

# AI로 개발을 어떻게 가속화하는가

## -드디어 열린 입코딩 시대-

네이버의 전사 개발자들과 디캠프의 스타트업들  
대상으로 했던 강의의 공개버전입니다.  
(2025.7월)

하용호

yongho.ha@gmail.com





데이터사이언티스트

NUMBER  
WORKS

창업&엑싯

kakao

비즈 데이터&이사



투자&컨설팅

지금은 아래의 것들을 하면서 삽니다.



CEO : 데이터와 AI를 잘 쓰고 싶은 회사들에 컨설팅을 합니다.



CDO : 거대 트래픽 대응 실시간 광고 엔진을 만들고 있습니다.

↳ indent  
~ 2024.12

CAIO : 커머스에 적용되는 여러 AI를 만들고 있습니다.



6살 때부터 코딩을 시작해 지난 삼십X년간 프로그래밍을 해왔다. 



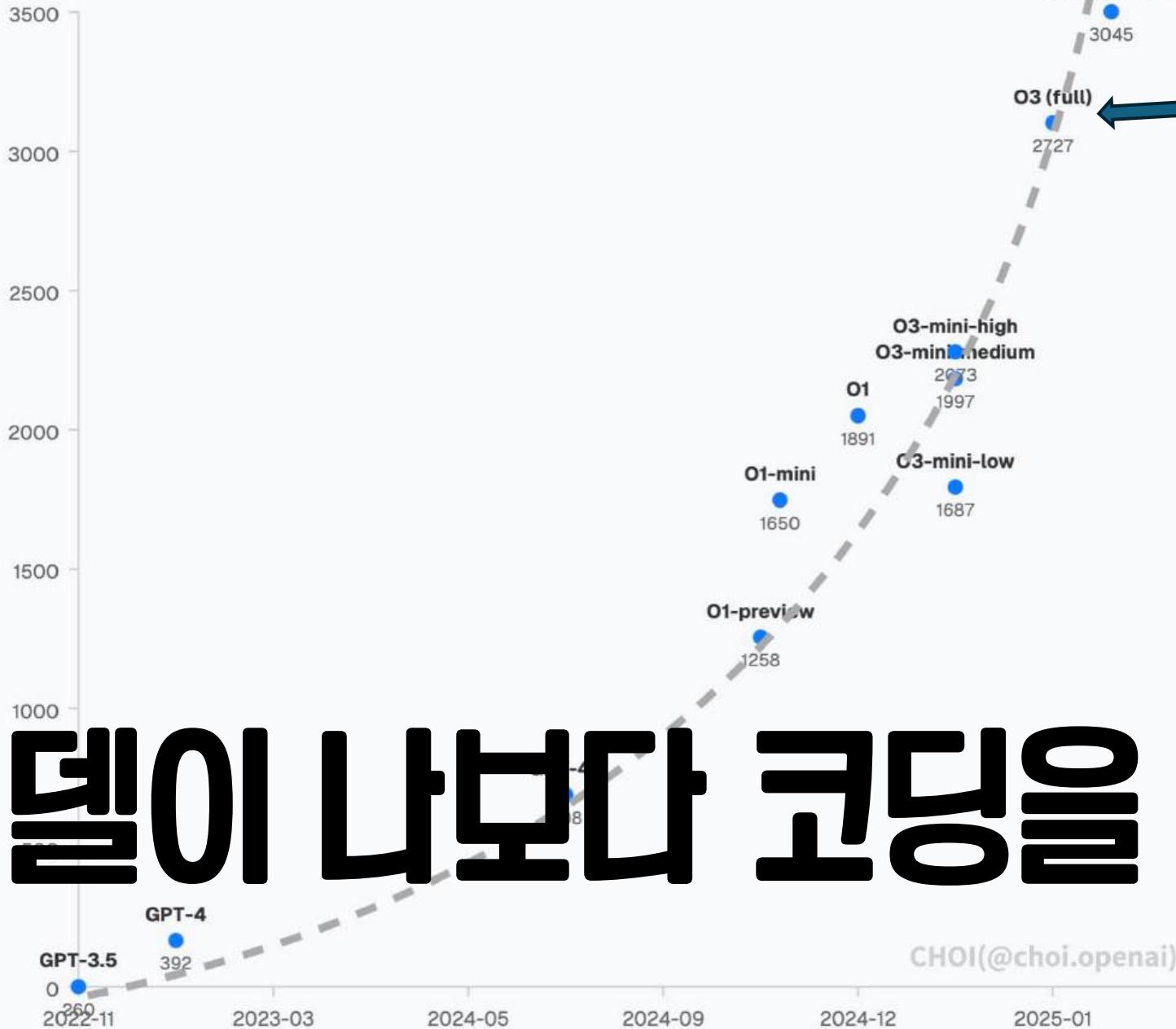
# 데이터 사이언티스트가 아니라 백엔드 엔지니어로서도 많은 코드를 질다.

- MMDB(Main Memory DataBase) 메모리기반 고속처리용 DB
- 기업용 검색엔진 and 웹스케일 검색엔진의 인덱서
- Paxos Algorithm을 이용한 zookeeper like 분산 KeyValue DB
- Hadoop같은 KT ucloud용 PB스케일 분산 파일 시스템
- SKT 전체 패킷 분석을 위한 PB스케일 streaming processing
- 뉴욕 소재 스타트업의 글로벌용 실시간 추천엔진 아키텍처링

여러 인생을 살았어도  
쓰쓰로 생각하기에  
나는 아직도 여전히 엔지니어

**근데 그게 끝나가는 것 같다.**

## OpenAI Models Codeforce Elo Ratings



현재 인간계 top 175위와 동등

**하지만 차라리 좋았어요.**



미라클모닝이고 나발이고  
내가 살아있는게 미라클이다

사실 일단  
나이가 드니까  
몸이 예전같지 않아요

일은 커녕  
게임도  
밤새면서 못하는 몸뚱이

내 생산성을 끌어올릴  
외부적 도구를  
극단적으로 써보자.

오늘은 제가 프로그래머로서  
AI를 활용하는 방법에 대해  
나누어 보려 합니다.

**사실 이건 나 말고도, 요즘 많은 기업들의 화두**

**우리 기업이 어떻게  
AI Native회사가 되어 생산성을 끌어올릴 것인가?**

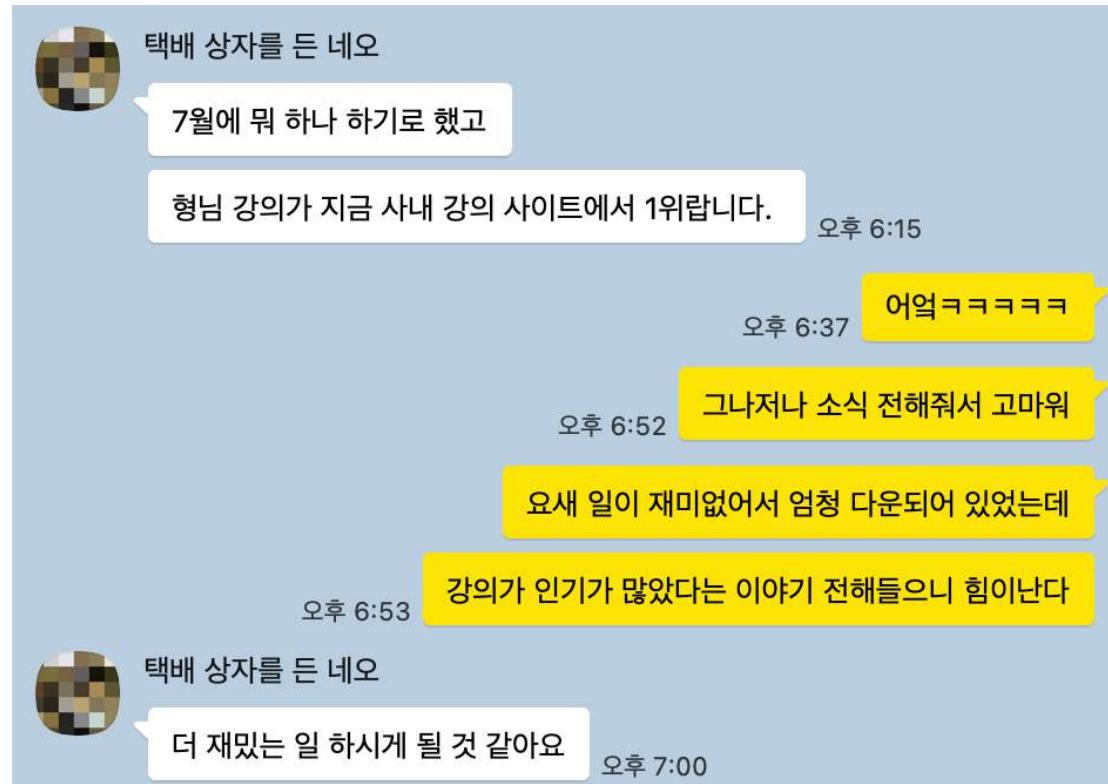
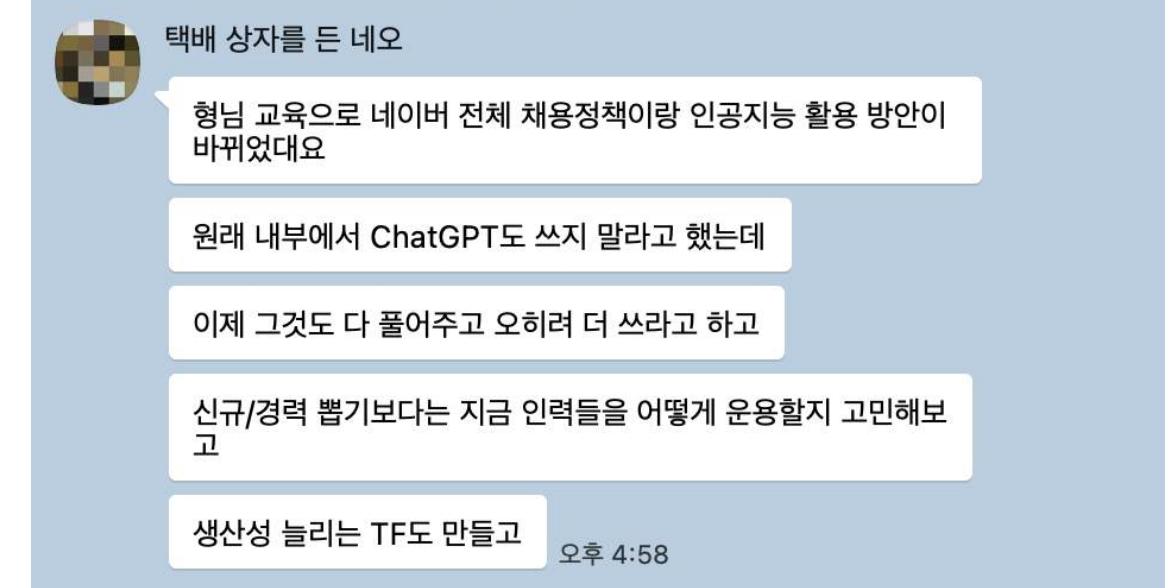
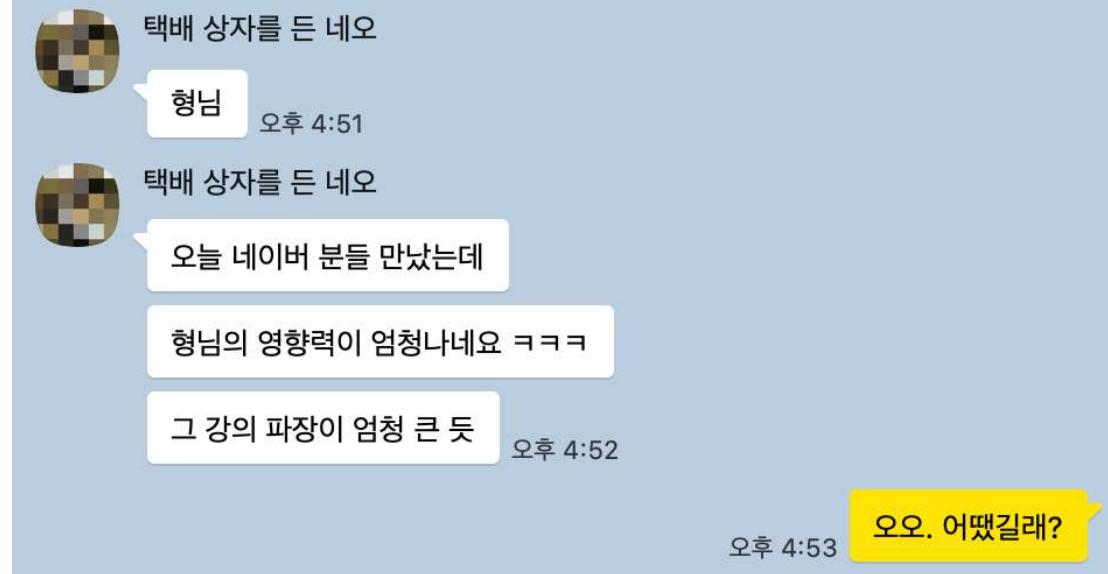
# 굴지의 네이버도 고민중 – 강의를 하고 왔음

AI로 개발을 어떻게 가속화하는가

-드디어 열린 입코딩 시대-

하옹호

- 2025.2 – 네이버 대표님, COO님, 개발 임원, 리더 100명 대상
- 2025.4 – 네이버 개발직군 전직원 대상



오늘 여러분과도 함께 고민해봅시다.

# 요즘 AI관련된 이야기를 하면 느끼는 고로움 두달만 지나도 절반이 바뀐다.

(첫 발표땐 생소한 용어였지만 지금은 모두 이미 익숙해진 바이브코딩 같은 용어 등등)



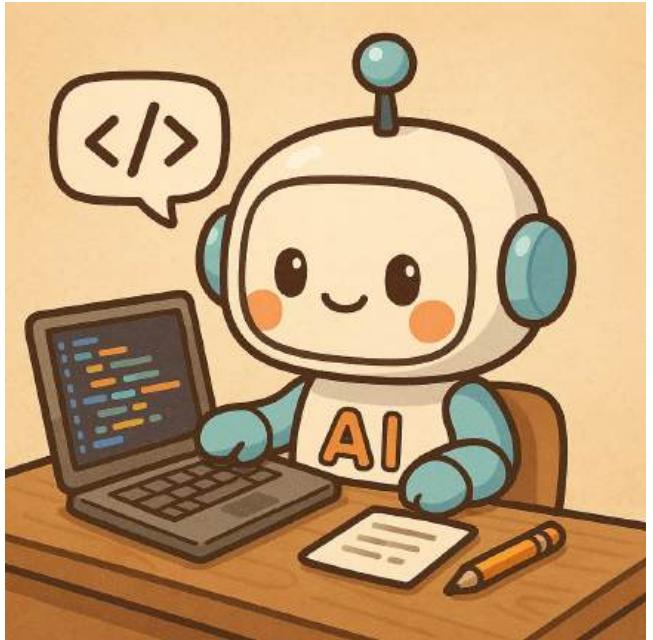
이 자료에도 세부 모델명은 시간이 지나 이미 바뀐 것들(예 o1pro → o3pro) 이 있습니다. 하지만 본질이 변함없기에 도움이 되리라 생각해봅니다.

최신 AI뉴스를 트래킹해서 따라 잡으려고  
너무 많은 시간을 끼기보다는 어느정도 대중화가 된 이후에  
정리된 결과물만 받아서 쓰시는게  
ROI가 있습니다.

‘와 이거 미쳤습니다~’류에 반응안하셔도 됩니다.

오늘은 cursor와 claude code을 커버.  
이 두 가지는 어느정도 대중화되어 스탠다드로 꽤나 갈 것 같습니다.

# AI의 가장 성공적인 케이스 두 가지를 꼽자면?



AI code assistant



지브리 프사

# 하지만 생산성은 코딩에서만 오지 않는다. 개발자는 어떤 일들을 하나?

관련 분야 리서치

제품스펙논의 (회의..회의..회의)

제품 아키텍처 설계 (회의..회의..회의)

실제 구현 (우리가 코딩이라 부르는 것)

QA&디버깅

모니터링, 운영 & 장애대응 (사람들이 잘 모르지만 막대한 시간을 잡아먹음)

다른 사람 코드 리뷰

추노하는 직원 보충할 신규 개발자 면접

코딩은 일부분

모든 부분에서 AI가 사용되어야 회사 전체가 빨라짐

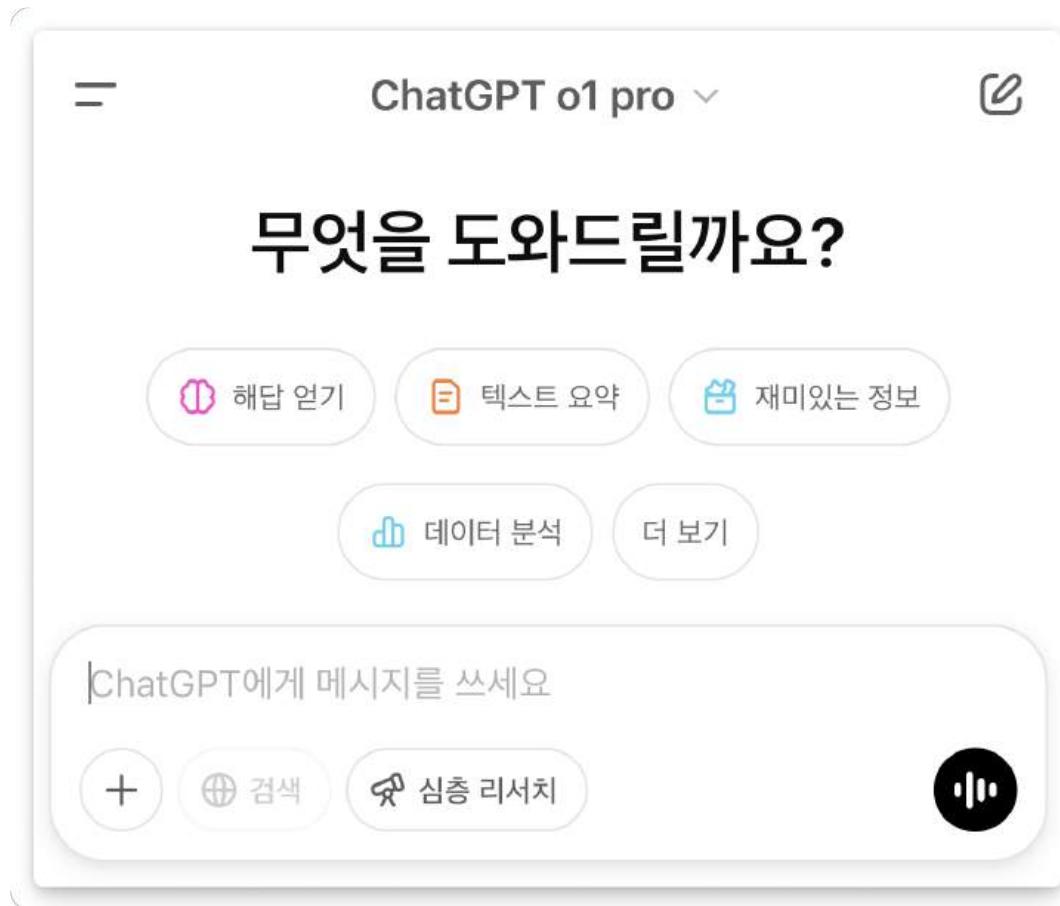
**리서치 : 고민을 아웃소싱해 봅니다.**

# 최근 이런 문제를 풀어야 했습니다.

- 우리가 가진 딥러닝 모델이,
- 개별 유저의 방문마다 최적의 광고는 잘 찾고 있다.
- 순간순간의 local optimal은 잘 찾아내고 있지만
- 어른의 사정을 포함한,
- 광고주들 간의 예산 배분이나, 수익전체에 대한 global optimal
- 이건 잘 찾아내고 있나?

# 매우긴 배웠는데..

- 회사는 유저의 클릭의 갯수를 최대로 늘려야 매출이 늘지만 (CTR)
- 광고주들은 전환이 잘되어야 오래 남아 계십니다. (CVR)
- 하지만 둘은 약간은 대립관계에 있습니다.
  - 클릭수를 무턱대고 늘리면, 클릭대비 전환비율은 떨어짐. 광고주 떠남
    - 더불어 전환이 잘되는 좋은 기회는 한정적이고, 광고주별로 잘 나누어야 함
- 이런 제약사항 아래서 최적점을 찾는거, 학교에서 배웠는데..
  - 후후.. 난 참 좋은 학생이었지. 교수님 죄송합니다.
- GPT님! 도와주세요!



**무려 200달러짜리를 구독하고 있는 플렉스인생**  
이야.. 내가 이거 사려고 엑싯했구나. 어우 비싸



, 600000, 35000, 110

위의 데이터를 기반으로 아래의 작업을 google ortools로 풀려고 한다.

정의가 엄밀하지 못한 부분을 지적하고, 채워줘  
문제 정의는 아래와 같다.

지금 풀어놓은 솔루션이 문제가 있다고 생각되어.

글로벌 상수 T이 있다.

- 개별 광고주(client\_id)는 1개의 총 예산(budget)을 가진다.
- 그 광고주(client\_id)의 각 지면(service\_id)에 자신의 트래픽을 배분할 수 있다.
- 각 광고주는 각 서비스에 노출되며, 해당 pair(client\_id, service\_id)로 각각 ctr과 cvr을 가진다.
- 때로 pair(client\_id, service\_id)의 값이 없는 경우, 해당 service의 ctr와 cvr을 대체하여 fill 해 쓸 수 있다.
- 개별 client가 각 서비스에 자신의 트래픽을 배분한다.
- 트래픽 비율 \* ctr \* T = 클릭수
- 클릭수 \* avg\_cpc가 총 비용
- 총 비용은 총 예산\*1.1을 넘을 수 없다.
- 총 클릭이 발생할 때, 총 클릭 \* cvr 만큼의 전환수 (conversion)가 생긴다.
- conversion이 생기면, 총 집행된 예산과 계산하여 cpa를 계산할 수 있다. 'cpa = 총 사용 예산/conversion수'
- 이 cpa는 tcpa의 2배 이하로 유지되어야 한다.



### 문제 정의와 개선점

이성적으로 판단함을 1m 42s 동안

아래 내용은 질문에서 주어진 문제정의를 좀 더 '수학적 모델' 형태로 엄밀하게 다듬고, 누락되거나 모호한 부분을 지적하고 채워 넣은 예시입니다. Google OR-Tools(예: CP-SAT이나 Linear Solver)로 풀려면 보통 "결정변수(Decision Variables)", "제약조건(Constraints)", "목적함수(Objective)"가 명확해야 합니다. 질문에서 주어진 조건 중, 실제 구현 시 애매모호하거나 누락된 점들이 있으므로, 이를 지적하고, 어떻게 보완할 수 있는지 예시를 들어 설명합니다.

## 1. 문제에서 다루는 데이터와 용어

### 1. 클라이언트 C

**C** = {1, 2, 3, 4}로 가정. 각 클라이언트 *c*는 다음의 상수를 가진다.

- $Budget_c$ : 총 예산(질문 예시에서는 500000, 300000, 400000, 600000 등)
- $TargetCPA_c$ : 타겟 CPA(질문 예시에서는 30000, 20000, 25000, 35000 등)
- $avgCPC_c$ : 평균 CPC(질문 예시에서는 100, 150, 120, 110 등)

### 2. 서비스(혹은 지면) S

**S** = {101, 102, 103, 104, 105, ...}로 가정.

- 일부 서비스는 'good', 'mediocre', 'bad' 같은 카테고리(quality)가 있지만, 실제 최적화 모델에는 직접적으로 'good', 'bad' 같은 라벨이 들어가기보다는, CTR·CVR 값으로만 반영할 가능성이 큼.

### 3. CTR, CVR 정보

- $serviceCTR_s$ ,  $serviceCVR_s$  : 서비스 자체가 평균적으로 가지고 있는 CTR, CVR (예: 0.05, 0.01

## 7. 정리된 모형 예시

아래는 “전환 최대화”를 목표로 하는, 선형(Linear) 형태의 모형 예시입니다.

(단, CPA 제약을 선형으로 처리하기 위해서는 전환  $\neq 0$ 이 보장되어야 한다는 점에 유의)

### 변수

$$x_{c,s} \geq 0, \quad \forall c \in \mathbf{C}, s \in \mathbf{S}.$$

### 파라미터 전처리

$$\text{CTR}_{c,s} = \begin{cases} \text{clientServiceCTR}_{c,s}, & \text{존재하면} \\ \text{serviceCTR}_s, & \text{없으면} \end{cases}; \quad \text{CVR}_{c,s} = \begin{cases} \text{clientServiceCVR}_{c,s}, & \text{존재하면} \\ \text{serviceCVR}_s, & \text{없으면} \end{cases}$$

$$a_{c,s} = \text{CTR}_{c,s} \times \text{avgCPC}_c, \quad b_{c,s} = \text{CTR}_{c,s} \times \text{CVR}_{c,s}.$$

### 목적함수 (전환 최대화)

$$\max \sum_{c \in \mathbf{C}} \sum_{s \in \mathbf{S}} (x_{c,s} \times b_{c,s}).$$

### 제약조건

- 글로벌 노출 한도 (필요 시)

$$\sum_{c,s} x_{c,s} = T.$$

또는

$$\sum_{c,s} x_{c,s} \leq T.$$

- 예산 110% 이내

$$\sum_{c,s} (a_{c,s} \times x_{c,s}) \leq 1.1 \times \text{Budget}_c, \quad \forall c.$$

# 뭔가 멋지게 정리해준다!

- 전환이 0이면 사실상 위 식이 “ $\text{cost} \leq 0$ ”가 되어버림 → 즉, 전환 0인 경우는 허가 되지 않음.
- 문제 세팅에 따라 “전환 0이면 CPA 제약은 무효”로 하고 싶은 경우, 추가적인 big-M 또는 piecewise 기법 필요.

- (옵션) 각 클라이언트가 전환 0이 되지 않도록

$$\sum_s b_{c,s} x_{c,s} \geq \epsilon > 0, \quad \forall c.$$

( $\epsilon$ 은 최소 전환량 등의 임의의 작은 값)



o1 pro mode에 요청

추론 완료

아래 요소를 고려해줘

- 개별 service마다, client의 avg cpc가 다를 수 있어. (같을 수도 있지만, 통상 다르다고 설정)
- T가 모호하면 제거해줘.  $X_{c,s}$ 를 그대로 트래픽(노출 수)로 정의해줘

목적함수는

- 가장 많은 수의 클라이언트가 tcpa 2배 이하 조건을 달성하고,
- 그와중에 모든 클라이언트, 지면의 모든 클릭수 \* cpc를 최대화 하는거야.

이를 기반으로 다시 전체 내용을 정리하고,

이를 python ortools 로도 표현해줘

아래에서는 질문에서 주어진 요구 사항을 좀 더 정형화하고, 그에 맞춰 \*\*Google OR-Tools(Python)\*\*로 모델을 구성하는 예시를 제시합니다.

본 예시는 “Multi-Objective(다중목표)”에 가까운 형태를, OR-Tools에서 흔히 쓰는 ‘가중합’을 이용하여 단일 목적함수로 표현하는 방식으로 구현합니다.

## 1. 문제 요구사항 정리

### 1. 결정변수

- $x_{c,s} \geq 0$ : “클라이언트  $c$ 가 서비스(지면)  $s$ 에 할당하는 트래픽(노출) 수”
- $z_c \in \{0, 1\}$ : “클라이언트  $c$ 가  $\text{CPA} \leq 2 \times \text{TargetCPA}_c$ 를 만족했는지 여부”
  - $z_c = 1$ 이면 ‘해당 클라이언트의 CPA 제약을 달성한다’
  - $z_c = 0$ 이면 ‘달성 못 함 (해당 제약 relax)’

### 2. 파라미터(상수)들

- $\text{Budget}_c$ : 클라이언트  $c$ 의 예산
- $\text{TargetCPA}_c$ : 클라이언트  $c$ 의 목표 CPA
- $\text{CPC}_{c,s}$ : 클라이언트  $c$ 가 서비스  $s$ 에 집행할 때의 평균 CPC (개별 service마다 다를 수 있음)
- $\text{CTR}_{c,s}, \text{CVR}_{c,s}$ : 클라이언트  $c$  - 서비스  $s$  페어의 CTR, CVR
  - 질문에서 “존재하지 않으면 서비스 단위 CTR, CVR로 대체” → 전처리로  $\text{CTR}_{c,s} \cdot \text{CVR}_{c,s}$ 를 채워넣



이미 많은 사람들의 간증이 있었지만  
복잡한 주제에 대한 조사와 고민에서

ChatGPT Pro + Deep Research는 신입니다.

o3 or o4 (pro)는 대화식으로 물기보다, 엄청 거대한 길이의 질문으로 1샷으로 물을 때 잘합니다.  
거대한 질문을 사람이 만드는 건 힘드니까, gpt 4o등으로 대화식으로 핑퐁하며 만듭니다.  
상세한 스펙이 들어간 긴 질문이 완성되면, 이걸 o3 등의 test-time-compute 모델에게!  
간단한 건 gemini pro를 꽤 쓰지만, o3 pro - 200달러짜리만이 주는 가치가 있기는 함

# 어떻게 하면 거대한 프롬프트를 만들까

- 지금은 이미 다들 잘 아는 내용이지만,
  - 가장 고급 모델들은 ‘거대한 프롬프트’를 넣어야 제대로 대답
- 할 때마다 ‘우리 회사의 자세한 도메인 지식’?
  - 매번 타자칠 수 없다.
  - 일단 손도 아프고 기억도 안난다. 첨부하고 싶다.
- **수시로 사용할 “회사 도메인 지식 베이스.md” 텍스트**
  - 회사 내에서 manage 되고 있는가?
  - 시간이 지나면서 상하지 않고, 계속 최신으로 관리되고 있는가?
- 제일 쉬운 방법은 사내 notion을 MCP로 LLM과 연결해두는 것
  - 대답할 때마다 notion의 노트들을 참조해서 대답해줌
  - 다만 아직 chatgpt는 (앱바인딩은 되지만) notion MCP와 아직 연결이 안됨
  - 그럼에도 도메인지식이 정리된 싱글페이지 문서는 추천함

**코딩 : 그럼 이제 코딩을 해보자.**



저번 네이버 발표할 당시만 해도  
저는 cursor가 메인이었는데  
요즘은 90%는 claude code로 합니다.

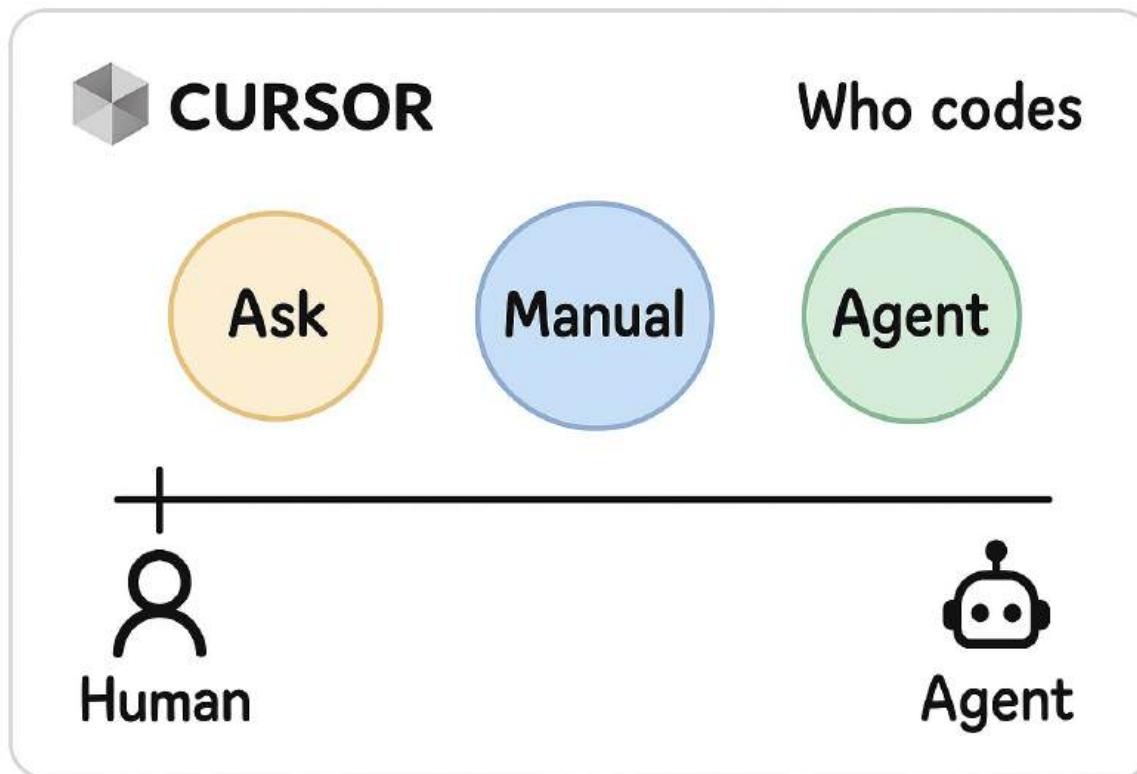
# 왜 이렇게 바꼈었나?

- 두 제품의 성격 차이
  - Cursor - 좋은 도구를 줘어주는 느낌. 일하는 건 여전히 나
  - Claude code - 좋은 주니어를 붙여주는 느낌. 이들을 부리는 나
- 각 제품의 자율성 레벨의 차이
  - Cursor의 에이전트 모드도 꽤나 많이 자율적이었으나
  - Claude code의 에이전트 레벨이 훨씬 더 높음
- 두 제품의 프라이싱의 차이
  - Cursor는 휴대폰같은 요금제. 개발자 1명당 1커서 (거기다 최근 비싸짐)
  - Claude code는 인터넷회선 같음. 너의 capa안에서는 얼마든지 써라.

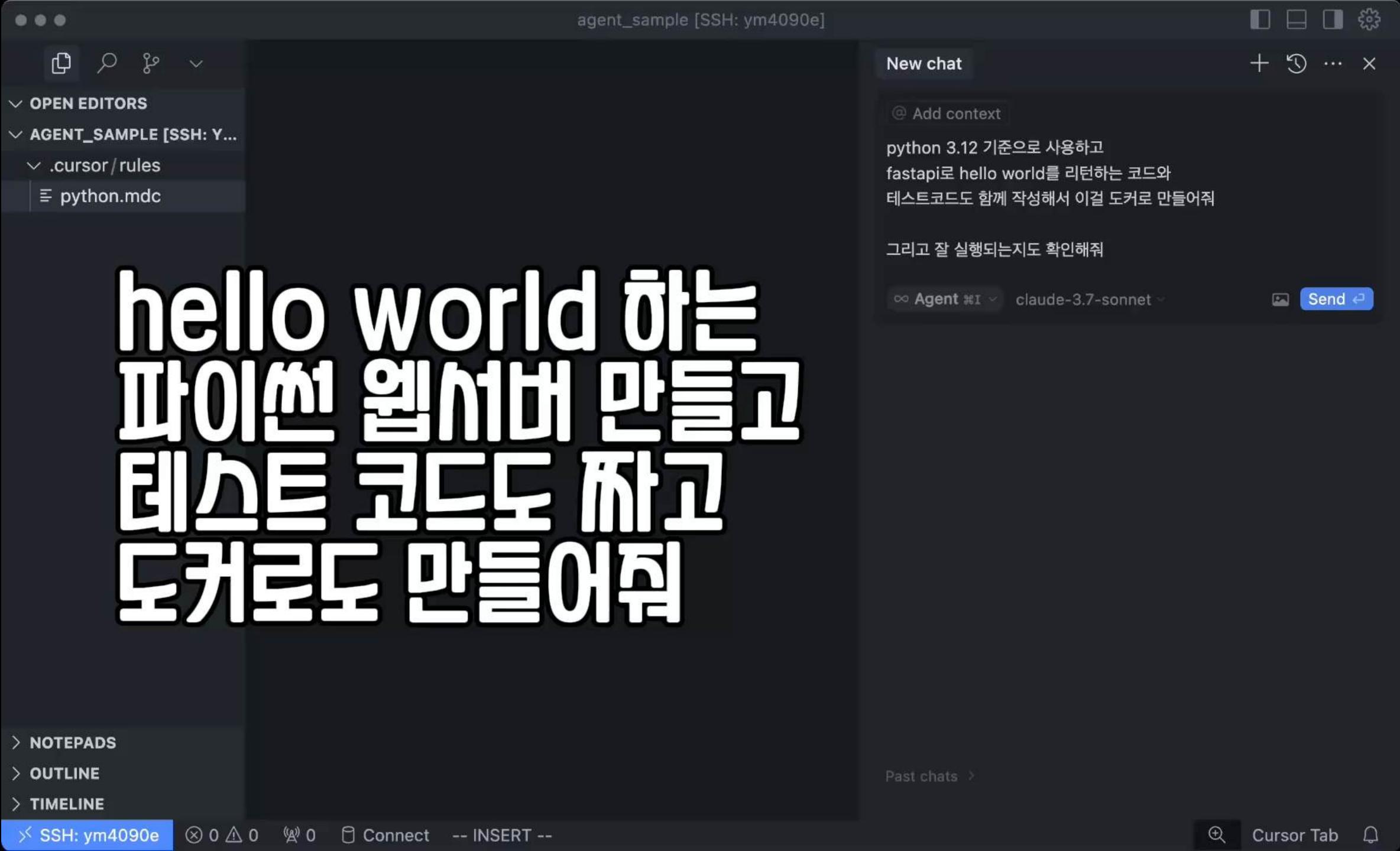
# 그래도 커서도 일단 보고 가시죠



# 초기 버전의 cursor와 달리 이제는 agent가 제철입니다.



Cursor의 동작 모드들



hello world 하는  
파이썬 웹서버 만들고  
테스트 코드도 짜고  
도커로도 만들어줘

테스트 코드 작성

프로젝트 환경설정하고  
작업하는데 필요한 패키지  
스스로 설치

코드 작성중

```
main.py - agent_sample [SSH: ym4090e]
main.py > test_main.py U
def main():
    print("Hello from agent-sample!")
    from fastapi import FastAPI
    app = FastAPI()
    @app.get("/")
    async def read_root():
        return {"message": "Hello World"}
```

```
main.py - agent_sample [SSH: ym4090e]
main.py +10 -1
def main():
    print("Hello from agent-sample!")
    from fastapi import FastAPI
    app = FastAPI()
    @app.get("/")
    async def read_root():
        return {"message": "Hello World"}
```

테스트 패스 확인

풀오토!

```
main.py - agent_sample [SSH: ym4090e]
main.py > README.md U
def main():
    print("Hello from agent-sample!")
    from fastapi import FastAPI
    app = FastAPI()
    @app.get("/")
    async def read_root():
        return {"message": "Hello World"}
```

```
main.py - agent_sample [SSH: ym4090e]
main.py > README.md U
def main():
    print("Hello from agent-sample!")
    from fastapi import FastAPI
    app = FastAPI()
    @app.get("/")
    async def read_root():
        return {"message": "Hello World"}
```

도커 빌드하고  
잘 되었는지 자체 확인

포트가 충돌나니까  
알아서 다른 포트를 통해  
재실행하고 테스트

도커 파일 작성

```
main.py - agent_sample [SSH: ym4090e]
main.py > Dockerfile U
def main():
    print("Hello from agent-sample!")
    from fastapi import FastAPI
    app = FastAPI()
    @app.get("/")
    async def read_root():
        return {"message": "Hello World"}
```

```
main.py - agent_sample [SSH: ym4090e]
main.py > Dockerfile U
def main():
    print("Hello from agent-sample!")
    from fastapi import FastAPI
    app = FastAPI()
    @app.get("/")
    async def read_root():
        return {"message": "Hello World"}
```

컨테이너 정리까지 마치고  
깔끔한 마무리!

# 예제로 무언가 만들면서 살펴보자.

- 하지만 어떤 예제가 중요한가.
- 내 삶에 와닿는 소재
- 회사 일인가? 정말?
- 주식! 주식을 하자
- 미국 주식을 해야한다.
- 무슨 종목을 사야하나?
- ETF말고 익사이팅하게. 개별종목으로.

# 뜻밖에 많은 분이 이런 접근을 하심

ChatGPT 4o ▾



무엇을 도와드릴까요?

부자가 되는 파이썬 프로그램 짜주세요



“신의 뜻대로”

= 인생의 코딩법

= 가챠식 코딩

= 기도 메타



# 어떻게 시작하죠?

- 나중에 cursor등에게 시키려면 지시가 명확하고 자세해야.
- 하지만 지시를 전부 다 작성해 넣기엔 내가 체력이 없다.
- 목표를 세분화 하는 일도 LLM에게 맡기자.
  - 생각도 LLM에게 맡기자.



‘나는 이런 걸 만들거야. 제품요구사항(PRD)’ 만들어줘”

‘나는 이런 PRD를 가지고 있어. 세부작업리스트(taskList) 원해”

# PRD 제작은 상대적으로 장문으로 작성해주는 o3나o4 으로

## 무엇을 도와드릴까요?

나는 주식 투자를 도와주는 프로그램을 만들고 싶어



나는 미국 주식들 중에서 nasdaq 의 시총 100위 안에 드는 주식들 중에서,

최근에 상승세인 주식들을 골라서 보여주는 나를 위한 서비스를 만들고 싶어.

최근에 상승세인 것은 중요하지만, 또 너무 급등해서 위험한 주식은 피하고 싶어.

이런 프로그램을 만들기 위해서 검토해야할 추가 요구사항들은 어떤 것들이 있을까?

논리적으로 보충할점이나, 고려해야할 점을 추가해서 PRD를 상세하게 작성해줘



아래는 Nasdaq 100(시총 상위 100개 종목) 중에서 최근 상승세를 보이지만 과열 위험이 상대적으로 낮은 주식을 선별하여 보여주는 프로그램(또는 서비스)을 만들기 위한 PRD(제품 요구사항 문서) 예시입니다. 각 세부 항목에서 프로그램 구현 시 고려해야 할 추가사항들을 가능한 한 구체적으로 나열해 보았습니다.

## 1. 제품 개요

- 목표: Nasdaq 100 종목 중 최근 일정 기간 동안 상승 추세를 보이는 주식을 찾아내고, 과도한 변동성(혹은 급등)으로 인한 위험이 큰 종목은 필터링하여 사용자에게 제공하는 서비스.
- 주요 기능:
  - 미국 주식 데이터(가격, 거래량, 시가총액, 지표 등) 수집
  - 상승 추세 판단 로직(기간 정의, 상승률 계산 등)
  - 변동성 혹은 과열 판단 로직(지나친 급등 감지 등)
  - 필터링 후 종목 리스트 제공, 정렬 및 시각화

## 2. 사용자 요구사항

- 사용자
- 원하는
- 프로그램
- 급격한

5. 최종 리스트를 사용자에게 보여준다. (차트, 지표, 종목별 요약 정보 등)

# PRD문서가 길고 풍부한 맥락을 가지고 만들어집니다. 이상하면 대화하며 revision!

## 3. 기능 요구사항

### 3.1 데이터 수집

- 데이터 소스: 주식 시세, 거래량, 뉴스, 재무제표 등. 예: 티커별 OHLCV(시가, 고가, 저가, 종가, 거래량) 데이터를 제공하는 API(예: Yahoo Finance, Alpha Vantage, 또는 유료 금융 데이터 공급사).
- 데이터 범위: 최소 1년 이상의 과거 시세 데이터(트렌드 분석과 변동성 판단에 활용).
- 업데이트 빈도: 실시간(실시간 또는 15분 지연), 일 단위, 주 단위 등 사용자 니즈에 따라 결정.

### 3.2 데이터 처리 및 보정

- 결측치 처리: API 자연이나 서버 장애로 인해 누락된 데이터가 있을 경우, 보간(interpolation) 또는 다른 API 소스 활용.
- 이상치 제거: 극단값(비정상적인 거래량, 가격 스파이크 등)을 제거 또는 재확인 로직 마련.
- 종목 리스트 동기화: Nasdaq 100 종목은 시가총액 변동에 따라 변경될 수 있으므로, Nasdaq 100에 속하는 종목의 티커 리스트를 주기적으로 업데이트.

### 3.3 상승 추세 판단 로직

- 결측치 처리: API 자연이나 서버 장애로 인해 누락된 데이터가 있을 경우, 보간(interpolation) 또는 다른 API 소스 활용.
- 이상치 제거: 극단값(비정상적인 거래량, 가격 스파이크 등)을 제거 또는 재확인 로직 마련.
- 종목 리스트 동기화: Nasdaq 100 종목은 시가총액 변동에 따라 변경될 수 있으므로, Nasdaq 100에 속하는 종목의 티커 리스트를 주기적으로 업데이트.

### 3.4 과열(급등) 위험 판단 로직

- 분석 주기 설정: 최근 1주/1개월/3개월/6개월 등 원하는 기간으로 설정 가능하도록 옵션화.
  - 상승률 계산: 기간 내 총가 변동률, 이동평균선의 변동, 또는 RSI와 같은 기술적 지표 활용.
    - 예: 20일 이동평균선이 60일 이동평균선보다 위에 존재하고, 최근 5일간 가격이 지속적으로 상승.
  - 추세 강도 지표: MACD, RSI, Stochastic 등 복합 지표 활용 여부 검토.
  - 최소 거래량 기준: 상승 추세가 미미한 거래량 속에서 나타난다면 정확도 저하 가능, 거래량 조건(평균 거래량 대비 몇 % 이상)을 고려.
- 단기 급등률: 특정 기간(예: 3일/5일/1주 등) 동안 비정상적으로 높은 상승률을 기록한 종목은 제외 또는 경고 표시.
    - 예: 5일간 15% 이상 급등하는 등 특정 임계값 설정.

### 3.5 알림 기능(옵션)

- 알림 기능(옵션): 원하는 임계값(상승률, 변동성 등)에 도달 시, 자동 푸시 알림 또는 이메일 전송.

## 4. 보완 고려사항

### 4.1 리스크 관리

- 백테스팅: 적용한 전략(상승 추세, 과열 지표 기준)이 과거 데이터에서 어느 정도 성과를 냈는지 백테스트 도구 및 리포트 제공.
- 분산 투자 제안(옵션): 특정 섹터나 종목에 몰릴 위험이 있으므로, 산업별 분산 또는 시장 상황 분석 기능 추가.
- 리스크 지표 추가: 베타(시장과의 연관성), 샤프 지수 등 투자 리스크 측정 지표를 통해 "위험 대비 수익률"을 고려 할 수도 있음.

### 4.2 사용자 맞춤형 설정

- 투자 성향: 보수적/중립적/공격적 투자 성향에 따라 과열 판단 로직이나 변동성 한도를 달리 적용.
- 커스텀 지표: 사용자 선호 지표(RSI, MACD, 이동평균선 기간 등)를 선택할 수 있도록 UI 제공.
- 감정 분석(옵션): 소셜 미디어, 뉴스 헤드라인 등을 분석해 시장 심리를 반영할 수도 있음.

### 4.3 성능 및 확장성

- API 키 및 개인정보 보호: 외부 데이터 소스 API 키 노출 방지, 사용자 계정 정보 암호화 저장.
- 규제 준수: 투자 자문이나 리서치 기능을 제공할 경우, 현지 금융 규제나 라이선스 문제 검토 필요.

### 4.5 유지보수 및 모니터링

- 오류 감지: 데이터 수집 실패, 로직 오류 등 발생 시 장애 알림을 받아 신속 대응할 수 있는 모니터링 시스템.
- 업데이트 로드맵: 지표 업데이트, 새로운 기능(예: AI 기반 예측 모델) 추가 계획 등을 문서화.

## 5. 기술 스택(예시)

- 프론트엔드: React/Angular/Vue.js 등(차트 시각화 라이브러리: D3.js, Highcharts 등).
- 백엔드: Python(Flask/FastAPI/Django), Node.js, Java 등 중 택 1.
- 데이터베이스: PostgreSQL/MySQL(관계형 DB), MongoDB 등(비정형 데이터 필요 시).
- 분석/알고리즘: Python의 Pandas, NumPy, scikit-learn, TA-Lib 등.
- 배포: AWS, GCP, Azure 등 클라우드 인프라(도커 컨테이너 활용 가능).

# 나중에 LLM에게 시킬것을 고려한, 상세한 Tasklist 를 요청한다.

ChatGPT o1 ▾

↑ 공유하기

위 PRD는 프로그램 구조를 체계적으로 잡는 동시에, 향후 유지보수와 고도화(예: AI 기반 예측, 소셜 미디어 감정 분석 연동 등)의 여지를 남겨두도록 작성되었습니다. 추가로 구현 후 실제 투자 시뮬레이션(백테스팅) 및 다양한 사용자 피드백을 통해 지표와 필터링 로직을 꾸준히 개선하는 것이 중요합니다.

프로그래머에게 지시를 내려서 1주일안에 만들 수 있는 정도로 스펙을 구체화 해줘.

여러 대안들이 있는 요구사항은 먼저 결과를 볼 수 있고, 구조를 가볍게 가져갈 수 있는 것으로 선택해줘.

일단 결과를 먼저 보는 것이 우선이고, 그 이외의 것들은 우선순위를 낮춰줘  
각종 라이브러리등은 가장 대중적인 것들로 골라줘.

나는 구현언어로 python을 쓸것이고, 패키지 관리자는 pip대신 uv를 쓸거야.  
최종 결과물은 Docker 이미지로 빌드될 거야.

주니어 개발자들이 일을 뜯어가서 하나씩 구현해도 할 수 있을 정도로  
상세한 task list로 만들어줘

- **기간을 명시하면**  
날짜별 todo를 줘서,  
나중에 잘라서 명령내리기 편함
- 내가 명확히 원하는  
라이브러리가 있는게 아니라면
  - 대중적인 라이브러리를 쓰도록 권고
  - 그래야 LLM이 train과정에서  
많이 학습한 것을 채택해서 잘 코딩
- **수준을 주니어 개발자 수준으로**  
지시하면 상세한 명령을 줌

ChatGPT 01 · 공유하기

이제는 특히 Nasdaq 100 티커를 인터넷에서 동적으로 가져오고, 2개월 치 데이터를 분석하고, 과열 종목을 제외하고, Streamlit을 통해 웹 UI를 구현하는 부분을 번역했습니다.

## 1. 개요

- 목표**
  - 인터넷에서 Nasdaq 100 종목 티커를 동적으로 가져옴.
  - 최근 2개월(약 60일) 치 주가 데이터를 수집해.
  - 상승 추세를 보이면서도 급등(과열) 위험이 낮은 종목을 필터링하여.
  - Streamlit 웹 UI로 결과를 확인할 수 있는 MVP를 만든다.
- 기술 스택**
  - 언어: Python
  - 패키지 관리자: uv
  - Deploy: Docker 이미지 빌드
  - 데이터 수집 라이브러리: yfinance
  - 웹 UI: streamlit
  - 데이터 처리: pandas, numpy
  - HTTP 요청(티커 리스트 스크래핑 필요 시): requests + BeautifulSoup (선택)
- 핵심 기능**
  - Nasdaq 100 티커 목록 가져오기
  - 2개월간의 주가 데이터 수집
  - 상승 추세 및 과열(금지) 필터링
  - 과열(금지) 종목 제외
  - Streamlit으로 결과 출력
  - Docker로 배포

## Day 2: Nasdaq 100 티커 리스트 동적 수집

- 단순 접근:**
  - 만약 간단한 라이브러리가 있다면(nasdaq100list 같은 것) 사용 가능.
  - 라이브러리가 없다면, 공식 Nasdaq 사이트나 다른 신뢰 가능한 사이트에서 HTML을 파싱해 티커를 추출. 예:

```
python
import requests
from bs4 import BeautifulSoup

def get_nasdaq100_tickers():
    url = "https://www.nasdaq.com/market-activity/quotes/nasdaq-idx-index"
    res = requests.get(url)
    soup = BeautifulSoup(res.text, 'html.parser')
    # soup 파일 경로로 티커 리스트 추출
    return ticker_list
```

  - 파싱 구조가 변동될 수 있으므로, 간단히 테스트하며 동작상 확인.
- Fallback:**
  - 사이트 구조가 자주 바뀌면 일시적으로 수동 리스트(JSON/CSV 등)를 쓰는 방법도 고려.
- 함수 구현 & 테스트**
  - src/data\_loader.py에 get\_nasdaq100\_tickers() 함수 구현.
  - 간단히 print해서 100개 티커가 잘 추출되는지 확인.

## Day 3: 상승 추세 & 과열 판별 로직 작성

- 단위 테스트:**
  - 종목마다 데이터가 없거나 API 호출 실패 시 None 혹은 빈 데이터프레임을 리턴.
- 단위 테스트:**
  - 실제 티커 및 개별 품목을 템아보아, 데이터 프레임 길이가 40~50행 이상 나오는지 확인(시장 휴장일 등 감안).

## Day 4: 과열(금지) 필터 (예시)

- 과열(금지) 필터 (예시):**

```
python
def calc_growth_rate(df, days=20):
    if len(df) < days:
        return None
    initial_price = df['Close'].iloc[-days]
    final_price = df['Close'].iloc[-1]
    growth = (final_price - initial_price) / initial_price * 100
    return growth

def is_overheated(df, days=5, limit=15):
    if len(df) < days:
        return False
    old_price = df['Close'].iloc[-days]
    new_price = df['Close'].iloc[-1]
    growth = (new_price - old_price) / old_price * 100
    return growth >= limit
```

## Day 5: 메인 로직 (파이프라인) 작성

- src/main.py:**
  - (1) get\_nasdaq100\_tickers()로 티커 리스트 획득
  - (2) 각 티커별 2개월 치 데이터 수집
  - (3) 상승 추세 / 과열 필터 통과 여부를 확인
  - (4) 결과 리스트(종목, 상승률, 등)를 담아서 리턴

```
python
from data_loader import get_nasdaq100_tickers, fetch_stock_data
from analysis import is_up_trend, is_overheated, calc_growth_rate
```

## Day 6: Streamlit UI 구축

- app\_streamlit.py 작성:**

```
python
import streamlit as st
from src.main import main_pipeline

def main():
    st.title("Nasdaq 100 Uptrend (Not Overheated) Stocks")
    st.write(main_pipeline())
```

## Day 7: 도커라이징 & QA

- Dockerfile (예시):**

```
FROM python:3.9-slim
WORKDIR /app
COPY . /app
```
- QA:**
  - 브라우저에서 http://localhost:8501 접속 후 확인 확인.
  - 실제로 티커 리스트가 잘 뜨는지, 필터 조건이 동작하는지(너무 적거나 너무 많은 결과가 나오지는 않는지) 확인.
  - 로그와 여러 처리 확인.

# 일간 할일로 정리된 상세한 task list 가 만들어 졌습니다.

## 2. 상세 Task List

### Day 1: 프로젝트 초기 세팅

- 리포지토리 생성
  - Git 저장소 초기화, .gitignore (Python, Docker 관련) 설정.
- Python 프로젝트 구조
  - 폴더 예시:

```
project/
  |- src/
  |   |- data_loader.py
  |   |- analysis.py
  |   |- main.py
  |- app_streamlit.py
  |- Dockerfile
  |- uv_dependencies.txt (or equivalent uv config)
  |- requirements.txt (장고용, uv 사용 시 생략 가능)
  |- ...
  
```
- uv 초기 설정
  - uv init (또는 해당 CLI) → 의존성 목록에 필요한 라이브러리(yfinance, pandas, numpy, streamlit, requests, beautifulsoup4 등) 등록.
- Dockerfile 초안 작성

### Day 2: Nasdaq 100 티커 리스트 동적 수집

- 단순 접근:
  - 만약 간단한 라이브러리가 있다면(nasdaq100list 같은 것) 사용 가능.
  - 라이브러리가 없다면, 공식 Nasdaq 사이트나 다른 신뢰 가능한 사이트에서 HTML을 파싱해 티커를 추출. 예:

```
python
import requests
from bs4 import BeautifulSoup

def get_nasdaq100_tickers():
    url = "https://www.nasdaq.com/market-activity/quotes/nasdaq-idx-index"
    res = requests.get(url)
    soup = BeautifulSoup(res.text, 'html.parser')
    # soup 파일 경로로 티커 리스트 추출
    return ticker_list
```

  - 파싱 구조가 변동될 수 있으므로, 간단히 테스트하며 동작상 확인.
- Fallback:
  - 사이트 구조가 자주 바뀌면 일시적으로 수동 리스트(JSON/CSV 등)를 쓰는 방법도 고려.
- 함수 구현 & 테스트
  - src/data\_loader.py에 get\_nasdaq100\_tickers() 함수 구현.
  - 간단히 print해서 100개 티커가 잘 추출되는지 확인.

### Day 3: 상승 추세 & 과열 판별 로직 작성

- 단위 테스트:
  - 종목마다 데이터가 없거나 API 호출 실패 시 None 혹은 빈 데이터프레임을 리턴.
- 단위 테스트:
  - 실제 티커 및 개별 품목을 템아보아, 데이터 프레임 길이가 40~50행 이상 나오는지 확인(시장 휴장일 등 감안).

### Day 4: 과열(금지) 필터 (예시)

- 과열(금지) 필터 (예시):
 

```
python
def calc_growth_rate(df, days=20):
    if len(df) < days:
        return None
    initial_price = df['Close'].iloc[-days]
    final_price = df['Close'].iloc[-1]
    growth = (final_price - initial_price) / initial_price * 100
    return growth

def is_overheated(df, days=5, limit=15):
    if len(df) < days:
        return False
    old_price = df['Close'].iloc[-days]
    new_price = df['Close'].iloc[-1]
    growth = (new_price - old_price) / old_price * 100
    return growth >= limit
```

### Day 5: 메인 로직 (파이프라인) 작성

- src/main.py:
  - (1) get\_nasdaq100\_tickers()로 티커 리스트 획득
  - (2) 각 티커별 2개월 치 데이터 수집
  - (3) 상승 추세 / 과열 필터 통과 여부를 확인
  - (4) 결과 리스트(종목, 상승률, 등)를 담아서 리턴

```
python
from data_loader import get_nasdaq100_tickers, fetch_stock_data
from analysis import is_up_trend, is_overheated, calc_growth_rate
```

### Day 6: Streamlit UI 구축

- app\_streamlit.py 작성:
 

```
python
import streamlit as st
from src.main import main_pipeline

def main():
    st.title("Nasdaq 100 Uptrend (Not Overheated) Stocks")
    st.write(main_pipeline())
```

### Day 7: 도커라이징 & QA

- Dockerfile (예시):
 

```
FROM python:3.9-slim
WORKDIR /app
COPY . /app
```
- QA:
  - 브라우저에서 http://localhost:8501 접속 후 확인 확인.
  - 실제로 티커 리스트가 잘 뜨는지, 필터 조건이 동작하는지(너무 적거나 너무 많은 결과가 나오지는 않는지) 확인.
  - 로그와 여러 처리 확인.

## 3. 마무리 & 추가 고려사항

### 1. 사용성

- 필터 파라미터(상승률 기준, 과열 임계치 등)를 Streamlit UI에서 사용자 입력값으로 받을 수도 있음.

### 2. 유지보수

- Nasdaq 100 티커 파싱 로직은 월 구조가 바뀔 때 깨질 수 있으므로, 예리 처리 또는 개성 진작을 고려.

### 3. 향후 확장

- RSI 등 다른 기술적 지표 적용, DB 저장, 배포 테스트 기능, 로그인/사용자별 세팅 관리 등은 2차 개발로.

이어서 Nasdaq 100 티커 동적 수집 → 2개월 데이터 분석 → 과열 제외 → Streamlit UI → Docker 배포를 이어가는 MVP는 7월 안에 완성할 수 있는 형태의 작업 흐름이 정리되었습니다.

각 Task를 여러 개발자가 나누어 맡아도(티커 수집, 데이터 수집, 분석 로직, Streamlit UI, Docker 세팅 등) 업무 분할이 용이하며, 결과물은 빠르게 시연할 수 있을 것입니다.

근데 저 많은 텍스트를  
코딩할 때 일일이 프롬프트 넣어야 하나?

그냥 파일로 저장해서  
프로젝트 끝에 두면 됩니다.

위 PRD는 프로그램 구조를 체계화  
연동 등)의 여지를 남겨두도록 작성  
백을 통해 지표와 필터링 로직을 구



복사

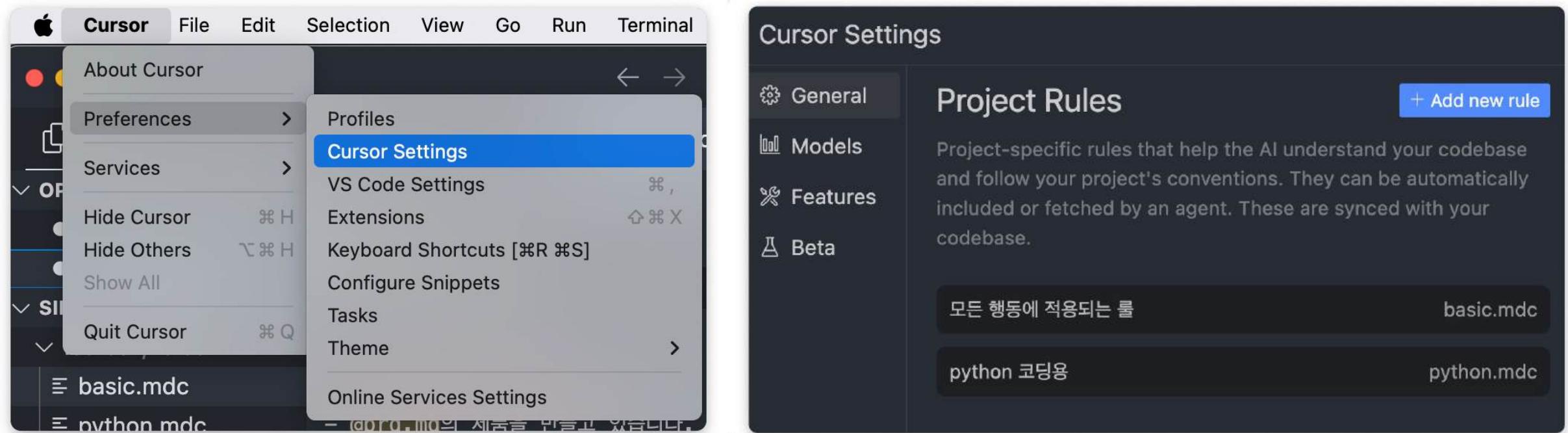
국지말고, 저걸 쓰면  
Markdown으로 복사된다

```
prd.md
1 ## 1. 제품 개요
2
3   tasklist.md
4     1. 개요
5       2. **기술 스택**
6         - 언어: Python
7         - 패키지 관리자: uv
8         - Deploy: Docker 이미지 빌드
9         - 데이터 수집 라이브러리: yfinance
10        - 웹 UI: streamlit
11        - 데이터 처리: pandas, numpy
12        - HTTP 요청(티커 리스트 스크래핑 필요 시): requests + BeautifulSoup (선택)
13
14   3. **핵심 기능/우선순위**
15     1. Nasdaq 100 티커 자동 수집
16     2. 2개월치 일봉 데이터 수집
17     3. 상승 추세 판별 (단순 상승률 or 이동평균 비교)
18     4. 과열(급등) 필터 (임계치 초과 시 제외)
19     5. Streamlit 대시보드에서 종목 리스트 표시
20     6. Docker 빌드 가능하게 마무리
21
22   ## 2. 상세 Task List
23
24   ### Day 1: 프로젝트 초기 세팅
25     1. 리포지토리 생성
```

에디터에 prd.md로 저장  
에디터에 tasklist.md로 저장

# 그리고 cursor가 따를 rule 설정

예전에는 .cursorrules 파일 1개였는데  
이제는 세팅에서 프로젝트 단위로 여러개를 추가할 수 있음



.cursor > rules > ≡ basic.mdc

Description ⓘ :

모든 행동에 적용되는 룰

Globs ⓘ :

\*.tsx, src/config/\*\*/\*.json, \*Test.cpp, ...

Show Preview

### # Most Important Rule

- @prd.md의 제품을 만들고 있습니다.
- 각 작업은 @tasklist.md 의 내용을 참고하여 수행합니다.
- 작업을 할 때, 그에 상응하는 테스트코드도 함께 만듭니다.
- 도커로 이미지를 빌드하고 테스트할 때는, 단지 서버를 띄우는 아니라, 외부의 curl 등에 잘 결과를 리턴하는지도 확인합니다.

모든 행동에서  
prd.md와  
tasklist.md를 참조하라고  
디폴트로 늘 넣어줌  
각자 자신에게 맞게 커스터마이즈

.cursor > rules > ≡ python.mdc

Description ⓘ :

python 코딩용

Globs ⓘ :

\*.tsx, src/config/\*\*/\*.json, \*Test.cpp, ...

Show Preview

### # My rule

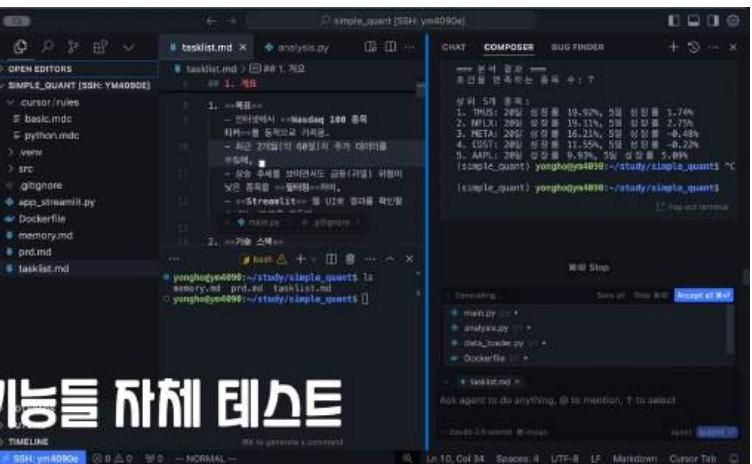
- 패키지 관리는 pip말고 uv를 사용해서, uv로 가상환경 venv를 만들고 거기에서 실행해주세요.
- 테스트는 pytest를 쓰세요.
- 웹이나 REST 작업할 일이 있으면 flask 말고 fastapi를 쓰세요.

예시를 위해 한국어로 쓰긴 했는데  
영어로 썼을 때  
미묘하게 시시를 더 잘 따르는 느낌  
Globs 는 해두면 적용 더 잘됨

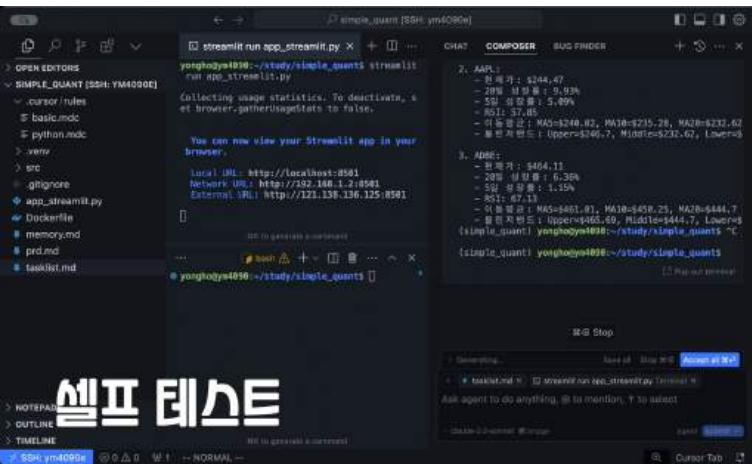




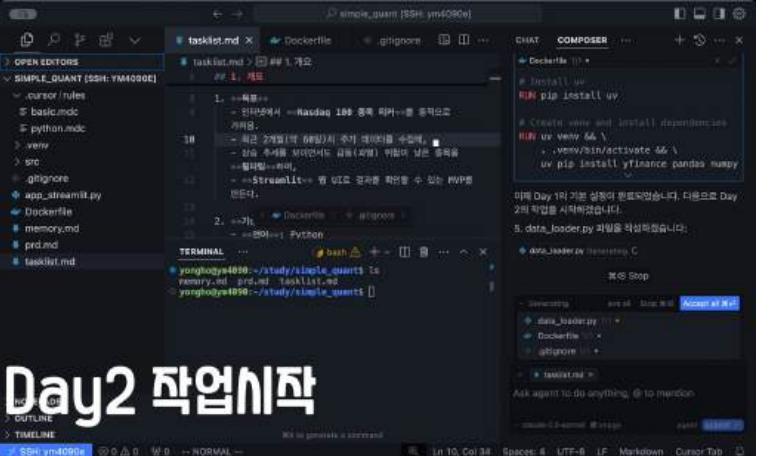
# Day1 징업시작



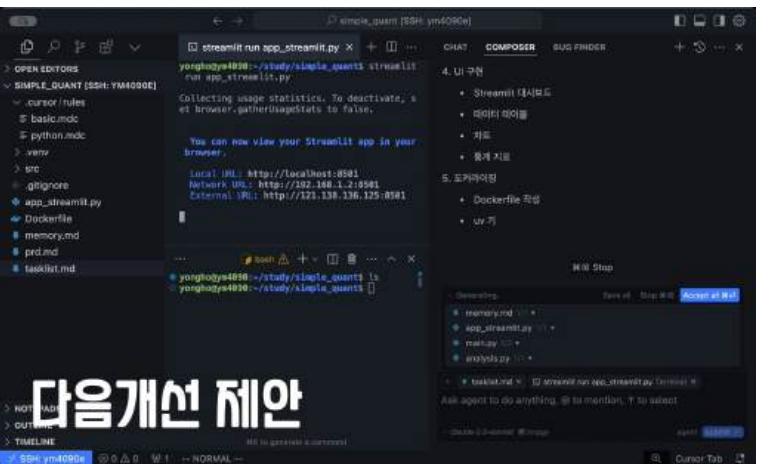
기능들 자체 테스트



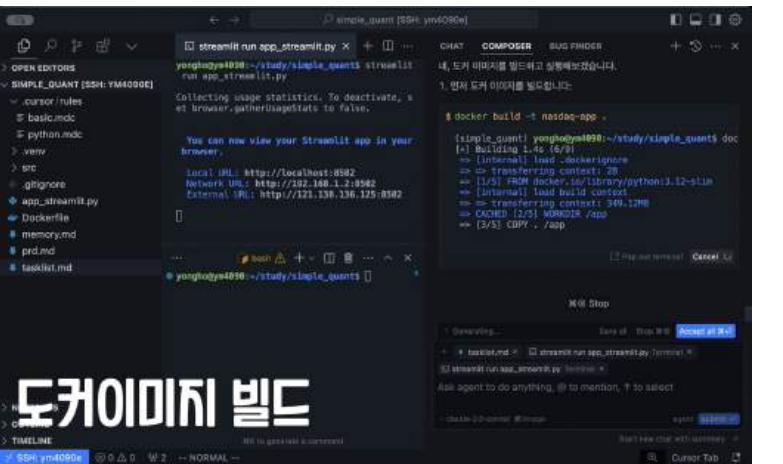
# 셀프 테스트



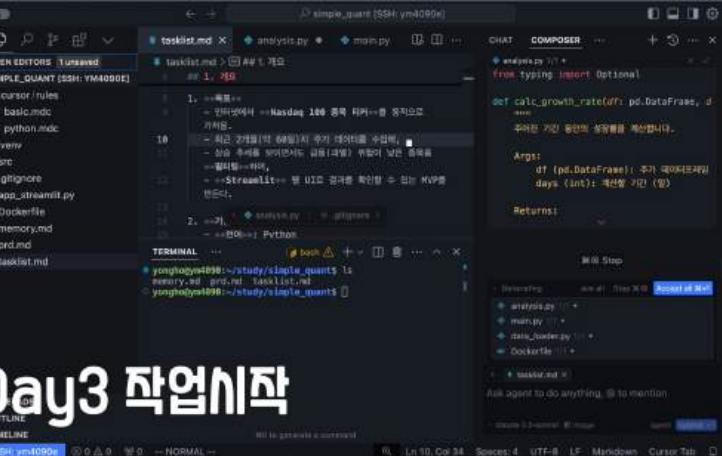
# Day2 징업시작



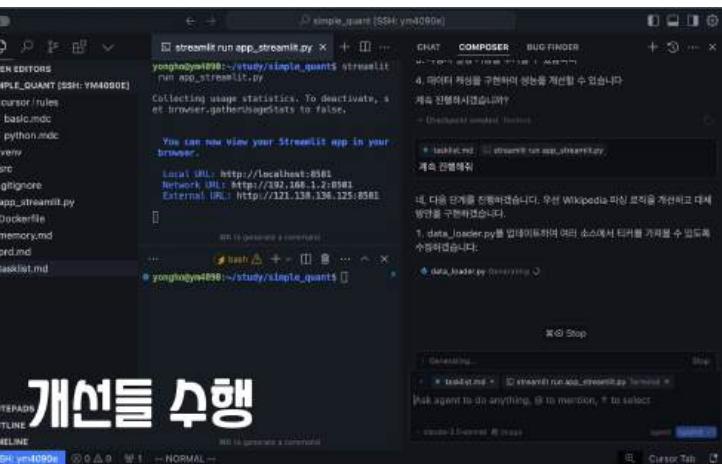
> HOT TUBS  
> OUTDOOR



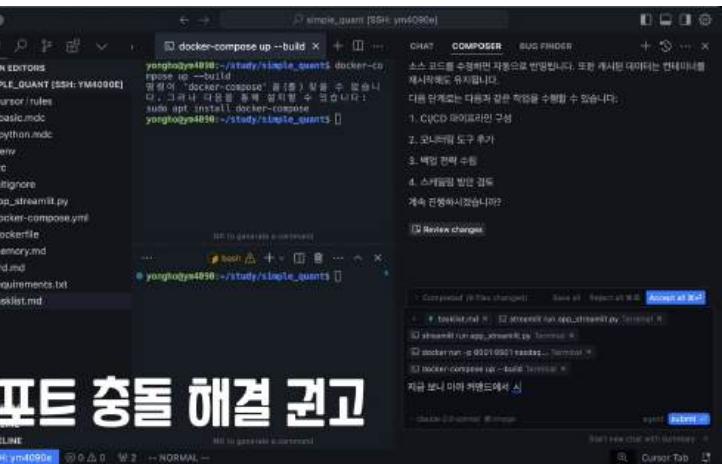
▶ N S  
▶ CUSTOM



# Day3 **작업시작**



개선을 수행  
TEPADS  
TLINE



# 포트 충돌 해결 권고

# 결과물. 두동.

Deploy

⋮



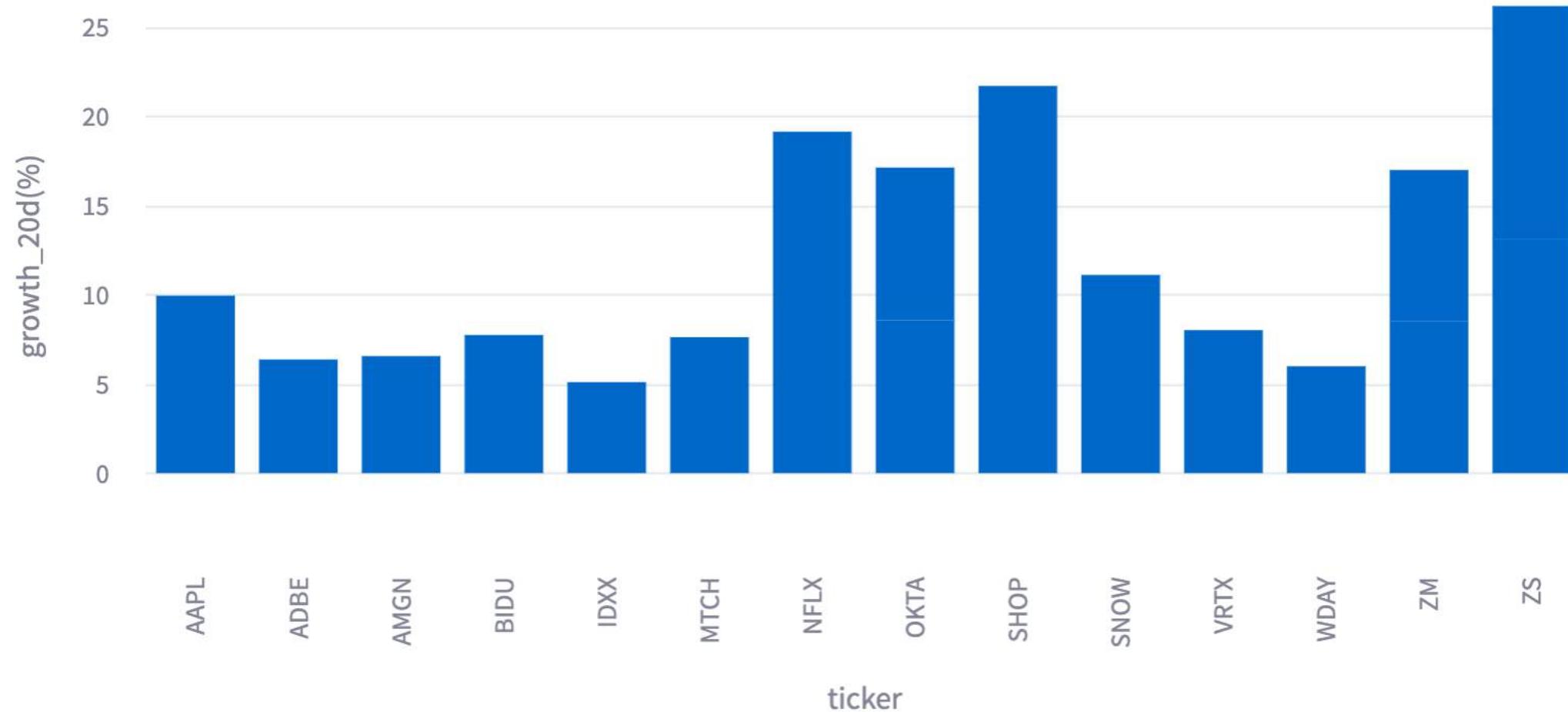
## Nasdaq 100 상승 추세 분석

이 대시보드는 Nasdaq 100 종목 중 다음 조건을 만족하는 종목을 보여줍니다:

- 20일 기준 상승 추세 (5% 이상 상승)
- 5일 기준 과열되지 않음 (15% 미만 상승)
- RSI 70 미만
- 단기 이동평균이 장기 이동평균보다 높음
- 현재가가 볼린저 밴드 상단 미만

## 차트 선택

20일 성장률



방금 본 것과 같은  
새로운 일하는 방식  
**‘vibe coding’**

(어우 ㅠㅠ 또 유행어야. 싫어엉)



Andrej Karpathy ✅  
@karpathy

There's a new kind of coding I call "vibe coding", where you fully give in to the vibes, embrace exponentials, and forget that the code even exists. It's possible because the LLMs (e.g. Cursor Composer w Sonnet) are getting too good. Also I just talk to Composer with SuperWhisper so I barely even touch the keyboard. I ask for the dumbest things like "decrease the padding on the sidebar by half" because I'm too lazy to find it. I "Accept All" always, I don't read the diffs anymore. When I get error messages I just copy paste them in with no comment, usually that fixes it. The code grows beyond my usual comprehension, I'd have to really read through it for a while. Sometimes the LLMs can't fix a bug so I just work around it or ask for random changes until it goes away. It's not too bad for throwaway weekend projects, but still quite amusing. I'm building a project or webapp, but it's not really coding - I just see stuff, say stuff, run stuff, and copy paste stuff, and it mostly works.

"바이브 코딩"이라고 부르는 새로운 종류의 코딩이 있는데, 완전히 바이브에 굴복하고 지수를 받아들이고 코드가 존재한다는 사실조차 잊어버리는 것입니다. LLMs(예: Cursor Composer w Sonnet)이 너무 좋아져서 가능합니다. 또한 저는 SuperWhisper로 Composer와 대화하기 때문에 키보드를 거의 만지지 않습니다. "사이드바 패딩을 절반으로 줄여줘"와 같은 가장 어리석은 것을 요청하는데, 찾기가 너무 귀찮기 때문입니다. 저는 항상 "모두 수락"하고, 더 이상 diff를 읽지 않습니다. 오류 메시지가 나타나면 주석 없이 복사해서 붙여넣기만 하면 대개 해결됩니다. 코드가 평소의 이해 범위를 넘어서서 정말 한참 동안 읽어야 합니다. 때로는 LLMs가 버그를 수정할 수 없어서 그냥 해결하거나 사라질 때까지 무작위로 변경해 달라고 요청합니다. 일회용 주말 프로젝트로는 나쁘지 않지만, 여전히 꽤 재밌습니다. 프로젝트나 웹앱을 만들고 있지만, 실제로 코딩하는 건 아닙니다. 그냥 보고, 말하고, 실행하고, 복사해서 붙여넣기만 하면 되는데, 대체로 작동합니다.

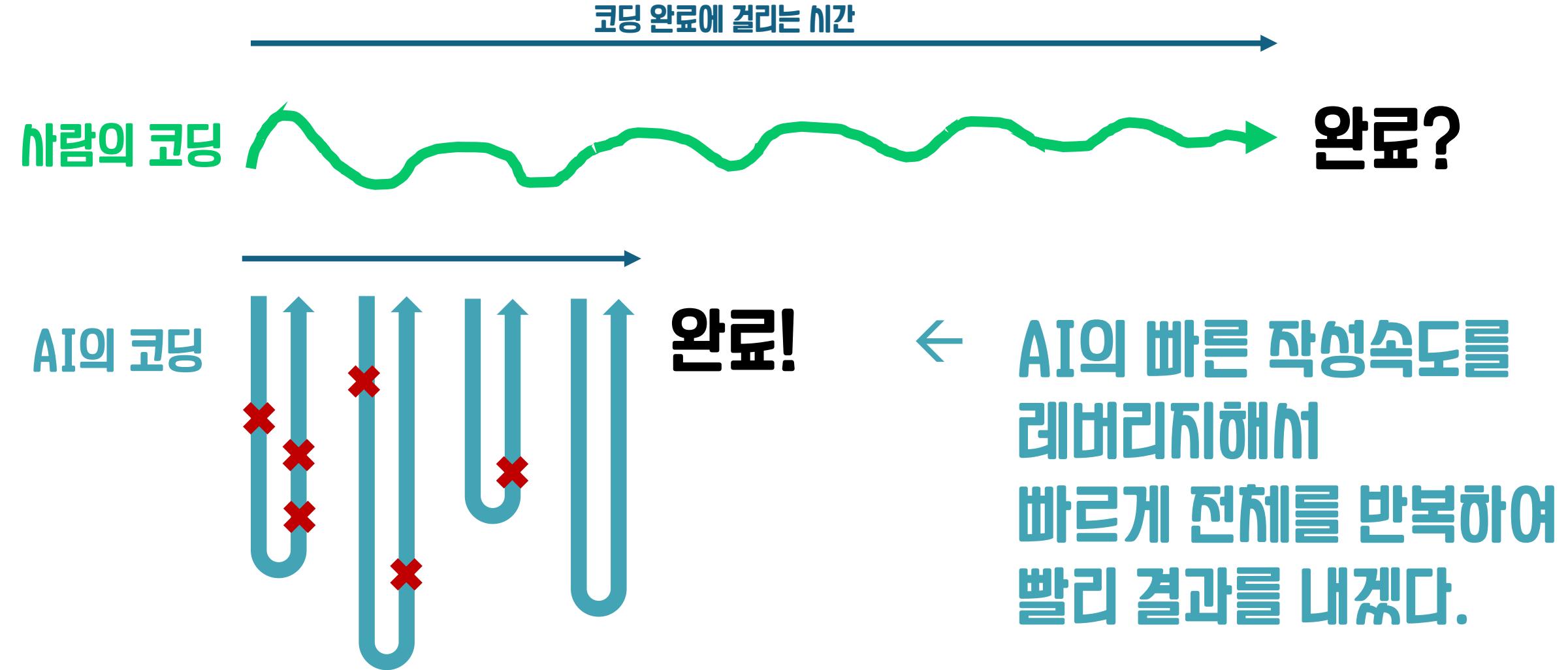
게시물 번역하기

# Vibe 코딩 '작성자 역전의 세계'

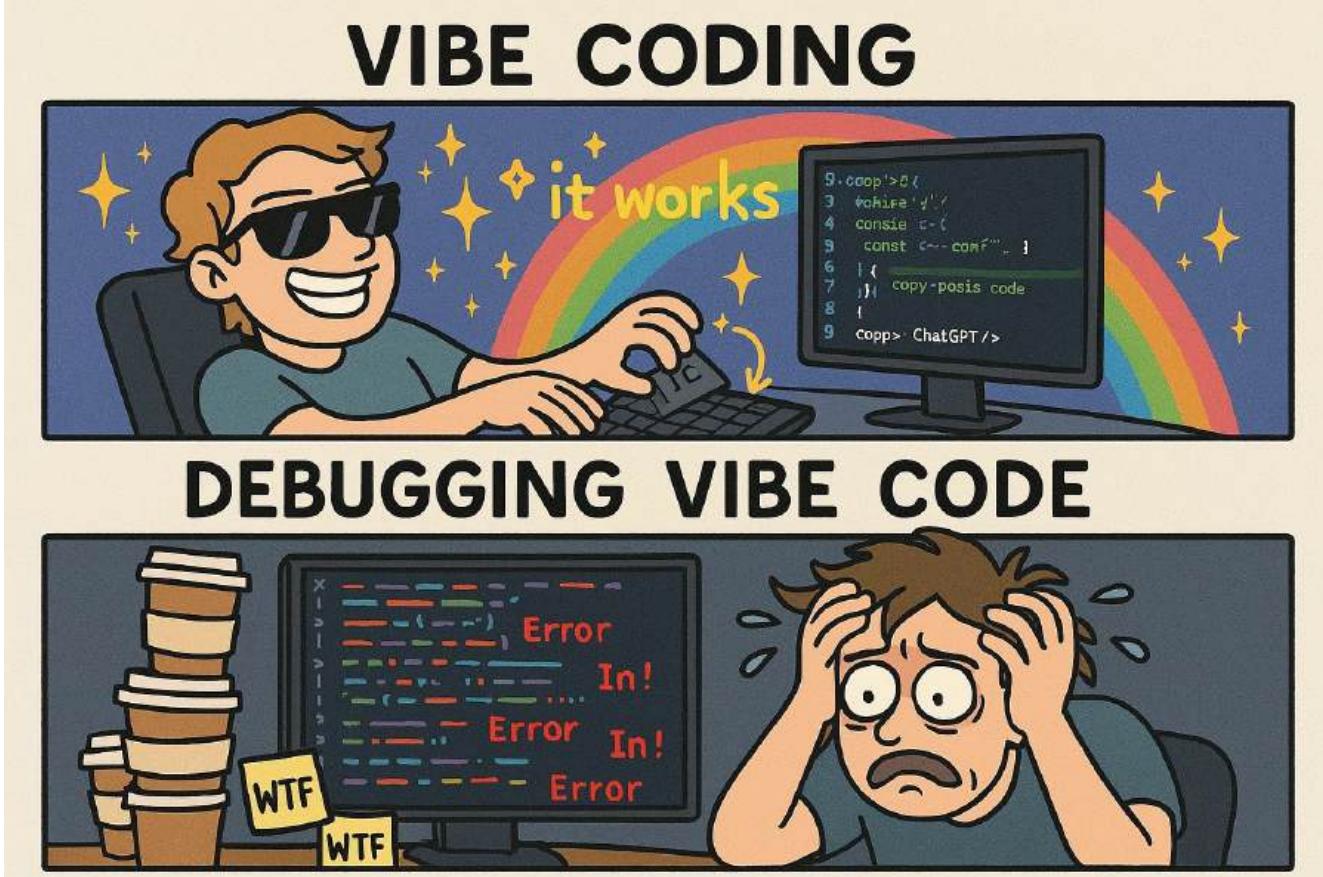
예전의 방식 – 내가 짜고 AI가 돋는다.

Vibe 코딩 – AI가 짜고 내가 돋는다.

# Vibe coding의 취지는 뭘까?



# Vibe 코딩에 대한 비판도 많습니다.



**“이건 베이스코드가  
없을 때만 가능하다!”**

**“코딩말고 디버깅하다  
사람이 죽겠다.”**

김

김대리. 내가 감히 조언 하고 싶은것이 있습니다.  
다른것이 아니고, 너무 엑셀 팡션? 사용 하지 마세요.  
편리함이 있다면, 위험성은 증대하죠. 소를 잡는데는  
그만한 칼날이 있고 닭잡는데는 칼이 필요 한가요?  
쉬운것이 정답 일수 있습니다.

MMS

오후 2:02

김

Data 취합, 정리, 단순한 방법 있어요. 별, 시간도  
필요 없고, 나중 마날로그 방법도 있죠.

오후 2:09

김

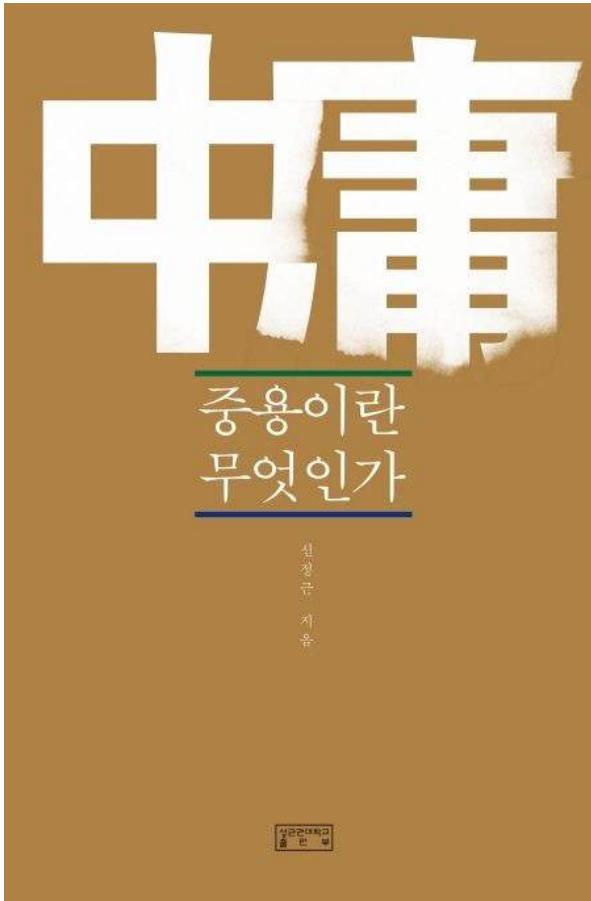
김대리가 전쟁터에 장군이라 가정하죠. 전쟁에서 이겨야  
하는것은 당연 한것 아닌가요? 그 상황에 맞는 전략?  
기상군으로만 계획한다? 아니죠. 거의 의견은 암산이  
빠를수 있고, 물론 사람에 차이는 있지만, 계산기가  
좋을수 있죠. 컴퓨터는 소잡는 칼 아닌가 해서 의견  
드립니다.

MMS

오후 2:25

또 하나의 팡션인가?

답은 언제나 극단 말고 중간에 있다.

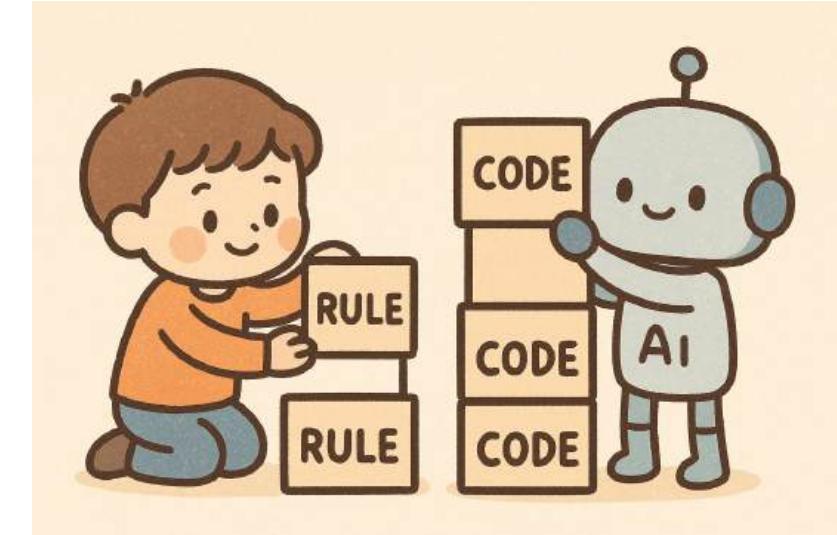


# (코드베이스가 있을 때) 나의 Vibe 코딩의 패턴

- 코드의 95%를 AI가 짜고 나는 구조를 검토하고 디버깅한다.
- 디버깅에서도 나의 역할은 문제를 발견하는 것까지. (변경 코드는 다 읽는다)
- 중요) 만약 중요한 새 원칙이 발견되면 rule이나 주석에 추가해 업데이트
- 해당 코드의 수정은 나 말고, 다시 AI에게 재작성을 맡긴다.
  - 내가 일일이 타자로 에러 수정하는 속도 << AI가 재작성하는 속도
  - 나의 역할은 에러를 자세히 리포트하고 수정방향을 권고하는 것
- 코드가 장황해지지 않기 위해 몇백라인정도 쌓이면 작은 리팩토링 지시
  - 큰 리팩토링을 LLM에게 한방에 맡기면 종종 망침.
  - 또 LLM들의 Context Window 사이즈를 생각하면 큰 파일보다 작은 파일 여러개가 유리

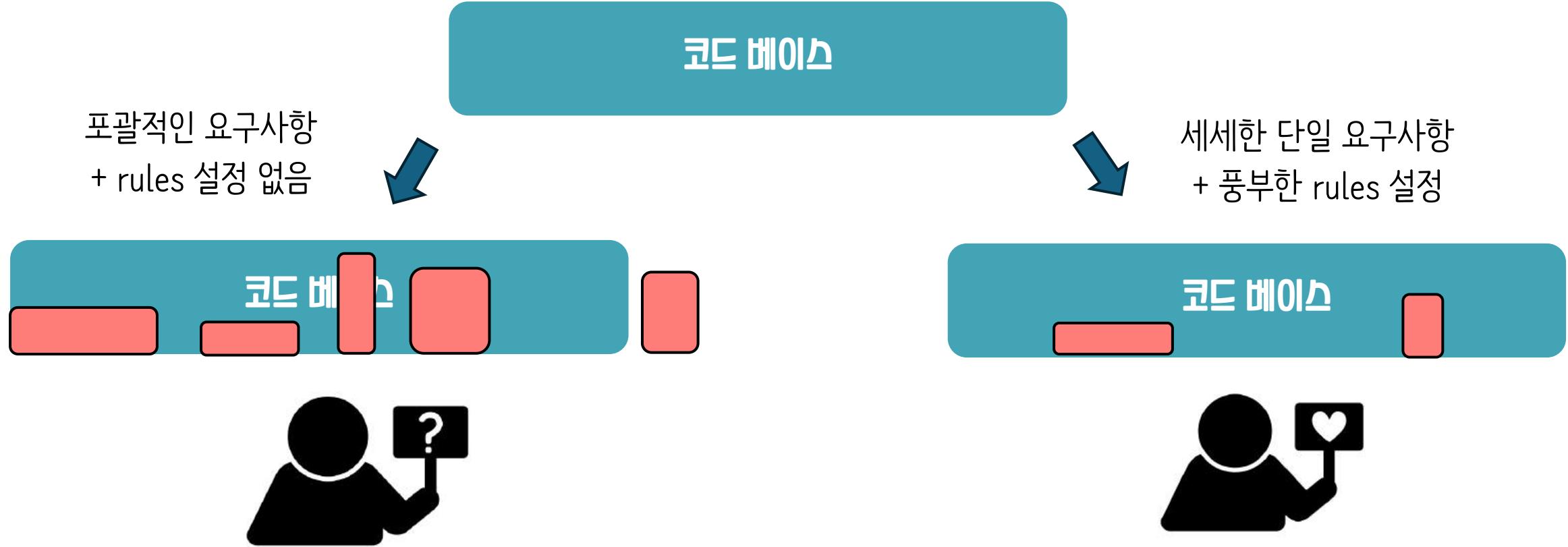
# Rule-Growing Development

- 예전의 Test Driven Development
  - 내가 테스트를 짜고
  - 코드를 짠다.
  - 테스트와 코드가 함께 자라난다.
- Rule Growing Development
  - LMM에게 시키고
  - 이상한 짓을 관찰한 후, 원하는 방향의 새 룰을 추가한다.
  - 룰은 개별 프로젝트마다 필요한 지식도 함께 포함한다
  - 코드와 룰 묶음이 함께 자라난다.
  - 이 룰과 지식도 팀 레포지토리에서 버전관리 대상



**이제 일의 산출물은 “코드” + “AI에게 막일 자료”라는 인식**

# AI 도움을 지속적으로 받으려면 어떻게?



# ‘도대체 뭘 해놓은거야?’ → 역사 AI 별로야. 손코딩 가자

# ‘이정도 변경이면 사람 검토 가능’ → AI 개발 지속 가능

# 일하며 배우게 된 팁들

## Rules

Rules provide more context to AI models to help them follow your personal preferences and operate more efficiently in your codebase. [Learn more about Rules](#)

### User Rules

These preferences get sent to the AI on all chats, composers and Command-K sessions.

Always respond in 한국어

# 0→1이 아니라 코드베이스 개선일 때

- ‘가능한 간단한 솔루션을 따르세요’
  - 이 부분은 내용 달리해서 여러번 반복 강조해야할 수 있음
  - ‘시킨 것만 고치세요’
  - ‘지금 내가 지시한 것과 상관 없는 것은 바꾸지 마세요’
  - ‘구조 자체를 바꾸지는 마세요’ 등등
  - 그렇게 해야, 한번에 수정하는 볼륨이 적어져서, 코드 검토할 때 편함
- Dev환경과 test, prod환경 구분을 잘하세요
  - 디폴트로 환경까지는 고려하지 않기 때문에, prod, dev가 나뉜 경우
  - 자세한 설명을 추가해두는 것이 필요

# 참고) IDE도구들이 참조할 코드를 찾는 법

- 이들도 결국 내 코드파일들을 작은 조각들로 나누고
- 조각 텍스트를 임베딩 벡터로 만들고
- 지금 해야할 일(프롬프트)와 similarity가 높은 조각을 찾음
- 이렇게 RAG 방식으로 새 코드를 작성

# 사람보다 AI에게 코드 주석이 더 중요

- 실제 의미론적으로는 유관한 코드인데, IDE내부 RAG가 발견을 못함
- 예를 들어 테이블 이름은 transaction인데,  
실제로는 구매나 은행 거래랑 상관없고, 실명 인증 기록이었다고 한다면?
  - 현실 세계에서는 전임자가 잘못 해놓은 네이밍으로 이런 일이 비일비재
- 이런 부분들이 RAG에서 잘 동작할 수 있게,  
우리가 명시적 주석이나 docstring이 코드 곁에 추가 되어 있어야 함
  - 어쩌면 사람에게 알려주는 것보다 더 중요
- 하다 못해 파일 이름이라도 길게 지어주면 도움이 됨
  - A.py (X) page.py(X)
  - user\_identity\_check.py (0)

# ‘중복된 평션이 있는지 체크하라’ 강조

- 생각보다 많은 실수가 나오는 곳이 여기
  - 대부분의 에디터들이 전체 코드베이스를 청크로 잘라 embedding하여 유관한 코드 부분을 탐색
  - 문제는 이게 적중률 100%가 아니기 때문에, 조금 탐색해보고
- 지금 이 평션이 없다. 라고 잘못 판단하고 새로 평션 작성
  - 철저히 찾아보고 정말 없을 때만 하라고 하거나
  - 내가 알고 있다면, 특정 평션 명시하고 이를 개선하라고 해야함
  - 그래도 발생할 수 있다… 가장 자주 발생하는 문제

# SQL 작성을 AI가 잘하게 하려면

- 내가 주로 쓰는 DB의 내용을 별도 파일로 알려준다.
  - 스키마
  - 샘플데이터
  - 샘플쿼리

The screenshot shows a configuration interface with the following details:

- Path: .cursor > rules > ≡ data-rule.mdc
- Rule Type: Always (selected)
- Description: This rule attached to every chat and command+k request
- Content:
  - # SQL 작성률
    - 모든 데이터들은 all\_data.duckdb에 duckdb 테이블로 들어가 있다.
    - sqlite의 특화기능을 쓰지말고, duckdb의 문법과 함수를 사용해라.
    - 각 테이블의 세부 스키마와 컬럼에 대한 설명은 @table\_desc.yaml 에 들어있다.
    - 각 테이블의 샘플데이터는 @table\_info\_and\_sample.txt 파일에 명시되어 있다.
    - SQL의 샘플들은 @sql\_examples.json 에 들어있다.
    - SQL 작성시 미리 주어진 세부 스펙 파일과 샘플파일을 항상 참조해서 작성해라.
  - # matplotlib visual rule
    - 한글표현을 위해 항상 이것을 지정 plt.rcParams ["font.family"] = "NanumGothic" 항상 지정
    - 마이너스 기호 깨짐 방지를 위해 plt.rcParams ["axes.unicode\_minus"] = False 항상 지정
    - 레티나 디스플레이 설정 plt.rcParams ["figure.dpi"] = 144 # 레티나 디스플레이를 위한 DPI 설정
    - 분석의 결과를 이미지 파일등으로 저장할 때는, 미리 파일이 저장될 폴더를 만들고, 그 안에 써라.

# 파이썬 프로그래머라면

- Python venv 관리자를 뭐 쓸지
  - Poetry, uv
  - Venv를 쓴다면 ‘작업하기 전에 항상 activate venv 하라’고 지시해주기
    - 그렇지 않으면 환경미스로 쓸데없이 자꾸 package 설치함
- Unittest쓰지 pytest 쓸지
- 그리고 내가 주로 쓰는 라이브러리들 명시.
  - 명시해주지 않으면 너무 대중픽으로 가는데 종종 맘에 들지 않음
  - 저는 주로 sqlalchemy 버전을 2 기준으로 하라는 것
  - Pandas 디스플레이할 때, tabulate와 CJK wide char 켜서 한글표도 예쁘게 보이게
  - Diskcache 라이브러리와 데코레이터를 활용해 펑션내용 디스크에 캐시하기
  - json 다룰 때 디폴트 json 대신에 orjson, msgspec 쓰라는 것 (이때 indent 안쓰기)
  - 로깅은 loguru를 쓰라는 것

# 너무 큰 파일이 인덱스 되지 않게 막을 것

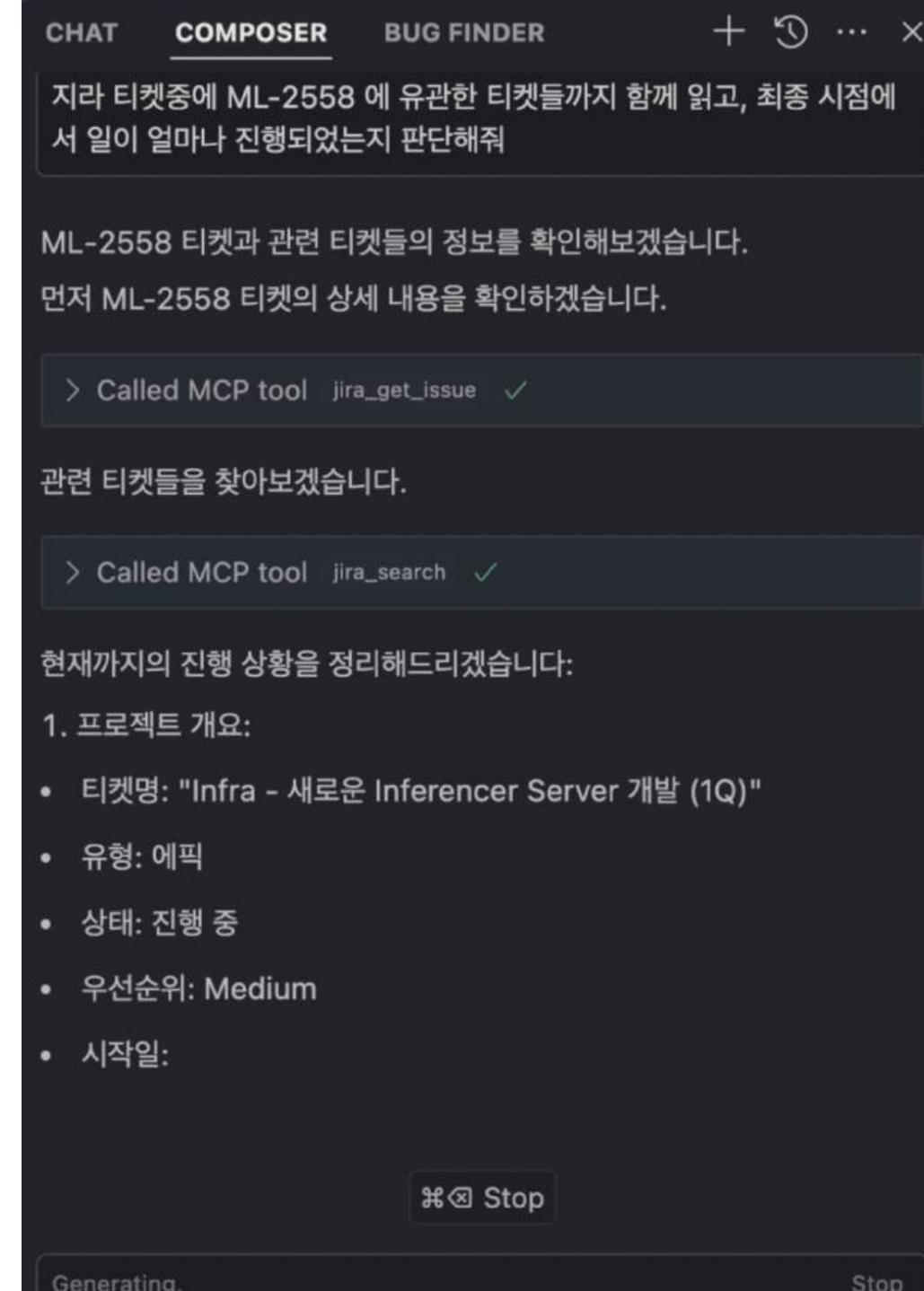
- 너무 큰 파일이 당신의 IDE의 Vecrto Stroage에 인덱스 되지 않도록 ignore 에 잘 추가해두도록.
  - 내 duckdb파일이 문제를 일으켰다.
  - Cursor의 .cursorignore에 추가

# AI의 해결책을 지켜보고 있어야 한다.

- 근본 원인은 해결 안하고, 증상만 해결할 때가 있다.
- 문제 : 메모리를 과도하게 씁니다.
  - 우리가 기대한 해결책 : 메모리를 적게 쓰는 효율적인 자료구조
  - AI가 한 일
    - 메모리를 늘린다?
    - 재시도 횟수를 늘린다?
    - 경고를 삭제한다???
- 자꾸 우회하는 해결책을 선택할 때가 있기 때문에
- 그런짓을 삼가하도록 rule에 추가해야 한다.

# 다른 서비스들의 문맥마저도 가져오는 MCP (Model Context Protocol)

- LLM이 여러 외부 데이터 소스나 도구들과 통합되게 하는 프로토콜
  - 슬랙에서 여러 앱 플러그인 쓰는 것과 비슷하다
  - 원래 Cluade 거였는데 앞으로 GPT도 지원한다고 함
- LLM에서 사람이 자연어로 치면
  - 이게 rest든, cmd콜이든 뭐든 실행해주고 결과를 돌려준다.
- 외부 개발 개발도구들과도 연동해 쓸 수 있다.



**https://smithery.ai/는  
다양한 MCP를 찾고,  
각 플랫폼에 설치할 때 세부 방법을 알려줘서 편함**

## Installation

AI Claude Cursor Windsurf Cline Typescript

### Install Command

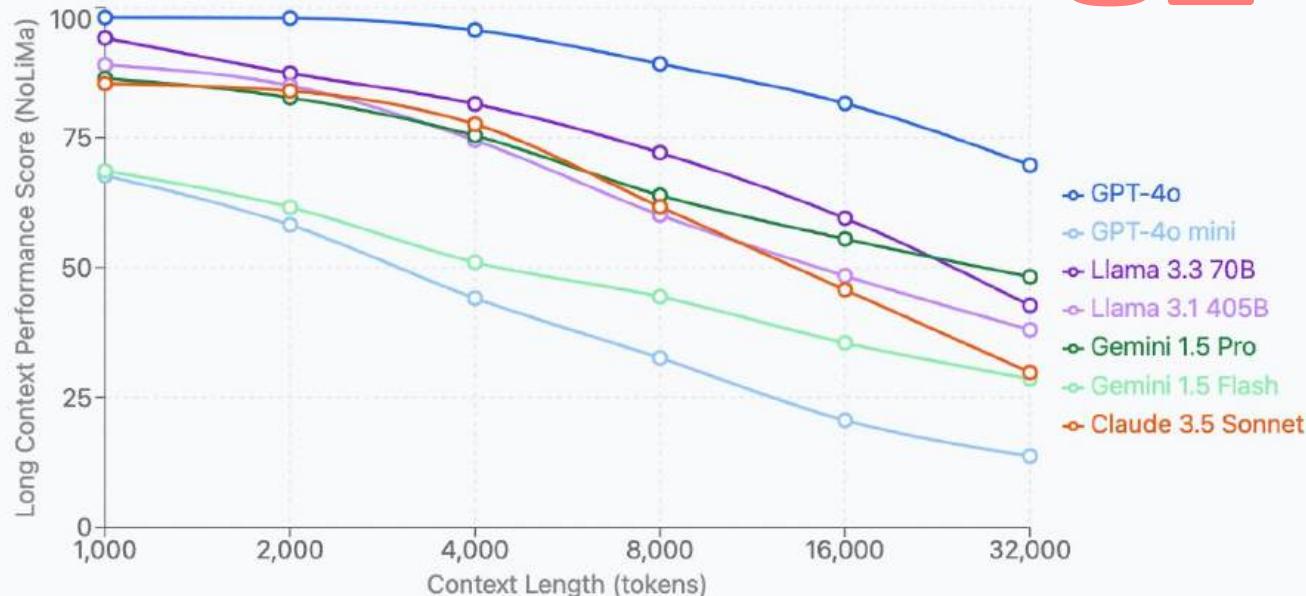
Integrate this tool for Cursor by copying the following into Cursor's MCP command.

```
npx -y @smithery/cli@latest run docker-mcp --config "{}"
```

Report Bug

## Model Performance vs Context Length

NoLiMa Benchmark Results



NOLIMA: Long-Context Evaluation Beyond Literal Matching

Models	Claimed Length	Effective Length	Base Score ( $\times 0.85$ : Thr.)	1K	2K	4K	8K	16K	32K
GPT-4o	128K	8K	99.3 (84.4)	98.1	98.0	95.7	89.2	81.6	69.7
Llama 3.3 70B	128K	2K	97.3 (82.7)	94.2	87.4	81.5	72.1	59.5	42.7
Llama 3.1 405B	128K	2K	94.7 (80.5)	89.0	85.0	74.5	60.1	48.4	38.0
Llama 3.1 70B	128K	2K	94.5 (80.3)	91.0	81.8	71.2	62.7	51.8	43.2
Gemini 1.5 Pro	2M	2K	92.6 (78.7)	86.4	82.7	75.4	63.9	55.5	48.2
Jamba 1.5 Mini	256K	<1K	92.4 (78.6)	76.3	74.1	70.8	62.2	52.7	43.6
Command R+	128K	<1K	90.9 (77.3)	77.0	73.5	66.3	39.5	21.3	7.4
Mistral Large 2	128K	2K	87.9 (74.7)	86.1	85.5	73.3	51.5	32.6	18.7
Claude 3.5 Sonnet	200K	4K	87.6 (74.4)	85.4	84.0	77.6	61.7	45.7	29.8
Gemini 1.5 Flash	1M	<1K	84.7 (72.0)	68.6	61.6	51.0	44.4	35.5	28.6
GPT-4o mini	128K	<1K	84.9 (72.2)	67.7	58.2	44.1	32.6	20.6	13.7
Llama 3.1 8B	128K	1K	76.7 (65.2)	65.7	54.4	44.1	31.9	22.6	14.2

Table 3. NOLIMA benchmark results on the selected models. Following Hsieh et al. (2024), we report the effective length alongside the claimed supported context length for each model. However, we define the effective length as the maximum length at which the score remains above a threshold set at 85% of the model's base score (shown in parentheses). Scores exceeding this threshold are underlined. Scores that are below 50% of the base score are shaded in red.

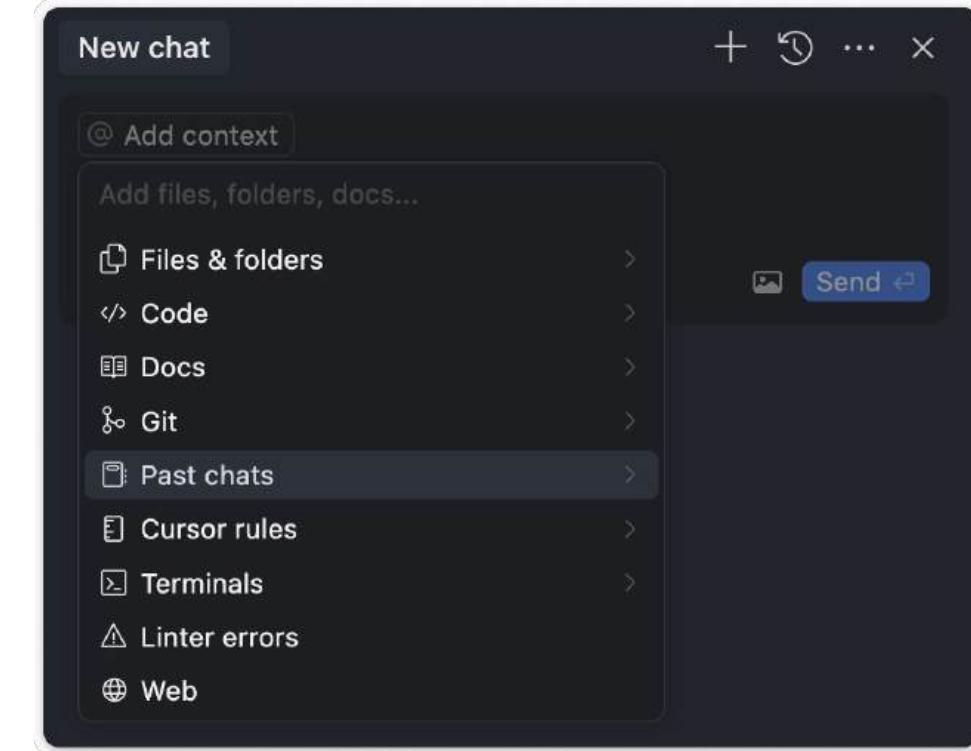
# 중간 중간 새로 세션을 여세요

다들 잘 알다시피 LLM은 1개의 대화세션이 길어지면 (context가 커지면) 바보가 되고 한다.  
2025년에도 여전히 그리고 있음

중간 중간  
적절한 만에서 끊고  
새로 대화를 시작해야 한다.

# 앞선 대화를 이어가는 새 채팅 열 때 팁

- ChatGPT를 쓸 때도 길어지면 새 대화 열어야 했다.
  - 매번 이전 대화 요약해서 붙이는 것 귀찮았음
- 다행히 cursor등에는 앞선 대화의 요약을 가져오는 기능이 있음
  - Past Chats
  - 어느 대화에서 문맥을 가져올 지 지정 가능!



# 테스트 코드를 함께 작성해달라고 하기

- 우리 전체 코드베이스를 참조해서 코딩을 하는 경향
- 그런데 고치다보면 어느샌가
  - A를 고치면 B가 고장나고
  - B를 고치면 A를 이상하게 고치고 하곤 함
  - 중대가 없어서 그러함
- 보통 대화 세션을 새로 열면 해결되지만,
  - 전체 작업 맥락 보존 어떻게 하지? (summarize 도 귀찮다)
- 작성해놓은 테스트코드가 코드베이스에 포함되어 가드레일이 되어줌
  - 이상한 곳으로 가는 것을 막아준다.
- AI시대에 와서 오히려 테스트코드가 중요해지는 듯
  - 우리가 직접 안짜도 된다는 점이 더욱 좋다

테스트 코드란

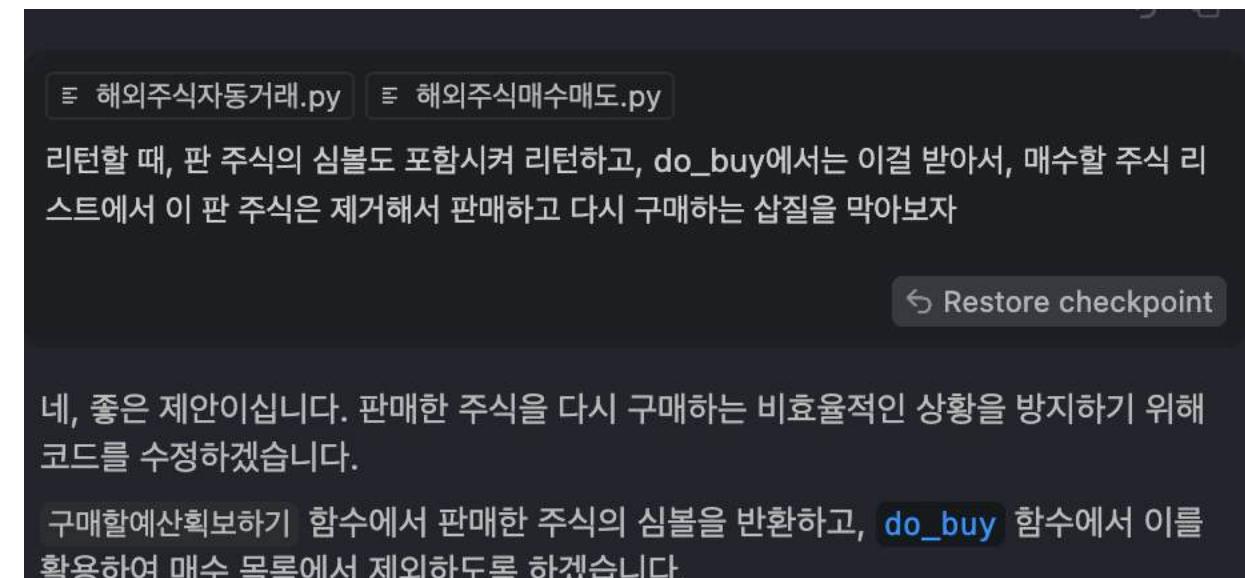
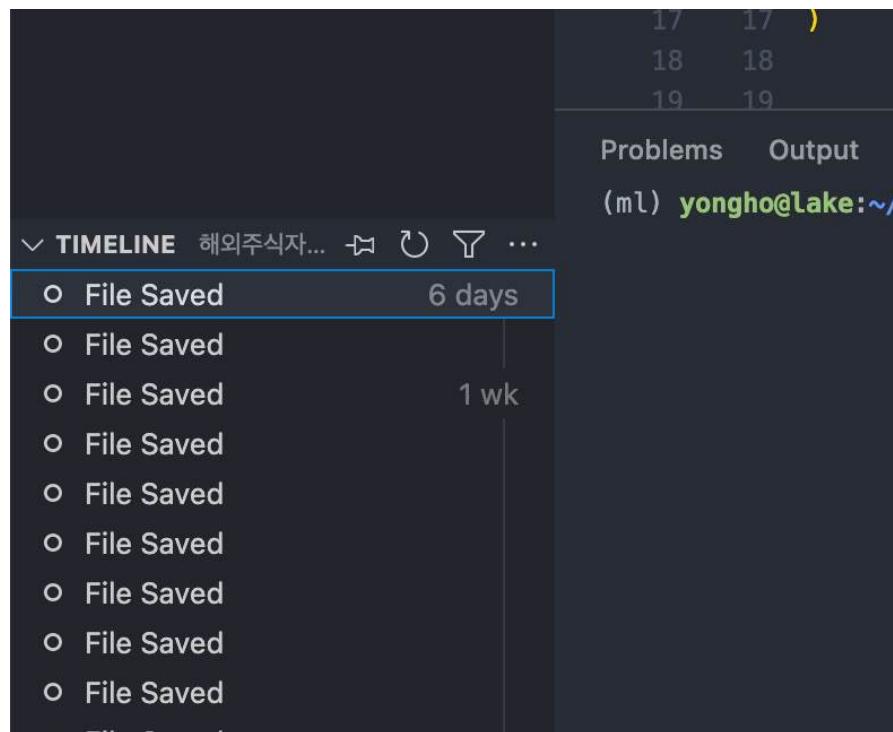
자신의 코드를 믿지 못하는

나약한 프로그래머들이 짜는 것이다!

(아니야. 틀려 😭)

# Commit 안했는데 뭔가 어떻게든 돌아가기

- AI 코딩은 평소보다 잳은 commit 이 필요. 하지만 안했다면.
- 왼쪽에 있는 TIMELINE을 이용하거나
- 대화 메시지에서 restore



여기까지는 cursor를 쓸 때의 팁들

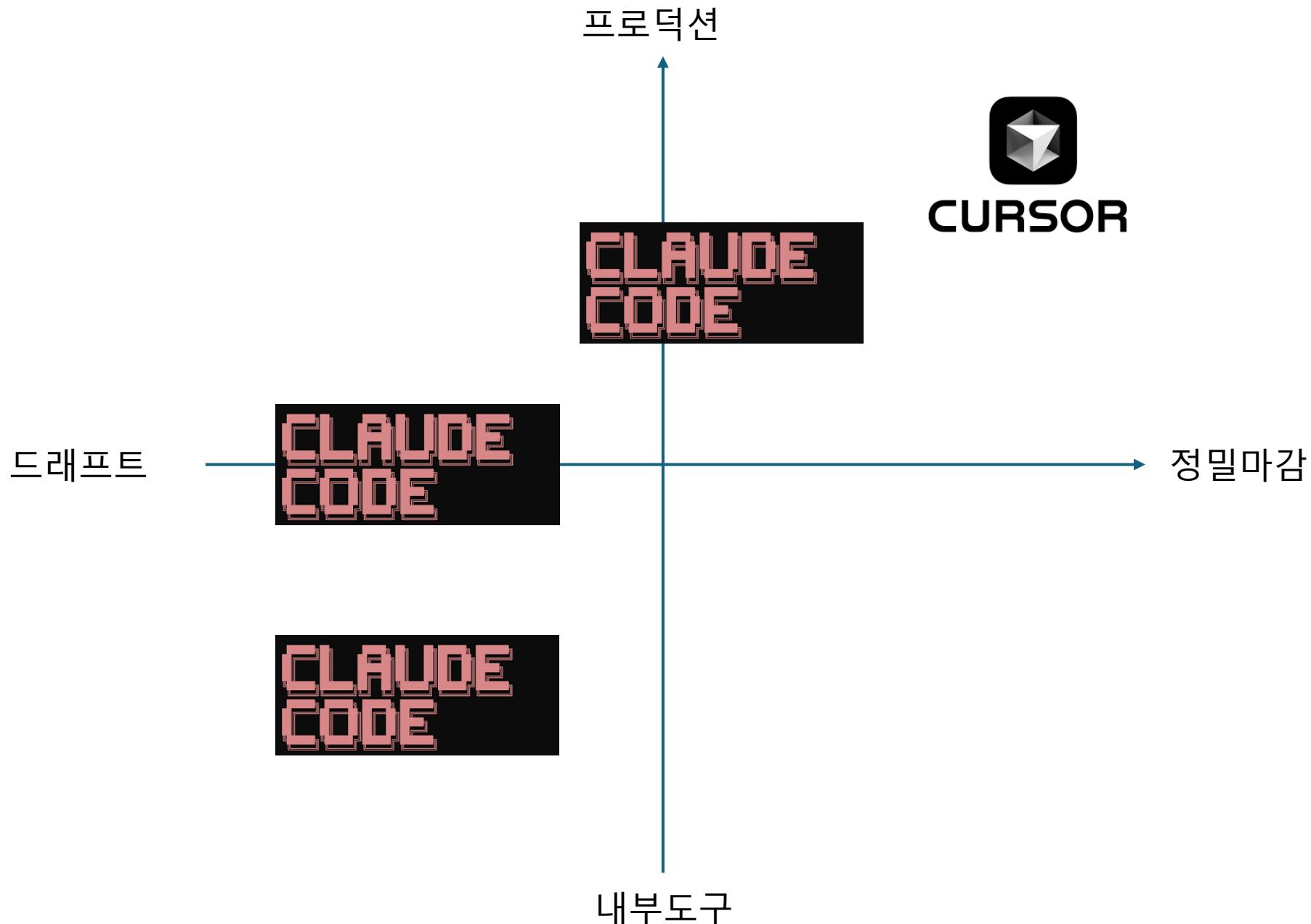


이제 클로드코드다!

# 일단 가장 놀랄만한 것은 에이전트 성능

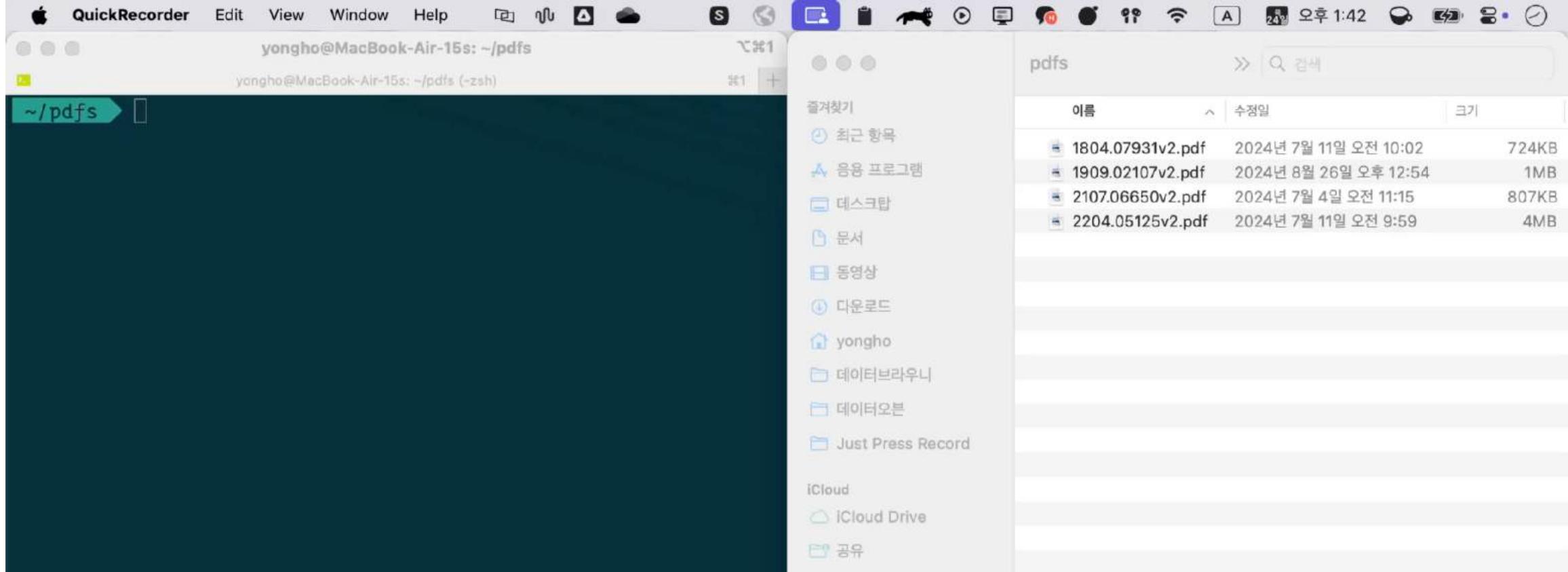
같은 에이전트 모드라 해도  
작업 완수율이 커서보다 클로드 코드가  
월등히 높다.

시켜놓고 지켜본다의 진정한 완성



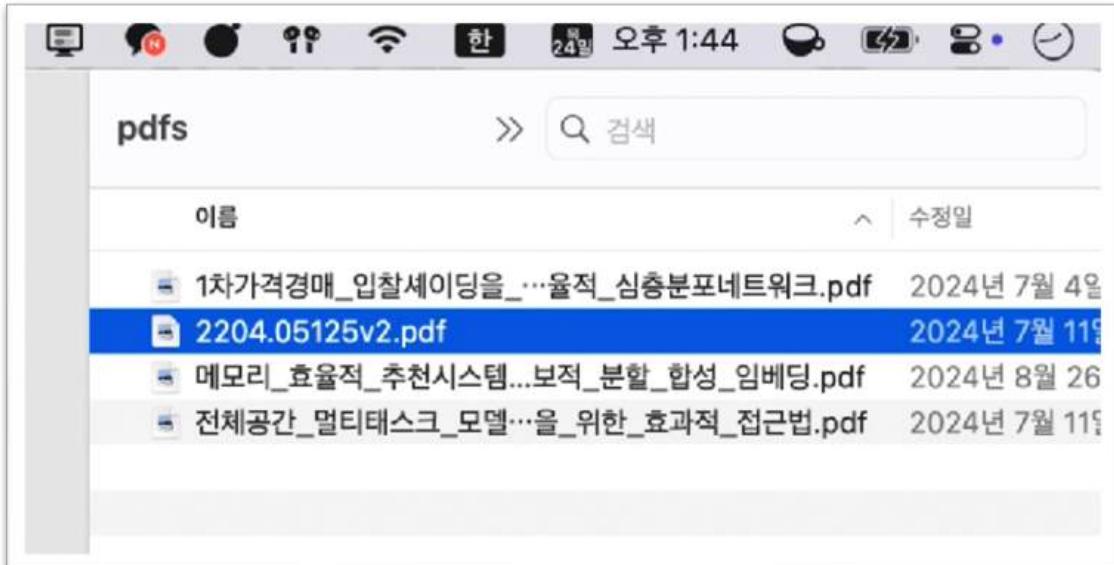
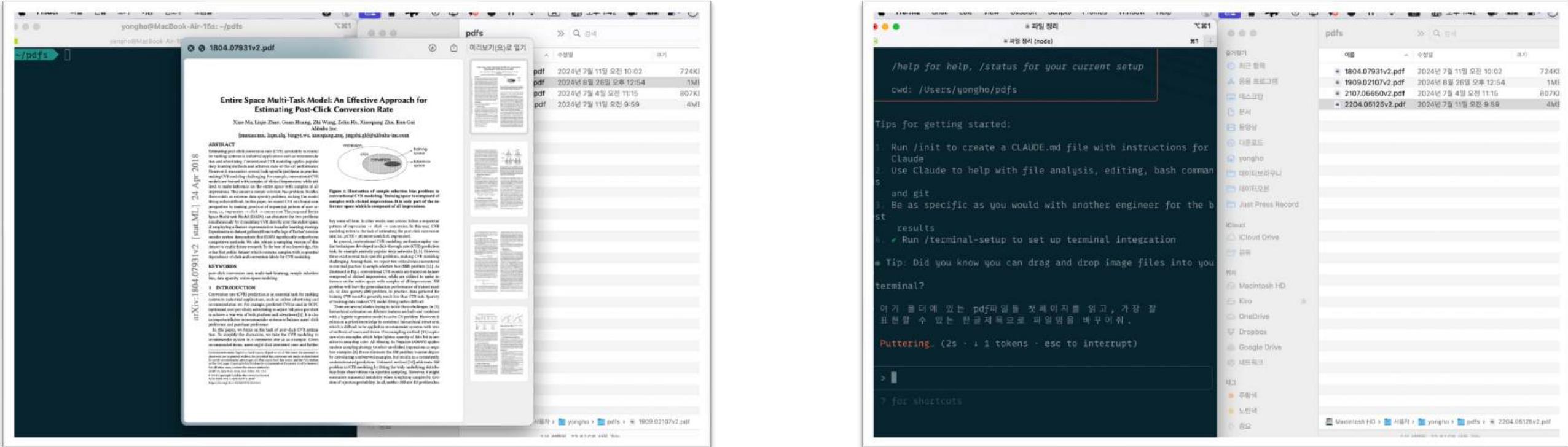
**좋은 점은 개발 이외 분야도 쓸 수 있다는 것**

**내 로컬 머신과 상호작용이 된다는 것**



폴더에 들어있는 파일 이름이 문제다.  
Arxiv에서 논문다운 받으면 이름이 이모양





**“파일을 첫페이지에 읽고, 가장 잘 표현할 수 있는 한글파일명으로 바꿔줘”**

**Arxiv에서 다른 받은 정체를 알 수 없는 파일의 내용을 보고 알아서 내 파일 이름을 변경**

\* File Documentation

\* File Documentation (node)

> 읽은 김에 각 파일이 어떤 내용인지도 정리해서 markdown 파일로 하나 만들어 줘 .

- Write(논문\_요약\_정리.md)
  - ↳ Wrote 78 lines to 논문\_요약\_정리.md
  - # PDF 논문 요약 정리
  - ## 1. 전체 공간\_멀티태스크\_모델\_클릭 후\_전환율\_추정을\_위한\_효과적\_접근법.pdf
  - \*\*원 제 목\*\*: Entire Space Multi-Task Model: An Effective Approach for Estimating Post-Click Conversion Rate
  - \*\*주요 내용\*\*:
    - 온라인 광고 및 추천 시스템에서 클릭 후 전환율 (CVR) 예측의 중요성
    - 기존 CVR 모델링의 두 가지 주요 문제점 :
      - 샘플 선택 편향 (Sample Selection Bias): 클릭된 노출만으로 학습하지만 전체 공간에서 추론
  - ... +68 lines (ctrl+r to expand)
- 논문 요약 파일을 생성했습니다. 각 논문의 주요 내용을 다음과 같이 정리했습니다 :

1. ESMM 논문 : 클릭 후 전환율 예측의 샘플 선택 편향과 데이터 희소성 문제를 해결하는 전체 공간 멀티태스크 모델
2. 합성 임베딩 논문 : 상보적 분할을 활용하여 추천 시스템의 메모리 효율성을 크게 개선하는 방법
3. 입찰 세이딩 논문 : 1차 가격 경매에서 최적 입찰 가를 결정하는 심층 분포 네트워크
4. ESCM<sup>2</sup> 논문 : ESMM의 이론적 한계 (내재적 추정 편향과 잠재적 독립성 우선 순위)를 극복한 개선된 모델

모든 논문이 실제 산업 환경에서 검증된 실용적인 해결책을 제시하고 있습니다 .

\* 논문\_요약\_정리.md

Warp(으)로 열기

## PDF 논문 요약 정리

### 1. 전체 공간\_멀티태스크\_모델\_클릭 후\_전환율\_추정을\_위한\_효과적\_접근법.pdf

원제목: Entire Space Multi-Task Model: An Effective Approach for Estimating Post-Click Conversion Rate

주요 내용:

- 온라인 광고 및 추천 시스템에서 클릭 후 전환율(CVR) 예측의 중요성
- 기존 CVR 모델링의 두 가지 주요 문제점:
  - 샘플 선택 편향(Sample Selection Bias): 클릭된 노출만으로 학습하지만 전체 공간에서 추론
  - 데이터 희소성(Data Sparsity): 전환 데이터의 극심한 부족
- ESMM(Entire Space Multi-task Model) 제안:
  - 사용자 행동의 순차적 패턴 활용: 노출 → 클릭 → 전환
  - 전체 공간에서 직접 CVR 모델링
  - 특징 표현 전이 학습(Feature Representation Transfer Learning) 전략 사용
- Taobao 추천 시스템 데이터셋에서 실험하여 기존 방법 대비 우수한 성능 입증

### 2. 메모리\_효율적\_추천시스템을\_위한\_상보적\_분할\_합성\_임베딩.pdf

원제목: Compositional Embeddings Using Complementary Partitions for Memory-Efficient Recommendation Systems

주요 내용:

- 현대 딥러닝 기반 추천 시스템의 메모리 병목 현상 해결
- 범주형 특징의 임베딩 테이블이 주요 메모리 소비 요인
- 상보적 분할(Complementary Partitions)을 활용한 합성 임베딩 제안:
  - 카테고리 집합의 상보적 분할을 활용하여 각 카테고리에 고유한 임베딩 벡터 생성
  - 여러 개의 작은 임베딩 테이블을 저장하고 결합하여 메모리 비용 절감
- Quotient-Remainder Trick 소개:
  - 몫과 나머지 함수를 사용하여 두 개의 임베딩 생성 후 결합
  - $O(|S|D)$ 에서  $O(\sqrt{|S|}D)$ 로 메모리 복잡도 감소
- Criteo Ad Kaggle 데이터셋에서 해싱 트리 대비 우수한 성능 실증

### 3. 1차가격경매\_입찰세이딩을\_위한\_효율적\_심층분포네트워크.pdf

원제목: An Efficient Deep Distribution Network for Bid Shading in First-Price Auctions

주요 내용:

- 2019년 이후 온라인 광고 업계의 2차 가격 경매에서 1차 가격 경매로의 전환
- 입찰 세이딩(Bid Shading): 2차 가격 경매용 입찰가를 1차 가격 경매에 맞게 조정
- 심층 분포 네트워크(Deep Distribution Network) 제안:
  - 공개(non-censored) 및 비공개(censored) 1차 가격 경매 모두에서 최적 인찰

# 다양한 사용례가 많다.

- 상상력만큼 가능
  - 여기 있는 사진들을 exif 정보보고 찍은 날짜별로 폴더 정리해줘
  - 전체 사진들 중에 단체 사진만 골라서 모아줘
  - 여기 폴더에 리포트들이 많은데,  
핵심 내용 정리해서 텍스트문서로 만들어줘
  - 지난 일주일 사이의 Git 이력들보고  
주로 어떤 변화가 있었는지 디테일하게 날짜별로 정리해줘
  - 얘가 도대체 뭘 했는지 모르겠는데, 이 사람 코드 변경 의도를 추론해줘.
- MCP를 응용하면 브라우저 자동화를 통해 나대신 여러가지 작업을 시킬 수 있다.

# 이런 작업들을 명령어(command)로 등록 가능

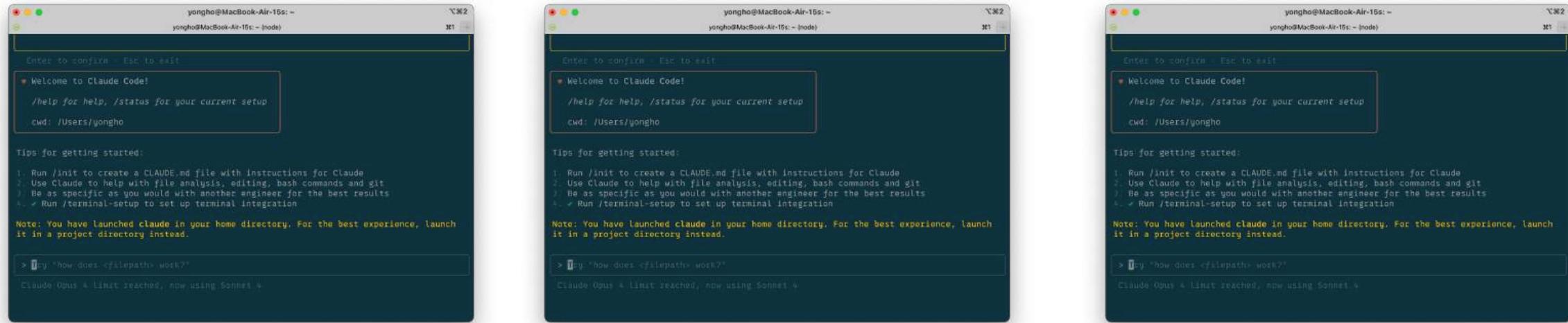
A screenshot of a terminal window titled "yongho@ym4090: ~/claude/commands". The window has two tabs: "yongho@MacBook-Air-15s: ~/claude (-zsh)" and "yongho@ym4090: ~/claude/commands (ssh)". The content of the terminal is as follows:

```
1 Please analyze and fix the GitHub issue: $ARGUMENTS.
2
3 다음 단계를 따라 진행하세요 :
4
5 1. gh issue view 명령어를 사용하여 이슈 내용을 확인합니다.
6 2. 이슈에 설명된 문제를 이해합니다.
7 3. 관련된 파일을 코드 베이스에서 검색합니다.
8 4. 문제를 해결하기 위한 필요한 변경사항을 구현합니다.
9 5. 수정 사항을 검증할 수 있는 테스트를 작성하고 실행합니다.
10 6. 코드가 린트(lint) 및 타입 검사(type checking)를 통과하는지 확인합니다.
11 7. 명확하고 설명적인 커밋 메시지를 작성합니다.
12 8. 변경 사항을 푸시하고 Pull Request(PR)를 생성합니다.
13
14 모든 GitHub 관련 작업은 GitHub CLI(gh)를 사용하세요.
```

The bottom status bar shows "NORMAL" and "fix-github.issue.md".

- Claude code 공식 베스트 프랙티스에 있는 것 한글버전
- Claude code안에서 /fix-github-issue 1234
- 깃헙 이슈번호만 치면, 프로그램이 문제를 파악하고, 풀어내고 PR을 보내온다.
- 주니어 개발자?! 채용?

# 또 좋은 점은 병렬화 하기가 쉽다는 것



## 창을 여러개 띄우면 N배의 속도!

(너가 필요한만큼 subagent띄워서 해결해줘라는 hack도 있지만 이건 토큰 소비량이 엄청나서 비추. 차라리 창을 띄워요)

# 각자 다른 이슈를 풀게된다면

- 병렬화 해서 각자 다른 이슈를 풀게 된다면
  - 동시 작업 가능하게, 각자 다른 작업 폴더를 가져야 하는데
  - 폴더마다 git clone이 뭔가 부자연 스럽다고 느낀다면
- Git worktree 기능을 강추
  - 본체는 유지한 채, branch를 다른 폴더에 따다주는 기능
  - 코드 전체 네트워크 다운 필요없음. 총알같이 새폴더
  - 이렇게 만들고 각 폴더에서 claude를 실행하면 됨

# 사실 worktree 명령 몰라도 클로드에게 시키면 됨

↳ \* Git Merge [yongho@ym4090: ~ (ssh)]

↳ \* Git Merge

\* Welcome to Claude Code!

/help for help, /status for your current setup

cwd: /home/yongho/work/spacevision\_sql\_agent

> git worktree로 만들어 낸 ../spacevision\_sql\_agent\_refactoring 가 현재 가장 최신의 코드이다. 이 코드를 지금 폴더의 메인에 병합해 달라. 문제가 생긴다면 spacevision\_sql\_agent\_refactoring쪽이 맞는 코드이다. 잘 성공하면 저 worktree는 제거 해줘.

- I'll help you merge the refactoring worktree into the main branch. Let me start by creating a todo list to track this process.
- Update Todos
  - Check current branch and worktree status
  - Review differences between main and refactoring worktree
  - Merge refactoring changes into main branch
  - Resolve any merge conflicts (if any)
  - Remove the worktree after successful merge
- Let me start by checking the current status:

\* Booping... (13s · 344 tokens · esc to interrupt)

# 약간의 팁 iterm + tmux

- Git worktree 등으로 여러개의 브랜치
  - 동시에 작업위해 여러개의 창을 띄워야 한다는 상황
  - 거기다 서버에 붙어서 작업하게 되는 경우
  - 터미널 유실로 인한 단절이 자주 일어남
- 때문에 screen이나 tmux 등을 쓰게 됨
- 만약 iterm을 쓴다면 tmux -CC 명령을 통해
  - Iterm 과 tmux가 연동되어, tmux 명령 기억하지 않아도
  - 네이티브 터미널 처럼 쉽게 쓸 수 있음. 강추
  - 내 노트북 일하다 던져버려도 서버 터미널 세션 쉽게 복원 가능

# 어쨌거나 클로드코드의 핵심 : Claude.md

- 작업할 때마다 클로드가 참고하는 메모파일. (Cursor의 rule에 해당)
- 프로젝트 전체, 또는 하위 폴더마다 따로 여러개 있어도 된다.
  - ~/proj
  - ~/proj/module\_a, ~/proj/module\_b
- 시스템와이드로 설정하려면
  - ~/.claude/CLAUDE.md
- 보통 기존 코드베이스 대상으로 작업하게 되면
  - /init 명령으로 자동 전체 소스파악해서 CLAUDE.md 만들게 하는게 국룰

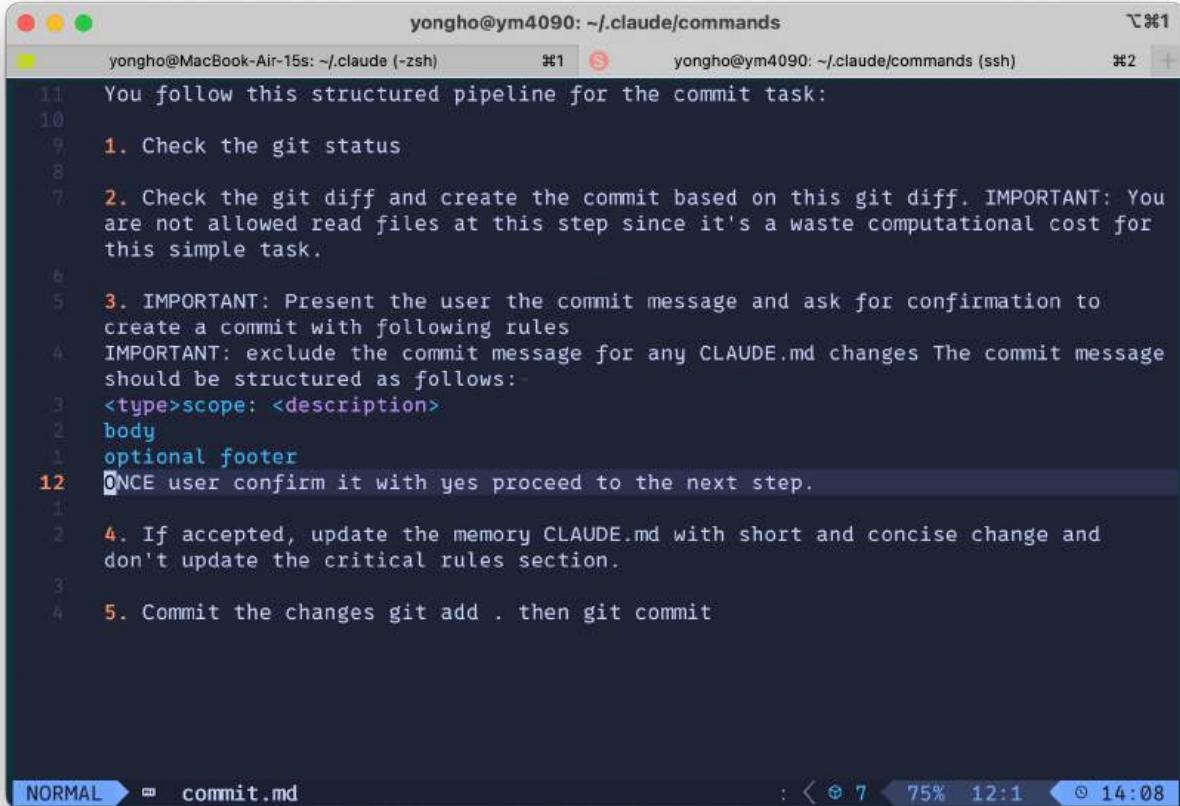
# Claude.md 운영 (/init 말고 추가해야 할 것들)

- 작업하다보면 내 의도와 다르게 하는 삽질들이 자주보임
  - 가장 자주 하는 실수는 패키지 관리 (pip vs uv)
  - 또는 버전을 바꾸어서 문제를 해결하려는 시도
    - AI에이전트를 만드는데 나는 gemini-flash-2.5 모델을 쓰고 싶은데, 원가 잘 안된다고 코드를 바꾸는게 아니라 모델을 gemini-flash-2.0 을 쓴다던가
    - Claude가 학습할 때 사용했던 구버전의 문법을 자꾸 쓴다던가.
- 이런 요소들을 꼭 CLAUME.md 에 추가해줘야 빡침 방지
  - LLM은 하지말라보다, 하라를 잘 듣는 경향이 있음. ‘금지 + 바른방향’ 함께 기술
  - ‘문제를 해결하기 위해 사용하는 LLM의 버전을 바꾸지 마라.  
바꿔야할 일이 있으면 작업을 중지하고 사용자에게 확인해라’
  - **# The Golden Rule**
    - When unsure about implementation details, **ALWAYS ask the developer.**

# Claude.md 에 테스트코드 변경가능 여부 명시하기

- 테스트 코드를 건드는 건 귀찮으므로 클로드코드를 시킬 수 있다.
- 하지만 여기는 진짜 눈으로 변경을 전수 검사를 해야 한다.
- 이상한 트릭을 잘쓴다.
  - 테스트를 실패한다고?
    - 그럼 그 테스트를 없애면 되겠군!
  - 점수가 낮다고?
    - 기준점을 낮추면 되겠군!
- 코드는 건드리지 말고 테스트 코드만 짜라 vs 테스트 코드는 건드리지 말고 코드는 짜라고 명시
  - ‘절대 테스트 코드는 건들지마’
  - ‘절대 API 이름과 파라미터는 바꾸지마’
  - ‘절대 자의적으로 데이터를 마이그레이션 하지마’
  - 위의 정도면 안전
- 테스트코드가 잘 확립되면, claude.md 에서 제한을 풀고 리팩토링을 맡겨도 잘함.

# claude.md 최신화: Commit과 claude.md 변경을 묶자



The screenshot shows a terminal window with two tabs. The active tab displays the help documentation for the 'commit' command in 'claude.commands'. The text describes a structured pipeline for committing changes:

- 1. Check the git status
- 2. Check the git diff and create the commit based on this git diff. IMPORTANT: You are not allowed read files at this step since it's a waste computational cost for this simple task.
- 3. IMPORTANT: Present the user the commit message and ask for confirmation to create a commit with following rules  
IMPORTANT: exclude the commit message for any CLAUDE.md changes. The commit message should be structured as follows:-  
<type><scope>: <description>  
body  
optional footer
- 12. ONCE user confirm it with yes proceed to the next step.
- 4. If accepted, update the memory CLAUDE.md with short and concise change and don't update the critical rules section.
- 5. Commit the changes git add . then git commit

The bottom status bar shows the file is 'commit.md' and the current line is '7'.

- 지금 작업한 최신 내용을 claude.md에 반영해 두기가 귀찮음
  - (아니 언제 또 따로 편집하나)
- 커밋할 때, 필요가 있다면 자동으로 변경하게 해두면 좋음
- 스샷의 4번 항목
- /commit 하면 git commit하고
- 그 변경사항 맞춰서 claude.md도 업데이트

> /commit is running...

- Bash(git status)
  - └ 현재 브랜치 : r...  
커밋 하도록 정한 커밋을 커밋  
... +19 lines (ctrl+r to expand)

- Bash(git diff --staged)
  - └ (No content)

- Bash(git diff)

```
diff --git a/CLAUDE.md b/CLAUDE.md
index d285f74..f4a9668 100644
--- a/CLAUDE.md
+++ b/CLAUDE.md
@@ -1,10 +1,10 @@
# Claude
```

- 다음 커밋 메시지로 커밋 하시겠습니까?

```
feat(sql-generation): LLM 기반 SQL 쿼리 생성 시스템 구현
  - 사용자 의도를 기반으로 관련 쿼리를 찾는 QueryFinder 클래스 추가
  - 찾은 쿼리들을 참조하여 Athena SQL을 생성하는 SQLGenerator 클래스 구현
  - query_intentions 테이블의 중복 제거를 위한 export 스크립트 개선
  - find-queries와 generate-sql CLI 명령어 추가
  - 의도 분석 시 redash_id 표시로 사용자 친화성 개선
  - 프롬프트와 생성된 SQL을 타임스탬프 기반 로그 파일로 저장
```

#### 주요 기능 :

- Gemini 2.5 Flash/Pro 모델을 활용한 쿼리 검색 및 SQL 생성
- structured output을 통한 안정적인 JSON 응답 처리
- 대규모 프롬프트 구성으로 참조 쿼리의 전체 SQL과 의도 정보 활용
- Athena SQL 문법 준수 및 충분한 주석이 포함된 SQL 생성

yes로 응답 하시면 커밋을 진행 하겠습니다.

> yes

# /Commit 메시지 자동 만들면서

> yes

- Update(CLAUDE.md)
  - └ Updated CLAUDE.md with 3 additions  
101

# Claude.md 도 자동으로 업데이트

Tip: Use # to memorize shortcut to quickly add to CLAUDE.md

- Update(CLAUDE.md)

```
Updated CLAUDE.md with 1 addition and 1 removal
```

```
72
73    ## CLI Commands
74
```

```
75 - The main interface is through 'main.py' with 9 commands
for the complete analysis pipeline:
```

```
- the complete analysis pipeline:
```

```
75 + The main interface is through 'main.py' with 10 commands
for the complete analysis pipeline:
```

```
+ for the complete analysis pipeline:
```

```
76
77     ```bash
78     # Verify all API connections
```

Tip: Use # to memorize shortcut to quickly add to CLAUDE.md

- Update(CLAUDE.md)

```
Updated CLAUDE.md with 1 addition
```

```
14 - 불필요한 웹 대시보드, 리포트, 피드백 시스템 제거
15 - Progressive 분석 : 이전 분석의 의도와 추론 과정을 hint로
활용하여 일관성 있는 개선된 분석 수행
16 - 실시간 분석 진행상황 표시 : 스트리밍을 통한 쿼리별 완료
상태 및 경과 시간 모니터링
```

```
17 + - **SQL 생성 기능 추가**: 사용자 의도 기반 관련 쿼리 검색
+ 및 Athena SQL 자동 생성
18
19     ### Architecture
20
```

Tip: Use # to memorize shortcut to quickly add to CLAUDE.md

- Bash(add .)

# Kimi k2 + groq + claude

- 클로드코드는 마치 인터넷 회선 같은 요금제
  - \$20짜리는 일정량 이상 쓰다보면, 제한 풀릴때까지 기다리라 함
  - 더 비싼 요금제 \$100, \$200 으로 올려도 또 제한 걸릴 때 있음
  - 한명이 N개의 병렬 작업하는데, M명이 동시에 쓰면 더 빨리.
  - 물론 클로드계정을 여러개 더 사는 방법도 있고
- kimi k2같이 클로드와 비슷한 성능의 오픈소스 모델을
  - 본인들 서버에서 초고속으로 서빙하는 groq같은데서 토큰등록해서
  - 프로그램은 여전히 claude code지만 LLM을 kimi 등으로 바꿀 수 있음
  - Groq (grok과 다른)은 LLM을 칩에 태워서 바로 쓰는 곳이라 초고속
  - 난이도 낮은 작업에 대해서는 더 빠른 속도로 할 수 있음 (5분->1분)
- 관심있는 분은 <https://github.com/musistudio/clause-code-router>

# 프론트코딩은 playwright로

- 이런저런 도구들이 backend 개발에 좀 더 쉽게 적용이 가능해보이는 것도 사실.
- 하지만 playwright mcp 등을 통해서 프론트개발과 디버깅도 claude code에게 맡길 수 있음
  - 다만 좀 자세한 가이드를 claude.md에 추가하고
  - 일시킬때도 좀 더 명시적으로 시켜야 한다.
- 서버 사이드에서 돌릴때는 headless 모드로 돌게 명시하기

1. `show_history()` 함수 :
  - `show_analysis_history()`에서 여전히 호출됨
  - 완전 제거 전 대체 구현 필요
2. `load_inference_data()` 함수 :
  - 일부 기존 페이지에서 여전히 사용 중
  - 단계적 제거 필요

현재 상태는 안전하게 통합된 상태이며, 추가 정리를 원한다면 위의 미사용 코드들을 단계적으로 제거할 수 있습니다. 하지만 기능상 문제는 전혀 없는 상태입니다.

> 현재의 대시보드를 Playwright를 통해서 제대로 잘 동작하는지 검증해 줘 .

- Playwright를 통해 대시보드가 제대로 동작하는지 검증해보겠습니다.

● `playwright - start_codegen_session (MCP)(options:`  
`{ "outputPath": "/home/yongho/work/spacevision_sql_agent/tests", "testNamePrefix": "DashboardTest", "includeComments": true } )`

```
└ {  
  "sessionId": "7ae6c164-b2c3-4b34-8da6-68bcadab7af0",  
  "options": {  
    ... +8 lines (ctrl+r to expand)  
  }
```

- 먼저 대시보드를 백그라운드에서 실행하겠습니다 :

● Bash(`python main.py web --port 8503 --host localhost &`)  
└ Error: Command timed out after 10s  
  🌐 웹 인터페이스를 시작합니다 ...  
  📍 주소 : <http://localhost:8503>  
  ⚠️ 종료 하려면 Ctrl+C를 누르세요 .

Collecting usage statistics. To deactivate, set  
browser.gatherUsageStats to false.

You can now view your Streamlit app in your browser.

... +1 line (ctrl+r to see all)

- 좋습니다 ! 대시보드가 실행되었습니다 . 이제 Playwright로 테스트해보겠습니다 :

# MCP팁

- 이런 MCP쓰다보면
  - Claude code에서 승인을 너무 자주 눌러야 한다.
- 프로젝트 단위보다 유저 단위로
  - ~/.claude/settings.json 을 관리
  - 여기서 영구적 승인을 해주자.
- Mcp 설치
  - 늘 쓰는 mcp는 자주 쓰게 된다.
  - 설치할 때도 유저단위로 설치해서 재설치 회피
- Context7은 리모트로 설치하면 자주 끊김
  - 속편하게 로컬모드로 설치하는게 차라리.

```
{  
    "permissions": {  
        "allow": [  
            "Bash(git:*)",  
            "Bash(ls:*)",  
            "Bash(cat:*)",  
            "Bash(uv:*)",  
            "Bash(source:*)",  
            "Bash(npm run lint)",  
            "Bash(npm run test:*)",  
            "Bash(steamlit:*)",  
            "Read(~/.zshrc)",  
            "Bash(playwright:*)",  
            "mcp__playwright",  
            "mcp__context7"  
        ]  
    }  
}
```

# 제가 쓰는 system wide claude.md 일부

```
20 ## MCP playwright
19 - always run in headless mode
18
17 ## Key Rules
16 1. Always use uv for package management operations
15 2. Never use pip install directly - use uv add instead
14 3. Keep .venv in the project root directory
13 4. Ensure virtual environment is activated before running Python code
12 5. 깃 험에서 풀 리퀘스트(PR) 또는 커밋 메시지를 작성할 때는 반드시 한국어로 작성한다.
11
10 ## Data Extraction Strategies
9 - 텍스트 덩어리에서, 예를 들어 SQL 다발에서 특정 정보를 뽑아 낼 때는 직접 문법을 통해
8 파싱하기보다는 가벼운 LLM 모델, 예를 들어 Gemini Flash 등을 이용하여 필요한 정보를 추출하는
7 방식으로 이용해 줘요.
6
5 ## API Documentation
4 - 문서의 api는 context7을 통해 최근 DOCUMENT를 참조해서 사용하도록. 특히 redash와 Google
3 Gemini API 관련해서 반드시 참조
2
1 ## Structured Output
3 - 재미나이 API의 STRUCTURED OUTPUT은 https://ai.google.dev/gemini-api/docs/structured-
2 output?hl=ko 이 DOCUMENT를 참조하여 IMPLEMENT 한다.
1
1 ## LLM Model Usage
52 ■ 사용하는 LLM모델은 2.5 모델만 사용한다. 절대 자의적으로 아래 버전으로 내리지 않는다.
51 문제가 생겨도 모델 버전을 바꾸어 해결하는 법은 시도하지 않고 유저에게 에러를 리포트 한다.
1
2 ## Gemini API Notes
3 - gemini의 토큰은 2M에 달하므로 토큰 제한이 있다면, 입력 토큰이 아니라 출력 토큰인
max_output_tokens 등의 설정의 문제일 가능성은 많다.
NORMAL ➞ CLAUDE.md gj < ⌂ 7 94% 52:1 ⌂ 14:37
```

AI의 도움을 받아  
데이터 분석 하기

# 분석의 자동화

- 데이터의 사례
- 사용자가 slack에서 데이터봇에게 멘션하면
- 쿼리를 작성하고
- 사내 데이터분석툴인 redash에 페이지를 만들고
- 링크도 걸어준다.

5개의 댓글



Seorim Lee (이서림) 어제 12:23



@DableTalk (dtalk) KST기준, 2025-07-14 부터 2025-07-20까지 7일동안 한국에서 가장 많이 클릭된 광고 컨텐트를 알려줘.



DableTalk (dtalk) 앱 어제 12:23

@Seorim Lee (이서림) 님의 질문에 대한 SQL을 생성하고 있습니다.

생성에 시간이 걸릴 수 있으니, 잠시만 기다려주세요!

Generating SQL for your question.

It may take some time, so please wait a moment!

Question: KST기준, 2025-07-14 부터 2025-07-20까지 7일동안 한국에서 가장 많이 클릭된 광고 컨텐트를 알려줘.

@Seorim Lee (이서림) 님의 질문에 대한 모든 작업이 완료되었습니다.(All tasks for the question are completed.)

Question: KST기준, 2025-07-14 부터 2025-07-20까지 7일동안 한국에서 가장 많이 클릭된 광고 컨텐트를 알려줘.

Redash Query: <https://redash.dable.io/queries/9852/source>

SQL Query:

```
SELECT
    t1.content_id,
    SUM(t1.clicks) AS total_clicks
FROM
    fact_hourly.ad_summary AS t1
JOIN
    (SELECT * FROM dimensional_reco.dable__service WHERE utc_basic_time = 'latest') AS t2
    ON t1.service_id = t2.service_id
WHERE
    t1.utc_basic_time >= '2025-07-13-15'
    AND t1.utc_basic_time < '2025-07-20-15'
    AND t2.country = 'KR'
GROUP BY
    t1.content_id
ORDER BY
    total_clicks DESC
LIMIT 10
```



8



# 좀 더 멋진 자동화

# data chat

- 오늘 발표에도 계심
  - 필요한 분석 사항을 이야기하면
  - 쿼리하고 추출하고
  - 그 데이터를 가장 잘 나타내는 방법으로 비주얼라이즈까지 자동
  - 필요한 비즈니스 정의도 직접 하며 할 수 있다.

알고 싶은 내용을 입력하세요



A screenshot of a web browser window. The address bar shows 'localhost:58815'. The main content area features a large title 'data chat' followed by a small blue robot icon. At the bottom, there is a red-bordered box containing Korean text: '가입하고 골드를 구매한 유저들은, 가입하고 골드를 안산유저들보다 더 잔존율이 높을까? 잔존율은 가입후 2주가 지나도 여전히 로그인하고 있는 것으로 할게.' A black right-pointing arrow is located at the bottom right of this box.

데이터 분석 어시스턴트

localhost:58815

RUNNING... Stop Deploy

응답 생성 중...

## 생성된 SQL 쿼리

```
-- [Step 1] 2022년 이후(이상) 가입자만 대상으로 분석합니다 (데이터 오류 방지)
-- [Step 2] 골드 구매자, 비구매자 구분
-- [Step 3] 가입 후 2주 이후에도 로그인 한 적 있으면 잔존자로 집계

WITH base_users AS (
    -- 2022년 이후 가입자만, 인덱스 활용 & 한국시간 처리
    SELECT id, created_at + INTERVAL '9 hours' AS created_at_kst
    FROM users
    WHERE created_at >= '2022-01-01'::date - INTERVAL '9 hours'
),
gold_buyers AS (
    -- 골드 구매 기록이 있는 user_id
    SELECT DISTINCT t.user_id
    FROM transactions t
    JOIN base_users u ON t.user_id = u.id
    WHERE t.type = 'gold_purchase' AND t.deleted_at IS NULL AND t.user_id IS NOT NULL
),
```

알고 싶은 내용을 입력하세요 ➤

데이터 분석 어시스턴트

localhost:58815

```
LEFT JOIN login_after_2w lazw ON g.user_id = lazw.user_id
GROUP BY user_group
ORDER BY user_group;
```

Deploy :

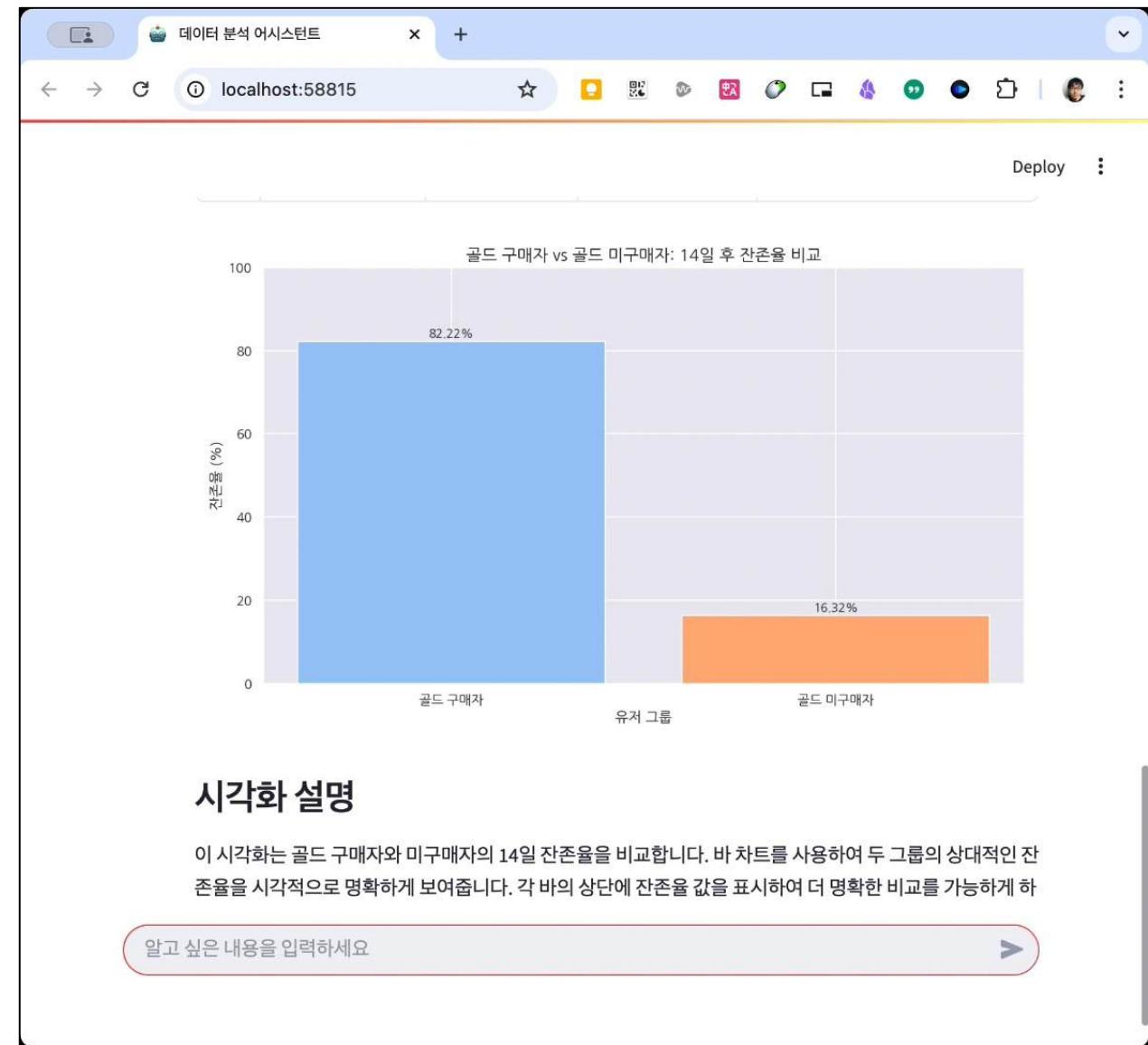
## 설명

이 쿼리는 2022년 이후에 가입한 모든 유저를 대상으로, '골드 구매 유저'와 '골드 미구매 유저'를 구분합니다. 그리고 각 그룹에서 가입 후 2주가 지난 시점에도 한 번 이상 로그인한 경험이 있는 유저를 잔존자(14일 retention)로 산정하여, 그룹별 전체 잔존율(%)을 비교합니다.

- base\_users: 가입자 중 2022년 이전 데이터 오류 방지
- gold\_buyers: 골드 현금 구매 트랜잭션 기록 여부로 구분
- login\_after\_2w: 가입일로부터 14일 이후에도 로그인한 경우만 추출
- grouped: 골드 구매 여부 플래그화
- 마지막 SELECT에서 그룹별 전체유저/잔존자 수/잔존율(%)를 출력하며, 모두 DuckDB KST 및 인덱스 친화적으로 작성하였습니다.

	user_group	total_users	retained_users	retention_rate_14d_percent
0	골드 구매자	10555	8678	82.22
1	골드 미구매자	874188	142672	16.32

알고 싶은 내용을 입력하세요 ➤



AI의 도움을 받아  
스터디 하기

**개발자란 끊임없이 공부하는 직업**

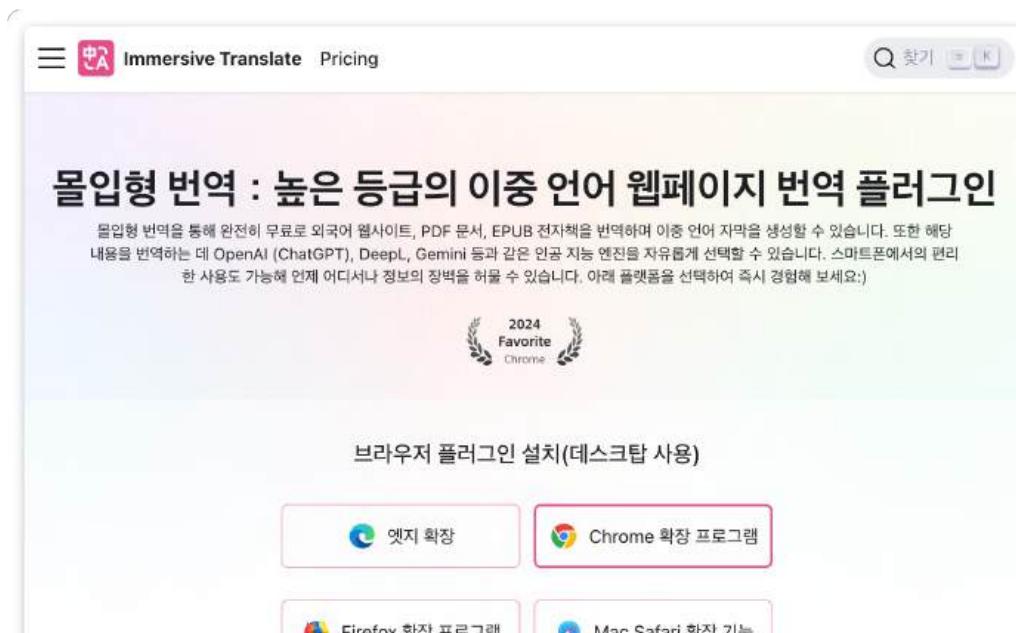
= **끊임없이 영어를 계속 봐야하는 직업**

쇼츠와 도파민에 절여진 뇌는 영어글을 볼 인내력을 잃었다. ADHD인가?



# 외국어로 된 개발 문서 읽을 때

- 브라우저에서 번역도 해주기도 하지만, 잘 번역되었는지 의심
- 그렇다고 영어 원문을 보는 것은 시간이 오래 걸린다.
- 그럴 때! “Immersive translate” Extension



# Causal ML for Root Cause Analysis

## 근본 원인 분석을 위한 인과 관계 ML



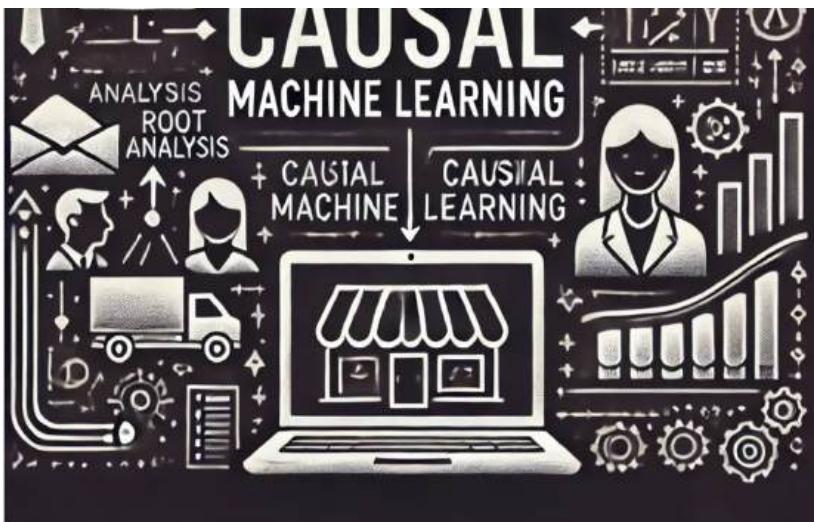
Ryuta Yoshimatsu · [Follow](#)  
12 min read · Feb 10, 2025

167

2

···


# 영한대역으로 번역해준다!



Generated by ChatGPT 4o. ChatGPT 4o에 의해 생성됨

## Root Cause Analysis in Industry

### 산업 분야의 근본 원인 분석

Root cause analysis is crucial in various industrial settings. Manufacturers use it to identify the underlying issues behind defects, inefficiencies, and failures in their production lines. Procurement teams use it to pinpoint bottlenecks in their supply chain, uncovering the reasons behind delays in lead times. By accurately diagnosing causes, organizations can implement targeted solutions to prevent recurrence, enhance quality and streamline operations.

근본 원인 분석은 다양한 산업 환경에서 매우 중요합니다. 제조업체는 생산 라인의 결함, 비효율성, 고장의 근본적인 문제를 파악하는 데 근본 원인 분석을 사용합니다. 조달 팀은 이를 사용하여 공급망의 병목 현상을 정확히 파악하고 리드 타임 지연의 원인을 밝혀냅니다. 원인을 정확하게 진단함으로써 조직은 재발을 방지하고 품질을 개선하며 운영을 간소화하기 위한 맞춤형 솔루션을 구현할 수 있습니다.

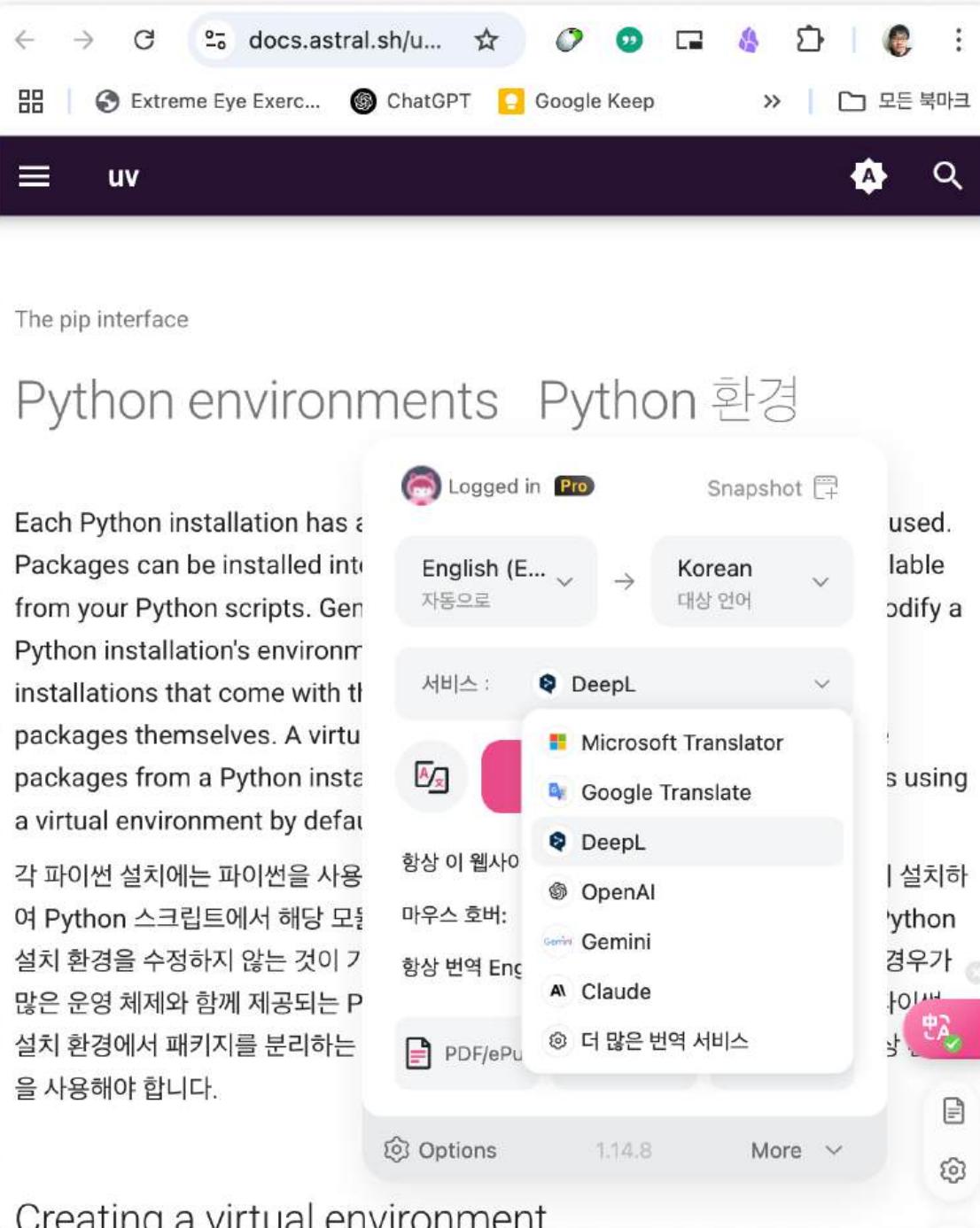
However, this process is far from simple. Modern production lines are highly complex, consisting of numerous interdependent processes where hundreds of factors influence the final product's quality. Supply chains are often global, involving multiple vendors with varying terms and conditions. External factors such as geopolitical events or adverse weather conditions can further disrupt distribution. Given these complexities, accurately identifying the true root cause of a problem remains a significant challenge.

하지만 이 과정은 결코 간단하지 않습니다. 현대의 생산 라인은 매우 복잡하며, 수백 가지 요소가 최종 제품의 품질에 영향을 미치는 수많은 상호 의존적인 프로세스로 구성되어 있습니다. 공급망은 다양한 조건을 가진 여러 공급업체가 참여하는 글로벌 공급망인 경우가 많습니다. 지정학적 사건이나 악천후와 같은 외부 요인으로 인해 유통에 더 큰 차질이 발생할 수 있습니다. 이러한 복잡성을 고려할 때 문제의 진정한 근본 원인을 정확하게 파악하는 것은 여전히 중요 한 과제입니다.

## Challenges with traditional machine learning models

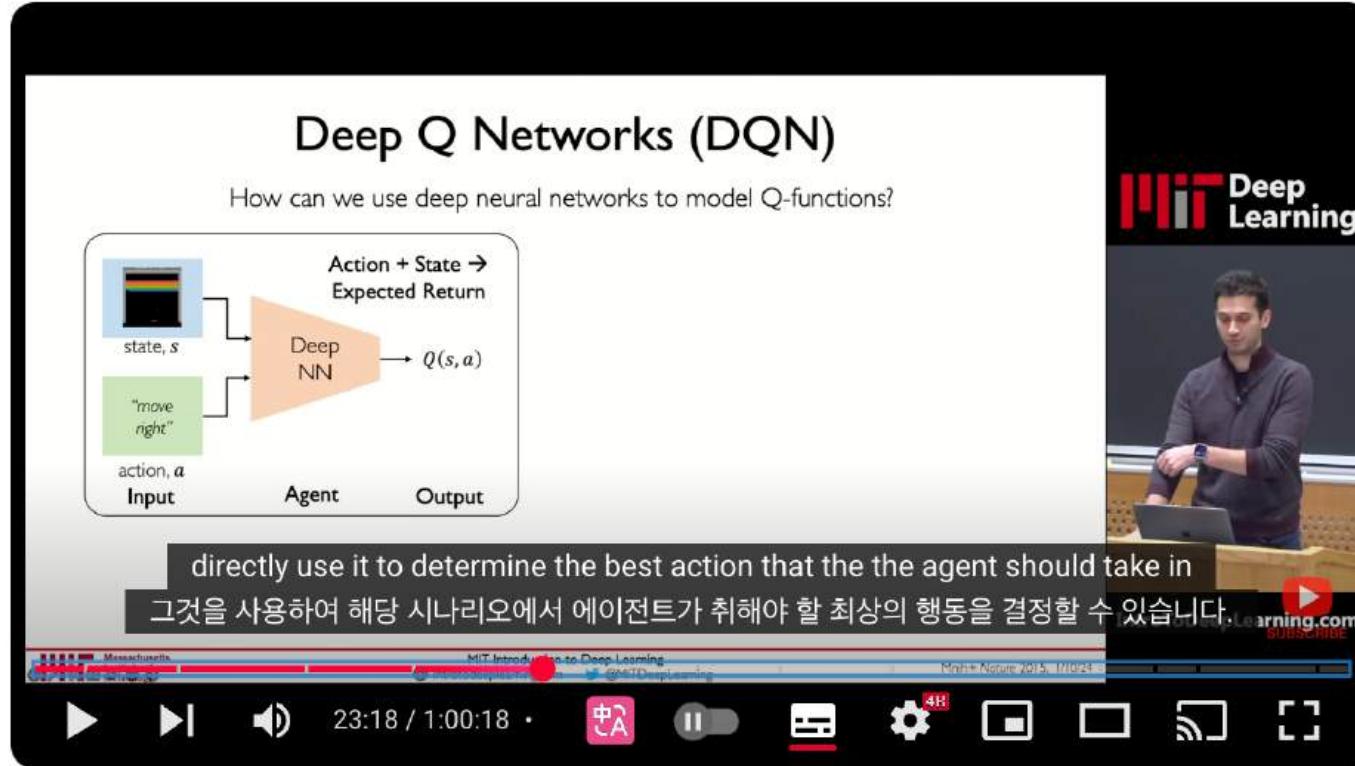
### 기존 머신러닝 모델의 문제점

With the rise of machine learning (ML) and AI, many industrial practitioners



- 번역 엔진을 고를 수 있음
- 무료인 구글 번역도 충분히 좋고
- DeepL이나 GPT, Claude, Gemini 등 LLM을 번역 엔진으로 쓸 수 있어 아주 매끄러운 번역이 가능
  - 어떤 페르소나 입장에서 번역할지 설정 가능
  - IT글을 IT전문가 입장에서 번역하라던가,
  - 금융글을 월가 전문가 입장에서 번역하라던가.
  - 전문 용어의 이상한 번역을 막을 수 있음

# 가장 좋은 기능은 youtube 동사자막



자막이 없는 영어 컨텐츠에 영한대역 자막을 추가해줌  
저는 주로 외국 세미나 자료들 볼 때 씀

**하지만 더 시간을 절약하는 방법은?**

**이게 읽을 만한 컨텐츠인가?**



## About Us

## Nikolas Laskaris

- Research Scientist @ Yale
- Founded an education startup
- Founding team member @ Comet
  - Data Science → built all GTM



MIT Deep Learning



IntroToDeepLearning.com

## Douglas Blank

- Professor Emeritus @ Bryn Mawr College
- Research into Deep Learning and Robotics
- Founding team member @ Comet
  - Head of Research



comet

NCO um I'm on the the founding team of comet started as a research scientist

NCO입니다. 저는Comet의 창립 팀에 속해 있습니다. Elel에서 연구 과학자로 시작했습니다.



채팅 다시보기 표시

## MIT 6.S191: Building AI Models in the Wild



Alexander Amini

구독자 28.7만명

구독

541



유튜브 요약은 livewiki 확장

LiveWiki

KO :

영상 요약하기

모두

시리즈

Alexander Amini 제공

추천



MIT Introduction to Deep Learning (2023) | 6.S191

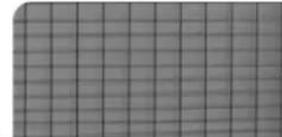
Alexander Amini

조회수 202만회 · 1년 전

Shorts

C.U.D.A Explained

by NVIDIA



How AI-assisted coding will c X +

newsletter.pragm... ☆

The Pragmatic Engineer

Subscribe Sign in

DEEPPDIVES

# How AI-assisted coding will change software engineering: hard truths

A field guide that also covers why we need to rethink our expectations, and what software engineering really is. A guest post by software engineer and engineering leader Addy Osmani

GERGELY OROSZ AND ADDY OSMANI  
JAN 06, 2025

897 10 125 Share

*Hi, this is Gergely with a bonus issue of the Pragmatic Engineer Newsletter. In every issue, we cover topics related to Big Tech and startups through the lens of software engineers and engineering leaders. To get articles like this in your inbox, every week, subscribe:*

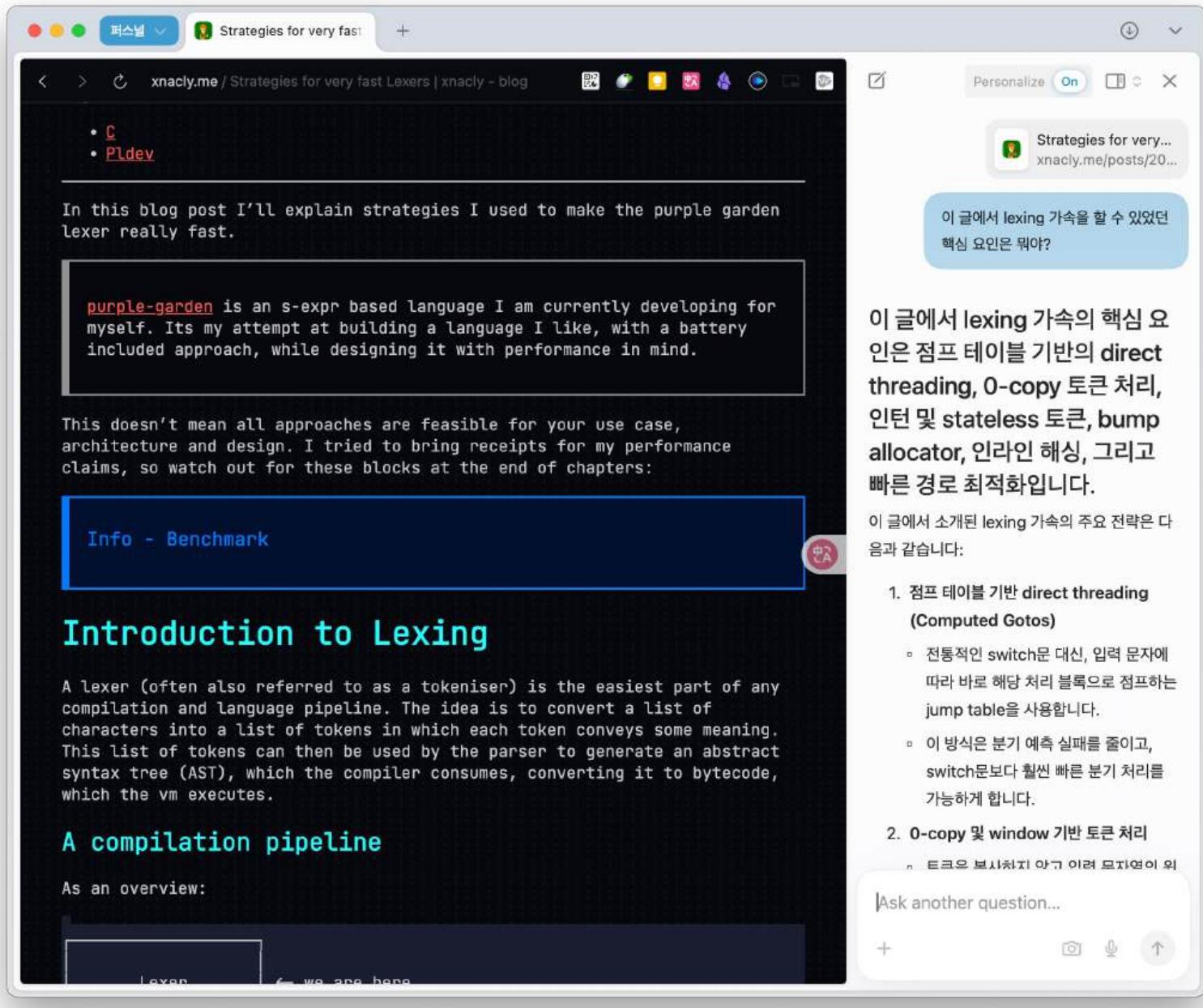
Type your email... Subscribe

Happy New Year! As we look toward the innovations that 2025 might bring, it is a sure bet that GenAI will continue to change how we do software engineering.

It's hard to believe that just over two years ago in November of 2022 was ChatGPT's first release. This was the point when large language models (LLMs) started to get widespread adoption. Even though LLMs are built in a surprisingly simple way, they produce impressive results in a variety of areas. Writing code turns out to be perhaps one of their strongest points. This is not all that surprising, given how

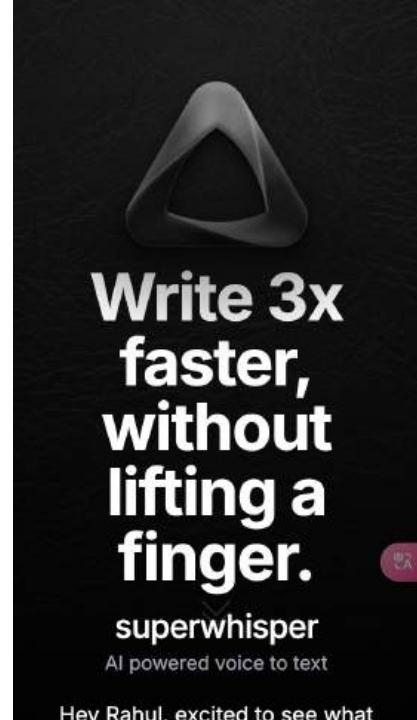
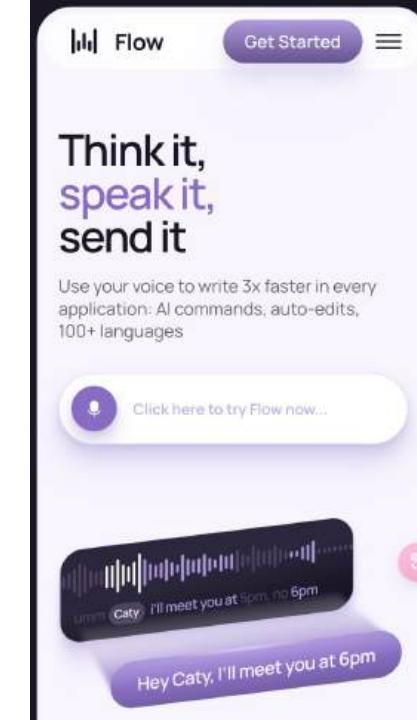
# Dia 브라우저

- 컨텐츠를 보며
- GPT와 함께 대화를 할 수 있다.  
(추정모델 gpt o4-mini)



# 입코딩의 완성은 보이스 딕테이션

- 입으로 녹음해서 글을 쓰는 편이 진짜로 편하다.
  - 우리는 이제 키보드를 두드릴 체력도 모자라다.
- Wispr Flow
  - 개인적으로는 wisp flow 추천
- Superwhisper
  - 좀 더 세팅이 많지만,
  - 생각보다 좀 잘 동작하지 않는 느낌도 있다.





mcptest [SSH: ym4090e]



pyproject.toml X

Dockerfile X

CHAT

COMPOSER



## OPEN EDITORS

X pyproject.  
Dockerfile  
test\_main  
main.py a  
requirements

## MCPTEST [SSH: YM

.cursor/rules  
custom-rule.r  
app  
main.py  
tests  
Dockerfile  
pyproject.toml  
requirements.t

## NOTEPADS

X Update Cursor?  
Read the changelog.



Home

Dictionary

History

Settings

Get Flow Pro

Refer a friend

Version 1.2.13

Good evening, Yong-Ho

Hold down fn + shift and speak into any textbox

Weekly streak

1st week

You are off to a great start!

Average Flowing speed

90 WPM 🌟

Faster than 97% of typers

Total words dictated

5,959 🔥

You've written 11 yelp reviews!

You are all set!

5/5 steps completed

- Send a message
- Respond to an email
- Ask AI a question
- Use Flow hands-free for longer dictations
- Learn about whispering mode

Flow Wrapped

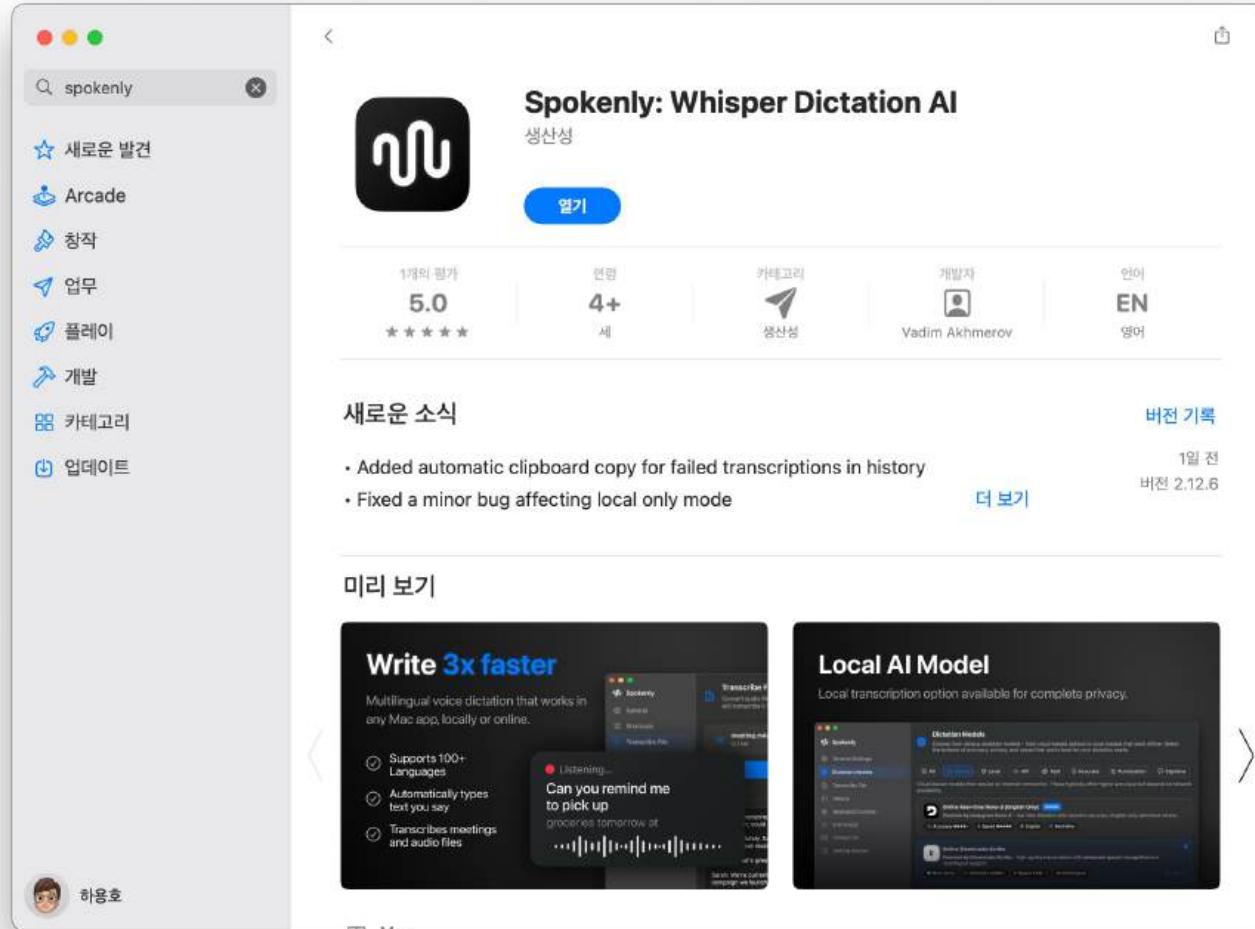
2024

Check out your Flow Wrapped

View your Wrapped

Past composers &gt;

# 최근에 가장 애용은 spokenly



- 최근에 써본 것들 중에 그래도 제일 인터페이스가 깔끔한 듯.
- 로컬엔진으로 쓰면 완전 무료 (한글 성능은 약간 떨어짐)
- 엔진을 gpt 4o mini transcribe를 선택하고, 회사가 쓰고 있는 OpenAI key같은 것을 쓰면, 월사용료 내는것 보다 더 저렴하게 가장 높은 정확도로 쓸 수 있다.

이렇게 AI가 다해주면  
이제 우린 잘리나요?

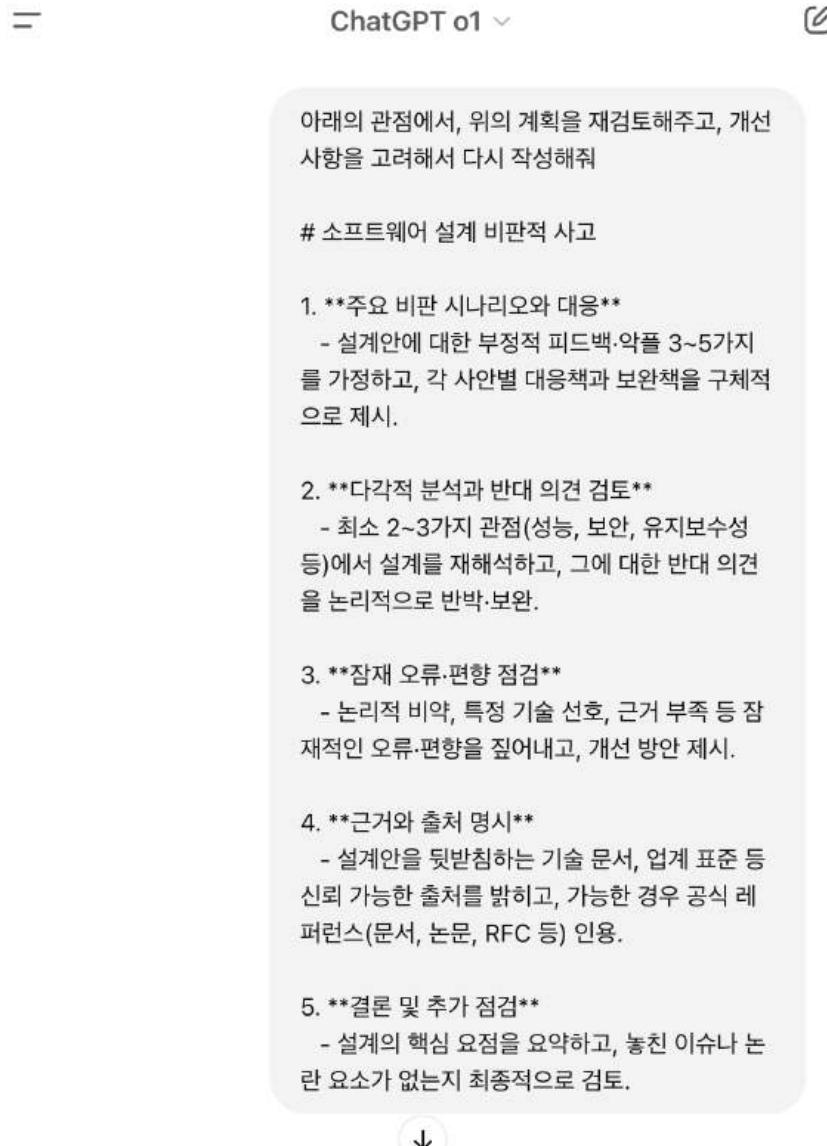
# 2025년의 10x 개발자



# 우리는 할 일이 없는가?

- 우리는 다들 훌륭한 시니어처럼 행동해야 한다.
- 이건 이렇게 잘못되었으니, 이걸 이렇게 개선해야 한다.
- 시어머니가 되라!
  - (어머니 죄송합니다)
- AI가 얼개는 잘 잡아도 논리오류는 잘 못잡는다.
  - 결국 코드를 면밀히 읽는 능력은 여전히 필요
  - 다만 이 과정도 LLM을 최대한 활용하면 좋음

# 근데 사실 주책맞은 관리도 LLM이 도울 수 있음



- 악플을 요청한다.
- 다른 관점을 요청한다.
- 잠재적인 오류를 추정하라고 한다.
- 근거와 출처를 요청한다.
- 놓친 이슈를 요청한다.
- 위 중 가장 효과가 좋은 건 악플요청
  - 어디서 많이 느껴본? 내 상사인가?!

1. “Nasdaq 100 티커를 웹 스크래핑으로 가져오는 건 너무 불안정해 보이는데?”
2. “2개월 치 데이터만 보면 단기 변동만 볼 뿐인데 신뢰도가 떨어지지 않을까?”
3. “yfinance API가 언제 어떻게 변할지 모르는데, 너무 의존적이야.”
4. “Streamlit만 쓰면 서버-클라이언트 분리가 제대로 안 된 거 아닌가?”
5. “과열 판단을 단순 5일 15% 상승률로만 하는 건 너무 단편적이다.”
  - 비판 내용: 단일 임계값(15%)으로 과열 여부를 판단하는 것은 시장마다, 업종마다 편차가 큼.
  - 대응책:
    - MVP에선 간단히 적용하되, 사용자가 직접 임계값을 입력하거나, RSI·볼린저밴드 등 추가 지표를 함께 고려 가능하도록 구조화.
    - 보완: 기술 지표 라이브러리(`ta` 등) 연동을 용이하게 만들어 다양한 필터 조건을 개발자가 추가할 수 있게 한다.

**앞선 예제 프로젝트도  
넉넉하게 힘난해 주셨습니다**

February 18, 2025 Publication

## Introducing the SWE-Lancer benchmark

Can frontier LLMs earn \$1 million from real-world freelance software engineering?

[Read paper ↗](#)[Access repository ↗](#)

▶ Listen to article 1:04

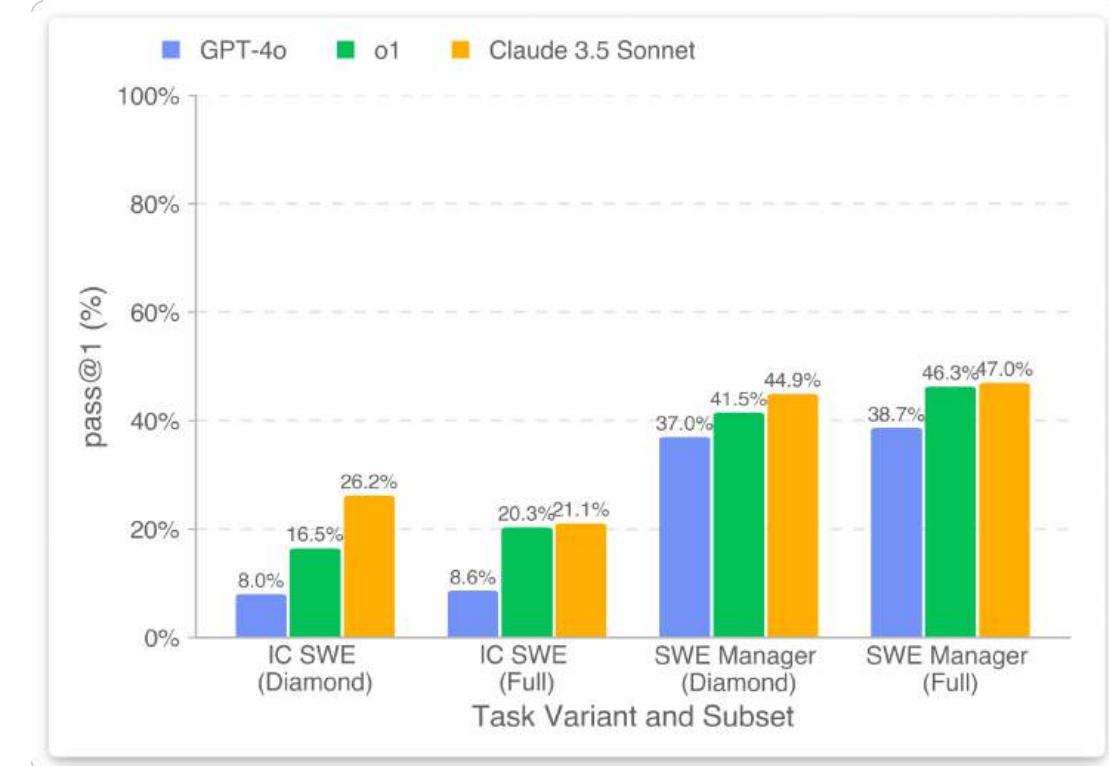
🔗 Share

We introduce SWE-Lancer, a benchmark of over 1,400 freelance software engineering tasks from Upwork, valued at \$1 million USD total in real-world payouts. SWE-Lancer encompasses both independent engineering tasks — ranging from \$50 bug fixes to \$32,000 feature implementations — and managerial tasks, where models choose between technical implementation proposals. Independent tasks are graded with end-to-end tests triple-verified by experienced software engineers, while managerial decisions are assessed against the choices of the original hired engineering managers. We evaluate model performance and find that frontier models are still unable to solve the majority of tasks. To facilitate future research, we open-source a

# 이제 우린 짤리나요?

- 2025/02/18, OpenAI의 페이퍼
- Upwork
  - IT 프리랜서 작업 올리고, 따와서 수행
  - 크몽? 같은 것
  - 여기 과제를 LLM이 수행해봄
  - 전체 과제중 일부를 추려서
  - 100만달러 어치의 과제들을 선별. 시켜봄
- 할만했나?

# 머신의 성능은?



- 과제에 걸린 돈을 다 따오지는 못했다!
- 첫시도(pass@1)에 성공확률은 아직 낮음

# 업무 분야별 과제들의 LLM의 성공률

Table 2. Diamond Task Pass Rates by Task Type

Task Type	IC SWE				SWE Manager			
	GPT-4o	o1	Sonnet 3.5	n	GPT-4o	o1	Sonnet 3.5	n
Application Logic (Client-Side)	8.0%	15.9%	23.9%	176	36.3%	42.3%	45.8%	201
UI/UX	2.4%	17.1%	31.7%	41	32.7%	32.7%	40.8%	49
Server-Side Logic	23.5%	23.5%	41.2%	17	53.8%	61.5%	38.5%	13
System-Wide Quality and Reliability Tasks	0.0%	0.0%	0.0%	3	100.0%	50.0%	100.0%	2

- 소프트웨어 엔지니어로서는(IC SWE) 아직 수준이 모자람
  - 서버는 좀 하지만, 아직 모자람
  - 우리가 함께 코딩하며 잘 도와서 이끌자
- 뜻밖에 테크리드(SWE Manager)로는 어느정도 합니다.
  - 고민을 함께 잘 나눕시다.
  - LLM들의 코멘트를 보면서 나도 함께 성장합시다.

# AI를 쓰는게 정말 효율적인가?

The Cybernetic Teammate: A Field Experiment on Generative AI Reshaping Teamwork and Expertise

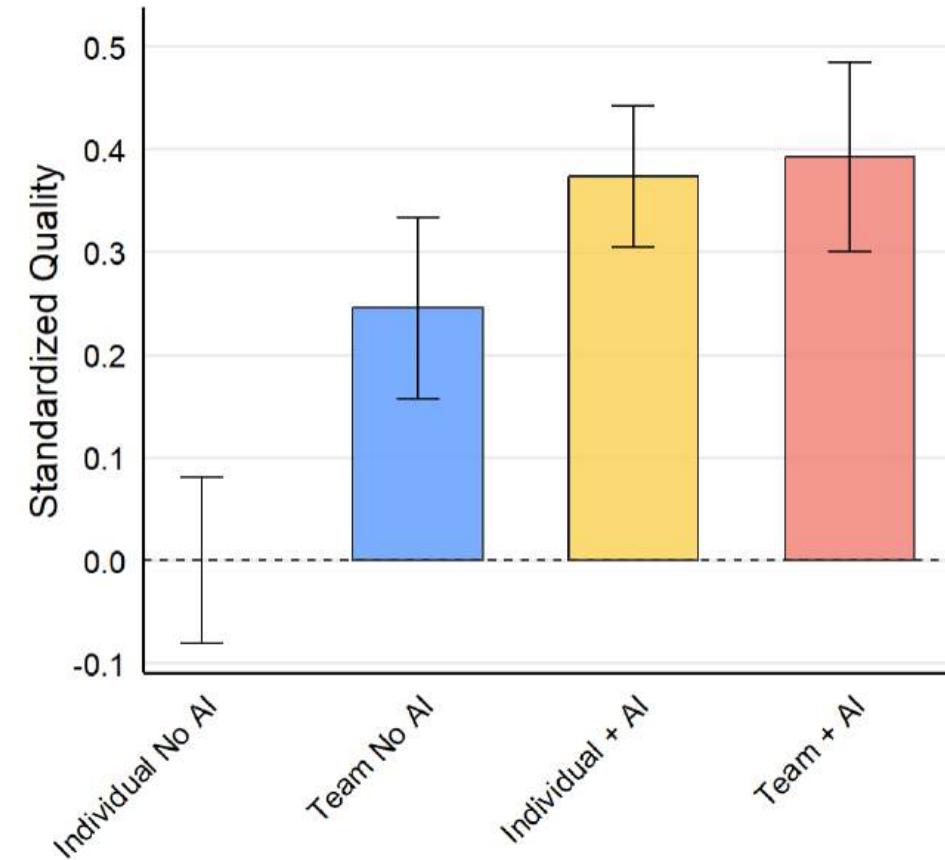
Harvard Business School Strategy Unit Working Paper No. 25-043

Harvard Business School Technology & Operations Mgt. Unit Working Paper No. 25-043

Harvard Business Working Paper No. No. 25-043

56 Pages • Posted: 21 Mar 2025 • Last revised: 21 Mar 2025

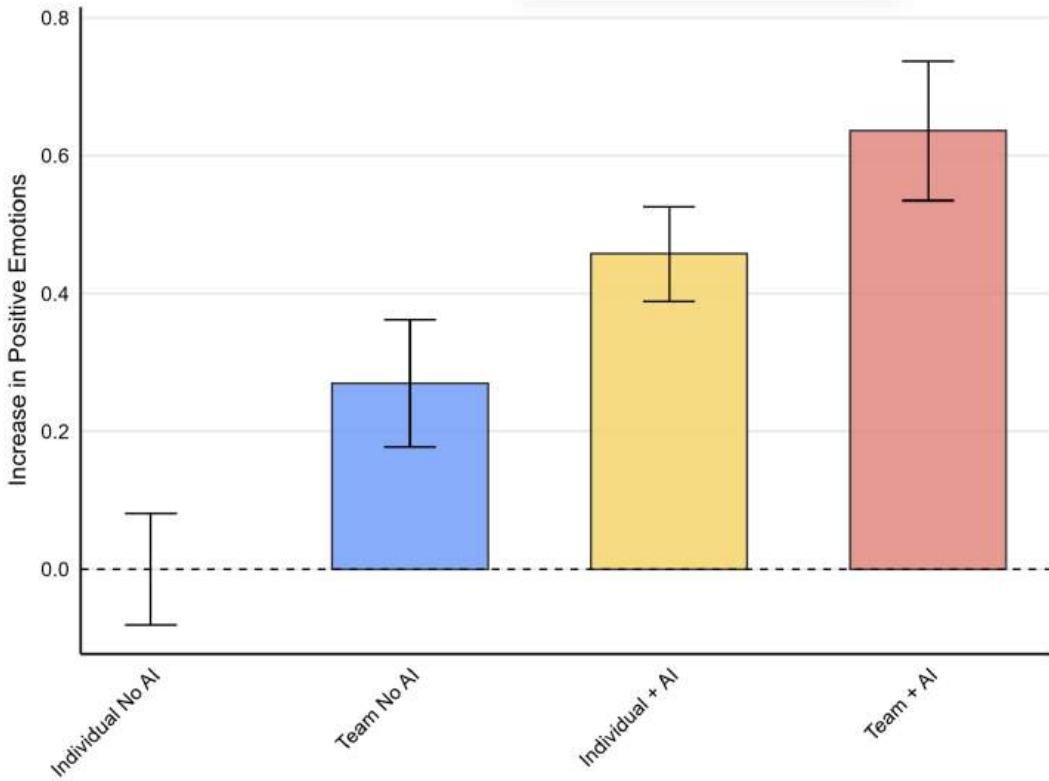
Figure 2: Average Solution Quality



# 사람이 언제 퇴사를 한다고 생각하나?

- 일이 안될 때? 힘들 때?
- 아니요. 빡칠 때입니다.
- AI를 쓰는 것이 덜 빡친다..

Figure 7: Evolution of Positive Emotions during the Task



AI slows down open source developers. Peter Naur can teach us why.

AI는 오픈소스 개발자의 작업을 지연시킵니다. 피터 나우르가 그 이유를 설명해 줄 수 있습니다.

Metr recently published [a paper](#) about the impact AI tools have on open-source developer productivity<sup>1</sup>. They show that when open source developers working in codebases that they are deeply familiar with use AI tools to complete a task, then they take longer to complete that task compared to other tasks where they are barred from using AI tools. Interestingly the developers predict that AI will make them faster, and continue to believe that it did make them faster, even after completing the task slower than they otherwise would!

Metr는 최근 [논문](#)을 발표하여 AI 도구가 오픈소스 개발자의 생산성에 미치는 영향을 분석했습니다<sup>1</sup>. 연구 결과, 오픈소스 개발자들이 자신이 깊이 이해하고 있는 코드 베이스에서 작업을 수행할 때 AI 도구를 사용하면 해당 작업을 완료하는 데 더 많은 시간이 소요된다는 것이 밝혀졌습니다. 흥미롭게도 개발자들은 AI가 작업을 더 빠르게 만들 것이라고 예측했으며, 실제로 AI를 사용하지 않았을 때보다 더 오래 걸렸음에도 불구하고 AI가 작업을 더 빠르게 만들었다고 계속 믿고 있습니다!

“When developers are allowed to use AI tools, they take 19% longer to complete issues—a significant slowdown that goes against developer beliefs and expert forecasts. This gap between perception and reality is striking: developers expected AI to speed them up by 24%, and even after experiencing the slowdown, they still believed AI had sped them up by 20%.

개발자들이 AI 도구를 사용할 수 있게 되면, 문제 해결에 19% 더 오래 걸립니다. 이는 개발자들의 믿음과 전문가들의 예측과 반대되는 심각한 작업 속도 저하입니다. 인식과 현실 간의 격차는 충격적입니다: 개발자들은 AI가 작업 속도를 24% 향상시킬 것으로 예상했으며, 심지어 작업 속도 저하를 경험한 후에도 AI가 작업 속도를 20% 향상시켰다고 믿었습니다.

— Measuring the Impact of Early-2025 AI on Experienced Open-Source Developer Productivity

2025년 초 AI가 경험이 풍부한 오픈소스 개발자의 생산성에 미치는 영향 측정

# 심지어 더 느리게 만들었다는 연구도 존재

- 하지만 여기엔 반전이 있음
- 여기 투입된 개발자들중 cursor등을 오래 써본 사람은 소수
- 즉 다들 새로운 도구에 잘 적응도 되지 않은 상태
- 이 연구도, 앞선 연구도 같은 한계가 있음
- 더불어 그들의 workflow도 AI기반으로 바뀌지 않았다.
- 업무 방식 자체를 AI친화적으로 재편해야 할 필요가 있다.
- 실제로 각 연구에서 이미 친화적인 사람은 효율 향상을 경험

AI시대에 내가 아무런 변화도 하기 싫다면  
나에게는 얼마의 시간이 남았나.

# METR 논문 (2025.3.18)

Measuring AI Ability to Complete Long Tasks

Thomas Kwa<sup>1</sup>, Ben West<sup>1\*</sup>, Joel Becker, Amy Deng, Kathryn Garcia,  
Max Hasin, Sami Jawhar, Megan Kinniment, Nate Rush, Sydney Von Arx

Ryan Bloom, Thomas Broadley, Haoming Du, Brian Goodrich, Nikola Jurkovic,  
Luke Harold Miles<sup>1</sup>, Seraphina Nix, Tao Lin, Neeru Parikh, David Rein,  
Lucas Jun Koba Sato, Hjalmar Wijk, Daniel M. Ziegler<sup>1</sup>

Elizabeth Barnes, Lawrence Chan

Model Evaluation & Threat Research (METR)

**Abstract**

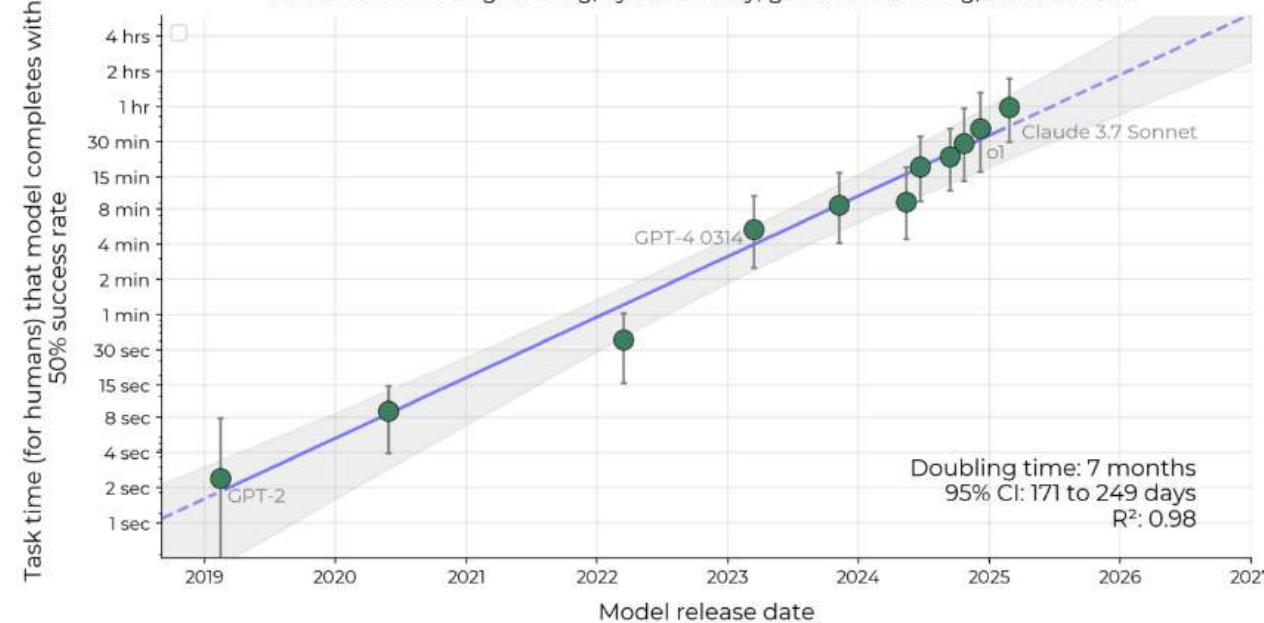
Despite rapid progress on AI benchmarks, the real-world meaning of benchmark performance remains unclear. To quantify the capabilities of AI systems in terms of human capabilities, we propose a new metric: *50%-task-completion time horizon*. This is the time humans typically take to complete tasks that AI models can complete with 50% success rate. We first timed humans with relevant domain expertise on a combination of RE-Bench, HCAST, and 66 novel shorter tasks. On these tasks, current frontier AI models such as Claude 3.7 Sonnet have a 50% time horizon of around 50 minutes. Furthermore, frontier AI time horizon has been doubling approximately every seven months since 2019, though the trend may have accelerated in 2024. The increase in AI models' time horizons seems to be primarily driven by greater reliability and ability to adapt to mistakes, combined with better logical reasoning and tool use capabilities. We discuss the limitations of our results—including their degree of external validity—and the implications of increased autonomy for dangerous capabilities. If these results generalize to real-world software tasks, extrapolation of this trend predicts that within 5 years, AI systems will be capable of automating many software tasks that currently take humans a month.

**1 Introduction**

In the last five years, frontier AI systems have undergone a dramatic transformation in capabilities, evolving from basic text generation [1] to autonomously executing complex multi-hour machine learning research projects [2]. Sufficiently capable AIs could perform dangerous, highly com-

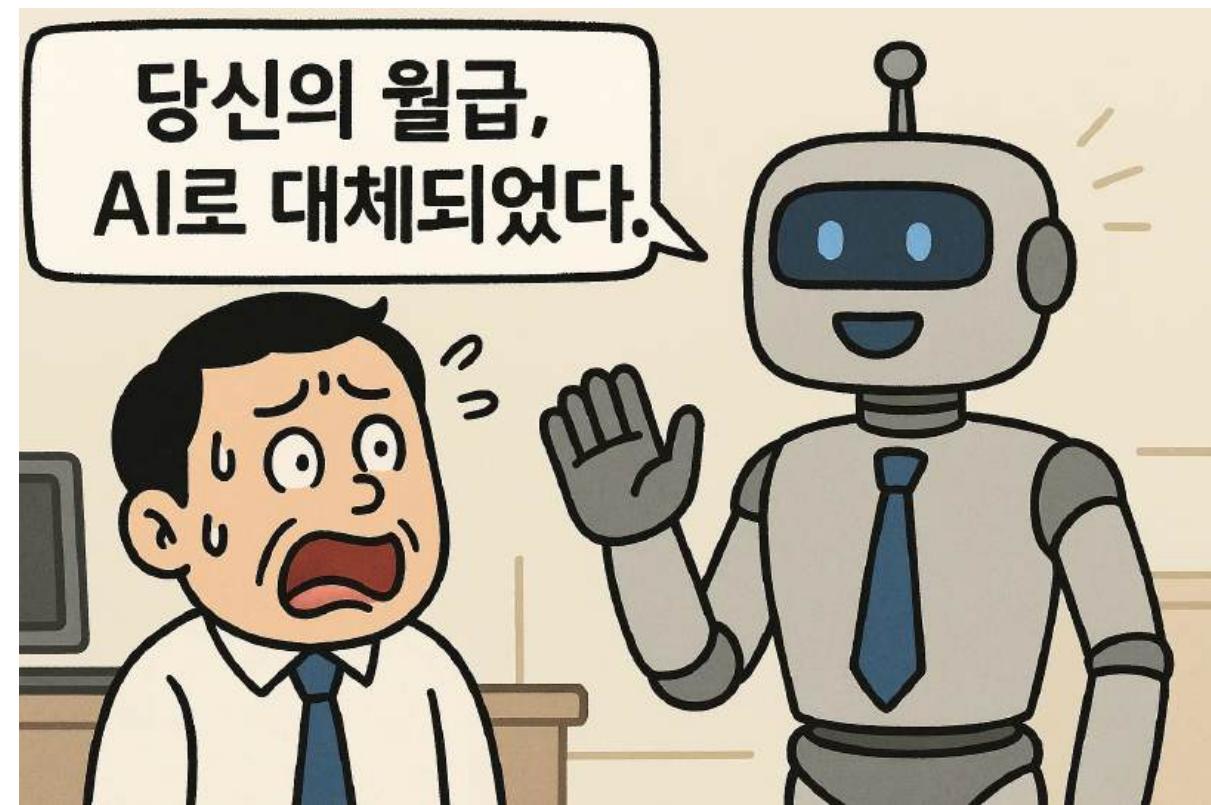
- <https://arxiv.org/abs/2503.14499>
  - 금붕어가 1만마리 있어도 사람만큼 똑똑하지 않다.
  - 1개의 AI가 얼마나 깊이 오래 문제를 풀 수 있는가가 AI의 한계
- AI는 어느정도 긴 작업 까지 잘할 수 있는가?
  - 예전 GPT2는 사람이 1분 걸리는 작업도 실패
  - Claude 3.7이 사람이 50분 걸리는 작업을 50% 정도로 성공
- 성공률을 높이려면, 한번 시키는 작업의 사이즈를 줄여 LLM이 운영할 수 있는 사이즈 안쪽으로 넣어줘야 한다.
- 아직 더 긴 사이즈의 일은 역시 인간이 해야 한다.
  - 인간에게도 오래 걸리는 일을 그냥 AI에게 시키면 복불복이다. (인샬라 코딩)

Length of tasks AI agents have been able to complete autonomously  
for 169 software engineering, cybersecurity, general reasoning, and ML tasks

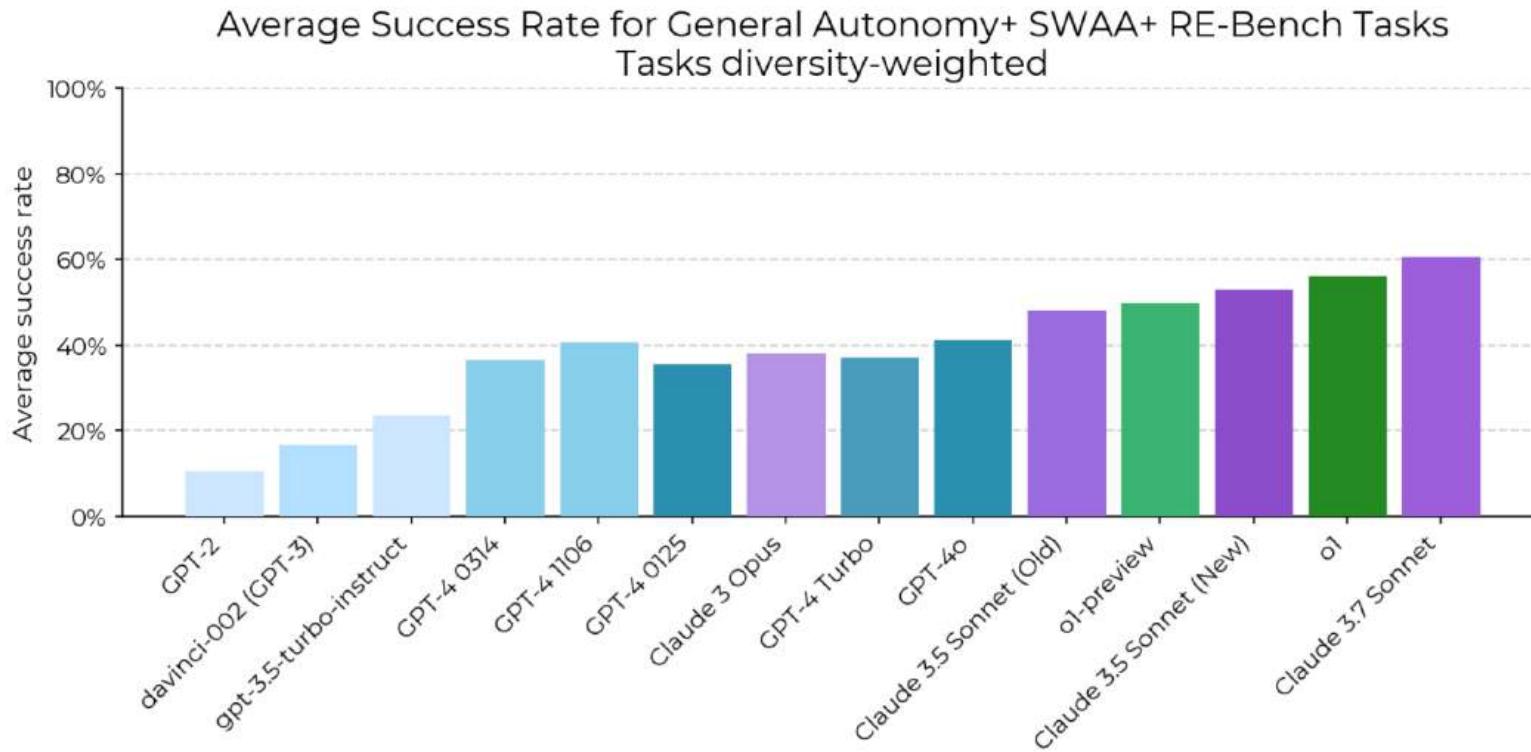


현재 발전 속도로는 7개월마다  
2배씩 더 긴 시간의 일을 할 수 있게 됨

2029년 안에  
인간이 1달 걸리는 작업을 할 수 있다고 함.  
1달?



# 그나저나 이 연구에서도 claude가 1등!



이것이 엔진이다!



I think I  
can fix this.

AI시대 결국 인간의 일이란 무엇인가?

**매우 명확한 미래**

**앞으로 몇 년 뒤에는 (비)개발자 1명이**

**100인분의 에이전트를 쪼며 코딩한다.**

**최신 람보르기니와 튼튼한 10톤트럭**

**AI가 선택지를 만들어줄 수 있다.**

**선택은 우리가 현명하게 해야한다.**

옳고 그름이 아니라, 옳음과 옳음 사이의 선택

Trade Off 사이의 가치의 선택자

# 감사합니다.

하용호

2025/07

yongho.ha@gmail.com