Not All Pull Request Rejections Are The Same

by

Amirreza Shamsolhodaei

A thesis
presented to the University of Waterloo
in fulfillment of the
thesis requirement for the degree of
Master of Mathematics
in
Computer Science

Waterloo, Ontario, Canada, 2024

Author's Declaration

Abstract

In the Open Source Software (OSS) development landscape, evaluating pull requests extends beyond code quality assessment. Recent research has revealed the significant influence of social dynamics and perceptions on pull request evaluations, a notion our study seeks to expand upon. By examining the intricate reasons behind pull request rejections, we aim to move beyond the traditional view of rejections as a monolithic category.

Utilizing a dataset comprising of 52,829 pull requests across 3,931 projects, we conduct a large-scale comprehensive analysis identifying twelve distinct categories of rejection reasons. Our findings underscore that although social ties and technical abilities are factors that influence pull request decisions, they are not consistent across all rejection reasons. Notably, certain characteristics, such as extensive line changes and team size, exhibit varied impacts on different types of rejections, indicating the complex interplay between social and technical factors in pull request assessments.

This study provides a multifaceted understanding of the OSS contribution evaluation process, highlighting the complexity and diversity of rejection reasons. By describing the specific features that influence distinct types of rejections, we contribute to the development of more nuanced strategies for managing contributions. Our findings offer valuable insights for both contributors and project maintainers, emphasizing the need for a tailored approach to understanding and enhancing the pull request evaluation process in OSS projects.

Acknowledgements

This thesis would not have been possible without the help, support, and guidance of many people. First and foremost, I would like to express my utmost gratitude to my supervisor, Dr. Meiyappan Nagappan, for being the most supportive supervisor and for his guidance and help during my studies. He has always shown immense patience and his unwavering support not only covered my studies, but it spanned other aspects of my life since I have known him.

I would also like to thank Dr. Rungroj Maipradit for providing guidance and feedback throughout my study, especially these last 6 months.

I am also grateful to Dr. Shane McIntosh and Dr. Pengyu Nie for accepting to be my reviewers and providing me with constructive feedback.

I wanted to thank the brilliant students at the Software Analytics Group (SWAG) who made my daily life in the lab a magnificent experience; Including Shaquille, Gareema, Joy, Farshad, Mahtab, Vikram, Ayinde, Noble, Partha, Arian, Akin, Favour, Mahmoud, Anubhav, and Gaosen. I also want to show my gratitude to Reza, a dear friend and an alumni of SWAG lab, who helped me massively in the initial steps and throughout my studies.

I am proud to be a part of the University of Waterloo community, I will always keep the knowledge and experiences I have gained in one of the most prestigious Computer Science programs close to my heart.

I also want to thank my friends who showed amazing support and helped me through different obstacles that I have faced; Mehran, Mahshid, Ehsan, Mahsa, Iman, Moein, Arshia, Amir, Amin, Houmaan, Ensie, and Sara to name a few.

Finally and most importantly, none of this could have happened without my parents, my brothers, and my sisters-in-law, Especially Amirali and Pendar, whose presence has been a solid rock that I could always rely on.

Dedication

This thesis is dedicated to my grandparents, Talat, Iraj, Zahra, and Hossein; my parents, Maryam and Mehdi; my brothers, Amirhossein and Amirali; my sisters-in-law, Tina and Pendar; and my nephew, Hiva.

Table of Contents

List of Figures

List of Tables

Chapter 1

Introduction

Open Source Software (OSS) development is a global phenomenon that uses platforms like GitHub at the forefront, facilitating collaborative efforts from developers worldwide. The lifecycle of collaborative development in GitHub often begins when a developer, known as the contributor, proposes changes to a project through a pull request. Upon reviewing the characteristics of the changes, including their defect-proneness, an integrator is assigned to review the pull request. They conduct a critical assessment of the proposed changes, considering factors such as code quality and the broader context of the contribution, to determine whether it should be merged to the code base [1, 2, 3].

Integrators are perceived to evaluate pull requests based on code quality and related technical factors [4]. However, other factors also influence the pull request outcome. Such as social dynamics and perceptions [5], social elements [6], ethnicity [7], gender [8], and geographical location [9]. For instance, women often experience lower acceptance rates compared to their counterparts, and developers' national origins also impact pull request acceptance rates. Furthermore, correlations have been observed between the personalities of developers and the likelihood of their pull requests being accepted[2]. Nonetheless, there remains an underexplored aspect in the study of pull request rejections. Recent studies often generalize these rejections, considering them as a uniform category. This perspective fails to recognize that rejections come in different forms, each influenced by distinct factors.

Recent contributions by Nadri et al. [10] were the start to tackling this problem, highlighting the complexity and diversity of pull request rejection. Understanding these reasons is crucial for targeted and effective strategies for managing contributions in OSS projects. By providing specific feedback regarding diverse reasons for rejections, project maintainers and integrators can enhance collaboration. Moreover, a detailed understanding of the rejection reasons can contribute to the development of better tools and processes for OSS project management, leading to a more efficient review process. Our research aims to address this gap, potentially improve the contribution process. This work aims to further explore this varied landscape by addressing a key research question:

RQ: Do distinct features uniquely influence specific types of pull request rejections?

Results: Our research question delves into the varied dynamics of pull request rejections within OSS projects. We uncover that there are various reasons for rejection influenced by distinct factors, challenging the traditional, one-size-fits-all understanding of contribution evaluation. Through detailed analysis, we identify patterns where certain features consistently reduce the likelihood of rejection across multiple reasons, while others show significance only in specific contexts. This approach reveals the complexity of decision-making processes in OSS projects, emphasizing the need for a tailored examination of contribution evaluations.

We have made the following contributions:

1. We did a large-scale qualitative analysis and labeled 10,000 pull requests in 12 categories of rejection reasons based on the pull request comments.

2. We empirically observed the relationship between various pull request rejection reasons with different characteristics of the pull request, submitter, and the corresponding project on 52,829 pull requests.

3. We have provided a replication package of our work that includes our scripts, data sets, and results to facilitate further research and reproducibility

The rest of the thesis is organized as follows. Section 2 discusses the background and related work. Section 3 presents our study design, including the data acquisition, our labeling, and the statistical modeling. Section 4 answers our research question and shows the findings of our study. Section 5 discusses the results. Section 6 highlights the threats to validity, and Section 7 provides a conclusion for the thesis.

Chapter 2

Related Work

In this section, we discuss the literature related to our work.

2.1 Factors Influencing Pull Request Decisions

Several studies highlight the complex nature of pull request decisions in collaborative software projects. Soares et al. [11] identified key elements leading to internal contribution rejection, highlighting inexperience with pull request mechanisms, contribution complexity, and the locality of the modified artifacts. Legay et al. [12] presented quantitative evidence underscoring the importance of a contributor's experience, as recurrent contributors with a history of pull request submissions have higher acceptance rates.

Pooput et al. [13] used data mining techniques to extract patterns associated with pull request rejections. Their analysis revealed that certain code-related metrics, such as the number of file changes, commit count, and code duplication measures, were strongly associated with pull request rejection. This underscores the importance of code quality and structured contributions for pull request acceptance. Zhang et al. [1] identified the relationship between contributor and integrator as a predominant factor, with self-integration significantly increasing the odds of pull request acceptance.

The research by Tsay et al. [6], Terrell et al. [8], Rastogi et al. [9], and Nadri et al. [7] contribute to the understanding of non-technical factors in pull request evaluations. These studies shed light on potential biases that arise from the perceptible social identity of contributors, such as race, ethnicity, and gender, suggesting that pull request decisions may be influenced by the integrators' unconscious biases based on submitters' personal attributes.

2.2 Pull Request Rejection Reasons

Recent literature has expanded the perspective on pull request rejection reasons beyond code quality, encompassing a range of diversity factors. Nadri et al. [10] emphasize that social identity elements, such as race, gender, and geographical location, may influence pull request outcomes. This is supported by a qualitative analysis of the comments on non-merged pull requests, revealing that rejection reasons may exhibit underlying biases, highlighting the significance of the comments as evidence. Lenarduzzi et al. [14] challenge the assumption that code quality metrics (e.g., code smells and anti-patterns) are pivotal for pull request acceptance. Instead, they highlight the developer's reputation as a critical determinant, suggesting that the community's perception could overshadow technical assessments. They also hint at the potential for unexplored variables, such as test coverage and the bug proneness of contributions, which may offer additional insights into pull request acceptance criteria.

Gousios et al. [15] identify that the likelihood of a pull request being merged is considerably affected by whether it modifies recently modified code. They also find that a majority of pull request rejections stem from reasons inherent to the distributed nature of pull-based development, rather than technical shortcomings. Golzadeh et al. [16] delve into the communicative aspect of pull requests, presenting an empirical study on the discussions surrounding pull requests. The findings suggest that rejected pull requests are characterized by more extensive discussions, involving a greater number of participants and exchanges, indicating that communication plays a significant role in the pull request lifecycle.

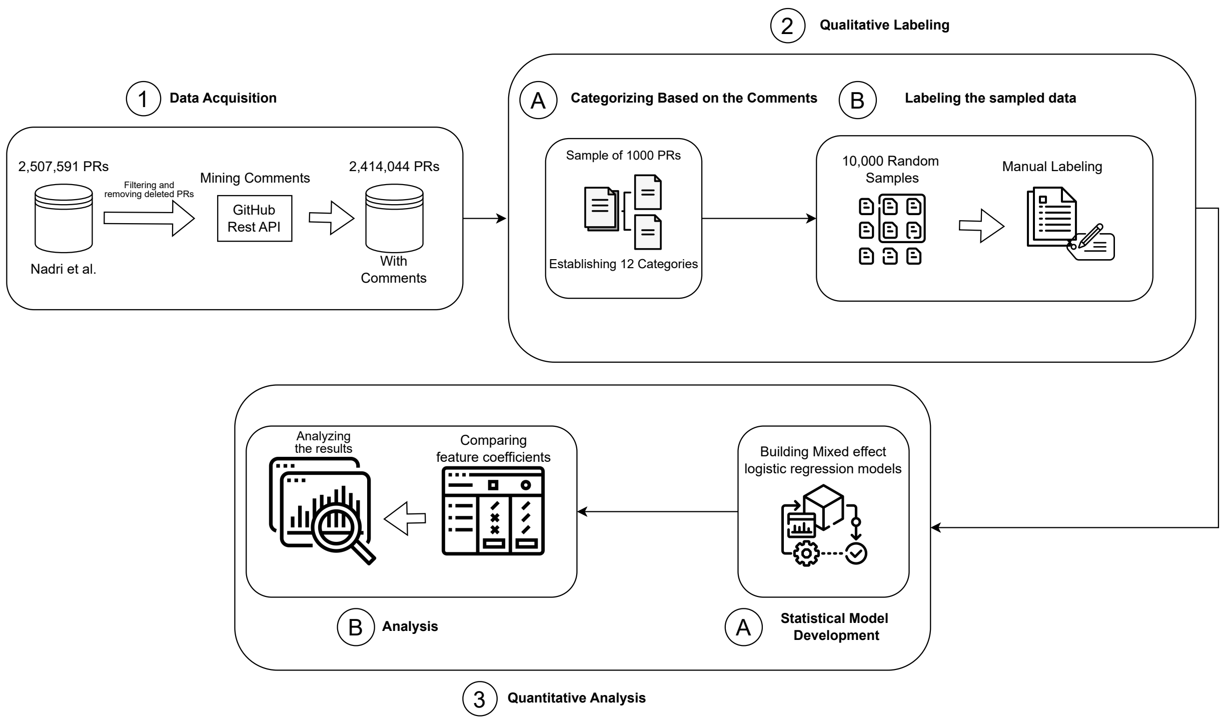
Steinmacher et al. [17] approach pull request rejection from the contributor's viewpoint. They uncover that pull requests are dismissed due to a misalignment between the developer's vision and the project team's objectives and the pull requests from contributors that are not part of the team, called quasi-contributors, are usually rejected with reasons such as replaced or duplicated. Their manual analysis also reveals additional factors, such as community relationships and contributor experience, as influential. Importantly, they highlight the emotional impact of rejection on contributors, with many expressing demotivation and reluctance to submit future pull requests.

Chapter 3

Study Design

In this section, we present our research rationale for selecting our questions and describe the different phases of our approach, including data acquisition, qualitative labeling, and quantitative analysis. illustrated in Figure 3.1. We take these steps to answer the following: **To what extent do various characteristics affect the rejection of a pull request?**
Motivation: Different factors influence the outcomes of pull requests, and understanding the important features leading to rejection is crucial in enhancing the collaborative process. By looking into the factors that affect pull request outcomes within a sampled dataset, we are following the steps of earlier studies but also tailoring our approach to fit the unique aspects and limitations of the data we have. Answering this establishes the foundation, and serves as the baseline to explain the dynamics between various pull request attributes and their eventual outcomes. This exploration provides a clearer understanding of the characteristics used in evaluating pull requests. Additionally, it establishes a solid foundation for the subsequent analysis presented in RQ.

RQ: Do distinct features uniquely influence specific types of pull request rejections?
Motivation: Understanding the dynamics of pull request rejections is essential in open-source software (OSS) development. Each pull request rejection reason might be influenced by a distinct set of factors. In this research question, we aim to examine the differing features associated with various types of pull request rejections and their potential impact on how contributions in OSS are perceived and evaluated. Conversely, consistent results could underscore the broader applicability and reliability of the general model, streamlining the predictive modeling approach for pull request outcomes.



2 Qualitative Labeling
1 Data Acquisition
2,507,591 PRs
Filtering and removing deleted PRs
Mining Comments
GitHub Rest API
2,414,044 PRs
With Comments
Nadri et al.
A Categorizing Based on the Comments
B Labeling the sampled data
Sample of 1000 PRs
Establishing 12 Categories
10,000 Random Samples
Manual Labeling
Analyzing the results
Comparing feature coefficients
Building Mixed effect logistic regression models
B Analysis
A Statistical Model Development
3 Quantitative Analysis

Figure 3.1: Steps for Methodology

3.1 Data Acquisition

Nadri et al. [7] conducted an empirical study on a dataset comprising 2,507,591 pull requests from 37,762 distinct projects was curated which provides different characteristics on pull requests, the submitters and the integrators which were chosen from three sources [6, 9, 18]. Building on their foundation, we excluded pull requests that were either removed or made private within their repositories. Through data mining GitHub REST API[1], we collected additional features that revolved around pull request descriptions and comments. In total, we have a refined dataset comprising 2,414,044 pull requests from 36,358 projects, as shown in the table 3.1.

[1]https://docs.github.com/en/rest

Table 3.1: Summary of the Datasets

Number of	**Original**	**Filtered**	**Sampled**
Projects	37,762	36,358	3,931
Merged PRs	2,039,601	1,980,653	42,829
Non-merged PRs	467,990	433,391	10,000
Total PR	2,507,591	2,414,044	52,829

3.2 Feature Selection

To understand the reasons behind pull request rejections, it's crucial to identify the associated characteristics and features. Previous studies have organized these features into three main categories: the characteristics of the project, those of the collaborator, and the specific attributes of the pull request itself, drawing on research in bug triaging, developer recommendation, and pull request and patch acceptance [18, 7].

Leveraging the dataset from Nadri et al. [7], we adopted the same features they analyzed but focused exclusively on those available to collaborators at the time of pull request submission. Consequently, we excluded the feature indicating the number of comments due to its unavailability at submission. We also disregarded ethnicity-related features, such as the contributor's and integrator's ethnicity, as they fell outside our research scope. This led to the selection of 15 pertinent features, detailed in table 3.3.

3.3 Qualitative Labeling

After collecting the necessary data, we build upon the work conducted by Nadri et al.[10], who conducted a qualitative study analyzing whether there is evidence of bias based on perceptible race in the written comments of non-merged pull requests on GitHub. Their analysis focused on the reasons provided by GitHub developers for rejecting 556 contributions. In our research, we extended their analysis by include the following components:

3.4 Categorizing based on the comments

In their study, Nadri et al. [7] identified four main reasons why a pull request might not be merged: "Successful", "Rejected", "Replaced", and "Resolved". They further categorized the rejection reasons into eight distinct categories. To gain a deeper understanding of these reasons, two researchers in our study reviewed the same 556 contributions that Nadri et al. [10] studied. Subsequently, an additional 1,000 pull requests were randomly chosen for further investigation, potentially expanding on these rejection reasons. This effort results in the creation of 12 distinct categories explaining why a pull request does not get merged. Detailed information regarding these categories is provided in Table 3.2.

3.4.1 Labeling the sampled data

In the subsequent step of our study, we sample data for labeling. To understand the reason behind the rejection, we randomly selected 10,000 samples. Two researchers were tasked with labeling and classifying each pull request based on the comments they contained.

To ensure the reliability of the labels assigned by the two researchers, both researchers independently labeled 2,000 samples. This joint effort yielded a Cohen's Kappa score of 77.6%. According to standards set by Landis et al. [19] and the comprehensive evaluation of Cohen's Kappa by Sun [20], this score indicates a substantial level of reliability within our labeling process. To address discrepancies in the categorization of rejection reasons for certain pull requests, the two researchers engaged in discussions to resolve the differences and arrived at an agreement for those cases. The decision to work with 10000 randomly chosen samples stemmed from our plan to allocate a dedicated 10-week period exclusively for data labeling. Following a preliminary assessment with a smaller sample size, we estimated that approximately 10,000 pull requests could be labeled within this 10-week timeframe. Consequently, our final sample count reached 10,000 pull requests.

3.5 Quantitative Analysis

The last step in our approach involved a thorough quantitative examination of our efforts. Guided by our research question, we construct statistical models and extracted findings that align with our objectives.

3.5.1 Statistical model development

To understand how various features of a pull request influence specific reasons for rejection, we developed two kinds of mixed effect logistic regression models [21], one for analyzing

Table 3.2: Pull Request Rejection Categories

Category	**Reasoning**	**Example of the last comment before closing**
Chaotic	PRs are unclear with numerous changes and commits. Often contains a lot of changes made at the same time.	”This includes too many changes.”, ”these fixes into 1.5.1, but the amount of changes for the other configs is too big to be viable.”
Duplicate	Content of the PR is already covered in a preceding PR. It is mentioned that the functionality has been covered elsewhere.	”Duplicate, sorry”, ”It seems like a dup of an old pull request, closing.”, ”We can close this PR in favor of another PR”
Merge Conflict	Conflicts in code arise due to simultaneous or competing changes. Could result in build errors, integration issues, or testing problems.	”There are merge conflicts here, please rebase.” ”deal with merge conflicts and I merge this shortly.”, ”Can you rebase and fix the conflicts? ”
No Comment	No comment exists in PR.	N/A
No Reason	The pull request gets closed without any provided insight. The PR does not fit any specific rejection category.	”Ok. So I am closing this now..” ”I am working on this PR.”
Not PR	The content does not describe a typical PR but rather a checklist. Might look like a suggestion rather than a proper PR. Contains phrases such as ”not a bug report” or ”make proper pr”.	”Closing, probably spam”, ”Pull requests are not to report issues.”, ”This is not an issue section”
Quality	PR does not meet the project’s quality requirements. Have code style problems or other quality concerns. Often suggests that improvements or alterations are needed.	”A validator makes sense. I’m closing this.”, ”Buggy function. On the algorithm which is O(n!).”, ”Closing due to CLA issues.”
Replaced	Another PR or action supersedes the current PR. References other PR or mentions the work is superseded elsewhere.	”Being dealt with by a new pull request”, ”I just saw that you made a new PR. Closing this.”, ”Closing in favor of a new PR”
Resolved	Changes have been manually merged or applied in a release. The integrator might have made the change themselves. Parts of changes have been chosen, has the word ”cherry-picked”.	”Thanks, it’s been resolved.”, ”I rebased against origin and resolved this manually.”, ”Cherry-picked into [commit], thanks!”
Stale	Closed due to inactivity from the submitter. The time between the last comment and PR closing is often long. Contains phrases such as ”Closing due to inactivity” or ”Very Old”.	”Closing this old pull request.”, ”closing as no reply.”, ”This is a stale pull request.”
Successful	PR is approved but closed to maintain commit history. Changes exist in another commit with credit given. Often contains positive affirmations or acknowledgments.	”merged in [commit number]”, ”Landed in [commit number], thank you!”, ”Merged [commit number], Thanks again for the fix”
Unnecessary	Deemed unnecessary by the project’s developers. Interferes with the developer’s plan. Changes should be made in another branch.	”This is not a bug, please use the suggestion site.”, ”It is fine how it is I think. It’s common for websites.”, ”this is not relevant if you are not cloning all of it”

Table 3.3: Independent Variables

Feature	**Literature**	**Description**
		Project Characteristics
repo_pr_tenure_mnth	[6],[9]	Numerical variable that indicates how long (in month) the repository has been active before the examination of pull requests.
repo_pr_popularity	[6],[9],[18]	Numerical variable that indicates how popular the repository is based on the number of watchers.
repo_pr_team_size	[6],[9],[18]	Numerical variable that indicates the number of users associated with the repository.
perc_external_contribs	[9],[18]	Numerical variable that indicates the percentage of contributors outside the project.
		Collaborator Characteristics
prs_experience	[9],[18]	Numerical variable that indicates the experience of the submitter by the number of pull requests submitted by them.
prs_succ_rate	[9],[18]	Numerical variable that indicates the success rate of the submitter when submitting a pull request.
prs_popularity	[6],[9],[18]	Numerical variable that indicates the popularity of the submitter by measuring the number of followers.
prs_watched_repo	[6],[9]	Binary variable that indicates if the submitter is watching the repository or not.
prs_tenure_mnth	[9]	Numerical variable that indicates how long (in months) the submitter's account has been active.
prs_main_team_member	[6],[9],[18]	Binary variable that indicates if the submitter is a main member of the repository
prs_followed_pri		Binary variable that shows if the submitter is following the integrator or not.
		Pull Request Characteristics
pr_files_changed	[6],[9],[18]	Numerical variable that indicates how many files have been changed while submitting the pull request.
pr_lines_changed		Numerical variable that indicates how many lines have been changed while submitting the pull request.
commit_counts	[9],[18]	Numerical variable that indicates how many commits exist in the pull request.
intra_branch	[9],[18]	Binary variable that indicates whether the pull request was made intra branch.

sampled dataset and the other for specific rejection reasons. This approach considers measurements from the same group as random effects. Submitter identity and repository identifier were incorporated as random effects, a methodology also adopted by Nadri et al. [7].

These models were designed to address our research question, with the pull request's status (merged or non-merged) as the dependent variable, and features related to submitters, closers, and project repository as independent variables.

To build our models, we required a balanced mix of both merged and non-merged pull requests. While we specifically labeled 10,000 non-merged pull requests, our approach for including merged pull requests was more representative. We aimed to mirror the actual distribution of data by sampling merged pull requests based on the ratio of non-merged to merged pull requests observed in each repository. For instance, in a repository present in our labeled dataset with a ratio of 3 non-merged to 9 merged pull requests, we maintained a sampling ratio of 1:3. This meant that for every non-merged pull request from this repository, we randomly selected 3 merged pull requests. We refer to this method as ratio sampling, ensuring our dataset accurately reflects the proportions of merged and non-merged pull requests across different repositories.

We employed the glmer function from the R package lme4, a generalized linear mixed-effect model [21] to craft these regression models. We established two types of models:

- **Model of sampled dataset:**
  To answer RQ, we developed a model using the labeled dataset of 10,000 pull requests and random sampling based on our ratio sampling which was explained before and resulted in 42,829 sampled merged pull requests. This model uses the pull request's status as the dependent variable, with variables listed in Table 3.3 serving as independent factors. The intent is to understand pivotal features influencing pull request rejections and establish a baseline for each rejection reason model.

- **Models for specific rejection reasons:**
  After building our first model, we constructed different datasets, each containing contributions with a certain rejection reason as one part of the dataset, and the other parts consist of randomly selected merged pull requests with ratio sampling from the same repositories to minimize bias, these are the same merged samples that we gathered in our base model. This resulted in 12 different models, corresponding to each pull request rejection reason that we have identified in previous steps. The resulting models and their distribution are in table 3.4

Table 3.4: Distribution of Pull Requests in Different Models

Model	**Merged**	**Non-merged**	**Total**
Chaotic	297	42	339
Duplicate	2,010	416	2,426
Merge Conflict	601	119	720
No Comment	4,109	928	5,037
No Reason	2,612	515	3,127
Not PR	112	24	136
Quality	3,562	702	4,264
Replaced	6,298	956	7,254
Resolved	4,858	1,195	6,053
Stale	1,616	309	1,925
Successful	6,085	2,736	8,821
Unnecessary	10,669	2,058	12,727
Total PR	42,829	10,000	52,829

3.5.2 Analysis

In the final step of our study, we analyze the results gathered from training our mixed-effect logistic regression models. This analysis allows us to discern the impact of various predictors and understand their significance in the context of the models' performance, which will be discussed in Section 4. This is done with three approaches in the specified order for our research question:

1. **Variance Inflation Factors (VIF):** After building our models, we assess multicollinearity through Variance Inflation Factors (VIF) analysis, using the "VIF" function from the car package in R [22]. We set a VIF threshold at 3 [7], an intermediate value considering previous thresholds of 5 or 2 in software engineering studies [23, 1]. This approach helps us identify and mitigate multicollinearity amongst the independent variables.

2. **Statistical Significance:**

   In our analysis, to indicate the strength of evidence against the null hypothesis, we address statistical significance by evaluating the p-value; a lower p-value indicates a higher statistical significance [24, 9, 25]. It is also worth noting that while looking at results, practical importance should also be considered[26]. For instance, a feature with a low p-value but a near-one odds ratio may not be as practically significant as one with a higher odds ratio.

3. **Odds ratio:** Drawing on established methodologies from previous research [7, 9, 6, 1], we apply odds ratio analyses to understand the relationship between our dependent and independent variables. This approach enables us to distinguish between interpretations for categorical predictors, where the odds ratio is compared to the default level of the categorical variable, and for continuous predictors, where the odds ratio indicates the increase or decrease in the odds of acceptance for a unit increase in the factor [6, 9].

   In this case, a unit of each measure is one standard deviation from the log-transformed variables or the presence of a dichotomous variable. An odds ratio above 1 indicates an increased likelihood of acceptance with each unit increase in the feature, while an odds ratio below 1 suggests a decrease, in a way where if the odds ratio is greater than 1, it indicates an increased likelihood of the event occurring. The percentage change for the odds ratio can be calculated as $(OddsRatio - 1) * 100$. For example, an odds ratio of 1.50 means there is a 50% increase in the odds of the event occurring compared to the baseline. It is important to note that in interpreting these odds ratios, an odds ratio close to 1 might not be practically significant despite its statistical significance.

Chapter 4

Results

In this section, we present the approach and findings for our baseline model and research question.

4.1 What is the relationship between characteristics and pull request rejection in the sampled data set?

Approach: The objective of this section is to analyze the different characteristics that are important when it comes to deciding the outcome of a pull request and whether it is merged or not. To achieve this, we curated a dataset comprising 52,829 pull requests. This dataset consists of 10,000 non-merged pull requests, and 42,829 merged pull requests. These were selected based on the ratio of non-merged to merged pull requests in our filtered dataset, ensuring both samples come from the same repositories. We then apply analyses based on three approaches: Variance Inflation Factors (VIF), Statistical Significance, and Odds Ratio.

Table 4.1 presents the results of our sampled dataset, revealing several key insights into the factors influencing pull request outcomes.

Table 4.1: Important Features for RQ

Feature	**Coeff.**	**P value**	**Odds Ratio**
Pull Request Characteristics			
commit_counts	0.11	2.25e-09 ***	1.12
pr_lines_changed	0.09	0.0013 **	1.09
pr_files_changed	0.00	0.9584	1.00
intra_branch1	-0.84	<2e-16 ***	0.43
Project Characteristics			
repo_pr_tenure_mnth	0.26	<2e-16 ***	1.29
perc_external_contribs	-0.08	3.47e-06 ***	0.92
repo_pr_team_size	-0.46	<2e-16 ***	0.63
repo_pr_popularity	-0.59	<2e-16 ***	0.55
Collaborator Characteristics			
prs_experience	0.26	<2e-16 ***	1.30
prs_succ_rate	0.04	0.0515	1.04
prs_tenure_mnth	0.00	0.9378	1.00
prs_popularity	-0.13	7.68e-10 ***	0.88
prs_followed_pri1	-0.30	<2e-16 ***	0.74
prs_watched_repo1	-0.47	<2e-16 ***	0.62
prs_main_team_member1	-0.85	<2e-16 ***	0.43

4.1.1 Pull Request Characteristics

The feature indicating the number of line changes in a pull request (**pr_lines_changed**), suggests that pull requests with more line changes are 9% more likely to be rejected. This finding aligns with insights from Bosu et al. [27], which indicate that larger pull requests, being more challenging to review, have a higher risk of rejection. It also reinforces the idea that extensive changes could overwhelm reviewers, potentially leading to misunderstandings. This is because larger pull requests are tougher to review thoroughly, whereas smaller ones tend to have a higher acceptance rate [27, 15]. Moreover, the **pr_files_changed** feature, with an odds ratio of 1, as well as not being a statistically significant feature, shows little impact on rejection probability, indicating that the number of files changed is not a significant factor in pull request evaluation. The **commit_counts** feature, which correlates with a 12% increase in rejection likelihood, suggests a modestly higher risk of rejection for pull requests with more commits. Finally, the **intra_branch** variable demonstrates a significant impact on rejection likelihood. Pull requests made within the same branch are 57% less likely to be rejected, suggesting they are perceived as less risky or more manageable.

4.1.2 Project Characteristics

In terms of project characteristics, both team size (**repo_pr_team_size**) and project popularity (**repo_pr_popularity**) are associated with a decrease in the likelihood of pull request rejection. Specifically, for every unit increase in **repo_pr_team_size**, the likelihood of rejection decreases by approximately 37%, and for each increase in **repo_pr_popularity** feature, the likelihood of rejection decreases by about 45%. These figures imply that pull requests in larger or more popular projects are less likely to be rejected, potentially due to these projects' ability to integrate contributions more efficiently, possibly thanks to better resources or more established processes. This observation contrasts with Gousios et al.'s [15] suggestion that larger teams might face more rejections due to coordination challenges. Furthermore, the proportion of external contributors (**perc_external_contribs**) is linked to a slight decrease in the likelihood of pull request rejection. A higher proportion of external contributions correlates with an 8% decrease in rejection rates. The feature with the most significant impact on pull request rejection is **repo_pr_tenure_month**. Each additional month in project tenure increases the likelihood of rejection by 29%, implying that more established projects on GitHub often have stricter standards for accepting pull requests; This pattern could reflect a higher bar for contributions in projects with longer histories or greater maturity.

4.1.3 Collaborator Characteristics

Contrary to the expected benefits of experience, **prs_experience** indicates that seasoned contributors are actually more likely to face pull request rejections. Specifically, for seasoned contributors, there is a 30% increase in the likelihood of their pull requests being rejected compared to less experienced contributors. This finding implies that, despite their experience, such contributors' pull requests might still undergo stringent scrutiny for various reasons which we have discussed in the results of RQ. Conversely, certain attributes are associated with a lower probability of pull request rejection. Specifically, being a main team member (**prs_main_team_member1**) is associated with a 57% decrease in rejection likelihood, following the integrator (**prs_followed_pri1**) correlates with a 26% decrease, and being a popular submitter (**prs_popularity**) is linked to a 12% decrease in the chances of rejection. These findings underscore the importance of established relationships and recognition within the project community. They align with Tsay et al.'s [6] and Thung et al.'s [28] observations regarding the significance of sustained engagement and social structures within project teams.

Summary: Strong social ties within the project community substantially lower the likelihood of pull request rejection. Conversely, the technical aspects, such as the scope of changes, require careful management to minimize rejection risks. These insights provide a foundational understanding for further investigation into the factors influencing PR evaluations.

4.2 RQ: Do distinct features uniquely influence specific types of pull request rejections?

Approach:

To tackle our Research Question, we construct 12 distinct models, each corresponding to a specific reason for pull request rejection. We ensure consistency in our approach by maintaining the same architectural framework across all models. Figure 4.1 showcases a heatmap that visualizes the *statistically significant* features for each rejection reason, side by side with our baseline model. In this heatmap, the odds ratios of significant features are represented in each cell. Cells with a deeper shade of red suggest a higher likelihood of rejection influenced by the respective feature, while cells tending towards blue indicate an increased likelihood of acceptance for that feature in the specific rejection category. This analysis aims to clarify the meaning behind these variables for each rejection category and to compare their impacts, building on the baseline model.

Here, we present an overview of the impact of features across categories, setting the stage for a more detailed discussion in section 5.

4.2.1 Pull Request Characteristics

In analyzing pull request outcomes, most features align with the base model's trends. **commit_counts** indicate a higher rejection likelihood for 5 out of the 12 categories, having

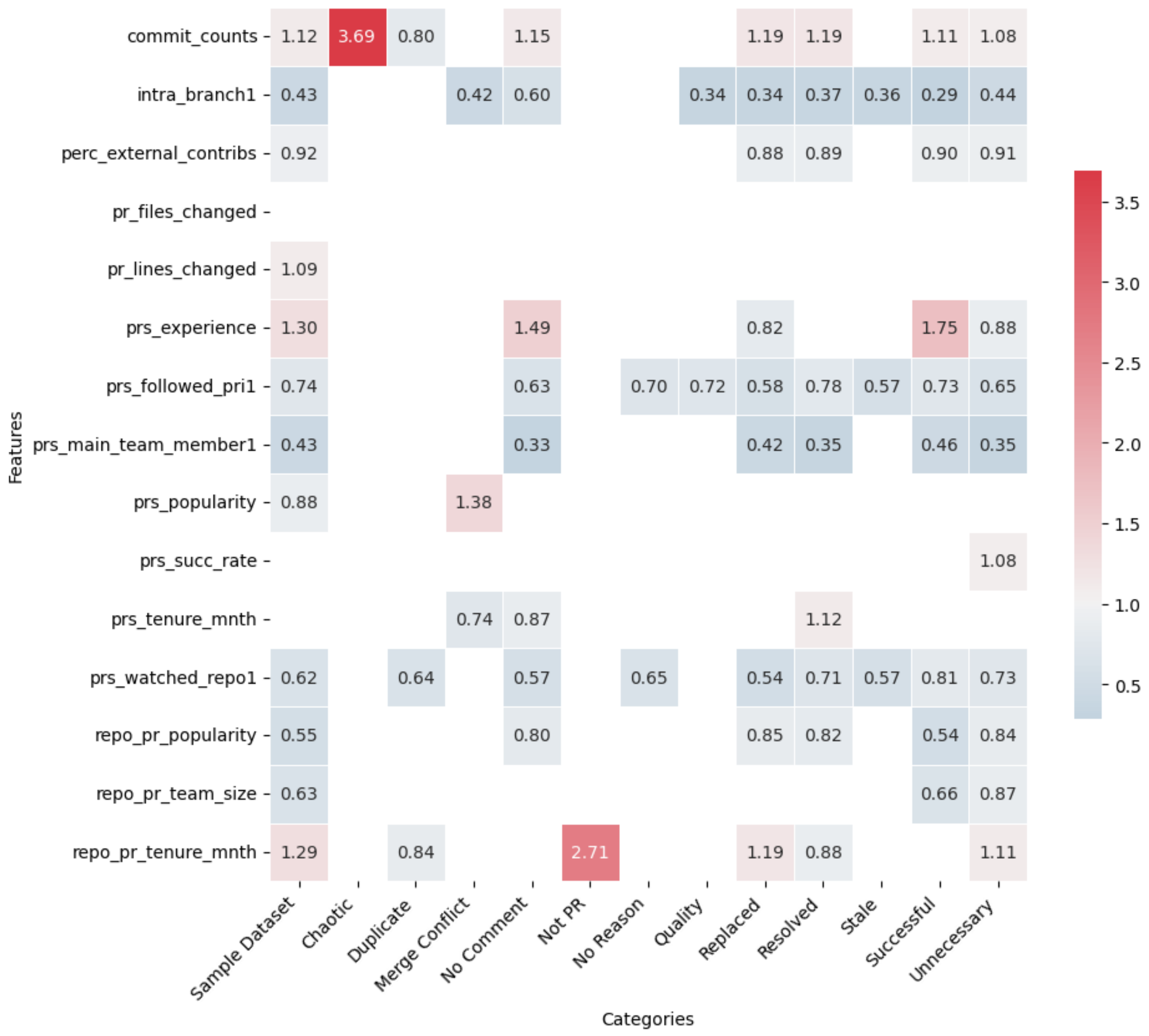


Figure 4.1: Results of Each model

the same direction as the base trend. However, in this category, duplicate pull requests highlight a unique trajectory and contrast with the general trend, which means a lower rejection likelihood. Moreover, **pr_files_changed** and **pr_lines_changed** consistently show minimal impact across all categories, reinforcing their limited influence observed in the base model. **Intra_branch** consistently demonstrates a lower rejection likelihood, indicating a preference for intra-branch pull requests in 8 out of 12 categories.

4.2.2 Project Characteristics

In the context of project characteristics, our findings reveal how certain features impact pull request outcomes in diverse ways. **repo_pr_team_size** shows a consistent trend where larger teams are associated with a lower rejection likelihood, as seen in successful and unnecessary categories, aligning with the base model. **repo_pr_popularity** demonstrates that more popular projects generally have a lower rejection rate, a pattern echoed across five categories, including the base.

repo_pr_tenure_mnth indicates a general increase in rejection likelihood with longer project tenure, significantly in the not PR category, and similarly impacts replaced and unnecessary categories. Yet, for resolved and duplicate pull requests, longer tenure correlates with a lower rejection likelihood, marking a departure from the general trend. **perc_external_contribs** consistently suggests that projects with more external contributions face a lower rejection likelihood, a trend maintained across eight categories including the base

4.2.3 Collaborator Characteristics

In examining the influences of collaborator characteristics on specific pull request rejection categories, several features exhibit notable trends. The features **prs_main_team_member1** and **prs_followed_pri1** align with the base model across all categories. However, the submitter's experience and popularity deviate from the observed base model trend. For **prs_experience**, merge conflict pull requests suggest that a more popular submitter increases the likelihood of rejection. Conversely, for **prs_popularity**, replaced and unnecessary pull requests indicate that a higher experience level of the submitter lowers the risk of rejection. Two features related to the submitter's previous success rate and account duration are not statistically significant in the base model but show significance in specific categories. Notably, **prs_succ_rate** is only significant in unnecessary pull requests, with each unit increase resulting in a minor rise in the likelihood of the pull request being rejected. Regarding **prs_tenure_mnth**, instances with no comments and conflict pull requests indicate a lower chance of the pull request being rejected when the submitter has a longer tenure at GitHub. However, in resolved pull requests, a longer tenure increases the risk of rejection.

RQ Summary: The results show that some features diverge from the base model's trend. In duplicate pull requests, more commits and longer project tenure decrease rejection likelihood, which contrasts with the baseline. Some features gain significance for specific rejection reasons. For example, the contributor success rate and the duration on GitHub show significance in categories like unnecessary pull requests and merge conflicts. Meanwhile, many categories and features maintain the same influence on rejection as the base model, such as an intra-branch pull request or a contributor watching the repository, all pointing to a lower likelihood of rejection with each unit increase.

Chapter 5

Discussion

In the previous section, we systematically analyze the influence of various features on pull request outcomes across different categories. In the discussion section, we conduct an in-depth analysis of each category of rejection reasons to shed light on the decision-making process in open-source software projects.

5.1 Chaotic

In chaotic pull request rejections, our findings highlight the number of commits as the primary factor influencing outcomes. According to the base model, an increase in the number of commits raises the chance of a pull request being rejected by 12%. This suggests a general preference for pull requests with fewer, yet more substantial commits. However, in this category, the effect is much more pronounced, with a likelihood increase of 269%. Essentially, a high number of commits in a pull request significantly boosts the likelihood of the request being perceived as disorganized or overly complex, leading to its rejection.

Takeaway: By looking at the chaotic pull requests, it is crucial to manage the number of commits to maintain clarity in the pull request, ensuring it is well-received and effectively reviewed by project maintainers.

5.2 Duplicate

In the duplicate category of pull request rejections, our analysis brings to light the significance of some key factors such as the number of commits, if the contributor is following the integrator. Diverging from the base model where each additional commit increases the likelihood of rejection by 12%, the duplicate pull requests present a different scenario. Here, the impact of the number of commits is less pronounced with 20% less likelihood of rejection, suggesting that the number of commits in a pull request is not as strongly associated with its rejection for duplication reasons. This indicates that, in cases of duplication, the focus may shift away from the volume of commits to other aspects of the pull request.

Regarding whether the contributor is following the integrator, its influence in the duplicate pull requests, with a 36% more chance of acceptance, closely replicates its impact in the base model. This consistency across different contexts implies that following the repository where change is happening has a uniform effect on pull request outcomes. In the context of duplicates, a lower odds ratio might suggest that submitters who actively follow the project are slightly less prone to having their contributions rejected as duplicates. This could be attributed to their broader exposure to the project, potentially making them more aware of existing contributions and reducing the likelihood of submitting overlapping content.

Takeaway: Developers can focus on community engagement and awareness of ongoing work within the project to minimize duplication and the developers who are more engaged and informed about a project's activities are better positioned to contribute effectively without redundancy.

5.3 Merge Conflict

For our model where pull requests are closed due to merge conflicts, the significant features are the popularity of the contributor, the tenure of the contributor, and if the pull request is intra-branch. popularity of the contributor shows a notable shift here compared to the base model. the pull requests with merge conflict present a contrasting picture and show that pull requests from popular submitters are 38% more likely to face rejection due to merge conflicts. This increase could imply that contributions from popular contributors and ones with more followers, which could be more complex or extensive, are more susceptible to integration challenges, potentially leading to merge conflicts. With each added month to the tenure of the contributor, it is 26% less likely for the pull request to get rejected because of merge conflicts. possibly due to increased familiarity with projects. It is worth mentioning that this feature is not significant in our base model.

Pull requests from the same branch are 58% less likely to be rejected, a similar pattern is also found in the base model. The lower odds ratio in this category could imply that intra-branch pull requests are less prone to merge conflicts, likely due to the simplicity and directness of changes made within a single branch.

Takeaway: Popular contributors should exercise caution when syncing their work, as their pull requests are prone to merge conflicts. Developers are advised to follow good branch management practices and consolidate changes into a single branch to avoid integration issues.

5.4 No Comment

In the no comment rejection category, the contributor experience feature indicates that more experienced contributors have a 49% higher chance of having their pull request rejected without comments compared to the base model, where the likelihood of rejection is 30% higher. During our labeling process, we observed that many pull requests from experienced contributors are merged into different branches, with their commit numbers visible on the pull request page despite the absence of comments. Their pull requests could be clear and concise and integrate directly into other branches, bypassing the need for extensive discussion.

Based on the number of commits, our results indicate that each additional commit marginally increases the likelihood of a pull request being rejected without any comments by 15%. This trend is slightly higher compared to the base model, suggesting a subtle shift towards preferring fewer commits in cases where pull requests are rejected without explanation. Regarding the remaining features, our results reveal a consistent pattern similar to our base model. This consistency indicates that factors such as project popularity if the contributor is watching the project, the duration of a contributor's involvement, and the relationship between the submitter and the integrator have a similar impact on the likelihood of a pull request being rejected without comment as in general pull request evaluations.

Takeaway: In analyzing the pull request without comments, we identify that closures without explicit feedback often signify procedural outcomes, such as integrating contributions into other branches, rather than direct rejections. Submitters with more experience tend to submit clearer and more concise pull requests. Emphasizing quality and well-structured submissions is crucial for successfully navigating the acceptance process in OSS projects. This suggests that silent closures might still represent positive outcomes.

5.5 Not PR

For the pull requests that are not considered a pull request by the reviewer, the only significant feature is how long the repository has existed. This feature, significantly impacts the rejection reason with 193% more likelihood of rejection, much higher than in the base model with 29% more chance. This suggests that longer-tenured projects are more likely to receive, and accordingly reject pull requests for not being pull requests, either due to clearer guidelines and expectations or the fact that they face a high amount of pull requests which are issues or general queries.

Takeaway: By looking at results from pull requests that are not genuine, we find that understanding and adhering to a project's guidelines becomes increasingly crucial in longer-term projects. These projects tend to have stricter conditions. Familiarizing oneself with a project's specific submission criteria can significantly enhance the acceptance rate of pull requests.

5.6 No Reason

In the pull requests where no reason is given for pull request rejection, our results indicate a 30% and 35% decrease, respectively, in the likelihood of rejection for pull requests where the submitter follows the integrator or watches the repository. Compared to the base model, which shows a decrease of 26% and 38%, we observe slightly less pronounced effects in instances where no reason is specified. This suggests that following the integrator and watching the repository reduces the chances of rejection, regardless of whether a reason is provided. This could imply that these social connections and engagement levels with the repository play a consistent role in different rejection contexts, emphasizing the varied influence of community involvement on pull request outcomes.

Takeaway: When looking into pull requests that are closed without any explicit reason, involvement with the project and its contributors improves the visibility of your contributions and aligns with successful pull request strategies across various scenarios, including those where no explicit feedback is provided. Therefore, encouraging these connections can be a strategic move for contributors aiming to improve their pull request acceptance rates.

5.7 Quality

In the pull requests where quality issues are the concern, the results indicate a 28% and 66% decrease in the likelihood of rejection for pull requests where the submitter follows the integrator or the contribution is done in the same branch respectively.

For the feature indicating if the contributor is following the integrator, there is a slight difference between this category and our base model which suggests that submitters who follow the integrator are less likely to have their pull requests rejected due to quality concerns.

As for intra-branch pull requests, our results indicate a more significant decrease in rejection likelihood of approximately 66% in the quality category, compared to a 55% decrease in the base model. This suggests that intra-branch pull requests are less likely to be rejected for quality reasons, emphasizing the advantage of keeping contributions within the same branch.

Takeaway: From these insights from pull requests with quality issues, maintaining social connections with the project's integrators and focusing contributions within the same branch can significantly influence the acceptance of pull requests, particularly when quality is a concern.

5.8 Replaced

In replaced pull requests, the results reveal an intriguing trend in which experienced contributors have a lower probability of their pull requests being replaced by 18%. This finding contrasts sharply with the base model, where experienced submitters face a 30% increased risk of rejection. The result suggests that experienced contributors' pull requests are less likely to be replaced by new ones, possibly because of their better alignment with project objectives and a more profound understanding of project needs from previous contributions.

With a 19% increase in the change of rejection, it is indicated that longer project tenure slightly increases the likelihood of pull requests being replaced. This trend aligns with the base model's increase of 29%, suggesting a consistent relationship between project tenure and pull request outcomes across different contexts. Our findings suggest that in the context of replacement, the longevity of a project influences decision-making. Furthermore, the longer the project has existed, the higher the likelihood that the pull request gets replaced with a brand-new one meeting certain standards.

Takeaway: For developers, the replaced pull requests highlight two key takeaways. Firstly, contributions from experienced contributors are less likely to be replaced, emphasizing the value of aligning closely with project goals. Secondly, pull requests in longer-term projects have a higher replacement risk, pointing to the need to be careful with established standards. Additionally, it's important to consider that someone with less experience might not be able to directly influence their experience level in a short period, but they should be aware of these dynamics and their implications. Understanding these factors is crucial for newer contributors as they navigate project contributions and aim to meet the established criteria.

5.9 Resolved

In our resolved category, we observe a 12% decrease in the likelihood of rejection for each additional month of project tenure, compared to the base model where an increase in tenure is associated with a 29% increase likelihood of pull request rejection. This indicates that the longer a project has been active, the slightly less likely it is for new pull requests to be resolved, perhaps due to evolving or stricter project standards over time.

With each additional commit, there is a 19% increase in the likelihood of a pull request getting rejected. However, some parts of the introduced code are manually integrated into the codebase. The effect is more pronounced than the base model's 12% increase, indicating that pull requests with a higher number of commits are slightly more likely to include detailed work or corrections that lead to manual project integration.

With each additional month of the submitter's tenure, there's a 12% increase in the likelihood of pull requests being manually integrated. This could indicate that the experience or established trust of the contributor might play a role in decisions to manually integrate their contributions.

Takeaway: By examining resolved pull requests, we can suggest that to minimize reliance on manual integration, it is beneficial to understand the dynamics of project tenure. While these factors can influence the likelihood of manual resolution, developers should strive to align their submissions closely with project standards and practices to maximize the chances of direct acceptance.

5.10 Stale

Based on our results, in the stale category, pull requests from contributors who actively watch the repository have a 43% decrease in the likelihood of becoming stale, compared to a 38% decrease in the base model. This suggests that contributors who actively watch the repository are more engaged, possibly leading to faster responses to discussions and updates, therefore reducing the risk of the pull request becoming inactive.

Similarly, for pull requests from contributors who follow the integrator, there is a 43% decrease in becoming stale in this category. Compared to the base model's 26% decrease, this highlights the importance of direct engagement and social connections within the project. Following the integrator may facilitate improved communication and awareness of the project's needs, encouraging timely updates and responses.

Pull requests within the same branch show a notable 64% decrease in the likelihood of becoming stale, compared to a 57% decrease in the base model. This indicates that intra-branch contributions, which are inherently simpler and more focused, are less prone to inactivity or being outdated. These contributions may facilitate easier to review and integration by teams, resulting in quicker action and a decreased likelihood of the pull request being overlooked.

Takeaway: To prevent pull requests from going stale, developers are advised to actively track project changes by watching repositories and following integrators, facilitating quicker updates and interactions. Concentrating on intra-branch contributions can further simplify integration, reducing the chance of inactivity. However, our findings indicate that staying informed on project developments is the most effective strategy to prevent pull requests from becoming outdated. Active engagement and strategic submissions are essential for keeping the pull request away from being stale.

5.11 Successful

In the successful category, where pull requests are accepted but not directly merged, two features stand out. The results reveal a 75% increase in the likelihood of rejection compared to the base model's 30% increase for more experienced contributors. This suggests that pull requests from experienced contributors are more likely to be recognized for their value, even if not directly merged, reflecting quality of the pull request.

In the feature indicating the number of commits, the results show an 11% increase in the likelihood of a pull request being successfully closed but not directly merged for every additional commit. this is a close resemblance to the base model which suggests a 12% increase. This minor variance underscores a generally consistent influence of the number of commits on the outcomes of pull requests across various scenarios.

Takeaway: In the successful category, pull requests submitted from experienced contributors may not be directly merged, however, their contributions are often integrated in other ways. This reveals an evaluation process where direct acceptance serves as one indicator of a contribution's value assessment. It highlights the significant, albeit indirect, impact experienced contributors have on a project's development.

5.12 Unnecessary

In our unnecessary category, the results indicate that pull requests in projects with longer tenure have an 11% increased likelihood of being deemed unnecessary, compared to a 29% increase in the base model. This suggests that as projects mature, their guidelines and project plans become more defined, possibly leading to a higher standard for accepting changes.

An increase in the number of commits leads to an 8% higher chance of the pull request being considered unnecessary. This is slightly lower than the 12% observed in the base model, indicating that while the volume of changes remains a factor, its impact is less pronounced in determining the necessity of a pull request. Furthermore, the historical success rate of a contributor, while not a significant feature in the base model, shows a subtle increase of 8% in the likelihood of rejection for contributors with a higher success rate. This could imply that even historically successful contributors can propose changes that are not aligned with the current project direction or needs.

We also observe that experienced contributors face a 12% reduced chance of their pull requests being labeled as unnecessary, a notable deviation from the base model where their experience increases the risk of rejection by 30%. This shift suggests that although experienced contributors might face higher expectations in general which makes their changes less likely to be discarded due to being unnecessary.

Takeaway: When analyzing unnecessary pull requests, we observe that in projects with a longer history, there is slightly higher scrutiny on pull requests, potentially leading to a higher rate of marking them as unnecessary. However, contributions from experienced developers are less likely to be dismissed as unnecessary, underlining the value of aligning closely with the project's current needs and direction. This underscores the importance of understanding a project's context and making directly relevant contributions, especially in well-established projects.

Chapter 6

Threats to Validity

We break the threats into four parts, external, construct, internal, and conclusion validity, following the convention in empirical software engineering research. [29]

6.1 Construct Validity

This threat is related to the degree our measure captures. As discussed in [1], the measure of relative importance may change with different methods. Using mixed-effect logistic regressions and odds ratio might pose issues, such as overestimation with a small sample size [30, 31]. However, in the context of logistic regression, odds ratios are a common metric used to represent the strength and direction of the association between a feature and the outcome [32]. Previous works have considered odds as a measurement to explain the results of logistic regression models and compare different coefficients with each other to conclusion [33, 34].

Another potential threat concerns the frequency of pull request rejection types. Since certain occurrences might be infrequent, it could impact the reliability of the relationship between feature and outcome [35]. However, in this study, we address this issue by ratio sampling and having the same ratio across different models between merged and non-merged pull requests to handle this problem.

Another construct validity in our study involves the potential issues in how certain features, such as experience, are quantified. We used the number of pull requests submitted by an individual as a measure of experience. However, this metric might not fully capture the true breadth or depth of a contributor's experience, as there may be cases where experience does not directly correlate with the number of contributions. Despite this, such cases are expected to be infrequent and, therefore, unlikely to pose a significant threat to the study's overall construct validity.

6.2 Internal Validity

These threats are related to inferences about cause-effect relationships. The first threat is associated with potential bias introduced during the labeling process. To mitigate bias, two researchers independently labeled 2,000 cases, resulting in a substantial agreement with a kappa score of 77.6%.

The other threat is insufficient instances in some categories. This could be due to random sampling or could show that there are not enough instances of OSS contributions that fit the criteria of these classes.

Another internal threat would be our focus on the rejection reasons without considering whether these rejections are final or whether the issues identified in the pull requests have been addressed and resubmitted successfully in a different form. This aspect of the pull request lifecycle, whether rejected contributions are refined and reintegrated into the project. This offers a potential area for future research, where the subsequent paths of rejected pull requests could be explored to provide a more comprehensive view of the contribution process.

6.3 External Validity

These threats concern the generalization of our findings. In this study, we conduct an experiment on OSS projects extracted from the GitHub platform as introduced by Nadri et al. [7]. They have used Reporeapers [36] to select non-trivial projects, and GHTorrent [37] and select a subset of the dataset based on their research needs. We cannot assert generalizability beyond the projects included in this dataset, which include the projects in which the submitter and integrator are the same people, non-OSS projects and those not hosted on the GitHub platform.

6.4 Conclusion Validity

These threats concern the robustness of our conclusions when using odds ratios and coefficient estimates in logistic regression models. Since odds ratios compare the likelihood of an event occurring between different levels of a predictor variable[32], any changes in the predictor's distribution can influence their magnitude. When comparing across models, this can lead to discrepancies in the perceived importance of the same factor, possibly affecting the consistency and correctness of our conclusions. We mitigated these issues by including consistent model architecture and using the same data pre-processing across different models.

Chapter 7

Conclusion

In this study, we have expanded our understanding of pull request rejections in Open Source Software (OSS) projects by doing a large-scale analysis of the pull request comments across 12 distinct rejection categories. Our analysis reveals that rejection rationales are influenced by diverse, context-specific factors, challenging the traditional, oversimplified view of OSS contributions. Highlighting the critical need for nuanced interpretations of rejection reasons, offering substantial insights for contributors seeking to enhance engagement within OSS projects. By acknowledging specific rejection reasons, both contributors and maintainers can effectively navigate the complexities of the OSS ecosystem.

Overall, the reasons behind the outcome of pull requests in OSS projects span a broad range of factors, reflecting the rich and varied landscape of OSS contributions. Embracing the detailed nuances and complexities of these rejection reasons paves the way for higher rates of acceptance and deeper community involvement. This study suggests a detailed examination of pull request decisions, encouraging an approach that delves into the details of contribution dynamics within OSS projects.

References

[1] Xunhui Zhang, Yue Yu, Georgios Gousios, and Ayushi Rastogi. Pull request decisions explained: An empirical overview. *IEEE Transactions on Software Engineering*, 49(2):849–871, 2022.

[2] Rahul N Iyer, S Alex Yun, Meiyappan Nagappan, and Jesse Hoey. Effects of personality traits on pull request acceptance. *IEEE Transactions on Software Engineering*, 47(11):2632–2643, 2019.

[3] Farshad Kazemi, Maxime Lamothe, and Shane McIntosh. Exploring the notion of risk in code reviewer recommendation. In *2022 IEEE International Conference on Software Maintenance and Evolution (ICSME)*, pages 139–150. IEEE, 2022.

[4] Walt Scacchi. Free/open source software development: Recent research results and methods. *Advances in Computers*, 69:243–295, 2007.

[5] Jennifer Marlow, Laura Dabbish, and Jim Herbsleb. Impression formation in online peer production: activity traces and personal profiles in github. In *Proceedings of the 2013 conference on Computer supported cooperative work*, pages 117–128, 2013.

[6] 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*, page 356–366, Hyderabad India, May 2014. ACM.

[7] Reza Nadri, Gema Rodriguez-Perez, and Meiyappan Nagappan. On the relationship between the developer's perceptible race and ethnicity and the evaluation of contributions in oss. *IEEE Transactions on Software Engineering*, 48(8):2955–2968, August 2022.

[8] Josh Terrell, Andrew Kofink, Justin Middleton, Clarissa Rainear, Emerson Murphy-Hill, Chris Parnin, and Jon Stallings. Gender differences and bias in open source: Pull request acceptance of women versus men. *PeerJ Computer Science*, 3:e111, 2017.

[9] Ayushi Rastogi, Nachiappan Nagappan, Georgios Gousios, and André van der Hoek. Relationship between geographical location and evaluation of developer contributions in github. In *Proceedings of the 12th ACM/IEEE international symposium on empirical software engineering and measurement*, pages 1–8, 2018.

[10] Reza Nadri, Gema Rodriguez-Perez, and Meiyappan Nagappan. Insights into non-merged pull requests in github: Is there evidence of bias based on perceptible race? *IEEE Software*, 38(2):51–57, March 2021.

[11] 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.

[12] Damien Legay, Alexandre Decan, and Tom Mens. On the impact of pull request decisions on future contributions. *arXiv preprint arXiv:1812.06269*, 2018.

[13] Panthip Pooput and Pornsiri Muenchaisri. Finding impact factors for rejection of pull requests on github. In *Proceedings of the 2018 VII International Conference on Network, Communication and Computing*, pages 70–76, 2018.

[14] 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.

[15] 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.

[16] Mehdi Golzadeh, Alexandre Decan, and Tom Mens. On the effect of discussions on pull request decisions. In *BENEVOL*, 2019.

[17] Igor Steinmacher, Gustavo Pinto, Igor Scaliante Wiese, and Marco A Gerosa. Almost there: A study on quasi-contributors in open source software projects. In *Proceedings of the 40th International Conference on Software Engineering*, pages 256–266, 2018.

[18] Georgios Gousios and Andy Zaidman. A dataset for pull-based development research. In *Proceedings of the 11th Working Conference on Mining Software Repositories*, pages 368–371, 2014.

[19] J. Richard Landis and Gary G. Koch. The measurement of observer agreement for categorical data. *Biometrics*, 33(1):159, March 1977.

[20] Shuyan Sun. Meta-analysis of cohen's kappa. *Health Services and Outcomes Research Methodology*, 11(3–4):145–163, December 2011.

[21] D. Bates, M. Maechler, B. Bolker, and S. Walker. *lme4: Linear mixed-effects models using Eigen and S4*, 2023. R package version 1.1-34.

[22] Jacob Cohen, Patricia Cohen, Stephen G West, and Leona S Aiken. *Applied multiple regression/correlation analysis for the behavioral sciences*. Routledge, 2013.

[23] Casey Casalnuovo, Bogdan Vasilescu, Premkumar Devanbu, and Vladimir Filkov. Developer onboarding in github: the role of prior social links and language experience. In *Proceedings of the 2015 10th joint meeting on foundations of software engineering*, pages 817–828, 2015.

[24] Yue Yu, Gang Yin, Tao Wang, Cheng Yang, and Huaimin Wang. Determinants of pull-based development in the context of continuous integration. *Science China Information Sciences*, 59:1–14, 2016.

[25] Daniel J Benjamin, James O Berger, Magnus Johannesson, Brian A Nosek, E-J Wagenmakers, Richard Berk, Kenneth A Bollen, Björn Brembs, Lawrence Brown, Colin Camerer, et al. Redefine statistical significance. *Nature human behaviour*, 2(1):6–10, 2018.

[26] David T Lykken. Statistical significance in psychological research. *Psychological bulletin*, 70(3p1):151, 1968.

[27] Amiangshu Bosu, Jeffrey C Carver, Munawar Hafiz, Patrick Hilley, and Derek Janni. Identifying the characteristics of vulnerable code changes: An empirical study. In *Proceedings of the 22nd ACM SIGSOFT international symposium on foundations of software engineering*, pages 257–268, 2014.

[28] Ferdian Thung, Tegawende F Bissyande, David Lo, and Lingxiao Jiang. Network structure of social coding in github. In *2013 17th European conference on software maintenance and reengineering*, pages 323–326. IEEE, 2013.

[29] Claes Wohlin, Per Runeson, Martin Höst, Magnus C Ohlsson, Björn Regnell, and Anders Wesslén. *Experimentation in software engineering*. Springer Science & Business Media, 2012.

[30] Szilard Nemes, Junmei Miao Jonasson, Anna Genell, and Gunnar Steineck. Bias in odds ratios by logistic regression modelling and sample size. *BMC medical research methodology*, 9:1–5, 2009.

[31] Huw Talfryn Oakley Davies, Iain Kinloch Crombie, and Manouche Tavakoli. When can odds ratios mislead? *Bmj*, 316(7136):989–991, 1998.

[32] Susan M Hailpern and Paul F Visintainer. Odds ratios and logistic regression: further examples of their use and interpretation. *The Stata Journal*, 3(3):213–225, 2003.

[33] Jiaxin Zhu, Minghui Zhou, and Audris Mockus. Effectiveness of code contribution: From patch-based to pull-request-based tools. In *Proceedings of the 2016 24th ACM SIGSOFT International Symposium on Foundations of Software Engineering*, pages 871–882, 2016.

[34] Caio Barbosa, Anderson Uchôa, Daniel Coutinho, Filipe Falcão, Hyago Brito, Guilherme Amaral, Vinicius Soares, Alessandro Garcia, Baldoino Fonseca, Marcio Ribeiro, et al. Revealing the social aspects of design decay: A retrospective study of pull requests. In *Proceedings of the XXXIV Brazilian Symposium on Software Engineering*, pages 364–373, 2020.

[35] Douglas G Altman, Jonathon J Deeks, and David L Sackett. Odds ratios should be avoided when events are common. *Bmj*, 317(7168):1318, 1998.

[36] Nuthan Munaiah, Steven Kroh, Craig Cabrey, and Meiyappan Nagappan. Curating github for engineered software projects. *Empirical Software Engineering*, 22:3219–3253, 2017.

[37] Georgios Gousios. The ghtorent dataset and tool suite. In *2013 10th Working Conference on Mining Software Repositories (MSR)*, pages 233–236. IEEE, 2013.