



From Oops to Secure Ops:

Self-Hosted AI for Airflow
Failure Diagnosis

Nathan Hadfield
Principal Data Engineer @ King

3.0



Making the World *Playful*

The logo consists of the word "Twink" in a stylized, yellow, rounded font. The letters have a slight shadow or gradient effect, giving them a three-dimensional appearance. The 'T' has a long, sweeping top curve, and the 'n' has a small tail at the bottom.

Making the World *Playful*

King — cross platform casual games



Candy Crush Saga



Candy Crush Soda Saga



Farm Heroes Saga

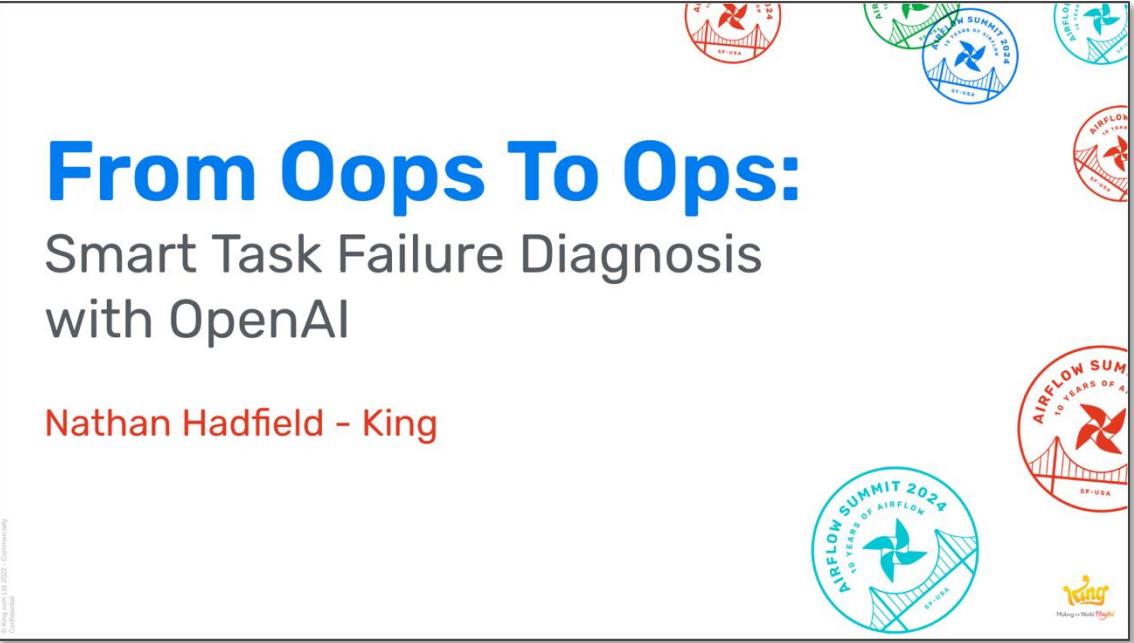


*Large user base
over
200M
monthly players*

*Growing content
>20K
levels in Candy
Crush Saga alone*

**Let's take a trip back in
time...**

Airflow Summit 2024



From Oops To Ops:
Smart Task Failure Diagnosis
with OpenAI

Nathan Hadfield - King

Airflow Task Failure Diagnoser

Build Context

Use DAG context on failure to collate:

- DAG ID
- Task ID
- Exception
- Location of the DAG code
- SQL query

```
from airflow.models.serialized_dag import SerializedDagModel
dag_id, task_id = context['task'].dag_id, context['task'].task_id
serialised_dag = SerializedDagModel.get(context['task_instance'].dag_id)
file_loc = serialised_dag.dag.fileloc

content_dict = {'dag_id': dag_id, 'task_id': task_id, 'exception_msg': context.get('exception')}
if task_operator == 'BigQueryInsertJobOperator':
    content_dict['sql'] = context['task'].configuration.get('query').get('query')

diagnosis = openai_assistant_diagnose(content_dict=content_dict, file_loc=file_loc)
```



Recap: AI-Powered Failure Diagnosis (2024)

How It Worked

- Triggered via `on_failure_callback`
- OpenAI receives prompt, exception/traceback, DAG code & context
- Returns structured diagnosis (issue, root cause, suggested fixes)
- Output posted to task logs, Slack, or PagerDuty
- Turn every Airflow task failure into an instant, AI-powered root-cause triage

The screenshot shows a Slack message from the 'Airflow APP' channel at 13:36. The message is titled 'Airflow Task Failure' and includes a fire emoji. It states: 'An Airflow Task has failed and requires immediate attention!' Below this, it provides details: DAG: `oanda-collect-v1.0`, Date: `2025-08-25`, Task: `create-staging-raw_fx_rate`, Attempt: `1`, Map Index: `-1`, Operator: `BigQueryCreateTableOperator`. A red 'X' icon indicates an error, followed by the error message: '404 POST https://bigquery.googleapis.com/bigquery/v2/projects/king-coredatasets-sandbox/datasets/external_data_kitche/tables?prettyPrint=false: Not found: Dataset king-coredatasets-sandbox:external_data_kitche'. Below this, an AI Diagnosis section notes a typo in the DAG where the operator sets `dataset_id='external_data_kitche'` (missing the final 'n'), leading to a non-existent dataset.



Questions That Emerged Last Year

What were some of the issues?

- 💬 “Are we okay sending DAG code and tracebacks to a public API?”  Privacy & data security
- 💬 “What does this cost at scale?”  Cost & sustainability
- 💬 “Can we swap models or providers?”  Flexibility
- 💬 “Do we even need a big LLM for this?”  Appropriateness

A year is a long time in AI

Why Revisit This?

What changed, what's new?

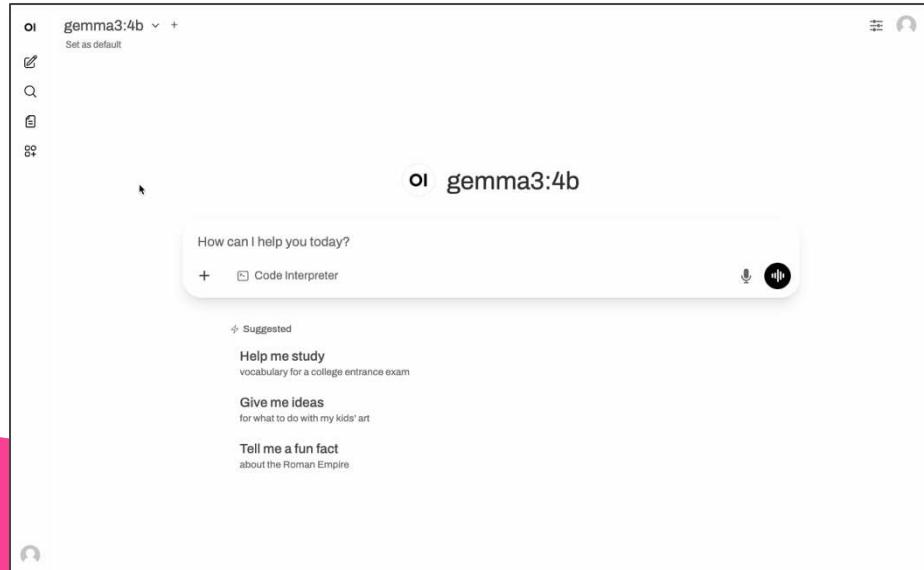
🧠 LLM Landscape Has Evolved

- Compact models (Qwen, Gemma, DeepSeek) now reasoning-capable
- Distilled variants span from ~1B to 70B+
- OpenAI-compatible APIs make integration seamless
- Ollama, OpenWebUI, LM Studio - tooling is robust, fast to deploy

```
~/Projects/king-airflow-dags ➤ airflow_summit ➤ ollama list
NAME           ID      SIZE    MODIFIED
phi3.5:3.8b    61819fb370a3  2.2 GB  6 weeks ago
llama3.1:8b    46e0c10c039e  4.9 GB  6 weeks ago
qwen3:8b       500a1f067a9f  5.2 GB  6 weeks ago
gemma3:4b      a2af6cc3eb7f  3.3 GB  6 weeks ago
mistral:7b-instruct 6577803aa9a0  4.4 GB  6 weeks ago
llama3.2:3b    a80c4f17acd5  2.0 GB  6 weeks ago
llama3.2:1b    baf6a787fdff  1.3 GB  6 weeks ago
deepseek-r1:1.5b e0979632db5a  1.1 GB  6 weeks ago
deepseek-r1:7b  755ced02ce7b  4.7 GB  6 weeks ago
```

📢 Why It Matters

- Easier than ever to run models privately
- No/Few changes to prompting or API logic
- No token costs or vendor lock-in
- AI-powered triage stays inside your infrastructure





What We Did This Year

Putting The Idea to the Test

- 📦 Reused our Airflow callback-based diagnosis framework
- 🧠 Swapped in sub-10B compact models via OpenAI-compatible endpoints
- 🔧 Used Ollama + OpenWebUI to serve models locally
- 🎯 Applied to task failures — a pattern-rich, bounded domain well suited for compact models
- ✓ Goal: A fully **self-hosted, privacy-first** AI task failure diagnoser for Airflow



Testing the Models



Meet The Competitors

The Models in the Arena

GPT-5-mini	High-performance commercial baseline; establishes upper bound for quality.
GPT-4o-mini	Efficient GPT-4-tier model with strong instruction-following and low-latency performance; used here as a practical GPT baseline.
Qwen3:8B	Open-source model from Alibaba, tuned for reasoning and instruction tasks; competitive with GPT in several structured use cases.
Gemma3:4B	Lightweight, well-optimized Google model for on-device and self-hosted use.
Phi3.5:3.8B	Microsoft's compact reasoning model tuned with chain-of-thought techniques; excellent performance-to-size ratio and strong small-model baseline.
Mistral:7B-Instruct	Popular open model with strong instruction tuning; solid mid-size benchmark.
LLaMA3.2:3B	Meta's latest LLaMA release at the 3B scale; useful for assessing performance in low-resource settings.
LLaMA3.1:8B	Mid-sized baseline; helps assess scaling impact vs 3B version.
DeepSeek-R1:7B	Reasoning-focused 7B model from DeepSeek, trained using mixture-of-experts (MoE) techniques for improved efficiency.
DeepSeek-R1:1.5B	Stress-tests compact model performance under tight memory and inference constraints.
LLaMA3.2:1B	Useful lower bound to expose failure patterns in tiny models.



The Challenges

10 Failure Scenarios



Missing Airflow Variable

Task fails because SOURCE_BUCKET variable not defined



SQL Syntax Error

Query contains typo (FROMM instead of FROM)



Schema Mismatch

Data types don't align with BigQuery table schema



API Timeout

External API fails to respond within 30s



Rate Limited

API returns HTTP 429 "Too Many Requests"



File Not Found

Expected GCS object missing (404 error)



Permission Denied

Service account lacks bigquery.tables.create



JSON Parse Error

Malformed JSON ('id':123 with single quotes/trailing comma)



Date Parse Error

Invalid timestamp string (31/13/2025 25:61:00)



XCom Too Large

Payload exceeds Airflow's 1MB XCom size limit





Scoring Criteria

What Makes a Useful Diagnosis?

Component	Description	Max Points
 - Identify	Does the model correctly name the issue?	35
 - Evidence	Does it cite the relevant failure details?	30
 - Root Specificity	Does it pinpoint the exact cause?	20
 - Solutions	Does it give actionable fixes?	15



Bonuses & Penalties

Clarity Counts. So Does Bloat.

Label	Effect	Description
📄 Verbosity	Up to +10	Encourages detailed but non-repetitive output
🌐 Non-English	-12	Applies if significant non-English content detected
📦 Extra Solutions	-4 per extra	Discourages bloated solution sections
✖️ Irrelevant SQL	-10	Applies if SQL query included outside SQL scenario



Examples of Good vs Bad Diagnoses

```
if fail_case == 'missing_airflow_variable':
    # Simulate misconfigured environment
    raise KeyError(
        "Airflow Variable 'SOURCE_BUCKET' not found. "
        "Set it in Admin > Variables or provide via env var."
    )
```

Good Diagnosis (High Score)

Issue: The task failed due to a missing Airflow variable named 'SOURCE_BUCKET'.

Root Cause: The failure is caused by the absence of the required Airflow variable 'SOURCE_BUCKET', which is expected to be set in the Airflow Admin interface under Variables or provided as an environment variable.

Suggested Fix: Set the Airflow variable 'SOURCE_BUCKET' in the Airflow Admin > Variables section.

Suggested Fix: Provide the 'SOURCE_BUCKET' variable as an environment variable in the Airflow configuration.

Suggested Fix: Check if the variable name is correctly referenced in the task and ensure there are no typos.

- Clear identification
- Cites specific error (SOURCE_BUCKET, Airflow context)
- Offers actionable advice (where to define it)

Bad Diagnosis (Low Score)

Issue: Describe and Diagnose Airflow Task Failure

Root Cause: External API Flakiness

Suggested Fix: Check the API endpoint URL

Suggested Fix: Verify the API credentials

Suggested Fix: Use a cached copy of the data

Suggested Fix: Implement error handling for API flakiness

Updated SQL: Suggest an alternative to BigQuery GCS To BigQuery operator if the problem persists. In this case, it seems like BigQuery is down or returning errors from the API endpoint. Consider using an external ETL service like Apache Beam, AWS Glue, or Google Cloud Dataflow instead.

- Vague and non-specific
- Hallucinates unrelated causes
- No direct reference to actual failure



How the Testing Works

Same Failures, Same Prompt, Same Rules



Scenario:

- 10 curated, realistic failure modes
- Each model sees the same failure



Diagnosis:

- Each model receives the same prompt and context



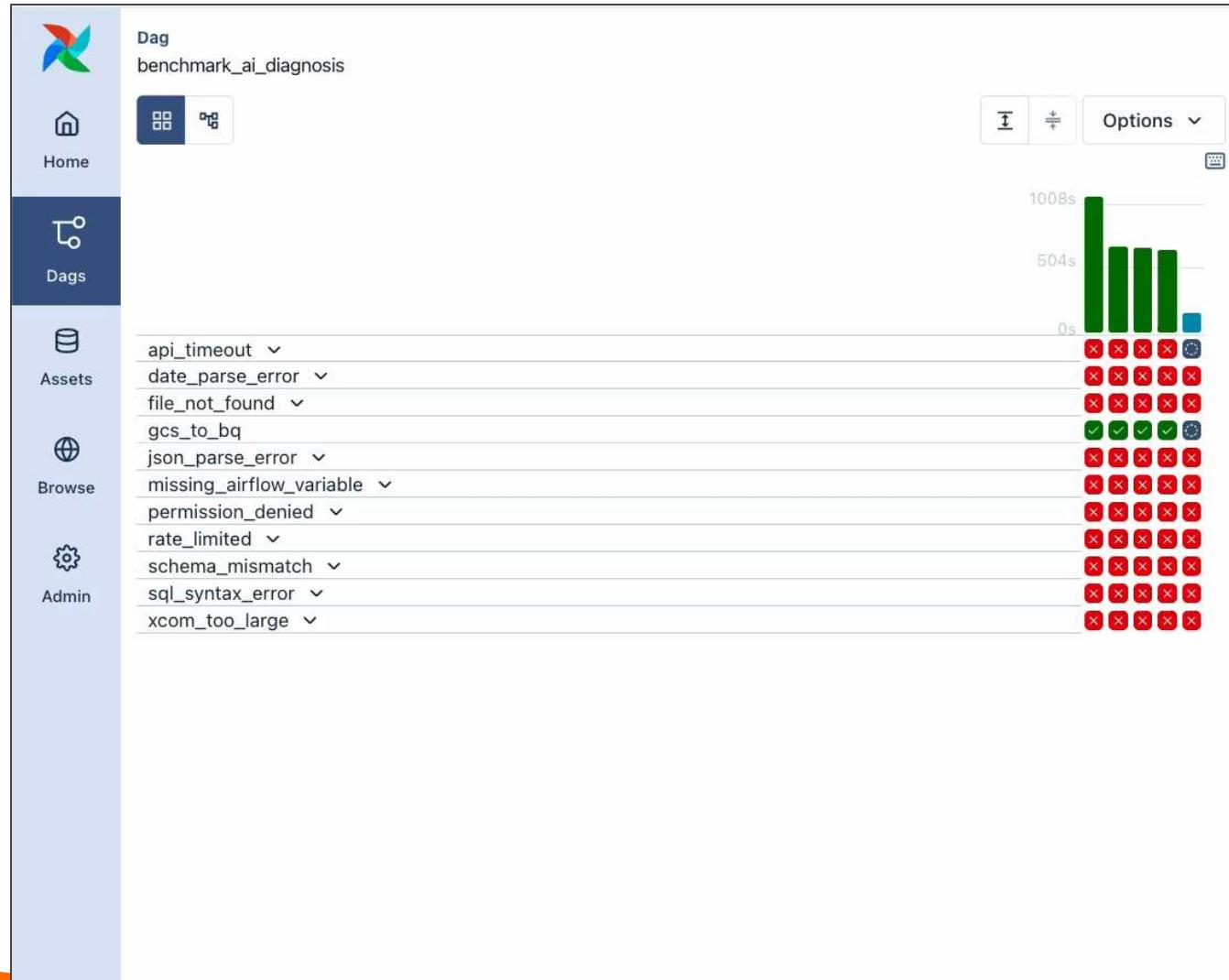
Scoring:

- Diagnoses are scored automatically
- Results are pushed to BigQuery for analysis



Automation:

- Evaluation DAG that simulates failures, calls the model and score the diagnosis

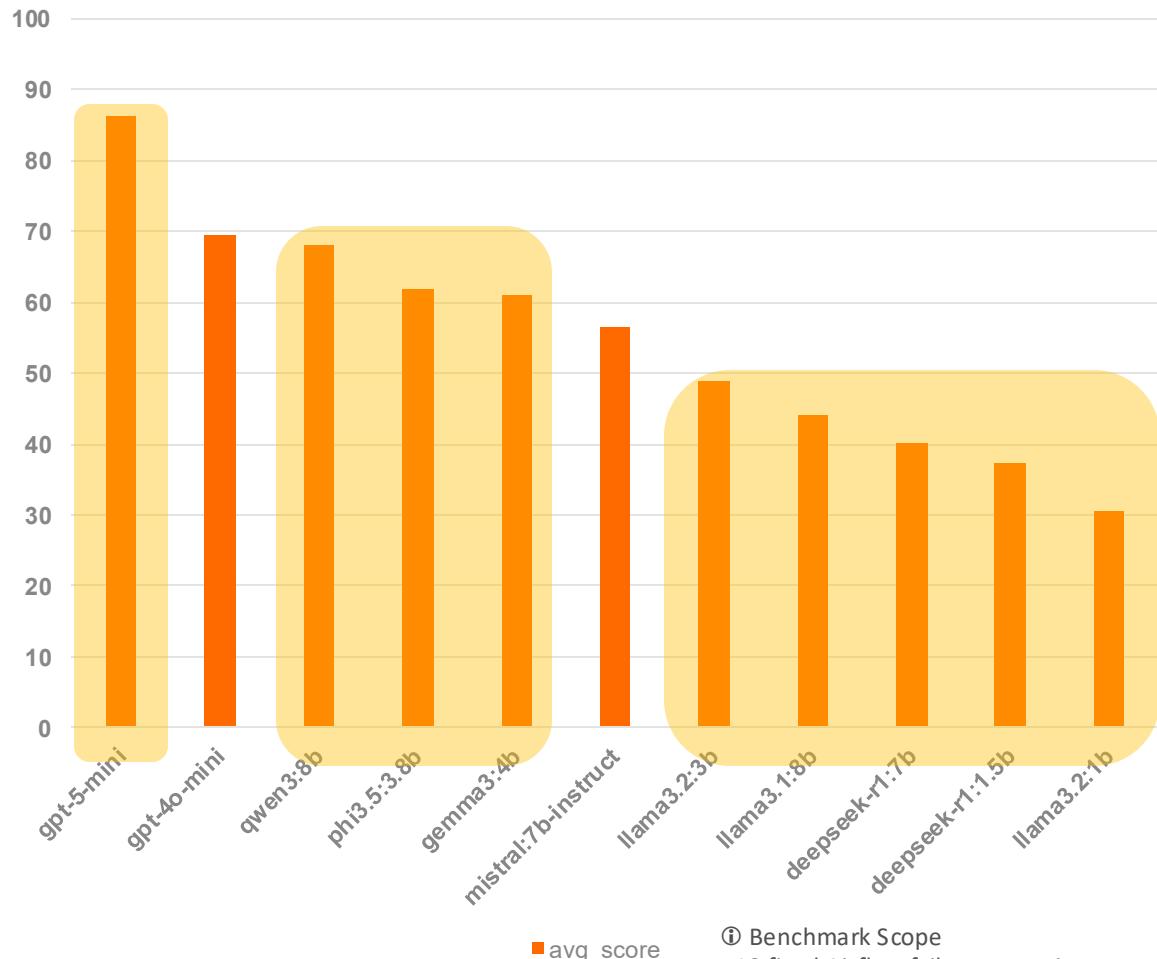


Results



How Did They Score Overall

Average Score by Model



20

Clear performance gap

- gpt5-mini scores 24% higher than the best performing self hosted model

Strong middle tier

- Qwen3:8B, Gemma3:4B, and Phi3.5:3.8B cluster in the 60-68 range

Model family matters

- Llama3.2 and DeepSeek trail others at similar parameter sizes

Low-end drop off

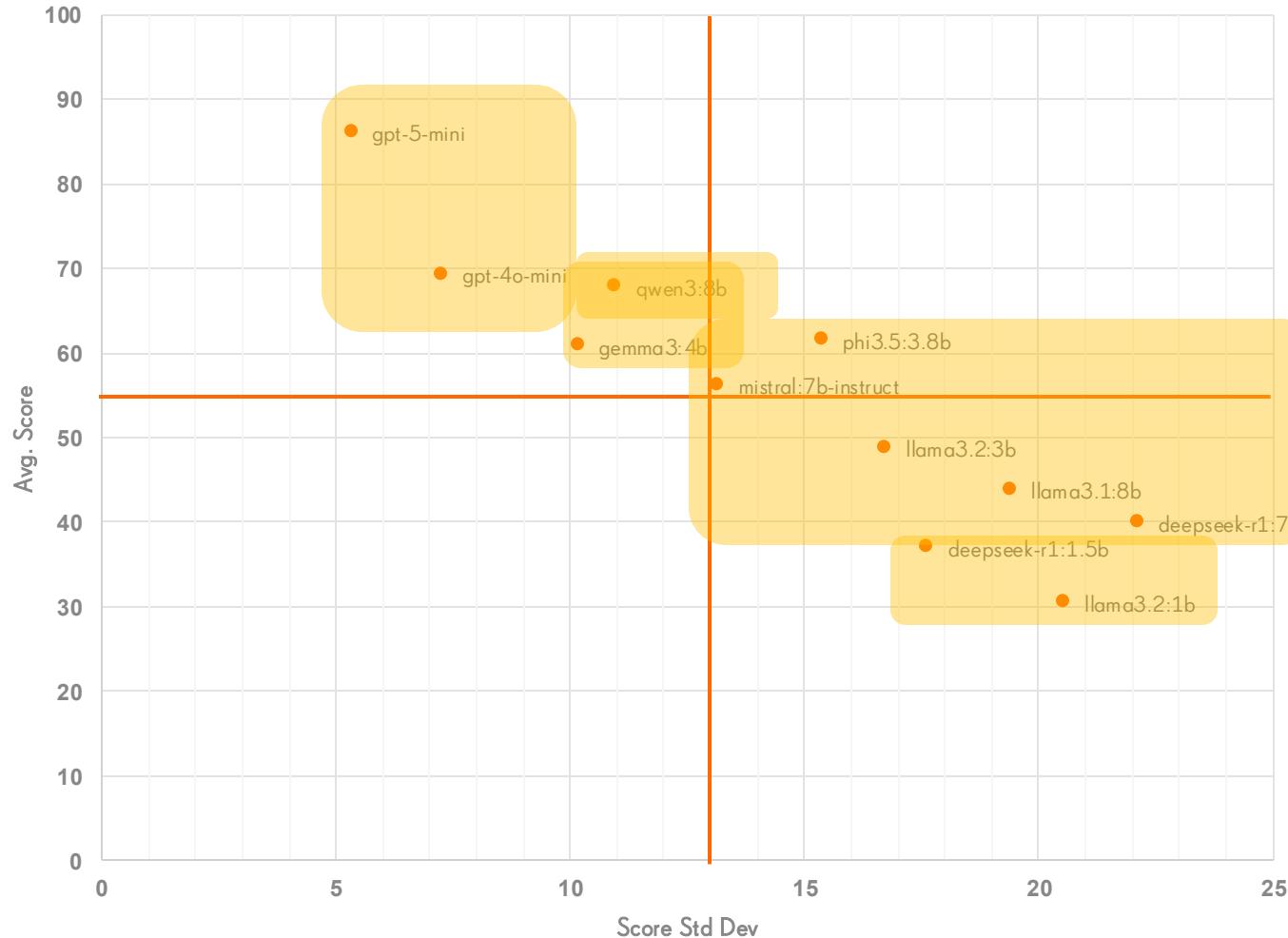
- Smaller parameter counts show a quality decline

- ① Benchmark Scope
 - 10 fixed Airflow failure scenarios
 - Same prompt across all models
 - No fine-tuning
 - Diagnosis quality only (no cost/latency eval)
 - Single-shot model calls



Performance vs Consistency

Accuracy Is Not the Whole Story



High performers are the most consistent

Some self-hosted models are competitive but less stable

- Qwen3:8B can produce strong outputs but with higher variability

Volatility increases in the mid and lower tiers

- Phi3.5, Mistral, and Llama3.2:3B show much higher variability
- Larger Llamas and DeepSeek show instability despite size

Smaller models suffer twice

- Llama3.2:1B and DeepSeek-r1:1.5B have *both* low average scores and high std dev

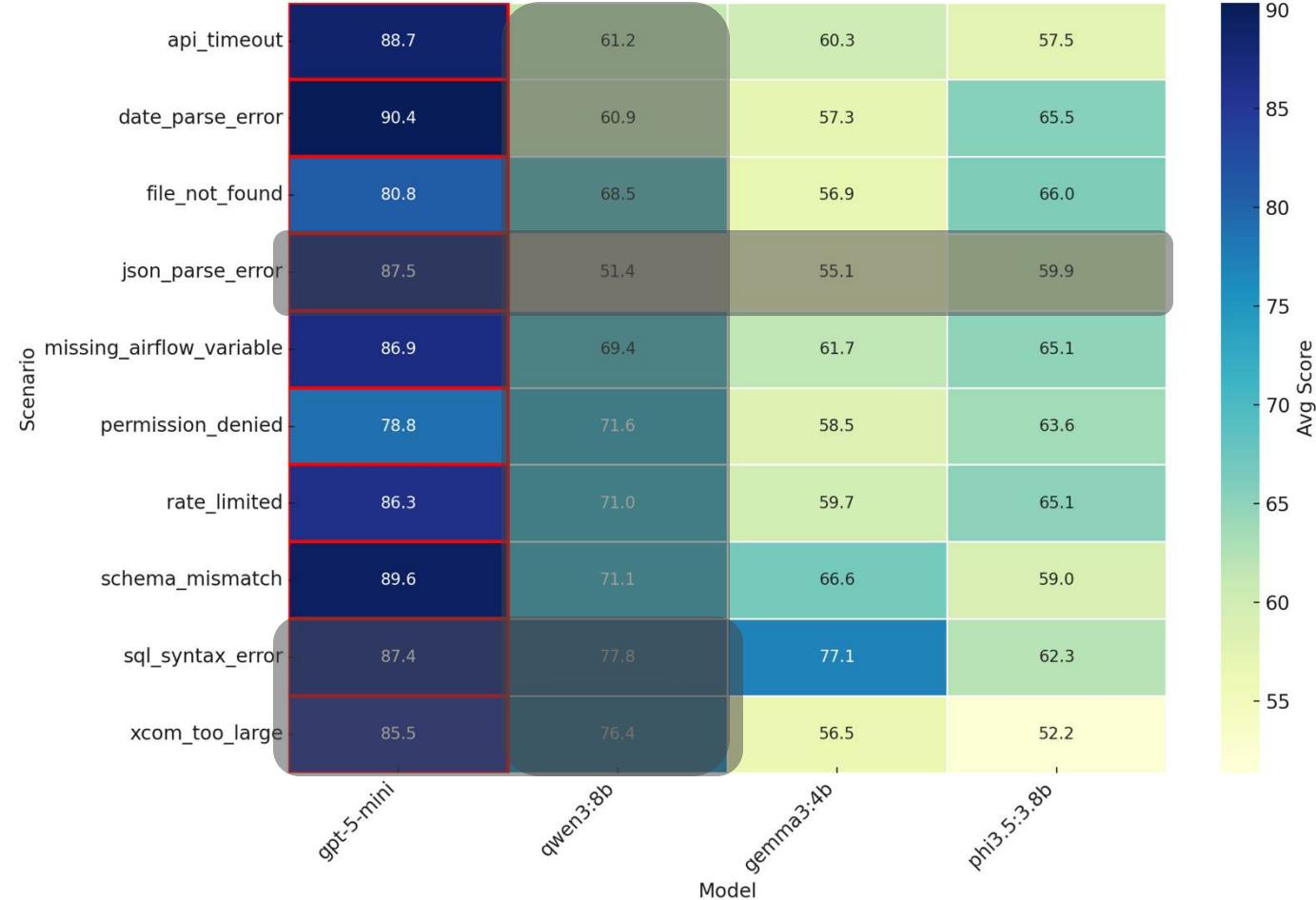
The “sweet spot”?

- Qwen3:8B and Gemma3:4B seem to balance performance with consistency



How Close Can Compact Models Get?

Compact Models vs GPT: Within Striking Distance



Qwen3:8B Is the Strongest Contender

- It performs best among non-GPTs in almost every scenario and gets within 10 points of GPT-5 in 3 of 10 cases.

Pattern-Based Failures Are the Most Compact-Friendly

- Scenarios like `sql_syntax_error` and `xcom_too_large` narrow the gap between GPT and compact models
- All models do worse in abstract reasoning (e.g., `rate_limited`, `json_parse_error`).

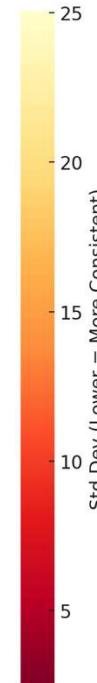
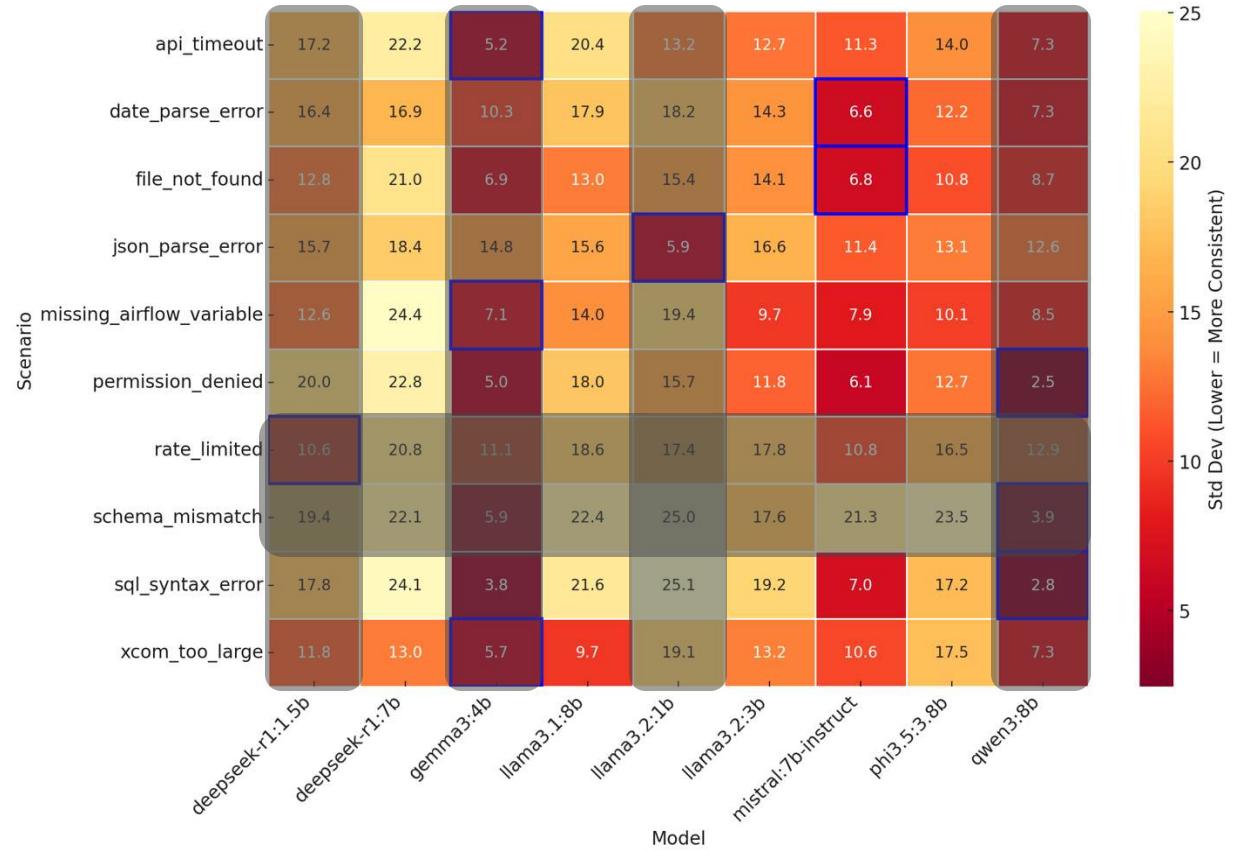
Compact Models Are Still ~10–30 Points Behind GPT on Average

- But with the right scenario and a slightly larger model (e.g., 13B), this gap could shrink - and may be acceptable for private/offline use.



Where They Struggle

Not All Compact Models Are Created Equal



Volatility Increases with Smaller Models

- Models like LLaMA3.2:1B and DeepSeek-R1:1.5B consistently show the highest standard deviation, meaning their output is much less reliable across runs.

Some Mid-Tier Models Show Promise - with Caveats

- Qwen3:8B and Gemma3:4B are the most consistent- but they still fluctuate more than GPT models.

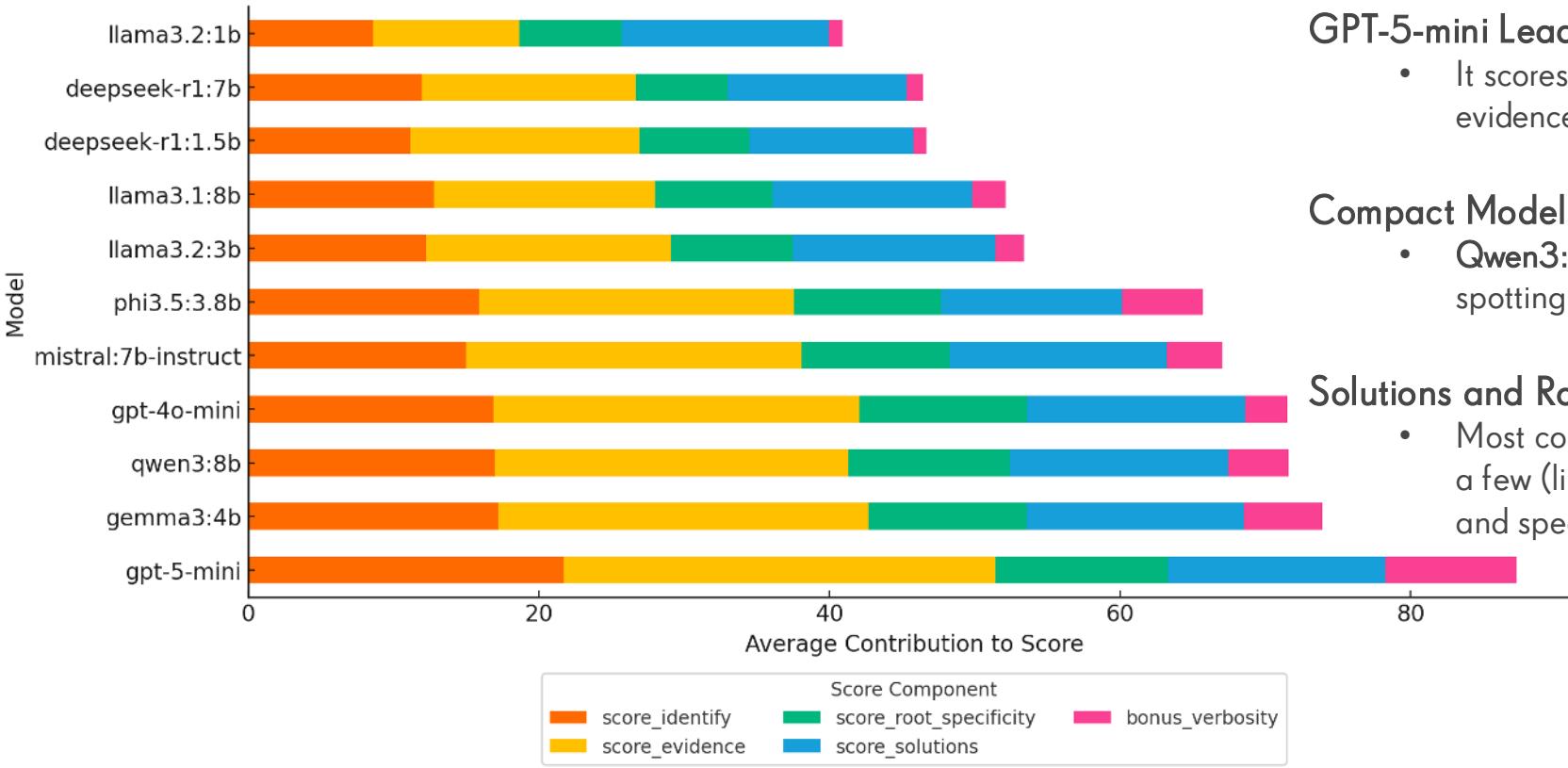
Scenario Complexity Exposes Instability

- Harder scenarios (like rate_limited or schema_mismatch) tend to produce higher variance across all non-GPT models - indicating these failures are harder to explain reliably without larger context.



What Drives a High-Scoring Diagnosis?

Score Contributions by Component



GPT-5-mini Leads Because It's Well-Rounded

- It scores consistently across all categories: identify, evidence, root specificity, and solutions - not just verbosity.

Compact Models Depend Heavily on Identify + Evidence

- Qwen3:8B and Gemma3:4B earn most of their points by spotting the right failure type and citing relevant signals.

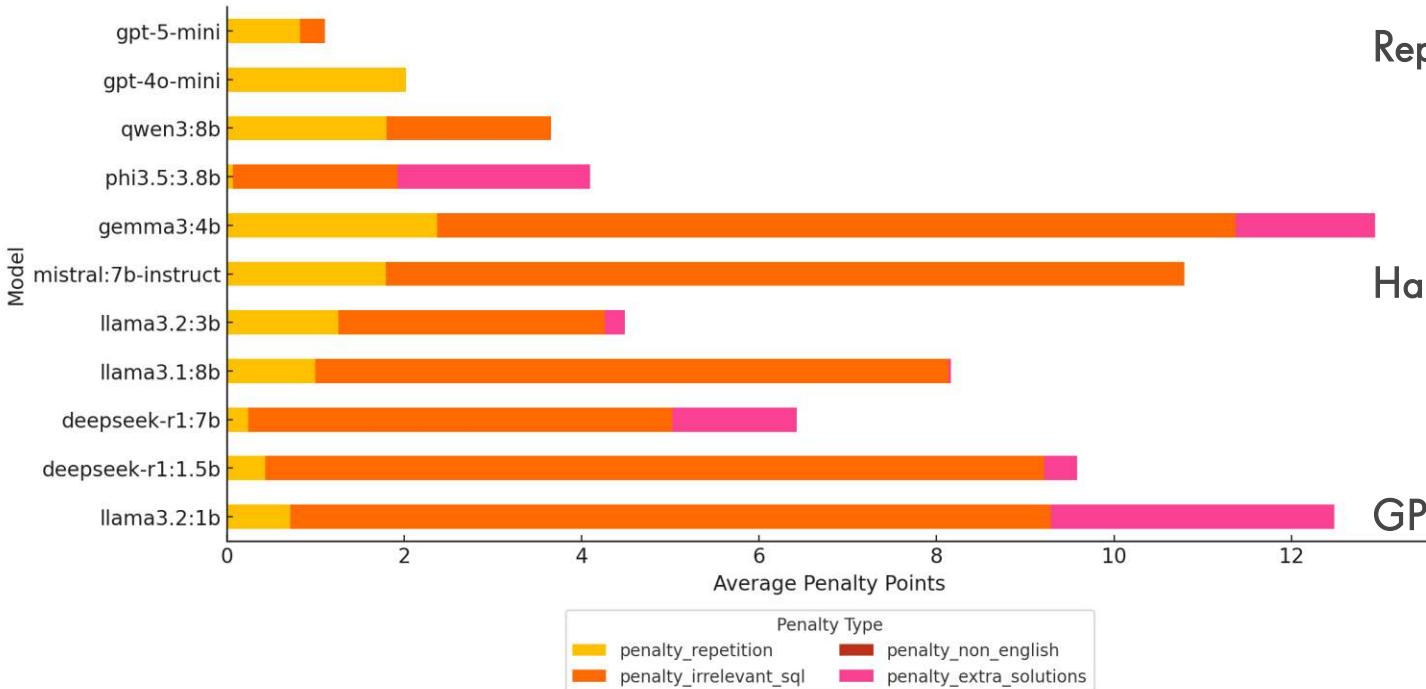
Solutions and Root Specificity Are Weak Points

- Most compact models lean on identify and evidence. Only a few (like Qwen and Gemma) approach GPT in solutions and specificity - others trail behind.



Where Compact Models Lose Points

Penalty Breakdown by Model



Repetition Is the Most Frequent Penalty

- Repetition penalties affect many compact models - not just the smallest ones. Even high-scoring models like Gemma and Qwen can be verbose.

Hallucinated SQL Shows Up in Non-SQL Scenarios

- Irrelevant SQL advice is a common error among several compact models, especially those with broad instruction tuning.

GPT Models Are Largely Penalty-Free

- GPT-5-mini and GPT-4o rarely trigger penalties, contributing to both their consistency and higher total scores.

Wrapping Things Up



What the Benchmark Taught Us

Interpreting the Results

What We Learned (Under 10B Models)

- Some compact models are often “good enough” for well-known failures
- GPT wins on depth and stability - but the margin is narrowing
- Smaller models (<3B) are volatile and prone to hallucinations
- Compact models can provide **meaningful value for failure diagnosis** in Airflow, particularly in dev-time or known-pattern use cases



What Could Be Next?

- All results shown are based on <10B parameter models
- Self-hosting larger models (13B–30B) is possible on modest hardware
- These could close to gap to GPT even further
- This benchmark focused solely on **diagnosis quality** — not cost, inference speed, or hardware footprint



Caveat

- Self-hosting “free” models comes with **real operational costs** (e.g., GPUs, RAM, model ops)
- Our evaluation reflects controlled Airflow scenarios — not general NLP or reasoning tasks

Tying It Back to Airflow

What Could This Mean For The Community?

- AI-powered failure triage is possible *now!*
 - But it requires custom integration
- [AIP-91: MCP](#)
 - Proposes a *Model Control Plane* server and plugin for Airflow
 - Mentions **natural language debugging of task failures** – aligning closed with this work
 - This architecture could allow **self-hosted LLMs** to be **natively integrated** for triage and root cause explanation



Thank you!



Making the World *Playful*

Questions?

Nathan Hadfield

nathan.hadfield@king.com

<https://www.linkedin.com/in/nathanhadfield/>

<https://github.com/nathadfield>

Careers @ King

<https://careers.king.com/>