

Name: **Cynthia Onyia**

Batch Code: **LISUM44**





Submission Date: **28th April 2025**

Submitted To: **Data Glacier Team**

Details: The following contains screenshots of the processes I used to create my **first Flask app** (Flask deployment) using the Iris datasets (toy data).


1. Folder structure (showing model.pkl, app.py, templates/ folder)

op > Cynthia > codes > flask_project

<input type="checkbox"/> Name	Date modified	Type	Size
 templates	27/04/2025 16:49	File folder	
 app.py	28/04/2025 22:26	Python File	1 KB
 model.pkl	27/04/2025 16:43	PKL File	161 KB
 model_train.py	27/04/2025 16:43	Python File	1 KB

The templates folder contains my index.html file

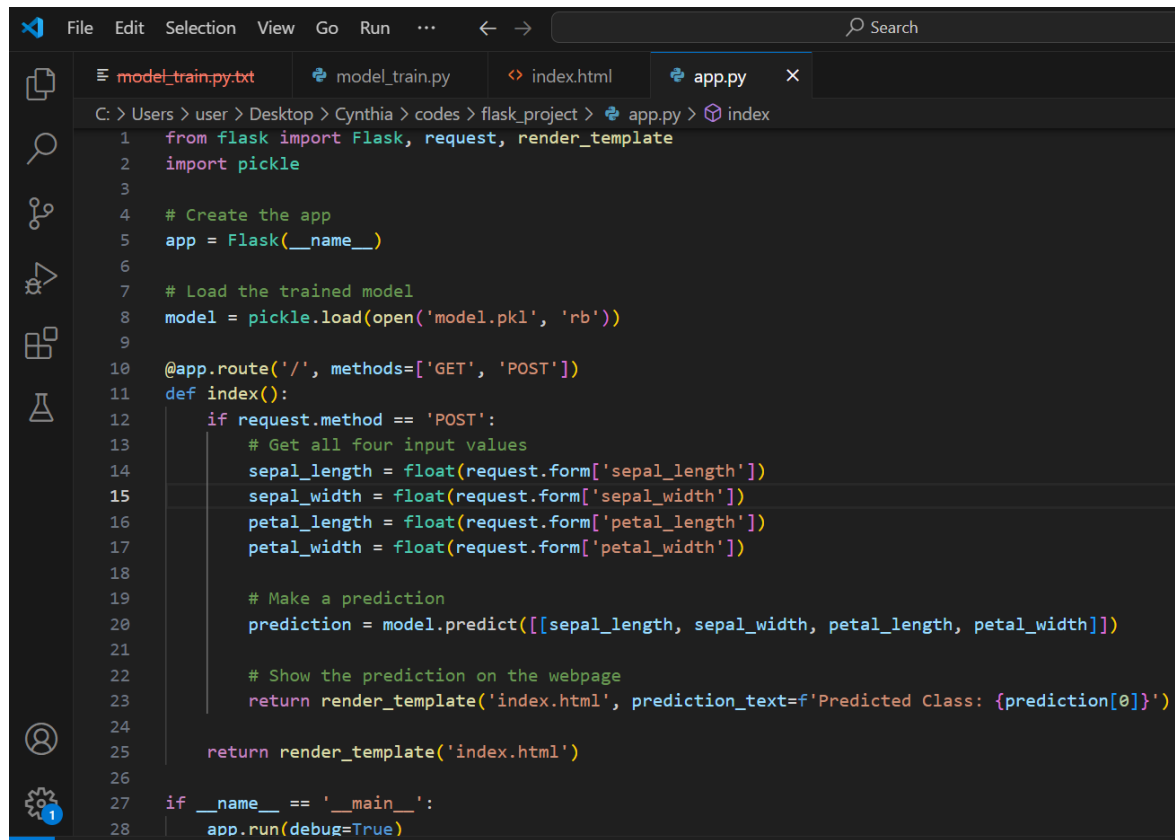
top > Cynthia > codes > flask_project > templates

<input type="checkbox"/> Name	Date modified	Type	Size
 index.html	28/04/2025 22:34	Chrome HTML Docu...	2 KB

2. Code for model_train.py

```
File Edit Selection View Go Run ... Search
model_train.py.txt model_train.py x index.html app.py
C:\Users\user\Desktop\Cynthia\codes\flask_project> model_train.py > ...
1 # model_train.py
2 from sklearn.datasets import load_iris
3 from sklearn.ensemble import RandomForestClassifier
4 import pickle
5
6 # Load data
7 iris = load_iris()
8 X, y = iris.data, iris.target
9
10 # Train model
11 model = RandomForestClassifier()
12 model.fit(X, y)
13
14 # Save the model
15 with open('model.pkl', 'wb') as f:
16     pickle.dump(model, f)
17
18 print("Model saved as model.pkl")
19
```

3. Code for app.py



```
1 from flask import Flask, request, render_template
2 import pickle
3
4 # Create the app
5 app = Flask(__name__)
6
7 # Load the trained model
8 model = pickle.load(open('model.pkl', 'rb'))
9
10 @app.route('/', methods=['GET', 'POST'])
11 def index():
12     if request.method == 'POST':
13         # Get all four input values
14         sepal_length = float(request.form['sepal_length'])
15         sepal_width = float(request.form['sepal_width'])
16         petal_length = float(request.form['petal_length'])
17         petal_width = float(request.form['petal_width'])
18
19         # Make a prediction
20         prediction = model.predict([[sepal_length, sepal_width, petal_length, petal_width]])
21
22         # Show the prediction on the webpage
23         return render_template('index.html', prediction_text=f'Predicted Class: {prediction[0]}')
24
25     return render_template('index.html')
26
27 if __name__ == '__main__':
28     app.run(debug=True)
```

4. Code for index.html



```
1 <!DOCTYPE html>
2 <html lang="en">
3 <head>
4   <meta charset="UTF-8">
5   <title>Flask Deployment Project</title>
6   <style>
7     body {
8       font-family: Arial, sans-serif;
9       text-align: center;
10      margin-top: 50px;
11      background-color: #f2f2f2;
12    }
13    form {
14      margin-top: 20px;
15    }
16    input[type="text"] {
17      padding: 10px;
18      width: 200px;
19      font-size: 16px;
20    }
21    button {
22      padding: 10px 20px;
23      font-size: 16px;
24      background-color: #4CAF50;
25      color: white;
26      border: none;
27      cursor: pointer;
28    }
29    button:hover {
30      background-color: #45a049;
31    }
32    h1, h2 {
33      color: #333;
34    }
35  </style>
36 </head>
37
38 <body>
39   <h1>Iris Prediction</h1>
40
41   <form action="/" method="post">
42     <label for="sepal_length">Sepal Length:</label><br>
43     <input type="text" id="sepal_length" name="sepal_length" required><br><br>
44
45     <label for="sepal_width">Sepal Width:</label><br>
46     <input type="text" id="sepal_width" name="sepal_width" required><br><br>
47
48     <label for="petal_length">Petal Length:</label><br>
49     <input type="text" id="petal_length" name="petal_length" required><br><br>
```

```

50
51     <label for="petal_width">Petal Width:</label><br>
52     <input type="text" id="petal_width" name="petal_width" required><br><br>
53
54     <button type="submit">Predict</button>
55 </form>
56
57 {% if prediction_text %}
58     <h2>{{ prediction_text }}</h2>
59 {% endif %}
60

```

5. Terminal running Flask (python app.py)

```

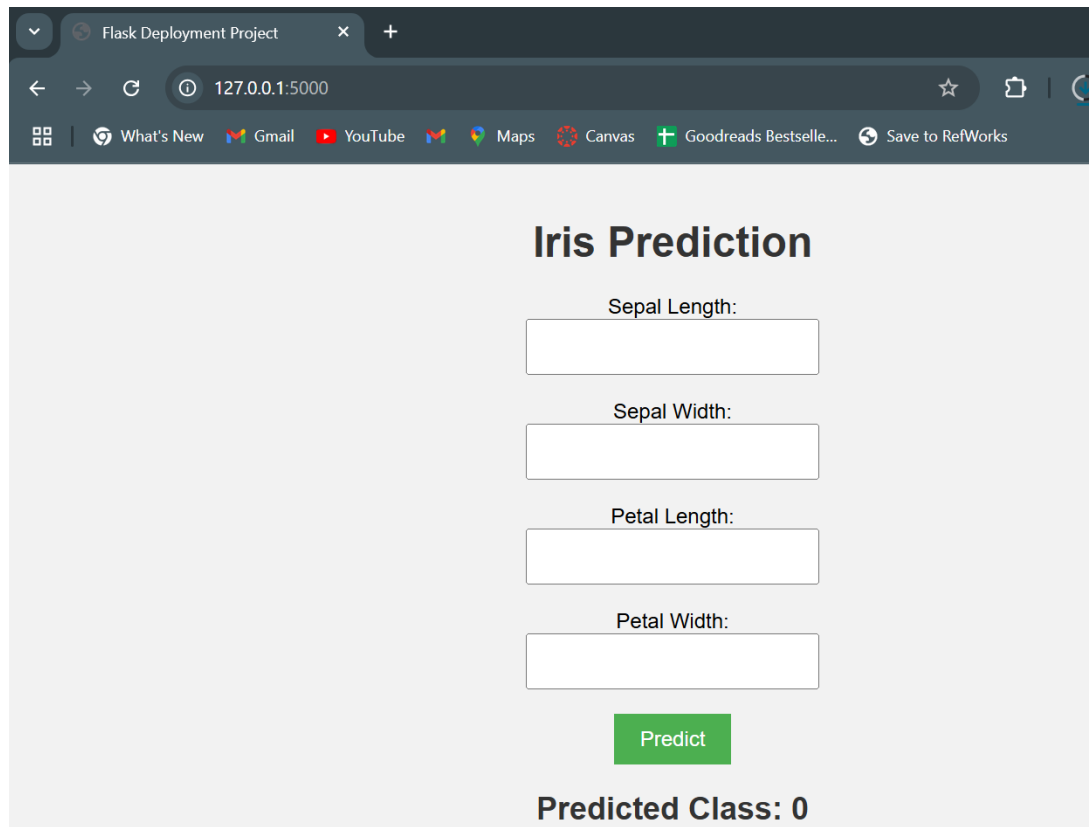
C:\Users\user\Desktop\Cynthia\codes\flask_project>python app.py
* Serving Flask app 'app'
* Debug mode: on
WARNING: This is a development server. Do not use it in a production deployment. Use a production WSGI server instead.
* Running on http://127.0.0.1:5000
Press CTRL+C to quit
* Restarting with stat
* Debugger is active!
* Debugger PIN: 897-334-901
127.0.0.1 - - [28/Apr/2025 22:35:25] "GET / HTTP/1.1" 200 -
127.0.0.1 - - [28/Apr/2025 22:36:14] "POST / HTTP/1.1" 200 -

```

6. Browser showing your form (before prediction)

The screenshot shows a web browser window titled 'Flask Deployment Project' with the address '127.0.0.1:5000'. The page displays the 'Iris Prediction' form. The form contains four input fields, each with a label above it: 'Sepal Length' (5.1), 'Sepal Width' (3.5), 'Petal Length' (1.4), and 'Petal Width' (0.2). Below these fields is a green button labeled 'Predict'.

7. Browser showing the prediction result



The screenshot shows a web browser window with the title "Flask Deployment Project". The address bar displays "127.0.0.1:5000". The browser's bookmark bar includes links for "What's New", "Gmail", "YouTube", "Maps", "Canvas", "Goodreads Bestselle...", and "Save to RefWorks".

The main content area of the browser displays the "Iris Prediction" interface. It features four input fields for "Sepal Length:", "Sepal Width:", "Petal Length:", and "Petal Width:". Below these fields is a green "Predict" button. At the bottom of the interface, the text "Predicted Class: 0" is displayed.

Iris Prediction

Sepal Length:

Sepal Width:

Petal Length:

Petal Width:

Predict

Predicted Class: 0