


DEPLOYING IN FLASK - ASSIGNMENT 4

Report date: 21/03/2021
Batch code: LISP01
Version: 1.0
Intern: Ajaegbu Ebuka Emmanuel
Submitted to: Data Glacier

STEP BY STEP FOR MODEL DEPLOYMENT

Step One: The model was built in Python using the Random Forest Classifier Algorithm with the help of pandas, NumPy and sklearn

[illegible]

jupyter Iris dataset prediction- Assignments Four (autosaved)  Logout

File Edit View Insert Cell Kernel Widgets Help Trusted Python 3

In [6]:

```
1  ### Splitting the data into Train and Test Sets
2  from sklearn.model_selection import train_test_split
3  from sklearn.ensemble import RandomForestClassifier
4  from sklearn.metrics import confusion_matrix
5  import seaborn as sns
6  import matplotlib.pyplot as plt
```

In [7]:

```
1  X_train,X_test,y_train,y_test =train_test_split(X,y,test_size= 0.30,random_state=1000)
2  X_train.shape,X_test.shape,y_train.shape,y_test.shape
```

Out[7]: ((105, 4), (45, 4), (105,), (45,))

In [8]:

```
1  # Initialize the model
2  randomforestclassifier= RandomForestClassifier()
3
4  # Fit the model
5  randomforestclassifier.fit(X_train,y_train)
6
7  # Evaluate using the train set
8  randomforestclassifier.score(X_test,y_test)
```

Out[8]: 0.9555555555555556


In [9]:

```
1  ## Prediction
2  y_pred = randomforestclassifier.predict(X_test)
```

In []:

```
1  plt.figure(figsize=(10,5))
2  sns.heatmap(confusion_matrix(y_test,y_pred), annot=True, cmap="Greens")
3  plt.title("Confusion Matrix")
```

Step two: The model was serialized using pickle modules

jupyter Iris dataset prediction- Assignments Four (autosaved)  Logout

File Edit View Insert Cell Kernel Widgets Help Trusted Python 3

6 plt.show()

Pickel Serialization

In []:

```
1  ## Serializing the Model
2  with open("rdmodelserialized","wb") as f:
3      pickle.dump(randomforestclassifier,f)
4
```

In []:

```
1  ## deserializing the modl
2  with open("rdmodelserialized","rb") as f:
3      rfmodel = pickle.load(f)
```

In []:

```
1  ## Prediction Usingf the Unseroialized model
2  y_pred1 = rfmodel.predict(X_test)
```

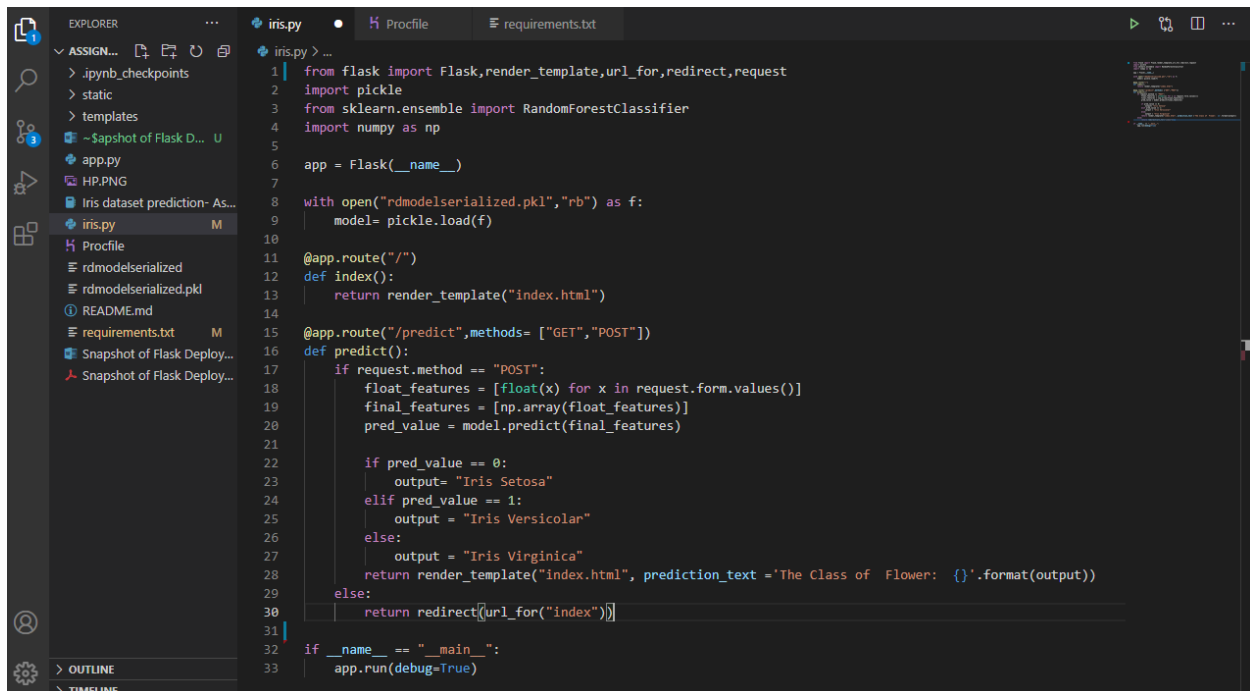
In []:

```
1  plt.figure(figsize=(10,5))
2  sns.heatmap(confusion_matrix(y_test,y_pred1), annot=True, cmap="Greens")
3  plt.title("Confusion Matrix")
4  plt.xlabel("True value")
5  plt.ylabel("Predicted value")
6  plt.show()
```

In []:

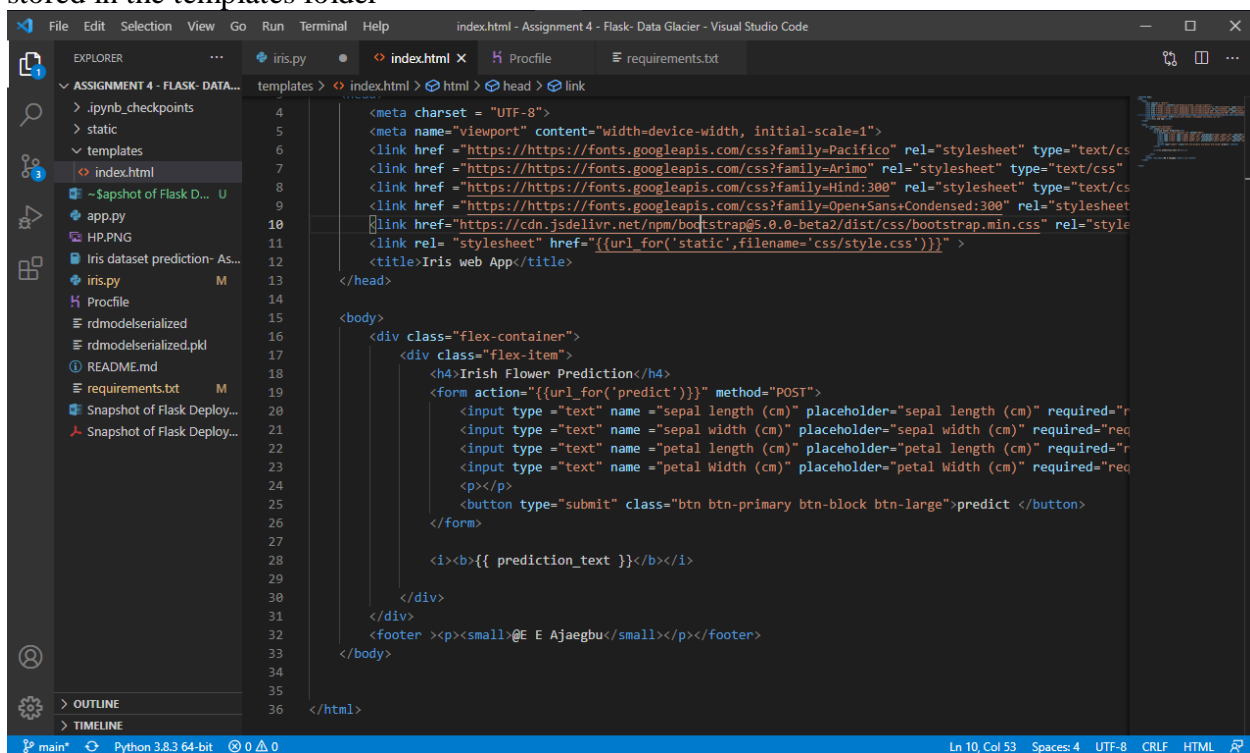
```
1  ### Thank You
```

Step Three: Flask app was built with the help of the Flask, render_emplate, request, url_for methods. The serialized model was then imported and unserialized and the used for prediction



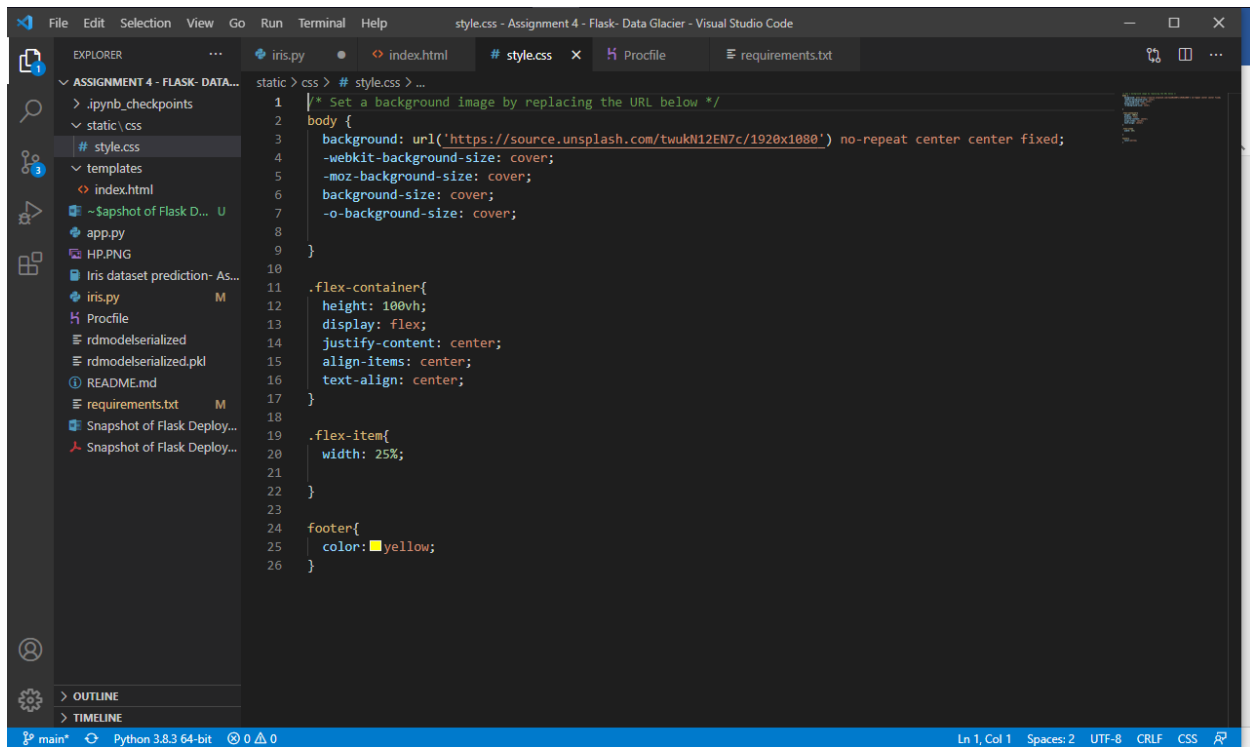
```
1 from flask import Flask, render_template, url_for, redirect, request
2 import pickle
3 from sklearn.ensemble import RandomForestClassifier
4 import numpy as np
5
6 app = Flask(__name__)
7
8 with open("rdmodelserialized.pkl", "rb") as f:
9     model = pickle.load(f)
10
11 @app.route("/")
12 def index():
13     return render_template("index.html")
14
15 @app.route("/predict", methods= ["GET", "POST"])
16 def predict():
17     if request.method == "POST":
18         float_features = [float(x) for x in request.form.values()]
19         final_features = [np.array(float_features)]
20         pred_value = model.predict(final_features)
21
22         if pred_value == 0:
23             output = "Iris Setosa"
24         elif pred_value == 1:
25             output = "Iris Versicolour"
26         else:
27             output = "Iris Virginica"
28         return render_template("index.html", prediction_text = 'The Class of Flower: {}'.format(output))
29     else:
30         return redirect(url_for("index"))
31
32 if __name__ == "__main__":
33     app.run(debug=True)
```

Step Four: The index.html file was created for building the structure of the websites. this file is stored in the templates folder



```
4 <meta charset = "UTF-8">
5 <meta name="viewport" content="width=device-width, initial-scale=1">
6 <link href = "https://https://fonts.googleapis.com/css?family=Pacifico" rel="stylesheet" type="text/css">
7 <link href = "https://https://fonts.googleapis.com/css?family=Arimo" rel="stylesheet" type="text/css">
8 <link href = "https://https://fonts.googleapis.com/css?family=Hind:300" rel="stylesheet" type="text/css">
9 <link href = "https://https://fonts.googleapis.com/css?family=Open+Sans+Condensed:300" rel="stylesheet">
10 <link href="https://cdn.jsdelivr.net/npm/bootstrap@5.0.0-beta2/dist/css/bootstrap.min.css" rel="stylesheet">
11 <link rel="stylesheet" href="{{url_for('static', filename='css/style.css')}}">
12 <title>Iris web App</title>
13 </head>
14
15 <body>
16     <div class="flex-container">
17         <div class="flex-item">
18             <h4>Irish Flower Prediction</h4>
19             <form action="{{url_for('predict')}}" method="POST">
20                 <input type="text" name="sepal length (cm)" placeholder="sepal length (cm)" required="required">
21                 <input type="text" name="sepal width (cm)" placeholder="sepal width (cm)" required="required">
22                 <input type="text" name="petal length (cm)" placeholder="petal length (cm)" required="required">
23                 <input type="text" name="petal Width (cm)" placeholder="petal Width (cm)" required="required">
24                 <p></p>
25                 <button type="submit" class="btn btn-primary btn-block btn-large">predict </button>
26             </form>
27
28             <div><b>{{ prediction_text }}</b></div>
29         </div>
30     </div>
31
32 <footer><p><small>© E Ajaegbu</small></p></footer>
33 </body>
34
35 </html>
```

Step Five: The CSS file was created for styling the web app and for also bootstrapping.



Step Six: The files will then uploaded to GitHub

Step Seven: The web application will be deployed in the cloud.- Heroku