

ML Deployment

Data Analysis Internship

2024



Task Done By:
Shahad Ali Aldawsari

Batch code:
LISUM33

Submission date:
2024 May 28

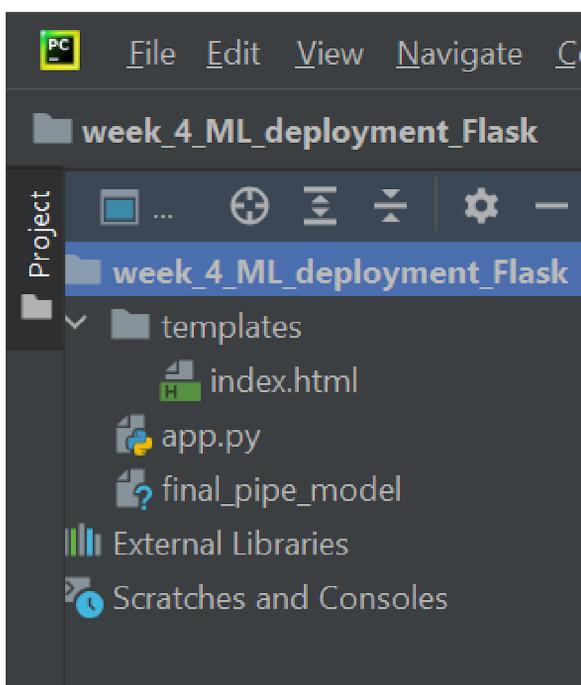
Content:

- 1. Over View**
- 2. HTML file**
- 3. Python file**
- 4. Pickled model file**
- 5. Output**

Over View:

To deploy a ML model on Flask you need to create a folder on Pycharm contains three files:

- HTML file
- Python file
- Pickled model file



The folder that I created

HTML file:

```
1  <!DOCTYPE html>
2  <html >
3  <!--From https://codepen.io/frytyler/pen/EGdtq-->
4  <head>
5      <meta charset="UTF-8">
6      <title>ML API</title>
7      <link href='https://fonts.googleapis.com/css?family=Pacifico' rel='stylesheet' type='text/css'>
8      <link href='https://fonts.googleapis.com/css?family=Arimo' rel='stylesheet' type='text/css'>
9      <link href='https://fonts.googleapis.com/css?family=Hind:300' rel='stylesheet' type='text/css'>
10     <link href='https://fonts.googleapis.com/css?family=Open+Sans+Condensed:300' rel='stylesheet' type='text/css'>
11 </head>
12 <body>
13     <div class="login">
14         <h1>Flower Class Prediction</h1>
15         <!-- Main Input For Receiving Query to our ML -->
16         <form action="{{ url_for('predict') }}" method="post">
17             <input type="text" name="Sepal_Length" placeholder="Sepal_Length" required="required" />
18             <input type="text" name="Sepal_Width" placeholder="Sepal_Width" required="required" />
19             <input type="text" name="Petal_Length" placeholder="Petal_Length" required="required" />
20             <input type="text" name="Petal_Width" placeholder="Petal_Width" required="required" />
21             <button type="submit" class="btn btn-primary btn-block btn-large">Predict</button>
22         </form>
23         <br>
24         <br>
25         {{ prediction_text }}
26     </div>
27 </body>
```

The HTML code



The HTML code output

Python file:

```
1 import numpy as np
2 import pickle
3 from flask import Flask, request, jsonify, render_template
4
5 app = Flask(__name__)
6 model = pickle.load(open("final_pipe_model", 'rb'))
7 @app.route("/")
8 def Home():
9     return render_template("index.html")
10 @app.route("/predict", methods = ["POST"])
11 def predict():
12     species_mapping = ['Setosa', 'Versicolor', 'Virginica']
13     float_features = [float(x) for x in request.form.values()]
14     features = [np.array(float_features)]
15     prediction = species_mapping[int(model.predict(features))])
16     return render_template("index.html", prediction_text = "The flower type is {}".format(prediction))
17 if __name__ == "__main__":
18     app.run(debug=True)
19
```

Pickled model file:

The summary of the main steps to pickle a model:

The screenshot shows a Jupyter Notebook interface with the following details:

- Contents:** A sidebar menu with a tree structure:
 - 1 Multi-Class Logistic Regression
 - 1.1 Exploratory Data Analysis and Visualization
 - 1.2 Train | Test Split and Modeling
 - 1.3 Model Performance
 - 1.3.1 With Default Parameters
 - 1.3.2 Cross Validate
 - 1.3.3 Cross Validate for versicolor
 - 1.4 Class prediction
 - 1.4.1 With Best Parameters (GridsearchCV)
 - 1.5 ROC (Receiver Operating Curve) and AUC
 - 1.6 Precision Recall Curve
 - 1.7 Final Model and Model Deployment
- Code Cell 1:** Multi-Class Logistic Regression code.

```
import numpy as np
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
plt.rcParams["figure.figsize"] = (7,4)
import warnings
warnings.filterwarnings("ignore")
warnings.warn("this will not show")
pd.set_option('display.float_format', lambda x: '%.3f' % x)
```
- Code Cell 2:** Reading the 'iris.csv' dataset.

```
df = pd.read_csv('iris.csv')
```
- Code Cell 3:** Displaying the first 5 rows of the dataset.

```
df.head()
```

Output 3:

	sepal_length	sepal_width	petal_length	petal_width	species
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

The dataset I used “iris”

Pickled model file:

The screenshot shows a Jupyter Notebook interface with a "Contents" sidebar on the left. The main area displays code for "Model Performance".

```
In [14]: 1 from sklearn.model_selection import GridSearchCV  
2 from sklearn.metrics import accuracy_score, confusion_matrix, classification_report, Confus  
3  
In [15]: 1 y_pred = pipe_model.predict(X_test)  
2 y_pred  
3  
4 # target can be numeric data or object data.  
5  
In [16]: 1 ConfusionMatrixDisplay.from_estimator(pipe_model, X_test, y_test);  
2  
3  
In [17]: 1 def eval_metric(model, X_train, y_train, X_test, y_test):  
2     y_train_pred = model.predict(X_train)  
3     y_pred = model.predict(X_test)  
4  
5     print("Test_Set")  
6     print(confusion_matrix(y_test, y_pred))  
7     print(classification_report(y_test, y_pred))  
8     print()
```

The model building

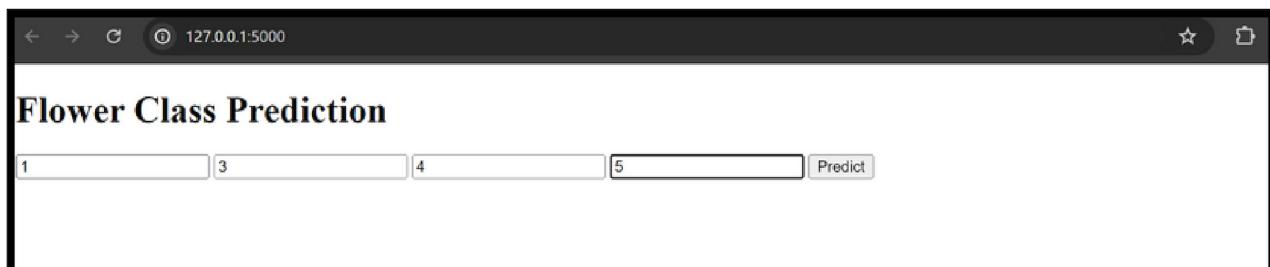
The screenshot shows a Jupyter Notebook interface with a "Contents" sidebar on the left. The main area displays code for "Final Model and Model Deployment".

```
In [41]: 1 from sklearn.metrics import average_precision_score, roc_auc_score  
2  
3 y_test_dummies = pd.get_dummies(y_test).values  
4  
5 average_precision_score(y_test_dummies[:, 1], y_pred_proba[:, 1])  
Out[41]: 1.0  
  
In [42]: 1 operations = [("scaler", StandardScaler()), ("logistic", LogisticRegression())]  
2  
3 final_model = Pipeline(steps=operations)  
4  
5 final_model.fit(X, y)  
6  
In [43]: 1 import pickle  
2 pickle.dump(final_model, open("final_pipe_model", "wb"))  
3  
In [44]: 1 new_model = pickle.load(open("final_pipe_model", "rb"))  
2  
In [45]: 1 X.describe().T
```

The model pickling

Output:

1. Run the code
2. You will get a locale link, click on it
3. A web page will appear
4. Enter your input values
5. Click on “Predict” button
6. Finally, you will see the prediction result
EX: “The flower type is Setosa”



Flower Class Prediction

The input



Flower Class Prediction

The flower type is Virginica

The output

Task Done By:
Shahad Ali Aldawsari

Batch code:
LISUM33

Submission date:
2024 May 28