On Wasted Contributions: Understanding the Dynamics of Contributor-Abandoned Pull Requests

A Mixed-Methods Study of 10 Large Open-Source Projects on GitHub

SAYEDHASSAN KHATOONABADI, Concordia University, Canada
DIEGO ELIAS COSTA, Concordia University, Canada
RABE ABDALKAREEM, Carleton University, Canada
EMAD SHIHAB, Concordia University, Canada

Pull-based development has enabled numerous volunteers to contribute to open-source projects with fewer barriers. Nevertheless, a considerable amount of pull requests (PRs) with valid contributions are abandoned by their *contributors*, wasting the effort and time put in by both their contributors and maintainers. To gain a more comprehensive understanding of the underlying dynamics of contributor-abandoned PRs, we conduct a mixed-methods study using both quantitative and qualitative methods. We curate a dataset consisting of 266,039 PRs including 4,450 abandoned ones from ten popular and mature GitHub projects and measure 16 features characterizing PRs, contributors, review processes, and projects. Using statistical and machine learning techniques, we observe that complex PRs, novice contributors, and lengthy reviews have a higher probability of abandonment and the rate of PR abandonment fluctuates alongside the projects' maturity or workload. To identify why contributors abandon their PRs, we also manually examine a random sample of 354 abandoned PRs. We find that the most frequent abandonment reasons are related to the obstacles faced by contributors, followed by the hurdles imposed by maintainers during the review process. Finally, we survey the top core maintainers of the study projects to understand their perspectives on dealing with PR abandonment and on our findings.

CCS Concepts: • **Software and its engineering → Collaboration in software development**; • **Human-centered computing → Empirical studies in collaborative and social computing**.

Additional Key Words and Phrases: socio-technical factors, pull-based development, modern code review, social coding platforms, open-source software, mixed-methods research

ACM Reference Format:
SayedHassan Khatoonabadi, Diego Elias Costa, Rabe Abdalkareem, and Emad Shihab. 2022. On Wasted Contributions: Understanding the Dynamics of Contributor-Abandoned Pull Requests: A Mixed-Methods Study of 10 Large Open-Source Projects on GitHub. *ACM Trans. Softw. Eng. Methodol.* 1, 1 (April 2022), 39 pages. https://doi.org/10.1145/nnnnnnn.nnnnnnn

Authors' addresses: SayedHassan Khatoonabadi, Data-driven Analysis of Software (DAS) Lab, Department of Computer Science & Software Engineering, Concordia University, Montreal, QC, Canada, s_khato@encs.concordia.ca; Diego Elias Costa, Data-driven Analysis of Software (DAS) Lab, Department of Computer Science & Software Engineering, Concordia University, Montreal, QC, Canada, d_damasc@encs.concordia.ca; Rabe Abdalkareem, School of Computer Science, Carleton University, Ottawa, ON, Canada, rabe.abdalkareem@carleton.ca; Emad Shihab, Data-driven Analysis of Software (DAS) Lab, Department of Computer Science & Software Engineering, Concordia University, Montreal, QC, Canada, eshihab@encs.concordia.ca.

1049-331X/2022/4-ART $15.00
https://doi.org/10.1145/nnnnnnn.nnnnnnn

1 INTRODUCTION

Pull-based development has been popularized by social coding platforms such as GitHub and is widely adopted by distributed software teams, especially within the open-source community [26]. In this model, developers fork a project (i.e., create a personal copy of the project) before making their changes. Whenever ready, the developers request their changes to get merged into the project by submitting a pull request (PR). The maintainers then review the PR and decide whether to merge it into their project. Compared to traditional methods, pull-based development reduces the time taken to review and merge the contributions [26, 83].

The streamlined contribution mechanism enabled by PRs has encouraged numerous external developers to contribute to open-source projects with fewer barriers [26, 59, 83]. However, an industrial report [11] estimates that 8% of PRs are wasted and never merged. Such PRs are either rejected by the maintainers or abandoned by their contributors. In contrast to rejected PRs, abandoned PRs are valid contributions that are not finalized because their contributors have left the review process unfinished. Unfortunately, abandoned PRs waste a considerable amount of time and effort that is often put in both by the contributors to prepare and submit such PRs and by the maintainers to manage and review them.

The literature has extensively studied how various technical, social, and personal factors influence the acceptance and review process of PRs. However, PR abandonment as a challenge that results in a great opportunity cost for the open-source community, especially the contributors and the reviewers of abandoned PRs, has only recently received attention from Li et al. [47]. Based on a survey with open-source developers, they explained how abandoned PRs impact project maintainers and discussed why developers abandon their PRs. While their findings shed light on PR abandonment from the perspective of developers, the influence of the factors related to PRs, contributors, review processes, and projects on PR abandonment is still not known.

To gain a better and more comprehensive understanding of the underlying dynamics of contributor-abandoned PRs, we conduct a mixed-methods study using both quantitative and qualitative methods [10]. *For the sake of brevity, we refer to external contributors as contributors throughout the paper.* First, we curate a dataset consisting of 266,039 contributor PRs from ten popular and mature GitHub projects (namely, Homebrew Cask, Kubernetes, Kibana, Ansible, DefinitelyTyped, Rust, Odoo, Legacy Homebrew, Elasticsearch, and Swift). Then, we devise heuristics to identify 4,450 candidate PRs with a high chance of being truly abandoned by their contributors. Next, we measure 16 features to characterize the PRs, their contributors, their review processes, and their projects for our quantitative analyses. We aim to answer the following four research questions in this paper:

RQ_1: What are the significant features of contributor-abandoned PRs in the studied projects?
We find that contributor-abandoned PRs are usually more complex, their contributors are usually less experienced, and their review process is usually lengthier than nonabandoned PRs. Furthermore, as the projects mature, contributor-abandoned PRs have become more frequent in three projects (DefinitelyTyped, Kubernetes, and Swift) and less frequent in five other projects (i.e., Ansible, Elasticsearch, Homebrew Cask, Kibana, and Odoo).

RQ_2: How do different features impact the probability of PR abandonment in the studied projects?
We find that the features of the review process, contributor, and project are more important in predicting PR abandonment than the features of PRs themselves. Specifically, PRs with more than three responses from the participants or the contributors, and those submitted by novice contributors are more likely to get abandoned. Also, the abandonment probability

changes as the projects evolve, with half of the projects showing a decrease of abandonment in their mature stages and the other half showing an increase of abandonment.

RQ_3: What are the probable reasons why contributors abandon their PRs in the studied projects?
We find that difficulty addressing the maintainers' comments, lack of review from the maintainers, difficulty resolving the CI failures, and difficulty resolving the merge issues are the most common reasons why contributors abandon their PRs.

Our Contributions. In summary, we make the following contributions in this paper:

- We identify the features of PRs, their contributors, their review processes, and their projects that significantly differ between abandoned and nonabandoned PRs.
- We rank the features based on their relative importance for predicting PR abandonment and describe how different values of these features vary the predicted probability of abandonment.
- We identify the probable reasons why contributors abandon their PRs and survey the core developers of study projects to understand their perspectives on dealing with PR abandonment and our findings.
- To promote the reproducibility of our study and facilitate future research, we also share our dataset at:
https://doi.org/10.5281/zenodo.4892277.

Paper Organization. The remainder of this paper is organized as follows. Section 2 presents our research methodology and Section 3 to 5 present our findings for each research question. Then, Section 6 reports the perspectives of maintainers on dealing with PR abandonment and our findings. Next, Section 7 further discusses our findings and Section 8 reviews the related work. Finally, Section 9 discusses the limitations of our study and Section 10 concludes the paper.

2 METHODOLOGY

In the following, we explain how we design our study (Section 2.1), select the study projects (Section 2.2), collect the required data (Section 2.3), identify abandoned PRs (Section 2.4), and extract features from PRs (Section 2.5).

2.1 Study Design

To gain a more comprehensive understanding of the underlying dynamics of contributor-abandoned PRs, we conduct a mixed-methods study using both quantitative and qualitative methods [10]. First, we perform statistical analysis to identify the significant features of abandoned PRs in RQ_1 (Section 3). Then, we use machine learning techniques to determine the relative importance of the features and describe how each feature impacts the predicted probability of abandonment in RQ_2 (Section 4). Finally, we manually examine a random sample of 4,450 abandoned PRs to identify the reasons why contributors abandon their PRs in RQ_3 (Section 5).

2.2 Study Projects

For our study, we need open-source projects that are popular among the community and have a rich history of adopting pull-based development. For this purpose, we rely on GitHub as a pioneer in supporting the pull request model and the largest open-source ecosystem [19], which have also been the subject of many software engineering studies [38]. To focus on the most popular projects, we use the number of stars as a proxy [4, 5] and retrieve the list of the top 1,000 most-starred projects. Among these projects, we focus on the top ten with the most number of PRs to ensure that each project has enough historical data for our study. As shown in Table 1, the study projects

Table 1. Overview of the projects selected to study contributor-abandoned PRs.

Project	PRs	Stars	Contributors	Months	Domain	Language(s)
Homebrew Cask	78,446	17,077	7,246	98	Package Manager	Ruby
Kubernetes	56,721	66,644	3,628	71	Container Orchestration	Go
Kibana	43,324	14,313	896	87	Analytics Dashboard	TypeScript
Ansible	42,338	43,333	7,168	98	Automation Platform	Python
DefinitelyTyped	38,645	28,316	13,866	91	Type Definitions	TypeScript
Rust	38,361	45,261	3,181	116	Programming Language	Rust
Odoo	38,241	17,636	1,822	72	Business Apps	JavaScript, Python
Legacy Homebrew	33,577	27,786	7,904	75	Package Manager	Ruby
Elasticsearch	33,411	49,134	2,350	116	Analytics Engine	Java
Swift	31,984	51,831	974	54	Programming Language	C++, Swift

cover multiple application domains and programming languages, with each project having at least 14 thousand stars, 31 thousand PRs, 800 external contributors, and 4 years of PR history.

2.3 Data Collection

To identify abandoned PRs, we require the timeline activity of PRs, which records all the events during the lifecycle of a PR. For this purpose, we use the PyGithub package [36] to retrieve the required data from GitHub. On May 30th, 2020, we collected the timeline, commits, and changed files metadata [22, 23] for the 435,048 PRs of the study projects.

2.4 Abandoned PRs Identification

After collecting the PRs data, we need to identify those abandoned by their contributors. Each PR in GitHub has one of the following three states: (i) *open* indicates that the PR is not finalized and might be under progress, (ii) *closed* indicates that the PR is either rejected by the maintainers or abandoned by its contributor, and (iii) *merged* indicates that the PR is merged into the project. Abandoned PRs are a subset of the *open* or *closed* PRs that are wasted because their contributors have left them unfinished. The contributors of such PRs may either explicitly declare their abandonment decision or implicitly stop addressing the maintainers' comments. The maintainers often employ bots like Stale [24] to close abandoned PRs after a period of inactivity [75], or they manually find and close the abandoned PRs.

However, GitHub does not assign a specific status for abandoned PRs to explicitly distinguish them from nonabandoned ones. Therefore, we cannot simply retrieve the list of abandoned PRs neither directly through the GitHub API [21] nor using existing archives such as GHTorrent [25] and GH Archive [29]. Therefore, we resort to heuristics to identify abandoned PRs based on the collected metadata. Heuristics are not guaranteed to be optimal and are subject to an inherent trade-off between their accuracy and completeness [39]. To determine the best balance between the precision and recall of our dataset, we experimented with different heuristics before finalizing the following:

Step 1: Exclude the PRs from core developers. This study focuses on contributions from external developers, which are more prone to get abandoned. Therefore, we exclude the PRs from core developers to focus on external contributions. GitHub defines eight roles for the authors of PRs within a project [20]. Among these roles, *owner* refers to the owners of the project, *member* refers to the members of the organization owning the project, and *collaborator* refers to those invited to collaborate on the project. Since these three roles typically have push/merge permissions within a GitHub repository, we consider them as core developers and exclude their PRs from our dataset.

Step 2: Exclude the PRs from deleted accounts. GitHub allows its users to remove their accounts permanently and afterward refers to the contributors of PRs from such accounts as *ghost*. Since there is no straightforward way to distinguish between the contributors of such PRs [38], we exclude them from our study.

Step 3: Exclude the recently updated PRs. To minimize the chance of marking PRs that are still under progress as abandoned, we exclude the PRs that their contributors have recently updated. To be conservative, we exclude the PRs that their contributors have updated (i.e., new comments or commits) within the last six months of the data collection date (i.e., May 30th, 2020).

Step 4: Exclude the merged PRs. We consider merged PRs as not wasted and thus not abandoned. However, the maintainers may not always use the merging methods provided through the GitHub interface to merge PRs. To account for such PRs, we resort to heuristics similar to Kalliamvakou et al. [38]. Specifically, we exclude the PRs with a *merged* status (i.e., merged using the GitHub interface) and those PRs that are closed without a *merged* status, but have a merged commit inside the project that references them (e.g., "Close #123").

Step 5: Search keywords in the discussion comments. As the last step for identifying abandoned PRs, we rely on keyword searching within all the discussion comments of PRs similar to Li et al. [47]. First, we remove code snippets and reply quotes from these comments and then search for keywords representing the unresponsiveness of contributors. To determine such keywords, we consider the keywords used in Li et al. [47] as our initial set. Then, we manually examine a sample of known abandoned PRs from our study projects and iteratively refine our keywords. Finally, we find the following keywords are commonly used to refer to abandoned PRs:
`{abandon, stale, any update, lack of update, no update, inactive, inactivity, lack of activity, no activity, not active, lack of reply, no reply, lack of response, no response}`.

Using our heuristics, we identified 4,450 abandoned PRs among the curated 266,039 PRs. As with any heuristic, ours may return some nonabandoned PRs (i.e., false positives), given that the review process of PRs often involves social interactions between the contributors and the reviewers before getting finalized. To validate the quality of our dataset, we manually analyze 100 PRs (10 PRs from each project) to verify if they have been truly marked as abandoned. We find seven false positives out of the 100 examined PRs as these PRs were rejected while including the keywords representing abandonment (e.g., [50]). Still, we believe that a false-positive rate of 7% gives us enough confidence to rely on this dataset for our study.

2.5 Feature Extraction

To identify the features that are possibly associated with PR abandonment, we consult the literature on pull-based development [27, 80–82]. As shown in Table 2, we extract 16 features covering four dimensions: (i) PR features, (ii) contributor features, (iii) review process features, and (iv) project features. In the following, we describe the extracted features for each dimension in more detail.

PR Features:

Description Length. The description length of PRs is found to negatively impact their acceptance probability and review time [79]. We aim to understand whether PRs with shorter descriptions are more frequently abandoned than verbosely described PRs. To characterize a PR's description length, we measure the number of words that have been used in its title and description (denoted by *pr_description*).

Table 2. Overview of the features extracted to characterize PRs, their contributors, their review process, and their projects.

Dimension	Feature	Description
Pull Request	pr_description	Number of words in the title and description of the PR
	pr_commits	Number of commits during the lifecycle of the PR
	pr_changed_lines	Number of changed lines during the lifecycle of the PR
	pr_changed_files	Number of changed files during the lifecycle of the PR
Contributor	contributor_contribution_period	Number of months since the first PR of the contributor in the project
	contributor_pulls	Number of prior PRs by the contributor in the project
	contributor_acceptance_rate	Ratio of previously merged PRs by the contributor in the project
	contributor_abandonment_rate	Ratio of previously abandoned PRs by the contributor in the project
Review Process	review_response_latency	Number of days till the first response in the PR
	review_participants	Number of participants during the lifecycle of the PR
	review_participants_responses	Number of responses by the participants during the lifecycle of the PR
	review_contributor_responses	Number of responses by the contributor during the lifecycle of the PR
Project	project_age	Number of months since the starting date of the project
	project_pulls	Number of prior PRs in the project
	project_contributors	Number of prior contributors in the project
	project_open_pulls	Number of open PRs in the project at the submission time of the PR

Change Complexity. The complexity of changes has been extensively shown to negatively impact the acceptance probability and the review time of PRs [41, 66, 73, 78, 79]. We aim to understand whether complex PRs are more prone to get abandoned. To characterize a PR's change complexity, we measure the number of commits that have been submitted during the PR's lifecycle (denoted by *pr_commits*); the number of lines (denoted by *pr_changed_lines*), and the number of files (denoted by *pr_changed_files*) that have been changed (i.e., additions or deletions) as part of the submitted commits.

Contributor Features:

Experience Level. The experience of contributors has been extensively shown to positively impact the acceptance probability and the review time of PRs [26, 41, 66, 79]. We aim to understand whether more experienced contributors are less likely to abandon their PRs. To characterize a contributor's experience within a project, we measure the number of months that have been elapsed since the first submitted PR of the contributor to the project (denoted by *pr_contribution_period*); the number of PRs that the contributor has previously submitted to the project (denoted by *contributor_pulls*); and the ratio of the previously submitted PRs by the contributor that had been merged into the project (denoted by *contributor_acceptance_rate*).

Abandonment History. To the best of our knowledge, the abandonment history of contributors has not been previously studied. We aim to understand whether contributors who have a long history of abandonment are more likely to abandon their PRs. To characterize a contributor's abandonment history within a project, we measure the ratio of the previously submitted PRs by the contributor that we have marked as abandoned in the project (denoted by *contributor_abandonment_rate*).

Review Process Features:

Response Latency. The response latency is found to negatively impact the acceptance probability and the review time of PRs [78, 79]. We aim to understand whether PRs that take longer to receive a first response from the reviewers are more likely to get abandoned. To characterize a PR's response

latency, we measure the number of days that have been taken to receive their first response (i.e., comment or review) from the participants (denoted by *pr_response_latency*).

Participants Activity. The activity of participants (i.e., anyone participating in the review process except the contributor) is found to negatively impact the acceptance probability of PRs [41, 73]. We aim to understand whether PRs with a higher activity from their participants are more likely to get abandoned. To characterize the participants' activity in a PR, we measure the number of participants in its review process (denoted by *review_participants*); and the number of responses (i.e., comments or reviews) that have been submitted by the participants (denoted by *review_participants_responses*) during the review process.

Contributor Activity. Similar to the participants' activity, we aim to understand whether PRs with a higher activity from their contributors are also more likely to get abandoned. To characterize the contributor's activity in a PR, we measure the number of responses (i.e., comments or self-reviews) that the contributor has submitted during the review process of the PR (denoted by *review_contributor_responses*).

Project Features:

Maturity Level. The maturity of projects is found to have a mixed impact on the acceptance probability and the review time of their PRs [73, 79]. We aim to understand whether the rate of abandoned PRs changes as projects become more mature. To characterize a project's maturity, we measure the number of months that have been elapsed since the creation date of the project until the submission date of the PR (denoted by *project_age*); the number of PRs that have been previously submitted to the project (denoted by *project_pulls*); and the number of developers who have previously contributed to the project (denoted by *project_contributors*) at the submission time of the PR.

Maintainers Workload. The workload of maintainers is found to negatively impact the acceptance probability and the review time of PRs [78, 79]. We aim to understand whether the high workload of maintainers increases the rate of abandoned PRs. To characterize a project's workload, we measure the number of submitted PRs that were still open at the submission time of the PR (denoted by *project_open_pulls*).

3 RQ_1: WHAT ARE THE SIGNIFICANT FEATURES OF CONTRIBUTOR-ABANDONED PRS IN THE STUDIED PROJECTS?

PR abandonment is a challenge that results in significant opportunity costs for the open-source community, especially the contributors and the reviewers of abandoned PRs. A recent study by Li et al. [47], has surveyed open-source developers to explain why PRs get abandoned. However, the influence of different factors on PR abandonment has not been studied yet. As our first research question, we aim to understand which features of PRs, their contributors, their review processes, and their projects are associated with PR abandonment. Specifically, we want to investigate how significantly abandoned PRs differ from nonabandoned ones.

3.1 Approach

We perform statistical analyses to identify the significant features of abandoned PRs in contrast to nonabandoned PRs. First, we compare the distribution of the extracted features between abandoned and nonabandoned PRs and then test their statistical and practical significance. In the following, we explain each step in more detail:

Step 1: Compare distribution of features. To compare the distribution of features between abandoned and nonabandoned PRs, we generate violin plots [33] using the ggstatsplot package [56] for each project. The generated plots for each feature are shown in Appendix A, specifying their median values (denoted by M), interquartile ranges (the box inside the violin), and probability densities (the width of the violin at each value).

Step 2: Test statistical significance of features. To test the statistical difference between the features of abandoned and nonabandoned PRs, we apply the Mann–Whitney U test [51] with a 95% confidence level (i.e., $\alpha = 0.05$). We use this nonparametric test because we cannot assume the distribution of our features to be normal. To calculate this statistic, we use the stats package [61] and add the results to the plots generated in Step 1. For easier comparison, we denote $p < 0.05$ with *, $p < 0.01$ with **, and $p < 0.001$ with ***.

Step 3: Test practical significance of features. While statistical significance verifies whether a difference exists between the features of abandoned and nonabandoned PRs, we also need to test their practical difference [40]. For this purpose, we use Cliff's delta [8] to estimate their magnitude of difference (i.e., effect size). The value of Cliff's delta (denoted by d) ranges from -1 to $+1$: a positive d implies that the values of the feature in abandoned PRs are often greater than those of nonabandoned PRs, while a negative d implies the opposite. To calculate this statistic, we use the effectsize package [2] and add the results to the plots generated in Step 1. For easier comparison, we convert the d values to qualitative magnitudes based on the following thresholds [32]:

$$\text{Effect size} = \begin{cases} \text{Negligible}, & \text{if } |d| \leq 0.147 \\ \text{Small}, & \text{if } 0.147 < |d| \leq 0.33 \\ \text{Medium}, & \text{if } 0.33 < |d| \leq 0.474 \\ \text{Large}, & \text{if } 0.474 < |d| \leq 1 \end{cases}$$

3.2 Findings

Table 3 summarizes the significance of different features across the study projects. We consider a feature significant if its difference between abandoned and nonabandoned PRs is both statistically significant (i.e., $p < 0.05$) and practically significant (i.e., the effect size is small, medium, or large) in at least one project. Overall, we observe that the most significant features are related to the review process and contributors of PRs. We also find that four features (characterizing the review process and contributor) are significant across all the projects, and eight remaining features (encompassing all the dimensions) are significant in at least half the projects. In the following, we discuss the significance of each dimension in more detail.

Abandoned PRs are usually more complex than nonabandoned PRs. As shown in the PR dimension of Table 3, abandoned PRs tend to have lengthier descriptions (8 projects), contain more commits (6 projects), and involve more changed lines (3 projects). However, abandoned and nonabandoned PRs tend to be similar in their number of changed files across all the projects. The results suggest that abandoned PRs received even more efforts from their contributors, highlighting the waste resulting from the abandonment.

The contributors of abandoned PRs usually have less experience than the contributors of nonabandoned PRs. As shown in the contributor dimension of Table 3, the contributors of abandoned PRs tend to have previously submitted fewer PRs (all the projects), have a lower acceptance rate (all the projects), have a lower contribution period (5 projects), and have a higher abandonment rate (2 projects). Note that the results cannot be attributed to the expected higher

Table 3. Significance of different features across the study projects. ↑ shows that abandoned PRs have values greater than nonabandoned ones, ↓ shows that abandoned PRs have values smaller than nonabandoned ones, and ↑↓ shows a mixed relationship.

Dimension	Feature	Significant	Small	Medium	Large
Pull Request	pr_description	8	7 (↑)	1 (↑)	–
	pr_commits	6	6 (↑)	–	–
	pr_changed_lines	3	1 (↑)	–	2 (↑)
	pr_changed_files	–	–	–	–
Contributor	contributor_pulls	**10**	4 (↓)	3 (↓)	3 (↓)
	contributor_acceptance_rate	**10**	5 (↓)	4 (↓)	1 (↓)
	contributor_contribution_period	5	2 (↓)	1 (↓)	2 (↓)
	contributor_abandonment_rate	2	2 (↑)	–	–
Review Process	review_participants_responses	**10**	1 (↑)	1 (↑)	8 (↑)
	review_participants	**10**	2 (↑)	1 (↑)	7 (↑)
	review_contributor_responses	7	3 (↑)	3 (↑)	1 (↑)
	review_response_latency	4	4 (↑)	–	–
Project	project_age	8	6 (↑↓)	–	2 (↑↓)
	project_pulls	8	6 (↑↓)	–	2 (↑↓)
	project_contributors	8	6 (↑↓)	–	2 (↑↓)
	project_open_pulls	6	4 (↑↓)	–	2 (↑↓)

familiarity and expertise of the maintainers because we only consider external contributors in our study (Section 2.4).

The review process of abandoned PRs is usually lengthier than the review process of nonabandoned PRs. As shown in the review process dimension of Table 3, the review process of abandoned PRs tends to receive more responses from its participants (all the projects), involve more participants (all the projects), receive more responses from the contributors (7 projects), and have a higher latency to receive the first response from the participants (4 projects). The results suggest that abandoned PRs are not just abandoned after the PR was submitted but have received even more effort from both their contributors and reviewer, again highlighting the waste resulting from the abandonment.

The project features play both a positive and negative role in PR abandonment. As shown in the project dimension of Table 3, we observe contrasting patterns in how the rate of abandoned PRs change alongside the project maturity or workload (8 projects). For easier comparison, we group these projects based on their similarities (i.e., positive or negative) in Table 4. In the first group (DefinitelyTyped, Kubernetes, and Swift), abandoned PRs tend to become more frequent as the projects become more mature (i.e., increase in project_age, project_pulls, and project_contributors). In two of these projects (Kubernetes and Swift), abandoned PRs also become more frequent as they experience a higher workload (i.e., increase in project_open_pulls). In contrast to the first group, the second group (Ansible, Elasticsearch, Homebrew Cask, Kibana, and Odoo) experienced fewer abandoned PRs as the projects become more mature (i.e., increase in project_age, project_pulls, and project_contributors). Surprisingly, in four of these projects (Ansible, Elasticsearch, Kibana, and Odoo), abandoned PRs are more frequent when the projects have a lower workload (i.e., decrease in

Table 4. Difference of the project features between abandoned and nonabandoned PRs.

Group	Project	Maturity			Workload
		project_age	*project_pulls*	*project_contributors*	*project_open_pulls*
I	Kubernetes	Large (↑)	Large (↑)	Large (↑)	Large (↑)
	Swift	Small (↑)	Small (↑)	Small (↑)	Small (↑)
	DefinitelyTyped	Small (↑)	Small (↑)	Small (↑)	–
II	Kibana	Large (↓)	Large (↓)	Large (↓)	Large (↓)
	Ansible	Small (↓)	Small (↓)	Small (↓)	Small (↓)
	Elasticsearch	Small (↓)	Small (↓)	Small (↓)	Small (↓)
	Odoo	Small (↓)	Small (↓)	Small (↓)	Small (↓)
	Homebrew Cask	Small (↓)	Small (↓)	Small (↓)	–

project_open_pulls). The results may be associated with the change in the team structure, policies, or processes. For example, DefinitelyTyped has refined its review process since 2016 by rotatively assigning a TypeScript employee each week to focus on merging PRs [72].

Summary of RQ_1. Our findings suggest that contributor-abandoned PRs are usually more complex, their contributors are usually less experienced, and their review process is usually lengthier than nonabandoned PRs. Furthermore, as the projects mature, contributor-abandoned PRs have become more frequent in three projects (DefinitelyTyped, Kubernetes, and Swift) and less frequent in five other projects (i.e., Ansible, Elasticsearch, Homebrew Cask, Kibana, and Odoo).

4 RQ_2: HOW DO DIFFERENT FEATURES IMPACT THE PROBABILITY OF PR ABANDONMENT IN THE STUDIED PROJECTS?

In RQ_1, we investigated what features of PRs, their contributors, their review processes, and their projects are associated with PR abandonment. As our second research question, we aim to better understand which PRs have a higher probability of getting abandoned by their contributors. Specifically, we want to identify which features are the most important for predicting PR abandonment and describe how each feature can influence the abandonment probability of PRs.

4.1 Approach

We use machine learning techniques to understand how each feature varies the predicted probability of PRs getting abandoned. First, we consider the features that we found to be significant in abandoned PRs and remove correlated and redundant features to ensure the quality of our models. Then, we build and evaluate the classifier models that we later use to analyze the relative importance and impact of each feature on the abandonment probability. In the following, we explain each step in more detail:

Step 1: Remove insignificant features. In RQ_1, we found that the number of changed files in a PR (i.e., *pr_changed_files*) does not significantly differ between abandoned and nonabandoned PRs in any of the study projects. Therefore, we exclude this feature because it is not valuable for analyzing the abandonment probability of PRs and consider the remaining 15 features for our analysis.

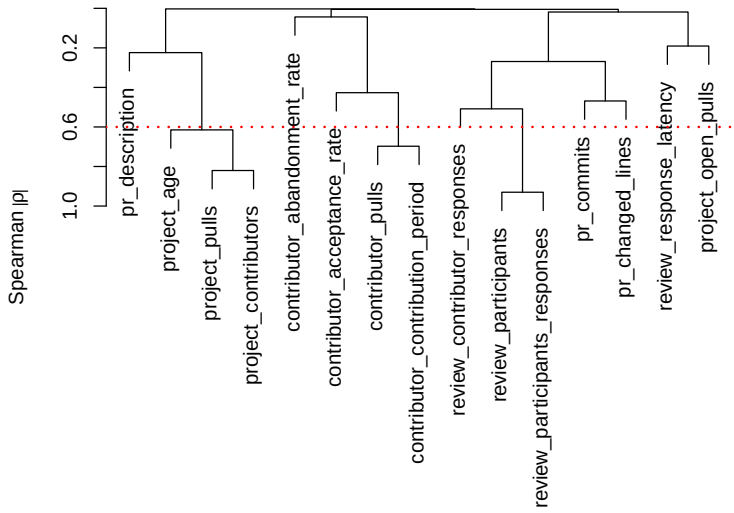

Fig. 1. Spearman's ρ correlation among different pairs of features across the combined data of the study projects.

Step 2: Remove correlated features. To focus on the most important features, we eliminate correlated features, which negatively affect the interpretation of models [13]. To check the monotonic relationship between each pair of features, we use Spearman's ρ [67] as a nonparametric test because we cannot assume the distribution of our features to be normal. We measure correlations on the combined data of the study projects to ensure that any correlation exists across all of them. Fig. 1 presents a hierarchical cluster of the correlations generated using the Hmisc package [31]. For each group of strongly correlated features (i.e., $|\rho| \geq 0.6$ as suggested by [14]), we keep the feature that is easier to interpret for our study and remove the rest. Accordingly, we drop the following four features from our analysis: *project_pulls*, *project_contributors*, *contributor_contribution_period*, and *review_participants*.

Step 3: Remove redundant features. While we remove correlated features in Step 1, we also need to eliminate redundant features to focus on the most important ones. To identify redundant features, we use the Hmisc package [31], which applies flexible parametric additive models to measure how well each feature can be predicted from other features. Similar to our correlation analysis, we measure redundancy on the combined data of the study projects to ensure that any redundancy exists across all of them. Accordingly, we did not find any redundant features.

Step 4: Build classifier models. To gain deeper insights on PR abandonment, we build a random forest classifier for each project to model the abandonment probability of PRs using the ranger package [76]. To model the abandonment probability, we consider the type of PR (i.e., abandoned or nonabandoned) as the dependent variable and the selected 11 features as the independent variables. Random forests [7] are commonly used in various domains and outperform linear models in both the predictive power and the ability to learn complex relations. To boost the predictive power of each model, we use the tuneRanger package [60]. This package automatically tunes the following three hyperparameters of random forests using sequential model-based optimization [37]: (i) the number of variables randomly drawn for each split, (ii) the fraction of instances randomly drawn for training each tree, and (iii) the minimum number of samples that a node must have to split.

Table 5. Performance scores of our model for each study project.

Project	AUC-ROC	AUC-PR	Baseline	AUC-PR / Baseline
Ansible	0.88	0.042	0.006	7.08x
DefinitelyTyped	0.86	0.262	0.054	4.84x
Elasticsearch	0.86	0.021	0.003	6.14x
Homebrew Cask	0.96	0.063	0.001	42.78x
Kibana	0.96	0.095	0.002	50.27x
Kubernetes	0.89	0.375	0.060	6.23x
Legacy Homebrew	0.92	0.116	0.013	8.85x
Odoo	0.77	0.029	0.002	17.25x
Rust	0.81	0.109	0.020	5.50x
Swift	0.82	0.059	0.003	19.36x

Step 5: Evaluate performance of models. To ensure that the models are reliable for our analysis, we evaluate their predictive power using the following two recommended metrics for binary classifiers [30]:

- **AUC-ROC:** which measures the area under the Receiver Operating Characteristic (ROC) curve [6]. The ROC curve plots the true positive rate (i.e., the ratio of correctly classified abandoned PRs to truly abandoned PRs) against the false positive rate (i.e., the ratio of incorrectly classified abandoned PRs to nonabandoned PRs) across different thresholds. The value of AUC-ROC ranges from 0 to 1, with values more than 0.5 indicating better performance than a no-skill classifier (i.e., baseline). Note that the value of AUC-ROC is the same for both positive (i.e., abandoned PRs) and negative (i.e., nonabandoned PRs) classes.
- **AUC-PR:** which measures the area under the precision-recall (PR) curve [16]. The PR curve plots the precision (i.e., the ratio of correctly classified abandoned PRs to all classified abandoned PRs) against recall (i.e., the ratio of correctly classified abandoned PRs to truly abandoned PRs) across different thresholds. The value of AUC-PR also ranges from 0 to 1, but the performance of a no-skill classifier (i.e., baseline) is determined by the distribution of classes in a dataset (i.e., distribution of abandoned and nonabandoned PRs). Note that, unlike AUC-ROC, the value of AUC-PR is different between positive (i.e., abandoned PRs) and negative (i.e., nonabandoned PRs) classes.

To reduce bias in our performance evaluations, we perform a stratified 10-fold cross-validation with ten repeats (a total of 100 iterations) for each model using the mlr package [3]. Table 5 presents the results of our performance evaluation for each model, where the baseline column shows the ratio of the minority class (i.e., abandoned PRs). We observe that our models have a good performance with an average AUC-ROC of 0.87 and perform at least four times better than the baseline in terms of AUC-PR.

Step 6: Analyze the importance of features. So far, we have build classifiers that can aptly model the abandonment probability of PRs. To compare the relative importance of different features, we perform permutation feature importance [17] for each model using the iml package [53]. This approach permutes a feature to break the association between the feature and the outcome (i.e., the abandonment probability in our case). The importance of the feature is then measured by how much error the permutated data introduces compared to the original error (i.e., loss in AUC in our

case) after 100 iterations. Therefore, the most important features have the largest impact on the performance of our models and thus are more valuable for predicting PR abandonment.

Step 7: Analyze the impact of features. After measuring the relative importance of the features in Step 6, we aim to describe how each feature varies the abandonment probability. For this purpose, we generate Accumulated Local Effects (ALE) plots [1] using the iml package [53]. ALE shows the effect of a feature at a certain value compared to the average prediction of the data. In other words, a downward trend implies a reduced probability of abandonment, an upward trend implies an increased probability of abandonment, and a stable ALE implies no changed probability of abandonment. To focus on the most common values of features, we filter out the values over the 99th percentile for each feature and each project. The plots are then created by dividing a feature into ten intervals selected based on its quantiles. For each interval, the PRs that fall into that interval are considered for calculating the difference in their prediction when replacing the value of the feature with the upper and lower limits of the interval. We model the abandonment probability as a function of the selected features, and thus, any relationship is causal for the model and may not hold in the real world [52].

4.2 Findings

Table 6 summarizes the importance of different features for each project model. We find that the features of the review process, contributors, and projects play a more prominent role in PR abandonment than the features of PRs themselves. Specifically, the number of responses from the participants is the most important feature by a large margin, indicating that the reviewers' activity is essential in classifying abandoned PRs. The second and third most important features are the acceptance rate and the number of previously submitted PRs of the contributor, respectively, highlighting the impact of the contributor experience on PR abandonment. The fourth and fifth most important features are the number of responses from the contributor and the age of the project. Other features, except for the abandonment rate of the contributor, are also among the top five in at least one project, showing that different features have a different impact on PR abandonment due to the inherent differences of the projects. Unexpectedly, we also observe that the number of commits in the PR, the abandonment rate of the contributor, and the latency to the first response from the participants are overall the least important features, respectively. In the following, we describe how the top five features impact the predicted abandonment probability of PRs. The ALE plots for the rest of the features can be found in Appendix B.

PRs with long discussions are more likely to get abandoned. Fig. 2 and 3 show how the number of responses from the participants and from the contributor varies the abandonment probability of a PR across the study projects, respectively. We find that the probability of abandonment increases in most of the projects as the number of responses from the participants or the contributor increase (i.e., upwards trend). We also observe that PRs that receive more than three responses from either the participants or the contributor have an increased probability of abandonment in most of the projects. The results provide further evidence that abandoned PRs often demand more time and effort from both their reviewers and their contributors (see Section 3.2).

Novice contributors are more likely to abandon their PRs. Fig. 4 and 5 show how the acceptance rate and the number of previously submitted PRs by the contributor vary the abandonment probability of a PR across the study projects, respectively. We observe that contributors with zero experience have the highest probability of abandonment in almost all the projects. Surprisingly, highly experienced contributors also have an increased probability of abandonment in a few projects.

Table 6. Importance of different features across the study projects.

Dimension	Feature	Ansible	DefenitelyTyped	Elasticsearch	Homebrew Cask	Kibana	Kubernetes	Legacy Homebrew	Odoo	Rust	Swift	Overall Rank	Average Loss
Pull Request	pr_changed_lines	7	8	7	**3**	6	9	**2**	7	6	9	6	1.55
	pr_description	8	**3**	**4**	**5**	8	6	9	9	7	10	8	1.48
	pr_commits	10	**5**	10	11	10	10	10	10	10	8	11	1.21
Contributor	*contributor_acceptance_rate*	**5**	**4**	**1**	6	**5**	**5**	8	**3**	**1**	**3**	**2**	1.95
	contributor_pulls	**3**	9	**3**	**4**	**3**	7	6	**4**	**4**	**1**	**3**	1.81
	contributor_abandonment_rate	11	11	11	8	11	8	11	11	9	6	10	1.24
Review Process	*review_participants_responses*	**1**	**1**	**2**	**1**	**1**	**1**	**1**	**1**	**5**	**2**	**1**	4.45
	review_contributor_responses	9	6	6	**2**	**2**	**4**	**3**	**5**	**2**	11	**4**	1.74
	review_response_latency	**2**	10	9	9	9	11	**5**	**2**	11	**5**	9	1.33
Project	*project_age*	6	**2**	**5**	7	**4**	**2**	**4**	8	**3**	7	**5**	1.73
	project_open_pulls	**4**	7	8	10	7	**3**	7	6	8	**4**	7	1.49

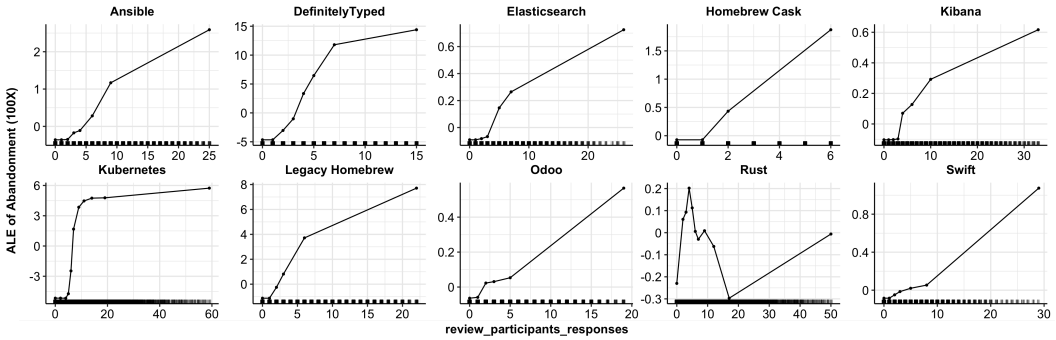

Fig. 2. ALE plots showing how *review_participants_responses* varies the abandonment probability of PRs.

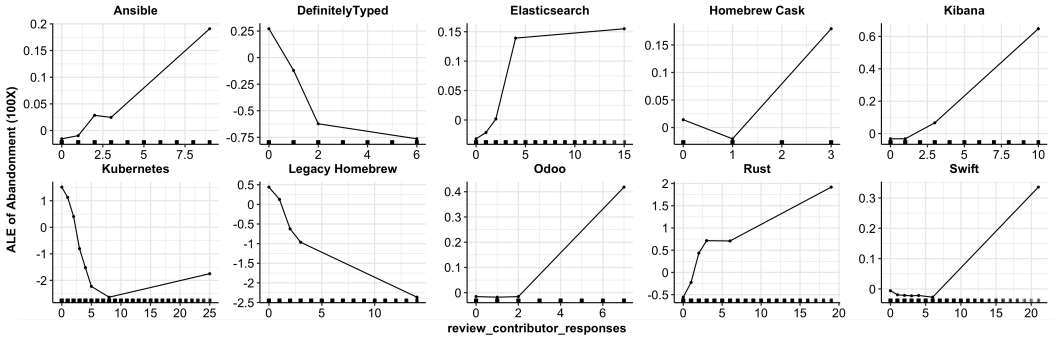

Fig. 3. ALE plots showing how *review_contributor_responses* varies the abandonment probability of PRs.

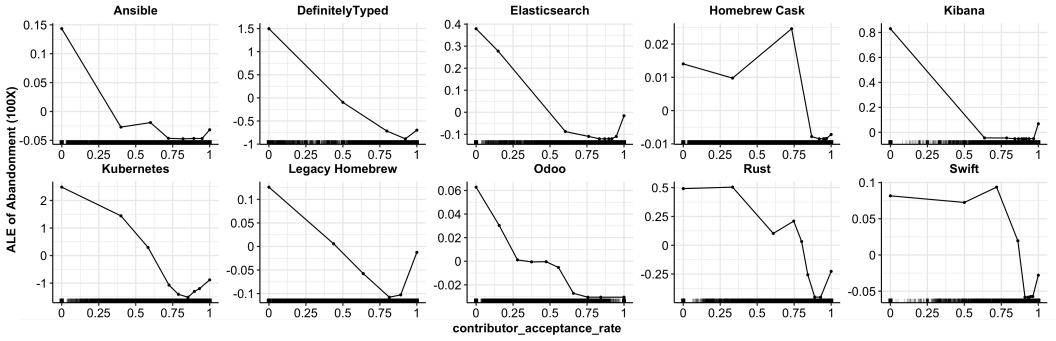

Fig. 4. ALE plots showing how *contributor_acceptance_rate* varies the abandonment probability of PRs.

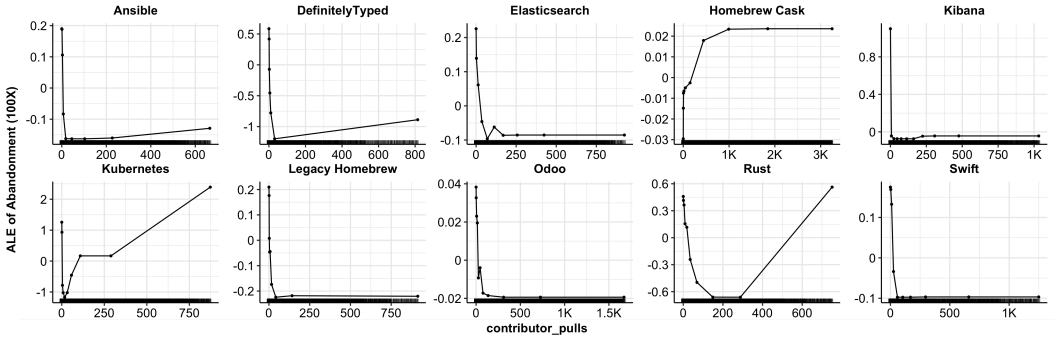

Fig. 5. ALE plots showing how *contributor_pulls* varies the abandonment probability of PRs.

The results suggest that the contributors of abandoned PRs may need more guidance and attention from the maintainers.

PR abandonment has changed throughout the history of projects. Fig. 6 shows how the age of projects varies the abandonment probability of a PR across the study projects. Similar to RQ_1 (Section 3.2), we observe two contrasting patterns: PR abandonment improves throughout the time in some projects and worsens in some other projects. In the first group (i.e., Ansible, DefinitelyTyped, Elasticsearch, Kibana, and Odoo), the abandonment probability is highest when the project age is low (i.e., downwards trend). However, in the second group (i.e., Homebrew Cask, Kubernetes, Legacy Homebrew, and Swift), the abandonment probability increases when the project age increases (i.e., upwards trend). While this fluctuation may be associated with the expected change in the workload of projects as they grow, we found that the number of open PRs is less important to our models than the age of the project.

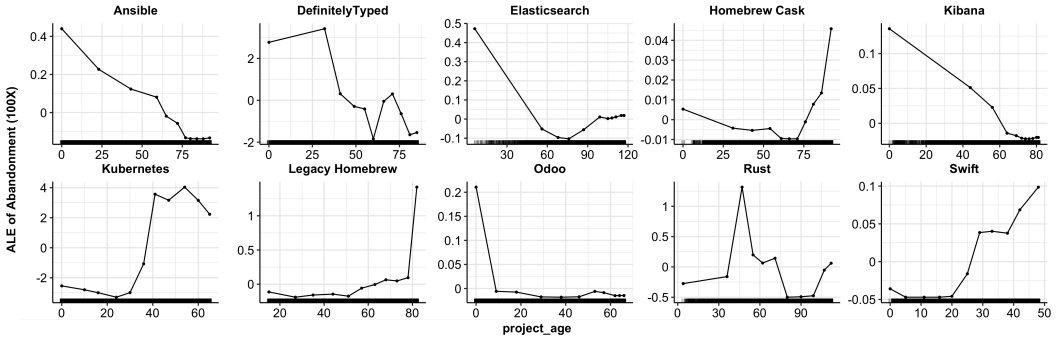

Fig. 6. ALE plots showing how *project_age* varies the abandonment probability of PRs.

Summary of RQ_2. Our findings suggest that the features of the review process, contributor, and project are more important in predicting PR abandonment than the features of PRs themselves. Specifically, PRs with more than three responses from the participants or the contributors, and those submitted by novice contributors are more likely to get abandoned. Also, the abandonment probability changes as the projects evolve, with half of the projects showing a decrease of abandonment in their mature stages and the other half showing an increase of abandonment.

5 RQ_3: WHAT ARE THE PROBABLE REASONS WHY CONTRIBUTORS ABANDON THEIR PRS IN THE STUDIED PROJECTS?

In RQ_1 and RQ_2, we quantitatively analyzed contributor-abandoned PRs to understand how different factors influence their abandonment probability. As our last research question, we aim to complement our previous findings and gain a deeper understanding of the underlying dynamics of abandoned PRs. Specifically, we want to look for clues in the review discussions of abandoned PRs to better understand why contributors abandon their PRs.

5.1 Approach

We perform a manual examination to establish a taxonomy of the abandonment reasons by following the coding guidelines presented in [65]. First, we label a random sample of abandoned PRs to identify the probable reasons why contributors abandon their PRs and then calculate our interrater agreement. In the following, we explain each step in more detail:

Step 1: Identify abandonment reasons. First, we randomly select 354 PRs from 4,450 abandoned PRs of the study projects (confidence level of 95% with a ±5% confidence interval). Then, the first three authors are required to manually examine the discussion comments of each PR and try to pinpoint the primary reason(s) why its contributor has abandoned it. In most cases, the contributor has abandoned the PR without any explanation, and thus we look for clues in the interactions between the contributor and the reviewers to identify the most probable reasons for the abandonment. In cases where the contributor provided a reason for their abandonment decision, we also investigate other major reasons that have led to their abandonment.

We perform the labeling in two rounds. In the first round, three annotators independently label a random sample of 60 PRs from the selected 354 PRs to establish the classification scheme. In the second round, we divide the sample 354 abandoned PRs (including the 60 PRs from the first

Table 7. Probable reasons why contributors abandon their PRs.

Category	Reason	Frequency (%)
Contributor-related	Difficulty addressing the maintainers' comments	45.8
	Difficulty resolving the CI failures	20.9
	Difficulty resolving the merge issues	14.1
	Difficulty complying with the project requirements	1.4
Maintainer-related	Lack of review from the maintainers	22.6
	Lack of answer from the maintainers	9.3
	Lack of integration by the maintainers	6.5
	Lack of consensus among the maintainers	4.5
PR-related	Existence of duplicated work	3.1
	Dependency on upcoming changes	1.4

round) into two sets to be labeled using the classification scheme from the previous round: the first set was labeled independently by the first and second authors, and the second set was labeled independently by the first and third authors. Finally, the annotators merged the labels and further refined the labels. In each round, when the annotators had different opinions, they discussed to reach an agreement and then retroactively updated all the previously labeled PRs to ensure a coherent classification.

Step 2: Calculate interrater agreement. To ensure the quality of our taxonomy, we calculate the Cohen's Kappa coefficient [9] using the scikit-learn package [57]. This statistic is commonly used to evaluate the interrater agreement in different domains. The value of Cohen's Kappa ranges from −1.0 to +1.0, with values more than 0 indicating an agreement better than chance. We obtained a Kappa score of 0.73, which is considered a substantial agreement [43].

5.2 Findings

As shown in Table 7, we identified ten major reasons why contributors abandon their PRs, grouped into three categories: (i) contributor-related reasons, (ii) maintainer-related reasons, and (iii) PR-related reasons. Note that the total frequency of the identified reasons is greater than 100% because we observe multiple reasons for some abandoned PRs. We find that the most frequent abandonment reasons are related to the obstacles faced by contributors followed by the hurdles imposed by maintainers during the review process. Particularly, difficulty addressing the maintainers' comments, lack of review from the maintainers, and difficulty resolving the CI failures are the most frequent reasons (observed in more than 20% of abandoned PRs). In the following, we discuss the identified reasons in the order of their frequency.

Difficulty addressing the maintainers' comments (45.8%). In almost half of the abandoned PRs, their contributors found it difficult to address the maintainers' comments, questions, or change requests. The contributors did not have the required technical knowledge or enough time to continue the work (e.g., [P198] said *"Nope. Had no time and will to take on this."*). Interestingly, contributors may ask others to continue the work (e.g., [P340] said *"If you don't mind taking it over, that would be fantastic. Happy to provide whatever help I can!"*).

Lack of review from the maintainers (22.6%). The second common reason why contributors abandon their PRs is that they did not receive a timely (if any) review from the maintainers.

Sometimes, a PR gets reviewed, and the contributor addresses the maintainers' comments, but the maintainers do not follow it up. For example, [P7] was closed after multiple rounds of discussions, and even though the contributor addressed the maintainers' comments (*"Oh dang, I guess it doesn't warn you after the first time. I'll get this re-made soon and try to be a little more proactive about getting input."*). Not receiving timely reviews from the maintainers may also send a signal to the contributors that their work is not treated seriously (e.g., [P182] said *"I'm happy to resolve these merge conflicts now, but before I do, I am curious if I have messed something else up in my contribution? Since this didn't get a comment since July of last year, I am afraid I've missed something crucial."*). If PRs are not reviewed in a timely manner, they may become outdated and require extra work from their contributors to rebase and update them.

Difficulty resolving the CI failures (20.9%). The third reason includes cases where the contributors found it difficult to resolve the continuous integration (CI) failures arisen during the review process. Such failures are often brought up by the project bots even before the PR gets a review from the maintainers. Sometimes, the contributors do not even know how to fix the CI failures and ask the maintainers for help (e.g., [P346] said *"I've looked at the errors in Travis. Most of them seem unrelated to the PR. Could you please help me out with this?"*).

Difficulty resolving the merge issues (14.1%). The fourth reason includes cases where the contributors found it difficult to resolve the merge issues arisen during the review process. If the project codebase has been updated, the contributors are asked to rebase their local branch, resolve any merge conflicts, and then push their changes again. Such issues typically arise when the maintainers take a long time to review the PR, and the PR becomes outdated (e.g., [P181] said *"I am willing to keep rebasing (and certainly willing to continue responding to comments), but not without some indication that it will eventually be merged."*). We also observe that contributors are sometimes asked to squash their pushed commits into a single commit to reduce the noise in the revision history, which requires more effort and time from them.

Lack of answer from the maintainers (9.3%). In some cases, we observe the reason for abandonment is because the contributors had not received a timely (if any) answer from the maintainers when they asked for help to complete a task (e.g., [P300] said *"CI is timing out. Everything is fine until the timeout. Anything I can do to get this merged?"*) or asked for clarification (e.g., [P313] said *"Modifying the passed in proxyTransport cannot be considered 'safe'?"*) or asked for confirmation (e.g., [P273] said *"Is that the right way?"*). Note that this reason is different from "lack of review from the maintainers" as such PRs are blocked because the contributor is awaiting an answer for a question from the maintainer and not awaiting a review for the applied changes.

Lack of integration by the maintainers (6.5%). This reason includes cases where a PR has been already approved by the reviewers but has been pending integration. In some projects, such as Kubernetes, a PR should undergo a two-phase review process. In the first round, reviewers approve the changes, and then project integrators need to merge the changes. However, contributors may abandon their PRs if the integrators do not attempt to merge the PR in a timely (if any) manner after the reviewers have approved the PR. For example, in [P1], the PR has been approved for integration multiple times. However, since the integrators were not responsive, the PR needs to be rebased, requiring additional work from the contributor. In [P307], a reviewer suggested *"Might be worth pinging the reviewers too if that is all that is stopping progress."*

Lack of consensus among the maintainers (4.5%). This reason includes cases where the reviewers could not reach a consensus on how to continue with the PR. This disagreement typically arises when there is no straightforward solution to resolve the PR issues, and each alternative has

its own advantages and disadvantages. Such PRs often undergo a long discussion and demand lots of time and effort from both reviewers and contributors (e.g., [P354]). However, the PR eventually gets abandoned by the contributor due to inconsistent feedback and often overlong discussions.

Existence of duplicated work (3.1%). This reason includes cases where the work is either a duplicate of an existing PR (e.g., [P294] said *"I am abandoning this PR in favor of npm-ramda, but I wont close it in case someone want to make use of it."*), another contributor has submitted a more comprehensive PR, or the issue addressed in the PR is no longer applicable (e.g., [P340] said *"pkg-config should no longer be able to pick up non-deps under superenv. Hopefully that means this is resolved."*).

Difficulty complying with the project requirements (1.4%). This reason includes rare cases where the contributors found it difficult to comply with the project-specific requirements. In projects such as Kubernetes, the contributors are asked to sign the contribution level agreements before their PR even gets reviewed by the maintainers (e.g., [P149]). Also, many projects provide templates for the PR description and ask the contributors to update the description according to the templates (e.g., [P134]).

Dependency on upcoming changes (1.4%). In rare cases, the abandoned PRs were not valuable on their own and depended on other changes that must be merged first before the proposed changes can be considered (e.g., [P11] said *"Hopefully, once mappable and partial types land in TS, I will fix this."*).

Summary of RQ_3. Our findings suggest that the difficulty addressing the maintainers' comments, lack of review from the maintainers, difficulty resolving the CI failures, and difficulty resolving the merge issues are the most common reasons why contributors abandon their PRs.

6 PERSPECTIVES OF THE PROJECT MAINTAINERS

To gain deeper insights on PR abandonment, we design a survey asking the core maintainers of the studied projects about their perspectives on our findings and how to tackle PR abandonment. After explaining the goal of this research and presenting a summary of our findings, we ask the following three open-ended questions in the survey:

- Does your team implement any approaches to deal with abandoned PRs? If so, what approaches does your team use?
- Do you suggest any approaches that can minimize the risk of a PR being abandoned by its contributor? (these can be approaches that your team does or does not use)
- Do you have any feedback about the four findings of our study? (we would love to hear it, positive or negative)

We send e-mails to the top 25 core developers of each study project (a total of 250 e-mails) to invite them to participate in our survey. From these invitations, we receive a total of 16 responses (6.5%) to our survey. Our response rate is similar to the 5% response rate, commonly observed in software engineering studies [45]. From the 10 study projects, we received responses from all the projects except Elasticsearch. As our sample is small, we refrain from discussing particular projects and instead present the overarching themes that appeared in the participant responses.

6.1 How do projects *deal* with abandoned PRs?

In this question, we aim to understand the processes and practices that the projects have already put in place to deal with abandoned PRs. Out of the 16 participants, six responded that their team had implemented approaches to deal with abandoned PRs. These approaches are:

Holding triage meetings (4x). The most commonly mentioned approach to deal with abandoned PRs is holding (recurrent) triage meetings, where developers review the status of PRs and support PRs that need attention. According to the survey participants, triage meetings streamline the communication and help finding problems before it causes the PRs to become abandoned.

> *"We have regular triage meetings to review the status of PRs that abandoned or about to be abandon. We also help new comers and new maintainers for reviving the old PRs and merge them."* [S1]
>
> *"Better upfront planning and communication help eliminate abandoned PRs. Many times, a quick zoom with a teammate to talk through the proposed change goes a long way into finding problems before a large effort is needed."* [S6]

Using bots to auto-close abandoned PRs (2x). While all the projects use Stale bot [24] or a similar in-house implementation (e.g., fejta-bot in Kubernetes [42]) to follow up with PRs that are about to get abandoned and auto-close already abandoned PRs, two participants explicitly mentioned their use of such bots. As a respondent explained:

> *"A bot first pings owners of the code after a week. Then it pings the submitter a couple of weeks later, telling them that it will be closed soon."* [S2]

Most of the participants (10 responses) reported that their project does not implement any particular approach to deal with abandoned PRs. From these participants, four explained why their team had not adopted any specific approach to prevent PR abandonment:

The maintainers are overwhelmed with work (2x). Two maintainers reported that they are overwhelmed with work and thus have decided to reduce their efforts on retaining PRs from external contributors. As a respondent explained:

> *"We are overwhelmed with work, and most community PRs have a low ration value/effort needed to merge it, so we essentially gave up, except for rare cases."* [S13]

The project does not rely on open source contributions (2x). Other two participants mentioned that their project does not rely on external contributions for the implementation of new features or improvements. Therefore, there is little incentive to spend special time and effort in retaining PRs from external contributors. As a respondent explained:

> *"We appreciate community PRs, but the vast majority of the contributors to our project are employees and so we don't rely on open source contributions for features or improvements. Community PRs are usually applicable to a specific niche usecase which we'd be happy to accept if the contributor is willing to go through the process with us."* [S5]

6.2 How do maintainers *recommend* to mitigate PR abandonment?

In this question, we aim to understand the approaches that the maintainers recommend to minimize the risk of PRs getting abandoned by their contributors, whether their team has adopted them yet or not. Out of the 16 participants, 15 responded with suggestions that they believe could mitigate

PR abandonment. In the following, we summarize these recommendations into suggestions for contributors, suggestions for maintainers, and suggestions for both maintainers and contributors.

Maintainers should strive to make the contributor experience as smooth as possible (6x) It stands to reason that the more obstacles contributors face, the higher the chances of abandonment. One participant (S5) mentioned that having helpful and understandable error messages can help contributors fix issues in their PRs. Another participant (S3) mentioned that maintainers should "put kid gloves on" when dealing with newcomers and make community contributions as painless as possible. One participant (S15) even suggested that maintainers merge PRs with minor issues and then either make the required changes themselves or open an issue in the project. As one respondent aptly summarizes:

> *"Responding quickly and encouragingly, and making your PR contributor experience as seamless as possible (automatic CI, helpful and understandable error messages) are likely all you can do. It's a lot of work even to submit a PR, the extra work necessary when updates are needed isn't something most people will be willing to give."* [S5]

The participants also mentioned that improving the project's testing documentation (S1) and contribution guidelines (S2), and making bot instructions more understandable (S2) could help to mitigate PR abandonment. One participant (S13) also cited that projects need to increase their available resources, with another participant (S1) mentioning that increasing community reach and having maintainers from the community may help maintainers better handle the required workload.

Maintainers should establish a triage process for external contributions (2x). Once again, the participants mentioned the importance of a triage process in mitigating PR abandonment. A triage process helps assign reviewers to a PR based on their expertise and experience and may lead to timely responses to contributors. One participant (S14) suggested that PRs should preferably be assigned to one reviewer instead of an entire team to compel reviewers to act and prevent idleness. Another respondent described that the triage process should also monitor the status of PRs and act if reviewers have not responded to the contributors yet:

> *"[Projects should have] a human rotation that triages pull requests and checks if they are progressing, or if someone has dropped the ball or is waiting on some event that will never happen."* [S16]

Contributors should create PRs that are clear and concise (3x). The participants emphasized the importance of PRs to focus only on a single use case and provide a clear description of changes. PRs that include multiple unrelated changes create a burden for reviewers that need to ensure all changes are correct. As two participants responded:

> *"Make the proposal clear and concise. The reviewer might take 10 or 15 seconds to figure out the problem being solved and the solution. If it takes longer, the reviewer will probably give up."* [S11]

> *"Smaller PRs are obviously better. We try not to nit-pick though it is human nature that a 100 line diff gets no nitpicking and a 4 line diff gets plenty. This can hardly make people want to contribute and is unfortunate."* [S9]

Contributors should assess the project's interest in the proposed changes before submitting PRs (3x). Managing expectations is important in contributing to open source projects. As different projects have different philosophies and needs regarding community contributions,

contributors should assess whether maintainers value their PRs. Typically, new features are harder to integrate than bug fixes and performance improvement patches.

> *"In large projects, a performance improvement or a bug fix proposed by an external contributor is more likely to be merged than a new feature. Not only because it's usually simpler, but because the feature might not be in line with the project's interests. A bug fix or a performance improvement is always in line with the project's interest."* [S10]
>
> *"Managing expectations could be a factor. We won't merge entirely new features just like that, simply because it is really important that Odoo remains and improves on being kept simple. E.g. PRs could come from a client project that needs a certain feature, but merging it like this, might do harm to a lot of other clients or cause more further problems where we need some distance to really think about the best solution."* [S12]

One way to assess the validity of the contribution is to open an issue in the project. The contribution guidelines of the majority of the projects explicitly state that contributors should first open an issue to discuss their contributions and defer the implementation until when the maintainers have agreed on the usefulness of the proposed changes. As a participant responded:

> *"Opening issues to discuss changes prior to posting PRs helps reduce abandoned PRs. Discussing the change ahead of time gives developers and the community time to explore the change and ensure that the 1) proposed change is one that the project wants to maintain, 2) proposed change is scalable, 3) proposed change is a feature that is needed by many use cases and not a one off for specific use case, 4) proposed implementation is maintainable and fits with the architecture and future of the project."* [S6]

Both contributors and maintainers should be more upfront about their intentions (2x). Two participants stated that communication should be improved from both sides. Maintainers need to be upfront about their intentions on merging (or not) the contribution from contributors to avoid wasting effort and time. As a participant responded:

> *"There needs to be a clear signal from the project to the PR contributor if there is no interest at all in the PR, or if there is interest, what needs to be fixed to be accepted. Then participate or help, or signal when the effort can no longer be sustained. If there is no clear signal, the contributor has no idea what's going on. Automated "closer-bots" cannot solve this problem (or make it worse)."* [S8]

Similarly, contributors should mention their willingness to make the requested changes. One participant mentioned that anxiety could play an important factor leading to PR abandonment, particularly when miscommunication happens to newcomers to the project:

> *"Anxiety about contribution probably leads to some abandonment. I suggest to all newish contributors that they remember the people in charge of these projects where just like them once."* [S9]

6.3 How do developers *interpret* our findings?

We provided participants with a pre-print of this manuscript and encouraged participants to report any negative or positive remarks they had about our findings. Out of the 16 participants, 14 participants commented on our findings (87.5%). Table 8 overviews the explanations provided by survey participants regarding each of our main findings. In the following, we discuss the survey responses for each of our findings in more detail.

Table 8. Overview of the explanation of our findings based on our survey responses.

Study Findings	Survey Explanations
Novice contributors are more likely to abandon their PRs.	• Maintainers expect quality changes regardless of the contributor experience. • Novice contributors may find the contribution process difficult.
PRs with long discussions are more likely to get abandoned.	• Long discussions often indicate a controversial PR. • Lack of unanimous decisions lengthens the review process.
Projects have a significant influence on the likelihood of PR abandonment.	• Maintainer team structure, attitude, and workload influence PRs. • Project scope, architecture, and ownership influence PRs.
Complex PRs are more likely to get abandoned.	• Complex PRs are less likely to get a timely review. • Contributor becomes frustrated by frequent change requests.

Complex PRs are more likely to get abandoned. Our findings suggest that PRs with lengthier descriptions, more commits, or more changed lines are more likely to get abandoned (RQ_1–RQ_2). The participants argued that such complex PRs require extra efforts from both their contributors and reviewers. Therefore, such PRs might linger for a while before getting reviewed, and also might require more changes from the contributor to become satisfactory:

> *"The more complex a PR is, the less likely a reviewer is going to spend valuable time on it, in particular if the contributor is not well known."* [S11]
>
> *"Either the maintainers leave them open for months or the contributor is frustrated by the numerous requested changes."* [S9]

Novice contributors are more likely to abandon their PRs. Our findings suggest that contributors who submitted fewer PRs, have a lower acceptance rate, have a lower contribution period, or have a higher abandonment rate within a project are more likely to abandon their PRs (RQ_1–RQ_2). The participants suggested that this can be due to projects expecting high-quality changes (with proper formatting, documentation, and description of changes) that often require access to experienced maintainers. However, projects can lower their expectations from external contributors:

> *"Like any wall in life, it is scary and something to throw yourself against. The project can help—making contributions feel more welcome. Unfortunately if you are too welcoming to contribution you get rapidly overwhelmed by contribution and cannot accept it all."* [S9]
>
> *"The main issue I see with abandoned PRs is that the bar for a commit is too high. To give you some perspective, new developers in my team take a few weeks to land their first commit. And that is with constant access to experienced developers/mentors. This is because we expect our commit to have: proper tests, linting, inline documentation, references to relevant commits, documentation in some cases, good commit message, and targetting the correct branch. I think we should lower the bar for external contributors (so they can quickly land a fix), but eventually, still add an additional commit with a test or something if needed. Clearly, doing that requires some resources. I tried doing that a few years ago, and was quickly slammed with pings everywhere to ask me to work on those PRs!"* [S13]

PRs with long discussions are more likely to get abandoned. Our findings suggest that PRs involving more participants, more responses from the participants or from the contributor, or with higher latency to get a response from a reviewer in a project are more likely to get abandoned (RQ_1–RQ_2). The participants argued that long discussions could indicate a controversial PR which lacks a unanimous solution to address the PR issues:

> *"Long discussions themselves are not a problem. However, they are often indicative of disagreement, a complex topic, or an absence of a solution (the solution in the PR is not correct, but commenters on the PR don't have a good suggestion either)."* [S16]
>
> *"[long discussions] seems like a proxy for controversial PRs, which I suspect is also a factor at play. In my personal experience, open source projects are often lead by passionate individuals with strong opinions about the way things should be done, which often conflict with the opinions of new people to the project. I think this could be addressed by discussing changes beforehand but this is another hurdle which makes contributing more challenging."* [S5]

However, a strong leadership team could prevent such controversial PRs from becoming extra lengthy:

> *"Good projects have strong leaders that make decisions and don't deliberate too long. Bitcoin suffered here greatly after Satoshi left."* [S9]

Projects have a significant influence on the likelihood of PR abandonment. Our findings suggest that projects have a significant influence over PR abandonment and that throughout the history of projects, the rate of PR abandonment has significantly fluctuated as the projects evolved. The participants suggested that such fluctuations might be related to changes in the attitude of the team, scope and architecture of the software, and popularity and ownership of the project:

> *"It's a multitude of factors. Attitudes of maintainers. Software architecture of the projects (if well designed you can expect well designed PRs). Scope of the projects (ie. documentation is clear on the scope to prevent PRs that feature creep). Fame: too big a project will get too much contribution and likely there will be insufficient people to manage it."* [S9]
>
> *"You might also take into account the ownership of the project: is it a community project? A company project? Who is writing the roadmap of the project? A company project has a roadmap in line with its business development, which is not the case for a community project."*

7 DISCUSSION

Combining the results from our quantitative (RQ_1-RQ_2) and qualitative (RQ_3) investigation provides evidence that contributors and the review process play a more prominent role in PR abandonment than projects and PRs themselves. In the following, we integrate the findings from our three research questions and our survey with core developers and further discuss the implications of our findings.

The Role of Contributors in PR Abandonment. Our findings indicate that the contributors of abandoned PRs usually have less experience than the contributors of nonabandoned PRs. Specifically, we observed that novice contributors who have submitted fewer PRs, have a lower acceptance rate, or have a lower contribution period within a project are more likely to abandon their PRs in most of our study projects (RQ_1–RQ_2). Our survey results suggest that novice contributors often find the contribution process more difficult as maintainers typically expect high-quality changes (with proper tests, formatting, documentation, and description of changes) regardless of contributor experience before approving the changes to get merged (Section 6). We also observed that the contributors of abandoned PRs have frequently faced many obstacles (due to lack of enough knowledge, time, or even interest) to continue and complete the review process (RQ_3). Indeed, inexperienced contributors face various barriers in making their contributions accepted [68–70]. Prior studies have also reported the positive impact of contributor experience in acceptance and review time of PRs [26, 41, 66, 79]. Our survey respondents recommend maintainers either lower

their expectations or be more attentive and supportive towards external contributors (especially casual contributors or newcomers) throughout the review process. Also, contributors can discuss their proposed changes before submitting a PR to facilitate the review process, especially if the change introduces new features or involves large changes. This discussion helps contributors to ensure that their proposed changes align with the project roadmap and design. Contributors are also expected to adhere to contribution guidelines and project conventions as it helps them have a better grasp of the review process.

The Role of Review Processes in PR Abandonment. Our findings indicate that the review process of abandoned PRs is usually lengthier than the review process of nonabandoned PRs. Specifically, we observed that lengthy PRs which involve more participants, or more responses from the participants or from the contributor are more likely to get abandoned in most of our study projects (RQ_1–RQ_2). Our survey results suggest that long discussions are often indicative of a controversial PR that is addressing a complex issue or does not have a unanimously accepted solution (Section 6). We also observed that abandoned PRs frequently lack a (timely) review, response, or even action from the maintainers which also unnecessarily lengthen the review process of a PR (RQ_3). Prior studies have also reported that high response latencies and lengthy discussions negatively impact the acceptance and review time of PRs [41, 73, 78, 79]. Our survey respondents recommend maintainers be more responsive and support external contributors (especially casual contributors or newcomers) till the completion of their PRs. In cases that a PR needs only trivial changes, maintainers can merge the PR as is and either implement the changes themselves or open a new issue for the required changes. Also, maintainers can hold recurrent triage meetings to review the status of PRs and support PRs that need attention to mitigate PR abandonment. In cases that a PR has become lengthy, lead maintainers should involve and decide on the outcome of the PR using a voting process.

The Role of Projects in PR Abandonment. Our findings indicate that projects have a significant influence over PR abandonment. Specifically, we observed that the rate of abandoned PRs has significantly fluctuated throughout the history of projects, with some projects constantly decreasing the abandonment rate as they become mature, i.e., Ansible, Kibana, and Odoo (RQ_1-RQ_2). Our survey results suggest that projects typically undergo changes in their team, size, architecture, scope, policies, practices, or even ownership during their development lifecycle. Such changes bring with thems both positive and negative aspects, which can fluctuate the rate of PR abandonment (Section 6). Prior studies have also reported that project maturity has a mixed impact on the acceptance and review time of PRs [73, 79]. Our survey respondents recommend projects to streamline their contribution process as much as possible.

The Role of PRs in PR Abandonment. Our findings indicate that abandoned PRs are usually more complex than nonabandoned PRs. Specifically, we observed that complex PRs which have lengthier descriptions or more commits are more likely to get abandoned in most of our study projects (RQ_1–RQ_2). A PR can be complex at submission time, when it contains too many commits or an abnormally lengthy description, or become more complex as its contributor submit additional commits (and thus makes more changed lines) during the review process to address the changes requested by the maintainers. Our survey results suggest that a complex PR is more likely to linger for a while before getting a first or even a follow-up review, especially if its contributor is not well-known to the maintainers (Section 6). We also observed the lack of review from the maintainers as a frequent reason among abandoned PRs (RQ_3). Prior studies have also reported that complex PRs negatively impact their acceptance and review time [41, 66, 73, 78, 79]. Our survey respondents recommend contributors to make their PRs clear, concise, and focused as complex

PRs are more difficult to review and require more interactions with the contributor to becoming ready (Section 6). Also, maintainers expect PRs to have proper tests, formatting, documentation, and description of changes according to the project requirements (Section 6).

8 RELATED WORK

PR Latency and Decision. The literature has extensively studied how various technical, social, and personal factors influence the acceptance and review process of PRs. Gousios et al. [26] investigated how technical factors affect the merge decision and merge time of PRs. They found that the merge decision is mainly affected by whether the PR touches recently modified code. Also, they observed that the contributor's experience as well as the project's size, test coverage, and openness to external contributions influence the merge time. Gousios et al. [28] found that the decision of maintainers to accept a PR is driven by its quality, especially conformance to the project style and architecture, code quality, and test coverage.

Tsay et al. [73] found that both technical and social factors influence PR acceptance, especially PRs with many comments are less likely to be accepted. Additionally, they observed that well-established projects are more conservative in accepting PRs. Soares et al. [66] found that the programming language, the number of commits, and the number of files added in a PR, as well as whether its contributor is an external developer and whether it is the contributor's first PR, influence the merge decision and merge time. Yu et al. [78] found that the PR size, the first response delay, and the availability of CI pipelines impact the review time of PRs. Yu et al. [79] found that projects prefer PRs that are small, have less controversy, and submitted by trusted contributors.

Kononenko et al. [41] found that the size of PRs, the number of participants in the review discussions, and the contributor's experience and affiliation influences both the review time and merge decision. Moreover, they reported that developers consider PR quality, type of change, and responsiveness of the contributor as important factors in the merge decision. Developers perceive the quality of a PR by its description, complexity, and revertability; and the quality of a review by its feedbacks, tests, and discussions. Pinto et al. [58] found that in company-owned open-source projects, external contributors compared to the employees face significantly more rejections and have to wait longer to receive a decision on their contributions. Zou et al. [84] found that PRs that violate the code style of projects are more likely to get rejected and taker longer to get closed. Lenarduzzi et al. [44] found that code quality does not affect the acceptance of PRs, and suggested that other factors such as the maintainer's reputation and the feature's importance might be more influential on PR acceptance.

Several studies have also investigated how demographic characteristics of contributors can influence the outcome of PRs. Terrell et al. [71] found that among external contributors, women whose gender is identifiable have lower acceptance rates. Rastogi [62] and Rastogi et al. [63] also found that PRs are more likely to get accepted when both the contributors and the maintainers are from the same geographical location. Moreover, Nadri et al. [54] found evidence of bias against perceptible non-White races. Later, Nadri et al. [55] found that contributions from perceptible White developers have a higher acceptance chance, and perceptible non-White contributors are more likely to get their PRs accepted if the maintainer is also from the same race. Furtado et al. [18] also found that contributors from countries with low human development indexes submit fewer PRs but face the most rejections. Beside social and technical factors, Iyer et al. [35] also studied how personality traits [12] influence PR acceptance. They found that personal and technical factors play a significant and comparable role in PR acceptance, but still not to the extent of social factors. Additionally, they observed that contributors who are more open and conscientious but less extroverted have a higher chance of getting their PRs accepted. Similarly, maintainers who are more conscientious, extroverted, and neurotic accept more PRs.

To prioritize PRs, Fan et al. [15] proposed a tool that predicts whether a code change will be merged at early stages based on the features of code, file history, owner experience, collaboration network, and text. Inspired by this work, Islam et al. [34] also proposed a similar tool for early prediction of merged code changes by considering features of reviewer, author, project, text, and code.

Duplicated PRs. Li et al. [48] found that duplicate PRs waste human and computing resources and adversely impact OSS development. To facilitate studies on duplicated PRs, Yu et al. [77] compiled a dataset of duplicated PRs from 26 popular GitHub projects. To identify duplicate PRs, Li et al. [46] proposed an approach that uses textual information within PRs to automatically identify similar PRs. Li et al. [49] extended the previous work by also considering the change information of PRs. Ren et al. [64] proposed an approach to identify redundant code changes in forks as early as possible. Wang et al. [74] enhanced the performance of the previous approach by considering the time factor.

Abandoned PRs. Recently, Li et al. [47] manually examined 321 abandoned PRs from five GitHub projects (namely, Cocos2d-x, Kubernetes, Node.js, Rails, and Rust) to identify the reasons why PRs get abandoned by their contributors, the impact of abandoned PRs on the maintainers, and the strategies adopted by the projects to deal with abandoned PRs. Then, they quantified the frequency of the identified reasons, impacts, and strategies by surveying 710 developers of 100 popular GitHub projects. They found that the reasons why contributors abandon their PRs relate to the lack of maintainers' responsiveness and the lack of contributors' time and interest. While this study discussed the developers' perspective on PR abandonment, the influence of the factors related to PRs, contributors, review processes, and projects on the abandonment probability of PRs is still not known. To fill this knowledge gap, we curated a larger dataset consisting of 10 popular and mature GitHub projects and analyzed abandoned PRs from both a quantitative and qualitative perspective.

9 LIMITATIONS

Threats to Internal Validity. Threats to internal validity are concerned about the issues that might affect our findings. The first threat is related to our definition of abandoned PRs. We define abandoned PRs as those promising PRs that have been neither integrated nor rejected because their contributors have left the review process unfinished. While in our preliminary investigation, we rarely found cases that another developer continues an abandoned PR, but this can be systematically investigated in future studies. The second threat is related to the process of identifying abandoned PRs. Our heuristics may have missed some truly abandoned PRs and wrongly marked some PRs as abandoned. To mitigate this threat, we considered as many relevant keywords as possible by iteratively refining our keywords as we observed new patterns in the discussion comments of known abandoned PRs. Also, we assessed the quality of our dataset by manually investigating 100 abandoned PRs. The third threat is related to the process of identifying the reasons why PRs get abandoned by their contributors. We may have drawn wrong conclusions in card sorting because the coders may have had preconceptions. To minimize this bias, each PR was independently labeled by at least two authors, and then the three authors discussed and merged the labels. The fourth threat is related to the completeness of the abandonment reasons. To further minimize this risk, we coded all the remaining cards when saturation was reached in card sorting. We also performed a second pass over all cards to ensure that we did not miss any important information.

Threats to External Validity. Threats to external validity are concerned about the generalizability of our findings across different projects. To conduct our study, we focused on ten popular GitHub projects with the richest historical PR data. Although the study projects cover several different

application domains and programming languages, they do not represent the entire open-source ecosystem. Therefore, our findings may not generalize beyond our study projects, especially since we observe conflicting patterns across different projects due to their inherent differences. Future replication studies with a more diverse selection of projects both inside and outside the open-source ecosystem are required to obtain more widely applicable insights. Also, our survey findings are based on the responses from 16 participants. While these participants are all among the top core maintainers of the study projects, different maintainers may have different perspectives, and thus our findings may not be generalized to other settings.

10 CONCLUSION

Abandoned PRs waste the time and effort of their contributors and their reviewers. To provide more comprehensive insights into the underlying dynamics of PR abandonment, we conducted a mixed-methods study on ten popular and mature GitHub projects. Using statistical techniques, we found that abandoned PRs tend to be more complex, their contributors tend to be less experienced, and their review processes tend to be lengthier than nonabandoned PRs. We then relied on machine learning techniques to determine the relative importance of the features and describe how they vary the predicted abandonment probability of PRs. We found that the features of review processes, contributors, and projects are more important for predicting PR abandonment than the features of PRs themselves. Specifically, PRs with more than three responses from either the participants or the contributors, and those submitted by novice contributors are more likely to get abandoned. Also, the abandonment probability changes as projects evolve, with half the projects showing a decrease of abandonment in their mature stages and the other half showing an increase of abandonment. To identify the probable reasons why contributors abandon their PRs, we manually examined a random sample of abandoned PRs. We found that difficulty addressing the maintainers' comments, lack of review from the maintainers, difficulty resolving the CI failures, and difficulty resolving the merge issues are the most common reasons why contributors abandon their PRs.

REFERENCES

[1] Daniel W. Apley and Jingyu Zhu. 2020. Visualizing the effects of predictor variables in black box supervised learning models. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)* 82, 4 (2020), 1059–1086. https://doi.org/10.1111/rssb.12377

[2] Mattan S. Ben-Shachar, Daniel Lüdecke, and Dominique Makowski. 2020. effectsize: Estimation of effect size indices and standardized parameters. *Journal of Open Source Software* 5, 56 (2020), 2815. https://doi.org/10.21105/joss.02815

[3] Bernd Bischl, Michel Lang, Lars Kotthoff, Julia Schiffner, Jakob Richter, Erich Studerus, Giuseppe Casalicchio, and Zachary M. Jones. 2016. mlr: Machine learning in R. *Journal of Machine Learning Research* 17, 170 (2016), 1–5. https://jmlr.org/papers/v17/15-066.html

[4] Hudson Borges, Andre Hora, and Marco Tulio Valente. 2016. Understanding the factors that impact the popularity of GitHub repositories. In *ICSME '16: Proceedings of the 32nd IEEE International Conference on Software Maintenance and Evolution*. 334–344. https://doi.org/10.1109/ICSME.2016.31

[5] Hudson Borges and Marco Tulio Valente. 2018. What's in a GitHub star? Understanding repository starring practices in a social coding platform. *Journal of Systems and Software* 146 (2018), 112–129. https://doi.org/10.1016/j.jss.2018.09.016

[6] Andrew P. Bradley. 1997. The use of the area under the roc curve in the evaluation of machine learning algorithms. *Pattern Recognition* 30, 7 (1997), 1145–1159. https://doi.org/10.1016/S0031-3203(96)00142-2

[7] Leo Breiman. 2001. Random forests. *Machine Learning* 45, 1 (2001), 5–32. https://doi.org/10.1023/A:1010933404324

[8] Norman Cliff. 1993. Dominance statistics: Ordinal analyses to answer ordinal questions. *Psychological Bulletin* 114, 3 (1993), 494–509. https://doi.org/10.1037/0033-2909.114.3.494

[9] Jacob Cohen. 1960. A coefficient of agreement for nominal scales. *Educational and Psychological Measurement* 20, 1 (1960), 37–46. https://doi.org/10.1177/001316446002000104

[10] John W. Creswell and J. David Creswell. 2017. *Research design: Qualitative, quantitative, and mixed methods approaches* (5th ed.). SAGE. https://us.sagepub.com/en-us/nam/research-design/book255675

[11] Noah Davis. 2018. 8% of pull requests are doomed. https://codeclimate.com/blog/abandoned-pull-requests/

[12] Colin G. DeYoung, Lena C. Quilty, and Jordan B. Peterson. 2007. Between facets and domains: 10 aspects of the big five. *Journal of Personality and Social Psychology* 93, 5 (2007), 880–896. https://doi.org/10.1037/0022-3514.93.5.880

[13] Carsten F. Dormann, Jane Elith, Sven Bacher, Carsten Buchmann, Gudrun Carl, Gabriel Carré, Jaime R. García Marquéz, Bernd Gruber, Bruno Lafourcade, Pedro J. Leitão, Tamara Münkemüller, Colin McClean, Patrick E. Osborne, Björn Reineking, Boris Schröder, Andrew K. Skidmore, Damaris Zurell, and Sven Lautenbach. 2013. Collinearity: A review of methods to deal with it and a simulation study evaluating their performance. *Ecography* 36, 1 (2013), 27–46. https://doi.org/10.1111/j.1600-0587.2012.07348.x

[14] James D. Evans. 1996. *Straightforward statistics for the behavioral sciences.* Brooks/Cole Pub. Co, Pacific Grove.

[15] Y. Fan, X. Xia, D. Lo, and S. Li. 2018. Early prediction of merged code changes to prioritize reviewing tasks. *Empirical Software Engineering* 23, 6 (2018), 3346–3393. https://doi.org/10.1007/s10664-018-9602-0

[16] Tom Fawcett. 2006. An introduction to ROC analysis. *Pattern Recognition Letters* 27, 8 (June 2006), 861–874. https://doi.org/10.1016/j.patrec.2005.10.010

[17] Aaron Fisher, Cynthia Rudin, and Francesca Dominici. 2019. All models are wrong, but many are useful: Learning a variable's importance by studying an entire class of prediction models simultaneously. *Journal of Machine Learning Research* 20, 177 (2019), 1–81. https://jmlr.org/papers/v20/18-760.html

[18] Leonardo B. Furtado, Bruno Cartaxo, Christoph Treude, and Gustavo Pinto. 2021. How successful are open source contributions from countries with different levels of human development? *IEEE Software* 38, 2 (2021), 58–63. https://doi.org/10.1109/MS.2020.3044020

[19] GitHub. 2020. The state of the Octoverse. https://octoverse.github.com/

[20] GitHub. 2021. Enums - GitHub Docs. https://docs.github.com/en/graphql/reference/enums

[21] GitHub. 2021. GitHub REST API - GitHub Docs. https://docs.github.com/en/rest

[22] GitHub. 2021. Issues - GitHub Docs. https://docs.github.com/en/rest/reference/issues

[23] GitHub. 2021. Pulls - GitHub Docs. https://docs.github.com/en/rest/reference/pulls

[24] GitHub. 2021. Stale. https://github.com/marketplace/stale

[25] Georgios Gousios. 2013. The GHTorent dataset and tool suite. In *MSR '13: Proceedings of the 10th Working Conference on Mining Software Repositories*. 233–236. https://doi.org/10.1109/MSR.2013.6624034

[26] Georgios Gousios, Martin Pinzger, and Arie van Deursen. 2014. An exploratory study of the pull-based software development model. In *ICSE '14: Proceedings of the 36th ACM/IEEE International Conference on Software Engineering*. 345–355. https://doi.org/10.1145/2568225.2568260

[27] Georgios Gousios and Andy Zaidman. 2014. A dataset for pull-based development research. In *MSR '14: Proceedings of the 11th Working Conference on Mining Software Repositories*. 368–371. https://doi.org/10.1145/2597073.2597122

[28] Georgios Gousios, Andy Zaidman, Margaret-Anne Storey, and Arie van Deursen. 2015. Work practices and challenges in pull-based development: The integrator's perspective. In *ICSE '15: Proceedings of the 37th International Conference on Software Engineering*. IEEE, 358–368. https://doi.org/10.1109/ICSE.2015.55

[29] Ilya Grigorik. 2021. GH Archive. https://www.gharchive.org/

[30] Haibo He and E.A. Garcia. 2009. Learning from Imbalanced Data. *IEEE Transactions on Knowledge and Data Engineering* 21, 9 (Sept. 2009), 1263–1284. https://doi.org/10.1109/TKDE.2008.239

[31] Frank E. Harrell. 2021. Hmisc: Harrell Miscellaneous. https://CRAN.R-project.org/package=Hmisc

[32] Melinda R. Hess and Jeffrey D. Kromrey. 2004. Robust confidence intervals for effect sizes: A comparative study of Cohen's d and Cliff's delta under non-normality and heterogeneous variances. In *AERA '04: Annual Meeting of the American Educational Research Association*. 1–30. https://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.487.8299

[33] Jerry L. Hintze and Ray D. Nelson. 1998. Violin plots: A box plot-density trace synergism. *The American Statistician* 52, 2 (1998), 181–184. https://doi.org/10.1080/00031305.1998.10480559

[34] Khairul Islam, Toufique Ahmed, Rifat Shahriyar, Anindya Iqbal, and Gias Uddin. 2022. Early prediction for merged vs abandoned code changes in modern code reviews. *Information and Software Technology* 142 (2022), 106756. https://doi.org/10.1016/j.infsof.2021.106756

[35] Rahul N. Iyer, S. Alex Yun, Meiyappan Nagappan, and Jesse Hoey. 2019. Effects of personality traits on pull request acceptance. *IEEE Transactions on Software Engineering* early access (2019). https://doi.org/10.1109/TSE.2019.2960357

[36] Vincent Jacques. 2021. PyGithub. https://github.com/PyGithub/PyGithub

[37] Donald R. Jones, Matthias Schonlau, and William J. Welch. 1998. Efficient global optimization of expensive black-box functions. *Journal of Global Optimization* 13, 4 (1998), 455–492. https://doi.org/10.1023/A:1008306431147

[38] Eirini Kalliamvakou, Georgios Gousios, Kelly Blincoe, Leif Singer, Daniel M. German, and Daniela Damian. 2016. An in-depth study of the promises and perils of mining GitHub. *Empirical Software Engineering* 21, 5 (2016), 2035–2071. https://doi.org/10.1007/s10664-015-9393-5

[39] SayedHassan Khatoonabadi, Shahriar Lotfi, and Ayaz Isazadeh. 2021. *GAP2WSS: A genetic algorithm based on the Pareto principle for Web service selection.* arXiv:2109.10430 [cs.NE]. http://arxiv.org/abs/2109.10430

[40] Roger E. Kirk. 1996. Practical significance: A concept whose time has come. *Educational and Psychological Measurement* 56, 5 (1996), 746–759. https://doi.org/10.1177/0013164496056005002

[41] Oleksii Kononenko, Tresa Rose, Olga Baysal, Michael Godfrey, Dennis Theisen, and Bart de Water. 2018. Studying pull request merges: A case study of Shopify's Active Merchant. In *ICSE-SEIP '18: Proceedings of the 40th ACM/IEEE International Conference on Software Engineering - Software Engineering in Practice*. 124–133. https://doi.org/10.1145/3183519.3183542

[42] Kubernetes. 2021. fejta-bot - GitHub user. https://github.com/fejta-bot

[43] J. Richard Landis and Gary G. Koch. 1977. The measurement of observer agreement for categorical data. *Biometrics* 33, 1 (1977), 159–174. https://doi.org/10.2307/2529310

[44] Valentina Lenarduzzi, Vili Nikkola, Nyyti Saarimäki, and Davide Taibi. 2021. Does code quality affect pull request acceptance? An empirical study. *Journal of Systems and Software* 171 (2021), 110806. https://doi.org/10.1016/j.jss.2020.110806

[45] Timothy C. Lethbridge, Susan Elliott Sim, and Janice Singer. 2005. Studying software engineers: Data collection techniques for software field studies. *Empirical Software Engineering* 10, 3 (2005), 311–341. https://doi.org/10.1007/s10664-005-1290-x

[46] Zhixing Li, Gang Yin, Yue Yu, Tao Wang, and Huaimin Wang. 2017. Detecting duplicate pull-requests in GitHub. In *Internetware '17: Proceedings of the 9th Asia-Pacific Symposium on Internetware*. 1–6. https://doi.org/10.1145/3131704.3131725

[47] Zhixing Li, Yue Yu, Tao Wang, Gang Yin, Shanshan Li, and Huaimin Wang. 2021. Are you still working on this? An empirical study on pull request abandonment. *IEEE Transactions on Software Engineering* early access (2021). https://doi.org/10.1109/TSE.2021.3053403

[48] Zhixing Li, Yue Yu, Minghui Zhou, Tao Wang, Gang Yin, Long Lan, and Huaimin Wang. 2020. Redundancy, context, and preference: An empirical study of duplicate pull requests in OSS projects. *IEEE Transactions on Software Engineering* early access (2020). https://doi.org/10.1109/TSE.2020.3018726

[49] Zhi-Xing Li, Yue Yu, Tao Wang, Gang Yin, Xin-Jun Mao, and Huai-Min Wang. 2021. Detecting duplicate contributions in pull-based model combining textual and change similarities. *Journal of Computer Science and Technology* 36, 1 (2021), 191–206. https://doi.org/10.1007/s11390-020-9935-1

[50] Callum Macrae. 2014. Added pixelmator. #4781. https://github.com/Homebrew/homebrew-cask/pull/4781

[51] Henry B. Mann and Donald R. Whitney. 1947. On a test of whether one of two random variables is stochastically larger than the other. *The Annals of Mathematical Statistics* 18, 1 (1947), 50–60. https://doi.org/10.1214/aoms/1177730491

[52] Christoph Molnar. 2021. *Interpretable Machine Learning: A Guide for Making Black Box Models Explainable*. https://christophm.github.io/interpretable-ml-book/

[53] Christoph Molnar, Giuseppe Casalicchio, and Bernd Bischl. 2018. iml: An R package for interpretable machine learning. *Journal of Open Source Software* 3, 27 (2018), 786. https://doi.org/10.21105/joss.00786

[54] Reza Nadri, Gema Rodríguez-Pérez, and Meiyappan Nagappan. 2021. Insights into nonmerged pull requests in GitHub: Is there evidence of bias based on perceptible race? *IEEE Software* 38, 2 (2021), 51–57. https://doi.org/10.1109/MS.2020.3036758

[55] Reza Nadri, Gema Rodríguez-Pérez, and Meiyappan Nagappan. 2021. On the relationship between the developer's perceptible race and ethnicity and the evaluation of contributions in OSS. *IEEE Transactions on Software Engineering* early access (2021). https://doi.org/10.1109/TSE.2021.3073773

[56] Indrajeet Patil. 2021. Visualizations with statistical details: The 'ggstatsplot' approach. *Journal of Open Source Software* 6, 61 (2021), 3167. https://doi.org/10.21105/joss.03167

[57] Fabian Pedregosa, Gaël Varoquaux, Alexandre Gramfort, Vincent Michel, Bertrand Thirion, Olivier Grisel, Mathieu Blondel, Peter Prettenhofer, Ron Weiss, Vincent Dubourg, Jake Vanderplas, Alexandre Passos, David Cournapeau, Matthieu Brucher, Matthieu Perrot, and Édouard Duchesnay. 2011. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research* 12, 85 (2011), 2825–2830. http://jmlr.org/papers/v12/pedregosa11a.html

[58] Gustavo Pinto, Luiz Felipe Dias, and Igor Steinmacher. 2018. Who gets a patch accepted first? Comparing the contributions of employees and volunteers. In *CHASE '18: Proceedings of the 11th ACM/IEEE International Workshop on Cooperative and Human Aspects of Software Engineering*. 110–113. https://doi.org/10.1145/3195836.3195858

[59] Gustavo Pinto, Igor Steinmacher, and Marco Aurélio Gerosa. 2016. More common than you think: An in-depth study of casual contributors. In *SANER '16: Proceedings of the 23rd IEEE International Conference on Software Analysis, Evolution, and Reengineering*. 112–123. https://doi.org/10.1109/SANER.2016.68

[60] Philipp Probst, Marvin N. Wright, and Anne-Laure Boulesteix. 2019. Hyperparameters and tuning strategies for random forest. *WIREs Data Mining and Knowledge Discovery* 9, 3 (2019), e1301. https://doi.org/10.1002/widm.1301

[61] R Core Team. 2021. R: A language and environment for statistical computing. https://www.R-project.org

[62] Ayushi Rastogi. 2016. Do biases related to geographical location influence work-related decisions in GitHub?. In *ICSE '16: Proceedings of the 38th ACM/IEEE International Conference on Software Engineering Companion*. 665–667.

https://doi.org/10.1145/2889160.2891035

[63] Ayushi Rastogi, Nachiappan Nagappan, Georgios Gousios, and André van der Hoek. 2018. Relationship between geographical location and evaluation of developer contributions in GitHub. In *ESEM '18: Proceedings of the 12th ACM/IEEE International Symposium on Empirical Software Engineering and Measurement*. 1–8. https://doi.org/10.1145/3239235.3240504

[64] Luyao Ren, Shurui Zhou, Christian Kästner, and Andrzej Wąsowski. 2019. Identifying redundancies in fork-based development. In *SANER '19: Proceedings of the 26th IEEE International Conference on Software Analysis, Evolution and Reengineering*. 230–241. https://doi.org/10.1109/SANER.2019.8668023

[65] C.B. Seaman. 1999. Qualitative methods in empirical studies of software engineering. *IEEE Transactions on Software Engineering* 25, 4 (Aug. 1999), 557–572. https://doi.org/10.1109/32.799955

[66] Daricélio Moreira Soares, Manoel Limeira de Lima Júnior, Leonardo Murta, and Alexandre Plastino. 2015. Acceptance factors of pull requests in open-source projects. In *SAC '15: Proceedings of the 30th Annual ACM Symposium on Applied Computing*. 1541–1546. https://doi.org/10.1145/2695664.2695856

[67] Charles Spearman. 2010. The proof and measurement of association between two things. *International Journal of Epidemiology* 39, 5 (2010), 1137–1150. https://doi.org/10.1093/ije/dyq191

[68] Igor Steinmacher, Ana Paula Chaves, Tayana Conte, and Marco Aurelio Gerosa. 2014. Preliminary empirical identification of barriers faced by newcomers to open source software projects. In *SBES '14: Proceedings of the 28th Brazilian Symposium on Software Engineering*. 51–60. https://doi.org/10.1109/SBES.2014.9

[69] Igor Steinmacher, Gustavo Pinto, Igor Scaliante Wiese, and Marco Aurélio Gerosa. 2018. Almost there: A study on quasi-contributors in open source software projects. In *ICSE '18: Proceedings of the 40th International Conference on Software Engineering*. ACM, 256–266. https://doi.org/10.1145/3180155.3180208

[70] Igor Steinmacher, Igor Wiese, Ana Paula Chaves, and Marco Aurélio Gerosa. 2013. Why do newcomers abandon open source software projects?. In *CHASE '13: Proceedings of the 6th International Workshop on Cooperative and Human Aspects of Software Engineering*. 25–32. https://doi.org/10.1109/CHASE.2013.6614728

[71] Josh Terrell, Andrew Kofink, Justin Middleton, Clarissa Rainear, Emerson Murphy-Hill, Chris Parnin, and Jon Stallings. 2017. Gender differences and bias in open source: Pull request acceptance of women versus men. *PeerJ Computer Science* 3 (2017), e111. https://doi.org/10.7717/peerj-cs.111

[72] Orta Therox. 2020. Changes to how we manage DefinitelyTyped. https://devblogs.microsoft.com/typescript/changes-to-how-we-manage-definitelytyped/

[73] Jason Tsay, Laura Dabbish, and James Herbsleb. 2014. Influence of social and technical factors for evaluating contribution in GitHub. In *ICSE '14: Proceedings of the 36th ACM/IEEE International Conference on Software Engineering*. 356–366. https://doi.org/10.1145/2568225.2568315

[74] Qingye Wang, Bowen Xu, Xin Xia, Ting Wang, and Shanping Li. 2019. Duplicate pull request detection: When time matters. In *Internetware '19: Proceedings of the 11th Asia-Pacific Symposium on Internetware*. 1–10. https://doi.org/10.1145/3361242.3361254

[75] Mairieli Wessel, Igor Steinmacher, Igor Wiese, and Marco A. Gerosa. 2019. Should I stale or should I close? An analysis of a bot that closes abandoned issues and pull requests. In *BotSE '19: Proceedings of the 1st IEEE/ACM International Workshop on Bots in Software Engineering*. 38–42. https://doi.org/10.1109/BotSE.2019.00018

[76] Marvin N. Wright and Andreas Ziegler. 2017. ranger: A fast implementation of random forests for high dimensional data in C++ and R. *Journal of Statistical Software* 77, 1 (2017), 1–17. https://doi.org/10.18637/jss.v077.i01

[77] Yue Yu, Zhixing Li, Gang Yin, Tao Wang, and Huaimin Wang. 2018. A dataset of duplicate pull-requests in GitHub. In *MSR '18: Proceedings of the 15th ACM/IEEE International Conference on Mining Software Repositories*. https://doi.org/10.1145/3196398.3196455

[78] Yue Yu, Huaimin Wang, Vladimir Filkov, Premkumar Devanbu, and Bogdan Vasilescu. 2015. Wait for it: Determinants of pull request evaluation latency on GitHub. In *MSR '15: Proceedings of the 12th Working Conference on Mining Software Repositories*. IEEE, 367–371. https://doi.org/10.1109/MSR.2015.42

[79] Yue Yu, Gang Yin, Tao Wang, Cheng Yang, and Huaimin Wang. 2016. Determinants of pull-based development in the context of continuous integration. *Science China Information Sciences* 59, 8 (2016), 080104. https://doi.org/10.1007/s11432-016-5595-8

[80] Xunhui Zhang, Ayushi Rastogi, and Yue Yu. 2020. On the shoulders of giants: A new dataset for pull-based development research. In *MSR '20: Proceedings of the 17th ACM/IEEE International Conference on Mining Software Repositories*. 543–547. https://doi.org/10.1145/3379597.3387489

[81] Xunhui Zhang, Yue Yu, Georgios Gousios, and Ayushi Rastogi. 2021. *Pull request decision explained: An empirical overview*. arXiv:2105.13970 [cs.SE]. http://arxiv.org/abs/2105.13970

[82] Xunhui Zhang, Yue Yu, Tao Wang, Ayushi Rastogi, and Huaimin Wang. 2021. *Pull request latency explained: An empirical overview*. arXiv:2108.09946 [cs.SE]. http://arxiv.org/abs/2108.09946

[83] Jiaxin Zhu, Minghui Zhou, and Audris Mockus. 2016. Effectiveness of code contribution: From patch-based to pull-request-based tools. In *FSE '16: Proceedings of the 24th ACM SIGSOFT International Symposium on Foundations of Software Engineering*. 871–882. https://doi.org/10.1145/2950290.2950364

[84] Weiqin Zou, Jifeng Xuan, Xiaoyuan Xie, Zhenyu Chen, and Baowen Xu. 2019. How does code style inconsistency affect pull request integration? An exploratory study on 117 GitHub projects. *Empirical Software Engineering* 24, 6 (2019), 3871–3903. https://doi.org/10.1007/s10664-019-09720-x

APPENDIX

A COMPARISON OF ABANDONED AND NONABANDONED PRS

A.1 PR Features:

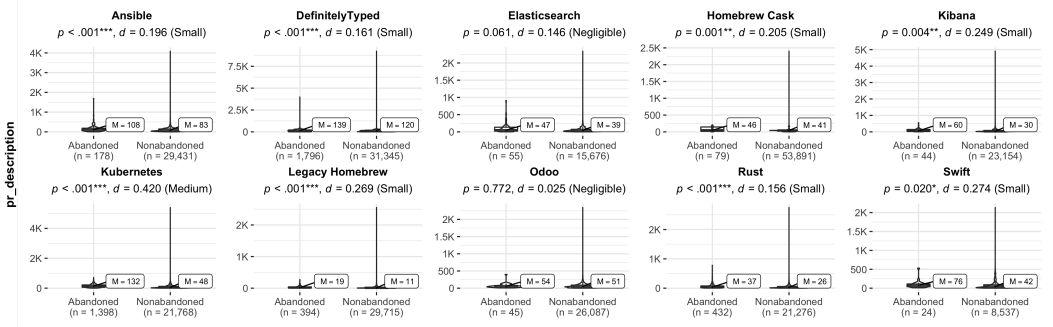

Fig. 7. Comparison of abandoned and nonabandoned PRs wrt *pr_description* across the study projects.

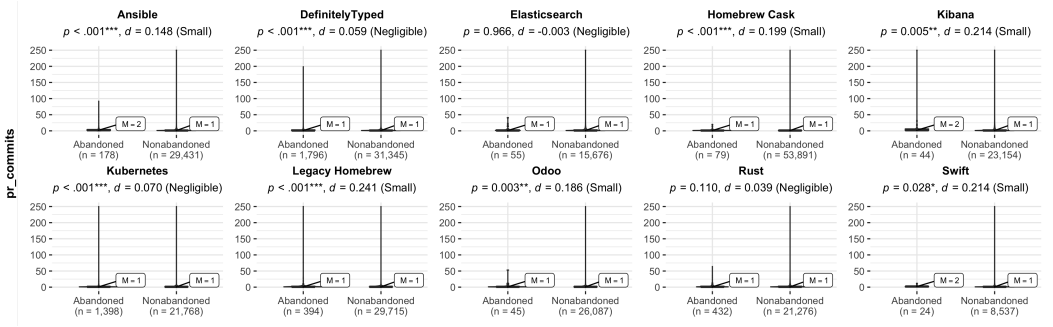

Fig. 8. Comparison of abandoned and nonabandoned PRs wrt *pr_commits* across the study projects.

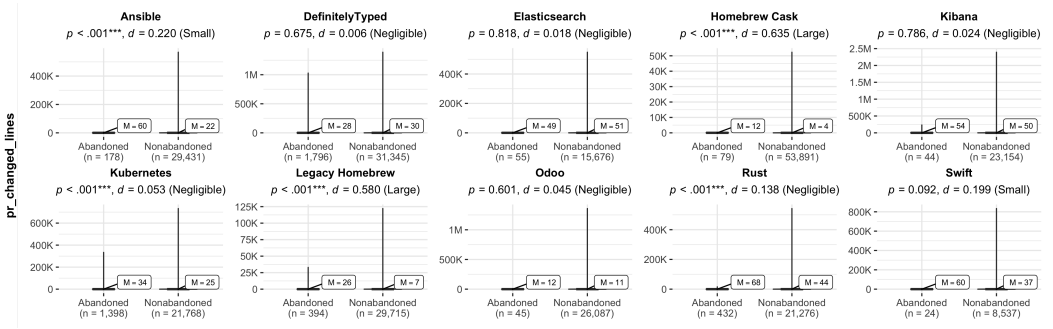

Fig. 9. Comparison of abandoned and nonabandoned PRs wrt *pr_changed_lines* across the study projects.

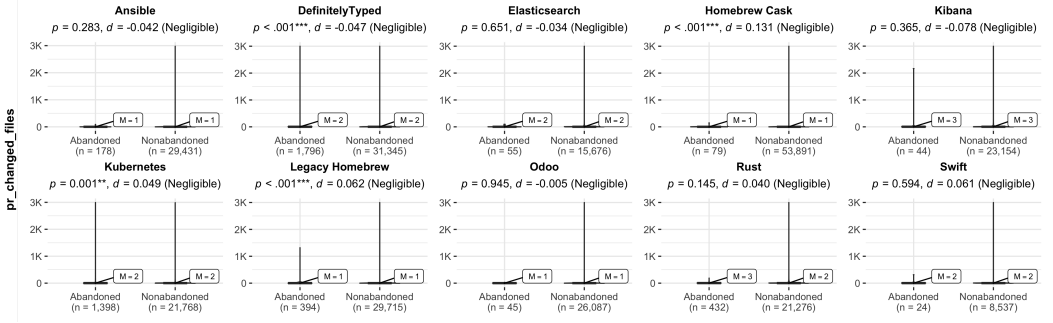

Fig. 10. Comparison of abandoned and nonabandoned PRs wrt *pr_changed_files* across the study projects.

A.2 Contributor Features:

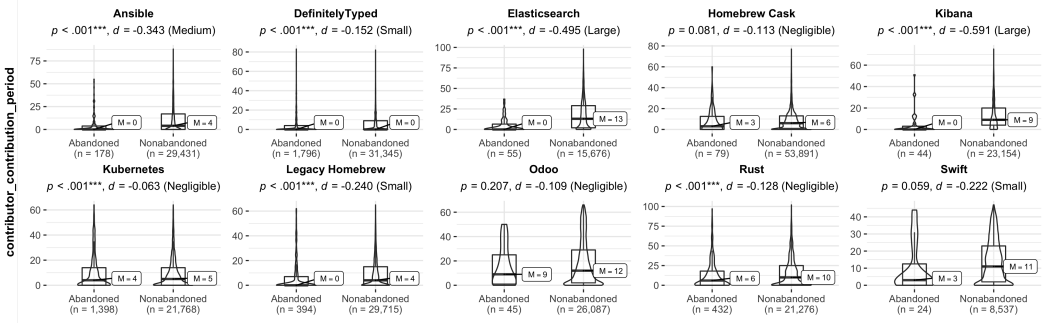

Fig. 11. Comparison of abandoned and nonabandoned PRs wrt *contributor_contribution_period* across the study projects.

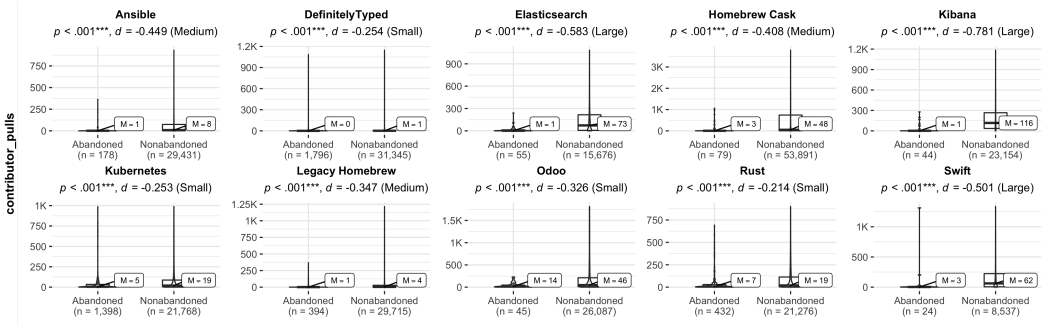

Fig. 12. Comparison of abandoned and nonabandoned PRs wrt *contributor_pulls* across the study projects.

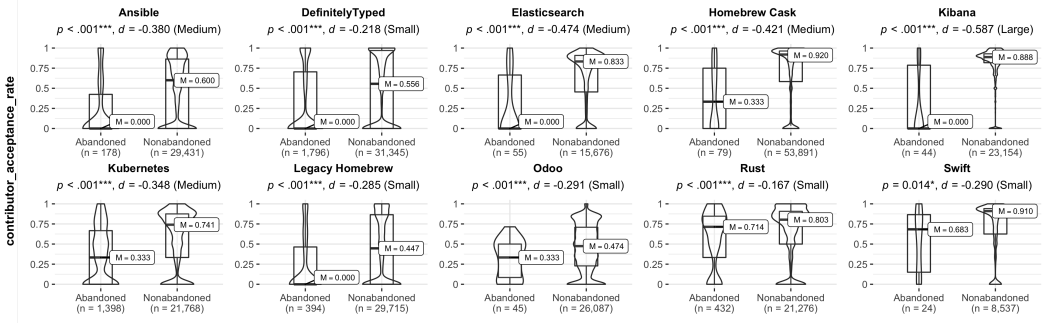

Fig. 13. Comparison of abandoned and nonabandoned PRs wrt *contributor_acceptance_rate* across the study projects.

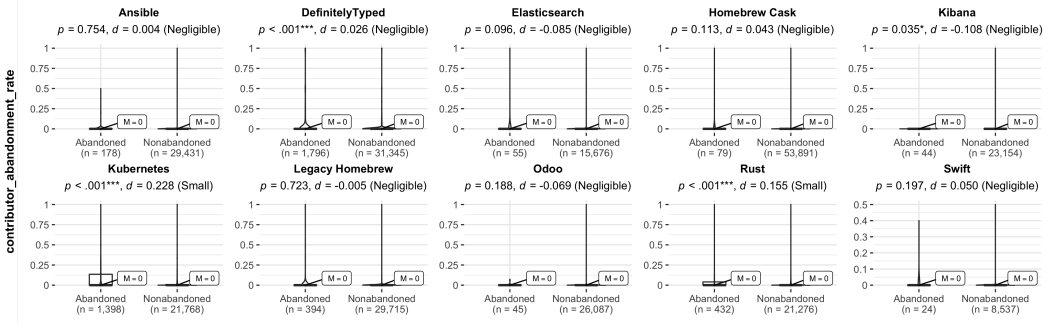

Fig. 14. Comparison of abandoned and nonabandoned PRs wrt *contributor_abandonment_rate* across the study projects.

A.3 Review Process Features:

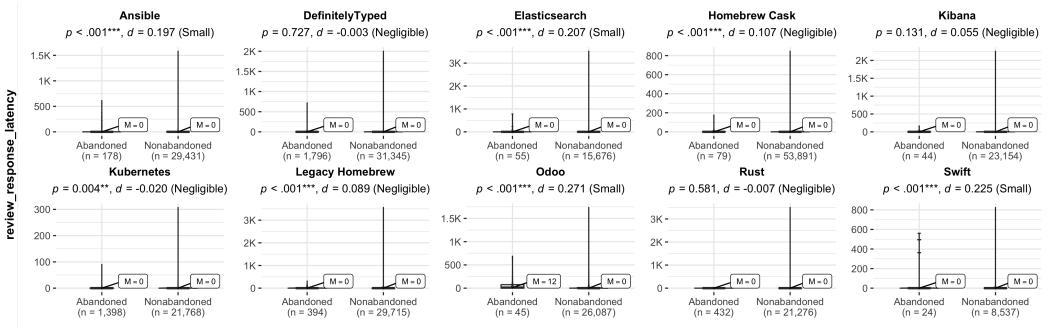

Fig. 15. Comparison of abandoned and nonabandoned PRs wrt *review_response_latency* across the study projects.

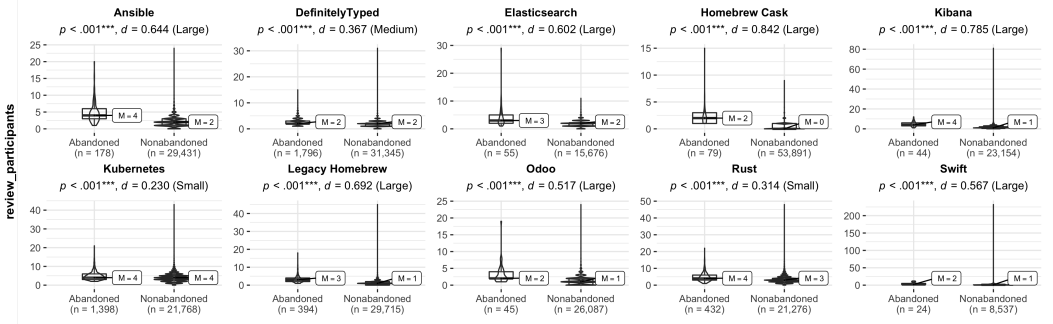

Fig. 16. Comparison of abandoned and nonabandoned PRs wrt *review_participants* across the study projects.

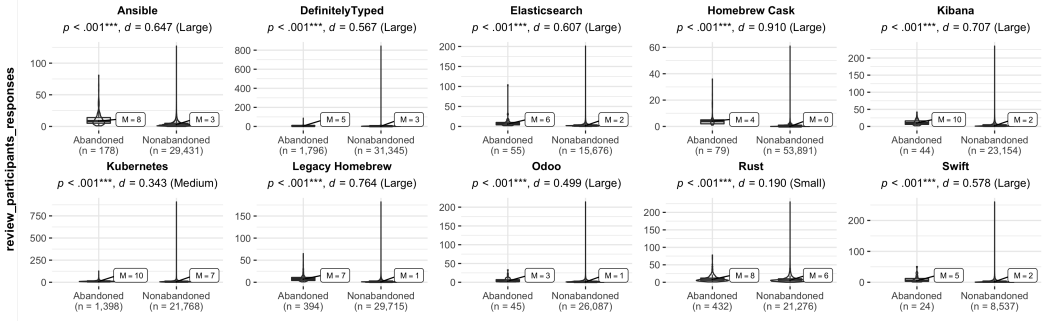

Fig. 17. Comparison of abandoned and nonabandoned PRs wrt *review_participants_responses* across the study projects.

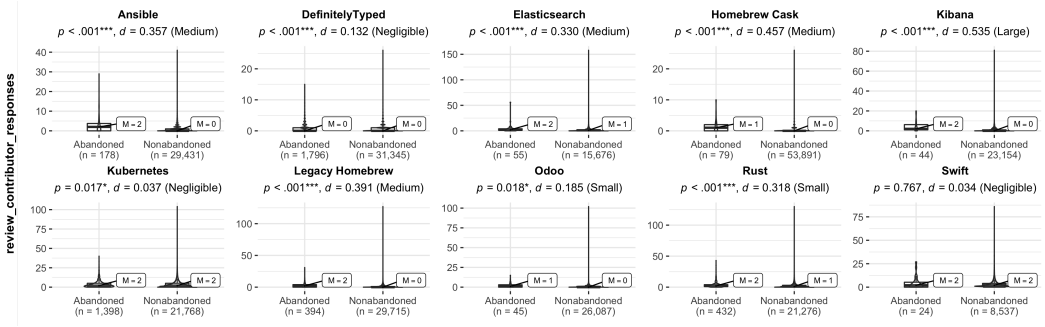

Fig. 18. Comparison of abandoned and nonabandoned PRs wrt *review_contributor_responses* across the study projects.

A.4 Project Features:

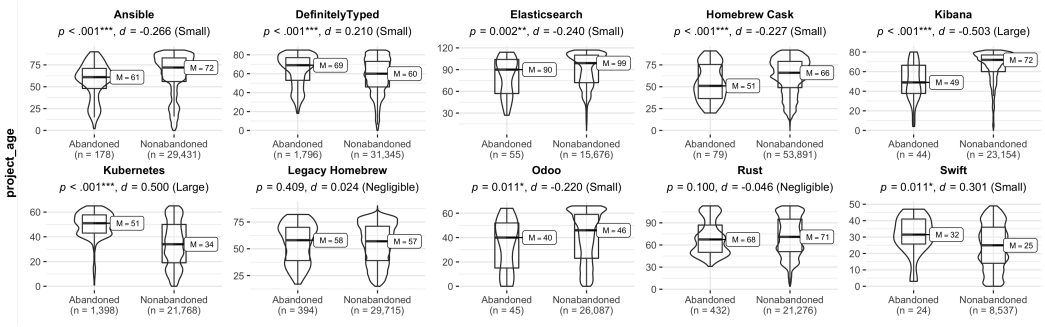

Fig. 19. Comparison of abandoned and nonabandoned PRs wrt *project_age* across the study projects.

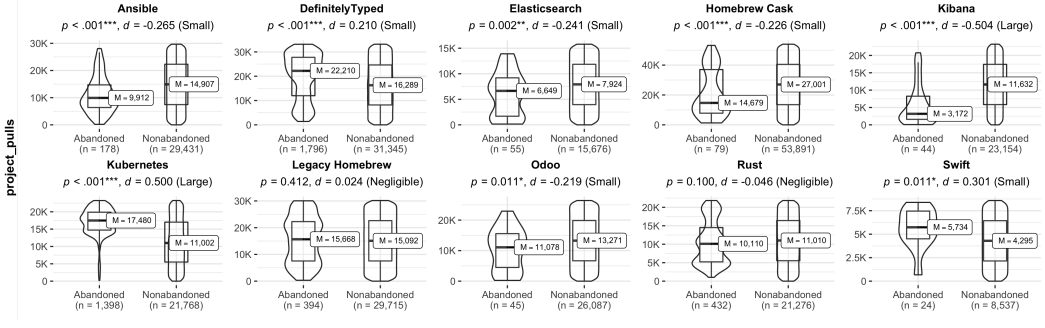

Fig. 20. Comparison of abandoned and nonabandoned PRs wrt *project_pulls* across the study projects.

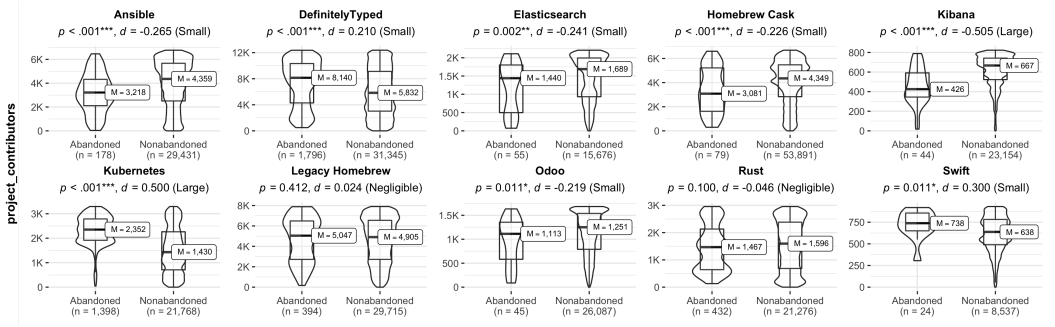

Fig. 21. Comparison of abandoned and nonabandoned PRs wrt *project_contributors* across the study projects.

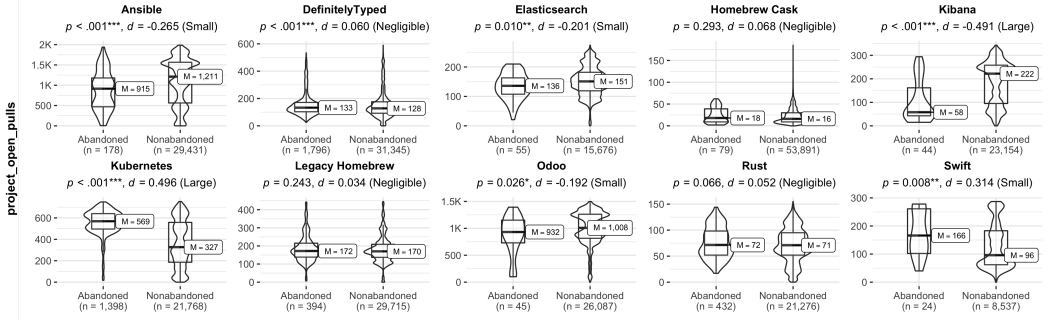

Fig. 22. Comparison of abandoned and nonabandoned PRs wrt *project_open_pulls* across the study projects.

B ALE PLOTS FOR DIFFERENT FEATURES

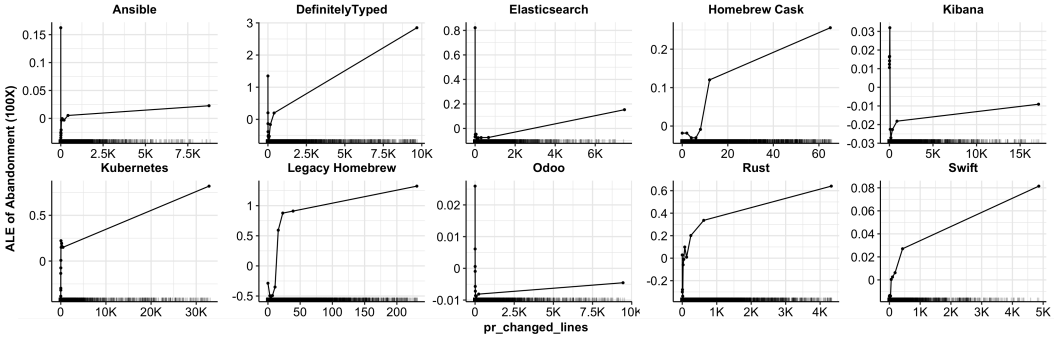

Fig. 23. ALE plots showing how *pr_changed_lines* varies the abandonment probability of PRs.

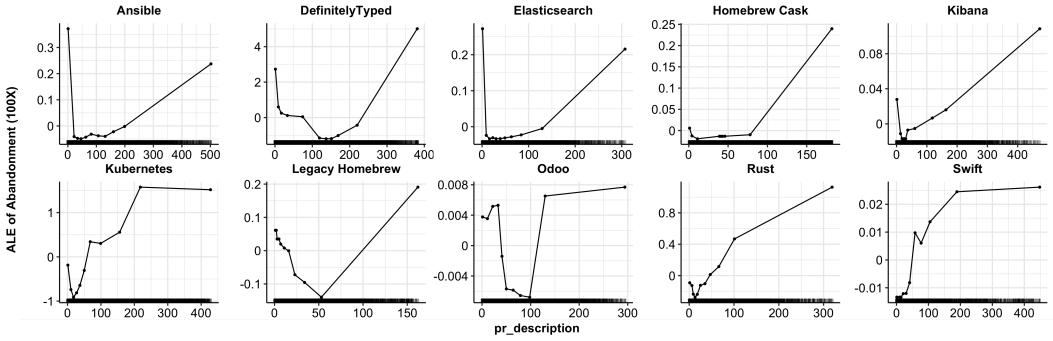

Fig. 24. ALE plots showing how *pr_description* varies the abandonment probability of PRs.

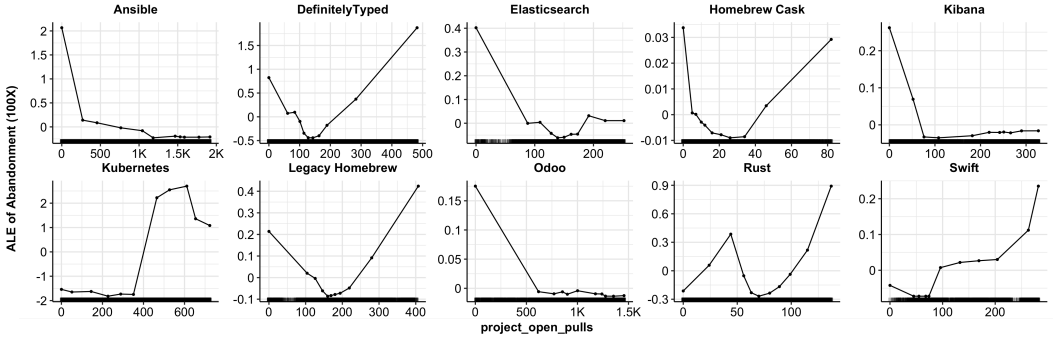

Fig. 25. ALE plots showing how *project_open_pulls* varies the abandonment probability of PRs.

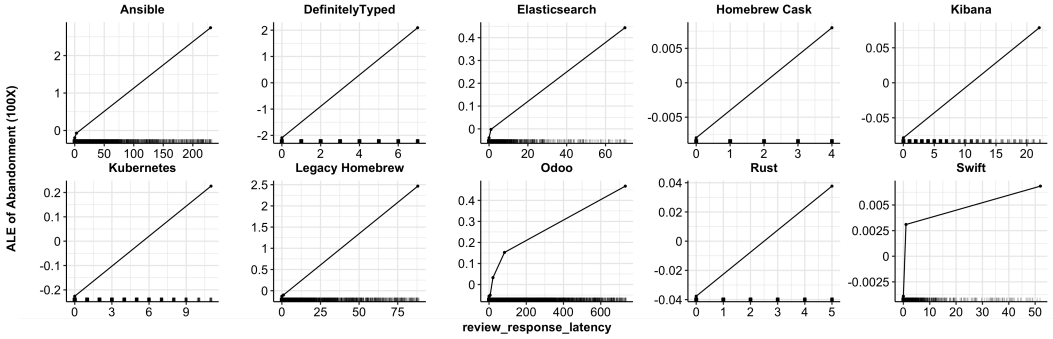

Fig. 26. ALE plots showing how *review_response_latency* varies the abandonment probability of PRs.

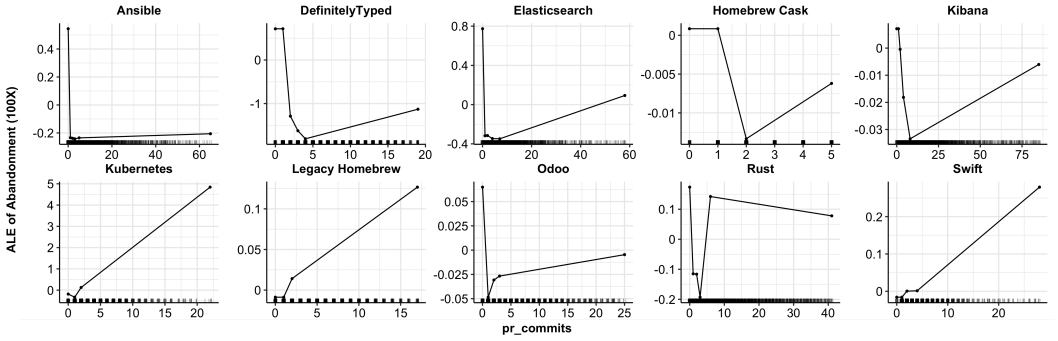

Fig. 27. ALE plots showing how *pr_commits* varies the abandonment probability of PRs.

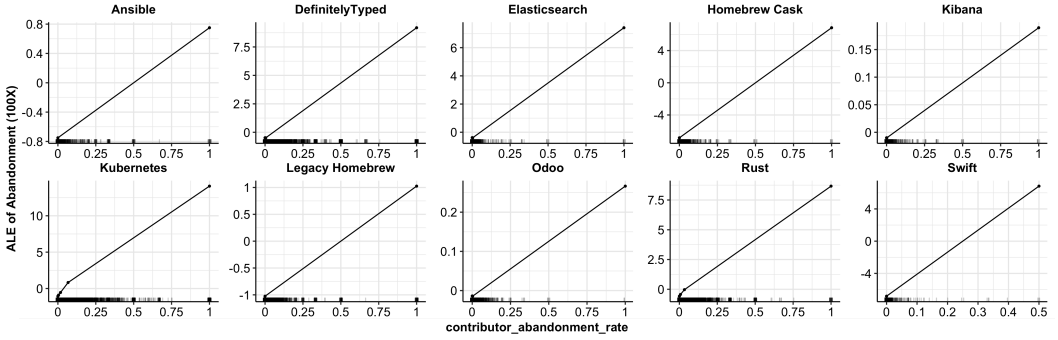

Fig. 28. ALE plots showing how *contributor_abandonment_rate* varies the abandonment probability of PRs.