PROJECT REPORT

**Name:** Mayank Mohan  
**Email:** mayankarya396@gmail.com  
**Title:** **Predicting Life Expectancy Using Machine Learning**  
**Category:** **Machine Learning**  
**Video Link**:

**YouTube** : <https://youtu.be/wWGHxaUMzJk>

**Google drive:** <https://photos.app.goo.gl/Zb2Jm1F7m8p9VZA66>  
**WebPage Link:** <https://node-red-ygfgx.eu-gb.mybluemix.net/ui/#!/0?socketid=iFp_5clVb6ApvU9KAAAS>

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. 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 Services, Python, Node-Red **Software Requirements**: Watson Studio, Node-Red  
**Project Deliverables**: Smartinternz Internship  
**Project Team**: INDIVIDUAL  
**Project Duration**: 30 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.

2. LITERATURE SURVEY

2.1. Existing Problem

Predicting a human’s life expectancy has been a long-term question to humankind. Many calculations and research have been done to create an equation despite it being impractical to simplify these variables into one equation. Currently there are various smart devices and applications such as smartphone apps and wearable devices that provide wellness and fitness tracking. Some apps provide health related data such as sleep monitoring, heart rate measuring, and calorie expenditure collected and processed by the devices and servers in the cloud. However no existing works provide the Personalised Life expectancy.

2.2. Proposed Solution

Steps:

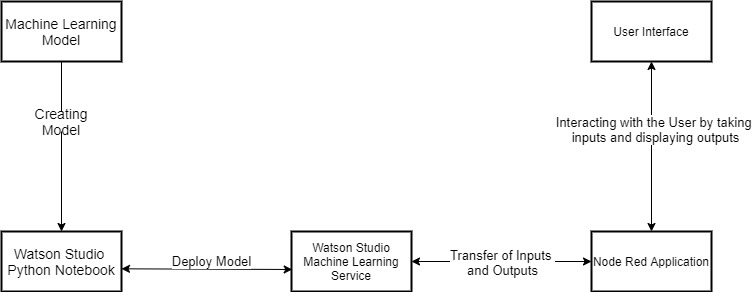
* Create IBM cloud services
* Configure Watson Studio
* Create Node-Red Flow to connect all services together
* Deploy and run Node-Red app

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

We train our regression models we will need to first split up our data into an A array that contains the features to train on, and a B array with the target variable, here it is "Life Expectancy column". We split the data into a training set and a testing set. We will train out model on the training set and then use the test set to evaluate the model and the best model is chosen to evaluate the predictions.

3. THEORETICAL ANALYSIS

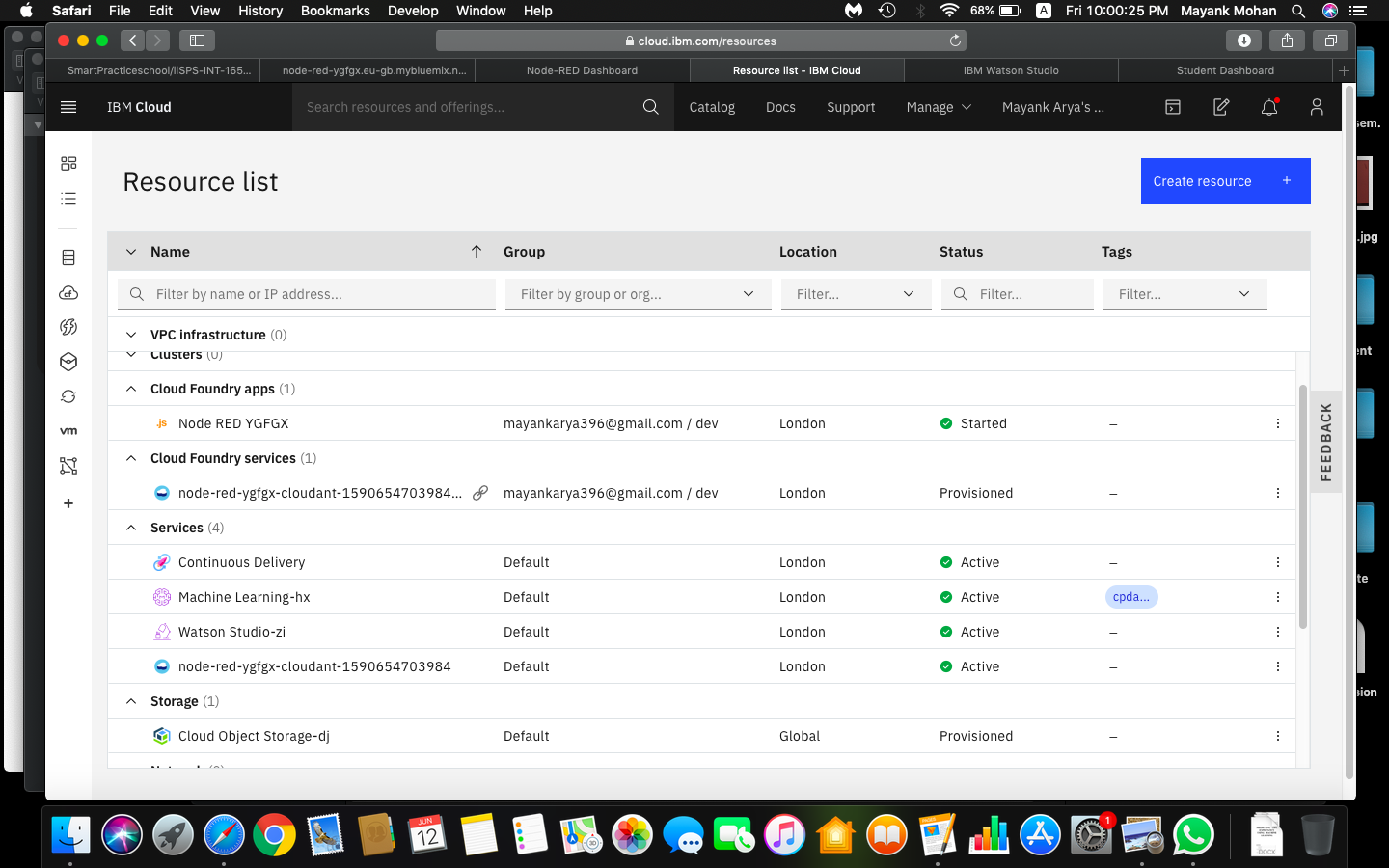
**3.1 Block Diagram:**

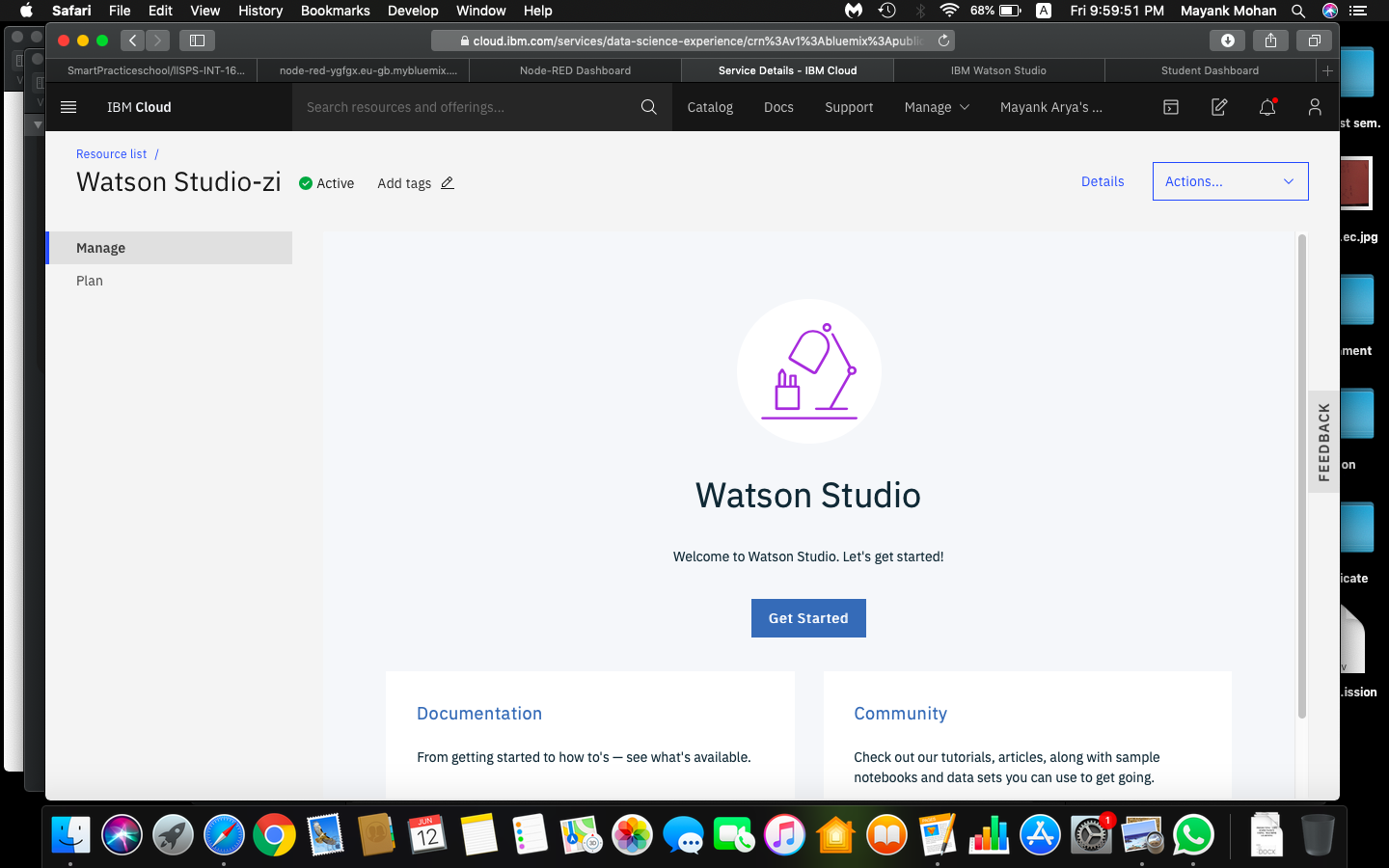


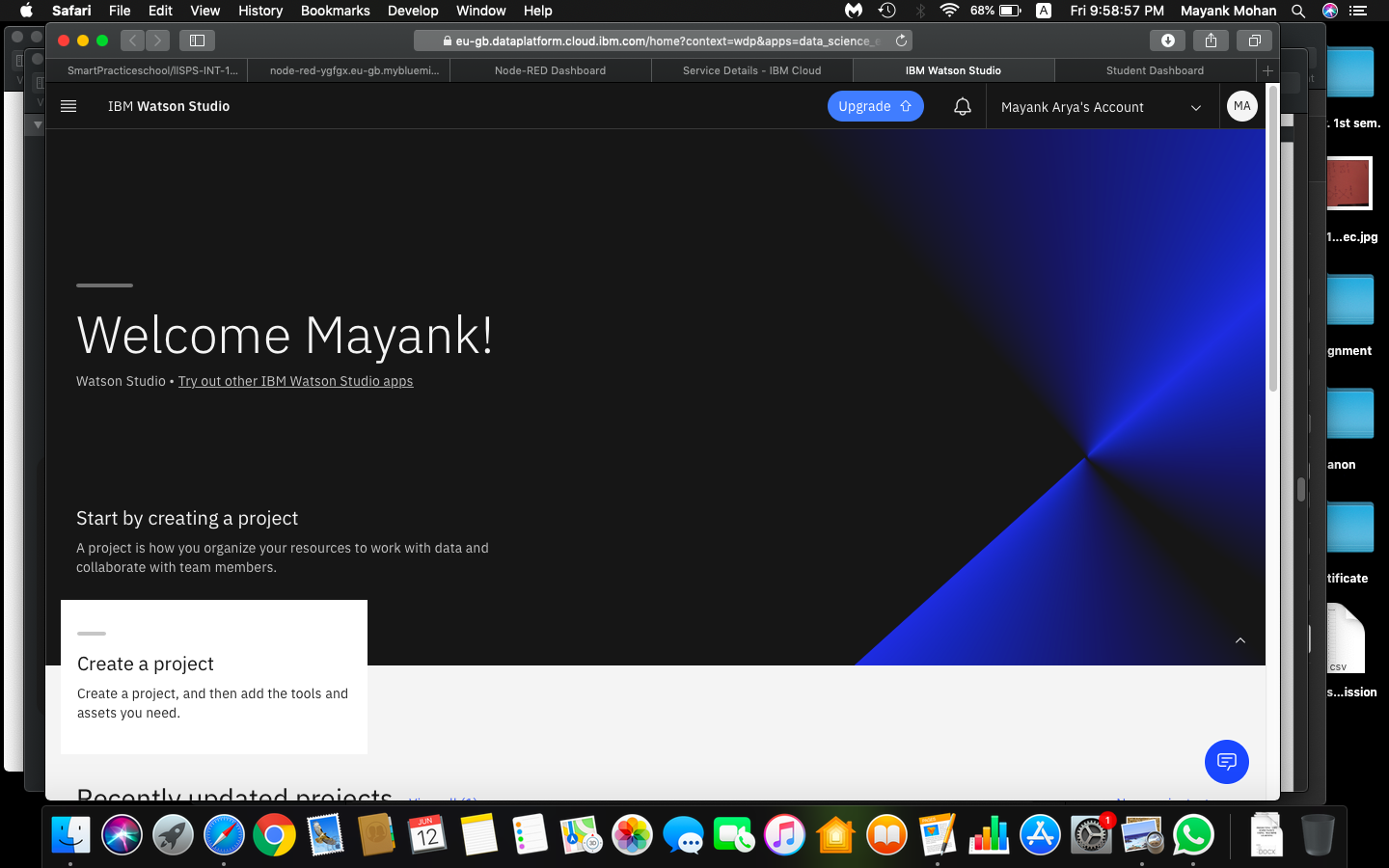
**3.2 Hardware/Software Designing:**

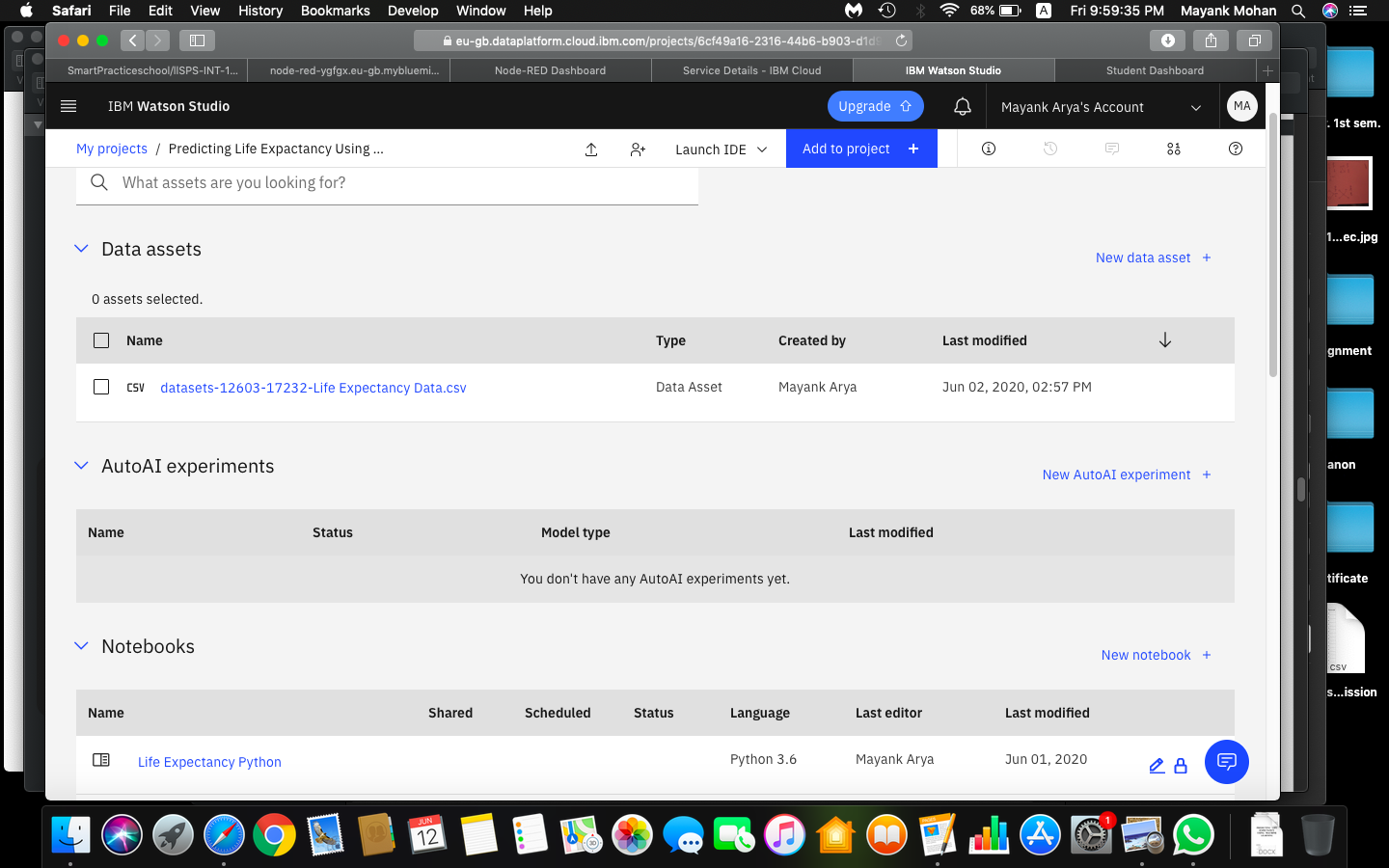
* **Project Requirements**: Python, IBM Cloud, IBM Watson
* **Functional Requirements**: IBM cloud
* **Technical Requirements**: Machine Learning, WATSON Services, Python, Node-Red
* **Software Requirements**: Python,Watson Studio, Node-Red

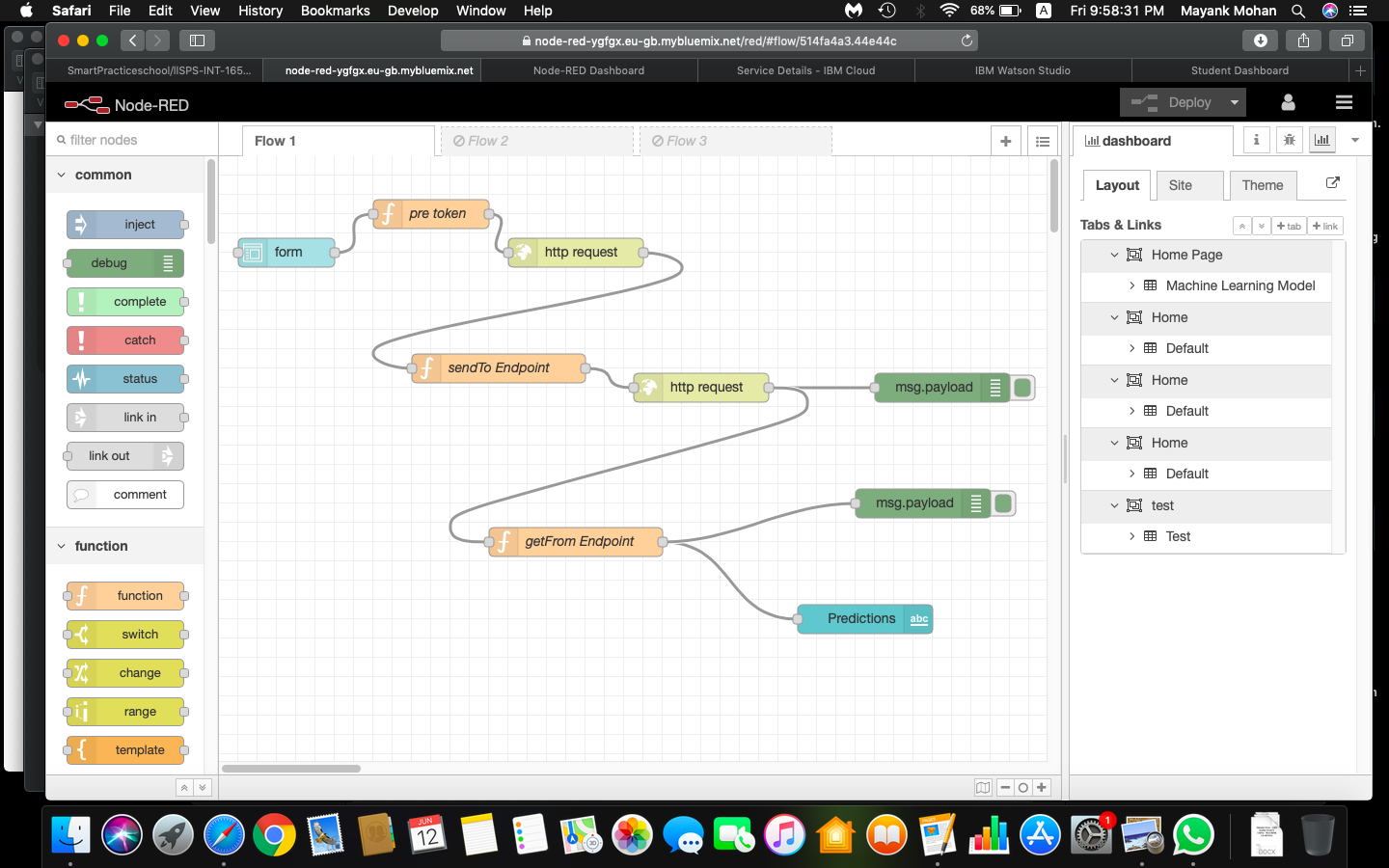
4. EXPERIMENTAL INVESTIGATIONS

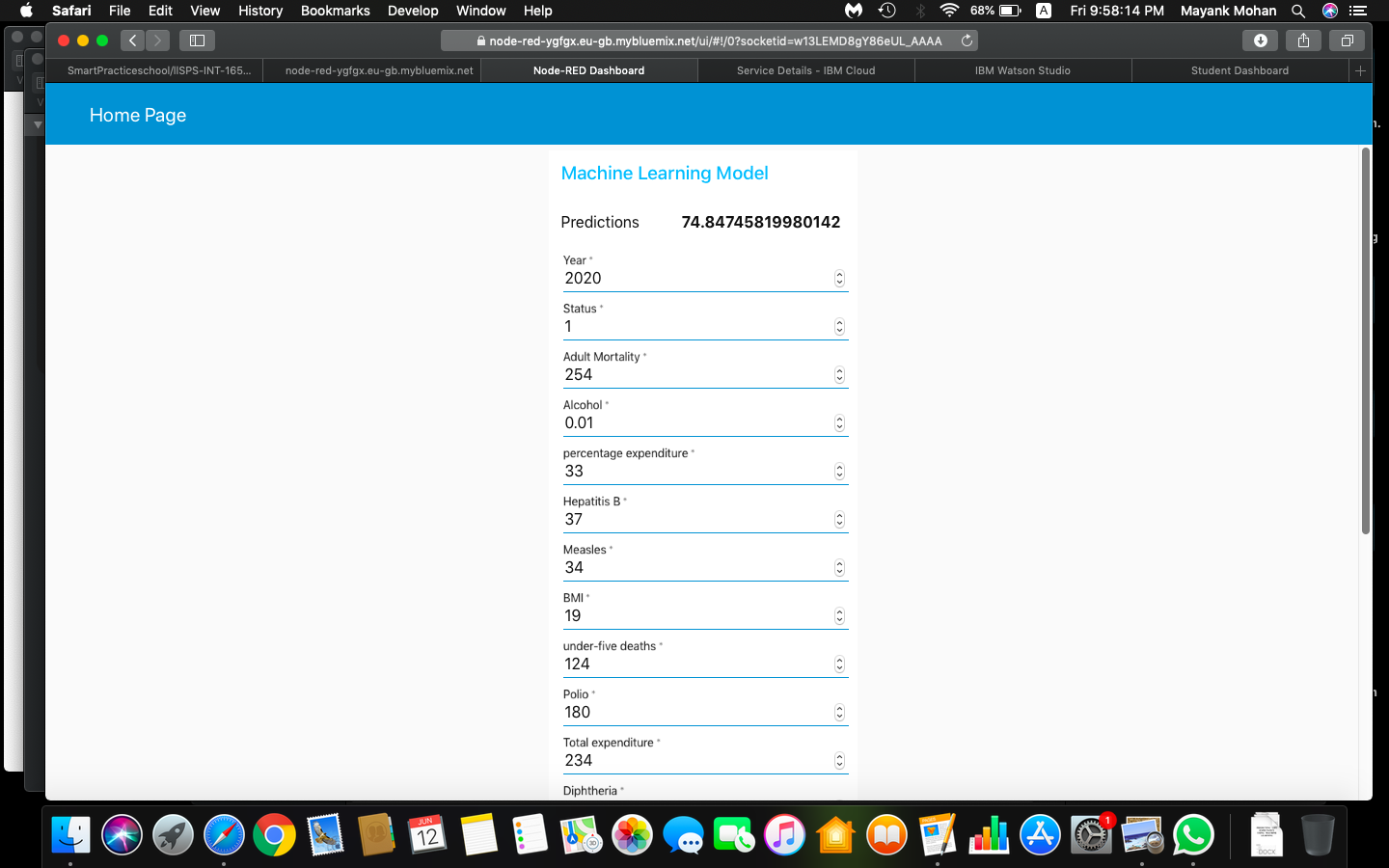
**A) IBM Cloud Resource List**

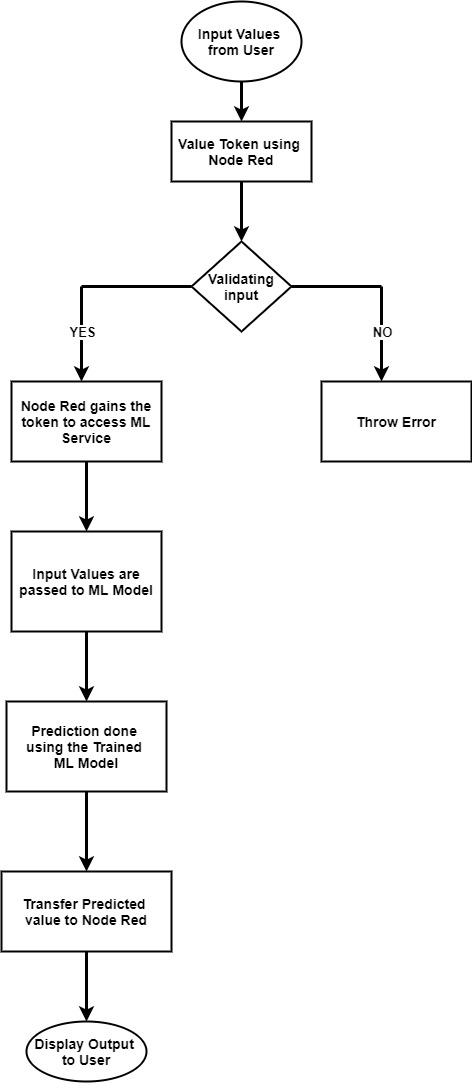
**B) IBM Watson Studio**



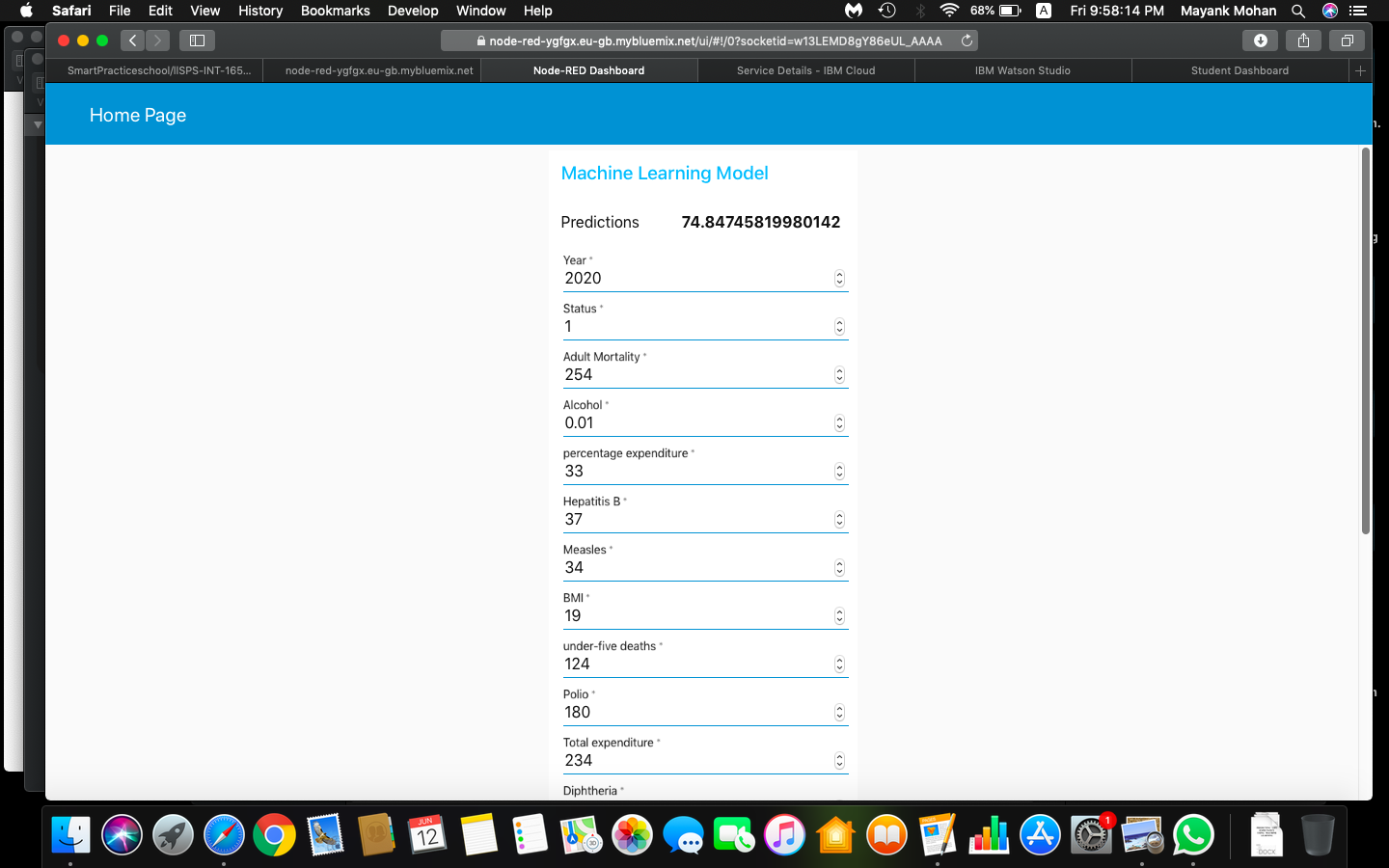
**C) IBM Cloud Project Details** 

**D)Node-Red Flow**

**E) Life Expectancy Prediction UI**

**5.** **Flow Chart**

6. RESULT

This is the Life Expectancy UI.

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 95% 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

One can plan to explore methods for gaining more insight in the nature of the patterns that are detected by neural networks, as well as making the determinants of a certain prediction transparent. For future use, one can integrate the life expectancy prediction with providing suggestions and medications to the individual using the application. This will help predict as well as increase the individual’s life expectancy.

The scalability and flexibility of the application can also be improved with advancement in technology and availability of new and improved resources. Also, with the growth in Artificial Neural networks and Deep learning, one can integrate that with our existing application. With the help of Convolutional Neural networks and Computer vision, we can also try to take into account the physical health and appearance of a person. Mental health can also be taken into account while predicting life expectancy with the help of sentiment analysis systems as well.

11. BIBLIOGRAPHY

· <https://developer.ibm.com/tutorials/how-to-create-a-node-red-starter-application/>

· [https://bookdown.org/caoying4work/watsonstudio-workshop/jn.html#deploy-model-as- web-service](https://bookdown.org/caoying4work/watsonstudio-workshop/jn.html)

· <https://www.ibm.com/watson/products-services>

**APPENDIX:**

**Source Code :**

**import** pandas **as** pd

**import** numpy **as** np

**import** matplotlib.pyplot **as** plt

**import** seaborn **as** sns

**import** types

**import** pandas **as** pd

**from** botocore.client **import** Config

**import** ibm\_boto3

**from** statsmodels.api **import** OLS

**from** pandas.plotting **import** scatter\_matrix

**from** sklearn.model\_selection **import** train\_test\_split

**from** sklearn.preprocessing **import** StandardScaler

**from** sklearn.linear\_model **import** LinearRegression

**from** sklearn.metrics **import** mean\_squared\_error, mean\_absolute\_error, r2\_score

**from** sklearn.ensemble **import** RandomForestRegressor

**from** watson\_machine\_learning\_client **import** WatsonMachineLearningAPIClient

plt.style.use('fivethirtyeight')

sns.set\_style(style='dark')

pd.options.display.max\_rows = **None**

pd.options.display.max\_columns = **None**

get\_ipython().run\_line\_magic('matplotlib', 'inline')

**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\_ebbb4a33aa644527a29c121fe811d861 = ibm\_boto3.client(service\_name='s3',

ibm\_api\_key\_id='Spw7WE7hvbAzqH7VVQzP371Dmff4KxXCXmP79ot4k55X',

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\_ebbb4a33aa644527a29c121fe811d861.get\_object(Bucket='predictinglifeexpactancyusingpyth-donotdelete-pr-q2i0haf7t7oea6',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 )

df = pd.read\_csv(body)

df.head()

df.describe()

df.columns = list(map(str.strip, df.columns.tolist()))

df.isnull().sum()

df.dropna(axis=0, inplace=**True**)

df.shape

country = df.Country

df.drop(['Country'], axis=1, inplace=**True**)

df.columns.tolist()

df.Status.value\_counts()

val = {'Developing':0, 'Developed':1}

df.Status = df.Status.map(val)

df.dtypes

y\_train = pd.DataFrame(df['Life expectancy'])

X\_train = df.drop(['Life expectancy'], axis=1)

fig, axes = plt.subplots(4,5, figsize=(19,15))

**for** index, column **in** enumerate(X\_train.columns.tolist()):

sns.distplot(a = X\_train[column],

color= '#009900' ,

ax = axes[index%4][index%5])

cor = df.corr()

cor['Life expectancy'].sort\_values(ascending=**False**)

sns.jointplot(x='Schooling', y='Life expectancy',

data=df, alpha=0.2,

color= '#ff4d4d', size=7)

sns.jointplot(x='Adult Mortality', y='Life expectancy',

data=df, alpha=0.2,

color= '#ff4d4d', size=7)

fig, axes = plt.subplots(2,2, figsize=(17,15))

size\_set = ((-100, 1300), (-200, 22000000))

**for** index, column **in** enumerate(['Measles', 'Population']):

sns.boxplot(y=column, data = X\_train,

color = '#e60073', ax = axes[index][0])

sns.boxplot(y=column, data=X\_train,

color = '#e60073', ax= axes[index][1])

axes[index][1].set\_ylim(size\_set[index])

axes[0][0].set\_title('Box plot for Measles')

axes[0][1].set\_title('Zooming Box plot for Measles to see the quartiles and IQR clearly')

axes[1][0].set\_title('Box plot for Population')

axes[1][1].set\_title('Zooming Box plot for Population to see the quartiles and IQR clearly')

plt.show()

X\_train.describe()[['Measles','Population']]

scatter\_matrix(X\_train[['Schooling','Income composition of resources','BMI','thinness 1-19 years','HIV/AIDS','Adult Mortality']],

figsize=(20,20))

plt.plot()

corr = X\_train.corr()

fig, ax = plt.subplots(figsize=(14, 12))

sns.heatmap(corr, linewidth=0.01, linecolor='black', annot=**True**, ax=ax)

**def** get\_corr(data, threshold):

cor\_set = set()

**for** i **in** range(data.shape[0]):

**for** j **in** range(i):

**if** abs(data.iloc[i, j]) >= threshold:

cor\_set.add(data.columns[j])

**return** cor\_set

corr\_group = get\_corr(corr, 0.95)

corr\_group

X\_train.shape

X\_train.dtypes

ols = OLS(endog=y\_train, exog=X\_train).fit()

ols.summary()

X\_train.drop(['thinness 5-9 years','Population','Total expenditure','Measles','Hepatitis B','Polio'],

axis=1,

inplace=**True**)

pd.options.display.max\_rows = **None**

pd.options.display.max\_columns = **None**

ols = OLS(endog=y\_train, exog=X\_train).fit()

ols.summary()

X\_train.columns

std = StandardScaler()

X\_train = std.fit\_transform(X\_train)

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X\_train, y\_train, test\_size=0.15, random\_state=4)

X\_train.shape, X\_test.shape

# ## Linear Regression

linear = LinearRegression()

linear.fit(X\_train, y\_train)

y\_pred = linear.predict(X\_test)

#r2\_score

score = r2\_score(y\_test, y\_pred)

print(f'r2 score: {score}')

#mse

error = mean\_squared\_error(y\_test, y\_pred)

print("mean squared error: {}".format(error))

#mae

error = mean\_absolute\_error(y\_test, y\_pred)

print(f'mean absolute error: {error}')

# # Random forest regressor

rf\_regressor = RandomForestRegressor(n\_estimators = 300, random\_state = 0)

rf\_regressor.fit(X\_train,y\_train)

print(rf\_regressor.score(X\_test, y\_test))

y\_pred = rf\_regressor.predict(X\_test)

#r2\_score

score = r2\_score(y\_test, y\_pred)

print(f'r2 score: {score}')

#mse

error = mean\_squared\_error(y\_test, y\_pred)

print("mean squared error: {}".format(error))

#mae

error = mean\_absolute\_error(y\_test, y\_pred)

print(f'mean absolute error: {error}')

get\_ipython().system('pip install watson-machine-learning-client')

wml\_credentials={

"apikey": "\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*",

"instance\_id": "\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*",

"url": "https://eu-gb.ml.cloud.ibm.com"

}

client = WatsonMachineLearningAPIClient( wml\_credentials )

model\_props = {client.repository.ModelMetaNames.AUTHOR\_NAME: "Mayank Mohan",

client.repository.ModelMetaNames.AUTHOR\_EMAIL: "mayankarya396@gmail.com",

client.repository.ModelMetaNames.NAME: "Life Expectancy Prediction"}

model\_artifact =client.repository.store\_model(linear, meta\_props=model\_props)

published\_model\_uid = client.repository.get\_model\_uid(model\_artifact)

published\_model\_uid

deployment = client.deployments.create(published\_model\_uid, name="LifeExpectancyPrediction")

scoring\_endpoint = client.deployments.get\_scoring\_url(deployment)

scoring\_endpoint