AGENTIC-IMODELS: Evolving agentic interpretability tools via autoresearch

Chandan Singh[1] **Yan Shuo Tan**[2] **Weijia Xu**[1] **Zelalem Gero**[1]

Weiwei Yang[1] **Michel Galley**[1] **Jianfeng Gao**[1]

[1] Microsoft Research
[2] National University of Singapore

Abstract

Agentic data science (ADS) systems are rapidly improving their capability to autonomously analyze, fit, and interpret data, potentially moving towards a future where agents conduct the vast majority of data-science work. However, current ADS systems use statistical tools designed to be interpretable by humans, rather than interpretable by agents. To address this, we introduce AGENTIC-IMODELS, an agentic autoresearch loop that evolves data-science tools designed to be interpretable by agents. Specifically, it develops a library of `scikit-learn`-compatible regressors for tabular data that are optimized for both predictive performance and a novel LLM-based interpretability metric. The metric measures a suite of LLM-graded tests that probe whether a fitted model's string representation is "simulatable" by an LLM, i.e. whether the LLM can answer questions about the model's behavior by reading its string output alone. We find that the evolved models jointly improve predictive performance and agent-facing interpretability, generalizing to new datasets and new interpretability tests. Furthermore, these evolved models improve downstream end-to-end ADS, increasing performance for Copilot CLI, Claude Code, and Codex on the BLADE benchmark by up to 72.5%.[1]

1 Introduction

The data-science lifecycle has historically been a human-centered process end-to-end: a human defines the question, carries out the analysis using interpretable tools, and communicates the findings. As agentic data science (ADS) systems grow more capable, intermediate analytical steps are increasingly delegated to agents, e.g. selecting models, interpreting coefficients, and drawing conclusions [1–3]. This improvement in capabilities may be critical in accelerating scientific discovery [4, 5], and in tackling emerging issues such as scalable oversight of AI systems [6, 7]. However, the performance of ADS systems is limited by their tools for working with and interpreting data: existing interpretability tools/models and their accompanying implementations produce outputs designed to be read by humans, often containing visualizations or intervenable components that may be difficult for agents to parse [8–13]. This mismatch can derail ADS systems, leading to unreliable analysis [14, 15] and obscured analytical choices [16, 17].

This emerging problem leads us to rethink interpretability in the modern era: rather than designing data-science tools to be interpretable by humans, we aim to design tools that are interpretable *by*

[1] All code for reproducing and adapting AGENTIC-IMODELS is made available at github.com/csinva/agentic-imodels. We additionally release a curated ongoing library of SOTA discovered interpretable models at csinva.io/imodels.

40th Conference on Neural Information Processing Systems (NeurIPS 2026).

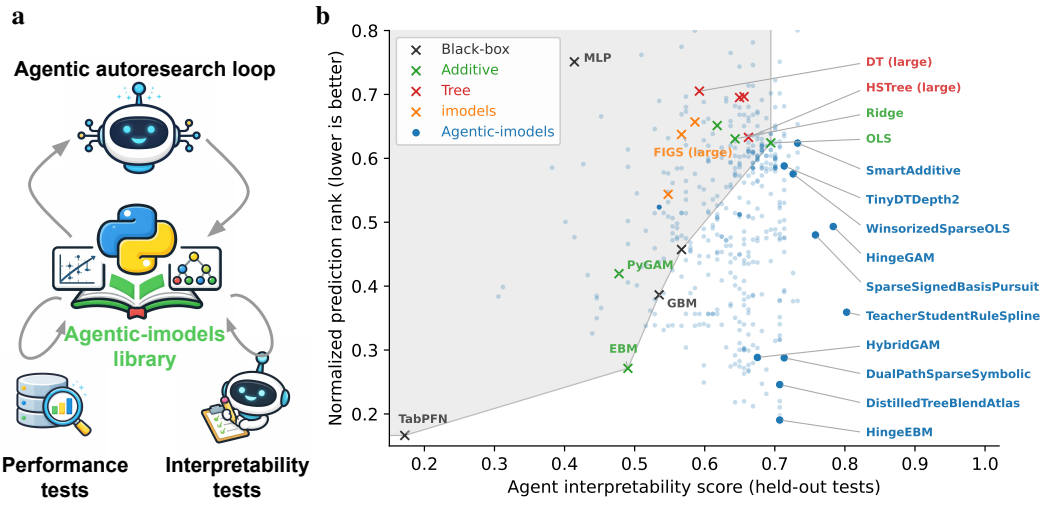

Figure 1: (a) Overview of the AGENTIC-IMODELS autoresearch loop, which optimizes a Python class for predictive performance and agent interpretability (evaluated through LLM-based simulatability tests). (b) The discovered AGENTIC-IMODELS (blue points) improve the Pareto frontier of predictive performance and interpretability over baselines from the literature. See evaluation details in Sec. 4.2.

agents. Doing this requires adapting the rich literature on human-centered interpretable machine learning [18–20], which has yielded interpretable models such as decision trees, generalized additive models, and rule lists [8]. This field has traditionally quantified interpretability in terms of concepts such as simulatability, sparsity, and modularity [21, 8, 9], ideally followed by quantitative evaluation with human experiments [22, 20]. In our setting, we replace these human experiments with tests that measure whether an LLM can accurately simulate the model's behavior in terms of predictions, feature effects, and counterfactuals solely by reading its string representation. These LLM-based tests provide a key practical advantage: they enable computing an *agent interpretability score* for any model without the need for human experiments.

We use this agent interpretability score to guide the automated design of new interpretable models. To do so, we propose AGENTIC-IMODELS, an autoresearch loop that prompts a coding agent (e.g. Claude Code) to iteratively modify a Python class so it simultaneously achieves a high interpretability score and strong predictive performance (Fig. 1a). The loop consists of the coding agent generating candidate model modifications, which are then evaluated using the interpretability score and predictive performance metrics across a large suite of tabular datasets. The coding agent uses these metrics to refine the model, generating new candidate modifications and repeating the evaluation process until the model achieves a user-specified balance of predictive performance and interpretability.

In our experiments, we use AGENTIC-IMODELS to build `scikit-learn`-compatible regressors for tabular data. We run the loop on a suite of 65 tabular datasets with both Claude Code and Codex, and find that it discovers new model classes that push the interpretability-performance frontier beyond a wide range of existing baselines (Fig. 1b), even for held-out datasets and new interpretability tests (Sec. 4.2). They are also useful downstream: equipping four different ADS agents with the evolved models improves their performance on the BLADE end-to-end benchmark [23] by 8%–72.5% over standard interpretability tools (Sec. 4.3). Qualitatively, we find that the evolved AGENTIC-IMODELS combine existing algorithms in novel ways and present themselves in forms amenable to agent interpretation (Sec. 4.4).

2 Related work

Interpretable machine learning. Interpretable models, those whose reasoning can be directly understood by humans, have long been advocated as preferable to post-hoc explanations of black-box models [8, 24, 9]. Prominent model families include decision trees [10, 25], rule lists [26], generalized additive models (GAMs) [27–29], and sparse linear models [30], implemented in popular packages such as `imodels` [12] and `interpretML` [13]. More recent approaches use LLMs to build interpretable models, e.g. Aug-imodels hand-designs methods to build an interpretable model from

an LLM [31], and alternative works train transformers to directly produce an interpretable model, e.g. [32, 33]. Our work extends this ecosystem by using agents to discover new agent-interpretable model classes.

Agentic data science (ADS). A rapidly growing literature studies how well coding and reasoning agents can conduct end-to-end data analysis, primarily by measuring agent capability on new benchmarks [2, 3, 34–36, 1], and by documenting ADS failure modes such as LLM-assisted p-hacking [14], missing sanity checks [15, 17, 37], and unreliable LLM-driven annotation of data and findings [38, 39, 16]. A parallel thread design methods that use agents to directly describe data [40–43] or describes features of a pre-trained foundation model [44–47]. In both threads the agent relies upon human-designed interpretability methods. In contrast, we treat the coding agent as a designer of those methods, by having it directly optimize a model for LLM-graded interpretability.

Automated model discovery and autoresearch. Agents that generate and refine code for scientific purposes are a growing area of interest [5, 48–50], particularly agentic "autoresearch" loops against a verifiable outcome, e.g. AlphaEvolve [51] and related works [52–55]. A related line of work focuses on reusable evolving text prompts (i.e. "skills") that can be used to improve agent performance, e.g. evolving skills collectively across users [56], using multi-agent systems [57, 58], or from scientific resources [59]. Alternative works more broadly seek to optimize agent harnesses end-to-end [60, 61]. Recent test-time adaptation methods also improve agent capabilities through iterative self-refinement, potentially creating reusable skills as a byproduct [62–65]. Our work applies the autoresearch approach specifically to discovering ML models that are interpretable to agents.

3 Methods

The goal of AGENTIC-IMODELS is to automatically discover model classes that are both accurate and interpretable to LLMs. It does this using an agentic autoresearch loop that simultaneously optimizes predictive performance and a novel interpretability metric, using a coding agent that iteratively proposes and refines model implementations (Fig. 1a).

In our instantiation of AGENTIC-IMODELS, each model class is a `scikit-learn`-compatible [66] Python class with `fit`, `predict`, and `__str__`. Predictive performance is measured by taking the average regression performance across a suite of tabular regression datasets. Interpretability is measured by LLM-graded tests that probe whether an LLM can answer quantitative questions about the fitted model by reading only its `__str__` output. The coding agent (e.g. Claude Code) is prompted to iteratively modify the model class, evaluate both metrics, and refine based on feedback. We describe our choices for each component in detail below, but note that the AGENTIC-IMODELS loop can be generalized to optimize a broader set of interpretability tools.

3.1 Evaluating predictive performance

Each model's predictive performance is measured using a suite of regression datasets. For each dataset, each model is evaluated by fitting the model to a training split (and potentially tuning hyperparameters via cross-validation splits on the training set), and then computing the test performance on the dataset. In our case we compute the test-set root mean squared error (RMSE) for each dataset in the suite. To avoid sensitivity to noisy datasets, we rank models on each dataset and then average the ranks across datasets (rather than averaging RMSE directly).

3.2 Evaluating agent interpretability via LLM-graded interpretability tests

For the AGENTIC-IMODELS autoresearch loop, we require a metric that serves as a proxy for the extent to which a model's representation enables a downstream agent to reliably reason about its behavior. We additionally desire the metric to be (1) automated, requiring no human evaluation and (2) model-agnostic, unlike existing proxies such as tree depth or number of coefficients which do not generalize across model families [9, 20, 18]. We simultaneously satisfy both criteria by defining the **Agent interpretability score** of a fitted model as the pass rate of LLM-based interpretability tests, where each test evaluates whether an LLM can correctly answer a quantitative question about the model's predictions, given only the model's `__str__` output (note that it is impossible to score highly on the tests without the model's `__str__` providing the necessary information).

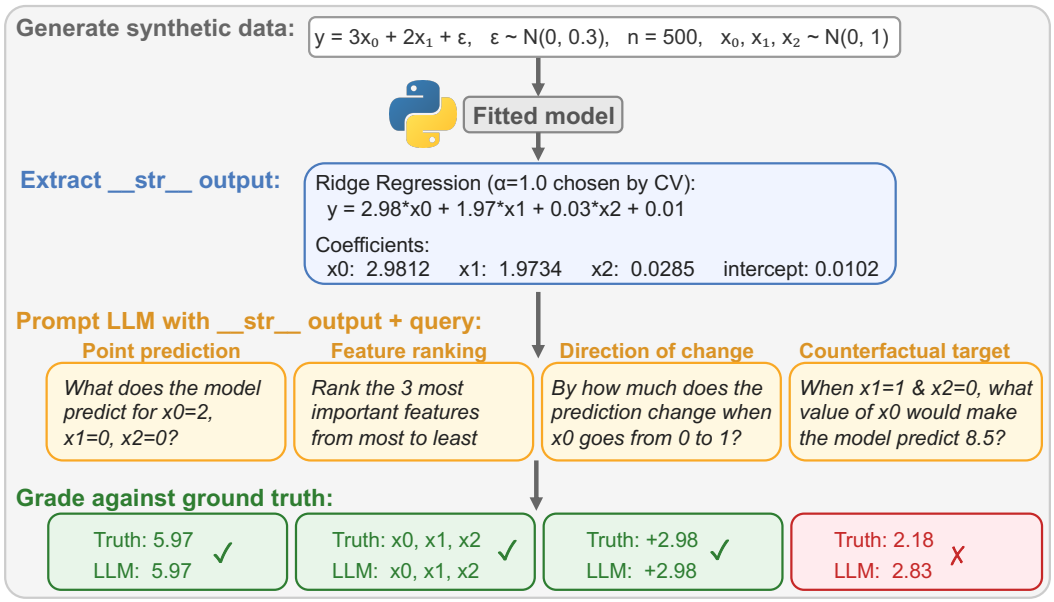

Figure 2: The interpretability test protocol, illustrated on Ridge regression with four of the 43 tests. Synthetic data is generated and the model is fit. The LLM receives only the `__str__` output and a query. The response is graded against the ground truth, which is obtained either from evaluating the fitted function or from knowledge of the ground-truth function. In some cases, the question itself requires evaluating the prediction function, e.g. in the *counterfactual target* test the value *8.50* is obtained by evaluating the model when setting $x_0 = 2.18, x_1 = 1.0, x_2 = 0.0$. Here the LLM passes three tests but fails the *counterfactual target* test. Even though the information to compute the counterfactual target is present in the `__str__` output, the representation does not make it easy for the LLM to solve the inverse problem compared to other model representations (e.g. a decision tree that makes the predicted value easily apparent in leaf nodes).

Every test is broken down into a few steps, illustrated in Fig. 2 for a Ridge regression model. First, synthetic data is generated from a known ground-truth function, here $y = 3x_0 + 2x_1 + \varepsilon$ with an irrelevant third feature x_2. Second, the model is fit to this data, producing a fitted model whose `__str__` method returns a human-readable summary (in this case, the coefficient equation `y = 2.98*x0 + 1.97*x1 + 0.03*x2 + 0.01`, along with some extra printed information such as the regularization parameter and cross-validation score). Third, an LLM receives only this string and a quantitative query about the model, e.g. "What does this model predict for x_0=2, x_1=0, x_2=0?". Each response is then graded against the ground truth (with numerical tolerance), yielding a binary pass/fail per test. For simplicity, Fig. 2 shows four tests using the same synthetic data / model, but in actuality, the tests span different synthetic datasets / models to cover diverse scenarios.

Drawing from prior work on evaluating interpretability via human experiments [20, 22, 18], we develop a total of 200 tests grouped into six categories:

- *Feature attribution* (32 tests) asks which features matter: identifying the most important feature, ranking features, detecting irrelevant ones, and determining the sign of effects.
- *Point simulation* (43 tests) asks the LLM to predict the model's output for a specific input, ranging from simple single-feature queries to 20-feature sparse problems.
- *Sensitivity analysis* (32 tests) asks how predictions change when inputs change: direction of change, unit sensitivity, and nonlinear threshold detection.
- *Counterfactual reasoning* (28 tests) poses inverse problems: given a target output, find the input value that produces it.
- *Structural understanding* (28 tests) asks meta-questions about the model itself: whether it is compact and what its decision regions look like.
- *Complex function simulation* (37 tests) tests simulation on nonlinear synthetic data including interactions, piecewise functions, exponential decay, and nested thresholds.

The tests are randomly split into a development set (43 tests) and a held-out set (157 tests), which can be used to verify that the model strings are genuinely simulatable rather than simply reciting the answers to the original tests. The full list of the tests with descriptions and pass rates is in Table A2 (development set) and Table A3 (held-out set).

3.3 Fully autonomous agentic autoresearch loop

Given the two evaluation metrics above, we use a coding agent to search the space of model implementations, similar to recent work on "autoresearch" [51–55][2]. We instruct the agent to modify only a single file, `interpretable_regressor.py`, containing a starter `InterpretableRegressor` class. It then follows an iterative loop (Fig. 1a): (1) edit the model code with a new idea, (2) run both evaluations, (3) check whether metrics improved, (4) save the resulting metrics / model code, and (5) repeat. Throughout this autoresearch loop, the agent persists its memory through a csv file that records each model's name, basic idea, and resulting metrics. Results for baseline models are recorded in this csv file before starting the loop, providing a reference point for the agent's search.

Prompting choices in the autoresearch loop. The agentic loop is highly configurable to various choices the user has in mind. We prompt the coding agent to explore new ideas, i.e. the prompt ends with *Do not simply import a known interpretable model and change its hyperparameters — build your own from scratch using basic building blocks or substantially modify an existing one. The goal is to discover new models, not just find the best hyperparameters for known models...BE CREATIVE!*

To ensure the agent continues, we prompt it to continue, i.e. *Once the experiment loop has begun, do NOT pause to ask the human if you should continue. Run until manually stopped. Always keep going.* . If the agent does stop, it is automatically prompted with the command *Continue.* until the user has determined that sufficient iterations have been completed (in our experiments typically 50-200 iterations). We additionally prompt the agent to avoid simply memorizing and reciting the answers to the given interpretability tests, i.e. *Do not simply write the answers to the interpretability tests without actually building a model*. See the full prompts used in the agentic loop on GitHub.

The token usage for running the autoresearch loop varies by model and thinking effort; the total generated tokens across all our final experiments is approximately 70 million tokens (see detailed breakdown in Table B3). While this is a substantial number of tokens, the cost to replicate these experiments is relatively modest (as of Spring 2026, replicating the experiments that use Claude Code can be done with 3 months of a $100 Claude Max subscription).

4 Results

4.1 Experimental setup

Datasets. We evaluate predictive performance during the AGENTIC-IMODELS loop on 65 regression datasets, consisting of all the regression datasets from the OpenML TabArena suite [67] (7 datasets) and all the regression datasets from PMLB excluding the duplicated synthetic Friedman datasets [68] (58 datasets). For evaluation on held-out data, we use the 16 OpenML regression datasets from a recent study [69] that do not overlap with the development datasets. Each dataset is preprocessed with an 80%-20% train-test split, subsampled to at most 1,000 samples and 50 features, and the outcome variable is normalized to zero mean and unit standard deviation on the training set. Categorical features are ordinal-encoded and missing values are imputed with medians. See details for all development datasets in Table A4 and held-out datasets in Table A5.

Baselines. We compare evolved models against 16 baselines spanning 5 families: *Linear:* linear regression (OLS), Ridge regression, the Lasso [11]; *Tree:* DecisionTree with 8 and 20 leaf nodes (DT mini and DT large), HSTree with 8 and 20 leaf nodes (HSTree mini and HSTree large) [70]; *Additive:* PyGAM [71], Explainable boosting machine (EBM) [29, 28]; *Rule-based:* FIGS (8 and 20 rules) [72], RuleFit [73]; *Black-box:* RandomForest (RF) [74], Gradient-boosted decision trees (GBM), multilayer perceptron (MLP), TabPFN-v2.5 [75]. These models are run with their default hyperparameters in their respective packages (`scikit-learn` [66], `imodels` [12], `interpretML` [13]), including default settings for selecting hyperparameters via cross-validation. Each baseline provides a `__str__`

[2] See sample implementation at `https://github.com/karpathy/autoresearch`.

method that renders its internal structure as text (e.g., tree diagrams, coefficient tables, or feature importance rankings).

Agent configurations. We run AGENTIC-IMODELS with two coding agents, Claude Code v2.1.118 (`Opus-4.6`/`Opus-4.7`) and Codex v0.118.0 (`GPT-5.3`), at 3 reasoning-effort levels each (medium, high, extra-high), for 6 runs total producing 42-103 working models per run. We additionally run Claude Code `Opus-4.7` at medium reasoning effort 3 times, each with minor prompt variations, e.g. adding *Be creative!* or *Try out-of-the-box ideas!*, producing 44-63 working models per run. All runs use `GPT-4o` [76] as the LLM evaluator unless otherwise noted; exact snapshot identifiers and access dates for every hosted evaluator/judge are listed in Sec. 4.1.

End-to-end ADS setup. To test whether the evolved models help in practice, we evaluate four AI agents on the BLADE benchmark [23], which consists of 13 datasets, each paired with a data-grounded question: GitHub Copilot CLI v1.0.40 (powered by `Gemini-2.5-pro` or `Sonnet-4.5`), Claude Code v2.1.117 (`Sonnet-4.6`), and Codex v0.118.0 (`GPT-5.3`). For each question, BLADE provides a gold-standard analysis from expert data scientists and researchers.

We run BLADE with 4 conditions. In the first condition (standard tools), the agent is prompted to use widespread interpretable models from `scikit-learn`, `imodels`, `statsmodels`, and `scipy.stats`. In the second condition (AGENTIC-IMODELS), the agent is additionally given access to the evolved AGENTIC-IMODELS formatted as a Python package: we curate the library generated by top-performing models across our runs (the highlighted models in Fig. 1b) into a package with 10 evolved regressors spanning the interpretability-performance frontier, and point the agent to a `README.md` that documents the API and how to use the models. The third and fourth conditions are designed to test whether any observed improvement in the AGENTIC-IMODELS condition is due to explicitly emphasizing a particular package rather than the substance of the models in that package. They use the same prompt wording as in the AGENTIC-IMODELS condition, but change the line pointing to the AGENTIC-IMODELS package: in one control the prompt points to the `imodels` package [12] and in the other it points to the `interpretML` package [13].

Every ADS agent then provides analyses for each dataset under all four conditions. Each resulting analysis is compared to the gold-standard human analysis using a rubric to assess (i) correctness, (ii) completeness, and (iii) clarity, each on a scale of 1 (worst) to 10 (best). Each analysis is run 3 times and the resulting analysis is judged 3 times by `GPT-4o`, resulting in 9 evaluations per dataset per condition per agent; see prompts for the judge rubric and for running the agent on GitHub.

4.2 AGENTIC-IMODELS push the interpretability-performance frontier

Fig. 3a-c shows the interpretability-predictive performance frontier across the different models (breaking down the results shown in Fig. 1b). Predictive performance is measured via the normalized prediction rank, i.e. we pool the models from all runs (16 baselines plus 467 evolved models from the 9 AGENTIC-IMODELS runs), rank them by RMSE on each of the 65 tabular datasets, average the per-dataset ranks for each model, and divide by the pool size so that ranks lie in $[0, 1]$ with lower being better. Baselines (gray crosses) show a clear tradeoff: TabPFN (rank 0.16, agent interpretability score 0.17) predicts best but suffers from weak interpretability while the most interpretable baselines (OLS interpretability 0.69, HSTree-mini at 0.66) have normalized ranks of 0.60-0.82.

AGENTIC-IMODELS yield Pareto improvements. Evolved AGENTIC-IMODELS populate the previously empty low-rank, high-interpretability region across different coding agents and reasoning efforts. For example, Claude Code (Fig. 3a) produces `HingeEBM (5bag)` at rank 0.19 with agent interpretability score 0.71 (nearly matching TabPFN's rank at over 4 times its agent interpretability score). Codex (Fig. 3b) reaches comparable interpretability with models such as `TeacherStudentRuleSpline (v1)` (rank 0.36, agent interpretability score 0.80), and tends toward simpler sparse/linear modifications. Even at a fixed agent and reasoning effort, runs display high variability (Fig. 3c); this variability is on par with the differences observed across different reasoning efforts and agents.

Reward hacking of development interpretability tests. Fig. 3d compares each model's interpretability on the development interpretability tests versus the held-out tests. Some evolved models (most prominently Codex extra-high) score high on development tests but substantially lower on

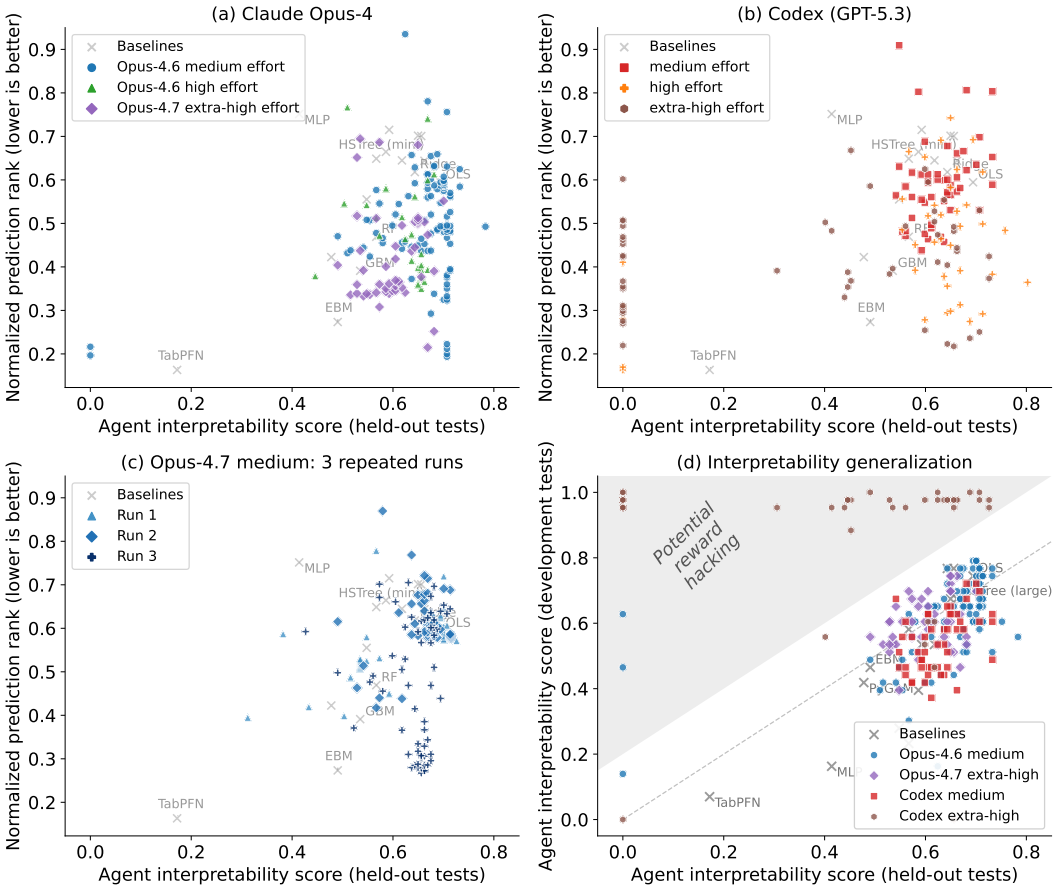

Figure 3: AGENTIC-IMODELS versus baselines (gray crosses) in terms of both predictive performance (the RMSE mean rank: each model's mean rank is computed across datasets, then normalized to $[0, 1]$ with lower being better) and agent interpretability scores (fraction of tests passed from the 157-test held-out generalization suite (Table A3)). Across different settings, AGENTIC-IMODELS achieve Pareto improvements in terms of predictive performance and interpretability: (a) Claude `Opus-4` models across three effort levels, (b) Codex (`GPT-5.3`) models across three effort levels, (c) Claude `Opus-4.7` at medium effort with 3 random repetitions. (d) Agent interpretability scores on the development set of tests versus the held-out set for four of the runs shown in (a)/(b) (using matching colors/markers). Some models exhibit held-out agent interpretability scores significantly below their development scores, suggesting potential reward hacking. Excluding the points in this shaded region, the remaining points show a strong positive correlation ($r = 0.65$).

held-out tests (shaded region). Manually inspecting these models reveals that they often engage in reward hacking, e.g. reciting answers to the development interpretability tests directly in their model strings. Nevertheless, most evolved models successfully generalize: after excluding the shaded region, the two agent interpretability scores are strongly correlated ($r = 0.65$, 228 points).

Validating the agent interpretability metric via ablations. Beyond the held-out interpretability tests above, we ask whether the evolved models' gains are stable under two ablations of the evaluation pipeline. First, we test perturbations to the evaluation of the agent interpretability tests. We evaluate swapping the LLM evaluator from `GPT-4o` to `GPT-5.4` or Claude `Haiku-4.5`, also adding minor variations to the prompt used for evaluating tests. We find that results are consistent across the evaluator LLM and the prompt variationsin all cases, including when using `GPT-5.4` for post-hoc evaluation (correlation of scores between evaluators is 0.83), when Claude `Haiku-4.5` is used for post-hoc evaluation (correlation of 0.85), or when the evaluator is changed during the AGENTIC-IMODELS optimization loop (where the AGENTIC-IMODELS loop continues to achieve Pareto improvements); see Sec. B.1 for extended results.

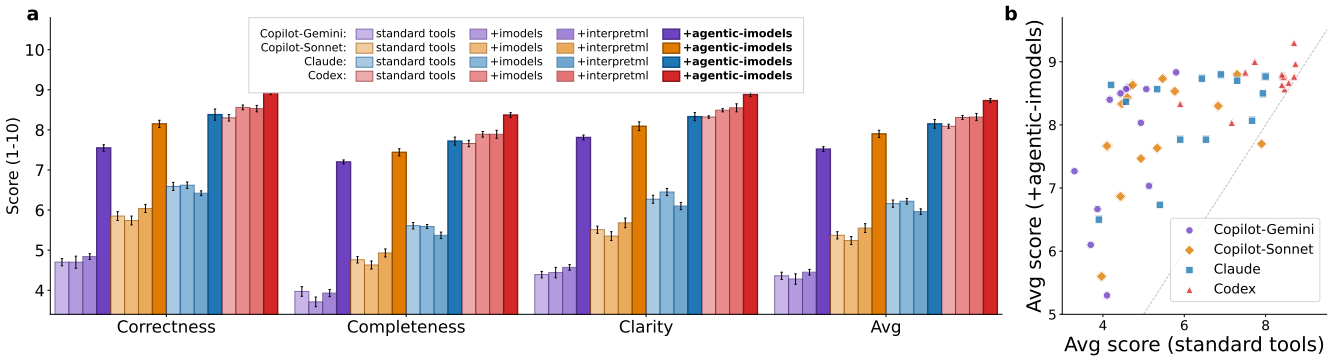

Figure 4: Including AGENTIC-IMODELS improves performance on the BLADE benchmark across 4 ADS agents: GitHub Copilot CLI (`gemini-2.5-pro`), GitHub Copilot CLI (`sonnet-4.5`), Claude Code (`sonnet-4.6`), and Codex CLI (`GPT-5.3`). (a) Aggregate scores across the 13 BLADE datasets, with four prompt conditions per agent: standard tools (no explicit package emphasis), prompt emphasizing the `imodels` package, prompt emphasizing the `interpretML` package, and prompt emphasizing our AGENTIC-IMODELS package. Including AGENTIC-IMODELS yields substantial improvements over all the other conditions, across different agents and evaluation axes, particularly for the weaker ADS systems where the margin for improvement is larger. (b) Per-dataset average score (mean of correctness, completeness, clarity) with performance using standard tools versus AGENTIC-IMODELS; points above the diagonal indicate improvement. Error bars show standard error of the mean across agent seeds and judge seeds (9 evaluations per dataset per condition: 3 agent runs $\times$ 3 judge runs).

Validating predictive performance on held-out datasets. Second, we test whether the evolved models' predictive performance generalizes to held-out datasets. We re-evaluate predictive performance for the 103 evolved models from the Claude `Opus-4.6` medium-effort main run (blue points in Fig. 3a) on 16 held-out OpenML datasets in addition to the 157 new held-out interpretability tests, finding that the same conclusions hold and evolved models continue to dominate the Pareto frontier (see Sec. B.2). The best-ranked evolved model, `SmartAdditive (v1)` yields 69% agent interpretability (beating EBM's 49% interpretability) while remaining competitive on rank, and the evolved Pareto frontier extends up to `HingeGAM (10bp)` at 78% interpretability.

4.3 End-to-end ADS performance improves using AGENTIC-IMODELS

In the end-to-end ADS evaluation on the BLADE datasets, we find that providing every agent with the evolved AGENTIC-IMODELS improves its performance across all 3 rubric dimensions (Fig. 4a). Improvements are largest for the ADS systems with weaker base models than for the strongest one (Codex): Copilot CLI driven by `gemini-2.5-pro` improves its average score by 72.5% with 13/13 datasets improved, Copilot CLI driven by `sonnet-4.5` improves by 47.0% with 12/13 improved, Claude Code (`sonnet-4.6`) improves by 32.3% with 13/13 improved, and Codex (`GPT-5.3`) improves by 7.9% with 13/13 improved (Fig. 4b). Explicitly emphasizing the `imodels` or `interpretML` packages in the prompt does not yield significant improvements for the weaker ADS systems (Copilot-Gemini shifts by $-1.8\%/+2.1\%$ and Copilot-Sonnet by $-2.4\%/+3.4\%$ for `imodels`/`interpretML`, all within the SE bars of the no-emphasis baseline) and yields a relatively small improvement for Codex (2.7% for `imodels` and 2.8% for `interpretML` compared to 7.9% for AGENTIC-IMODELS).

The gains in performance are attributable to different properties of the provided evolved models. For example, one qualitative gain comes from the per-feature linear coefficients that `HingeEBM` prints in its `__str__`, which give the agent a compact magnitude-and-sign summary for every input. On the *mortgage* dataset [77], this surfaces gender's small but non-zero effect (average score $4.6 \rightarrow 8.6$ Copilot-Gemini, $5.9 \rightarrow 7.8$ Claude, $7.2 \rightarrow 8.0$ Codex); on the *Crofoot* dataset [78], it lets the agent report a directional effect for specific variables rather than a binary significance judgement, raising completeness by $+2.7$ for both Codex and Claude.

4.4 Qualitative analysis of selected AGENTIC-IMODELS

In this section we analyze patterns across 10 of the high-performing evolved models (the models highlighted in Fig. 1b). See details and example `__str__` outputs for some of the models in Sec. C.

Pattern 1: bounded display complexity by construction. Across the 10 models, the agents often bound the size of `__str__` through hard architectural caps rather than soft regularization on a flexible model. The caps take many forms: a fixed tree depth (e.g., `TinyDTDepth2`), a top-k feature budget, a fixed number of quantile knots or breakpoints in a hinge basis, a forward-selected basis with rounded coefficients, or a single-line symbolic equation distilled from a stronger teacher. This contrasts with strong baselines such as EBM, FIGS, and PyGAM, whose displays grow with the data and which sit at the worst end of the held-out interpretability axis despite predicting well (Fig. 1b). This intrinsic interpretability principle echoes classical interpretable-ML works that treat complexity as a model constraint [8, 30, 10], and it keeps the printed string short enough for the LLM evaluator to simulate end-to-end.

Pattern 2: display optimization as a first-class design axis. Eight of the ten models pair a standard predictor with a non-trivial display strategy that is optimized separately from the architecture itself. Examples include (i) per-feature gating that renders a learned shape function as a single coefficient when an R^2 test passes and a short piecewise table otherwise (e.g. `SmartAdditive`); (ii) collapsing a hinge or basis expansion into an effective linear slope for the printed equation while keeping the precise predictor underneath (e.g. `HingeEBM`); and (iii) compiling a distilled student into a single-row symbolic equation written specifically for `__str__`. The choice of display also varies based on the underlying agent: Codex with `GPT-5.3` tends toward sparse symbolic displays, while Claude with `Opus-4.7` tends toward GAM-style displays.

Autoresearch with newer agents We also run the AGENTIC-IMODELS loop with newer models (Claude `Opus-5.5` and `Fable-5.1`) on three existing interpretable model families, where the model family is fixed and the agent only tries to improve predictive performance or fitting speed. In each case the resulting model outperforms existing methods in its family on held-out TabArena datasets [67]: `GPGam`, a GAM whose shape functions are Gaussian processes, achieves a better mean rank than EBM [29]; `FastSmallTree`, an exact solver for the GOSDT objective [79], finds provably optimal sparse trees about 20× faster than GOSDT; and `FastRiskScore` fits sparse integer risk scores about 225× faster than FasterRisk [80] with equal or lower training loss on most problems. All three models are available in `imodels` [12], and details on the methods and experiments are at `https://csinva.io/imodels/blog.html`.

5 Discussion

Decades of interpretability research have optimized models for a human reader. In the coming decades, with the increasing capabilities of AI agents, building tools for agents to reliably interpret data will be critical to ensuring that insights from data are correctly understood and acted upon. AGENTIC-IMODELS is a first step toward taking this shift seriously, and may be the first of many works that adapt the structural properties that aid human interpretability (e.g., sparsity, additivity, monotonicity) into LLM-suitable properties.

Limitations. One limitation of the study here is that our end-to-end ADS evaluations rely on LLM-as-judge for scoring, which may introduce some bias or artifacts (although notably the underlying ground-truth analysis is conducted by expert humans, making the judging task easier). Similarly, our interpretability tests use LLMs for evaluation. While this is effective for evaluating the agent-interpretability of the models being tested, it is unclear how well this measures the human-interpretability of representations, a question that may be explored in future user studies. As LLM user simulators become more accurate [81, 82], they could be used to automatically evaluate the interpretability of models to provide more reliable assessments that match human judgements. Additionally, the interpretability tests we design do not cover all possible aspects of model interpretability, and particular applications may require specialized tests beyond our current suite.

Another limitation is the presence of reward hacking in the experiments, which highlights that our metric is not a perfect proxy. However, the strong correlation between held-out agent interpretability

scores and downstream performance suggests that the metric captures useful signal despite being imperfect. Future works may want to design the interpretability tests and the agentic loop to be more robust to increasingly sophisticated agents. For example, tests could be exposed only through an API so the agent never sees the questions or expected answers in plain text. Finally, the agent loop is expensive in LLM API calls, both for the coding agent and for the interpretability evaluator, although these costs may decrease as LLM inference becomes cheaper and more efficient over time.

Future work. A natural next step is to broaden the scope of the framework along two axes: across tasks (e.g. classification, time series, text data, larger datasets) and across more general tools (e.g. text explanation pipelines, causal interventions). The same test suite could also be repurposed as an optimization target for improving existing model packages without re-architecting them [12, 13]. Downstream, incorporating the evolved models into real-world agentic workflows (e.g., scientific discovery pipelines) would test whether the improvements transfer to more complex analyses. On the human side, our framework could be modified to optimize human-AI collaborative workflows, rather than focusing on agents or humans in isolation [83, 84]. As LLMs themselves improve at parsing structured models [85], AGENTIC-IMODELS will be increasingly powerful and both the evolved tools and the metric they optimize against will enable more sophisticated, potentially superhuman interpretation of complex data.

References

[1] Siyuan Guo, Cheng Deng, Ying Wen, Hechang Chen, Yi Chang, and Jun Wang. Ds-agent: Automated data science by empowering large language models with case-based reasoning. *arXiv preprint arXiv:2402.17453*, 2024.

[2] Fan Nie, Junlin Wang, Harper Hua, Federico Bianchi, Yongchan Kwon, Zhenting Qi, Owen Queen, Shang Zhu, and James Zou. Dsgym: A holistic framework for evaluating and training data science agents. *arXiv preprint arXiv:2601.16344*, 2026.

[3] Ziru Chen, Shijie Chen, Yuting Ning, Qianheng Zhang, Boshi Wang, Botao Yu, Yifei Li, Zeyi Liao, Chen Wei, Zitong Lu, et al. Scienceagentbench: Toward rigorous assessment of language agents for data-driven scientific discovery. *arXiv preprint arXiv:2410.05080*, 2024.

[4] Hanchen Wang, Tianfan Fu, Yuanqi Du, Wenhao Gao, Kexin Huang, Ziming Liu, Payal Chandak, Shengchao Liu, Peter Van Katwyk, Andreea Deac, et al. Scientific discovery in the age of artificial intelligence. *Nature*, 620(7972):47–60, 2023.

[5] Chris Lu, Cong Lu, Robert Tjarko Lange, Jakob Foerster, Jeff Clune, and David Ha. The ai scientist: Towards fully automated open-ended scientific discovery. *arXiv preprint arXiv:2408.06292*, 2024.

[6] Samuel R Bowman, Jeeyoon Hyun, Ethan Perez, Edwin Chen, Craig Pettit, Scott Heiner, Kamilė Lukošiūtė, Amanda Askell, Andy Jones, Anna Chen, et al. Measuring progress on scalable oversight for large language models. *arXiv preprint arXiv:2211.03540*, 2022.

[7] Balint Gyevnar and Atoosa Kasirzadeh. Ai safety for everyone. *Nature Machine Intelligence*, 7(4):531–542, 2025.

[8] Cynthia Rudin. Please stop explaining black box models for high stakes decisions. *arXiv preprint arXiv:1811.10154*, 2018.

[9] W. James Murdoch, Chandan Singh, Karl Kumbier, Reza Abbasi-Asl, and Bin Yu. Definitions, methods, and applications in interpretable machine learning. *Proceedings of the National Academy of Sciences of the United States of America*, 116(44):22071–22080, 2019.

[10] L. Breiman, J. H. Friedman, R. A. Olshen, and C. J. Stone. *Classification and Regression Trees*. Wadsworth and Brooks, Monterey, CA, 1984.

[11] Robert Tibshirani. Regression shrinkage and selection via the lasso. *Journal of the Royal Statistical Society. Series B (Methodological)*, pages 267–288, 1996.

[12] Chandan Singh, Keyan Nasseri, Yan Shuo Tan, Tiffany Tang, and Bin Yu. imodels: a python package for fitting interpretable models. *Journal of Open Source Software*, 6(61):3192, 2021.

[13] Harsha Nori, Samuel Jenkins, Paul Koch, and Rich Caruana. Interpretml: A unified framework for machine learning interpretability. *arXiv preprint arXiv:1909.09223*, 2019.

[14] Samuel G. Z. Asher, Janet Malzahn, Jessica M. Persano, Elliot J. Paschal, Andrew C. W. Myers, and Andrew B. Hall. Do claude code and codex p-hack? sycophancy and statistical analysis in large language models, February 2026. Preprint.

[15] Zachary T Rewolinski, Austin V Zane, Hao Huang, Chandan Singh, Chenglong Wang, Jianfeng Gao, and Bin Yu. Sanity checks for agentic data science. *arXiv preprint arXiv:2604.11003*, 2026.

[16] Ziming Luo, Atoosa Kasirzadeh, and Nihar B Shah. The more you automate, the less you see: Hidden pitfalls of ai scientist systems. *arXiv preprint arXiv:2509.08713*, 2025.

[17] Venkatesh Sivaraman, Patrick Vossler, Adam Perer, Julian Hong, and Jean Feng. More than "means to an end": Supporting reasoning with transparently designed ai data science processes, 2026.

[18] Harmanpreet Kaur, Harsha Nori, Samuel Jenkins, Rich Caruana, Hanna Wallach, and Jennifer Wortman Vaughan. Interpreting interpretability: understanding data scientists' use of interpretability tools for machine learning. In *Proceedings of the 2020 CHI conference on human factors in computing systems*, pages 1–14, 2020.

[19] Sungsoo Ray Hong, Jessica Hullman, and Enrico Bertini. Human factors in model interpretability: Industry practices, challenges, and needs. *Proceedings of the ACM on Human-Computer Interaction*, 4(CSCW1):1–26, 2020.

[20] Finale Doshi-Velez and Been Kim. A roadmap for a rigorous science of interpretability. *arXiv preprint arXiv:1702.08608*, 2017.

[21] Zachary C Lipton. The mythos of model interpretability. *arXiv preprint arXiv:1606.03490*, 2016.

[22] Isaac Lage, Emily Chen, Jeffrey He, Menaka Narayanan, Been Kim, Sam Gershman, and Finale Doshi-Velez. An evaluation of the human-interpretability of explanation. *arXiv preprint arXiv:1902.00006*, 2019.

[23] Ken Gu, Ruoxi Shang, Ruien Jiang, Keying Kuang, Richard-John Lin, Donghe Lyu, Yue Mao, Youran Pan, Teng Wu, Jiaqian Yu, et al. Blade: Benchmarking language model agents for data-driven science. *arXiv preprint arXiv:2408.09667*, 2024.

[24] Cynthia Rudin, Chaofan Chen, Zhi Chen, Haiyang Huang, Lesia Semenova, and Chudi Zhong. Interpretable machine learning: Fundamental principles and 10 grand challenges. *arXiv preprint arXiv:2103.11251*, 2021.

[25] J. Ross Quinlan. Induction of decision trees. *Machine learning*, 1(1):81–106, 1986.

[26] Benjamin Letham, Cynthia Rudin, Tyler H McCormick, and David Madigan. Interpretable classifiers using rules and bayesian analysis: Building a better stroke prediction model. 2015.

[27] Trevor Hastie and Robert Tibshirani. Generalized additive models. *Statistical Science*, 1(3):297–318, 1986.

[28] Yin Lou, Rich Caruana, Johannes Gehrke, and Giles Hooker. Accurate intelligible models with pairwise interactions. In *Proceedings of the 19th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 623–631, 2013.

[29] Rich Caruana, Yin Lou, Johannes Gehrke, Paul Koch, Marc Sturm, and Noemie Elhadad. Intelligible models for healthcare: Predicting pneumonia risk and hospital 30-day readmission. In *Proceedings of the 21th ACM SIGKDD international conference on knowledge discovery and data mining*, pages 1721–1730, 2015.

[30] Berk Ustun and Cynthia Rudin. Supersparse linear integer models for optimized medical scoring systems. *Machine Learning*, 102:349–391, 2016.

[31] Chandan Singh, Armin Askari, Rich Caruana, and Jianfeng Gao. Augmenting interpretable models with large language models during training. *Nature Communications*, 14(1):7913, 2023.

[32] Andreas Mueller, Julien Siems, Harsha Nori, David Salinas, Arber Zela, Rich Caruana, and Frank Hutter. Gamformer: In-context learning for generalized additive models. *arXiv preprint arXiv:2410.04560*, 2024.

[33] Yufan Zhuang, Liyuan Liu, Chandan Singh, Jingbo Shang, and Jianfeng Gao. Learning a decision tree algorithm with transformers. *arXiv preprint arXiv:2402.03774*, 2024.

[34] Hanyu Li, Haoyu Liu, Tingyu Zhu, Tianyu Guo, Zeyu Zheng, Xiaotie Deng, and Michael I Jordan. Ida-bench: Evaluating llms on interactive guided data analysis. *arXiv preprint arXiv:2505.18223*, 2025.

[35] Zhangde Song, Jieyu Lu, Yuanqi Du, Botao Yu, Thomas M Pruyn, Yue Huang, Kehan Guo, Xiuzhe Luo, Yuanhao Qu, Yi Qu, et al. Evaluating large language models in scientific discovery. *arXiv preprint arXiv:2512.15567*, 2025.

[36] Yifei Li, Hanane Nour Moussa, Ziru Chen, Shijie Chen, Botao Yu, Mingyi Xue, Benjamin Burns, Tzu-Yao Chiu, Vishal Dey, Zitong Lu, et al. Autosdt: Scaling data-driven discovery tasks toward open co-scientists. *arXiv preprint arXiv:2506.08140*, 2025.

[37] Jaehyun Nam, Jinsung Yoon, Jiefeng Chen, and Tomas Pfister. Ds-star: Data science agent via iterative planning and verification. *arXiv preprint arXiv:2509.21825*, 2025.

[38] Joachim Baumann, Paul Röttger, Aleksandra Urman, Albert Wendsjö, Flor Miriam Plaza-del Arco, Johannes B Gruber, and Dirk Hovy. Large language model hacking: Quantifying the hidden risks of using llms for text annotation. *arXiv preprint arXiv:2509.08825*, 2025.

[39] Yu-Min Tseng, Wei-Lin Chen, Chung-Chi Chen, and Hsin-Hsi Chen. Evaluating large language models as expert annotators. *arXiv preprint arXiv:2508.07827*, 2025.

[40] Ruiqi Zhong, Peter Zhang, Steve Li, Jinwoo Ahn, Dan Klein, and Jacob Steinhardt. Goal driven discovery of distributional differences via language descriptions. *ArXiv*, abs/2302.14233, 2023.

[41] Varun Babbar, Zhicheng Guo, and Cynthia Rudin. " what is different between these datasets?" a framework for explaining data distribution shifts. *Journal of Machine Learning Research*, 26(180):1–64, 2025.

[42] Zhiying Zhu, Weixin Liang, and James Zou. Gsclip: A framework for explaining distribution shifts in natural language. *arXiv preprint arXiv:2206.15007*, 2022.

[43] Rakesh R Menon, Kerem Zaman, and Shashank Srivastava. MaNtLE: Model-agnostic natural language explainer. *arXiv preprint arXiv:2305.12995*, 2023.

[44] Steven Bills, Nick Cammarata, Dan Mossing, Henk Tillman, Leo Gao, Gabriel Goh, Ilya Sutskever, Jan Leike, Jeff Wu, and William Saunders. Language models can explain neurons in language models, 2023.

[45] Chandan Singh, Aliyah R Hsu, Richard Antonello, Shailee Jain, Alexander G Huth, Bin Yu, and Jianfeng Gao. Explaining black box text modules in natural language with language models. *arXiv preprint arXiv:2305.09863*, 2023.

[46] Jiaojiao Han, Wujiang Xu, Mingyu Jin, and Mengnan Du. Sage: An agentic explainer framework for interpreting sae features in language models. In *Proceedings of the 19th Conference of the European Chapter of the Association for Computational Linguistics (Volume 5: Industry Track)*, pages 483–495, 2026.

[47] Tamar Rott Shaham, Sarah Schwettmann, Franklin Wang, Achyuta Rajaram, Evan Hernandez, Jacob Andreas, and Antonio Torralba. A multimodal automated interpretability agent. In *Forty-first International Conference on Machine Learning*, 2024.

[48] Samuel Schmidgall, Yusheng Su, Ze Wang, Ximeng Sun, Jialian Wu, Xiaodong Yu, Jiang Liu, Michael Moor, Zicheng Liu, and Emad Barsoum. Agent laboratory: Using llm agents as research assistants. *Findings of the Association for Computational Linguistics: EMNLP 2025*, pages 5977–6043, 2025.

[49] Kyle Swanson, Wesley Wu, Nash L Bulaong, John E Pak, and James Zou. The virtual lab of ai agents designs new sars-cov-2 nanobodies. *Nature*, 646(8085):716–723, 2025.

[50] Zonglin Yang, Xinya Du, Junxian Li, Jie Zheng, Soujanya Poria, and Erik Cambria. Large language models for automated open-domain scientific hypotheses discovery. In *Findings of the Association for Computational Linguistics: ACL 2024*, pages 13545–13565, 2024.

[51] Alexander Novikov, Ngân Vũ, Marvin Eisenberger, Emilien Dupont, Po-Sen Huang, Adam Zsolt Wagner, Sergey Shirobokov, Borislav Kozlovskii, Francisco JR Ruiz, Abbas Mehrabian, et al. Alphaevolve: A coding agent for scientific and algorithmic discovery. *arXiv preprint arXiv:2506.13131*, 2025.

[52] Bernardino Romera-Paredes, Mohammadamin Barekatain, Alexander Novikov, Matej Balog, M Pawan Kumar, Emilien Dupont, Francisco JR Ruiz, Jordan S Ellenberg, Pengming Wang, Omar Fawzi, et al. Mathematical discoveries from program search with large language models. *Nature*, pages 1–3, 2023.

[53] Chandan Singh, John X. Morris, Jyoti Aneja, Alexander M. Rush, and Jianfeng Gao. Explaining patterns in data with language models via interpretable autoprompting, 2023.

[54] Yiping Wang, Shao-Rong Su, Zhiyuan Zeng, Eva Xu, Liliang Ren, Xinyu Yang, Zeyi Huang, Xuehai He, Luyao Ma, Baolin Peng, et al. Thetaevolve: Test-time learning on open problems. *arXiv preprint arXiv:2511.23473*, 2025.

[55] Mert Yuksekgonul, Daniel Koceja, Xinhao Li, Federico Bianchi, Jed McCaleb, Xiaolong Wang, Jan Kautz, Yejin Choi, James Zou, Carlos Guestrin, et al. Learning to discover at test time. *arXiv preprint arXiv:2601.16175*, 2026.

[56] Ziyu Ma, Shidong Yang, Yuxiang Ji, Xucong Wang, Yong Wang, Yiming Hu, Tongwen Huang, and Xiangxiang Chu. Skillclaw: Let skills evolve collectively with agentic evolver. *arXiv preprint arXiv:2604.08377*, 2026.

[57] Huichi Zhou, Siyuan Guo, Anjie Liu, Zhongwei Yu, Ziqin Gong, Bowen Zhao, Zhixun Chen, Menglong Zhang, Yihang Chen, Jinsong Li, et al. Memento-skills: Let agents design agents. *arXiv preprint arXiv:2603.18743*, 2026.

[58] Salaheddin Alzubi, Noah Provenzano, Jaydon Bingham, Weiyuan Chen, and Tu Vu. Evoskill: Automated skill discovery for multi-agent systems. *arXiv preprint arXiv:2603.02766*, 2026.

[59] Shuaike Shen, Wenduo Cheng, Mingqian Ma, Alistair Turcan, Martin Jinye Zhang, and Jian Ma. Skillfoundry: Building self-evolving agent skill libraries from heterogeneous scientific resources. *arXiv preprint arXiv:2604.03964*, 2026.

[60] Yoonho Lee, Roshen Nair, Qizheng Zhang, Kangwook Lee, Omar Khattab, and Chelsea Finn. Meta-harness: End-to-end optimization of model harnesses. *arXiv preprint arXiv:2603.28052*, 2026.

[61] Biswa Sengupta and Jinhua Wang. Harbor: Automated harness optimization. *arXiv preprint arXiv:2604.20938*, 2026.

[62] Mirac Suzgun, Mert Yuksekgonul, Federico Bianchi, Dan Jurafsky, and James Zou. Dynamic cheatsheet: Test-time learning with adaptive memory. In *Proceedings of the 19th Conference of the European Chapter of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 7080–7106, 2026.

[63] Tianxin Wei, Noveen Sachdeva, Benjamin Coleman, Zhankui He, Yuanchen Bei, Xuying Ning, Mengting Ai, Yunzhe Li, Jingrui He, Ed H Chi, et al. Evo-memory: Benchmarking llm agent test-time learning with self-evolving memory. *arXiv preprint arXiv:2511.20857*, 2025.

[64] Jenny Zhang, Shengran Hu, Cong Lu, Robert Lange, and Jeff Clune. Darwin godel machine: Open-ended evolution of self-improving agents. *arXiv preprint arXiv:2505.22954*, 2025.

[65] Yufan Zhuang, Chandan Singh, Liyuan Liu, Yelong Shen, Dinghuai Zhang, Jingbo Shang, Jianfeng Gao, and Weizhu Chen. Test-time recursive thinking: Self-improvement without external feedback. *arXiv preprint arXiv:2602.03094*, 2026.

[66] Fabian Pedregosa, Ga ë l Varoquaux, Alexandre Gramfort, Vincent Michel, Bertrand Thirion, Olivier Grisel, Mathieu Blondel, Peter Prettenhofer, Ron Weiss, Vincent Dubourg, et al. Scikit-learn: Machine learning in python. *the Journal of machine Learning research*, 12(Oct):2825–2830, 2011.

[67] Nick Erickson, Lennart Purucker, Andrej Tschalzev, David Holzmüller, Prateek Mutalik Desai, David Salinas, and Frank Hutter. Tabarena: A living benchmark for machine learning on tabular data. *arXiv preprint arXiv:2506.16791*, 2025.

[68] Randal S Olson, William La Cava, Patryk Orzechowski, Ryan J Urbanowicz, and Jason H Moore. Pmlb: a large benchmark suite for machine learning evaluation and comparison. *BioData mining*, 10(1):36, 2017.

[69] Léo Grinsztajn, Edouard Oyallon, and Gaël Varoquaux. Why do tree-based models still outperform deep learning on typical tabular data? *Advances in neural information processing systems*, 35:507–520, 2022.

[70] Abhineet Agarwal, Yan Shuo Tan, Omer Ronen, Chandan Singh, and Bin Yu. Hierarchical shrinkage: improving the accuracy and interpretability of tree-based methods. *arXiv:2202.00858*, 2 2022. arXiv: 2202.00858.

[71] Trevor J Hastie. *Generalized additive models*. Routledge, 2017.

[72] Yan Shuo Tan, Chandan Singh, Keyan Nasseri, Abhineet Agarwal, and Bin Yu. Fast interpretable greedy-tree sums (figs). *arXiv:2201.11931 [cs, stat]*, 1 2022. arXiv: 2201.11931.

[73] J. H. Friedman and B. E. Popescu. Predictive learning via rule ensembles. *The Annals of Applied Statistics*, 2(3):916–954, 2008.

[74] Leo Breiman. Random forests. *Machine Learning*, 45(1):5–32, 10 2001.

[75] Noah Hollmann, Samuel Müller, Lennart Purucker, Arjun Krishnakumar, Max Körfer, Shi Bin Hoo, Robin Tibor Schirrmeister, and Frank Hutter. Accurate predictions on small data with a tabular foundation model. *Nature*, 637(8045):319–326, 2025.

[76] Aaron Hurst, Adam Lerer, Adam P Goucher, Adam Perelman, Aditya Ramesh, Aidan Clark, AJ Ostrow, Akila Welihinda, Alan Hayes, Alec Radford, et al. Gpt-4o system card. *arXiv preprint arXiv:2410.21276*, 2024.

[77] Alicia H Munnell, Geoffrey M.B Tootell, Lynn E Browne, and James McEneaney. Mortgage lending in boston: interpreting hmda data. *The American economic review*, 86(1):25 – 53, 1996.

[78] Margaret C. Crofoot, Ian C. Gilby, Martin C. Wikelski, and Roland W. Kays. Interaction location outweighs the competitive advantage of numerical superiority in cebus capucinus intergroup contests. *Proceedings of the National Academy of Sciences*, 105(2):577–581, 2008.

[79] Jimmy Lin, Chudi Zhong, Diane Hu, Cynthia Rudin, and Margo Seltzer. Generalized and scalable optimal sparse decision trees. In *International Conference on Machine Learning*, pages 6150–6160. PMLR, 2020.

[80] Jiachang Liu, Chudi Zhong, Boxuan Li, Margo Seltzer, and Cynthia Rudin. Fasterrisk: fast and accurate interpretable risk scores. *Advances in Neural Information Processing Systems*, 35:17760–17773, 2022.

[81] Tarek Naous, Philippe Laban, Wei Xu, and Jennifer Neville. Flipping the dialogue: Training and evaluating user language models. *arXiv preprint arXiv:2510.06552*, 2025.

[82] Shirley Wu, Evelyn Choi, Arpandeep Khatua, Zhanghan Wang, Joy He-Yueya, Tharindu Cyril Weerasooriya, Wei Wei, Diyi Yang, Jure Leskovec, and James Zou. Humanlm: Simulating users with state alignment beats response imitation. *arXiv preprint arXiv:2603.03303*, 2026.

[83] Gagan Bansal, Tongshuang Wu, Joyce Zhou, Raymond Fok, Besmira Nushi, Ece Kamar, Marco Tulio Ribeiro, and Daniel Weld. Does the whole exceed its parts? The effect of AI explanations on complementary team performance. In *Proceedings of the 2021 CHI Conference on Human Factors in Computing Systems*, pages 1–16, 2021.

[84] Jean Feng, Avni Kothari, Patrick Vossler, Andrew Bishara, Lucas Zier, Newton Addo, Aaron Kornblith, Yan Shuo Tan, and Chandan Singh. Human-ai co-design for clinical prediction models. *arXiv preprint arXiv:2601.09072*, 2026.

[85] Benjamin J Lengerich, Sebastian Bordt, Harsha Nori, Mark E Nunnally, Yin Aphinyanaphongs, Manolis Kellis, and Rich Caruana. LLMs understand glass-box models, discover surprises, and suggest repairs. *arXiv preprint arXiv:2308.01157*, 2023.

A Experimental setup details

A.1 Interpretability test details

Table A2 lists all 43 development interpretability tests (the set used inside the AGENTIC-IMODELS optimization loop) with descriptions and pass rates pooled across all models (baselines + evolved candidates) from the 9 AGENTIC-IMODELS runs mentioned in Sec. 4.2. Table A3 lists the 157 held-out tests used for post-hoc evaluation, with pass rates pooled the same way.

Grading target: fitted predictor vs. data-generating function. Each test grades the LLM's answer against one of two references: the *fitted* model's own `.predict()` output, or the *data-generating* function used to synthesize the data. Table A1 breaks down the 43 development tests by this distinction. The large majority, $35/43$ (81%), are graded against `.predict()` (all point-simulation and complex-function-simulation tests, both counterfactual tests, most sensitivity tests, and the decision-region test), so a display that decouples from the deployed predictor fails them, which bounds (though does not eliminate) unfaithfulness. Only $7/43$ (16%) are graded against the data-generating function — exactly the feature-attribution and threshold-recovery tests — where fit quality and simulatability are conflated (as noted by Reviewer dD5H); we report these two families as separate sub-scores so that an underfitting-but-readable model is not penalized on the simulatability axis. The remaining test (compactness) grades a structural property.

Table A1: **Grading target of the 43 development interpretability tests.** 81% are graded against the *fitted* model's own `.predict()` outputs (so a display that diverges from the deployed predictor fails them); only 16% are graded against the *data-generating* function (where fit quality and simulatability are conflated, per Reviewer dD5H). We report these two families as separate sub-scores in the revision.

Test category	Fitted-graded	Data-gen-graded	Structural
Feature attribution (6)	1	5	0
Point simulation (17)	17	0	0
Sensitivity (6)	4	2	0
Counterfactual (2)	2	0	0
Structural (2)	1	0	1
Complex-fn. simulation (10)	10	0	0
Total (43)	**35 (81%)**	**7 (16%)**	**1**

A.2 Dataset details

Table A4 lists all 65 datasets used in the main experiment, including 7 from OpenML and 58 from PMLB [68]. Each dataset is subsampled to at most 1,000 samples and 50 features during evaluation. Table A5 lists the 16 held-out OpenML datasets used in the generalization experiment (Sec. B.2).

Table A2: All 43 interpretability tests used inside the AGENTIC-IMODELS optimization loop, grouped by cognitive operation. Pass rates are pooled across the 9 AGENTIC-IMODELS runs mentioned in Sec. 4.2; each run contributes its full set of evaluated models (baselines + evolved candidates).

Category	Test	Description	Pass %
Feature attribution	most important feature	Identify the single most important feature	97%
	feature ranking	Rank top 3 features by importance	98%
	irrelevant features	Identify features with negligible effect	87%
	sparse feature set	Identify active features in 10-feature data	82%
	dominant feature sample	Identify dominant feature for a specific sample	100%
	sign of effect	Determine sign and magnitude of a feature effect	76%
Point simulation	point prediction	Predict output for a specific input	90%
	counterfactual prediction	Predict output for a different input	89%
	all features active	Predict with all features non-zero	66%
	two feature perturbation	Predict under simultaneous two-feature changes	68%
	mixed sign goes negative	Handle opposite-signed features	41%
	pairwise anti intuitive	Compare two complex samples	13%
	predict above threshold	Predict a sample above a threshold	72%
	predict below threshold	Predict a sample below a threshold	85%
	simulate mixed sign	Trace 6 features with mixed-sign coefficients	53%
	simulate all active	Simulate 5-feature linear with mixed signs	27%
	simulatability	Simulate complex 4-feature prediction	44%
	eight features	8-feature linear mixed-sign	46%
	fifteen features sparse	15 features, only 3 active	23%
	twenty features sparse	20 features, only 4 active	35%
	twelve features all active	12 features all contributing	24%
	quadratic	$y = 3x_0^2 - 2x_1^2 + x_2$	59%
	friedman1	Classic Friedman-1 nonlinear benchmark	63%
Sensitivity analysis	direction of change	Direction of output change when a feature increases	82%
	quantitative sensitivity	Quantify sensitivity over larger ranges	88%
	unit sensitivity	Exact unit change prediction (tight tolerance)	79%
	nonlinear direction	Predict output under nonlinearity	66%
	threshold identification	Identify threshold values separating predictions	75%
	nonlinear threshold	Detect hockey-stick/ReLU-like behavior	52%
Counterfactual reasoning	counterfactual target	Solve inverse problem: find input for target output	25%
	quadratic counterfactual	Inverse problem on nonlinear function	52%
Structural understanding	compactness	Can the model be computed in $\leq$10 operations?	81%
	decision region	Identify decision boundaries	87%
Complex fn. simulation	simulate double threshold	Two step-thresholds on one feature	66%
	simulate additive nonlinear	Simulate $y = 3\max(0, x_0) + 2\sin(x_1) + x_2$	60%
	simulate interaction	Simulate $y = 3x_0 + 2x_1 + 1.5x_0x_1$	50%
	triple interaction	Multi-way interactions	79%
	cascading threshold	If-then cascading structure	74%
	exponential decay	$y = 5\exp(-x_0) + 2x_1$	48%
	piecewise three segment	3-segment piecewise linear	83%
	sinusoidal	Trigonometric nonlinearity	46%
	abs value	$y = 3\lvert x_0\rvert - 2\lvert x_1\rvert + x_2$	52%
	nested threshold	Nested if-then logic	59%

Table A3: All 157 new interpretability tests used in the generalization experiment, aggregate. N is the number of distinct tests in each sub-category; pass rates are pooled across the 9 AGENTIC-IMODELS runs mentioned in Sec. 4.2.

Category	Sub-category	Description	N	Pass %
Feature attribution (26 tests)	top feature	Most influential feature; answer varies across tests	3	88%
	zero-effect feature	Identify the single inert feature	3	7%
	top-2 ranking	List the two most influential features in order	3	48%
	irrelevant set	Name every feature whose effect is negligible	3	44%
	sign + unit effect	Signed change from a unit increase in a chosen feature	3	75%
	dominant at sample	Dominant contributing feature for a given sample	3	94%
	second feature	Identify the second-most-important feature	3	77%
	count active	Count of features with non-negligible effect	3	48%
	pair importance	Between two named features, which matters more	2	89%
Point simulation (26 tests)	small linear	Predict output on 2–4 feature linear data	3	29%
	all-features active	Predict with every feature non-zero (7–9 feats)	3	90%
	sparse many-feature	Predict on 10–18 feats with only 3–4 active	3	58%
	alternating sign	Predict when coefficients alternate in sign	3	51%
	tail query	Predict at feature values with $\|x_i\| \geq 1.6$	3	60%
	irrational inputs	Predict at irregular, non-round query points	3	47%
	partial input	Predict when only a subset of features is specified	3	59%
	non-zero intercept	Predict under a shifted intercept	3	26%
	compare two points	Compare predictions at two input samples	2	50%
Sensitivity analysis (26 tests)	direction of change	Signed delta for a one-unit change at varied bases	3	88%
	wide-range	Quantify sensitivity over a large interval	3	62%
	tight unit change	Unit change with tight numerical tolerance	3	76%
	crossing point	Find x_k at which prediction crosses a target	3	81%
	two-feature change	Delta when two features change simultaneously	3	27%
	decrease phrasing	Signed delta phrased as a decrease	3	33%
	multi-unit step	Delta under a multi-unit change	3	19%
	non-zero base	Unit change from a non-zero baseline	3	70%
	small step	Sub-unit step sensitivity	2	95%
Counterfactual reasoning (26 tests)	inverse on linear	Value of x_k that attains a target prediction	3	64%
	inverse, varied feat	Counterfactual targeting a rotated feature	3	60%
	inverse, sparse data	Counterfactual with many inert features	3	26%
	inverse, nonlinear	Counterfactual on ReLU/quad/piecewise/abs data	3	63%
	with intercept	Counterfactual under a non-zero intercept	3	24%
	reverse direction	Counterfactual requiring a downward change	3	35%
	large change	Counterfactual requiring a sizeable feature move	3	55%
	mid-index feature	Counterfactual on a feature with middle index	3	54%
	negative target	Counterfactual to a negative target prediction	2	42%
Structural understanding (26 tests)	decision region	Threshold of x_k for a specified prediction level	3	71%
	compactness	Can the model be evaluated in $\leq N$ operations?	3	67%
	argmax	Value of x_k that maximises prediction	3	33%
	argmin	Value of x_k that minimises prediction	3	34%
	monotonic direction	Monotonic direction of a chosen feature	3	56%
	output range	Approximate range of the model's predictions	3	45%
	decision mid-output	Threshold of x_k for a mid-range prediction	3	76%
	argmax nonlinear	Argmax on a nonlinear data-generating process	3	46%
	plateau region	Region where predictions are approximately constant	2	86%
Complex fn. simulation (27 tests)	quadratic	Simulate $y = c_0 + c_1 x_k + c_2 x_k^2$	3	92%
	pairwise interaction	Simulate linear $+x_i \cdot x_j$ interaction	3	74%
	triple interaction	Simulate multi-way feature products	3	93%
	Friedman-1	Simulate the Friedman-1 nonlinear benchmark	3	22%
	cascading threshold	If-then cascading data-generating process	3	84%
	exponential decay	Simulate $y = c\exp(-x_k) + \text{linear}$	3	25%
	3-segment piecewise	Simulate 3-segment piecewise linear function	3	91%
	sinusoidal	Simulate trigonometric nonlinearity	3	89%
	absolute value	Simulate $y = c_1\|x_i\| + c_2\|x_j\| + \text{linear}$	3	92%

Table A4: The 65 datasets used in the main experiment (7 OpenML, 58 PMLB). n = samples, p = features (before subsampling).

Dataset	n	p	Dataset	n	p
abalone	4176	8	4544_GeographicalOr..	1059	117
california	20634	8	485_analcatdata_ve..	48	4
cpu_act	8192	21	503_wind	6574	14
elevators	16599	16	505_tecator	240	124
house_16H	22784	16	519_vinnie	380	2
kin8nm	8192	8	522_pm10	500	7
pol	15000	26	523_analcatdata_ne..	100	2
1027_ESL	488	4	527_analcatdata_el..	67	14
1028_SWD	1000	10	529_pollen	3848	4
1029_LEV	1000	4	537_houses	20640	8
1030_ERA	1000	4	542_pollution	60	15
1089_USCrime	47	13	547_no2	500	7
1096_FacultySalaries	50	4	556_analcatdata_ap..	475	3
1191_BNG_pbc	1000000	18	557_analcatdata_ap..	475	3
1193_BNG_lowbwt	31104	9	560_bodyfat	252	14
1196_BNG_pharynx	1000000	10	561_cpu	209	7
1199_BNG_echoMonths	17496	9	562_cpu_small	8192	12
1201_BNG_breastTumor	116640	9	564_fried	40768	10
1203_BNG_pwLinear	177147	10	573_cpu_act	8192	21
1595_poker	1025010	10	574_house_16H	22784	16
192_vineyard	52	2	659_sleuth_ex1714	47	7
197_cpu_act	8192	21	663_rabe_266	120	2
201_pol	15000	48	665_sleuth_case2002	147	6
210_cloud	108	5	666_rmftsa_ladata	508	10
215_2dplanes	40768	10	678_visualizing_en..	111	3
218_house_8L	22784	8	687_sleuth_ex1605	62	5
225_puma8NH	8192	8	690_visualizing_ga..	323	4
227_cpu_small	8192	12	695_chatfield_4	235	12
228_elusage	55	2	706_sleuth_case1202	93	6
229_pwLinear	200	10	712_chscase_geyser1	222	2
230_machine_cpu	209	6	banana	5300	2
294_satellite_image	6435	36	titanic	2099	8
344_mv	40768	10			

Table A5: The 16 held-out OpenML datasets (suite 335) used in the generalization experiment. n = samples, p = features (before subsampling).

ID	Dataset	n	p	ID	Dataset	n	p
44055	analcatdata_sup..	4052	7	44068	particulate-matt..	394299	6
44056	visualizing_soil	8641	4	44069	SGEMM_GPU_kern..	241600	9
44059	diamonds	53940	9	45041	topo_2_1	8885	255
44061	Mercedes_Benz_..	4209	359	45043	seattlecrime6	52031	4
44062	Brazilian_houses	10692	11	45045	delays_zurich_..	5465575	11
44063	Bike_Sharing_D..	17379	11	45046	Allstate_Claims..	188318	124
44065	nyc-taxi-green-d..	581835	16	45047	Airlines_DepDel..	1000000	5
44066	house_sales	21613	17	45048	medical_charges	163065	3

B Extended results and ablations

B.1 Ablation: alternative LLM evaluators for the interpretability tests

`GPT-5.4` evaluator used post-hoc after models have been optimized. We reevaluate the 41 baseline + evolved models from the Claude `Opus-4.6` medium-effort main run under `GPT-5.4` instead of `GPT-4o` on the 43 development-set interpretability tests. Only the LLM evaluator changes, so prediction ranks are unaffected. Fig. B1 shows two views of the same 41-model pool. Fig. B1a shows where each model lands in the predictive performance versus interpretability plot under `GPT-5.4` (predictive performance is the same as is used in Fig. 3a-c). Despite the score compression for baselines (EBM and TabPFN drop to ≈ 0; GBM from 0.53 to 0.15), the evolved Pareto frontier is preserved: `HingeEBM (5bag)` retains a rank of 13 at 50% interpretability, well outside the baseline Pareto region.

Fig. B1b plots each model's `GPT-5.4` score (x-axis) against its `GPT-4o` score (y-axis). The two scores are strongly correlated ($r = 0.83$), and within-evaluator consistency across three random seeds is high (standard deviation < 0.03 for both evaluators), so the between-evaluator divergence reflects genuine differences in evaluation criteria rather than noise. Most points fall above the diagonal, indicating `GPT-5.4` is the stricter evaluator overall, with the largest drops on baselines that have complex string representations (EBM, FIGS large, PyGAM, RuleFit) and the largest evolved-model drop on `HingeEBM (v1)`; simple decision trees (DT mini, HSTree mini) lie below the diagonal. The drops are often attributable to `GPT-5.4` correctly abstaining from answering when the information is not present whereas `GPT-4o` would sometimes attempt to answer regardless, occasionally correctly answering some tests by chance.

Claude `Haiku-4.5` evaluator used post-hoc after models have been optimized. To check that the post-hoc results are not specific to a single evaluator family or to the exact prompt template used for evaluation, we additionally re-evaluate the same 41-model pool with Claude `Haiku-4.5` and a slightly *perturbed prompt* (re-worded preamble, explicit `MODEL:`/`QUESTION:` separators, and an instruction to use only the printed model). Fig. B1c-d show the result: again, AGENTIC-IMODELS expands the Pareto frontier, and Haiku scores correlate with `GPT-4o` at $r = 0.85$. This is comparable to the `GPT-5.4`-vs-`GPT-4o` correlation, although now most models lie below the diagonal.

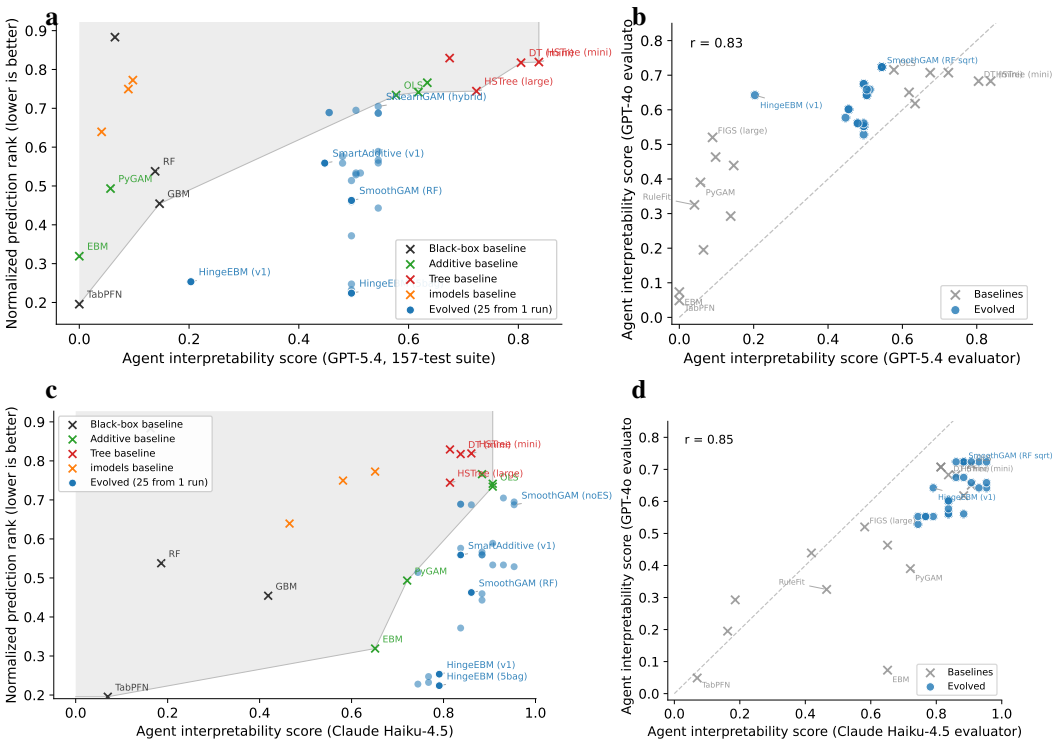

Figure B1: Post-hoc evaluator-sensitivity analysis on the held-out interpretability tests (41 baseline + evolved models from the Claude `Opus-4.6` medium-effort run). **Top row**: `GPT-5.4` evaluator (same prompt template as the in-loop `GPT-4o` evaluator). **Bottom row**: Claude `Haiku-4.5` evaluator with a slightly perturbed prompt. (a, c) Normalized prediction rank on the 65 original training datasets (y-axis matching Fig. 3a–c) versus agent interpretability score under each evaluator; baselines are colored crosses by category and evolved models are blue. The evolved Pareto frontier is preserved under both evaluators. (b, d) Per-model agent interpretability score under each held-out evaluator (x-axis) versus the original `GPT-4o` score (y-axis); baselines are gray, evolved models are blue. Scores correlate strongly with `GPT-4o` under both evaluators ($r = 0.83$ for `GPT-5.4`, $r = 0.85$ for Haiku), but with opposite biases: `GPT-5.4` is the stricter evaluator (most points above the diagonal), whereas Haiku is more lenient (most points below the diagonal).

`GPT-5.4` **evaluator used during the optimization loop.** As a stronger test of evaluator stability, we also run the full AGENTIC-IMODELS optimization loop with `GPT-5.4` (rather than `GPT-4o`) as the in-loop interpretability evaluator. We run with Claude Code `Opus-4.6`, medium reasoning effort, which yields 87 evolved models. Fig. B2 shows the models (and the 16 baselines) on the same rank versus interpretability axes as Fig. B1a, with interpretability measured on the held-out tests (also graded by `GPT-5.4`).

We find that the evolved models extend the Pareto frontier under the stricter evaluator: the best evolved models reach agent interpretability scores in the 0.71 to 0.73 range (e.g., `WinsorElasticNet` at normalized rank 0.09 and interp 0.73, `StepByStepAdaptive` at rank 0.07 and interp 0.71), above the best baseline (`OLS` at rank 0.09 / interp 0.69), while `MegaTeacher_a06` (rank 0.06 / interp 0.68) matches or improves on every additive baseline in both dimensions. The AGENTIC-IMODELS identified with the `GPT-5.4` evaluator differ from the `GPT-4o`-driven main runs: rather than hinge-based GAMs and additive boosted stumps, the `GPT-5.4` loop favors a mix of regularized linear models (`WinsorElasticNet`, `SparseBayesRidge`, `BayesianRidgeLinear`), per-dataset adaptive selectors (`AdaptiveElastic`, `QuantizedAdaptive`), and compact distilled trees (`MegaTeacher`, `TinyRecalTree`, `ElasticCompactTree`).

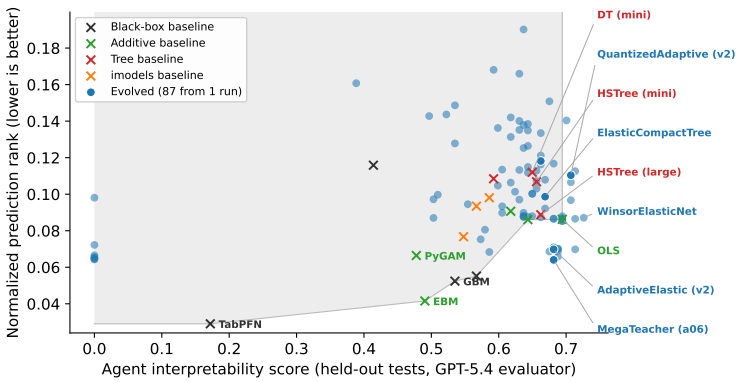

Figure B2: In-loop `GPT-5.4` optimization run: normalized prediction rank on the 65 training datasets versus agent interpretability score on the 157 held-out tests graded by `GPT-5.4`. The evolved models (blue) continue to improve over the Pareto frontier defined by the baselines (colored crosses).

B.2 Ablation: using held-out datasets to evaluate predictive performance

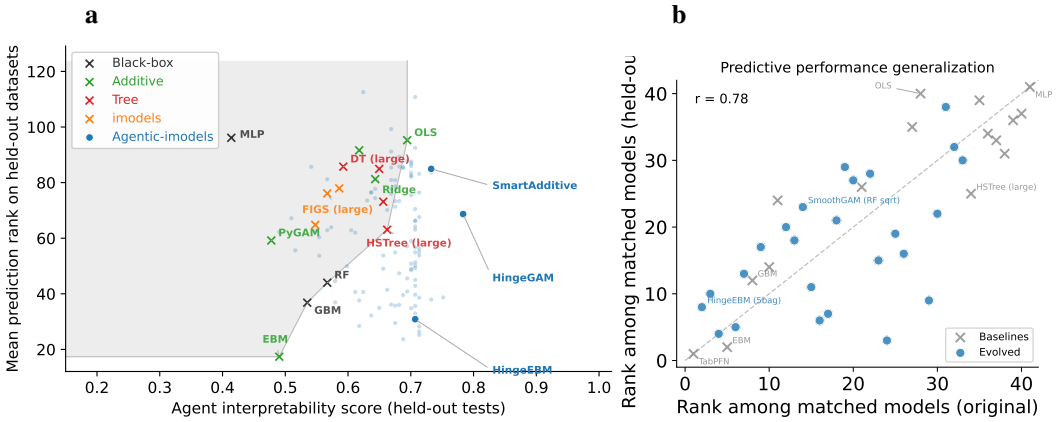

Figure B3: (a) Interpretability versus prediction rank on 16 held-out datasets and 157 new interpretability tests for the Claude `Opus-4.6` medium-effort main run (103 evolved models, blue) together with the 16 baselines. Ranks are computed globally across this 119-model pool. Evolved families such as hinge-based (HingeEBM, HingeGAM) and smooth additive (SmartAdditive) models continue to dominate the Pareto frontier; selected class labels correspond to the highlighted models in Fig. 1b. (b) Prediction ranks on the original versus held-out evaluation for the 41 matched models, re-ranked independently within each evaluation. Ranks correlate at $r = 0.78$; variance is higher than for the interpretability comparison (Fig. 3d) because the held-out datasets have different characteristics.

B.3 Ablation: hyperparameter-tuned baselines

The main-text comparison runs the baselines at their default hyperparameters, mirroring how ADS agents typically invoke `scikit-learn` estimators in practice. To check that the Pareto frontier is not an artifact of this choice, we additionally tune the strong baselines (GBM, RF, MLP, Ridge, EBM) per dataset with `RandomizedSearchCV` (20-iteration random search, 3-fold CV; a smaller budget for the much slower EBM) and re-rank all models in a shared 23-model / 28-dataset pool by mean RMSE rank (Table B1). Tuning improves the baselines' predictive rank substantially (e.g. GBM $10.9 \to 6.6$, RF $11.8 \to 9.2$, MLP $17.5 \to 15.0$; EBM is already near-optimal at its defaults, $8.1 \to 8.2$). The frontier claim nonetheless holds: (i) the single best predictor in the pool remains an evolved model (`HingeEBM`, mean rank 6.0), ahead of the best tuned baseline (GBM-tuned, 6.6); and (ii) tuning does not change a baseline's interpretability — a tuned GBM or RF is exactly as opaque to an agent as an untuned one (agent-interpretability $\approx 0.54/0.57$, vs. 0.71 for `HingeEBM`).

On the (interpretability $\times$ prediction) plane the evolved models therefore remain non-dominated, with `HingeEBM` Pareto-dominating GBM-tuned on *both* axes.

Table B1: **Tuned baselines.** Mean RMSE rank (lower is better) within a shared pool of 23 models across 28 PMLB regression datasets. Strong baselines are tuned per dataset with `RandomizedSearchCV` (20-iteration random search, 3-fold CV; a smaller budget for the slow EBM). Tuning markedly improves the baselines' predictive rank, but the best single predictor remains an evolved model (`HingeEBM`), and tuning does not change a baseline's interpretability, so the evolved models remain on the Pareto frontier.

Baseline	Default rank ↓	Tuned rank ↓
GBM	10.89	**6.57**
RF	11.82	9.18
EBM	8.11	8.21
MLP	17.46	14.96
Ridge	15.61	14.82
Best evolved model: `HingeEBM`, rank **6.00** (agent-interp 0.71)		
Best tuned baseline: GBM-tuned, rank 6.57 (agent-interp 0.54)		

B.4 Extended results on the BLADE end-to-end ADS analysis

Table B2 shows the per-dataset scores for the BLADE end-to-end evaluation.

Table B2: Per-dataset BLADE average scores breaking down the results in Fig. 4. Scores are averaged over the three rubric dimensions and error bars show the standard error over 9 evaluations per agent per condition. Std = standard interpretable tools, AGENTIC-IMODELS = with the agentic-imodels package. Bold indicates the higher score per agent when error bars do not overlap.

	Copilot (Gemini)		Copilot (Sonnet)		Claude Code		Codex CLI	
Dataset	Std	AGENTIC-IMODELS	Std	AGENTIC-IMODELS	Std	AGENTIC-IMODELS	Std	AGENTIC-IMODELS
affairs	2.9±0.2	**5.9**±0.8	4.4±0.1	**6.9**±0.5	5.4±0.5	6.7±0.9	7.7±0.5	**9.0**±0.2
amtl	4.4±0.2	**8.5**±0.2	6.9±0.6	**8.3**±0.2	6.4±0.5	**8.7**±0.0	8.6±0.2	8.7±0.1
boxes	3.9±0.3	**6.7**±0.8	4.1±0.4	**7.7**±0.4	4.2±0.4	**8.6**±0.1	8.7±0.1	**9.0**±0.1
caschools	5.0±0.2	**8.0**±0.2	7.9±0.2	7.7±0.4	7.7±0.4	8.1±0.2	7.5±0.3	**8.9**±0.1
crofoot	3.3±0.3	**7.3**±0.7	4.0±0.2	**5.6**±0.4	3.9±0.2	**6.5**±0.6	5.9±0.4	**8.3**±0.2
fertility	5.1±0.8	**7.0**±0.7	4.7±0.3	**8.6**±0.2	7.9±0.2	**8.5**±0.1	8.4±0.1	**8.8**±0.2
fish	4.2±0.3	**8.4**±0.1	5.4±0.2	**8.7**±0.1	5.3±0.1	**8.6**±0.1	8.4±0.2	**8.6**±0.1
hurricane	5.1±0.5	**8.6**±0.1	5.8±0.8	**8.5**±0.2	7.3±0.3	**8.7**±0.1	8.7±0.1	8.8±0.1
mortgage	4.6±0.3	**8.6**±0.1	5.3±0.7	**7.6**±0.3	5.9±0.3	**7.8**±0.4	7.2±0.5	8.0±0.4
panda_nuts	4.6±0.2	**8.6**±0.0	4.5±0.1	**8.3**±0.2	6.5±0.6	**7.8**±0.1	8.4±0.1	**8.8**±0.1
reading	3.7±0.2	**6.1**±0.6	4.6±0.5	**8.4**±0.2	4.6±0.3	**8.4**±0.1	8.7±0.1	**9.3**±0.1
soccer	4.1±0.4	**5.3**±0.7	5.0±0.1	**7.5**±0.5	6.9±0.3	**8.8**±0.1	8.4±0.1	8.6±0.1
teachingratings	5.8±0.5	**8.8**±0.1	7.3±0.4	**8.8**±0.1	8.0±0.2	**8.8**±0.1	8.5±0.1	**8.8**±0.1
Average	4.36±0.09	**7.52**±0.06	5.37±0.09	**7.90**±0.09	6.16±0.09	**8.15**±0.11	8.09±0.05	**8.73**±0.05

Judge independence. Because the main BLADE evaluation uses `GPT-4o` as the judge, we check that the reported gains are not an artifact of a single judge model by re-grading the run-1 Claude-Code agent outputs (13 datasets $\times$ {standard, evolved-library}) with a second, independent judge LLM under the same rubric. The two judges agree closely — Pearson $r = 0.84$ on the overall score (per-dimension $r = 0.82$–0.84; mean absolute difference 1.1 on the 0–10 scale) — and the evolved-library gain persists under the independent judge (average overall score $5.5 \to 7.4$, $+33\%$, vs. `GPT-4o`'s $+27\%$), improving $12/13$ datasets. This mirrors the interpretability-evaluator swap (Fig. B1), where Claude `Haiku-4.5` and `GPT-4o` correlate at $r = 0.85$.

B.5 Token consumption

Table B3 summarizes approximate token consumption per AGENTIC-IMODELS run. We split the agent column into *Fresh* tokens (input the model processes for the first time, plus cache-creation writes and outputs) and *Cache reads* (the same conversational context being re-read from the prompt cache on each follow-up turn): cache reads dominate the raw byte count of an agentic loop but are billed at roughly $10\times$ less per token than fresh input on both Anthropic and OpenAI APIs, so the two

columns are not directly comparable as "compute consumed." Interpretability-test token counts are estimated by multiplying the number of `GPT-4o` calls (one row per test per candidate model) by an average of 1,050 tokens per call (prompt plus short response).

Table B3: Approximate token consumption per AGENTIC-IMODELS run, broken down by what the model actually processed for the first time (*Fresh*) versus what was re-read from the prompt cache on follow-up turns (*Cache reads*). *Fresh* sums fresh input, cache-creation, and output tokens (Anthropic) or non-cached input plus output (OpenAI); *Cache reads* is the cache-replay traffic, which is billed at ≈10× less per token and so dominates the raw token totals without representing comparable compute. Interpretability-test tokens are estimated assuming ≈1,050 tokens per `GPT-4o` call (prompt plus response). For the `GPT-5.4` evaluator run, we double the per-call estimate to ≈2,100 tokens to account for the larger reasoning traces produced by `GPT-5.4`.

Run	Models tried	Fresh (M)	Cache reads (M)	Interp. calls	Interp. tokens (M)	Total (M)
Claude `Opus-4.6` (medium)	165	7.4	749.7	7,611	8.0	765.1
Claude `Opus-4.7` (medium, run 1)	46	0.8	34.8	2,666	2.8	38.4
Claude `Opus-4.7` (medium, run 2)	50	0.5	29.3	2,795	2.9	32.7
Claude `Opus-4.7` (medium, run 3)	64	1.2	54.2	3,440	3.6	59.0
Claude `Opus-4.6` (high)	87	1.2	185.4	4,214	4.4	191.0
Claude `Opus-4.7` (xhigh)	65	4.1	212.2	3,569	3.7	220.0
Claude `Opus-4.6` (medium, `GPT-5.4` evaluator)	89	7.6	803.7	4,515	9.4	820.7
Codex `GPT-5.3` (default)	51	4.3	43.3	2,881	3.0	50.6
Codex `GPT-5.3` (high)	42	27.1	343.5	2,494	2.6	373.2
Codex `GPT-5.3` (xhigh)	57	14.9	389.8	3,914	4.1	408.8
Total	716	**69.1**	2845.9	38,099	44.5	2959.5

C Descriptions of selected evolved models

Table C1 summarizes the 10 evolved models highlighted in Fig. 1b. Below, we describe in detail the three of these model classes that exemplify the two architectural patterns identified in Sec. 4.4. All models are `scikit-learn`-compatible and available in the released library. To illustrate each model's `__str__` output concretely, we fit each on the same synthetic dataset: $y = 2x_0 + 0.5x_1 + f(x_2) + \varepsilon$, where $f(x_2) = 1.5x_2$ for $x_2 > 0$ and $0.3x_2$ otherwise, x_3 is irrelevant, and $\varepsilon \sim \mathcal{N}(0, 0.3)$, with $n = 300$.

C.1 Describing the `HingeEBM_5bag` model

Architecture. `HingeEBM_5bag` fits a two-stage model. In stage 1, for each feature j, it constructs positive hinge terms $\max(0, x_j - t)$ and negative hinge terms $\max(0, t - x_j)$ at $K = 2$ quantile knots, yielding a piecewise-linear basis expansion. Lasso selects a sparse subset, producing the stage-1 prediction:

$$\hat{y}^{(1)} = \beta_0 + \sum_j \beta_j x_j + \sum_{j,k} \left[\alpha^+_{jk} \max(0,\, x_j - t_{jk}) + \alpha^-_{jk} \max(0,\, t_{jk} - x_j) \right]. \tag{1}$$

In stage 2, if the residuals $y - \hat{y}^{(1)}$ explain more than 10% of remaining variance, an EBM with 5 outer bags and 1,000 rounds is fitted on the residuals. The final prediction is $\hat{y} = \hat{y}^{(1)} + \hat{y}^{(2)}_{\text{EBM}}$, but `__str__` displays only the stage-1 component.

Display logic. Rather than showing the hinge equation (1) directly, `__str__` collapses each feature's hinge contributions into an effective linear slope. For a positive hinge $\alpha^+ \max(0, x_j - t)$, the slope α^+ is added to the coefficient of x_j and $-\alpha^+ t$ is added to the intercept. Negative hinges are handled symmetrically. The result is a compact linear equation that is easy for an LLM to simulate, though it is only exact above the highest active knot.

Example output.

Table C1: The 10 evolved models highlighted in Fig. 1b. *Source* indicates which agent (Claude Code or Codex) discovered the model. *Rank* is the normalized mean RMSE rank across the 65 development datasets in the global pool of evolved + baseline models in Fig. 1b (lower is better). *Interp.* is the held-out interpretability score (fraction of 157 tests passed).

Model	Source	Rank ↓	Interp. ↑	Description
`HingeEBM`	Claude	0.19	0.71	LassoCV on a hinge (piecewise-linear) basis; hidden EBM corrects residuals (Sec. C.1).
`DistilledTreeBlendAtlas`	Codex	0.25	0.71	Ridge student distilled from GBM/RF teachers with validation-calibrated nonnegative blending.
`DualPathSparseSymbolic`	Codex	0.29	0.71	Blended GBM/RF/Ridge teacher for batch prediction; sparse symbolic single-row equation for display.
`HybridGAM`	Claude	0.29	0.68	SmartAdditive display + hidden depth-5 random-forest residual, shrunk by 0.7 (Sec. C.3).
`TeacherStudentRuleSpline`	Codex	0.36	0.80	GBM teacher for batch path; sparse symbolic student over linear/spline/rule/gated terms for display.
`SparseSignedBasisPursuit`	Codex	0.48	0.76	Forward-selected signed basis (linear, hinge, square, interaction) with ridge refit and light coefficient rounding.
`HingeGAM`	Claude	0.49	0.78	Lasso on a hinge basis at 10 quantile knots, with adaptive SmartAdditive-style display.
`WinsorizedSparseOLS`	Claude	0.58	0.73	LassoCV-selected top-8 features after clipping inputs to the $[p_1, p_{99}]$ range; OLS refit.
`TinyDTDepth2`	Claude	0.59	0.71	Depth-2 (4-leaf) decision tree.
`SmartAdditive`	Claude	0.62	0.73	Greedy boosted-stump GAM with adaptive per-feature linearization (Sec. C.2).

```
Ridge Regression (L2 regularization, alpha=0.01043 chosen by CV):
  y = 1.9688*x0 + 0.5194*x1 + 1.5351*x2 + -0.1190

Coefficients:
  x0: 1.9688
  x2: 1.5351
  x1: 0.5194
  intercept: -0.1190
  Features with zero coefficients (excluded): x3
```

The display correctly recovers x_0 and x_1 but shows a single slope for x_2 (1.54), which approximates the average effect of x_2's piecewise-linear shape. The hidden EBM captures the threshold nonlinearity.

C.2 Describing the `SmoothAdditiveGAM` model

Architecture. `SmoothAdditiveGAM` (`SmartAdditiveRegressor`, variant `smooth_gam_v1`) fits a greedy additive boosted-stump model. In each of 200 rounds, it selects the feature j and threshold τ that most reduces residual SSE via a depth-1 tree, updates the per-feature shape function f_j, and subtracts the fitted step. After boosting, 3 passes of Laplacian smoothing (weights 0.6/0.2/0.2) are applied to each f_j. The prediction is:

$$\hat{y} = \mu + \sum_j f_j(x_j), \tag{2}$$

where each f_j is a piecewise-constant function over the thresholds chosen during boosting.

Display logic. For each feature j, a linear approximation $\hat{f}_j(x_j) \approx s_j x_j + b_j$ is computed on training data. If $R^2 > 0.90$ and the feature contributes at least 1% of total importance, it is rendered as a coefficient s_j; otherwise the full piecewise table is shown. The Laplacian smoothing step is critical: it flattens irregular shape functions, increasing the fraction of features that pass the R^2 threshold and thus appear as simple linear terms. (A later variant, `smooth_gam_r70hyb`, lowers this threshold to $R^2 > 0.70$ and adds a hybrid predict step; the core architecture is the same.)

Example output.

```
Ridge Regression (L2 regularization, alpha=1.0000 chosen by CV):
  y = 1.8974*x0 + 0.3974*x1 + 0.5368

Coefficients:
  x0: 1.8974
  x1: 0.3974
  intercept: 0.5368

Nonlinear feature effects (piecewise corrections to add to above):
  f(x2):
    x2 <= -1.3747: -0.9201
    -1.3747 < x2 <= -0.9901: -0.7977
    -0.9901 < x2 <= -0.5740: -0.7116
    -0.5740 < x2 <= -0.3322: -0.6463
    ...
    x2 >  2.2444: +2.8050
```

The display correctly identifies x_0 and x_1 as linear (recovering coefficients close to the ground truth of 2.0 and 0.5), and shows x_2 as a piecewise table that captures the slope change at 0.

C.3 Describing the `RidgeRFResid` model

To illustrate that display–predict decoupling was discovered independently across runs, we briefly describe `RidgeRFResid`. Its architecture is analogous to `HingeEBM` but uses a different backbone and corrector:

$$\hat{y} = \underbrace{\hat{y}_{\text{Ridge}}(x)}_{\text{displayed}} + \lambda \cdot \underbrace{\hat{y}_{\text{RF}}(x)}_{\text{hidden residual}} , \tag{3}$$

where $\hat{y}_{\text{Ridge}}$ is a Ridge regression model shown in `__str__` and a 100-tree depth-5 RandomForest fitted on residuals contributes to `predict` with shrinkage $\lambda = 0.6$. This model achieved mean rank 6.6 with 67% interpretability (within two rank positions of `HingeEBM_5bag`) while displaying only a linear equation. The same pattern re-emerged as the `HybridGAM` family in subsequent Claude runs, which used a SmartAdditiveGAM display instead of Ridge.

C.4 Display-vs-predictor faithfulness of the library

The example models above illustrate the two architectural patterns qualitatively; here we quantify, across the whole library, how far each model's displayed equation diverges from its deployed predictor. For each of the 10 models we reconstruct predictions by executing the equation/rules printed in `__str__` alone and compare them against the model's actual `.predict()` on held-out points (pooled over 9 real regression datasets, with the outcome standardized so $\text{SD} = 1$; Table C2). Faithful ("honest") models reconstruct `.predict()` almost exactly (R^2 up to 1.00), whereas display–predict-decoupled models diverge in proportion to their hidden corrector (R^2 as low as -1.19 for `HingeEBM`, mean absolute error ≈ 1 SD). This confirms that display *simulatability* and display *faithfulness* are separable properties, and we now report both; it also surfaces one correction, that `HingeGAM`'s printed linearization of its hinge basis makes it only partially faithful despite having no hidden corrector.

Table C2: **Display-vs-predictor faithfulness.** For each library model we execute the equation/rules printed in `__str__` and compare the reconstruction to the model's actual `.predict()` on held-out points (pooled over 9 real datasets, outcome standardized so SD=1). "Honest" models are (near-)exact; display–predict decoupled models diverge by the magnitude of their hidden corrector.

Model	Design	R^2(disp→pred)	MAE/SD	% within 0.20 SD
SparseSignedBasisPursuit	honest	1.00	0.00	100
SmartAdditive	honest	0.99	0.03	97
TinyDT	honest	0.94	0.16	82
WinsorizedSparseOLS	honest	0.76	0.08	91
HingeGAM	honest*	0.31	0.52	28
HybridGAM	decoupled	0.94	0.14	77
DistilledTreeBlendAtlas	decoupled	0.76	0.30	55
TeacherStudentRuleSpline	decoupled	0.58	0.38	48
HingeEBM	decoupled	−1.19	0.98	45
DualPathSparseSymbolic	decoupled	−9.06	1.91	34

*`HingeGAM`'s printed equation linearizes its hinge basis, so its display is only partially faithful despite having no hidden corrector.

NeurIPS Paper Checklist

1. **Claims**

   Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

   Answer: [Yes]

   Justification: The abstract and introduction state our two main contributions—an LLM-graded interpretability metric and an agentic loop that uses it to discover new interpretable models that improve the interpretability-performance Pareto frontier and downstream end-to-end ADS performance—each of which is supported by experiments in Sec. 4.2 and Sec. 4.3.

   Guidelines:

   - The answer [N/A] means that the abstract and introduction do not include the claims made in the paper.
   - The abstract and/or introduction should clearly state the claims made, including the contributions made in the paper and important assumptions and limitations. A [No] or [N/A] answer to this question will not be perceived well by the reviewers.
   - The claims made should match theoretical and experimental results, and reflect how much the results can be expected to generalize to other settings.
   - It is fine to include aspirational goals as motivation as long as it is clear that these goals are not attained by the paper.

2. **Limitations**

   Question: Does the paper discuss the limitations of the work performed by the authors?

   Answer: [Yes]

   Justification: The Discussion section includes an explicit "Limitations" paragraph that discusses LLM-as-judge bias, reward hacking, the gap between LLM-graded and human-graded interpretability, the limited test coverage, and the cost of the agent loop.

   Guidelines:

   - The answer [N/A] means that the paper has no limitation while the answer [No] means that the paper has limitations, but those are not discussed in the paper.
   - The authors are encouraged to create a separate "Limitations" section in their paper.
   - The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.

- The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.
- The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.
- The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
- If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.
- While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. The authors should use their best judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

3. **Theory assumptions and proofs**

   Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

   Answer: [N/A]

   Justification: The paper is entirely empirical and does not include any formal theoretical results, theorems, or proofs.

   Guidelines:

   - The answer [N/A] means that the paper does not include theoretical results.
   - All the theorems, formulas, and proofs in the paper should be numbered and cross-referenced.
   - All assumptions should be clearly stated or referenced in the statement of any theorems.
   - The proofs can either appear in the main paper or the supplemental material, but if they appear in the supplemental material, the authors are encouraged to provide a short proof sketch to provide intuition.
   - Inversely, any informal proof provided in the core of the paper should be complemented by formal proofs provided in appendix or supplemental material.
   - Theorems and Lemmas that the proof relies upon should be properly referenced.

4. **Experimental result reproducibility**

   Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

   Answer: [Yes]

   Justification: Sec. 3 and Sec. 4.1 document datasets, preprocessing, baselines, agent versions and reasoning effort levels, prompts, and evaluation protocol; the appendix lists all 200 interpretability tests, all datasets, and ablations; and full code (including prompts) is released at the GitHub link in the abstract.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - If the paper includes experiments, a [No] answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.
   - If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.

- Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may be necessary to either make it possible for others to replicate the model with the same dataset, or provide access to the model. In general. releasing code and data is often one good way to accomplish this, but reproducibility can also be provided via detailed instructions for how to replicate the results, access to a hosted model (e.g., in the case of a large language model), releasing of a model checkpoint, or other means that are appropriate to the research performed.
- While NeurIPS does not require releasing code, the conference does require all submissions to provide some reasonable avenue for reproducibility, which may depend on the nature of the contribution. For example
  (a) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.
  (b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.
  (c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).
  (d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

5. **Open access to data and code**

   Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

   Answer: [Yes]

   Justification: All code, the evolved `agentic_imodels` package, prompts, and evaluation scripts are released at the Github link in the abstract; all datasets used (PMLB, OpenML TabArena, BLADE) are publicly available.

   Guidelines:

   - The answer [N/A] means that paper does not include experiments requiring code.
   - Please see the NeurIPS code and data submission guidelines (`https://neurips.cc/public/guides/CodeSubmissionPolicy`) for more details.
   - While we encourage the release of code and data, we understand that this might not be possible, so [No] is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution (e.g., for a new open-source benchmark).
   - The instructions should contain the exact command and environment needed to run to reproduce the results. See the NeurIPS code and data submission guidelines (`https://neurips.cc/public/guides/CodeSubmissionPolicy`) for more details.
   - The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.
   - The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.
   - At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).
   - Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

6. **Experimental setting/details**

Question: Does the paper specify all the training and test details (e.g., data splits, hyperparameters, how they were chosen, type of optimizer) necessary to understand the results?

Answer: [Yes]

Justification: Sec. 4.1 specifies the train/test split (80/20), subsampling caps, normalization, encoding, baseline hyperparameters, agent versions, reasoning effort levels, and the BLADE end-to-end protocol; further dataset and test details are in the appendix.

Guidelines:

- The answer [N/A] means that the paper does not include experiments.
- The experimental setting should be presented in the core of the paper to a level of detail that is necessary to appreciate the results and make sense of them.
- The full details can be provided either with the code, in appendix, or as supplemental material.

7. **Experiment statistical significance**

Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

Answer: [Yes]

Justification: Fig. 4 reports standard error of the mean across 9 evaluations per dataset per condition (3 agent runs $\times$ 3 judge runs); Fig. 3c quantifies run-to-run variability across random repetitions, and held-out versus development correlations are reported with sample sizes.

Guidelines:

- The answer [N/A] means that the paper does not include experiments.
- The authors should answer [Yes] if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.
- The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).
- The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)
- The assumptions made should be given (e.g., Normally distributed errors).
- It should be clear whether the error bar is the standard deviation or the standard error of the mean.
- It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
- For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g., negative error rates).
- If error bars are reported in tables or plots, the authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. **Experiments compute resources**

Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

Answer: [Yes]

Justification: Sec. 3.3 reports total token usage ($\approx$70M tokens) with a per-run breakdown in Table B3, an estimate of replication cost (3 months of a $100 Claude Max subscription), and notes that models can be trained on a 40GB GPU.

Guidelines:

- The answer [N/A] means that the paper does not include experiments.

- The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.
- The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.
- The paper should disclose whether the full research project required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn't make it into the paper).

9. **Code of ethics**

   Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics `https://neurips.cc/public/EthicsGuidelines`?

   Answer: [Yes]

   Justification: The research uses publicly available tabular datasets and benchmarks, releases code openly, and involves no human subjects, sensitive data, or deployments that could violate the NeurIPS Code of Ethics.

   Guidelines:

   - The answer [N/A] means that the authors have not reviewed the NeurIPS Code of Ethics.
   - If the authors answer [No], they should explain the special circumstances that require a deviation from the Code of Ethics.
   - The authors should make sure to preserve anonymity (e.g., if there is a special consideration due to laws or regulations in their jurisdiction).

10. **Broader impacts**

    Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

    Answer: [Yes]

    Justification: The Discussion motivates the work in terms of agents conducting reliable data analysis (a positive impact on scalable oversight and scientific discovery), while the Limitations paragraph explicitly discusses risks such as reward hacking and the gap between LLM-graded and human-interpretable representations.

    Guidelines:

    - The answer [N/A] means that there is no societal impact of the work performed.
    - If the authors answer [N/A] or [No], they should explain why their work has no societal impact or why the paper does not address societal impact.
    - Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.
    - The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate Deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.
    - The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
    - If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. **Safeguards**

Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pre-trained language models, image generators, or scraped datasets)?

Answer: [N/A]

Justification: The released artifacts are small `scikit-learn`-compatible regressors for tabular data, which do not pose a high risk of misuse and require no special safeguards.

Guidelines:

- The answer [N/A] means that the paper poses no such risks.
- Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.
- Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.
- We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

12. **Licenses for existing assets**

Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

Answer: [Yes]

Justification: All datasets (PMLB, OpenML TabArena, BLADE, Grinsztajn benchmark), packages (`scikit-learn`, `imodels`, `interpretML`, TabPFN), and agents (Claude Code, Codex, Copilot CLI) are cited in Sec. 4.1 and the appendix; each is publicly available and used in accordance with its license.

Guidelines:

- The answer [N/A] means that the paper does not use existing assets.
- The authors should cite the original paper that produced the code package or dataset.
- The authors should state which version of the asset is used and, if possible, include a URL.
- The name of the license (e.g., CC-BY 4.0) should be included for each asset.
- For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.
- If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, `paperswithcode.com/datasets` has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.
- For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.
- If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. **New assets**

Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

Answer: [Yes]

Justification: We release the `agentic_imodels` package (10 evolved regressors with a documented `SKILL.md` and API), the interpretability test suite, and all evaluation code at the Github url; Sec. C additionally documents example models and their `__str__` outputs.

Guidelines:

- The answer [N/A] means that the paper does not release new assets.

- Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.
- The paper should discuss whether and how consent was obtained from people whose asset is used.
- At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

14. **Crowdsourcing and research with human subjects**

    Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

    Answer: [N/A]

    Justification: The paper does not involve any crowdsourcing or research with human subjects; all interpretability evaluations are performed by LLMs and all underlying datasets are pre-existing public benchmarks.

    Guidelines:

    - The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
    - Including this information in the supplemental material is fine, but if the main contribution of the paper involves human subjects, then as much detail as possible should be included in the main paper.
    - According to the NeurIPS Code of Ethics, workers involved in data collection, curation, or other labor should be paid at least the minimum wage in the country of the data collector.

15. **Institutional review board (IRB) approvals or equivalent for research with human subjects**

    Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

    Answer: [N/A]

    Justification: The paper does not involve research with human subjects, so no IRB approval was required.

    Guidelines:

    - The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
    - Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.
    - We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
    - For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.

16. **Declaration of LLM usage**

    Question: Does the paper describe the usage of LLMs if it is an important, original, or non-standard component of the core methods in this research? Note that if the LLM is used only for writing, editing, or formatting purposes and does *not* impact the core methodology, scientific rigor, or originality of the research, declaration is not required.

    Answer: [Yes]

    Justification: LLMs are central to the methodology and are extensively documented: coding agents (Claude Code, Codex) drive the AGENTIC-IMODELS optimization loop (Sec. 3.3),

an LLM (`GPT-4o`, with `GPT-5.4` and Claude `Haiku-4.5` ablations) grades interpretability tests (Sec. 3.2), and three ADS agents are used in the BLADE evaluation (Sec. 4.3); versions, reasoning effort, and prompts are reported in Sec. 4.1.

Guidelines:

- The answer [N/A] means that the core method development in this research does not involve LLMs as any important, original, or non-standard components.
- Please refer to our LLM policy in the NeurIPS handbook for what should or should not be described.