

**NAME: Sarima Iyayi**

**BATCH CODE: LISUM33**

### **Data Selection:**

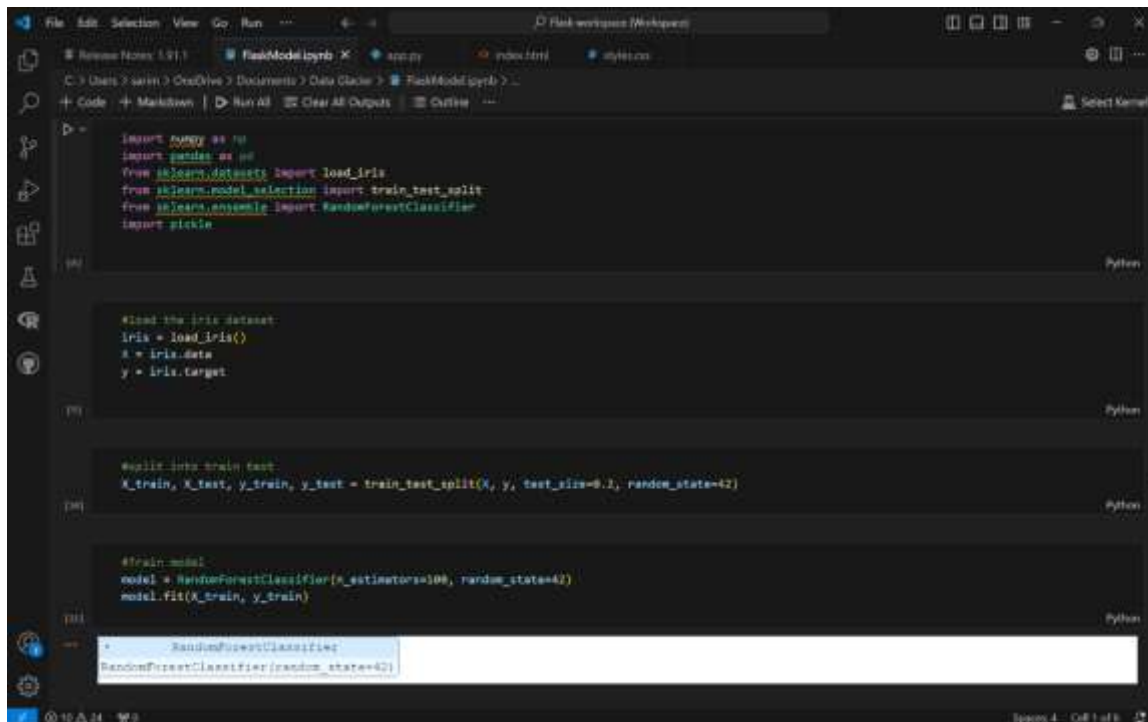
**The Iris dataset** comprises 150 samples of iris flowers, with 50 samples from each of three species: Iris setosa, Iris virginica, and Iris versicolor. The dataset contains four features measured from each flower: sepal length, sepal width, petal length, and petal width.

**Out[1]:**

	sepalength	sepalwidth	petallength	petalwidth	class
0	5.1	3.5	1.4	0.2	Iris-setosa
1	4.9	3.0	1.4	0.2	Iris-setosa
2	4.7	3.2	1.3	0.2	Iris-setosa
3	4.6	3.1	1.5	0.2	Iris-setosa
4	5.0	3.6	1.4	0.2	Iris-setosa
...	...	...	...	...	...
145	6.7	3.0	5.2	2.3	Iris-virginica
146	6.3	2.5	5.0	1.9	Iris-virginica
147	6.5	3.0	5.2	2.0	Iris-virginica
148	6.2	3.4	5.4	2.3	Iris-virginica
149	5.9	3.0	5.1	1.8	Iris-virginica

150 rows x 5 columns

## Model Training:



```
import numpy as np
import pandas as pd
from sklearn.datasets import load_iris
from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestClassifier
import pickle


#load the iris dataset
iris = load_iris()
x = iris.data
y = iris.target

#split into train test
x_train, x_test, y_train, y_test = train_test_split(x, y, test_size=0.2, random_state=42)

#train model
model = RandomForestClassifier(n_estimators=100, random_state=42)
model.fit(x_train, y_train)

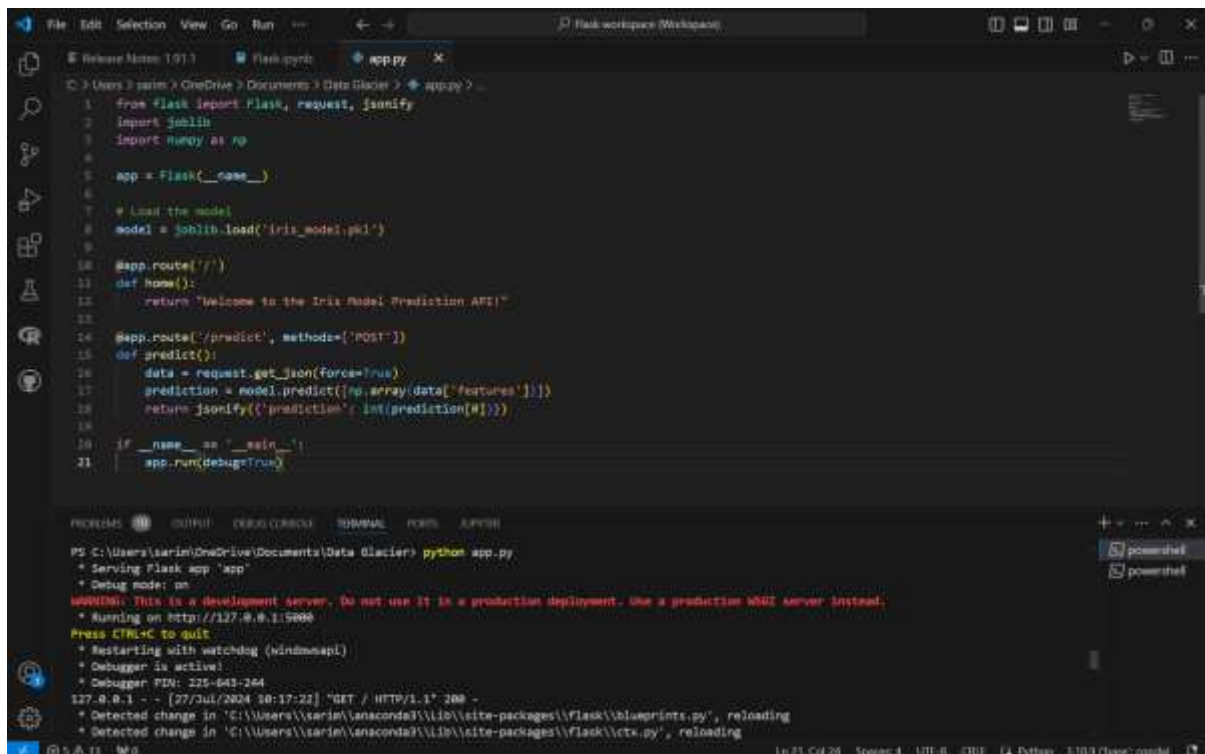
# Random Forest Classifier
RandomForestClassifier(random_state=42)
```

## Saving the Model:



```
#save model
with open('iris_model.pkl', 'wb') as f:
    pickle.dump(model, f)
```

## Developing Flask Application:



```
from flask import Flask, request, jsonify
import joblib
import numpy as np

app = Flask(__name__)

# Load the model
model = joblib.load('iris_model.pkl')

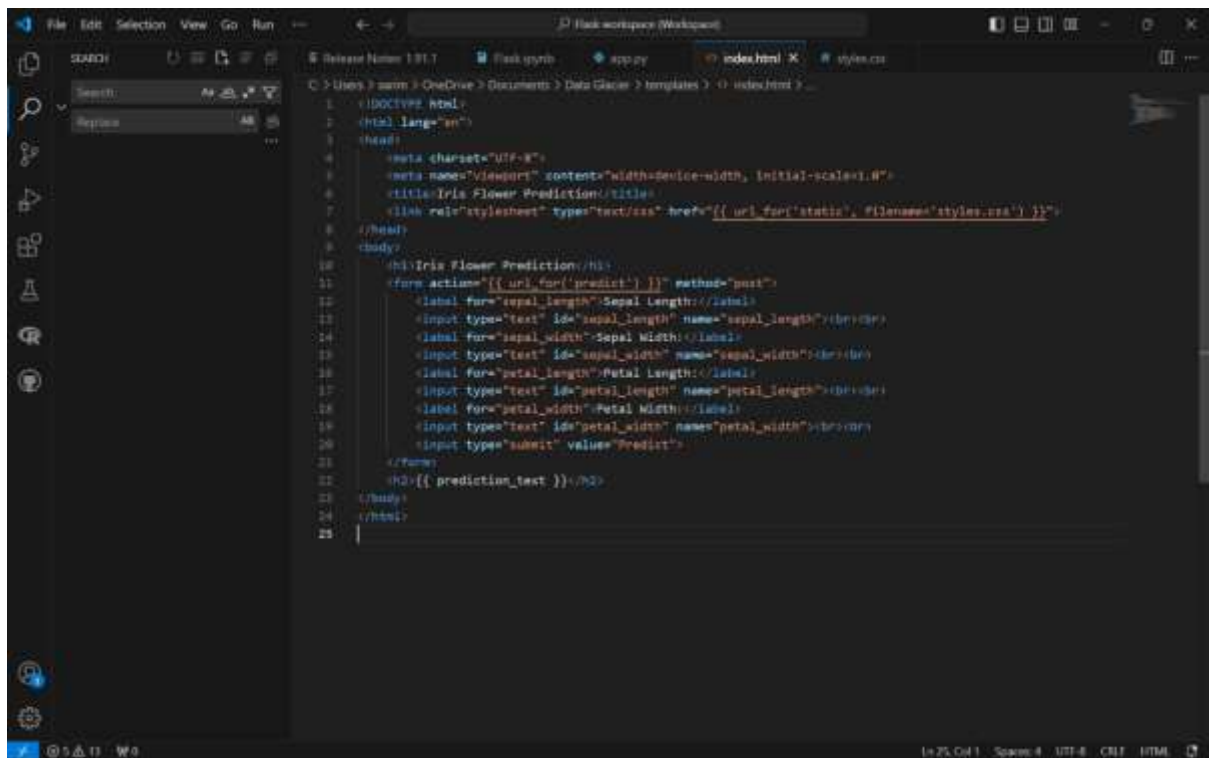
@app.route('/')
def home():
    return "Welcome to the Iris Model Prediction API!"

@app.route('/predict', methods=['POST'])
def predict():
    data = request.get_json(force=True)
    prediction = model.predict([np.array(data['features'])])
    return jsonify({'prediction': int(prediction[0])})

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

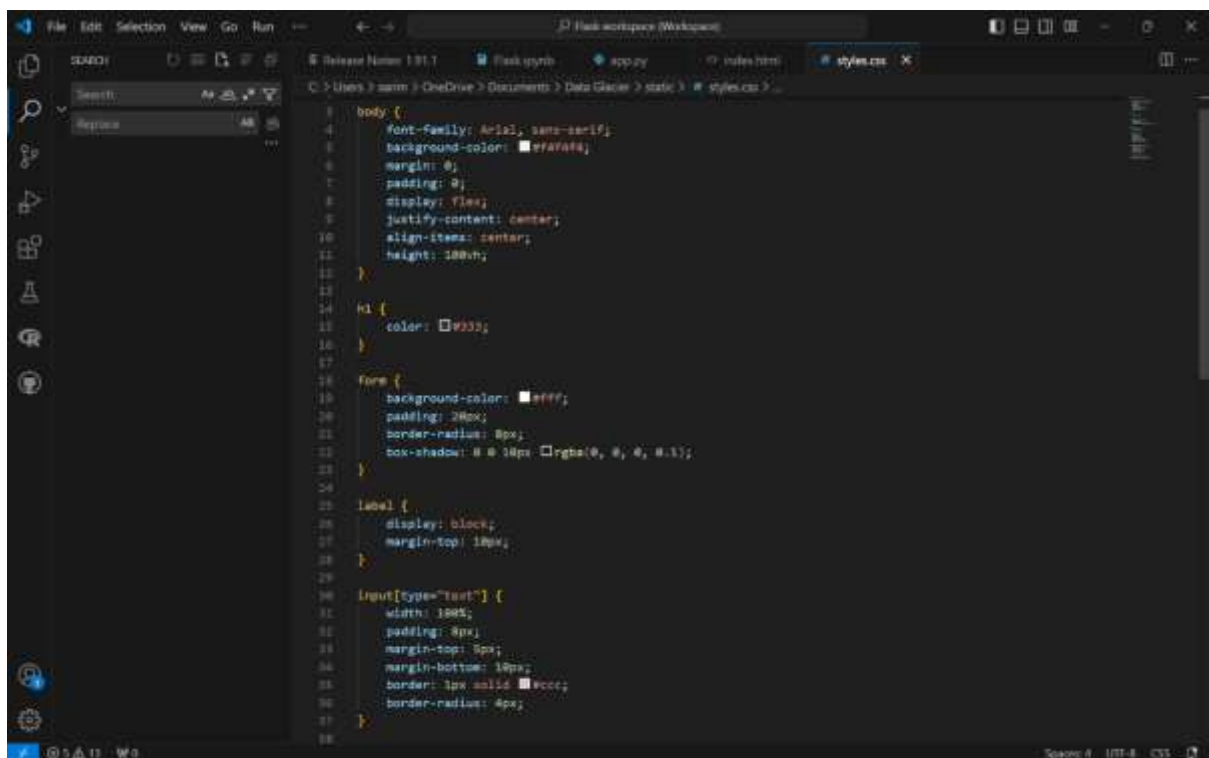
PS C:\Users\sarim\OneDrive\Documents\Data Glacie> python app.py  
\* Serving Flask app "app"  
\* Debug mode: on  
WARNING: This is a development server. Do not use it in a production deployment. Use a production WSGI server instead.  
\* Running on http://127.0.0.1:5000  
Press CTRL-C to quit  
\* Restarting with watchdog (winmmap)  
\* Debugger is active  
\* Debugger PIN: 225-645-264  
127.0.0.1 - - [27/Jul/2024 10:17:22] "GET / HTTP/1.1" 200 -  
\* Detected change in 'C:\Users\sarim\anaconda3\lib\site-packages\flask\blueprints.py', reloading  
\* Detected change in 'C:\Users\sarim\anaconda3\lib\site-packages\flask\ctx.py', reloading

## Creating index.html:



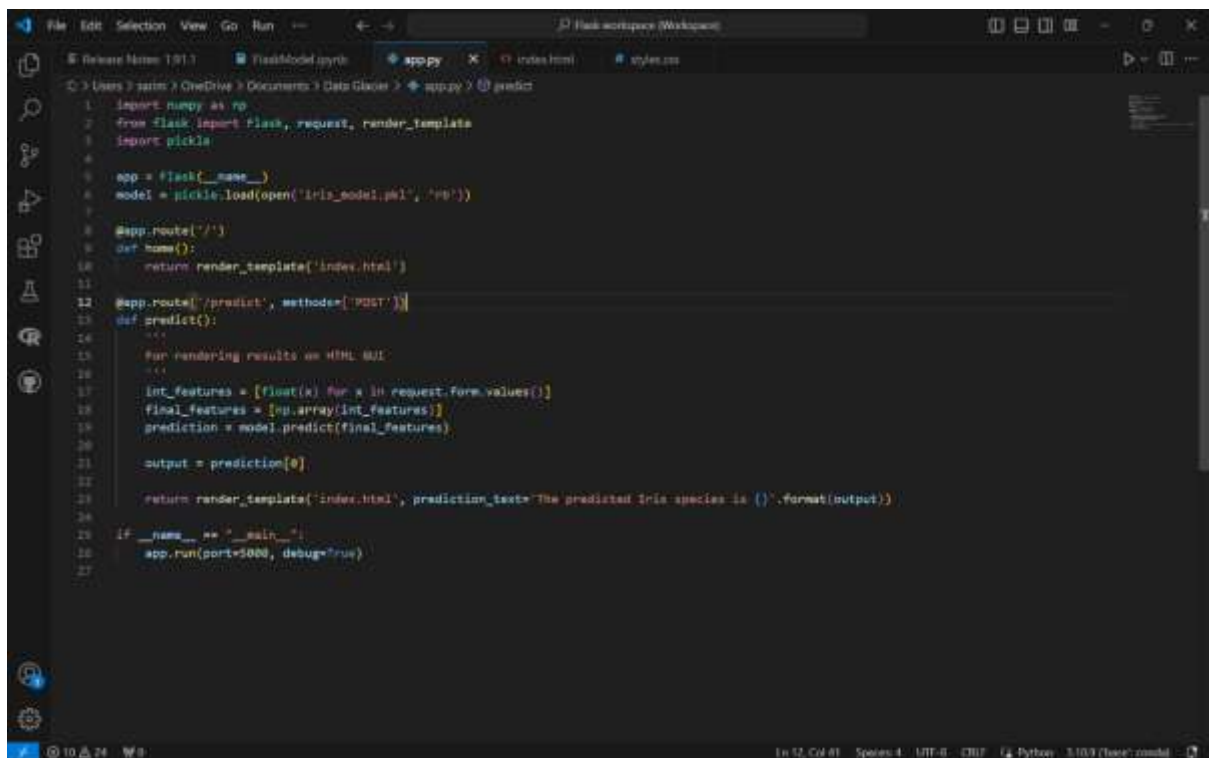
```
1 <!DOCTYPE html>
2 <html lang="en">
3 <head>
4   <meta charset="UTF-8">
5   <meta name="viewport" content="width=device-width, initial-scale=1.0">
6   <title>Iris Flower Prediction</title>
7   <link rel="stylesheet" type="text/css" href="{{ url_for('static', filename='style.css') }}">
8 </head>
9 <body>
10   <h1>Iris Flower Prediction</h1>
11   <form action="{{ url_for('predict') }}" method="post">
12     <label for="sepal_length">Sepal Length:</label>
13     <input type="text" id="sepal_length" name="sepal_length"></input>
14     <label for="sepal_width">Sepal Width:</label>
15     <input type="text" id="sepal_width" name="sepal_width"></input>
16     <label for="petal_length">Petal Length:</label>
17     <input type="text" id="petal_length" name="petal_length"></input>
18     <label for="petal_width">Petal Width:</label>
19     <input type="text" id="petal_width" name="petal_width"></input>
20     <input type="submit" value="Predict">
21   </form>
22   <h2>{{ prediction_text }}</h2>
23 </body>
24 </html>
25
```

## Creating style.css:



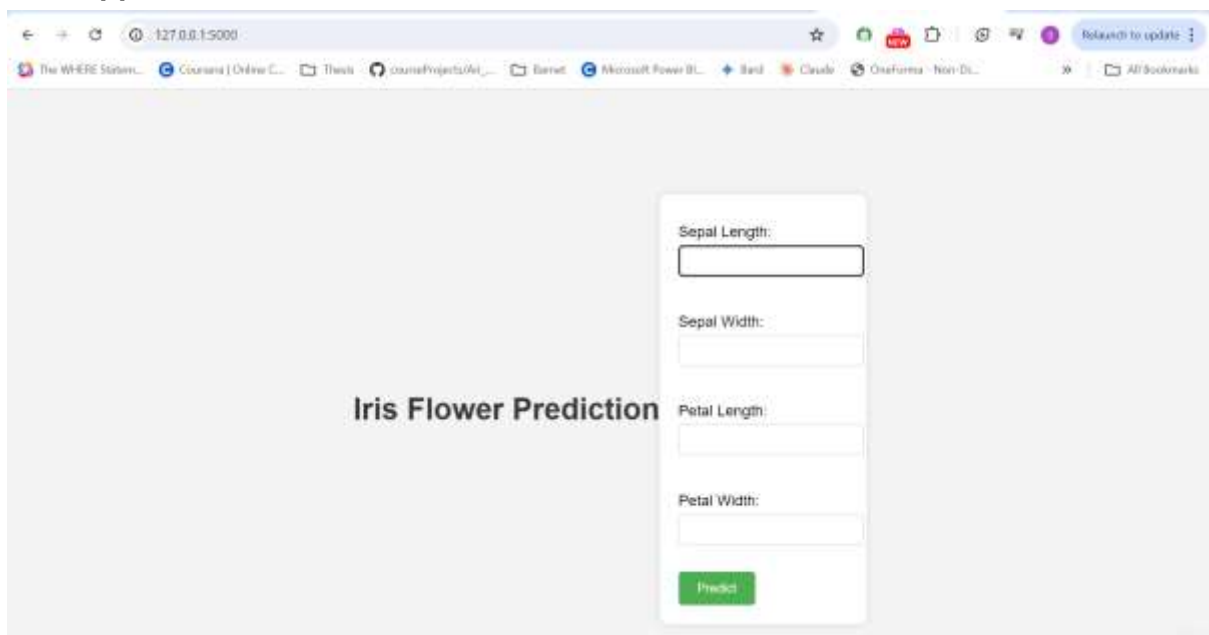
```
1 body {
2   font-family: Arial, sans-serif;
3   background-color: #e1f5fe;
4   margin: 0;
5   padding: 0;
6   display: flex;
7   justify-content: center;
8   align-items: center;
9   height: 100vh;
10 }
11
12 h1 {
13   color: #003366;
14 }
15
16 form {
17   background-color: #e1f5fe;
18   padding: 20px;
19   border-radius: 8px;
20   box-shadow: 0 0 10px #003366;
21 }
22
23 label {
24   display: block;
25   margin-top: 10px;
26 }
27
28 input[type="text"] {
29   width: 100%;
30   padding: 8px;
31   margin-top: 5px;
32   margin-bottom: 10px;
33   border: 1px solid #003366;
34   border-radius: 4px;
35 }
36
```

## Update Flask application:



```
1 import numpy as np
2 from flask import Flask, request, render_template
3 import pickle
4
5 app = Flask(__name__)
6 model = pickle.load(open('iris_model.pkl', 'rb'))
7
8 @app.route('/')
9 def home():
10     return render_template('index.html')
11
12 @app.route('/predict', methods=['POST'])
13 def predict():
14     """
15     For rendering results on HTML GUI
16     """
17     int_features = [float(x) for x in request.form.values()]
18     final_features = np.array(int_features)
19     prediction = model.predict(final_features)
20
21     output = prediction[0]
22
23     return render_template('index.html', prediction_text='The predicted Iris species is {}'.format(output))
24
25 if __name__ == '__main__':
26     app.run(port=5000, debug=True)
```

## Web Application:



The web application interface is displayed in a browser window. The title bar shows the address bar with the URL `127.0.0.1:5000`. The browser's address bar and tabs are visible at the top. The main content area has a light gray background. In the center, the text "Iris Flower Prediction" is displayed. To the right of the text, there is a white form with a green "Predict" button. The form contains four input fields labeled "Sepal Length:", "Sepal Width:", "Petal Length:", and "Petal Width:". The "Predict" button is green and located at the bottom of the form.

