FastAPI Tutorial Session

MLOps 에서 API 이해하기

튜토리얼 Part 1: 개념 설명

🎯 학습 목표

- API의 기본 개념과 필요성 이해
- HTTP 통신과 REST API 원칙 마스터
- MLOps 파이프라인에서 API의 역할 체험
- FastAPI로 ML 모델 서빙 구현

API 란 무엇인가?

Application Programming Interface

- 프로그램들이 서로 대화하는 방법
- 요청(Request)과 응답(Response)의 규칙

레스토랑 비유:

손님(클라이언트) → 메뉴판(API 문서) → 주문

↓

웨이터(API) → 주방(서버)

↓

음식(데이터) → 손님에게 전달

ML 모델의 경우:

사용자 → API 요청 → ML 모델 처리 → 예측 결과 반환

HTTP 통신 기초

📝 HTTP 요청(Request)의 구성

1. URL (어디로?): 예: https://api.mymodel.com/predict

2. Method (뭘 하고 싶어?)

   - GET: 데이터 조회, - POST: 데이터 처리/생성, - PUT: 데이터 수정, - DELETE: 데이터 삭제

3. Headers (메타정보)

   - Content-Type: application/json

   - Authorization: Bearer token123

4. Body (실제 데이터): {"features": [1.2, 3.4, 5.6]}

📥 HTTP 응답(Response)

Status Code: 200 (성공), 2404(없음), 500(에러)

Body: {"prediction": "positive", "score": 0.87}

REST API 설계 원칙

🎯 **REST(Representational State Transfer) 원칙을 따르는 API 설계 방식**

❌ **나쁜 설계 (동사 중심)**

- /getModelInfo
- /trainNewModel
- /makePrediction
- /deleteOldModel

✅ **좋은 설계 (명사 중심)**

- GET /models # 모델 목록 조회
- POST /models # 새 모델 생성
- GET /models/{id} # 특정 모델 조회
- DELETE /models/{id} # 모델 삭제
- POST /models/{id}/predict # 예측 요청

핵심 원칙:

- URL은 명사로 (자원 표현)
- 동작은 HTTP 메서드로
- 무상태성 유지
- 일관된 구조

MLOps 파이프라인에서 API

🔄 MLOps 전체 파이프라인

[데이터 수집] → [모델 학습] → [모델 검증] → [배포] → [서빙] → [모니터링]

↓ ↓ ↓ ↓ ↓ ↓

Data API Train API Valid API Deploy API Predict API Monitor API

각 API의 역할:

- **Data API**: 실시간 데이터 수집
- **Training API**: 재학습 트리거
- **Validation API**: 모델 성능 검증
- **Deploy API**: 모델 배포/롤백
- **Predict API**: 실제 예측 서비스
- **Monitor API**: 성능 모니터링

실제 사례: 구글 이미지 번역

1. 사진 업로드
   ↓
2. API: "이 이미지 번역해줘"
   ↓
3. ML 모델: 이미지 처리, 번역 수행
   ↓
4. API 응답: 사진, 번역된 내용
   ↓
5. 화면에 번역 표시

⏱️ 총 0.3초 | 📊 일일 10억 건

튜토리얼 Part 2: Fastapi 코드 실습

🎯 실습 목표

- FastAPI로 ML API 서버 구축
  - 붓꽃 분류 모델 서빙
- 실시간 예측 엔드포인트 구현
  - API 문서 자동 생성

FastAPI

🚀 FastAPI란?

현대적인 Python 웹 프레임워크

- 2018년 출시, 현재 가장 빠르게 성장하는 Python 프레임워크
- **속도**: Node.js, Go와 비슷한 성능 (Starlette + Pydantic 기반)
- **개발 속도**: 2-3배 빠른 개발, 40% 적은 버그

📊 빅테크 채택 현황

OpenAI → ChatGPT API, GPT-4 API 백엔드
Netflix → 내부 Crisis Management 도구
Microsoft → Azure 서비스 일부
Uber → ML 플랫폼 Ludwig

왜 FastAPI인가?

- **자동 문서화**: Swagger UI 자동 생성 (/docs)
- **타입 안정성**: Pydantic으로 입력/출력 자동 검증
- **비동기 지원**: async/await 네이티브 지원
- **표준 준수**: OpenAPI, JSON Schema 표준
- **IDE 지원**: 자동완성, 타입 체크 완벽 지원

OpenAI 퇴사자 회고록

Code

OpenAI uses a **giant monorepo** which is ~mostly Python (though there is a growing set of Rust services and a handful of Golang services sprinkled in for things like network proxies). This creates a lot of strange-looking code because there are so many ways you can write Python. You will encounter both libraries designed for scale from 10y Google veterans as well as throwaway Jupyter notebooks from newly-minted PhDs. Pretty much everything operates around FastAPI to create APIs and Pydantic for validation. But there aren't style guides enforced writ-large.

OpenAI **runs everything on Azure**. What's funny about this is there are exactly three services that I would consider trustworthy: Azure Kubernetes Service, CosmosDB (Azure's document storage), and BlobStore. There's no true equivalents of Dynamo,

https://calv.info/openai-reflections

API 만들기 사전 설치

pip install "fastapi[standard]" scikit-learn requests

API 만들기 사전 설치

pip install "fastapi[standard]" scikit-learn requests