ML model Deployment on Flask

Author: Tomisin Abimbola Adeniyi

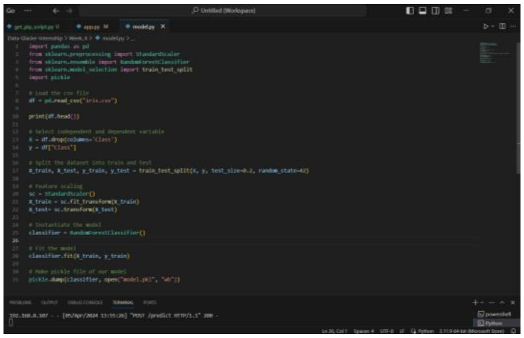
Submission Date: 5/04/2024

Batch Code: LISUM31

Submitted To: GitHub

Introduction: The aim of this project is to build a machine learning model using the Iris dataset provided by scikit-learn and deploy it on Flask. The model is a Random Forest Classifier that predicts the type of flower specie based on their sepal length, sepal width, petal length, and petal width.

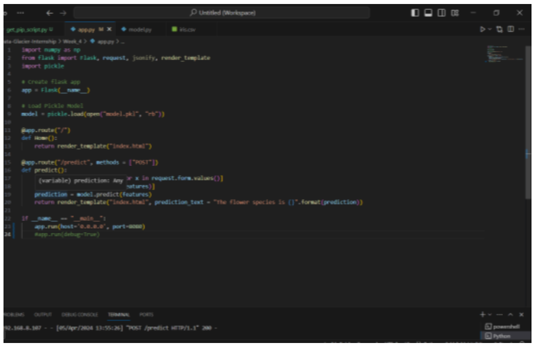
Model Development: The model was developed using Python. The Iris dataset was loaded and split into training and testing sets. Feature scaling was applied to the dataset. A Random Forest Classifier was instantiated and fitted to the training data. The trained model was then saved as a pickle file for later use.



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

# Load the csv file
df = pd.read_csv("iris.csv")

print(df.head())

# Select independent and dependent variable
X = df.drop(columns='Class')
y = df["Class"]

# Split the dataset into train and test
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

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

# Instantiate the model
classifier = RandomForestClassifier()

# Fit the model
classifier.fit(X_train, y_train)

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

```
192.168.0.107 - - [05/Apr/2024 13:55:26] "POST /predict HTTP/1.1" 200 -
```

App Development: A Flask app was created to serve the model. The app has two routes - a home route that renders an HTML form for the user to input the features of a flower, and a predict route that takes these features, makes a prediction using the model, and returns the predicted flower species.

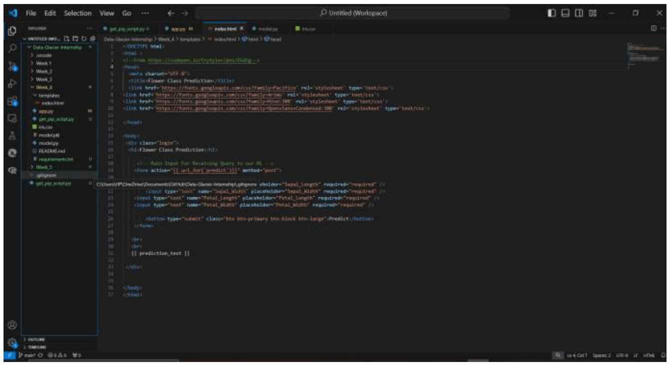


```python
import numpy as np
from flask import Flask, request, jsonify, render_template
import pickle

# Create flask app
app = Flask(__name__)

# Load Pickle Model
model = pickle.load(open("model.pkl", "rb"))

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

@app.route("/predict", methods = ["POST"])
def predict():
    (variable) prediction: Any  or x in request.form.values()]
                                atures)]
    prediction = model.predict(features)
    return render_template("index.html", prediction_text = "The flower species is {}".format(prediction))

if __name__ == "__main__":
    app.run(host='0.0.0.0', port=8080)
    #app.run(debug=True)
```

```
92.168.0.107 - - [05/Apr/2024 13:55:26] "POST /predict HTTP/1.1" 200 -
```

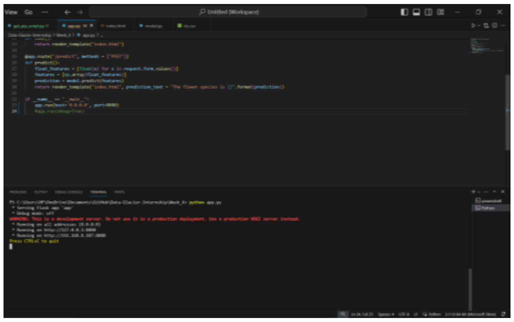
ML model Deployment on Flask

HTML File: An HTML file was created for the Flask app. This file contains a form for the user to input the features of a flower, and a section to display the predicted flower species.



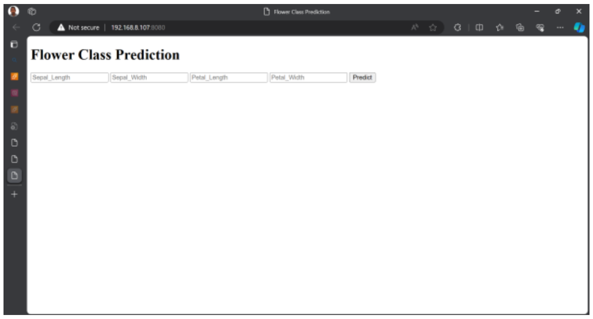
```
<!DOCTYPE html>
<html >
<!--From https://codepen.io/frytyler/pen/EGdtg-->
<head>
  <meta charset="UTF-8">
  <title>Flower Class Prediction</title>
  <link href='https://fonts.googleapis.com/css?family=Pacifico' rel='stylesheet' type='text/css'>
<link href='https://fonts.googleapis.com/css?family=Arimo' rel='stylesheet' type='text/css'>
<link href='https://fonts.googleapis.com/css?family=Hind:300' rel='stylesheet' type='text/css'>
<link href='https://fonts.googleapis.com/css?family=Open+Sans+Condensed:300' rel='stylesheet' type='text/css'>

</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">
C:\Users\HP\OneDrive\Documents\GitHub\Data-Glacier-Internship\.gitignore aholder="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>
```

App Running: The Flask app was run locally for testing. The app successfully received input from the user, made a prediction, and displayed the predicted flower species.



```
    return render_template("index.html")

@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("index.html", prediction_text = "The flower species is {}".format(prediction))

if __name__ == "__main__":
    app.run(host='0.0.0.0', port=8080)
    #app.run(debug=True)
```

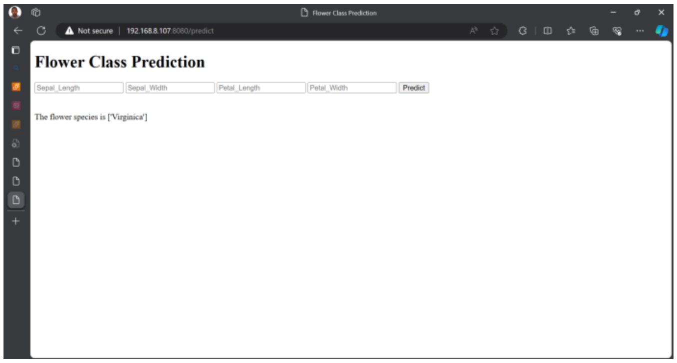
```
PS C:\Users\HP\OneDrive\Documents\GitHub\Data-Glacier-Internship\Week_4> python app.py
 * Serving Flask app 'app'
 * Debug mode: off
WARNING: This is a development server. Do not use it in a production deployment. Use a production WSGI server instead.
 * Running on all addresses (0.0.0.0)
 * Running on http://127.0.0.1:8080
 * Running on http://192.168.8.107:8080
Press CTRL+C to quit
```



Not secure | 192.168.8.107:8080

Flower Class Prediction

Sepal_Length | Sepal_Width | Petal_Length | Petal_Width | Predict



Not secure | 192.168.8.107:8080/predict

Flower Class Prediction

Sepal_Length | Sepal_Width | Petal_Length | Petal_Width | Predict

The flower species is ['Virginica']

This report provides a comprehensive overview of the process of developing a machine learning model and deploying it as a Flask app. It demonstrates the practical application of machine learning in a web development context.