The Importance of Attributes in Predicting the Lifetime of Human and Automated Pull Requests

Leandro Ferrarezi Valiante[1], **Mairieli Wessel**[2], **Manoel Limeira de Lima Júnior**[1]

[1]Graduate Program in Computer Science – Universidade Federal do Acre (UFAC)
Rio Branco, AC, Brazil

[2]Radboud University - Nijmegen, The Netherlands

leoferrarezi@gmail.com, mairieli.wessel@ru.nl, juniorlimeiras@gmail.com

Abstract. *Pull request(PR)-based workflows improves collaboration in software development; however, the influx of PRs in certain repositories is challenging. Bots, like Dependabot, automate PR creation but can cause communication noise, indicating the need for smarter tools. We investigated* $197,736$ *PRs from* 90 *open-source projects using regression algorithms to predict PR lifetimes. Results showed that in* 21 *repositories, Dependabot PRs were reviewed faster, whereas in 47 repositories, human PRs were quicker. The RMSE difference was notable, with* $18,338$ *minutes for human PRs and* $5,585$ *minutes for Dependabot PRs. Key attributes for prediction were similar across scenarios, although the number of commits was very important only for Dependabot PRs.*

1. Introduction

GitHub is one of the leading software repository hosting sites that facilitates the workflow of developers across multiple software repositories, and allows the Pull Request (PR) contribution method [Chacon and Straub 2014], through which a contributor can submit a PR to be reviewed by the project main developers. In large Open-Source Software (OSS) repositories, the influx of PRs is a major challenge. Ruby on Rails repository gets over 300 PRs monthly [Yu et al. 2015]. Thus, core team developers need automated tools to manage tasks efficiently and without distractions [Mirhosseini and Parnin 2017].

Efforts to automate tedious and repetitive tasks have emerged [Wessel et al. 2018]. Development bots have been introduced into OSS repositories for tasks like dependency updates, fixes, or optimizations. There are about 1,295 bots involved in automated PRs on GitHub [Wyrich et al. 2021]. A manual analysis of 351 projects found that 26.5% utilized at least one bot [Wessel et al. 2018]. Bots can answer frequently asked questions, guide contributors to sign agreements, notify about failed continuous integration tests, and assign reviewers. These initiatives attempt to enhance developers' productivity by saving time and allowing them to focus on improving software quality.

Dependabot is a tool designed to automatically propose updates to external library dependencies, aiming to address vulnerabilities and maintain project stability [Alfadel et al. 2021]. Dependabot monitors the GitHub Advisory Database, and upon identifying a vulnerability, it submits a PR to update the library version and resolve the issue. Despite the advantages of using Dependabot and automating PRs, recent research highlights several challenges in managing PRs generated by bots over time, including the necessity of multiple PR reversions, continuous integration issues, the annoyance caused

by excessive notifications and noise, and other project-related issues that can hinder developer productivity [Mirhosseini and Parnin 2017]. A recent study has also found low acceptance rates and delayed reviews for bot-generated PRs [Wyrich et al. 2021].

In this context, enhancing automation tools to estimate the lifetime of a PR could be beneficial. This capability could help conserve resources, prevent repository overload from excessive activity in a short period, and promote more efficient management of PRs. Towards this aim, we investigates the following three research questions:

RQ1. What is the lifetime difference in PRs created by humans vs. Dependabot?
RQ2. Is there any difference in PR lifetime prediction between human and Dependabot PRs?
RQ3. What are the most important attributes in the PR lifetime prediction in human and Dependabot scenarios?

In this study, we evaluated prediction models for the lifetime of automated PRs generated by Dependabot and also PRs created by humans across 90 software repositories spanning from 2011 to 2021. The SMOReg algorithm using the RBF kernel emerged as the best prediction model for PR lifetime in both scenarios. In the scenario with Dependabot PRs, the improvement over the baseline was approximately 68%, and in the human PRs scenario, this improvement was approximately 40%. Additionally, it was identified that the "workload" and the "number of participants" have greater importance for both scenarios in PRs lifetime prediction. However, the "number of commits" seems very important only in Dependabot scenario (36 victories), while the "requester" is the top 3 attribute in the human scenario, with 48 victories.

2. Methodology

The choice of Dependabot was motivated by the following factors: (i) given that the study aims to predict the lifetime of automated PRs, it was crucial to analyze data from PRs created by bots; (ii) since May 2019, Dependabot has been integrated with GitHub, the platform used to collect data for this study. This integration underscores Dependabot's widespread adoption across repositories; and (iii) the two Dependabot users registered on GitHub (*dependabot[bot]* and *dependabot-preview[bot]*) collectively generate more PRs than all other bots combined [Wyrich et al. 2021]. This dominance further emphasizes Dependabot's significance in the scenario of automated PRs.

To explore repositories proficient in the PR paradigm and Dependabot usage, a query was conducted using the GHTorrent database [Gousios and Zaidman 2014a]. Specifically, data from a dump released in 2021 via Twitter (now "X") that contained repositories hosted on GitHub up to March 6, 2021. The query criteria focused on repositories meeting the following conditions: (i) having a minimum of 500 PRs created either by Dependabot or by humans; (ii) at least 10% of all PRs in the repository were generated by Dependabot; and (iii) the repository had a minimum of 20 contributors.

In total, 101 repositories were selected for data collection. The individual PR data extraction was made by using the "Pullreq Analysis" tool [Gousios and Zaidman 2014b] and the GitHub API. A total of $227,242$ PRs were collected, out of which $43,202$ were generated by Dependabot. The dataset used in this study is available on Zenodo [Ferrarezi 2023].

The final set of 55 attributes collected was then categorized into three high-level groups: **38 attributes characterizing the PR**, **9 attributes of the Project**, and **8 attributes of the Developer**. The complete description of each attribute is available on our replication package [Ferrarezi 2023]. The PR1 attribute (*lifetime_minutes*) serves as the prediction target. This attribute indicates the lifetime of a PR in minutes, representing the time interval between its submission and the decision to integrate or reject it.

In the data mining process, outlier detection is crucial because using outlier instances during the training phase can lead to inaccurate results from predictive models [Ramírez-Gallego et al. 2017]. In this study, the outlier detection method utilized was the Support Vector Machines (SVM) approach [Scikit Learn 2024], which identified and removed 10% of instances from each database identified as outliers by the method.

The experimental process took place in the following sequence: (i) defined the sizes of the training and test windows, ranging from 10% to 40% for training, and 1% to 10% for testing; (ii) calibrated algorithms parameters to generate and evaluate the predictive models using the defined windows, calculating the average RMSE (Root Mean Squared Error) for these models; (iii) evaluated regression algorithms with varied execution configurations on the selected datasets; (iv) generated tables presenting the average RMSE metrics, mean ranking and number of victories for each algorithm; (v) evaluation of the best settings for each algorithm; and (vi) application of the attribute selection strategy to identify the most important attributes in each prediction scenario.

The calibration step aimed to find the optimal configuration for each regression algorithm. A subset of 11 repositories, approximately 10% of the collected data, was selected based on the following criteria: (i) four repositories with minimal involvement of Dependabot; (ii) four repositories with the highest involvement of Dependabot; and (iii) three repositories with the highest number of PRs. These criteria were used to ensure a diverse representation of repository characteristics for effective algorithm calibration.

The method adopted for splitting the training and testing datasets was the "Training-Test Sliding Validation" (TTSV) approach [de Lima Júnior et al. 2018]. This method maintains the chronological order of instances in the dataset by dividing them into training and testing sets. In this approach, each evaluation round typically involves selecting, for example, 10% of instances for training and the immediately subsequent 1% for testing. The method's performance is evaluated by computing the arithmetic mean of the RMSE values obtained from each testing round.

Initially, the regression algorithm was evaluated using 1% of the dataset for testing, while varying the training size from 10% to 40%. After determining that a 40% training window provided optimal results, further testing was conducted by varying the test size from 1% to 10%. The best performance was achieved with a 40% training size and a 1% testing size for both Dependabot and human-generated PR scenarios.

The 11 repositories selected for algorithm calibration have an average of $2,682$ total instances, with 23.59% being PRs created by Dependabot and 68.31% being PRs created by humans. In this dataset, Dependabot PRs have an average lifetime of $3,472$ minutes, while human PRs have an average lifetime of $19,640$ minutes.

In the predictive experiments, the SMOreg, IBk, M5P, and Random Forest (RF) algorithms were evaluated, alongside Linear Regression (LR). These algorithms were

chosen due to their well-established presence in research, proven performance across diverse datasets and scenarios [Lessmann et al. 2008], and wide use in similar prior studies [Gousios et al. 2014, Yu et al. 2015, e Silva and de Lima Júnior 2021]. The parameter variations for each algorithm were conducted as follows: SMOreg with PolyKernel, RBF, and Puk kernels; iBK with number of odd k (neighbors), ranging from 1 to 35; M5P with pruned and unpruned trees; and RF with the number of generated trees ranging from 50 to 1200, with an increment of 50.

To evaluate the prediction results of each configuration, three metrics were used: (i) **RMSE**, which indicates the average error obtained for the predicted values in each test instance; (ii) a ranking metric derived from RMSE results, called **Mean Ranking**, and obtained as follows: for each repository, the RMSE results receive an ordered rank number, being 1 for the lowest error, 2 for the second lowest and so on up to n for the highest calculated error, where n is the number of repositories evaluated, then, the ranks for each configuration are summed, and the result is divided by the number of repositories, where a lower mean, a better performance of the algorithm; and (iii) the third metric is the number of **Victories**, defined by the count of how many times each algorithm achieved the lowest RMSE value, considering all repositories.

The attribute selection process aims to enhance predictive task results and identify the most important attributes. Moreover, selecting a subset of attributes reduces the dataset's dimensionality, thereby improving the performance during the evaluation of predictive models. By executing the *Relief-F* attribute selection, a ranking of attributes was generated for each model tested using the TTSV method. The aforementioned ranking metric was used to highlight the most important attributes for each scenario, considering the average importance value generated by the *Relief-F* strategy for each model.

3. Results and Discussion

The adoption of Dependabot by GitHub repositories over time was verified using PR data from the 101 repositories extracted from GitHub. This adoption started to grow in 2019, especially after Dependabot integration as an official GitHub tool. However, PRs created by humans still constitute the majority (75% on average), even among the repositories with expertise in using Dependabot. Previous research conducted in the context of PR lifetime does not distinguish between automated and human PRs, which can significantly influence tasks such as predicting the PR lifetime.

To answer the RQ1, the lifetime of PRs in both scenarios were compared by calculating the quartiles (Q) of the lifetime. Then the PRs were separated by the limits between Q1 to Q3, representing 50% of the instances. The average lifetime between these limits was considered to perform the lifetime comparison. In this way, it was possible to define that in 21 repositories Dependabot has PRs reviewed in less time, while in 47 repositories human PRs are faster. In 33 repositories there was no significant difference.

The lifetime contrast between human and Dependabot PRs can be highlighted considering the characteristics of two repositories: in the "activeadmin", the lifetime of 50% of the Dependabot PRs is concentrated between 18 minutes and 2.7 hours, while 50% of human PRs have their lifetime concentrated between 3.85 hours and 9.92 days. On the other hand, in the "draft-js", the lifetime of 50% of Dependabot PRs is concentrated between 6.15 days and 2.69 weeks, while the lifetime of 50% of human PRs is

concentrated between 6 hours and 2.51 weeks. These examples underscore the importance of distinguishing between scenarios based on the author of the PR (human or bot), particularly in the case of Dependabot, when studying PRs lifetime. Previous research has not made this distinction, which may lead to less accurate results.

To determine the best configuration for the regression algorithms in each scenario, variations in the parameter values of these algorithms were applied and executed for each model created using the training and test windows, as well as for each dataset collected for this study. The M5P and SMOreg algorithms achieved the best results with the same configuration in both scenarios. On the other hand, the iBK and RF algorithms showed differences in the configuration that yielded the best results. For the scenario with PRs created by humans, iBK performed best with $k = 35$, and RF achieved the best performance with 750 trees. In the scenario with Dependabot PRs, iBK performed best with $k = 5$, and RF achieved the best results with 900 trees.

Having defined the best settings for the four algorithms in each scenario, the next objective was to evaluate these algorithms on the selected calibration datasets. In both scenarios, the SMOreg algorithm with the RBF Kernel achieved the best results on average. In the scenario with Dependabot PRs, the improvement over the baseline (LR algorithm) was approximately 38% (RMSE from 6,039 to 3,754), and in the scenario with human PRs, this improvement was approximately 31% (RMSE from 24,976 to 17,406).

The next experiment aimed to evaluate the algorithms with the best configurations in each model generated using the TTSV technique across the 90 repositories, considering the two scenarios separately. Table 1 displays the average RMSE among the models, mean ranking, and the number of victories for each algorithm in each of the scenarios. It is evident that the SMOreg algorithm with the RBF Kernel achieved the best results in terms of RMSE, mean ranking, and victories in both scenarios. It is worth noting that the RMSE difference between the scenarios is significant, with 18,338 minutes (12.7 days) for PRs created by humans and 5,585 minutes (3.8 days) for PRs from Dependabot, **answering our RQ2**, and highlighting, once again, the importance of distinguishing between both scenarios, which can significantly reduce the error rate of predictive algorithms.

Table 1. Metrics obtained from algorithms evaluation on 90 repositories

	Dependabot Pull Requests					Human-created Pull Requests				
	LR	**M5P Unpruned**	**SMOreg RBF**	**iBK $k = 5$**	**RF 900 trees**	**LR**	**M5P Unpruned**	**SMOreg RBF**	**iBK $k = 35$**	**RF 750 trees**
Average RMSE	17,573	6,246	**5,585**	6,219	6,123	30,415	22,270	**18,338**	20,779	31,444
Mean Ranking	4.90	3.08	**1.43**	2.80	2.78	4.66	3.43	**1.12**	2.44	3.34
Victories	0	13	**64**	3	11	0	9	**81**	0	0

In Table 1, it is possible to observe the improvement in the algorithm results compared to the baseline (LR algorithm). In the scenario with human PRs, there was an improvement of approximately 40% when comparing the RMSE averages between the LR and SMOreg algorithms. In the scenario with Dependabot PRs, comparing the same algorithms, an even more significant improvement of approximately 68% is observed.

Table 2 was compiled to analyze individual repositories in each scenario and assess the improvement in predicting the lifetime of PRs. It lists the 5 lowest and 5 highest RMSE values achieved with the SMOReg algorithm (best overall result) in each scenario.

The table includes the repository names, the RMSE values with the LR algorithm baseline, and the percentage improvement in the RMSE metric for each repository.

Table 2. Best and worst RMSE results among the repositories

Dependabot Pull Requests				Human Pull Requests			
Repository	**LR**	**SMOreg RBF**	**Improvement**	**Repository**	**LR**	**SMOreg RBF**	**Improvement**
open-event-server	641	208	67.55%	mediathread	1,349	1,191	11.71%
jhipster	725	355	51.03%	weblate	3,470	1,335	61.53%
coredns	858	497	42.07%	nuxeo-drive	2,790	2,029	27.28%
openstreetmap-website	1,993	551	72.35%	MozillaAddonsServer	3,006	2,918	2.93%
aws-operator	1,598	581	63.64%	fpl-ccd-config	3,679	2,962	19.49%
...	...	...	...	...	...	...	...
WikiEduDashboard	18,125	16,376	9.65%	hugo	56,734	55,995	1.30%
janusgraph	27,847	17,216	38.18%	alloy	142,075	74,989	47.22%
elide	45,122	17,423	61.39%	openstreetmap-website	143,877	75,563	47.48%
cloudcontroller	26,234	18,406	29.84%	draft-js	108,834	82,201	24.47%
graylog	27406	19,974	27.12%	rubygems.org	13,8698	88,010	36.55%
Average	17,572	5,585	68.21%	**Average**	30,415	18,338	39.70%

In Table 2, the best result for Dependabot PRs in the RMSE metric was achieved by the SMOReg algorithm in the "open-event-server" repository with a value of 208. This means that the algorithm would have an average error of 208 minutes when predicting the lifetime of a new PR created by Dependabot. Additionally, the SMOreg algorithm improved this metric by 67.55% compared to the baseline. For human PRs, a notable improvement of 61.53% was achieved in the "weblate" repository by the SMOreg algorithm, reducing the average error from $3,470$ to $1,335$ minutes compared to the baseline.

Additionally, when analyzing Table 2, it is observed that the "openstreet-map-website" repository appears in both tables, indicating its inclusion in both scenarios. In the Dependabot PRs scenario, the RMSE metric with SMOreg was 551, ranking as the 4_{th} best result. However, in the human PRs scenario, the same repository is among the worst results, ranking third-to-last with a RMSE of $75,563$. **As an answer to our RQ2**, this result underscores the importance of separating PRs for lifetime analysis.

The *Relief-F* attribute selection strategy was employed to detect and investigate the most important attributes for predictive models. It generated rankings of attributes for each model across datasets. A table summarizing the average position in the ranking and the number of victories for each dataset was then created. Table 3 highlights the top ten most important attributes identified in each scenario based on these rankings.

Table 3. Most important attributes for each scenario

Dependabot Pull Requests				Human-created Pull Requests			
ID	**Attribute**	**Mean Ranking**	**Victories**	**ID**	**Attribute**	**Mean Ranking**	**Victories**
PR4	num_commits	3.53	36	P8	workload	4.18	35
P8	workload	5.40	22	PR6	num_participants	5.19	11
PR6	num_participants	6.76	23	PR36	requester	7.83	48
PR34	at_mentions_comments	10.55	3	PR34	at_mentions_comments	8.07	4
PR3	conflict	16.27	9	PR4	num_commits	10.50	1
P9	perc_external_contribs	17.92	0	PR27	src_added	13.45	0
PR37	weekday	18.90	2	PR29	src_churn	14.85	0
PR29	src_churn	19.30	0	PR14	files_added	16.77	0
D1	prev_pullreqs	19.49	0	PR17	files_changed	17.15	0
PR28	src_deleted	19.56	0	P9	perc_external_contribs	18.02	1

In Table 3, most of the attributes identified as significant are common to both scenarios. The "workload" attribute is one of them, and indicates the workload in the repository, i.e., the number of open PRs at the moment a new PR is created. The attribute *num_commits* stands out as more important in the Dependabot scenario compared to human PRs. It was observed that in Dependabot PRs, additional commits beyond the initial one by Dependabot are often made by humans to adjust the proposed solution. This tends to significantly increase the lifetime of the PR.

The attribute "num_participants" obtained 23 victories in the Dependabot PRs scenario and is also among the most important in both scenarios. This aligns with the analysis presented regarding the "num_commits" attribute: when it is necessary for a Dependabot PR to has more than one participant, it indicates that the proposed bot's solution required adjustments, and thus, the PR lifetime tends to increase.

In the scenario with Dependabot PRs, the attribute "weekday" was identified as highly significant. It achieved victories in the "Guttenberg-mobile" and "Spreed" repositories during experiments. In "Guttenberg-mobile", PRs created on Fridays or Saturdays tended to have longer lifetimes compared to other days, influencing the algorithm decision. In the "Spreed" repository, a notable pattern was observed where 92% of Dependabot PRs created on Fridays and Saturdays exhibited similar lifetimes. On Sundays, only 4 PRs were created, but they had longer lifetimes, while Mondays saw only 3 PRs with shorter lifetimes. While these patterns may not apply to all repositories, they suggest that the "weekday" attribute could be important for predictive modeling in specific scenarios.

Answering our RQ3, since 6 out of the top 10 most important attributes for predictive models are common between the two scenarios (PR4, PR6, PR29, PR34, P8 and P9), it can be concluded that the main attributes in predicting the lifetime of PRs are generally common between the scenarios with Dependabot PRs and human PRs. However, the "number of commits" seems very important only in Dependabot scenario (36 victories). On the other hand, the "requester" attribute has 48 victories in human scenario, and has no importance on Dependabot scenario, since the user is always the same. Also, separating automated PRs allowed identifying that characteristics related to the core team's response time may differ in each scenario. The varying attributes employed in the models across different scenarios may provide valuable insights into understanding prediction performance, which warrants further investigation in future studies.

4. Related Work

Predicting lifetime can help reviewers save time, improve efficiency, and optimize PR review prioritization. Researchers have focused on identifying the factors that influence lifetime [Soares et al. 2021, Silva et al. 2020, Gousios et al. 2014, Alfadel et al. 2021, He et al. 2023, Nasrabadi et al. 2023], predicting lifetime [de Lima Júnior et al. 2021, e Silva and de Lima Júnior 2021], and evaluation time [Yu et al. 2015]. Previous research, however, has not considered differences between automated and human PRs.

Recent studies explored the integration (acceptance or rejection) and lifetime of automated PRs [Alfadel et al. 2021, Wyrich et al. 2020, Wyrich et al. 2021, He et al. 2023, Nasrabadi et al. 2023]. [Wyrich et al. 2021] conducted a comparative study of automatically and manually created PRs using a large dataset ($20,623,320$ PRs), but with $4,654$ users identified as bots, even recommending in their conclusion that the

comparison can be done individually for bots. Additionally, the analyzed data corresponds to a short period (January 2019 to May 2020). The conclusions of [Wyrich et al. 2021] point out that automated PRs that are accepted have, on average, a 10 hours lifetime, which is significantly longer than that of humans. However, this does not apply to all projects, as indicated by our study. To the best of our knowledge, there are no scientific publications that analyze the lifetime of automated PRs in individual projects.

In software development, bots are commonly used for repetitive tasks such as bug fixing [Monperrus 2019], dependency updates [Alfadel et al. 2021], defect prediction [Khanan et al. 2020], code refactoring and improvement [Wyrich and Bogner 2019], and more. Studies related to bots (e.g. [Mirhosseini and Parnin 2017]) emphasize annoying behavior such as verbosity, redundancy, excessive actions, unsolicited or unwanted tasks in PRs, generating communication noise, as well as excessive information due to the creation of "new voices" [Monperrus 2019] in developer conversations, which are already burdened with online communication.

5. Limitations

Our study focused on OSS repositories using Dependabot, so the findings may not fully apply to closed-source repositories or other bots. While we do not expect big differences, further research is necessary to investigate the transferability of the results. We have also ensured the collection of a large and diverse sample of repositories to mitigate this limitation. Furthermore, GitHub API allows only 5000 requests per hour [GitHub Docs 2022]. Given that each commit, PR, and comment requires a new request, repositories with significant activity can take several days to complete the extraction process.

6. Conclusion and Future Work

This study demonstrated that human and automated PRs have different lifetime characteristics and should, therefore, be studied and assessed separately, mainly with predictive tasks. We evaluated 5 predictive algorithms (Linear Regression, M5P, SMOReg, iBK, and RandomForest) using the TTSV method with PR data collected from 101 software repositories. The experiments considered two scenarios: models generated with human PRs and models with Dependabot PRs. The SMOreg algorithm with the RBF Kernel achieved the best results, on average. In the scenario with Dependabot PRs, the improvement over the baseline (LR algorithm) was approximately 68%, and in the human PRs scenario, this improvement was approximately 40%. Additionally, 6 of the top 10 most important attributes are the same for Dependabot and human-created PRs, but the number of commits, participants, and the day of the week have greater importance for Dependabot PRs.

Bots that generate PRs can overwhelm repositories by creating multiple requests in quick succession, often before existing ones have been reviewed. Although it is possible to configure the bot to create PRs at specific times, this may delay critical updates, including those that address severe security vulnerabilities. The PR lifetime prediction could prevent the creation of new PRs until pending ones have been adequately reviewed, which would also reduce the computational costs associated with these verification.

Researchers point out that smarter tools should be built to prevent the communication "noise" caused by bots in software repositories. One way to build smarter tools to avoid this noise may be to incorporate the ability to predict the PR lifetime, preventing

overload in software projects. It is believed that the study of factors related to the lifetime of automated PRs and the generation of prediction models can be beneficial in moving toward this goal. Future work can also include in-depth investigations into the influence of each attribute on an automated PR, and other bots beyond Dependabot can be studied and evaluated. Other techniques, such as the extraction of association rules, may bring some new insights.

References

Alfadel, M., Costa, D. E., Shihab, E., and Mkhallalati, M. (2021). On the use of dependabot security pull requests. In *2021 IEEE/ACM 18th International Conference on Mining Software Repositories (MSR)*, pages 254–265.

Chacon, S. and Straub, B. (2014). *Pro git*. Springer Nature.

de Lima Júnior, M. L., Soares, D., Plastino, A., and Murta, L. (2021). Predicting the lifetime of pull requests in open-source projects. *Journal of Software: Evolution and Process*, 33(6):e2337.

de Lima Júnior, M. L., Soares, D. M., Plastino, A., and Murta, L. (2018). Automatic assignment of integrators to pull requests: The importance of selecting appropriate attributes. *J. Syst. Softw.*, 144:181–196.

e Silva, J. M. and de Lima Júnior, M. L. (2021). Prediction of pull requests review time in open source projects. In *Proceedings of the XX Brazilian Symposium on Software Quality*, pages 1–10.

Ferrarezi, L. (2023). Data used for the research 'The Importance of Attributes in Predicting the Lifetime of Human and Automated Pull Requests'. `https://zenodo.org/doi/10.5281/zenodo.8199923`. [Online: Accessed on 07/04/2024].

GitHub Docs (2022). Rate limits for the REST API. `https://docs.github.com/en/rest/using-the-rest-api/rate-limits-for-the-rest-api?apiVersion=2022-11-28#primary-rate-limit-for-authenticated-users`. [Online: Accessed on 07/04/2024].

Gousios, G., Pinzger, M., and Deursen, A. v. (2014). An exploratory study of the pull-based software development model. In *Proceedings of the 36th international conference on software engineering*, pages 345–355.

Gousios, G. and Zaidman, A. (2014a). A dataset for pull-based development research. In *Proceedings of the 11th Working Conference on Mining Software Repositories*, pages 368–371.

Gousios, G. and Zaidman, A. (2014b). Pullreq Analysis. `https://github.com/gousiosg/pullreqs`. [Online: Accessed on 07/04/2024].

He, R., He, H., Zhang, Y., and Zhou, M. (2023). Automating dependency updates in practice: An exploratory study on github dependabot. *IEEE Transactions on Software Engineering*.

Khanan, C., Luewichana, W., Pruktharathikoon, K., Jiarpakdee, J., Tantithamthavorn, C., Choetkiertikul, M., Ragkhitwetsagul, C., and Sunetnanta, T. (2020). Jitbot: an

explainable just-in-time defect prediction bot. In *Proceedings of the 35th IEEE/ACM international conference on automated software engineering*, pages 1336–1339.

Lessmann, S., Baesens, B., Mues, C., and Pietsch, S. (2008). Benchmarking classification models for software defect prediction: A proposed framework and novel findings. *IEEE Transactions on Software Engineering*, 34(4):485–496.

Mirhosseini, S. and Parnin, C. (2017). Can automated pull requests encourage software developers to upgrade out-of-date dependencies? In *2017 32nd IEEE/ACM international conference on automated software engineering (ASE)*, pages 84–94. IEEE.

Monperrus, M. (2019). Explainable software bot contributions: Case study of automated bug fixes. In *2019 IEEE/ACM 1st international workshop on bots in software engineering (BotSE)*, pages 12–15. IEEE.

Nasrabadi, H. M., Agaronian, A. E., Zannone, N., Constantinou, E., and Serebrenik, A. (2023). Investigating the resolution of vulnerable dependencies with dependabot security updates. In *Mining Software Repositories conference*.

Ramírez-Gallego, S., Krawczyk, B., García, S., Woźniak, M., and Herrera, F. (2017). A survey on data preprocessing for data stream mining: Current status and future directions. *Neurocomputing*, 239:39–57.

Scikit Learn (2024). OneClassSVM. `https://scikit-learn.org/stable/modules/generated/sklearn.svm.OneClassSVM.html`. [Online: Accessed on 07/04/2024].

Silva, D. A. N. d., Soares, D. M., and Gonçalves, S. A. (2020). Measuring unique changes: How do distinct changes affect the size and lifetime of pull requests? In *Proceedings of the 14th Brazilian Symposium on Software Components, Architectures, and Reuse*, pages 121–130.

Soares, D. M., de Lima Júnior, M. L., Murta, L., and Plastino, A. (2021). What factors influence the lifetime of pull requests? *Software: Practice and Experience*, 51(6):1173–1193.

Wessel, M., De Souza, B. M., Steinmacher, I., Wiese, I. S., Polato, I., Chaves, A. P., and Gerosa, M. A. (2018). The power of bots: Characterizing and understanding bots in oss projects. *Proceedings of the ACM on Human-Computer Interaction*, 2(CSCW):1–19.

Wyrich, M. and Bogner, J. (2019). Towards an autonomous bot for automatic source code refactoring. In *2019 IEEE/ACM 1st international workshop on bots in software engineering (BotSE)*, pages 24–28. IEEE.

Wyrich, M., Ghit, R., Haller, T., and Müller, C. (2021). Bots don't mind waiting, do they? comparing the interaction with automatically and manually created pull requests. In *2021 IEEE/ACM Third International Workshop on Bots in Software Engineering (BotSE)*, pages 6–10. IEEE.

Wyrich, M., Hebig, R., Wagner, S., and Scandariato, R. (2020). Perception and acceptance of an autonomous refactoring bot. *arXiv preprint arXiv:2001.02553*, 1:303–310.

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