Noah Igram
DataGlacier
LISUM11
September 6, 2022

Flask Model Deployment

This report details the deployment of a model to a simple flask web app. The model is taken from Stats Wire, a channel on YouTube made for teaching statistics and data science concepts. The model is a flower classifier model which takes four inputs: Sepal length, Sepal width, Petal length, and Petal Width. It then outputs a predicted flower species.

Flower Classification Model

The model takes the following 4 inputs: Sepal length, Sepal width, Petal length, and Petal Width. It then outputs the predicted flower species out of the following choices: Virginica, Versicolor, Setosa. Below is a snapshot of the python script which creates the model using SkLearn's Random Forest Classifier. The final line of the code uses the pickle module Dump to serialize the data encapsulated in the model.

model.py > ...

```python
# This file contains the model which we deploy to a flask app

import pickle
import pandas as pd
from sklearn.preprocessing import StandardScaler
from sklearn.ensemble import RandomForestClassifier
from sklearn.model_selection import train_test_split

df_iris = pd.read_csv("iris.csv")
print(df_iris.head())

x = df_iris[["Sepal_Length", "Sepal_Width", "Petal_Length", "Petal_Width"]]
y = df_iris["Class"]

# Split train and test sets
x_train, x_test, y_train, y_test = train_test_split(
    x, y, test_size=0.25, random_state=40)

# Scale features
scale = StandardScaler()
x_train = scale.fit_transform(x_train)
x_test = scale.transform(x_test)

classifier = RandomForestClassifier()

# Fit the model to the training data
classifier.fit(x_train, y_train)

# Dump to pickle file
pickle.dump(classifier, open("model.pkl", "wb"))

```

HTML for Web App

Basic HTML was written to create an interactive web app which allows the user to input custom Sepal length and width, and Petal length and width and receive a prediction from the classification model. Below is a snapshot of the code. Bootstrap was used to help make the website a bit less bland.

```html
<!DOCTYPE html>
<html>
<!--From https://codepen.io/frytyler/pen/EGdtg-->

<head>
    <meta charset="UTF-8">
    <title>ML API</title>
    <!-- CSS only -->
    <link href="https://cdn.jsdelivr.net/npm/bootstrap@5.2.0/dist/css/bootstrap.min.css" rel="stylesheet"
        integrity="sha384-gH2yIJqKdNHPEq0n4Mqa/HGKIhSkIHeL5AyhkYV8i59U5AR6csBvApHHNl/vI1Bx" crossorigin="anonymous">
</head>

<body>
    <div class="login">
        <h1>Flower Class Prediction</h1>

        <!-- 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" />
            <input type="text" name="Sepal Width" placeholder="Sepal Width" required="required" />
            <input type="text" name="Petal Length" placeholder="Petal Length" required="required" />
            <input type="text" name="Petal Width" placeholder="Petal Width" required="required" />

            <button type="submit" class="btn btn-primary btn-block btn-large">Predict</button>
        </form>

        <br>
        <br>
        {{ prediction_text }}

    </div>


</body>

</html>
```

Web Deployment with Flask

Flask was used to deploy the web app to a local server for personal interaction and testing. Below shows the code which makes this happen.

app.py > ...

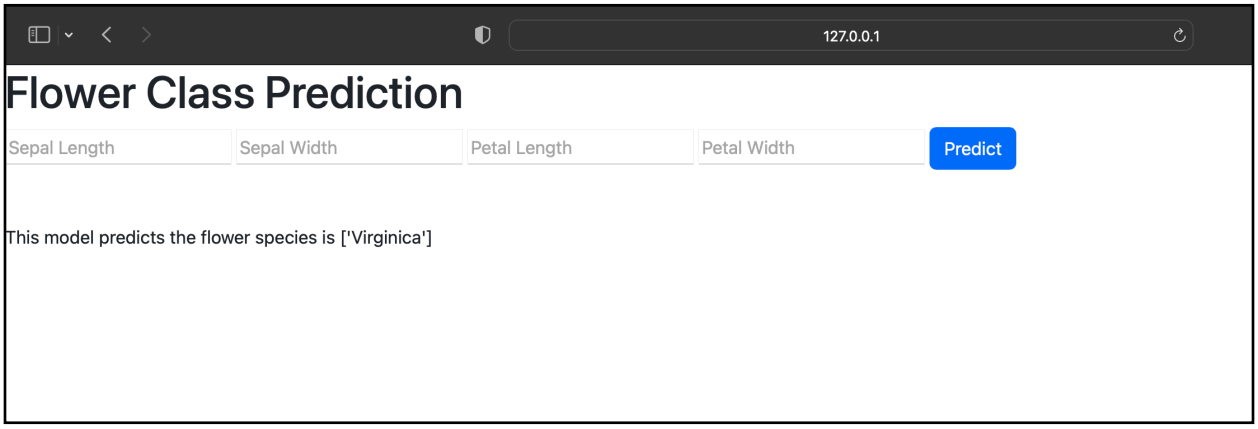
```python
import numpy as np
from flask import Flask, request, jsonify, render_template
import pickle

app = Flask(__name__)
model = pickle.load(open("model.pkl", "rb"))


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


@app.route("/predict", methods=["POST"])
def predict():
    features = [float(x) for x in request.form.values()]
    features = [np.array(features)]
    prediction = model.predict(features)
    return render_template("index.html", prediction_text="This model predicts the flower species is {}".format(prediction))


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



The above image is a snapshot of the web app created. It allows for the user to interact and predict a flower species and was run on a local server. Lastly, below is a snapshot of the terminal which demonstrates what happens when the Flask app is deployed to the local server.