flask-1

flask-1

Flask는 파이썬으로 구현된 웹 프레임워크로, 마이크로(micro) 웹 프레임워크라고도 불립니다. **Flask**는 다양한 확장 기능을 지원하며, 간단한 구조와 유연한 설계로 인해 많은 개발자들에게 인기가 있습니다. **Flask**는 모듈화가 잘 되어 있어 필요한 부분만 선택하여 사용할 수 있습니다.

Flask의 특징은 다음과 같습니다.

- 가볍고 간결한 설계
- 확장성이 높은 구조
- Jinja2 템플릿 엔진을 이용한 간편한 HTML 렌더링
- RESTful 요청을 지원하는데 필요한 메서드와 데코레이터를 내장

Model Serving

flask -> 웹서비스 프레임워크 API + front -> container(x)

fastapi -> backend + API 를 생성 -> container(x)

bentoml -> ML/DL + API 만 생성 -> container(x)

kfserving -> ML/DL + API 만 생성 + container(o) (단점, 쿠버네티스를 설치해야 한다.)

flask

- MSA(마이크로 서비스 아키텍쳐)

- 예) 서비스 개발 : 이미지를 입력받아서 학습시킨 모델을 이용해 추론후 결과를 출력

-> 이미지를 입력(프론트앤드) <-> 전처리 <-> model serving (추론)

- vs Django

- 장고에 비해 가볍고, 쉽다.

- 단점, 장고에 비해 기능이 없다.

flask

conda install -c conda-forge flask

mkdir flask-tuto

cd flask-tuto

code requirements.txt

Flask==2.2.3

Decorator

쇼핑몰 게시글 작성

- 로그인 0-> 게시글 작성

def 로그인():

@로그인()

def 게시글():

실습

curl 로 POST 요청을 받고 이름을 출력하는 function을 만들어 보세요

오민엽 이고 39세 입니다.

from flask import request

request.get_json()

curl -X POST -H "Content-Type: application/json" --data '{"name" : "오민엽", "age":"39"}' http://127.0.0.1:5000/user # json 형식으로 데이터를 전달함을 명시 한다.

오민엽 이고 39세 입니다.

실습

- requirements.txt

```
Flask==2.2.3
scikit-learn==1.2.1
matplotlib==3.7.1
```

```
from flask import Flask, request
import json

app = Flask(__name__)

@app.route("/user", methods=["POST"])  # route : 경로를 의미 한다.  GET 과 POST 요청을 받는 주소
def user():
    request_body = request.get_json()
    age = request_body["age"]
    name = request_body["name"]
    result = json.dumps({"result":f"나의 이름은 {name} 이고, 나이는 {age}입니다."}, ensure_ascii=False)
    return result

@app.route("/") # route Decorator 는 python 함수를 web server URI 에 mapping 시키는 역할  https://localhost/
def hello_world():
    return "<h1> Hello world</h1>"
```

```
@app.route("/multi13")  # route : 경로를 의미 한다.
def hello_multi():
    return "<h1> Hello multi13</h1>"

@app.route("/predict", methods=["POST","GET"])  # route : 경로를 의미 한다.  GET 과 POST 요청을 받는 주소
def inference():
    # model ~~~
    result = json.dumps({"predict":"[0.0001,0.002, 0.0051, 0.9]"})
    return result

if __name__=="__main__":
    app.run(debug=True, host="127.0.0.1", port=5000)
```

```
> python app.py
```

- client

```
> curl -X POST http://127.0.0.1:5000/predict
> curl -X GET http://127.0.0.1:5000/predict
> curl -X POST -H "Content-Type: application/json" --data '{"name" : "오민엽", "age":"39"}' http://127.0.0.1:5000/user
```

- model 학습후 저장

```
import sklearn
import numpy as np
import matplotlib.pyplot as plt
from sklearn.tree import DecisionTreeRegressor
import joblib

# Create a random dataset
rng = np.random.RandomState(1)
X = np.sort(200 * rng.rand(100, 1) - 100, axis=0)
y = np.array([np.pi * np.sin(X).ravel(), np.pi * np.cos(X).ravel()]).T
y[::5, :] += 0.5 - rng.rand(20, 2)

# Fit regression model
regr_1 = DecisionTreeRegressor(max_depth=2)
regr_2 = DecisionTreeRegressor(max_depth=5)
regr_3 = DecisionTreeRegressor(max_depth=8)
regr_1.fit(X, y)
regr_2.fit(X, y)
regr_3.fit(X, y)

# Predict
X_test = np.arange(-100.0, 100.0, 0.01)[:, np.newaxis]
y_1 = regr_1.predict(X_test)
y_2 = regr_2.predict(X_test)
y_3 = regr_3.predict(X_test)

model_path="model.joblib"
joblib.dump(regr_3, model_path)
```

- 학습된 model을 이용해 추론 test

```
import sklearn
import joblib
import numpy as np

from flask import Flask, request, jsonify
import json

def load_model():
    file_name = "model.joblib"
    model = joblib.load(file_name)
    return model

model = load_model()

app = Flask(__name__)

@app.route("/predict", methods=["POST","GET"])  # route : 경로를 의미 한다.  GET 과 POST 요청을 받는 주소
def inference():
    X_test = np.arange(-100.0, 100.0, 0.01)[:, np.newaxis]
    y_1 = model.predict(X_test)
    response = jsonify(result=y_1.(tolist))
    return response

if __name__ == "__main__":
    app.run(debug=True, host="127.0.0.1", port=5000)
```