Week 4 Task: Deployment on Flask

Name: Harshad Chormare

Batch code: LISUM19

Submission date: 28/03/2023

Submitted to: Data Glacier

Dataset used: iris flower data set

ML Model: RandomForest Classifier

A) Building a Model

In [2]:
```
import pandas as pd
from sklearn.preprocessing import StandardScaler
from sklearn.ensemble import RandomForestClassifier
from sklearn.model_selection import train_test_split
import pickle
```

In [7]:
```
# load the csv file
df = pd.read_csv("C://ML Model Deployment//IRIS.csv")
```

In [8]:
```
print(df.head())

```

```
   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
```

In [9]:
```
# select independent and dependent variable
X = df[["sepal_length", "sepal_width", "petal_length", "petal_width"]]
Y = df["class"]

```

In [10]:
```
# 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)

```

In [11]:
```
# feature selection
sc = StandardScaler()
X_train = sc.fit_transform(X_train)
X_test = sc.fit_transform(X_test)

```

In [15]:
```
# Instantiate the model
classifier =  RandomForestClassifier()

```

In [18]:
```
# fit the model
classifier.fit(X_train, Y_train)
```

Out[18]:
```
RandomForestClassifier()
```

In [19]:
```
# Make pickle file of the model
pickle.dump(classifier, open("model.pkl", "wb"))

```

In [ ]:
```

```

B) Model to Web Application

App.py

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

# create flask app
app = Flask(__name__)

# load the 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():
    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(debug=True)
```

Index.html

```html
<html>
<!--from https://codepen.io/frytyler/pen/EGdtg-->
<head>
 <meta charset"UTF-8">
 <title> ML API</title>
 <link href='https://fonts.googleapis.com/css?family=Pacifio'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">
       <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>
```

C) Deployment Procedure

```
app
 * Serving Flask app "app" (lazy loading)
 * Environment: production
   WARNING: This is a development server. Do not use it in a production deployment.
   Use a production WSGI server instead.
 * Debug mode: on
 * Restarting with windowsapi reloader
 * Debugger is active!
 * Debugger PIN: 308-885-937
 * Running on http://127.0.0.1:5000/ (Press CTRL+C to quit)
```

D) Web App Result