Deployment on Flask

Name: Dong Han Batch code: LISUM16 Submission date: 28.12.2022

Task:

Develop the classification ML model for Iris dataset and deploy on Flask.

Dataset:

iris

Sepal_Length	Sepal_Width	Petal_Length	Petal_Width	Class
5.1	3.5	1.4	0.2	Setosa
4.9	3	1.4	0.2	Setosa
4.7	3.2	1.3	0.2	Setosa
4.6	3.1	1.5	0.2	Setosa
5	3.6	1.4	0.2	Setosa
5.4	3.9	1.7	0.4	Setosa
4.6	3.4	1.4	0.3	Setosa
5	3.4	1.5	0.2	Setosa
4.4	2.9	1.4	0.2	Setosa
4.9	3.1	1.5	0.1	Setosa
5.4	3.7	1.5	0.2	Setosa
4.8	3.4	1.6	0.2	Setosa

Environment Setup (with conda):

conda create -n flask_development python=3.7

HTML Creation (according to the application needs):

Flower Species Prediction (based on Iris dataset)

Sepal_Length | Sepal_Width | Petal_Length | Petal_Width | Predict

{{ prediction_text }}

Model Creation (create ML model and trained on Iris dataset):

```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[["Sepal_Length", "Sepal_Width", "Petal_Length", "Petal_Width"]]
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=0.3, random_state=50)

# 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"))
```

Flask Application Creation:

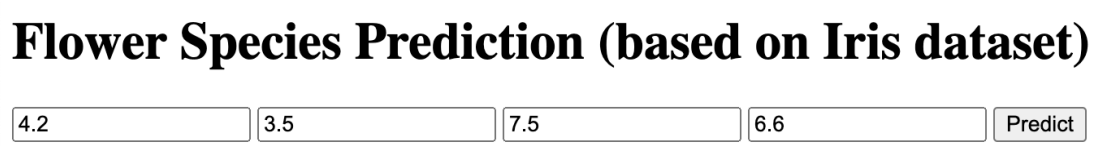
```python
import numpy as np
from flask import Flask, request, jsonify, render_template
import pickle

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

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

@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)
    output = prediction[0]
    return render_template("index.html",
    prediction_text = "The predicted flower specie is {class_name}".format(class_name=output))

if __name__ == "__main__":
    flask_app.run(port=5000,debug=True)
```

Run the Application:

Run the app.py in terminal.

```
❯ python -m app run
 * 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: 894-466-153
```

Open the server and input the values.



Make prediction.