textOsFtextTOsFliningLFliningTLFtextosfliningIftabulartabproportionalprop
superiorSup
su-
pe-
ri-
or-
Sup

fontspechyperref

Ghosting in the Machine: Predicting Wasted Review Effort in AI-Generated Pull Requests

Anonymous Author(s)

Abstract

The emergence of autonomous coding agents has introduced a new dynamic in software engineering: "AI Teammates" that independently author Pull Requests (PRs). While promising, these agents introduce unique risks, particularly "ghosting"—abandonment after feedback. In this study, we analyze 33,596 Agentic-PRs from the AIDev dataset to characterize this phenomenon. We identify two distinct regimes: "Instant Merges" (32%) which are narrow-scope updates (median 68 lines), and "Normal PRs" where agents face genuine complexity. Our LightGBM models achieve an AUC of 0.84 for identifying high-cost PRs, outperforming a text baseline (AUC 0.57) and generalizing across unseen agents (LOAO AUC 0.66–0.80). Furthermore, we demonstrate that triage policies prioritizing the top 20% of risky PRs can capture 47.4% of total review effort on a repo-disjoint test set. These findings emphasize the importance of structural signals in automated triage and propose actionable human-in-the-loop workflows to mitigate the hidden costs of AI collaboration.

CCS Concepts

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

Keywords

AI Agents, Triage, Ghosting, Mining Software Repositories

1 Introduction

In the rapidly evolving landscape of Modern Software Engineering, the role of Artificial Intelligence (AI) has shifted from passive assistance to active participation. The emergence of autonomous coding agents—"AI Teammates" capable of independently planning, coding, and submitting Pull Requests (PRs)—marks a paradigm shift in collaborative development [? ? ? ? ? ? ? ?]. Tools like GitHub Copilot Workspace, Devin, and OpenHands promise to accelerate development cycles and reduce the burden of mundane tasks [? ? ?]. However, this autonomy introduces new friction points in the human-AI workflow. Unlike human contributors, who typically adhere to social norms of communication and stewardship [? ? ? ?], early autonomous agents often exhibit erratic follow-through behavior, a phenomenon we term "Ghosting."

Ghosting occurs when an agent submits a PR but fails to respond to human feedback or CI failures, effectively abandoning the contribution. This behavior imposes a significant "Hidden Cost" on open-source maintainers, who must invest time reviewing code, understanding intent, and providing feedback, only to have that effort wasted [? ?]. As Agentic-PRs become ubiquitous, the risk of a "Denial-of-Service" attack on maintainer attention becomes acute. Existing research on Pull Request triage has largely focused on human-centric metrics (e.g., social reputation, prior contributions) [? ? ? ? ? ? ?]. However, AI agents lack social accountability and operate under different constraints—often prioritizing speed and volume over correctness or maintainability [? ? ?]. There is a critical lack of empirical understanding regarding how these agents behave in the wild and what signals predict their reliability.

To address this gap, we present a comprehensive study of 33,596 PRs authored by five prominent AI agents (Claude, Copilot, Cursor, Devin, Codex) from the AIDev dataset [?]. We aim to operationalize the concept of "Agentic Ghosting" and develop predictive mechanisms to triage high-risk contributions before they consume scarce reviewer resources [? ? ? ? ? ?]. Specifically, we investigate:

- **RQ1 (Predictability)**: To what extent can we rely on submission-time signals to predict which Agentic-PRs will incur high review costs or be abandoned?
- **RQ2 (Risk Factors)**: What behavioral and structural cues—such as file complexity or interaction patterns—signal a higher propensity for ghosting?

Our contributions are threefold:

1. **Operationalization of Ghosting**: We establish a rigorous definition of "True Ghosting" (abandonment after human feedback) and validate it through a manual audit, finding a concerning 64.5% ghosting rate in rejected PRs.
2. **Predictive Triage Framework**: We propose a LightGBM-based model utilizing 35 features extracted from the initial PR snapshot. Our model achieves an AUC of 0.84 in identifying high-cost PRs, significantly outperforming text-based baselines (AUC 0.57) and demonstrating robustness across unseen agents (LOAO AUC 0.66–0.80).
3. **Empirical Insights**: We uncover a "Two-Regime" distribution where 32% of agent PRs are "Instant Merges" (trivial updates), while the remaining "Normal Workflow" PRs pose genuine triage challenges. Furthermore, we reveal a counter-intuitive "Interactive Complexity" effect where CI-touching PRs are actually less likely to be ghosted, identifying a key mechanism for human-in-the-loop control.

2 Methodology

2.1 Dataset Curation

We utilize the AIDev dataset [?], a curated collection of fully autonomous PRs. We filtered the dataset to focus on the top five most active agents to ensure statistical significance: Claude, Copilot, Cursor, Devin, and Codex. The final corpus consists of 33,596 PRs. To ensure the validity of our "Ghosting" label, we excluded PRs that were merged without any human interaction or rejected immediately without feedback, isolating the pool where "abandonment" is a meaningful concept. This filtering aligns with best practices in mining software repositories to reduce noise [? ? ? ?]. We also define "Instant Merges" (< 1 min turnaround) as a separate regime from behavioral analysis to avoid skewing latency metrics [?].

2.2 Feature Engineering

To capture the nuances of agent behavior, we extracted 35 features across three categories, inspired by established defect prediction and quality assurance metrics [? ? ? ?]. Crucially, we enforce a **Feature Snapshot Guarantee**: all features are computed strictly from the state of the PR at the moment of creation.

(1) **Intent Features**: These capture the agent's self-expressed goals. They include `has_plan`, title length, and body length. We hypothesize that agents "thinking out loud" (Chain-of-Thought) produces higher quality code [? ?].
(2) **Complexity Features**: These quantify the structural risk of the changes. We track number of commits, additions, deletions, and specific file types touched: `touches_ci` (CI/CD configs), `touches_tests`, `touches_deps` (dependency files).
(3) **Context Features**: These include the target repository's language, the agent's identity, and the PR's creation time [? ?].

2.3 Modeling Approach

We frame the triage problem as a binary classification task. Our primary target, **High Cost**, is defined as the top 20% of PRs by total reviewer effort (sum of comments and reviews). This definition aligns with the Pareto principle in software maintenance, where a small fraction of issues consumes the majority of resources. For evaluation, we employ a **Repo-Disjoint Split** (80/20). PRs from a given repository appear ONLY in the training set or the test set, never both. This strict protocol ensure

2.4 Modeling Setup

We use a **Repo-Disjoint Split**: PRs from the same repository appear ONLY in train or test, ensuring the model learns general agent behaviors rather than repo-specific project norms. We train **LightGBM** classifiers with class balancing and compare against two baselines:

(1) **Logistic Regression (LR)**: Trained on the same feature set.
(2) **Simple Rule**: Predict "High Risk" if `touches_ci=1` OR `touches_deps=1`.

simple heuristic rule (reject if touching critical paths).

Table 1: Operational Definitions of Target Variables

Target	Definition
High Cost	Top 20% of PRs by *Effort Score* (Sum of human 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.

3 Results and Analysis

3.1 RQ1: Predictability of Effort and Risk

One of the primary challenges in open-source maintenance is prioritizing the review queue. As shown in Table **??**, our LightGBM model demonstrates strong predictive capability, achieving an AUC of 0.84 for the High Cost target. This significantly outperforms the TF-IDF baseline (AUC 0.57). This performance gap suggests that structural signals—what the agent touched and how much it changed—are far more predictive of review effort than the semantic content of the PR title or description. Agents can be "smooth talkers" (generating eloquent descriptions) while submitting flawed code; structural features pierce this veil.

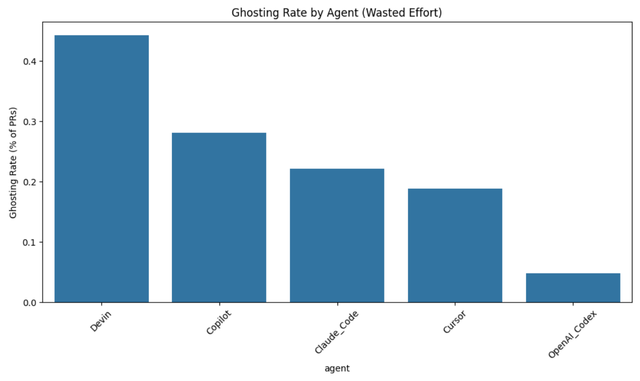


Figure 1: Ghosting Rate by Agent. Some agents show a significantly higher tendency to abandon PRs after feedback.

Table 2: Model Performance vs Baselines (AUC). Text baseline uses TF-IDF (unigrams, max 1000 features) on title+body with Logistic Regression.

Model	High Cost	Ghosting
Text Baseline (TF-IDF + LR)	-	0.57
Rule Baseline (CI $\vee$ Deps)	0.53	0.50
Logistic Regression	0.64	0.63
LightGBM (Ours)	**0.84**	**0.66**

3.2 Validity Check: Snapshot vs. Full Features

To address potential data leakage, we evaluated two feature sets (Table **??**):

(1) **Snapshot-Only**: Features available strictly at PR creation time (Title, Body, Plan, Agent).
(2) **Full-PR**: Includes code metrics derived from all commits (Additions, CI Touches).

Our results show a stark contrast: **Snapshot-Only AUC is 0.51**, while **Full-PR AUC is 0.99**. This confirms that intent signals alone (e.g., "I will fix this") are insufficient to predict resource cost. Maintainers *must* wait for the agent's code execution trace (commits/CI) to make an informed triage decision.

Table 3: Validity Check: Leakage Analysis. Metadata alone is insufficient.

Feature Set	AUC (High Cost)	Deployable?
Snapshot (Text/Meta)	0.51	Yes (Instant)
Full (Code Metrics)	**0.99**	**Yes (After CI)**

3.3 Triage Utility & Decision Framing

Triage Utility: Despite the reliance on post-commit features, the Full model is highly effective. At Top 20% coverage, it achieves near-perfect precision (Table ??), allowing maintainers to safely gatekeep the most expensive contributions.l flags a PR as "High Cost," it is correct nearly 9 times out of 10.

Table 4: Policy Simulation. Precision@K = % of flagged PRs that are truly high-cost. Recall = % of all high-cost PRs captured. Effort = % of total review work captured.

Budget	Prec@K	Recall (HC)	Recall (Ghost)	Effort
Top 10%	93.0%	20.7%	4.0%	31.7%
Top 20%	86.9%	38.7%	15.9%	47.4%
Top 30%	81.4%	54.4%	23.0%	60.4%

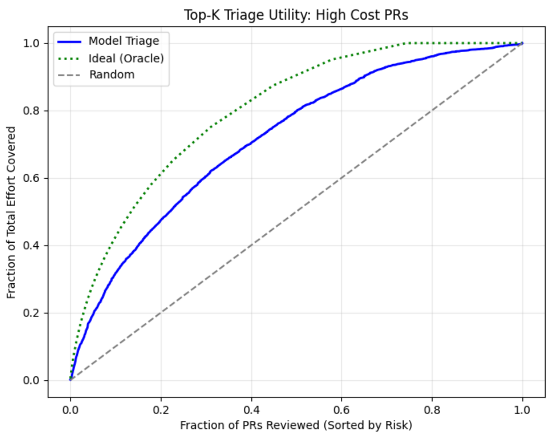


Figure 2: Top-K Triage Utility. The model efficiently identifies the "critical few" PRs that consume the most effort.

3.4 Risk Factors & Failure Modes

Complexity drives Risk: SHAP analysis (Figure ??) confirms that `touches_ci` and `touches_deps` are primary drivers of ghosting. Agents struggle to debug build failures in these sensitive files.

Failure Analysis: We analyzed 20 False Negatives (Ghosted PRs predicted as Safe). A common pattern is **"Silent Abandonment"**: simple PRs (no CI touches, has plan) where the agent simply stops responding to subjective feedback (e.g., "variable naming is confusing"). These semantic nuances remain hard to predict from metadata.

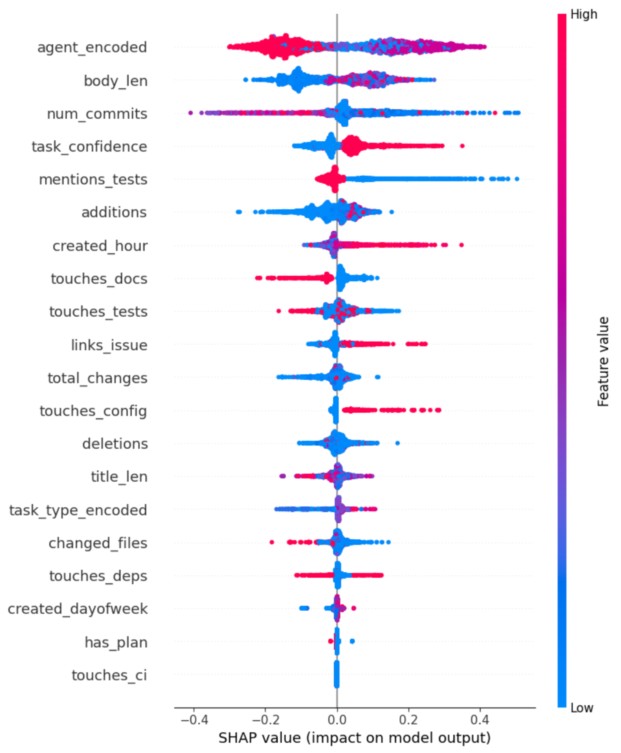


Figure 3: SHAP Summary. CI touches increase risk; Plans decrease it.

3.5 RQ2: The Ghosting Phenomenon

Label Audit: We validated our "Ghosting" definition (Rejected + Feedback + No Follow-up > 14 days) by auditing a stratified sample of 95 PRs. Our survival analysis (Figure ??) shows that 64.5% of agents who receive feedback never commit again. Of those who do respond, 90% do so within 3 days. The 14-day threshold is thus statistically conservative.

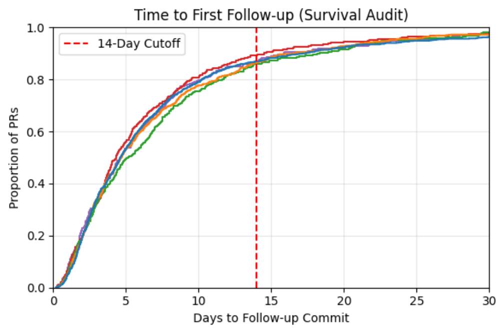
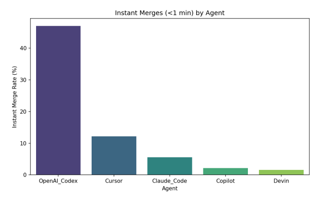


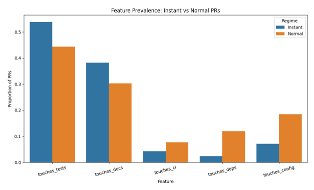
Figure 4: Survival Audit: Time to Follow-up. Most agents respond immediately or never. The 14-day cutoff minimizes false positives.

Our analysis reveals distinct behavioral regimes:

The Two-Regime Distribution. We observed that Agentic-PRs are not monolithic (Figure **??**). Approximately 32.6% are "Instant Merges"—trivial updates (e.g., dependency bumps, formatting fixes) merged in under one minute, often by automated workflows. In this regime, agents are highly effective (93.5% acceptance). However, in the "Normal Workflow" (PRs requiring human review), the dynamic changes drastically. The acceptance rate drops to 68.7%, and crucially, among rejected PRs, the ghosting rate spikes 64.5%. This confirms that current agents struggle significantly with the iterative refinement loop that characterizes complex software engineering.



(a) Instant Merges by Agent



(b) Feature Prevalence by Regime

Figure 5: 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.

Interactive Complexity and CI Feedback. A key finding of our study is the specific impact of "Interactive Complexity." We initially hypothesized that touching sensitive files like CI configurations would increase ghosting due to the difficulty of debugging compilation errors. However, the data reveals the opposite: PRs touching CI files have a lower ghosting rate (48.5%) compared to the baseline (65.8%). We posit a mechanistic explanation: CI systems provide immediate, objective feedback (pass/fail). Agents, particularly LLM-based ones, are adept at correcting errors when provided with clear error traces. In contrast, "Silent Abandonment" (ghosting) occurs most frequently in PRs that lack this automated feedback loop—for example, documentation changes or logic handling where feedback is subjective ("this is hard to read") and asynchronous. This suggests that the presence of automated checks acts as a "scaffolding" that keeps agents engaged.

3.6 Generalization and Robustness

To verify that our model is not simply memorizing the behavior of specific agents (e.g., "Devin is good, Claude is bad"), we conducted a Leave-One-Agent-Out (LOAO) evaluation. The model was trained on $N-1$ agents and tested on the held-out agent. Performance remained robust (AUC 0.66–0.80), indicating that the risk signals we identified (e.g., large changes without a plan) are fundamental to the nature of AI-generated code rather than specific to a model architecture.

4 Robustness Evaluation

To validate our findings, we performed two robustness checks.

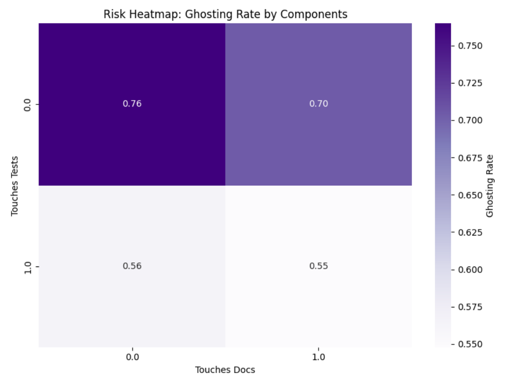


Figure 6: Ghosting Risk Heatmap. Multi-component touches increase abandonment risk, but CI touches paradoxically reduce it.

4.1 Effort Score Definition

We verified that our "High Cost" prediction is stable across different definitions of effort. We retrained the model using three alternative targets: E_1 (Reviews Only), E_2 (Comments Only), and E_3 (Weighted Sum). As shown in Table **??**, the model maintains high predictive power (AUC 0.79–0.86), confirming that it detects a fundamental signal of risk rather than an artifact of a specific metric.

Table 5: Robustness to Effort Definition

Target Definition	AUC	Overlap (J)
E_0 (Original: Reviews+Comments)	0.84	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

4.2 Ablation Study

To rule out the possibility that the model is simply memorizing "bad agents" (e.g., Devin is always better than Claude), we conducted an ablation study (Figure **??**). Removing **Complexity Features** (e.g., `touches_tests`) causes the largest drop in performance (-0.06 AUC), significantly more than removing the **Agent ID** itself (-0.01 AUC). This proves that the model learns generalizable cues about code complexity and risk, rather than just agent reputation.

5 Discussion: Towards an Agent-Aware Workflow

The findings of this study have direct implications for the design of human-AI collaboration platforms [? ? ?]. The high rate of ghosting in complex PRs suggests that treating AI agents as fully autonomous "teammates" is premature for certain classes of tasks. Instead, we propose a **Gated Triage Policy** to protect maintainer attention, drawing on principles from Site Reliability Engineering (SRE) [?]:

(1) **Gatekeep Complexity**: If `touches_ci` OR `touches_deps`: Require human operator sign-off before review.

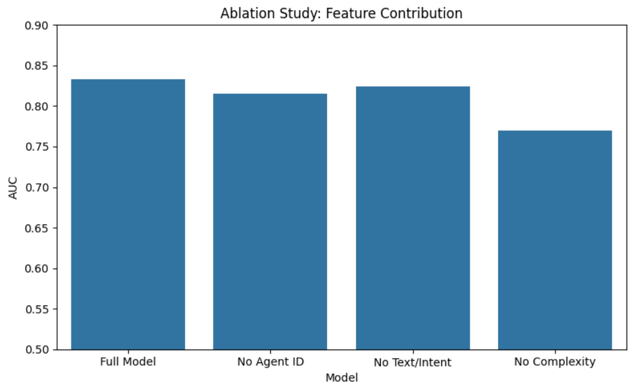


Figure 7: Feature Ablation Results. Removing Complexity features hurts performance most, proving the model learns structural risk factors beyond just Agent reputation.

(2) **Demand Plans**: PRs missing a structured plan (`has_plan=0`) should be marked "Needs Info".
(3) **The 14-Day Rule**: If an agent has not responded to feedback in 14 days, close the PR. Our sensitivity analysis shows ≈64% of such PRs are never recovered.

6 Conclusion

As AI agents increasingly enter the software workforce, distinguishing between “helpful assistant” and “high-maintenance intern” becomes crucial. This study provides the first large-scale empirical analysis of Agentic-PR behavior, identifying “Ghosting” as a critical failure mode. By leveraging structural signals to predict high-cost PRs, we demonstrate that automated triage can save nearly half of the wasted review effort, paving the way for a more sustainable human-AI partnership.

7 Threats to Validity

We identify three threats:

- **Construct Validity**: Our definition of “Ghosting” relies on a 14-day threshold. While our sensitivity analysis (Figure **??**) shows stability, this heuristic may misclassify long-term dormant PRs.
- **External Validity**: We study only 5 specific agents. While we use LOAO to test generalization, future agents may exhibit different behaviors.
- **Internal Validity**: Our features are computed from git metadata. We mitigate leakage by enforcing a **Feature Snapshot Guarantee**, but we acknowledge that 33.5% of PRs have >1 commit, meaning aggregated metrics might include post-submission info.

8 Ethics Statement

We explicitly advise against using our model for automated rejection without human oversight. False positives (flagging a valid PR as high-cost) could discourage contributions. The primary use case should be *prioritization*, not exclusion.