PROJECT REPORT

PREDICTING LIFE EXPECTANCY USING MACHINE LEARNING

Internship under:

TheSMARTBRIDGE

Submitted by: Shah Aakash

EMAIL: aakashatul2000@gmail.com

PROJECT ID: SBS_PRO_215

INTERNSHIP TITLE: Predicting Life Expectancy using Machine Learning – SB45626

Category: Machine Learning

Table of Content

1. INTRODUCTION

1.1. Overview:

Life expectancy is a statistical measure of the average time a human being is expected to live. 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.

We will use dataset which has factors specified and apply different Regression Machine Learning Algorithms to get life expectancy for a particular human being.

Project Requirements:

Cloud for deploying Project
Watson for building AI Project

Functional Requirements:

High Performance
Accurate Result
Efficiency
Platform Independent

Technical Requirements:

Knowledge Python, IBM Cloud, IBM Watson

Software Requirements:

Slack
IBM Cloud
Watson Studio
ZOHO writer

Project Deliverables:

By this project, 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.

Project Team: Aakash Shah
Project Duration: Completion of project done in 30 Days.

1.2. Purpose:

Life expectancy is the most important factor for decision making. Good prognostication for example helps to determine the course of treatment and helps to anticipate the procurement of health care services and facilities, or more broadly: facilitates Advance Care Planning. We should take full advantage of this new era advanced technology to improve the future by predicting it in the present.

2. LITERATURE SURVEY

a. Existing Solution:

We have reviewed existing works and techniques used in the prediction of the human life expectancy, and reached a conclusion that it is feasible for individuals using evolving technologies and devices wearable and mobile health monitoring devices. We have also identified that the factors used for predicting were just personal causes and not related to the surrounding, healthcare facilities, demographic, social, regional and economic factors of the country he resides.

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.

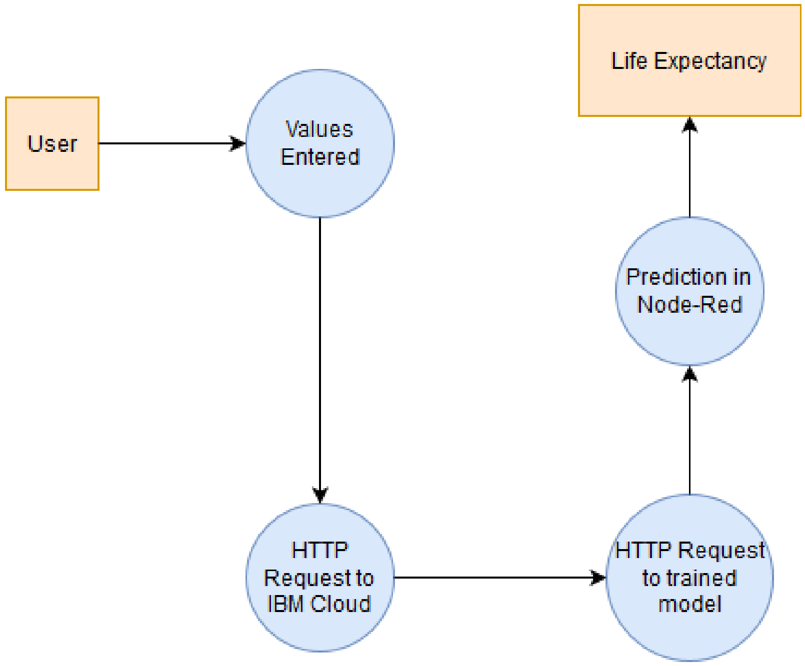
b. Proposed Solution:

To get better insights and predict the life expectancy more accurately, we need to consider some additional features such as country or surrounding dependent features. The previous factors were more human based but it is important to know the economical, regional, social and demographic factors like the GDP, population, education, immunizations, history of illness, health care facility, funds allocated by the government, schemes, medical expenditures like if it is very high then people will shy away to get regular medical checkups, and many more. Due to the large data, we will use IBM Cloud to build our model which will increase the project efficiency.

We will be using the dataset: Life Expectancy by WHO from Kaggle.

3. THEORITICAL ANALYSIS

a. Block/Flow Diagram:



b. Hardware / Software designing:

1. Create necessary IBM Cloud services
2. Create Watson studio project
3. Configure Watson Studio
4. Create IBM Machine Learning instance
5. Create machine learning model in Jupyter notebook
6. Deploy the machine learning model
7. Create flow and configure node
8. Integrate node red with machine learning model
9. Deploy and run Node Red app.

Input is taken from the user using a “Form” element in Node-Red. Then, an HTTP request is made to the IBM cloud that further makes an HTTP request to the deployed model using model’s instance id. After verification of id, the model sends an HTTP response which is finally parsed by the Node-Red application and the result is displayed on the user screen.

4. EXPERIMENTAL INVESTIGATIONS

a. Following factors are taken into account for predicting the life expectancy of a country.

Country : Name of the country

Year : Year (2000-2015)

Status : Developing or Developed

Life expectancy : Life expectancy in age

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

infant deaths: Number of Infant Deaths per 1000 population

Alcohol : Alcohol, recorded per capita (15+) consumption (in liters of pure alcohol)

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

Hepatitis B: Hepatitis B (HepB) immunization coverage among 1-year-olds (%)

Measles : Measles - number of reported cases per 1000 population

BMI : Average Body Mass Index of entire population

under-five deaths: Number of under-five deaths per 1000 population

Polio: Polio (Pol3) immunization coverage among 1-year-olds (%)

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

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

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

GDP : Gross Domestic Product per capita (in USD)

Population : Population of the country

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

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

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

Schooling: Number of years of Schooling (years)

b. Finding the most suitable algorithm:

```
models = OrderedDict([
    ( "Decision Tree Regressor", Pipeline([
                                          ('preprocessor', preprocessor),
                                           ('DTRegressor', DecisionTreeRegressor())])  ),
    ( "Random Forest Regressor", Pipeline([
                                          ('preprocessor', preprocessor),
                                           ('RFRegressor', RandomForestRegressor())])  ),
    ( "Extra Tree Regressor", Pipeline([
                                          ('preprocessor', preprocessor),
                                           ('ETRegressor', ExtraTreesRegressor())])  )
])
```

```
# finding accuracy for above three models.
scores = {}
for (name, model) in models.items():
    model.fit(X_train,Y_train)
    scores[name] =r2_score(model.predict(X_test), Y_test)

scores = OrderedDict(sorted(scores.items()))
scores
```

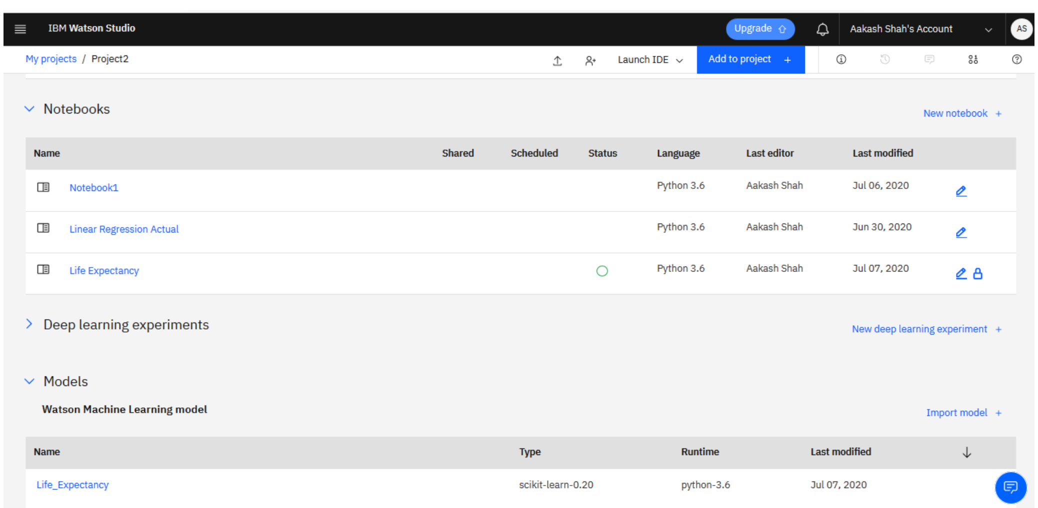
```
/opt/conda/envs/Python36/lib/python3.6/site-packages/sklearn/ensemble/forest.py:246: FutureWarning: The defaul
t value of n_estimators will change from 10 in version 0.20 to 100 in 0.22.
  "10 in version 0.20 to 100 in 0.22.", FutureWarning)
/opt/conda/envs/Python36/lib/python3.6/site-packages/sklearn/ensemble/forest.py:246: FutureWarning: The defaul
t value of n_estimators will change from 10 in version 0.20 to 100 in 0.22.
  "10 in version 0.20 to 100 in 0.22.", FutureWarning)
```

```
OrderedDict([('Decision Tree Regressor', 0.9186643307146558),
             ('Extra Tree Regressor', 0.9680619610322221),
             ('Random Forest Regressor', 0.9593256644793017)])
```

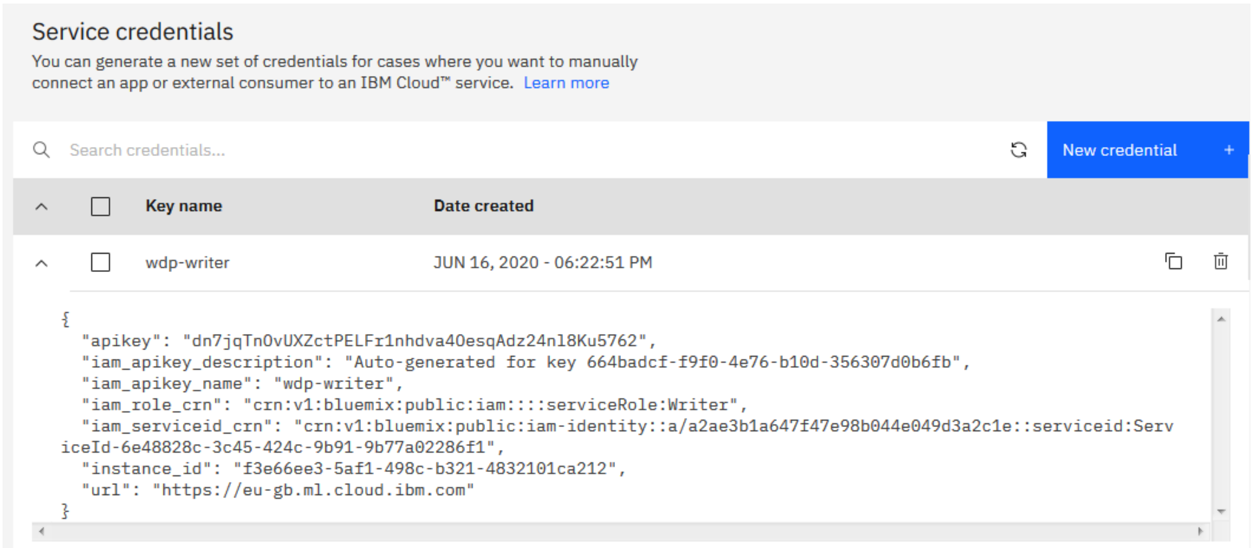
Since Extra Tree Regressor Gives best accuracy amongst all the algorithms, we will use it for this project.

SCREENSHOT

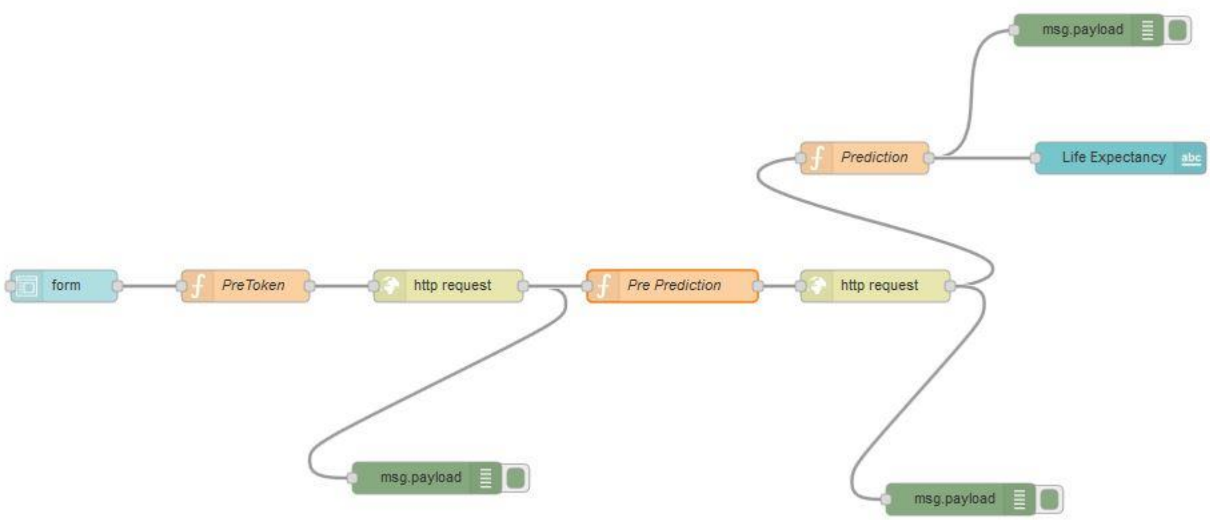
WATSON STUDIO:



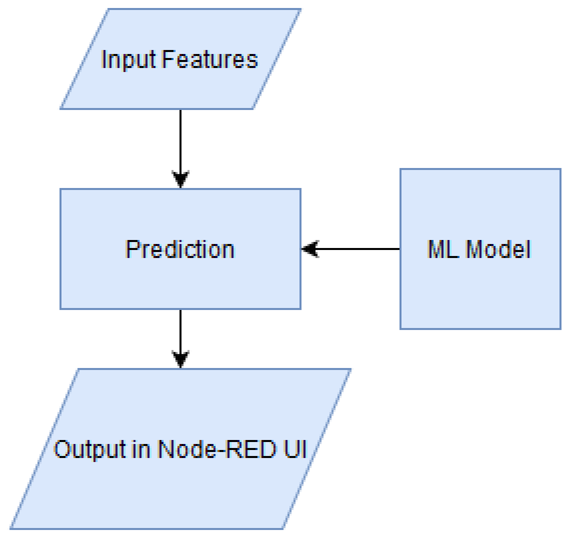
WATSON MACHINE LEARNING SERVICE:



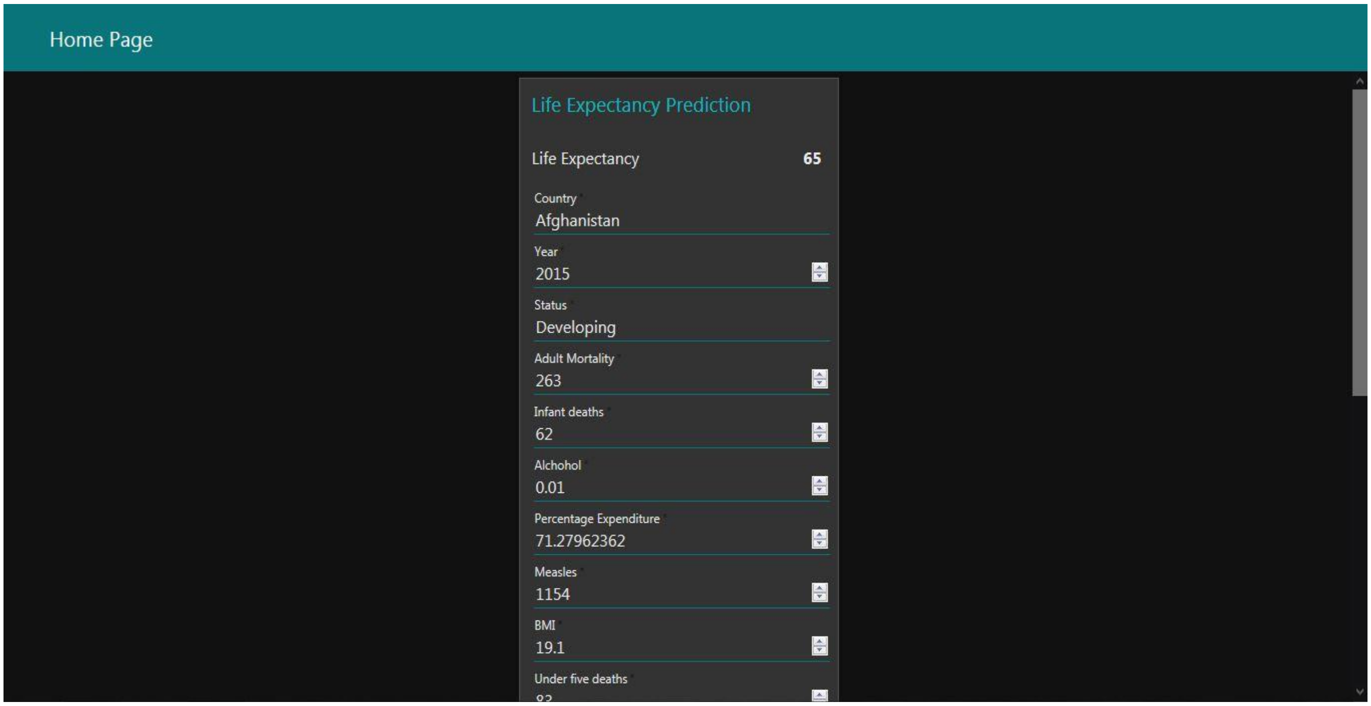
NODE RED FLOW:

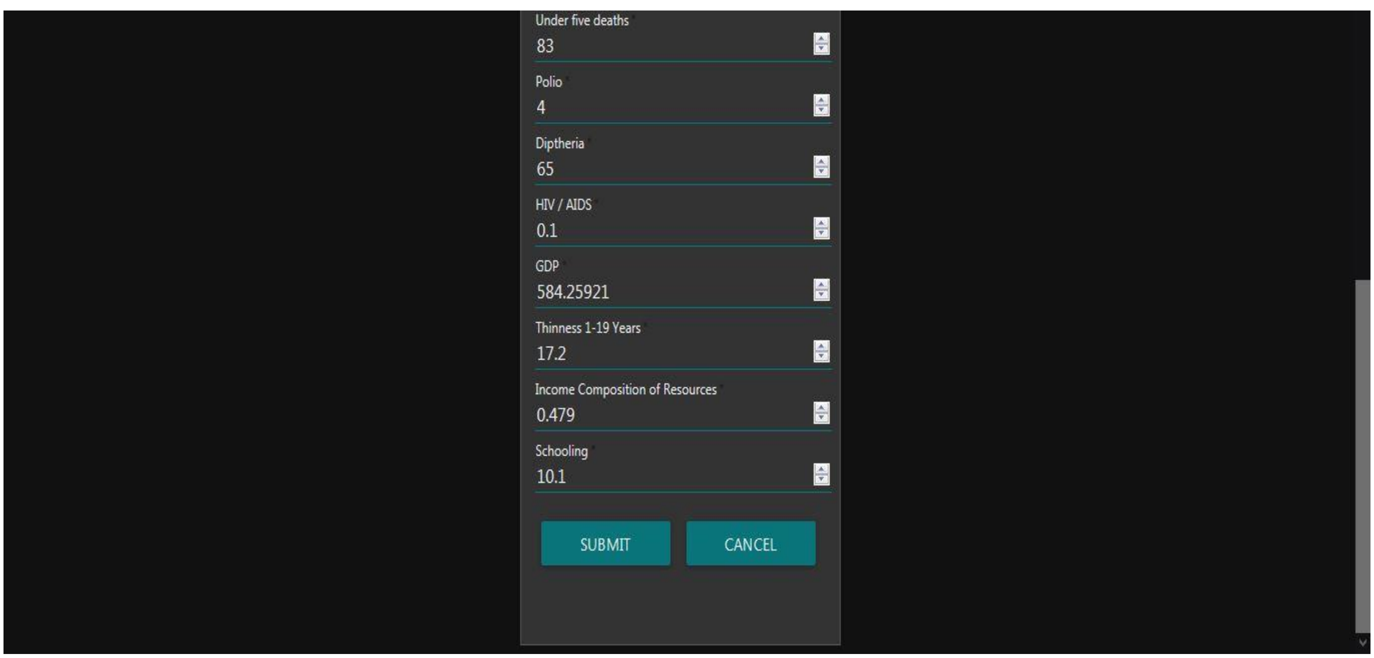


5. FLOWCHART



6. RESULTS


Home Page
Life Expectancy Prediction
Life Expectancy
65
Country
Afghanistan
Year
2015
Status
Developing
Adult Mortality
263
Infant deaths
62
Alchohol
0.01
Percentage Expenditure
71.27962362
Measles
1154
BMI
19.1
Under five deaths


Under five deaths
83
Polio
4
Diptheria
65
HIV / AIDS
0.1
GDP
584.25921
Thinness 1-19 Years
17.2
Income Composition of Resources
0.479
Schooling
10.1
SUBMIT
CANCEL

7. ADVANTAGES & DISADVANTAGES

- **Advantages :**

1. The life expectancy predictor will give important insights and help people achieve good quality of life in future. The country can plan and improve various healthcare facilities.
2. Advantages of using IBM Cloud: Easy to use and deploy, easy to connect with UI, takes care of large storage space.
3. The application is easy and simple to use.
4. Extra Tree Regressor require much more time to train as compared to decision trees as it generates a lot of trees (instead of one tree in case of decision tree) and makes decision on the majority of votes.

- **Disadvantages :**

1. Can be only used by the people having the knowledge of data analysis.
2. As the model is deployed on cloud, so one requires good internet connection to use the application.
3. The model creates a lot of trees (unlike only one tree in case of decision tree) and combines their outputs. By default, it creates 100 trees in Python sklearn library. To do so, this algorithm requires much more computational power and resources.
4. The Node-Red application needs to make HTTP request to IBM cloud and then another HTTP request to the model before providing the prediction. That makes the application a bit slow.

8. APPLICATIONS

1. 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. It will be easier for a country to determine the predicting factor which is contributing to lower value of life expectancy and can be used in various organizations to improve the quality of service.
3. The governments can plan and develop their health infrastructures by keeping the most correlated factors in mind.
4. The project can help governments to keep track of their country's health status so they can plan for the future accordingly.

9. CONCLUSION

By doing the above procedure and all we successfully created Life expectancy prediction system using IBM Watson studio, Watson machine learning and Node-RED service. The potential use of project is not limited to health care in practice, but could also be useful in other clinical applications such as clinical trials. The project makes a good use of machine learning in predicting life expectancy of a country that can help respective government in making policies that will serve for the benefit of the nation and entire humankind.

10. FUTURE SCOPE

The accuracy of the model can be increased. This can be done by training more data. Also, the website can be added with many more features to improve the user experience. The user input can be connected to the database for future purposes.

11. BIBILOGRAPHY

• https://cloud.ibm.com/

• IBM Developer, "IBM Watson Studio: Create a project", 2019. [Online]. Available: https://www.youtube.com/watch?v=-CUi8GezG1I&list=PLzpeuWUENMK2PYtasCaKK4bZjaYzhW23L&index=2

•IBM Cloud setup [Online]. Available: https://www.ibm.com/cloud/get-started .

•IBM Developer, "Node-RED starter tutorial" [Online]. Available: https://developer.ibm.com/tutorials/how-to-create-a-node-red-starter-application/ .

•"Infuse AI into your applications with Watson AI to make more accurate predictions". [Online]. Available: https://www.ibm.com/watson/products-services

•IBM Developer, "IBM Watson Machine Learning: Get Started in IBM Cloud", 2020 [Online]. Available: https://www.youtube.com/watch?v=NmdjtezQMSM .

•Kumar Rajarshi, "Life Expectancy (WHO) Statistical Analysis on factors influencing Life Expectancy", 2018. [Online]. Available: https://www.kaggle.com/kumarajarshi/life-expectancy-who

•IBM Developer, "IBM Watson: Sign up for Watson Studio and Watson Knowledge Catalog", 2019. [Online]. Available: https://www.youtube.com/watch?v=DBRGlAHdj48&list=PLzpeuWUENMK2PYtas CaKK4bZjaYzhW23L .

APPENDIX

A. Source code

Installing and importing all the libraries:

```
pip install pandas-profiling[notabook,html]
import pandas as pd
import numpy as np
import types
import pandas as pd
from botocore.client import Config
import ibm_boto3
from pandas_profiling import ProfileReport
import seaborn as sns
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split
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.tree import DecisionTreeRegressor
from sklearn.ensemble import RandomForestRegressor
from sklearn.ensemble import ExtraTreesRegressor
from collections import OrderedDict
from sklearn.metrics import accuracy_score
from sklearn.metrics import r2_score,mean_squared_error
from watson_machine_learning_client import WatsonMachineLearningAPIClient
```

Loading the file :

```
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_77be1ba6853e4234a807562522c0f9fe=ibm_boto3.client(service_name='s3',
ibm_api_key_id='RhfUtb9kpFy4x4nD9GSzRDe2nZNtKasRmIHGivr9TUUO',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_77be1ba6853e4234a807562522c0f9fe.get_object(Bucket='project2-donotdelete-pr-5jade4kd7cqsin',Key='datasets_12603_17232_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 )
dataset1 = pd.read_csv(body)
dataset1.head()
```

Exploratory data analysis:

```
profile = ProfileReport(data, title='Pandas Profiling Report', explorative=True)
profile.to_notebook_iframe()
profile.to_file("Data_Analysis.html")
```

Feature Selection:

```
#get correlations of each features in dataset
corrmat = dataset1.corr()
top_corr_features = corrmat.index
plt.figure(figsize=(20,20))
#plot heat map
g=sns.heatmap(dataset1[top_corr_features].corr(),annot=True,cmap="RdYlGn")
```

Dropping Unnecessary features:

```
dataset1.drop(['Hepatitis B','Total expenditure',' thinness 5-9 years','Population'],axis=1,inplace=True)
```

Handling missing values:

BMI:-https://ourworldindata.org/obesity
HDI and Schooling:-http://hdr.undp.org/en/countries
GDP:-https://www.worldometers.info/gdp/gdp-per-capita/
Thinness:https://apps.who.int/gho/data/view.main.NCDBMIMINUS205-19Cv
Alcohols:-https://ourworldindata.org/alcohol-consumption

OR

```
dataset1['column_name'].fillna(dataset1.groupby(['Country'])['Life Expectancy'].transform('median'))
dataset1.to_csv('Life_Expectancy.csv',index=False)
```

Loading New Dataset:

```
body=client_77be1ba6853e4234a807562522c0f9fe.get_object(Bucket='project2-donotdelete-pr-5jade4kd7cqsin',Key='Life_Expectancy.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 )
dataset = pd.read_csv(body)
dataset.head()
print(dataset.columns)
dataset.rename(columns = {'Life expectancy ' : 'Life Expectancy'}, inplace = True)
X = dataset.drop(['Life Expectancy'],axis=1)
Y = dataset['Life Expectancy']
```

Splitting Dataset for Training and Testing:

```
X_train, X_test, Y_train, Y_test = train_test_split(X, Y, test_size = 0.2, random_state = 0,shuffle=True)
```

Separating Categorical and Numerical Features:

```
numeric_features = ['Year', 'Adult Mortality',
       'infant deaths', 'Alcohol', 'percentage expenditure', 'Measles',
       ' BMI ', 'under-five deaths ', 'Polio', 'Diphtheria', ' HIV/AIDS',
       'GDP', ' thinness  1-19 years', 'Income composition of resources',
       'Schooling']
categorical_features = ['Country', 'Status']
```

Creating Pipeline:

```
categorical_transformer = Pipeline(steps=[('onehot', OneHotEncoder(handle_unknown='ignore'))])
numeric_transformer=Pipeline(steps=[('imputer',SimpleImputer(strategy='median'))])
preprocessor = ColumnTransformer(transformers=[ ('cat', categorical_transformer, categorical_features),('num', numeric_transformer, numeric_features)])
```

Finding Best Algorithm:

```
models = OrderedDict([
    ( "Decision Tree Regressor", Pipeline([('preprocessor', preprocessor),
                                   ('DTRegressor', DecisionTreeRegressor())]) ),
    ( "Random Forest Regressor", Pipeline([('preprocessor', preprocessor),
                                   ('RFRegressor', RandomForestRegressor())])  ),
    ( "Extra Tree Regressor", Pipeline([ ('preprocessor', preprocessor),
```

```
                                          ('ETRegressor', ExtraTreesRegressor())]))
])
scores = {}
for (name, model) in models.items():
    model.fit(X_train,Y_train)
    scores[name] =r2_score(model.predict(X_test), Y_test)
scores = OrderedDict(sorted(scores.items()))
scores
```

#Output:-

```
OrderedDict([('Decision Tree Regressor',
0.9186643307146558),
             ('Extra Tree Regressor',
0.9680619610322221),
             ('Random Forest Regressor',
0.9593256644793017)])
```

Since Extra Tree Regressor Gives best accuracy amongst all the algorithms, we will use it for this project.

```
et_model = Pipeline([ ('preprocessor', preprocessor),('ETRegressor',
ExtraTreesRegressor())])
```

Train our model:

```
et_model.fit(X_train,Y_train)
X_test.shape
```

Making prediction:

```
predict = et_model.predict(X_test)
print(Y_train)

predict
error = abs(predict - Y_test)
print('Average absolute error:', round(np.mean(error), 2), 'degrees.')
```

#Output:-

```
Average absolute error: 0.95 degrees.
```

Finding Accuracy:

```
mape = 100 * (error / Y_test)
```

```
accuracy = 100 - np.mean(mape)
print('Accuracy:', round(accuracy, 2), '%.')
```

#Output:

```
Accuracy: 98.58 %
```

Creating Watson Machine Learning Client:

```
wml_credentials = {
  "apikey": "dn7jqTnOvUXZctPELFr1nhdva4OesqAdz24nl8Ku5762",
  "iam_apikey_description": "Auto-generated for key 664badcf-f9f0-4e76-b10d-
    356307d0b6fb",
  "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/a2ae3b1a647f47e98b044e049d3a2c1e::serviceid:ServiceId-
    6e48828c-3c45-424c-9b91-9b77a02286f1",
  "instance_id": "f3e66ee3-5af1-498c-b321-4832101ca212",
  "url": "https://eu-gb.ml.cloud.ibm.com"
}
client = WatsonMachineLearningAPIClient(wml_credentials)
print(client.version)
```

#Output:

```
1.0.378
```

 # Set meta-data to our model:

```
model_props = {client.repository.ModelMetaNames.AUTHOR_NAME:
"Aakash",  client.repository.ModelMetaNames.AUTHOR_EMAIL:
"aakashatul2000@gmail.com",client.repository.ModelMetaNames.NAME:
"Life_Expectancy"}
```

 # Storing our model to the repository:

```
model_artifact = client.repository.store_model(et_model,
meta_props=model_props)
```

 # After storing the model,we will get model id which will be used for deploying the model

```
published_model_uid =
client.repository.get_model_uid(model_artifact)published_model_uid
```

#Output:

```
'7a780358-3ba5-4654-a26a-9db9e86c4384'
```

```
# Creating deployment of our model
# After successfully deployment of our model, we will get scoring endpoint, to
which our Node-RED app send the data.
deployment = client.deployments.create(published_model_uid,
name="Life_Expectancy")
scoring_endpoint = client.deployments.get_scoring_url(deployment)
scoring_endpoint
```

#Output:

```
'https://eu-gb.ml.cloud.ibm.com/v3/wml_instances/f3e66ee3-5af1-498c-b321-4832101ca212/deployments/113c0c75-061c-41ee-9140-4136d373a45c/online'
```

Node RED : Flows.json


```
[
    {
        "id": "8ec212aa.ebcd38",
        "type": "tab",
        "label": "Flow 1",
        "disabled": false,
        "info": ""
    },
    {
        "id": "f26ffe1a.b269f8",
        "type": "function",
        "z": "8ec212aa.ebcd38",
        "name": "PreToken",
        "func":
"global.set(\"ctry\",msg.payload.ctry)\nglobal.set(\"yr\",msg.payload.yr)\nglobal.se
t(\"sts\",msg.payload.sts)\nglobal.set(\"adult_mor\",msg.payload.adult_mor)\nglob
al.set(\"inf_deaths\",msg.payload.inf_deaths)\nglobal.set(\"alcohol\",msg.payload.a
lcohol)\nglobal.set(\"per_expend\",msg.payload.per_expend)\nglobal.set(\"measles
\",msg.payload.measles)\nglobal.set(\"bmi\",msg.payload.bmi)\nglobal.set(\"under
```

_five_deaths\",msg.payload.under_five_deaths)\nglobal.set(\"polio\",msg.payload.polio)\nglobal.set(\"diptheria\",msg.payload.diptheria)\nglobal.set(\"hiv_aids\",msg.payload.hiv_aids)\nglobal.set(\"gdp\",msg.payload.gdp)\nglobal.set(\"thinness\",msg.payload.thinness)\nglobal.set(\"hdi\",msg.payload.hdi)\nglobal.set(\"edu_index\",msg.payload.edu_index)\nvar apikey=\"dn7jqTnOvUXZctPELFr1nhdva4OesqAdz24nl8Ku5762\";\nmsg.headers={\"content-type\":\"application/x-www-form-urlencoded\"}\nmsg.payload={\"grant_type\":\"urn:ibm:params:oauth:grant-type:apikey\",\"apikey\":apikey}\nreturn msg;",
        "outputs": 1,
        "noerr": 0,
        "x": 260,
        "y": 300,
        "wires": [
            [
                "327b778f.d09f18"
            ]
        ]
    },
    {
        "id": "327b778f.d09f18",
        "type": "http request",
        "z": "8ec212aa.ebcd38",
        "name": "",
        "method": "POST",
        "ret": "obj",
        "paytoqs": false,
        "url": "https://iam.cloud.ibm.com/identity/token",
        "tls": "",
        "persist": false,
        "proxy": "",
        "authType": "",
        "x": 450,
        "y": 300,
        "wires": [
            [
                "2f3caa1.6e4e0d6",
                "f1d24c55.3f4f6"
            ]
        ]

},
{
"id": "5d9748f3.7b2d68",
"type": "debug",
"z": "8ec212aa.ebcd38",
"name": "",
"active": true,
"tosidebar": true,
"console": false,
"tostatus": false,
"complete": "payload",
"targetType": "msg",
"x": 1050,
"y": 60,
"wires": []
},
{
"id": "f1d24c55.3f4f6",
"type": "function",
"z": "8ec212aa.ebcd38",
"name": "Pre Prediction",
"func": "var ctry = global.get('ctry')\nvar yr = global.get('yr')\nvar sts = global.get('sts')\nvar adult_mor = global.get('adult_mor')\nvar inf_deaths = global.get('inf_deaths')\nvar alcohol = global.get('alcohol')\nvar per_expend = global.get('per_expend')\nvar measles = global.get('measles')\nvar bmi = global.get('bmi')\nvar under_five_deaths = global.get('under_five_deaths')\nvar polio = global.get('polio')\nvar diptheria = global.get('diptheria')\nvar hiv_aids = global.get('hiv_aids')\nvar gdp = global.get('gdp')\nvar thinness = global.get('thinness')\nvar hdi = global.get('hdi')\nvar edu_index = global.get('edu_index')\nvar token=msg.payload.access_token\nvar instance_id=\"f3e66ee3-5af1-498c-b321-4832101ca212\"\nmsg.headers={'Content-Type': 'application/json',\"Authorization\":\"Bearer \"+token,\"ML-Instance-ID\":instance_id}\nmsg.payload={\"fields\":[\"Country\", \"Year\", \"Status\", \"Adult Mortality\",\"infant deaths\", \"Alcohol\", \"percentage expenditure\", \"Measles\",\" BMI \", \"under-five deaths \", \"Polio\", \"Diphtheria\", \" HIV/AIDS\",\"GDP\", \" thinness 1-19 years\", \"Income composition of resources\",\"Schooling\"],\"values\":[[ctry,yr,sts,adult_mor,inf_deaths,alcohol,per_expend,measles,bmi,under_five_deaths,polio,diptheria,hiv_aids,gdp,thinness,hdi,edu_index]]}\nreturn msg;",

"outputs": 1,
"noerr": 0,
"x": 660,
"y": 300,
"wires": [
[
"175819f6.2a7976"
]
]
},
{
"id": "175819f6.2a7976",
"type": "http request",
"z": "8ec212aa.ebcd38",
"name": "",
"method": "POST",
"ret": "obj",
"paytoqs": false,
"url": "https://eu-gb.ml.cloud.ibm.com/v3/wml_instances/f3e66ee3-5af1-498c-b321-4832101ca212/deployments/113c0c75-061c-41ee-9140-4136d373a45c/online",
"tls": "",
"persist": false,
"proxy": "",
"authType": "",
"x": 850,
"y": 300,
"wires": [
[
"a69d19a3.0e7a28",
"8a7b84c0.1e7ed"
]
]
},
{
"id": "f9402e74.d34668",
"type": "ui_form",
"z": "8ec212aa.ebcd38",
"name": "",
"label": "",

"group": "b0859f33.0bc788",
"order": 0,
"width": 0,
"height": 0,
"options": [
    {
        "label": "Country",
        "value": "ctry",
        "type": "text",
        "required": true,
        "rows": null
    },
    {
        "label": "Year",
        "value": "yr",
        "type": "number",
        "required": true,
        "rows": null
    },
    {
        "label": "Status",
        "value": "sts",
        "type": "text",
        "required": true,
        "rows": null
    },
    {
        "label": "Adult Mortality",
        "value": "adult_mor",
        "type": "number",
        "required": true,
        "rows": null
    },
    {
        "label": "Infant deaths",
        "value": "inf_deaths",
        "type": "number",
        "required": true,
        "rows": null
    },

{
  "label": "Alchohol",
  "value": "alcohol",
  "type": "number",
  "required": true,
  "rows": null
},
{
  "label": "Percentage Expenditure",
  "value": "per_expend",
  "type": "number",
  "required": true,
  "rows": null
},
{
  "label": "Measles",
  "value": "measles",
  "type": "number",
  "required": true,
  "rows": null
},
{
  "label": "BMI",
  "value": "bmi",
  "type": "number",
  "required": true,
  "rows": null
},
{
  "label": "Under five deaths",
  "value": "under_five_deaths",
  "type": "number",
  "required": true,
  "rows": null
},
{
  "label": "Polio",
  "value": "polio",
  "type": "number",
  "required": true,

```
    "rows": null
},
{
    "label": "Diptheria",
    "value": "diptheria",
    "type": "number",
    "required": true,
    "rows": null
},
{
    "label": "HIV / AIDS",
    "value": "hiv_aids",
    "type": "number",
    "required": true,
    "rows": null
},
{
    "label": "GDP",
    "value": "gdp",
    "type": "number",
    "required": true,
    "rows": null
},
{
    "label": "Thinness 1-19 Years",
    "value": "thinness",
    "type": "number",
    "required": true,
    "rows": null
},
{
    "label": "Income Composition of Resources",
    "value": "hdi",
    "type": "number",
    "required": true,
    "rows": null
},
{
    "label": "Schooling",
    "value": "edu_index",
```

"type": "number",
"required": true,
"rows": null
}
],
"formValue": {
"ctry": "",
"yr": "",
"sts": "",
"adult_mor": "",
"inf_deaths": "",
"alcohol": "",
"per_expend": "",
"measles": "",
"bmi": "",
"under_five_deaths": "",
"polio": "",
"diptheria": "",
"hiv_aids": "",
"gdp": "",
"thinness": "",
"hdi": "",
"edu_index": ""
},
"payload": "",
"submit": "submit",
"cancel": "cancel",
"topic": "",
"x": 90,
"y": 300,
"wires": [
[
"f26ffe1a.b269f8"
]]},{
"id": "dc897067.e585a",
"type": "ui_text",
"z": "8ec212aa.ebcd38",
"group": "b0859f33.0bc788",
"order": 1,
"width": 0,

```
    "height": 0,
    "name": "",
    "label": "Life Expectancy",
    "format": "{{msg.payload}}",
    "layout": "row-spread",
    "x": 1080,
    "y": 180,
    "wires": []
},{
    "id": "a69d19a3.0e7a28",
    "type": "function",
    "z": "8ec212aa.ebcd38",
    "name": "Prediction",
    "func": "msg.payload=msg.payload.values[0][0]\nreturn msg;",
    "outputs": 1,
    "noerr": 0,
    "x": 840,
    "y": 180,
    "wires": [
        [
            "5d9748f3.7b2d68",
            "dc897067.e585a"
        ]]},
{
    "id": "2f3caa1.6e4e0d6",
    "type": "debug",
    "z": "8ec212aa.ebcd38",
    "name": "",
    "active": true,
    "tosidebar": true,
    "console": false,
    "tostatus": false,
    "complete": "payload",
    "targetType": "msg",
    "x": 430,
    "y": 480,
    "wires": []
},
{
    "id": "8a7b84c0.1e7ed",
```

"type": "debug",
"z": "8ec212aa.ebcd38",
"name": "",
"active": true,
"tosidebar": true,
"console": false,
"tostatus": false,
"complete": "payload",
"targetType": "msg",
"x": 930,
"y": 500,
"wires": []
},
{
"id": "b0859f33.0bc788",
"type": "ui_group",
"z": "",
"name": "Life Expectancy Prediction",
"tab": "4c0410f9.39d26",
"order": 1,
"disp": true,
"width": "6",
"collapse": false
},
{
"id": "4c0410f9.39d26",
"type": "ui_tab",
"z": "",
"name": "Home Page",
"icon": "dashboard",
"disabled": false,
"hidden": false
}
]

Github Link: https://github.com/SmartPracticeschool/llSPS-INT-2916-Predicting-Life-Expectancy-using-Machine-Learning

Demonstration of project Link: https://youtu.be/0gkRJplAPb8
Feedback Link: https://youtu.be/Mt8_BMM_DSY

THANK YOU