Research Square

Preprints are preliminary reports that have not undergone peer review.
They should not be considered conclusive, used to inform clinical practice,
or referenced by the media as validated information.

What Makes a Programming Problem Hard for a Language Model? An Empirical Study of Item Difficulty Across Code LLMs on Two Benchmarks

TANZIM ISLAM KHAN

dihan2468@gmail.com

Independent Researcher

Research Article

Keywords: code generation, large language models, benchmarking, item response theory, problem difficulty, predictive modeling, data-centric evaluation, computing education

Posted Date: July 13th, 2026

DOI: https://doi.org/10.21203/rs.3.rs-10241058/v1

License: This work is licensed under a Creative Commons Attribution 4.0 International License.
Read Full License

Additional Declarations: No competing interests reported.

What Makes a Programming Problem Hard for a Language Model?

An Empirical Study of Item Difficulty Across Code LLMs on Two Benchmarks

Tanzim Islam Khan

Independent Researcher, Melbourne, Australia

Corresponding author: dihan2468@gmail.com

ORCID: 0009-0008-0385-4681

Abstract

Large language models (LLMs) for code are now evaluated with benchmarks such as HumanEval and MBPP, and they are increasingly used to generate, grade, and explain exercises in introductory programming courses. Yet the standard reporting practice, a single aggregate `pass@k` score per model, treats every problem as interchangeable and tells us nothing about *which* problems are hard or *why*. We present a data-driven empirical study of per-problem difficulty for code generation across two benchmarks. Using the publicly released greedy completions of code LLMs that span 1B parameters to GPT-4 scale (31 models on HumanEval, 22 on MBPP), we re-execute every completion against the benchmark tests in a sandboxed harness, 13,400 executions, and derive an empirical difficulty for each problem as the cross-model failure rate. We confirm that this signal behaves as a genuine item-difficulty parameter on both benchmarks: a one-parameter logistic (Rasch) model that controls for model ability reproduces the same ordering almost exactly (Spearman $\rho \approx 1.00$ on HumanEval), the ordering is essentially unchanged when any single model is removed (leave-one-model-out $\rho \geq 0.99$ on both), and it transfers across capability tiers ($\rho = 0.78$ and 0.79). We then ask what makes a problem hard. We extract a broad set of features describing the natural-language prompt, the reference solution, and the test suite, and find that on HumanEval the strongest predictors are properties of the *specification*, not of the algorithm: the number of worked input-output examples and the presence of type hints make problems easier, while longer prompts make them harder. Prompt features *alone* predict difficulty with cross-validated $R^2 = 0.45$, matching the full feature set (0.46), whereas reference-solution complexity features alone reach only 0.11. On MBPP, whose prompts are uniformly terse and contain no worked examples, the picture inverts: solution-complexity features carry the signal instead. The lesson is that what makes a problem hard depends on what varies in how it is specified, and both this conclusion and the difficulty ranking survive re-grading against the hardened HumanEval+ and MBPP+ suites (80 and 35 times more tests) and a switch from greedy decoding to temperature sampling. Finally, we place these per-problem panels in the context of the present-day (June 2026) model landscape, from the 125-model EvalPlus leaderboard up to the current frontier (GPT-5.x, Claude Opus 4.8, Gemini 3.1 Pro), which has saturated and abandoned HumanEval for repository-level benchmarks, showing that item difficulty becomes more important, not less, as aggregate scores compress. We discuss implications for difficulty-aware benchmarking, autograder design, and the use of LLM-generated exercises in computing education, and we release all data, code, and the response matrices.

Keywords: code generation; large language models; benchmarking; item response theory; problem difficulty; predictive modeling; data-centric evaluation; computing education

1 Introduction

Code-generating large language models (LLMs) have moved from research prototypes to everyday tools for software engineers and for students learning to program. Their progress is tracked almost entirely through functional-correctness benchmarks, of which HumanEval (Chen et al., 2021) and MBPP (Austin et al., 2021) are the most widely used. The headline metric is `pass@k`, the probability that at least one of k sampled programs passes a hidden test suite. This single number is what populates leaderboards, what model reports advertise, and what practitioners use to choose a model.

A scalar score per model is convenient, but it hides structure. It implicitly assumes that all problems contribute equally to model quality and that the only interesting axis of variation is the model. In educational and software-testing settings, the opposite assumption is often more useful: the items, not the test takers, are the object of study. Classical psychometrics has long separated *ability*, a property of the test taker, from *difficulty*, a property of the item (Rasch, 1960; Baker, 2001), and recent work in natural language processing has argued that the same separation should reshape how we read leaderboards (Lalor et al., 2016; Rodriguez et al., 2021; Vania et al., 2021). Viewed this way, benchmark evaluation becomes a data-analysis problem in its own right: the raw material is a matrix of per-item outcomes across models, and the task is to extract a latent structure from that matrix, validate that the structure is stable, and explain it from observable features of the items. For code LLMs, the analogous questions have received much less attention: How is difficulty distributed across the problems in a benchmark? Is difficulty a stable property of a problem, or an artifact of which models happen to be on the leaderboard? And, most importantly for both benchmark designers and educators, what makes one programming problem harder than another?

These questions matter beyond leaderboard hygiene. In computing education, instructors are now using LLMs to draft exercises, to generate worked solutions, and to provide automated feedback (Finnie-Ansley et al., 2022; Becker et al., 2023; Sarsa et al., 2022; Denny et al., 2024; Prather et al., 2023). If we can characterize which specifications an LLM is likely to handle well, we can build difficulty-aware autograders, flag exercises that an assistant is likely to get wrong, and design prompts that scaffold both the model and the learner. A predictive account of difficulty is also a lever for benchmark construction: it lets us audit whether a benchmark over-samples trivial problems, and it offers cheap, static proxies for difficulty that do not require running a fleet of models.

This paper provides an empirical answer grounded in real model behavior rather than in synthetic estimates. Our study rests on a simple but, to our knowledge, under-used resource: the actual completions that many code LLMs produced on HumanEval and MBPP, released publicly by the EvalPlus project (Liu et al., 2023). We assemble panels that range from 1B-parameter base models to GPT-4 (OpenAI, 2023) (31 models on HumanEval, 22 on MBPP), re-execute every greedy completion against the benchmark tests in a sandboxed harness, and treat the per-problem cross-model failure rate as an empirical difficulty label. We then extract a broad set of static features and study, with standard statistical and machine-learning tools, what predicts difficulty, and whether the answer

generalizes from one benchmark to the other. To connect the analysis to the current state of the field, we additionally use the present-day (June 2026) EvalPlus leaderboard of 125 models, which includes systems such as GPT-4o, the o1 series, DeepSeek-V3, and Qwen2.5-Coder, as an aggregate reference point. We further document that the very latest frontier, including GPT-5.x, Claude Opus 4.7 and 4.8, and Gemini 3.1 Pro, has abandoned HumanEval entirely in favor of harder, repository-level benchmarks such as SWE-bench (Jimenez et al., 2024), which underlines rather than undercuts the case for studying item difficulty.

Research questions. We organize the study around four questions.

- **RQ1 (Difficulty as a measurable quantity).** How is per-problem difficulty distributed across HumanEval, and does it behave like a genuine item-difficulty parameter when model ability is controlled for?
- **RQ2 (Correlates).** Which static features of the prompt, the reference solution, and the test suite are associated with difficulty?
- **RQ3 (Prediction).** Can difficulty be predicted from static features, and which family of features carries the signal?
- **RQ4 (Robustness and generalization).** Is the difficulty ordering stable across the choice of models, across capability tiers, and does the predictive account transfer between HumanEval and MBPP?

Contributions.

1. A reproducible execution study that turns the public completions of 31 models on HumanEval and 22 on MBPP into binary response matrices and validated per-problem difficulty scales for both benchmarks, including a Rasch ability-difficulty decomposition, graded against both the base and the hardened (HumanEval+, MBPP+) test suites, plus a temperature-sampling robustness check, for roughly 68,000 executions in total.
2. Evidence that difficulty is a stable, model-agnostic property of a problem on both benchmarks: it survives leave-one-model-out perturbation ($\rho \geq 0.99$), bootstrap resampling of the panel, and transfer between weaker and stronger models ($\rho = 0.78$ on HumanEval, 0.79 on MBPP).
3. A predictive analysis showing that difficulty is predictable (HumanEval cross-validated $R^2 = 0.46$), together with a cross-benchmark result on *where* the signal lives: on HumanEval it is the natural-language specification (prompt features alone match the full model, $R^2 = 0.45$, while solution complexity alone explains little), whereas on MBPP, whose prompts are uniformly terse, solution-complexity features carry the signal. What makes a problem hard depends on what varies in how it is specified.
4. A present-day (June 2026) contextualization using the 125-model EvalPlus leaderboard, quantifying HumanEval saturation and test-suite leniency, alongside a public replication package.

All artifacts, including the response matrix, extracted features, analysis scripts, and figures, are released for replication (Section 8).

2 Background and Related Work

2.1 Benchmarks for code generation

HumanEval (Chen et al., 2021) introduced 164 hand-written Python problems, each consisting of a function signature, a docstring with worked examples, and a hidden unit-test suite, scored with `pass@k`. MBPP (Austin et al., 2021) followed with shorter crowd-sourced tasks, and later benchmarks broadened the setting to competition problems (Hendrycks et al., 2021; Li et al., 2022), class-level code (Du et al., 2024), complex library use (Zhuo et al., 2024), and contamination-controlled evaluation (Jain et al., 2024). A recurring methodological concern is that the test suites shipped with these benchmarks may be too weak, so that incorrect programs pass; EvalPlus (Liu et al., 2023) augments HumanEval and MBPP with far larger automatically generated test suites and, importantly for us, publicly releases the raw program samples produced by dozens of models. We reuse those samples as the empirical substrate of our study. Almost all of this literature reports aggregate per-model scores. We instead treat the problem as the unit of analysis.

2.2 Item response theory for model evaluation

Item response theory (IRT) models the probability that a test taker answers an item correctly as a function of the taker's latent ability and the item's latent difficulty (Rasch, 1960; Baker, 2001). The NLP community has adopted IRT to build evaluation scales (Lalor et al., 2016), to argue that test examples are not equally informative (Rodriguez et al., 2021), and to compare test sets (Vania et al., 2021). We bring this lens to code generation, using a one-parameter logistic (Rasch) model to validate that cross-model solve rate is a sound difficulty estimate, and we then go further by asking what *observable* features explain the latent difficulty.

2.3 Software complexity metrics

Software engineering has a long tradition of quantifying code complexity, including McCabe's cyclomatic complexity (McCabe, 1976) and Halstead's software-science measures (Halstead, 1977). These metrics describe the artifact a programmer writes. A natural hypothesis is that problems whose reference solutions are more complex are harder for an LLM. We test this hypothesis directly on two benchmarks and find that it holds only when the specification is held roughly constant: on MBPP, whose prompts are uniformly terse, solution complexity is the dominant signal, but on HumanEval, whose prompts vary widely in richness, specification features dominate solution-complexity features.

2.4 LLMs and computing education

Code LLMs solve a large fraction of typical introductory assignments (Finnie-Ansley et al., 2022), which has prompted active discussion of how computing education should adapt (Becker et al., 2023; Prather et al., 2023; Denny et al., 2024). LLMs are also used constructively, for example to generate programming exercises and explanations (Sarsa et al., 2022). A predictive account of which specifications a model handles well is directly relevant: it informs which exercises an instructor can safely delegate to an assistant and which require human review. Our finding that

worked examples are the single strongest predictor of easiness connects to the worked-example effect in the cognitive-load literature (Sweller, 1988; Atkinson et al., 2000), suggesting that the same scaffolding that helps novices also helps models.

3 Study Design

3.1 Benchmarks and difficulty target

We study two widely used benchmarks. HumanEval (Chen et al., 2021) contributes 164 problems, each with a prompt (signature plus docstring with worked examples), a canonical reference solution, an entry-point function name, and a canonical test function `check`. MBPP (Austin et al., 2021) contributes the 378 problems of the human-verified EvalPlus subset (Liu et al., 2023), each with a one-line natural-language description, a canonical solution, and a list of assertion tests. The two benchmarks deliberately differ in specification style: HumanEval prompts are rich and example-laden, while MBPP prompts are terse and example-free, which lets us test whether the determinants of difficulty generalize. A candidate program is scored *correct* if and only if it passes the benchmark's tests when executed, with a time limit to bound non-terminating programs.

3.2 Model panel

We use the temperature-0 (greedy, deterministic) HumanEval completions released by EvalPlus (Liu et al., 2023) for 31 distinct models. The panel is deliberately heterogeneous and spans the families CodeGen and CodeGen2 (Nijkamp et al., 2023), CodeT5+ (Wang et al., 2023), Code Llama and its Phind fine-tune (Rozière et al., 2023), StarCoder and StarCoderBase (Li et al., 2023), SantaCoder (Allal et al., 2023), InCoder (Fried et al., 2023), WizardCoder (Luo et al., 2023), Mistral (Jiang et al., 2023), Vicuna (Zheng et al., 2023), and the GPT family including GPT-3.5 and GPT-4 (OpenAI, 2023), alongside PolyCoder, GPT-J, GPT-Neo, and StableLM. Sizes range from roughly 1B parameters to GPT-4 scale, and the panel mixes base and instruction-tuned models. Table 1 lists every model with its family, size, and the `pass@1` we measured. For MBPP we use the corresponding EvalPlus greedy completions for 22 models from the same release line, including DeepSeek-Coder, WizardCoder, Code Llama, GPT-3.5, and GPT-4-1106. Because instruction-tuned MBPP completions are released as raw model output, we apply a sanitizer that extracts the runnable function (stripping markdown fences and surrounding prose) before grading.

3.3 Execution harness

We re-execute all completions ourselves rather than trusting any pre-computed pass or fail labels. Each candidate program is run in an isolated Python subprocess (the interpreter is launched in isolated mode) with output suppressed and a 12-second wall-clock timeout to bound non-terminating programs. The harness records a single bit per (model, problem) pair, yielding a binary 31×164 response matrix on HumanEval and a 22×378 matrix on MBPP, for $5{,}084 + 8{,}316 = 13{,}400$ executions. As an external validity check, our per-model `pass@1` values reproduce published results closely on both benchmarks. On HumanEval, GPT-4 reaches 88.4%, GPT-3.5 reaches 71.3%, Code Llama 34B reaches 51.2%, and StarCoder reaches 34.1%.

Table 1: The 31-model panel and the HumanEval `pass@1` we measured by re-executing each model's greedy completions. Params in billions; `--` marks undisclosed sizes.

Model	Family	Params (B)	pass@1 (%)
gpt-4	GPT	–	88.4
gpt-4-1106	GPT	–	85.4
gpt-3.5-turbo	GPT	–	71.3
phind-codellama-34b	CodeLlama	34.0	71.3
code-llama-34b	CodeLlama	34.0	51.2
code-llama-13b	CodeLlama	13.0	42.7
code-llama-7b	CodeLlama	7.0	37.8
starcoder	StarCoder	15.0	34.1
codegen-16b	CodeGen	16.0	32.9
codet5p-16b	CodeT5+	16.0	31.7
codegen-6b	CodeGen	6.0	29.3
codet5p-6b	CodeT5+	6.0	29.3
mistral-7b	Mistral	7.0	28.7
codet5p-2b	CodeT5+	2.0	25.0
codegen-2b	CodeGen	2.0	24.4
starcoderbase-7b	StarCoder	7.0	24.4
codegen2-16b	CodeGen2	16.0	19.5
codegen2-7b	CodeGen2	7.0	18.3
starcoderbase-3b	StarCoder	3.0	17.7
vicuna-13b	Vicuna	13.0	17.1
codegen2-3b	CodeGen2	3.0	15.9
incoder-6b	InCoder	6.0	15.9
santacoder	SantaCoder	1.1	14.6
starcoderbase-1b	StarCoder	1.0	14.6
gpt-j	GPT-J	6.0	12.2
incoder-1b	InCoder	1.0	12.2
vicuna-7b	Vicuna	7.0	11.6
codegen2-1b	CodeGen2	1.0	11.0
gpt-neo-2b	GPT-Neo	2.7	7.9
polycoder	PolyCoder	2.7	6.1
stablelm-7b	StableLM	7.0	2.4

On MBPP, GPT-4-1106 reaches 84.9%, GPT-3.5 reaches 84.7%, and DeepSeek-Coder 33B-instruct reaches 80.7%, all within the range reported by EvalPlus and public leaderboards. We later re-grade every completion a second time against the hardened HumanEval+ and MBPP+ suites, and run a temperature-sampling check, for robustness (Section 4.8), bringing the study to roughly 68,000 executions.

3.4 Present-day leaderboard (2026)

To situate the per-problem panels in the current landscape, we additionally use the EvalPlus leaderboard snapshot `results.json` retrieved in June 2026, which records `pass@1` on HumanEval, HumanEval+, MBPP, and MBPP+ for 125 models, including recent systems such as the o1 series, GPT-4o, DeepSeek-V3, and Qwen2.5-Coder. This layer is aggregate (one score per model per benchmark) because per-problem completions are not publicly released for current closed frontier models, so we use it only for population-level statements about saturation and test-suite strength, not for per-problem difficulty.

3.5 Difficulty operationalization

We define the empirical *solve rate* of problem i as the fraction of the 31 models that solve it, and the *difficulty* as $d_i = 1 - \text{solve rate}_i$. To check that this aggregate is not merely an artifact of which models we chose, we also fit a Rasch (1PL) model in which the probability that model j solves problem i is $\sigma(\theta_j - b_i)$, with θ_j a model-ability parameter and b_i the item difficulty. Parameters are estimated by regularized joint maximum likelihood. We then compare the Rasch difficulty b_i with the raw solve rate.

3.6 Features

For each problem we extract 36 features in three families. *Prompt features (12)* describe the specification the model reads: character, word, and line counts of the prompt and docstring, the number of worked input-output examples (counted as doctest lines), the number of example markers, the number of formal parameters, the presence of type hints, a Flesch reading-ease score of the docstring, average word length, and the docstring sentence count. *Reference-solution features (22)* describe the canonical solution: lines of code and character count, abstract-syntax-tree node count, counts of loops, con-

ditionals, returns, calls, arithmetic and boolean operators, comparisons, comprehensions, assignments, and distinct names, maximum nesting and maximum loop-nesting depth, counts of built-in calls and imports, indicators for recursion and string-method use, McCabe cyclomatic complexity (McCabe, 1976), and Halstead volume, difficulty, and effort (Halstead, 1977). *Test features (2)* describe the test suite: the number of assertions and its character count. For MBPP we extract the analogous families (31 features in total). The prompt family is necessarily smaller because MBPP descriptions are single sentences with no docstring or worked examples, so it comprises length, readability, parameter count, and type-hint features computed from the description and the canonical signature; the solution and test families are identical to HumanEval.

3.7 Statistical analysis

For RQ2 we compute Spearman and Pearson correlations between each feature and difficulty, and control the false discovery rate across the 36 tests with the Benjamini-Hochberg procedure (Benjamini and Hochberg, 1995) at $q < 0.05$. For RQ3 we predict difficulty with ridge regression, random forests (Breiman, 2001), and gradient boosting, all evaluated with 10 times repeated 5-fold cross-validation, and we report R^2 and rank correlation against a mean-predicting baseline. Feature importance is computed by permutation on held-out folds, and we run a feature-family ablation to localize the signal. For RQ4 we measure leave-one-model-out and bootstrap stability of the difficulty ranking, transfer between the weaker and stronger halves of the panel, and we repeat the entire pipeline on MBPP to test cross-benchmark generalization. All analyses use scikit-learn (Pedregosa et al., 2011) with fixed random seeds.

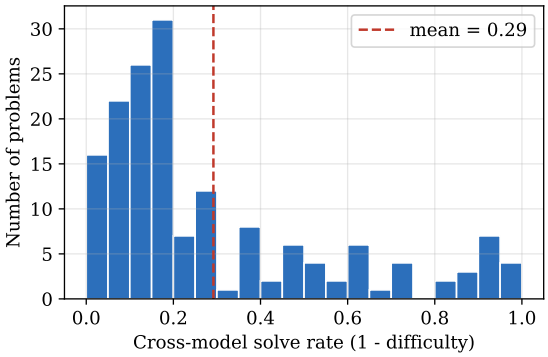


Figure 1: Distribution of per-problem cross-model solve rate over the 31-model panel. Difficulty (1− solve rate) is widely spread; the mean solve rate is 0.29.

4 Results

4.1 RQ1: Difficulty is measurable and well spread

Across the 31-model panel, the mean solve rate is 0.29 and the median is 0.19, so the typical HumanEval problem is solved by fewer than a third of the models. Figure 1 shows that difficulty is widely and non-uniformly distributed rather than concentrated at a single level. Six problems are solved by no model in the panel and two are solved by every model, and 38 problems are solved by at most 10% of models. The per-model `pass@1` in Figure 2 spans a wide range, from 2.4% to 88.4%, confirming that the panel mixes weak and strong solvers, which is exactly the heterogeneity needed to estimate item difficulty.

The Rasch decomposition confirms that the cross-model solve rate is a sound difficulty estimate. The estimated item difficulties b_i, which control for each model's ability, reproduce the solve-rate ordering almost perfectly (Spearman

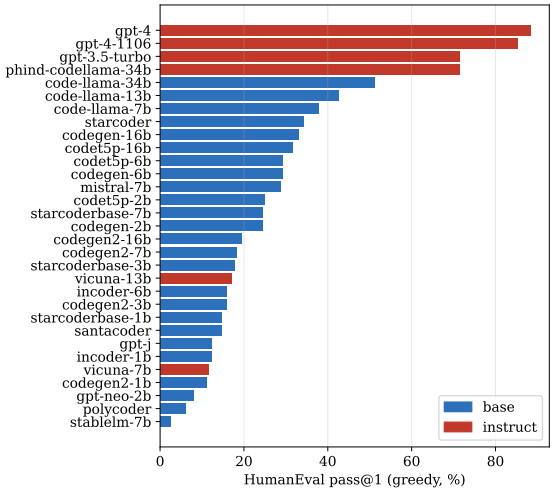


Figure 2: Measured HumanEval `pass@1` for each model in the panel (greedy decoding), colored by base versus instruction-tuned. Values reproduce published results, validating the harness.

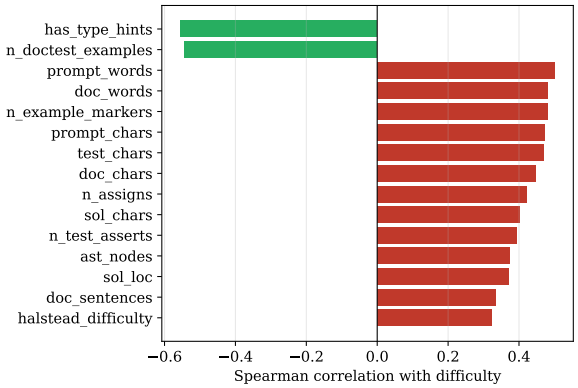


Figure 3: Spearman correlation between each feature and difficulty (top 15 by magnitude). Green bars (negative) mark features that make problems easier; red bars (positive) mark features that make them harder.

$\rho = 1.00$). In other words, on this benchmark the simple cross-model failure rate already captures what a latent-trait model would, so we use it as the difficulty target in the remaining analyses while reporting Rasch as validation.

4.2 RQ2: Specification features dominate the correlates

Of the 36 features, 27 correlate with difficulty at $q < 0.05$ after Benjamini-Hochberg correction. Table 2 and Figure 3 list the strongest. Two patterns stand out. First, the strongest correlates are properties of the *specification*, not of the algorithm. The presence of type hints (Spearman $\rho = -0.56$) and the number of worked input-output examples ($\rho = -0.54$) are associated with *easier* problems, while a longer prompt ($\rho = +0.50$ for word count) and a longer docstring ($\rho = +0.48$) are associated with *harder* problems. Second, reference-solution complexity does correlate with difficulty but more weakly: the number of assignments ($\rho = +0.42$), solution length in characters ($\rho = +0.40$), AST node count ($\rho = +0.37$), and Halstead difficulty ($\rho = +0.32$) all point in the expected direction, yet none reaches the magnitude of the leading prompt features. Test-suite size correlates positively as well ($\rho = +0.47$ for test characters), which we interpret as a proxy for problem richness rather than as a cause.

4.3 RQ3: Difficulty is predictable, and the signal is in the prompt

Table 3 reports cross-validated predictive performance. Against a mean-predicting baseline ($R^2 = -0.07$), all three learners explain a substantial share of variance: ridge regression reaches $R^2 = 0.35$, random forests reach 0.44, and gradient boosting reaches 0.46, with out-of-fold rank correlation around 0.52 to 0.60. Figure 4 shows the out-of-fold predicted difficulty against the true difficulty for the gradient-boosting model.

The most informative result is the feature-family ablation in Figure 5. Using only the 12 prompt features, gradient boosting reaches

Table 2: Features most strongly correlated with difficulty (top 14 by $|\rho|$). Negative values mark features associated with easier problems. q-values are Benjamini-Hochberg corrected across all 36 features.

Feature	Spearman ρ	Pearson r	q-value
has_type_hints	-0.555	-0.582	<0.001
n_doctest_examples	-0.544	-0.514	<0.001
prompt_words	+0.500	+0.446	<0.001
doc_words	+0.482	+0.444	<0.001
n_example_markers	+0.481	+0.330	<0.001
prompt_chars	+0.474	+0.443	<0.001
test_chars	+0.469	+0.439	<0.001
doc_chars	+0.447	+0.421	<0.001
n_assigns	+0.421	+0.406	<0.001
sol_chars	+0.402	+0.404	<0.001
n_test_asserts	+0.394	+0.380	<0.001
ast_nodes	+0.374	+0.420	<0.001
sol_loc	+0.372	+0.379	<0.001
doc_sentences	+0.333	+0.296	<0.001

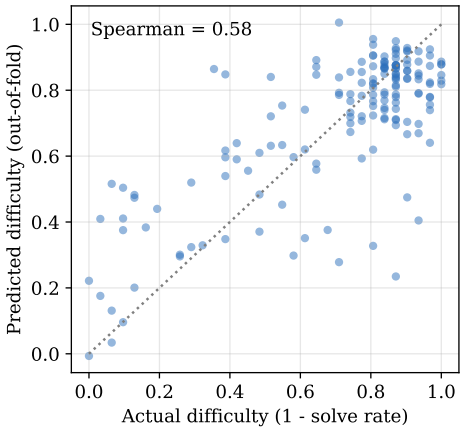


Figure 4: Out-of-fold predicted difficulty versus actual difficulty (gradient boosting, 5-fold). The model recovers the difficulty ordering well (Spearman shown inset).

$R^2 = 0.45$, essentially equal to the full 36-feature model ($R^2 = 0.46$). Using only the 22 reference-solution complexity features, it reaches just $R^2 = 0.11$, and using only the two test features it does no better than the baseline. Combining prompt and solution features yields no meaningful gain over prompt features alone. Permutation importance (Figure 6) tells the same story at the level of individual features: the number of worked examples in the prompt is by far the single most important predictor, followed by the presence of type hints and the docstring word count. A parsimonious model using only these three prompt features already reaches $R^2 = 0.25$.

The practical reading is direct. The difficulty of a HumanEval problem for a code LLM is largely determined by how the task is *specified*, in particular how many concrete examples and type cues the prompt provides and how long the description is, rather than by the structural complexity of the code required to solve it. Difficulty is, to a first approximation, legible from the problem statement that a learner reads before writing any code.

Figure 7 visualizes the strongest single effect. Problems whose prompts contain more worked input-output examples have systematically higher solve rates, and the relationship is close to monotonic in the per-bin means. This mirrors the worked-example effect known from instructional research (Sweller, 1988; Atkinson et al., 2000): concrete examples reduce the effort of inferring the intended behavior, for the model as for the human learner.

4.4 RQ4: Difficulty is a stable property of the problem

The difficulty ordering is highly robust to the composition of the panel. Removing any single model and recomputing solve rates leaves the ranking almost untouched (leave-one-model-out Spearman $\rho \geq 0.99$, mean 1.00). Bootstrap resampling of the 31 models gives a mean rank correlation of 0.96 with the full-panel ranking (95% interval 0.93 to 0.99). Most tellingly, difficulty transfers across capability tiers: when we split

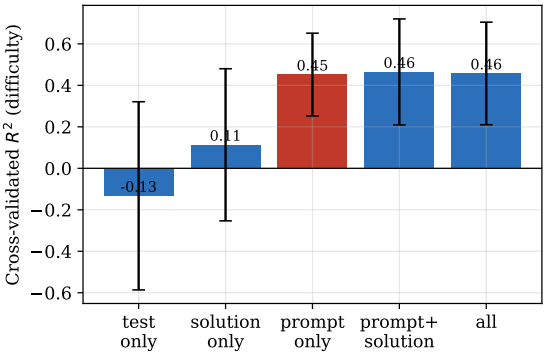


Figure 5: Feature-family ablation (gradient boosting, cross-validated R^2). Prompt features alone (red) match the full feature set, while reference-solution complexity features alone explain little and test features explain nothing.

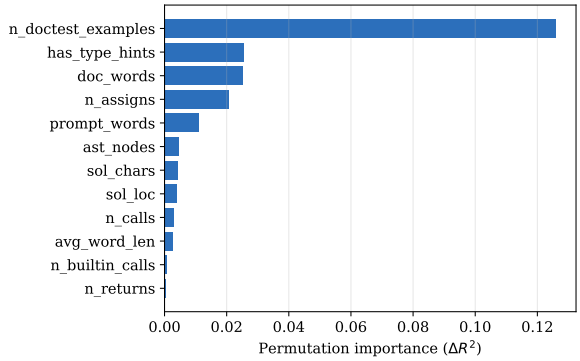


Figure 6: Permutation importance (mean drop in R^2 on held-out folds, random forest). The count of worked input-output examples dominates.

the panel into its weaker and stronger halves by `pass@1` and compute solve rates separately, the two orderings agree at Spearman $\rho = 0.78$ ($p < 10^{-30}$), as shown in Figure 8. Problems that are hard for small base models are, by and large, the same problems that are hard for GPT-4. This supports treating difficulty as a property of the item rather than of any particular solver, and it justifies using static, model-free features to predict it.

Table 4 grounds the scale qualitatively. The hardest problems involve multi-step specifications with implicit constraints and several interacting cases (for example `order_by_points`, `is_nested`, and `do_algebra`), whereas the easiest are short, single-operation tasks with unambiguous behavior (for example `strlen`, `max_element`, and `add`). This is consistent with the quantitative finding that specification clarity, more than algorithmic depth, governs difficulty.

Table 3: Cross-validated prediction of difficulty (10 times repeated 5-fold). The feature-family ablation shows prompt features alone match the full feature set, while reference-solution complexity alone explains little.

Predictor set / model	R^2	Spearman ρ
Baseline (predict mean)	-0.072	–
All 36 features		
Ridge	0.348±0.254	0.600±0.145
RandomForest	0.441±0.221	0.529±0.157
GradientBoosting	0.457±0.247	0.519±0.175
Feature-family ablation (gradient boosting)		
Test features only (2)	-0.133±0.454	–
Solution features only (22)	0.113±0.367	–
Prompt features only (12)	0.452±0.200	–
Prompt + solution (34)	0.465±0.256	–
All features (36)	0.457±0.247	–

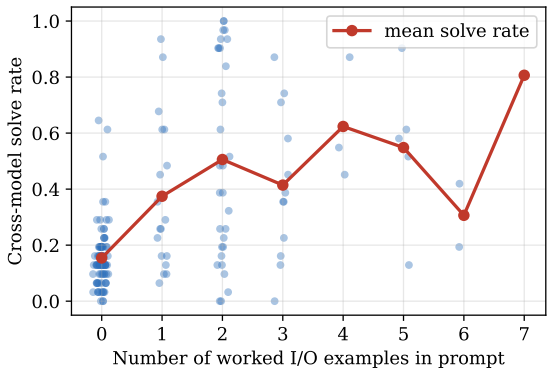


Figure 7: Cross-model solve rate against the number of worked input-output examples in the prompt. Points are individual problems (jittered); the red line is the per-count mean.

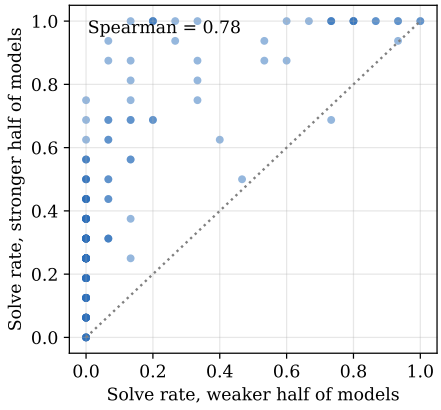


Figure 8: Per-problem solve rate for the weaker half of the panel versus the stronger half. The strong agreement ($\rho = 0.78$) shows difficulty transfers across capability tiers.

Table 4: The hardest and easiest HumanEval problems by cross-model solve rate.

Task	Function	Solve rate
Hardest		
HumanEval/108	`count_nums`	0.000
HumanEval/109	`move_one_ball`	0.000
HumanEval/93	`encode`	0.000
HumanEval/132	`is_nested`	0.000
HumanEval/145	`order_by_points`	0.000
HumanEval/163	`generate_integers`	0.000
HumanEval/91	`is_bored`	0.032
HumanEval/32	`find_zero`	0.032
Easiest		
HumanEval/35	`max_element`	1.000
HumanEval/23	`strlen`	1.000
HumanEval/29	`filter_by_prefix`	0.968
HumanEval/53	`add`	0.968
HumanEval/42	`incr_list`	0.935
HumanEval/45	`triangle_area`	0.935
HumanEval/28	`concatenate`	0.935
HumanEval/58	`common`	0.903

4.5 Cross-benchmark generalization on MBPP

Repeating the entire pipeline on MBPP (378 problems, 22 models) tests whether these conclusions are specific to HumanEval. Two things generalize cleanly and one changes in an informative way. What generalizes is the *nature* of difficulty. MBPP difficulty is again a stable, model-agnostic property: leave-one-model-out rank correlation is at least 0.996, and the weaker and stronger halves of the panel agree at Spearman $\rho = 0.79$, almost identical to HumanEval. Of the 31 MBPP features, 24 correlate significantly with difficulty after correction. MBPP is easier overall, with a mean solve rate of 0.55 against 0.29 for HumanEval, as expected for shorter tasks.

What changes is *where* the predictive signal lives, and this is the central cross-benchmark result. Table 5 and Figure 9 contrast the feature-family ablation on the two benchmarks. On HumanEval, prompt features carry essentially all the signal. On MBPP the opposite holds: solution-complexity features explain the most variance (cross-validated $R^2 = 0.13$), while prompt features alone are not predictive at all (negative R^2), and the strongest individual correlates are the abstract-syntax-tree node count ($\rho = 0.46$) and Halstead volume rather than any prompt property. The explanation is straightforward and is exactly what a moderation account predicts. HumanEval prompts vary widely in richness, including from zero to several worked examples, so that variation dominates and predicts difficulty. MBPP prompts are uniformly terse single sentences with no worked examples, so prompt features have little variance to contribute, and difficulty is instead governed by the intrinsic complexity of the solution the task demands. The unifying statement is therefore not that prompts always matter more than algorithms, but that difficulty is driven by whatever varies most in how the problem is specified. Overall predictability is also lower on MBPP ($R^2 = 0.14$ for the full gradient-boosting model), consistent with terse specifications leaving less

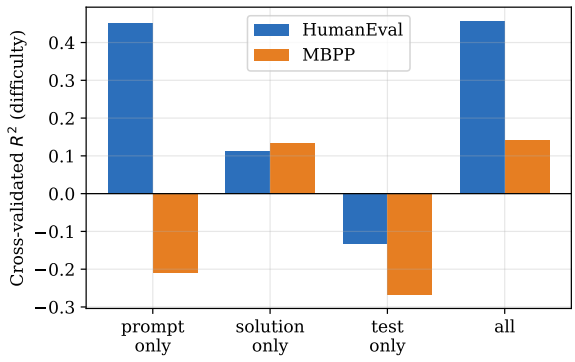


Figure 9: Feature-family ablation on the two benchmarks (gradient boosting, cross-validated R^2). On HumanEval the prompt carries the signal; on MBPP, whose prompts are uniformly terse, the reference-solution complexity does.

Table 5: Cross-benchmark summary. Difficulty is a stable, predictable property on both benchmarks, but the predictive signal moves from the prompt (HumanEval) to the solution (MBPP) with the specification style.

Property	HumanEval	MBPP
Problems	164	378
Models in panel	31	22
Mean solve rate	0.29	0.55
Significant features (FDR)	27/36	24/31
Predicting difficulty, cross-validated R^2		
Gradient boosting (all)	0.46	0.14
Prompt features only	0.45	-0.21
Solution features only	0.11	0.13
Difficulty stability		
Leave-one-model-out ρ (min)	0.99	1.00
Weak/strong tier transfer ρ	0.78	0.79

that is legible from static features.

4.6 Present-day context: saturation and test strength in 2026

The per-problem panels are drawn from models whose raw completions are public, which is the 2023 to early 2024 generation. To connect the study to the current state of the field, we turn to the June 2026 EvalPlus leaderboard of 125 models, which adds recent systems such as the o1 series, GPT-4o, DeepSeek-V3, and Qwen2.5-Coder. Three observations follow, all at the aggregate level (Table 7). First, HumanEval is now saturated: the top model reaches 96.3%, the median across all 125 models is 56.7%, and seven models exceed 90% (Figure 10). Eight current models score above the strongest model in our HumanEval panel, so the frontier has clearly moved beyond the panel. Second, the base test suites are lenient: across models with both scores, moving to the hardened EvalPlus tests lowers `pass@1` by 5.0 points on average for HumanEval and by 11.3 points for MBPP (Figure 11). The larger MBPP gap is consistent with our use of base MBPP tests and means our MBPP solve rates somewhat overstate absolute capability, though they preserve the relative difficulty ordering that our analysis uses. Third, the two benchmarks rank models almost identically at the population level (Spearman $\rho = 0.94$), which complements our per-problem finding that the benchmarks nonetheless probe difficulty through different surface features.

Saturation is precisely why item difficulty matters going forward. As aggregate `pass@k` compresses toward 100%, the information in a benchmark concentrates in its hard tail, and a difficulty-aware view is what distinguishes a benchmark that has been solved from one whose few remaining hard items still separate models. Our evidence that difficulty is stable and transfers across capability tiers suggests that the hard items identified with the present panel remain informative for stronger models, even though we cannot execute those models' per-problem outputs.

4.7 The mid-2026 frontier has outgrown HumanEval

The trend is even clearer one model generation further on. By the middle of 2026 the leading coding systems are Claude Opus 4.7 and 4.8 and the access-restricted Fable 5, GPT-5.4 and 5.5, Gemini 3.1 Pro, and open-weight models such as DeepSeek-V4 and Qwen3.x. None of these systems report HumanEval or MBPP in their model cards, and the public leaderboards that track them no longer use these benchmarks. They are evaluated instead on repository-level and contamination-resistant suites: SWE-bench Verified (Jimenez et al., 2024), the harder standardized SWE-bench Pro, and LiveCodeBench (Jain et al., 2024). Table 6 reports representative externally published numbers. Two points matter for this paper. First, the migration away from HumanEval is itself difficulty-seeking behavior by the community, the same logic our analysis formalizes at the level of individual items: once a benchmark is solved, attention moves to where difficulty still lives. Second, the new benchmarks are far from solved, with even the strongest systems below 90% on SWE-bench Verified and below 60% on the standardized SWE-bench Pro harness, so per-instance difficulty modeling of the kind we demonstrate here is, if anything, more useful on them than on HumanEval. Our methodology, which only requires a set of per-instance pass or fail outcomes across a panel of solvers, applies directly to these benchmarks as such results are released.

Table 6: The mid-2026 coding frontier is measured on SWE-bench-class benchmarks, not HumanEval, which it has saturated.

Frontier model (mid-2026)	SWE-bench Verified (%)
Claude Fable 5	95.0
Claude Opus 4.8	88.6
Claude Opus 4.7	87.6
Claude Opus 4.5	80.9
Gemini 3.1 Pro	80.6
DeepSeek-V4-Pro-Max	80.6
Qwen3.7 Max	80.4

Externally reported scores from public mid-2026 leaderboards and vendor cards, not re-executed by us. Fable 5 is access-restricted; DeepSeek-V4 and Qwen3.x are open-weight; Gemini 3.1 Pro leads LiveCodeBench. None of these systems report HumanEval, which they have saturated. On the standardized SWE-bench Pro harness, scores fall by 30 points or more (GPT-5.4 59.1%, Claude Opus 4.6 51.9%, Gemini 3.1 Pro 46.1%), showing that harder, contamination-resistant benchmarks retain difficulty headroom.

4.8 Robustness to hardened test suites

A standing concern with HumanEval and MBPP is that their base test suites are lenient, so a program can pass while being subtly wrong. If our difficulty labels were driven by such false passes, they might not survive stronger testing. We therefore re-graded every completion a second time against the hardened EvalPlus suites, HumanEval+ with roughly 80 times more tests and MBPP+ with roughly 35 times more, a further 13,400 executions. As expected, scores fall: `pass@1` drops by 3.4 points on average across our HumanEval panel and 8.4 points across our MBPP panel, with the larger MBPP gap matching the leniency we measured on the full 2026 leaderboard, and stronger models losing the most.

The two central claims of the paper are unchanged under hardened testing (Table 8 and Figure 12). First, difficulty remains a stable property of the problem. Per-problem difficulty under the base and hardened suites is rank-correlated at Spearman $\rho = 0.88$ on HumanEval and 0.82 on MBPP, and the panels' model rankings are essentially identical across

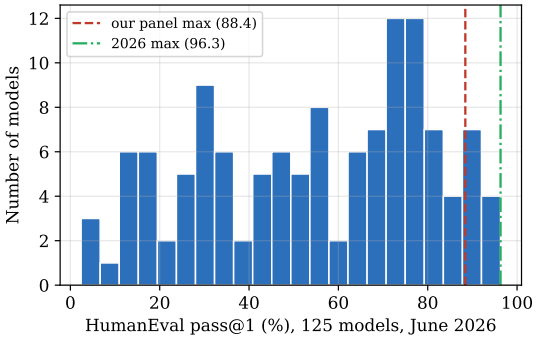


Figure 10: HumanEval `pass@1` across the 125 models on the June 2026 EvalPlus leaderboard. The benchmark is saturating: many models cluster near the ceiling, and eight exceed the best model in our per-problem panel.

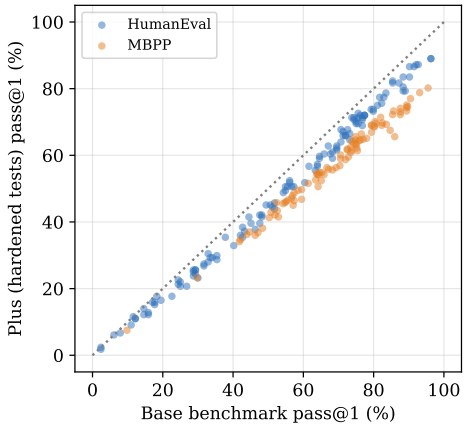


Figure 11: Base versus hardened (plus) `pass@1` for the 125 models. Points fall below the diagonal, more so for MBPP, showing that the base test suites are lenient.

Table 7: Present-day (June 2026) EvalPlus leaderboard context, 125 models.

Current top model (June 2026)	HE	HE+
O1 Preview (Sept 2024)	96.3	89.0
O1 Mini (Sept 2024)	96.3	89.0
GPT 4o (Aug 2024)	92.7	87.2
Qwen2.5-Coder-32B-Instruct	92.1	87.2
DeepSeek-V3 (Nov 2024)	91.5	86.6
GPT-4-Turbo (April 2024)	90.2	86.6
DeepSeek-V2.5 (Nov 2024)	90.2	83.5
Gemini 1.5 Pro 002	89.0	79.3

125 models in snapshot. HumanEval median 56.7, max 96.3; 7 models reach $\geq$90%. Base-to-plus mean drop: HE 5.0 pts, MBPP 11.3 pts. HE/MBPP model ranking agree at $\rho = 0.94$.

Table 8: Robustness to the hardened EvalPlus test suites. The difficulty ranking and the prompt-versus-solution conclusion survive an 80x (HumanEval) and 35x (MBPP) increase in test strength.

Robustness check	HumanEval	MBPP
Difficulty rank base vs plus (ρ)	0.877	0.821
Model pass@1 rank base vs plus (ρ)	0.995	0.987
Mean pass@1 drop to plus (pts)	3.4	8.4
Prompt-only / solution-only ablation R^2 on hardened difficulty		
Prompt features only	0.279	-0.208
Solution features only	0.142	0.21

the two regimes ($\rho \geq 0.99$ on both). About 29% of HumanEval problems and 39% of MBPP problems become harder under hardening, more for MBPP because its base tests were weaker, but very few problems are reordered drastically. Second, the cross-benchmark moderation holds: re-running the feature-family ablation against hardened difficulty, prompt features still dominate on HumanEval (prompt-only $R^2 = 0.28$ versus solution-only 0.14) and solution-complexity features still dominate on MBPP (solution-only $R^2 = 0.21$ versus a negative prompt-only R^2). The one informative shift is that on HumanEval the absolute predictability falls, from 0.46 to 0.28, and the solution family gains slightly while the prompt family loses, which is sensible: hardened tests add edge-case failures that depend on getting the implementation exactly right, a solution-side property, on top of the specification-comprehension failures that the base tests already expose. The qualitative conclusion, that difficulty is legible from whatever the benchmark leaves most variable in its specification, is unaffected.

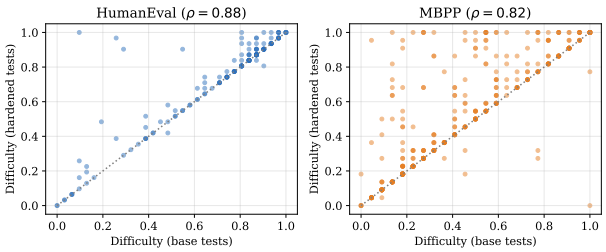


Figure 12: Per-problem difficulty under base versus hardened tests. Points lie close to the diagonal (most slightly above, that is, harder under hardening), and the rank correlation stays high on both benchmarks.

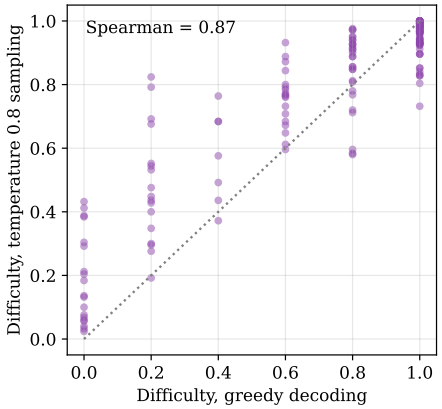


Figure 13: Greedy versus temperature-0.8 sampling difficulty for five models with 200 released samples per problem. The difficulty ranking is essentially unchanged by the decoding regime.

Decoding temperature. Greedy decoding is only one way to run a model, so difficulty might look different under stochastic sampling. Five models in our panel also released 200 samples per problem at temperature 0.8. We executed 50 samples each against the canonical HumanEval tests, a further 41,000 executions, and computed a sampling-based difficulty as one minus the expected per-sample pass rate. It ranks problems almost identically to greedy difficulty for the same five models (Spearman $\rho = 0.87$, Figure 13), with mean pass@1 falling from 0.30 to 0.24 and pass@10 rising to 0.53, as expected for sampling. The prompt-versus-solution ablation is again preserved: prompt features predict sampling difficulty at $R^2 = 0.44$ against 0.19 for solution-complexity features. Item difficulty is therefore stable along three independent axes, the choice of model panel, the strength of the test suite, and the decoding regime.

5 Discussion and Implications

Difficulty-aware benchmarking. Because difficulty is stable and predictable, benchmark reports can and should go beyond a single `pass@k`. A difficulty profile, for instance the fraction of items in easy, medium, and hard bands, would expose whether a benchmark is saturating because models improved or because most of its items are trivial. Six problems in HumanEval are solved by no model in our panel and two by all of them, so both tails carry little discriminative information for current models, echoing the argument that evaluation examples are not equally informative (Rodriguez et al., 2021). Static difficulty predictors offer a cheap way to curate balanced benchmarks without first running a fleet of models.

Autograding and feedback. For LLM-assisted autograders and tutoring systems, a static difficulty estimate is a useful confidence signal. An assistant can defer to a human, or request additional examples, on exactly those specifications that the predictor flags as hard, before generating a possibly wrong solution or explanation. When the specification is rich, as in HumanEval, the estimate is available at authoring time from the prompt alone; when it is terse, as in MBPP, a reference solution or a cheap proxy

for solution complexity is needed. Either way the estimate does not require running a fleet of models.

Computing education. Our central finding has a constructive reading for instructors who use LLMs to generate or grade exercises (Sarsa et al., 2022; Denny et al., 2024). The features that make a problem easy for a model, abundant worked examples and explicit type information, are also classic scaffolds for novice programmers (Sweller, 1988; Atkinson et al., 2000). This alignment is a double-edged result. On one hand, exercises written with rich examples are precisely those an assistant can reliably solve and explain, which is helpful for generating reference material. On the other hand, the same exercises are the ones a struggling student is most likely to off-load to an assistant, which has implications for assessment design (Becker et al., 2023; Prather et al., 2023). Instructors who wish an exercise to remain challenging for an assistant can, paradoxically, do so by specifying it more tersely, though this trades off against clarity for students. Difficulty prediction makes this trade-off explicit and measurable.

6 Threats to Validity

Construct validity. We operationalize difficulty as the cross-model failure rate under greedy decoding. This conflates several sources of difficulty. We mitigate the concern by validating against a Rasch model that separates ability from difficulty, with near-perfect agreement, by showing tier transfer, and by confirming that the ranking is essentially unchanged under temperature-0.8 sampling (Section 4.8). Some features, notably test-suite size, are best read as proxies for problem richness rather than causes; our causal language is deliberately restricted to the specification features whose direction is interpretable.

Internal validity. We grade primarily against the base benchmark tests, which are known to be incomplete (Liu et al., 2023), so a small number of programs may pass spuriously. Rather than leave this as a caveat, we re-graded every completion against the hardened HumanEval+ and MBPP+ suites; as reported in Section 4.8, both the difficulty ranking (Spearman $\rho = 0.88$ and 0.82 to the base ordering) and the prompt-versus-solution conclusion are preserved, so our results do not depend on test leniency. A second internal threat is the sanitizer applied to raw MBPP completions; a faulty sanitizer would understate a model's score. Our recovered MBPP `pass@1` values match published EvalPlus numbers for the same models on both base and hardened tests, which bounds this risk.

External validity. Our per-problem results concern HumanEval and MBPP, two benchmarks of short, self-contained Python functions, and panels of models whose raw completions are public, which is the 2023 to early 2024 generation. The findings may not transfer to repository-level tasks (Du et al., 2024; Zhuo et al., 2024), to other languages, or to the most recent reasoning-oriented models, and both benchmarks are subject to possible training-data contamination (Jacovi et al., 2023; Jain et al., 2024). We mitigate the model-recency concern with the 125-model June 2026 leaderboard layer, which shows that difficulty remains a meaningful axis as aggregate scores saturate, but we cannot compute per-problem difficulty for current

closed frontier models because their completions are not released. The cross-benchmark contrast between HumanEval and MBPP shows that the methodology generalizes and, importantly, that its conclusions are sensitive to specification style in a predictable way rather than being a single fixed rule.

Conclusion validity. With 164 and 378 problems, cross-validated estimates carry non-trivial variance, which we report as standard deviations across repeated folds. We control the family-wise error of the correlation analyses with false-discovery-rate correction. Our central cross-benchmark claim, that the predictive signal sits in the prompt on HumanEval and in the solution on MBPP, is a qualitative reversal that is large relative to fold-to-fold noise.

7 Conclusion

We studied per-problem difficulty for code LLMs by re-executing the public completions of 31 models on HumanEval and 22 on MBPP, analyzing 13,400 outcomes. On both benchmarks difficulty is measurable, broadly distributed, and a stable property of the problem rather than of the solver: it survives model removal, bootstrap resampling, and transfer across capability tiers. It is also predictable, and the cross-benchmark comparison locates the predictive signal precisely. On HumanEval, whose prompts are rich and example-laden, difficulty is legible from the natural-language specification, and prompt features alone match the full model. On MBPP, whose prompts are uniformly terse, the signal moves to the structural complexity of the required solution. The unifying lesson is that a problem is hard for a code LLM along whatever dimension its specification leaves most variable. Placing these panels against the June 2026 leaderboard of 125 models shows HumanEval saturating for frontier systems, which makes a difficulty-aware view more, not less, important. For benchmark designers, this argues for difficulty-aware reporting and curation. For computing educators, it links the scaffolds that help models to the scaffolds that help learners, and it gives a concrete, static handle on how hard an LLM will find a given exercise. We release all data and code to support replication and extension to other benchmarks, languages, and model generations.

8 Data Availability

The replication package contains the download and execution scripts, the binary response matrices for both benchmarks, the per-problem feature tables, all analysis scripts, and the figures and tables in this paper. The underlying model completions are the public EvalPlus releases (Liu et al., 2023), the HumanEval and MBPP problems are from their original releases (Chen et al., 2021; Austin et al., 2021), and the 2026 context is the public EvalPlus leaderboard snapshot; our scripts record the source URLs and content hashes of every downloaded artifact. The complete replication package is archived at `https://doi.org/10.5281/zenodo.20766479`.

Statements and Declarations

Funding. The author received no specific funding for this work.

Competing interests. The author declares no competing interests.

Ethics approval and consent. Not applica-

ble; the study analyzes publicly released model outputs and benchmark artifacts and involves no human subjects.

Data availability. A complete replication package (download and execution scripts, the binary response matrices for both benchmarks under base and hardened tests, the per-problem feature tables, all analysis scripts, and the figures and tables in this paper) is archived at https://doi.org/10.5281/zenodo.20766479. The underlying model completions are the public EvalPlus releases; the HumanEval and MBPP problems are from their original releases; the 2026 context uses the public EvalPlus leaderboard snapshot. All download scripts record the source URLs and content hashes of every artifact.

Use of generative AI. The author used a large language model to assist with drafting and with implementing analysis scripts; the author verified all results and takes full responsibility for the content.

Author contributions. The sole author designed the study, implemented the analysis, and wrote the manuscript.

References

Loubna Ben Allal, Raymond Li, Denis Kocetkov, Chenghao Mou, Christopher Akiki, Carlos Munoz Ferrandis, Niklas Muennighoff, Mayank Mishra, Alex Gu, et al. SantaCoder: don't reach for the stars! *arXiv preprint arXiv:2301.03988*, 2023.

Robert K. Atkinson, Sharon J. Derry, Alexander Renkl, and Donald Wortham. Learning from examples: Instructional principles from the worked examples research. *Review of Educational Research*, 70(2):181–214, 2000. doi: 10.3102/00346543070002181.

Jacob Austin, Augustus Odena, Maxwell Nye, Maarten Bosma, Henryk Michalewski, David Dohan, Ellen Jiang, Carrie Cai, Michael Terry, Quoc Le, and Charles Sutton. Program synthesis with large language models. *arXiv preprint arXiv:2108.07732*, 2021.

Frank B. Baker. *The Basics of Item Response Theory*. ERIC Clearinghouse on Assessment and Evaluation, 2nd edition, 2001.

Brett A. Becker, Paul Denny, James Finnie-Ansley, Andrew Luxton-Reilly, James Prather, and Eddie Antonio Santos. Programming is hard – or at least it used to be: Educational opportunities and challenges of AI code generation. In *Proceedings of the 54th ACM Technical Symposium on Computer Science Education (SIGCSE)*, pages 500–506, 2023. doi: 10.1145/3545945.3569759.

Yoav Benjamini and Yosef Hochberg. Controlling the false discovery rate: A practical and powerful approach to multiple testing. *Journal of the Royal Statistical Society: Series B*, 57(1):289–300, 1995.

Leo Breiman. Random forests. *Machine Learning*, 45(1):5–32, 2001. doi: 10.1023/A:1010933404324.

Mark Chen, Jerry Tworek, Heewoo Jun, Qiming Yuan, Henrique Ponde de Oliveira Pinto, Jared Kaplan, Harri Edwards, Yuri Burda, Nicholas Joseph, Greg Brockman, et al. Evaluating large language models trained on code. *arXiv preprint arXiv:2107.03374*, 2021.

Paul Denny, James Prather, Brett A. Becker, James Finnie-Ansley, Arto Hellas, Juho

Leinonen, Andrew Luxton-Reilly, Brent N. Reeves, Eddie Antonio Santos, and Sami Sarsa. Computing education in the era of generative AI. *Communications of the ACM*, 67 (2):56–67, 2024. doi: 10.1145/3624720.

Xueying Du, Mingwei Liu, Kaixin Wang, Hanlin Wang, Junwei Liu, Yixuan Chen, Jiayi Feng, Chaofeng Sha, Xin Peng, and Yiling Lou. Evaluating large language models in class-level code generation. In *Proceedings of the IEEE/ACM 46th International Conference on Software Engineering (ICSE)*, 2024.

James Finnie-Ansley, Paul Denny, Brett A. Becker, Andrew Luxton-Reilly, and James Prather. The robots are coming: Exploring the implications of OpenAI codex on introductory programming. In *Proceedings of the 24th Australasian Computing Education Conference (ACE)*, pages 10–19, 2022. doi: 10.1145/3511861.3511863.

Daniel Fried, Armen Aghajanyan, Jessy Lin, Sida Wang, Eric Wallace, Freda Shi, Ruiqi Zhong, Wen-tau Yih, Luke Zettlemoyer, and Mike Lewis. InCoder: A generative model for code infilling and synthesis. In *International Conference on Learning Representations (ICLR)*, 2023.

Maurice H. Halstead. *Elements of Software Science*. Elsevier North-Holland, 1977.

Dan Hendrycks, Steven Basart, Saurav Kadavath, Mantas Mazeika, Akul Arora, Ethan Guo, Collin Burns, Samir Puranik, Horace He, Dawn Song, and Jacob Steinhardt. Measuring coding challenge competence with APPS. In *Advances in Neural Information Processing Systems (NeurIPS), Datasets and Benchmarks Track*, 2021.

Alon Jacovi, Avi Caciularu, Omer Goldman, and Yoav Goldberg. Stop uploading test data in plain text: Practical strategies for mitigating data contamination by evaluation benchmarks. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 5075–5084, 2023.

Naman Jain, King Han, Alex Gu, Wen-Ding Li, Fanjia Yan, Tianjun Zhang, Sida Wang, Armando Solar-Lezama, Koushik Sen, and Ion Stoica. LiveCodeBench: Holistic and contamination free evaluation of large language models for code. *arXiv preprint arXiv:2403.07974*, 2024.

Albert Q. Jiang, Alexandre Sablayrolles, Arthur Mensch, Chris Bamford, Devendra Singh Chaplot, Diego de las Casas, Florian Bressand, Gianna Lengyel, Guillaume Lample, et al. Mistral 7B. *arXiv preprint arXiv:2310.06825*, 2023.

Carlos E. Jimenez, John Yang, Alexander Wettig, Shunyu Yao, Kexin Pei, Ofir Press, and Karthik Narasimhan. SWE-bench: Can language models resolve real-world GitHub issues? In *International Conference on Learning Representations (ICLR)*, 2024.

John P. Lalor, Hao Wu, and Hong Yu. Building an evaluation scale using item response theory. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 648–657, 2016. doi: 10.18653/v1/D16-1062.

Raymond Li, Loubna Ben Allal, Yangtian Zi, Niklas Muennighoff, Denis Kocetkov, Chenghao Mou, Marc Marone, Christopher Akiki, Jia Li, et al. StarCoder: may the source be

with you! *Transactions on Machine Learning Research (TMLR)*, 2023.

Yujia Li, David Choi, Junyoung Chung, Nate Kushman, Julian Schrittwieser, Rémi Leblond, Tom Eccles, James Keeling, Felix Gimeno, et al. Competition-level code generation with AlphaCode. *Science*, 378(6624):1092–1097, 2022. doi: 10.1126/science.abq1158.

Jiawei Liu, Chunqiu Steven Xia, Yuyao Wang, and Lingming Zhang. Is your code generated by chatGPT really correct? rigorous evaluation of large language models for code generation. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2023.

Ziyang Luo, Can Xu, Pu Zhao, Qingfeng Sun, Xiubo Geng, Wenxiang Hu, Chongyang Tao, Jing Ma, Qingwei Lin, and Daxin Jiang. WizardCoder: Empowering code large language models with evol-instruct. *arXiv preprint arXiv:2306.08568*, 2023.

Thomas J. McCabe. A complexity measure. *IEEE Transactions on Software Engineering*, SE-2(4):308–320, 1976. doi: 10.1109/TSE.1976.233837.

Erik Nijkamp, Bo Pang, Hiroaki Hayashi, Lifu Tu, Huan Wang, Yingbo Zhou, Silvio Savarese, and Caiming Xiong. CodeGen: An open large language model for code with multi-turn program synthesis. In *International Conference on Learning Representations (ICLR)*, 2023.

OpenAI. GPT-4 technical report. *arXiv preprint arXiv:2303.08774*, 2023.

Fabian Pedregosa, Gaël Varoquaux, Alexandre Gramfort, Vincent Michel, Bertrand Thirion, Olivier Grisel, Mathieu Blondel, Peter Prettenhofer, Ron Weiss, et al. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.

James Prather, Paul Denny, Juho Leinonen, Brett A. Becker, Ibrahim Albluwi, Michelle Craig, Hieke Keuning, Natalie Kiesler, Tobias Kohn, et al. The robots are here: Navigating the generative AI revolution in computing education. In *Proceedings of the 2023 Working Group Reports on Innovation and Technology in Computer Science Education (ITiCSE-WGR)*, pages 108–159, 2023. doi: 10.1145/3623762.3633499.

Georg Rasch. *Probabilistic Models for Some Intelligence and Attainment Tests*. Danish Institute for Educational Research, Copenhagen, 1960.

Pedro Rodriguez, Joe Barrow, Alexander Miserlis Hoyle, John P. Lalor, Robin Jia, and Jordan Boyd-Graber. Evaluation examples are not equally informative: How should that change NLP leaderboards? In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 4486–4503, 2021. doi: 10.18653/v1/2021.acl-long.346.

Baptiste Rozière, Jonas Gehring, Fabian Gloeckle, Sten Sootla, Itai Gat, Xiaoqing Ellen Tan, Yossi Adi, Jingyu Liu, Tal Remez, et al. Code llama: Open foundation models for code. *arXiv preprint arXiv:2308.12950*, 2023.

Sami Sarsa, Paul Denny, Arto Hellas, and Juho Leinonen. Automatic generation of programming exercises and code explanations using large language models. In *Proceedings of the*

2022 ACM Conference on International Computing Education Research (ICER), pages 27–43, 2022. doi: 10.1145/3501385.3543957.

John Sweller. Cognitive load during problem solving: Effects on learning. *Cognitive Science*, 12(2):257–285, 1988. doi: 10.1207/s15516709cog1202_4.

Clara Vania, Phu Mon Htut, William Huang, Dhara Mungra, Richard Yuanzhe Pang, Jason Phang, Haokun Liu, Kyunghyun Cho, and Samuel R. Bowman. Comparing test sets with item response theory. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 1141–1158, 2021. doi: 10.18653/v1/2021.acl-long.92.

Yue Wang, Hung Le, Akhilesh Deepak Gotmare, Nghi D.Q. Bui, Junnan Li, and Steven C.H. Hoi. CodeT5+: Open code large language models for code understanding and generation. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 2023.

Lianmin Zheng, Wei-Lin Chiang, Ying Sheng, Siyuan Zhuang, Zhanghao Wu, Yonghao Zhuang, Zi Lin, Zhuohan Li, Dacheng Li, Eric P. Xing, et al. Judging LLM-as-a-judge with MT-bench and chatbot arena. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2023.

Terry Yue Zhuo, Minh Chien Vu, Jenny Chim, Han Hu, Wenhao Yu, Ratnadira Widyasari, Imam Nur Bani Yusuf, Haolan Zhan, Junda He, et al. BigCodeBench: Benchmarking code generation with diverse function calls and complex instructions. *arXiv preprint arXiv:2406.15877*, 2024.