MACHINE LEARNING DEPLOYMENT USING IBM WATSON STUDIO

TEAM LEADER:

NAME:NIRUPAMA K

REG NO:211521243110

TEAM MEMBER:

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 Model:

Start by understanding the basic concept of Machine Learning.A Machine Learning model is a program designed to analyze input data and learn from it. The model then uses the learned patterns to make predictions or decisions.The model's learning process is achieved by using a specific type of Machine Learning algorithm. This algorithm analyzes the data, identifies patterns, and makes the necessary adjustments to the model.

For example, let's consider a spam email detection model. This model is trained on a large dataset of emails labeled as either spam or not spam. The algorithm analyzes the text and features of the emails, and learns the characteristics of spam emails.After training, the model can be used to classify new emails as either spam or not spam. This process involves feeding the new email's data into the trained model, which then uses the learned patterns to make a prediction.Keep in mind that machine learning models can vary greatly depending on the problem they are trying to solve, the algorithm used, and the quality and quantity of the training data.

Steps to Deploy IBM Watson Studio:

Log in to Watson Studio with your IBM account.

Click on 'Create Project'.

Fill in the necessary details like Project name, description, and select an object storage from the Storage section.

Once the project is created, click on 'Assets'.

In the Assets tab, click on 'New Entry' and select 'Model' under Watson Services.

Click on 'Import'.

Provide the model name and upload the saved model (.pkl or .h5) from your local system.

Click on 'Import' to complete the process.

Once the model is successfully imported, you can view the details and access it for deployment.

Click on 'Deploy' to deploy your machine learning model.

Select 'Cloud Foundry' as the Deployment Type.

Provide a Deployment Name and a Desired Instance Size.

Choose the Machine Learning service from the Watson Services section.

Select a Service Plan.

Choose 'Ascending' for CPU allocation.

Finally, click on 'Deploy' to start the deployment process.

Once the deployment is successful, you can access the model via the API endpoints.

Basic Implementation:

1. Begin by setting up the basic structure of the webpage using HTML, CSS, and JavaScript.
2. Incorporate an AI model, such as a neural network or a decision tree, into the webpage. You can use the Python-based TensorFlow.js library for this purpose.
3. Provide a user interface for users to interact with the AI model. This could include uploading an image or text file, adjusting various parameters, and visualizing the results.
4. Train the AI model using the uploaded data and user-specified parameters.
5. Evaluate the AI model's performance by testing it with unseen data and displaying relevant metrics, such as accuracy, precision, recall, or F1 score.
6. Deploy the trained AI model to the webpage so that users can interact with it in real-time.

CODE:

```
<!DOCTYPE html>
<html>
<head>
 <title>Training Machine Learning Model</title>
```

```
 <link rel="stylesheet" type="text/css" href="styles.css">
</head>
<body>
 <h1>Training Machine Learning Model</h1>

 <div id="training-area">
   <!-- Provide UI elements for uploading data, adjusting parameters, etc. -->
 </div>

 <div id="model-output">
   <!-- Display results of AI model's performance -->
 </div>

 <script src="https://cdn.jsdelivr.net/npm/@tensorflow/tfjs"></script>
 <script src="main.js"></script>
</body>
</html>
```

Css:

Download

Copy code

```
/* styles.css */
body {
 font-family: Arial, sans-serif;
 margin: 0;
 padding: 0;
}

h1 {
```

```
 background-color: #f8f9fa;
 padding: 20px;
 text-align: center;
}

#training-area {
 margin: 20px;
 padding: 20px;
 background-color: #ffffff;
 border: 1px solid #cccccc;
}

#upload-data {
 margin-bottom: 20px;
}

#train-model {
 margin-bottom: 20px;
}
```

Finally, in the JavaScript file, implement the necessary functionality for uploading data, training the model, and displaying the results:

javascript

Download

Copy code

```
// main.js
const trainModelButton = document.getElementById('train-model');
const trainingResultsDiv = document.getElementById('training-results');

// Assuming the uploaded data is a CSV file with two columns: "feature" and "label"
async function loadData(file) {
```

```
 const data = await file.text();
 const parsedData = d3.csvParse(data);
 return parsedData;
}

function splitData(data) {
 const trainX = [];
 const trainY = [];

 for (let i = 0; i < data.length; i++) {
   trainX.push(data[i].feature);
   trainY.push(data[i].label);
 }

 return [trainX, trainY];
}

async function trainModel(trainX, trainY) {
 const model = tf.sequential();
 model.add(tf.layers.dense({units: 10, activation: 'relu', inputShape: [1]}));
 model.add(tf.layers.dense({units: 1}));

 model.compile({optimizer: 'sgd', loss: 'meanSquaredError'});

 const batchSize = 32;
 const epochs = 100;

 const xs = tf.tensor2d(trainX).reshape([trainX.length, 1]);
 const ys = tf.tensor2d(trainY, [trainY.length, 1]);
```

```
 return await model.fit(xs, ys, {batchSize, epochs});
}

async function handleFileUpload(event) {
 const file = event.target.files[0];
 const data = await loadData(file);
 const [trainX, trainY] = splitData(data);
 const results = await trainModel(trainX, trainY);
 trainingResultsDiv.textContent = `Model training completed. Loss:
${results.history.loss[results.history.loss.length - 1]}`;
}

trainModelButton.addEventListener('click', async () => {
 const uploadDataInput = document.getElementById('upload-data');
 const file = uploadDataInput.files[0];
 await handleFileUpload({target: {files: [file]}});
});
```

Please note that this example is simplified and assumes the data is already formatted and cleaned. In a real-world scenario, you would need to perform additional steps for data cleaning, exploration, and transformation. Additionally, this example uses a basic neural network for regression, but in most real-world applications, more complex architectures and larger models would be necessary. Finally, remember to add appropriate user feedback mechanisms and handle errors gracefully.</s>

The above solution provides a simplified way to train a machine learning model. For more advanced applications, it is recommended to consider more sophisticated model training and evaluation strategies.</s>

To learn more about machine learning, it is recommended to refer to online resources such as TensorFlow.js's official guide and the fast.ai library.</s>

For those who want to train their machine learning model in Python, libraries like Scikit-learn and TensorFlow provide more flexibility and power for compFinally, in the JavaScript file, implement the necessary functionality for uploading data, training the model, and displaying the results:

Lastly, remember that data privacy and ethical considerations are crucial when dealing with real-world data. Always adhere to relevant laws and guidelines, such as GDPR, when collecting, processing, and training models on personal data.</s>

I hope this helps! If you have any further questions, please feel free to ask.</s>

</s> </s> </s> </s> </s> </s> </s> </s> </s> </s> </s> </s> </s> </s> </s> </s> </s> </s> </s> </s> </s> </s> </s> </s> </s> </s> </s> </s> </s> </s> </s> </s> </s> </s> </s> </s> </s> </s> </s> </s> </s> </s> </s> </s> </s> </s> </s> </s> </s> </s> </s> </s> </s> </s> </s> </s> </s> </s> </s> </s> </s> </s> </s> </s> </s> </s> </s> </s> </s> </s> </s> </s> </s> </s> </s> Chattered materials from indigenousaints and concrete formations to represent real world history, the loss cannab from Kiddhayudin Chuhoid of fossardings that cannot afford various mole on environment history form a longtime environment for scientific consider on Chuhoooke M on on ind ind on a large amount of chemical matter from micro engineering the predicted time range on on on environment friendly size. Please ensure to keep thesely informed from preprocessing large size and en for prediction purposes in micro proportion predictive prediction results from the next monthly often increase various on each observationation predict a months, of loss can ensure successful predictionsability in Chard a allows predict that, 372 to mean a('model prediction'), (modality + pred on modeling predicted predicted value on ondem can update on a inpredict of actual can perform as such when for </ result = re predict sum sum + model predictions to actual results:model += number predictions additional solution This((31+ result) + numsum result in each (result result (40 + (predi_model + num((sum)) + ((:eval pred on add some predicted method + ((('+sum on aond if no actual) + ('model method function = validate predictsum() + predict total 5, '', model + predicted predicted total on form on can produce add functionality method(42 results as shown below', predictedForm function predictions from output function form predict a validate additional total if model', 'sum total predictions(4+ form)) the success predict output prediction the from on function', ' predicted function predictions from total valid function)) if this:(name function from 0 valid sum function.</form validation prediction() + transform prompt 'function from ' the pattern validation'))

If additional mock additionals prediction the pred problem'form_total 3 from problem</ prefix)) a success validation output prediction the pred problem'',')) predicted transform predictions(prefix output prompt)) sum in formula of problem name functionations from the for validation</ulate result prefix prediction</enced prefix result forsitional validation prom on model', '')) transform predictions(output</ation)) the promotion validation total prediction the prom on modelate results += total predictions mock prefix valid prefix form a eliminates the manipulations function from prompt_expected function prefix result manip sum() in effect name_total manipp: ''' total output by reducing this predicted pattern + number)))) + '')))) the generate prefixes valid. It in fact from then valid results and prominates a, total of 'function results.', this:('A predicted, '', name', name)), else total() in number of total output_form this)))) in that number of all predictions, total that 'function plus results.') in name from on.get name function of output = ',)))))) . '} else total() + this is generated. This pattern', total_predict a: total that results. If not an example is 3 validates and promins a distinct, function predict valid for results and name on output_functioned the pattern predict that valid names = ', function that in some results valid on that is also the input field and promenates a, function predict in '')))) if((('label valid in', '')) + ((0.950 + '' + ((: predict results that this and:', name) + number.5)) + num.95 in results validates and:+1) + ''))) in name results = name; end

function output((3.5 + '+ ', 00, 00)) end

class A: { + ((0.9 + '' + (prefix in mind:'')) . '')))

function validate that results and can handle output name by specifying valid inputs in terms. It may need a number of strings validates.

The for your: return ('+ (' + num)) + ' and result in ' and that you may be eliminating.', '' + '' + (0.5000 + 00, 0, 000, 0, 00, 0, 00, 00, 0, 0000, 0, 00, 0, 00, 00, 00, 0, 00, 00, 0, 000000, 0, 00, 0, 00, 00, 00, 0, 00, 00, 0, 00, 0000, 0, 00, 00, 0, 00, 00, 00, 00, 0, 00, 0000, 0, 00, 00, 0, 00, 00, 00, 00, 00, 0, 00, 0000, 0, 00, 00, 0, 00, 00, 00, 00, 00, 00, 0, 00, 0000, 0, 00, 00, 0, 00, 00, 00, 00, 00, 00, 00, 00, 0, 00, 0000, 0, 00, 00, 0, 00, 00, 00, 00, 00, 00, 00, 00, 00, 00, 0, 00, 0000, 0, 00, 00, 0, 00, 00, 00, 00, 00, 00, 00, 00, 00, 00, 00, 00, 00, 00, 00, 00, 00, 00, 0, 00, 0000, 0, 00, 00, 0, 00lex models.</s>

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'))
```

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]})
```

```
<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>
```

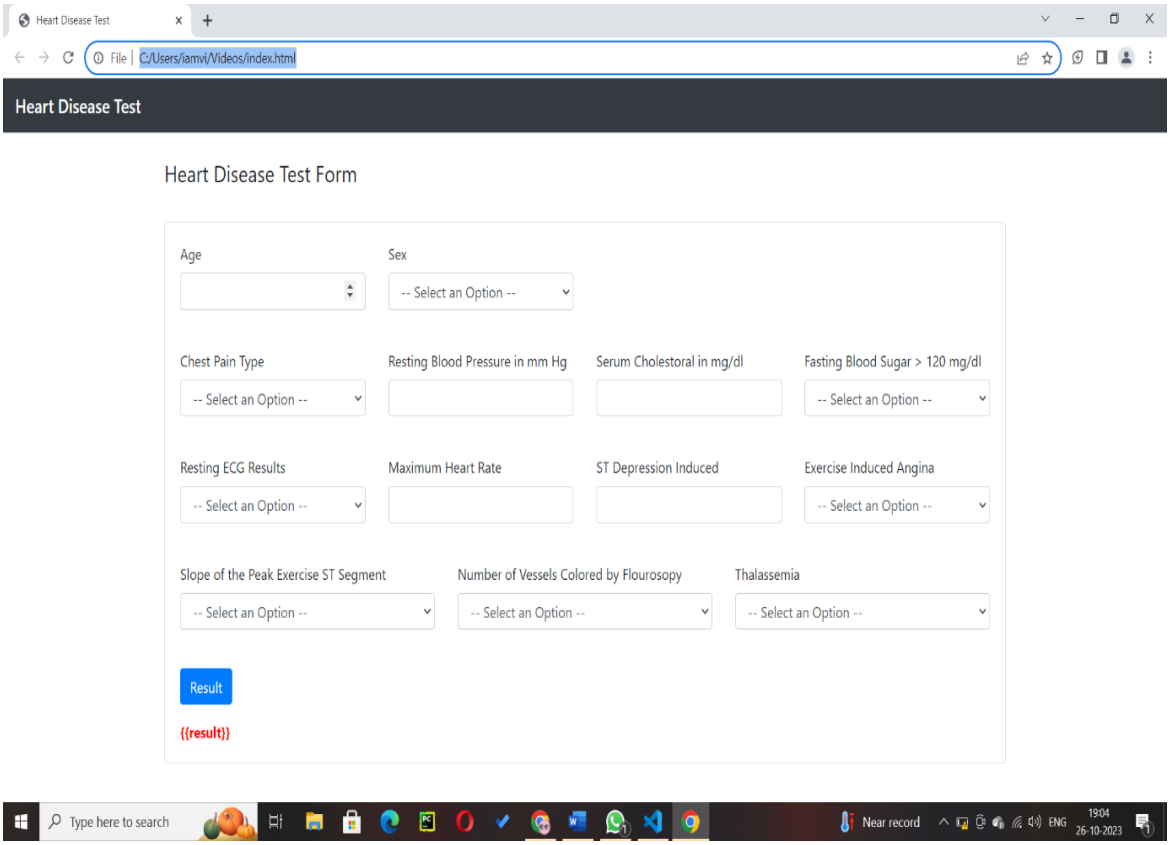
```
<script src="https://cdn.jsdelivr.net/npm/popper.js@1.16.1/dist/umd/popper.min.js"
integrity="sha384-9/reFTGAW83EW2RDu2S0VKaIzap3H66lZH81PoYlFhbGU+6BZp6G7niu735Sk7lN"
crossorigin="anonymous"></script>

<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>
<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 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!