Data Glacier

Your Deep Learning Partner

Model deployment (Cloud and API)

Virtual Internship

Bilikis O. Alayo

LISUM31

April 5, 2024

https://github.com/Omolara-Alayo/Data-Science-Internship/tree/main/Week%205/ML%20model%20deployment

The dataset was scaled and divided into training and test sets at a ratio of 70:30.

Dataset

- A simple heart disease dataset consisting of 304 rows and 13 features was used for this study. The data was cleaned and preprocessed for model training.
- The dataset was scaled and split into training set and test set in the ratio of 70:30.

In [2]:
```
# Read the heart disease data into a DataFrame
heart_disease_data  = pd.read_csv('heart.csv')
```

In [3]:
```
# Display the fisrt 5 rows of the dataset
heart_disease_data.head()
```

Out[3]:

	age	sex	cp	trestbps	chol	fbs	restecg	thalach	exang	oldpeak	slope	ca	thal	target
0	63	1	3	145	233	1	0	150	0	2.3	0	0	1	1
1	37	1	2	130	250	0	1	187	0	3.5	0	0	2	1
2	41	0	1	130	204	0	0	172	0	1.4	2	0	2	1
3	56	1	1	120	236	0	1	178	0	0.8	2	0	2	1
4	57	0	0	120	354	0	1	163	1	0.6	2	0	2	1

```
# Checking the shape of the dataset

print(("rows, columns"),heart_disease_data.shape)
```

Model training

- A logistic regression classifier was used for the training and the model was validated and persisted for later use.

Training the model Using Logistic Regression

In [11]:
```
import time
import pickle
from sklearn.linear_model import LogisticRegression
from sklearn.model_selection import cross_val_score
from sklearn.metrics import confusion_matrix, classification_report
from sklearn import metrics
```

In [13]:
```
# Initializing and training the model

start_time = time.time()   #Initializing the start time to capture the execution time
logreg =  LogisticRegression()
logreg_model= logreg.fit(X_train, y_train)

logreg_time = time.time() - start_time
print('Accuracy of Logistics Regression classifier on training set: {:.2f}'.format(logreg_model.score(X_train, y_train)))
print('Accuracy of Logistics Regression classifier on test set: {:.2f}'.format(logreg_model.score(X_test, y_test)))
print('Logistic Regression classifier Training Time (seconds):',logreg_time)
```

```
Accuracy of Logistics Regression classifier on training set: 0.86
Accuracy of Logistics Regression classifier on test set: 0.79
Logistic Regression classifier Training Time (seconds): 0.006406545639038086
```

In [14]:
```
#predicting heart dissease using with the logistics regression model

heart_disease_prediction = logreg_model.predict(X_test)
heart_disease_prediction[:10]
```

Out[14]:
```
array([1, 1, 1, 0, 1, 1, 0, 1, 1, 1], dtype=int64)
```

In [15]:
```
# Evaluating the Logistics Regression Classifier using cross validation setting CV = 5.

logreg_accuracy = cross_val_score(logreg_model, X_train, y_train, cv=5)
print("Logistic Regression Accuracy:",round(logreg_accuracy.mean()*100,2),"%")
```

```
Logistic Regression Accuracy: 84.41 %
```

[21]:
```
# Persisting the Heart disease predictive model
import pickle

model_dump = pickle.dumps(logreg_model)
with open('heart_disease_model.pkl', 'wb') as model_file:
    model_file.write(model_dump)
```

[23]:
```
# Predicting heart_disease from the model

#Loading the Logistic Regression model
with open('heart_disease_model.pkl', 'rb') as model_file:
    loaded_model = pickle.load(model_file)

heart_disease_predict = loaded_model.predict(X_test)
print("Heart disease Predictions using the loaded Logistic Regression model:")
print(heart_disease_predict[:10])
```

```
Heart disease Predictions using the loaded Logistic Regression model:
[1 1 1 0 1 1 0 1 1 1]
```

Flask App

After training the model, a flask app was built for a web application.

app.py

C: > Users > berly > OneDrive > Desktop > Data Glacier_intern > Week 5 > New folder > app.py > ...

```python
#!/usr/bin/env python
# coding: utf-8

# Flask app for heart predictive model

Run Cell | Run Below | Debug Cell
# In[ ]:


import numpy as np
from flask import Flask, request, render_template
import pickle

app = Flask(__name__)

#Prediction function
def HeartDiseasePredictor(predict_list):
    to_predict  = np.array(predict_list).reshape(1,13)
    loaded_model = pickle.load(open("heart_disease_model.pkl", "rb"))
    result = loaded_model.predict(to_predict)
    return result[0]

@app.route('/')
def home():
    return render_template('index.html')
```

```python
@app.route('/predict', methods=['POST'])
def predict():
    if request.method == 'POST':
        predict_list = request.form.to_dict()
        predict_list = list(predict_list.values())
        try:
            predict_list = [int(float(x)) for x in predict_list]
        except ValueError:
            return "Invalid input. Please provide valid integer or float values."

        result = HeartDiseasePredictor(predict_list)
        if int(result) == 1:
            prediction = 'This person is suffering from heart disease..'
        else:
            prediction = 'This person is not suffering from heart disease..'
        return render_template("index.html", prediction=prediction)


if __name__=="__main__":
    app.run(port=5000)


Run Cell | Run Above | Debug Cell
# In[ ]:
```

App form

A HTML file was created for the web application form

app.py | index.html

C: > Users > berly > OneDrive > Desktop > Data Glacier_intern > Week 5 > New folder > templates > ind

```html
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Heart Disease Predictor</title>
    <style>
        body {
            font-family: Arial, sans-serif;
            background-color: #19dfdfef;
        }
        h3 {
            color: #333;
        }
        form {
            background-color: #f5e08bd3;
            border-radius: 10px;
            padding: 20px;
            box-shadow: 0 0 10px rgba(0, 0, 0, 0.1);
            width: 500px;
            margin: 0 auto;
        }
        label {
            display: block;
            margin-bottom: 5px;
            color: #555;
        }
        input[type="number"],
        select {
            width: calc(100% - 22px);
            padding: 8px;
            margin-bottom: 10px;
            border: 1px solid #ccc;
            border-radius: 5px;
        }
        input[type="submit"] {
```

```html
        input[type="submit"] {
            width: 100%;
            padding: 10px;
            background-color: #4CAF50;
            color: white;
            border: none;
            border-radius: 5px;
            cursor: pointer;
            font-size: 16px;
        }
        input[type="submit"]:hover {
            background-color: #45a049;
        }
    </style>
</head>
<body>
    <h1>Heart Disease Predictor</h1>
    <div>
        <form action="/predict" method="POST">
            <label for="age">Age</label>
            <input type="number" id="age" name="age">
            <label for="sex">Sex</label>
            <select id="sex" name="sex">
                <option value="0">Female</option>
                <option value="1">Male</option>
            </select>
            <label for="cp">Chest Pain Type</label>
            <select id="cp" name="cp">
                <option value="0">Angina</option>
                <option value="1">Pericarditis</option>
                <option value="2">Myocarditis</option>
                <option value="3">Cardiomyopathy</option>
            </select>
```

App form (contd)

A HTML file was created for the web application form

```html
<label for="trestbps">Resting blood pressure (in mm Hg on admission to the hospital)</label>
<input type="number" id="trestbps" name="trestbps">
<label for="chol">Serum Cholesterol in mg/dl</label>
<input type="number" id="chol" name="chol">
<label for="fbs">Fasting blood sugar > 120 mg/dl</label>
<select id="fbs" name="fbs">
    <option value="0">False</option>
    <option value="1">True</option>
</select>
<label for="restecg">Resting electrocardiographic measurement</label>
<select id="restecg" name="restecg">
    <option value="0">Normal</option>
    <option value="1">ST-T wave abnormality</option>
    <option value="2">Probable or definite left ventricular hypertrophy</option>
</select>
<label for="thalach">Maximum heart rate achieved</label>
<input type="number" id="thalach" name="thalach">
<label for="exang">Exercise induced angina</label>
<select id="exang" name="exang">
    <option value="0">No</option>
    <option value="1">Yes</option>
</select>
<label for="oldpeak">ST depression induced by exercise relative to rest</label>
<input type="number" id="oldpeak" name="oldpeak" step="0.1">

<label for="slope">The slope of the peak exercise ST segment</label>
<select id="slope" name="slope">
    <option value="1">Upsloping</option>
    <option value="2">Flat</option>
    <option value="3">Downsloping</option>
</select>
<label for="ca">Number of major vessels (0-3)</label>
<input type="number" id="ca" name="ca">
<label for="thal">Blood disorder called thalassemia</label>
<select id="thal" name="thal">
    <option value="1">Normal</option>
    <option value="2">Fixed defect</option>
    <option value="3">Reversible defect</option>
</select>
<input type="submit" value="Predict">
</form>
<br>
<br>
{{ prediction }}

</div>
<script
  src="{{ url_for('static',filename='main.js') }}"
  type="text/javascript"
></script>
</body>
</html>
```

API Ap

An API app was created for deploying and testing the model

app.py | index.html | APIapp.py

C: > Users > berly > OneDrive > Desktop > Data Glacier_intern > Week 5 > APIapp.py > ...

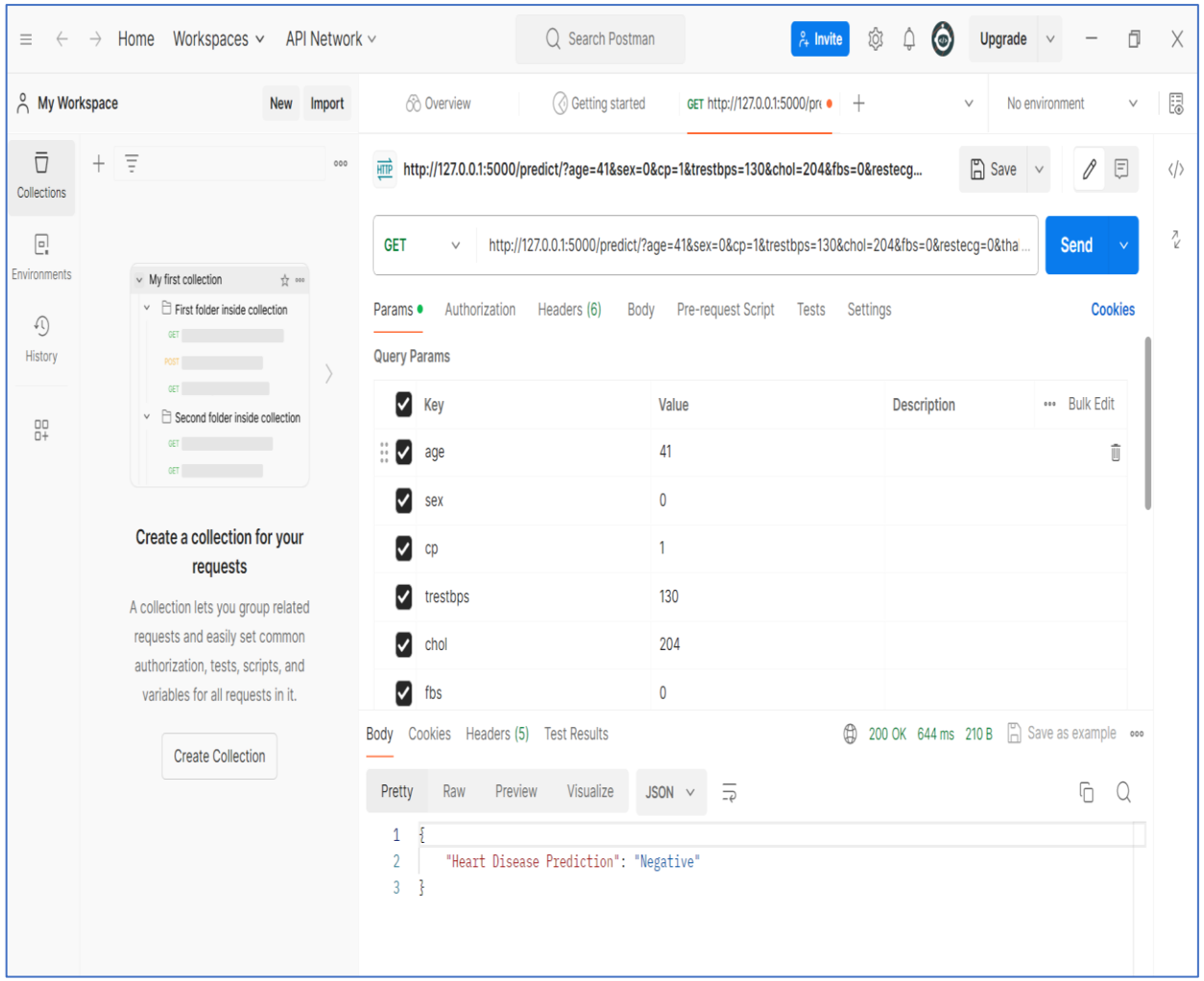
```python
# Using Flask to make an API
# Import the necessary libraries and functions
from flask import Flask, jsonify, request
import pickle
import pandas as pd

# Creating a Flask app
app = Flask(__name__)

@app.route('/', methods = ['GET', 'POST'])
def home ():
    if(request.method == 'GET'):
        data = "Hello, this is my first API"
        return jsonify({'data':data})



@app.route('/predict/')
def heart_disease_predict():
    model = pickle.load(open('heart_disease_model.pkl', 'rb'))
    age = request.args.get('age')
    sex = request.args.get('sex')
    cp = request.args.get('cp')
    trestbps= request.args.get('trestbps')
    chol = request.args.get('chol')
    fbs = request.args.get('fbs')
    restecg = request.args.get('restecg')
    thalach = request.args.get('thalach')
    exang = request.args.get('exang')
    oldpeak = request.args.get('oldpeak')
    slope  = request.args.get('slope')
    ca = request.args.get('ca')
    thal = request.args.get('thal')
```

```python

    test_df = pd.DataFrame({'Age':[age], 'Sex':[sex], 'Chest Pain Type':[cp], 'Resting Blood Pressure':[trestbps],
                            'Serum Cholesterol':[chol], 'Fasting Blood Sugar':[fbs], 'Resting electrocardiographic measurement':[restecg],
                            'Maximum Heart Rate':[thalach], 'Exercise Induce Angina':[exang], 'ST Depression':[oldpeak], 'Slope':[slope],
                            'Blood Vessel':[ca], 'Thalassemia Blood Disorder':[thal]})

    pred_heart_disease = model.predict(test_df)
    if pred_heart_disease == 1:
        return jsonify({'Heart Disease Prediction':"Positive"})
    else:
        return jsonify({'Heart Disease Prediction':"Negative"})

# Driver function
if __name__ == '__main__':
    app.run(debug=True)
```

API Testing

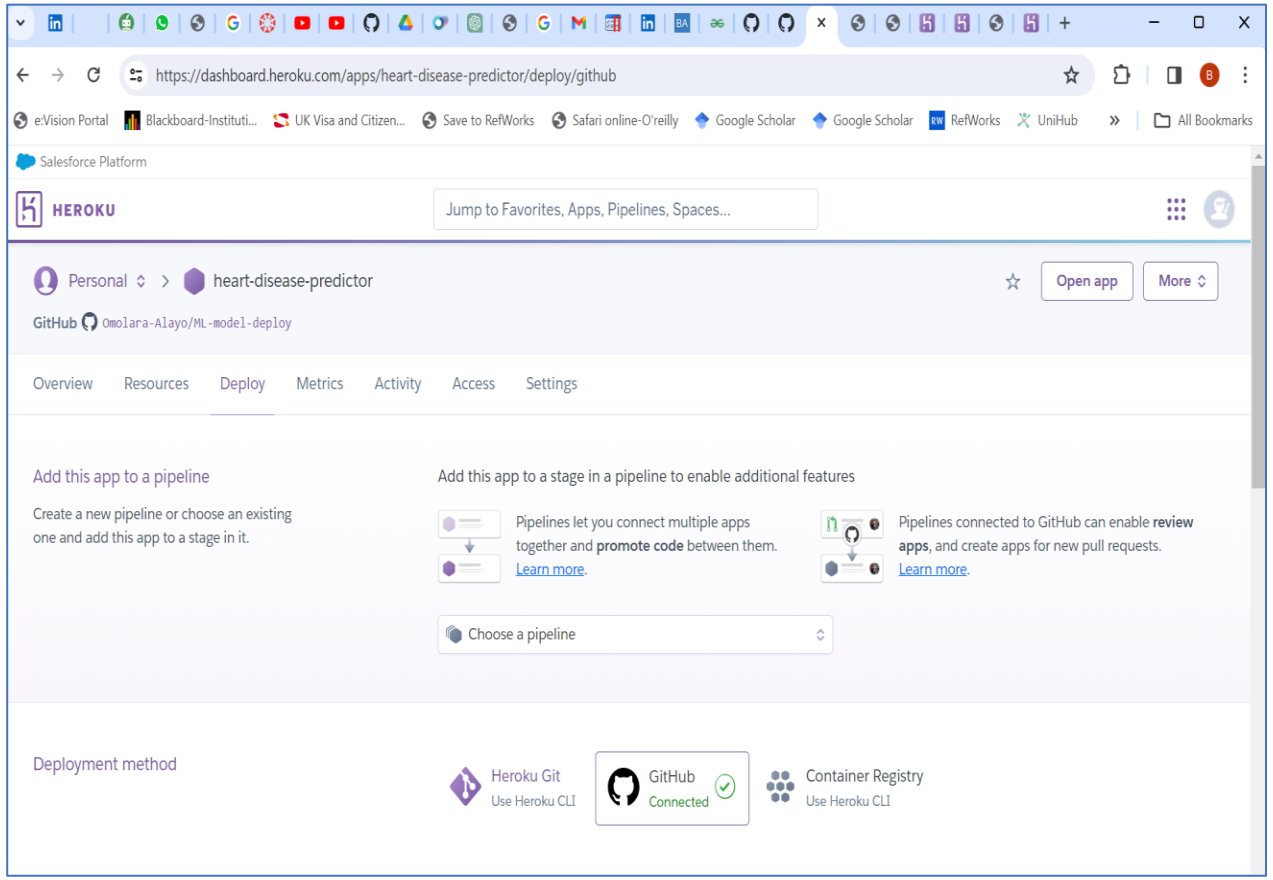
After the creation of all the necessary files, the model was deployed and tested with API

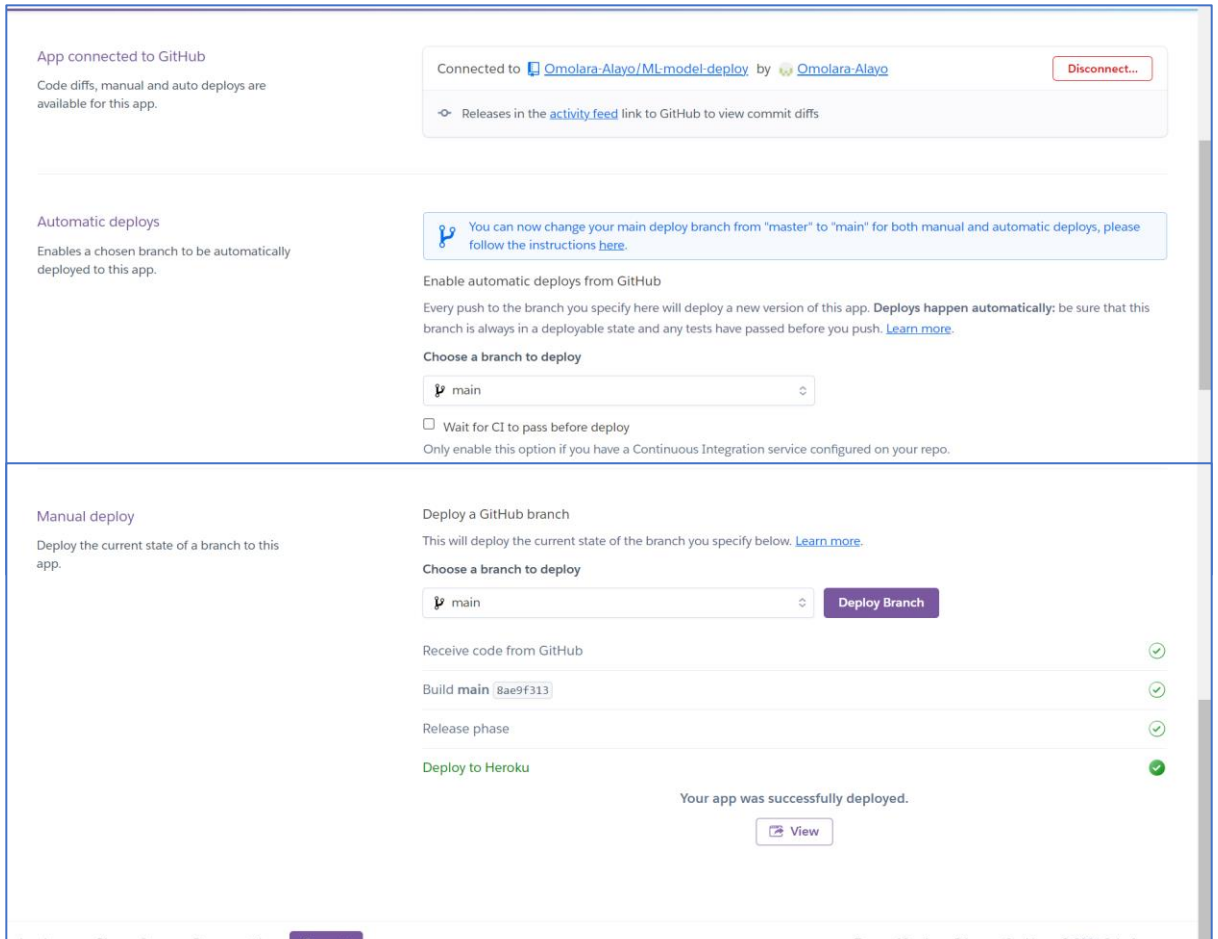
```
Command Prompt - python /

C:\Users\berly\OneDrive>cd desktop

C:\Users\berly\OneDrive\Desktop>cd Data Glacier_intern

C:\Users\berly\OneDrive\Desktop\Data Glacier_intern>cd Week 5

C:\Users\berly\OneDrive\Desktop\Data Glacier_intern\Week 5>dir
 Volume in drive C is OS
 Volume Serial Number is BAED-4E4F

 Directory of C:\Users\berly\OneDrive\Desktop\Data Glacier_intern\Week 5

03/04/2024  15:47    <DIR>          .
02/04/2024  23:45    <DIR>          ..
03/04/2024  15:45             1,827 APIapp.py
29/03/2024  00:17               818 heart_disease_model.pkl
               2 File(s)          2,645 bytes
               2 Dir(s)  284,749,574,144 bytes free

C:\Users\berly\OneDrive\Desktop\Data Glacier_intern\Week 5>python APIapp.py
 * Serving Flask app 'APIapp'
 * Debug mode: on
WARNING: This is a development server. Do not use it in a production deployment. Use a production WSGI server instead.
 * Running on http://127.0.0.1:5000
Press CTRL+C to quit
 * Restarting with stat
 * Debugger is active!
 * Debugger PIN: 274-366-615
```



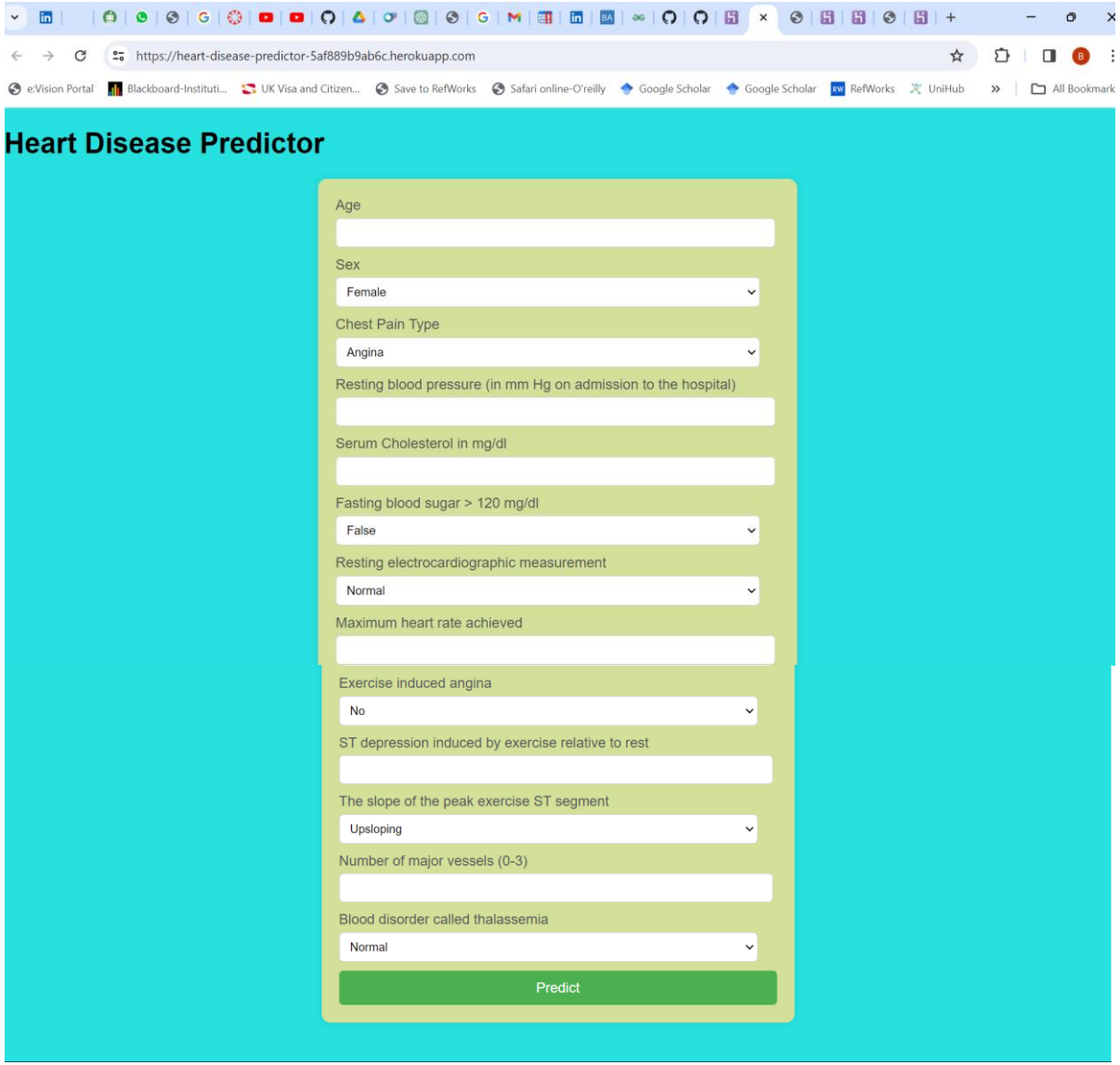
Cloud deployment using Heroku

The model was also deployed to the cloud using Heroku by creating the app and connecting to the GitHub repository where all the files have been committed to.



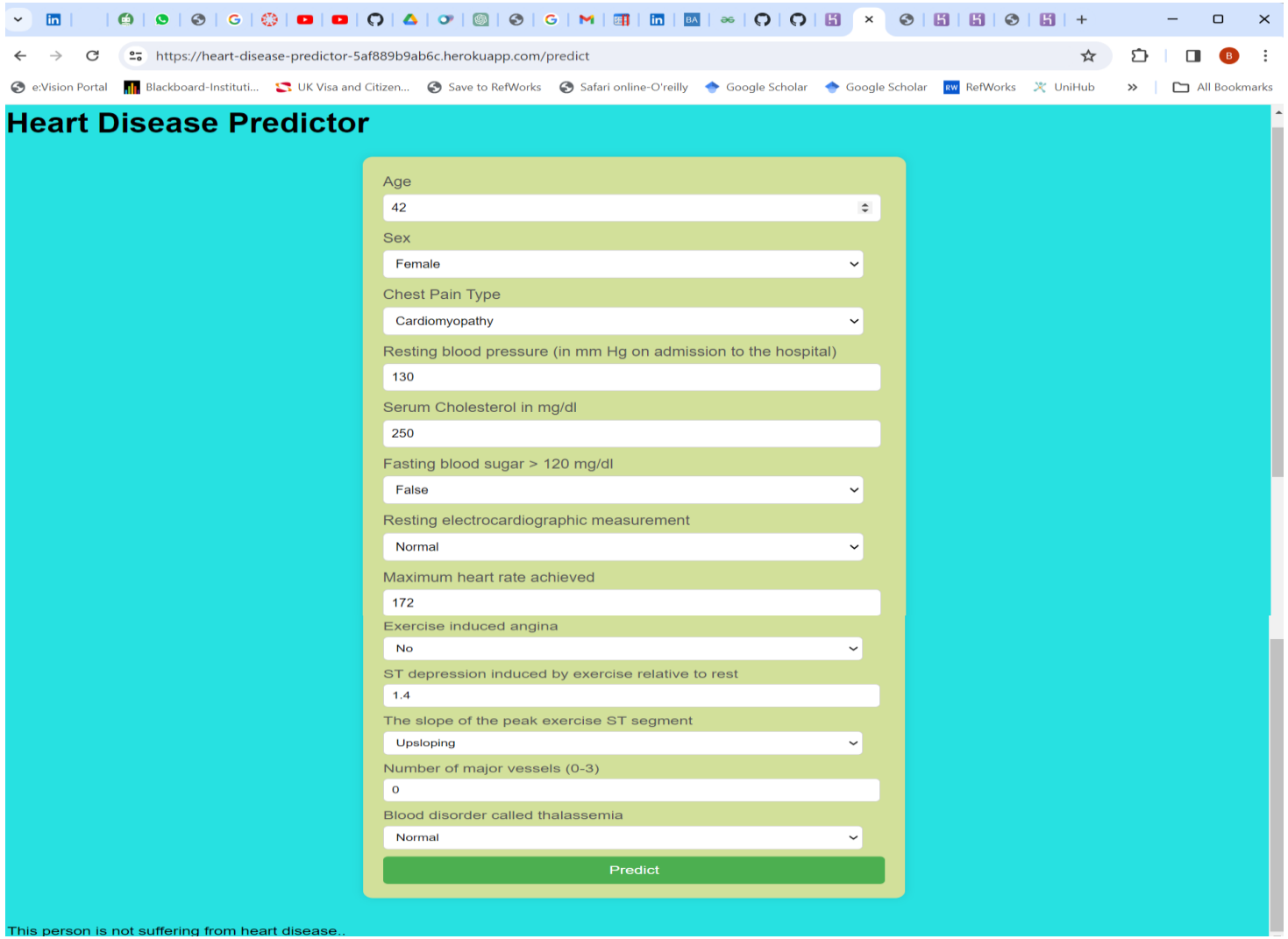


Heart-disease-predictive app



Upon the successful deployment of the model as an app on the web, it was subsequently launched and tested.

Heart-disease-predictive app



- The web app was hosted on: https://heart-disease-predictor-5af889b9ab6c.herokuapp.com/
- It allows any user to enter data and make predictions based on the dataset

Above is the prediction for a particular observation.

Thank You