

Jennifer Turley

Data Glacier Internship

Cohort LISUM05

Feb 15, 2022

Uploaded to https://github.com/jenturley27/IRIS_Classifier

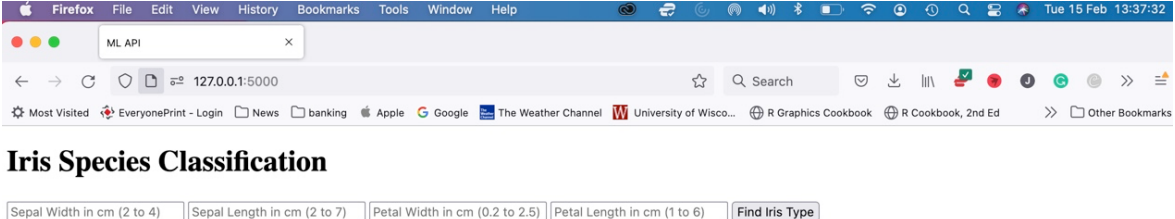
Deployment on Flask

This app uses the iris data set from the UCI Machine Learning Repository [<http://archive.ics.uci.edu/ml>] (the credit for the data is on the home screen for the app).

At the above github link is the code for this project.

I ran the app.py on Spyder, with the following results:

1. At the specified browser address, here is the screen for inputting iris measurements:

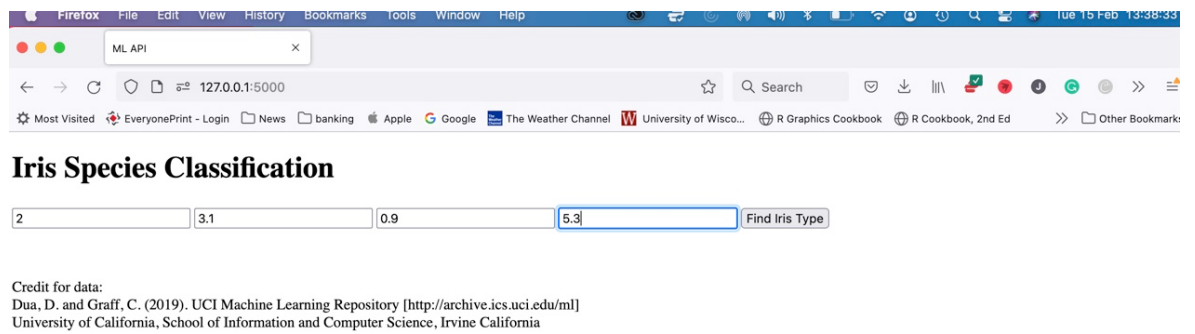


The screenshot shows a Firefox browser window with the address bar set to 127.0.0.1:5000. The page title is "ML API". The main heading is "Iris Species Classification". Below the heading, there are four input fields for iris measurements: "Sepal Width in cm (2 to 4)", "Sepal Length in cm (2 to 7)", "Petal Width in cm (0.2 to 2.5)", and "Petal Length in cm (1 to 6)". A "Find Iris Type" button is located to the right of these fields. The browser's bookmark bar is visible at the bottom, showing various links like "EveryonePrint - Login", "News", "banking", "Apple", "Google", "The Weather Channel", "University of Wisco...", "R Graphics Cookbook", and "R Cookbook, 2nd Ed".

Credit for data:

Dua, D. and Graff, C. (2019). UCI Machine Learning Repository [<http://archive.ics.uci.edu/ml>]
University of California, School of Information and Computer Science, Irvine California

2. Here are sample values entered, with 'Find iris Type' not yet selected:

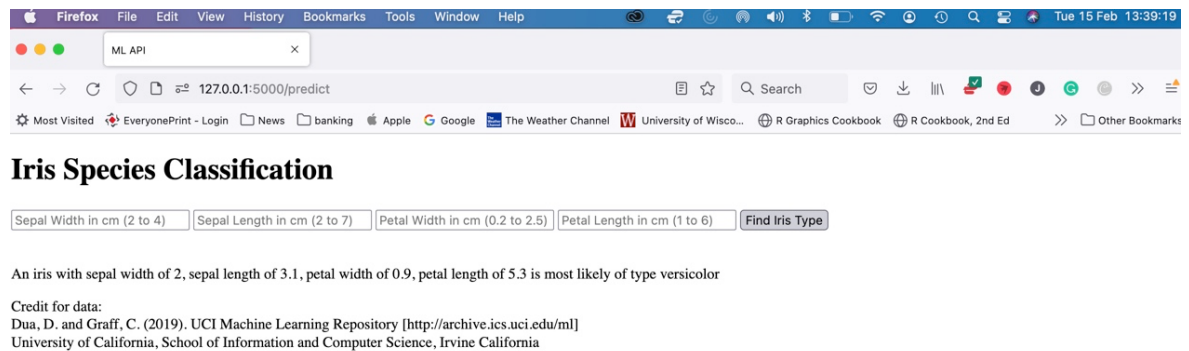


Iris Species Classification

2 3.1 0.9 5.3 Find iris Type

Credit for data:
Dua, D. and Graff, C. (2019). UCI Machine Learning Repository [<http://archive.ics.uci.edu/ml>]
University of California, School of Information and Computer Science, Irvine California

3. Results:



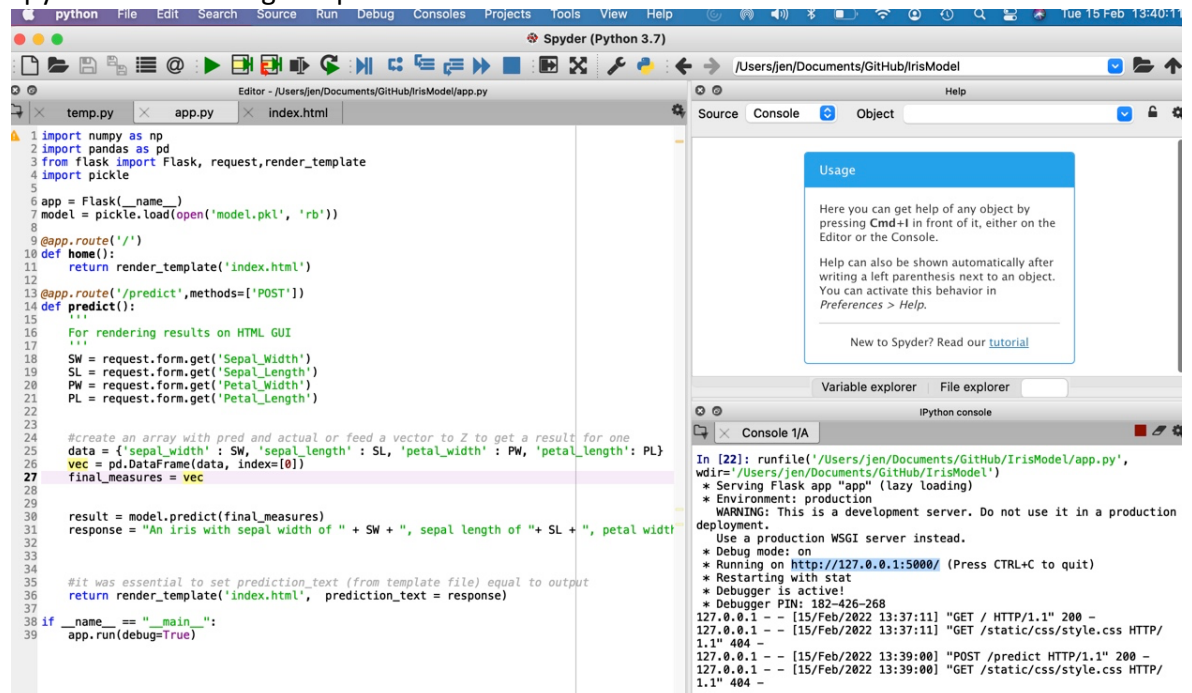
Iris Species Classification

Sepal Width in cm (2 to 4) Sepal Length in cm (2 to 7) Petal Width in cm (0.2 to 2.5) Petal Length in cm (1 to 6) Find iris Type

An iris with sepal width of 2, sepal length of 3.1, petal width of 0.9, petal length of 5.3 is most likely of type versicolor

Credit for data:
Dua, D. and Graff, C. (2019). UCI Machine Learning Repository [<http://archive.ics.uci.edu/ml>]
University of California, School of Information and Computer Science, Irvine California

4. Spyder screen during this process:



The screenshot displays the Spyder Python IDE interface. The main editor window shows a Python script named `app.py` located at `/Users/jen/Documents/GitHub/IrisModel/app.py`. The code is a Flask web application that loads a pre-trained model and provides a simple web interface for predictions. The code includes imports for `numpy`, `pandas`, `Flask`, `request`, `render_template`, and `pickle`. It defines a `home` route and a `predict` route. The `predict` route takes form data for sepal width, sepal length, petal width, and petal length, processes it, and returns a prediction message. The application is run in debug mode.

```
1 import numpy as np
2 import pandas as pd
3 from flask import Flask, request, render_template
4 import pickle
5
6 app = Flask(__name__)
7 model = pickle.load(open('model.pkl', 'rb'))
8
9 @app.route('/')
10 def home():
11     return render_template('index.html')
12
13 @app.route('/predict', methods=['POST'])
14 def predict():
15     """
16     For rendering results on HTML GUI
17     """
18     SW = request.form.get('Sepal_Width')
19     SL = request.form.get('Sepal_Length')
20     PW = request.form.get('Petal_Width')
21     PL = request.form.get('Petal_Length')
22
23
24     #create an array with pred and actual or feed a vector to Z to get a result for one
25     data = {'sepal_width': SW, 'sepal_length': SL, 'petal_width': PW, 'petal_length': PL}
26     vec = pd.DataFrame(data, index=[0])
27     final_measures = vec
28
29     result = model.predict(final_measures)
30     response = "An iris with sepal width of " + SW + ", sepal length of " + SL + ", petal width"
31
32
33
34
35     #it was essential to set prediction_text (from template file) equal to output
36     return render_template('index.html', prediction_text = response)
37
38 if __name__ == "__main__":
39     app.run(debug=True)
```

The right-hand side of the interface shows the IPython console. It displays the output of the `runfile` command, indicating that the Flask application is running on `http://127.0.0.1:5000/`. The console also shows several GET and POST requests being handled by the application, including requests for the static CSS file.

```
In [22]: runfile('/Users/jen/Documents/GitHub/IrisModel/app.py',
wdir='/Users/jen/Documents/GitHub/IrisModel')
* Serving Flask app "app" (lazy loading)
* Environment: production
WARNING: This is a development server. Do not use it in a production
deployment.
Use a production WSGI server instead.
* Debug mode: on
* Running on http://127.0.0.1:5000/ (Press CTRL+C to quit)
* Restarting with stat
* Debugger is active!
* Debugger PIN: 182-426-268
127.0.0.1 - - [15/Feb/2022 13:37:11] "GET / HTTP/1.1" 200 -
127.0.0.1 - - [15/Feb/2022 13:37:11] "GET /static/css/style.css HTTP/
1.1" 404 -
127.0.0.1 - - [15/Feb/2022 13:39:00] "POST /predict HTTP/1.1" 200 -
127.0.0.1 - - [15/Feb/2022 13:39:00] "GET /static/css/style.css HTTP/
1.1" 404 -
```