

Elements of AIML Assignment 2.1

SUBMITTED BY :

HARSHIL MITTAL
R2142230326
500121907
B-11

SUBMITTED TO:

MR. CHANDRA MANI SHARMA
ASSISTANT PROFESSOR
(SENIOR SCALE)

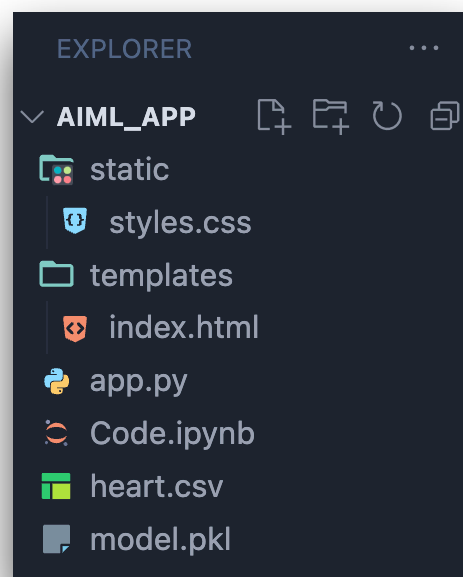
AIM : Develop a Flask-based UI to use the ML model developed in the Assignment – 1.

GitHub Repository for Assignment 1 :

<https://github.com/Harshil-Mittal/Assignment-1>

Reference : In Assignment - 1, I trained a Machine learning model for the Heart Disease Prediction based on the SDG 3 among the 17 SDG. In Assignment 2.2, I have developed a Flask UI based app using Flask, Python, HTML, CSS and the trained model.

STEP 1 : Creating the folder structure for the App.



Here:

- Static : Contains the CSS file for the designing and formatting.
- Templates : Contains the HTML file for UI.
- APP.PY : The flask app code.

- Code.ipynb : Python code for training the machine learning model
- Heart.csv : The dataset used to train the model
- model.pkl : The trained model used in the flask code.

Step 2 : Training the ML Model

The ML model was trained in the Assignment-1.

Refer the Explanation_Doc present in the GitHub repository for more details.

Step 3 : Saving the Trained Model as “model.pkl”



```
import joblib
# Assuming your model variable is `model`
joblib.dump(model, 'model.pkl')

[10]

... ['model.pkl']
```

Added this part to the existing python notebook file to export the trained model into a pkl file, which can be used in the flask app.

Step 4 : Creating app.py :

```
app.py > ...
1  from flask import Flask, request, render_template
2  import joblib
3  import numpy as np
4
5  app = Flask(__name__)
6
7  # Load the pre-trained model
8  model = joblib.load('model.pkl')
9
10 @app.route('/')
11 def index():
12     return render_template('index.html')
13
14 @app.route('/predict', methods=['POST'])
15 def predict():
16     try:
17         # Collect all input features from the form
18         features = [
19             float(request.form['age']),
20             float(request.form['sex']),
21             float(request.form['cp']),
22             float(request.form['trestbps']),
23             float(request.form['cho1']),
24             float(request.form['fbs']),
25             float(request.form['restecg']),
26             float(request.form['thalach']),
27             float(request.form['exang']),
28             float(request.form['oldpeak']),
29             float(request.form['slope']),
30             float(request.form['ca']),
31             float(request.form['thal'])
32         ]
33
34         # Convert input to a NumPy array with the required shape
35         input_data = np.array([features])
36
37         # Make prediction
38         prediction = model.predict(input_data)
39
40         # Interpret result
41         result = "Heart Disease Detected" if prediction[0] == 1 else "No Heart Disease"
42
43     except ValueError:
44         # Error message for invalid inputs
45         result = "Please enter valid numerical values for all fields."
46
47     return render_template('index.html', prediction_text=result)
48
49 if __name__ == "__main__":
50     app.run(debug=True)
```

This Flask application provides a simple web interface for predicting heart disease based on user input. The app allows users to enter key health parameters such as age, resting blood pressure, cholesterol levels, and other relevant metrics. It uses a pre-trained machine learning model (saved as `model.pkl`) to make predictions.

How It Works:

- The user enters their data into the form and submits it.
- The app retrieves the input, processes it into the correct format for the model, and makes a prediction.
- The prediction result is displayed dynamically on the same page.

Step 5 : Defining the Web Layout and Design.

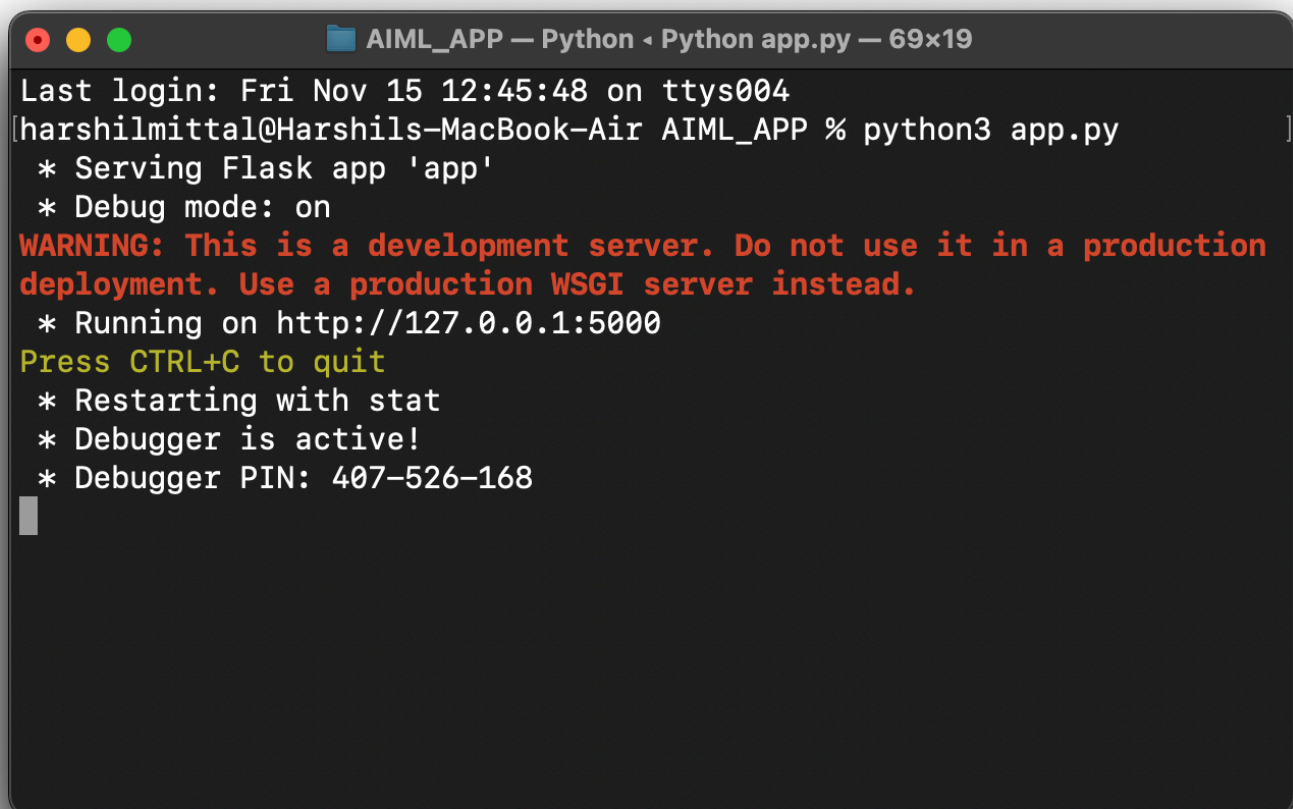
• **Index.html** : The `index.html` file is the main HTML template for the Flask application. It provides a simple form where users can enter their health details (e.g., age, blood pressure, cholesterol) to check for heart disease risk. Upon submission, the form sends data to the server, which processes it and displays the prediction result (“Heart Disease Detected” or “No Heart Disease”) on the same page.

• **Styles.css** : The `style.css` file styles the app’s webpage, making it look clean and easy to read. It centers the form, adds light colors, and makes the input fields and buttons user-friendly.

Step 6 : Running the app.

To run this app make sure Flask is installed on your current working environment.

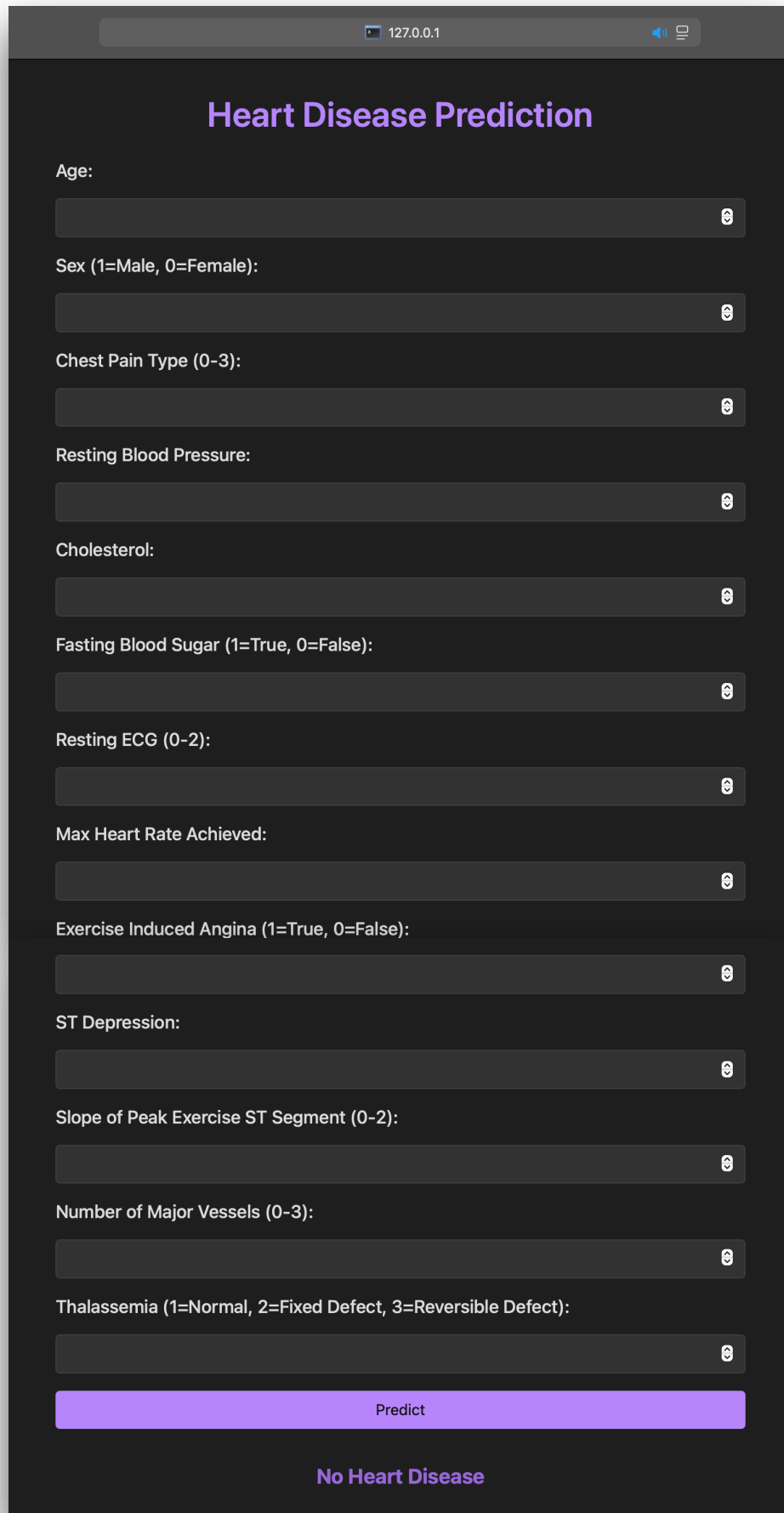
Open the directory in the terminal where all the files are stored and run the command “`python3 app.py`”.

A terminal window titled "AIML_APP — Python < Python app.py — 69x19" is shown. The window has a dark background with light-colored text. The output of the command "python3 app.py" is displayed. It starts with "Last login: Fri Nov 15 12:45:48 on ttys004", followed by the prompt "[harshilmittal@Harshils-MacBook-Air AIML_APP % python3 app.py]". The output then shows several status messages: "* Serving Flask app 'app'", "* Debug mode: on", a red warning message "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" in yellow, "* Restarting with stat", "* Debugger is active!", and "* Debugger PIN: 407-526-168". A cursor is visible at the end of the last line.

```
AIML_APP — Python < Python app.py — 69x19
Last login: Fri Nov 15 12:45:48 on ttys004
[harshilmittal@Harshils-MacBook-Air AIML_APP % python3 app.py]
* Serving Flask app 'app'
* 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: 407-526-168
```

Final Output :

Go to <http://127.0.0.1:5000> to view to app.



127.0.0.1

Heart Disease Prediction

Age:

Sex (1=Male, 0=Female):

Chest Pain Type (0-3):

Resting Blood Pressure:

Cholesterol:

Fasting Blood Sugar (1=True, 0=False):

Resting ECG (0-2):

Max Heart Rate Achieved:

Exercise Induced Angina (1=True, 0=False):

ST Depression:

Slope of Peak Exercise ST Segment (0-2):

Number of Major Vessels (0-3):

Thalassemia (1=Normal, 2=Fixed Defect, 3=Reversible Defect):

Predict

No Heart Disease

We used multiple test cases to test this model and got really accurate results. The final web output of this heart disease prediction app displays a simple interface where users can enter health-related data, such as age, cholesterol levels, and blood pressure. After submitting the information, the app instantly provides a prediction result on the same page, indicating whether the user is at risk for heart disease.

References :

1. UCI Heart Disease Dataset: This dataset, available from the UCI Machine Learning Repository, was used to train the machine learning model for heart disease prediction. It contains various health attributes, including age, sex, cholesterol levels, and more, to predict the likelihood of heart disease.

2. AIML (Artificial Intelligence and Machine Learning)

Techniques: The project utilized machine learning algorithms such as Logistic Regression, Decision Tree, and Support Vector Machine (SVM) for classification and prediction tasks. The techniques were implemented using Python libraries such as scikit-learn.

3. Udemy Course: “The 2024 WebDev Bootcamp”: This course provided me with essential skills in HTML and CSS, which were applied to create the user interface for the Flask-based web application in the project.

THANK YOU