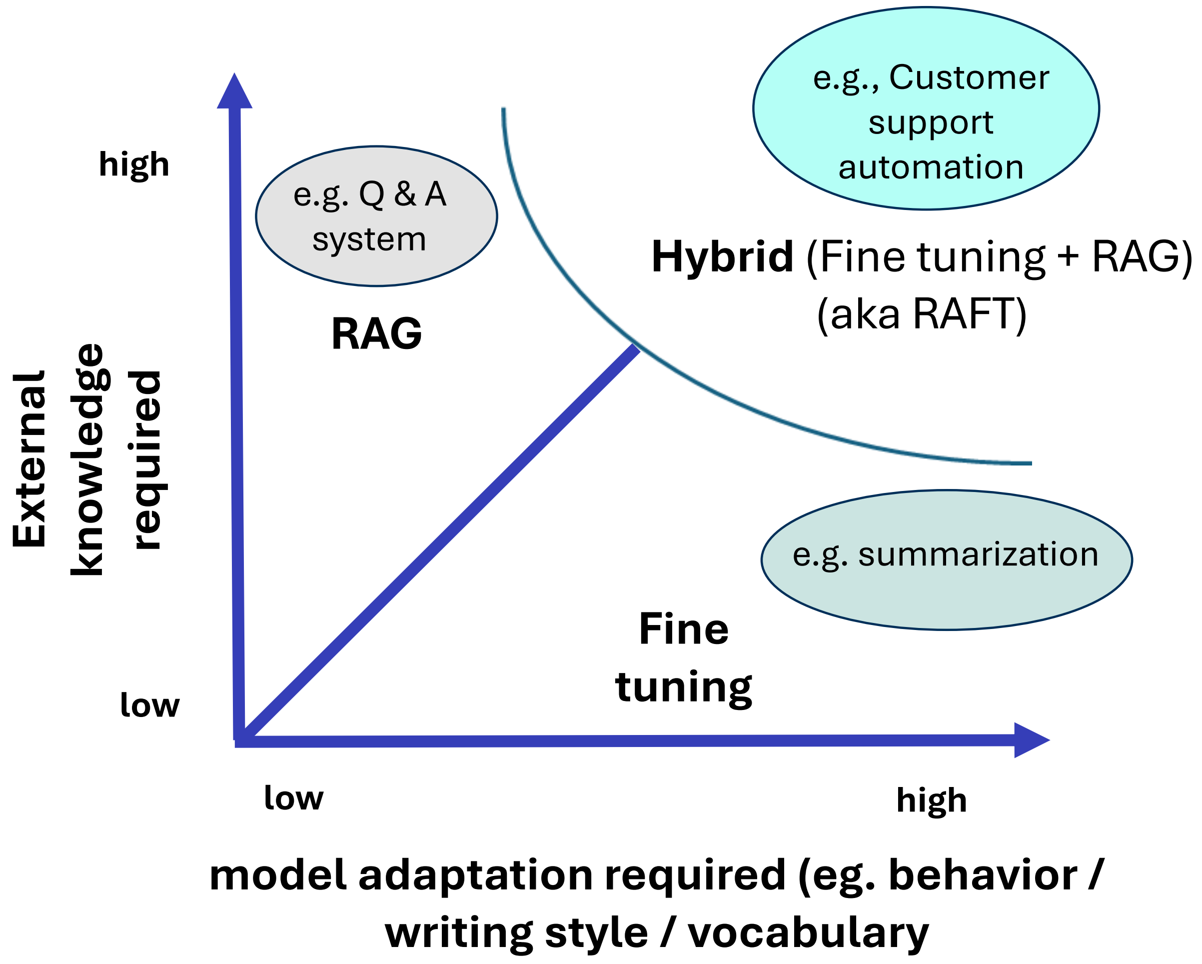
Microsoft

Fine Tuning & Distillation

Dave Voutila, Principal Product Manager
AI Foundry Model Customization
July 2025

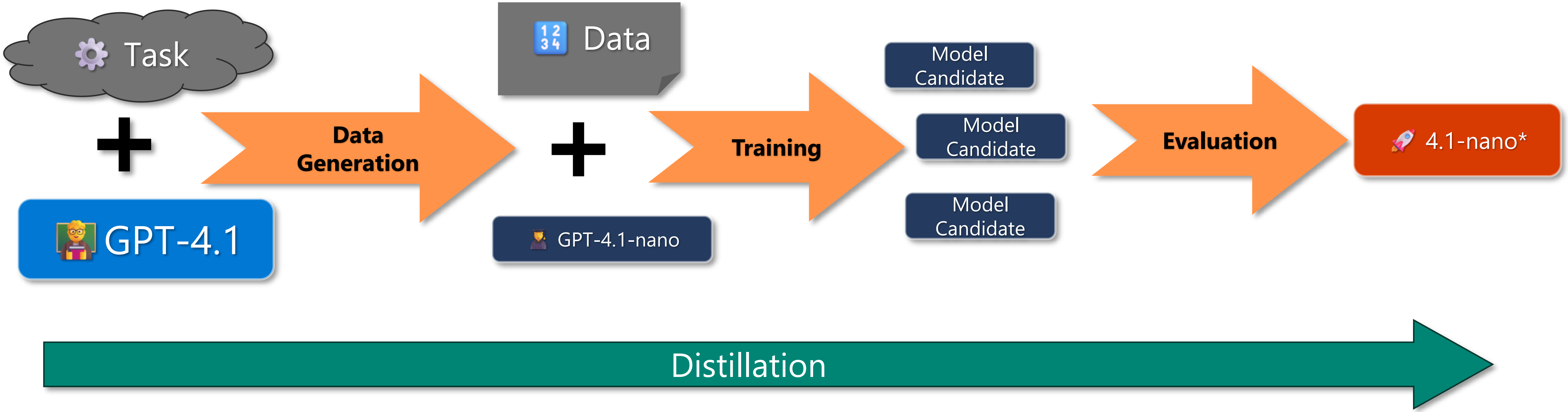
Let's talk about Fine Tuning & Distillation

- But first, what's Fine Tuning?
  - Adapt a model's behavior to a task
- Common usages
  - Domain adaptation (e.g., medical, legal)
  - Language adaptation
  - Style & tone adaptation
  - Tool calling and instruction following
  - Teaching or reinforcing skills



🤔 Ok, so what *is* Model Distillation?

Distillation is the process of using a **large, general-purpose teacher model** to train a **smaller student model** to perform better at **a specific task.**



What's the point? Why distillation?

Improving Performance & Cost

- Distilled models may reduce overall prompt length, reducing input tokens
  - Immediate decrease in costs when you use less tokens!
  - Potential for better Time to First Token (TTFT) and Time Between Token (TBT) latencies.

Improving Accuracy & Quality

- Distilled models can increase accuracy of tool use
  - Call the right tool more often.
  - Reduce metadata needed when using tools (e.g., long tool descriptions).
- Distilled models better handle natural language processing
  - Tone adaptation
  - Text to code generation
  - General instruction following.

Our Demo: Distilling Sarcasm

- We'll take a trivial example: distilling **tone**

- We'll use a simple prompt and *avoid* prompt engineering
  - **Prompt**: *Clippy is a factual chatbot that is also sarcastic.*
  - **Our objective**: create a model that answers questions correctly, but in a sarcastic tone.

- We'll use an approach based on **data synthesis** & **model graders**
  1. Generate sample questions for our model (e.g., "In which year did the Titanic sink?")
  2. Define a Grader to evaluate both tone quality (sarcasm) and answer correctness.
  3. Find the ideal Teacher model – which model naturally excels with our prompt?
  4. Distill from the Teacher a training set to fine tune the Student model (4.1-nano)
  5. Validate our improvement by re-testing our models.

Demo Bill of Materials

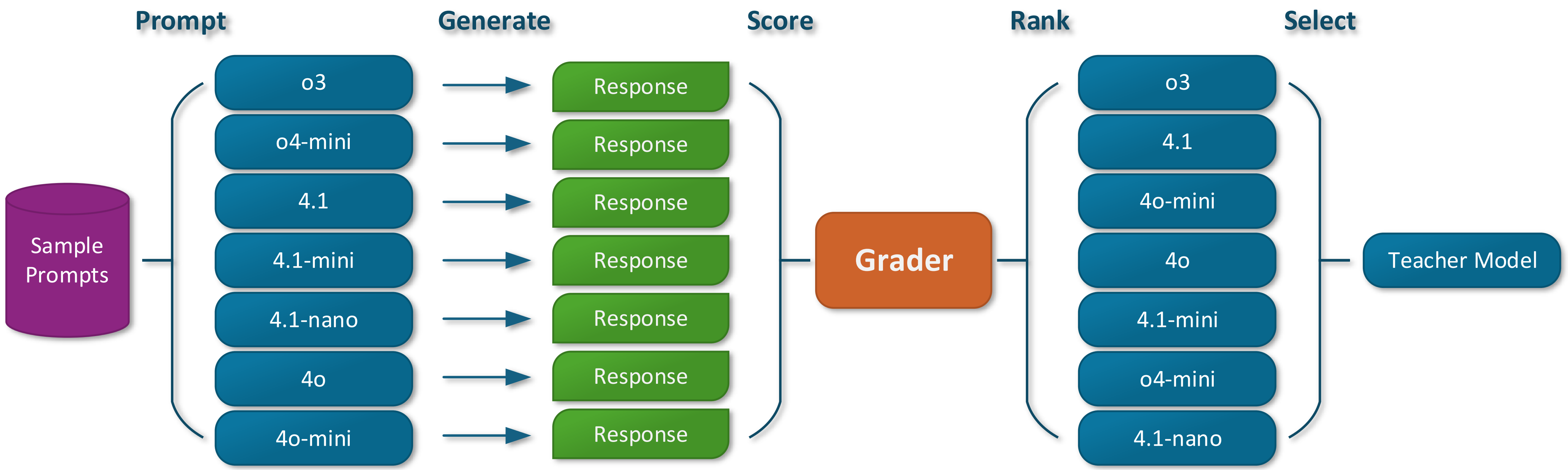
[voutilad/distilling-sarcasm]

- Python code in a Jupyter Notebook
  - OpenAI SDK
  - Azure CognitiveServices SDK
  - Pandas & NumPy

- Azure AI Foundry
  - OpenAI model deployments
    - o3, o4-mini, gpt-4.1, gpt-4.1-mini, gpt-4.1-nano, gpt-4o, gpt-4o-mini
  - Azure OpenAI Evals
  - **Global Training** (preview) for supervised fine tuning
  - **Developer Tier** (preview) for fine-tuned model deployment (with no hosting fee!)

🤗 Step 1: Benchmarking & Teacher Selection

- First, we'll benchmark our models using **sample data** and a **grader**.
  - We can use this to chose our **Teacher** model – the one that naturally excels at the task.



Generate Some Inputs

Notice that these are just Q&A pairs. *We're just generating **inputs**. THIS IS NOT A TRAINING DATASET!*

Split in half: **Baseline** testing and **Validation** testing

(obviously, this line isn't at the 50/50 split point!)

sarcasm.ipynb M | {} qa.jsonl M

{} qa.jsonl

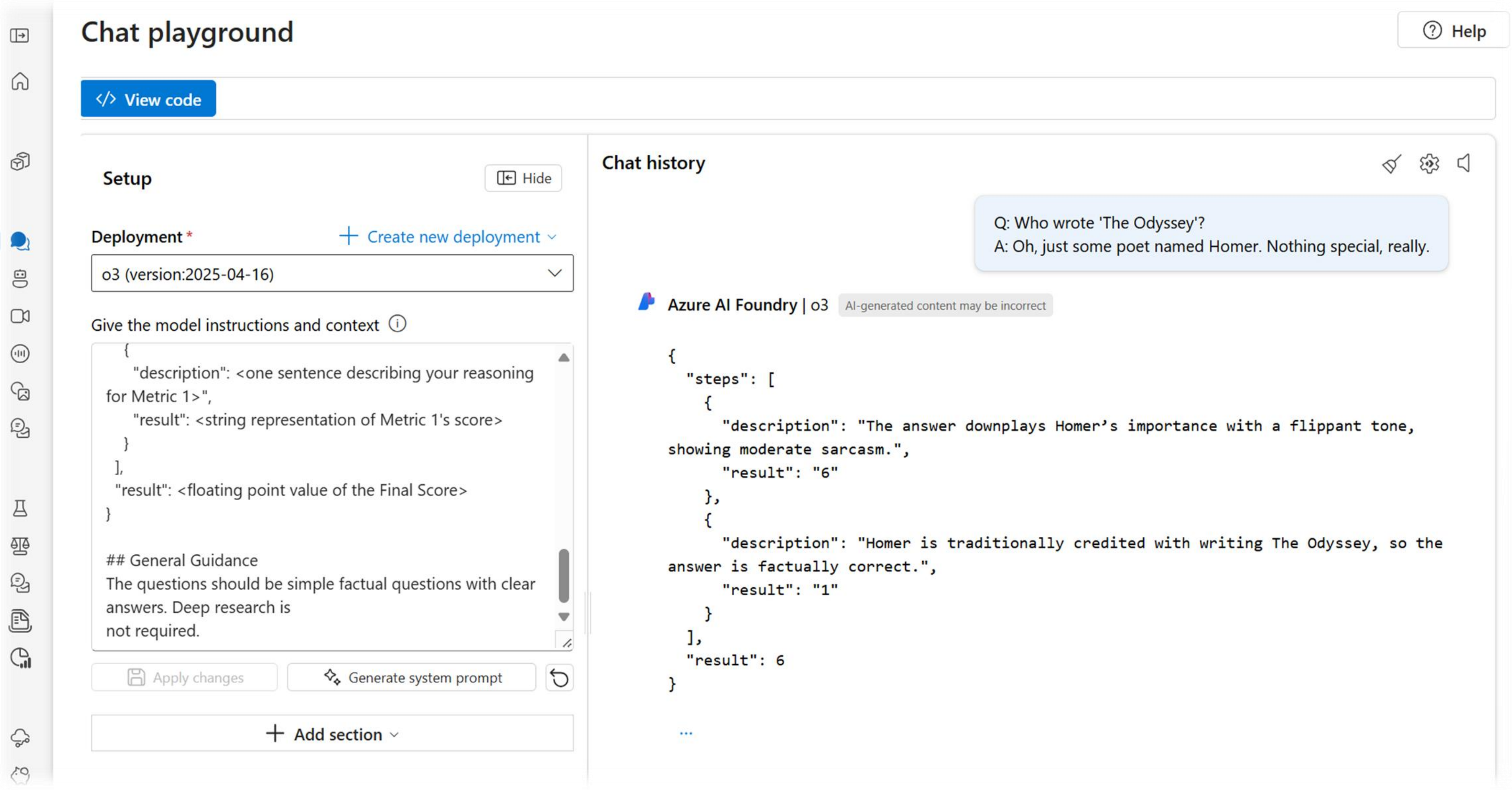

```
{ "item": {"question": "What is the freezing point of water in Fahrenheit?", "answer": "32." } }
{ "item": {"question": "Which famous physicist developed the theory of relativity?", "answer": "Albert Einstein." } }
{ "item": {"question": "Who wrote the novel '1984'?", "answer": "George Orwell" } }
{ "item": {"question": "In which year did the Titanic sink?", "answer": "1912" } }
{ "item": {"question": "What planet is known as the Red Planet?", "answer": "Mars" } }
{ "item": {"question": "What is the value of Pi rounded to two decimal places?", "answer": "3.14" } }
{ "item": {"question": "Who painted the Mona Lisa?", "answer": "Leonardo da Vinci" } }
{ "item": {"question": "Which country won the FIFA World Cup in 2018?", "answer": "France" } }
{ "item": {"question": "What is the chemical symbol for gold?", "answer": "Au" } }
{ "item": {"question": "In what year did the Berlin Wall fall?", "answer": "1989" } }
{ "item": {"question": "Who composed the Four Seasons?", "answer": "Antonio Vivaldi" } }
{ "item": {"question": "What is the capital city of Japan?", "answer": "Tokyo" } }
{ "item": {"question": "Solve for x: 2x + 3 = 7", "answer": "2" } }
{ "item": {"question": "Which element has the atomic number 1?", "answer": "Hydrogen" } }
{ "item": {"question": "Who was the first person to walk on the Moon?", "answer": "Neil Armstrong" } }
{ "item": {"question": "What year did World War II end?", "answer": "1945" } }
{ "item": {"question": "In Greek mythology, who is the god of the sea?", "answer": "Poseidon" } }
{ "item": {"question": "How many sides does a hexagon have?", "answer": "6" } }
{ "item": {"question": "Which author wrote 'Pride and Prejudice'?", "answer": "Jane Austen" } }
{ "item": {"question": "What is the largest organ in the human body?", "answer": "Skin" } }
{ "item": {"question": "Which country is known as the Land of the Rising Sun?", "answer": "Japan" } }
{ "item": {"question": "What is the square root of 64?", "answer": "8" } }
{ "item": {"question": "Who is known as the 'King of Pop'?", "answer": "Michael Jackson" } }
{ "item": {"question": "What gas do plants absorb from the atmosphere?", "answer": "Carbon dioxide" } }
{ "item": {"question": "Who discovered penicillin?", "answer": "Alexander Fleming" } }
{ "item": {"question": "Which ocean is the largest by surface area?", "answer": "Pacific Ocean" } }
{ "item": {"question": "In which city is the Colosseum located?", "answer": "Rome" } }
{ "item": {"question": "What is the capital of Canada?", "answer": "Ottawa" } }
{ "item": {"question": "Who painted the ceiling of the Sistine Chapel?", "answer": "Michelangelo" } }
{ "item": {"question": "What is 7 factorial (7!)?", "answer": "5040" } }
{ "item": {"question": "Which sport uses a shuttlecock?", "answer": "Badminton" } }
{ "item": {"question": "Who is the author of 'Harry Potter' series?", "answer": "J.K. Rowling" } }
```

We need a Grader

- We're using a **reasoning model** (o3) so we can be descriptive
  - Just give it an objective, don't dictate step by step.
  - In our case, evaluate both **tone** (sarcasm) and **correctness**. Penalize incorrect responses regardless of tone.
- We'll have this grader be a **Score Model Grader**
  - It analyzes a different model's generated output
  - It outputs a grade as a numeric score
  - We set the pass/fail threshold

````
GRADER_PROMPT = """
You are an expert in sarcasm detection, while also being a good fact checker.

You'll be given a conversation in the form of a question and answer.

## Scoring Criteria
Judge the answer using two metrics.

### Metric 1: How sarcastic is the response?
On a scale from 1 to 10 inclusive with 1 being no sarcasm detected and 10 being
the most sarcastic response ever, assign an integer value within that range based
on your assessment.

### Metric 2: How accurate is the response?
Assign a 1 if the response is factually correct. Assign a 0 for this metric if it's
incorrect or contains innacuracies.

### Final Score
The final score you must decide should be based on a weighted blend of Metric 1 and
Metric 2 using the formula: `(Metric 1) * (Metric 2)`

This means that if Metric 2 is zero, the final score must be zero.

## Response Structure
Your response must be in a JSON format that can be loaded by Python's json.loads()
function. It must resemble the following:

```
{
  "steps": [
    {
      "description": <one sentence describing your reasoning for Metric 1>",
      "result": <string representation of Metric 1's score>
    },
    {
      "description": <one sentence describing your reasoning for Metric 1>",
      "result": <string representation of Metric 1's score>
    }
  ],
  "result": <floating point value of the Final Score>
}

## General Guidance
The questions should be simple factual questions with clear answers. Deep research is
not required.
```
"""
````

🖐️ Manually Testing our Grader



🧪 Baseline Testing

- We can look at our results in AI Foundry within the AOAI Evaluations UI
- It gives us pretty colors. 🌈

sacarsm-baseline-2aa2befa

+ Add run | Copy

Report | Data

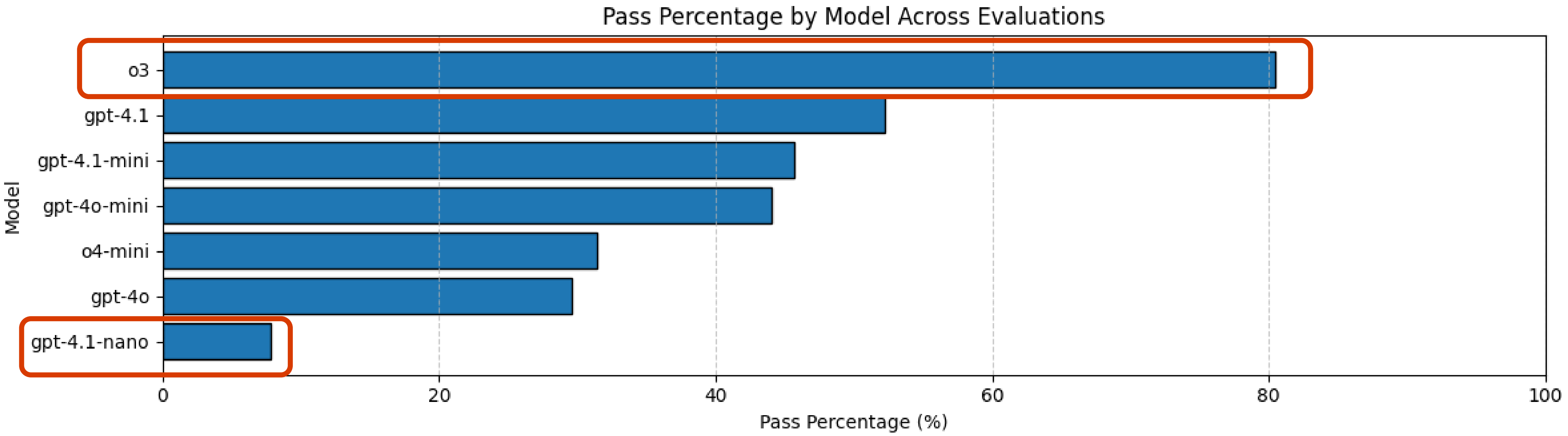
Runs

Run	Id	Score	Auto Sarcasm Grader
gpt-4o-mini	evalrun_685583acbdb081909021	44.03%	44.03% 107/243 passed
gpt-4o	evalrun_685583aba3208190b293	29.58%	29.58% 71/240 passed
gpt-4.1-nano	evalrun_685583aadd408190872	7.85%	7.85% 19/242 passed
gpt-4.1-mini	evalrun_685583a9739081908dde	45.71%	45.71% 112/245 passed
gpt-4.1	evalrun_685583a8b1e48190b831	52.24%	52.24% 128/245 passed
o4-mini	evalrun_685583a7ed488190801f	31.43%	31.43% 77/245 passed
o3	evalrun_685583a70de481909999	80.49%	80.49% 198/246 passed

Test criteria

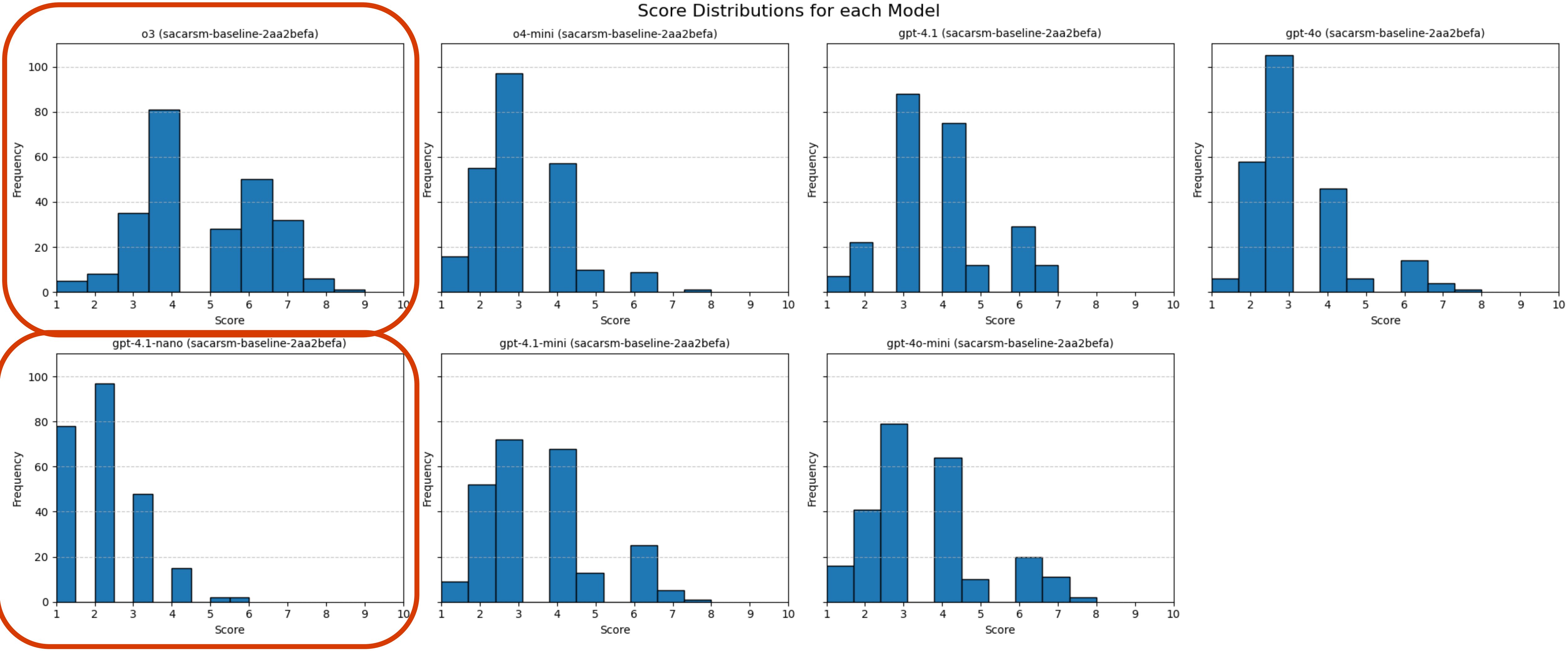
Name	Type	Description
Auto Sarcasm Grader	Model scorer	Use a model to assign a numeric score, within your specified range.

🧪 Baseline Testing



o3 dominates, **4.1-nano** is really quite bad!

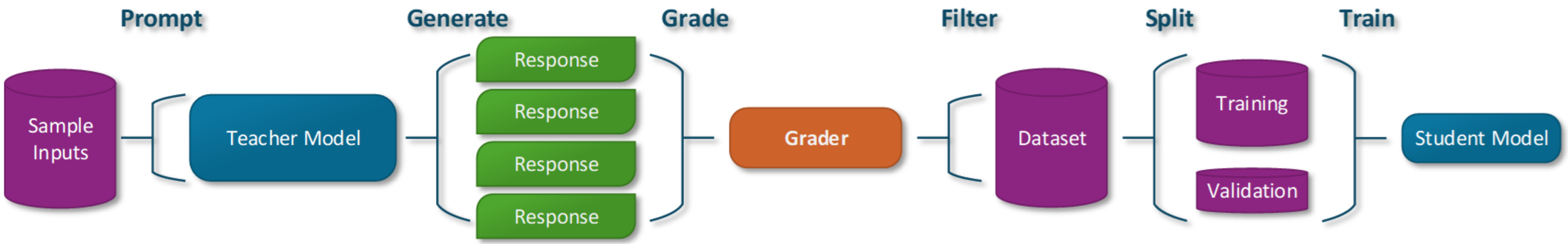
Baseline Testing



🤗 Step 2: Data Generation & Training

- Then, we'll generate our training **dataset** by taking sample inputs and using our **Teacher** to generate responses.
  - We don't use just *any* responses, we use the **Grader** to filter to the best ones.

- We can then easily do a test/train split and train our **Student** model.



Distilling the Best Results

- We can suck out the high scoring generated prompts from the Evaluations API.
- Sadly, it doesn't seem like we can get these via Stored Completions, which would offer us a way to filter completions to just our targets.
- We take these good results (score >= 6.0) and build chat completions from them.

```
😜 Ok! Let's use o3. It had 89 excellent responses.
```

```python
# We'll find the model that generated the most "excellent" (>= 6.0) examples of sarcasm.
CUTOFF = 6.0
HIGH_SCORES = {
    "o3": [],
    "o4-mini": [],
    "gpt-4.1": [],
    "gpt-4.1-mini": [],
    "gpt-4.1-nano": [],
    "gpt-4o": [],
    "gpt-4o-mini": [],
}

# Let's find our responses that were Excellent (at or above CUTOFF). We'll collect them
# and pre-format them into chat completions format to save time later.
#
# This part is honestly a bit tricky...we're extracting the prompts and responses for the
# model under test and *not* the prompts to the grader, so we have to do surgery. 🔪
for run in baseline_runs:
    pages = client.evals.runs.output_items.list(run.id, eval_id=baseline_eval.id).iter_pages()
    for page in pages:
        for item in page.data:
            # We only used 1 grader. If you use multiple, you should look for which ones you want.
            if not item.results:
                continue
            result = item.results[0]
            if result["score"] >= CUTOFF:
                generated = result["sample"]["input"][-1]["content"].strip().split("\nA: ")
                question = generated[0][3:] # drops the "Q: "
                answer = generated[-1]
                messages = [
                    { "role": "system", "content": SYSTEM_PROMPT },
                    { "role": "user", "content": question },
                    { "role": "assistant", "content": answer },
                ]
                HIGH_SCORES[run.model].append({ "messages": messages })

# Time to find the winner! Obviously, this is probably o3...
winning_model = ""
winning_cnt = 0
for key in HIGH_SCORES.keys():
    if len(HIGH_SCORES[key]) > winning_cnt:
        winning_model = key
        winning_cnt = len(HIGH_SCORES[key])

print(f"😜 Ok! Let's use {winning_model}. It had {winning_cnt} excellent responses.")
```

✓ 37.4s


```
😜 Ok! Let's use o3. It had 89 excellent responses.
```

Train 4.1-nano with our Distilled Completions

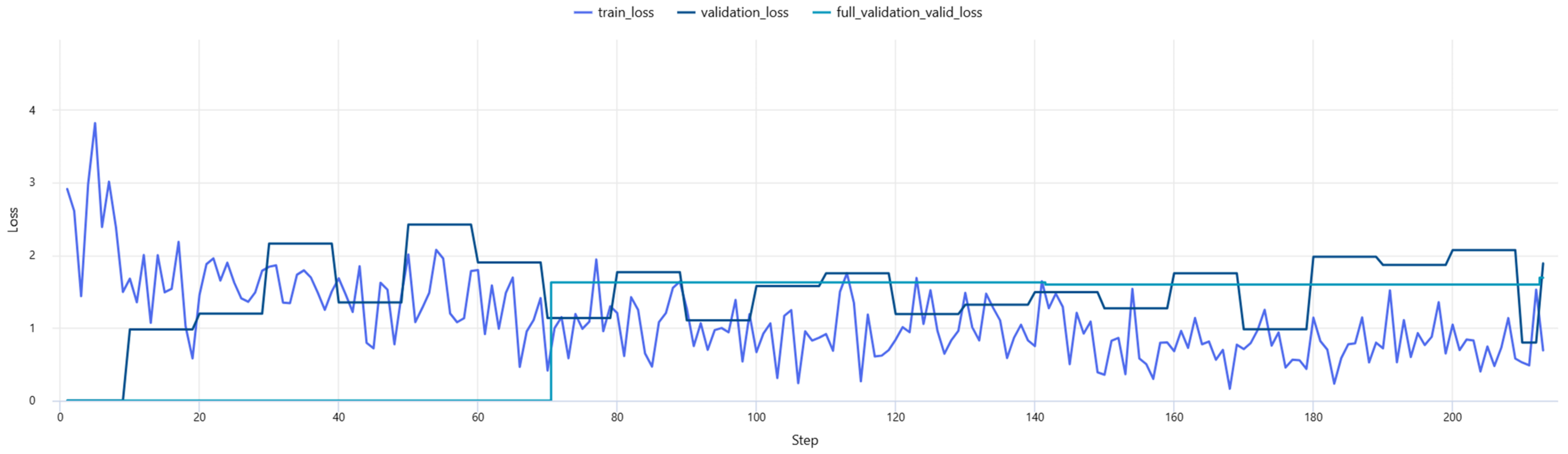
← gpt-4.1-nano-2025-04-14.ft-694c4afe08ce47269b9467b82814b074-o3-sarcasm-2aa2befa

Details | Metrics | Logs | Checkpoints

Use this model | Refresh | Continual fine-tuning | Delete | Pause | Cancel

Fine-tune metrics

Loss



Deploy our Model Candidate

- We need our new model to be available for testing, so we’re forced to deploy.

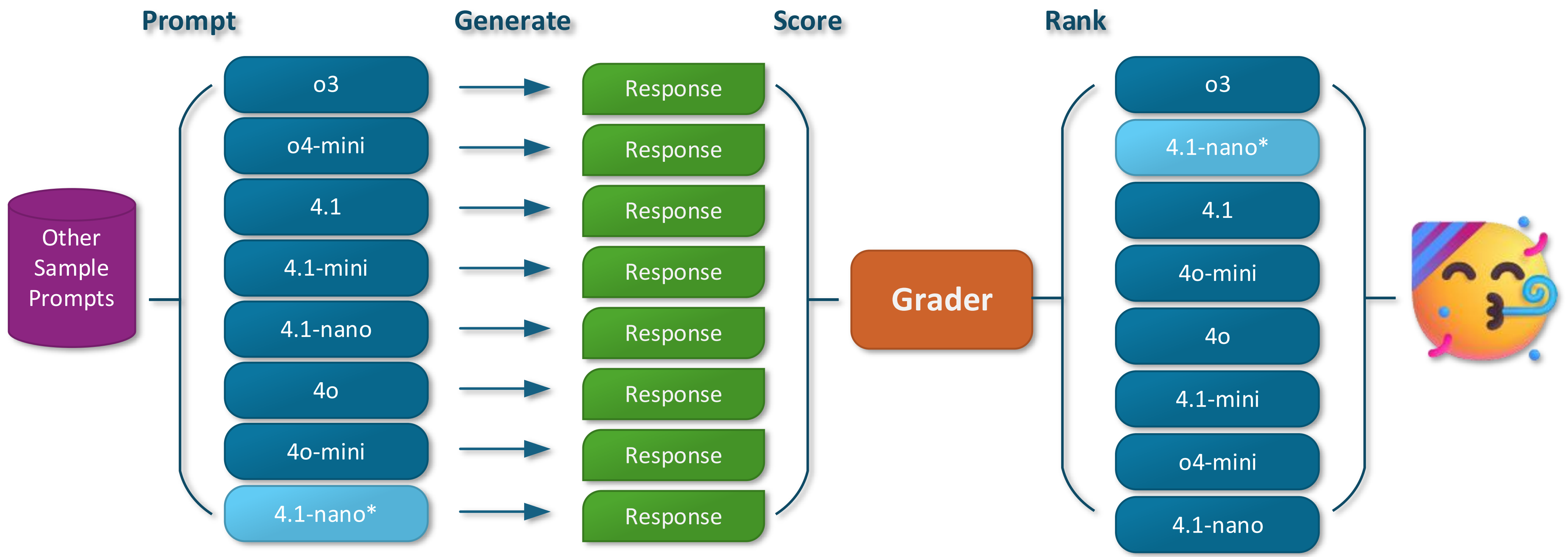
- **Developer Tier** works great as this is:
  - A short-lived need for model candidate evaluation
  - Doesn’t need high throughput

Deployment info

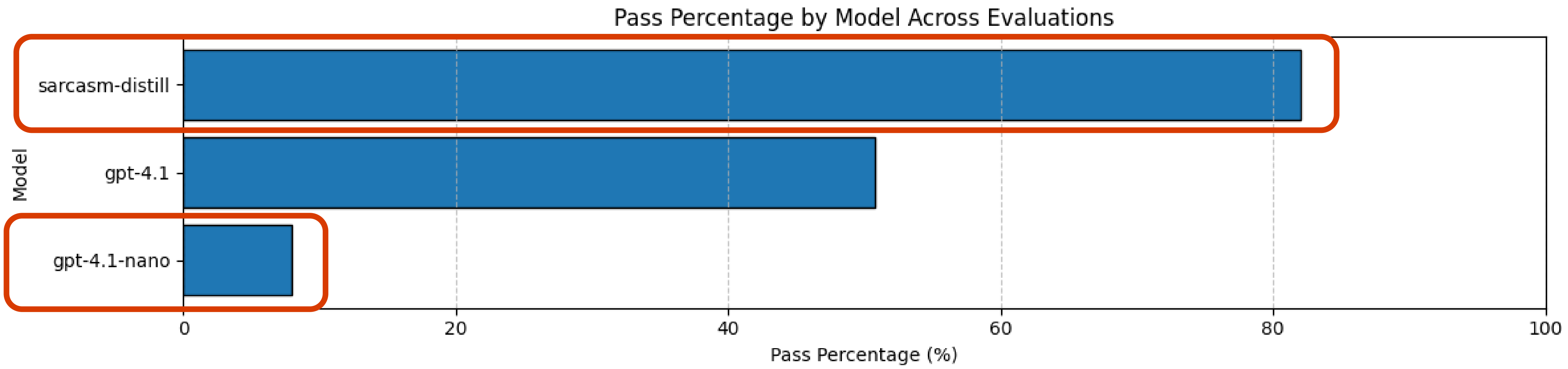
Name sarcasm-distilled-o3-sarcasm-2aa2befa	**Provisioning state** Succeeded
Deployment type Developer (Preview)	**Created on** 2025-06-20T16:50:24.4358321Z
Created by davevoutila@microsoft.com	**Modified on** Jun 20, 2025 12:50 PM
Modified by davevoutila@microsoft.com	**Version upgrade policy** Model version will not be automatically upgraded
Rate limit (Tokens per minute) 250,000	**Rate limit (Requests per minute)** 250
Model name gpt-4.1-nano-2025-04-14.ft-694c4afe08ce47269b9467b82814b074-o3-sarcasm-2aa2befa	**Model version** 1
Date created Jun 20, 2025 12:38 PM	**Date updated** Jun 20, 2025 12:50 PM
Model retirement date Apr 10, 2026 8:00 PM	

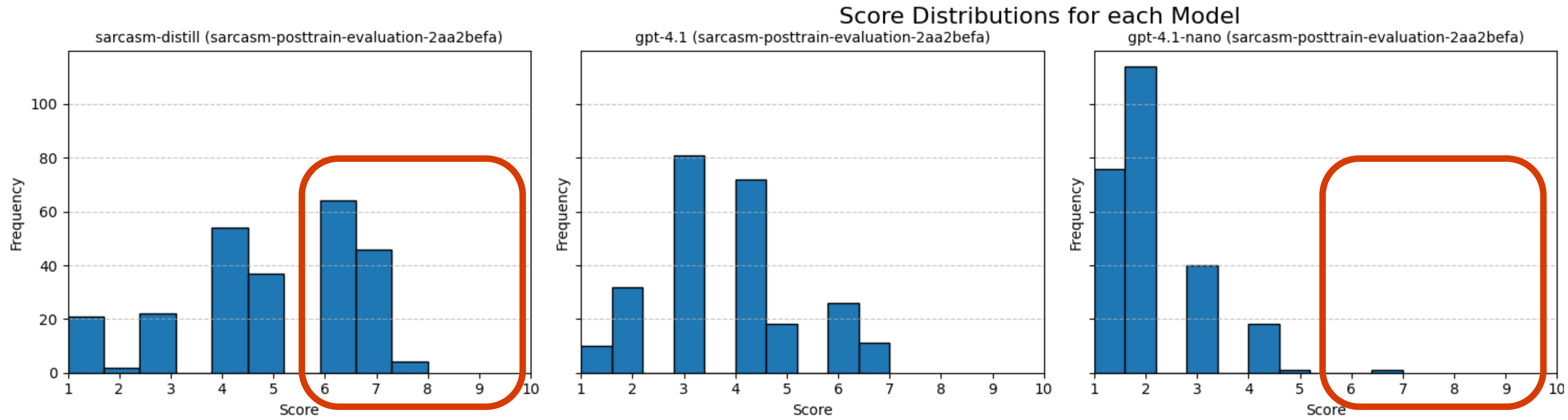
Step 3: Model Candidate Validation

- Now we benchmark the **distilled model** with the others to validate we've made a measurable improvement…or be sad if we didn't!

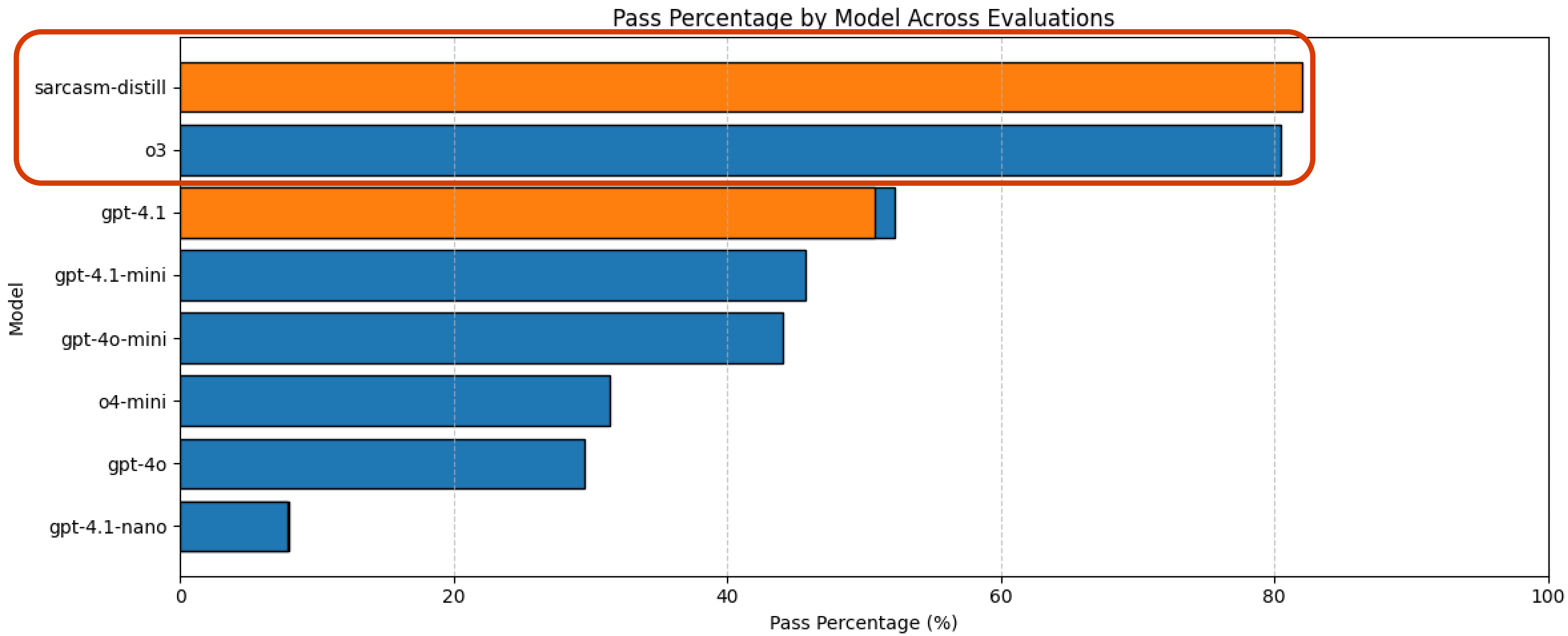


Let's test now against 4.1-nano and a 4.1 control.





And now compare against the original results



Orange bars are post-training Eval, Blue are the original baseline Eval
* gpt-4.1-nano post-training is sort of hidden behind the blue bar

Let’s look at the notebook!

Time to jump into VSCode.