MACHINE LEARNING DEPLOYMENT USING IBM WATSON STUDIO

TEAM LEADER:

NAME:NIRUPAMA K

REG NO:211521243110

TEAM MEMBERS:

2.NAME:NEHA SHRUTHI.U

REG NO:211521243109

3. NAME:ASWATHI SWARNA SREE

REG NO:211521243025

4. NAME:PRIYADHARSHINI N

REG NO: 211521243121

MACHINE LEARNING DEPLOYMENT USING IBM WATSON STUDIO

MACHINE LEARNING DEPLOYMENT:

1. Collect and Clean Data: Gather data relevant to your model, which should ideally come from multiple sources to improve accuracy. Ensure the data is clean, with any inconsistencies or missing values addressed.

2. Develop a Machine Learning Model: Train your model using historical data, fine-tuning the parameters and structure of the model to achieve optimal performance. This process typically involves experimentation, where different approaches are tested to identify the most effective method.

3. Model Validation: Validate your model by comparing its predictions with actual outcomes, either from historical data or from a separate testing set. This process helps identify any overfitting, underfitting, or other potential issues with the model.

4. Model Testing: Thoroughly test your model by feeding it various scenarios and data points. Evaluate its performance by examining metrics such as accuracy, precision, recall, and F1-score. Additionally, ensure that the model can handle any potential edge cases or outliers in the data.

5. Model Documentation: Document your model, including its purpose, the machine learning algorithm used, and any specific requirements or limitations for deployment. This documentation is essential for anyone who may interact with the model, such as other data scientists or developers responsible for integrating the model into an application.

6. Deployment Planning: Determine the most suitable environment for deploying your machine learning model. Consider factors such as latency requirements, data privacy, and scalability.

7. Deployment Execution: Implement the deployment plan by deploying the trained machine learning model to the chosen environment. This process may involve using

specific tools or libraries, depending on the type of model and the deployment platform.

8. Monitoring and Maintenance: Continuously monitor the performance of your deployed machine learning model, updating the model and its documentation as necessary to maintain optimal performance. This process may also involve addressing any errors or exceptions that arise during the model's execution.

Overall, the deployment of a machine learning model involves multiple stages, from data collection and cleaning to monitoring and maintenance. By following a systematic approach, you can ensure that your model is deployed successfully and efficiently, ultimately contributing to the successful implementation of your machine learning project.</s>

To create an advanced website for detecting machine learning models and HTML code, we can follow these steps:

1. Use the latest front-end frameworks like React or Angular for better UI/UX.
2. Design a clean and intuitive interface with various UI components such as forms, buttons, dropdowns, and tables.
3. For machine learning model detection, use the following Python library and save it as a model file:

Model Deployment

It is time to start deploying and building the web application using Flask web application framework. For the web app, we have to create:

1. ***Web app python code (API)*** to load the model, get user input from the HTML template, make the prediction, and return the result.

4. 2. ***An HTML** template* for the front end to allow the user to input heart disease symptoms of the patient and display if the patient has heart disease or not.
5. The structure of the files is like the following:
6. /
7. ├── model.pkl
8. ├── heart_disease_app.py
9. ├── templates/
10. └── Heart Disease Classifier.html
11.
12. Web App Python Code
13. You can find the full code of the web app here.
14. As a first step, we have to import the necessary libraries.
15. import numpy as np
16. import pickle
17. from flask import Flask, request, render_template
18. Then, we create app object.
19. # Create application
20. app = Flask(__name__)
21. After that, we need to load the saved model model.pkl in the app.
22. # Load machine learning model
23. model = pickle.load(open('model.pkl', 'rb'))
24. After that home() function is called when the root endpoint '/' is hit. The function redirects to the home page Heart Disease Classifier.html of the website.
25. # Bind home function to URL
26. @app.route('/')
27. def home():
28.   return render_template('Heart Disease Classifier.html')
29. Now, create predict() function for the endpoint '/predict'. The function is defined as this endpoint with POST method. When the user submits the form, the API receives a POST request, the API extracts all data from the form using flask.request.form function. Then, the API uses the model to predict the result. Finally, the function renders the Heart Disease Classifier.html template and returns the result.
30. # Bind predict function to URL
31. @app.route('/predict', methods =['POST'])
32. def predict():
33.
34.   # Put all form entries values in a list
35.   features = [float(i) for i in request.form.values()]
36.   # Convert features to array
37.   array_features = [np.array(features)]
38.   # Predict features
39.   prediction = model.predict(array_features)
40.
41.   output = prediction
42.

43. # Check the output values and retrieve the result with html tag based on the value
44. if output == 1:
45. return render_template(‘Heart Disease Classifier.html’,
46. result = ‘The patient is not likely to have heart disease!’)
47. else:
48. return render_template(‘Heart Disease Classifier.html’,
49. result = ‘The patient is likely to have heart disease!’)
50. Finally, start the flask server and run our web page locally on the computer by calling app.run() and then enter http://localhost:5000 on the browser.
51. if __name__ == ‘__main__’:
52. #Run the application
53. app.run()
54.

Python:

```
import pickle

from sklearn.model_selection import train_test_split

from sklearn.ensemble import RandomForestClassifier

from sklearn.feature_extraction.text import CountVectorizer

from sklearn.metrics import accuracy_score

# assuming data is loaded in the 'df' variable

# separate features and labels

features = df['html_code']

labels = df['label']

# split the data into training and testing sets

features_train, features_test, labels_train, labels_test = train_test_split(features, labels,
test_size=0.2, random_state=42)

# convert the features into vectors

vectorizer = CountVectorizer()

features_train_vectorized = vectorizer.fit_transform(features_train)

features_test_vectorized = vectorizer.transform(features_test)
```

```
# create and train the model

model = RandomForestClassifier()

model.fit(features_train_vectorized, labels_train)

# evaluate the model

predictions = model.predict(features_test_vectorized)

accuracy = accuracy_score(labels_test, predictions)

print(f"Model accuracy: {accuracy}")

# save the model to disk

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

4. In the website's back-end, create a REST API to handle HTTP requests. For example, using Flask, you can create a POST request that sends the HTML code to the back-end, which then uses the machine learning model to detect the corresponding machine learning model.

```
from flask import Flask, request, jsonify

import pickle

app = Flask(__name__)

@app.route('/detect', methods=['POST'])

def detect_model():

    data = request.get_json()

    html_code = data['html_code

# load the saved model from disk

    model = pickle.load(open('model.pkl', 'rb'))
```

```
# convert the html code into vectors
    vectorizer = CountVectorizer()
    features_vectorized = vectorizer.transform([html_code])
 # use the model to predict the machine learning model
    prediction = model.predict(features_vectorized)
 # return the prediction as a JSON response
    return jsonify({'prediction': prediction[0]})


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

5. Implement a code editor on the front-end that allows users to input their HTML code. Use libraries like CodeMirror or Ace to create the code editor.
6. Style the website using CSS. Consider using CSS pre-processors like Sass or Less to make the styling process more efficient.
7. Implement advanced features such as real-time code detection, auto-formatting, and code snippets using libraries like PrismJS or Ace.
8. Add a testing environment to your back-end using libraries like Pytest. This will ensure the accuracy and reliability of your machine learning model.

CODE:

```
<html>
<head>
<!-- Bootstrap CSS -->
```

```
<link rel="stylesheet"
href="https://stackpath.bootstrapcdn.com/bootstrap/4.5.2/css/bootstrap.min.css"
integrity="sha384-
JcKb8q3iqJ61gNV9KGb8thSsNjpSL0n8PARn9HuZOnIxN0hoP+VmmDGMN5t9UJ0Z"
crossorigin="anonymous">

<script src="https://code.jquery.com/jquery-3.5.1.slim.min.js" integrity="sha384-
DfXdz2htPH0lsSSs5nCTpuj/zy4C+OGpamoFVy38MVBnE+IbbVYUew+OrCXaRkfj"
crossorigin="anonymous"></script>

<script src="https://cdn.jsdelivr.net/npm/popper.js@1.16.1/dist/umd/popper.min.js"
integrity="sha384-
9/reFTGAW83EW2RDu2S0VKaIzap3H66lZH81PoYlFhbGU+6BZp6G7niu735Sk7lN"
crossorigin="anonymous"></script>

<script src="https://stackpath.bootstrapcdn.com/bootstrap/4.5.2/js/bootstrap.min.js"
integrity="sha384-
B4gt1jrGC7Jh4AgTPSdUtOBvfO8shuf57BaghqFfPlYxofvL8/KUEfYiJOMMV+rV"
crossorigin="anonymous"></script>

<title>Heart Disease Test</title>

</head>

<body>

<!-- Java Script -->

<script src="https://code.jquery.com/jquery-3.5.1.slim.min.js" integrity="sha384-
DfXdz2htPH0lsSSs5nCTpuj/zy4C+OGpamoFVy38MVBnE+IbbVYUew+OrCXaRkfj"
crossorigin="anonymous"></script>

<script src="https://cdn.jsdelivr.net/npm/popper.js@1.16.1/dist/umd/popper.min.js"
integrity="sha384-
9/reFTGAW83EW2RDu2S0VKaIzap3H66lZH81PoYlFhbGU+6BZp6G7niu735Sk7lN"
crossorigin="anonymous"></script>

<script src="https://stackpath.bootstrapcdn.com/bootstrap/4.5.2/js/bootstrap.min.js"
integrity="sha384-
B4gt1jrGC7Jh4AgTPSdUtOBvfO8shuf57BaghqFfPlYxofvL8/KUEfYiJOMMV+rV"
crossorigin="anonymous"></script>
```

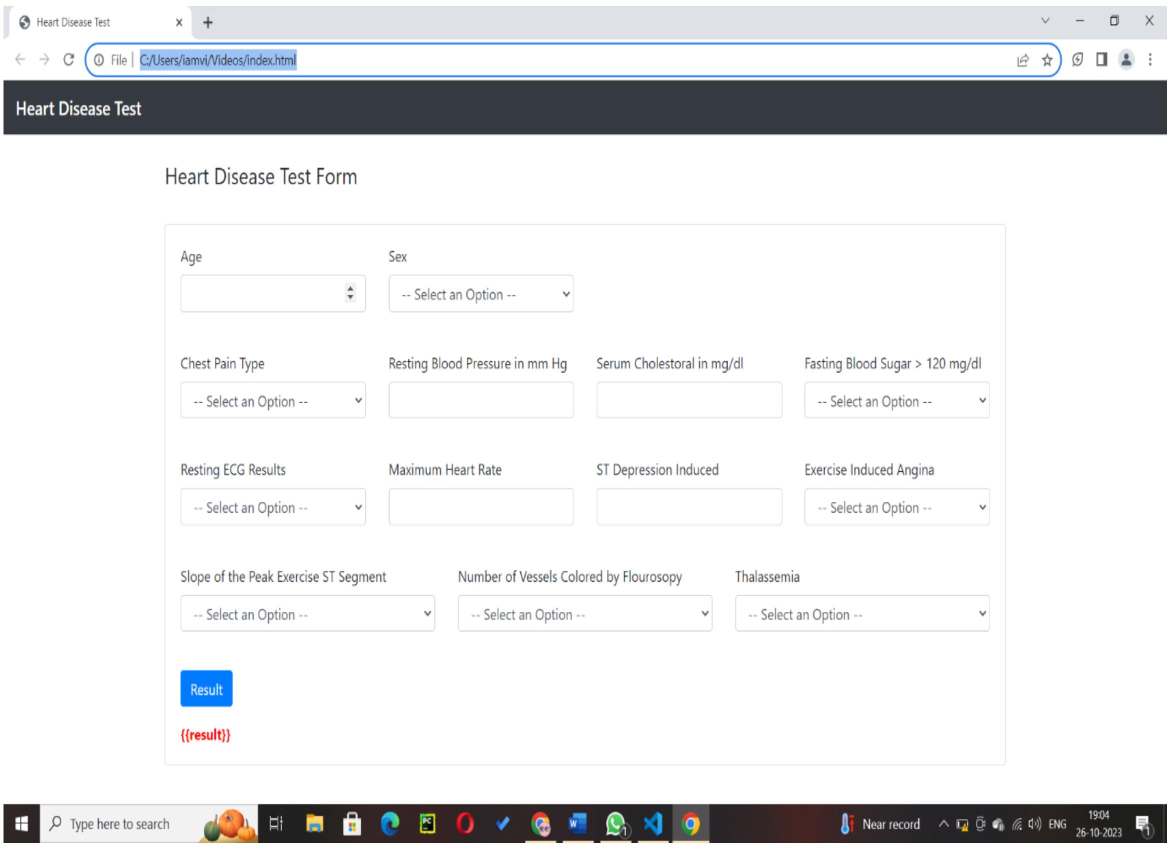
```
<!-- Navbar-->
                    <nav class="navbar navbar-dark bg-dark">
                              <span class="navbar-brand mb-0 h1">Heart Disease Test</span>
                    </nav>
<div class="container">
                              <br>
                              <!--Form-->
                              <form action = "{{url_for('predict')}}" method ="POST" >
                                        <fieldset>
                                        <legend>Heart Disease Test Form</legend><br>
                  <div class="card card-body" >
                    <div class="form-group  row">
                                         <div class="col-sm-3">
<label for="age">Age</label>
                    <input type="number" class="form-control" id="age" name="age" required>
</div>


<div class="col-sm-3">
          <label for="sex">Sex</label>
          <select class="form-control" id="sex"  name="sex" required>
          <option disabled selected value> -- Select an Option -- </option>
           <option value = "0">Male</option>
           <option value = "1">Female</option>
          </select>
 </div>
</div>
 <br>
```

```html
<div class="form-group  row">
<div class="col-sm">
<label for="cp">Chest Pain Type</label>
<select class="form-control" id="cp" name = "cp" required>
<option disabled selected value> -- Select an Option -- </option>
<option value = "0">Typical Angina</option>
<option value = "1">Atypical Angina</option>
<option value = "2">Non-anginal Pain</option>
 <option value = "3">Asymptomatic</option>
</select>
 </div>
<div class="col-sm"
<label for="trestbps">Resting Blood Pressure in mm Hg</label>
<input type="number" class="form-control" id="trestbps" name="trestbps"  required>
 </div>
 <div class="col-sm">
<label for="chol">Serum Cholestoral in mg/dl</label>
<input type="number" class="form-control" id="chol" name="chol" required>
</div>
<div class="col-sm">
<label for="fbs">Fasting Blood Sugar > 120 mg/dl</label>
<select class="form-control" id="fbs" name="fbs" required>
 <option disabled selected value> -- Select an Option -- </option>
<option value = "0">False</option>
<option value = "1">True</option>
```

```
</select>

</div>

</div>


 <br>

<div class="form-group row">

<div class="col-sm">

<label for="restecg">Resting ECG Results </label>

<select class="form-control" id="restecg" name="restecg" required>

<option disabled selected value> -- Select an Option -- </option>

<option value = "0">Normal</option>

<option value = "1">Having ST-T wave abnormality  </option>

<option value = "2"> Probable or definite left ventricular hypertrophy  </option>

</select>

</div>

  <div class="col-sm">

<label for="thalach">Maximum Heart Rate</label>

<input type="number" class="form-control" id="thalach" name="thalach" required>

 </div>

<div class="col-sm">

<label for="exang">ST Depression Induced</label>

<input type="number" step="any" class="form-control" id="exang" name="exang" required>

</div>

<div class="col-sm">

<label for="exang">Exercise Induced Angina </label>
```

```
<select class="form-control" id="oldpeak" name="oldpeak" required>
 <option disabled selected value> -- Select an Option -- </option>
<option value = "0">No</option>
<option value = "1">Yes</option>
</select>
 </div>
</div>
<br>
<div class="form-group  row">
<div class="col-sm">
<label for="slope">Slope of the Peak Exercise ST Segment </label>
<select class="form-control" id="slope" name="slope" required>
<option disabled selected value> -- Select an Option -- </option>
<option value = "0">Upsloping</option>
<option value = "1">Flat</option>
<option value = "2">Downsloping</option>
</select>
</div>
<div class="col-sm">
<label for="ca">Number of Vessels Colored by Flourosopy</label>
<select class="form-control" id="ca" name = "ca" required>
<option disabled selected value> -- Select an Option -- </option>
<option value = "0">0</option>
<option value = "1">1</option>
<option value = "2">2</option>
<option value = "3">3</option>
</select>
```

```
</div>
<div class="col-sm">
<label for="thal">Thalassemia</label>
<select class="form-control" id="thal" name = "thal" required>
<option disabled selected value> -- Select an Option -- </option>
<option value = "3">Normal</option>
<option value = "6">Fixed defect</option>
<option value = "7">Reversable defect</option>
</select>
 </div>
</div>
<br><div class="form-group">
<input class="btn btn-primary" type="submit" value="Result">
</div>
 <!--Prediction Result-->
<div id ="result">
<strong style="color:red">{{result}}</strong>
</div>
 </div>
</fieldset>
</form>
</div>
</body>
</html>
```



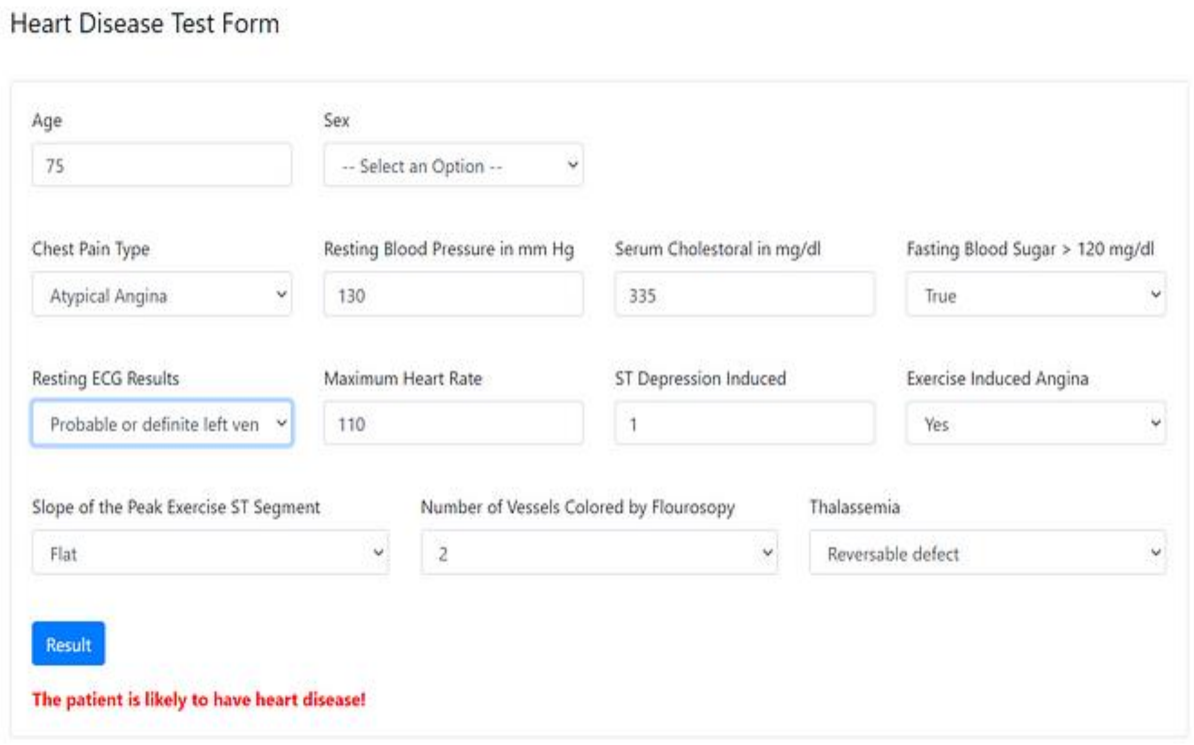
Heart Disease Test

Heart Disease Test Form

Age	Sex: -- Select an Option --		
Chest Pain Type: -- Select an Option --	Resting Blood Pressure in mm Hg	Serum Cholestoral in mg/dl	Fasting Blood Sugar > 120 mg/dl: -- Select an Option --
Resting ECG Results: -- Select an Option --	Maximum Heart Rate	ST Depression Induced	Exercise Induced Angina: -- Select an Option --
Slope of the Peak Exercise ST Segment: -- Select an Option --	Number of Vessels Colored by Flourosopy: -- Select an Option --	Thalassemia: -- Select an Option --	

Result

{{result}}



Heart Disease Test Form

Age: 75	Sex: -- Select an Option --		
Chest Pain Type: Atypical Angina	Resting Blood Pressure in mm Hg: 130	Serum Cholestoral in mg/dl: 335	Fasting Blood Sugar > 120 mg/dl: True
Resting ECG Results: Probable or definite left ven	Maximum Heart Rate: 110	ST Depression Induced: 1	Exercise Induced Angina: Yes
Slope of the Peak Exercise ST Segment: Flat	Number of Vessels Colored by Flourosopy: 2	Thalassemia: Reversable defect	

Result

The patient is likely to have heart disease!