

Name: Kierra Dangerfield
Batch Code: LISUM23: 30 June - 30 Sep. 2023
Submission Date: July 26, 2023
Submitted to:

Deploying ML Model with Flask

1. Coding Model

iris-modeling.ipynb

I created my model in a jupyter notebook. I start off by collecting my data that I got from Kaggle. I used the Iris dataset.

```
In [1]: 1 import pandas as pd
        2 import numpy as np
        3 import pickle
        4
        5 from sklearn.neighbors import KNeighborsClassifier
        6 from sklearn.model_selection import train_test_split
        7 from sklearn.preprocessing import StandardScaler
        8
        9 from sklearn.metrics import classification_report, confusion_matrix
```

Collect data

```
In [2]: 1 iris = pd.read_csv("../Data/iris.csv")
```

```
In [3]: 1 iris.head()
```

```
Out[3]:
```

	Id	SepalLengthCm	SepalWidthCm	PetalLengthCm	PetalWidthCm	Species
0	1	5.1	3.5	1.4	0.2	Iris-setosa
1	2	4.9	3.0	1.4	0.2	Iris-setosa
2	3	4.7	3.2	1.3	0.2	Iris-setosa
3	4	4.6	3.1	1.5	0.2	Iris-setosa
4	5	5.0	3.6	1.4	0.2	Iris-setosa

I then did a quick check for null and duplicate values

```
In [4]: 1 iris.info()
```

```
<class 'pandas.core.frame.DataFrame'>  
RangeIndex: 150 entries, 0 to 149  
click to expand output; double click to hide output ):  
#      Column      Non-Null Count  Dtype  
---  -  
0     Id           150 non-null     int64  
1     SepalLengthCm 150 non-null     float64  
2     SepalWidthCm  150 non-null     float64  
3     PetalLengthCm 150 non-null     float64  
4     PetalWidthCm  150 non-null     float64  
5     Species       150 non-null     object  
dtypes: float64(4), int64(1), object(1)  
memory usage: 7.2+ KB
```

```
In [5]: 1 iris.duplicated().sum()
```

```
Out[5]: 0
```

```
In [6]: 1 iris.isnull().sum()
```

```
Out[6]: Id           0  
SepalLengthCm      0  
SepalWidthCm       0  
PetalLengthCm      0  
PetalWidthCm       0  
Species            0  
dtype: int64
```

Model

1. Store the input features that I want in X and the output feature (Species) in y.
2. train/test split
3. Feature Standardization with X_train and X_test
4. Train and fit model using a KNN classifier
5. Make predictions on X_test
6. Check results

Model

```
In [7]: 1 X = iris[["SepalLengthCm", "SepalWidthCm", "PetalLengthCm", "PetalWidthCm"]]
2 y = iris["Species"]
3
4 #train test split
5 X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.20, random_state=0)
6
7
8 # feature standardization
9 scaler = StandardScaler()
10 scaler.fit(X_train)
11
12 X_train = scaler.transform(X_train)
13 X_test = scaler.transform(X_test)
```

```
In [8]: 1 #Train and fit model
2 knn = KNeighborsClassifier(n_neighbors=5).fit(X_train, y_train)
3
4
5 #make predictions
6 y_predict = knn.predict(X_test)
7
8 #Check results
9 print(confusion_matrix(y_test, y_predict))
10 print(classification_report(y_test, y_predict))
```

```
[[11  0  0]
 [ 0 13  0]
 [ 0  0  6]]
```

	precision	recall	f1-score	support
Iris-setosa	1.00	1.00	1.00	11
Iris-versicolor	1.00	1.00	1.00	13
Iris-virginica	1.00	1.00	1.00	6
accuracy			1.00	30
macro avg	1.00	1.00	1.00	30
weighted avg	1.00	1.00	1.00	30

Save Model

1. Save model to disk using pickle called "iris_model.pkl"
2. Load model to compare the results
3. Evaluate model

Save Model

```
In [9]: 1 # Saving model to disk
        2 pickle.dump(knn, open("iris_model.pkl", 'wb'))
        3
        4 # Loading model to compare the results
        5 model = pickle.load(open('iris_model.pkl', 'rb'))
```

```
In [10]: 1 print(model.predict([[5.0, 3.0, 1.5, 0.2]]))
        ['Iris-virginica']
```

```
In [11]: 1 # evaluate model
        2 y_predict = model.predict(X_test)
        3
        4 # check results
        5 print(classification_report(y_test, y_predict))
```

	precision	recall	f1-score	support
Iris-setosa	1.00	1.00	1.00	11
Iris-versicolor	1.00	1.00	1.00	13
Iris-virginica	1.00	1.00	1.00	6
accuracy			1.00	30
macro avg	1.00	1.00	1.00	30
weighted avg	1.00	1.00	1.00	30

2. Create flask application

model_deploy.py

I created the Flask application in visual studio code.

```
model_deploy.py X index.html # style.css iris-modeling.ipynb Python - Get Started

model_deploy.py > ...
1 import numpy as np
2 from flask import Flask, request, render_template
3 import pickle
4
5 app = Flask(__name__)
6 model = pickle.load(open("iris_model.pkl", "rb"))
7
8
9 @app.route('/')
10 def home():
11     return render_template("index.html")
12
13
14 @app.route('/predict', methods=['POST'])
15 def predict():
16     '''
17     For rendering results on HTML GUI
18     '''
19     int_features = [float(x) for x in request.form.values()]
20     final_features = [np.array(int_features)]
21     prediction = model.predict(final_features)
22
23     return render_template('index.html', prediction_text=''.join('The species is a ' + prediction))
24
25
26 if __name__ == "__main__":
27     app.run(port=5000, debug=True)
28
```

I then created the **index.html** file for output.

```
model_deploy.py  index.html  # style.css  iris-modeling.ipynb  Python - Get Started

templates > index.html > html > head > title
1  <!DOCTYPE html>
2  <html>
3
4  <head>
5      <meta charset="UTF-8">
6      <title>Model Deployment</title>
7      <link rel="stylesheet" href="/static/css/style.css">
8
9  </head>
10
11 <body>
12     <div class="container">
13         <div class="row" style="height:500px;">
14             <h1 class="purple">Predict Iris Species</h1>
15
16             <!-- Main Input For Receiving Query to our ML -->
17             <form action="{{ url_for('predict')}}" method="post">
18                 <div class="form-group">
19                     <input type="text" name="SepalLengthCm" placeholder="Sepal Length in Cm" required="required">
20                 </div>
21                 <div class="form-group">
22                     <input type="text" name="SepalWidthCm" placeholder="Sepal Width in Cm" required="required">
23                 </div>
24                 <div class="form-group">
25                     <input type="text" name="PetalLengthCm" placeholder="Petal Length in Cm" required="required">
26                 </div>
27                 <div class="form-group">
28                     <input type="text" name="PetalWidthCm" placeholder="Petal Width Cm" required="required">
29                 </div>
30
31                 <button type="submit" class="btn">Predict</button>
32             </form>
33
34             <br>
35             <br>
36             {{ prediction_text }}
37         </div>
38     </div>
39
40
41 </body>
42
43 </html>
```

This is the output without CSS styling

Predict Iris Species

Please fill out this field.

The species is a Iris-virginica

3. CSS Styling

Added some CSS for styling.

```
model_deploy.py  index.html  # style.css  iris-modeling.ipynb  P
static > css > # style.css > .form-group
1  .container {
2      text-align: center;
3      margin: 5% 40%;
4  }
5  .purple{
6      color: #7213DC;
7  }
8
9  h1 {
10     font-family: Georgia, serif;
11     font-size: xxx-large;
12 }
13
14 .btn{
15     border: none;
16     height: 40px;
17     width: 75px;
18     font-size: medium;
19     text-align: center;
20     background-color: #7213DC;
21     color: white;
22     margin-top: 20px;
23     border-radius: 5px;
24     box-shadow: 0 7px 9px 0 rgba(0,0,0,0.2), 0 2px 20px 0 rgba(0,0,0,0.2)
25 }
26
27 html {
28     background-color: rgba(114,19,220, 0.3)!important;
29 }
30
31 .form-group {
32     padding: 5px;
33 }
34
35 input[type="text"] {
36     height: 40px;
37     border-radius: 5px;
38     border: 3px solid #7213DC;
39 }
40
41 form{
42     font-family: 'Trebuchet MS', sans-serif;
43 }
```

Final output

Predict Iris Species