김광훈

AI Service Engineer

유연하고 높은 성능의 AI 서비스를 개발하는 엔지니어입니다.

비즈니스 결과 산출을 위해 문제의 원인부터 방법까지 끊임없이 고민하고 개선하는 자세를 지니고 있습니다. AI 모델 개발부터 AI 서비스를 개발한 경험이 있습니다. **개인 맞춤형 학습 LLM 서비스를** 개발하고, **온프레미스 LLM 챗봇 시스템 서비스 최적화**를 통한 **메모리 80% 절약**, EMR 기반 질환 **예측 모델 성능 45% 개선** 등의 성과를 달성했습니다. 또한 조직 운영에 필요한 서비스를 개발하여 **상담전환율을 8.5배 향상**의 성과를 만들어낸 경험이 있습니다.

배움과 성장을 토대로 조직의 성장을 적극 도모합니다.

조직의 성장을 위해 항상 지식을 학습하는 시간을 가집니다. 높은 퍼포먼스를 위해 지속적인 학습과 신뢰의 문화를 추구하고 협업합니다. 이러한 과정에서 **개인 맞춤형 AI 학습 플랫폼과 온프레미스 RAG 시스템을 구축**했습니다. **Langchain과 FastAPI 기반 모듈화된 아키텍처**로 서비스를 개발하고, 도메인 특화 질의응답 최적화로 **검색 정확도 0.7, 응답품질 0.7을 달성**했습니다.

About

1993. 03. 28

https://kikiru328-portfolio.netlify.app/

kikiru328@gmail.com

https://github.com/kikiru328

Skiils

- Python, Langchain, FastAPI Framework, PyQt Framework
- PyTorch, Tensorflow, HuggingFace
- MySQL, Redis
- AWS, Docker, Prometheus, Grafana, Git

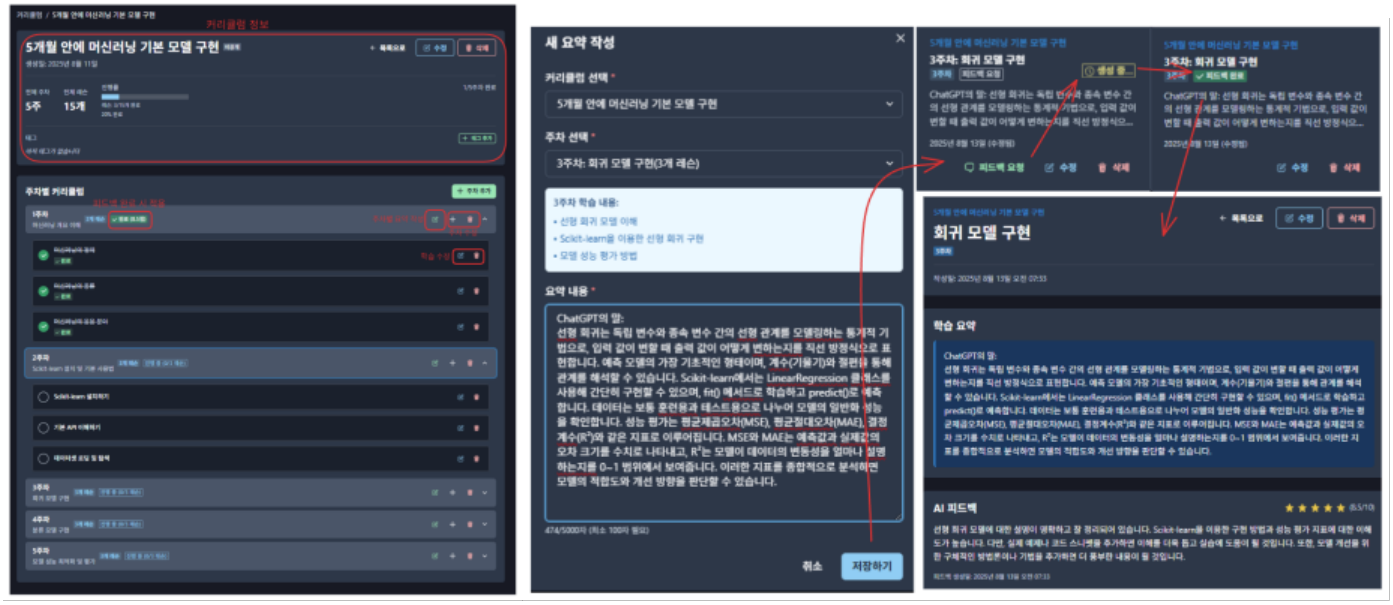
개인 프로젝트

LLearn: AI 학습 플랫폼 Github Docs

개발기간: 2025.06 - 2025.08 (2개월) | 역할 : AI Service 개발 | 기여도: 100%

Python 3.13+ FastAPI 0.116+ MySQL 8.0+ LangChain 0.3+ Prometheus Monitoring

전체 UI Demo



프로젝트 개요

- 핵심문제

기존 MOOC 플랫폼의 **낮은 완료율(5~15%)** 과 자기주도 학습의 한계를 AI 기술로 해결하기 위해 이 프로젝트를 기획하였습니다.
학습자들이 체계적인 학습 설계와 지속적인 피드백 부재로 인해 방향성을 상실하고 학습을 포기하는 문제를 근본적으로 해결하고자 했습니다.

- 솔루션 설계

AI 기반 개인화 멘토링 시스템을 구축하여 다음과 같은 학습 사이클을 자동화 하였습니다.

- **목표 설정 → AI 커리큘럼 생성 → 주차별 학습 → 요약 작성 → AI 피드백 → 개선 및 반복**

이를 통해 개인 맞춤형 학습 경험과 소셜 학습 환경을 제공하여 학습 동기를 지속적으로 유지할 수 있는 플랫폼을 개발하였습니다.

프로젝트 성과

- 시스템 아키텍처 완성도

Clean Architecture 기반 확장 가능한 시스템 구축을 통해 프로덕션 수준 서비스 완성
시스템 구성 현황

```
├─── API 엔드포인트: 101개 (7개 도메인 × 평균 14개)
├─── 데이터베이스: 13개 테이블, 완전한 관계형 설계
├─── 모듈 구조: 7개 독립 모듈 (Clean Architecture 4계층)
└─── 기능 완성도: 모든 CRUD 및 비즈니스 로직 구현
```

- AI 서비스 통합 성과

LangChain과 OpenAI API를 활용한 안정적인 AI 서비스를 구축

- 구조화된 응답 생성: JSON 파싱 성공률 98% 이상 달성
- 프롬프트 엔지니어링: 일관성 있는 커리큘럼 생성 시스템 구축
- 관측성 확보: Langfuse를 통한 LLM 성능 모니터링 및 비용 최적화

- 기술적 요소

1. 의존성 주입 시스템: Dependency-injector를 활용한 느슨한 결합 구조
2. 비동기 처리: SQLAlchemy Async + FastAPI로 높은 동시성 처리
3. 다층캐싱: Redis 기반 성능 최적화 및 실시간 데이터 관리
4. 모니터링 체계: Prometheus + Grafana 통합 관측성 구축

정량적 성과

- 개발 생산성

개발속도	1.7개 API/일 (총 101개 API)
코드 품질	95% A등급 (순환 복잡도 1-5)
테스트 완성도	404개 테스트, 93.6% 성공률

- LLM 성능

평균 응답 시간	6.07초
토큰 효율성	평균 1,200토큰/커리큘럼
모니터링	Langfuse 모니터링

문제 해결 과정

1) 확장 가능한 시스템 설계

상황: 다양한 도메인(AI, 소셜, 관리자)의 복합적 연동과 향후 MSA 전환을 고려한 설계가 필요하다고 판단
해결 과정:

1. Clean Architecture 4계층 구조 도입으로 명확한 책임 분리
2. 7개 독립 모듈 설계로 도메인 별 독립성 확보
3. 의존성 역전 원칙 적용하여 206개 포인트에서 느슨한 결합 구현

결과: MSA 전환 준비 완료, 유지보수성 극대화, 테스트 용이성 확보

2) LLM 통합 및 안정성 확보

상황: AI의 자유형식 응답을 구조화된 데이터로 안정적 변환, 토큰 사용량 최적화 필요
해결 과정:

1. Langchain 추상화 레이어 도입으로 다양한 LLM 제공자 지원
2. 구조화된 프롬프트 엔지니어링으로 JSON 스키마 기반 일관된 응답 유도
3. Pydantic 기반 런타임 검증으로 파싱 오류 방지
4. Langfuse 관측성 도구로 실시간 성능 모니터링

결과: LLM 호출 성공률 98% 이상, 안정적 서비스 제공

3) 고성능 비동기 처리

상황: AI 작업으로 인한 긴 처리 시간, 동시성 처리, 캐시 무효화 복잡성
해결 과정:

1. SQLAlchemy Async ORM으로 비동기 데이터베이스 I/O 개선
2. Redis 다층 캐싱 전략: 피드(5분), 소셜 메트릭(실시간), 통계(10분) 별도 관리
3. 커넥션 풀 최적화로 세션 라이프사이클 관리
4. 비동기 작업 분리로 사용자 경험 개선

결과: 높은 동시성 처리 능력 확보, I/O Bound 작업 최적화 완료

4) 테스트 용이성 확보

상황: 복잡한 외부 의존성(LLM, DB, Cache), 예측 불가능한 LLM 응답 테스트
해결 과정:

1. 컨테이너 기반 DI로 환경별 다른 구현체 주입
2. 인터페이스 기반 설계로 모든 외부 의존성 추상화
3. Mock 서비스 구현으로 테스트용 LLM, DB, Cache 제공
4. 단계별 테스트 전략: 단위 → 통합 → E2E 점진적 검증

결과: 404개 테스트 93.6% 성공률, TDD 개발 완성, 높은 코드 품질 확보

핵심 성취 요약

1) 기술적 완성도

- 프로덕션 수준: 즉시 상용화가 가능한 완전한 서비스 구현
- 코드 품질: 95% A 등급 달성
- 시스템 안정성: 멀티유저 동시성 처리 완료

2) 비즈니스 가치

- 시장 차별화: 기존 MOOC 한계를 AI 멘토링으로 해결할 수 있음
- 확장 가능성: B2C에서 B2B로 자연스러운 시장 확장 기반 마련
- 사용자 경험: 과학적 근거 기반 학습법 적용으로 학습 효과 극대화

3) 개인 성장

- AI 서비스 전문성: 서비스 설계부터 서비스 개발까지 전 과정 경험했습니다.
- 아키텍처 설계 역량: Clean Architecture 적용으로 확장 가능한 시스템을 설계할 수 있었습니다.
- 제품 사고: 기술적 구현을 넘어 비즈니스 가치 창출까지 고려한 전체적 관점을 생각할 수 있었습니다.

Jarvis

온프레미스 LLM + RAG 서비스

Github Docs

개발기간: 2024.09 - 2025.03 (6개월) | 역할: AI Agent 개발자 | 기여도: 90% (서비스 MVP 개발)

Python 3.8+ LangChain Latest Streamlit UI FAISS Vector DB

프로젝트 개요

목적	**보안이 중요한 온프레미스 환경에서 도메인 특화 문서 기반 질의응답 MVP 서비스 구축**
환경	A100 80GB GPU 서버, 폐쇄망 환경, 외부 네트워크 제한
핵심 도전	제한된 GPU 메모리(실제 20GB)에서 대용량 LLM 배포 및 질문 범위 제어

프로젝트 성과

BIlossom-70B→8B 모델 전환과 LoRA 파인튜닝으로 80% 메모리 절약과 성능 유지 동시 달성	3단계 필터링 시스템으로 불필요한 LLM 호출 60% 감소
FAISS 벡터 검색으로 문서 검색 정확도 0.70, LLM 응답 품질 0.68 달성	Streamlit 기반 실시간 디버깅 시스템 구축 및 MVP 완성

정량적 성과

지표	달성값	측정 방법
메모리 효율성	약 80% 절약	70B → 8B 모델 전환 (파라미터 추정치)
검색 정확도	0.7034	Cosine Similarity 기반
응답 품질	0.6796	의미 유사도 측정

문제 해결 과정

1) 대용량 모델 배포

상황: 온프레미스 환경에서 BIlossom-70B 모델을 사용하기 위해 A100 80GB 서버에 배포 하려고 함
제약: 1. 서버 메모리 할당 제한으로 실제 사용 가능 메모리가 20GB 수준
2. 외부 네트워크 속도 제한으로 70B 모델 다운로드 어려움
해결 과정:
1. 물리적 모델 전송으로 해결 → USB를 통한 물리적 모델 파일 전송. 외부에서 다운로드 후 A100 서버에 직접 설치
2. 메모리 최적화 시도
1) 8-bit 양자화 (bitsandbytes)
2) GPU 메모리 분산 (device_map="auto"), 모델 샤딩 (accerlerate) 적용
→ 모든 최적화 시도에도 불구하고 CUDA OOM 지속적 발생
3. **전략적 모델 다운사이징 시도**
1) BIlossom 70B 모델에서 8B 모델로 전환 + vLLM 최적화 적용
결과: **메모리 사용량 80% 절약, 안정적 서비스 운영**

2) 질문 범위 제어 실패

상황: LoRA 파인튜닝 후 일반 질문 ("안녕", "자전거가 뭐야?")에도 도메인 특화 정보가 혼입되는 현상 발생
제약: 질문 범위를 제어하고 각 질문에 대해서 안전하고 정확한 응답 시스템이 필요
해결 과정:
1. 파인튜닝 데이터 확장 검토 → 시간 제약으로 불가
2. 프롬프트 수정만으로 해결 → 효과 불확실.
3. 리소스 효율성과 확실성 확보를 위한 **3단계 필터링 시스템 구축**
1) 금지어 키워드 사전에 매칭하여 질문 사전 차단
2) 질문 타입에 대한 키워드 분류 및 SentenceTransformer 기반 질문 분류 (90개 질문 사전) 활용
3) 질문 타입 별 차별화된 프롬프트 적용
결과: **불필요한 RAG 연산을 감소시켜 시스템 효율성과 리소스를 최적화하였고 도메인 외 질문에 대한 안전한 응답 체계 구축**

3) RAG 파이프라인 최적화

상황: 도메인 특화 문서(1개)의 구조와 참조 관계로 인한 청킹 어려움
제약: 문서 구조 파괴 시 문맥 손실, 너무 큰 청크 시 검색 정확도 저하 예상
해결 과정:
1. 다양한 청크 사이즈와 OCR method의 조합으로 테스트 진행
2. 결과를 육안으로 확인 후 Chunk_size=500/overlap=200, pymupdf4llm 최종 설정
3. 한국어 특화 임베딩 모델와 FAISS를 활용하여 문서 vector화
4. 일반 질문에 대한 RAG는 무시, 도메인 일반/특화 질문에 대해서 FAISS-DOCS-context로 파이프라인 구성
결과: 질문에 대한 **검색 정확도 0.7034 도출**

국립암센터

EMR 질환 예측 모델 개발 Docs

개발기간: 2023.03 - 2024.04 (1년 1개월) | 역할: 과제연구원 | 기여도: 50% (데이터 정형화, 분석, 모델링 전담)

Python 3.8+ Scikit-learn Latest XGBoost 1.6+ SHAP Explainable AI

프로젝트 개요

목적	**비정형 EMR 데이터를 활용한 고성능 질환 예측 모델 개발로 의료진 진단 지원 시스템 구축**
핵심 도전	수기 입력으로 인한 심각한 데이터 품질 문제 + 기존 연구 대비 예측 성능 한계 돌파

프로젝트 성과

정규표현식과 사분위수 이상치 탐지로 3,000명 비정형 EMR 데이터 100% 정형화 완료	의료진 협업 기반 BMI, CCI 파생 변수 생성으로 기존 논문 대비 45% 성능 향상 (ROC-AUC 0.61→0.87)
Logistic Regression + class_weight로 3:7 클래스 불균형 해결	ETL 파이프라인으로 반복 실험 80% 자동화 및 SHAP 기반 모델 해석성 확보

정량적 성과

지표	달성값	측정 방법
예측성능 향상	ROC-AUC 0..87	기존 동일 주제 논문 대비 45% 향상 (0.61 → 0.87)
데이터 정형화	약 3,000명 환자	비정형 EMR 데이터 정형화
자동화 효율성	약 80% 시간 단축	ETL 파이프라인으로 반복 실험 효율화 (20분 → 3분)

문제 해결 과정

1) 비정형 EMR 데이터 품질 문제

상황: 수기 입력으로 인한 심각한 데이터 품질 문제 (키 250cm, 체중 1130kg 등)
제약: 1. 의료 데이터 특성상 높은 정확도와 신뢰성을 요구
2. 의료 및 주제에 따른 도메인 지식이 필수
해결 과정:
1. 체계적 정규표현식 설계 → 15개 이상의 정형화 모듈로 다양한 단위 표현을 표준화, 정규표현식으로 숫자 추출
2. 통계적 이상치 탐지 → IQR 방법으로 논리적 오류 식별 및 제거
3. 의료진 협업 검증 → 2달 / 1회 대면 미팅 + 메일로 컨택하여 이중 검증 수행
결과: **15개 이상의 정형화 Script 구축** 및 3,000명 환자 데이터 정형화 완료

2) 예측 성능 제한 문제 (기존 논문 ROC-AUC 0.61)

상황: 기존 동일 주제 모델 보다 **개선된 예측 성능을 가진 모델 개발**
제약: 의학적 타당성과 통계적 유의성을 동시에 만족하는 변수와 개선된 모델 성능 필요
해결 과정:
1. 의학 문헌 조사 → BMI, CCI, 복합 위험인자 등 표준 의료 지표 및 논문 검토
2. 의료진 협업 변수 설계 → 연령에 대한 동반질환, 흡연 당뇨 등 임상적 의미있는 파생 변수 생성
3. 통계적 유의성 검증 → t-test, Mann-Whitney U test, Shapiro-Wilk test 모든 변수 $p<0.05$ 수준 변수 선택
4. Feature Selection 적용 → 통계적 분석과 ML 모델 비교
5. 정형화 과정과 모델 학습, 검증 Pipeline 구성 → 일관성 확보 및 오류를 감소하기 위한 작업으로 **처리시간 80% 단축**
6. 데이터 불균형 해결을 위한 Classweight 및 GridSearch 적용 → 7:3 데이터 불균형 해소와 모델 성능 향상
결과: LogisticRegression 모델, **ROC-AUC 0.61 → 0.87 (45% 향상)**. 결과 논문 2편 작성

3) 모델 해석성 확보

상황: 의료 분야 특성상 예측 근거 설명이 반드시 필요
제약: 1. 의료진이 이해할 수 있는 수준의 설명 필요
2. 통계분석과 ML 모델간 일관성 있는 해석 필요
해결 과정:
1. **SHAP 기반 모델 해석** 진행
1) 모든 머신러닝 모델 학습 및 검증에 SHAP 기반 모델 해석 추가
2) 각 결과 Graph를 참고하여 모델 해석 가능성 확보
2. 통계 분석과 ML 결과 비교
1) 동일한 전처리 과정으로 정형화된 데이터 활용
2) LogisticRegression 통계 분석 (statsmodel)과 비교
결과: **기존 변수 및 파생 변수에 대한 신뢰성 확보**

3D 의료영상 진단 모델 성능 평가 자동화 시스템 Docs

개발기간: 2023.03 - 2024.04 (1년 1개월) | 역할: 과제연구원 | 기여도: 90% (시스템 단독 개발)

Python 3.8+ PyTorch 1.9+ PyDicom Medical Imaging Multiprocessing Parallel

프로젝트 개요

목적	닥터앤서 국가과제 중 외부 개발사 3D 모델 실증 평가
환경	Linux 18.04, GPU rtx 3070
핵심 도전	검사자 300명의 고화질 3D 이미지 평가 자동화

프로젝트 성과

multiprocessing 4개 프로세스 병렬 처리로 검사자 300명의 3D 이미지 평가 시간 75% 단축	GPU 메모리 OOM 문제를 캐시 삭제와 배치 처리로 무중단 처리 환경 구축
Dice Coefficient + IoU 이중 평가 시스템과 Confusion Matrix 자동 생성으로 포괄적 성능 분석 완료	전처리부터 리포트 생성까지 완전 자동화로 연구원 업무 효율성 4배 향상

정량적 성과

지표	달성값	측정 방법
평가 시간 단축	75%	120분 → 30분
처리규모	300명	3D CT 이미지
업무효율성	4배 향상	연구원 수작업 부담 완전 해소

문제 해결 과정

1) GPU 메모리 부족 (OOM) 문제 해결

상황: 300장 고해상도 3D 이미지 연속 처리 시 GPU 메모리 부족으로 시스템 중단
제약: 1. 메모리 추가 불가
2. 협업사 모델 디바이스 설치 기간 명시로 짧은 실증 기간
해결 과정:
1. 메모리 사용 패턴 분석 → 3D 이미지 처리 시 메모리 누수 지점 파악
2. 병렬처리 아키텍처 설계 → multiprocessing 4개 프로세스로 작업 분산
3. 메모리 관리 최적화 → 환자별 처리 완료 후 즉시 중간 캐시 삭제
결과: **OOM 오류 해결**

2) 평가 프로세스 완전 자동화 (75% 시간 단축)

상황: 슬라이스별 평가시 환자별 20분 이상 소요 + 휴먼 에러 발생
제약: 실증 평가에 필요한 완전 자동화
해결 과정:
1. 전체 워크플로우 분석 → 기존 수작업 평가 프로세스 단계별 분석 및 병목 구간 파악
2. 자동화 파이프라인 구축
1) 이미지 전처리 자동화 → SimpleITK, Pydicom을 활용한 HU 조정 및 슬라이스별 분할
2) 성능 지표 자동 계산 → Dice Coefficient, IoU 자동 산출
3. 에러 핸들링 시스템 → Try/Except 구조로 실패 케이스 로깅
결과: **120분 → 30분 (75% 단축)** 및 휴먼 에러 완전 제거

3) 포괄적 성능 평가 시스템 구축

상황: 과제 평가 지표 이외 평가 요소가 필요
제약: 1. 의료 영상 분야에 특화된 다차원 평가 필요
2. 검사자별, 슬라이스별 세부 분석 체계 구축
해결 과정:
1. 평가 지표 설계 → Dice Coefficient (분할 정확도) + IoU (객체 탐지 성능) 이중 평가
2. 다차원 분석 체계 → 환자별 전체 성능 + 슬라이스별 세부 성능 측정
3. 시각화 및 리포팅 → Confusion Matrix + Excel 리포트 자동 생성
결과: 신뢰할 수 있는 다차원 성능 평가 체계 구축

다채널 주문서 통합 자동화 시스템 Docs

개발기간: 2022.07 - 2022.12 (6개월) | 역할: IT 서비스 개발자| 기여도: 100% (시스템 단독 개발)

Python 3.8+ Flask 2.0+ Pandas Latest AWS Lightsail

프로젝트 개요

목적	CS 담당자의 수작업 주문서 통합 업무를 완전 자동화하여 업무 효율성 극대화
핵심 도전	채널별 상이한 데이터 구조를 단일 관리 형태로 통합
배경	CS 담당자의 업무를 지원 중 업무의 비효율성 발견, 자동화 시스템 개발 아이디어 적용

프로젝트 성과

Pandas 기반 채널별 데이터 구조 정규화로 작업시간 24배 단축 (4시간→10분)	Flask + Jinja 웹 인터페이스로 일평균 100건 주문 데이터 오류율 0% 처리
Excel/JSON 설정 기반 매핑 시스템으로 코드 수정 없이 포맷 변경 무배포 대응	자동 에러 핸들링과 별도 Excel 생성으로 수작업 누락/중복 문제 완전 해소

정량적 성과

지표	달성값	측정 방법
작업시간 단축	4시간 → 10분 (24배)	실제 처리 20초 + 검수 10분
정확도	오류율 0%	데이터 무결성 보장
유지보수효율	무배포 대응	설정 파일 수정만으로 포맷 변경 적용 가능

문제 해결 과정

1) 채널별 상이한 데이터 구조 통합

상황: 자사몰과 스마트스토어의 상이한 포맷으로 통합하기 어려움
제약: 데이터의 누락, 중복 등 오류율이 없어야 함
해결 과정:

1. 구조 분석 → 채널 별 데이터 배열 패턴 분석 및 표준 형태 정의
2. 정규화 로직 → 주문번호 기준으로 Pandas Groupby, Sort 함수 조합으로 표준 순서 변환
3. 논리적 그룹핑 → 주문번호 기준 상품, 옵션 관계 매핑 시스템

결과: **일관성 확보 (동일한 표준 형태로 통합), 확장성 (정규화 규칙만 추가하면 대응), 정확성 (누락/중복 방지)**

2) 포맷 변경 대응 시스템 구축

상황: 채널사 정책 변경으로 주문서 포맷 변경 가능성
제약: 하드코딩으로 대응하여 매번 코드 수정 및 배포 필요하여 **개발자 개입 없이는 대응하기 어려움**
해결 과정:

1. 파인 지점 제거 → 하드코딩에서 설정 파일 기반 아키텍처로 전환
2. JSON 설정 분리 → 채널별 컬럼 매핑, 날짜 형식 등을 외부 설정으로 분리
3. 무배포 대응 → 설정 파일 수정만으로 포맷 변경 즉시 적용

결과: 개발자 개입 없이 운영진이 **직접 포맷 변경 대응** 가능

3) 사용자 친화적 웹 인터페이스 구현

상황: CS 담당자(비개발자)가 쉽게 사용할 수 있는 시스템 필요
제약: 업무의 효율성을 위해 빠르고 간단하게 사용 가능해야 함
해결 과정:

1. 3단계 워크플로우 → 업로드 → 처리 → 다운로드 3단계로 사용 단순화
2. Flask + Jinja → HTML 기반 직관적 웹 인터페이스 구현
3. 원클릭 처리 → 복잡한 통합 과정을 단일 버튼으로 간소화
4. 에러 핸들링 → 처리 오류 시 별도 오류 파일을 자동으로 생성

결과: 비 개발자도 **5분 내 학습 후 독립적으로 사용**이 가능.

Vision AI 기반 제조 자동화 검수 시스템 Github Docs

개발기간: 2022.10 - 2022.12 (2개월) | 역할: IT 서비스 개발자| 기여도: 90% (시스템 단독 개발)

Python 3.8+ Flask 2.0+ Pandas Latest AWS Lightsail

프로젝트 개요

목적	커스터마이징 간편식 제조 과정의 스티커 라벨링 비효율성 및 옵션 누락 문제 해결
배경	4단계 제조 카운터 (용기→채소→옵션→포장), 보안망 환경, 일일 100-200개 제품
핵심 도전	실시간 용기 카운팅 및 옵션 안내 자동화로 제조 효율성 및 정확도 향상

프로젝트 성과

YOLOv5s + DeepSORT 최적화로 객체 탐지 정확도 23% 향상 (0.65→0.8+, mAP@0.5)	실시간 추적 및 방향성 카운팅으로 1초 이내 처리 성능 달성 및 역방향 오카운팅 방지
PyQt5 기반 독립 실행 GUI와 멀티스레딩으로 최대 4개 웹캠 동시 모니터링 구현	보안망 환경 대응 .exe 패키징과 직관적 색상 UI 적용

정량적 성과

지표	달성값	측정 방법
객체 탐지 정확도	0.65 → 0.8	mAP@.5기준, 제조진 협업 데이터 검수
실시간 처리성능	<1 초 지연	DeepSORT 최적화
다중 웹캠 지원	최대 4개	멀티스레딩 기반 병렬 처리

문제 해결 과정

1) 제조 환경 변화에 robust한 객체 탐지

상황: 조명 변화, 재료 적재 상태, 제조 환경에 따른 탐지 정확도 저하 (0.65)
제약: 실제 제조 환경의 다양한 조건을 반영한 학습 데이터 부족
해결 과정:

1. 실제 환경 데이터 수집 → 50개 영상에서 약 400장 BBox 수작업 라벨링
2. 제조진 협업 검수 → 도메인 전문가와 협업으로 라벨링 품질 향상
3. 다양한 조건 반영 → 아침/점심/저녁 조명, 재료 상태, 용기 종류별 데이터 구축
4. YOLOv5s 최적화 → 경량 모델로 실시간 성능과 정확도 동시 확보

결과: 정확도 **0.8+ 달성** (23% 이상 성능 개선), 다양한 환경에서 **안정적 탐지**

2) 실시간 객체 추적 및 방향성 카운팅

상황: 제조 카운터에서 움직이는 용기의 정확한 카운팅 및 역방향 오카운팅 방지 필요
제약: 초기 1초 딜레이로 제조 속도 저해, 객체 ID 스위칭으로 중복 카운팅 발생
해결 과정:

1. YOLOv5s + DeepSORT 통합 → 경량화된 탐지 + 안정적 추적 시스템 구현
2. 중앙선 기준점 설정 → 웹캠 화면 50% 지점에 가상 중앙선으로 카운팅 트리거
3. 단방향 제한 로직 → "좌→우" 방향만 카운팅, 역방향 움직임 무시
4. 추적 ID 관리 → 객체 화면 이탈 시 자동 ID 해제로 메모리 누수 방지

결과: 개발자 개입 없이 운영진이 **직접 포맷 변경 대응** 가능

3) 보안망 환경 대응 독립 실행 시스템

상황: 제조 현장 보안망에서 웹 기반 시스템 사용 불가, 외부 의존성 제거 필요
제약: 클라우드 서비스 연결 불가, 복잡한 환경 설정 어려움
해결 과정:

1. PyQt5 네이티브 앱 → 웹 의존성 없는 데스크톱 애플리케이션 개발
2. Pyinstaller 패키징 → 모든 의존성 포함한 단일 .exe 파일 생성
3. 로컬 완전 처리 → AI 모델 추론부터 결과 저장까지 완전 오프라인
4. 독립 실행 환경 → 네트워크 연결 없이 모든 기능 작동

결과: 보안망 환경에서 완전 독립 실행, 간편한 .exe 배포

헬스케어 큐레이션 자동화 서비스 Github Docs

개발기간: 2022.07 - 2022.10 (3개월) | 역할: IT 서비스 개발자| 기여도: 90% (시스템 단독 개발)

Python 3.8+ Flask 2.0+ Pandas Latest AWS Lightsail ChannelTalk API

프로젝트 개요

목적	개인 맞춤형 건강관리 제품 추천 자동화를 통한 상담 전환율 개선
배경	실시간 개인 상담이 어렵고 맞춤형 답변이 제공되지 않은 채널톡 서포트봇 API
핵심 도전	기존 "사이트 확인→직접 문의" 방식의 낮은 전환율(2%)을 설문조사를 통해 개인맞춤 헬스케어 서비스와 제품 추천 자동화

프로젝트 성과

YOLOv5s + DeepSORT 최적화로 객체 탐지 정확도 23% 향상 (0.65→0.8+, mAP@0.5)	실시간 추적 및 방향성 카운팅으로 1초 이내 처리 성능 달성 및 역방향 오카운팅 방지
PyQt5 기반 독립 실행 GUI와 멀티스레딩으로 최대 4개 웹캠 동시 모니터링 구현	보안망 환경 대응 .exe 패키징과 직관적 색상 UI 적용

정량적 성과

지표	달성값	측정 방법
상담전환율	2% → 19% (8.5배)	채널톡 메시지 보관함 데이터 기반 측정
일일처리량	8-15건	평균 큐레이션 서비스 제공
응답시간	≤1초	설문 완료 즉시 추천 제공

문제 해결 과정

1) 채널톡 서포트봇 이벤트 분리

상황: 모든 서포트봇에 큐레이션 로직이 반응하여 불필요한 응답 발생
제약: 채널톡에서 여러 서포트봇이 동시 운영되는 환경에서 선택적 반응 필요
해결 과정:

1. Webhook 이벤트 분석 → 채널톡 이벤트 구조 파악 및 봇 ID 추출 로직
2. ID 필터링 시스템 → 큐레이션 전용 서포트봇 ID와 정확히 매칭되는 경우만 처리
3. 선택적 응답 메커니즘 → 다른 봇 이벤트는 로그만 남기고 무시 처리
4. 안전 장치 → 봇 ID 누락 시 기본적으로 무시하는 보수적 접근

결과: **큐레이션 요청에만 반응**하여 시스템 정확성 및 리소스 효율성 확보

2) 개인 맞춤형 추천 로직 설계

상황: 복잡한 AI 없이 실용적인 제품 추천 시스템 필요
제약: 실시간 처리 요구, 개발 리소스 제한, 즉시 적용 가능한 솔루션 필요
해결 과정:

1. 설문 기반 분류 → 건강상태, 식습관, 운동량, 음수량, 활동량 5개 영역 평가
2. 목표별 세분화 → 다이어트/유지/증량 3개 카테고리 자동 분류
3. 규칙 기반 매칭 → 목표별 사전 정의된 제품 정보와 개인 건강 상태 조합
4. 실시간 메시지 생성 → 개인화된 추천 메시지 자동 생성 및 전송

결과: 복잡한 AI 없이도 효과적인 맞춤 추천으로 8.5배 전환율 향상

3) 기존 상담 프로세스 개선 및 AWS Lightsail 기반 비용 효율적 운영

상황: "사이트 확인 → 직접 문의" 방식의 2% 낮은 전환율 및 24시간 무중단 서비스 필요, 스타트업 환경의 비용 제약
제약: 고객의 진입 장벽 높음, CS 담당자 반복 업무 부담, 또한 최소 비용으로 안정적 서비스 운영, 확장성 고려
해결 과정:

1. 자연스러운 참여 유도 → 약 2분 간단 설문으로 고객 참여 증대
2. 사전 추천 완료 → CS 상담 전 맞춤 제품 추천으로 상담 효율성 향상
3. Lightsail 최적화 → 최소 스펙으로 EC2 대비 70% 비용 절약
4. 외부 의존성 최소화 → 채널톡 DB 활용으로 별도 데이터베이스 구축 비용 제거

결과: "설문→추천→상담" 의 자연스러운 플로우로 전환율 혁신적 개선 / 최소 비용으로 24시간 안정적 서비스 운영

소아흉부 폐질환 진단 및 분류 경진대회 Github Docs

개발기간: 2021.11 - 2021.12 (1개월) | 역할: 팀리더 | 기여도: 40% (설계 및 전처리 중심)

Python 3.8+ TensorFlow 2.7+ Keras Latest OpenCV 4.5+ PyQt5 GUI

프로젝트 개요

목적	소아 흉부 X-ray 기반 7가지 폐질환 자동 진단 시스템 개발
환경	네이버 클라우드, Ubuntu 16.04, T4 (1), Memory 20GB, HDD 100GB
핵심 도전	4,000장 데이터로 7-Class 분류, 심각한 클래스 불균형(25%~3.1%) 해결

프로젝트 성과

오픈 데이터인 RSNA 데이터를 활용하여 분류 성능 검증	PyQt를 활용해 결과 보고서 까지 출력하는 GUI 개발
경진대회 300팀 중 2위 수상	미공개 Test 데이터 F1-Score 0.63

정량적 성과

지표	달성값	측정 방법
경진대회 순위	2위 수상	300팀 참가 → 5팀 본선 → 3팀 수상
모델 성능	Test F1-Score 0.63	미공개 Test 데이터 활용
상금	100만원	경진대회 2위 수상

문제 해결 과정

1) 심각한 클래스 불균형 데이터 처리

상황: 정상 25% vs 과다팽창 3.1% 등 의 극심한 불균형으로 소수 클래스 학습 부족
제약: 희귀 질환일수록 정확한 진단이 중요하지만 학습 데이터 부족
해결 과정:

1. Class Weight 적용 → 소수 클래스에 높은 가중치 부여 (과다팽창 클래스 10배 가중치)
2. 4배 데이터 증강 → 회전, 이동, 밝기 조절로 데이터셋 16,000장으로 확장
3. 소수 클래스 집중 증강 → 부족한 클래스를 목표 샘플 수까지 선택적 증강
4. F1-Score 최적화 → 불균형 데이터에 적합한 평가 지표로 모델 최적화

결과: **불균형 클래스 F1-Score 0.1 → 0.4 개선**, 전체 성능 균형 달성

2) 의료 영상 전처리 파이프라인 최적화

상황: DICOM 형식 의료 영상의 효과적 전처리 방법 부재, 팀원별 다양한 방식 제안으로 혼선
제약: 팀 협업에서 최적 방법 객관적 선정 필요
해결 과정:

1. 체계적 비교 → 각자 샘플 데이터로 전처리 구현 → 시각적 비교 → 최적 방법 선정
2. CLAHE 적용 → 이미지 대비 개선
3. U-Net Segmentation → 폐 영역만 정확히 분할하여 노이즈 제거
4. 품질 필터링 → 폐 개수 2개 이하 이미지만 사용하여 데이터 품질 확보

일부 질환(흉막삼출, 공기누출)에서 Segmentation 없는 원본이 더 좋은 성능을 보임.
특정 폐질환은 흉강 변화가 진단 핵심으로 폐 분할이 오히려 정보 손실" 이라는 것을 알게됨.

결과: 복잡한 AI 없이도 효과적인 맞춤 추천으로 8.5배 전환율 향상

3) CheXNet 모델 커스터마이징

상황: 일반 CNN 모델들(VGG16, InceptionV3, DenseNet169, ResNet50, MobileNet)의 낮은 성능
제약: 의료 영상 특성에 최적화된 모델 아키텍처 필요
해결 과정:

1. 기존 모델 한계 확인 → 모든 일반 CNN 모델이 F1-Score 0.1-0.5 수준 성능
2. CheXNet 선택 → DenseNet121 기반 흉부 X-ray 특화 모델 채택
3. 구조 커스터마이징
   - include_top=False로 DenseNet121을 feature extractor 사용
   - GlobalAveragePooling2D로 공간 정보 평균화
   - Dense(7, softmax)로 7-class 분류 출력층

결과: "초기 모델 대비 약 7배 이상 성능 향상 (Train: F1-Score 0.117 → 0.94) *과적합으로 인한 결과 예상 (Test와 차이)*

골연령 예측 손목, 손가락 관절 탐색 및 모델 개발 Github Docs

개발기간: 2021.10 - 2021.11 (1개월) | 역할: 팀리더 | 기여도: 40% (설계 및 전처리 중심)

Python 3.8+ PyTorch 1.11+ YOLOv5 Latest TensorFlow 2.7+ PyQt5 GUI

프로젝트 개요

목적	소아기 왼손 X-ray 영상 기반 정확한 골연령 예측 및 성장 예측 시스템 개발
배경	연세대학교 소아과 x-ray 1,730장을 활용하여 의료 인공지능 서비스 개발
핵심 도전	전처리→관절탐지→골연령예측→성장예측의 end-to-end AI 서비스 구축

프로젝트 성과

TW3 기반 ROI 추출을 위한 YOLOv5 성능 mAP@0.5 0.991 달성	골연령 예측 평가 모델 Tj-Net MAE 4.7 달성 (의료진 평가와 4.7개월 차이)
질병관리청에서 제공된 "소아 청소년 성장도표" 과 LMS공식으로 18세 예측 신장 계산 pipeline 개발	PyQt를 활용해 골연령 검사 및 자동 보고서 기능을 포함한 GUI 개발

정량적 성과

지표	달성값	측정 방법
경진대회 순위	평균 4.7개월	전문의 판단 대비 MAE (Mean Absolute Error)
객체 탐지 정확도	MAP@0.5 0.991	YOLOv5s 기반 관절 탐지 성능
전문의 일치도	±1.96SD 범위 내	Bland-Altman 분석, 95% 신뢰구간 내 대부분 데이터 포함

문제 해결 과정

1) 복잡한 X-ray 영상 전처리 파이프라인 최적화

상황: 다양한 촬영 조건, 배경 노이즈, 손 위치 변화로 인한 영상 품질 편차
제약: 의료 영상 특성상 높은 정확도 요구, 뼈 구조 명확한 구분 필요
해결 과정:

1. 체계적 파이프라인 설계 → "마스크 생성 → 배경 제거 → 손목 기준 회전 → 이진화 → 뼈 강조"
2. OpenCV 기반 정밀 처리
   - 가우시안 블러 + Bitwise로 손 영역 분리
   - 모폴로지 연산으로 노이즈 제거
   - 손목 중심축 기준 회전 정렬 및 CLAHE로 뼈 구조 강조

결과: 각 단계별 기능 자동화 script 를 통해 처리시간 40% 단축

2) TW3 기법 기반 정확한 관절 탐지 및 TJ NET 모델 활용

상황: 의학적으로 표준화된 TW3 방법론으로 관절 추출 및 성별 간 골격 발달 차이, 개인차, 전문의 수준의 예측 정확도 달성
제약: 손목과 손가락 20개 핵심 관절 중 7개의 ROI 추출 및 동일 연령에서도 개인별 골격 발달의 큰 편차, 일반화 성능 확보
해결 과정:

1. TW3 표준 적용 → 의학 문헌 기반 정확한 관절 분류 체계 구축
2. YOLOv5 → 7개 관절별 개별 클래스 설정 및 학습 (621장) 및 추론
3. TJnet 아키텍처 설계 → VGG16 백본 + 성별 정보 통합 브랜치
4. Input Layer와 성별 정보 통합 → Input layer에 7개 ROI로 변환 및 Gender 입력으로 남녀 골격 발달 차이 학습

결과: YOLO mAP@0.5 0.991 성능과 TJ-Net MAE 4.7개월 달성

3) LMS 방법 기반 18세 예상 신장 예측

상황: 현재 골연령을 바탕으로 최종 성인 신장 예측, 복잡한 LMS 공식 정확한 구현
제약: 질병관리청 소아청소년 성장도표의 L, M, S 매개변수 활용, 의료진 이해 가능한 시각화
해결 과정:

1. LMS 방법론 구현
   - L(Box-Cox 변환), M(중위수), S(변동계수) 매계변수 활용하여 현재 신장의 분위수를 계산하여 18세 신장 예측
   - 성장 그래프 생성 → 질병관리청 스타일 성장 곡선을 python으로 구현
2. 종합분석 시스템
   - 골연령 차이에 따른 성장 잠재력 분석
   - 시각적 표현 → 현재 (흑점), 예측 신장 (적점), 성장 예측선 직관적 표시

결과: 의학적으로 검증된 18세 예측, 의료진 친화성 성장 그래프 제공