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.

```python
from sklearn.datasets import load_iris
from sklearn.model_selection import train_test_split
from sklearn.neural_network import MLPClassifier
import joblib
from sklearn.metrics import classification_report, accuracy_score
# import pickle

# データ取得
iris = load_iris()
x, y = iris.data, iris.target

# 訓練データとテストデータに分割
x_train, x_test, y_train, y_test = train_test_split(x, y, test_size=0.5, random_state=0)

# solverには確率的勾配降下法(sgd)やadamなどが利用可能です。
model = MLPClassifier(solver="sgd", random_state=0, max_iter=3000)

# 学習
model.fit(x_train, y_train)
pred = model.predict(x_test)

# 学習済みモデルの保存
joblib.dump(model, "nn.pkl", compress=True)

# filename = 'nn.sav'
# pickle.dump(model, open(filename, 'wb'))

# 予測精度
print("result: ", model.score(x_test, y_test))
print(classification_report(y_test, pred))
```

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.

```python
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])

                pred = predict(x)
                irisName = getName(pred)

                return render_template("result.html",irisName = irisName)
        elif request.method == "GET":
            return render_template("index.html", form = form)

if __name__ == "__main__":
    app.debug = True
    app.run()
```

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.

```html
<title>Iris Predict App</title>
<style>
#wrapper {
   text-align: center;
}
</style>

      <div id="wrapper">
         {% for message in form.SepalLength.errors %}
            <div>{{ message }}</div>
         {% endfor %}

         {% for message in form.SepalWidth.errors %}
            <div>{{ message }}</div>
         {% endfor %}

         {% for message in form.PetalLength.errors %}
            <div>{{ message }}</div>
         {% endfor %}

         {% for message in form.PetalWidth.errors %}
            <div>{{ message }}</div>
         {% endfor %}

         <form method="post">
            {{ form.SepalLength.label }}<br>
            {{ form.SepalLength }}
            <br>
            {{ form.SepalWidth.label }}<br>
            {{ form.SepalWidth }}
            <br>
            {{ form.PetalLength.label }}<br>
            {{ form.PetalLength }}
            <br>
            {{ form.PetalWidth.label }}<br>
            {{ form.PetalWidth }}
            <br>
            {{ form.submit }}
         </form>
      </div>
```

result.html

```html
<!DOCTYPE html>
<title>Iris Predict App</title>
{% if irisName %}
  <h1>This is {{ irisName }}.</h1>
  <a href="./">Return to Top</a>
{% else %}
  <a href="./">Return to Top</a>
{% endif %}
```

Deployment on flask