Flask 실습 자료 (2)

1. Flask 에서 사용할 모델 학습 및 저장

- Flask 는 사용하기 쉽고, 간단한 기능을 가볍게 구현하기에 적합하기 때문에 대부분의 ML Model 의 첫 배포 Step 으로 자주 사용하는 Framework 중 하나입니다.
- 이번 시간에는 iris data 를 사용한 간단한 classification model 을 학습한 뒤, 모델을 pickle 파일로 저장하고, Flask 를 사용해 해당 파일을 load 하여 predict 하는 server 를 구현할 것입니다.
- 그 이후, 해당 server 를 run 하여 직접 http request 를 요청하여 정상적으로 response 가 반환되는지 확인할 것입니다.

1) Sample code

- 여러분이 이전에 학습하신 모델이 준비되어있다면 어떤 모델이든 사용하셔도 좋습니다.

▼ Sample python code

- requirements
    - scikit-learn==1.0

▼ 소스코드 (왼쪽 화살표를 클릭하시면 코드를 볼 수 있습니다.)

```
import os
import pickle

from sklearn.datasets import load_iris
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import accuracy_score, classification_report
from sklearn.model_selection import train_test_split

RANDOM_SEED = 1234

# STEP 1) data load
data = load_iris()

# STEP 2) data split
X = data['data']
y = data['target']

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3,
                                                    random_state=RANDOM_SEED)

# STEP 3) train model
model = RandomForestClassifier(n_estimators=300, random_state=RANDOM_SEED)
model.fit(X_train, y_train)

# STEP 4) evaluate model
print(f"Accuracy :  {accuracy_score(y_test, model.predict(X_test))}")
print(classification_report(y_test, model.predict(X_test)))

# STEP 5) save model to ./build/model.pkl
os.makedirs("./build", exist_ok=True)
pickle.dump(model, open('./build/model.pkl', 'wb'))
```

2) 모델 학습 및 저장

- 위의 python code 를 복사한 후, 실행시킵니다.

```
cd flask-tutorial

# 파이썬 버전을 확인합니다.
python -V
```

```
# requirements 를 설치합니다.
pip install scikit-learn==1.0

# 위의 코드를 복사 후 붙여넣습니다.
vi train.py

# 위의 코드를 실행시킵니다.
python train.py
```

- 다음과 같은 메시지가 출력되는 것을 확인합니다.

```
Accuracy :  0.9555555555555556
              precision    recall  f1-score   support

           0       1.00      1.00      1.00        16
           1       0.94      0.94      0.94        17
           2       0.92      0.92      0.92        12

    accuracy                           0.96        45
   macro avg       0.95      0.95      0.95        45
weighted avg       0.96      0.96      0.96        45
```

- `build` 디렉토리 내부에 `model.pkl` 파일이 생성되는 것을 확인합니다.

```
cd build

ls
# model.pkl 파일 존재
```

---

2. Flask server 구현

- 1.에서 학습 후 저장했던 모델(pickle 파일)을 load 하여, POST /predict API 를 제공하는 Flask Server 를 구현합니다.

▼ Sample code

```
import pickle

import numpy as np
from flask import Flask, jsonify, request

# 지난 시간에 학습한 모델 파일을 불러옵니다.
model = pickle.load(open('./build/model.pkl', 'rb'))

# Flask Server 를 구현합니다.
app = Flask(__name__)


# POST /predict 라는 API 를 구현합니다.
@app.route('/predict', methods=['POST'])
def make_predict():
    # API Request Body 를 python dictionary object 로 변환합니다.
    request_body = request.get_json(force=True)

    # request body 를 model 의 형식에 맞게 변환합니다.
    X_test = [request_body['sepal_length'], request_body['sepal_width'],
              request_body['petal_length'], request_body['petal_width']]
    X_test = np.array(X_test)
    X_test = X_test.reshape(1, -1)

    # model 의 predict 함수를 호출하여, prediction 값을 구합니다.
    y_test = model.predict(X_test)

    # prediction 값을 json 화합니다.
    response_body = jsonify(result=y_test.tolist())

    # predict 결과를 담아 API Response Body 를 return 합니다.
    return response_body


if __name__ == '__main__':
    app.run(port=5000, debug=True)
```

```
# 파이썬 버전을 확인합니다.
python -V

# requirements 를 설치합니다.
pip install scikit-learn==1.0

# 위의 코드를 복사 후 붙여넣습니다.
vi flask_server.py
```

---

3. API 테스트

- 위의 Flask server 를 run 합니다.
  - `python flask_server.py`
- 다음과 같은 메시지가 출력되면 정상적으로 Flask server 가 동작하는 것을 의미합니다.

```
FLASK_APP = flask_server.py
FLASK_ENV = development
FLASK_DEBUG = 0
In folder /fast-campus-demo/flask-tutorial
/.pyenv/versions/flask-tutorial/bin/python -m flask run
 * Serving Flask app 'flask_server.py' (lazy loading)
 * Environment: development
 * Debug mode: off
 * Running on http://127.0.0.1:5000/ (Press CTRL+C to quit)
```

  - `http://127.0.0.1:5000/` 가 flask server 의 주소를 의미합니다.
- 해당 Flask server 에 `POST /predict` API 를 요청하여, 어떤 결과가 반환되는지 확인합니다.

```
curl -X POST -H "Content-Type:application/json" --data '{"sepal_length": 5.9, "sepal_width": 3.0, "petal_length": 5.1, "petal_width":

# {"result":[2]}
# 0, 1, 2  중의 하나의 type 으로 classification 하게 됩니다.
```