An Empirical Study on the Latency of Time to First Response In GitHub Pull Requests

Kazi Amit Hasan
School of Computing, Queen's University, Canada

***Abstract*—Modern open-source software development has become part of a collaborative and distributed environment. This makes it possible for development teams to get help from external developers who want to work on a project for different reasons. Developers contribute to these projects with the help of the pull request paradigm. A pull request passes through multiple stages before getting its final decision. Delays during the lifetime of pull requests can slow down the overall development of software. In order to understand the underlying dynamics of the long response time of pull requests, we analyzed a dataset consisting of 3,347,937 closed pull requests and selected ten popular and mature open-source GitHub projects. We identified several critical concerns in analyzing the time to first response in pull requests. Also, we developed an approach to identify and remove the bot-generated first responses and fixed the time to first response value of an existing benchmark dataset. Our investigation revealed that the first response time of a pull request is a noisy feature, e.g., it can be impacted by different types of bots. Some pull requests' lifetime can be shortened if they get their first response early. Also, we find that complex pull requests, newly created projects, and a small team have a higher probability of causing a long waiting time for the first responses in pull requests.**

I. Introduction

Modern open-source software development has been embedded in a collaborative and distributed environment, allowing development teams to receive contributions from external developers interested in collaborating on a particular project for various reasons. They can contribute to the development of an open-source project by coding, documenting, testing, or doing any other essential task.

Pull requests are a new paradigm for systematizing contributions to open-source projects. According to this paradigm, external contributors can independently modify project artifacts and subsequently request that these changes be merged back into the main repository. A pull request (PR) including either bug fixes or new features may be accepted and merged into the project's main repository following assessment by a core team member, or it may be rejected [1].

Pull request latency is a major concern. Understanding the pull request latency makes the development process more predictable for reviewers and facilitates the planning and decision-making of managers. For contributors, the latency prediction can notify developers of the remaining time and expedite the fulfillment of pull requests, hence reducing contributor abandonment [2].

A pull request passes through multiple stages before getting its final decision, e.g., waiting for the reviewer stage, review stage, and integration stage. Furthermore, it could be delayed in any of its stages, affecting the entire lifetime of the pull request. For example, a pull request can face delays in its early stages. It might take days, weeks, or even months to get the first response from the reviewer. Similarly, for some PRs, it might take significantly longer to complete the review process. There might be various factors associated with it. Most existing PR latency analysis studies focus on the whole lifetime of a PR, rather than a particular stage. Therefore, important questions like the factors related to each stage's latency in a PR's life remain unknown. To fill the gap, in this study, we perform a fine-grained analysis on PR latency, focusing on the phase between a PR that has been reported and the first response made, i.e., the *time-to-first-response* stage. There are several reasons for studying the time to first response in pull requests. Firstly, studies have found that time to first response is an important feature for pull request decision and latency prediction [3]. Secondly, tools like Nudge [4] proposed by Microsoft would warn people if they did not respond in a predicted time. Though Nudge is trained from large-scale PRs based on their lifetime, it only warns one time for each PR. In other words, Nudge mainly uses the predicted whole lifetime of PR to speed up the first response. Lastly, new pull-based development practices, especially bots, may impact the analysis of the PR latency in different stages, but its impact is still unknown. Our fine-grained analysis of the first response stage would shed light on future research leveraging GitHub pull requests to explain and optimize the pull-based development process.

Specifically, we aim to answer the following three research questions in this report:

RQ1: ***What are the characteristics of time-to-first-response in pull requests on GitHub?*** This research question attempts to report the characteristics of first response time and explain certain extreme cases, i.e., PRs that received their first response in an extremely short or long period in open source GitHub projects.

RQ2: ***What's the impact of pull request bots on time-to-first-response?*** The goal of this research question is to find out how bot-generated responses impact the latency analysis of the first response time. Since bot identification requires manual verification, we select ten large software projects on GitHub. Our hypothesis is that "time-to-first-response" is a noisy feature in existing PR benchmarks. Many bot-generated first responses should be considered to avoid impacting the analysis of the actual latency of PR.

RQ3: ***What are the characteristics of long time-to-first-response PRs?*** This research question aims to analyze the

factors that can explain time-to-first-response within an OSS project. This research question also investigates the possibility of predicting time-to-first-response leveraging PR features.

To answer the above research questions, we leverage a benchmark pull request dataset provided by Zhang et al. [2] for their large-scale analysis of pull request latency. For RQ1, we report the statistics of time-to-first-response on all the pull requests extracted from 10,000+ GitHub projects and sampled some PRs for extreme case analysis. For RQ2 and RQ3, we selected ten projects based on their popularity and number of pull requests that have comments[1], following literature work [3], [5].

In summary, we make the following key contributions in this project:

1) We identify several issues in analyzing time-to-first-response in GitHub pull requests.
2) We fix the time-to-first-response related feature values in the largest benchmark for pull request latency analysis.
3) We conduct the first empirical to understand factors that explain time-to-first-response and the potentiality to predict it within an OSS project.

The rest of this report is organized as follows. Section II presents the related works and Section III represents the dataset that we have used for this project. Section IV,V, VI presents the methodology and results for three research questions. We discuss the threats to validity in Section VII and the implications of the findings for researchers and developers in Section VIII. Finally, Section IX concludes the study and mentions future work.

II. Related Work

A. Studies on Pull Request Latency

Pull requests can be delayed at different stages of their lifetime. Most of the pull request latency research focuses on the whole lifetime of PR. There has been limited focus on the stage-level delays in pull requests.

Pull requests can slow down pull-based development due to various factors and inactive engagement between the author and reviewer. Zhang et al. [2] collected features related to pull request latency through a literature review approach. Also, they evaluated their relative importance in five scenarios and six different contexts using a mixed-effect linear regression model. The authors also identified several effective insights. For example, process-related factors significantly gain more importance after the closure of a pull request. They have also used time-to-first-response as one of the important features while conducting their research.

Kudrjavets et al. [6] define time-to-first-response as the time from the publication of a code change until the first acceptance, comment, inline comment (comment on specific code fragments), or rejection by a person (excludes bots) other than the author of the code. They also mentioned that the literature classifies review time into three categories: (i) review delay (time from the first review request submission to the first reviewer feedback), (ii) review duration (time from the first review request submission to the review conclusion), and (iii) review speed (rate of SLOC reviewed per hour). They considered time-to-first-response as non-productive time. The authors applied stratified sampling considering the number of code reviews in each project where the time-to-first-response exceeds a typical eight-hour workday.

Recently, Microsoft developed a tool named Nudge [4] to support the management of pull requests leveraging the prediction of PR lifetime. Nudge would only warn people if they responded in a predicted time. However, though Nudge is trained from large-scale PRs based on their lifetime, it only warns one time for each PR. This means they mostly use the predicted whole lifetime of PR to speed up the first response.

B. Other Studies on Pull Requests

Golzadeh et al. [7] developed a tool named 'BoDeGha' which can identify bots from pull requests by analyzing pull requests and issue comments. The tool accepts a GitHub repository's name and requires a GitHub API key to compute its result in three steps. The initial step is to utilize the GitHub GraphQL API to retrieve all pull request comments and issue comments from the specified repository. This action produces a list of commenters and their accompanying pull request and issue comments. The second phase involves computing the necessary features for the classification model: the number of comments, empty comments, comment patterns, and inequality between the number of comments inside patterns. The final phase applies the classification model to the repository data and outputs the classification model's predictions regarding the bots. In this project, we leverage BoDeGha to collect the candidate bots. However, we find that many of its reported bots are human, including core members of the developer team (ref. V).

Zhang et al. [8] investigated different factors influencing pull request decisions. Their 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. They also reported 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.

III. Dataset

A. Benchmark Dataset

We consider the benchmark data provided by Zhang et al. [2]. The dataset comprises 45 features extracted from 3,347,937 closed pull requests (meaning a decision has been made) in 11,230 OSS projects on GitHub. As our study focuses on the time-to-first-response, we exclude all pull requests that do not have any comments. Finally, the filtered data contains 1,613,224 pull requests from 10,374 projects.

[1]Pull requests without comments do not have the first response.

Table I: 18 factors which may influence pull request's time-to-first-reponse on GitHub.

Dimension	Feature	Description
Pull Request	ci_exists	uses CI? yes/no
	ci_latency	minutes from pull request creation to the first CI build finish time
	description_length	length of pull request description
	churn_addition	# of added lines of code
	files_changed	# of files touched
Review	num_participants	# of participants in pull request comments
	lifetime_minutes	whole lifetime minutes of a pull request
	part_num_code	# of participants in pull request and commit comments
	merged_or_not	merged in the end? yes/no
Project	team_size	# of active core team members in the last three months
	project_age	# of months from project to pull request creation
	open_pr_num	# of open pull requests
	open_issue_num	# of open issues
Developer	prior_interaction	# of interactions with a project in the last three months
	requester_succ_rate	past pull request success rate
	same_user	same contributor and integrator? yes/no
	core_member	core member? yes/no
	first_pr	first pull request? yes/no

B. Features

Table I represents the list of factors which may influence pull request's time-to-first-response. We are considering features from four dimensions, i.e., pull request, review, project, and developer.

1) Pull Request: From the pull request dimension, we are considering whether the pull request uses CI tool or not, the number of minutes from pull request creation to the first CI build finishes, length of pull request descriptions, number of added lines in the code and number of files touched.

2) Review: From the pull request review perspective, we are considering the number of participants in pull request comments, the whole lifetime minutes of a pull request, number of participants in pull request and commit messages and whether the pull request merged or not.

3) Project: From the project dimension, we are considering the number of active members in the last three months, number of months from project to pull request creation, number of opened pull requests, number of open issues.

4) Developer: From the developer dimension, we are considering number of interactions with a project in the last three months, past pull request success rate, whether the developer was same contributor and integrator or not, whether the developer is a core member or not and whether the pull request is a first pull request or not.

C. Studied Projects

In order to answer the second and third research questions, we selected specific projects based on literature work [3], [5]. We selected projects based on rich history of pull request data and popularity. To be specific, we selected 10 projects with more than 1,000 closed PRs and popular (with more than 100 stars).

While selecting the projects, we filtered out the pull requests that didn't have any comments. After that, we sorted all projects by the number of valid pull requests. Also, we picked software tools and excluded projects that lacked significant history on GitHub or changed their repository. A full list of projects is shown in Table II.

IV. RQ1: What are the characteristics of time-to-first-response in pull requests on GitHub?

This research question aims to identify the characteristics of first response time and explain some extreme cases (response in 10 minutes, response in more than 1 month). For this research question, we are considering all PRs from 10,374 projects.

In order to get a better understanding of the characteristics of first response time, we will investigate the distribution of first responses in the benchmark dataset. After that, we will calculate the percentage of pull requests that belong to each category based on first-response-time latency and remaining time latency. This will assist in determining whether or not the first response time is truly delaying the processing of pull requests.

A. Approach

In order to answer this research question, we analyzed 10,374 projects collected in Section III. We categorize the lifetime of a PR into two stages, i.e., the *time-to-first-response stage* and the *after-first-response stage*. For each stage, we define the following four intervals to characterize the *time-to-first-response* and the *time-after-first-response*:

- **Within 1 day:** This category includes all pull requests that were responded within one day.
- **1 day to 1 week:** This category includes all pull requests that were responded within one day to 1 week.
- **1 week to 1 month:** This category includes all pull requests that were responded within 1 week to 1 month.
- **More than 1 month:** This category includes all pull requests that were responded extremely late (more than 1 month).

We assign a time duration category label for each PR in our dataset and analyze its distribution leveraging the plotly

Table II: Considered 10 OSS projects on GitHub.

Project	# PR	Creation Date	# Stars	Domain	Programming Language
ansible/ansible	23,451	2012-03-06	39,616	Automation Platform	Python
apache/spark	22,874	2014-02-25	23,331	Data Analytics Engine	Scala, Python
kubernetes/kubernetes	20,494	2014-06-06	45,686	Container Orchestration	Go
rails/rails	15,772	2008-04-11	48,587	Web Application Platform	Ruby
saltstack/salt	15,465	2011-02-20	10,869	Automation Platform	Python
odoo/odoo	13,357	2014-05-13	14,666	Business Apps	JavaScript, Python
nodejs/node	11,891	2014-11-26	67,208	JavaScript Runtime Environment	JavaScript
akka/akka	7,421	2009-02-16	10,717	Application Development Toolkit	Scala
discourse/discourse	6,747	2013-01-12	30,508	Community Discussion Platform	Ruby
hazelcast/hazelcast	6,521	2012-03-21	3,358	Distributed Computation and Storage Platform	Java

package [9]. To understand how many PRs could be sped up by reducing their time-to-first-response, we analyze the distribution of time-after-first-response for each type of PRs defined by their time-to-first-response. In other words, we report how many pull requests only get delayed in the time-to-first-response stage and then quickly close after a short review process.

B. Results

Figure 1 represents the number of pull requests in each category in the First Time Response stage. We find that:

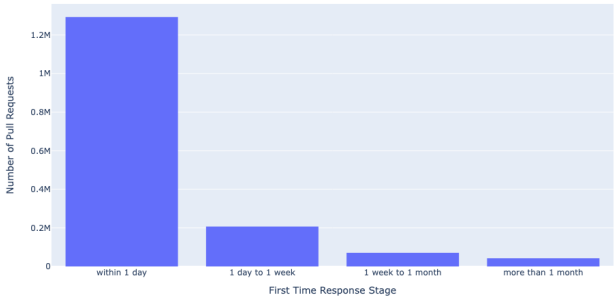

Figure 1: Number of pull requests in each category based on time-to-first-response.

- **Most of the pull requests receive their first response within less than a day.** Around 80.2% and 93% of the PRs received their first response within one day and one week. Some PRs (7%) waited more than one week for their first response. Even there are PRs that had to wait for a month to get their first response.
- **Good amount of PRs received their first response in workday hours.** According to [6], 8 hours is commonly considered workday hours. Our analysis found that 68.4% of pull requests received the first response within 8 hours.
- **A significant amount of pull requests receive their first response within 10 minutes**. We analyzed further deeper in the first category (within a day) and got some interesting results. Our findings suggested that, among all PRs receiving the first response within one day, 39.8% of them (514,380 pull requests) received the first response in 10 minutes. We then conducted a manual study by randomly sampling 50 PRs with time-to-first-response less than or equal to 10 minutes to understand why those PRs responded so fast, and we found:
  - **35/50 (70%) of the first response is generated by a bot**. We observe CLA (Contributor Licence Agreement) bots, review supporting bots, and reviewable services[2].
  - If we do not consider bot response, the first response from the reviewer could take more than one week.
  - There are 4/50 (8%) cases we can not find any response in the given time, i.e., the collected time for the first response is wrong
  - 10/50 (20%) cases are real reviewers' first responses. One even responded in one minute.

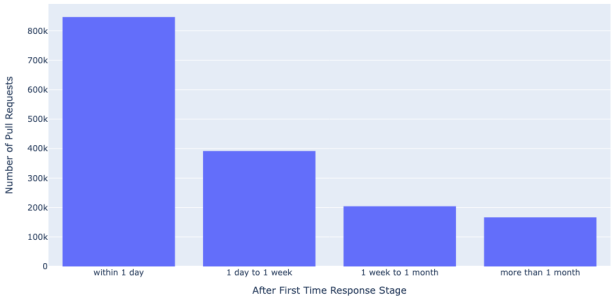

Figure 2: Number of pull requests in each category based on time-after-first-response.

Furthermore, we analyzed the *time-after-first-response* and found that 52.7% and 77% of PRs complete the rest of their lifetime within 1 day and 1 week respectively. 23% PRs take more than 1 week to complete the process. In order to better understand how this related to first time response, we generated Figure 3 and found that:

- **More than 50% PRs' receive their decision within a day if they receive the first response early.** To be more specific, our analysis suggested that PRs receiving the first response within one day, 56% of them take less than one day to close, and 24% take more than one day but less than one week to close which is faster than other

[2]https://reviewable.io/

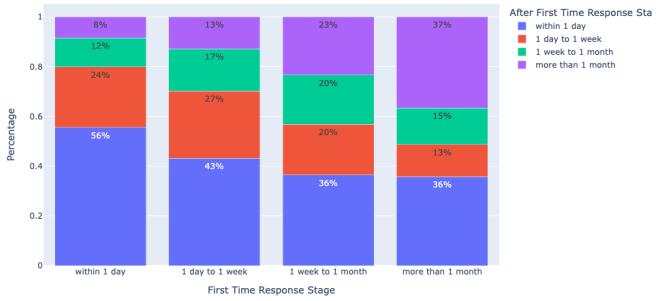

Figure 3: Percentages of after first response time categories in each time to first response category.

PRs with longer first response time. This aligns with [2]'s findings.

- **Despite receiving their first response after weeks, some PRs were quickly reviewed and merged.** Our analysis revealed that, 36% of the PRs take more than one week to receive the first response and take less than one day to complete their life. For those PRs, if their first response can arrive earlier, they might have a significantly shorter lifetime. To understand why such cases happen, we conducted a manual study by randomly sampling 50 PRs in this group and find that
  - 37/50 (74%) the reviewer responded after 1 week but quickly merged the PR after reviewing it. Even in some cases, the authors had to grab the attention of reviewer by mentioning about their PRs' in other PR comments. **If first response can arrive earlier, they might have a significantly shorter lifetime.**
  - 10/50 (20%) of pull requests were incomplete while their submission. The author added more information separately after the first submission of pull request.
  - Apart from these, 1/50 (2%) of pull requests were unknown (written in another language), 1/50 pull requests, the author and reviewers were waiting for an existing conflict to be fixed. Lastly, 1/50 pull request reported wrong first response. This also raises concerns about the concerns of correct measurements of time-to-first-response.

The time-to-first-response reported in the benchmark is not clean, i.e., the benchmark considered first responses generated by bots rather than humans. Some pull requests' lifetimes can be shortened if they get their first response early.

V. RQ2: What's the impact of Pull Request Bots on time-to-first-response?

RQ1 shows that the benchmark data did not have the correct time-to-first-response values. In RQ2, we develop an approach to identify the bots from the selected projects and analyze the statistics of time-to-first-response before and after considering bots.

A. Approach

1) Additional Data Collection: In our benchmark dataset, there's not any information available to identify the bots that are providing the first time response. As we observed a significant presence of bots in the first time response stage, this raises the importance of extracting additional information and using external tools. The publicly available dataset doesn't come with actual project id numbers. So, after contacting the original authors, we received the actual project id number [2]. By using these, we traced back to the actual repository and extracted additional information using the GitHub v4 API[3]. During our data extraction, we found some pull requests that were deleted from GitHub. We didn't consider them during our next portion of analysis as the number is small.

2) Bot Identification: There are a variety of approaches to identify bots, some of which include recognizing them based on the user name, information about commits, and information about comments [10], [11], [12]. We have used the bot identification method described in literature [7]. They developed the tool called "BoDeGHa" in order to identify bots in a repository by analyzing the comments left on pull requests and issues.

The steps to identify bots are as follows:

- Step 1: run BoDeGHa on the selected 10 projects to collect an initial set of bots.
- Step 2: wrote a Python script to extract all bot profiles and their comments in pull requests of the corresponding project.
- Step 3: manually verified the identified bots by checking the profiles and comments.

While using the BoDeGha, we encountered several wrong predictions. This is why we had to go with manual investigation on the selected projects. As we are focusing on the first time response generation by bots, we sorted the pull request comments according to their dates and checked how many pull requests received their first response from bots. Another important point to mention is that, we encountered some deleted pull requests. You may notice the total number of pull requests are slightly low, as shown in Table III.

3) Pull Request Filtering: After performing the bot identification, we excluded all the first responses generated by bots from our dataset. We are also carefully handled some issues where PR author kept the PR incomplete.

B. Results

The table III represents the first response generated by bots in their respective projects. We find that:

- **Half of the selected projects contain a significant amount of pull requests with bot-generated first response**. We find that the first response in 70.53%,

[3]https://docs.github.com/en/graphql

Table III: Percentage of pull requests having bot-generated first response in selected projects.

Project	Bot(s)	% Bot Generated First Response PR	# PRs
akka/akka	akka-ci	70.43%	7,278
ansible/ansible	ansibot	47.10%	21,825
apache/spark	-	-	-
discourse/discourse	discoursebot	50.40%	6,696
hazelcast/hazelcast	-	-	-
kubernetes/kubernetes	k8s-ci-robot, k8s-github-robot, fejta-bot	21.06%	20,487
nodejs/node	nodejs-github-bot	0.13%	11,225
rails/ rails	rails-bot	11.57%	14,258
saltstack/salt	-	-	-
odoo/odoo	-	-	-

Table IV: Basic statistics of time-to-first-response (in minutes) in selected projects.

Project Name	Mean		Standard Deviation		Median	
	Before fixing first response time	After fixing first response time	Before fixing first response time	After fixing first response time	Before fixing first response time	After fixing first response time
akka/akka	274.50	700.76	2,258.96	2,787.97	26	39
ansible/ansible	9,246.51	15,106.78	53,008.80	65,150.90	9	45
apache/spark	36.76	35.91	810.56	831.04	3	3
discourse/discourse	19.23	37.67	463.15	646.73	0	0
hazelcast/hazelcast	3047.59	3,027.92	21,133.69	22,542.20	56	51
kubernetes/kubernetes	17.93	12.10	320.43	197.77	0	0
nodejs/node	1273.54	1,349.22	8,326.79	8,993.53	60	64
rails/ rails	6194.74	7,656.38	49,908.45	57,276.31	7	16
saltstack/salt	938.77	854.73	4,417.45	4,057.05	44	40
odoo/odoo	51,999.75	63,984.83	174,483.59	197,464.34	1,388	1,553

50.40%, 47.10%, 21.06%, and 11.57% of the pull requests in project *Akka, Discourse, Ansible, Kubernetes, and Rails* are generated by bots, respectively.

- **There could be multiple bots generating the first response in a pull request.** In project Kubernetes, we find three bots that have generated the first responses in pull requests.
- **Automated bot identification tool could generate many false positives.** During our manual examination, we noticed that BoDeGha classifies a few highly involved humans as bots. While checking their user profiles, we found that the wrong classified bots are human, and they are extremely active in their respective projects. This might indicate that they are the core/team members of the project. All the bots identified by the tool BoDeGha for project *Spark, Hazelcast, Salt, Oddo* are humans.

Table IV represents the statistics of time-to-first-response (in minutes) among the selected projects before and after excluding pull requests having bot-generated first response. **We observe that the mean time-to-first-response values in projects Akka, Discourse, and Ansible have increased by 155.28%, 95.89%, and 63.38%, respectively**. The increase for project Rails is much less, i.e., 23.59%. Surprisingly, we observe that the mean time-to-first-response value for project Kubernetes has decreased by 32.52%.

In 5 out of the 10 target projects, a significant percentage (with a maximum of 70.43%) of the first responses are generated by bots. The presence of bots could largely impact the statics and our understanding of time-to-first-response in GitHub pull requests.

VI. RQ3: What are the characteristics of long time-to-first-response PRs?

PR latency is a challenge that slows down pull requests and increases the complexity of the projects. Zhang et al. [2] explained different factors and their influence in pull request latency. However, the impact of various factors on first-time responses to pull requests has not been thoroughly investigated. This research question aims to analyze the characteristics of long time-to-first-response pull requests.

A. Approach

We categorize all pull requests into two categories by their time-to-first-response, i.e., long time-to-first-response PRs which take more than one week to receive their first responses, and other PRs. Next, we perform statistical tests comparing the feature values on two groups of PRs. In the end, we report the performance of automated prediction for time-to-first-response in the selected projects leveraging considered features.

Note that, following our categorization, we observe that projects *akka, apache, discourse and kubernetes* have less than 100 long time-to-first-response PRs. Thus we exclude them in RQ3.

1) Statistical Analysis: We conduct the following two types of statistical tests:

- To test the statistical difference between the features of pull requests taking more than 1 week and pull requests taking less than 1 week to first response, we apply the Mann–Whitney test [13] with a 95% confidence level [14]. We only selected the features that are statistically significant (p-value less than 0.05).
- While statistical significance verifies whether a difference exists between the features of pull requests taking more than 1 week and pull requests taking less than 1 week to first response, we also need to test their practical difference. For this purpose, we use Cliff's delta to estimate their magnitude of difference (i.e., effect size). The value of Cliff's delta (denoted by d) ranges from -1 to $+1$. Negative d means long time-to-first-response PRs' feature values are often smaller than the ones of other PRs, while a positive d implies the opposite.
- Table V represents the significance of different features across the studied projects.

2) Prediction: We consider two prediction scenarios, i.e., predicting the time interval category for each PR (within 1 day, 1 day to 1 week, 1 week to 1 month, more than 1 month) and predicting the time-to-first-response in minutes. The first prediction task is a classification task and the second task is a regression task. We leverage the features listed in Table I and select random forest as the prediction model. We are well aware with the fact that the data might have class imbalance problem. To mitigate the bias, we have used the *'balanced'* class weights in random forest classifier. Additionally we have maintained the train and test set in 80:20 ratio for all the projects.

While designing the regression task (predicting first response time), we have used *random forest regressor* model. We have used *GridSearchCV* in order to get the best parameters for the prediction model. The train and test split were remain same in this prediction task too.

B. Results

Table V represents the significance of different features across the studied projects. The findings are discussed below:

- **Pull requests that take a long time to respond are usually more complex than short PRs.** As shown in Table V, Long time PRs tend to have more descriptive descriptions (3 projects). Also, there are a greater number of added lines of code in the pull requests. However, the results suggests that pull requests having short description tends to have short time to response.
- **Number of participants are small in PRs' that taking long time to response.** Our findings revealed that, among 3/6 projects, the number of participants in pull request comments was lower. Also, all 6 projects tend to have higher lifetime minutes if PRs take a long time to respond, which makes sense. One of the primary causes of the overall lifetime process delay is the delay in first time to respond.
- **Smaller teams have difficulty managing the PR workload.** A small team size is a major contributor to pull request response times. Due to the volume of work in smaller teams, reviewers may not have the time to examine all pull requests, causing undesirable delays in the initial reply.
- **When a project is relatively new, PRs typically take a long time to respond.** In the project dimension, our findings suggest that pull requests take a long time to respond when the projects are newly created.
- As shown in Table V, the number of open issues and the number of open pull requests are showing mixed relationship. Projects with high number of open pull requests and issues (hazelcast, salt) take long time to response.
- **Pull request processing times can be slowed down when there is insufficient communication between the projects and developers.** Our findings suggest that fewer interactions with projects increase the chances of longer response. Additionally, delays arise when two different developers interact with a pull request *same_user*).
- We have used random forest classification to predict the time interval category. We handled the class imbalance issue while experimenting. The results shown in Table VI are promising. Additionally, we have build a prediction model to predict the first time response of a pull request by analyzing its multiple features. The results of regression analysis is shown in Table VII.

Our investigation suggests that pull requests that are complex, descriptive, less communicated take long time to response. Additionally, features from four dimensions revealed several factors behind the long response of pull requests.

VII. Threats to Validity

This work is based on the benchmark data provided by Zhang et al. [8] and our newly collected information. The construction of our dataset relies on the measurement methods of related work and the GitHub v4 API and GHTorrent dataset. Therefore, this paper inherits all the threats of the previous work. At the same time, the paper also has the following limitations:

- We identify bots based on the results returned by the tool BoDeGHa. We observed many false positives in the results. Therefore, it is also possible that we would miss some bots which generate the first response in the considered pull requests. We propose another method to detect users who post highly similar comments in target pull requests and cross-check if they are bots.
- We exclude all pull requests, the first response of which is generated by bots. This may lead to biased results in our statistical analysis and prediction results in RQ2 and RQ3. We plan to recheck those pull requests and propose a new method to identify the real first responses.

Table V: Significance of different features across the studied projects. (+) shows that long time to response PRs' feature value are often greater than short time to first response features. And (-) represents the opposite and (+,-) shows a mixed relationship.

Dimension	Features	Significance	Small	Medium	Large	Negligible
Pull Request	ci_exists	2	(-) 1	-	-	(-) 1
	ci_latency	-	-	-	-	-
	description_length	5	(+) 2	(+) 1	-	(-) 2
	churn_addition	6	(+) 2	(+) 1	-	(+) 3
	files_changed	6	-	(-) 1	-	(+) 5
Review	num_participants	4	(-) 3	-	-	(-) 1
	lifetime_minutes	6	-	-	(+) 6	-
	part_num_code	6	(+) 2	-	-	(-) 4
	merged_or_not	5	(-) 2	-	-	(-) 3
Project	team_size	4	-	(-) 2	(-) 2	-
	project_age	5	(-) 1	(-) 3	(-) 1	-
	open_pr_num	5	-	(-,+) 3	(+) 1	(-) 1
	open_issue_num	5	(-.+) 2	(-) 1	(+) 1	(-) 1
Developer	prior_interaction	4	(-) 2	(-) 1	-	(-) 1
	requester_succ_rate	3	-	-	-	(-) 3
	same_user	3	(-) 2	-	-	(-) 1
	core_member	5	(-) 2	-	-	(-,+) 3
	first_pr	5	(+) 1	-	-	(-,+) 4

Table VI: Performance of Random Forest Classifier while predicting the time interval category.

Project Name	Accuracy	Recall	Precision	F1 Score
ansible/ansible	0.67	0.67	0.64	0.62
nodejs/node	0.90	0.90	0.89	0.88
rails/rails	0.89	0.89	0.86	0.85
hazelcast/hazelcast	0.86	0.86	0.82	0.83
saltstack/salt	0.90	0.90	0.87	0.88
odoo/odoo	0.73	0.73	0.72	0.72

Table VII: Performance of Random Forest Regressor while predicting the first response time in minutes.

Project Name	Mean Absolute Error	Mean Squared Error	Root Mean Square Deviation
ansible/ansible	292.95	135,448.40	368.03
nodejs/node	291.52	188,723.95	434.42
rails/rails	214.45	119,995.87	346.40
hazelcast/hazelcast	164.28	67,909.78	260.60
saltstack/salt	209.32	141,436.86	376.08
odoo/odoo	663.63	910,808.27	954.36

- We have yet to consider the intent of the first response and verify if they are the reviewers. We plan to conduct a manual qualitative study to improve the robustness of the study.
- In RQ3, we only consider six projects because the other four did not have enough pull requests that waited for their first response for more than one week. In the future, we plan to adjust the criteria for determining the long time-to-first-response pull requests within a project.
- In RQ3, the performance of regression task could be improved. We plan to investigate it with more features and use other machine learning algorithms.

VIII. Implications

The findings of this paper have important implications for future research and practice. First of all, our study demonstrates a fine-grained analysis of the pull request latency by considering its stages. Secondly, our study highlights the need to carefully check the bot(s) usage in GitHub pull requests. For a task like first response time analysis, bot-generated first responses should not be considered. Finally, we suggest using tools that can warn people to improve the efficiency of the first response and demonstrate a possible solution.

IX. Conclusion and Future Work

Delays in receiving first response from reviewer can decelerated the overall velocity of a software development. We analyzed a dataset consisting of 3,347,937 closed pull requests and selected ten popular and mature open source GitHub projects in order to gain a better understanding of the fundamental dynamics of the long response time of pull requests. We have identified different characteristics of long time to first response pull requests. Our analysis indicates that pull requests that are complex, descriptive, less communicated take long time to response. In addition, four-dimensional characteristics indicated multiple causes for the slow reaction to pull requests. During our experimentation, we encountered several bot generated responses in pull requests. In this project, we have formulated an approach to identify the bot generated first time response and excluded them before proceeding further. We have conducted statistical and predictive analysis to identify the characteristics of long time to first response pull requests.

As we are aiming to catch upcoming MSR 2023 conference with this project, we have plan to improve project from all aspects. Also, we plan to investigate the false responses generated by the bot identification tool. We aim to add manual qualitative study to improve the robustness of the study.

X. Acknowledgements

I would like to thank my supervisor, Dr. Yuan Tian who consistently guided me since the start of this project. Also, thanks to Dr. Bram Adams for his detailed instructions while formulating this project.

REFERENCES

[1] D. Moreira Soares, M. L. de Lima Júnior, L. Murta, and A. Plastino, "What factors influence the lifetime of pull requests?" *Software: Practice and Experience*, vol. 51, no. 6, pp. 1173–1193, 2021.

[2] X. Zhang, Y. Yu, T. Wang, A. Rastogi, and H. Wang, "Pull request latency explained: An empirical overview," *Empirical Software Engineering*, vol. 27, no. 6, pp. 1–38, 2022.

[3] Y. Yu, H. Wang, V. Filkov, P. Devanbu, and B. Vasilescu, "Wait for it: Determinants of pull request evaluation latency on github," in *2015 IEEE/ACM 12th working conference on mining software repositories*. IEEE, 2015, pp. 367–371.

[4] C. Maddila, S. S. Upadrasta, C. Bansal, N. Nagappan, G. Gousios, and A. van Deursen, "Nudge: accelerating overdue pull requests towards completion," *arXiv preprint arXiv:2011.12468*, 2020.

[5] J. Han, S. Deng, X. Xia, D. Wang, and J. Yin, "Characterization and prediction of popular projects on github," in *2019 IEEE 43rd annual computer software and applications conference (COMPSAC)*, vol. 1. IEEE, 2019, pp. 21–26.

[6] G. Kudrjavets, A. Kumar, N. Nagappan, and A. Rastogi, "Mining code review data to understand waiting times between acceptance and merging: An empirical analysis," *arXiv preprint arXiv:2203.05048*, 2022.

[7] M. Golzadeh, A. Decan, D. Legay, and T. Mens, "A ground-truth dataset and classification model for detecting bots in github issue and pr comments," *Journal of Systems and Software*, vol. 175, may 2021.

[8] X. Zhang, Y. Yu, G. Georgios, and A. Rastogi, "Pull request decisions explained: An empirical overview," *IEEE Transactions on Software Engineering*, 2022.

[9] P. T. Inc. (2015) Collaborative data science. Montreal, QC. [Online]. Available: https://plot.ly

[10] M. Wessel, B. M. De Souza, I. Steinmacher, I. S. Wiese, I. Polato, A. P. Chaves, and M. A. Gerosa, "The power of bots: Characterizing and understanding bots in oss projects," *Proceedings of the ACM on Human-Computer Interaction*, vol. 2, no. CSCW, pp. 1–19, 2018.

[11] T. Dey, S. Mousavi, E. Ponce, T. Fry, B. Vasilescu, A. Filippova, and A. Mockus, "Detecting and characterizing bots that commit code," in *Proceedings of the 17th international conference on mining software repositories*, 2020, pp. 209–219.

[12] N. Cassee, C. Kitsanelis, E. Constantinou, and A. Serebrenik, "Human, bot or both? a study on the capabilities of classification models on mixed accounts," in *2021 IEEE International Conference on Software Maintenance and Evolution (ICSME)*. IEEE, 2021, pp. 654–658.

[13] H. B. Mann and D. R. Whitney, "On a test of whether one of two random variables is stochastically larger than the other," *The annals of mathematical statistics*, pp. 50–60, 1947.

[14] S. Khatoonabadi, D. E. Costa, R. Abdalkareem, and E. Shihab, "On wasted contributions: Understanding the dynamics of contributor-abandoned pull requests," *arXiv preprint arXiv:2110.15447*, 2021.