

# Deployment on Flask

Name: Anastasiia Trofymova

Batch code: LISUM28

Submission date: 28 December 2023

Submitted to: Data Glacier

## 1. Select any toy data (simple data).

**iris.data.csv** (4.55 kB)

Detail   Compact   Column

# 5.1	# 3.5	# 1.4	# 0.2	▲ Iris-setosa
4.8	3.0	1.4	0.3	Iris-setosa
5.1	3.8	1.6	0.2	Iris-setosa
4.6	3.2	1.4	0.2	Iris-setosa
5.3	3.7	1.5	0.2	Iris-setosa
5.0	3.3	1.4	0.2	Iris-setosa
7.0	3.2	4.7	1.4	Iris-versicolor
6.4	3.2	4.5	1.5	Iris-versicolor
6.9	3.1	4.9	1.5	Iris-versicolor
5.5	2.3	4.0	1.3	Iris-versicolor
6.5	2.8	4.6	1.5	Iris-versicolor
5.7	2.8	4.5	1.3	Iris-versicolor
6.3	3.3	4.7	1.6	Iris-versicolor

## 2. Save the model.

### model.py

```
# Load Iris Dataset
from sklearn import datasets
iris = datasets.load_iris()
X = iris.data
y = iris.target

# Split to Train and Test Datasets
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.20)

# Transform Features
from sklearn.preprocessing import StandardScaler
scaler = StandardScaler()
scaler.fit(X_train)
X_train = scaler.transform(X_train)
X_test = scaler.transform(X_test)

# Build and Train Model
from sklearn.neural_network import MLPClassifier
mlp = MLPClassifier(hidden_layer_sizes=(10, 10, 10), max_iter=1000)
mlp.fit(X_train, y_train)
predictions = mlp.predict(X_test)

# Evaluation of the Model
from sklearn.metrics import classification_report, confusion_matrix
print(confusion_matrix(y_test, predictions))
print(classification_report(y_test, predictions))
```

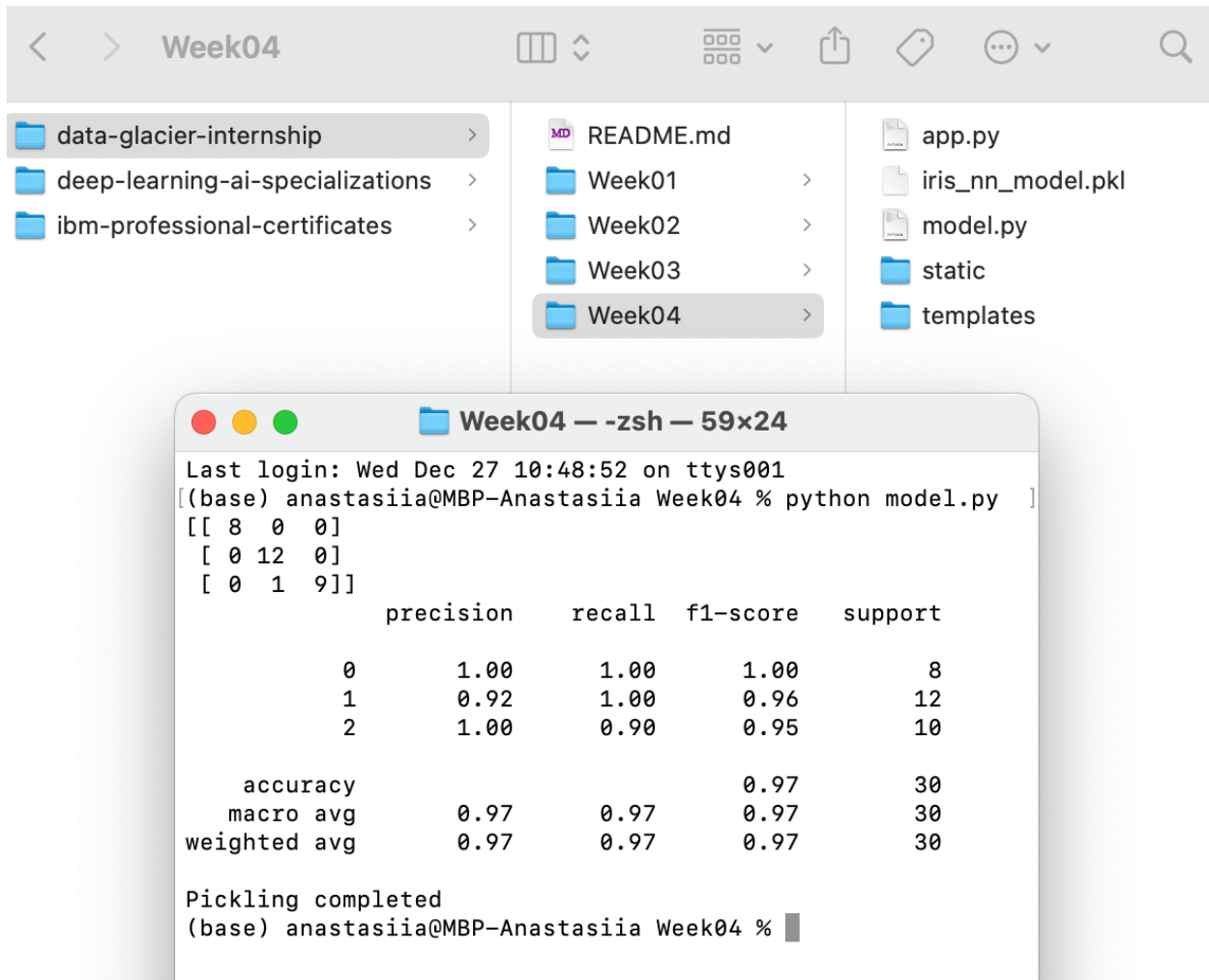
### Object Serialization by using Pickle:

```
import pickle
```

### model

```
# Save the model to a file
with open('iris_nn_model.pkl', 'wb') as f:
    pickle.dump(mlp, f)
    print("Pickling completed")
```

## In Terminal:



```
Week04
```

data-glacier-internship >  
deep-learning-ai-specializations >  
ibm-professional-certificates >

README.md  
Week01 >  
Week02 >  
Week03 >  
Week04 >

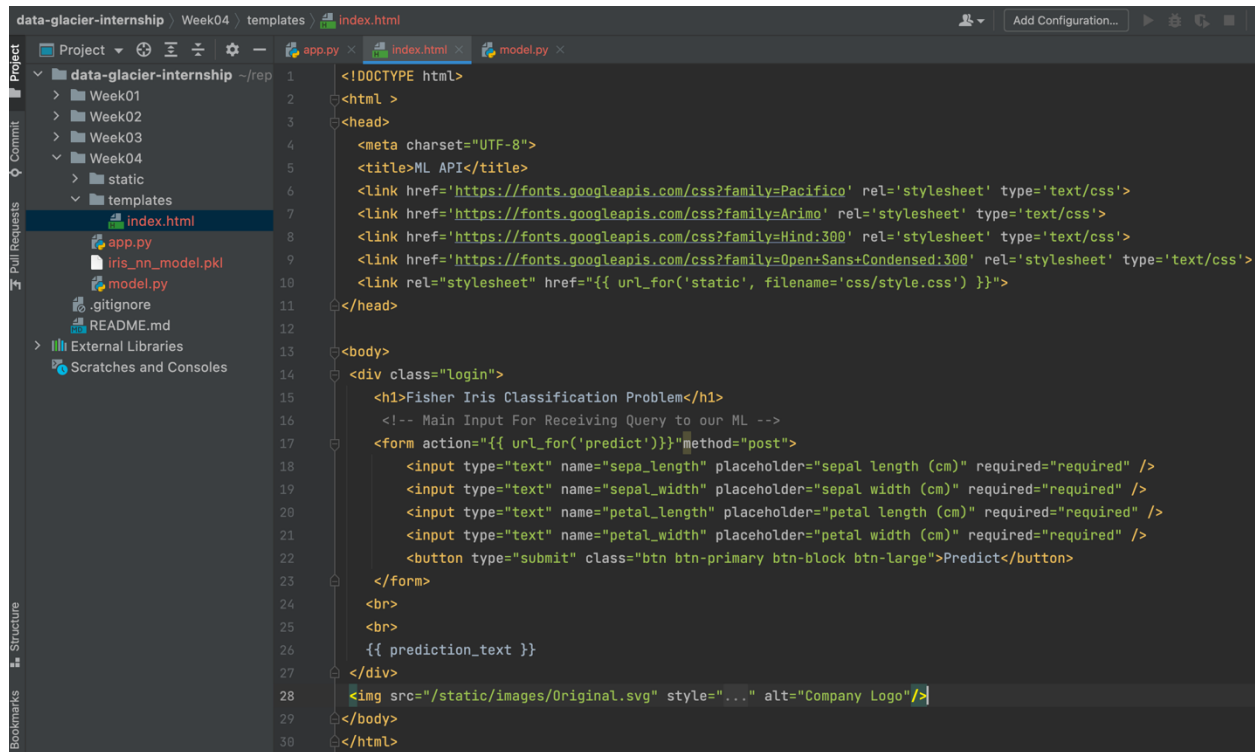
app.py  
iris\_nn\_model.pkl  
model.py  
static  
templates

```
Week04 — -zsh — 59x24  
Last login: Wed Dec 27 10:48:52 on ttys001  
[(base) anastasiia@MBP-Anastasiia Week04 % python model.py ]  
[[ 8  0  0]  
 [ 0 12  0]  
 [ 0  1  9]]  
  
              precision    recall  f1-score   support  
  
         0         1.00      1.00      1.00         8  
         1         0.92      1.00      0.96        12  
         2         1.00      0.90      0.95        10  
  
   accuracy               0.97               30  
  macro avg              0.97              0.97               30  
weighted avg              0.97              0.97               30  
  
Pickling completed  
(base) anastasiia@MBP-Anastasiia Week04 %
```

Now we have file `iris_nn_model.pkl`

### 3. Deploy the model on flask (web app).

Create the file index.html for the model.



```
1 <!DOCTYPE html>
2 <html>
3 <head>
4   <meta charset="UTF-8">
5   <title>ML API</title>
6   <link href='https://fonts.googleapis.com/css?family=Pacifico' rel='stylesheet' type='text/css'>
7   <link href='https://fonts.googleapis.com/css?family=Arimo' rel='stylesheet' type='text/css'>
8   <link href='https://fonts.googleapis.com/css?family=Hind:300' rel='stylesheet' type='text/css'>
9   <link href='https://fonts.googleapis.com/css?family=Open+Sans+Condensed:300' rel='stylesheet' type='text/css'>
10  <link rel="stylesheet" href="{{ url_for('static', filename='css/style.css') }}">
11 </head>
12
13 <body>
14   <div class="login">
15     <h1>Fisher Iris Classification Problem</h1>
16     <!-- Main Input For Receiving Query to our ML -->
17     <form action="{{ url_for('predict') }}" method="post">
18       <input type="text" name="sepa_length" placeholder="sepal length (cm)" required="required" />
19       <input type="text" name="sepal_width" placeholder="sepal width (cm)" required="required" />
20       <input type="text" name="petal_length" placeholder="petal length (cm)" required="required" />
21       <input type="text" name="petal_width" placeholder="petal width (cm)" required="required" />
22       <button type="submit" class="btn btn-primary btn-block btn-large">Predict</button>
23     </form>
24     <br>
25     <br>
26     {{ prediction_text }}
27   </div>
28   
29 </body>
30 </html>
```

We use deserialization with Pickle.

```
from flask import Flask, render_template, request
import pickle
import numpy as np

app = Flask(__name__)
model = pickle.load(open('iris_nn_model.pkl', 'rb'))

@app.route('/')
def home():
    return render_template('index.html')

@app.route('/predict', methods=['POST'])
def predict():
    # for rendering results on HTML GUI
    int_features = [float(x) for x in request.form.values()]
    final_features = [np.array(int_features)]
    prediction = model.predict(final_features)

    iris_names = ['Setosa', 'Versicolor', 'Virginica']
    predicted_name = iris_names[int(prediction[0])]
```

```
    return render_template('index.html', prediction_text='Predicted Iris  
Species: {}'.format(predicted_name))  
  
if __name__ == "__main__":  
    app.run(port=5000, debug=True)
```

## In Terminal:

```
[(base) anastasiia@MBP-Anastasiia Week04 % python app.py  
* Serving Flask app 'app'  
* Debug mode: on  
WARNING: This is a development server. Do not use it in a production deployment.  
Use a production WSGI server instead.  
* Running on http://127.0.0.1:5000  
Press CTRL+C to quit  
* Restarting with watchdog (fsevents)  
* Debugger is active!  
* Debugger PIN: 515-690-735
```


Copy: <http://127.0.0.1:5000>

← → ↻ ⓘ 127.0.0.1:5000/predict ☆

## Fisher Iris Classification Problem

Predict

Predicted Iris Species: Versicolor

**Data Glacier**  
Your Deep Learning Partner