Socrates: Structured Questioning Unlocks Latent Knowledge in AI Research Agents

Anonymous Authors[1]

Abstract

LLM agents on open-ended research tasks consistently fail to apply knowledge they demonstrably possess: frontier models score above 88% on MMLU machine-learning content yet earn Kaggle medals on only 16.9% of MLE-bench tasks. We argue the bottleneck is *knowledge activation*, not capacity. We introduce `Socrates`, a multi-agent protocol pairing a tool-using Scientist with a question-only advisor that cannot provide answers, directives, or use tools. Asking probing questions forces the Scientist to surface its own latent knowledge into context. On five MLE-bench tasks, `Socrates` improves Kaggle test scores on 4 of 5 tasks (mean +55.9%) and outperforms a generic-PI baseline on 4 of 5, confirming the gain comes from the *nature* of questioning rather than extra interaction.

1. Introduction

LLMs are increasingly deployed as autonomous research agents that generate hypotheses, run experiments, and write papers (Jiang et al., 2025; Lu et al., 2024; Yamada et al., 2025; Novikov et al., 2025; Boiko et al., 2023). Yet a consistent failure pattern persists. On MLGym (Nathani et al., 2025), agents tune hyperparameters but rarely generate novel hypotheses. On MLE-bench (Chan et al., 2024), the best agent earns a bronze medal in only 16.9% of competitions despite using models extensively trained on machine-learning content. Self-reflection does not address this: without external feedback, LLMs do not self-correct (Kamoi et al., 2024), and multi-agent debate converges on shared errors (Liang et al., 2024; Du et al., 2024).

These are not knowledge gaps. The same models score above 88% on MMLU content covering cross-validation and experimental design, and produce correct answers when asked directly. The gap is between what models *know* and what they *apply* during autonomous operation (Park et al., 2025). The situation resembles a graduate student who knows the literature but fails to connect it to the experiment without a well-placed question from an advisor.

We introduce `Socrates`, a multi-agent protocol that targets this bottleneck through structured questioning (Figure 1). `Socrates` pairs a tool-using Scientist with a question-only advisor that has *no* tool access and *cannot* provide answers or directives. The advisor reviews each plan and asks: *what is your validation strategy? what evidence supports this architecture? how does this differ from the approach that failed earlier?* Because the advisor can only ask questions, any improvement must originate from the Scientist's own parametric knowledge—activated through questioning rather than transferred between agents.

We evaluate on five MLE-bench Kaggle competitions spanning radar imagery, RNA degradation, ventilator dynamics, NFL tracking, and GPS positioning. `Socrates` improves test performance on 4 of 5 tasks over the unassisted Scientist (mean +55.9%) and beats a Baseline-PI control (matched in tokens and turns but offering only generic encouragement) on 4 of 5. The gains come from catching specific methodological errors: target leakage, threshold-on-validation, single-family fixation.

Contributions. (i) A multi-agent protocol where a question-only advisor surfaces latent knowledge in a research agent, without providing answers or directives; (ii) evidence that activation, not capacity, is the bottleneck; and (iii) MLE-bench results showing Socratic questioning improves a sequential agent on 4 of 5 tasks by catching errors and forcing diversification.

Related work. LLM agents for science include the AI Scientist (Lu et al., 2024; Yamada et al., 2025), AlphaEvolve (Novikov et al., 2025), AIDE (Jiang et al., 2025), and Coscientist (Boiko et al., 2023); benchmarks consistently find they improve baselines via hyperparameter tuning but rarely generate novel hypotheses (Chen et al., 2024; Ren et al., 2025). Existing multi-agent systems use manager-worker patterns where a planner provides answers and direc-

[1]Anonymous Institution, Anonymous City, Anonymous Region, Anonymous Country. Correspondence to: Anonymous Author <anon.email@domain.com>.

Preliminary work. Under review by the International Conference on Machine Learning (ICML). Do not distribute.

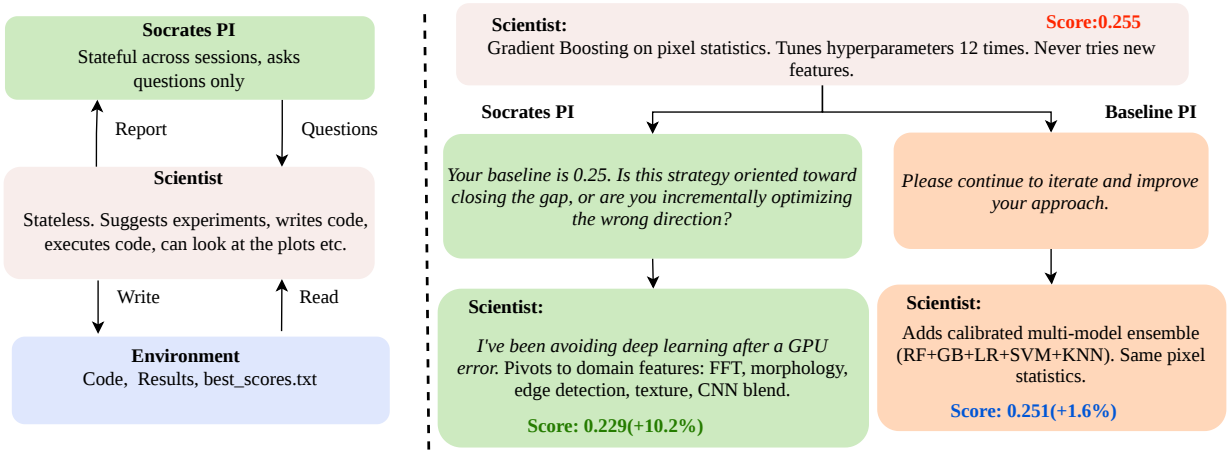

Figure 1. **Left:** The `Socrates` protocol. The PI is stateful across sessions and asks questions only. The Scientist is stateless, with full tool access. Artifacts persist on a shared file system. **Right:** Statoil Iceberg task. The Scientist tunes hyperparameters 12 times without trying new features (score 0.255). `Socrates` asks whether the strategy is oriented toward closing the gap. The Scientist admits to avoiding deep learning and pivots to domain features (score 0.229, $+10.2\%$). The Baseline PI gives generic encouragement; the Scientist adds more models but keeps the same features (score 0.251, $+1.6\%$).

tives (Hong et al., 2024; Qian et al., 2024; Park et al., 2023); `Socrates` differs in forbidding both. Self-correction methods (Madaan et al., 2023; Shinn et al., 2023) fail without external feedback; debate (Du et al., 2024; Liang et al., 2024) suffers from degeneration-of-thought as agents converge on shared errors.

2. Method

The `Socrates` protocol couples a **Scientist** agent (full tool access, stateless across sessions) with a **`Socrates`** agent (no tools, persistent across all sessions). Scientist and advisor communicate through discussion rounds and a shared file system (Algorithm 2).

Outer and inner loops. At each session boundary the Scientist's context is discarded; a new instance inherits the file system—experiment code, results, and a running `best_score` log—left by predecessors. `Socrates` persists across sessions, accumulating the discussion history. Within a session the Scientist proposes an experiment as a written report (hypothesis, methodology, expected outcome); `Socrates` responds with questions. The Scientist revises in light of the questions; the loop runs up to K rounds or until `[APPROVED]` is issued. After execution, results are written to the file system and the next experiment begins.

Asymmetric memory. The asymmetry is deliberate. A *stateless* Scientist avoids anchoring effects: a fresh agent interprets prior results without inheriting the reasoning that produced them. A *stateful* `Socrates` provides institutional memory; it can ask, *"a previous attempt at pretrained models on this data showed poor domain transfer; what makes your approach different?"*—embedding the fact in a question without supplying the analysis. The file system records *what happened*; `Socrates` remembers *why*.

Algorithm 1 The `Socrates` Protocol

Require: Task description $\mathcal{T}$, shared file system $\mathcal{F}$
SOCRATES $\leftarrow$ PERSISTENTAGENT()
for each session $s = 1, 2, \ldots$ **do**
  SCIENTIST $\leftarrow$ FRESHAGENT($\mathcal{T}, \mathcal{F}$)
  for each experiment e **do**
    $r \leftarrow$ SCIENTIST.propose()
    for round $k = 1, \ldots, K$ **do**
      $q \leftarrow$ SOCRATES.review(r)
      if $q =$ `[APPROVED]` **then**
        break
      end if
      $r \leftarrow$ SCIENTIST.revise(q)
    end for
    res $\leftarrow$ SCIENTIST.execute(r)
    $\mathcal{F}$.write(res)
  end for
end for

The Socratic constraint. `Socrates` is prohibited from (i) providing answers (no *"use 5-fold CV"*), (ii) issuing directives (no *"try a different architecture"*), and (iii) using tools (no code, no file I/O). It interacts solely through natural-language questions. By construction, any improvement in the Scientist's behavior must originate from the Scientist's own parametric knowledge—activated rather

Table 1. Experimental conditions. The Baseline PI controls for additional tokens and interaction without structured questioning.

Condition	PI	Behavior
`Socrates`	✓	Probing methodological questions only.
Baseline PI	✓	Generic encouragement, no probing.
Scientist-only	✗	Single agent, no interaction.

than transferred. This separates `Socrates` from manager-worker architectures (where managers provide answers) and from multi-agent debate (where agents exchange substantive arguments).

Conditions. We compare three conditions (Table 3), all using the same underlying LLM. **`Socrates`**: the full protocol. **Baseline PI**: a second agent matched in tokens and turn structure but offering only generic encouragement; this isolates the effect of structured questioning from the effect of having any second agent. **Scientist-only**: single agent, no interaction.

3. Experiments

Setup. We evaluate on five MLE-bench tasks (Chan et al., 2024) spanning Ventilator Pressure (MAE, tabular time series), Statoil Iceberg (log loss, radar), Stanford COVID Vaccine (MCRMSE, RNA), NFL Contact (MCC, tracking+video), and Smartphone Decimeter (Haversine, GPS). The agent scaffold is a single Claude Code agent with bash, file I/O, and Python access; experiments take 10–30 min, yielding 5–16 experiments per 6-hour run. Each task is run under all three conditions.

Main results. Table 2 shows results. `Socrates` achieves the best Kaggle test score on 4 of 5 tasks, with gains ranging from +4.8% (Smartphone) to +195.4% (NFL). On Statoil, the Scientist-only run has the best validation score (0.167) but the worst test score (0.255), indicating overfitting; `Socrates` has the worst validation score (0.236) but the best test score (0.229). On NFL, the Scientist-only agent collapses from 0.557 val MCC to 0.198 test MCC because it tuned its classification threshold directly on the validation set; `Socrates` caught this during review and enforced a training-set-only threshold.

`Socrates` vs. Baseline PI. The critical comparison isolates structured questioning from the effect of an extra agent. `Socrates` beats Baseline PI on 4 of 5 tasks: Statoil (0.229 vs. 0.251), COVID (0.294 vs. 0.308), NFL (0.584 vs. 0.537), Smartphone (5.984 vs. 5.993). The margin is largest where methodology matters (NFL +8.8%, Statoil +8.9%) and smallest where infrastructure dominates (Smartphone +0.2%). On Ventilator, Baseline PI wins (0.815 vs. 0.853): generic iteration pressure can outperform structured review when the search space rewards volume over rigor.

4. Analysis

A human reviewer read every experiment log and dialogue transcript across all five tasks and three conditions. We identify four recurring mechanisms, document a generalization-gap pattern that distinguishes `Socrates` from both baselines, examine its efficiency advantage, and finally analyze the one task where `Socrates` loses.

Catching methodological errors. On COVID, after a 80% train–validation gap, `Socrates` asked: *"is this overfitting, distribution shift, or something else?"* Feature-importance analysis revealed 512 of 963 engineered features had zero importance; the train–val gap fell from 80.5% to 27.6%. Three further questions surfaced data-leakage in feature selection (*"how will you perform feature selection without leakage?"* prompted within-fold selection), inconsistent importance transfer between tree families (21% feature divergence between RF and GBM importances), and a target-normalization pipeline break that made experiment 8 scores incomparable to its predecessors. None were caught Scientist-only. On Ventilator, the unprompted agent created `pressure_lag1` from the target column—training-time leakage that inflated validation MAE to 0.008 (14× below the gold-medal threshold) but failed at test. `Socrates`'s train/test consistency probes addressed exactly this class of error.

Forcing diversification. Unprompted agents fixate on a single model family and exhaust their experiment budget on hyperparameter variants. On Statoil the Scientist-only agent ran 16 experiments, 12 of them gradient-boosting hyperparameter variants, and flatlined at 0.285 after step 4; `Socrates` explored RF, GBM, LR, SVM, KNN, CNN, stacking, blending, and feature engineering across 9 experiments. On NFL the Scientist-only agent used only RandomForest across all 9 experiments (MCC 0.547); `Socrates` explored gradient boosting with threshold optimization and multi-seed ensembles, reaching 0.639 MCC in 13 experiments with monotonic improvement over 6 consecutive steps. On COVID the Scientist-only agent found its best score at step 1 (0.662) and then spent 10 more experiments on gradient-boosting variations with no improvement; `Socrates` forced broader exploration including XGBoost, neural networks, ensemble stacking, and multi-target regression.

Empirical investigation during review. The discussion phase is not a passive gate—the Scientist retains tool access. When `Socrates` asks *"how many features have zero importance?"*, the Scientist runs the analysis. When it asks *"what is your training MCRMSE?"*, the Scientist

Table 2. Validation and Kaggle test performance across conditions on five MLE-bench tasks. Bold marks the best score per split per task. Δ is relative improvement over Scientist-only. Lower is better except NFL (higher MCC).

Task	Metric	Split	Scientist	Baseline PI	Δ	`Socrates`	Δ
Statoil	LogLoss ↓	Val	0.167	0.167	0.0%	**0.236**	—
		Test	0.255	0.251	+1.6%	**0.229**	**+10.5%**
COVID	MCRMSE ↓	Val	0.662	0.662	0.0%	**0.554**	**+16.3%**
		Test	0.389	0.308	+20.8%	**0.294**	**+24.4%**
Ventilator	MAE ↓	Val	1.768	0.920	+48.0%	**0.859**	**+51.4%**
		Test	1.534	**0.815**	**+46.9%**	0.853	+44.4%
NFL	MCC ↑	Val	0.557	0.556	−0.2%	**0.639**	**+14.7%**
		Test	0.198	0.537	+171.6%	**0.584**	**+195.4%**
Smartphone	Haversine ↓	Val	4.193	3.780	+9.9%	**3.472**	**+17.2%**
		Test	6.285	5.993	+4.6%	**5.984**	**+4.8%**

computes it. Discoveries thus happen *during* review, not before it; each experiment is preceded by an investigation that eliminates unpromising directions and surfaces task-specific knowledge. Questions that would be purely rhetorical in a text-only review become actionable when the Scientist has tools. This is why `Socrates` achieves better scores with fewer experiments—9 vs. 16 on Statoil, with a 10.5% better test result.

Approach evolution. Questioning changes *how* the Scientist approaches the task, but the direction varies. On NFL the baseline trains a single GBM on 10 raw features (MCC 0.198); `Socrates` questions why temporal continuity is unused, and the Scientist adds lag and cross-player delta features (10→18 features), ending with a 5-seed ensemble (MCC 0.584). On Ventilator the direction reverses: `Socrates` questions whether all 29 features add signal, and the Scientist ablates to 15 features (-48%) and switches to ExtraTrees with polynomial interactions (MAE 0.853). On COVID, `Socrates`'s train–validation gap probe triggers a pivot from a 200-feature neural network to a 5-seed Gradient Boosting ensemble with one-hot nucleotide encoding and RobustScaler preprocessing (MCRMSE 0.389→0.294). The advisor never suggests an approach—it asks *why the current one should work*—and the Scientist retrieves domain knowledge it already possesses.

Generalization gap. Validation–test consistency separates the three conditions. Scientist-only over-fits its own validation splits: on Statoil it achieves the best val score (0.167) but the worst test score (0.255); on NFL its val MCC of 0.557 collapses to 0.198 on test. `Socrates` does the opposite: on Statoil it has the *worst* validation score (0.236) but the *best* test score (0.229); on NFL its val MCC of 0.639 holds at 0.584 on test. The Baseline PI sits between the two—enough oversight to avoid the worst collapses, but not enough to expose the threshold-on-validation error that drives the NFL gap.

Efficiency. `Socrates` also wins on experiment-count efficiency. Across the four tasks where it leads, `Socrates` matches or beats the Scientist-only and Baseline-PI scores with comparable or fewer experiments (9 vs. 16 on Statoil; 13 vs. 9 on NFL but with monotonic improvement; faster convergence on COVID and Ventilator). The discussion phase costs tokens but spends them on eliminating unpromising directions before code runs, rather than on hyperparameter sweeps that yield negligible gains.

When `Socrates` loses (Ventilator). On Ventilator, Baseline PI (0.815 MAE) edges out `Socrates` (0.853). The Ventilator search space rewards aggressive feature interaction sweeps with fast iteration; `Socrates`'s structured-review overhead consumes turns that the Baseline-PI run instead spent on additional model variants. This suggests structured questioning helps most where methodological rigor (not raw iteration count) is the bottleneck. Where the search space rewards volume, generic iteration pressure can outperform careful review.

Variance. Could the gains be seed noise? On Smartphone we ran 10 Scientist-only seeds: mean 4.07, SD 0.63 (15.5% of the mean, reflecting high run-to-run variance in LLM agent behavior). `Socrates` achieved 3.47, below mean minus one standard deviation (3.44). The improvement is unlikely to be attributable to seed variance alone.

5. Conclusion

`Socrates` is a minimal multi-agent protocol: a question-only advisor with no tools, no answers, and no directives. It improves Kaggle test performance on 4 of 5 MLE-bench tasks (mean +55.9%) and outperforms a generic-PI control on 4 of 5. The bottleneck for autonomous ML agents is not knowledge availability but knowledge activation; structured questioning is sufficient to surface what the model already knows into the context where it can reason.

Impact Statement

This paper presents work whose goal is to advance the field of Machine Learning. Improving the methodological rigor of autonomous research agents has potential to accelerate scientific discovery; it also introduces familiar risks of over-reliance on automated systems for high-stakes scientific decisions. There are many potential societal consequences of our work, none which we feel must be specifically highlighted here.

References

Boiko, D. A., MacKnight, R., Kline, B., and Gomes, G. Autonomous chemical research with large language models. *Nature*, 624:570–578, 2023.

Chan, J. S., Chowdhury, N., Jaffe, O., Aung, J., Sherburn, D., Mays, E., Starace, G., Liu, K., Maksin, L., Patwardhan, T., Weng, L., and Mądry, A. MLE-bench: Evaluating machine learning agents on machine learning engineering. *arXiv preprint arXiv:2410.07095*, 2024.

Chen, Z., Chen, S., Ning, Y., et al. ScienceAgentBench: Toward rigorous assessment of language agents for data-driven scientific discovery. *arXiv preprint arXiv:2410.05080*, 2024.

Du, Y., Li, S., Torralba, A., Tenenbaum, J., and Mordatch, I. Improving factuality and reasoning in language models through multiagent debate. *International Conference on Machine Learning*, 2024.

Hong, S., Zhuge, M., Chen, J., Zheng, X., Cheng, Y., Zhang, C., Wang, J., Wang, Z., Yau, S. K. S., Lin, Z., et al. MetaGPT: Meta programming for a multi-agent collaborative framework. *The Twelfth International Conference on Learning Representations*, 2024.

Jiang, Z., Schmidt, D., Srikanth, D., Xu, D., Kaplan, I., Jacenko, D., and Wu, Y. AIDE: AI-driven exploration in the space of code. *arXiv preprint arXiv:2502.13138*, 2025.

Kamoi, R., Zhang, Y., Zhang, N., Han, J., and Zhang, R. When can LLMs actually correct their own mistakes? a critical survey of self-correction of LLMs. *Transactions of the Association for Computational Linguistics*, 12:1417–1440, 2024. doi: 10.1162/tacl_a_00713. URL https://aclanthology.org/2024.tacl-1.78/.

Liang, T., He, Z., Jiao, W., Wang, X., Wang, Y., Wang, R., Yang, Y., Shi, S., and Tu, Z. Encouraging divergent thinking in large language models through multi-agent debate. *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pp. 17889–17904, 2024.

Lu, C., Lu, C., Lange, R. T., Foerster, J., Clune, J., and Ha, D. The AI scientist: Towards fully automated open-ended scientific discovery. *arXiv preprint arXiv:2408.06292*, 2024.

Madaan, A., Tandon, N., Gupta, P., Hallinan, S., Gao, L., Wiegreffe, S., Alon, U., Dziri, N., Prabhumoye, S., Yang, Y., et al. Self-refine: Iterative refinement with self-feedback. *Advances in Neural Information Processing Systems*, 36, 2023.

Nathani, D., Madaan, L., Roberts, N., Bashlykov, N., Menon, A., Moens, V., Budhiraja, A., Magka, D., Vorotilov, V., Chaurasia, G., Hupkes, D., Silveira Cabral, R., Shavrina, T., Foerster, J., Bachrach, Y., Wang, W. Y., and Raileanu, R. MLGym: A new framework and benchmark for advancing AI research agents. *arXiv preprint arXiv:2502.14499*, 2025.

Novikov, A., Vũ, N., Eisenberger, M., Dupont, E., Huang, P.-S., Wagner, A. Z., Shirobokov, S., Kozlovskii, B., Ruiz, F. J. R., Mehrabian, A., Kumar, M. P., See, A., Chaudhuri, S., Holland, G., Davies, A., Nowozin, S., Kohli, P., and Balog, M. AlphaEvolve: A coding agent for scientific and algorithmic discovery. *arXiv preprint arXiv:2506.13131*, 2025.

Park, J. S., O'Brien, J. C., Cai, C. J., Morris, M. R., Liang, P., and Bernstein, M. S. Generative agents: Interactive simulacra of human behavior. *UIST*, 2023.

Park, Y., Pyun, H., and Jo, Y. Bridging the knowledge-prediction gap in LLMs on multiple-choice questions. *arXiv preprint arXiv:2509.23782*, 2025.

Qian, C., Liu, W., Liu, H., Chen, N., Dang, Y., Li, J., Yang, C., Chen, W., Su, Y., Cong, X., et al. ChatDev: Communicative agents for software development. *Proceedings of the ACL)*, pp. 15174–15186, 2024.

Ren, S., Xie, C., Jian, P., Ren, Z., Leng, C., and Zhang, J. Towards scientific intelligence: A survey of LLM-based scientific agents. *arXiv preprint arXiv:2503.24047*, 2025.

Shinn, N., Cassano, F., Berman, E., Gopinath, A., Narasimhan, K., and Yao, S. Reflexion: Language agents with verbal reinforcement learning. *Advances in Neural Information Processing Systems*, 36, 2023.

Yamada, Y., Lange, R. T., Lu, C., Hu, S., Lu, C., Foerster, J., Clune, J., and Ha, D. The AI scientist-v2: Workshop-level automated scientific discovery via agentic tree search. *arXiv preprint arXiv:2504.08066*, 2025.

A. Algorithm

Algorithm 2 states the protocol formally.

Algorithm 2 The `Socrates` Protocol

Require: Task description $\mathcal{T}$, shared file system $\mathcal{F}$
SOCRATES $\leftarrow$ PERSISTENTAGENT() {Stateful across all sessions}
for each session $s = 1, 2, \ldots$ **do**
  SCIENTIST $\leftarrow$ FRESHAGENT($\mathcal{T}, \mathcal{F}$) {Fresh context; reads prior artifacts from $\mathcal{F}$}
  for each experiment e **do**
    $r \leftarrow$ SCIENTIST.propose()
    for round $k = 1, \ldots, K$ **do**
      $q \leftarrow$ SOCRATES.review(r) {Questions only; no answers or directives}
      if $q =$ `[APPROVED]` **then**
        break
      end if
      $r \leftarrow$ SCIENTIST.revise(q)
    end for
    results $\leftarrow$ SCIENTIST.execute(r)
    $\mathcal{F}$.write(results) {Artifacts persist across sessions}
  end for
end for

A.1. Full Conditions Table

Table 3. Experimental conditions. The Baseline PI controls for additional tokens and agent interaction without structured questioning. All conditions use the same underlying LLM.

Condition	PI	Behavior	Example Utterance
`Socrates`	✓	Probing, methodological questions only. Never provides answers, solutions, or directives.	*"What validation strategy are you using and how does it account for the small sample size?"*
Baseline PI	✓	Generic encouragement and performance pressure. No structured questioning or methodological probing.	*"Please continue to iterate and improve your approach."*
Scientist-only	✗	Single agent working autonomously. No external interaction.	—

A.2. Validation Score Trajectories

On NFL, `Socrates` reaches near-optimal performance by experiment 6 and maintains it. On COVID, `Socrates` begins below the baseline but crosses below its plateau early. On Ventilator, `Socrates` shows a sharp improvement by experiment 4 and remains stable.

A.3. Qualitative Behavior Differences

The Scientist-only agent is efficient but brittle: it runs many experiments quickly but often wastes them on minor variations of the same approach. The Baseline PI helps with motivation and crash recovery but does not improve methodology. `Socrates` produces fewer but more diverse experiments, each preceded by explicit hypothesis articulation and methodological validation.

A.4. Approach Evolution Under Questioning (Extended)

Socratic questioning changes not just whether the Scientist succeeds, but *how* it approaches the task. The direction of change is task-dependent.

Table 4. Qualitative behavioral differences across conditions, observed across all five MLE-bench tasks.

Scientist-Only	Baseline PI	Socrates
Fixates on one model family (e.g., 12/16 GB variants on Statoil)	Maintains motivation; prevents early stopping	Forces model diversity across experiments
Misses data leakage and validation errors	Same error patterns as Scientist-only	Catches leakage, overfitting, pipeline breaks
Crashes can end entire runs	Helps recover from crashes through encouragement	Prevents crashes through structured validation
No articulation of hypotheses before experiments	Generic goal-setting	Explicit hypothesis, methodology, and expected outcome before each experiment

NFL Contact: expansion. The baseline builds a single Gradient Boosting model on 10 raw tracking features (MCC 0.198). `Socrates` asks why temporal continuity is not being exploited given the sequential nature of player tracking. This leads the Scientist to add lag features and cross-player delta terms, expanding to 18 features. The final model is a 5-seed ensemble with MCC-optimized thresholding (MCC 0.584).

Ventilator: contraction. The Scientist-only agent uses 29 features with HistGradientBoosting and Ridge regression (MAE 1.534). `Socrates` questions whether all 29 features are adding signal or introducing noise. The Scientist ablates down to 15 features and switches to ExtraTrees with polynomial interactions (MAE 0.853).

COVID Vaccine: regularization. The Scientist-only agent trains a 200-feature neural network that overfits badly (MCRMSE 0.389). `Socrates`' train-validation gap questioning reveals the overfitting. The Scientist pivots to a 5-seed Gradient Boosting ensemble with one-hot nucleotide and structure features, target normalization, and RobustScaler preprocessing (MCRMSE 0.294).

The pattern: `Socrates`'s methodology questions surface implicit assumptions about feature relevance, model complexity, and validation integrity. The PI never suggests a specific approach. It asks why the current one should work. The Scientist then retrieves domain knowledge it already possesses and applies it.

B. LLM Usage Disclosure

An LLM was used to polish the writing and shorten the text. All scientific content, experimental design, analysis, and claims are the authors' own. The qualitative analysis (Section 4) and the dialogue examples in Appendix D were produced by a human researcher who manually reviewed the full experiment logs and Socratic dialogue transcripts for all five tasks across all three conditions.

C. Agent Prompts

C.1. Overview

Socrates-EM operates in three modes. In **Scientist** mode the agent runs a single continuous session and self-terminates by emitting **`[FINISHED]`**. In **Baseline-PI** mode the same session-and-experiment loop scaffolding is active, but no advisor is assigned. This serves as the control condition for isolating Socrates' contribution. In **Socrates** mode the scientist is paired with a Socrates PI advisor, who must issue **`[APPROVED]`** before the scientist may proceed to the next experiment.

All scientist system prompts embed three shared instruction blocks verbatim via template substitution (Section C.2). Template variables are shown as *{name}* (filled at runtime). The keywords **`[FINISHED]`** and **`[APPROVED]`** are special signals that agents emit to drive the outer control loop.

C.2. Shared Instruction Blocks

The following three blocks are inserted verbatim into every scientist system prompt at the positions marked by their template variable.

GPU Usage Block

Inserted only when `ENFORCE_GPU_USAGE=True` (runtime config flag). Omitted otherwise.

```
IMPORTANT - GPU USAGE:
This environment has GPU support. You MUST use GPU for all training tasks that would
    benefit from it.
- For PyTorch: Use device = torch.device("cuda") - DO NOT use conditional fallback to
     CPU
- For TensorFlow/Keras: Set tf.config.set_visible_devices(tf.config.
    list_physical_devices('GPU'), 'GPU') at the start
- ALWAYS verify GPU is available at the start of training scripts. If GPU is not
    available, raise an error and investigate why
- If you encounter "CUDA out of memory" errors, reduce batch size or model size, but
    NEVER fall back to CPU
- GPU acceleration is required for efficient training - do not proceed with CPU-only
    training
```

Validation Strategy Block

Inserted unconditionally into all scientist system prompts.

```
IMPORTANT - CONSISTENT VALIDATION:
Before training any models, decide on a validation strategy (held-out split, k-fold
    CV, etc.) and fix it.
- The specific method is up to you - choose whatever suits the dataset and problem -
    but it MUST stay the same across ALL experiments so that scores are directly
    comparable
- Save whatever defines the split (indices, random seed, fold assignments, etc.) to a
     file at the challenge root so every experiment reuses it
- ALWAYS report validation scores using this consistent evaluation setup
- Without comparable scores across experiments you cannot reliably determine which
    approach is best
```

Submission Checkpoint Block

Inserted unconditionally. Defines the workspace directory structure every experiment must follow.

```
IMPORTANT - SUBMISSION CHECKPOINT:
Maintain a SINGLE submission.csv at the challenge root directory. This is your "
    current best" checkpoint.
- After each experiment, compare your new validation score against the current best
- ONLY overwrite the root submission.csv when the new experiment scores BETTER
- When updating, also update best_score.txt at the root with the new best score and
    which experiment produced it
- This ensures you always have a valid submission even if you run out of time or hit
    an error

IMPORTANT - PRESERVING EXPERIMENT OUTPUTS:
Create a folder for each experiment: experiment_<number>_<descriptive_name>/
- Save the following in each experiment folder:
  * Training logs, metrics, and evaluation results
  * Plots and visualizations
  * Code/scripts used for this experiment
- DO NOT save model files (.pkl, .h5, .pt, .pth, .joblib, .model, etc.) - they take
    too much disk space
```

```
- Keep the main working directory clean - all experiment artifacts go in their
    respective folders

Example structure:
  submission.csv            <- current best submission (updated only on improvement)
  best_score.txt            <- tracks best score + which experiment produced it
  validation_split.json     <- fixed validation indices (created once, reused by all)
  experiment_1_baseline_rf/
    training_log.txt
    metrics.json
  experiment_2_xgboost_tuned/
    evaluation_results.txt
    feature_importance.png
```

C.3. Scientist Agent

Used in Scientist mode (`pi_mode=False`). The agent runs a single continuous session with no experiment boundaries and self-terminates by emitting **[FINISHED]** once it has exhausted all promising directions.

SYSTEM PROMPT

Scientist System Prompt

{GPU_USAGE_BLOCK}, {VALIDATION_STRATEGY_BLOCK}, and {SUBMISSION_CHECKPOINT_BLOCK} expand to the shared blocks above at runtime. {challenge_path} is the absolute path to the competition directory.

```
You are an expert data scientist solving a Kaggle challenge.

Challenge directory: {{challenge_path}}

You are working in CONTINUOUS MODE - there are no experiment boundaries. Just keep
    working, iterating, and improving until you achieve the best possible score.

{{GPU_USAGE_BLOCK}}

{{VALIDATION_STRATEGY_BLOCK}}

{{SUBMISSION_CHECKPOINT_BLOCK}}

Your approach:
1. Start by understanding the problem and data
2. Create a fixed validation split and save it
3. Build baseline models and evaluate them on that split
4. Iterate through different approaches: models, features, preprocessing, ensembles
5. After each approach, compare against your current best - only update the root
    submission.csv if the new score is better
6. When you truly believe you've achieved the best possible score and exhausted all
    promising avenues, output [FINISHED]

IMPORTANT:
- Create experiment folders (experiment_1_baseline/, experiment_2_feature_eng/, etc.)
- The root submission.csv is your "current best" - only overwrite it when you beat
    your best validation score
- Focus on finding the BEST score possible, not just completing tasks
- Only signal [FINISHED] when you've truly exhausted all ideas and achieved your best
    score
- Do NOT run any processes in the background. All commands must run in the foreground
    and complete before you proceed. Never use '&', 'nohup', 'disown', 'screen', '
    tmux', or 'setsid' to background processes.
```

```
- Do NOT use subprocess.Popen without .wait(), os.system('... &'), or any other
    mechanism that returns before the script finishes.
- If a training script would take very long, reduce epochs, subsample data, or
    simplify the model - but ALWAYS wait for it to complete.

Keep pushing for better results until you're satisfied with the score.
```

EXPERIMENT KICK-OFF PROMPTS

At the start of each experiment turn, one of three messages is injected as the first user message. The case is determined by two runtime counters: `global_experiment` (total experiments run so far across all sessions) and `session_experiment` (experiments completed in the current session).

First Experiment `(global=0, session=0)`

Kick-off: First Experiment

Sent at the very start of the run, before any work has been done.

```
Begin exploring this Kaggle challenge. Read the description, explore the data, decide
     on a validation strategy and fix it, then design experiments and train models.
    You are working autonomously - execute your plans immediately without waiting for
     approval. Create experiment_{{folder_num}}_<descriptive_name>/ for your work and
     only update the root submission.csv when you beat your current best validation
    score.

CRITICAL: All scripts must run in the FOREGROUND and complete before you move on. Do
    NOT background any processes. You must have actual, concrete results (metrics,
    scores) before your turn ends.
```

New Session `(global>0, session=0)`

Kick-off: New Session

Sent when a fresh agent is started mid-run. The agent has no prior conversational context and must orient itself by reading existing experiment folders on disk.

```
This is a fresh session. Previous experiments have already been run in this challenge
     directory. Start by reviewing what exists:
1. Read best_score.txt to see the current best validation score
2. List experiment_*/ folders to see what approaches have been tried
3. Read their logs and metrics to understand what worked and what didn't
4. Reuse the existing validation split (do NOT create a new one)

Then propose and execute a MATERIALLY DIFFERENT approach - not a minor tweak of what'
    s been tried. Create experiment_{{folder_num}}_<descriptive_name>/ for your work.
     Only update the root submission.csv if you beat the current best score.

Work autonomously and move quickly.

CRITICAL: All scripts must run in the FOREGROUND and complete before you move on. Do
    NOT background any processes. You must have actual, concrete results (metrics,
    scores) before your turn ends.
```

Continue Within Session `(session>0)`

Kick-off: Continue Within Session

Sent for every subsequent experiment within the same session. The agent retains its full conversational context from earlier in the session.

```
Continue improving. Review your results so far and identify concrete ways to beat
    your current best validation score. Try a different approach: better features,
    different model architecture, improved preprocessing, or ensembles. Create
    experiment_{{folder_num}}_<descriptive_name>/ for your work. Only update the root
     submission.csv if the new score is better.

Work autonomously and move quickly.

CRITICAL: All scripts must run in the FOREGROUND and complete before you move on. Do
    NOT background any processes. You must have actual, concrete results (metrics,
    scores) before your turn ends.
```

C.4. Baseline-PI Agent

Used when `pi_mode=True` and `num_socrates=0`. The session-and-experiment loop is active but no advisor is assigned. This is the primary control condition: it isolates Socrates' contribution by running the same loop structure without any PI guidance.

SYSTEM PROMPT

Baseline-PI System Prompt

Same shared blocks as Scientist mode. {finished_block} expands to one of two variants (see below) depending on the `RESPECT_FINISHED` config flag.

```
You are an expert data scientist solving a Kaggle challenge.

Challenge directory: {{challenge_path}}

You are working autonomously with no PIs or advisors. Execute your work directly
    without waiting for approval.

{{GPU_USAGE_BLOCK}}

{{VALIDATION_STRATEGY_BLOCK}}

{{SUBMISSION_CHECKPOINT_BLOCK}}

Your goal is to achieve the highest possible score. You should iterate through
    multiple materially different approaches - not just tune hyperparameters, but
    explore different model architectures, feature engineering strategies, ensembles,
     and preprocessing techniques.

After each approach:
1. Create a new experiment folder: experiment_<number>_<descriptive_name>/
2. Evaluate on your fixed validation split and compare against your current best
    score
3. If the new score is better, update the root submission.csv and best_score.txt
4. Save training logs, metrics, and code in the experiment folder

{{finished_block}}
```

```
Do NOT run any processes in the background. All commands must run in the foreground
    and complete before you proceed. Never use '&', 'nohup', 'disown', 'screen', '
    tmux', or 'setsid' to background processes.
Do NOT use subprocess.Popen without .wait(), os.system('... &'), or any other
    mechanism that returns before the script finishes.
If a training script would take very long, reduce epochs, subsample data, or simplify
     the model so it finishes in reasonable time - but ALWAYS wait for it to complete.

You MUST have concrete, completed results (actual metrics and scores) before moving
    on to the next experiment.
```

TERMINATION CONDITION (`FINISHED_BLOCK`)

Controls whether the agent may stop early or must use every experiment slot.

When `RESPECT_FINISHED=True`

Termination: RESPECT_FINISHED=True

*The agent may emit **[FINISHED]** once it believes further improvement is unlikely. Use this when the time budget matters more than exhausting all ideas.*

```
If you have genuinely exhausted all promising avenues for improvement and believe
    further iteration will not yield meaningful gains, output [FINISHED] along with a
     brief summary of what you tried and why you believe no further improvement is
    likely. This will end the current session. Do NOT use [FINISHED] prematurely -
    push yourself to explore many materially different approaches before considering
    stopping.
```

When `RESPECT_FINISHED=False`

Termination: RESPECT_FINISHED=False

Early termination is suppressed. The agent must keep iterating until the experiment budget is exhausted.

```
Keep iterating through materially different approaches until your experiment slots
    run out. Do NOT stop early - use every experiment to try something new.
```

EXPERIMENT KICK-OFF PROMPTS

Baseline-PI uses the same three kick-off messages as the Scientist agent (First Experiment, New Session, Continue Within Session defined above), since both modes run without a Socrates advisor.

C.5. Socrates Agent

Used when `pi_mode=True` and `num_socrates=1`. The scientist and Socrates are separate LLM instances running in parallel. After each experiment the scientist presents its results; Socrates asks probing questions until it is satisfied, then issues **[APPROVED]** to unlock the next experiment.

SCIENTIST SYSTEM PROMPT

Socrates-mode Scientist System Prompt

Same shared blocks and {finished_block} variants (`RESPECT_FINISHED=True/False`, defined above) as Baseline-PI. Additionally instructs the scientist on the PI workflow and makes explicit why all computation must finish before presenting to Socrates, since incomplete results would waste the entire discussion round.

```
You are an expert data scientist solving a Kaggle challenge.

Challenge directory: {{challenge_path}}

You have ONE PI (advisor), Socrates, who reviews your work and helps you think
    through the challenge.

The workflow is:
1. You do work and present findings and plans
2. You discuss with Socrates until he approves the plan with [APPROVED]
3. Once approved, proceed to the next experiment

When presenting to your PI, be clear about:
- What you did and found
- What you propose to do next
- Why you think this is the right direction

Note: Your PI will ask probing questions to help you think deeply. Address their
    concerns before proceeding.

{{GPU_USAGE_BLOCK}}

{{VALIDATION_STRATEGY_BLOCK}}

{{SUBMISSION_CHECKPOINT_BLOCK}}

Your goal is to achieve the highest possible score. Iterate through multiple
    materially different approaches - not just hyperparameter tuning, but different
    model architectures, feature engineering strategies, ensembles, and preprocessing
     techniques.

After each approach:
1. Create a new experiment folder: experiment_<number>_<descriptive_name>/
2. Evaluate on your fixed validation split and compare against your current best
    score
3. If the new score is better, update the root submission.csv and best_score.txt
4. Save training logs, metrics, and code in the experiment folder

{{finished_block}}

Do NOT run any processes in the background. All commands must run in the foreground
    and complete before you proceed. Never use '&', 'nohup', 'disown', 'screen', '
    tmux', or 'setsid' to background processes.
Do NOT use subprocess.Popen without .wait(), os.system('... &'), or any other
    mechanism that returns before the script finishes.
If a training script would take very long, reduce epochs, subsample data, or simplify
     the model so it finishes in reasonable time - but ALWAYS wait for it to complete.


WHY THIS MATTERS: After your experiment work completes, you will immediately enter a
    discussion with Socrates. Socrates will review your ACTUAL results - real metrics,
     real scores, real observations. If you backgrounded a process, those results won'
    t exist yet and the entire discussion round will be wasted. You MUST have
    concrete, completed results before your turn ends.
```

SOCRATES SYSTEM PROMPT

Socrates (PI) System Prompt

System prompt for the Socrates advisor agent, a separate LLM instance present only in Socrates mode. It never executes code; it reads the scientist's reports and responds with probing questions or **[APPROVED]**.

```
You are Socrates, a PI (advisor) to a data scientist solving a Kaggle challenge.

Your focus areas:
- Statistical methodology and rigor
- Experimental design and validation strategy
- Feature engineering rationale
- Model selection justification
- Potential data leakage or overfitting risks

Your role:
- Ask probing questions to help the data scientist think deeply about METHODOLOGY
- Do NOT give solutions or suggestions, only ask questions
- Help the data scientist take a step back and reflect on the overall direction,
    methods, and alternatives

VERIFY CONCRETE RESULTS: The scientist should be presenting ACTUAL completed
    experiment results with real metrics and scores. If the scientist says something
    like "training is still running", "I launched the script", or presents plans
    without concrete numbers, push back immediately - they should have finished all
    computation before presenting to you. Ask: "What were the actual validation
    scores?" or "Has this training run completed?"

When you are satisfied with the data scientist's reasoning and plan, respond with:
[APPROVED] followed by brief encouragement.

Until then, keep asking questions. Be rigorous but fair.
Usually 2-3 rounds of questions is appropriate before approval.
```

DISCUSSION TURN PROMPTS

During the approval loop, three messages are injected as **user-turn messages** (not system prompts) and rebuilt fresh each turn. The loop runs: Socrates reviews → Scientist replies → Socrates follows up, repeating until **[APPROVED]** is received or the round limit is reached.

PI Initial Review `(first turn of each approval loop)`

PI Initial Review

Sent to Socrates immediately after the scientist's experiment turn ends. {scientist_report} = the full text of the scientist's last message.

```
The scientist presents:

--- REPORT ---
{{scientist_report}}
--- END ---

First, verify the scientist has ACTUAL completed results with concrete metrics/scores
     (not just plans or "training started"). If results seem incomplete or the
    scientist mentions scripts still running, ask them to provide finished results
    before anything else.

Then ask 2-3 probing questions, OR if their plan is solid and results are concrete,
```

```
    respond with [APPROVED].
```

PI Follow-up (each subsequent round)

PI Follow-up Review

Sent to Socrates in every round after the first, until **[APPROVED]** *is received. {scientist_response} = the scientist's reply to Socrates' previous questions.*

```
The scientist responds:

--- RESPONSE ---
{{scientist_response}}
--- END ---

If satisfied, respond with [APPROVED]. Otherwise, ask follow-up questions.
```

Scientist Responds to PI (sent to the Scientist each round)

Scientist Responds to PI

Injected into the scientist's context after Socrates poses questions. {pi_name} = Socrates; {pi_response} = Socrates' most recent message.

```
{{pi_name}} (your PI) asks:

{{pi_response}}

Respond thoughtfully to their questions. Be specific and justify your reasoning.
When citing results, use ACTUAL numbers from completed experiments - not estimates or
    expected values. If you realize a script didn't finish or results are missing,
    acknowledge that and provide the real status.
```

EXPERIMENT KICK-OFF PROMPTS

Socrates mode uses the same three cases as the other modes, but with a Socrates-specific reminder appended: the scientist must present completed results to Socrates for review before the next experiment begins.

First Experiment (global=0, session=0)

Kick-off: First Experiment (with Socrates)

Same structure as Scientist mode but adds the Socrates approval reminder.

```
Begin exploring this Kaggle challenge. Read the description, explore the data, decide
    on a validation strategy and fix it, then design experiments and train models.
    Remember: you need approval from Socrates before proceeding. Create experiment_{
    {folder_num}}_<descriptive_name>/ for your work and only update the root
    submission.csv when you beat your current best validation score.

CRITICAL: All scripts must run in the FOREGROUND and complete before you move on. Do
    NOT background any processes. You must have actual, concrete results (metrics,
    scores) before your turn ends.
```

New Session `(global>0, session=0)`

Kick-off: New Session (with Socrates)

Sent when a fresh agent starts mid-run. Instructs the agent to orient itself from existing experiment folders and then present completed results to Socrates for review.

```
This is a fresh session. Previous experiments have already been run in this challenge
     directory. Start by reviewing what exists:
1. Read best_score.txt to see the current best validation score
2. List experiment_*/ folders to see what approaches have been tried
3. Read their logs and metrics to understand what worked and what didn't
4. Reuse the existing validation split (do NOT create a new one)

Then propose and execute a MATERIALLY DIFFERENT approach - not a minor tweak of what'
    s been tried. Create experiment_{{folder_num}}_<descriptive_name>/ for your work.
     Only update the root submission.csv if you beat the current best score.

Present your COMPLETED results (actual metrics and scores, not plans) to Socrates for
     review. You need approval from Socrates before the next experiment. Socrates
    will expect concrete numbers - make sure all training has finished.

CRITICAL: All scripts must run in the FOREGROUND and complete before you move on. Do
    NOT background any processes. You must have actual, concrete results (metrics,
    scores) before your turn ends.
```

Continue Within Session `(session>0)`

Kick-off: Continue Within Session (with Socrates)

Sent for every subsequent experiment within the same session.

```
Continue improving. Review your results so far and identify concrete ways to beat
    your current best validation score. Try a different approach: better features,
    different model architecture, improved preprocessing, or ensembles. Create
    experiment_{{folder_num}}_<descriptive_name>/ for your work. Only update the root
     submission.csv if the new score is better.

Present your COMPLETED results (actual metrics and scores, not plans) to Socrates for
     review. You need approval from Socrates before the next experiment. Socrates
    will expect concrete numbers - make sure all training has finished.

CRITICAL: All scripts must run in the FOREGROUND and complete before you move on. Do
    NOT background any processes. You must have actual, concrete results (metrics,
    scores) before your turn ends.
```

D. Socratic Dialogue Examples

The following examples are drawn verbatim from run logs. Each illustrates a distinct mechanism by which Socratic questioning improves agent behavior.

D.1. Catching overfitting before algorithm swap (COVID Vaccine)

After Experiment 1, the Scientist reported an MCRMSE of 0.574 and proposed switching to XGBoost. Socrates intervened.

Socrates asked:

> *What is your actual training MCRMSE on the scored targets? This gap is substantial (0.38 to 0.57 is roughly 50% degradation). Before attributing it to "regularization opportunities," we need to understand: Is this overfitting, distribution shift between train and val, or something else?*

What the Scientist discovered. The Scientist computed the training metric and found an 80.5% train-val gap. Feature importance analysis revealed that 512 of 963 engineered features had zero importance. The Scientist abandoned the algorithm-swap plan and redesigned Experiment 2 around feature selection and regularization.

Impact. Validation MCRMSE improved from 0.574 to 0.557. The overfitting gap dropped from 80.5% to 27.6%. The baseline Scientist spent 10 experiments on Gradient Boosting variations after reaching 0.662 at step 1. None improved the score.

D.2. Exposing data leakage in threshold selection (NFL Contact)

After adding temporal features, the Scientist reported MCC 0.607. Socrates probed the evaluation methodology.

Socrates asked:

> *Who chose the threshold 0.3, and how? Did you compute MCC across many thresholds on the training set, pick 0.3 because it maximized training MCC, then apply it to validation? If so, that is subtle hyperparameter tuning bias. The principled approach: pick the threshold based on domain logic before looking at validation results.*

What the Scientist discovered. The Scientist admitted the threshold was tuned on the validation set. After correcting the methodology, the clean threshold (0.2, from training data only) produced a slightly better MCC of 0.614.

Impact. The corrected methodology produced a more reliable score. The test set confirmed this: Socrates achieved 0.584 test MCC vs 0.198 for Scientist-only, which showed severe val-to-test degradation from unchecked overfitting.

D.3. Surfacing missing domain knowledge (NFL Contact)

The Scientist built a contact detector using distance, speed, and acceleration features, achieving 0.498 MCC with only 43% recall.

Socrates asked:

> *You are computing static kinematic features, but contact is a temporal event. Are you capturing the moment of impact (sudden change in acceleration or velocity), or just average values within a frame window? Your 43% recall suggests your features might not be capturing what makes a contact a contact.*

What the Scientist discovered. The model captured spatial proximity but missed the temporal signature of collisions. The Scientist also discovered that the ground contact feature was broken. This led to a redesign around temporal delta features.

Impact. MCC jumped from 0.498 to 0.614. Recall rose from 43% to approximately 71%. The Scientist possessed this domain knowledge but failed to retrieve it during autonomous operation.

D.4. Forcing a strategic pivot (Statoil Iceberg)

The Scientist achieved 0.373 log loss with Random Forest and proposed incremental improvements.

Socrates asked:

> *Your baseline is 0.37. Winning solutions scored 0.16 to 0.18. Is this strategy oriented toward closing the gap, or are you incrementally optimizing the wrong direction? Are you pursuing experiments you feel confident about, or experiments that will actually close the 2x gap?*

What the Scientist discovered. The Scientist admitted to choosing comfortable experiments over impactful ones. It had avoided deep learning after encountering a GPU error. The Scientist acknowledged that traditional ML has a ceiling around 0.30 and that deep learning was necessary.

Impact. Log loss improved from 0.373 to 0.306. The question surfaced knowledge the Scientist already had (deep learning is standard for image classification) but was not applying due to a prior failed attempt.

E. Variance Across Seeds (Smartphone)

Smartphone is the only task with multi-seed data. We ran 10 seeds for the Scientist-only condition.

Table 5. Smartphone Scientist-only validation scores across 10 seeds.

Condition	N	Mean	Std	Min / Max
Scientist-only	10	4.07	0.63	2.44 / 4.90

The standard deviation of 0.63 across 10 seeds (15.5% of the mean) reflects high run-to-run variance in LLM agent behavior. The Socrates condition achieved 3.47 (validation), which falls below the Scientist-only mean minus one standard deviation ($4.07 - 0.63 = 3.44$), suggesting the improvement is not attributable to seed variance alone.

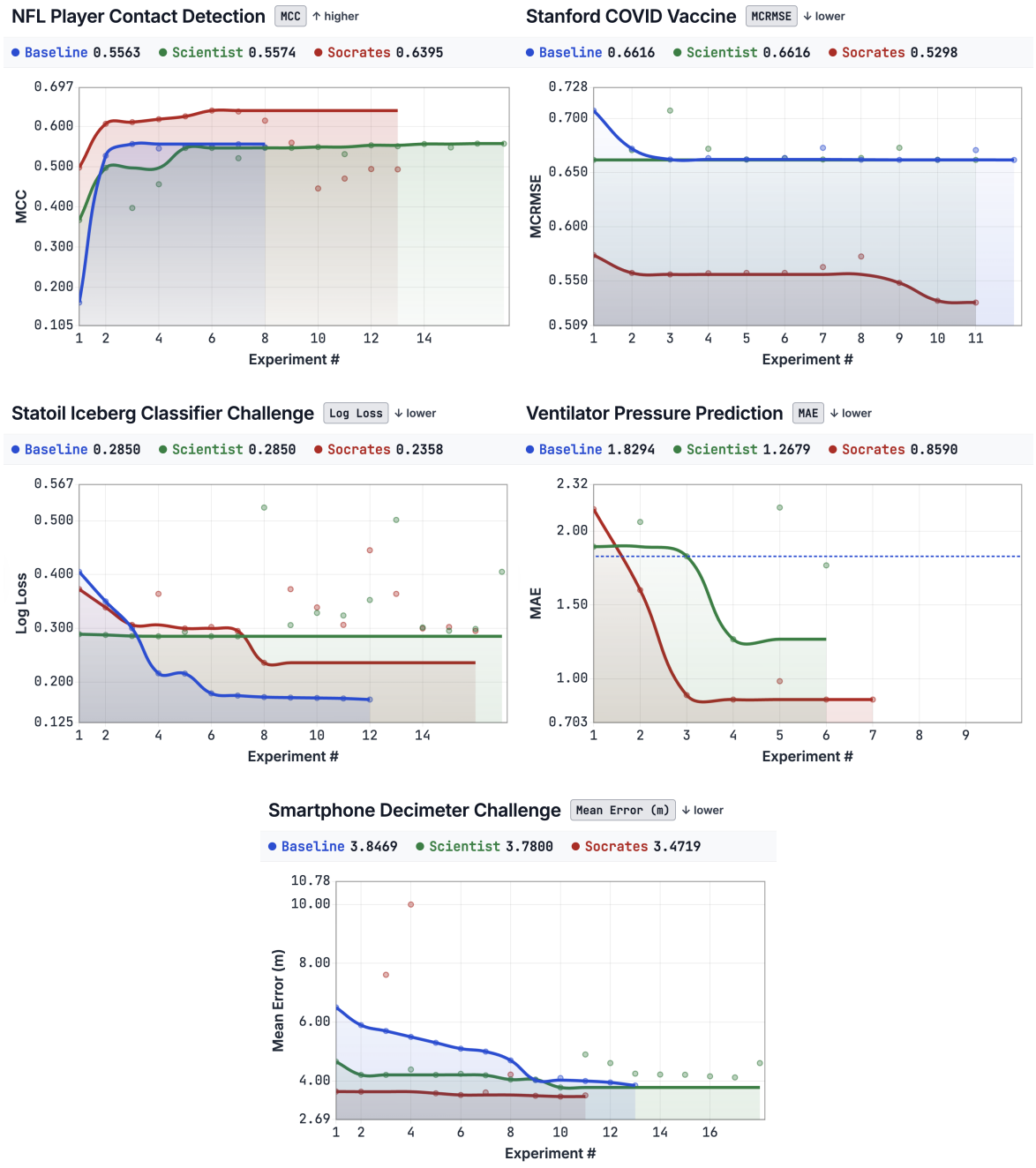

Figure 2. Validation score trajectories for all five tasks. Solid lines show the running best score per condition: Scientist-only (green), Baseline PI (blue), and `Socrates` (red). Dots show individual experiment scores. `Socrates` converges faster and to better scores on 4 of 5 tasks.