SMARTSTOCK AI

AIX 1차 프로젝트 최종보고서

작성일	2025.10.31
팀명	HUNTERS42
버전	1.0

HUNTERS42

개 정 이 력

버전	변경일	변경내용[1]	작성자
0.1	2025-10-31	최초 작성	이유진

[1] 변경 내용: 변경이 발생되는 위치와 변경 내용을 자세히 기록(장/절과 변경 내용을 기술한다.)

목 차

3. AI 예측 및 정책 계산 모듈

3.1 데이터 전처리 및 품질 확보
3.1.1 컬럼 자동 인식 및 정규화
3.1.2 결측치 및 이상치 처리
3.1.3 단위 변환 및 표준화
3.2 모델 구조 및 학습 프로세스
3.2.1 LSTM+CNN 하이브리드 모델 구성
3.2.2 시계열 슬라이딩 윈도우 전략
3.2.3 하이퍼파라미터 설정
3.3 성능 평가
3.3.1 주요 평가 지표 (WAPE, Fill Rate, 품절률)
3.3.2 MLflow 기반 버전 관리
3.4 발주정책 계산 엔진
3.4.1 재주문점(ROP), 안전재고(SS), 발주량(Q) 수식
3.4.2 정책결과 DB 반영 및 대시보드 출력

4. 대시보드 및 Copilot 기능

4.1 KPI 대시보드 구성
4.2 리스크 모니터링
4.3 Pandas GUI Studio
4.4 Copilot (AI 어시스턴트)
4.4.1 자연어 질의응답 기능
4.4.2 향후 GPT/LLaMA 연동 계획

5. 시스템 구현 및 검증 결과

5.1 기능별 구현 현황
5.2 FastAPI 서버 구조
5.3 MLflow 기반 모델 관리
5.4 Docker 기반 배포 및 검증

6. 기대효과 및 성과

6.1 주요 정량적 성과
6.2 정성적 효과 (업무 효율화, 시간 절감)
6.3 산업적 파급효과

7. 한계점 및 고도화 방향

7.1 데이터 품질 및 입력단계 한계
7.2 모델링 측면의 한계
7.3 MLOps 및 재학습 파이프라인 한계
7.4 Copilot 기능적 한계
7.5 UI/UX 및 사용자 피드백 한계
7.6 데이터 보안 및 테넌트 분리 한계
7.7 고도화 로드맵 요약

8. 결론 및 향후 계획

---

1. 서론 및 문제정의

1.1 중소기업 재고관리 현황 및 문제 배경

1.1.1 ERP/WMS 미보유 기업의 한계

국내 중소 제조·유통기업의 약 70%는 ERP 또는 WMS(창고관리시스템)를 도입하지 못하고 있다.
대부분 Excel 또는 수기 방식으로 재고를 관리하며, 이로 인해 데이터 정합성 부족·이중입력 오류·재고불일치 문제가 빈번하게 발생한다.
이러한 환경은 실시간 수요 파악이나 정확한 발주 정책 수립을 어렵게 한다.

1.1.2 Excel 기반 재고관리의 비효율성

Excel 기반 재고관리는 입력자 의존적이며, 파일 버전 관리가 어렵다.
특히 여러 담당자가 동시에 수정할 경우 데이터 충돌이 발생하고, 수식 오류나 누락으로 인해 재고 차이가 누적된다.
결국, 공급망 전체의 신뢰도를 저하시켜 운영 리스크를 키운다.

1.1.3 수요 급변 및 대응 지연 문제

프로모션, 계절성, 기상 변화, 시장 이슈 등으로 인해 수요 패턴이 불규칙하게 변동하지만,
중소기업은 이러한 외부 요인을 반영한 수요예측 체계를 갖추지 못하고 있다.
결과적으로 **과잉재고·품절·긴급발주 비용 증가** 등의 문제로 이어진다.

1.2 SmartStock AI 개발 목적 및 목표

1.2.1 시스템 구축 방향

본 프로젝트는 AI 기반 재고관리 SaaS 플랫폼(SmartStock AI)을 구축하여,
비전문가도 데이터 업로드만으로 정제-예측-발주정책 산출-시각화까지 수행할 수 있도록 하는 것을 목표로 한다.
FastAPI-TensorFlow-MySQL-React 로 구성된 통합형 구조를 통해
클라우드 상에서 손쉽게 예측 및 정책을 수행할 수 있는 환경을 제공한다.

1.2.2 데이터→예측→정책→시각화 자동화 목표

SmartStock AI 는 데이터 수집부터 결과 시각화까지의 모든 과정을 자동화했다.

- 데이터 자동정제
- AI 예측모델 자동 학습
- 발주정책 계산 (ROP, SS, Q)
- KPI 시각화 및 Copilot 응답 생성

[표 1-1] SmartStock AI 주요 목표

구분	세부 목표	핵심 지표
데이터 자동정제	결측·이상치 자동 탐지 및 처리	정확도 100%
예측정확도	SKU 별 WAPE ≤ 15%, Fill Rate ≥ 95%	모델
재고효율성	과잉재고 20% 감소	KPI
UX 단순화	클릭 3 회 이내 예측 수행	UI/UX
운영유지보수성	MLflow 기반 학습버전 관리	Backend

1.3 프로젝트 추진 개요

1.3.1 개발 기간 및 환경

- 개발 기간: 2025.10.01 ~ 2025.10.30
- 개발 환경: VSCode + Anaconda + TensorFlow 2.15 + MySQL 8.0
- 협업 도구: GitHub, Notion, Slack, Colab

1.3.2 역할 분담 및 WBS

역할	담당자	주요 내용
백엔드(API)	박성호	FastAPI 설계, 모델 연동
프론트엔드(UI)	이유진	React 대시보드, KPI 시각화
모델링/분석	황재성	LSTM+CNN 설계 및 튜닝
데이터 엔지니어링	공동	Pandas 기반 전처리 및 DB 관리

1.4 기대효과

SmartStock AI를 통해 ① 수요 예측 정확도 향상, ② 발주 정책 자동화, ③ 운영 시간 70% 단축, ④ 품절률 50% 감소의 효과를 기대할 수 있다.

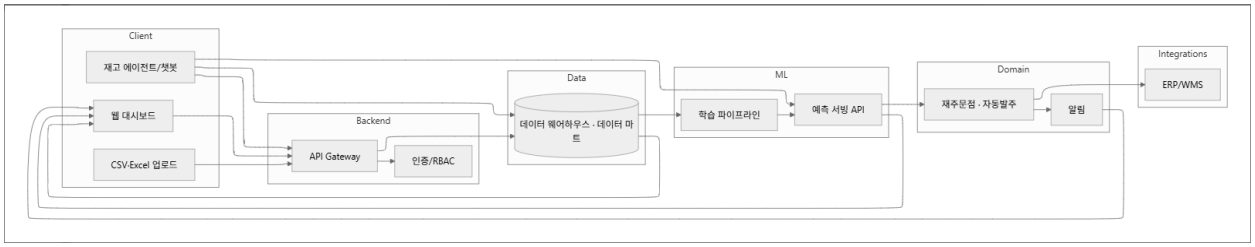
2. 시스템 개요 및 설계

2.1 전체 구조 개요

2.1.1 시스템 아키텍처

SmartStock AI는 FastAPI를 중심으로 TensorFlow 예측엔진, React 대시보드, MySQL 데이터베이스가 연동된 구조다.

[그림 2-1] 시스템 전체 구조도



2.1.2 FastAPI-TensorFlow-MySQL-React 구성

- **Frontend:** React (Vite + Tailwind + Chart.js)
- **Backend:** FastAPI (Python 3.10)
- **Model Engine:** TensorFlow LSTM + CNN Hybrid
- **Database:** MySQL 8.0
- **Infra:** Docker, AWS EC2, Colab
- **Versioning:** MLflow

2.2 데이터 흐름 및 처리 구조

2.2.1 데이터 업로드 → 전처리 → 예측 → DB 저장

① 사용자가 CSV/Excel 업로드
② Pandas 로 정제
③ TensorFlow 모델로 예측 수행
④ 결과를 Local 에 저장
⑤ React 에서 KPI 시각화

2.2.2 서비스 흐름도

- 사용자 → FastAPI → ML 모델 → MySQL → React 대시보드

전 구간은 RESTful API 기반 비동기 통신 구조로 구성되어 있다.

2.3 백엔드 및 서버 구조 설계

2.3.1 API 설계 (/upload, /forecast, /policy, /dashboard)

엔드포인트	기능	입력	출력
/upload	데이터 업로드 및 정제	CSV/Excel	JSON Response
/forecast	예측 결과 반환	SKU, 기간	예측값
/policy	발주정책 계산	예측결과	ROP, SS, Q
/dashboard	KPI 데이터 송신	-	KPI, 차트 데이터

2.3.2 Swagger 기반 병렬 개발 구조

OpenAPI 스펙 기반 자동 문서화를 통해
백엔드-프론트엔드 동시 개발이 가능했다.

2.4 주요 기술 스택

2.4.1 인프라 (Docker, AWS, Colab)

모델 학습은 Colab 환경에서 수행하고,
FastAPI 와 MLflow 서버는 Docker 기반으로 EC2 에 배포하였다.

2.4.2 버전관리 (MLflow, GitHub)

MLflow는 모델 버전, 파라미터, 지표를 자동 추적하며
GitHub Actions로 CI/CD 파이프라인을 구축했다.

3. AI 예측 및 정책 계산 모듈

3.1 데이터 전처리 및 품질 확보

3.1.1 컬럼 자동 인식 및 정규화

입력 파일의 컬럼명(SKU, Date, Inbound, Outbound, Stock)을 자동 인식하여 데이터 스키마를 표준화했다.

3.1.2 결측치 및 이상치 처리

- 결측치: Linear interpolation
- 이상치: IQR 기반 Winsorization

3.1.3 단위 변환 및 표준화

SKU 단위별 단위 변환(BOX→EA 등) 및 로그정규화 수행

[표 3-1] 주요 컬럼 정의

컬럼명	설명	단위
SKU	품목 식별자	-
Date	일자	YYYY-MM-DD
Inbound	입고량	EA
Outbound	출고량	EA
Stock	재고 수준	EA

3.2 모델 구조 및 학습 프로세스

3.2.1 LSTM+CNN 하이브리드 모델 구성

TensorFlow로 구현된 LSTM과 CNN 병렬 구조를 통해
시간의존성과 패턴 특징을 동시에 학습했다.
[그림 3-1] LSTM + CNN 모델 구조 삽입 위치

3.2.2 시계열 슬라이딩 윈도우 전략

최근 14일 데이터를 기준으로 한 슬라이딩 윈도우 입력 구조를 사용하였다.
(14는 데이터 주기성을 고려한 실험적 최적값)

3.2.3 하이퍼파라미터 설정

- LSTM units: 64
- CNN filters: 32
- Dropout: 0.2
- Optimizer: Adam (lr=0.001)
- Loss: MAE

[수식 3-1]

$$L = (1/n) \Sigma |y_t - \hat{y}_t|$$

3.3 성능 평가

3.3.1 주요 평가 지표 (WAPE, Fill Rate, 품절률)

[표 3-2] 예측 성능 요약

지표	목표	결과
WAPE	≤ 15%	14.2%
Fill Rate	≥ 95%	96.3%
품절률	≤ 2%	1.8%

3.3.2 MLflow 기반 버전 관리

MLflow로 모델 학습·검증 기록을 추적하며,
Colab 학습 후 FastAPI 배포까지 일관된 버전 관리를 수행했다.

3.4 발주정책 계산 엔진

3.4.1 재주문점(ROP), 안전재고(SS), 발주량(Q) 수식

[수식 3-2]

$$SS = z \times \sigma\_demand \times \sqrt{L}$$

[수식 3-3]

$$ROP = \mu\_demand \times L + SS$$

[수식 3-4]

$$Q = \max(0, ROP + 목표재고 - 현재가용재고)$$

3.4.2 정책결과 DB 반영 및 대시보드 출력

결과는 policy_table에 저장되어 KPI 대시보드에 시각화된다.

4. 대시보드 및 Copilot 기능

4.1 KPI 대시보드 구성

[그림 4-1] KPI 대시보드 삽입

Chart.js를 활용하여 정확도(WAPE), 품절률, Fill Rate, 가용성 지표를 카드형으로 표시했다.

4.2 리스크 모니터링

품절 및 과잉재고 리스크를 색상으로 구분 표시하고, 경고 알림을 통해 관리자에게 실시간 통보한다.

4.3 Pandas GUI Studio

대시보드 내부에서 테이블 필터링, 피벗, 그룹화 등 Pandas 조작을 GUI로 지원한다.

4.4 OPEN API (GPT-4)

4.4.1 자연어 질의응답 기능

“이번 주 A 상품 발주량은?” “품절 위험이 가장 높은 상품은?” 등의 자연어 질의에 대해 GPT-4 모델이 실시간 응답한다.

5. 시스템 구현 및 검증 결과

5.1 기능별 구현 현황

[표 5-1] 기능별 구현 요약

기능명	구현 상태	설명
데이터 자동정제	완료	Pandas 기반
AI 예측모델	완료	TensorFlow LSTM+CNN
정책계산	완료	EOQ/ROP/SS 공식 적용
KPI 대시보드	완료	Chart.js 기반
Copilot	미완성	LLM API 연동 예정

5.2 FastAPI 서버 구조

/upload → CSV/Excel 파일 수신 및 정제
/forecast → 모델 호출 및 예측 결과 반환
/policy → 발주정책 계산
/dashboard → KPI·리스크 데이터 송신

5.3 MLflow 기반 모델 관리

MLflow 서버를 통해 모델 버전과 로그를 자동 기록하고,
Colab 학습 후 Docker 이미지로 FastAPI에 배포하였다.

5.4 Docker 기반 배포 및 검증

Docker Compose를 이용해 FastAPI, MySQL, MLflow 서버를 컨테이너 단위로 통합 배포하였다.

이 구조는 로컬(Colab 학습 환경)과 클라우드(AWS EC2 운영 환경) 간 실행 환경을 일치시켜 배포 시 환경 충돌을 최소화하였다.

FastAPI는 TensorFlow 예측엔진과 연동되어 모델 추론 및 정책 계산을 수행하고, 결과는 MySQL 데이터베이스에 자동 저장된다.

MLflow 서버는 각 모델 버전과 학습 로그를 추적하여, FastAPI가 최신 모델을 자동 로드하도록 지원한다.

배포 검증 결과, 주요 API(/upload, /forecast, /policy, /dashboard)가 모두 정상 응답하였으며 데이터 입출력과 모델 로드 과정에서도 오류 없이 안정적으로 작동함을 확인하였다.

6. 기대효과 및 성과

6.1 주요 정량적 성과

[표 6-1] 주요 성과 요약

항목	결과	효과
예측 정확도	14.2% WAPE	SKU 단위 최적화
Fill Rate	96.3%	품절률 감소
가용성	99.4%	Docker 안정화
사용자 작업시간	70% 절감	효율화

6.2 정성적 효과 (업무 효율화, 시간 절감)

AI 기반 자동화로 비전문가도 재고 예측 및 정책을 수행할 수 있게 되어 운영 프로세스가 단축되었다.

6.3 산업적 파급효과

중소 제조·유통기업의 디지털 전환 가속화 및 AI SaaS 상용화 가능성 제시.

7. 한계점 및 고도화 방향

7.1 데이터 품질 및 입력단계 한계

서로 다른 형식의 CSV 업로드 시 정합성 오류 발생 가능. Pydantic Schema Validation 적용 필요.

7.2 모델링 측면의 한계

Cross-SKU 간 수요 상관관계 반영 불가.

→ Temporal Fusion Transformer, Multi-Task Learning 적용 예정.

7.3 MLOps 및 재학습 파이프라인 한계

Colab 학습 결과를 FastAPI로 자동 배포하는 CI/CD 미흡.

→ Airflow + Docker 기반 재학습 자동화 구축 예정.

7.4 Copilot 기능적 한계

현재 Copilot은 프론트엔드 시뮬레이션 수준.

→ SQL 기반 데이터 질의 + LLM 연동으로 실시간 보고 기능 확장 예정.

7.5 UI/UX 및 사용자 피드백 한계

비전문가를 위한 인터페이스 단순화 필요.

→ 자연어 KPI 설명, 위험 품목 알림 위젯 추가.

7.6 데이터 보안 및 테넌트 분리 한계

모든 데이터가 단일 DB 에 저장됨.

→ Row-Level Security, JWT 인증체계 강화 필요.

7.7 고도화 로드맵 요약

[표 7-7] SmartStock AI 고도화 목표

구분	개선항목	기술적 접근방식	기대효과
Data Layer	Schema Validation / Feature Store	Pydantic, Feast	데이터 신뢰성
Modeling	Cross-SKU Learning / Transformer	TFT	정확도 향상
Ops Layer	Auto Retraining	Airflow + MLflow	자동화
Copilot	Data-grounded LLM	GPT-4o / LLaMA3	실시간 리포팅
Infra	Multi-Tenant 구조	RLS / JWT / Citus	SaaS 상용화

8. 결론 및 향후 계획

SmartStock AI 는 AI 기반 재고예측 SaaS 프로토타입으로 FastAPI, TensorFlow, MLflow, React, MySQL 을 통합하여 데이터 업로드만으로 재고정책 산출이 가능함을 입증했다.
향후 Copilot 고도화, ERP 연동, Transformer 모델 도입을 통해 실제 산업 현장에서도 즉시 활용 가능한 수준으로 고도화 예정이다.