Diabetes Mellitus Prediction Using IBM Auto AI Service

Category: IBM

Cloud Application

Skills Required:
IBM Nodered,IBM Watson Studio,IBM Machine Learning

1. INTRODUCTION

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 centres.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 h applications in healthcare domain, very effectively used in prediction of diseases.Various life style and other parameters have an impa 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 diabete.

Diabetes mellitus 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 netw 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.

1.1. Overview

This Diabetes prediction application is built by using IBM Watson Studio associated with Machine Learning service to perform the prediction task and Node-Red is used to build the web application.

1.2. 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.

2 LITERATURE SURVEY

2.1 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.

2.2 Proposed solution

The proposed system uses 8 vital parameters that can predict the chances of acquiring the diabetes which uses machine learning techniques to learn from hundreds of patient data, fit the best performing model consistent with the training examples given and it is tested/validated with unseen data also. It is found to perform with about good accuracy in its predictions. So, an application built on this will be highly reliable with good prediction accuracy.

3 . THEORITICAL ANALYSIS

Diabetes mellitus is a chronic disease characterized by hyperglycemia. It may cause many complications. According to the growing morbidity in recent years, in 2040, the world's diabetic patients will reach 642 million, which means that one of the ten adults in the future is suffering from diabetes.

In this project, you need to build a machine learning model that can efficiently discover the rules to predict diabetes
mellitus of patients based on the given parameter about their health. The model needs to be deployed in the IBM cloud to get scoring endpoint which can be used as API in web app building. . The model prediction needs to be showcased on User Interface.

Services Used:

1. IBM Watson Studio
2. IBM Watson Machine Learning
3. Node-RED
4. IBM Cloud Object Storage

4. EXPERIMENTAL INVESTIGATIONS

3.1 Block diagram

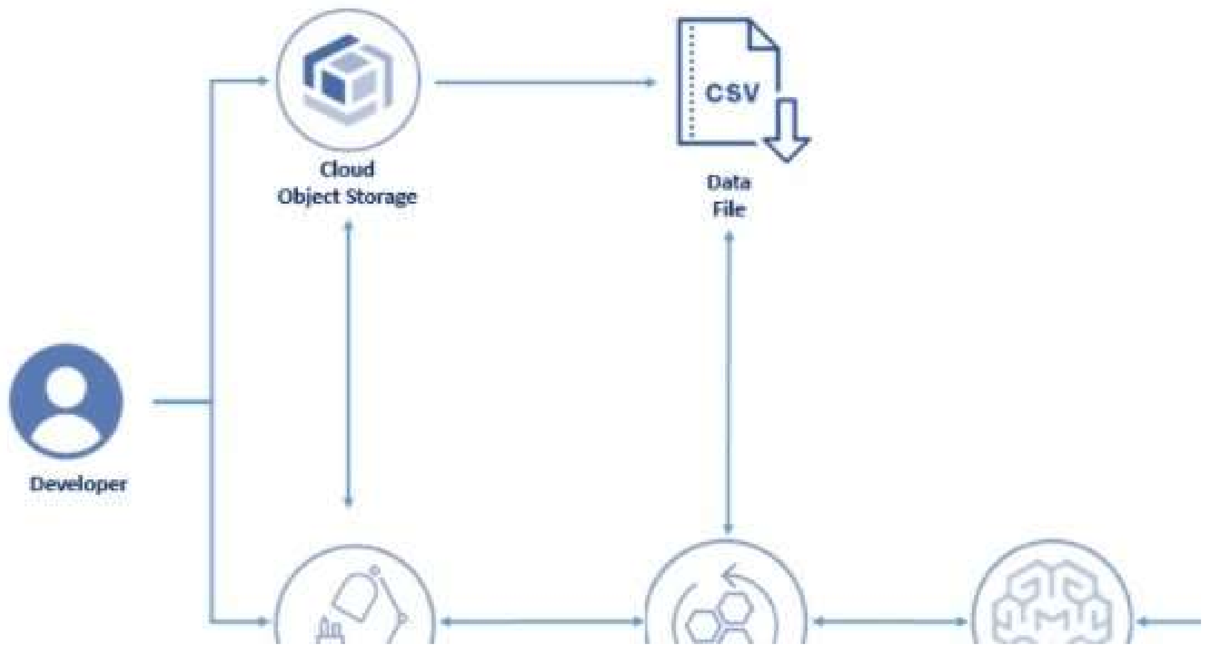


fig3.1: Proposed Architecture for diabetes mellitus prediction model

3.2 Hardware/ Software designing:

a. IBM Watson Studio

b. AI/Machine Learning Service

c. IBM Cloud Object Storage

d. Node Red flow-based development tool

e.

5. Dataset collection :

Pima Indians Diabetes dataset was collected from Kaggle.
All data is from females atleast 21 years old of Pima Indian heritage.
Dataset description
a. 1. No. of Pregnancies - Number of times pregnant
2. Random Glucose - Plasma glucose concentration a 2 hours in an oral glucose tolerance test
3. Blood Pressure - Diastolic blood pressure (mm Hg)
4. Skin Thickness - Triceps skin fold thickness (mm)
5. Insulin - 2-Hour serum insulin (mu U/ml)
6. Body Mass Index - Body mass index (weight in kg/(height in m)^2)

7. Diabetes Pedigree Function - A function which scores the likelihood of diabetes based on family history. It provided some data on diabetes mellitus history in relatives and the genetic relationship of those relatives to the patient
8. Age - Age (in years)
9. Outcome - The outcome label 1 for Yes (for chances of acquiring diabetes and 0 for No (for no chances of acquiring diabetes)

6. FLOW CHART

6. Steps followed to build the project
1) Create a project in Watson Studio – Diabetes Prediction
2) Add Auto AI experiment
3) Create a Machine Learning instance
4) Associate ML instance to the project
5) Load the dataset to cloud object storage
6) Select the target variable (prediction parameter) in the dataset
7) Train the model
8) Deploy
9) Build web application using Node-Red

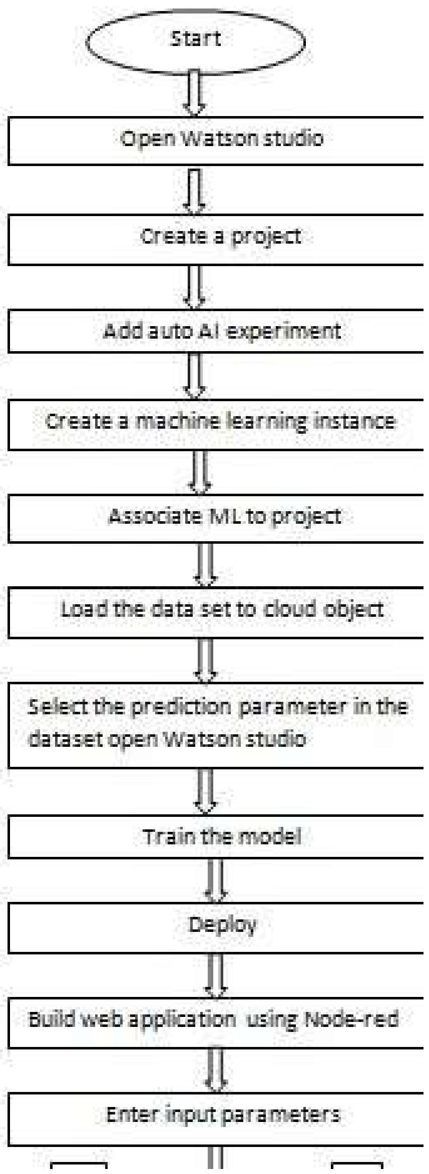
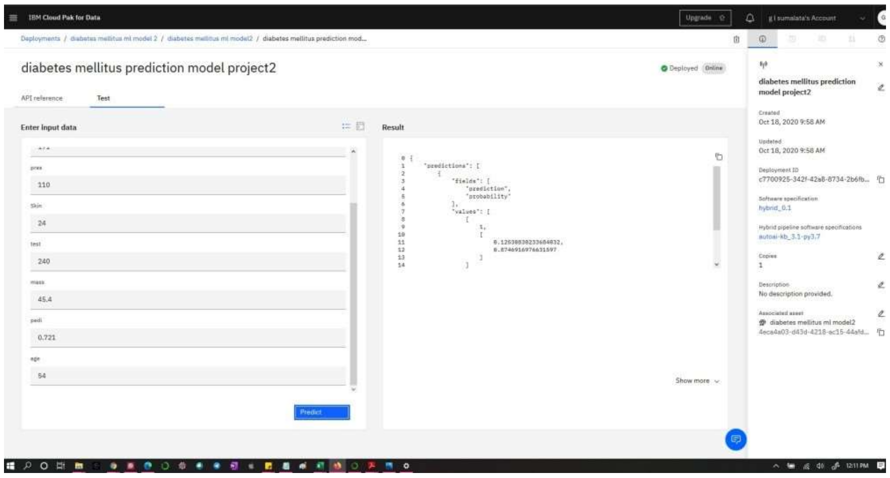

Start
Open Watson studio
Create a project
Add auto AI experiment
Create a machine learning instance
Associate ML to project
Load the data set to cloud object
Select the prediction parameter in the dataset open Watson studio
Train the model
Deploy
Build web application using Node-red
Enter input parameters

Fig 6.1 :Flowchart for diabetes mellitus prediction process using IBM Auto AI

7.Auto AI Experiment Results

Grading Boosting Classifier is selected by the Auto AI experiment as the best performing model after fine tuning all the hyper-parameters. It is found to give about 77.1% accuracy with 90% training set size and 10% test set size. The Area Under the Curve (AUC) is also satisfactory which depicts the TPR (sensitivity) and FPR (specificity). The models having higher AUC are said to perform better.


IBM Cloud Pak for Data
Upgrade
g l sumalata's Account
Deployments / diabetes mellitus ml model 2 / diabetes mellitus ml model2 / diabetes mellitus prediction mod...
API reference
Test
Deployed
Online
Predict
Show more

diabetes mellitus prediction model project2

Enter input data

pres	110
Skin	24
test	240
mass	45.4
pedi	0.721
age	54

Result

```
{
    "predictions": [
        {
            "fields": [
                "prediction",
                "probability"
            ],
            "values": [
                [
                    1,
                    [
                        0.12638838233684832,
                        0.8746916976631597
                    ]
                ]
```

diabetes mellitus prediction model project2

Created
Oct 18, 2020 9:58 AM

Updated
Oct 18, 2020 9:58 AM

Deployment ID
c7700925-342f-42a8-8734-2b6fb...

Software specification
hybrid_0.1

Hybrid pipeline software specifications
autoai-kb_3.1-py3.7

Copies
1

Description
No description provided.

Associated asset
diabetes mellitus ml model2
4eca4a03-d43d-4218-ac15-44afd...

Fig7.1:Deployement and test

8. Node Red Flow

Node-RED is a flow-based development tool for visual programming developed originally by IBM for wiring together hardware devices, APIs and online services. The flows created in Node-RED are stored using JSON.

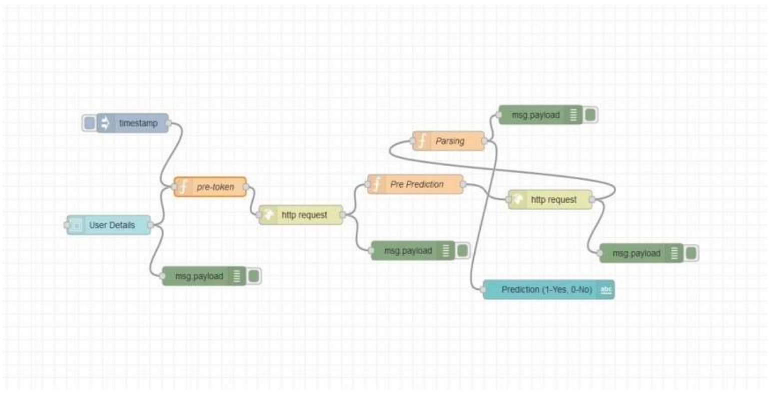
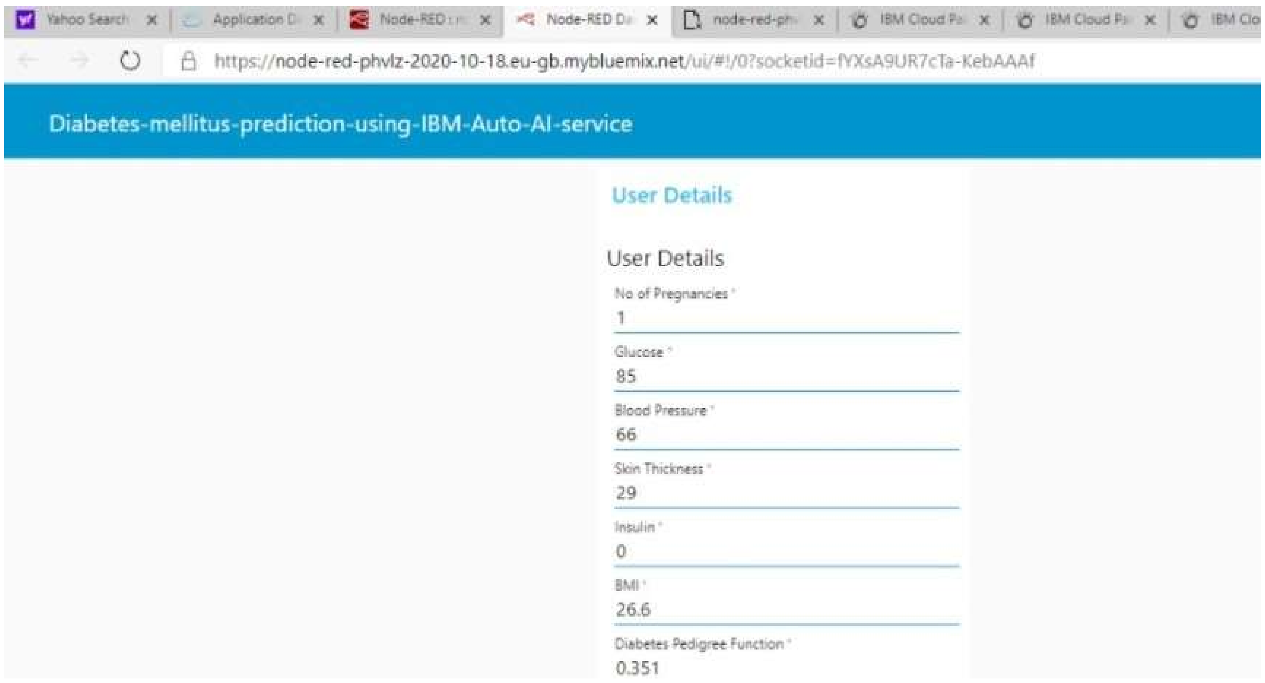


Fig. 8.1 Node-Red Flow

9. Demonstration of the application with the screenshots This application predicts the chance of acquiring diabetes based on the features/information the user enters through the user interface

i. Home page of the application


Yahoo Search | Application D | Node-RED : n | Node-RED Da | node-red-ph | IBM Cloud Pa | IBM Cloud Pa | IBM Clo

https://node-red-phvlz-2020-10-18.eu-gb.mybluemix.net/ui/#!/0?socketid=fYXsA9UR7cTa-KebAAAf

Diabetes-mellitus-prediction-using-IBM-Auto-AI-service

User Details

User Details

No of Pregnancies *
1

Glucose *
85

Blood Pressure *
66

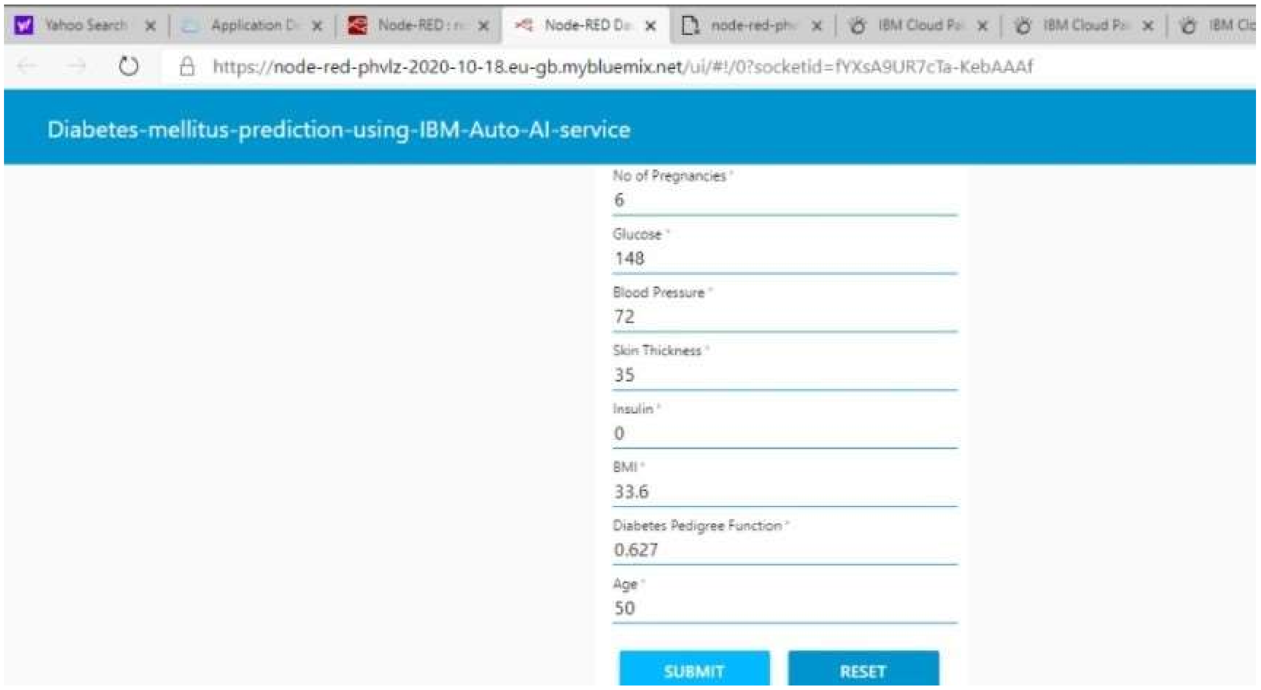
Skin Thickness *
29

Insulin *
0

BMI *
26.6

Diabetes Pedigree Function *
0.351

Fig. 9.1 Home page


Yahoo Search | Application D | Node-RED : n | Node-RED Da | node-red-ph | IBM Cloud Pa | IBM Cloud Pa | IBM Clo

https://node-red-phvlz-2020-10-18.eu-gb.mybluemix.net/ui/#!/0?socketid=fYXsA9UR7cTa-KebAAAf

Diabetes-mellitus-prediction-using-IBM-Auto-AI-service

No of Pregnancies *
6

Glucose *
148

Blood Pressure *
72

Skin Thickness *
35

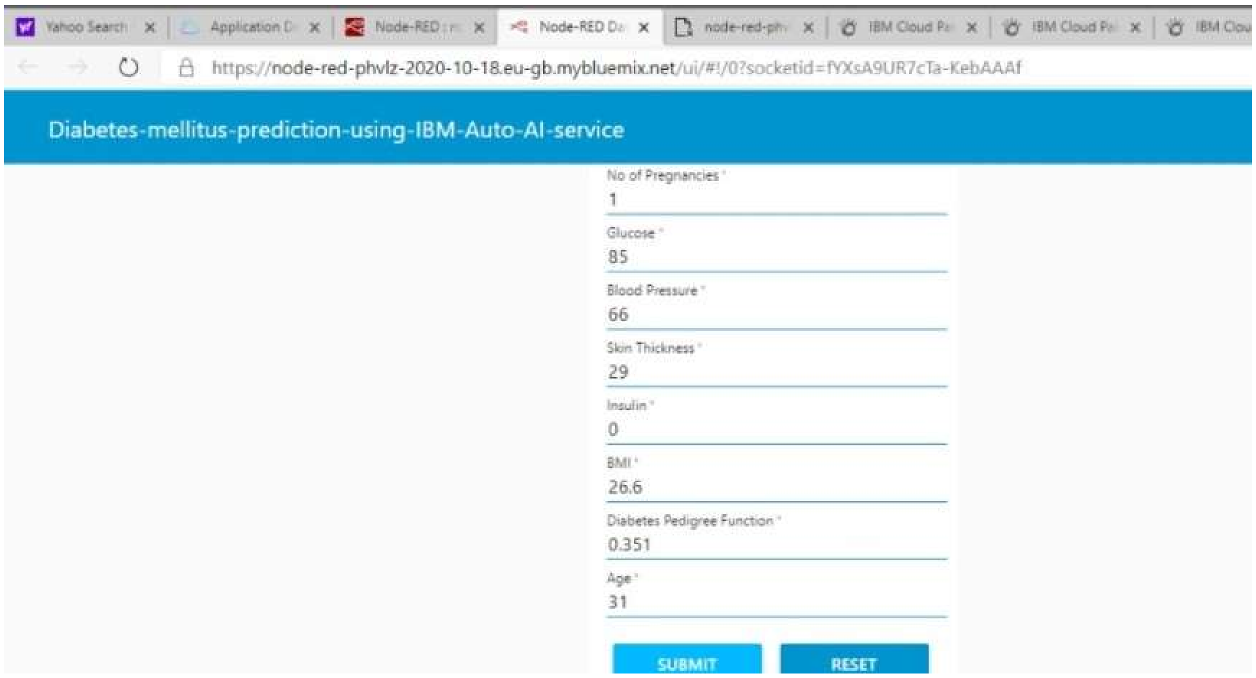
Insulin *
0

BMI *
33.6

Diabetes Pedigree Function *
0.627

Age *
50

ii. Reset button can be pressed to clear the previous input and enter fresh details



https://node-red-phvlz-2020-10-18.eu-gb.mybluemix.net/ui/#!/0?socketid=fYXsA9UR7cTa-KebAAAf

Diabetes-mellitus-prediction-using-IBM-Auto-AI-service

No of Pregnancies *
1

Glucose *
85

Blood Pressure *
66

Skin Thickness *
29

Insulin *
0

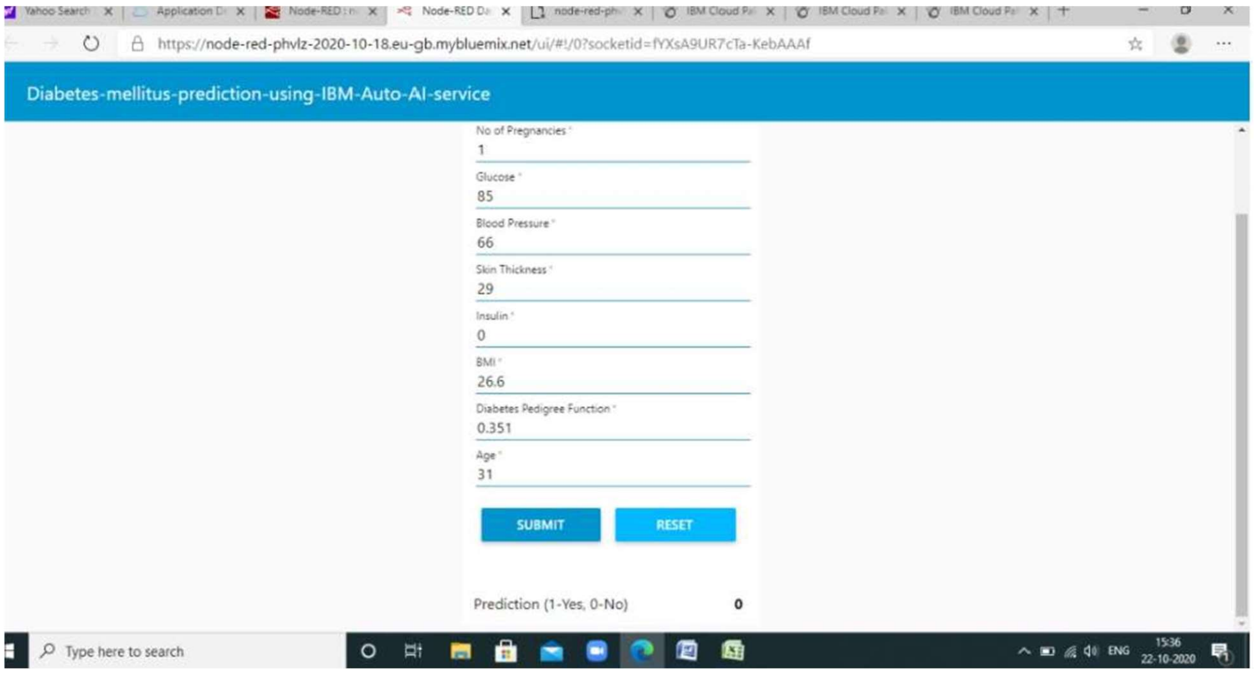
BMI *
26.6

Diabetes Pedigree Function *
0.351

Age *
31

Fig. 9.2 The data can be Reset for entering new data

iii. New user details entry and submit



https://node-red-phvlz-2020-10-18.eu-gb.mybluemix.net/ui/#!/0?socketid=fYXsA9UR7cTa-KebAAAf

Diabetes-mellitus-prediction-using-IBM-Auto-AI-service

No of Pregnancies *
1

Glucose *
85

Blood Pressure *
66

Skin Thickness *
29

Insulin *
0

BMI *
26.6

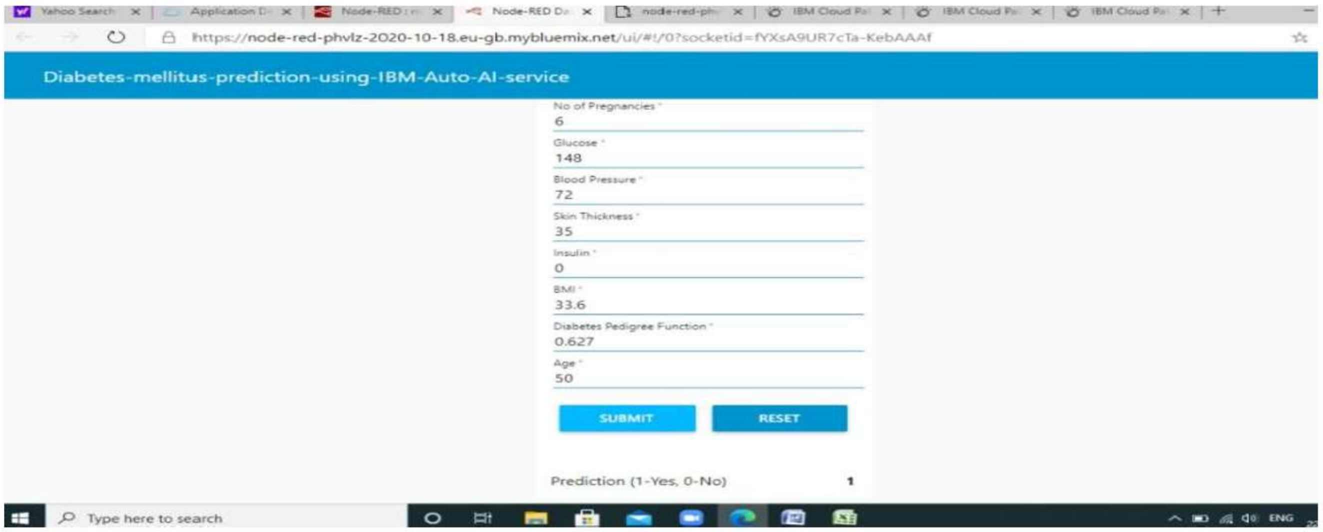
Diabetes Pedigree Function *
0.351

Age *
31

Prediction (1-Yes, 0-No) 0

Fig. 9.3 Node-RED App Output for 0

iv. Prediction display


https://node-red-phvlz-2020-10-18.eu-gb.mybluemix.net/ui/#!/0?socketid=fYXsA9UR7cTa-KebAAAf

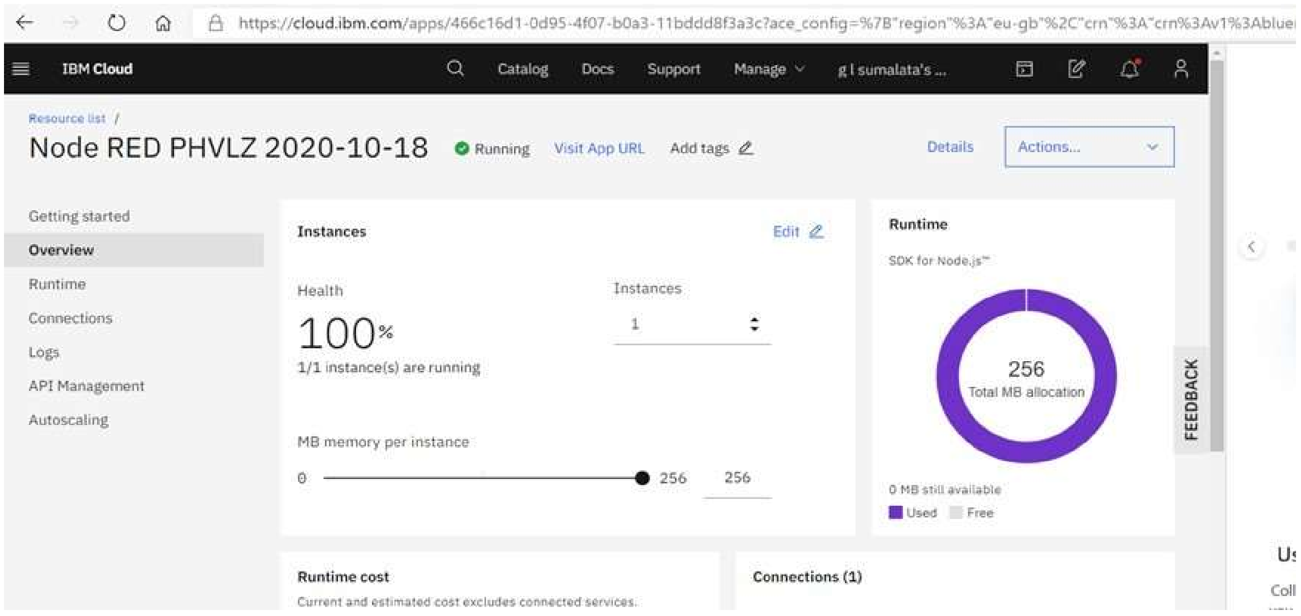
Diabetes-mellitus-prediction-using-IBM-Auto-AI-service

No of Pregnancies *
6
Glucose *
148
Blood Pressure *
72
Skin Thickness *
35
Insulin *
0
BMI *
33.6
Diabetes Pedigree Function *
0.627
Age *
50

SUBMIT RESET

Prediction (1-Yes, 0-No) 1

Fig. 9.4 Node-RED App Output for 1


https://cloud.ibm.com/apps/466c16d1-0d95-4f07-b0a3-11bddd8f3a3c?ace_config=%7B"region"%3A"eu-gb"%2C"crn"%3A"crn%3Av1%3Ablue

Node RED PHVLZ 2020-10-18 Running Visit App URL Add tags

Instances

Health 100% 1/1 instance(s) are running

Instances 1

MB memory per instance 0 256 256

Runtime

SDK for Node.js™

256 Total MB allocation

0 MB still available

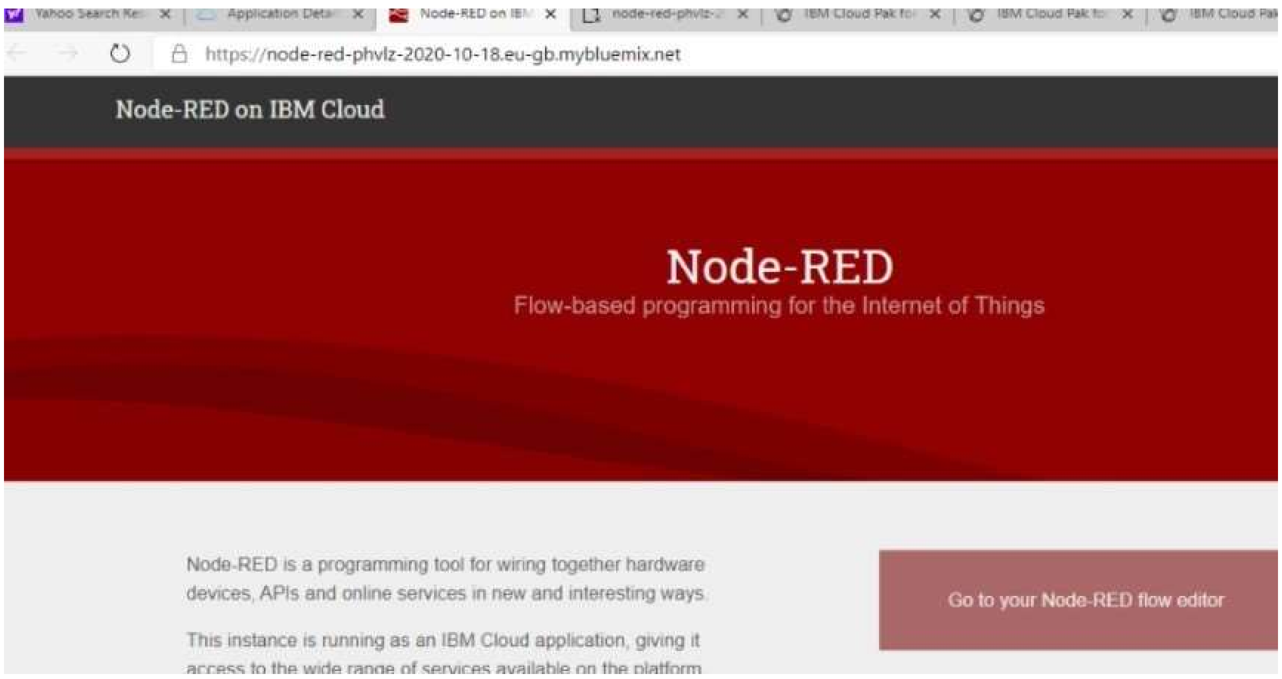
Used Free

Runtime cost

Current and estimated cost excludes connected services.

Connections (1)

Fig. 9.5 Node-RED App


https://node-red-phvlz-2020-10-18.eu-gb.mybluemix.net

Node-RED on IBM Cloud

Node-RED

Flow-based programming for the Internet of Things

Node-RED is a programming tool for wiring together hardware devices, APIs and online services in new and interesting ways.

This instance is running as an IBM Cloud application, giving it access to the wide range of services available on the platform

Go to your Node-RED flow editor

Fig: 9.6 Node-Red on IBM cloud

Node-Red is a programming tool for wiring the hardware devices.It also connects APIs and online services.It is used for building user interface for users to enter data so that the status of diabetic canbe predicted.

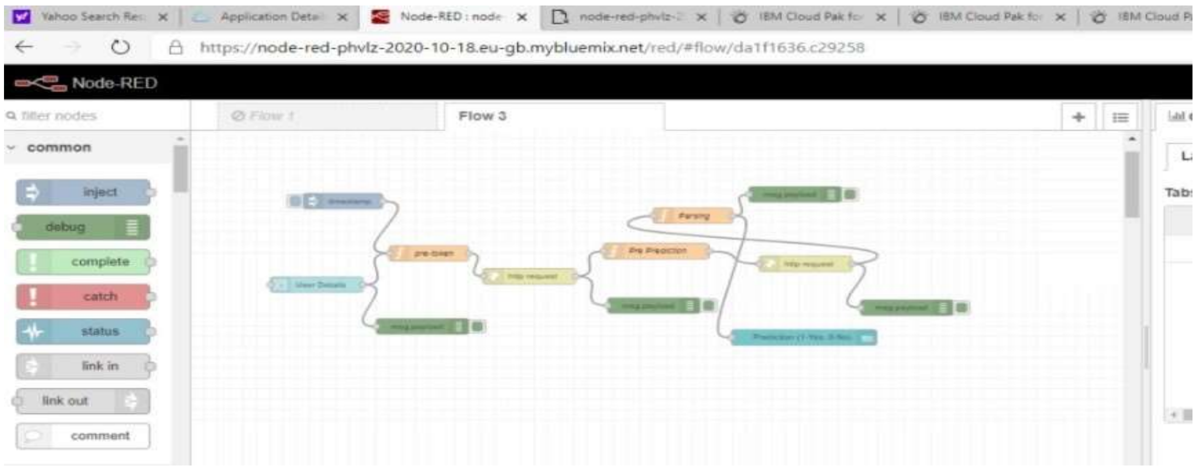


Fig. 9.7 Node-RED App Flow

7 ADVANTAGES & DISADVANTAGES

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

DISADVANTAGES:

1. Prediction can be wrong sometimes according to confusion matrix which may create worst consequencies
2. Accuracy can be less sometimes
3. Some more parameters can be needed sometimes for predicting accurately

8 APPLICATIONS

1. Diabetes Mellitus prediction for kidney patients
2. Diabetes Mellitus prediction for heart diseased patients

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.

10 FUTURE SCOPE `

The Accuracy obtained is 77%.It can be increased by following different methodseither 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.

ACKNOWLEDGEMENT:

I thank all the faculty members, IBM Team ,Gurucool Program and mentors who provided the excellent training and support for the knowledge shared and for completing the project. .

11 BIBILOGRAPH

Y

APPE

NDIX

A. Source code

IBM AutoAI-SDK Auto-Generated Notebook v1.14.1

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

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.

Notebook content

This notebook contains steps and code to demonstrate AutoAI pipeline. This notebook introduces commands for getting data,
pipeline model, model inspection and testing.

Some familiarity with Python is helpful. This notebook uses Python 3.

Notebook goals

- inspection of trained pipeline via graphical vizualization and source code preview
- pipeline evaluation
- pipeline deployment and webservice scoring.

Contents

This notebook contains the following parts:

Setup

Before you use the sample code in this notebook, you must perform the following setup tasks:

- `ibm_watson_machine_learning` installation
- `autoai-libs` installation/upgrade
- `lightgbm` or `xgboost` installation/downgrade if they are needed.

In [ ]:

```
!pip install -U ibm-watson-machine-learning
```

In [ ]:

```
!pip install -U autoai-libs
```

AutoAI experiment metadata

This cell defines COS credentials required to retrieve AutoAI pipeline.

In [ ]:

```
# @hidden_cell
```

```
from ibm_watson_machine_learning.helpers import DataConnection, S3Connection, S3Locatio
n

training_data_reference = [DataConnection(
    connection=S3Connection(
        api_key='ZtDre_BgetFAfRkC2GN8jA66ayld4p0wgWJ6cmGkZoYK',
        auth_endpoint='https://iam.bluemix.net/oidc/token/',
        endpoint_url='https://s3-api.us-geo.objectstorage.softlayer.net'
    ),
        location=S3Location(
        bucket='diabetesmellituspredictionusingib-donotdelete-pr-8ifczp1e9igaiw',
        path='pima-indians-diabetes.data.csv'
    ))
]
training_result_reference = DataConnection(
    connection=S3Connection(
        api_key='ZtDre_BgetFAfRkC2GN8jA66ayld4p0wgWJ6cmGkZoYK',
        auth_endpoint='https://iam.bluemix.net/oidc/token/',
        endpoint_url='https://s3-api.us-geo.objectstorage.softlayer.net'
    ),
    location=S3Location(
        bucket='diabetesmellituspredictionusingib-donotdelete-pr-8ifczp1e9igaiw',
        path='auto_ml/a70aa112-5cb7-494b-8b6e-151da38ac9b2/wml_data/4f55c607-5012-4b35-
b854-00150a5befe2/data/automl',
        model_location='auto_ml/a70aa112-5cb7-494b-8b6e-151da38ac9b2/wml_data/4f55c607-
5012-4b35-b854-00150a5befe2/data/automl/hpo_c_output/Pipeline1/model.pickle',
        training_status='auto_ml/a70aa112-5cb7-494b-8b6e-151da38ac9b2/wml_data/4f55c607
-5012-4b35-b854-00150a5befe2/training-status.json'
    ))
```

Following cell contains input parameters provided to run the AutoAI experiment in Watson Studio

In [ ]:

```
experiment_metadata = dict(
   prediction_type='classification',
   prediction_column='class',
   test_size=0.1,
   scoring='accuracy',
   project_id='dc710200-9a27-4ecd-88ac-d027fc7b4875',
   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'
```

Pipeline inspection

In this section you will get the trained pipeline model from the AutoAI experiment and inspect it.
You will see pipeline as a pythone code, graphically visualized and at the end, you will perform a local test.

Get historical optimizer instance

The next cell contains code for retrieving fitted optimizer.

In [ ]:

```
from ibm_watson_machine_learning.experiment import AutoAI

optimizer = AutoAI().runs.get_optimizer(metadata=experiment_metadata)
```

Get pipeline model

The following cell loads selected AutoAI pipeline model. If you want to get pure scikit-learn pipeline specify `as_type='sklearn'` parameter. By default enriched scikit-learn pipeline is returned `as_type='lale'`.

In [ ]:

```
pipeline_model = optimizer.get_pipeline(pipeline_name=pipeline_name)
```

Preview pipeline model as python code

In the next cell, downloaded pipeline model could be previewed as a python code.
You will be able to see what exact steps are involved in model creation.

In [ ]:

```
pipeline_model.pretty_print(combinators=False, ipython_display=True)
```

Visualize pipeline model

Preview pipeline model stages as graph. Each node's name links to detailed description of the stage.

In [ ]:

```
pipeline_model.visualize()
```

Read training data

Retrieve training dataset from AutoAI experiment as pandas DataFrame.

In [ ]:

```
train_df = optimizer.get_data_connections()[0].read()
test_df = train_df.sample(n=5).drop([experiment_metadata['prediction_column']], axis=1)
```

Test pipeline model locally

You can predict target value using trained AutoAI model by calling `predict()`.

In [ ]:

```
y_pred = pipeline_model.predict(test_df.values)
print(y_pred)
```

Pipeline refinery and testing (optional)

In this section you will learn how to refine and retrain the best pipeline returned by AutoAI. It can be performed by:

- modifying pipeline definition source code
- using lale library for semi-automated data science

Note: In order to run this section change following cells to 'code' cell.

Pipeline definition source code

Following cell lets you experiment with pipeline definition in python, e.g. change steps parameters.

It will inject pipeline definition to the next cell.
pipeline_model.pretty_print(combinators=False, ipython_display='input')

Lale library

Note: This is only an exemplary usage of lale package. You can import more different estimators to refine downloaded pipeline model.

Import estimators

from sklearn.linear_model import LogisticRegression as E1 from sklearn.tree import DecisionTreeClassifier as E2 from sklearn.neighbors import KNeighborsClassifier as E3 from lale.lib.lale import Hyperopt from lale.operators import TrainedPipeline from lale import wrap_imported_operators from lale.helpers import import_from_sklearn_pipeline wrap_imported_operators()

Pipeline decomposition and new definition

In this step the last stage from pipeline is removed.
prefix = pipeline_model.remove_last().freeze_trainable() prefix.visualize()new_pipeline = prefix >> (E1 | E2 | E3) new_pipeline.visualize()

New optimizer `hyperopt` configuration and training

This section can introduce other results than the original one and it should be used by more advanced users.

New pipeline is re-trained by passing train data to it and calling `fit` method.

Following cell performs dataset split for refined pipeline model.
from sklearn.model_selection import train_test_split train_X = train_df.drop([experiment_metadata['prediction_column']], axis=1).values train_y = train_df[experiment_metadata['prediction_column']].values train_X, test_X, train_y, test_y = train_test_split(train_X, train_y, test_size=experiment_metadata['test_size'], stratify=train_y, random_state=experiment_metadata['random_state'])hyperopt = Hyperopt(estimator=new_pipeline, cv=3, max_evals=20) fitted_hyperopt = hyperopt.fit(train_X, train_y)hyperopt_pipeline = fitted_hyperopt.get_pipeline() new_pipeline = hyperopt_pipeline.export_to_sklearn_pipeline()prediction = new_pipeline.predict(test_X)from sklearn.metrics import accuracy_score score = accuracy_score(y_true=test_y, y_pred=prediction) print('accuracy_score: ', score)

Deploy and Score

In this section you will learn how to deploy and score pipeline model as webservice using WML instance.

Connection to WML

Authenticate the Watson Machine Learning service on IBM Cloud.

Tip: Your Cloud API key can be generated by going to the **Users** section of the Cloud console. From that page, click your name, scroll down to the **API Keys** section, and click **Create an IBM Cloud API key**. Give your key a name and click **Create**, then copy the created key and paste it below.

Note: You can also get service specific apikey by going to the **Service IDs** section of the Cloud Console. From that page, click **Create**, then copy the created key and paste it below.

Action: Enter your `api_key` in the following cell.

In [ ]:

```
api_key = "PUT_YOUR_API_KEY_HERE"

wml_credentials = {
  "apikey": api_key,
  "url": experiment_metadata["deployment_url"]
}
```

Create deployment

Action: If you want to deploy refined pipeline please change the `pipeline_name` to `new_pipeline`. If you prefer you can also change the `deployment_name`.

Action: To perform deployment please specify `target_space_id`.

In [ ]:

```
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'
)
```

Deployment object could be printed to show basic information:

In [ ]:

```
print(service)
```

To be able to show all available information about deployment use `.get_params()` method:

In [ ]:

```
service.get_params()
```

Score webservice

You can make scoring request by calling `score()` on deployed pipeline.

In [ ]:

```
predictions = service.score(payload=test_df)
predictions
```

If you want to work with the webservice in external Python application you can retrieve the service object by:

- initialize service by:
    - `service = WebService(target_wml_credentials=wml_credentials,`

      `target_space_id=target_space_id)`

- get deployment_id by `service.list()` method
- get webservice object by `service.get('deployment_id')` method

After that you can call `service.score()` method.

Delete deployment

You can delete an existing deployment by calling `service.delete()`.

Authors