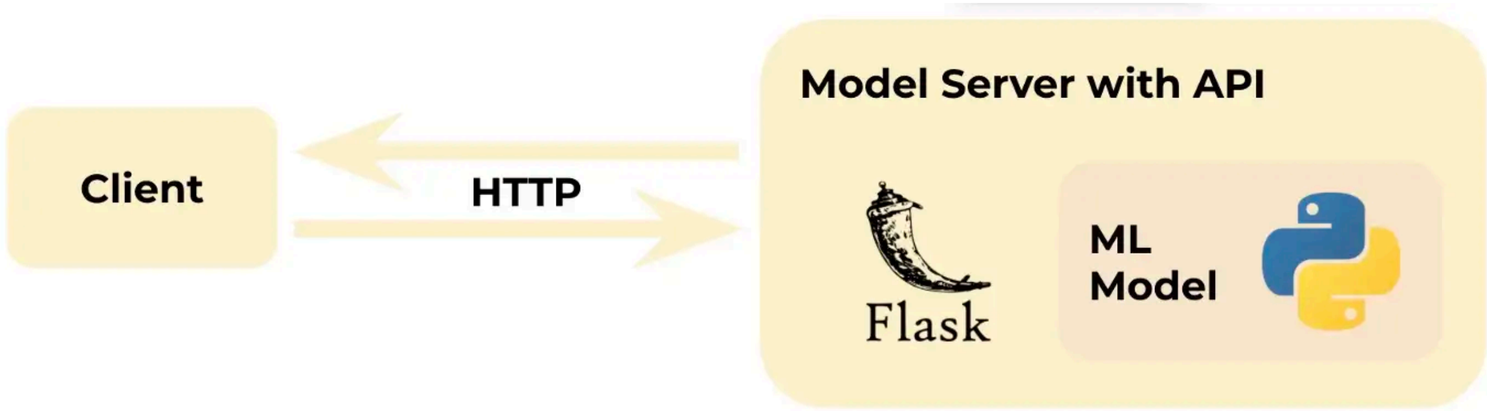
기계학습 모델 생성 및 테스트

회귀모델



기계학습 모델
배포 과정

- 가상환경 구축

  $ virtualenv -p python3 flaskapp

  $ source flaskapp/bin/activate

- flask 프레임워크(framework) 설치

  $ pip install wheel

  $ pip install flask

  $ pip install uwsgi

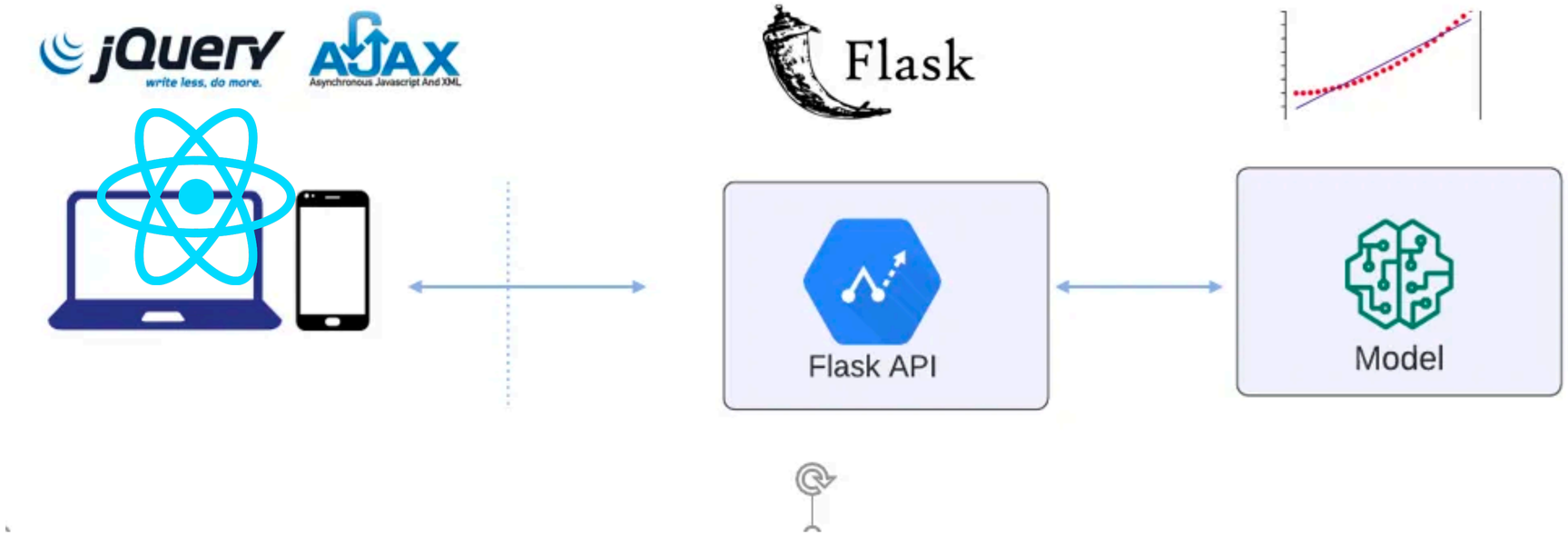
  $ pip freeze > requirements.txt

- 기계학습 모형개발
- 개발된 기계학습 모형 배포

기계학습 모델

배포 과정

- 클라이언트는 React 웹응용으로 구성하여 진행함



flask 웹응용 작성

시작하기

- 작성된 app.py

```
from flask import Flask

app = Flask(__name__)

@app.route("/")

def index():

    return "<h1>헬로 Flask!!!</h1>"

if __name__ == "__main__":

    app.run(host='0.0.0.0')
```

- flask 웹응용 실행

```
$ python app.py
```

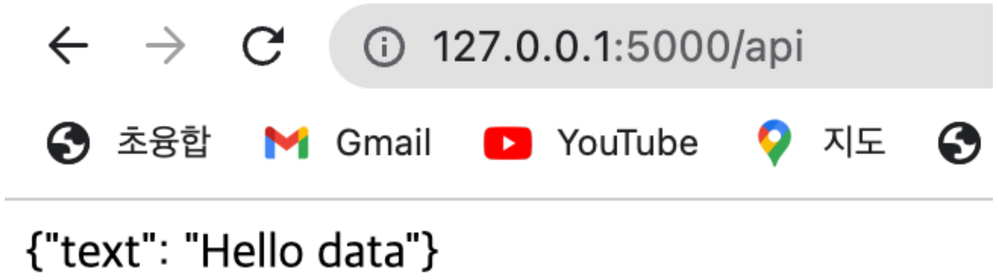
flask 웹응용 작성

시작하기

- flask 웹응용 테스트하기
  - 웹 브라우저 주소창에 URL 입력, http://0.0.0.0:5000/

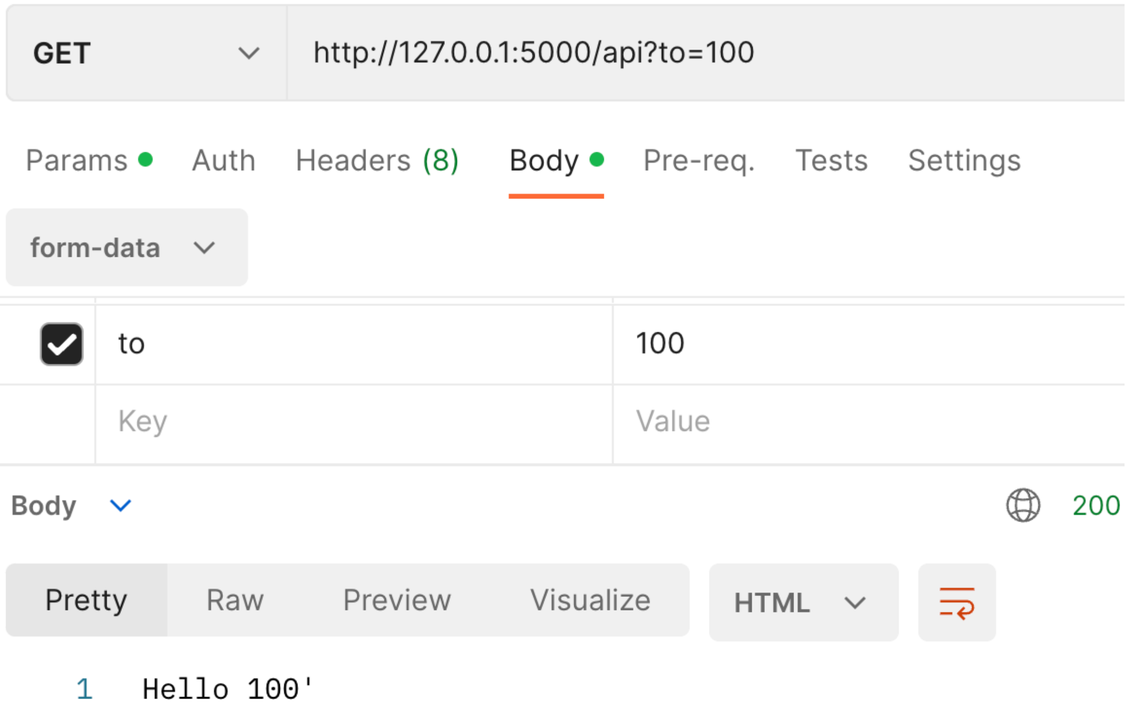
flask 웹응용 작성
시작하기

```
from flask import Flask
import json


app = Flask(__name__)


@app.route('/api')
def api():
    d = {"text": "Hello data"}
    return json.dumps(d)


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



flask 웹응용 작성
시작하기

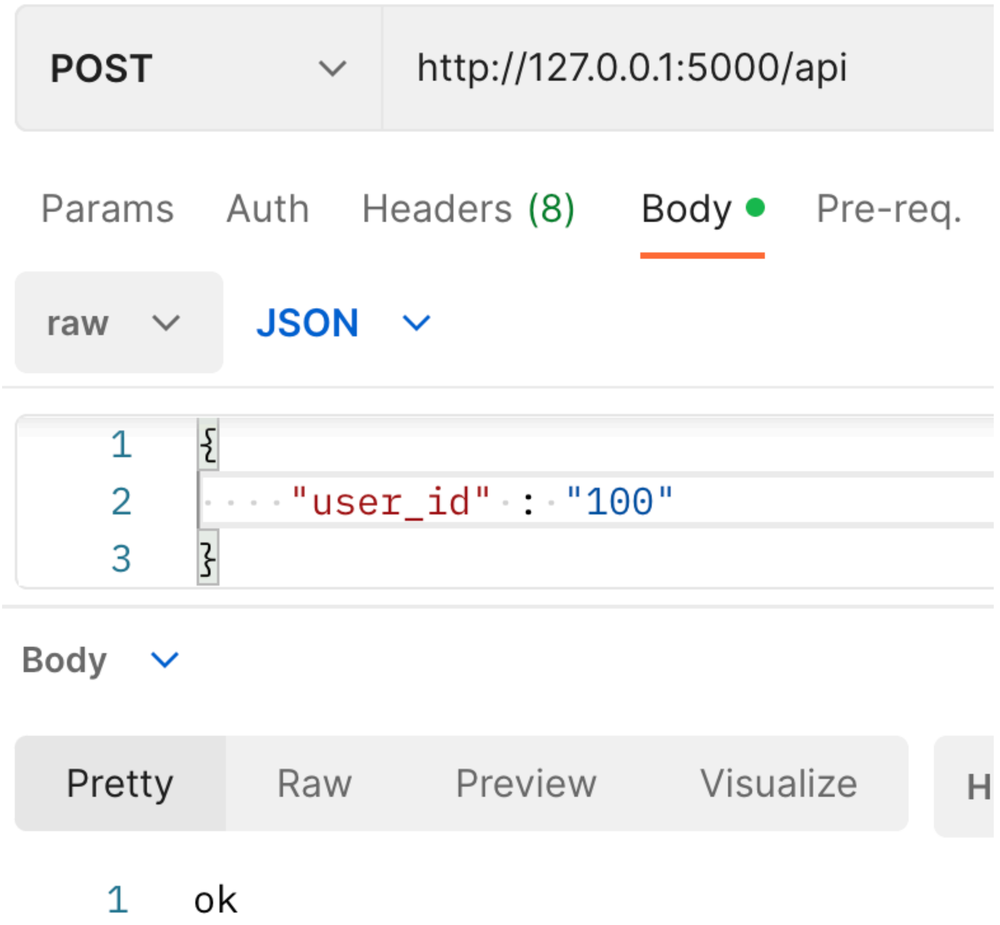
```
from flask import Flask
from flask import request
app = Flask(__name__)
@app.route('/api')
def api():
    to = request.form.get('to')
    return "Hello {}'".format(to)
if __name__ == "__main__":
    app.run(host='127.0.0.1', port=5000, debug=False)
```



flask 웹응용 작성

json 자료전달

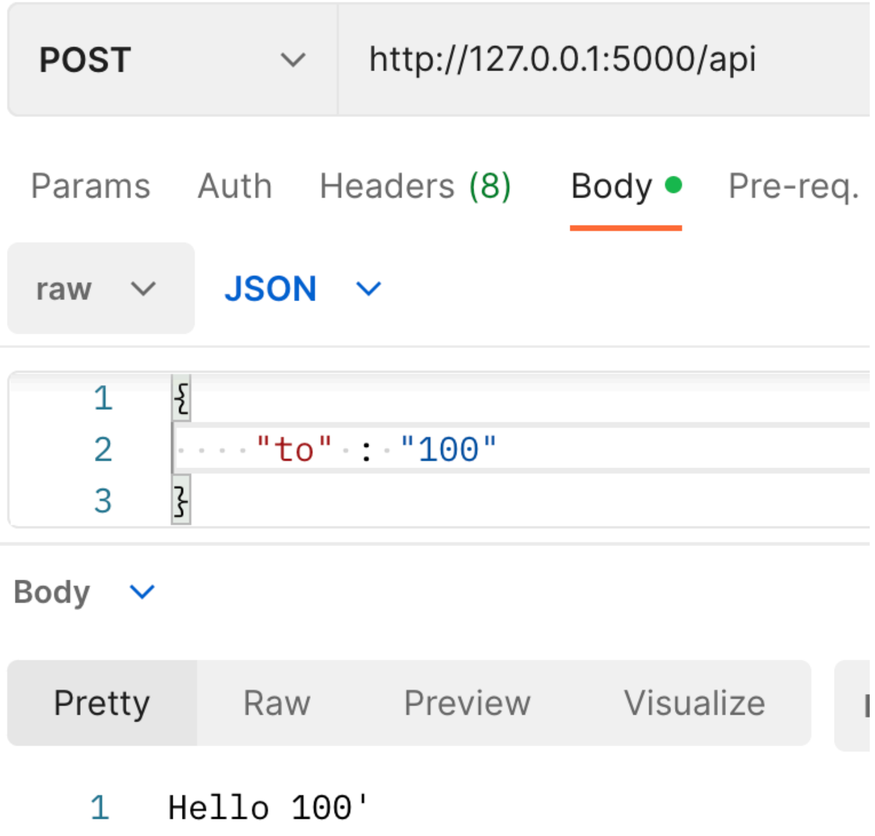
```
@app.route('/api', methods=['POST'])
def api():
    print(request.is_json)
    params = request.get_json()
    print(params['user_id'])
    return 'ok'
```



flask 웹응용 작성

json 자료전달

```
from flask import Flask
from flask import request


app = Flask(__name__)
@app.route('/api', methods=['post'])
def index():
    d = request.get_json()
    to = d['to']
    return "Hello {}'".format(to)


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



기계학습 모형 배포

개요

- 연봉 데이터를 바탕으로 근무연한에 따라 연봉을 예측하는 회귀분석 모형을 개발
  - 회귀분석 모형을 model.py로 작성
  - 예측모형을 피클 파일형식으로 model/salary_model.pkl 디렉토리에 저장
- 회귀모델 서비스 제공 API 서버 개발
  - 회귀모델 서비스를 위해 server.py로 작성
  - predict() 함수를 개발하여 근무연수(exp)를 넣게 되면 연봉을 예측
  - ‘model/salary_model.pkl’ 파일을 로딩후 API 요청을 처리함
- 예측모형 서비스 호출
  - 클라이언트 request.py를 실행시켜 연봉예측 결과를 얻도록 실행

기계학습 모형 배포
모델 저장

- pickle 모듈
  - 객체를 파일로 저장한 후 필요시 로딩하여 사용

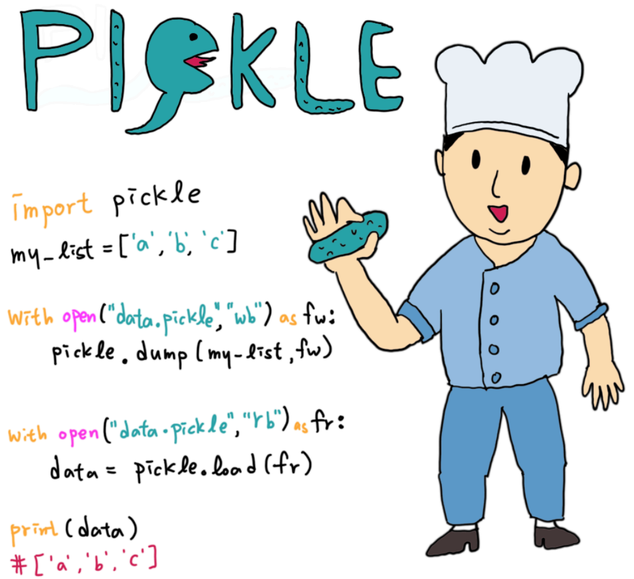
`pickle` — Python object serialization

Source code: Lib/pickle.py

The `pickle` module implements binary protocols for serializing and de-serializing a Python object structure. *"Pickling"* is the process whereby a Python object hierarchy is converted into a byte stream, and *"unpickling"* is the inverse operation, whereby a byte stream (from a binary file or bytes-like object) is converted back into an object hierarchy. Pickling (and unpickling) is alternatively known as "serialization", "marshalling," [1] or "flattening"; however, to avoid confusion, the terms used here are "pickling" and "unpickling".

기계학습 모형 배포
모델 저장

- pickle 모듈
  - 파이썬 객체 구조의 직렬화와 역 직렬화를 위한 바이너리 프로토콜을 구현
  - 직렬화 dump() 함수를 호출
  - 역직렬화하려면 load() 함수를 호출



기계학습 모형 배포

예측모형 개발

```
## model.py

import numpy as np

import pandas as pd

from sklearn.model_selection import train_test_split

from sklearn.linear_model import LinearRegression

import pickle

import requests

import json
```

기계학습 모형 배포

예측모형 개발

```
## model.py

dataset = pd.read_csv('data/Salary_Data.csv')

X = dataset.iloc[:, :-1].values

y = dataset.iloc[:, 1].values

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.33, random_state = 777)

salary_model = LinearRegression()

salary_model.fit(X_train, y_train)

y_pred = salary_model.predict(X_test)

pickle.dump(salary_model, open('model/salary_model.pkl','wb'))
```

기계학습 모형 배포

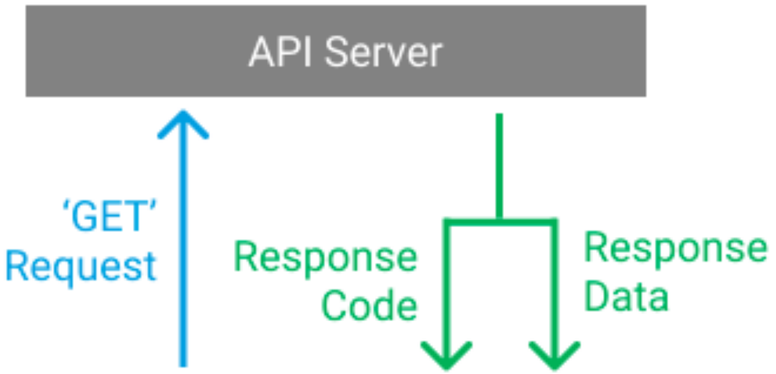
예측모형 응용서버

```
# server.py
import numpy as np
from flask import Flask, request, jsonify
import pickle

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

@app.route('/api',methods=['POST'])
def predict():
    data = request.get_json(force=True)
    prediction = model.predict([[np.array(data['exp'])]])
    output = prediction[0]
    return jsonify(output)

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

기계학습 모형 배포

requests

- API는 애플리케이션 소프트웨어를 구축하고 통합하기 위한 정의 및 프로토콜 세트을 제공
  - 정보 제공자와 정보 사용자 간의 계약
  - simple HTTP library for Python



기계학습 모형 배포

requests

- 설치 requests 버젼 확인

```
import requests

print(requests.__version__)
```

- GET요청

```
response = requests.get(URL)
```

```
import requests

response = requests.get('https://chercher.tech/sample/api/product/read')

print(response.json())
```

기계학습 모형 배포

requests

- 자료읽기

```
import requests

response = requests.get('https://chercher.tech/sample/api/product/read')

print(response.json())

json_data = response.json()

print ("** Response JSON ***")

print(json_data['records'][0])
```

기계학습 모형 배포

requests

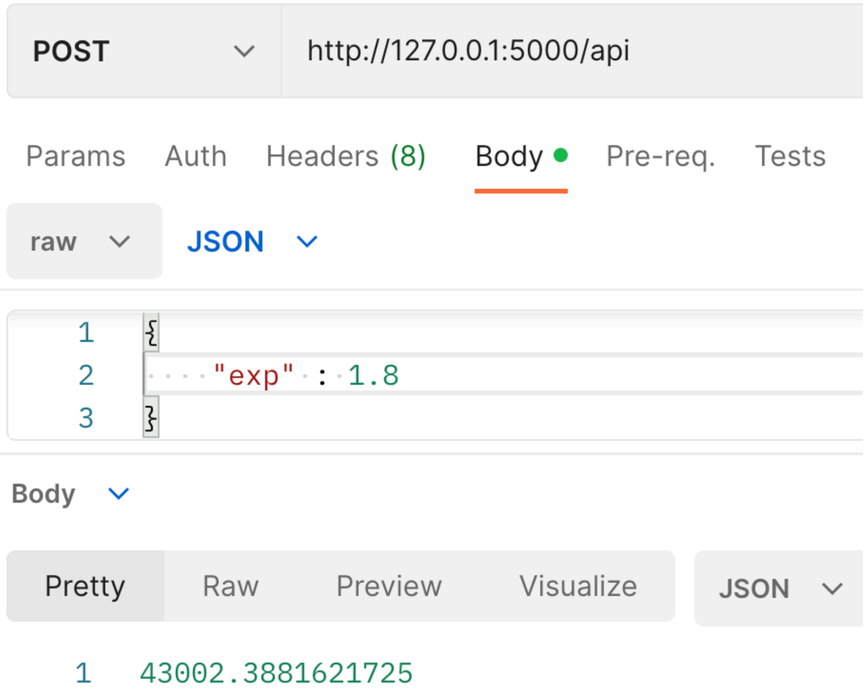
- 첫번째 자료의 가격(price) 정보를 확인한다

print(json_data['records'][0]['price'])

기계학습 모형 배포

예측모형 서비스 호출

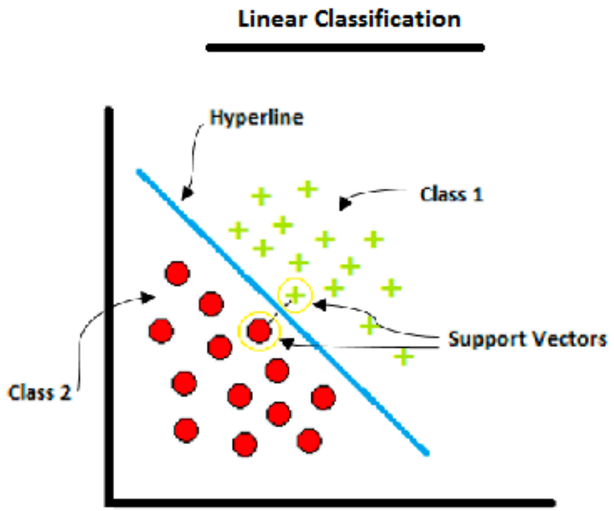
```
# request.py

import requests
url = 'http://localhost:5000/api'
r = requests.post(url, json={'exp':1.8,})
print(r.json())

$ python request.py
```

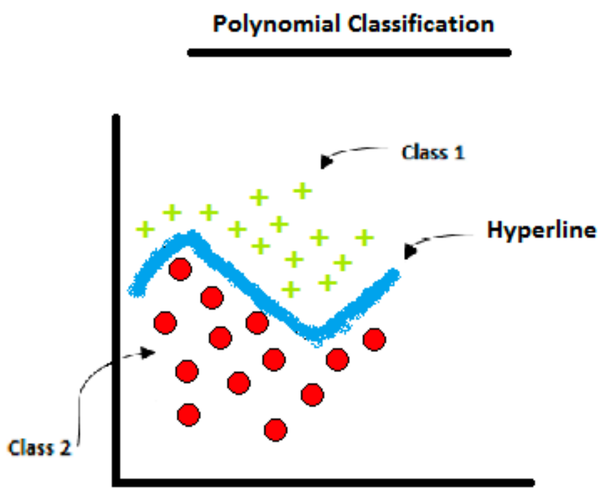


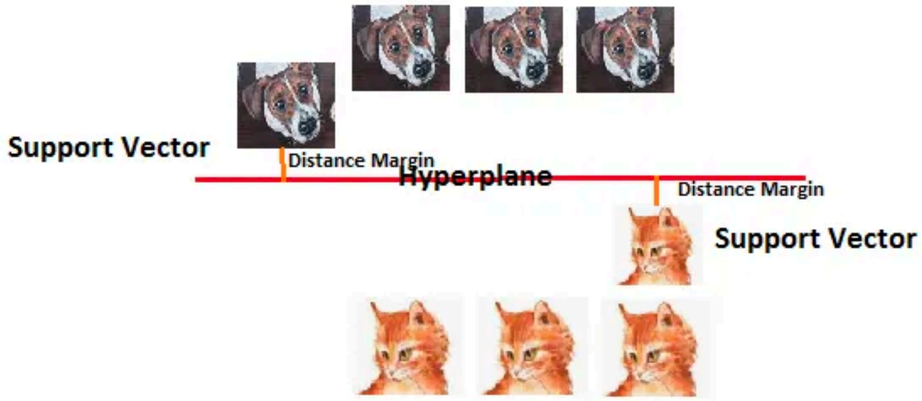
기계학습 모형 배포

SVM

- SVM







기계학습 모형 배포

SVM

```
##########데이터 로드
x_data = np.array([
    [2, 1],
    [3, 2],
    [3, 4],
    [5, 5],
    [7, 5],
    [2, 5],
    [8, 9],
    [9, 10],
    [6, 12],
    [9, 2],
    [6, 10],
    [2, 4]
])
y_data = np.array([0, 0, 0, 1, 1, 0, 1, 1, 1, 1, 1, 0])

labels = ['fail', 'pass']
```

기계학습 모형 배포

SVM

##########데이터 전처리

x_train, x_test, y_train, y_test = train_test_split(x_data, y_data, test_size=0.3, random_state=777, stratify=y_data)

##########모델 생성

model = SVC(probability=True)

##########모델 학습

model.fit(x_train, y_train)

##########모델 검증

print(model.score(x_train, y_train)) #

print(model.score(x_test, y_test)) #1.0

기계학습 모형 배포

SVM

```
##########모델 예측
x_test = np.array([
    [4, 6]
])
y_predict = model.predict(x_test)
label = labels[y_predict[0]]
y_predict = model.predict_proba(x_test)
confidence = y_predict[0][y_predict[0].argmax()]
print(label, confidence) #
```

기계학습 모형 배포

예측모형 개발

- svm_test_train.py

```
x_data = np.array([
    [2, 1],
    [3, 2],
    [3, 4],
    [5, 5],
    [7, 5],
    [2, 5],
    [8, 9],
    [9, 10],
    [6, 12],
    [9, 2],
    [6, 10],
    [2, 4]
])
y_data = np.array([0, 0, 0, 1, 1, 0, 1, 1, 1, 1, 1, 0])
labels = ['fail', 'pass']
```

기계학습 모형 배포

예측모형 개발

##########데이터 전처리

x_train, x_test, y_train, y_test = train_test_split(x_data, y_data, test_size=0.3, random_state=777, stratify=y_data)

##########모델 생성

model = SVC(probability=True)

##########모델 학습

model.fit(x_train, y_train)

기계학습 모형 배포

예측모형 개발

```
 ##########모델 검증

print(model.score(x_train, y_train)) #

print(model.score(x_test, y_test)) #1.0


############모델 저장

pickle.dump(model, open('./model/svm_test_model.pkl','wb'))
```

기계학습 모형 배포

예측모형 테스트

- svm_test_test.py

```
labels = ['fail', 'pass']
with open('./model/svm_test_model.pkl', 'rb') as f:
        model = pickle.load(f)
x_test = np.array([
    [4, 6]
])
y_predict = model.predict(x_test)
label = labels[y_predict[0]]
print (label)

y_predict = model.predict_proba(x_test)
confidence = y_predict[0][y_predict[0].argmax()]
print(label, confidence) #
```

기계학습 모형 배포

numpy 배열

- 배열 x는 12개의 원소를 갖는 1차원 배열로 구성

x = np.arange(12)

- 배열 x를 3행 4열의 2차원 배열로 구성함

x = x.reshape(3,4)

- 배열 x를 2개의 원소를 갖는 1차원 배열로 구성

x.reshape(-1,2)

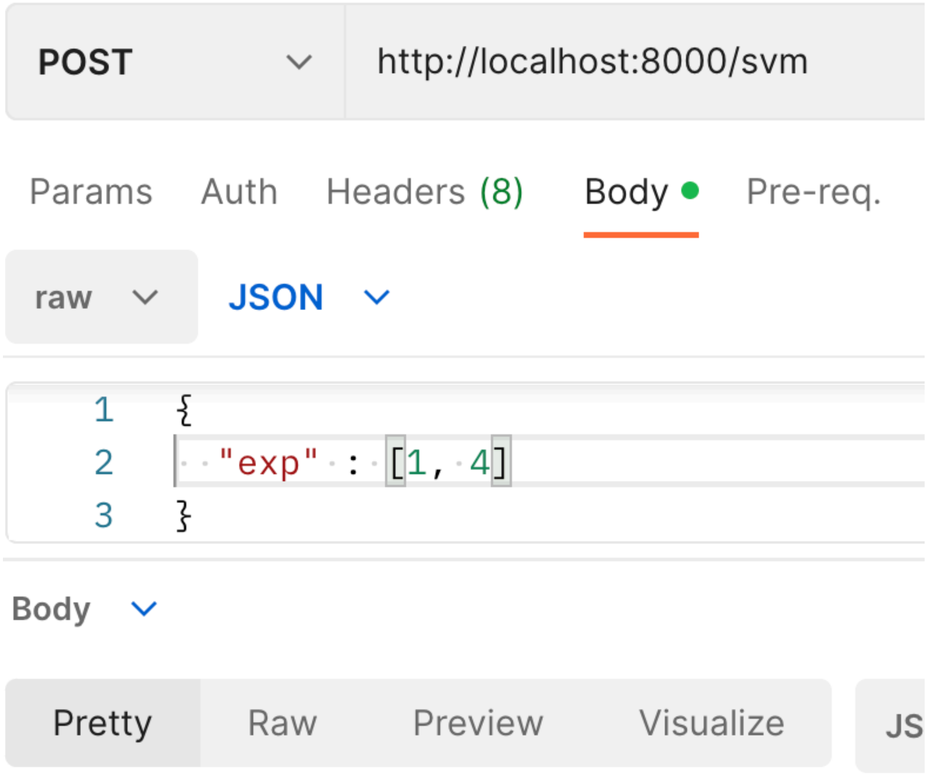
- reshape() 에서 -1은 “원래 배열의 길이와 남은 차원으로 부터 추정”
- 배열 x의 배열의 행을 알아서 지정함

기계학습 모형 배포

예측모형 테스트

- server.py

```
@app.route('/svm/',methods=['POST'])
def svm_predict():
    # Get the data from the POST request.
    data = request.get_json(force=True)
     X_new = np.array(data['exp'])
    X_new = X_new.reshape(-1,2)
    # Make prediction using model loaded from disk
    prediction = svm_model.predict(X_new)
    # Take the first value of prediction
    output = prediction[0]
    output = str(output)
    return jsonify(output)
```



SVM 모델 실습

기계학습 모형 배포

(실습) 예측모형 개발

```
# Importing the libraries

import numpy as np

import matplotlib.pyplot as plt

import pandas as pd


# Importing the dataset

dataset = pd.read_csv('Social_Network_Ads.csv')

X = dataset.iloc[:, [2, 3]].values

y = dataset.iloc[:, 4].values
```

기계학습 모형 배포

예측모형 개발

```
# Splitting the dataset into the Training set and Test set

from sklearn.model_selection import train_test_split

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.25, random_state = 0)


# Feature Scaling

from sklearn.preprocessing import StandardScaler

sc = StandardScaler()

X_train = sc.fit_transform(X_train)

X_test = sc.transform(X_test)
```

기계학습 모형 배포

예측모형 개발

```
# Fitting the classifier classifier to the Training set
from sklearn.svm import SVC
# 커널 함수, 새로운 특성 만들어 데이터를 상위 차원 공간으로 옮겨주는 함수
classifier = SVC(kernel = 'linear', random_state= 0)
classifier.fit(X_train, y_train)


#output:
SVC(C=1.0, cache_size=200, class_weight=None, coef0=0.0,
  decision_function_shape='ovr', degree=3, gamma='auto_deprecated',
  kernel='linear', max_iter=-1, probability=False, random_state=0,
  shrinking=True, tol=0.001, verbose=False)
```

기계학습 모형 배포

예측모형 개발

- 생성된 모델인 classifier 를 model 폴더에 svm_model.pkl로 저장하도록 프로그램을 확장하시오

기계학습 모형 배포

예측모형 테스트

```
# Predicting the Test set results

y_pred = classifier.predict(X_test)
```

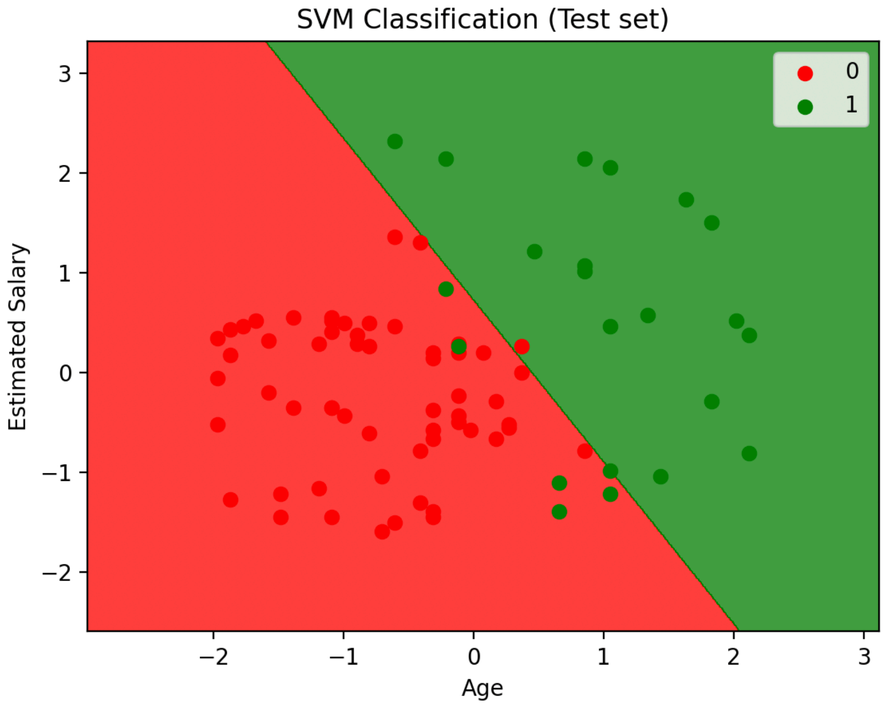
기계학습 모형 배포

예측모형 테스트

- model 폴더의 생성된 모델인 svm_model.pkl를 로딩하여 학습모델 테스트에 사용하도록 프로그램을 확장하시오

기계학습 모형 배포

- 예측결과 시각화



기계학습 모형 배포

예측모형 응용서버 및 테스트

- Flask 웹을 통해 생성된 모델에 대한 테스트를 진행하도록 프로그램을 작성하시오

Titanic 데이터셋 활용