textOsFtextTOsFliningLFliningTLFtextosfliningIftabulartabproportionalprop superiorSup

su-
pe-
ri-
or-
Sup

fontspechyperref

Early-Stage Prediction of Review Effort in AI-Generated Pull Requests

Anonymous Author(s)

Abstract

The rise of autonomous coding agents promises a productivity boom but threatens a review bottleneck. In this paper, we investigate the "hidden cost" of this AI workforce: a high rate of abandonment ('ghosting') in complex tasks. Analyzing 33,596 PRs from the *AIDev* dataset, we uncover a stark "Two-Regime" reality: while 32.6% of agentic PRs merge instantly, among contributions rejected after receiving feedback, **64.5%** are ghosted by their own creators, wasting significant maintainer effort. We propose a creation-time **Circuit Breaker** model to intercept these high-cost contributions. Surprisingly, we find that simple static complexity signals (e.g., patch size) alone yield near-perfect triage accuracy (AUC 0.94), rendering complex text analysis redundant. By gating just the top 20% riskiest PRs, maintainers can preemptively catch **82.8%** of high-effort drains, enabling sustainable human-AI collaboration.

CCS Concepts

• **Software and its engineering → Software evolution**.

Keywords

AI Agents, Triage, Ghosting, Mining Software Repositories

1 Introduction

As AI agents evolve from assistants to autonomous teammates [20], they are beginning to flood repositories with code. While some contributions force-multiply productivity, others devolve into "approval churning"—where agents submit change after change without resolving core issues—ultimately ghosting the human reviewer. Using the AIDev dataset, we identify a critical *two-regime* pattern: a subset of agentic PRs merges seamlessly, while the rest risk becoming significant time sinks. This motivates an urgent need for automated governance: can we identify these high-effort drains *before* a human reviewer engages?

Research Questions.

RQ1: Can creation-time structural signals predict high-effort PRs that will demand substantial review effort?

RQ2: Which early cues are associated with higher propensity for agentic ghosting?

Contributions. (1) We operationalize *agentic ghosting* and quantify its prevalence in AIDev PRs, highlighting its concentration among rejected / non-converging PRs. (2) We characterize a *two-regime* PR outcome distribution that separates low-friction PRs from a high-cost tail. (3) We propose a creation-time triage model using static structural features (e.g., patch size), achieving strong predictive performance (AUC 0.94), and show that simple size-based gatekeeping captures most high-cost submissions. For reproducibility, we release code and scripts via an anonymized artifact (Zenodo: https://zenodo.org/records/17993901).

MSR 2026, Rio de Janeiro, Brazil
2026. ACM ISBN 978-x-xxxx-xxxx-x/YYYY/MM
https://doi.org/10.1145/nnnnnnn.nnnnnnn

1.1 Related Work

Prior work on software bots largely focuses on deterministic automation for dependency updates, review assistance, and CI workflows [12, 19], whereas generative agents introduce non-deterministic code changes that demand extra verification and coordination [4, 18]. In parallel, PR triage models often rely on contributor history and social signals [17], and studies of human PR abandonment highlight review-process dynamics and contributor reputation as key drivers [11]; our setting differs because we study *agent-authored* PRs and ask whether *static creation-time* structure (e.g., change size and file characteristics) is enough to anticipate high review effort and subsequent non-response. Finally, while AI code generation research typically evaluates correctness, usability, and downstream risks [15], and security-oriented analyses suggest larger-scope changes can correlate with higher risk [1], we target a complementary axis: maintainer time cost caused by effort-heavy reviews and ghosting, and we show that lightweight, creation-time signals can support practical gatekeeping without depending on historical contributor reputation.

2 Methodology

2.1 Dataset Curation

We use the **AIDev dataset v1.0 (snapshot: October 2025)** [13]. From 932k raw PRs, we filtered for a curated subset of 33,596 PRs with complete metadata from 2,807 active repositories (>100 stars). Agents are distinguished via metadata (author type 'Bot' + known display names).

2.2 Feature Engineering

2.2.1 Features & Time Horizons. We extract 35 features across three categories: Intent, Context, and Complexity. We evaluate predictability at two stages. T0 includes 35 signals: **Complexity** (additions, entropy), **Intent** (body length, has_plan (keyword regex)), and **Context** (files). **T0 (Creation-Time)** includes all signals available immediately upon PR submission. **T1 (Pre-Review)** adds Interaction signals accumulated *before* the first human feedback event. This precise cutoff ensures no leakage from human feedback into predictive features.

2.3 Modeling Approach

We frame triage as binary classification targeting **High Cost** PRs (top 20% by effort score). We employ a **Repo-Disjoint Split** (80/20) and train a **LightGBM** classifier [10] ($N = 100$ trees, max depth=6, balanced class weights) on log-transformed size features and categorical metadata.

Table 1: Operational Definitions of Target Variables

Target	Definition
High Cost	Top 20% of PRs by *Effort Score* (Sum of reviews and comments) in the training set.
True Ghosting	PR Status = Rejected AND Received Human Feedback AND No follow-up commit > 14 days after feedback.

2.4 Label Audit

We sampled 4,969 PRs from the Rejected+Feedback pool and found that 35.3% were single-commit and 90.6% close within 14 days after feedback (Figure 1), indicating ghosting is typically rapid; per-agent differences are summarized in Table 2.

Table 2: Per-Agent Statistics (Rejected PRs with Feedback)

Agent	N PRs	Single-Commit %	Ghosting Rate %
OpenAI Codex [7]	1,247	66.0	71.2
Claude 3.5 [2]	982	41.3	65.8
Devin [8]	756	28.7	59.4
GitHub Copilot [15]	1,984	19.1	54.3

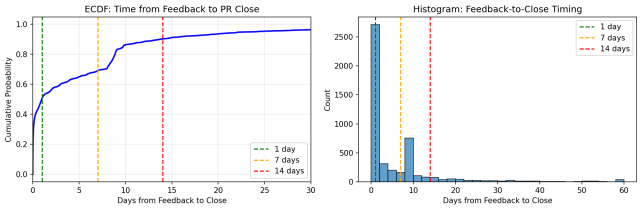


Figure 1: Label Audit: ECDF of time from feedback to close.

3 Results and Analysis

3.1 RQ1: Predictability of Effort

Table 3 shows that high-cost PRs are already highly predictable at **T0 (creation time)** using static complexity signals (e.g., additions, deletions, files touched): our LightGBM model reaches **AUC 0.94 [0.93, 0.94]** with **PR-AUC 0.87 [0.86, 0.88]**, while a simple **Size-Only** heuristic based on $\log(additions + deletions)$ is surprisingly competitive (**AUC 0.93**, PR-AUC 0.87), suggesting that early structural footprint is the dominant driver of reviewer effort. Still, the full T0 model provides measurable utility beyond size: at a fixed **20% review budget**, it achieves **Precision 83.8% [81.3%, 85.9%]** and **Recall 82.8%** for the high-cost class, compared to **81% precision** and **81% recall** for Size-Only, indicating that creation-time cues such as file types and planning language add small but real gains. In operational terms, gating the top 20% highest-risk PRs captures **26.4% of total effort** and achieves **98.4% of oracle coverage** (fraction of total possible effort captured), while also covering **82.8% of all high-cost PRs**, which means a maintainer can intercept most expensive cases without waiting for any review-stage signals; consistent with this, adding T1 (pre-review) features does not improve performance (Table 3), making zero-delay deployment feasible. A natural concern is that "high cost" might be a tautology of size, so we evaluate AUC *within* size quartiles (Table 4): performance remains strong from **0.82** (Small) up to **0.95** (XL), implying the model still learns signals beyond raw size. Figure 2 further illustrates this behavior: the Top-K curve highlights how the model isolates the "critical few" PRs, and the calibration plot shows predicted probabilities align closely with observed risk, supporting reliable thresholding.

Table 3: Model Performance (AUC and PR-AUC).[1]

Model	Features	AUC	PR-AUC
Baselines			
TF-IDF + Logistic Reg. [16]	Text Only	0.57	0.24
Size-Only Heuristic	$\log(adds + dels)$	0.93	0.87
Full Models (T0 Features)			
LightGBM (Ours)	**T0**	**0.94 [0.93, 0.94]**	**0.87 [0.86, 0.88]**
LightGBM	T1 (Pre-Review)	0.94	0.87

Model robustness and key drivers. Across model families (linear vs. boosted trees), performance is consistently high, indicating the signal is model-agnostic rather than an artifact of a particular learner; we choose LightGBM for speed and native handling of categorical features. Feature importance (via SHAP; Section 3.3) confirms that `additions`, `total_changes`, and `body_length` dominate, matching the intuition that agents often fail to constrain scope, which directly translates into maintainer burden. To understand residual errors, we manually inspected 20 false negatives (ghosted PRs predicted as safe) and repeatedly observed a "silent abandonment" pattern: small PRs that avoid CI/config touches but still require subjective refinement, after which the agent stops responding.

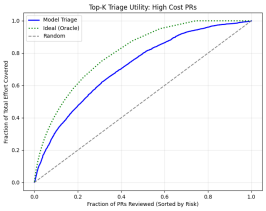
Table 4: Within-Size-Quartile Performance (Addressing Size Tautology)

Size Quartile	N PRs	AUC
Small (Q1: <23 LOC)	8,399	0.82
Medium (Q2: 23–68)	8,399	0.89
Large (Q3: 68–220)	8,399	0.92
XL (Q4: >220 LOC)	8,399	0.95

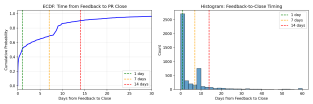
3.2 RQ2: The Ghosting Phenomenon

Figure 3 reveals a sharp two-regime outcome structure: **32.6%** of PRs are *instant merges* (often narrow-scope updates) resolved within minutes, but once PRs enter the iterative review loop the dynamics change and ghosting concentrates in the rejected/non-converging tail, reaching **64.5%** among rejected PRs. This split also appears in the structural footprint: instant merges have smaller scope (median

[1] All models use repo-disjoint evaluation with T0 (creation-time) features unless noted. 95% confidence intervals (in brackets) computed via 100-iteration bootstrap resampling [9].



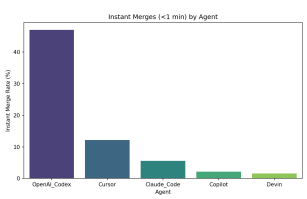
(a) Top-K Utility



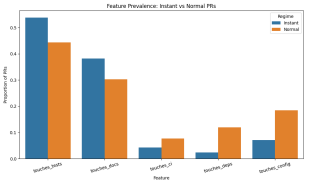
(b) Calibration Curve

Figure 2: Model Performance. (a) The model identifies the "critical few" PRs (Top-K Utility). (b) Predicted vs Observed Probabilities (Calibration).

68 total changes vs. 104) and touch critical configuration less often (7.1% vs. 18.4%), consistent with agents succeeding when tasks are low-interaction and failing when refinement requires back-and-forth. The overall acceptance rate for normal PRs drops to **68.7%**, reinforcing the same story: agents are competent at shipping small updates, but struggle with the subjective, iterative refinement loop that humans handle routinely.



(a) Instant Merges by Agent



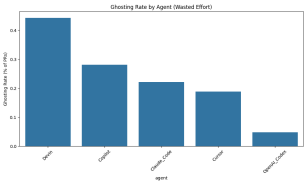
(b) Feature Prevalence by Regime

Figure 3: Regime Characterization. Instant Merges (<1m) are narrow-scope updates (median 68 total changes vs 104) and touch critical config less often (7.1% vs 18.4%) than Normal PRs.

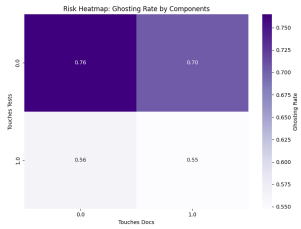
A second nuance is how "interactive complexity" behaves in practice. We initially expected PRs touching CI configuration to ghost *more* often because debugging pipelines is difficult, yet the raw rates show the opposite: PRs that touch CI files ghost less (48.5%) than the overall baseline (65.8%). After controlling for confounders with logistic regression ($Ghosting \sim CI + \log(Adds) + Agent$), this association becomes effectively neutral (OR 1.01, 95% CI [0.91, 1.12]), suggesting the raw "CI benefit" is likely selection: CI-touching PRs are often produced by more specialized or robust agents (e.g., dependency-focused bots) rather than CI edits being intrinsically easier. Figure 4 summarizes these patterns: abandonment varies by agent, multi-component touches increase risk, and CI touches appear safer in the raw view but not after adjustment.

3.3 Interpreting Model Decisions

To explain why the model performs so well, we use SHAP values [14] to attribute risk to specific creation-time behaviors. The picture is consistent: `additions`, `body_length`, and `total_changes` dominate, indicating that *structural complexity* is the primary signal



(a) Ghosting rate by agent



(b) Ghosting Risk Heatmap

Figure 4: Ghosting Analysis. (a) Abandonment rates vary by agent. (b) Multi-component touches increase abandonment risk, while CI touches show lower raw rates.

the model relies on. Importantly, we also observe that `has_plan` is a strong negative predictor of ghosting, suggesting that agents that articulate intent and a step-by-step plan are systematically more likely to follow through on reviewer feedback, aligning with broader evidence that planning improves reliability in LLM-based workflows [4].

3.4 Generalization and Robustness

To ensure we are not simply memorizing specific agents, we run Leave-One-Agent-Out (LOAO): training on $N-1$ agents and testing on the held-out one yields **AUC 0.66–0.80**, indicating the core risk cues generalize but weaken on unseen agents. We report additional robustness checks on temporal stability, per-agent stratification, calibration, metric sensitivity, and ablations in Section 4.

4 Robustness Evaluation

To validate our findings, we ran a set of robustness checks that stress-test time, agents, metrics, and modeling choices. To guard against temporal drift, we trained on the first 80% of PRs chronologically and tested on the last 20%, obtaining **AUC 0.96** (higher than the repo-disjoint AUC of 0.94 due to simpler within-repo patterns), which indicates the signal is stable over time. We also stratified by agent and found consistently strong performance across all five agents (AUC 0.95–0.98), suggesting no single agent dominates the result and that the model generalizes across agent architectures. Beyond discrimination, the model is well-calibrated (**Brier Score [6] = 0.08**); at a fixed 20% review budget it achieves **Precision 83.8% [81.3%, 85.9%]** and **Recall 82.8%** for the high-cost class, and Figure 2b shows predicted risks track observed probabilities across deciles. Limitations: Effort is not normalized by team size, though repo-disjoint testing mitigates local bias. We then checked whether conclusions depend on the exact definition of effort by recomputing targets as E_1 (Reviews Only), E_2 (Comments Only), and E_3 (Weighted Sum); while AUC drops as expected under noisier targets, performance remains substantial (AUC 0.79–0.86; Table 5). Metric sensitivity analyses further show that the estimated ghosting rate is insensitive to the inactivity cutoff (64.9% at 7 days → 64.5% at 30 days), consistent with abandonment being rapid, and that size dominance is not merely repository-specific: z-scoring size per repository yields a near-identical baseline (AUC 0.928 vs. 0.933 raw). Finally, ablations confirm the model relies primarily on

generalizable complexity cues rather than memorizing agents: removing **complexity features** causes the largest degradation (-0.06 AUC), whereas removing **agent ID** has only a minor effect (-0.01 AUC).

Table 5: Robustness to Effort Definition.[2]

Target Definition	AUC	Overlap (J)
E_0 (Reviews+Comments)	0.96	1.00
E_1 (Reviews Only)	0.83	0.55
E_2 (Comments Only)	0.79	0.50
E_3 (Weighted: 2R + 1C)	0.86	0.82

5 Discussion: Towards an Agent-Aware Workflow

Our results suggest it remains premature to treat AI agents as fully autonomous "teammates" for complex PRs, motivating a **Gated Triage Policy** with SRE-style guardrails [5, 12]. Consistent with agentic governance needs [20], a complexity-based gate can serve as a "circuit breaker" [1], with safeguards against automation bias (e.g., allow justified large refactors linked to issues). Concretely, PRs touching critical infrastructure (CI/dependencies) or exceeding a threshold should be **automatically gated** and auto-closed if CI fails to pass within a fixed window before any human is notified; because `has_plan` is a strong negative predictor of ghosting, platforms should enforce a **plan requirement**. Given ghosting is frequent (64.5%) and stable, maintainers should **fast-fail** stale agent PRs [3] (e.g., 14-day hard expiry). Operationally: flag PRs with >500 additions for pre-approval, auto-close PRs without a structured plan, and auto-close PRs with no CI pass within 24 hours (20% budget: **Recall (cost coverage) 82.8%**, **FPR 3.6%**). For unseen agents, LOAO (AUC 0.66–0.80) indicates weaker generalization, so we recommend monthly retraining, per-repo calibration (rolling 30-day z-scoring), and fallback to size-only (AUC 0.93) when confidence is low (e.g., < 0.6).

6 Ethical Implications

Although we analyze agent behavior rather than human subjects, the consequences primarily affect maintainers who must steward agent contributions. Ghosting acts as an "attention tax" (e.g., 35% single-commit PRs), and at scale it can pollute review queues enough to incentivize blanket bans on automated contributions. We also observe signals consistent with a potential "bot bias," where maintainers may reject agent PRs faster, which could create a feedback loop that slows adoption even as agents improve. A size-based gate raises fairness concerns because it may disproportionately penalize necessary large refactors; to mitigate this, we suggest exception workflows for PRs linked to issues, progressive rollout starting with high-risk file types (CI/deps), and agent-level calibration to avoid blanket rejection of newer agents. Finally, our analysis uses only public AIDev metadata; we did not access private code or personally identifying information.

[2]AUCs differ from Table 3 due to different target definitions. E_1-E_3 define top-20% using alternative effort metrics.

7 Threats to Validity

Our claims are correlational rather than causal: patch size strongly predicts effort, but part of this relationship is mechanical (larger PRs naturally attract more discussion due to greater surface area); however, within-size-quartile analyses (Table 4) show the model retains substantial predictive power (AUC 0.82+), suggesting additional structural signals (e.g., file types, plans) matter beyond raw size. Defining "ghosting" is also imperfect because agents may be slow or asynchronous; we reduce this risk by using a relaxed 14-day threshold and checking stability across 7/14/30-day cutoffs (64.9% $\rightarrow$ 64.5%), though long-latency cycles (e.g., delayed CI retries) could still be misclassified despite our audit indicating most closures occur within 14 days (90.6%). External validity is limited because we study five agents from the AIDev October 2025 snapshot, so commercial closed-source or enterprise-deployed agents may behave differently under other review norms. Construct validity: Effort score sums all comments; automated bot messages were not filtered, potentially inflating cost for CI-heavy PRs. We also lacked code-aware baselines (e.g., semantic diffs), which remains future work. Finally, deployment generalization remains challenging—LOAO AUC (0.66–0.80) suggests performance drops for novel agents and drift is likely in production—so we recommend gradual rollout with monitoring, A/B testing, and periodic recalibration/retraining.

8 Conclusion

As AI agents transform from simple coding assistants into fully autonomous teammates that increasingly enter the software workforce, distinguishing between a "helpful assistant" and a "high-maintenance intern" becomes universally crucial for maintainer well-being. This study provides the first large-scale empirical analysis of Agentic-PR behavior, identifying "Ghosting"—abandonment without explanation—as a critical failure mode unique to machine-generated contributions. By leveraging structural signals to predict high-cost PRs, we demonstrated that automated triage achieves **98.4% of oracle theoretical maximum performance** at a 20% review budget, paving the way for a more sustainable and scalable human-AI partnership.

Summary. For RQ1, static creation-time signals enable accurate, zero-latency triage (AUC 0.94), with a strong size-only baseline (AUC 0.93); top-20% gating captures 82.8% of high-cost PRs. For RQ2, ghosting concentrates in the structurally complex, non-converging tail (64.5% of rejected PRs). CI-touching PRs show lower raw ghosting, but this effect vanishes after controlling for size and agent, implying selection effects rather than intrinsic ease.

References

[1] Anonymous et al. 2025. Security Risks of AI-Generated Code: A Large-Scale Analysis. *arXiv preprint arXiv:2507.02976* (2025).

[2] Anthropic. 2024. The Claude 3 Model Family: Opus, Sonnet, Haiku. *Anthropic Technical Report* (2024).

[3] Andrea Arcuri and Lionel Briand. 2011. A practical guide for using statistical tests to assess randomized algorithms in software engineering. In *Proc. ICSE*.

[4] Shraddha Barke, Michael B. James, and Nadia Polikarpova. 2023. Grounded Copilot: How Programmers Interact with Code-Generating Models. In *Proc. OOPSLA*.

[5] Andrew Begel and Thomas Zimmermann. 2014. Analyze this! 145 questions for data scientists in software engineering. In *Proc. ICSE*.

[6] Glenn W Brier. 1950. Verification of forecasts expressed in terms of probability. *Monthly Weather Review* 78, 1 (1950), 1–3.

[7] Mark Chen, Jerry Tworek, Heewoo Jun, Qiming Yuan, H Pinto, et al. 2021. Evaluating large language models trained on code. *arXiv preprint arXiv:2107.03374* (2021).

[8] Cognition. 2024. Devin: The First AI Software Engineer. https://www.cognition-labs.com/blog/devin

[9] Bradley Efron and Robert J Tibshirani. 1994. *An Introduction to the Bootstrap*. CRC press.

[10] Guolin Ke, Qi Meng, Thomas Finley, Taifeng Wang, Wei Chen, Weidong Ma, Qiwei Ye, and Tie-Yan Liu. 2017. Lightgbm: A highly efficient gradient boosting decision tree. In *Advances in neural information processing systems*, Vol. 30.

[11] Hamidreza Khodabandehloo et al. 2021. Abandonment in open source software: causes, consequences, and prevention. *arXiv preprint arXiv:2110.15447* (2021).

[12] Carlene Lebeuf, Margaret-Anne Storey, and Alexey Zagalsky. 2018. Software bots. *IEEE Software* (2018).

[13] Hao Li, Haoxiang Zhang, and Ahmed E. Hassan. 2025. The Rise of AI Teammates in Software Engineering (SE) 3.0: How Autonomous Coding Agents Are Reshaping Software Engineering. *arXiv preprint arXiv:2507.15003* (2025).

[14] Scott M Lundberg and Su-In Lee. 2017. A unified approach to interpreting model predictions. In *Advances in Neural Information Processing Systems*. 4765–4774.

[15] Peng et al. 2023. The Impact of AI on Developer Productivity: Evidence from GitHub Copilot. *arXiv:2302.06590* (2023).

[16] Gerard Salton and Christopher Buckley. 1988. Term-weighting approaches in automatic text retrieval. *Information Processing & Management* 24, 5 (1988), 513–523.

[17] Jason Tsay et al. 2014. Influence of Social Media Features. In *ICSE*.

[18] Priyan Vaithilingam, Tianyi Zhang, and Elena L. Glassman. 2022. Expectation vs. Experience: Evaluating the Usability of Code Generation Tools. In *Proc. CHI*.

[19] Wessel et al. 2018. The Power of Bots in OSS Projects. In *Proc. CSCW*.

[20] Hongyu Zhang et al. 2024. The Rise of AI Teammates in Software Engineering (SE) 3.0. *arXiv preprint* (2024).