

Deployment Report

Name: Iris Prediction on Flask

Internship Batch: <LISUM19>

Issued by: <Kohsuke Uchimura>

Submission date: <03/27/2023>

Submitted to: < <https://github.com/KHUC1998/Model-deployment-on-flask> >

Outline

This is web application for identifying species of iris. Iris has three types of species, Setosa, Versicolor, Virginica. Users can identify species of iris inputting features of iris into web app. Sample dataset is iris dataset from sklearn's toy data. I deployed multilayer perceptron model on python and pass it to flask application.

Model section

Use MLP classifier model to identify species of iris. After deploying this model, dump this model to pkl file to use on flask. Test score was measured by using classification report.

```
1 from sklearn.datasets import load_iris
2 from sklearn.model_selection import train_test_split
3 from sklearn.neural_network import MLPClassifier
4 import joblib
5 from sklearn.metrics import classification_report, accuracy_score
6 # import pickle
7
8 # データ取得
9 iris = load_iris()
10 x, y = iris.data, iris.target
11
12 # 訓練データとテストデータに分割
13 x_train, x_test, y_train, y_test = train_test_split(x, y, test_size=0.5, random_state=0)
14
15 # solverには確率的勾配降下法(sgd)やadamなどが利用可能です。
16 model = MLPClassifier(solver="sgd", random_state=0, max_iter=3000)
17
18 # 学習
19 model.fit(x_train, y_train)
20 pred = model.predict(x_test)
21
22 # 学習済みモデルの保存
23 joblib.dump(model, "nn.pkl", compress=True)
24
25 # filename = 'nn.sav'
26 # pickle.dump(model, open(filename, 'wb'))
27
28 # 予測精度
29 print("result: ", model.score(x_test, y_test))
30 print(classification_report(y_test, pred))
31
```

```
In [14]: import joblib
from sklearn.metrics import classification_report, accuracy_score
joblib.dump(model, "iris Predict/nn.pkl", compress = True)

print(f"result: {model.score(X_test, y_test)}")
print(classification_report(y_test, pred))
```

```
result: 0.9809523809523809
      precision  recall f1-score  support
0          1.00    1.00    1.00        33
1          0.97    0.97    0.97        34
2          0.97    0.97    0.97        38

      accuracy          0.98    105
macro avg    0.98    0.98    0.98    105
weighted avg    0.98    0.98    0.98    105
```

Application Section

Deploy the model on flask. Flask app include four layer to input features, Sepal Length, Sepal Width, Petal Length, and Petal Width. After input features and click identify button, the model is deployed automatically, and show result on web screen.

```

class IrisForm(Form):
    SepalLength = FloatField("Sepal Length(cm)",
                             [validators.InputRequired("Input Required"),
                              validators.NumberRange(min = 0, max = 10)])

    SepalWidth = FloatField("Sepal Width(cm)",
                             [validators.InputRequired("Input Required"),
                              validators.NumberRange(min = 0, max = 10)])

    PetalLength = FloatField("Petal Length(cm)",
                             [validators.InputRequired("Input Required"),
                              validators.NumberRange(min = 0, max = 10)])

    PetalWidth = FloatField("Petal Width(cm)",
                             [validators.InputRequired("InputRequired"),
                              validators.NumberRange(min = 0, max = 10)])

    submit = SubmitField("Classify")

@app.route('/', methods = ['GET', 'POST'])

def predicts():
    form = IrisForm(request.form)
    if request.method == "POST":
        if form.validate() == False:
            flash("Input Required.")
            return render_template("index.html", form = form)
        else:
            SepalLength = float(request.form["SepalLength"])
            SepalWidth = float(request.form["SepalWidth"])
            PetalLength = float(request.form["PetalLength"])
            PetalWidth = float(request.form["PetalWidth"])

            x = np.array([SepalLength, SepalWidth, PetalLength, PetalWidth])

87         pred = predict(x)
88         irisName = getName(pred)
89
90         return render_template("result.html", irisName = irisName)
91     elif request.method == "GET":
92         return render_template("index.html", form = form)
93
94
95 if __name__ == "__main__":
96     app.debug = True
97     app.run()
98

```

HTML Section

Set input section as defined application section. To show result, result.html display answer passed by.pkl file which contains MLP classifier model.

```

1 <title>Iris Predict App</title>
2 <style>
3 #wrapper {
4   text-align: center;
5 }
6 </style>
7
8 <div id="wrapper">
9   {% for message in form.SepalLength.errors %}
10    <div>{{ message }}</div>
11    {% endfor %}
12
13   {% for message in form.SepalWidth.errors %}
14    <div>{{ message }}</div>
15    {% endfor %}
16
17   {% for message in form.PetalLength.errors %}
18    <div>{{ message }}</div>
19    {% endfor %}
20
21   {% for message in form.PetalWidth.errors %}
22    <div>{{ message }}</div>
23    {% endfor %}
24
25   <form method="post">
26     {{ form.SepalLength.label }}<br>
27     {{ form.SepalLength }}
28     <br>
29     {{ form.SepalWidth.label }}<br>
30     {{ form.SepalWidth }}
31     <br>
32     {{ form.PetalLength.label }}<br>
33     {{ form.PetalLength }}
34     <br>
35     {{ form.PetalWidth.label }}<br>
36     {{ form.PetalWidth }}
37     <br>
38     {{ form.submit }}
39   </form>
40 </div>
41

```

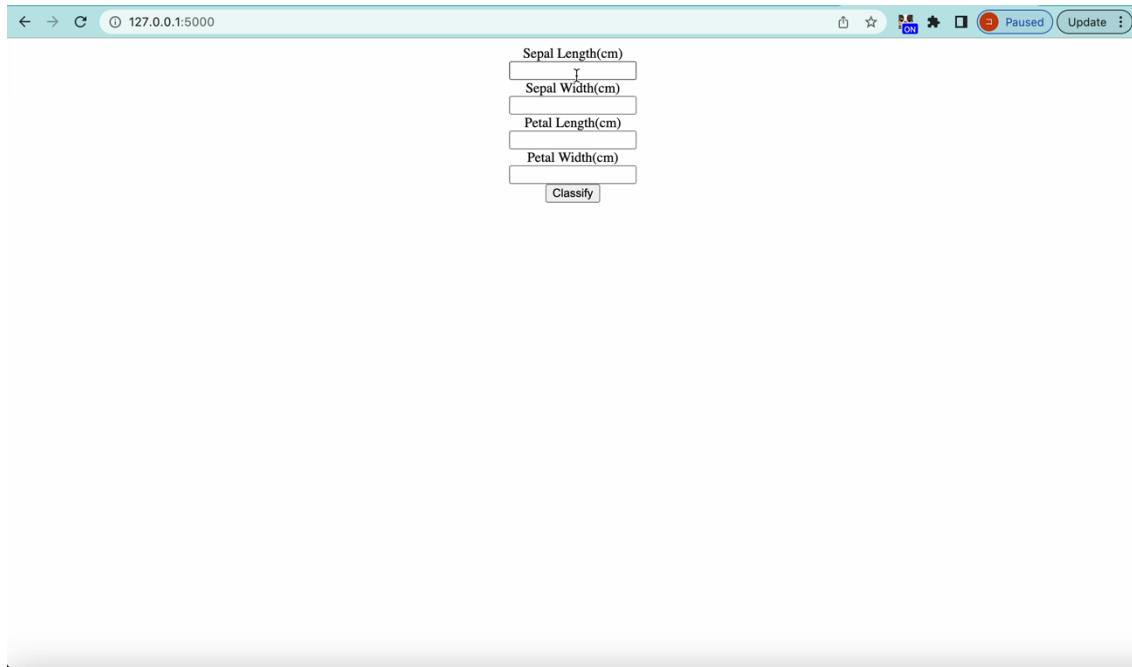
result.html

```

1 <!DOCTYPE html>
2 <title>Iris Predict App</title>
3 {% if irisName %}
4   <h1>This is {{ irisName }}.</h1>
5   <a href="/">Return to Top</a>
6 {% else %}
7   <a href="/">Return to Top</a>
8 {% endif %}
9

```

Deployment on flask



A screenshot of a web browser displaying a web application. The browser's address bar shows the URL "127.0.0.1:5000". The application interface is centered and contains five input fields stacked vertically, each with a label above it: "Sepal Length(cm)", "Sepal Width(cm)", "Petal Length(cm)", and "Petal Width(cm)". Below these input fields is a "Classify" button. The browser's top bar includes navigation icons, a star icon, and a "Paused" status indicator. An "Update" button is visible in the top right corner of the application area.