Model Optimization and Tuning Phase Template

Date	11th July 2024
Team ID	SWTID1720109498
Project Title	Blueberry Yeild Predictor
Maximum Marks	10 Marks

Model Frontend Implementation

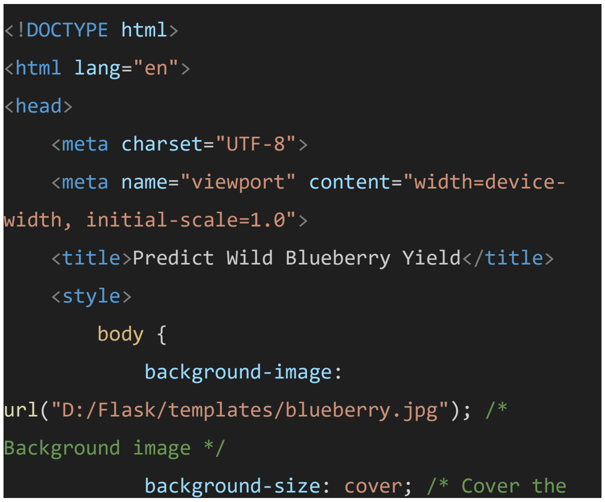
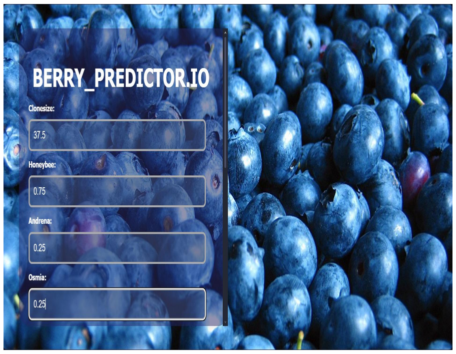
The Model Frontend Implementation Phase involves creating front end implementations so that real time users can interact with it without having any prior ML or DL knowledge. It is a Flask Framework based frontend implementation having a main folder called Flask .Flask folder has multiple sub-folders called templates and models and a file called app.py, templates folder has two files called index.html, predict.html and also a background image, models folder has the model file named model0.pkl.

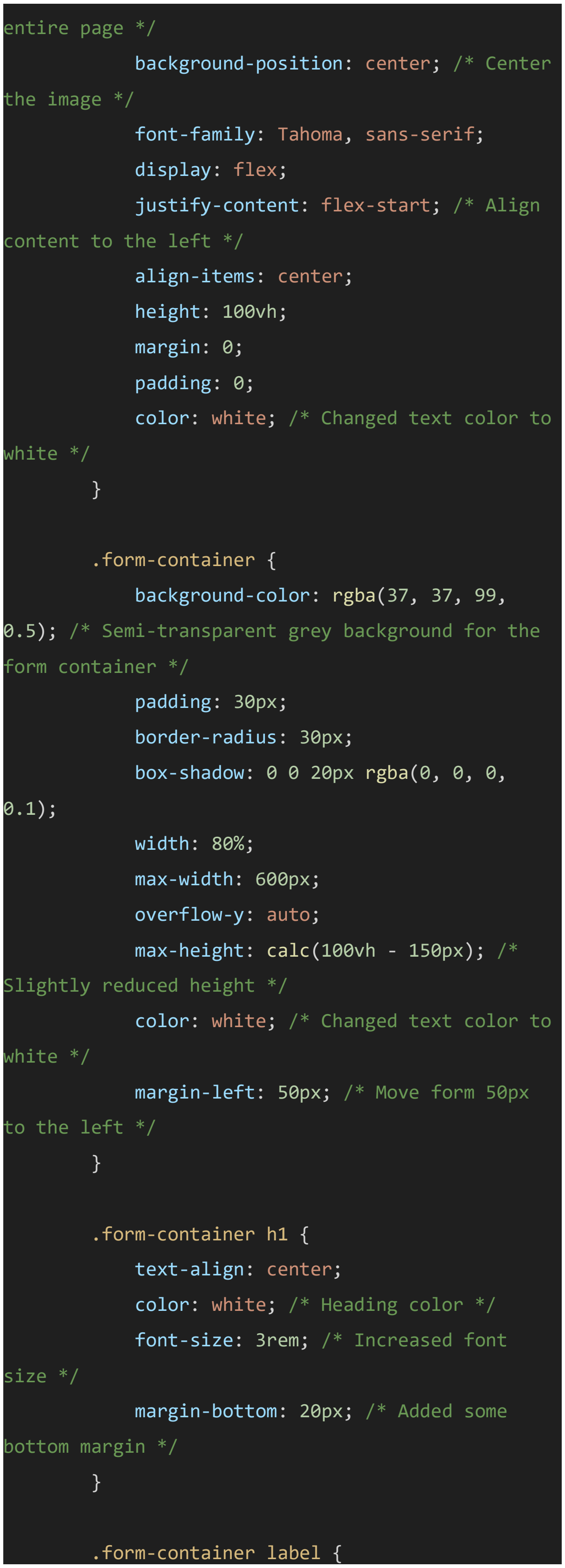
While execution we shall go to the Flask folder directory through anaconda prompt and type

Python app.py followed by enter.

Hyperparameter Tuning Documentation (6 Marks):

FILE NAMES	CODE	IMAGE

Index
<html>
	
	<!DOCTYPE html>
	<html lang="en">
	<head>
	    <meta charset="UTF-8">
	    <meta name="viewport" content="width=device-width, initial-scale=1.0">
	    <title>Predict Wild Blueberry Yield</title>
	    <style>
	        body {
	            background-image: url("D:/Flask/templates/blueberry.jpg"); /* Background image */
	            background-size: cover; /* Cover the
		
		BERRY_PREDICTOR.IO
		Clonesize:
		37.5
		Honeybee:
		0.75
		Andrena:
		0.25
		Osmia:
		0.25

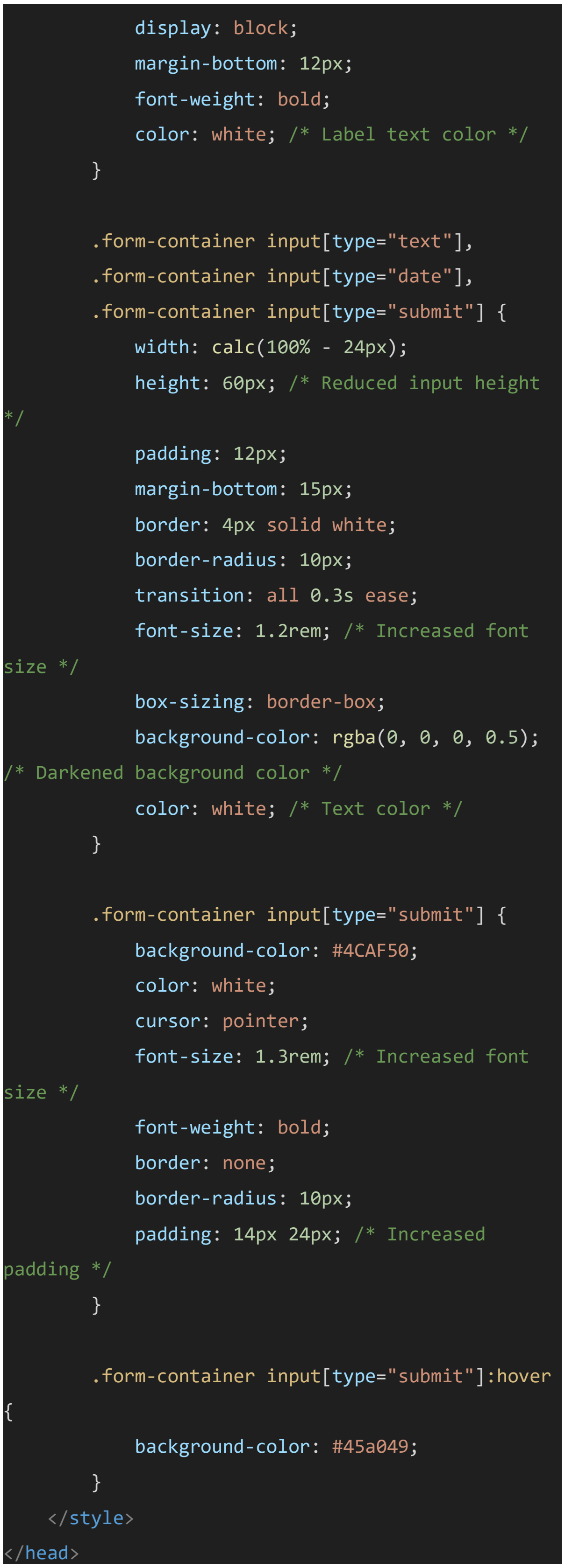


```
entire page */
            background-position: center; /* Center
the image */
            font-family: Tahoma, sans-serif;
            display: flex;
            justify-content: flex-start; /* Align
content to the left */
            align-items: center;
            height: 100vh;
            margin: 0;
            padding: 0;
            color: white; /* Changed text color to
white */
        }

        .form-container {
            background-color: rgba(37, 37, 99,
0.5); /* Semi-transparent grey background for the
form container */
            padding: 30px;
            border-radius: 30px;
            box-shadow: 0 0 20px rgba(0, 0, 0,
0.1);
            width: 80%;
            max-width: 600px;
            overflow-y: auto;
            max-height: calc(100vh - 150px); /*
Slightly reduced height */
            color: white; /* Changed text color to
white */
            margin-left: 50px; /* Move form 50px
to the left */
        }

        .form-container h1 {
            text-align: center;
            color: white; /* Heading color */
            font-size: 3rem; /* Increased font
size */
            margin-bottom: 20px; /* Added some
bottom margin */
        }

        .form-container label {
```



```
            display: block;
            margin-bottom: 12px;
            font-weight: bold;
            color: white; /* Label text color */
        }

        .form-container input[type="text"],
        .form-container input[type="date"],
        .form-container input[type="submit"] {
            width: calc(100% - 24px);
            height: 60px; /* Reduced input height
*/
            padding: 12px;
            margin-bottom: 15px;
            border: 4px solid white;
            border-radius: 10px;
            transition: all 0.3s ease;
            font-size: 1.2rem; /* Increased font
size */
            box-sizing: border-box;
            background-color: rgba(0, 0, 0, 0.5);
/* Darkened background color */
            color: white; /* Text color */
        }

        .form-container input[type="submit"] {
            background-color: #4CAF50;
            color: white;
            cursor: pointer;
            font-size: 1.3rem; /* Increased font
size */
            font-weight: bold;
            border: none;
            border-radius: 10px;
            padding: 14px 24px; /* Increased
padding */
        }

        .form-container input[type="submit"]:hover
{
            background-color: #45a049;
        }
    </style>
</head>
```

```
<body>
    <div class="form-container">
        <h1>BERRY_PREDICTOR.IO</h1>
        <form action="/predict" method="post">
            <label
for="clonesize">Clonesize:</label>
            <input type="text" id="clonesize"
name="clonesize"><br>
            <label
for="honeybee">Honeybee:</label>
            <input type="text" id="honeybee"
name="honeybee"><br>
            <label for="andrena">Andrena:</label>
            <input type="text" id="andrena"
name="andrena"><br>
            <label for="osmia">Osmia:</label>
            <input type="text" id="osmia"
name="osmia"><br>
            <label for="MinOfUpperTRange">Min Of
Upper T Range:</label>
            <input type="text"
id="MinOfUpperTRange" name="MinOfUpperTRange"><br>
            <label
for="AverageOfUpperTRange">Average Of Upper T
Range:</label>
            <input type="text"
id="AverageOfUpperTRange"
name="AverageOfUpperTRange"><br>
            <label
for="AverageOfLowerTRange">Average Of Lower T
Range:</label>
            <input type="text"
id="AverageOfLowerTRange"
name="AverageOfLowerTRange"><br>
            <label for="RainingDays">Raining
Days:</label>
            <input type="text" id="RainingDays"
name="RainingDays"><br>
            <label for="seeds">Seeds:</label>
            <input type="text" id="seeds"
name="seeds"><br>
            <input type="submit" value="Predict">
        </form>
    </div>
```

<table>
<tr>
<td></td>
<td>

```html
</body>
</html>
```

</td>
<td></td>
</tr>
<tr>
<td>Predict
<html></td>
<td>

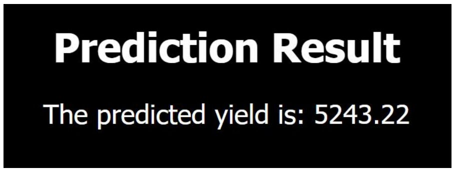
```html
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <title>Prediction Result</title>
    <style>
        body {
            background-color: #000000; /* Pitch black background color */
            font-family: Tahoma, sans-serif;
            display: flex;
            justify-content: center; /* Center content horizontally */
            align-items: center;
            height: 100vh;
            margin: 0;
            padding: 0;
            color: white; /* Changed text color to white */
        }

        .result-container {
            text-align: center;
        }

        .result-container h1 {
            font-size: 3rem;
            margin-bottom: 20px;
        }

        .result-container p {
            font-size: 2rem;
        }
    </style>
</head>
<body>
    <div class="result-container">
        <h1>Prediction Result</h1>
```

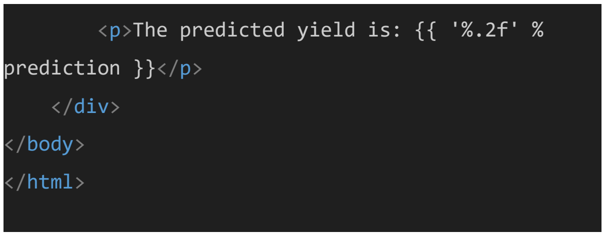
</td>
<td>



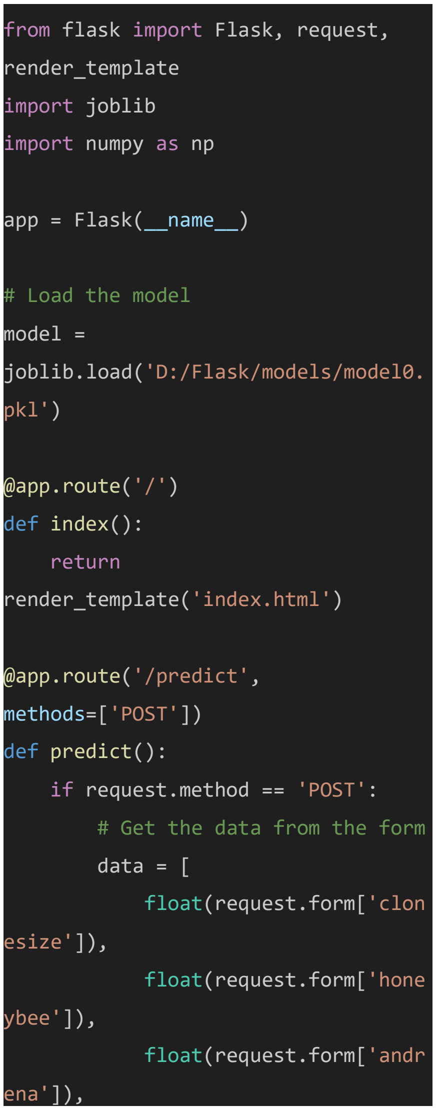
Prediction Result

The predicted yield is: 5243.22

</td>
</tr>
</table>

<table>
<tr>
<td></td>
<td>
<pre>
        <p>The predicted yield is: {{ '%.2f' % prediction }}</p>
    </div>
</body>
</html>
</pre></td>
<td></td>
</tr>
</table>

Main Launching code <app.py>:

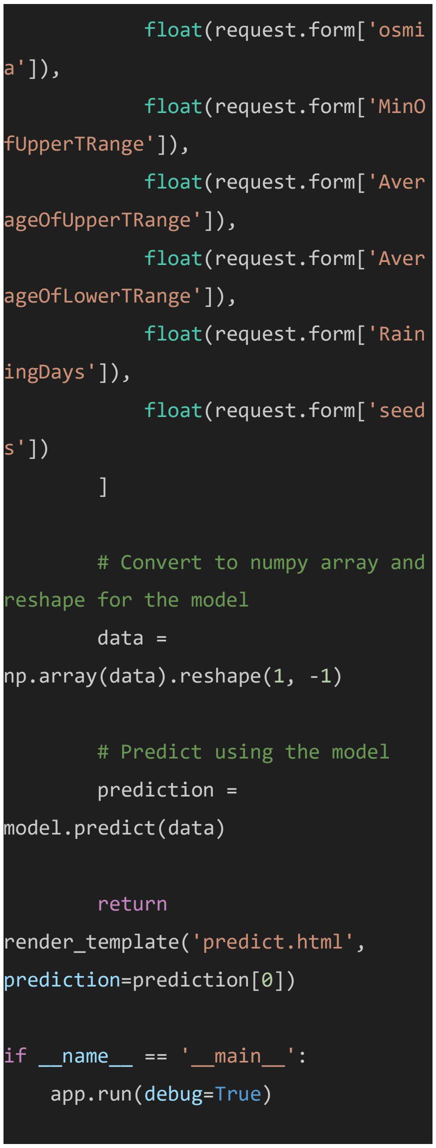
<table>
<tr>
<th>FILE NAMES</th>
<th>CODE</th>
<th>IMAGE</th>
</tr>
<tr>
<td>app.py</td>
<td>
<pre>
from flask import Flask, request, render_template
import joblib
import numpy as np

app = Flask(__name__)

# Load the model
model = joblib.load('D:/Flask/models/model0.pkl')

@app.route('/')
def index():
    return render_template('index.html')

@app.route('/predict', methods=['POST'])
def predict():
    if request.method == 'POST':
        # Get the data from the form
        data = [
            float(request.form['clonesize']),
            float(request.form['honeybee']),
            float(request.form['andrena']),
</pre></td>
<td>NO VISIBLE OUTPUT</td>
</tr>
</table>



```
            float(request.form['osmia']),
            float(request.form['MinOfUpperTRange']),
            float(request.form['AverageOfUpperTRange']),
            float(request.form['AverageOfLowerTRange']),
            float(request.form['RainingDays']),
            float(request.form['seeds'])
        ]

        # Convert to numpy array and reshape for the model
        data = np.array(data).reshape(1, -1)

        # Predict using the model
        prediction = model.predict(data)

        return render_template('predict.html', prediction=prediction[0])

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

Working ALGORITHM:

The app.py code helps us launch the Flask application, at first the index.html file is activated on the browser and a form appears to cater to the model by providing space for model input, once the predict button is clicked, the predict html template is activated and the yield prediction is seen on the browser window. This entire process is smoothly executed using the model file that has been saved on the computer.