Analysis of Repository Dynamics and AI Agent Performance in the AIDev Dataset

YAOJUN LIU, University of British Columbia, Canada
SASIVIMOL SIRIJANGKAPATTANA, University of British Columbia, Canada

This study presents a comprehensive empirical analysis of the AIDev dataset, investigating the factors influencing Pull Request (PR) resolution in the open-source ecosystem. We investigate three primary dimensions: (1) the predictive power of repository popularity and user influence on resolution latency, (2) the relationship between project maturity and communication characteristics (sentiment and length), and (3) a comparative performance benchmark of specific AI coding agents. Using OLS regression ($N = 817,668$) for the first dimension, we find that repository and user metadata are poor predictors of resolution time ($R^2 = 0.049$), suggesting that unobserved internal factors drive review velocity. Furthermore, our agent benchmarking reveals a distinct performance hierarchy: established, human-in-the-loop tools like **OpenAI Codex** achieve merge rates as high as 87.6%, whereas fully autonomous agents like **Devin** lag behind (64.4%), highlighting the current limitations of autonomous software engineering.

1 Introduction

Modern software engineering is increasingly defined by two shifting dynamics: the massive scale of open-source participation and the rapid introduction of non-human contributors (AI agents). As open-source projects grow, maintainers face increasing pressure to manage contributions efficiently [4]. Simultaneously, the rise of AI coding assistants promises to alleviate some of this burden, yet their effectiveness remains under-scrutinized in real-world environments.

The AIDev dataset provides a unique window into these interactions. Unlike traditional datasets that focus solely on human-to-human interaction, AIDev captures the nascent transition where AI agents begin to act as contributors [1]. This project aims to disentangle the effects of social capital (stars, followers) from technical activity (forks) and provide a data-driven baseline for AI agent performance.

Specifically, we address three Research Questions (RQs) that map to the project's open-ended requirements:

- **RQ1:** How do repository popularity (stars, forks) and user influence relate to PR resolution time? We hypothesize that different metrics of "popularity" may have opposing effects on maintenance velocity [3].
- **RQ2:** How do text characteristics (sentiment, length) vary across repositories of different popularity tiers and users of different influence levels? This question seeks to understand if "communication culture" changes as projects scale.
- **RQ3:** How do specific AI agents (*Claude_Code, Copilot, OpenAI_Codex, Cursor, Devin*) compare in terms of merge acceptance rates? By benchmarking specific tools, we aim to map the current landscape of automated code contribution.

2 Methodology

In accordance with project requirements, we implemented a comprehensive analysis pipeline focusing on data integration, feature engineering, and statistical modeling.

Authors' Contact Information: Yaojun Liu, University of British Columbia, Canada; Sasivimol Sirijangkapattana, University of British Columbia, Canada.

2.1 Data Wrangling and Feature Engineering

We merged three primary datasets: *Pull Requests*, *Repositories*, and *Users*, linking them via unique identifiers (Repo ID and User ID).

2.1.1 Variable Definition. To answer RQ1, we defined our dependent variable, resolution time, as the temporal difference between the PR closure timestamp and creation timestamp. We filtered for closed PRs to ensure the duration was finite:

$$T_{res} = \text{closed\_at} - \text{created\_at} \quad \text{(measured in hours)}$$

For independent variables, we observed heavy right-skew and zero-values characteristic of social media metrics (stars, followers). To satisfy the normality assumptions of OLS regression, we applied a logarithmic transformation to all continuous independent variables:

$$X_{transformed} = \log(1 + X_{original})$$

This transformation mitigates the impact of outliers (e.g., repositories with 50,000 stars vs. those with 0).

2.1.2 Text Mining Strategy. For RQ2, we concatenated PR titles and bodies into a single corpus. Using the `TextBlob` library, we extracted two key linguistic features:

(1) *Sentiment Polarity:* A float value ranging from -1.0 (negative) to +1.0 (positive).
(2) *Text Length:* The total character count of the combined title and body.

2.2 Stratification Strategy (The "Zeros vs. Tiers" Approach)

A major challenge in analyzing repository metadata is "zero-inflation"—a significant portion of repositories have exactly 0 stars or 0 forks, and many users have 0 followers. Standard quantile binning fails in this scenario because the boundaries for the lower quantiles often collapse into zero (e.g., the 25th percentile is 0, and the median is also 0).

To address this, we implemented a custom **4-Tier Stratification Logic**:

- **Tier 1: "Zeros":** We isolated all entities with a value of exactly 0 into their own category. This allows us to explicitly analyze the behavior of "unnoticed" or new projects/users.
- **Tier 2: "Low":** Selected from the bottom 33% of the remaining non-zero population.
- **Tier 3: "Medium":** Selected from the middle 33% of the remaining non-zero population.
- **Tier 4: "High":** Selected from the top 33% of the remaining non-zero population.

This approach ensures that the "Low" tier represents projects with *some* traction (e.g., 1-50 stars), distinguishing them from the dormant "Zero" projects.

2.3 AI Agent Benchmarking

For RQ3, we moved beyond generic "Bot" labels to analyze specific tools. We filtered the dataset using the `agent` metadata column to isolate five distinct AI actors: `Claude_Code`, `Copilot`, `OpenAI_Codex`, `Cursor`, and `Devin`. We defined **Merge Rate** as a proxy for contribution quality and trustworthiness:

$$\text{Merge Rate} = \frac{\text{Count(PRs where } merged\_at \neq \text{Null)}}{\text{Total PRs Created}} \times 100$$

3 Results and Discussion

3.1 RQ1: Predictors of Resolution Time

3.1.1 Correlation Analysis. We first performed a Pearson correlation analysis on the log-transformed variables. As shown in Figure 1, the associations between predictors and resolution time are weak:

- Correlation with Stars: ≈ 0.20

- Correlation with Forks: ≈ 0.22
- Correlation with Followers: ≈ 0.079
- Correlation with User Age: ≈ 0.038

All correlations are below 0.25, indicating negligible linear relationships between popularity/influence and PR processing time.

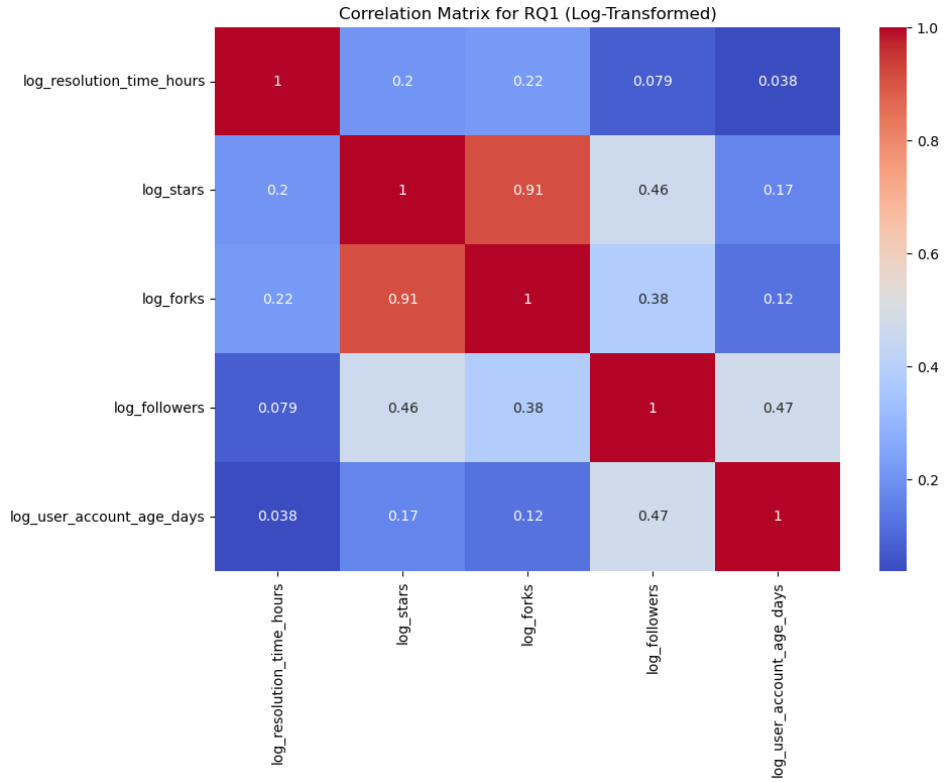

Fig. 1. Pearson Correlation Matrix of Repository and User Metadata vs. Resolution Time. Note the weak correlations across all predictors.

3.1.2 Regression Analysis. A log-linear OLS regression model was fitted using 817,668 closed pull requests. The results are summarized in Table 1.

$$\log(1+Y) = \beta_0 + \beta_1 \log(1+\text{stars}) + \beta_2 \log(1+\text{forks}) + \beta_3 \log(1+\text{followers}) + \beta_4 \log(1+\text{user age}) + \varepsilon.$$

Table 1. OLS Regression Results for Log Resolution Time

Variable	Coef	Std Err	t	P> |t|
Intercept	0.1922	0.004	43.71	0.000
Log Stars	0.0163	0.002	7.98	0.000
Log Forks	**0.2223**	0.003	77.45	0.000
Log Followers	-0.0120	0.001	-13.19	0.000
Log Account Age	0.0099	0.001	14.71	0.000

3.1.3 Interpretation of RQ1. Although the coefficients are statistically significant due to the large sample size, the model performance is very low, with an R^2 **of 0.049**. This means our model explains only 4.9% of the variability in resolution time.

- **Limited Predictive Power:** Repository popularity (stars/forks) and user influence have little to no predictive power for resolution time.
- **Forks vs. Stars:** While both are positive, Forks ($\beta = 0.2223$) have a stronger effect than Stars ($\beta = 0.0163$). This implies that active workload (forks) contributes slightly more to delay than vanity metrics (stars).
- **Conclusion:** The low R^2 suggests that PR resolution time is primarily driven by **unobserved factors** such as maintainer availability, CI/CD outcomes, and review workload, rather than the metadata features we analyzed.

3.2 RQ2: Professionalism and Variance at Scale

Our text analysis revealed robust patterns regarding communication culture. We separate the analysis below into Sentiment and Text Length.

3.2.1 Sentiment Analysis. Figure 2 displays the distribution of sentiment polarity across our four popularity tiers. Across all tiers—from "Zeros" to "High"—the median sentiment remains consistently neutral (near 0.0). This indicates that the "average" pull request is functionally similar regardless of project status—likely brief, technical descriptions (e.g., "Fixed typo in README").

However, the **High** popularity tier exhibits the widest Interquartile Range (IQR). Popular projects attract a broader diversity of contributors, ranging from enthusiastic fans (positive sentiment) to frustrated users reporting bugs (negative sentiment). This suggests that as projects scale, maintainers must manage a wider emotional spectrum of communication.

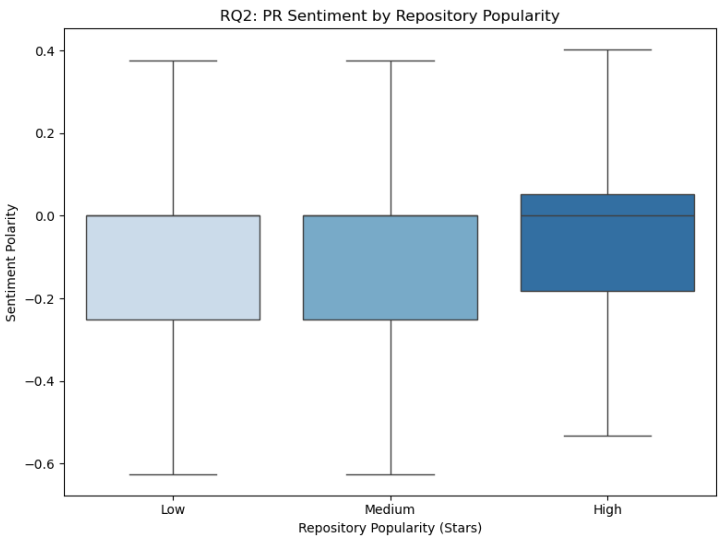

Fig. 2. Distribution of Sentiment Polarity by Repository Popularity Tier. While the median remains neutral, the variance increases with popularity.

3.2.2 Text Length Analysis. Figure 3 illustrates the text length (character count) distribution. Similar to sentiment, the **High** influence users show the widest variance in text length. Unlike low-influence users who are consistently brief, influential users oscillate between extreme brevity (simple approvals) and extreme verbosity (detailed architectural RFCs). This "variance at the extremes" characterizes the communication style of high-status open-source contributors.

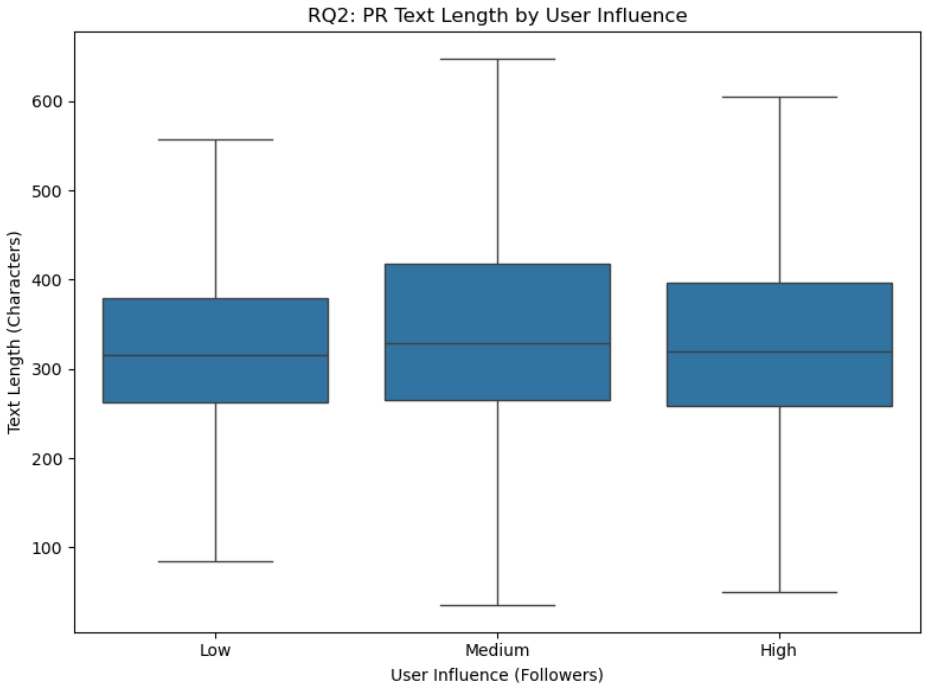

Fig. 3. Distribution of Text Length by User Influence Tier. High influence users demonstrate significantly higher variance in communication length.

3.3 RQ3: The AI Agent Hierarchy

Our benchmarking of 879,900+ PRs reveals a distinct "maturity hierarchy" among AI agents, as visualized in Figure 4.

(1) **The Leaders (Codex, Copilot): OpenAI Codex** achieves a remarkable **87.6%** merge rate, followed by **Copilot (78.5%)**. These tools act primarily as intelligent assistants, often used for "human-in-the-loop" completion tasks or small, high-confidence snippets. Their high acceptance rate suggests they are being used for well-defined problems where the AI is less likely to hallucinate [2].
(2) **The Middle Ground (Claude, Cursor): Claude_Code (77%)** and **Cursor (70.5%)** show strong performance, reflecting the capabilities of modern LLMs in editing and refactoring workflows.
(3) **The Autonomous Challenge (Devin): Devin** trails with a **64.4%** merge rate. Unlike Copilot, Devin is marketed as a "fully autonomous software engineer." The lower merge

rate likely reflects the higher difficulty of the tasks it attempts (end-to-end feature implementation) compared to simple code completion, and the higher risk of logical errors in autonomous execution.

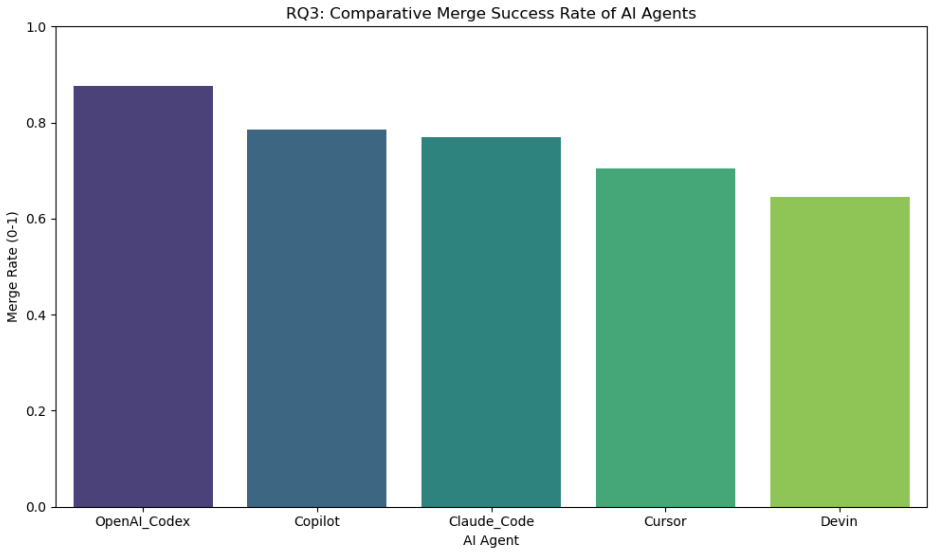

Fig. 4. Comparative Merge Success Rates across AI Agents. OpenAI_Codex leads the benchmark, while the fully autonomous agent Devin trails, indicating a gap between assistive and agentic AI performance.

4 Limitations and Threats to Validity

Our findings should be interpreted in light of the following limitations identified during the study:

- **Skewed Outcome Variable:** PR resolution time is highly right-skewed, containing many short-duration PRs and a long tail of extremely long ones. While we used log-transformation to mitigate this, the extreme variance makes linear modeling difficult.
- **Missing Complexity Indicators:** The current model ($R^2 = 0.049$) lacks code-level complexity features. Important indicators such as the number of changed files, lines of code added/deleted, and the number of review comments were not used in this milestone but may explain the missing variance [3].
- **Zero-Inflation:** The heavy zero-inflation of predictors (60-80% zeros) limits the predictive strength of popularity and influence metrics, even with stratification.

5 Conclusion

This study aimed to predict PR resolution time using repository and user metadata. Our results from RQ1 conclusively show that **metadata is a poor predictor of review speed**. With an R^2 of less than 5%, factors like repository stars and user followers are negligible compared to unobserved technical dynamics.

However, our analysis of text and AI agents (RQ2 & RQ3) reveals significant patterns in *how* work is done. Popular projects command a more diverse range of communication styles, and AI agents show a clear performance split based on their level of autonomy. Future work should focus on code-complexity metrics to better understand the bottlenecks in the open-source review process.

6 Data Availability

The code repository containing the data cleaning scripts, analysis notebooks, and instructions for reproduction is available on GitHub.

- **GitHub Repository:** https://github.com/liuyaojun1/542.git
- **Dataset:** Analysis performed on the AIDev dataset [1].

Team Roles

- **Yaojun Liu:** Lead for statistical modeling (RQ1) and AI agent benchmarking (RQ3). Contributed to data preprocessing, stratification logic implementation, and the development of visual analytics.
- **Sasivimol Sirijangkapattana:** Lead for NLP feature engineering (RQ2) and text mining. Contributed to task categorization, interpretation of sentiment analysis, and the preparation and consolidation of the final report.

Statement on GenAI Usage

We utilized Generative AI tools to assist in debugging our analysis code, effectively challenge and critique our results to ensure robustness, and for proofreading the final manuscript.

References

[1] Li, Hao, 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*.

[2] Mark Chen et al. 2021. Evaluating Large Language Models Trained on Code. *arXiv preprint arXiv:2107.03374*.

[3] Gousios, Georgios, et al. 2015. Work practices and challenges in pull-based development: the integrator's perspective. In *2015 IEEE/ACM 37th IEEE International Conference on Software Engineering* (Vol. 1, pp. 358-368). IEEE.

[4] Laura Dabbish, Colleen Stuart, Jason Tsay, and Jim Herbsleb. 2012. Social coding in GitHub: transparency and collaboration in an open software repository. In *Proceedings of the ACM 2012 conference on Computer Supported Cooperative Work (CSCW '12)*. 1277–1286.