Pull Request Decisions Explained
An Empirical Overview

Zhang, Xunhui; Yu, Yue; Georgios, Gousios; Rastogi, Ayushi

DOI
10.1109/TSE.2022.3165056

Publication date
2023

Document Version
Accepted author manuscript

Published in
IEEE Transactions on Software Engineering

Pull Request Decisions Explained: An Empirical Overview

Xunhui Zhang, Yue Yu*, Georgios Gousios, and Ayushi Rastogi

Abstract—*Context*: The pull-based development model is widely used in open source projects, leading to the emergence of trends in distributed software development. One aspect that has garnered significant attention concerning pull request decisions is the identification of explanatory factors. *Objective*: This study builds on a decade of research on pull request decisions and provides further insights. We empirically investigate how factors influence pull request decisions and the scenarios that change the influence of such factors. *Method*: We identify factors influencing pull request decisions on GitHub through a systematic literature review and infer them by mining archival data. We collect a total of 3,347,937 pull requests with 95 features from 11,230 diverse projects on GitHub. Using these data, we explore the relations among the factors and build mixed effects logistic regression models to empirically explain pull request decisions. *Results*: Our study shows that a small number of factors explain pull request decisions, with that concerning whether the integrator is the same as or different from the submitter being the most important factor. We also note that the influence of factors on pull request decisions change with a change in context; *e.g.*, the area hotness of pull request is important only in the early stage of project development, however it becomes unimportant for pull request decisions as projects become mature.

Index Terms—pull-based development, pull request decision, distributed software development, GitHub

1 Introduction

The pull-based development model is an important paradigm for global collaboration in open source projects. In this model [1], contributors *(also known as requesters and submitters)* submit their proposed code changes to a base repository by creating a pull request from their cloned repository for the reviewers to inspect. The integrator *(also known as the closer and the merger)* evaluates the proposed changes and decides whether to accept or reject the pull request. However, this process is made complex by additional actors and mechanisms. For instance, during the review, anyone can discuss the feature(s), correctness, etc., of the pull request. Moreover, DevOps tools that automatically check code adaptability and provide results to contributors and integrators exist.

Many studies on understanding pull-based development have emerged in recent years to improve developer contributions, balance integrators' workloads, optimize review processes, etc. There are studies on pull request decisions [2], their latency [3], reviewer recommendations [4], [5], the duplication of pull requests [6], [7], the automatic generation of pull request descriptions [8], and the prioritization of pull request lists [9], among others. This study focuses on explaining pull request decisions.

* is the corresponding author

- X. Zhang and Y. Yu are with the National Key Lab of Parallel Distribution, National University of Defense Technology, Changsha, Hunan 410073, China. E-mail: {zhangxunhui, yuyue}@nudt.edu.cn.
- A. Rastogi is with the Faculty of Science and Engineering, the University of Groningen, The Netherlands. A part of the work was performed while the author was affiliated to TU Delft. E-mail: a.rastogi@rug.nl.
- G. Gousios is with the TU Delft. E-mail: g.gousios@tudelft.nl.

Many studies have made strides in explaining pull request decisions by introducing new factors in the past decade. Some examples of these factors are continuous integration (CI) [10], [11], geographical location [12], and bot usage [13], [14]. Relatedly, a few studies have presented a list of factors that can influence pull request decisions. One outstanding work along this line of Gousios et al. [1] provided a list of developer, project, and pull request characteristics. Tsay et al. [15] split factors into two categories, *i.e.*, social- and technical-related factors. A more recent study by Dey et al. [16] combined many such factors (50) to rank their importance for prediction.

While several studies have contributed individual pieces to understand pull request decisions, a systematic synthesis of the body of knowledge to explain such decisions is missing. If new mechanisms emerge and a new set of factors occurs. Researchers need to decide which factors are more critical when selecting control variables for an empirical study to find their impact on pull request decisions. However, there lack relevant studies to tell them how to make choices. Also, understanding factors' influence in different contexts is essential for researchers to select projects and factors. From developers' perspectives, when creating predictive tools, it is also important to consider the impact of different contexts. *E.g.*, how to choose factors if reviewers comment during the review process? What factors should be considered if a pull request uses CI tools? Factors, if properly selected, not only maintain accuracy but also significantly improve the efficiency of decision prediction. Therefore, our current work presents an empirical investigation explaining pull request decisions from GitHub in terms of the factors known to influence them. Particularly, we explore the following two research questions:

RQ1 *How do these factors influence pull request decisions?*

RQ2 *How do the factors influencing pull request decisions change with a change in context?*

First, we conduct a systematic literature review (SLR) to identify a comprehensive list of factors known to influence pull request decisions. Then, we create a large and diverse dataset of pull requests and factors (or their indicators) that can be mined from archival software data. Finally, we build models (mixed effects logistic regression models) that suggest the relations between each factor and pull request decisions in general, specific scenarios (*e.g.*, when pull requests use CI), and different contexts (*e.g.*, the time when pull requests are closed).

This paper makes the following contributions to software engineering research and practice:

1) We present a curated dataset of 11,230 projects on GitHub with 95 factors and 3,347,937 pull requests. Our dataset is diverse in terms of the number of contributors, programming language, and activities (see Table 1). It also covers the entire project lifecycle as a representation of diversity in time. Future researchers can use and extend our large and rich dataset[1] to conduct deeper investigations and use scripts to replicate the results.[2]
2) We present a synthesis of the factors identified in the literature, indicating their significance and direction.
3) We show the importance of these factors in explaining pull request decisions and how these decisions change with a change in context.

The rest of the paper is organized as follows. In Section 2, we explain our research design. In Section 3, we present the results. In Section 4, we conduct a case study about affiliation-related factors. We discuss the implications in Section 5 and present the threats in Section 6. In Section 7, we describe the related work of this study. In Section 8, we present our conclusions and directions for future work.

2 Study Design

The framework of our study is shown in Figure 1, which mainly comprises four parts presenting the steps to empirically explain pull request decisions. First, we gather a comprehensive list of the factors known to influence pull request decisions (see the SLR part in Figure 1). Next, we collect data from diverse collaboratively developed software projects on GitHub to use as proxies for the factors identified above (see the Data Collection part in Figure 1). Then, we transform the data and transfer them into a form usable for analysis (see the Data Preprocessing part in Figure 1). Finally, we model the data to answer our research questions, starting with an exploratory data analysis (see the Statistical Modeling part in Figure 1).

2.1 Systematic literature review

To collect all factors known to influence pull request decisions, we conducted a systematic literature review (see the SLR part in Figure 1(a)), which was based on the guidelines from Kitchenham et al. [17].

1. https://zenodo.org/record/4837134#.YLEWyY3isdW
2. https://github.com/zhangxunhui/TSE_pull-based-development

Our search strategy was to identify all scientific articles relating to pull request decisions. We selected two widely used search terms, "pull request" and "pull based", which are often used interchangeably as pull request models, pull-based development, and similar variants. We combined the two search terms with a logical "OR" operator (*i.e.*, "pull request" OR "pull based") defining our search space. We searched for ("pull request" OR "pull based") on Google Scholar, ACM Digital Library, IEEEXplore, Web of Science and Ei Compendex, resulting in a total of 3,941 papers. We ran the query on April 17th, 2020. We identified 1,000 papers from Google Scholar, 1,433 from ACM Digital Library, 352 from IEEEXplore, 487 from Web of Science, and 669 papers from Ei Compendex. We performed an additional step of searching Google Scholar for papers published only in 2020. (Here, we only consider 2020 because we can get all relevant papers through the backward snowballing process [17]. Therefore, we don't have to perform searches for each year.) This step was necessary since Google Scholar retrieves only the top 1,000 results, which means that it is likely to miss many articles [18], [19]. The additional search (also conducted on April 17th, 2020) resulted in 610 more papers, leading to a total of 4,551 papers for backward snowballing.

To identify the factors influencing pull request decisions, the first author manually analyzed the title and abstract of each paper and selected all studies presenting all the factors influencing pull request decisions that can be inferred by mining software archives. The search resulted in 19 papers after excluding papers for the following reasons:

- they were written in languages other than English (45 papers)
- they were duplicates (1,181 papers)
- they were initial versions of the papers when extended versions were available (12 papers)
- they presented factors not applicable to GitHub (5 papers); *e.g., a study on Firefox and Mozilla core projects shows that "bug severity" and "bug priority" influence patch acceptance [20]. These attributes do not exist on GitHub*
- they were not related to pull request decisions (3,277 papers)
- they were related to pull request decisions but difficult to reproduce (4 papers), *e.g., using medical equipment to track the eyes of reviewers [21]*
- they included factors not generalizable to a wider range of software projects on GitHub (4 papers), *e.g., labels [22] that vary across communities*
- they presented different operationalizations of related concepts (3 papers); *e.g., emotions can be measured directly as joy, love, sadness, and anger; indirectly via valence, arousal, and dominance [23]; and abstractly based on polarity [24]. We choose one of three representations of emotions, i.e., polarity. As another example, Calefato et al. [25] measured trust using agreeableness, one of the five personality traits used by Iyer et al. [2]. Thus, we chose five personality traits*
- they presented factors not measurable quantitatively (1 paper), *i.e., the features relating to pull request decisions found in a qualitative study [26]*

Next, we identified other relevant articles by considering the references of the 19 selected seed articles. We applied

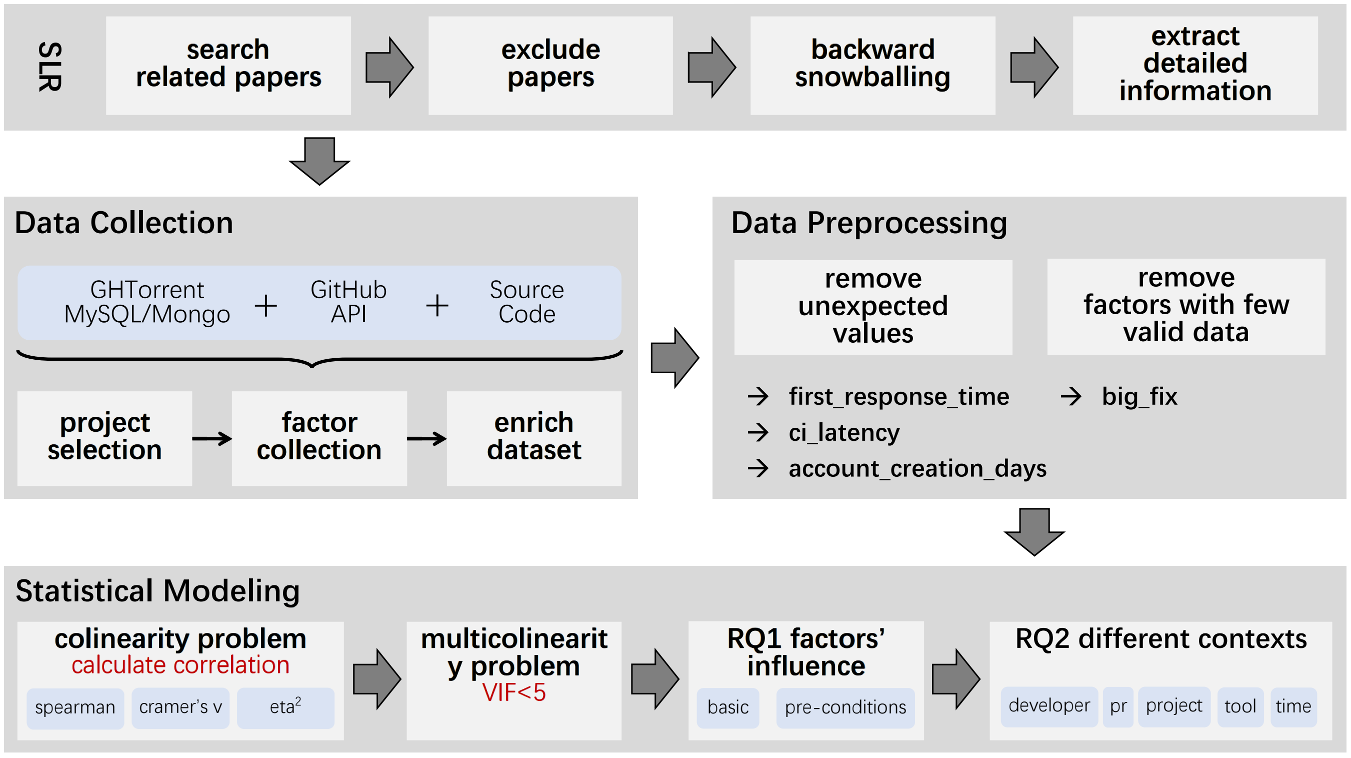

Fig. 1: Framework of this paper

the backward snowballing method [17] twice, meaning that we selected (a) the references of the 19 articles and (b) the references of the references. After two rounds, we did not find any new related papers. This process resulted in 7 new papers, bringing the total to 26 papers presenting the factors related to pull request decisions.

An overview of the 94 features (the factor *same_user* was not considered in previous studies) found in the systematic literature review is shown in Table 2, which lists the symbolic representations of the features in columns 1 and 3, followed by their descriptions in columns 2 and 4, respectively. All the features are classified as developer, project, and pull request characteristics. Furthermore, Table 10 shows the relations between each of the factors and pull request decisions, as identified in the 26 selected research articles.

For the accuracy and validity of the data extraction process, the first and the last author did the whole process together. First, in the paper screening phase, the first author got the initial results. Then the first author and the last author met to discuss the paper with uncertainty and finally reached an agreement. *E.g.*, the paper [26] was a relevant study on pull request decisions, but as a qualitative study, it lacked a measure of certainty about the relevant factors, so we removed the paper. After that, in the factor extraction stage, the first author extracted the initial factors, including the name of the factor, the related description, the category to which it belongs (pull request, project, or developer), and the description of related findings, forming a list. The first and the last author then met to discuss and agree on the information in the list, which consisted of the following steps.

1) For relevant factors with unclear descriptions, reach an agreement, *e.g.*, factor *pushed_delta* (see Table 2).
2) Remove factors that are not applicable for GitHub, *e.g.*, bug severity.
3) Remove factors that are difficult to reproduce, *e.g.*, eye tracking of reviewers.
4) Confirm the category to which factors belong.
5) The last author maintained a list of relevant factors in advance based on the research experience and checked during the meeting to see if they all appeared in the list provided by the first author.

After the above process, we finally identified the 94 relevant factors.

2.2 Data collection

We collected data on a variety of software projects hosted on GitHub as a proxy for the factors identified above. The dataset used for this study came from our prior work [27], featuring 96 factors collected from 11,230 projects. Furthermore, we enriched the dataset with missing factors and values (see the Data Collection part in Figure 1).

Our initial dataset [27] was built on the publicly available GHTorrent MySQL data dump dated June 1st, 2019.[3] It features 96 factors relating to pull requests, developers, or projects (derived from 76 research articles published between 2009 and 2019) for 11,230 software projects. The screening steps of GitHub projects are summarized as follows:

1) Filter forked or deleted repositories based on GHTorrent.[3]

3. http://ghtorrent-downloads.ewi.tudelft.nl/mysql/mysql-2019-06-01.tar.gz

2) Filter repositories that do not have any pull requests in the last three months.
3) Select projects from six programming languages (as against 4 programming languages in the case of Gousios et al.'s [28] dataset). The extended JavaScript and Go languages are the most popular programming languages on Github [4] and the fastest growing programming languages in recent years, respectively. [5]
4) Select all projects with at least 33 submitted pull requests. These projects constitute the top 3% of all projects in terms of pull request count (as against the top 1% in the case of Gousios et al.'s [28] dataset). The top 3% here is chosen to make some extensions based on Gousios et al.' dataset [28]. With the development of Github, a large number of open source projects have emerged. In addition to the most active open source projects, we also want to include a wide range of projects, including small and relatively less active projects. After discussion, we have chosen the top 3% of projects.
5) Split projects according to the tertile thresholds of the number of developers in the project, *i.e.,* small-sized teams (low tertile) with 12 or fewer developers, medium-sized teams (middle tertile) with 13 and up to 30 developers, large-sized teams (high tertile) with more than 30 developers. Randomly select 4,000 projects in each class.
6) Remove the data holding project "everypolitician/everypolitician-data", which is extremely large, and we lack the ability to collect related factors.
7) After discussion among authors, remove projects with less than 20 closed pull requests related to their default branch to ensure enough data for the subsequent steps required in research.

After the above steps, 11,230 projects remained, which offers a total of 3,347,937 closed pull requests (meaning a decision has been made) submitted to the repository's default branch.

TABLE 1: Description of project diversity

category	type	project count	percentage
language	JavaScript	3,879	34.5%
	Python	3,055	27.2%
	Java	1,823	16.2%
	Ruby	1,243	11.1%
	Go	913	8.1%
	Scala	317	2.8%
project size	small ≤ 12 *developers*	3,711	33%
	mid ≤ 31 *developers*	3,634	32.4%
	large > 31 *developers*	3,885	34.6%
project activity	min $= 33$ *pull requests*	-	-
	25% ≤ 55 *pull requests*	2,843	25.3%
	50% ≤ 106 *pull requests*	2,796	24.9%
	75% ≤ 261 *pull requests*	2,791	24.9%
	max $= 38,953$ *pull requests*	-	-

Our initial dataset is futuristic and emphasizes generalizability - a design choice for a wide range of explorations [27]. Moreover, our dataset has 12 times more projects and 10 times more pull requests than Gousios et al.'s [28] dataset and is more diverse than any of the datasets of prior studies focusing on pull request decisions, which have, until now, largely focused on the most popular projects.

From Table 1, we can see that the diversity of selected projects is mainly manifested in three aspects, *i.e.,* covering 6 languages, containing different numbers of contributors, and including projects with different activity levels (the number of pull requests ranges from 33 to more than 30 thousand). Our dataset has features that are applicable to projects outside GitHub and has additional features that are likely to influence pull request development - an extrapolation of existing features.

For our analysis, we selected data related to the factors identified by our systematic literature review from the initial dataset. We noticed that 14 factors identified by our systematic literature review did not exist in the initial dataset, so we added these missing features. Table 2 presents a complete list of the factors known to influence pull request decisions on GitHub. Factors marked as $\star$ are additions to those of the initial dataset [27].

Finally, we enriched our dataset by filling in missing values wherever possible based on GHTorrent[6], GitHub API and source code of repository. For example, the initial dataset used the tool by Vasilescu et al. [29] to infer country information. The resulting dataset, however, had a large number of missing values. We applied several steps, such as using *country_code* information and *pycountry* package[7] to extract country names. In this way, we were able to derive the country information of an additional 546,682 contributors (1,473,008 previously), 747,204 integrators (1,580,256 previously) and 796,083 same-country participants (1,081,668 previously). The expanded country information can be seen on GitHub.[8] To verify the validity of the data, we randomly selected 100 developers with predicted country information. Then, the first author manually checks the accuracy according to the developer's GitHub homepage and the given external site. Only two developers made a mistake in their predictions, and another two developers' country information could not be judged based on the existing knowledge. Therefore, the precision of the extracted country information $\approx 96\%$.

We added a factor, *same_user*, that did not exist in prior studies (marked as ● in Table 2). While the information on the same user is not useful itself, it adds meaning to factors such as *same_country*, *same_affiliation*, and personality-difference-related factors (*e.g., open_diff*), which make sense only when the contributor and integrator are not the same users. In our dataset, we found that 43.6% of the pull requests were integrated by submitters (85.7% of them were core contributors, and 14.3% were external contributors). Compared to directly committing to code repositories, pull-based development is becoming a standard collaborative model in which not only external contributors but also core

4. https://octoverse.github.com/#top-languages-over-the-years
5. https://hub.packtpub.com/why-golan-is-the-fastest-growing-language-on-github/
6. https://ghtorrent.org/
7. https://pypi.org/project/pycountry/
8. https://github.com/zhangxunhui/TSE_pull-based-development/blob/master/country_info.csv

TABLE 2: Comprehensive list of the factors known to influence pull request decisions on GitHub

Factor	Description	Factor	Description
	Developer Characteristics		
first_pr	first pull request? yes/no	prior_review_num	# of previous reviews in a project
core_member	core member? yes/no	first_response_time	# of minutes from pull request creation to the reviewer's first response
contrib_gender	gender? male or female	contrib_country	country of residence
same_country	same country contributor/integrator? yes/no	prior_interaction	# of interactions with a project in the last three months
same_affiliation	same affiliation contributor/integrator? yes/no	contrib/inte_affiliation	contributor/integrator affiliation
contrib/inte_**X**	contributor/integrator personality traits (*open*: openness; *cons*: conscientious; *extra*: extraversion; *agree*: agreeableness; *neur*: neuroticism)	perc_contrib/inte_X_emo	% of contributor/integrator (*neg*: negative/*pos*: positive) emotion in comments
X_diff	absolute difference in the personality traits of the contributor and the integrator	contrib/inte_first_emo	emotion in contributor/integrator's first comment
social_strength	fraction of team members interacted with in the last three months	contrib_follow_integrator	contributor followed integrator before pull request creation? yes/no
followers	# of followers at pull request creation time	same_user ●	same contributor and integrator? yes/no
prev_pullreqs	# of previous pull requests	account_creation_days	# of days from the contributor's account creation to pull request creation
contrib_perc_commit ⋆	% of the contributor's previous commit	requester_succ_rate	past pull request success rate
	Project Characteristics		
sloc	executable lines of code	team_size	# of active core team members in the last three months
language	programming language	open_issue_num	# of open issues
project_age	# of months from project to pull request creation	open_pr_num	# of open pull requests
pushed_delta	# of seconds between two latest pull requests open	fork_num	# of forks
pr_succ_rate	pull request acceptance rate of project	test_lines_per_kloc	# of test lines per 1K lines of code
stars	# of stars	integrator_availability ⋆	latest activity of the two most active integrators
test_cases_per_kloc	# of test cases per 1K lines of code	asserts_per_kloc	# of assertions per 1K lines of code
perc_external_contribs	% of external pull request contributions		
	Pull Request Characteristics		
churn_addition	# of added lines of code	churn_deletion	# of deleted lines of code
bug_fix	fixes a bug? yes/no	description_length	length of pull request description
test_inclusion	test case existing? yes/no	comment_conflict	keyword "conflict" exists in comments? yes/no
hash_tag	"#" tag exists? yes/no	num_participants	# of participants in pull request comments
lifetime_minutes	# of minutes from pull request creation to latest close time	part_num_code	# of participants in pull request and commit comments
ci_exists	uses CI? yes/no	ci_build_num	# of CI builds
ci_latency	# of minutes from pull request creation to the first CI build finish time	perc_neg_emotion	% of negative emotion in comments
num_code_comments ⋆	# of code comments	perc_pos_emotion	% of positive emotion in comments
test_churn	# of lines of test code changed (added + deleted)	num_code_comments_con ⋆	# of contributor's code comments
ci_test_passed	all CI builds passed? yes/no	ci_first_build_status	CI first build result
ci_failed_perc	% of CI builds failed	ci_last_build_status	CI last build status
num_commits	# of commits	src_churn	# of lines changed (added + deleted)
files_added	# of files added	files_deleted	# of files deleted
files_changed	# of files touched	Friday_effect ⋆	pull request submitted on a Friday? yes/no
reopen_or_not ⋆	pull request is reopened? yes/no	commits_on_files_touched	# of commits on files touched
has_comments ⋆	pull request has a comment? yes/no	num_comments	# of comments
has_participants ⋆	has a participant? yes/no	core_comment ⋆	has a core member comment? yes/no
contrib_comment ⋆	has a contributor comment? yes/no	inte_comment ⋆	has an integrator comment? yes/no
has_exchange ⋆	has contributor and integrator comments? yes/no	other_comment ⋆	has noncontributor/core team comment? yes/no
num_comments_con ⋆	# of contributor comments	at_tag	"@" tag exists? yes/no

NOTE: Factors marked as ⋆ are additions of our study to the latest MSR Data Showcase pull request dataset [27], while ● are additions to previous studies.
All metrics are relative to a referenced pull request in a project.
Factors that change over time (e.g., core team) are measured using the previous three months of development activities in a project.
The related paper information and the nature of each factor can be seen in Table 10.

members are interested. Therefore, it is necessary to add this factor and study its influence on pull request decisions.

For factor *bug_fix*, we followed Fan et al.'s [30] method in finding the tag for determining whether the pull request is a bug fix or not. In their method, they manually found the most used tags for bug-prone and non-bug-prone issues. (The tags are listed in Table 3.) Therefore, we first check whether the pull request has a tag marking its type. If not, we link the pull request to an issue [31]. If the pull request fixes an issue, we check the related issue's tag to see whether the pull request fixes a bug or not. To ensure data accuracy, we did not use a prediction model to predict the type of pull request.

TABLE 3: Bug and non-bug tags

Category	Tags
Bug	"bug"; "defect"; "type:bug"
Non-bug	"enhancement"; "feature"; "question"; "feature request"; "documentation"; "improvement"; "docs"

2.3 Data preprocessing

Our exploration of the resulting dataset (manually and using data distribution graphs) showed some unexpected data values for factors such as *first_response_time*, *ci_latency*, *account_creation_days* and *project_age*. It is important to fix them for reliable inferences (see the Technical Report [32] for examples).

- *first_response_time* has *negative* values for some pull requests. One possible reason is that our metric considers the discussion under a pull request and the comments under the related code. Since some comments exist before pull request creation, our data show negative values. We fix this issue by excluding pull requests with negative values (0.4%).
- *ci_latency* has *negative* values for some pull requests. CI latency measures the time from pull request creation to CI build finish time. In some cases, however, commits exist prior to pull request creation, and the time of first build recorded is earlier than the creation time of a pull request. We fix this problem by removing such pull requests (1.5%).
- *account_creation_days* and *project_age* have *negative* values, which happens in special cases where the creation time of a user account on GHTorrent is different from that on Github API. Here too, we remove such cases (0.1%).
- *bug_fix* has 99.3% empty values. We remove this factor, which otherwise can adversely affect the analysis.

For the time-related factors, we verified the accuracy of the remaining data by randomly selecting 100 records. We found that the inconsistency between the GHTorrent MySQL version and GitHub API resulted in the accuracy of *first_response_time*, *account_creation_days*, *project_age*, and *ci_latency* at about 98%, 97%, 96%, and 94%, respectively. We have added this part to the Threats to Validity section.

2.4 Statistical modeling

Presenting a comprehensive analysis of the factors influencing pull request decisions, we build generic models comprising all the factors and models representing specific cases. We also build models within different contexts. However, first, we explore relationships among the factors identified above.

Our preliminary exploration into the relationship among factors started with calculating the correlations among all the factors. We calculated the Spearman correlation coefficient (ρ) for continuous factors [1], Cramér's V (Φ_c) for categorical factors [33], and partial Eta-squared (η^2) for the correlation between continuous and categorical factors [34]. We consider $\rho > 0.7$ [1], $\Phi_c > \frac{0.5}{df}$ [35] and $\eta^2 > 0.14$ [35] as strong correlations.

A list of strongly correlated factors is presented in Table 4, in which the strongly correlated factors are separated from the other factors by a dotted line. For a complete list of correlations between each pair of factors, refer to our technical report [32].

Next, we built mixed effects logistic regression models to empirically explain the factors influencing pull request decisions. The models used the project identifier as a random effect, indicating similarity among the pull requests of a project [36]. All other factors had fixed effects. The resulting model indicated the significance of a factor and direction of its association with a pull request decision (accept or reject). We used the *glmer* function of the *lme4* [37] package in R to model pull request decisions.

To build an explanatory model, we included all factors that could be meaningfully added together, did not present

TABLE 4: Choices and corresponding reasons for strongly correlated factors

Correlated factors	Selected factor	Reason
test_lines_per_kloc test_cases_per_kloc asserts_per_kloc	*test_lines_per_kloc*	previous study
src_churn churn_addition churn_deletion	*src_churn*	frequency
num_comments at_tag num_participants num_comments_con	*num_comments*	frequency
core_member perc_external_contribs social_strength requester_succ_rate	*core_member*	frequency
stars fork_num inte_affiliation	*stars*	frequency
prev_pullreqs prior_interaction	*prev_pullreqs*	frequency
num_code_comments part_num_code num_code_comments_con	*num_code_comments*	frequency
open_pr_num fork_num	*open_pr_num*	frequency
ci_latency ci_build_num	*ci_latency*	promising performance
sloc language	*sloc*	promising performance
has_comments has_participants core_comment contrib_comment inte_comment has_exchange	*has_comments*	expressiveness
prior_review_num inte_affiliation	*prior_review_num*	data availability
open_issue_num inte_affiliation	*open_issue_num*	data availability
inte_cons inte_affiliation	*inte_cons*	data availability
inte_extra inte_affiliation	*inte_extra*	data availability
inte_agree inte_affiliation	*inte_agree*	data availability
same_country contrib_country	*same_country*	discussion
perc_contrib_pos_emo contrib_first_emo	*perc_contrib_pos_emo*	discussion
perc_inte_neg_emo inte_first_emo	*perc_inte_neg_emo*	discussion
perc_inte_pos_emo inte_first_emo	*perc_inte_pos_emo*	discussion
same_user inte_first_emo inte_affiliation contrib_affiliation contrib_rate_author	*same_user*	discussion
same_affiliation contrib_affiliation	*same_affiliation*	discussion
perc_neg_emotion perc_contrib_neg_emo contrib_first_emo inte_first_emo	*perc_neg_emotion*	discussion
perc_pos_emotion inte_first_emo	*perc_pos_emotion*	discussion
ci_failed_perc ci_test_passed ci_first_build_status ci_last_build_status	*ci_failed_perc*	discussion

the same or similar information as other factors, and were easy to interpret.

1) *Adding meaningful factors*. While adding factors to a model, we observed that 17 factors (postconditional factors in Table 5) did not make sense outside a specific context. For example, if the contributor and integrator were the same, then factors such as "personality difference" did not exist and made no sense. We refer to such factors as *"preconditional factors"* and *"postconditional factors"*. *"Preconditional factors"* are those that must exist for another factor to exist and make sense (*e.g., same_user* in the previous example). Conversely, *"postconditional factors"* are the factors in which their existence is conditional on preconditional factors (*e.g., open_diff*). All the other factors are classified into the "others" category. A complete list of pre- and postconditional factors is presented in Table 5.
2) *Factors presenting the same information*. Our preliminary investigation showed that several factors identified from the literature were strongly correlated with each other (see Table 4 for a list of strongly correlated factors). When two related factors were added to a model, they changed not only pull request decisions but also other factors, which could change the estimated effect of these factors on pull request decisions and their significance, also referred to as a multicollinearity problem [38]. To avoid multicollinearity, we selected one of the many strongly correlated factors. Our choice of the selection of a factor was influenced by its use in previous studies (*e.g.,* [1] chose *test_lines_per_kloc*), frequency of occurrence in the literature (*e.g., core_member* appeared most often), promising performance (indicating the likelihood of strong correlation with pull request decisions) (*e.g., sloc* significantly influences pull request decisions [12], while *language* does not have such a conclusion according to previous studies), expressiveness (*e.g., has_comments* is broader and more informative than *contrib_comment*), data availability (*e.g., open_issue_num* has most nonempty values), and otherwise in discussion with the last author (*e.g., perc_pos_emotion* is more representative for the whole review process than *inte_first_emo*; *same_country* takes the country relationship between the contributor and the integrator into consideration; *same_user* is the precondition for eight factors (see Table 5)). We also excluded factors with variance inflation factor (VIF) values ≥ 5, as such values could inflate variance, measured using the *vif* function of the *car* package in R [39]. In this way, we removed *num_code_comments* that were otherwise moderately correlated with *num_comments* ($\rho = 0.63$).
3) *Ease of interpretation*. Models perform better when features are approximately normal and in a comparable scale.[9] We stabilized the variance in features by adding a value "1" and log-transforming the continuous variables. Then, we transformed the features into a comparable scale with a mean value of "0" and a standard deviation of "1".

9. https://medium.com/@sjacks/feature-transformation-21282d1a3215

TABLE 5: Factors with dependency

postconditional factor	preconditional factor
perc_pos_emotion *perc_neg_emotion* *first_response_time*	*has_comments*
perc_contrib_pos_emo *perc_contrib_neg_emo*	*contrib_comment*
perc_inte_neg_emo *perc_inte_pos_emo*	*inte_comment*
ci_latency *ci_failed_perc*	*ci_exists*
same_country *same_affiliation* *contrib_follow_integrator* *open_diff* *cons_diff* *extra_diff* *agree_diff* *neur_diff*	*same_user*

2.4.1 Factors influencing pull request decisions

To explain pull request decisions, we intended to build a model with all the known factors. However, in practice, this is not possible. We noticed that the postconditional factors (see Table 5) did not make sense unless a precondition was met. For example, the factor *ci_latency* was meaningful only when the factor *ci_exists* was true. Here, *ci_exists* presents a precondition contingent on which factors, such as *ci_latency*, are meaningful, which are also referred to as postconditional factors. Table 5 presents a complete list of the dependent factors in our dataset. The remaining factors have no such dependency on other factors.

To understand how the identified factors influence pull request decisions, we built two types of models.

1) We built a *basic model* that comprised all the factors with no dependencies on each other and preconditional factors. This model offered an overview without entering the details offered by the postconditional factors.
2) Next, we built models for the *special cases* relating to preconditions: developer, pull request, and tools as identified in Table 5.
   - *developer:* when the contributor and the integrator are not the same users (*same_user=0*)
   - *pull request:* when a pull request has comments (*has_comment=1*)
   - *tool:* when a pull request uses the CI tool (*ci_exists=1*).

   Each of these special case models are built on a subset of the data used in the basic model that meets the precondition.

2.4.2 Influence of context

To explore the relevance of context in explaining pull request decisions, we studied five scenarios relating to the developer, pull request, project, tools, and time. Figure 2 presents a pictorial depiction of the five scenarios in relation to the pull request decision and metrics. To study the influence of context, we trained the same model on different observations representing specific contexts.

- *developer characteristic:* We chose the factor *same_user* indicating whether a pull request is submitted and integrated by the same user. It is the most important developer characteristic influencing pull request decisions

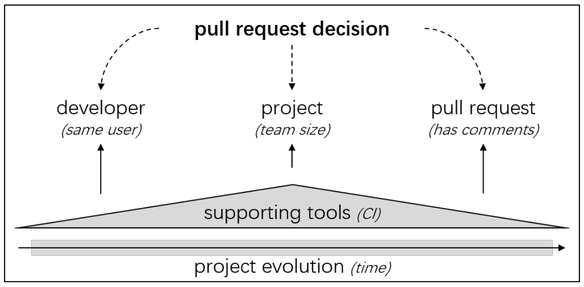

Fig. 2: Contexts in pull request decisions

(see the basic model in Table 6) and a precondition for a range of factors. We think that pull requests integrated by oneself behave differently than those integrated by others.

- *pull request characteristic:* We chose the factor *has_comments* as an indicator of a pull request characteristic influencing the decision [15]. It is one of the top five factors influencing decisions (see the basic model in Table 6) and a precondition for several factors, including *perc_pos_emotion* and *first_response_time* (see Table 5). This factor explores decisions for pull requests both with and without comments.
- *project characteristic:* We selected the factor *team_size* as an indicator of project characteristics such as project popularity and maturity. We assumed that teams of different sizes represented different contexts (as also seen in other studies [40], [41]). We studied three team sizes: small ($team\_size \leq 4$), medium ($4 < team\_size \leq 10$), and large ($team\_size > 10$). Here we split the pull request according to the tertile of factor *team_size*. [10]
- *supporting tools:* We selected the factor *ci_exists* for its reported influence on pull request decisions [10] and relevance in our special case model (refer to Table 6). In addition, a previous study has shown that the usage of CI tools changes during the development of projects [43]. Therefore, we assumed that factors influence pull request decisions differently depending on whether they are pull requests using CI tools or those not using CI tools.
- *project evolution:* We studied temporal evolution to see if the process changed over time. We studied decision-making in three time periods: before June 1st, 2016, between June 1st, 2016, and June 1st, 2018, and between June 1st, 2018, and June 1st, 2019 (aka after June 1st, 2018). A pull request belonged to a time period when it was integrated. For this scenario, we included only projects (and their pull requests) active in all three time periods. [11]

2.4.3 Interpretation of statistical models

The resulting mixed effects logistic regression models explain the influence of factors in models and their relative relevance. Section 3 presents the findings from these mixed effects logistic regression models. Each model has two parts: an intercept and influence of a factor, expressed as follows:

$$odds\ ratio^{p-value}[percentage\ variance] \quad (1)$$

The *odds ratio* expresses the association between a factor and a pull request decision as "the increase or decrease in the odds of acceptance for a 'unit' increase of a factor" [15]. In this work, a "unit" of each factor was one standard deviation from the standardization of the log-transformed factors. The term *p value* indicates the statistical significance of a factor, which was indicated by asterisks: *** p<0.001; ** p<0.01; * p<0.05 [10], [12]. It represents the probability of the evidence against the null hypothesis, *i.e.,* "there is no association between each factor and pull request decisions." Finally, the *percentage of explained variance* was used as a proxy for the relative importance of a factor. The variance explained by each factor is derived from ANOVA Type-II analysis [44]. When it is relative to the total amount of variance (the percentage of explained variance), the result can serve as a proxy for effect size, which means how much effect one factor has in explaining pull request decisions. This metric is similar to the percentage of total variance explained by least squares regression [39] and has been used in prior studies [45].

We reported the *goodness of fit* of each model using the area under the receiver operating characteristic curve (*AUC*) value (for training data), where an *AUC* value greater than 0.5 indicated the effectiveness of the model [12]. We also reported the predictive performance of related models using the weighted precision, weighted recall, and weighted f-score [46].

In practice, we split the pull requests in close time and used the first 90% of pull requests for training and the remaining 10% for testing. We measured the predictive performance of the basic model only to present the prediction effect of pull request decisions by integrating as many factors as possible and to explain factor performance in other situations, without reporting their prediction performance. The above metrics collectively indicated the predictive performance of both the baseline and logistic regression models for our highly imbalanced dataset [46].

3 Results

This section presents how factors influence pull request decisions (answering **RQ1**) via a basic model, which comprises all the factors likely to influence pull request decisions, excluding those that cannot make it to the basic model. Next, we describe how the factors influencing pull request decisions change with a change in context (answering **RQ2**). We present five scenarios representing developer, pull request, project, tool, and time characteristics.

3.1 RQ1: How do factors influence pull request decisions?

3.1.1 Basic model

Our basic model in Table 6 (column 3) shows 46 factors known to influence pull request decisions arranged in non-increasing order of relative relevance. In comparison to a

10. A sensitivity analysis with threshold values (small size ranging from 2-6, large size ranging from 8-12) yielded similar results. See the technical report [42] for the detailed results.

11. A sensitivity analysis with threshold values (first period ranging from December 1st, 2015 to December 1st, 2016, third period ranging from December 1st, 2017 to December 1st, 2018) yielded similar results. See the technical report [42] for the detailed results.

random classifier (with weighted precision: 0.81, weighted recall: 0.79, weighted f-score: 0.80, and AUC_test: 0.50), our basic model performed better (with weighted precision: 0.89, weighted recall: 0.90, weighted f-score: 0.89, and AUC_test: 0.82), suggesting an improvement in our model in terms of decision making.

The five most important factors influencing pull request decisions are *same_user*, *lifetime_minutes*, *prior_review_num*, *has_comments* and *core_member*. Table 6 (column 3) shows that these top five factors (shown in dark gray) explain approximately 83% of the variance. This number reaches approximately 95% when considering the influence of the top 10 factors. The remaining 36 factors collectively explain 5% of the explained variance.

The most important factor influencing pull request decisions was *same_user* (with 31% variance). Moreover, *same_user* decreased the odds of acceptance of a pull request by 48% per unit when a pull request was integrated by the contributor. One possible explanation for this observation relates to the process of pull-based development. Due to the standardized process of such development, contributions should be reviewed and merged by others during the process. However, since all contributors could close their own pull requests, it was possible for them to find problems in their pull requests from others' comments or CI build results and close their own pull requests.

Through Table 8, we can see that many project related factors, including *open_pr_num*, *project_age*, *sloc*, were found to influence pull request decisions in related works significantly. However, through integrating various factors, we find that the project related factors did not contribute greatly to pull request decisions, as these factors explained only approximately 1% of the variance. However, the developer- and pull-request-related factors are more important, explaining 52% and 46% of the variance, respectively. See the dynamic treemap to compare the relative importance of factors in different categories visually. [12]

3.1.2 Special cases

Table 6 shows the results of the three special cases in the last three columns. Factors ranking the top 5 in each model (T_{1-5}) are shown in deep gray, and factors ranking in the top 6-10 in each model (T_{6-10}) are shown in light gray.

When the contributor and integrator were different users (*same_user=0*) (see column 4 in Table 6), we found that three additional factors had a small effect on pull request decisions. The only factor that made it into the top 10 factors was personality difference, namely, differences in agreeableness (*agree_diff*). The two other factors were differences in openness to experience (*open_diff*), also indicating differences in personality, and the same affiliation of the contributor and integrator (*same_affiliation*).

When there existed at least one comment (*has_comments=1*) (see column 5 in Table 6), positive emotion became relatively important, with a sizable effect ($> 3\%$ variance). This change can be attributed to the phenomenon that positive reactions during the code review process can lead to contributors' active participation and increase the likelihood of pull request acceptance. However, negative emotion is not important in pull request decisions. A possible explanation for this is that different developers tend to act differently toward negative emotion. Therefore, negative emotion during discussion faces difficulty in effectively making the final decision. To verify our observation, we built models for pull requests that had at least one comment from a contributor (*contrib_comment=1*) or at least one comment from an integrator (*inte_comment=1*) [42]. We found that both *perc_contrib_pos_emo* and *perc_inte_pos_emo* explained more than 3% of the variance, which was much higher than that of negative emotion.

When pull requests used CI tools (*ci_exists=1*) (see column 6 in Table 6), factor *ci_failed_perc* stood out, explaining 18% of the variance, which implies that the build status of CI tools is important for review decisions, especially the percentage of build failures.

> Pull request decisions is mostly explained by a few factors (5 to 10 factors) such that developer and pull request characteristics are more important than project characteristics.
> The relation between contributor and integrator (*same_user*) is the most important factor influencing pull request decisions.
> In special cases, when a pull request has comments, comment's positive emotion is linked to pull request acceptance. Likewise, when pull requests use CI tools, the percentage of failed CI builds become important for pull request decisions.

3.2 RQ2: How do the factors influencing pull request decisions change with a change in context?

3.2.1 Developer characteristic

Table 7 shows that in comparison to the pull requests submitted and integrated by the same user, when the contributor and integrator are not the same person, the variance explained by the experience of the integrator (*prior_review_num*) decreases from 31% (row 1, column 2 - same user: yes) to 0% (row 1, column 3 - same user: no). This finding implies that the integrator's experience plays a limited role when making decisions regarding others' contributions. However, this factor becomes very important for an integrator's own contributions. One way to explain this observation can be that external contributors, without review experience, generally do not have the right to merge the code. Experienced integrators, in contrast, are familiar with the management process, know when to merge a pull request, and have the ability to merge a pull request. In this way, differences in permission linked to integrators' experience can influence pull request decisions.

For the lifetime of pull requests (*lifetime_minutes*), the percentage of explained variance increased from 19% (row 2, column 2 - same user: yes) to 44% (row 2, column 3 - same user: no). A possible explanation for this observation is that when there is no response from a contributor for a long time, a pull request is more likely to be closed by the reviewer. However, when the pull request is reviewed by

12. https://github.com/zhangxunhui/TSE_pull-based-development/blob/main/treemap-basic-model.html

TABLE 6: Results of special cases. - means the factor is not included in the model. Color: deep gray represents factors with explained variance rank in the Top 5 and light gray represents factors rank in the Top 6-10.

Factor Index		*Dependent variable: merged_or_not=1*			
		basic model	*same_user=0*	*has_comments=1*	*ci_exists=1*
	(Intercept)	21.1^{***}	32.5^{***}	15.4^{***}	26.1^{***}
(1)	same_user	0.52^{***} [31.17]	-	0.60^{***} [21.53]	0.50^{***} [21.27]
(2)	lifetime_minutes	0.61^{***} [21.10]	0.52^{***} [43.08]	0.53^{***} [30.40]	0.50^{***} [26.08]
(3)	prior_review_num	1.53^{***} [13.53]	1.06^{**} [0.20]	1.50^{***} [13.94]	1.50^{***} [8.01]
(4)	has_comments	0.63^{***} [11.97]	0.52^{***} [25.39]	-	0.64^{***} [6.70]
(5)	core_member	1.29^{***} [5.29]	1.02 [0.05]	1.30^{***} [5.86]	1.32^{***} [3.58]
(6)	num_commits	1.30^{***} [4.49]	1.35^{***} [6.67]	1.58^{***} [13.65]	1.56^{***} [7.43]
(7)	other_comment	1.21^{***} [3.76]	1.27^{***} [6.47]	1.12^{***} [1.58]	1.24^{***} [2.88]
(8)	ci_exists	1.16^{***} [1.47]	1.25^{***} [5.13]	1.11^{***} [0.93]	-
(9)	hash_tag	1.12^{***} [1.36]	1.06^{***} [0.51]	1.10^{***} [1.15]	1.13^{***} [1.04]
(10)	files_added	0.91^{***} [0.74]	0.96^{**} [0.18]	0.91^{***} [0.61]	0.90^{***} [0.48]
(11)	prev_pullreqs	1.15^{***} [0.73]	1.10^{***} [0.51]	1.16^{***} [0.99]	1.15^{***} [0.43]
(12)	commits_on_files_touched	1.09^{***} [0.59]	1.13^{***} [1.51]	1.05^{***} [0.22]	1.03^{***} [0.04]
(13)	open_pr_num	0.82^{***} [0.47]	1.16^{***} [0.26]	0.94^{***} [0.05]	0.87^{***} [0.15]
(14)	account_creation_days	1.06^{***} [0.41]	1.16^{***} [2.52]	1.06^{***} [0.41]	1.09^{***} [0.53]
(15)	first_pr	0.95^{***} [0.36]	0.99 [0.01]	0.96^{***} [0.27]	0.96^{***} [0.16]
(16)	test_churn	1.07^{***} [0.27]	1.10^{***} [0.59]	1.11^{***} [0.59]	1.12^{***} [0.42]
(17)	files_changed	0.92^{***} [0.26]	0.91^{***} [0.42]	0.94^{***} [0.14]	0.97^{***} [0.02]
(18)	project_age	1.11^{***} [0.26]	1.06 [0.08]	1.08^{***} [0.19]	1.21^{***} [0.53]
(19)	reopen_or_not	0.97^{***} [0.25]	0.99 [0.05]	0.98^{***} [0.12]	0.98^{***} [0.08]
(20)	contrib_open	1.06^{***} [0.24]	1.05^{***} [0.27]	1.05^{***} [0.20]	1.07^{***} [0.20]
(21)	stars	0.86^{***} [0.22]	0.88^{***} [0.22]	0.79^{***} [0.69]	0.89^{***} [0.10]
(22)	inte_open	1.06^{***} [0.21]	1.10^{***} [0.46]	1.10^{***} [0.64]	0.98^{***} [0.01]
(23)	description_length	1.04^{***} [0.17]	1.02 [0.04]	1.01 [0.00]	1.01^{***} [0.01]
(24)	pushed_delta	1.04^{***} [0.15]	1.06^{***} [0.39]	1.04^{***} [0.22]	1.04^{***} [0.12]
(25)	followers	1.04^{***} [0.12]	0.92^{***} [0.52]	1.02^{***} [0.04]	1.03^{***} [0.03]
(26)	contrib_cons	1.03^{***} [0.07]	1.04^{**} [0.16]	1.05^{***} [0.17]	1.03^{***} [0.03]
(27)	team_size	1.06^{***} [0.06]	1.02 [0.00]	1.06^{***} [0.07]	1.07^{***} [0.06]
(28)	contrib_gender	0.98^{***} [0.05]	0.93^{***} [0.54]	0.97^{***} [0.10]	0.98^{***} [0.03]
(29)	files_deleted	0.98^{***} [0.03]	0.99 [0.02]	0.96^{***} [0.18]	0.96^{***} [0.10]
(30)	pr_succ_rate	0.98^{***} [0.03]	1.09^{***} [0.73]	0.98^{***} [0.05]	0.96^{***} [0.06]
(31)	contrib_agree	0.98^{***} [0.02]	0.99 [0.00]	0.97^{***} [0.05]	0.98^{***} [0.02]
(32)	contrib_extra	0.99^{***} [0.02]	0.94^{***} [0.29]	0.97^{***} [0.07]	0.97^{***} [0.04]
(33)	contrib_neur	1.02^{***} [0.02]	1.07^{***} [0.41]	1.01^{**} [0.01]	1.00 [0.00]
(34)	inte_neur	1.02^{***} [0.02]	1.00 [0.00]	1.04^{***} [0.08]	0.99 [0.00]
(35)	num_comments	1.02^{***} [0.02]	1.00 [0.00]	0.91^{***} [0.88]	0.97^{***} [0.04]
(36)	comment_conflict	1.01^{***} [0.01]	1.00 [0.00]	1.02^{***} [0.05]	1.01^{***} [0.01]
(37)	friday_effect	1.01^{***} [0.01]	1.01 [0.02]	1.02^{***} [0.06]	1.02^{***} [0.02]
(38)	inte_agree	1.02^{***} [0.01]	0.89^{***} [0.54]	0.98^{***} [0.02]	1.02^{*} [0.01]
(39)	inte_extra	1.01^{***} [0.01]	1.02 [0.01]	1.01^{*} [0.01]	1.06^{***} [0.10]
(40)	open_issue_num	1.03^{***} [0.01]	1.08 [0.07]	1.02 [0.00]	1.03 [0.00]
(41)	sloc	1.02^{***} [0.01]	0.97 [0.04]	1.04^{***} [0.04]	0.93^{***} [0.06]
(42)	test_inclusion	1.02^{***} [0.01]	1.00 [0.00]	1.01^{*} [0.01]	1.02^{***} [0.01]
(43)	inte_cons	1.01 [0.00]	1.04 [0.07]	1.00 [0.00]	0.99 [0.00]
(44)	integrator_availability	1.00 [0.00]	1.04^{**} [0.17]	1.01^{**} [0.01]	1.01 [0.00]
(45)	src_churn	1.00 [0.00]	1.00 [0.00]	1.05^{***} [0.17]	1.07^{***} [0.15]
(46)	test_lines_per_kloc	1.01 [0.00]	0.91^{***} [0.32]	1.02^{***} [0.02]	1.02^{*} [0.00]
(47)	agree_diff	-	0.93^{***} [0.55]	-	-
(48)	cons_diff	-	0.98 [0.03]	-	-
(49)	contrib_follow_integrator	-	1.01 [0.01]	-	-
(50)	extra_diff	-	0.99 [0.01]	-	-
(51)	neur_diff	-	0.98 [0.05]	-	-
(52)	open_diff	-	0.97^{*} [0.09]	-	-
(53)	same_affiliation	-	1.06^{***} [0.30]	-	-
(54)	same_country	-	1.02 [0.03]	-	-
(55)	perc_pos_emotion	-	-	1.18^{***} [3.14]	-
(56)	perc_neg_emotion	-	-	0.96^{***} [0.37]	-
(57)	first_response_time	-	-	1.01^{***} [0.02]	-
(58)	ci_failed_perc	-	-	-	0.65^{***} [18.28]
(59)	ci_latency	-	-	-	1.11^{***} [0.67]
	Observations	1,765,730	91,874	839,505	954,386
	AUC_train	0.848	0.891	0.850	0.865

TABLE 7: Partial results in different contexts. Whole results are shown in Appendix A.
Gray color marks the factors that have more than 5% difference of explained variance in different contexts.
Value before bracket means the odds ratio, value in bracket means the percentage of explained variance, - means the factor is not included in the model.

(1)	(2)	(3)	(4)	(5)	(6)	(7)	(8)	(9)	(10)	(11)	(12)	(13)
	Dependent variable: merged_or_not=1											
	same user or not		has comments or not		ci exists or not		different team sizes			different periods		
	yes	no	yes	no	yes	no	small	mid	large	before 2016.6	2016.6-2018.6	after 2018.6
(Intercept)	10.4	34.4	13.1	42.4	20.4	13.3	24.9	20.7	15.9	6.9	16.6	7.1
(1) prior_review_num	2.86[31]	0.98[0]	1.51[14]	1.91[22]	1.53[14]	1.53[12]	1.59[11]	1.41[9]	1.57[19]	1.30[6]	1.63[14]	1.72[17]
(2) lifetime_minutes	0.66[19]	0.52[44]	0.61[30]	0.70[13]	0.60[22]	0.61[21]	0.54[24]	0.61[20]	0.67[17]	0.65[20]	0.57[21]	0.62[13]
(3) core_member	1.26[9]	1.13[1]	1.29[6]	1.33[6]	1.30[5]	1.26[5]	1.42[6]	1.28[5]	1.19[3]	1.27[6]	1.34[5]	1.29[3]
(4) num_commits	1.23[4]	1.46[10]	1.49[11]	0.98[0]	1.32[5]	1.25[4]	1.36[5]	1.31[5]	1.26[4]	1.18[2]	1.32[4]	1.36[5]
(5) commits_on_files_touched	1.06[0]	1.11[1]	1.05[0]	1.18[2]	1.06[0]	1.13[1]	1.10[0]	1.12[1]	1.05[0]	1.30[7]	0.99[0]	0.99[0]
(6) has_comments	0.68[10]	0.50[27]	-	-	0.65[10]	0.52[25]	0.57[13]	0.64[10]	0.66[12]	0.63[15]	0.62[10]	0.55[14]
(7) same_user	-	-	0.56[29]	0.42[42]	0.51[33]	0.59[23]	0.49[24]	0.49[36]	0.55[33]	0.57[31]	0.46[33]	0.46[29]
⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮	⋮
Observations	950,985	1,010,937	1,152,714	809,208	1,611,277	350,645	601,460	703,396	701,900	512,707	585,401	274,121
AUC_train	0.862	0.874	0.837	0.872	0.843	0.884	0.877	0.843	0.837	0.850	0.867	0.879

the contributor himself/herself, he/she knows exactly what is happening, and the related decision making is thus not influenced as much by the lifetime of a pull request. Likewise, for the number of commits (*num_commits*), the percentage of explained variance increased from 4% (row 4, column 2 - same user: yes) to 10% (row 4, column 3 - same user: no). It is likely that during the interaction, the integrator will ask the contributor to modify the contribution, increase the number of commits, and then make decisions according to these changes.

When comments were present (*has_comments*), the explained variance increased when a pull request was integrated by another person in comparison to oneself from 10% (row 6, column 2 - same user: yes) to 27% (row 6, column 3 - same user: no). This result can be explained by the fact that when integrating pull requests submitted by others, it is common for the integrator to understand the contribution by communicating with the contributor.

For whether the contributor is a core developer (*core_member*), we find a notable difference in the influence of this factor on the pull request decisions in the set of self-integrated pull requests (row 3, column 2 - same user: yes) and the other-integrated pull requests (row 3, column 3 - same user: no). Although this factor is positively correlated with the pull request decisions in both cases (odds ratio>1), *i.e.*, pull requests submitted by core developers are more likely to be accepted than those submitted by external contributors; the explained variance reduces from 9% to 1%. This indicates that whether the contributor is a core developer becomes less important than other factors for pull requests integrated by others.

Whether the contributor and integrator is the same person or not influences pull request decisions the most.
If the contributor and integrator is the same, pull request decisions depend on the contributor's relationship to the target project (*prior_review_num* and *core_member*).
When the contributor and integrator are different, pull request decisions depend on the interaction between contributor and integrator (*has_comments*, *lifetime_minutes*) and the intermediate results during the process (*num_commits*).

3.2.2 Pull request characteristic

When a pull request did not have comments, the percentage of explained variance of *same_user* increased from 29% (row 7, column 4 - has comments: yes) to 42% (row 7, column 5 - has comments: no). This situation illustrates that the factor *same_user* is more associated with pull request decisions for those without comments. To investigate the reason, we calculated the merging rate of pull requests in four situations (see Table 8).

TABLE 8: Pull request merge rate for *has_comments* and *same_user* cross situations

	has_comments=true	*has_comments=false*
same_user=true	74.5%	88.3%
same_user=false	82.1%	93.3%

From the table, we can find that for pull requests without comment, the merge rate increases for both cases of factor *same_user*. However, we find that the merge rate even reaches 93% when *same_user=false*. Such high probability may be why this factor plays a decisive role in explaining pull request decisions when there is no comment.

Regarding integrator experience (*prior_review_num*), the explained variance increased from 14% (row 1, column 4 - has comments: yes) to 22% (row 1, column 5 - has comments: no). It is likely that when there are no comments, there are cases in which developers close or merge their own pull requests. In comparison to core members, external developers do not have the right to merge. This restricted permission linked to the integrator's review experience can potentially influence the pull request decision.

For the lifetime of a pull request (*lifetime_minutes*) and the number of commits included in a pull request (*num_commits*), when there exist comments, the integrator tends to make the decision based on the contributor's response speed and how he/she modifies the contribution according to the integrator's suggestions. This can be a reason why there exists a higher percentage of variance in situations where comments exist.

When there is no communication between the contributor and reviewers, factors indicating the affiliation of a contributor to the project - whether the contributor and the integrator are the same (*same_user*) and review experience (*prior_review_num*), are important in influencing pull request decisions.
When there is communication between the contributor and reviewers, factors representing the activeness of the interaction (*lifetime_minutes*, *num_commits*) have a bigger influence on pull request decisions.

3.2.3 Project characteristic

As team size increased, the variance explained by the experience of the integrator (*prior_review_num*) initially decreased from 11% (row 1, column 8 - team size: small) to 9% (row 1, column 9 - team size: mid) and then increased from 9% (row 1, column 9 - team size: mid) to 19% (row 1, column 10 - team size: large).

When considering whether pull requests were submitted and integrated by the same user (*same_user*), the change trend was the opposite, increasing from 24% (row 7, column 8 - team size: small) to 36% (row 7, column 9 - team size: mid) and then decreasing from 36% (row 7, column 9 - team size: mid) to 33% (row 7, column 10 - team size: large).

These two types of change indicate that for pull requests targeting teams of different sizes, the importance of *prior_review_num* and *same_user* changed nonlinearly. However, we have no explanation for this observation.

As team size increases, integrator's experience (*prior_review_num*) and whether submitter and integrator are the same (*same_user*) have a V-shaped and inverted V-shaped relations to pull request decisions respectively.

3.2.4 Supporting tools

When not using CI tools, the percentage of variance explained by comments (*has_comments*) was 25% (row 6, column 7 - ci exists: no), which was higher than that of pull requests using CI tools (10%) (row 6, column 6 - ci exists: yes). This result can be explained by the fact that when there are no CI tools, contributors can obtain feedback only from reviewers. Therefore, whether comments exist matters greatly in pull request decisions. When using CI tools, contributors can first obtain responses from CI outcomes, which can help with making decisions.

For factor *same_user*, its explained variance decreases from 33% (row 7, column 6 - ci exists: yes) to 23% (row 7, column 7 - ci exists: no). According to the previous study [11], teams using CI tools are more effective at merging pull requests submitted by core members. Therefore, we think that the existence of CI tools leads contributors to be more able to make judgments about their own contributions through the build outcome.[13][14]

The use of CI tools leads to significant changes in the influence of two factors on pull request decisions, *i.e.,* whether the pull request contains comments and whether the contributor and the reviewer are the same people. When using CI tools, the availability of CI build results makes the comments less important in explaining pull request decisions, while the influence of contributor and integrator's relationship becomes stronger.

3.2.5 Project evolution

Before June 2016, the experience of the integrator (*prior_review_num*) explained just 6% (row 1, column 11 - period: before 2016.6) of the variance, which increased to 17% after June 2018 (row 1, column 13 - period: after 2018.6). We calculated the experience of integrators corresponding to pull requests at different periods of project development, as shown in Figure 3. We find that the gap between integrators' experience for merged and unmerged pull requests gradually increases as projects become mature. This is why the variance explained by factor *prior_review_num* gradually increases. This indicates that the integrator's experience gradually becomes an important indicator of pull request decisions as the project evolves.

For the area hotness of contributions (*commits_on_files_touched*), before June 2016, it had a moderate effect on the decision-making of pull requests, which explained 7% of the variance (row 5, column 11 - period: before 2016.6), and increased the odds of acceptance by 30% per unit. However, as projects became mature, the variance explained decreased to 0% (row 5, column 13 - period: after 2018.6). For the three periods, we also calculated the mean value of *commits_on_files_touched* (before 2016.6: 40, 2016.6-2018.6: 33, and after 2018.6: 28), which shows that the contributions in the early stage of the project were more concentrated. In other words, as projects become larger and more mature, contributions are more widely distributed, and the area hotness of pull requests can hardly contribute to the merging of pull requests for mature projects.

For the lifetime of pull requests (*lifetime_minutes*), the explained variance decreased from 20% (row 2, column 11

13. https://github.com/react-boilerplate/react-boilerplate/pull/2256
14. https://github.com/mggg/GerryChain/pull/290

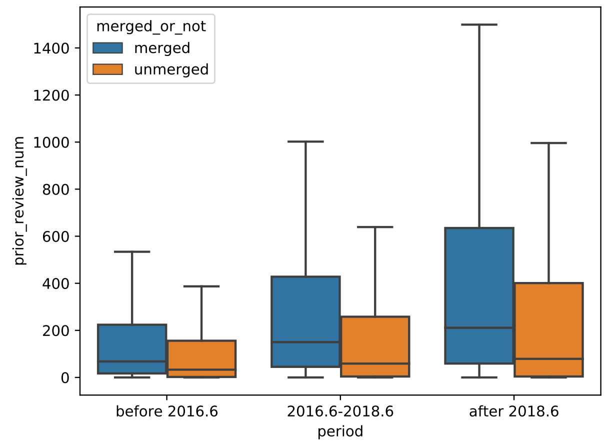

Fig. 3: The comparison between integrators' experience

- period: before 2016.6) to 13% (row 2, column 13 - period: after 2018.6). Although this factor negatively influenced the pull request merging in all three time periods, the effect size decreased. We calculated the changes in the pull request lifetime median value as projects evolve. It is found that the overall processing time of pull requests increases significantly (before 2016.6: 802min, 2016.6-2018.6: 1,188min, and after 2018.6: 1,316min). There are many possible reasons for this situation. *E.g.*, at the beginning of a project, the development team is small, and the pull requests that have been left unprocessed for a long time are likely to be rejected. As the project develops, more pull requests are left unprocessed (before 2016.6: 58, 2016.6-2018.6: 112, and after 2018.6: 174). The reviewers have their processing order, so the overall processing time of pull requests grows, but the impact on the decision becomes smaller. Also, we think as projects become mature, the use of various supporting mechanisms in the review process becomes stabilized, *e.g.*, the use of CI tools [10], the request of reviews [47], etc. These mechanisms lead to the increase of pull request lifetime. However, the standardized processes reduce the impact of processing time on the final result. There may not be a single reason for the change in results. Still, the result reveals that pull request processing time on decision-making decreases as the project develops.

> As a project evolves, the integrator's experience (*prior_review_num*) becomes more and more important for pull request decisions, while the area hotness of contribution (*commits_on_files_touched*) no longer influences the decision making.
> Compared to the early stages of project evolution, the influence of pull request lifetime (*lifetime_minutes*) on pull request decisions decreases.

4 Case Study

Since companies' contribution is relatively high in the open source world [48], the strategy, decision making, and participation patterns of different companies in open source vary greatly [49]. The participation of companies in open source projects also impacts the inflow and retention of external contributors [50]. Therefore, we also consider it interesting to analyze the impact of affiliation-related factors on pull request decisions. Therefore, we added the analysis of affiliation-related factors.

We first merge developer accounts and ignore those with more than one affiliation (this may be due to developers' affiliation). When considering the merge rate (Figure 4,5,6), we only consider the pull requests submitted and integrated by different users, as factor *same_user* significantly influences pull request decisions and acts as the precondition of factor *same_affiliation*.

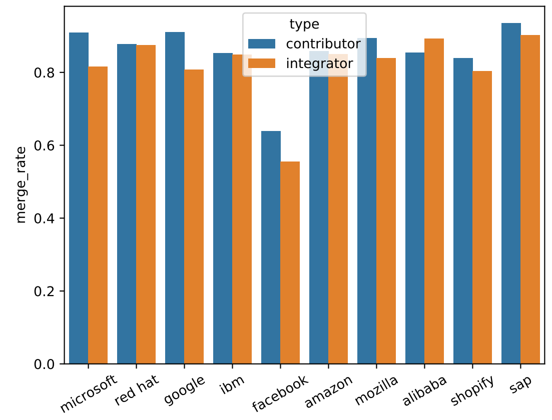

Fig. 4: Merge rate of top 10 affiliation when acting as contributor and integrator respectively

Different affiliations have different contribution intensities regarding the number of submitted and integrated pull requests [49]. Figure 4 shows that the merge rate for different affiliations varies a lot. For Facebook, its related pull requests' merge rate is much lower than other affiliations. This may be related to differences in policies or the way contributions are handled by different companies.

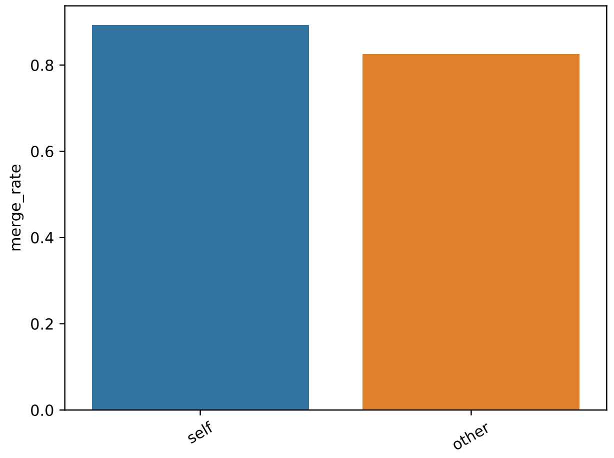

Fig. 5: Overall merge rate for affiliations integrating their own contributions (self) or contributions from other affiliations (other)

Second, we consider the effect of whether the pull request submitter and the integrator are from the same affiliation on pull request decisions. In the overall case, the merging probability is higher for pull requests submitted by their colleagues than those by developers from other affiliations (see Figure 5), which is in line with our perception. However, from the result of **RQ2** (Table 5 *same_user*=0), we found that when considering together with other factors, the factor *same_affiliation*, although significantly associated with pull request decisions, is less effective (explaining only 0.3% variance).

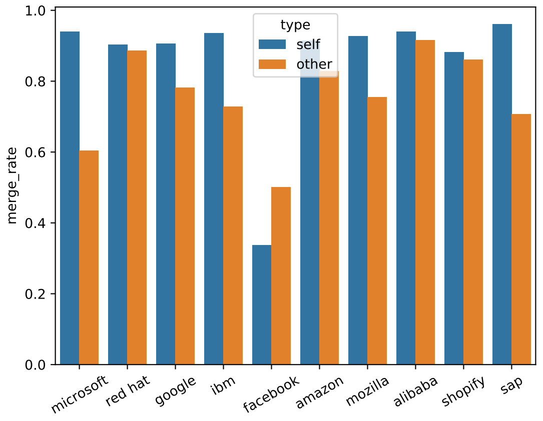

Fig. 6: Merge rate of different affiliations when integrating their own contributions (self) or contributions from other affiliations (other)

Our statistical analysis of each company reveals differences in the way companies treat their own contributions and external contributions (see Figure 6). For Facebook, the probability of merging external contributions is even higher than that of merging internal contributions. We think that the policy and openness of different companies lead to the different treatment of external contributions.

5 Discussion

5.1 Pull request decisions explained

Our study shows that there is no one answer to our research questions. Instead, there are generic answers and specific answers for the context represented, given the dependencies among factors. Generally, whether a pull request is submitted and integrated by the same person, its lifetime, experience of the integrator, presence of comments, and coreness of the contributor play decisive roles in pull request decisions. When comments in pull requests exist, the positive emotion for communication influences pull request decisions. When pull requests use CI tools, the percentage of build failure influences the decision.

Interestingly, the influence of the factors changes with a change in context:

Developer characteristic (same user or not): Compared to pull requests integrated by different persons, when pull requests are submitted and integrated by the same person, the importance of the integrator's experience and the contributor's coreness increase for pull request decisions, while the importance of the pull request lifetime and the included number of commits decreases (Section 3.2.1).

Pull request characteristic (has comments or not): When pull requests have comments, the lifetime and the number of commits included are more important compared to pull requests without any comment. In contrast, the importance of the integrator's experience and whether the contributor and integrator are the same person are less important when comments exist (Section 3.2.2).

Project characteristic (different team sizes): The importance of the integrator's experience and whether the contributor and the integrator are the same person for pull request decisions changes nonlinearly for teams of different sizes (Section 3.2.3).

Tool (CI exists or not): The use of CI tools decreases the importance of comment existence, but the importance of whether the contributor and the integrator are the same person increases for pull request decisions (Section 3.2.4).

Project evolution (different periods): The importance of the integrator's experience in pull request decisions increases as projects evolve, while the importance of area hotness and the lifetime of the contribution decreases (Section 3.2.5).

5.2 Relations to the literature

5.2.1 Discussion of previous conclusions

Referring to the literature (summarized in Table 10), relatively speaking, project-related factors are less discussed than pull-request- and developer-related factors. To this end, our study contributes in that not only have few project characteristics been explored in the literature, but they have been considered relatively less important (explains 2% of the variance) than developers (explains 52% variance) and pull request characteristics (explains 46% variance). Our study further provides evidence that human factors are as important or more important than technical factors [51].

When comparing the findings of previous studies with each other and those of our study, we found that in most of the cases, the results were consistent. Only four factors had opposite findings regarding the direction of influence, *i.e.*, *files_changed*, *project_age*, *team_size* and *num_commits*. One potential explanation that has emerged from our study is that all these factors are relatively less important for pull request decisions, which can potentially explain the differences in the findings. Alternatively, this can simply be due to the differences in the dataset used. Interestingly, many factors that are widely studied across related works, *e.g.*, *core_member* and *src_churn*, indicating that these factors are likely to influence the decision, are not as important for pull request decisions.

For the factor *num_commits*, which is relatively important, ranking in the top 10 across models (Table 6), we focus on this factor to uncover the reasons for conflict findings between previous studies. Yu et al. [10] found a positive effect (the likelihood of pull requests being accepted increases as the number of commits increases), while other studies [52], [53], [54] found a negative effect. Our results are consistent with Yu et al. and argue that the number of commits cannot simply indicate the contribution size. At the time of submission, the number of commits indicates the contribution's size to some extent. However, as the pull request review process continues, contributors will modify their contributions based on the review feedback and thus complete more commits to facilitate the merging of contributions. Accordingly, we collect the number of commits contained in a pull request at both open time and close time, investigate their effects on pull request merging separately, and find that the number of commits at commit time is negatively correlated with pull request merging. At the same time, it shifts to a positive correlation at close time. [15] Therefore, when a pull request is submitted, the number of commits represents the size of the contribution [52], [53], [54]. However, commits during the review process represent the changes made by the contributor according to the reviewers' comments, thus increasing the likelihood of pull request acceptance [10].

5.2.2 Findings in general context

Considering all pull requests without distinguishing between contexts, the top 5 factors for explaining pull request decisions are: whether the contributor and integrator are the same people (*same_user*), the lifetime of pull requests (*lifetime_minutes*), the experience of the integrator (*prior_review_num*), whether there exists comment (*has_comments*), whether the contributor is the core member (*core_member*).

1) *same_user*. The association of this factor reflects the decision propensity of self-integration in the pull-based development model, *i.e.,* a preference for self-rejected rather than self-approved. As you can see from the related work [55], the self-approved patch is defect-prone. To address this situation, future researchers need to consider whether to change the pull-based development model, *e.g.,* for self-approved contributions, generate a warning to other developers in the community.
2) *lifetime_minutes*. In related works [52], [56], they only discussed the direction of the association of this factor with pull request decisions. We found that, compared to other factors, the correlation between lifetime and pull request decisions is relatively high. In the future, when exploring the influence of factors on pull request decisions, the lifetime should be considered as an essential control variable.
3) *prior_review_num*. This factor is not considered to have a significant association with pull request decisions in related work [57]. However, our result shows that it is significantly important, which ranks the third when considering other factors in an overall perspective. The conflict of conclusion here is not to negate the past research but offers a view applicable at a large scale, as Baysal et al. only did a case study on two projects.
4) *has_comments*. Many previous studies focused on the association between the number of comments and pull request decisions [2], [10], [12], [15], [24]. Although there were studies focused on comment existence [53], [58], there is no discussion on its importance and comparing these two factors. Our result finds that the existence of comments is relatively important and can replace the number of comments in explaining pull request decisions.
5) *core_member*. For this factor, compared to previous studies [2], [10], [15], [24], [59], [60], we not only conclude a positive correlation of consistency but also find that the factor has a sizable effect when compared with all the other factors. Unlike the top 4 factors, this factor is present at the time of pull request submission. Therefore, this factor has an irreplaceable effect on predicting pull request decisions at the open time of pull requests.

5.2.3 Findings in different contexts

Under different contexts, we find the relative importance of the influence of postconditional factors. In previous studies, while Iyer et al. [24] found that both positive emotion and negative emotion significantly affect pull request decisions, our results, on the other hand, found that only positive emotion had a sizable effect when considering all factors. It also illustrates that when there exist comments, effectively tapping the hidden positive emotion in comments is important for predicting the final states of pull requests. Also, for pull requests using CI tools [10], the pass of CI builds positively and significantly influences the merging of pull requests. However, our model verifies its relative importance compared to other factors, *i.e.,* the decisions of pull requests are heavily influenced by the outcome of CI builds, which is the third most important factor in explaining pull request decisions.

While having comments leads to a lower probability of merging pull requests, it is needed to differentiate according to the characteristics of the commenter. We found that if there exist comments from others (*other_comment*), e.g., end-users or external developers, the pull request is more likely to be merged (Section 3.1.1). Different from Golzadeh et al. [61], we validated on a much larger dataset and consider different kinds of projects instead of just Cargo ecosystem.

The importance of factors changes and varies significantly as the context changes. And these findings have not been explicitly discussed in previous studies. We find that the number of commits has a sizable effect on the decision-making of pull requests containing comments. For those without comments, the effect is relatively small. This leads to the fact that when studying factors' association with pull request decisions, the impact of the number of commits on pull request decisions should be fully considered when there is no comment. Similarly, for pull requests that do not use CI tools, more significant consideration needs to be given to the weight of the comment. As the project develops, the importance of the factors changes. Among them, the influence of contribution's area hotness (*commits_on_files_touched*) on pull request decisions should be considered for the early stage of the project. And as projects become mature, the experience of integrators becomes important.

5.3 Implications

Our findings have implications for research and practice. Unlike related work, we construct a model from a more comprehensive perspective by collecting measurable factors from all pull request decision-related papers to explain the

15. https://github.com/zhangxunhui/TSE_pull-based-development/blob/main/technical_report.pdf

association and relative importance of factors with pull request decisions. The discussion of different contexts reveals the influence of context on the relevance of factors, which guides future related studies to select appropriate control variables when empirically analyzing pull request decisions in global or different contexts. Some findings from the study also provide theoretical support for future research and the optimization of pull-based development models. Next, we will discuss the implications in detail.

5.3.1 For research

For future research, this paper can give some guidance. For example:

When conducting research on pull request decisions, researchers can find usable findings from our paper for both a general overview and specific contexts (see Section 3.1). *E.g.*, when studying the association of new factors with pull request decisions, different factors should be considered as control factors for different situations, and here we give the recommended list (see Table 9) (the set of factors with more than 1% of explained variance in various situations). For other contexts, our dataset and scripts can be used to find the factors that rank high on the explanation of pull request decisions in the corresponding contexts as control variables.

Since the impact of a factor on the decision may vary at different periods of the pull request (e.g., *num_commits* - Section 5.2), we think that future research and the construction of evaluation tools need to consider the impact of changing factor dynamics.

When conducting research related to pull-based development, researchers can find useful data and conclusions. *E.g.,* when studying how CI tools influence the code review process, researchers can easily find that in an overall perspective, the usage of CI tools increases the likelihood of pull request acceptance (Section 3.1.1), and the outcome of CI builds significantly influence the decision making with large effect (Section 3.1.2). However, there still exist exceptional cases, *e.g.,* merge without passing CI builds. Thus, subsequent studies can be conducted based on our data and findings.

5.3.2 For practice

The results of our study can provide open source contributors and maintainers with many recommendations for practices to follow. For example:

For pull request contributors, if they want to increase the chances of having their contributions being accepted, they should respond to criticism from stakeholders on time, as the lifetime significantly influences pull request decisions with a large effect size.

Suppose there are other non-reviewers involved in the discussion (*other_comment* exists). In that case, the pull request is more likely to be merged, and contributors are advised not to give up and modify it according to the project requirements. As "developers need be more aware of the human-centric issues of their end-users," [62] one possible explanation for the influence of *other_comment* is that end-user feedback can help a lot in improving the quality of the software. [16] The discussion may be closely related to the project requirements and development direction, which directly influences whether the contribution can be merged or not [63].

16. https://github.com/rails/rails/pull/20851

For pull request maintainers, as the build outcome of CI tools significantly influences pull request decisions, we recommend maintainers install related CI tools to help improve the merge rate of contributions.

Contributions that remain unprocessed for a long time are likely not to be merged. On the one hand, maintainers purposely do not pick pull requests that are either not to their interest or do not need immediate attention. On the other hand, reviewers do not respond at the right time [64]. The delay of response may lead to the loss of peripheral contributors [65] and produce many abandoned contributions in the long run [66]. We think project managers can use the mention-bots to reduce the response time [67]. Or predict and alert on pull request remaining processing time to speed up the code review [3].

For both contributors and integrators, we suggest they participate in the review process with a positive attitude and promote the merging of contributions encouragingly. Our study further solidifies the importance of positive emotion for pull request decisions by integrating multiple factors. A positive atmosphere is of great importance for intra-project communication and efficient collaboration [68].

For the improvement of the pull-based model, as we find that self-integrated pull requests are likely to be rejected, and a previous study [55] found that self-approved contributions are bug-prone. Therefore, some adjustments can be made to self-integration. For self-integrated pull requests, the integrator's experience is a determinant factor for the decision of pull requests. We wonder if a warning flag could be added to pull requests integrated by inexperienced integrators to attract others for verification.

6 Threats to Validity

Our work builds on a decade of research on pull-based development, extracting the features relevant for pull request decision-making. In this way, we stand on the shoulders of giants and hence benefit from it and inherit the limitations of the features they present. In addition, we face the following limitations and classify them into four categories, *i.e.,* construct validity, internal validity, external validity, and conclusion validity [69].

6.1 Construct Validity

- The measure of relative importance may change if we choose a different method, which may lead to a different conclusion. There are different ways to calculate the importance of factors in a logistic regression model, *e.g.,* the percentage of variance explained by each factor [45], which is similar to the percentage of total variance explained by least squares regression [39], the standardized coefficient [70], and the change in logistic pseudo partial correlation [71]. This is a research field in itself and relates to the choice of the algorithm [72], [73]. To compare the importance of factors in different models, in this paper, we choose the percentage of explained variance to represent factor importance. The choice of the metric may affect the consistency of the

TABLE 9: The recommended control factors for different contexts

	overall	other-integrated	self-integrated	has comment	no comment	use CI	no CI	early stage of projects
same_user	✓			✓	✓	✓	✓	✓
lifetime_minutes	✓	✓	✓	✓	✓	✓	✓	✓
prior_review_num	✓		✓	✓	✓	✓	✓	✓
has_comments	✓	✓	✓			✓	✓	✓
core_member	✓		✓	✓	✓	✓	✓	✓
num_commits	✓	✓	✓	✓		✓	✓	✓
other_comment	✓	✓	✓	✓		✓		✓
ci_exists	✓	✓			✓			
hash_tag	✓		✓	✓		✓	✓	
account_creation_days		✓			✓			
commits_on_files_touched		✓			✓		✓	✓
reopen_or_not			✓		✓			
open_pr_num			✓		✓			✓
prev_pullreqs			✓					
first_pr			✓					
files_added			✓					
contrib_open			✓					
perc_pos_emotion				✓				
description_length					✓			
ci_failed_perc						✓		
num_comments							✓	
files_changed								✓
followers								✓

Note: ✓marks the recommended control factors when building logistic regression models for pull request decisions

conclusion to a certain extent. However, as this metric is widely used in many related works [10], [12], [74], our result can reflect the influence of factors on pull request decisions to a certain extent.

- The inconsistency between the GHTorrent dataset and the results returned by the GitHub API brought about errors in the time-related factors, which may influence the results. We checked 100 randomly selected records for each of the four factors *first_response_time*, *account_creation_days*, *project_age*, and *ci_latency*, and the precision was 98%, 97%, 96%, and 94%, respectively. Our dataset has inherited the problems, but from our investigation, the number of errors in our dataset is small compared to the size we have used for analysis.
- A developer may have multiple accounts in GitHub. We did not combine the accounts in our model. However, we analyzed this situation with a relevant tool [75] and found that 94% of the accounts in our dataset corresponds to only one developer. Due to the importance of the factor *same_user* in our model, we examined the reliability of the factor and found that the case of a user having multiple accounts does not affect its accuracy.
- For RQ2, we divided the data according to team size and the closing time of pull requests. This paper does not discuss the robustness of threshold selection, which may lead to less reliable conclusions. However, according to previous studies [52], [53], they split the data into three subsets for the trend analysis. Also, there are infinite ways to select the data division threshold, which can lead to differences in data size for different subsets. While optimizing the differences of data subsets, our result effectively reflects different contexts' influence on pull request decisions.

6.2 Internal Validity

- The absence of factors may have an impact on the relative importance of factors in the conclusion. We consider factors that can be mined from archival data and exclude those factors, *e.g.,* eye track-related factors [21], that are difficult to quantify in a scalable manner. These factors also include factors that focus only on specific scenarios, *e.g.,* factors related to Microsoft [3] and npm ecosystems only [16]. Because these factors also influence pull request decisions, as mentioned in previous studies, removing them can impact factors' relative importance on affecting pull request decisions. We are not sure how these factors perform together with our collected factors. At least, we have collected as many relevant factors as possible, quantified them, and added them to our dataset. Also, during data preprocessing, we remove the factor *bug_fix* due to 99.3% missing values, and thus, we are not sure how this factor affects pull request decision-making. Although many tools can predict whether a pull request fixes a bug, we only use the manually added label to classify pull requests to ensure data's accuracy. Future studies that want to delve deeper into the impact of these deleted factors can use other tools to complement this data for further analysis.
- There lacks a careful consideration of different types of projects. It is undeniable that when building models, it's better to consider different kinds of projects separately. However, the heterogeneity of projects has many dimensions, not only limited to the code contribution and

review process. Therefore it isn't easy to achieve accurate classification of projects. This paper has considered the issue of project heterogeneity to some extent, which includes many project related factors and treats team size as a project context.

6.3 External Validity

- Project selection introduces data bias when building models, resulting in our conclusions that may not apply to the complete set of GitHub data or some specific types of projects. *E.g.*, projects written in programming languages other than Java, Python, Ruby, JavaScript, Scala, and Go. Since it is impractical to model using data from the complete GitHub collection, the diversity of our data can help avoid this problem to a certain extent. Similarly, when selecting the projects, we selected the top 3% of projects in terms of the number of submitted pull requests and filtered out the projects in which the number of closed pull requests was less than 20. In Section 2.2, we mentioned that we specified these thresholds through discussion for the scalability and validity of the dataset. We cannot guarantee that our conclusions are available for other projects. We have at least tried to select the proper set of projects.
- The generalizability of our study is not verified in other social coding platforms (other than GitHub) or other modern code review tools, *e.g.*, Gerrit. One major reason for differences can be the factors influencing pull request decisions on different platforms. The comparison of factors' influence on contribution decisions under different platforms or tools belongs to another research in the future.

6.4 Conclusion Validity

- For logistic regression models, comparing the variance explained by the same factor in different models is not accurate. This may affect the correctness of the conclusions, as the variance explained by the factors in different regression models fluctuates when different models use different training sets. But there is not a good solution to the problem. However, in our study, we consider only the factors that change dramatically in different contexts. When building models with the same set of predictors, large changes in explained variance can be used to describe the change in factor importance.

7 Related Work

The related work of this paper is mainly divided into four parts. The first subsection introduces modern code review. The second subsection introduces factors influencing pull request decisions. Third, we introduce papers that tried to integrate related factors and explain the relative importance of the factors influencing pull request decisions. Fourth, we discuss other studies that have introduced scientific research methods based on big data.

7.1 Modern Code Review

Although Fagan et al. developed a structure of code inspection in 1976 [76], it is very time-consuming and not applicable in practice [77]. Therefore, modern code review comes into being, which is informal, tool-based, and occurs regularly in practice [78].

Many tools or platforms support modern code review. Different companies and organizations use various tools and have their policy during the code review process [79]. CRITICS [80], ReviewClipse [81], and Mylyn Reviews [82] are code review tools integrated into IDE, combining the code review and development process. Another popular tool called Gerrit [83], which supported many projects including Android, OpenStack is a Git-based tool. CodeFlow, which is similar to Gerrit, is widely used by Microsoft [78].

In recent years, the pull-based development model has become a new paradigm for distributed software development. Many code-hosting sites, notably GitHub, support the model by integrating it with code review systems [1]. Unlike Gerrit, pull requests on GitHub focus not only on a single commit but also on a whole branch [84]. In contrast, pull request is easy to participate in the contribution process without having to master many git operations [85]. Its well-designed user interface and support for social collaboration help improve the usability and code review process of GitHub [86]. These characteristics help GitHub get more than 79 million users and 238 million repositories. Therefore, we would like to start with GitHub's pull-based model to explain the factors associated with pull request decisions.

7.2 Factors influencing pull request decisions

The factors influencing pull request decisions can be divided into three categories, namely, developer characteristics, project characteristics and pull request characteristics.

7.2.1 *Developer characteristics*

Developer characteristics are related to the contributor and the integrator. This category contains factors related to human beings and interactions between two contributors or a contributor and a project. This category includes `basic information` on developers, including their *gender* [87], *country* information [12], and *affiliation* [88], [89]. Some studies focus on `personal features`, including the *personality* and *emotion* of developers [2], [24], while others studied the `relationship` between the developer and the target project, including the *experience* of developers, which is conceptualized as the count of previous pull requests, accepted commit count [90], days since account creation [91], whether it is the first pull request of the contributor [52], [53], the prior reviews of the integrator [89], the *coreness* of the contributor [10], [15], [52], [59], [92], [93], the *social distance* [15] and *social strength* [10] of contributor to the integrator, and the *response time* of the integrator to the pull request [10].

7.2.2 *Project characteristics*

Studies on project characteristics mainly talk about the `basic information` of target projects when submitting pull requests, which can be summarized into the following aspects: *programming language* [52], [58], [91], project *popularity*, measured as watcher count [28], star count [28], fork count [54], [91], *age* of the project [10], [15], *workload* measured as the number of open pull requests [10], [89], *activeness* measured as the time interval in seconds between the opening time of the two latest pull requests [54], and *openness* measured as the count of open issues [54].

7.2.3 Pull request characteristics

Related works focus on the `basic information` of pull requests, which includes the *size of the change* measured at the file level, commit level, and code level [10]; the *complexity of a pull request* measured as the length of description [10]; the *nature of pull requests* measured as bug fixes [58], [90], the *test inclusion* of pull requests [10], [15], [92], and the *hotness* or relevance of a PR [1], [10], [15], [53], [88], [90]. Additionally, some studies focus on the `process information` of pull requests generated during the code review process, including the *reference* of a contributor, issue or pull request [10], [25]; the *conflict* of a pull request [1]; the *complexity of discussion* [28]; the *emotion in discussion* [24]; and *CI tool usage* during the review process [10], [11], [26], [94], [95].

7.3 Attempts at explaining pull request decisions

Few studies have tried to integrate the factors related to pull request decisions and have explored their relative importance in predicting outcomes. Gousios et al. [1] first collected a set of factors and performed a preliminary exploration of relative importance based on the random forest method. However, it was in the early stage of this study area. Tsay et al. [15] used an explanatory method to explore the importance of social and technical factors. However, similar to Gousios et al.'s work [1], their work also acted as groundbreaking research, leading to the emergence of many other studies. Since then, a few follow-ups have come into being, *e.g.,* personality-related factors [2], geographical location [12], and CI-related factors [10]. In 2020, Dey et al. [16] collected 50 factors of 483,988 pull requests based on 4,218 projects. They also used random the forest method to determine the important factors in predicting the decision. However, they focused only on the npm community and gathered factors without conducting a systematic literature review. As a result, factors related to CI, personality, emotion, geographical, etc., were missing. Furthermore, to the best of our knowledge, no study has synthesized the existing body of knowledge to empirically explain pull request decisions.

7.4 Big-data-based scientific research methods

Big data has provided many research opportunities, for which there are mainly two research methods, *i.e.,* data-driven and theory-driven methods. Maass et al. [96] discussed the difference between these two methods and found that the data-driven method first focuses on the data and then extracts patterns and forms into theory. However, the theory-driven method first comes up with a theory and uses data to prove it. Therefore, our study is data driven, finding patterns in different subsets of data and forming them into theory.

For the process of a data-driven study, Kar et al. [97] suggested that there are 6 main steps for building up a theory, *i.e.,* data acquisition, data conversion, data analysis, factor identification, theory development and model validation.

There are many studies in different research areas that have used data-driven research methods. For example, Greenwood et al. [98] studied the influence of race, gender, and socioeconomic status on the incidence rate of human immunodeficiency virus (HIV) infection using data from 12 million patients. Likewise, other previous studies [1], [10], [12], [15] on pull request decisions all used data-driven methods.

However, for the data acquisition part, previous studies focused only on one specific type of factor or several self-defined factors. Without including all the related factors, one can hardly gain an overall grasp of the influence of all factors. Therefore, we conducted a systematic literature review in this study. According to Kitchenham et al. [99], a systematic literature review is an important part of evidence-based software engineering (EBSE), as it can aggregate all existing evidence and provide guidelines for researchers.

8 Conclusions

This study synthesizes the existing body of knowledge to empirically explain pull request decisions. Our mixed effects logistic regression models built on large and diverse GitHub project data show that a handful of factors (5 to 10) explain pull request decisions the most. The most important factor influencing pull request decisions is whether the contributor and the integrator are the same user, explaining more than 30% of the variance. Surprisingly, this factor did not surface in any of the prior works and is thus a contribution of this study. In addition, positive emotions during discussion and CI build results become relatively more important when a pull request has comments and uses CI tools, respectively. Furthermore, we noticed that the use of CI tools replaced the function of comments, indicating changes in the influence of these factors. We think that this study has empirically synthesized an explanation for pull request decisions that is useful for research and practice.

Acknowledgment

This work is supported by National Key R&D Program of China (2020AAA0103504). Thank you Rahul N. Iyer, Frenk van Mil, Celal Karakoc, Leroy Velzel, Daan Groenewegen, and Sarah de Wolf for your help in implementation. Thanks Mengluan Cai for validating the validity of factor extraction.

References

[1] G. Gousios, M. Pinzger, and A. v. Deursen, "An exploratory study of the pull-based software development model," in *Proceedings of the 36th International Conference on Software Engineering*, ser. ICSE 2014. New York, NY, USA: Association for Computing Machinery, 2014, p. 345–355. [Online]. Available: https://doi.org/10.1145/2568225.2568260

[2] R. N. Iyer, S. A. Yun, M. Nagappan, and J. Hoey, "Effects of personality traits on pull request acceptance," *IEEE Transactions on Software Engineering*, pp. 1–1, 2019.

[3] C. Maddila, C. Bansal, and N. Nagappan, "Predicting pull request completion time: a case study on large scale cloud services," in *Proceedings of the 2019 27th ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering*, 2019, pp. 874–882.

[4] J. Jiang, Y. Yang, J. He, X. Blanc, and L. Zhang, "Who should comment on this pull request? analyzing attributes for more accurate commenter recommendation in pull-based development," *Information and Software Technology*, vol. 84, pp. 48–62, 2017.

[5] Y. Yu, H. Wang, G. Yin, and C. X. Ling, "Who should review this pull-request: Reviewer recommendation to expedite crowd collaboration," in *2014 21st Asia-Pacific Software Engineering Conference*, vol. 1, Dec 2014, pp. 335–342.

[6] Y. Yu, Z. Li, G. Yin, T. Wang, and H. Wang, "A dataset of duplicate pull-requests in github," in *Proceedings of the 15th International Conference on Mining Software Repositories*, 2018, pp. 22–25.

[7] Q. Wang, B. Xu, X. Xia, T. Wang, and S. Li, "Duplicate pull request detection: When time matters," in *Proceedings of the 11th Asia-Pacific Symposium on Internetware*, 2019, pp. 1–10.

[8] Z. Liu, X. Xia, C. Treude, D. Lo, and S. Li, "Automatic generation of pull request descriptions," in *2019 34th IEEE/ACM International Conference on Automated Software Engineering (ASE)*. IEEE, 2019, pp. 176–188.

[9] E. v. d. Veen, G. Gousios, and A. Zaidman, "Automatically prioritizing pull requests," in *2015 IEEE/ACM 12th Working Conference on Mining Software Repositories*, 2015, pp. 357–361.

[10] Y. Yu, G. Yin, T. Wang, C. Yang, and H. Wang, "Determinants of pull-based development in the context of continuous integration," *Science China Information Sciences*, vol. 59, no. 8, p. 080104, 2016. [Online]. Available: https://doi.org/10.1007/s11432-016-5595-8

[11] B. Vasilescu, Y. Yu, H. Wang, P. Devanbu, and V. Filkov, "Quality and productivity outcomes relating to continuous integration in github," in *Proceedings of the 2015 10th Joint Meeting on Foundations of Software Engineering*, ser. ESEC/FSE 2015. New York, NY, USA: Association for Computing Machinery, 2015, p. 805–816. [Online]. Available: https://doi.org/10.1145/2786805.2786850

[12] A. Rastogi, N. Nagappan, G. Gousios, and A. van der Hoek, "Relationship between geographical location and evaluation of developer contributions in github," in *Proceedings of the 12th ACM/IEEE International Symposium on Empirical Software Engineering and Measurement*, ser. ESEM '18. New York, NY, USA: Association for Computing Machinery, 2018. [Online]. Available: https://doi.org/10.1145/3239235.3240504

[13] Z. Hu and E. Gehringer, "Use bots to improve github pull-request feedback," in *Proceedings of the 50th ACM Technical Symposium on Computer Science Education*, 2019, pp. 1262–1263.

[14] Z. Peng and X. Ma, "Exploring how software developers work with mention bot in github," *CCF Transactions on Pervasive Computing and Interaction*, vol. 1, no. 3, pp. 190–203, 2019.

[15] J. Tsay, L. Dabbish, and J. Herbsleb, "Influence of social and technical factors for evaluating contribution in github," in *Proceedings of the 36th International Conference on Software Engineering*, ser. ICSE 2014. New York, NY, USA: Association for Computing Machinery, 2014, p. 356–366. [Online]. Available: https://doi.org/10.1145/2568225.2568315

[16] T. Dey and A. Mockus, "Which pull requests get accepted and why? a study of popular npm packages," *arXiv preprint arXiv:2003.01153*, 2020.

[17] B. Kitchenham and S. Charters, "Guidelines for performing systematic literature reviews in software engineering," 2007.

[18] A. W. Harzing, "The publish or perish book: Your guide to effective and responsible citation analysis," *International Review of Research in Open & Distance Learning*, vol. 13, no. 3, pp. 314–315, 2012.

[19] M. Gusenbauer, "Google scholar to overshadow them all? comparing the sizes of 12 academic search engines and bibliographic databases," *Scientometrics*, vol. 118, 2019.

[20] G. Jeong, S. Kim, T. Zimmermann, and K. Yi, "Improving code review by predicting reviewers and acceptance of patches," *Research on software analysis for error-free computing center Tech-Memo (ROSAEC MEMO 2009-006)*, pp. 1–18, 2009.

[21] D. Ford, M. Behroozi, A. Serebrenik, and C. Parnin, "Beyond the code itself: How programmers really look at pull requests," in *2019 IEEE/ACM 41st International Conference on Software Engineering: Software Engineering in Society (ICSE-SEIS)*, 2019, pp. 51–60.

[22] P. Pooput and P. Muenchaisri, "Finding impact factors for rejection of pull requests on github," in *Proceedings of the 2018 VII International Conference on Network, Communication and Computing*, ser. ICNCC 2018. New York, NY, USA: Association for Computing Machinery, 2018, p. 70–76. [Online]. Available: https://doi.org/10.1145/3301326.3301380

[23] M. Ortu, M. Marchesi, and R. Tonelli, "Empirical analysis of affect of merged issues on github," in *2019 IEEE/ACM 4th International Workshop on Emotion Awareness in Software Engineering (SEmotion)*, 2019, pp. 46–48.

[24] Iyer, Rahul, "Effects of personality traits and emotional factors in pull request acceptance." 2019. [Online]. Available: http://hdl.handle.net/10012/14952

[25] F. Calefato, F. Lanubile, and N. Novielli, "A preliminary analysis on the effects of propensity to trust in distributed software development," in *2017 IEEE 12th International Conference on Global Software Engineering (ICGSE)*, 2017, pp. 56–60.

[26] G. Gousios, A. Zaidman, M. Storey, and A. v. Deursen, "Work practices and challenges in pull-based development: The integrator's perspective," in *2015 IEEE/ACM 37th IEEE International Conference on Software Engineering*, vol. 1, 2015, pp. 358–368.

[27] X. Zhang, A. Rastogi, and Y. Yu, "On the shoulders of giants: A new dataset for pull-based development research," in *Proceedings of the 17th International Conference on Mining Software Repositories*, 2020, pp. 543–547.

[28] G. Gousios and A. Zaidman, "A dataset for pull-based development research," in *Proceedings of the 11th Working Conference on Mining Software Repositories*, ser. MSR 2014. New York, NY, USA: Association for Computing Machinery, 2014, p. 368–371. [Online]. Available: https://doi.org/10.1145/2597073.2597122

[29] B. Vasilescu, A. Capiluppi, and A. Serebrenik, "Gender, representation and online participation: A quantitative study," *Interacting with Computers*, vol. 26, no. 5, pp. 488–511, 2014.

[30] Q. Fan, Y. Yu, G. Yin, T. Wang, and H. Wang, "Where is the road for issue reports classification based on text mining?" in *2017 ACM/IEEE International Symposium on Empirical Software Engineering and Measurement (ESEM)*. IEEE, 2017, pp. 121–130.

[31] "Linking a pull request to an issue," https://docs.github.com/en/issues/tracking-your-work-with-issues/linking-a-pull-request-to-an-issue, [Online; accessed 4-November-2021].

[32] X. Zhang, A. Rastogi, and Y. Yu, "Technical Report," https://github.com/zhangxunhui/new_pullreq_msr2020/blob/master/technical_report.pdf, 2020, [Online; accessed 3-March-2021].

[33] StatsTest.com, "Cramer's V," https://www.statstest.com/cramers-v-2/, [Online; accessed 3-March-2021].

[34] J. S. Jones, *Learn to Use the Eta Coefficient Test in SPSS With Data From the NIOSH Quality of Worklife Survey (2014)*, 2019.

[35] J. Cohen, *Statistical power analysis for the behavioral sciences*. Academic press, 1969.

[36] A. Gałecki and T. Burzykowski, "Linear mixed-effects model," in *Linear Mixed-Effects Models Using R*. Springer, 2013, pp. 245–273.

[37] D. M. Bates, "lme4: Mixed-effects modeling with r," 2010.

[38] J. Frost, "Multicollinearity in Regression Analysis: Problems, Detection, and Solutions," https://statisticsbyjim.com/regression/multicollinearity-in-regression-analysis/#:~:text=Multicollinearity%20occurs%20when%20independent%20variables%20in%20a%20regression,you%20fit%20the%20model%20and%20interpret%20the%20results., [Online; accessed 6-April-2021].

[39] J. Cohen, P. Cohen, S. G. West, and L. S. Aiken, *Applied multiple regression/correlation analysis for the behavioral sciences*. Routledge, 2013.

[40] P. C. Pendharkar and J. A. Rodger, "An empirical study of the impact of team size on software development effort," *Information Technology & Management*, vol. 8, no. 4, pp. 253–262, 2007.

[41] S. W. Chou and M. Y. He, "The factors that affect the performance of open source software development – the perspective of social capital and expertise integration," *Information Systems Journal*, vol. 21, no. 2, pp. 195–219, 2011.

[42] X. Zhang, Y. Yu, G. Gousios, and R. Ayushi, "Technical Report," https://github.com/zhangxunhui/TSE_pull-based-development/blob/main/technical_report.pdf, 2021, [Online; accessed 23-March-2022].

[43] Y. Zhao, A. Serebrenik, Y. Zhou, V. Filkov, and B. Vasilescu, "The impact of continuous integration on other software development practices: A large-scale empirical study," in *2017 32nd IEEE/ACM*

International Conference on Automated Software Engineering (ASE), 2017, pp. 60–71.

[44] Ø. Langsrud, "Anova for unbalanced data: Use type ii instead of type iii sums of squares," *Statistics and Computing*, vol. 13, no. 2, pp. 163–167, 2003.

[45] B. Ray, D. Posnett, V. Filkov, and P. Devanbu, "A large scale study of programming languages and code quality in github," in *Proceedings of the 22nd ACM SIGSOFT International Symposium on Foundations of Software Engineering*, ser. FSE 2014. New York, NY, USA: Association for Computing Machinery, 2014, p. 155–165. [Online]. Available: https://doi.org/10.1145/2635868.2635922

[46] A. P. Bradley, "The use of the area under the roc curve in the evaluation of machine learning algorithms," *Pattern recognition*, vol. 30, no. 7, pp. 1145–1159, 1997.

[47] GitHub, "Approving a pull request with required reviews," https://docs.github.com/en/pull-requests/collaborating-with-pull-requests/reviewing-changes-in-pull-requests/approving-a-pull-request-with-required-reviews, [Online; accessed 18-November-2021].

[48] J. Corbet and G. Kroah-Hartman, "2017 linux kernel development report," *A Publication of The Linux Foundation*, 2017.

[49] Y. Zhang, M. Zhou, A. Mockus, and Z. Jin, "Companies' participation in oss development-an empirical study of openstack," *IEEE Transactions on Software Engineering*, 2019.

[50] M. Zhou, A. Mockus, X. Ma, L. Zhang, and H. Mei, "Inflow and retention in oss communities with commercial involvement: A case study of three hybrid projects," *ACM Transactions on Software Engineering and Methodology (TOSEM)*, vol. 25, no. 2, pp. 1–29, 2016.

[51] J.-M. Hoc, *Psychology of programming*. Academic Press, 2014.

[52] D. M. Soares, M. L. de Lima Júnior, L. Murta, and A. Plastino, "Acceptance factors of pull requests in open-source projects," in *Proceedings of the 30th Annual ACM Symposium on Applied Computing*, ser. SAC '15. New York, NY, USA: Association for Computing Machinery, 2015, p. 1541–1546. [Online]. Available: https://doi.org/10.1145/2695664.2695856

[53] D. M. Soares, M. L. d. L. Júnior, L. Murta, and A. Plastino, "Rejection factors of pull requests filed by core team developers in software projects with high acceptance rates," in *2015 IEEE 14th International Conference on Machine Learning and Applications (ICMLA)*, Dec 2015, pp. 960–965.

[54] N. Khadke, M. H. Teh, and M. Shen, "Predicting acceptance of github pull requests," 2012.

[55] S. McIntosh, Y. Kamei, B. Adams, and A. E. Hassan, "The impact of code review coverage and code review participation on software quality: A case study of the qt, vtk, and itk projects," in *Proceedings of the 11th Working Conference on Mining Software Repositories*, 2014, pp. 192–201.

[56] D. Legay, A. Decan, and T. Mens, "On the impact of pull request decisions on future contributions," *CoRR*, vol. abs/1812.06269, 2018. [Online]. Available: http://arxiv.org/abs/1812.06269

[57] O. Baysal, O. Kononenko, R. Holmes, and M. W. Godfrey, "Investigating technical and non-technical factors influencing modern code review," *Empirical Software Engineering*, vol. 21, no. 3, pp. 932–959, 2016. [Online]. Available: https://doi.org/10.1007/s10664-015-9366-8

[58] R. Padhye, S. Mani, and V. S. Sinha, "A study of external community contribution to open-source projects on github," in *Proceedings of the 11th Working Conference on Mining Software Repositories*, ser. MSR 2014. New York, NY, USA: Association for Computing Machinery, 2014, p. 332–335. [Online]. Available: https://doi.org/10.1145/2597073.2597113

[59] A. Bosu and J. C. Carver, "Impact of developer reputation on code review outcomes in oss projects: An empirical investigation," in *Proceedings of the 8th ACM/IEEE International Symposium on Empirical Software Engineering and Measurement*, ser. ESEM '14. New York, NY, USA: Association for Computing Machinery, 2014. [Online]. Available: https://doi.org/10.1145/2652524.2652544

[60] A. Lee and J. C. Carver, "Are one-time contributors different? a comparison to core and periphery developers in floss repositories," in *2017 ACM/IEEE International Symposium on Empirical Software Engineering and Measurement (ESEM)*, 2017, pp. 1–10.

[61] M. Golzadeh, A. Decan, and T. Mens, "On the effect of discussions on pull request decisions," 2019.

[62] H. Khalajzadeh, M. Shahin, H. O. Obie, and J. Grundy, "How are diverse end-user human-centric issues discussed on github?" *arXiv preprint arXiv:2201.05927*, 2022.

[63] J. Tsay, L. Dabbish, and J. Herbsleb, "Let's talk about it: evaluating contributions through discussion in github," in *Proceedings of the 22nd ACM SIGSOFT international symposium on foundations of software engineering*, 2014, pp. 144–154.

[64] X. Tan and M. Zhou, "How to communicate when submitting patches: An empirical study of the linux kernel," *Proceedings of the ACM on Human-Computer Interaction*, vol. 3, no. CSCW, pp. 1–26, 2019.

[65] I. Steinmacher, G. Pinto, I. S. Wiese, and M. A. Gerosa, "Almost there: A study on quasi-contributors in open-source software projects," in *2018 IEEE/ACM 40th International Conference on Software Engineering (ICSE)*, 2018, pp. 256–266.

[66] Z. Li, Y. Yu, T. Wang, G. Yin, S. Li, and H. Wang, "Are you still working on this an empirical study on pull request abandonment," *IEEE Transactions on Software Engineering*, 2021.

[67] Z. Peng and X. Ma, "Exploring how software developers work with mention bot in github," *CCF Transactions on Pervasive Computing and Interaction*, vol. 1, no. 3, pp. 190–203, 2019.

[68] X. Lu, Y. Cao, Z. Chen, and X. Liu, "A first look at emoji usage on github: An empirical study," *arXiv preprint arXiv:1812.04863*, 2018.

[69] L. Gren, "Standards of validity and the validity of standards in behavioral software engineering research: the perspective of psychological test theory," in *Proceedings of the 12th ACM/IEEE International Symposium on Empirical Software Engineering and Measurement*, 2018, pp. 1–4.

[70] S. Tonidandel and J. M. LeBreton, "Relative importance analysis: A useful supplement to regression analysis," *Journal of Business and Psychology*, vol. 26, no. 1, pp. 1–9, 2011.

[71] V. Agrawal, "Interpreting importance of features in logistic regression model [closed]," https://stats.stackexchange.com/questions/233050/interpreting-importance-of-features-in-logisitic-regression-model, [Online; accessed 24-March-2021].

[72] Aliweb, "How to choose the best algorithm for measuring attribute importance/relevance?" https://stats.stackexchange.com/questions/251248/how-to-choose-the-best-algorithm-for-measuring-attribute-importance-relevance, [Online; accessed 24-March-2021].

[73] M. Drury, "What are variable importance rankings useful for?" https://stats.stackexchange.com/questions/202277/what-are-variable-importance-rankings-useful-for, [Online; accessed 24-March-2021].

[74] C. Overney, J. Meinicke, C. Kastner, and B. Vasilescu, "How to not get rich: An empirical study of donations in open ource," *Proceedings - International Conference on Software Engineering*, pp. 1209–1221, 2020.

[75] B. Vasilescu, A. Serebrenik, and V. Filkov, "A data set for social diversity studies of GitHub teams," in *12th Working Conference on Mining Software Repositories, Data Track*, ser. MSR. IEEE, 2015, pp. 514–517.

[76] M. Fagan, "Design and code inspections to reduce errors in program development," in *Software pioneers*. Springer, 2002, pp. 575–607.

[77] F. Shull and C. Seaman, "Inspecting the history of inspections: An example of evidence-based technology diffusion," *IEEE software*, vol. 25, no. 1, pp. 88–90, 2008.

[78] A. Bacchelli and C. Bird, "Expectations, outcomes, and challenges of modern code review," in *2013 35th International Conference on Software Engineering (ICSE)*. IEEE, 2013, pp. 712–721.

[79] A. Bosu, "Modeling modern code review practices in open source software development organizations," in *Proceedings of the 11th International Doctoral Symposium on Empirical Software Engineering*, 2013.

[80] T. Zhang, M. Song, and M. Kim, "Critics: An interactive code review tool for searching and inspecting systematic changes," in *Proceedings of the 22nd ACM SIGSOFT International Symposium on Foundations of Software Engineering*, 2014, pp. 755–758.

[81] M. Bernhart, A. Mauczka, and T. Grechenig, "Adopting code reviews for agile software development," in *2010 Agile Conference*. IEEE, 2010, pp. 44–47.

[82] "Mylyn reviews," https://projects.eclipse.org/projects/mylyn.reviews, [Online; accessed 4-November-2021].

[83] "Gerrit code review," https://www.gerritcodereview.com/, [Online; accessed 4-November-2021].

[84] L. Vogel, "Gerrit code review - tutorial," https://www.vogella.com/tutorials/Gerrit/article.html, 2020, [Online; accessed 4-November-2021].

[85] "Gerrit code review, or github's fork and pull model?" https://softwareengineering.stackexchange.com/questions/173262/gerrit-code-review-or-githubs-fork-and-pull-model, 2012, [Online; accessed 4-November-2021].

[86] "Gerritforge blog - git and gerrit code review supported and delivered to your enterprise," https://gitenterprise.me/2013/10/17/gerrit-code-review-or-githubs-fork-and-pull-take-both/, 2013, [Online; accessed 4-November-2021].

[87] J. Terrell, A. Kofink, J. Middleton, C. Rainear, E. Murphy-Hill, C. Parnin, and J. Stallings, "Gender differences and bias in open source: Pull request acceptance of women versus men," *PeerJ Computer Science*, vol. 3, p. e111, 2017.

[88] O. Kononenko, T. Rose, O. Baysal, M. Godfrey, D. Theisen, and B. de Water, "Studying pull request merges: A case study of shopify's active merchant," in *Proceedings of the 40th International Conference on Software Engineering: Software Engineering in Practice*, ser. ICSE-SEIP '18. New York, NY, USA: Association for Computing Machinery, 2018, p. 124–133. [Online]. Available: https://doi.org/10.1145/3183519.3183542

[89] O. Baysal, O. Kononenko, R. Holmes, and M. W. Godfrey, "The influence of non-technical factors on code review," in *2013 20th Working Conference on Reverse Engineering (WCRE)*, Oct 2013, pp. 122–131.

[90] Y. Jiang, B. Adams, and D. M. German, "Will my patch make it? and how fast? case study on the linux kernel," in *2013 10th Working Conference on Mining Software Repositories (MSR)*, May 2013, pp. 101–110.

[91] M. M. Rahman and C. K. Roy, "An insight into the pull requests of github," in *Proceedings of the 11th Working Conference on Mining Software Repositories*, ser. MSR 2014. New York, NY, USA: Association for Computing Machinery, 2014, p. 364–367. [Online]. Available: https://doi.org/10.1145/2597073.2597121

[92] G. Pinto, L. F. Dias, and I. Steinmacher, "Who gets a patch accepted first? comparing the contributions of employees and volunteers," in *Proceedings of the 11th International Workshop on Cooperative and Human Aspects of Software Engineering*, ser. CHASE '18. New York, NY, USA: Association for Computing Machinery, 2018, p. 110–113. [Online]. Available: https://doi.org/10.1145/3195836.3195858

[93] O. Baysal, O. Kononenko, R. Holmes, and M. W. Godfrey, "The secret life of patches: A firefox case study," in *2012 19th Working Conference on Reverse Engineering*, Oct 2012, pp. 447–455.

[94] F. Zampetti, G. Bavota, G. Canfora, and M. D. Penta, "A study on the interplay between pull request review and continuous integration builds," in *2019 IEEE 26th International Conference on Software Analysis, Evolution and Reengineering (SANER)*, Feb 2019, pp. 38–48.

[95] Y. Tao, D. Han, and S. Kim, "Writing acceptable patches: An empirical study of open source project patches," in *2014 IEEE International Conference on Software Maintenance and Evolution*, Sep. 2014, pp. 271–280.

[96] W. Maass, J. Parsons, S. Purao, V. C. Storey, and C. Woo, "Data-driven meets theory-driven research in the era of big data: Opportunities and challenges for information systems research," *Journal of the Association for Information Systems*, 2018.

[97] A. K. Kar and Y. K. Dwivedi, "Theory building with big data-driven research – moving away from the "what" towards the "why"," *International Journal of Information Management*, vol. 54, 2020.

[98] B. N. Greenwood and R. Agarwal, "Matching platforms and hiv incidence: An empirical investigation of race, gender, and socioeconomic status," *Management Science*, vol. 62, no. 8, pp. págs. 2281–2303, 2016.

[99] B. Kitchenham, O. P. Brereton, D. Budgen, M. Turner, J. Bailey, and S. Linkman, "Systematic literature reviews in software engineering–a systematic literature review," *Information and software technology*, vol. 51, no. 1, pp. 7–15, 2009.

[100] P. Weißgerber, D. Neu, and S. Diehl, "Small patches get in!" in *Proceedings of the 2008 International Working Conference on Mining Software Repositories*, ser. MSR '08. New York, NY, USA: Association for Computing Machinery, 2008, p. 67–76. [Online]. Available: https://doi.org/10.1145/1370750.1370767

[101] J. Jiang, A. Mohamed, and L. Zhang, "What are the characteristics of reopened pull requests? a case study on open source projects in github," *IEEE Access*, vol. 7, pp. 102 751–102 761, 2019.

[102] C. Hechtl, "On the influence of developer coreness on patch acceptance: A survival analysis," 2020.

[103] V. Kovalenko and A. Bacchelli, "Code review for newcomers: Is it different?" in *Proceedings of the 11th International Workshop on Cooperative and Human Aspects of Software Engineering*, ser. CHASE '18. New York, NY, USA: Association for Computing Machinery, 2018, p. 29–32. [Online]. Available: https://doi.org/10.1145/3195836.3195842

[104] M. Hilton, T. Tunnell, K. Huang, D. Marinov, and D. Dig, "Usage, costs, and benefits of continuous integration in open-source projects," in *2016 31st IEEE/ACM International Conference on Automated Software Engineering (ASE)*, 2016, pp. 426–437.

Xunhui Zhang received his BS in Computer Science from Sichuan University in 2015. He received his MS in Software Engineering from National University of Defense Technology in 2017. He is now a PhD candidate in Software Engineering, National University of Defense Technology. His work interests include open source software engineering, data mining, recommendation system, cross community analysis and code clone.

Yue Yu is an associate professor in the College of Computer at National University of Defense Technology (NUDT). He received his Ph.D. degree in Computer Science from NUDT in 2016. He has won Outstanding Ph.D. Thesis Award from Hunan Province. His research findings have been published on ICSE, FSE, ASE, TSE, MSR, IST, ICSME, ICDM and ESEM. His current research interests include software engineering, data mining and computer-supported cooperative work.

Georgios Gousios is a research scientist at Facebook and an associate professor at the Delft University of Technology. His work is on applying techniques from the domains of static analysis, machine learning and software analytics to improve developer productivity and operational efficiency.

Ayushi Rastogi is an Assistant Professor in the Faculty of Science and Engineering at the University of Groningen, the Netherlands. Her research interests include software analytics, empirical software engineering, and mining software repositories. She studies human and social aspects of software engineering for improving developer productivity and promoting diversity and inclusion.

TABLE 10: Factors related to pull request decisions in related articles.
First column lists factors in alphabet ascending order in each class, the rest columns list related articles and the result of each factor.
Horizontal Line in the middle of shape (⊖) means the factor is removed when building models because of multicollinearity.
Filling: Filled (●) means *significance is reported* and unfilled (○) means *significance is not reported because of not using statistical model or inconsistent conclusions*.
Size of filled shape: Big shape (●) shows *statistically significant* relation and small shape (•) *statistically insignificant* with 95% confidence threshold.
Color: Blue 🔵 means a *positive relation* (meaning increase in the chances of pull request acceptance), red 🔴 means a *negative relation*, gray ⚫ means *uncertain relation* because of not using statistical model or nonlinear conclusion.

	[1]	[15]	[10]	[57]	[2]	[24]	[12]	[54]	[100]	[88]	[94]	[52]	[53]	[61]	[58]	[56]	[101]	[91]	[93]	[92]	[59]	[60]	[102]	[103]	[87]	[104]
Developer Characteristics																										
account_creation_days							🔴											○								
agree_diff					🔵	🔵																				
cons_diff					🔵	🔵																				
contrib_affiliation				○						🔵																
contrib_agree					🔹	🔹																				
contrib_cons					🔵	🔵																				
contrib_country							○																			
contrib_extra					🔴	🔴																				
contrib_first_emo						🔵																				
contrib_follow_integrator		🔵			🔵	🔵	🔵																			
contrib_gender																									🔵	
contrib_neur					🔸	🔸																				
contrib_open					🔵	🔵																				
contrib_rate_author											○															
core_member		🔵	🔵		🔵	🔵	🔹				○	○							○	○	🔵	🔵	○			
extra_diff					🔵	🔵																				
first_pr												○	○									🔴		○		
first_response_time			🔴																							
followers		🔵	🔵		•	•	🔵																			
inte_affiliation				○																						
inte_agree					🔹	🔹																				
inte_cons					🔵	🔵																				
inte_extra					🔵	🔵																				
inte_first_emo						🔵																				
inte_neur					🔵	🔵																				
inte_open					🔵	🔵																				
neur_diff					🔵	🔵																				
open_diff					🔹	🔹																				
perc_contrib_neg_emo						🔴																				
perc_contrib_pos_emo						🔵																				
perc_inte_neg_emo						🔴																				
perc_inte_pos_emo						🔵																				

TABLE 10: Factors related to pull request decisions in related articles.
First column lists factors in alphabet ascending order in each class, the rest columns list related articles and the result of each factor.
Horizontal Line in the middle of shape (⊖) means the factor is removed when building models because of multicollinearity.
Filling: Filled (●) means *significance is reported* and unfilled (○) means *significance is not reported because of not using statistical model or inconsistent conclusions*.
Size of filled shape: Big shape (●) shows *statistically significant* relation and small shape (•) *statistically insignificant* with 95% confidence threshold.
Color: Blue 🔵 means a *positive relation* (meaning increase in the chances of pull request acceptance), red 🔴 means a *negative relation*, gray ● means *uncertain relation* because of not using statistical model or nonlinear conclusion.

	[1]	[15]	[10]	[57]	[2]	[24]	[12]	[54]	[100]	[88]	[94]	[52]	[53]	[61]	[58]	[56]	[101]	[91]	[93]	[92]	[59]	[60]	[102]	[103]	[87]	[104]
prev_pullreqs	○			●			🔵			🔵						○										
prior_interaction		🔵			🔵	🔵																				
prior_review_num				•																						
requester_succ_rate	○						🔵	○			○															
same_affiliation				○																						
same_country							🔵																			
social_strength			🔵																							
Project Characteristics																										
asserts_per_kloc	○						⊖																			
fork_num		⊖						○										○								
integrator_availability			🔴																							
language												○						○								
open_issue_num								○																		
open_pr_num			🔴	○																						
perc_external_contribs	○						🔴																			
project_age		🔴	🔵		🔴	🔴	🔴											○								
pr_succ_rate								○																		
pushed_delta								○																		
sloc	○						🔴																			
stars		🔴			🔴	🔴	🔴	○																		
team_size	○	🔴	🔹		🔸	🔸	⊖	○										○								
test_cases_per_kloc	⊖						⊖																			
test_lines_per_kloc	○						🔵																			
Pull Request Characteristics																										
at_tag			🔹																							
bug_fix											○				○											
churn_addition			🔴																							
churn_deletion			🔵								○															
ci_build_num											○															
ci_exists																										🔴
ci_failed_perc											○															
ci_first_build_status											🔵															
ci_last_build_status											🔵															

TABLE 10: Factors related to pull request decisions in related articles.
First column lists factors in alphabet ascending order in each class, the rest columns list related articles and the result of each factor.
Horizontal Line in the middle of shape (⊖) means the factor is removed when building models because of multicollinearity.
Filling: Filled (●) means *significance is reported* and unfilled (○) means *significance is not reported because of not using statistical model or inconsistent conclusions*.
Size of filled shape: Big shape (●) shows *statistically significant* relation and small shape (•) *statistically insignificant* with 95% confidence threshold.
Color: Blue 🔵 means a *positive relation* (meaning increase in the chances of pull request acceptance), red 🔴 means a *negative relation*, gray ● means *uncertain relation* because of not using statistical model or nonlinear conclusion.

	[1]	[15]	[10]	[57]	[2]	[24]	[12]	[54]	[100]	[88]	[94]	[52]	[53]	[61]	[58]	[56]	[101]	[91]	[93]	[92]	[59]	[60]	[102]	[103]	[87]	[104]
ci_latency			🔵																							
ci_test_passed			🔵																							
comment_conflict	○																									
commits_on_files_touched	○		🔵																							
contrib_comment														○												
description_length			🔴																							
files_added												○														
files_changed	○	🔴			⊖	⊖	•	○		⊖	○	○	○													
files_deleted												○														
friday_effect			•																							
has_comments													○	○												
has_exchange														○												
hash_tag	○		•																							
has_participants														○												
inte_comment														○												
lifetime_minutes											○	○				○										
num_code_comments										⊖	○															
num_code_comments_con										⊖																
num_comments	○	🔴	🔴		🔴	🔴	🔴			⊖	○															
num_comments_con										•																
num_commits	○		🔵					○		⊖	○	○	○													
num_participants	○									🔴																
other_comment														○												
part_num_code										🔵																
perc_neg_emotion						🔴																				
perc_pos_emotion						🔵																				
reopen_or_not																	○									
core_comment														○												
src_churn	○	🔴		•	🔴	🔴	🔴	○	○	🔴																
test_churn	○																									
test_inclusion		🔵	🔵		🔵	🔵	🔵				○															

APPENDIX A
RESULTS OF DIFFERENT CONTEXTS

TABLE 11: Results in different contexts

	Dependent variable: merged_or_not=1											
	same user or not		has comments or not		ci exists or not		different team sizes			different periods		
	yes	no	yes	no	yes	no	small	mid	large	before 2016.6	2016.6-2018.6	after 2018.6
(Intercept)	10.4***	34.4***	13.1***	42.4***	20.4***	13.3***	24.9***	20.7***	15.9***	6.9***	16.6***	7.1***
prior_review_num	2.86*** [31.17]	0.98*** [0.04]	1.51*** [14.40]	1.91*** [22.12]	1.53*** [13.76]	1.53*** [11.95]	1.59*** [11.27]	1.41*** [8.80]	1.57*** [18.89]	1.30*** [6.25]	1.63*** [14.12]	1.72*** [17.00]
lifetime_minutes	0.66*** [19.09]	0.52*** [43.67]	0.61*** [29.79]	0.70*** [12.97]	0.60*** [21.52]	0.61*** [20.78]	0.54*** [24.47]	0.61*** [20.16]	0.67*** [16.55]	0.65*** [20.26]	0.57*** [20.83]	0.62*** [12.69]
core_member	1.26*** [9.36]	1.13*** [1.31]	1.29*** [5.55]	1.33*** [5.85]	1.30*** [5.28]	1.26*** [4.66]	1.42*** [5.90]	1.28*** [5.24]	1.19*** [3.24]	1.27*** [5.99]	1.34*** [4.94]	1.29*** [3.14]
prev_pullreqs	0.61*** [5.85]	1.17*** [1.24]	1.13*** [0.69]	0.95*** [0.09]	1.14*** [0.64]	1.12*** [0.56]	1.21*** [0.79]	1.21*** [1.37]	1.13*** [0.84]	1.08*** [0.29]	1.26*** [1.59]	1.24*** [1.32]
num_commits	1.23*** [3.78]	1.46*** [10.43]	1.49*** [11.33]	0.98** [0.03]	1.32*** [4.85]	1.25*** [3.57]	1.36*** [4.64]	1.31*** [4.53]	1.26*** [4.36]	1.18*** [2.31]	1.32*** [4.13]	1.36*** [4.84]
hash_tag	1.14*** [2.52]	1.10*** [1.34]	1.14*** [2.27]	1.09*** [0.65]	1.12*** [1.46]	1.10*** [1.14]	1.13*** [1.37]	1.11*** [1.16]	1.11*** [1.38]	1.04*** [0.22]	1.13*** [1.44]	1.13*** [1.30]
first_pr	0.91*** [1.82]	0.96*** [0.30]	0.95*** [0.32]	0.95*** [0.31]	0.94*** [0.45]	0.96*** [0.27]	0.95*** [0.23]	0.97*** [0.15]	0.96*** [0.31]	0.95*** [0.50]	0.96*** [0.20]	0.94*** [0.32]
files_added	0.88*** [1.57]	0.95*** [0.30]	0.90*** [0.83]	0.92*** [0.57]	0.90*** [0.90]	0.92*** [0.53]	0.90*** [0.52]	0.89*** [1.04]	0.90*** [0.94]	0.97*** [0.11]	0.88*** [0.95]	0.86*** [1.34]
reopen_or_not	0.93*** [1.52]	0.99· [0.01]	0.97*** [0.17]	0.92*** [2.39]	0.96*** [0.40]	0.97*** [0.24]	0.97*** [0.23]	0.96*** [0.48]	0.96*** [0.43]	0.99*** [0.03]	0.97*** [0.18]	0.93*** [1.06]
open_pr_num	0.73*** [1.37]	1.06*** [0.05]	0.92*** [0.09]	0.60*** [3.07]	0.83*** [0.33]	0.76*** [0.93]	0.81*** [1.26]	0.83*** [0.81]	0.98 [0.01]	0.77*** [1.34]	0.72*** [0.53]	0.75*** [0.28]
contrib_open	1.13*** [1.28]	1.06*** [0.38]	1.05*** [0.25]	1.09*** [0.24]	1.05*** [0.15]	1.07*** [0.34]	1.05*** [0.12]	1.12*** [0.70]	1.05*** [0.25]	1.07*** [0.46]	1.07*** [0.27]	1.02** [0.02]
description_length	1.06*** [0.45]	1.02*** [0.05]	1.01*** [0.03]	1.12*** [1.11]	1.04*** [0.17]	1.04*** [0.13]	1.03*** [0.08]	1.03*** [0.07]	1.05*** [0.29]	0.99* [0.01]	1.06*** [0.26]	1.07*** [0.36]
commits_on_files_touched	1.06*** [0.41]	1.11*** [1.16]	1.05*** [0.23]	1.18*** [1.86]	1.06*** [0.24]	1.13*** [1.39]	1.10*** [0.61]	1.12*** [0.90]	1.05*** [0.20]	1.30*** [7.25]	0.99 [0.00]	0.99 [0.00]
stars	0.83*** [0.40]	0.94*** [0.05]	0.83*** [0.39]	0.83*** [0.37]	0.85*** [0.24]	0.75*** [0.81]	0.81*** [0.69]	0.89*** [0.17]	1.05** [0.01]	0.83*** [0.45]	0.72*** [0.44]	0.71*** [0.50]
project_age	1.08*** [0.19]	1.24*** [1.37]	1.08*** [0.16]	1.16*** [0.53]	1.10*** [0.20]	1.15*** [0.50]	1.04*** [0.05]	1.08*** [0.15]	0.98* [0.01]	0.89*** [0.40]	1.69*** [2.27]	3.80*** [4.81]
files_changed	0.94*** [0.18]	0.90*** [0.54]	0.95*** [0.11]	0.90*** [0.43]	0.94*** [0.15]	0.90*** [0.47]	0.93*** [0.16]	0.91*** [0.33]	0.93*** [0.21]	0.86*** [1.13]	0.95*** [0.09]	0.94*** [0.13]
test_churn	1.05*** [0.15]	1.09*** [0.52]	1.08*** [0.39]	0.99 [0.01]	1.07*** [0.27]	1.05*** [0.15]	1.11*** [0.45]	1.07*** [0.25]	1.04*** [0.11]	1.07*** [0.34]	1.10*** [0.40]	1.08*** [0.23]
account_creation_days	1.03*** [0.13]	1.11*** [1.70]	1.05*** [0.28]	1.17*** [2.26]	1.08*** [0.58]	1.04*** [0.22]	1.06*** [0.33]	1.11*** [1.04]	1.02*** [0.06]	0.99* [0.01]	1.02*** [0.02]	1.03*** [0.06]
team_size	0.94*** [0.08]	1.09*** [0.19]	1.06*** [0.07]	0.92*** [0.14]	1.02* [0.00]	0.96* [0.02]	1.06*** [0.30]	1.00 [0.00]	0.96*** [0.04]	0.88*** [0.37]	1.09*** [0.07]	0.85*** [0.16]
pushed_delta	1.02*** [0.07]	1.06*** [0.46]	1.03*** [0.15]	1.06*** [0.38]	1.04*** [0.17]	1.04*** [0.18]	1.06*** [0.31]	1.05*** [0.25]	1.02*** [0.08]	1.04*** [0.26]	1.03*** [0.05]	1.04*** [0.12]
integrator_availability	0.98*** [0.07]	1.03*** [0.15]	1.00 [0.00]	0.99 [0.01]	0.99*** [0.03]	1.01* [0.03]	1.03*** [0.07]	1.01 [0.00]	0.97*** [0.14]	1.00 [0.00]	0.97*** [0.06]	0.98** [0.03]
test_inclusion	1.03*** [0.06]	1.00 [0.00]	1.02*** [0.02]	1.02* [0.02]	1.03*** [0.05]	0.97*** [0.07]	1.00 [0.00]	1.03*** [0.05]	1.01** [0.02]	1.00 [0.00]	1.01* [0.01]	1.01 [0.00]
contrib_neur	1.02*** [0.04]	1.05*** [0.27]	1.01* [0.01]	1.04*** [0.05]	1.01· [0.00]	1.03** [0.05]	1.00 [0.00]	1.07*** [0.25]	0.99** [0.02]	1.02*** [0.03]	1.02*** [0.03]	0.99 [0.00]
contrib_cons	1.02*** [0.04]	1.04*** [0.17]	1.05*** [0.20]	0.94*** [0.12]	1.03*** [0.05]	1.02** [0.04]	1.05*** [0.12]	1.03*** [0.04]	1.03*** [0.07]	1.01** [0.02]	1.03*** [0.05]	1.12*** [0.54]
contrib_gender	0.98*** [0.04]	0.97*** [0.13]	0.97*** [0.10]	0.99 [0.00]	0.98*** [0.06]	0.98** [0.04]	0.97*** [0.08]	0.97*** [0.09]	0.99** [0.02]	0.99 [0.01]	0.97*** [0.10]	1.01 [0.00]
pr_succ_rate	0.98*** [0.04]	0.99* [0.01]	0.98*** [0.03]	0.97*** [0.05]	0.97*** [0.06]	1.02** [0.04]	0.94*** [0.25]	0.99 [0.01]	0.96*** [0.10]	0.97*** [0.14]	1.00 [0.00]	1.11*** [0.13]
contrib_agree	0.98*** [0.04]	0.99* [0.01]	0.98*** [0.03]	0.96** [0.04]	0.99** [0.01]	0.97*** [0.07]	0.96*** [0.09]	0.99 [0.00]	0.98*** [0.02]	0.97*** [0.05]	0.96*** [0.08]	1.00 [0.00]
friday_effect	1.01*** [0.03]	1.01* [0.01]	1.01*** [0.03]	1.01 [0.01]	1.01** [0.01]	1.02** [0.06]	1.01 [0.00]	1.01*** [0.02]	1.01*** [0.02]	1.02*** [0.05]	1.01* [0.01]	1.01 [0.00]
contrib_extra	0.98*** [0.03]	0.99* [0.01]	0.98*** [0.05]	1.08*** [0.18]	0.99· [0.00]	1.00 [0.00]	0.98*** [0.02]	0.99 [0.00]	1.00 [0.00]	0.99** [0.02]	0.97*** [0.06]	0.95*** [0.11]
src_churn	0.99** [0.02]	1.01 [0.01]	1.05*** [0.15]	0.90*** [0.65]	1.00 [0.00]	1.00 [0.00]	1.05*** [0.10]	1.00 [0.00]	0.96*** [0.10]	0.98*** [0.05]	1.01 [0.00]	1.00 [0.00]
open_issue_num	0.96** [0.02]	1.15*** [0.22]	0.99 [0.00]	1.12*** [0.13]	1.07*** [0.04]	0.90*** [0.15]	1.02 [0.01]	1.03** [0.01]	0.96 [0.01]	0.94*** [0.04]	1.07** [0.02]	1.05 [0.01]
sloc	1.02* [0.01]	1.04*** [0.03]	1.04*** [0.04]	0.96** [0.04]	0.99 [0.00]	1.00 [0.00]	1.00 [0.00]	0.98* [0.01]	1.07*** [0.08]	1.13*** [0.33]	0.92*** [0.07]	0.94*** [0.05]
files_deleted	0.99* [0.01]	0.98*** [0.08]	0.96*** [0.18]	1.05*** [0.28]	0.98*** [0.06]	1.00 [0.00]	0.97*** [0.06]	0.98*** [0.04]	0.99* [0.01]	1.02*** [0.05]	0.98*** [0.04]	0.97*** [0.07]
test_lines_per_kloc	0.98* [0.01]	0.99 [0.00]	1.03*** [0.02]	0.93*** [0.14]	1.01 [0.00]	0.92*** [0.17]	0.98** [0.01]	1.02* [0.01]	0.98 [0.01]	1.09*** [0.22]	0.95*** [0.05]	0.94*** [0.06]
followers	1.00 [0.00]	0.96*** [0.18]	1.03*** [0.06]	1.05*** [0.12]	1.04*** [0.12]	1.02** [0.04]	1.04*** [0.07]	1.01 [0.00]	1.07*** [0.39]	1.14*** [1.40]	1.06*** [0.16]	1.04*** [0.07]
has_comments	0.68*** [10.43]	0.50*** [27.35]	-	-	0.65*** [10.11]	0.52*** [24.50]	0.57*** [13.21]	0.64*** [10.27]	0.66*** [11.93]	0.63*** [14.94]	0.62*** [9.60]	0.55*** [13.78]
other_comment	1.24*** [5.84]	1.18*** [3.60]	-	-	1.23*** [4.18]	1.08*** [0.71]	1.31*** [6.33]	1.23*** [4.15]	1.14*** [1.92]	1.14*** [2.25]	1.22*** [3.07]	1.25*** [2.84]
ci_exists	1.11*** [0.95]	1.19*** [2.52]	1.14*** [1.64]	1.16*** [1.16]	-	-	1.13*** [0.76]	1.12*** [0.79]	1.09*** [0.66]	1.08*** [0.64]	1.12*** [0.44]	1.17*** [0.66]
num_comments	1.12*** [0.88]	0.96*** [0.11]	-	-	1.01· [0.00]	1.18*** [1.38]	0.91*** [0.37]	1.01* [0.01]	1.13*** [0.79]	1.08*** [0.31]	1.02** [0.01]	1.07*** [0.16]
comment_conflict	1.01*** [0.03]	1.01*** [0.04]	-	-	1.01* [0.01]	1.02** [0.06]	1.01** [0.02]	1.00 [0.00]	1.01*** [0.02]	1.01*** [0.04]	1.00 [0.00]	1.02*** [0.04]
same_user	-	-	0.56*** [29.27]	0.42*** [41.50]	0.51*** [32.75]	0.59*** [23.27]	0.49*** [24.47]	0.49*** [36.03]	0.55*** [32.58]	0.57*** [31.20]	0.46*** [33.20]	0.46*** [28.79]
inte_open	-	-	1.10*** [0.61]	1.01 [0.01]	1.10*** [0.51]	1.04*** [0.07]	0.98* [0.01]	0.92*** [0.25]	1.18*** [2.12]	0.97*** [0.05]	1.03*** [0.04]	1.25*** [2.21]
inte_neur	-	-	1.03*** [0.05]	0.99 [0.01]	1.06*** [0.15]	0.93*** [0.24]	0.96*** [0.05]	0.93*** [0.18]	1.10*** [0.58]	0.96*** [0.11]	0.98* [0.01]	1.13*** [0.57]
inte_agree	-	-	1.00 [0.00]	1.05*** [0.06]	1.00 [0.00]	1.07*** [0.14]	1.03*** [0.02]	1.01 [0.00]	0.98*** [0.03]	1.02** [0.02]	1.02** [0.01]	0.92*** [0.16]
inte_extra	-	-	1.01· [0.00]	0.97* [0.02]	1.02*** [0.01]	0.97* [0.03]	1.06*** [0.11]	1.07*** [0.21]	0.95*** [0.14]	1.02*** [0.03]	1.04*** [0.06]	1.02 [0.02]
inte_cons	-	-	1.00 [0.00]	1.05*** [0.06]	1.01 [0.00]	0.98· [0.02]	1.01 [0.00]	1.00 [0.00]	0.99** [0.01]	1.01 [0.01]	1.03*** [0.02]	0.97** [0.03]
Observations	950,985	1,010,937	1,152,714	809,208	1,611,277	350,645	601,460	703,396	701,900	512,707	585,401	274,121
AUC_train	0.862	0.874	0.837	0.872	0.843	0.884	0.877	0.843	0.837	0.850	0.867	0.879