Data Glacier

Data Science Intern at Data Glacier

Week 5: Cloud API Development

Name: Ishwarya Arulvel

Batch Code: LISUM14

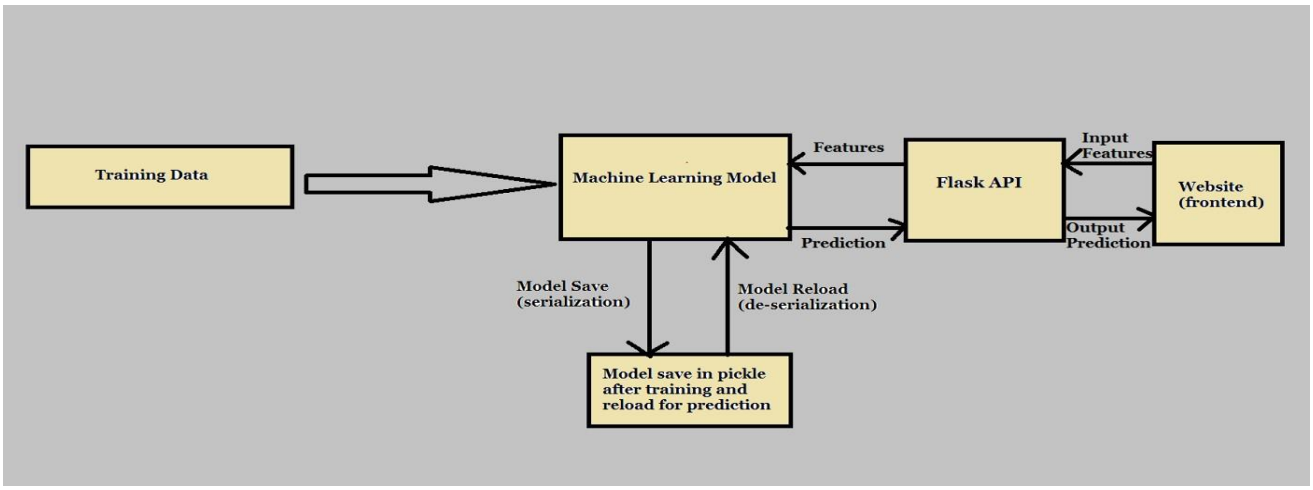
Date: 3 NOV 2022

Submitted to: Data Glacier

Table of Contents:

1. Introduction

In this project, we are going to deploying machine learning model using the Flask Framework. As a demonstration, our model helps to predict the values based on the input given in the heart disease Dataset.

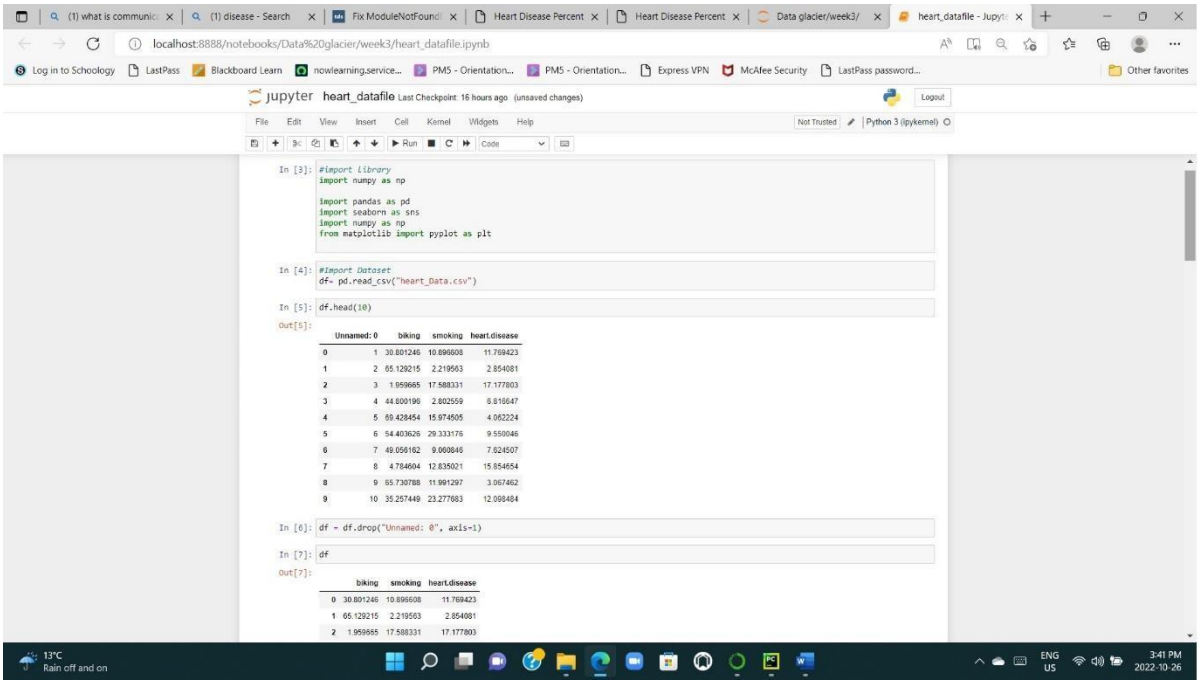


We will focus on both: building a machine learning model for heart disease Dataset, then creating an API for the model, using Flask, the Python micro-framework for building web applications. This API allows us to utilize predictive capabilities through HTTP requests.

2. Building a Model

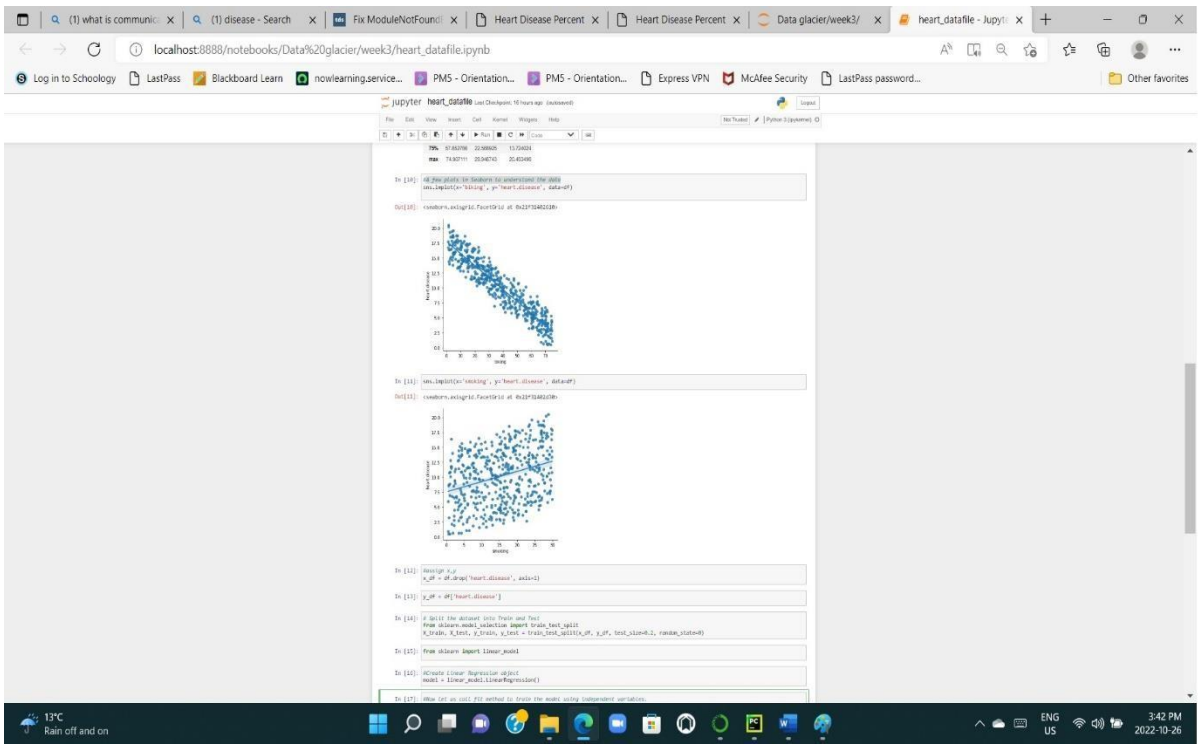
2.1.1 Import Required Libraries and Dataset

In this part, we import libraires and datasets which contain the information of heart disease dataset.



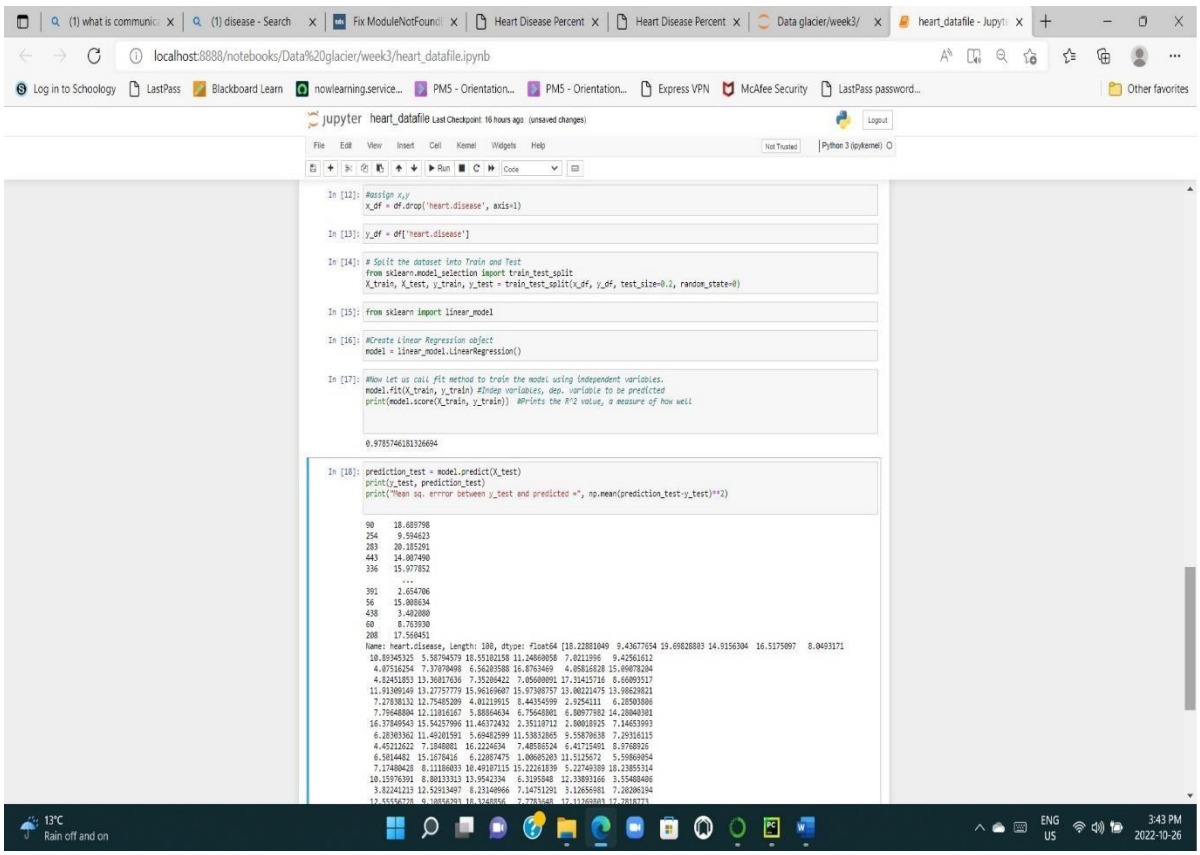
3.1.1 Data Preprocessing

The dataset used here is split into 80% for the training set and the remaining 20% for the test set. Use seaborn library and plots data in Seaborn to understand the data.



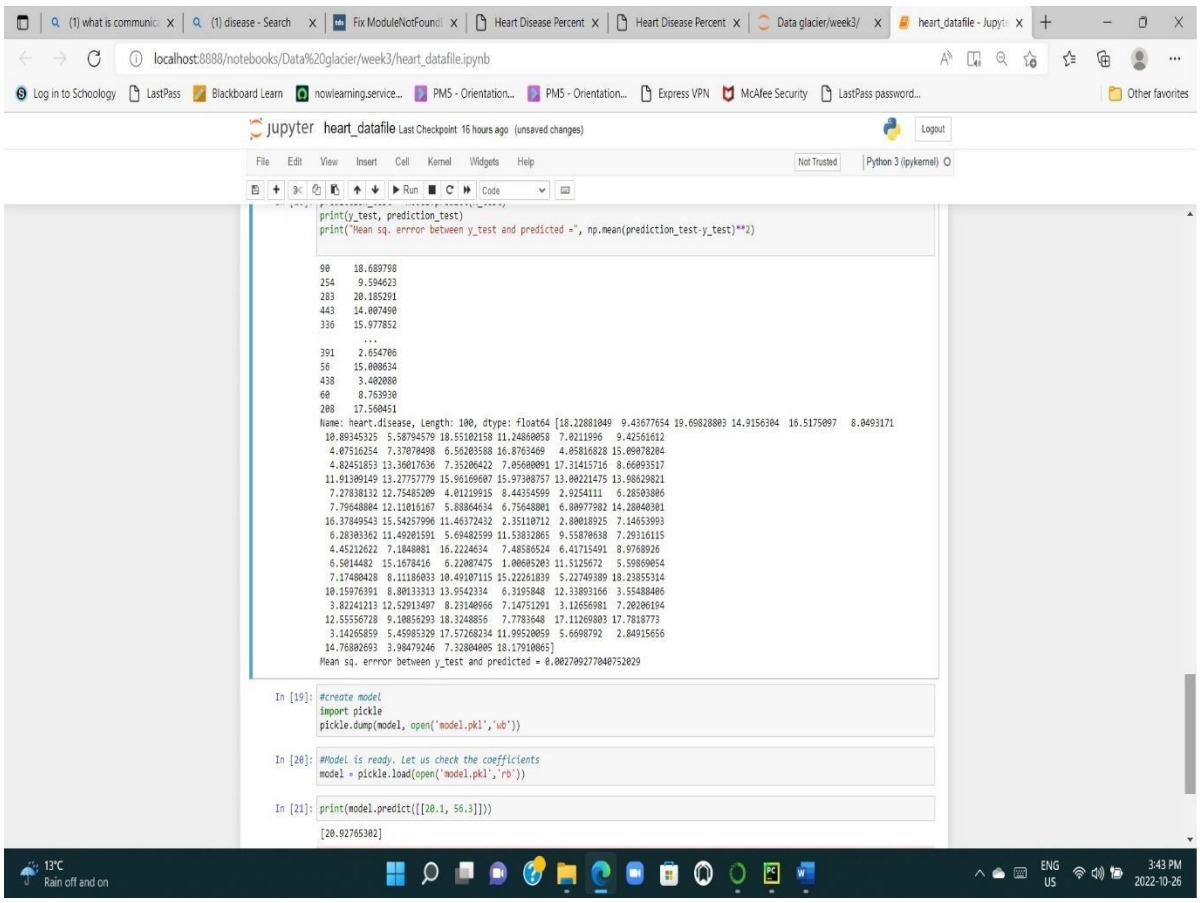
3.1.2 Build Model

After data preprocessing, we implement machine learning model to predict the heart disease value in the dataset. For this purpose, we import sklearn. After importing and initializing linear regression we fit into the training dataset. Also, we also calculate r square value and mean square error value.



3.1.3 Save the Model

After that we save our model using pickle



4.Turning Model into Web Application

We develop a web application that consists of a simple web page with a form field with entering the values and predict. After submitting the message to the web application, it will predict the heart disease rate. First, we create a folder for this project called python project 4 , this is the directory tree inside the folder. We will explain each file.

Table 3.1: Application Folder File Directory

app.py	
Templates	**Index.html**
Model	**model.pkl**
Dataset/	**Heart_disease.csv**

3.1 App.py

The *app.py* file contains the main code that will be executed by the Python interpreter to run the Flask web application, it included the ML code for classifying SD.

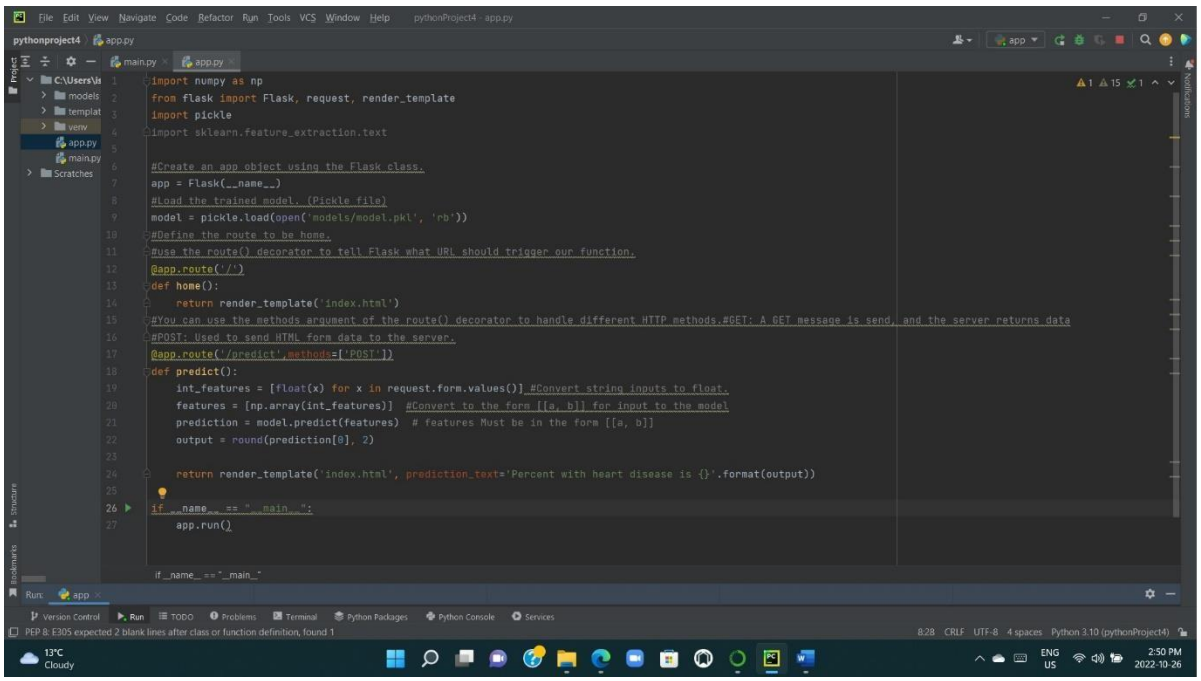


Figure 3.1: App.py

- We ran our application as a single module; thus, we initialized a new Flask instance with the argument *__name* to let Flask know that it can find the HTML template folder (*templates*) in the same directory where it is located.

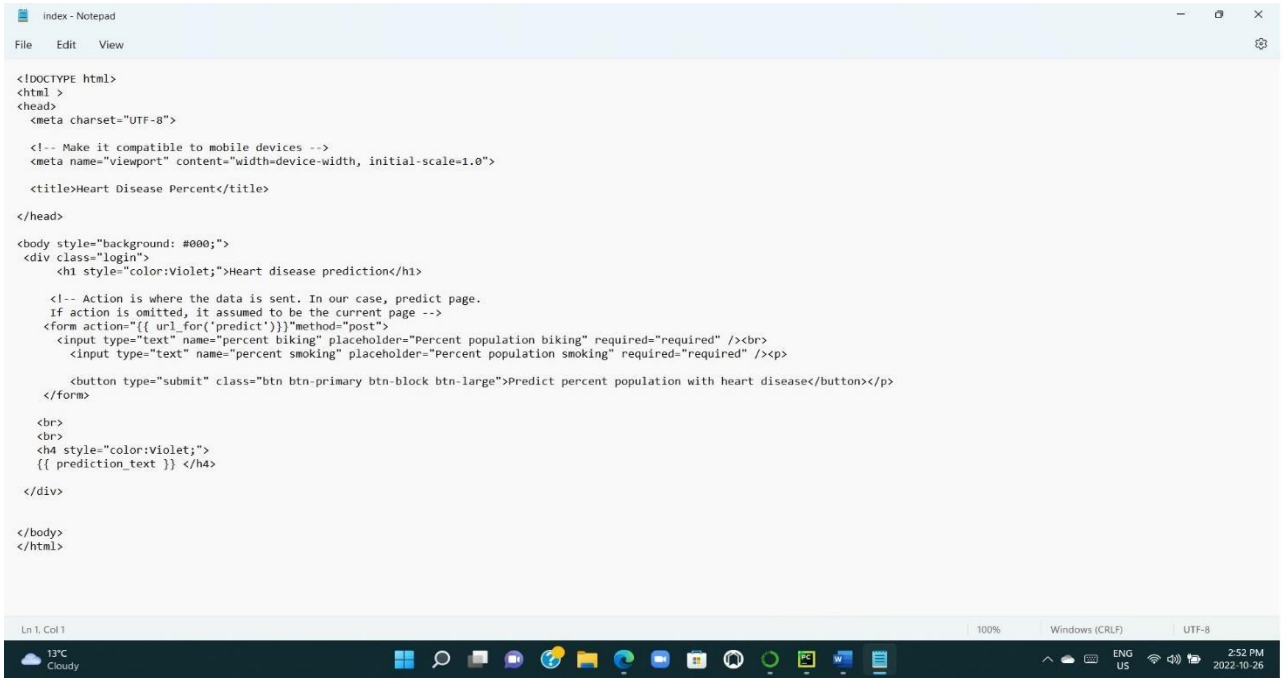
- Our *home* function simply rendered the *index.html* HTML file, which is located in the *templates* folder.

- Inside the *predict* function, we access the heart disease data set, pre-process the text, and make predictions, then store the model. We access the new message entered by the user and use our model to make a prediction for its label.
- we used the *POST* method to transport the form data to the server in the message body.
- Lastly, we used the *run* function to only run the application on the server when this script is directly executed by the

Python interpreter, which we ensured using the *if* statement with *__name__ == '__main__'*.

- **3.2 Index.html**

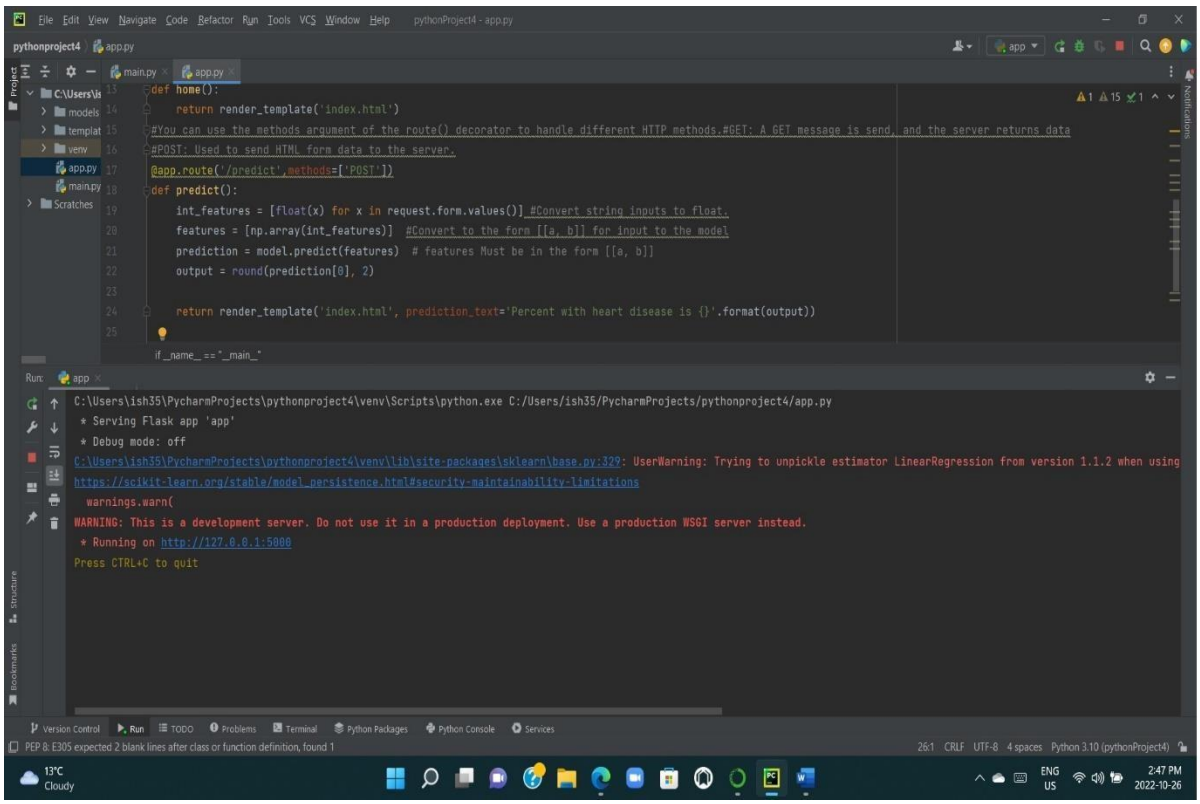
The following are the contents of the *Index.html* file that will enter the values to predict the flowers.



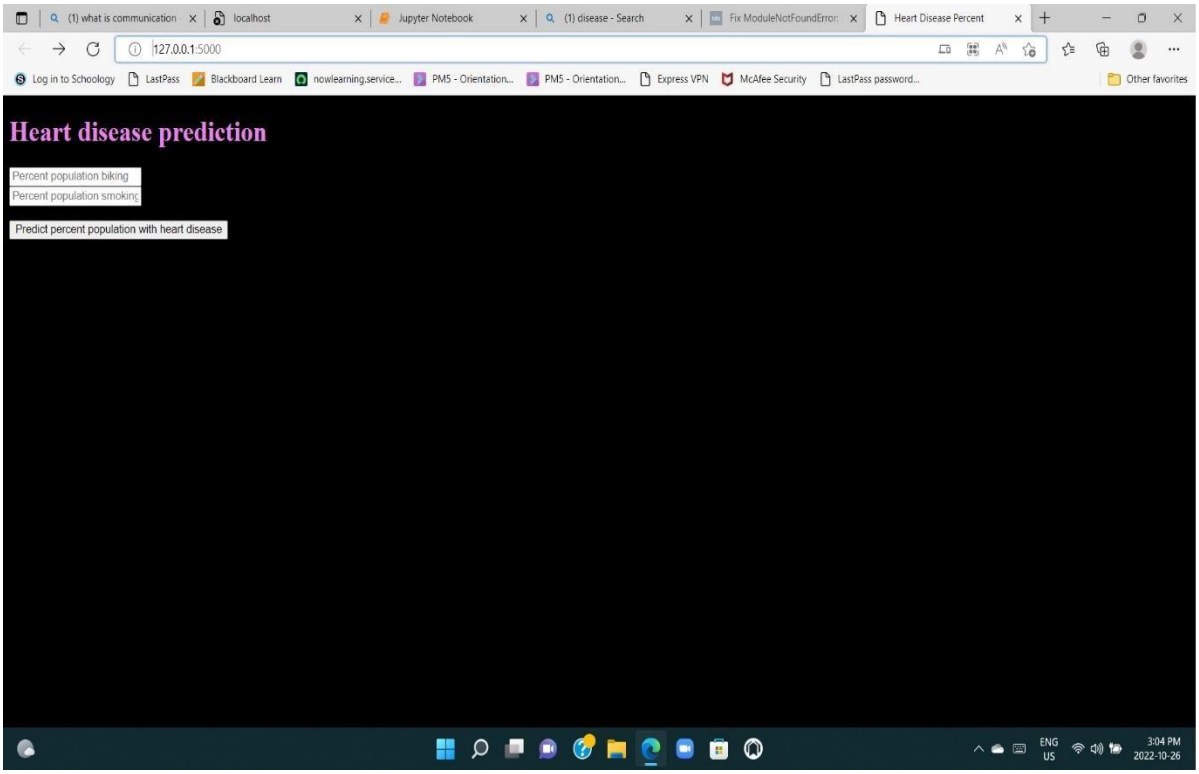
```
<!DOCTYPE html>
<html >
<head>
  <meta charset="UTF-8">

  <!-- Make it compatible to mobile devices -->
  <meta name="viewport" content="width=device-width, initial-scale=1.0">

  <title>Heart Disease Percent</title>

</head>

<body style="background: #000;">
 <div class="login">
      <h1 style="color:Violet;">Heart disease prediction</h1>

     <!-- Action is where the data is sent. In our case, predict page.
     If action is omitted, it assumed to be the current page -->
    <form action="{{ url_for('predict')}}"method="post">
      <input type="text" name="percent biking" placeholder="Percent population biking" required="required" /><br>
        <input type="text" name="percent smoking" placeholder="Percent population smoking" required="required" /><p>

        <button type="submit" class="btn btn-primary btn-block btn-large">Predict percent population with heart disease</button></p>
    </form>

   <br>
   <br>
   <h4 style="color:Violet;">
   {{ prediction_text }} </h4>

 </div>


</body>
</html>
```

- **4.1.2 Running Procedure**

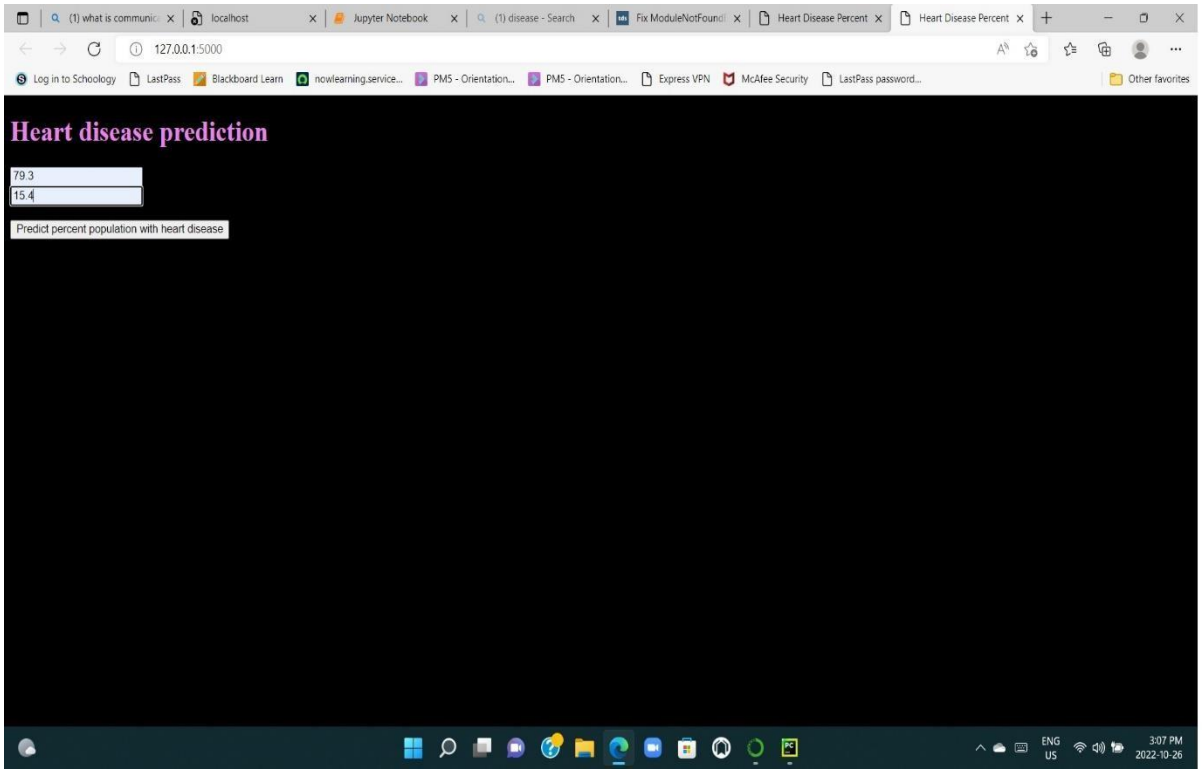
Once we have done all of the above, we can start running the API by clicking run button in *app.py* screen:

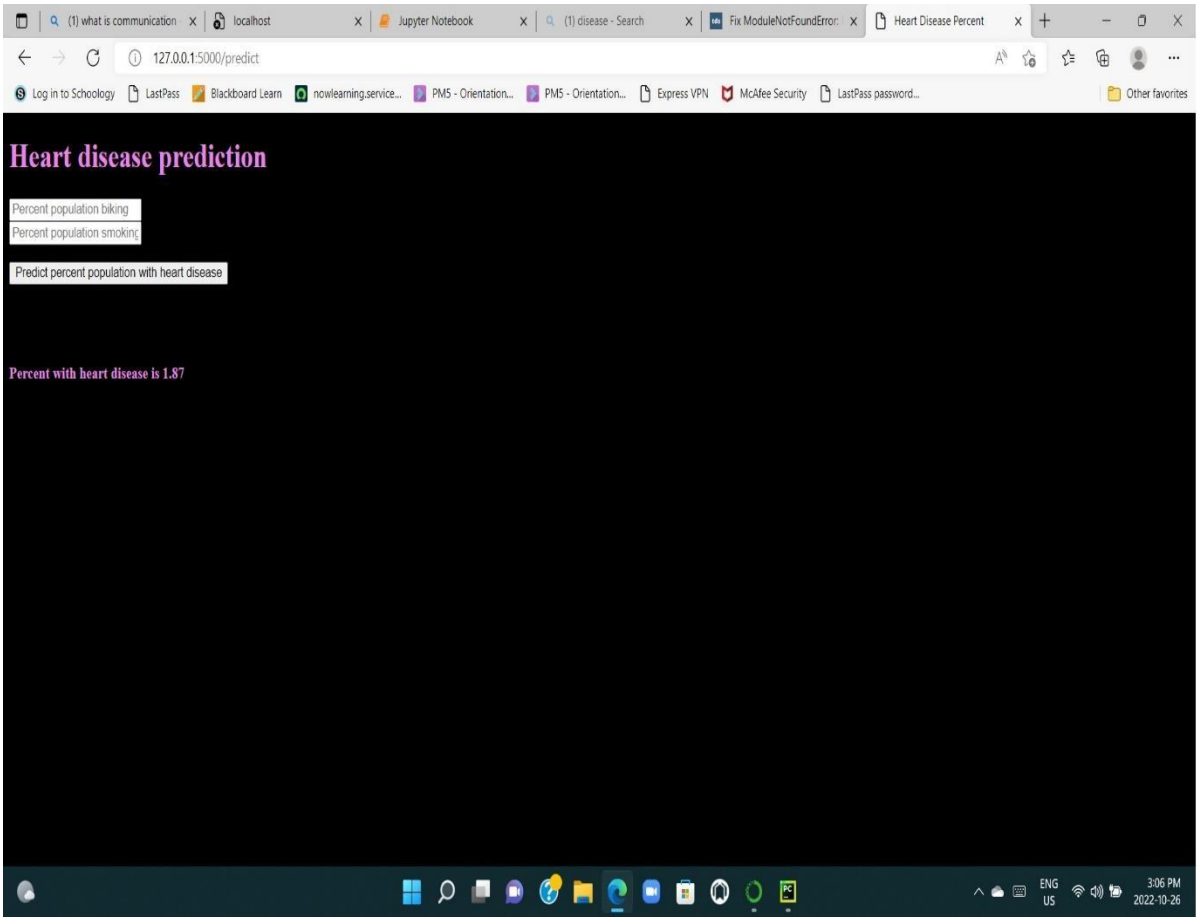


Now we could open a web browser and navigate to http://127.0.0.1:5000, we should see a simple website with the content like so



Now we enter input in the values in textbox and the value is predicted.




(1) what is communication
localhost
Jupyter Notebook
(1) disease - Search
Fix ModuleNotFoundError:
Heart Disease Percent
127.0.0.1:5000/predict
Log in to Schoology
LastPass
Blackboard Learn
nowlearning.service...
PM5 - Orientation...
PM5 - Orientation...
Express VPN
McAfee Security
LastPass password...
Other favorites
Heart disease prediction
Percent population biking
Percent population smoking
Predict percent population with heart disease
Percent with heart disease is 1.87
ENG
US
3:06 PM
2022-10-26

5.Model deployment using Heroku

We're ready to start our Heroku deployment now that our model has been trained, the machine learning pipeline has been set up, and the application has been tested locally. There are a few ways to upload the application source code onto Heroku. The easiest way is to link a GitHub repository to your Heroku account.

Requirement.txt

It is a text file containing the flask, NumPy, request, pickles, kit-learn library required to execute the application.

Runtime.txt

It is a text file containing the python packages required to execute the application.

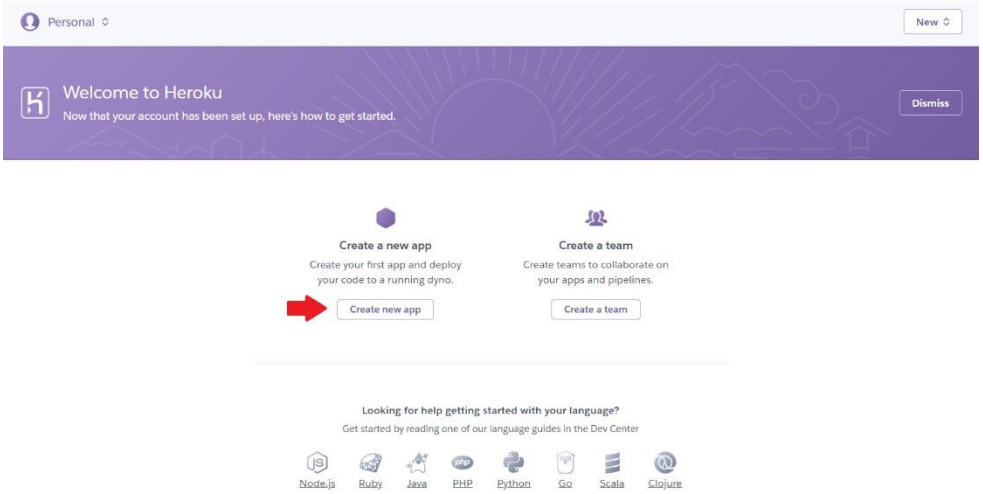
Procfile

Procfile is a file that specifies the commands that are executed by an Heroku app on Startup.

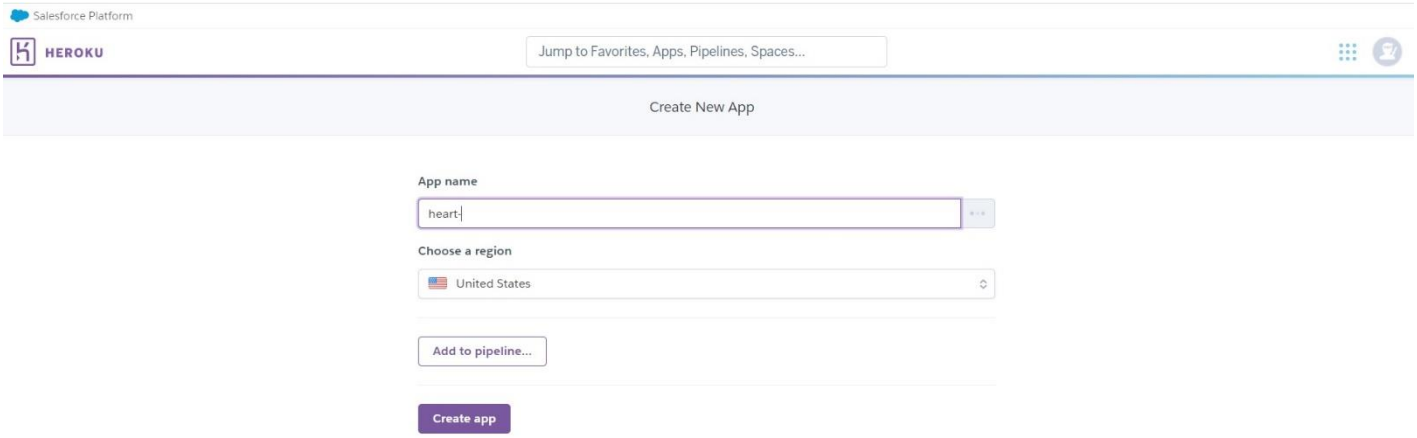
5.1Steps for Model Deployment Using Heroku

Once we uploaded files to the GitHub repository, we are now ready to start deployment on Heroku. Follow the steps below:

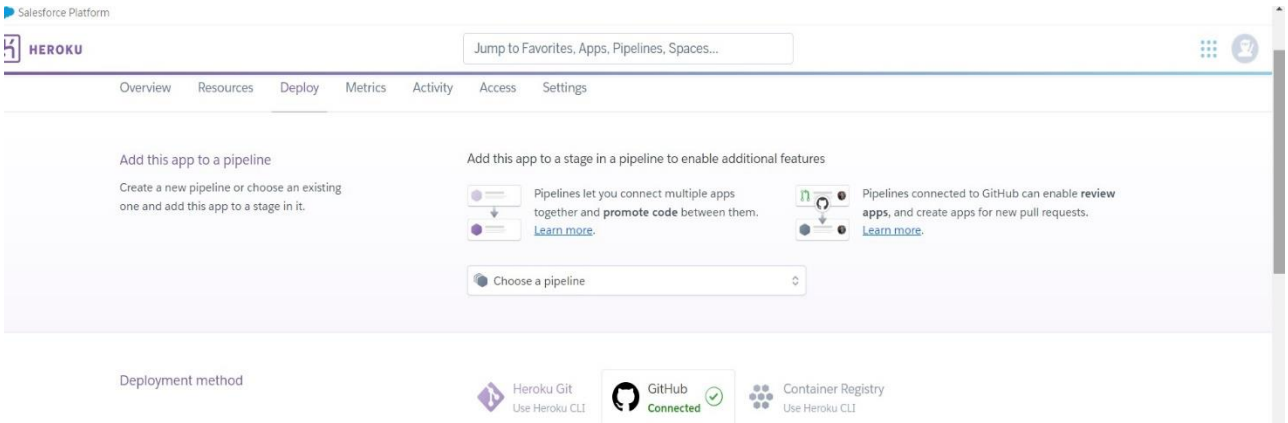
1. After sign up on **heroku.com** then click on **Create new app.**



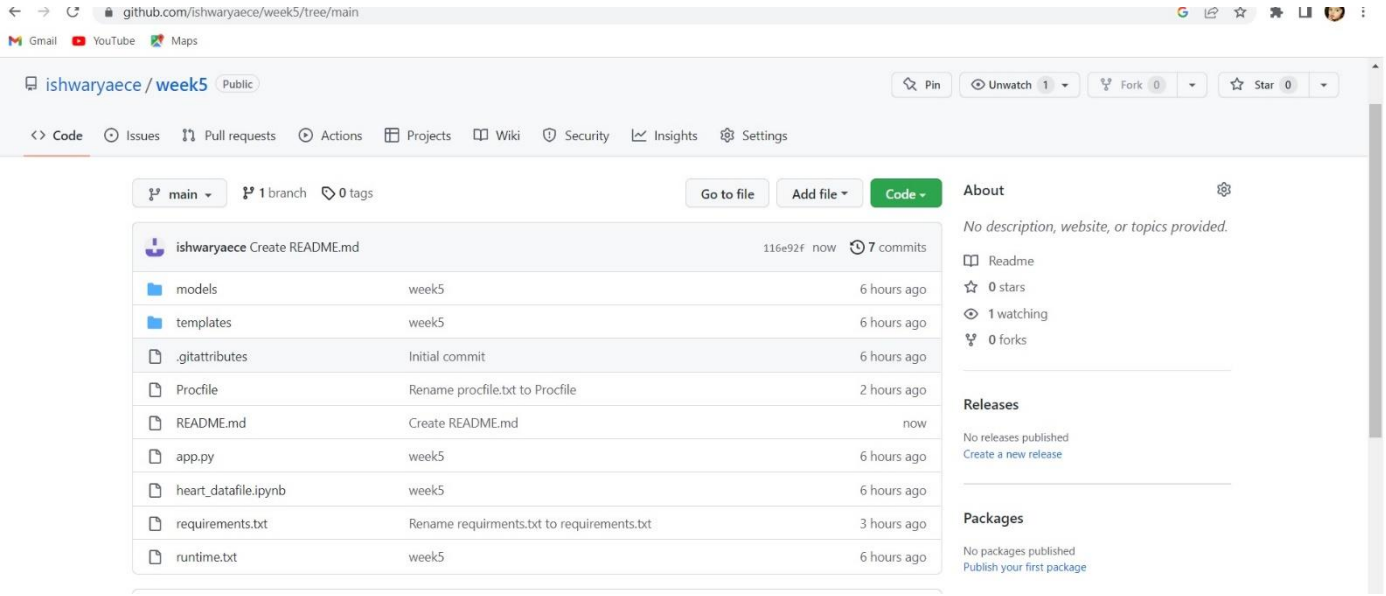
2.Enter **App name and region**



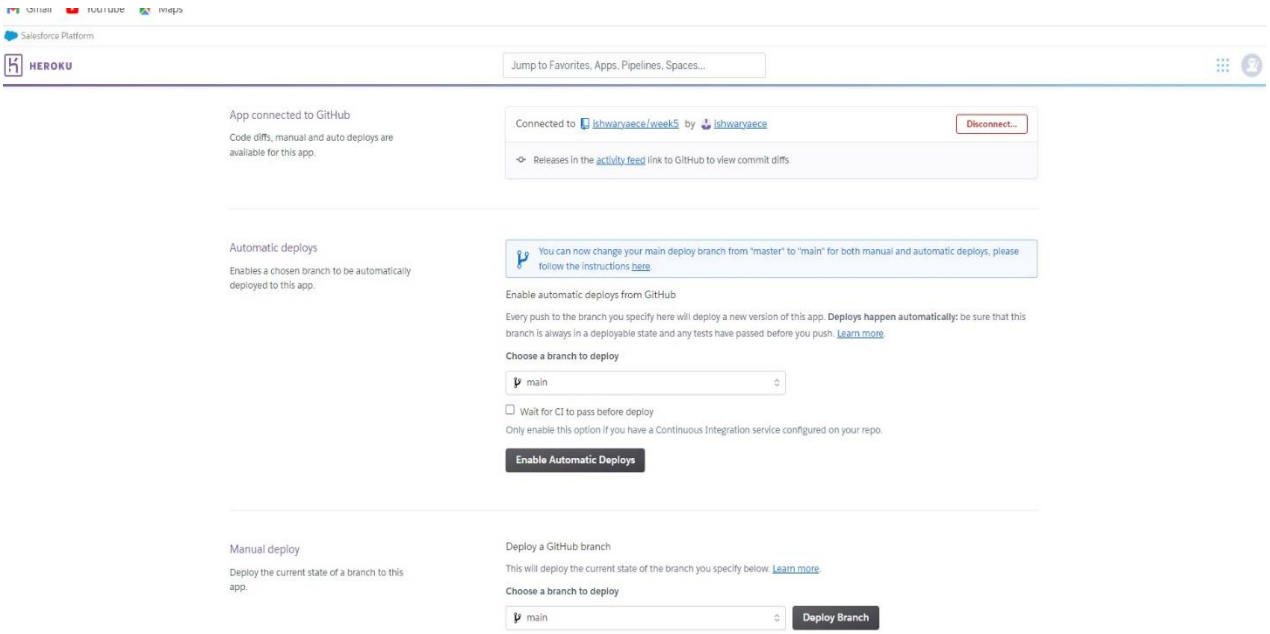
2. Connect to GitHub repository where code is I uploaded.



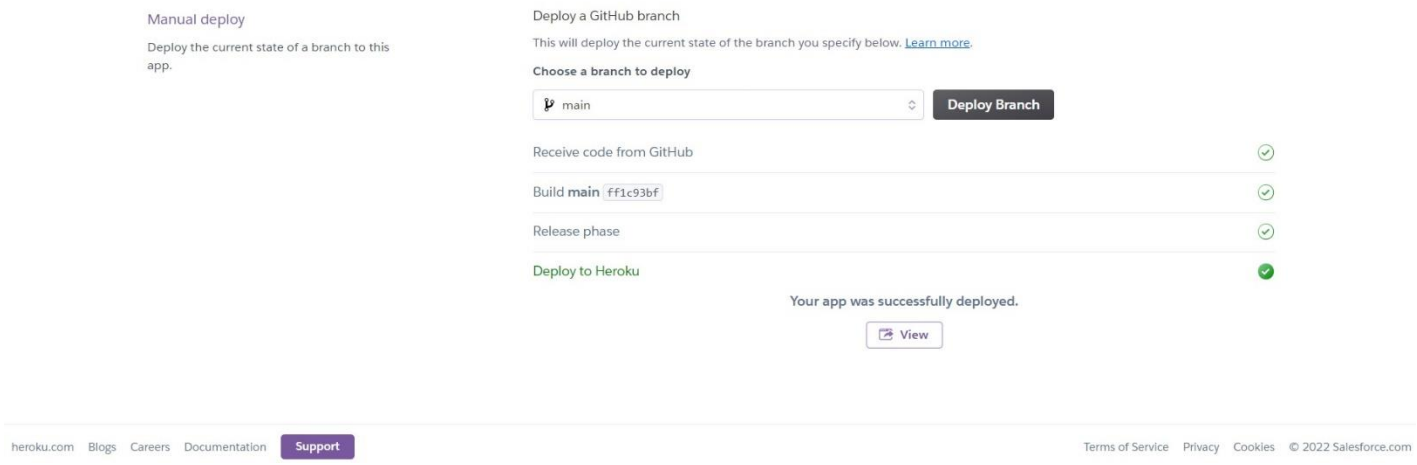
After that I choose the repository where I upload the code.



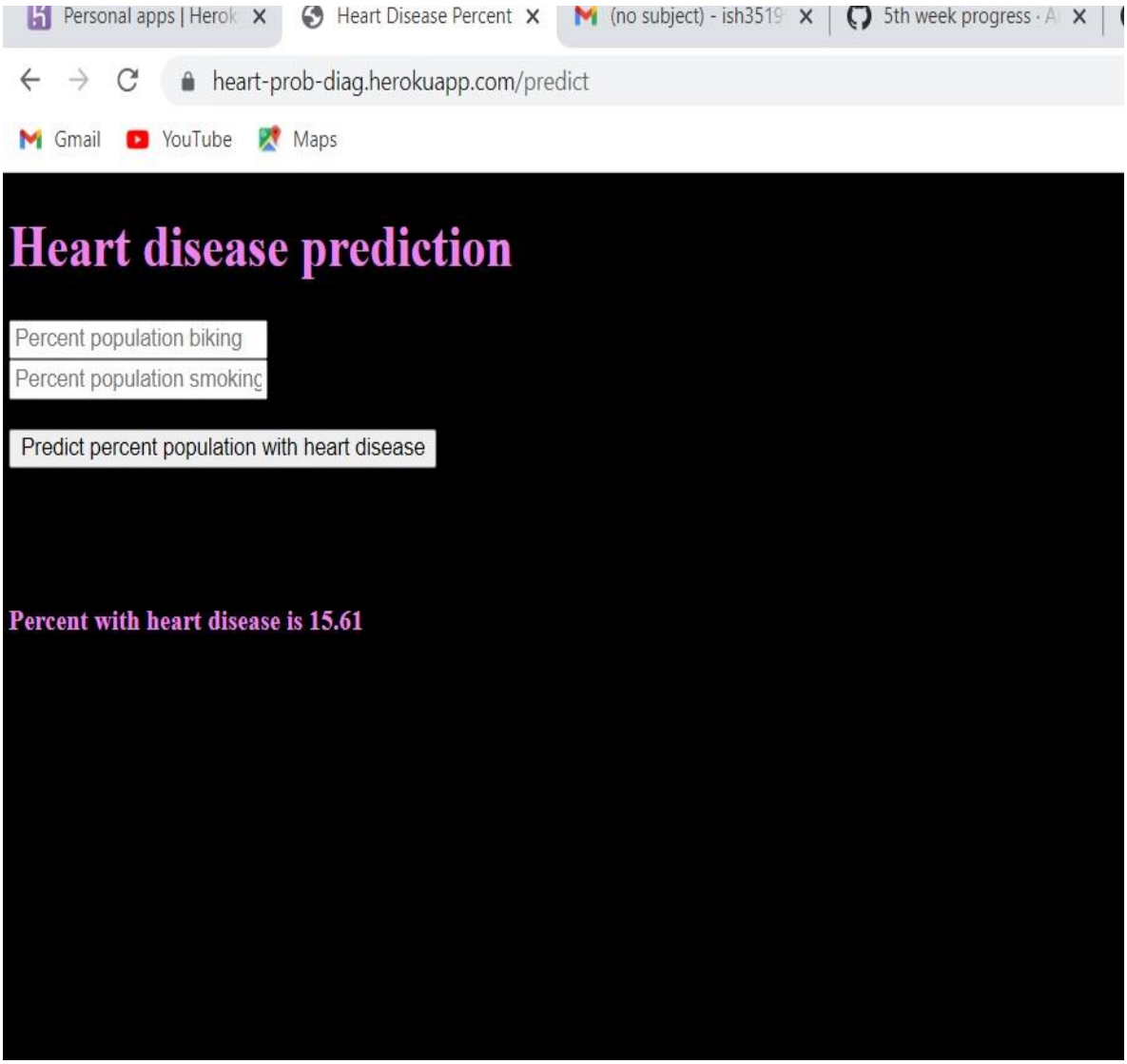
Deploy branch



After waiting 5 to 15 minutes our application is Ready



Finally click view then we get the expected result



The app is published at https://heart-prob-diag.herokuapp.com/predict