

이번 시간에는 머신러닝 모델을 어떻게 API로 서비스하는지 배울것이다.



## 4. API to serve ML model

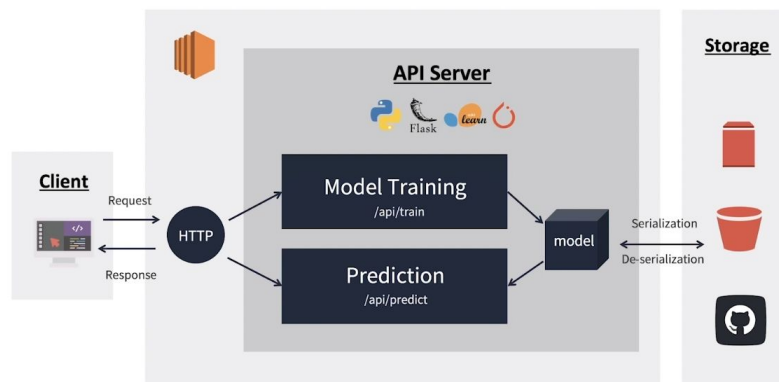
Architecture of API to serve ML model  
RESTful API for ML/DL model inference  
Practical process of machine learning  
Model Serving  
Serialization & De-serialization  
Skeleton of handler to serve model

4. API to serve ML model

52

### Architecture of API to serve ML model

AWS EC2와 Python Flask 기반 모델 학습 및 추론을 요청/응답하는 API 서버 개발

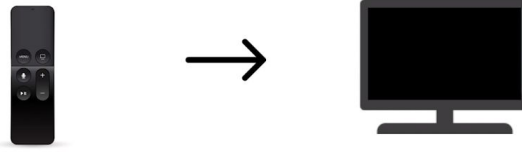


머신러닝 모델은 보통 클라이언트로 부터 입력된, 요청된 데이터를 통해서 이미 사전에 학습한 모델을 활용해서 추론을 하게 된다. 추론된 결과를 다시 응답을 하고 그 응답결과는 클라이언트에게 확인 할 수있는 형태가 된다. 본 실습에서는 Flask를 활용해서 API서버를 만드는 것을 진행하고 기본 시스템은 아마존의 EC2, storage는 EBS를 활용해서 간단하게 어떻게 하면 ML모델을 서비스 할 수있는지 아키텍처를 구성해 볼 것이다. 먼저 API를 설명하기 전에 간단한 기본개념부터 다루도록 한다.

## Interface

사용자는 기계와 소프트웨어를 제어하기 위해 인터페이스를 정해진 매뉴얼에 따라 활용하여 원하는 경험을 획득

컴퓨터의 마우스, 키보드와 같이 입력을 위한 인터페이스와 모니터나 프린터와 같이 정보를 받는 출력을 위한 인터페이스가 있음  
인터페이스는 상호 합의된 매뉴얼에 따라 적절한 입력을 받아 기대되는 출력을 제공할 수 있어야 함



Interface는 기계와 소프트웨어를 제어할 때 있어서 소통하기 위한 장치이다. 예를들어 리모컨을 사용해서 tv라는 기계 안에서 소프트웨어를 제어하게 된다. 컴퓨터도 마우스와 키보드 입력장치를 통해서 컴퓨터를 제어하게 된다. 이 인터페이스를 어떻게 설계할 지도 중요한 이슈중에 하나이다. 상호 합의된 매뉴얼이란 우리가 s사의 리모컨으로 s사의 tv만 제어할 수있으며 l사의 tv는 제어할 수없는것 처럼 특정 매뉴얼에 해당하는 것만 작동하게 설계한 것이다.

## API란?

Application Programming Interface의 약자로 기계와 기계, 소프트웨어와 소프트웨어 간의 커뮤니케이션을 위한 인터페이스를 의미

노드와 노드 간 데이터를 주고 받기 위한 인터페이스로, 사전에 정해진 정의에 따라 입력이 들어왔을 때 적절한 출력을 전달해야 함



API는 지정된 형식으로 소프트웨어간에 소통할 수 있도록 만들어진 인터페이스이다. 소프트웨어 간에 정보를 요청하고 받기 위해서는 교환을 위한 일정한 매뉴얼이 있어야 문제 없이 교환이 이루어 질 수있다.

예를들어 iMac, macbook, apple watch로 나의 스케줄을 관리한다고 했을 때 기계간에 일정한 인터페이스로 연결되어있지 않는다면 우리의 일정들이 동기화가 되기 어려울 것이다.

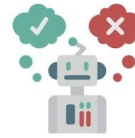
마찬가지로 우리가 브라우저를 통해서 어떤 웹 홈페이지나 사이트 또는 앱을 통해서 이용할때 웹 사이트에서 제공하는 API에 맞게 호출을 하지 않으면 결과를 제대로 받아올 수가 없을 것이다.

그래서 사전에 API를 정의하는 방식이 중요하고 머신러닝이나 딥러닝같은 모델을 제공할 때에도 마찬가지로 API서버에 어떻게 데이터를 입력하고 추론된 결과로의 응답을 어떻게 받는지가 중요한 약속, 규칙이 될 것이다.

## RESTful API for ML/DL model inference

REST 아키텍처를 따르는 API로 HTTP URI를 통해 자원을 명시하고 HTTP Method를 통해 필요한 연산을 요청하고 반환하는 API를 지칭

RESTful API는 데이터나 정보의 교환/요청 등을 위한 인터페이스를 REST 아키텍처를 따라 구현한 API  
일반적으로 데이터 값을 담아 요청하고 모델이 추론한 결과에 대한 return을 json 형태로 보통 반환하도록 설계  
RESTful API는 요청 메시지만 봐도 어떤 내용으로 되어있는지 알 수 있도록 표현됨



REST란 우리가 알 고있는 웹 동작방식과 유사한 일종의 프로토콜이다. 예를들어 클라이언트 웹 브라우저가 있고 접속할 웹 사이트에 대한 서버가 있다고 했을때 우리는 일반적으로 HTTP를 통해서 식별 URL주소를 입력하고 정보를 요청했을 때 서버는 브라우저의 서버에서 가지고 온 결과를 띄워서 사용자가 볼 수있게 한다.

REST는 기본적으로 웹의 기본 기술과 HTTP프로토콜을 그대로 활용하고 있기 때문에 웹의 장점을 최대한 활용할 수있는 아키텍처 스타일이다.

REST는 그 자체가 단어가 아니라 representational state transfer의 약자이다. 이는 REST가 서버의 자원, 리소스를 이름으로 구분해서 해당 자원의 상태를 주고 받기 때문에, 조금 기술적으로 얘기하자면 REST에서 HTTP URI(uniform resource identifier)가 자원을 명시하고 HTTP메서드를 통해서 해당 자원에 대해서 어떤 연산을 적용하는 것이 REST의 동작방식이다.

RESTAPI의 가장 큰 장점은 요청 메세지만 보고도 어떻게 AP가 동작하는지 이해할 수 있는 자체 표현구조로 되어있다. 보통 JSON형태로 REST방법을 요청하거나 결과를 받을 때 JSON형태로 많이 사용한다.

REST의 구성요소는 크게 3가지로 볼 수있는데 자원에 해당되는 리소스, URI 보통우리가 서버의 주소, 우리가 원하는 지원에 대한 명시된 URI가 있고 어떻게 받을지, HTTP 프로토콜과 관련된 방식인데 대체로 많이 사용하는 것이 get, post, put 같은 메서드, 마지막 구성요소로는 representation of resource 클라이언트와 서버가 어떻게 데이터를 주고 받을 것이냐 이다. ex) JSON, XML, 단순 TEXT, RSS, 최근에는 key, value형태의 JSON형태로 사용하는게 일반적이다.

RESTFUL API는 REST방식들을기반으로 API를 구현 한 것이라 볼 수있다. 기계간에 소프트웨어 간에 데이터나 정보의 교환, 요청등을 위해서 인터페이스를 REST architecture를 사용했다고 보면 된다.

동작방식이 HTTP표준을 기반으로 프로그래밍 언어로 쉽게 구현이 가능하고 시스템 분산, 확장에 용이해서 재사용이나 유지보수에 장점을 가진다.

그럼 RESTful은 무엇이나 REST라는 architecture를 사용하고있다. 정도로 보면 된다.

정리하자면 RESTAPI를 사용한다는 것은 어떤 정해진 규칙에 따라 API주소, REQUEST 구조, 리턴 구조를 만들어야 한다는 것이고 우리가 실습을 하는것은 RESTful API, 엄밀히 말하면 RESTAPI는 아니지만 이러한 REST의 아키텍처 스타일을 따르는 방식이라고 보면 된다.

간단히 RESTAPI 가 작동하는 방식을 살펴보자

만약 해커가 어떤 버그나 악성 프로그램이 숨겨진 데이터를 가지고 서버에 요청을 했다고 치자 근대 이 서버에서는 사용자가 보내온 데이터를 문제가 있는지, 어떤 해킹 코드가 있는지 프로그램이


있는지 이를 detecting하는 인공지능 모델이 있다고 하자 post로 이 서버에서 인공지능 서버에 머신러닝 딥러닝 서버 API에 이를 호출 했다고 하자

4. API to serve ML model55

### RESTful API for ML/DL model inference

REST 아키텍처를 따르는 API로 HTTP URI를 통해 자원을 명시하고 HTTP Method를 통해 필요한 연산을 요청하고 반환하는 API를 지칭

RESTful API는 데이터나 정보의 교환/요청 등을 위한 인터페이스를 REST 아키텍처를 따라 구현한 API  
일반적으로 데이터 값을 담아 요청하고 모델이 추론한 결과에 대한 return을 json 형태로 보통 반환하도록 설계  
RESTful API는 요청 메시지만 봐도 어떤 내용으로 되어있는지 알 수 있도록 표현됨



그 결과로서 이 모델이 비 정상적인 데이터라고 리턴을 주고 서버는 그 결과에 따라서 이 사용자에게 대한 접근이나 요청을 BLOCK할 수있다.

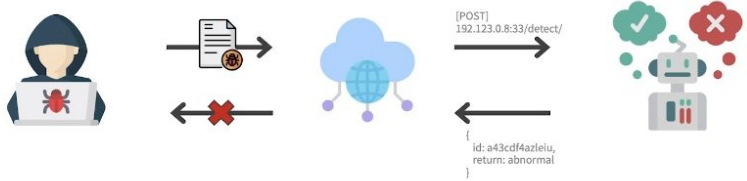
이렇게 RESTful API는 이때 여기서 클라이언트는 중간에 있는 클라우드 서비스가 될 것이다. 그리고 서버로서는 ML/DL 모델을 서빙하고 있는 API 서버가 클라이언트로 부터 요청을 받는 서버가 될 것이다.

4. API to serve ML model55

### RESTful API for ML/DL model inference

REST 아키텍처를 따르는 API로 HTTP URI를 통해 자원을 명시하고 HTTP Method를 통해 필요한 연산을 요청하고 반환하는 API를 지칭

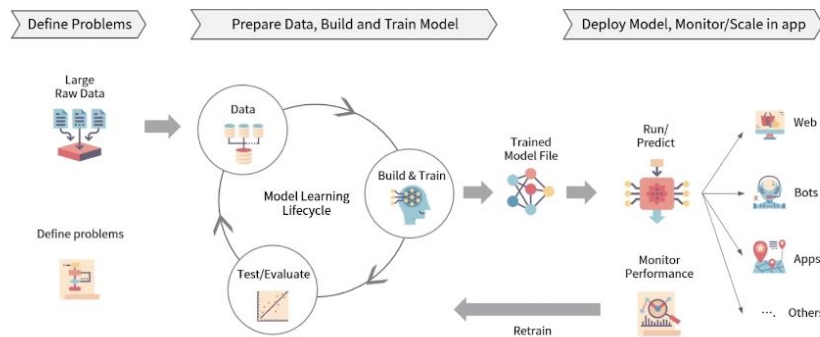
RESTful API는 데이터나 정보의 교환/요청 등을 위한 인터페이스를 REST 아키텍처를 따라 구현한 API  
일반적으로 데이터 값을 담아 요청하고 모델이 추론한 결과에 대한 return을 json 형태로 보통 반환하도록 설계  
RESTful API는 요청 메시지만 봐도 어떤 내용으로 되어있는지 알 수 있도록 표현됨



이번 강의에서 다루는 방식은 어떻게 하면 결과를 배포할 것인가에 대해서 중점적으로 다룰것이다.

## Practical process of machine learning

문제정의, 데이터준비, 모델 학습 및 검증, 모델 배포, 모니터링 등의 과정을 통해 실제 서비스에 기계학습 모델을 적용



## Model Serving

학습된 모델을 REST API 방식으로 배포하기 위해서 학습된 모델의 Serialization과 웹 프레임워크를 통해 배포 준비 필요

모델을 서빙할 때는 학습 시의 데이터 분포나 처리 방법과의 연속성 유지 필요

모델을 배포하는 환경에 따라 다양한 Serving Framework를 고려하여 활용

### Model Training

- Data preprocessing
- Model fitting
- Evaluation



### Serializing Model

- Save trained model



### Serving Model

- Load trained model
- Define inference
- Deployment

학습된 모델을 배포하기 위해서는 기존에 학습된 모델을 어떻게 직렬화(Serialization)를 하고 웹 프레임워크를 통해 배포를 할지 이러한 준비과정이 필요하다

학습한 모델을 직렬화 과정을 통해서 모델을 디스크에 저장하게 된다. 그리고 실제로 배포할 때에는

학습된, 직렬화된 모델을 불러오고 입력된 데이터에 대해서 추론을 하는 모델을 배포하게 된다. 과정을 실습을 통해 다룰 예정이다.

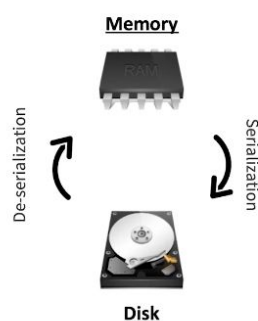
## Serialization & De-serialization

학습한 모델의 재사용 및 배포를 위해서 저장하고 불러오는 것

Serialization을 통해 ML/DL model object를 disk에 write하여 어디든 전송하고 불러올 수 있는 형태로 변환

De-Serialization을 통해 Python 혹은 다른 환경에서 model을 불러와 추론/학습에 사용

모델을 배포하는 환경을 고려해 환경에 맞는 올바른 방법으로 Serialization을 해야 De-serialization이 가능



Serialization은 통일된 방식으로 플랫폼에서도 이를 불러올수 있고 읽을 수있는 형태로 변환한다는 의미를 가지고 있음

De-Serialization은 앞에서 Serialization은 한 모델에 대해서 파이썬이나 다른 환경에서 불러와서 메모리에서 추론 또는 학습에 사용할 수있도록 하는것

최근에는 ONNX 프로젝트를 통해서 파이토치에서 학습한 모델을 다른 프레임워크에서 불러와서 inference를 하는 등 여기저기서 많이 사용하게되는, 공통적으로 어떤 하드웨어나 시스템 플랫폼에서도 사용할 수있도록 하는 일종의 단일 플랫폼 방식으로 Serialization하는 프로젝트이다. 최근에는 ONNX를 활용한 프로젝트가 굉장히 많이 진행되고 있고 실제로 모델을 배포 하는데 있어서 편리성도 있지만 속도의 향상된 부분도 존재한다고 한다.

## Skeleton of handler to serve model

```
class ModelHandler(BaseHandler):
    def __init__(self):
        ...

    def initialize(self, **kwargs):
        ...

    def preprocess(self, data):
        ...

    def inference(self, data):
        ...

    def postprocess(self, data):
        ...

    def handle(self, data):
        ...
```

모델을 서비스 하기 위해서는 우리가 Serialization한것을 어떻게 De-Serialization 할 것인지 우리가 모델을 학습할때 처리했던 방식들, 추론하는 방법, 추론된 결과를 어떻게 하면 우리가 API에서 정리한 인터페이스 방식에 따라서 어떻게 서비스할지 그런 것들을 다루는 핸들러를 개발하려고 한다.

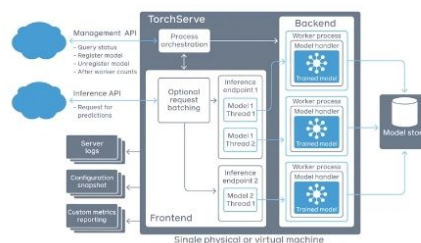
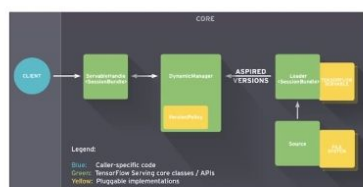
## Model serving을 위한 다양한 Frameworks

딥러닝 모델의 안정적인 serving을 위해 TensorFlow serving이나 TorchServe, TensorRT 같은 프레임워크를 사용하는 것이 일반적

Flask와 같은 웹프레임워크는 클라이언트로부터의 요청을 처리하기 위해 주로 사용

별도의 모델 추론을 위한 AP 서버를 운영하여 내부 혹은 외부 통신을 통해 예측/추론값 반환

대용량 데이터 배치처리와 딥러닝 모델의 활용이 늘면서 multi node, multi GPU 환경에서의 안정적인 모델 서빙을 위한



모델은 사실이 서빙을 위해서 다양한 프레임워크들이 존재, 딥러닝 모델은 서빙을 위해서 플라스크 같은 이제 micro web framework 보다는 서빙 자체의 좀 더 특화된 Tensorflow serving이나 TorchServe, 그리고 triton inference serving 같은 프레임워크를 사용하는 것이 일반적이다. 특히 GPU 사용에 있어서 이제 안정적으로 handel 하기 위해서는 이러한 전용 serving framework를 쓰는게 좋다

