if request.method == "POST":

        data = request.json

        leaves = int(data["leaves"])

        height = float(data["height"])

        temperature = float(data["temperature"])

        humidity = float(data["humidity"])

# Create a DataFrame with the correct feature names

        input\_data = pd.DataFrame([[leaves, height, temperature, humidity]], *columns*=feature\_names\_diameter)

            # Perform the prediction using the model

        diameter = model\_diameter.predict(input\_data)

        #predicted\_diameter = float(diameter[0])

        # Convert it to a JSON serializable type (e.g., float)

        #diameter = float(diameter)

        #diameter = round(diameter, 2)

    #    # print("Scaled diamerer", diameter);

    #     # Create a DataFrame for new data

    #     new\_data\_df = pd.DataFrame(input\_data, columns=feature\_names\_diameter)

    #      # Use the loaded scaler to scale the new data

    #     scaled\_new\_data = model\_feature.transform(new\_data\_df)

    #       # Use the loaded model to make predictions on the scaled new data

    #     predicted\_scaled = model\_diameter.predict(scaled\_new\_data)

    #     # If the target variable was scaled, inverse transform the prediction

    #     diameter2 = model\_target.inverse\_transform(predicted\_scaled.reshape(-1, 1))

     # Scale the input data

        scaled\_input\_data = model\_feature.transform(input\_data)

        # Perform the prediction using the model

        predicted\_scaled = model\_diameter.predict(scaled\_input\_data)

          # Calculate SHAP values for the prediction

        explainer = shap.Explainer(model\_diameter)

        shap\_values = explainer(scaled\_input\_data)

        # Extract SHAP values for each feature

        shap\_values\_list = shap\_values.values[0].tolist()

        shap\_values\_dict = dict(zip(feature\_names\_diameter, shap\_values\_list))

        print(shap\_values\_list)

          # Categorize features based on SHAP values

        positive\_impact\_features = []

        negative\_impact\_features = []

        # for feature, shap\_value in shap\_values\_dict.items():

        #     if shap\_value > 0:

        #         #positive\_impact\_features.append(f"{feature} (+{shap\_value:.2f})")

        #         positive\_impact\_features.append(f"{feature} ")

        #     elif shap\_value < 0:

        #         #negative\_impact\_features.append(f"{feature} ({shap\_value:.2f})")

        #         negative\_impact\_features.append(f"{feature} ")

        # # Construct the message

        # if positive\_impact\_features:

        #     positive\_message = f"The factors that influenced the prediction the most (in order from highest to lowest) are: {', '.join(sorted(positive\_impact\_features, reverse=True))}."

        # else:

        #     positive\_message = "No significant factors influenced the prediction."

        # if negative\_impact\_features:

        #     negative\_message = f"Other factors that influenced the prediction (in order from highest to lowest) are: {', '.join(sorted(negative\_impact\_features, reverse=True))}."

        # else:

        #     negative\_message = "No other factors influenced the prediction."

        # Combine the messages

        #final\_message = positive\_message + " " + negative\_message

        # Sort the shap\_values\_dict by SHAP value first, in descending order

        sorted\_shap\_values = sorted(shap\_values\_dict.items(), *key*=*lambda* *x*: abs(*x*[1]), *reverse*=True)

        # Create a list of features in sorted order, without distinguishing positive or negative

        influential\_features = [feature for feature, shap\_value in sorted\_shap\_values]

        # Construct the message

        if influential\_features:

            message = *f*"The factors that influenced the predicted plan diameter the most (in order from highest to lowest) are: {', '.join(influential\_features)}."

        else:

            message = "No significant factors influenced the prediction."

        # Inverse transform the prediction

        diameter = model\_target.inverse\_transform(predicted\_scaled.reshape(-1, 1))

        diameter = float(diameter[0])

        diameter = round(diameter, 2)

        print("Predicted values (unscaled):", diameter)

        # Display the unscaled predictions

        print("Predicted values (unscaled):", np.round(diameter, 2))

        print( message)

        return jsonify(*predicted\_diameter*=diameter,

*message1* =message)

@app.route("/predict\_height", *methods*=["POST"])

*def* predict\_height():

    print("hello\_height")

    if request.method == "POST":

        data = request.json

        leaves = int(data["leaves"])

        diameter = float(data["diameter"])

        temperature = float(data["temperature"])

        humidity = float(data["humidity"])

        input\_data = pd.DataFrame([[leaves, diameter, temperature, humidity]], *columns*=feature\_names\_height)

            # Perform the prediction using the model

        height = model\_height.predict(input\_data)

scaled\_input\_data = model\_feature.transform(input\_data)

        # Perform the prediction using the model

        predicted\_scaled = model\_height.predict(scaled\_input\_data)

          # Calculate SHAP values for the prediction

        explainer = shap.Explainer(model\_diameter)

        shap\_values = explainer(scaled\_input\_data)

        # Extract SHAP values for each feature

        shap\_values\_list = shap\_values.values[0].tolist()

        shap\_values\_dict = dict(zip(feature\_names\_height, shap\_values\_list))

        print(shap\_values\_list)

          # Categorize features based on SHAP values

        positive\_impact\_features = []

        negative\_impact\_features = []

        # Sort the shap\_values\_dict by SHAP value first, in descending order

        sorted\_shap\_values = sorted(shap\_values\_dict.items(), *key*=*lambda* *x*: abs(*x*[1]), *reverse*=True)

        # Create a list of features in sorted order, without distinguishing positive or negative

        influential\_features = [feature for feature, shap\_value in sorted\_shap\_values]

        # Construct the message

        if influential\_features:

            message\_height = *f*"The factors that influenced the predicted plant height the most (in order from highest to lowest) are: {', '.join(influential\_features)}."

        else:

            message\_height = "No significant factors influenced the prediction."

        # Inverse transform the prediction

       height = model\_target.inverse\_transform(predicted\_scaled.reshape(-1, 1))

        height = float(height[0])

        height = round(height, 2)

        print("Predicted values (unscaled):", height)

        # Display the unscaled predictions

        print("Predicted values (unscaled):", np.round(height, 2))

        print( message)

        return jsonify(*predicted\_height*=height,

*message1* =message\_height)

<!DOCTYPE html>

<html lang="en">

<head>

    <meta name="viewport" content="width=device-width, initial-scale=1">

    <link href=

"https://cdn.jsdelivr.net/npm/bootstrap@5.1.3/dist/css/bootstrap.min.css"

    rel="stylesheet"

    />

    <link rel="stylesheet" type="text/css" href='../static/css/b\_stylediameter.css'>

</head>

<body>

    <nav class="navbar navbar-expand-lg navbar-dark bg-success">

        <div class="container">

            <a class="navbar-brand" href="./">AquaGrowth Forecast</a>

            <button class="navbar-toggler" type="button" data-bs-toggle="collapse" data-bs-target="#navbarNav"

                aria-controls="navbarNav" aria-expanded="false" aria-label="Toggle navigation">

                <span class="navbar-toggler-icon"></span>

            </button>

            <div class="collapse navbar-collapse" id="navbarNav">

                <ul class="navbar-nav ms-auto">

                    <li class="nav-item">

                        <a class="nav-link fw-bold" href="/b\_aquaponics">Aquaponics</a>

                    </li>

                    <li class="nav-item">

                        <a class="nav-link fw-bold" href="/b\_predict">Prediction</a>

                    </li>

                    <li class="nav-item">

                        <a class="nav-link fw-bold" href="https://docs.google.com/forms/d/e/1FAIpQLSf0xjJ3pZ2b509GDvp5LiJ4iGKFKIVorS3NPaYtJXQkgF1fNQ/viewform?usp=pp\_url">Feedback</a>

                    </li>

                    <li class="nav-item">

                        <a class="nav-link fw-bold" href="/">Back to home page</a>

                    </li>

                </ul>

            </div>

        </div>

    </nav>

    <div class="  container-about">

        <h2>Welcome to diameter prediction </h2>

        <br>

        <p>

            Here, you can input the relevant values and click "Submit" button to proceed to diameter prediction.

        </p>

        <br>

        <form id="predictionForm">

            <div class="row">

                <div class="form-group col-md-6">

                    <label for="p\_leaves" class="bold-black">Number of leaves/ Leaves</label>

                    <div class="input-group">

                        <input type="number" class="form-control custom-input" id="p\_leaves" placeholder="Enter number of leaves" step="0.01">

                    </div>

                </div>

                <br>

                <div class="form-group col-md-6">

                    <label for="p\_height" class="bold-black">Plant height (cm)/ Height </label>

                    <div class="input-group">

                        <input type="number" class="form-control custom-input" id="p\_height" placeholder="Enter plant height in cm" step="0.01">

                    </div>

                </div>

                <br>

                <br>

                <div class="form-group col-md-6">

                    <label for="p\_temp" class="bold-black">Ambient temperature (°C)/ Temp</label>

                    <div class="input-group">

                        <input type="number" class="form-control custom-input" id="temp" placeholder="Enter temperature in °C" step="0.01">

                    </div>

                </div>

                <br>

                <div class="form-group col-md-6">

                    <label for="p\_humidity" class="bold-black">Ambient Humidity (%)/ Humidity</label>

                    <div class="input-group">

                        <input type="number" class="form-control  custom-input" id="humidity"  placeholder="Enter the humidity as a percentage" step="0.01" >

                    </div>

                   <!-- <small id="humidityHelp" class="form-text text-muted">Enter the humidity as a percentage.</small> -->

                </div>

            </div>

            <br>

            <div class="form-group text-center">

                <button type="submit" class="btn btn-success" id="submitBtn">Submit</button>

                <button type="button" class="btn btn-secondary" id="clearBtn">Clear</button>

            </div>

            <div class="form-group text-center">

                <label for="predictedDiameter" class="bold-black" style="display: inline-block; margin-right: 10px;">Predicted Diameter in cm:</label>

                <input type="text" class="form-control" id="predictedDiameter" readonly style="display: inline-block; width: auto; max-width: 150px;">

            </div>

            <!-- Right Column: Description Section (also aligned left, next to input) -->

        <div class="description-column">

          <h4 class=" text-center">Explanation</h4>

          <div id="predictionMessage"></div>

      </div>

    </div>

  </div>

        </form>

    </div>

    <script src=

"https://cdn.jsdelivr.net/npm/bootstrap@5.1.3/dist/js/bootstrap.bundle.min.js"> </script>

<script>

    //document.getElementById("predictionForm").addEventListener("submit", function(event) {

     //   event.preventDefault(); // Prevent default form submission

    //});

    document.getElementById("clearBtn").addEventListener("click", *function*() {

        // Clear all input fields

        document.getElementById("p\_leaves").value = "";

        document.getElementById("p\_height").value = "";

        document.getElementById("temp").value = "";

        document.getElementById("humidity").value = "";

        document.getElementById("predictedDiameter").value = "";

         // Clear the description message

        document.getElementById("predictionMessage").innerHTML = "";

    });

    document.getElementById("predictionForm").addEventListener("submit", *function*(*event*) {

*event*.preventDefault();

*var* leaves = parseInt(document.getElementById("p\_leaves").value);

*var* height = parseFloat(document.getElementById("p\_height").value);

*var* temperature = parseFloat(document.getElementById("temp").value);

*var* humidity = parseFloat(document.getElementById("humidity").value);

 // Check if any field is empty

    if (!leaves || !height || !temperature || !humidity) {

        alert("Please fill in all the fields.");

        return; // Stop form submission

    }

            fetch('/predict', {

                method: 'POST',

                headers: {

                    'Content-Type': 'application/json',

                },

                body: JSON.stringify({

                    leaves:leaves,

                    height: height,

                    temperature: temperature,

                    humidity: humidity

                })

            })

            .then(*response* *=>* *response*.json())

            .then(*data* *=>* {

                document.getElementById("predictedDiameter").value = *data*.predicted\_diameter;

                // Set the dynamic message in the description section

                document.getElementById("predictionMessage").innerHTML = *data*.message1;

           // Show the description message

            document.querySelector(".description-column").style.display = "block";

            })

            .catch((*error*) *=>* {

                console.error('Error:', *error*);

            });

        });

</script>

</body>

</html>

 <form id="predictionForm">

            <div class="form-row">

                <div class="form-group col-md-6">

                    <label for="w\_tds">Total Dissolved Solids (TDS) (ppm) </label>

                    <div class="input-group">

                        <input type="number" class="form-control" id="w\_tds" placeholder="Enter TDS in pmm" step="0.01">

                    </div>

                </div>

                <div class="form-group col-md-6">

                    <label for="w\_ec">Conductivity</label>

                    <div class="input-group">

                        <input type="number" class="form-control" id="w\_ec" placeholder="Enter EC in μS/cm" step="0.01">

                    </div>

                </div>

                <div class="form-group col-md-6">

                    <label for="w\_temp">Water temperature (°C)</label>

                    <div class="input-group">

                        <input type="number" class="form-control" id="w\_temp" placeholder="Enter temperature in °C" step="0.01">

                    </div>

                </div>

            </div>

            <br>

            <div class="form-group">

                <button type="submit" class="btn btn-success" id="submitBtn">Submit</button>

                <button type="button" class="btn btn-secondary" id="clearBtn">Clear</button>

            </div>

            <div class="form-group">

                <label for="predictedpH">Predicted pH:</label>

                <input type="text" class="form-control" id="predictedpH" readonly>

            </div>

        </form>