Week 5: Deployment on Flask and Heroku(PAAS)

Name: **Saanjanna Yuvaraj**

Batch Code: **LISUM23:30**

Submission Date: **05/08/2023**

Submitted to: **Data Glacier**

Building a Model
Step1 :

Import Required Libraries and Dataset

In this part, we import libraries and dataset pre-processing the dataset, Training the model which is shown below.

```
"""
Created on Fri Jul 28 08:59:59 2023

@author: sanjukarthick
"""

import pandas as pd
import seaborn as sns
import numpy as np
df= pd.read_csv("/Users/sanjukarthick/Desktop/heart_data1.csv")
print(df.head())

df = df.drop("Unnamed: 0", axis=1)
#A few plots in Seaborn to understand the data

sns.lmplot(x='biking', y='heart.disease', data=df)
sns.lmplot(x='smoking', y='heart.disease', data=df)


x_df = df.drop('heart.disease', axis=1)
y_df = df['heart.disease']

from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(x_df, y_df, test_size=0.3, random_state=42)

from sklearn import linear_model

#Create Linear Regression object
model = linear_model.LinearRegression()

#Now let us call fit method to train the model using independent variables.
#And the value that needs to be predicted (Images_Analyzed)

model.fit(X_train, y_train) #Indep variables, dep. variable to be predicted
print(model.score(X_train, y_train))  #Prints the R^2 value, a measure of how well


prediction_test = model.predict(X_test)
print(y_test, prediction_test)
print("Mean sq. errror between y_test and predicted =", np.mean(prediction_test-y_test)**2)
```

Step 2: Save the model

After that we save our model using pickle

```
import pickle
pickle.dump(model, open('model.pkl','wb'))

model = pickle.load(open('model.pkl','rb'))
print(model.predict([[20.1, 56.3]]))


#Model is ready. Let us check the coefficients, stored as reg.coef_.
#These are a, b, and c from our equation.
#Intercept is stored as reg.intercept_
#print(model.coef_, model.intercept_)

#All set to predict the number of images someone would analyze at a given time
#print(model.predict([[13, 2, 23]]))
```

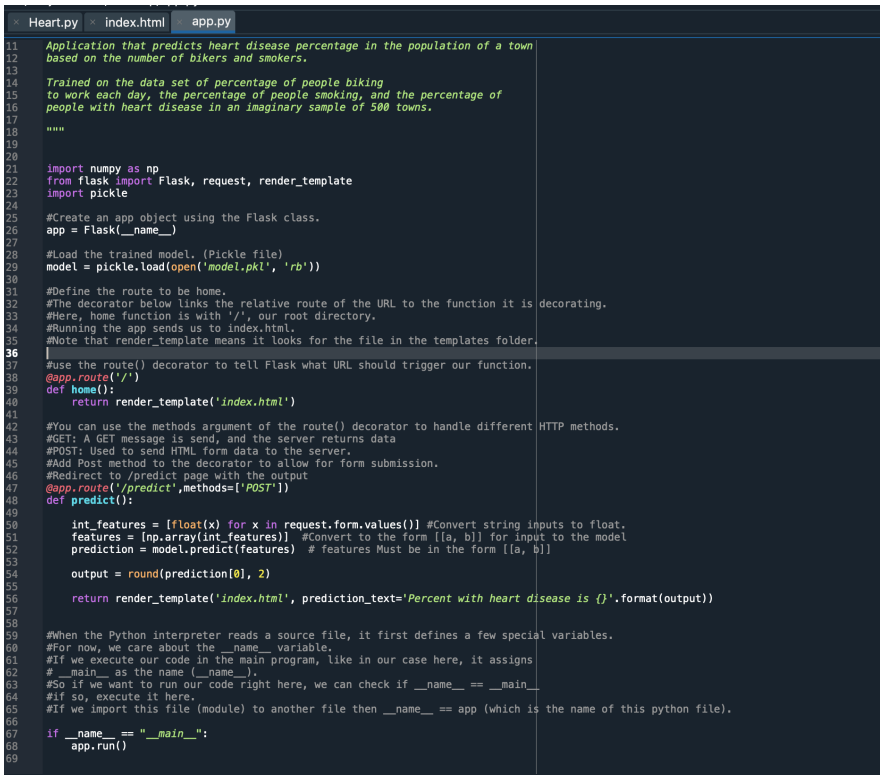
Step3: Turning the model in to web Application

We develop a web application that consists of a simple web page with a form field that lets us enter a message. After submitting the message to the web application, it will render it on a new page which gives us a result

Step 3.1: Writing Flask Application

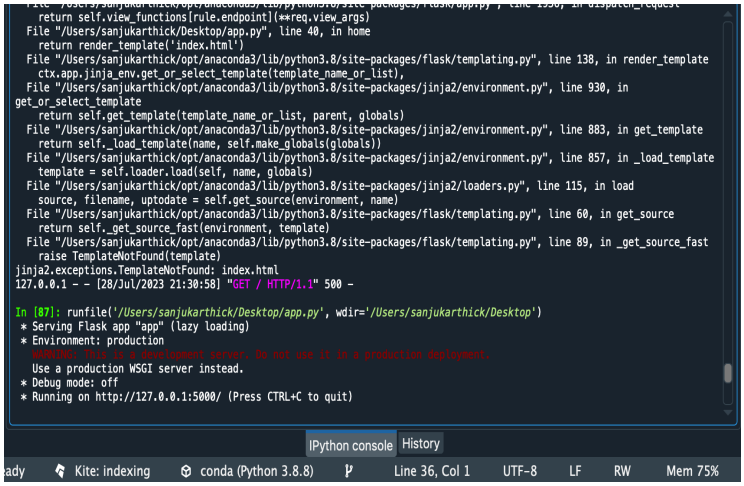
App.py

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



```
Heart.py  index.html  app.py

Application that predicts heart disease percentage in the population of a town
based on the number of bikers and smokers.

Trained on the data set of percentage of people biking
to work each day, the percentage of people smoking, and the percentage of
people with heart disease in an imaginary sample of 500 towns.

"""


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

#Create an app object using the Flask class.
app = Flask(__name__)

#Load the trained model. (Pickle file)
model = pickle.load(open('model.pkl', 'rb'))

#Define the route to be home.
#The decorator below links the relative route of the URL to the function it is decorating.
#Here, home function is with '/', our root directory.
#Running the app sends us to index.html.
#Note that render_template means it looks for the file in the templates folder.

#use the route() decorator to tell Flask what URL should trigger our function.
@app.route('/')
def home():
    return render_template('index.html')

#You can use the methods argument of the route() decorator to handle different HTTP methods.
#GET: A GET message is send, and the server returns data
#POST: Used to send HTML form data to the server.
#Add Post method to the decorator to allow for form submission.
#Redirect to /predict page with the output
@app.route('/predict',methods=['POST'])
def predict():

    int_features = [float(x) for x in request.form.values()] #Convert string inputs to float.
    features = [np.array(int_features)]  #Convert to the form [[a, b]] for input to the model
    prediction = model.predict(features)  # features Must be in the form [[a, b]]

    output = round(prediction[0], 2)

    return render_template('index.html', prediction_text='Percent with heart disease is {}'.format(output))


#When the Python interpreter reads a source file, it first defines a few special variables.
#For now, we care about the __name__ variable.
#If we execute our code in the main program, like in our case here, it assigns
# __main__ as the name (__name__).
#So if we want to run our code right here, we can check if __name__ == __main__
#if so, execute it here.
#If we import this file (module) to another file then __name__ == app (which is the name of this python file).

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

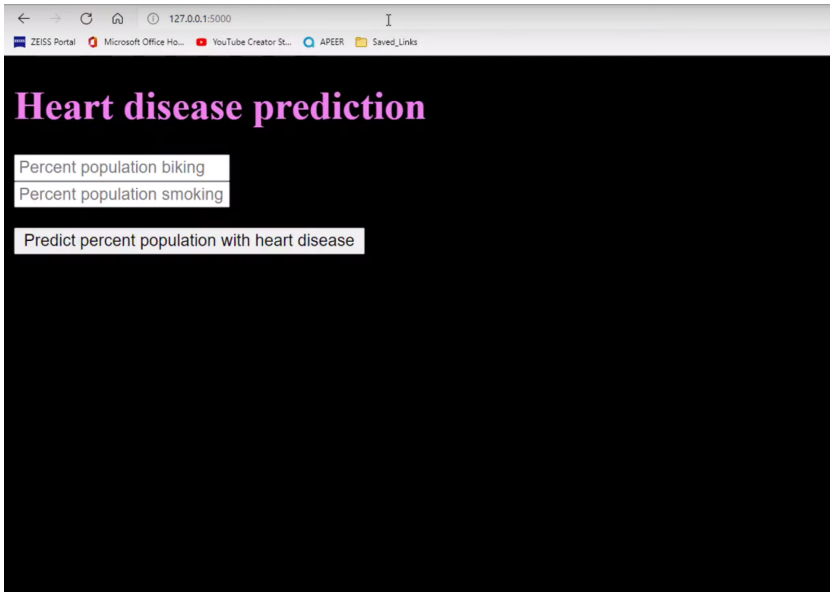
Step 3.2: Running the Flask Application



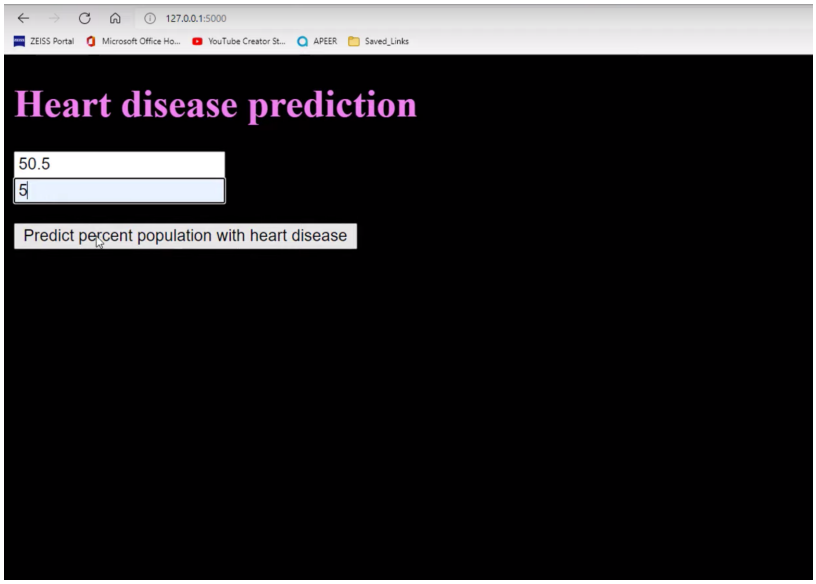
```
    return self.view_functions[rule.endpoint](**req.view_args)
  File "/Users/sanjukarthick/Desktop/app.py", line 40, in home
    return render_template('index.html')
  File "/Users/sanjukarthick/opt/anaconda3/lib/python3.8/site-packages/flask/templating.py", line 138, in render_template
    ctx.app.jinja_env.get_or_select_template(template_name_or_list),
  File "/Users/sanjukarthick/opt/anaconda3/lib/python3.8/site-packages/jinja2/environment.py", line 930, in
get_or_select_template
    return self.get_template(template_name_or_list, parent, globals)
  File "/Users/sanjukarthick/opt/anaconda3/lib/python3.8/site-packages/jinja2/environment.py", line 883, in get_template
    return self._load_template(name, self.make_globals(globals))
  File "/Users/sanjukarthick/opt/anaconda3/lib/python3.8/site-packages/jinja2/environment.py", line 857, in _load_template
    template = self.loader.load(self, name, globals)
  File "/Users/sanjukarthick/opt/anaconda3/lib/python3.8/site-packages/jinja2/loaders.py", line 115, in load
    source, filename, uptodate = self.get_source(environment, name)
  File "/Users/sanjukarthick/opt/anaconda3/lib/python3.8/site-packages/flask/templating.py", line 60, in get_source
    return self._get_source_fast(environment, template)
  File "/Users/sanjukarthick/opt/anaconda3/lib/python3.8/site-packages/flask/templating.py", line 89, in _get_source_fast
    raise TemplateNotFound(template)
jinja2.exceptions.TemplateNotFound: index.html
127.0.0.1 - - [28/Jul/2023 21:30:58] "GET / HTTP/1.1" 500 -

In [87]: runfile('/Users/sanjukarthick/Desktop/app.py', wdir='/Users/sanjukarthick/Desktop')
 * Serving Flask app "app" (lazy loading)
 * Environment: production
   WARNING: This is a development server. Do not use it in a production deployment.
   Use a production WSGI server instead.
 * Debug mode: off
 * Running on http://127.0.0.1:5000/ (Press CTRL+C to quit)

IPython console  History

ady  Kite: indexing  conda (Python 3.8.8)  Line 36, Col 1  UTF-8  LF  RW  Mem 75%
```

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

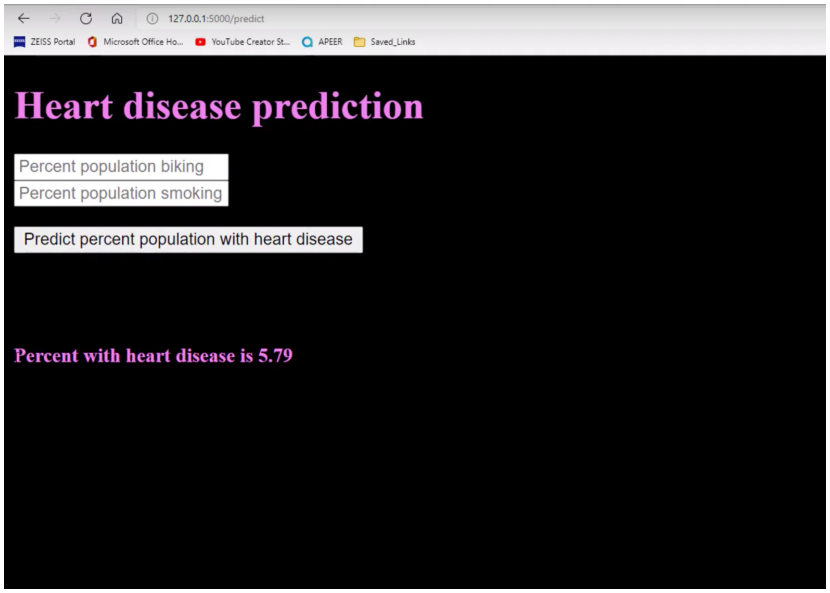
Step 3.3: Open the link in the browser



Step 3.4: Testing the model



Step 3.5: Getting the result



4.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.

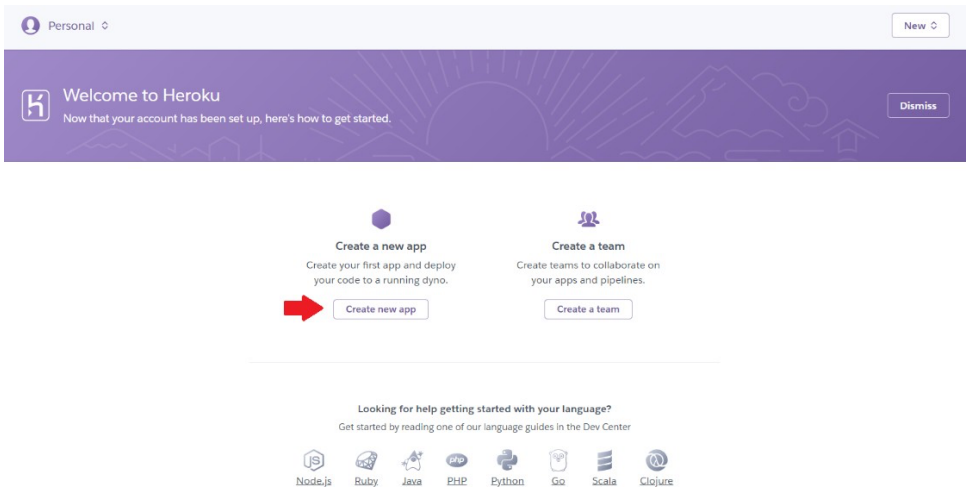
Requirement.txt

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

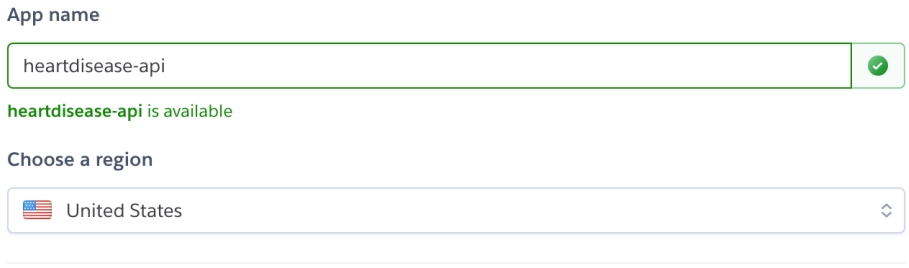
Steps for Model Deployment Using Heroku

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.**

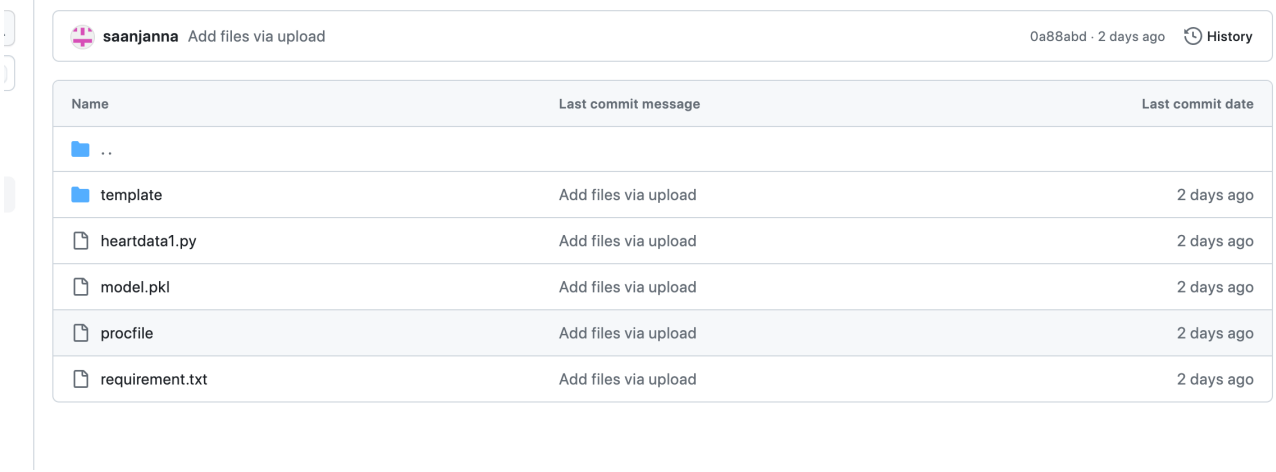


2.Enter App Name and Region

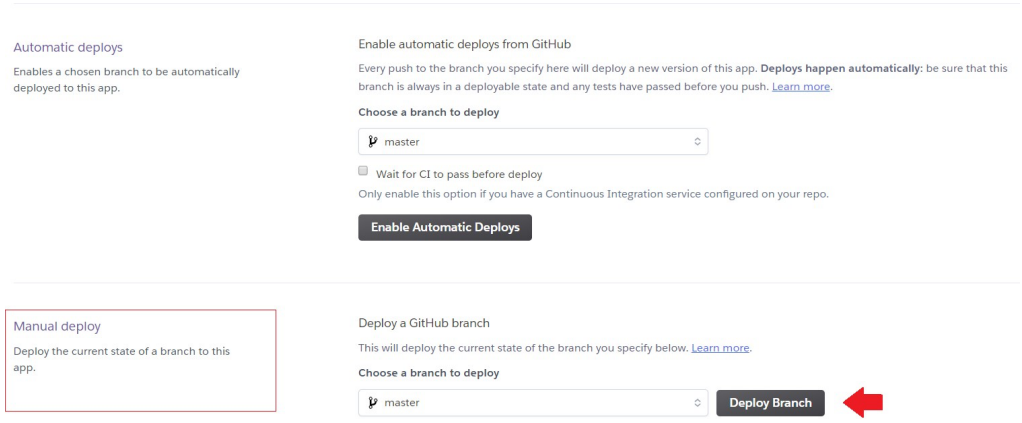


3. Connect to GitHub repository where code is I uploaded.

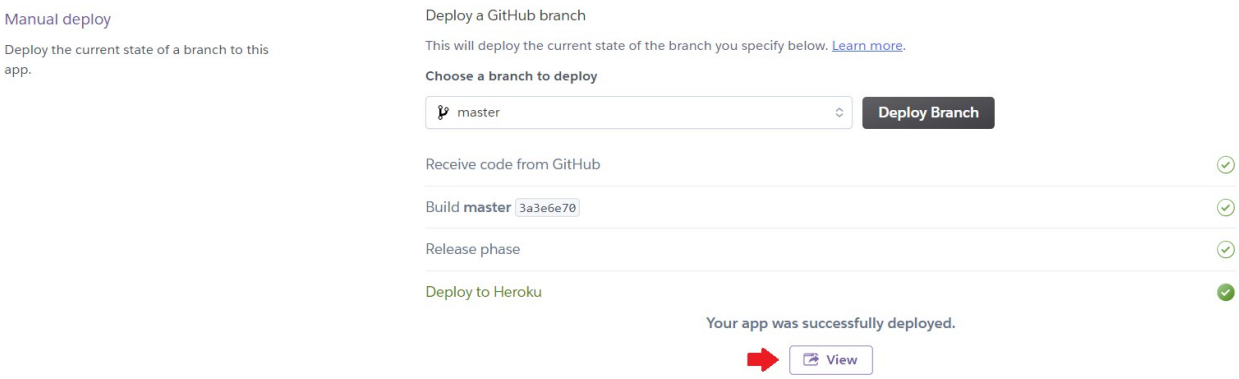
After that I choose the repository where I upload the code.



4.Deploy Branch:



5. After Few minutes the app is successfully deployed



The app is published at

https:// heartdisease-api.herokuapp.com/