

## Week 4 Assignment - Week4: Deployment on Flask

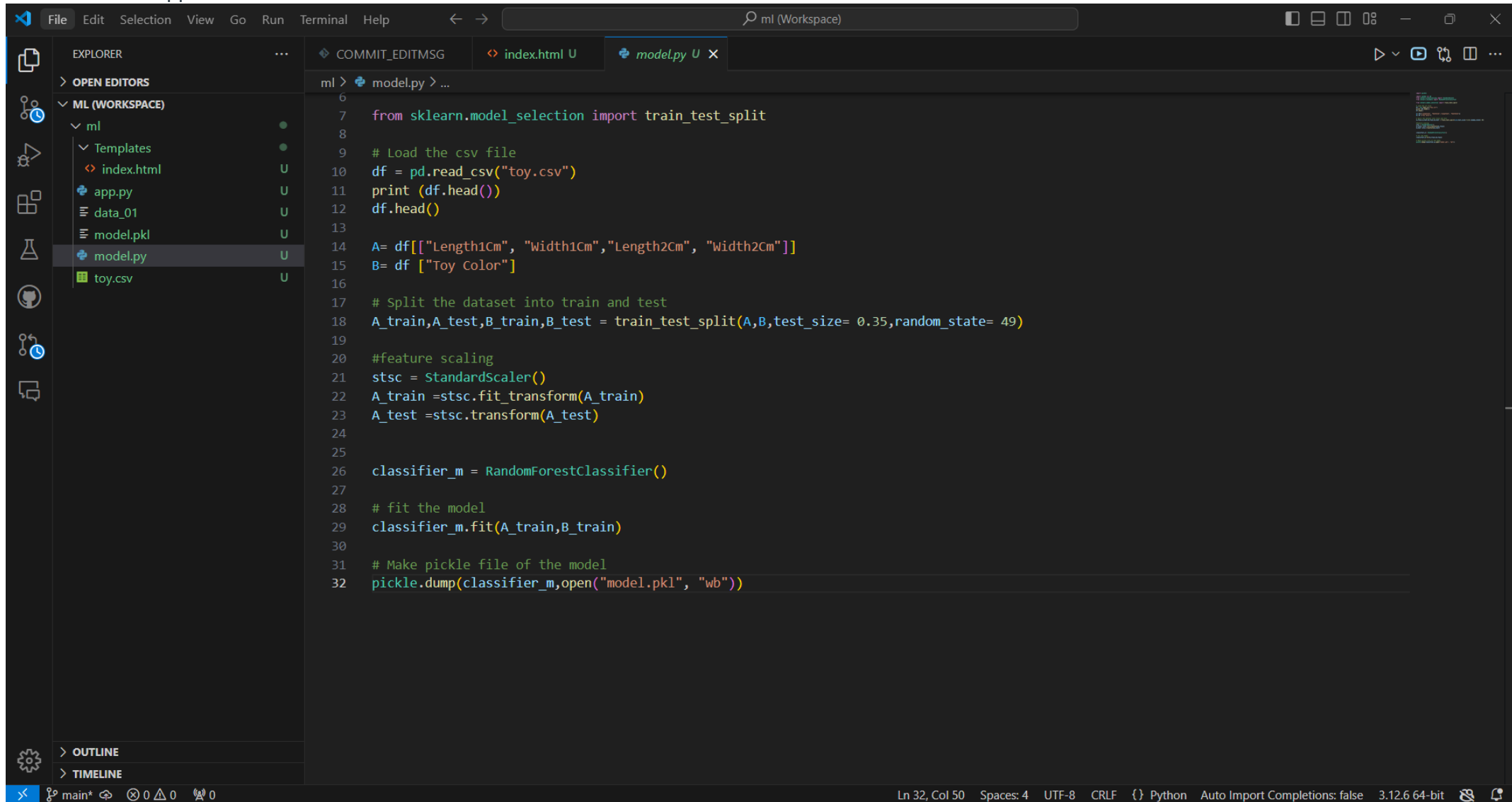
Name: Vijayarajan Vijaya Jothi

Batch code: LISUM37

Submission date: Sep 23, 2024

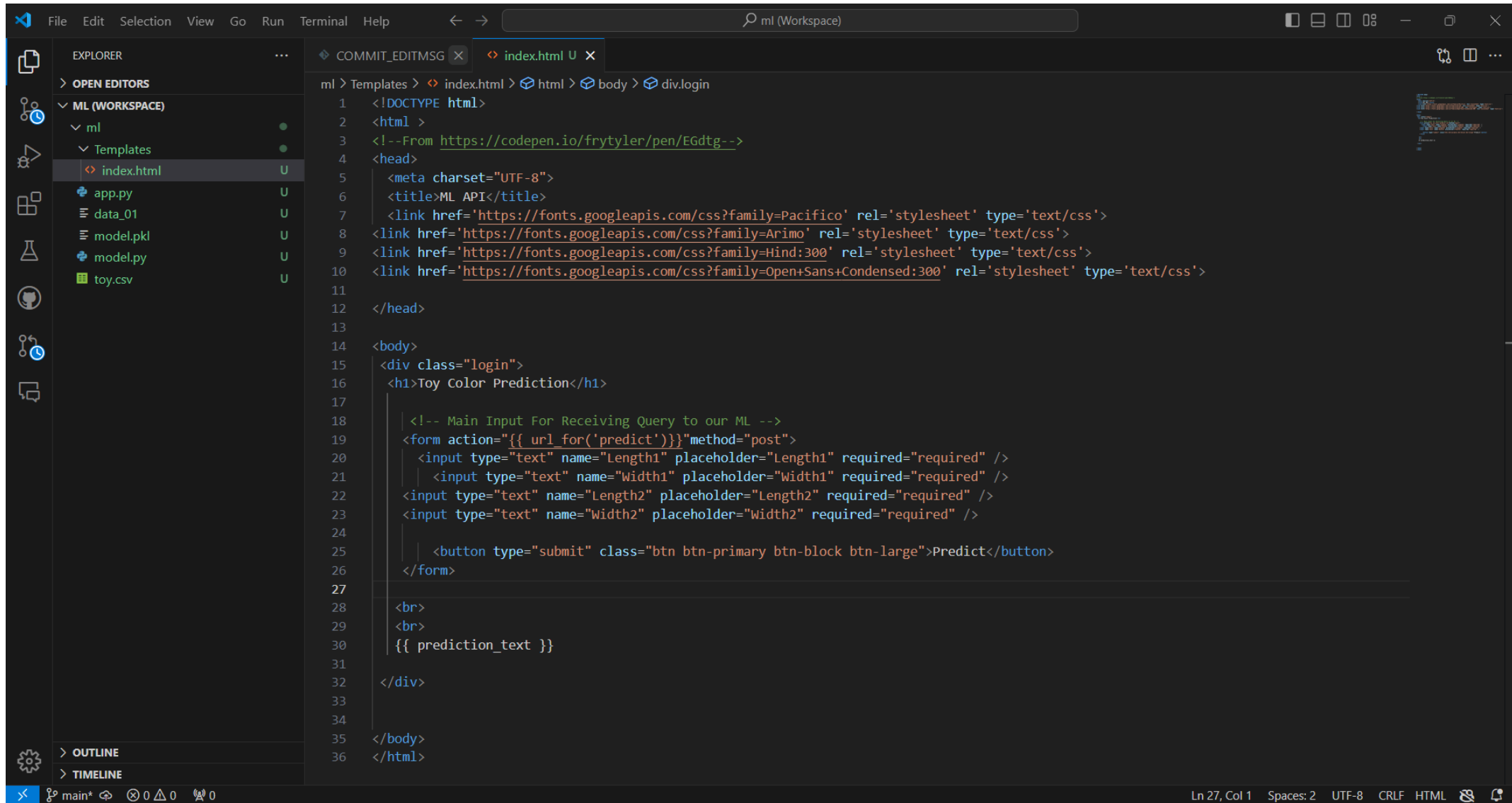
Submitted to: Data Glacier

### 1. Code of Flask app



```
6
7 from sklearn.model_selection import train_test_split
8
9 # Load the csv file
10 df = pd.read_csv("toy.csv")
11 print (df.head())
12 df.head()
13
14 A= df[["Length1Cm", "Width1Cm","Length2Cm", "Width2Cm"]]
15 B= df ["Toy Color"]
16
17 # Split the dataset into train and test
18 A_train,A_test,B_train,B_test = train_test_split(A,B,test_size= 0.35,random_state= 49)
19
20 #feature scaling
21 stsc = StandardScaler()
22 A_train =stsc.fit_transform(A_train)
23 A_test =stsc.transform(A_test)
24
25
26 classifier_m = RandomForestClassifier()
27
28 # fit the model
29 classifier_m.fit(A_train,B_train)
30
31 # Make pickle file of the model
32 pickle.dump(classifier_m,open("model.pkl", "wb"))
```

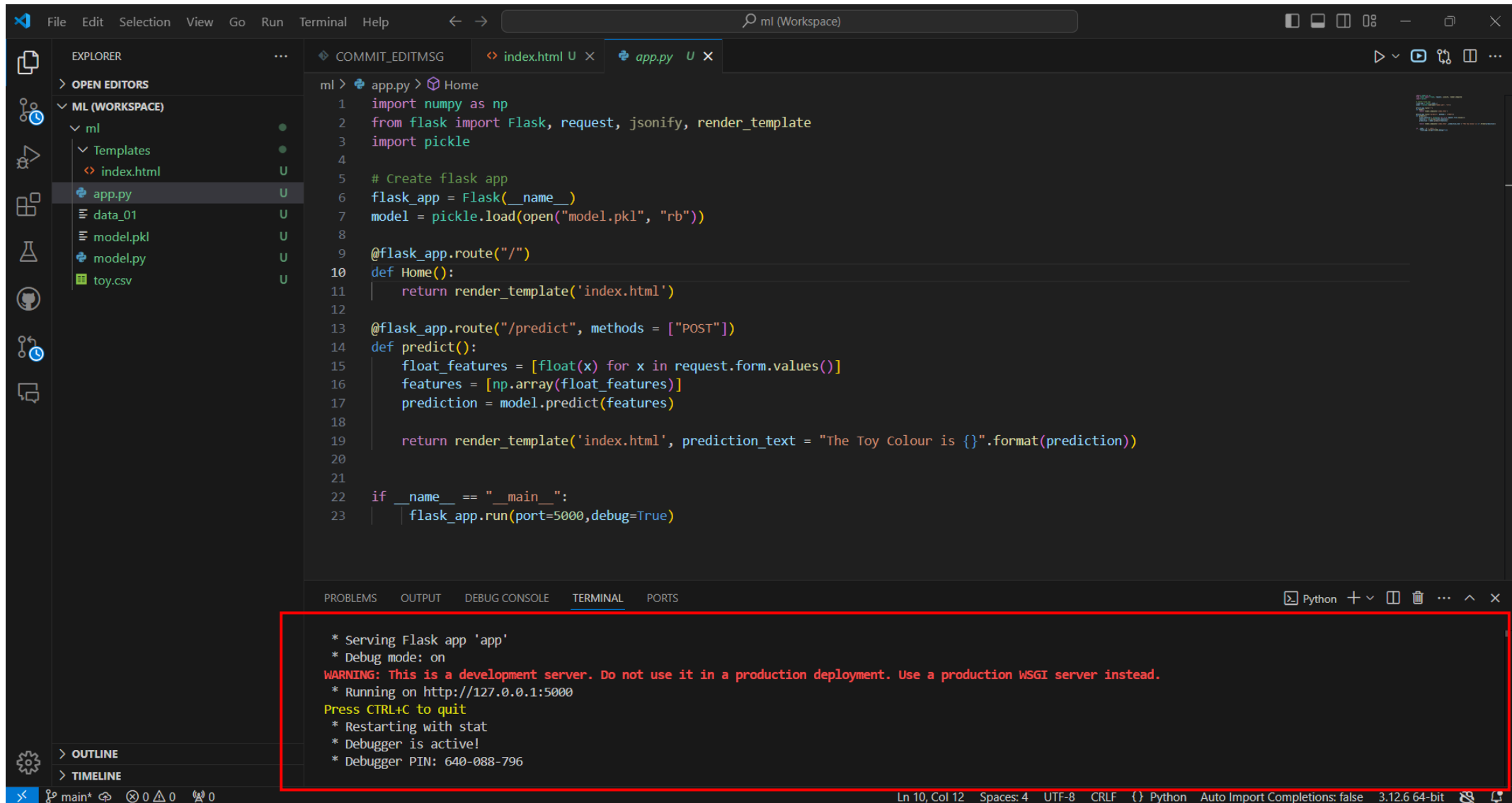
## 2. index.html



The image shows a Visual Studio Code editor window with a workspace named 'ml'. The Explorer sidebar on the left shows the file structure: 'ML (WORKSPACE)' containing a folder 'ml' which has a subfolder 'Templates' and a file 'index.html'. Other files in the workspace include 'app.py', 'data\_01', 'model.pkl', 'model.py', and 'toy.csv'. The 'index.html' file is open in the editor, showing HTML code for a login form. The code includes a DOCTYPE declaration, a title 'ML API', and several links to Google Fonts (Pacifico, Arimo, Hind, and Open+Sans+Condensed). The form has four text inputs for 'Length1', 'Width1', 'Length2', and 'Width2', and a 'Predict' button. A placeholder for the prediction result is also present. The status bar at the bottom indicates the cursor is at line 27, column 1, with 2 spaces, UTF-8 encoding, and CRLF line endings.

```
1 <!DOCTYPE html>
2 <html>
3 <!--From https://codepen.io/frytyler/pen/EGdtg-->
4 <head>
5   <meta charset="UTF-8">
6   <title>ML API</title>
7   <link href='https://fonts.googleapis.com/css?family=Pacifico' rel='stylesheet' type='text/css'>
8   <link href='https://fonts.googleapis.com/css?family=Arimo' rel='stylesheet' type='text/css'>
9   <link href='https://fonts.googleapis.com/css?family=Hind:300' rel='stylesheet' type='text/css'>
10  <link href='https://fonts.googleapis.com/css?family=Open+Sans+Condensed:300' rel='stylesheet' type='text/css'>
11
12 </head>
13
14 <body>
15   <div class="login">
16     <h1>Toy Color Prediction</h1>
17
18     <!-- Main Input For Receiving Query to our ML -->
19     <form action="{{ url_for('predict')}}" method="post">
20       <input type="text" name="Length1" placeholder="Length1" required="required" />
21       <input type="text" name="Width1" placeholder="Width1" required="required" />
22       <input type="text" name="Length2" placeholder="Length2" required="required" />
23       <input type="text" name="Width2" placeholder="Width2" required="required" />
24
25       <button type="submit" class="btn btn-primary btn-block btn-large">Predict</button>
26     </form>
27
28     <br>
29     <br>
30     {{ prediction_text }}
31
32   </div>
33
34
35 </body>
36 </html>
```

### 3. Running Flask in debug mode in VisualStudioCode



4. Running app homepage

GitHub Dashboard

×

ML API

×

+

←

↺

i

127.0.0.1:5000

Toy Color Prediction

Length1

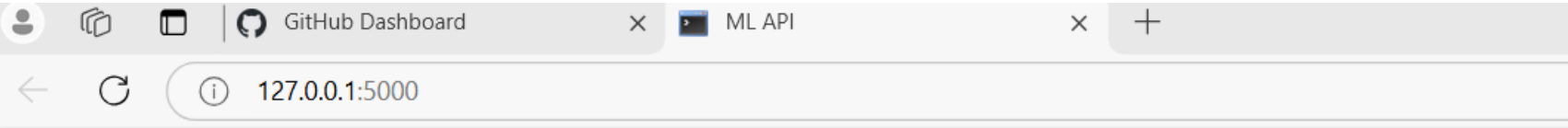
Width1

Length2

Width2

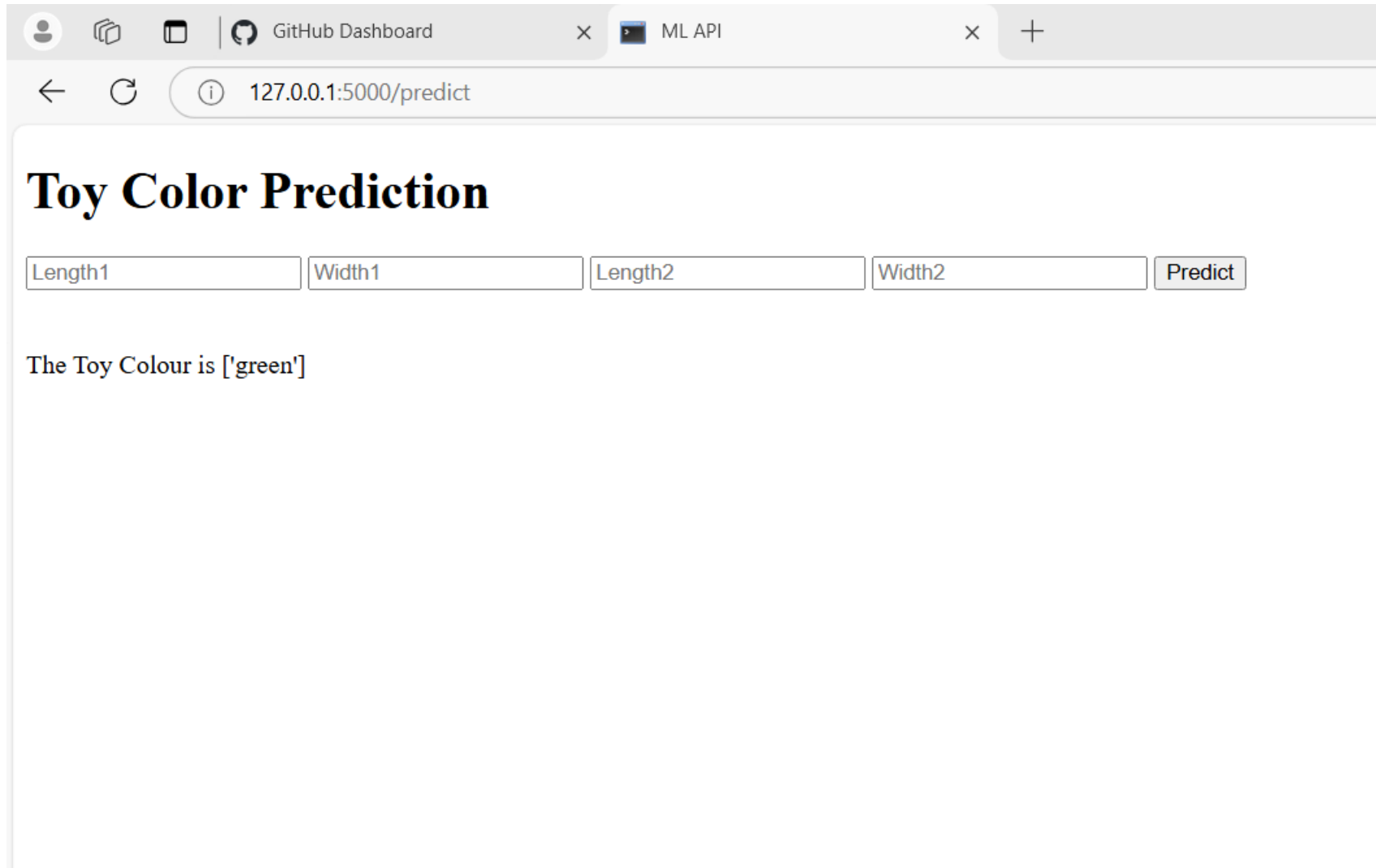
Predict

5. Filling test data



# Toy Color Prediction

6. Prediction result shown on web page



The screenshot shows a web browser window with two tabs: 'GitHub Dashboard' and 'ML API'. The address bar displays '127.0.0.1:5000/predict'. The page title is 'Toy Color Prediction'. Below the title, there are four input fields labeled 'Length1', 'Width1', 'Length2', and 'Width2', followed by a 'Predict' button. The output of the prediction is displayed as 'The Toy Colour is ['green']'.

**Toy Color Prediction**

Length1 Width1 Length2 Width2 Predict

The Toy Colour is ['green']

## 7. Flask output in terminal window

The screenshot displays the Visual Studio Code interface with a workspace named "ml". The Explorer sidebar on the left shows the project structure:

- ML (WORKSPACE)
  - ml
    - Templates
      - index.html
    - app.py
    - data\_01
    - model.pkl
    - model.py
    - toy.csv

The main editor area shows the code for `app.py`:

```
1 import numpy as np
2 from flask import Flask, request, jsonify, render_template
3 import pickle
4
5 # Create flask app
6 flask_app = Flask(__name__)
7 model = pickle.load(open("model.pkl", "rb"))
8
9 @flask_app.route("/")
10 def Home():
11     return render_template('index.html')
12
13 @flask_app.route("/predict", methods = ["POST"])
14 def predict():
15     float_features = [float(x) for x in request.form.values()]
16     features = [np.array(float_features)]
17     prediction = model.predict(features)
18
19     return render_template('index.html', prediction_text = "The Toy Colour is {}".format(prediction))
20
21
22 if __name__ == "__main__":
23     flask_app.run(port=5000, debug=True)
```

The TERMINAL panel at the bottom shows the output of the application:

```
Press CTRL+C to quit
* Restarting with stat
* Debugger is active!
* Debugger PIN: 640-088-796
127.0.0.1 - - [20/Sep/2024 16:36:16] "GET / HTTP/1.1" 200 -
127.0.0.1 - - [20/Sep/2024 16:36:18] "GET / HTTP/1.1" 200 -
127.0.0.1 - - [20/Sep/2024 16:37:16] "GET / HTTP/1.1" 200 -
127.0.0.1 - - [20/Sep/2024 16:38:06] "POST /predict HTTP/1.1" 200 -
127.0.0.1 - - [20/Sep/2024 16:39:21] "POST /predict HTTP/1.1" 200 -
127.0.0.1 - - [20/Sep/2024 16:39:33] "GET / HTTP/1.1" 200 -
127.0.0.1 - - [20/Sep/2024 16:39:34] "GET / HTTP/1.1" 200 -
```