What Attracts Contributors to OSS Projects?

Wenke Yang[1], Yuan Wei[2], Fu Yin[3], and Zhengyi Yang[1,a,*]

[1]School of Computer Science and Engineering, UNSW, Sydney, Australia

[2]Realtek Semiconductor Corp., Singapore

[3]Xilinx, Inc, Cambridge, United Kingdom

[a] zyang@cse.unsw.edu.au

*corresponding author

Keywords: open source, Travistorrent, Pearson correlation, feature selection, data analysis, machine learning

Abstract: Open source software (OSS) has played an essential role in this era. Many companies and researches rely on open source software. However, not all open source projects attract enough developers and get improved regularly. In this paper, to find out factors on contributing to OSS's; we analysed projects with large team size from the TravisTorrent dataset. Machine learning based feature selection and Pearson's correlation analysis are applied to the data and the results show that test density and assert density have strong negative impacts on attracting new contributors. To attract more members in OSS, we suggest that one may pay more attention on production files other than focusing too much on tests. Future researchers may also apply the methodologies used in this paper to explore other factors and/or apply it on different datasets.

1. Introduction

In the last decade, open source software (OSS) has played an essential role in both industry and research. A survey[1] conducted in 2015 revealed that about 78% of companies run on open source. The reason why they use OSS relies on three main factors according to [2]; the dominance of OSS's in their product category, investment amounts on OSS's and the revolutionised working structure.

The choice on OSS's makes it important to attract developers to the project since it solely depends on developers' potential contribution. Therefore, the reasons behind the attraction should be investigated and then, they will then be used to popularise the project and increase the success rate.

While there have been many survey-based studies which looked at the motivations of open-source contributors in general[3], as well as the processes through which new developers join open source projects[4], there are fewer studies published on why developers join or leave a particular project[5, 6], and these tend to be based on psychometric analysis of textual data such as comments close to joining or leaving times rather than directly correlating quantifiable events to joins and leaves.

The question of why developers join OSS projects is widely debated and has received much attention in research. However, much of this attention has been focused on why developers contribute to open source in general, forgoing direct monetary rewards, and not why they join or leave particular software projects in terms of the events rather than psychological characterizations. For example, in Understanding Sustained Participation in Open Source Software Projects the authors quote a contributor to the *phpMyAdmin* project:

"For me it started when I noticed the phpMyAdmin project was `clinically dead': no new release . . . for the last 18 months, no feedback about submitted patches, etc. [phpMyAdmin could] take away all the pain of creating and maintaining a MySQL database"

The authors characterized this reason as “software use value”, being more focused on what the developer said was his motivation (i.e. taking away the pain of maintaining a MySQL database),

rather than on the metrics that were objectively measurable (i.e. no release for last 18 months). This study, in contrast to most of the previous studies, focuses on deducing developer motivations from objectively measurable metrics rather than relying on analysis of developer comments.

It has been found that the main drivers of motivation are enjoyment-based, with intellectual stimulation being much more important than extrinsic rewards such as career advancement[3]. By gaining an understanding of why developers join or leave open source projects, we can obtain insights into the causes of the success of open source projects, and potentially use this information in order to directly influence the success outcome of open source software projects, and possibly software projects in general.

The rest of the paper is organized as follows. Section 2 introduces the necessary background. Section 3 explains the approaches we used to conduct the analysis. The results are shown in Section 4. Finally, section 5 concludes the paper with some directions for the future works.

2. Background

2.1. TravisTorrent

The dataset used in the analysis is from *TravisTorrent*[7]. The GitHub repositories are used for forming *TravisTorrent* and is obtained from 17,313,330 OSS active projects. Non-popular, non-fork and non-toy projects are filtered and excluded for better analysis. Travis users are chosen, and only Java and Ruby language-based projects are included. The dataset used for this project is *'travistorrent_27_10_2016'*.

Each row in the dataset represents unique builds done in the Travis. These rows encapsulate three different data sources; git repository of the specific project, GitHub data of the project and data from Travis API. However, the value in the *'gh_team_size'* field is the number of contributors who have made a commit within the last 3 months. Therefore, a new contributor joining the team will be instantly reflected in an increase in *'gh_team_size'*, but a contributor leaving the team will only cause *'gh_team_size'* to change 3 months after he has stopped contributing. In order to find out the team size at any given moment and to pinpoint the exact time at which contributors join and leave (since the number of contributors joining and leaving could cancel each other out by coincidence), it is necessary to know the author of each commit so that a proper count of joins and leaves can be obtained.

2.2. Related Work

Although some research has been done to analyse the factors that affect the team size in OSS's, there are still different aspects to analyse the correlation between the team size and the unanalysed aspects. Moreover, researchers tried to find the reasons behind the motivations to participate in OSS, there is not a single research based on Travis CI data.

For the same analysis with this paper but with different data and different aspects; first, the intrinsic and extrinsic reasons have been analysed by Wang, He and Chen and they have found out that intrinsic motivations are leading for the individual contributors whereas extrinsic motivations are the reasons for firms participating in OSS[8]. In addition, an analysis on GitHub for understanding the correlation between popularity of OSS projects and some specific factors have been made. The results showed that projects owned by organisations are more likely to attract developers compared to the ones owned by individuals. The same paper also considered the first release date of the software and revealed the correlation between the increase of number of stars after the first release[9].

In addition, the effect of social factors has been investigated for contributions on GitHub repositories. The results indicate that social factors like the social relationship between committer and pull requester are highly correlated[10]. Moreover, another analysis revealed the main functionalities and benefits of GitHub does not affect increasing the factor of attracting software developers for contributing[11].

3. Approach

We have studied the *TravisTorrent* dataset to find out factors motivates people to join a team. However, as mentioned in Section 2, the team size in the dataset does not actually reflect the actual team size precisely. Therefore, we fetched the author information for all commits used from GitHub, to identify those commits made by new developers. We marked the author for a given commit as *'new_joiner'* if he/she never made any commits within the last three months and insert this binary data into the original dataset as a new column named *'is_new_joiner'*.

We will analyse the following seven different projects with highest team size and have sufficient test information on Travis from the dataset: *rails/rails*, *rapid7/metasploit-framework*, *fog/fog*, *geoserver/geoserver*, *ManageIQ/manageiq*, *cloudfoundry/cloud_controller_ng*, and *mitchellh/vagrant*. A brief description of data in these projects is given in TABLE 1. It is worth to note that a single commit may trigger multiple builds, hence, we merged duplicated commits and take their mean for all numerical values. Moreover, we need to shift the *'is_new_joiner'* and map it to its previous commits since people joining the team is a future effect of a certain commit. Thus, we could correctly predict if new contributor would join the project in the future.

Table 1 Brief description of the data.

Project Name	fog	geoserver	manageiq	metasploit	rails	clound_controller_ng	vagrant
Number of Records	8448	2274	13350	53168	430948	9712	2647
Number of Unique Commits	1139	954	1723	1866	2753	2417	2565
Rate of Commits Made by New Joiner(%)	36.0	13.0	4.5	9.7	15.8	17.7	17.9
Max Team Size	89	53	59	65	198	69	56
Min Team Size	7	19	39	31	68	29	3
Mean Team Size	59.4	40.2	51.4	45.4	142	47.5	41.2
Team Size Standard Deviation	22.8	8.91	5.66	9.35	37.1	7.74	14.3

The prediction was considered as a binary classification task since *'is_new_joiner'* is a binary value: for a certain commit, there either is a new contributor or not. As the result, we then applied feature selection and correlation analysis to rank the importance of different features. These techniques are very commonly used in machine learning and statistics to simplify models, shorter training time and reduce over-fitting. From the human perspective, the importance ranking can give OSS teams some suggestions of the aspects they can prioritise when trying to attract more contributors.

Features we do not want to encounter and all low-variance features are first removed before inputting to feature selectors. The low-variance threshold is set to zero in our analysis that filtered out cumulus with all values are identical. Pre-processed data is then fed into the following classifier to rank all features:

1) Extremely randomized trees classifier: Extremely randomized trees[12], denoted as "*ERT*" in this paper, use ensembles of decision trees which can compute the relative importance of each attribute. They are among the most popular machine learning methods for feature ranking thanks to

their good accuracy, robustness and fast training speed.

2) Logistic regression classifier: Logistic regression, denoted as "*LR*" in this paper, is a common method for binary classification. It measures the relationship between the categorical dependent variable and independent variables by estimating probabilities using a logistic function, which is the cumulative logistic distribution. The coefficient of each independent variable indicates the impact, on whether will new contributor join, of the corresponding feature.

Furthermore, we employ *recursive feature elimination (RFE)* to rank features. RFE recursively considering smaller and smaller sets of features until the given number of features are selected. In order to rank all features properly, we make recursive selections with a step of 1 until the best feature is selected. We use Python as our tool of analysis with the help of *Pandas* and *Scikit-learn* libraries. Extremely randomized trees and logistic regression are both used as the estimators in the recursive feature elimination process, and the number of trees in the forest is set to 250 for the extremely randomized trees.

After obtained all these features, for each selection method, the overall ranking for each feature was calculated based on its ranking in each project. Similarly, ranks for each feature in different selection method will be summed together to get final top ranked features.

The *Pearson Correlation Coefficient (PCC)* will then be computed between there top features and the total number of joiners within 3 months after each commit. This will review if the value of a feature has a linear impact on the number of new joiners in long term and discover whether the impact is positive or negative if it exists.

4. Results

We first performed machine learning based feature ranking as described in Section 3 to pick up important factors that affect new joiners. The ranking varies from projects to projects; however, we produce a combined ranking according to the sum of rank in each project. A summary of the top 20 ranked factors using different classifiers is presented in TABLE 2. The detailed description of listed features can be found in [7].

Table 2 Feature ranking.

Project Name	ERT	ERT-RFE	LR	LR-RFE	Overall
gh_test_cases_per_kloc	4	6	17	3	1
gh_test_lines_per_kloc	6	5	2	17	1
gh_src_files	15	12	6	4	3
gh_files_modified	14	13	4	7	4
gh_asserts_cases_per_kloc	7	8	11	13	5
tr_testduration	3	3	14	20	6
tr_setup_time	12	16	10	2	6
gh_commits_on_files_touched	9	7	5	22	8
gh_description_complexity	8	10	1	25	9
gh_team_size	13	14	3	16	10
tr_duration	1	1	16	30	11
gh_src_churn	11	9	9	26	12
tr_tests_skipped	19	18	8	10	12
gh_num_commit_comments	23	23	13	1	14
tr_tests_run	10	11	12	28	15
gh_sloc	5	4	16	29	16
gh_test_churn	16	15	7	27	17
gh_other_files	21	21	18	6	18
gh_num_issue_comments	17	19	22	8	18
tr_ci_latency	2	2	32	32	20

We can notice that the top 5 ranked factors in the overall ranking are:

1) *gh_test_cases_per_kloc* - Test density. Number of test cases per 1,000 executable production source lines of code.

2) *gh_test_lines_per_kloc* - Test density. Number of lines in test cases per 1,000 executable production source lines of code.

3) *gh_src_files* - Number of production files in the new commits in this build.

4) *gh_asserts_cases_per_kloc* - Assert density. Number of assertions per 1,000 executable production source lines of code.

5) *gh_files_modified* - Number of files modified by the new commits in this build.

PCC for these features was then computed to verify our findings and further investigate the relationship. Figure 1 shows the PCC for these features for selected projects. As we can see, test density and assert density have a strong negative correlation with the total number of joiners within 3 months in almost all selected projects. In our PCC calculation, in addition, *p-values* for test density and assert density are extremely close to 0 which further state the significance of our result.

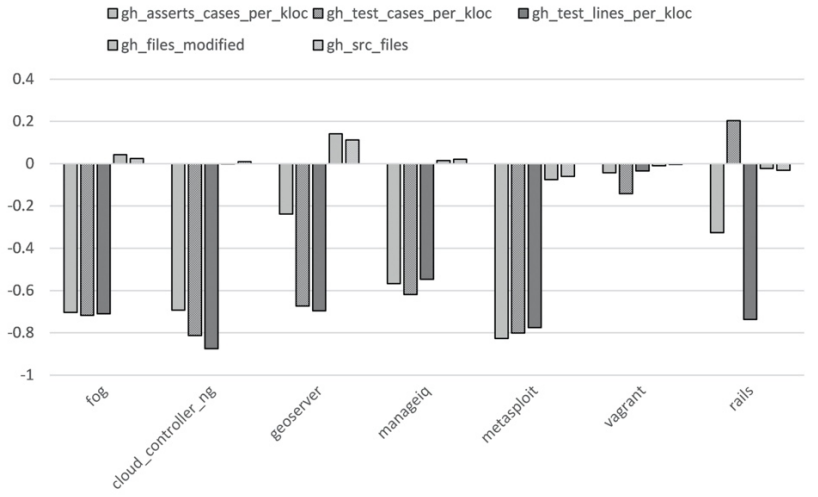

Figure 1 The correlation of factors with joiner rate.

On the other hand, our top features also contain *gh_src_files* and *gh_files_modified*, which shows the number of production files in new commits and number of files modified by new commits. No obvious linear relations have been found in these two features. Intuitively, the strong test negative correlation for test and asserts density may mean reducing the number of test files could attract new developers. However, experiences show test cases normally won't get deleted once they've been created, hence, we believe the reduction of test density actually reflects the increment of production files. Based on all these results, we believe the more active in production files a team is, the more developers it could attract.

5. Conclusion

Although researchers have investigated the factors that affect the team size of OSS, there are, to the best of our knowledge, no research based on the *TravisTorrent* data, a comprehensive dataset with continuous integration testing integrated. In addition, those approaches are considering intuition-based effects. Focusing on the *TravisTorrent* dataset with additional data extracted from GitHub, we applied traditional machine learning technique, feature selection, to explore the possible importance features, for attracting new developers. We also presented correlation analysis between 5 top factors with the change of team size, among a variety of factors presented in the machine learning approach.

The results suggest two possible factors that mainly attracts developers from three features and focuses on one field, test. PCC analysis shows strong negative correlation between test density and asserts density. As a conclusion, we suggest OSS teams focus on production files instead of tests. Besides, OSS teams may apply the methodologies proposed in this paper to discover other important features for their projects.

However, there are limitations for this work. All projects analysed are written in either Ruby or Java, therefore, the result may be biased. In addition, only 7 selected projects on *TravisTorrent* are

explored. In our future works, analysis will be performed on more projects and enlarged datasets to find out the similarity and diversity among projects. Moreover, we will possibly apply more complex models to catch non-linear relations and other influential patterns that related to the number of new joiners in OSS. Finally, our results can be evaluated though time by those developers who will adopt our suggestion.

References

[1] The ninth annual future of open source survey. [Online]. Available: https://www.blackducksoftware.com/2015-future-of-open-source.

[2] J. Lerner and J. Tirole, "Some simple economics of open source," The journal of industrial economics, vol.50, no. 2, pp. 197–234, 2002.

[3] J. Feller, B. Fitzgerald, S. A. Hissam, and K. R. huff, "Why hackers do what they do: Understanding motivation and effort in free/open source software projects," in Perspectives on Free and Open Source Software. MIT Press, 2007, pp. 3–21.

[4] I. Herraiz, G. Robles, J. J. Amor, T. Romera, andJ. M. Gonz´alez Barahona, "The processes of joining in global distributed software projects," in Proceedings of the 2006 International Workshop on Global Software Development for the Practitioner, ser. GSD '06, Shanghai, China: ACM, 2006, pp. 27–33.

[5] P. C. Rigby and A. E. Hassan, "What can oss mailing lists tell us? a preliminary psychometric text analysis of the apache developer mailing list," in Fourth International Workshop on Mining Software Repositories (MSR'07: ICSE Workshops 2007), 2007, pp. 23.

[6] Y. Fang and D. Neufeld, "Understanding sustained participation in open source software projects," J. Manage. Inf. Syst., vol. 25, no. 4, pp. 9–50, Apr. 2009.

[7] M. Beller, G. Gousios, and A. Zaidman, "Travistorrent: Synthesizing travis ci and github for full-stack research on continuous integration," in Proceedings of the 14th working conference on mining software repositories, 2017.

[8] F.-R. Wang, D. He, and J. Chen, "Motivations of individuals and firms participating in open source com-munity," in2005 International Conference on Machine Learning and Cybernetics, vol. 1, 2005, pp. 309–314.

[9] H. Borges, A. Hora, and M. T. Valente, "Understanding the factors that impact the popularity of github repositories," ArXiv preprint arXiv:1606.04984, 2016.

[10] 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, ACM, 2014, pp. 356–366.

[11] J. C. Izquierdo, V. Cosentino, and J. Cabot, "Attracting contributions to your github project, "The Journal of Object Technology, 2015.

[12] P. Geurts, D. Ernst, and L. Wehenkel, "Extremely randomized trees, "Machine Learning, vol. 63, no. 1, pp. 3–42, 2006.

[13] K. Boudreau, "Let a thousand flowers bloom? an early look at large numbers of software app developers and patterns of innovation," Organization Science, 2012.

[14] J. Roberts, I. Hann, and S. Slaughter, "Understanding the motivations, participation, and performance of opensource software developers: A longitudinal study of the apache projects," Management science, 2006.

[15] Y. Ye and K. Kishida, "Toward an understanding of the motivation open source software developers," pp. 419–429, 2003