DIABETES PREDICTION SYSTEM BASED ON LIFE STYLE

Prepared by:

Aishwarya Srinivas

Information Science and Engineering

HKBK College of Engineering

Bangalore-560045

GitHub repository link: https://github.com/SmartPracticeschool/SPS-6756-Diabetics-Prediction-System-based-on-Life-Style

INDEX

CHAPTER 1

INTRODUCTION

1.1 OVERVIEW

Diabetes is a chronic disease with the potential to cause a worldwide health care crisis. According to International Diabetes Federation 382 million people are living with diabetes across the whole world. By 2035, this will be doubled as 592 million. Diabetes mellitus or simply diabetes is a disease caused due to the increased level of blood glucose. Various traditional methods, based on physical and chemical tests, are available for diagnosing diabetes. However, early prediction of diabetes is quite challenging task for medical practitioners due to complex interdependence on various factors as diabetes affects human organs such as kidney, eye, heart, nerves, foot etc.

In this project, 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.

1.2 PURPOSE

Machine learning is an emerging scientific field in data science dealing with the ways in which machines learn from experience. The objective of this project is to build a predictive machine learning model to predict based on diagnostic measurements whether a patient has diabetes.

This is a binary classification project with supervised learning. The aim of this project is to develop a system which can perform early prediction of diabetes for a patient with a higher accuracy by combining the results of different machine learning techniques.

CHAPTER 2

ERATULITRE SURVEY

2.1 EXISTING PROBLEM

Accuracy has always been the main problem for any newly developed system. When it comes to the field of health, proper results are very crucial for the doctor to take important decisions for improving the patients' health. For accurate outputs, we use the machine learning algorithms. As we know, machine learning is a process where the machine learns from its past decisions and performs tasks, without requiring explicit programming. So, it is a well suited approach for Diabetes Prediction System.
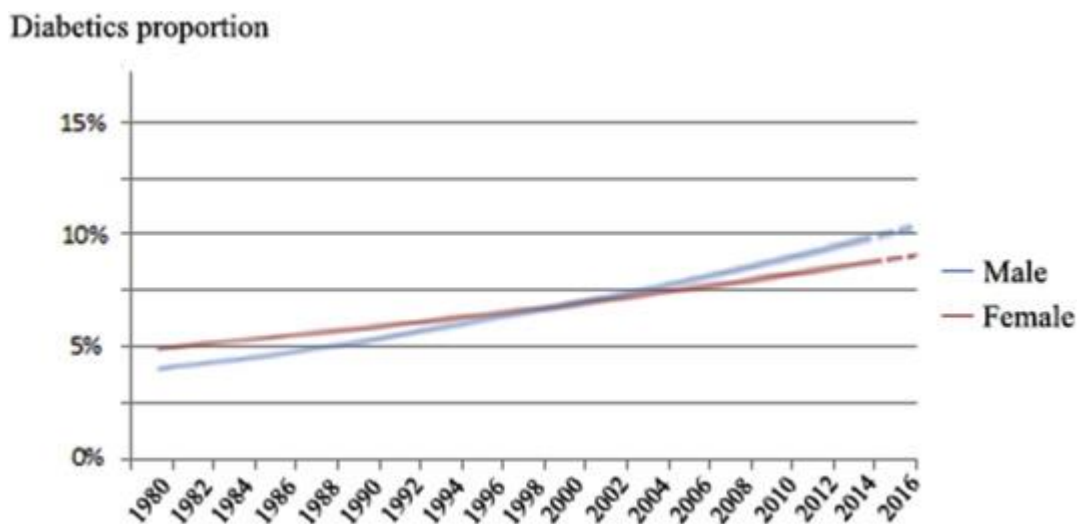

Fig 1: Trend of Diabetes proportion over the years

In the past 30 years of development in China, with rising number of diabetics, people have started to realize that this chronic disease has deeply impacted every family and everyone's daily life. There is an ascending trend in the proportion of diabetics in the general population.

In 2017, National Diabetes Statistic Report for Center Disease Control and Prevention (CDC), gives the facts give an account of the United States that 30.3 million individuals have diabetes, among that 23.1 are analysed and 7.2 million are undiscovered individuals. • In 2018, the American Diabetes Association models of therapeutic care in diabetes discharges a

report about "Order and finding of diabetes" which incorporates the arrangement of diabetes, diabetes care, treatment objectives, criteria for conclusion test ranges and dangers esteems, chance engaged with diabetes. • In 2017, Global provides details regarding Diabetes by world wellbeing association; it expresses the weight of diabetes, hazard components and inconveniences of diabetes. Likewise, gives the data about counteracting diabetes in individuals with high hazard and overseeing diabetes at beginning times with fundamental solutions to be taken.

The need for a system which accurately predicts the outcome is very essential as it pertains to the life of a person, and diseases like diabetes, if not detected in early stages, can prove to be very fatal.

As we all know, the number of diabetics is large, and it is continuously increasing. Additionally, most people know little about their health quality. Therefore, believe it is necessary to establish a model that can predict if the patient has traces of diabetes based on various parameters (pertaining to their health) from the first examination time for the high-risk DM group.

2.2 PROPOSED SOLUTION

In recent years, using the Machine Learning technique has been used with increasing frequency to predict the possibility of disease. Many algorithms and toolkits have been created and studied by researchers. These have highlighted the tremendous potential of this research field.

Machine Learning has been successfully applied to various fields in human society, such as weather prognosis, market analysis, engineering diagnosis, and customer relationship management. However, the application in disease prediction and medical data analysis still has room for improvement. For example, every hospital possesses a plethora of patient's basic and medical information, and it is essential to revise, supplement, and extract meaningful knowledge from these data to support clinical analysis and diagnosis. It is reasonable to believe that there are various valuable patterns and waiting for researchers to explore them.

In this project, 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. The "Binary Classification" is used to predict the output, i.e. the binary numbers (0 and 1) are used as output prediction. If there is detection of presence of diabetes the result is 1.

CHAPTER 3

THEORITICAL ANALYSIS

3.1 BLOCK DIAGRAM

WATSON STUDIO

IBM CLOUD OBJECT STORAGE

FLASK APPLICATION
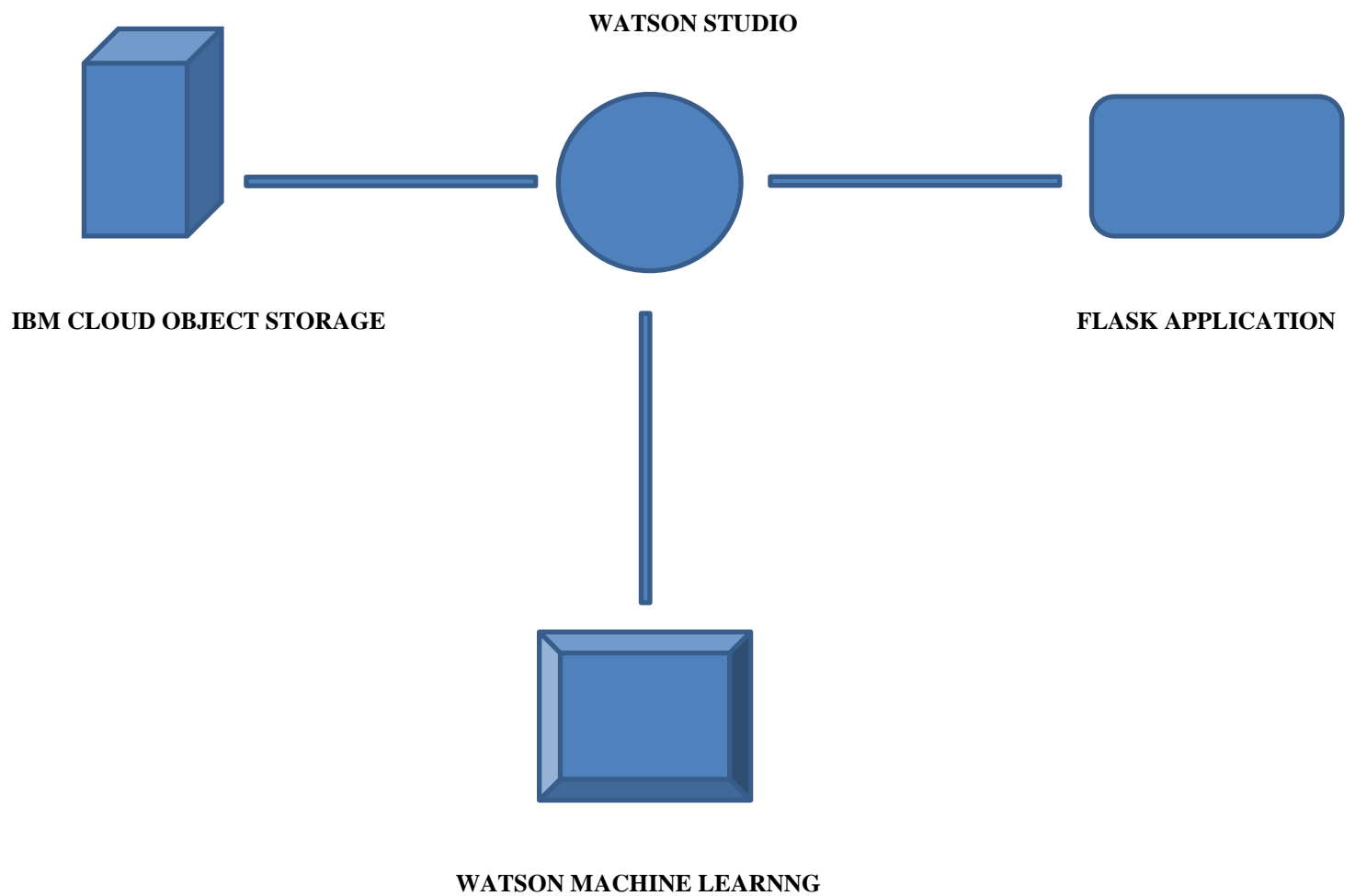
WATSON MACHINE LEARNNG

Fig 2 : Block Diagram of Diabetes Prediction System

3.2 INTRODUCTION TO MACHINE LEARNING

Machine Learning (ML) is a subset of AI. Machine Learning is a latest buzzword floating around. It deserves to, as it is one of the most interesting subfield of Computer Science Machine Learning is the most in-demand technology in today's market. Here's a list of reasons why Machine Learning is so important:

- Increase in Data Generation: Due to excessive production of data, we need a method that can be used to structure, analyse and draw useful insights from data. This is where Machine Learning comes in. It uses data to solve problems and find solutions to the most complex tasks faced by organizations.

- Improve Decision Making: By making use of various algorithms, Machine Learning can be used to make better business decisions. For example, Machine Learning is used to forecast sales, predict downfalls in the stock market, identify risks and anomalies, etc.

- Uncover patterns & trends in data: Finding hidden patterns and extracting key insights from data is the most essential part of Machine Learning. By building predictive models and using statistical techniques, Machine Learning allows you to dig beneath the surface and explore the data at a minute scale. Understanding data and extracting patterns manually will take days, whereas Machine Learning algorithms can perform such computations in less than a second.

- Solve complex problems: From detecting the genes linked to the deadly ALS disease to building self driving cars, Machine Learning can be used to solve the most complex problems.

3.3 SOFTWARE DESIGNING

We develop an end-to-end web application that predicts the probability of females having diabetes. The application must be built with Python-Flask with the machine learning model trained & deployed on IBM Watson Studio.

The software requirements are:

- IBM Watson Studio
- IBM Cloud
- Node-Red
- Visual Studio Code

The languages used are:

- Python
- HTML

3.3.1 IBM WATSON STUDIO

IBM Watson Studio helps you build and scale AI with trust and transparency by automating AI lifecycle management. It empowers you to organize data, build, run and manage AI models, and optimize decisions across any cloud using IBM Cloud Pak for Data.

With Watson Studio, we are doing the following:

- Extending the capabilities we provide around deep learning, including TensorFlow scoring
- Allowing you to access pre-trained models from the Watson Services, such as Watson Visual Recognition
- Enabling you to bring in non-structured data
- Further automating and providing insight into model management
- Continuing to provide you with a choice of best-in-breed data science/ML tools
- Enabling you to visualize the insights with dynamic dashboards

3.3.2 IBM CLOUD

The IBM cloud platform combines platform as a service (PaaS) with infrastructure as a service (IaaS) to provide an integrated experience. The platform scales and supports both small development teams and organizations, and large enterprise businesses. Globally deployed across data centres around the world, the solution you build on IBM Cloud spins up fast and performs reliably in a tested and supported environment you can trust.

Whether you need to migrate apps to the cloud, modernize your existing apps by using cloud services, ensure data resiliency against regional failure, or leverage new paradigms and deployment topologies to innovate and build your cloud-native apps, the platform's open architecture is built to accommodate your use case.

Using the IBM Cloud catalog, we can perform the following tasks:

- Create resources

- Manage our resources

- Add Cloud Foundary services and develop Cloud Foundary apps(Node-red)

- Add all necessary services.

3.3.3 NODE-RED

Node-RED is a programming tool for wiring together hardware devices, APIs and online services in new and interesting ways. It provides a browser-based editor that makes it easy to wire together flows using the wide range of nodes in the palette that can be deployed to its runtime in a single-click.

After creating the node-network (e.g.: sample.json file), import the JSON file and then deploy the file to save it. We can then easily give the input values for our project and determine the output, and also see the request-flow among the nodes, and how exactly the response is obtained.

Node-RED is based on Node. js which is JavaScript. JavaScript is the language of the web, Node.js allows you to use JavaScript as a server (back-end) language.

3.3.4 VISUAL STUDIO CODE

Visual Studio Code is a streamlined code editor with support for development operations like debugging, task running, and version control. It aims to provide just the tools a developer needs for a quick code-build-debug cycle and leaves more complex workflows to fuller featured IDEs, such as Visual Studio IDE.

We can work with different coding languages like Python, HTML; all clubbed in a single folder used in our project, and execute the programs in the VS Code terminal.

CHAPTER 4

EXPERIMENTAL INVESTIGATIONS

4.1 DATASET

1	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreeFunction	Age	Outcome
2	6	148	72	35	0	33.6	0.627	50	1
3	1	85	66	29	0	26.6	0.351	31	0
4	8	183	64	0	0	23.3	0.672	32	1
5	1	89	66	23	94	28.1	0.167	21	0
6	0	137	40	35	168	43.1	2.288	33	1
7	5	116	74	0	0	25.6	0.201	30	0
8	3	78	50	32	88	31	0.248	26	1
9	10	115	0	0	0	35.3	0.134	29	0
10	2	197	70	45	543	30.5	0.158	53	1
11	8	125	96	0	0	0	0.232	54	1
12	4	110	92	0	0	37.6	0.191	30	0
13	10	168	74	0	0	38	0.537	34	1
14	10	139	80	0	0	27.1	1.441	57	0
15	1	189	60	23	846	30.1	0.398	59	1

Fig 3: Diabetes Data set

This initial data is collected from Kaggle. It will be used in comparative analysis of different machine learning techniques, where the best technique will be applied to predict the outcome.
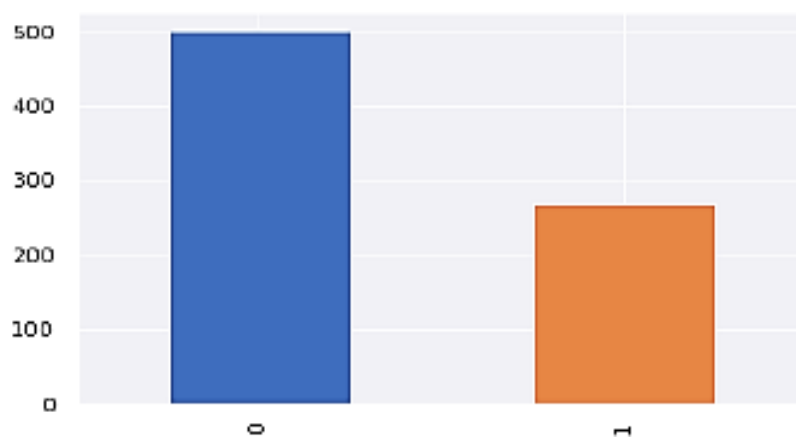

Fig 4: Graph of outcome predictions

The above graph shows that the data is biased towards data points having outcome value as 0 where it means that diabetes was not present actually. The number of non-diabetics is almost twice the number of diabetic patients

4.2 CREATING AND TRAINING A DATA MODEL

- A model based on the data set is read and split into training data and test data.

- Then the training data is read and pre-processing takes place before the model is selected.

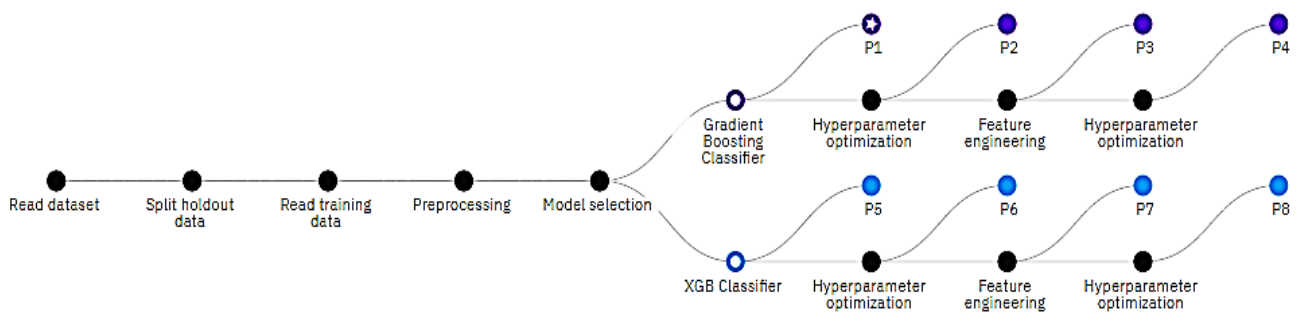- Different types of classifier algorithms are used and as a result, pipelines are developed.

Fig 5: Progress map of the data model

90% of data from the data set is chosen as training data.8 pipelines are generated from the Gradient Boosting Classifier and XGB Classifier as shown below:
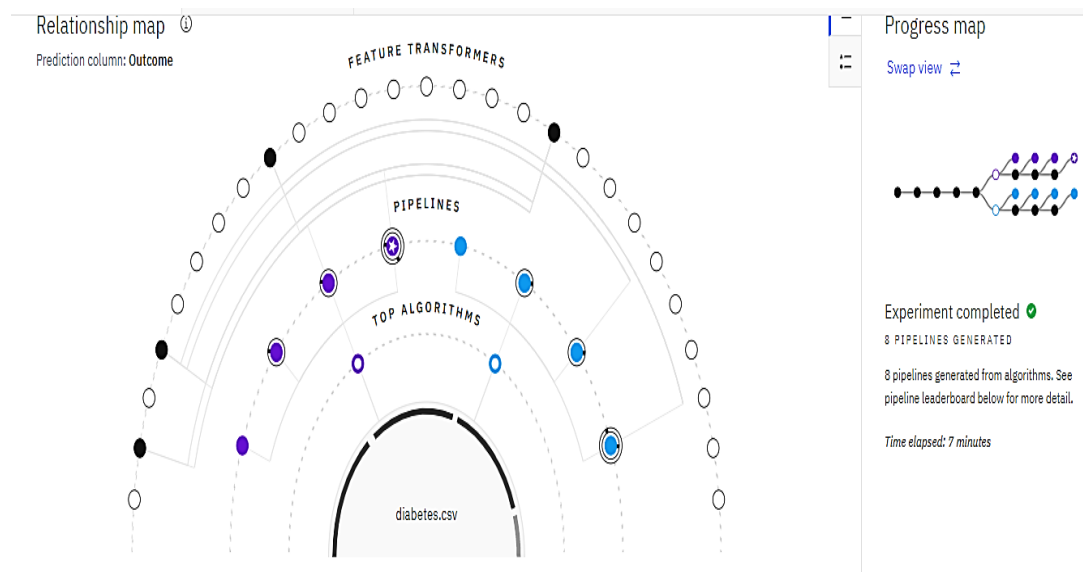
Fig 6: Generation of pipelines using different algorithms

4.3 DEPLOYING THE MODEL TO IBM CLOUD

The trained model is deployed on the IBM Cloud, to check if the predictions match the values of the dataset being used.

Fig 5: Model-deployment

After the model is deployed to the cloud, test data is selected from the data set and the outcome is tested
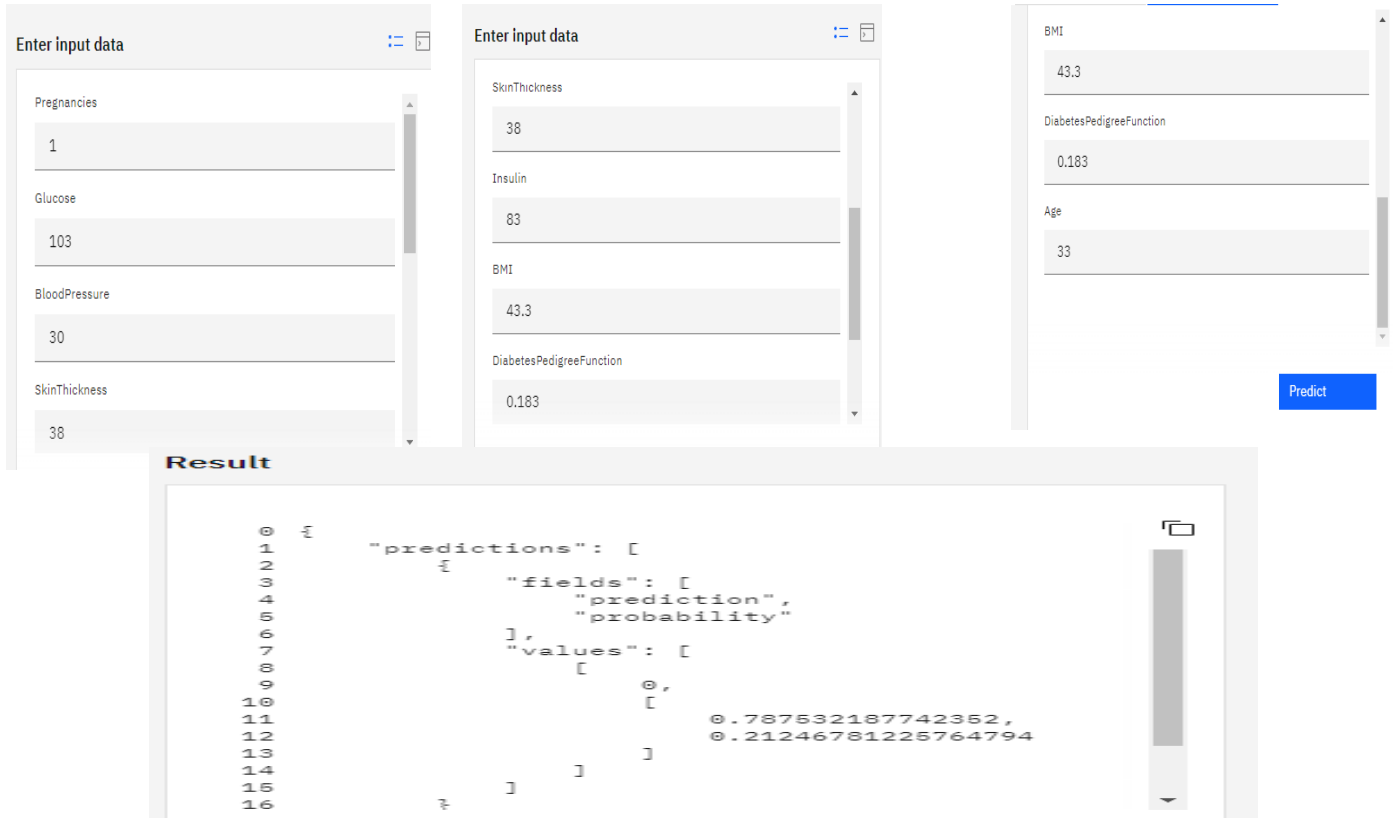

Fig 6: Test-data and its outcome

4.4 NODE CREATION USING NODE-RED

A Cloud Foundry app using node red is created for the Diabetes Prediction model. This is done with the help of Node-Red cloudant service.
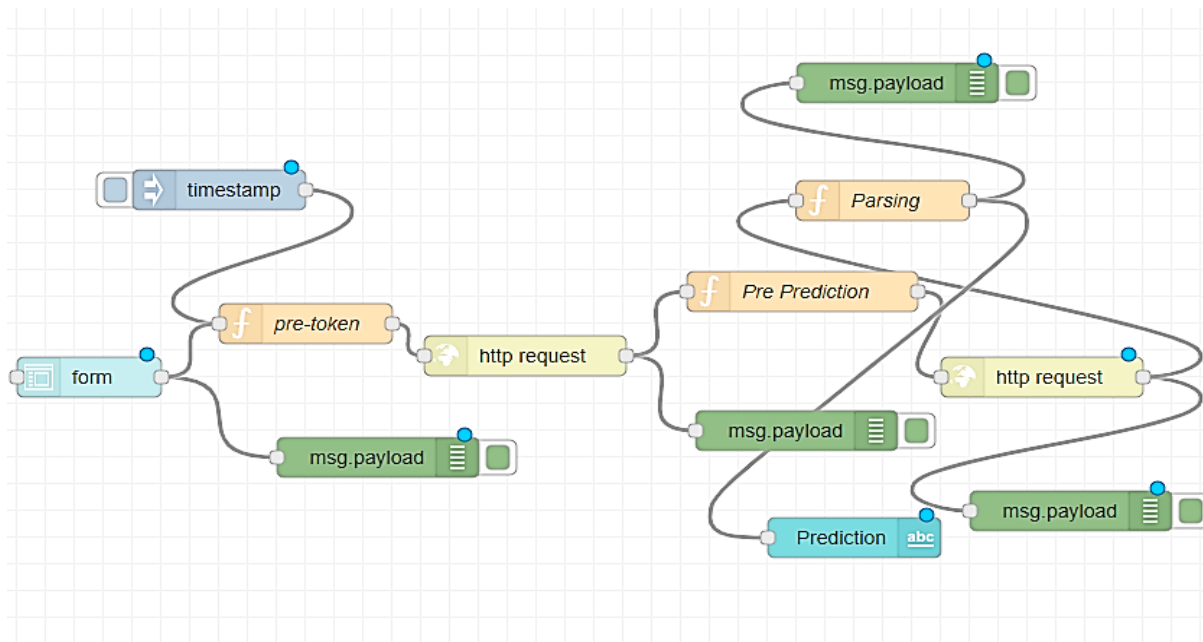

Fig 7: Creation of the nodes network

After the creation of the node network, it is deployed and tested for any errors. The token message and the prediction of the outcome is displayed in the Debug Console as shown below:

Fig 8: Debug console output in node-red

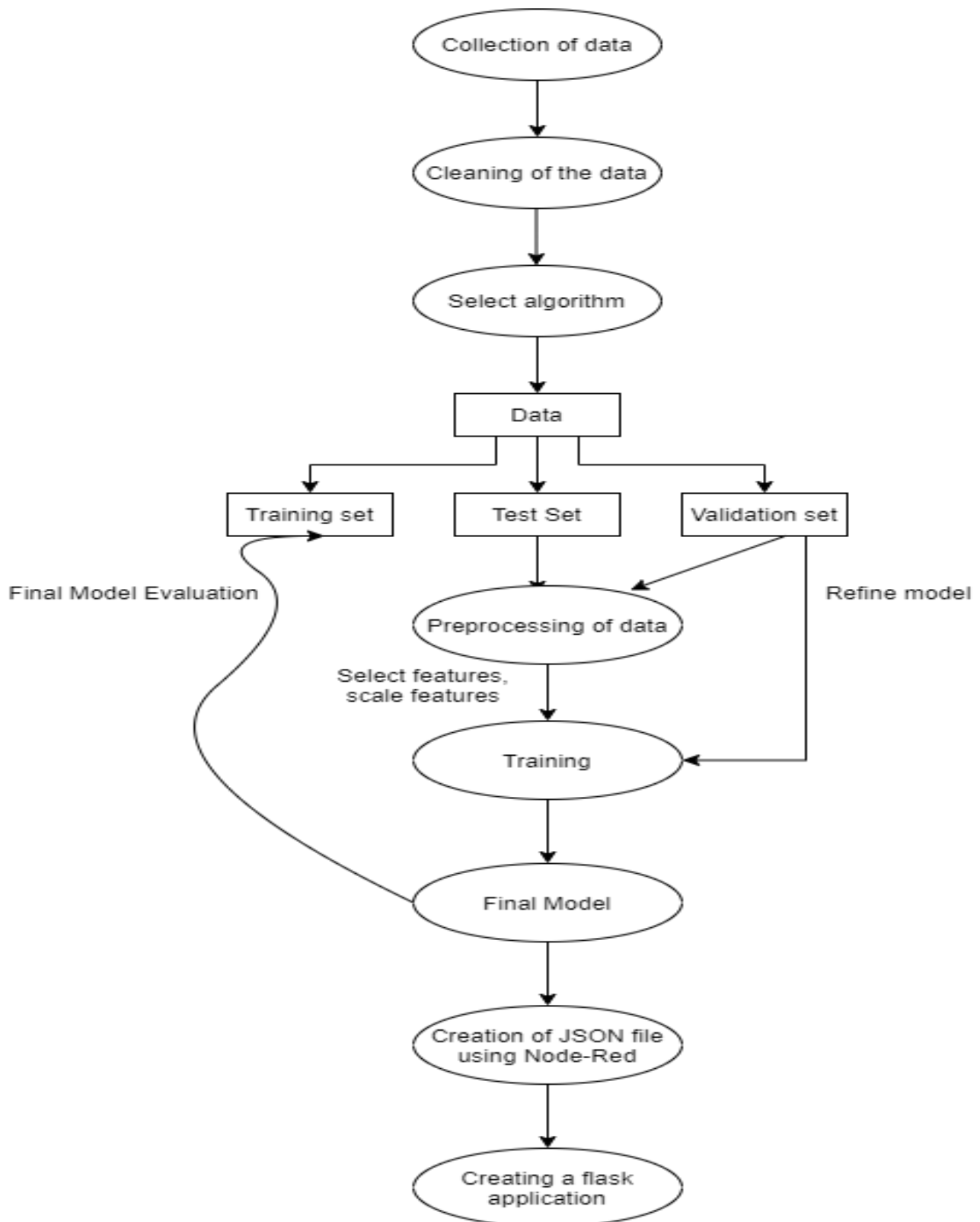Now, the test-data values are tested by launching the output of the created nodal network as shown below:

Fig 9: Testing the values to test the prediction of outcome

CHAPTER 5

FLOWCHART

CHAPTER 6

RESULT

We open a new terminal in VS Code, and execute the python program using the command:

Python file_name.py

This is done as shown below:

Fig 10: Execution of the program in the terminal

Now to open the front-end(flask application, we must press Ctrl+ mouse click on the link obtained after successful compilation of the program, i.e. http://127.0.0.1.5000/

Now, it opens the flask application where we have to give values of various parameters and the output is displayed as Prediction (1 or 0)
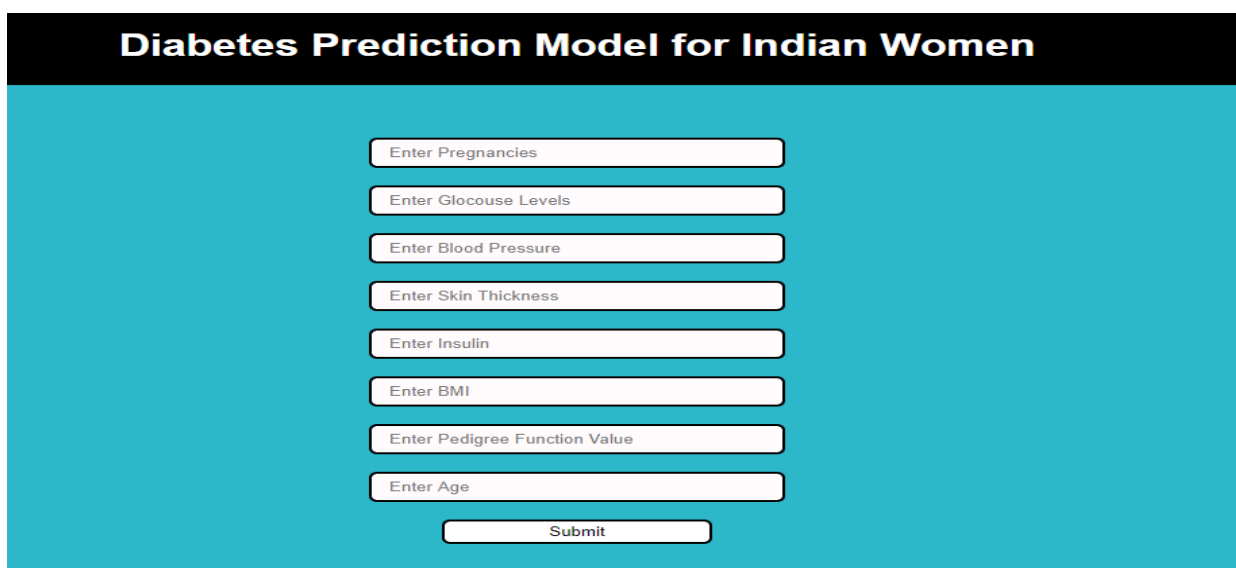
1- Woman is diabetic
2- Woman has no risk of diabetes

Fig 11: Before entering the values

Fig 12: After entering the values

Fig 13: Entering the input values in test model

Result

```
 0  {
 1       "predictions": [
 2           {
 3               "fields": [
 4                   "prediction",
 5                   "probability"
 6               ],
 7               "values": [
 8                   [
 9                       0,
10                       [
11                           0.787532187742352,
12                           0.21246781225764794
13                       ]
14                   ]
15               ]
16           }
```

Fig 14: Test Model output

Diabetics   **Prediction**

Prediction   **1**

Pregnancies *
6

Glucose *
148

BloodPressure *
72

SkinThickness *
35

Insulin *
0

BMI *
33.6

DiabetesPedigreeFunction *
0.627

Age *
50

SUBMIT   CANCEL

Fig 15: Node-Red app output

```python
1    from flask import Flask,render_template,request,url_for
2    import requests
3    import urllib3, json
4    app=Flask(__name__)
5    @app.route('/',methods=['POST','GET'])
6    def hello():
7        if request.method=='POST':
8            preg=request.form['a']
9            glc=request.form['b']
10           bp=request.form['c']
11           skt=request.form['d']
12           ins=request.form['e']
13           bmi=request.form['f']
14           dpf=request.form['g']
15           age=request.form['h']
16           print(age,glc,bp,skt,ins,bmi,dpf,age)
17           try:
18               preg=int(preg)
19               glc=int(glc)
20               bp=int(bp)
21               skt=int(skt)
22               ins=int(ins)
23               bmi=float(bmi)
24               dpf=float(dpf)
25               age=float(age)
26           except:
27               return render_template('data.html',err_msg='Enter Valid Data')
28           url = "https://iam.cloud.ibm.com/identity/token"
29           headers = {"Content-Type": "application/x-www-form-urlencoded"}
30           data = "apikey=" + 'Htz3R7tkRis7wh3Z79LvTHiOU4lysoh4qygEUBZ9gikB' + "&grant_type=urn:ibm:params:oauth:gr
31           IBM_cloud_IAM_uid = "bx"
32           IBM_cloud_IAM_pwd = "bx"
```

Fig 16: dpapp.py code written in VS Code

```html
1    <!DOCTYPE html>
2    <html lang="en">
3    <head>
4        <title>Sample Form</title>
5        <style>
6            .diabetis
7            {
8                width:300px;
9                height:25px;
10               background-color:■#fffbfc;
11               border-style: ridge;
12               border-color:□rgb(0, 0, 0);
13               border-radius:6px;
14           }
15           body
16           {
17               font-family:sans-serif;
18           }
19           #sub
20           {
21               width:200px;
22               height:25px;
23               background-color:■#ffffff;
24               border-style: ridge;
25               border-color:□rgb(0, 0, 0);
26               border-radius:6px;
27           }
28       </style>
29   </head>
30   <body style="background-color:□#000000">
31       <center>
```

Fig 17: main.html code written in VS Code

CHAPTER 7

ADVANTAGES AND DISADVANTAGES

7.1 ADVANTAGES

- Identification without going to Diagnostic centres.

- No need to waste money for tests

- No waiting time and anxiety for reports

- Can sidentify at early stage so that care can be taken.

7.2 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.

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

- Accuracy can be less sometimes

- Some more parameters can be needed sometimes for predicting accurately

CHAPTER 8

APPLICATIONS

The project serves as a prototype to a real-time system which can be created using Machine Learning algorithms. The developed project helps us understand how the developed model works, its behaviour and the output for different test data. Understanding these aspects before in hand will help us develop the real-time system much easily without much difficulty.

Here, we use Binary Classifier algorithm. Contrastingly, we can use K-means or Decision tree approaches as well to get the same predictions output, which can later be implemented in real-time.

CHAPTER 9

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.

CHAPTER 10

FUTURE SCOPE

he 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.

s

CHAPTER 11

BIBLIOGRAPHY

11.1 REFERENCES

- https://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

11.2 APPENDIX

1. 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=" + 'Htz3R7tkRis7wh3Z79LvTHiOU4lysoh4qygEUBZ9gikB' +
"&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","DiabetesPedig
reeFunction","Age"],

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

```
response_scoring = requests.post(

    'https://us-south.ml.cloud.ibm.com/ml/v4/deployments/893accc1-42cd-4542-a819-
2ce902a7a5e3/predictions?version=2020-10-23',

    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('main.html', result=pred)

else:

return render_template('main.html')


if __name__ == '__main__':

    app.run(debug=True)
```

2. main.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(0, 0, 0);

        border-radius:6px;

      }

      body

      {

        font-family:sans-serif;

      }

      #sub

      {

        width:200px;

        height:25px;

        background-color:#ffffff;

        border-style: ridge;

        border-color:rgb(0, 0, 0);

        border-radius:6px;

      }

   </style>

</head>

<body style="background-color:#000000">

   <center>

      <h1 style="color:rgb(255, 255, 255);">Diabetes Prediction Model for Indian
Women</h1>

      <div style="background-color:#2db8ca;">

      <br/><br/>

      {% if result %}

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

      {% endif %}
```

```
{% if err_msg %}

<p style="color : black;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>
```

3. IBM Auto-AI code

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.

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.

Setup:

pip install -U ibm-watson-machine-learning

pip install -U autoai-libs

Auto AI Experiment Metadata:

```
# @hidden_cell

from ibm_watson_machine_learning.helpers import DataConnection, S3Connection,
S3Location

training_data_reference = [DataConnection(

  connection=S3Connection(

    api_key='Hpv6EaXERHSPabKzh1gZqJNRlq1KMR1NekUdvUGqq4LI',

    auth_endpoint='https://iam.bluemix.net/oidc/token/',

    endpoint_url='https://s3-api.us-geo.objectstorage.softlayer.net'

  ),

    location=S3Location(

    bucket='diabetespredictionsystembasedonli-donotdelete-pr-2sazjjwazadya9',

    path='diabetes.csv'
```

```
))

]

training_result_reference = DataConnection(

    connection=S3Connection(

        api_key='Hpv6EaXERHSPabKzh1gZqJNRlq1KMR1NekUdvUGqq4LI',

        auth_endpoint='https://iam.bluemix.net/oidc/token/',

        endpoint_url='https://s3-api.us-geo.objectstorage.softlayer.net'

    ),

    location=S3Location(

        bucket='diabetespredictionsystembasedonli-donotdelete-pr-2sazjjwazadya9',

        path='auto_ml/4acd09e0-6742-4df5-beac-c9bee8515568/wml_data/aae043c5-f18e-4947-aee5-80424f4d4bde/data/automl',

        model_location='auto_ml/4acd09e0-6742-4df5-beac-c9bee8515568/wml_data/aae043c5-f18e-4947-aee5-80424f4d4bde/data/automl/hpo_c_output/Pipeline1/model.pickle',

        training_status='auto_ml/4acd09e0-6742-4df5-beac-c9bee8515568/wml_data/aae043c5-f18e-4947-aee5-80424f4d4bde/training-status.json'

    ))
```

Extract needed parameter values to run Auto AI experiment

```
    experiment_metadata = dict(

    prediction_type='classification',

    prediction_column='Outcome',

    test_size=0.1,

    scoring='accuracy',

    project_id='42edfe38-d6ae-4877-a41c-b2be2e10f5d8',

    csv_separator=',',

    random_state=33,
```

max_number_of_estimators=2,

training_data_reference = training_data_reference,

training_result_reference = training_result_reference,

deployment_url='https://us-south.ml.cloud.ibm.com')

pipeline_name='Pipeline_4'

Create Deployment:

target_space_id = "PUT_YOUR_TARGET_SPACE_ID_HERE"

from ibm_watson_machine_learning.deployment import WebService

service = WebService(source_wml_credentials=wml_credentials,

　　　　　target_wml_credentials=wml_credentials,

　　　　　source_project_id=experiment_metadata['project_id'],

　　　　　target_space_id=target_space_id)

service.create(

model=pipeline_name,

metadata=experiment_metadata,

deployment_name=f'{pipeline_name}_webservice'

)