Analyzing Pull Request Acceptance Rates

Baolin Yang
University of Saskatchewan
Canada
nyk750@usask.ca

Thomas Bratvold
University of Saskatchewan
Canada
tnb078@usask.ca

Yuvraj Korotana
University of Saskatchewan
Canada
yvk110@usask.ca

Cjaiyle Alexandria H. Torres
University of Saskatchewan
Canada
cht314@usask.ca

ABSTRACT

In an open-source software development, pull request (PR) is heavily relied on to integrate code change contributions. However, the acceptance rate of PRs varies greatly due to factors such as contributor experience, the complexity of proposed changes, and project-specific review standards. This paper aims to examine the factors that may affect the PR acceptance rate and how understanding these factors can lead to better collaboration between contributors, researchers, and maintainers. It aims to identify the factors that can increase contributor success rates, improve overall project quality, and improve software development workflows. We gather PRs from 20 open-source Github projects where data with contributor-related aspects and PR attributes were extracted and analyzed. Existing literature on PR acceptance rates were also reviewed. Our analysis identified several factors that influence the acceptance of PRs (PRs) in open source projects. The size of the PR, the number of lines of code inserted or deleted, is a determinant of acceptance; short PRs are more likely to be accepted than long ones. This is because PRs that have more review comments are more likely to be accepted and short collaborative contributions are preferred. The acceptance rate decreases for PRs that have been open for more than a year and for those that have been reviewed by more than three people. The biggest repositories with more stars and forks are much pickier (correlations of -0.30 and -0.59). The PRs identified as bug fixes or dependency updates have the highest acceptance rate, while documentation and new feature additions are less desirable. Both the PR structure and the project scale are also found to affect acceptance rate using logistic regression.

KEYWORDS

Pull request attributes, open-source, acceptance rates, contributor experience, review behaviors

I. INTRODUCTION

Pull requests (PRs) are a key concept of collaborative development in open source software that allows developers to submit changes and receive timely feedback from the community [9]. From bug fixes to feature enhancements, these inputs can have a big impact on code quality, contributor retention, and the direction of the project [5]. However, not all PRs are accepted and the rejection rates vary greatly depending on the project. Major platforms such as GitHub offer structured review and discussion features however most of the PRs are getting rejected for various reasons such as lack of testing, poor documentation, or unfamiliarity with the project's guidelines [2], [9]. These rejections present several issues: newcomers may feel discouraged and thus less inclined to contribute [2], and maintainers may get overwhelmed by the number of PRs which in turn can slow down the release processes and negatively impact the community's sentiment [26], [29]. Previous studies show that variables such as contributor's reputation, code complexity and the existence of tests can affect the probability of accepting the PR [9]. However, there is variability in the practice of reviews and in the phases of the project [19], such as release freezes, which may increase the likelihood of scrutiny [2]. Achieving a deeper understanding of these drivers, from contributor specific traits, such as prior merges, to PR level attributes, like lines of code changed and discussion quality, and contextual project policies can support maintainers and contributors in their collaboration [7], and support the researchers in developing automated triage systems [2], [9], [5], [17]. However, existing research on code review automation [17], machine learning prioritization [2], and contributor trust metrics typically involves restricted variables or distinct repositories, thus demanding a holistic perspective that would incorporate several factors that may affect PR acceptance.

The purpose of this analysis is to determine what factors contribute to varying levels of PR acceptance rates. By examining key attributes, patterns may emerge that explain why some PRs are merged and some are don't which will be beneficial for both the contributors and the maintainers in respect of the submissions and reviews [6].

One of the key factors is the level of experience of the contributors [15]. Previous contributors may have higher acceptance rates due to established trust, while new contributors will have more scrutiny. Another factor is the size of the PR, where large changes may need more comprehensive review and thus lead to lower acceptance rates. There is also a possibility of time-related factors such as the time between the creation of the PR and its merging or closing which is also an important factor that may play a role.

The levels of discussion and feedback the PR receives through comments and review interactions may also have an impact on the success of the PR, but it is yet necessary to understand whether high levels of engagement result in approvals or rejections. The presence of the reviewers may also affect the acceptance rates since structured feedback may increase the chances of approval of the PR or may lead to more scrutiny. Similarly, the role of draft PRs in acceptance is not well understood either, in that it is still vital to establish whether they are indeed converted to approved contributions or are left unmerged.

Other factors include repository size and activity, where more active projects may have different PR review patterns, and labels like 'bug' or 'enhancement' which might affect prioritization. Furthermore, the analysis of the PR trends across projects will help to understand whether different projects have their own rules for PR review and approval.

The review behaviours such as the frequency of the comments, the type of comments made, help to understand the role that collaborative feedback plays in the PR processes. Constructive feedback and thorough code reviews contribute to improving the quality of PRs, making them more likely to be accepted. On the other hand, PRs that get little or no interaction may face rejection. Analyzing the review behaviors can help develop the best practices for maintainers and contributors to enhance the PR approval process. [29].

Through the consideration of these variables, this study is to present real trends rather than relying on assumptions. It can therefore be useful for contributors to know these trends when contributing to a project so as to ensure that their submissions are in line with the project's goals and stand a better chance of being accepted. Additionally, knowing these trends can give maintainers an idea on how to optimize their review workflows to balance quality and efficiency.

Understanding PR acceptance rates provides valuable insights into the dynamics of open-source development. In this paper, we will discuss how PR approval factors can be identified to help contributors improve their PR submission tactics or, most importantly, help maintainers improve their review workflows to ensure that PRs are being handled efficiently and in a collaborative manner. This research aims to contribute to a better understanding of how open source communities work and, therefore, to improving the practices of code contribution and project management.

II. BACKGROUND

This section provides an overview of the role of pull requests in open-source development, the key factors influencing their acceptance, the impact of project release cycles, and the broader significance of PR practices for both the open-source community and the software industry. This section

A. *Pull Requests in Open-Source Software*

Open-source software thrives on distributed development, where contributors worldwide collaborate through version control and review platforms like GitHub and GitLab [2], [14] [9]. A typical PR life cycle involves a contributor forking a repository, implementing changes on a branch, and submitting a merge request for review [9]. Maintainers then evaluate the submission based on multiple factors, from code quality to alignment with the project's roadmap [26].

B. *Determinants of PR Acceptance*

Several empirical studies have highlighted key determinants of PR acceptance. Gousios et al. [9] found that contributor reputation and past successful merges boost the likelihood of a PR merge. Azeem et al. [2] underscored the role of prompt maintainer feedback and efficient review processes, while Soares et al. [22] focused on specific rejection patterns in high-acceptance-rate projects.

C. *Project Release Cycles*

Open-source projects typically adopt iterative development cycles, releasing major versions periodically. Maintaining code stability during a release freeze can impose stricter standards on incoming PRs, thereby lowering acceptance rates [2]. Understanding how these temporal phases intersect with PR decisions can shed light on "when" contributors are most likely to have their submissions accepted.

D. *Significance for Community and Industry*

High acceptance rates for quality contributions strengthen the project's ecosystem by retaining enthusiastic contributors and ensuring rapid integration of innovations [17] [1]. For businesses reliant on open-source software, efficient and predictable PR acceptance processes can accelerate product releases and reduce costs [26], [29]. Conversely, elevated rejection rates may create friction, discourage volunteer developers, and delay critical features or bug fixes [2].

III. RELATED WORK

Analyzing aspects of GitHub pull requests is a topic that has been explored for many different purposes in previously published papers. Yu et al. in [26] and Denae Ford [8] explored how factors such as the size of a PR and the number of comments in the PR can affect the time it takes for a PR to be reviewed. References [27] and [29] analyze how reviewers can be automatically recommended to specific PRs based on their contents. There are also others who seek to predict the success of PRs, such as [5], [17], and [13], which investigate the predictive power of a combination of quantitative and qualitative characteristics on whether or not a PR will be merged. In addition, [11] [18] studied how personality traits and emotional factors in PR comments could affect the acceptance rate of PRs. Some studies like [12] have tested applying the results of other papers to create a predictor that can automatically reject PRs to reduce the workload of reviewers.

Unlike [26], [29], and [12] this study will not focus on the review process of a PR. Yu et al. in [26] focus on the time it takes for a PR to be reviewed. However, while this can be affected by the characteristics of a PR, it is also affected by factors independent of the PR such as the number of PRs that need to be reviewed or reviewer availability. This is why our study focuses on the acceptance rate of PRs, since whether a PR is accepted may provide better insight into what a contributor can do to provide meaningful contributions to a project regardless of factors which are out of their control. While [29] and [12] attempt to automate PR rejection and reviewer assignment in order to improve reviewer efficiency from the point of view of project maintainers. In one of the earliest studies on PR discussions, Tsay et al. [24] analyzed extended discussions from a sample of GitHub PRs and conducted interviews with developers. The study found that the role of social interaction, stakeholder influence, and community dynamics has a huge influence in determining PR outcomes.

Relating to this, Khatoonabadi et al. [16] explored PR abandonment by analyzing 265,325 PRs across 10 large OSS projects, identifying 4,450 contributor-abandoned PRs. They found that abandonment was often due to novice contributors facing barriers, long review delays, or unconstructive feedback from maintainers. While Khatoonabadi et al. [16] shed light on why contributors abandon PRs, Shamsolhodaei et al. [21] looked deeper into why PRs are actively rejected by maintainers. They conducted a large-scale study involving 52,829 PRs from 3,931 projects and their analysis identified twelve distinct categories of rejection reasons, emphasizing that rejections are not homogenous and are influenced by a mix of technical and social factors. Our work draws from these studies by treating contributor experience and review timeliness as core variables influencing PR acceptance. We hope our results will provide useful insight to potential contributors, just like these studies, as to what practices make for successful PRs, thereby improving the overall quality of submitted PRs.

Specifically, code quality is a well-documented barrier to PR acceptance. Azeem et al. [3] studied 21,000 PRs from 25 popular Java projects and found that 44% of rejected PRs and 37% of accepted PRs contained code smells, with god classes and long methods being the most common. Smelly PRs were larger, had longer review times, and received more comments, suggesting that perceived technical debt strongly impacts reviewer perception and PR acceptance. Zou et al. [30] also conducted a study on code style inconsistency in 50,092 closed PRs from 117 GitHub projects and found that it negatively affects PR acceptance decision.

Another study was made where Ye [25] proposed a learning-to-rank model to identify suitable reviewers for PRs, based on 14 features measuring the relationship between reviewers and submitters. Features like prior collaboration history and file path similarity were identified as crucial in PR acceptance.

Focusing on the role of integrators, Gousios et al. [10] conducted a mixed-methods study, combining a survey of 749 integrators with project-level data and they found that integrators are heavily influenced by concerns about project quality and contributor reliability, and they struggle with prioritizing which PRs to merge. Reviewer recommendation is another domain closely tied to PR acceptance, as selecting an appropriate reviewer can expedite the review process and increase the likelihood of constructive feedback. Yu et al. [28] advanced reviewer recommendation research by proposing a hybrid approach using machine learning and social comment networks wherein they found out the importance of incorporating social dynamics and historical collaboration patterns in reviewer assignments, which indirectly influences PR acceptance through faster and more relevant reviews.

Similar to these studies and other related studies [5], [17], [13], [11], and [18], this paper analyzes the effect that specific characteristics have on the success rate of PRs. However, unlike those studies, this study focuses solely on quantitative characteristics. This approach allows us to avoid the time-intensive process of subjectively analyzing PRs, which increases the number of projects we can include in our data. These factors also have the benefit of being easier to see when looking at any given PR, and will not require subjective analysis. This means any discoveries will be more easily applied when manually investigating PRs.

Our approach intends to illuminate any correlation between easily visible quantitative factors of a PR and its acceptance rate across a large variety of open-source projects. Although a study from a socio-emotional angle made by Rishi [20] wherein their findings suggest that emotional tone—whether positive, neutral, or negative—can influence the reviewer's reaction and decision-making process is helpful, we hope that any correlation can provide insight into what will make a contributor's changes more likely to be accepted by reviewers regardless of project, without the need for any qualitative analysis, and thus help inform best practices when contributing to open source projects.

These studies demonstrate that PR acceptance is shaped by a confluence of factors including technical complexity, code quality, team dynamics, contributor experience, emotional tone, reviewer assignment, and social context. But unlike most of these studies, our paper focuses on what contributors can do to improve the acceptance rate of their PRs, rather than on optimizing the efficiency of the review process. We do this by analyzing the effect that quantitative characteristics have on their acceptance rate and avoiding subjective characteristics such as emotional tone. Our aim is to offer practical, data-driven guidance that contributors can apply across diverse projects that can help them increase the quality of PRs before submission for a higher acceptance rate.

IV. Methodology

This section outlines the study's approach, research inquiries, data collection, data cleaning, and analysis procedures.

A. *Goal*

The objective of this study is structured using the Goal-Question-Metric (GQM) framework [4], which provides a systematic approach for defining and interpreting research goals. The components of the GQM model for this study are defined as follows:

Purpose: To evaluate the factors that influence pull request acceptance

Issue: Understanding how contributor experience and PR attributes affect acceptance outcomes

Object: Pull requests within open-source software projects

Viewpoint: From the perspective of both contributors and project maintainers

B. *Research Questions*

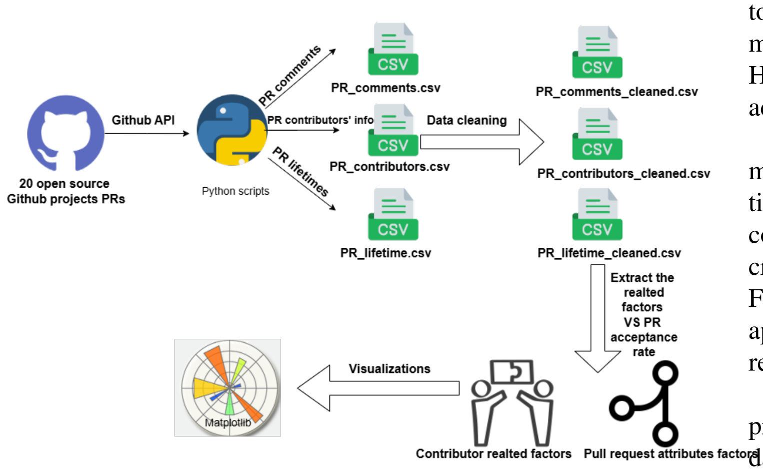
Based on the stated goal, we derive the following Research Questions (RQs):

RQ1: How do contributor-related factors, such as experience and prior contributions, impact the acceptance rates of pull requests in open-source projects?
This research question investigates how a contributor's background—such as the number of previously merged PRs, GitHub reputation, and activity level—influences the likelihood of their PRs being accepted. Contributors with a proven history of engagement and successful merges are more likely to be trusted by maintainers, resulting in higher acceptance rates. Furthermore, active participation in discussions, issue reporting, and consistent collaboration can positively affect the review process and increase the chance of acceptance.

RQ2: What is the influence of pull request attributes, such as code changes and review comments, on their likelihood of acceptance?
This question explores the characteristics of the pull requests themselves, such as the size of code changes, documentation quality, test coverage, and the volume and nature of review interactions. PRs that are smaller, well-documented, and accompanied by adequate tests are often easier to review and integrate, making them more likely to be accepted. Additionally, frequent, timely, and constructive review comments contribute to collaborative improvement of the PR, further enhancing the likelihood of acceptance.

C. *Methodology Overview*

Our approach combines quantitative metrics (e.g., lines added, review comments, timing) and qualitative factors (e.g., type of feedback, major release periods) to investigate PR acceptance outcomes, and use python libraries to visualize our results



D. *Data Construction and Collection*

(a) Repository Selection Criteria To construct a dataset representative of diverse open-source projects, we selected 20 GitHub repositories to strike a balance between dataset richness and analytical feasibility. This number was chosen to provide a sufficiently large and heterogeneous sample that captures a wide range of contributor behaviors, project sizes, and review practices, while keeping the scope manageable within the time and resource constraints of the study. Previous research on pull request evaluation has also relied on a comparable or smaller number of repositories to derive generalizable insights [5], [9]. Repositories were chosen across various domains based on the following criteria: **(1) Popularity** — measured by the number of stars and forks, ensuring a broad and active user base; **(2) Activity Level** — projects exhibiting a consistent flow of pull requests and commits; and **(3) Contributor Diversity** — repositories with 300 to 800 contributors, enabling analysis across different levels of experience and engagement. To promote domain diversity, each team member selected and collected data from five repositories using the GitHub REST API. Data was extracted in both CSV and JSON formats, with CSV files—following data cleaning and standardization—used for the final analysis to maintain consistency and ensure compatibility across all preprocessing scripts.

(b) Data Categories and Attributess The dataset is divided into three major components: Contributor Data: Information about developers, including experience level, past PRs, repository affiliations, and GitHub reputation metrics (followers, stars received), Pull Request Data: Attributes such as PR size (lines changed, files modified), review interactions (comments, approvals), and PR lifecycle events (created, merged, closed timestamps), Issue/Comment Data: Discussions surrounding PRs, including sentiment of review comments, volume of interactions, and dispute resolution trends [23].

E. *Data Cleaning and Preprocessing*

(a) Standardization and Formatting: To ensure uniform datasets, we performed the following preprocessing steps: Column Name Standardization: All field names were converted to lowercase and formatted using underscores, Duplicate Removal: Eliminated redundant PR entries to prevent bias, Date Handling: Converted timestamps to a consistent format for accurate temporal analysis.

(b) Noise Reduction and Filtering Bot Detection: Removed automated bot activity to focus on human contributions, Handling Missing Values: Critical fields (e.g., PR ID, contributor usernames) were dropped if missing, while non-critical fields (e.g., review comments) were filled with defaults, Feature Engineering: Created derived attributes such as PR approval rate (total merged PRs/total submitted PRs) and review engagement scores (comment density per PR).

(c) Merging Cleaned Datasets: Following individual preprocessing for each repository, we consolidated the cleaned data into three comprehensive datasets—one for pull request–related information, one for contributor-related data, and one for comment-related data. These datasets were constructed by aggregating and merging the corresponding cleaned files from all 20 selected repositories. This structured separation allows for modular analysis while maintaining a holistic view of the data across diverse open-source projects.

F. Statistical Methods and Analysis

To help analyze the relationship between pull request (PR) acceptance rates and different factors, Python based data analysis techniques were used. The analysis was done using pandas for data manipulation and matplotlib for data visualization and graphical representation. **(a)** The dataset for analyzing the lines added, lines deleted, comments, review comments, average review comments vs. acceptance rate contains 58,317 pull requests (PRs). The data was collected from 20 different repositories and combined into a single CSV file for analysis. The analysis involved categorizing numerical values into meaningful bins, calculating acceptance rates for each category, and visualizing the results using bar charts in Matplotlib. **(b)** for analyzing the acceptance rate vs PRs lifetime, we derived key metrics such as PR acceptance rate (defined as the ratio of merged PRs to total PRs) and PR lifetime (computed as the difference between closed_at and created_at timestamps). The cleaned data was grouped into defined lifetime intervals, and acceptance rates were computed per group. The statistical analysis involved grouping PRs by these lifetime bins and calculating aggregated acceptance rates to identify trends and correlations. Visualizations, such as bar charts and line plots, were used to represent these trends clearly. These plots provided insights into how PR lifetime influences the likelihood of acceptance, helping to validate hypotheses about review responsiveness and contributor engagement. The methodology enabled an interpretable, reproducible, and scalable analysis workflow applicable to diverse open-source repositories.

(c) For analyzing Acceptance rate vs Repository Size and PR Labels, we used over 25,000 PRs from multiple repositories, capturing essential attributes such as PR metadata, repository size, labels assigned, timestamps of PR events, and acceptance status. The analysis was structured into three key phases: data preprocessing and cleaning, exploratory data analysis (EDA) and correlation analysis, and statistical modeling (logistic regression [6]).

Repository size metrics (stars, forks, contributors, commits) were extracted from a combined metadata column and converted into numerical values. PR event timestamps (created, merged, closed) were converted to datetime format for time-based analysis. PR acceptance was defined as a binary variable (1 if merged, 0 otherwise) to facilitate classification modeling. PR labels were categorized into four key groups: Bug Fix, Feature, Documentation, and Dependencies. This was done by analyzing label keywords within PR metadata.

Stars ranged from a few thousand to over 97,000. Forks ranged from 6,000 to 26,000.The dataset was highly skewed, with a large portion of PRs originating from high-profile repositories like Angular (96k stars, 25.8k forks) and PyTorch, while smaller repositories had considerably fewer contributions.

Maintainers used labels for both technical classification (e.g., area: docs, module: inductor, feature enhancement) and process tracking (e.g., action: merge, target: patch, dependencies). The most frequently used labels included: "Action: Merge" – Assigned to 3,360 PRs, "Target: Patch" – Assigned to 1,991 PRs, Area-based labels (e.g., area: core, area: docs). Many PRs had multiple labels, often combining an area tag, a target tag, and a status tag.

To quantify the relationships between repository size, PR labels, and PR acceptance, we computed Pearson correlation coefficients. The Pearson formula used was:

$$r = \frac{\sum(X_i - \bar{X})(Y_i - \bar{Y})}{\sqrt{\sum(X_i - \bar{X})^2}\sqrt{\sum(Y_i - \bar{Y})^2}}$$

Fig. 1. PR Acceptance Rates Across Repositories

where X and Y represent variables such as forks and PR acceptance, and $\overline{X}$ and $\overline{Y}$ are their respective means. Spearman correlation was also used to detect non-linear relationships where appropriate.

For **Statistical Modeling (Logistic Regression)** to predict PR acceptance probability, we used a logistic regression model, incorporating Repository size (stars, forks), PR Labels (Bug Fix, Feature, Documentation, Dependencies). The logit function used was:

$$\text{logit}(P) = \beta_0 + \beta_1 X_1 + \beta_2 X_2 + \cdots + \beta_n X_n$$

Fig. 2. PR Acceptance Rates Across Repositories

where P represents the probability of PR acceptance, and $X_1, X_2, ..., X_n$ are predictor variables, including repository size (forks, stars) and PR labels. Fork count was chosen as the primary size indicator due to its strong correlation with project popularity. To interpret the magnitude of effects, odds ratios were calculated, and statistical significance (p-values) was used to determine which variables significantly influenced PR acceptance. The model was evaluated for goodness-of-fit using pseudo R^2 values and chi-square tests, ensuring that the predictors contributed meaningfully.

This methodology was chosen to provide a rigorous yet interpretable analysis of PR acceptance trends. Prior research has established that repository popularity influences PR review processes, but few studies have quantified the impact of PR labels in combination with project size. Studies such as [2] and [11] have examined reviewer influence and contributor experience, but our approach extends this by incorporating structural repository attributes (stars, forks) and PR metadata to provide a more holistic view.

V. RESULTS

This section analyzed 6 key factors that affect the acceptance rate of the pull request, detailing the infulence by visualizations

A. *Acceptance Rate vs. Lines Added/Lines Deleted*

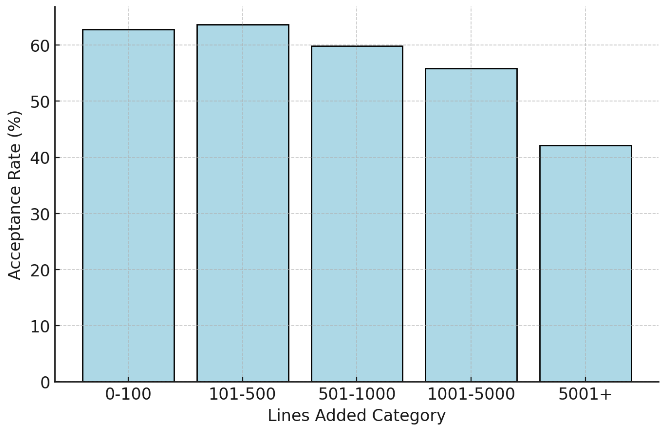


Fig. 3. Acceptance Rate vs Lines Added

Figure 3 shows that PRs that were not accepted had a significantly higher average number of lines added compared to accepted PRs. Large PRs may introduce complexity, making them harder to review and more prone to rejection, whereas smaller, more focused PRs are generally preferred as they are easier to review and integrate.

Similarly to the lines added, Figure 4 shows that the PRs with a higher number of lines deleted had a lower acceptance rate. Large deletions might indicate a major refactor, which could be harder to validate and more likely to conflict with other changes. Reviewers may be hesitant to approve such sweeping changes without extensive testing. PRs with more than 1,000 deleted lines were significantly less likely to merge, possibly due to the inherent risk of breaking dependencies or removing the necessary functionality.

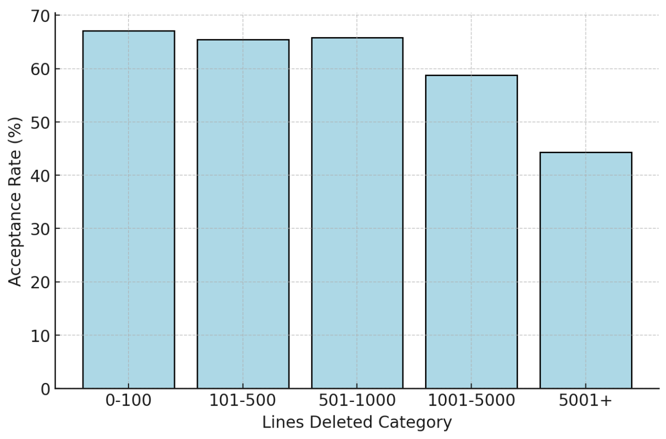


Fig. 4. Acceptance Rate vs Lines Deleted

B. *Acceptance Rate vs. Comments/Review Comments/Average Review Comments*

Figure 5 shows that PRs that were not accepted had slightly more comments than accepted PRs. Increased discussion in comments can indicate unresolved issues, disagreements, or lack of clarity, all of which could contribute to PR rejection. Accepted PRs might be better structured, requiring fewer clarifications. Although collaboration is important, excessive back-and-forth discussions might signal underlying issues in the code, contributing to a lower acceptance rate.

Figure 6 shows that accepted PRs had more review comments on average compared to not accepted PRs. A higher number of review comments in accepted PRs suggests that they undergo a thorough review process, and necessary improvements are made before merging. PRs that receive minimal feedback might indicate a lack of interest or poor quality, leading to rejection. The PRs with the highest rating had a higher acceptance rate, indicating that thorough review engagement leads to more successful merges.

Figure 7 shows that PRs that were accepted had a higher average number of review comments compared to not accepted PRs. Thorough code reviews improve PR quality; therefore, PRs with higher review engagement likely receive the necessary changes and refinements, increasing their chances of acceptance.

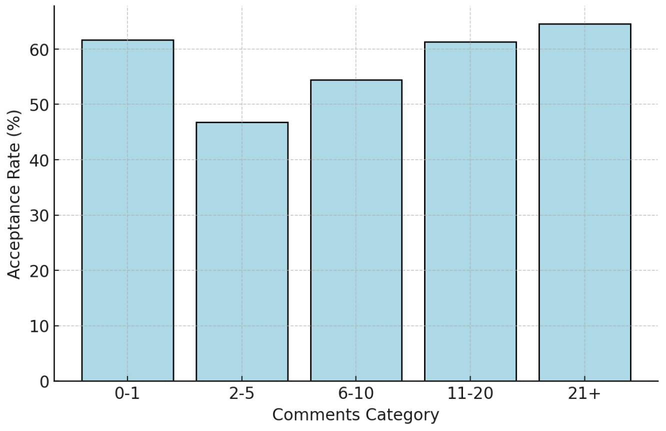


Fig. 5. Acceptance Rate vs Comments

C. *Acceptance Rate vs. Pull request lifetime group*

Figure 8 shows the relationship between PR (Pull Request) acceptance rate and PR lifetime grouped by years. The results demonstrate a clear inverse correlation between PR lifetime and acceptance probability. PRs that are closed within 0–1 year exhibit the highest acceptance rate, exceeding 50 percent, suggesting that shorter-lived PRs are more likely to be merged successfully. In contrast, PRs that remain open for longer durations show progressively lower acceptance rates. PRs in the 1–2 year and 2–3 year bins drop to around 25 percent, while PRs extending beyond 3 years show acceptance rates below 20 percent, with the 4+ year group having the lowest acceptance rate, near 10 percent.

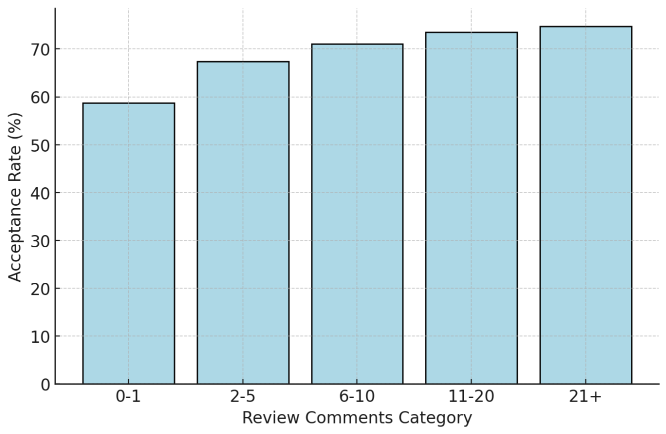


Fig. 6. Acceptance Rate vs Review Comments

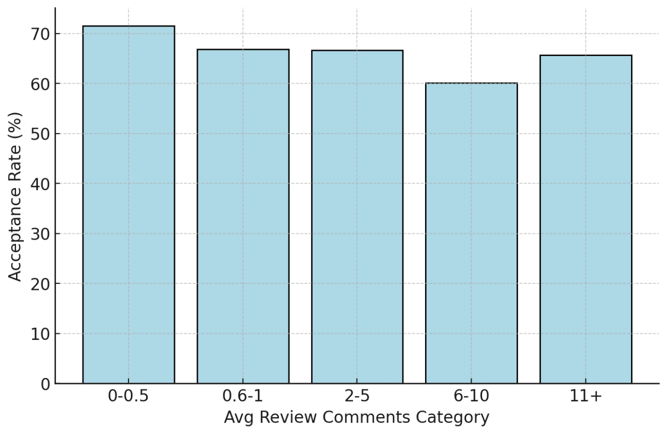


Fig. 7. Acceptance Rate vs Average Review Comments

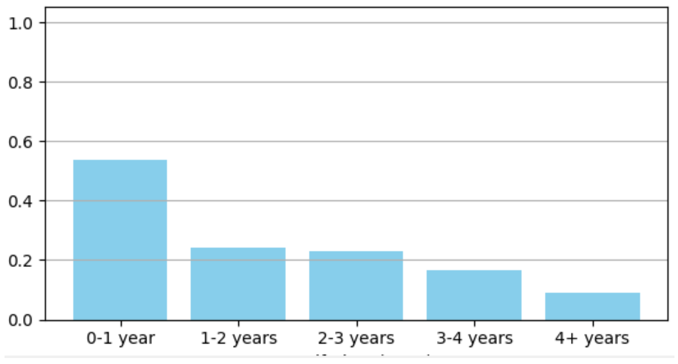


Fig. 8. Acceptance Rate vs PRs lifetime grouped by years

D. Acceptance Rate vs. Reviewer Count

Figure 9 demonstrates the correlation between the number of users who reviewed a PR and its average acceptance rate. PRs reviewed by 2 or 3 users have the highest acceptance rate, however, 0 or 1 is not much lower. The most significant difference arises when 4 or more users review a PR, with 0% acceptance for PRs with 7 or more reviewers. It is important to note that approximately 90% of PRs were reviewed by 0 to 3 reviewers. So, while it appears that a larger number of reviewers are less likely to accept a PR (perhaps because it is more likely that one of the reviewers finds an issue), it could be the case that unique PRs that require more reviewers are naturally less likely to be accepted due to their complexity or other reasons unrelated to the number of reviewers.

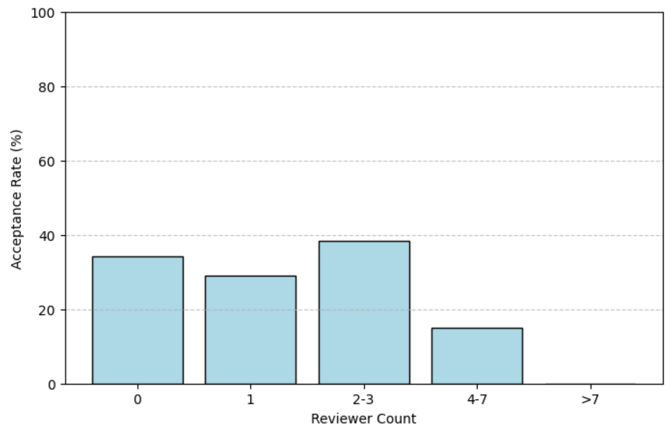


Fig. 9. Acceptance Rate vs. Reviewer Count

E. Acceptance Rate vs. Pull request Repository Size(Stars, Forks, Contributors, Commits)

Exploratory Data Analysis: One of the first insights we noticed in the data was PR acceptance rates across different repositories. Some projects were significantly more open or welcoming to contributions than others. For example:
Netdata accepted 92.3% of PRs, while Puppeteer accepted 83.3%.
Scrapy and EthereumData had acceptance rates of around 63%.
Angular accepted only 6% of PRs, while PyTorch had an even lower rate of 1%.

This indicates that larger, high-profile repositories tend to be much more selective or restrictive in merging PRs, whereas smaller, community-driven projects tend to be less selective and accept more contributions. Certain projects had almost no PR acceptance, probably indicating highly competitive or internal development-focused repositories.

Stars Distribution: Most repositories have a relatively high number of stars, but only a few repositories dominate with significantly larger counts. Forks Distribution: Similar to stars, forks also show a skewed distribution, where a few repositories

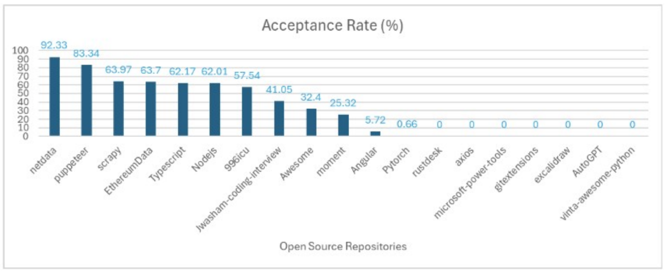


Fig. 10. PR Acceptance Rates Across Repositories

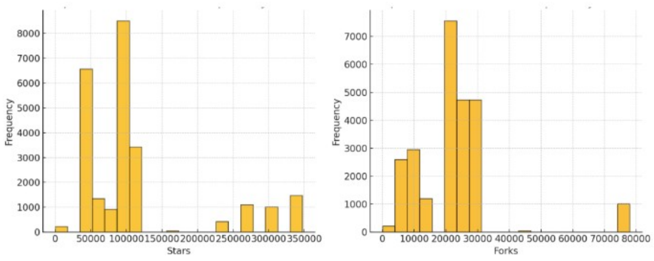


Fig. 11. Repository Popularity Distribution

have an extremely high number of forks. Repositories with higher stars and forks had lower PR acceptance rates. Smaller repositories (fewer stars/forks) were more lenient with PR contributions.

After gaining an initial understanding of PR acceptance trends and repository size distributions, the next step was to quantify these relationships. Specifically, we wanted to answer:

How does repository size (stars, forks) influence PR acceptance rates?

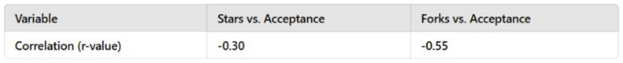
Do PRs with certain labels (e.g., bug fixes, feature enhancements, documentation updates) have higher or lower acceptance rates?

Is there a pattern in how labels are applied across different repositories?

To measure these associations, we computed Pearson correlation coefficients.

Correlation Analysis Our correlation analysis denoted a significant inverse relationship between repository size and pull request (PR) acceptance rates. Specifically, the fork count showed a Pearson correlation coefficient (r) of approximately -0.59, while the star count exhibited a correlation of around -0.30. This data confirms that larger, more popular projects, like Angular and PyTorch, tend to be more selective in merging PRs. This aligns with prior research that identifies project popularity as a strong negative predictor for pr acceptance likelihood. This could be due to maintainers feeling ”audience pressure” to be more cautious with changes in widely-used projects. Furthermore, stars and forks are highly correlated $r \approx 0.80$, indicating they serve as overlapping measures of project popularity[7].

To quantify the relationships between repository size (stars, forks) and PR acceptance, we computed Pearson's correlation coefficients.



Variable	Stars vs. Acceptance	Forks vs. Acceptance
Correlation (r-value)	-0.30	-0.55

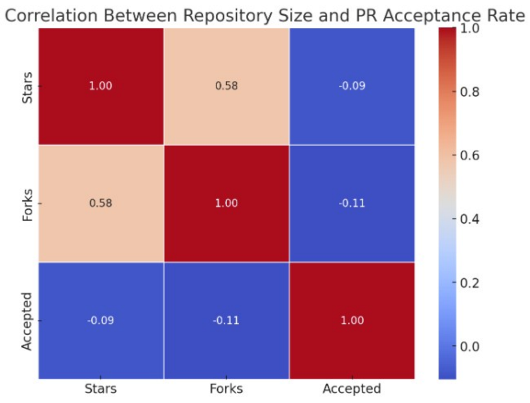


Fig. 12. Correlation table and Heatmap

Stars and PR Acceptance (-0.30 correlation): More popular repositories (higher stars) are less likely to accept PRs.

Forks and PR Acceptance (-0.55 correlation): The number of forks shows an even stronger negative correlation with PR acceptance, meaning that highly forked repositories are more restrictive in merging PRs. Forks often indicate high interest in a project but can also mean many competing contributions, making PR approval more difficult.

F. *Acceptance Rate vs. Pull Request Labeling*

The number of labels applied to a pull request showed a modest negative correlation with acceptance $r \approx -0.32$. This suggests that PRs requiring more labels, potentially indicating complexity or contention, are less likely to be merged. However, this correlation is confounded by project size, as larger projects tend to apply more labels and have lower acceptance rates. Therefore, the rejection is not solely due to the number of labels. Furthermore, we observed that certain label categories significantly influence acceptance rates.

Bug fixes comparatively have a higher acceptance rate (56%) than general PRs, indicating that maintainers prioritize fixing issues over adding new features.

Feature additions (25%) are less likely to be accepted, possibly because they require extensive review or align poorly with project roadmaps.

Documentation PRs (18%) are the least accepted, suggesting that maintainers prefer handling documentation internally or require strict quality standards.

Dependency updates (68%) have the highest acceptance rate, likely due to automated PRs from tools like Dependabot, which are easy to verify and merge.

Although many labels serve as process indicators, such as ”action: merge,” these do not guarantee acceptance. We

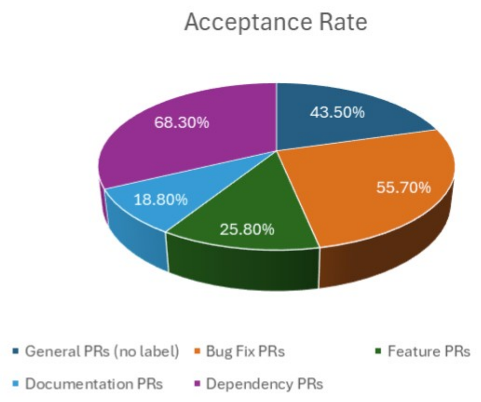
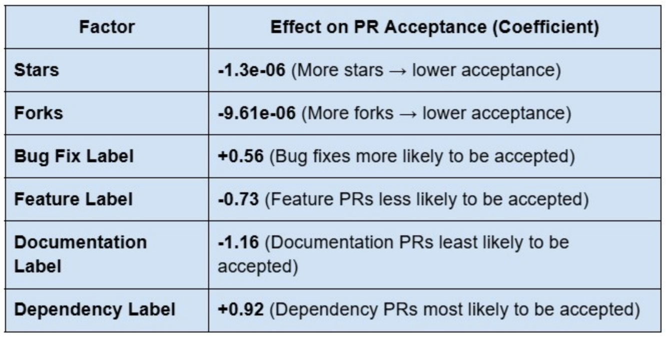


Fig. 13. PR acceptance rates for PR labels

found that a significant number of PRs labeled "action: merge" were still closed without merging, highlighting that labels are informative but not definitive.

Logistic Regression: Predicting PR Acceptance

To further quantify these relationships, we built a logistic regression model predicting PR acceptance probability based on repository size and PR labels. The model estimated the likelihood of a PR being accepted given different conditions.



Factor	Effect on PR Acceptance (Coefficient)
Stars	**-1.3e-06** (More stars → lower acceptance)
Forks	**-9.61e-06** (More forks → lower acceptance)
Bug Fix Label	**+0.56** (Bug fixes more likely to be accepted)
Feature Label	**-0.73** (Feature PRs less likely to be accepted)
Documentation Label	**-1.16** (Documentation PRs least likely to be accepted)
Dependency Label	**+0.92** (Dependency PRs most likely to be accepted)

Fig. 14. PR acceptance based on logistic regression coefficients

Every increase in repository stars and forks decreases PR acceptance probability, reinforcing that larger projects are more selective. Bug fixes and dependency updates significantly increase PR acceptance odds, confirming that maintainers prioritize stability over new features. Documentation PRs are the least likely to be accepted, indicating stricter editorial control. The logistic regression model had a pseudo R^2 value of 0.029, meaning it explains some variance in PR acceptance but suggests additional factors might also play a role.

VI. Discussions

This section discusses the research questions and elaborates on the findings from the analysis, with detailed interpretations and insights into how contributor-related factors and specific pull request attributes impact acceptance rates in open-source software projects.

A. *RQ1: Contributor Impact*

Our analysis reveals that contributor-related factors strongly indicate their impact on open-source project PR acceptance rates. Experienced contributors with a proven record of previously merged pull requests demonstrate substantially higher acceptance rates(see Figure 10). This observation corresponds with the established idea in the literature that maintainers trust contributors based on their prior positive interactions and demonstrated capability [2], [9]. In our analysis, projects consistently showed higher merge rates for submissions made by frequent contributors which shows the importance of building a solid reputation within an open-source community. Furthermore, contributors who actively participate in discussions and issue resolution significantly increased their acceptance likelihood which supports the correlation between contributor engagement and successful PR merges as found in previous research [23].

In practical terms, this indicates that newcomers may face initial barriers to acceptance, necessitating proactive community engagement, smaller and clearer initial submissions, and consistent contribution to build trust. This finding is crucial as it encourages open-source communities to explicitly document contribution guidelines and review practices which aids in a smoother onboarding of new contributors.

B. *RQ2: Attributes Impact*

Our findings clearly show that PR-specific attributes considerably influence acceptance likelihood. Smaller and well-documented PRs, which involve fewer lines of code added or deleted, are significantly more likely to be accepted (see Figures 3 and 4). This confirms prior studies, such as by Yu et al. [26] that shows that maintainers prefer concise changes as these require less review effort and pose fewer integration risks.

Review comments also played a critical role in acceptance rates. PRs that received more constructive and detailed feedback from reviewers had higher acceptance chances, as demonstrated in Figures 6 and 7. This illustrates the value of thorough code reviews and clear communication in enhancing PR quality, corroborating previous research findings [23] [29]. However, excessive comments, often indicative of unresolved issues or controversies, correlated negatively with PR acceptance, suggesting that balanced and targeted feedback is essential.

The analysis of PR lifetime further shows a clear inverse relationship between the duration a PR remains open and its acceptance rate (see Figure 8). Shorter-lived PRs (less than one year) had notably higher acceptance rates, whereas PRs open longer than three years showed acceptance rates below 20 percent. This corresponds with findings from previous literature [2] [9], which suggests that timely reviews and quicker resolutions of PRs significantly improve the chances of successful merges.

Lastly, repository-specific attributes, such as project size indicated by star and fork counts, were found to inversely correlate with PR acceptance rates (Figures 10 and 12).

Larger and more prominent projects, such as Angular and PyTorch, exhibited notably lower acceptance rates compared to smaller, community-focused projects. Additionally, PR labels influenced acceptance, with labels indicating ”bug fixes” and ”dependency updates” showing significantly higher acceptance probabilities compared to documentation or feature-related PRs (see Figure 13). These results provide actionable insights which suggests that maintainers prioritize stability and clear project alignment in reviewing submissions.

VII. Implication

This section summarizes the practical implications of our findings for key stakeholders—researchers, contributors, and maintainers in open-source software development.

A. For Researchers

This study provides insights into the critical factors affecting pull request (PR) acceptance rates in open-source software projects. Our findings indicate that experienced contributors, concise PRs, and timely constructive feedback significantly increase acceptance likelihood. Researchers interested in software engineering, collaboration dynamics, and open-source community behaviors can utilize these insights to further explore how contributor experience and specific PR attributes impact project outcomes. Future research could build on these findings by integrating qualitative data, such as maintainer feedback patterns or contributor motivations, to deepen understanding of open-source collaboration.

B. For Contributors

The results highlight clear strategies that contributors can adopt to increase their PR acceptance chances. Contributors should prioritize submitting smaller, well-documented, and thoroughly tested pull requests. Engaging actively in community discussions and maintaining consistent, high-quality contributions also significantly enhance contributor reputation and trust, leading to higher acceptance rates. New contributors particularly benefit from these insights by understanding the importance of gradually building their presence within open-source projects.

C. For Maintainers

Maintainers can leverage our findings to refine their review processes and guidelines, aiming to foster effective collaboration and maintain high project standards. By emphasizing prompt and constructive feedback, maintainers can facilitate quicker PR resolutions and enhance community engagement. Additionally, clearly labeling PRs according to their nature (e.g., bug fixes, features, documentation updates) and setting explicit guidelines around PR size and scope can help streamline project management and improve overall efficiency.

VIII. Threats to Validity

Although we carefully designed this study, several validity threats still exist:

A. Construct Validity

Construct validity refers to the extent to which the metrics and operational definitions accurately capture the concepts they intend to measure. In our study, we relied primarily on quantifiable attributes such as lines changed, PR lifetime, and the number of review comments to assess pull request acceptance. However, qualitative dimensions such as the semantic quality of code changes, the tone of reviewer feedback, or contributor motivation were not included. Prior research suggests that such non-technical factors—like trust, communication clarity, and alignment with community values—can strongly influence PR evaluation outcomes [23]. Future studies may benefit from integrating natural language processing or sentiment analysis techniques to extract richer, context-aware insights from PR discussions and comments.

B. Internal Validity

Internal validity concerns the ability to draw causal conclusions from observed relationships. Although we observed correlations between certain contributor behaviors and PR acceptance, we cannot conclusively claim causation due to potential confounding variables such as project-specific norms, hidden maintainer preferences, or undocumented review policies. We mitigated this issue by selecting a diverse set of 20 repositories with varying contributor sizes, popularity levels, and development maturity. Nonetheless, selection bias may still persist, particularly if maintainers applied undocumented filters or practices that influenced PR outcomes in ways our model could not detect.

C. External Validity

External validity relates to the generalizability of the results beyond the studied context. As our dataset primarily includes well-known and highly active GitHub repositories, the findings may not extrapolate to smaller, less maintained, or more niche projects. Smaller communities may have different collaboration patterns, more lenient review processes, or fewer contributors involved in evaluation. To improve external validity, future research could incorporate repositories from different domains, hosting platforms (e.g., GitLab, Bitbucket), or organizational types (e.g., academic, corporate, hobbyist) to understand the broader landscape of pull request management.

D. Conclusion Validity

Conclusion validity addresses the extent to which the analysis and interpretations are statistically sound. While we used well-established quantitative metrics and visualization tools, alternative measures such as reviewer response time, code complexity, or review depth could offer complementary insights. Moreover, some metrics like PR lifetime might be skewed due to outliers (e.g., PRs left open for years without response). Statistical robustness could be further enhanced by applying multiple machine learning models or using statistical significance tests to validate observed trends. Cross-validation with larger or longitudinal datasets would also strengthen the reliability of future conclusions [5].

By acknowledging these threats and proposing mitigation strategies, we aim to ensure transparency and promote ongoing improvement in the study of pull request acceptance dynamics.

IX. Conclusion and Future Work

In conclusion, this study underscores the multifaceted nature of pull request acceptance in open-source communities. By focusing on quantitative factors such as contributor experience, lines added or deleted, review discussions, and repository size, we illustrate that both social and technical elements converge to shape PR outcomes. Smaller, well-documented submissions that garner timely and constructive feedback are particularly prone to merging, while large or long-running pull requests in high-profile repositories face stricter scrutiny.

From a contributor perspective, our findings emphasize the importance of building trust through consistent engagement and concise, high-quality pull requests. For maintainers, the results highlight the value of clear review guidelines and proactive labeling to streamline decisions. By identifying potential areas for automated tooling—such as prioritizing bug fixes or dependency updates—we envision faster and more collaborative contribution cycles across diverse open-source projects.

For future exploration, although the present analysis covers key dimensions (e.g., contributor experience, review intensity, project popularity), the following directions may further enrich our understanding:

- *Qualitative Analysis:* Incorporating sentiment or topic modeling of review comments could shed light on how communication tone affects acceptance rates, complementing the quantitative features used here.
- *Extended Repository Selection:* Examining smaller or less active repositories might reveal community norms distinct from the high-traffic projects studied, enhancing the generalizability of findings.
- *Temporal Studies:* Investigating how acceptance trends evolve throughout project lifecycles (e.g., release freezes, major refactors) may inform adaptive review processes.
- *Refined Predictive Models:* Combining logistic regression with newer machine learning approaches (e.g., gradient boosting, neural networks) could capture complex interactions among PR attributes and better predict acceptance outcomes.

By demonstrating how contributor engagement, project structure, and repository dynamics interplay in pull request evaluation, we hope this work inspires both developers and maintainers to adopt evidence-based best practices. Through continued research and data-driven tooling, open-source ecosystems can further optimize collaborative development, welcoming diverse contributions that fuel project sustainability and growth.

References

[1] Adam Alami, Marisa Leavitt Cohn, and Andrzej Waisowski. How do foss communities decide to accept pull requests? In *Proceedings of the 24th International Conference on Evaluation and Assessment in Software Engineering*, pages 220–229, 2020.

[2] Muhammad Ilyas Azeem, Qiang Peng, and Qing Wang. Pull request prioritization algorithm based on acceptance and response probability. In *2020 IEEE 20th International Conference on Software Quality, Reliability and Security (QRS)*, pages 231–242. IEEE, 2020.

[3] Muhammad Ilyas Azeem, Saad Shafiq, Atif Mashkoor, and Alexander Egyed. Code smells in pull requests: An exploratory study. *Software: Practice and Experience*, 54(3):419–436, 2024.

[4] Victor R Basili. *Software modeling and measurement: the Goal/Question/Metric paradigm*. University of Maryland at College Park, 1992.

[5] Di Chen, Kathyrn T Stolee, and Tim Menzies. Replication can improve prior results: A github study of pull request acceptance. In *2019 IEEE/ACM 27th International Conference on Program Comprehension (ICPC)*, pages 179–190. IEEE, 2019.

[6] Tapajit Dey and Audris Mockus. Effect of technical and social factors on pull request quality for the npm ecosystem. In *Proceedings of the 14th ACM/IEEE International Symposium on Empirical Software Engineering and Measurement (ESEM)*, pages 1–11, 2020.

[7] Tapajit Dey and Audris Mockus. Which pull requests get accepted and why? a study of popular npm packages. *arXiv preprint arXiv:2003.01153*, 2020.

[8] Denae Ford, Mahnaz Behroozi, Alexander Serebrenik, and Chris Parnin. Beyond the code itself: How programmers really look at pull requests. In *2019 IEEE/ACM 41st International Conference on Software Engineering: Software Engineering in Society (ICSE-SEIS)*, pages 51–60. IEEE, 2019.

[9] Georgios Gousios, Martin Pinzger, and Arie van Deursen. An exploratory study of the pull-based software development model. In *Proceedings of the 36th international conference on software engineering*, pages 345–355, 2014.

[10] Georgios Gousios, Andy Zaidman, Margaret-Anne Storey, and Arie Van Deursen. Work practices and challenges in pull-based development: The integrator's perspective. In *2015 IEEE/ACM 37th IEEE International Conference on Software Engineering*, volume 1, pages 358–368. IEEE, 2015.

[11] Rahul Iyer. Effects of personality traits and emotional factors in pull request acceptance. Master's thesis, University of Waterloo, 2019.

[12] Jing Jiang, Jia-teng Zheng, Yun Yang, and Li Zhang. Ctcppre: A prediction method for accepted pull requests in github. *Journal of Central South University*, 27(2):449–468, 2020.

[13] Jing Jiang, Jiateng Zheng, Yun Yang, Li Zhang, and Jie Luo. Predicting accepted pull requests in github. *Sci. China Inf. Sci*, 64:179105, 2021.

[14] Eirini Kalliamvakou, Daniela Damian, Kelly Blincoe, Leif Singer, and Daniel M German. Open source-style collaborative development practices in commercial projects using github. In *2015 IEEE/ACM 37th IEEE international Conference on software engineering*, volume 1, pages 574–585. IEEE, 2015.

[15] SayedHassan Khatoonabadi. *Pull Request Abandonment in Open-Source Projects*. PhD thesis, Concordia University, 2023.

[16] SayedHassan Khatoonabadi, Diego Elias Costa, Rabe Abdalkareem, and Emad Shihab. On wasted contributions: Understanding the dynamics of contributor-abandoned pull requests–a mixed-methods study of 10 large open-source projects. *ACM Transactions on Software Engineering and Methodology*, 32(1):1–39, 2023.

[17] Valentina Lenarduzzi, Vili Nikkola, Nyyti Saarimäki, and Davide Taibi. Does code quality affect pull request acceptance? an empirical study. *Journal of Systems and Software*, 171:110806, 2021.

[18] Marco Ortu, Giuseppe Destefanis, Daniel Graziotin, Michele Marchesi, and Roberto Tonelli. How do you propose your code changes? empirical analysis of affect metrics of pull requests on github. *IEEE access*, 8:110897–110907, 2020.

[19] Hocine Rebatchi, Tégawendé F Bissyandé, and Naouel Moha. Dependabot and security pull requests: large empirical study. *Empirical Software Engineering*, 29(5):128, 2024.

[20] Deepak Rishi. Affective sentiment and emotional analysis of pull request comments on github. Master's thesis, University of Waterloo, 2017.

[21] Amirreza Shamsolhodaei, Rungroj Maipradit, and Meiyappan Nagappan. 12 reasons why: Not all pull request rejections are the same. *Available at SSRN 4946578*, 12.

[22] Daricélio Moreira Soares, Manoel L De Lima Junior, Leonardo Murta, and Alexandre Plastino. Rejection factors of pull requests filed by core team developers in software projects with high acceptance rates.

In *2015 IEEE 14th international conference on machine learning and applications (ICMLA)*, pages 960–965. IEEE, 2015.

[23] Jason Tsay, Laura Dabbish, and James Herbsleb. Influence of social and technical factors for evaluating contribution in github. In *Proceedings of the 36th international conference on Software engineering*, pages 356–366, 2014.

[24] Jason Tsay, Laura Dabbish, and James Herbsleb. Let's talk about it: evaluating contributions through discussion in github. In *Proceedings of the 22nd ACM SIGSOFT international symposium on foundations of software engineering*, pages 144–154, 2014.

[25] Xin Ye. Learning to rank reviewers for pull requests. *IEEE Access*, 7:85382–85391, 2019.

[26] Yue Yu, Huaimin Wang, Vladimir Filkov, Premkumar Devanbu, and Bogdan Vasilescu. Wait for it: Determinants of pull request evaluation latency on github. In *2015 IEEE/ACM 12th working conference on mining software repositories*, pages 367–371. IEEE, 2015.

[27] Yue Yu, Huaimin Wang, Gang Yin, and Charles X Ling. Reviewer recommender of pull-requests in github. In *2014 IEEE International Conference on Software Maintenance and Evolution*, pages 609–612. IEEE, 2014.

[28] Yue Yu, Huaimin Wang, Gang Yin, and Charles X Ling. Who should review this pull-request: Reviewer recommendation to expedite crowd collaboration. In *2014 21st Asia-Pacific Software Engineering Conference*, volume 1, pages 335–342. IEEE, 2014.

[29] Yue Yu, Huaimin Wang, Gang Yin, and Tao Wang. Reviewer recommendation for pull-requests in github: What can we learn from code review and bug assignment? *Information and software technology*, 74:204–218, 2016.

[30] Weiqin Zou, Jifeng Xuan, Xiaoyuan Xie, Zhenyu Chen, and Baowen Xu. How does code style inconsistency affect pull request integration? an exploratory study on 117 github projects. *Empirical Software Engineering*, 24:3871–3903, 2019.