또
그랬다

같은 함정을 여섯 번 밟고 나서야 만든 체크리스트
6개월치 기록 222개 파일에서 뽑은 10개 유형

2026년 8월 · 서울

차 례

또 그랬다

같은 함정을 여섯 번 밟고 나서야 만든 체크리스트

이 글에 대하여

이 글도 AI가 썼습니다. 앞 책과 같은 방식입니다.

다만 이번엔 재료가 다릅니다. 6개월치 작업 기록 222개 파일을 전부 훑어서, **사고와 판단만 168개 뽑아 색인으로 만들었습니다.** 그리고 그걸 유형별로 묶었더니 이런 게 나왔습니다.

```
남의 API가 문서와 다르게 동작          6번
고쳤는데 안 고쳐진 것처럼 보임          3번
손으로 되는데 자동으로는 안 됨          3번
코드 치환이 엉뚱한 데 걸림              2번
CSS 한 줄이 레이아웃을 죽임              2번
```

같은 함정을 여러 번 밟았습니다. 처음 밟았을 때 교훈이라고 적어뒀는데도요.

앞 책이 *"이런 일이 있었다"* 라면, 이 책은 *"이 유형은 이렇게 반복된다"* 입니다. 읽는 책이 아니라 **옆에 두고 확인하는 책**에 가깝습니다.

미리 밝혀둘 것이 하나 있습니다. 여기 나오는 원인 분석은 대부분 제가 한 게 아닙니다. 저는 "이상한데?"라고 말한 사람이고, 파고들어서 원인을 찾은 건 AI입니다. 어떤 건 이 글을 정리하면서 처음 알았습니다.

그래서 이 책은 **"제가 알아낸 것"이 아니라 "제 프로젝트에서 벌어진 것"** 의 목록입니다. 코드를 못 써도 이 목록은 쓸모가 있습니다. 대부분의 함정은 코드가 아니라 **확인을 건너뛴 자리**에서 나오니까요.

1장. 남의 시스템은 문서를 믿지 마라

사례 9건. 가장 많이 밟은 함정.

증상

```
문서엔 이렇게 써 있는데 실제로는 다르게 동작한다
그리고 에러를 안 낸다
```

사례

요청값을 무시하고 있었습니다.

공공데이터 API에서 건물 정보를 받는데, "한 번에 1000건씩 주세요"라고 요청했습니다. 그런데 실제로는 100건씩만 왔습니다. **문제는 프로그램이 "1000건 요청했으니 1000건 받았겠지"라고 가정하고 다음 페이지를 계산했다는 것**입니다.

결과적으로 7,997건을 받고 "다 받았다"고 판단했습니다. 제대로 세어보니 **26,723건**이었습니다. 3분의 1도 안 받고 완료 처리한 겁니다.

> 응답 `body.numOfRows` 기준으로 수정 후 318페이지/26,723건 전량 수집

성공이라고 답하면서 본문엔 에러를 담아 보냈습니다.

지도 API를 쓰는데 어느 날부터 결과가 비었습니다. 그런데 **HTTP 상태코드는 200(성공)** 이었습니다.

본문을 열어보니 `INCORRECT_KEY`라고 적혀 있었습니다. 운영용 키는 도메인 제한이 걸려 있어서, 요청에 출처(Referer)를 안 붙이면 이렇게 됩니다.

상태코드만 확인하는 코드로는 절대 못 잡습니다.

필드 이름이 문서와 달랐습니다.

증권사 API에서 `appsecret`이라는 이름으로 인증 정보를 보냈는데 계속 거부당했습니다. 실제로는 `secretkey`였습니다. 문서에는 다르게 적혀 있었고요.

동시 접속 제한이 문서에 없었습니다.

실시간 시세를 받으려고 연결을 8개 만들었는데 1개만 살아남았습니다. 알고 보니 해외주식은 **1개만 허용**이었습니다. 어디에도 안 적혀 있었습니다.

요금제를 올려도 안 되는 게 있었습니다.

지도 서비스에서 "도보 15분 거리" 영역을 그리려는데 결과가 이상했습니다. 알아보니 **정점을 19~37개까지만** 만들어줍니다. 유료로 바꾸면 나아질 줄 알았는데 아니었습니다. **요금제와 무관한 알고리즘 특성**이었습니다.

> "유료면 나아진다"는 내 추측이 틀렸던 것

결제 수단을 연결했더니 자동으로 유료가 됐습니다.

AI 서비스를 무료 한도 안에서 쓰고 있었습니다. 다른 이유로 결제 수단을 연결했더니, **그 순간 유료 티어로 전환**됐습니다. "무료 한도까지만 쓰겠다"는 선택지가 없었어요.

공공데이터의 지역 코드가 실제 행정구역과 안 맞았습니다.

서울 자치구별로 데이터를 받는데 일부 구가 계속 비었습니다. 데이터를 제공하는 쪽의 코드 체계와 **실제 행정구역 코드가 달랐던 것**입니다. 어디에도 "우리는 다른 체계를 씁니다"라고 안 적혀 있었습니다.

막힌 이유가 '너무 빨라서'였습니다.

주소를 좌표로 바꾸는 작업을 대량으로 돌렸는데 중간에 연결이 끊겼습니다. 알고 보니 **초당 4건이 상한**이었고, 그걸 넘기면 연결을 끊어버립니다. 물론 "초당 4건"이라는 숫자는 문서에 없었습니다. 실험해서 알아냈습니다.

왜 반복되나

문서는 **만든 사람이 의도한 것**을 적어둔 겁니다. 실제 동작은 그것과 다를 수 있고, 특히 **한계와 예외**는 대개 안 적혀 있습니다.

그리고 남의 시스템은 우리 사정을 봐주지 않습니다. 요청이 잘못되면 조용히 다른 걸 주거나, 성공이라고 답하면서 다른 내용을 보냅니다.

체크리스트

```
□ 요청한 개수와 실제 받은 개수를 세어봤는가
□ 상태코드 말고 본문 내용을 확인했는가
□ 응답에 적힌 값(총 개수·페이지 수)을 기준으로 반복하는가
□ "이 정도면 되겠지"로 넘긴 한계값이 있는가
□ 유료로 바꾸면 해결된다는 건 확인한 사실인가, 추측인가
```

핵심은 하나입니다. 내가 요청한 것 말고 실제로 받은 것을 확인하세요.

2장. 고쳤는데 안 고쳐진 것처럼 보일 때

사례 6건. 제일 사람 미치게 하는 유형.

증상

```
분명히 고쳤는데 화면이 그대로다
다시 고쳐본다. 그대로다
내가 잘못 고쳤나 싶어서 처음부터 다시 본다
```

사례

이틀치 작업이 통째로 반영 안 되고 있었습니다.

회사 PC에서 돌리는 서비스를 이틀간 고쳤습니다. 배포하고, 재시작하고, 확인했습니다. 화면이 그대로였습니다.

원인을 찾는 데 한참 걸렸습니다. **6월 29일부터 떠 있던 프로세스가 계속 살아서 옛날 코드를 서빙 중**이었습니다.

재시작 명령은 실행됐는데 실제로는 아무 일도 안 일어났습니다. 죽이려던 프로세스가 안 죽고, 새로 뜨려던 프로세스는 포트가 이미 잡혀 있어서 실패했거든요. **그래서 옛날 프로세스가 계속 돌았습니다.**

> 오늘 한 작업이 전부 실제 라이브 서버엔 미반영이었음

이걸 세 번 겪었습니다.

```
3월 18일  고아 프로세스가 포트를 잡고 있었음
7월  1일  좀비가 이틀치 배포를 삼킴
6월 17일  또 옛 프로세스가 옛 화면을 서빙
```

세 번 다 증상이 같았습니다. *"고쳤는데 안 고쳐졌다."*

브라우저 캐시도 같은 짓을 합니다.

디자인을 고쳤는데 사파리에서만 반영이 안 됐습니다. 몇 번을 고쳐도 그대로라서 *"내가 잘못 고쳤나"* 하고 코드를 다시 뜯었습니다.

사파리는 스타일 파일을 아주 공격적으로 캐시합니다. 파일 이름 뒤에 `?v=20260411` 같은 걸 붙여서 **다른 파일인 척**해야 새로 받아옵니다.

서버는 살아 있는데 기능만 죽어 있었습니다.

서비스를 재시작했는데 주소 검색만 안 됐습니다. 원인은 **환경변수 없이 실행**했기 때문이었습니다. 프로그램은 정상으로 떠 있으니 죽은 줄도 몰랐습니다.

인덱스는 시작할 때 한 번만 만들어집니다.

검색 데이터를 추가했는데 검색이 안 됐습니다. 데이터는 분명히 들어갔는데요. 검색 인덱스가 **서버 시작 시 1회만 빌드**되는 구조였습니다. 재시작해야 반영됩니다.

여섯 명이 각자 다른 기억을 갖고 있었습니다.

이미지 생성 작업이 자꾸 사라졌습니다. 분명히 시작했는데 진행 상황을 물어보면 "그런 작업 없다"고 했어요.

원인은 **처리 담당자를 6명으로 늘린 것**이었습니다. 작업 목록을 각자 자기 머릿속에만 갖고 있어서, A가 받은 작업을 B에게 물어보면 모른다고 하는 상황이었습니다.

왜 반복되나

“명령을 실행했다”와 “명령이 효과를 냈다”는 다릅니다.

재시작 명령은 성공했다고 나옵니다. 실제로 프로세스가 바뀌었는지는 별도로 확인해야 합니다. 그런데 우리는 대개 명령이 성공하면 됐다고 생각합니다.

체크리스트

- ☐ 재시작 후 프로세스 번호(PID)가 실제로 바뀌었는가
- ☐ 그 포트를 잡고 있는 게 정말 새 프로세스인가
- ☐ 브라우저에서 강제 새로고침을 해봤는가
- ☐ 파일 이름에 버전을 붙여봤는가
- ☐ 인덱스·캐시처럼 시작할 때만 만들어지는 게 있는가
- ☐ 환경변수가 빠지지 않았는가

한 줄로 하면: 고쳤다고 믿기 전에 프로세스 번호를 보세요.

3장. 손으로 되는 게 자동으로는 안 될 때

사례 6건.

증상

```
터미널에서 실행하면 잘 된다
자동으로 돌게 걸어놨다
안 된다. 에러도 없다
```

사례

자동 실행이 외장 디스크를 못 읽었습니다.

월간 데이터 갱신을 자동화했습니다. 손으로 실행하면 잘 됩니다. 자동으로 걸어놨습니다. 안 됩니다. **에러도 안 납니다. 조용히 아무것도 안 합니다.**

맥의 보안 정책 때문이었습니다. 자동 실행되는 프로그램은 외장 디스크 접근이 차단돼 있습니다. 제가 손으로 실행할 땐 제 권한으로 도니까 됐던 거고요.

> 교훈: launchd 잡은 반드시 강제 실행해봐야 함. 수동 셸에서 되는 것이 자동 실행에서 되는 것을 보장하지 않음

시스템 기본 파이썬을 쓰고 있었습니다.

자동 실행이 계속 실패했습니다. 원인은 **맥에 기본으로 깔린 파이썬**을 쓰고 있었기 때문입니다. 제가 설치한 라이브러리들이 거기엔 없었습니다.

없는 파일을 실행하려 하고 있었습니다.

매일 06시에 도는 작업이 계속 실패했습니다. 원인은 단순했습니다. **실행하려는 스크립트 파일이 없었습니다.**

더 심각한 건 이겁니다 — **5월부터 8월까지 매일 실패하는 걸 감지했는데, 아무도 안 고쳤습니다.** 매일 상태 점검에 "실패" 표시가 뜨는데 그냥 넘어갔어요. 익숙해진 겁니다.

라이브러리 하나 추가했더니 서버가 재기동마다 죽었습니다.

지도 관련 라이브러리를 넣었더니 서버가 시작할 때마다 크래시 루프에 빠졌습니다. 맥 특유의 안전 정책 때문인데, **환경변수 하나를 추가하면 해결**됩니다.

재밌는 건 다른 서비스에는 그 설정이 이미 있었다는 겁니다. 예전에 같은 문제를 겪고 넣어뒀던 건데, 새 서비스엔 안 넣었던 거죠.

원격 접속을 끊었더니 서버도 같이 죽었습니다.

회사 PC에 원격으로 접속해서 서버를 띄웠습니다. 잘 돌았습니다. 접속을 끊었더니 **서버도 같이 종료**됐습니다.

원격으로 띄운 프로그램은 그 접속의 자식으로 취급돼서, 부모가 사라지면 같이 사라집니다. 별도로 "이건 계속 살아 있어라"고 등록해야 합니다.

로그가 안 보여서 멈춘 줄 알았습니다.

몇 시간 걸리는 작업을 돌렸는데 화면에 아무것도 안 나왔습니다. 죽은 줄 알고 껐다가 다시 켰습니다.

안 죽어 있었습니다. **출력이 버퍼에 쌓여서 화면에 안 나온 것뿐**이었습니다. 진행 상황을 볼 수 없으니 살아 있는지 죽었는지 판단할 방법이 없었던 겁니다.

왜 반복되나

자동 실행은 나와 다른 사람으로 취급됩니다.

```
내가 실행      내 권한 · 내 환경변수 · 내 파이썬
자동 실행      제한된 권한 · 빈 환경 · 시스템 기본값
```

터미널에서 되는 걸 확인하고 "됐다"고 생각하는 게 함정입니다.

체크리스트

- ☐ 자동 등록 후 강제로 한 번 실행해봤는가
- ☐ 실행되는 프로그램의 전체 경로를 지정했는가
- ☐ 필요한 환경변수를 명시했는가
- ☐ 외장 디스크·네트워크 폴더에 접근하는가
- ☐ 매일 뜨는 "실패" 표시를 그냥 넘기고 있지 않은가

마지막 줄이 중요합니다. 실패가 익숙해지면 그건 이미 없는 기능입니다.

4장. 조용히 실패하는 것들

사례 8건. 가장 위험한 유형.

증상

```
에러가 안 난다
화면도 멀쩡하다
그런데 결과가 없거나 틀렸다
```

사례

화면 전체가 백지가 됐는데 콘솔에도 안 떴습니다.

작업 화면이 통째로 안 뜨는 문제가 있었습니다. 콘솔을 열어봐도 에러가 없었습니다.

원인은 코드 한 곳에서 **같은 변수를 두 번 선언**한 것이었습니다. 이건 문법 오류라서, 그 파일의 코드가 **통째로 실행되지 않습니다.** 실행이 안 되니 에러도 안 납니다.

> 파싱 에러는 브라우저에서 무음 실패 (콘솔에도 안 뜰 수 있음)

검증 기능이 몇 주 동안 작동 안 하고 있었습니다.

답변 품질을 검증하는 기능을 붙여놨습니다. 코드 감사를 하다가 발견했는데, **그 기능이 호출하는 모델이 실제로는 없었습니다.** 그래서 매번 실패하고 예비 경로로 넘어가고 있었어요.

실패해도 다음 단계가 있으니 겉으로는 멀쩡했습니다. **검증 없이 통과시키고 있었던 겁니다.**

법령 검색이 전부 실패하고 있었습니다.

후속 질문을 하면 관련 법령이 하나도 안 붙었습니다. 원인은 **반환 타입을 잘못 쓴 것**이었습니다. 함수가 A 형태로 주는데 B 형태로 받아 쓰고 있었어요.

에러는 안 났습니다. 그냥 빈 결과가 나왔을 뿐입니다.

“그냥 넘어가라”가 문제를 숨겼습니다.

5,000줄짜리 파일을 나누다가 필요한 선언들이 빠졌습니다. 그런데 일부는 에러조차 안 났습니다. 코드에 **“무슨 일이 생기든 그냥 넘어가라”** 는 설정이 있었거든요.

원래는 사소한 오류로 서비스가 죽는 걸 막으려고 넣는 건데, 이번엔 **고장을 숨기는 역할**을 했습니다.

제목이 없는 문서 14건이 조용히 버려졌습니다.

법령 데이터를 넣는데 몇 건이 안 들어갔습니다. 원인은 **제목 칸이 비어 있는 문서**였습니다. 저장하려다 실패했는데, 실패한 걸 세지 않고 넘어갔습니다.

전체 수천 건 중 14건이라 눈에 안 띄었습니다. 그런데 그 14건이 필요한 순간에 없으면 답이 틀립니다.

한 글자 차이로 조례 전체가 검색에서 빠졌습니다.

지자체 조례가 검색에 하나도 안 잡혔습니다. 데이터는 다 들어가 있는데요.

원인은 검색 조건을 쓰는 방식이었습니다. **“포함”이라는 조건에 값을 리스트로 넣어야 하는데 문자열로 넣고 있었습니다.** 조용히 아무것도 안 맞는 조건이 됐고, 에러 대신 빈 결과가 나왔습니다.

지도 범위를 넓히면 필지가 사라졌습니다.

넓은 지역을 조회하면 일부 땅이 안 나왔습니다. 코드 안에 **“최대 몇 개까지만 가져오기”** 라는 숫자가 박혀 있었는데, 그걸 넘으면 나머지를 그냥 안 줬습니다. 경고도 없이요.

비교 대상이 하나도 없을 때 저장이 통째로 실패했습니다.

계산 결과를 저장하는데 어느 날부터 전부 실패했습니다. 원인은 **비교할 게 하나도 없을 때 “무한대”라는 값을 넣도록 만든 것**이었습니다.

그 값은 파일로 저장할 수 없는 값입니다. 그래서 그 한 칸 때문에 **파일 전체 저장이 실패**했습니다.

왜 반복되나

우리는 에러를 기준으로 문제를 판단합니다. 에러가 없으면 괜찮다고 생각하죠.

그런데 조용한 실패는 그 반대입니다. **에러가 안 나는 게 증상**입니다.

체크리스트

- [ ] 결과가 "비어 있음"인지 "정상적으로 비어 있음"인지 구분되는가
- [ ] 예비 경로(fallback)로 넘어갈 때 기록을 남기는가
- [ ] "그냥 넘어가라" 처리가 어디에 있는지 아는가
- [ ] 화면이 안 뜨면 문법 검사부터 돌려보는가
- [ ] 잘 되는지 눈으로 확인했는가, 에러가 없는 것만 확인했는가

사례 13건.

증상

"완료했습니다"
그런데 파일을 열어보면 안 바뀌어 있다

사례

exit 0인데 아무것도 안 고쳤습니다.

큰 파일 수정을 AI에게 맡겼습니다. **"완료했습니다"** 하고 정상 종료했습니다. 파일을 열어보니 **변경사항 0건**이었습니다.

이게 명확한 실패보다 나쁩니다. 실패했으면 다시 시키면 되는데, 성공했다고 하니 다음 단계로 넘어가거든요.

같은 파일에서 두 번 망가뜨렸습니다.

한 화면 파일이 있는데, 여기에 AI를 붙일 때마다 깨졌습니다.

```
4월 9일   기능 추가 →  없는 함수를 지어내서 호출, 이상한 문자 삽입
4월 10일  디자인만 →  또 깨짐
```

두 번째는 "디자인만 손대니까 안전하겠지"였는데 아니었습니다. 그래서 **이 파일은 AI에게 안 맡긴다**고 규칙으로 못 박았습니다.

시키지 않아도 요약해버립니다.

보고서를 만드는데 내용이 자꾸 짧아졌습니다. 원본을 다 넣으라고 해도 그랬습니다.

원인은 구조였습니다. 원본을 AI에게 주고 정리시키는 단계가 있었는데, **거기서 자연스럽게 요약이 일어났습니다.** 지시로는 못 막습니다.

해결은 구조를 바꾸는 거였습니다. AI에게는 뼈대만 만들게 하고, **본문은 원본을 그대로 붙였습니다.**

검증을 AI에게 시키면 같은 함정에 빠집니다.

검색에서 뭐가 빠졌는지 확인하는 기능을 만들려고 했습니다. 처음엔 그것도 AI에게 시키려 했는데, 이런 지적을 받았습니다.

> 누락 검증을 또 LLM으로 돌리면 같은 함정 — 결정론적 완비성 검사로 구현해야 안전

빠뜨린 놈에게 "뭐 빠뜨렸어?"라고 물어보는 셈입니다. 그래서 규칙 기반 체크리스트로 만들었습니다.

temperature=0인데도 결과가 달랐습니다.

같은 코드를 두 번 돌렸는데 결과가 달랐습니다. 무작위성을 0으로 설정했는데도요.

```
같은 코드 2회 실행 → 누락 7건 / 추가 9건 차이
```

그래서 개선 효과를 측정할 때 **먼저 "아무것도 안 바꾸고 두 번 돌린 차이"부터 재게** 됐습니다. 그 차이보다 작으면 의미 없는 변화니까요.

한계는 AI가 아니라 제가 준 형식이었습니다.

패널 배치를 AI에게 시키는데 결과가 계속 엉망이었습니다. *"모델을 바꿔야 하나"* 싶었습니다.

알고 보니 **제가 준 표현 형식이 문제**였습니다. "몇 번째 줄, 몇 번째 칸" 식으로 설명하게 했는데, 이게 표현력이 부족했어요. 좌표 방식으로 바꾸니 바로 해결됐습니다.

모델을 바꾸기 전에 입출력 형식을 의심해보세요.

3단 위임은 실패했습니다.

한때 이런 구조로 일했습니다.

```
나 → 검증하는 AI → 코딩하는 AI
```

검증 담당을 두면 좋아질 줄 알았는데 반대였습니다. 파일이 계속 바뀌는 상황에서 **보고가 한 박자씩 늙습니다.** 검증 AI가 "숫자가 안 맞습니다"라고 하면, 그 사이 또 바뀌어 있고요.

그리고 제가 "그만"이라고 해도 **중간 단계가 자기 판단으로 계속 움직였습니다.**

같은 지시를 두 번 보냈습니다. 두 번이나요.

AI에게 작업을 시켰는데 답이 없었습니다. 한참 기다리다 다시 보냈습니다. 알고 보니 **그쪽 세션이 리셋돼서 제 지시를 못 받은 상태**였습니다.

같은 일이 또 있었습니다. 이번엔 반대로 **이미 처리 중인 걸 모르고 또 보내서** 같은 작업이 두 번 돌았습니다.

5,000줄 리팩토링은 AI보다 손이 빨랐습니다.

큰 파일을 나누는 작업을 AI에게 맡겼다가 실패하고, 결국 직접 했습니다. 명령어 몇 줄로 자르고 손으로 정리하는 게 **훨씬 빠르고 정확**했습니다.

AI가 잘하는 건 "무엇을 어떻게 나눌지 판단하는 것"이지, 5,000줄을 옮기는 노동이 아니었습니다.

코드 점검도 마찬가지였습니다.

"이 함수가 어디서 쓰이는지 다 찾아줘"를 AI에게 시켰는데 느리고 빠뜨렸습니다. **검색 명령 한 줄이 더 정확**했습니다.

> 기계적으로 셀 수 있는 건 기계에게, 판단이 필요한 건 AI에게

AI가 못 하는 영역이 있습니다.

회사 PC에 뭔가를 설치하고 방화벽을 열고 계정 권한을 주는 일 — 이건 **제가 직접 그 앞에 앉아서** 해야 했습니다. 원격으로 붙어 있어도 마지막 확인 버튼은 사람이 눌러야 합니다.

AI가 저를 다른 이름으로 불렀습니다.

기본 설정에 남아 있던 예시 이름을 그대로 쓰고 있었습니다. 사소해 보이지만, **AI가 뭘 참고하고 있는지 제가 모르고 있었다**는 뜻이기도 합니다.

왜 반복되나

AI는 **맥락을 모르고, 자기가 뭘 모르는지도 모릅니다.** 그리고 대부분의 경우 그럴듯한 답을 내놓습니다.

체크리스트

```
□ "완료했습니다" 받고 파일을 직접 열어봤는가
□ 맡긴 작업이 500줄을 넘는가 (넘으면 쪼개거나 직접)
□ 검증을 또 AI에게 시키고 있지 않은가
□ 개선 효과를 재기 전에 "아무것도 안 한 차이"를 먼저 쟀는가
□ 모델을 바꾸기 전에 입출력 형식을 의심했는가
□ 위임이 2단계를 넘는가
□ 멈추라고 할 때 정말 멈추는 구조인가
```

6장. 숫자가 거짓말할 때

사례 10건.

증상

결과가 아주 좋다
그런데 뭔가 이상하다

사례

수익률 64만 퍼센트가 나왔습니다.

전략 검증을 돌렸더니 $500이 **$3,248,329**가 됐습니다. 649,565%입니다.

버그였습니다. 특정 조건에서 청산할 때, **보유 종목이 아니라 시장 지수 가격으로 계산**하고 있었습니다.

고치니까 $6.74가 됐습니다. **−98.65%.** 그게 진짜 성능이었습니다.

1등 전략이 물리적으로 실행 불가능했습니다.

네 가지 매매 방식을 비교했더니 한 조합이 압도적이었습니다. 수익률이 다른 것의 **2배 이상**이었습니다.

그런데 그 방식은 **"그날 종가에 산다"** 였습니다. 매수 신호는 장 마감 후에 나오는데요. **시간을 거스르지 않으면 불가능합니다.**

수익률 절반짜리 조합을 골랐습니다.

> 데이터를 보여주는 것과 "이게 더 좋으니 바꾸자"는 다르다 실행 가능성을 먼저 확인하고 백테스트해야 한다 조합 비교 = 파라미터 추가 = 과적합 위험

데이터를 6배 늘리자 성과가 사라졌습니다.

한 신호를 테스트했더니 결과가 좋았습니다. 442건 기준이었습니다.

데이터 수집을 개선해서 2,978건으로 늘렸습니다. **성과가 완전히 증발했습니다.**

> 적은 데이터에서 좋아보이면 생존 편향 의심

N=15짜리를 좋다고 할 뻔했습니다.

기존 전략에 조건을 하나 더 얹었더니 승률이 73%로 뛰었습니다. 그런데 해당 사례가 15건이었습니다. 통계적으로 아무 의미 없는 숫자입니다.

"미래를 아는" 조건으로 테스트하고 있었습니다.

실적 발표 전날 사는 전략을 테스트했더니 16개 조합 중 15개가 통과했습니다.

문제는 **"실적이 좋게 나올 종목"을 미리 알아야** 한다는 것이었습니다. 발표 전날엔 아무도 모릅니다.

만점 지표가 사실은 68점 만점이었습니다.

점수 체계를 100점 만점으로 설계했는데, 가중치를 적용하다 보니 실제 최대값이 68점이었습니다. 그래서 "65점 이상이면 매수" 같은 기준이 의도와 전혀 다르게 작동했습니다.

지표를 개선했더니 -99%가 됐습니다.

분석 결과 "이 구간이 좋다"가 나왔습니다. 그래서 그 항목의 가중치를 키웠습니다.

−99%.

분석 결과를 잘못 읽은 거였습니다. "그 구간일 때만 진입하라"는 뜻이었는데, 가중치를 키우니 **모든 신호가 기준을 통과**해버렸습니다.

같은 로직인데 표본 수가 전부 달랐습니다.

세 군데에서 같은 조건으로 집계했는데 나오는 건수가 다 달랐습니다. 조건은 분명히 같았는데요.

원인은 **같은 판정을 두 곳에서 각각 계산**하고 있었기 때문입니다. 한쪽만 고치면 다른 쪽이 어긋납니다. 결국 판정 함수를 하나로 합쳤습니다.

> 같은 판정을 두 군데서 하면 언젠가 어긋난다

논문에 나온 성과도 수수료를 넣으면 적자였습니다.

Sharpe 3.8이라는 좋은 전략을 참고했습니다. 그대로 구현하고 **실제 거래 비용을 넣었더니 적자**가 됐습니다.

논문은 비용을 빼고 계산한 거였습니다. 거래 횟수가 많은 전략일수록 이 차이가 치명적입니다.

왜 반복되나

좋은 결과가 나오면 검증 의욕이 떨어집니다. 나쁜 결과는 원인을 찾는데, 좋은 결과는 그냥 믿게 됩니다.

체크리스트

```
□ 결과가 비현실적으로 좋은가 (그러면 버그다)
□ 이 전략을 실제로 실행할 수 있는가
□ 표본이 몇 개인가 (30개 미만이면 의미 없음)
□ 미래 정보를 쓰고 있지 않은가
□ 데이터를 더 모으면 결과가 유지되는가
□ 지표의 실제 최대값이 의도와 같은가
□ 이 검사를 통과하면서 최대한 망가뜨릴 방법이 있는가
```

마지막 질문이 제일 유용합니다. 답이 쉽게 나오면 그 지표는 부족한 겁니다.

7장. 데이터가 배신할 때

사례 12건.

증상

```
프로그램은 정상이다
그런데 답이 이상하다
알고 보니 데이터의 전제가 틀렸다
```

사례

땅의 중심점 79%가 엉뚱한 곳에 있었습니다.

지도에서 필지를 클릭하면 엉뚱한 게 선택되는 문제가 있었습니다.

원인은 **중심점 계산**이었습니다. 위도·경도 숫자를 그대로 넣고 계산했는데, 값이 127.0523… 같은 식이라 작은 필지에서는 소수점 아래가 뭉개집니다.

전체를 검사했더니:

```
6,088개 필지 중
자기 영역 밖에 중심점이 찍힌 것   4,811개 (79%)
최대 오차                          95km
```

대로가 지적도에 도로로 등록돼 있지 않았습니다.

건물과 도로의 거리를 계산하는 기능이 자꾸 틀렸습니다. 테헤란로 바로 옆 필지인데 **"도로에 안 접한다"** 고 나왔어요.

원인은 데이터였습니다. 프로그램은 지적도에서 지목이 '도로'인 필지를 찾는데, **테헤란로 같은 큰 길은 거기 등록돼 있지 않습니다.** 가장 가까운 '도로' 필지가 73m 밖에 있었어요.

코드는 정확했습니다. 데이터의 전제가 틀렸습니다.

도로 하나가 동네 전체였습니다.

지도에서 아무 데나 클릭하면 화면 전체가 파랗게 칠해지는 일이 있었습니다.

알고 보니 공공 지적도에서 **도로는 지번 하나가 동네 전체 도로망을 덮는 단일 도형**이었습니다. 실측하니 210개 조각이 하나로 묶여 2.6km ×

2.4km를 덮고 있었습니다.

같은 조문이 '77'과 '제48조'로 섞여 있었습니다.

법령 링크가 어떤 건 되고 어떤 건 안 됐습니다. 데이터베이스를 보니 조문 번호가 **숫자만 있는 것과 '제~조'가 붙은 것이 섞여** 있었습니다.

5,000자 덩어리라 검색이 안 됐습니다.

법령 부록 표가 통째로 하나의 덩어리로 저장돼 있었습니다. 그러니 "수목장림" 같은 세부 항목을 찾을 수가 없었어요. 3,059개 항목으로 쪼개니 해결됐습니다.

서울시 조례가 검색에서 밀려났습니다.

서울 땅을 검토하는데 서울시 조례가 안 나왔습니다. 검색 알고리즘이 본문 단어만 보다 보니, **다른 시·군 조례에 순위가 밀려서 아예 후보에 못 들어간** 거였습니다.

"15층 이하"를 "높이 15m"로 읽었습니다.

지구단위계획에서 높이 제한을 읽는데, **층수를 미터로 잘못 해석**했습니다.

하룻밤 작업이 깨진 파일 하나로 날아갔습니다.

며칠간 모은 데이터를 하나의 큰 파일에 저장하고 있었습니다. 304MB짜리였습니다.

어느 날 그 파일이 **중간에 잘려서** 읽히지 않았습니다. 저장 중에 뭔가 끊긴 겁니다. 하룻밤 작업이 통째로 사라졌습니다.

중단 신호를 안 받아서 75%를 잃었습니다.

장시간 수집 작업이 시스템에 의해 종료됐습니다. 그때까지 모은 것의 **75%가 메모리에만 있었고 저장되지 않았습니다.**

"끝나면 한 번에 저장" 구조의 대가였습니다. 중간중간 저장하도록 바꿨습니다.

상장폐지된 종목의 가격이 없었습니다.

과거 데이터를 분석하는데 일부 종목의 가격이 통째로 비어 있었습니다. 원인은 **상장폐지된 종목은 일반 서비스에서 데이터를 지워버린다**는 것이었습니다.

이게 왜 중요하냐면, 망한 종목만 빠지면 **결과가 실제보다 좋게 나옵니다.** 다른 경로로 505개를 복구해서 넣었습니다.

시간대가 달라서 날짜가 하루 밀렸습니다.

미국 주식 데이터를 한국 시간 기준으로 처리하고 있었습니다. 그래서 **진입 날짜가 하루씩 밀렸습니다.** 하루 차이가 결과를 완전히 바꿉니다.

왜 반복되나

데이터는 만든 사람의 목적에 맞게 만들어집니다. 우리 목적과 다를 수 있고, 대개 그 차이는 문서에 안 적혀 있습니다.

체크리스트

- ☐ 이 데이터의 원래 목적이 무엇인가
- ☐ 같은 항목이 여러 형태로 섞여 있지 않은가
- ☐ 하나의 항목이 비정상적으로 크지 않은가
- ☐ 계산 전에 값의 범위를 정규화했는가
- ☐ 결과의 몇 %가 이상한지 전수로 세어봤는가
- ☐ 검색 결과에 있어야 할 게 아예 후보에도 못 들어가지 않았는가

"몇 %가 이상한가"를 세는 게 핵심입니다. 하나씩 보면 예외 같은데, 세어보면 79%일 수 있습니다.

8장. 내가 나를 망가뜨린 방법

사례 15건. 남 탓 못 하는 것들.

사례

코드 치환이 엉뚱한 데 먼저 걸렸습니다.

지도 화면에 기능을 추가하려고 특정 코드를 찾아 바꿨습니다. 화면이 깨졌습니다.

찾으려던 패턴이 **함수 안쪽에 먼저 등장**해서, 엉뚱한 곳이 바뀐 겁니다.

그리고 한 달 뒤에 또 그랬습니다.

발표 자료의 한글을 일괄 치환했는데, 그 문자열이 **이미지·영상 데이터 안에도 들어 있었습니다.** 이미지가 깨지고 영상이 안 나왔습니다.

```
4월  6일  코드 치환이 엉뚱한 데 걸림 → 교훈 기록
4월 30일  전역 치환이 이미지 데이터를 오염 → 같은 유형
```

교훈을 적어놨는데도 다시 밟았습니다.

롤백하다 옆 줄까지 지웠습니다.

기능 하나를 되돌리면서 관련 코드를 지웠는데, **바로 옆에 있던 다른 기능의 코드까지** 지웠습니다. 화면이 안 떴습니다.

검증 스크립트에 필터를 실수로 넣었습니다.

검증을 돌렸더니 결과가 예전과 달랐습니다. 한참 원인을 찾다가 발견했는데, **제가 검증 스크립트에 필터를 하나 넣어놨던** 거였습니다. 그것 때문에 136건이 빠졌습니다.

서버와 스크립트가 동시에 DB를 만졌습니다.

데이터베이스 인덱스가 깨졌습니다. 원인은 **서버가 돌고 있는 상태에서 별도 스크립트로 같은 DB를 건드린 것**이었습니다.

진행 상태 파일을 안 지웠습니다.

DB를 초기화하고 전체를 다시 넣었는데 261개가 빠졌습니다. **"어디까지 했는지" 기록해두는 파일을 안 지운 것**이 원인이었습니다. 프로그램이 "이미 했다"고 판단하고 건너뛴 겁니다.

내 사본이 더 오래된 걸 모르고 덮어썼습니다.

회사 PC에 있는 스크립트를 고치려고 제 맥에 있는 걸 수정해서 보냈습니다. **제 사본이 더 오래된 버전이었습니다.** 회사 PC의 최신 작업을 덮어썼어요.

플러그인 하나 바꿨더니 스마트홈 설정이 전부 날아갔습니다.

집에 있는 기기 연동 플러그인을 교체했습니다. 설정 파일은 백업했습니다.

그런데 **방 배정·기기 이름·자동화가 전부 풀렸습니다.** 그건 설정 파일이 아니라 다른 곳에 저장돼 있었거든요. 9개 기기를 손으로 다시 설정했습니다.

백업 규칙을 하루에 네 번 어겼습니다.

이건 그냥 기록에 이렇게 적혀 있습니다.

> 백업 먼저 — 🔴 2026-07-26 하루에 4번 어긴 항목. 예외 없음.

초기 설정을 다시 돌렸더니 설정이 날아갔습니다.

문제를 해결하려고 초기 설정 마법사를 다시 실행했습니다. **그게 기존 설정을 덮어썼습니다.** AI 하위 설정들이 전부 사라졌어요.

“다시 설정하면 되겠지”라고 생각한 게 화근이었습니다.

자동 업데이트가 설정에 뭔가를 심었습니다.

어느 날부터 AI가 매 응답마다 즉시 죽었습니다. 원인은 **앱이 자동 업데이트되면서 설정 파일에 자기 코드를 주입**한 것이었습니다.

제가 바꾼 게 없는데 망가지는 상황이 제일 찾기 어렵습니다. “내가 뭘 건드렸지?”부터 생각하니까요.

설정 하나가 15초 만에 AI를 죽였습니다.

AI 하위 작업이 계속 16초 만에 끝났습니다. 인증 문제인 줄 알고 한참 헤맸는데, **제가 “최대한 깊게 생각하라”고 설정해둔 것**이 원인이었습니다.

깊게 생각하느라 15초 제한을 넘겼고, 그래서 강제 종료된 거였습니다. **제가 켠 옵션이 제 작업을 죽이고 있었습니다.**

스크래퍼가 제 브라우저를 통째로 닫았습니다.

자료 수집 프로그램을 돌렸는데, **제가 로그인해서 쓰고 있던 브라우저 창까지 전부 닫혔습니다.** 작업 중이던 탭들도 같이요.

프로그램이 “브라우저 정리”를 하면서 제 것까지 정리한 겁니다.

리사이즈 핸들이 9픽셀이었습니다.

패널 크기를 조절하는 손잡이를 만들었는데 잘 안 잡혔습니다. 재보니 실제로 잡히는 폭이 **9픽셀**이었어요. 다른 요소에 가려서 잘려 있었습니다.

만든 사람은 어디를 눌러야 하는지 아니까 잘 됩니다. 처음 쓰는 사람은 그걸 모릅니다.

왜 반복되나

"이건 간단하니까" 라고 생각하는 순간입니다.

간단한 치환이 이미지를 깨고, 간단한 롤백이 옆 기능을 지우고, 간단한 플러그인 교체가 설정을 날립니다.

체크리스트

- ☐ 백업했는가 (판단하지 말고 그냥)
- ☐ 치환할 패턴이 다른 곳에도 있는가
- ☐ 지우려는 줄 옆에 다른 기능이 있는가
- ☐ 서버가 돌고 있는데 같은 데이터를 만지려 하는가
- ☐ 진행 상태를 기록하는 파일이 있는가
- ☐ 내 사본이 원본보다 최신인가
- ☐ 설정 파일 말고 다른 데 저장되는 것이 있는가

9장. 진단이 틀릴 때

사례 6건. 증상을 고치고 원인은 그대로 두는 경우.

증상

```
원인을 찾았다고 생각했다
고쳤다
또 터졌다
```

사례

사용자 관찰이 제 가설 두 개를 이틀 연속으로 죽였습니다.

도면 처리가 느린 원인을 찾고 있었습니다. 측정해보고 “여기가 병목이다”라고 판단했습니다.

그런데 영빈님이 실제로 쓰면서 관찰한 게 달랐습니다. 제 측정은 **실제 문제가 생기는 상황과 다른 조건**에서 한 거였어요.

> 사용자 관찰이 가설 두 개를 죽였음(이틀 연속) — 진단은 반드시 실제 문제가 발생하는 프로세스 기준으로 측정할 것

정규식으로 추측하다 두 번 연속 오탐했습니다.

보고서에 이상한 문자가 남는 문제가 있었습니다. 패턴을 만들어서 몇 건인지 세어봤습니다. 550건이 나왔습니다.

패턴이 잘못돼서 정상 문서까지 걸린 거였습니다. 고쳐서 다시 셌더니 334건. 이것도 틀렸습니다.

그래서 **추측을 그만두고 실제로 변환을 돌려봤습니다.** 진짜 잔존은 12개였습니다.

> 정규식으로 추측하지 말고 실제로 변환해서 결과를 볼 것

원인은 모델이 아니라 배달부였습니다.

매일 정해진 시간에 오는 알림이 안 왔습니다. 처음엔 AI 모델 호출이 실패하는 줄 알았습니다. 모델을 바꿔보고, 설정을 바꿔보고, 계속 안 됐습니다.

진짜 원인은 **작업을 실행시키는 스케줄러가 60초 동안 멈춰 있는 것**이었습니다. 모델은 아무 문제가 없었어요.

첫 증상만 고쳤다가 발표 중에 사고 날 뻔했습니다.

발표 자료에서 영상이 안 나오는 문제를 고쳤습니다. 됐다고 생각했습니다.

그런데 **영상을 튼 다음에 슬라이드가 안 넘어갔습니다.** 첫 증상만 잡고 그 뒤 흐름을 안 봤던 거예요.

> 첫 증상만 고치고 끝내면 발표 중 사고. 브라우저 도구로 실제 클릭·키 입력까지 검증할 것

"유료면 나아지겠지"가 틀렸습니다.

지도에서 도보권 영역을 그리는데 모양이 거칠었습니다. 무료 요금제라 그렇겠거니 하고 유료 전환을 검토했습니다.

알아보니 **요금제와 무관한 알고리즘 특성**이었습니다. 돈을 내도 똑같았을 겁니다.

진단 도구가 권하는 대로 했더니 실패했습니다.

시스템 점검 도구가 "이 설정을 이렇게 바꾸라"고 계속 권고했습니다. 그대로 했더니 **서비스가 안 떴습니다.**

공식 권고가 현재 버전에서는 안 맞는 상황이었어요. 결국 권고를 무시하고 기존 설정을 유지했습니다.

왜 반복되나

그럴듯한 원인을 찾으면 거기서 멈춥니다.

특히 이런 것들이 위험합니다.

```
"측정해봤더니 여기가 느리다"    ← 실제 상황에서 측정했나
"패턴으로 세어보니 N건이다"      ← 패턴이 맞나
"공식 도구가 이렇게 하래"        ← 지금 버전에서도 맞나
"유료면 되겠지"                  ← 확인한 사실인가
```

체크리스트

- ☐ 실제로 문제가 나는 그 조건에서 측정했는가
- ☐ 추측(패턴·계산) 대신 실제로 돌려봤는가
- ☐ 증상을 고친 뒤 그다음 동작까지 해봤는가
- ☐ 사용자가 관찰한 것과 내 측정이 일치하는가
- ☐ "이러면 되겠지"가 확인한 사실인가

제일 강력한 건 이겁니다. 추측하지 말고 돌려보세요. 550건 → 334건 → 12건이 그 차이입니다.

10장. 나중에 문제가 되는 것들

사례 7건. 당장은 아무 일도 안 생기는 것들.

사례

관리자 비밀번호에 기본값이 박혀 있었습니다.

보안 점검을 하다 발견했습니다. 환경변수가 없으면 `6165` **라는 기본 비밀번호로 동작**하게 돼 있었습니다. 개발할 때 편하려고 넣은 게 그대로 남은 거죠.

코드를 넘기면 회원 정보가 같이 나갈 뻔했습니다.

누군가에게 코드를 공유하려고 폴더를 통째로 압축하려던 참이었습니다. 그 안에 뭐가 들었는지 확인해봤습니다.

```
회원 189명 실명·이메일
이용기록 4,485행
저장된 보고서 30건
서비스 계정 인증 파일
환경설정 파일 (API 키 23개)
```

그리고 제가 직접 쓴 개인정보처리방침에 **"제3자에게 제공하지 않습니다"**라고 적혀 있었습니다.

쓰던 라이브러리가 상용 배포에 저자 합의가 필요했습니다.

3D 계산에 쓰던 라이브러리를 확인해보니, **상업적으로 배포하려면 원저자와 별도 합의**가 필요한 조건이었습니다. 무료로 쓸 땐 문제없지만 유료화하면 걸립니다.

폰트도 마찬가지였습니다.

문서 변환에서 글씨가 깨지는 문제가 있었습니다. 원인은 필요한 폰트가 없어서였는데, **그 폰트들은 합법적으로 다운로드할 방법이 없었습니다.** 상용 폰트라서요.

결제를 연결하니 자동으로 유료가 됐습니다.

AI 서비스를 무료 한도 안에서 쓰고 있었습니다. 다른 이유로 결제 수단을 연결했더니, **그 순간 유료 티어로 전환**됐습니다. 요금이 붙기 시작했어요.

실제로 재보니 쓸수록 손해였습니다.

사용량 기록을 붙이고 실제 비용을 계산해봤습니다.

```
하루 35건 처리    = 50,887원
건당                1,454원
매출                0원
월 환산             87만 ~ 153만원
```

감으로는 "얼마 안 하겠지"였는데, 재보니 아니었습니다.

왜 반복되나

이런 건 에러를 안 냅니다. 서비스는 잘 돌아가고, 사용자도 불편이 없습니다. 문제는 나중에 터집니다 — 공유할 때, 유료화할 때, 감사받을 때.

체크리스트

```
□ 기본값으로 동작하는 비밀번호·키가 있는가
□ 이 폴더를 통째로 넘기면 뭐가 같이 나가는가
□ 쓰는 라이브러리·폰트의 상업적 사용 조건을 확인했는가
□ 무료 한도가 어떤 조건에서 풀리는가
□ 실제 원가를 재봤는가, 감으로 짐작하고 있는가
□ 내가 쓴 약관·방침과 실제 동작이 일치하는가
```

마지막 줄이 의외로 중요합니다. 개인정보처리방침은 쓸 때 잘 쓰고 나서 잊어버리기 쉽습니다.

맺으며 — 그래서 무엇이 남았나

이 목록을 만들며 알게 된 것

처음 밟았을 때 교훈을 적어뒀는데도 다시 밟았습니다.

```
#API              6번
#데이터           4번
#좀비프로세스     3번
#launchd          3번
#문자열치환       2번
#CSS              2번
```

기록만으로는 부족했습니다. 적어두는 것과 **작업 전에 확인하는** 것은 다르니까요.

그래서 이 책은 읽는 책이 아닙니다

각 장 끝에 체크리스트를 붙인 이유가 이겁니다. **작업 시작 전에 해당 장을 펴서 확인하는 용도**로 만들었습니다.

특히 이 일곱 개는 거의 모든 상황에 적용됩니다.

```
1. 요청한 것 말고 받은 것을 확인하라
2. 고쳤다고 믿기 전에 프로세스 번호를 봐라
3. 자동 실행은 반드시 강제로 한 번 돌려봐라
4. 에러가 없는 것과 잘 되는 것은 다르다
5. 추측하지 말고 실제로 돌려봐라
6. 넘기기 전에 뭐가 같이 나가는지 열어봐라
7. 백업은 판단하지 말고 그냥 해라
```

마지막으로

이 목록의 168개 항목 중 **제가 원인을 직접 찾은 건 손에 꼽습니다.**

대부분은 AI가 파고들어서 알아냈고, 저는 *"이거 왜 이래?"* 라고 물은 사람입니다. 어떤 건 이 책을 정리하면서 처음 알았고요.

그런데 그 질문이 없었으면 아무것도 시작되지 않았습니다. 검사가 전부 초록불일 때 화면을 본 것도, *"이게 단지 안 도로 때문이야?"* 라고 범위를 좁힌 것도, *"멈춰"* 라고 한 것도 제가 했습니다.

코드를 못 써도 이 목록은 쓸 수 있습니다. 여기 있는 함정 대부분은 코드 실력이 아니라 **확인을 건너뛴 자리**에서 나왔으니까요.

그리고 하나 더 — 이 168개는 **6개월간 실제로 뭔가를 만들었기 때문에** 생긴 것들입니다. 안 만들었으면 실패도 없었을 거고, 이 책도 없었을 겁니다.

넘어진 자리가 결국 남는 게 맞더군요.

2026년 8월, 서울에서

겪은 사람: 나 / 쓴 것: AI / 재료: 6개월치 기록 222개 파일