

Week 4 – Deployment on Flask

Name: Archana Devi Ramesh

Batch code: LISUM16

Submission date: 28th December 2022

Submitted to: Data Glacier

Submission Link: <https://github.com/ArchanaDeviRamesh/Data-Glacier/tree/main/Week4>

Flask Deployment steps

1. Dataset used – Iris Flower detection dataset

<https://archive.ics.uci.edu/ml/datasets/Iris>

2. Open Visual Studio IDE, create a virtual environment and activate it

```
conda create -p venv python==3.8
```

3. Add the required libraries in a requirements.txt file and install the packages

```
pip install -r requirements.txt
```

4. Create a notebook or python file to train the Iris dataset

- a. Import the required libraries

```
#import required libraries
import pandas as pd
from sklearn.preprocessing import StandardScaler
from sklearn.ensemble import RandomForestClassifier
from sklearn.model_selection import train_test_split
import pickle
```

Python

- b. Reading the dataset

```
df = pd.read_csv("iris.csv")
df.head()
```

Python

	sepal.length	sepal.width	petal.length	petal.width	variety
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

- c. Creating X and y train and test

```
# Select independent and dependent variable
X = df[["sepal.length", "sepal.width", "petal.length", "petal.width"]]
y = df["variety"]

# Split the dataset into train and test
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=50)
```

Python

d. Feature scaling of the features

```
# Feature scaling
sc = StandardScaler()
X_train = sc.fit_transform(X_train)
X_test= sc.transform(X_test)
```

Python

e. Training the model using RandomForestClassifier

```
# Instantiate the model
model = RandomForestClassifier()
# Fit the model
model.fit(X_train, y_train)
```

Python

f. Saving the model using pickle in write mode

```
# Make pickle file of our model
pickle.dump(model, open("iris_model.pkl", "wb"))
```

Python

5. Create an app.py file to deploy the model on server using flask

a. Import the required modules

```
import numpy as np
from flask import Flask, request, render_template
import pickle
```

b. Create the flask app and load the pickle file that contains the model trained using read mode

```
# Create flask app
flask_app = Flask(__name__)
model = pickle.load(open("iris_model.pkl", "rb"))
```

c. **route()** method is used to navigate the api URL based on the path given. Defining the method Home() which will be called when the app is taken to the home page.

```
@flask_app.route("/")
def Home():
    return render_template("home.html")
```

This will load the home.html page

d. Defining the method predict() which will be invoked when the app navigates to the predict page. This method is mainly used for rendering the results in the HTML page. The method POST is used since we are receiving the independent features from the form.

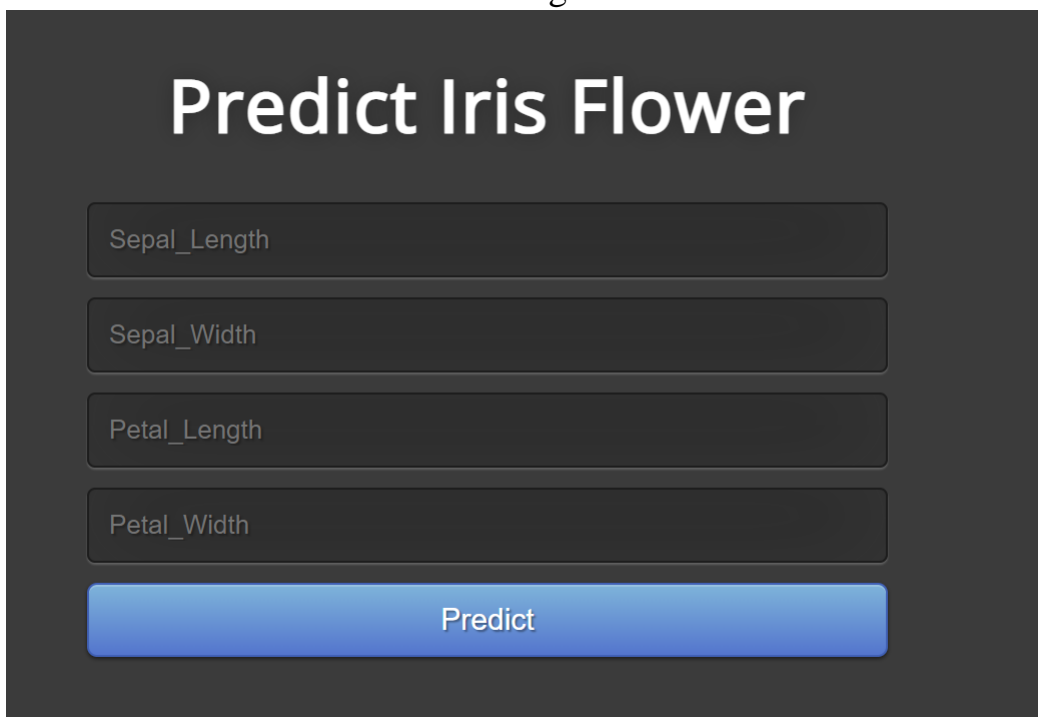
```
@flask_app.route("/predict", methods = ["POST"])
def predict():
    float_features = [float(x) for x in request.form.values()]
    features = [np.array(float_features)]
    prediction = model.predict(features)
    return render_template("home.html", prediction_text =
        "The flower species is {}".format(prediction))
```

This will read the values entered in the form and use the model to make predictions and display the result using the variable prediction_text in the page.

- The form below is written in the home.html page. The form action attribute contains the value url_for('predict') which means, when the form is submitted, which method to be invoked in the app.py file

```
<!-- Main Input For Receiving Query to our ML -->
<form action="{{ url_for('predict')}}" method="post">
<input type="text" name="Sepal_Length" placeholder="Sepal_Length" required="required" />
<br>
<input type="text" name="Sepal_Width" placeholder="Sepal_Width" required="required" />
<br>
<input type="text" name="Petal_Length" placeholder="Petal_Length" required="required" />
<br>
<input type="text" name="Petal_Width" placeholder="Petal_Width" required="required" />
<br>
<button type="submit" class="btn btn-primary btn-block btn-large">Predict</button>
</form>
<br>
{{ prediction_text }}
```

- The screenshots of the server running



The screenshot shows a web interface with a dark background. At the top, the title "Predict Iris Flower" is displayed in large white font. Below the title, there are four stacked text input fields with light gray borders and placeholder text: "Sepal_Length", "Sepal_Width", "Petal_Length", and "Petal_Width". At the bottom of the form is a prominent blue button with rounded corners and the text "Predict" in white.

Predict Iris Flower

5.1

3.5

1.5

0.2

Predict

Predict Iris Flower

Sepal_Length

Sepal_Width

Petal_Length

Petal_Width

Predict

The flower species is ['Virginica']