "crn:v1:bluemix:public:iam-identity::a/b6dfe7663fe3412cb59ed3a1ff7168f0::serviceid:ServiceId-56a0233e-cae6-474f-8e16-cda052fc87ff",
  "instance_id": "d5cbda19-4748-4443-ae76-e89896be3af2",
  "url": "https://eu-gb.ml.cloud.ibm.com"
}
```

```
from watson_machine_learning_client import WatsonMachineLearningAPIClient
client = WatsonMachineLearningAPIClient( wml_credentials )
```

```
model_props= {client.repository.ModelMetaNames.AUTHOR_NAME : "Pradhuman
Gupta",
             client.repository.ModelMetaNames.AUTHOR_EMAIL :
"pradhumangupta099@gmail.com",
             client.repository.ModelMetaNames.NAME : "Life-Expectancy"}
```

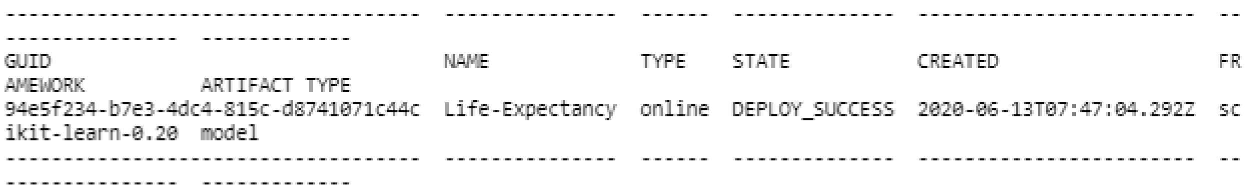
```
model_artifact = client.repository.store_model(lm, meta_props=model_props)
published_model_uid = client.repository.get_model_uid(model_artifact)
```

```
published_model_uid
```



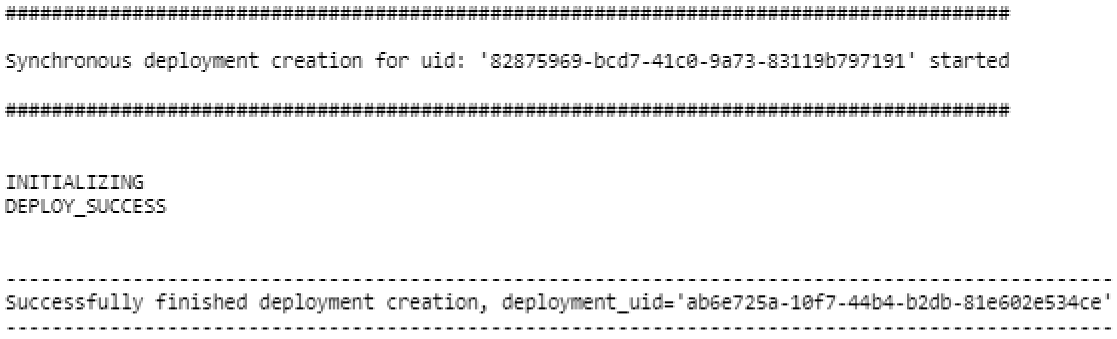
```
Out[52]: '82875969-bcd7-41c0-9a73-83119b797191'
```

```
client.deployments.list()
```



```
------------------------------------  ---------------  ------  --------------  ------------------------  --
---------------  -------------
GUID                                  NAME             TYPE    STATE           CREATED                   FR
AMEWORK          ARTIFACT TYPE
94e5f234-b7e3-4dc4-815c-d8741071c44c  Life-Expectancy  online  DEPLOY_SUCCESS  2020-06-13T07:47:04.292Z  sc
ikit-learn-0.20  model
------------------------------------  ---------------  ------  --------------  ------------------------  --
---------------  -------------
```

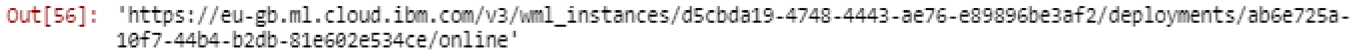
```
deployment= client.deployments.create(published_model_uid, name='Life-
Expectancy')
```



```
#######################################################################################

Synchronous deployment creation for uid: '82875969-bcd7-41c0-9a73-83119b797191' started

#######################################################################################


INITIALIZING
DEPLOY_SUCCESS


------------------------------------------------------------------------------------------------
Successfully finished deployment creation, deployment_uid='ab6e725a-10f7-44b4-b2db-81e602e534ce'
------------------------------------------------------------------------------------------------
```

scoring_endpoints = client.deployments.get_scoring_url(deployment)

scoring_endpoints



```
Out[56]: 'https://eu-gb.ml.cloud.ibm.com/v3/wml_instances/d5cbda19-4748-4443-ae76-e89896be3af2/deployments/ab6e725a-
         10f7-44b4-b2db-81e602e534ce/online'
```