

하자 처리 시간 예측 모델

데이터 분석부터 배포까지 단독 수행

하자 관리 서비스 '채들'의 입주민은 하자를 접수해도 언제 고쳐질지 알 수 없었습니다. 관리자가 입력하는 처리 예정일은 고지용이라 실제 완료일과 차이가 컸습니다. 하자 처리 이력 440만 건으로 **보수 시작부터 완료까지 걸리는 시간**을 예측하는 모델을, 데이터 분석·모델링·배포 설계까지 혼자 만들었습니다.

작성 하성진	역할 단독 수행 (분석 · 모델링 · 배포)	스택 Python · LightGBM · CatBoost · KServe	데이터 하자 처리 이력 440만 건
-----------	-----------------------------	---	------------------------

95.2%	81.3%	3.8 → 3.5일
사전점검 하자 실제 완료일 ±3일 이내 예측	사전점검 하자 실제 완료일 ±1일 이내 예측	5-fold 교차검증 평균 오차(MAE) 라벨 → 타겟 인코딩

01 무엇을 예측할지부터 정했다

하자는 네 단계로 처리됩니다. 처음에는 접수 시각부터 재려 했지만, **접수대기는 아직 처리가 시작되지 않은 상태**입니다. 작업 지시가 들어가야 보수가 시작되므로, 예측 대상을 작업 시작부터 보수 완료까지로 정했습니다.

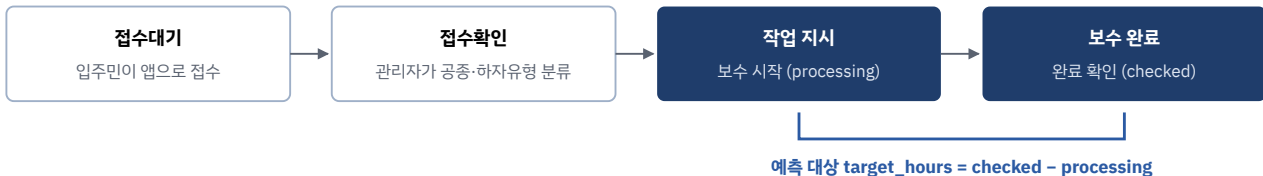


그림 1. 하자 처리 단계와 예측 구간. 30일을 넘거나 음수인 값은 데이터 오류로 보고 제외했다.

딥러닝은 쓰지 않았습니다. 입력의 대부분이 공종·하자유형 같은 범주형 표 데이터였고, 현장 담당자가 예측 근거를 이해할 수 있어야 했기 때문입니다.

02 데이터에서 찾은 문제와 판단

2.1 결측치가 컬럼마다 극단적으로 다름

현상	업체 연결 일시(mapped_at)는 98.9%, 방문 일시(visited_at)는 91.1%가 비어 있었습니다. 공중·하자유형은 일부만 비어 있었습니다.
판단	90% 이상 빈 컬럼은 삭제하고, 5% 이하인 컬럼은 해당 행만 지웠습니다. 재처리·공중이관 컬럼은 결측이 99% 가까이 됐지만, 도메인 특성상 보수완료와 함께 기록될 수 없는 값이라 오류가 아니었습니다. 그래서 1차 모델은 보수완료 하자에 집중했습니다.

2.2 보수완료 하자의 44%에 협력업체 정보가 없음

현상	보수완료 200만 건 중 88만 건에 협력업체가 비어 있었습니다.
판단	누락이 아니라 처리 방식 자체가 두 가지라고 가설을 세웠습니다. 사전점검은 매니저가 직접 처리하고, 입주·품질점검은 외부 업체를 거친다는 것입니다. 이를 협력업체 관여 여부(has_partner) 변수로 만들어 모델에 넣었습니다.

2.3 카테고리가 많고, 대부분이 희소함

현상	세부공종은 628개였고 그중 414개가 50건 미만, 37%가 10건 미만이었습니다(아래 표).
판단	희소 카테고리를 빼면 모델은 안정되지만 전체 하자의 약 60%만 예측 할 수 있었습니다. "이 하자는 예측할 수 없습니다"는 서비스로 쓸 수 없는 답이라 빠지지 않았습니다. 대신 카테고리 이름 대신 자재 필요 여부, 전문 인력 필요 여부, 구조적 작업 여부 같은 속성으로 바꿔, 처음 보는 공종도 예측할 수 있는 방향을 택했습니다.

범주형 변수	카테고리 수	50건 미만	50건 이상
공종	114	54 (47%)	60
세부공종	628	414 (66%)	212
하자유형	114	66 (58%)	48

03 숫자가 좋아도 배포하지 않은 이유

모델링은 네 번 고쳤습니다. 가장 중요한 결정은 **지표가 좋아진 모델을 서비스에 쓰지 않기로 한 것**이었습니다.

#	시도	결과	판단
1	LightGBM + 라벨 인코딩 보수완료 82만 건	단일 분할 1.8일 5-fold CV 3.8일	한 번 나눈 결과가 우연히 좋았던 것입니다. 이후로는 교차검증 값만 성능으로 인정했습니다.
2	K-fold 타겟 인코딩	CV MAE 3.5일 (약 8% 개선)	오차 중앙값은 2.5일이었지만 최대 오차가 25.3일이었습니다. 평균만 보면 이 1%의 실패가 가려집니다.
3	피쳐 중요도 검증	서비스 부적합 판정	하자 ID, 호수, 처리 날짜가 상위였고 하자유형은 10위였습니다. 도메인으로 설명이 안 되는 대리 변수를 외운 것으로 보고, 지표는 좋지만 배포하지 않기로 했습니다.
4	데이터 재정비 + 점검 유형별 모델 분리	사전점검 ±3일 95.2% 입주점검 ±3일 52.7%	중복·이상치를 걸러 40만 건으로 학습했습니다. 처리 방식이 다른 사전점검(약 36만 건)과 입주점검을 따로 학습했습니다.

점검 유형별 정확도 (실제 완료일과의 차이)

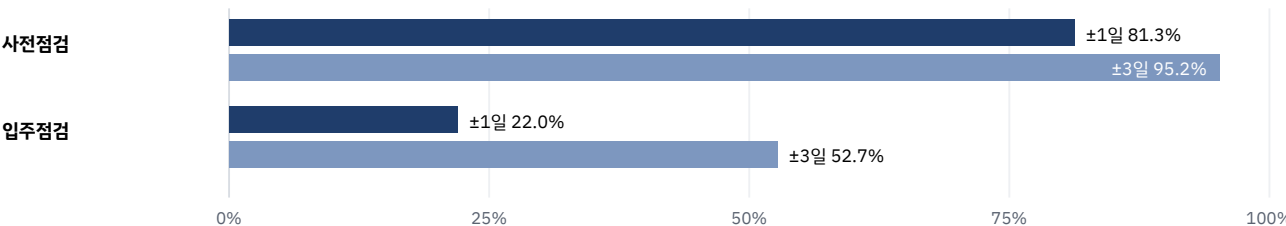


그림 2. 사전점검은 서비스에 쓸 수 있는 수준이지만, 입주점검은 데이터에 비해 경우의 수가 많아 정확도가 낮다. 범위를 ±7일로 넓히면 수치는 오르지만 입주민에게 의미가 없어 기준으로 삼지 않았다.

04 운영에서 무너지지 않게 배포를 설계했다

타겟 인코딩은 "공종별 평균 처리 시간" 같은 값으로 범주를 바꿉니다. 하자 데이터는 계속 쌓이므로 **모델은 그대로여도 이 평균값은 계속 변합니다**. 그래서 모델과 인코딩 데이터를 분리해 배포 경로를 설계했습니다.



그림 3. 학습부터 새도우 배포까지의 흐름. 기존에 운영하던 EKS·KServe 서버 인프라 위에 배포 경로를 엮었다.

설계	이유
새도우 배포	평균값은 이상치에 끌려갑니다. 갱신한 인코딩이 이상한 예측을 내더라도 사용자에게 보이기 전에 실제 트래픽으로 확인하고 모니터링할 수 있게 했습니다.
추론용 저장 요소 4가지	학습과 추론의 변환이 어긋나면 모델이 맞아도 결과가 틀립니다. 타겟 인코딩 맵, 라벨 인코더, 전체 평균(처음 보는 카테고리용), 피쳐 순서를 함께 저장합니다.
테스트 환경	Minikube에 KServe InferenceService를 올리고, 실제 "처리중" 하자를 넣어 예측 일수를 확인하는 API와 화면을 만들었습니다.

05 남은 한계와 다음 과제

한계	계획
입주점검 정확도	± 3 일 52.7%로 아직 서비스 수준이 아닙니다. 입주점검만의 처리 패턴을 반영한 변수를 추가하고 데이터를 보강할 예정입니다.
드문 극단 오차	학습 사례가 적은 하자에서 25일이 넘는 오차가 납니다. 유사 사례가 10건 미만이면 단일 값 대신 신뢰도와 예상 범위를 함께 보여주는 방식을 설계했습니다.
시간에 따른 변화	무작위 분할 대신 시간 순서 분할로 다시 검증하고, 주간 보정과 월간 재학습으로 데이터 변화에 대응할 계획입니다.

이 프로젝트에서 얻은 것 — 머신러닝 모델은 과거 사례를 빠르게 찾아 주는 도구입니다. 그래서 무엇을 근거로 예측하는지가 지표만큼 중요했습니다. 지표가 좋아진 3번째 모델을 스스로 기각하고 데이터부터 다시 본 경험이, 이후 모든 모델 검증의 기준이 됐습니다.

06 기술

구분	내용
분석 · 모델링	Python · Pandas · scikit-learn · LightGBM · CatBoost · K-fold 타겟 인코딩
배포 · 서빙	Docker · AWS ECR · S3 · KServe InferenceService · Kubernetes(EKS · Minikube) · 새도우 배포
파이프라인	Kubeflow Pipelines

수치는 2025년 9월 측정값입니다. 정확도는 실제 완료일과 예측 완료일의 차이가 기준 일수 이내인 비율이며, MAE는 5-fold 교차검증 평균입니다.