Data Glacier

Your Deep Learning Partner

Data Science Internship

Week 4: Flask Model Deployment Report

By

Name: Sukurat Salam

Email: salamsukurat@gmail.com

Batch No: LISUM24

Date:17th August 2023

Table of Content

1.0 Introduction

This section will give the overview of the task and how this would be achieved. This project will build a model for predicting heart disease attack in patient using some medical features or characteristics that can easily be provided without going through laboratory testing or x-trays. A supervise learning model approach which is suitable for binary classification logistic regression would be used to train the model and will be deploy using flask.

This would be achieved with the following steps.

- Downloading the data
- Data pre-processing
- Training the model
- Deploying the model using Flask
- Building the web application

2.0 Data Description

For this project, 2015 dataset that was pre-processed by (Alex Teboul 2021) was downloaded from Kaggle. The dataset contains 253,680 survey responses from cleaned BRFSS 2015 to be used mainly for the binary classification of heart disease attack. The dataset contains 21 features with 1 binary target variable, but just 5 most important variable for the prediction would be consider for this project.

Source Link: *https://www.kaggle.com/alexteboul/heart-disease-health-indicators-dataset.*

S/N	Variable	Description	Values
1	HeartDiseaseAttack	Respondents that have reported having coronary heart disease or Myocardial Infection	0 for No 1 for Yes
2	BMI	Body Mass Index	Numeric
3	Age	Patient Age Category "1= 18-24, 2= 25-30,3= 31-35, 4= 36-40,5= 41-45,6= 46-50,7= 51-55,8= 56-60,9= 61-65,10= 66-70,11= 71-75,12 =76-80 ,13 = 80 above"	
4	Gender	Patient Sex	0 for Female, 1 for Male

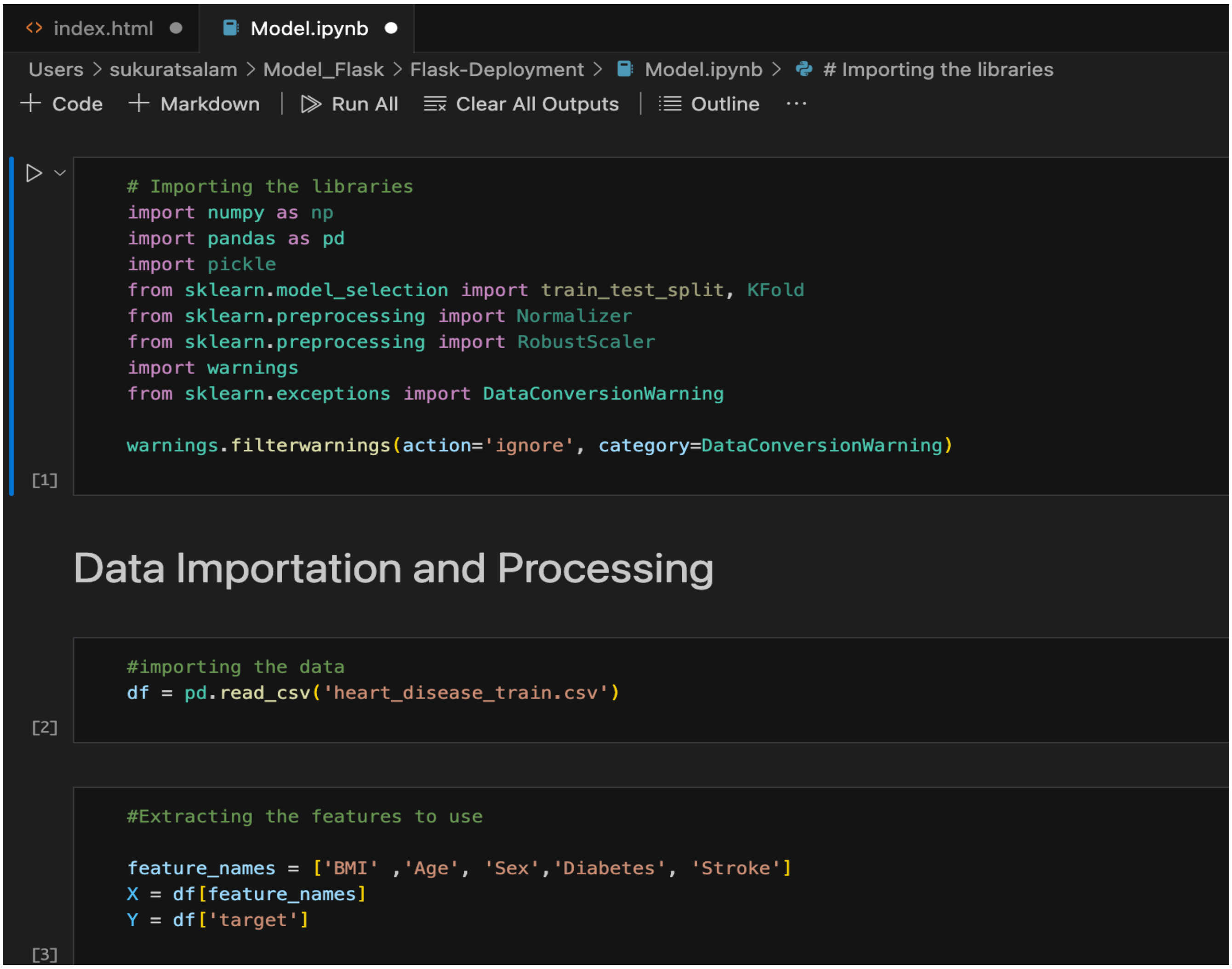
5	Diabetes	"0 for No Diabetes, 1 for Pre-Diabetes, 2 for Diabetes"	0 for No Diabetes, 1 for Pre-Diabetes, 2 for Diabetes
6	Stroke	Ever told you have stroke?	0 for No, 1 for Yes

3.0 Machine Learning Model Implementation

The first step to take when deploying a machine learning model in Flask is to train the model, and for the research, we will employ the use of logistic regression for binary classification of the heart disease attack (if a patient has had it in the past or not).

After training the model, it will be sterilized using a pickle library to save it in the file which will be loaded later when serving predictions. The training of the model and sterilization was done in Jupiter notebook and the screenshot of some parts of the Python code was given below.

3.1 Data Importation and Pre-processing


index.html
Model.ipynb
Users > sukuratsalam > Model_Flask > Flask-Deployment > Model.ipynb > # Importing the libraries
+ Code
+ Markdown
Run All
Clear All Outputs
Outline
[1]
[2]
[3]

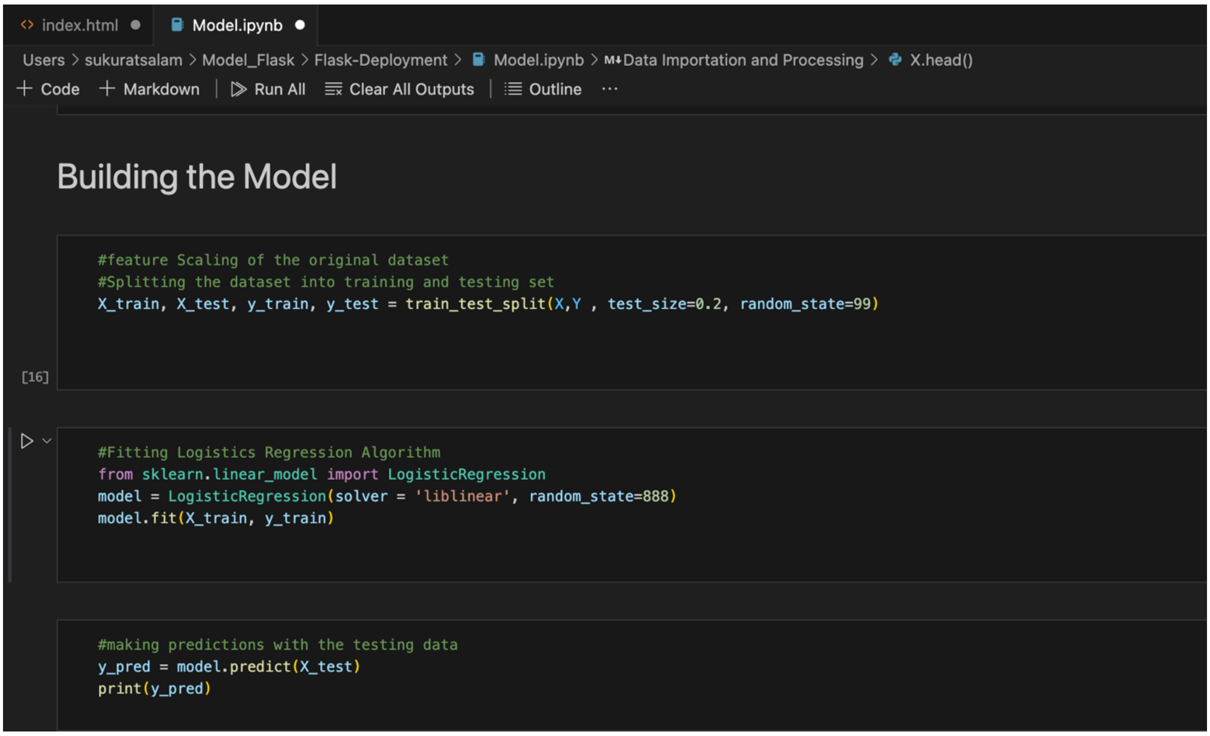
```
# Importing the libraries
import numpy as np
import pandas as pd
import pickle
from sklearn.model_selection import train_test_split, KFold
from sklearn.preprocessing import Normalizer
from sklearn.preprocessing import RobustScaler
import warnings
from sklearn.exceptions import DataConversionWarning

warnings.filterwarnings(action='ignore', category=DataConversionWarning)
```

Data Importation and Processing

```
#importing the data
df = pd.read_csv('heart_disease_train.csv')
```

```
#Extracting the features to use

feature_names = ['BMI' ,'Age', 'Sex','Diabetes', 'Stroke']
X = df[feature_names]
Y = df['target']
```

3.2 Training the Logistics Regression Model


index.html ●
Model.ipynb ●
Users > sukuratsalam > Model_Flask > Flask-Deployment > Model.ipynb > M↓Data Importation and Processing > X.head()
+ Code
+ Markdown
Run All
Clear All Outputs
Outline
···
[16]

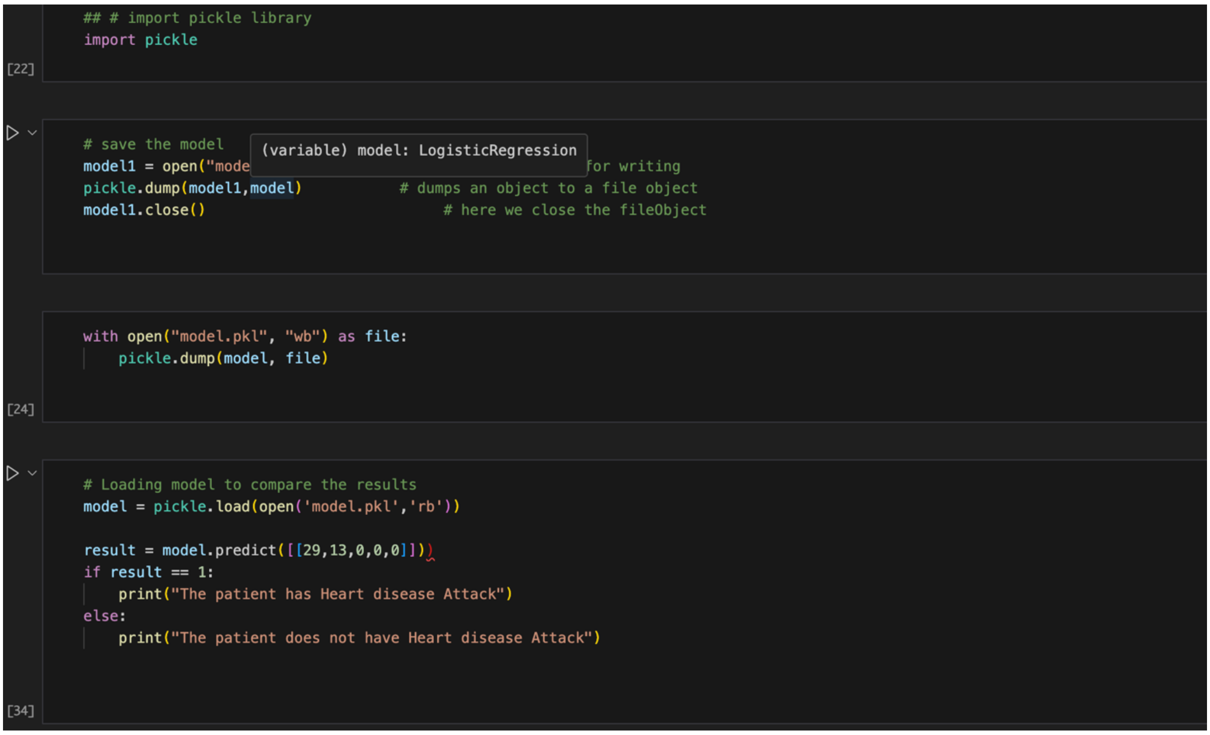
Building the Model

```
#feature Scaling of the original dataset
#Splitting the dataset into training and testing set
X_train, X_test, y_train, y_test = train_test_split(X,Y , test_size=0.2, random_state=99)
```

```
#Fitting Logistics Regression Algorithm
from sklearn.linear_model import LogisticRegression
model = LogisticRegression(solver = 'liblinear', random_state=888)
model.fit(X_train, y_train)
```

```
#making predictions with the testing data
y_pred = model.predict(X_test)
print(y_pred)
```

3.3 Sterilization of the Model using Pickle



```
## # import pickle library
import pickle
```

```
# save the model
model1 = open("mode                                    for writing
pickle.dump(model1,model)           # dumps an object to a file object
model1.close()                           # here we close the fileObject
```

```
with open("model.pkl", "wb") as file:
    pickle.dump(model, file)
```

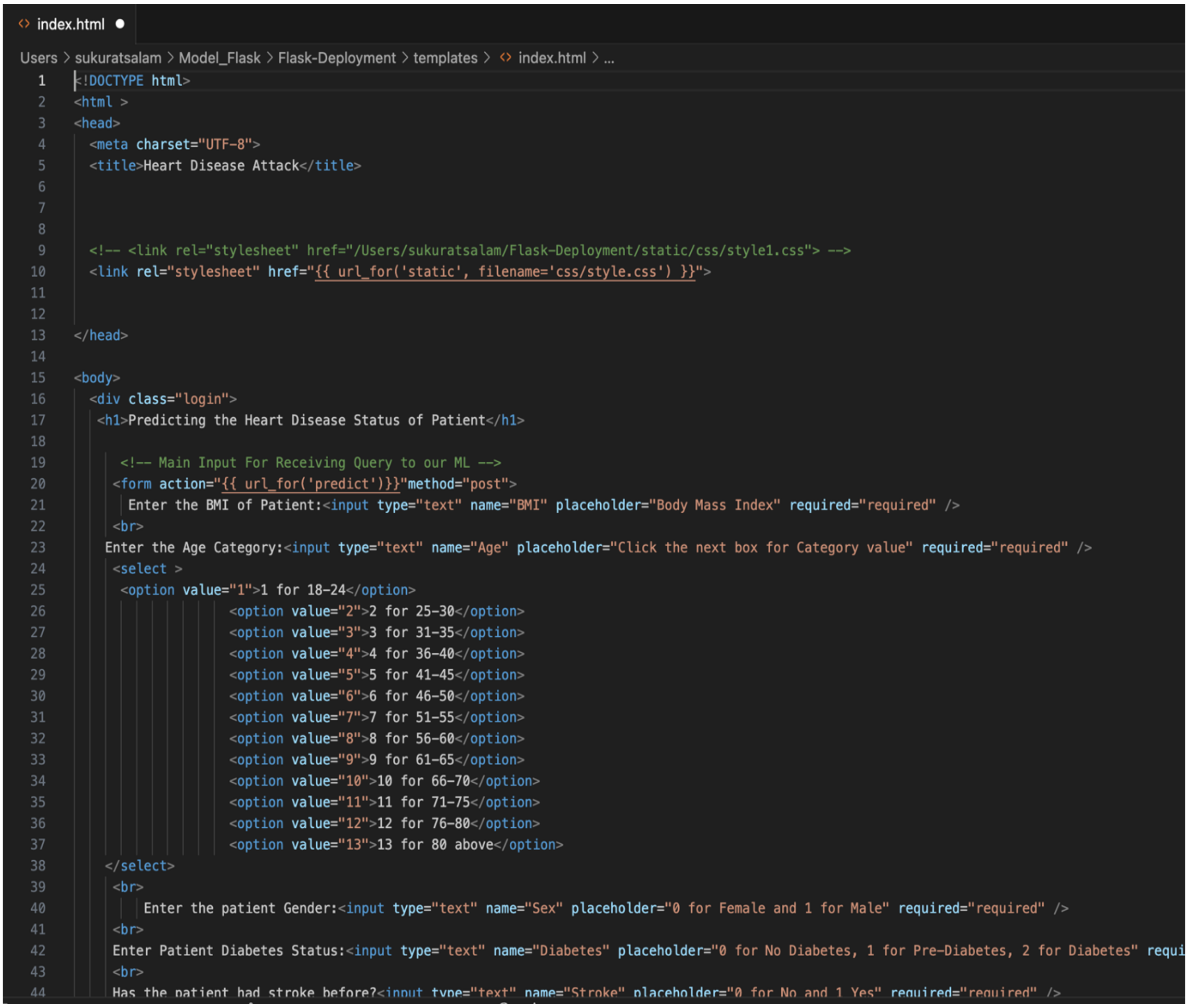
```
# Loading model to compare the results
model = pickle.load(open('model.pkl','rb'))

result = model.predict([[29,13,0,0,0]])
if result == 1:
    print("The patient has Heart disease Attack")
else:
    print("The patient does not have Heart disease Attack")
```

4.0 Model Deployment in Flask

Deploying a machine learning model with Flask entails multiple phases, beginning with the creation of a web page and ending with the model being served via a Python application.

4.1 Create a Web Page

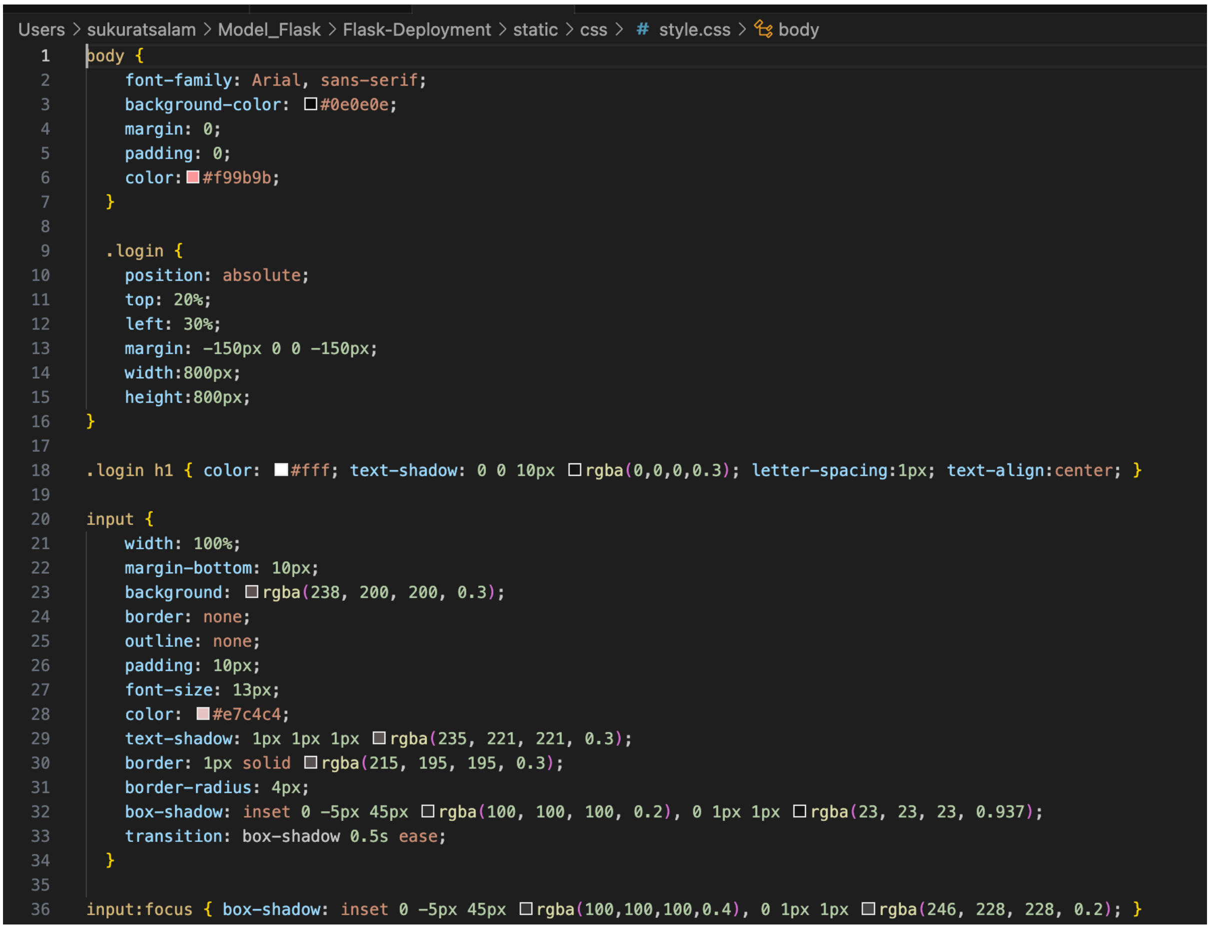
A basic HTML web page was created to interact with the deployed model, This includes the forms in which the user can input data for prediction, and display the result message after the prediction is done.



index.html
Users > sukuratsalam > Model_Flask > Flask-Deployment > templates > index.html > ...

```html
<!DOCTYPE html>
<html >
<head>
  <meta charset="UTF-8">
  <title>Heart Disease Attack</title>



  <!-- <link rel="stylesheet" href="/Users/sukuratsalam/Flask-Deployment/static/css/style1.css"> -->
  <link rel="stylesheet" href="{{ url_for('static', filename='css/style.css') }}">


</head>

<body>
  <div class="login">
   <h1>Predicting the Heart Disease Status of Patient</h1>

      <!-- Main Input For Receiving Query to our ML -->
     <form action="{{ url_for('predict')}}"method="post">
       Enter the BMI of Patient:<input type="text" name="BMI" placeholder="Body Mass Index" required="required" />
     <br>
    Enter the Age Category:<input type="text" name="Age" placeholder="Click the next box for Category value" required="required" />
     <select >
      <option value="1">1 for 18-24</option>
                    <option value="2">2 for 25-30</option>
                    <option value="3">3 for 31-35</option>
                    <option value="4">4 for 36-40</option>
                    <option value="5">5 for 41-45</option>
                    <option value="6">6 for 46-50</option>
                    <option value="7">7 for 51-55</option>
                    <option value="8">8 for 56-60</option>
                    <option value="9">9 for 61-65</option>
                    <option value="10">10 for 66-70</option>
                    <option value="11">11 for 71-75</option>
                    <option value="12">12 for 76-80</option>
                    <option value="13">13 for 80 above</option>
    </select>
     <br>
         Enter the patient Gender:<input type="text" name="Sex" placeholder="0 for Female and 1 for Male" required="required" />
     <br>
     Enter Patient Diabetes Status:<input type="text" name="Diabetes" placeholder="0 for No Diabetes, 1 for Pre-Diabetes, 2 for Diabetes" requi
     <br>
     Has the patient had stroke before?<input type="text" name="Stroke" placeholder="0 for No and 1 Yes" required="required" />
```

CSS/Style Page

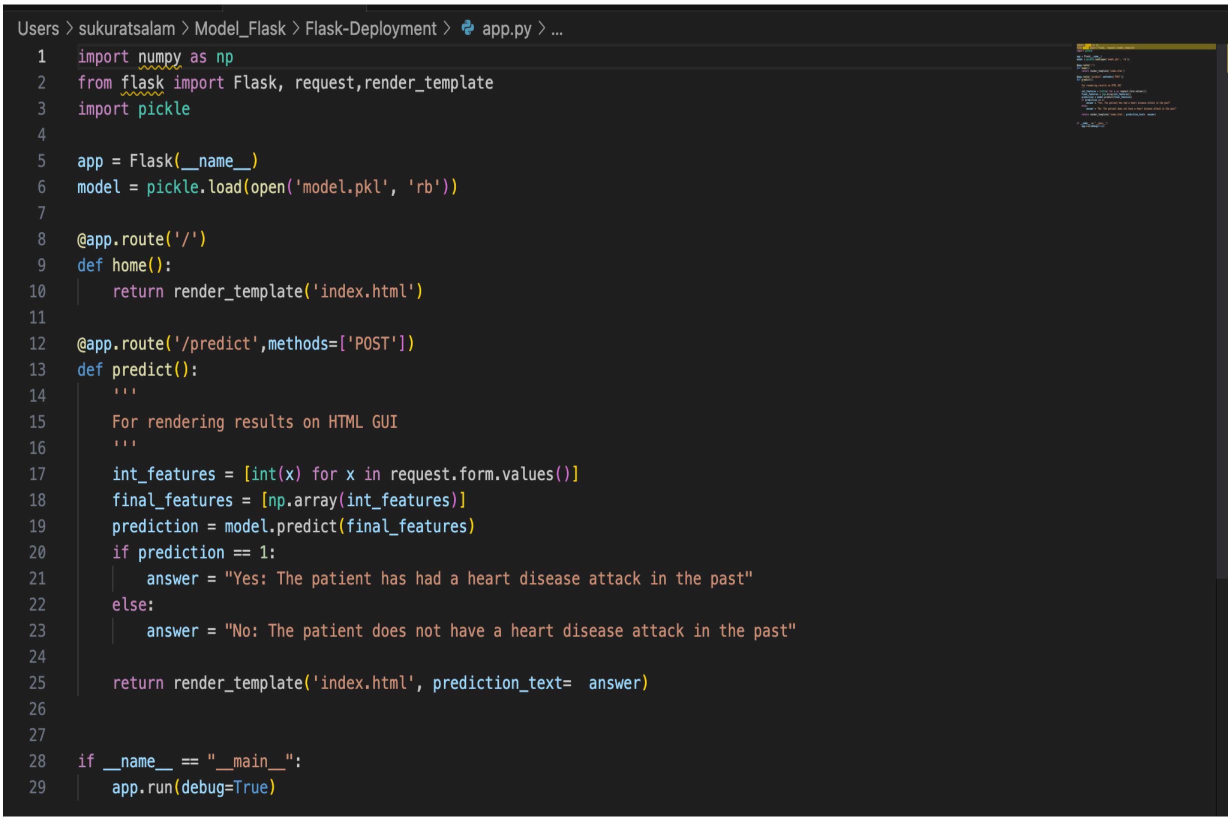

Users > sukuratsalam > Model_Flask > Flask-Deployment > static > css > # style.css > body

```css
body {
    font-family: Arial, sans-serif;
    background-color: #0e0e0e;
    margin: 0;
    padding: 0;
    color:#f99b9b;
  }

  .login {
    position: absolute;
    top: 20%;
    left: 30%;
    margin: -150px 0 0 -150px;
    width:800px;
    height:800px;
}

.login h1 { color: #fff; text-shadow: 0 0 10px rgba(0,0,0,0.3); letter-spacing:1px; text-align:center; }

input {
    width: 100%;
    margin-bottom: 10px;
    background: rgba(238, 200, 200, 0.3);
    border: none;
    outline: none;
    padding: 10px;
    font-size: 13px;
    color: #e7c4c4;
    text-shadow: 1px 1px 1px rgba(235, 221, 221, 0.3);
    border: 1px solid rgba(215, 195, 195, 0.3);
    border-radius: 4px;
    box-shadow: inset 0 -5px 45px rgba(100, 100, 100, 0.2), 0 1px 1px rgba(23, 23, 23, 0.937);
    transition: box-shadow 0.5s ease;
  }

input:focus { box-shadow: inset 0 -5px 45px rgba(100,100,100,0.4), 0 1px 1px rgba(246, 228, 228, 0.2); }
```

4.2 Flask Application Setup

Flask which is the micro web framework for Python was used to create an application that handles incoming HTTP requests and serves the prediction. This stage involves several steps which include,

- installation of flask
- app structure for organizing the project
- creating the flask app by importing flask to create app.py
- loading the serialized model (**model. pkl**) using pickle
- defining the route for the application, we have a route to render the HTML form, and another one was created to handle the predictions

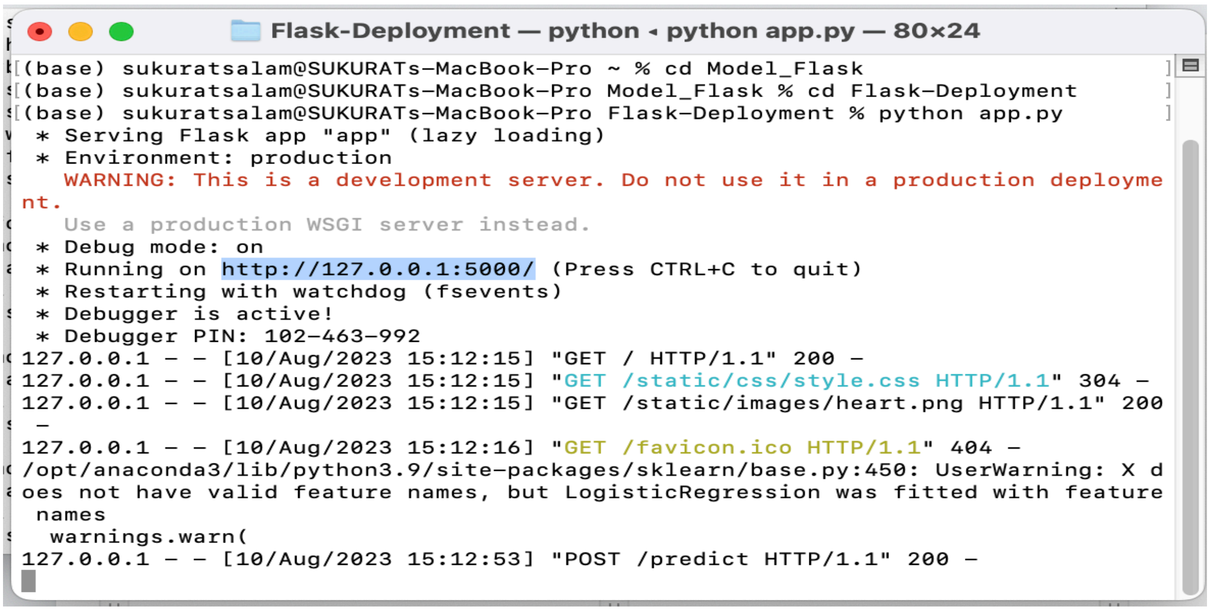


Users > sukuratsalam > Model_Flask > Flask-Deployment > app.py > ...

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

app = Flask(__name__)
model = pickle.load(open('model.pkl', 'rb'))

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

@app.route('/predict',methods=['POST'])
def predict():
    '''
    For rendering results on HTML GUI
    '''
    int_features = [int(x) for x in request.form.values()]
    final_features = [np.array(int_features)]
    prediction = model.predict(final_features)
    if prediction == 1:
        answer = "Yes: The patient has had a heart disease attack in the past"
    else:
        answer = "No: The patient does not have a heart disease attack in the past"

    return render_template('index.html', prediction_text=  answer)


if __name__ == "__main__":
    app.run(debug=True)
```

4.3 Running the Application

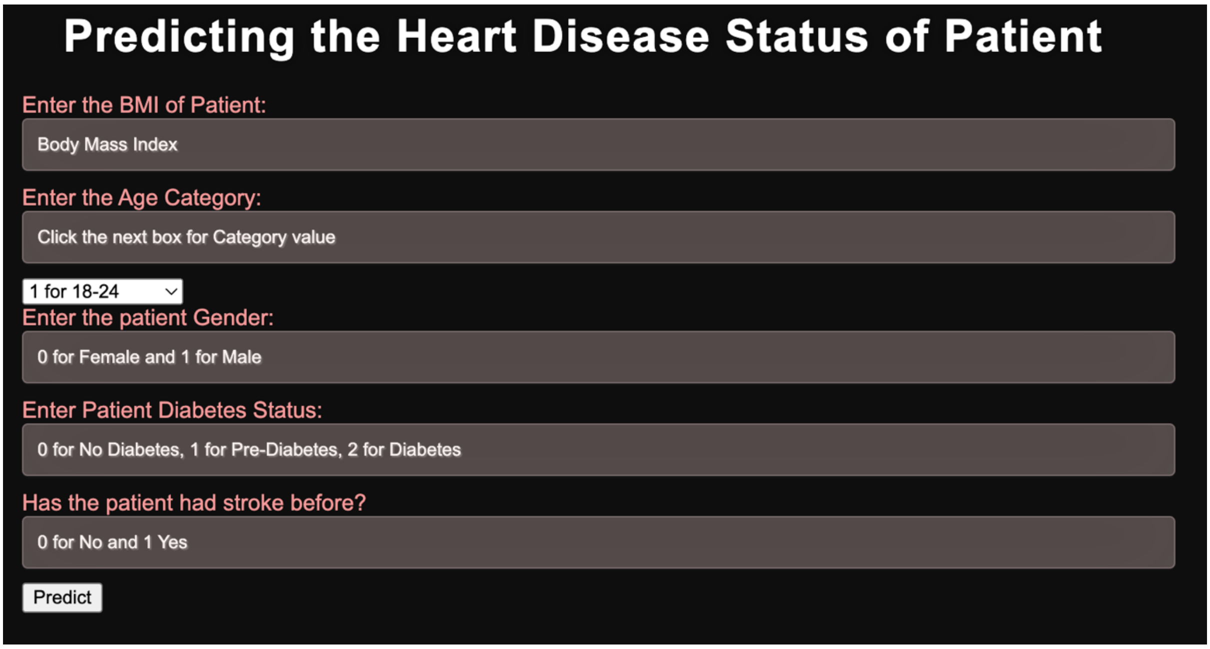
The application was run in the terminal by navigating to the Flask model- deployment folder and running the code *python app.py* to start the Flask app.



```
(base) sukuratsalam@SUKURATs-MacBook-Pro ~ % cd Model_Flask
(base) sukuratsalam@SUKURATs-MacBook-Pro Model_Flask % cd Flask-Deployment
(base) sukuratsalam@SUKURATs-MacBook-Pro Flask-Deployment % python app.py
 * Serving Flask app "app" (lazy loading)
 * Environment: production
   WARNING: This is a development server. Do not use it in a production deployme
nt.
   Use a production WSGI server instead.
 * Debug mode: on
 * Running on http://127.0.0.1:5000/ (Press CTRL+C to quit)
 * Restarting with watchdog (fsevents)
 * Debugger is active!
 * Debugger PIN: 102-463-992
127.0.0.1 - - [10/Aug/2023 15:12:15] "GET / HTTP/1.1" 200 -
127.0.0.1 - - [10/Aug/2023 15:12:15] "GET /static/css/style.css HTTP/1.1" 304 -
127.0.0.1 - - [10/Aug/2023 15:12:15] "GET /static/images/heart.png HTTP/1.1" 200
 -
127.0.0.1 - - [10/Aug/2023 15:12:16] "GET /favicon.ico HTTP/1.1" 404 -
/opt/anaconda3/lib/python3.9/site-packages/sklearn/base.py:450: UserWarning: X d
oes not have valid feature names, but LogisticRegression was fitted with feature
 names
  warnings.warn(
127.0.0.1 - - [10/Aug/2023 15:12:53] "POST /predict HTTP/1.1" 200 -
```

4.4 Access the Web Page

To visit the web page, open your web browser and navigate to http://127.0.0.1:5000/. You can now enter data, submit the form, and view the results.



Predicting the Heart Disease Status of Patient

Enter the BMI of Patient: Body Mass Index

Enter the Age Category: Click the next box for Category value

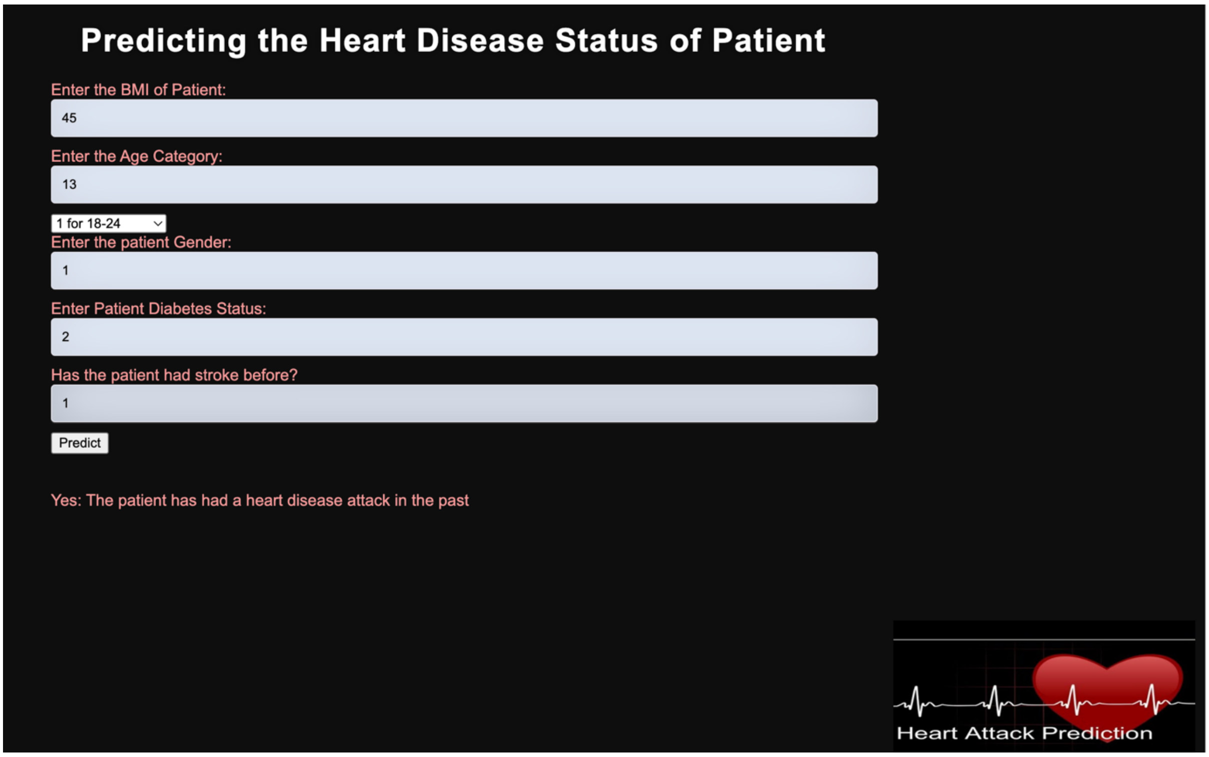
1 for 18-24

Enter the patient Gender: 0 for Female and 1 for Male

Enter Patient Diabetes Status: 0 for No Diabetes, 1 for Pre-Diabetes, 2 for Diabetes

Has the patient had stroke before? 0 for No and 1 Yes

Input and Result Page



Predicting the Heart Disease Status of Patient

Enter the BMI of Patient: 45

Enter the Age Category: 13

1 for 18-24

Enter the patient Gender: 1

Enter Patient Diabetes Status: 2

Has the patient had stroke before? 1

Yes: The patient has had a heart disease attack in the past

4.0 Model Deployment in Heroku

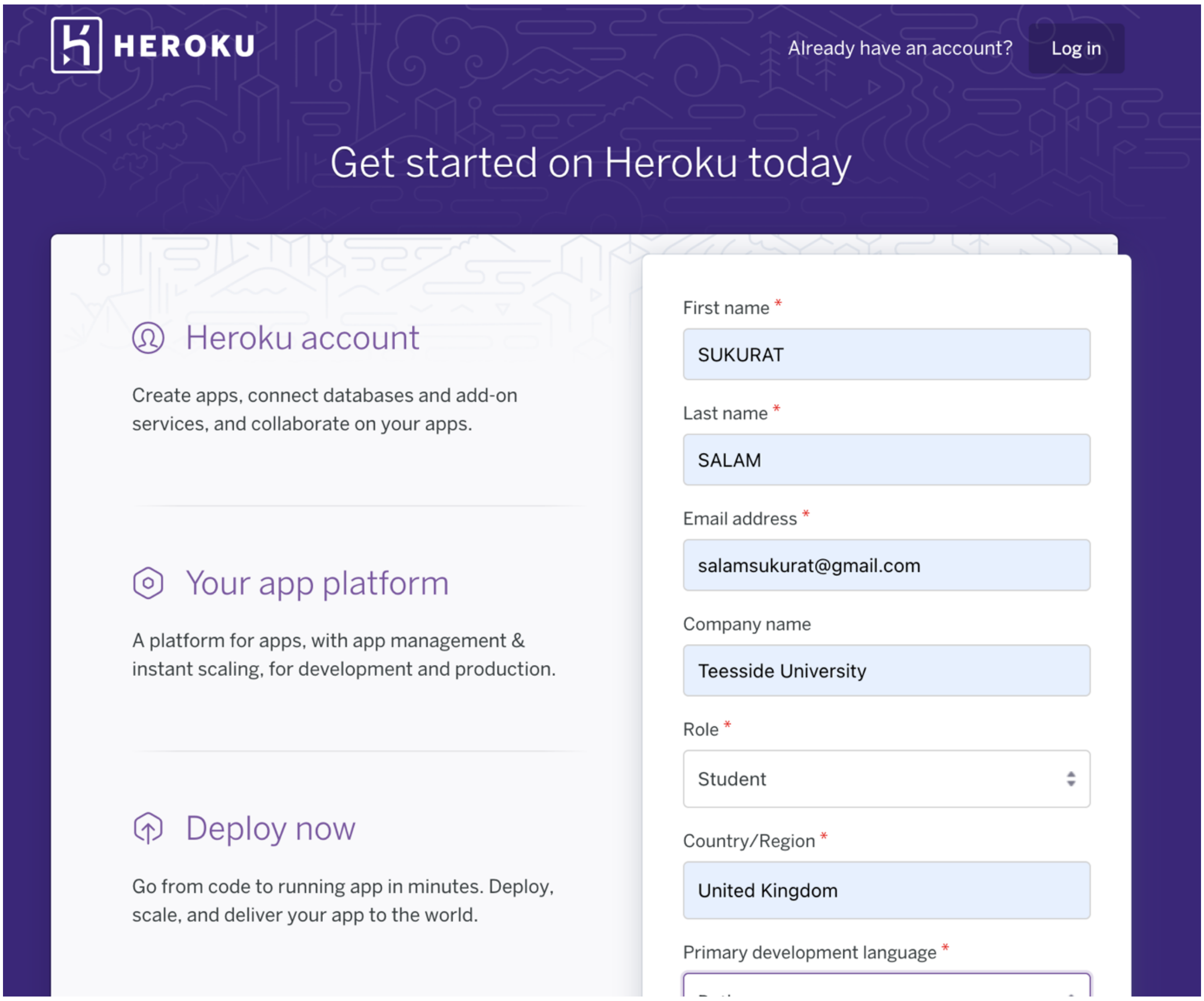
After the deployment of our machine learning model locally, we proceeded to deploy the model in the cloud using the Heroku API. There are several methods to achieve this but for this assignment, the model was deployed by connecting to the GitHub repository that contains the app file through my HEROKU account.

The **Requirement.txt** which is a file that contains the required Python package to run the app was created and the file was pushed to the repository containing the app files and the machine learning model to deployed.

The **Procfile** file with no file extension was created in the app's directory, Heroku uses a **Procfile** to determine how to run your app. The command Heroku should use to launch the app in this file was specified and this is given as **web: gunicorn app:app.**

The following steps were taken to deploy the model in the cloud using Heroku

1. Create a Heroku Account: An account was created at heroku.com



Get started on Heroku today

Heroku account

Create apps, connect databases and add-on services, and collaborate on your apps.

Your app platform

A platform for apps, with app management & instant scaling, for development and production.

Deploy now

Go from code to running app in minutes. Deploy, scale, and deliver your app to the world.

First name *
SUKURAT

Last name *
SALAM

Email address *
salamsukurat@gmail.com

Company name
Teesside University

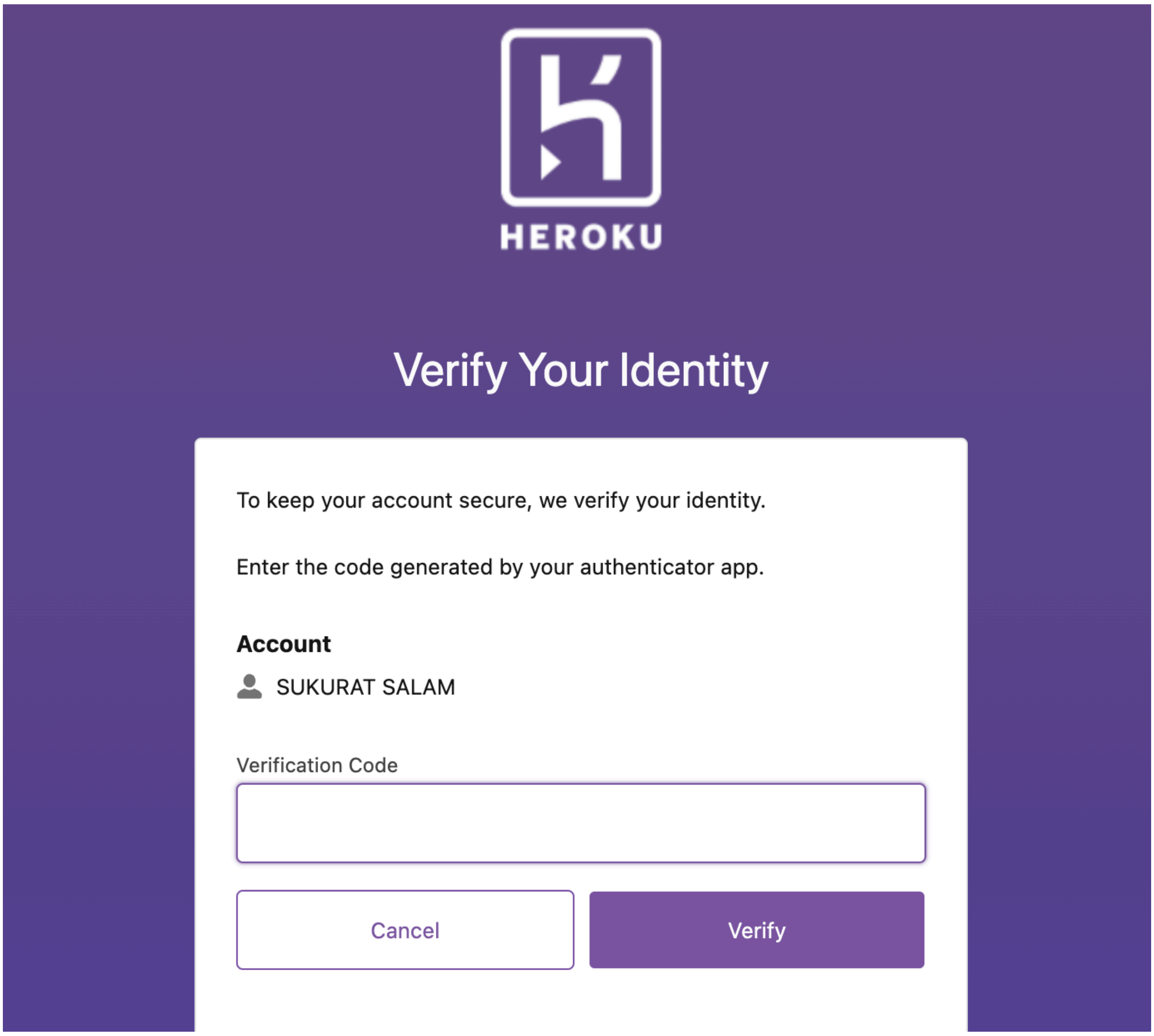
Role *
Student

Country/Region *
United Kingdom

Primary development language *

2. Install Heroku CLI: Download and install the Heroku Command Line Interface (CLI) on your local machine if you haven't already.

3. Login to Heroku: Open your terminal and use the heroku login command to authenticate your Heroku account.



Verify Your Identity

To keep your account secure, we verify your identity.

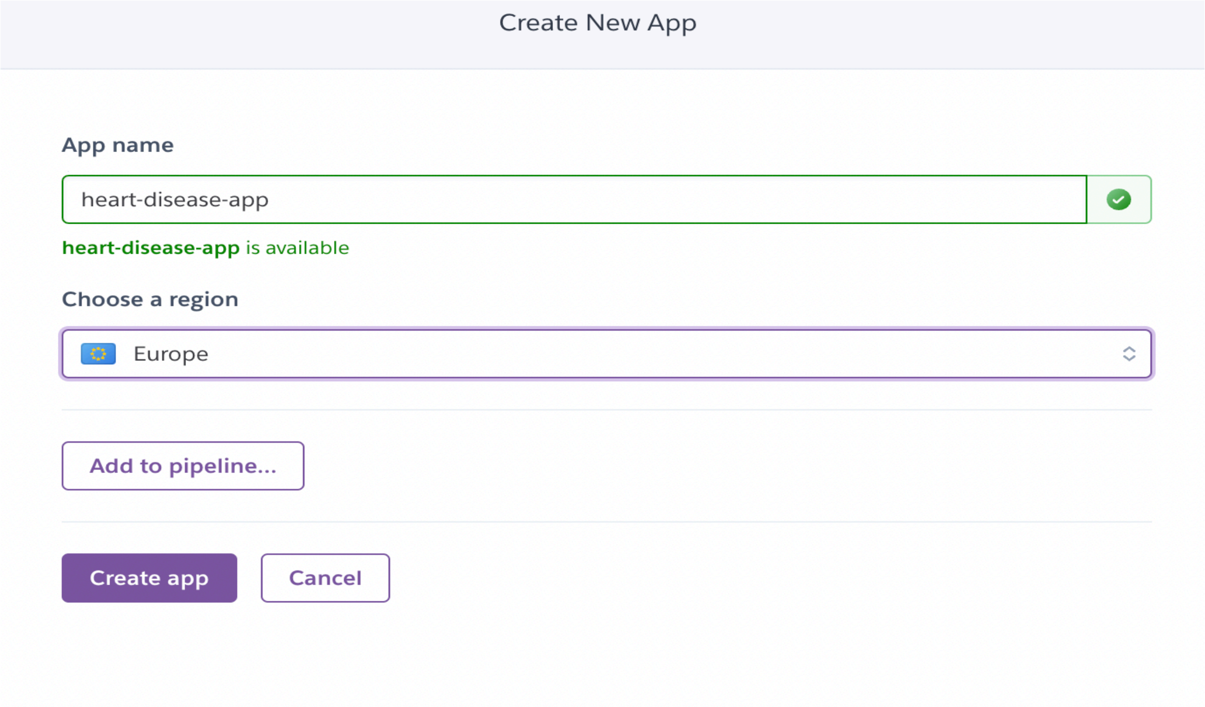
Enter the code generated by your authenticator app.

Account

SUKURAT SALAM

Verification Code

4. Create a New App on Heroku: Run the command heroku create to create a new app on Heroku. This will also add a remote named "heroku" to your local Git repository.



Create New App

App name

heart-disease-app ✓

heart-disease-app is available

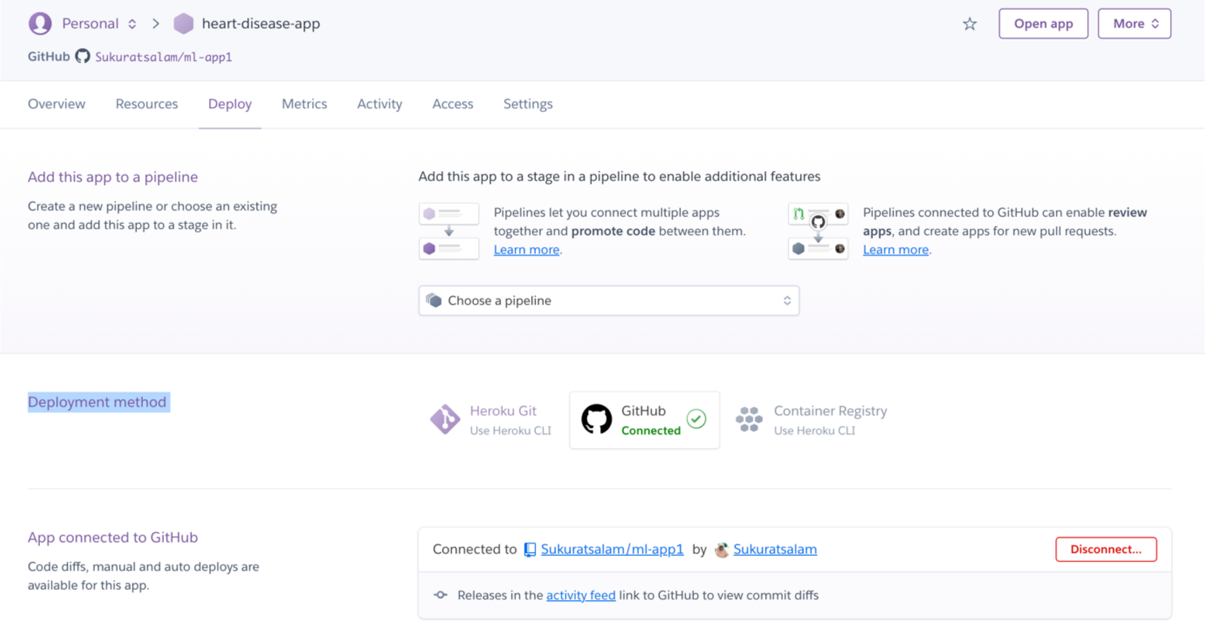
Choose a region

Europe

5. Connect to GitHub

6. Enable Automatic Deployments: In your Heroku Dashboard, go to the "Deploy" tab for your app. Find the "Deployment method" section and select "GitHub" as the deployment method.

7. Connect to GitHub Repository: Heroku prompted us to connect to the GitHub account and select the repository to deploy. Once connected, we choose the repository **ml-app1** and the main branch that has the model we want to deploy.



Personal > heart-disease-app

GitHub Sukuratsalam/ml-app1

Add this app to a pipeline

Create a new pipeline or choose an existing one and add this app to a stage in it.

Add this app to a stage in a pipeline to enable additional features

Pipelines let you connect multiple apps together and **promote code** between them. Learn more.

Pipelines connected to GitHub can enable **review apps**, and create apps for new pull requests. Learn more.

Choose a pipeline

Deployment method

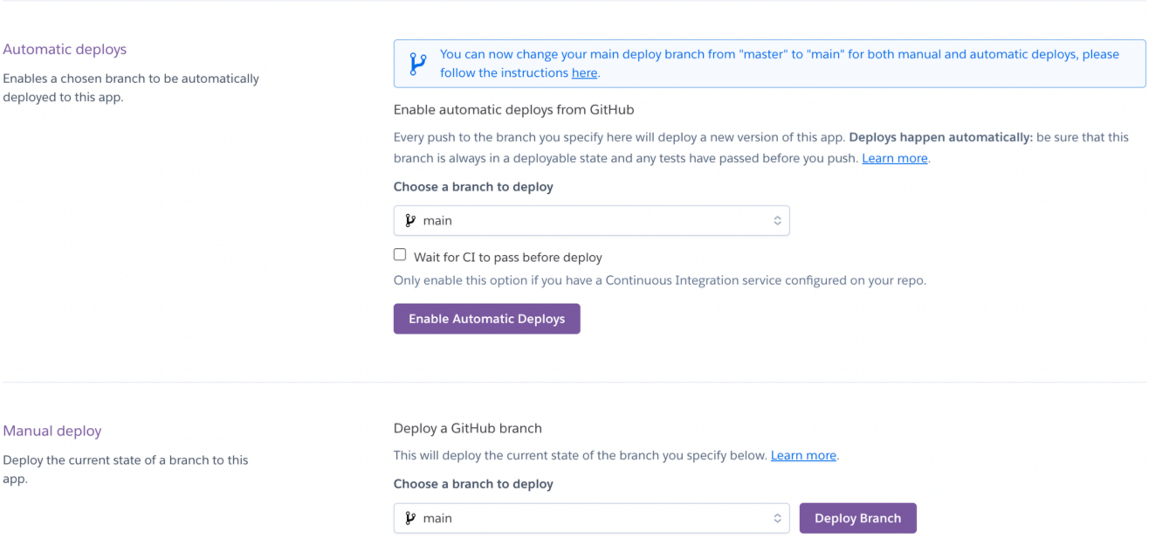
App connected to GitHub

Code diffs, manual and auto deploys are available for this app.

Connected to Sukuratsalam/ml-app1 by Sukuratsalam

Releases in the activity feed link to GitHub to view commit diffs

8. Enable Automatic Deploys: Below the repository selection, you'll find an option to enable automatic deploys. This means that whenever we push changes to the selected branch on GitHub, Heroku will automatically trigger a deployment.



Automatic deploys

Enables a chosen branch to be automatically deployed to this app.

You can now change your main deploy branch from "master" to "main" for both manual and automatic deploys, please follow the instructions here.

Enable automatic deploys from GitHub

Every push to the branch you specify here will deploy a new version of this app. **Deploys happen automatically:** be sure that this branch is always in a deployable state and any tests have passed before you push. Learn more.

Choose a branch to deploy

main

□ Wait for CI to pass before deploy

Only enable this option if you have a Continuous Integration service configured on your repo.

Enable Automatic Deploys

Manual deploy

Deploy the current state of a branch to this app.

Deploy a GitHub branch

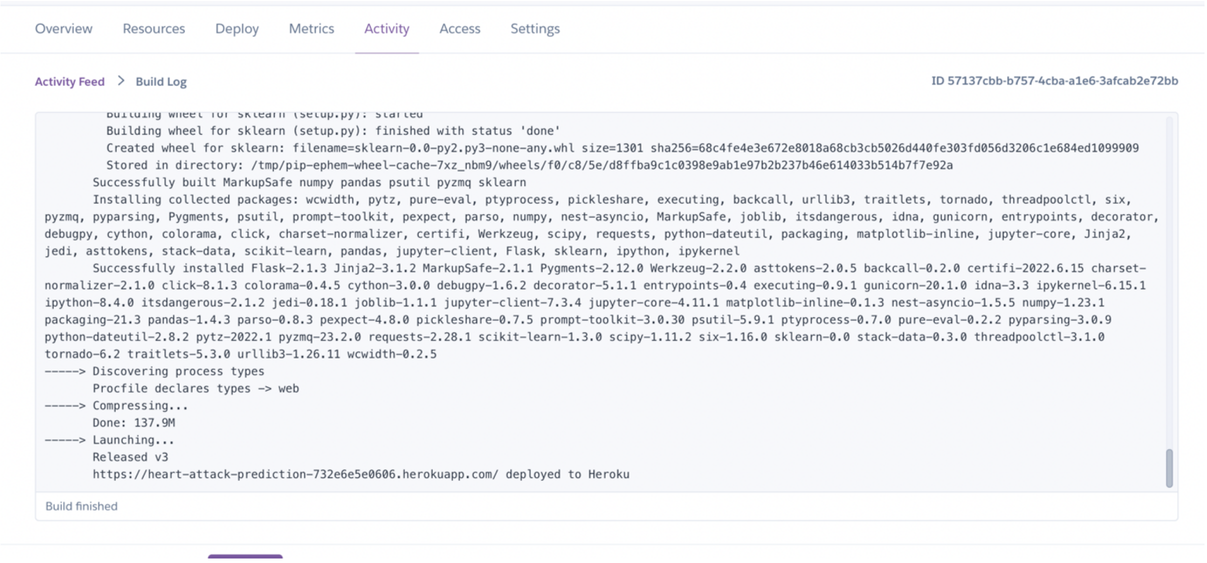
This will deploy the current state of the branch you specify below. Learn more.

Choose a branch to deploy

main

Deploy Branch

9. View Your Deployed App: Once the deployment is successful, Heroku provide us with a URL where the app was hosted. This can be open with the URL below in a web browser to see our deployed app.



Activity Feed > Build Log

ID 57137cbb-b757-4cba-a1e6-3afcab2e72bb

```
         Building wheel for sklearn (setup.py): started
         Building wheel for sklearn (setup.py): finished with status 'done'
         Created wheel for sklearn: filename=sklearn-0.0-py2.py3-none-any.whl size=1301 sha256=68c4fe4e3e672e8018a68cb3cb5026d440fe303fd056d3206c1e684ed1099909
         Stored in directory: /tmp/pip-ephem-wheel-cache-7xz_nbm9/wheels/f0/c8/5e/d8ffba9c1c0398e9ab1e97b2b237b46e614033b514b7f7e92a
       Successfully built MarkupSafe numpy pandas psutil pyzmq sklearn
       Installing collected packages: wcwidth, pytz, pure-eval, ptyprocess, pickleshare, executing, backcall, urllib3, traitlets, tornado, threadpoolctl, six,
pyzmq, pyparsing, Pygments, psutil, prompt-toolkit, pexpect, parso, numpy, nest-asyncio, MarkupSafe, joblib, itsdangerous, idna, gunicorn, entrypoints, decorator,
debugpy, cython, colorama, click, charset-normalizer, certifi, Werkzeug, scipy, requests, python-dateutil, packaging, matplotlib-inline, jupyter-core, Jinja2,
jedi, asttokens, stack-data, scikit-learn, pandas, jupyter-client, Flask, sklearn, ipython, ipykernel
       Successfully installed Flask-2.1.3 Jinja2-3.1.2 MarkupSafe-2.1.1 Pygments-2.12.0 Werkzeug-2.2.0 asttokens-2.0.5 backcall-0.2.0 certifi-2022.6.15 charset-
normalizer-2.1.0 click-8.1.3 colorama-0.4.5 cython-3.0.0 debugpy-1.6.2 decorator-5.1.1 entrypoints-0.4 executing-0.9.1 gunicorn-20.1.0 idna-3.3 ipykernel-6.15.1
ipython-8.4.0 itsdangerous-2.1.2 jedi-0.18.1 joblib-1.1.1 jupyter-client-7.3.4 jupyter-core-4.11.1 matplotlib-inline-0.1.3 nest-asyncio-1.5.5 numpy-1.23.1
packaging-21.3 pandas-1.4.3 parso-0.8.3 pexpect-4.8.0 pickleshare-0.7.5 prompt-toolkit-3.0.30 psutil-5.9.1 ptyprocess-0.7.0 pure-eval-0.2.2 pyparsing-3.0.9
python-dateutil-2.8.2 pytz-2022.1 pyzmq-23.2.0 requests-2.28.1 scikit-learn-1.3.0 scipy-1.11.2 six-1.16.0 sklearn-0.0 stack-data-0.3.0 threadpoolctl-3.1.0
tornado-6.2 traitlets-5.3.0 urllib3-1.26.11 wcwidth-0.2.5
-----> Discovering process types
       Procfile declares types -> web
-----> Compressing...
       Done: 137.9M
-----> Launching...
       Released v3
       https://heart-attack-prediction-732e6e5e0606.herokuapp.com/ deployed to Heroku
```

Build finished

URL: https://heart-attack-prediction-732e6e5e0606.herokuapp.com