# Iris Model Deployment

June 24, 2024


### 0.1 Carmelo R. Casiraro


### 0.2 BATCH CODE: LISUM34

### 0.3 SUBMIT DATE: 06/23/24

### 0.4 Model Deployment Using Flask- Carmelo R. Casiraro, USA- Batch: LISUM34 - Week #4 Assignment- Data Glacier Internship

Step 1- Pick Iris Toy Data Set

| | Sepal_Length | Sepal_Width | Petal_Length | Petal_Width | Class |
|---|---|---|---|---|---|
| 1 | 5.1 | 3.5 | 1.4 | 0.2 | Setosa |
| 2 | 4.9 | 3 | 1.4 | 0.2 | Setosa |
| 3 | 4.7 | 3.2 | 1.3 | 0.2 | Setosa |
| 4 | 4.6 | 3.1 | 1.5 | 0.2 | Setosa |
| 5 | 5 | 3.6 | 1.4 | 0.2 | Setosa |
| 6 | 5.4 | 3.9 | 1.7 | 0.4 | Setosa |
| 7 | 4.6 | 3.4 | 1.4 | 0.3 | Setosa |
| 8 | 5 | 3.4 | 1.5 | 0.2 | Setosa |
| 9 | 4.4 | 2.9 | 1.4 | 0.2 | Setosa |
| 10 | 4.9 | 3.1 | 1.5 | 0.1 | Setosa |
| 11 | 5.4 | 3.7 | 1.5 | 0.2 | Setosa |
| 12 | 4.8 | 3.4 | 1.6 | 0.2 | Setosa |
| 13 | 4.8 | 3 | 1.4 | 0.1 | Setosa |
| 14 | 4.3 | 3 | 1.1 | 0.1 | Setosa |
| 15 | 5.8 | 4 | 1.2 | 0.2 | Setosa |
| 16 | 5.7 | 4.4 | 1.5 | 0.4 | Setosa |
| 17 | 5.4 | 3.9 | 1.3 | 0.4 | Setosa |
| 18 | 5.1 | 3.5 | 1.4 | 0.3 | Setosa |
| 19 | 5.7 | 3.8 | 1.7 | 0.3 | Setosa |
| 20 | 5.1 | 3.8 | 1.5 | 0.3 | Setosa |
| 21 | 5.4 | 3.4 | 1.7 | 0.2 | Setosa |
| 22 | 5.1 | 3.7 | 1.5 | 0.4 | Setosa |

Step 2- Pre Processing & Modeling Import Libraries

[7]:
```
import pandas as pd
from sklearn.ensemble import RandomForestClassifier
from sklearn.model_selection import train_test_split
```

```
from sklearn.metrics import accuracy_score
import pickle
from flask import Flask, render_template, request
```

[8]:
```
data = pd.read_csv('iris.csv')
data
```

[8]:

| | Sepal_Length | Sepal_Width | Petal_Length | Petal_Width | Class |
|---|---|---|---|---|---|
| 0 | 5.1 | 3.5 | 1.4 | 0.2 | Setosa |
| 1 | 4.9 | 3.0 | 1.4 | 0.2 | Setosa |
| 2 | 4.7 | 3.2 | 1.3 | 0.2 | Setosa |
| 3 | 4.6 | 3.1 | 1.5 | 0.2 | Setosa |
| 4 | 5.0 | 3.6 | 1.4 | 0.2 | Setosa |
| .. | … | … | … | … | … |
| 145 | 6.7 | 3.0 | 5.2 | 2.3 | Virginica |
| 146 | 6.3 | 2.5 | 5.0 | 1.9 | Virginica |
| 147 | 6.5 | 3.0 | 5.2 | 2.0 | Virginica |
| 148 | 6.2 | 3.4 | 5.4 | 2.3 | Virginica |
| 149 | 5.9 | 3.0 | 5.1 | 1.8 | Virginica |

[150 rows x 5 columns]

Step 3- Split The Data

[10]:
```
X = data.drop('Class', axis=1)
y = data['Class']
x_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,␣
↪random_state=42)
```

Step 4- Train Model

[12]:
```
model = RandomForestClassifier(n_estimators=100, random_state=42)
model.fit(x_train, y_train)
```

[12]:
```
RandomForestClassifier(random_state=42)
```

Step 5- Analyze Model

[14]:
```
y_pred = model.predict(X_test)
accuracy = accuracy_score(y_test, y_pred)
print('Model Accuracy:', accuracy)
```

```
Model Accuracy: 1.0
```

Step 6- Create PKL file

[16]:
```
with open('model.pkl', 'wb') as file:
    pickle.dump(model, file)
```

[17]:
```
import sklearn
print(sklearn.__version__)
```

```
1.5.0
```

[18]:
```
import sys
!{sys.executable} -m pip install scikit-learn==1.5.0
```

```
Defaulting to user installation because normal site-packages is not writeable
Looking in links: /usr/share/pip-wheels
Requirement already satisfied: scikit-learn==1.5.0 in
./.local/lib/python3.10/site-packages (1.5.0)
Requirement already satisfied: numpy>=1.19.5 in
/opt/conda/envs/anaconda-2024.02-py310/lib/python3.10/site-packages (from
scikit-learn==1.5.0) (1.26.4)
Requirement already satisfied: scipy>=1.6.0 in
/opt/conda/envs/anaconda-2024.02-py310/lib/python3.10/site-packages (from
scikit-learn==1.5.0) (1.12.0)
Requirement already satisfied: joblib>=1.2.0 in
/opt/conda/envs/anaconda-2024.02-py310/lib/python3.10/site-packages (from
scikit-learn==1.5.0) (1.2.0)
Requirement already satisfied: threadpoolctl>=3.1.0 in
./.local/lib/python3.10/site-packages (from scikit-learn==1.5.0) (3.5.0)
```

## 0.5 Step 7- Created Html and CSS files for Web Page Deployment

## 0.6 Step 7.1- index.html file- input the values to predict

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Iris Identifier</title>
    <link rel="stylesheet" href="{{ url_for('static', filename='styles.css') }}">
</head>
<body>
    <div class="container">
        <h1>Iris Identifier</h1>
        <form action="{{ url_for('predict') }}" method="POST">
            <div class="form-group">
                <label for="sepal-length">Sepal Length</label>
                <input type="number" min="0.0" max="10.0" step="0.1" id="sepal-length" name="sepal-length" required>            </div>
            <div class="form-group">
                <label for="sepal-width">Sepal Width</label>
                <input type="number" min="0.0" max="10.0" step="0.1" id="sepal-width" name="sepal-width" required>            </div>
            <div class="form-group">
                <label for="petal-length">Petal Length</label>
                <input type="number" min="0.0" max="10.0" step="0.1" id="petal-length" name="petal-length" required>            </div>
            <div class="form-group">
                <label for="petal-width">Petal Width</label>
                <input type="number" min="0.0" max="10.0" step="0.1" id="petal-width" name="petal-width" required>            </div>
            <button type="submit">Submit</button>
        </form>
    </div>
</body>
</html>
```

## 0.7 Step 7.2- result.html file- this shows the predicted results

```
<!doctype html>
<html lang="en">
  <head>
    <meta charset="utf-8">
    <title>Prediction Result</title>
    <link rel="stylesheet" href="{{ url_for('static', filename='styles.css') }}">
  </head>
  <body>
    <div class="prediction-result">
        <h1>Prediction Result</h1>
        <p>Predicted class: {{ prediction }}</p>
        <a href="{{ url_for('home') }}">Go back</a>
    </div>
  </body>
</html>
```

## 0.8 Step 7.3- styles.css file- this formats the web page

```css
body {
    font-family: Arial, sans-serif;
    background-color: #f4f4f4;
    margin: 0;
    padding: 0;
    display: flex;
    justify-content: center;
    align-items: center;
    height: 100vh;
}

.container {
    background-color: #fff;
    padding: 20px;
    border-radius: 8px;
    box-shadow: 0 0 10px rgba(0, 0, 0, 0.1);
    width: 300px;
}

h1 {
    text-align: center;
    color: #333;
    margin-bottom: 20px;
}

.form-group {
    margin-bottom: 15px;
}

.form-group label {
    display: block;
    margin-bottom: 5px;
    color: #555;
}

.form-group input {
    width: 100%;
    padding: 8px;
    box-sizing: border-box;
    border: 1px solid #ccc;
    border-radius: 4px;
}

button {
    width: 100%;
    padding: 10px;
    background-color: #ff7500;
    border: none;
    border-radius: 4px;
    color: white;
    font-size: 16px;
    cursor: pointer;
}

button:hover {
```

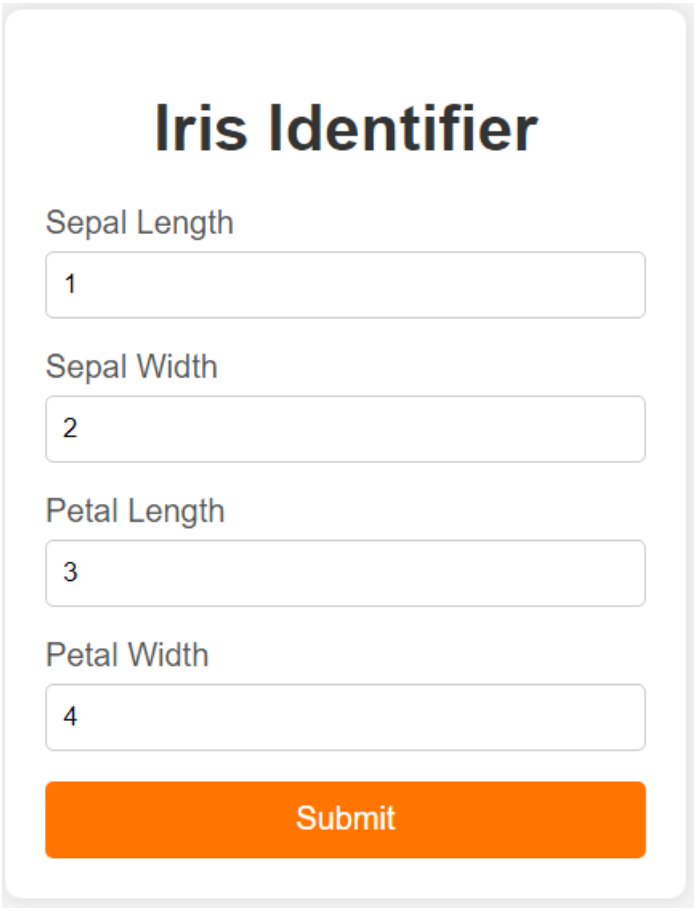
## 0.9 Step 8- Create a Flask Application

```python
from flask import Flask, redirect, url_for, render_template, request
import pickle

app = Flask(__name__)

with open('model.pkl', 'rb') as model_file:
    model = pickle.load(model_file)

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

@app.route('/predict', methods=['POST'])
def predict():
    sepal_length = float(request.form['sepal-length'])
    sepal_width = float(request.form['sepal-width'])
    petal_length = float(request.form['petal-length'])
    petal_width = float(request.form['petal-width'])

    # Model prediction
    features = [[sepal_length, sepal_width, petal_length, petal_width]]
    prediction = model.predict(features)[0]

    return render_template('result.html', prediction=prediction)

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

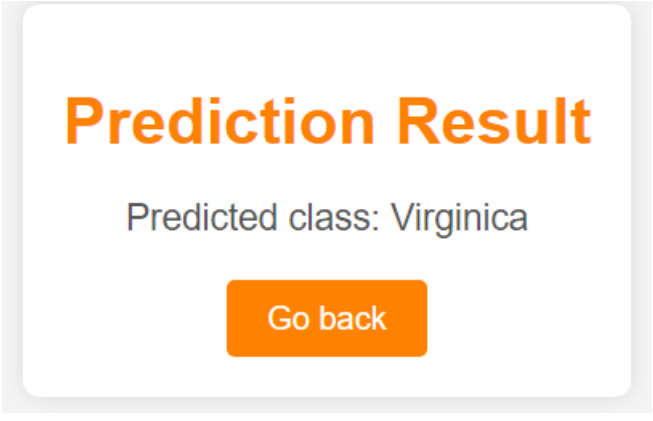
## 0.10 Step 9- Deploy Application Using Command Prompt

```
cralp@JESUSSAVES CLANGARM64 ~/OneDrive/Desktop/DATA_GLACIER_INTERNSHIP (master)
$ cd Week#4_Flask_Assignment

cralp@JESUSSAVES CLANGARM64 ~/OneDrive/Desktop/DATA_GLACIER_INTERNSHIP/Week#4_Flask_Assignment (master)
$ 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: 934-605-373
```

### 0.11 Step 10- Web Page Test




### 0.12 Step 11- Predicted Result



### 0.13 Summary of Project

1 - This project involved creating a machine learning model to predict the species of a Iris flower

2 based on sepal and petal dimensions.

3 - The Iris data set was used to train and test the model.

4 - The model was built using Python sklearn library.

5 - A flask application was created allowing users to input different numbers for

6 dimensions to predict the type of flower.

7 - Then, html files were created. First file was index.html file to inputing values, the

8 second file was a results.html file to predict the results.

9 - The application was then deployed using Visual studio code using command prompt.

10 - After deployment, user inputed values for dimensions for flower then clicked Submit.

11 - The result of the submission, showed the flower predicted as Virginica.

12 In conclusion, this project shows how to create a machine learning model & web application,

13 then, finally deploying it all using a Flask application.

[ ]: