**Diabetics Prediction System based on Life Style**

**Prepared by**

# Kakkireni Naga Pavani Assistant Professor

**Department of Electronics and Communication Engineering. Gokaraju Rangaraju Institute of Engineering & Technology.**

**Hyderabad.**



**Index**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **1** | |  | **INTRODUCTION** | |
|  | |  | 1.1 Overview | |
|  | |  | 1.2 Purpose | |
| **2** | |  | **LITERATURE SURVEY** | |
|  | |  | 2.1 Existing problem | | |
|  | |  | 2.2 Proposed solution | | |
| **3** | |  | **THEORITICAL ANALYSIS** | |
|  | |  | 3.1 Block diagram | |
|  | |  | 3.2 Hardware / Software designing | | |
| **4** | |  | **EXPERIMENTAL INVESTIGATIONS** | | |
| **5** | |  | **FLOWCHART** | |
| **6** | |  | **RESULT** |  |
| **7** | |  | **ADVANTAGES & DISADVANTAGES** | | |
| **8** | |  | **CONCLUSION** | |
| **9** | |  | **FUTURE SCOPE** | |
| **10** | |  | **BIBILOGRAPHY** | |
|  | |  | **APPENDIX** | |
|  | |  | A. Source code | |
|  |  | |  | |

1. INTRODUCTION
   1. Overview

Diabetes is one of the terrible diseases in the world which is not only a disease but a root cause for varieties of diseases like heart strokes, blindness, kidney related diseases, etc. The normal procedure is regularly visiting hospitals or labs for testing whether it is normal or not and eagerly waiting for the reports with tension because the report brings lot of changes in our life style .This process is not only mental stress but also waste of money for paying for diagnostic centers. The solution is modern approach which uses auto AI and machine learning models for prediction. The bigger advantage is early prediction saves the risc from being critical. The main aim of the project is to develop an auto AI IBM model for prediction of diabetes Mellitus accurately.

Artificial Intelligence is a wide domain which contains wide spectrum of methods for modelling variety of solutions to problems. It has applications in healthcare domain, very effectively used in prediction of diseases. Various life style and other parameters have an imp progression of this disease. These parameters and the outcome for number of patients already available in the domain can be used to m from and predict with good amount of accuracy when new/unseen data points are posed to it. This would open new avenues for Mach of healthcare that would assist the healthcare stakeholders to make informed decisions by studying large number of patient data. This them to strategize customized treatment methods to better meet the varying disease symptoms in case of chronic diseases like diabetic.

Diabetes is one of the most common chronic diseases. The number of cases of diabetes in the world is likely to increase m the next 30 years; from 115 million in 2000 to 284 million in 2030. In type I diabetes, the disease is caused by the failure of the pancr sufficient amount of insulin which leads to an uncontrolled increase in blood glucose unless the patient administers insulin, typically b injection. This work is concerned with managing diabetic patients by trying to predict their glucose levels in the near future (30 minut current levels. The goal of this paper is to determine the future blood glucose value of a diabetic patient using an artificial neural network approaches, which involve questioning the patients, feature extraction procedures were implemented (diabetic dynamic model) on dia time series, in order to extract a knowledge (how patient blood glucose level will change a according to the external food intake and a activities). An artificial neural network was then trained using these features in order to predict the future value of blood glucose level accuracy.

* 1. Purpose

The purpose of building this application is to predict whether or not a patient acquires diabetes based on some diagnostic features. Though this application would not substitute a doctor or professional medical advice, it serves as a preliminary investigator the symptoms a person has w.r.t. diabetes.

1. ITERATURE SURVEY
   1. Existing problem

Diabetes disease prediction is one of the progressive research area in the healthcare fields.Although many data mining techniques have been employed to assess the main causes of diabetes, but only few sets of clinical risk factors are considered. Due to this, some important factors like pre-diabetes health conditions are not considered in their analysis. So the results produced by such techniques may not represent appropriate diabetes pattern and risk factors appropriately.

* 1. Proposed solution

In this, we need to diagnostically predict whether or not a patient has diabetes, based on certain diagnostic measurements included in the dataset. Several constraints were placed on the selection of these instances from a larger database. In particular, all patients here are females at least 21 years old of Pima Indian heritage.

The datasets consist of several medical predictor variables and one target variable, Diabetes. Predictor variables include the number of pregnancies the patient has had, their BMI, insulin level, age, and so on.

**Skills Required:**

● Flask Integration

● IBM Nodered

● IBM Watson Studio

● IBM Machine Learning

● IBM Cloud Object Storage

**Software Requirements:**

1. Operating system: Windows 10 9/8/19

2. Coding Language: HTML, PYTHON

3. Services used:

○ IBM Watson Studio-ky

○ Machine Learning-te

○ node-red-hzblo-2020--cloudant

○ Continuous Delivery

4. Cloud Foundry apps: Node RED HZBLO

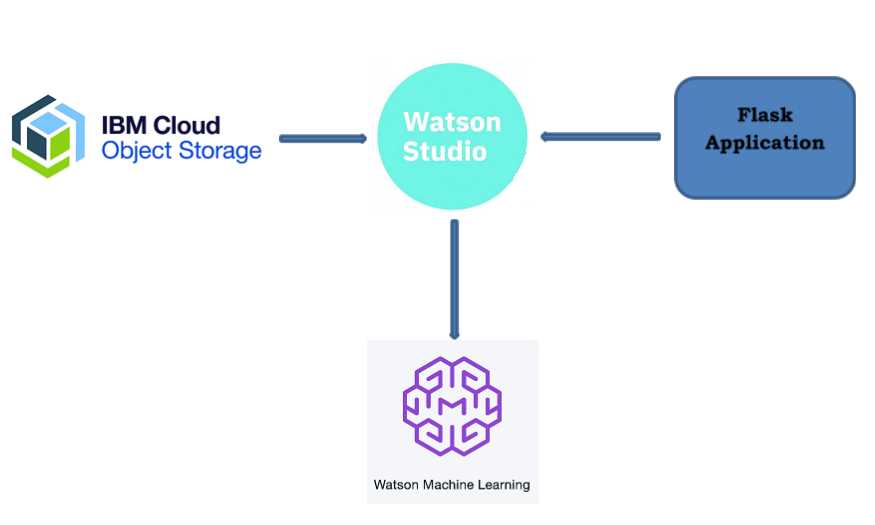
5. Storage: Cloud Object Storage-ol

**Outcome:**

Develop an end-to-end web application that predicts the probability of females having diabetes. The application has to be built with Python-Flask or Django framework with the machine learning model trained & deployed on IBM Watson Studio.

1. THEORITICAL ANALYSIS
   1. Block diagram

**Proposed Technical Architecture:**



**Fig. 3.1 Proposed Technical Architecture**

1. EXPERIMENTAL INVESTIGATIONS

The tools in Machine Learning and Watson Studio available in IBM services catalog were explored to create the project in addition to Node-RED to create the UI.

4.1 Tasks Completed:

* Collecting Dataset
* Create IBM Academic Initiative Account
* Login to IBM Cloud
* Create Cloud Object Storage Service
* Download Watson Studio Desktop
* Create Watson Studio Platform
* Create Machine Learning Service
* Create A Project in Watson Studio
* Upload the Dataset
* Train a model in Watson Studio
* Deploying the Model
* Create UI using Node Red service
* Build Flask Application

* 1. **Steps to be followed to build the project:**
* Create a project in IBM Watson Studio – Predict Heart Failure.
* Add Auto AI experiment.
* Create a Machine Learning instance.
* Associate ML instance to the project.
* Load the dataset to cloud object storage.
* Select the target variable (HEARTFAILURE parameter) in the dataset
* Train the model.
* Deploy.
* Build web application using Node-Red.

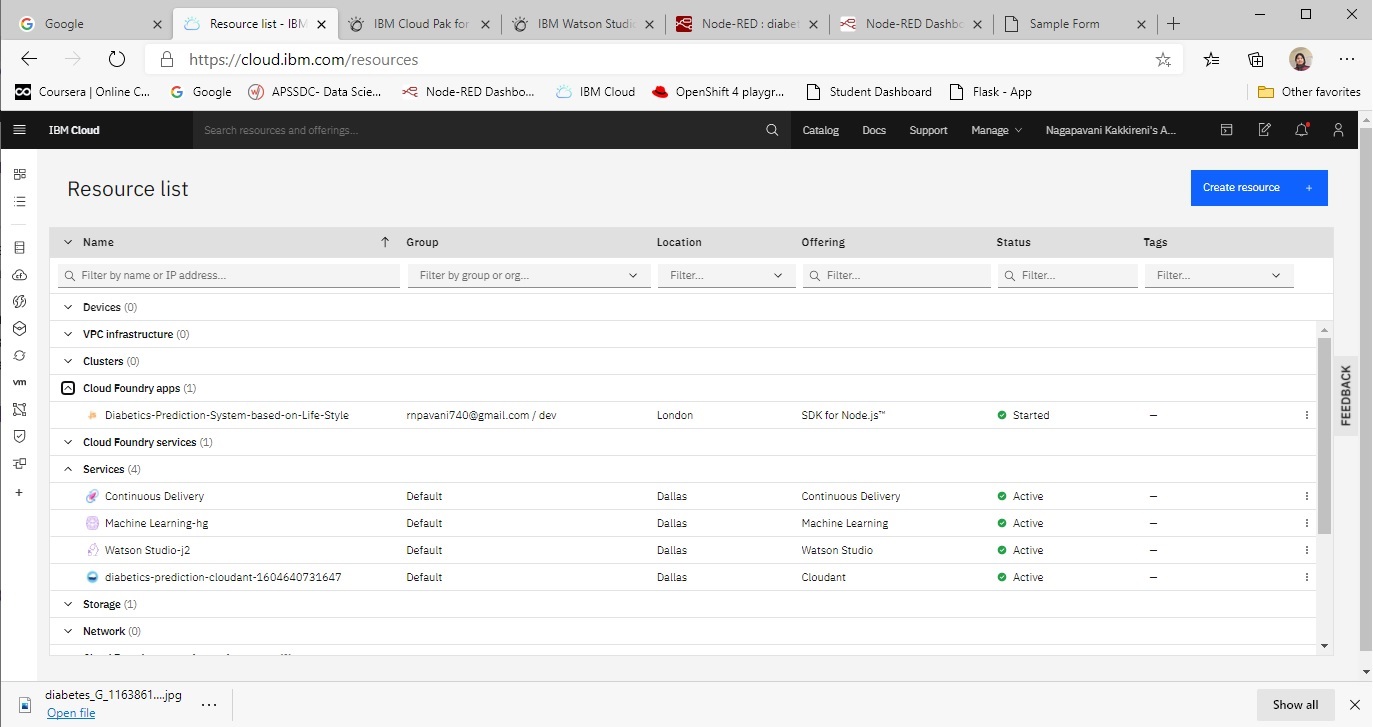


Figure 1:-IBM Cloud services, storage space and cloud foundry app

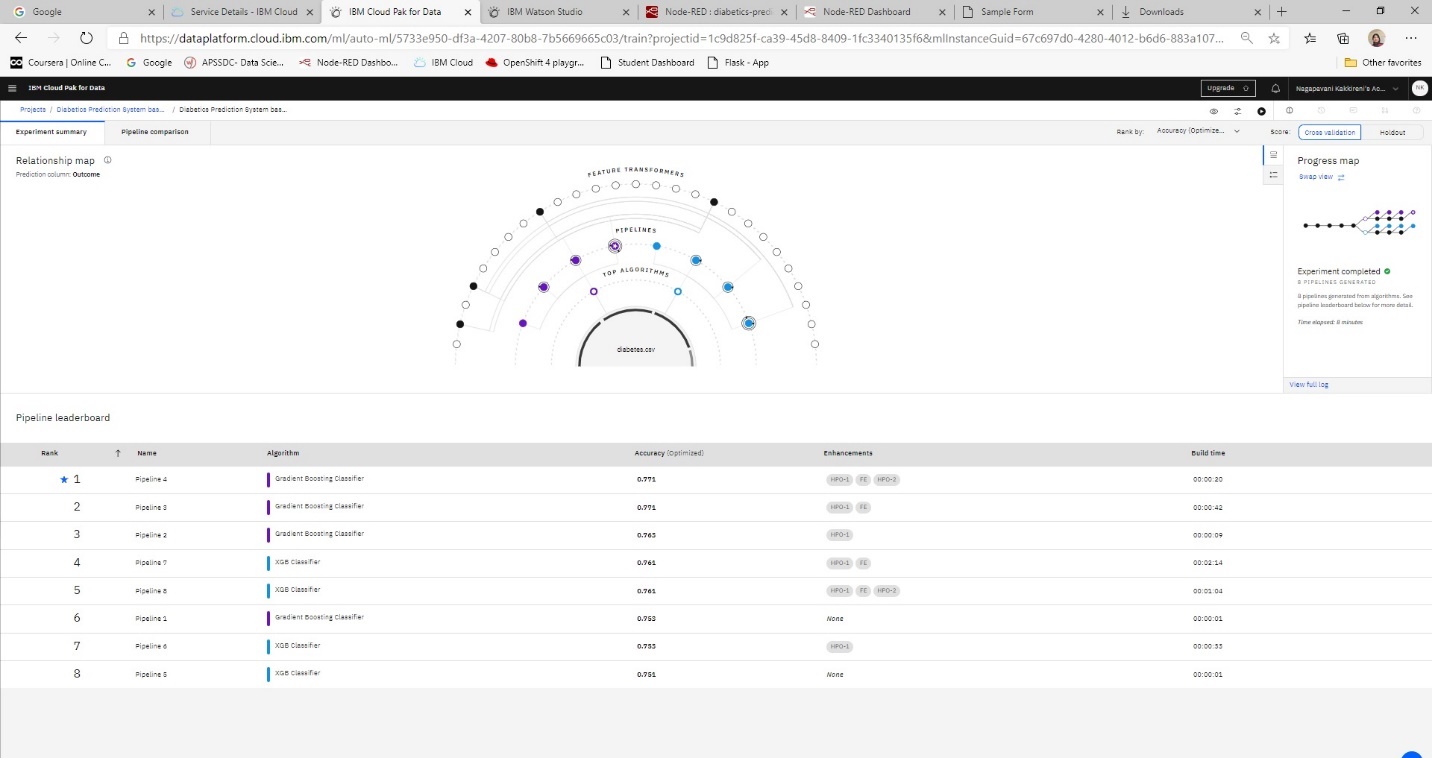


Figure 2:- Comparison between various algorithms

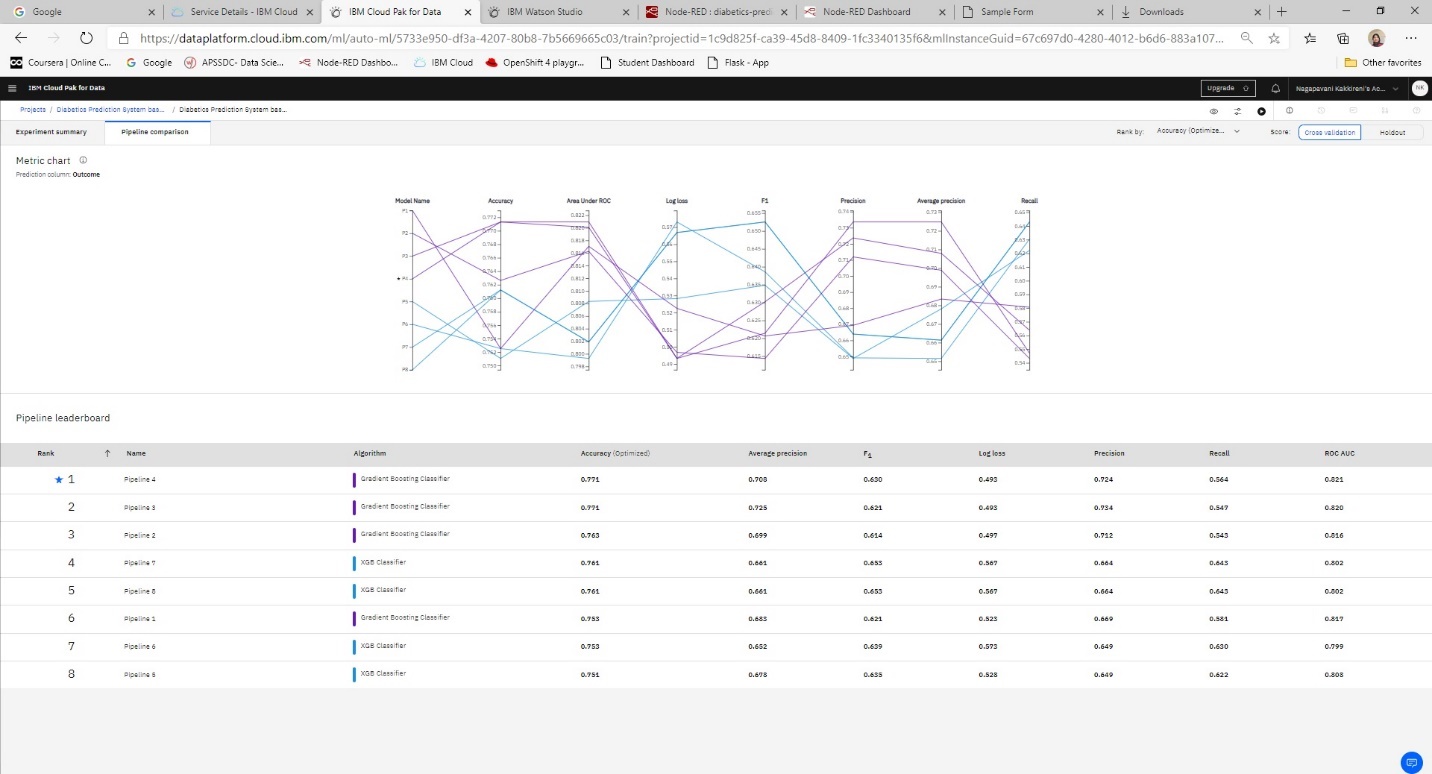


Figure 3:-Pipeline Comparison

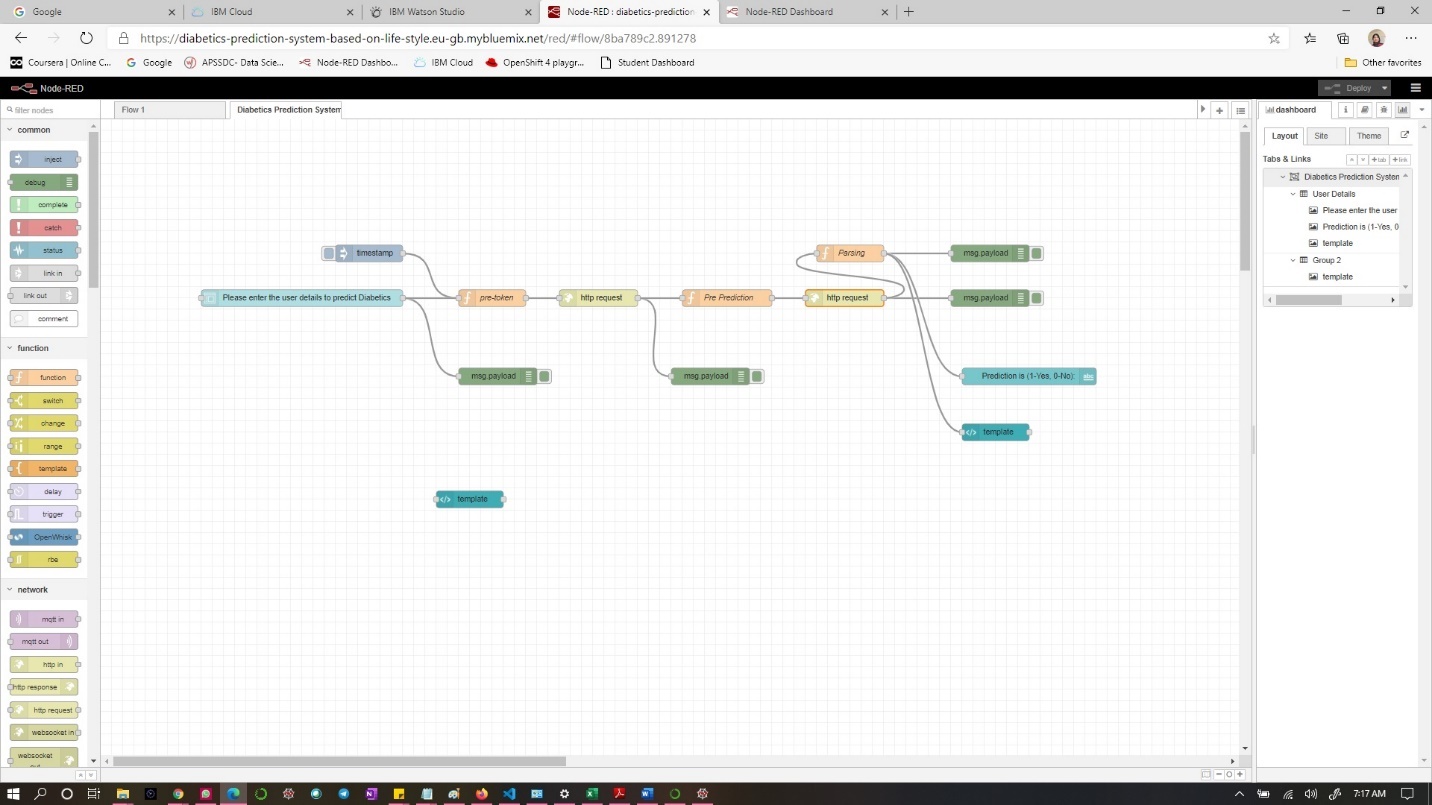
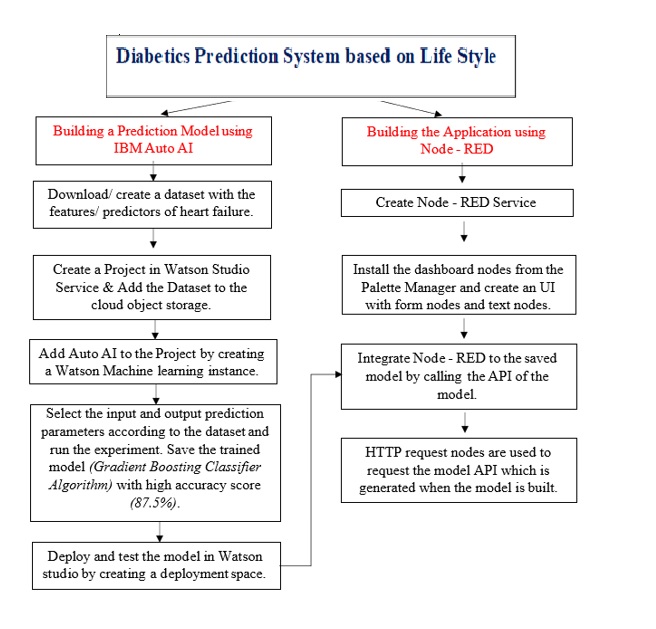


Figure 4:- Node-RED-Flow

1. FLOWCHART

Fig 5 : Flowchart

1. RESULT

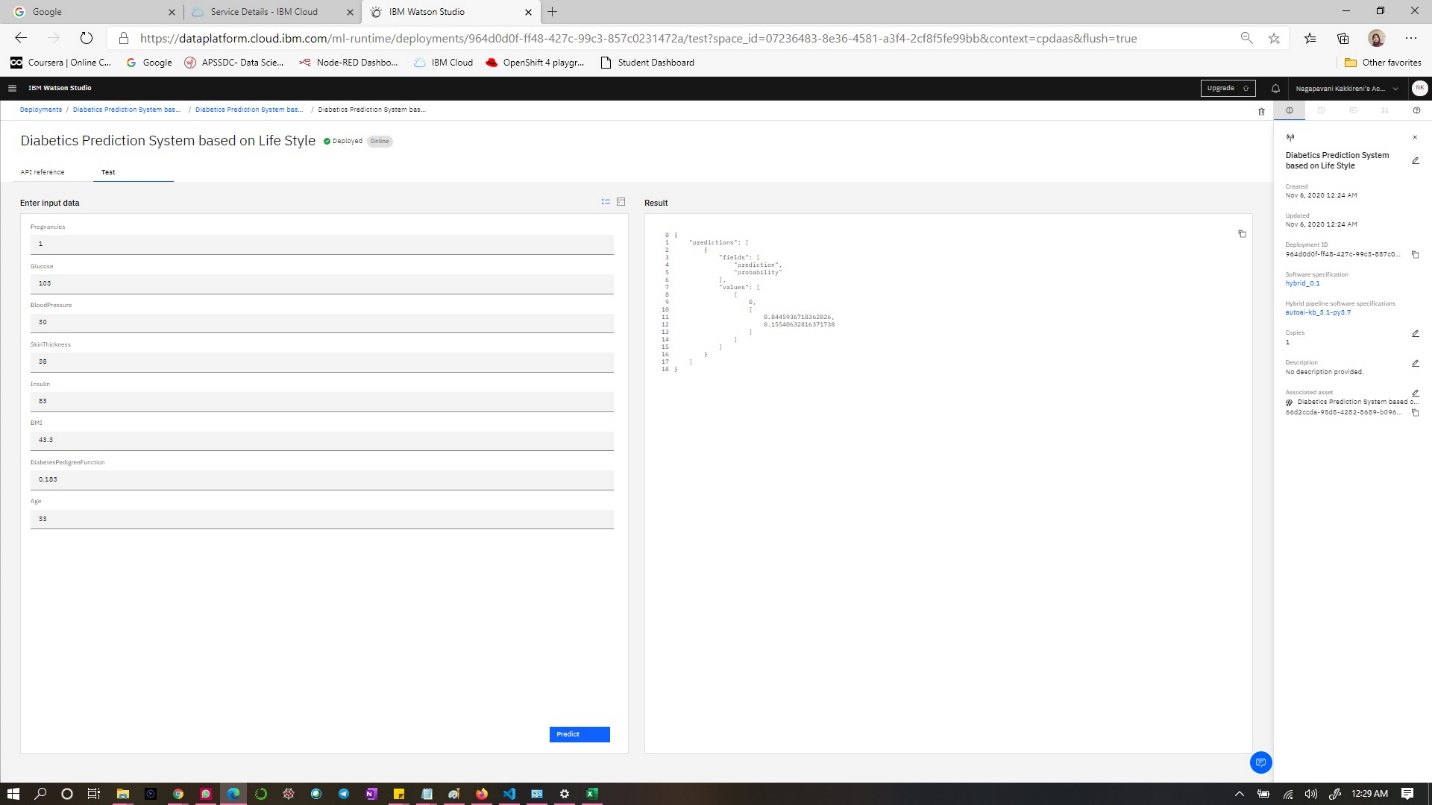


Figure 6:- Test\_Model\_Output(0)

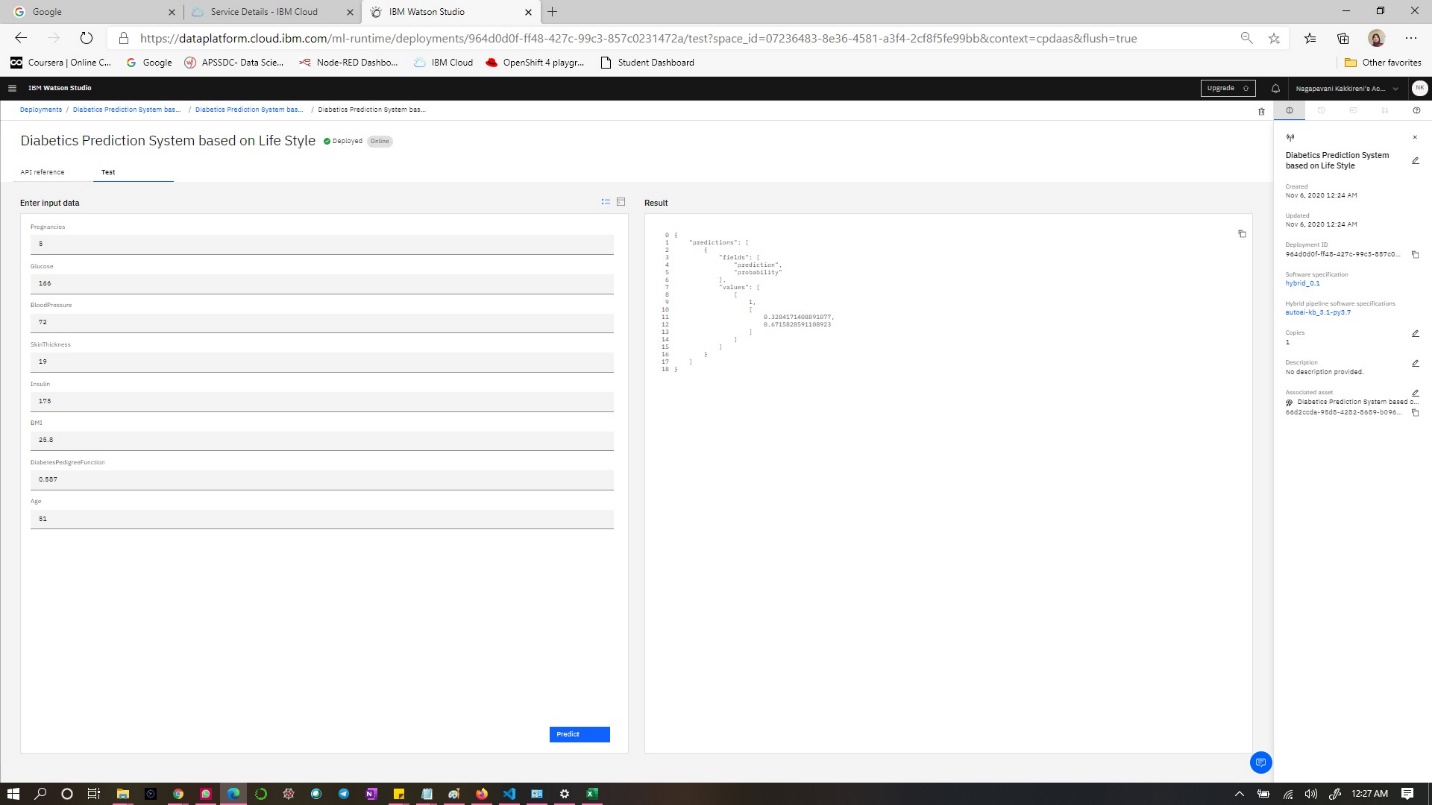


Figure 7:- Test\_Model\_Output(1)

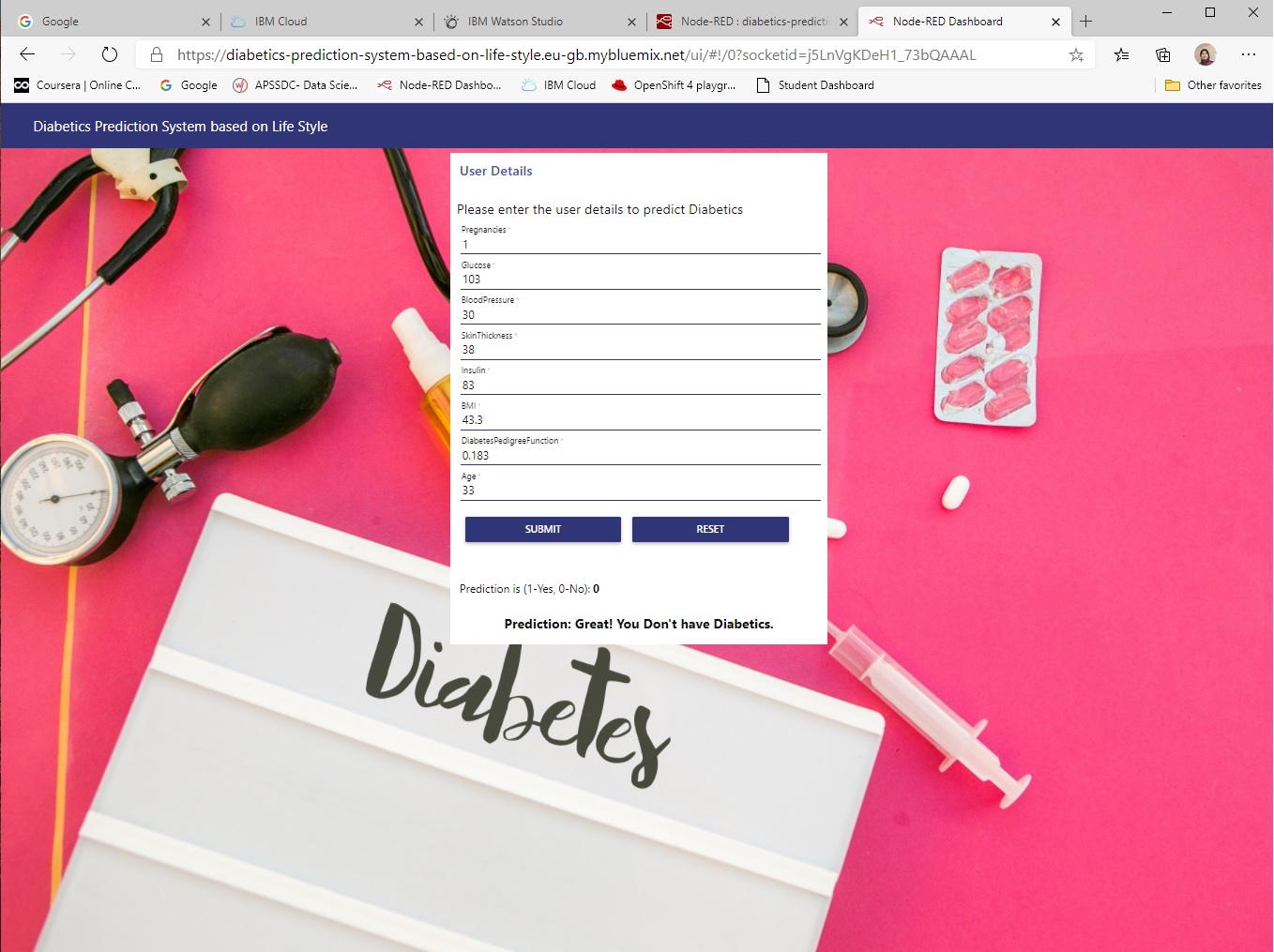


Figure 8:- Node -RED- App - O u t p u t ( 0)

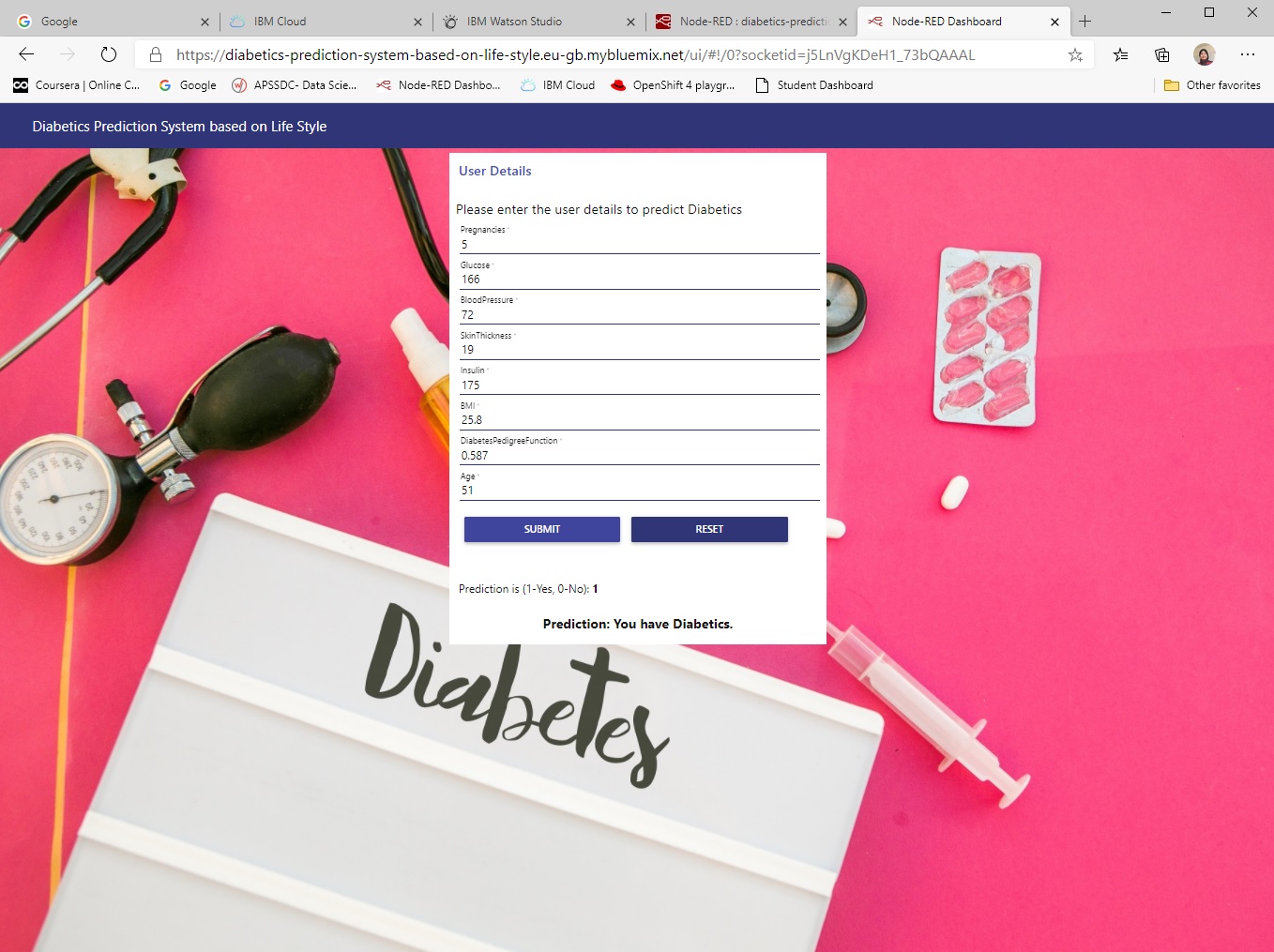
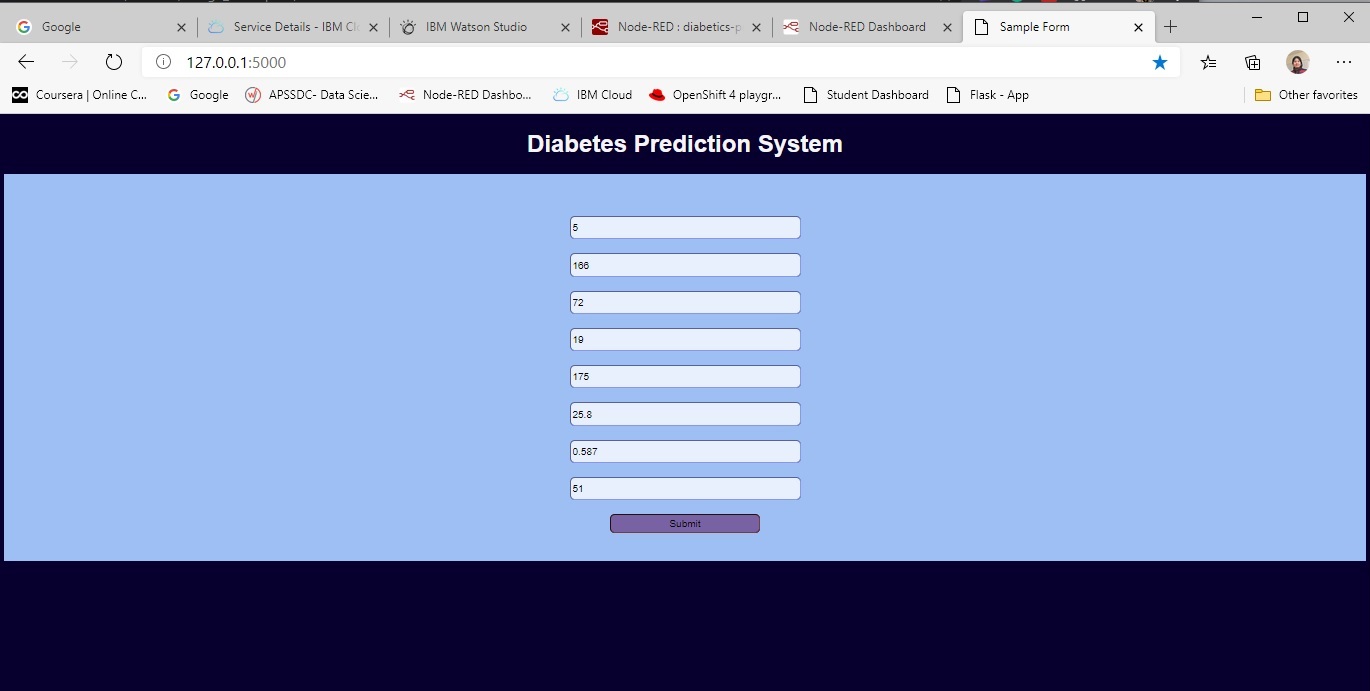
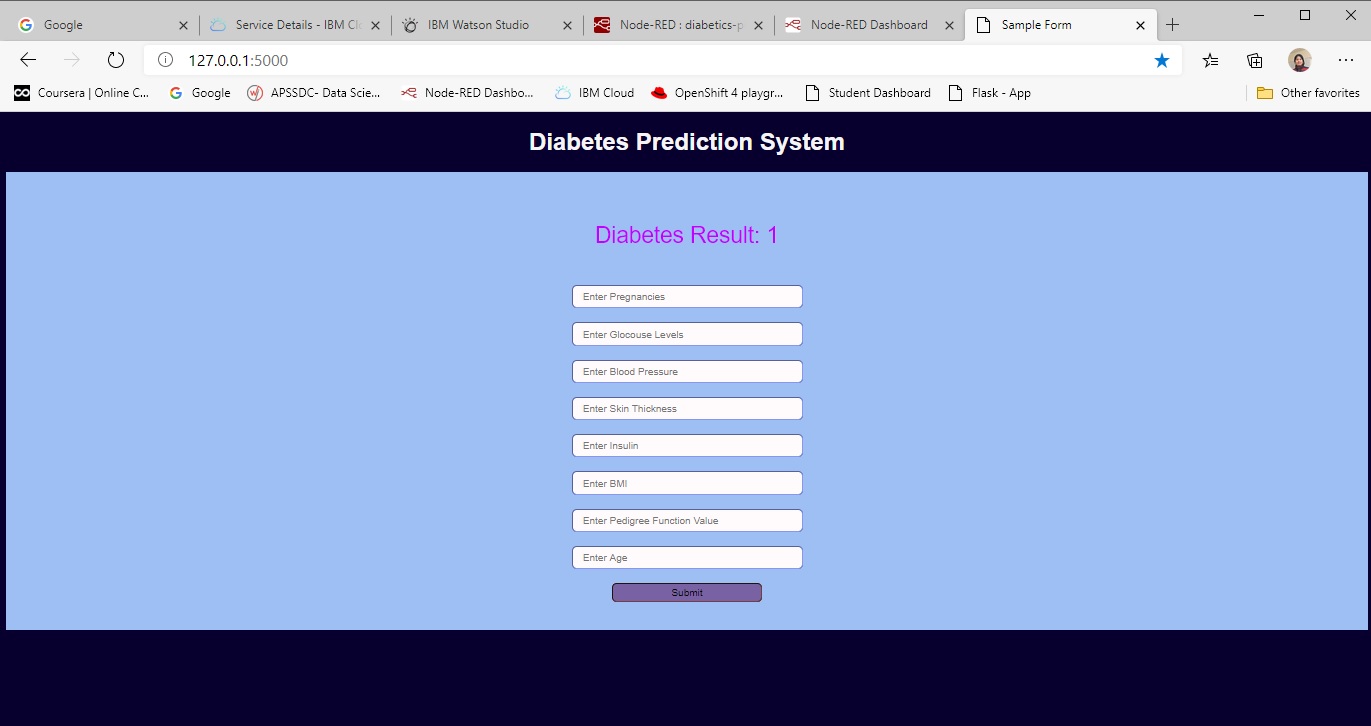
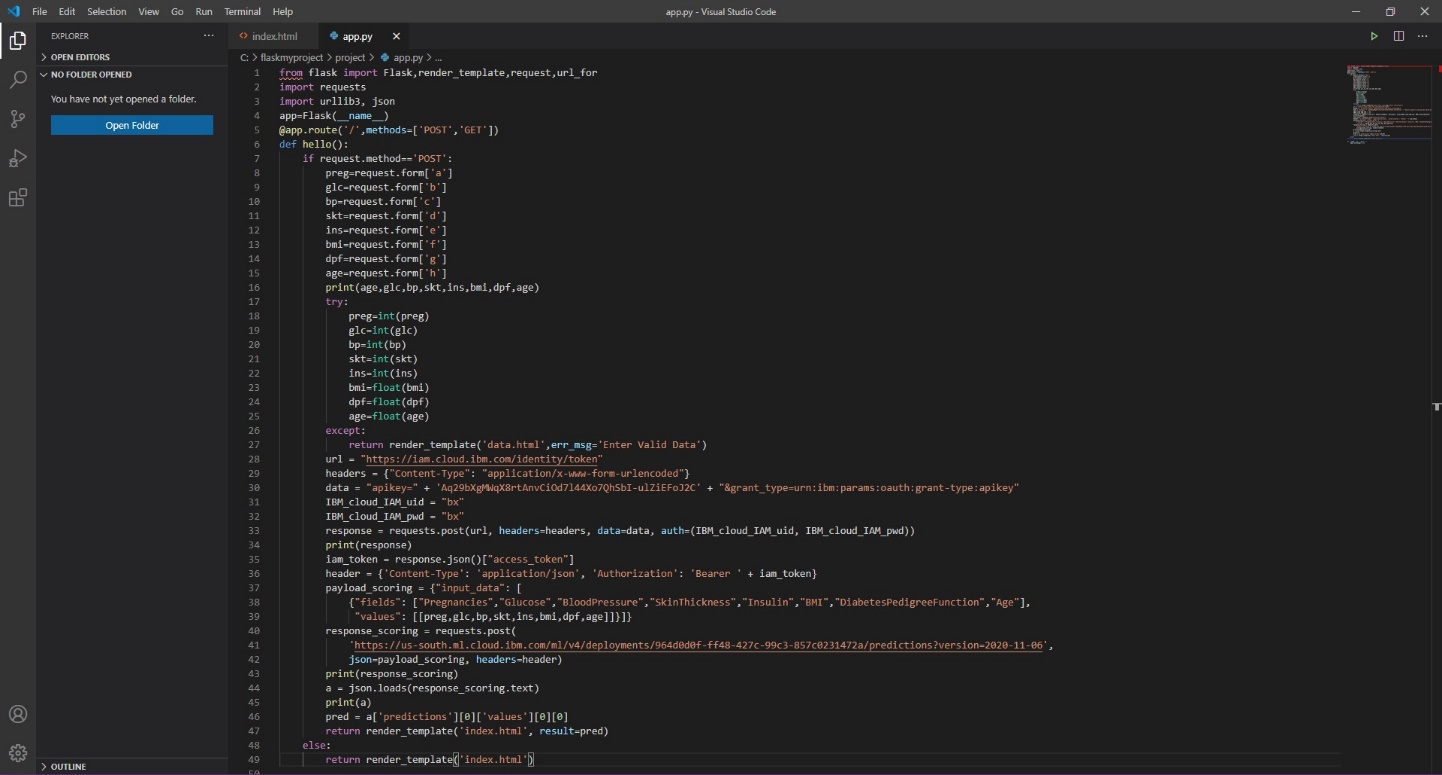
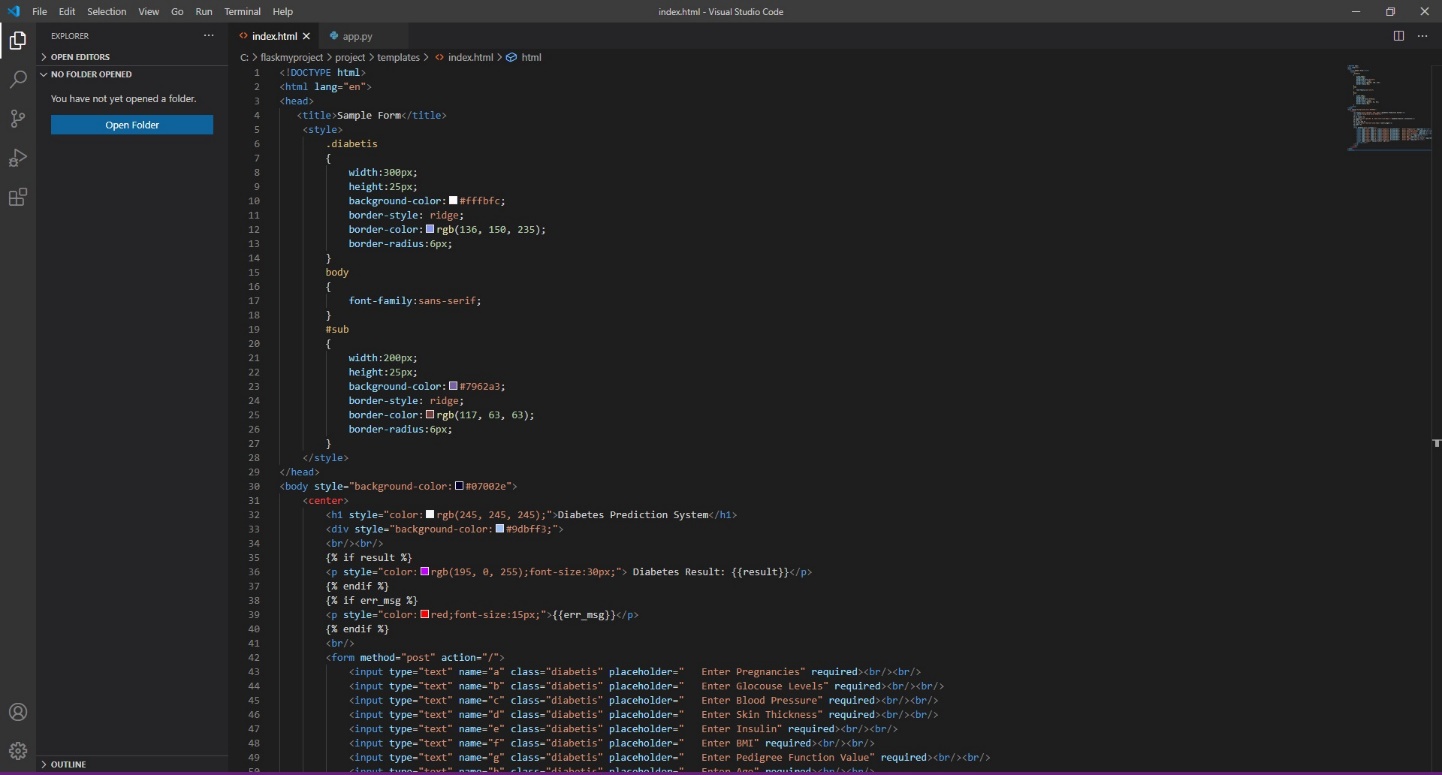


Figure 9:- Node-RED-App-Output(1)

Figure 10:- Flask-Application-Input

Figure 11:- Flask-Application-Output

 Figure 12:- app.py written in VS code

 Figure 13:- index.html written in VS code

1. ADVANTAGES & DISADVANTAGES
   1. Advantages

1. Identification without going to Diagnostic centers.

2. No need to waste money for tests

3. No waiting time and anxiety for reports

4. can Identify at early stage so that care can be taken.

* 1. Disadvantages

The disadvantage of the online prediction tool is its sensitivity and accuracy for clinical use. It completely depends on the dataset used to train the model for prediction.

1. Prediction can be wrong sometimes according to confusion matrix which may create worst consequences.

2. Accuracy can be less sometimes

3. Some more parameters can be needed sometimes for predicting accurately

1. CONCLUSION

Most of the population all around the world has diabetes. Predicting this disease at earlier stages plays vital role for treatment of patient. The Pima Indian diabetes dataset has been used in this model. The algorithm is best among others and got 77% accuracy. The obtained results appear to be very important for predicting diabetes accurately. The main theme is to prevent and cure diabetes and to improve the lives of all people affected by diabetes at faster rate and to help the doctors to start the treatments early.

1. FUTURE SCOPE

The Accuracy obtained is 77%.It can be increased by following different methods either by considering more parameters or huge data set can be used to train the model. The confusion matrix clearly states that there can be false positive, false negative cases which can cause to serious consequences and to be overcome in future researches.

1. BIBILOGRAPHY
   1. REFERENCES
      * https:/[/www.kaggle.com/datasets](http://www.kaggle.com/datasets)
      * <https://cloud.ibm.com/>
      * <https://cloud.ibm.com/catalog/services/watson-studio>
      * <https://cloud.ibm.com/developer/appservice/create-app>
      * [https://smartinternz.com/assets/Steps-to-be-followed-to-download- Watson-Studio-in-your-Local-System.pdf](https://smartinternz.com/assets/Steps-to-be-followed-to-download-%20Watson-Studio-in-your-Local-System.pdf)
   2. APPENDIX
2. Source code:
3. **App.py code**

from flask import Flask,render\_template,request,url\_for

import requests

import urllib3, json

app=Flask(\_\_name\_\_)

@app.route('/',methods=['POST','GET'])

def hello():

if request.method=='POST':

preg=request.form['a']

glc=request.form['b']

bp=request.form['c']

skt=request.form['d']

ins=request.form['e']

bmi=request.form['f']

dpf=request.form['g']

age=request.form['h']

print(age,glc,bp,skt,ins,bmi,dpf,age)

try:

preg=int(preg)

glc=int(glc)

bp=int(bp)

skt=int(skt)

ins=int(ins)

bmi=float(bmi)

dpf=float(dpf)

age=float(age)

except:

return render\_template('data.html',err\_msg='Enter Valid Data')

url = "https://iam.cloud.ibm.com/identity/token"

headers = {"Content-Type": "application/x-www-form-urlencoded"}

data = "apikey=" + 'Aq29bXgMWqX8rtAnvCiOd7l44Xo7QhSbI-ulZiEFoJ2C' + "&grant\_type=urn:ibm:params:oauth:grant-type:apikey"

IBM\_cloud\_IAM\_uid = "bx"

IBM\_cloud\_IAM\_pwd = "bx"

response = requests.post(url, headers=headers, data=data, auth=(IBM\_cloud\_IAM\_uid, IBM\_cloud\_IAM\_pwd))

print(response)

iam\_token = response.json()["access\_token"]

header = {'Content-Type': 'application/json', 'Authorization': 'Bearer ' + iam\_token}

payload\_scoring = {"input\_data": [

{"fields": ["Pregnancies","Glucose","BloodPressure","SkinThickness","Insulin","BMI","DiabetesPedigreeFunction","Age"],

"values": [[preg,glc,bp,skt,ins,bmi,dpf,age]]}]}

response\_scoring = requests.post(

'https://us-south.ml.cloud.ibm.com/ml/v4/deployments/964d0d0f-ff48-427c-99c3-857c0231472a/predictions?version=2020-11-06',

json=payload\_scoring, headers=header)

print(response\_scoring)

a = json.loads(response\_scoring.text)

print(a)

pred = a['predictions'][0]['values'][0][0]

return render\_template('index.html', result=pred)

else:

return render\_template('index.html')

if \_\_name\_\_ == '\_\_main\_\_':

app.run(debug=True)

1. **Index.html code**

<!DOCTYPE html>

<html lang="en">

<head>

<title>Sample Form</title>

<style>

.diabetis

{

width:300px;

height:25px;

background-color:#fffbfc;

border-style: ridge;

border-color:rgb(136, 150, 235);

border-radius:6px;

}

body

{

font-family:sans-serif;

}

#sub

{

width:200px;

height:25px;

background-color:#7962a3;

border-style: ridge;

border-color:rgb(117, 63, 63);

border-radius:6px;

}

</style>

</head>

<body style="background-color:#07002e">

<center>

<h1 style="color:rgb(245, 245, 245);">Diabetes Prediction System</h1>

<div style="background-color:#9dbff3;">

<br/><br/>

{% if result %}

<p style="color:rgb(195, 0, 255);font-size:30px;"> Diabetes Result: {{result}}</p>

{% endif %}

{% if err\_msg %}

<p style="color:red;font-size:15px;">{{err\_msg}}</p>

{% endif %}

<br/>

<form method="post" action="/">

<input type="text" name="a" class="diabetis" placeholder=" Enter Pregnancies" required><br/><br/>

<input type="text" name="b" class="diabetis" placeholder=" Enter Glocouse Levels" required><br/><br/>

<input type="text" name="c" class="diabetis" placeholder=" Enter Blood Pressure" required><br/><br/>

<input type="text" name="d" class="diabetis" placeholder=" Enter Skin Thickness" required><br/><br/>

<input type="text" name="e" class="diabetis" placeholder=" Enter Insulin" required><br/><br/>

<input type="text" name="f" class="diabetis" placeholder=" Enter BMI" required><br/><br/>

<input type="text" name="g" class="diabetis" placeholder=" Enter Pedigree Function Value" required><br/><br/>

<input type="text" name="h" class="diabetis" placeholder=" Enter Age" required><br/><br/>

<input type="submit" value="Submit" id="sub">

<br/><br/><br/>

</form>

</div>

</center>

</body>

</html>

1. **IBM Auto AI code**

Trusted

Jupyter Server: local

Python 3: Not Started

################################################################################ #Licensed Materials - Property of IBM #(C) Copyright IBM Corp. 2020 #US Government Users Restricted Rights - Use, duplication disclosure restricted #by GSA ADP Schedule Contract with IBM Corp. ################################################################################

The auto-generated notebooks are subject to the International License Agreement for Non-Warranted Programs (or equivalent) and License Information document for Watson Studio Auto-generated Notebook (License Terms), such agreements located in the link below. Specifically, the Source Components and Sample Materials clause included in the License Information document for Watson Studio Auto-generated Notebook applies to the auto-generated notebooks. By downloading, copying, accessing, or otherwise using the materials, you agree to the License Terms. http://www14.software.ibm.com/cgi-bin/weblap/lap.pl?li\_formnum=L-AMCU-BHU2B7&title=IBM%20Watson%20Studio%20Auto-generated%20Notebook%20V2.1

IBM AutoAI Auto-Generated Notebook v1.14.2

Note: Notebook code generated using AutoAI will execute successfully. If code is modified or reordered,

there is no guarantee it will successfully execute. This pipeline is optimized for the original dataset.

The pipeline may fail or produce sub-optimium results if used with different data. For different data,

please consider returning to AutoAI Experiments to generate a new pipeline. Please read our documentation

for more information:

(Cloud Platform) https://dataplatform.cloud.ibm.com/docs/content/wsj/analyze-data/autoai-notebook.html . (Cloud Pak For Data) https://www.ibm.com/support/knowledgecenter/SSQNUZ\_3.0.0/wsj/analyze-data/autoai-notebook.html .

Before modifying the pipeline or trying to re-fit the pipeline, consider:

The notebook converts dataframes to numpy arrays before fitting the pipeline

(a current restriction of the preprocessor pipeline). The known\_values\_list is passed by reference

and populated with categorical values during fit of the preprocessing pipeline. Delete its members before re-fitting.

Representing Pipeline\_4

1. Set Up

If lightgbm or xgboost installation fails, please follow:

lightgbm docs

xgboost docs

[-]

try:

import autoai\_libs

except Exception as e:

import subprocess

out = subprocess.check\_output('pip install autoai-libs'.split(' '))

for line in out.splitlines():

print(line)

import autoai\_libs

import sklearn

[-]

# compose a decorator to assist pipeline instantiation via import of modules and installation of packages

def decorator\_retries(func):

def install\_import\_retry(\*args, \*\*kwargs):

retries = 0

successful = False

failed\_retries = 0

while retries < 100 and failed\_retries < 10 and not successful:

retries += 1

failed\_retries += 1

try:

2. Compose Pipeline

[-]

# metadata necessary to replicate AutoAI scores with the pipeline

\_input\_metadata = {'separator': ',', 'excel\_sheet': 0, 'target\_label\_name': 'Outcome', 'learning\_type': 'classification', 'subsampling': None, 'pos\_label': 1, 'pn': 'P4', 'cv\_num\_folds': 3, 'holdout\_fraction': 0.1, 'optimization\_metric': 'accuracy', 'random\_state': 33, 'data\_source': ''}

# define a function to compose the pipeline, and invoke it

@decorator\_retries

def compose\_pipeline():

import numpy

from numpy import nan

3. Extract needed parameter values from AutoAI run metadata

[-]

# Metadata used in retrieving data and computing metrics. Customize as necessary for your environment.

#data\_source='replace\_with\_path\_and\_csv\_filename'

target\_label\_name = \_input\_metadata['target\_label\_name']

learning\_type = \_input\_metadata['learning\_type']

optimization\_metric = \_input\_metadata['optimization\_metric']

random\_state = \_input\_metadata['random\_state']

cv\_num\_folds = \_input\_metadata['cv\_num\_folds']

holdout\_fraction = \_input\_metadata['holdout\_fraction']

if 'data\_provenance' in \_inpu

4. Create dataframe from dataset in Cloud Object Storage

[-]

# @hidden\_cell

# The following code contains the credentials for a file in your IBM Cloud Object Storage.

# You might want to remove those credentials before you share your notebook.

credentials\_0 = {

'ENDPOINT': 'https://s3-api.us-geo.objectstorage.softlayer.net',

'IBM\_AUTH\_ENDPOINT': 'https://iam.bluemix.net/oidc/token/',

'APIKEY': 'UEESapZQQ3jWKIOk\_eUF6Oz9t-ARMCPLvx4AKCYUUbDt',

'BUCKET': 'diabeticspredictionsystembasedonl-donotdelete-pr-mgqupj7gse7154',

'FILE': 'diabetes.

[-]

# Read the data as a dataframe

import pandas as pd

csv\_encodings=['UTF-8','Latin-1'] # supplement list of encodings as necessary for your data

df = None

readable = None # if automatic detection fails, you can supply a filename here

# First, obtain a readable object

# Cloud Object Storage data access

# Assumes COS credentials are in a dictionary named 'credentials\_0'

5. Preprocess Data

[-]

# Drop rows whose target is not defined

target = target\_label\_name # your target name here

if learning\_type == 'regression':

df[target] = pd.to\_numeric(df[target], errors='coerce')

df.dropna('rows', how='any', subset=[target], inplace=True)

[-]

# extract X and y

df\_X = df.drop(columns=[target])

df\_y = df[target]

[-]

# Detach preprocessing pipeline (which needs to see all training data)

preprocessor\_index = -1

preprocessing\_steps = []

for i, step in enumerate(pipeline.steps):

preprocessing\_steps.append(step)

if step[0]=='preprocessor':

preprocessor\_index = i

break

#if len(pipeline.steps) > preprocessor\_index+1 and pipeline.steps[preprocessor\_index + 1][0] == 'cognito':

#preprocessor\_index += 1

[-]

# Preprocess X

# preprocessor should see all data for cross\_validate on the remaining steps to match autoai scores

known\_values\_list.clear() # known\_values\_list is filled in by the preprocessing\_pipeline if needed

preprocessing\_pipeline.fit(df\_X.values, df\_y.values)

X\_prep = preprocessing\_pipeline.transform(df\_X.values)

6. Split data into Training and Holdout sets

[-]

# determine learning\_type and perform holdout split (stratify conditionally)

if learning\_type is None:

# When the problem type is not available in the metadata, use the sklearn type\_of\_target to determine whether to stratify the holdout split

# Caution: This can mis-classify regression targets that can be expressed as integers as multiclass, in which case manually override the learning\_type

from sklearn.utils.multiclass import type\_of\_target

if type\_of\_target(df\_y.values) in ['m

7. Generate features via Feature Engineering pipeline

[-]

#Detach Feature Engineering pipeline if next, fit it, and transform the training data

fe\_pipeline = None

if pipeline.steps[0][0] == 'cognito':

try:

fe\_pipeline = Pipeline(steps=[pipeline.steps[0]])

X = fe\_pipeline.fit\_transform(X, y)

X\_holdout = fe\_pipeline.transform(X\_holdout)

pipeline.steps = pipeline.steps[1:]

except IndexError:

try:

8. Additional setup: Define a function that returns a scorer for the target's positive label

[-]

# create a function to produce a scorer for a given positive label

def make\_pos\_label\_scorer(scorer, pos\_label):

kwargs = {'pos\_label':pos\_label}

for prop in ['needs\_proba', 'needs\_threshold']:

if prop+'=True' in scorer.\_factory\_args():

kwargs[prop] = True

if scorer.\_sign == -1:

kwargs['greater\_is\_better'] = False

from sklearn.metrics import make\_scorer

scorer=make\_scorer(scorer.\_score\_func, \*\*kwargs)

9. Fit pipeline, predict on Holdout set, calculate score, perform cross-validation

[-]

# fit the remainder of the pipeline on the training data

pipeline.fit(X,y)

[-]

# predict on the holdout data

y\_pred = pipeline.predict(X\_holdout)

[-]

# compute score for the optimization metric

# scorer may need pos\_label, but not all scorers take pos\_label parameter

from sklearn.metrics import get\_scorer

scorer = get\_scorer(optimization\_metric)

score = None

#score = scorer(pipeline, X\_holdout, y\_holdout) # this would suffice for simple cases

pos\_label = None # if you want to supply the pos\_label, specify it here

if pos\_label is None and 'pos\_label' in \_input\_metadata:

pos\_label=\_input\_metadata['pos\_label']

try:

[-]

# cross\_validate pipeline using training data

from sklearn.model\_selection import cross\_validate

from sklearn.model\_selection import StratifiedKFold, KFold

if learning\_type == 'classification':

fold\_generator = StratifiedKFold(n\_splits=cv\_num\_folds, random\_state=random\_state)

else:

fold\_generator = KFold(n\_splits=cv\_num\_folds, random\_state=random\_state)

cv\_results = cross\_validate(pipeline, X, y, cv=fold\_generator, scoring={optimization\_metric:scorer}, return\_train\_score=True)

import num

[-]

cv\_results